MPC565 / MPC566

REFERENCE MANUAL

Revised 15 October 2000

© Copyright 2000 MOTOROLA; All Rights Reserved

MPC565 / MPC566

REFERENCE MANUAL

Revised 15 October 2000

© Copyright 2000 MOTOROLA; All Rights Reserved

Paragraph Number

TABLE OF CONTENTS

Page Number

PREFACE

Section 1
OVERVIEW

1.1 Introduction 1-1
1.2 Block Diagram 1-2
1.2.1 Detailed Feature List 1-2
1.2.1.1 MPC565 / MPC566 Optional Features 1-6
1.2.1.2 Supporting Documentation List 1-6
1.2.2 Module or Mode Descriptions 1-7
1.2.2.1 Comparison of MPC565 / MPC566 and MPC555 / MPC556 1-7
1.2.2.2 Additional MPC565 / MPC566 Differences 1-7
1.2.2.3 SRAM Keep-Alive Power Behavior 1-9
1.3 MPC565 / MPC566 Address Map 1-9

Section 2
SIGNAL DESCRIPTIONS

2.1 Pad Function Description 2-1
2.1.1 MPC565 / MPC566 Pin Sharing 2-3
2.2 Pad Module Configuration Register (PDMCR) 2-3
2.3 Pad Module Configuration Register (PDMCR2) 2-5
2.4 Signal Descriptions 2-6
2.4.1 USIU Pads 2-6
2.4.1.1 ADDR[8:31]/SGPIOA[8:31] 2-6
2.4.1.2 DATA[0:31]/SGPIOD[0:31] 2-6
2.4.1.3 $\overline{\text{IRQ[0]}}$/SGPIOC[0] 2-6
2.4.1.4 $\overline{\text{IRQ[1]}}$/$\overline{\text{RSV}}$/SGPIOC[1] 2-6
2.4.1.5 $\overline{\text{IRQ[2]}}$/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$ 2-7
2.4.1.6 $\overline{\text{IRQ[3]}}$/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC[3] 2-7
2.4.1.7 $\overline{\text{IRQ[4]}}$/AT[2]/SGPIOC[4] 2-7
2.4.1.8 $\overline{\text{IRQ[5]}}$/SGPIOC[5]/MODCK[1] 2-8
2.4.1.9 $\overline{\text{IRQ[6:7]}}$/MODCK[2:3] 2-8
2.4.1.10 TSIZ[0:1] 2-8
2.4.1.11 RD/$\overline{\text{WR}}$ 2-8
2.4.1.12 $\overline{\text{BURST}}$ 2-8
2.4.1.13 $\overline{\text{BDIP}}$ 2-9
2.4.1.14 $\overline{\text{TS}}$ 2-9
2.4.1.15 $\overline{\text{TA}}$ 2-9
2.4.1.16 $\overline{\text{TEA}}$ 2-9
2.4.1.17 $\overline{\text{RSTCONF}}$/TEXP 2-9
2.4.1.18 $\overline{\text{OE}}$ 2-10

Paragraph Number		Page Number

2.4.1.19 $\overline{BI}$/$\overline{STS}$ 2-10
2.4.1.20 $\overline{CS[0:3]}$ 2-10
2.4.1.21 $\overline{WE[0:3]}$/$\overline{BE[0:3]}$/AT[0:3] 2-10
2.4.1.22 $\overline{PORESET}$/$\overline{TRST}$ 2-10
2.4.1.23 $\overline{HRESET}$ 2-11
2.4.1.24 $\overline{SRESET}$ 2-11
2.4.1.25 SGPIOC[6]/FRZ/$\overline{PTR}$ 2-11
2.4.1.26 SGPIOC[7]/$\overline{IRQOUT}$/LWP[0] 2-11
2.4.1.27 $\overline{BG}$/VF[0]/LWP[1] 2-12
2.4.1.28 $\overline{BR}$/VF[1]/IWP[2] 2-12
2.4.1.29 $\overline{BB}$/VF[2]/IWP[3] 2-12
2.4.1.30 IWP[0:1]/VFLS[0:1] 2-12
2.4.1.31 TMS 2-13
2.4.1.32 TDI/DSDI 2-13
2.4.1.33 TCK/DSCK 2-13
2.4.1.34 TDO/DSDO 2-13
2.4.1.35 $\overline{JCOMP}$ 2-13
2.4.1.36 XTAL 2-14
2.4.1.37 EXTAL 2-14
2.4.1.38 XFC 2-14
2.4.1.39 CLKOUT 2-14
2.4.1.40 EXTCLK 2-14
2.4.1.41 VDDSYN 2-14
2.4.1.42 VSSSYN 2-14
2.4.1.43 ENGCLK/BUCLK 2-14
2.4.1.44 PULLSEL 2-15
2.4.2 QSMCM_A/QSMCM_B/DLCMD2 (J1850) PADS 2-15
2.4.2.1 PCS[0]/$\overline{SS}$/QGPIO[0] 2-15
2.4.2.2 PCS[1:2]/QGPIO[1:2] 2-16
2.4.2.3 PCS[3]/QGPIO[3]/J1850_TX 2-16
2.4.2.4 MISO/QGPIO[4] 2-16
2.4.2.5 MOSI/QGPIO[5] 2-16
2.4.2.6 SCK/QGPIO[6] 2-16
2.4.2.7 TXD[1:2]/QGPO[1:2] 2-17
2.4.2.8 RXD[1:2]/QGPI[1:2], RXD[1]/QGPI1, RXD[2]/J1850_RX 2-17
2.4.2.9 ECK 2-17
2.4.3 MIOS14 Pads 2-17
2.4.3.1 MDA[11, 13, 27, 30] 2-17
2.4.3.2 MDA[12, 14, 28, 31] 2-18
2.4.3.3 MDA[15], [27:31] 2-18
2.4.3.4 MPWM[0:3] 2-18
2.4.3.5 MPWM[16, 18] 2-18
2.4.3.6 MPWM[17, 19] 2-18

Paragraph Number		Page Number

2.4.3.7 VF[0:2]/MPIO32B[0:2] 2-19
2.4.3.8 VFLS[0:1]/MPIO32B[3:4] 2-19
2.4.3.9 MPWM[4:5]/MPIO32B[5:6] 2-19
2.4.3.10 MDO[7:4]/MPIO32B[7:10] 2-19
2.4.3.11 MPWM[20:21]/MPIO32B[11:12] 2-19
2.4.3.12 C_CNTX0/C_CNRX0/MPIO32B[13:14] 2-20
2.4.3.13 MPIO32B15 2-20
2.4.4 32-KHz Oscillator Pads 2-20
2.4.4.1 XTAL32 2-20
2.4.4.2 EXTAL32 2-20
2.4.4.3 VDDRTC 2-20
2.4.4.4 VSSRTC 2-20
2.4.5 TPU3_A/TPU3_B/TPU3_C Pads 2-20
2.4.5.1 TPUCH[0:15] 2-20
2.4.5.2 T2CLK 2-21
2.4.6 QADC64_A/QADC64_B Pads 2-21
2.4.6.1 ETRIG[1:2] 2-21
2.4.6.2 AN[44]/ANw/PQB[0] 2-21
2.4.6.3 AN[45]/ANx/PQB[1] 2-21
2.4.6.4 AN[46]/ANy/PQB[2] 2-21
2.4.6.5 AN[47]/ANz/PQB[3] 2-22
2.4.6.6 AN[48:51]/PQB[4:7] 2-22
2.4.6.7 AN[52:54]/MA[0:2]/PQA[0:2] 2-22
2.4.6.8 AN[55:59]/PQA[3:7] 2-22
2.4.6.9 AN[64:71]/PQB[0:7] 2-22
2.4.6.10 AN[72:74]/MA[0:2]/PQA[0:2] 2-23
2.4.6.11 AN[75:79]/PQA[3:7] 2-23
2.4.6.12 AN[80:87] 2-23
2.4.6.13 VRH 2-23
2.4.6.14 VRL 2-23
2.4.6.15 ALTREF 2-23
2.4.6.16 VDDA 2-24
2.4.6.17 VSSA 2-24
2.4.7 TOUCAN_A / TOUCAN_B / TOUCAN_C Pads 2-24
2.4.7.1 CNTX0 2-24
2.4.7.2 CNRX0 2-24
2.4.8 READI Pads 2-24
2.4.8.1 $\overline{\text{MSEO}}$ 2-24
2.4.8.2 MDO[0:3] 2-24
2.4.8.3 MCKO 2-25
2.4.8.4 $\overline{\text{RSTI}}$ 2-25
2.4.8.5 $\overline{\text{EVTI}}$ 2-25
2.4.8.6 $\overline{\text{MSEI}}$ 2-25

Paragraph Number		Page Number

2.4.8.7 $\overline{\text{MDI[0:1]}}$ 2-25
2.4.8.8 MCKI. 2-25
2.4.9 UC3F Pads 2-25
2.4.9.1 EPEE 2-25
2.4.9.2 B0EPEE 2-26
2.4.9.3 VFLASH 2-26
2.4.9.4 VDDF 2-26
2.4.9.5 VSSF 2-26
2.4.10 Global Power Supplies 2-26
2.4.10.1 NVDDL 2-26
2.4.10.2 QVDDL 2-26
2.4.10.3 VDDH 2-27
2.4.10.4 VDD 2-27
2.4.10.5 KAPWR 2-27
2.4.10.6 VDDSRAM1 2-27
2.4.10.7 VDDSRAM2 2-27
2.4.10.8 VDDSRAM3 2-27
2.4.10.9 VSS 2-27
2.5 Reset State 2-27
2.5.1 Pin Functionality Configuration Out of Reset. 2-28
2.5.2 Pin State During Reset. 2-28
2.5.3 Power-On Reset and Hard Reset 2-28
2.5.4 Pin Reset States. 2-29
2.6 Pad Types 2-36
2.7 Electrical Characteristics 2-39
2.7.1 Target DC Characteristics 2-39
2.7.2 Pad Measurement Criteria 2-39
2.7.2.1 Input Buffer 2-40
2.7.2.2 Output Driver 2-41
2.8 Package Description 2-53
2.8.1 Package Diagrams 2-53
2.8.2 Bumped Die 2-56
2.8.3 MPC565 / MPC566 Ball Diagram 2-57

Section 3
CENTRAL PROCESSING UNIT

3.1 RCPU Features 3-1
3.2 RCPU Block Diagram. 3-2
3.3 Instruction Sequencer 3-3
3.4 Independent Execution Units 3-4
3.4.1 Branch Processing Unit (BPU) 3-5
3.4.2 Integer Unit (IU) 3-5
3.4.3 Load/Store Unit (LSU) 3-6

Paragraph Number		Page Number

3.4.4 Floating-Point Unit (FPU) 3-6
3.5 Levels of the PowerPC Architecture 3-7
3.6 RCPU Programming Model 3-7
3.7 PowerPC UISA Register Set 3-11
3.7.1 General-Purpose Registers (GPRs) 3-12
3.7.2 Floating-Point Registers (FPRs) 3-12
3.7.3 Floating-Point Status and Control Register (FPSCR) 3-12
3.7.4 Condition Register (CR) 3-15
3.7.4.1 Condition Register CR0 Field Definition 3-16
3.7.4.2 Condition Register CR1 Field Definition 3-16
3.7.4.3 Condition Register CR*n* Field — Compare Instruction 3-17
3.7.5 Integer Exception Register (XER) 3-17
3.7.6 Link Register (LR) 3-18
3.7.7 Count Register (CTR) 3-19
3.8 PowerPC VEA Register Set — Time Base 3-19
3.9 PowerPC OEA Register Set 3-20
3.9.1 Machine State Register (MSR) 3-20
3.9.2 DAE/Source Instruction Service Register (DSISR) 3-22
3.9.3 Data Address Register (DAR) 3-22
3.9.4 Time Base Facility (TB) — OEA 3-23
3.9.5 Decrementer Register (DEC) 3-23
3.9.6 Machine Status Save/Restore Register 0 (SRR0) 3-24
3.9.7 Machine Status Save/Restore Register 1 (SRR1) 3-24
3.9.8 General SPRs (SPRG0–SPRG3) 3-25
3.9.9 Processor Version Register (PVR) 3-25
3.9.10 Implementation-Specific SPRs 3-26
3.9.10.1 EIE, EID, and NRI Special-Purpose Registers 3-26
3.9.10.2 Floating-Point Exception Cause Register (FPECR) 3-26
3.9.10.3 Additional Implementation-Specific Registers 3-27
3.10 Instruction Set 3-28
3.10.1 Instruction Set Summary 3-29
3.10.2 Recommended Simplified Mnemonics 3-33
3.10.3 Calculating Effective Addresses 3-33
3.11 Exception Model 3-34
3.11.1 Exception Classes 3-34
3.11.2 Ordered Exceptions 3-34
3.11.3 Unordered Exceptions 3-35
3.11.4 Precise Exceptions 3-35
3.11.5 Exception Vector Table 3-35
3.12 Instruction Timing 3-36
3.13 PowerPC User Instruction Set Architecture (UISA) 3-38
3.13.1 Computation Modes 3-38
3.13.2 Reserved Fields 3-38

Paragraph Number		Page Number

3.13.3 Classes of Instructions . . . 3-39
3.13.4 Exceptions . . . 3-39
3.13.5 The Branch Processor . . . 3-39
3.13.6 Instruction Fetching . . . 3-39
3.13.7 Branch Instructions . . . 3-39
3.13.7.1 Invalid Branch Instruction Forms . . . 3-39
3.13.7.2 Branch Prediction . . . 3-39
3.13.8 The Fixed-Point Processor . . . 3-40
3.13.8.1 Fixed-Point Instructions . . . 3-40
3.13.9 Floating-Point Processor . . . 3-40
3.13.9.1 General . . . 3-40
3.13.9.2 Optional instructions . . . 3-41
3.13.10 Load/Store Processor . . . 3-41
3.13.10.1 Fixed-Point Load With Update and Store With Update Instructions . . . 3-41
3.13.10.2 Fixed-Point Load and Store Multiple Instructions . . . 3-41
3.13.10.3 Fixed-Point Load String Instructions . . . 3-41
3.13.10.4 Storage Synchronization Instructions . . . 3-41
3.13.10.5 Floating-Point Load and Store With Update Instructions . . . 3-41
3.13.10.6 Floating-Point Load Single Instructions . . . 3-41
3.13.10.7 Floating-Point Store Single Instructions . . . 3-42
3.13.10.8 Optional Instructions . . . 3-42
3.13.10.9 Little-Endian Byte Ordering . . . 3-42
3.14 PowerPC Virtual Environment Architecture (VEA) . . . 3-42
3.14.1 Atomic Update Primitives . . . 3-42
3.14.2 Effect of Operand Placement on Performance . . . 3-42
3.14.3 Storage Control Instructions . . . 3-43
3.14.4 Instruction Synchronize (isync) Instruction . . . 3-43
3.14.4.1 Enforce In-Order Execution of I/O (eieio) Instruction . . . 3-43
3.14.5 Timebase . . . 3-43
3.15 PowerPC Operating Environment Architecture (OEA) . . . 3-43
3.15.1 Branch Processor Registers . . . 3-43
3.15.1.1 Machine State Register (MSR) . . . 3-43
3.15.1.2 Branch Processors Instructions . . . 3-44
3.15.2 Fixed-Point Processor . . . 3-44
3.15.2.1 Special Purpose Registers . . . 3-44
3.15.3 Storage Control Instructions . . . 3-44
3.15.4 Interrupts . . . 3-44
3.15.4.1 System Reset Interrupt . . . 3-44
3.15.4.2 Machine Check Interrupt . . . 3-45
3.15.4.3 Data Storage Interrupt . . . 3-46
3.15.4.4 Instruction Storage Interrupt . . . 3-46
3.15.4.5 Alignment Interrupt . . . 3-46
3.15.4.6 Floating-Point Enabled Exception Type Program Interrupt . . . 3-47

3.15.4.7 Illegal Instruction Type Program Interrupt 3-47
3.15.4.8 Privileged Instruction Type Program interrupt 3-47
3.15.4.9 Floating-Point Unavailable Interrupt 3-47
3.15.4.10 Trace Interrupt 3-47
3.15.4.11 Floating-Point Assist Interrupt 3-48
3.15.4.12 Implementation-Dependent Software Emulation Interrupt 3-49
3.15.4.13 Implementation-Specific Instruction Storage Protection Error Interrupt 3-50
3.15.4.14 Implementation-Specific Data Storage Protection Error Interrupt 3-51
3.15.4.15 Implementation-Specific Debug Interrupts 3-52
3.15.4.16 Partially Executed Instructions 3-53
3.15.5 Timer Facilities 3-54
3.15.6 Optional Facilities and Instructions 3-54

Section 4
BURST BUFFER CONTROLLER MODULE

4.1 Overview and General Description 4-1
4.2 Feature List 4-2
4.2.1 BIU Key Features 4-2
4.2.2 IMPU Key Features 4-3
4.2.3 ICDU Key Features 4-3
4.2.3.1 DECRAM Key Features 4-4
4.3 Class Based Compression Model Main Principles 4-4
4.3.1 Compression Model Features 4-4
4.3.2 Model Limitations 4-5
4.3.3 Instruction Classes Based Compression Algorithm 4-5
4.3.4 Compression Process 4-7
4.3.5 Decompression 4-8
4.3.6 Compression Environment Initialization 4-9
4.4 Operation Modes 4-9
4.4.1 Instruction Fetch 4-9
4.4.1.1 “Decompression OFF” Mode 4-10
4.4.1.2 “Decompression ON” Mode 4-10
4.4.1.3 Show Cycles in “Decompression ON” Mode 4-10
4.4.2 Burst Operation of the BBC 4-11
4.4.3 Access Violation Detection 4-11
4.4.4 Slave Operation 4-12
4.4.5 Reset Behavior 4-12
4.4.6 Debug Operation Mode 4-13
4.5 Exception Table Relocation (ETR) 4-13
4.5.1 ETR Overview 4-13
4.5.2 ETR Operation 4-14
4.5.3 Enhanced External Interrupt Relocation (EEIR) 4-17
4.6 Decompressor RAM (DECRAM) Functionality 4-18

4.6.1 Vocabulary Table Storage Operation 4-19
4.6.2 General Purpose Memory Operation 4-19
4.6.2.1 Memory Protection Violations 4-20
4.6.2.2 DECRAM StandBy Operation Mode 4-20
4.7 Branch Target Buffer 4-21
4.7.1 BTB Operation 4-21
4.7.1.1 BTB Invalidation 4-22
4.7.1.2 BTB Enabling/Disabling 4-23
4.7.1.3 BTB Inhibit Regions 4-23
4.8 BBC Programming Model 4-23
4.8.1 Address Map 4-23
4.8.1.1 BBC Special Purpose Registers (SPRs) 4-24
4.8.1.2 DECRAM and DCCR Block 4-24
4.8.2 BBC Registers Description 4-25
4.8.2.1 BBC Module Configuration Register BBCMCR 4-25
4.8.2.2 Region Base Address Registers MI_RBA0-3 4-27
4.8.2.3 Region Attribute Registers MI_RA[1:3] 4-27
4.8.2.4 Global Region Attribute Register MI_GRA 4-29
4.8.2.5 External Interrupt Relocation Table Base Address Register — EIBADR 4-31
4.8.2.6 Decompressor Class Configuration Registers (DCCR1-15) 4-31
4.8.2.7 Decompressor Class Configuration Registers (DCCR1-DCCR15) 4-32

Section 5
UNIFIED SYSTEM INTERFACE UNIT (USIU)

5.1 USIU Introduction 5-1
5.2 USIU Module Overview 5-1
5.2.1 USIU Address Map 5-3
5.2.2 USIU Special Purpose Registers 5-7

Section 6
SYSTEM CONFIGURATION AND PROTECTION

6.1 System Configuration 6-3
6.1.1 USIU Pins Multiplexing 6-4
6.1.2 Memory Mapping 6-4
6.1.3 Arbitration Support 6-5
6.2 External Master Modes 6-5
6.2.1 Operation in External Master Modes 6-6
6.2.2 Address Decoding for External Accesses 6-7
6.3 USIU General-Purpose I/O 6-7
6.4 Enhanced Interrupt Controller 6-9
6.4.1 Key Features 6-9
6.4.2 Interrupt Configuration 6-9
6.4.3 Regular Interrupt Controller Operation (MPC555/MPC556 Compatible Mode) 6-11
6.4.4 Enhanced Interrupt Controller 6-12

Paragraph Number		Page Number

6.4.4.1 General Operation 6-12
6.4.4.2 Lower Priority Request Masking 6-14
6.4.4.3 Backward Compatibility with MPC555/MPC556 6-15
6.5 Interrupt Overhead Estimation for Enhanced Interrupt Controller Mode 6-17
6.6 Hardware Bus Monitor 6-18
6.7 MPC565 / MPC566 Decrementer 6-19
6.8 MPC565 / MPC566 Time Base (TB) 6-20
6.9 Real-Time Clock (RTC) 6-21
6.10 Periodic Interrupt Timer (PIT) 6-21
6.11 Software Watchdog Timer (SWT) 6-22
6.12 Freeze Operation 6-24
6.13 Low Power Stop Operation 6-24
6.14 System Configuration and Protection Registers 6-24
6.14.1 System Configuration Registers 6-25
6.14.1.1 SIU Module Configuration Register (SIUMCR) 6-25
6.14.1.2 Internal Memory Map Register 6-28
6.14.1.3 External Master Control Register (EMCR) 6-29
6.14.2 SIU Interrupt Controller Registers 6-30
6.14.2.1 SIU Interrupt Pending Register 6-31
6.14.2.2 SIU Interrupt Pending Register 2 6-31
6.14.2.3 SIU Interrupt Pending Register 3 6-32
6.14.2.4 SIU Interrupt Mask Register 6-32
6.14.2.5 SIU Interrupt Mask Register 2 6-33
6.14.2.6 SIU Interrupt Mask Register 3 6-33
6.14.2.7 SIU Interrupt Edge Level Register (SIEL) 6-33
6.14.2.8 SIU Interrupt Vector Register 6-34
6.14.2.9 Interrupt In-Service Registers 6-35
6.14.3 System Protection Registers 6-36
6.14.3.1 System Protection Control Register (SYPCR) 6-36
6.14.3.2 Software Service Register (SWSR) 6-37
6.14.3.3 Transfer Error Status Register (TESR) 6-38
6.14.4 System Timer Registers 6-39
6.14.4.1 Decrementer Register 6-39
6.14.4.2 Time Base SPRs 6-39
6.14.4.3 Time Base Reference Registers 6-40
6.14.4.4 Time Base Control and Status Register 6-40
6.14.4.5 Real-Time Clock Status and Control Register 6-41
6.14.4.6 Real-Time Clock Register (RTC) 6-42
6.14.4.7 Real-Time Clock Alarm Register (RTCAL) 6-42
6.14.4.8 Periodic Interrupt Status and Control Register (PISCR) 6-43
6.14.4.9 Periodic Interrupt Timer Count Register (PITC) 6-43
6.14.4.10 Periodic Interrupt Timer Register (PITR) 6-44
6.14.5 General-Purpose I/O Registers 6-45

6.14.5.1 SGPIO Data Register 1 (SGPIODT1) 6-45
6.14.5.2 SGPIO Data Register 2 (SGPIODT2) 6-45
6.14.5.3 SGPIO Control Register (SGPIOCR) 6-46

Section 7
RESET

7.1 Reset Operation 7-1
7.1.1 Power-On Reset 7-1
7.1.2 Hard Reset 7-2
7.1.3 Soft Reset 7-2
7.1.4 Loss of Lock 7-3
7.1.5 On-Chip Clock Switch 7-3
7.1.6 Software Watchdog Reset 7-3
7.1.7 Checkstop Reset 7-3
7.1.8 Debug Port Hard Reset 7-3
7.1.9 Debug Port Soft Reset 7-3
7.1.10 JTAG Reset 7-3
7.2 Reset Actions Summary 7-4
7.3 Data Coherency During Reset 7-4
7.4 Reset Status Register 7-5
7.5 Reset Configuration 7-6
7.5.1 Hard Reset Configuration 7-6
7.5.2 Hard Reset Configuration Word 7-11
7.5.3 Soft Reset Configuration 7-13

Section 8
CLOCKS AND POWER CONTROL

8.1 Overview 8-1
8.2 System Clock Sources 8-3
8.3 System PLL 8-3
8.3.1 Frequency Multiplication 8-4
8.3.2 Skew Elimination 8-4
8.3.3 Pre-Divider 8-4
8.3.4 PLL Block Diagram 8-4
8.3.5 PLL Pins 8-5
8.4 System Clock During PLL Loss of Lock 8-6
8.5 Low-Power Divider 8-6
8.6 MPC565 / MPC566 Internal Clock Signals 8-7
8.6.1 General System Clocks 8-9
8.6.2 Clock Out (CLKOUT) 8-12
8.6.3 Engineering Clock (ENGCLK) 8-12
8.7 Clock Source Switching 8-13
8.8 Low-Power Modes 8-15
8.8.1 Entering a Low-Power Mode 8-15

8.8.2 Power Mode Descriptions 8-16
8.8.3 Exiting from Low-Power Modes 8-16
8.8.3.1 Exiting from Normal-Low Mode 8-17
8.8.3.2 Exiting From Doze Mode 8-17
8.8.3.3 Exiting From Deep-Sleep Mode 8-18
8.8.3.4 Exiting from Power-Down Mode 8-18
8.8.3.5 Low-Power Modes Flow 8-18
8.9 Basic Power Structure 8-20
8.9.1 General Power Supply Definitions 8-20
8.9.2 Chip Power Structure 8-21
8.9.2.1 NVDDL 8-21
8.9.2.2 QVDDL 8-21
8.9.2.3 VDD 8-21
8.9.2.4 VDDSYN, VSSSYN 8-21
8.9.2.5 KAPWR 8-21
8.9.2.6 VDDA, VSSA 8-22
8.9.2.7 VFLASH 8-22
8.9.2.8 VDDF, VSSF 8-22
8.9.2.9 VDDH 8-22
8.9.2.10 VDDSRAM1 8-22
8.9.2.11 VDDSRAM2 8-22
8.9.2.12 VDDSRAM3 8-22
8.9.2.13 VDDRTC 8-22
8.9.2.14 VSS 8-22
8.9.3 Keep Alive Power 8-23
8.9.3.1 Keep Alive Power Configuration 8-23
8.9.3.2 Keep-Alive Power Registers Lock Mechanism 8-24
8.10 VDDSRAM Supply Failure Detection 8-26
8.11 Power-Up/Down Sequencing 8-26
8.12 Clocks Unit Programming Model 8-29
8.12.1 System Clock Control Register (SCCR) 8-29
8.12.2 PLL, Low-Power, and Reset-Control Register (PLPRCR) 8-33
8.12.3 Change of Lock Interrupt Register (COLIR) 8-36
8.12.4 VDDSRAM Control Register (VSRMCR) 8-36

Section 9
EXTERNAL BUS INTERFACE

9.1 Features 9-1
9.2 Bus Transfer Signals 9-1
9.3 Bus Control Signals 9-2
9.4 Bus Interface Signal Descriptions 9-3
9.5 Bus Operations 9-7
9.5.1 Basic Transfer Protocol 9-8

9.5.2 Single Beat Transfer 9-8
9.5.2.1 Single Beat Read Flow 9-8
9.5.2.2 Single Beat Write Flow 9-11
9.5.2.3 Single Beat Flow with Small Port Size 9-14
9.5.3 Burst Transfer 9-15
9.5.4 Burst Mechanism 9-16
9.5.5 Alignment and Packaging of Transfers 9-28
9.5.6 Arbitration Phase 9-30
9.5.6.1 Bus Request 9-31
9.5.6.2 Bus Grant 9-32
9.5.6.3 Bus Busy 9-32
9.5.6.4 Internal Bus Arbiter 9-33
9.5.7 Address Transfer Phase Signals 9-35
9.5.7.1 Transfer Start 9-36
9.5.7.2 Address Bus 9-36
9.5.7.3 Read/Write 9-36
9.5.7.4 Burst Indicator 9-36
9.5.7.5 Transfer Size 9-37
9.5.7.6 Address Types 9-37
9.5.7.7 Burst Data in Progress 9-39
9.5.8 Termination Signals 9-39
9.5.8.1 Transfer Acknowledge 9-39
9.5.8.2 Burst Inhibit 9-39
9.5.8.3 Transfer Error Acknowledge 9-39
9.5.8.4 Termination Signals Protocol 9-39
9.5.9 Storage Reservation 9-41
9.5.10 Bus Exception Control Cycles 9-44
9.5.10.1 Retrying a Bus Cycle 9-44
9.5.10.2 Termination Signals Protocol Summary 9-48
9.5.11 Bus Operation in External Master Modes 9-48
9.5.12 Contention Resolution on External Bus 9-53
9.5.13 Show Cycle Transactions 9-55

Section 10
MEMORY CONTROLLER

10.1 Overview 10-1
10.2 Memory Controller Architecture 10-3
10.2.1 Associated Registers 10-4
10.2.2 Port Size Configuration 10-5
10.2.3 Write-Protect Configuration 10-5
10.2.4 Address and Address Space Checking 10-5
10.2.5 Burst Support 10-5
10.3 Chip-Select Timing 10-6

Paragraph Number | Page Number

10.3.1 Memory Devices Interface Example 10-7
10.3.2 Peripheral Devices Interface Example 10-9
10.3.3 Relaxed Timing Examples 10-11
10.3.4 Extended Hold Time on Read Accesses 10-15
10.3.5 Summary of GPCM Timing Options 10-19
10.4 Global (Boot) Chip-Select Operation 10-21
10.5 Write and Byte Enable Signals. 10-22
10.6 Dual Mapping of the Internal Flash EEPROM Array 10-23
10.7 Memory Controller External Master Support 10-25
10.8 Programming Model. 10-28
10.8.1 General Memory Controller Programming Notes 10-28
10.8.2 Memory Controller Status Registers (MSTAT). 10-29
10.8.3 Memory Controller Base Registers (BR[0] – BR[3]) 10-29
10.8.4 Memory Controller Option Registers (OR[0]– OR[3]) 10-31
10.8.5 Dual-Mapping Base Register (DMBR) 10-33
10.8.6 Dual-Mapping Option Register 10-34

Section 11
L-BUS TO U-BUS INTERFACE (L2U)

11.1 General Features 11-1
11.2 DMPU Features 11-1
11.3 L2U Block Diagram 11-2
11.4 Modes Of Operation. 11-2
11.4.1 Normal Mode 11-3
11.4.2 Reset Operation 11-3
11.4.3 Factory Test Mode 11-3
11.4.4 Peripheral Mode 11-4
11.5 Data Memory Protection 11-4
11.5.1 Functional Description 11-4
11.5.2 Associated Registers 11-5
11.5.3 L-bus Memory Access Violations 11-7
11.6 Reservation Support 11-7
11.6.1 The Reservation Protocol 11-7
11.6.2 L2U Reservation Support 11-8
11.6.3 Reserved Location (Bus) and Possible Actions 11-9
11.7 L-Bus Show Cycle Support 11-9
11.7.1 Programming Show Cycles 11-9
11.7.2 Performance Impact 11-10
11.7.3 Show Cycle Protocol 11-10
11.7.4 L-Bus Write Show Cycle Flow 11-10
11.7.5 L-Bus Read Show Cycle Flow 11-11
11.7.6 Show Cycle Support Guidelines 11-11
11.8 L2U Programming Model 11-12

11.8.1 U-bus Access 11-13
11.8.2 Transaction Size 11-13
11.8.3 L2U Module Configuration Register (L2U_MCR) 11-13
11.8.4 Region Base Address Registers (L2U_RBAx) 11-14
11.8.5 Region Attribute Registers (L2U_RAx) 11-15
11.8.6 Global Region Attribute Register 11-16

Section 12
U-BUS TO IMB3 BUS INTERFACE (UIMB)

12.1 Features 12-1
12.2 UIMB Block Diagram 12-2
12.3 Clock Module 12-2
12.4 Interrupt Operation 12-3
12.4.1 Interrupt Sources and Levels on IMB 12-4
12.4.2 IMB Interrupt Multiplexing 12-4
12.4.3 ILBS Sequencing 12-4
12.4.4 Interrupt Synchronizer 12-6
12.5 Programming Model 12-7
12.5.1 UIMB Module Configuration Register (UMCR) 12-7
12.5.2 Test control register (UTSTCREG) 12-8
12.5.3 Pending Interrupt Request Register (UIPEND) 12-8

Section 13
QUEUED ANALOG-TO-DIGITAL CONVERTER (QADC64E)

13.1 Features, Overview and Quick Reference Diagrams 13-1
13.1.1 Features of the QADC64E (Each Module) 13-1
13.1.2 QADC64E Block Diagrams 13-2
13.1.3 Memory Map 13-4
13.1.4 Using the Queue and Result Word Table 13-6
13.1.5 External Multiplexing 13-7
13.2 Programming the QADC64E Registers 13-9
13.2.1 QADC64E Module Configuration Register 13-10
13.2.1.1 Low Power Stop Mode 13-10
13.2.1.2 Freeze Mode 13-11
13.2.1.3 Supervisor/Unrestricted Address Space 13-12
13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks 13-13
13.2.2 QADC64E Interrupt Register 13-13
13.2.3 Port Data Register 13-15
13.2.4 Port Data Direction Register 13-15
13.2.5 Control Register 0 13-16
13.2.6 Control Register 1 13-18
13.2.7 Control Register 2 13-20

Paragraph Number		Page Number

13.2.8 Status Registers . . . 13-23
13.2.9 Conversion Command Word Table . . . 13-30
13.2.10 Result Word Table . . . 13-36
13.3 Analog Subsystem . . . 13-38
13.3.1 Analog-to-Digital Converter Operation . . . 13-38
13.3.1.1 Conversion Cycle Times . . . 13-39
13.3.2 Channel Decode and Multiplexer . . . 13-40
13.3.3 Sample Buffer Amplifier . . . 13-40
13.3.4 Digital to Analog Converter (DAC) Array . . . 13-40
13.3.5 Comparator . . . 13-41
13.3.6 Bias . . . 13-41
13.3.7 Successive Approximation Register . . . 13-41
13.3.8 State Machine . . . 13-41
13.4 Digital Subsystem . . . 13-41
13.4.1 Queue Priority . . . 13-42
13.4.2 Sub-Queues That are Paused . . . 13-42
13.4.3 Boundary Conditions . . . 13-44
13.4.4 Scan Modes . . . 13-45
13.4.4.1 Disabled Mode . . . 13-45
13.4.4.2 Reserved Mode . . . 13-46
13.4.4.3 Single-Scan Modes . . . 13-46
13.4.4.4 Software Initiated Single-Scan Mode . . . 13-47
13.4.4.5 External Trigger Single-Scan Mode . . . 13-47
13.4.4.6 External Gated Single-Scan Mode . . . 13-48
13.4.4.7 Periodic/Interval Timer Single-Scan Mode . . . 13-48
13.4.4.8 Continuous-Scan Modes . . . 13-49
13.4.4.9 Software Initiated Continuous-Scan Mode . . . 13-50
13.4.4.10 External Trigger Continuous-Scan Mode . . . 13-50
13.4.4.11 External Gated Continuous-Scan Mode . . . 13-51
13.4.4.12 Periodic/Interval Timer Continuous-Scan Mode . . . 13-52
13.4.5 QADC64E Clock (QCLK) Generation . . . 13-52
13.4.6 Periodic/Interval Timer . . . 13-54
13.4.7 Configuration And Control using the IMB Interface . . . 13-55
13.4.7.1 QADC64E Bus Interface Unit . . . 13-55
13.4.7.2 QADC64E Bus Accessing . . . 13-55
13.5 Trigger and Queue Interaction Examples . . . 13-57
13.5.1 Queue Priority Schemes . . . 13-57
13.5.2 Conversion Timing Schemes . . . 13-68
13.6 QADC64E Integration Requirements . . . 13-71
13.6.1 Port Digital Input/Output Pins . . . 13-71
13.6.2 External Trigger Input Pins . . . 13-72
13.6.3 Analog Power Pins . . . 13-72
13.6.3.1 Analog Supply Filtering and Grounding . . . 13-74

13.6.4 Analog Reference Pins 13-76
13.6.5 Analog Input Pins 13-76
13.6.5.1 Analog Input Considerations 13-78
13.6.5.2 Settling Time for the External Circuit 13-80
13.6.5.3 Error Resulting from Leakage. 13-80
13.6.5.4 Accommodating Positive/Negative Stress Conditions 13-81

Section 14
QUEUED SERIAL MULTI-CHANNEL MODULE

14.1 Overview 14-1
14.2 Block Diagram 14-1
14.2.1 MPC565 / MPC566 QSMCM Details 14-2
14.3 Signal Descriptions 14-3
14.4 Memory Maps 14-3
14.5 QSMCM Global Registers 14-6
14.5.1 Low-Power Stop Operation 14-6
14.5.2 Freeze Operation 14-7
14.5.3 Access Protection. 14-7
14.5.4 QSMCM Interrupts 14-7
14.5.5 QSMCM Configuration Register (QSMCMMCR) 14-9
14.5.6 QSMCM Test Register (QTEST) 14-9
14.5.7 QSMCM Interrupt Level Registers (QDSCI_IL, QSPI_IL) 14-9
14.6 QSMCM Pin Control Registers 14-10
14.6.1 Port QS Data Register (PORTQS) 14-11
14.6.2 PORTQS Pin Assignment Register (PQSPAR) 14-12
14.6.3 PORTQS Data Direction Register (DDRQS) 14-13
14.7 Queued Serial Peripheral Interface 14-14
14.7.1 QSPI Registers 14-16
14.7.1.1 QSPI Control Register 0 14-17
14.7.1.2 QSPI Control Register 1 14-19
14.7.1.3 QSPI Control Register 2 14-20
14.7.1.4 QSPI Control Register 3 14-20
14.7.1.5 QSPI Status Register 14-21
14.7.2 QSPI RAM 14-22
14.7.2.1 Receive RAM 14-23
14.7.2.2 Transmit RAM. 14-23
14.7.2.3 Command RAM 14-23
14.7.3 QSPI Pins 14-24
14.7.4 QSPI Operation 14-25
14.7.4.1 Enabling, Disabling, and Halting the SPI 14-26
14.7.4.2 QSPI Interrupts. 14-27
14.7.4.3 QSPI Flow 14-27
14.7.5 Master Mode Operation 14-34

Paragraph Number		Page Number

14.7.5.1 Clock Phase and Polarity 14-35
14.7.5.2 Baud Rate Selection 14-35
14.7.5.3 Delay Before Transfer 14-36
14.7.5.4 Delay After Transfer 14-36
14.7.5.5 Transfer Length 14-37
14.7.5.6 Peripheral Chip Selects 14-37
14.7.5.7 Master Wraparound Mode 14-38
14.7.6 Slave Mode 14-38
14.7.6.1 Description of Slave Operation 14-39
14.7.7 Slave Wraparound Mode 14-41
14.7.8 Mode Fault 14-42
14.8 Serial Communication Interface 14-42
14.8.1 SCI Registers 14-45
14.8.2 SCI Control Register 0 14-46
14.8.3 SCI Control Register 1 14-46
14.8.4 SCI Status Register (SCxSR) 14-48
14.8.5 SCI Data Register (SCxDR) 14-50
14.8.6 SCI Pins 14-51
14.8.7 SCI Operation 14-51
14.8.7.1 Definition of Terms 14-51
14.8.7.2 Serial Formats 14-52
14.8.7.3 Baud Clock 14-52
14.8.7.4 Parity Checking 14-53
14.8.7.5 Transmitter Operation 14-53
14.8.7.6 Receiver Operation 14-55
14.8.7.7 Receiver Functional Operation 14-57
14.8.7.8 Idle-Line Detection 14-58
14.8.7.9 Receiver Wake-Up 14-59
14.8.7.10 Internal Loop Mode 14-59
14.9 SCI Queue Operation 14-60
14.9.1 Queue Operation of SCI1 for Transmit and Receive 14-60
14.9.2 Queued SCI1 Status and Control Registers 14-60
14.9.2.1 QSCI1 Control Register 14-60
14.9.2.2 QSCI1 Status Register 14-62
14.9.3 QSCI1 Transmitter Block Diagram 14-62
14.9.4 QSCI1 Additional Transmit Operation Features 14-63
14.9.5 QSCI1 Transmit Flow Chart Implementing the Queue 14-65
14.9.6 Example QSCI1 Transmit for 17 Data Bytes 14-67
14.9.7 Example SCI Transmit for 25 Data Bytes 14-68
14.9.8 QSCI1 Receiver Block Diagram 14-69
14.9.9 QSCI1 Additional Receive Operation Features 14-69
14.9.10 QSCI1 Receive Flow Chart Implementing The Queue 14-72
14.9.11 QSCI1 Receive Queue Software Flow Chart 14-73

14.9.12 Example QSCI1 Receive Operation of 17 Data Frames 14-74

Section 15
DATA LINK CONTROLLER MODULE (DLCMD2)

15.1 Scope 15-1
15.2 Features 15-1
15.3 1.4 Background 15-2
15.4 1.5 Applicable Documents 15-2
15.5 1.6 General Requirements 15-2
15.6 Logic Description 15-2
15.6.1 Block Diagram 15-2
15.6.2 DLCMD2 Operation 15-3
15.6.2.1 General 15-3
15.6.2.2 Logic Section Description and Relation to Transceiver 15-4
15.6.2.3 DLCMD2 Transmit/Receive Operation 15-4
15.6.2.4 Message Transmission 15-6
15.6.2.5 Message Reception 15-6
15.6.2.6 Sleep Mode 15-7
15.6.2.7 Debug Mode 15-7
15.6.2.8 4X Speed Mode 15-7
15.6.2.9 Block Mode 15-8
15.6.2.10 Error Detection 15-8
15.6.2.11 Arbitration 15-9
15.6.2.12 Timebase Generation 15-9
15.6.2.13 Receive and Transmit Message Buffers 15-10
15.6.2.14 Bus Waveforms Generation 15-11
15.6.2.15 Huntzicker Encoding 15-11
15.6.3 TData Link Controller Module (DLCMD2) 15-13
15.7 Signals Overview 15-13
15.7.1 J1850 Bus Waveforms 15-14
15.7.1.1 Start of Frame (SOF) 15-14
15.7.1.2 Data Bits 15-14
15.7.1.3 “0” bit 15-14
15.7.1.4 “1” bit 15-15
15.7.1.5 End of Data (EOD) 15-15
15.7.1.6 Normalization Bit 15-15
15.7.1.7 End of Frame (EOF) 15-15
15.7.1.8 Break 15-16
15.7.2 General Symbol Transmission 15-16
15.7.3 General Symbol Reception 15-16
15.7.4 Support For External Transceiver 15-17
15.8 Operating Modes 15-17
15.8.1 Power Off 15-18

Paragraph Number		Page Number

15.8.2 Reset 15-18
15.8.3 Run 15-19
15.8.4 DLCMD2 STOP and LPSTOP 15-19
15.8.4.1 DLCMD2 STOP mode 15-19
15.8.4.2 DLCMD2 LPSTOP mode 15-19
15.8.5 DLCMD2 DEBUG 15-19
15.9 CPU Interface 15-22
15.9.1 Parallel Interface Requirements 15-22
15.9.2 Reset Operation 15-23
15.10 Operational Information 15-24
15.10.1 Initialization 15-24
15.10.1.1 Step 1 — Initialize MCR 15-24
15.10.1.2 Step 2 — Initialize ILR and IVR registers if Interrupts Employed 15-24
15.10.1.3 Step 3 — Initialize SCTL and SDATA registers 15-24
15.10.1.4 Step 4 — Enable DLCMD2 by exiting DEBUG mode 15-24
15.10.2 Transmitting a Message 15-24
15.10.3 Receiving a Message 15-25
15.10.4 Receiving a Message in Block Mode 15-26
15.10.5 Transmitting a Message in Block Mode 15-26
15.10.6 Receiving a Message in 4X Mode 15-26
15.10.7 Transmitting a Message in 4x Mode 15-27
15.11 Register Descriptions 15-27
15.11.1 Module Configuration Register (MCR) 15-27
15.11.2 Interrupt Pending Register (IPR) 15-30
15.11.3 Interrupt Level Register (ILR) 15-31
15.11.4 Interrupt Vector Register (IVR) 15-32
15.11.5 Symbol Timing Control and Pre-Scaler Register (SCTL) 15-32
15.11.6 Symbol Timing Data Register (SDATA) 15-34
15.11.7 Transmit Command Register (CMD) 15-35
15.11.8 Transmit Data Register (TDATA) 15-39
15.11.9 TxFIFO Command Load Sequences 15-39
15.11.10 Transmit Data Register (TDATA) 15-40
15.11.11 Receive Status Register (STAT) 15-41
15.11.12 Receive Data Register (RDATA) 15-44
15.11.13 Completion Code 15-44
15.11.14 Bus Errors 15-47
15.11.15 Data Link Controller Module (DLCMD2) 15-49
15.12 Mask Programmable Bus Error (BERR) Functionality 15-50
15.12.1 BERR_PLUG = 0 15-50
15.12.2 BERR_PLUG = 1 15-50
15.13 Interrupt 15-51
15.13.1 DLCMD2 Interrupts 15-51
15.13.2 Interrupt Structure 15-52

Paragraph Number **Page Number**

15.14 In-Frame Response 15-54
15.14.1 IFR Operation 15-54
15.14.2 IFR Abort Conditions 15-56
15.14.3 IFR Types 15-56
15.14.3.1 Type 1 IFR 15-57
15.14.3.2 1.Type 2 IFR 15-57
15.14.3.3 Type 3 IFR 15-57
15.15 System Overview 15-58
15.16 Test Operation 15-58
15.16.1 Test Configuration Register (TCR) 15-58
15.17 Module I/O Signals 15-59
15.17.1 Signal Descriptions 15-59
15.17.2 External Connections 15-59
15.17.3 Signal Functions 15-59
15.17.3.1 CL2Tx 15-59
15.17.3.2 CL2Rx 15-60
15.17.3.3 4XEN 15-60

Section 16
CAN 2.0B CONTROLLER MODULE

16.1 Features 16-1
16.2 External Pins 16-2
16.3 TouCAN Architecture 16-3
16.3.1 Tx/Rx Message Buffer Structure 16-3
16.3.1.1 Common Fields for Extended and Standard Format Frames 16-4
16.3.1.2 Fields for Extended Format Frames 16-6
16.3.1.3 Fields for Standard Format Frames 16-6
16.3.1.4 Serial Message Buffers 16-6
16.3.1.5 Message Buffer Activation/Deactivation Mechanism 16-7
16.3.1.6 Message Buffer Lock/Release/Busy Mechanism 16-7
16.3.2 Receive Mask Registers 16-7
16.3.3 Bit Timing 16-8
16.3.3.1 Configuring the TouCAN Bit Timing 16-9
16.3.4 Error Counters 16-9
16.3.5 Time Stamp 16-10
16.4 TouCAN Operation 16-11
16.4.1 TouCAN Reset 16-11
16.4.2 TouCAN Initialization 16-11
16.4.3 Transmit Process 16-12
16.4.3.1 Transmit Message Buffer Deactivation 16-13
16.4.3.2 Reception of Transmitted Frames 16-13
16.4.4 Receive Process 16-13
16.4.4.1 Receive Message Buffer Deactivation 16-14

16.4.4.2 Locking and Releasing Message Buffers 16-15
16.4.5 Remote Frames 16-15
16.4.6 Overload Frames 16-16
16.5 Special Operating Modes 16-16
16.5.1 Debug Mode 16-16
16.5.2 Low-Power Stop Mode 16-17
16.5.3 Auto Power Save Mode 16-18
16.6 Interrupts 16-18
16.7 Programmer's Model 16-20
16.7.1 TouCAN Module Configuration Register 16-22
16.7.2 TouCAN Test Configuration Register 16-24
16.7.3 TouCAN Interrupt Configuration Register 16-24
16.7.4 Control Register 0 16-25
16.7.5 Control Register 1 16-26
16.7.6 Prescaler Divide Register 16-27
16.7.7 Control Register 2 16-28
16.7.8 Free Running Timer 16-29
16.7.9 Receive Global Mask Registers 16-29
16.7.10 Receive Buffer 14 Mask Registers 16-30
16.7.11 Receive Buffer 15 Mask Registers 16-30
16.7.12 Error and Status Register 16-30
16.7.13 Interrupt Mask Register 16-32
16.7.14 Interrupt Flag Register 16-33
16.7.15 Error Counters 16-33

Section 17
MODULAR I/O SYSTEM (MIOS14)

17.1 Overview 17-1
17.1.1 Documentation Conventions 17-3
17.1.1.1 Terminology Definitions: 17-3
17.1.2 Bit and Byte Ordering 17-3
17.1.2.1 Supported Data Sizes 17-3
17.1.2.2 MIOS14 Port Size: 17-4
17.1.3 Submodule Numbering Convention 17-4
17.1.4 MIOS14 Module/Submodule Addressing 17-4
17.1.4.1 MIOS14 Base Address 17-4
17.1.4.2 Submodule Base Addresses 17-4
17.1.4.3 Bus Error Support 17-4
17.1.4.4 Wait States 17-4
17.1.4.5 Unimplemented Locations 17-5
17.1.4.6 Unimplemented Bits 17-5
17.2 MIOS14 Naming Convention 17-5
17.2.1 Submodule Naming Convention 17-5

Paragraph Number		Page Number

17.2.2 MIOS14 Pin Naming Convention 17-5
17.3 MIOS14 Functional Overview 17-7
17.3.1 MIOS14 Configuration 17-8
17.4 MIOS14 Feature List 17-8
17.5 MIOS14 Pin Count 17-11
17.6 Submodule Overview 17-11
17.6.1 MIOS Bus Interface Submodule (MBISM) 17-11
17.6.2 MIOS Modulus Counter Submodule (MMCSM) 17-11
17.6.3 MIOS Double Action Submodule (MDASM) 17-12
17.6.4 MIOS Pulse Width Modulation Submodule (MPWMSM) 17-13
17.6.5 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM) 17-13
17.6.6 MIOS Real-Time Clock Submodule (MRTCSM) 17-13
17.6.7 Timing Function Examples 17-13
17.6.7.1 Single-Edge Input Capture 17-14
17.6.7.2 Input Double-Edge Pulse Width Measurement 17-15
17.6.7.3 Input Double-Edge Period Measurement 17-16
17.6.7.4 Single-Edge Output Compare 17-17
17.6.7.5 Double-Edge Single Output Pulse Generation 17-18
17.6.7.6 Output Pulse Width Modulation With MDASM 17-19
17.6.7.7 Input Pulse Accumulation 17-21
17.7 MIOS14 I/O Ports 17-21
17.7.1 Using and Clearing Flag Bits 17-21
17.8 MIOS14 Bus System Description 17-21
17.8.1 Modular I/O Bus Description 17-21
17.8.2 Read/Write and Control Bus Description 17-22
17.8.3 Request Bus Description 17-22
17.8.4 Counter Bus Set Description 17-22
17.9 MIOS14 Address Map 17-22
17.9.1 Interrupts 17-24
17.9.2 MIOS14 Interrupt Structure 17-24
17.9.3 Request Submodule (RQSM) 17-25
17.9.4 MIRSM0 Registers 17-27
17.9.4.1 MIOS14SR0 Interrupt Status Register 17-27
17.9.4.2 MIOS14ER0 Interrupt Enable Registers 17-27
17.9.4.3 MIOS14RPR0 Request Pending Register 17-28
17.9.5 MIRSM1 Registers 17-28
17.9.5.1 MIOS14SR1 Interrupt Status Register 17-28
17.9.5.2 MIOS14ER1 Interrupt Enable Registers 17-29
17.9.5.3 MIOS14RPR1 Request Pending Register 17-29
17.9.6 MBISM Interrupt Registers 17-30
17.9.6.1 MIOS14LVL0 Register 17-30
17.9.6.2 MIOS14LVL1 Register 17-30
17.9.7 Interrupt Control Section (ICS) 17-30

Paragraph Number		Page Number

17.9.7.1 Non-Vectored Interrupt 17-31
17.10 Bus Interface Submodule (MBISM) 17-31
17.10.1 MBISM Overview Description 17-31
17.10.2 MBISM Registers 17-31
17.10.2.1 MBISM Registers Organization 17-31
17.10.2.2 MIOS14 Test and Pin Control Register (MIOS14TPCR) 17-32
17.10.2.3 MIOS14 Vector Register (MIOS14VECT) 17-33
17.10.2.4 MIOS14 Module-Version Number Register (MIOS14VNR) 17-33
17.10.2.5 MIOS14 Module Configuration Register (MIOS14MCR) 17-33
17.11 Counter Prescaler Submodule (MCPSM) 17-34
17.11.1 MCPSM Overview Description 17-34
17.11.1.1 MCPSM features 17-35
17.11.1.2 MCPSM Pin Functions 17-35
17.11.1.3 Modular I/O Bus (MIOB) Interface 17-35
17.11.2 Effect of RESET on MCPSM 17-36
17.11.3 MCPSM Registers 17-36
17.11.3.1 MCPSM Registers Organization. 17-36
17.11.3.2 MCPSMSCR — MCPSM Status/Control Register 17-36
17.12 Modulus Counter Submodule (MMCSM) 17-37
17.12.1 MMCSM Overview Description. 17-37
17.12.2 MMCSM features 17-39
17.12.2.1 MMCSM Pin Functions. 17-39
17.12.3 MMCSM Prescaler 17-40
17.12.4 Modular I/O Bus (MIOB) Interface 17-40
17.12.5 Effect of RESET on MMCSM 17-40
17.12.6 MMCSM Registers 17-41
17.12.7 MMCSM Register Organization 17-41
17.12.8 MMCSM Up-Counter Register (MMCSMCNT). 17-42
17.12.8.1 MMCSM Modulus Latch Register (MMCSMML) 17-42
17.12.8.2 MMCSMSCRD — MMCSM Status/Control Register (Duplicated). 17-43
17.12.8.3 MMCSM Status/Control Register (MMCSMSCR) 17-43
17.12.9 Double Action Submodule (MDASM) 17-45
17.12.10 MDASM Overview Description 17-45
17.12.11 MDASM Features. 17-46
17.12.11.1 MDASM Pin Functions 17-47
17.12.12 MDASM Description 17-47
17.12.13 MDASM Modes of Operation 17-48
17.12.13.1 Disable (DIS) Mode 17-49
17.12.13.2 Input Pulse Width Measurement (IPWM) Mode 17-49
17.12.14 Input Period Measurement (IPM) Mode 17-50
17.12.15 Input Capture (IC) Mode. 17-51
17.12.15.1 Output Compare (OCB and OCAB) Modes 17-52
17.12.15.2 Single Shot Output Pulse Operation. 17-53

Paragraph Number		Page Number

17.12.15.3 Single Output Compare operation 17-54
17.12.15.4 Output Port Bit operation 17-55
17.12.15.5 Output Pulse Width Modulation (OPWM) mode 17-55
17.12.16 Modular I/O Bus (MIOB) Interface 17-57
17.12.17 Effect of RESET on MDASM 17-58
17.12.18 MDASM Registers 17-58
17.12.18.1 MDASM Registers Organization. 17-58
17.12.18.2 MDASM DataA (MDASMAR) Register Bits 17-60
17.12.18.3 MDASM DataB (MDASMBR) Register Bits 17-61
17.12.19 MDASMSCRD — MDASM Status/Control Register (Duplicated) 17-62
17.12.20 MDASMSCR — MDASM Status/Control Register 17-62
17.13 Pulse Width Modulation Submodule (MPWMSM) 17-64
17.13.1 MPWMSM Overview Description 17-64
17.13.2 MPWMSM Terminology 17-65
17.13.2.1 MPWMSM Features 17-65
17.13.3 MPWMSM Description 17-66
17.13.3.1 Clock Selection. 17-67
17.13.3.2 Counter. 17-67
17.13.3.3 Period Register. 17-67
17.13.3.4 Pulse Width Registers 17-68
17.13.3.5 0% and 100% Duty Cycles. 17-69
17.13.3.6 Tables. 17-70
17.13.3.7 MPWMSM Status and Control Register (SCR) 17-70
17.13.3.8 MPWMSM Interrupt 17-71
17.13.3.9 MPWMSM Port Functions 17-71
17.13.3.10 MPWMSM Data Coherency 17-71
17.13.4 MIO Bus (MIOB) Interface 17-72
17.13.5 Effect of RESET on MPWMSM 17-72
17.13.6 MPWMSM Registers 17-72
17.13.6.1 MPWMSM Registers Organization. 17-72
17.13.6.2 MPWMPERR — MPWMSM Period Register 17-74
17.13.6.3 MPWMPULR — MPWMSM Pulse Width Register 17-75
17.13.6.4 MPWMCNTR — MPWMSM Counter Register. 17-76
17.13.6.5 MPWMSCR — MPWMSM Status/Control Register 17-76
17.14 Parallel Port I/O Submodule (MPIOSM) 17-79
17.14.1 MPIOSM Overview Description 17-79
17.14.2 MPIOSM features 17-80
17.14.3 MPIOSM Pin Functions 17-80
17.14.4 MPIOSM Description 17-80
17.14.4.1 MPIOSM Port function 17-80
17.14.4.2 Non-Bonded MPIOSM Pads. 17-81
17.14.5 Modular I/O Bus (MIOB) Interface 17-81
17.14.6 Effect of RESET on MPIOSM 17-81

Paragraph Number		Page Number

17.14.7 MPIOSM Testing 17-81
17.14.8 MPIOSM Registers 17-81
17.14.9 MPIOSM Register Organization 17-82
17.14.9.1 MPIOSMDR — MPIOSM Data Register 17-82
17.14.9.2 MPIOSMDDR — MPIOSM Data Direction Register 17-82
17.15 Real-Time Clock Submodule (MRTCSM) 17-82
17.15.1 MRTCSM Overview Description 17-83
17.15.1.1 MRTCSM Terminology 17-83
17.15.1.2 MRTCSM Features 17-83
17.15.1.3 MRTCSM Pad Functions 17-85
17.15.2 MRTCSM Description 17-85
17.15.2.1 Oscillator 17-85
17.15.2.2 Standby Supply and Power Switch 17-85
17.15.2.3 Counter Chain 17-86
17.15.2.4 15-Bit Prescaler 17-87
17.15.2.5 32-Bit Free-running Counter 17-87
17.15.2.6 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers 17-87
17.15.3 Modes of Operation 17-87
17.15.3.1 Enabling the MRTCSM 17-87
17.15.3.2 15-Bit Prescaler and 32-Bit Free-Running Counter Buffer Updates 17-88
17.15.3.3 Read of 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers 17-88
17.15.3.4 Write to 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers 17-89
17.15.4 MRTCSM Interrupt 17-90
17.15.5 Chip Wake-Up Feature 17-90
17.15.6 Modular I/O Bus (MIOB) Interface 17-90
17.15.6.1 Low Power Mode — Peripheral Bus Clock Running 17-90
17.15.6.2 Low Power Mode — Peripheral Bus Clock Stopped 17-90
17.15.7 Effect of Standby Mode on MRTCSM 17-91
17.15.8 Effect of RESET on MRTCSM 17-91
17.15.9 MRTCSM Registers 17-92
17.15.10 MRTCSM Register Organization 17-92
17.15.10.1 MRTCSM Free-Running Counter High Buffer (MRTCFRCH) Register Bits . . 17-92
17.15.10.2 MRTCSM Free-Running Counter Low Buffer (MRTCFRCL) Register Bits . . 17-93
17.15.10.3 MRTCSM Prescaler Counter Buffer (MRTCPR) Register Bits 17-93
17.15.10.4 MRTCSMSCR — MRTCSM Status/Control Register 17-93

Section 18
TIME PROCESSOR UNIT 3

18.1 Overview 18-1
18.2 TPU3 Components 18-2
18.2.1 Time Bases 18-2
18.2.2 Timer Channels 18-2
18.2.3 Scheduler 18-2

18.2.4 Microengine 18-2
18.2.5 Host Interface 18-3
18.2.6 Parameter RAM 18-3
18.3 TPU Operation 18-3
18.3.1 Event Timing 18-3
18.3.2 Channel Orthogonality 18-3
18.3.3 Interchannel Communication 18-4
18.3.4 Programmable Channel Service Priority 18-4
18.3.5 Coherency 18-4
18.3.6 Emulation Support 18-4
18.3.7 TPU3 Interrupts 18-5
18.3.8 Prescaler Control for TCR1 18-5
18.3.9 Prescaler Control for TCR2 18-7
18.4 Programming Model 18-8
18.4.1 TPU Module Configuration Register 18-11
18.4.2 TPU3 Test Configuration Register 18-13
18.4.3 Development Support Control Register 18-13
18.4.4 Development Support Status Register 18-15
18.4.5 TPU3 Interrupt Configuration Register 18-15
18.4.6 Channel Interrupt Enable Register 18-16
18.4.7 Channel Function Select Registers 18-16
18.4.8 Host Sequence Registers 18-18
18.4.9 Host Service Request Registers 18-18
18.4.10 Channel Priority Registers 18-19
18.4.11 Channel Interrupt Status Register 18-20
18.4.12 Link Register 18-21
18.4.13 Service Grant Latch Register 18-21
18.4.14 Decoded Channel Number Register 18-21
18.4.15 TPU3 Module Configuration Register 2 18-21
18.4.16 TPU Module Configuration Register 3 18-23
18.4.17 TPU3 Test Registers 18-23
18.4.18 TPU3 Parameter RAM 18-23
18.5 Time Functions 18-24

Section 19
DUAL-PORT TPU RAM (DPTRAM)

19.1 Features 19-2
19.2 DPTRAM Configuration and Block Diagram 19-2
19.3 Programming Model 19-2
19.3.1 DPTRAM Module Configuration Register (DPTMCR) 19-4
19.3.2 DPTRAM Test Register 19-5
19.3.3 Ram Base Address Register (RAMBAR) 19-5
19.3.4 MISR High (MISRH) and MISR Low (MISRL) 19-5

19.3.5 MISC Counter (MISCNT) 19-6
19.4 Operation 19-6
19.4.1 Normal Operation 19-6
19.4.2 Standby Operation 19-7
19.4.3 Reset Operation 19-7
19.4.4 Stop Operation 19-7
19.4.5 Freeze Operation 19-8
19.4.6 TPU3 Emulation Mode Operation 19-8
19.5 Multiple Input Signature Calculator (MISC) 19-8

Section 20
CDR3 FLASH (UC3F) EEPROM

20.1 Introduction 20-1
20.1.1 Features of the CDR3 FLASH EEPROM (UC3F) 20-3
20.1.2 Glossary of Terms Used 20-4
20.2 UC3F Interface 20-5
20.2.1 External Interface 20-5
20.3 Programmer's Model 20-6
20.3.0.1 UC3F EEPROM Module Control Register Addressing 20-6
20.3.1 UC3F EEPROM Control Registers 20-6
20.3.2 UC3F EEPROM Configuration Register (UC3FMCR) 20-7
20.3.3 UC3F EEPROM Extended Configuration Register (UC3FMCRE) 20-10
20.3.4 UC3F EEPROM High Voltage Control Register (UC3FCTL) 20-13
20.3.5 UC3F EEPROM Array Addressing 20-17
20.3.6 UC3F EEPROM Shadow Row 20-17
20.3.6.1 Reset Configuration Word (UC3FCFIG) 20-18
20.3.7 UC3F EEPROM 512-Kbyte Array Configuration 20-21
20.4 Operation 20-22
20.4.1 Reset 20-22
20.4.2 Register Read and Write Operation 20-23
20.4.3 Array Read Operation 20-23
20.4.3.1 Array On-Page Read Operation 20-24
20.4.4 Shadow Row Select Read Operation 20-24
20.4.5 Array Program/Erase Interlock Write Operation 20-24
20.4.6 High Voltage Operations 20-24
20.4.6.1 Overview of Program/Erase Operation 20-24
20.4.7 Programming 20-24
20.4.7.1 Program Sequence 20-25
20.4.7.2 Program Shadow Information 20-28
20.4.7.3 Program Suspend 20-28
20.4.8 Erasing 20-28
20.4.8.1 Erase Sequence 20-29
20.4.8.2 Erasing Shadow Information Words 20-31

20.4.8.3 Erase Suspend 20-31
20.4.9 Stop Operation 20-32
20.4.10 Disabled 20-32
20.4.11 Censored Accesses and Non-Censored Accesses 20-33
20.4.11.1 Setting and Clearing Censor 20-34
20.4.11.2 Setting Censor 20-35
20.4.11.3 Clearing Censor 20-35
20.4.11.4 Switching The UC3F EEPROM Censorship 20-36
20.4.12 Background Debug Mode or Freeze Operation 20-37

Section 21
CALRAM OPERATION

21.1 Definitions and Acronyms 21-1
21.1.1 Key Feature List 21-2
21.2 CALRAM Introduction 21-2
21.3 Modes of Operation 21-6
21.3.1 Reset 21-6
21.3.2 One-Cycle Mode 21-6
21.3.2.1 CALRAM Access/Privilege Violations 21-6
21.3.3 Two-Cycle Mode 21-7
21.3.4 Standby Operation / Keep-Alive Power 21-7
21.3.5 Stop Operation 21-7
21.3.6 Overlay Mode Operation 21-7
21.3.6.1 Overlay Mode Configuration 21-8
21.3.6.2 Priority of Overlay Regions 21-13
21.3.6.3 Normal (Non-Overlay) Access to Overlay Regions 21-13
21.3.6.4 Calibration Write Cycle Flow 21-14
21.4 Register Definitions 21-14
21.4.1 CALRAM Module Configuration Register (CRAMMCR) 21-15
21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx) 21-18
21.4.3 CALRAM Overlay Configuration Register (CRAMOVLCR) 21-20
21.4.4 CALRAM Ownership Trace Register (CRAMOTR) 21-21

Section 22
DEVELOPMENT SUPPORT

22.1 Overview 22-1
22.2 Program Flow Tracking 22-1
22.2.1 Program Trace Cycle 22-2
22.2.1.1 Instruction Queue Status Pins — VF [0:2] 22-3
22.2.1.2 History Buffer Flushes Status Pins— VFLS [0..1] 22-4
22.2.1.3 Queue Flush Information Special Case 22-4
22.2.2 Program Trace when in Debug Mode 22-4
22.2.3 Sequential Instructions Marked as Indirect Branch 22-4
22.2.4 The External Hardware 22-5

Paragraph Number		Page Number

22.2.4.1 Synchronizing the Trace Window to the CPU Internal Events 22-5
22.2.4.2 Detecting the Trace Window Start Address 22-6
22.2.4.3 Detecting the Assertion/Negation of VSYNC 22-6
22.2.4.4 Detecting the Trace Window End Address. 22-7
22.2.4.5 Compress 22-7
22.2.5 Instruction Fetch Show Cycle Control 22-7
22.3 Watchpoints and Breakpoints Support. 22-8
22.3.1 Internal Watchpoints and Breakpoints 22-10
22.3.1.1 Restrictions. 22-12
22.3.1.2 Byte and Half-Word Working Modes 22-12
22.3.1.3 Examples 22-13
22.3.1.4 Context Dependent Filter 22-14
22.3.1.5 Ignore First Match. 22-15
22.3.1.6 Generating Six Compare Types. 22-15
22.3.2 Instruction Support 22-15
22.3.2.1 Load/Store Support 22-16
22.3.3 Watchpoint Counters 22-20
22.3.3.1 Trap Enable Programming 22-20
22.4 Development System Interface 22-20
22.4.1 Debug Mode Support 22-23
22.4.1.1 Debug Mode Enable vs. Debug Mode Disable. 22-25
22.4.1.2 Entering Debug Mode. 22-25
22.4.1.3 The Check Stop State and Debug Mode 22-28
22.4.1.4 Saving Machine State upon Entering Debug Mode 22-28
22.4.1.5 Running in Debug Mode. 22-29
22.4.1.6 Exiting Debug Mode 22-29
22.5 Development Port 22-30
22.5.1 Development Port Pins. 22-30
22.5.2 Development Serial Clock 22-30
22.5.3 Development Serial Data In 22-30
22.5.4 Development Serial Data Out. 22-31
22.5.5 Freeze Signal 22-31
22.5.5.1 SGPIO6/FRZ/$\overline{\text{PTR}}$ Pin 22-31
22.5.5.2 IWP[0:1]/VFLS[0:1] Pins. 22-31
22.5.5.3 VFLS[0:1]_MPIO32B[3:4] Pins. 22-31
22.5.6 Development Port Registers. 22-31
22.5.6.1 Development Port Shift Register 22-32
22.5.6.2 Trap Enable Control Register. 22-32
22.5.6.3 Development Port Registers Decode 22-32
22.5.6.4 Development Port Serial Communications — Clock Mode Selection 22-32
22.5.6.5 Development Port Serial Communications — Trap Enable Mode 22-37
22.5.6.6 Serial Data into Development Port — Trap Enable Mode 22-37
22.5.6.7 Serial Data Out of Development Port — Trap Enable Mode 22-38

22.5.6.8 Development Port Serial Communications — Debug Mode 22-38
22.5.6.9 Serial Data Into Development Port 22-39
22.5.6.10 Serial Data Out of Development Port 22-40
22.5.6.11 Fast Download Procedure 22-40
22.6 Software Monitor Debugger Support 22-42
22.6.1 Freeze Indication 22-42
22.7 Development Support Registers 22-42
22.7.1 Register Protection 22-43
22.7.2 Comparator A–D Value Registers (CMPA–CMPD) 22-44
22.7.3 Comparator E–F Value Registers 22-45
22.7.4 Breakpoint Address Register (BAR) 22-45
22.7.5 Comparator G–H Value Registers (CMPG–CMPH) 22-45
22.7.6 I-Bus Support Control Register 22-46
22.7.7 L-Bus Support Control Register 1 22-48
22.7.8 L-Bus Support Control Register 2 22-49
22.7.9 Breakpoint Counter A Value and Control Register 22-50
22.7.10 Breakpoint Counter B Value and Control Register 22-51
22.7.11 Exception Cause Register (ECR) 22-51
22.7.12 Debug Enable Register (DER) 22-53
22.7.13 Development Port Data Register (DPDR) 22-55

Section 23
READI MODULE

23.1 Overview 23-1
23.1.1 General Description 23-1
23.1.2 Feature Summary List 23-1
23.1.3 Functional Block Diagram 23-2
23.1.4 Modes of Operation 23-3
23.1.4.1 READI Reset Configuration 23-4
23.1.4.2 Security 23-4
23.1.4.3 Disabled 23-4
23.1.5 Parametrics 23-4
23.1.6 Programmer’s Model 23-5
23.1.7 Messages 23-5
23.1.8 Terms and Definitions 23-6
23.2 Programmer’s Model 23-8
23.2.1 Register Map 23-8
23.2.1.1 User Mapped Register 23-8
23.2.1.2 Tool Mapped Registers 23-9
23.2.1.3 Device ID (DID) Register 23-9
23.2.1.4 Development Control (DC) Register 23-10
23.2.1.5 RCPU Development Access Modes 23-11
23.2.1.6 User Base Address (UBA) Register 23-11

Paragraph Number		Page Number

23.2.1.7 Read/Write Access (RWA) Register . . . 23-13
23.2.1.8 Upload/Download Information (UDI) Register . . . 23-14
23.2.1.9 Data Trace Attributes 1 and 2 (DTA1 and DTA2) Registers . . . 23-16
23.2.2 Accessing Memory Mapped Locations Via the Auxiliary Port . . . 23-17
23.2.3 Accessing READI Tool Mapped Registers Via the Auxiliary Port . . . 23-18
23.2.4 Partial Register Updates . . . 23-19
23.2.5 Programming Considerations . . . 23-19
23.2.5.1 Program Trace Guidelines . . . 23-19
23.2.5.2 Compressed Code Mode Guidelines . . . 23-19
23.2.5.3 Reset Sequence Guidelines . . . 23-21
23.2.5.4 BDM Guidelines . . . 23-21
23.3 Pin Interface . . . 23-21
23.3.1 Functional Description . . . 23-21
23.3.1.1 Pins Implemented . . . 23-21
23.3.2 Functional Block Diagram . . . 23-22
23.3.2.1 Message Priority . . . 23-23
23.3.2.2 Pin Protocol . . . 23-23
23.3.2.3 Messages . . . 23-26
23.3.2.4 Message Formats . . . 23-29
23.3.2.5 Rules of Messages . . . 23-31
23.3.2.6 Examples . . . 23-31
23.3.2.7 Non-Temporal Ordering of Transmitted Messages . . . 23-33
23.3.3 READI Reset Configuration . . . 23-33
23.3.3.1 Reset Configuration for Debug Mode . . . 23-34
23.3.3.2 Reset Configuration for Non-Debug Mode . . . 23-34
23.3.3.3 Secure Mode . . . 23-34
23.3.3.4 Disabled Mode . . . 23-34
23.3.3.5 Guidelines For Transmitting Input Messages . . . 23-35
23.4 Program Trace . . . 23-35
23.4.1 Branch Trace Messaging . . . 23-35
23.4.1.1 RCPU Instructions that Cause BTM Messages . . . 23-36
23.4.2 Review of RCPU Instruction Execution . . . 23-36
23.4.2.1 RCPU Pipeline and Execution Model . . . 23-36
23.4.2.2 RCPU Branch Trace Indicators . . . 23-38
23.4.3 BTM Message Formats . . . 23-40
23.4.3.1 Direct Branch Messages . . . 23-40
23.4.3.2 Indirect Branch Messages . . . 23-40
23.4.3.3 Program Trace Correction Message . . . 23-41
23.4.3.4 Error Message (Queue Overflow) . . . 23-44
23.4.3.5 Program Trace Synchronization Messages . . . 23-45
23.4.3.6 Direct Branch Synchronization Message . . . 23-46
23.4.3.7 Indirect Branch Synchronization Message . . . 23-46
23.4.3.8 Direct Branch Synchronization Message With Compressed Code . . . 23-47

23.4.3.9 Indirect Branch Synchronization Message with Compressed Code 23-47
23.4.3.10 Relative Addressing 23-47
23.4.4 BTM Operation 23-48
23.4.4.1 BTM Capture and Encoding Algorithm 23-48
23.4.4.2 Instruction Fetch Snooping 23-48
23.4.4.3 Instruction Execution Tracking 23-49
23.4.4.4 Instruction Flush Cases 23-51
23.4.5 BTM Queueing 23-51
23.4.6 Timing Diagram 23-52
23.4.7 Program Trace Guidelines 23-56
23.5 Data Trace 23-56
23.5.1 Data Trace for the Load/Store Bus (L-bus) 23-56
23.5.2 Data Trace Message Formats 23-56
23.5.2.1 Data Write Message 23-57
23.5.2.2 Data Read Message 23-57
23.5.2.3 Data Trace Synchronization Messages 23-57
23.5.2.4 Data Write Synchronization Message 23-58
23.5.2.5 Data Read Synchronization Messaging 23-58
23.5.2.6 Relative Addressing 23-59
23.5.3 Error Message (Queue Overflow) 23-59
23.5.4 Data Trace Operation 23-59
23.5.5 Data Trace Windowing 23-60
23.5.6 Special L-bus Cases 23-61
23.5.7 Data Trace Queuing 23-61
23.5.8 Throughput and Latency 23-62
23.5.8.1 Assumptions for Throughput Analysis 23-62
23.5.8.2 Throughput Calculations 23-62
23.5.9 Timing Diagrams 23-62
23.6 Read/Write Access 23-64
23.6.1 Functional Description 23-64
23.6.2 Write Operation to Memory Mapped Locations and PowerPC Registers 23-66
23.6.2.1 Single Write Operation 23-66
23.6.2.2 Block Write Operation 23-66
23.6.3 Read Operation to Memory Mapped Locations and PowerPC Registers 23-67
23.6.3.1 Single Read Operation 23-67
23.6.3.2 Block Read Operation 23-68
23.6.4 Read/Write Access to Internal READI Registers 23-69
23.6.4.1 Write Operation 23-69
23.6.4.2 Read Operation 23-69
23.6.5 Error Handling 23-69
23.6.5.1 Access Alignment 23-69
23.6.5.2 L-bus Address Error 23-70
23.6.5.3 L-bus Data Error 23-70

Paragraph Number		Page Number

23.6.6 Exception Sequences 23-70
23.6.7 Secure Mode 23-71
23.6.8 Error Messages 23-71
23.6.8.1 Read/Write Access Error 23-71
23.6.8.2 Invalid Message 23-71
23.6.8.3 Invalid Access Opcode 23-72
23.6.9 Faster Read/Write Accesses With Default Attributes 23-72
23.6.10 Throughput and Latency 23-72
23.6.10.1 Assumptions for Throughput Analysis 23-72
23.7 Timing Diagrams 23-74
23.8 Watchpoint Support 23-77
23.8.1 Watchpoint Messaging 23-77
23.8.1.1 Watchpoint Source Field 23-78
23.8.2 Error Message (Watchpoint Overrun) 23-78
23.8.3 Synchronization 23-79
23.8.4 Timing Diagrams 23-79
23.9 Ownership Trace 23-79
23.9.1 Ownership Trace Messaging 23-80
23.9.2 Error Message (Queue Overflow) 23-80
23.9.2.1 OTM Flow 23-80
23.9.2.2 OTM Queueing 23-81
23.9.3 Timing Diagram 23-81
23.10 RCPU Development Access 23-82
23.10.1 RCPU Development Access Messaging 23-83
23.10.1.1 DSDI Message 23-83
23.10.1.2 DSDO Message 23-84
23.10.1.3 BDM Status Message 23-84
23.10.1.4 Error Message (Invalid Message) 23-85
23.10.2 RCPU Development Access Operation 23-85
23.10.2.1 Enabling RCPU Development Access Via READI Pins 23-86
23.10.2.2 Enabling Background Debug Mode (BDM) via READI Pins 23-86
23.10.2.3 Entering Background Debug Mode (BDM) Via READI Pins 23-87
23.10.2.4 Non-Debug Mode Access of RCPU Development Access 23-87
23.10.2.5 RCPU Development Access Flow Diagram 23-87
23.10.3 Throughput 23-89
23.10.4 Timing Diagrams 23-89
23.11 Power Management 23-92
23.11.1 Functional Description 23-92
23.11.2 Low Power Modes 23-93
23.12 Application Notes 23-93
23.12.1 Automotive Calibration 23-93
23.12.1.1 Calibration Variable Acquisition 23-93
23.12.2 Calibration Variables Located In Contiguous Memory Locations 23-94

23.12.2.1 Data Read Messaging 23-94
23.12.2.2 Read/Write Access 23-94
23.12.3 Calibration Variables Not Located in Contiguous Memory Locations 23-94
23.12.3.1 Data Write Messaging 23-94
23.12.3.2 Read/Write Access 23-95
23.12.3.3 Calibration Constant Tuning 23-96
23.12.4 UC3F Programming Guideline Flowchart via READI Read/Write Access 23-96

Section 24
IEEE 1149.1-COMPLIANT INTERFACE (JTAG)

24.1 IEEE 1149.1 Test Access Port (TAP) and Joint Test Action Group (JTAG) 24-1
24.2 IEEE 1149.1 Test Access Port 24-1
24.2.1 Overview 24-2
24.2.1.1 TAP CONTROLLER 24-3
24.2.1.2 Boundary Scan Register 24-4
24.2.2 Instruction Register 24-23
24.2.2.1 EXTEST 24-24
24.2.2.2 SAMPLE/PRELOAD 24-24
24.2.2.3 BYPASS 24-24
24.2.2.4 CLAMP 24-25
24.2.3 HI-Z 24-25
24.3 MPC565 / MPC566 Restrictions 24-25
24.3.1 Non-Scan Chain Operation 24-26
24.3.2 Motorola MPC565 / MPC566 BSDL Description 24-27

Appendix A
INTERNAL MEMORY MAP

Appendix B
REGISTER GENERAL INDEX

Appendix C
REGISTER DIAGRAM INDEX

Appendix D
TPU ROM FUNCTIONS

D.1 Overview D-1
D.2 Programmable Time Accumulator (PTA) D-4
D.3 Queued Output Match TPU Function (QOM) D-6
D.4 Table Stepper Motor (TSM) D-8
D.5 Frequency Measurement (FQM) D-11
D.6 Universal Asynchronous Receiver/Transmitter (UART) D-13
D.7 New Input Capture/Transition Counter (NITC) D-16
D.8 Multiphase Motor Commutation (COMM) D-18
D.9 Hall Effect Decode (HALLD) D-20
D.10 Multichannel Pulse-Width Modulation (MCPWM) D-22
D.11 Fast Quadrature Decode TPU Function (FQD) D-29

Paragraph Number | **Page Number**

D.12 Period/Pulse-Width Accumulator (PPWA) D-32
D.13 Output Compare (OC) D-34
D.14 Pulse-Width Modulation (PWM) D-36
D.15 Discrete Input/Output (DIO) D-38
D.16 Synchronized Pulse-Width Modulation (SPWM) D-40
D.17 Read / Write Timers and Pin TPU Function (RWTPIN) D-43
D.18 ID TPU Function (ID) D-45
D.19 Serial Input/Output Port (SIOP) D-47
D.19.1 Parameters D-48
D.19.1.1 CHAN_CONTROL D-49
D.19.1.2 BIT_D D-50
D.19.1.3 HALF_PERIOD D-50
D.19.1.4 BIT_COUNT D-50
D.19.1.5 XFER_SIZE D-50
D.19.1.6 SIOP_DATA D-50
D.19.2 Host CPU Initialization of the SIOP Function D-51
D.19.3 SIOP Function Performance D-51
D.19.3.1 XFER_SIZE Greater Than 16 . . . . D-52
D.19.3.2 Data Positioning D-52
D.19.3.3 Data Timing D-52

Appendix E
ELECTRICAL CHARACTERISTICS

E.1 Absolute Maximum Ratings (VSS = 0V) E-2
E.2 Package E-3
E.3 EMI Characteristics E-3
E.3.1 Reference Documents E-3
E.3.2 Definitions and Acronyms E-3
E.3.3 EMI Testing Specifications E-3
E.4 Thermal Characteristics E-4
E.4.1 Thermal References E-6
E.5 ESD Protection E-7
E.6 DC Electrical Characteristics E-7
E.7 Oscillator and PLL Electrical Characteristics E-12
E.8 Flash Electrical Characteristic E-12
E.9 Power-Up/Down Sequencing E-13
E.9.1 Power-Up/Down Option A E-14
E.9.2 Power-Up/Down Option B E-16
E.10 Issues Regarding Power Sequence E-19
E.10.1 Flash Issue E-19
E.10.2 Keep-Alive RAM E-19
E.11 Timing E-20
E.12 Debug Port Timing E-45
E.13 Pin Electrical Characteristics E-46
E.13.1 AC Electrical Characteristics E-46
E.14 RESET Timing E-49
E.15 IEEE 1149.1 Electrical Characteristics E-53
E.16 QADC64E Electrical Characteristics E-57
E.17 QSMCM Electrical Characteristics E-58
E.18 GPIO Electrical Characteristics E-61
E.19 TPU3 Electrical Characteristics E-62
E.20 TouCAN Electrical Characteristics E-63
E.21 MIOS Timing Characteristics E-63

Paragraph Number	Page Number

E.21.1 MPWMSM Timing Characteristics E-64
E.21.2 MMCSM Timing Characteristics. E-67
E.21.3 MDASM Timing Characteristics E-70
E.22 MSASM Timing Characteristics E-73
E.22.1 MPIOSM Timing Characteristics E-76

INDEX

Online publishing by **JABIS**™, *http://www.jabis.com*

LIST OF FIGURES

Figure Number		Page Number
1-1	MPC565 / MPC566 Block Diagram	1-2
1-2	MPC565 / MPC566 Memory Map	1-10
1-3	MPC565 / MPC566 Internal Memory Block	1-11
1-4	MPC565 / MPC566 Peripheral Memory Map	1-12
2-1	Input Propagation Delay	2-40
2-2	Input Levels	2-40
2-3	Output Propagation Delay	2-41
2-4	Output Levels and Timing	2-41
2-5	MPC565 / MPC566 Package Footprint (1 of 2)	2-54
2-5	Figure 2-5 MPC565 / MPC566 Package Footprint (2 of 2)	2-55
2-6	MPC565 / MPC566 Redistributed Bump Map	2-56
2-7	MPC565 / MPC566 Ball Diagram	2-57
2-8	MPC565 / MPC566 Ball Diagram Legend	2-58
2-9	MPC565 / MPC566 Ball Map (Black and White, page 1)	2-59
2-9	Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 2)	2-60
2-9	Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 3)	2-61
2-9	Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 4)	2-62
3-1	RCPU Block Diagram	3-2
3-2	Sequencer Data Path	3-4
3-3	RCPU Programming Model	3-8
3-4	Basic Instruction Pipeline	3-37
4-1	BBC Module Block Diagram	4-2
4-2	Instruction Compression Alternatives	4-6
4-3	Compressed Address Format	4-7
4-4	Code Compression Process	4-8
4-5	Code Decompression Process	4-9
4-6	Exception Table Entries Mapping	4-14
4-7	External Interrupt Vectors Splitting	4-18
4-8	DECRAM Interfaces Block Diagram	4-19
4-9	BTB Block Diagram	4-22
5-1	USIU Block Diagram	5-3
6-1	System Configuration and Protection Logic	6-3
6-2	MPC565 / MPC566 Memory Map	6-5
6-3	SGPIO Cell	6-9
6-4	MPC565 / MPC566 Interrupt Structure	6-10
6-5	Lower Priority Request Masking — One Bit Diagram	6-15
6-6	MPC565 / MPC566 Interrupt Controller Block Diagram	6-16
6-7	Typical Interrupt Handler Routine	6-18

Figure Number		Page Number

6-8 RTC Block Diagram 6-21
6-9 PIT Block Diagram 6-22
6-10 SWT Block Diagram 6-24
6-11 Example of SIVEC Register Usage for Interrupt Table Handling 6-35

7-1 Reset Configuration Basic Scheme 7-7
7-2 Reset Configuration Sampling Scheme For “Short” $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled 7-8
7-3 Reset Configuration Timing for “Short” $\overline{\text{PORESET}}$ Assertion, Limp Mode Enabled 7-9
7-4 Reset Configuration Timing for “Long” $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled 7-9
7-5 Reset Configuration Sampling Timing Requirements 7-10

8-1 Clock Unit Block Diagram 8-2
8-2 Main System Oscillator Crystal Configuration 8-3
8-3 System PLL Block Diagram 8-5
8-4 MPC565 / MPC566 Clocks 8-7
8-5 General System Clocks Select 8-10
8-6 Divided System Clocks Timing Diagram 8-11
8-7 Clocks Timing For DFNH = 1 (or DFNL = 0) 8-12
8-8 Clock Source Switching Flow Chart 8-14
8-9 MPC565 / MPC566 Low-Power Modes Flow Diagram 8-19
8-10 Basic Power Supply Configuration 8-23
8-11 External Power Supply Scheme 8-24
8-12 Keep-Alive Register Key State Diagram 8-26
8-13 No Standby, No KAPWR, All System Power On/Off 8-27
8-14 Standby and KAPWR, Other Power On/Off 8-28

9-1 Input Sample Window 9-2
9-2 MPC565 / MPC566 Bus Signals 9-3
9-3 Basic Transfer Protocol 9-8
9-4 Basic Flow Diagram of a Single Beat Read Cycle 9-9
9-5 Single Beat Read Cycle–Basic Timing–Zero Wait States 9-10
9-6 Single Beat Read Cycle – Basic Timing – One Wait State 9-11
9-7 Basic Flow Diagram of a Single Beat Write Cycle 9-12
9-8 Single Beat Basic Write Cycle Timing — Zero Wait States 9-13
9-9 Single Beat Basic Write Cycle Timing — One Wait State 9-14
9-10 Single Beat 32-Bit Data Write Cycle Timing — 16-Bit Port Size 9-15
9-11 Basic Flow Diagram Of A Burst Read Cycle 9-18
9-12 Burst-Read Cycle — 32-Bit Port Size — Zero Wait State 9-19
9-13 Burst-Read Cycle — 32-Bit Port Size — One Wait State 9-20
9-14 Burst-Read Cycle — 32-Bit Port Size — Wait States Between Beats 9-21
9-15 Burst-Read Cycle — 16-Bit Port Size 9-22
9-16 Basic Flow Diagram of a Burst Write Cycle 9-23

Figure Number		Page Number

9-17 Burst-Write Cycle, 32-Bit Port Size, Zero Wait States (Only for External Master Memory Controller Service Support) 9-24
9-18 Burst-Inhibit Cycle, 32-Bit Port Size (Emulated Burst) 9-25
9-19 Non-Wrap Burst with Three Beats 9-26
9-20 Non-Wrap Burst with One Data Beat 9-27
9-21 Internal Operand Representation 9-28
9-22 Interface To Different Port Size Devices 9-29
9-23 Bus Arbitration Flowchart 9-31
9-24 Masters Signals Basic Connection 9-32
9-25 Bus Arbitration Timing Diagram 9-33
9-26 Internal Bus Arbitration State Machine 9-35
9-27 Termination Signals Protocol Basic Connection 9-40
9-28 Termination Signals Protocol Timing Diagram 9-41
9-29 Reservation On Local Bus 9-42
9-30 Reservation On Multilevel Bus Hierarchy 9-43
9-31 Retry Transfer Timing — Internal Arbiter 9-45
9-32 Retry Transfer Timing — External Arbiter 9-46
9-33 Retry On Burst Cycle 9-47
9-34 Basic Flow of an External Master Read Access 9-49
9-35 Basic Flow of an External Master Write Access 9-50
9-36 Peripheral Mode: External Master Reads from MPC565 / MPC566 — Two Wait States 9-51
9-37 Peripheral Mode: External Master Writes to MPC565 / MPC566 — Two Wait States 9-52
9-38 Flow of Retry of External Master Read Access 9-54
9-39 Retry of External Master Access (Internal Arbiter) 9-55
9-40 Instruction Show Cycle Transaction 9-57
9-41 Data Show Cycle Transaction 9-58

10-1 Memory Controller Function Within the USIU 10-1
10-2 Memory Controller Block Diagram 10-2
10-3 MPC565 / MPC566 Simple System Configuration 10-3
10-4 Bank Base Address and Match Structure 10-4
10-5 MPC565 GPCM–Memory Devices Interface 10-8
10-6 Memory Devices Interface Basic Timing (ACS = 00,TRLX = 0) 10-9
10-7 Peripheral Devices Interface 10-10
10-8 Peripheral Devices Basic Timing (ACS = 11,TRLX = 0) 10-10
10-9 Relaxed Timing — Read Access (ACS = 11, SCY = 1, TRLX = 1) 10-12
10-10 Relaxed Timing — Write Access (ACS = 10, SCY = 0, CSNT = 0, TRLX = 1) 10-13
10-11 Relaxed Timing — Write Access (ACS = 11, SCY = 0, CSNT = 1, TRLX = 1) 10-14

Figure Number		Page Number

10-12 Relaxed Timing — Write Access (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1 10-15
10-13 Consecutive Accesses (Write After Read, EHTR = 0) 10-16
10-14 Consecutive Accesses (Write After Read, EHTR = 1) 10-17
10-15 Consecutive Accesses (Read After Read From Different Banks, EHTR = 1) 10-18
10-16 Consecutive Accesses (Read After Read From Same Bank, EHTR = 1) 10-19
10-17 Aliasing Phenomena Illustration 10-24
10-18 Synchronous External Master Configuration For GPCM-Handled Memory Devices 10-26
10-19 Synchronous External Master Basic Access (GPCM Controlled) 10-27

11-1 L2U Bus Interface Block Diagram 11-2
11-2 DMP Basic Functional Diagram 11-4
11-3 Region Base Address Example 11-7

12-1 UIMB Interface Module Block Diagram 12-2
12-2 IMB Clock – Full-Speed IMB Bus 12-3
12-3 IMB Clock – Half-Speed IMB Bus 12-3
12-4 Interrupt Synchronizer Signal Flow 12-4
12-5 Time-Multiplexing Protocol for IRQ pins 12-5
12-6 Interrupt Synchronizer Block Diagram 12-6

13-1 Dual QADC64Es on MPC565 / MPC566 13-3
13-2 Block Diagram of QADC64E 13-4
13-3 CCW Queue and Result Table Block Diagram 13-7
13-4 Example of External Multiplexing 13-8
13-5 Interrupt Levels on IRQ with ILBS 13-14
13-6 Queue Status Transition 13-29
13-7 QADC64E Conversion Queue Operation 13-31
13-8 QADC64E Analog Subsystem Block Diagram 13-39
13-9 Conversion Timing 13-40
13-10 QADC64E Queue Operation With Pause 13-43
13-11 QADC64E Clock Subsystem Functions 13-53
13-12 Bus Cycle Accesses 13-56
13-13 CCW Priority Situation 1 13-59
13-14 CCW Priority Situation 2 13-60
13-15 CCW Priority Situation 3 13-60
13-16 CCW Priority Situation 4 13-61
13-17 CCW Priority Situation 5 13-61
13-18 CCW Priority Situation 6 13-62
13-19 CCW Priority Situation 7 13-63
13-20 CCW Priority Situation 8 13-63
13-21 CCW Priority Situation 9 13-64

Figure Number		Page Number

13-22 CCW Priority Situation 10 13-64
13-23 CCW Priority Situation 11 13-65
13-24 CCW Freeze Situation 12 13-65
13-25 CCW Freeze Situation 13 13-66
13-26 CCW Freeze Situation 14 13-66
13-27 CCW Freeze Situation 15 13-66
13-28 CCW Freeze Situation 16 13-67
13-29 CCW Freeze Situation 17 13-67
13-30 CCW Freeze Situation 18 13-67
13-31 CCW Freeze Situation 19 13-68
13-32 External Trigger Mode (Positive Edge) Timing With Pause 13-69
13-33 Gated Mode, Single-Scan Timing 13-70
13-34 Gated Mode, Continuous Scan Timing 13-71
13-35 Equivalent Analog Input Circuitry 13-73
13-36 Errors Resulting from Clipping 13-74
13-37 Star-Ground at the Point of Power Supply Origin 13-75
13-38 Electrical Model of an A/D Input Pin 13-77
13-39 External Multiplexing of Analog Signal Sources 13-79
13-40 Input PIn Subjected to Negative Stress 13-82
13-41 Input Pin Subjected to Positive Stress 13-82

14-1 QSMCM Block Diagram 14-2
14-2 QSMCM Interrupt Levels 14-7
14-3 QSPI Interrupt Generation 14-8
14-4 QSPI Block Diagram 14-15
14-5 QSPI RAM 14-23
14-6 Flowchart of QSPI Initialization Operation 14-28
14-7 Flowchart of QSPI Master Operation (Part 1) 14-29
14-7 Flowchart of QSPI Master Operation (Part 2) 14-30
14-7 Flowchart of QSPI Master Operation (Part 3) 14-31
14-8 Flowchart of QSPI Slave Operation (Part 1) 14-32
14-8 Flowchart of QSPI Slave Operation (Part 2) 14-33
14-9 SCI Transmitter Block Diagram 14-43
14-10 SCI Receiver Block Diagram 14-44
14-11 Start Search Example 14-57
14-12 Queue Transmitter Block Enhancements 14-63
14-13 Queue Transmit Flow 14-65
14-14 Queue Transmit Software Flow 14-66
14-15 Queue Transmit Example for 17 Data Bytes 14-67
14-16 Queue Transmit Example for 25 Data Frames 14-68
14-17 Queue Receiver Block Enhancements 14-69
14-18 Queue Receive Flow 14-72
14-19 Queue Receive Software Flow 14-73
14-20 Queue Receive Example for 17 Data Bytes 14-74

15-1 DLCMD2 Block Diagram 15-3

Figure Number		Page Number

15-2 Typical Transmit/Receive Operation 15-5
15-3 VPW Signal Waveforms 15-13
15-4 Huntzicker Waveform Message 15-14
15-5 Start of Frame Symbol 15-14
15-6 Passive “0” and Active “0” 15-15
15-7 Passive “1” and Active “1’ 15-15
15-8 End of Data Symbol 15-15
15-9 End of Frame Symbol 15-16
15-10 Break Symbol Controller Module (DLCMD2) 15-16
15-11 Support For External Transceiver 15-17
15-12 DLCMD2 Operation Modes 15-18
15-13 STOP Power Mode (No STOP Bit Set) 15-20
15-14 DEBUG Power Mode (IFEEZEB or SOFT_FRZ) 15-21
15-15 STOP Power Mode (STOP Bit Set) 15-21
15-16 LPSTOP Power Mode 15-22
15-17 Interrupt Request Logic Path 15-30
15-18 DLCMD2 SDATA Block Diagram 15-35
15-19 TxFIFO Block Diagram 15-41
15-20 RxFIFO Block Diagram 15-44
15-21 Completion Code Byte Bit Definitions 15-45
15-22 SOF Symbol Too Short 15-48
15-23 Figure 1-23 Short Bit Too Short 15-48
15-24 Long Bit Too Long 15-49
15-25 EOD and EOF Too Short 15-49
15-26 Normalization Bit Too Short 15-50
15-27 Normalization Bit Too Long 15-50
15-28 Interrupt Request Multiplex Timing 15-53
15-29 DLCMD2 Interrupt Vector Generation(IRQ_PLUG = 1) 15-54
15-30 Types of IFR 15-57
15-31 J1850 Node 15-58
15-32 DLCMD2 External Connections 15-59

16-1 TouCAN Block Diagram 16-1
16-2 Typical CAN Network 16-3
16-3 Extended ID Message Buffer Structure 16-4
16-4 Standard ID Message Buffer Structure 16-4
16-5 Interrupt levels on IRQ with ILBS 16-20
16-6 TouCAN Message Buffer Memory Map 16-22

17-1 MPC565 / MPC566 MIOS14 Block Diagram 17-2
17-2 Example: Single-Edge Input Capture 17-14
17-3 Example: Double-Edge Pulse Width Measurement 17-16
17-4 Example: Double-Edge Period Measurement 17-17
17-5 MIOS Example: Single-Edge Output Compare 17-18
17-6 MIOS Example: Double-Edge Output Compare 17-19
17-7 MIOS Example: Pulse Width Modulation Output 17-20

Figure Number		Page Number

17-8 MIOS14 Memory Map 17-23
17-9 MIOS Interrupt Structure 17-25
17-10 MCPSM Block Diagram 17-35
17-11 MMCSM Block Diagram 17-38
17-12 MMCSM Modulus Up-Counter 17-39
17-13 MDASM Block Diagram 17-46
17-14 Input Pulse Width Measurement Example 17-50
17-15 Input Period Measurement Example 17-51
17-16 MDASM Input Capture Example 17-52
17-17 Single Shot Output Pulse Example 17-54
17-18 Single Shot Output Transition Example 17-54
17-19 MDASM Output Pulse Width Modulation Example 17-56
17-20 MPWMSM Block Diagram 17-65
17-21 MPIOSM 1-Bit Block Diagram 17-80
17-22 MRTCSM Block Diagram 17-84
17-23 MIOS RTC Oscillator Circuit 17-85
17-24 Selection of vt_vddrtc 17-86
17-25 15-Bit Prescaler and 32-Bit Free-Running Counter Buffer Updates 17-88

18-1 TPU3 Block Diagram 18-1
18-2 TPU3 Interrupt Levels 18-5
18-3 TCR1 Prescaler Control 18-7
18-4 TCR2 Prescaler Control 18-8

19-1 DPTRAM Configuration 19-2
19-2 DPTRAM Memory Map 19-3

20-1 Block Diagram for a 512-Kbyte UC3F Module Configuration 20-2
20-2 PEGOOD Valid Time 20-16
20-3 Shadow Information 20-18
20-4 512-Kbyte Array Configurations 20-22
20-5 Program State Diagram 20-26
20-6 Erase State Diagram 20-30
20-7 Censorship States and Transitions 20-36

21-1 System Block Diagram 21-3
21-2 MPC565 / MPC566 Memory Map with CALRAM Address Ranges 21-4
21-3 Standby Power Supply Configuration for CALRAM Array 21-5
21-4 CALRAM_A array 21-8
21-5 CALRAM_A Module Overlay Map of Flash (CLPS = 0) 21-9
21-6 CALRAM_B and CALRAM_A Address Map (When CLPS = 0) 21-10
21-7 CALRAM Module Overlay Map of FLASH (CLPS = 1) 21-11
21-8 CALRAM_B and CALRAM_A Address Map (When CLPS = 1 for Both CALRAM_A and CALRAM_B) 21-12

Figure Number		Page Number

22-1 Watchpoints and Breakpoint Support in the CPU 22-9
22-2 Partially Supported Watchpoint/Breakpoint Example 22-14
22-3 Instruction Support General Structure 22-16
22-4 Load/Store Support General Structure 22-19
22-5 Functional Diagram of MPC565 / MPC566 Debug Mode Support 22-22
22-6 Debug Mode Logic 22-24
22-7 Debug Mode Reset Configuration 22-26
22-8 Asynchronous Clock Serial Communications 22-34
22-9 Synchronous Self Clock Serial Communication 22-35
22-10 Enabling Clock Mode Following Reset 22-36
22-11 Download Procedure Code Example 22-41
22-12 Slow Download Procedure Loop 22-41
22-13 Fast Download Procedure Loop 22-41

23-1 READI Functional Block Diagram 23-3
23-2 Enabling Program Trace out of System Reset 23-20
23-3 Functional Diagram of Pin Interface 23-22
23-4 Auxiliary Pin Packet Structure for Program Trace Indirect Branch Message 23-24
23-5 MSEI/MSEO Transfers 23-25
23-6 Transmission Sequence of messages. 23-31
23-7 READI Module Enabled 23-34
23-8 READI Module Disabled 23-35
23-9 RCPU Instruction Flow 23-37
23-10 Direct Branch Message Format 23-40
23-11 Indirect Branch Message Format 23-41
23-12 Indirect Branch Message Format with Compressed Code 23-41
23-13 Program Trace Correction Message Format 23-44
23-14 Error Message (Queue Overflow) Format 23-45
23-15 Direct Branch Synchronization Message Format 23-46
23-16 Indirect Branch Synchronization Message Format 23-46
23-17 Direct Branch Synchronization Message Format With Compressed Code 23-47
23-18 Indirect Branch Synchronization Message Format With Compressed Code 23-47
23-19 Relative Address Generation and Re-creation 23-48
23-20 VF State Decoding 23-49
23-21 BTM Encoding Flow 23-50
23-22 BTM Encoding Flow — Compressed Code Support 23-51
23-23 Direct Branch Message 23-52
23-24 Indirect Branch Message 23-52
23-25 Indirect Branch Message With Compressed Code 23-53
23-26 Program Trace Correction Message 23-53
23-27 Error Message (Program/Data/Ownership Trace Overrun) 23-54
23-28 Direct Branch Synchronization Message 23-54
23-29 Indirect Branch Synchronization Message 23-55

Figure Number		Page Number

23-30 Direct Branch Synchronization Message With Compressed Code 23-55
23-31 Indirect Branch Synchronization Message With Compressed Code 23-56
23-32 Data Write Message Format 23-57
23-33 Data Read Message Format 23-57
23-34 Data Write Synchronization Message Format 23-58
23-35 Data Read Synchronization Message Format 23-58
23-36 Error Message (Queue Overflow) Format 23-59
23-37 Data Trace Flow Diagram for Non-Pipelined Access 23-60
23-38 Date Write Message 23-62
23-39 Data Read Message 23-63
23-40 Data Write Synchronization Message 23-63
23-41 Data Read Synchronization Message 23-64
23-42 Error Message (Program/Data/Ownership Trace Overrun) 23-64
23-43 Read/Write Access Flow Diagram 23-65
23-44 Error Message (Read/Write Access Error) Format 23-71
23-45 Error Message (Invalid Message) Format 23-71
23-46 Error Message (Invalid Access Opcode) Format 23-72
23-47 Block Write Access 23-74
23-48 Block Read Access 23-74
23-49 Device Ready for Upload/Download Request Message 23-75
23-50 Upload Request Message 23-75
23-51 Download Request Message 23-76
23-52 Upload/Download Information Message 23-76
23-53 Error Message (Invalid Access Opcode) 23-77
23-54 Watchpoint Message Format 23-78
23-55 Error Message (Watchpoint Overrun) Format 23-78
23-56 Watchpoint Message 23-79
23-57 Error Message (Watchpoint Overrun) 23-79
23-58 Ownership Trace Message Format 23-80
23-59 Error Message Format 23-80
23-60 Ownership Trace Message 23-81
23-61 Error Message (Program/Data/Ownership Trace Overrun) 23-82
23-62 RCPU Development Access Multiplexing Between READI and BDM Pins 23-83
23-63 DSDI Message Format 23-84
23-64 DSDO Message Format 23-84
23-65 BDM Status Message Format 23-85
23-66 Error Message (Invalid Message) Format 23-85
23-67 RCPU Development Access Flow Diagram 23-88
23-68 RCPU Development Access Timing Diagram — Debug Mode Entry Out-of-Reset 23-90
23-69 Transmission Sequence of DSDx Data Messages 23-91
23-70 Error Message (Invalid Message) 23-91
23-71 DSDI Data Message (Assert Non-Maskable Breakpoint) 23-91

Figure Number		Page Number

23-72 DSDI Data Message (CPU Instruction — rfi) 23-92
23-73 DSDO Data Message (CPU Data Out) 23-92
23-74 Programming UC3F Flash via READI R/W Access (Sheet 1 of 2) 23-97

24-1 Pin Requirement on JTAG 24-1
24-2 Test Logic Block Diagram 24-3
24-3 TAP Controller State Machine 24-4
24-4 Bypass Register 24-25

D-1 TPU3 Memory Map D-1
D-2 PTA Parameters D-5
D-3 QOM Parameters D-7
D-4 TSM Parameters — Master Mode D-9
D-5 TSM Parameters — Slave Mode D-10
D-6 FQM Parameters D-12
D-7 UART Transmitter Parameters D-14
D-8 UART Receiver Parameters D-15
D-9 NITC Parameters D-17
D-10 COMM Parameters (Part 1 of 2) D-19
D-11 COMM Parameters (Part 2 of 2) D-20
D-12 HALLD Parameters D-21
D-13 MCPWM Parameters — Master Mode D-23
D-14 MCPWM Parameters — Slave Edge-Aligned Mode D-24
D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode D-25
D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode D-26
D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode D-27
D-18 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode D-28
D-19 FQD Parameters — Primary Channel D-30
D-20 FQD Parameters — Secondary Channel D-31
D-21 PPWA Parameters D-33
D-22 OC Parameters D-35
D-23 PWM Parameters D-37
D-24 DIO Parameters D-39
D-25 SPWM Parameters, Part 1 of 2 .....D-41
D-26 SPWM Parameters, Part 2 of 2 .....D-42
D-27 RWTPIN Parameters D-44
D-28 ID Parameters D-46
D-29 Two Possible SIOP Configurations D-47
D-30 SIOP Parameters D-49
D-31 SIOP Function Data Transition Example D-53

E-1 Option A Power-Up Sequence Without Keep-Alive Supply E-15

Figure Number		Page Number

E-2 Option A Power-Up Sequence With Keep-Alive Supply E-15
E-3 Option A Power-Down Sequence Without Keep-Alive Supply E-16
E-4 Option A Power-Down Sequence With Keep-Alive Supply E-16
E-5 Option B Power-Up Sequence Without Keep-Alive Supply E-17
E-6 Option B Power-Up Sequence With Keep-Alive Supply E-17
E-7 Option B Power-Down Sequence Without Keep-Alive Supply E-18
E-8 Option B Power-Down Sequence with Keep-Alive Supply E-18
E-9 Generic Timing Examples E-20
E-10 CLKOUT Pin Timing E-27
E-11 Synchronous Output Signals Timing E-28
E-12 Synchronous Active Pull-Up And Open Drain Outputs Signals Timing E-29
E-13 Synchronous Input Signals Timing E-30
E-14 Input Data Timing In Normal Case E-31
E-15 External Bus Read Timing (GPCM Controlled – ACS = ‘00’) E-32
E-16 External Bus Read Timing (GPCM Controlled – TRLX = ‘0’ ACS = ‘10’) E-33
E-17 External Bus Read Timing (GPCM Controlled – TRLX = ‘0’ ACS = ‘11’) E-34
E-18 External Bus Read Timing (GPCM Controlled – TRLX = ‘1’, ACS = ‘10’, ACS = ‘11’) E-35
E-19 Address Show Cycle Bus Timing E-36
E-20 Address and Data Show Cycle Bus Timing E-37
E-21 External Bus Write Timing (GPCM Controlled – TRLX = ‘0’, CSNT = ‘0’) E-38
E-22 External Bus Write Timing (GPCM Controlled – TRLX = ‘0’, CSNT = ‘1’) E-39
E-23 External Bus Write Timing (GPCM Controlled – TRLX = ‘1’, CSNT = ‘1’) E-40
E-24 External Master Read From Internal Registers Timing E-41
E-25 External Master Write To Internal Registers Timing E-42
E-26 Interrupt Detection Timing for External Level Sensitive Lines E-43
E-27 Interrupt Detection Timing for External Edge Sensitive Lines E-44
E-28 Debug Port Clock Input Timing E-45
E-29 Debug Port Timings E-46
E-30 Auxiliary Port Data Input Timing Diagram E-47
E-31 Auxiliary Port Data Output Timing Diagram E-47
E-32 Enable Auxiliary From $\overline{\text{RSTI}}$ E-48
E-33 Disable Auxiliary From $\overline{\text{RSTI}}$ E-48
E-34 Reset Timing – Configuration from Data Bus E-50
E-35 Reset Timing – Data Bus Weak Drive During Configuration E-51
E-36 Reset Timing – Debug Port Configuration E-52
E-37 JTAG Test Clock Input Timing E-53
E-38 JTAG Test Access Port Timing Diagram E-54
E-39 JTAG TRST Timing Diagram E-55
E-40 Boundary Scan (JTAG) Timing Diagram E-56

Figure Number		Page Number

E-41 QSPI Timing – Master, CPHA = 0 ..... E-59
E-42 QSPI Timing – Master, CPHA = 1 ..... E-60
E-43 QSPI Timing – Slave, CPHA = 0 ..... E-60
E-44 QSPI Timing – Slave, CPHA = 1 ..... E-61
E-45 TPU3 Timing E-62
E-46 MCPSM Enable to vs_pclk Pulse Timing Diagram E-64
E-47 MPWMSM Minimum Output Pulse Example Timing Diagram E-65
E-48 MCPSM Enable To MPWMO Output Pin Rising Edge Timing Diagram E-66
E-49 MPWMSM Enable To MPWMO Output Pin Rising Edge Timing Diagram E-66
E-50 MPWMSM Interrupt Flag To MPWMO Output Pin Falling Edge Timing Diagram E-67
E-51 MMCSM Minimum Input Pin (Either Load Or Clock) Timing Diagram E-68
E-52 MMCSM Clock Pin To Counter Bus Increment Timing Diagram E-68
E-53 MMCSM Load Pin To Counter Bus Reload Timing Diagram E-69
E-54 MMCSM Prescaler Clock Select To Counter Bus Increment Timing Diagram E-69
E-55 MDASM Minimum Input Pin Timing Diagram E-70
E-56 MDASM Input Pin To Counter Bus Capture Timing Diagram E-71
E-57 MDASM Input Pin to MDASM Interrupt Flag Timing Diagram E-71
E-58 MDASM Minimum Output Pulse Width Timing Diagram E-71
E-59 Counter Bus to MDASM Output Pin Change Timing Diagram E-72
E-60 Counter Bus to MDASM Interrupt Flag Setting Timing Diagram E-72
E-61 MSASM Minimum Input Pin Timing Diagram E-73
E-62 MSASM Input Pin To Counter Bus Capture Timing Diagram E-74
E-63 MSASM Input Pin to MSASM Interrupt Flag Timing Diagram E-74
E-64 MSASM Minimum Output Pulse Width Timing Diagram E-74
E-65 Counter Bus to MSASM Output Pin Change Timing Diagram E-75
E-66 Counter Bus to MSASM Interrupt Flag Setting Timing Diagram E-75
E-67 MPIOSM input pin to MPIOSM_DR (Data Register) Timing Diagram E-76

LIST OF TABLES

Table Number | **Page Number**

1-1 Differences Between MPC555 / MPC556 and MPC565 / MPC566 1-7

2-1 MPC565 / MPC566 Pad Function Summary 2-1
2-2 MPC565 / MPC566 Pin Sharing Description 2-3
2-3 PDMCR Bit Descriptions 2-4
2-4 PDMCR2 Bit Descriptions 2-6
2-5 DLCMD2 / QSMCM_B SCI2 Pin MUX Control 2-15
2-6 Pin Reset State 2-29
2-7 Functional Pad Types 2-36
2-8 Other Pad Types 2-38
2-9 Target DC Electrical Characteristics 2-39
2-10 Pin Names and Abbreviations 2-42

3-1 RCPU Execution Units 3-5
3-2 Supervisor-Level SPRs 3-9
3-3 Development Support SPRs 3-11
3-4 FPSCR Bit Categories 3-13
3-5 FPSCR Bit Descriptions 3-14
3-6 Floating-Point Result Flags in FPSCR 3-15
3-7 Bit Settings for CR0 Field of CR 3-16
3-8 Bit Settings for CR1 Field of CR 3-17
3-9 CR*n* Field Bit Settings for Compare Instructions 3-17
3-10 Integer Exception Register Bit Descriptions 3-18
3-11 Time Base Field Definitions (Read Only) 3-20
3-12 Machine State Register Bit Descriptions 3-21
3-13 Floating-Point Exception Mode Bits 3-22
3-14 Time Base Field (Write Only) Bit Descriptions 3-23
3-15 Uses of SPRG0–SPRG3 3-25
3-16 Processor Version Register Bit Descriptions 3-26
3-17 EIE, EID, AND NRI Registers 3-26
3-18 FPECR Bit Descriptions 3-27
3-19 Instruction Set Summary 3-29
3-20 MPC565 / MPC566 Exception Classes 3-34
3-21 Exception Vector Offset Table 3-36
3-22 Instruction Latency and Blockage 3-38
3-23 Floating-Point Exception Mode Encoding 3-43

4-1 Exception Addresses Mapping 4-16
4-2 Exception Relocation Page Offset 4-17
4-3 BBC SPRs 4-24
4-4 BBCMCR BBC Module Configuration Register Bit Descriptions 4-25
4-5 MI_RBA[0:3] Registers Bit Descriptions 4-27
4-6 MI_RA[0:3] Registers Bits Description 4-28

Table Number		Page Number

4-7 Region Size Programming Possible Values 4-29
4-8 MI_GRA Global Region Attribute Register Bits Description 4-30
4-9 EIBADR External Interrupt Relocation Table Base Address Register Bit Descriptions 4-31
4-10 DCCR0-DCCR15 Decompressor Class Configuration Registers Bit Descriptions 4-33
4-11 Instruction Layout Encoding 4-34

5-1 USIU Address Map 5-3
5-2 USIU Special Purpose Registers 5-8
5-3 MPC565 / MPC566 Address Format for SPR Access From External Bus 5-8

6-1 USIU Pins Multiplexing Control 6-4
6-2 SGPIO Configuration 6-8
6-3 Priority Of interrupt Sources — Regular Operation 6-12
6-4 Priority of Interrupt Sources — Enhanced Operation 6-13
6-5 Interrupt Latency Estimation for Three Typical Cases 6-17
6-6 Decrementer Time-Out Periods 6-20
6-7 SIUMCR Bit Descriptions 6-25
6-8 Debug Pins Configuration 6-27
6-9 Debug Port Pins Configuration 6-27
6-10 General Pins Configuration 6-27
6-11 Single-Chip Select Field Pin Configuration 6-27
6-12 Multi-Level Reservation Control Pin Configuration 6-28
6-13 IMMR Bit Descriptions 6-29
6-14 EMCR Bit Descriptions 6-30
6-15 SYPCR Bit Descriptions 6-37
6-16 SWSR Bit Descriptions 6-38
6-17 TESR Bit Descriptions 6-38
6-18 TBSCR Bit Descriptions 6-41
6-19 RTCSC Bit Descriptions 6-42
6-20 PISCR Bit Descriptions 6-43
6-21 PITC Bit Descriptions 6-44
6-22 PIT Bit Descriptions 6-44
6-23 SGPIODT1 Bit Descriptions 6-45
6-24 SGPIODT2 Bit Descriptions 6-46
6-25 SGPIOCR Bit Descriptions 6-46
6-26 Data Direction Control 6-47

7-1 Reset Action Taken For Each Reset Cause 7-4
7-2 Reset Configuration Word and Data Corruption/Coherency 7-5
7-3 Reset Status Register Bit Descriptions 7-5
7-4 Reset Configuration Options 7-7
7-5 RCW Bit Descriptions 7-12

8-1 Reset Clocks Source Configuration 8-9

Table Number		Page Number

8-2 TMBCLK Divisions 8-9
8-3 Status of Clock Source 8-15
8-4 Power Mode Control Bit Settings 8-16
8-5 Power Mode Descriptions 8-16
8-6 Power Mode Wake-Up Operation 8-17
8-7 Power Supplies 8-20
8-8 KAPWR Registers and Key Registers 8-25
8-9 SCCR Bit Descriptions 8-30
8-10 COM and CQDS Bits Functionality 8-33
8-11 PLPRCR Bit Descriptions 8-34
8-12 COLIR Bit Descriptions 8-36
8-13 VSRMCR Bit Descriptions 8-37

9-1 MPC565 / MPC566 SIU Signals 9-4
9-2 Data Bus Requirements For Read Cycles 9-30
9-3 Data Bus Contents for Write Cycles 9-30
9-4 Priority Between Internal and External Masters over External Bus 9-34
9-5 Burst Length and Order 9-37
9-6 BURST/TSIZE Encoding 9-37
9-7 Address Type Pins 9-38
9-8 Address Types Definition 9-38
9-9 Termination Signals Protocol 9-48

10-1 Timing Attributes Summary 10-7
10-2 Programming Rules for Strobes Timing 10-20
10-3 Boot Bank Fields Values After Hard Reset 10-21
10-4 Write Enable/Byte Enable Signals Function 10-22
10-5 Memory Controller Address Map 10-28
10-6 MSTAT Bit Descriptions 10-29
10-7 BR[0] – BR[3] Bit Descriptions 10-30
10-8 OR[0] – OR[3] Bit Descriptions 10-32
10-9 DMBR Bit Descriptions 10-34
10-10 DMOR Bit Descriptions 10-35

11-1 DMPU Registers 11-6
11-2 Reservation Snoop Support 11-9
11-3 L2U_MCR LSHOW Modes 11-10
11-4 L2U Show Cycle Support Chart 11-12
11-5 L2U (PPC) Register Decode 11-13
11-6 Hex Address For SPR Cycles 11-13
11-7 L2U_MCR Bit Descriptions 11-14
11-8 L2U_RBAx Bit Descriptions 11-15
11-9 L2U_RAx Bit Descriptions 11-16
11-10 L2U_GRA Bit Descriptions 11-17

12-1 STOP and HSPEED Bit Functionality 12-2

Table Number		Page Number

12-2 Bus Cycles and System Clock Cycles 12-3
12-3 ILBS Signal functionality 12-5
12-4 IRQMUX Functionality 12-5
12-5 UIMB Interface Register Map 12-7
12-6 UMCR Bit Descriptions 12-8
12-7 UIPEND Bit Descriptions 12-9

13-1 QADC64E_A Address Map 13-5
13-2 QADC64E_B Address Map 13-6
13-3 Multiplexed Analog Input Channels 13-8
13-4 Analog Input Channels 13-9
13-5 QADCMCR Bit Descriptions 13-10
13-6 QADC64E Bus Error Response 13-12
13-7 QADCMCR Bit Descriptions 13-14
13-8 QACR0 Bit Descriptions 13-17
13-9 Prescaler FSYSCLK Divide-by Values 13-17
13-10 QACR1 Bit Descriptions 13-18
13-11 Queue 1 Operating Modes 13-19
13-12 QACR2 Bit Descriptions 13-20
13-13 Queue 2 Operating Modes 13-22
13-14 QASR0 Bit Descriptions 13-23
13-15 Pause Response 13-27
13-16 Queue Status 13-28
13-17 QASR1 Bit Descriptions 13-30
13-18 CCW Bit Descriptions 13-33
13-19 Input Sample Times 13-34
13-20 Multiplexed Channel Assignments and Pin Designations 13-35
13-21 QADC64E Clock Programmability 13-54
13-22 Trigger Events 13-58
13-23 Status Bits 13-58
13-24 External Circuit Settling Time to 1/2 LSB (10-Bit Conversions) 13-80
13-25 Error Resulting From Input Leakage (IOFF) 13-81

14-1 DLCMD2 / QSMCM_B SCI2 Pin MUX Control 14-2
14-2 QSMCM Pins on QSMCM_A and QSMCM_B 14-3
14-3 QSMCM_A and QSMCM_B Register Map 14-4
14-4 QSMCM_A Global Registers 14-6
14-5 QSMCM_B Global Registers 14-6
14-6 Interrupt Levels 14-8
14-7 QSMCMMCR Bit Descriptions 14-9
14-8 QDSCI_IL Bit Descriptions 14-10
14-9 QSPI_IL Bit Descriptions 14-10
14-10 QSMCM Pin Control Registers 14-10
14-11 Effect of DDRQS on QSPI Pin Function 14-11
14-12 QSMCM_A Pin Functions 14-12
14-13 QSMCM_B Pin Functions 14-12

Table Number		Page Number

14-14 PQSPAR Bit Descriptions 14-13
14-15 DDRQS Bit Descriptions 14-14
14-16 QSPI Register Map 14-17
14-17 SPCR0 Bit Descriptions 14-18
14-18 Bits Per Transfer 14-18
14-19 SPCR1 Bit Descriptions 14-19
14-20 SPCR2 Bit Descriptions 14-20
14-21 SPCR3 Bit Descriptions 14-21
14-22 SPSR Bit Descriptions 14-22
14-23 Command RAM Bit Descriptions 14-24
14-24 QSPI Pin Functions 14-25
14-25 Example SCK Frequencies with a 40-MHz IMB3 clock 14-36
14-26 SCI Registers 14-45
14-27 SCCxR0 Bit Descriptions 14-46
14-28 SCCxR1 Bit Descriptions 14-47
14-29 SCxSR Bit Descriptions 14-49
14-30 SCxDR Bit Descriptions 14-51
14-31 SCI Pin Functions 14-51
14-32 Serial Frame Formats 14-52
14-33 Examples of SCIx Baud Rates 14-53
14-34 Effect of Parity Checking on Data Size 14-53
14-35 QSCI1CR Bit Descriptions 14-61
14-36 QSCI1SR Bit Descriptions 14-62

15-1 DLCMD2 Requirements 15-2
15-2 Digital Filter Output 15-8
15-3 Receive Windows 15-12
15-4 Transmit Windows 15-12
15-5 CPU/DLCMD2 Transfers 15-23
15-6 DLCMD2 Memory Map 15-27
15-7 MCR Bit Descriptions 15-28
15-8 Freeze Bit Field Description 15-29
15-9 Interrupt Pending (IPR) Bit Descriptions 15-30
15-10 Interrupt Level (ILR) Bit Descriptions 15-31
15-11 Interrupt Levels 15-31
15-12 Interrupt Vector (IVR) Bit Descriptions 15-32
15-13 Symbol Timing Control and Pre-Scaler Bit Descriptions 15-32
15-14 DLCMD2 Pre-Scaler Rate Selection 15-33
15-15 Symbol Timing Data (SDATA) Bit Descriptions 15-34
15-16 Timing Parameter Table 15-35
15-17 Transmit Command Register (CMD) Bit Descriptions 15-36
15-18 General Commands 15-37
15-19 Type and Destination of Accompanying Byte Commands 15-38
15-20 RxFIFO Commands 15-39
15-21 Command Load Sequences 15-40

Table Number		Page Number

15-22 Transmit Data Register (TDATA) Bit Descriptions 15-41
15-23 Receive Status Register (STAT) Bit Descriptions 15-42
15-24 RxFIFO Status 15-42
15-25 Data Link Idle Status 15-43
15-26 Transmitter Shorted Status 15-43
15-27 TxFIFO Underrunning Status 15-43
15-28 TxFIFO Status 15-43
15-29 Receive Data Register (RDATA) Bit Descriptions 15-44
15-30 Receive Error Status 15-45
15-31 RxFIFO Overrun Status 15-46
15-32 Transmitter Action Status 15-46
15-33 IFR Bit Status 15-46
15-34 IFR With/Without a CRC Bit Status 15-46
15-35 Error Code Status1, 2, 3 15-47
15-36 Interrupt Operations 15-52
15-37 Interrupt Levels 15-53
15-38 IFR Aborted Conditions 15-56
15-39 Signal Names 15-59

16-1 Common Extended/Standard Format Frames 16-5
16-2 Message Buffer Codes for Receive Buffers 16-5
16-3 Message Buffer Codes for Transmit Buffers 16-5
16-4 Extended Format Frames 16-6
16-5 Standard Format Frames 16-6
16-6 Receive Mask Register Bit Values 16-8
16-7 Mask Examples for Normal/Extended Messages 16-8
16-8 Example System Clock, CAN Bit Rate and S-Clock Frequencies 16-9
16-9 Interrupt Levels 16-19
16-10 TouCAN Register Map 16-21
16-11 TCNMCR Bit Descriptions 16-23
16-12 CANICR Bit Descriptions 16-24
16-13 CANCTRL0 Bit Descriptions 16-25
16-14 Rx MODE[1:0] Configuration 16-25
16-15 Transmit Pin Configuration 16-26
16-16 CANCTRL1 Bit Descriptions 16-27
16-17 PRESDIV Bit Descriptions 16-28
16-18 CANCTRL2 Bit Descriptions 16-28
16-19 TIMER Bit Descriptions 16-29
16-20 RxGMSKHI, RxGMSKLO Bit Descriptions 16-30
16-21 ESTAT Bit Descriptions 16-31
16-22 Transmit Bit Error Status 16-32
16-23 Fault Confinement State Encoding 16-32
16-24 IMASK Bit Descriptions 16-32
16-25 IFLAG Bit Descriptions 16-33
16-26 RxECTR, TxECTR Bit Descriptions 16-33

Table Number		Page Number

17-1 MIOS14 Configuration Description 17-6
17-2 MIOS14SR0 Bit Descriptions 17-27
17-3 MIOS14ER0 Bit Descriptions 17-27
17-4 MIOS14PR0 Bit Descriptions 17-28
17-5 MIOS14SR1 Bit Descriptions 17-28
17-6 MIOS14ER1 Bit Descriptions 17-29
17-7 MIOS14PR1 Bit Descriptions 17-29
17-8 MIOS14LVL0 Bit Descriptions 17-30
17-9 MIOS14LVL1 Bit Descriptions 17-30
17-10 MBISM Registers 17-31
17-11 MIOS14TPCR Bit Descriptions 17-32
17-12 MIOS14VECT Bit Descriptions 17-33
17-13 MIOS14VNR Bit Descriptions 17-33
17-14 MIOS14MCR Bit Descriptions 17-34
17-15 MCPSMSCR Bit Descriptions 17-36
17-16 Clock Prescaler Setting 17-37
17-17 MMCSM Address Map 17-41
17-18 MMCSMCNT Bit Descriptions 17-42
17-19 MMCSMML Bit Descriptions 17-43
17-20 MMCSMSCR Bit Descriptions 17-43
17-21 MMCSMCNT Edge Sensitivity 17-44
17-22 MMCSMCNT Clock Signal 17-44
17-23 Prescaler Values 17-44
17-24 MDASM Modes Of Operation 17-48
17-25 MDASM PWM Example Output Frequencies/Resolutions at f[SYS] = 40 MHz 17-57
17-26 MDASM Address Map 17-58
17-27 MDASMAR Bit Descriptions 17-60
17-28 MDASMBR Bit Descriptions 17-61
17-29 MDASMSCR Bit Descriptions 17-63
17-30 PWM Pulse/Frequency Ranges (in Hz) Using /1 Option (40 MHz) 17-70
17-31 PWM Pulse/frequency Ranges (in Hz) Using /256 Option (40 MHz) 17-71
17-32 MPWMSM Address Map 17-73
17-33 MPWMPERR Bit Descriptions 17-75
17-34 MPWMPULR Bit Descriptions 17-76
17-35 MPWMCNTR Bit Descriptions 17-76
17-36 MPWMSCR Bit Descriptions 17-77
17-37 PWMSM Output Pin Polarity Selection 17-78
17-38 Prescaler Values 17-79
17-39 MPIOSM I/O Pin Function 17-80
17-40 MPIOSMDR Bit Descriptions 17-82
17-41 MPIOSMDDR Bit Descriptions 17-82
17-42 MRTCSMFRCH Bit Descriptions 17-92
17-43 MRTCSMFRCL Bit Descriptions 17-93
17-44 MRTCPR Bit Descriptions 17-93
17-45 MRTCSMSCR Bit Descriptions 17-94

Table Number		Page Number

17-46 Interrupt Rate Selection 17-95

18-1 Enhanced TCR1 Prescaler Divide Values 18-6
18-2 TCR1 Prescaler Values 18-6
18-3 TCR2 Counter Clock Source 18-7
18-4 TCR2 Prescaler Control 18-8
18-5 TPU3 Register Map 18-9
18-6 TPUMCR Bit Descriptions 18-12
18-7 DSCR Bit Descriptions 18-14
18-8 DSSR Bit Descriptions 18-15
18-9 TICR Bit Descriptions 18-16
18-10 CIER Bit Descriptions 18-16
18-11 CFSRx Bit Descriptions 18-17
18-12 HSQRx Bit Descriptions 18-18
18-13 HSSRx Bit Descriptions 18-19
18-14 CPRx Bit Descriptions 18-20
18-15 Channel Priorities 18-20
18-16 CISR Bit Descriptions 18-20
18-17 TPUMCR2 Bit Descriptions 18-21
18-18 Entry Table Bank Location 18-22
18-19 System Clock Frequency/Minimum Guaranteed Detected Pulse 18-22
18-20 TPUMCR3 Bit Descriptions 18-23
18-21 Parameter RAM Address Offset Map 18-24

19-1 DPTRAM Register Map 19-3
19-2 DPTMCR Bit Settings 19-4
19-3 RAMBAR Bit Settings 19-5

20-1 UC3F External Interface Signals 20-5
20-2 UC3F Register Programmer's Model 20-6
20-3 UC3FMCR Bit Descriptions 20-8
20-4 UC3FMCRE Bit Descriptions 20-11
20-5 UC3FCTL Bit Descriptions 20-14
20-6 RCW Bit Descriptions 20-20
20-7 Program Interlock State Descriptions 20-27
20-8 Erase Interlock State Descriptions 20-30

21-1 Priorities of Overlay Regions 21-13
21-2 CALRAM A and B Control Registers 21-15
21-3 CRAMMCR Bit Descriptions 21-16
21-4 CRAMMCR Privilege Bit Assignment for 8-Kbyte Array Blocks 21-18
21-5 CRAMMCR Bit Descriptions 21-19
21-6 RGN_SIZE Encoding 21-19
21-7 CRAMOVLCR Bit Descriptions 21-20

22-1 VF Pins Instruction Encodings 22-3

Table Number		Page Number

22-2 VF Pins Queue Flush Encodings 22-3
22-3 VFLS Pin Encodings 22-4
22-4 Detecting the Trace Buffer Start Point 22-6
22-5 Fetch Show Cycles Control 22-7
22-6 Instruction Watchpoints Programming Options 22-16
22-7 Load/Store Data Events 22-17
22-8 Load/Store Watchpoints Programming Options 22-18
22-9 The Check Stop State and Debug Mode 22-28
22-10 Trap Enable Data Shifted into Development Port Shift Register 22-37
22-11 Debug Port Command Shifted Into Development Port Shift Register 22-37
22-12 Status / Data Shifted Out of Development Port Shift Register 22-38
22-13 Debug Instructions / Data Shifted Into Development Port Shift Register 22-39
22-14 Development Support Programming Model 22-43
22-15 Development Support Registers Read Access Protection 22-44
22-16 Development Support Registers Write Access Protection 22-44
22-17 CMPA-CMPD Bit Descriptions 22-44
22-18 CMPE-CMPF Bit Descriptions 22-45
22-19 BAR Bit Descriptions 22-45
22-20 CMPG-CMPH Bit Descriptions 22-45
22-21 ICTRL Bit Descriptions 22-46
22-22 ISCT_SER Bit Descriptions 22-47
22-23 LCTRL1 Bit Descriptions 22-48
22-24 LCTRL2 Bit Descriptions 22-49
22-25 Breakpoint Counter A Value and Control Register (COUNTA) 22-51
22-26 Breakpoint Counter B Value and Control Register (COUNTB) 22-51
22-27 ECR Bit Descriptions 22-52
22-28 DER Bit Descriptions 22-53

23-1 READI Reset Configuration Options 23-4
23-2 Public Messages 23-5
23-3 Vendor-Defined Messages 23-6
23-4 Terms and Definitions 23-6
23-5 READI_OTR Bit Descriptions 23-8
23-6 Tool Mapped Register Space 23-9
23-7 READI_DID Bit Descriptions 23-10
23-8 READI_DC Bit Descriptions 23-10
23-9 RCPU Development Access Modes 23-11
23-10 READI_UBA Bit Descriptions 23-12
23-11 READI_RWA Read/Write Access Bit Descriptions 23-14
23-12 READI_UDI Bit Descriptions 23-15
23-13 Read Access Status 23-15
23-14 Write Access Status 23-16
23-15 RWD Field Configuration 23-16
23-16 READI_DTA 1 AND 2 Bit Descriptions 23-17
23-17 Data Trace Values 23-17
23-18 Description of READI Pins 23-21

Table Number		Page Number

23-19 MSEI/MSEO protocol 23-23
23-20 Public Messages Supported 23-26
23-21 Error Codes 23-28
23-22 Vendor-Defined Messages Supported 23-28
23-23 Message Field Sizes 23-29
23-24 Indirect Branch Message 23-32
23-25 Direct Branch Message 23-33
23-26 READI Reset Configuration Options 23-33
23-27 VF Instruction Type Encoding 23-39
23-28 VF Queue Flush Information 23-39
23-29 VFLS History Buffer Flush Encoding 23-40
23-30 Program Trace Correction Due to a Mispredicted Branch 23-42
23-31 Program Trace Correction Due to an Exception 23-43
23-32 Special L-bus Case Handling 23-61
23-33 Throughput Comparison for FPM and RPM MDO/MDI Configurations 23-73
23-34 Watchpoint Source 23-78
23-35 Power Management Mechanism Overview 23-93

24-1 Boundary Scan Bit Definition 24-5
24-2 Instruction Decoding 24-23

A-1 SPR (Special Purpose Registers) A-2
A-2 UC3F Flash Array A-4
A-3 DECRAM SRAM Array A-4
A-4 BBC (Burst Buffer Controller Module) A-4
A-5 USIU (Unified System Interface Unit) A-5
A-6 CDR3 Flash Control Registers EEPROM (UC3F) A-8
A-7 DPTRAM AB and C Control Registers A-9
A-8 DLCMD2 (Data Link Controller Module) A-9
A-9 DPTRAM Memory Arrays A-10
A-10 Time Processor Unit 3 A and B (TPU3 A and B) A-10
A-11 QADC64E A and B (Queued Analog-to-Digital Converter) A-13
A-12 QSMCM A and B (Queued Serial Multi-Channel Module) A-15
A-13 Time Processor Unit 3 C (TPU3_C)
(See **Table A-10** for Bit Descriptions) A-17
A-14 MIOS14 (Modular Input/Output Subsystem) A-19
A-15 TouCAN A, B and C (CAN 2.0B Controller) A-24
A-16 UIMB (U-Bus to IMB Bus Interface) A-29
A-17 CALRAM_A and CALRAM_B Control Registers A-29
A-18 CALRAM_B and CALRAM_A Array A-31

D-1 Bank 0 Functions D-2
D-2 Bank 1 Functions D-3
D-3 QOM Bit Encoding D-6
D-4 SIOP Function Valid CHAN_Control Options D-50
D-5 SIOP State Timing D-52

Table Number		Page Number

E-1 Absolute Maximum Ratings E-2
E-2 Thermal Characteristics E-4
E-3 ESD Protection E-7
E-4 DC Electrical Characteristics E-7
E-5 Oscillator and PLL E-12
E-6 Array Program and Erase Characteristics E-12
E-7 CENSOR Cell Program and Erase Characteristics E-13
E-8 Module Life E-13
E-9 MPC565 / MPC566 Power Supply Pin Groups E-14
E-10 Bus Operation Timing E-21
E-11 Interrupt Timing E-43
E-12 Debug Port timing E-45
E-13 AC Electrical Characteristics E-47
E-14 RESET Timing E-49
E-15 JTAG timing E-53
E-16 QADC64E Conversion Characteristics (Enhanced Mode Operation) E-57
E-17 QSPI Timing E-58
E-18 GPIO Timing E-61
E-19 TPU3 Timing E-62
E-20 TouCAN Timing E-63
E-21 MCPSM Timing Characteristics E-63
E-22 MPWMSM Timing Characteristics E-65
E-23 MMCSM Timing Characteristics E-67
E-24 MMCSM Counter Bus Reload To Interrupt Flag Setting Timing Diagram E-69
E-25 MDASM Timing Characteristics E-70
E-26 MSASM Timing Characteristics E-73
E-27 MPIOSM Timing Characteristics E-76

Table Number	Page Number

PREFACE

This manual describes the capabilities, operation, and functions of the MPC565 / MPC566 microcontroller unit. Documentation for the modular microcontroller family follows the modular construction of the devices in the product line. Each device has a comprehensive user's manual which provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Please refer to the list of documents below for further information.

Boxed sections appear throughout this manual. These boxes designate optional features that are only available on the MPC566.

Audience

This manual is intended for system software and hardware developers and applications programmers who want to develop products for MPC565 / MPC566. It is assumed that the reader understands operating systems, microprocessor and microcontroller system design.

Additional Reading

As a complement to the present document, the following Motorola documents provide an in-depth functional description of the MPC565 / MPC566 modules:

- QSM Reference Manual (**QSMRM/AD**)
- TPU documentation (**TPULITPAK/D**, including the **TPURM/AD**)
- MPC555/MPC556 User's Manual (**MPC555/MPC556UM/AD**)
- RCPU Reference Manual (**RCPURM/AD**)

Conventions

This document uses the following notational conventions:

ACTIVE_HIGH names for signals that are active high are shown in uppercase text without an overbar. Signals that are active high are referred to as asserted when they are high and negated when they are low.

ACTIVE_LOW shows a bar over a signal name indicates that the signal is active low. Active-low signals are referred to as asserted (active) when they are low and negated when they are high.

0x0F shows hexadecimal numbers

0b0011 shows binary numbers

mnemonics — Instruction mnemonics are shown in lowercase bold.

italics — Italics indicate variable command parameters, for example, **bcctr***x*.

A **specific bit or signal** within a range is referred to by mnemonic and number. For example, ADDR15 is bit 15 of the address bus. A **range of bits or signals** is referred to by mnemonic and the numbers that define the range. For example, DATA[7:0] form the low byte of the data bus.

REG[FIELD] are abbreviations or acronyms for registers are shown in uppercase text. Specific bit fields or ranges are shown in brackets.

x in certain contexts, such as a signal encoding, indicates a don't care. For example, if a field is binary encoded 0bx001, the state of the first bit is a don't care.

NOTE:

Throughout this manual references to 3 V refer to the nominal supply voltage of 3.3 V.

Nomenclature

Logic level one is the voltage that corresponds to Boolean true (1) state.

Logic level zero is the voltage that corresponds to Boolean false (0) state.

To **set** a bit or bits means to establish logic level one on the bit or bits.

To **clear** a bit or bits means to establish logic level zero on the bit or bits.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

Asserted means that a signal is in active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

Negated means that an asserted signal changes logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

A range of mnemonics is referred to by mnemonic and the numbers that define the range. For example, VBR[4:0] are bits four to zero of the vector base register.

Symbols

Symbols and Operators

Symbol	Function
+	Addition
–	Subtraction (two's complement) or negation
*	Multiplication
/	Division
>	Greater
<	Less
=	Equal
Š	Equal or greater
£	Equal or less
¼	Not equal
•	AND
;	Inclusive OR (OR)
Ý	Exclusive OR (EOR)
NOT	Complementation
:	Concatenation
?	Transferred
¤	Exchanges
±	Sign bit; also used to show tolerance
«	Sign extension
%	Binary value
$	Hexadecimal value

References

It is recommended to use the Sematech Official Dictionary and the Reference Guide to Letter Symbols for Semiconductor Devices by the JEDEC Council/Electronics Industries Association as references for terminology and symbology.

SECTION 1
OVERVIEW

The purpose of this section is to give an overview of the MPC565 / MPC566 part, including the features, module mix, pins, address map, package, and electrical characteristics. The module mix of the part is shown in the block diagram and the text.

The remaining sections of the document describe the modules which are used on the chip. The MPC565 / MPC566 has duplicates of several modules, including TPU3, TouCAN, QSMCM, QADC64, CALRAM, UC3F, DPTRAM.

1.1 Introduction

The MPC565 / MPC566 is a member in the Motorola MPC500 PowerPC^{TM} RISC Microcontroller family. As shown in **Figure 1-1**, it is composed of:

— A PowerPC core with FPU and BBC
— 36 Kbytes of static RAM (two CALRAM modules)
  • Eight Kbytes of normal access or overlay access (sixteen 512-byte regions)
    — (four Kbytes in CALRAM_A, four Kbytes in CALRAM_B)
— One Mbyte of flash memory (UC3F)
— Unified system integration unit (USIU), a flexible memory controller, an improved interrupt controller
— Three time processor units (TPU3)
  • TPU3_A and TPU3_B are connected to DPTRAM_AB (six Kbytes)
  • TPU3_C is connected to DPTRAM_C (four Kbytes)
— A 22-timer channel modular I/O system (MIOS14)
  • Same as MIOS1 plus a real-time clock sub-module (MRTCSM), four counter sub-modules (MCSM), and four PWM sub-modules (MPWMSM)
— Three TouCAN modules (TOUCAN_A, TOUCAN_B, TOUCAN_C)
— Two enhanced queued analog system (QADC64E_A, QADC64E_B) with analog multiplexors (AMUX) for 40 total analog channels. These modules are configured so each module can access all 40 of the analog inputs to the part.
— Two queued serial multi-channel module (QSMCM_A, QSMCM_B), each of which contains a queued serial peripheral interface (QSPI) and two serial controller interfaces (SCI/UART)
— A J1850 (DLCMD2) communications module
— A NEXUS debug port (class 3) – IEEE-ISTO 5001-1999
— JTAG and background debug mode (BDM)

1.2 Block Diagram

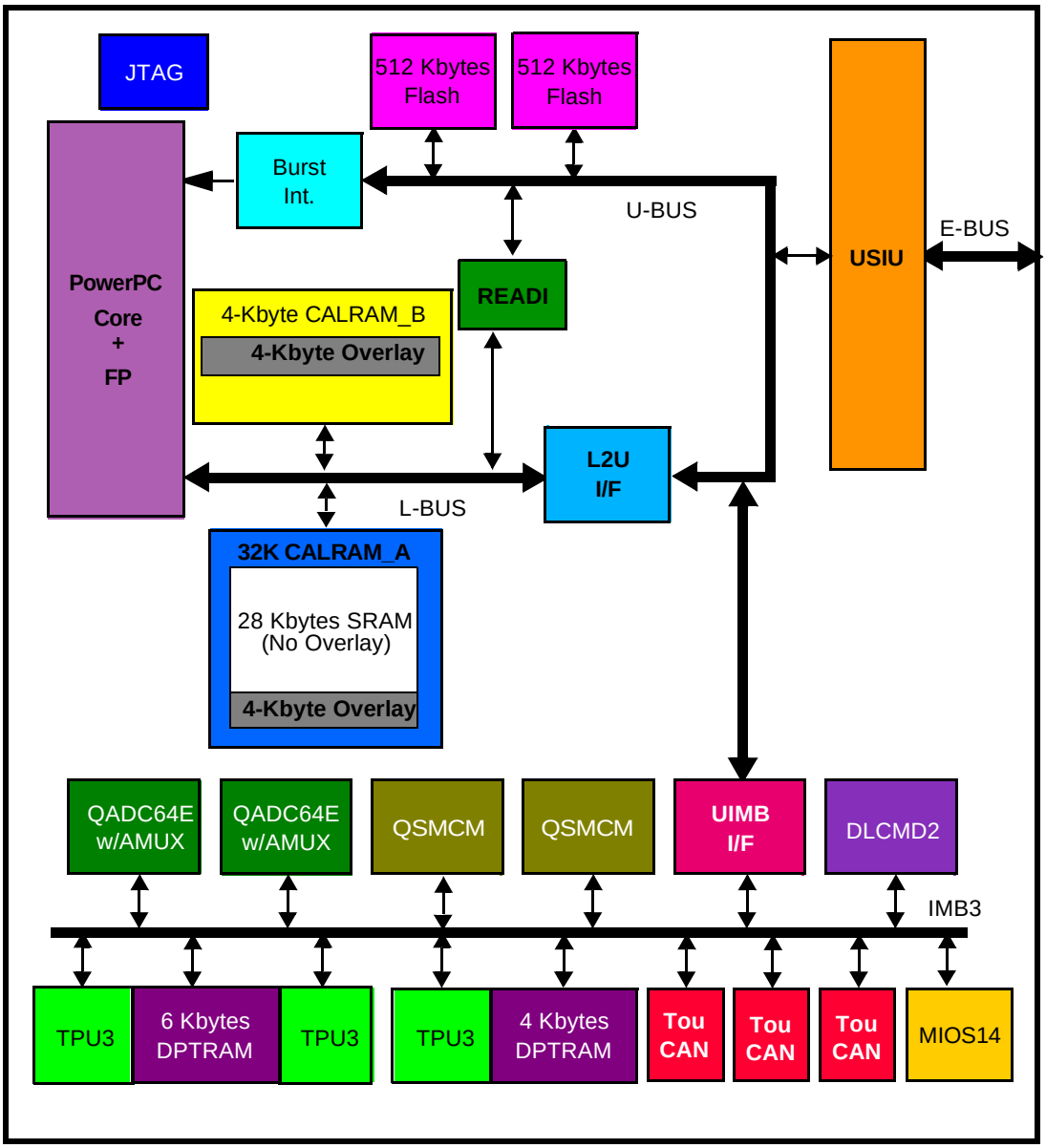


Figure 1-1 MPC565 / MPC566 Block Diagram

1.2.1 Detailed Feature List

The MPC565 / MPC566 key features are as follows. The information inside boxes are optional features.

- 40 MHz / 56 MHz operation
- 56 MHz operation is available as an option.
  - — -40° – 125° C ambient temperature

— 2.6 V ± 0.1 V external bus
  - External bus is compatible with external memory devices operating from 2.5 V to 3.4 V.
  - Extended voltage range (2.7 – 3.4 V) degrades data drive timing by 1.1 ns on date writes.
— 2.6 ± 0.1 V internal logic
— 5-V I/O (5.0 ± 0.25 V)

- High performance CPU system
  — High performance core
    - PowerPC single issue integer core
    - Precise exception model
    - Floating point
    - Code compression supported on the MPC566
      — Compression reduces usage of internal or external flash memory
      — Compression optimized for automotive (non-cached) applications
      — New compression scheme increases compression performance to 40% – 50% compression
      — 4-Kbyte static DECRAM can be used as memory if Compression is not used.
    - Extensive system development support
      — On-chip watchpoints and breakpoints
      — Program flow tracking
      — Background debug mode (BDM)
  — MPC500 PowerPC system interface (USIU, BBC, L2U)
    - Periodic interrupt timer, bus monitor, clocks, decrementer and time base
    - Clock synthesizer, power management, reset controller
    - External bus tolerates 5-V inputs, provides 3.3-V outputs
    - Enhanced interrupt controller supports a separate interrupt vector for up to eight external and 40 Internal interrupts
    - IEEE 1149.1 JTAG test access port
    - Bus supports multiple master designs
    - Flexible memory protection units in BBC (IMPU) and L2U (DMPU)
    - Flexible chip selects via memory controller
      — 24-bit address and 32-bit data buses
      — Four- to 16-Mbyte (data) or 4-Gbyte (instruction) region size support
      — Four-beat transfer bursts, two-clock minimum bus transactions
      — Use with SRAM, EPROM, flash and other peripherals
      — Byte selects or write enables
      — 32-bit address decodes with bit masks
      — Four instruction regions
      — Four data regions
    - Default attributes available in one global entry
    - Attribute support for speculative accesses
    - Exception vector table relocation features allow exception table to be relocated to following locations:
      — 0x0000 0100 (normal PowerPC exception table location)
      — 0x0001 0000 (0 + 64 Kbytes; second page of internal flash)

— Second internal flash module
— Internal SRAM
— 0x0FFF_0100 (external memory space; normal PowerPC exception table location)
• USIU supports dual-mapping of flash to move part of internal flash memory to external bus for development

— Fully static design
— Four major power saving modes
• On, doze, nap and sleep
— One Mbyte flash
• Two UC3F modules, 512 Kbytes each
• Page mode read
• Block (64-Kbyte) erasable
• External 4.75- to 5.25-V V_{PP} program and erase power supply
— 36-Kbyte static RAM (CALRAM)
• Composed of four- and 32-Kbyte CALRAM modules
• Fast access: one clock
• Keep-alive power
• Soft defect detection (SDD)
• 4-Kbyte calibration (overlay) RAM per module (eight Kbytes total)
• Eight 512-byte overlay regions per module (16 regions total)
— NEXUS/GEPDIS/IEEE – ISTO 5001-1999 debug port (Class 3)
• Nine- or 16-pin interface
— General purpose I/O support
• Address (24) and data (32) pins can be used as GPIO in single chip mode
• 16 GPIO in MIOS14
• Many peripheral pins can be used as GPIO when not used as primary functions
• 5-V outputs with slew rate control

• Integrated I/O System

— True 5-V I/O
— Three time processing units (TPU3)
• 16 channels each
• Each TPU3 is a microcoded timer subsystem
• One 6-Kbyte and one 4-Kbyte dual port TPU RAM (DPTRAM), one (6-Kbyte) shared by two TPU3 modules for TPU microcode and the 4-Kbyte dedicated to the third TPU3 for microcode.
— 22-channel MIOS timer (MIOS14)
• Six modulus counter sub-module (MCSM)
— Four additional MCSM submodules compared to MIOS1
• 10 double action sub-module (DASM).
• 12 dedicated PWM sub-modules (PWMSM)
— Four additional PWM submodules compared to MIOS1 (shared with MIOS GPIO pins)
• Real-time clock sub-module (MRTCSM) provides low power clock/counter
— Requires external 32-KHz crystal
— Uses four pins: two for 32-KHz crystal, two for power/ground.

— Two queued analog-to-digital converter modules (QADC64E_A, QADC64E_B) with AMUXes providing a total of 40 analog channels.
  - 40 total input channels on the two modules with internal multiplexing (AMUXes)
  - Each QADC64E can see all 40 input channels
  - 10 bit A/D converter with internal sample/hold
  - Typical conversion time is 4 µs (250-Kbyte samples/sec)
  - Two conversion command queues of variable length
  - Automated queue modes initiated by:
    - — External edge trigger/level gate
    - — Software command
    - — Periodic/interval timer, assignable to both queue 1 and 2
  - 64 result registers in each QADC64E module
  - Synchronized clock mode allows both QADC64Es to see the same conversion clock. This allows the two modules to look like one large QADC with four queues.
  - Conversions alternate reference (ALTREF) pin. This pin can be connected to a different reference voltage
  - Output data is right or left justified, signed or unsigned
— Message data link controller (DLCMD2) module
  - Two pins muxed with QSMCMB pins. Muxing controlled by QSMCMB PCS3 pin assignment register
  - SAE J1850 Class B data communications network interface compatible and ISO compatible for low-speed (≥125 Kbps) serial data communications in automotive applications
  - 10.4 Kbps variable pulse width (VPW) bit format
  - Digital noise filter, collision detection
  - Hardware cyclical redundancy check (CRC) generation and checking
  - Block mode receive and transmit supported
  - 4X receive mode supported (41.6 Kbps)
  - Digital loopback mode
  - In-frame response (IFR) types 0, 1, 2, and 3 supported
  - Dedicated register for symbol timing adjustments
  - Inter-module bus 3 (IMB3) slave interface
  - Power-saving IMB3 stop mode with automatic wakeup on network activity
  - Power-saving IMB3 CLOCKDIS mode
  - Debug mode available through IMB3 FREEZE signal or user controllable SOFT_FRZ bit
  - Polling and IMB3 interrupt generation with vector lookup available
— Three TouCAN™ modules (TOUCAN_A, TOUCAN_B, TOUCAN_C)
  - 16 message buffers each, programmable I/O modes
  - Maskable interrupts
  - Programmable loop-back for self test operation
  - Independent of the transmission medium (external transceiver is assumed)
  - Open network architecture, multimaster concept
  - High immunity to EMI
  - Short latency time for high-priority messages

- Low power sleep mode, with programmable wake up on bus activity
- TOUCAN_C pins shared with MIOS14 GPIO pins

— Two queued serial modules with one queued-SPI and two SCI each (QSMCM_A, QSMCM_B)
- QSMCM_A matches full MPC555/MPC556 QSMCM functionality
- QSMCM_B has pins muxed with DLCMD2 module
  — Two pins are muxed with DLCMD2 (J1850) transmit and receive pins (B_PCS3_J1850_TX and B_RXD2_J1850_RX)
  — QSMCM_B vs J1850 mux control provided by QPAPCS3 bit in QSMCM pin assignment register (PQSPAR)
- Queued-SPI
  — Provides full-duplex communication port for peripheral expansion or inter-processor communication
  — Up to 32 preprogrammed transfers, reducing overhead
  — Synchronous serial interface with baud rate of up to system clock / 4
  — Four programmable peripheral-selects pins support up to 16 devices
  — Special wrap-around mode allows continuous sampling of a serial peripheral for efficient interfacing to serial analog-to-digital (A/D) converters
- SCI
  — UART mode provides NRZ format and half- or full-duplex interface
  — 16 register receive buffer and 16 register transmit buffer on one SCI
  — Advanced error detection, and optional parity generation and detection
  — word length programmable as eight or nine bits
  — Separate transmitter and receiver enable bits, and double buffering of data
  — Wake-up functions allow the CPU to run uninterrupted until either a true idle line is detected, or a new address byte is received
  — External source clock for baud generation

- Available in package or bumped die

  — Plastic ball grid array (PBGA) packaging
    — 352/388 ball PBGA
    — 27 mm x 27 mm body size
    — 1.0 mm ball pitch

1.2.1.1 MPC565 / MPC566 Optional Features

The following features of the MPC565 / MPC566 are optional features and may not appear in certain configurations:

- 56-MHz operation (40-MHz is default)
- MPC566 supports code compression

1.2.1.2 Supporting Documentation List

This list contains references to currently available and planned documentation.

- RCPU

  — ***RCPU Reference Manual*** (RCPURM/AD)
- NEXUS Standard Specification Rev 1.0 (IEEE-ISTO 5001-1999)

— NEXUS Web Site: ***http://www.ieee-isto.org/Nexus5001/***

- JTAG

  — IEEE 1149.1 Specification

1.2.2 Module or Mode Descriptions

1.2.2.1 Comparison of MPC565 / MPC566 and MPC555 / MPC556

The MPC565 / MPC566 is a derivative of the MPC555 / MPC556. Most functional features of the MPC555 / MPC556 are unchanged on the MPC565 / MPC566. **Table 1-1** shows the high level differences.

Table 1-1 Differences Between MPC555 / MPC556 and MPC565 / MPC566

MODULE	MPC555 / MPC556	MPC565 / MPC566
CPU Core	Identical	
BBC	BBC	BBC w/Improved Code Compression[1]
L2U	Identical	
SRAM	26 Kbytes	36-Kbyte CALRAM with Overlay features
Flash	448-Kbyte CMF	1-Mbyte UC3F (New Programming, etc.)
USIU	USIU	USIU w/Enhanced Interrupt Controller
JTAG	Identical	
READI	None	New Module
UIMB	Identical	
QADC64	2 QADC64 (16 channels on each QADC for 32 total channels)	2 QADC64E w/AMUXes (40 channels accessible from either QADC64E)
QSMCM	(1) Identical (2)	
DLCMD2 (J1850)	None	1
MIOS	MIOS1	MIOS14: MIOS1 with Real-Time Clock (MRTCSM), 4 more PWMSMs and 4 more MCSMs
TouCAN	(2) Identical (3)	
TPU3	(2) Identical (3)	
DPTRAM	(6 Kbytes) Identical (6 Kbytes, 4 Kbytes)	
Power Supplies		
—	40 MHz with two power supplies: nominal 3.3-V to 5.0-V power supplies	56 MHz with two power supplies: 5.0-V I/O, 2.6-V internal logic

NOTES:
1. Available on some options.

1.2.2.2 Additional MPC565 / MPC566 Differences

The following are detailed differences between the MPC565 / MPC566 and the MPC555 / MPC556.

- SPI (MISO, MOSI and SCK) pin drive.
  - MPC565 / MPC566 provides 21-ns rise/fall with 200-pf load using CMOS (20%/70%) levels
- GPIO on MODCK1 pin outputs only 2.6 V
  - MODCK1 pin is in keep-alive power section with no 5-V rail available
  - 5.0-V compatibility modes
    - Input is 5-V friendly
    - 2.6-V output has less slew rate control
    - 2.6-V bus VOH = 2.3 V
- Power supplies for external bus pins
  - QVDDL is quiet supply to hold non-switching outputs quiet even when noisy supply (NVDDL) sags
  - QVDDL supplies pre-drive and other pad logic
  - NVDDL only supplies final PMOS driver stage
  - QVDDL and NVDDL shorted on customer board after filtering
- Pull-up and pull-down changes during $\overline{\text{PORESET}}$ and $\overline{\text{HRESET}}$
  - All 2.6-V/5-V pads (ext. bus: addr/data/control) pull down at reset
  - All 5-V pads pull up at reset
  - Additional control granularity in the PDMCR register
- No pull-ups on QSMCM SCI receive pads
- A_RXD1_QGPI1, A_RXD2_QGPI2, B_RXD1_QGPI1 pins do not have weak pull-up during reset or any other time
- CLKOUT has 3 drive strength options
  - Better matches drive to requirements to reduce EMI
  - 25, 50, 100 pf instead of 45 and 90 pf
- Change reset value of ENGCLK to maximum divide (crystal/128)
  - For a 4-MHz crystal, this is 31.25 kHz
    - ENGCLK is selectable between 2.6 V and 5 V
- A daisy chain between UC3F modules allows either module to provide the reset configuration word (RCW)
- Censorship operation:
  - An RCW bit controls whether or not the entire C3F can be erased while censorship is violated
- BBC SPRs (PPC regs) access in two clocks instead of one clock
- CALRAM internal protection block size is eight Kbytes
  - Instead of four Kbytes on MPC555/MPC556 LRAM
- CALRAM causes machine check exception instead of data storage interrupt (DSI) exception in certain cases
  - For non-overlay CPU core accesses, a DSI exception is taken
  - For overlay accesses and any non-core access (slave mode), a machine check exception is taken
- CALRAM causes DSI exception only if the data relocation (DR) bit in the core ma-

chine state register, MSR[DR], is set. Refer to **Table 3-12**.

— L2U on MPC555/MPC556 already followed this protocol. but the LRAM did not. Now all L-bus peripherals follow this protocol.

— The MSR[DR] bit is described in the ***MPC555/MPC556 User's Manual* (MPC555/MPC556UM/AD)**. Refer to **3.9.1 Machine State Register (MSR)** for more information.

- Four additional PRDS control bits were added to the USIU to allow more granularity of PRDS control on a part
- BBC includes a 4-Kbyte DECRAM that can be used if compression is not used or is not available.

1.2.2.3 SRAM Keep-Alive Power Behavior

The SRAM has three keep-alive power pins ($V_{DDSRAM1}$, $V_{DDSRAM2}$, and $V_{DDSRAM3}$). These pins provide keep-alive power to the SRAM arrays in the CALRAM modules and the DPTRAM modules.

The $V_{DDSRAM1}$ pin powers the 32-Kbyte CALRAM_A during keep-alive while power is off to the MPC565 / MPC566 (except for the keep-alive power supplies). CALRAM_A keeps all of its 32 Kbytes powered during power down.

The $V_{DDSRAM2}$ pin powers the 4-Kbyte CALRAM_B module. The $V_{DDSRAM3}$ pin powers the DPTRAM modules during keep-alive as well as during normal operation. The CALRAM modules only power their arrays from the V_{DDSRAM} pins during keep-alive. During normal operation, they are powered by the normal internal V_{DD} of the part.

The DPTRAM modules (six Kbytes and four Kbytes) and the four-Kbyte DECRAM in the BBC module power their arrays via the VDDSRAM3 pin during during keep-alive and are supplied by VDD during normal operation.

1.3 MPC565 / MPC566 Address Map

The internal memory map is organized as a single 4-Mbyte block. This is shown in **Figure 1-3**. This block can be moved to one of eight different locations. The internal memory space is divided into the following sections:

- Flash memory (one Mbyte) — U-bus memory
- Static RAM memory (26 Kbytes CALRAM) — L-bus memory
- Control registers and IMB3 modules (64 Kbytes), which is partitioned as

  — USIU and flash control registers
  — UIMB interface and IMB3 modules
  — CALRAM and READI control registers (L-bus control register space)

The internal memory block can reside in one of eight possible 4-Mbyte memory spaces. These eight locations are the first eight 4-Mbyte memory blocks starting with address 0x0000 0000, as shown in **Figure 1-2**. There is a user programmable register in the USIU to configure the internal memory map to one of the eight possible locations. Programmability of internal memory map location allows multiple chip system.

The IMB3 address space block in **Figure 1-4** shows memory allocation for IMB3 modules. It does not show the actual memory space required for individual modules. All these modules are mapped to the low address, numerically, of the memory allocated for that module in the IMB3 address space.

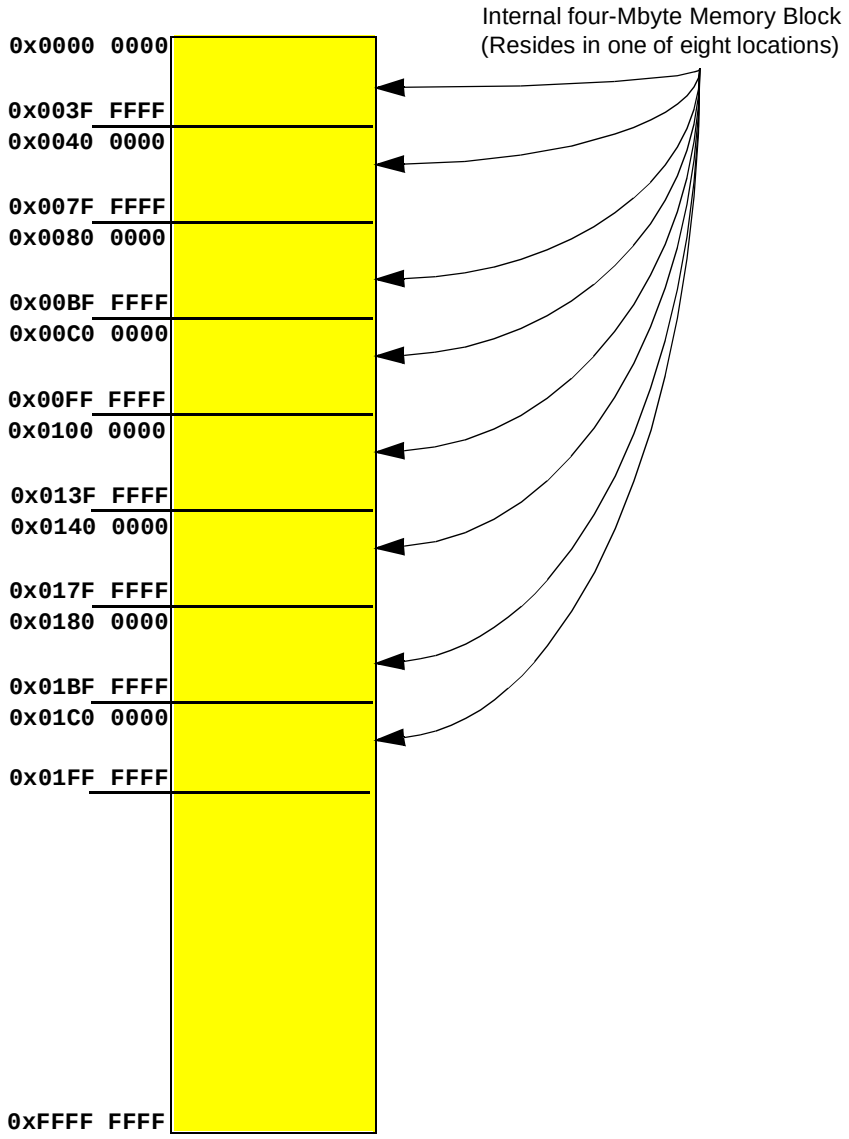


Figure 1-2 MPC565 / MPC566 Memory Map

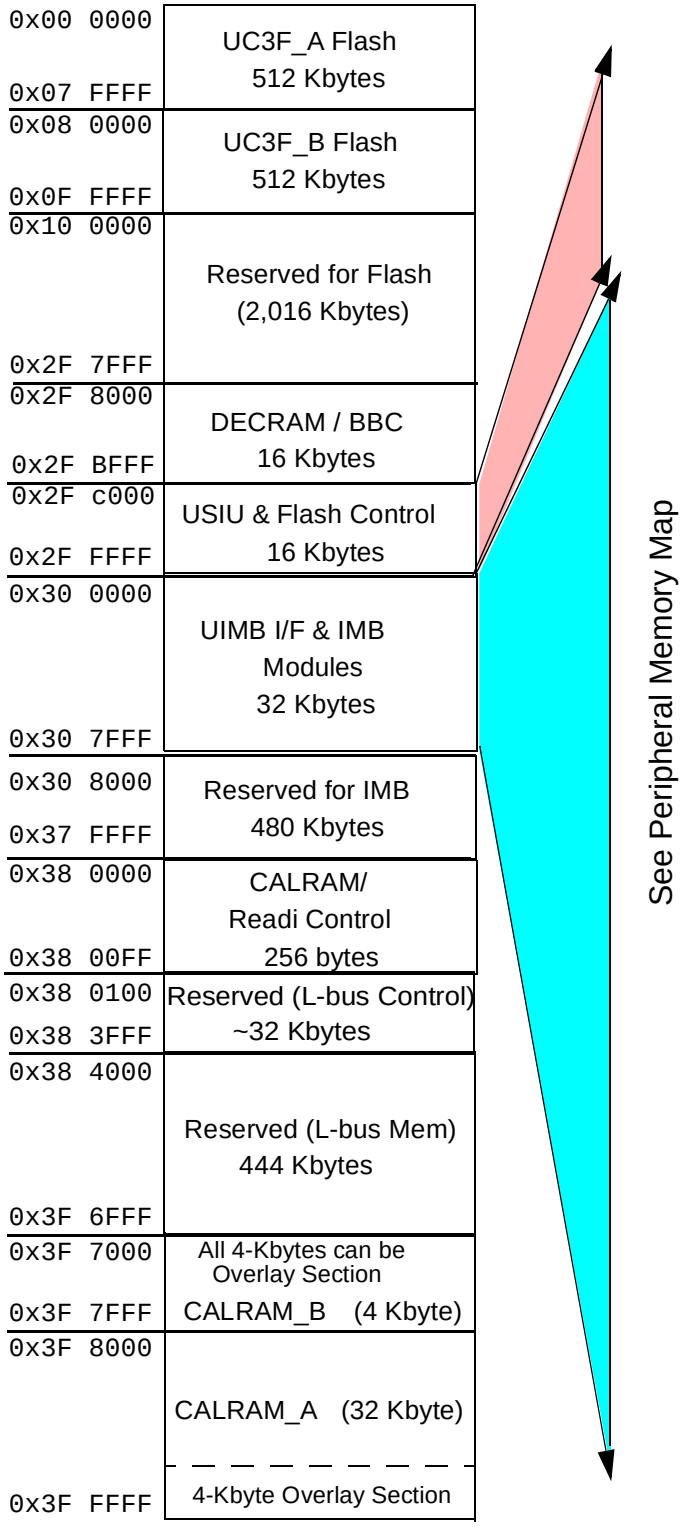


Figure 1-3 MPC565 / MPC566 Internal Memory Block

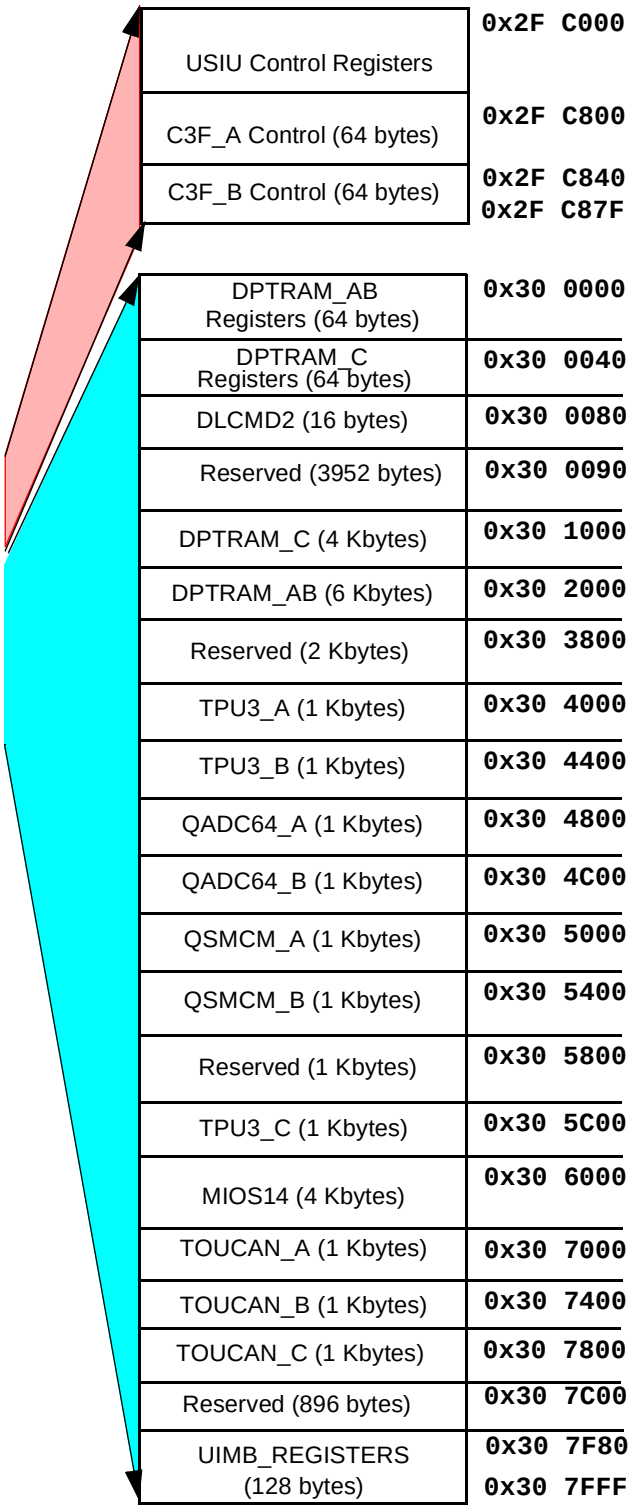


Figure 1-4 MPC565 / MPC566 Peripheral Memory Map

SECTION 2
SIGNAL DESCRIPTIONS

2.1 Pad Function Description

Table 2-1 shows a summary of the functions of the pads of MPC565 / MPC566. The total signals column (total number of pads) shows the pads on the module used. The pads used column (number of pads used) shows how many pads are bonded out on the MPC565 / MPC566.

Table 2-1 MPC565 / MPC566 Pad Function Summary

Functional Group Description	Signals[1]	Total No. Signals	5 V / 2.6 V
24 Address lines (16 Mbytes address space)	A[8:31] — 24 SGPIO	24	2.6-V BUS/ 5-V GPIO
32-bit data bus	D[0:31] — 32 SGPIO	32	
External Interrupts	IRQ[0] — SGPIOC[0]	1	2.6-V BUS/ 2.6-V GPIO[2]
	IRQ[1] — $\overline{\text{RSV}}$ — SGPIOC[1]	6	2.6-V BUS/ 5-V GPIO
	IRQ[2] — $\overline{\text{CR}}$ — SGPIOC[2] — $\overline{\text{MTS}}$		
	IRQ[3] — $\overline{\text{KR}}$ — $\overline{\text{RETRY}}$ — SGPIOC[3]		
	IRQ[4] — AT[2] — SGPIOC[4]		
	IRQ[6:7] — MODCK[2:3]		
	IRQ[5] — MODCK[1] — SPGIOC[5]	1	2.6 V
Bus Control	TSIZ[0:1], RD/$\overline{\text{WR}}$, $\overline{\text{BURST}}$, $\overline{\text{BDIP}}$, $\overline{\text{TS}}$, $\overline{\text{TA}}$, $\overline{\text{TEA}}$, $\overline{\text{OE}}$	11	2.6 V
	$\overline{\text{RSTCONF}}$ — TEXP		
	$\overline{\text{BI}}$ — $\overline{\text{STS}}$		
Development and Debug Support	PTR — FRZ — SGPIOC[6]	7	2.6-V BUS/ 5-V GPIO
	IRQ_OUT — LWP[0] — SGPIOC[7]		
	$\overline{\text{BG}}$ — VF[0] — LWP[1]		
	$\overline{\text{BR}}$ — VF[1] — IWP[2]		
	$\overline{\text{BB}}$ — VF[2] — IWP[3]		
	VFLS[0:1] — IWP[0:1]		
Chip Selects	CS[0:3]	4	2.6 V
Write Enables / Byte Enables	$\overline{\text{WE}}$[0:3] — $\overline{\text{BE}}$[0:3] — AT[0:3]	4	2.6 V
Reset Pins	$\overline{\text{PORESET}}$/$\overline{\text{TRST}}$, $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$	3	2.6 V
JTAG/BDM	TMS, TDI — DSDI, TCK — DSCK, TDO — DSDO, JCOMP	5	2.6 V
READI Port	MCKO, MDO[3:0], $\overline{\text{MSEO}}$, MCKI, MDI[1:0], $\overline{\text{MSEI}}$, $\overline{\text{EVTI}}$, $\overline{\text{RSTI}}$	12	2.6 V
	MDO[4:7] (Shared with MIOS14 GPIO pins)	4 pins, MUXed with MIOS14	2.6 V

Table 2-1 MPC565 / MPC566 Pad Function Summary (Continued)

Functional Group Description	Signals[1]	Total No. Signals	5 V / 2.6 V
Clocks and PLL	XTAL, EXTAL, XFC, CLKOUT, KAPWR, EXTCLK, V_{DDSYN}, V_{SSSYN}, ENGCLK	9	2.6 V
DLCMD2 (J1850) — Shared with QSMCM_B pins	RX, TX	2 0 Pins, MUXed with QSMCMB pins	5 V
TOUCAN_A	CNTX0_A, CNRX0_A	2	5 V
TOUCAN_B	CNTX0_B, CNRX0_B	2	5 V
TOUCAN_C — Shared with MIOS14 GPIO pins	CNTX0_B, CNRX0_B	2 0 Pins, MUXed with MIOS14	5 V
UC3F Flash EEPROM	EPEE, B0EPEE, VFLASH VDDF, VSSF	5	5-V VFLASH, others 2.6 V
CALRAM, DPTRAM	VDDSRAM1, VDDSRAM2, VDDSRAM3	3	2.6 V
QADC64_A QADC64_B	40 analog channels, 2 pwr (VDDA, VSSA), 2 ref, ALTREF, 2 trig	47	5 V
QSMCM_A	PCS[0] — $\overline{SS}$ — QGPIO[0]	12	5 V
	PCS[1:3] — QGPIO[1:3]		
	MISO — QGPIO[4], MOSI — QGPIO[5], SCK — QGPIO[6]		
	TXD[1:2] — QGPO[1:2]		
	RXD[1:2] — QGPI[1:2]		
	ECK		
QSMCM_B, DLCMD2 (J1850)	PCS[3] — J1850_TX	12	5 V
	PCS[0] — $\overline{SS}$ — QGPIO[0]		
	PCS[1:2], MISO — QGPIO[4], MOSI — QGPIO[5], SCK — QGPIO[6]		
	TXD[1:2] — QGPO[1:2]		
	RXD2 — J1850_RX		
	RXD1 — QGPI[1], ECK		
MIOS14	MDA[11:15] MDA[27:31] MPWM[0:3] MPWM[16:19]	18	5 V
	VF[0:2] — MPIO32B[0:2]	5	2.6-V / 5-V GPIO
	VFLS[0:1] — MPIO32B[3:4]		
	C_CNTX0 — MPIO32B[13] C_CNRX0 — MPIO32B[14] MPWM[4:5, 20:21] — MPIO32B[5:6, 11:12]	6	5 V
	READI:MDO[4:7] — MPIO32B[7:10]	4	5 V / 2.6 V
	MPIO32B[15]	1	5 V
	MRTCSM:VDDRTC, VSSRTC, EXTAL32, XTAL32	4	2.6 V

Table 2-1 MPC565 / MPC566 Pad Function Summary (Continued)

Functional Group Description	Signals[1]	Total No. Signals	5 V / 2.6 V
TPU3_A	TPU3_A_CH[0:15]	16	5 V
	TCR2	1	
TPU3_B	TPU3_B_CH[0:15]	16	5 V
	TCR2	1	
TPU3_C	TPU3_C_CH[0:15]	16	5 V
	TCR2	1	

NOTES:
1. "—" implies that the corresponding functions are multiplexed on the pin.
2. This pin was 5-V GPIO on mask set K85H.

2.1.1 MPC565 / MPC566 Pin Sharing

This section describes the additional pin multiplexing on the MPC565 / MPC566. This multiplexing is used to share pins between modules. These functions are described in **Table 2-2**.

Table 2-2 MPC565 / MPC566 Pin Sharing Description

Pin Name	Function
c_cntx0_mpio32b13, c_cnrx0_mpio32b14	TOUCAN_C shared with MIOS14 GPIO
mdo_4_mpio32b10, mdo_5_mpio32b9, mdo_6_mpio32b8, mdo_7_mpio32b7	READI shared with MIOS14 GPIO
mpwm20_mpio32b11, mpwm21_mpio32b12, mpwm4_mpio32b5, mpwm5_mpio32b6	MIOS14 PWM submodule shared with MIOS14 GPIO
vf0_mpio32b0, vf1_mpio32b1, vf2_mpio32b2,	VF pins shared with MIOS14 GPIO
vfls0_mpio32b3, vfls1_mpio32b4	VFLS shared with MIOS14 GPIO
b_pcs3_j1850_tx, b_rxd2_j1850_rx	QSMCM_B pins are muxed with DLCMD2 (J1850). These pins are transmit and receive pins. See **2.4.2 QSMCM_A/QSMCM_B/ DLCMD2 (J1850) PADS.**

2.2 Pad Module Configuration Register (PDMCR)

The slew rate and weak pull-up/pull-down characteristics of some pins are controlled by bits in the PDMCR. This register resides in the SIU memory map. The contents of

the PDMCR are illustrated below. The $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ signal resets all the PDMCR bits asynchronously.

PDMCR — Pads Module Configuration Register **0x2F C03C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SLRC				Reserved		PRDS	SPRDS	T2CLK_PU	PULL_DIS						Re-served
$\overline{\text{HRESET}}$:															
0				0		0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SPULL_DIS		Reserved													
$\overline{\text{HRESET}}$:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-3 PDMCR Bit Descriptions

Bit(s)	Name	Description
0	SLRC0	SLRC0 controls the slew rate of pins on the following modules: TPU3, QADC64, USIU_GPIO0 0 = Slow slew rate for pins. (200 nsec). 1 = Normal slew rate for pins.
1	SLRC1	SLRC1 controls the slew rate of pins on the following modules: QSPI, TOUCAN_A, TOUCAN_B. 0 = Slow slew rate for pins. (50 nsec). 1 = Normal slew rate for pins.
2	SLRC2	SLRC2 controls the slew rate of pins on the QSCI in QSMCM_A and QSMCM_B. 0 = Slow slew rate for pins. (200 ns). 1 = Normal slew rate for pins.
3	SLRC3	SLRC3 controls the slew rate of pins on the following modules: MIOS14, TOUCAN_C 0 = Slow slew rate for pins. (50 ns for TOUCAN_C, 200 nsec for others). 1 = Normal slew rate for pins.
4:5	—	Reserved
6	PRDS	When set, the PRDS bit disables weak pull-up-or-down devices which are enabled at the assertion of $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$. The pads affected by this bit are as follows: All SGPIO, all TPU3 except for T2CLK, all QADC with GPIO, all QSMCM pads. The pads are shown in **Table 2-6**. 0 = Enable weak pull-up/pull-down devices on pads controlled by this signal. 1 = Disable weak pull-up/pull-down devices on pads controlled by this signal.
7	SPRDS	When set, the SPRDS bit disables weak pull up-or-down devices which are enabled at the assertion of $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$. The pads affected by this bit are as follows: $\overline{\text{BDIP}}$, $\overline{\text{TA}}$, $\overline{\text{TS}}$, $\overline{\text{TEA}}$, RD/$\overline{\text{WR}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, $\overline{\text{BB}}$, TSIZ, $\overline{\text{BI}}$/$\overline{\text{STS}}$, $\overline{\text{BURST}}$. The pads are shown in **Table 2-6**. 0 = Enable weak pull-up/pull-down devices on pads controlled by this signal. 1 = Disable weak pull-up/pull-down devices on pads controlled by this signal.
8	T2CLK_PU	This bit controls the pull-up on the TPU T2CLK pins. 0 = pull-ups are enabled if the T2CLK pins are defined as inputs 1 = Pull-ups are disabled on the T2CLK pins

Table 2-3 PDMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
9:14	PULL_DIS	When set, the PULL_DIS bits disables weak pull up-or-down devices which are enabled at the assertion of $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$. The pads affected by these bits are as follows: PULL_DIS[0] (bit 9): all MIOS input pins except C_CNTX0/MPIO32B[13] and C_CNRX0/MPIO32B[14][1] PULL_DIS[1] (bit 10): all QSMCM input pins[2] PULL_DIS[2] (bit 11): all QADC input pins, except ETRIG1 and ETRIG2[3] PULL_DIS[3] (bit 12): all TouCAN input pins[4] PULL_DIS[4] (bit 13): READI module input pins PULL_DIS[5] (bit 14): ETRIG1 and ETRIG2[5] 0 = Enable weak pull-up/pull-down devices on pads controlled by this signal. 1 = Disable weak pull-up/pull-down devices on pads controlled by this signal.
15	—	Reserved
16:17	SPULL_DIS	When set, the SPULL_DIS bits disables weak pull up-or-down devices which are enabled at the assertion of poreset_b_trst_b or hreset_b. The pads affected by these bits are as follows: SPULL_DIS[0] (bit 16): IRQ5_SGPIO5_MODCK1 SPULL_DIS[1] (bit 17): JTAG/BDM pins (TMS, TDI/DSDI, JCOMP, TCK/DSCK) 0 = Enable weak pull-up/pull-down devices on pads controlled by this signal. 1 = Disable weak pull-up/pull-down devices on pads controlled by this signal.
31:18	—	Reserved

NOTES:
1. Was ETRIG1, ETRIG2 in mask set K85H.
2. Was None in mask set K85H.
3. Was TOUCAN_A, TOUCAN_B in mask set K85H.
4. Was TOUCAN_C and MIOS MPIO32B13 and MPIO32B14 in mask set K85H.
5. Was RESERVED in mask set K85H.

2.3 Pad Module Configuration Register (PDMCR2)

The PDMCR2 controls the pre-discharge circuitry for the data bus. This allows for 5-V friendliness on the data bus.

PDMCR2 — Pads Module Configuration Register 2 **0x2F C038**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PREDIS_EN	RESERVED														
$\overline{\text{HRESET}}$:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
$\overline{\text{HRESET}}$:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2-4 PDMCR2 Bit Descriptions

Bit(s)	Name	Description
0	PREDIS_EN	PREDIS_EN 0 = Bus pre-discharge disabled 1 = Bus pre-discharge enabled
1:31	—	Reserved

2.4 Signal Descriptions

The pad ring supports 352 functional pins (388 including all power and ground). Each pin and the functionality it supports are described in this section.

2.4.1 USIU Pads

2.4.1.1 ADDR[8:31]/SGPIOA[8:31]

Pin Name: addr_sgpioa8 – addr_sgpioa31 (24 pins)

Address Bus – Specifies the physical address of the bus transaction. The address is driven onto the bus and kept valid until a transfer acknowledge is received from the slave. ADDR8 is the most significant signal for this bus.

Port SGPIOA[8:31] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.1.2 DATA[0:31]/SGPIOD[0:31]

Pin Name: data_sgpiod0 – data_sgpiod31 (32 pins)

Data Bus – Provides the general-purpose data path between the chip and all other devices. Although the data path is a maximum of 32 bits wide, it can be sized to support 8-, 16-, or 32-bit transfers. DATA[0] is the MSB of the data bus.

Port SGPIOD[0:31] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.1.3 $\overline{\text{IRQ}}$[0]/SGPIOC[0]

Pin Name: irq0_b_sgpioc0

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU. IRQ0 is a nonmaskable interrupt (NMI).

Port SGPIOC[0] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.1.4 $\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1]

Pin Name: irq1_b_rsv_b_sgpioc1

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Reservation – This line used together with the address bus to indicate that the internal core initiated a transfer as a result of a STWCX or a LWARX instruction.

Port SGPIOC[1] – This function allows the pins to be used as general-purpose inputs/ outputs.

2.4.1.5 $\overline{\text{IRQ}}$[2]/$\overline{\text{CR}}$/SGPIOC[2]/$\overline{\text{MTS}}$

Pin Name: irq2_b_cr_b_sgpioc2_mts_b

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Cancel Reservation – Instructs the chip to clear its reservation, some other master has touched its reserved space. An external bus snooper would assert this signal.

Port SGPIOC[2] – This function allows the pins to be used as general-purpose inputs/ outputs.

Memory Transfer Start – This pin is the transfer start signal from the MPC565 / MPC566 memory controller to allow external memory access by an external bus master.

2.4.1.6 $\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$/$\overline{\text{RETRY}}$/SGPIOC[3]

Pin Name: irq3_b_kr_b_retry_b_sgpioc3

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Kill Reservation – In case of a bus cycle initiated by a STWCX instruction issued by the CPU core to a non-local bus on which the storage reservation has been lost, this signal is used by the non-local bus interface to back-off the cycle.

Retry – Indicates to a master that the cycle is terminated but should be repeated. As an input, it is driven by the external slave to retry a cycle.

Port SGPIOC[3] – This function allows the pins to be used as general-purpose inputs/ outputs.

2.4.1.7 $\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4]

Pin Name: irq4_b_at2_sgpioc4

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Address Type – A bit from the address type bus which indicates one of the 16 “address types” to which the address applies. The address type signals are valid at the rising edge of the clock in which the special transfer start ($\overline{STS}$) is asserted.

Port SGPIOC[4] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.1.8 $\overline{IRQ}$[5]/SGPIOC[5]/MODCK[1]

Pin Name: irq5_b_sgpioc5_modck1

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Port SGPIOC[5] – This function allows the pins to be used as general-purpose inputs/outputs.

Mode Clock [1] – Sampled at the negation of $\overline{PORESET}$/$\overline{TRST}$ in order to configure the phase-locked loop (PLL)/clock mode of operation.

2.4.1.9 $\overline{IRQ}$[6:7]/MODCK[2:3]

Pin Name: irq6_b_modck2 – irq7_b_modck3 (two pins)

Interrupt Request – One of the eight external lines that can request, by means of the internal interrupt controller, a service routine from the RCPU.

Mode Clock [2:3] – Sampled at the negation of $\overline{PORESET}$/$\overline{TRST}$ in order to configure the PLL/clock mode of operation.

2.4.1.10 TSIZ[0:1]

Pin Name: tsiz0 – tsiz1 (two pins)

Transfer size – Indicates the size of the requested data transfer in the current bus cycle.

2.4.1.11 RD/$\overline{WR}$

Pin Name: rd_wr_b

Read/Write – Indicates the direction of the data transfer for a transaction. A logic one indicates a read from a slave device; a logic zero indicates a write to a slave device.

2.4.1.12 $\overline{BURST}$

Pin Name: burst_b

Burst Indicator – Indicates whether the current transaction is a burst transaction or not.

2.4.1.13 $\overline{\text{BDIP}}$

Pin Name: bdip_b

Burst Data In Progress – Indicates to the slave that there is a data beat following the current data beat.

2.4.1.14 $\overline{\text{TS}}$

Pin Name: ts_b

Transfer Start – Indicates the start of a bus cycle that transfers data to/from a slave device. This signal is driven by the master only when it has gained ownership of the bus. Every master should negate this signal before the bus is relinquished. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

2.4.1.15 $\overline{\text{TA}}$

Pin Name: ta_b

Transfer Acknowledge – This line indicates that the slave device addressed in the current transaction has accepted the data transferred by the master (write) or has driven the data bus with valid data (read). The slave device negates the $\overline{\text{TA}}$ signal after the end of the transaction and immediately three-state it to avoid contentions on the line if a new transfer is initiated addressing other slave devices. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

2.4.1.16 $\overline{\text{TEA}}$

Pin Name: tea_b

Transfer Error Acknowledge – This signal indicates that a bus error occurred in the current transaction. The MCU asserts this signal when the bus monitor does not detect a bus cycle termination within a reasonable amount of time. The assertion of $\overline{\text{TEA}}$ causes the termination of the current bus cycle, regardless of the state of $\overline{\text{TEA}}$. An external pull-up device is required to negate $\overline{\text{TEA}}$ quickly, before a second error is detected. That is, the pin must be pulled up within one clock cycle of the time it was three-stated by the MPC565 / MPC566.

2.4.1.17 $\overline{\text{RSTCONF}}$/TEXP

Pin Name: rstconf_b_texp

Reset Configuration – Input. This input line is sampled by the chip during the assertion of the $\overline{\text{HRESET}}$ signal in order to sample the reset configuration. If the line is asserted, the configuration mode is sampled from the external data bus. When this line is negated, the configuration mode adopted by the chip is the default one.

Timer Expired – Output. This output line reflects the status of the TEXPS bit in the PLPRCR register in the USIU. This indicates an expired timer value.

2.4.1.18 $\overline{\text{OE}}$

Pin Name: oe_b

Output Enable – This output line is asserted when a read access to an external slave controlled by the GPCM in the memory controller is initiated by the chip.

2.4.1.19 $\overline{\text{BI}}$/$\overline{\text{STS}}$

Pin Name: bi_b_sts_b

Burst Inhibit – This bidirectional, active low, three-state line indicates that the slave device addressed in the current burst transaction is not able to support burst transfers. When the chip drives out the signal for a specific transaction, it asserts or negates $\overline{\text{BI}}$ during the transaction according to the value specified in the appropriate control registers. Negation of the signal occurs after the end of the transaction followed by the immediate three-state. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

Special Transfer Start – This output signal is driven by the chip to indicate the start of a transaction on the external bus or signals the beginning of an internal transaction in showcycle mode.

2.4.1.20 $\overline{\text{CS}}$[0:3]

Pin Name: cs0_b – cs3_b (4 pins)

Chip Select – These output signals enable peripheral or memory devices at programmed addresses if defined appropriately in the memory controller. $\overline{\text{CS}}$[0] can be configured to be the global chip select for the boot device.

2.4.1.21 $\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3]/AT[0:3]

Pin Name: we_b_at0 – we_b_at3 (4 pins)

Write Enable[0:3]/Byte Enable[0:3] – This output line is asserted when a write access to an external slave controlled by the memory controller is initiated by the chip. It can be optionally be asserted on all read and write accesses. See WEBS bit definition in **Table 10-7**. $\overline{\text{WE}}$[0]/$\overline{\text{BE}}$[0] is asserted if the data lane DATA[0:7] contains valid data to be stored by the slave device. $\overline{\text{WE}}$[1]/$\overline{\text{BE}}$[1] is asserted if the data lane DATA[8:15] contains valid data to be stored by the slave device. $\overline{\text{WE}}$[2]/$\overline{\text{BE}}$[2] is asserted if the data line DATA[16:23] contains valid data to be stored by the slave device. $\overline{\text{WE}}$[3]/$\overline{\text{BE}}$[3] is asserted if the data lane DATA[24:31] contains valid data to be stored by the slave device.

Address Type – Indicates one of the 16 address types to which the address applies. The address type signals are valid at the rising edge of the clock in which the special transfer start ($\overline{\text{STS}}$) is asserted.

2.4.1.22 $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$

Pin Name: poreset_b_trst_b

Power-On Reset – This pin should be activated as a result of a voltage failure on the keep-alive power pins. The pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ signal is asserted only if $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ is asserted for more than 100 ns. See **SECTION 7 RESET** for more details on timing.

Test Reset – This input provides asynchronous reset to the test logic (JTAG).

2.4.1.23 $\overline{\text{HRESET}}$

Pin Name: hreset_b

Hard Reset – The chip can detect an external assertion of $\overline{\text{HRESET}}$ only if it occurs while the chip is not asserting reset. After negation of $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ is detected, a 16-cycle period is taken before testing the presence of an external reset. The internal $\overline{\text{HRESET}}$ signal is asserted only if $\overline{\text{HRESET}}$ is asserted for more than 100 ns. To meet external timing requirements, an external pull-up device is required to negate $\overline{\text{HRESET}}$. See **SECTION 7 RESET** for more details on timing.

2.4.1.24 $\overline{\text{SRESET}}$

Pin Name: sreset_b

Soft Reset – The chip can detect an external assertion of $\overline{\text{SRESET}}$ only if it occurs while the chip is not asserting reset. After negation of $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ is detected, a 16-cycle period is taken before testing the presence of an external soft reset. To meet external timing requirements, an external pull-up device is required to negate $\overline{\text{SRESET}}$. See **SECTION 7 RESET** for more details on timing.

2.4.1.25 SGPIOC[6]/FRZ/$\overline{\text{PTR}}$

Pin Name: sgpioc6_frz_ptr_b

Port SGPIOC[6] – This function allows the pins to be used as general-purpose inputs/outputs.

Freeze – Indicates that the RCPU is in debug stopped mode.

Program Trace – Indicates an instruction fetch is taking place in order to allow program flow tracking.

2.4.1.26 SGPIOC[7]/$\overline{\text{IRQOUT}}$/LWP[0]

Pin Name: sgpioc7_irqout_b_lwp0

Port SGPIOC[7] – This function allows the pin to be used as general-purpose inputs/outputs.

Interrupt Out – Indicates that an interrupt has been requested to all external devices.

Load/Store Watchpoint 0 – This output line reports the detection of a data watchpoint in the program flow executed by the RCPU. See **SECTION 22 DEVELOPMENT SUPPORT** for more details.

2.4.1.27 $\overline{\text{BG}}$/VF[0]/LWP[1]

Pin Name: bg_b_vf0_lwp1

Bus Grant – Indicates external data bus status. Is asserted low when the arbiter of the external bus grants to the specific master the ownership of the bus.

Visible Instruction Queue Flush Status – This output line together with VF[1] and VF[2] is output by the chip when a program instructions flow tracking is required. VF report the number of instructions flushed from the instruction queue in the internal core. See **SECTION 22 DEVELOPMENT SUPPORT** for more details.

Load/Store Watchpoint – This output line reports the detection of a data watchpoint in the program flow executed by the RCPU.

2.4.1.28 $\overline{\text{BR}}$/VF[1]/IWP[2]

Pin Name: br_b_vf1_iwp2

Bus Request – Indicates that the data bus has been requested for external cycle.

Visible Instruction Queue Flush Status – This output line together with VF[1] and VF[2] is output by the chip when a program instructions flow tracking is required. VF report the number of instructions flushed from the instruction queue in the internal core. See **SECTION 22 DEVELOPMENT SUPPORT** for more details.

Instruction Watchpoint 2 – This output line reports the detection of an instruction watchpoint in the program flow executed by the RCPU.

2.4.1.29 $\overline{\text{BB}}$/VF[2]/IWP[3]

Pin Name: bb_b_vf2_iwp3

Bus Busy – Indicates that the master is using the bus. This pin is an active negate signal and may need an external pull-up resistor to ensure proper operation and signal timing specifications.

Visible Instruction Queue Flush Status – This output line together with VF[0] and VF[1] is output by the chip when a program instructions flow tracking is required. VF report the number of instructions flushed from the instruction queue in the internal core.

Instruction Watchpoint 3 – This output line reports the detection of an instruction watchpoint in the program flow executed by the internal core.

2.4.1.30 IWP[0:1]/VFLS[0:1]

Pin Name: iwp0_vfls0 – iwp1_vfls1 (two pins)

Instruction Watchpoint – These output lines report the detection of an instruction watchpoint in the program flow executed by the RCPU.

Visible History Buffer Flush Status – These signals are output by the chip to enable program instruction flow tracking. They report the number of instructions flushed from the history buffer in the RCPU. See **SECTION 22 DEVELOPMENT SUPPORT** for details.

2.4.1.31 TMS

Pin Name: tms

Test Mode Select – This input controls test mode operations for on-board test logic (JTAG).

2.4.1.32 TDI/DSDI

Pin Name: tdi_dsdi

Test Data In – This input is used for serial test instructions and test data for on-board test logic (JTAG).

Development Serial Data Input – This input line is the data in for the debug port interface. See **SECTION 22 DEVELOPMENT SUPPORT** for details.

2.4.1.33 TCK/DSCK

Pin Name: tck_dsck

Test Clock – This input provides a clock for on-board test logic (JTAG).

Development Serial Clock – This input line is the clock for the debug port interface. See **SECTION 22 DEVELOPMENT SUPPORT** for details.

2.4.1.34 TDO/DSDO

Pin Name: tdo_dsdo

Test Data Out – This output is used for serial test instructions and test data for on-board test logic (JTAG).

Development Serial Data Output – This output line is the data-out line of the debug port interface. See **SECTION 22 DEVELOPMENT SUPPORT** for details.

2.4.1.35 JCOMP

Pin Name: jcomp_b

JTAG Compliancy – This signal enables the IEEE1149.1 JTAG circuitry in the MPC565 / MPC566. JTAG circuitry in the MPC565 / MPC566. This pin was $\overline{\text{TRST}}$ on the K85H mask set of the MPC566.

2.4.1.36 XTAL

Pin Name: xtal

XTAL – This output line is one of the connections to an external crystal for the internal oscillator circuitry.

2.4.1.37 EXTAL

Pin Name: extal

EXTAL – This line is one of the connections to an external crystal for the internal oscillator circuitry. If this pin is unused, it must be grounded.

2.4.1.38 XFC

Pin Name: xfc

External Filter Capacitance – This input line is the connection pin for an external capacitor filter for the PLL circuitry.

2.4.1.39 CLKOUT

Pin Name: clkout

Clock Out – This output line is the clock system frequency. The CLKOUT drive strength can be configured to full strength, half strength, quarter strength, or disabled. The drive strength is configured using the COM[0:1] bits and CQDS bits in the SCCR register in the USIU.

2.4.1.40 EXTCLK

Pin Name: extclk

EXTCLK – Input. This is the external frequency source for the chip. If this is unused, the pin must be grounded.

2.4.1.41 VDDSYN

Pin Name: vddsyn

VDDSYN – This is the power supply of the PLL circuitry.

2.4.1.42 VSSSYN

Pin Name: vsssyn

VSSSYN – This is the ground reference of the PLL circuitry.

2.4.1.43 ENGCLK/BUCLK

Pin Name: engclk_buclk

ENGCLK – This is the engineering clock output. Drive voltage can be configured to 2.6 V, 5 V (with slew-rate control) or disabled. The drive voltage is configured using the EECLK[0:1] bits in the SCCR register in the SIU.

BUCLK – When the chip is in limp mode, it is operating from a less precise on-chip ring oscillator to allow the system to continue minimum functionality until the system clock is fixed. This backup clock can be seen externally if selected by the values of the EECLK[0:1] bits in the SCCR register in the USIU.

2.4.1.44 PULLSEL

Pin Name: pullsel

Pull Select – This pin determines whether the pull devices on the MIOS and TPU pins are pull-ups or pull-downs. When pull-ups are selected, the pull-ups are to5.0 V. The following MIOS pins pins always have pull down resistors unless disabled in the PDMCR register: VF[0:2]/MPIO32B[0:2], VFLS[0:1]/MPIO32B[3:4], and MDO[7:4]/ MPIO32B[7:10].

2.4.2 QSMCM_A/QSMCM_B/DLCMD2 (J1850) PADS

The MPC565 / MPC566 has two QSMCM modules, QSMCM_A and QSMCM_B. QSMCM_A has identical function to the MPC555/MPC556. QSMCMB has its RXD2 and PCS[3] pins muxed with the DLCMD2 (J1850) module. The muxing of the pins is controlled by the QPAPCS3 bit in QSMCM_B pin assignment reg. ($\overline{\text{PQSPAR}}$), according to **Table 2-5**. The muxed pins default to DLCMD2 function at reset. Since the normal function of the PCS pins within the QSMCM require that this bit be written before the PCS pin is used, the muxing appears transparent to both the QSMCM_B and the DLCMD2. However, only one of the modules, DLCMD2 or QSMCM_B can use the pins in a system. Because of this muxed function on QSMCM_B, the general-purpose input and output functions are not available on the B_PCS[3] and B_RXD2 pins

Table 2-5 DLCMD2 / QSMCM_B SCI2 Pin MUX Control

QPAPCS3 Bit Value	QSMCM_B / DLCMD2 Pin Function
0	B_PCS[3]/J1850_TX pin assigned to J1850_TX. B_RXD2/J1850_RX pin assigned to J1850_RX. Pins are assigned to DLCMD2 (J1850_TX & J1850_RX)
1	B_PCS[3]/J1850_TX pin assigned to PCS[3]. B_RXD2/J1850_RX pin assigned to B_RXD2. Pins are assigned to QSMCMB SCI2 (B_PCS[3], B_RXD2)

2.4.2.1 PCS[0]/$\overline{\text{SS}}$/QGPIO[0]

Pin Name: a_pcs0_ss_b_qgpio0, b_pcs0_ss_b_qgpio0

PCS[0] – This signal provides QSPI peripheral chip select 0.

$\overline{\text{SS}}$ – Assertion of this bidirectional signal places the QSPI in slave mode.

Port QGPIO[0] – When this pin is not needed for a QSPI application it can be configured as a general-purpose input/output.

2.4.2.2 PCS[1:2]/QGPIO[1:2]

Pin Name: a_pcs1_qgpio1, a_pcs1_qgpio1, b_pcs1_qgpio1, b_pcs2_qgpio2

PCS[1:2] – These signals provide two QSPI peripheral chip selects.

Port QGPIO[1:2] – When these pins are not needed for QSPI applications they can be configured as general-purpose input/outputs.

2.4.2.3 PCS[3]/QGPIO[3]/J1850_TX

Pin Name: a_pcs3_qgpio3, b_pcs3_j1850_tx

PCS3 – These signals provide two QSPI peripheral chip selects.

Port QGPIO[3] – When the QSMCM_A PCS3 pin is not needed for QSPI applications it can be configured as a general-purpose input/output.

J1850_TX – The QSMCM_B PCS3 pin can be configured as the J1850 transmit pin for the DLCMC2 module.

2.4.2.4 MISO/QGPIO[4]

Pin Name: a_miso_qgpio4, b_miso_qgpio4

Master-In Slave-Out (MISO) – This bidirectional signal furnishes serial data input to the QSPI in master mode, and serial data output from the QSPI in slave mode.

Port QGPIO[4] – When this pin is not needed for a QSPI application it can be configured as a general-purpose input/output.

2.4.2.5 MOSI/QGPIO[5]

Pin Name: a_mosi_qgpio5, b_mosi_qgpio5

Master-Out Slave-In (MOSI) – This bidirectional signal furnishes serial data output from the QSPI in master mode and serial data input to the QSPI in slave mode.

Port QGPIO[5] – When this pin is not needed for a QSPI application it can be configured as a general-purpose input/output.

2.4.2.6 SCK/QGPIO[6]

Pin Name: a_sck_qgpio6, b_sck_qgpio6

SCK – This bidirectional signal furnishes the clock from the QSPI in master mode or furnishes the clock to the QSPI in slave mode.

Port QGPIO[6] – When this pin is not needed for a QSPI application, it can be configured as a general-purpose input/output. When the QSPI is enabled for serial transmitting, the pin can *not* function as a GPIO.

2.4.2.7 TXD[1:2]/QGPO[1:2]

Pin Name: a_txd1_qgpo1, b_txd1_qgpo1, a_txd2_qgpo2, b_txd2_qgpo2

Transmit Data – These output signals are the serial data outputs from the SCI1 and SCI2.

Port QGPO[1:2] – When these pins are not needed for a SCI applications, they can be configured as general-purpose outputs. When the transmit enable bit in the SCI control register is set to a logic 1, these pins can *not* function as general-purpose outputs

2.4.2.8 RXD[1:2]/QGPI[1:2], RXD[1]/QGPI1, RXD[2]/J1850_RX

Pin Name: a_rxd1_qgpi1, a_rxd2_qgpi2, b_rxd1_qgpi1, b_rxd2_j1850_rx

Receive Data (RXD1, RXD2) – These input signals furnish serial data inputs to the SCI1 and SCI2.

Port QGPI[1:2] – When this pin is not needed for SCI1 applications, it can be configured as a general-purpose input. When the receive enable bit in the SCI control register is set to a logic 1, this pin can *not* function as a general-purpose input.

J1850_RX – The QSMCM_B RXD2 pin can be configured as the J1850 transmit pin for the DLCMC2 module.

2.4.2.9 ECK

Pin Name: b_eck

External Baud Clock (ECK) – This signal provides an external baud clock used by SCI1 and SCI2. (Note: this pin is not usable on the MPC565 / MPC566.)

2.4.3 MIOS14 Pads

Some of the MIOS14 signals are shared with other functions. Only one function can be used at a time. These extra functions include debug (VF[0:2], VFLS[0:1]), READI (MDO[4:7]), TouCAN (C_CNTX0, C_CNRX0), and the additional MIOS14 functions MPWM[4:5, 20:21], and 32KCLKOUT. For additional information about MIOS14 testing and pin usage, see **17.10.2.2 MIOS14 Test and Pin Control Register (MIOS14TPCR)**.

2.4.3.1 MDA[11, 13, 27, 30]

Pin Name: mda11, mda13, mda27, mda30 (four pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

Clock Input – Each of these pins provide a clock input to the modulus counter submodule. MDA11 can be used as the clock input to the MMCSM6 modulus counter. MDA13 can be used as the clock input to the MMCSM22 modulus counter.

2.4.3.2 MDA[12, 14, 28, 31]

Pin Name: mda12, mda14, mda28, mda31(four pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

Load Input – Each of these pins provide a load input to the modulus counter submodule. MDA12 can be used as the load input to the MMCSM6 modulus counter. MDA14 can be used as the load input to the MMCSM22 modulus counter.

2.4.3.3 MDA[15], [27:31]

Pin Name: mda15, mda27 – mda31 (six pins)

Double Action – Each of these pins provide a path for two 16-bit input captures and two 16-bit output compares.

2.4.3.4 MPWM[0:3]

Pin Name: mpwm0 – mpwm3 (four pins)

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

2.4.3.5 MPWM[16, 18]

Pin Name: mpwm16, mpwm18 (two pins)

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

Clock Input – Each of these pins provide a clock input to the modulus counter submodule. MPWM16 can be used as the clock input to the MMCSM6 modulus counter. MPWM18 can be used as the clock input to the MMCSM22 modulus counter.

2.4.3.6 MPWM[17, 19]

Pin Name: mpwm17, mpwm19 (two pins)

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

Load Input – Each of these pins provide a load input to the modulus counter submodule. MPWM17 can be used as the load input to the MMCSM6 modulus counter. MPWM19 can be used as the load input to the MMCSM22 modulus counter.

2.4.3.7 VF[0:2]/MPIO32B[0:2]

Pin Name: vf0_mpio32b0 – vf2_mpio32b2 (three pins)

Visible Instruction Queue Flush Status – These lines output by the chip when program instruction flow tracking is required. VF reports the number of instructions flushed from the instruction queue in the internal core.

Port MIOS GPIO[0:2] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.3.8 VFLS[0:1]/MPIO32B[3:4]

Pin Name: vfls0_mpio32b3 – vfls1_mpio32b4 (two pins)

Visible History Buffer Flush Status – These signals are output by the chip to allow program instruction flow tracking. They report the number of instructions flushed from the history buffer in the RCPU. See **SECTION 22 DEVELOPMENT SUPPORT** for details.

MIOS GPIO – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.3.9 MPWM[4:5]/MPIO32B[5:6]

Pin Names: mpwm4_mpio32b5 – mpwm5_mpio32b6

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

Port MIOS GPIO[5:6] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.3.10 MDO[7:4]/MPIO32B[7:10]

Pin Names: mdo_7_mpio32b7, mdo_6_mpio32b8, mdo_5_mpio32b9, mdo_4_mpio32b10

Port MIOS GPIO[7:10] – This function allows the pins to be used as general-purpose inputs/outputs.

READI (NEXUS) Data Out – This function allows the pins to be used by the READI modules as NEXUS data output pins.

2.4.3.11 MPWM[20:21]/MPIO32B[11:12]

Pin Names: mpwm20_mpio32b11– mpwm21_mpio32b12

Pulse Width Modulation – These pins provide variable pulse width output signals at a wide range of frequencies.

Port MIOS GPIO[11:12] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.3.12 C_CNTX0/C_CNRX0/MPIO32B[13:14]

Pin Names: c_cntx0_mpio32b13 – c_cnrx0_mpio32b14

Port MIOS GPIO[13:14] – This function allows the pins to be used as general-purpose inputs/outputs.

TOUCAN_C Receive/Transmit – These pins provide the receive and transmit pins for the TOUCAN_C module.

2.4.3.13 MPIO32B15

Pin Name: mpio32b15

Port MIOS GPIO[15] – This function allows the pins to be used as general-purpose inputs/outputs.

2.4.4 32-KHz Oscillator Pads

2.4.4.1 XTAL32

Pin Name: xtal32

XTAL32 – This output line is one of the connections to an external 32-KHz crystal for the MIOS14 real-time clock submodule (MRTCSM).

2.4.4.2 EXTAL32

Pin Name: extal32

EXTAL32 – This line is one of the connections to an external 32-KHz crystal for the internal oscillator circuitry used by the MRTCSM. If this pin is unused, it must be grounded.

2.4.4.3 VDDRTC

Pin Name: VDDRTC

VDDRTC – This is the power supply of the 32-KHz oscillator circuitry and the MRTCSM.

2.4.4.4 VSSRTC

Pin Name: VSSRTC

VSSRTC – This is the power supply of the 32-KHz oscillator circuitry.

2.4.5 TPU3_A/TPU3_B/TPU3_C Pads

2.4.5.1 TPUCH[0:15]

Pin Name: a_tpuch0 – a_tpuch15 (16 pins for TPU3_A), b_tpuch0 -– b_tpuch15 (16 pins for TPU3_B), c_tpuch0 – c_tpuch15 (16 pins for TPU3_C)

TPU Channels – These signals provide each TPU3 with 16 input/output programmable timed events.

2.4.5.2 T2CLK

Pin Name: a_t2clk (one pin for TPU3_A) – b_t2clk (one pin for TPU3_B) – c_t2clk (one pin for TPU3_C)

T2CLK – This signal is used to clock or gate the timer count register 2 (TCR2) within the TPU. This pin is an output-only in special test mode.

2.4.6 QADC64_A/QADC64_B Pads

2.4.6.1 ETRIG[1:2]

Pin Name: etrig1 – etrig2

ETRIG – These are the external trigger inputs to the QADC64_A and QADC64_B modules. ETRIG[1] can be configured to be used by both QADC64_A and QADC64_B. Likewise, ETRIG[2] can be used for both QADC64_B and QADC64_A. The trigger input pins are associated with the scan queues.

2.4.6.2 AN[44]/ANw/PQB[0]

Pin Name: an44_anw_a_pqb0

Analog Channel (AN[44]) – Internally multiplexed input-only analog channels. Passed on as a separate signal to the QADC64E.

Multiplexed Analog Input (ANw) – Externally multiplexed analog input.

Port (A_PQB[0]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.3 AN[45]/ANx/PQB[1]

Pin Name: an45_anx_a_pqb1

Analog Channel (AN[45]) – Internally multiplexed input-only analog channels. Passed on as a separate signal to the QADC64E.

Multiplexed Analog Input (ANx) – Externally multiplexed analog input.

Port (A_PQB[1]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.4 AN[46]/ANy/PQB[2]

Pin Name: an46_any_a_pqb2

Analog Channel (AN[46]) – Internally multiplexed input-only analog channel. The input is passed on as a separate signal to the QADC64E.

Multiplexed Analog Input (ANy) – Externally multiplexed analog input.

Port (A_PQB[2]) –When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.5 AN[47]/ANz/PQB[3]

Pin Name: an47_anz_a_pqb3

Analog Input (AN[47]) – Internally multiplexed input-only analog channel. The input is passed on as a separate signal to the QADC64E.

Multiplexed Analog Input (ANz) – Externally multiplexed analog input.

Port (A_PQB[3]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.6 AN[48:51]/PQB[4:7]

Pin Name: an48_a_pqb4 – an51_a_pqb7

Analog Input (AN[48:51]) – Analog input channel. The input is passed on as a separate signal to the QADC64E.

Port (A_PQB[4:7]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.7 AN[52:54]/MA[0:2]/PQA[0:2]

Pin Name: an52_a_ma0_pqa0 – an54_a_ma2_pqa2

Analog Input (AN[52:54]) – Input-only. The input is passed on as a separate signal to the QADC64E.

Multiplexed Address (MA[0:2]) – Output. Provides a three-bit multiplexed address output to the external multiplexer chip to allow selection of one of the eight inputs.

Port (A_PQA[0:2]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.8 AN[55:59]/PQA[3:7]

Pin Name: an55_a_pqa3 – an59_a_pqa7

Analog Input (AN[55:59]) – Input-only. The input is passed on as a separate signal to the QADC64E.

Port (A_PQA[3:7]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.9 AN[64:71]/PQB[0:7]

Pin Name: an64_b_pqb0 – an71_b_pqb7

Analog Input (AN[55:59]) – Input-only. The input is passed on as a separate signal to the QADC64E.

Port (B_PQB[3:7]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.10 AN[72:74]/MA[0:2]/PQA[0:2]

Pin Name: an72_b_ma0_pqa0 – an74_b_ma2_pqa2

Analog Input (AN[72:74]) – Input-only. The input is passed on as a separate signal to the QADC64E.

Multiplexed Address (MA[0:2]) – Output. Provides a three-bit multiplexed address output to the external multiplexer chip to allow selection of one of the eight inputs.

Port (B_PQA[0:2]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.11 AN[75:79]/PQA[3:7]

Pin Name: an75_b_pqa3 – an79_b_pqa7

Analog Input (AN[75:79]) – Input-only. The input is passed on as a separate signal to the QADC64E.

Port (B_PQA[3:7]) – When this pin is not needed for QADC converter functions, it can be used as a general-purpose input or output.

2.4.6.12 AN[80:87]

Pin Name: an80 – an87

Analog Input (AN[80:87]) – Input-only. The input is passed on as a separate signal to the QADC64E by the AMUX.

2.4.6.13 VRH

Pin Name: vrh

VRH – Input pin for high reference voltage for the QADC64_A and QADC64_B modules.

2.4.6.14 VRL

Pin Name: vrl

VRL – Input pin for low reference voltage for the QADC64_A and QADC64_B modules.

2.4.6.15 ALTREF

Pin Name: altref

ALTREF – Input pin for alternate reference voltage for the QADC64_A and QADC64_B modules.

2.4.6.16 VDDA

Pin Name: vdda

VDDA – Power supply input to analog subsystems of the QADC64_A and QADC64_B modules.

2.4.6.17 VSSA

Pin Name: vssa

VSSA – Input. Ground level for analog subsystems of the QADC64_A and QADC64_B modules.

2.4.7 TOUCAN_A / TOUCAN_B / TOUCAN_C Pads

2.4.7.1 CNTX0

Pin Name: a_cntx0 (one pin for first CAN), b_cntx0 (one pin for second CAN), c_cntx0_mpio32b13 (one pin for CAN_C, muxed with a MIOS14 GPIO)

TouCAN Transmit Data 0 – This signal is the serial data output.

2.4.7.2 CNRX0

Pin Name: a_cnrx0 (one pin for first CAN), b_cnrx0 (one pin for second CAN), c_cnrx0_mpio32b14 (one pin for CAN_C, MUXed with a MIOS14 GPIO)

TouCAN Receive Data – This signal furnishes serial input data.

2.4.8 READI Pads

Some READI pads are shared with MIOS14 functions, see **2.4.3 MIOS14 Pads**.

2.4.8.1 $\overline{\text{MSEO}}$

Pin Name: mseo_b

$\overline{\text{MSEO}}$ – Message start/end out ($\overline{\text{MSEO}}$) is an output pin which indicates when a message on the MDO pins has started, when a variable length packet has ended, and when the message has ended. External latching of $\overline{\text{MSEO}}$ occurs on rising edge of MCKO.

2.4.8.2 MDO[0:3]

Pin Name: mdo_3

MDO[0:3] – Message Data Output. Message data out (MDO[7:0] or MDO[1:0]) are output pins used for uploading OTM, BTM, DTM, and read/write accesses. External

latching of MDO occurs on rising edge of MCKO. Eight pins are implemented; four (MDO[4:7]) are shared with MIOS14 GPIO.

2.4.8.3 MCKO

Pin Name: mcko

MCKO – Output. Message clock-out (MCKO) is a free-running output clock to development tools for timing of MDO and MSEO pin functions. MCKO is the same as the MCU system clock.

2.4.8.4 $\overline{\text{RSTI}}$

Pin Name: rsti_b

$\overline{\text{RSTI}}$ – Reset In. $\overline{\text{RSTI}}$ is the NEXUS port reset input.

2.4.8.5 $\overline{\text{EVTI}}$

Pin Name: evti_b

$\overline{\text{EVTI}}$ – Event in (EVTI) is level sensitive when configured for breakpoint generation, otherwise it is edge sensitive.

2.4.8.6 $\overline{\text{MSEI}}$

Pin Name: msei_b

$\overline{\text{MSEI}}$ – Message Start/End Input. The $\overline{\text{MSEI}}$ input is a NEXUS input pin which indicates when a message on the MDI pins has started, when a variable length packet has ended, and when the message has ended. Internal latching of $\overline{\text{MSEI}}$ occurs on rising edge of MCKI.

2.4.8.7 MDI[0:1]

Pin Name: mdi_1

MDI[0:1] – Message Data Input. MDI[1:0] or MDI[0] are NEXUS input pins used for downloading configuration information, writes to user resources, etc. Internal latching of MDI will occur on rising edge of MCKI.

2.4.8.8 MCKI

Pin Name: mcki

MCKI – Message Clock Input. MCKI is the NEXUS message clock input.

2.4.9 UC3F Pads

2.4.9.1 EPEE

Pin Name: epee

EPEE – Input. This control signal externally controls the program or erase operations. When low, program or erase operations in both of the UC3F modules (UC3F_512KA and UC3F_512KB) are disabled.

2.4.9.2 B0EPEE

Pin Name: b0epee

B0EPEE – Input. This control signal externally controls the program or erase operations of block 0 of UC3F_512KA. When low, program or erase operations in block 0 of the UC3F_512KA module are disabled.

2.4.9.3 VFLASH

Pin Name: vflash

VFLASH – Input. Flash supply voltage (5-V supply) used during all operations of the UC3F.

2.4.9.4 VDDF

Pin Name: vddf

VDDF – Flash core voltage input (2.6-V supply).

2.4.9.5 VSSF

Pin Name: vssf

VSSF – Flash core zero supply input.

2.4.10 Global Power Supplies

2.4.10.1 NVDDL

Pin Name: nvddl

NVDDL – Noisy 2.6-V voltage supply input. This supplies the final output stage of the 2.6-V pad output drivers.

The NVDDL and QVDDL supplies should be connected to the same power supply in a user's system.

2.4.10.2 QVDDL

Pin Name: qvddl

QVDDL – Quiet 2.6-V voltage supply input. This supplies all pad logic and pre-driver circuitry, except for the the final output stage of the 2.6-V pad output drivers.

The NVDDL and QVDDL supplies should be connected to the same power supply in a user's system.

2.4.10.3 VDDH

Pin Name: vddh

VDDH – 5-V voltage supply input.

2.4.10.4 VDD

Pin Name: vdd

VDD – 2.6-V voltage supply input for internal logic.

2.4.10.5 KAPWR

Pin Name: kapwr

Keep-Alive Power – 2.6-V voltage supply input for the oscillator and keep-alive registers.

2.4.10.6 VDDSRAM1

Pin Name: vddsram1

SRAM Keep-Alive Power – 2.6-V voltage supply input for the keep-alive section of the CALRAM_A (32K) module. This pin supplies only keep-alive power to the CALRAM_A (32K) module. Run current is supplied by normal VDD.

2.4.10.7 VDDSRAM2

Pin Name: vddsram2

SRAM Keep-Alive Power – 2.6-V voltage supply input for the CALRAM_B (4 Kbyte) module. This pin supplies only keep-alive power to the CALRAM_B (4 Kbyte) module. Run current is supplied by normal VDD.

2.4.10.8 VDDSRAM3

Pin Name: vddsram3

SRAM Keep-Alive Power – 2.6-V voltage supply input for the arrays in the DPTRAM_AB (6 Kbytes), DPTRAM_C (4 Kbytes), and the BBC DECRAM (4 Kbytes) modules. This pin supplies only keep-alive power to both DPTRAM arrays and the DECRAM module. Run current is supplied by normal VDD.

2.4.10.9 VSS

Pin Name: vss

VSS – Ground level reference input.

2.5 Reset State

All input pins, with the exception of the power supply and clock related pins, are “weakly pulled” to a value during reset by a 130-µA (maximum) resistor based on cer-

tain conditions. See **APPENDIX E ELECTRICAL CHARACTERISTICS** for minimum specifications. In reset state all I/O pins become inputs, and all outputs except CLKOUT, $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$, are pulled only by the pull-up/pull-down.

2.5.1 Pin Functionality Configuration Out of Reset

The functionality out of reset of some pins that support multiple functionality is defined in the SIUMCR through the reset configuration word. For details on which multiplexed pins are configured by the reset configuration word and how they are configured, refer to **7.5.2 Hard Reset Configuration Word**.

The 2.6-V related pins have selectable output buffer drive strengths which are controlled by the COM[0] bit in the USIU's system clock and reset control register (SCCR). The control is as follows:

0 = 2.6-V bus pins full drive (50-pF load)*
1 = 2.6-V bus pins reduced drive (25-pF load)

* The bus pin drive selectability definition is inverted from the selectability of the pin control in the PDMCR register (for the TPU, QADC64E, USIU (SGPIO), QSPI, TouCAN, QSCI, and MIOS pins).

2.5.2 Pin State During Reset

While $\overline{\text{HRESET}}$ is asserted, the reset configuration value is latched from the data bus into various bits on the part. The function of many pins depends upon the value latched. If the value on the data bus changes, then the function of various pins may also change. This is especially true if the reset configuration word (RCW) comes from the flash, since the flash won't drive the RCW until 256 clocks after the start of $\overline{\text{HRESET}}$. However, the pins must not cause any spurious conditions or consume an excessive amount of power during reset. To prevent these conditions, the pins need to have a defined reset state. **Table 2-6** describes the reset state of the pins based on pin functionality.

All pins are initialized to a "reset state" during reset. This state remains active until reset is negated or until software disables the pull-up or pull-down device based on the pin functionality. Upon assertion of the corresponding bits in the pin control registers and negation of reset, the pin acquires the functionality that was programmed.

2.5.3 Power-On Reset and Hard Reset

Power-on reset and hard reset affect the functionality of the pins out of reset. (During soft reset, the functionality of the pins is unaltered.) Upon assertion of the power-on reset signal ($\overline{\text{PORESET}}$/$\overline{\text{TRST}}$) the functionality of the pin is not yet known. The weak pull-up or weak pull-down resistors are enabled. The reset configuration word configures the system, and towards the end of reset the pin functionality is known. Based upon pin functionality, the pull-up or pull-down devices are either disabled immediately at the negation of reset or remain enabled. This is shown in **Table 2-6**.

NOTE

PD refers to a weak pull-down, PU_2.6V refers to a weak pull-up to 2.6 V, and PU_5V refers to a weak pull-up to 5 V. All control of the weak-pull devices is in the PDMCR, described in **Table 2-3**

Hard reset can occur at any time, and there may be a bus cycle pending. For this reason, the bits in PDMCR that control the enabling and disabling of the pull-up or pull-down resistors in the pads are set or reset synchronously. ($\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ affects these bits asynchronously.) This causes the pull-up or pull-down resistors to be enabled at a time when they do not cause contention on the pins and are disabled before they can cause any contention on the pins.

CAUTION

2.6-V inputs are 5-V tolerant, but 2.6-V outputs are not. Do not connect 2.6-V outputs to a driver or pull-up greater than 3.1 V.

2.5.4 Pin Reset States

Table 2-6 summarizes the reset states of all the pins on the MPC565 / MPC566.

Table 2-6 Pin Reset State

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
USIU					
ADDR[8:31]/ SGPIOA[8:31]	ADDR[8:31][1]	I/O	2.6 V	PD until reset negates[2]	Controlled by SC bit in the reset config word. See **Table 6-11**
	SGPIOA[8:31][3]	I/O	5 V	PD until PRDS is set	
DATA[0:31]/ SGPIOD[0:31]	DATA[0:31][1]	I/O	2.6 V	PD until reset negates	Controlled by SC bit in the reset config word. See **Table 6-11**.
	SGPIOD[0:31][3]	I/O	5 V	PD until PRDS is set	
$\overline{\text{IRQ}}$[0]/SGPIOC[0]	$\overline{\text{IRQ}}$[0]	I	2.6 V	Pin floats during reset, ext. PU required.	$\overline{\text{IRQ}}$[0]
	SGPIOC[0]	I/O	2.6 V[4]	Pin floats during reset	
$\overline{\text{IRQ}}$[1]/ $\overline{\text{RSV}}$/SGPIOC[1]	$\overline{\text{IRQ}}$[1]	I	2.6 V	PD until reset negates[2,5]	$\overline{\text{IRQ}}$[1]
	$\overline{\text{RSV}}$[6]	O	2.6 V	PD until reset negates[2]	
	SGPIOC[1][3]	I/O	5 V	PD until PRDS is set	
$\overline{\text{IRQ}}$[2]/ $\overline{\text{CR}}$/SGPIOC[2]/ $\overline{\text{MTS}}$	$\overline{\text{IRQ}}$[2]	I	2.6 V	PD until reset negates[2,5]	$\overline{\text{IRQ}}$[2]
	$\overline{\text{CR}}$	I	2.6 V	PD until reset negates[2,5]	
	SGPIOC[2][3]	I/O	5 V	PD until PRDS is set	
	$\overline{\text{MTS}}$[6]	O	2.6 V	PD until PRDS negates	
$\overline{\text{IRQ}}$[3]/ $\overline{\text{KR}}$, $\overline{\text{RETRY}}$/ SGPIOC[3]	$\overline{\text{IRQ}}$[3]	I	2.6 V	PD until reset negates[2,5]	$\overline{\text{IRQ}}$[3]
	$\overline{\text{KR}}$, $\overline{\text{RETRY}}$[6]	I/O	2.6 V	PD when driver not enabled[7]	
	SGPIOC[3][3]	I/O	5 V	PD until PRDS is set	

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
$\overline{\text{IRQ}}$[4]/ AT[2]/ SGPIOC[4]	$\overline{\text{IRQ}}$[4]	I	2.6 V	PD until reset negates[2,5]	$\overline{\text{IRQ}}$[4]
	AT[2][6]	O	2.6 V	PD until reset negates[2]	
	SGPIOC[4]	I/O	5 V	PD until PRDS is set	
$\overline{\text{IRQ}}$[5]/ SGPIOC[5]/ MODCK[1][8]	$\overline{\text{IRQ}}$[5]	I	2.6 V	PU_2.6V until reset negates[2,9]	MODCK[1] until reset negates
	SGPIOC[5][6]	I/O	2.6 V	PU_2.6V until SPULL_DIS0 is set	
	MODCK[1]	I	2.6 V	PU_2.6V until reset negates[2]	
$\overline{\text{IRQ}}$[6:7]/ MODCK[2:3][8]	$\overline{\text{IRQ}}$[6:7]	I	2.6 V	PU_2.6V until SPRDS is set[9]	MODCK[2:3] until reset negates
	MODCK[2:3]	I	2.6 V	PU_2.6V until reset negates	
TSIZ[0:1]	TSIZ[0:1][6]	I/O	2.6 V	PD when driver not enabled or until SPRDS is set	TSIZ[0:1]
RD/$\overline{\text{WR}}$	RD/$\overline{\text{WR}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	RD/$\overline{\text{WR}}$
$\overline{\text{BURST}}$	$\overline{\text{BURST}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	$\overline{\text{BURST}}$
$\overline{\text{BDIP}}$	$\overline{\text{BDIP}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	$\overline{\text{BDIP}}$
$\overline{\text{TS}}$[10]	$\overline{\text{TS}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set An external pull-up is required in order to guarantee the pin does not assert between bus cycles.	$\overline{\text{TS}}$
$\overline{\text{TA}}$[10]	$\overline{\text{TA}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	$\overline{\text{TA}}$
$\overline{\text{TEA}}$	$\overline{\text{TEA}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set An external pull-up is required in order to negate the pin in appropriate time	$\overline{\text{TEA}}$
$\overline{\text{RSTCONF}}$/TEXP[8]	$\overline{\text{RSTCONF}}$	I	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	$\overline{\text{RSTCONF}}$ until reset negates. Following reset, the function is defined by the RCTX bit in the SIUMCR. See **Table 6-5**.
	TEXP[6]	O	2.6 V		
$\overline{\text{OE}}$	$\overline{\text{OE}}$[6]	O	2.6 V	PU_2.6V until reset negates	$\overline{\text{OE}}$
$\overline{\text{BI}}$/$\overline{\text{STS}}$	$\overline{\text{BI}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	Controlled by DBGC in the reset config word. See **Table 6-8**.
	$\overline{\text{STS}}$[6]	O	2.6 V		
$\overline{\text{CS}}$[0:3]	$\overline{\text{CS}}$[0:3][6]	O	2.6 V	PU_2.6V until reset negates	$\overline{\text{CS}}$[0:3]
$\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3]/ AT[0:3]	$\overline{\text{WE}}$[0:3]/$\overline{\text{BE}}$[0:3][6]	O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	Controlled by bit ATWC (bit 12) of the Reset Configuration Word. See **Table 6-7**.
	AT[0:3][6]	O	2.6 V		

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
$\overline{\text{PORESET}}$/ $\overline{\text{TRST}}$[8, 11]	$\overline{\text{PORESET}}$/$\overline{\text{TRST}}$	I	2.6 V	—	$\overline{\text{PORESET}}$/$\overline{\text{TRST}}$
$\overline{\text{HRESET}}$[8]	$\overline{\text{HRESET}}$	I/O	2.6 V[12]	PU_2.6V when driver not enabled An external pull-up is required in order to negate the pin in appropriate time	$\overline{\text{HRESET}}$
$\overline{\text{SRESET}}$[8]	$\overline{\text{SRESET}}$	I/O	2.6 V[12]	PU_2.6V when driver not enabled An external pull-up is required in order to negate the pin in appropriate time	$\overline{\text{SRESET}}$
SGPIOC[6]/ FRZ/ $\overline{\text{PTR}}$	SGPIOC[6][3]	I/O	5 V	PD until PRDS is set	$\overline{\text{PTR}}$
	FRZ[6]	O	2.6 V	PD until reset negates[2]	
	$\overline{\text{PTR}}$[6]	O	2.6 V	PD until reset negates[2]	
SGPIOC[7]/ $\overline{\text{IRQOUT}}$/LWP[0]	SGPIOC[7][3]	I/O	5 V	PD until PRDS is set	LWP[0]
	$\overline{\text{IRQOUT}}$[6]	O	2.6 V	PD until reset negates[2]	
	LWP[0][6]	O	2.6 V	PD until reset negates[2]	
$\overline{\text{BG}}$/ VF[0]/ LWP[1]	$\overline{\text{BG}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	Controlled by DBGC in reset config word. See **Table 6-8**.
	VF[0][6]	O	2.6 V		
	LWP[1][6]	O	2.6 V		
$\overline{\text{BR}}$/ VF[1]/ IWP[2]	$\overline{\text{BR}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	Controlled by DBGC in reset config word. See **Table 6-8**.
	VF[1][6]	O	2.6 V		
	IWP[2][6]	O	2.6 V		
$\overline{\text{BB}}$/ VF[2]/ IWP[3]	$\overline{\text{BB4}}$[6]	I/O	2.6 V	PU_2.6V when driver not enabled or until SPRDS is set	Controlled by DBGC in reset config word. See **Table 6-8**
	VF[2][6]	O	2.6 V		
	IWP[3][6]	O	2.6 V		
IWP[0:1]/ VFLS[0:1]	IWP[0:1][6]	O	2.6 V	Output Only, no weak pull	Controlled by DBGC in the reset config word. See **Table 6-8**.
	VFLS[0:1][6]	O	2.6 V		
TMS	TMS	I	2.6 V	PU_2.6V until SPULL_DIS1 is set	TMS
TDI/ DSDI	TDI	I	2.6 V	PU_2.6V until SPULL_DIS1 is set	Controlled by DBPC in reset config word. See **Table 6-9**
	DSDI	I	2.6 V		
TCK/ DSCK	TCK	I	2.6 V	PD until SPULL_DIS1 is set	Controlled by DBPC in reset config word. See **Table 6-9**
	DSCK	I	2.6 V		
TDO/ DSDO	TDO[6]	O	2.6 V	—	Controlled by DBPC in reset config word. See **Table 6-9**
	DSDO[6]	O	2.6 V		
JCOMP	JCOMP[13]	I	2.6 V	—	JCOMP

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
XTAL[8]	XTAL	I	2.6 V	—	XTAL
EXTAL[8]	EXTAL	I	2.6 V	—	EXTAL
XFC	XFC	I	2.6 V	—	XFC
CLKOUT	CLKOUT[6]	O	2.6 V	—	CLKOUT
EXTCLK[8]	EXTCLK	I	2.6 V	—	EXTCLK
ENGCLK/ BUCLK	ENGCLK[3, 6]	O	2.6 / 5 V	—	ENGCLK (2.6 V)
	BUCLK[6]	O	2.6 V	—	
VDDSYN	VDDSYN	I	2.6 V	—	VDDSYN
VSSSYN	VSSSYN	I	0 V	—	VSSSYN
PULL_SEL[14]	PULL_SEL	I	5 V	PU_5	PULL_SEL
QSMCM_A/QSMCM_B/DLCMD2					
A_PCS[0]/$\overline{\text{SS}}$/ QGPIO[0]	PCS[0]	I/O	5 V	PU_5V until PRDS is set	A_QGPIO[0]
	$\overline{\text{SS}}$	I/O	5 V		
	QGPIO[0]	I/O	5 V		
A_PCS[1:3]/ QGPIO[1:3]	PCS[1:3]	I/O	5 V	PU_5V until PRDS is set	A_QGPIO[1:3]
	QGPIO[1:3]	I/O	5 V		
A_MISO/ QGPIO[4]	MISO	I/O	5 V	PU_5V until PRDS is set	A_QGPIO[4]
	QGPIO[4]	I/O	5 V		
A_MOSI/ QGPIO[5]	MOSI	I/O	5 V	PU_5V until PRDS is set	A_QGPIO[5]
	QGPIO[5]	I/O	5 V		
A_SCK/ QGPIO[6]	SCK	I/O	5 V	PU_5V until PRDS is set	A_QGPIO[6]
	QGPIO[6]	I/O	5 V		
A_TXD[1:2]/ QGPO[1:2]	TXD[1:2]	O	5 V	PU_5V until PRDS is set	A_QGPO[1:2]
	QGPO[1:2]	O	5 V		
A_RXD[1:2]/ QGPI[1:2]	RXD[1:2]	I	5 V	None	A_QGPI[1:2]
	QGPI[1:2]	I	5 V		
B_PCS[0]/ $\overline{\text{SS}}$/ QGPIO[0]	PCS[0]	I/O	5 V	PU_5V until PRDS is set	B_QGPIO[0]
	$\overline{\text{SS}}$	I/O	5 V		
	QGPIO[0]	I/O	5 V		
B_PCS[1:2]/ QGPIO[1:2]	PCS[1:2]	I/O	5 V	PU_5V until PRDS is set	B_QGPIO[1:2]
	QGPIO[1:2]	I/O	5 V		
B_PCS[3]/ J1850_TX	PCS[3]	I/O	5 V	PU_5V until PRDS is set	J1850_TX
	J1850_TX	I/O	5 V		
B_MISO/ QGPIO[4]	MISO	I/O	5 V	PU_5V until PRDS is set	B_QGPIO[4]
	QGPIO[4]	I/O	5 V		
B_MOSI/ B_QGPIO[5]	MOSI	I/O	5 V	PU_5V until PRDS is set	B_QGPIO[5]
	QGPIO[5]	I/O	5 V		
B_SCK/QGPIO[6]	SCK	I/O	5 V	PU_5V until PRDS is set	B_QGPIO[6]
	QGPIO[6]	I/O	5 V		

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
B_TXD[1:2]/ QGPO[1:2]	TXD[1:2]	O	5 V	PU_5V until PRDS is set	B_QGPO[1:2]
	QGPO[1:2]	O	5 V		
B_RXD[1]/ QGPI[1]	RXD[1]	I	5 V	None	B_QGPI[1]
	QGPI[1]	I	5 V		
B_RXD[2]/ J1850_RX	RXD[2]	I	5 V	None	J1850_RX
	J1850_RX	I	5 V		
B_ECK	B_ECK	I	5 V	PU_5V until PRDS is set	—
MIOS14					
MDA[11:15], [27:31]	MDA[11:15], [27:31]	I/O	5 V	Pull device enabled until PULL_DIS[0] is set [15]	MDA[11:15], [27:31]
MPWM[0:3], [16:19]	MPWM[0:3], [16:19]	I/O	5 V	Pull device enabled until PULL_DIS[0] is set [15]	MPWM[0:3], [16:19]
VF[0:2]/ MPIO32B[0:2]	VF[0:2][6]	O	2.6 V	Pull down until PULL_DIS[0] is set	MPIO32B[0:2]
	MPIO32B[0:2][3]	I/O	5 V		
VFLS[0:1]/ MPIO32B[3:4]	VFLS[0:1][6]	O	2.6 V	Pull down until PULL_DIS[0] is set	MPIO32B[3:4]
	MPIO32B[3:4][3]	I/O	5 V		
MPWM[4:5]/ MPIO32B[5:6]	MPWM[4:5]	I/O	5 V	Pull device enabled until PULL_DIS[0] is set[15]	MPIO32B[5:6]
	MPIO32B[5:6]	I/O	5 V		
MDO[7:4]/ MPIO32B[7:10]	MDO[7:4][6]	O	2.6 V	Pull down until PULL_DIS[0] is set	Controlled by READI enable ($\overline{\text{EVTI}}$ and MDI[0])
	MPIO32B[7:10][3]	I/O	5 V		
MPWM[20:21]/ MPIO32B[11:12]	MPWM[20:21]	I/O	5 V	Pull device enabled until PULL_DIS[0] is set[15]	MPIO32B[11:12]
	MPIO32B[11:12]	I/O	5 V		
C_CNTX0/ MPIO32B[13]	C_CNTX0	I/O	5 V	Pull device enabled until PULL_DIS[3] is set[16]	MPIO32B[13]
	MPIO32B[13]	I/O	5 V		
C_CNRX0/ MPIO32B[14]	C_CNRX0	I/O	5 V	Pull device enabled until PULL_DIS[3] is set[16]	MPIO32B[14]
	MPIO32B[14]	I/O	5 V		
MPIO32B[15]	MPIO32B[15]	I/O	5 V	Pull device enabled until PULL_DIS[0] is set[15]	MPIO32B[15]
EXTAL32	EXTAL32	I	2.6 V	—	EXTAL32
XTAL32	XTAL32	I	2.6 V	—	XTAL32
VDDRTC	VDDRTC	I	2.6 V	—	VDDRTC
VSSRTC	VSSRTC	I	0 V	—	VSSRTC
TPU3_A/TPU3_B/TPU3_C					
A_TPUCH[0:15]	A_TPUCH[0:15]	I/O	5 V	Pull device enabled until PRDS is set[17]	A_TPUCH[0:15]
A_T2CLK	A_T2CLK	I/O	5 V	Pull up enabled until T2CLK_PU is set	A_T2CLK
B_TPUCH[0:15]	B_TPUCH[0:15]	I/O	5 V	Pull device enabled until PRDS is set[17]	B_TPUCH[0:15]
B_T2CLK	B_T2CLK	I/O	5 V	Pull up enabled until T2CLK_PU is set	B_T2CLK

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
C_TPUCH[0:15]	C_TPUCH[0:15]	I/O	5 V	Pull device enabled until PRDS is set[17]	C_TPUCH[0:15]
C_T2CLK	C_T2CLK	I/O	5 V	Pull up enabled until T2CLK_PU is set	C_T2CLK
QADC64_A/QADC64_B					
ETRIG[1:2]	ETRIG[1:2]	I	5 V	PD until PULL_DIS0 is set	ETRIG[1:2]
AN[44]/ANW/ A_PQB[0]	AN[44]	I	5 V	PU_5V until PRDS is set	AN[44]
	ANW	I	5 V	PU_5V until PRDS is set	
	A_PQB[0]	I	5 V	PU_5V until PRDS is set	
AN[45]/ANX/ A_PQB[1]	AN[45]	I	5 V	PU_5V until PRDS is set	AN[45]
	ANX	I	5 V	PU_5V until PRDS is set	
	A_PQB[1]	I	5 V	PU_5V until PRDS is set	
AN[46]/ANY/ A_PQB[2]	AN[46]	I	5 V	PU_5V until PRDS is set	AN[46]
	ANY	I	5 V	PU_5V until PRDS is set	
	A_PQB[2]	I	5 V	PU_5V until PRDS is set	
AN[47]/ANZ/ A_PQB[3]	AN[47]	I	5 V	PU_5V until PRDS is set	AN[47]
	ANZ	I	5 V	PU_5V until PRDS is set	
	A_PQB[3]	I	5 V	PU_5V until PRDS is set	
AN[48:51]/ A_PQB[4:7]	AN[48:51]	I	5 V	PU_5V until PRDS is set	AN[48:51]
	A_PQB[4:7]	I	5 V	PU_5V until PRDS is set	
AN[52:54]/ A_MA[0:2]/ PQA[0:2]	AN[52:54]	I	5 V	PU_5V until PRDS is set	AN[52:54]
	A_MA[0:2]	I	5 V	PU_5V until PRDS is set	
	A_PQA[0:2]	I/O	5 V	PU_5V until PRDS is set	
AN[55:59]/ A_PQA[3:7]	AN[55:59]	I	5 V	PU_5V until PRDS is set	AN[55:59]
	A_PQA[3:7]	I/O	5 V	PU_5V until PRDS is set	
AN[64:71]/ B_PQB[0:7]	AN[64:71]	I	5 V	PU_5V until PRDS is set	AN[64:71]
	B_PQB[0:7]	I	5 V	PU_5V until PRDS is set	
AN[72:74]/ B_MA[0:2]/ B_PQA[0:2]	AN[72:74]	I	5 V	PU_5V until PRDS is set	AN[72:74]
	B_MA[0:2]	I	5 V	PU_5V until PRDS is set	
	B_PQA[0:2]	I/O	5 V	PU_5V until PRDS is set	
AN[75:79]/ B_PQA[3:7]	AN[75:79]	I	5 V	PU_5V until PRDS is set	AN[75:79]
	B_PQA[3:7]	I/O	5 V	PU_5V until PRDS is set	
AN[80:87]	AN[80:87]	I	5 V	None	AN[80:87]
VRH	VRH	I	5 V	—	VRH
VRL	VRL	I	—	—	VRL
VDDA	VDDA	I	5 V	—	VDDA
VSSA	VSSA	I	—	—	VSSA
TouCAN_A/TouCAN_B					
A_CNTX0	A_CNTX0	O	5 V	PU_5V until PULL_DIS2 is set	A_CNTX0
B_CNTX0	B_CNTX0	O	5 V	PU_5V until PULL_DIS2 is set	B_CNTX0
A_CNRX0	A_CNRX0	I	5 V	PU_5V until PULL_DIS2 is set	A_CNRX0

Table 2-6 Pin Reset State (Continued)

Pin	Function	Port	Voltage	Reset State	Function After $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ or $\overline{\text{HRESET}}$
B_CNRX0	B_CNRX0	I	5 V	PU_5V until PULL_DIS2 is set	B_CNRX0
UC3F					
EPEE	EPEE	I	2.6 V	PU_2.6V	EPEE
B0EPEE	B0EPEE	I	2.6 V	PU_2.6V	B0EPEE
VFLASH	VFLASH	I	5 V	—	VFLASH
VDDF	VDDF	I	2.6 V	—	VDDF
VSSF	VSSF	I	2.6 V	—	VSSF
READI					
$\overline{\text{MSEO}}$	$\overline{\text{MSEO}}$[6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	$\overline{\text{MSEO}}$
MDO[3]	MDO[3][6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	MDO[3]
MDO[2]	MDO[2][6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	MDO[2]
MDO[1]	MDO[1][6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	MDO[1]
MDO[0]	MDO[0][6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	MDO[0]
MCKO	MCKO[6]	O	2.6 V	PU_2.6V while $\overline{\text{RSTI}}$ is asserted	MCKO
$\overline{\text{RSTI}}$	$\overline{\text{RSTI}}$	I	2.6 V	PD until PULL_DIS4 is set	$\overline{\text{RSTI}}$
$\overline{\text{EVTI}}$	$\overline{\text{EVTI}}$	I	2.6 V	PU_2.6V until PULL_DIS4 is set	$\overline{\text{EVTI}}$
$\overline{\text{MSEI}}$	$\overline{\text{MSEI}}$	I	2.6 V	PU_2.6V until PULL_DIS4 is set	$\overline{\text{MSEI}}$
MDI[1]	MDI[1]	I	2.6 V	PU_2.6V until PULL_DIS4 is set	MDI[1]
MDI[0]	MDI[0]	I	2.6 V	PU_2.6V until PULL_DIS4 is set	MDI[0]
MCKI	MCKI	I	2.6 V	PU_2.6V until PULL_DIS4 is set	MCKI
Global Power Supplies					
QVDDL	QVDDL	I	2.6 V	—	QVDDL
NVDDL	NVDDL	I	2.6 V	—	NVDDL
VDDH	VDDH	I	5 V	—	VDDH
VDD	VDD	I	2.6 V	—	VDD
VSS	VSS	I	0 V	—	VSS
KAPWR[8]	KAPWR	I	2.6 V	—	KAPWR
VDDSRAM[1:3]	VDDSRAM[1:3]	I	2.6 V	—	VDDSRAM[1:3]

NOTES:

1. 2.6-V outputs cannot be connected to a pull-up or driver greater than 3.1 V.
2. During reset, the output enable to the pad driver is negated and the PD is active. After reset is negated, the PD is disabled.
3. Care should be taken that neither a pull-up to greater than 3.1 V or an external output that can drive greater than 3.1 V is connected to this pin while the 2.6-V driver is enabled.
4. SGPIO[0] was 5 V on mask set K85H.
5. This pin requires a pull-up to 2.6 V if interrupts are ever enabled for this IRQ input.
6. 2.6-V outputs cannot be connected to a pull-up or driver greater than 3.1 V.
7. Pull-up/pull-down is active when pin is defined as an input and/or during reset; therefore, output enable is negated. This also means that external pull-up/pull-down is *not* required unless specified.
8. These pins are powered by KAPWR (keep-alive power supply).

9. The MODCK[1:3] are shared functions with $\overline{\text{IRQ}}$[5:7]. If $\overline{\text{IRQ}}$[5:7] are used as interrupts, the interrupt source should be removed during $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$ to insure the MODCK pins are in the correct state on the rising edge of $\overline{\text{PORESET}}$/$\overline{\text{TRST}}$.
10. This pin is an active negate signal and require an external pull-up resister. On dual-voltage type 4 pads, hysteresis is always enabled on the 5-V input buffer, and hysteresis affects only the 2.6-V input buffers on which hysteresis is enabled.
11. This pin was $\overline{\text{PORESET}}$ only on mask set K85H.
12. This pin is 5-V tolerant, even when in output mode.
13. This pin was $\overline{\text{TRST}}$ in mask set K85H.
14. This pin was A_ECK in mask set K85H.
15. Whether the Pull device is a pull-up or a pull-down is determined by the state of the PULL_SEL pin.
16. Whether the Pull device is a pull-up or a pull-down is determined by the state of the PULL_SEL pin.
17. Whether the Pull device is a pull-up or a pull-down is determined by the state of the PULL_SEL pin.

2.6 Pad Types

There are different pad types based on functional characteristics. Even pads with the same functionality may be different due to different electrical characteristics. All 5-V inputs have hysteresis. Refer to **Figure 2-1**.

Table 2-7 Functional Pad Types

Type	Name	V	Dir.	Delay (ns)		Drive Strength (Rise/Fall)	Drive Load	Weak Devices	Hyst.
				In	Out[1]	Fast; Slow	Fast; Slow	PU; PD	
1	pad_5vsa	5 V	IO	5	17 ; 400	10 ns ; 200 ns	50 pF ; 50 pF	Selectable	Selectable
2	pad_5vfa	5 V	IO	5	17 ; 100	10 ns ; 40 ns	50 pF ; 50 pF	Selectable	Selectable
					16 ; 100	9 ns ; 40 ns	45 pF ; 45 pF		
					27 ; 110	18 ns ; 47 ns	90 pF ; 90 pF		
3	pad_26v	2.6 V	IO	4	5 ; 5	2.9 ns ; 2.9 ns	25 pF ; 50 pF	Selectable	Selectable
4	pad_26v5vs	2.6 V	IO	4	5 ; 5	2.9 ns ; 2.9 ns	25 pF ; 50 pF	PD Selectable	Selectable
		5 V	IO	5	17 ; 400	10 ns ; 200 ns	50 pF ; 50 pF	PD Selectable	Enabled
5	pad_5vh	5 V	IO	5	30 ; 400	21 ns ; 200 ns	200 pF ; 50 pF	Selectable	Selectable
6	pad_5vido	5 V	I	5	NA	NA	NA	NA	Selectable
7	pad_extclk	2.6 V	I		NA	NA	NA	NA	NA
8	pad_e_xtal_oscpll	2.6 V	IO	NA	NA	NA ; NA	NA ; NA	NA	NA
9	pad_e_xtal_osc32	2.6 V	IO	NA	NA	NA ; NA	NA ; NA	NA	NA
10	pad_26vf	2.6 V	IO	4	5 ; 5	2.3 ns ; 2.3 ns	25 pF ; 50 pF	Selectable	Selectable
11	pad_26vs5vr	2.6 V	IO	4	11 ; 11	7 ns ; 7 ns	25 pF ; 50 pF	PD Selectable	Selectable
		5 V	IO	5	12 ; 50	6 ns ; 25 ns	25 pF ; 25 pF	PD Selectable	Enabled

NOTES:

1. Output delay includes pad prop delay (out) + rise/fall time.

Table 2-8 Other Pad Types

Name	V	Used For
pad_vddh	5 V	VDDH
pad_nvddl	2.6 V	NVDDL
pad_qvddl	2.6 V	QVDDL
pad_vddint	2.6 V	VDD, VDDSRAM[1:3], VDDF, VDDSYN, VDDRTC
pad_vdda	5 V	VDDA
pad_vss		VSS
pad_vssint		VSSA, VSSF, VRL
pad_vsssyn		VSSSYN, VSSRTC
pad_vflashvrh	5 V	VFLASH, VRH, ALTREF
pad_kapwr	2.6 V	KAPWR
pad_xfc	2.6 V	XFC
pad_esdcnrcell		CORNER

2.7 Electrical Characteristics

2.7.1 Target DC Characteristics

Table 2-9 Target DC Electrical Characteristics

Spec	Characteristic	Symbol	Min	Max	Unit
1	2.6-V Supply Voltage	VDDI	2.5	2.7	V
2	5-V Supply Voltage	VDDH	4.75	5.25	V
3	2.6-V Input High Voltage	VIH_L	2.0	VDDH + 0.3	V
4	5-V Input High Voltage	VIH_H	0.7 * VDDH	VDDH + 0.3	V
5	2.6-V Input Low Voltage	VIL_L	VSS - 0.3	0.8	V
6	5-V Input Low Voltage	VIL_H	VSS - 0.3	0.48 * VDDH	V
7	Input Hysteresis	VHYS	500 (target spec)		mV
8	Mode Select/Pull-up/Pull-down Current	IACT	20	130	µA
9	Input Leakage Current (pull-up/pull-down inactive)	IINACT	—	2.5	µA
10	2.6-V Output High Voltage [IOH_L = -1 mA] (IOH_L = -2 mA)	VOH_L	[2.4] [2.3]	—	V
11	5-V Output High Voltage (IOH_H = -2 mA)	VOH_H	VDDH - 0.7	—	V
12	2.6-V Output Low Voltage [IOL_L = 3.2 mA] (Iol_l =2mA)	VOL_L	—	[0.5] [0.45]	V
13	5-V Output Low Voltage (IOL_H = 2 mA)	VOL_H	—	0.5	V

These DC characteristics are not all inclusive and any individual pad. Electrical characteristics may have other requirements. In case of a conflict, the **APPENDIX E ELECTRICAL CHARACTERISTICS** gives the actual electrical characteristics of any particular pin, but **Table 2-9** can be useful to show general capabilites of the pads.

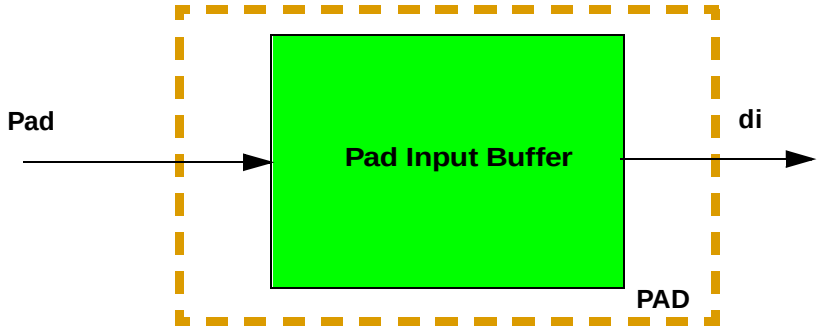
2.7.2 Pad Measurement Criteria

- Pad rise/fall times measured between VOL and VOH of output driver. (For 2.6-V drivers, lowest minimum VOH is used.)
- Pad prop delays for outputs are measured at VDD/2 for internal signals to VOL or VOH at pin load.
- Propagation delays for inputs are measured from VIL for a falling input or VIH for a rising input at the pin, to VDD/2 at internal signal load. Rise/fall times for the input signal at the internal signal load are measured from 10% to 90% of VDD. The rise/fall time for the input signal at the pad shall be two ns for the full swing from VSS to VDD.
- All inputs except the oscillator are 5-V friendly.

- Minimum input buffer hysteresis is 500 mV.
- Weak pull-up current is measured at VIL and weak pull-down current is measured at VIH.

2.7.2.1 Input Buffer

1. Voltages received above VIH min. guarantee a logical “1” response.
2. Voltages received below VIL max guarantee a logical “0” response.
3. Received voltages that fall between the two thresholds may be interpreted by the input buffer as a “1”, as a “0”, or as an indeterminate state.



Input prop delay measured from pad to di_line.

Figure 2-1 Input Propagation Delay

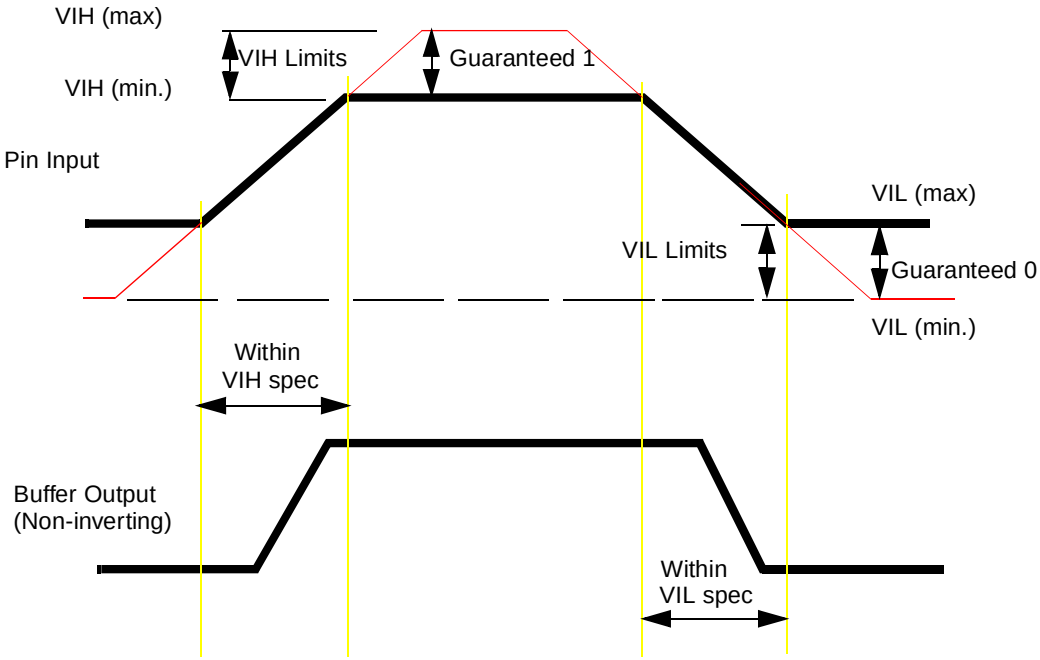
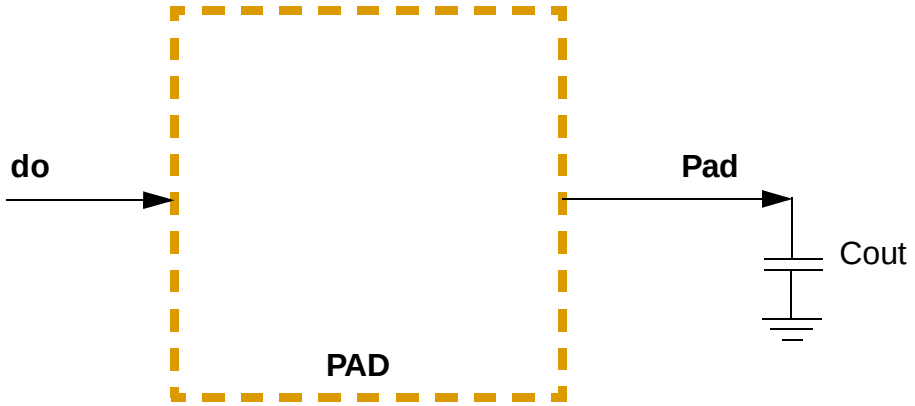


Figure 2-2 Input Levels

2.7.2.2 Output Driver



NOTE: Package parasitic resistance and inductance have been added to simulation delay path.

Figure 2-3 Output Propagation Delay

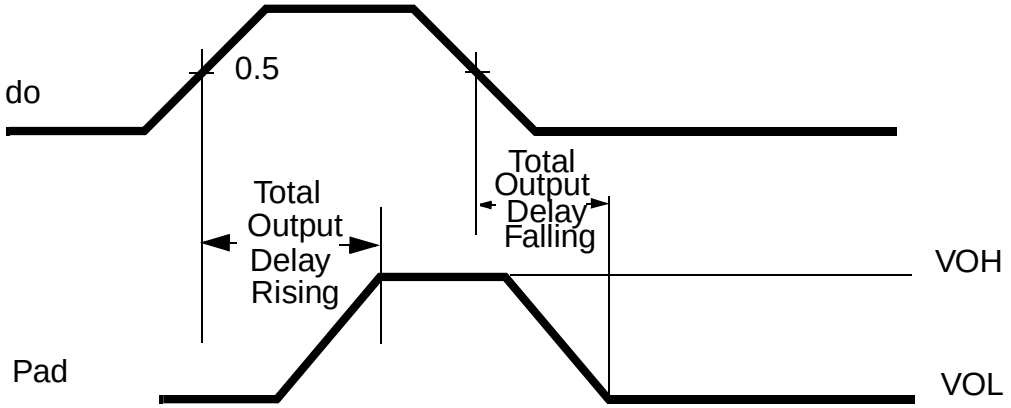


Figure 2-4 Output Levels and Timing

Table 2-10 Pin Names and Abbreviations

Pin List	Pin Name	Ball	Pad Type
DATA[0:31]/SGPIOD[0:31]	data_sgpiod[0]	AF3	26v5vs
	data_sgpiod[1]	AE4	26v5vs
	data_sgpiod[2]	AF4	26v5vs
	data_sgpiod[3]	AE5	26v5vs
	data_sgpiod[4]	AF5	26v5vs
	data_sgpiod[5]	AE6	26v5vs
	data_sgpiod[6]	AF6	26v5vs
	data_sgpiod[7]	AE7	26v5vs
	data_sgpiod[8]	AF7	26v5vs
	data_sgpiod[9]	AE8	26v5vs
	data_sgpiod[10]	AF8	26v5vs
	data_sgpiod[11]	AE9	26v5vs
	data_sgpiod[12]	AF9	26v5vs
	data_sgpiod[13]	AE10	26v5vs
	data_sgpiod[14]	AF10	26v5vs
	data_sgpiod[15]	AE11	26v5vs
	data_sgpiod[16]	AF11	26v5vs
	data_sgpiod[17]	AE12	26v5vs
	data_sgpiod[18]	AF12	26v5vs
	data_sgpiod[19]	AD13	26v5vs
	data_sgpiod[20]	AC12	26v5vs
	data_sgpiod[21]	AD12	26v5vs
	data_sgpiod[22]	AC11	26v5vs
	data_sgpiod[23]	AD11	26v5vs
	data_sgpiod[24]	AC10	26v5vs
	data_sgpiod[25]	AD10	26v5vs
	data_sgpiod[26]	AD9	26v5vs
	data_sgpiod[27]	AC8	26v5vs
	data_sgpiod[28]	AD8	26v5vs
	data_sgpiod[29]	AC7	26v5vs
	data_sgpiod[30]	AD7	26v5vs
	data_sgpiod[31]	AD6	26v5vs

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
ADDR[8:31]/SGPIOA[8:31]	addr_sgpioa[8]	U3	26v5vs
	addr_sgpioa[9]	V3	26v5vs
	addr_sgpioa[10]	V4	26v5vs
	addr_sgpioa[11]	W3	26v5vs
	addr_sgpioa[12]	W4	26v5vs
	addr_sgpioa[13]	Y3	26v5vs
	addr_sgpioa[14]	Y4	26v5vs
	addr_sgpioa[15]	AA3	26v5vs
	addr_sgpioa[16]	U1	26v5vs
	addr_sgpioa[17]	U2	26v5vs
	addr_sgpioa[18]	V1	26v5vs
	addr_sgpioa[19]	V2	26v5vs
	addr_sgpioa[20]	W1	26v5vs
	addr_sgpioa[21]	W2	26v5vs
	addr_sgpioa[22]	Y1	26v5vs
	addr_sgpioa[23]	Y2	26v5vs
	addr_sgpioa[24]	AA1	26v5vs
	addr_sgpioa[25]	AA2	26v5vs
	addr_sgpioa[26]	AB1	26v5vs
	addr_sgpioa[27]	AB2	26v5vs
	addr_sgpioa[28]	AC1	26v5vs
	addr_sgpioa[29]	AD1	26v5vs
	addr_sgpioa[30]	AA4	26v5vs
	addr_sgpioa[31]	AB3	26v5vs
$\overline{\text{IRQ}}$[0]/SGPIOC[0]	irq0_b_sgpioc0	AF16	26v
$\overline{\text{IRQ}}$[1]/$\overline{\text{RSV}}$/SGPIOC[1]	irq1_b_rsv_b_sgpioc1	AF13	26v5vs
$\overline{\text{IRQ}}$[2]/CR/SGPIOC[2]/MTS	irq2_b_cr_b_sgpio2_mts	AD16	26v5vs
$\overline{\text{IRQ}}$[3]/$\overline{\text{KR}}$, $\overline{\text{RETRY}}$/SGPIOC[3]	irq3_b_kr_b_retry_b_sgpioc3	AE13	26v5vs
$\overline{\text{IRQ}}$[4]/AT[2]/SGPIOC[4]	irq4_b_at2_sgpioc4	AD14	26v5vs
$\overline{\text{IRQ}}$[5]/SGPIOC[5]/MODCK[1]	irq5_b_sgpioc5_modck1	AD25	26v5vs
$\overline{\text{IRQ}}$[6:7]/MODCK[2:3]	irq6_b_modck2	AB24	26v
	irq7_b_modck3	AC25	26v
TSIZ[0:1]	tsiz0	AD19	26v
	tsiz1	AF19	26v
RD/$\overline{\text{WR}}$	rd_wr_b	AE15	26v
$\overline{\text{BURST}}$	burst_b	AE19	26v
$\overline{\text{BDIP}}$	bdip_b	AE21	26v
$\overline{\text{TS}}$	ts_b	AE20	26v
$\overline{\text{TA}}$	ta_b	AF20	26v
$\overline{\text{TEA}}$	tea_b	AD15	26v
$\overline{\text{RSTCONF}}$/TEXP	rstconf_b_texp	AB25	26v
$\overline{\text{OE}}$	oe_b	AE16	26v

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
$\overline{BI}/\overline{STS}$	bi_b_sts_b	AC19	26v
$\overline{CS}$[0:3]	cs0_b	AE18	26v
	cs1_b	AD18	26v
	cs2_b	AF18	26v
	cs3_b	AC18	26v
$\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3]	we_b_be_b_at[0]	AE17	26v
	we_b_be_b_at[1]	AC16	26v
	we_b_be_b_at[2]	AD17	26v
	we_b_be_b_at[3]	AF17	26v
$\overline{PORESET}/\overline{TRST}$[1]	poreset_b_trst_b	AA23	26v
$\overline{HRESET}$	hreset_b	AB23	26v (Rev0); 5vsa (RevA)
$\overline{SRESET}$	sreset_b	AC24	26V (Rev0); 5vsa (RevA)
SGPIOC[6]/FRZ/$\overline{PTR}$	sgpioc6_frz_ptr_b	T4	26v5vs
SGPIOC[7]/$\overline{IRQOUT}$/LWP[0]	sgpioc7_irqout_b_lwp0	AC14	26v5vs
$\overline{BG}$/VF[0]/LWP[1]	bg_b_vf0_lwp1	AF14	26v
$\overline{BR}$/VF[1]/IWP[2]	br_b_vf1_iwp2	AF15	26v
$\overline{BB}$/VF[2]/IWP[3]	bb_b_vf2_iwp3	AE14	26v
IWP[0:1]/VFLS[0:1]	iwp0_vfls0	T3	26v
	iwp1_vfls1	R4	26v
TMS	tms	N1	26v
TDI/DSDI	tdi_dsdi	M1	26v
TCK/DSCK	tck_dsck	L2	26v
TDO/DSDO	tdo_dsdo	R2	26v
JCOMP[2]	jcomp_b	P2	26v
XTAL	xtal	AA26	extal_oscpll
EXTAL	extal	AB26	extal_oscpll
XFC	xfc	AD26	xfc
CLKOUT	clkout	AD21	26vf
EXTCLK	extclk	Y24	extclk
VDDSYN	vddsyn	AE26	vddint
VSSSYN	vsssyn	AC26	vsssyn
ENGCLK/BUCLK	engclk_buclk	AF22	26v5vsr
PULL_SEL[3]	pullsel	W26	5vfa

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
QSMCM_A			
A_PCS[0]/$\overline{SS}$/QGPIO[0]	a_pcs0_ss_b_qgpio0	V23	5vfa
A_PCS[1:3]/QGPIO[1:3]	a_pcs1_qgpio1	W25	5vfa
	a_pcs2_qgpio2	U23	5vfa
	a_pcs3_qgpio3	U26	5vfa
A_MISO/QGPIO[4]	a_miso_qgpio4	T24	5vh
A_MOSI/QGPIO[5]	a_mosi_qgpio5	U25	5vh
A_SCK/QGPIO[6]	a_sck_qgpio6	T26	5vh
A_TXD[1:2]/QGPO[1:2]	a_txd1_qgpo1	T23	5vsa
	a_txd2_qgpo2	V24	5vsa
A_RXD[1:2]/QGPI[1:2]	a_rxd1_qgpi1	U24	5vido
	a_rxd2_qgpi2	V25	5vido
B_PCS[0]/$\overline{SS}$/QGPIO[0]	b_pcs0_ss_b_qgpio0	N25	5vfa
B_PCS[1:2]/QGPIO[1:2]	b_pcs1_qgpio1	N26	5vfa
	b_pcs2_qgpio2	R24	5vfa
B_PCS[3]/J1850_TX	b_pcs3_j1850_tx	P25	5vfa
B_MISO/QGPIO[4]	b_miso_qgpio4	P24	5vh
B_MOSI/QGPIO[5]	b_mosi_qgpio5	P26	5vh
B_SCK/QGPIO[6]	b_sck_qgpio6	R23	5vh
B_TXD[1:2]/QGPO[1:2]	b_txd1_qgpo1	R25	5vsa
	b_txd2_qgpo2	R26	5vsa
B_RXD[1]/QGPI[1]	b_rxd1_qgpi1	V26	5vido
B_RXD[2]/J1850_RX	b_rxd2_j1850_rx	T25	5vsa
B_ECK	b_eck	P23	5vsa
MIOS14/TOUCAN_C			
MDA[11:15], MDA[27:31]	mda11	F25	5vsa
	mda12	G23	5vsa
	mda13	F26	5vsa
	mda14	K24	5vsa
	mda15	K23	5vsa
	mda27	G24	5vsa
	mda28	G25	5vsa
	mda29	G26	5vsa
	mda30	H23	5vsa
	mda31	H24	5vsa

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
MPWM[0:3], [16:19]	mpwm0	H25	5vsa
	mpwm1	H26	5vsa
	mpwm2	J24	5vsa
	mpwm3	J23	5vsa
	mpwm16	J25	5vsa
	mpwm17	E26	5vsa
	mpwm18	F24	5vsa
	mpwm19	L25	5vsa
VF[0:2]/MPIO32B[0:2]	vf0_mpio32b0	L26	26v5vs
	vf1_mpio32b1	M23	26v5vs
	vf2_mpio32b2	M24	26v5vs
VFLS[0:1]/MPIO32B[3:4]	vfls0_mpio32b3	M26	26v5vs
	vfls1_mpio32b4	N24	26v5vs
MPWM[4:5]/MPIO32B[5:6]	mpwm4_mpio32b5	M25	5vsa
	mpwm5_mpio32b6	F23	5vsa
MDO[7:4]/MPIO32B[7:10]	mdo7_mpio32b7	P1	26v5vs
	mdo6_mpio32b8	N3	26v5vs
	mdo5_mpio32b9	N4	26v5vs
	mdo4_mpio32b10	N2	26v5vs
MPWM[20:21]/MPIO32B[11:12]	mpwm20_mpio32b11	J26	5vsa
	mpwm21_mpio32b12	K25	5vsa
C_CNTX0/MPIO32B[13]	c_cntx0_mpio32b13	K26	5vfa
C_CNRX0/MPIO32B[14]	c_cnrx0_mpio32b14	L23	5vsa
MPIO32B[15]	mpio32b15	L24	5vsa
EXTAL32	extal32	D1	26v
XTAL32	xtal32	E1	26v
VDDRTC	vddrtc	C1	vddint
VSSRTC	vssrtc	F1	vsssyn

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
TPU3_A/TPU3_B/TPU3_C			
A_TPUCH[0:15]	a_tpuch0	D20	5vsa
	a_tpuch1	A21	5vsa
	a_tpuch2	A16	5vsa
	a_tpuch3	D17	5vsa
	a_tpuch4	A17	5vsa
	a_tpuch5	B17	5vsa
	a_tpuch6	A18	5vsa
	a_tpuch7	D18	5vsa
	a_tpuch8	B18	5vsa
	a_tpuch9	C18	5vsa
	a_tpuch10	A19	5vsa
	a_tpuch11	B19	5vsa
	a_tpuch12	C19	5vsa
	a_tpuch13	D19	5vsa
	a_tpuch14	A20	5vsa
	a_tpuch15	C20	5vsa
A_T2CLK	a_t2clk	B20	5vsa
B_TPUCH[0:15]	b_tpuch0	D26	5vsa
	b_tpuch1	E24	5vsa
	b_tpuch2	D25	5vsa
	b_tpuch3	B21	5vsa
	b_tpuch4	C21	5vsa
	b_tpuch5	A22	5vsa
	b_tpuch6	B22	5vsa
	b_tpuch7	B23	5vsa
	b_tpuch8	C23	5vsa
	b_tpuch9	D21	5vsa
	b_tpuch10	A23	5vsa
	b_tpuch11	C22	5vsa
	b_tpuch12	A24	5vsa
	b_tpuch13	B24	5vsa
	b_tpuch14	A25	5vsa
	b_tpuch15	C26	5vsa
B_T2CLK	b_t2clk	E25	5vsa

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
C_TPUCH[0:15]	c_tpuch0	K3	5vsa
	c_tpuch1	K2	5vsa
	c_tpuch2	K1	5vsa
	c_tpuch3	J3	5vsa
	c_tpuch4	K4	5vsa
	c_tpuch5	J2	5vsa
	c_tpuch6	J1	5vsa
	c_tpuch7	H2	5vsa
	c_tpuch8	H3	5vsa
	c_tpuch9	H1	5vsa
	c_tpuch10	G1	5vsa
	c_tpuch11	G2	5vsa
	c_tpuch12	G3	5vsa
	c_tpuch13	J4	5vsa
	c_tpuch14	F2	5vsa
	c_tpuch15	F3	5vsa
C_T2CLK	c_t2clk	H4	5vsa
QADC64_A/QADC64_B			
ETRIG[1:2]	etrig1	C16	5vsa
	etrig2	B16	5vsa
AN[44]/ANw/A_PQB[0]	an44_anw_a_pqb0	B3	5vsa
AN[45]/ANx/A_PQB[1]	an45_anx_a_pqb1	C4	5vsa
AN[46]/ANy/A_PQB[2]	an46_any_a_pqb2	C7	5vsa
AN[47]/ANz/A_PQB[3]	an47_anz_a_pqb3	D8	5vsa
AN[48:51]/A_PQB[4:7]	an48_a_pqb4	A7	5vsa
	an49_a_pqb5	B7	5vsa
	an50_a_pqb6	C8	5vsa
	an51_a_pqb7	D9	5vsa
AN[52:54]/A_MA[0:2]/PQA[0:2]	an52_a_ma0_pqa0	B8	5vsa
	an53_a_ma1_pqa1	A8	5vsa
	an54_a_ma2_pqa2	C9	5vsa
AN[55:59]/A_PQA[3:7]	an55_a_pqa3	D10	5vsa
	an56_a_pqa4	B9	5vsa
	an57_a_pqa5	C10	5vsa
	an58_a_pqa6	B10	5vsa
	an59_a_pqa7	D11	5vsa

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
AN[64:71]/B_PQB[0:7]	an64_b_pqb0	A2	5vsa
	an65_b_pqb1	A14	5vsa
	an66_b_pqb2	B14	5vsa
	an67_b_pqb3	A13	5vsa
	an68_b_pqb4	D14	5vsa
	an69_b_pqb5	B13	5vsa
	an70_b_pqb6	C14	5vsa
	an71_b_pqb7	C13	5vsa
AN[72:74]/B_MA[0:2]/PQA[0:2]	an72_b_ma0_pqa0	A12	5vsa
	an73_b_ma1_pqa1	B12	5vsa
	an74_b_ma2_pqa2	D13	5vsa
AN[75:79]/B_PQA[3:7]	an75_b_pqa3	C12	5vsa
	an76_b_pqa4	A11	5vsa
	an77_b_pqa5	B11	5vsa
	an78_b_pqa6	D12	5vsa
	an79_b_pqa7	C11	5vsa
AN[80:87]	an80	A6	5vsa
	an81	B6	5vsa
	an82	D7	5vsa
	an83	C6	5vsa
	an84	A5	5vsa
	an85	B5	5vsa
	an86	D6	5vsa
	an87	C5	5vsa
VRH	vrh	A3	vflashvrh
VRL	vrl	A4	vssint
ALTREF	altref	B4	vflashvrh
VDDA	vdda	A9	vdda
VSSA	vssa	A10	vssint
TOUCAN_A/TOUCAN_B			
A_CNTX0	a_cntx0	Y25	5vfa
B_CNTX0	b_cntx0	E2	5vfa
A_CNRX0	a_cnrx0	AA24	5vsa
B_CNRX0	b_cnrx0	C17	5vsa
UC3F			
EPEE	epee	AF21	26v
B0EPEE	b0epee	AD20	26v
VFLASH	vflash	W24	vflashvrh
VDDF	vddf	Y23	vddint
VSSF	vssf	AA25	vssint

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
READI			
$\overline{\text{MSEO}}$	mseo_b	T2	26v
MDO[3]	mdo_3	T1	26v
MDO[2]	mdo_2	R3	26v
MDO[1]	mdo_1	R1	26v
MDO[0]	mdo_0	P4	26v
MCKO	mcko	P3	26v
$\overline{\text{RSTI}}$	rsti_b	M3	26v
$\overline{\text{EVTI}}$	evti_b	M2	26v
$\overline{\text{MSEI}}$	msei_b	M4	26v
MDI[1]	mdi_1	L3	26v
MDI[0]	mdi_0	L1	26v
MCKI	mcki	L4	26v
Global Power Supplies			
NVDDL	nvddl	F4, U4, AC9, AC13, D22, W23, AC15, AC17	nvddl
QVDDL	qvddl	AB4, AC3, AD2, AE1, A15, B15, C15, D15, AC23, AE25, AF26, AD24	qvddl
VDDH	vddh	AC6, D16, N23, AC20, D5	vddh
KAPWR	kapwr	Y26	kapwr
VDDSRAM1	vddsram1	E3	vddint
VDDSRAM2	vddsram2	D2	vddint
VDDSRAM3	vddsram3	G4	vddint

Table 2-10 Pin Names and Abbreviations (Continued)

Pin List	Pin Name	Ball	Pad Type
VDD	vdd	D24, E23, AC21, AD22, AE23, AF24, A1, B2, B26, C25, C3, D4, AC5, AD4, AE3, AF2	vddint

Table 2-10 Pin Names and Abbreviations (Continued)

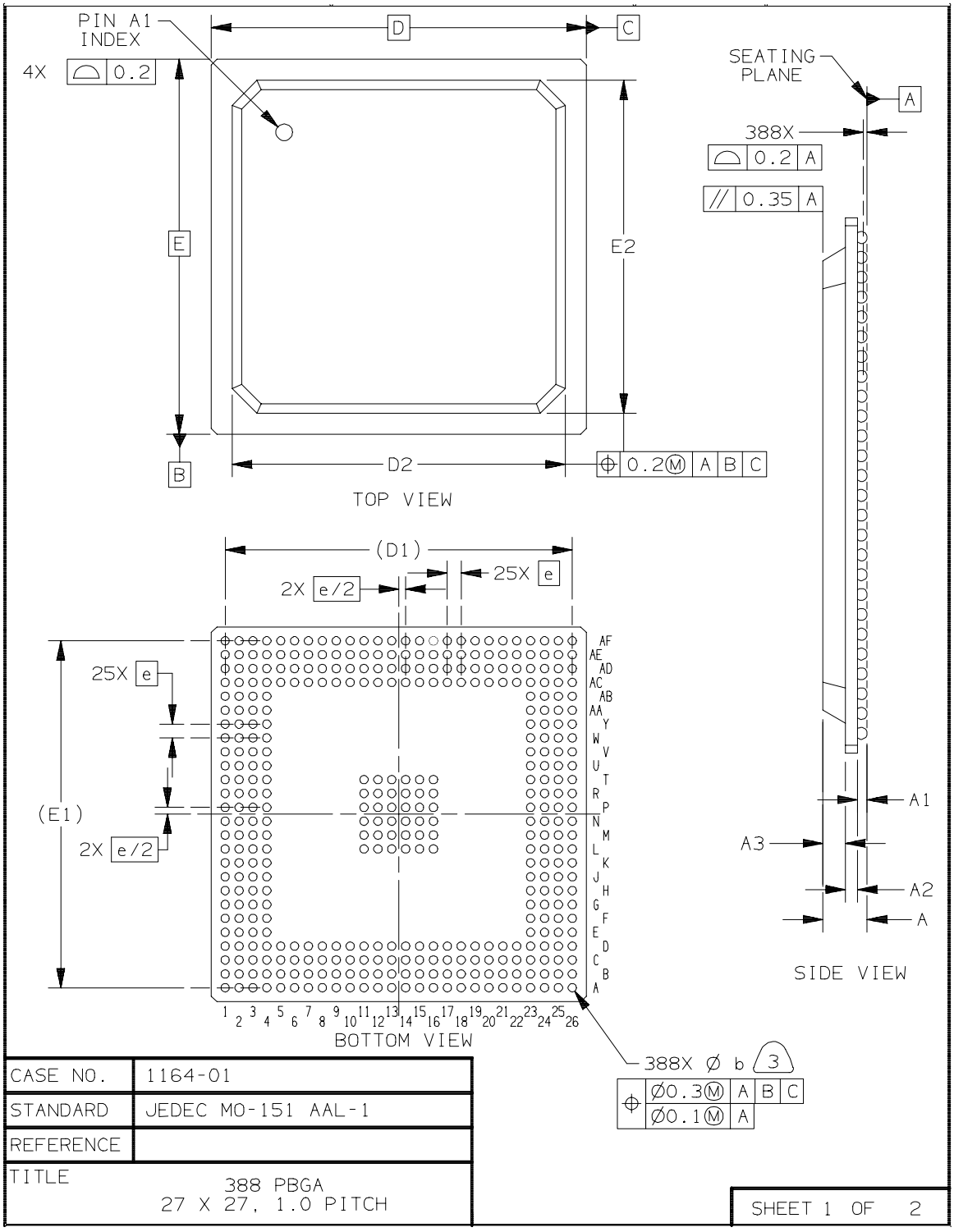
Pin List	Pin Name	Ball	Pad Type
VSS	vss	A26, B25, AC22, D23, B1, N14, N15, N16, D3, M14, M15, M16, C2, E4, L14, L15, L16, P14, P15, P16, AD23, AE24, AF25, L11, L12, L13, M11, M12, M13, N11, N12, N13, P11, P12, P13, R11, R12, R13, T11, T12, T13, AC4, AD3, AE2, AF1, R14, R15, R16, T14, T15, T16, C24	vss
NC	no_connect	AE22, AF23, AC2, AD5	

NOTES:
1. This pin was $\overline{\text{PORESET}}$ only on mask set K85H.
2. This pin was $\overline{\text{TRST}}$ in mask set K85H.
3. This pin was A_ECK in mask set K85H.

2.8 Package Description

2.8.1 Package Diagrams

The package for the MPC565 / MPC566 is the 352/388 PBGA (27 x 27 mm, 1.0 mm ball pitch). This package has 352 balls in the four perimeter rows and 36 ground balls in the center island for a total of 388 balls. The ball footprint is shown in **Figure 2-5**.



NOTE: Top Down View

Figure 2-5 MPC565 / MPC566 Package Footprint (1 of 2)

NOTES

1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M-1994.

2. DIMENSIONS IN MILLIMETERS.

3. DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM A.

4. PRIMARY DATUM A AND THE SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX	DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	1.95	2.55							
A1	0.40	0.60							
A2	0.50	0.70							
A3	1.05	1.25							
b	0.50	0.70							
D	27.00 BSC								
D1	25.00 REF								
D2	23.30	24.70							
E	27.00 BSC								
E1	25.00 REF								
E2	23.30	24.70							
e	1.00 BSC								

CASE NO.	1164-01
STANDARD	JEDEC MO-151 AAL-1
REFERENCE	
TITLE	388 PBGA 27 X 27, 1.0 PITCH

SHEET 2 OF 2

Figure 2-5 MPC565 / MPC566 Package Footprint (2 of 2)

2.8.2 Bumped Die

The MPC565 / MPC566 is available as known good die (KGD) in a bumped die configuration. The bump pitch is seven mils. The bump configuration is shown in **Figure 2-6** through **Figure 2-9**. **Figure 2-7**, along with **Figure 2-8**, shows the categories of function pins. **Figure 2-9** shows the pins in a larger diagram in black and white.

7 mil trace pitch

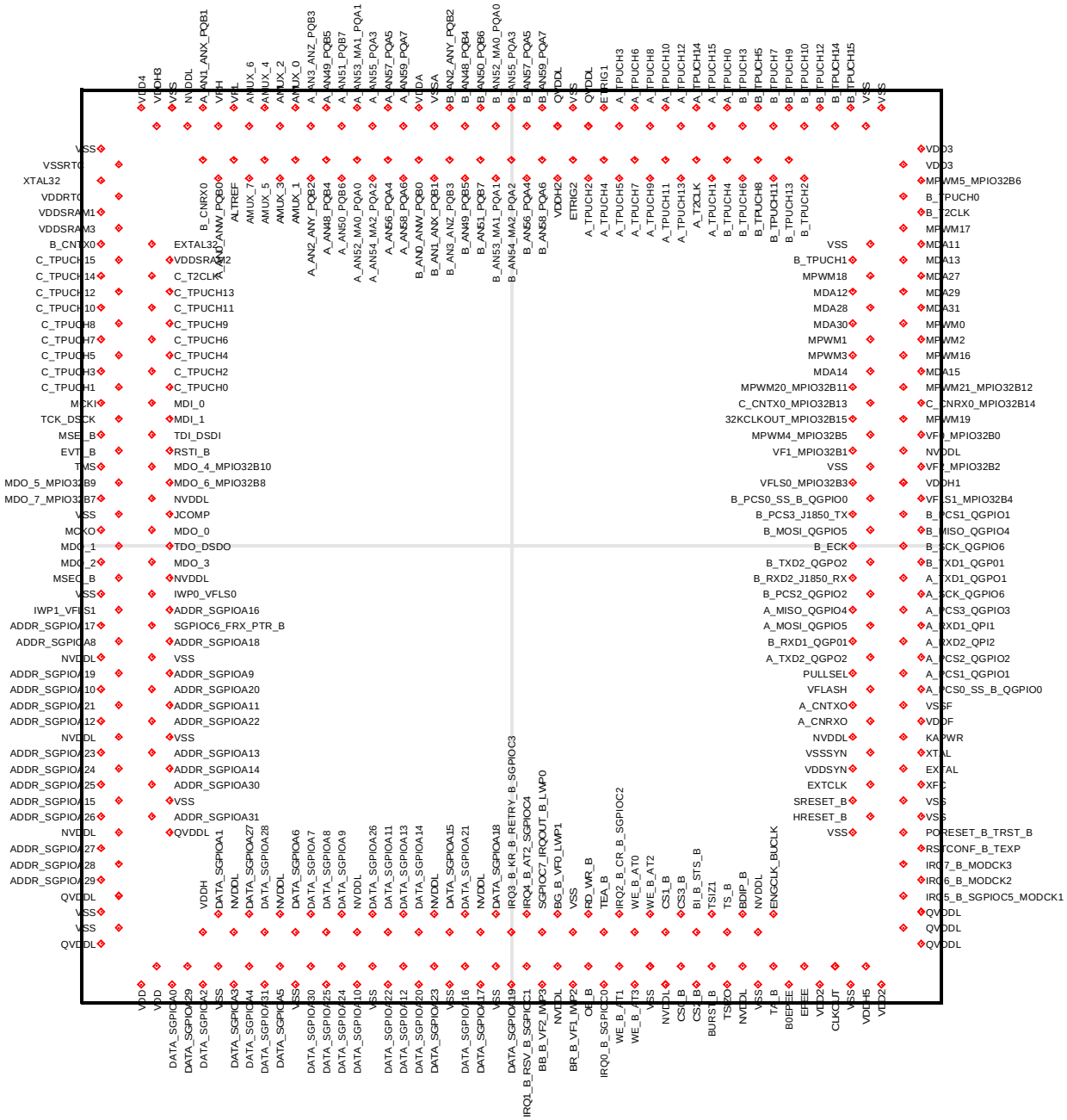


Figure 2-6 MPC565 / MPC566 Redistributed Bump Map

2.8.3 MPC565 / MPC566 Ball Diagram

The ball diagram of the MPC565 / MPC566 is shown in **Figure 2-7**.

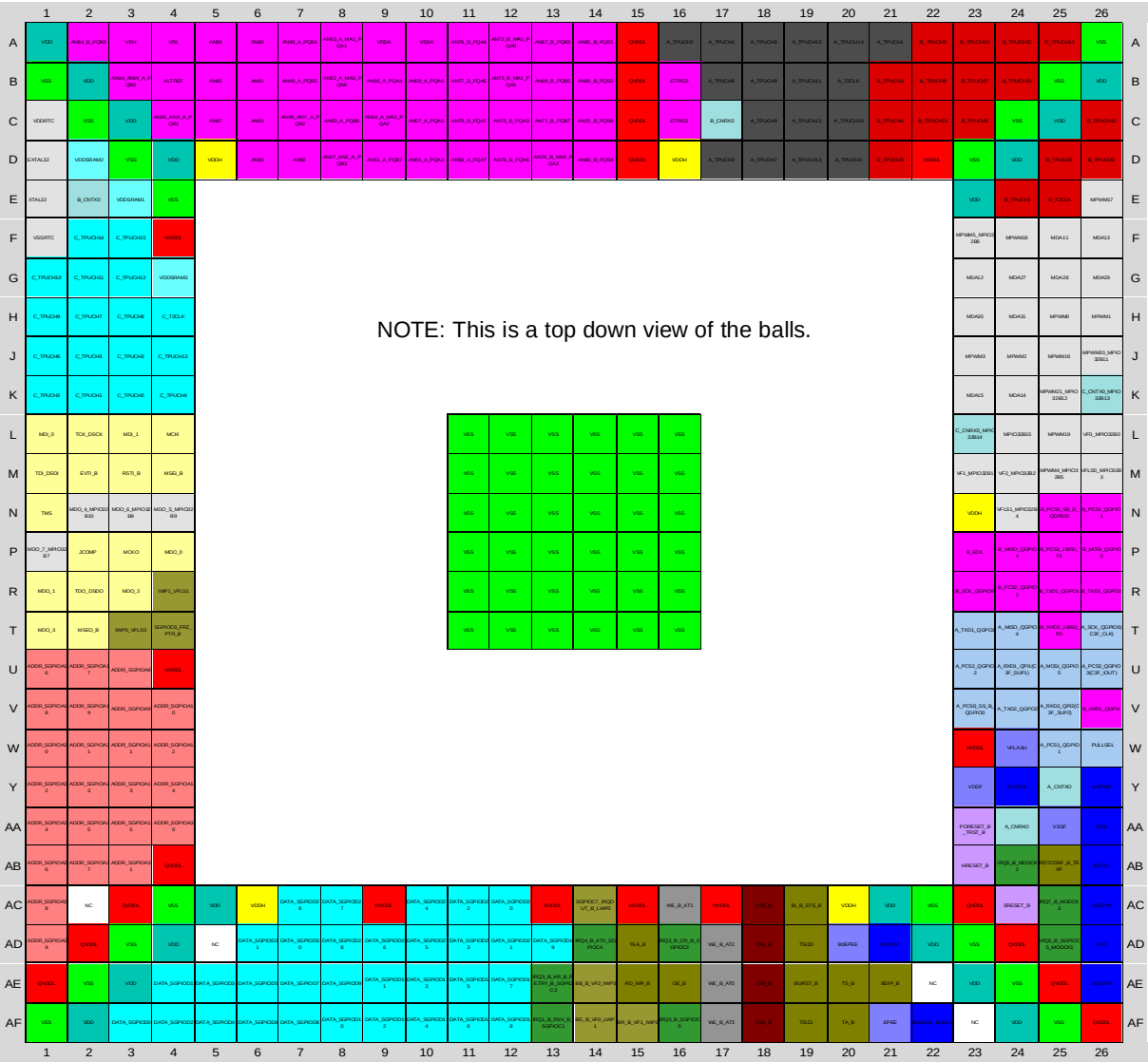


Figure 2-7 MPC565 / MPC566 Ball Diagram

GROUP NAME	COLOR CODE
ADDRESS BUS	
BUS CONTROL	
CHIP SELECT	
CLOCK/PLL	
DATA BUS	
DEBUG	
FLASH	
INTERRUPTS	
JTAG	
MIOS	
NC	
NVDDL	
NVSS	
QADC A	
QADC B	
QSM A	
QSM B	
QVDDL	
READI	
READI	
RESET	
SRAM	
TOUCAN A	
TOUCAN B	
TPU3 A	
TPU3 B	
TPU3 C	
VDD	
VDDH	
VSS	
WRITE ENABLE	

Figure 2-8 MPC565 / MPC566 Ball Diagram Legend

	1	2	3	4	5	6	7	8	9	10	11	12	13
A	VDD	AN64_B_PQB0	VRH	VRL	AN84	AN80	AN48_A_PQB4	AN53_A_MA1_PQA1	VDDA	VSSA	AN76_B_PQA4	AN72_B_MA0_PQA0	AN67_B_PQB3
B	VSS	VDD	AN44_ANW_A_PQB0	ALTREF	AN85	AN81	AN49_A_PQB5	AN52_A_MA0_PQA0	AN56_A_PQA4	AN58_A_PQA6	AN77_B_PQA5	AN73_B_MA1_PQA1	AN69_B_PQB5
C	VDDRTC	VSS	VDD	AN45_ANX_A_PQB1	AN87	AN83	AN46_ANY_A_PQB2	AN50_A_PQB6	AN54_A_MA2_PQA2	AN57_A_PQA5	AN79_B_PQA7	AN75_B_PQA3	AN71_B_PQB7
D	EXTAL32	VDDSRAM2	VSS	VDD	VDDH	AN86	AN82	AN47_ANZ_A_PQB3	AN51_A_PQB7	AN55_A_PQA3	AN59_A_PQA7	AN78_B_PQA6	AN74_B_MA2_PQA2
E	XTAL32	B_CNTX0	VDDSRAM1	VSS									
F	VSSRTC	C_TPUCH14	C_TPUCH15	NVDDL									
G	C_TPUCH10	C_TPUCH11	C_TPUCH12	VDDSRAM3									
H	C_TPUCH9	C_TPUCH7	C_TPUCH8	C_T2CLK									
J	C_TPUCH6	C_TPUCH5	C_TPUCH3	C_TPUCH13									
K	C_TPUCH2	C_TPUCH1	C_TPUCH0	C_TPUCH4									
L	MDI_0	TCK_DSCK	MDI_1	MCKI							VSS	VSS	VSS
M	TDI_DSDI	EVTI_B	RSTI_B	MSEI_B							VSS	VSS	VSS
N	TMS	MDO_4_MPIO32B10	MDO_6_MPIO32B8	MDO_5_MPIO32B9							VSS	VSS	VSS

NOTE: This pinout is a top down view of the package.

Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 1)

P	MDO_7_MPIO32B_7	JCOMP	MCKO	MDO_0							VSS	VSS	VSS
R	MDO_1	TDO_DSDO	MDO_2	IWP1_VFLS1							VSS	VSS	VSS
T	MDO_3	MSEO_B	IWP0_VFLS0	SGPIOC6_FRZ_PTR_B							VSS	VSS	VSS
U	ADDR_SGPIOA16	ADDR_SGPIOA17	ADDR_SGPIOA8	NVDDL									
V	ADDR_SGPIOA18	ADDR_SGPIOA19	ADDR_SGPIOA9	ADDR_SGPIOA10									
W	ADDR_SGPIOA20	ADDR_SGPIOA21	ADDR_SGPIOA11	ADDR_SGPIOA12									
Y	ADDR_SGPIOA22	ADDR_SGPIOA23	ADDR_SGPIOA13	ADDR_SGPIOA14									
AA	ADDR_SGPIOA24	ADDR_SGPIOA25	ADDR_SGPIOA15	ADDR_SGPIOA30									
AB	ADDR_SGPIOA26	ADDR_SGPIOA27	ADDR_SGPIOA31	QVDDL									
AC	ADDR_SGPIOA28	NC	QVDDL	VSS	VDD	VDDH	DATA_SGPIOD29	DATA_SGPIOD27	NVDDL	DATA_SGPIOD24	DATA_SGPIOD22	DATA_SGPIOD20	NVDDL
AD	ADDR_SGPIOA29	QVDDL	VSS	VDD	NC	DATA_SGPIOD31	DATA_SGPIOD30	DATA_SGPIOD28	DATA_SGPIOD26	DATA_SGPIOD25	DATA_SGPIOD23	DATA_SGPIOD21	DATA_SGPIOD19
AE	QVDDL	VSS	VDD	DATA_SGPIOD1	DATA_SGPIOD3	DATA_SGPIOD5	DATA_SGPIOD7	DATA_SGPIOD9	DATA_SGPIOD11	DATA_SGPIOD13	DATA_SGPIOD15	DATA_SGPIOD17	IRQ3_B_KR_B_RETRY_B_SGPIOC3
AF	VSS	VDD	DATA_SGPIOD0	DATA_SGPIOD2	DATA_SGPIOD4	DATA_SGPIOD6	DATA_SGPIOD8	DATA_SGPIOD10	DATA_SGPIOD12	DATA_SGPIOD14	DATA_SGPIOD16	DATA_SGPIOD18	IRQ1_B_RSV_B_SGPIOC1
	1	2	3	4	5	6	7	8	9	10	11	12	13

NOTE: This pinout is a top down view of the package.

Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 2)

14	15	16	17	18	19	20	21	22	23	24	25	26	
AN65_B_PQB1	QVDDL	A_TPUCH2	A_TPUCH4	A_TPUCH6	A_TPUCH10	A_TPUCH14	A_TPUCH1	B_TPUCH5	B_TPUCH10	B_TPUCH12	B_TPUCH14	VSS	A
AN66_B_PQB2	QVDDL	ETRIG2	A_TPUCH5	A_TPUCH8	A_TPUCH11	A_T2CLK	B_TPUCH3	B_TPUCH6	B_TPUCH7	B_TPUCH13	VSS	VDD	B
AN70_B_PQB6	QVDDL	ETRIG1	B_CNRX0	A_TPUCH9	A_TPUCH12	A_TPUCH15	B_TPUCH4	B_TPUCH11	B_TPUCH8	VSS	VDD	B_TPUCH15	C
AN68_B_PQB4	QVDDL	VDDH	A_TPUCH3	A_TPUCH7	A_TPUCH13	A_TPUCH0	B_TPUCH9	NVDDL	VSS	VDD	B_TPUCH2	B_TPUCH0	D
									VDD	B_TPUCH1	B_T2CLK	MPWM17	E
									MPWM5_MPIO32B6	MPWM18	MDA11	MDA13	F
									MDA12	MDA27	MDA28	MDA29	G
									MDA30	MDA31	MPWM0	MPWM1	H
									MPWM3	MPWM2	MPWM16	MPWM20_MPIO32B11	J
									MDA15	MDA14	MPWM21_MPIO32B12	C_CNTX0_MPIO32B13	K
VSS	VSS	VSS							C_CNRX0_MPIO32B14	MPIO32B15	MPWM19	VF0_MPIO32B0	L
VSS	VSS	VSS							VF1_MPIO32B1	VF2_MPIO32B2	MPWM4_MPIO32B5	VFLS0_MPIO32B3	M
VSS	VSS	VSS							VDDH	VFLS1_MPIO32B4	B_PCS0_SS_B_QGPIO0	B_PCS1_QGPIO1	N

NOTE: This pinout is a top down view of the package.

Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 3)

VSS	VSS	VSS							B_ECK	B_MISO_QGPIO4	B_PCS3_J1850_TX	B_MOSI_QGPIO5	P
VSS	VSS	VSS							B_SCK_QGPIO6	B_PCS2_QGPIO2	B_TXD1_QGPO1	B_TXD2_QGPO2	R
VSS	VSS	VSS							A_TXD1_QGPO1	A_MISO_QGPIO4	B_RXD2_J1850_RX	A_SCK_QGPIO6	T
									A_PCS2_QGPIO2	A_RXD1_QPI1	A_MOSI_QGPIO5	A_PCS3_QGPIO3	U
									A_PCS0_SS_B_QGPIO0	A_TXD2_QGPO2	A_RXD2_QPI2	B_RXD1_QGPI1	V
									NVDDL	VFLASH	A_PCS1_QGPIO1	PULLSEL	W
									VDDF	EXTCLK	A_CNTXO	KAPWR	Y
									PORESET_B_TRST_B	A_CNRXO	VSSF	XTAL	AA
									HRESET_B	IRQ6_B_MODCK2	RSTCONF_B_TEXP	EXTAL	AB
SGPIOC7_IRQOUT_B_LWP0	NVDDL	WE_B_AT1	NVDDL	CS3_B	BI_B_STS_B	VDDH	VDD	VSS	QVDDL	SRESET_B	IRQ7_B_MODCK3	VSSSYN	AC
IRQ4_B_AT2_SGPIOC4	TEA_B	IRQ2_B_CR_B_SGPIOC2	WE_B_AT2	CS1_B	TSIZ0	B0EPEE	CLKOUT	VDD	VSS	QVDDL	IRQ5_B_SGPIOC5_MODCK1	XFC	AD
BB_B_VF2_IWP3	RD_WR_B	OE_B	WE_B_AT0	CS0_B	BURST_B	TS_B	BDIP_B	NC	VDD	VSS	QVDDL	VDDSYN	AE
BG_B_VF0_LWP1	BR_B_VF1_IWP2	IRQ0_B_SGPIOC0	WE_B_AT3	CS2_B	TSIZ1	TA_B	EPEE	ENGCLK_BUCLK	NC	VDD	VSS	QVDDL	AF
14	15	16	17	18	19	20	21	22	23	24	25	26	

NOTE: This pinout is a top down view of the package.

Figure 2-9 MPC565 / MPC566 Ball Map (Black and White, page 4)

SECTION 3
CENTRAL PROCESSING UNIT

The PowerPC™-based RISC processor (RCPU) used in the MPC500 family of microcontrollers integrates five independent execution units: an integer unit (IU), a load/store unit (LSU), and a branch processing unit (BPU), floating-point unit (FPU) and integer multiplier divider (IMD). The use of simple instructions with rapid execution times yields high efficiency and throughput for MPC500-based systems.

Most integer instructions execute in one clock cycle. Instructions can complete out of order for increased performance; however, the processor makes execution appear sequential.

This section provides an overview of the RCPU. For a detailed description of this processor, refer to the ***RCPU Reference Manual* (RCPURM/AD)**.

3.1 RCPU Features

Major features of the RCPU include the following:

- High-performance microprocessor
  - — Single clock-cycle execution for many instructions
- Five independent execution units and two register files
  - — Independent LSU for load and store operations
  - — BPU featuring static branch prediction
  - — A 32-bit IU
  - — Fully IEEE 754-compliant FPU for both single- and double-precision operations
  - — Thirty-two general-purpose registers (GPRs) for integer operands
  - — Thirty-two floating-point registers (FPRs) for single- or double-precision operands
- Facilities for enhanced system performance
  - — Programmable big- and little-endian byte ordering
  - — Atomic memory references
- In-system testability and debugging features
- High instruction and data throughput
  - — Condition register (CR) look-ahead operations performed by BPU
  - — Branch-folding capability during execution (zero-cycle branch execution time)
  - — Programmable static branch prediction on unresolved conditional branches
  - — A pre-fetch queue that can hold up to four instructions, providing look-ahead capability
  - — Interlocked pipelines with feed-forwarding that control data dependencies in hardware

3.2 RCPU Block Diagram

Figure 3-1 provides a block diagram of the RCPU.

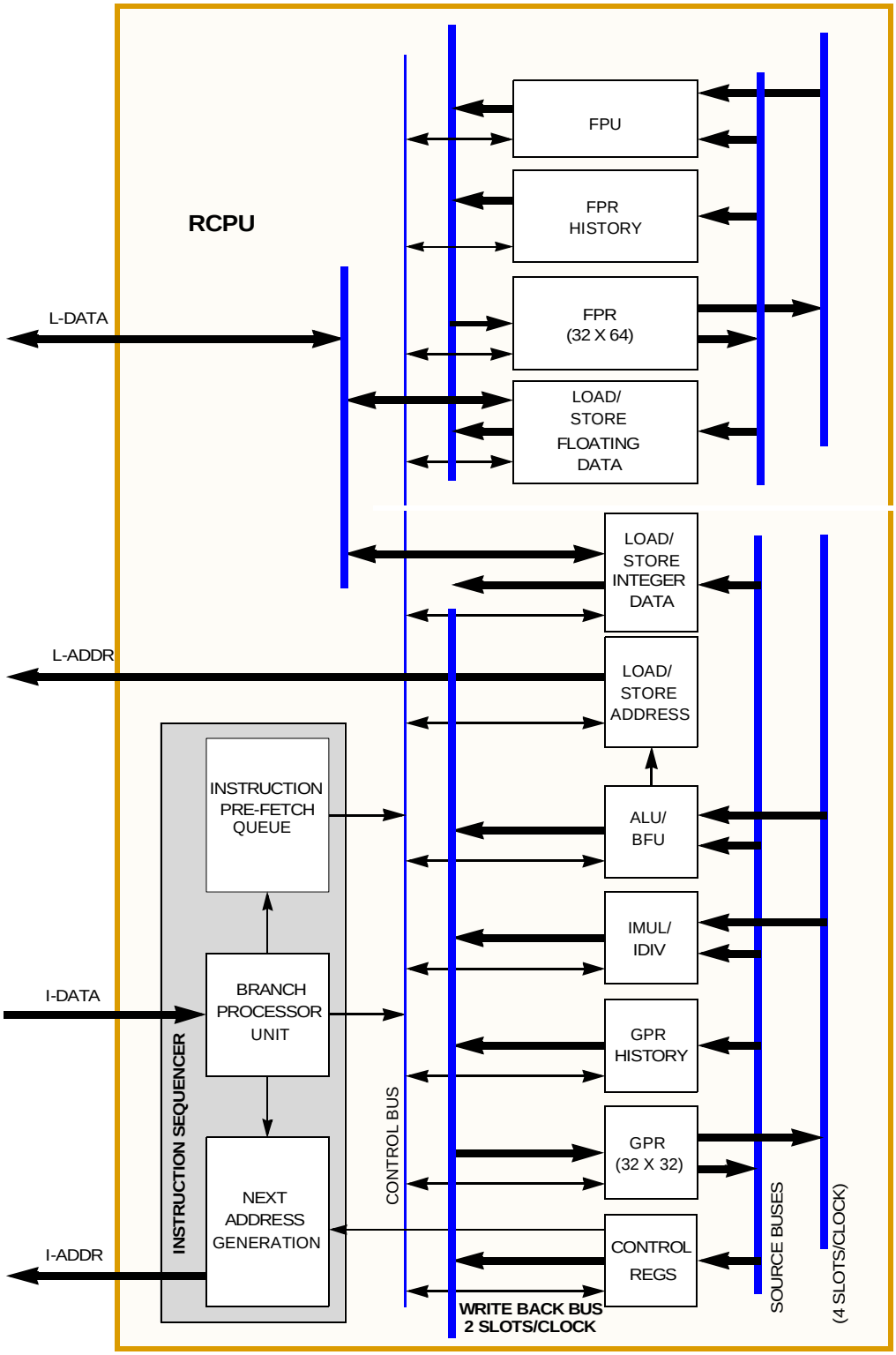


Figure 3-1 RCPU Block Diagram

3.3 Instruction Sequencer

The instruction sequencer provides centralized control over data flow between execution units and register files. The sequencer implements the basic instruction pipeline, fetches instructions from the memory system, issues them to available execution units, and maintains a state history so it can back the machine up in the event of an exception.

The instruction sequencer fetches instructions from the burst buffer controller into the instruction pre-fetch queue. The BPU extracts branch instructions from the pre-fetch queue and uses static branch prediction on unresolved conditional branches to allow the instruction unit to fetch instructions from a predicted target instruction stream while a conditional branch is evaluated. The BPU folds out branch instructions for unconditional branches or conditional branches unaffected by instructions in the execution stage.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

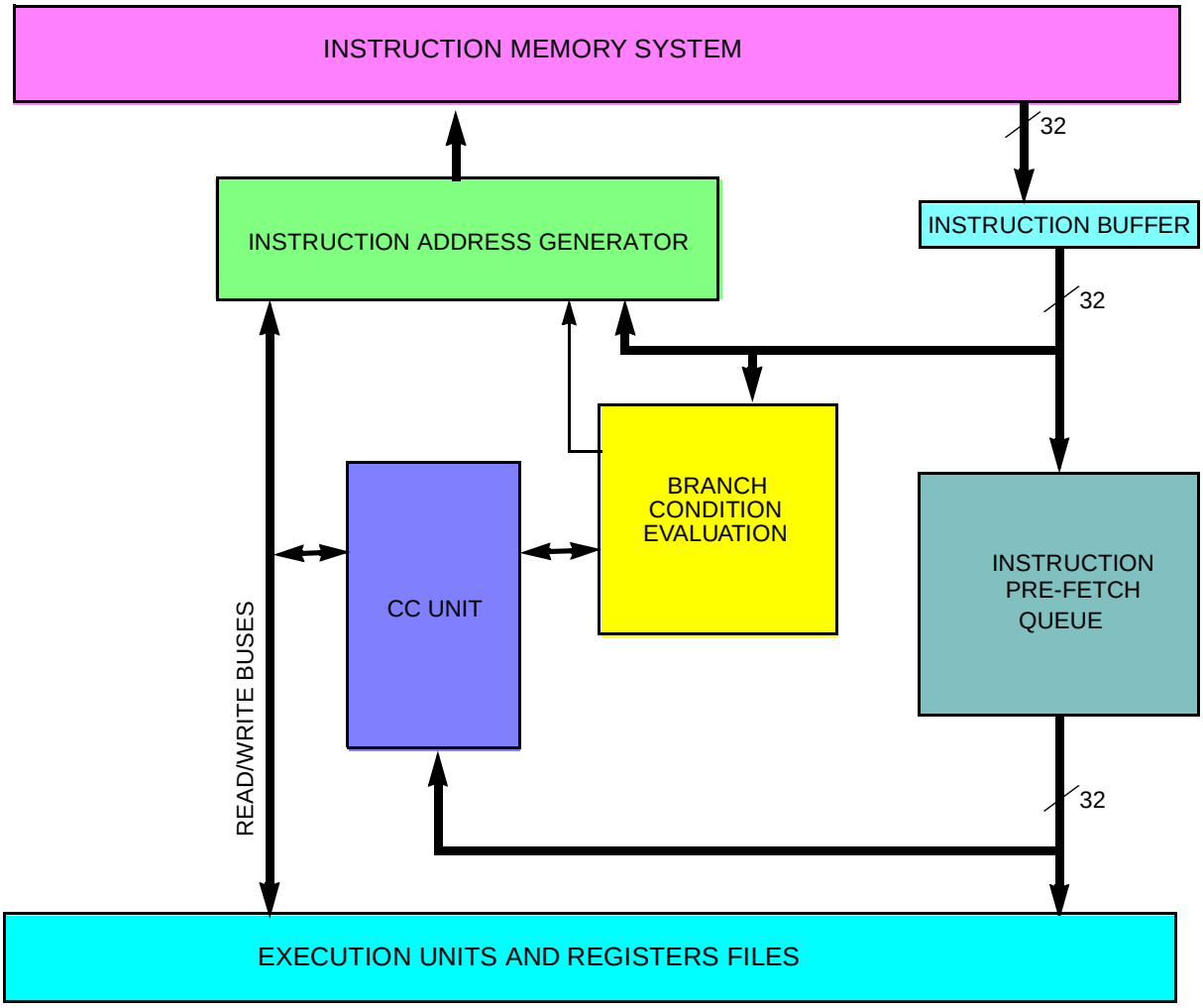


Figure 3-2 Sequencer Data Path

3.4 Independent Execution Units

The PowerPC architecture supports independent floating-point, integer, load/store, and branch processing execution units, making it possible to implement advanced features such as look-ahead operations. For example, since branch instructions do not depend on GPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

Table 3-1 summarizes the RCPU execution units.

Table 3-1 RCPU Execution Units

Unit	Description
Branch processing unit (BPU)	Includes the implementation of all branch instructions
Load/store unit (LSU)	Includes implementation of all load and store instructions, whether defined as part of the integer processor or the floating-point processor
Integer unit (IU)	Includes implementation of all integer instructions except load/store instructions. This module includes the GPRs (including GPR history and scoreboard) and the following subunits: The IMUL-IDIV includes the implementation of the integer multiply and divide instructions. The ALU-BFU includes implementation of all integer logic, add and subtract instructions, and bit field instructions.
Floating-point unit (FPU)	Includes the FPRs (including FPR history and scoreboard) and the implementation of all floating-point instructions except load and store floating-point instructions

The following sections describe the execution units in greater detail.

3.4.1 Branch Processing Unit (BPU)

The BPU, located within the instruction sequencer, performs condition register look-ahead operations on conditional branches. The BPU looks through the instruction queue for a conditional branch instruction and attempts to resolve it early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the processor pre-fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an calculation feature to compute branch target addresses and three special-purpose, user-accessible registers: the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it into the LR. The LR also contains the branch target address for the branch conditional to link register (**bclr*x***) instruction. The CTR contains the branch target address for the branch conditional to count register (**bcctr*x***) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than general-purpose or floating-point registers, execution of branch instructions is independent from execution of integer instructions.

3.4.2 Integer Unit (IU)

The IU executes all integer processor instructions, except the integer storage access instructions, which are implemented by the load/store unit. The IU contains the following subunits:

- The IMUL–IDIV unit includes the implementation of the integer multiply and divide instructions.
- The ALU–BFU unit includes the implementation of all integer logic, add and subtract, and bit field instructions.

The IU also includes the integer exception register (XER) and the general-purpose register file.

IMUL–IDIV and ALU–BFU are implemented as separate execution units. The ALU–BFU unit can execute one instruction per clock cycle. IMUL–IDIV instructions require multiple clock cycles to execute. IMUL–IDIV is pipelined for multiply instructions, so that consecutive multiply instructions can be issued on consecutive clock cycles. Divide instructions are not pipelined; an integer divide instruction preceded or followed by an integer divide or multiply instruction results in a stall in the processor pipeline. Note that since IMUL–IDIV and ALU–BFU are implemented as separate execution units, an integer divide instruction preceded or followed by an ALU–BFU instruction does not cause a delay in the pipeline.

3.4.3 Load/Store Unit (LSU)

The load/store unit handles all data transfer between the general-purpose register file and the internal load/store bus (L-bus). The load/store unit is implemented as an independent execution unit so that stalls in the memory pipeline do not cause the master instruction pipeline to stall (unless there is a data dependency). The unit is fully pipelined so that memory instructions of any size may be issued on back-to-back cycles.

There is a 32-bit wide data path between the load/store unit and the general-purpose register file. Single-word accesses can be achieved with an internal on-chip data RAM, resulting in two clocks latency. Double-word accesses require two clocks, resulting in three clocks latency. Since the L-bus is 32 bits wide, double-word transfers require two bus accesses. The load/store unit performs zero-fill for byte and half-word transfers and sign extension for half-word transfers.

Addresses are formed by adding the source one register operand specified by the instruction (or zero) to either a source two register operand or to a 16-bit, immediate value embedded in the instruction.

3.4.4 Floating-Point Unit (FPU)

The FPU contains a double-precision multiply array, the floating-point status and control register (FPSCR), and the FPRs. The multiply-add array allows the MPC565 / MPC566 to efficiently implement floating-point operations such as multiply, multiply-add, and divide.

The MPC565 / MPC566 depends on a software envelope to fully implement the IEEE floating-point specification. Overflows, underflows, NaNs, and denormalized numbers cause floating-point assist exceptions that invoke a software routine to deliver (with hardware assistance) the correct IEEE result.

To accelerate time-critical operations and make them more deterministic, the MPC565 / MPC566 provides a mode of operation that avoids invoking the software envelope and attempts to deliver results in hardware that are adequate for most applications, if not in strict conformance with IEEE standards. In this mode, denormalized numbers, NaNs, and IEEE invalid operations are treated as legitimate, returning default results rather than causing floating-point assist exceptions.

3.5 Levels of the PowerPC Architecture

The PowerPC architecture consists of three layers. Adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture are implemented:

- PowerPC user instruction set architecture (UISA) — Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- PowerPC virtual environment architecture (VEA) — Describes the memory model for a multiprocessor environment, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- PowerPC operating environment architecture (OEA) — Defines the memory management model, supervisor-level registers, synchronization requirements, and the exception model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

3.6 RCPU Programming Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between memory and on-chip registers.

PowerPC processors have two levels of privilege: supervisor mode of operation (typically used by the operating environment) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, special-purpose registers (SPRs), and several miscellaneous registers.

Supervisor-level access is provided through the processor's exception mechanism. That is, when an exception is taken (either due to an error or problem that needs to be serviced, or deliberately through the use of a trap instruction), the processor begins operating in supervisor mode. The level of access is indicated by the privilege-level (PR) bit in the machine state register (MSR).

Figure 3-3 shows the user-level and supervisor-level RCPU programming models and also illustrates the three levels of the PowerPC architecture. The numbers to the left of the SPRs indicate the decimal number that is used in the syntax of the instruction operands to access the register.

Note that registers such as the general-purpose registers (GPRs) are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as move to special-purpose register (**mtspr**) and move from special-purpose register (**mfspr**)

instructions) or implicitly as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

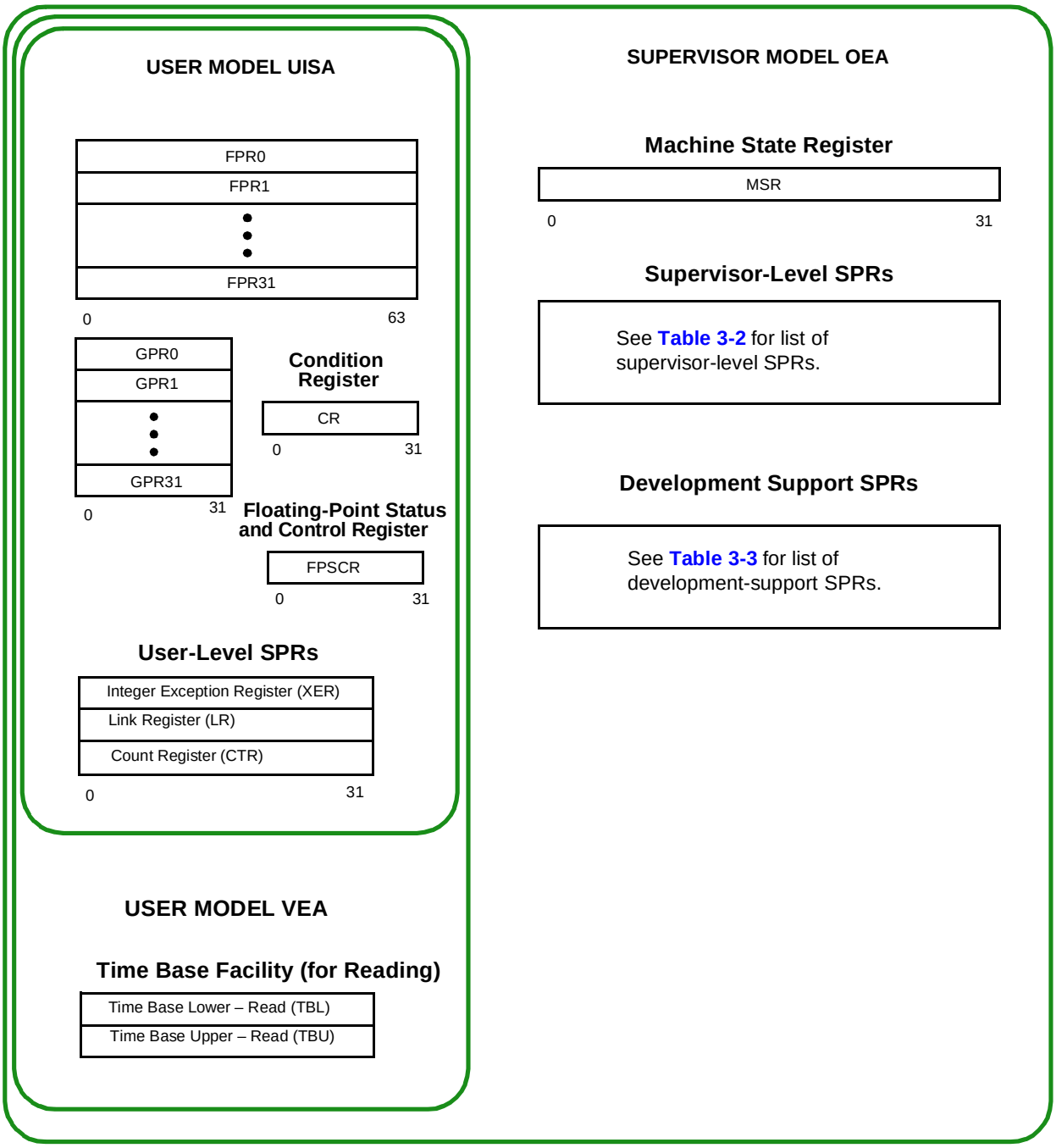


Figure 3-3 RCPU Programming Model

Table 3-2 lists the MPC565 / MPC566 supervisor-level registers.

Table 3-2 Supervisor-Level SPRs

SPR Number (Decimal)	Special-Purpose Register
18	DAE/Source Instruction Service Register (DSISR) See **3.9.2 DAE/Source Instruction Service Register (DSISR)** for bit descriptions.
19	Data Address Register (DAR) See **3.9.3 Data Address Register (DAR)** for bit descriptions.
22	Decrementer Register (DEC) See **3.9.5 Decrementer Register (DEC)** for bit descriptions.
26	Save and Restore Register 0 (SRR0) See **3.9.6 Machine Status Save/Restore Register 0 (SRR0)** for bit descriptions.
27	Save and Restore Register 1 (SRR1) See **3.9.7 Machine Status Save/Restore Register 1 (SRR1)** for bit descriptions.
80	External Interrupt Enable (EIE)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
81	External Interrupt Disable (EID)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
82	Non-Recoverable Interrupt (NRI)[1] See **3.9.10.1 EIE, EID, and NRI Special-Purpose Registers** for bit descriptions.
272	SPR General 0 (SPRG0)[1] See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
273	SPRGeneral 1 (SPRG1)[1] See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
274	SPR General 2 (SPRG2) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
275	SPR General 3 (SPRG3) See **3.9.8 General SPRs (SPRG0–SPRG3)** for bit descriptions.
284	Time Base Lower – Write (TBL) See **Table 3-14** for bit descriptions.
285	Time Base Upper – Write (TBU) See **Table 3-14** for bit descriptions.
287	Processor Version Register (PVR) See **Table 3-16** for bit descriptions.
528	IMPU Global Region Attribute (MI_GRA)1 See **Table 4-8** for bit descriptions.
536	L2U Region Attribute (L2U_GRA) See **Table 11-10** for bit descriptions.
560	BBC Module Configuration Register (BBC_MCR)[1] See **Table 4-4** for bit descriptions.

Table 3-2 Supervisor-Level SPRs (Continued)

SPR Number (Decimal)	Special-Purpose Register
568	L2U Module Configuration Register (L2U_MCR)[1] See **Table 11-7** for bit descriptions.
784	L2U Region Base Address Register 0 (L2U_RBA0)[1] See **Table 4-5** for bit descriptions.
785	IMPU Region Base Address Register 1 (MI_RBA1)[1]. See **Table 4-5** for bits descriptions.
786	IMPU Region Base Address Register 2 (MI_RBA2)[1]. See **Table 4-5** for bits descriptions.
787	IMPU Region Base Address Register 3 (MI_RBA3[1]). See **Table 4-5** for bits descriptions.
816	IMPU Region Attribute Register 0 (MI_RA0)[1]. See **Table 4-6** for bits descriptions.
817	IMPU Region Attribute Register 1 (MI_RA1)[1]. See **Table 4-6** for bits descriptions.
818	IMPU Region Attribute Register 2 (MI_RA2)[1]. See **Table 4-6** for bits descriptions.
819	IMPU Region Attribute Register 3(MI_RA3)[1]. See **Table 4-6** for bits descriptions.
792	L2U Region Base Address Register 0 (L2U_RBA0)[1] See **Table 11-8** for bit descriptions.
793	L2U Region Base Address Register 1 (L2U_RBA1)[1] See **Table 11-8** for bit descriptions.
794	L2U Region Base Address Register 2 (L2U_RBA2)[1] See **Table 11-8** for bit descriptions.
795	L2U Region Base Address Register 3 (L2U_RBA3)[1] See **Table 11-8** for bit descriptions.
824	L2U Region Attribute Register 0 (L2U_RA0)[1] See **Table 11-9** for bit descriptions.
825	L2U Region Attribute Register 1 (L2U_RA1)[1] See **Table 11-9** for bit descriptions.
826	L2U Region Attribute Register 2 (L2U_RA2)[1] See **Table 11-9** for bit descriptions.
827	L2U Region Attribute Register 3 (L2U_RA3)[1] See **Table 11-9** for bit descriptions.
1022	Floating-Point Exception Cause Register (FPECR)[1] See **3.9.10.2 Floating-Point Exception Cause Register (FPECR)** for bit descriptions.

NOTES:
1. Implementation-specific SPR.

Table 3-3 lists the MPC565 / MPC566 SPRs used for development support.

Table 3-3 Development Support SPRs[1]

SPR Number (Decimal)	Special-Purpose Register
144	Comparator A Value Register (CMPA) See **Table 22-17** for bit descriptions.
145	Comparator B Value Register (CMPB) See **Table 22-17** for bit descriptions.
146	Comparator C Value Register (CMPC) See **Table 22-17** for bit descriptions.
147	Comparator D Value Register (CMPD) See **Table 22-17** for bit descriptions.
148	Exception Cause Register (ECR) See **Table 22-27** for bit descriptions.
149	Debug Enable Register (DER) See **Table 22-28** for bit descriptions.
150	Breakpoint Counter A Value and Control (COUNTA) See **Table 22-25** for bit descriptions.
151	Breakpoint Counter B Value and Control (COUNTB) See **Table 22-26** for bit descriptions.
152	Comparator E Value Register (CMPE) See **Table 22-18** for bit descriptions.
153	Comparator F Value Register (CMPF) See **Table 22-18** for bit descriptions.
154	Comparator G Value Register (CMPG) See **Table 22-20** for bit descriptions.
155	Comparator H Value Register (CMPH) See **Table 22-20** for bit descriptions.
156	L-bus Support Comparators Control 1 (LCTRL1) See **Table 22-23** for bit descriptions.
157	L-bus Support Comparators Control 2 (LCTRL2) See **Table 22-24** for bit descriptions.
158	I-bus Support Control Register (ICTRL) See **Table 22-21** for bit descriptions.
159	Breakpoint Address Register (BAR) See **Table 22-19** for bit descriptions.
630	Development Port Data Register (DPDR) See **22.7.13** for bit descriptions.

NOTES:
1. All development-support SPRs are implementation-specific.

Where not otherwise noted, reserved fields in registers are ignored when written and return zero when read. An exception to this rule is XER[16:23]. These bits are set to the value written to them and return that value when read.

3.7 PowerPC UISA Register Set

The PowerPC UISA registers can be accessed by either user- or supervisor-level instructions. The general-purpose registers are accessed through instruction operands.

3.7.1 General-Purpose Registers (GPRs)

Integer data is manipulated in the integer unit's thirty-two 32-bit GPRs, shown below. These registers are accessed as source and destination registers through operands in the instruction syntax.

GPRs — General-Purpose Registers

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
GPR0																															
GPR1																															
. . .																															
. . .																															
GPR31																															

RESET: UNCHANGED

3.7.2 Floating-Point Registers (FPRs)

The PowerPC architecture provides thirty-two 64-bit FPRs. These registers are accessed as source and destination registers through operands in floating-point instructions. Each FPR supports the double-precision, floating-point format. Every instruction that interprets the contents of an FPR as a floating-point value uses the double-precision floating-point format for this interpretation. That is, all floating-point numbers are stored in double-precision format.

All floating-point arithmetic instructions operate on data located in FPRs and, with the exception of the compare instructions (which update the CR), place the result into an FPR. Information about the status of floating-point operations is placed into the floating-point status and control register (FPSCR) and in some cases, into the CR, after the completion of the operation's writeback stage. For information on how the CR is affected by floating-point operations, see **3.7.4 Condition Register (CR)**.

FPRs— Floating-Point Registers

MSB 0	LSB 63
FPR0	
FPR1	
. . .	
. . .	
FPR31	

RESET: UNCHANGED

3.7.3 Floating-Point Status and Control Register (FPSCR)

The FPSCR controls the handling of floating-point exceptions and records status resulting from the floating-point operations. FPSCR[0:23] are status bits. FPSCR[24:31] are control bits.

FPSCR[0:12] and FPSCR[21:23] are floating-point exception condition bits. These bits are sticky, except for the floating-point enabled exception summary (FEX) and floating-point invalid operation exception summary (VX). Once set, sticky bits remain set until they are cleared by an **mcrfs**, **mtfsfi**, **mtfsf**, or **mtfsb0** instruction.

Table 3-4 summarizes which bits in the FPSCR are sticky status bits, which are normal status bits, and which are control bits.

Table 3-4 FPSCR Bit Categories

Bits	Type
[0], [3:12], [21:23]	Status, sticky
[1:2], [13:20]	Status, not sticky
[24:31]	Control

FEX and VX are the logical ORs of other FPSCR bits. Therefore these two bits are not listed among the FPSCR bits directly affected by the various instructions.

FPSCR — Floating-Point Status and Control Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FX	FEX	VX	OX	UX	ZX	XX	VXS-NAN	VXISI	VXIDI	VXZDZ	VXIMZ	VXVC	FR	FI	FPRF 0

RESET: UNCHANGED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
FPRF[1:4]				0	VX-SOFT	VX-SQRT	VXCVI	VE	OE	UE	ZE	XE	NI	RN	

RESET: UNCHANGED

A listing of FPSCR bit settings is shown in **Table 3-5**.

Table 3-5 FPSCR Bit Descriptions

Bit(s)	Name	Description
0	FX	Floating-point exception summary. Every floating-point instruction implicitly sets FPSCR[FX] if that instruction causes any of the floating-point exception bits in the FPSCR to change from 0 to 1. The **mcrfs** instruction implicitly clears FPSCR[FX] if the FPSCR field containing FPSCR[FX] is copied. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions can set or clear FPSCR[FX] explicitly. This is a sticky bit.
1	FEX	Floating-point enabled exception summary. This bit signals the occurrence of any of the enabled exception conditions. It is the logical OR of all the floating-point exception bits masked with their respective enable bits. The **mcrfs** instruction implicitly clears FPSCR[FEX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[FEX] explicitly. This is not a sticky bit.
2	VX	Floating-point invalid operation exception summary. This bit signals the occurrence of any invalid operation exception. It is the logical OR of all of the invalid operation exceptions. The **mcrfs** instruction implicitly clears FPSCR[VX] if the result of the logical OR described above becomes zero. The **mtfsf**, **mtfsfi**, **mtfsb0**, and **mtfsb1** instructions cannot set or clear FPSCR[VX] explicitly. This is not a sticky bit.
3	OX	Floating-point overflow exception. This is a sticky bit.
4	UX	Floating-point underflow exception. This is a sticky bit.
5	ZX	Floating-point zero divide exception. This is a sticky bit.
6	XX	Floating-point inexact exception. This is a sticky bit.
7	VXSNAN	Floating-point invalid operation exception for SNaN. This is a sticky bit.
8	VXISI	Floating-point invalid operation exception for ×-×. This is a sticky bit.
9	VXIDI	Floating-point invalid operation exception for ×/×. This is a sticky bit.
10	VXZDZ	Floating-point invalid operation exception for 0/0. This is a sticky bit.
11	VXIMZ	Floating-point invalid operation exception for ×*0. This is a sticky bit.
12	VXVC	Floating-point invalid operation exception for invalid compare. This is a sticky bit.
13	FR	Floating-point fraction rounded. The last floating-point instruction that potentially rounded the intermediate result incremented the fraction. This bit is not sticky.
14	FI	Floating-point fraction inexact. The last floating-point instruction that potentially rounded the intermediate result produced an inexact fraction or a disabled exponent overflow. This bit is not sticky.
[15:19]	FPRF	Floating-point result flags. This field is based on the value placed into the target register even if that value is undefined. Refer to **Table 3-6** for specific bit settings. 15 Floating-point result class descriptor (C). Floating-point instructions other than the compare instructions may set this bit with the FPCC bits, to indicate the class of the result. 16–19 Floating-point condition code (FPCC). Floating-point compare instructions always set one of the FPCC bits to one and the other three FPCC bits to zero. Other floating-point instructions may set the FPCC bits with the C bit, to indicate the class of the result. Note that in this case the high-order three bits of the FPCC retain their relational significance indicating that the value is less than, greater than, or equal to zero. 16 Floating-point less than or negative (FL or <) 17 Floating-point greater than or positive (FG or >) 18 Floating-point equal or zero (FE or =) 19 Floating-point unordered or NaN (FU or ?)
20	—	Reserved

Table 3-5 FPSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
21	VXSOFT	Floating-point invalid operation exception for software request. This bit can be altered only by the **mcrfs**, **mtfsfi**, **mtfsf**, **mtfsb0**, or **mtfsb1** instructions. The purpose of VXSOFT is to allow software to cause an invalid operation condition for a condition that is not necessarily associated with the execution of a floating-point instruction. For example, it might be set by a program that computes a square root if the source operand is negative. This is a sticky bit.
22	VXSQRT	Floating-point invalid operation exception for invalid square root. This is a sticky bit. This guarantees that software can simulate **fsqrt** and **frsqrte**, and to provide a consistent interface to handle exceptions caused by square-root operations.
23	VXCVI	Floating-point invalid operation exception for invalid integer convert. This is a sticky bit.
24	VE	Floating-point invalid operation exception enable.
25	OE	Floating-point overflow exception enable.
26	UE	Floating-point underflow exception enable. This bit should not be used to determine whether denormalization should be performed on floating-point stores.
27	ZE	Floating-point zero divide exception enable.
28	XE	Floating-point inexact exception enable.
29	NI	Non-IEEE mode bit.
30–31	RN	Floating-point rounding control. 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward -infinity

Table 3-6 illustrates the floating-point result flags that correspond to FPSCR[15:19].

Table 3-6 Floating-Point Result Flags in FPSCR

Result Flags (Bits 15–19) C<>=?	Result value class
10001	Quiet NaN
01001	– Infinity
01000	– Normalized number
11000	– Denormalized number
10010	– Zero
00010	+ Zero
10100	+ Denormalized number
00100	+ Normalized number
00101	+ Infinity

3.7.4 Condition Register (CR)

The condition register (CR) is a 32-bit register that reflects the result of certain operations and provides a mechanism for testing and branching. The bits in the CR are grouped into eight 4-bit fields, CR0 to CR7.

CR — Condition Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
CR0				CR1				CR2				CR3				CR4				CR5				CR6				CR7			

RESET: UNCHANGED

The CR fields can be set in the following ways:

- Specified fields of the CR can be set by a move instruction (**mtcrf**) to the CR from a GPR.
- Specified fields of the CR can be moved from one CR *x* field to another with the **mcrf** instruction**.**
- A specified field of the CR can be set by a move instruction (**mcrxr**) to the CR from the XER.
- Condition register logical instructions can be used to perform logical operations on specified bits in the condition register.
- CR0 can be the implicit result of an integer operation.
- A specified CR field can be the explicit result of an integer compare instruction.

Instructions are provided to test individual CR bits.

3.7.4.1 Condition Register CR0 Field Definition

In most integer instructions, when the CR is set to reflect the result of the operation (that is, when Rc = 1), and for **addic.**, **andi.**, and **andis.**, the first three bits of CR0 are set by an algebraic comparison of the result to zero; the fourth bit of CR0 is copied from XER[SO]. For integer instructions, CR[0:3] are set to reflect the result as a signed quantity. The result as an unsigned quantity or a bit string can be deduced from the EQ bit.

The CR0 bits are interpreted as shown in **Table 3-7**. If any portion of the result (the 32-bit value placed into the destination register) is undefined, the value placed in the first three bits of CR0 is undefined.

Table 3-7 Bit Settings for CR0 Field of CR

CR0 Bit	Description
0	Negative (LT) — This bit is set when the result is negative.
1	Positive (GT) — This bit is set when the result is positive (and not zero).
2	Zero (EQ) — This bit is set when the result is zero.
3	Summary overflow (SO) — This is a copy of the final state of XER[SO] at the completion of the instruction.

3.7.4.2 Condition Register CR1 Field Definition

In all floating-point instructions when the CR is set to reflect the result of the operation (that is, when Rc = 1), the CR1 field (bits 4 to 7 of the CR) is copied from FPSCR[0:3] to indicate the floating-point exception status. For more information about the FPSCR,

see **3.7.3 Floating-Point Status and Control Register (FPSCR)**. The bit settings for the CR1 field are shown in **Table 3-8**.

Table 3-8 Bit Settings for CR1 Field of CR

CR1 Bit	Description
0	Floating-point exception (FX) — This is a copy of the final state of FPSCR[FX] at the completion of the instruction.
1	Floating-point enabled exception (FEX) — This is a copy of the final state of FPSCR[FEX] at the completion of the instruction.
2	Floating-point invalid exception (VX) — This is a copy of the final state of FPSCR[VX] at the completion of the instruction.
3	Floating-point overflow exception (OX) — This is a copy of the final state of FPSCR[OX] at the completion of the instruction.

3.7.4.3 Condition Register CR*n* Field — Compare Instruction

When a specified CR field is set by a compare instruction, the bits of the specified field are interpreted as shown in **Table 3-9**. A condition register field can also be accessed by the **mfcr**, **mcrf**, and **mtcrf** instructions.

Table 3-9 CR*n* Field Bit Settings for Compare Instructions

CR*n* Bit[1]	Description
0	Less than, floating-point less than (LT, FL). For integer compare instructions, (**r**A) < SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) < (**fr**B).
1	Greater than, floating-point greater than (GT, FG). For integer compare instructions, (**r**A) > SIMM, UIMM, or (**r**B) (algebraic comparison) or (**r**A) SIMM, UIMM, or (**r**B) (logical comparison). For floating-point compare instructions, (**fr**A) > (**fr**B).
2	Equal, floating-point equal (EQ, FE). For integer compare instructions, (**r**A) = SIMM, UIMM, or (**r**B). For floating-point compare instructions, (**fr**A) = (**fr**B).
3	Summary overflow, floating-point unordered (SO, FU). For integer compare instructions, this is a copy of the final state of XER[SO] at the completion of the instruction. For floating-point compare instructions, one or both of (**fr**A) and (**fr**B) is not a number (NaN).

NOTES:
1. Here, the bit indicates the bit number in any one of the four-bit subfields, CR0–CR7

3.7.5 Integer Exception Register (XER)

The integer exception register (XER) is a user-level, 32-bit register.

XER — Integer Exception Register **SPR 1**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SO		OV		CA		Reserved																			BYTES						
RESET:																															
U	U	U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U

The bit descriptions for XER, shown in **Table 3-10**, are based on the operation of an instruction considered as a whole, not on intermediate results. For example, the result of the subtract from carrying (**subfc***x*) instruction is specified as the sum of three values. This instruction sets bits in the XER based on the entire operation, not on an intermediate sum.

In most cases, reserved fields in registers are ignored when written to and return zero when read. However, XER[16:23] are set to the value written to them and return that value when read.

Table 3-10 Integer Exception Register Bit Descriptions

Bit(s)	Name	Description
0	SO	Summary Overflow (SO) — The summary overflow bit is set whenever an instruction sets the overflow bit (OV) to indicate overflow and remains set until software clears it. It is not altered by compare instructions or other instructions that cannot overflow.
1	OV	Overflow (OV) — The overflow bit is set to indicate that an overflow has occurred during execution of an instruction. Integer and subtract instructions having OE=1 set OV if the carry out of bit 0 is not equal to the carry out of bit 1, and clear it otherwise. The OV bit is not altered by compare instructions or other instructions that cannot overflow.
2	CA	Carry (CA) — In general, the carry bit is set to indicate that a carry out of bit 0 occurred during execution of an instruction. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA to one if there is a carry out of bit 0, and clear it otherwise. The CA bit is not altered by compare instructions or other instructions that cannot carry, except that shift right algebraic instructions set the CA bit to indicate whether any ‘1’ bits have been shifted out of a negative quantity.
3:24	—	Reserved
25:31	BYTES	This field specifies the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

3.7.6 Link Register (LR)

The 32-bit link register supplies the branch target address for the branch conditional to link register (**bclr***x*) instruction, and can be used to hold the logical address of the instruction that follows a branch and link instruction.

Note that although the two least-significant bits can accept any values written to them, they are ignored when the LR is used as an address.

Both conditional and unconditional branch instructions include the option of placing the effective address of the instruction following the branch instruction in the LR. This is done regardless of whether the branch is taken.

LR — Link Register **SPR 8**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Branch Address																															

RESET: UNCHANGED

3.7.7 Count Register (CTR)

The count register (CTR) is a 32-bit register for holding a loop count that can be decremented during execution of branch instructions that contain an appropriately coded BO field. If the value in CTR is 0 before being decremented, it is –1 afterward. The count register provides the branch target address for the Branch Conditional to Count Register (**bcctr*x***) instruction.

CTR — Count Register **SPR 9**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Loop Count																															

RESET: UNCHANGED

3.8 PowerPC VEA Register Set — Time Base

The PowerPC virtual environment architecture (VEA) defines registers in addition to those in the UISA register set. The PowerPC VEA register set can be accessed by all software with either user- or supervisor-level privileges.

The PowerPC VEA includes the time base facility (TB), a 64-bit structure that contains a 64-bit unsigned integer that is incremented periodically. The frequency at which the counter is updated is implementation-dependent. For details on the time base clock in the MPC565 / MPC566, refer to **6.8 MPC565 / MPC566 Time Base (TB)**, **8.6 MPC565 / MPC566 Internal Clock Signals**, and **8.12.1 System Clock Control Register (SCCR)**.

The TB consists of two 32-bit registers: time base upper (TBU) and time base lower (TBL). In the context of the VEA, user-level applications are permitted read-only access to the TB. The OEA defines supervisor-level access to the TB for writing values to the TB. Different SPR encodings are provided for reading and writing the time base.

TB — Time Base (Read Only) **SPR 268, 269**

0	31	32	63
TBU		TBL	

RESET: UNCHANGED

Table 3-11 Time Base Field Definitions (Read Only)

Bits	Name	Description
0-31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32-63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

In 32-bit PowerPC implementations such as the RCPU, it is not possible to read the entire 64-bit time base in a single instruction. The **mftb** simplified mnemonic copies the lower half of the time base register (TBL) to a GPR, and the **mftbu** simplified mnemonic copies the upper half of the time base (TBU) to a GPR.

3.9 PowerPC OEA Register Set

The PowerPC operating environment architecture (OEA) includes a number of SPRs and other registers that are accessible only by supervisor-level instructions. Some SPRs are RCPU-specific; some RCPU SPRs may not be implemented in other PowerPC processors, or may not be implemented in the same way.

3.9.1 Machine State Register (MSR)

The machine state register is a 32-bit register that defines the state of the processor. When an exception occurs, the current contents of the MSR are loaded into SRR1, and the MSR is updated to reflect the exception-processing machine state. The MSR can also be modified by the **mtmsr**, **sc**, and **rfi** instructions. It can be read by the **mfmsr** instruction.

MSR — Machine State Register

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED													POW	0	ILE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
EE	PR	FP	ME	FE0	SE	BE	FE1	0	IP	IR	DR	RE-SERVED	DCMPEN[2]	RI	LE
RESET[1]:															
0	0	0	U	0	0	0	0	0	ID1*	0	0	0	0	0	0

NOTES:
1. Reset value of this bit depends on the value of the internal data bus line during reset.
2. This bit is available only on MPC566 options.

Table 3-12 shows the bit definitions for the MSR.

Table 3-12 Machine State Register Bit Descriptions

Bit(s)	Name	Description
0:12	—	Reserved
13	POW	Power management enable 0 = Power management disabled (normal operation mode) 1 = Power management enabled (reduced power mode)
14	—	Reserved
15	ILE	Exception little-endian mode. When an exception occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the exception. 0 = Processor runs in big-endian mode during exception processing. 1 = Processor runs in little-endian mode during exception processing.
16	EE	External interrupt enable. Interrupts should only be negated while the EE bit is disabled (0). Software should disable interrupts in the CPU core prior to masking or disabling any interrupt which might be currently pending at the CPU core. For external interrupts, it is recommended that the edge triggered interrupt scheme be used. 0 = The processor delays recognition of external interrupts and decrementer exception conditions. 1 = The processor is enabled to take an external interrupt or the decrementer exception.
17	PR	Privilege level 0 = The processor can execute both user- and supervisor-level instructions. 1 = The processor can only execute user-level instructions.
18	FP	Floating-point available 0 = The processor prevents dispatch of floating-point instructions, including floating-point loads, stores and moves. Floating-point enabled program exceptions can still occur and the FPRs can still be accessed. 1 = The processor can execute floating-point instructions, and can take floating-point enabled exception type program exceptions.
19	ME	Machine check enable 0 = Machine check exceptions are disabled. 1 = Machine check exceptions are enabled.
20	FE0	Floating-point exception mode 0 (See **Table 3-13**.)
21	SE	Single-step trace enable 0 = The processor executes instructions normally. 1 = The processor generates a single-step trace exception upon the successful execution of the next instruction. When this bit is set, the processor dispatches instructions in strict program order. Successful execution means the instruction caused no other exception. Single-step tracing may not be present on all implementations.
22	BE	Branch trace enable 0 = No trace exception occurs when a branch instruction is completed 1 = Trace exception occurs when a branch instruction is completed
23	FE1	Floating-point exception mode 1 (See **Table 3-13**.)
24	—	Reserved
25	IP	Exception prefix. The setting of this bit specifies the location of the exception vector table. 0 = Exception vector table starts at the physical address 0x0000 0000. 1 = Exception vector table starts at the physical address 0xFFF0 0000.
26	IR	Instruction relocation. 0 = Instruction address translation is off, the BBC IMPU does not check for address permission attributes. 1 = Instruction address translation is on, the BBC IMPU checks for address permission attributes.
27	DR	Data relocation 0 = Data address translation is off, the L2U DMPU does not check for address permission attributes. 1 = Data address translation is on, the L2U DMPU checks for address permission attributes.
28	—	Reserved

Table 3-12 Machine State Register Bit Descriptions (Continued)

Bit(s)	Name	Description
29	DC-MPEN[1]	Decompression On/Off 0 = RCPU Normal Operation 1 = RCPU is running in Compressed mode
30	RI	Recoverable exception (for machine check and non-maskable breakpoint exceptions) 0 = Machine state is not recoverable. 1 = Machine state is recoverable.
31	LE	Little-endian mode 0 = Processor operates in big-endian mode during normal processing. 1 = Processor operates in little-endian mode during normal processing.

NOTES:
1. This option is only available on the MPC566.

The floating-point exception mode bits are interpreted as shown in **Table 3-13**.

Table 3-13 Floating-Point Exception Mode Bits

FE[0:1]	Mode
00	Ignore exceptions mode — Floating-point exceptions do not cause the floating-point assist error handler to be invoked.
01, 10, 11	Floating-point precise mode — The system floating-point assist error handler is invoked precisely at the instruction that caused the enabled exception.

3.9.2 DAE/Source Instruction Service Register (DSISR)

The 32-bit DSISR identifies the cause of data access and alignment exceptions.

DSISR — DAE/Source Instruction Service Register **SPR 18**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
DSISR																															

RESET: UNCHANGED

3.9.3 Data Address Register (DAR)

After an alignment exception, the DAR is set to the effective address of a load or store element.

DAR — Data Address Register **SPR 19**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Data Address																															

RESET: UNCHANGED

3.9.4 Time Base Facility (TB) — OEA

As described in **3.8 PowerPC VEA Register Set — Time Base**, the time base (TB) provides a 64-bit incrementing counter. The VEA defines user-level, read-only access to the TB. Writing to the TB is reserved for supervisor-level applications such as operating systems and bootstrap routines. The OEA defines supervisor-level, write access to the TB.

TB — Time Base (Write Only) **SPR 284, 285**

0 — 31	32 — 63
TBU	TBL

RESET: UNCHANGED

Table 3-14 Time Base Field (Write Only) Bit Descriptions

Bits	Name	Description
0:31	TBU	Time Base (Upper) — The high-order 32 bits of the time base
32:63	TBL	Time Base (Lower) — The low-order 32 bits of the time base

The TB can be written to at the supervisor privilege level only. The **mttbl** and **mttbu** simplified mnemonics write the lower and upper halves of the TB, respectively. The **mtspr**, **mttbl**, and **mttbu** instructions treat TBL and TBU as separate 32-bit registers; setting one leaves the other unchanged. It is not possible to write the entire 64-bit time base in a single instruction.

For information about reading the time base, refer to **3.8 PowerPC VEA Register Set — Time Base**.

3.9.5 Decrementer Register (DEC)

The decrementer (DEC, SPR 22) is a 32-bit decrementing counter defined by the MPC565 / MPC566 to provide a decrementer exception after a programmable delay. The DEC satisfies the following requirements:

- Loading a GPR from the DEC has no effect on the DEC.
- Storing a GPR to the DEC replaces the value in the DEC with the value in the GPR.
- Whenever bit 0 of the DEC changes from zero to one, a decrementer exception request (unless masked) is signaled. Multiple DEC exception requests may be received before the first exception occurs; however, any additional requests are canceled when the exception occurs for the first request.
- If the DEC is altered by software and the content of bit 0 is changed from zero to one, an exception request is signaled.
- $\overline{\text{PORESET}}$ resets and stops the decrementer, $\overline{\text{HRESET}}$/$\overline{\text{SRESET}}$ do not.

The decrementer frequency is based on a subdivision of the processor clock. A bit in the system clock control register (SCCR) in the SIU determines the clock source of both the decrementer and the time base. For details on the decrementer and time base clock in the MPC565 / MPC566, refer to **6.7 MPC565 / MPC566 Decrementer**, **8.6 MPC565 / MPC566 Internal Clock Signals**, and **8.12.1 System Clock Control Register (SCCR)**.

The DEC does not run after power-up and must be enabled by setting the TBE bit in the TBSCR register, see **Table 6-18**. Power-on reset stops its counting and clears the register. A decrementer exception may be signaled to software prior to initialization.

DEC — Decrementer Register **SPR 22**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Decrementing Counter																															

RESET: UNCHANGED

3.9.6 Machine Status Save/Restore Register 0 (SRR0)

The machine status save/restore register 0 (SRR0) is a 32-bit register that identifies where instruction execution should resume when an **rfi** instruction is executed following an exception. It also holds the effective address of the instruction that follows the system call (**sc**) instruction.

When an exception occurs, SRR0 is set to point to an instruction such that all prior instructions have completed execution and no subsequent instruction has begun execution. The instruction addressed by SRR0 may not have completed execution, depending on the exception type. SRR0 addresses either the instruction causing the exception or the immediately following instruction. The instruction addressed can be determined from the exception type and status bits.

SRR0 — Machine Status Save/Restore Register 0 **SPR 26**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SRR0																															

RESET: UNDEFINED

When an exception occurs, SRR0 is set to point to an instruction such that all prior instructions have completed execution and no subsequent instruction has begun execution. The instruction addressed by SRR0 may not have completed execution, depending on the exception type. SRR0 addresses either the instruction causing the exception or the immediately following instruction. The instruction addressed can be determined from the exception type and status bits.

3.9.7 Machine Status Save/Restore Register 1 (SRR1)

SRR1 is a 32-bit register used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.

SRR1 — Machine Status Save/Restore Register 1 **SPR 27**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SRR1																															

RESET: UNDEFINED

In general, when an exception occurs, SRR1[0:15] are loaded with exception-specific information, and MSR[16:31] are placed into SRR1[16:31].

3.9.8 General SPRs (SPRG0–SPRG3)

SPRG0–SPRG3 are 32-bit registers provided for general operating system use, such as performing a fast-state save and for supporting multiprocessor implementations. SPRG0–SPRG3 are shown below.

SPRG0–SPRG3 — General Special-Purpose Registers 0–3 **SPR 272 – SPR 275**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SPRG0																															
SPRG1																															
SPRG2																															
SPRG3																															

RESET: UNCHANGED

Uses for SPRG0–SPRG3 are shown in **Table 3-15**.

Table 3-15 Uses of SPRG0–SPRG3

Register	Description
SPRG0	Software may load a unique physical address in this register to identify an area of memory reserved for use by the exception handler. This area must be unique for each processor in the system.
SPRG1	This register may be used as a scratch register by the exception handler to save the content of a GPR. That GPR then can be loaded from SPRG0 and used as a base register to save other GPRs to memory.
SPRG2	This register may be used by the operating system as needed.
SPRG3	This register may be used by the operating system as needed.

3.9.9 Processor Version Register (PVR)

The PVR is a 32-bit, read-only register that identifies the version and revision level of the PowerPC processor. The contents of the PVR can be copied to a GPR by the **mfspr** instruction. Read access to the PVR is available in supervisor mode only; write access is not provided.

PVR — Processor Version Register **SPR 287**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
VERSION																REVISION															

RESET: UNCHANGED

Table 3-16 Processor Version Register Bit Descriptions

Bit(s)	Name	Description
0:15	VERSION	A 16-bit number that identifies the version of the processor and of the PowerPC architecture. MPC565 / MPC566 value is 0x0002.
16:31	REVISION	A 16-bit number that distinguishes between various releases of a particular version. The MPC565 / MPC566 value is 0x0020.

3.9.10 Implementation-Specific SPRs

The MPC565 / MPC566 includes several implementation-specific SPRs that are not defined by the PowerPC architecture. These registers can be accessed by supervisor-level instructions only. These registers are listed in **Table 3-2** and **Table 3-3**.

3.9.10.1 EIE, EID, and NRI Special-Purpose Registers

The RCPU includes three implementation-specific SPRs to facilitate the software manipulation of the MSR[RI] and MSR[EE] bits. Issuing the **mtspr** instruction with one of these registers as an operand causes the RI and EE bits to be set or cleared as shown in **Table 3-17**.

A read (**mfspr**) of any of these locations is treated as an unimplemented instruction, resulting in a software emulation exception.

Table 3-17 EIE, EID, AND NRI Registers

SPR Number (Decimal)	Mnemonic	MSR[EE]	MSR[RI]
80	EIE	1	1
81	EID	0	1
82	NRI	0	0

3.9.10.2 Floating-Point Exception Cause Register (FPECR)

The FPECR is a 32-bit supervisor-level internal status and control register used by the floating-point assist firmware envelope. It contains four status bits indicating whether the result of the operation is tiny and whether any of three source operands are denormalized. In addition, it contains one control bit to enable or disable SIE mode. This register must not be accessed by user code.

FPECR — Floating-Point Exception Cause Register **SPR 1022**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SIE	RESERVED														
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED												DNC	DNB	DNA	TR
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A listing of FPECR bit settings is shown in **Table 3-18**.

Table 3-18 FPECR Bit Descriptions

Bit(s)	Name	Description
0	SIE	SIE mode control bit 0 = Disable SIE mode 1 = Enable SIE mode
[1:27]	—	Reserved
28	DNC	Source operand C denormalized status bit 0 = Source operand C is not denormalized 1 = Source operand C is denormalized
29	DNB	Source operand B denormalized status bit 0 = Source operand B is not denormalized 1 = Source operand B is denormalized
30	DNA	Source operand A denormalized status bit 0 = Source operand A is not denormalized 1 = Source operand A is denormalized
31	TR	Floating-point tiny result 0 = Floating-point result is not tiny 1 = Floating-point result is tiny

NOTE

Software must insert a **sync** instruction before reading the FPECR.

3.9.10.3 Additional Implementation-Specific Registers

Refer to the following sections for details on additional implementation-specific registers in the MPC565 / MPC566:

- **4.8 BBC Programming Model**
- **6.14.1.2 Internal Memory Map Register**
- **11.8 L2U Programming Model**
- **SECTION 22 DEVELOPMENT SUPPORT**

3.10 Instruction Set

All PowerPC instructions are encoded as single words (32 bits). Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

The PowerPC instructions are divided into the following categories:

- Integer instructions include computational and logical instructions.
  - — Integer arithmetic instructions
  - — Integer compare instructions
  - — Integer logical instructions
  - — Integer rotate and shift instructions
- Floating-point instructions include floating-point computational instructions, as well as instructions that affect the floating-point status and control register (FPSCR).
  - — Floating-point arithmetic instructions
  - — Floating-point multiply/add instructions
  - — Floating-point rounding and conversion instructions
  - — Floating-point compare instructions
  - — Floating-point status and control instructions
- Load/store instructions include integer and floating-point load and store instructions.
  - — Integer load and store instructions
  - — Integer load and store multiple instructions
  - — Floating-point load and store
  - — Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - — Branch and trap instructions
  - — Condition register logical instructions
- Processor control instructions are used for synchronizing memory accesses.
  - — Move to/from SPR instructions
  - — Move to/from MSR
  - — Synchronize
  - — Instruction synchronize

NOTE

This grouping of the instructions does not indicate which execution unit executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are

four bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs.

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

PowerPC processors follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

3.10.1 Instruction Set Summary

Table 3-19 provides a summary of RCPU instructions. Refer to the ***RCPU Reference Manual* (RCPURM/AD)** for a detailed description of the instruction set.

Table 3-19 Instruction Set Summary

Mnemonic	Operand Syntax	Name
add (add. addo addo.)	**r**D,**r**A,**r**B	Add
addc (addc. addco addco.)	**r**D,**r**A,**r**B	Add Carrying
adde (adde. addeo addeo.)	**r**D,**r**A,**r**B	Add Extended
addi	**r**D,**r**A,SIMM	Add Immediate
addic	**r**D,**r**A,SIMM	Add Immediate Carrying
addic.	**r**D,**r**A,SIMM	Add Immediate Carrying and Record
addis	**r**D,**r**A,SIMM	Add Immediate Shifted
addme (addme. addmeo addmeo.)	**r**D,**r**A	Add to Minus One Extended
addze (addze. addzeo addzeo.)	**r**D,**r**A	Add to Zero Extended
and (and.)	**r**A,**r**S,**r**B	AND
andc (andc.)	**r**A,**r**S,**r**B	AND with Complement
andi.	**r**A,**r**S,UIMM	AND Immediate
andis.	**r**A,**r**S,UIMM	AND Immediate Shifted
b (ba bl bla)	target_addr	Branch
bc (bca bcl bcla)	BO,BI,target_addr	Branch Conditional
bcctr (bcctrl)	BO,BI	Branch Conditional to Count Register
bclr (bclrl)	BO,BI	Branch Conditional to Link Register
cmp	**crf**D,L,**r**A,**r**B	Compare
cmpi	**crf**D,L,**r**A,SIMM	Compare Immediate
cmpl	**crf**D,L,**r**A,**r**B	Compare Logical
cmpli	**crf**D,L,**r**A,UIMM	Compare Logical Immediate
cntlzw (cntlzw.)	**r**A,**r**S	Count Leading Zeros Word
crand	**crb**D,**crb**A, **crb**B	Condition Register AND
crandc	**crb**D,**crb**A, **crb**B	Condition Register AND with Complement
creqv	**crb**D,**crb**A, **crb**B	Condition Register Equivalent

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
crnand	**crb**D,**crb**A,**crb**B	Condition Register NAND
crnor	**crb**D,**crb**A,**crb**B	Condition Register NOR
cror	**crb**D,**crb**A, **crb**B	Condition Register OR
crorc	**crb**D,**crb**A, **crb**B	Condition Register OR with Complement
crxor	**crb**D,**crb**A, **crb**B	Condition Register XOR
divw (divw. divwo divwo.)	**r**D,**r**A,**r**B	Divide Word
divwu divwu. divwuo divwuo.	**r**D,**r**A,**r**B	Divide Word Unsigned
eieio	—	Enforce In-Order Execution of I/O
eqv (eqv.)	**r**A,**r**S,**r**B	Equivalent
extsb (extsb.)	**r**A,**r**S	Extend Sign Byte
extsh (extsh.)	**r**A,**r**S	Extend Sign Half Word
fabs (fabs.)	**fr**D,**fr**B	Floating Absolute Value
fadd (fadd.)	**fr**D,**fr**A,**fr**B	Floating Add (Double-Precision)
fadds (fadds.)	**fr**D,**fr**A,**fr**B	Floating Add Single
fcmpo	**crf**D,**fr**A,**fr**B	Floating Compare Ordered
fcmpu	**crf**D,**fr**A,**fr**B	Floating Compare Unordered
fctiw (fctiw.)	**fr**D,**fr**B	Floating Convert to Integer Word
fctiwz (fctiwz.)	**fr**D,**fr**B	Floating Convert to Integer Word with Round Toward Zero
fdiv (fdiv.)	**fr**D,**fr**A,**fr**B	Floating Divide (Double-Precision)
fdivs (fdivs.)	**fr**D,**fr**A,**fr**B	Floating Divide Single
fmadd (fmadd.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Add (Double-Precision)
fmadds (fmadds.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Add Single
fmr (fmr.)	**fr**D,**fr**B	Floating Move Register
fmsub (fmsub.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Subtract (Double-Precision)
fmsubs (fmsubs.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Multiply-Subtract Single
fmul (fmul.)	**fr**D,**fr**A,**fr**C	Floating Multiply (Double-Precision)
fmuls (fmuls.)	**fr**D,**fr**A,**fr**C	Floating Multiply Single
fnabs (fnabs.)	**fr**D,**fr**B	Floating Negative Absolute Value
fneg (fneg.)	**fr**D,**fr**B	Floating Negate
fnmadd (fnmadd.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Add (Double-Precision)
fnmadds (fnmadds.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Add Single
fnmsub (fnmsub.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Subtract (Double-Precision)
fnmsubs (fnmsubs.)	**fr**D,**fr**A,**fr**C,**fr**B	Floating Negative Multiply-Subtract Single
frsp (frsp.)	**fr**D,**fr**B	Floating Round to Single
fsub (fsub.)	**fr**D,**fr**A,**fr**B	Floating Subtract (Double-Precision)
fsubs (fsubs.)	**fr**D,**fr**A,**fr**B	Floating Subtract Single
isync	—	Instruction Synchronize
lbz	**r**D,d(**r**A)	Load Byte and Zero
lbzu	**r**D,d(**r**A)	Load Byte and Zero with Update
lbzux	**r**D,**r**A,**r**B	Load Byte and Zero with Update Indexed

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
lbzx	**r**D,**r**A,**r**B	Load Byte and Zero Indexed
lfd	**fr**D,d(**r**A)	Load Floating-Point Double
lfdu	**fr**D,d(**r**A)	Load Floating-Point Double with Update
lfdux	**fr**D,**r**A,**r**B	Load Floating-Point Double with Update Indexed
lfdx	**fr**D,**r**A,**r**B	Load Floating-Point Double Indexed
lfs	**fr**D,d(**r**A)	Load Floating-Point Single
lfsu	**fr**D,d(**r**A)	Load Floating-Point Single with Update
lfsux	**fr**D,**r**A,**r**B	Load Floating-Point Single with Update Indexed
lfsx	**fr**D,**r**A,**r**B	Load Floating-Point Single Indexed
lha	**r**D,d(**r**A)	Load Half-Word Algebraic
lhau	**r**D,d(**r**A)	Load Half-Word Algebraic with Update
lhaux	**r**D,**r**A,**r**B	Load Half-Word Algebraic with Update Indexed
lhax	**r**D,**r**A,**r**B	Load Half-Word Algebraic Indexed
lhbrx	**r**D,**r**A,**r**B	Load Half-Word Byte-Reverse Indexed
lhz	**r**D,d(**r**A)	Load Half-Word and Zero
lhzu	**r**D,d(**r**A)	Load Half-Word and Zero with Update
lhzux	**r**D,**r**A,**r**B	Load Hal-Word and Zero with Update Indexed
lhzx	**r**D,**r**A,**r**B	Load Half-Word and Zero Indexed
lmw	**r**D,d(**r**A)	Load Multiple Word
lswi	**r**D,**r**A,NB	Load String Word Immediate
lswx	**r**D,**r**A,**r**B	Load String Word Indexed
lwarx	**r**D,**r**A,**r**B	Load Word and Reserve Indexed
lwbrx	**r**D,**r**A,**r**B	Load Word Byte-Reverse Indexed
lwz	**r**D,d(**r**A)	Load Word and Zero
lwzu	**r**D,d(**r**A)	Load Word and Zero with Update
lwzux	**r**D,**r**A,**r**B	Load Word and Zero with Update Indexed
lwzx	**r**D,**r**A,**r**B	Load Word and Zero Indexed
mcrf	**crf**D,**crf**S	Move Condition Register Field
mcrfs	**crf**D,**crf**S	Move to Condition Register from FPSCR
mcrxr	**crf**D	Move to Condition Register from XER
mfcr	**r**D	Move from Condition Register
mffs (mffs.)	**fr**D	Move from FPSCR
mfmsr	**r**D	Move from Machine State Register
mfspr	**r**D,SPR	Move from Special Purpose Register
mftb	**r**D, TBR	Move from Time Base
mtcrf	CRM,**r**S	Move to Condition Register Fields
mtfsb0 (mtfsb0.)	**crb**D	Move to FPSCR Bit 0
mtfsb1 (mtfsb1.)	**crb**D	Move to FPSCR Bit 1
mtfsf (mtfsf.)	FM,**fr**B	Move to FPSCR Fields
mtfsfi (mtfsfi.)	**crf**D,IMM	Move to FPSCR Field Immediate
mtmsr	**r**S	Move to Machine State Register

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
mtspr	SPR,**r**S	Move to Special Purpose Register
mulhw (mulhw.)	**r**D,**r**A,**r**B	Multiply High Word
mulhwu (mulhwu.)	**r**D,**r**A,**r**B	Multiply High Word Unsigned
mulli	**r**D,**r**A,SIMM	Multiply Low Immediate
mullw (mullw. mullwo mullwo.)	**r**D,**r**A,**r**B	Multiply Low
nand (nand.)	**r**A,**r**S,**r**B	NAND
neg (neg. nego nego.)	**r**D,**r**A	Negate
nor (nor.)	**r**A,**r**S,**r**B	NOR
or (or.)	**r**A,**r**S,**r**B	OR
orc (orc.)	**r**A,**r**S,**r**B	OR with Complement
ori	**r**A,**r**S,UIMM	OR Immediate
oris	**r**A,**r**S,UIMM	OR Immediate Shifted
rfi	—	Return from Interrupt
rlwimi (rlwimi.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then Mask Insert
rlwinm (rlwinm.)	**r**A,**r**S,SH,MB,ME	Rotate Left Word Immediate then AND with Mask
rlwnm (rlwnm.)	**r**A,**r**S,**r**B,MB,ME	Rotate Left Word then AND with Mask
sc	—	System Call
slw (slw.)	**r**A,**r**S,**r**B	Shift Left Word
sraw (sraw.)	**r**A,**r**S,**r**B	Shift Right Algebraic Word
srawi (srawi.)	**r**A,**r**S,SH	Shift Right Algebraic Word Immediate
srw (srw.)	**r**A,**r**S,**r**B	Shift Right Word
stb	**r**S,d(**r**A)	Store Byte
stbu	**r**S,d(**r**A)	Store Byte with Update
stbux	**r**S,**r**A,**r**B	Store Byte with Update Indexed
stbx	**r**S,**r**A,**r**B	Store Byte Indexed
stfd	**fr**S,d(**r**A)	Store Floating-Point Double
stfdu	**fr**S,d(**r**A)	Store Floating-Point Double with Update
stfdux	**fr**S,**r**B	Store Floating-Point Double with Update Indexed
stfdx	**fr**S,**r**B	Store Floating-Point Double Indexed
stfiwx	**fr**S,**r**B	Store Floating-Point as Integer Word Indexed
stfs	**fr**S,d(**r**A)	Store Floating-Point Single
stfsu	**fr**S,d(**r**A)	Store Floating-Point Single with Update
stfsux	**fr**S,**r**B	Store Floating-Point Single with Update Indexed
stfsx	**fr**S,**r** B	Store Floating-Point Single Indexed
sth	**r**S,d(**r**A)	Store Half Word
sthbrx	**r**S,**r**A,**r**B	Store Half Word Byte-Reverse Indexed
sthu	**r**S,d(**r**A)	Store Half Word with Update
sthux	**r**S,**r**A,**r**B	Store Half Word with Update Indexed
sthx	**r**S,**r**A,**r**B	Store Half Word Indexed
stmw	**r**S,d(**r**A)	Store Multiple Word

Table 3-19 Instruction Set Summary (Continued)

Mnemonic	Operand Syntax	Name
stswi	**r**S,**r**A,NB	Store String Word Immediate
stswx	**r**S,**r**A,**r**B	Store String Word Indexed
stw	**r**S,d(**r**A)	Store Word
stwbrx	**r**S,**r**A,**r**B	Store Word Byte-Reverse Indexed
stwcx.	**r**S,**r**A,**r**B	Store Word Conditional Indexed
stwu	**r**S,d(**r**A)	Store Word with Update
stwux	**r**S,**r**A,**r**B	Store Word with Update Indexed
stwx	**r**S,**r**A,**r**B	Store Word Indexed
subf (subf. subfo subfo.)	**r**D,**r**A,**r**B	Subtract From
subfc (subfc. subfco subfco.)	**r**D,**r**A,**r**B	Subtract from Carrying
subfe (subfe. subfeo subfeo.)	**r**D,**r**A,**r**B	Subtract from Extended
subfic	**r**D,**r**A,SIMM	Subtract from Immediate Carrying
subfme (subfme. subfmeo subfmeo.)	**r**D,**r**A	Subtract from Minus One Extended
subfze (subfze. subfzeo subfzeo.)	**r**D,**r**A	Subtract from Zero Extended
sync	—	Synchronize
tw	TO,**r**A,**r**B	Trap Word
twi	TO,**r**A,SIMM	Trap Word Immediate
xor (xor.)	**r**A,**r**S,**r**B	XOR
xori	**r**A,**r**S,UIMM	XOR Immediate
xoris	**r**A,**r**S,UIMM	XOR Immediate Shifted

3.10.2 Recommended Simplified Mnemonics

To simplify assembly language coding, a set of alternative mnemonics is provided for some frequently used operations (such as no-op, load immediate, load address, move register, and complement register).

For a complete list of simplified mnemonics, see the ***RCPU Reference Manual* (RCPURM/AD)**. Programs written to be portable across the various assemblers for the PowerPC architecture should not assume the existence of mnemonics not described in that manual.

3.10.3 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + 16-bit offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the storage operand is considered to wrap around from the maximum effective address to effective address 0.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

3.11 Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions. When exceptions occur, information about the state of the processor is saved to certain registers, and the processor begins execution at an address (exception vector) predetermined for each exception. Processing of exceptions occurs in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception — for example, the DAE/source instruction service register (DSISR). Additionally, some exception conditions can be explicitly enabled or disabled by software.

3.11.1 Exception Classes

The MPC565 / MPC566 exception classes are shown in **Table 3-20**.

Table 3-20 MPC565 / MPC566 Exception Classes

Class	Exception Type
Asynchronous, unordered	Machine check System reset
Asynchronous, ordered	External interrupt Decrementer
Synchronous (ordered, precise)	Instruction-caused exceptions

3.11.2 Ordered Exceptions

In the MPC565 / MPC566, all exceptions except for reset, debug port non-maskable interrupts, and machine check exceptions are ordered. Ordered exceptions satisfy the following criteria:

- Only one exception is reported at a time. If, for example, a single instruction encounters multiple exception conditions, those conditions are encountered sequentially. After the exception handler handles an exception, instruction execution continues until the next exception condition is encountered.
- When the exception is taken, no program state is lost.

3.11.3 Unordered Exceptions

Unordered exceptions may be reported at any time and are not guaranteed to preserve program state information. The processor can never recover from a reset exception. It can recover from other unordered exceptions in most cases. However, if a debug port non-maskable interrupt or machine check exception occurs during the servicing of a previous exception, the machine state information in SRR0 and SRR1 (and, in some cases, the DAR and DSISR) may not be recoverable; the processor may be in the process of saving or restoring these registers.

To determine whether the machine state is recoverable, the RI (recoverable exception) bit in SRR1 can be read. During exception processing, the RI bit in the MSR is copied to SRR1 and then cleared. The operating system should set the RI bit in the MSR at the end of each exception handler's prologue (after saving the program state) and clear the bit at the start of each exception handler's epilogue (before restoring the program state). Then, if an unordered exception occurs during the servicing of an exception handler, the RI bit in SRR1 will contain the correct value.

3.11.4 Precise Exceptions

In the MPC565 / MPC566, all synchronous (instruction-caused) exceptions are precise. When a precise exception occurs, the processor backs the machine up to the instruction causing the exception. This ensures that the machine is in its correct architecturally-defined state. The following conditions exist at the point a precise exception occurs:

1. Architecturally, no instruction following the faulting instruction in the code stream has begun execution.
2. All instructions preceding the faulting instruction appear to have completed with respect to the executing processor.
3. SRR0 addresses either the instruction causing the exception or the immediately following instruction. Which instruction is addressed can be determined from the exception type and the status bits.
4. Depending on the type of exception, the instruction causing the exception may not have begun execution, may have partially completed, or may have completed execution.

3.11.5 Exception Vector Table

The setting of the exception prefix (IP) bit in the MSR determines how exceptions are vectored. If the bit is cleared, the exception vector table begins at the physical address 0x0000 0000; if IP is set, the exception vector table begins at the physical address 0xFFF0 0000. **Table 3-21** shows the exception vector offset of the first instruction of the exception handler routine for each exception type.

NOTE

In the MPC565 / MPC566, the exception table can additionally be relocated by the BBC module to internal memory and reduce the total size required by the exception table (see **4.5 Exception Table Relocation (ETR)**.

Table 3-21 Exception Vector Offset Table

Vector Offset (Hexadecimal)	Exception Type
00000	Reserved
00100	System reset, NMI interrupt
00200	Machine check
00300	Reserved
00400	Reserved
00500	External Interrupt
00600	Alignment
00700	Program
00800	Floating-Point Unavailable
00900	Decrementer
00A00	Reserved
00B00	Reserved
00C00	System call
00D00	Trace
00E00	Floating-Point Assist
01000	Implementation-Dependent Software Emulation
01100	Reserved
01200	Reserved
01300	Implementation-Dependent Instruction Protection Error
01400	Implementation-Dependent Data Protection Error
01500–01BFF	Reserved
01C00	Implementation-Dependent Data Breakpoint
01D00	Implementation-Dependent Instruction Breakpoint
01E00	Implementation-Dependent Maskable External Breakpoint
01F00	Implementation-Dependent Non-maskable External Breakpoint

3.12 Instruction Timing

The MPC565 / MPC566 processor is pipelined. Because the processing of an instruction is broken into a series of stages, an instruction does not require the entire resources of the processor.

The instruction pipeline in the MPC565 / MPC566 has four stages:

1. The dispatch stage is implemented using a distributed mechanism. The central dispatch unit broadcasts the instruction to all units. In addition, scoreboard information (regarding data dependencies) is broadcast to each execution unit.

Each execution unit decodes the instruction. If the instruction is not implemented, a program exception is taken. If the instruction is legal and no data dependency is found, the instruction is accepted by the appropriate execution unit, and the data found in the destination register is copied to the history buffer. If a data dependency exists, the machine is stalled until the dependency is resolved.

2. In the execute stage, each execution unit that has an executable instruction executes the instruction. (For some instructions, this occurs over multiple cycles.)
3. In the writeback stage, the execution unit writes the result to the destination register and reports to the history buffer that the instruction is completed.
4. In the retirement stage, the history buffer retires instructions in architectural order. An instruction retires from the machine if it completes execution with no exceptions and if all instructions preceding it in the instruction stream have finished execution with no exceptions. As many as six instructions can be retired in one clock.

The history buffer maintains the correct architectural machine state. An exception is taken only when the instruction is ready to be retired from the machine (i.e., after all previously-issued instructions have already been retired from the machine). When an exception is taken, all instructions following the excepting instruction are canceled, (i.e., the values of the affected destination registers are restored using the values saved in the history buffer during the dispatch stage).

Figure 3-4 shows basic instruction pipeline timing.

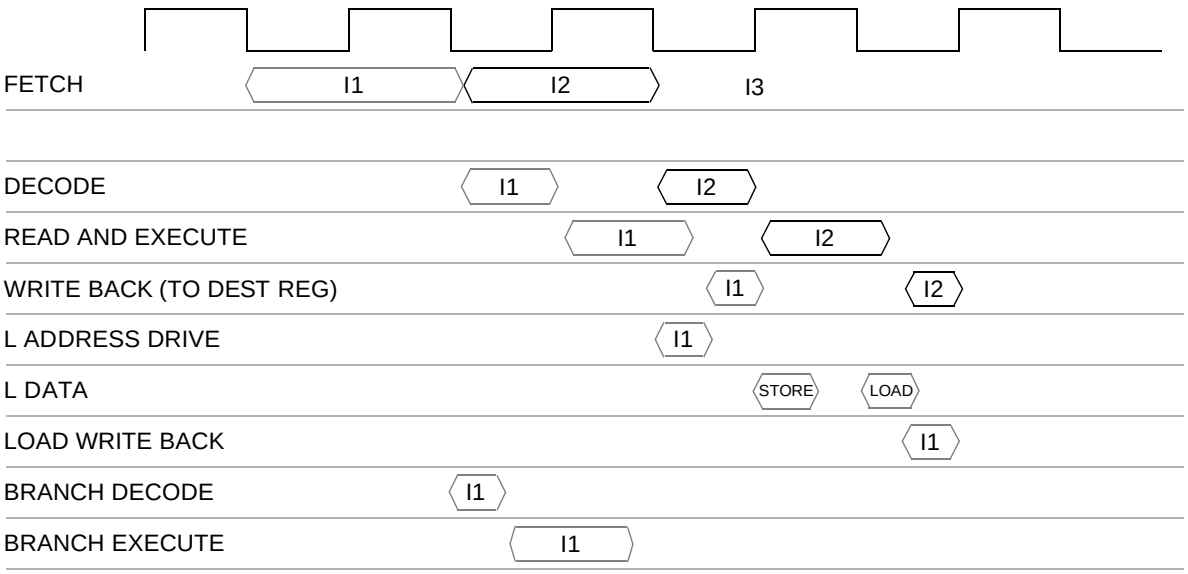


Figure 3-4 Basic Instruction Pipeline

Table 3-22 indicates the latency and blockage for each type of instruction. Latency refers to the interval from the time an instruction begins execution until it produces a result that is available for use by a subsequent instruction. Blockage refers to the interval from the time an instruction begins execution until its execution unit is available for a subsequent instruction.

NOTE

When the blockage equals the latency, it is not possible to issue another instruction to the same unit in the same cycle in which the first instruction is being written back.

Table 3-22 Instruction Latency and Blockage

Instruction Type	Precision	Latency	Blockage
Floating-point multiply-add	Double Single	7 6	7 6
Floating-point add or subtract	Double Single	4 4	4 4
Floating-point multiply	Double Single	5 4	5 4
Floating-point divide	Double Single	17 10	17 10
Integer multiply	—	2	1 or 2[1]
Integer divide	—	2 to 11[1]	2 to 11[1]
Integer load/store	—	See note[1]	See note[1]

NOTES:
1. Refer to Section 7 Instruction Timing, in the ***RCPU Reference Manual* (RCPURM/AD)** for details.

3.13 PowerPC User Instruction Set Architecture (UISA)

3.13.1 Computation Modes

The core of the MPC565 / MPC566 is a 32-bit implementation of the PowerPC architecture. Any reference in the PowerPC Architecture books (UISA, VEA, OEA) regarding 64-bit implementations are not supported by the core. All registers except the floating-point registers are 32 bits wide.

3.13.2 Reserved Fields

Reserved fields in instructions are described under the specific instruction definition sections. Unless otherwise stated in the specific instruction description, fields marked ***"I"***, "***II"*** and "***III"*** in the instruction are discarded by the core decoding. Thus, this type of invalid form instructions yield results of the defined instructions with the appropriate field zero.

In most cases, the reserved fields in registers are ignored on write and return zeros for them on read on any control register implemented by the MPC565 / MPC566. Exception to this rule are bits [16:23] of the fixed-point exception cause register (XER) and

the reserved bits of the machine state register (MSR), which are set by the source value on write and return the value last set for it on read.

3.13.3 Classes of Instructions

Non-optional instructions are implemented by the hardware. Optional instructions are executed by implementation-dependent code and any attempt to execute one of these commands causes the MPC565 / MPC566 to take the implementation-dependent software emulation interrupt (offset 0x01000 of the vector table).

Illegal and reserved instruction class instructions are supported by implementation-dependent code and, thus, the MPC565 / MPC566 hardware generates the implementation-dependent software emulation interrupt. Invalid and preferred instruction forms treatment by the MPC565 / MPC566 is described under the specific processor compliance sections.

3.13.4 Exceptions

Invocation of the system software for any instruction-caused exception in the MPC565 / MPC566 is precise, regardless of the type and setting.

3.13.5 The Branch Processor

3.13.6 Instruction Fetching

The core fetches a number of instructions into its internal buffer (the instruction pre-fetch queue) prior to execution. If a program modifies the instructions it intends to execute, it should call a system library program to ensure that the modifications have been made visible to the instruction fetching mechanism prior to execution of the modified instructions.

3.13.7 Branch Instructions

The core implements all the instructions defined for the branch processor by the **UISA** in the hardware. For performance of various instructions, refer to **Table 3-22** of this manual.

3.13.7.1 Invalid Branch Instruction Forms

Bits marked with ***z*** in the BO encoding definition are discarded by the MPC565 / MPC566 decoding. Thus, these types of invalid form instructions yield result of the defined instructions with the ***z*** bit zero. If the decrement and test CTR option is specified for the **bcctr** or **bcctrl** instructions, the target address of the branch is the new value of the CTR. Condition is evaluated correctly, including the value of the counter after decrement.

3.13.7.2 Branch Prediction

The core uses the ***y*** bit to predict path for pre-fetch. Prediction is only done for not-ready branch conditions. No prediction is done for branches to link or count register if

the target address is not ready. Refer to the ***RCPU Reference Manual*** **(RCPURM/AD)** (conditional branch control) for more information.

3.13.8 The Fixed-Point Processor

3.13.8.1 Fixed-Point Instructions

The core implements the following instructions:

- Fixed-point arithmetic instructions
- Fixed-point compare instructions
- Fixed-point trap instructions
- Fixed-point logical instructions
- Fixed-point rotate and shift instructions
- Move to/from system register instructions

All instructions are defined for the fixed-point processor in the **UISA** in the hardware. For performance of the various instructions, refer to **Table 3-22**.

— **Move To/From System Register Instructions.** Move to/from invalid special registers in which spr0 = 1 yields invocation of the privilege instruction error interrupt handler if the processor is in problem state. For a list of all implemented special registers, refer to **Table 3-2**, and **Table 3-3.**

— **Fixed-Point Arithmetic Instructions.** If an attempt is made to perform any of the divisions in the divw[o][.] instruction:
0x80000000 ÷ -1
<anything> ÷ 0
Then, the contents of RT are 0x80000000 and if Rc =1, the contents of bits in CR field 0 are LT = 1, GT = 0, EQ = 0, and SO is set to the correct value. If an attempt is made to perform any of the divisions in the divw[o][.] instruction, <anything> ÷ 0. Then, the contents of RT are 0x80000000 and if Rc = 1, the contents of bits in CR field 0 are LT = 1, GT = 0, EQ = 0, and SO is set to the correct value. In cmpi, cmp, cmpli, and cmpl instructions, the L-bit is applicable for 64-bit implementations. In 32-bit implementations, if L = 1 the instruction form is invalid. The core ignores this bit and therefore, the behavior when L = 1 is identical to the valid form instruction with L= 0

3.13.9 Floating-Point Processor

3.13.9.1 General

The MPC565 / MPC566 implements all floating-point features as defined in the **UISA**, including the non-IEEE working mode. Some features require software assistance. For more information refer to the ***RCPU Reference Manual*** **(RCPURM/AD)** (Floating-point Load Instructions) for more information.

3.13.9.2 Optional instructions

The only optional instruction implemented by MPC565 / MPC566 hardware is store floating-point as integer word indexed (**stfiwx**). An attempt to execute any other optional instruction causes the implementation dependent software emulation interrupt to be taken.

3.13.10 Load/Store Processor

The load/store processor supports all of the 32-bit implementation fixed-point PowerPC load/store instructions in the hardware.

3.13.10.1 Fixed-Point Load With Update and Store With Update Instructions

For load with update and store with update instructions, where RA = 0, the EA is written into R0. For load with update instructions, where RA = RT, RA is boundedly undefined.

3.13.10.2 Fixed-Point Load and Store Multiple Instructions

For these types of instructions, EA must be a multiple of four. If it is not, the system alignment error handler is invoked. For a **lmw** instruction (if RA is in the range of registers to be loaded), the instruction completes normally. RA is then loaded from the memory location as follows:

$$RA \leftarrow MEM(EA+(RA-RT)*4, 4)$$

3.13.10.3 Fixed-Point Load String Instructions

Load string instructions behave the same as load multiple instructions, with respect to invalid format in which RA is in the range of registers to be loaded. In case RA is in the range, it is updated from memory.

3.13.10.4 Storage Synchronization Instructions

For these type of instructions, EA must be a multiple of four. If it is not, the system alignment error handler is invoked.

3.13.10.5 Floating-Point Load and Store With Update Instructions

For Load and Store with update instructions, if RT = 0 then the EA is written into R0.

3.13.10.6 Floating-Point Load Single Instructions

In case the operand falls in the range of a single denormalized number the floating-point assist interrupt handler is invoked.

Refer to the ***RCPU Reference Manual*** **(RCPURM/AD)** (Floating-point Assist For Denormalized Operands) for complete description of handling denormalized floating-point numbers.

3.13.10.7 Floating-Point Store Single Instructions

In case the operand falls in the range of a single denormalized number, the floating-point assist interrupt handler is invoked.

In case the operand is ZERO it is converted to the correct signed ZERO in single-precision format.

In case the operand is between the range of single denormalized and double denormalized it is considered a programming error. The hardware will handle this case as if the operand was single denormalized.

In case the operand falls in the range of double denormalized numbers it is considered a programming error. The hardware will handle this case as if the operand was ZERO.

The following check is done on the stored operand in order to determine whether it is a denormalized single-precision operand and invoke the floating-point assist interrupt handler:

$$(FRS_{1:11} \neq 0) \text{ AND } (FRS_{1:11} \leq 896)$$

Refer to the ***RCPU Reference Manual* (RCPURM/AD)** (Floating-Point Assist for Denormalized Operands) for complete description of handling denormalized floating-point numbers.

3.13.10.8 Optional Instructions

No optional instructions are supported.

3.13.10.9 Little-Endian Byte Ordering

The load/store unit supports little-endian byte ordering as specified in the **UISA**. In little-endian mode, if an attempt is made to execute an individual scalar unaligned transfer, as well as a multiple or string instruction, an alignment interrupt is taken.

NOTE

The USIU and the BBC do not support little endian mode.

3.14 PowerPC Virtual Environment Architecture (VEA)

3.14.1 Atomic Update Primitives

Both the **lwarx** and **stwcx** instructions are implemented according to the PowerPC architecture requirements. The MPC565 / MPC566 does not provide support for snooping an external bus activity outside the chip. The provision is made to cancel the reservation inside the MPC565 / MPC566 by using the CR_B and KR_B input pins.

3.14.2 Effect of Operand Placement on Performance

The load/store unit hardware supports all of the PowerPC load/store instructions. An optimal performance is obtained for naturally aligned operands. These accesses result

in optimal performance (one bus cycle) for up to 4 bytes in size and good performance (two bus cycles) for double precision floating-point operands. Unaligned operands are supported in hardware and are broken into a series of aligned transfers. The effect of operand placement on performance is as stated in the **VEA,** except for the case of 8-byte operands. In that case, since the MPC565 / MPC566 uses a 32-bit wide data bus, the performance is good rather than optimal**.**

3.14.3 Storage Control Instructions

The MPC565 / MPC566 does not implement cache control instructions (**icbi**, **isync**, **dcbt**, **dcbi**, **dcbf**, **dcbz**, **dcbst**, and **dcbtst**) .

3.14.4 Instruction Synchronize (`isync`) Instruction

The **`isync`** instruction causes a reflect which waits for all prior instructions to complete and then executes the next sequential instruction. Any instruction after an **`isync`** will see all effects of prior instructions.

3.14.4.1 Enforce In-Order Execution of I/O (`eieio`) Instruction

When executing an **`eieio`** instruction, the load/store unit will wait until all previous accesses have terminated before issuing cycles associated with load/store instructions following the **`eieio`** instruction.

3.14.5 Timebase

A description of the timebase register may be found in **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** and in **SECTION 8 CLOCKS AND POWER CONTROL**.

3.15 PowerPC Operating Environment Architecture (OEA)

The MPC565 / MPC566 has an internal memory space that includes memory-mapped control registers and internal memory used by various modules on the chip. This memory is part of the main memory as seen by the MPC565 / MPC566 but cannot be accessed by any external system master.

3.15.1 Branch Processor Registers

3.15.1.1 Machine State Register (MSR)

The Floating-Point exception mode encoding in the MPC565 / MPC566 core is as follows:

Table 3-23 Floating-Point Exception Mode Encoding

Mode	FE0	FE1
Ignore exceptions	0	0
Precise	0	1

Table 3-23 Floating-Point Exception Mode Encoding

Mode	FE0	FE1
Precise	1	0
Precise	1	1

The SF bit is reserved set to zero

The IP bit initial state after reset is set as programmed by the reset configuration as specified by the USIU characteristics.

3.15.1.2 Branch Processors Instructions

The MPC565 / MPC566 implements all the instructions defined for the branch processor in the **UISA** in the hardware.

3.15.2 Fixed-Point Processor

3.15.2.1 Special Purpose Registers

- **Unsupported Registers —** The following registers are not supported by the MPC565 / MPC566: SDR, EAR, IBAT0U, IBAT0L, IBAT1U, IBAT1L, IBAT2U, IBAT2L, IBAT3U, IBAT3L, DBAT0U, DBAT0L, DBAT1U, DBAT1L, DBAT2L, DBAT3U, DBAT3L
- **Added Registers —** For a list of added special purpose registers, refer to **Table 3-2**, and **Table 3-3**.

3.15.3 Storage Control Instructions

Storage control Instructions **mtsr, mtsrin, mfsr, mfsrin, dcbi, tlbie, tlbia,** and **tlbsync** are not implemented by the MPC565 / MPC566.

3.15.4 Interrupts

The core implements all storage-associated interrupts as precise interrupts. This means that a load/store instruction is not complete until all possible error indications have been sampled from the load/store bus. This also implies that a store, or a nonspeculative load instruction is not issued to the load/store bus until all previous instructions have completed. In case of a late error, a store cycle (or a nonspeculative load cycle) can be issued and then aborted.

In each interrupt handler, when registers SRR0 and SRR1 are saved, MSR_{RI} can be set to 1.

The following paragraphs define the types of OEA interrupts The exception table vector defines the offset value by interrupt type. Refer to **Table 3-21**.

3.15.4.1 System Reset Interrupt

A system reset interrupt occurs when the IRQ0 pin is asserted and the following registers are set.

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that the processor attempts to execute next if no interrupt conditions are present
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

3.15.4.2 Machine Check Interrupt

A machine check interrupt indication is received from the U-bus as a possible response either to the address or data phase. It is usually caused by one of the following conditions:

- The accessed address does not exist
- A data error is detected

As defined in the **OEA**, machine check interrupts are enabled when MSR_{ME} = 1. If MSR_{ME} = 0 and a machine check interrupt indication is received, the processor enters the checkstop state. The behavior of the MPC565 / MPC566 in checkstop state is dependent on the working mode as defined in **22.4.1.1 Debug Mode Enable vs. Debug Mode Disable**. When the processor is in debug mode enable, it enters the debug mode instead of the checkstop state. When in debug mode disable, instruction processing is suspended and cannot be restarted without resetting the core.

An indication is sent to the SIU which may generate an automatic reset in this condition. Refer to **SECTION 7 RESET** for more details. If the machine check interrupt is enabled, MSR_{ME} = 1, it is taken. If SRR1 Bit 30 = 1, the interrupt is recoverable and the following registers are set.

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt

Register Name	Bits	Description
Save/Restore Register 1 (SRR1)	1	Set to 1 for instruction fetch-related errors and 0 for load/ store-related errors
	2:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

For load/store bus cases, these registers are also set:

Register Name	Bits	Description
Data/Storage Interrupt Status Register (DSISR)	0:14	Set to 0
	15:16	Set to bits 29:30 of the instruction if X-form and to 0b00 if D-form
	17	Set to Bit 25 of the instruction if X-form and to Bit 5 if D-form
	18:21	Set to bits 21:24 of the instruction if X-form and to bits 1:4 if D-form
	22:31	Set to bits 6:15 of the instruction
Data Address Register (DAR)		Set to the effective address of the data access that caused the interrupt

Execution resumes at offset 0x00200 from the base address indicated by MSR_{IP}.

3.15.4.3 Data Storage Interrupt

A data storage interrupt is never generated by the hardware. The software may branch to this location as a result of implementation-specific data storage protection error interrupt.

3.15.4.4 Instruction Storage Interrupt

An instruction storage interrupt is never generated by the hardware. The software may branch to this location as a result of an implementation-specific instruction storage protection error interrupt.

3.15.4.5 Alignment Interrupt

An alignment exception occurs as a result of one of the following conditions:

- The operand of a floating-point load or store is not word aligned.
- The operand of load/store multiple is not word aligned.
- The operand of **lwarx** or **stwcx** is not word aligned.
- The operand of load/store individual scalar instruction is not naturally aligned when $MSR_{LE} = 1$.
- An attempt to execute multiple/string instruction is made when $MSR_{LE} = 1$.

3.15.4.6 Floating-Point Enabled Exception Type Program Interrupt

A floating-point enabled exception type program interrupt is generated if $((MSR_{FE0} \mid MSR_{FE1})\ \&FPSCR_{FEX})$ is set as a result of move to FPSCR instruction, move to MSR instruction or the execution of the **rfi** instruction. A floating-point enabled exception type program interrupt is not generated by floating-point arithmetic instructions. Instead if $((MSR_{FE0} \mid MSR_{FE1})\ \&FPSCR_{FEX})$ is set, the floating-point assist interrupt is generated.

3.15.4.7 Illegal Instruction Type Program Interrupt

An illegal instruction type program interrupt is not generated by the MPC565 / MPC566. An implementation dependent software emulation interrupt is generated instead.

3.15.4.8 Privileged Instruction Type Program interrupt

A privileged instruction type program interrupt is generated for an on-core valid SPR field or any SPR encoded as an external to the core special register if $SPR_0 = 1$ and $MSR_{PR} = 1$, as well as an attempt to execute privileged instruction when $MSR_{PR} = 1$.

3.15.4.9 Floating-Point Unavailable Interrupt

The floating-point unavailable interrupt is generated by the MPC565 / MPC566 core as defined in the **OEA**.

3.15.4.10 Trace Interrupt

A trace interrupt occurs if $MSR_{SE} = 1$ and any instruction except **rfi** is successfully completed or $MSR_{BE} = 1$ and a branch is completed. Notice that the trace interrupt does not occur after an instruction that caused an interrupt (for instance, **sc**). A monitor/debugger software must change the vectors of other possible interrupt addresses to single-step such instructions. If this is unacceptable, other debug features can be used. Refer to **SECTION 22 DEVELOPMENT SUPPORT** for more information. The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction following the executed instruction
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x00D00 from the base address indicated by MSR_{IP}.

3.15.4.11 Floating-Point Assist Interrupt

A floating-point assist interrupt occurs in the following cases:

- When a floating-point exception condition is detected, the corresponding floating-point enable bit in the FPSCR (floating-point status and control register) is set (exception enabled) and ((MSR_{FE0} | MSR_{FE1}) = 1). Note that when ((MSR_{FE0} | MSR_{FE1}) and $FPSCR_{FEX}$) is set as a result of move to FPSCR, move to MSR or **rfi**, the floating-point assist interrupt handler is not invoked.
- When an intermediate result is detected and the floating-point underflow exception is disabled ($FPSCR_{UE}$ = 0)
- In some cases when at least one of the source operands is denormalized.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR[1]
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

NOTES:

1. In the current implementation bit 30 of the SRR1 is never cleared other then by loading zero value from MSR RI.

Execution resumes at offset 0x00E00 from the base address indicated by MSR_{IP}.

3.15.4.12 Implementation-Dependent Software Emulation Interrupt

An implementation-dependent software emulation interrupt occurs in the following instances:

- When executing any non-implemented instruction. This includes all illegal and unimplemented optional instructions and all floating-point instructions.
- When executing a **mtspr** or **mfspr** that specifies on-core non-implemented register, regardless of SPR_0.
- When executing a **mtspr** or **mfspr** that specifies off-core non-implemented register and $SPR_0 = 0$ or $MSR_{PR} = 0$ (no program interrupt condition).
- Program interrupt is generated if ((MSR_{FE0} | MSR_{FE1}) and $FPSCR_{FEX}$) is set as a result of move to FPSCR instruction, move to MSR instruction, or the execution of the **rfi** instruction.
- Floating-point enabled exception type program interrupt is not generated by floating-point arithmetic instructions, instead if ((MSR_{FE0} | MSR_{FE1}) &$FPSCR_{FEX}$) is set, the floating-point assist interrupt is generated.

In addition, the following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x01000 from the base address indicated by MSR_{IP}.

3.15.4.13 Implementation-Specific Instruction Storage Protection Error Interrupt

The implementation-specific instruction storage protection error interrupt occurs in the following cases:

- The fetch access violates storage protection.
- The fetch access is to guarded storage and $MSR_{IR} = 1$.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1	Set to 0
	2	Set to 0
	3	Set to 1 if the fetch access was to a guarded storage when MSR_{IR} = 1, otherwise set to 0
	4	Set to 1 if the storage access is not permitted by the protection mechanism; otherwise set to 0
	10	Set to 0
	11:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Execution resumes at offset 0x01300 from the base address indicated by MSR_{IP}.

3.15.4.14 Implementation-Specific Data Storage Protection Error Interrupt

The implementation-specific data storage protection error interrupt occurs in the following case:

- The access violates the storage protection.

The following registers are set:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		Set to the effective address of the instruction that caused the interrupt
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0
Data/Storage Interrupt Status Register (DSISR)	0	Set to 0
	1	Set to 0
	2:3	Set to 0
	4	Set to 1 if the storage access is not permitted by the protection mechanism. Otherwise set to 0
	5	Set to 0
	6	Set to 1 for a store operation and to 0 for a load operation
	7:31	Set to 0
Data Address Register (DAR)		Set to the effective address of the data access that caused the interrupt

Execution resumes at offset 0x01400 from the base address indicated by MSR_{IP}.

3.15.4.15 Implementation-Specific Debug Interrupts

Implementation-specific debug interrupts occur in the following cases:

- When there is an internal breakpoint match (for more details, refer to **SECTION 22 DEVELOPMENT SUPPORT**.
- When a peripheral breakpoint request is asserted to the MPC565 / MPC566 core.
- When the development port request is asserted to the MPC565 / MPC566 core. Refer to **SECTION 22 DEVELOPMENT SUPPORT** for details on how to generate the development port-interrupt request.

The following registers are set:

For L-bus breakpoint instances, these registers are set to:

Register Name	Bits	Description
Save/Restore Register 0 (SRR0)		For I-breakpoints, set to the effective address of the instruction that caused the interrupt. For L-breakpoint, set to the effective address of the instruction following the instruction that caused the interrupt. For development port maskable request or a peripheral breakpoint, set to the effective address of the instruction that the processor would have executed next if no interrupt conditions were present. If the development port request is asserted at reset, the value of SRR0 is undefined.
Save/Restore Register 1 (SRR1)	1:4	Set to 0
	10:15	Set to 0
	Other	Loaded from bits 16:31 of MSR. In the current implementation, Bit 30 of the SRR1 is never cleared, except by loading a zero value from MSR_{RI}. If the development port request is asserted at reset, the value of SRR1 is undefined.
Machine State Register (MSR)	IP	No change
	ME	No change
	LE	Bit is copied from ILE
	Other	Set to 0

Register Name	Bits	Description
BAR		Set to the effective address of the data access as computed by the instruction that caused the interrupt
DAR and DSISR		Do not change

Execution resumes at offset from the base address indicated by MSR_{IP} as follows:

- 0x01D00 – For instruction breakpoint match
- 0x01C00 – For data breakpoint match
- 0x01E00 – For development port maskable request or a peripheral breakpoint
- 0x01F00 – For development port non-maskable request

3.15.4.16 Partially Executed Instructions

In general, the architecture permits instructions to be partially executed when an alignment or data storage interrupt occurs. In the core, instructions are not executed at all if an alignment interrupt condition is detected and data storage interrupt is never generated by the hardware. In the MPC565 / MPC566, the instruction can be partially executed only in the case of the load/store instructions that cause multiple access to the memory subsystem. These instructions are:

- Multiple/string instructions
- Unaligned load/store instructions

In the last case, the store instruction can be partially completed if one of the accesses (except the first one) causes the data storage protection error. The implementation-specific data storage protection interrupt is taken in this case. For the update forms, the update register (RA) is not altered.

3.15.5 Timer Facilities

Descriptions of the timebase and decrementer registers can be found in **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** and in **SECTION 8 CLOCKS AND POWER CONTROL**.

3.15.6 Optional Facilities and Instructions

Any other **OEA** optional facilities and instructions (except those that are discussed here) are not implemented by the MPC565 / MPC566 hardware. Attempting to execute any of these instructions causes an implementation dependent software emulation interrupt to be taken.

SECTION 4
BURST BUFFER CONTROLLER MODULE

4.1 Overview and General Description

The burst buffer controller module (BBC) consists of three main functional parts: bus interface unit (BIU), instruction memory protection unit (IMPU) and instruction code decompressor unit (ICDU) See **Figure 4-1**.

The BBC master BIU interfaces between the RCPU instruction port and internal U-bus and can support burstable and non-burstable accesses on U-bus.

The IMPU allows the instruction memory to be divided into four regions with different protection attributes. The IMPU compares the attributes of the RCPU memory access request with the attributes of the appropriate region. If the access is allowed, the proper signals are sent to the BIU. If the memory region is protected, an interrupt is sent to the RCPU and master BIU cancels U-bus access.

The IMPU is able to relocate the RCPU exception vectors. The IMPU always maps the exception vectors into the internal memory space of the MPC565 / MPC566. This feature is important for multi-MPC565 / MPC566 system, where although the internal memories of some of controllers are shifted not to be on the lower four Mbytes, they can still have their own internal exception vector tables with the same exception addresses issued by their RCPU cores.

The IMPU also supports MPC565 / MPC566 enhanced interrupt controller by extending exception vectors relocation mechanism to translate the RCPU External Interrupt exception vector separately and splitting it to up to 48 different vectors, corresponding to the code, generated by the interrupt controller. See also **6.4.4 Enhanced Interrupt Controller**

ICDU is responsible for on-line (previously compressed) instruction code decompression in the “Decompression ON” mode.

NOTE

The code compression features of the MPC566 are slightly different than the code compression of the MPC556.

The ICDU contains a four-Kbyte RAM (DECRAM). DECRAM is used for decompressor vocabulary tables storage of the general-purpose memory.

The DECRAM can also serve as general purpose RAM memory on U-bus.

The ICDU includes special branch target buffer (BTB) that improves theMPC565 performance by holding previously fetch instructions to RCPU core.

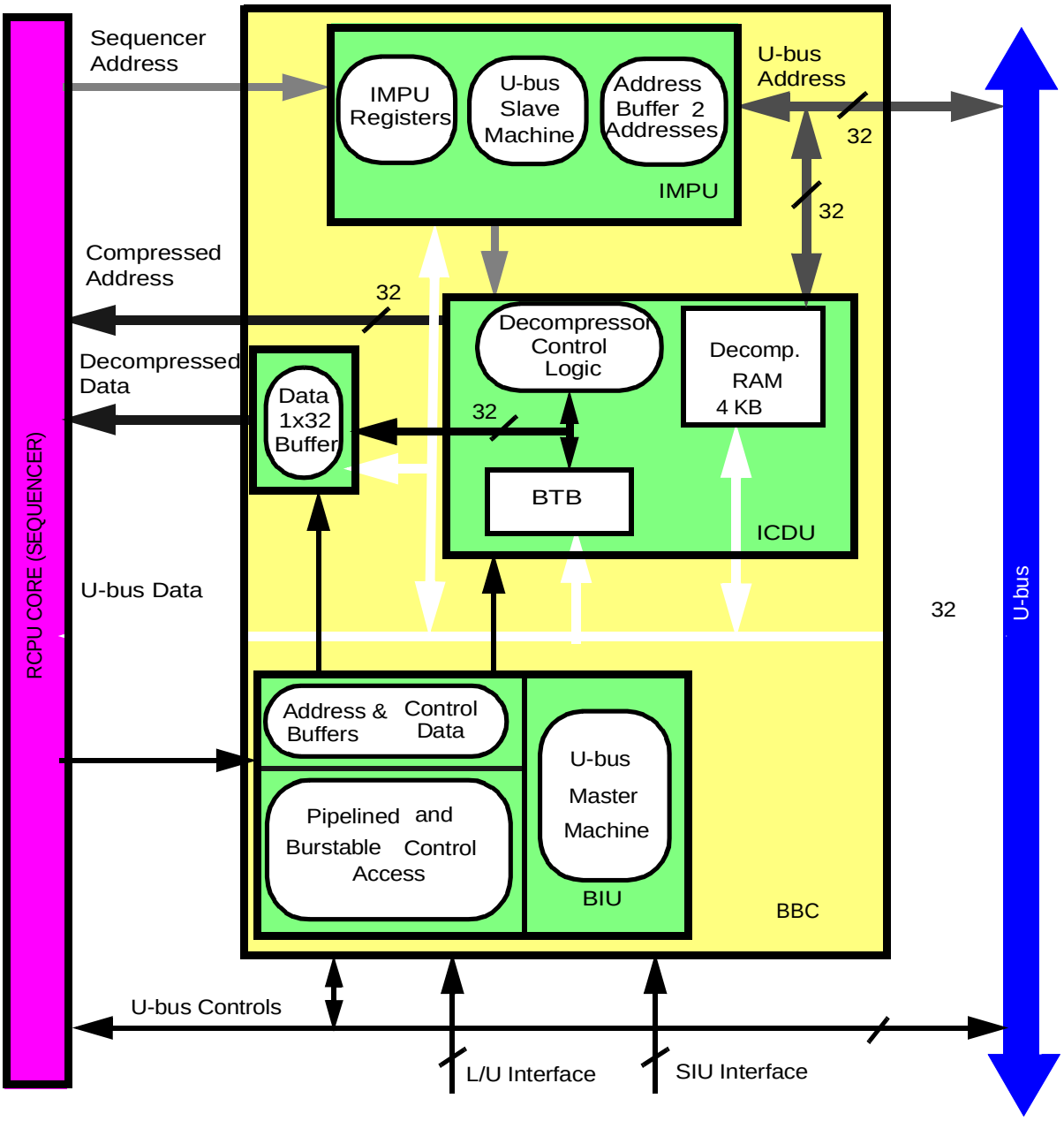


Figure 4-1 BBC Module Block Diagram

4.2 Feature List

4.2.1 BIU Key Features

- Supports pipelined and burstable accesses to internal and external memories
- Supports the de-coupled interface with the RCPU instruction unit
- Parked master on the U-bus, resulting with zero clocks delay for RCPU fetch access to the U-bus

- Full utilization of the U-bus pipeline for fetch accesses
- Tight interface with L2U module, taking advantage of full U-bus bandwidth and back-to-back accesses avoiding undesirable bubbles
- Supports program trace and show cycles

4.2.2 IMPU Key Features

- Four regions in which the base address and size can be programmed
- Available region sizes are: four Kbytes, eight Kbytes, 16 Kbytes, 32 Kbytes, 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, one Mbyte, two Mbytes, four Mbytes, eight Mbytes, 16 Mbytes....four Gbytes
- Overlap between regions is allowed
- Each of the four regions supports the following attributes:
  - — user/supervisor
  - — Guard attribute (causes an interrupt in case of speculative fetch attempt)
  - — compressed/non-compressed
- can be enabled or disabled by software
- Global region entry declares the default access attributes for all memory areas not covered by the four regions:
- RCPU gets the instruction storage protection exception generated upon:
  - — Access violation of protection attributes
  - — Fetch from guarded region
- RCPU **MSR[IR]** bit controls IMPU protection
- Programming is done using PowerPC's **mtspr/mfspr** instructions to/from implementation specific special purpose registers.
- Compressed/non-compressed region with enable/disable option
- Exception table relocation: the IMPU supplies relocation address of all exceptions within the internal memory space
- External interrupt vector splitting to reduce the external interrupt latency
- Special reset exception vector for "Decompression ON" mode

4.2.3 ICDU Key Features

- Instruction code on-line decompression based on "instruction classes" algorithm.
- No need for address translation between compressed and non-compressed address spaces — ICDU provides "next instruction address" to the RCPU
- Instruction decompression takes one clock mostly
- Code decompression is pipelined:
  - — No performance penalty during sequential program flow execution
  - — Minimal performance penalty due to change of program flow execution
- Two operation modes are available: "Decompression ON" and "Decompression

OFF". Switch between compressed and non-compressed user application software parts is possible.

- adaptive vocabularies scheme is supported; each user application can have its own optimum vocabularies.
- Special branch target buffer (BTB) to improve system performance

4.2.3.1 DECRAM Key Features

- Four Kbytes RAM for decompression vocabulary tables
- Two clock read/write access U-bus general purpose RAM
- Four clock load/store access from the L-bus
- Byte, half-word (16-bit) or word (32-bit) read/write accesses and fetches
- Special access protection functions
- Low power standby operation for data retention

4.3 Class Based Compression Model Main Principles

4.3.1 Compression Model Features

- Implemented for PowerPC architecture
- Up to 50% code size reduction
- No need for address translation tables
- No changes in the CPU architecture
- Compression is done off line by a special "compressor" tool, using Instruction classes based algorithm optimized for PowerPC instructions set
- Decompression is done at run-time by special hardware
- Optimized for cache-less systems:
  - — Highly effective in system solutions for low cache-hit ratio environment and for systems with fast embedded program memory
  - — Deterministic program execution time
  - — No performance penalty during sequential program flow execution
  - — Minimal performance penalty due to change of program flow execution
- Switch between compressed and non-compressed user application sections is possible. (Compressed subroutine can call non-compressed one and be called from non-compressed portion of user application)
- Adaptive vocabularies, generated for particular application
- Slight changes in the core and existing RISC development tools — compilers, simulators, manually coded libraries
- Compressed address space is up to one Gbyte
- Branch displacement from its target:

— Conditional branch displacement is up to four Kbytes
— Unconditional branch displacement is up to four Mbytes

NOTE

Branch displacement is hardware limited. The compiler can enlarge the branch scope by creating branch chains.

4.3.2 Model Limitations

No address arithmetic is allowed, because the address map changes during compression and no software tool can identify address arithmetic structures in the code.

4.3.3 Instruction Classes Based Compression Algorithm

The code compression algorithm is based on creating vocabularies of frequently appearing PowerPC RISC instructions or instructions halves and replacing these instructions with pointers to the vocabularies. The system contains several sets of vocabularies for different groups of instructions. These groups are referred to as *classes*.

Every instruction belongs to exactly one class. Compression of the instructions in a class may be in one of the following modes. Refer to **Figure 4-2**.

1. Compression of the whole instruction into one vocabulary pointer
2. Compression of each half of the instruction into a different vocabulary
3. Compression of one of the instruction's halves into a vocabulary pointer and bypass of the other half. Bypass is the placing of the field's data in the compressed code, sometimes after compaction defined in the class.
4. Bypass of the whole instruction. No compaction permitted.

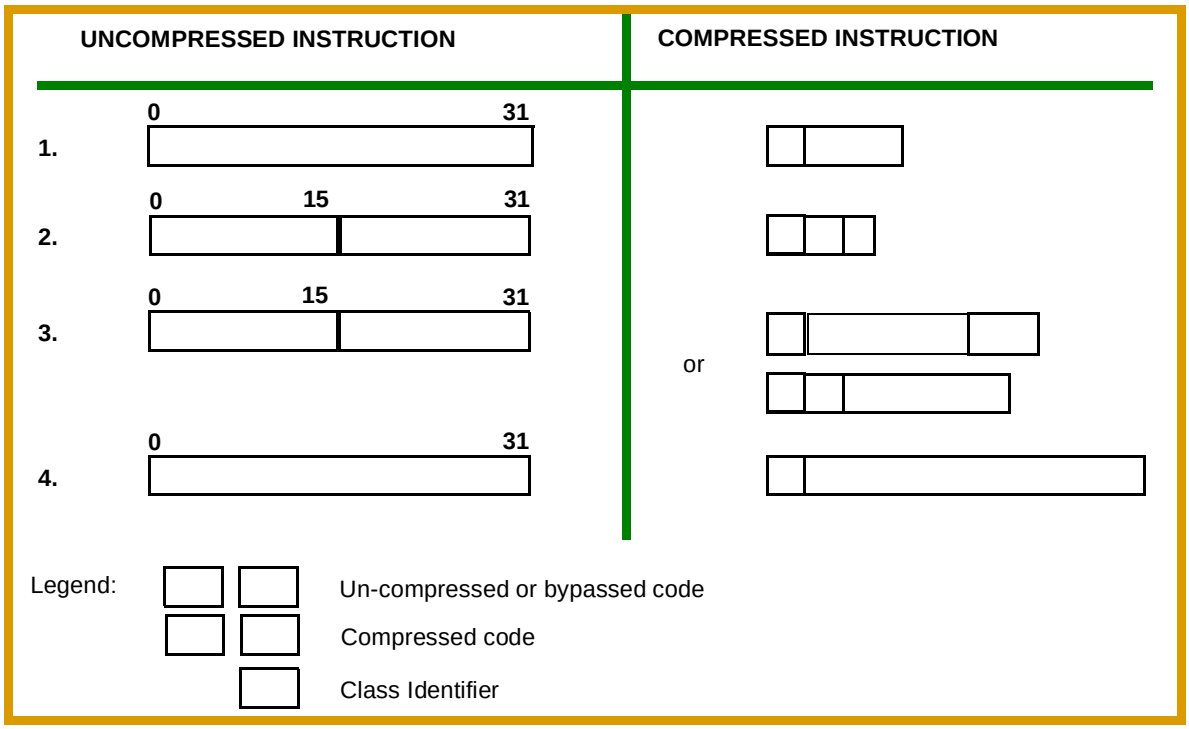


Figure 4-2 Instruction Compression Alternatives

A 4-bit class identifier is added to the head of each compressed instruction in order to enable class identification during decompression. Compressed and bypass fields' length may vary.

The compressed instruction is guaranteed to be of even length of bits. Thus, four bits are needed to locate such instruction inside a memory word. The instruction address in "Decompression ON" mode consists of a 28-bit word address and a 4-bit instruction pointer (IP). See **Figure 4-3**.

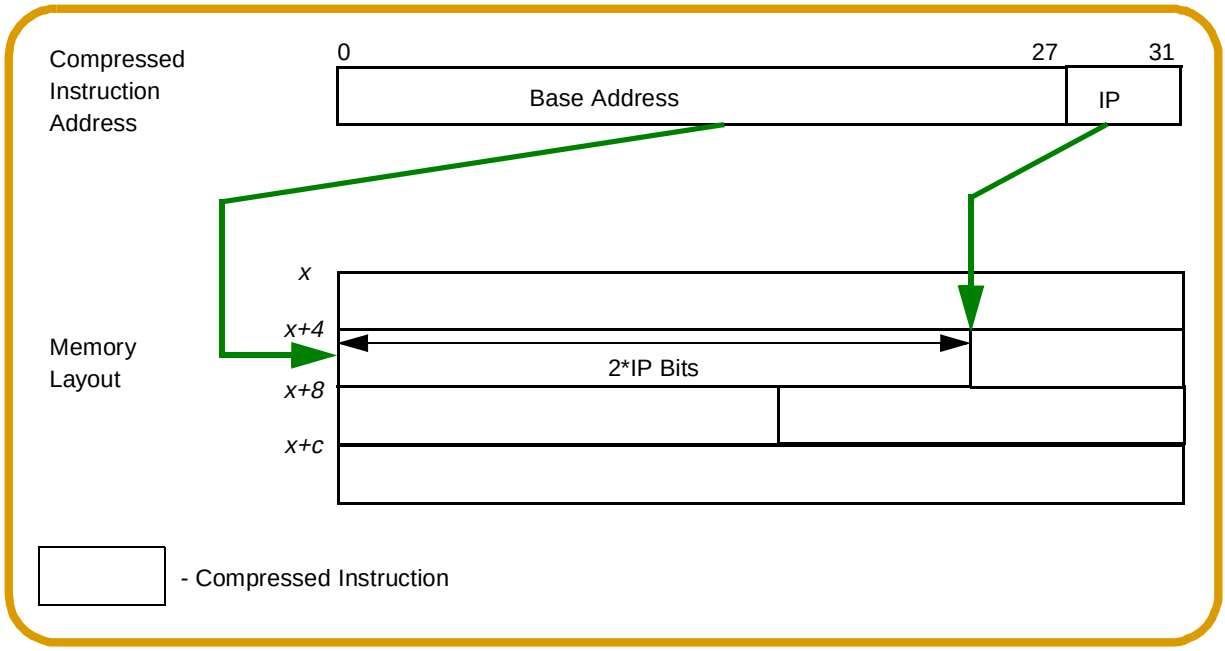


Figure 4-3 Compressed Address Format

4.3.4 Compression Process

Compression process is implemented by the following steps: (See **Figure 4-4**.)

- User code compilation/linking
- Vocabulary and classes generation
- User application code compression by software compression tool.

The vocabularies and classes configuration are generated by profiling the static code, based on instruction classes algorithm.

The code compression can be done by using either default or specific for particular application vocabularies, generated at previous step. In case of default vocabularies, the previous step can be omitted, but compression efficiency is reduced.

Compression tool replaces regular PowerPC instruction by their “compressed” representation containing less data bits. The tool also updates offset fields in direct branch instructions to include compressed formatted offset (four bits IP and word offset). Thus maximum branch offsets in “Decompression On” mode are reduced. The RCPU uses the word offset for direct branch target address computation. The RCPU provides IP portion of the branch offset field to the decompression unit as it is represented in the branch instruction.

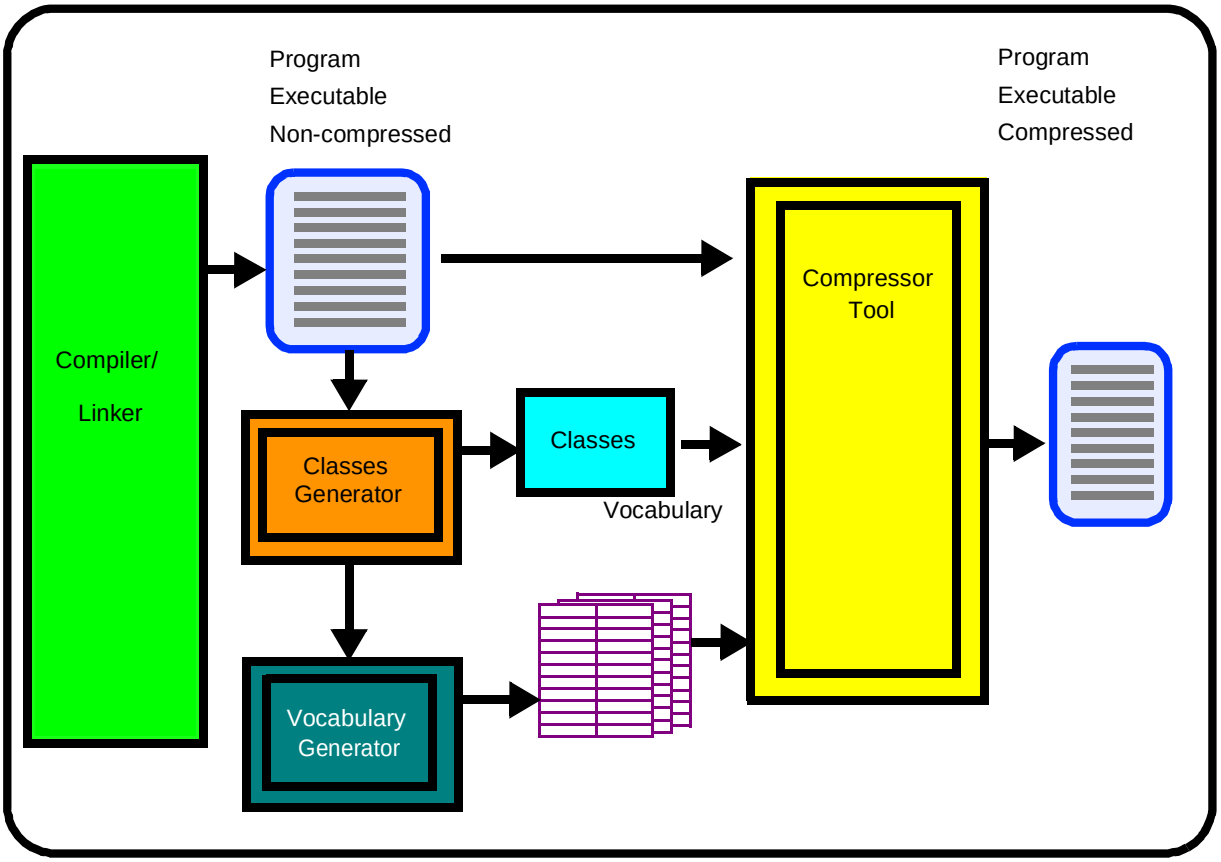


Figure 4-4 Code Compression Process

4.3.5 Decompression

- The instruction code is stored in the memory in the compressed form
- The vocabularies are stored in a dedicated ICDU RAM (DECRAM)
- The classes configuration is stored in a dedicated ICDU registers (DCCRs)
- The decompression is done on-line by the dedicated decompressor unit
- Decompression flow: (See **Figure 4-5**)
  - — RCPU provides to the BBC **"2-bit aligned COF[1] address"**
  - — ICDU:
    - Converts COF address to **"word aligned physical address"** to access the memory
    - Fetches the compressed instruction code data from the memory, decompresses it and delivers "**non-compressed instruction code**" together with the bit aligned "**next instruction address**" to the RCPU, which uses it for subroutine and exceptions handling.

[1] COF = Change of Flow

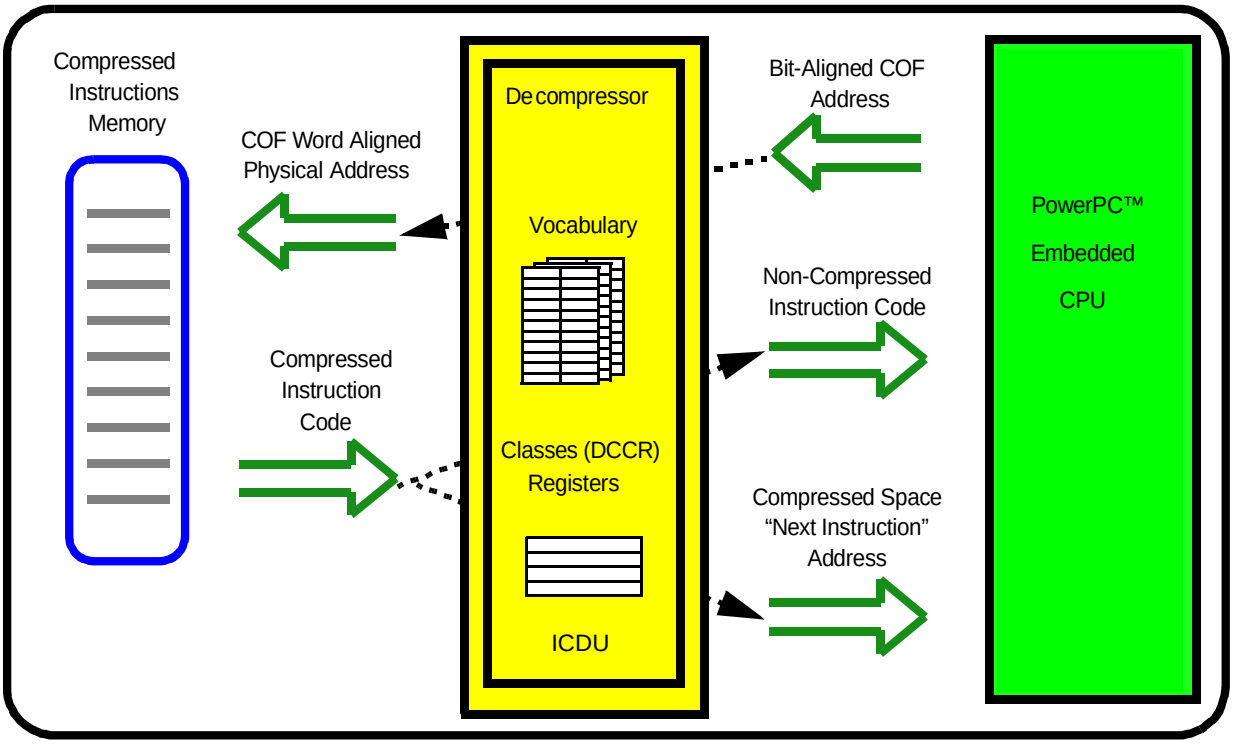


Figure 4-5 Code Decompression Process

4.3.6 Compression Environment Initialization

In order to commence execution of compressed code, the DECRAM and the classes information (in DCCR registers) should be programmed. The data to be programmed is supplied by the compressor tool and the vocabulary generator. There are two initialization scenarios:

1. Wake up in "Decompression OFF" mode — If the chip wakes up with decompression disabled, the initialization routine can be executed at any time before entering "Decompression ON" mode. After the compression environment is initialized, the operational mode would be changed to "Decompression ON".
2. Wake up in "Decompression ON" mode — If the chip wakes up in "Decompression ON" mode, it has to perform compressed instruction without the vocabularies and class parameters. Thus, all instructions executed until the end of the initialization routine will be compressed in the global bypass format. Perform the routine as close as possible to the wake up.

4.4 Operation Modes

4.4.1 Instruction Fetch

BBC provides two instruction fetch modes: "Decompression OFF" and "Decompression ON". The operational mode is defined by the RCPU **MSR[DCMPEN]** bit. If the bit

is set, the mode is “Decompression ON”. If it is not set, the mode is “Decompression OFF”.

4.4.1.1 “Decompression OFF” Mode

In this mode the BIU module transfers fetch accesses from the RCPU to the U-bus. When a new access is issued by the RCPU, it is transferred in parallel to both IMPU and BIU. The IMPU compares the address of the access to its region programming. The BIU checks if the access can be immediately transferred to the U-bus, otherwise it requests the U-bus for the next clock.

The BIU may be programmed for burstable or non-burstable access. If the BIU is programmed for burstable access, U-bus address phase transaction is accompanied by the burst request attribute. If burstable access is allowed by the U-bus slave, the BIU continues current access as burstable, otherwise current access is executed as a single access. If any protection violation is detected by the IMPU, the current U-bus access is aborted by the BIU and exception is signaled to the RCPU.

Show cycle, program trace and debug port access attributes accompanying the RCPU access are forwarded by the BIU along with the U-bus access.

4.4.1.2 “Decompression ON” Mode

In this mode the RCPU sends the two-bit aligned change of flow address to the BBC. The BIU transfers word portion of the address to the U-bus. The BBC continues to prefetch the data from the consequent memory addresses regardless RCPU requests in order to supply data to the ICDU.

The data coming from the instruction memory is not provided directly to the RCPU, but loaded into the ICDU for decompression. Decompressed instruction code together with “next instruction address” are provided to the RCPU whenever it requires another instruction fetch.

All addresses issued by the BIU to the U-bus are transferred in parallel to the IMPU. The IMPU compares the address of the access to its region programming. If any protection violation is detected by the IMPU, the current U-bus access is aborted by the BIU and an exception is signaled to the RCPU.

Show cycle and program trace access attributes accompanying the COF RCPU access only are forwarded by the BIU along with the U-bus access. Additional information about “IP” part of compressed instruction address is provided on U-bus data bus. Refer to **4.4.1.3 Show Cycles in “Decompression ON” Mode** for more details.

In this mode the ICDU DECRAM is used as decompressor vocabulary storage and may not be used as a general purpose RAM.

4.4.1.3 Show Cycles in “Decompression ON” Mode

In the case of “Decompression ON” mode the instruction address consists of instruction “base address” and four bits of the “instruction bit pointer”. In order to provide the capability to show full instruction address, including instruction bit pointer, on the exter-

nal bus, show cycles information is presented not only on the address bus, but also on some bits of the data bus:

- **ADDR[0:29]** - show the value of the base address of compressed instruction (word pointer into the memory)
- **DATA[0]** - shows in which mode the MPC565 / MPC566 is operating

  0 = “Decompression OFF” mode.
  1 = “Decompression ON” mode.
- **DATA[1:4]** - represent an instruction bit pointer within the word.

NOTE

The **BBCMCR[DECOMP_SC_EN]** bit determines if the data portion (**DATA[0:4]**) of the instruction show cycle is driven or not, regardless of decompression mode (**BBCMCR[EN_COMP]** bit)

If the **BBCMCR[DECOMP_SC_EN]** bit is set the show cycle may be delayed by one clock by the USIU. This happens if the show cycle occurs after external device read cycle.

4.4.2 Burst Operation of the BBC

The BBC may initiate and handle burst accesses on the U-bus. The **BBCMCR[BE]** bit determines whether the BBC operates burst cycles or not. Burst requests are enabled when the BE bit is set. BBC handles non wrap-around bursts with up to 4 data beats on internal U-bus. Refer to **4.8.2.1 BBC Module Configuration Register BBCMCR**.

NOTES

The burst operation in the MPC565 / MPC566 is useful if a user system implements burstable memory devices on external bus. Otherwise the mode will cause the MPC565 / MPC566 performance degradation.

When the RCPU runs in serialized mode it’s recommended to disable bursts by the BBC to speed up MPC565 / MPC566 operation

Burst operation in “Decompression ON” and “Debug” mode is disabled regardless of **BBCMCR[BE]** bit setting.

4.4.3 Access Violation Detection

Instruction memory protection is assigned on a regional basis. Default operation of IMPU is done on a global region. The IMPU has control registers which contain the following information: region protection on/off, region base address, size and access permissions.

Protection logic is activated only if RCPU **MSR[IR]** is set.

During each fetch request from the RCPU core to instruction memory, the address is compared to value in region base address of enabled regions. Any address, matching

the specific region within its appropriate size as defined in the region attribute register, sets a match indication.

When more than one match indication occurs, the effective region is the region with the highest priority. Priority is determined by region number. The lowest region number has highest priority. The global region has lowest priority.

When no match happens, the effective region is the global region.

The region attribute registers contain the region protection fields (**PP, G, CMPR**). The protection fields are compared to address attributes issued by the RCPU. If the access is permitted the address is passed to BIU and further to U-bus.

Whenever IMPU detects access violation, the following actions are taken:

1. The request, forwarded to the BIU is canceled
2. The RCPU is informed that the requested address caused an access violation by exception request.

However, if the required address contains show cycle attribute, the BIU delivers the access onto the U-bus to obtain program tracking

The exception vector (address) that the RCPU issues for this exception has a 0x1300 offset in PowerPC exception vector table. The access violation status is provided in RCPU **SRR1** special purpose register. The encoding of the status bits is as follows:

- **SRR1 [1]** = 0
- **SRR1 [3]** = Guarded storage
- **SRR1 [4]** = Protected storage or Compression violation
- **SRR1 [10]** = 0

Only one bit is set at a time.

4.4.4 Slave Operation

The BBC is operating as a U-bus slave when the IMPU registers, decompressor RAM (DECRAM) or ICDU registers are accessed from the U-bus. The IMPU registers programming is done using PowerPC **mtspr/mfspr** instructions. The ICDU configuration registers (DCCRs) (see **Table 4-10**) and DECRAM are mapped into the chip memory space and accessed by PowerPC load/store instructions. DCCR and DECRAM accesses may be disabled by **BBCMCR[DCAE]** bit. Refer to **4.8.2.1 BBC Module Configuration Register BBCMCR**.

4.4.5 Reset Behavior

Upon “soft” reset the BBC switches to an idle state and all pending U-bus accesses are ignored, ICDU internal queue is flushed and IMPU switches to a disabled state where all memory space is accessible for both user and supervisor.

“Hard” reset sets some of the fields and bits in the BBC configuration registers to their default reset state. Some bits in **BBCMCR** register get their values from the Reset Configuration Word.

4.4.6 Debug Operation Mode

When the MPC565 / MPC566 RCPU core is under debug (in debug mode) the BBC initiates not-burstable access to the debug port and ICDU is bypassed, (i.e., instructions, transmitted to debug port must be non-compressed, regardless of operational mode).

4.5 Exception Table Relocation (ETR)

4.5.1 ETR Overview

The BBC is able to relocate the exception addresses of the RCPU. The relocation feature always maps the exception addresses into the internal memory space of the MPC565 / MPC566 (See **Figure 4-6**). This feature is important in multi MPC565 / MPC566 system, where although the memory map of some of them was shifted not to be on the lower 4 MB, their RCPU cores can still access their own exception handlers in their internal flash in spite of RCPU issues the same exception addresses.

The relocation also saves the wasted space between the exception table entries in the case where each exception entry contained only a branch instruction to the exception routine, which is located elsewhere.

Exception vector table may be programmed to be located in four places in MPC565 / MPC566 internal memory space.

The exception table relocation is supported in both operation modes “Decompression ON” and “Decompression OFF”.

The RESET routine vector is relocated differently in “Decompression ON” and in “Decompression OFF” modes. This feature may be used by software code compression tool to guarantee that a vocabulary table initialization routine always executed before application code running.

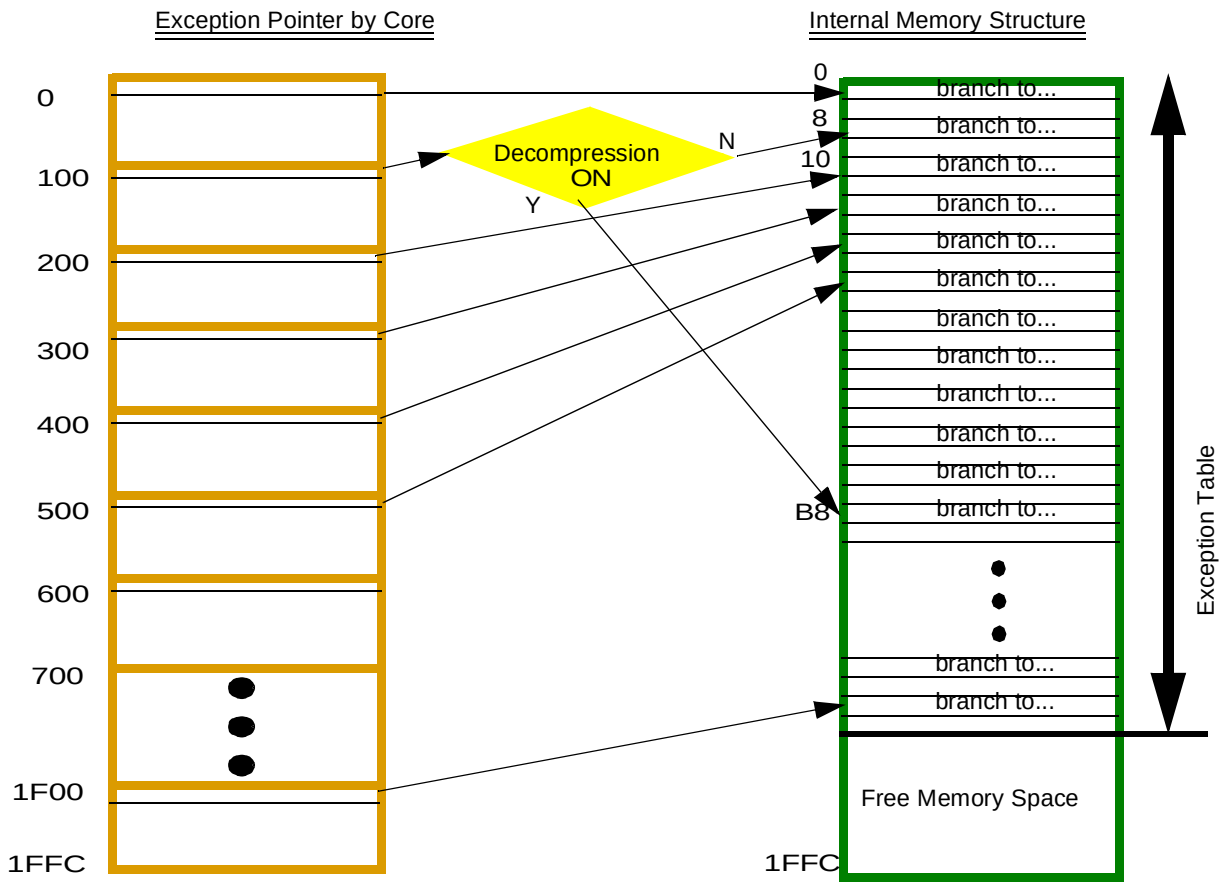


Figure 4-6 Exception Table Entries Mapping

4.5.2 ETR Operation

The exception vectors are 0x100 bytes apart from each other, starting at address 0x0000 0100 or 0xFFF0 0100, depending on the value of **MSR[IP]** bit in the RCPU.

If the ETR is disabled, the BBC transfers the exception fetch address to the U-bus of the MPC565 / MPC566 with no interference. In this case, normal PowerPC exception addressing is implementing.

If the relocation feature is enabled, the BBC translates the exception vector into the exception relocation address. At that location, a branch instruction with absolute addressing must be placed. Each instruction branches to the required exception routine. These branch instructions should be successive in that region of the memory; that way, a table of branch instructions is implemented. Executing the branch instruction causes the core to branch twice until it gets to the exception routine.

Each exception relocation table entry occupies two words to support “Decompression ON” mode, where a branch instruction can be more than 32 bits long.

NOTES

The eight Kbytes allocated for the original PowerPC exception table can be almost fully utilized. This is possible if the MPC565 / MPC566 system memory is NOT mapped to the exception address space, (i.e., the addresses 0xFFF0_0000 to 0xFFF0_1FFF are not used).

In such case, these eight Kbytes can fully be utilized by the compiler, except for the lower 64 words (256 bytes) which are dedicated for the branch instructions.

If the RCPU, while executing an exception, issues any address between two successive exception entries (e.g., 0xFFF0_0104), then the operation of the MPC565 / MPC566 is not guaranteed if the ETR is enabled.

In order to activate the exception table relocation feature, the following steps are required:

1. Set the RCPU **MSR[IP]** bit
2. Set the **BBCMCR[ETRE]** bit. See **4.8.2.1 BBC Module Configuration Register BBCMCR** for programming details.
3. Program the **BBCMCR[OERC]** bits to determine the exception branch table location in the memory, according the description in **Table 4-2**.

The ETR feature can be activated from reset, by setting corresponded bits in the reset configuration word.

Table 4-1 Exception Addresses Mapping

Name of Exception	Original Address Issues by Core	Mapped Address by Exception Table Relocation Logic	
System Reset	0xFFF0_0100	EN_COMP= 0	EN_COMP= 1
		Page_Offset+0x08[1]	Page_Offset+0x0B8
Machine Check	0xFFF0 0200	Page_Offset+0x010	
Data Storage	0xFFF0 0300	Page_Offset+0x018	
Alignment	0xFFF0 0600	Page_Offset+0x030	
Program	0xFFF0 0700	Page_Offset+0x038	
Floating Point unavailable	0xFFF0 0800	Page_Offset+0x040	
Decrementer	0xFFF0 0900	Page_Offset+0x048	
System Call	0xFFF0 0C00	Page_Offset+0x060	
Trace	0xFFF0 0D00	Page_Offset+0x068	
Floating Point Assist	0xFFF0 0E00	Page_Offset+0x070	
Implementation Dependent Software Emulation	0xFFF0 1000	Page_Offset+0x080	
Implementation Dependent Instruction Storage Protection Error	0xFFF0 1300	Page_Offset+0x098	
Implementation Dependent Data Storage Protection Error	0xFFF0 1400	Page_Offset+0x0A0	
Implementation Dependent Data Breakpoint	0xFFF0 1C00	Page_Offset+0x0E0	
Implementation Dependent Instruction Breakpoint	0x0FFF 1D00	Page_Offset+0x0E8	
Implementation Dependent Maskable External Breakpoint	0xFFF0 1E00	Page_Offset+0x0F0	
Non-Maskable External Breakpoint	0xFFF0 1F00	Page_Offset+0x0F8	

NOTES:
1. Refer to **Table 4-2**.

Table 4-2 Exception Relocation Page Offset

BBCMCR(OERC[0:1])		Page Offset	Comments
0	0	0x0 + ISB offset[1]	0
0	1	0x1 0000 + ISB offset	64 Kbytes[2]
1	0	0x8 0000 + ISB offset	512 Kbytes
1	1	0x3F E000 + ISB offset	L-bus (CALRAM) Address

NOTES:
1. ISB offset is equal 4M * ISB (0x400000 * ISB), where ISB is value of bit field in USIU IMMR register.
2. Note that this offset was 32 Kbytes on the MPC555 due to the smaller flash block size (32 Kbytes) than the MPC565 (64 Kbytes).

4.5.3 Enhanced External Interrupt Relocation (EEIR)

The BBC also supports the enhanced external interrupt model of the MPC565 / MPC566 which allows the removal of the interrupt requesting source detection stage from the interrupt routine. When the RCPU receives an external interrupt, it provides its' sole PowerPC external interrupt vector. BBC logic detects this address and replaces it with another address corresponding to the module which initiated the interrupt. See **Figure 4-7**. Precise information about the interrupt source module is provided by the interrupt controller of USIU.

The external interrupt relocation table should start at physical address defined in the external interrupt relocation table base address register. See **4.8.2.5 External Interrupt Relocation Table Base Address Register — EIBADR**.

Each table entry must contain a branch absolute (**ba**) instruction to the first instruction of interrupt service routine. Each table entry occupies two words (eight bytes) to support “Decompression ON” mode, where branch instruction can be more than 32 bits long.

The memory space allocated for the external interrupt relocation table is up to two Kbytes. If part of the external interrupt relocation table entries are not used, it may be utilized for another purpose as either instruction code or data.

In order to activate the external interrupt relocation feature, the following steps are required:

1. Program the **EIBADR** register to the external interrupt branch table base address. See **4.8.2.5 External Interrupt Relocation Table Base Address Register — EIBADR**
2. Set the **MSR[IP]** bit
3. Set the **BBCMCR[EIR]** bit. See **4.8.2.1 BBC Module Configuration Register BBCMCR** for programming details.
4. Set **SIUMCR[EIC]** bit. See **6.4.4 Enhanced Interrupt Controller**.

NOTES

If both the enhanced external interrupt relocation and exception table relocation functions are activated simultaneously, the final external interrupt vector is defined by EEIR mechanism.

When the EEIR function is activated, any branch instruction execution with the 0xFFF0_0500 target address may cause unpredictable program execution.

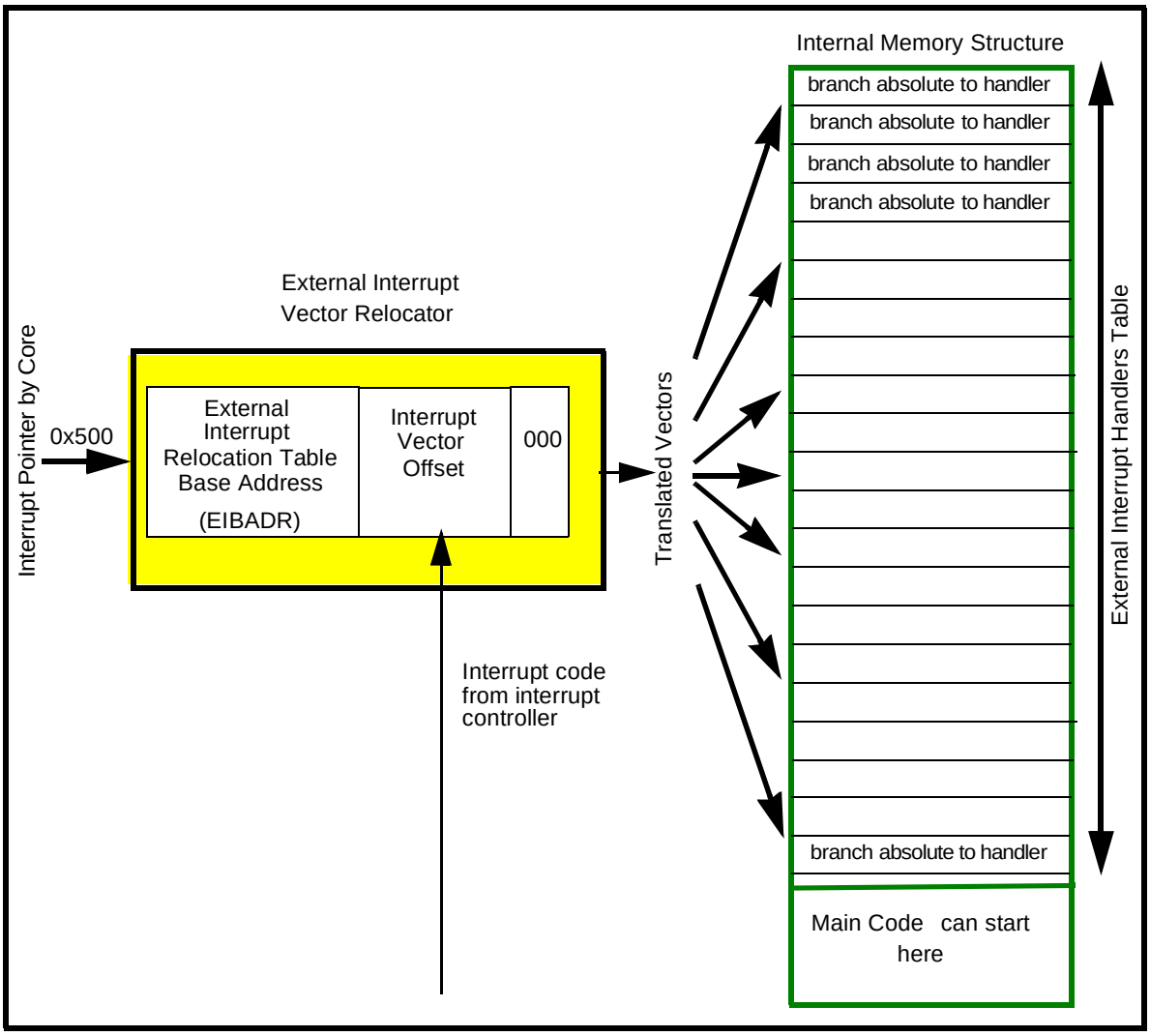


Figure 4-7 External Interrupt Vectors Splitting

4.6 Decompressor RAM (DECRAM) Functionality

Decompressor RAM (DECRAM) is a part of the ICDU. It occupies a 4-Kbyte physical RAM array block. It is mapped both in ICDU internal address space and chip memory address space. It is a single port memory and may not be accessed simultaneously from the ICDU and U-bus.

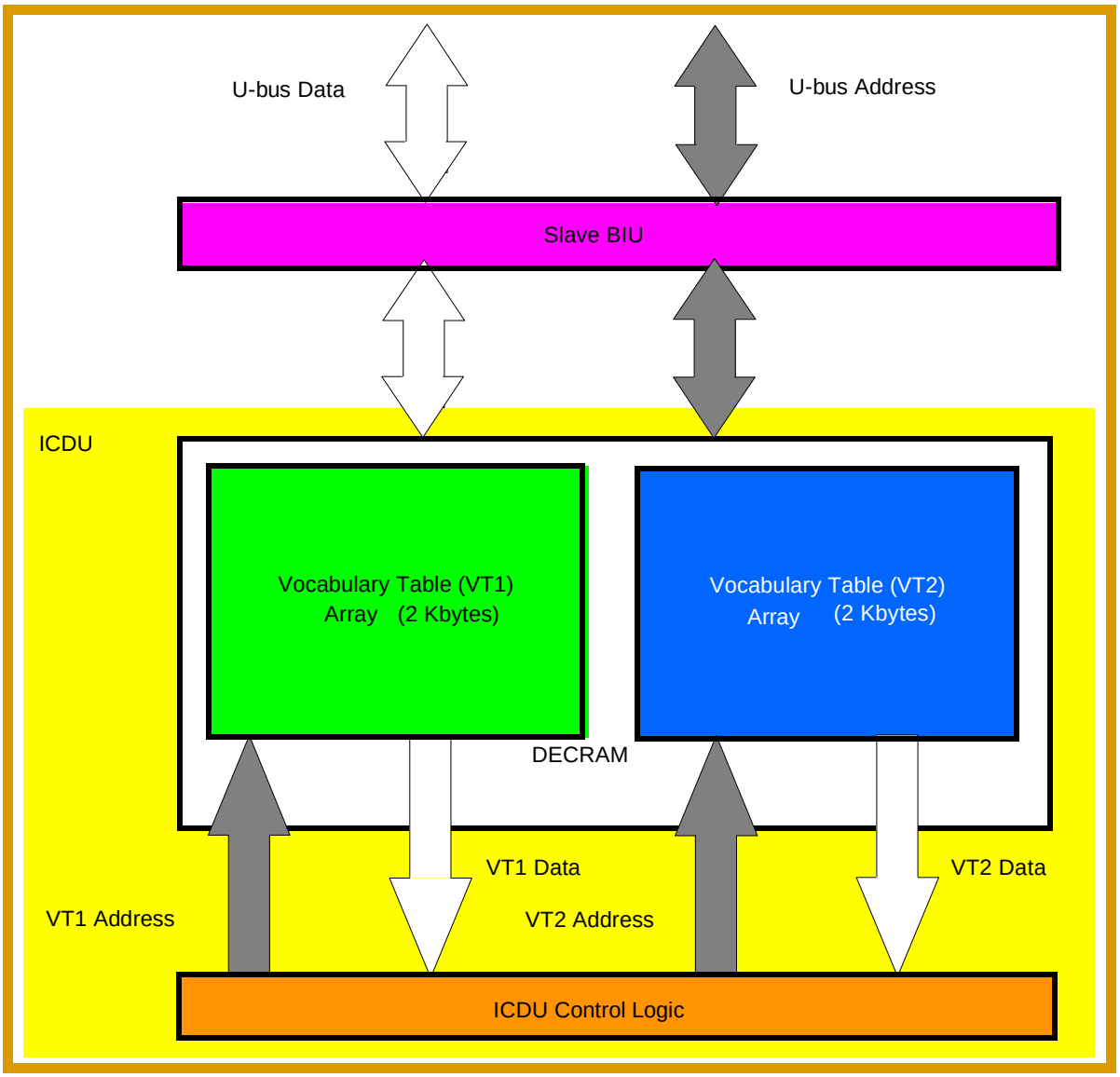


Figure 4-8 DECRAM Interfaces Block Diagram

4.6.1 Vocabulary Table Storage Operation

DECRAM is used for decompressor vocabulary tables (VT1 and VT2) storage in "Decompression ON" mode. The ICDU utilizes DECRAM as two separately accessed 2-Kbyte RAM arrays (16 bits wide) which are accessed via internal ICDU buses. The VTs should be loaded before the decompression process starts. In order to allow decompression, the DECRAM must be disabled for U-bus accesses after VTs and decompressor class configuration registers (DCCRs) are initialized.

4.6.2 General Purpose Memory Operation

In the case when "Decompression ON" mode is not activated in the chip, the DECRAM can serve as a two-clock access general purpose RAM for U-bus instruction fetches

or read/write data operations. The base address of the DECRAM is 0x2F 8000. The proper access rights to the DECRAM array may be defined by programming the R, D, and S bits of **BBCMCR** register:

- Read/write or read only
- Instruction/data or data only
- Supervisor/user or supervisor only

External access mode of the RAM is activated by the **BBCMCR[DCAE]** bit setting (see **4.8.2.1 BBC Module Configuration Register BBCMCR**). In this mode the DECRAM can be accessed from U-bus and cannot be accessed by the ICDU logic.

In this mode:

- The DECRAM supports word, half-word and byte operations.
- The DECRAM is emulated to be 32 bit wide. For example: load access from offset 0 in the DECRAM will deliver the concatenation of the first word in each of the DECRAM banks when RAM 1 contains the 16 LSB of the word and RAM 2 contains the 16 MSB.
- Load accesses at any width are supplied with 32 bits of valid data.
- The DECRAM can be accessed while the MPC565 / MPC566 is in “Decompression ON” mode, but all fetched instructions should be in the global bypass format.
- The DECRAM communicates with the U-bus pipeline but does not support pipelined accesses to itself. If a store operation is second in the U-bus pipe, the store is carried out immediately and the U-bus acknowledgment is performed when the previous transaction in the pipe completes.
- Burst access is not supported.

4.6.2.1 Memory Protection Violations

The DECRAM module does not acknowledge U-bus accesses that violate the configuration defined in the **BBCMCR**. This causes the “machine check” exception for internal RCPU or error condition for MPC565 / MPC566 external master.

4.6.2.2 DECRAM StandBy Operation Mode

The bus interface and DECRAM control logic are powered by VDD. The memory array(s) is supplied by a separate power pin (VDDSRAM3). If main power is shut off, VDDSRAM3 may subsequently be lowered for purposes of low voltage data retention.

When the DECRAM array is powered by the VDDSRAM3 pin, access to the RAM array is blocked. The application software may use **VSRMCR[LVDRS]** bit value to determine if the content of DECRAM is valid (See **8.12.4 VDDSRAM Control Register (VSRMCR)** for details).

4.7 Branch Target Buffer

In order to reduce branch impact on performance, the branch target buffer (BTB) is implemented as an optional part of the ICDU.

Following is a summary of the BTB features:

- Software controlled BTB enable/disable, lock/unlock and invalidate
- User transparent — no user management required

BTB compensates the branch and branch miss-prediction impact on system performance. It consists of eight “branch target entries” (BTE) and one “branch conditional miss-predicted entry” (BCME). Refer to **Figure 4-9**.

All entries are managed as a fully associative cache. Each entry contains a *tag* and several data buffers related to this tag

4.7.1 BTB Operation

When the RCPU generates a Change Of Flow (COF) address for instruction fetch, the BTB control logic compares it to the tag values currently stored in the tag register file and following events can happen:

- **BTE Miss** — The target address and instruction code data will be stored in one of the BTE entries defined by its control logic. Up to four instructions and their corresponding addresses subsequent to COF target instruction may be saved in each BTE entry. The number of valid instructions currently stored in the BTE entry is written into VDC field of the current BTE entry. The “Valid” flag is set in the end of this process. The entry to be replaced upon miss is chosen based on FIFO replacement method. Thus the BTB can support up to eight different branch target addresses in a program loop.
- **BCME Miss** — Whenever the BTB logic detects that a conditional branch instruction has been transferred to the RCPU, the instructions that follow the branch conditional instruction are stored in the BCME entry. Up to four instruction codes and their adjacent addresses are saved in the BCME entry. The number of valid instructions currently stored in the BCME is written into the VDC field of BCME. The “Valid” flag is set in the end of this process.
- **BTE/BCME Hit** — When the target address of a branch matches one of the valid BTE entries or the BCME entry, two activities take place in parallel:
  - — The BTB supplies all the valid instructions in the matched entry to the RCPU.
  - — The ICDU (which is flushed due to the COF) starts to prefetch compressed instructions (and decompress them) from the address following the last instruction which is stored in the matched BTB entry; It will supply these instructions to the RCPU after the all stored instruction in the matched BTB entry were delivered.

Thus in case of BTB hit the impact of instruction decompression latency is eliminated as well as a latency of instruction storage memory device.

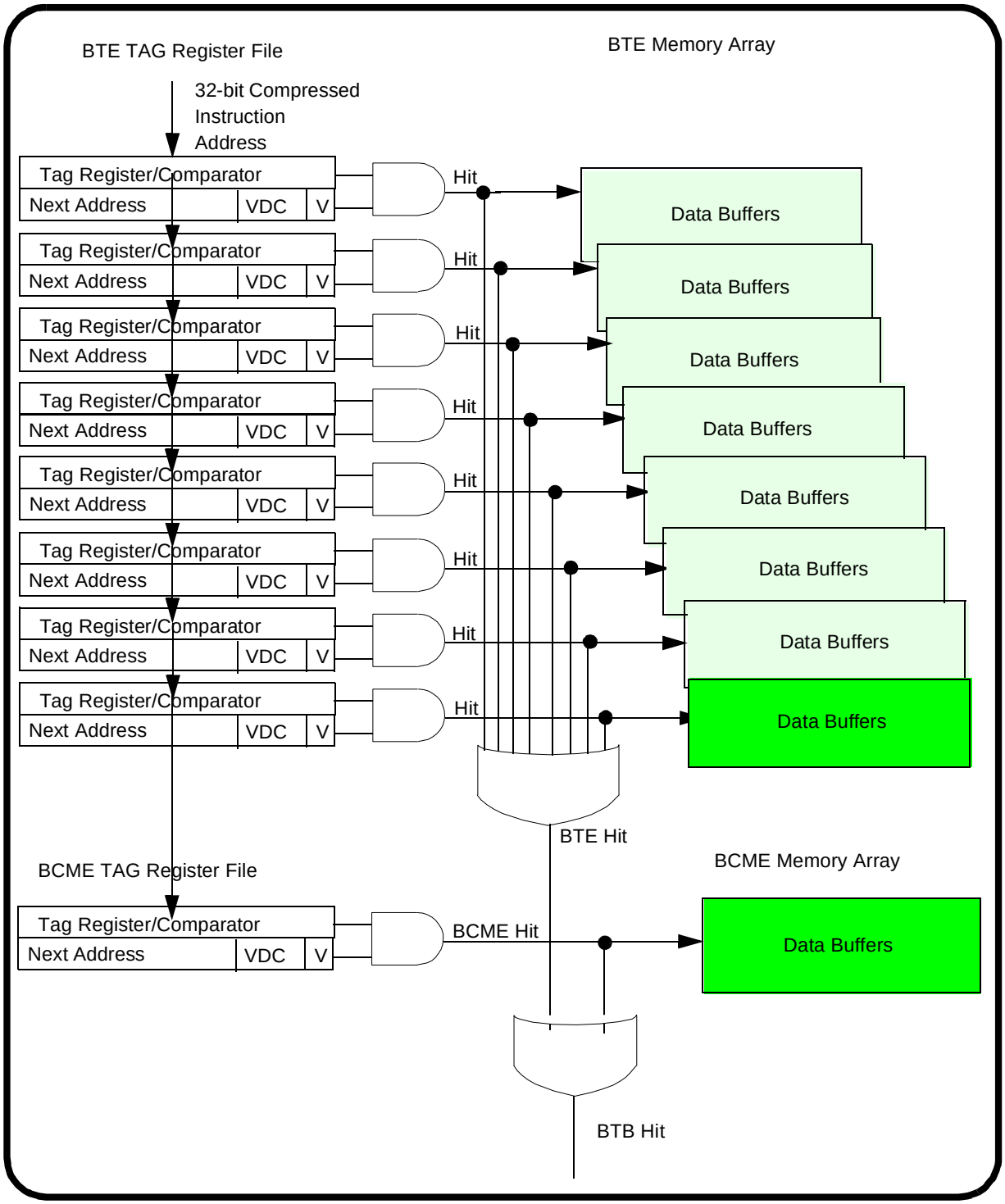


Figure 4-9 BTB Block Diagram

4.7.1.1 BTB Invalidation

Write access to any BBC special purpose register invalidates all BTB entries.

4.7.1.2 BTB Enabling/Disabling

The BTE and BCME operation may be enabled/disabled by programming **BTEE** and **BCMEE** bits in **BBCMCR** register.

4.7.1.3 BTB Inhibit Regions

The BTB operation may be inhibited for some memory regions. The BTB caching is inhibited for a region if the **BTBINH** bit set in the region attribute register (or global region attribute register). See **4.8.2.3 Region Attribute Registers MI_RA[1:3]** and **4.8.2.4 Global Region Attribute Register MI_GRA** for details.

4.8 BBC Programming Model

4.8.1 Address Map

The BBC consists of three separately addressable sections within the internal chip address space.

1. BBC and IMPU control registers. These are mapped in the SPR registers area and may be programmed by using PowerPC's **mtspr/mfspr** instructions.
2. DECompressor vocabulary RAM (DECRAM). DECRAM array occupies the four-Kbyte physical memory (eight-Kbyte MPC565 / MPC566 address space is allocated for DECRAM).
3. Decompressor class configuration registers (DCCR) block. It consists of 15 decompression class configuration registers. These registers are available for word wide read/write accesses through U-bus. The registers occupy a 64-byte physical block (eight-Kbyte chip address space is allocated for the register block).

4.8.1.1 BBC Special Purpose Registers (SPRs)

Table 4-3 BBC SPRs

SPR Number (Decimal)	Address for External Master Access (Hex)	Register Name
528	0x2100	IMPU Global Region Attribute Register (MI_GRA). See **Table 4-8** for bits descriptions.
529	0x2300	External Interrupt Relocation Table Base Address Register (EIBADR). See **Table 4-9** for bits descriptions.
560	0x2110	BBC Module Configuration Register (BBCMCR). See **Table 4-4** for bits descriptions
784	0x2180	IMPU Region Base Address Register 0 (MI_RBA0). See **Table 4-5** for bits descriptions.
785	0x2380	IMPU Region Base Address Register 1 (MI_RBA1). See **Table 4-5** for bits descriptions.
786	0x2580	IMPU Region Base Address Register 2 (MI_RBA2). See **Table 4-5** for bits descriptions.
787	0x2780	IMPU Region Base Address Register 3 (MI_RBA3). See **Table 4-5** for bits descriptions.
816	0x2190	IMPU Region Attribute Register 0 (MI_RA0). See **Table 4-6** for bits descriptions.
817	0x2390	IMPU Region Attribute Register 1 (MI_RA1). See **Table 4-6** for bits descriptions.
818	0x2590	IMPU Region Attribute Register 2 (MI_RA2). See **Table 4-6** for bits descriptions.
819	0x2790	IMPU Region Attribute Register 3(MI_RA3). See **Table 4-6** for bits descriptions.

All the above registers may be accessed in the supervisor mode only. An exception is internally generated by the RCPU if there is an attempt to access them in user mode. An external master receives a transfer error acknowledge when attempting to access a register in user mode.

All the registers are reset using $\overline{\text{HRESET}}$. $\overline{\text{SRESET}}$ alone has no effect on them.

4.8.1.2 DECRAM and DCCR Block

The DECRAM occupies addresses from **0x2F8000** to **0x2F8FFF**. The DCCR block occupies addresses from **0x2FA004** to **0x2FA03F.**

Address for non-implemented memory blocks is not acknowledged, and causes an error condition.

4.8.2 BBC Registers Description

4.8.2.1 BBC Module Configuration Register BBCMCR

BBCMCR — BBC Module Configuration Register **SPR 560**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	D	S	TEST					RESERVED							
HRESET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RE-SERVED		BE	ETRE	EIR	EN_ COMP[2]	EXC_ COMP[2]	DECOMP _SC_EN[2]	OERC[0:1]		BTEE	BCMEE	RESERVED		DCA E	TST
HRESET															
0	0	0	ID[1](19)	0	ID(21)	ID(22)	ID(21)	ID(24:25)		0	0	0	0	0	0

NOTES:
1. ID - gets value of corresponding bit of Reset Configuration Word
2. Available only on the MPC566.

Table 4-4 BBCMCR BBC Module Configuration Register Bit Description s

Bit(s)	Name	Description
0	R	Read Only — For any attempt to write to the DECRAM array while R is set, is terminated with an error. This causes a machine check exception for RCPU. 0 = DECRAM array is Readable and Writable. 1 = DECRAM array is Read only.
1	D	Data Only — The DECRAM array may be used for Instructions and Data or for Data storage only. Any attempt to load instructions from the DCRAM array, while D is set, is terminated with an error This causes a machine check exception for the RCPU. 0 = DECRAM array holds Data and/or Instruction. 1 = DECRAM array holds Data only.
2	S	Supervisor Only. When the bit set (S = 1), only a Supervisor program may access the DECRAM. If a Supervisor program is accessing the array, normal read/write operation will occur. If a User program is attempting to access the array, the access will be terminated with an error This causes a machine check exception for the RCPU. If S = 0, the RAM array is placed in Unrestricted Space and access by both Supervisor and User programs is allowed.
3:7	TEST	The bits can be set in Factory test mode only, User should treat the bits as reserved
8:17	—	Reserved
18	BE	Burst Enable 0 = Burst access is disabled. 1 = Burst access is enabled.
19	ETRE	Exception Table Relocation Enable 0 = Exception Table Relocation is off: BBC does NOT map exception addresses. 1 = Exception Table Relocation is on: BBC maps exception addresses to a table holding branch instructions two memory words apart from each other. The reset value is taken from the reset configuration word bit #19.

Table 4-4 BBCMCR BBC Module Configuration Register Bit Descriptions

Bit(s)	Name	Description
20[1]	EIR	Enhanced External Interrupt Relocation Enable— This bit activates the external interrupt relocation table mechanism. This bit is independent from the value of ETR bit, but if EIR and ETR are enabled, the mapping of external interrupt will be via EIR. 0 = EIR function is disabled. 1 = EIR function is active.
21	EN_ COMP[2]	Enable COMPression — This bit enables the operation of the MPC565 / MPC566 in “Compression ON” mode. The default state is disabled. This bit is read only. 0 = “Decompression ON” mode is disabled. The MPC565 / MPC566 operates only in “Decompression OFF” mode. 1 = “Decompression ON” mode is enabled. The MPC565 / MPC566 may operates with both “Decompression ON” and “Decompression OFF” modes. The bit value is determined by reset configuration word, bit #21.
22	EXC_ COMP[2]	Exception Compression — This bit determines the operation of the MPC565 / MPC566 with exceptions. If this bit is set, the MPC565 / MPC566 assumes that the all exception routines code is compressed; otherwise it is assumed that all exception routines code is not compressed. The reset value is determined by reset configuration word bit #22. 0 = The MPC565 / MPC566 assumes that exception routines are non-compressed 1 = The MPC565 / MPC566 assumes that ALL exception routines are compressed. This bit effects only when EN_COMP bit is set.
23	DECOMP _SC_EN[2]	DECOMPression Show Cycle ENable — This bit determines the way the MPC565 / MPC566 executes instruction show-cycle. The reset value is determined by configuration word bit #21. For further details regarding show cycles execution in “Decompression ON” mode see **4.4.1.2 “Decompression ON” Mode**. 0 = Decompression Show Cycle does not include the bit pointer. 1 = Decompression Show Cycles includes the bit pointer information on the data bus.
24:25	OERC[0:1]	Other Exceptions Relocation Control — These bits effect only if ETRE was enabled; See details in **4.5.2 ETR Operation**. 00: offset 0 01: offset 64 Kbytes 10: offset 512 Kbytes. 11: offset to 0x003FE000 (SRAM start address) The reset value is determined by reset configuration word bits #24, #25
26	BTEE[2]	Branch Target Entries Enable — This bit enables Branch Target Entries of BTB operation 0 = BTE operation is disable 1 = BTE operation is enable.
27	BCMEE[2]	Branch Conditional Miss-predicted Entry Enable — This bit enables Branch Conditional miss-predicted Entry of BTB operation 0 = BCME operation is disable. 1 = BCME operation is enable.
28:29	—	Reserved
30	DCAE	Decompressor Configuration Access Enable — This bit enables DECRAM and DCCR registers accesses from U-bus master (i.e. RCPU, external master). 0 = DECRAM and DCCR registers are locked. 1 = DECRAM allows accesses from the U-bus only. DCAE bit should be set before vocabulary tables loading via U-bus.
31	—	Reserved for BBC Test Operations.

NOTES:
1. Bit 20 of the BBCMCR on the MPC555/556 was OERC. The OERC bits are now bits 24:25.
2. This bit is only available on the MPC566.

BBCMCR PROGRAMMING NOTE

When writing to the **BBCMCR** register, the following instruction after **mtspr BBCMCR, Rx** should be **ISYNC**, to make sure that the programmed value will come into effect before any further action.

4.8.2.2 Region Base Address Registers MI_RBA0-3

The following registers contain 32 bits and define the starting address of the protected regions. There is one register for each of four regions.

MI_RBA[0:3] — Region Base Address Register **SPR 784 – 787**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RA															
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RA				RESERVED											
HRESET															
U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-5 MI_RBA[0:3] Registers Bit Descriptions

Bit(s)	Name	Description
0:19	RA	Region Base address. The RA field provides the base address of the region. The region base address should start on the memory block boundary for the corresponded region size, specified in the region attribute register MI_RA.
20-31	—	Reserved

NOTE

When the MPC565 / MPC566 operates “Decompression ON” mode, a minimum four unused words MUST be left after the last instruction in any region.

4.8.2.3 Region Attribute Registers MI_RA[1:3]

The following registers define protection attributes and size for four memory regions.

MI_RA[0:3] — Region Attribute Register **SPR 816 – 819**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RS															
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RS				PP		RESERVED			G	CMPR‘		BTBINH	RESERVED		
HRESET															
U	U	U	U	U	U	0	0	0	U	U	U	U	0	0	0

Table 4-6 MI_RA[0:3] Registers Bits Description

Bit(s)	Name	Description
0:19	RS	Region size. For byte size by region, see **Table 4-7**.
20:31	PP[1]	Protection bits: 00: Supervisor — No Access, User — No Access. 01: Supervisor — Fetch, User — No Access. 1x: Supervisor — Fetch, User — Fetch.
22:24	—	Reserved
25	G	Guard attribute for region 0 = Speculative fetch is not prohibited from region. Region is not guarded. 1 = Speculative fetch is prohibited from guarded region. An exception will occur under such attempt.
26:27	CMPR[2]	Compressed Region. x0 = The region in not restricted 01 = Region is considered a non-compressed code region Access to the region is allowed only in “Decompression Off” mode 11 = Region is considered a compressed code region.Access to the region is allowed only in “Decompression On” mode
28	BTBINH[2]	BTB Inhibit region 0 = BTB operation is not prohibited for current memory region 1 = BTB operation is prohibited for current memory region.
29:30	—	Reserved

NOTES:

1. G and PP attributes perform similar protection activities on a region. The more protective attribute will be implied on the region if the attributes programming oppose each other.
2. This bit is available only on MPC566.

Table 4-7 Region Size Programming Possible Values

RS field value (Binary)	Size
0000_0000_0000_0000_0000	4 Kbytes
0000_0000_0000_0000_0001	8 Kbytes
0000_0000_0000_0000_0011	16 Kbytes
0000_0000_0000_0000_0111	32 Kbytes
0000_0000_0000_0000_1111	64 Kbytes
0000_0000_0000_0001_1111	128 Kbytes
0000_0000_0000_0011_1111	256 Kbytes
0000_0000_0000_0111_1111	512 Kbytes
0000_0000_0000_1111_1111	1 Mbyte
0000_0000_0001_1111_1111	2 Mbytes
0000_0000_0011_1111_1111	4 Mbytes
0000_0000_0111_1111_1111	8 Mbytes
0000_0000_1111_1111_1111	16 Mbytes
0000_0001_1111_1111_1111	32 Mbytes
0000_0011_1111_1111_1111	64 Mbytes
0000_0111_1111_1111_1111	128 Mbytes
0000_1111_1111_1111_1111	256 Mbytes
0001_1111_1111_1111_1111	512 Mbytes
0011_1111_1111_1111_1111	1 Gbyte
0111_1111_1111_1111_1111	2 Gbytes
1111_1111_1111_1111_1111	4 Gbytes

4.8.2.4 Global Region Attribute Register MI_GRA

MI_GRA register defines protection attributes for memory region, not covered by MI_RB0-3/MI_RBA0-3 registers. It also contains protection regions 0-4 enable bits.

MI_GRA — Global Region Attribute Register **SPR 528**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ENR 0	ENR1	ENR2	ENR3	RESERVED											
HRESET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				PP		RESERVED			G	CMPR		BTBINH	RESERVED		
HRESET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-8 MI_GRA Global Region Attribute Register Bits Descripti on

Bit(s)	Name	Description
0	ENR0	Enable IMPU Region 0 0 = Region 0 is off. 1 = Region 0 is on.
1	ENR1	Enable IMPU Region 1 0 = Region 1 is off. 1 = Region 1 is on.
2	ENR2	Enable IMPU Region 2 0 = Region 2 is off. 1 = Region 2 is on.
3	ENR3	Enable IMPU Region 3 0 = Region 3 is off. 1 = Region 3 is on.
4:19	—	Reserved
20:21	PP	Protection Bits 00: Supervisor – No Access, User – No Access. 01: Supervisor – Fetch, User – No Access. 1x: Supervisor – Fetch, User – Fetch.
22:24	—	Reserved
25	G	Guard attribute for region 0 = Fetch is not prohibited from region. Region is not guarded. 1 = Fetch is prohibited from guarded region. An exception will occur under such attempt.
26:27	CMPR[1]	Compressed Region. x0 = The region in not restricted. 01= Region is considered a non-compressed code region. Access to the region is allowed only in “Decompression Off” mode. 11= Region is considered a compressed code region. Access to the region is allowed only in “Decompression On” mode.
28	BTBINH[2]	BTB Inhibit region 0 = BTB operation is not prohibited for current memory region. 1 = BTB operation is prohibited for current memory region.
29:31	—	Reserved

NOTES:
1. This bit is only available on the MPC566.

NOTE

The MI_GRA register should be programmed to enable fetch access (PP and G bits) before RCPU **MSR[IR]** bit is set.

4.8.2.5 External Interrupt Relocation Table Base Address Register — EIBADR

EIBADR — External Interrupt Relocation Table Base Address Register **SPR 529**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base Address															
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Base Address					RESERVED										
HRESET															
U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-9 EIBADR External Interrupt Relocation Table Base Address Register Bit Descriptions

Bit(s)	Name	Description
0:20	BA	External Interrupt Relocation Table Base Address bits [0:20]
21:31	—	Reserved

4.8.2.6 Decompressor Class Configuration Registers (DCCR1-15)

The DCCR fields are programmed to achieve maximum flexibility in the vocabulary tables placement into the two DECRAM banks under constraints, implied by hardware, which are:

- a bypass field must always be the second field in the compressed instruction;
- when fetching 32 bits of de-compressed instruction from the DECRAM, each 16 bits will be read from different RAM banks.

The DCCR registers should be programmed with data supplied by the code compression tool, in order to be correlated with the compressed code.

4.8.2.7 Decompressor Class Configuration Registers (DCCR1-DCCR15)

DCCR1— Decompressor Class Configuration Registers	**0x2F A004**
DCCR2	**0x2F A008**
DCCR3	**0x2F A00C**
DCCR4	**0x2F A010**
DCCR5	**0x2F A014**
DCCR6	**0x2F A018**
DCCR7	**0x2F A01C**
DCCR8	**0x2F A020**
DCCR9	**0x2F A024**
DCCR10	**0x2F A028**
DCCR11	**0x2F A02C**
DCCR12	**0x2F A030**
DCCR13	**0x2F A034**
DCCR14	**0x2F A038**
DCCR15	**0x2F A03C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TP1LEN				TP2LEN				TP1BA							TP2BA
HRESET															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
TP2BA						AS	DS	RESERVED							
HRESET															
U	U	U	U	0	U	U	U	0	0	0	0	0	0	0	0

Table 4-10 DCCR0-DCCR15 Decompressor Class Configuration Registers Bit Descriptions

Bit(s)	Name	Description
0:3	TP1LEN	Length and Type of Table Pointer 1 — This field's value defines the length of the field that contains a pointer to the first vocabulary table allocated for the class. 0x0 - Empty field 0x1 - Reserved 0x2 - TP1 length is 2 bits 0x3 - TP1 length is 3 bits 0x4 - TP1 length is 4 bits 0x5 - TP1 length is 5 bits 0x6 - TP1 length is 6 bits 0x7 - TP1 length is 7 bits 0x8 - TP1 length is 8 bits 0x9 - TP1 length is 9 bits 0xA to 0xF - Reserved
4:7	TP2LEN	Length and Type of Table Pointer 2 — This field's value defines the length of the field that contains either a pointer to the second vocabulary table allocated for the class or a bypass field. 0x0 - Empty field 0x1 - Reserved 0x2 - TP2 length is 2 bits 0x3 - TP2 length is 3 bits 0x4 - TP2 length is 4 bits 0x5 - TP2 length is 5 bits 0x6 - TP2 length is 6 bits 0x7 - TP2 length is 7 bits 0x8 - TP2 length is 8 bits 0x9 - TP2 length is 9 bits 0xA - Reserved 0xB - Reserved 0xC - TP2 field is a 10 bits compact bypass field 0xD - TP2 field is a 15 bits compact bypass field 0xE - TP2 field is a 16 bits bypass field 0xF - Reserved.
8:14	TP1BA	Base address for vocabulary table in RAM Bank 1 — This field specifies the base page address of the class' vocabulary table that resides in RAM Bank 1.
15:21	TP2BA	Base address for vocabulary table in RAM Bank 2 — This field specifies the base page address of the class' vocabulary table that resides in RAM Bank 2.
22	AS	Address Swap specification 0 = Address swap operation will not be performed for the class. 1 = Address swap operation will be performed for the class For further details concerning AS operation refer to **Table 4-11**.
23	DS	Data swap specification 0 = Data swap operation will not be performed for the class. 1 = Data swap operation will be performed for the class. For further details concerning DS operation refer to **Table 4-11**.
24:31	—	Reserved

Table 4-11 Instruction Layout Encoding

Configuration	Configu-ration Code	TP1 Points To RAM #	TP2 Points To RAM #	TP1BA Points To		TP2BA Points To		AS	DS	Com-pressed. Instruction Layout
				RAM #	Vocab.	RAM #	Vocab.			
Single Segment Full Compression	CLASS 1	1 and 2	—	1	V1	2	V2	—	—	X1[1]
Twin Segments Full Compression	CLASS 2a	1	2	1	V1	2	V2	—	0	X1 X2
Twin Segments Full Compression With Swapped Vocabularies (Vocabulary In RAM #2 For MSB Segment)	CLASS 2b				V2		V1		1	X2 X1
Left Segment Compression, Right Segment Bypassed, Vocabulary In RAM #1	CLASS 3a	1	Bypass	1	V1	—	—	0	0	X1 BP[2]
Left Segment Compression, Right Segment Bypassed, Vocabulary In RAM #2	CLASS 3b	2		—	-	2	V1	1		X1 BP
Left Segment Bypassed, Right Segment Compression, Vocabulary In RAM #1	CLASS 4b	1		1	V2	—	—	0	1	X2 BP
Left Segment Bypassed, Right Segment Compression, Vocabulary In RAM #2	CLASS 4a	2		—	—	2	V2	1		X2 BP

NOTES:
1. X1,X2 - pointers to vocabularies
2. BP - the bypassed data

SECTION 5
UNIFIED SYSTEM INTERFACE UNIT (USIU)

5.1 USIU Introduction

The unified system interface unit (USIU) of the MPC565 / MPC566 consists of several functional modules that control system start-up, system initialization and operation, system protection, and the external system bus. The MPC565 / MPC566 USIU functions include the following:

- System configuration and protection
- Enhanced interrupt controller
- System reset monitoring and generation
- Clock synthesizer
- Power management
- External bus interface (EBI)
- Memory controller supports four memory banks
- Debug support

5.2 USIU Module Overview

The system configuration and protection features control the overall system configuration and provides various monitors and timers, including the bus monitor, software watchdog timer, periodic interrupt timer, PPC decrementer, time base, and real-time clock. The interrupt controller supports up to eight external interrupts, eight levels for all internal USIU interrupt sources and 32 levels for internal peripheral modules reside on IMB bus. It has an enhanced mode of operation, which simplifies the MPC565 / MPC566 interrupt structure and speeding up interrupt processing.

The USIU provides several pinout configurations that allow up to 64 general-purpose I/O, external 32-bit port that supports internal and external masters, and various debug functions. Refer to **SECTION 6 SYSTEM CONFIGURATION AND PROTECTION** for details.

The USIU supports the internal flash censorship mechanism. Any attempt to operate the MPC565 / MPC566 from the external world, while the internal flash is in censor mode, locks the internal flash. The flash is blocked after following events:

- Booting from external memory;
- External master access into the MPC565 / MPC566;
- Debug port/READI accesses are performed.

The clock synthesizer generates the clock signals used by the USIU as well as the other modules and external devices. This circuitry can generate the system clock from a 4-MHz or 20-MHz crystal.

The USIU supports various low-power modes. Each one supplies a different range of power consumption, functionality and wake-up time. The clock scheme supports low power modes for applications that use the baud rate generators and/or serial ports during the standby mode. The main system clock can be changed dynamically while the baud rate generators and serial ports work with a fixed frequency. Refer to **SECTION 8 CLOCKS AND POWER CONTROL** for details.

The EBI handles the transfer of information between the internal busses and the memory or peripherals in the external address space. The MPC565 / MPC566 is designed to allow external bus masters to request and obtain mastership of the system bus, and if required access the on-chip memory and registers. Refer to **SECTION 9 EXTERNAL BUS INTERFACE** for details.

The memory controller module provides glueless interface to many types of memory devices and peripherals. It supports up to four memory banks.Refer to **SECTION 10 MEMORY CONTROLLER** for details.

Figure 5-1 shows the USIU block diagram.

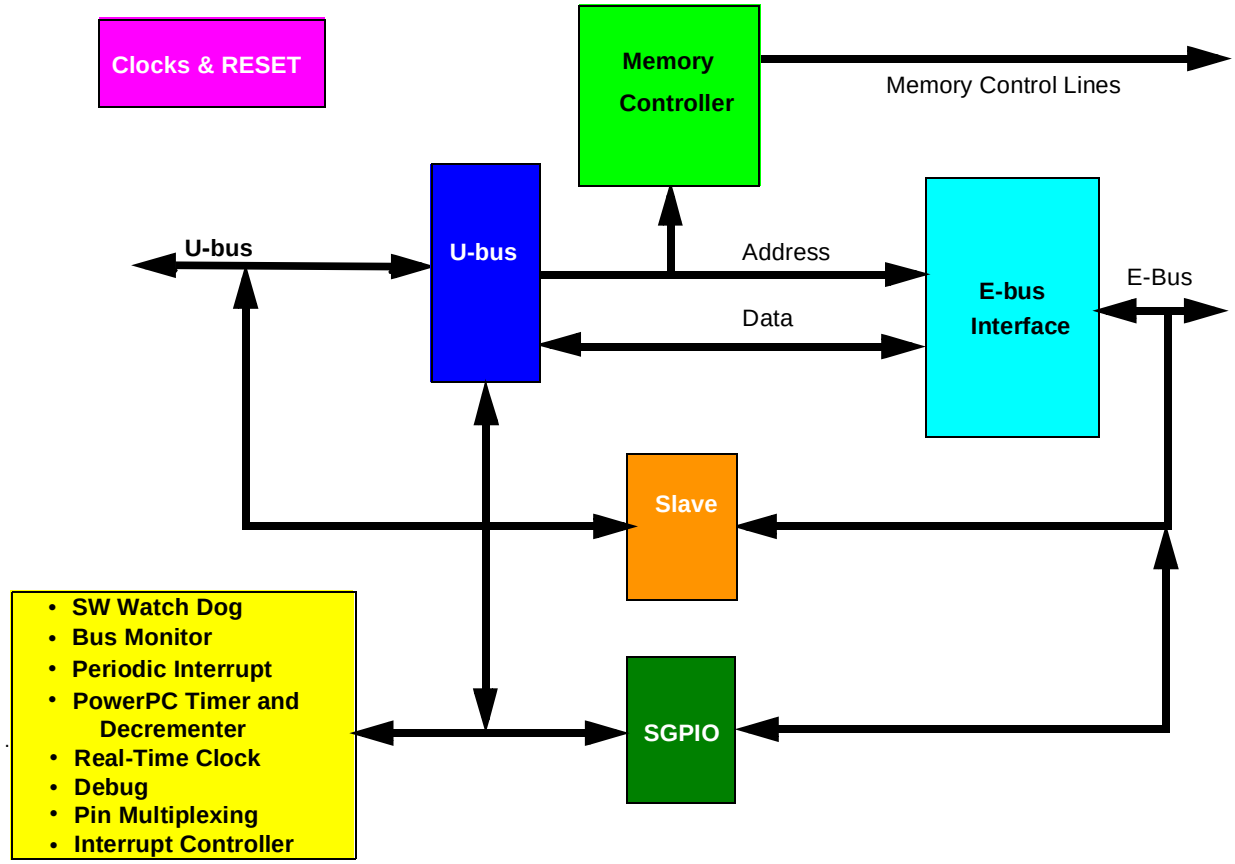


Figure 5-1 USIU Block Diagram

5.2.1 USIU Address Map

Table 5-1 is an address map of the SIU registers. Where not otherwise noted, registers are 32 bits wide. The address shown for each register is relative to the base address of the MPC565 / MPC566 internal memory map. The internal memory block can reside in one of eight possible four-Mbyte memory spaces. See **Figure 1-3** for details.

Table 5-1 USIU Address Map

Address	Register
0x2F C000	SIU Module Configuration Register (SIUMCR) See **Table 6-7** for bit descriptions.
0x2F C004	System Protection Control Register (SYPCR) See **Table 6-15** for bit descriptions.
0x2F C008	Reserved
0x2F C00E[1]	Software Service Register (SWSR) See **Table 6-16** for bit descriptions.
0x2F C010	Interrupt Pending Register (SIPEND).

Table 5-1 USIU Address Map (Continued)

Address	Register
0x2F C014	Interrupt Mask Register (SIMASK) See **SIMASK is a 32 bit read/write register. Each bit in the register corresponds to an interrupt request bit in the SIPEND register.** for bit descriptions.
0x2F C018	Interrupt Edge Level Mask (SIEL) See **6.14.2.7 SIU Interrupt Edge Level Register (SIEL)** for bit descriptions.
0x2F C01C	Interrupt Vector (SIVEC) See **6.14.2.8 SIU Interrupt Vector Register** for bit descriptions.
0x2F C020	Transfer Error Status Register (TESR) See **Table 6-17** for bit descriptions.
0x2F C024	USIU General-Purpose I/O Data Register (SGPIODT1) See **Table 6-23** for bit descriptions.
0x2F C028	USIU General-Purpose I/O Data Register 2 (SGPIODT2) See **Table 6-24** for bit descriptions.
0x2F C02C	USIU General-Purpose I/O Control Register (SGPIOCR) See **Table 6-25** for bit descriptions.
0x2F C030	External Master Mode Control Register (EMCR) See **Table 6-14** for bit descriptions.
0x2F C03C[1]	Pads Module Configuration Register (PDMCR) See **Table 2-3** for bit descriptions.
0x2F C040	Interrupt Pend2 Register (SIPEND2) See **6.14.2.2 SIU Interrupt Pending Register 2** for bit descriptions.
0x2F C044	Interrupt Pend3 Register (SIPEND3) See **6.14.2.3 SIU Interrupt Pending Register 3** for bit descriptions.
0x2F C048	Interrupt Mask2 Register (SIMASK2) See **6.14.2.5 SIU Interrupt Mask Register 2** for details.
0x2F C04C	Interrupt Mask3 Register (SIMASK3) See **6.14.2.6 SIU Interrupt Mask Register 3** for details.
0x2F C050	Interrupt In-Service2 Register (SISR2) See **6.14.2.9 Interrupt In-Service Registers** for details.
0x2F C054	Interrupt In-Service3 Register (SISR3) See **6.14.2.9 Interrupt In-Service Registers** for details.
0x2F C0FC — 0x2F C0FF	Reserved
Memory Controller Registers	
0x2F C100	Base Register 0 (BR0) See **Table 10-7** for bit descriptions.
0x2F C104	Option Register 0 (OR0) See **Table 10-8** for bit descriptions.
0x2F C108	Base Register 1 (BR1) See **Table 10-7** for bit descriptions.
0x2F C10C	Option Register 1 (OR1) See **Table 10-8** for bit descriptions.
0x2F C110	Base Register 2 (BR2) See **Table 10-7** for bit descriptions.
0x2F C114	Option Register 2 (OR2) See **Table 10-8** for bit descriptions.

Table 5-1 USIU Address Map (Continued)

Address	Register
0x2F C118	Base Register 3 (BR3) See **Table 10-7** for bit descriptions.
0x2F C11C	Option Register 3 (OR3) See **Table 10-8** for bit descriptions.
0x2F C120 — 0x2F C13C	Reserved
0x2F C140	Dual-Mapping Base Register (DMBR) See **Table 10-9** for bit descriptions.
0x2F C144	Dual-Mapping Option Register (DMOR) See **Table 10-10** for bit descriptions.
0x2F C148 — 0x2F C174	Reserved
0x2F C178[1]	Memory Status (MSTAT) See **Table 10-6** for bit descriptions.
0x2F C17A — 0x2F C1FC	Reserved
System Integration Timers	
0x2F C200	Time Base Status and Control (TBSCR) See **Table 6-18** for bit descriptions.
0x2F C204	Time Base Reference 0 (TBREF0) See **6.14.4.3 Time Base Reference Registers** for bit descriptions.
0x2F C208	Time Base Reference 1 (TBREF1) See **6.14.4.3 Time Base Reference Registers** for bit descriptions.
0x2F C20C — 0x2F C21C	Reserved
0x2F C220	Real-Time Clock Status and Control (RTCSC) See **Table 6-19** for bit descriptions.
0x2F C224	Real-Time Clock (RTC) See **6.14.4.6 Real-Time Clock Register (RTC)** for bit descriptions.
0x2F C228	Real-Time Alarm Seconds (RTSEC) — Reserved
0x2F C22C	Real-Time Alarm (RTCAL) See **6.14.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.
0x2F C230 — 0x2F C23C	Reserved
0x2F C240	PIT Status and Control (PISCR) See **Table 6-20** for bit descriptions.
0x2F C244	PIT Count (PITC) See **Table 6-21** for bit descriptions.
0x2F C248	PIT Register (PITR) See **Table 6-22** for bit descriptions.
0x2F C24C — 0x2F C27C	Reserved
Clocks and Reset	
0x2F C280	System Clock Control Register (SCCR) See **Table 8-9** for bit descriptions.
0x2F C284	PLL Low-Power and Reset Control Register (PLPRCR) See **Table 8-11** for bit descriptions.

Table 5-1 USIU Address Map (Continued)

Address	Register
0x2F C288[1]	Reset Status Register (RSR) See **Table 7-3** for bit descriptions.
0x2F C28C[1]	Change of Lock Interrupt Register (COLIR) See **Table 8-12** for bit descriptions.
0x2F C290[1]	VDDSRAM1 Control Register (VSRCR) See **Table 8-13** for bit descriptions.
0x2F C294 — 0x2F C2FC	Reserved
System Integration Timer Keys	
0x2F C300	Time Base Status and Control Key (TBSCRK) See **Table 8-8** for bit descriptions.
0x2F C304	Time Base Reference 0 Key (TBREF0K) See **Table 8-8** for bit descriptions.
0x2F C308	Time Base Reference 1 Key (TBREF1K) See **Table 8-8** for bit descriptions.
0x2F C30C	Time Base and Decrementor Key (TBK) See **Table 8-8** for bit descriptions.
0x2F C310 — 0x2F C31C	Reserved
0x2F C320	Real-Time Clock Status and Control Key (RTCSCK) See **Table 8-8** for bit descriptions.
0x2F C324	Real-Time Clock Key (RTCK) See **Table 8-8** for bit descriptions.
0x2F C328	Real-Time Alarm Seconds Key (RTSECK) See **Table 8-8** for bit descriptions.
0x2F C32C	Real-Time Alarm Key (RTCALK) See **Table 8-8** for bit descriptions.
0x2F C330 — 0x2F C33C	Reserved
0x2F C340	PIT Status and Control Key (PISCRIK) See **Table 8-8** for bit descriptions.
0x2F C344	PIT Count Key (PITCK) See **Table 8-8** for bit descriptions.
0x2F C348 — 0x2F C37C	Reserved
Clocks and Reset Keys	
0x2F C380	System Clock Control Key (SCCRK) See **Table 8-8** for bit descriptions.
0x2F C384	PLL Low-Power and Reset Control Register Key (PLPRCRK) See **Table 8-8** for bit descriptions.
0x2F C388	Reset Status Register Key (RSRK) See **Table 8-8** for bit descriptions.
0x2F C38C — 0x2F C3FC	Reserved

NOTES:
1. 16-bit register.

5.2.2 USIU Special Purpose Registers

Table 5-2 lists the MPC565 / MPC566 special purpose registers (SPR). These registers can be accessed with the PowerPC™ **mtspr** and **mfspr** instructions, or from an external master (refer to **6.2 External Master Modes** for details). All registers are 32 bits wide.

NOTE

RCPU special purpose registers can not be accessed by external master.

Table 5-2 USIU Special Purpose Registers

Internal Address[0:31]	Register	Decimal Address spr[5:9]:spr[0:4][1]
0x2C00	Decrementer (DEC). See **3.9.5 Decrementer Register (DEC)** for more information.	22
0x1880	Time Base — Read (TB). See **Table 3-11** for bit descriptions	268
0x1A80	Time Base Upper — Read (TBU). See **6.14.4.2 Time Base SPRs** for bit descriptions	269
0x3880	Time Base — Write (TB). See **Table 3-14** for bit descriptions	284
0x3A80	Time Base Upper — Write (TBU). See **6.14.4.2 Time Base SPRs** for bit descriptions	285
0x3B30	Internal Memory Mapping Register (IMMR). See **Table 6-13** for bit descriptions.	638

NOTES:
1. Bits [0:17] and [28:31] are all 0.

Table 5-3 shows the MPC565 / MPC566 address format for special purpose registers access. For an external master, accessing a PowerPC SPR, address bits [0:17] and [28:31] are compared to zeros to confirm that an SPR access is valid. See **6.2.1 Operation in External Master Modes** for more details.

Table 5-3 MPC565 / MPC566 Address Format for SPR Access From External Bus

0:17	18:27	28:31
0..0	spr[0:9]	0000

SECTION 6
SYSTEM CONFIGURATION AND PROTECTION

The MPC565 / MPC566 incorporates many system functions that normally must be provided in external circuits. In addition, it is designed to provide maximum system safeguards again hardware and/or software faults. The system configuration and protection sub-module provides the following features:

- **System Configuration** — The USIU allows the configuration of the system according to the particular requirements. The functions include control of show cycle operation, pin multiplexing, and internal memory map location. System configuration also includes a register containing part and mask number constants to identify the part in software.

- **Interrupt Controller** — The interrupt controller receives interrupt requests from a number of internal and external sources and directs them on a single interrupt-request line to the RCPU.

- **General-Purpose I/O** — The USIU provides 64 pins for general-purpose I/O. The SGPIO pins are multiplexed with the address and data pins.

- **External Master Modes Support —** External master modes are special modes of operation that allow an alternate master on the external bus to access the internal modules for debugging and backup purposes.

- **Bus Monitor** — The SIU provides a bus monitor to watch internal to external accesses. It monitors the transfer acknowledge (TA) response time for internal to external transfers. A transfer error acknowledge ($\overline{\text{TEA}}$) is asserted if the TA response limit is exceeded. This function can be disabled.

- **Software Watchdog Timer (SWT)** — The SWT asserts a reset or non-maskable interrupt (as selected by the system protection control register) if the software fails to service the SWT for a designated period of time (e.g., because the software is trapped in a loop or lost). After a system reset, this function is enabled with a maximum time-out period and asserts a system reset if the time-out is reached. The SWT can be disabled or its time-out period can be changed in the SYPCR. Once the SYPCR is written, it cannot be written again until a system reset.

- **Periodic Interrupt Timer (PIT)** — The SIU provides a timer to generate periodic interrupts for use with a real-time operating system or the application software. The PIT provides a period from one µs to four seconds with a four-MHz crystal or 200 ns to 0.8 ms with a 20-MHz crystal. The PIT function can be disabled.

- **Power-PC Time Base Counter (TB)** — The TB is a 64-bit counter defined by the MPC565 / MPC566 architecture to provide a time base reference for the operating system or application software. The TB has four independent reference reg-

isters which can generate a maskable interrupt when the time-base counter reaches the value programmed in one of the four reference registers. The associated bit in the TB status register will be set for the reference register which generated the interrupt.

- **Power-PC Decrementer (DEC)** — The DEC is a 32-bit decrementing counter defined by the MPC565 / MPC566 architecture to provide a decrementer interrupt. This binary counter is clocked by the same frequency as the time base (also defined by the MPC565 / MPC566 architecture). The period for the DEC when driven by a 4-MHz oscillator is 4295 seconds, which is approximately 71.6 minutes.

- **Real-Time Clock (RTC)** — The RTC is used to provide time-of-day information to the operating system or application software. It is composed of a 45-bit counter and an alarm register. A maskable interrupt is generated when the counter reaches the value programmed in the alarm register. The RTC is clocked by the same clock as the PIT.

- **Freeze Support** — The SIU allows control of whether the SWT, PIT, TB, DEC and RTC should continue to run during the freeze mode.

Figure 6-1 shows a block diagram of the system configuration and protection logic.

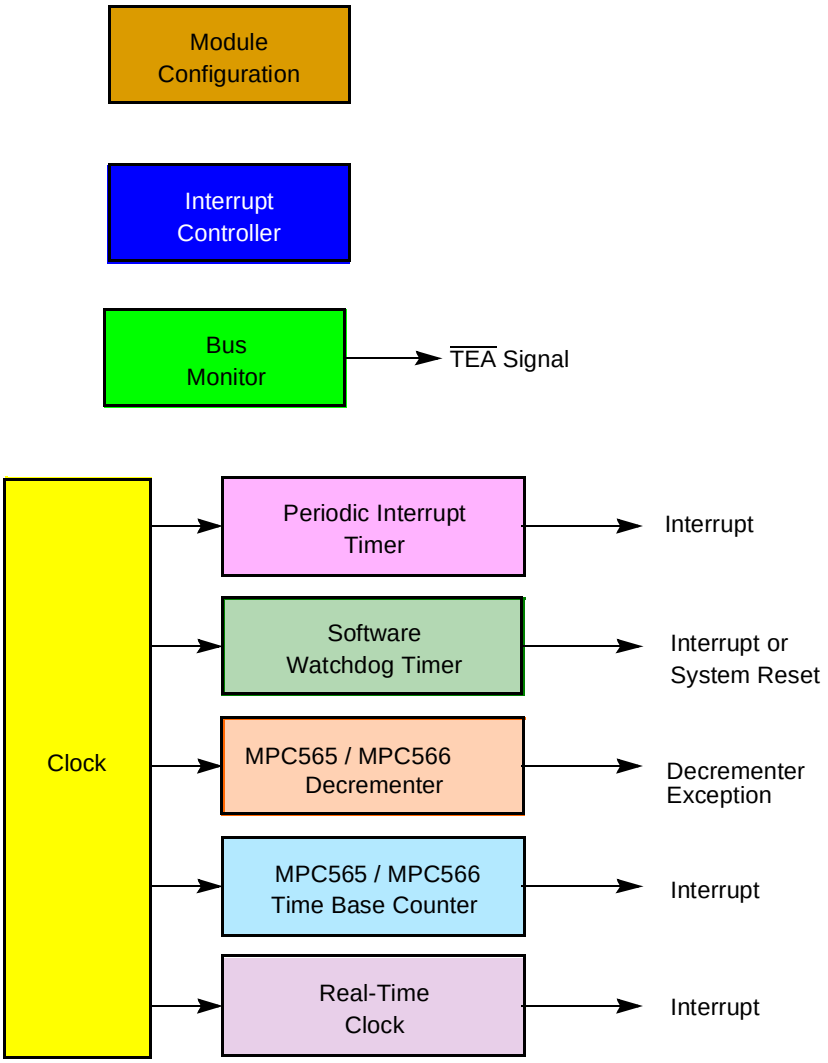


Figure 6-1 System Configuration and Protection Logic

6.1 System Configuration

The SIU allows the configuration of the system according to the particular requirements. The functions include control of show cycle operation, pin multiplexing, and internal memory map location. System configuration also includes a register containing part and mask number constants to identify the part in software.

System configuration registers include the system configuration register (SIUMCR), the internal memory mapping register (IMMR). Refer to **6.14 System Configuration and Protection Registers** for register diagrams and bit descriptions.

6.1.1 USIU Pins Multiplexing

Some of the functions defined in the various sections of the SIU (external bus interface, memory controller, and general-purpose I/O) share pins. **Table 6-1** summarizes how the pin functions of these multiplexed pins are assigned.

Table 6-1 USIU Pins Multiplexing Control

Pin Name	Multiplexing Controlled By:
IRQ0/SGPIOC0 IRQ1/RSV/SGPIOC1 IRQ2/CR/SGPIOC2/MTS IRQ3/KR/RETRY/SGPIOC3 IRQ4/AT2/SGPIOC4 IRQ5/SGPIOC5/MODCK1 IRQ6/MODCK2 IRQ7/MODCK3	At Power-On Reset: MODCK[1:3] Otherwise: Programmed in SIUMCR
SGPIOC6/FRZ/PTR SGPIOC7/IRQ_OUT/LWP0 BG/VF0/LWP1 BR/VF1/IWP2 BB/VF2/IWP3 IWP[0:1]/VFLS[0:1] BI/STS WE(0:3)/BE(0:3)/AT(0:3) TDI/DSDI TCK/DSCK TDO/DSDO	Programmed in SIUMCR and Hard Reset Configuration
DATA[0:31]/SGPIOD[0:31] ADDR[8:31]/SGPIOA[8:31]	Programmed in SIUMCR
RSTCONF/TEXP	At Power-On Reset: RSTCONF Otherwise: Programmed in SIUMCR

6.1.2 Memory Mapping

The MPC565 / MPC566 internal memory space can be assigned to one of eight locations.

The internal memory map is organized as a single four-Mbyte block. The user can assign this block to one of eight locations by programming the ISB field in the internal memory mapping register (IMMR). The eight possible locations are the first eight four-Mbyte memory blocks starting with address 0x0000 0000. (Refer to **Figure 6-2**.)

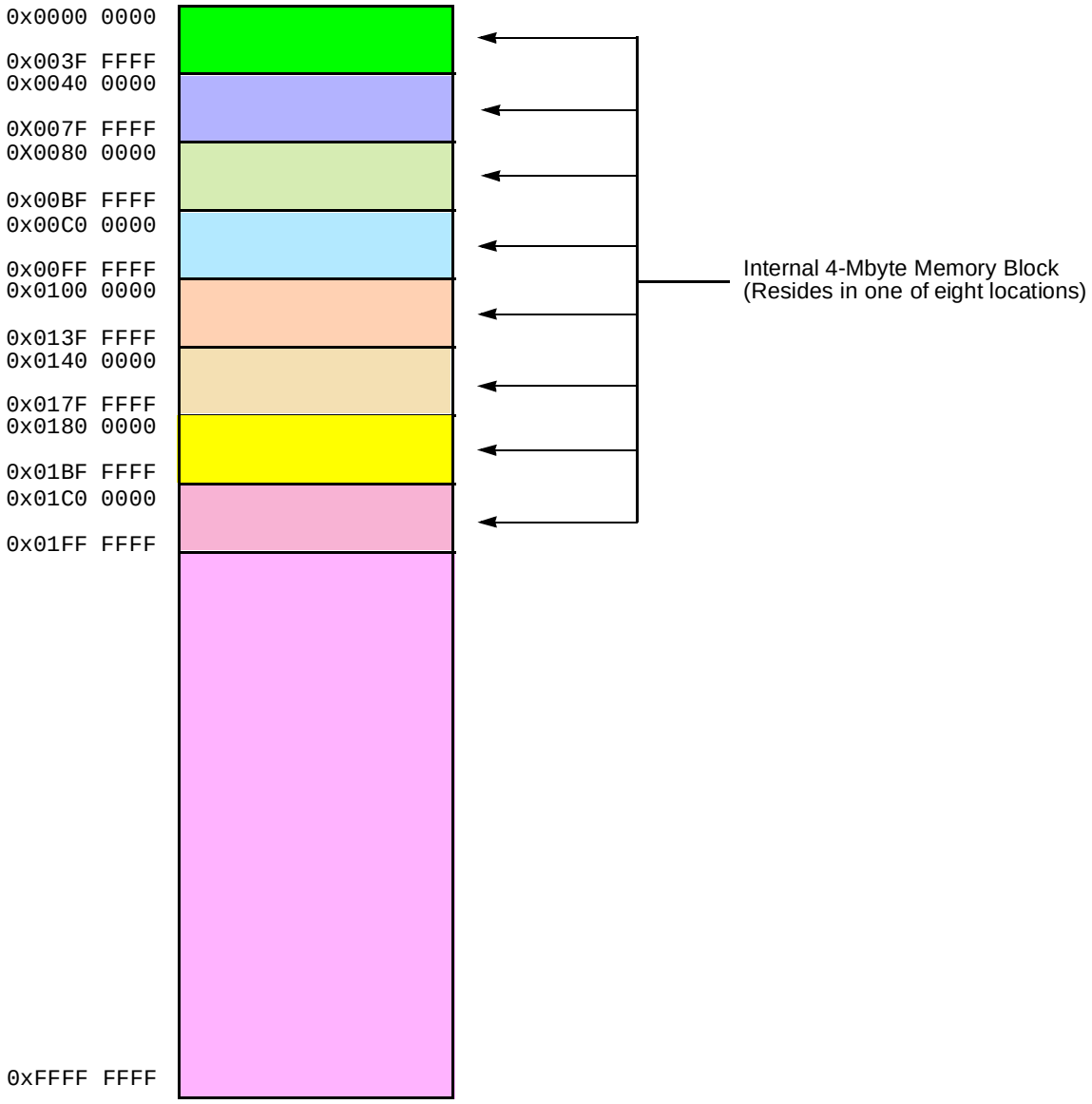


Figure 6-2 MPC565 / MPC566 Memory Map

6.1.3 Arbitration Support

Two bits in the SIUMCR control USIU bus arbitration. The external arbitration (EARB) bit determines whether arbitration is performed internally or externally. If EARB is cleared (internal arbitration), the external arbitration request priority (EARP) bit determines the priority of an external master's arbitration request. The operation of the internal arbiter is described in **9.5.6.4 Internal Bus Arbiter**.

6.2 External Master Modes

External master modes are special modes of operation that allow an alternative master on the external bus to access the internal modules for debugging and backup pur-

poses. They provide access to the internal buses (U-bus and L-bus) and to the intermodule bus (IMB3).

There are two external master modes. Peripheral mode (enabled by setting PRPM in the EMCR) uses a special slave mechanism which shuts down the RCPU and an alternative master on the external bus can perform accesses to any internal bus slave. Slave mode (enabled by setting SLVM and clearing PRPM in the EMCR) enables an external master to access any internal bus slave while the RCPU is fully operational. Both modes can be enabled and disabled by software. In addition, peripheral mode can be selected from reset.

The internal bus is not capable of providing fair priority between internal RCPU accesses and external master accesses. If the bandwidth of external master accesses is large, it is recommended that the system forces gaps between external master accesses in order to avoid suspension of internal RCPU activity.

The MPC565 / MPC566 does not support burst accesses from an external master; only single accesses of 8, 16, or 32 bits can be performed. The MPC565 / MPC566 asserts burst inhibit ($\overline{BI}$) on any attempt to initiate a burst access to internal memory.

The MPC565 / MPC566 provides memory controller services for external master accesses (single and burst) to external memories. See **SECTION 10 MEMORY CONTROLLER** for details.

6.2.1 Operation in External Master Modes

The external master modes are controlled by the EMCR register, which contains the internal bus attributes. The default attributes in the EMCR enable the external master to configure EMCR with the required attributes, and then access the internal registers. The external master must be granted external bus ownership in order to initiate the external master access. The SIU compares the address on the external bus to the allocated internal address space. If the address is within the internal space, the access is performed with the internal bus. The internal address space is determined according to ISB field (see **6.14.1.2 Internal Memory Map Register** for details). The external master access is terminated by the $\overline{TA}$, $\overline{TEA}$ or $\overline{RETRY}$ signal on the external bus.

A deadlock situation might occur if an internal-to-external access is attempted on the internal bus while an external master access is initiated on the external bus. In this case, the SIU will assert the $\overline{RETRY}$ on the external bus in order to relinquish and retry the external access until the internal access is completed. The internal bus will deny other internal accesses for the next eight clocks in order to complete the pending accesses and prevent additional internal accesses from being initiated on the internal bus. The SIU will also mask internal accesses to support consecutive external accesses if the delay between the external accesses is less than four clocks. The external master access and retry timings are described in **9.5.11 Bus Operation in External Master Modes**.

The external master may access the internal MPC565 / MPC566 special registers that are located outside the RCPU. In order to access one of these MPC565 / MPC566 registers, program the EMCR to MPC565 / MPC566 special register access (CONT = 1

and SUPU = 0 in EMCR). Next, access the register by providing the address according to the MPC565 / MPC566 address map. Only the first external master access that follows EMCR setting will be assigned to the special register map; the next accesses will be directed to the normal address map. This is done in order to enable access to the EMCR again after the required MPC565 / MPC566 special register access.

Peripheral mode does not require external bus arbitration between the external master and the internal RCPU, since the internal RCPU is disabled. The $\overline{BR}$ and $\overline{BB}$ signals should be connected to ground, and the internal bus arbitration should be selected in order to prevent the “slave” MPC565 / MPC566 from occupying the external bus. Internal bus arbitration is selected by clearing the EARB bit in the SIUMCR (see **6.14.1.1 SIU Module Configuration Register (SIUMCR)**).

6.2.2 Address Decoding for External Accesses

During an external master access, the USIU compares the external address with the internal address block to determine if MPC565 / MPC566 operation is required. Since only 24 of the 32 internal address bits are available on the external bus, the USIU assigns zeros to the most significant address bits (ADDR[0:7]).

The address compare sequence can be summarized as follows:

- Normal external access. If the CONT bit in EMCR is cleared, the address is compared to the internal address map. Refer to **6.14.1.3 External Master Control Register (EMCR)**.
- MPC565 / MPC566 special register external access. If the CONT bit in EMCR is set by the previous external master access, the address is compared to the MPC565 / MPC566 special address range. See **5.2.2 USIU Special Purpose Registers** for a list of the SPRs in the USIU.
- Memory controller external access. If the first two comparisons do not match, the internal memory controller determines whether the address matches an address assigned to one of the regions. If it finds a match, the memory controller generates the appropriate chip select and attribute accordingly

When trying to fetch an MPC565 / MPC566 special register from an external master, the address might be aliased to one of the external devices on the external bus. If this device is selected by the MPC565 / MPC566 internal memory controller, this aliasing does not occur since the chip select is disabled. If the device has its own address decoding or is being selected by external logic, this case should be resolved.

6.3 USIU General-Purpose I/O

The USIU provides 64 general-purpose I/O (SGPIO) pins. The SGPIO pins are multiplexed with the address and data pins. In single-chip mode, where communicating with external devices is not required, all 64 SGPIO pins can be used. In multiple-chip mode, only eight SGPIO pins are available. Another configuration allows the use of the address bus for instruction show cycles while the data bus is dedicated to SGPIO functionality. The functionality of these pins is assigned by the single-chip (SC) bit in the SIUMCR. (See **6.14.1.1 SIU Module Configuration Register (SIUMCR)**.)

SGPIO pins are grouped as follows:

- Six groups of eight pins each, whose direction is set uniformly for the whole group
- 16 single pins whose direction is set separately for each pin

Table 6-2 describes the SGPIO signals, and all available configurations. The SGPIO registers are described in **6.14.5 General-Purpose I/O Registers**.

Table 6-2 SGPIO Configuration

SGPIO Group Name	Individual Pin Control	Direction Control	Available When SC = 00 (32-bit Port Size Mode)	Available When SC = 01 (16-bit Port Size Mode)	Available When SC = 10 (Single-Chip Mode with Trace)	Available When SC = 11 (Single-Chip Mode)
SGPIOD[0:7]		GDDR0			X	X
SGPIOD[8:15]		GDDR1			X	X
SGPIOD[16:23]		GDDR2		X	X	X
SGPIOD[24:31]	X	SDDRD[23:31]		X	X	X
SGPIOC[0:7][1]	X	SDDRC[0:7]				
SGPIOA[8:15]		GDDR3				X
SGPIOA[16:23]		GDDR4				X
SGPIOA[24:31]		GDDR5				X

NOTES:
1. SGPIOC[0:7] is selected according to GPC and MLRC fields in SIUMCR. See **6.14.1.1 SIU Module Configuration Register (SIUMCR)**.

Figure 6-3 illustrates the functionality of the SGPIO.

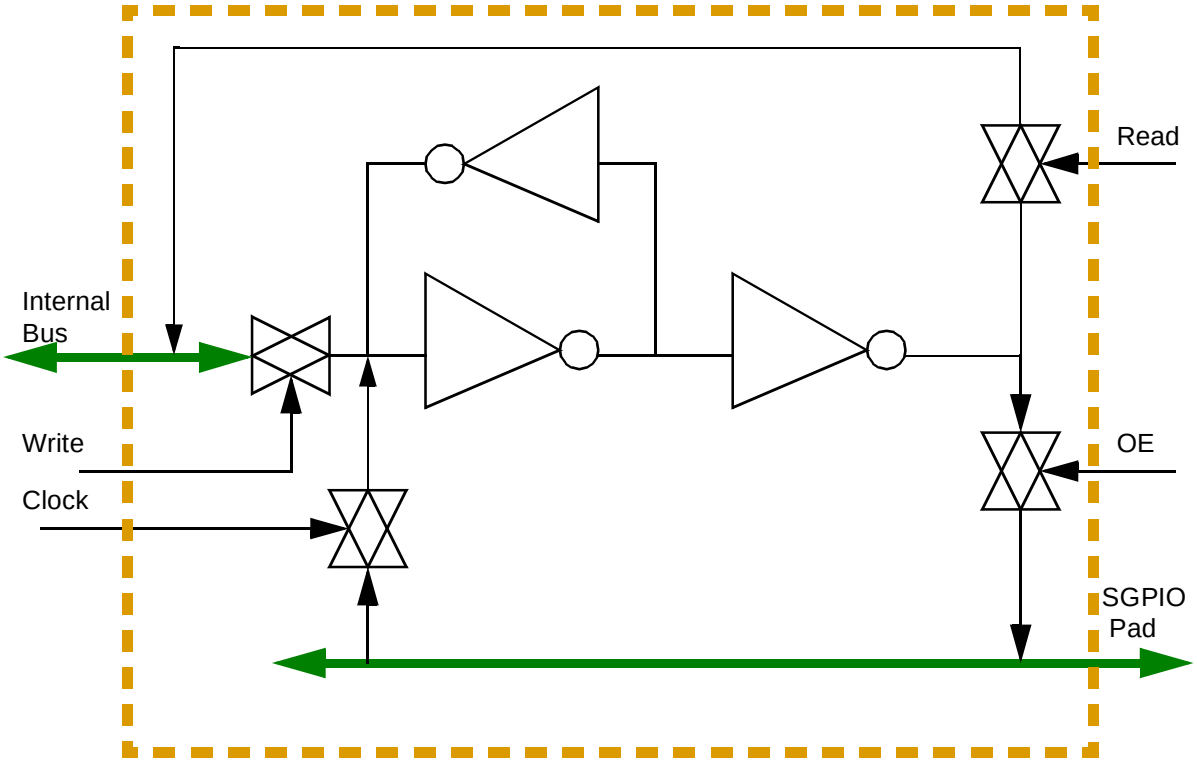


Figure 6-3 SGPIO Cell

6.4 Enhanced Interrupt Controller

6.4.1 Key Features

- Significant interrupt latency reduction
- Simplified interrupt structure
- Up to 48 different interrupt requests
- Splitting of single PowerPC™ external interrupt vector into up to 48 vectors, one for each source
- Automatic lower priority requests masking
- Full backward compatibility with MPC555/MPC556 (enhanced mode is software programmable.)

6.4.2 Interrupt Configuration

An overview of the MPC565 / MPC566 interrupt structure is shown in **Figure 6-4**. The interrupt controller receives interrupts from USIU internal sources, such as PIT, RTC, from the UIMB module (which has its own interrupt controller) or from the IMB3 bus (directly from IMB modules) and from external pins $\overline{\text{IRQ}}$[0:7].

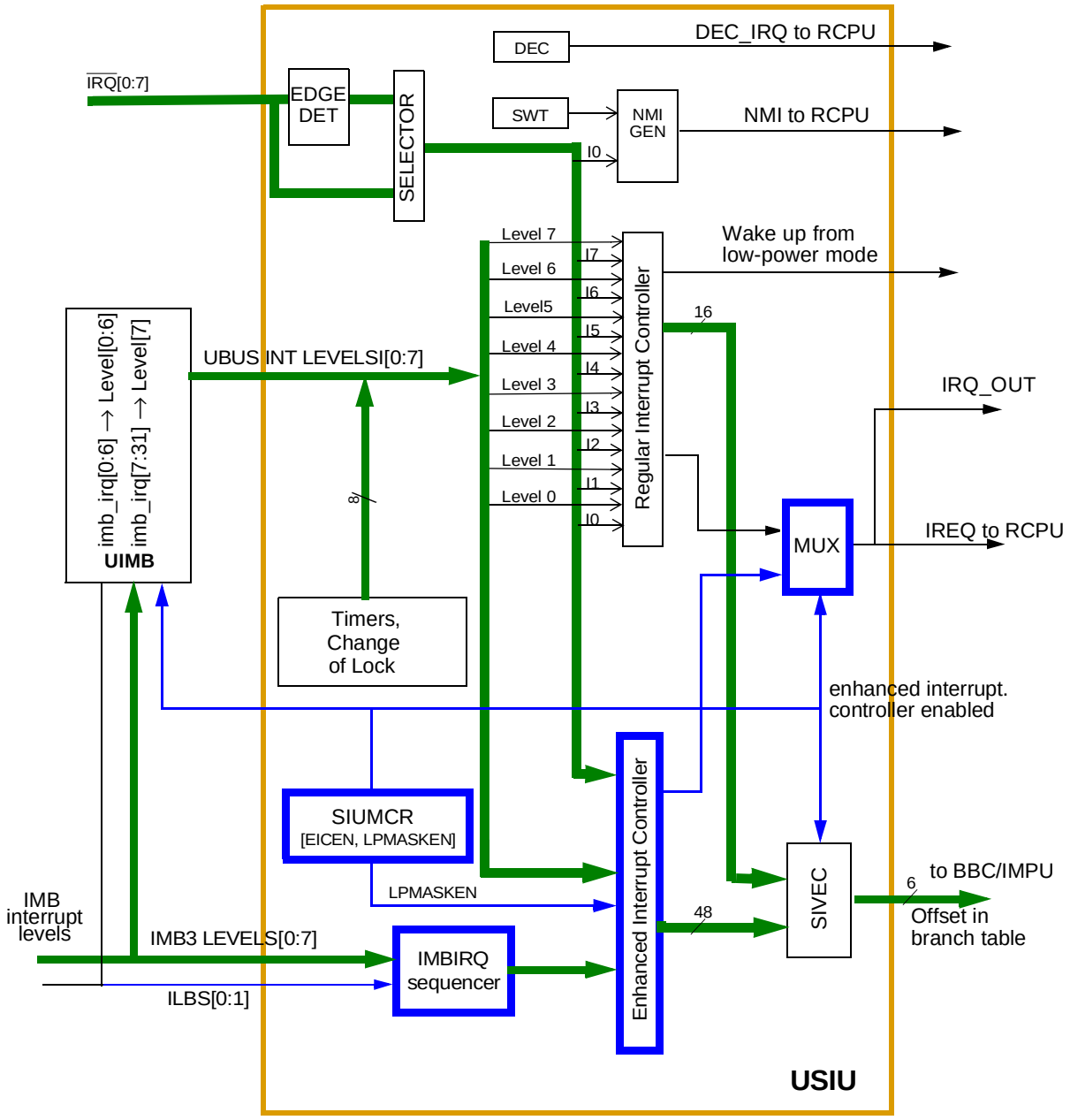


Figure 6-4 MPC565 / MPC566 Interrupt Structure

If it was programmed to generate interrupt, the SWT and external pin $\overline{\text{IRQ}}$[0] always generate an NMI, non-maskable interrupt to the RCPU.

NOTE

The RCPU takes the system reset exception when an NMI is asserted, the external interrupt exception for any other asserted interrupt request and the decrementer exception when the decrementer MSB changes from 0 to 1.

Decrementer interrupt request is not a part of interrupt controller. Each one of the external pins $\overline{IRQ}$[1:7] has its own dedicated assigned priority level. $\overline{IRQ}$[0] is also mapped, but it should be used only as a status bit indicating that $\overline{IRQ}$[0] was asserted and generated NMI interrupt. There are additional eight interrupt priority levels. Each one of the SIU internal interrupt sources, or any of the peripheral module interrupt sources can be assigned by a software to any one of those eight interrupt priority levels. Thus, a very flexible interrupt scheme is implemented. The interrupt request signal generated by the interrupt controller is driven to the PPC core and to IRQ_OUT pin (optionally). This pin may be used in peripheral mode, when the RCPU is disabled, and the internal modules are accessed externally. The IMB interrupts are controlled by the UIMB. The IMB provides 32 interrupt levels, and any interrupt source could be configured to any IMB interrupt level. The UIMB contains 32-bit register that holds the IMB interrupt requests, and drives them on the USIU eight interrupt levels.

NOTE

If one interrupt level was configured to more than one interrupt source, the software should read the UIPEND register in the UIMB module, and the particular status bits in order to identify which interrupt was asserted.

The interrupt controller may be programmed to operate in two modes — a regular mode and an enhanced mode.

6.4.3 Regular Interrupt Controller Operation (MPC555/MPC556 Compatible Mode)

In regular operation mode (default setting) the interrupt controller receives interrupt requests from internal sources, such as timers, PLL lock detector and IMB modules and from external pins IRQ[0:7]. All the internal interrupt sources may be programmed to drive one or more of eight U-bus interrupt level lines while the RCPU, upon receiving an interrupt request, has to read the USIU and UIMB status register in order to determine the interrupt source.

The SIVEC register contains an 8-bit code representing the unmasked interrupt request which has the highest priority level. The priority between all interrupt sources for the regular interrupt controller operation is shown in **Table 6-3**.

Table 6-3 Priority Of interrupt Sources — Regular Operation

Number	Priority Level	Interrupt Source Description	Offset in Branch Table (Hex)	SIVEC Contents (8-ms bit)
0	Highest	$\overline{\text{EXT_IRQ}}$[0]	0x0000	00000000
1	—	Level 0	0x0008	00000100
2	—	EXT_$\overline{\text{IRQ}}$[1]	0x0010	00001000
3	—	Level 1	0x0018	00001100
4	—	EXT_$\overline{\text{IRQ}}$[2]	0x0020	00010000
5	—	Level 2	0x0028	00010100
6	—	EXT_$\overline{\text{IRQ}}$[3]	0x0030	00011000
7	—	Level 3	0x0038	00011100
8	—	EXT_$\overline{\text{IRQ}}$[4]	0x0040	00100000
9	—	Level 4	0x0048	00100100
10	—	EXT_$\overline{\text{IRQ}}$[5]	0x0050	00101000
11	—	Level 5	0x0058	00101100
12	—	EXT_$\overline{\text{IRQ}}$[6]	0x0060	00110000
13	—	Level 6	0x0068	00110100
14	—	EXT_$\overline{\text{IRQ}}$[7]	0x0070	00111000
15	Lowest	Level 7	0x0078	00111100

Each interrupt request from external lines and from USIU internal interrupt sources in the case of its assertion will set a corresponding bit in SIPEND register, whose influence may be neutralized by clearing an appropriate bit in SIMASK register.

6.4.4 Enhanced Interrupt Controller

6.4.4.1 General Operation

The enhanced interrupt controller operation may be turned on by setting a EICEN control bit in SIUMCR register. In this mode the 32 IMB interrupt levels will be latched by USIU using eight IMB interrupt lines and two lines of ILBS via time multiplexing scheme defined by the UIMB module. In addition to the IMB interrupt sources the external interrupts and timer interrupts are available by the same way as in the regular

scheme. In this mode, the UIMB module **does not drive** UBUS interrupt level lines. Each interrupt request will set a corresponding bit in SIPEND2 or SIPEND3 registers, whose influence may be neutralized by clearing an appropriate bit in SIMASK2 or SIMASK3 registers.

The priority logic is provided in order to determine the highest unmasked interrupt request, and interrupt code is generated in SIVEC register. See **Table 6-4**.

Table 6-4 Priority of Interrupt Sources — Enhanced Operation

Number	Priority Level	Interrupt Source Description	Offset in Branch Table (Hex)	SIVEC Contents (8-ms bit)
0	Highest	EXT_IRQ 0	0x0000	00000000
1	—	Level 0	0x0008	00000100
2	—	IMB_IRQ 0	0x0010	00001000
3	—	IMB_IRQ 1	0x0018	00001100
4	—	IMB_IRQ 2	0x0020	00010000
5	—	IMB_IRQ 3	0x0028	00010100
6	—	EXT_IRQ 1	0x0030	00011000
7	—	Level 1	0x0038	00011100
8	—	IMB_IRQ 4	0x0040	00100000
9	—	IMB_IRQ 5	0x0048	00100100
10	—	IMB_IRQ 6	0x0050	00101000
11	—	IMB_IRQ 7	0x0058	00101100
12	—	EXT_IRQ 2	0x0060	00110000
13	—	Level 2	0x0068	00110100
14	—	IMB_IRQ 8	0x0070	00111000
15	—	IMB_IRQ 9	0x0078	00111100
16	—	IMB_IRQ 10	0x0080	01000000
17	—	IMB_IRQ 11	0x0088	01000100
18	—	EXT_IRQ 3	0x0090	01001000
19	—	Level 3	0x0098	01001100
20	—	IMB_IRQ 12	0x00a0	01010000
21	—	IMB_IRQ 13	0x00a8	01010100
22	—	IMB_IRQ 14	0x00b0	01011000
23	—	IMB_IRQ 15	0x00b8	01011100
24	—	EXT_IRQ 4	0x00c0	01100000
25	—	Level 4	0x00c8	01100100
26	—	IMB_IRQ 16	0x00d0	01101000
27	—	IMB_IRQ 17	0x00d8	01101100
28	—	IMB_IRQ 18	0x00e0	01110000
29	—	IMB_IRQ 19	0x00e8	01110100
30	—	EXT_IRQ 5	0x00f0	01111000
31	—	Level 5	0x00f8	01111100
32	—	IMB_IRQ 20	0x0100	10000000

Table 6-4 Priority of Interrupt Sources — Enhanced Operation (Continued)

Number	Priority Level	Interrupt Source Description	Offset in Branch Table (Hex)	SIVEC Contents (8-ms bit)
33	—	IMB_IRQ 21	0x0108	10000100
34	—	IMB_IRQ 22	0x0110	10001000
35	—	IMB_IRQ 23	0x0118	10001100
36	—	EXT_IRQ 6	0x0120	10010000
37	—	Level 6	0x0128	10010100
38	—	IMB_IRQ 24	0x0130	10011000
39	—	IMB_IRQ 25	0x0138	10011100
40	—	IMB_IRQ 26	0x0140	10100000
41	—	IMB_IRQ 27	0x0148	10100100
42	—	EXT_IRQ 7	0x0150	10101000
43	—	Level 7	0x0158	10101100
44	—	IMB_IRQ 28	0x0160	10110000
45	—	IMB_IRQ 29	0x0168	10110100
46	—	IMB_IRQ 30	0x0170	10111000
47	Lowest	IMB_IRQ 31	0x0178	10111100

A value from SIVEC register is supplied internally to the BBC module and can be used for the external interrupt relocation feature as an offset to the branch table start address. Thus a fast way to a specific interrupt source routine is provided without software overhead. The BBCMCR (see **4.8.2.1 BBC Module Configuration Register BBCMCR**) and EIBADR (see **4.8.2.5 External Interrupt Relocation Table Base Address Register — EIBADR**) registers must be programmed to enable this feature in the BBC. Additionally, the SIPEND2 and SIPEND3 registers contain the information about all the interrupt requests that are asserted at a given time, so a software can always read them.

NOTE

In the case of enhanced interrupt controller the SIPEND and SIMASK registers are not used.

6.4.4.2 Lower Priority Request Masking

This feature (if enabled) simplifies the masking of lower priority interrupt requests when a request of certain priority is in service in applications that require interrupt nesting. **The highest (pending) request is also masked by itself.** The masking is accomplished in the following way.

Upon asserting an interrupt request the BBC generates an acknowledge signal to notify the interrupt controller that the request and the branch table offset have been latched. The interrupt controller then sets a bit in SISR (in service register), according to the asserted request and all the other requests whose priority is **lower** than or **equal** to the one that currently in service, become masked. The mask remains set until the SISR bit is cleared by a software (by interrupt handler), writing ‘1’ value to the corre-

sponding bit. The lower priority request masking diagram is presented in the **Figure 6-5**.

The lower priority request masking feature is disabled by $\overline{\text{HRESET}}$ and it may be enabled by setting the LPMASK_EN bit in the SIUMCR register.

NOTES

In regular mode of the interrupt controller the lower priority request masking feature is not possible.

The feature must be activated only together with exception table relocation in the BBC module.

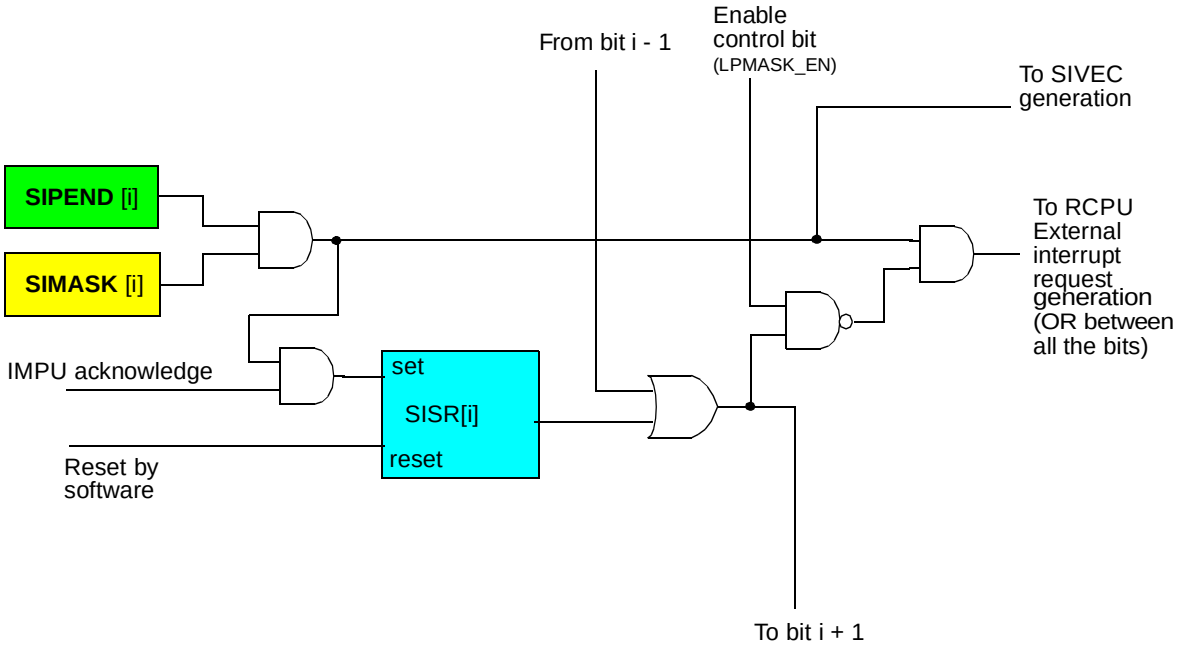


Figure 6-5 Lower Priority Request Masking — One Bit Diagram

6.4.4.3 Backward Compatibility with MPC555/MPC556

The enhanced interrupt controller is a feature that may be enabled according to a user's application using the EICEN control bit in SIUMCR register, which can be set and cleared at any time by a software. If the bit is cleared, the default interrupt controller operation is available, as described in **6.4.3 Regular Interrupt Controller Operation (MPC555/MPC556 Compatible Mode)**. The regular operation is fully compatible with the interrupt controller already implemented in MPC555 / MPC556.

Figure 6-6 illustrates the interrupt controller functionality in the MPC565 / MPC566.

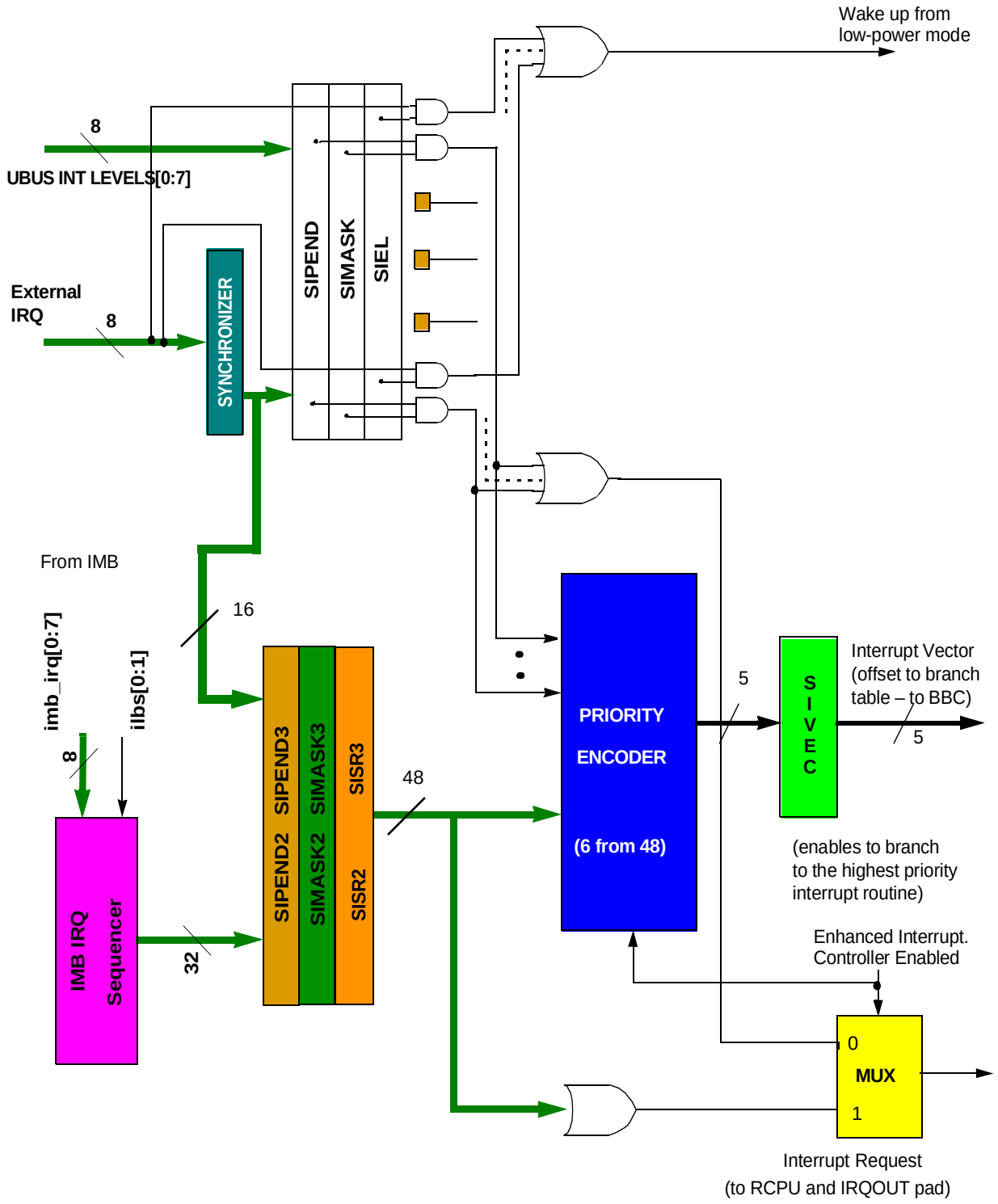


Figure 6-6 MPC565 / MPC566 Interrupt Controller Block Diagram

6.5 Interrupt Overhead Estimation for Enhanced Interrupt Controller Mode

The interrupt overhead includes two main parts:

- Store general, and special purpose registers
- Interrupt source recognitions

The interrupt overhead leads to large latency, and decreases the overall system performance. The registers saving overhead can be reduced by improving the operating system. The number of registers that should be saved, can be reduced when each interrupt event will have it's own interrupt vector. The current solution solves the interrupt source recognition overhead. **Table 6-5** below illustrates the improvements.

Only registers required for the recognition routine are considered to be saved in the calculations. Modules internal events/channels recognition is out of the scope of the calculations. See also typical interrupt handler flowchart in the **Figure 6-7**.

Table 6-5 Interrupt Latency Estimation for Three Typical Cases

	MPC555 / MPC556 Architecture without SIVEC Use	MPC565 / MPC566 Architecture with SIVEC Feature Use	MPC565 / MPC566 Architecture with Enhanced Interrupt Features
Operation Details	Interrupt propagation from request module to RCPU — **8 clocks** Store of some GPR and SPR — **10 clocks** Read SIPEND — **4 clocks** Read SIMASK — **4 clocks** SIPEND data processing — **20 clocks** (find first set, access to LUT in the flash, branches) Read UIPEND — **4 clocks** UIPEND data processing — **20 clocks** (find first set, access to LUT in the flash, branches)	Interrupt propagation from request module to RCPU — **8 clocks** Store of some GPR and SPR — **10 clocks** Read SIVEC — **4 clocks** Branch to routine — **10 clocks** Read UIPEND — **4 clocks** UIPEND data processing — **20 clocks** (find first set, access to LUT in the flash, branches)	Interrupt propagation from request module to RCPU — **6 clocks** Store of some GPR and SPR — **10 clocks** Only one branch should be executed to arrive appropriate interrupt handler belong to its own interrupt requesting device — **2 clocks**
Notes:	If there is a need to enable nesting of interrupts during source recognition procedure, at least **30 clocks** should be added to the interrupt latency estimation	To use this feature in compressed mode some tens of clocks should be added to make a branch to compressed address of the routine	—
Total:	**At Least 70-80 Clocks**	**At Least 50-60 Clocks**	**20 Clocks**

NOTE

Compiler and bus collisions overhead are not included in the calculations.

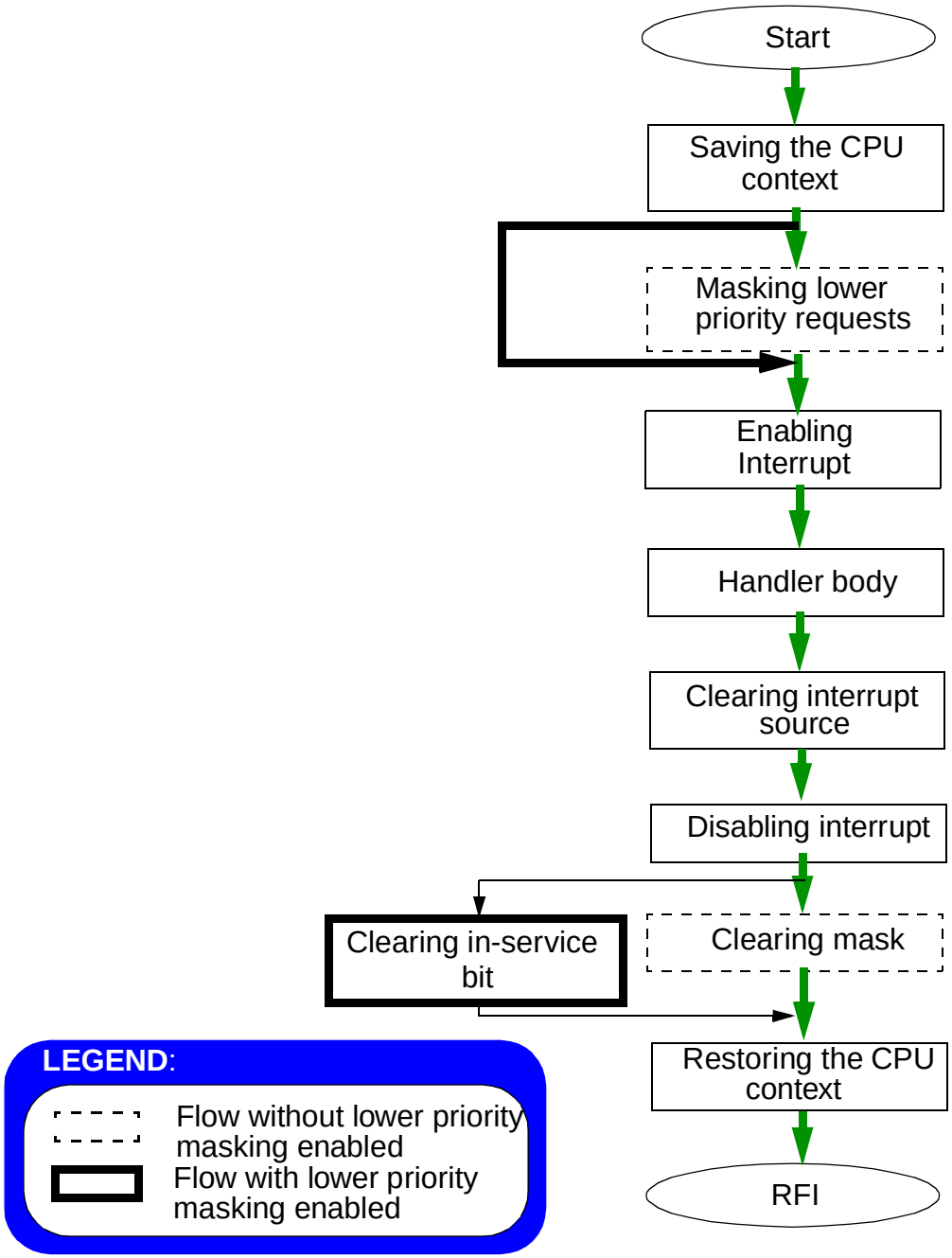


Figure 6-7 Typical Interrupt Handler Routine

6.6 Hardware Bus Monitor

The bus monitor ensures that each bus cycle is terminated within a reasonable period of time. The USIU provides a bus monitor option to monitor internal to external bus accesses on the external bus. The monitor counts from transfer start to transfer

acknowledge and from transfer acknowledge to transfer acknowledge within bursts. If the monitor times out, transfer error acknowledge ($\overline{TEA}$) is asserted internally.

The bus monitor timing bit in the system protection control register (SYPCR) defines the bus monitor time-out period. The programmability of the time-out allows for variation in system peripheral response time. The timing mechanism is clocked by the system clock divided by eight. The maximum value is 2040 system clock cycles.

The bus monitor enable (BME) bit in the SYPCR enables or disables the bus monitor. The bus monitor is always enabled, however, when freeze is asserted or when a debug mode request is pending, regardless of the state of this bit.

6.7 MPC565 / MPC566 Decrementer

The decrementer (DEC) is a 32-bit decrementing counter defined by the MPC565 / MPC566 architecture to provide a decrementer interrupt. This binary counter is clocked by the same frequency as the time base (also defined by the MPC565 / MPC566 architecture). The operation of the time base and decrementer are therefore coherent. In the MPC565 / MPC566, the DEC is clocked by the TMBCLK clock. The decrementer period is computed as follows:

$$T_{dec} = \frac{2^{32}}{F_{tmbclk}}$$

The state of the DEC is not affected by any resets and should be initialized by software. The DEC runs continuously after power-up once the time base is enabled by setting the TBE bit of the TBSCR (see **Table 6-18**) (unless the clock module is programmed to turn off the clock). The decrementer continues counting while reset is asserted.

Loading from the decrementer has no effect on the counter value. Storing to the decrementer replaces the value in the decrementer with the value in the GPR.

Whenever bit zero (the MSB) of the decrementer changes from zero to one, a decrementer exception occurs. If software alters the decrementer such that the content of bit 0 is changed to a value of 1, a decrementer exception occurs.

A decrementer exception causes a decrementer interrupt request to be pending in the RCPU. When the decrementer exception is taken, the decrementer interrupt request is automatically cleared.

Table 6-6 illustrates some of the periods available for the decrementer, assuming a 4-MHz or 20-MHz crystal, and TBS = 0 which selects **TBCLK** division to 4.

Table 6-6 Decrementer Time-Out Periods

Count Value	Time-Out @ 4 MHz	Time-Out @ 20 MHz
0	1.0 µs	0.2 µs
9	10 µs	2.0 µs
99	100 µs	20 µs
999	1.0 ms	200 µs
9999	10.0 ms	2 ms
999999	1.0 s	200 ms
9999999	10.0 s	2.0 s
99999999	100.0 s	20 s
999999999	1000. s	200 s
(hex) FFFFFFFF	4295 s	859 s

Refer to **3.9.5 Decrementer Register (DEC)** for more information.

6.8 MPC565 / MPC566 Time Base (TB)

The time base (TB) is a 64-bit free-running binary counter defined by the MPC565 / MPC566 architecture. The TB has two independent reference registers which can generate a maskable interrupt when the time base counter reaches the value programmed in one of the two reference registers. The period of the TB depends on the driving frequency. In the MPC565 / MPC566, the TB is clocked by the TMBCLK clock. The period for the TB is:

$$T_{TB} = \frac{2^{64}}{F_{tmbclk}}$$

The state of the time base is not affected by any resets and should be initialized by software. Reads and writes of the TB are restricted to special instructions. Separate special-purpose registers are defined in the MPC565 / MPC566 architecture for reading and writing the time base. For the MPC565 / MPC566 implementation, it is not possible to read or write the entire TB in a single instruction. Therefore, the **mttb** and **mftb** instructions are used to move the lower half of the time base (TBL) while the **mttbu** and **mftbu** instructions are used to move the upper half (TBU).

Two reference registers are associated with the time base: TBREF0 and TBREF1. A maskable interrupt is generated when the TB count reaches to the value programmed in one of the two reference registers. Two status bits in the time base control and status register (TBSCR) indicate which one of the two reference registers generated the interrupt.

Refer to **6.14.4 System Timer Registers** for diagrams and bit descriptions of time base registers. Refer to **3.9.4 Time Base Facility (TB) — OEA** and to ***RCPU Reference Manual* (RCPURM/AD)** for additional information regarding the MPC565 / MPC566 time base.

6.9 Real-Time Clock (RTC)

The RTC is a 32-bit counter and pre-divider used to provide a time-of-day indication to the operating system and application software. It is clocked by the *pitrtclk* clock.The counter is not affected by reset and operates in all low-power modes. It is initialized by software. The RTC can be programmed to generate a maskable interrupt when the time value matches the value programmed in its associated alarm register. It can also be programmed to generate an interrupt once a second. A control and status register is used to enable or disable the different functions and to report the interrupt source.

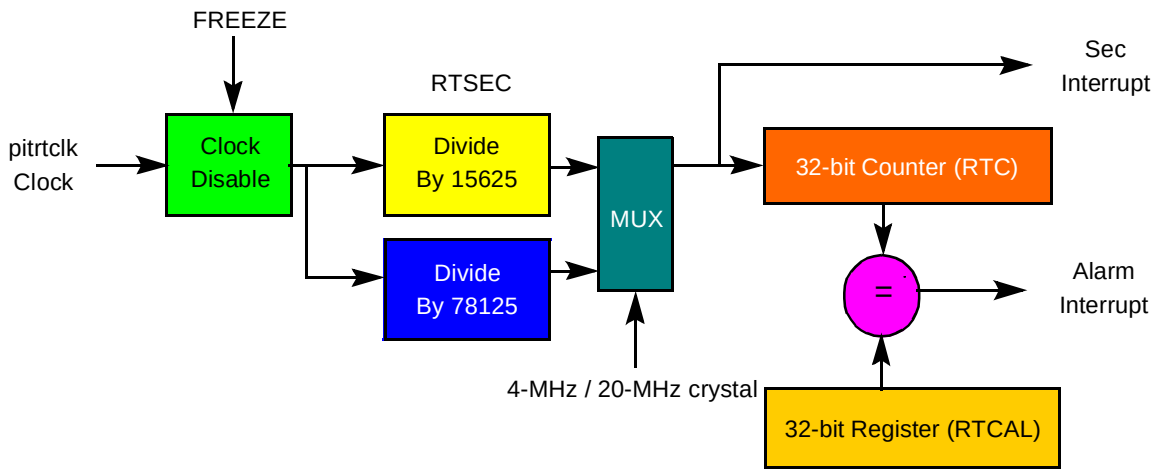


Figure 6-8 RTC Block Diagram

6.10 Periodic Interrupt Timer (PIT)

The periodic interrupt timer consists of a 16-bit counter clocked by the PITRCLK clock supplied by the clock module.

The 16-bit counter counts down to zero when loaded with a value from the PITC. After the timer reaches zero, the PS bit is set and an interrupt is generated if the PIE bit is a logic one. The software service routine should read the PS bit and then write it to zero to terminate the interrupt request. At the next input clock edge, the value in the PITC is loaded into the counter, and the process starts over again.

When a new value is loaded into the PITC, the periodic timer is updated, the divider is reset, and the counter begins counting. If the PS bit is not cleared, an interrupt request is generated. The request remains pending until PS is cleared. If the PS bit is set again prior to being cleared, the interrupt remains pending until PS is cleared.

Any write to the PITC stops the current countdown, and the count resumes with the new value in PITC. If the PTE bit is not set, the PIT is unable to count and retains the old count value. Reads of the PIT have no effect on the counter value.

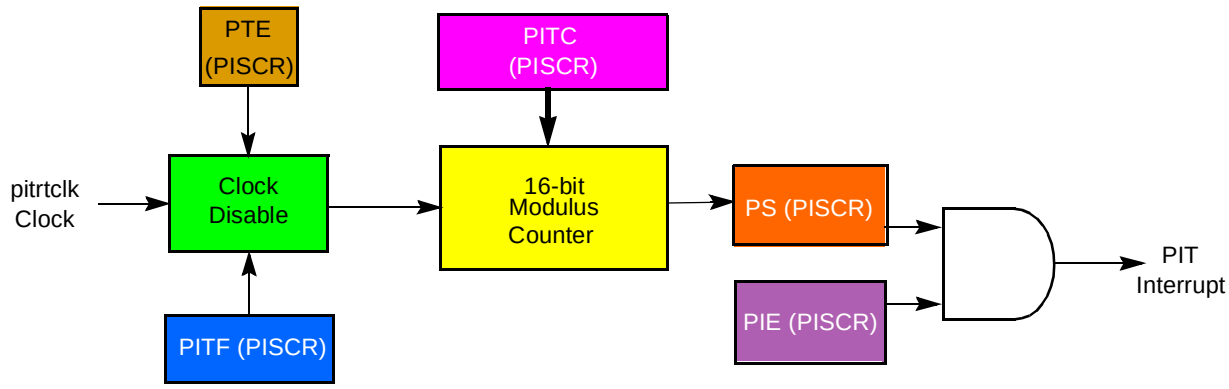


Figure 6-9 PIT Block Diagram

The timeout period is calculated as:

$$PIT_{period} = \frac{PITC + 1}{F_{pitrtclk}} = \frac{PITC + 1}{\left(\frac{ExternalClock}{4or256}\right)}$$

Solving this equation using a 4-MHz external clock and a pre-divider of 256 gives:

$$PITperiod = \frac{PITC + 1}{15625}$$

This gives a range from 64 microseconds, with a PITC of 0x0000, to 4.19 seconds, with a PITC of 0xFFFF. When a 20-MHz crystal is used with a pre-divider of 256, the range is between 12.8 microseconds to 0.84 seconds.

6.11 Software Watchdog Timer (SWT)

The software watchdog timer (SWT) prevents system lockout in case the software becomes trapped in loops with no controlled exit. The SWT is enabled after system reset to cause a system reset if it times out. it. The SWT requires a special service sequence to be executed on a periodic basis. If this periodic servicing action does not occur, the SWT times out and issues a reset or a non-maskable interrupt (NMI), depending on the value of the SWRI bit in the SYPCR.

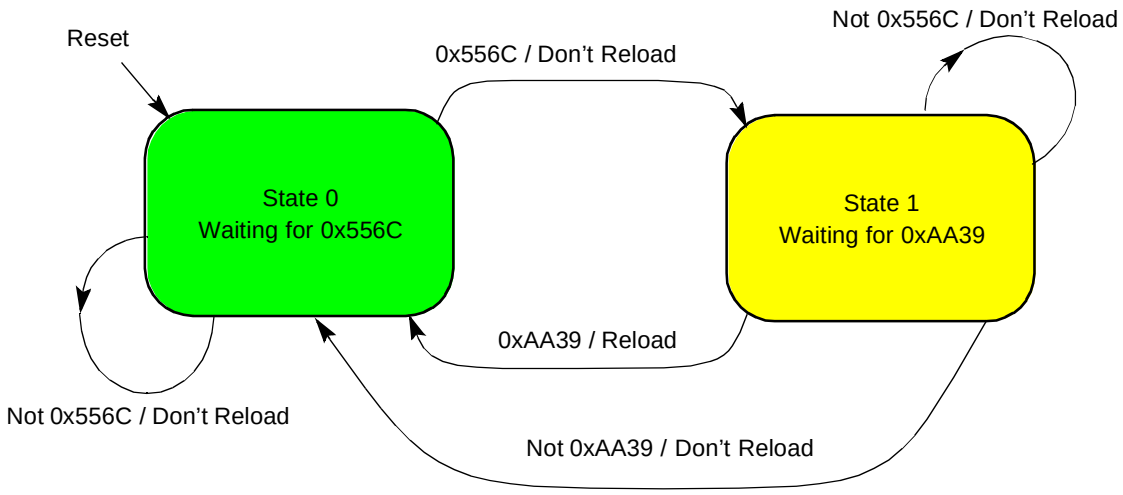
The SWT can be disabled by clearing the SWE bit in the SYPCR. Once the SYPCR is written by software, the state of the SWE bit cannot be changed.

The SWT service sequence consists of the following two steps:

1. Write 0x556C to the software service register (SWSR)
2. Write 0xAA39 to the SWSR

The service sequence clears the watchdog timer and the timing process begins again. If any value other than 0x556C or 0xAA39 is written to the SWSR, the entire sequence must start over.

Although the writes must occur in the correct order prior to time-out, any number of instructions may be executed between the writes. This allows interrupts and exceptions to occur, if necessary, between the two writes.



Although most software disciplines support the watchdog concept, different systems require different time-out periods. For this reason, the software watchdog provides a selectable range for the time-out period.

In **Figure 6-10**, the range is determined by the value SWTC field. The value held in the SWTC field is then loaded into a 16-bit decrementer clocked by the system clock. An additional divide by 2048 prescaler is used if necessary. The decrementer begins counting when loaded with a value from the software watchdog timing count field (SWTC). After the timer reaches 0x0, a software watchdog expiration request is issued to the reset or NMI control logic.

Upon reset, the value in the SWTC is set to the maximum value and is again loaded into the software watchdog register (SWR), starting the process over. When a new value is loaded into the SWTC, the software watchdog timer is not updated until the servicing sequence is written to the SWSR. If the SWE is loaded with the value zero, the modulus counter does not count.

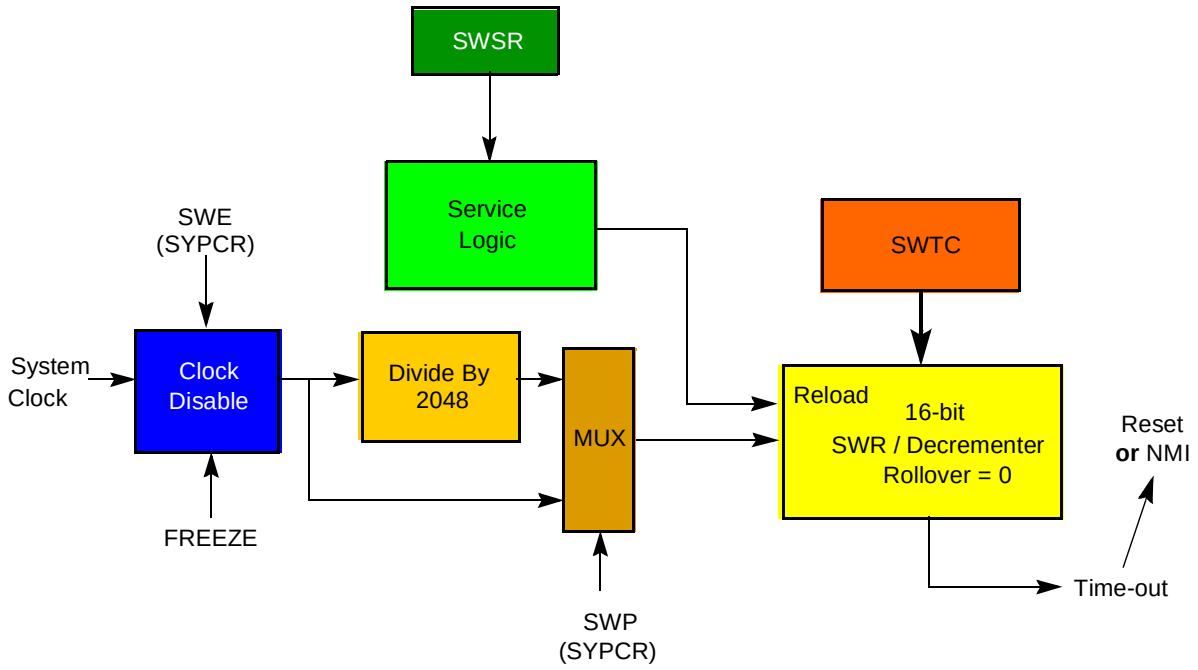


Figure 6-10 SWT Block Diagram

6.12 Freeze Operation

When the FREEZE line is asserted, the clocks to the software watchdog, the periodic interrupt timer, the real-time clock, the time base counter, and the decrementer can be disabled. This is controlled by the associated bits in the control register of each timer. If programmed to stop during FREEZE assertion, the counters maintain their values while FREEZE is asserted, unless changed by the software. The bus monitor, however, remains enabled regardless of this signal.

6.13 Low Power Stop Operation

When the processor is set in a low-power mode (doze, sleep, or deep sleep), the software watchdog timer is frozen. It remains frozen and maintain its count value until the processor exits this state and resumes executing instructions.

The periodic interrupt timer, decrementer, and time base are not affected by these low-power modes. They continue to run at their respective frequencies. These timers are capable of generating an interrupt to bring the MCU out of these low-power modes.

6.14 System Configuration and Protection Registers

This section provides diagrams and bit descriptions of the system configuration and protection registers.

6.14.1 System Configuration Registers

System configuration registers include the SIUMCR, the IMMR and the EMCR registers.

6.14.1.1 SIU Module Configuration Register (SIUMCR)

The SIUMCR contains bits which configure various features in the SIU module. The register contents are shown below.

SIUMCR — SIU Module Configuration Register **0x2F C000**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	EARP			Reserved				DSHW	DBGC		DBPC	ATWC	GPC		DLK
HRESET:															
ID(0)[1]	0	0	0	0	0	0	0	0	ID(9:10)[2]		ID(11)[3]	ID(12)[4]	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Re-served	SC		RCTX	MLRC		Reserved		MTSC	NOSHOW	EICEN	LPMASK _EN	Reserved			
HRESET:															
0	ID(17:18)[5]		0	00		0	0	0	0	0	0	0	0	0	0

NOTES:

1. The HRESET value is a reset configuration word value, extracted from the internal data bus ID(0) line.
2. The HRESET value is a reset configuration word value, extracted from the internal data bus ID(9:10) lines.
3. The HRESET value is a reset configuration word value, extracted from the internal data bus ID(11) line.
4. The HRESET value is a reset configuration word value, extracted from the internal data bus ID(12) line.
5. The HRESET value is a reset configuration word value, extracted from the internal data bus ID(17:18) lines.

WARNING

All SIUMCR fields which are controlled by the reset configuration word should not be changed by software while the corresponding functions are active.

Table 6-7 SIUMCR Bit Descriptions

Bit(s)	Name	Description
0	EARB	External arbitration 0 = Internal arbitration is performed 1 = External arbitration is assumed
1:3	EARP	External arbitration request priority. This field defines the priority of an external master's arbitration request. This field is valid when EARB is cleared. Refer to **9.5.6.4 Internal Bus Arbiter** for details.
4:7	—	Reserved

Table 6-7 SIUMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
8	DSHW	Data show cycles. This bit selects the show cycle mode to be applied to U-bus data cycles (data cycles to IMB modules and flash EEPROM). This field is locked by the DLK bit. Note that instruction show cycles are programmed in the ICTRL and L-bus data show cycles (to SRAM) are programmed in the L2UMCR. 0 = Disable show cycles for all internal data cycles 1 = Show address and data of all internal data cycles
9:10	DBGC	Debug pins configuration. Refer to **Table 6-8**.
11	DBPC	Debug port pins configuration. Refer to **Table 6-9**.
12	ATWC	Address write type enable configuration. This bit configures the pins to function as byte write enables or address types for debugging purposes. 0 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as $\overline{WE}$[0:3]/$\overline{BE}$[0:3][1] 1 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as AT[0:3]
13:14	GPC	This bit configures the pins as shown in **Table 6-10**.
15	DLK	Debug register lock 0 = Normal operation 1 = SIUMCR is locked and can be written only in test mode or when the internal freeze signal is asserted.
16	—	Reserved
17:18	SC	Single-chip select. This field configures the functionality of the address and data buses. Changing the SC field while external accesses are performed is not supported. Refer to **Table 6-11**.
19	RCTX	Reset configuration/timer expired. During reset the RSTCONF/TEXP pin functions as RSTCONF. After reset the pin can be configured to function as TEXP, the timer expired signal that supports the low-power modes. 0 = RSTCONF/TEXP functions as RSTCONF 1 = RSTCONF/TEXP functions as TEXP
20:21	MLRC	Multi-level reservation control. This field selects between the functionality of the reservation logic and IRQ pins, refer to **Table 6-12**.
22:23	—	Reserved
24	MTSC	Memory transfer start control. 0 = $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ functions according to the MLRC bits setting 1 = $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ functions as $\overline{MTS}$
25	NOSHOW	Instruction show cycles disabled — If the NOSHOW bit is set (1), then all instruction show cycles are NOT transmitted to the external bus.
26	EICEN	Enhanced interrupt controller enabled — If the EICEN bit is set (1) then the enhanced interrupt controller operation is enabled. If it is cleared (0), then the regular interrupt controller operation is enabled.
27	LPMASK _EN	Low priority request masking enabled — If the LPMASK_EN bit is set (1) then the mechanism of masking lower priority interrupt requests is enabled. If it is cleared (0), then the lower priority interrupt requests are not masked automatically.
28:31	—	Reserved

NOTES:
1. $\overline{WE}$/$\overline{BE}$ is selected per memory region by WEBS in the appropriate BR register in the memory controller.

Table 6-8 Debug Pins Configuration

DBGC	Pin Function				
	IWP[0:1]/VFLS[0:1]	$\overline{BI}$/$\overline{STS}$	$\overline{BG}$/VF0/LWP1	$\overline{BR}$/VF1/IWP2	$\overline{BB}$/VF2/IWP3
00	VFLS[0:1]	$\overline{BI}$	$\overline{BG}$	$\overline{BR}$	$\overline{BB}$
01	IWP[0:1]	$\overline{STS}$	$\overline{BG}$	$\overline{BR}$	$\overline{BB}$
10	VFLS[0:1]	$\overline{STS}$	VF[0]	VF[1]	VF[2]
11	IWP[0:1]	$\overline{STS}$	LWP[1]	IWP[2]	IWP[3]

Table 6-9 Debug Port Pins Configuration

DBPC	Pin Function		
	TCK/DSCK	TDI/DSDI	TDO/DSDO
0	DSCK	DSDI	DSDO
1	TCK	TDI	TDO

Table 6-10 General Pins Configuration

GPC	Pin Function	
	FRZ/PTR/SGPIOC6	$\overline{IRQOUT}$/LWP0/SGPIOC7
00	PTR	$\overline{LWP0}$
01	SGPIOC[6]	SGPIOC[7]
10	FRZ	LWP[0]
11	FRZ	$\overline{IRQOUT}$

Table 6-11 Single-Chip Select Field Pin Configuration

SC	Pin Function		
	DATA[0:15]/ SGPIOD[0:15]	DATA[16:31] SGPIOD[16:31]	ADDR[8:31]/ SGPIOA[8:31]
00 (multiple chip, 32-bit port size)	DATA[0:15]	DATA[16:31]	ADDR[8:31]
01 (multiple chip, 16-bit port size	DATA[0:15]	SPGIOD[16:31]	ADDR[8:31]
10 (single-chip with address show cycles for debugging)	SPGIOD[0:15]	SPGIOD[16:31]	ADDR[8:31]
11 (single-chip)	SPGIOD[0:15]	SPGIOD[16:31]	SPGIOA[8:31]

Table 6-12 Multi-Level Reservation Control Pin Configuration

MLRC	Pin Function					
	$\overline{IRQ}$[0]/ SGPIOC[0]	**$\overline{IRQ}$[1]/$\overline{RSV}$/ SGPIOC[1]**	**$\overline{IRQ}$[2]/$\overline{CR}$/ SGPIOC[2]/ $\overline{MTS}$**	**$\overline{IRQ3}$/$\overline{KR}$/$\overline{RETRY}$ /SGPIOC3**	**$\overline{IRQ4}$/AT[2]/ SGPIOC[4]**	**$\overline{IRQ}$[5]/ SGPIOC[5]/ MODCK[1][1]**
00	$\overline{IRQ}$[0]	$\overline{IRQ}$[1]	$\overline{IRQ}$[2][2]	$\overline{IRQ}$[3]	$\overline{IRQ}$[4]	$\overline{IRQ}$[5] / MODCK[1]
01	$\overline{IRQ}$[0]	$\overline{RSV}$	$\overline{CR}$[2]	$\overline{KR}$/$\overline{RETRY}$	AT[2]	$\overline{IRQ}$[5] / MODCK[1]
10	SGPIOC0	SGPIOC[1]	SGPIOC[2][2]	SGPIOC[3]	SGPIOC[4]	SGPIOC[5]/ MODCK[1]
11	$\overline{IRQ}$[0]	$\overline{IRQ}$[1]	SGPIOC[2][2]	$\overline{KR}$/$\overline{RETRY}$	AT[2]	SGPIOC[5]/ MODCK[1]

NOTES:
1. Operates as MODCK[1] during reset.
2. This holds if MTSC bit is reset to 0. Otherwise $\overline{IRQ}$[2]/$\overline{CR}$/SGPIOC[2]/$\overline{MTS}$ will function as $\overline{MTS}$.

6.14.1.2 Internal Memory Map Register

The internal memory map register (IMMR) is a special register located within the MPC565 / MPC566 special register space. The IMMR contains identification of a specific device as well as the base for the internal memory map. Based on the value read from this register, software can deduce availability and location of any on-chip system resources.

This register can be read by the **mfspr** instruction. The ISB field can be written by the **mtspr** instruction. The PARTNUM and MASKNUM fields are mask programmed and cannot be changed.

IMMR — Internal Memory Mapping Register **SPR 638**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PARTNUM								MASKNUM							
RESET:															
Read-Only Fixed Value								Read-Only Fixed Value							

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				FLEN	RESERVED		Re-served	RESERVED				ISB			0
RESET:															
0	0	0	0	ID20*	0	0	ID23*	0	0	0	0	ID[28:30]*			0

* The reset value is a reset configuration word value extracted from the indicated bits of the internal data bus. Refer to **7.5.2 Hard Reset Configuration Word**.

Table 6-13 IMMR Bit Descriptions

Bit(s)	Name	Description
0:7	PARTNUM	This read-only field is mask programmed with a code corresponding to the part number of the part on which the SIU is located. It is intended to help factory test and user code which is sensitive to part changes. This changes when the part number changes. For example, it would change if any new module is added, if the size of any memory module is changed. It would not change if the part is changed to fix a bug in an existing module. The MPC565 / MPC566 chip has an ID of 0x33.
8:15	MASKNUM	This read-only field is mask programmed with a code corresponding to the mask number of the part. It is intended to help factory test and user code which is sensitive to part changes.
16:19	—	Reserved
20	FLEN	Flash enable. The default state of FLEN is negated, meaning that the boot is performed from external memory. This bit can be set at reset by through the reset configuration word. 0 = On-chip flash memory is disabled, and all internal cycles to the allocated flash address space are mapped to external memory 1 = On-chip flash memory is enabled
21:22	—	Reserved
23	—	Reserved. This bit should be programmed to 0 at all times.
24:27	—	Reserved
28:30	ISB	This read-write field defines the base address of the internal memory space. The initial value of this field can be configured at reset to one of eight addresses, and then can be changed to any value by software. Internal base addresses are as follows: 000 = 0x0000 0000 001 = 0x0040 0000 010 = 0x0080 0000 011 = 0x00C0 0000 100 = 0x0100 0000 101 = 0x0140 0000 110 = 0x0180 0000 111 = 0x01C0 0000
31	—	Reserved

6.14.1.3 External Master Control Register (EMCR)

The external master control register selects the external master modes and determines the internal bus attributes for external-to-internal accesses.

EMCR — External Master Control Register **0x2F C030**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
PRPM	SLVM	0	SIZE		SUPU	INST	RESERVED		RESV	CONT	0	TRAC	SIZEN	RESERVED	
RESET:															
ID16*	0	0	0	1	0	1	0	0	1	1	0	1	1	0	0

* The reset value is a reset configuration word value, extracted from the indicated internal data bus line. Refer to **7.5.2 Hard Reset Configuration Word**.

Table 6-14 EMCR Bit Descriptions

Bit(s)	Name	Description
0:15	—	Reserved
16	PRPM	Peripheral mode. In this mode, the internal RCPU core is shut off and an alternative master on the external bus can access any internal slave module. The reset value of this bit is determined by the reset configuration word bit 16. The bit can also be written by software. 0 = Normal operation 1 = Peripheral mode operation
17	SLVM	Slave mode (valid only if PRPM = 0). In this mode, an alternative master on the external bus can access any internal slave module while the internal RCPU core is fully operational. If PRPM is set, the value of SLVN is a “don’t care.” 0 = Normal operation 1 = Slave mode
18	—	Reserved
19:20	SIZE	Size attribute. If SIZEN = 1, the SIZE bits controls the internal bus attributes as follows: 00 = Double word (8 bytes) 01 = Word (4 bytes) 10 = Half word (2 bytes) 11 = Byte
21	SUPU	Supervisor/user attribute. SUPU controls the supervisor/user attribute as follows: 0 = Supervisor mode access permitted to all registers 1 = User access permitted to registers designated “user access”
22	INST	Instruction attribute. INST controls the internal bus instruction attribute as follows: 0 = Instruction fetch 1 = Operand or non-CPU access
23:24	—	Reserved
25	RESV	Reservation attribute. RESV controls the internal bus reservation attribute as follows: 0 = Storage reservation cycle 1 = Not a reservation
26	CONT	Control attribute. CONT drives the internal bus control bit attribute as follows: 0 = Access to MPC565 / MPC566 control register, or control cycle access 1 = Access to global address map
27	—	Reserved
28	TRAC	Trace attribute. TRAC controls the internal bus program trace attribute as follows: 0 = Program trace 1 = Not program trace
29	SIZEN	External size enable control bit. SIZEN determines how the internal bus size attribute is driven: 0 = Drive size from external bus signals TSIZE[0:1] 1 = Drive size from SIZE0, SIZE1 in EMCR
30:31	—	Reserved

6.14.2 SIU Interrupt Controller Registers

The SIU interrupt controller contains the following registers: SIPEND, SIPEND2 and SIPEND3 (pending interrupt), SIMASK, SIMASK2 and SIMASK3 registers (mask interrupt), SIEL, SIVEC, SISR2 and SISR3.

SIPEND, SIPEND2 and SIPEND3 are a 32 bit registers. Each bit in the register corresponds to an interrupt request. The bits associated with internal exceptions indicate, if set, that an interrupt service is requested. These bits reflect the status of the internal requesting device, and will be cleared when the appropriate actions are initiated by software in the device itself. Writing to these bits has no effect.

The SIPEND and SIMASK registers are applicable in the regular operation mode while the SIPEND2,3 and SIMASK2,3 registers are implemented for the enhanced operation mode.

The bits associated with the $\overline{IRQ}$ pins have a different behavior depending on the sensitivity defined for them in the SIEL register. When the $\overline{IRQ}$ is defined as a “level” interrupt the corresponding bit behaves similar to the bits associated with internal interrupt sources, (i.e., it reflects the status of the $\overline{IRQ}$ pin). This bit can not be changed by software, it will be cleared when the external signal is negated. When the $\overline{IRQ}$ is defined as an “edge” interrupt, if the corresponding bit is set, it indicates that a falling edge was detected on the line. The bit must be reset by software by writing a ‘1’ to it.

6.14.2.1 SIU Interrupt Pending Register

SIPEND — SIU Interrupt Pending Register **0x2F C010**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ0	LVL0	IRQ1	LVL1	IRQ2	LVL2	IRQ3	LVL3	IRQ4	LVL4	IRQ5	LVL5	IRQ6	LVL6	IRQ7	LVL7
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.14.2.2 SIU Interrupt Pending Register 2

SIPEND2 — SIU Interrupt Pending Register2 **0x2F C040**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ0	LVL0	IMB IRQ0	IMB IRQ1	IMB IRQ2	IMB IRQ3	IRQ1	LVL1	IMB IRQ4	IMB IRQ5	IMB IRQ6	IMB IRQ7	IRQ2	LVL2	IMB IRQ8	IMB IRQ9
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
IMB IRQ10	IMB IRQ11	IRQ3	LVL3	IMB IRQ12	IMB IRQ13	IMB IRQ14	IMB IRQ15	IRQ4	LVL4	IMB IRQ16	IMB IRQ17	IMB IRQ18	IMB IRQ19	IRQ5	LVL5
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.14.2.3 SIU Interrupt Pending Register 3

SIPEND3 — SIU Interrupt Pending Register3 **0x2F C044**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IMB IRQ20	IMB IRQ21	IMB IRQ22	IMB IRQ23	IRQ6	LVL6	IMB IRQ24	IMB IRQ25	IMB IRQ26	IMB IRQ27	IRQ7	LVL7	IMB IRQ28	IMB IRQ29	IMB IRQ30	IMB IRQ31
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
reserved															
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIMASK is a 32 bit read/write register. Each bit in the register corresponds to an interrupt request bit in the SIPEND register.

SIMASK2 is a 32 bit read/write register. Each bit in the register corresponds to an interrupt request bit in the SIPEND2 register.

SIMASK3 is a 32 bit read/write register. Each bit in the register corresponds to an interrupt request bit in the SIPEND3 register.

When the bit is set, it enables the generation of an interrupt request to the CPU core. SIMASK, SIMASK2, SIMASK3 are updated by software and cleared upon reset. It is the responsibility of the software to determine which of the interrupt sources are enabled at a given time.

6.14.2.4 SIU Interrupt Mask Register

SIMASK — SIU Interrupt Mask Register **0x2F C014**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRM0*	LVM0	IRM1	LVM1	IRM2	LVM2	IRM3	LVM3	IRM4	LVM4	IRM5	LVM5	IRM6	LVM6	IRM7	LVM7
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*$\overline{IRQ}$[0] of the SIPEND register is not affected by the setting or clearing of the IRM0 bit of the SIMASK register. $\overline{IRQ}$[0] is a non-maskable interrupt.

6.14.2.5 SIU Interrupt Mask Register 2

SIMASK2 — SIU Interrupt Mask Register2 **0x2F C048**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ0	LVL0	IMB IRQ0	IMB IRQ1	IMB IRQ2	IMB IRQ3	IRQ1	LVL1	IMB IRQ4	IMB IRQ5	IMB IRQ6	IMB IRQ7	IRQ2	LVL2	IMB IRQ8	IMB IRQ9
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
IMB IRQ10	IMB IRQ11	IRQ3	LVL3	IMB IRQ12	IMB IRQ13	IMB IRQ14	IMB IRQ15	IRQ4	LVL4	IMB IRQ16	IMB IRQ17	IMB IRQ18	IMB IRQ19	IRQ5	LVL5
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*IRQ[0] of the SIPEND register is not affected by the setting or clearing of the IRM0 bit of the SIMASK register. IRQ[00 is a non-maskable interrupt.

6.14.2.6 SIU Interrupt Mask Register 3

SIMASK3 — SIU Interrupt Mask Register3 **0x2F C04C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IMB IRQ20	IMB IRQ21	IMB IRQ22	IMB IRQ23	IRQ6	LVL6	IMB IRQ24	IMB IRQ25	IMB IRQ26	IMB IRQ27	IRQ7	LVL7	IMB IRQ28	IMB IRQ29	IMB IRQ30	IMB IRQ31
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.14.2.7 SIU Interrupt Edge Level Register (SIEL)

The SIEL is a 32-bit read/write register. Each pair of bits corresponds to an external interrupt request. The EDx bit, if set, specifies that a falling edge in the corresponding $\overline{IRQ}$ line will be detected as an interrupt request. When the EDx bit is 0, a low logical level in the $\overline{IRQ}$ line will be detected as an interrupt request. The WMx (wake-up mask) bit, if set, indicates that an interrupt request detection in the corresponding line causes the MPC565 / MPC566 to exit low-power mode.

SIEL — SIU Interrupt Edge Level Register **0x2F C018**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ED0	WM0	ED1	WM1	ED2	WM2	ED3	WM3	ED4	WM4	ED5	WM5	ED6	WM6	ED7	WM7
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.14.2.8 SIU Interrupt Vector Register

The SIVEC is a 32-bit read-only register that contains an 8-bit code representing the unmasked interrupt source of the highest priority level. The SIVEC can be read as either a byte, half word, or word. When read as a byte, a branch table can be used in which each entry contains one instruction (branch). When read as a half-word, each entry can contain a full routine of up to 256 instructions. The interrupt code is defined such that its two least significant bits are 0, thus allowing indexing into the table. The two possible ways of the code usage are shown on **Figure 6-11**

SIVEC — SIU Interrupt Vector Register **0x2F C01C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INTERRUPT CODE								RESERVED							
RESET:															
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
INTR:•••

        Save state
        R3 <- @SIVEC
        R4 <-- Base of branch table

        •••

        lbz     RX, R3 (0)# load as byte
        add     RX, RX, R4
        mtspr   CTR, RX
        bctr
```

BASE	b Routine1
BASE + 4	b Routine2
BASE + 8	b Routine3
BASE + C	b Routine4
BASE +10	•
BASE + n	•

```
INTR:•••

        Save state
        R3 <- @SIVEC
        R4 <-- Base of branch table

        •••

        lhz RX, R3 (0) # load as half
        add     RX, RX, R4
        mtspr CTR, RX
        bctr
```

BASE	1st Instruction of Routine1
BASE + 400	1st Instruction of Routine2
BASE + 800	1st Instruction of Routine3
BASE + C00	1st Instruction of Routine4
BASE +1000	•
BASE + n	•

Figure 6-11 Example of SIVEC Register Usage for Interrupt Table Handling

6.14.2.9 Interrupt In-Service Registers

SISR2, SISR3 are 32 bit read/write registers. Each bit in the register corresponds to an interrupt request. The bit is set if:

- there is a pending interrupt request (SIPEND2/3), that is not masked by (SIMASK2/3), and
- the BBC/IMPU acknowledges interrupt request and latches SIVEC value.

Once a bit is set, all the requests with lower or equal priority become masked (i.e., — will not generate any interrupt request to the RCPU) until the bit is cleared by writing “1” by a software. Writing “0” has no effect.

SISR2 — Interrupt In-Service Register 2 **0x2F C050**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRQ0	LVL0	IMB IRQ0	IMB IRQ1	IMB IRQ2	IMB IRQ3	IRQ1	LVL1	IMB IRQ4	IMB IRQ5	IMB IRQ6	IMB IRQ7	IRQ2	LVL2	IMB IRQ8	IMB IRQ9
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
IMB IRQ10	IMB IRQ11	IRQ3	LVL4	IMB IRQ12	IMB IRQ13	IMB IRQ14	IMB IRQ15	IRQ4	LVL4	IMB IRQ16	IMB IRQ17	IMB IRQ18	IMB IRQ19	IRQ5	LVL5
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SISR3 — Interrupt In-Service Register 3 **0x2F C054**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IMB IRQ20	IMB IRQ21	IMB IRQ22	IMB IRQ23	IRQ6	LVL6	IMB IRQ24	IMB IRQ25	IMB IRQ26	IMB IRQ27	IRQ7	LVL7	IMB IRQ28	IMB IRQ29	IMB IRQ30	IMB IRQ31
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.14.3 System Protection Registers

6.14.3.1 System Protection Control Register (SYPCR)

The system protection control register (SYPCR) controls the system monitors, the software watchdog period, and the bus monitor timing. This register can be read at any time, but can be written only once after system reset.

SYPCR — System Protection Control Register **0x2F C004**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SWTC															
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BMT								BME	RESERVED			SWF	SWE	SWRI	SWP
RESET:															
1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1

Table 6-15 SYPCR Bit Descriptions

Bit(s)	Name	Description
0:15	SWTC	Software watchdog timer count. This field contains the count value of the software watchdog timer.
16:23	BMT	Bus monitor timing. This field specifies the time-out period, in eight-system-clock resolution, of the bus monitor.
24	BME	Bus monitor enable 0 = Disable bus monitor 1 = Enable bus monitor
25:27	—	Reserved
28	SWF	Software watchdog freeze 0 = Software watchdog continues to run while FREEZE is asserted 1 = Software watchdog stops while FREEZE is asserted
29	SWE	Software watchdog enable. Software should clear this bit after a system reset to disable the SWT. 0 = Watchdog is disabled 1 = Watchdog is enabled
30	SWRI	Software watchdog reset/interrupt select 0 = Software watchdog time-out causes a non-maskable interrupt to the RCPU 1 = Software watchdog time-out causes a system reset
31	SWP	Software watchdog prescale 0 = Software watchdog timer is not pre-scaled 1 = Software watchdog timer is prescaled by 2048

6.14.3.2 Software Service Register (SWSR)

The SWSR is the location to which the SWT servicing sequence is written. To prevent SWT time-out, a 0x556C followed by 0xAA39 should be written to this register. The SWSR can be written at any time but returns all zeros when read.

SWSR — Software Service Register **0x2F C00E**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SWSR															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-16 SWSR Bit Descriptions

Bit(s)	Name	Description
0:15	SWSR	SWT servicing sequence is written to this register. To prevent SWT time-out, a 0x556C followed by 0xAA39 should be written to this register. The SWSR can be written at any time but returns all zeros when read.

6.14.3.3 Transfer Error Status Register (TESR)

The transfer error status register contains a bit for each exception source generated by a transfer error. A bit set to logic 1 indicates what type of transfer error exception occurred since the last time the bits were cleared by reset or by the normal software status bit-clearing mechanism.

NOTE

These bits may be set due to canceled speculative accesses which do not cause an interrupt. The register has two identical sets of bit fields; one is associated with instruction transfers and the other with data transfers.

TESR — Transfer Error Status Register **0x2F C020**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED		IEXT	IBMT	RESERVED						DEXT	DBM	RESERVED			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-17 TESR Bit Descriptions

Bit(s)	Name	Description
0:17	—	Reserved
18	IEXT	Instruction external transfer error acknowledge. This bit is set if the cycle was terminated by an externally generated $\overline{\text{TEA}}$ signal when an instruction fetch was initiated.

Table 6-17 TESR Bit Descriptions (Continued)

Bit(s)	Name	Description
19	IBMT	Instruction transfer monitor time out. This bit is set if the cycle was terminated by a bus monitor time-out when an instruction fetch was initiated.
20:25	—	Reserved
26	DEXT	Data external transfer error acknowledge. This bit is set if the cycle was terminated by an externally generated $\overline{\text{TEA}}$ signal when a data load or store is requested by an internal master.
27	DBM	Data transfer monitor time out. This bit is set if the cycle was terminated by a bus monitor time-out when a data load or store is requested by an internal master.
28:31	—	Reserved

6.14.4 System Timer Registers

The following sections describe registers associated with the system timers. These facilities are powered by the KAPWR and can preserve their value when the main power supply is off. Refer to **8.3.3 Pre-Divider** for details on the required actions needed in order to guarantee this data retention.

6.14.4.1 Decrementer Register

The 32-bit decrementer register is defined by the PowerPC architecture. The values stored in this register are used by a down counter to cause decrementer exceptions. The decrementer causes an exception whenever bit zero changes from a logic zero to a logic one. A read of this register always returns the current count value from the down counter.

Contents of this register can be read or written to by the **mfspr** or the **mtspr** instruction. The decrementer register is reset by $\overline{\text{PORESET}}$. $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ do not affect this register. The decrementer is powered by standby power and can continue to count when standby power is applied.

Decrementer counts down the time base clock and the counting is enabled by TBE bit in TBCSR register **6.14.4.4 Time Base Control and Status Register**

DEC — Decrementer Register **SPR 22**

MSB 0																															LSB 31
Decrementing Counter																															
PORESET																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HRESET/SRESET: UNCHANGED																															

Refer to **3.9.5 Decrementer Register (DEC)** for more information on this register.

6.14.4.2 Time Base SPRs

The TB is a 64-bit register containing a 64-bit integer that is incremented periodically. There is no automatic initialization of the TB; the system software must perform this

initialization. The contents of the register may be written by the **mttbl** or the **mttbu** instructions, see **3.9.4 Time Base Facility (TB) — OEA**.

Refer to **3.8 PowerPC VEA Register Set — Time Base** and **3.9.4 Time Base Facility (TB) — OEA** for more information on reading and writing the TBU and TBL registers.

TB — Time Base (Reading) **SPR 268, 269**

MSB 0 … 31	32 … LSB 63
TBU	TBL

RESET: UNCHANGED

TB — Time Base (Writing) **SPR 284, 285**

MSB 0 … 31	32 … LSB 63
TBU	TBL

RESET: UNCHANGED

6.14.4.3 Time Base Reference Registers

Two reference registers (TBREF0 and TBREF1) are associated with the lower part of the time base (TBL). Each is a 32-bit read/write register. Upon a match between the contents of TBL and the reference register, a maskable interrupt is generated.

TBREF0 — Time Base Reference Register 0 **0x2F C204**

MSB 0 … LSB 31
TBREF0

RESET:

TBREF1 — Time Base Reference Register 1 **0x2F C208**

MSB 0 … LSB 31
TBREF1

RESET:

6.14.4.4 Time Base Control and Status Register

The TBSCR is 16-bit read/write register. It controls the TB, decrementer count enable, and interrupt generation and is used for reporting the source of the interrupts. The register can be read anytime. A status bit is cleared by writing a one to it. (Writing a zero has no effect.) More than one bit can be cleared at a time.

TBSCR — Time Base Control and Status Register **0x2F C200**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TBIRQ								REFA	REFB	RESERVED		REFAE	REFBE	TBF	TBE
PORESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-18 TBSCR Bit Descriptions

Bit(s)	Name	Description
0:7	TBIRQ	Time base interrupt request. These bits determine the interrupt priority level of the time base. Refer to **6.4 Enhanced Interrupt Controller** for interrupt level encoding.
8	REFA	Reference A (TBREF0) interrupt status. 0 = No match detected 1 = TBREF0 value matches value in TBL
9	REFB	Reference B (TBREF1) interrupt status. 0 = No match detected 1 = TBREF1 value matches value in TBL
10:11	—	Reserved
12	REFAE	Reference A (TBREF0) interrupt enable. If this bit is set, the time base generates an interrupt when the REFA bit is set.
13	REFBE	Reference B (TBREF1) interrupt enable. If this bit is set, the time base generates an interrupt when the REFB bit is set.
14	TBF	Time base freeze. If this bit is set, the time base and decrementer stop while FREEZE is asserted.
15	TBE	Time base enable 0 = Time base and decrementer are disabled 1 = Time base and decrementer are enabled

6.14.4.5 Real-Time Clock Status and Control Register

The RTCSC is used to enable the different RTC functions and to report the source of the interrupts. The register can be read anytime. A status bit is cleared by writing it to a one. (Writing a zero does not affect a status bit's value.) More than one status bit can be cleared at a time. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep-Alive Power Registers Lock Mechanism**.

RTCSC — Real-Time Clock Status and Control Register **0x2F C220**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RTCIRQ								SEC	ALR	Re-served	4M	SIE	ALE	RTF	RTE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	—	0	0	0	—

Table 6-19 RTCSC Bit Descriptions

Bit(s)	Name	Description
0:7	RTCIRQ	Real-time clock interrupt request. Thee bits determine the interrupt priority level of the RTC. Refer to **6.4 Enhanced Interrupt Controller** for interrupt level encoding.
8	SEC	Once per second interrupt. This status bit is set every second. It should be cleared by the software.
9	ALR	Alarm interrupt. This status bit is set when the value of the RTC equals the value programmed in the alarm register.
10	—	Reserved
11	4M	Real-time clock freeze 0 = RTC assumes that it is driven by 20 MHz to generate the seconds pulse. 1 = RTC assumes that it is driven by 4 MHz
12	SIE	Second interrupt enable. If this bit is set, the RTC generates an interrupt when the SEC bit is set.
13	ALE	Alarm interrupt enable. If this bit is set, the RTC generates an interrupt when the ALR bit is set.
14	RTF	Real-time clock freeze. If this bit is set, the RTC stops while FREEZE is asserted.
15	RTE	Real-time clock enable 0 = RTC is disabled 1 = RTC is enabled

6.14.4.6 Real-Time Clock Register (RTC)

The real-time clock register is a 32-bit read write register. It contains the current value of the real-time clock. A write to the RTC resets the seconds timer to zero. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep-Alive Power Registers Lock Mechanism**.

RTC —Real-Time Clock Register **0x2F C224**

MSB 0		LSB 31
	RTC	

RESET: UNCHANGED

6.14.4.7 Real-Time Clock Alarm Register (RTCAL)

The RTCAL is a 32-bit read/write register. When the value of the RTC is equal to the value programmed in the alarm register, a maskable interrupt is generated.

The alarm interrupt will be generated as soon as there is a match between the ALARM field and the corresponding bits in the RTC. The resolution of the alarm is 1 second. This register is locked after RESET. Unlocking is accomplished by writing 0x55CCAA33 to its associated key register. See **8.9.3.2 Keep-Alive Power Registers Lock Mechanism**.

RTCAL — Real-Time Clock Alarm Register **0x2F C22C**

MSB 0		LSB 31
	ALARM	

RESET: UNCHANGED

6.14.4.8 Periodic Interrupt Status and Control Register (PISCR)

The PISCR contains the interrupt request level and the interrupt status bit. It also contains the controls for the 16 bits to be loaded into a modulus counter. This register can be read or written at any time.

PISCR — Periodic Interrupt Status and Control Register **0x2F C240**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIRQ								PS	RESERVED				PIE	PITF	PTE
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-20 PISCR Bit Descriptions

Bit(s)	Name	Description
0:7	PIRQ	Periodic interrupt request. These bits determine the interrupt priority level of the PIT. Refer to **6.4 Enhanced Interrupt Controller** for interrupt level encoding.
8	PS	Periodic interrupt status. This bit is set if the PIT issues an interrupt. The PIT issues an interrupt after the modulus counter counts to zero. PS can be negated by writing a one to it. A write of zero has no affect.
9:12	—	Reserved
13	PIE	Periodic interrupt enable. If this bit is set, the time base generates an interrupt when the PS bit is set.
14	PITF	PIT freeze. If this bit is set, the PIT stops while FREEZE is asserted.
15	PTE	Periodic timer enable 0 = PIT stops counting and maintains current value 1 = PIT continues to decrement

6.14.4.9 Periodic Interrupt Timer Count Register (PITC)

The PITC register contains the 16 bits to be loaded in a modulus counter. This register is readable and writable at any time.

PITC — Periodic Interrupt Timer Count **0x2F C244**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PITC															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-21 PITC Bit Descriptions

Bit(s)	Name	Description
0:15	PITC	Periodic interrupt timing count. This field contains the 16-bit value to be loaded into the modulus counter that is loaded into the periodic timer. This register is readable and writable at any time.
16:31	—	Reserved

6.14.4.10 Periodic Interrupt Timer Register (PITR)

The periodic interrupt register is a read-only register that shows the current value in the periodic interrupt down counter. Read or writing this register does not affect the register.

PITR — Periodic Interrupt Timer Register **0x2F C248**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PIT															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-22 PIT Bit Descriptions

Bit(s)	Name	Description
0:15	PIT	Periodic interrupt timing count — This field contains the current count remaining for the periodic timer. Writes have no effect on this field.
16:31	—	Reserved

6.14.5 General-Purpose I/O Registers

6.14.5.1 SGPIO Data Register 1 (SGPIODT1)

SGPIODT1 — SGPIO Data Register 1 **0x2F C024**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SGPIOD[0:7]								SGPIOD[8:15]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SGPIOD[16:23]								SGPIOD[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-23 SGPIODT1 Bit Descriptions

Bit(s)	Name	Description
0:7	SGPIOD[0:7]	SIU general-purpose I/O Group D[0:7]. This 8-bit register controls the data of general-purpose I/O pins SGPIOD[0:7]. The direction (input or output) of this group of pins is controlled by the GDDR0 bit in the SGPIO control register.
8:15	SGPIOD[8:15]	SIU general-purpose I/O Group D[8:15]. This 8-bit register controls the data of general-purpose I/O pins SGPIOD[8:15]. The direction (input or output) of this group of pins is controlled by the GDDR1 bit in the SGPIO control register.
16:23	SGPIOD[16:23]	SIU general-purpose I/O Group D[16:23]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOD[16:23]. The direction (input or output) of this group of pins is controlled by the GDDR2 bit in the SGPIO control register
24:31	SGPIOD[24:31]	SIU general-purpose I/O Group D[24:31]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOD[24:31]. The direction of SGPIOD[24:31] is controlled by eight dedicated direction control signals SDDRD[24:31]. Each pin in this group can be configured separately as general-purpose input or output.

6.14.5.2 SGPIO Data Register 2 (SGPIODT2)

SGPIODT2 — SGPIO Data Register 2 **0x2F C028**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SGPIOC[0:7]								SGPIOA[8:15]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SGPIOA[16:23]								SGPIOA[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-24 SGPIODT2 Bit Descriptions

Bit(s)	Name	Description
0:7	SGPIOC[0:7]	SIU general-purpose I/O Group C[0:7]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOC[0:7]. The direction of SGPIOC[0:7] is controlled by 8 dedicated direction control signals SDDRC[0:7] in the SGPIO control register. Each pin in this group can be configured separately as general-purpose input or output.
8:15	SGPIOA[8:15]	SIU general-purpose I/O Group A[8:15]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[8:15]. The GDDR3 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.
16:23	SGPIOA [16:23]	SIU general-purpose I/O Group A[16:23]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[16:23]. The GDDR4 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.
24:31	SGPIOA [24:31]]	SIU general-purpose I/O Group A[24:31]. This 8-bit register controls the data of the general-purpose I/O pins SGPIOA[24:31]. The GDDR5 bit in the SGPIO control register configures these pins as a group as general-purpose input or output.

6.14.5.3 SGPIO Control Register (SGPIOCR)

SGPIOCR — SGPIO Control Register **0x2F C02C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SDDRC[0:7]								RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
GDDR 0	GDDR 1	GDDR 2	GDDR 3	GDDR 4	GDDR 5	RESERVED		SDDRD[24:31]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-25 SGPIOCR Bit Descriptions

Bit(s)	Name	Description
0:7	SDDRC[0:7]	SGPIO data direction for SGPIOC[0:7]. Each SDDR bit zero to seven controls the direction of the corresponding SGPIOC pin zero to seven
8:15	—	Reserved
16	GDDR0	Group data direction for SGPIOD[0:7]
17	GDDR1	Group data direction for SGPIOD[8:15]

Table 6-25 SGPIOCR Bit Descriptions (Continued)

Bit(s)	Name	Description
18	GDDR2	Group data direction for SGPIOD[16:23]
19	GDDR3	Group data direction for SGPIOA[8:15]
20	GDDR4	Group data direction for SGPIOA[16:23]
21	GDDR5	Group data direction for SGPIOA[24:31]
22:23	—	Reserved
24:31	SDDRD [24:31]	SGPIO data direction for SGPIOD[24:31]. Each SDDRD bit 24:31 controls the direction of the corresponding SGPIOD pin [24:31].

Table 6-26 describes the bit values for data direction control.

Table 6-26 Data Direction Control

SDDR/GDDR	Operation
0	SGPIO configured as input
1	SGPIO configured as output

SECTION 7
RESET

This section describes the MPC565 / MPC566 reset sources, operation, control, and status.

7.1 Reset Operation

The MPC565 / MPC566 has several inputs to the reset logic which include the following:

- Power-on reset
- External hard reset pin ($\overline{\text{HRESET}}$)
- External soft reset pin ($\overline{\text{SRESET}}$)
- Loss of lock
- On-chip clock switch
- Software watchdog reset
- Checkstop reset
- Debug port hard reset
- Debug port soft reset
- JTAG reset

All of these reset sources are fed into the reset controller. The control logic determines the cause of the reset, synchronizes it if necessary, and resets the appropriate logic modules, depending on the source of the reset. The memory controller, system protection logic, interrupt controller, and parallel I/O pins are initialized only on hard reset. External soft reset initializes internal logic while maintaining system configuration.

The reset status register (RSR) reflects the most recent source to cause a reset.

7.1.1 Power-On Reset

The power-on reset pin, $\overline{\text{PORESET}}$, is an active low input. In a system with power-down low-power mode, this pin should be activated only as a result of a voltage failure in the KAPWR pin. After detecting the assertion of $\overline{\text{PORESET}}$, the MPC565 / MPC566 enters the power-on reset state. During this state the MODCK[1:3] signals determine the oscillator frequency, PLL multiplication factor, and the PITRCLK and TMBCLK clock sources. In addition, the MPC565 / MPC566 asserts the $\overline{\text{SRESET}}$ and $\overline{\text{HRESET}}$ pins.

The $\overline{\text{PORESET}}$ pin should be asserted for a minimum time of 100,000 cycles of clock oscillator after a valid level has been reached on the KAPWR supply. After detecting

the assertion of $\overline{\text{PORESET}}$, the MPC565 / MPC566 remains in the power-on reset state until the last of the following two events occurs:

- The Internal PLL enters the lock state and the system clock is active.
- The $\overline{\text{PORESET}}$ pin is negated.

If the MPC565 / MPC566 is in single-chip mode and limp mode is enabled, the internal PLL is not required to be locked before the chip exits power-on reset.

After exiting the power-on reset state, the MCU continues to drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins for 512 system clock cycles. When the timer expires (after 512 cycles), the configuration is sampled from data bus pins, if required (see **7.5.1 Hard Reset Configuration**) and the MCU stops driving the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins. In addition, the internal MODCK[1:3] values are sampled.

The $\overline{\text{PORESET}}$ pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{\text{PORESET}}$ signal asserts only if the $\overline{\text{PORESET}}$ pin asserts for more than 100 ns.

7.1.2 Hard Reset

$\overline{\text{HRESET}}$ (hard reset) is an active low, bidirectional I/O pin. The MPC565 / MPC566 can detect an external assertion of $\overline{\text{HRESET}}$ only if it occurs while the MCU is not asserting reset.

When the MPC565 / MPC566 detects assertion of the external $\overline{\text{HRESET}}$ pin or a cause to assert the internal $\overline{\text{HRESET}}$ line, is detected the chip starts to drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ for 512 cycles. When the timer expires (after 512 cycles) the configuration is sampled from data pins (refer to **7.5.1 Hard Reset Configuration**) and the chip stops driving the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins. An external pull-up resistor should drive the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ pins high. After detecting the negation of $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$, the MCU waits 16 clock cycles before testing the presence of an external hard or soft reset.

The $\overline{\text{HRESET}}$ pin has a glitch detector to ensure that low spikes of less than 20 ns are rejected. The internal $\overline{\text{HRESET}}$ will be asserted only if $\overline{\text{HRESET}}$ is asserted for more than 100 ns.

The $\overline{\text{HRESET}}$ is an open collector type pin.

7.1.3 Soft Reset

$\overline{\text{SRESET}}$ (soft reset) is an active low, bidirectional I/O pin. The MPC565 / MPC566 can only detect an external assertion of $\overline{\text{SRESET}}$ if it occurs while the MPC565 / MPC566 is not asserting reset.

When the MPC565 / MPC566 detects the assertion of external $\overline{\text{SRESET}}$ or a cause to assert the internal $\overline{\text{SRESET}}$ line, the chip starts to drive the $\overline{\text{SRESET}}$ for 512 cycles. When the timer expires (after 512 cycles) the debug port configuration is sampled from the DSDI and DSCK pins and the chip stops driving the $\overline{\text{SRESET}}$ pin. An external pull-up resistor should drive the $\overline{\text{SRESET}}$ pin high. After the MPC565 / MPC566 detects

the negation of $\overline{\text{SRESET}}$, it waits 16 clock cycles before testing the presence of an external soft reset.

7.1.4 Loss of Lock

If the PLL detects a loss of lock, erroneous external bus operation will occur if synchronous external devices use the MPC565 / MPC566 input clock. Erroneous operation could also occur if devices with a PLL use the MPC565 / MPC566 CLKOUT signal. This source of reset can be optionally asserted if the LOLRE bit in the PLL, low-power, and reset control register (PLPRCR) is set. The enabled PLL loss of lock event generates an internal hard reset sequence. Refer to **SECTION 8 CLOCKS AND POWER CONTROL** for more information on loss of lock.

7.1.5 On-Chip Clock Switch

If the system clocked is switched to the backup clock or switched from backup clock to another clock source an internal hard reset sequence is generated. Refer to **SECTION 8 CLOCKS AND POWER CONTROL**.

7.1.6 Software Watchdog Reset

When the MPC565 / MPC566 software watchdog counts to zero, a software watchdog reset is asserted. The enabled software watchdog event generates an internal hard reset sequence.

7.1.7 Checkstop Reset

When the RCPU enters a checkstop state, and the checkstop reset is enabled (the CSR bit in the PLPRCR is set), a checkstop reset is asserted. The enabled checkstop event generates an internal hard reset sequence. Refer to the ***RCPU Reference Manual* (RCPURM/AD)** for more information.

7.1.8 Debug Port Hard Reset

When the development port receives a hard reset request from the development tool, an internal hard reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. In this case the development tool must reconfigure the debug port. Refer to **SECTION 22 DEVELOPMENT SUPPORT** for more information.

7.1.9 Debug Port Soft Reset

When the development port receives a soft reset request from the development tool, an internal soft reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. In this case the development tool must reconfigure the debug port. Refer to **SECTION 22 DEVELOPMENT SUPPORT** for more information.

7.1.10 JTAG Reset

When the JTAG logic asserts the JTAG soft reset signal, an internal soft reset sequence is generated, see **SECTION 8 CLOCKS AND POWER CONTROL**. Refer

to **SECTION 24 IEEE 1149.1-COMPLIANT INTERFACE (JTAG)** for more information.

7.2 Reset Actions Summary

Table 7-1 summarizes the action taken for each reset.

Table 7-1 Reset Action Taken For Each Reset Cause

Reset Source	Reset Logic and PLL States Reset	System Configuration Reset	Clock Module Reset	$\overline{\text{HRESET}}$ Pin Driven	Debug Port Configuration	Other Internal Logic Reset	$\overline{\text{SRESET}}$ Pin Driven
Power-On Reset	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hard Reset Sources External Hard Reset Loss of Lock On-Chip Clock Switch Illegal Low-Power Mode Software Watchdog Checkstop Debug Port Hard Reset	**No**	Yes	Yes	Yes	Yes	Yes	Yes
Soft Reset Sources External Soft Reset Debug Port Soft Reset JTAG Reset	**No**	**No**	**No**	**No**	Yes	Yes	Yes

7.3 Data Coherency During Reset

The MPC565 / MPC566 supports data coherency and avoids data corruption while reset. If a cycle is to be executed when detecting any $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$ source, then the cycle will either complete or will not start before generating the corresponding reset control signal. There are reset sources, however, when the MPC565 / MPC566 generates an internal reset due to special internal situation where this protection is not supported. See **7.4 Reset Status Register**.

In the case of large operand size (32 or 16 bits) transaction to a smaller port size, the cycle is split into two 16-bit or four 8-bit cycles. In this case, data coherency is assured and data will not be corrupted.

In the case where the core executes an unaligned load/store cycle which is broken down into multiple cycles, data coherency is NOT assured between these cycles (i.e., data could be corrupted).

A contention on the data pins may occur while asserting external reset ($\overline{\text{EXT_RESET}}$) if the data coherency mechanism is required, and thus enables a cycle to complete, while external hardware drives the data for the configuration word. See **Table 7-2** for a description of the required $\overline{\text{EXT_RESET}}$ line source in a system.

Table 7-2 Reset Configuration Word and Data Corruption/Coherency

Reset Driven	Reset to Use for Data Coherency ($\overline{\text{EXT_RESET}}$)	Comments
$\overline{\text{HRESET}}$	$\overline{\text{SRESET}}$	
$\overline{\text{SRESET}}$	$\overline{\text{HRESET}}$	
$\overline{\text{HRESET}}$ & $\overline{\text{SRESET}}$	$\overline{\text{HRESET}}$ || $\overline{\text{SRESET}}$	Provided only one of them is driven into the MPC565 / MPC566 at a time

7.4 Reset Status Register

All of the reset sources are fed into the reset controller. The 16-bit reset status register (RSR) reflects the most recent source, or sources, of reset. (Simultaneous reset requests can cause more than one bit to be set at the same time.) This register contains one bit for each reset source. A bit set to logic one indicates the type of reset that occurred.

Once set, individual bits in the RSR remain set until software clears them. Bits in the RSR can be cleared by writing a one to the bit. A write of zero has no effect on the bit. The register can be read at all times. The reset status register receives its default reset values during power-on reset. The RSR is powered by the KAPWR pin.

RSR — Reset Status Register **0x2F C288**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EHRS	ESRS	LLRS	SWRS	CSRS	DBHRS	DBSRS	JTRS	OCCS	ILBC	GPOR	GHRST	GSRST	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Table 7-3 Reset Status Register Bit Descriptions

Bit(s)	Name	Description
0	EHRS[1]	External hard reset status 0 = No external hard reset has occurred 1 = An external hard reset has occurred
1	ESRS[1]	External soft reset status 0 = No external soft reset has occurred 1 = An external soft reset has occurred
2	LLRS	Loss of lock reset status 0 = No enabled loss-of-lock reset has occurred 1 = An enabled loss-of-lock reset has occurred
3	SWRS	Software watchdog reset status 0 = No software watchdog reset has occurred 1 = A software watchdog reset has occurred
4	CSRS	Checkstop reset status 0 = No enabled checkstop reset has occurred 1 = An enabled checkstop reset has occurred
5	DBHRS	Debug port hard reset status 0 = No debug port hard reset request has occurred 1 = A debug port hard reset request has occurred

Table 7-3 Reset Status Register Bit Descriptions (Continued)

Bit(s)	Name	Description
6	DBSRS	Debug port soft reset status 0 = No debug port soft reset request has occurred 1 = A debug port soft reset request has occurred
7	JTRS	JTAG reset status 0 = No JTAG reset has occurred 1 = A JTAG reset has occurred
8	OCCS	On-chip clock switch 0 = No on-chip clock switch reset has occurred 1 = An on-chip clock switch reset has occurred
9	ILBC	Illegal bit change. This bit is set when the MPC565 / MPC566 changes any of the following bits when they are locked: LPM[0:1], locked by the LPML bit MF[0:11], locked by the MFPDL bit DIVF[0:4], locked by the MFPDL bit
10	GPOR	Glitch detected on $\overline{\text{PORESET}}$ pin. This bit is set when the $\overline{\text{PORESET}}$ pin is asserted for more than TBD ns 0 = No glitch was detected on the $\overline{\text{PORESET}}$ pin 1 = A glitch was detected on the $\overline{\text{PORESET}}$ pin
11	GHRST	Glitch detected on $\overline{\text{HRESET}}$ pin. This bit is set when the $\overline{\text{HRESET}}$ pin is asserted for more than TBD ns 0 = No glitch was detected on the $\overline{\text{HRESET}}$ pin 1 = A glitch was detected on the $\overline{\text{HRESET}}$ pin
12	GSRST	Glitch detected on $\overline{\text{SRESET}}$ pin. If the $\overline{\text{SRESET}}$ pin is asserted for more than TBD ns the GHRST bit will be set. If an internal or external SRESET is generated the $\overline{\text{SRESET}}$ pin is asserted and the GSRST bit will be set. The GSRST bit remains set until software clears it. The GSRST bit can be negated by writing a one to GSRST. A write of zero has no effect on this bit. 0 = No glitch was detected on $\overline{\text{SRESET}}$ pin 1 = A glitch was detected on $\overline{\text{SRESET}}$ pin.
13:15	—	Reserved

NOTES:

1. In the USIU RSR, if both EHRS and ESRS are set, the reset source is internal. The EHRS and ESRS bits in RSR register are set for any internal reset source in addition to external $\overline{\text{HRESET}}$ and external $\overline{\text{SRESET}}$ events. If both internal and external indicator bits are set, then the reset source is internal.

7.5 Reset Configuration

7.5.1 Hard Reset Configuration

When a hard reset event occurs, the MPC565 / MPC566 reconfigures its hardware system as well as the development port configuration The logical value of the bits that determine its initial mode of operation, are sampled from the following:

- The external data bus pins DATA[0:31]
- An internal default constant (0x0000 0000)
- An internal NVM register value (UC3FCFIG)

If at the sampling time $\overline{\text{RSTCONF}}$ is asserted, then the configuration is sampled from the data bus. If $\overline{\text{RSTCONF}}$ is negated and a valid NVM value exists (UC3FCFIG bit $\overline{\text{HC}}$=0), then the configuration is sampled from the NVM register in the UC3F module. If $\overline{\text{RSTCONF}}$ is negated and no valid NVM value exists (UC3FCFIG bit $\overline{\text{HC}}$=1), then

the configuration word is sampled from the internal default. $\overline{HC}$ will be “1” if the internal flash is erased. **Table 7-4** summarizes the reset configuration options.

Table 7-4 Reset Configuration Options

$\overline{RSTCONF}$	$\overline{Has\ Configuration\ (HC)}$	Internal Configuration Word
0	x	DATA[0:31] pins
1	0	NVM flash EEPROM register (UC3FCFIG)
1	1	Internal data word default (0x0000 0000)

If the PRDS control bit in the PDMCR register is set and $\overline{HRESET}$ and $\overline{RSTCONF}$ are asserted, the MPC565 / MPC566 pulls the data bus low with a weak resistor. The user can overwrite this default by driving the appropriate bit high. See **Figure 7-1** for the basic reset configuration scheme.

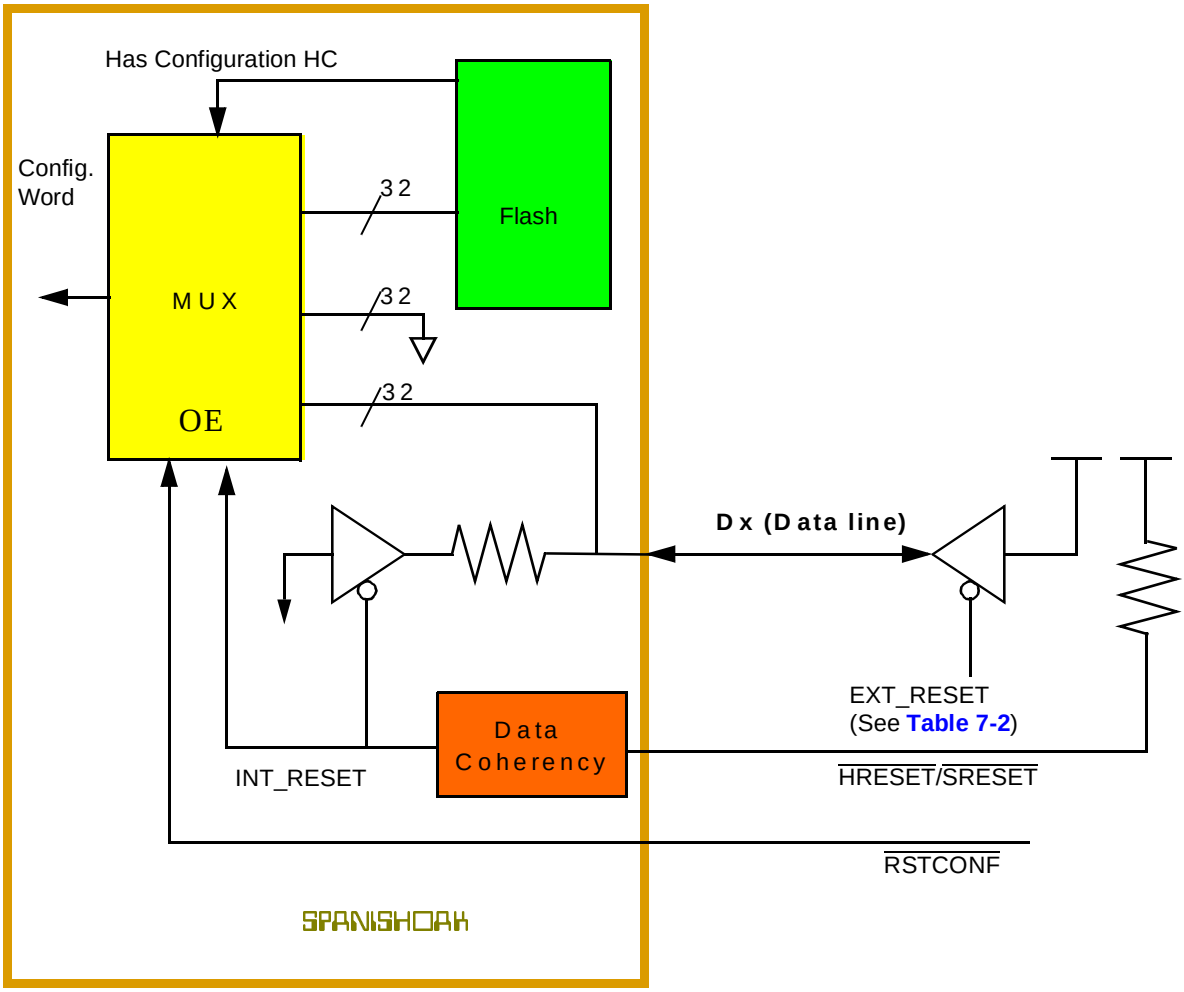


Figure 7-1 Reset Configuration Basic Scheme

During the assertion of the $\overline{\text{PORESET}}$ input signal, the chip assumes the default reset configuration. This assumed configuration changes if the input signal $\overline{\text{RSTCONF}}$ is asserted when the $\overline{\text{PORESET}}$ is negated or the CLKOUT starts to oscillate. To ensure that stable data is sampled, the hardware configuration is sampled every eight clock cycles on the rising edge of CLKOUT with a double buffer. The setup time required for the data bus is approximately 15 cycles, and the maximum rise time of $\overline{\text{HRESET}}$ should be less than six clock cycles. In systems where an external reset configuration word and the TEXP output function are both required, $\overline{\text{RSTCONF}}$ should be asserted until $\overline{\text{SRESET}}$ is negated.

Figure 7-2 to **Figure 7-5** provide sample reset configuration timings.

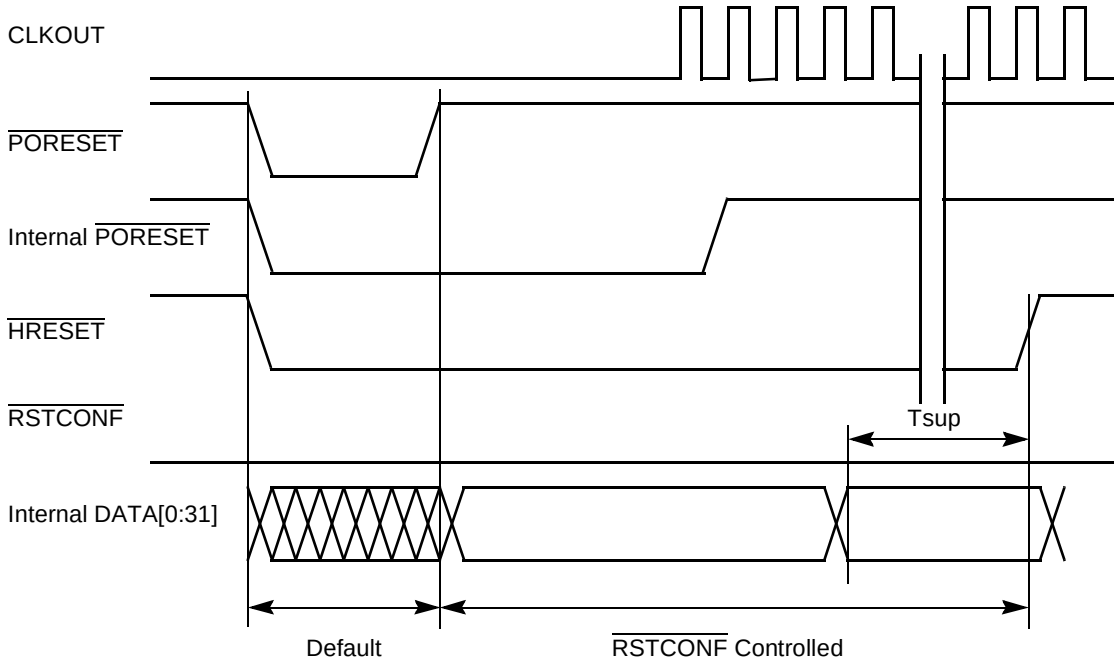


Figure 7-2 Reset Configuration Sampling Scheme For "Short" $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled

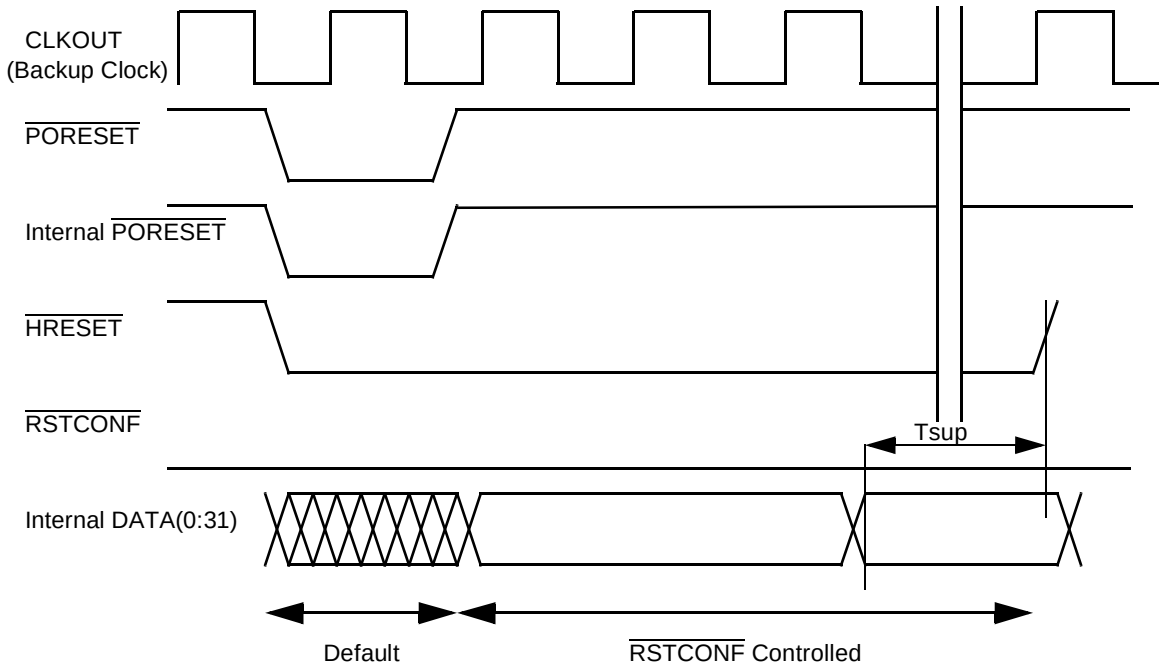


Figure 7-3 Reset Configuration Timing for "Short" $\overline{\text{PORESET}}$ Assertion, Limp Mode Enabled

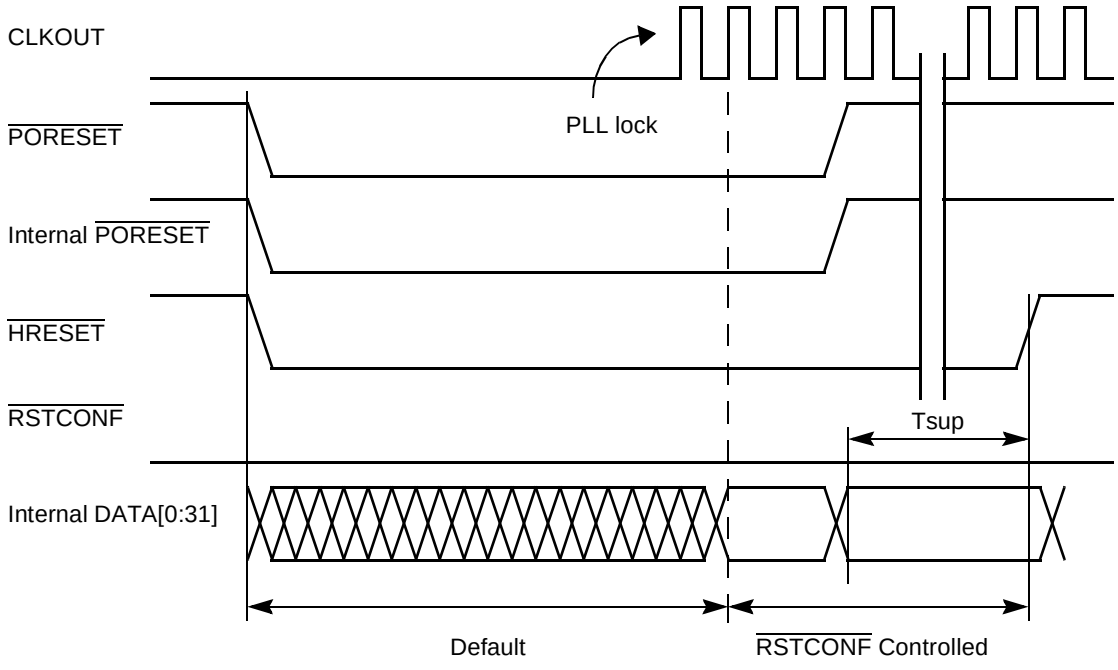


Figure 7-4 Reset Configuration Timing for "Long" $\overline{\text{PORESET}}$ Assertion, Limp Mode Disabled

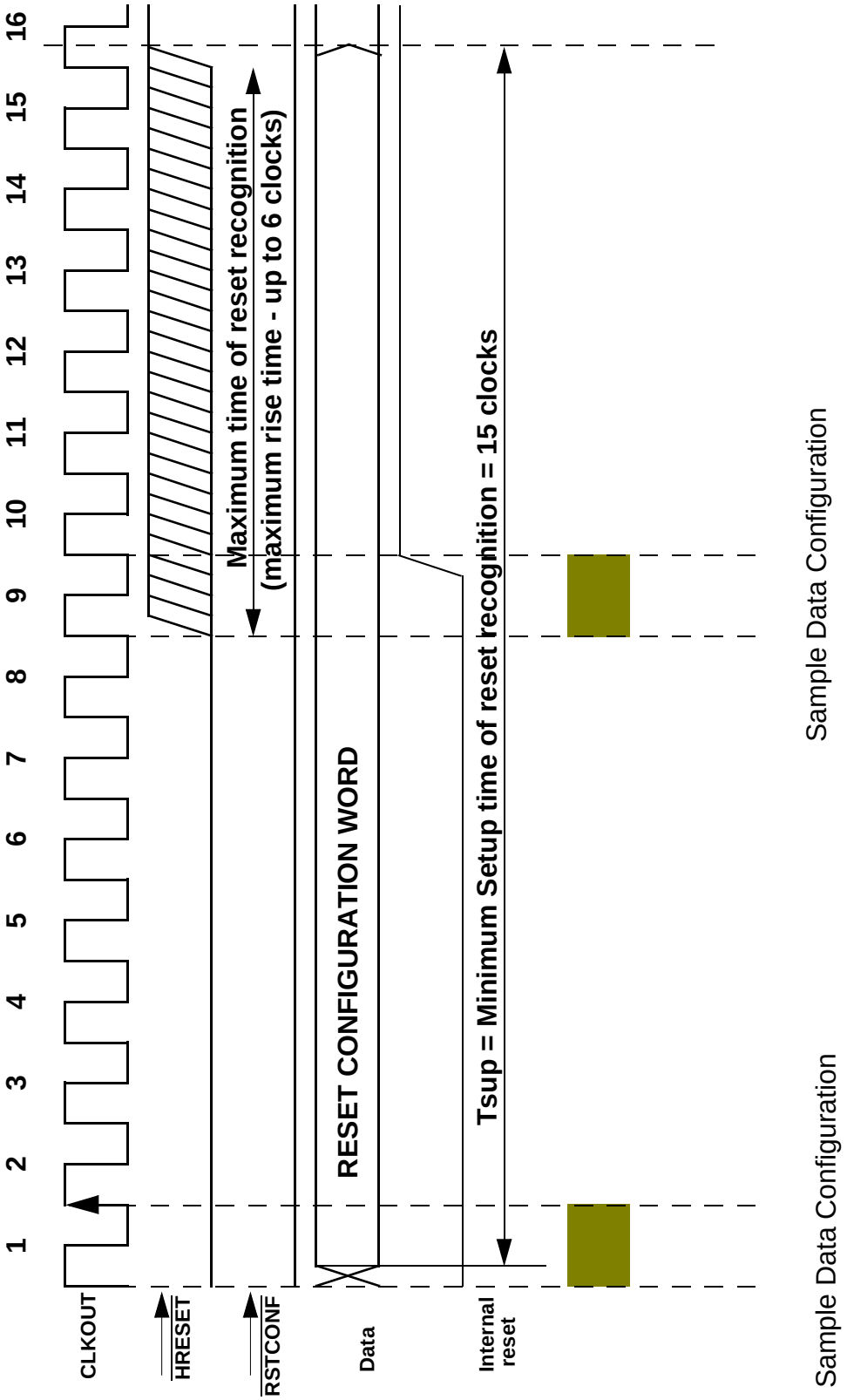


Figure 7-5 Reset Configuration Sampling Timing Requirements

7.5.2 Hard Reset Configuration Word

Following is the hard reset configuration word that is sampled from the internal data bus, data_sgpiod(0:31) on the negation of $\overline{\text{HRESET}}$. The reset configuration word is not a register in the memory map. Most of the bits in the configuration are located in registers in the SIU. The user should refer to **Table 7-5** for a detailed description of each control bit.

RCW— Reset Configuration Word

MSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	IP	BDRV	BDIS	BPS[0:1]		Reserved			DBGC[0:1]		DBPC	ATWC	EBDF[0:1]		Re-served
Reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRPM	SC		ETRE	FLEN	EN_COMP[1]	EXC_COMP[1]	Re-served	OERC		RESERVED		ISB			DME
Reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. Available only on the MPC566.

Table 7-5 RCW Bit Descriptions

Bit(s)	Name	Description
0	EARB	External Arbitration — Refer to **9.5.6 Arbitration Phase** for a detailed description of Bus arbitration. The default value is that internal arbitration hardware is used. 0 = Internal arbitration is performed 1 = External arbitration is assumed
1	IP	Initial Interrupt Prefix — This bit defines the initial value of the MSR_{IP} immediately after reset. The MSR_{IP} bit defines the Interrupt Table location. If IP is zero then the MSR_{IP} initial value is zero, If the IP is one, then the MSR_{IP} initial value is one. Default value is zero. See **Table 3-12** for more information. 0 = MSR[IP] = 0 after reset 1 = MSR[IP] = 1 after reset
2	BDRV	Bus Pins Drive Strength — This bit determines the bus pins (address, data and control) driving capability to be either full or reduced drive. The bus default drive strength is full; Upon default, it also effects the CLKOUT drive strength to be full. See **Table 6-7** for more information. BDRV controls the default state of COM[1] in the SIUMCR. 0 = Full drive 1 = Reduced drive
3	BDIS	Boot Disable — If the BDIS bit is set, then memory controller is not activated after reset. If it is cleared then the memory controller bank 0 is active immediately after reset such that it matches any addresses. If a write to the OR0 register occurs after reset this bit definition is ignored. The default value is that the memory controller is enabled to control the boot with the $\overline{CS}$0 pin. See **10.4 Global (Boot) Chip-Select Operation** for more information. 0 = Memory controller bank 0 is active and matches all addresses immediately after reset 1 = Memory controller is not activated after reset.
4:5	BPS	Boot Port Size — This field defines the port size of the boot device on reset (BR0[PS]). If a write to the OR0 register occurs after reset this field definition is ignored. See**Table 10-3** and **Table 10-7** for more information. 00 = 32-bit port (default) 01 = 8-bit port 10 = 16-bit port 11 = Reserved
6:8	—	Reserved. This bit must not be high in the reset configuration word.
9:10	DBGC[0:1]	Debug Pins Configuration — See **6.14.1.1 SIU Module Configuration Register (SIUMCR)** for this field definition. The default value is that these pins function as: VFLS[0:1], $\overline{BI}$, $\overline{BR}$, $\overline{BG}$ and $\overline{BB}$. See **Table 6-8**.
11	DBPC	Debug Port Pins Configuration — The default value is that these pins function as the debug port pins for development support. See **Table 6-9**.
12	ATWC	Address Type Write Enable Configuration — The default value is that these pins function as $\overline{WE}$ pins. 0 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as $\overline{WE}$[0:3]/$\overline{BE}$[0:3] 1 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as AT[0:3] See **Table 6-7**.
13:14	EBDF	External Bus Division Factor — This field defines the initial value of the external bus frequency. The default value is that CLKOUT frequency is equal to that of the internal clock (no division). See **Table 8-9**.
15	—	Reserved. This bit must not be high in the reset configuration word.
16	PRPM	Peripheral Mode Enable — This bit determines if the chip is in peripheral mode. A detailed description is in **Table 6-14** The default value is no peripheral mode enabled.
17:18	SC	Single Chip Select — This field defines the mode of the MPC565 / MPC566. 00 = Extended chip, 32 bits data 01 = Extended chip, 16 bits data 10 = Single chip and show cycles (address) 11 = Single chip See **Table 6-11**.

Table 7-5 RCW Bit Descriptions (Continued)

Bit(s)	Name	Description
19	ETRE	Exception Table Relocation Enable — This field defines whether the Exception Table Relocation feature in the BBC is enabled or disabled; The default state for this field is disabled. For more details, see **Table 4-4**.
20	FLEN	Flash Enable — This field determines whether the on-chip flash memory is enabled or disabled out of reset. The default state is disabled, which means that by default, the boot is from external memory. Refer to **Table 6-13** for more details. 0 = Flash disabled — boot is from external memory 1 = Flash enabled
21	EN_ COMP[1]	Enable Compression — This bit enables the operation of the MPC565 / MPC566 with compressed code. The default state is disabled. See **Table 4-4**.
22	EXC_ COMP[1]	Exception Compression — This bit determines the operation of the MPC565 / MPC566 with exceptions. If this bit is set, than the MPC565 / MPC566 assumes that ALL the exception routines are in compressed code. The default indicates the exceptions are all non-compressed. See **Table 4-4**.
23	—	Reserved. This bit must not be high in the reset configuration word.
24:25	OERC	Other Exceptions Relocation Control — These bits effect only if ETRE was enabled. Relocation offset: 00 = Offset 0 01 = Offset 64 KB 10 = Offset 512 KB 11 = Offset to 0x003 FE000 (SRAM start address) See **Table 4-2**.
26:27	—	Reserved
28:30	ISB	Internal Space Base Select — This field defines the initial value of the ISB field in the IMMR register. A detailed description is in **Table 6-13**. The default state is that the internal memory map is mapped to start at address 0x0000_0000 hex. This bit must not be high in the reset configuration word.
31	DME	Dual Mapping Enable — This bit determines whether Dual mapping of the internal flash is enabled. For a detailed description refer to **Table 10-9**. The default state is that dual mapping is disabled. 0 = Dual mapping disabled 1 = Dual mapping enabled

NOTES:
1. Available only on the MPC566.

7.5.3 Soft Reset Configuration

When a soft reset event occurs, the MPC565 / MPC566 reconfigures the development port. Refer to **SECTION 22 DEVELOPMENT SUPPORT** for details.

SECTION 8
CLOCKS AND POWER CONTROL

8.1 Overview

The main timing reference for the MPC565 / MPC566 can monitor any of the following:

- A crystal with a frequency of four MHz or 20 MHz
- An external frequency source with a frequency of four MHz
- An external frequency source at the system frequency

The system operating frequency is generated through a programmable phase-locked loop, the system PLL (SPLL). The SPLL runs at twice the system speed. The SPLL is programmable in integer multiples of the input oscillator frequency to generate the internal (VCO/2) operating frequency. A pre-divider before the SPLL enables the division of the high frequency crystal oscillator. The internal operating SPLL frequency should be at least 30 MHz. It can be divided by a power-of-two divider to generate the system operating frequencies.

In addition to the system clock, the clocks submodule provides the following:

- TMBCLK to the time base (TB) and decrementer (DEC)
- PITRTCLK to the periodic interrupt timer (PIT) and PPC real-time clock (RTC)

NOTE

The PPC RTC is separate from the MIOS14 real-time clock sub-module. See **17.15 Real-Time Clock Submodule (MRTCSM)**.

The oscillator, TB, DEC, RTC, and the PIT are powered from the keep alive power supply (KAPWR) pin. This allows the counters to continue to count (increment/decrement) at the oscillator frequency even when the main power to the MCU is off. While the power is off, the PIT may be used to signal to the power supply IC to enable power to the system at specific intervals. This is the power-down wake-up feature. When the chip is not in power-down low-power mode, the KAPWR is powered to the same voltage value as the voltage of the I/O buffers and logic.

The MPC565 / MPC566 clock module consists of the main crystal oscillator, the SPLL, the low-power divider, the clock generator, the system low-power control block, and the limp mode control block. The clock module receives control bits from the system clock control register (SCCR), change of lock interrupt register (COLIR), the low-power and reset-control register (PLPRCR), and the PLL.

Figure 8-1 illustrates the functional block diagram of the clock unit.

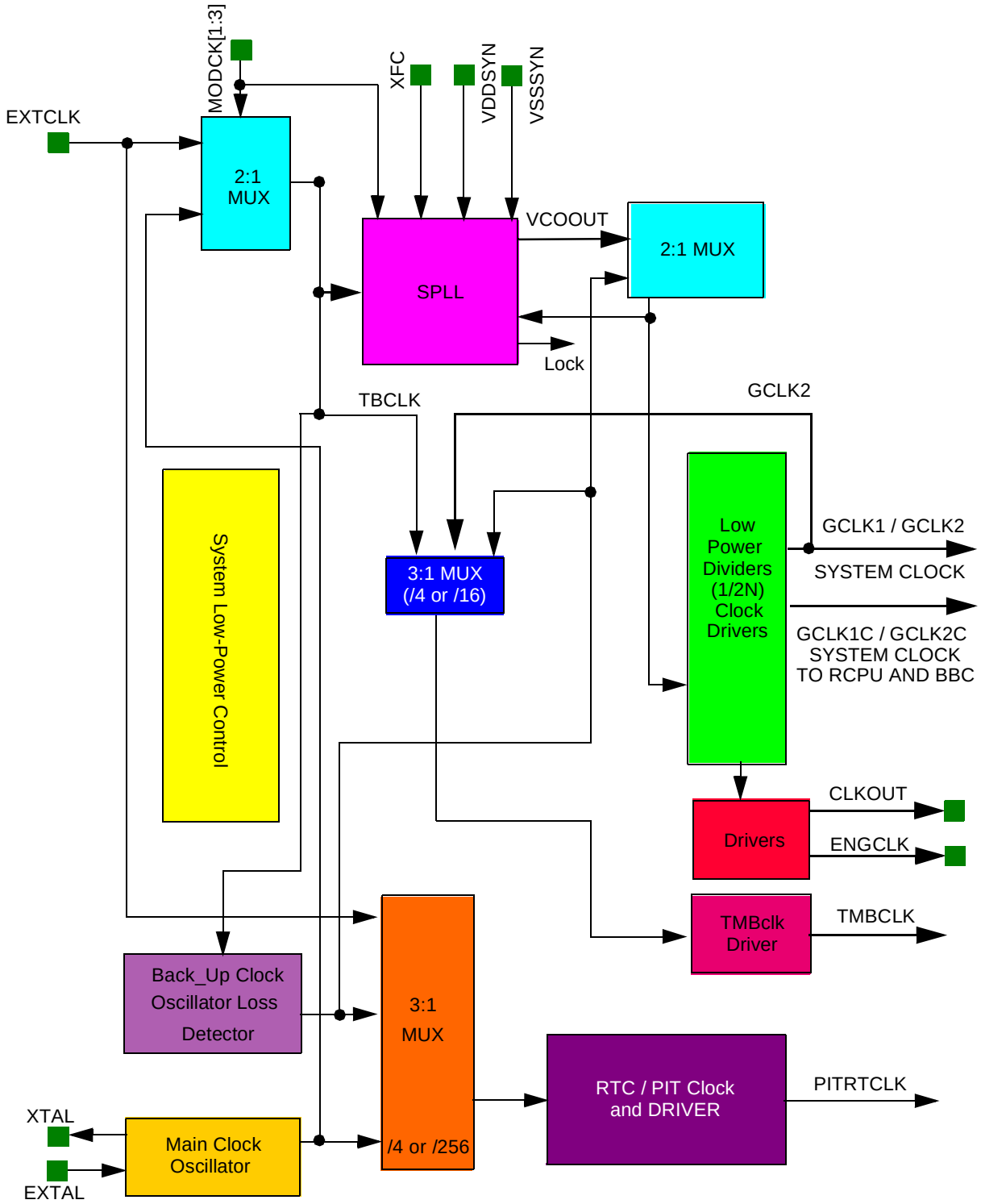


Figure 8-1 Clock Unit Block Diagram

8.2 System Clock Sources

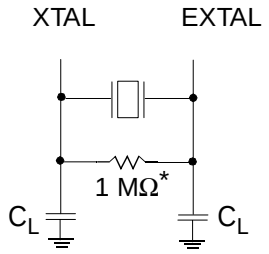
The system clock can be provided by the main system oscillator, an external clock input, or the backup clock (BUCLK) on-chip ring oscillator, see **Figure 8-2**.

The main system oscillator uses either a 4-MHz or 20-MHz crystal to generate the PLL reference clock. When the main system oscillator output is the timing reference to the system PLL, skew elimination between the XTAL/EXTAL pins and CLKOUT is not guaranteed.

The external clock input (EXTCLK pin) can receive a clock signal from an external source. The clock frequency must be either in the range of three-five MHz or at the system frequency of at least 15 MHz (1:1 mode). When the external clock input is the timing reference to the system PLL skew between the EXTCLK pin and the CLKOUT is less than ± 1 ns.

The backup clock on-chip ring oscillator enables the MPC565 / MPC566 to function with a less precise clock. When operating from the backup clock, the MPC565 / MPC566 is in **limp mode**. This enables the system to continue minimum functionality until the system is fixed. The BUCLK frequency is approximately seven MHz (see **APPENDIX E ELECTRICAL CHARACTERISTICS** for the complete frequency range).

For normal operation, at least one clock source (EXTCLK or main system oscillator) must be active. A configuration with both clock sources active is possible as well. At this configuration EXTCLK provides the system clock and main system oscillator provides the PITRTCLK. The input of an unused timing reference (EXTCLK or EXTAL) must be grounded.



*Resistor is not currently required on the board but space should be available for its addition in the future.

Figure 8-2 Main System Oscillator Crystal Configuration

8.3 System PLL

The PLL allows the processor to operate at a high internal clock frequency using a low frequency clock input, a feature which offers two benefits. Lower frequency clock input reduces the overall electromagnetic interference generated by the system, and the ability to oscillate at different frequencies reduces cost by eliminating the need to add an additional oscillator to a system.

The PLL can perform the following functions:

- Frequency multiplication
- Skew elimination

- Frequency division

8.3.1 Frequency Multiplication

The PLL can multiply the input frequency by any integer between one and 4096. The multiplication factor depends on the value of the MF[0:11] bits in the PLPRCR register. While any integer value from one to 4096 can be programmed, the resulting VCO output frequency must be at least 15 MHz. The multiplication factor is set to a predetermined value during power-on reset as defined in **Table 8-1**.

8.3.2 Skew Elimination

The PLL is capable of eliminating the skew between the external clock entering the chip (EXTCLK) and both the internal clock phases and the CLKOUT pin, making it useful for tight synchronous timings. Skew elimination is active only when the PLL is enabled and programmed with a multiplication factor of one or two (MF = 0 or 1). The timing reference to the system PLL is the external clock input (EXTCLK pin).

8.3.3 Pre-Divider

A pre-divider before the phase comparator enables additional system clock resolution when the crystal oscillator frequency is 20 MHz. The division factor is determined by the DIVF[0:4] bits in the PLPRCR.

8.3.4 PLL Block Diagram

As shown in **Figure 8-3**, the reference signal, OSCCLK, goes to the phase comparator. The phase comparator controls the direction (up or down) that the charge pump drives the voltage across the external filter capacitor (XFC). The direction depends on whether the feedback signal phase lags or leads the reference signal. The output of the charge pump drives the VCO. The output frequency of the VCO is divided down and fed back to the phase comparator for comparison with the reference signal, OSCCLK. The MF values, zero to 4095, are mapped to multiplication factors of one to 4096. Note that when the PLL is operating in 1:1 mode (refer to **Table 8-1**), the multiplication factor is one (MF = 0). The PLL output frequency is twice the maximum system frequency. This double frequency is needed to generate GCLK1 and GCLK2 clocks. On power-up, with a 4-MHz or 20-MHz crystal and the default MF settings, VCOOUT will be 40 MHz and the system clock will be 20 MHz. The equation for VCOOUT is shown below:

$$\text{VCOOUT} = \frac{\text{OSCCLK}}{\text{DIVF} + 1} \times (\text{MF} + 1) \times 2$$

NOTE

When operating with the backup clock, the system clock (and CLKOUT) is one-half of the ring oscillator frequency, (i.e., the system clock is approximately seven MHz). The time base and PIT clocks will be twice the system clock frequency.

The PLL maximum lock time is determined by the input clock to the phase detector. The PLL locks within 500 input clock cycles.

NOTE

Upon initial system power up and after KAPWR is lost, an external circuit must assert power on reset ($\overline{\text{PORESET}}$). If limp mode will be enabled during power-on reset, $\overline{\text{PORESET}}$ must be asserted for at least 100,000 cycles of input PLL clock after a valid level has been reached on the KAPWR supply. If limp mode will be disabled, $\overline{\text{PORESET}}$ should be asserted for approximately three µs after a valid level has been reached on the KAPWR supply.

Whenever power-on reset is asserted, the MF bits are set according to **Table 8-1**, and the DFNH and DFNL bits in SCCR are set to the value of 0 (÷1 and 2), respectively.

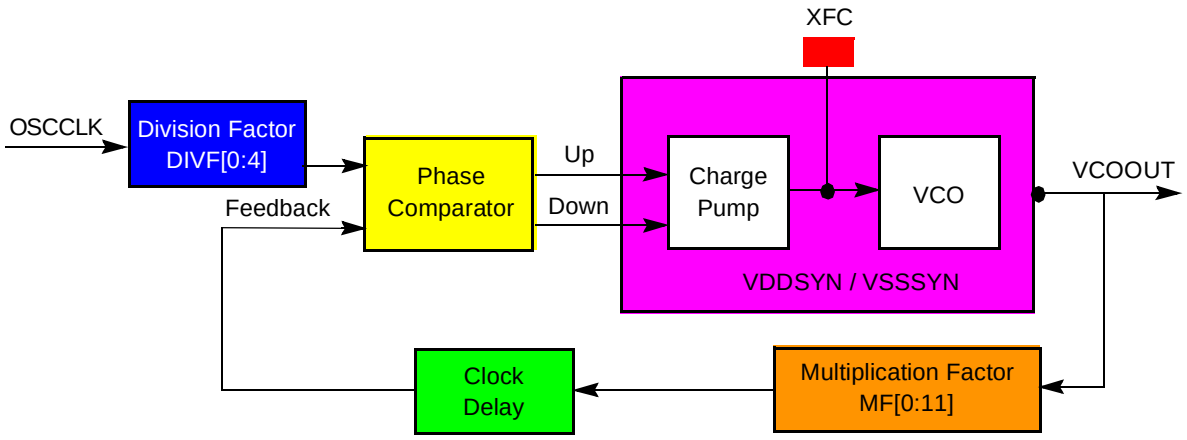


Figure 8-3 System PLL Block Diagram

8.3.5 PLL Pins

The following pins are dedicated to the PLL operation:

- **VDDSYN** — Drain voltage. This is the V_{DD} dedicated to the analog PLL circuits. The voltage should be well-regulated and the pin should be provided with an extremely low impedance path to the V_{DD} power rail. VDDSYN should be bypassed to VSSSYN by a 0.1 µF capacitor located as close as possible to the chip package.
- **VSSSYN** — Source voltage. This is the V_{SS} dedicated to the analog PLL circuits. The pin should be provided with an extremely low impedance path to ground. VSSSYN should be bypassed to VDDSYN by a 0.1 µF capacitor located as close as possible to the chip package.
- **XFC** — External filter capacitor. XFC connects to the off-chip capacitor for the PLL filter. One terminal of the capacitor is connected to XFC, and the other terminal is connected to VDDSYN.

The off-chip capacitor must have the following values (preliminary):

$0 < MF + 1 < 4$	$(1130 \times (MF + 1) - 80)$ pF
$MF + 1 \geq 4$	$2100 \times (MF + 1)$ pF

Where MF = the value stored on MF[0:11]. This is one less than the desired frequency multiplication.

8.4 System Clock During PLL Loss of Lock

At reset, until the SPLL is locked, the SPLL output clock is disabled.

During normal operation (once the PLL has locked), either the oscillator or an external clock source is generating the system clock. In this case, if loss of lock is detected and the LOLRE (loss of lock reset enable) bit in the PLPRCR is cleared, the system clock source continues to function as the PLL's output clock. The USIU timers can operate with the input clock to the PLL, so that these timers are not affected by the PLL loss of lock. Software can use these timers to measure the loss-of-lock period. If the timer reaches the user-preset software criterion, the MPC565 / MPC566 can switch to the backup clock by setting the switch to backup clock (STBUC) bit in the SCCR, provided the limp mode enable (LME) bit in the SCCR is set.

If loss of lock is detected during normal operation, assertion of $\overline{\text{HRESET}}$ (for example, if LOLRE is set) disables the PLL output clock until the lock condition is met. During hard reset, the STBUC bit is set as long as the PLL lock condition is not met and clears when the PLL is locked. If STBUC and LME are both set, the system clock switches to the backup clock, and the chip operates in limp mode until STBUC is cleared.

Every change in the lock status of the PLL can generate a maskable interrupt.

NOTE

When the VCO is the system clock source, chip operation is unpredictable while the PLL is unlocked. Note further that a switch to the backup clock is possible only if the LME bit in the SCCR is set.

8.5 Low-Power Divider

The output of the PLL is sent to a low-power divider block. (In limp mode the BUCLK is sent to a low-power divider block.) This block generates all other clocks in normal operation, but has the ability to divide the output frequency of the VCO before it generates the general system clocks sent to the rest of the MPC565 / MPC566. The PLL VCOOUT is always divided by at least two.

The purpose of the low-power divider block is to allow reduction and restoration of the operating frequencies of different sections of the MPC565 / MPC566 without losing the PLL lock. Using the low-power divider block, full chip operation can still obtained, but at a lower frequency. This is called gear mode. The selection and speed of gear mode can be changed at any time, with changes occurring immediately.

The low-power divider block is controlled in the system clock control register (SCCR). The default state of the low-power divider is to divide all clocks by one. Thus, for a 40-MHz system, the general system clocks are each 40 MHz.

8.6 MPC565 / MPC566 Internal Clock Signals

The internal clocks generated by the clocks module are shown in **Figure 8-4**. The clocks module also generates the CLKOUT and ENGCLK external clock signals. The PLL synchronizes these signals to each other. The PITRTCLK frequency and source are specified by the RTDIV and RTSEL bits in the SCCR. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source and is twice the MPC565 / MPC566 system clock.

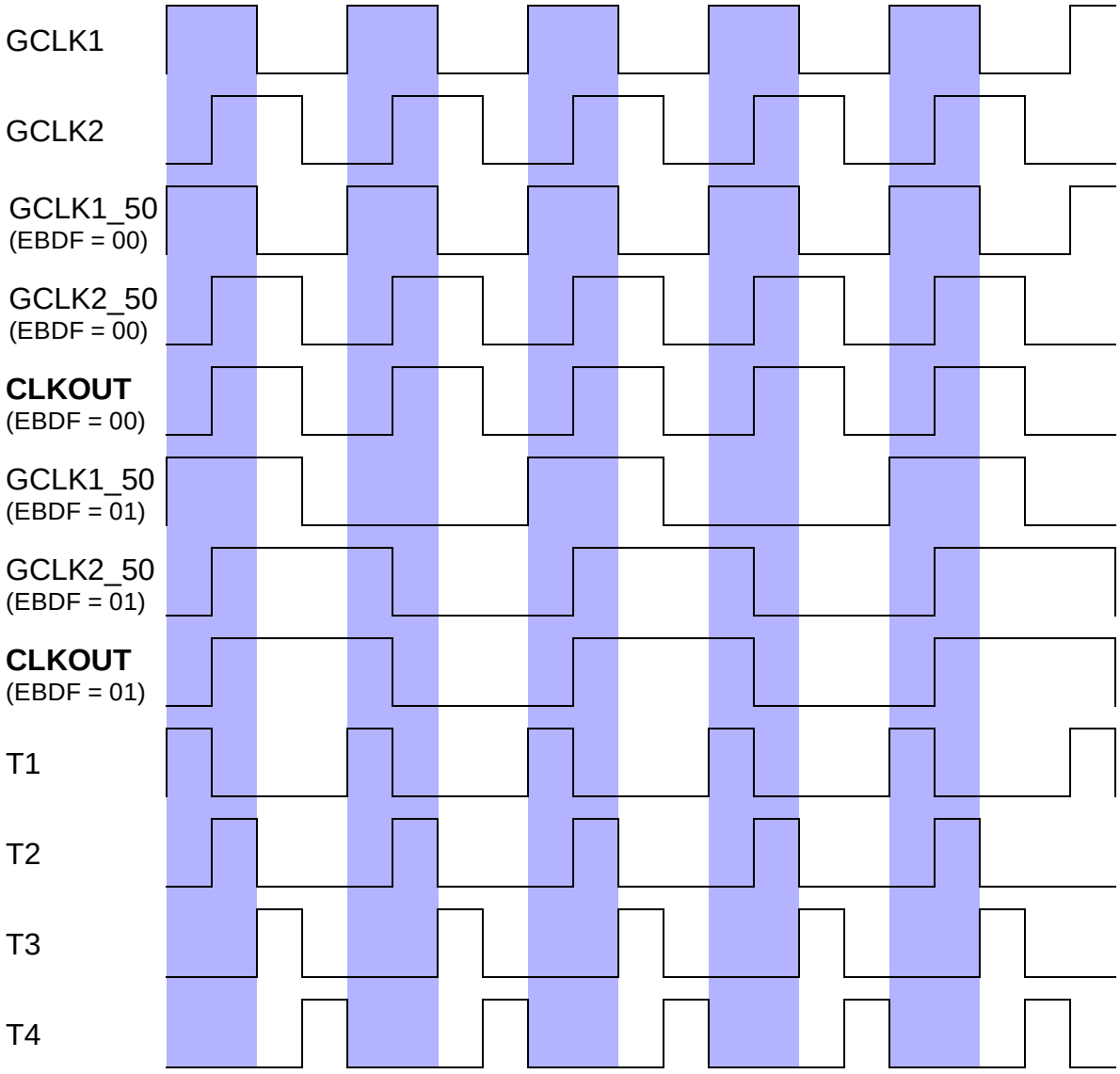


Figure 8-4 MPC565 / MPC566 Clocks

Note that GCLK1_50, GCLK2_50, and CLKOUT can have a lower frequency than GCLK1 and GCLK2. This is to enable the external bus operation at lower frequencies (controlled by EBDF in the SCCR). GCLK2_50 always rises simultaneously with

GCLK2. When DFNH = 0, GCLK2_50 has a 50% duty cycle. With other values of DFNH or DFNL, the duty cycle is less than 50%. Refer to **Figure 8-7**. GCLK1_50 rises simultaneously with GCLK1. When the MPC565 / MPC566 is not in gear mode, the falling edge of GCLK1_50 occurs in the middle of the high phase of GCLK2_50. EBDF determines the division factor between GCLK1/GCLK2 and GCLK1_50/GCLK2_50.

During power-on reset, the MOCCK1, MODCK2, and MODCK3 pins determine the clock source for the PLL and the clock drivers. These pins are latched on the positive edge of $\overline{\text{PORESET}}$. Their values must be stable as long as this line is asserted. The configuration modes are shown in **Table 8-1**. MODCK1 specifies the input source to the SPLL (main system oscillator or EXTCLK). MODCK1, MODCK2, and MODCK3 together determine the multiplication factor at reset and the functionality of limp mode.

If the configuration of PITRTCLK and TMBCLK and the SPLL multiplication factor is to remain unchanged in power-down low-power mode, the MODCK signals should not be sampled at wake-up from this mode. In this case the $\overline{\text{PORESET}}$ pin should remain negated and $\overline{\text{HRESET}}$ should be asserted during the power supply wake-up stage.

When MODCK1 is cleared, the output of the main oscillator is selected as the input to the SPLL. When MODCK1 is asserted, the external clock input (EXTCLK pin) is selected as the input to the SPLL. In all cases, the system clock frequency ($freq_{gclk2}$) can be reduced by the DFNH[0:2] bits in the SCCR. Note that $freq_{gclk2(max)}$ occurs when the DFNH bits are cleared.

The TBS bit in the SCCR selects the time base clock to be either the SPLL input clock or GCLK2. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

The PITRTCLK frequency and source are specified by the RTDIV and RTSEL bits in the SCCR. When the backup clock is functioning as the system clock, the backup clock is automatically selected as the time base clock source.

When the $\overline{\text{PORESET}}$ pin is negated (driven to a high value), the MODCK1, MODCK2, and MODCK3 values are not affected. They remain the same as they were defined during the most recent power-on reset.

Table 8-1 shows the clock configuration modes during power-on reset ($\overline{\text{PORESET}}$ asserted).

NOTE

The MODCK[1:3] are shared functions with IRQ[5:7]. If $\overline{\text{IRQ[5:7]}}$ are used as interrupts, the interrupt source should be removed during $\overline{\text{PORESET}}$ to insure the MODCK pins are in the correct state on the rising edge of $\overline{\text{PORESET}}$.

Table 8-1 Reset Clocks Source Configuration

MODCK[1:3][1]	LME	Default Values after $\overline{\text{PORESET}}$			SPLL Options
		MF + 1	PITCLK Division	TMBCLK Division	
000	0	1	4	4	Used for testing purposes.
001	0	1	256	16	Normal operation, PLL enabled. Main timing reference is crystal osc (20 MHz). Limp mode disabled.
010	1	5	256	4	Normal operation, PLL enabled. Main timing reference is crystal osc (4 MHz). Limp mode enabled.
011	1	1	256	16	Normal operation, PLL enabled. Main timing reference is crystal osc (20 MHz). Limp mode enabled.
100 101	0	1	256	16	Normal operation, PLL enabled. 1:1 Mode Main timing reference is EXTCLK pin (>15MHz) Limp mode disabled.
110	0	5	256	4	Normal operation, PLL enabled. Main timing reference is EXTCLK (3-5 MHz). Limp mode disabled.
111	1	1	256	16	Normal operation, PLL enabled. 1:1 Mode Main timing reference is EXTCLK pin (>15MHz) Limp mode enabled.

NOTES:
1. indicates MODCK pins value during Power On Reset

NOTE

The reset value of the PLL pre-divider is one.

The values of the PITRTCLK clock division and TMBCLK clock division can be changed by software. The RTDIV bit value in the SCCR register defines the division of PITRTCLK. All possible combinations of the TMBCLK divisions are listed in **Table 8-2**.

Table 8-2 TMBCLK Divisions

SCCR[TBS]	MF + 1	TMBCLK Division
1	—	16
0	1, 2	16
0	> 2	4

8.6.1 General System Clocks

The general system clocks (GCLK1C, GCLK2C, GCLK1, GCLK2, GCLK1_50, and GCLK2_50) are the basic clock supplied to all modules and sub-modules on the MPC565 / MPC566. GCLK1C and GCLK2C are supplied to the RCPU and to the BBC. GCLK1C and GCLK2C are stopped when the chip enters the doze-low power mode. GCLK1 and GCLK2 are supplied to the SIU and the clock module. The external bus

clock GCLK2_50 is the same as CLKOUT. The general system clock defaults to VCO/2 = 20 MHz (assuming a 20-MHz system frequency) with default power-on reset MF values.

The general system clock frequency can be switched between different values. The highest operational frequency can be achieved when the system clock frequency is determined by DFNH (CSRC bit in the PLPRCR is cleared) and DFNH = 0 (division by one). The general system clock can be operated at a low frequency (gear mode) or a high frequency. The DFNL bits in SCCR define the low frequency. The DFNH bits in SCCR define the high frequency.

The frequency of the general system clock can be changed dynamically with the system clock control register (SCCR), as shown in **Figure 8-5**.

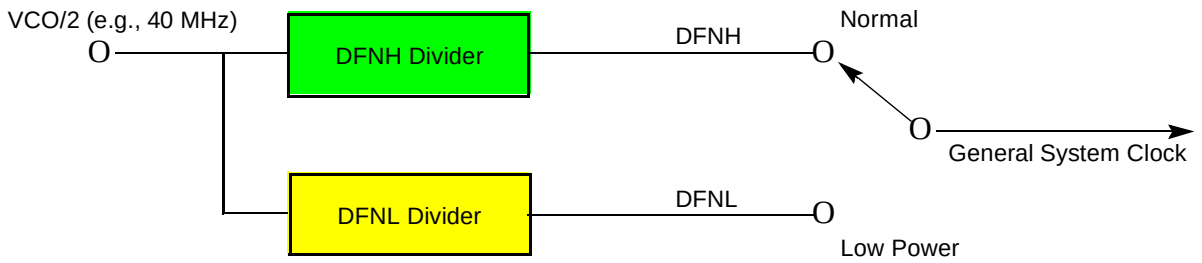


Figure 8-5 General System Clocks Select

The frequency of the general system clock can be changed “on the fly” by software. The user may simply cause the general system clock to switch to its low frequency. However, in some applications, there is a need for a high frequency during certain periods. Interrupt routines, for example, may require more performance than the low frequency operation provides, but must consume less power than in maximum frequency operation. The MPC565 / MPC566 provides a method to automatically switch between low and high frequency operation whenever one of the following conditions exists:

- There is a pending interrupt from the interrupt controller. This option is maskable by the PRQEN bit in the SCCR.
- The (POW) bit in the MSR is clear in normal operation. This option is maskable by the PRQEN bit in the SCCR.

When neither of these conditions exists and the CSRC bit in PLPRCR is set, the general system clock switches automatically back to the low frequency.

Abrupt changes in the divide ratio can cause linear changes in the operating currents of the MPC565 / MPC566. Insure that the proper power supply filtering is available to handle this change instantaneously.

When the general system clock is divided, its duty cycle is changed. One phase remains the same (e.g., 12.5 ns @ 40 MHZ) while the other become longer.

NOTE

CLKOUT does not have a 50% duty cycle when the general system clock is divided. The CLKOUT wave form is the same as that of GCLK2_50.

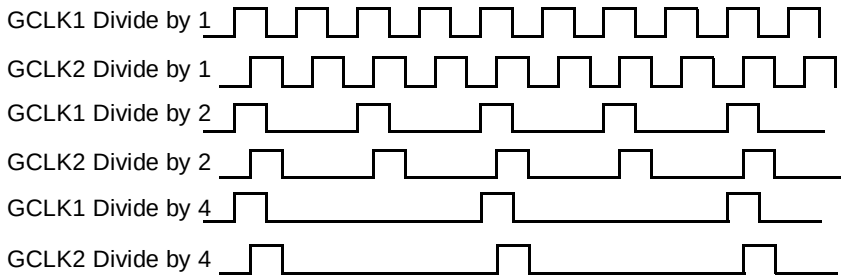


Figure 8-6 Divided System Clocks Timing Diagram

The system clocks GCLK1 and GCLK2 frequency is:

$$FREQ_{sys} = \frac{FREQsysmax}{(2^{DFNH})or(2^{DFNL+1})}$$

where FREQsysmax = VCOOUT/2

Therefore, the complete equation for determining the system clock frequency is:

$$\text{System Frequency} = \frac{OSCCLK}{DIVF + 1} \times \frac{(MF + 1)}{(2^{DFNH} \text{ or } 2^{DFNL+1})} \times \frac{2}{2}$$

The clocks GCLK1_50 and GCLK2_50 frequency is:

$$FREQ_{50} = \frac{FREQsysmax}{(2^{DFNH})or(2^{DFNL+1})} \times \frac{1}{EBDF + 1}$$

Figure 8-7 shows the timing of USIU clocks when DFNH = 1 or DFNL = 0.

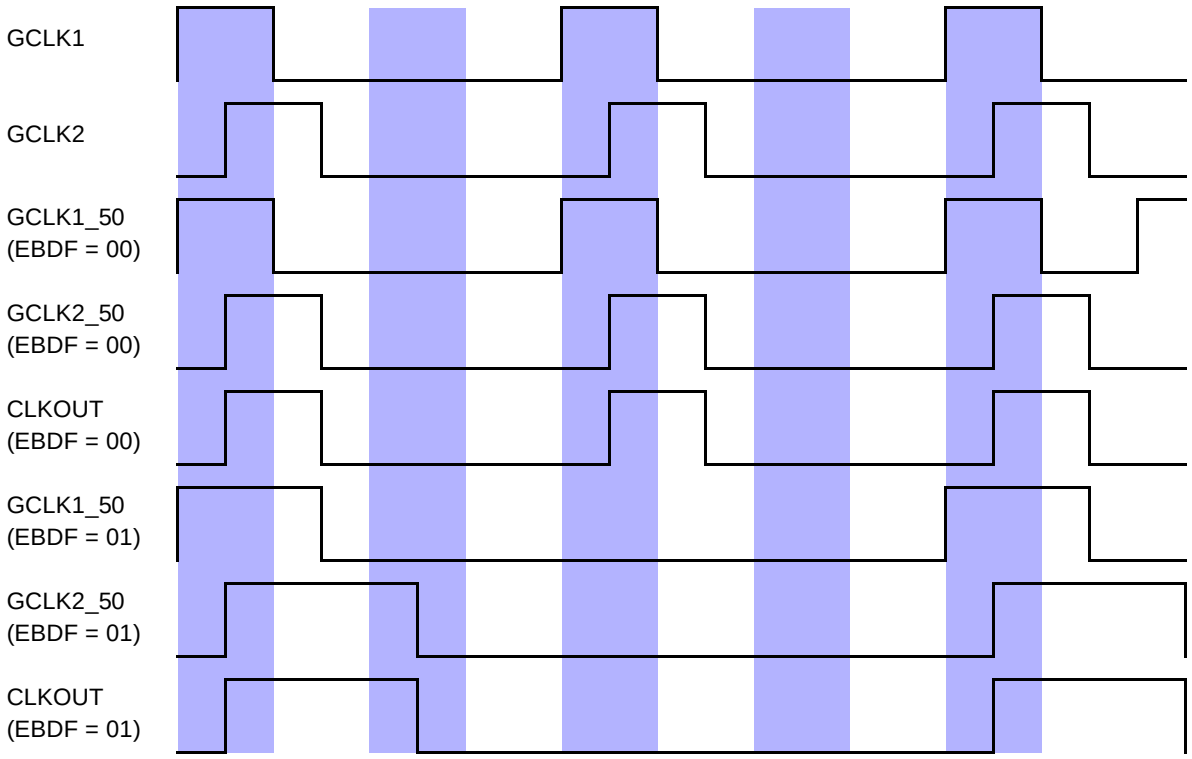


Figure 8-7 Clocks Timing For DFNH = 1 (or DFNL = 0)

8.6.2 Clock Out (CLKOUT)

CLKOUT has the same frequency as the general system clock (GCLK2_50). Unlike the main system clock GCLK1/GCLK2 however, CLKOUT (and GCLK2_50) represents the external bus clock, and thus will be one-half of the main system clock if the external bus is running at half speed (EBDF = 0b01). The CLKOUT frequency defaults to VCO/2. CLKOUT can drive full, half, or quarter strength or be disabled. The drive strength is controlled in the system clock and reset-control register (SCCR) by the COM[0:1] and CQDS bits. (See **8.12.1 System Clock Control Register (SCCR)**). Disabling or decreasing the strength of CLKOUT can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

When the PLL is acquiring lock, the CLKOUT signal is disabled and remains in the low state (provided that BUCS = 0).

8.6.3 Engineering Clock (ENGCLK)

ENGCLK is an output clock with a 50% duty cycle. Its frequency defaults to VCO/128, which is one-sixtyfourth of the main system frequency. ENGCLK frequency can be programmed to the main system frequency divided by a factor from one to 64, as controlled by the ENGDIV[0:5] bits in the SCCR. ENGCLK can drive full- or half-strength or be disabled (remaining in the high state). The drive strength is controlled by the EECLK[0:1] bits in the SCCR. Disabling ENGCLK can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

NOTE

The full strength ENGCLK setting selects a 5-V driver, while the half strength selection is a 2.6-V driver.

When the PLL is acquiring lock, the ENGCLK signal is disabled and remains in the low state (provided that BUCS = 0).

NOTE

Skew elimination between CLKOUT and ENGCLK is not guaranteed.

8.7 Clock Source Switching

For limp mode support, clock source switching is supported. If for any reason the clock source for the chip is not functioning, the option is to switch the system clock to the backup clock ring oscillator, BUCLK.

This circuit consists of a loss-of-clock detector, which sets the LOCS status bit and LOCSS sticky bit in the PLPRCR. If the LME bit in the SCCR is set, whenever LOCS is asserted the clock logic switches the system clock automatically to BUCLK and asserts hard reset to the chip. Switching the system clock to BUCLK is also possible by software setting the STBUC bit in SCCR. Switching from limp mode to normal system operation is accomplished by clearing STBUC and LOCSS bits. This operation also asserts hard reset to the chip.

At $\overline{\text{HRESET}}$ assertion, if the PLL output clock is not valid, the BUCLK will be selected until software clears LOCSS bit in SCCR. At $\overline{\text{HRESET}}$ assertion, if the PLL output clock is valid, the system will switch to oscillator/external clock. If during $\overline{\text{HRESET}}$ the PLL loses lock or the clock frequency becomes slower than the required value, the system will switch to the BUCLK. After $\overline{\text{HRESET}}$ negation the PLL lock condition does not effect the system clock source selection.

If the LME bit is clear, the switch to the backup clock is disabled and assertion of STBUC bit is ignored. If the chip is in limp mode, clearing the LME bit switches the system to normal operation and asserts hard reset to the chip.

Figure 8-8 describes the clock switching control logic. **Table 8-3** summarizes the status and control for each state.

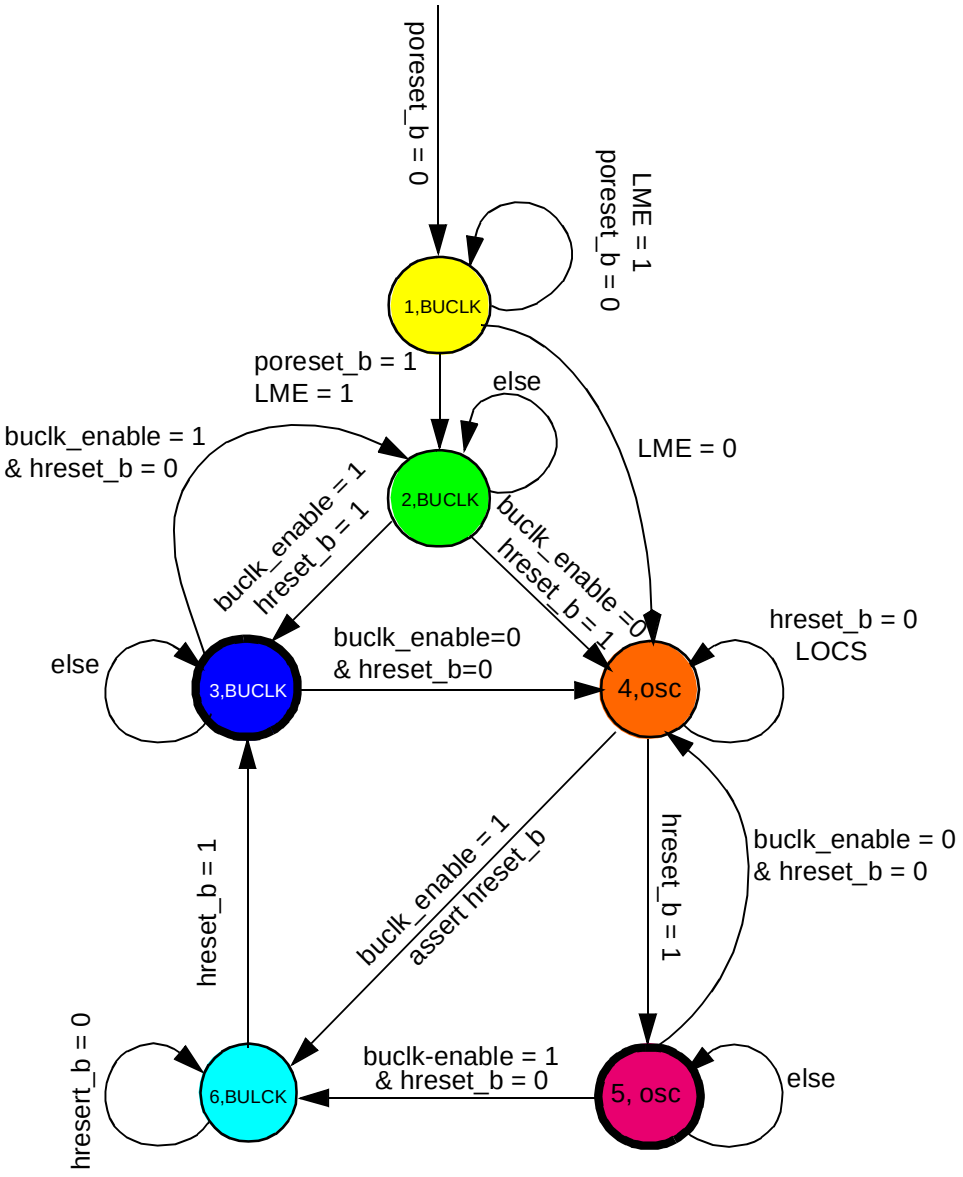


Figure 8-8 Clock Source Switching Flow Chart

NOTES

BUCLK_ENABLE = (STBUC | LOC) & LME lock indicates loss of lock status bit (LOCS) for all cases and loss of clock sticky bit (LOCSS) when state 3 is active. When BUCLK_ENABLE is changed, the chip asserts $\overline{\text{HRESET}}$ to switch the system clock to BUCLK or PLL.

At $\overline{\text{PORESET}}$ negation, if the PLL is not locked, the loss-of-clock sticky bit (LOCSS) is asserted, and the chip should operate with BUCLK.

The switching from state three to state four is accomplished by clearing the STBUC and LOCSS bits. If the switching is done when the PLL is not locked, the system clock will not oscillate until lock condition is met.

Table 8-3 Status of Clock Source

STATE	$\overline{\text{PORESET}}$	$\overline{\text{HRESET}}$	LME	LOCS (status)	LOCSS (sticky)	STBUC	BUCS	Chip Clock Source
1	0	0	1	0	0	0	1	BUCLK
2	1	0	1	0/1	0	0	1	BUCLK
3[1]	1	1	1	x[2]	0/1	0/1	1	BUCLK
4	1	0	0/1	0	x[2]	0	0	Oscillator
5	1	1	0/1	0	x[2]	0	0	Oscillator
6	1	0	1	0/1	1	0/1	1	BUCLK

NOTES:
1. At least one of the two bits, LOCSS or BUCS, must be asserted (one) in this state.
2. X = don't care.

The default value of the LME bit is determined by MODCK[1:3] during assertion of the $\overline{\text{PORESET}}$ line. The configuration modes are shown in **Table 8-1**.

8.8 Low-Power Modes

The LPM and other bits in the PLPRCR are encoded to provide one normal operating mode and four low-power modes. In normal and doze modes the system can be in high state with frequency defined by the DFNH bits, or in the low state with frequency defined by the DFNL bits. The normal-high operating mode is the state out of reset. This is also the state of the bits after the low-power mode exit signal arrives.

There are four low-power modes:

- Doze mode
- Sleep mode
- Deep-sleep mode
- Power-down mode

8.8.1 Entering a Low-Power Mode

Low-power modes are enabled by setting the POW bit in the MSR and clearing the LPML (low-power mode lock) bit in the PLPRCR. Once enabled, a low-power mode is entered by setting the LPM bits to the appropriate value. This can be done only in one of the normal modes. The user cannot change the LPM or CSRC bits when the MCU is in doze mode.

Table 8-6 summarizes the control bit settings for the different clock power modes.

Table 8-4 Power Mode Control Bit Settings

Power Mode	LPM[0:1]	CSRC	TEXPS
Normal-high	00	0	X
Normal-low (“gear”)	00	1	X
Doze-high	01	0	X
Doze-low	01	1	X
Sleep	10	X	X
Deep-sleep	11	X	1
Power-down	11	X	0

8.8.2 Power Mode Descriptions

Table 8-5 describes the power consumption, clock frequency, and chip functionality for each power mode.

Table 8-5 Power Mode Descriptions

Operation Mode	SPLL	Clocks	Functionality
Normal-high	Active	Full frequency ÷ 2^{DFNH}	Full functions not in use are shut off
Normal-low (“gear”)	Active	Full frequency ÷ 2^{DFNL+1}	
Doze-high	Active	Full frequency ÷ 2^{DFNH}	Enabled: RTC, PIT, TB and DEC, memory controller Disabled: extended core (RCPU, BBC, FPU)
Doze-low	Active	Full frequency ÷ 2^{DFNL+1}	
Sleep	Active	Not active	Enabled: RTC, PIT, TB and DEC
Deep-sleep	Not active	Not active	
Power-down	Not active	Not active	
VDDSRAM	Not active	Not active	SRAM data retention

8.8.3 Exiting from Low-Power Modes

Exiting from low-power modes occurs through an asynchronous interrupt or a synchronous interrupt generated by the memory controller. Any enabled asynchronous interrupt clears the LPM bits but does not change the PLPRCR[CSRC] bit.

The exit from normal-low, doze-high, and low modes and sleep mode to normal-high mode is accomplished with the asynchronous interrupt. The sources of the asynchronous interrupt are:

- Asynchronous wake-up interrupt from the interrupt controller
- RTC, PIT, or time base interrupts (if enabled)
- Decrementer exception

The system response to asynchronous interrupts is fast. The wake-up time from normal-low, doze-high, doze-low, and sleep mode due to an asynchronous interrupt or decrementer exception is only three to four clock cycles of maximum system frequency. In 40-MHz systems, this wake-up requires 75 to 100 ns. The asynchronous wake-up interrupt from the interrupt controller is level sensitive one. It will therefore be negated only after the reset of interrupt cause in the interrupt controller.

The timers (RTC, PIT, time base, or decrementer) interrupts indication set status bits in the PLPRCR (TMIST). The clock module considers this interrupt to be pending asynchronous interrupt as long as the TMIST is set. The TMIST status bit should be cleared before entering any low-power mode.

Table 8-7 summarizes wake-up operation for each of the low-power modes.

Table 8-6 Power Mode Wake-Up Operation

Operation Mode	Wake-up Method	Return Time from Wake-up Event to Normal-High
Normal-low (“gear”)	Software or Interrupt	Asynchronous interrupts: 3-4 maximum system cycles Synchronous interrupts: 3-4 actual system cycles
Doze-high	Interrupt	
Doze-low	Interrupt	
Sleep	Interrupt	3-4 maximum system clocks
Deep-sleep	Interrupt	< 500 Oscillator Cycles 125 µs – 4 MHz 25 µs – 20 MHz
Power-down	Interrupt	< 500 oscillator cycles + power supply wake-up
V_{DDSRAM}	External	Power-on sequence

8.8.3.1 Exiting from Normal-Low Mode

In normal mode (as well as doze mode), if the PLPRCR[CSRC] bit is set, the system toggles between low frequency (defined by PLPRCR[DFNL]) and high frequency (defined by PLPRCR[DFNH]. The system switches from normal-low mode to normal-high mode if either of the following conditions is met:

- An interrupt is pending from the interrupt controller; or
- The MSR[POW] bit is cleared (power management is disabled).

When neither of these conditions are met, the PLPRCR[CSRC] bit is set, and the asynchronous interrupt status bits are reset, the system returns to normal-low mode.

8.8.3.2 Exiting From Doze Mode

The system changes from doze mode to normal-high mode whenever an interrupt is pending from the interrupt controller.

8.8.3.3 Exiting From Deep-Sleep Mode

The system switches from deep-sleep mode to normal-high mode if any of the following conditions is met:

- An interrupt is pending from the interrupt controller
- An interrupt is requested by the RTC, PIT, or time base
- A decrementer exception

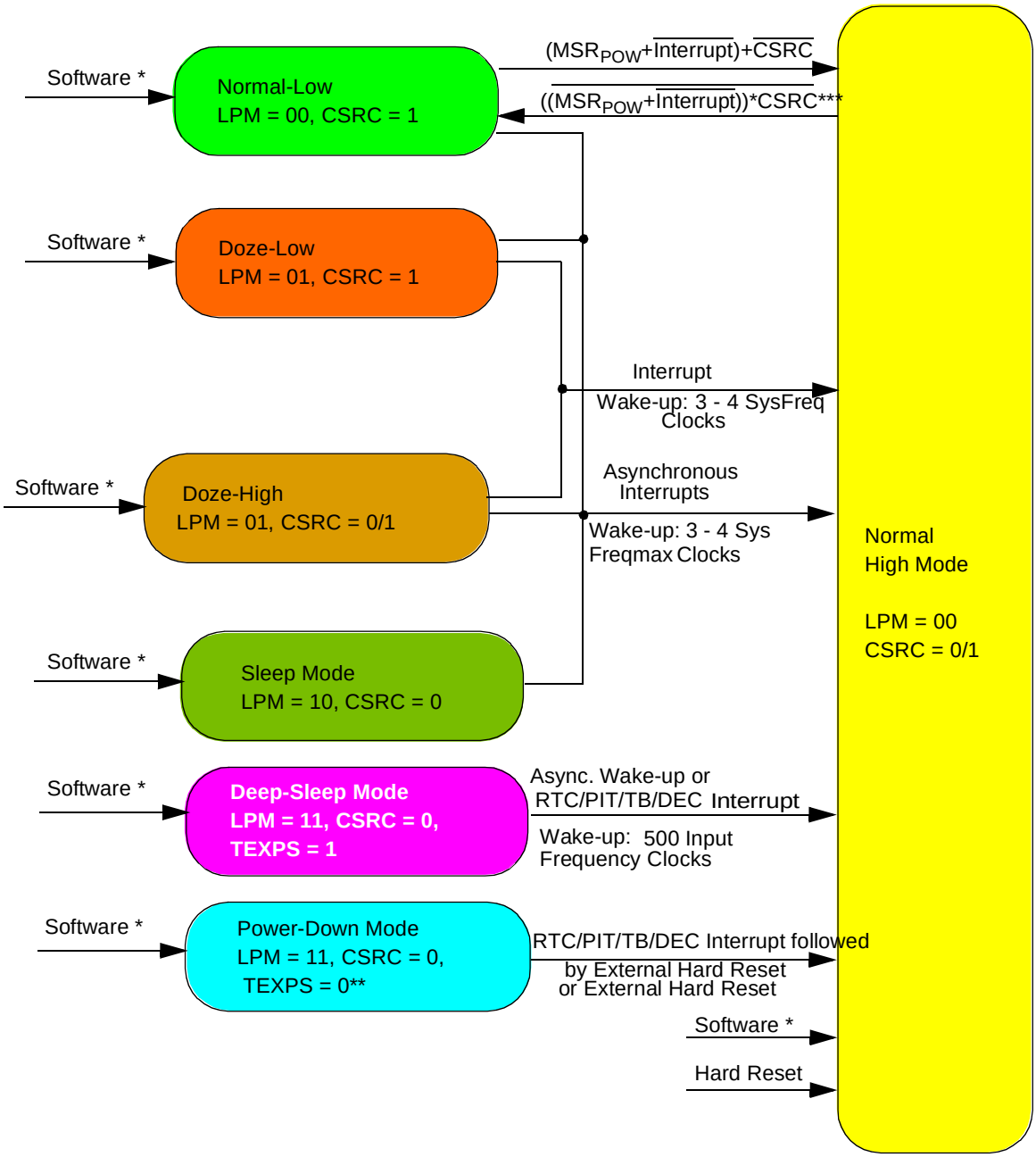
In deep-sleep mode the PLL is disabled. The wake-up time from this mode is up to 500 PLL input frequency clocks. In one-to-one mode the wake-up time may be up to 100 PLL input frequency clocks. For a PLL input frequency of four MHz, the wake-up time is less than 125 µs.

8.8.3.4 Exiting from Power-Down Mode

Exit from power-down mode is accomplished through hard reset. External logic should assert $\overline{\text{HRESET}}$ in response to the TEXPS bit being set and TEXP pin being asserted. The TEXPS bit is set by an enabled RTC, PIT, time base, or decrementer interrupt. The hard reset should be asserted for no longer than the time it takes for the power supply to wake-up in addition to the PLL lock time. When the TEXPS bit is cleared (and the TEXP signal is negated), assertion of hard reset sets the bit, causes the pin to be asserted, and causes an exit from power-down low-power mode. Refer to **8.9.3 Keep Alive Power** for more information.

8.8.3.5 Low-Power Modes Flow

Figure 8-9 shows the flow among the different power modes.



* Software is active only in normal-high/low modes
** TEXPS receives the zero value by writing one. Writing of zero has no effect on TEXPS.
*** The switch from normal-high to normal-low is enable only if the conditions to asynchronous interrupt are cleared

Figure 8-9 MPC565 / MPC566 Low-Power Modes Flow Diagram

8.9 Basic Power Structure

8.9.1 General Power Supply Definitions

KAPWR and VSS power the following clock unit modules: oscillator, PITRTCLK and TMBCLK generation logic, timebase, decrementer, RTC, PIT, system clock control register (SCCR), low-power and reset-control register (PLPRCR), and reset status register (RSR). All other circuits are powered by the normal supply pins, VDD, QVDDL, NVDDL, VDDF, VDDSYN, VDDRTC, VFLASH, VDDH and VSS. The power supply for each block is listed in **Table 8-7**.

Table 8-7 Power Supplies

Circuit	Power Supply
CLKOUT SPLL (digital), System low-power control Internal logic Clock drivers	NVDDL/QVDDL
SPLL (analog)	VDDSYN
Main oscillator Reset machine Limp mode mechanism Register control SCCR, PLLRCR and RSR PPC RTC, PIT, TB, and DEC	KAPWR
CALRAM_A (32 Kbytes Only)	VDDSRAM1/VDD[1]
CALRAM_B (4 Kbytes)	VDDSRAM2/VDD[1]
DPTRAM (AB and C), DECRAM	VDDSRAM3/VDD[1]
MIOS RTCSM	VDDRTC

NOTES:
1. Keep-alive power is supplied by VDDSRAMx, but run current is provided through VDD

The following are the relations between different power supplies:

- VDD = QVDDL = NVDDL = VDDSYN = VDDF = 2.6 V ± 0.1 V
- KAPWR = VDD ± 0.2 V (during normal operation)
- VDDRTC = VDDL ± 0.2 V (during normal operation)
- VDDSRAM[1,2,3] = VDD – 0.3 V (during normal operation)
- VDDH = VDDA = VFLASH = 5.0 ±-5%"
- VDDSRAM[1,2,3] ≥ 1.4 V (during standby operation)
- KAPWR = 2.6 ± -0.1 V (during standby operation)
- VDDRTC = 2.6 ± -0.1 V (during standby operation)

NOTE

The power supply inputs VDD, QVDDL, NVDDL, VDDSYN, and VDDF should all be connected to the same 2.6-V power supply. The power supplies VDDSRAM1, VDDSRAM2, VDDSRAM3, VDDRTC, and KAPWR can, in any combination, be connected to a 2.6-V standby power supply. Standby power pins that are not connected to the standby power supply, should be connected to the same power supply as VDD. Additionally, the standby power supply for VDDSRAM1, VDDSRAM2, and VDDSRAM3 only can be connected to a supply as low as 1.4 V. The power supply inputs VDDH and VFLASH should be connected to the same 5.0-V supply. VDDA can be isolated from VDDH, but should be the same approximate voltage.

8.9.2 Chip Power Structure

The MPC565 / MPC566 provides a wide range of possibilities for power supply connections. **Figure 8-10** illustrates the different power supply sources for each of the basic units on the chip.

8.9.2.1 NVDDL

This supplies the final output stage of the 2.6-V pad output drivers.

8.9.2.2 QVDDL

This supplies all pad logic and pre-driver circuitry, except for the the final output stage of the 2.6-V pad output drivers.

8.9.2.3 VDD

VDD powers the internal logic of the MPC565 / MPC566, nominally 2.6 V.

8.9.2.4 VDDSYN, VSSSYN

The charge pump and the VCO of the SPLL are fed by a separate 2.6-V power supply (VDDSYN) in order to improve noise immunity and achieve a high stability in its output frequency. VSSSYN provides an isolated ground reference for the PLL.

8.9.2.5 KAPWR

The oscillator, time base counter, decrementer, periodic interrupt timer and the real-time clock are fed by the KAPWR rail. This allows the external power supply unit to disconnect all other sub-units of the MCU in low-power deep-sleep mode. The TEXP pin (fed by the same rail) can be used by the external power supply unit to switch between sources. The $\overline{\text{IRQ[6:7]}}$/MODCK[2:3], $\overline{\text{IRQ5}}$/MODCK1, XTAL, EXTAL, EXT-CLK, $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$, and $\overline{\text{RSTCONF}}$/TEXP input pins are powered by KAPWR. Circuits, including pull-up resisters, driving these inputs should be powered by KAPWR.

8.9.2.6 VDDA, VSSA

VDDA supplies power to the analog subsystems of the QADCA and QADCB modules; it is nominally 5.0 V. VDDA is the ground reference for the analog subsystems.

8.9.2.7 VFLASH

VFLASH supplies the UC3F normal operating voltage. It is nominally 5.0 V.

8.9.2.8 VDDF, VSSF

VDDF provides internal core voltage to the UC3F flash module; it should be a nominal 2.6 V. VSSF provides an isolated ground for the UC3F flash module.

8.9.2.9 VDDH

VDDH provides power for the 5-V I/O operations. It is a nominal 5.0 V.

8.9.2.10 VDDSRAM1

This is the 2.6-V voltage supply input for the keep-alive section of the CALRAM_A(32K) module. This pin supplies only keep-alive power to the CALRAM_A(32K) module. This supply can be between 1.4 and 2.7 V. Run current is supplied by normal VDD.

8.9.2.11 VDDSRAM2

This is the 2.6-V voltage supply input for the CALRAM_B (4K) module. This supply can be between 1.4 and 2.7 V. This pin supplies keep-alive power only to the CALRAM_B(4K) module. Run current is supplied by normal VDD.

8.9.2.12 VDDSRAM3

This is the 2.6-V voltage supply input for the arrays in the DPTRAM_AB (6K), DPTRAM_C (4K), and DECRAM modules. This pin supplies keep-alive voltage to the DPTRAM arrays and the DECRAM. Run current is supplied by the normal VDD.

8.9.2.13 VDDRTC

VDDRTC supplies power to the MIOS real-time clock submodule and its 32.768-KHz oscillator. The RTC can be kept running when all of the other power supplies are turned off.

8.9.2.14 VSS

VSS provides the ground reference for the MPC565 / MPC566.

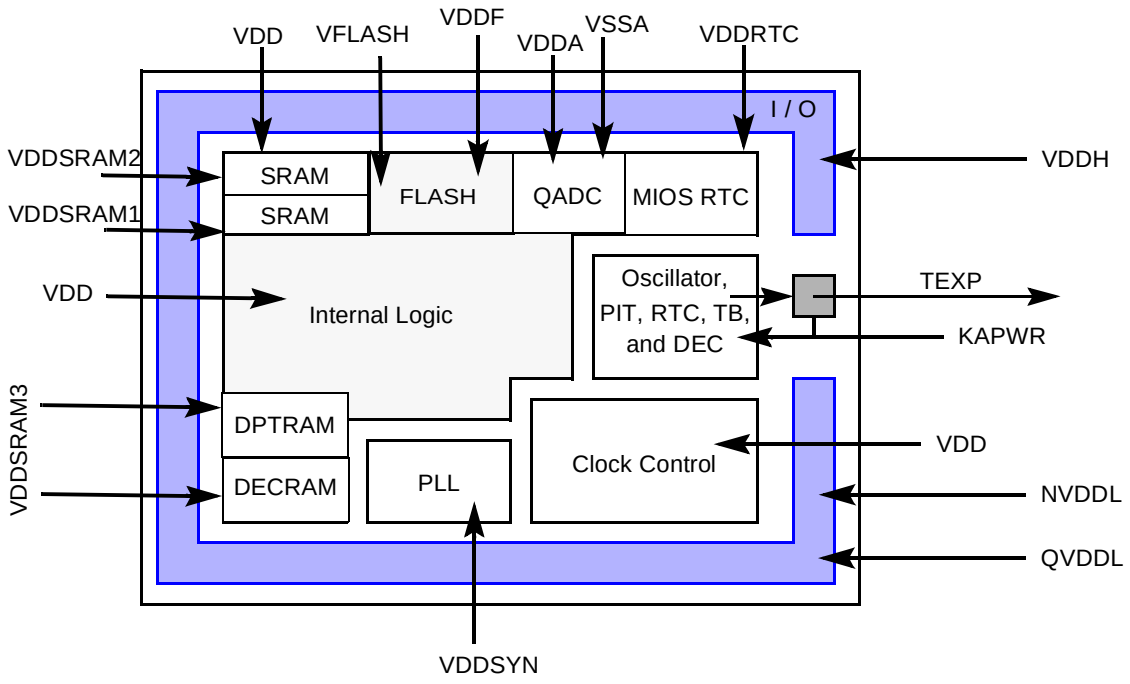


Figure 8-10 Basic Power Supply Configuration

8.9.3 Keep Alive Power

8.9.3.1 Keep Alive Power Configuration

Figure 8-11 illustrates an example of a switching scheme for an optimized low-power system. SW1 and SW2 can be unified in only one switch if VDDSYN and VDD/NVDDL/QVDDL are supplied by the same source.

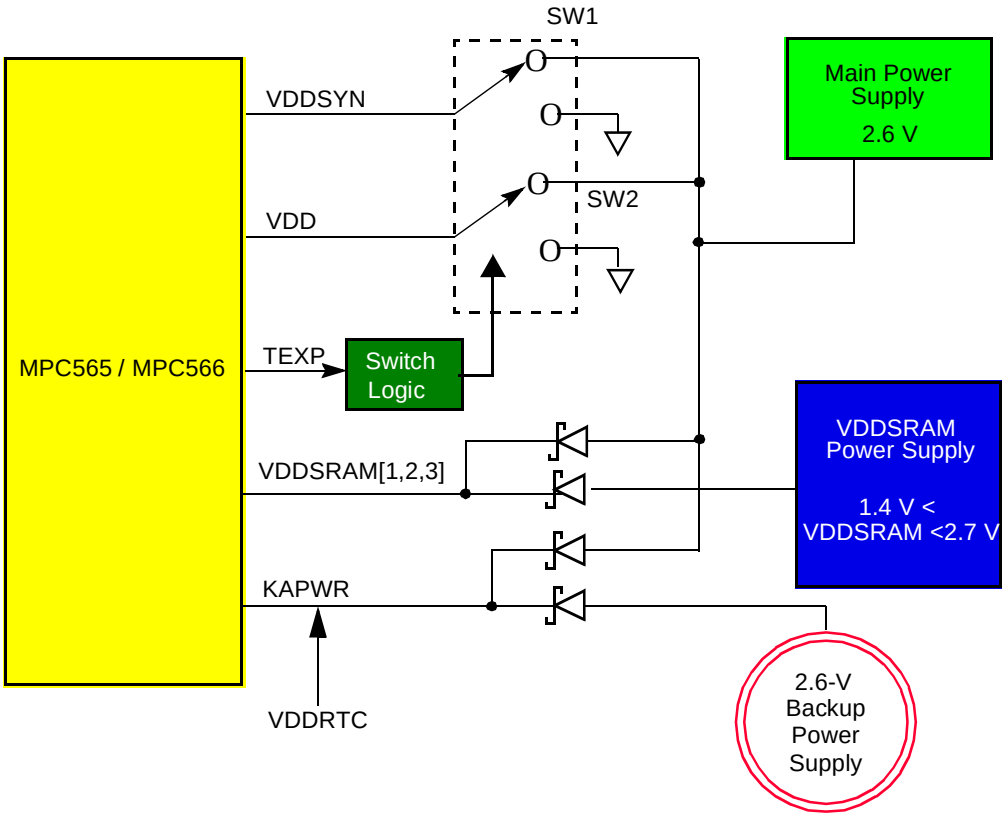


Figure 8-11 External Power Supply Scheme

The MPC565 / MPC566 asserts the TEXP signal, if enabled, when the RTC or TB time value matches the value programmed in the associated alarm register or when the PIT or DEC value reaches zero. The TEXP signal is negated when the TEXPS status bit is written to one.

The KAPWR power supply feeds the main crystal oscillator (OSCM). The condition for the main crystal oscillator stability is that the power supply value changes slowly. The maximum slope must be less than 5 mV per oscillation cycle (τ > 200-300/$freq_{oscm}$).

8.9.3.2 Keep-Alive Power Registers Lock Mechanism

The USIU timer, clocks, reset, power, decrementer, and time base registers are powered by the KAPWR supply. When the main power supply is disconnected after power-down mode is entered, the value stored in any of these registers is preserved. If power-down mode is not entered before power disconnect, there is a chance of data loss in these registers. To minimize the possibility of data loss, the MPC565 / MPC566 includes a key mechanism that ensures data retention as long as a register is locked. While a register is locked, writes to this register are ignored.

Each of the registers in the KAPWR region have a key that can be in one of two states: open or locked. At power-on reset the following keys are locked: RTC, RTSEC,

RTCAL, and RTCSC. All other registers are unlocked. Each key has an address associated with it in the internal memory map.

A write of 0x55CCAA33 to the associated key register changes the key to the open state. A write of any other data to this location changes the key to the locked state. The key registers are write-only. A read of the key register has undefined side effects and may be interpreted as a write that locks the associated register.

Table 8-8 lists the registers powered by KAPWR and the associated key registers.

Table 8-8 KAPWR Registers and Key Registers

KAPWR Register		Associated Key Register	
Address or SPR Number	**Register**	**Address**	**Register**
0x2F C200	Time Base Status and Control (TBSCR) See **Table 6-18** for bit descriptions.	0x2F C300	Time Base Status and Control Key (TBSCRK)
0x2F C204	Time Base Reference 0 (TBREF0) See **6.14.4.3 Time Base Reference Registers** for bit descriptions.	0x2F C304	Time Base Reference 0 Key (TBREF0K)
0x2F C208	Time Base Reference 1 (TBREF1) See **6.14.4.3 Time Base Reference Registers** for bit descriptions.	0x2F C308	Time Base Reference 1 Key (TBREF1K)
0x2F C220	Real Time Clock Status and Control (RTCSC) See **Table 6-19** for bit descriptions.	0x2F C320	Real Time Clock Status and Control Key (RTCSCK)
0x2F C224	Real Time Clock (RTC) See **6.14.4.6 Real-Time Clock Register (RTC)** for bit descriptions.	0x2F C324	Real Time Clock Key (RTCK)
0x2F C228	Real Time Alarm Seconds (RTSEC) Reserved	0x2F C328	Real Time Alarm Seconds Key (RTSECK)
0x2F C22C	Real Time Alarm (RTCAL) See **6.14.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.	0x2F C32C	Real Time Alarm Key (RTCALK)
0x2F C240	PIT Status and Control (PISCR) See **Table 6-20** for bit descriptions.	0x2F C340	PIT Status and Control Key (PISCRK)
0x2F C244	PIT Count (PITC) See **Table 6-21** for bit descriptions.	0x2F C344	PIT Count Key (PITCK)
0x2F C280	System Clock Control Register (SCCR) See **Table 8-9** for bit descriptions.	0x2F C380	System Clock Control Key (SCCRK)
0x2F C284	PLL Low-Power and Reset-Control Register (PLPRCR) See **Table 8-11** for bit descriptions.	0x2F C384	PLL Low-Power and Reset-Control Register Key (PLPRCRK)
0x2F C288	Reset Status Register (RSR) See **Table 7-3** for bit descriptions.	0x2F C388	Reset Status Register Key (RSRK)
SPR 22	Decrementer See **3.9.5 Decrementer Register (DEC)** for bit descriptions.	0x2F C30C	Time Base and Decrementer Key (TBK)
SPR 268, 269, 284, 285,	Time Base See **Table 3-11** and **Table 3-14** for bit descriptions.		

Figure 8-12 illustrates the process of locking or unlocking a register powered by KAPWR.

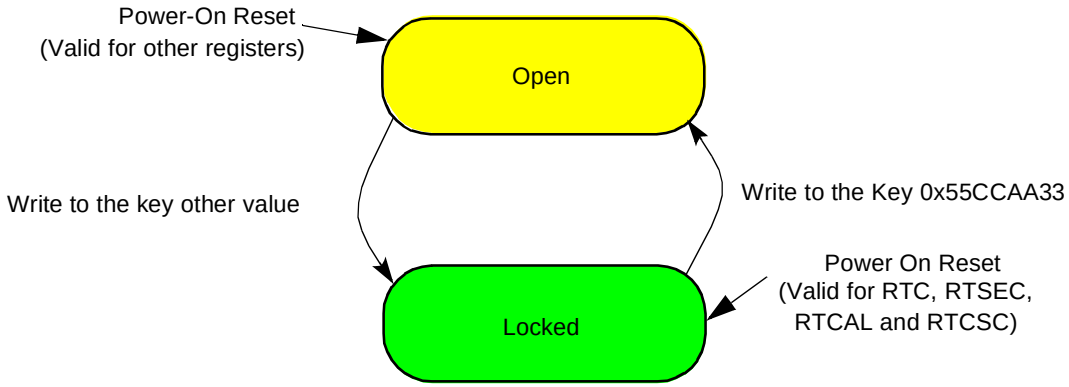


Figure 8-12 Keep-Alive Register Key State Diagram

8.10 VDDSRAM Supply Failure Detection

A special circuit for VDDSRAM supply failure detection is provided. In the case of supply failure detection, the dedicated sticky bits LVSRS in the VSRMCR register are asserted. Software can read or clear these bits. The user should enable the detector and then clear these bits. If t any of the LVSR bits are read as one, then a power failure of VDDSRAM has occurred. The circuit is capable of detecting supply failure below T.B.D. V. Also, enable/disable control bit for the VDDSRAM detector may be used to disconnect the circuit and save the detector power consumption.

8.11 Power-Up/Down Sequencing

Figure 8-13 and **Figure 8-14** detail the power-up sequencing for MPC565 / MPC566 during normal operation. Note that for each of the conditions detailing the voltage relationships the absolute bounds of the minimum and maximum voltage supply cannot be violated, (i.e., the value of VDDL) cannot fall below 2.5 V or exceed 2.7 V and the value of VDDH cannot fall below 4.75 V or exceed 5.25 V for normal operation. Power consumption during power up sequencing can not be specified prior to evaluation and characterization of production silicon. The goal is to keep the power consumption during power up sequencing below the operating power consumption.

During the power down sequence $\overline{\text{PORESET}}$ needs to asserted while VDD, NVDDL, and QVDDL are at a voltage equal or greater to 2.5 V. Below this voltage the power supply chip can be turned off. If the turn-off voltage of the power supply chip is greater than 0.74 V for the 2.6-V supply and greater than 0.8 V for the 5-V supply, then the circuitry inside the MPC565 / MPC566 will act as a load to the respective supply and will discharge the supply line down to these values. Since the 2.6-V logic represents a larger load to the supply chip, the 2.6-V supply line will decay faster than the 5-V supply line.

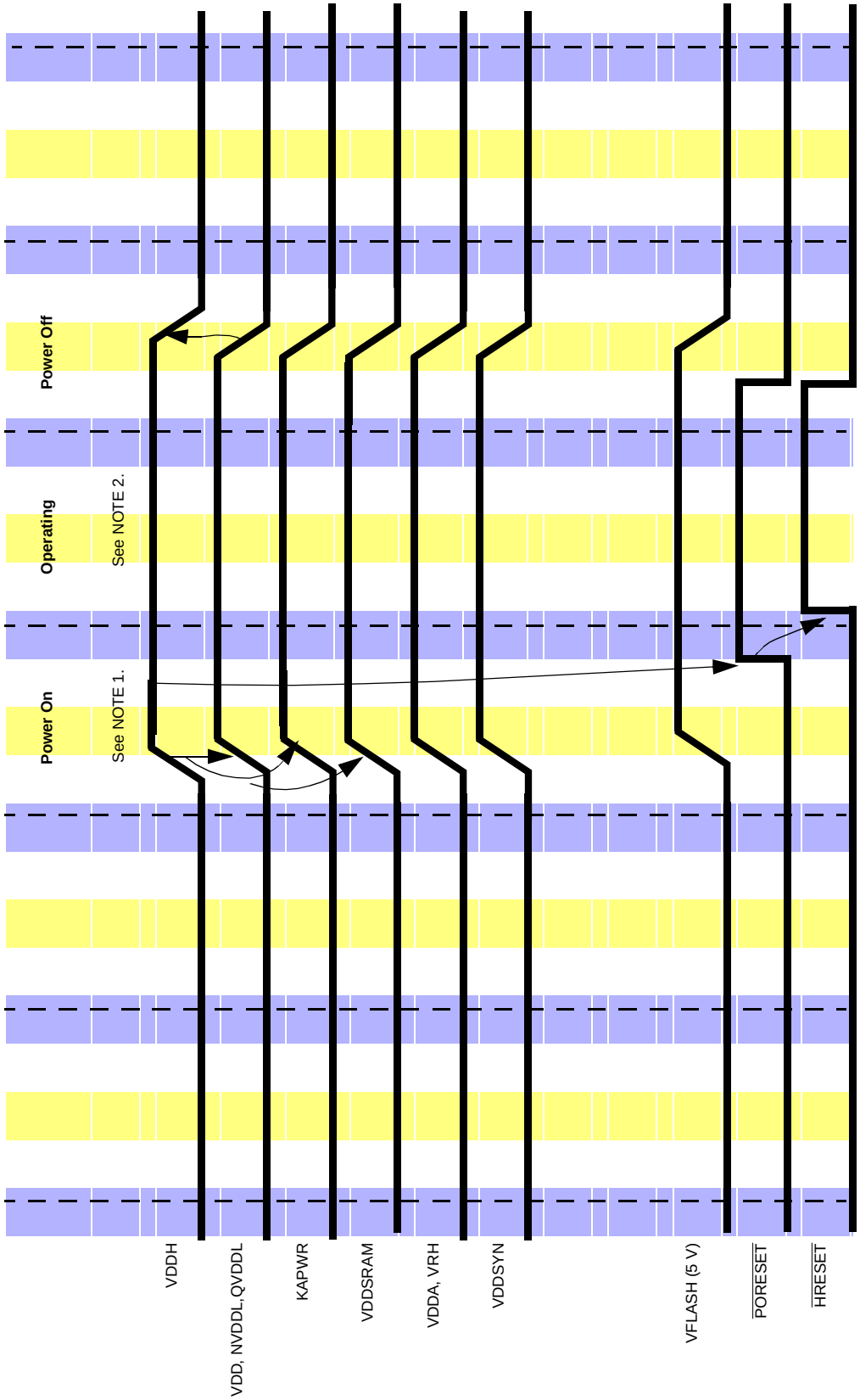


Figure 8-13 No Standby, No KAPWR, All System Power On/Off

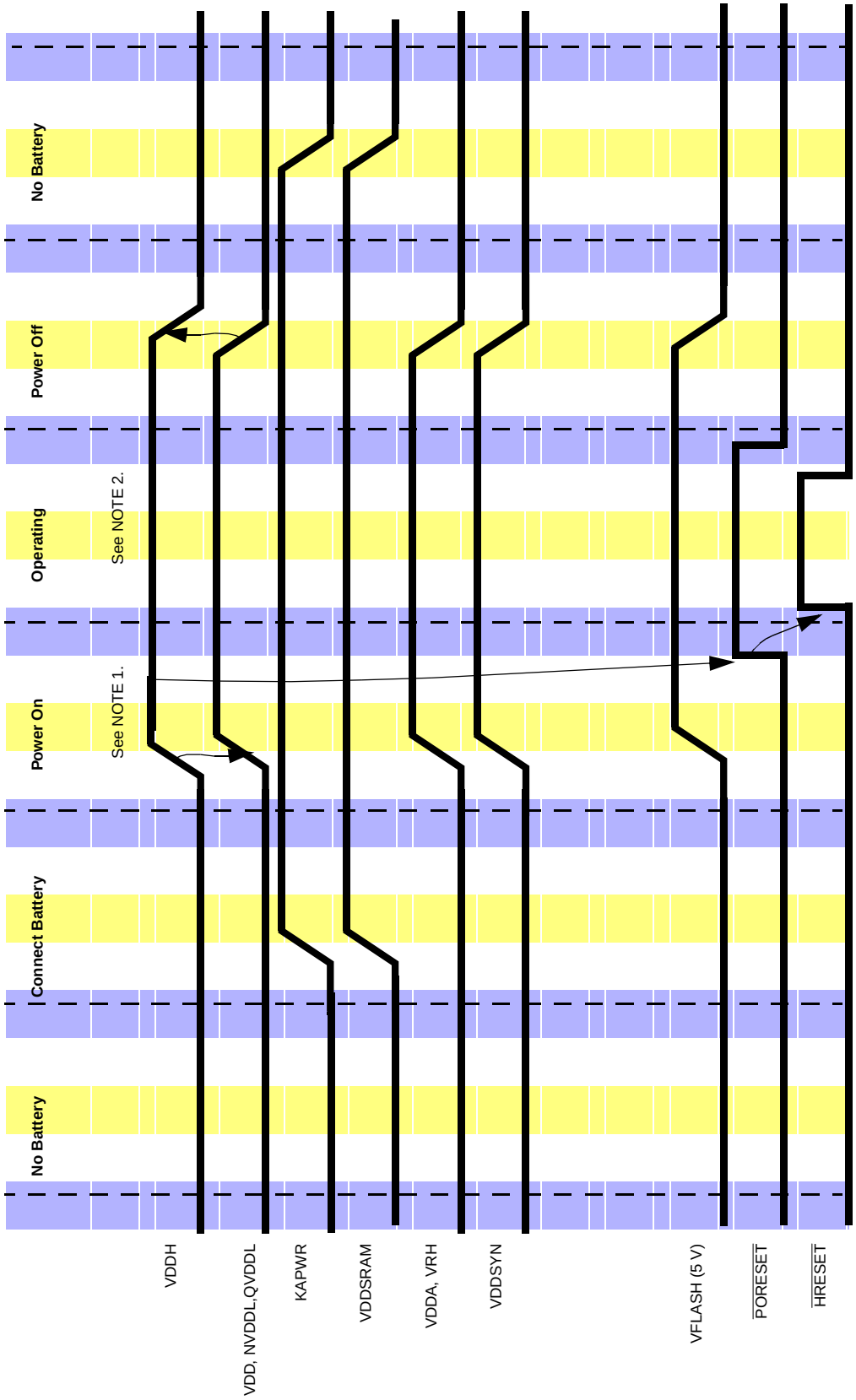


Figure 8-14 Standby and KAPWR, Other Power On/Off

NOTE

The following notes apply to **Figure 8-13** and **Figure 8-14** above. For more detailed information on power sequencing see **E.9 Power-Up/ Down Sequencing**.

1. VDDH ≥ QVDDL - 0.35 V (0.5 V max. at temperature extremes)

   VDDA can lag VDDH, and VDDSYN can lag QVDDL, but both must be at a valid level before resets are negated.

2. If keep alive functions are NOT used, then when system power is on:

   KAPWR = VDDSRAM[1:3] = QVDDL ± 0.1 V
   KAPWR ≤ 2.7 V

3. If keep alive functions ARE used, then

   KAPWR = VDDSRAM[1:3] = QVDDL = NVDDL = 2.7 V ± 0.1 V when system power is on
   VDDSRAM[1:3] ≥ 1.8 V and optionally KAPWR = 2.7 V ± 0.1 V when system power is off

   Normal system power is defined as
   QVDDL = VDDI = VDDF = VDDSYN = VDDSRAM[1:3] = KAPWR = 2.7 ± 0.1 V and VDDA = VDDH = VFLASH = 5.0 ± 0.25 V

   Flash Programming requirements are the same as normal system power. VFLASH should always be 5.0 ± 0.25 V.

4. Do not hold the 3-V supplies at ground while VDDH/VDDA is ramping to 5 V.

5. If 5 V is applied before the 3-V supply, all 5-V outputs will be in indeterminate states until the 3-V supply reaches a level that allows reset to be distributed throughout the device

8.12 Clocks Unit Programming Model

8.12.1 System Clock Control Register (SCCR)

The SPLL has a 32-bit control register, SCCR, which is powered by keep-alive power.

SCCR — System Clock and Reset Control Register **0x2F C280**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DBCT	COM		DCSLR	MFPDL	LPML	TBS	RTDIV	STBUC	CQDS	PRQEN	RTSEL	BUCS	EBDF[0:1]		LME
PORESET:															
					1		0	1	0	0	0	0		0	1
HRESET:															
U	0	ID(2)[1]	U	U	U	U	U	U	U	1	U	U	ID(13:14)[2]		U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
EECLK[0:1]		ENGDIV[0:5]						Re-served	DFNL[0:2]			Re-served	DFNH[0:2]		
PORESET:															
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
HRESET:															
U	U	U	U	U	U	U	U	0	0	0	0	0	0	0	0

NOTES:
1. The hreset value is a reset configuration word value, extracted from the internal data bus ID(2) line.
2. The hreset value is a reset configuration word value, extracted from the internal data bus ID(13:14) lines.

NOTE

COM[1] bit default value is determined during by BDRV reset configuration bit; See **7.5.2 Hard Reset Configuration Word**.

Table 8-9 SCCR Bit Descriptions

Bit(s)	Name	Description
0	DBCT	
1:2	COM	Clock Output Mode – The COM and CQDS bits control the output buffer strength of the CLKOUT and external bus pins. When both COM bits are set the CLKOUT pin is held in the high (1) state and external bus pins are driven at reduced drive. These bits can be dynamically changed without generating spikes on the CLKOUT and external bus pins. If CLKOUT pin is not connected to external circuits, set both bits (disabling CLKOUT) to minimize noise and power dissipation. The default value for COM[1] is determined by the BDRV bit in the reset configuration word. See **Table 7-5**. For CLKOUT control see **Table 8-10**.
3	DCSLR	Disable clock switching at loss of lock during reset. When DCSLR is clear and limp mode is enabled, the chip will switch automatically to the backup clock if the PLL losses lock during $\overline{\text{HRESET}}$. When DCSLR is asserted, a PLL loss-of-lock event does not cause clock switching. If $\overline{\text{HRESET}}$ is asserted and DCSLR is set, the chip will not negate $\overline{\text{HRESET}}$ until the PLL acquires lock. 0 = Enable clock switching if the PLL loses lock during reset 1 = Disable clock switching if the PLL loses lock during reset

Table 8-9 SCCR Bit Descriptions (Continued)

Bit(s)	Name	Description
4	MFPDL	MF and pre-divider lock. Setting this control bit disables writes to the MF and DIVF bits. This helps prevent runaway software from changing the VCO frequency and causing the SPLL to lose lock. In addition, to protect against hardware interference, a hardware reset will be asserted if these fields are changed while LPML is asserted. This bit is writable once after power-on reset. 0 = MF and DIVF fields are writable 1 = MF and DIVF fields are locked
5	LPML	LPM lock. Setting this control bit disables writes to the LPM and CSRC control bits. In addition, for added protection, a hardware reset is asserted if any mode is entered other than normal-high mode. This protects against runaway software causing the MCU to enter low-power modes. (The MSR[POW] bit provides additional protection). LPML is writable once after power-on reset. 0 = LPM and CSRC bits are writable 1 = LPM and CSRC bits are locked and hard reset will occur if the MCU is not in normal-high mode
6	TBS	Time base source. Note that when the chip is operating in limp mode (BUCS = 1), TBS is ignored, and the backup clock is the time base clock source. 0 = Source is OSCCLK divided by either four or 16 1 = Source is system clock divided by 16
7	RTDIV	RTC (and PIT) clock divider. At power-on reset this bit is cleared if MODCK[1:3] are all low; otherwise the bit is set. 0 = RTC and PIT clock divided by four 1 = RTC and PIT clock divided by 256
8	STBUC	Switch to backup clock control. When software sets this bit, the system clock is switched to the on-chip backup clock ring oscillator, and the chip undergoes a hard reset. The STBUC bit is ignored if LME is cleared. 0 = Do not switch to the backup clock ring oscillator 1 = Switch to backup clock ring oscillator
9	CQDS	Clock quarter drive strength — The COM and CQDS bits control the output buffer strength of the CLKOUT, see **Table 8-10**.
10	PRQEN	Power management request enable 0 = Remains in the lower frequency (defined by DFNL) even if the power management bit in the MSR is reset (normal operational mode) or if there is a pending interrupt from the interrupt controller 1 = Switches to high frequency (defined by DFNH) when the power management bit in the MSR is reset (normal operational mode) or there is a pending interrupt from the interrupt controller
11	RTSEL	RTC circuit input source select. At power-on reset RTSEL receives the value of the MODCK1 bit. Note that if the chip is operating in limp mode (BUCS = 0), the RTSEL bit is ignored, and the backup clock is the clock source for the RT and PIT clocks 0 = OSCM clock is selected as input to RTC and PIT 1 = EXTCLK clock is selected as the RTC and PIT clock source
12	BUCS	Backup clock status. This status bit indicates the current system clock source. When loss of clock is detected and the LME bit is set, the clock source is the backup clock and this bit is set. When the STBUC bit and LME bit are set, the system switches to the backup clock and BUCS is set. 0 = System clock is not the backup clock 1 = System clock is the backup clock

Table 8-9 SCCR Bit Descriptions (Continued)

Bit(s)	Name	Description
13:14	EBDF[0:1]	External bus division factor. These bits define the frequency division factor between (GCLK1 and GCLK2) and (GCLK1_50 and GCLK2_50). CLKOUT is similar to GCLK2_50. The GCLK2_50 and GCKL1_50 are used by the external bus interface and memory controller in order to interface to the external system. The EBDF bits are initialized during hard reset using the hard reset configuration mechanism. 00 = CLKOUT is GCKL2 divided by 1 01 = CLKOUT is GCKL2 divided by 2 1x = Reserved
15	LME	Limp mode enable. When LME is set, the loss-of-clock monitor is enabled and any detection of loss of clock will switch the system clock automatically to backup clock. It is also possible to switch to the backup clock by setting the STBUC bit. If LME is cleared, the option of using limp mode is disabled. The loss of clock detector is not active, and any write to STBUC is ignored. The LME bit is writable once, by software, after power-on reset, when the system clock is not backup clock (BUCS = 0). During power-on reset, the value of LME is determined by the MODCK[1:3] bits. (Refer to **Table 8-1**.) 0 = Limp mode disabled 1 = Limp mode enabled
16:17	EECLK[0:1]	Enable engineering clock. This field controls the output buffer voltage of the ENGCLK pin. When both bits are set the ENGCLK pin is held in the high state. These bits can be dynamically changed without generating spikes on the ENGCLK pin. If ENGCLK is not connected to external circuits, set both bits (disabling ENGCLK) to minimize noise and power dissipation. For measurement purposes the backup clock (BUCLK) can be driven externally on the ENGCLK pin. 00 = Engineering clock enabled, 2.6 V output buffer 01 = Engineering clock enabled, 5 V output buffer 10 = BUCLK is the output on the ENGCLK 2.6 V output buffer 11 = Engineering clock disabled
18:23	ENGDIV[0:5]	Engineering clock division factor. These bits define the frequency division factor between VCO/2 and ENGCLK. Division factor can be from 1 (ENGDIV = 000000) to 64 (ENGDIV = 111111). These bits can be read and written at any time. They are not affected by hard reset but are cleared during power-on reset. **NOTE**: If the engineering clock division factor is not a power of two, synchronization between the system and ENGCLK is not guaranteed.
24	—	Reserved
25:27	DFNL[0:2]	Division factor low frequency. The user can load these bits with the desired divide value and the CSRC bit to change the frequency. Changing the value of these bits does not result in a loss of lock condition. These bits are cleared by power-on or hard reset. Refer to **8.6.1 General System Clocks** and **Figure 8-5** for details on using these bits. 000 = Divide by 2 001 = Divide by 4 010 = Divide by 8 011 = Divide by 16 100 = Divide by 32 101 = Divide by 64 110 = Reserved 111 = Divide by 256

Table 8-9 SCCR Bit Descriptions (Continued)

Bit(s)	Name	Description
28	—	Reserved
29:31	DFNH	Division factor high frequency. These bits determine the general system clock frequency during normal mode. Changing the value of these bits does not result in a loss of lock condition. These bits are cleared by power-on or hard reset. The user can load these bits at any time to change the general system clock rate. Note that the GCLKs generated by this division factor are not 50% duty cycle (i.e. CLKOUT). 000 = Divide by 1 001 = Divide by 2 010 = Divide by 4 011 = Divide by 8 100 = Divide by 16 101 = Divide by 32 110 = Divide by 64 111 = Reserved

Table 8-10 COM and CQDS Bits Functionality

COM[0:1]	CQDS	Function
00	x	Clock Output Enabled Full-Strength Output Buffer, Bus pins full drive
01	0	Clock Output Enabled Half-Strength Output Buffer, Bus pins reduced drive
01	1	Clock Output Enabled Quarter-Strength Output Buffer, Bus pins reduced drive
10	x	Clock Output Disabled, Bus pins full drive
11	x	Clock Output Disabled, Bus pins reduced drive

8.12.2 PLL, Low-Power, and Reset-Control Register (PLPRCR)

The PLL, low-power, and reset-control register (PLPRCR) is a 32-bit register powered by the keep-alive power supply.

PLPRCR — PLL, Low-Power, and Reset-Control Register **0x2F C284**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MF												RE-SERVED	LOCS	LOCSS	SPLS
POWER-ON RESET:															
0 OR 4												0	0	0	0
HARD RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
SPLSS	TEXPS	TEXP INV –	TMIST	RE-SERVED	CSRC	LPM		CSR	LOLRE	RE-SERVED	DIVF				
POWER-ON RESET:															
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HARD RESET:															
U	1	U	0	U	0	0	0	U	U		U	U	U	U	U

U = Unaffected by reset

Table 8-11 PLPRCR Bit Descriptions

Bit(s)	Name	Description
0:11	MF	Multiplication factor bits. The output of the VCO is divided to generate the feedback signal to the phase comparator. The MF bits control the value of the divider in the SPLL feedback loop. The phase comparator determines the phase shift between the feedback signal and the reference clock. This difference results in either an increase or decrease in the VCO output frequency. The MF bits can be read and written at any time. However, this field can be write-protected by setting the MF and pre-divider lock (MFPDL) bit in the SCCR. Changing the MF bits causes the SPLL to lose lock. Also, the MF field should not be modified when entering or exiting from low power mode (LPM change), or when back-up clock is active. The normal reset value for the DFNH bits is zero (divide by one). When the PLL is operating in one-to-one mode, the multiplication factor is set to x1 (MF = 0).
12	—	Reserved
13	LOCS	Loss of clock status. When the oscillator or external clock source is not at the minimum frequency, the loss-of-clock circuit asserts the LOCS bit. This bit is cleared when the oscillator or external clock source is functioning normally. This bit is reset only on power-on reset. Writes to this bit have no effect. 0 = No loss of oscillator is currently detected 1 = Loss of oscillator is currently detected
14	LOCSS	Loss of clock sticky. If, after negation of $\overline{\text{PORESET}}$, the loss-of-clock circuit detects that the oscillator or external clock source is not at a minimum frequency, the LOCSS bit is set. LOCSS remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. The reset value is determined during hard reset. The STBUC bit will be set provided the PLL lock condition is not met when $\overline{\text{HRESET}}$ is asserted, and cleared if the PLL is locked when $\overline{\text{HRESET}}$ is asserted. 0 = No loss of oscillator has been detected 1 = Loss of oscillator has been detected

Table 8-11 PLPRCR Bit Descriptions (Continued)

Bit(s)	Name	Description
15	SPLS	System PLL lock status bit 0 = SPLL is currently not locked 1 = SPLL is currently locked
16	SPLSS	SPLL lock status sticky bit. An out-of-lock sets the SPLSS bit. The bit remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. The bit is cleared at power-on reset. This bit is not affected due to a software initiated loss-of-lock (MF change and entering deep-sleep or power-down mode). The SPLSS bit is not affected by hard reset. 0 = SPLL has remained in lock 1 = SPLL has gone out of lock at least once (not due to software-initiated loss of lock)
17	TEXPS	Timer expired status bit. This bit controls whether the chip negates the TEXP pin in deep-sleep mode, thus enabling external circuitry to switch off the VDD (power-down mode). When LPM = 11, CSRC = 0, and TEXPS is high, the TEXP pin remains asserted. When LPM = 11, CSRC = 0, and TEXPS is low, the TEXPS pin is negated. To enable automatic wake-up TEXPS is asserted when one of the following occurs: • The PIT is expired • The real-time clock alarm is set • The time base clock alarm is set • The decrementer exception occurs The bit remains set until software clears it by writing a one to it. A write of zero has no effect on this bit. TEXPS is set by power-on or hard reset. 0 = TEXP is negated in deep-sleep mode 1 = TEXP pin remains asserted always
18	TEXP_INVP	Timer Expired Pin Inversed Polarity – The TEX_INVP bit controls whether the polarity of the TEXP pin will be active high (normal default) or active low. 0 = The TEXP pin is active high 1 = The TEXP pin is active low
19	TMIST	Timers interrupt status.TMIST is set when an interrupt from the RTC, PIT, TB or DEC occurs. The TMIST bit is cleared by writing a one to it. Writing a zero has no effect on this bit. The system clock frequency remains at its high frequency value (defined by DFNH) if the TMIST bit is set, even if the CSRC bit in the PLPRCR is set (DFNL enabled) and conditions to switch to normal-low mode do not exist. This bit is cleared during power-on or hard reset. 0 = No timer expired event was detected 1 = A timer expire event was detected
20	—	Reserved
21	CSRC	Clock source. This bit is cleared at hard reset. 0 = General system clock is determined by the DFNH value 1 = General system clock is determined by the DFNL value
22:23	LPM	Low-power mode select. These bits are encoded to provide one normal operating mode and four low-power modes. In normal and doze modes, the system can be in high state (frequency determined by the DFNH bits) or low state (frequency defined by the DFNL bits). The LPM field can be write-protected by setting the LPM and CSRC lock (LPML) bit in the PLPRCR Refer to **Table 8-4** and **Table 8-5**.
24	CSR	Checkstop reset enable. If this bit is set, then an automatic reset is generated when the RCPU signals that it has entered checkstop mode, unless debug mode was enabled at reset. If the bit is clear and debug mode is not enabled, then the USIU will not do anything upon receiving the checkstop signal from the RCPU. If debug mode is enabled, then the part enters debug mode upon entering checkstop mode. In this case, the RCPU will not assert the checkstop signal to the reset circuitry. This bit is writable once after soft reset. 0 = No reset will occur when checkstop is asserted 1 = Reset will occur when checkstop is asserted

Table 8-11 PLPRCR Bit Descriptions (Continued)

Bit(s)	Name	Description
25	LOLRE	Loss of lock reset enable 0 = Loss of lock does not cause $\overline{\text{HRESET}}$ assertion 1 = Loss of lock causes $\overline{\text{HRESET}}$ assertion Note: if limp mode is enabled, use the COLIR feature instead of setting the LOLRE bit. See **8.12.3 Change of Lock Interrupt Register (COLIR)**.
26	—	Reserved
27:31	DIVF	The DIVF bits control the value of the pre-divider in the SPLL circuit. The DIVF bits can be read and written at any time. However, the DIVF field can be write-protected by setting the MF and pre-divider lock (MFPDL) bit in the SCCR. Changing the DIVF bits causes the SPLL to lose lock.

8.12.3 Change of Lock Interrupt Register (COLIR)

The COLIR is 16-bit read/write register. It controls the change of lock interrupt generation, and is used for reporting a loss of lock interrupt source. It contains the interrupt request level and the interrupt status bit. This register is readable and writable at any time. A status bit is cleared by writing a one (writing a zero does not affect a status bit's value). The COLIR is memory mapped into the MPC565 / MPC566 USIU register map.

COLIR — Change of Lock Interrupt Register **0x2F C28C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
COLIRQ								COLIS	Re-served	COLIE	Reserved				
RESET:															
0	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U

U = Unaffected by reset

Table 8-12 COLIR Bit Descriptions

Bit(s)	Name	Description
0:7	COLIRQ	Change of lock interrupt request level. These bits determine the interrupt priority level of the change of lock. To specify certain level, the appropriate one of these bits should be set.
8	COLIS	If set ("one"), the bit indicates that a change in the PLL lock status was detected. The PLL was locked and lost lock, or the PLL was unlocked and got locked. The bit should be cleared by writing a one.
9	—	Reserved
10	COLIE	Change of Lock Interrupt enable. If COLIE bit is asserted, an interrupt will generate when the COLIS bit is asserted. 0 = Change of lock Interrupt disable 1 = Change of lock Interrupt enable
10:15	—	Reserved

8.12.4 VDDSRAM Control Register (VSRMCR)

This register contains control bits for enabling or disabling the VDDSRAM supply detection circuit. There are also four bits that indicate the failure detection. All four bits have the same function and are required to improve the detection capability in extreme cases.

VSRMCR — VDDSRAM Control Register **0x2F C290**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	LVSRS				VSRDE	LVDRS	RESERVED								
HARD RESET:															
	U	U	U	U	0	U									

U = Unaffected by reset

Table 8-13 VSRMCR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:4	LVSRS	Loss of VDDSRAM1 sticky. These status bits indicate whether a VDDSRAM1 supply failure occurred. In addition, when the power is turned on for the first time, VDDSRAM1 rises and these bits are set. The LVSRS bits are cleared by writing them to ones. A write of zero has no effect on these bits. 0 = No VDDSRAM1 supply failure was detected 1 = VDDSRAM1 supply failure was detected
5	VSRDE	VDDSRAM1 detector disable. 0 = VDDSRAM1 detection circuit is enabled 1 = VDDSRAM1 detection circuit is disabled
6	LVDRS	Loss of VDDSRAM1 for DECRAM Sticky — The status bit, dedicated especially for the BBC DECRAM, which indicates if there was VDDSRAM1 supply failure. When the power is turned on for the first time, VDDSRAM1 rises also and the bits will be asserted. The LVDECRAM bit can be cleared by writing ones to LVDECRAM. A write of zero has no effect on this bit. The bit may be used by application software, to decide if there is need to load decompression vocabularies during reset routine. 0 = VDDSRAM1 supply failure was not detected 1 = VDDSRAM1 supply failure was detected **NOTE**: The LVDRS bit is provided as a convenience for indicating that the DECRAM has lost power. It requires that the VDDSRAM1 and VDDSRAM3 pins are connected to the same power supply. It actually only monitors the VDDSRAM1 supply.
7:15	—	Reserved

SECTION 9
EXTERNAL BUS INTERFACE

The MPC565 / MPC566 external bus is a synchronous, burstable bus. Signals driven on this bus are required to make the setup and hold time relative to the bus clock's rising edge. The bus has the ability to support multiple masters. The MPC565 / MPC566 external bus interface architecture supports byte, half-word, and word operands allowing access to 8-, 16-, and 32-bit data ports through the use of synchronous cycles controlled by the size outputs (TSIZ0, TSIZ1). For accesses to 16- and 8-bit ports, the slave must be controlled by the memory controller.

9.1 Features

The external bus interface features are listed below.

- 32-bit address bus with transfer size indication (only 24 available on pins)
- 32-bit data bus
- Bus arbitration logic on-chip supports an external master
- Internal chip-select and wait state generation to support peripheral or static memory devices through the memory controller
- Supports various memory (SRAM, EEPROM) types: synchronous and asynchronous, burstable and non-burstable
- Supports non-wrap bursts with up to four data beats
- Flash ROM programming support
- Compatible with PowerPC™ architecture
- Easy to interface to slave devices
- Bus is synchronous (all signals are referenced to rising edge of bus clock)
- Bus can operate at the same frequency as the internal RCPU core of MPC565 / MPC566 or half the frequency.

9.2 Bus Transfer Signals

The bus transfers information between the MPC565 / MPC566 and external memory of a peripheral device. External devices can accept or provide 8, 16, and 32 data bits in parallel and must follow the handshake protocol described in this section. The maximum number of bits accepted or provided during a bus transfer is defined as the port width.

The MPC565 / MPC566 obtains an address lines that specify the address for the transfer and a data lines that transfer the data. Control signals indicate the beginning and type of the cycle, as well as the address space and size of the transfer. The selected

device then controls the length of the cycle with the signal(s) used to terminate the cycle. A strobe signal for the address lines indicates the validity of the address.

The MPC565 / MPC566 bus is synchronous. The bus and control input signals must be timed to setup and hold times relative to the rising edge of the clock. Bus cycles can be completed in two clock cycles.

For all inputs, the MPC565 / MPC566 latches the level of the input during a sample window around the rising edge of the clock signal. This window is illustrated in **Figure 9-1**, where **tsu** and **tho** are the input setup and hold times, respectively. To ensure that an input signal is recognized on a specific falling edge of the clock, that input must be stable during the sample window. If an input makes a transition during the window time period, the level recognized by the MPC565 / MPC566 is not predictable; however, the MPC565 / MPC566 always resolves the latched level to either a logic high or low before using it. In addition to meeting input setup and hold times for deterministic operation, all input signals must obey the protocols described in this section.

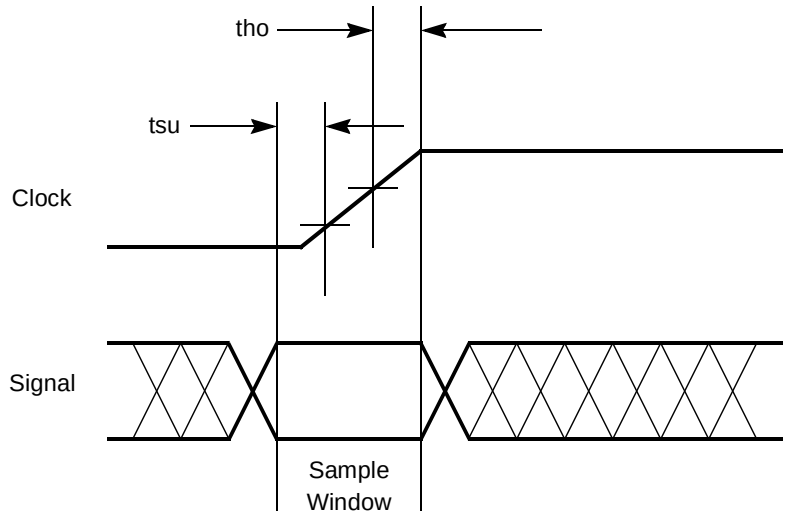


Figure 9-1 Input Sample Window

9.3 Bus Control Signals

The MPC565 / MPC566 initiates a bus cycle by driving the address, size, address type, cycle type, and read/write outputs. At the beginning of a bus cycle, TSIZ0 and TSIZ1 are driven with the address type signals. TSIZ0 and TSIZ1 indicate the number of bytes remaining to be transferred during an operand cycle (consisting of one or more bus cycles). These signals are valid at the rising edge of the clock in which the transfer start ($\overline{TS}$) signal is asserted.

The read/write (RD/$\overline{WR}$) signal determines the direction of the transfer during a bus cycle. Driven at the beginning of a bus cycle, RD/$\overline{WR}$ is valid at the rising edge of the clock in which $\overline{TS}$ is asserted. The logic level of RD/$\overline{WR}$ only changes when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for consecutive write cycles.

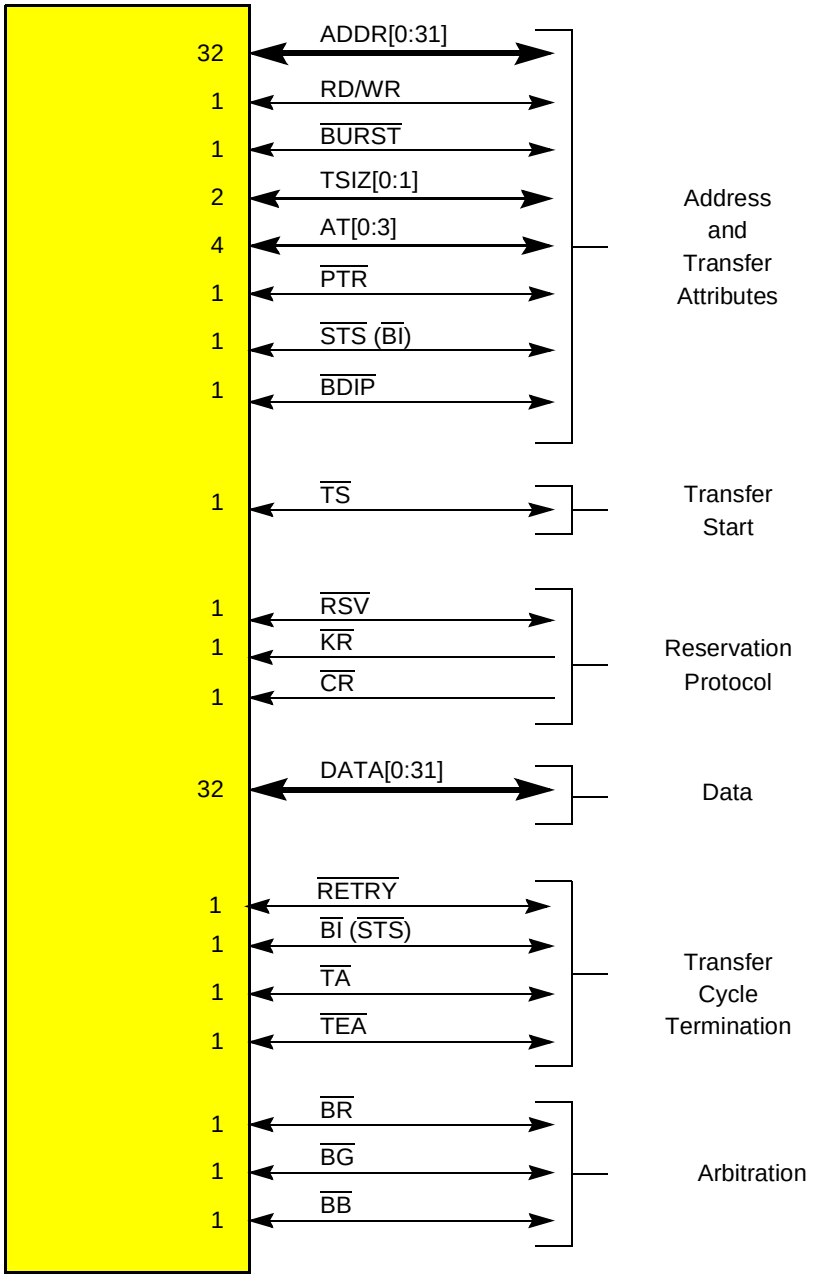


Figure 9-2 MPC565 / MPC566 Bus Signals

9.4 Bus Interface Signal Descriptions

Table 9-1 diatribes each signal in the bus interface unit. More detailed descriptions can be found in subsequent subsections. The buses are described in big endian manner, that means that bit 0 is most significant bit in a bus.

Table 9-1 MPC565 / MPC566 SIU Signals

Signal Name	Pins	Active	I/O	Description
Address and Transfer Attributes				
ADDR[0:31] Address bus	24 [8:31]	High	O	Specifies the physical address of the bus transaction.
			I	Driven by an external bus master when it owns the external bus. Only for testing purposes.
RD/$\overline{\text{WR}}$ Read/write	1	HIgh	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Driven high indicates that a read access is in progress. Driven low indicates that a write access is in progress.
			I	Driven by an external master when it owns the external bus. Driven high indicates that a read access is in progress. Driven low indicates that a write access is in progress.
$\overline{\text{BURST}}$ Burst transfer	1	Low	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Driven low indicates that a burst transfer is in progress. Driven high indicates that the current transfer is not a burst.
			I	Driven by an external master when it owns the external bus. Driven low indicates that a burst transfer is in progress. Driven high indicates that the current transfer is not a burst. The MPC565 / MPC566 does not support burst accesses to internal slaves.
TSIZ[0:1] Transfer size	2	High	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Specifies the data transfer size for the transaction.
			I	Driven by an external master when it owns the external bus. Specifies the data transfer size for the transaction.
AT[0:3] Address type	3	High	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.
RSV Reservation transfer	1	Low	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.
PTR Program trace	1	High	O	Driven by the MPC565 / MPC566 along with the address when it owns the external bus. Indicates additional information about the address on the current transaction.
			I	Only for testing purposes.

Table 9-1 MPC565 / MPC566 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
$\overline{\text{RETRY}}$	1	Low	I	In the case of regular transaction, this signal is driven by the slave device to indicate that the MPC565 / MPC566 must relinquish the ownership of the bus and retry the cycle.
			O	When an external master owns the bus and the internal MPC565 / MPC566 bus initiates access to the external bus at the same time, this signal is used to cause the external master to relinquish the bus for one clock to solve the contention.
$\overline{\text{BDIP}}$ Burst data in progress	1	Low	O	Driven by the MPC565 / MPC566 when it owns the external bus. It is part of the burst protocol. When $\overline{\text{BDIP}}$ is asserted, the second beat in front of the current one is requested by the master. This signal is negated prior to the end of a burst to terminate the burst data phase early.
			I	Driven by an external master when it owns the external bus. When $\overline{\text{BDIP}}$ is asserted, the second beat in front of the current one is requested by the master. This signal is negated prior to the end of a burst to terminate the burst data phase early. The MPC565 / MPC566 does not support burst accesses to internal slaves.
Transfer Start				
$\overline{\text{TS}}$ Transfer start	1	Low	O	Driven by the MPC565 / MPC566 when it owns the external bus. Indicates the start of a transaction on the external bus.
			I	Driven by an external master when it owns the external bus. It indicates the start of a transaction on the external bus or (in show cycle mode) signals the beginning of an internal transaction.
$\overline{\text{STS}}$ Special transfer start	1	Low	O	Driven by the MPC565 / MPC566 when it owns the external bus. Indicates the start of a transaction on the external bus or signals the beginning of an internal transaction in show cycle mode.
Reservation Protocol				
CR Cancel reservation	1	Low	I	Each PowerPC CPU has its own $\overline{\text{CR}}$ signal. Assertion of $\overline{\text{CR}}$ instructs the bus master to clear its reservation; some other master has touched its reserved space. This is a pulsed signal.
$\overline{\text{KR}}$ Kill reservation	1	Low	I	In case of a bus cycle initiated by a STWCX instruction issued by the RCPU to a non-local bus on which the storage reservation has been lost, this signal is used by the non-local bus interface to back-off the cycle. Refer to **9.5.9 Storage Reservation** for details.

Table 9-1 MPC565 / MPC566 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
Data				
DATA[0:31] Data bus	32	High		The data bus has the following byte lane assignments: Data Byte — Byte Lane DATA[0:7] — 0 DATA[8:15] — 1 DATA[16:23] — 2 DATA[24:31] — 3
			O	Driven by the MPC565 / MPC566 when it owns the external bus and it initiated a write transaction to a slave device. For single beat transactions, the byte lanes not selected for the transfer by ADDR[30:31] and TSIZ[0:1] do not supply valid data. In addition, the MPC565 / MPC566 drives DATA[0:31] when an external master owns the external bus and initiated a read transaction to an internal slave module.
			I	Driven by the slave in a read transaction. For single beat transactions, the MPC565 / MPC566 does not sample byte lanes that are not selected for the transfer by ADDR[30:31] and TSIZ[0:1]. In addition, an external master that owns the bus and initiated a write transaction to an internal slave module drives DATA[0:31].
Transfer Cycle Termination				
$\overline{\text{TA}}$ Transfer acknowledge	1	LOW	I	Driven by the slave device to which the current transaction was addressed. Indicates that the slave has received the data on the write cycle or returned data on the read cycle. If the transaction is a burst, $\overline{\text{TA}}$ should be asserted for each one of the transaction beats.
			O	Driven by the MPC565 / MPC566 when the slave device is controlled by the on-chip memory controller or when an external master initiated a transaction to an internal slave module.
$\overline{\text{TEA}}$ Transfer error acknowledge	1	Low	I	Driven by the slave device to which the current transaction was addressed. Indicates that an error condition has occurred during the bus cycle.
			O	Driven by the MPC565 / MPC566 when the internal bus monitor detected an erroneous bus condition, or when an external master initiated a transaction to an internal slave module and an internal error was detected.
$\overline{\text{BI}}$ Burst inhibit	1	Low	I	Driven by the slave device to which the current transaction was addressed. Indicates that the current slave does not support burst mode.
			O	Driven by the MPC565 / MPC566 when the slave device is controlled by the on-chip memory controller. the MPC565 / MPC566 also asserts $\overline{\text{BI}}$ for any external master burst access to internal MPC565 / MPC566 memory space.

Table 9-1 MPC565 / MPC566 SIU Signals (Continued)

Signal Name	Pins	Active	I/O	Description
ARBITRATION				
$\overline{BR}$ Bus request	1	Low	I	When the internal arbiter is enabled, $\overline{BR}$ assertion indicates that an external master is requesting the bus.
			O	Driven by the MPC565 / MPC566 when the internal arbiter is disabled and the chip is not parked.
$\overline{BG}$ Bus grant	1	Low	O	When the internal arbiter is enabled, the MPC565 / MPC566 asserts this signal to indicate that an external master may assume ownership of the bus and begin a bus transaction. The $\overline{BG}$ signal should be qualified by the master requesting the bus in order to ensure it is the bus owner: Qualified bus grant = $\overline{BG}$ & ~ $\overline{BB}$
			I	When the internal arbiter is disabled, $\overline{BG}$ is sampled and properly qualified by the MPC565 / MPC566 when an external bus transaction is to be executed by the chip.
$\overline{BB}$ Bus busy	1	Low	O	When the internal arbiter is enabled, the MPC565 / MPC566 asserts this signal to indicate that it is the current owner of the bus. When the internal arbiter is disabled, the MPC565 / MPC566 asserts this signal after the external arbiter has granted the ownership of the bus to the chip and it is ready to start the transaction.
			I	When the internal arbiter is enabled, the MPC565 / MPC566 samples this signal to get indication of when the external master ended its bus tenure ($\overline{BB}$ negated). When the internal arbiter is disabled, the $\overline{BB}$ is sampled to properly qualify the $\overline{BG}$ line when an external bus transaction is to be executed by the chip.

9.5 Bus Operations

This section provides a functional description of the system bus, the signals that control it, and the bus cycles provided for data transfer operations. It also describes the error conditions, bus arbitration, and reset operation.

The MPC565 / MPC566 generates a system clock output (CLKOUT). This output sets the frequency of operation for the bus interface directly. Internally, the MPC565 / MPC566 uses a phase-lock loop (PLL) circuit to generate a master clock for all of the MPC565 / MPC566 circuitry (including the bus interface) which is phase-locked to the CLKOUT output signal.

All signals for the MPC565 / MPC566 bus interface are specified with respect to the rising edge of the external CLKOUT and are guaranteed to be sampled as inputs or changed as outputs with respect to that edge. Since the same clock edge is referenced for driving or sampling the bus signals, the possibility of clock skew could exist

between various modules in a system due to routing or the use of multiple clock lines. It is the responsibility of the system to handle any such clock skew problems that could occur.

9.5.1 Basic Transfer Protocol

The basic transfer protocol defines the sequence of actions that must occur on the MPC565 / MPC566 bus to perform a complete bus transaction. A simplified scheme of the basic transfer protocol is illustrated in **Figure 9-3**.

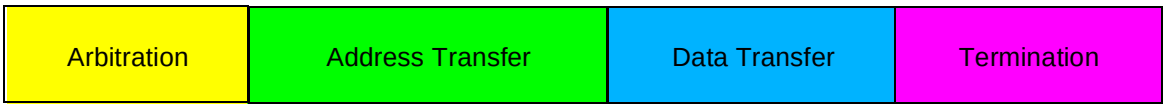


Figure 9-3 Basic Transfer Protocol

The basic transfer protocol provides for an arbitration phase and an address and data transfer phase. The address phase specifies the address for the transaction and the transfer attributes that describe the transaction. The data phase performs the transfer of data (if any is to be transferred). The data phase may transfer a single beat of data (four bytes or less) for nonburst operations, a 4-beat burst of data (4 x 4 bytes), an 8-beat burst of data (8 x 2 bytes) or a 16-beat burst of data (16 x 1 bytes).

9.5.2 Single Beat Transfer

During the data transfer phase, the data is transferred from master to slave (in write cycles) or from slave to master (on read cycles).

During a write cycle, the master drives the data as soon as it can, but never earlier than the cycle following the address transfer phase. The master has to take into consideration the “one dead clock cycle” switching between drivers to avoid electrical contentions. The master can stop driving the data bus as soon as it samples the $\overline{TA}$ line asserted on the rising edge of the CLKOUT.

During a read cycle, the master accepts the data bus contents as valid at the rising edge of the CLKOUT in which the $\overline{TA}$ signal is sampled/asserted.

9.5.2.1 Single Beat Read Flow

The basic read cycle begins with a bus arbitration, followed by the address transfer, then the data transfer. The handshakes are illustrated in the following flow and timing diagrams as applicable to the fixed transaction protocol.

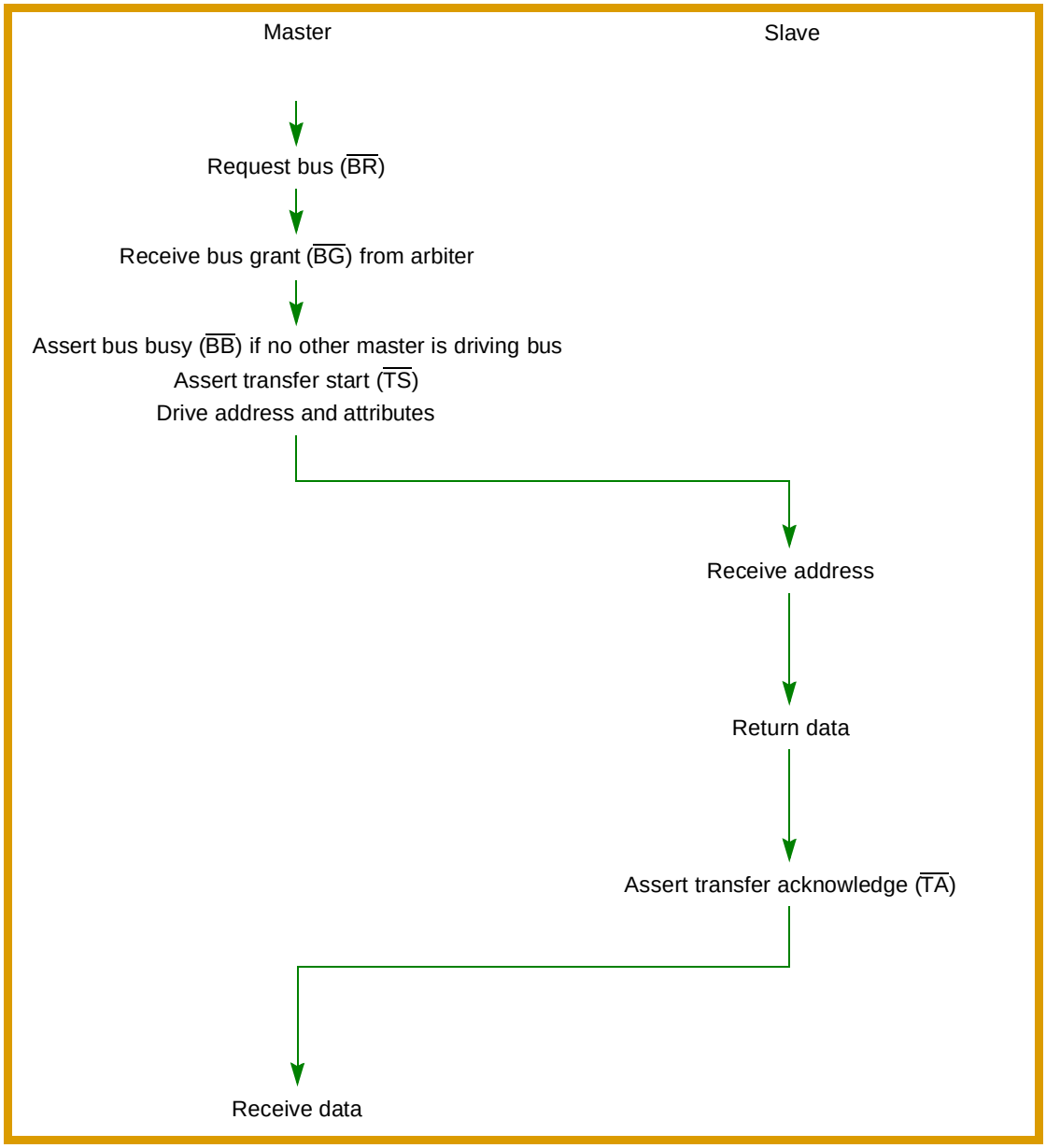


Figure 9-4 Basic Flow Diagram of a Single Beat Read Cycle

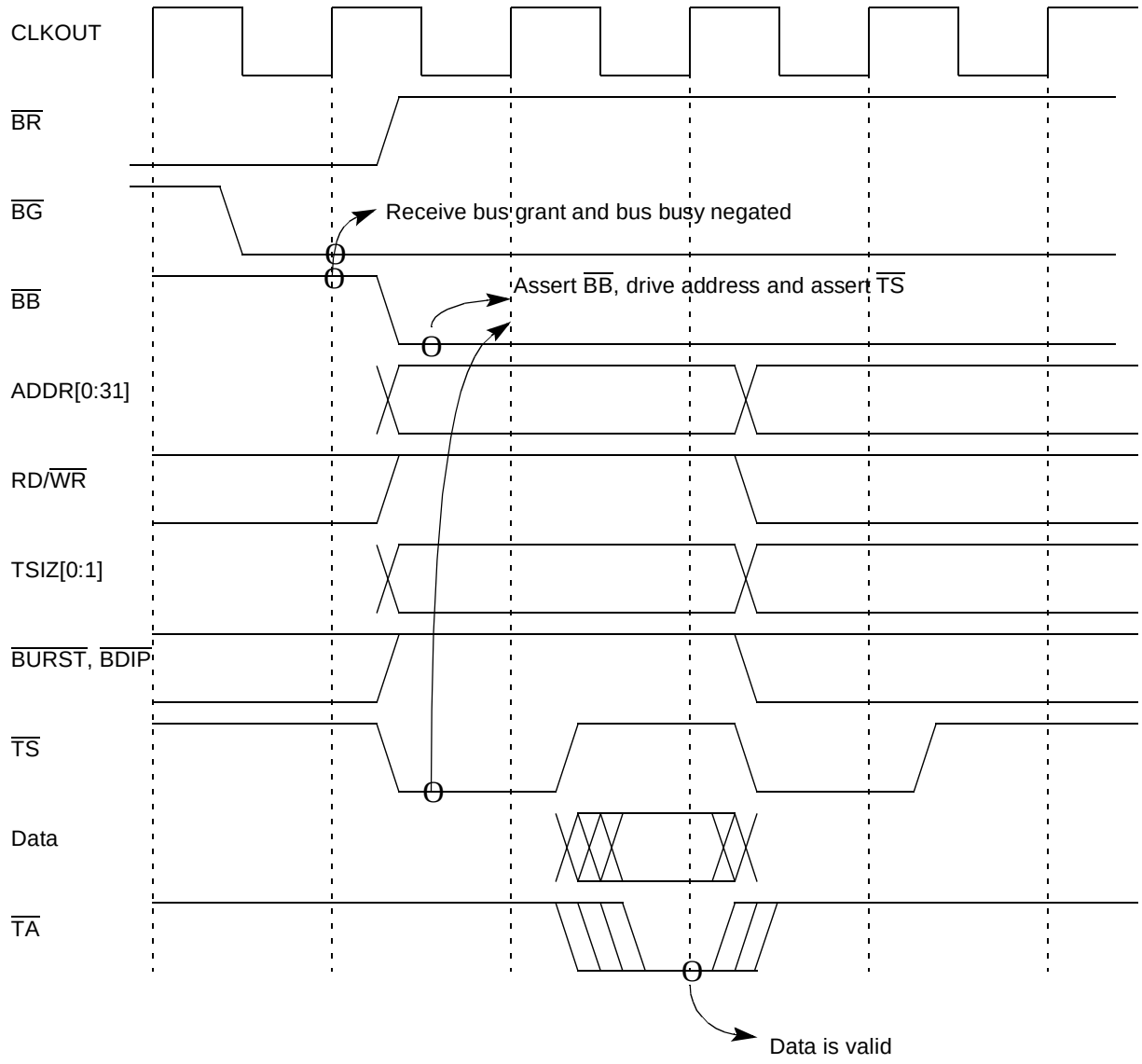


Figure 9-5 Single Beat Read Cycle–Basic Timing–Zero Wait States

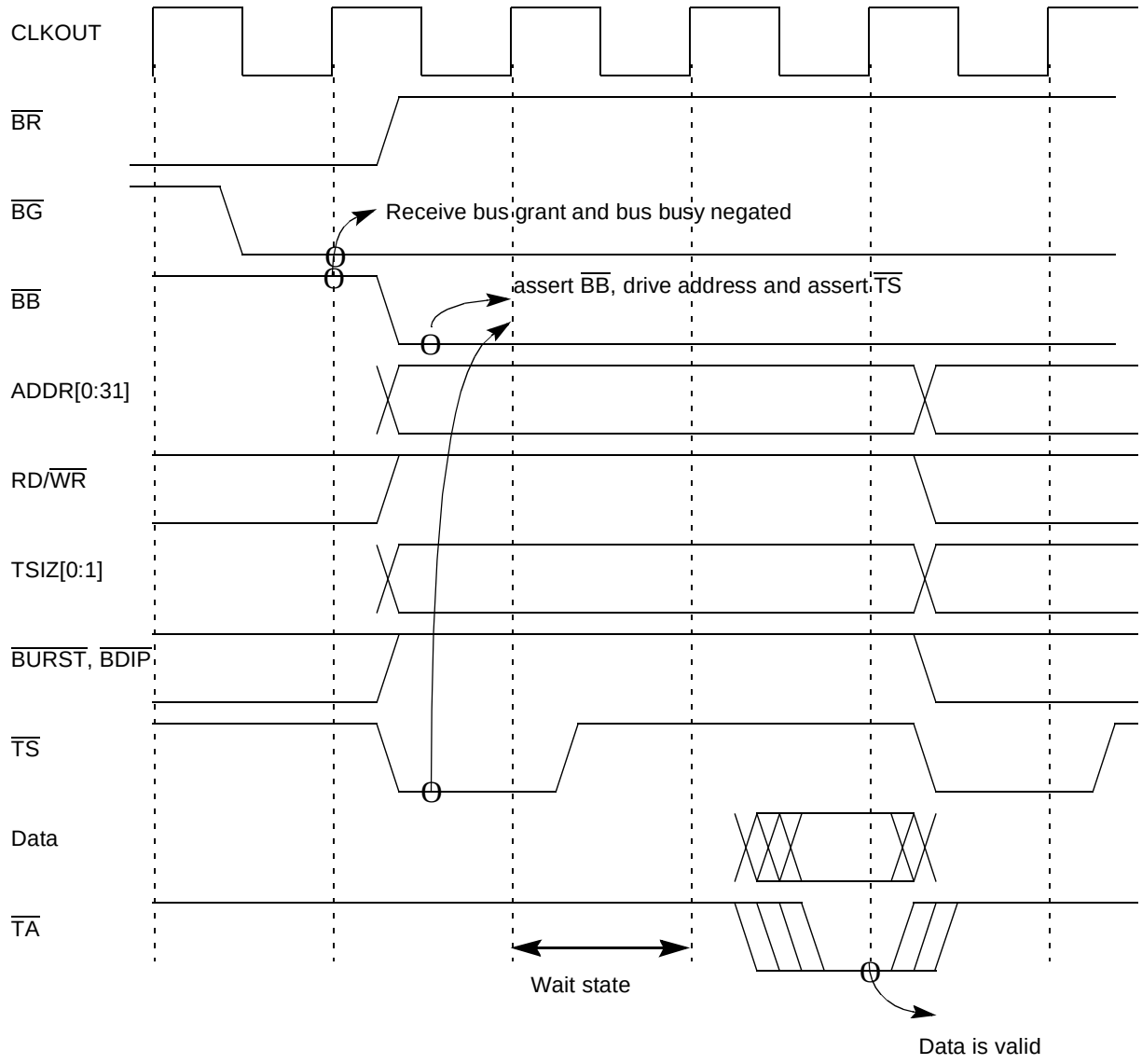


Figure 9-6 Single Beat Read Cycle – Basic Timing – One Wait State

9.5.2.2 Single Beat Write Flow

The basic write cycle begins with a bus arbitration, followed by the address transfer, then the data transfer. The handshakes are illustrated in the following flow and timing diagrams as applicable to the fixed transaction protocol.

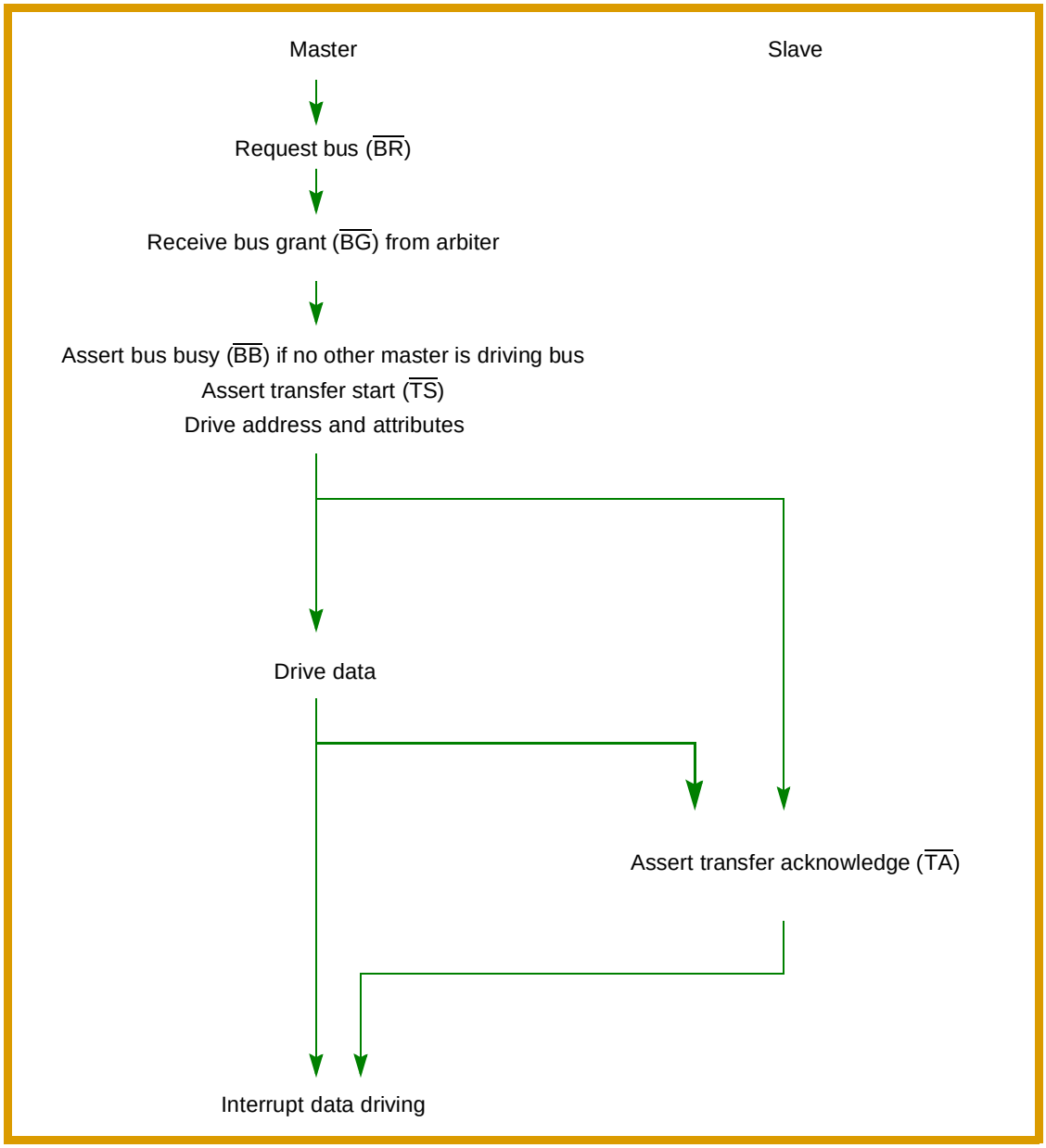


Figure 9-7 Basic Flow Diagram of a Single Beat Write Cycle

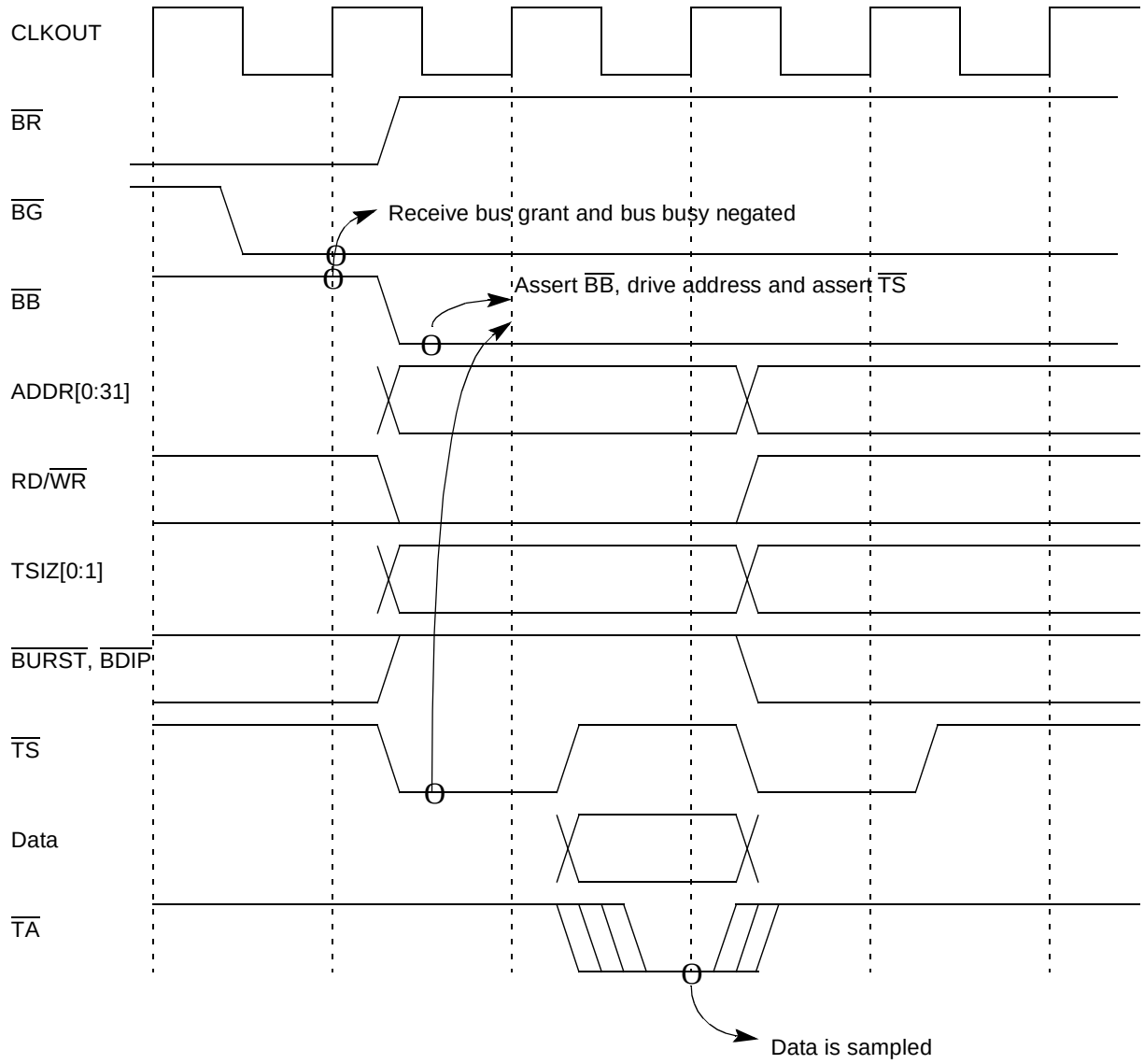


Figure 9-8 Single Beat Basic Write Cycle Timing — Zero Wait States

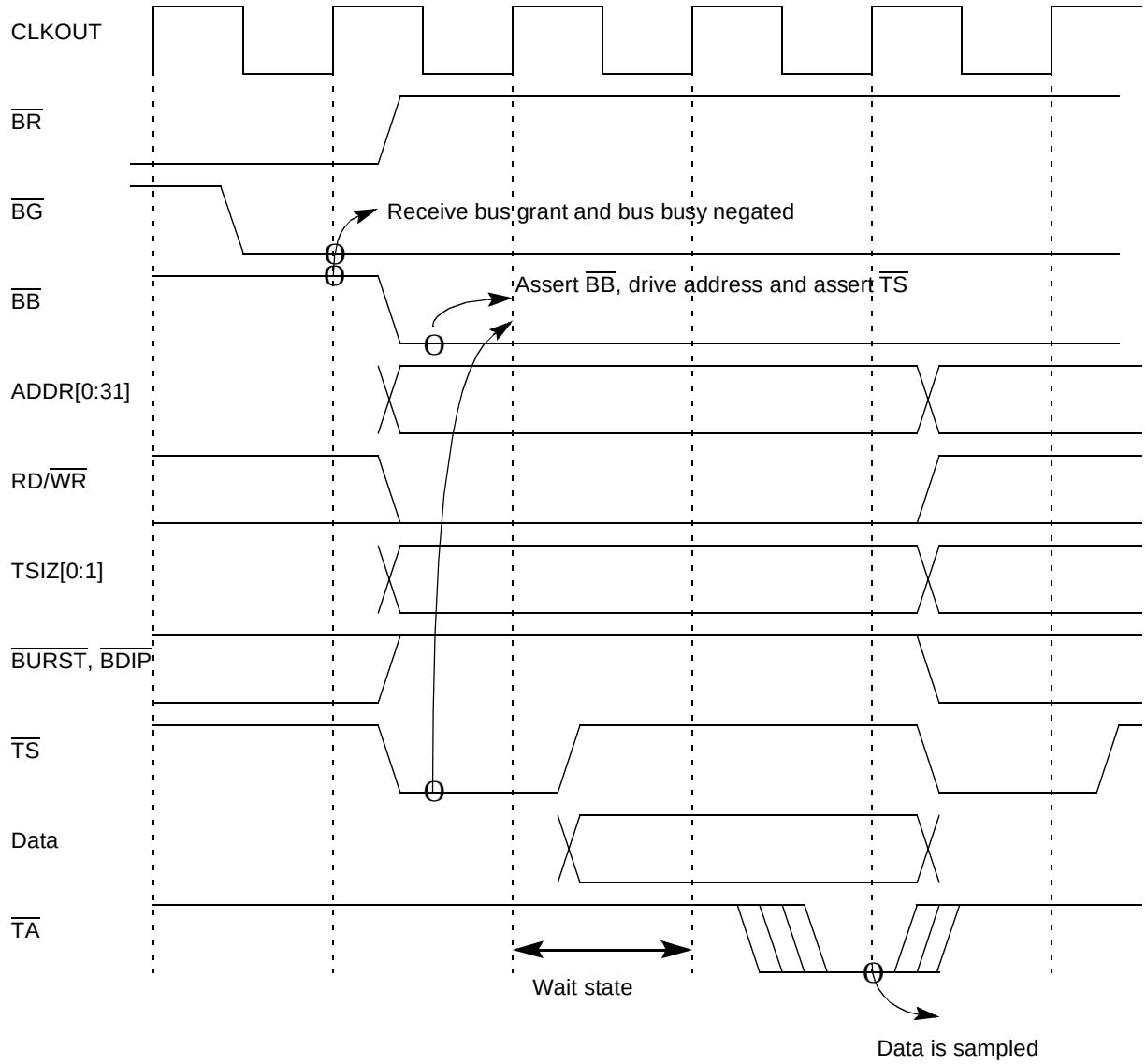


Figure 9-9 Single Beat Basic Write Cycle Timing — One Wait State

9.5.2.3 Single Beat Flow with Small Port Size

The general case of single beat transfers assumes that the external memory has a 32-bit port size. The MPC565 / MPC566 provides an effective mechanism for interfacing with 16-bit and 8-bit port size memories, allowing transfers to these devices when they are controlled by the internal memory controller.

In this case, the MPC565 / MPC566 attempts to initiate a transfer as in the normal case. If the bus interface receives a small port size (16 or 8 bits) indication before the transfer acknowledge to the first beat (through the internal memory controller), the MCU initiates successive transactions until the completion of the data transfer. Note that all the transactions initiated to complete the data transfer are considered to be part of an atomic transaction, so the MCU does not allow other unrelated master accesses

or bus arbitration to intervene between the transfers. If any of the transactions except the first is re-tried during an access to a small port, then an exception is generated to the RCPU.

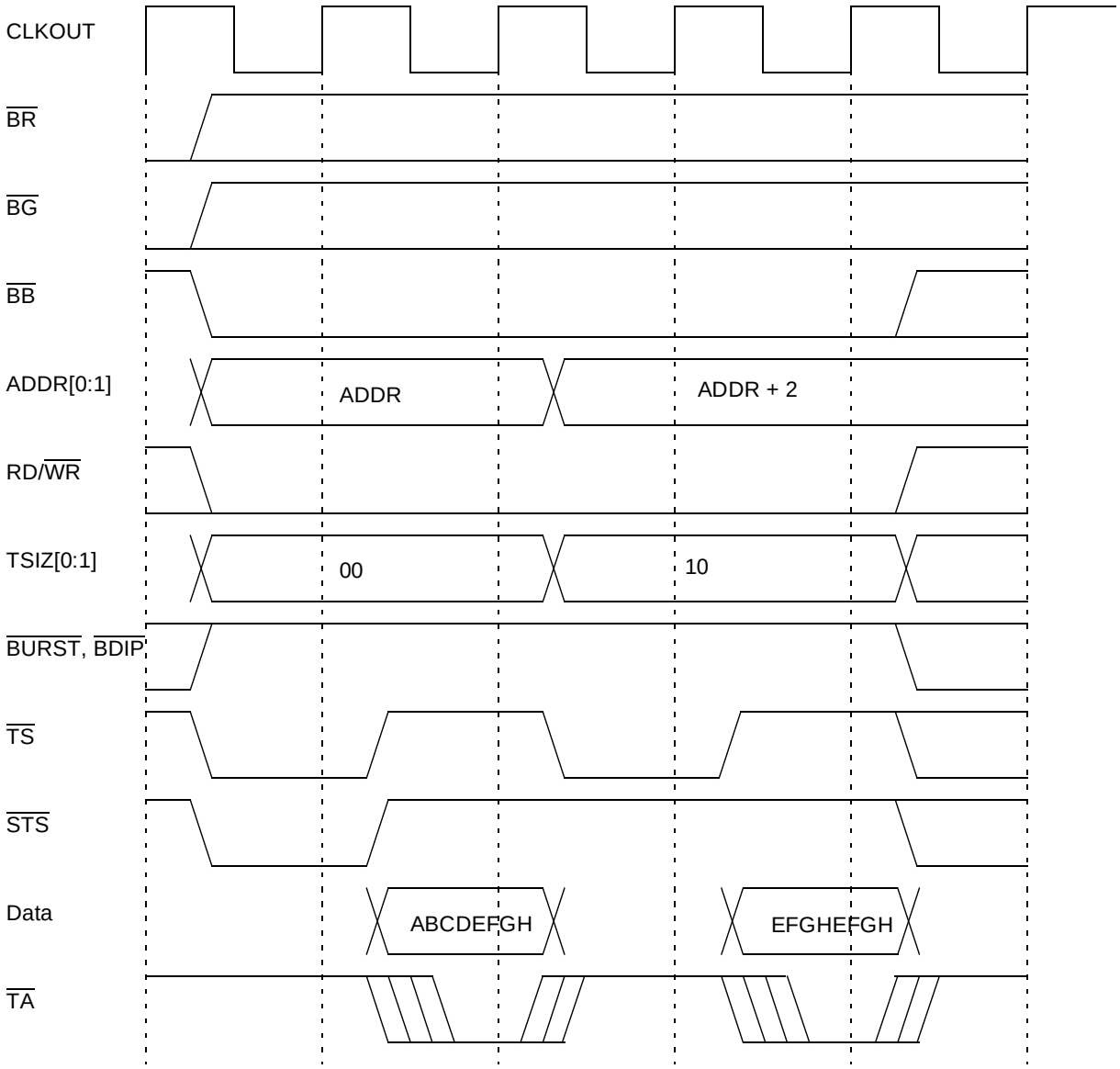


Figure 9-10 Single Beat 32-Bit Data Write Cycle Timing — 16-Bit Port Size

9.5.3 Burst Transfer

The MPC565 / MPC566 uses non-wrapping burst transfers to access operands of up to 16 bytes (four words). A non-wrapping burst access stops accessing the external device when the word address is modulo four. The MPC565 / MPC566 begins the access by supplying a starting address that points to one of the words and requiring the memory device to sequentially drive or sample each word on the data bus. The

selected slave device must internally increment ADDR[28] and ADDR[29] (and ADDR[30]in the case of a 16-bit port slave device, and also ADDR[31] in the case of an 8-bit port slave device) of the supplied address for each transfer, causing the address to reach a four-word boundary, and then stop. The address and transfer attributes supplied by the MPC565 / MPC566 remain stable during the transfers. The selected device terminates each transfer by driving or sampling the word on the data bus and asserting $\overline{TA}$.

The MPC565 / MPC566 also supports burst-inhibited transfers for slave devices that are unable to support bursting. For this type of bus cycle, the selected slave device supplies or samples the first word the MPC565 / MPC566 points to and asserts the burst-inhibit signal with $\overline{TA}$ for the first transfer of the burst access. The MPC565 / MPC566 responds by terminating the burst and accessing the remainder of the 16-byte block. These remaining accesses use up to three read/write bus cycles (each one for a word) in the case of a 32-bit port width slave, up to seven read/write bus cycles in the case of a 16-bit port width slave, or up to fifteen read/write bus cycles in the case of a 8-bit port width slave.

The general case of burst transfers assumes that the external memory has a 32-bit port size. The MPC565 / MPC566 provides an effective mechanism for interfacing with 16-bit port size memories and 8-bit port size memories allowing bursts transfers to these devices when they are controlled by the internal memory controller.

In this case, the MPC565 / MPC566 attempts to initiate a burst transfer as in the normal case. If the memory controller signals to the bus interface that the external device has a small port size (8 or 16 bits), and if the burst is accepted, the bus interface completes a burst of 8 or 16 beats.

Each of the data beats of the burst transfers effectively only one or two bytes. Note that this burst of 8 or 16 beats is considered an atomic transaction, so the MPC565 / MPC566 does not allow other unrelated master accesses or bus arbitration to intervene between the transfers.

9.5.4 Burst Mechanism

In addition to the standard bus signals, the MPC565 / MPC566 burst mechanism uses the following signals:

- The $\overline{BURST}$ signal indicates that the cycle is a burst cycle.
- The burst data in progress ($\overline{BDIP}$) signal indicates the duration of the burst data.
- The burst inhibit ($\overline{BI}$) signal indicates whether the slave is burstable.

At the start of the burst transfer, the master drives the address, the address attributes, and the $\overline{BURST}$ signal to indicate that a burst transfer is being initiated, and asserts $\overline{TS}$. If the slave is burstable, it negates the burst-inhibit ($\overline{BI}$) signal. If the slave cannot burst, it asserts $\overline{BI}$.

During the data phase of a burst write cycle the master drives the data. It also asserts $\overline{BDIP}$ if it intends to drive the data beat following the current data beat. When the slave

has received the data, it asserts the signal transfer acknowledge to indicate to the master that it is ready for the next data transfer. The master again drives the next data and asserts or negates the $\overline{\text{BDIP}}$ signal. If the master does not intend to drive another data beat following the current one, it negates $\overline{\text{BDIP}}$ to indicate to the slave that the next data beat transfer is the last data of the burst write transfer.

$\overline{\text{BDIP}}$ has two basic timings: normal and late (see **Figure 9-13** and **Figure 9-14**). In the late timing mode, assertion of $\overline{\text{BDIP}}$ is delayed by the number of wait states in the first data beat. This implies that for zero-wait-state cycles, $\overline{\text{BDIP}}$ assertion time is identical in normal and late modes. Cycles with late $\overline{\text{BDIP}}$ generation can occur only during cycles for which the memory controller generates $\overline{\text{TA}}$ internally. Refer to **SECTION 10 MEMORY CONTROLLER** for more information.

In the MPC565 / MPC566, no internal master initiates write bursts. The MPC565 / MPC566 is designed to perform this kind of transaction in order to support an external master that is using the memory controller services. Refer to **10.7 Memory Controller External Master Support**.

During the data phase of a burst read cycle, the master receives data from the addressed slave. If the master needs more than one data beat, it asserts $\overline{\text{BDIP}}$. Upon receiving the second-to-last data beat, the master negates $\overline{\text{BDIP}}$. The slave stops driving new data after it receives the negation of the $\overline{\text{BDIP}}$ signal at the rising edge of the clock.

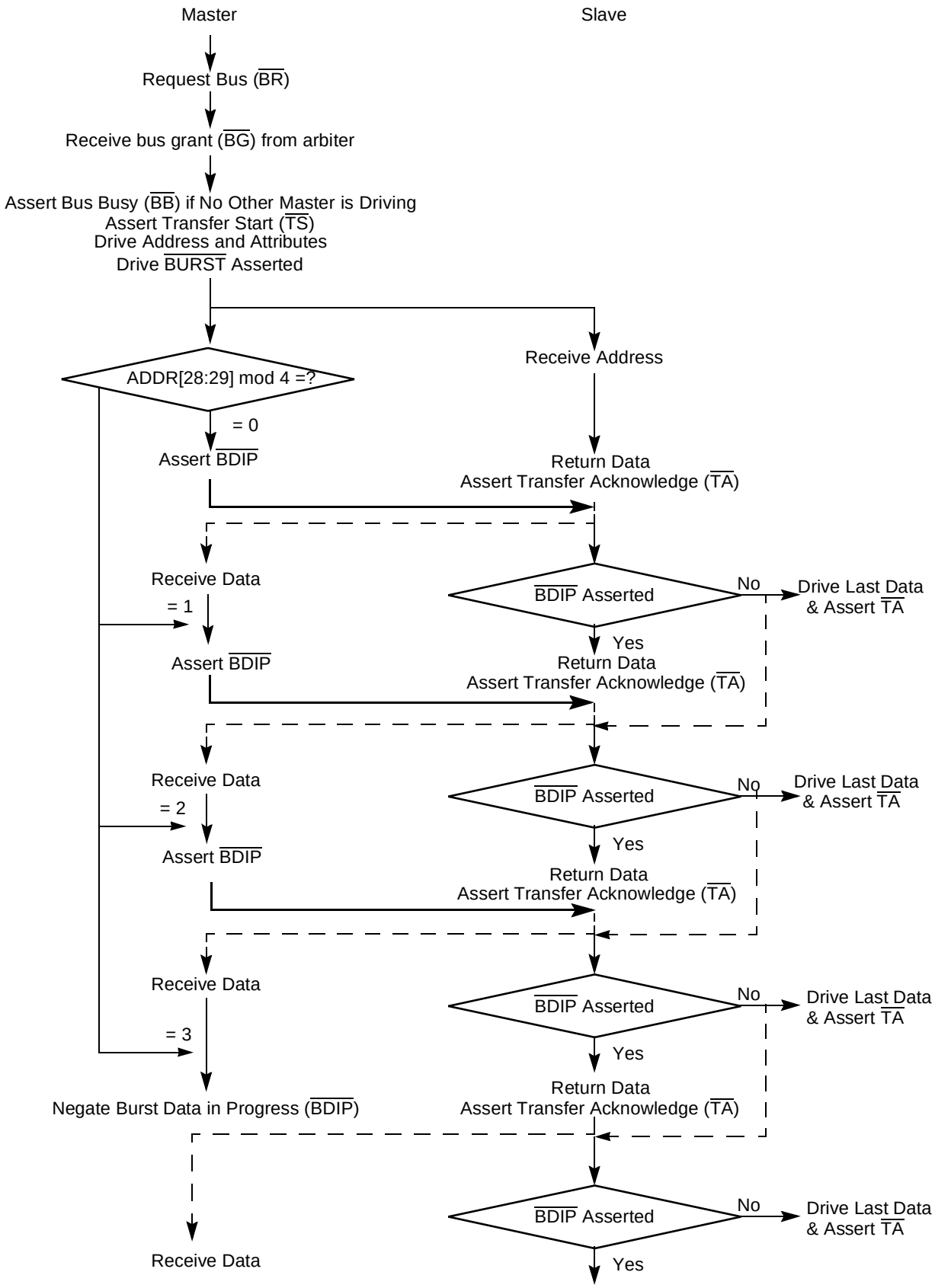


Figure 9-11 Basic Flow Diagram Of A Burst Read Cycle

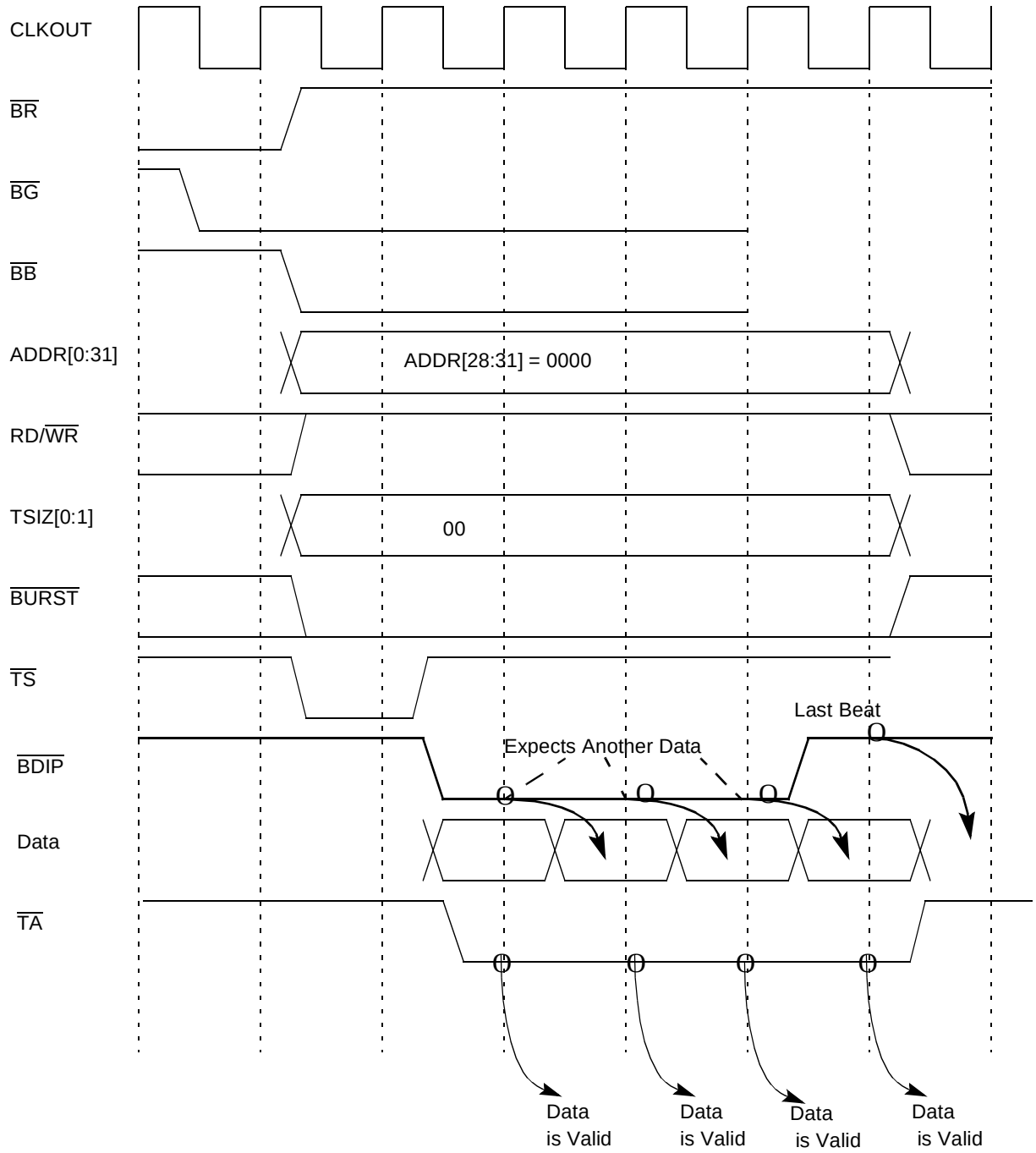


Figure 9-12 Burst-Read Cycle — 32-Bit Port Size — Zero Wait State

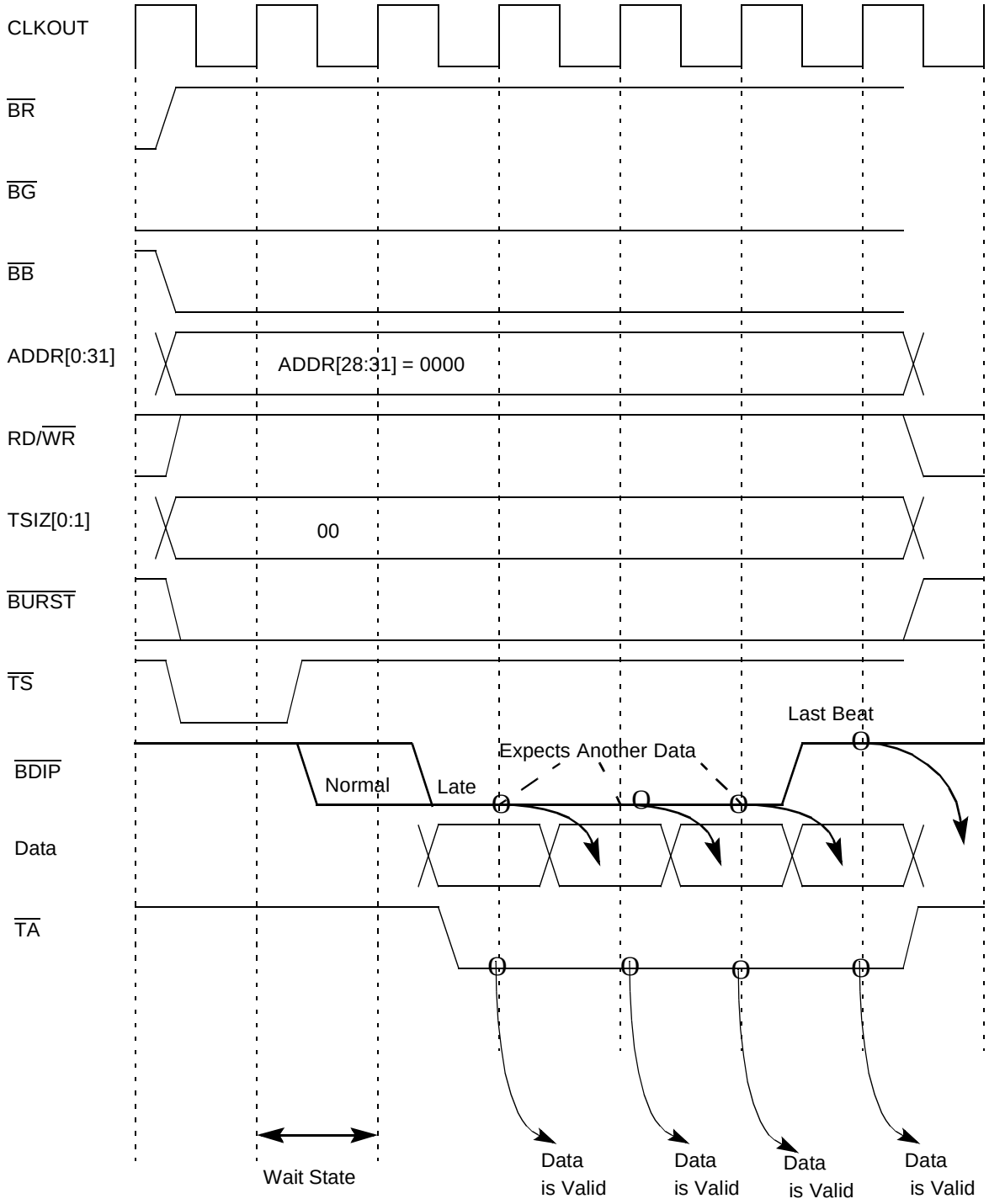


Figure 9-13 Burst-Read Cycle — 32-Bit Port Size — One Wait State

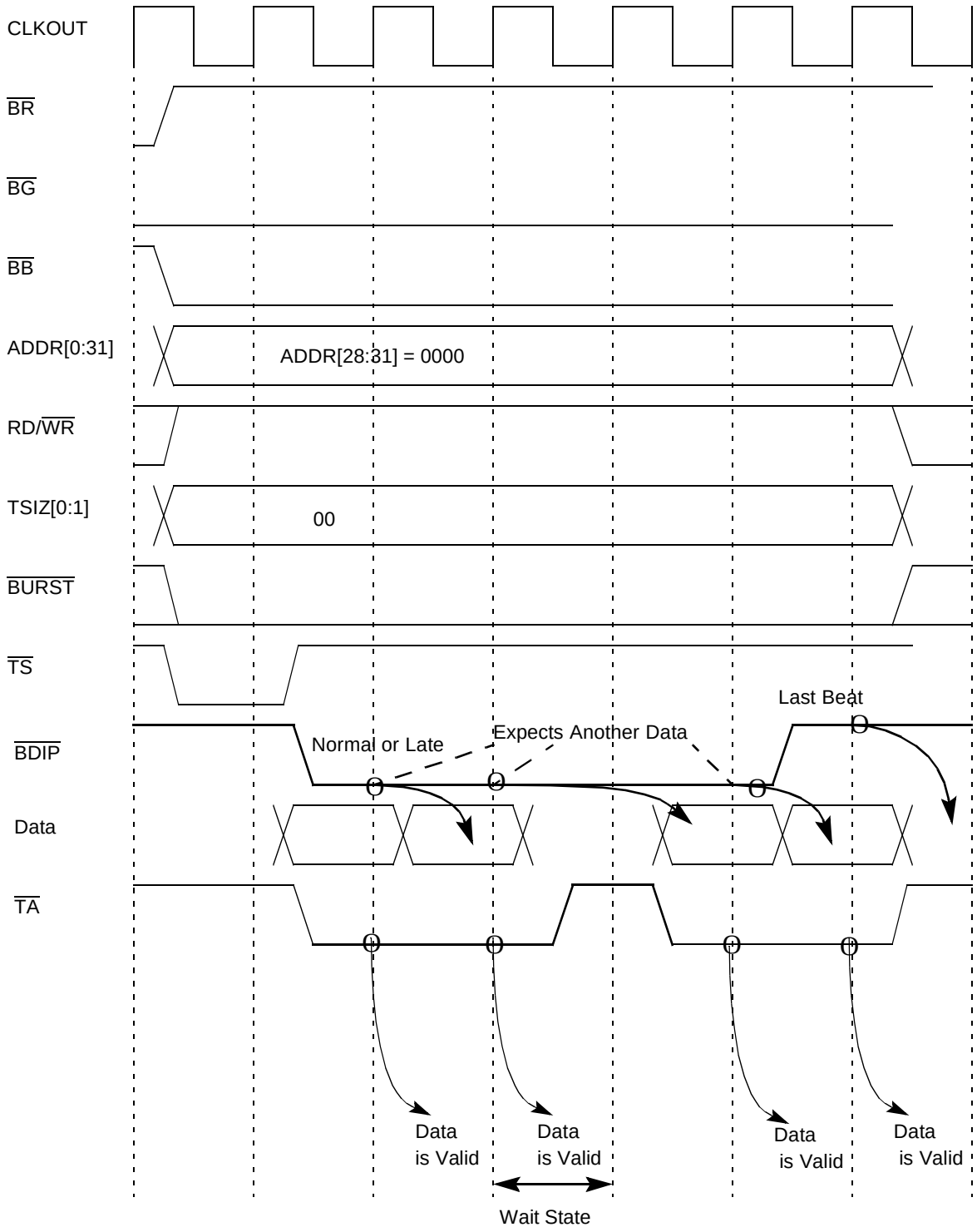


Figure 9-14 Burst-Read Cycle — 32-Bit Port Size — Wait States Between Beats

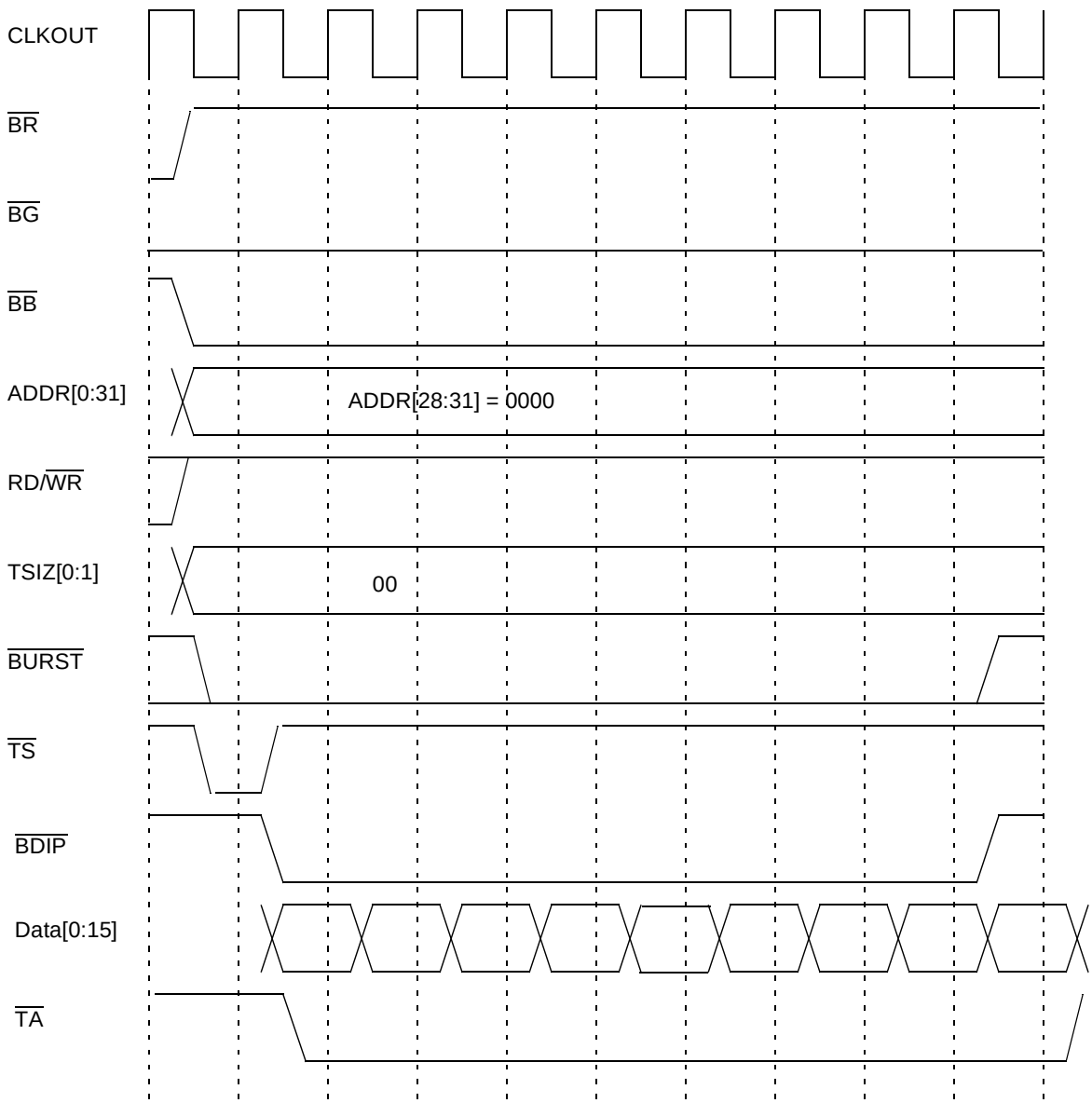


Figure 9-15 Burst-Read Cycle — 16-Bit Port Size

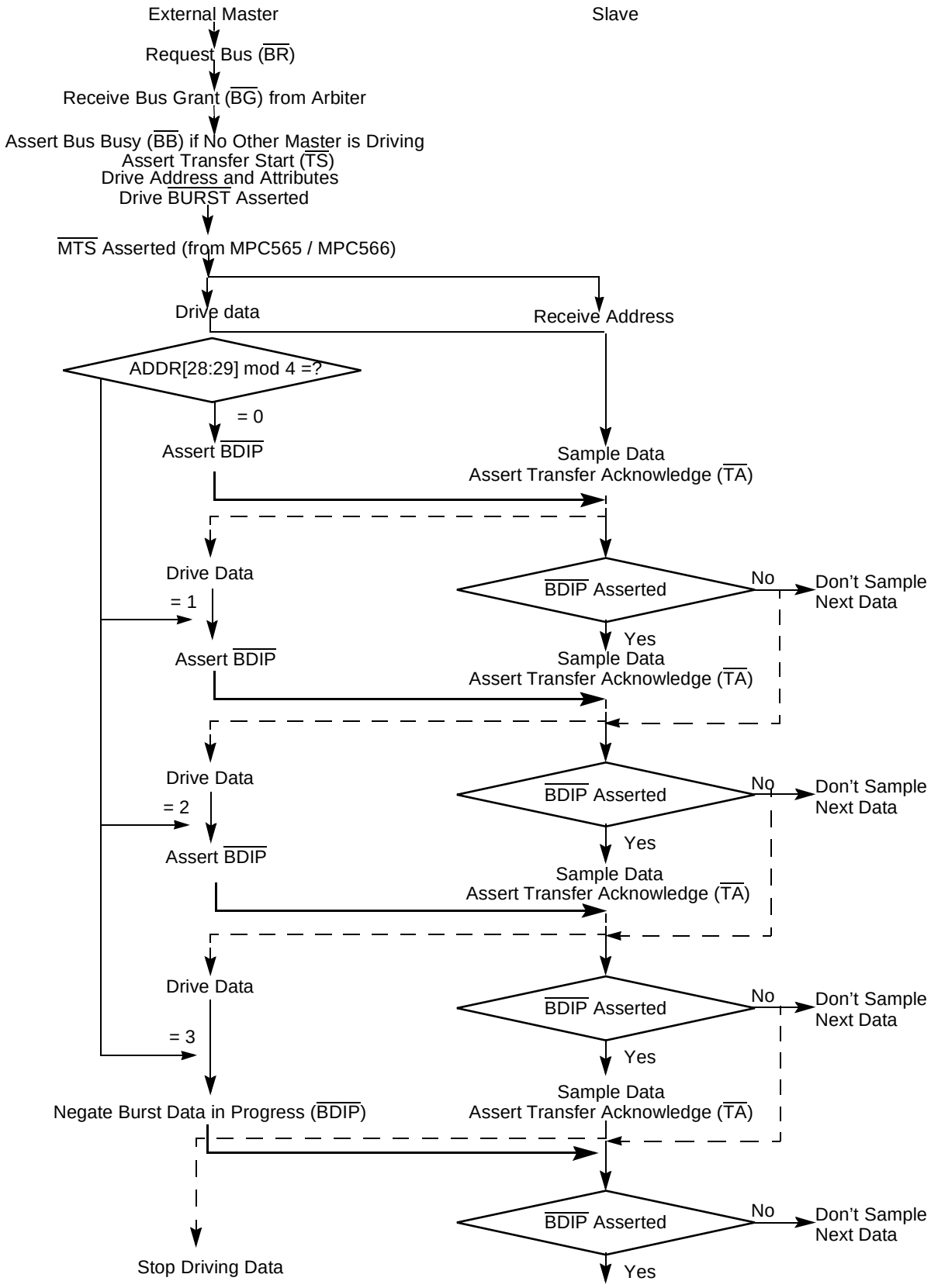


Figure 9-16 Basic Flow Diagram of a Burst Write Cycle

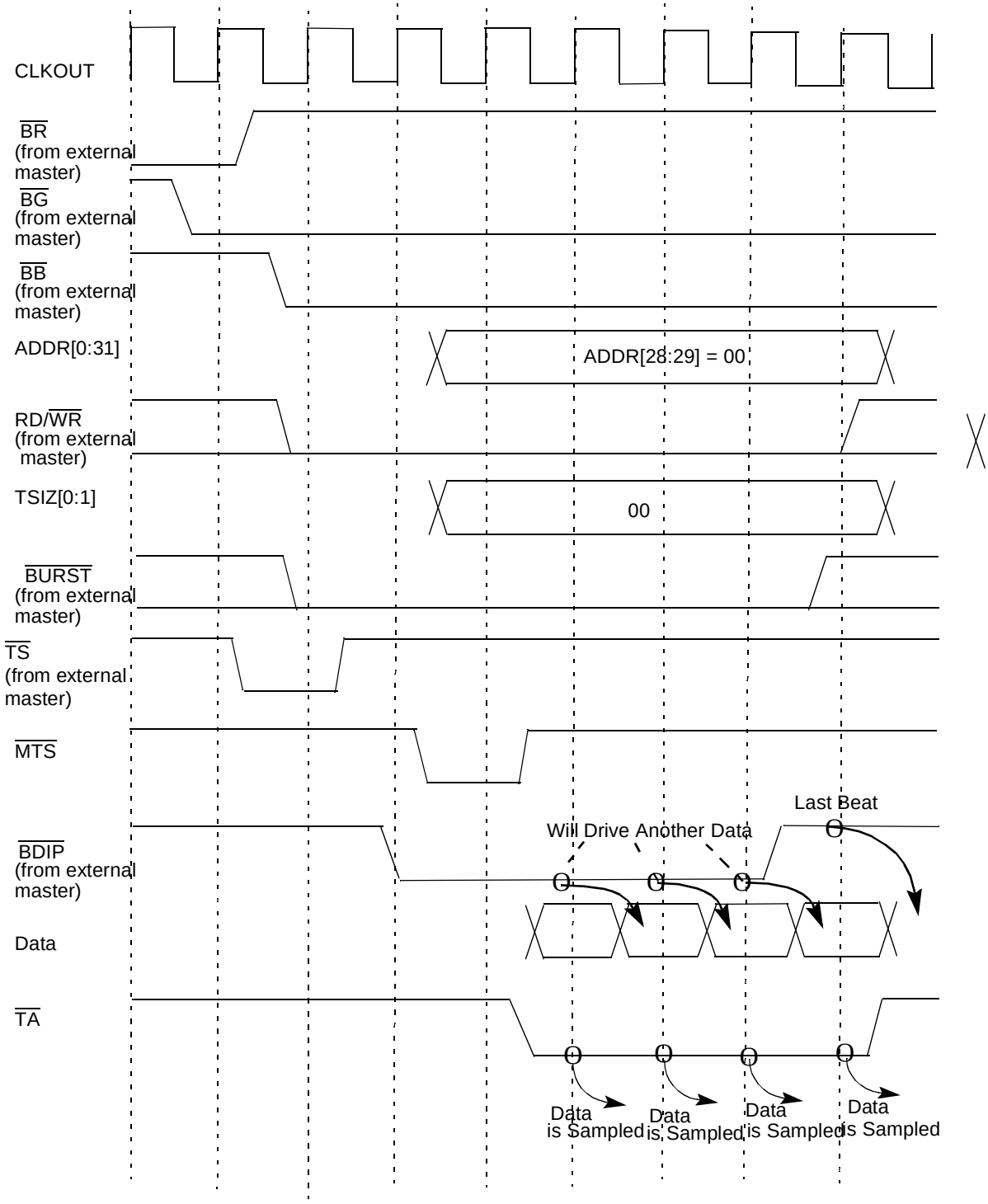
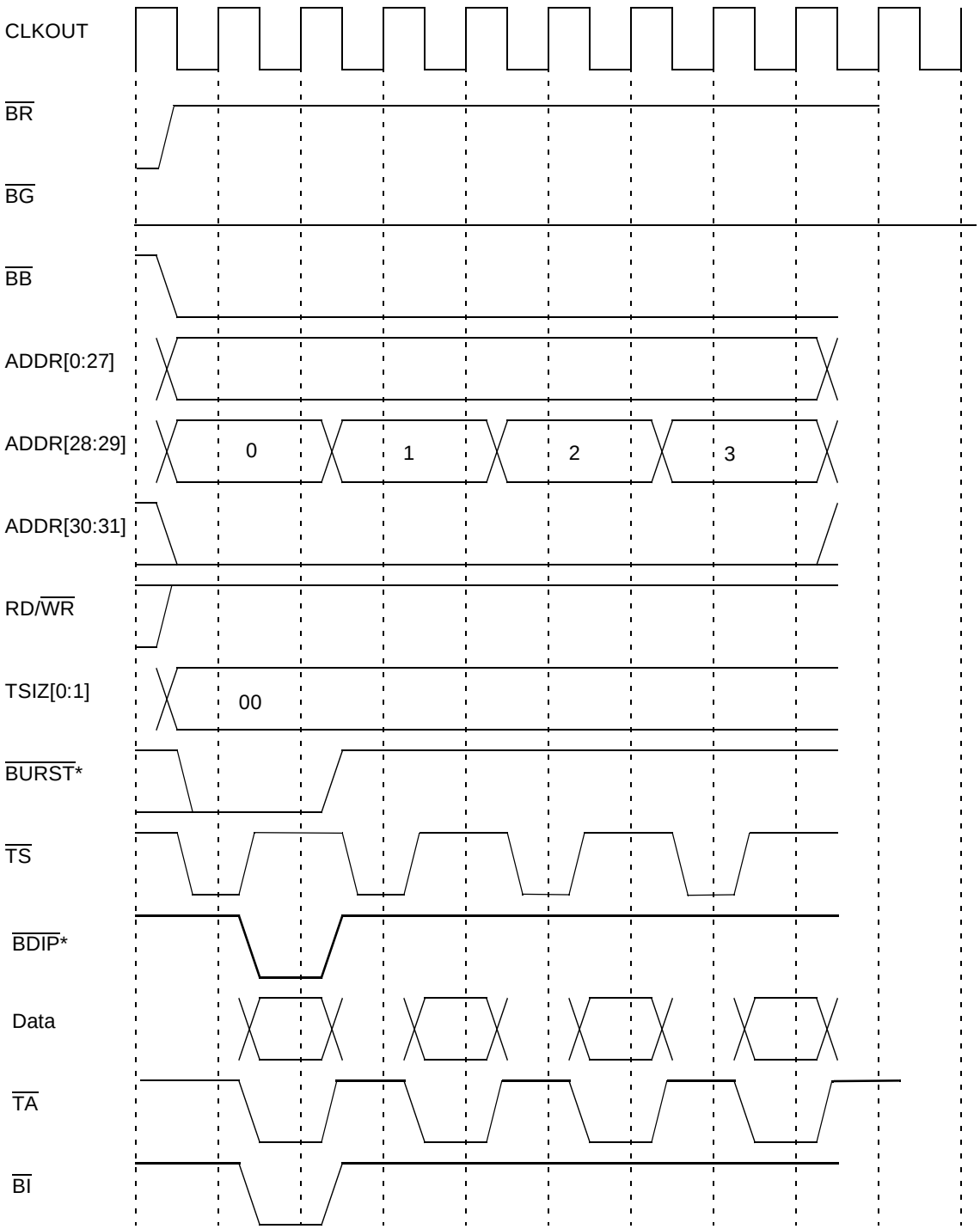


Figure 9-17 Burst-Write Cycle, 32-Bit Port Size, Zero Wait States (Only for External Master Memory Controller Service Support)



* $\overline{\text{BURST}}$ and $\overline{\text{BDIP}}$ will be asserted for one cycle if the RCPU core requests a burst, but the USIU splits it into a sequence of normal cycles.

Figure 9-18 Burst-Inhibit Cycle, 32-Bit Port Size (Emulated Burst)

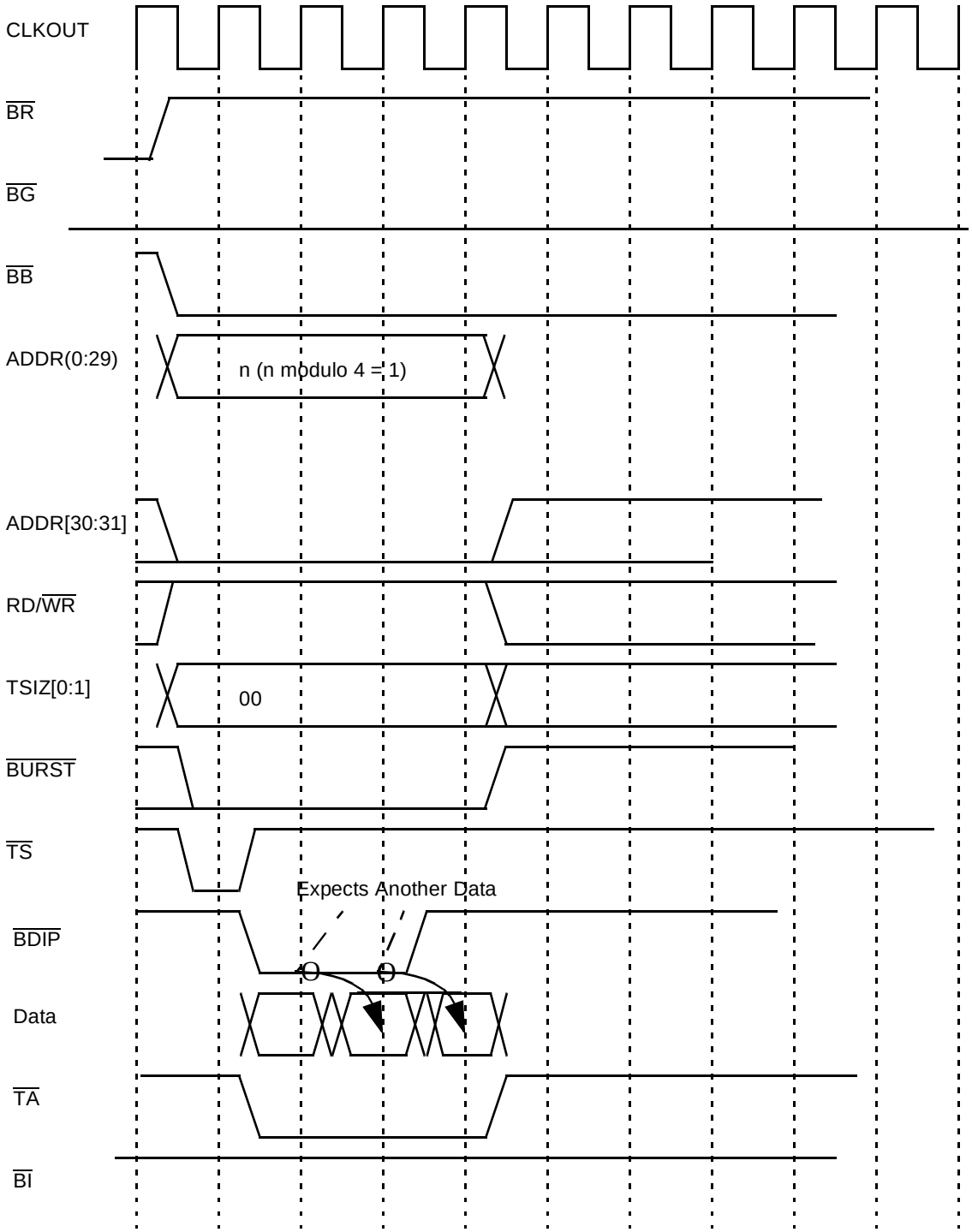


Figure 9-19 Non-Wrap Burst with Three Beats

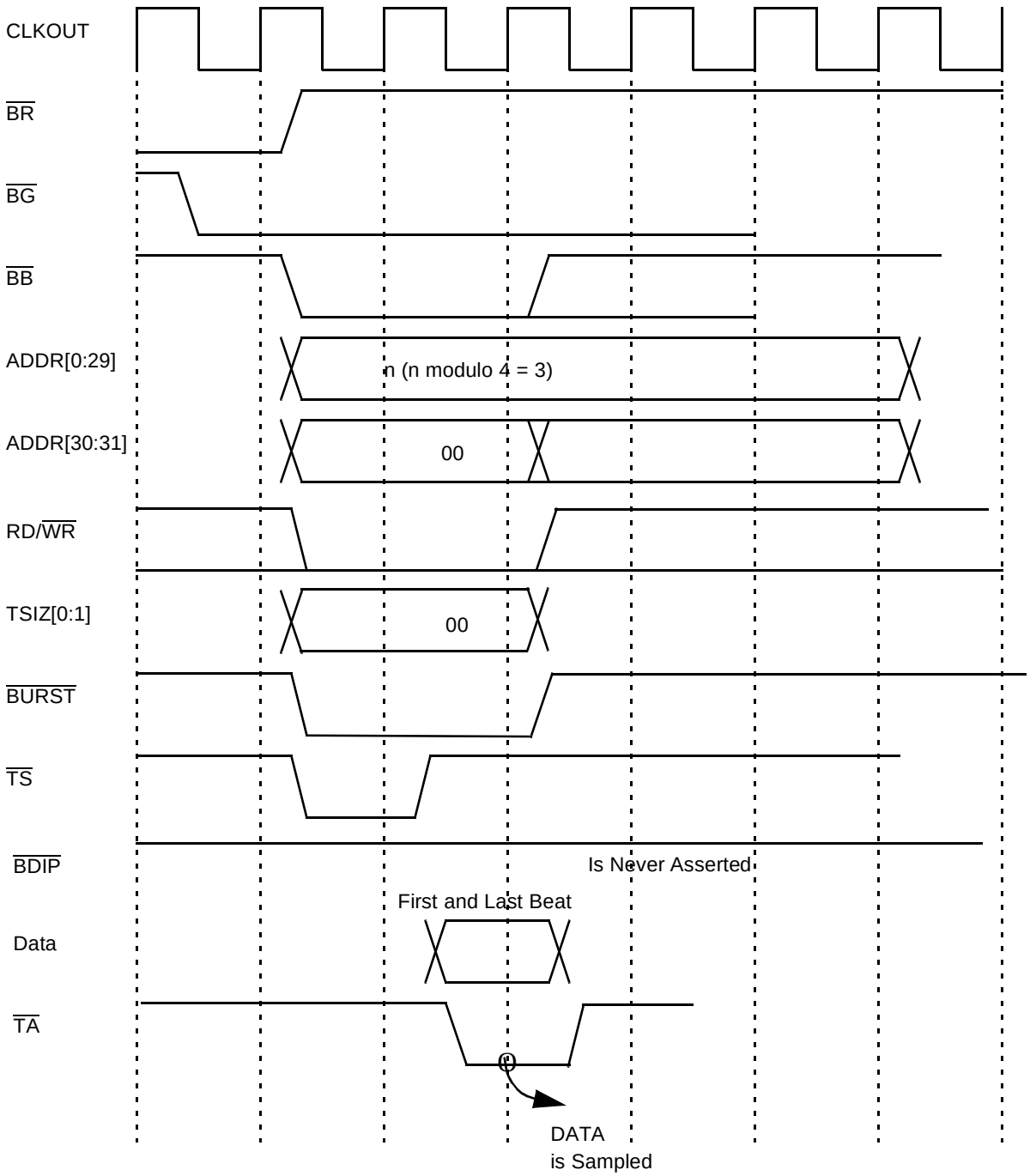


Figure 9-20 Non-Wrap Burst with One Data Beat

9.5.5 Alignment and Packaging of Transfers

The MPC565 / MPC566 external bus requires natural address alignment:

- Byte accesses allow any address alignment
- Half-word accesses require address bit 31 to equal zero
- Word accesses require address bits 30 – 31 to equal zero
- Burst accesses require address bits 30 – 31 to equal zero

The MPC565 / MPC566 performs operand transfers through its 32-bit data port. If the transfer is controlled by the internal memory controller, the MPC565 / MPC566 can support 8- and 16-bit data port sizes.

The bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on DATA[0:31], a 16-bit port must reside on DATA[0:15], and an 8-bit port must reside on DATA[0:7]. The MPC565 / MPC566 always tries to transfer the maximum amount of data on all bus cycles. For a word operation, it always assumes that the port is 32 bits wide when beginning the bus cycle.

In **Figure 9-21**, **Figure 9-22**, **Table 9-2**, and **Table 9-3**, the following conventions are used:

- OP0 is the most-significant byte of a word operand and OP3 is the least-significant byte.
- The two bytes of a half-word operand are either OP0 (most-significant) and OP1 or OP2 (most-significant) and OP3, depending on the address of the access.
- The single byte of a byte-length operand is OP0, OP1, OP2, or OP3, depending on the address of the access.

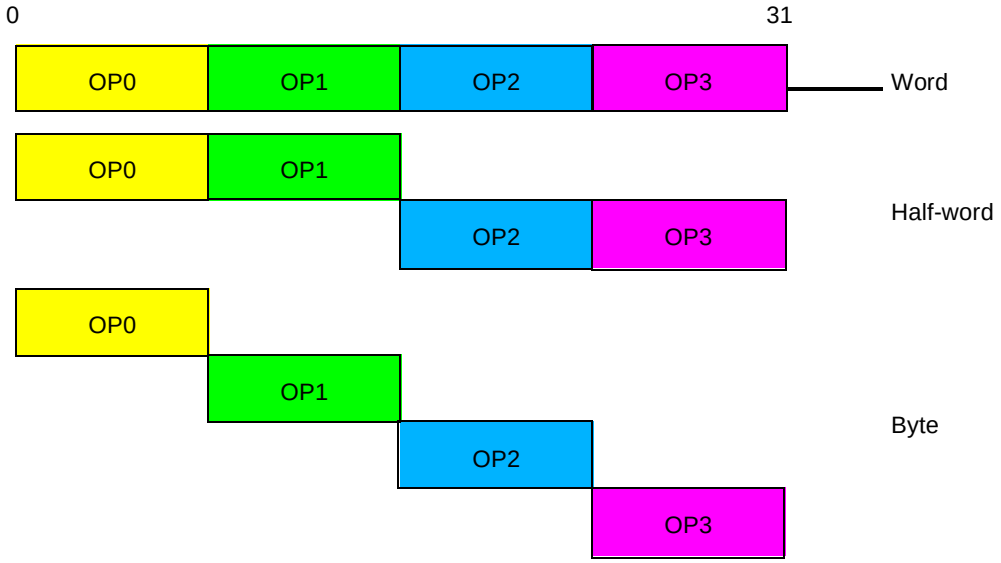


Figure 9-21 Internal Operand Representation

Figure 9-22 illustrates the device connections on the data bus.

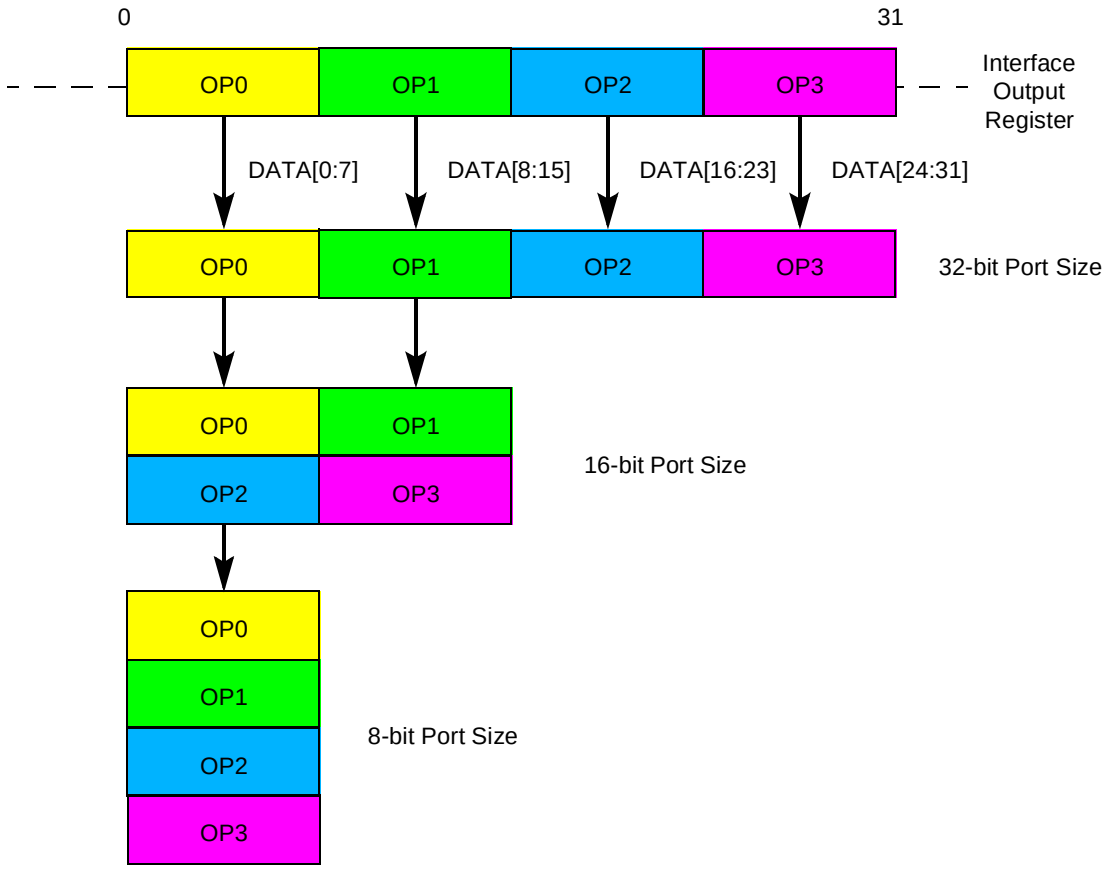


Figure 9-22 Interface To Different Port Size Devices

Table 9-2 lists the bytes required on the data bus for read cycles.

Table 9-2 Data Bus Requirements For Read Cycles

Transfer Size	TSIZE [0:1]	Address	32-bit Port Size				16-bit Port Size		8-bit Port Size
		ADDR [30:31]	DATA [0:7]	DATA [8:15]	DATA [16:23]	DATA [24:31]	DATA [0:7]	DATA [8:15]	DATA [0:7]
Byte	01	00	OP0	—	—	—	OP0	—	OP0
	01	01	—	OP1	—	—	—	OP1	OP1
	01	10	—	—	OP2	—	OP2	—	OP2
	01	11	—	—	—	OP3	—	OP3	OP3
Half-word	10	00	OP0	OP1	—	—	OP0	OP1	OP0
	10	10	—	—	OP2	OP3	OP2	OP3	OP2
Word	00	00	OP0	OP1	OP2	OP3	OP0	OP1	OP0

NOTE: "—" denotes a byte not required during that read cycle.

Table 9-3 lists the patterns of the data transfer for write cycles when the MPC565 / MPC566 initiates an access.

Table 9-3 Data Bus Contents for Write Cycles

Transfer Size	TSIZE[0:1]	Address	External Data Bus Pattern			
		ADDR [30:31]	DATA [0:7]	DATA [8:15]	DATA [16:23]	DATA [24:31]
Byte	01	00	OP0	—	—	—
	01	01	OP1	OP1	—	—
	01	10	OP2	—	OP2	—
	01	11	OP3	OP3	—	OP3
Half-word	10	00	OP0	OP1	—	—
	10	10	OP2	OP3	OP2	OP3
Word	00	00	OP0	OP1	OP2	OP3

NOTE: "—" denotes a byte not driven during that write cycle.

9.5.6 Arbitration Phase

The external bus design provides for a single bus master at any one time, either the MPC565 / MPC566 or an external device. One or more of the external devices on the bus can have the capability of becoming bus master for the external bus. Bus arbitration may be handled either by an external central bus arbiter or by the internal on-chip arbiter. In the latter case, the system is optimized for one external bus master besides the MPC565 / MPC566. The arbitration configuration (external or internal) is set at system reset.

Each bus master must have bus request ($\overline{BR}$), bus grant ($\overline{BG}$), and bus busy ($\overline{BB}$) signals. The device that needs the bus asserts $\overline{BR}$. The device then waits for the arbiter to assert $\overline{BG}$. In addition, the new master must look at $\overline{BB}$ to ensure that no other master is driving the bus before it can assert $\overline{BB}$ to assume ownership of the bus. Any time the arbiter has taken the bus grant away from the master and the master wants to exe-

cute a new cycle, the master must re-arbitrate before a new cycle can be executed. The MPC565 / MPC566, however, guarantees data coherency for access to a small port size and for decomposed bursts. This means that the MPC565 / MPC566 will not release the bus before the completion of the transactions that are considered atomic. **Figure 9-23** describes the basic protocol for bus arbitration.

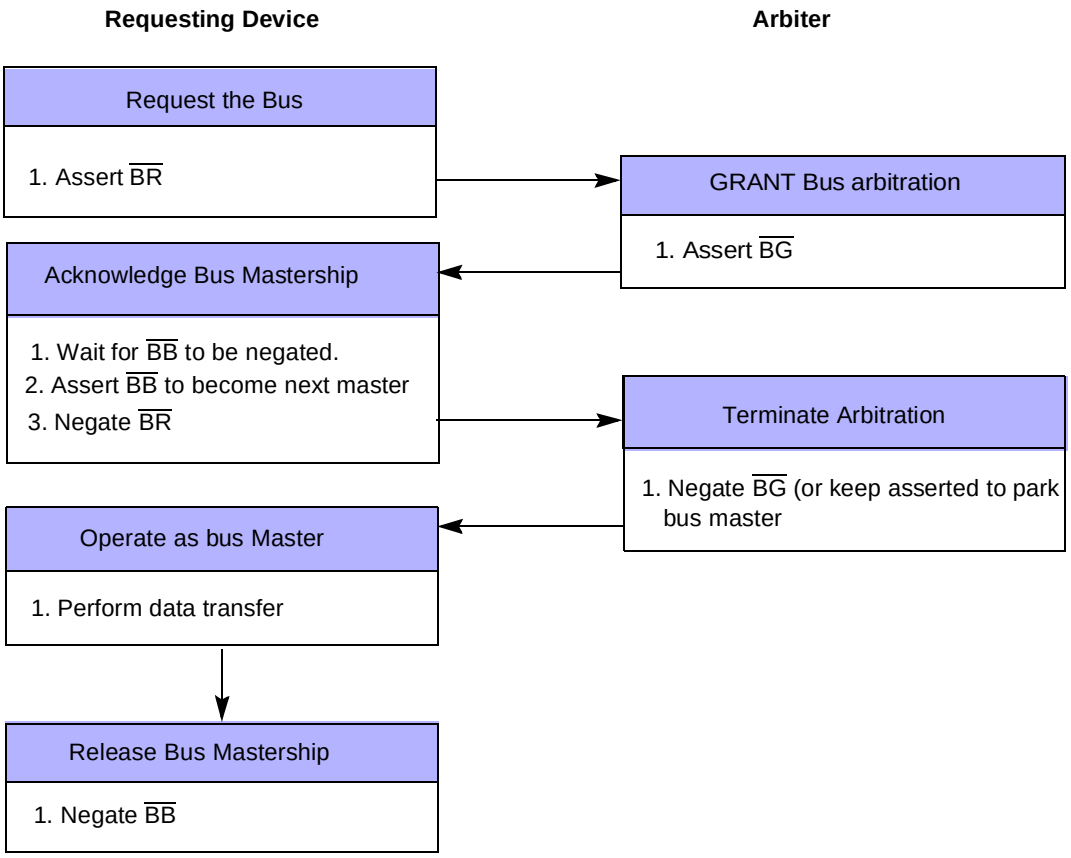


Figure 9-23 Bus Arbitration Flowchart

9.5.6.1 Bus Request

The potential bus master asserts $\overline{BR}$ to request bus mastership. $\overline{BR}$ should be negated as soon as the bus is granted, the bus is not busy, and the new master can drive the bus. If more requests are pending, the master can keep asserting its bus request as long as needed. When configured for external central arbitration, the MPC565 / MPC566 drives this signal when it requires bus mastership. When the internal on-chip arbiter is used, this signal is an input to the internal arbiter and should be driven by the external bus master.

9.5.6.2 Bus Grant

The arbiter asserts $\overline{BG}$ to indicate that the bus is granted to the requesting device. This signal can be negated following the negation of $\overline{BR}$ or kept asserted for the current master to park the bus.

When configured for external central arbitration, $\overline{BG}$ is an input signal to the MPC565 / MPC566 from the external arbiter. When the internal on-chip arbiter is used, this signal is an output from the internal arbiter to the external bus master.

9.5.6.3 Bus Busy

$\overline{BB}$ assertion indicates that the current bus master is using the bus. New masters should not begin transfer until this signal is negated. The bus owner should not relinquish or negate this signal until the transfer is complete. To avoid contention on the $\overline{BB}$ line, the master should three-state this signal when it gets a logical one value. This requires the connection of an external pull-up resistor to ensure that a master that acquires the bus is able to recognize the $\overline{BB}$ line negated, regardless of how many cycles have passed since the previous master relinquished the bus. Refer to **Figure 9-24**.

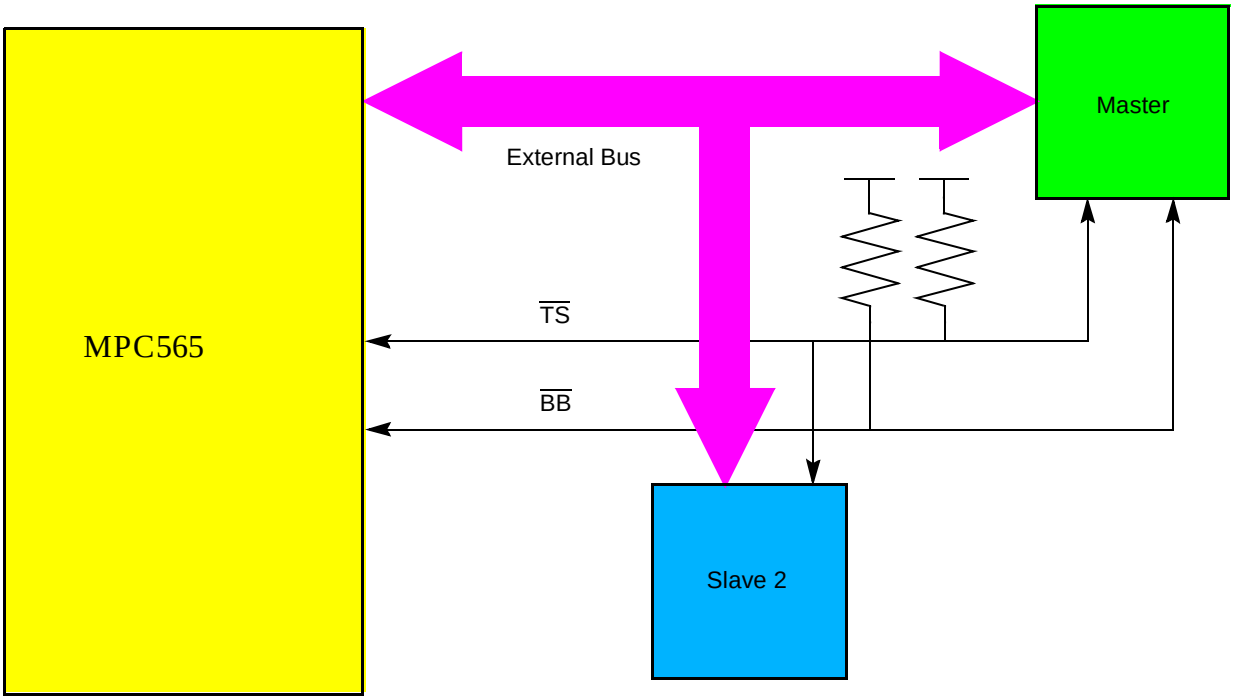


Figure 9-24 Masters Signals Basic Connection

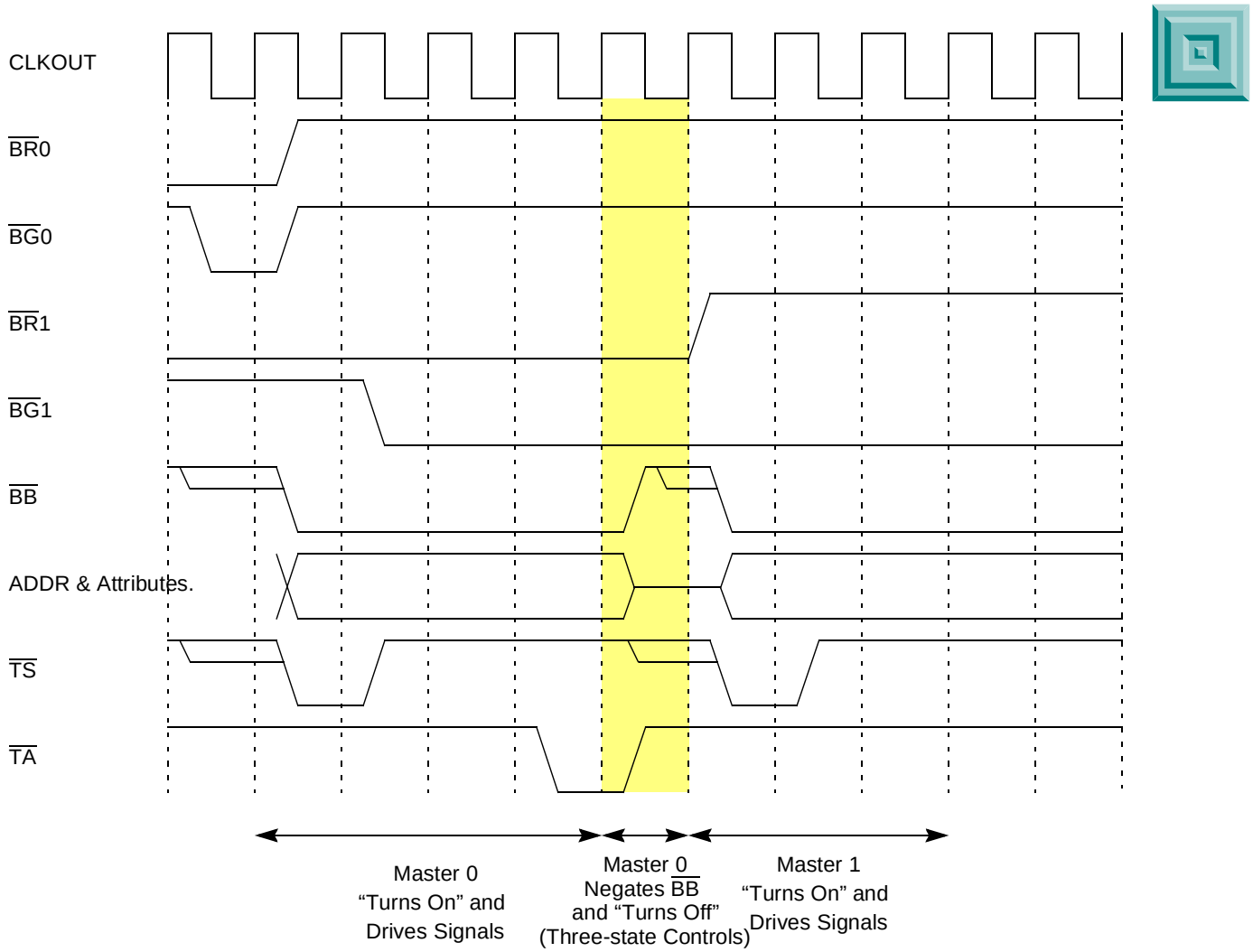


Figure 9-25 Bus Arbitration Timing Diagram

9.5.6.4 Internal Bus Arbiter

The MPC565 / MPC566 can be configured at system reset to use the internal bus arbiter. In this case, the MPC565 / MPC566 will be parked on the bus. The parking feature allows the MPC565 / MPC566 to skip the bus request phase, and if $\overline{BB}$ is negated, assert $\overline{BB}$ and initiate the transaction without waiting for $\overline{BG}$ from the arbiter.

The priority of the external device relative to the internal MPC565 / MPC566 bus masters is programmed in the SIU module configuration register. If the external device requests the bus and the MPC565 / MPC566 does not require it, or if the external device has higher priority than the current internal bus master, the MPC565 / MPC566 grants the bus to the external device.

Table 9-4 describes the priority mechanism used by the internal arbiter.

Table 9-4 Priority Between Internal and External Masters over External Bus[1]

Type	Direction	Priority
Parked access[2]	Internal → external	0
Instruction access	Internal → external	3
Data access	Internal → external	4
External access	external → external/internal	EARP (could be programmed to 0 – 7)

NOTES:
1. External master will be granted external bus ownership if EARP is greater than the internal access priority.
2. Parked access is instruction or data access from the RCPU which is initiated on the internal bus without requesting it first in order to improve performance.

Figure 9-26 illustrates the internal finite-state machine that implements the arbiter protocol.

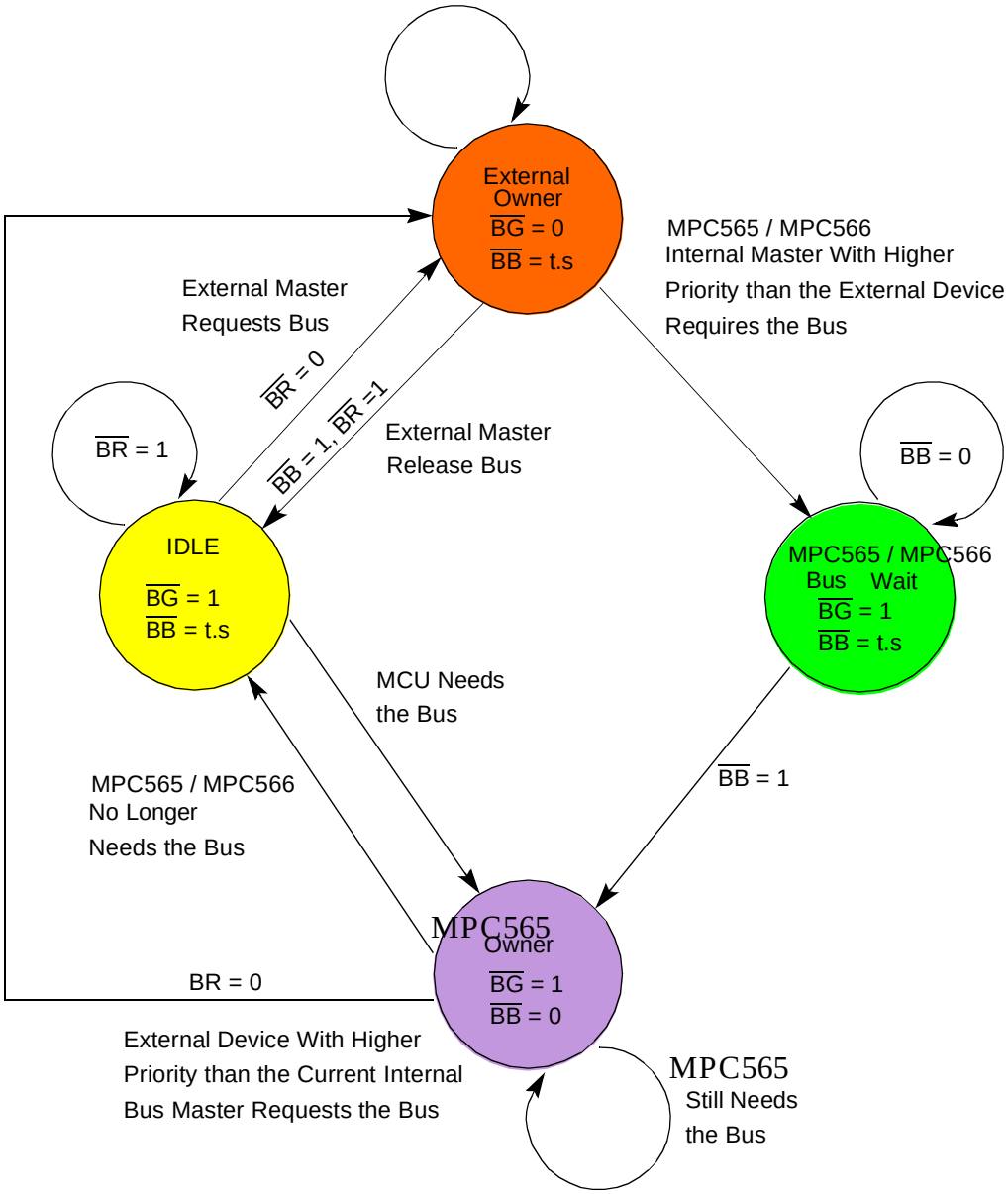


Figure 9-26 Internal Bus Arbitration State Machine

9.5.7 Address Transfer Phase Signals

Address transfer phase signals include the following:

- Transfer start
- Address bus
- Transfer attributes

Transfer attributes signals include RD/$\overline{\text{WR}}$, $\overline{\text{BURST}}$, TSIZ[0:1], AT[0:3], $\overline{\text{STS}}$, and $\overline{\text{BDIP}}$. With the exception of the $\overline{\text{BDIP}}$, these signals are available at the same time as the address bus.

9.5.7.1 Transfer Start

This signal ($\overline{TS}$) indicates the beginning of a transaction on the bus addressing a slave device. This signal should be asserted by a master only after the ownership of the bus was granted by the arbitration protocol. This signal is asserted for the first cycle of the transaction only and is negated in successive clock cycles until the end of the transaction. The master should three-state this signal when it relinquishes the bus to avoid contention between two or more masters in this line. This situation indicates that an external pull-up resistor should be connected to the $\overline{TS}$ signal to avoid having a slave recognize this signal as asserted when no master drives it. Refer to **Figure 9-24**.

9.5.7.2 Address Bus

The address bus consists of 32 bits, with ADDR0 the most significant bit and ADDR31 the least significant bit. The bus is byte-addressable, so each address can address one or more bytes. The address and its attributes are driven on the bus with the transfer start signal and kept valid until the bus master receives the transfer acknowledge signal from the slave. To distinguish the individual byte, the slave device must observe the TSIZ signals.

9.5.7.3 Read/Write

A high value on the RD/$\overline{WR}$ line indicates a read access. A low value indicates a write access.

9.5.7.4 Burst Indicator

$\overline{BURST}$ is driven by the bus master at the beginning of the bus cycle along with the address to indicate that the transfer is a burst transfer.

The MPC565 / MPC566 supports a non-wrapping, four-beat maximum, critical word first burst type. The maximum burst size is 16 bytes. For a 32-bit port, the burst includes four beats. For a 16-bit port, the burst includes eight beats. For an 8-bit port, the burst includes 16 beats at most.

NOTE

Eight- and 16-bit ports must be controlled by the memory controller.

The actual size of the burst is determined by the address of the starting word of the burst. Refer to **Table 9-5** and **Table 9-6**.

Table 9-5 Burst Length and Order

Starting Address ADDR[28:29]	Burst Order (Assuming 32-bit Port Size)	Burst Length in Words (Beats)	Burst Length in Bytes	Comments
00	word 0 → word 1 → word 2 → word 3	4	16	
01	word 1 → word 2 → word 3	3	12	
10	word 2 → word 3	2	8	
11	word 3	1	4	$\overline{\text{BDIP}}$ never asserted

9.5.7.5 Transfer Size

The transfer size signals (TSIZ[0:1]) indicate the size of the requested data transfer. During each transfer, the TSIZ signals indicate how many bytes are remaining to be transferred by the transaction. The TSIZ signals can be used with $\overline{\text{BURST}}$ and ADDR[30:31] to determine which byte lanes of the data bus are involved in the transfer. For non-burst transfers, the TSIZ signals specify the number of bytes starting from the byte location addressed by ADDR[30:31]. In burst transfers, the value of TSIZ is always 00.

Table 9-6 $\overline{\text{BURST}}$/TSIZE Encoding

$\overline{\text{BURST}}$	TSIZ[0:1]	Transfer Size
Negated	01	Byte
Negated	10	Half-word
Negated	11	x
Negated	00	Word
Asserted	00	Burst (16 bytes)

9.5.7.6 Address Types

The address type (AT[0:3]), program trace ($\overline{\text{PTR}}$), and reservation transfer (RSV) signals are outputs that indicate one of 16 address types. These types are designated as either a normal or alternate master cycle, user or supervisor, and instruction or data type. The address type signals are valid at the rising edge of the clock in which the special transfer start ($\overline{\text{STS}}$) signal is asserted.

A special use of the $\overline{\text{PTR}}$ and $\overline{\text{RSV}}$ signals is for the reservation protocol described in **9.5.9 Storage Reservation.** Refer to **9.5.13 Show Cycle Transactions** for information on show cycles.

Table 9-7 summarizes the pins used to define the address type.**Table 9-8** lists all the definitions achieved by combining these pins.

Table 9-7 Address Type Pins

Pin	Function
$\overline{STS}$	0 = Special transfer 1 = Normal transfer
$\overline{TS}$	0 = Start of transfer 1 = No transfer
AT[0]	Must equal zero on MPC565 / MPC566
AT[1]	0 = Supervisor mode 1 = User mode
AT[2]	0 = Instruction 1 = Data
AT[3]	Reservation/Program Trace
$\overline{PTR}$	0 = Program trace 1 = No program trace
$\overline{RSV}$	0 = Reservation data 1 = No reservation

Table 9-8 Address Types Definition

$\overline{STS}$	$\overline{TS}$	AT[0]	AT[1]	AT[2]	AT[3]	$\overline{PTR}$	$\overline{RSV}$	Address Space Definitions
1	x	x	x	x	x	1	1	No transfer
0	0[1]	0	0	0	0	0	1	RCPU, normal instruction, program trace, supervisor mode
					1	1	1	RCPU, normal instruction, supervisor mode
				1	0	1	0	RCPU, reservation data, supervisor mode
					1	1	1	RCPU, normal data, supervisor mode
			1	0	0	0	1	RCPU, normal instruction, program trace, user mode
					1	1	1	RCPU, normal instruction, user mode
				1	0	1	0	RCPU, reservation data, user mode
					1	1	1	RCPU, normal data, user mode
		1	?	?	?	1	1	Reserved
	1	0	0	0	0	0	1	RCPU, show cycle address instruction, program trace, supervisor mode
					1	1	1	RCPU, show cycle address instruction, supervisor mode
				1	0	1	0	RCPU, reservation show cycle data, supervisor mode
					1	1	1	RCPU, show cycle data, supervisor mode
			1	0	0	0	1	RCPU, show cycle address instruction, program trace, user mode
					1	1	1	RCPU, show cycle address instruction, user mode
				1	0	1	0	RCPU, reservation show cycle data, user mode
					1	1	1	RCPU, show cycle data, user mode
		1	?	?	?	1	1	Reserved

NOTES:
1. Cases in which both $\overline{TS}$ and $\overline{STS}$ are asserted indicate normal cycles with the show cycle attribute.

9.5.7.7 Burst Data in Progress

This signal is sent from the master to the slave to indicate that there is a data beat following the current data beat. The master uses this signal to give the slave advance warning of the remaining data in the burst. $\overline{\text{BDIP}}$ can also be used to terminate the burst cycle early. Refer to **9.5.3 Burst Transfer** and **9.5.4 Burst Mechanism** for more information.

9.5.8 Termination Signals

The EBI uses three termination signals:

- Transfer acknowledge ($\overline{\text{TA}}$)
- Burst inhibit ($\overline{\text{BI}}$)
- Transfer error acknowledge ($\overline{\text{TEA}}$)

9.5.8.1 Transfer Acknowledge

Transfer acknowledge indicates normal completion of the bus transfer. During a burst cycle, the slave asserts this signal with every data beat returned or accepted.

9.5.8.2 Burst Inhibit

A slave sends the $\overline{\text{BI}}$ signal to the master to indicate that the addressed device does not have burst capability. If this signal is asserted, the master must transfer in multiple cycles and increment the address for the slave to complete the burst transfer. For a system that does not use the burst mode at all, this signal can be tied low permanently.

9.5.8.3 Transfer Error Acknowledge

The $\overline{\text{TEA}}$ signal terminates a bus cycle under one or more bus error conditions. The current bus cycle must be aborted. This signal overrides any other cycle termination signals, such as transfer acknowledge.

9.5.8.4 Termination Signals Protocol

The transfer protocol was defined to avoid electrical contention on lines that can be driven by various sources. To this end, a slave must not drive signals associated with the data transfer until the address phase is completed and it recognizes the address as its own. The slave must disconnect from signals immediately after it has acknowledged the cycle and no later than the termination of the next address phase cycle. This means that the termination signals must be connected to power through a pull-up resistor to avoid the situation in which a master samples an undefined value in any of these signals when no real slave is addressed.

Refer to **Figure 9-27** and **Figure 9-28**.

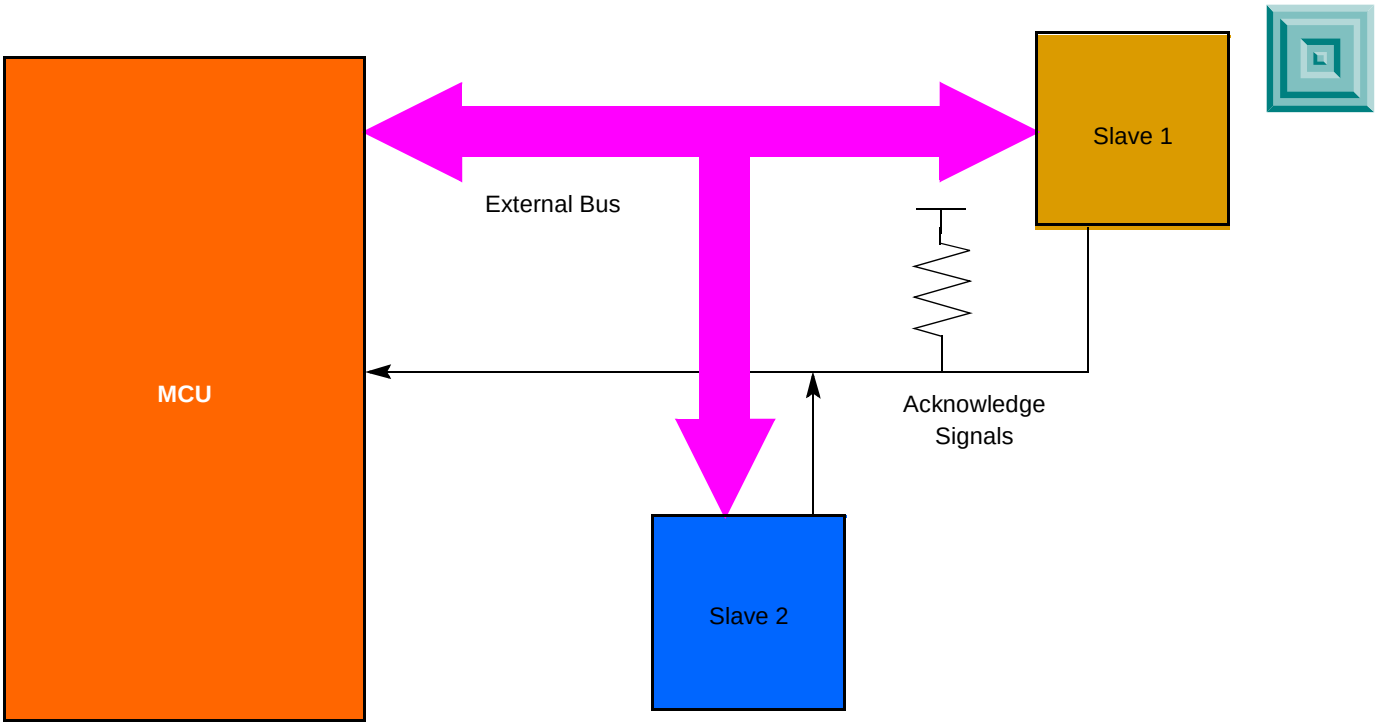


Figure 9-27 Termination Signals Protocol Basic Connection

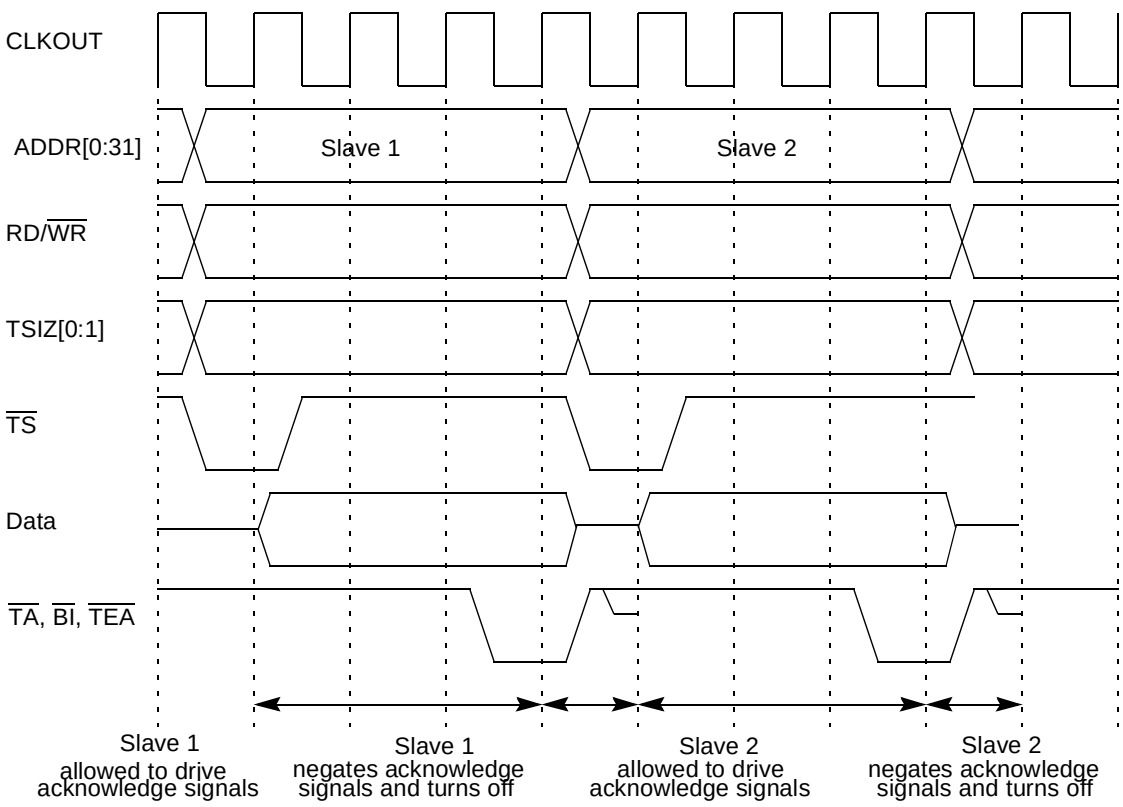


Figure 9-28 Termination Signals Protocol Timing Diagram

9.5.9 Storage Reservation

The MPC565 / MPC566 storage reservation protocol supports a multi-level bus structure. For each local bus, storage reservation is handled by the local reservation logic.

The protocol tries to optimize reservation cancellation such that a PowerPC processor is notified of storage reservation loss on a remote bus only when it has issued a **stwcx** cycle to that address. That is, the reservation loss indication comes as part of the **stwcx** cycle. This method avoids the need to have very fast storage reservation loss indication signals routed from every remote bus to every PowerPC master.

The storage reservation protocol makes the following assumptions:

- Each processor has, at most, one reservation flag
- **lwarx** sets the reservation flag
- **lwarx** by the same processor clears the reservation flag related to a previous **lwarx** instruction and again sets the reservation flag
- **stwcx** by the same processor clears the reservation flag
- Store by the same processor does *not* clear the reservation flag
- Some other processor (or other mechanism) store to the same address as an ex-

isting reservation clears the reservation flag

- In case the storage reservation is lost, it is guaranteed that **stwcx** will not modify the storage

The reservation protocol for a single-level (local) bus is illustrated in **Figure 9-29**. The protocol assumes that an external logic on the bus carries out the following functions:

- Snoops accesses to all local bus slaves
- Holds one reservation for each local master capable of storage reservations
- Sets the reservation when that master issues a load and reserve request
- Clears the reservation when some other master issues a store to the reservation address

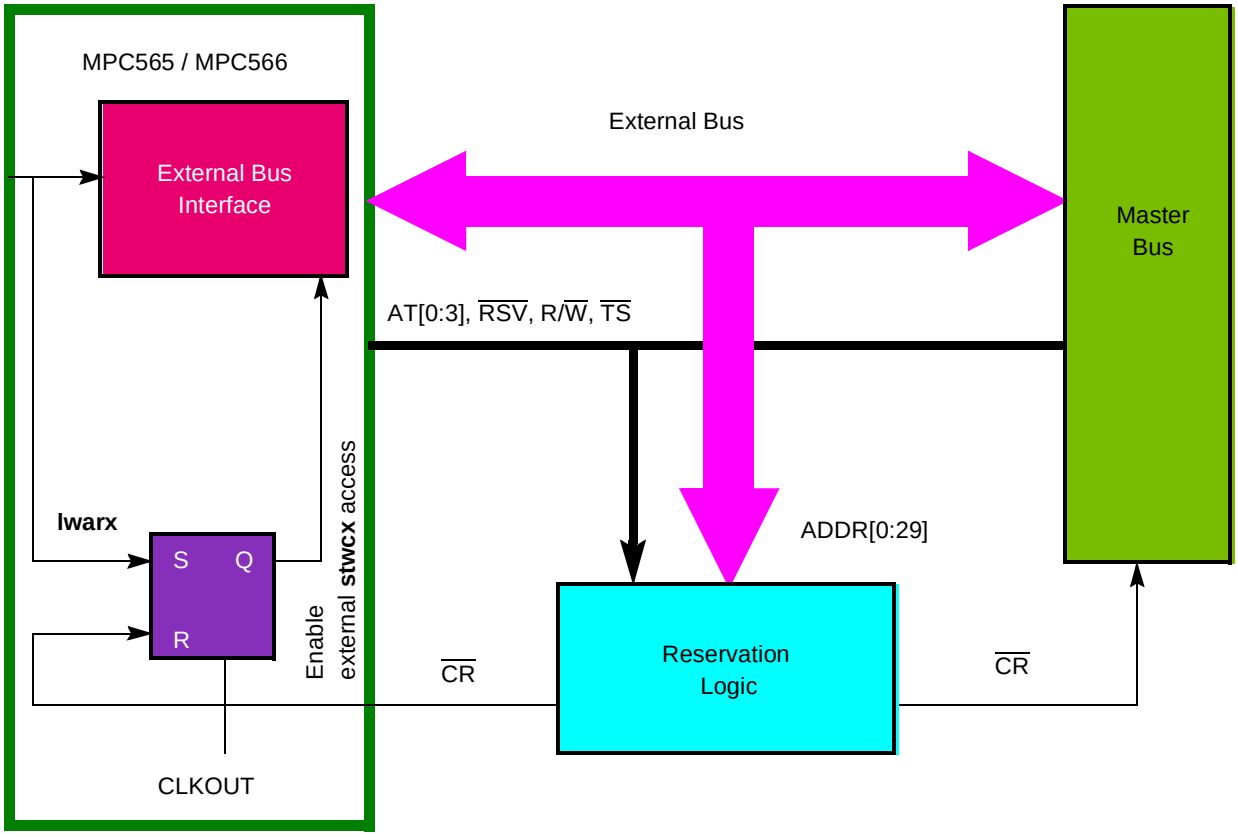


Figure 9-29 Reservation On Local Bus

The MPC565 / MPC566 samples the $\overline{CR}$ line at the rising edge of CLKOUT. When this signal is asserted, the reservation flag is reset.

The EBI samples the logical value of the reservation flag prior to externally starting a bus cycle initiated by the RCPU **stwcx** instruction. If the reservation flag is set, the EBI begins with the bus cycle. If the reservation flag is reset, no bus cycle is initiated externally, and this situation is reported to the RCPU.

The reservation protocol for a multi-level (local) bus is illustrated in **Figure 9-30**. The system describes the situation in which the reserved location is sited in the remote bus.

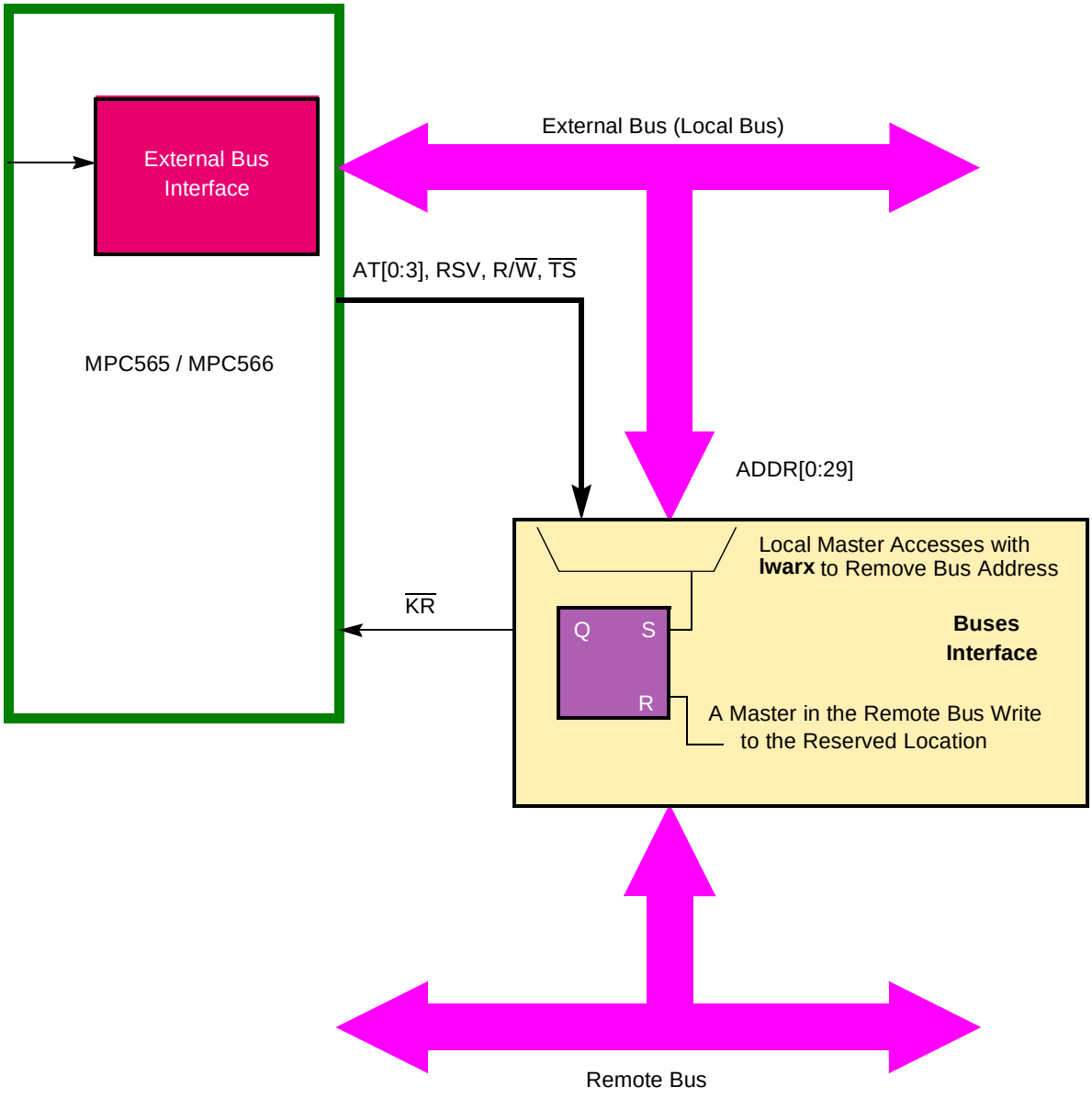


Figure 9-30 Reservation On Multilevel Bus Hierarchy

In this case, the bus interface block implements a reservation flag for the local bus master. The reservation flag is set by the bus interface when a load with reservation is issued by the local bus master and the reservation address is located on the remote bus. The flag is reset when an alternative master on the remote bus accesses the same location in a write cycle. If the MPC565 / MPC566 begins a memory cycle to the previously reserved address (located in the remote bus) as a result of an **stwcx** instruction, the following two cases can occur:

- If the reservation flag is set, the buses interface acknowledges the cycle in a normal way
- If the reservation flag is reset, the bus interface should assert the $\overline{KR}$. However, the bus interface should not perform the remote bus write-access or abort it if the remote bus supports aborted cycles. In this case the failure of the **stwcx** instruction is reported to the RCPU.

9.5.10 Bus Exception Control Cycles

The MPC565 / MPC566 bus architecture requires assertion of $\overline{TA}$ from an external device to signal that the bus cycle is complete. $\overline{TA}$ is not asserted in the following cases:

- The external device does not respond
- Various other application-dependent errors occur

External circuitry can provide $\overline{TEA}$ when no device responds by asserting $\overline{TA}$ within an appropriate period of time after the MPC565 / MPC566 initiates the bus cycle (it can be the internal bus monitor). This allows the cycle to terminate and the processor to enter exception-processing for the error condition (each one of the internal masters causes an internal interrupt under this situation). To properly control termination of a bus cycle for a bus error, $\overline{TEA}$ must be asserted at the same time or before $\overline{TA}$ is asserted. $\overline{TEA}$ should be negated before the second rising edge after it was sampled as asserted to avoid the detection of an error for the next initiated bus cycle. $\overline{TEA}$ is an open drain pin that allows the “wired-or” of any different sources of error generation.

9.5.10.1 Retrying a Bus Cycle

When an external device asserts the $\overline{RETRY}$ signal during a bus cycle, the MPC565 / MPC566 enters a sequence in which it terminates the current transaction, relinquishes the ownership of the bus, and retries the cycle using the same address, address attributes, and data (in the case of a write cycle).

Figure 9-31 illustrates the behavior of the MPC565 / MPC566 when the $\overline{RETRY}$ signal is detected as a termination of a transfer. As seen in this figure, in the case when the internal arbiter is enabled, the MPC565 / MPC566 negates $\overline{BB}$ and asserts $\overline{BG}$ in the clock cycle following the retry detection. This allows any external master to gain bus ownership. In the next clock cycle, a normal arbitration procedure occurs again. As shown in the figure, the external master did not use the bus, so the MPC565 / MPC566 initiates a new transfer with the same address and attributes as before.

In **Figure 9-32**, the same situation is shown except that the MPC565 / MPC566 is working with an external arbiter. In this case, in the clock cycle after the $\overline{RETRY}$ signal is detected asserted, $\overline{BR}$ is negated together with $\overline{BB}$. One clock cycle later, the normal arbitration procedure occurs again.

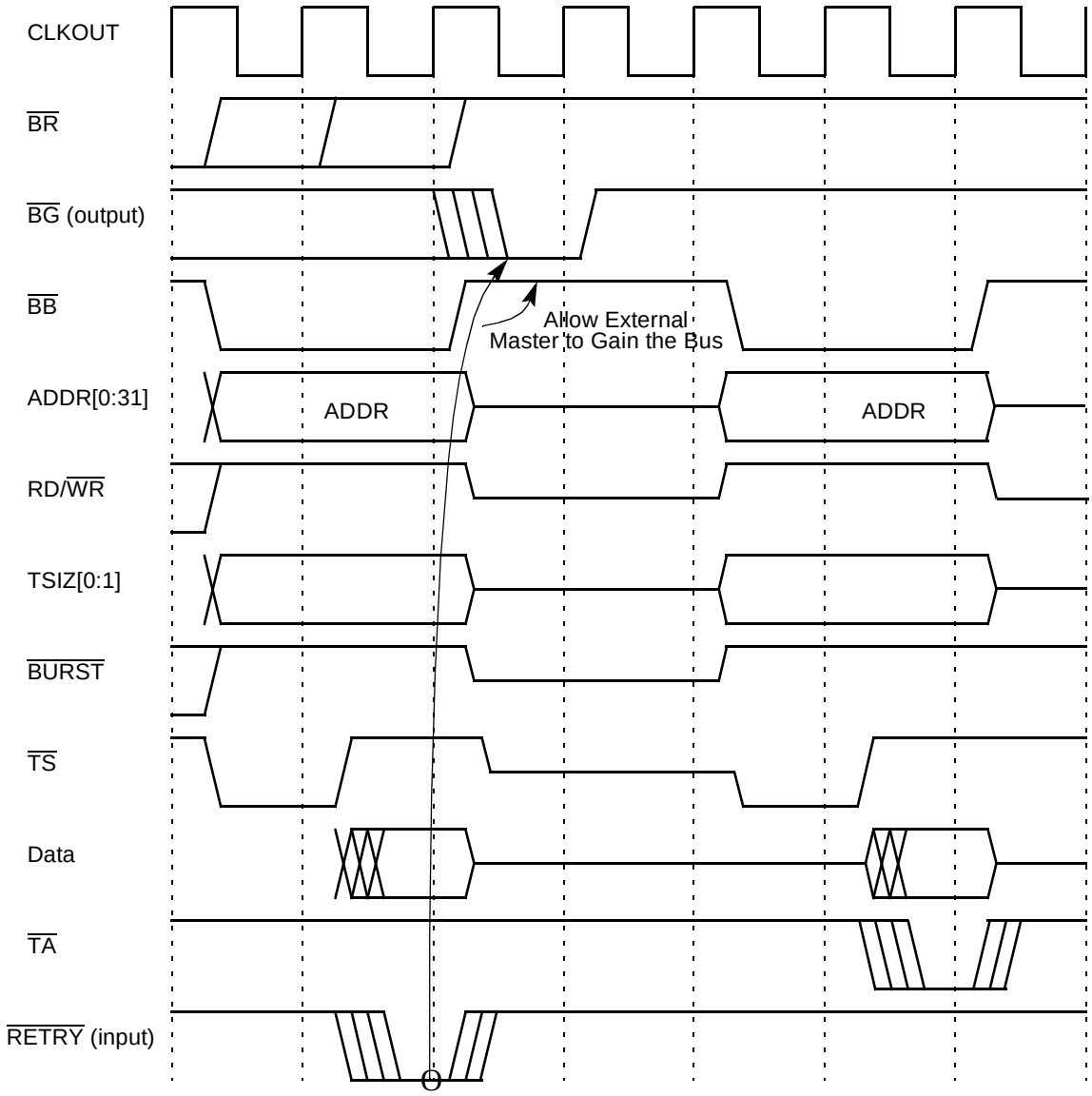


Figure 9-31 Retry Transfer Timing— Internal Arbiter

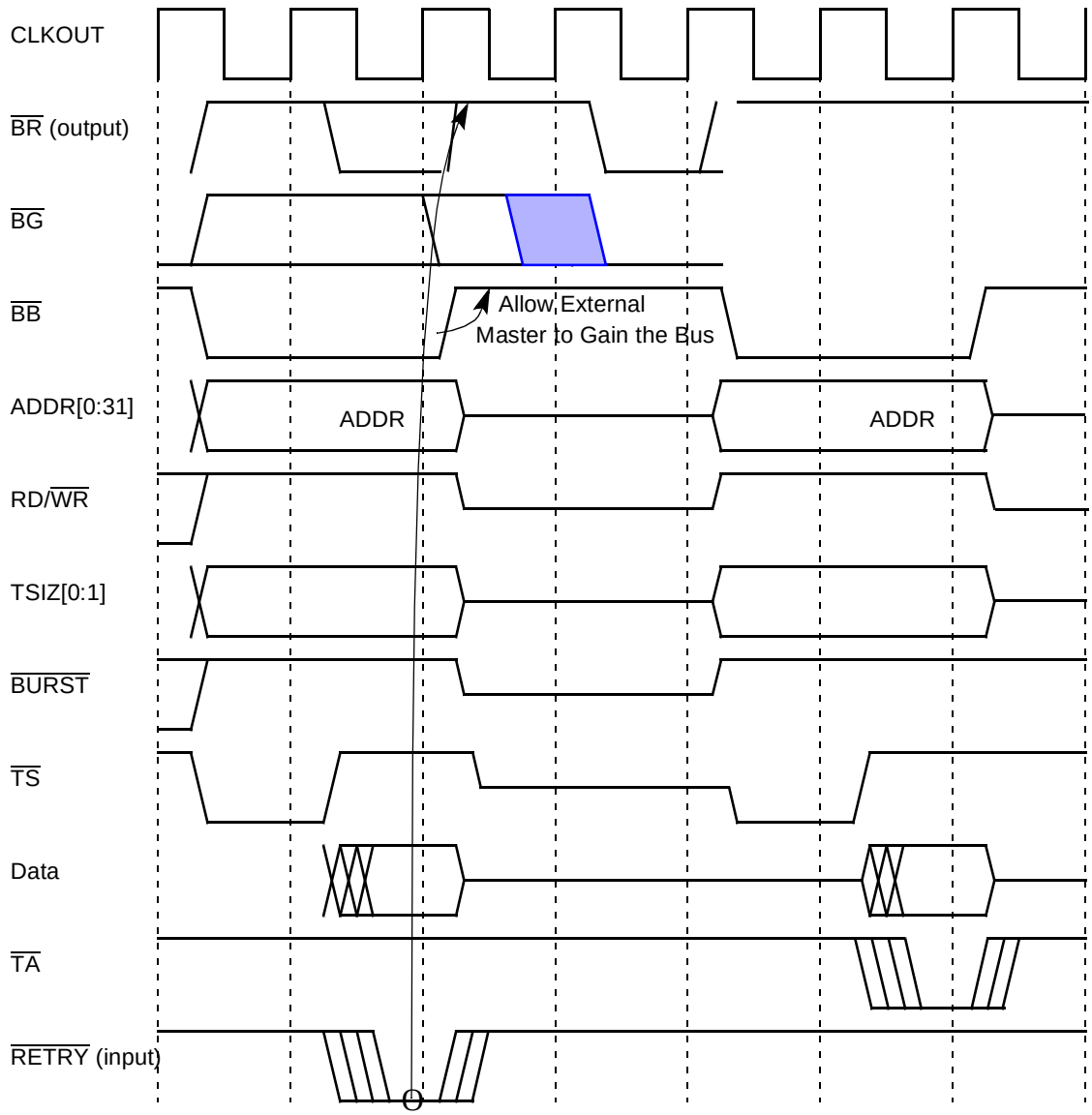


Figure 9-32 Retry Transfer Timing — External Arbiter

When the MPC565 / MPC566 initiates a burst access, the bus interface recognizes the RETRY assertion as a retry termination only if it detects it before the first data beat was acknowledged by the slave device. When the RETRY signal is asserted as a termination signal on any data beat of the access after the first (being the first data beat acknowledged by a normal TA assertion), the MPC565 / MPC566 recognizes RETRY as a transfer error acknowledge.

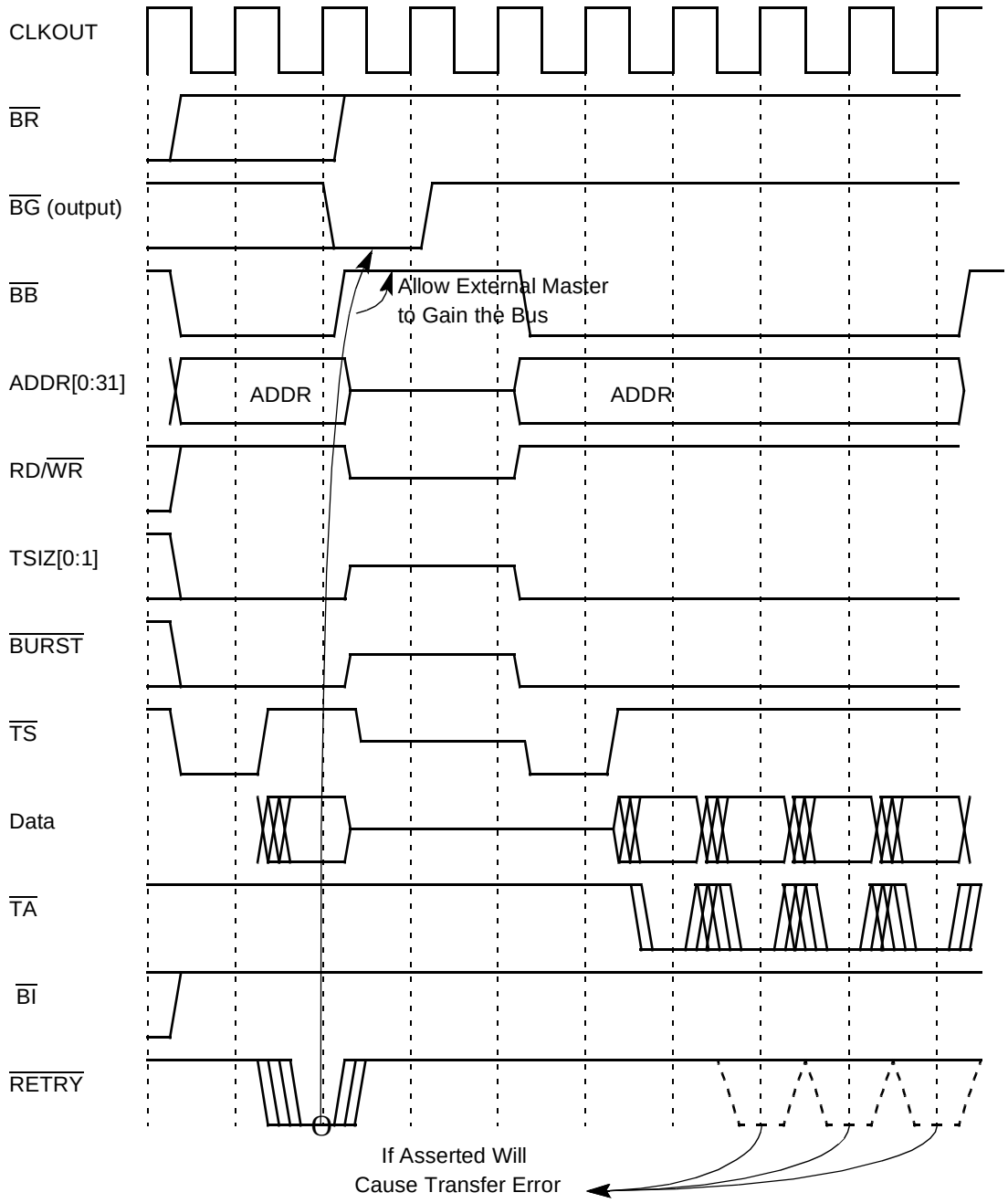


Figure 9-33 Retry On Burst Cycle

If a burst access is acknowledged on its first beat with a normal $\overline{TA}$ but with the $\overline{BI}$ signal asserted, the following single-beat transfers initiated by the MPC565 / MPC566 to complete the 16-byte transfer recognizes the $\overline{RETRY}$ signal assertion as a transfer error acknowledge.

In the case in which a small port size causes the MPC565 / MPC566 to break a bus transaction into several small transactions, terminating any transaction with $\overline{RETRY}$

causes a transfer error acknowledge. See **9.5.2.3 Single Beat Flow with Small Port Size**.

9.5.10.2 Termination Signals Protocol Summary

Table 9-9 summarizes how the MPC565 / MPC566 recognizes the termination signals provided by the slave device that is addressed by the initiated transfer.

Table 9-9 Termination Signals Protocol

$\overline{TEA}$	$\overline{TA}$	$\overline{RETRY}$	Action
Asserted	X	X	Transfer error termination
Negated	Asserted	X	Normal transfer termination
Negated	Negated	Asserted	Retry transfer termination

9.5.11 Bus Operation in External Master Modes

When an external master takes ownership of the external bus and the MPC565 / MPC566 is programmed for external master mode operation, the external master can access the internal space of the MPC565 / MPC566 (see **6.2 External Master Modes**). In an external master mode, the external master owns the bus, and the direction of most of the bus signals is inverted, relative to its direction when the MPC565 / MPC566 owns the bus.

The external master gets ownership of the bus and asserts $\overline{TS}$ in order to initiate an external master access. The access is directed to the internal bus only if the input address matches the internal address space. The access is terminated with one of the followings outputs: $\overline{TA}$, $\overline{TEA}$, or $\overline{RETRY}$. If the access completes successfully, the MPC565 / MPC566 asserts $\overline{TA}$, and the external master can proceed with another external master access or relinquish the bus. If an address or data error is detected internally, the MPC565 / MPC566 asserts $\overline{TEA}$ for one clock. $\overline{TEA}$ should be negated before the second rising edge after it is sampled asserted in order to avoid the detection of an error for the next bus cycle initiated. $\overline{TEA}$ is an open drain pin, and the negation timing depends on the attached pull-up. The MPC565 / MPC566 asserts the $\overline{RETRY}$ signal for one clock in order to retry the external master access.

If the address of the external access does not match the internal memory space, the internal memory controller can provide the chip-select and control signals for accesses that belong to one of the memory controller regions. This feature is explained in **SECTION 10 MEMORY CONTROLLER**.

Figure 9-34 and **Figure 9-35** illustrate the basic flow of read and write external master accesses.

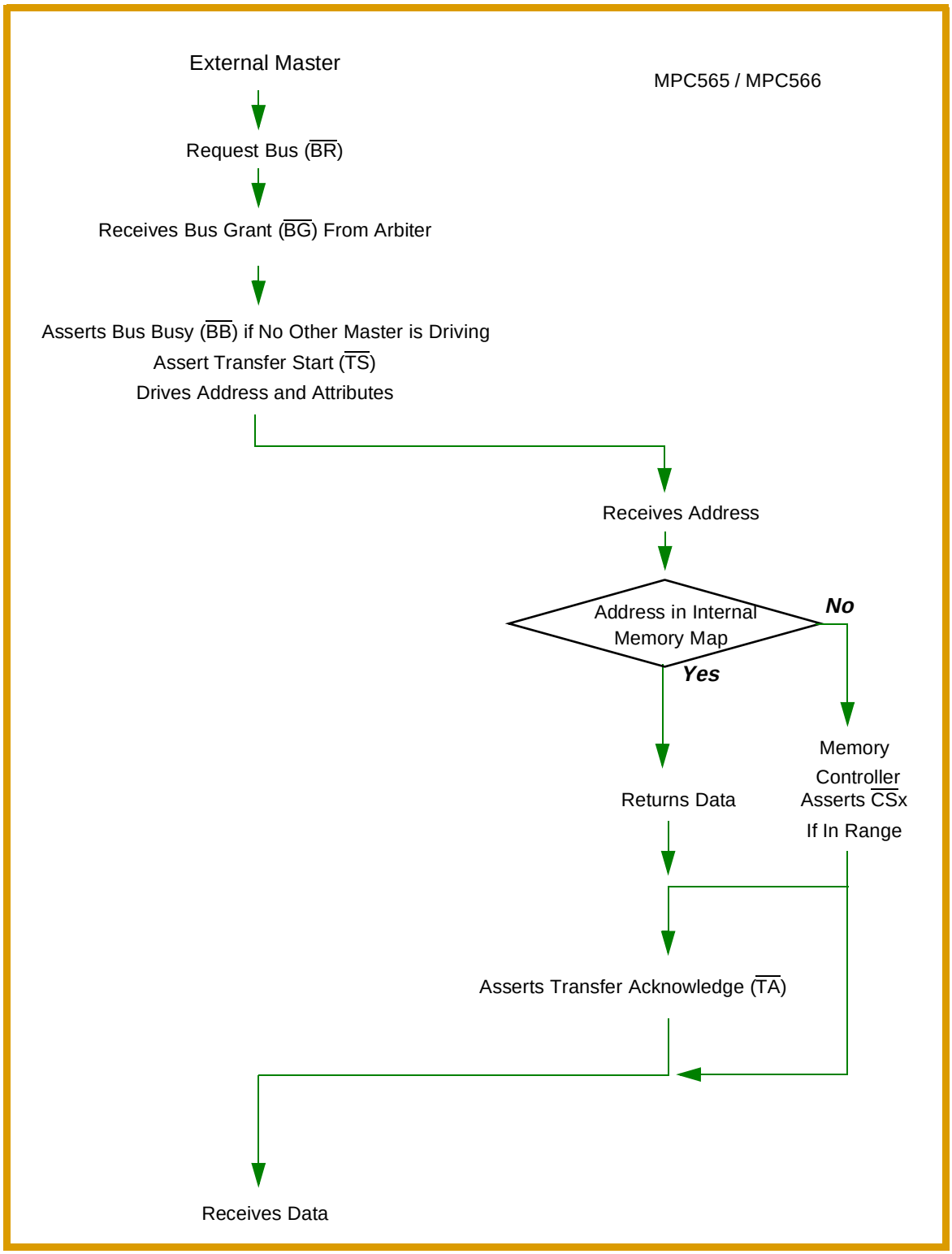


Figure 9-34 Basic Flow of an External Master Read Access

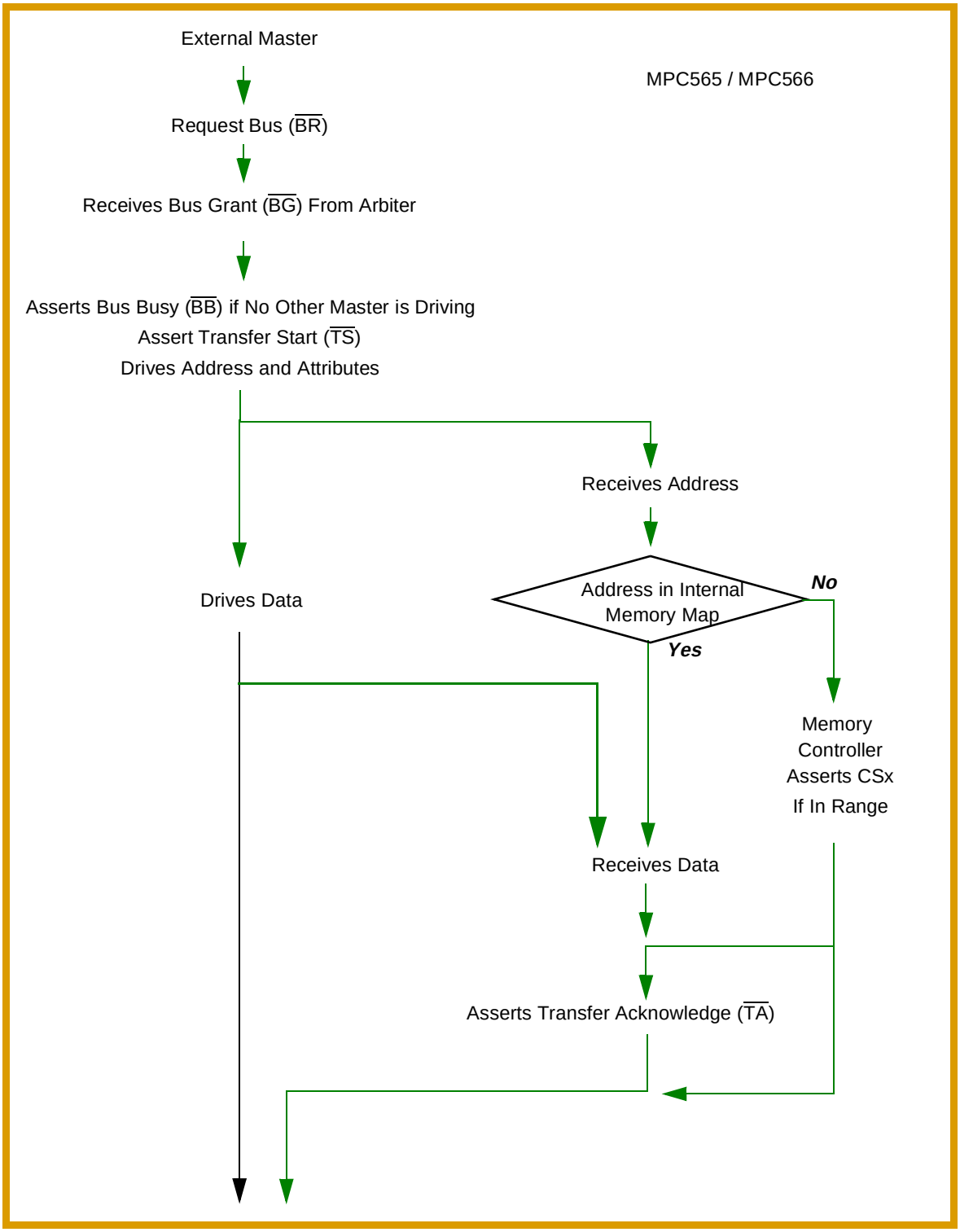


Figure 9-35 Basic Flow of an External Master Write Access

Figure 9-36, **Figure 9-37** and **Figure 9-38** describe read and write cycles from an external master accessing internal space in the MPC565 / MPC566.

NOTE

The minimum number of wait states for such access is two clocks. The accesses in these figures are valid for both peripheral mode and slave mode.

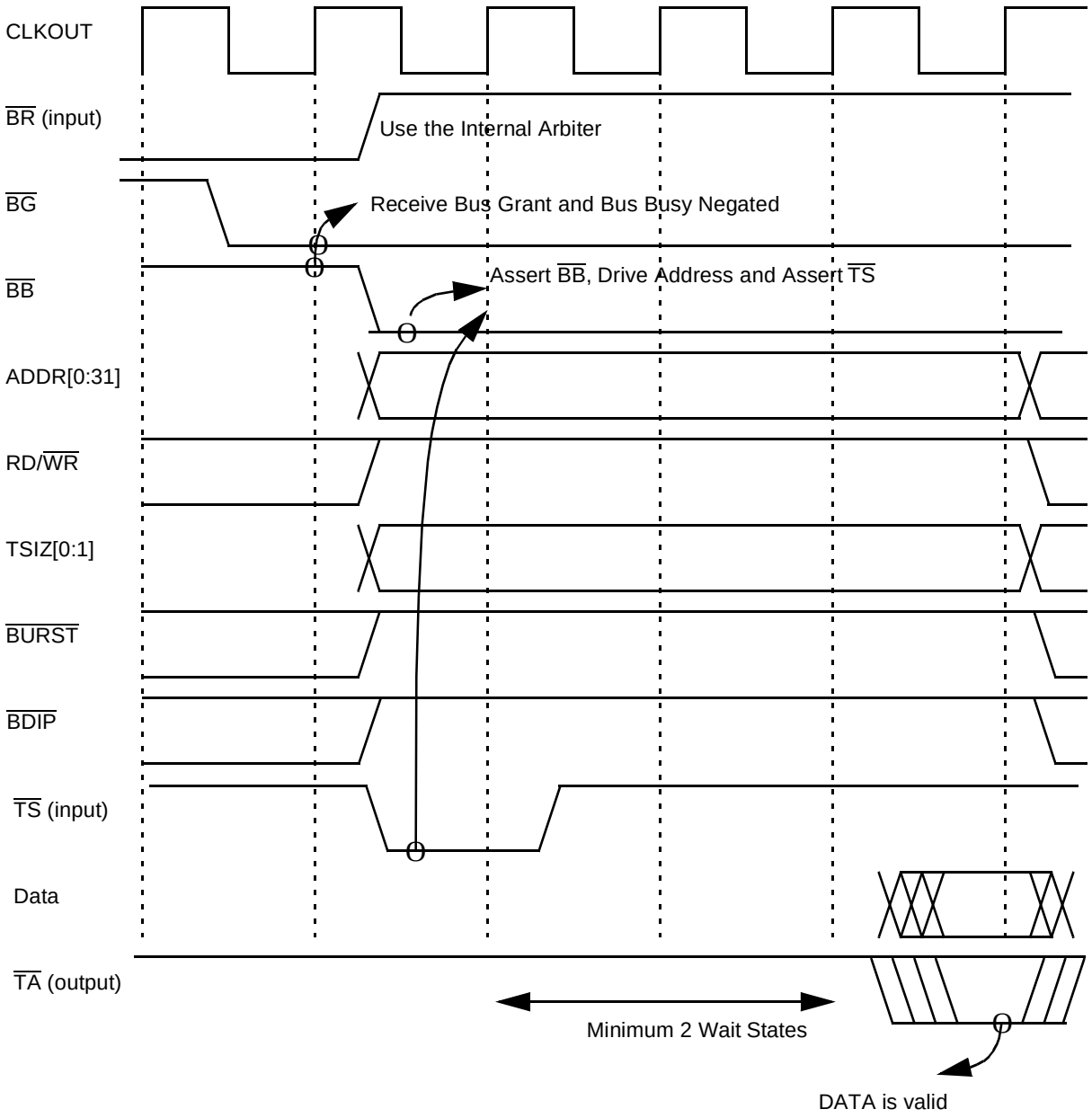


Figure 9-36 Peripheral Mode: External Master Reads from MPC565 / MPC566 — Two Wait States

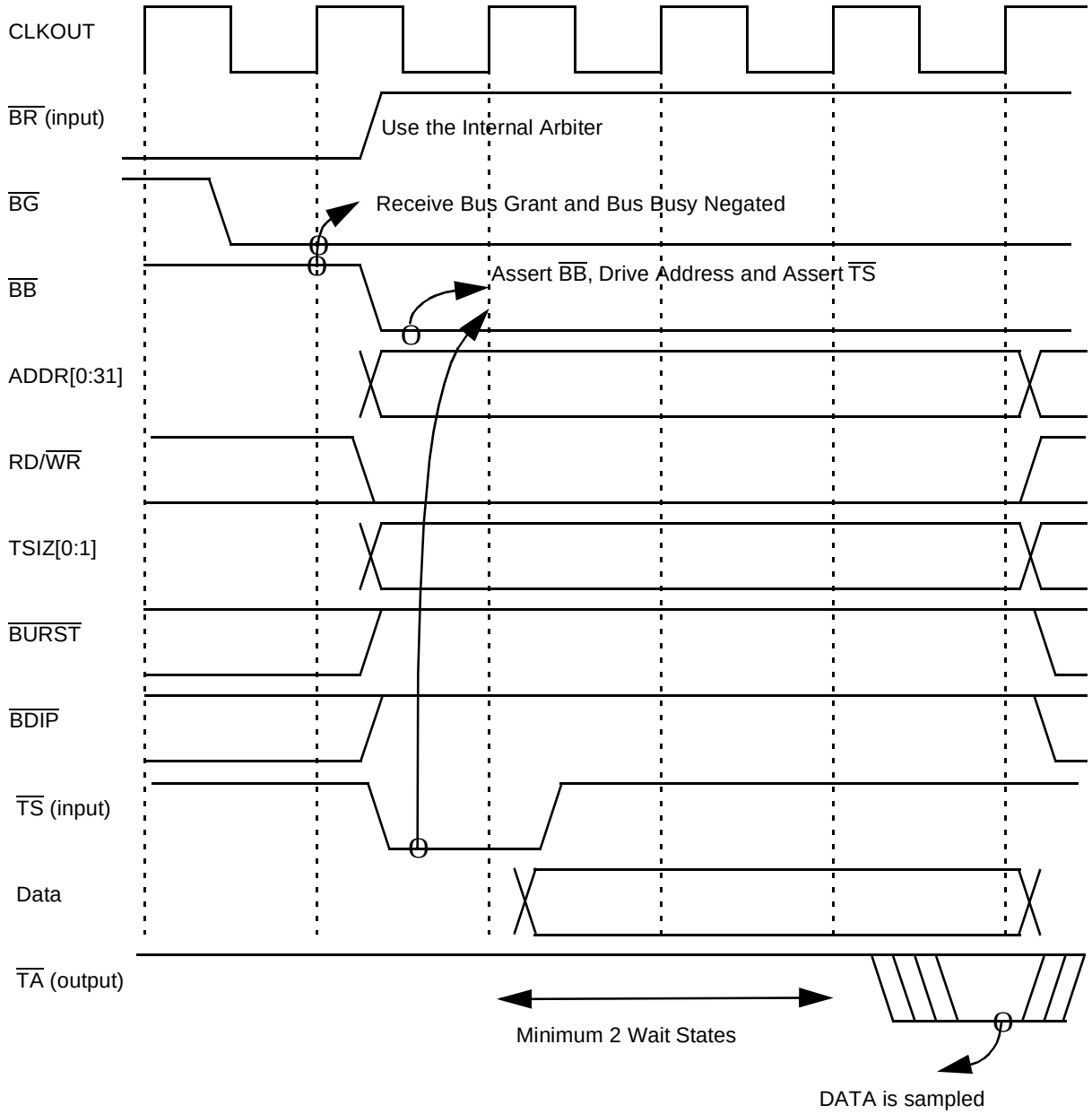


Figure 9-37 Peripheral Mode: External Master Writes to MPC565 / MPC566 — Two Wait States

9.5.12 Contention Resolution on External Bus

When the MPC565 / MPC566 is in slave mode, external master access to the MPC565 / MPC566 internal bus can be terminated with relinquish and retry in order to allow a pending internal-to-external access to be executed. The $\overline{\text{RETRY}}$ signal functions as an output that signals the external master to release the bus ownership and retry the access after one clock.

Figure 9-38 describes the flow of an external master retried access. **Figure 9-39** shows the timing when an external access is retried and a pending internal-to-external access follows.

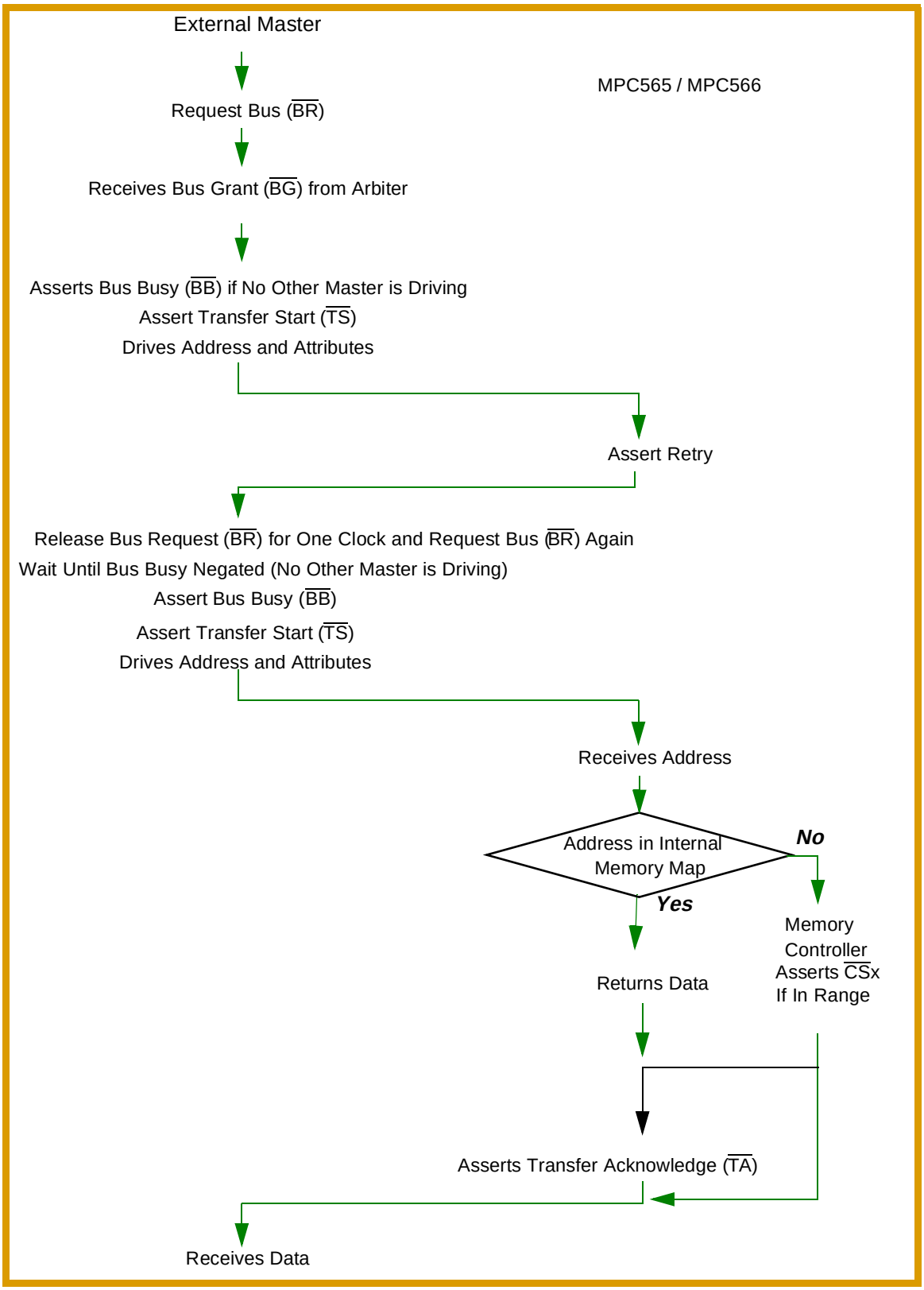
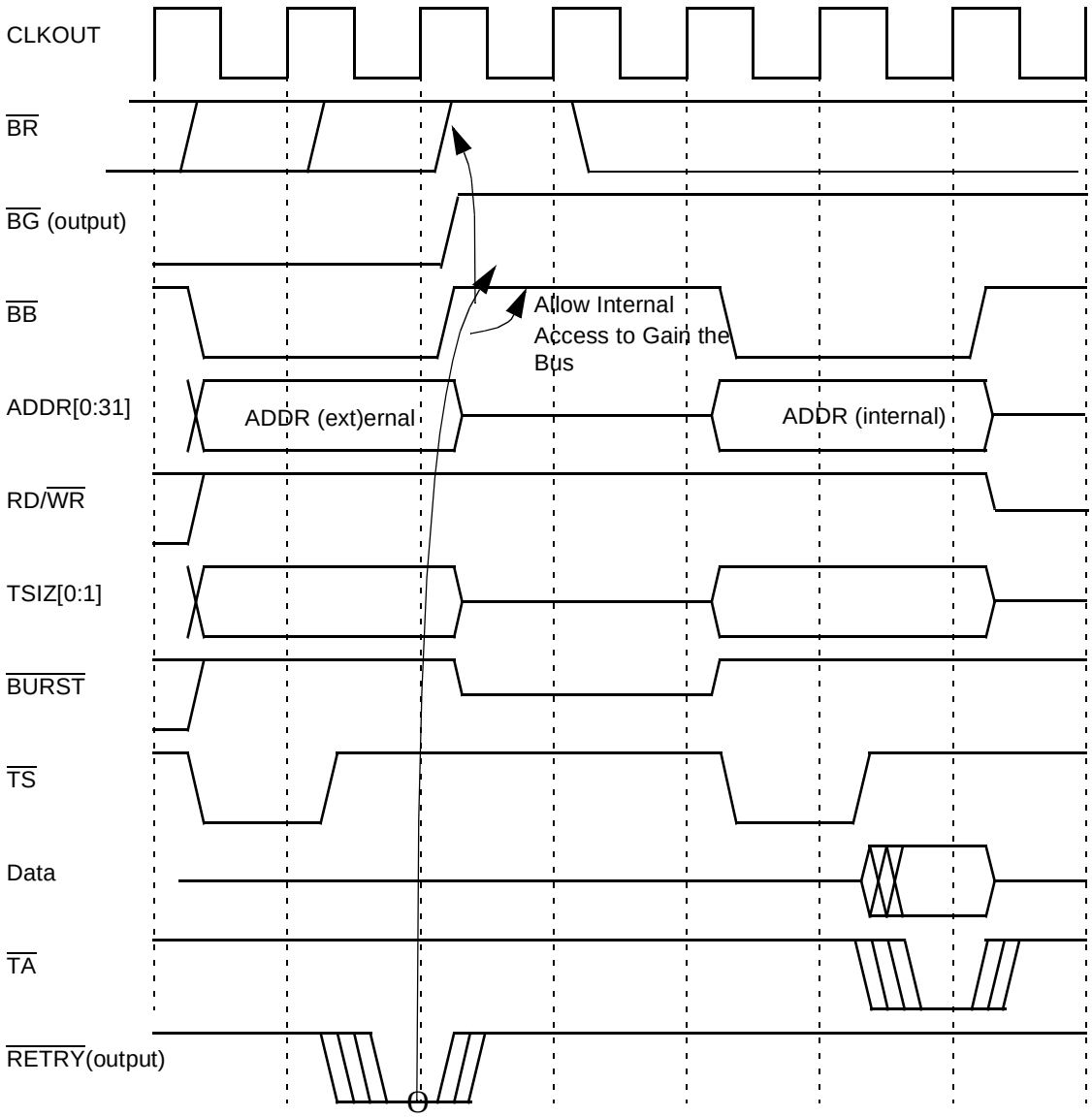


Figure 9-38 Flow of Retry of External Master Read Access



Note: the delay for the internal to external cycle may be one clock or greater.

Figure 9-39 Retry of External Master Access (Internal Arbiter)

9.5.13 Show Cycle Transactions

Show cycles are representations of RCPU accesses of the to MPC565 / MPC566 internal devices. These accesses are driven externally for emulation, visibility, and debugging purposes. A show cycle can have one address phase and one data phase, or just an address phase in the case of instruction show cycles. The cycle can be a write or a read access. The data for both the read and write accesses should be driven by the bus master. (This is different from normal bus read and write accesses.) The address and data of the show cycle must each be valid on the bus for one clock. The data phase must not require a transfer acknowledge to terminate the bus show cycle.

In a burst show cycle only the first data beat is shown externally. Refer to **Table 9-8** for show cycle transaction encodings.

Instruction show cycle bus transactions have the following characteristics: (see **Figure 9-40**)

- One clock cycle
- Address phase only (in “Decompression On” mode part of “compressed” address is driven on data lines together with address lines. External bus interface adds one clock delay between a read cycle and such show cycle.
- $\overline{STS}$ assertion only (no $\overline{TA}$ assertion)

The “compressed” address is driven on the external bus in following manner:

- ADDR[0:29] = the word “base” address;
- DATA[0] = operating mode:
  - — 0 = Decompression Off mode;
  - — 1 = Decompression On mode;
- DATA[1:4] = “bit” pointer

See **SECTION 4 BURST BUFFER CONTROLLER MODULE** for more details about decompression mode.

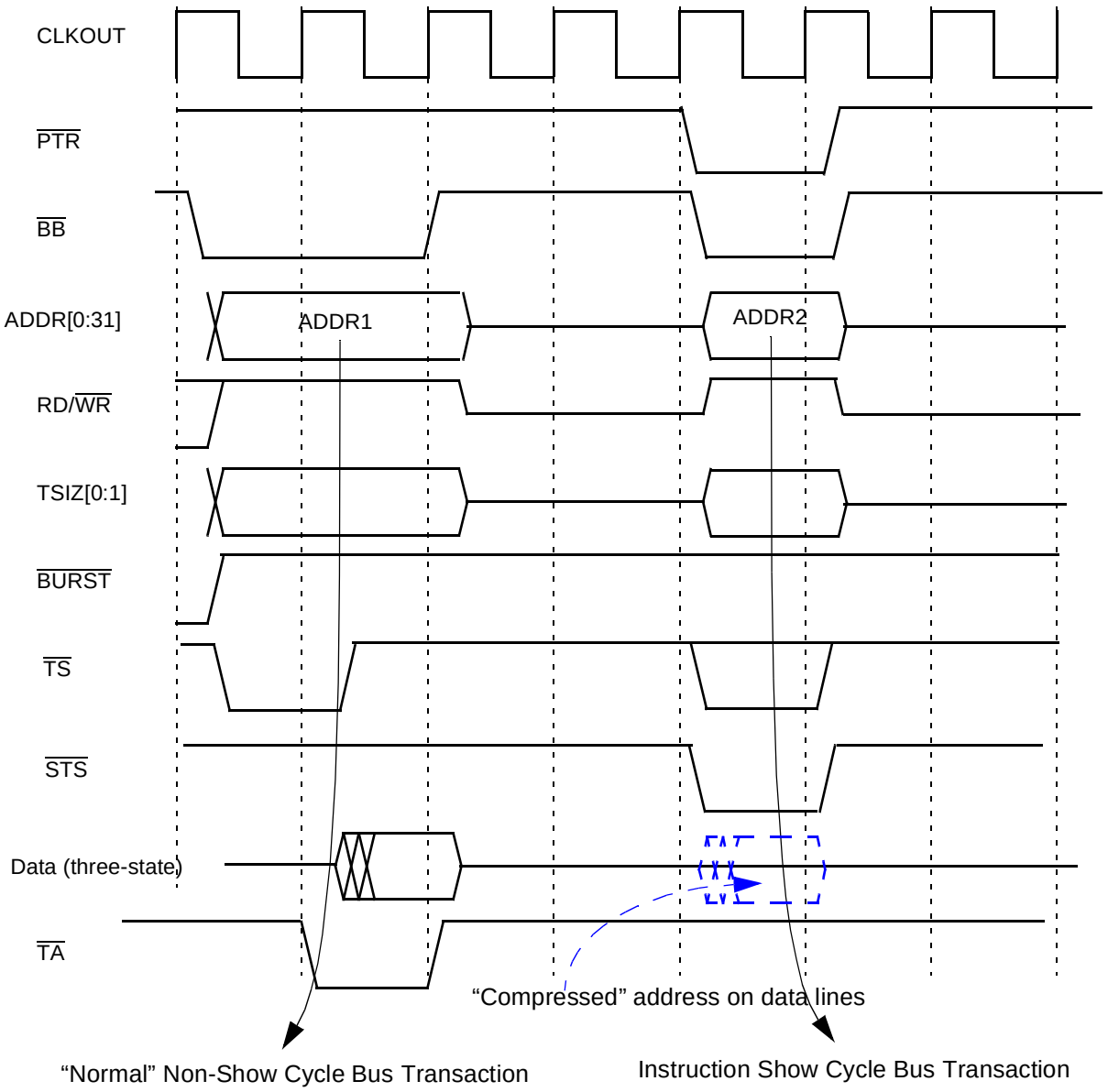


Figure 9-40 Instruction Show Cycle Transaction

Both read and write data show cycles have the following characteristics: (see **Figure 9-41**)

- Two clock cycle duration
- Address valid for two clock cycles
- Data is valid only in the second clock cycle
- $\overline{STS}$ signal only is asserted (no $\overline{TA}$ or $\overline{TS}$)

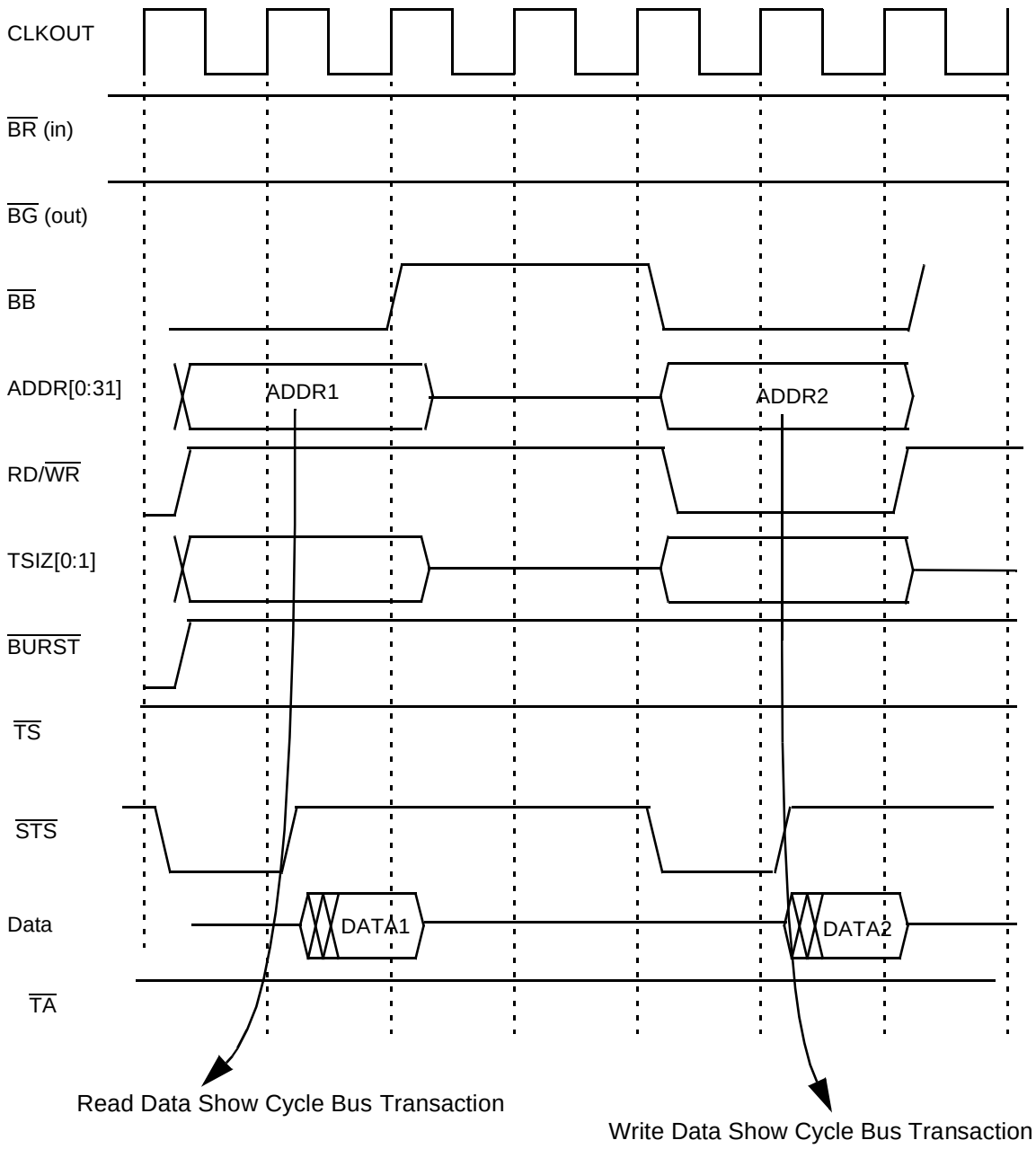


Figure 9-41 Data Show Cycle Transaction

SECTION 10
MEMORY CONTROLLER

The memory controller generates interface signals to support a glueless interface to external memory and peripheral devices. It supports four regions, each with its own programmed attributes. The four regions are reflected on four chip-select pins. Read and write strobes are also provided.

The memory controller operates in parallel with the external bus interface to support external cycles. When an access to one of the memory regions is initiated, the memory controller takes ownership of the external signals and controls the access until its termination. Refer to **Figure 10-1**.

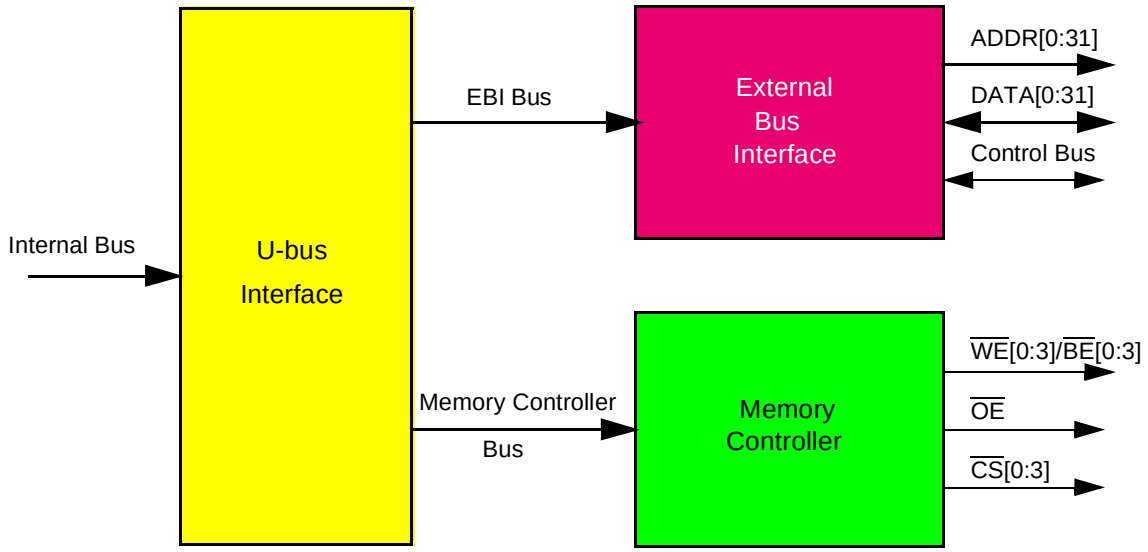


Figure 10-1 Memory Controller Function Within the USIU

10.1 Overview

The memory controller provides a glueless interface to EPROM, static RAM (SRAM), flash EPROM (FEPROM), and other peripherals. The general-purpose chip-selects are available on lines $\overline{CS}$[0] through $\overline{CS}$[3]. $\overline{CS}$[0] also functions as the global (boot)

chip-select for accessing the boot flash EEPROM. The chip select allows zero to 30 wait states.

Figure 10-2 is a block diagram of the MPC565 / MPC566 memory controller.

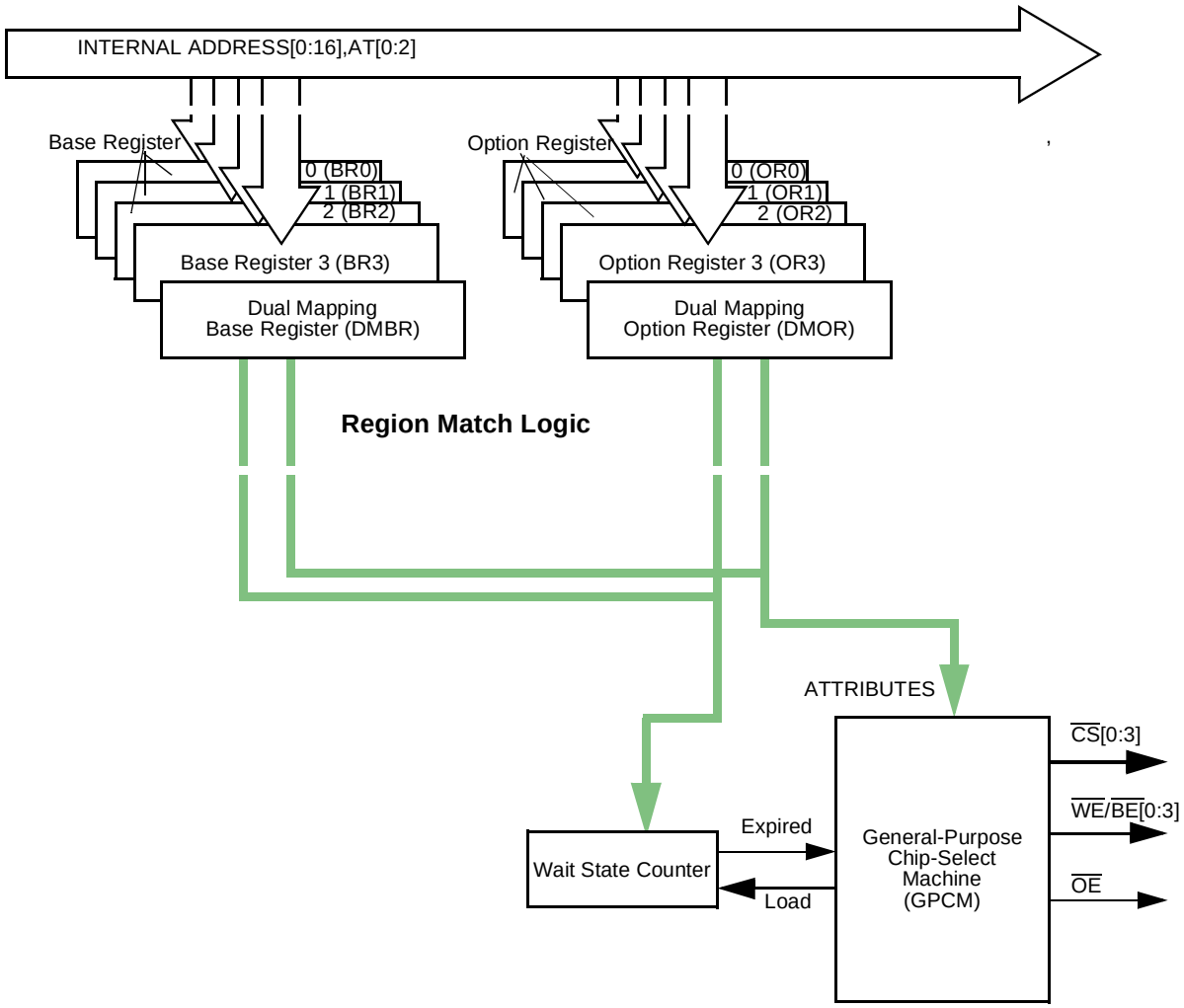


Figure 10-2 Memory Controller Block Diagram

Most memory controller features are common to all four banks. (For features unique to the $\overline{CS0}$ bank, refer to **10.4 Global (Boot) Chip-Select Operation**.) A full 32-bit address decode for each memory bank is possible with 17 bits having address masking. The full 32-bit decode is available, even if all 32 address bits are not sent to the MPC565 / MPC566 pins.

Each memory bank includes a variable block size of 32 Kbytes, 64 Kbytes and up to four Gbytes. Each memory bank can be selected for read-only or read/write operation.

The access to a memory bank can be restricted to certain address type codes for system protection. The address type comparison occurs with a mask option as well.

From zero to 30 wait states can be programmed with $\overline{TA}$ generation. Four byte-write and read-enable signals ($\overline{WE}/\overline{BE}$[0:3]) are available for each byte that is written to memory. An output enable ($\overline{OE}$) signal is provided to eliminate external glue logic. A memory transfer start ($\overline{MTS}$) strobe permits one master on a bus to access external memory through the chip selects on another.

The memory controller functionality allows MPC565 / MPC566-based systems to be built with little or no glue logic. A minimal system using no glue logic is shown in **Figure 10-3**. In this example $\overline{CS}$[0] is used for the 16-bit boot EPROM and $\overline{CS}$[1] is used for the 32-bit SRAM. The $\overline{WE}/\overline{BE}$[0:3] signals are used both to program the EPROM and to enable write access to various bytes in the RAM.

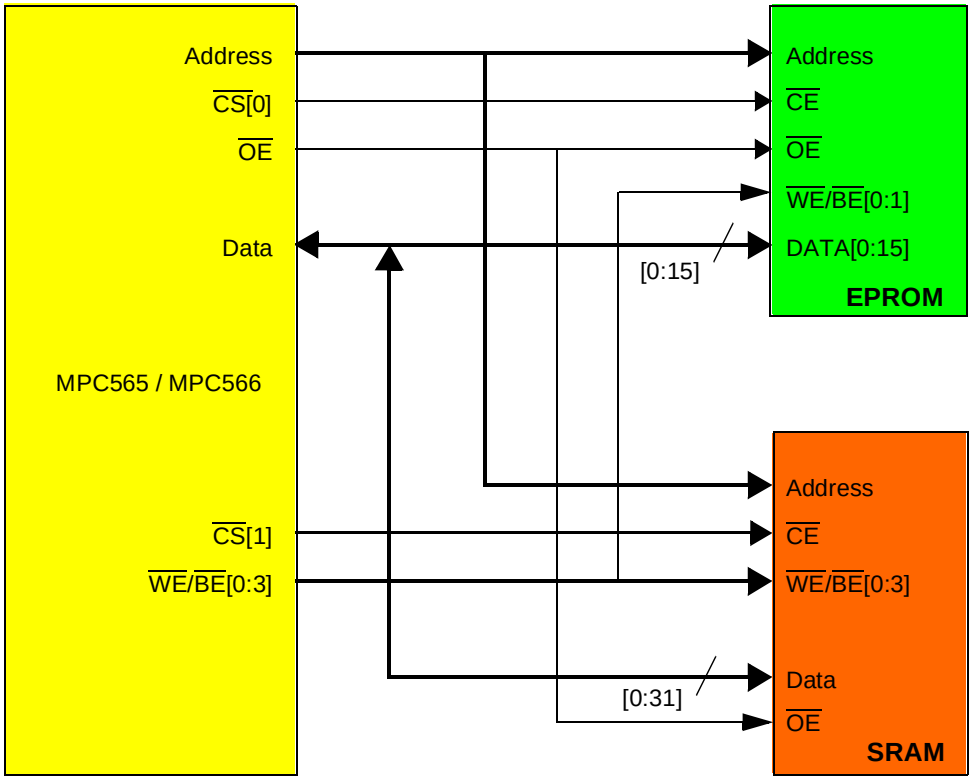


Figure 10-3 MPC565 / MPC566 Simple System Configuration

10.2 Memory Controller Architecture

The memory controller consists of a basic machine that handles the memory access cycle: the general-purpose chip-select machine (GPCM).

When a new access to external memory is requested by any of the internal masters, the address of the transfer (with 17 bits having mask) and the address type (with three

bits having mask) are compared to each one of the valid banks defined in the memory controller. Refer to **Figure 10-4**.

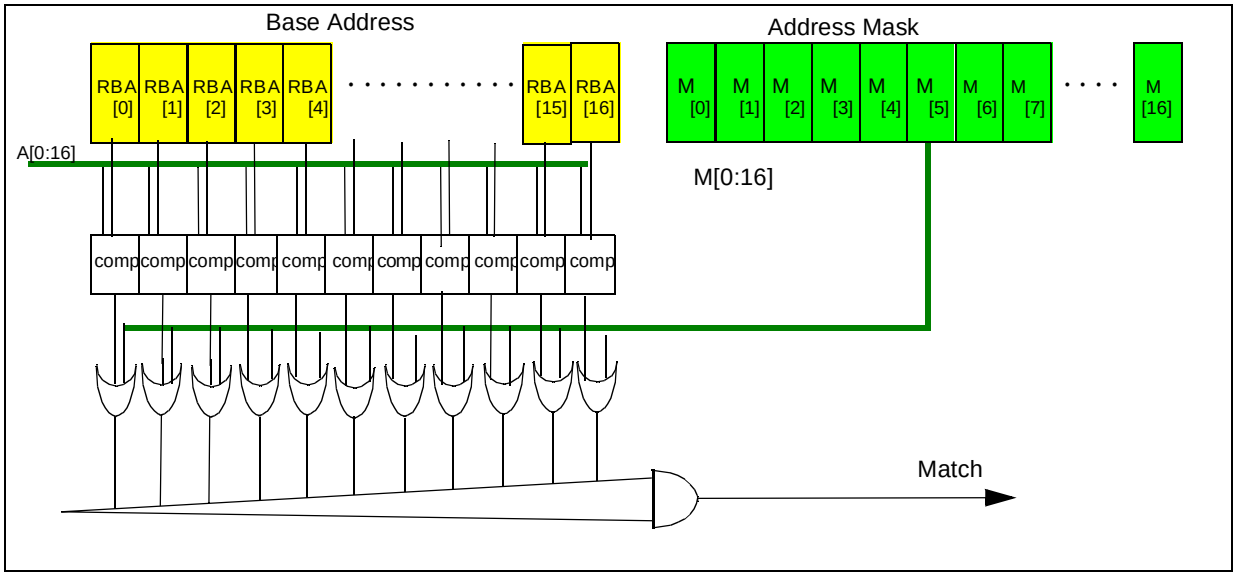


Figure 10-4 Bank Base Address and Match Structure

When a match is found on one of the memory banks, its attributes are selected for the functional operation of the external memory access:

- Read-only or read/write operation
- Number of wait states for a single memory access, and for any beat in a burst access
- Burst-inhibit indication. Internal burst requests are still possible during burst-inhibited cycles; the memory controller emulates the burst cycles
- Port size of the external device

Note that if more than one region matches the internal address supplied, then the lowest region is selected to provide the attributes and the chip select.

10.2.1 Associated Registers

Status bits for each memory bank are found in the memory control status register (MSTAT). The MSTAT reports write-protect violations for all the banks.

Each of the four banks has a base register (BR) and an option register (OR). The BRx and ORx registers contain the attributes specific to bank x. The base register contains a valid bit (V) that indicates that the register information for that chip select is valid.

10.2.2 Port Size Configuration

The memory controller supports dynamic bus sizing. Defined 8-bit ports can be accessed as odd or even bytes. Defined 16-bit ports, when connected to data bus lines zero to 15, can be accessed as odd bytes, even bytes, or even half-words. Defined 32-bit ports can be accessed as odd bytes, even bytes, odd half-words, even half-words, or words on word boundaries. The port size is specified by the PS bits in the base register.

10.2.3 Write-Protect Configuration

The WP bit in each base register can restrict write access to its range of addresses. Any attempt to write this area results in the associated WPER bit being set in the MSTAT.

If an attempt to access an external device results in a write-protect violation, the memory controller considers the access to be no match. No chip-select line is asserted externally, and the memory controller does not terminate the cycle. The external bus interface generates a normal cycle on the external bus. Since the memory controller does not acknowledge the cycle internally, the cycle may be terminated by external logic asserting $\overline{TA}$ or by the on-chip bus monitor asserting $\overline{TEA}$.

10.2.4 Address and Address Space Checking

The base address is written to the BR. The address mask bits for the address are written to the OR. The address type access value, if desired, is written to the AT bits in the BR. The ATM bits in the OR can be used to mask this value. If address type checking is not desired, program the ATM bits to zero.

Each time an external bus cycle access is requested, the address and address type are compared with each one of the banks. If a match is found, the attributes defined for this bank in its BR and OR are used to control the memory access. If a match is found in more than one bank, the lowest bank matched handles the memory access (e.g., bank zero is selected over bank one).

NOTE

When an external master accesses a slave on the bus, the internal AT[0:2] lines reaching the memory controller are forced to 100.

10.2.5 Burst Support

Burst support is for read only. The memory controller supports burst accesses of external burstable memory. To enable bursts, clear the $\overline{BI}$ in the appropriate base register.

Bursts are four beats and non-wrapping. That is, the memory controller executes up to four one-word accesses, but when a modulo four limit is reached, the burst is terminated (even if fewer than four words have been accessed).

When the SIU initiates a burst access, if no match is found in any of the memory controller's regions then a burst access is initiated to the external bus. The termination of each beat for this access is externally controlled.

To support different types of memory devices, the memory controller supports two types of timing for the $\overline{\text{BDIP}}$ signal: normal and late.

NOTE

The $\overline{\text{BDIP}}$ pin itself is controlled by the external bus interface logic. Refer to **Figure 9-13** and **Figure 9-14** in **SECTION 9 EXTERNAL BUS INTERFACE**.

If the memory controller is used to support an external master accessing an external device with bursts, the $\overline{\text{BDIP}}$ input pin is used to indicate to the memory controller when the burst is terminated.

For addition details, refer to **9.5.3 Burst Transfer**.

10.3 Chip-Select Timing

The GPCM allows a glueless and flexible interface between the MPC565 and SRAM, EPROM, EEPROM, ROM devices and external peripherals. When an address and address type matches the values programmed in the BR and OR for one of the memory controller banks, the attributes for the memory cycle are taken from the OR and BR registers as well. These attributes include the following fields: CSNT, ACS, SCY, BSCY, WP, TRLX, BI, PS, and SETA.

Byte write and read-enable signals ($\overline{\text{WE}}$/$\overline{\text{BE}}$[0:3]) are available for each byte that is written to or read from memory. An output enable ($\overline{\text{OE}}$) signal is provided to eliminate external glue logic for read cycles. Upon system reset, a global (boot) chip select is available. This provides a boot ROM chip select before the system is fully configured. **Table 10-1** summarizes the chip-select timing options.

Table 10-1 Timing Attributes Summary

Timing Attribute	Bits/Fields	Description
Access speed	TRLX	The TRLX (timing relaxed) bit determines strobe timing to be fast or relaxed.
Intercycle space time	EHTR	The EHTR (extended hold time on read accesses) bit is provided for devices that have long disconnect times from the data bus on read accesses. EHTR specifies whether the next cycle is delayed one clock cycle following a read cycle, to avoid data bus contentions. EHTR applies to all cycles following a read cycle except for another read cycle to the same region.
Synchronous or asynchronous device	ACS, CSNT	The ACS (address-to-chip-select setup) and CSNT (chip-select negation time) bits cause the timing of the strobes to be the same as the address bus timing, or cause the strobes to have setup and hold times relative to the address bus.
Wait states	SCY, BSCY, SETA, TRLX	From zero to 15 wait states can be programmed for any cycle that the memory controller generates. The transfer is then terminated internally. In simplest case, the cycle length equals (2 + SCY) clock cycles, where SCY represents the programmed number of wait states (cycle length in clocks). The number of wait states is doubled if the TRLX bit is set. When the SETA (external transfer acknowledge) bit is set, $\overline{TA}$ must be generated externally, so that external hardware determines the number of wait states.

NOTE

When a bank is configured for $\overline{TA}$ to be generated externally (SETA bit is set) and the TRLX is set, the memory controller requires the external device to provide at least one wait state before asserting $\overline{TA}$ to complete the transfer. In this case, the minimum transfer time is three clock cycles.

The internal $\overline{TA}$ generation mode is enabled if the SETA bit in the OR register is negated. However, if the $\overline{TA}$ pin is asserted externally at least two clock cycles before the wait states counter has expired, this assertion terminates the memory cycle. When SETA is cleared, it is forbidden to assert external $\overline{TA}$ less than two clocks before the wait states counter expires.

10.3.1 Memory Devices Interface Example

Figure 10-5 describes the basic connection between the MPC565 and a static memory device. In this case $\overline{CS}$x is connected directly to the chip enable ($\overline{CE}$) of the memory device. The $\overline{WE}/\overline{BE}$[0:3] lines are connected to the respective $\overline{W}$ in the memory device where each $\overline{WE}/\overline{BE}$ line corresponds to a different data byte.

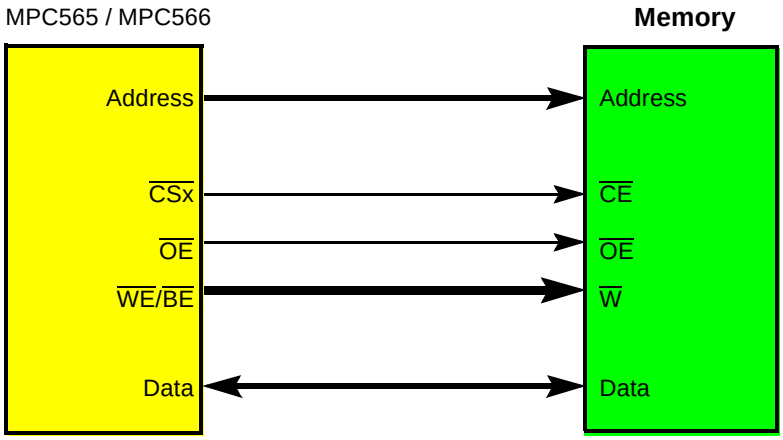
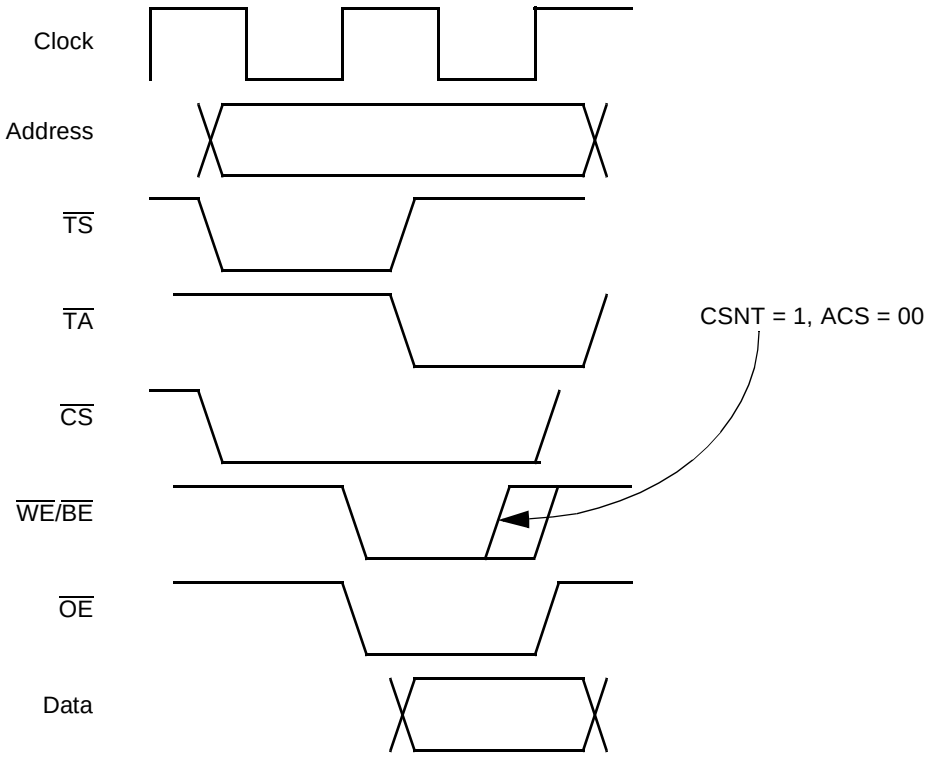


Figure 10-5 MPC565 GPCM–Memory Devices Interface

In **Figure 10-6**, the $\overline{CS}$x timing is the same as that of the address lines output. The strobes for the transaction are supplied by the $\overline{OE}$ and the $\overline{WE}$/$\overline{BE}$ lines (if programmed as $\overline{WE}$/$\overline{BE}$). Because the ACS bits in the corresponding ORx register = 00, $\overline{CS}$ is asserted at the same time that the address lines are valid.

NOTE

Because CSNT is set, the $\overline{WE}$ signal is negated a quarter of a clock earlier than normal.



NOTE: In this and subsequent timing diagrams in this section, the data bus refers to a read cycle. In a write cycle, the data immediately follows TS.

Figure 10-6 Memory Devices Interface Basic Timing (ACS = 00,TRLX = 0)

10.3.2 Peripheral Devices Interface Example

Figure 10-7 illustrates the basic connection between the MPC565 and an external peripheral device. In this case CSx is connected directly to the chip enable (CE) of the memory device and the R/W line is connected to the R/W in the peripheral device. The CSx line is the strobe output for the memory access.

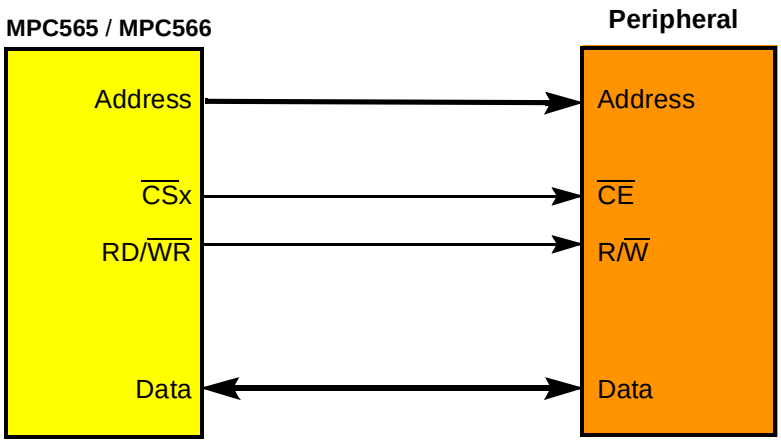


Figure 10-7 Peripheral Devices Interface

The $\overline{CS}$x timing is defined by the setup time required between the address lines and the $\overline{CE}$ line. The memory controller allows specification of the $\overline{CS}$ timing to meet the setup time required by the peripheral device. This is accomplished through the ACS field in the base register. In **Figure 10-8**, the ACS bits are set to 11, so $\overline{CS}$x is asserted half a clock cycle after the address lines are valid.

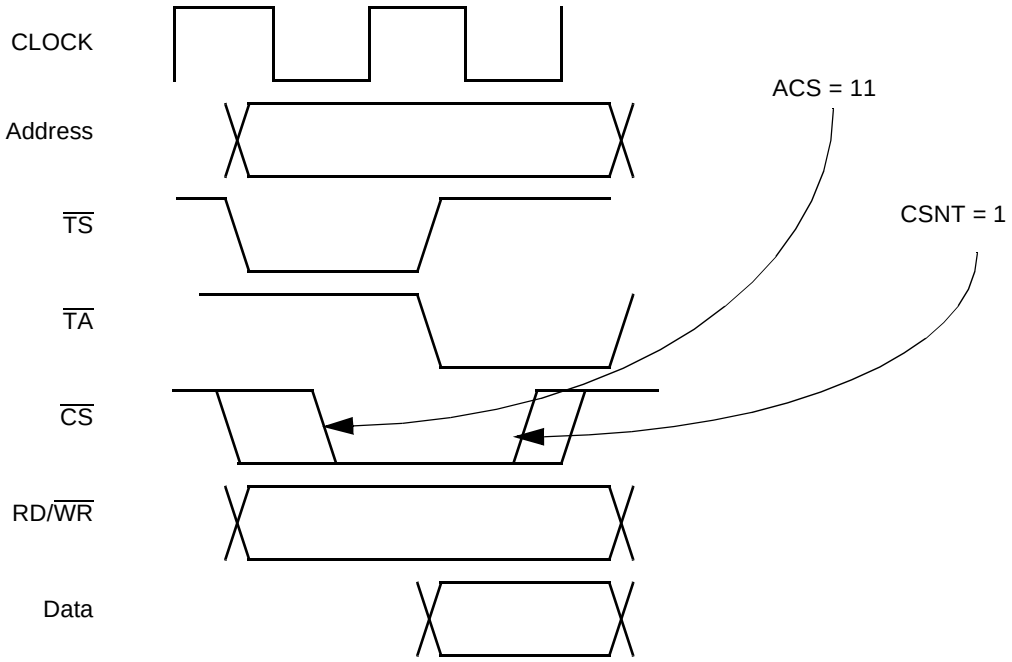


Figure 10-8 Peripheral Devices Basic Timing (ACS = 11,TRLX = 0)

10.3.3 Relaxed Timing Examples

The TRLX field is provided for memory systems that need a more relaxed timing between signals. When TRLX is set and ACS = 0b00, the memory controller inserts an additional cycle between address and strobes ($\overline{CS}$ line and $\overline{WE}$/$\overline{OE}$).

When TRLX and CSNT are both set in a write to memory, the strobe lines ($\overline{WE}$/$\overline{BE}$[0:3] and $\overline{CS}$, if ACS = 0b00) are negated one clock earlier than in the regular case.

Note that in the case of a bank selected to work with external transfer acknowledge (SETA = 1) and TRLX = 1, the memory controller does not support external devices providing $\overline{TA}$ to complete the transfer with zero wait states. The minimum access duration in this case equals three clock cycles.

Figure 10-9 shows a read access with relaxed timing. Note the following:

- Strobes ($\overline{OE}$ and $\overline{CS}$) assertion time is delayed one clock relative to address (TRLX bit set effect).
- Strobe ($\overline{CS}$) is further delayed (half-clock) relative to address due to ACS field being set to 11.
- Total cycle length = 5, is determined as follows:
  - — Two clocks for basic cycle
  - — SCY = 1 determines 1 wait state, which is multiplied by two due to TRLX being set.
  - — Extra clock is added due to TRLX effect on the strobes.

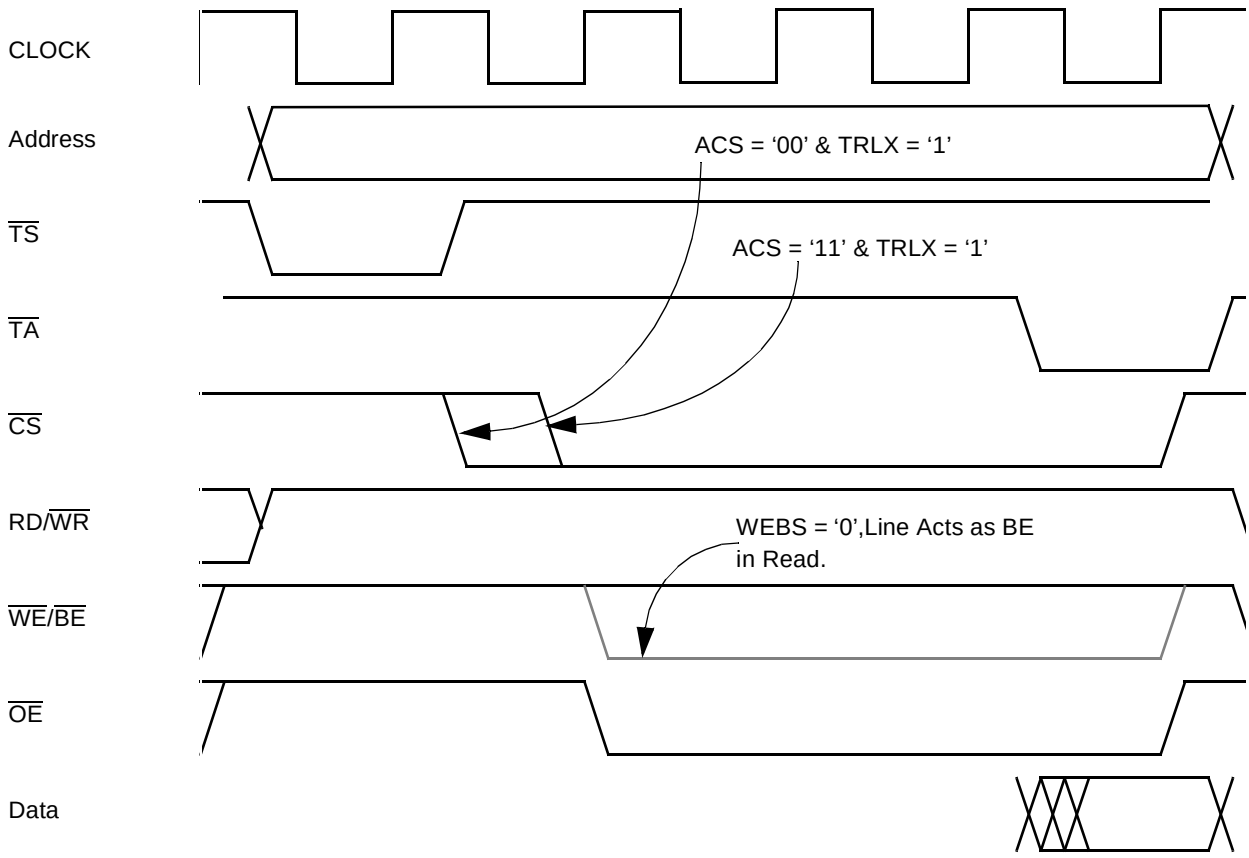


Figure 10-9 Relaxed Timing — Read Access (ACS = 11, SCY = 1, TRLX = 1)

Figure 10-10 through **Figure 10-12** are examples of write accesses using relaxed timing. In **Figure 10-10**, note the following points:

- Because TRLX is set, assertion of the $\overline{CS}$ and $\overline{WE}$ strobes is delayed by one clock cycle.
- $\overline{CS}$ assertion is delayed an additional one quarter clock cycle because ACS = 10.
- The total cycle length = three clock cycles, determined as follows:
  - — The basic memory cycle requires two clock cycles.
  - — An extra clock cycle is required due to the effect of TRLX on the strobes.

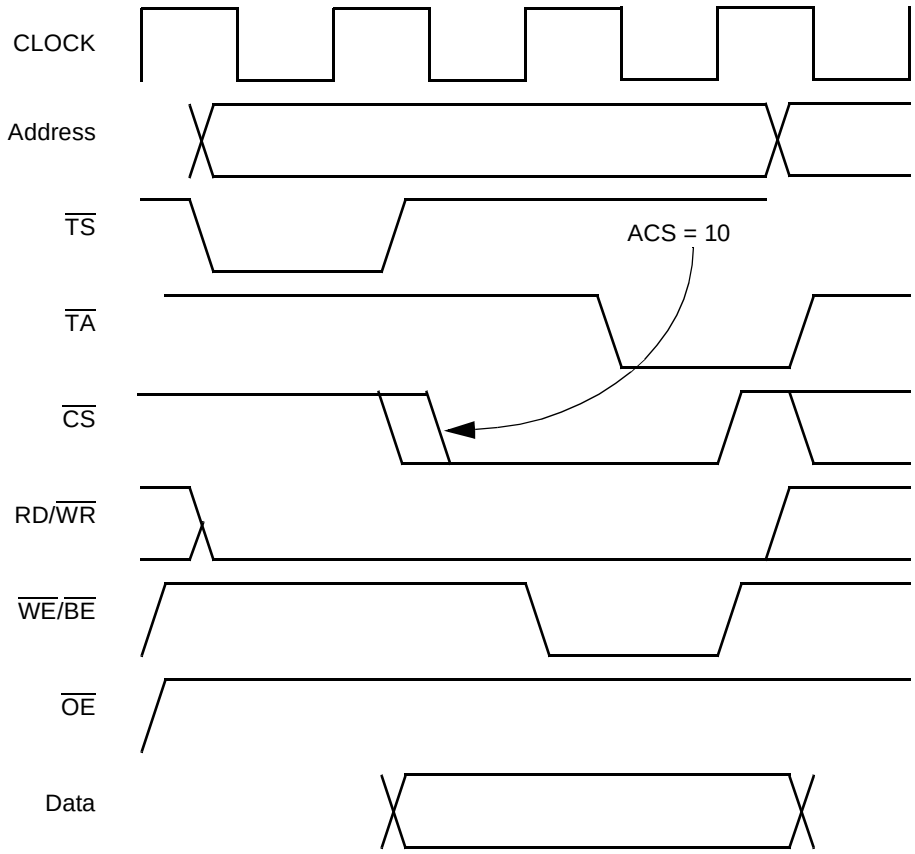


Figure 10-10 Relaxed Timing — Write Access (ACS = 10, SCY = 0, CSNT = 0, TRLX = 1)

In **Figure 10-11**, note the following:

- Because the TRLX bit is set, the assertion of the $\overline{CS}$ and $\overline{WE}$ strobes is delayed by one clock cycle.
- Because ACS = 11, the assertion of $\overline{CS}$ is delayed an additional half clock cycle.
- Because CSNT = 1, $\overline{WE}$ is negated one clock cycle earlier than normal. (Refer to **Figure 10-6**). The total cycle length is four clock cycles, determined as follows:
  - — The basic memory cycle requires two clock cycles.
  - — Two extra clock cycles are required due to the effect of TRLX on the assertion and negation of the $\overline{CS}$ and $\overline{WE}$ strobes.

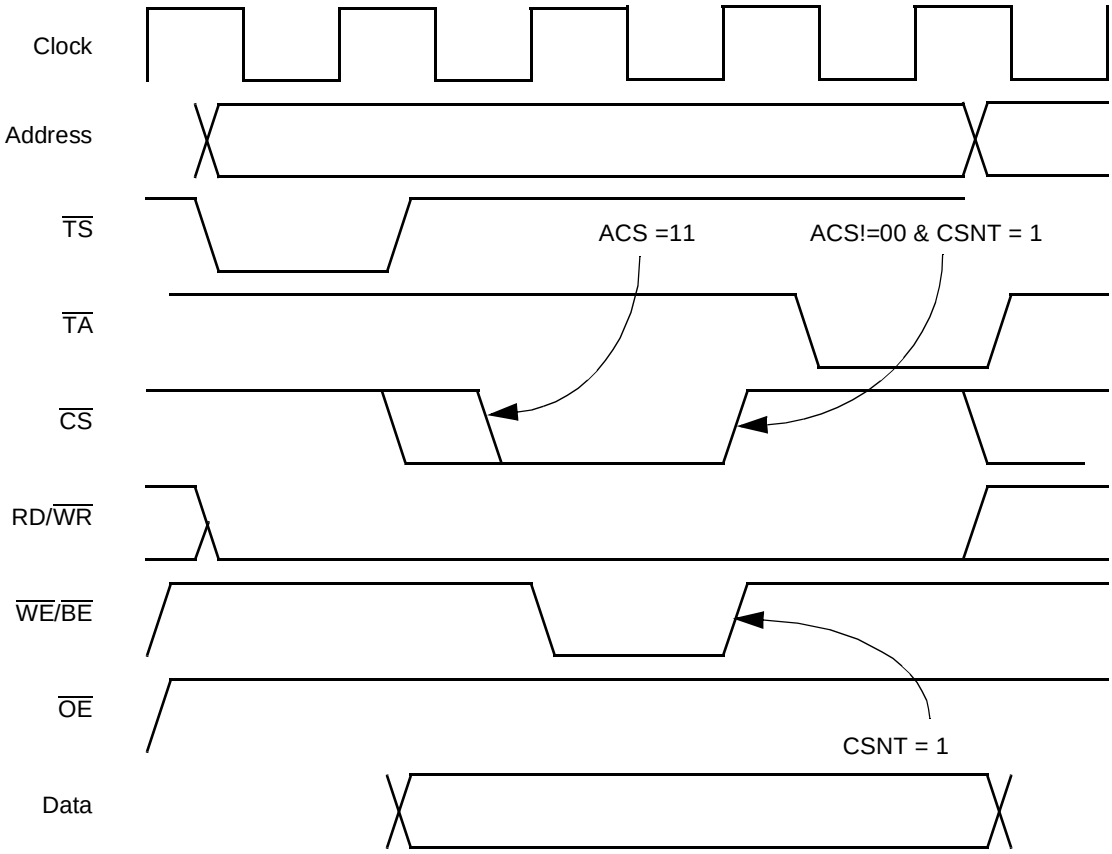


Figure 10-11 Relaxed Timing — Write Access (ACS = 11, SCY = 0, CSNT = 1, TRLX = 1)

In **Figure 10-12**, notice the following:

- Because ACS = 0, TRLX being set does not delay the assertion of the $\overline{CS}$ and $\overline{WE}$ strobes.
- Because CSNT = 1, $\overline{WE}$/$\overline{BE}$ is negated one clock cycle earlier than normal. (Refer to **Figure 10-6**).
- $\overline{CS}$ is not negated one clock cycle earlier, since ACS = 00.
- The total cycle length is three clock cycles, determined as follows:
  - — The basic memory cycle requires two clock cycles.
  - — One extra clock cycles are required due to the effect of TRLX on the negation of the $\overline{WE}$/$\overline{BE}$ strobes.

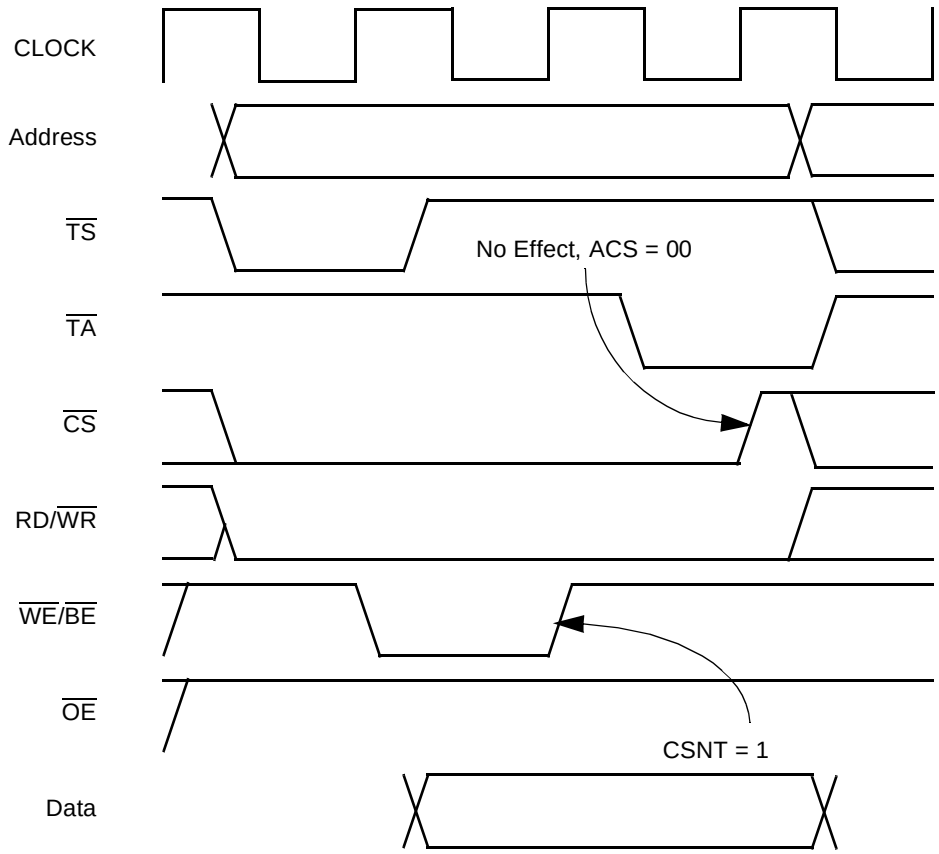


Figure 10-12 Relaxed Timing — Write Access (ACS = 00, SCY = 0, CSNT = 1, TRLX = 1

10.3.4 Extended Hold Time on Read Accesses

For devices that require a long disconnection time from the data bus on read accesses, the bit EHTR in the corresponding OR register can be set. In this case any MPC565 / MPC566 access to the external bus following a read access to the referred

memory bank is delayed by one clock cycle unless it is a read access to the same bank. **Figure 10-13** through **Figure 10-16** show the effect of the EHTR bit on memory controller timing.

Figure 10-13 shows a write access following a read access. Because EHTR = 0, no extra clock cycle is inserted between memory cycles.

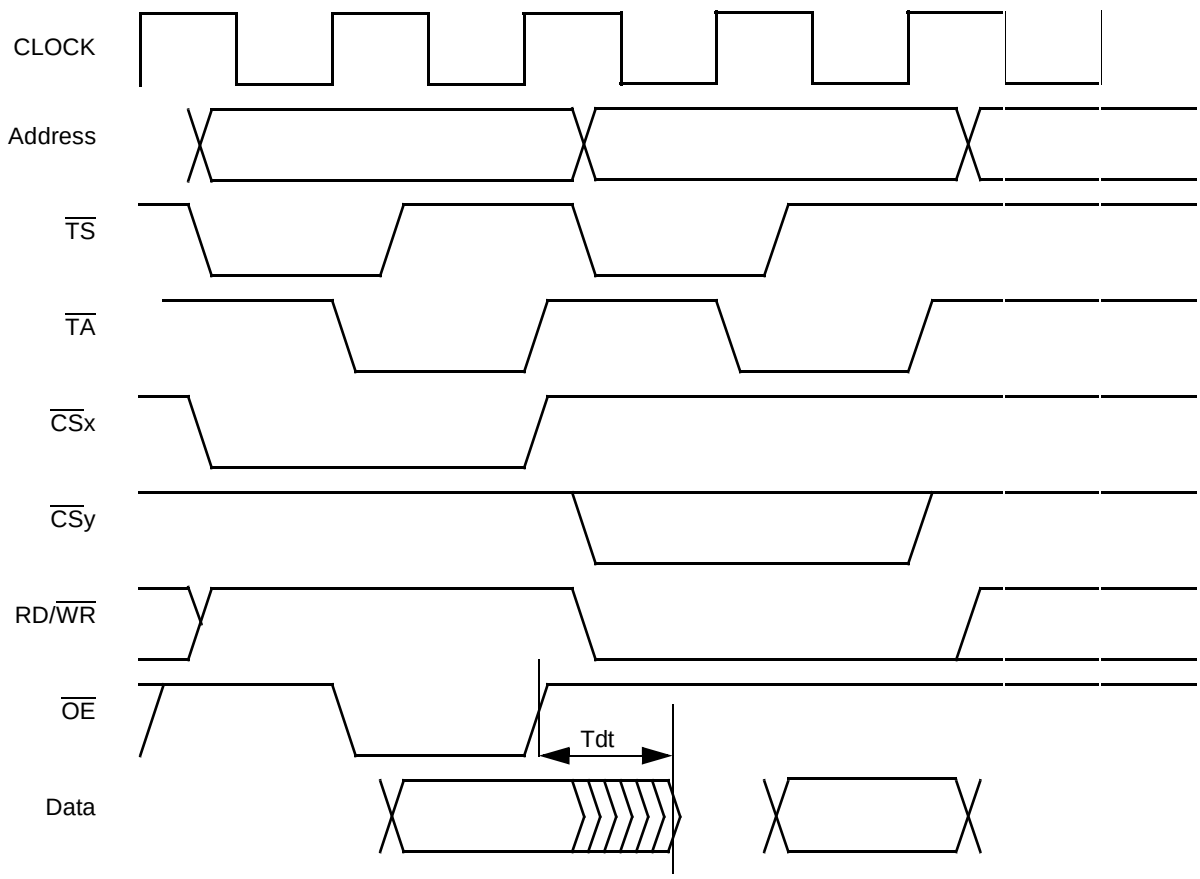


Figure 10-13 Consecutive Accesses (Write After Read, EHTR = 0)

Figure 10-14 shows a write access following a read access when EHTR = 1. An extra clock is inserted between the cycles. For a write cycle following a read, this is true regardless of whether both accesses are to the same region.

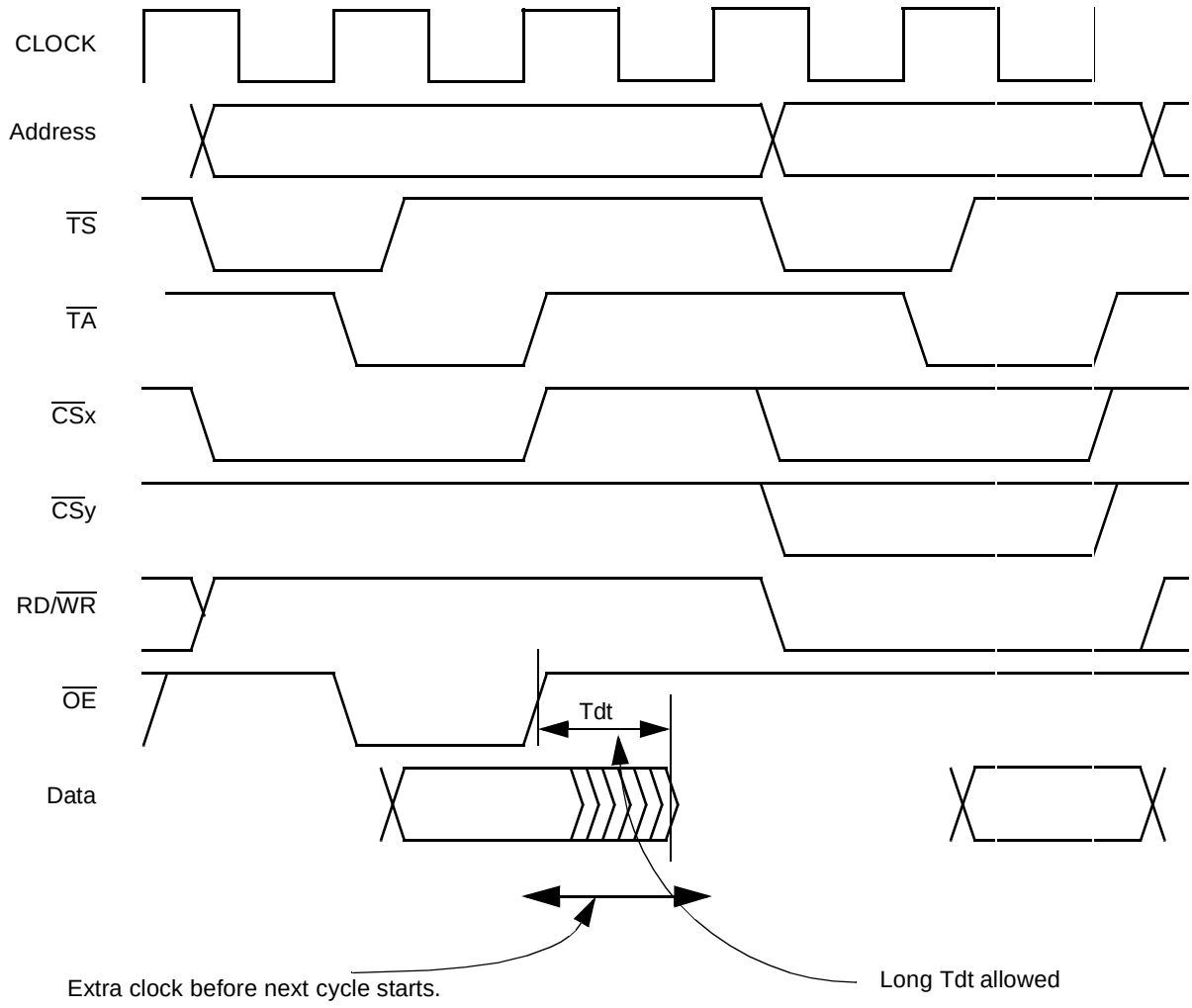


Figure 10-14 Consecutive Accesses (Write After Read, EHTR = 1)

Figure 10-15 shows consecutive accesses from different banks. Because EHTR = 1 (and the accesses are to different banks), an extra clock cycle is inserted.

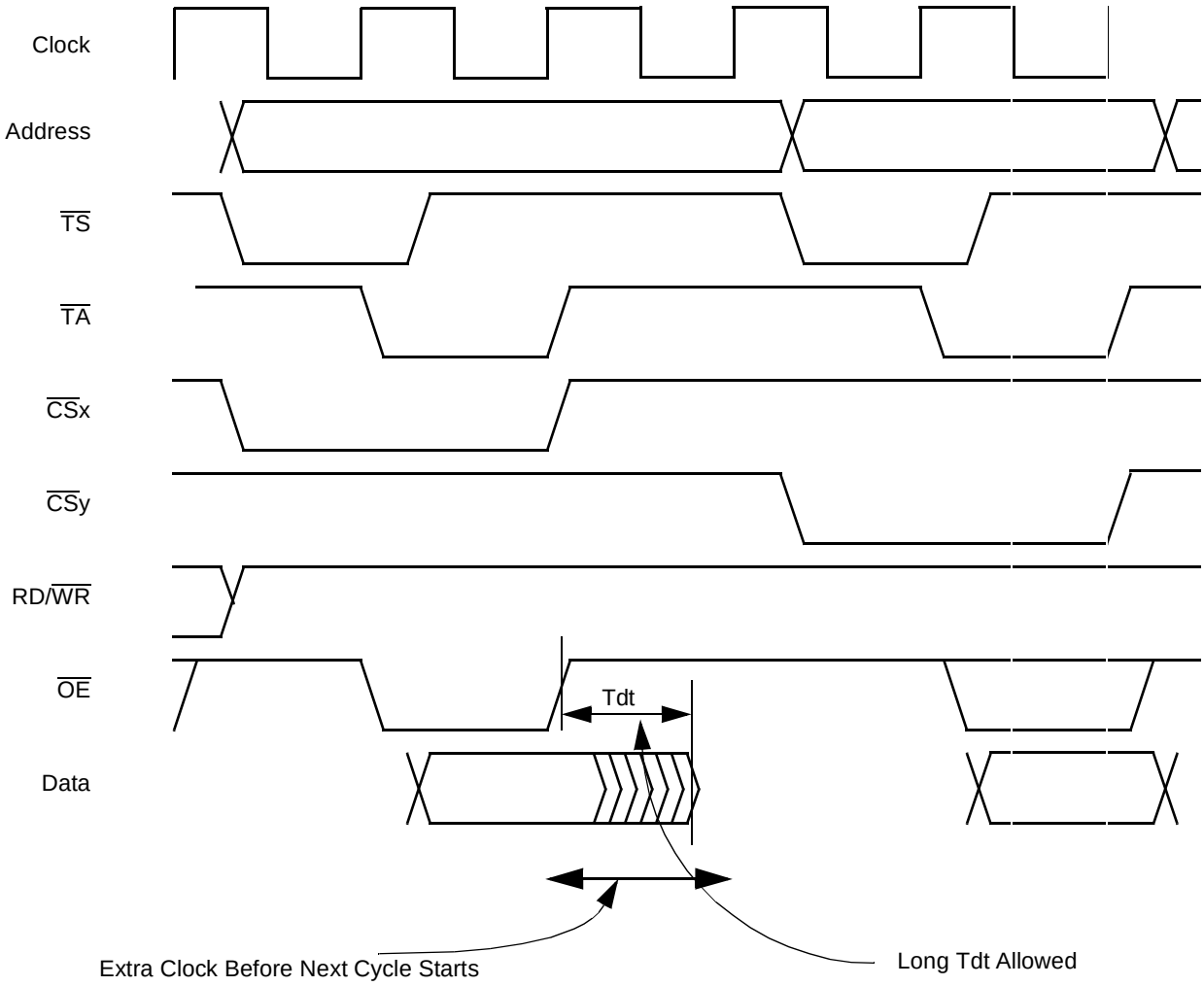


Figure 10-15 Consecutive Accesses (Read After Read From Different Banks, EHTR = 1)

Figure 10-16 shows two consecutive read cycles from the same bank. Even though EHTR = 1, no extra clock cycle is inserted between the memory cycles. (In the case of two consecutive read cycles to the same region, data contention is not a concern.)

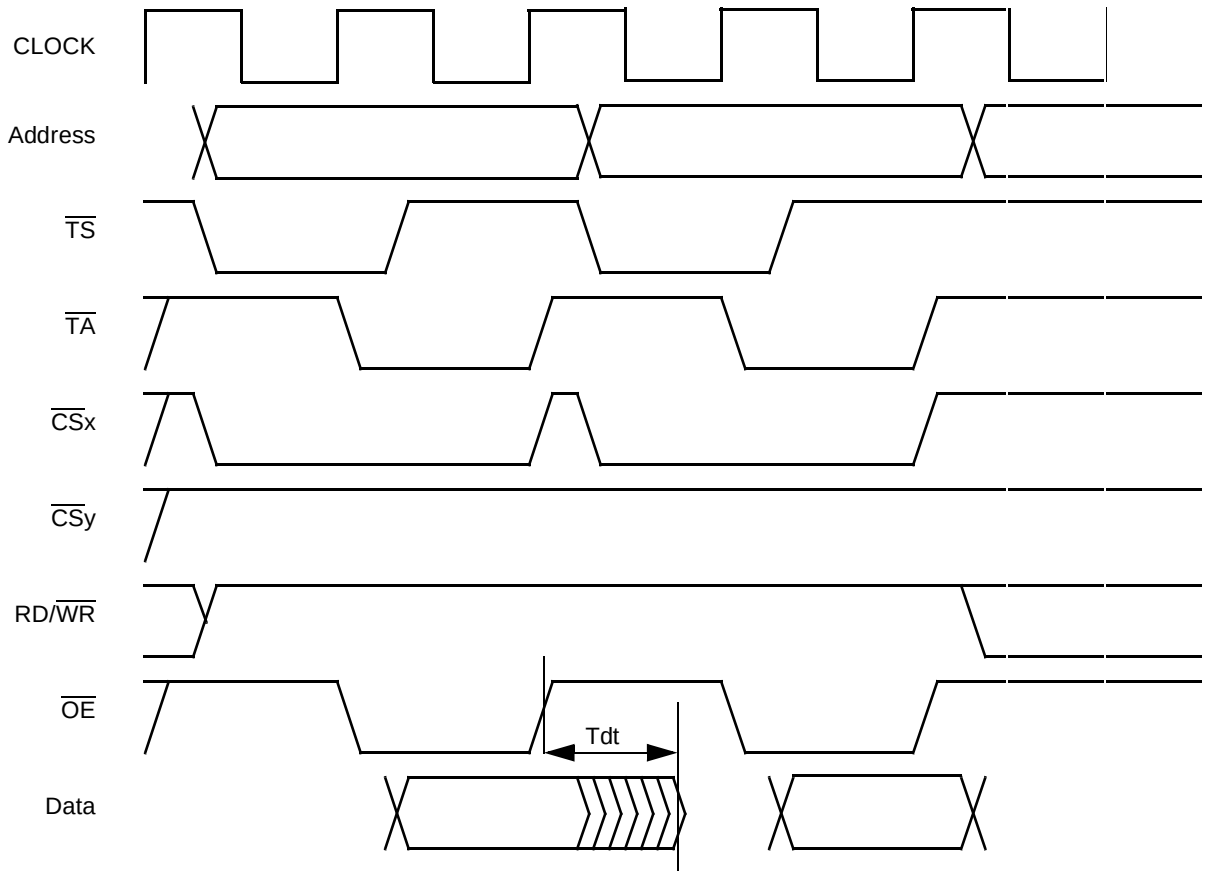


Figure 10-16 Consecutive Accesses (Read After Read From Same Bank, EHTR = 1)

10.3.5 Summary of GPCM Timing Options

Table 10-2 summarizes the different combinations of timing options.

Table 10-2 Programming Rules for Strobes Timing

TRLX	Access Type	ACS	CSNT	Address to $\overline{CS}$ Asserted	$\overline{CS}$ Negated to Add/Data Invalid	Address to $\overline{WE}/\overline{BE}$ or $\overline{OE}$ Asserted	$\overline{WE}/\overline{BE}$ Negated to Add/Data Invalid	$\overline{OE}$ Negated to Add/Data Invalid	Total Number of Cycles
0	read	00	X	0	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	read	10	X	1/4 * clock	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	read	11	X	1/2 * clock	1/4 * clock	3/4 * clock	X	1/4 * clock	2 + SCY
0	write	00	0	0	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	write	10	0	1/4 * clock	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	write	11	0	1/2 * clock	1/4 * clock	3/4 * clock	1/4 * clock	X	2 + SCY
0	write	00	1	0	1/4 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
0	write	10	1	1/4 * clock	1/2 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
0	write	11	1	1/2 * clock	1/2 * clock	3/4 * clock	1/2 * clock	X	2 + SCY
1	read	00	X	0	1/4 * clock	3/4 clock	X	1/4 * clock	2 + 2 * SCY
1	read	10	X	(1 + 1/4) * clock	1/4 * clock	(1 + 3/4) * clock	X	1/4 * clock	3 + 2 * SCY
1	read	11	X	(1 + 1/2) * clock	1/4 * clock	(1 + 3/4) * clock	X	1/4 * clock	3 + 2 * SCY
1	write	00	0	0	1/4 * clock	3/4 clock	1/4 * clock	X	2 + 2 * SCY
1	write	10	0	(1 + 1/4) * clock	1/4 * clock	(1 + 3/4) * clock	1/4 * clock	X	3 + 2 * SCY
1	write	11	0	(1 + 1/2) * clock	1/4 * clock	(1 + 3/4) clock	1/4 * clock	X	3 + 2 * SCY
1	write	00	1	0	1/4 * clock	3/4 clock	(1 + 1/2) * clock	X	3 + 2 * SCY
1	write	10	1	(1 + 1/4) * clock	(1 + 1/2) * clock	(1 + 3/4) clock	(1 + 1/2) * clock	X	4 + 2 * SCY
1	write	11	1	(1 + 1/2) * clock	(1 + 1/2) * clock	(1 + 3/4) clock	(1 + 1/2) * clock	X	4 + 2 * SCY

NOTE: Timing in this table refers to the typical timing only. Consult the electrical characteristics for exact worst-case timing values. 1/4 clock actually means 0 to 1/4 clock, 1/2 clock means 1/4 to 1/2 clock.

Additional timing rules not covered in **Table 10-2** include the following:

- If SETA = 1, an external $\overline{TA}$ signal is required to terminate the cycle.
- If TRLX = 1 and SETA = 1, the minimum cycle length = 3 clock cycles (even if SCY = 0000)
- If TRLX = 1, the number of wait states = 2 * SCY & 2 * BSCY
- If EHTR = 1, an extra (idle) clock cycle is inserted between a read cycle and a following read cycle to another region, or between a read cycle and a following write cycle to any region.
- If LBDIP = 1 (late $\overline{BDIP}$ assertion), the $\overline{BDIP}$ pin is asserted only after the number of wait states for the first beat in a burst have elapsed. See **Figure 9-13** in **SECTION 9 EXTERNAL BUS INTERFACE** as well as **9.5.4 Burst Mechanism**.

NOTE

This function can operate only when the cycle termination is internal, using the number of wait states programmed in one of the ORx registers

10.4 Global (Boot) Chip-Select Operation

Global (boot) chip-select operation allows address decoding for a boot ROM before system initialization. $\overline{CS}$[0] is the global chip-select output. Its operation differs from that of the other external chip-select outputs following a system reset. When the RCPU begins accessing memory after a system reset, $\overline{CS}$[0] is asserted for every address, unless an internal device (register) is accessed.

The global chip select provides a programmable port size at system reset using the reset BPS pins ([4:5]) of the reset configuration word, allowing a boot ROM to be located anywhere in the address space. For more information, see **7.5.2 Hard Reset Configuration Word**. The global chip select does not provide write protection and responds to all address types. $\overline{CS}$[0] operates in this way until the first write to the $\overline{CS}$[0] option register (OR0). The pin can be programmed to continue decoding a range of addresses after this write, provided the preferred address range is first loaded into base register zero. After the first write to OR0, the global chip select can only be restarted with a system reset.

The memory controller operates in boot mode until the values are modified in OR0 and BR to the ones desired.

Table 10-3 shows the initial values of the “boot bank” in the memory controller.

Table 10-3 Boot Bank Fields Values After Hard Reset

Field	Value (Binary)
PS	From reset configuration (bits 4:5 BPS)
WP	0
V	From reset configuration (bit 3 BDIS)
AM[0:16]	0 0000 0000 0000 0000
ATM[0:2]	000
CSNT	0
ACS[0:1]	00
$\overline{BI}$	1
SCY[0:3]	1111
BSCY[0:2]	011
SETA	0
TRLX	0

NOTE

If the MPC565 / MPC566 is configured (in the reset configuration word) to use the internal flash EEPROM as boot memory $\overline{CS}$[0] is not asserted.

10.5 Write and Byte Enable Signals

The GPCM determines the timing and value of the $\overline{WE}$/$\overline{BE}$ signals if allowed by the port size of the accessed bank, the transfer size of the transaction and the address accessed.

The functionality of the $\overline{WE}$/$\overline{BE}$[0:3] pins depends upon the value of the write enable/byte select (WEBS) bit in the corresponding BR register. Setting WEBS to 1 will enable these pins as $\overline{BE}$, while resetting it to zero will enable them as $\overline{WE}$. $\overline{WE}$ is asserted only during write access, while $\overline{BE}$ is asserted for both read and write accesses. The timing of the $\overline{WE}$/$\overline{BE}$ pins remains the same in either case, and is determined by the TRLX, ACS and CSNT bits.

The upper $\overline{WE}$/$\overline{BE}$ ($\overline{WE}$[0]/$\overline{BE}$[0]) indicates that the upper eight bits of the data bus (D0–D7) contains valid data during a write/read cycle. The upper-middle write byte enable ($\overline{WE}$[1]/$\overline{BE}$[1]) indicates that the upper-middle eight bits of the data bus (D8–D15) contains valid data during a write/read cycle. The lower-middle write byte enable ($\overline{WE}$[2]/$\overline{BE}$[2]) indicates that the lower-middle eight bits of the data bus (D16–D23) contains valid data during a write/read cycle. The lower write/read enable ($\overline{WE}$[3]/$\overline{BE}$[3]) indicates that the lower eight bits of the data bus contains valid data during a write cycle.

The write/byte enable lines affected in a transaction for 32-bit port (PS = 00), a 16-bit port (PS = 10) and a 8-bit port (PS = 01) are shown in **Table 10-4**. This table shows which write enables are asserted (indicated with an ‘X’) for different combinations of port size and transfer size

Table 10-4 Write Enable/Byte Enable Signals Function

Transfer Size	TSIZ		Address		32-bit Port Size				16-bit Port Size				8-bit Port Size			
			A30	A31	$\overline{WE}$[0] /$\overline{BE}$[0]	$\overline{WE}$[1] /$\overline{BE}$[1]	$\overline{WE}$[2] /$\overline{BE}$[2]	$\overline{WE}$[3] /$\overline{BE}$[3]	$\overline{WE}$[0] /$\overline{BE}$[0]	$\overline{WE}$[1] /$\overline{BE}$[1]	$\overline{WE}$[2] /$\overline{BE}$[2]	$\overline{WE}$[3] /$\overline{BE}$[3]	$\overline{WE}$[0] /$\overline{BE}$[0]	$\overline{WE}$[1] /$\overline{BE}$[1]	$\overline{WE}$[2] /$\overline{BE}$[2]	$\overline{WE}$[3] /$\overline{BE}$[3]
Byte	0	1	0	0	X				X				X			
	0	1	0	1		X				X			X			
	0	1	1	0			X		X				X			
	0	1	1	1				X		X			X			
Half-Word	1	0	0	0	X	X			X	X			X			
	1	0	1	0			X	X	X	X			X			
Word	0	0	0	0	X	X	X	X	X	X			X			

10.6 Dual Mapping of the Internal Flash EEPROM Array

Enable mapping of the internal flash EEPROM (CMF) module to an external memory region controlled by the memory controller. Only one region can be programmed to be dual-mapped. When dual mapping is enabled (DME bit is set in DMBR), an internal address matches the dual-mapped address range (as programmed in the DMBR), and the cycle type matches AT/ATM field in DMBR/DMOR registers, then the following occur:

- The internal flash memory does not respond to that address
- The memory controller takes control of the external access
- The attributes for the access are taken from one of the base and option registers of the appropriate chip select
- The chip-select region selected is determined by the "$\overline{CS}$ line select" bit field (**10.8.5 Dual-Mapping Base Register (DMBR)**).

NOTE

Dual mapping can operate only for addresses within the flash pre-allocated address (up to two Mbytes). This is achieved by programming only six bits for the base address [11:16]; The upper bits are always set as follows:

bus_addr[0:10]={0000000,isb[0:2],0}

where ISB[0:2] represents the bit field in IMMR register that determines the location of the address map of the MPC565 / MPC566.

With dual mapping, aliasing of address spaces may occur. This happens when the region is dual-mapped into a region which is also mapped into one of the four regions available in the memory controller. If code or data is written to the dual-mapped region, care must be taken to avoid overwriting this code or data by normal accesses of the chip-select region.

There is a match if:

bus_address[0:16] == {0000000,isb[0:2],0,dmbr_reg_value[1:6]}

Care must also be taken to avoid overwriting "normal" CSx data with dual-mapped code or data.

One way to avoid this situation is by disabling the chip-select region and enabling only the dual-mapped region (DME = 1, but Vx = 0, where x = selected region, 0.3). **Figure 10-17** illustrates the phenomena.

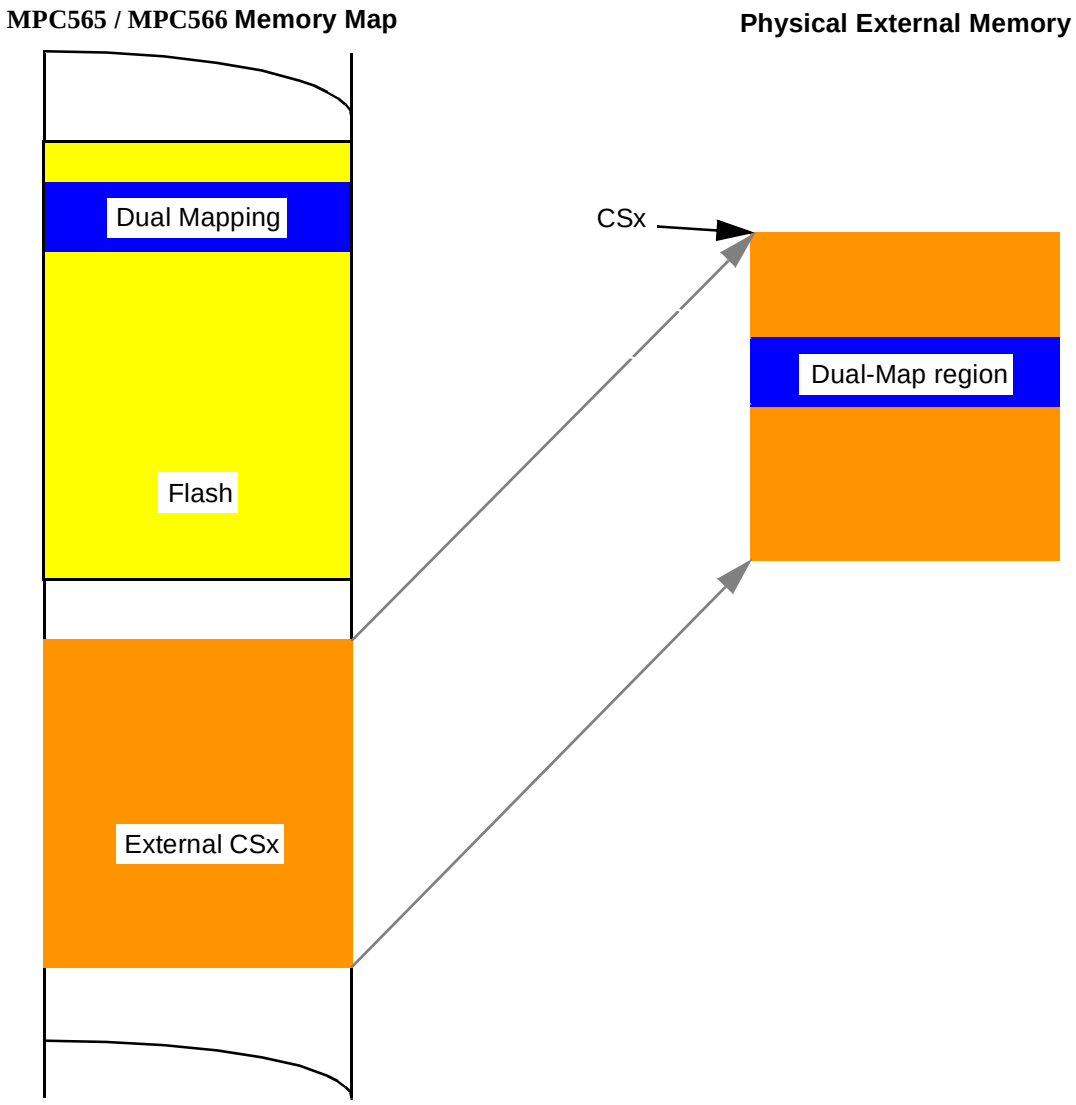


Figure 10-17 Aliasing Phenomena Illustration

The default state is to allow dual-mapping data accesses only; this means that dual mapping is possible only for data accesses on the internal bus. Also, the default state takes the lower two Mbytes of the MPC565 / MPC566 internal flash memory. Hence, caution should be taken to change the dual-mapping setup before the first data access.

NOTE

Dual mapping is *not* supported for an external master when the memory controller serves the access; In such a case, the MPC565 / MPC566 terminates the cycle by asserting $\overline{\text{TEA}}$.

10.7 Memory Controller External Master Support

The memory controller in the MPC565 / MPC566 supports accesses initiated by both internal and external bus masters to external memories. If the address of any master is mapped within the internal MPC565 / MPC566 address space, the access will be directed to the internal device, and will be ignored by the memory controller. If the address is not mapped internally, but rather mapped to one of the memory controller regions, the memory controller will provide the appropriate chip select and strobes as programmed in the corresponding region (see **6.14.1.3 External Master Control Register (EMCR)**.

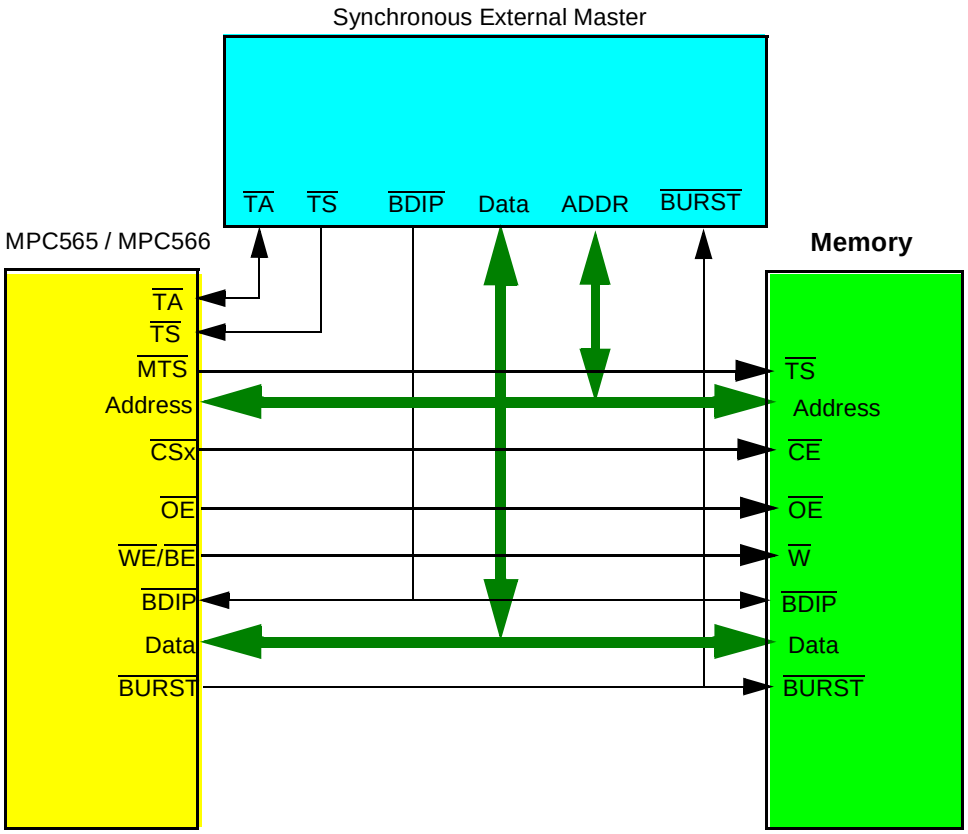
The MPC565 / MPC566 supports only synchronous external bus masters. This means that the external master works with CLKOUT and implements the MPC565 / MPC566 bus protocol to access a slave device.

A synchronous master initiates a transfer by asserting $\overline{TS}$. The ADDR[0:31] signals must be stable from the rising edge of CLKOUT during which $\overline{TS}$ is sampled, until the last $\overline{TA}$ acknowledges the transfer. Since the external master works synchronously with the MPC565 / MPC566, only setup and hold times around the rising edge of CLKOUT are important. Once the $\overline{TS}$ is detected/asserted, the memory controller compares the address with each one of its defined valid banks to find a possible match. But, since the external address space is shorter than the internal space, the actual address that is used for comparing against the memory controller regions is in the format of: {00000000, bits [8:16] of the external address}. In the case where a match is found, the controls to the memory devices are generated and the transfer acknowledge indication ($\overline{TA}$) is supplied to the master.

Since it takes two clocks for the external address to be recognized and handled by the memory controller, the $\overline{TS}$ which is generated by the external master is ahead of the corresponding $\overline{CS}$ and strobes which are asserted by the memory controller. This 2-clock delay might cause problems in some synchronous memories. To overcome this, the memory controller generates the $\overline{MTS}$ (memory transfer start) strobe which can be used in the slave's memory instead of the external master's $\overline{TS}$ signal. As seen in **Figure 10-18**, the $\overline{MTS}$ strobe is synchronized to the assertion of $\overline{CS}$ by the memory controller so that the external memory can latch the external master's address correctly. To activate this feature, the MTSC bit must be set in the SIUMCR register.

When the external master accesses the internal flash when it is disabled, then the access is terminated with transfer error acknowledge ($\overline{TEA}$ pin) asserted, and the memory controller does not support this access in any way.

When the memory controller serves an external master, the $\overline{BDIP}$ pin becomes an input pin. This pin is watched by the memory controller to detect when the burst is terminated.



NOTE: The memory controller's $\overline{\text{BDIP}}$ line is used as a burst_in_progress signal.

Figure 10-18 Synchronous External Master Configuration For GPCM-Handled Memory Devices

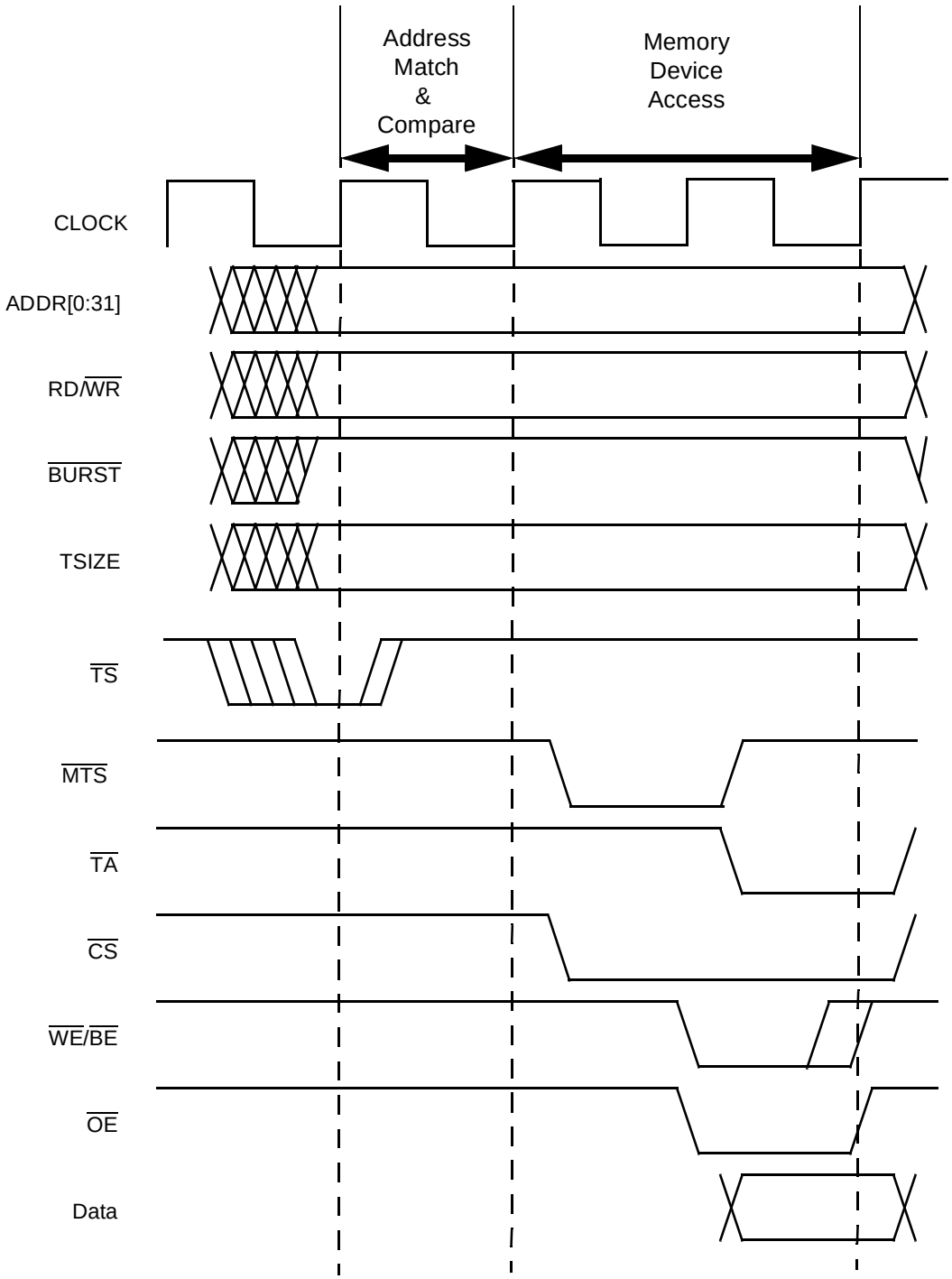


Figure 10-19 Synchronous External Master Basic Access (GPCM Controlled)

NOTE

Since the MPC565 / MPC566 has only 24 address pins, the eight most significant internal address lines are driven as 0x0000_0000, and so compared in the memory controller's regions.

10.8 Programming Model

The following registers are used to control the memory controller.

Table 10-5 Memory Controller Address Map

Address	Register
0x2F C100	Base Register Bank 0 (BR[0])
0x2F C104	Option Register Bank 0 (OR[0])
0x2F C108	Base Register Bank 1 (BR[1])
0x2F C10C	Option Register Bank 1 (OR[1])
0x2F C110	Base Register Bank 2 (BR[2])
0x2F C114	Option Register Bank 2 (OR[2])
0x2F C118	Base Register Bank 3 (BR[3])
0x2F C11C	Option Register Bank 3 (OR[3])
0x2F C120 — 0x13F	Reserved
0x2F C140	Dual-Mapping Base Register (DMBR)
0x2F C144	Dual-Mapping Option Register (DMOR)
0x2F C148 — 0x2F C174	Reserved
0x2F C178	Memory Status Register (MSTAT)

NOTE:
In all subsequent registers bit tables, if two reset values are given: the upper is for $\overline{CS}$x, x = 1, 2, 3, and the lower is dedicated to $\overline{CS}$[0].

10.8.1 General Memory Controller Programming Notes

1. In the case of an external master that accesses an internal MPC565 / MPC566 module (in slave or peripheral mode), if that slave device address also matches one of the memory controller's regions, the memory controller will not issue any $\overline{CS}$ for this access, nor will it terminate the cycle. Thus, this practice should be avoided. Be aware also that any internal slave access prevents memory controller operation.
2. If the memory controller serves an external master, then it can support accesses to 32-bit port devices only. This is because the MPC565 / MPC566 external bus interface cannot initiate extra cycles to complete an access to a smaller port-size device as it does not own the external bus.
3. When the SETA bit in the base register is set, then the timing programming for the various strobes ($\overline{CS}$, $\overline{OE}$ and $\overline{WE}$/$\overline{BE}$) may become meaningless.

10.8.2 Memory Controller Status Registers (MSTAT)

MSTAT — Memory Controller Status Register **0x2F C178**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED								WPER0	WPER1	WPER2	WPER3	RESERVED			
HRESET															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-6 MSTAT Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8:11	WPER0 – WPER3	Write protection error for bank x. This bit is asserted when a write-protect error occurs for the associated memory bank. A bus monitor (responding to $\overline{TEA}$ assertion) will, if enabled, prompt the read of this register if $\overline{TA}$ is not asserted during a write cycle. WPERx is cleared by writing one to the bit or by performing a system reset. Writing a zero has no effect on WPER.
12:15	—	Reserved

10.8.3 Memory Controller Base Registers (BR[0] – BR[3])

BR[0] – BR[3] — Memory Controller Base Registers 0 – 3 **0x2F C100**

0x2F C108, 0x2F C110, 0x2F C118

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BA															
HRESET (BR[0])															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HRESET (BR[1:3])															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BA	AT			PS		SST	WP	RE-SERVED	BL	WEBS	TBDIP	LBDIP	SETA	BI	V
HRESET (BR[0])															
U	U	U	U	ID[4:5][1]		0	0	U	U	0	U	U	U	1	ID[3][1]
HRESET (BR[1:3])															
U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:

1. The BR[0] Reset value is determined by the value on the internal data bus during reset (reset-configuration word).

Table 10-7 BR[0] – BR[3] Bit Descriptions

Bit(s)	Name	Description
0:16	BA	Base address. These bits are compared to the corresponding unmasked address signals among ADDR[0:16] to determine if a memory bank controlled by the memory controller is being accessed by an internal bus master. (The address types are also compared.) These bits are used in conjunction with the AM[0:16] bits in the OR.
17:19	AT	Address type. This field can be used to require accesses of the memory bank to be limited to a certain address space type. These bits are used in conjunction with the ATM bits in the OR. Note that the address type field uses only AT[0:2] and does not need AT[3] to define the memory type space. For a full definition of address types, refer to **9.5.7.6 Address Types**.
20:21	PS	Port size 00 = 32-bit port 01 = 8-bit port 10 = 16-bit port 11 = Reserved
22	SST	Short Setup Time – This field specifies the setup time required for this memory region. 0 = Normal setup time (like the MPC555/MPC556) 1 = Short Setup Time selected
23	WP	Write protect. An attempt to write to the range of addresses specified in a base address register that has this bit set can cause the $\overline{TEA}$ signal to be asserted by the bus-monitor logic (if enabled), causing termination of this cycle. 0 = Both read and write accesses are allowed 1 = Only read accesses are allowed. The $\overline{CS}$x signal and $\overline{TA}$ are not asserted by the memory controller on write cycles to this memory bank. WPER is set in the MSTAT register if a write to this memory bank is attempted
24	—	Reserved
25	BL	Burst Length – This field specifies the maximum number of beats that may comprise a burst access for this memory region. This filed has an effect only in the case when the burst accesses are initiated by the USIU. 0 = Burst access of up to 4 beats 1 = Burst access of up to 8 beats
26	WEBS	Write-enable/byte-select. This bit controls the functionality of the $\overline{WE}$/$\overline{BE}$ pads. 0 = The $\overline{WE}$/$\overline{BE}$ pads operate as $\overline{WE}$ 1 = The $\overline{WE}$/$\overline{BE}$ pads operate as $\overline{BE}$
27	TBDIP	Toggle-burst data in progress. TBDIP determines how long the $\overline{BDIP}$ strobe will be asserted for each data beat in the burst cycles.
28	LBDIP	Late-burst-data-in-progress (LBDIP). This bit determines the timing of the first assertion of the $\overline{BDIP}$ pin in burst cycles. **NOTE**: it is not allowed to set both LBDIP and TBDIP bits in a region's base registers; the behavior of the design in such cases is unpredictable. 0 = Normal timing for $\overline{BDIP}$ assertion (asserts one clock after negation of $\overline{TS}$ 1 = Late timing for $\overline{BDIP}$ assertion (asserts after the programmed number of wait states
29	SETA	External transfer acknowledge 0 = $\overline{TA}$ generated internally by memory controller 1 = $\overline{TA}$ generated by external logic. Note that programming the timing of $\overline{CS}$/$\overline{WE}$/$\overline{OE}$ strobes may have no meaning when this bit is set
30	BI	Burst inhibit 0 = Memory controller drives $\overline{BI}$ negated (high). The bank supports burst accesses. 1 = Memory controller drives $\overline{BI}$ asserted (low). The bank does not support burst accesses. **NOTE**: Following a system reset, the $\overline{BI}$ bit is set in OR[0].
31	V	Valid bit. When set, this bit indicates that the contents of the base-register and option-register pair are valid. The $\overline{CS}$ signal does not assert until the V-bit is set. **NOTE**: An access to a region that has no V-bit set may cause a bus monitor timeout. Note also that following a system reset, the V-bit in BR[0] reflects the value of $\overline{ID}$[3] in the reset configuration word.

10.8.4 Memory Controller Option Registers (OR[0]– OR[3])

OR[0] – OR[3] — Memory Controller Option Registers 0 – 3 **0x2F C104, 0x2F C10C, 0x2F C114, 0x2F C11C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AM*															
HRESET: (OR[1:3])															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HRESET (OR[0])															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
AM*	ATM			CSNT	ACS		EHTR	SCY				BSCY			TRLX
HRESET: (OR[1:3]):															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
HRESET (OR[0])															
0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0

*It is recommended that this field would hold values that are the power of 2 minus 1 (e.g., - 2^3 - 1 = 7 [0b111]).

Table 10-8 OR[0] – OR[3] Bit Descriptions

Bit(s)	Name	Description
0:16	AM	Address mask. This field allows masking of any corresponding bits in the associated base register. Masking the address bits independently allows external devices of different size address ranges to be used. Any clear bit masks the corresponding address bit. Any set bit causes the corresponding address bit to be used in comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at anytime. Following a system reset, the AM bits are reset in OR[0].
17:19	ATM	Address type mask. This field masks selected address type bits, allowing more than one address space type to be assigned to a chip-select. Any set bit causes the corresponding address type code bits to be used as part of the address comparison. Any cleared bit masks the corresponding address type code bit. Clear the ATM bits to ignore address type codes as part of the address comparison. Note that the address type field uses only AT[0:2] and does not need AT[3] to define the memory type space. Following a system reset, the ATM bits are reset in OR[0].
20	CSNT	Chip-select negation time. Following a system reset, the CSNT bit is reset in OR[0]. 0 = $\overline{CS}$/$\overline{WE}$ are negated normally. 1 = $\overline{CS}$/$\overline{WE}$ are negated a quarter of a clock earlier than normal Following a system reset, the CSNT bit is reset in OR[0].
21:22	ACS	Address to chip-select setup. Following a system reset, the ACS bits are reset in OR[0]. 00 = $\overline{CS}$ is asserted at the same time that the address lines are valid. 01 = Reserved 10 = $\overline{CS}$ is asserted a quarter of a clock after the address lines are valid. 11 = $\overline{CS}$ is asserted half a clock after the address lines are valid Following a system reset, the ACS bits are reset in OR[0].
23	EHTR	Extended hold time on read accesses. This bit, when asserted, inserts an idle clock cycle after a read access from the current bank and any MPC565 / MPC566 write accesses or read accesses to a different bank. 0 = Memory controller generates normal timing 1 = Memory controller generates extended hold timing
24:27	SCY	Cycle length in clocks. This four-bit value represents the number of wait states inserted in the single cycle, or in the first beat of a burst, when the GPCM handles the external memory access. Values range from 0 (0b0000) to 15 (0b1111). This is the main parameter for determining the length of the cycle. The total cycle length may vary depending on the settings of other timing attributes. The total memory access length is (2 + SCY) x Clocks. If an external $\overline{TA}$ response is selected for this memory bank (by setting the SETA bit), then the SCY field is not used. **NOTE**: Following a system reset, the SCY bits are set to 0b1111 in OR[0].

Table 10-8 OR[0] – OR[3] Bit Descriptions (Continued)

Bit(s)	Name	Description
28:30	BSCY	Burst beats length in clocks. This field determines the number of wait states inserted in all burst beats except the first, when the GPCM starts handling the external memory access and thus using SCY[0:3] as the main parameter for determining the length of that cycle. The total cycle length may vary depending on the settings of other timing attributes. The total memory access length for the beat is (1 + BSCY) x Clocks. If an external $\overline{TA}$ response has been selected for this memory bank (by setting the SETA bit) then BSCY[0:3] are not used. 000 = 0-clock-cycle (1 clock per data beat) 001 = 1-clock-cycle wait states (2 clocks per data beat) 010 = 2-clock-cycle wait states (3 clocks per data beat) 011 = 3-clock-cycle wait states (4 clocks per data beat) 1xx = Reserved
31	TRLX	Timing relaxed. This bit, when set, modifies the timing of the signals that control the memory devices during a memory access to this memory region. Relaxed timing multiplies by two the number of wait states determined by the SCY and BSCY fields. Refer to **10.3.5 Summary of GPCM Timing Options** for a full list of the effects of this bit on pins timing. 0 = Normal timing is generated by the GPCM. 1 = Relaxed timing is generated by the GPCM Following a system reset, the TRLX bit is set in OR0.

10.8.5 Dual-Mapping Base Register (DMBR)

DMBR — Dual-Mapping Base Register **0x2F C140**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	BA						RESERVED			AT			RESERVED		
HARD RESET:															
0	U	U	U	U	U	U	0	0	0	0	0	1	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED												DMCS			DME
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ID31*

*The reset value is a reset configuration word value extracted from the indicated internal data bus lines.

Table 10-9 DMBR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:6	BA	Base address. The base address field is compared (along with the address type field) to the address of the address bus to determine whether an address should be dual-mapped by one of the memory banks controlled by the memory controller. These bits are used in conjunction with the AM[11:16] bits in the OR. Bit 10: is cleared at reset. That way, the default range for the dual mapping is 2 Mbytes. **NOTE**: By setting this bit, the range becomes 4 Mbytes, which includes memory space beyond the flash EEPROM memory.
7:9	—	Reserved
10:12	AT	Address type. This field can be used to specify that accesses involving the memory bank are limited to a certain address space type. These bits are used in conjunction with the ATM bits in the OR. The default value at reset is to map data only. For a full definition of address types, refer to **9.5.7.6 Address Types**.
13:27	—	Reserved
28:30	DMCS	Dual-mapping chip select. This field determines which chip-select pin is assigned for dual mapping. 000 = $\overline{CS}$[0] 001 = $\overline{CS}$[1] 010 = $\overline{CS}$[2] 011 = $\overline{CS}$[3] 1xx = Reserved
31	DME	Dual mapping enabled. This bit indicates that the contents of the dual-mapping registers and associated base and option registers are valid and enables the dual-mapping operation. The default value at reset comes from the internal data bus that reflects the reset configuration word. See **10.6 Dual Mapping of the Internal Flash EEPROM Array** for more information. 0 = Dual mapping is not active 1 = Dual mapping is active

10.8.6 Dual-Mapping Option Register

DMOR — Dual-Mapping Option Register **0x2F C144**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	AM*						RESERVED			ATM			RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*It is recommended that this field would hold values that are the power of 2 minus 1 (e.g., - 2^3 - 1 = 7 [0b111]).

Table 10-10 DMOR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1:6	AM	Address mask. The address mask field of each option register provides for masking any of the corresponding bits in the associated base register. By masking the address bits independently, external devices of different size address ranges can be used. Any clear bit masks the corresponding address bit. Any set causes the corresponding address bit to be used in the comparison with the address pins. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. This field can be read or written at any time.
7:9	—	Reserved
10:12	ATM	Address type mask. This field can be used to mask certain address type bits, allowing more than one address space type to be assigned to a chip select. Any set bit causes the corresponding address type code bits to be used as part of the address comparison. Any cleared bit masks the corresponding address type code bit. To instruct the memory controller to ignore address type codes as part of the address comparison, clear the ATM bits. **NOTE**: Following a system reset, the ATM bits are cleared in DMOR, except the ATM2 bit. This means that only data accesses are dual mapped. Refer to the address types definition in **Table 9-8**.
13:31	—	Reserved

SECTION 11
L-BUS TO U-BUS INTERFACE (L2U)

The L-bus to U-bus interface unit (L2U) provides an interface between the load/store bus (L-bus) and the unified bus (U-bus). The L2U module includes the data memory protection unit (DMPU), which provides protection for data memory accesses.

The L2U is bidirectional. It allows load/store accesses not intended for the L-bus data RAM to go to the U-bus. It also allows code execution from the L-bus data RAM and read/write accesses from the U-bus to the L-bus.

The L2U directs bus traffic between the L-bus and the U-bus. When transactions start concurrently on both buses, the L2U interface arbitrates to select which transaction is handled. The top priority is assigned to U-bus to L-bus accesses; lower priority is assigned to the load/store accesses by the RCPU.

11.1 General Features

- Non-pipelined master and slave on U-bus
  - — Does not start two back-to-back accesses on the U-bus
  - — Supports the U-bus pipelining by starting a cycle on the U-bus when U-bus pipe depth is zero or one
  - — Does not accept back-to-back accesses from the U-bus master
- Non-pipelined master and slave on the L-bus
- Generates module selects for L-bus memory-mapped resources within a programmable, contiguous block of storage
- Programmable data memory protection unit (DMPU)
- L-bus and U-bus snoop logic for PowerPC™ reservation protocol
- L2U does not support dual mapping of L-bus or IMB3 space
- Show cycles for RCPU accesses to the SRAM (none, all, writes)
  - — Protection for SRAM accesses from the U-bus side (all accesses to the SRAM from the U-bus side are blocked once the SRAM protection bit is set)

11.2 DMPU Features

- Supports four memory regions whose base address and size can be programmed
  - — Available sizes are four Kbytes, eight Kbytes, 16 Kbytes, 32 Kbytes, 64 Kbytes, 128 Kbytes, 256 Kbytes, 512 Kbytes, one Mbyte, two Mbytes, four Mbytes, eight Mbytes, and 16 Mbytes
  - — Region must start on the specified region size boundary
  - — Overlap between regions is allowed
- Each of the four regions supports the following attributes:
  - Access protection: user or supervisor

- Guarded attribute: speculative or non-speculative
- Enable/disable option
- Read only option

- Supports a default global entry for memory space not covered by other regions:
  - — Default access protection
  - — Default guarded attribute
- Interrupt generated upon:
  - — Access violation
  - — Load from guarded region
  - — Write to read-only region
- The PowerPC MSR[DR] bit (data relocate) controls DMPU protection on/off operation
- Programming is done using PowerPC's **mtspr/mfspr** instructions to/from implementation specific special purpose registers.
- No protection for accesses to the SRAM module on the L-bus (SRAM has its own protection options)

11.3 L2U Block Diagram

Figure 11-1 shows a block diagram of the L-bus to U-bus interface.

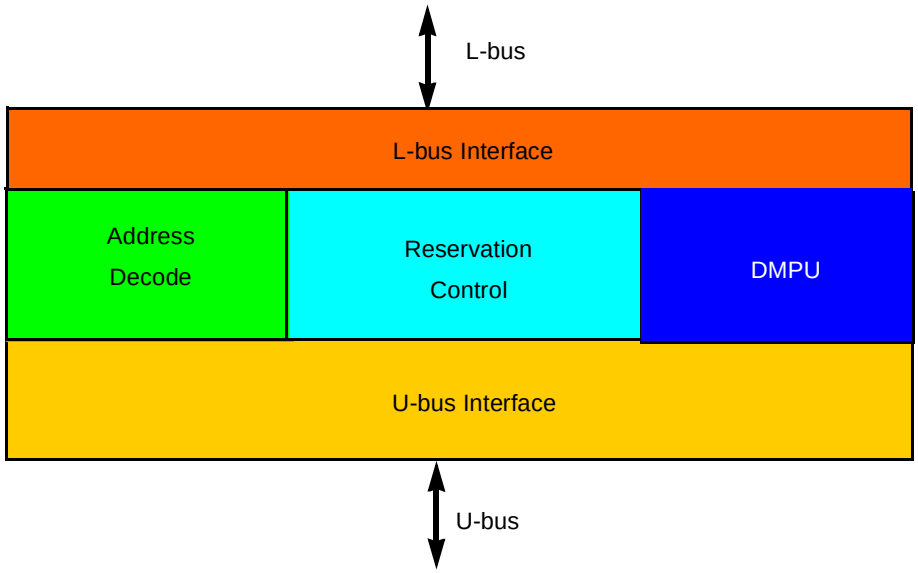


Figure 11-1 L2U Bus Interface Block Diagram

11.4 Modes Of Operation

The L2U module can operate in the following modes:

- Normal mode

- Reset operation
- Factory test mode
- Peripheral mode

11.4.1 Normal Mode

In normal mode (master or slave) the L2U module acts as a bidirectional protocol translator. In master mode the CPU is fully operational, and there is no external master access to the U-bus. Slave mode enables an external master to access any internal bus slave while the CPU is fully operational. The L2U transfers load/store accesses from the RCPU to the U-bus and the read/write accesses by the U-bus master to the L-bus.

In addition to the bus protocol translation, the L2U supports other functions such as show cycles, data memory protection and PowerPC reservation protocol.

When a load from the U-bus resource or store to the U-bus resource is issued by the RCPU, it is compared against the DMPU region access (address and attribute) comparators. If none of the access attributes are violated, the access is directed to the U-bus by the L2U module. If the DMPU detects an access violation, it informs the error status to the master initiating the cycle.

When show cycles are enabled, accesses to all of the L-bus resources by the RCPU are made visible on the U-bus side by the L2U.

The L2U is responsible for handling the effects of reservations on the L-bus and the U-bus. For the L-bus and the U-bus, the L2U detects reservation losses and updates the RCPU core with the reservation status.

11.4.2 Reset Operation

Upon soft reset assertion, the L2U goes to an idle state and all pending accesses are ignored. The L2U module control registers are not initialized on the assertion of a soft reset, keeping the system configuration unchanged.

Upon assertion of hard reset, the L2U control registers are initialized to their reset states.

While reset (hard or soft) is asserted on the U-bus, the L2U asserts the corresponding L-bus reset signals. The L2U also drives the reset configuration word from the U-bus to the L-bus upon assertion of hard reset.

11.4.3 Factory Test Mode

Factory test mode is a special mode of operation that allows access to the internal modules for testing. This mode is not intended for general use and is not supported for normal applications.

11.4.4 Peripheral Mode

In the peripheral mode of operation the RCPU is shut down and an alternative master on the external bus can perform accesses to any internal bus (U-bus and L-bus) slave.

The external master can also access the internal PowerPC special registers that are located in L2U. In order to access one of these PowerPC registers the EMCR[CONT] bit in the USIU must be cleared.

11.5 Data Memory Protection

The data memory protection unit (DMPU) in the L2U module provides access protection for the memory regions on the U-bus side from load/store accesses by the RCPU. (Only U-bus space is protected.) The DMPU does not protect PowerPC register accesses initiated by the RCPU on the L-bus. The user can assign up to four regions of access protection attributes and can assign global attributes to any space not included in the active regions. When it detects an access violation, the L2U generates an exception request to the CPU.

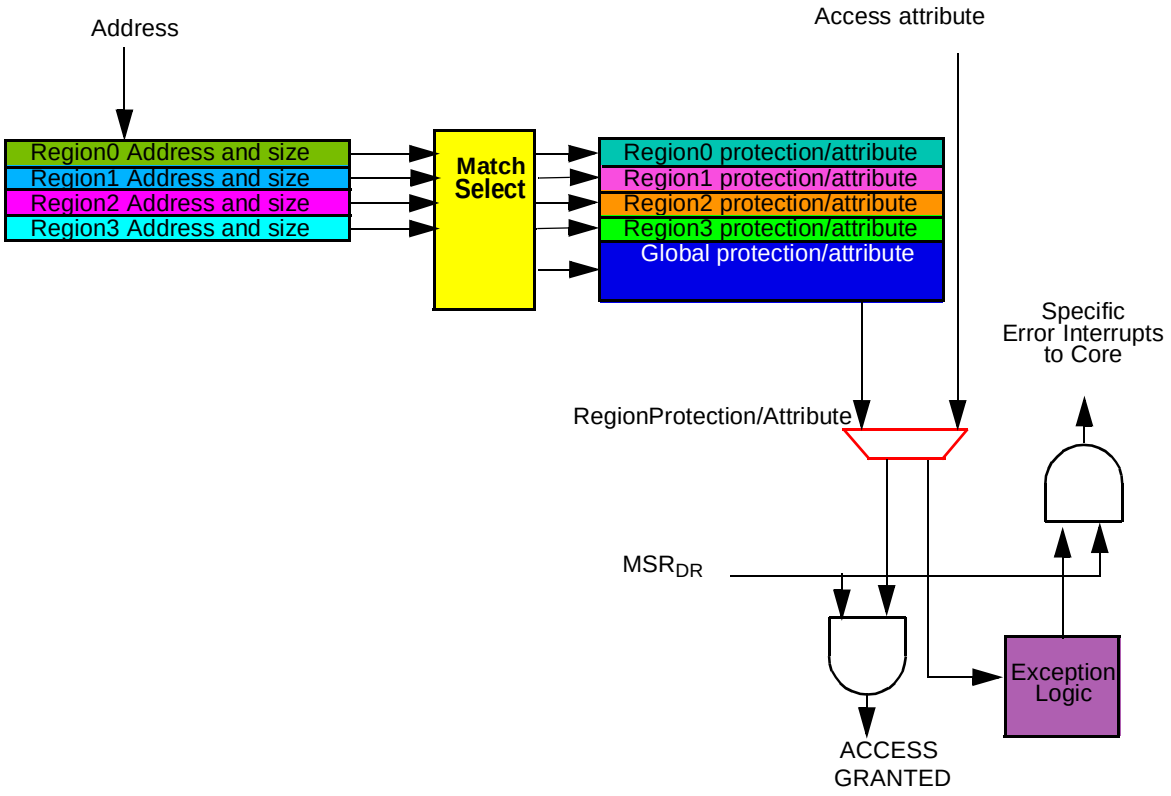


Figure 11-2 DMP Basic Functional Diagram

11.5.1 Functional Description

Data memory protection is assigned on a regional basis. Default manipulation of the DMPU is done on a global region. The DMPU has control registers which contain the following information: region protection on/off, region base address, region size, and

the region's access permissions. Each region's protection attributes can be turned on/off by configuring the enable attribute bit (ENRx) located in the global region attribute register.

During each load/store access from the RCPU core to the U-bus, the address is compared to the value in the region base address register of each enabled region. Any access that matches the specific region within its appropriate size, as defined by the region size field (RS) of the region attribute register, sets a match indication.

When more than one match indication occurs, the effective region is the region with the highest priority. Priority is determined by region number; highest priority corresponds to the lowest region number.

When no match occurs, the effective region is the global region. The global region has the lowest priority.

The region attribute register also contains the region's protection fields. The protection field (PP) of the effective region is compared to the access attributes. If the attributes match, the access is permitted. When the access is permitted, a U-bus access may be generated according to the specific attribute of the effective region.

When the access by the RCPU is not permitted, the L2U module asserts a data memory storage exception to the RCPU.

For speculative load/store accesses from the RCPU to a region marked as guarded (G bit of region attribute register is set), the L2U asks the RCPU to retry the L-bus cycle until either the access is not speculative, or it is canceled by the RCPU.

In the case of attempted accesses to a guarded region together with any other protection violation (no access), the L2U retries the access. The L2U handles this event as a data storage violation only when the access becomes non-speculative.

Note that access protection is active only when the PowerPC's MSR[DR] = 1. When MSR[DR] = 0, DMPU exceptions are disabled, all accesses are considered to be to a guarded memory area, and no speculative accesses are allowed. In this case, if the L-bus master [RCPU] initiates a non-SRAM cycle (access through the L2U) that is marked speculative, the L2U asks the RCPU to retry the L-bus cycle until either the access is not speculative, or it is canceled by the RCPU Core.

Note that the programmer must not overlap the SRAM memory space with any enabled region. Overlapping an enabled region with SRAM memory space disables the L2U data memory protection for that region.

If an enabled region overlaps with the L-bus space, the DMPU ignores all accesses to addresses within the L-bus space. If an enabled region overlaps with PowerPC register addresses, the DMPU ignores any access marked as a PowerPC access.

11.5.2 Associated Registers

The following registers are used to control the DMPU of the L2U module. All the registers are special purpose registers which are accessed via the PowerPC **mtspr**/

mfspr instructions. The registers are also accessed by an external master when EMCR[CONT] = 0. See **11.8 L2U Programming Model** for register diagrams and bit descriptions.

Table 11-1 DMPU Registers

Name	Description
L2U_RBA[0]	Region Base Address Register 0
L2U_RBA[1]	Region Base Address Register 1
L2U_RBA[2]	Region Base Address Register 2
L2U_RBA[3]	Region Base Address Register 3
L2U_RA[0]	Region Attribute Register 0
L2U_RA[1]	Region Attribute Register 1
L2U_RA[2]	Region Attribute Register 2
L2U_RA[3]	Region Attribute Register 3
L2U_GRA	Global Region Attribute

CAUTION

The appropriate DMPU registers must be programmed before the MSR[DR] bit is set. Otherwise, DMPU operation is not guaranteed.

Program the region base address in the L2U_RBAx registers to the lower boundary of the region specified by the corresponding L2U_RAx[RS] field. If the region base address does not correspond to the boundary of the block size programmed in the L2U_RAx, the DMPU snaps the region base to the lower boundary of that block. For example, if the block size is programmed to 16 Kbytes for region zero (i.e., L2U_RA0[RS] = 0x3) and the region base address is programmed to 0x1FFF(i.e., L2U_RBA0[RBA] = 0x1), then the effective base address of region zero is 0x0. See **Figure 11-3**.

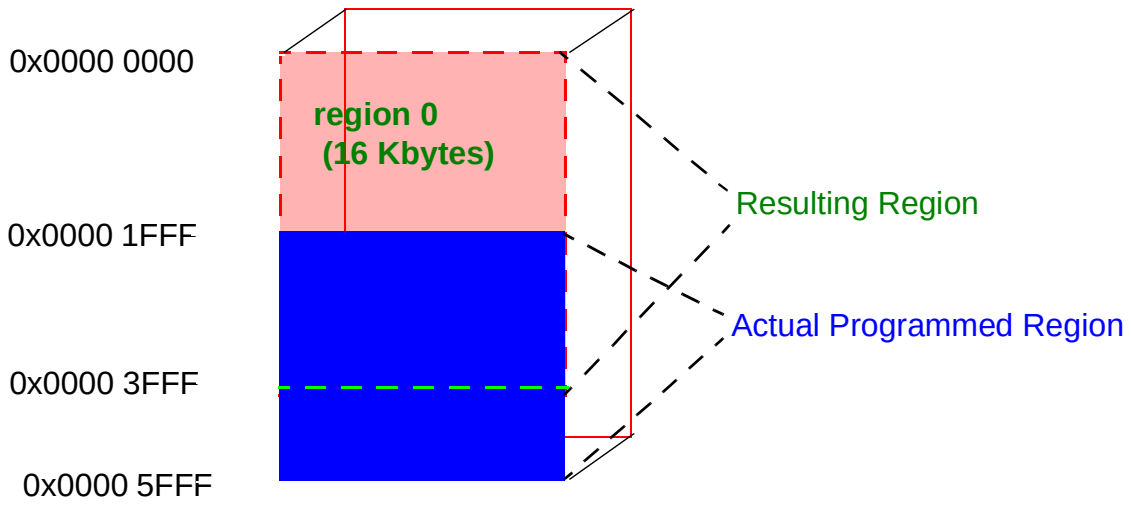


Figure 11-3 Region Base Address Example

It requires external action to program only legal region sizes. The L2U does not check whether the value is legal. If an illegal region size is programmed, the region calculation may not be successful.

11.5.3 L-bus Memory Access Violations

All L-bus slaves have their own access protection logic. For consistency, all storage access violations have the same termination result. Thus access violations for load/store accesses started by the RCPU always have the same termination from all slaves: assertion of the data storage exception. All other L-bus masters cause machine check exceptions.

11.6 Reservation Support

The RCPU storage reservation protocol supports a multi-level bus structure. For each local bus, storage reservation is handled by the local reservation logic. The protocol tries to optimize reservation cancellation such that a PowerPC processor (RCPU) is notified of storage reservation loss on a remote bus (U-bus, IMB or external bus) only when it has issued a **stwcx** cycle to that address. That is, the reservation loss indication comes as part of the **stwcx** cycle.

11.6.1 The Reservation Protocol

The reservation protocol operates under the following assumptions:

- Each processor has at most 1 reservation flag
- A **lwarx** instruction sets the reservation flag

- Another **lwarx** instruction by same processor clears the reservation flag related to a previous **lwarx** instruction and sets again the reservation flag
- A **stwcx** instruction by same processor clears the reservation flag
- A store instruction by same processor does not clear the reservation flag
- Some other processor (or other mechanism) store to an address with an existing reservation clears the reservation flag
- In case the storage reservation is lost, it is guaranteed that **stwcx** will not modify the storage

11.6.2 L2U Reservation Support

The L2U is responsible for handling the effects of reservations on the L-bus and the U-bus. For the L-bus and the U-bus, the L2U detects reservation losses.

The reservation logic in the L2U performs the following functions:

- Snoops accesses to all L-bus and U-bus slaves
- Holds one reservation (address) for the core
- Sets the reservation flag when the CPU issues a load-with-reservation request

The unit for reservation is one word. A byte or half-word store request by another master will clear the reservation flag.

A load-with-reservation request by the CPU updates the reservation address related to a previous load-with-reservation request and sets the reservation flag for the new location. A store-with-reservation request by the CPU clears the reservation flag. A store request by the CPU does not clear the flag. A store request by some other master to the reservation address clears the reservation flag.

If the storage reservation is lost, it is guaranteed that a store-with-reservation request by the CPU will not modify the storage.

The L2U does not start a store-with-reservation cycle on the U-bus if the reserved location on the U-bus has been touched by another master. The L2U drives the reservation status back to the core.

When the reserved location in the SRAM on the L-bus is touched by an alternate master, on the following clock, the L2U indicates to the CPU that the reservation has been touched. On assertion of the cancel-reservation signal, the RCPU clears the internal reservation bit. If an **stwcx** cycle has been issued at the same time, the RCPU aborts the cycle.

Storage reservation is set regardless of the termination status (address or data phase) of the **lwarx** access. Storage reservation is cleared regardless of the data phase termination status of the **stwcx** access if the address phase is terminated normally.

Storage reservation will be cleared regardless of the data phase termination status of the write requests by another master to the reserved address if the address phase of the write access is terminated normally on the destination (U-bus/L-bus) bus.

If the programmable memory map of the part is modified between a **lwarx** and a **stwcx** instruction, the reservation is not guaranteed.

11.6.3 Reserved Location (Bus) and Possible Actions

Once the CPU core reserves a memory location, the L2U module is responsible for snooping L-bus and U-bus for possible intrusion of the reserved location. Under certain circumstances, the L2U depends on the USIU or the UIMB to provide status of reservation on external bus and the IMB3 respectively.

Table 11-2 lists all reservation protocol cases supported by the L2U snooping logic.

Table 11-2 Reservation Snoop Support

Reserved Location On	Intruding Alternate Master	Action Taken on stwcx cycle
L-bus	L-master	Request to cancel the reservation.[1]
	U-master	Request to cancel the reservation.
U-bus	L-master	Block **stwcx**[2]
	U-master	Block **stwcx**
External Bus	L-master	Block **stwcx**
	U-master	Block **stwcx**
	Ext-master	Transfer Status[3]
IMB3	L-master	Block **stwcx**
	U-master	Block **stwcx**
	IMB3-master	Transfer Status

NOTES:
1. If the RCPU tries to modify (**stwcx**) that location, the L2U does not have enough time to stop the write access from completing. In this case, the L2U will drive cancel-reservation signal back to the core as soon as it comes to know that the alternate master on the U-bus has touched the reserved location.
2. If the RCPU tries to modify (**stwcx**) that location, the L2U does not start the cycle on the U-bus and it communicates to the core that the current write has been aborted by the slave with no side effects.
3. If the RCPU tries to modify (**stwcx**) that location, the L2U runs a write-cycle-with-reservation request on the U-bus. The L2U samples the status of the reservation along with the U-bus cycle termination signals and it communicates to the core if the current write has been aborted by the slave with no side effects.

11.7 L-Bus Show Cycle Support

The L2U module provides support for L-bus show cycles. L-bus show cycles are external visibility cycles that reflect activity on the L-bus that would otherwise not be visible to the external bus. L-bus show cycles are software controlled.

11.7.1 Programming Show Cycles

L-bus show cycles are disabled during reset and must be configured by writing the appropriate bits in the L2U_MCR control register. L-bus show cycles are programmed

by setting the LSHOW[0:1] bits in the L2U_MCR. The **Table 11-3** shows the configurations of the LSHOW[0:1] bits.

Table 11-3 L2U_MCR LSHOW Modes

LSHOW	Action
00	Disable L-bus show cycles
01	Show address and data of all L-bus space write cycles
10	Reserved (Disable L-bus show cycles)
11	Show address and data of all L-bus space read and write cycles

11.7.2 Performance Impact

When show cycles are enabled in the L2U module, there is a performance penalty on the L-bus. This occurs because the L2U module does not support more than one access being processed at any time. To ensure that only one access at a time can be processed, and not lose an L-bus access that would have been show cycled, the L2U module will arbitrate for the L-bus whenever it is processing any access. This L-bus arbitration will prevent any other L-bus master from starting a cycle that might turn out to be a qualifiable L-bus show cycle.

For L-bus show cycles, the minimum performance impact on the L-bus will be three clocks. This minimum impact assumes that the L-bus slave access is a 1-clock access, and the L2U module acquires immediate bus grant on the U-bus. The L2U has to wait two clocks before completing the show cycle on the U-Bus, thus using up five clocks for the complete process.

A retried access on the L-bus (no address acknowledge) that qualifies to be show cycled, will be accepted when it is actually acknowledged. This will cause a 1-clock delay before an L-bus master can retry the access on the L-bus, because the L2U module will release L-bus one clock later.

L2U asserts the internal bus request signal on the U-bus for a minimum of two clocks when starting a show cycle on the U-bus.

11.7.3 Show Cycle Protocol

The L2U module behaves as both a master and a slave on the U-bus during show cycles. The L2U starts the U-bus transfer as a a bus master and then completes the address phase and data phase of the cycle as a slave. The L2U follows U-bus protocol of in-order termination of the data phase.

The USIU can control the start of show cycles on the U-bus by asserting the no-show cycle indicator. This will cause the L2U module to release the U-bus for at least one clock before retrying the show cycle.

11.7.4 L-Bus Write Show Cycle Flow

The L2U performs the following sequence of actions for an L-bus-write show cycle.

1. Arbitrates for the L-bus to prevent any other L-bus cycles from starting
2. Latches the address and the data of the L-bus access, along with all address attributes
3. Waits for the termination of the L-bus access and latches the termination status (data error)
4. Arbitrate for the U-bus, and when granted, starts the U-bus access, asserting show cycle request on the U-bus, along with address, attributes and the write data. The L2U module provides address recognize and acknowledgment for the address phase. If the no-show cycle indicator from the U-bus is asserted, the L2U does not start the show cycle. The L2U module releases the U-bus until the no-show cycle indicator is negated and then arbitrates for the U-bus again.
5. When the L2U module has U-bus data bus grant, it drives the data phase termination handshakes on the U-bus.
6. Releases the L-bus

11.7.5 L-Bus Read Show Cycle Flow

The L2U performs the following sequence of actions for an L-bus read show cycle.

1. Arbitrates for the L-bus to prevent any other L-bus cycle from starting
2. Latches the address of the L-bus access, along with all address attributes
3. Waits for the data phase termination on the L-bus and latch the read data, and the termination status from the L-bus
4. Arbitrate for the U-bus, and when granted, starts the U-bus access, asserting the show cycle request on the U-bus, along with address attributes. The L2U module provides address recognize/acknowledgment for the address phase. If the no-show cycle indicator from the U-bus is asserted, the L2U does not start the show cycle. The L2U module releases the U-bus until the no-show cycle indicator is negated and then arbitrates for the U-bus again.
5. When the L2U module has U-bus data bus grant, it drives the read data and the data phase termination handshakes on the U-bus
6. Release the L-bus.

11.7.6 Show Cycle Support Guidelines

The following are the guidelines for L2U show cycle support:

- The L2U module provides address and data for all qualifying L-bus cycles when the appropriate mode bits are set in the L2U_MCR.
- The L2U-module-only show cycles L-bus activity that is not targeted for the U-bus or the L2U module internal registers, irrespective of the termination status of such activity.
- The L2U module does not show cycle any access to a PowerPC special purpose register.

- The L2U does not start a show cycle for an L-bus access that is retried. This decision to not start the show cycle causes a clock delay before the cycle can be retried, since the L2U module will have arbitrated away the L-bus immediately on detecting the show cycle, before the retry information is available.
- The L2U module does not show cycle any L-bus activity that is aborted.
- The L2U module backs off the U-bus if the USIU inhibits show cycle activity on the U-bus.
- The L2U does not show cycle any L-bus addresses that fall in the L-bus SRAM address space if the SRAM protection [SP] bit is set in the L2U_MCR.

Table 11-4 summarizes the L2U show cycle support.

Table 11-4 L2U Show Cycle Support Chart

Case	Destination	LB AACK	LB ABORT	Comments
1	L-bus Slave[1]	No	X	Not show cycled [Cycle will be retried one clock later][4]
2	L2U [2]	X	X	Not show cycled
3	U-bus/E-bus [3]	X	X	Not show cycled
4	L-bus slave	Yes	No	Show cycled
5	L-bus slave	Yes	Yes	Not show cycled [L-bus will be released next clock]

1. L-bus slave includes all address in the L-bus address space.
2. L2U indicates L2U registers.
3. U-bus/E-bus refers to all destinations through the L2U interface.
4. There will be a 1-clock turnaround because the L-bus retry information is not available in time to negate the L-bus arbitration.

NOTE: X indicates don't care conditions.

11.8 L2U Programming Model

The L2U control registers control the L2U bus interface and the DMPU. They are accessible via the MPC565 / MPC566 **mtspr** and **mfspr** instructions. They are also accessible by an external master when EMCR[CONT] bit is cleared. L2U control registers are accessible from both the L-bus side and the U-bus side in one clock cycle. As with all SPRs, L2U registers are accessible in supervisor mode only.

Any unimplemented bits in L2U registers return 0's on a read, and the writes to those register bits are ignored.

The **Table 11-5** shows L2U registers along with their SPR numbers and hexadecimal addresses which are used to access L2U registers during a peripheral mode access.

Table 11-5 L2U (PPC) Register Decode

Name	SPR #	SPR[5:9]	SPR[0:4]	Address for External Master Access	Access	Description
L2U_MCR	568	10001	11000	0x0000_3110	SUPR	L2U Module Configuration Register
L2U_RBA[0]	792	11000	11000	0x0000_3180	SUPR	Region Base Address Register 0
L2U_RBA[1]	793	11000	11001	0x0000_3380	SUPR	Region Base Address Register 1
L2U_RBA[2]	794	11000	11010	0x0000_3580	SUPR	Region Base Address Register 2
L2U_RBA[3]	795	11000	11011	0x0000_3780	SUPR	Region Base Address Register 3
L2U_RA[0]	824	11001	11000	0x0000_3190	SUPR	Region Attribute Register 0
L2U_RA[1]	825	11001	11001	0x0000_3390	SUPR	Region Attribute Register 1
L2U_RA[2]	826	11001	11010	0x0000_3590	SUPR	Region Attribute Register 2
L2U_RA[3]	827	11001	11011	0x0000_3790	SUPR	Region Attribute Register 3
L2U_GRA	536	10000	11000	0x0000_3100	SUPR	Global Region Attribute

For these registers a bus cycle will be performed on the L-bus and the U-bus with the address as shown in**Table 11-6**.

Table 11-6 Hex Address For SPR Cycles

A[0:17]	A[18:22]	A[23:27]	A[28:31]
0	spr0:4	spr5:9	0

11.8.1 U-bus Access

The L2U registers are accessible from the U-bus side only if it is a supervisor mode data access and the register address is correct and it is indicated on the U-bus that it is a PPC register access.

A user mode access, or an access marked as instruction, to L2U registers from the U-bus side will cause a data error on the U-bus.

11.8.2 Transaction Size

All L2U registers are defined by PowerPC architecture as being 32-bit registers. There is no PowerPC instruction to access either a half word or a byte of the special purpose register. All L2U registers are only word accessible (read and write) in peripheral mode. A half-word or byte access in peripheral mode will result in a word transaction.

11.8.3 L2U Module Configuration Register (L2U_MCR)

The L2U module configuration register (L2U_MCR) is used to control the L2U module operation.

L2U_MCR — L2U Module Configuration Register **SPR 568**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SP	LSHOW		RESERVED												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-7 L2U_MCR Bit Descriptions

Bit(s)	Name	Description
0	SP	SRAM Protection (SP) bit is used to protect the SRAM on the L-bus from U-bus accesses. This bit can be set or cleared from the L-bus side. It can be set or cleared from the U-bus side when factory test mode is enabled. When not in factory test mode, any attempt to set or clear the SP bit from the U-bus side has no affect. Once this bit is set, the L2U blocks all SRAM accesses initiated by the U-bus masters and the access is terminated with a data error on the U-bus. If L-bus show cycles are enabled, setting this bit will disable L-bus SRAM show cycles.
1:2	LSHOW	LSHOW bits are used to configure the show cycle mode for cycles accessing the L-bus slave e.g. SRAM 00 = Disable show cycles 01 = Show address and data of all L-bus space write cycles 10 = Reserved 11 = Show address and data of all L-bus space read and write cycles
3:31	—	Reserved

11.8.4 Region Base Address Registers (L2U_RBAx)

The region base address register defines the base address of a specific region protected by the data memory protection unit. There are four registers (x = 0...3), one for each supported region.

L2U_RBAx — L2U Region x Base Address Register **SPR 792 – 795**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RBA															
RESET:															
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RBA				RESERVED											
RESET:															
x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0

x = Undefined

Table 11-8 L2U_RBAx Bit Descriptions

Bit(s)	Name	Description
0:19	RBA	Region base address. The RBA field provides the base address of the region. The region base address should start on the block boundary for the corresponding block size attribute specified in the region attribute register (L2U_RAx).
20:31	—	Reserved

11.8.5 Region Attribute Registers (L2U_RAx)

Each region attribute register defines the protection attributes associated with a specific region protected by the data memory protection unit. There are four registers (x = 0...3), one for each supported region.

L2U_RAx — L2U Region X Attribute Register **SPR 824 – 827**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED								RS							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RS				PP		RESERVED			G	RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-9 L2U_RAx Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8:19	RS	Region size 0000_0000_0000 = 4 Kbytes 0000_0000_0001 = 8 Kbytes 0000_0000_0011 = 16 Kbytes 0000_0000_0111 = 32 Kbytes 0000_0000_1111 = 64 Kbytes 0000_0001_1111 = 128 Kbytes 0000_0011_1111 = 256 Kbytes 0000_0111_1111 = 512 Kbytes 0000_1111_1111 = 1 Mbyte 0001_1111_1111 = 2 Mbytes 0011_1111_1111 = 4 Mbytes 0111_1111_1111 = 8 Mbytes 1111_1111_1111 = 16 Mbytes
20:21	PP	Protection bits 00 = No supervisor access, no user access 01 = Supervisor read/write access, no user access 10 = Supervisor read/write access, user read-only access 11 = Supervisor read/write access, user read/write access
22:24	—	Reserved
25	G	Guarded attribute 0 = Not guarded from speculative accesses 1 = Guarded from speculative accesses
26:31	—	Reserved

11.8.6 Global Region Attribute Register

The global region attribute register defines the protection attributes associated with the memory region which is not protected under the four DMPU regions. This register also provides enable/disable control for the four DMPU regions.

L2U_GRA — L2U Global Region Attribute Register **SPR 536**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ENR0	ENR1	ENR2	ENR3	RESERVED											
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED				PP		RESERVED			G	RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-10 L2U_GRA Bit Descriptions

Bit(s)	Name	Description
0	ENR0	Enable attribute for region 0 0 = Region attribute is off 1 = Region attribute is on
1	ENR1	Enable attribute for region 1 0 = Region attribute is off 1 = Region attribute is on
2	ENR2	Enable attribute for region 2 0 = Region attribute is off 1 = Region attribute is on
3	ENR3	Enable attribute for region 3 0 = Region attribute is off 1 = Region attribute is on
4:19	—	Reserved
20:21	PP	Protection bits 00 = No supervisor access, no user access 01 = Supervisor read/write access, no user access 10 = Supervisor read/write access, user read-only access 11 = Supervisor read/write access, user read/write access
22:24	—	Reserved
25	G	Guarded attribute 0 = Not guarded from speculative accesses 1 = Guarded from speculative accesses
26:31	—	Reserved

SECTION 12
U-BUS TO IMB3 BUS INTERFACE (UIMB)

The U-bus to IMB3 bus interface (UIMB) Interface structure is used to connect the CPU internal unified bus (U-bus) to the intermodule bus 3 (IMB3). It controls bus communication between the U-bus and the IMB3.

The UIMB interface consists of seven submodules that control bus interface timing, address decode, data multiplexing, intrasystem communication (interrupts), and clock generation to allow communication between U-bus and the IMB3. The seven submodules are:

- U-bus interface
- IMB3 interface
- Address decoder
- Data multiplexer
- Interrupt synchronizer
- Clock control
- Scan control

12.1 Features

- Provides complete interfacing between the U-bus and the IMB3:
  - — 15 bits (32 Kbytes) of address decode on IMB3
  - — 32-bit data bus
  - — Read/write access to IMB3 module registers
  - — Interrupt synchronizer
  - — Monitoring of accesses to unimplemented addresses within UIMB interface address range
  - — Burst-inhibited accesses to the modules on IMB3
- Support of 32-bit and 16-bit BIUs for IMB3 modules
- Half and full speed operation of IMB3 bus with respect to U-bus
- Simple “slave only” U-bus interface implementation
  - — Transparent mode operation not supported
  - — Relinquish and retry not supported
- Supports scan control for modules on the IMB3 and on the U-bus

NOTE

Modules on the IMB3 bus can only be reset by $\overline{\text{SRESET}}$. Some modules may have a module reset, also.

WARNING

The user should not perform instruction fetches from modules on the IMB.

12.2 UIMB Block Diagram

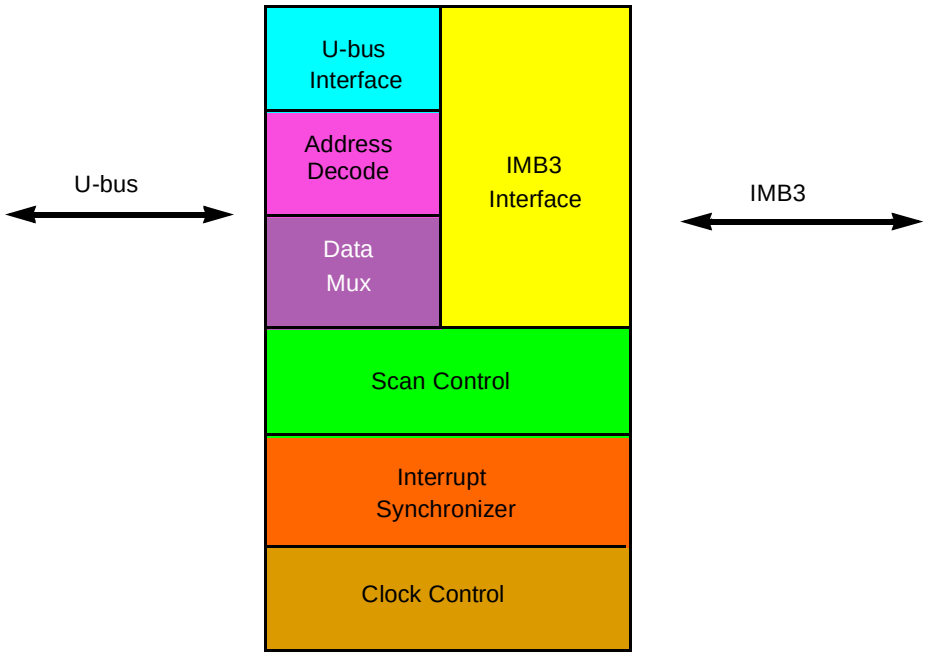


Figure 12-1 UIMB Interface Module Block Diagram

12.3 Clock Module

The clock module within the UIMB interface generates the IMB clock. The IMB clock is the main timing reference used within the IMB modules.

The IMB clock is generated based on the STOP and HSPEED bits in the UIMB module configuration register (UMCR). If the STOP bit is 1, the IMB clock is not generated. If the STOP bit is 0 and the HSPEED bit is 0, the IMB clock is generated as the inversion of the internal system clock. This is the same frequency as the CLKOUT if EBDF is 0b00 – full speed external bus. (See **Figure 12-2**.) If the HSPEED bit is 1, then the IMB clock is one-half of the internal system frequency. (See **Figure 12-3**.)

Table 12-1 STOP and HSPEED Bit Functionality

STOP	HSPEED	Functionality
0	0	IMB bus frequency is the same as U-bus frequency.
0	1	IMB bus frequency is half that of the U-bus frequency.
1	X	IMB clock is not generated.

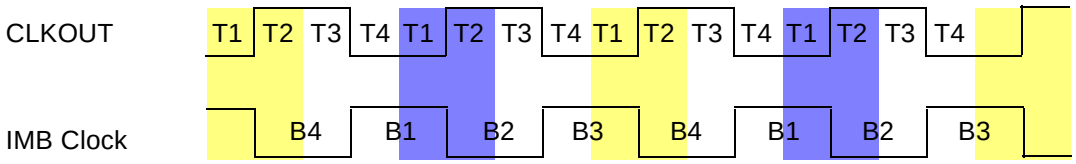


Figure 12-2 IMB Clock – Full-Speed IMB Bus

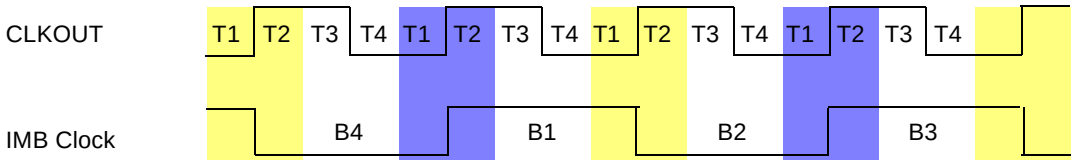


Figure 12-3 IMB Clock – Half-Speed IMB Bus

Table 12-2 shows the number of system clock cycles that the UIMB requires to perform each type of bus cycle. It is assumed in this table that the IMB3 is available to the UIMB at all times (fastest possible case).

Table 12-2 Bus Cycles and System Clock Cycles

Bus Cycle (from U-bus Transfer Start to U-bus Transfer Acknowledge)	Number of System Clock Cycles	
	Full Speed	Half Speed
Normal write	4	6
Normal read	4	6
Dynamically-sized write	6	10
Dynamically-sized read	6	10

NOTE

The UIMB interface dynamically interprets the port size of the addressed module during each bus cycle, allowing bus transfers to and from 16-bit and 32-bit IMB modules. During a bus transaction, the slave module on the IMB signals its port size (16- or 32-bit) via an internal port size signal.

12.4 Interrupt Operation

The interrupts from the modules on the IMB3 are propagated to the interrupt controller in the USIU through the UIMB interface. The UIMB interrupt synchronizer latches the Interrupts from the IMB3 and drives them onto the U-bus, where they are latched by the USIU interrupt controller.[1]

[1] The MPC565 / MPC566 includes an option for an enhanced interrupt controller. See **6.4.4 Enhanced Interrupt Controller** for operation details.

12.4.1 Interrupt Sources and Levels on IMB

The IMB3 has eight interrupt lines. There can be a maximum of 32 levels of interrupts from the modules on IMB bus. A single module can be a source for more than one interrupt. For example, the QSMCM can generate two interrupts (one for QSCI1/ QSCI2 and another for QSPI). In this case, the QSMCM has two interrupt sources. Each of these two sources can assert the interrupt on any of the 32 levels.

It is possible for multiple interrupt sources to assert the same interrupt level. To reduce the latency, it is a good practice for each interrupt source to assert an interrupt on a level on which no other interrupt source is mapped.

12.4.2 IMB Interrupt Multiplexing

The IMB has 10 lines for interrupt support. Eight lines are for interrupts and two for ILBS. These lines will transfer the 32 interrupt levels to the interrupt synchronizer. A diagram of the interrupt flow is shown in **Figure 12-4**.

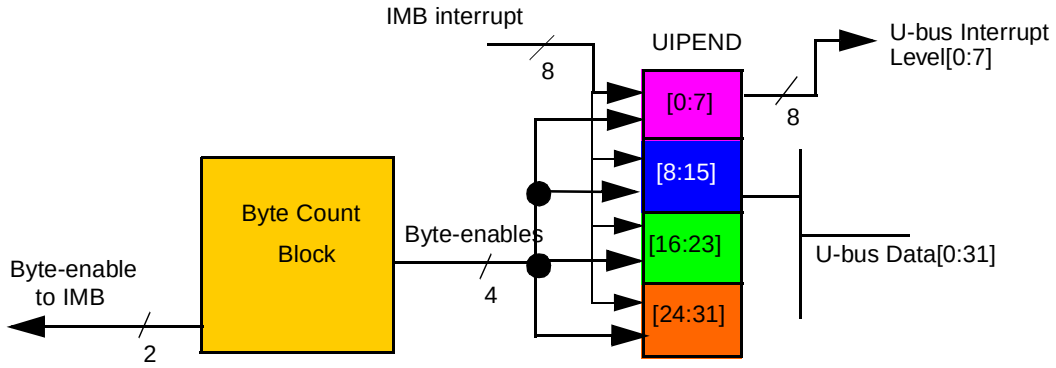


Figure 12-4 Interrupt Synchronizer Signal Flow

Latching 32 interrupt levels using eight IMB interrupt lines is accomplished with a 4:1 time-multiplexing scheme. The UIMB drives two signals (ILBS[0:1]) with a multiplexer select code that tells all interrupting modules on the IMB about which group of signals to drive during the next clock.

12.4.3 ILBS Sequencing

The IMB interface drives the ILBS signals continuously, incrementing through a code sequence (0b00, 0b01, 0b10, 0b11) once every clock The IRQMUX[0:1] bits in the IMB module configuration register select which type of multiplexing the Interrupt synchronizer will perform. The IRQMUX field can select time-multiplexing protocols for 8, 16, 24 or 32 interrupt sources. These protocols would take one, two, three or four clocks, respectively.

Table 12-4 shows ILBS sequencing. Programming IRQMUX[0:1] to 0b00 disables time multiplexing. In this case the ILBS lines remain at 0b00 at all times, as shown in **Table 12-4**. In this mode, no interrupts from IMB modules which assert on levels 8 through 31 are ever latched by the interrupt synchronizer. Time multiplexing is disabled during reset, but the reset default value enables time multiplexing as soon as reset is released if the reset default value is not 0b00.

The timing for the scheme and the values of ILBS and the interrupt levels driven onto the IMB IRQ lines are shown in **Figure 12-5**. This scheme causes a maximum latency of four clocks and an average latency of two clocks before the interrupt request can reach the interrupt synchronizer.

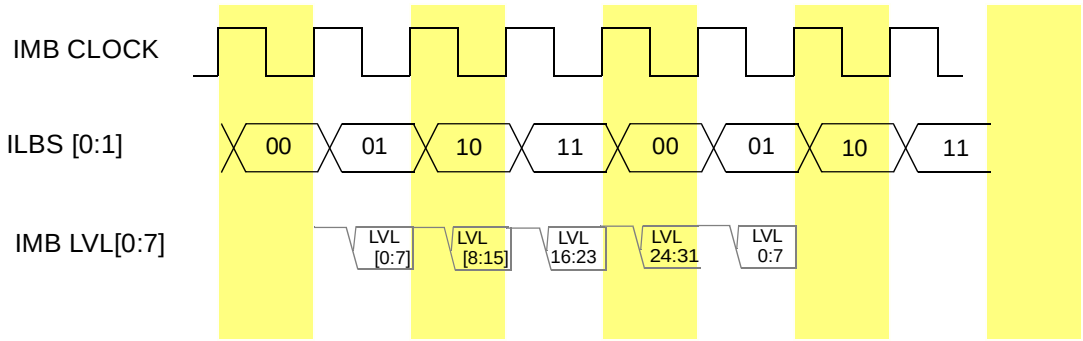


Figure 12-5 Time-Multiplexing Protocol for IRQ pins

Table 12-3 ILBS Signal functionality

ILBS[0:1]	Description
00	IMB interrupt sources mapped onto 0:7 levels will drive interrupts onto IMB IRQ[0:7]
01	IMB interrupt sources mapped onto 8:15 levels will drive interrupts onto IMB IRQ[0:7]
10	IMB interrupt sources mapped onto 16:23 levels will drive interrupts onto IMB IRQ[0:7]
11	IMB interrupt sources mapped onto 24:31 levels will drive interrupts onto IMB IRQ[0:7]

The IRQMUX bits determine how many levels of IMB interrupts are sampled. Refer to **Table 12-4**.

Table 12-4 IRQMUX Functionality

IRQMUX[0:1]	ILBS sequence	Description
00	00, 00, 00.....	Latch 0:7 IMB interrupt levels
01	00, 01, 00, 01....	Latch 0:15 IMB interrupt levels
10	00, 01, 10, 00, 01, 10,.....	Latch 0:23 IMB interrupt levels
11	00, 01, 10, 11, 00, 01, 10, 11,....	Latch 0:31 IMB interrupt levels

12.4.4 Interrupt Synchronizer

The interrupt synchronizer latches the 32 levels of interrupts from the IMB bus into a register which can be read by the CPU or other U-bus master. Since there are only eight lines for interrupts on the IMB and 32 levels of interrupts are possible, the 32 interrupt levels are multiplexed onto eight IMB interrupt lines. Apart from latching these interrupts in the register (UIPEND register), the interrupt synchronizer drives the interrupts onto the U-bus, where they are latched by the interrupt controller in the USIU.

If IMB modules drive interrupts on any of the 24 levels (levels eight through 31), they will be latched in the Interrupt pending register (UIPEND) in the UIMB. If any of the register bits 7 to 31 are set, then bit 7 will be set as well. Software must poll this register to find out which of the levels 7 to 31 are asserted.

The UIPEND register contains a status bit for each of the 32 interrupt levels. Each bit of the register is a read-only status bit, reflecting the current state of the corresponding interrupt signal. For each of the 32 interrupt levels, a corresponding bit of the UIPEND register is set.

Figure 12-4 shows how the eight interrupt lines are connected to the UIPEND register to represent 32 levels of interrupts. **Figure 12-6** shows the implementation of the interrupt synchronizer.

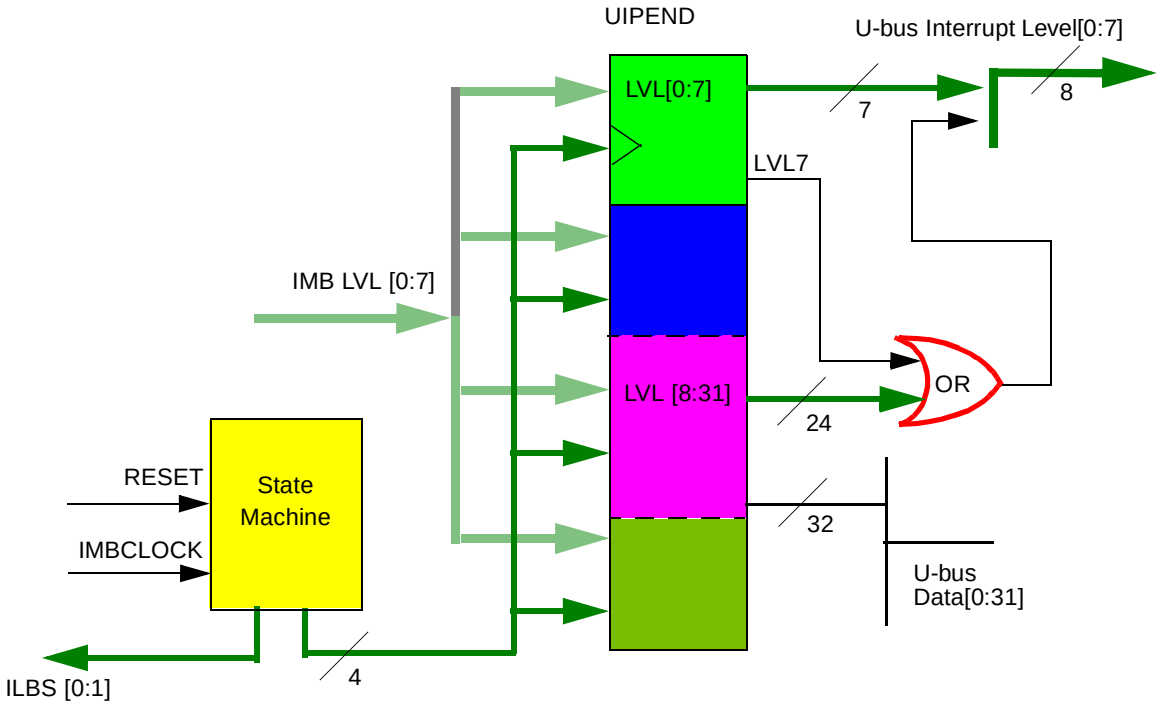


Figure 12-6 Interrupt Synchronizer Block Diagram

12.5 Programming Model

Table 12-5 lists the registers used for configuring and testing the UIMB module. The address offset shown in this table is from the start of the block reserved for UIMB registers. As shown in **Figure 1-2**, this block begins at offset 0x30 7F80 from the start of the MPC565 / MPC566 internal memory map (the last 128-byte sub-block of the UIMB interface memory map).

Table 12-5 UIMB Interface Register Map

Access	Base Address	Register
S[1]	0x30 7F80	UIMB Module Configuration Register (UMCR) See **Table 12-6** for bit descriptions.
—	0x30 7F84 — 0x30 7F8C	Reserved
S/T	0x30 7F90	UIMB Test Control Register (UTSTCREG) Reserved
—	0x30 7F94 — 0x30 7F9C	Reserved
S	0x30 7FA0	Interrupt Request Pending (UIPEND) See **12.5.3 Pending Interrupt Request Register (UIPEND)** for bit descriptions.

NOTES:
1. S = Supervisor mode only, T = Test mode only

Any word, half-word or byte access to a 32-bit location within the UIMB interface register decode block that is unimplemented (defined as reserved) causes the UIMB interface to asserting a data error exception on the U-bus.The entire 32-bit location must be defined as reserved in order for a data error exception to be asserted.

Unimplemented bits in a register return zero when read.

12.5.1 UIMB Module Configuration Register (UMCR)

The UIMB module configuration register (UMCR) is accessible in supervisor mode only.

UMCR — UIMB Module Configuration Register **0x30 7F80**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	IRQMUX		HSPEED	RESERVED											
HRESET:															
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
RESERVED															
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-6 UMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop enable. 0 = Enable system clock for IMB bus 1 = Disable IMB system clock To avoid complications at restart and data corruption, system software must stop each slave on the IMB before setting the STOP bit. Software must also ensure that all IMB interrupts have been serviced before setting this bit.
1:2	IRQMUX	Interrupt request multiplexing. These bits control the multiplexing of the 32 possible interrupt requests onto the eight IMB interrupt request lines. 00 = Disables the multiplexing scheme on the interrupt controller within this interface. What this means is that the IMB IRQ [0:7] signals are non-multiplexed, only providing 8 [0:7] interrupt request lines to the interrupt controller 01 = Enables the IMB IRQ control logic to perform a 2-to-1 multiplexing to allow transferring of 16 [0:15] interrupt sources 10 = Enables the IMB IRQ control logic to perform a 3-to-1 multiplexing to allow transferring of 24 [0:23]interrupt sources 11 = Enables the IMB IRQ control logic to perform a 4-to-1 multiplexing to allow transferring of 32 [0:31] interrupt sources
3	HSPEED	Half speed. The HSPEED bit controls the frequency at which the IMB3 runs with respect to the U-bus. This is a modify-once bit. Software can write the reset value of this bit any number of times. However, once logic 0 is written to this location, any attempt to rewrite this bit to a logic 1 will have no effect. 0 = IMB frequency is the same as that of the U-bus 1 = IMB frequency is one half that of the U-bus
4:31	—	Reserved

12.5.2 Test control register (UTSTCREG)

The UTSTCREG register is used for factory testing only.

12.5.3 Pending Interrupt Request Register (UIPEND)

The UIPEND register is a read-only status register which reflects the state of the 32 interrupt levels. The state of the IRQ[0] is shown in bit 0, the state of IRQ[1] is shown in bit 1 and so on. This register is accessible only in supervisor mode.

UIPEND — Pending Interrupt Request Register **0x30 7FA0**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LVL0	LVL1	LVL2	LVL3	LVL4	LVL5	LVL6	LVL7	LVL8	LVL9	LVL0	LVL11	LVL12	LVL13	LVL14	LVL15
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
LVL16	LVL17	LVL18	LVL19	LVL20	LVL21	LVL22	LVL23	LVL24	LVL25	LVL26	LVL27	LVL28	LVL29	LVL30	LVL31
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 12-7 UIPEND Bit Descriptions

Bit(s)	Name	Description
0:31	LVLx	Pending interrupt request level. Accessible only in supervisor mode. LVLx identifies the interrupt source as UIMB LVLx, where x is the interrupt number.

SECTION 13
QUEUED ANALOG-TO-DIGITAL CONVERTER (QADC64E)

The two queued analog-to-digital converter (QADC) modules on MPC565 / MPC566 are 10-bit, unipolar, successive approximation converters with analog multiplexers. These modules are configured so each module can access all 40 of the analog inputs to the part.

For this revision of the **QADC,** the name **QADC64E** implies an enhanced version of the **QADC64**. For simplicity, the names QADC and QADC64E may be used interchangeably throughout this document.

13.1 Features, Overview and Quick Reference Diagrams

This section gives an overview of the implementation of the two QADC64E modules on MPC565 / MPC566. It can also be used for a quick reference while programming the modules.

13.1.1 Features of the QADC64E (Each Module)

- Internal sample and hold
- Up to 40 analog input channels using QADC64E multiplexing and AMUX multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 65 total input channels using QADC64E, AMUX, and external multiplexing
- Supports AMUX without loss of Port A and Port B functionality
- Programmable input sample time for various source impedances
- Typical conversion time of less than 5 µs (> 200 kSamples/s)
- Modulus prescaler can divide the system clock for the converter by two to 128
- Two conversion command queues with a total of 64 entries
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Queue pointers indicate current location for each queue
- Automated queue modes initiated by:
  - — External edge trigger and gated trigger
  - — Periodic/interval timer, within QADC64E module [queue 1 and 2]
  - — Software command
- Single-scan or continuous-scan of queues

- 64 result registers in each QADC64E
- Output data readable in three formats:
  - — Right-justified unsigned
  - — Left-justified signed
  - — Left-justified unsigned
- Unused analog channels can be used as digital ports
- Multi-module operation allows shared channels (in all queues) and shared clock for synchronization
- Alternate reference input, with control in CCW

13.1.2 QADC64E Block Diagrams

Figure 13-1 displays the major components of the QADC64E modules on the MPC565 / MPC566 including the pads, AMUX and both QADCs.

Figure 13-2 shows the sub-module arrangement of the QADC64E.

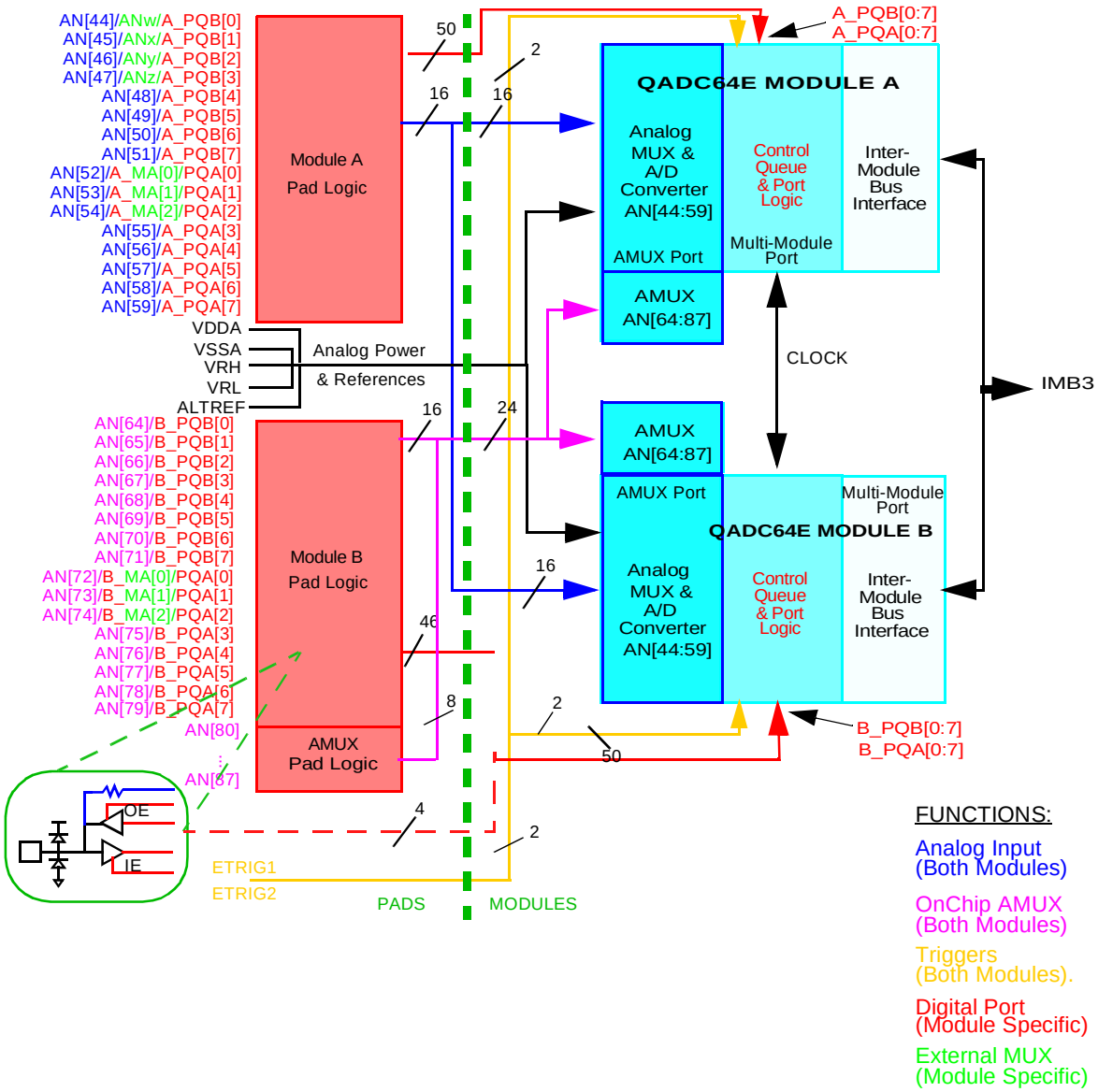


Figure 13-1 Dual QADC64Es on MPC565 / MPC566

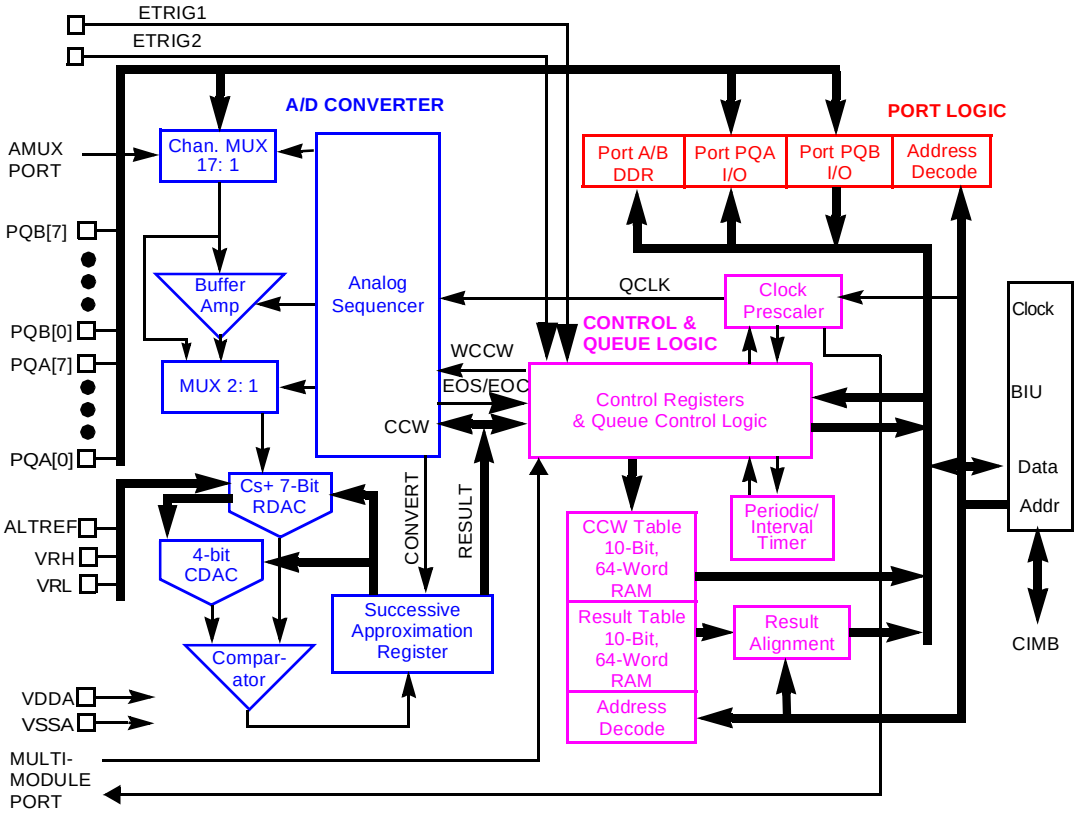


Figure 13-2 Block Diagram of QADC64E

The analog section includes input pins, an analog multiplexer, and the sample and hold circuits. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array and a high-gain comparator.

The digital control section contains queue control logic to sequence the conversion process and interrupt generation logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result table RAM.

The bus interface unit (BIU) allows the QADC64E to operate with the applications software through the IMB environment.

13.1.3 Memory Map

The QADC64E occupies one Kbyte, or 512 16-bit entries, of address space. Ten 16-bit registers are control, port, and status registers, 64 16-bit entries are the CCW table, and 64 16-bit entries are the result table, and occupy 192 16-bit address locations because the result data is readable in three data alignment formats.

Each QADC on MPC565 / MPC566 has its own memory location. **Table 13-1** shows the memory map for QADC64E module A. Module B has the same offset scheme starting at 0x30 4C00 (refer to **Table 13-2**).

QADC64E_A occupies 0x30 4800 to 0x30 4BFF

QADC64E_B occupies 0x30 4C00 to 0x30 4FFF

Table 13-1 QADC64E_A Address Map

Address	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15	Register
0x30 4800	STOP	FRZ	--						SUPV	MSTR	EXT CLK	--					Module Config.*
0x30 4802	TEST MODE				---												Test
0x30 4804	IRL1					IRL2					--						Interrupt
0x30 4806	PORTQA								PORTQB								Port Data
0x30 4808	DDRQA								DDRQB								Port Direction
0x30 480A	EMUX	---		TRIG	--					QCLK PRESCALER							Control 0
0x30 480C	CIE1	PIE1	SSE1	MQ1					---								Control 1
0x30 480E	CIE2	PIE2	SSE2	MQ2					RESU.	BQ2							Control 2
0x30 4810	CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP						Status 0
0x30 4812	--		CWPQ1						---		CWPQ2						Status 1
0x30 4814-0x30 49FF	--14 Words Reserved --																Reserved
0x30 4A00-0x30 4A7F	--						PAUS	REF	IST	CHAN							CCWs
0x30 4A80-0x30 4AFF	0000 00						UNSIGNED RIGHT JUSTIFIED										Results
0x30 4B00-0x30 4B7F	SIGN	SIGNED LEFT JUSTIFIED									00 0000						Results
0x30 4B80 0x30 4BFF	UNSIGNED LEFT JUSTIFIED										00 0000						Results

***Registers in RED are accessible only as supervisor data space**

Table 13-2 QADC64E_B Address Map

Address	MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15	Register
0x30 4C00	STOP	FRZ	--						SUPV	MSTR	EXT CLK	--					Module Config.*
0x30 4C02	TEST MODE				---												Test
0x30 4C04	IRL1					IRL2					--						Interrupt
0x30 4C06	PORTQA								PORTQB								Port Data
0x30 4C08	DDRQA								DDRQB								Port Direction
0x30 4C0A	EMUX	---		TRIG	--					QCLK PRESCALER							Control 0
0x30 4C0C	CIE1	PIE1	SSE1	MQ1					---								Control 1
0x30 4C0E	CIE2	PIE2	SSE2	MQ2					RESU.	BQ2							Control 2
0x30 4C10	CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP						Status 0
0x30 4C12	--		CWPQ1						---		CWPQ2						Status 1
0x30 4C14-0x30 4DFF	--14 Words Reserved --																Reserved
0x30 4E00-0x30 4E7F	--						PAUS	REF	IST	CHAN							CCWs
0x30 4E80-0x30 4EFF	0000 00						UNSIGNED RIGHT JUSTIFIED										Results
0x30 4F00-0x30 4F7F	SIGN	SIGNED LEFT JUSTIFIED									00 0000						Results
0x30 4F80 0x30 4FFF	UNSIGNED LEFT JUSTIFIED										00 0000						Results

***Registers in RED are accessible only as supervisor data space**

Access to supervisor-only data space is permitted only when the bus master is operating in supervisor access mode. Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user data space addresses. See **13.2.1.3 Supervisor/Unrestricted Address Space**.

13.1.4 Using the Queue and Result Word Table

The heart of the QADC is its conversion command word (CCW) queues. This is where the module is programmed to convert a particular channel according to a particular requirement. The queues are created by writing CCWs into the CCW table in the register memory. The queues are controlled by the three control registers, and their status can be read from status registers. As conversions are completed the digital value is written into the result word table. **Figure 13-3** shows the CCW queue and the result word table.

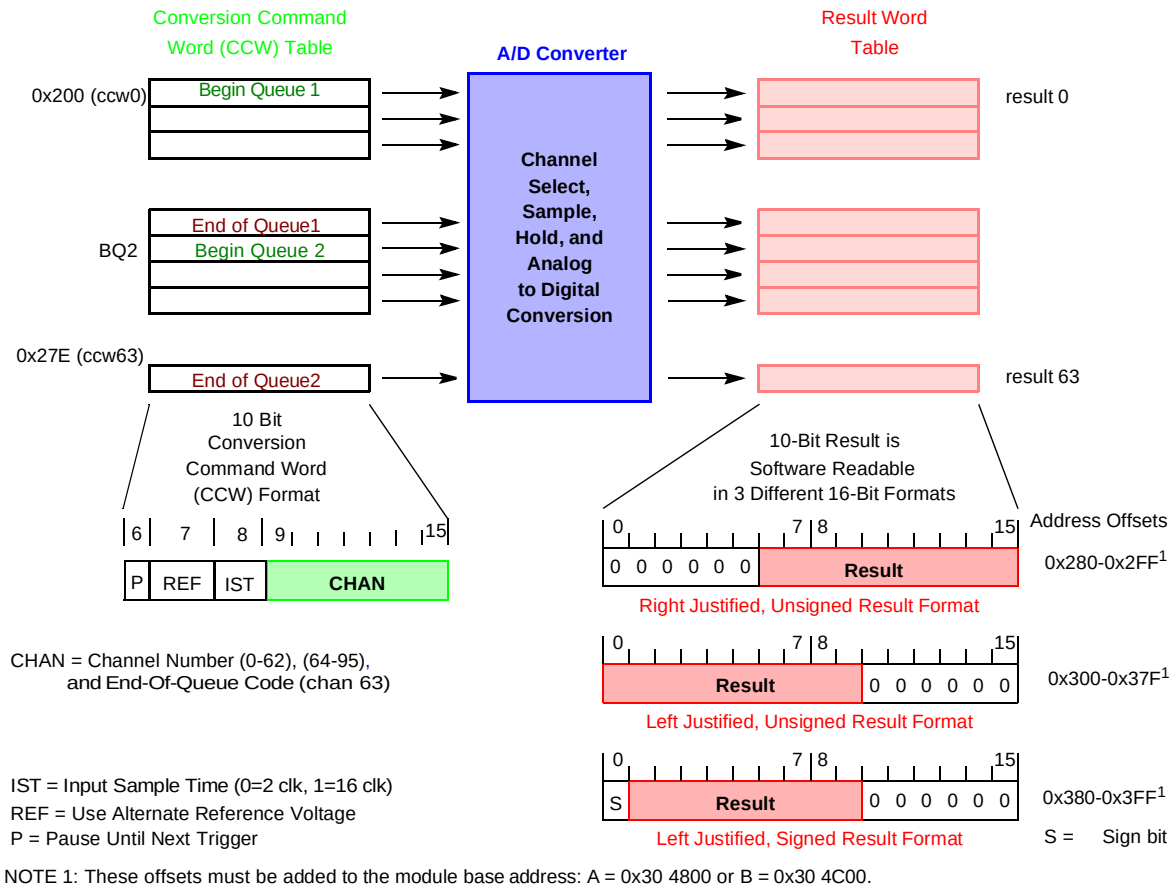


Figure 13-3 CCW Queue and Result Table Block Diagram

13.1.5 External Multiplexing

The QADC can use from one to four 8-input external multiplexer chips to expand the number of analog signals that may be converted. The externally multiplexed channels are automatically selected from the channel field of the conversion command word (CCW) table. The software selects the external multiplexed mode by setting the MUX bit in control register 0 (QACR0).

Figure 13-4 shows the maximum configuration of four external multiplexer chips connected to the QADC. The QADC provides three multiplexed address signals — MA[0], MA[1], and MA[2], to select one of eight inputs. These inputs are connected to all four external multiplexer chips.

The analog output of the four multiplex chips are each connected to four separate QADC inputs — AN_W, AN_X, AN_Y, and AN_Z. The QADC converts the proper input channel (AN_W, AN_X, AN_Y, and AN_Z) by interpreting the CCW channel number.

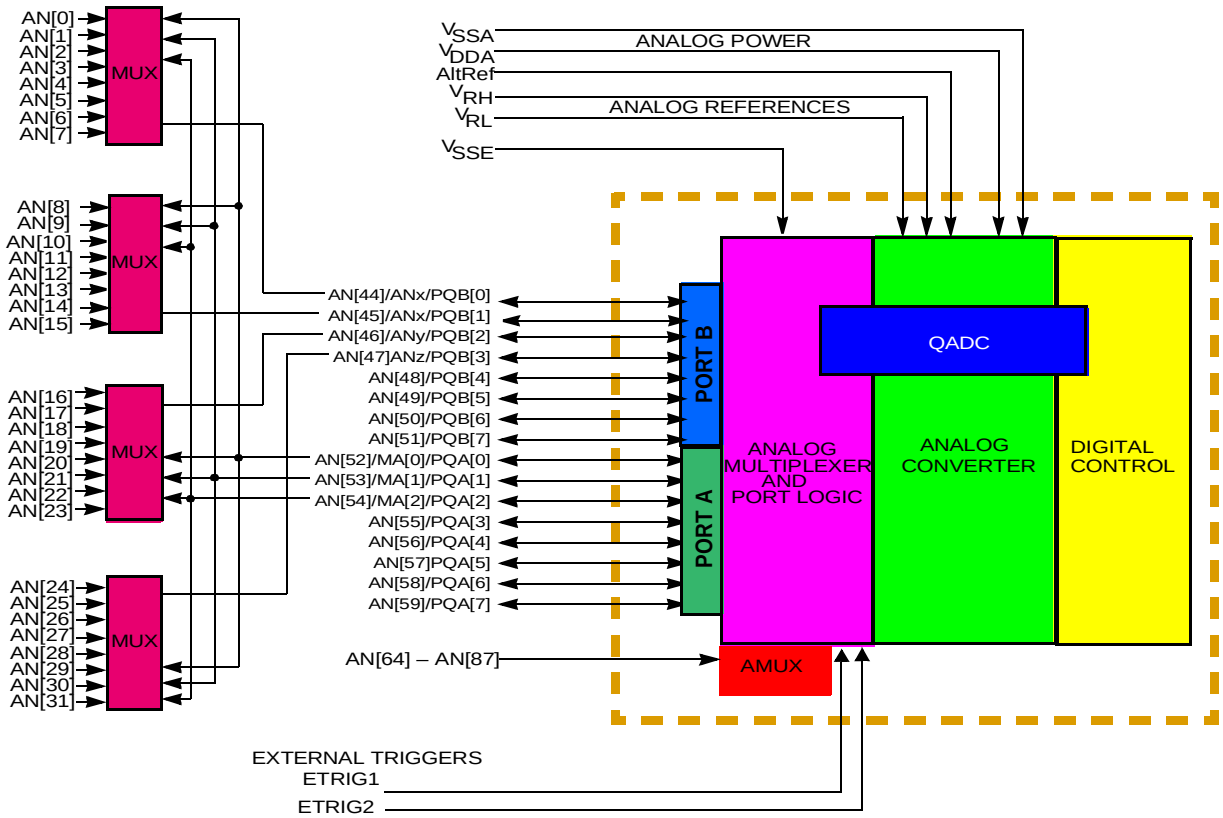


Figure 13-4 Example of External Multiplexing

In the external multiplexed mode, four of the port A pins are redefined to each represent eight input channels. Refer to **Table 13-3**.

Table 13-3 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw (AN[44])	Channels from 0 to 7
ANx (AN[45])	Channels from 8 to 15
ANy (AN[46])	Channels from 16 to 23
ANz (AN[47])	Channels from 24 to 31

Table 13-4 shows the total number of analog input channels supported with zero to four external multiplexer chips using one QADC module.

Table 13-4 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels				
No External MUX Chips	**One External MUX Chip**	**Two External MUX Chips**	**Three External MUX Chips**	**Four External MUX Chips**
40	36 + 8 = 44	35 + 16 = 51	34 + 24 = 58	33 + 32 = 65

NOTE: QADC64E External MUX Users

If either QADC64E_A or QADC64E_B is in external MUX (EMUX) mode, AN[44:47] (analog inputs) and AN[52:54] and/or AN[72:74] should not be programmed into queues as they will be EMUX input and address pins.

If both QADC64E_A or QADC64E_B are in external MUX mode, they independently control MUX addresses, but SHARE INPUT PINS so a total of four external multiplexers can be used, but each should be assigned to only A or B QADCs. Each QADC module queue can only control external addresses that are controlled by its own MA[2:0] multiplexer address pins.

13.2 Programming the QADC64E Registers

The QADC64E has three global registers for configuring module operation: the module configuration register (**13.2.1 QADC64E Module Configuration Register**), the interrupt register (**13.2.2 QADC64E Interrupt Register**), and a test register (QADCTEST). The global registers are always defined to be in supervisor-only data space. Refer to **Table 13-1** for the QADC64E_A address map and **Table 13-2** for the QADC64E_B address map. See **13.2.1.3 Supervisor/Unrestricted Address Space** for access modes of these registers.

The remaining five registers in the control register block control the operation of the queuing mechanism, and provide a means of monitoring the operation of the QADC64E.

- Control register 0 (QACR0) contains hardware configuration information (**13.2.5 Control Register 0**)
- Control register 1 (QACR1) is associated with queue 1 (**13.2.6 Control Register 1**)
- Control register 2 (QACR2) is associated with queue 2 (**13.2.7 Control Register 2**)
- Status registers (QASR0 and QASR1) provide visibility on the status of each queue and the particular conversion that is in progress (**13.2.8 Status Registers**)

The CCW table follows the register block in the address map. There are 64 entries to hold the desired analog conversion sequences. Each CCW is a 16-bit entry, with ten implemented bits in four fields.

The final block of address space belongs to the result word table, which appears in three places in the memory map. Each result word table location holds one 10-bit conversion value.

13.2.1 QADC64E Module Configuration Register

The QADCMCR contains fields and bits that control freeze and stop modes and determine the privilege level required to access most registers

QADCMCR — Module Configuration Register **0x30 4800**
0x30 4C00

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	RESERVED						SUPV	MSTR	EXT CLK	RESERVED				
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 13-5 QADCMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop Enable — Refer to **13.2.1.1 Low Power Stop Mode** for more information. 0 = Disable stop mode 1 = Enable stop mode
1	FRZ	Freeze Enable — Refer to **13.2.1.2 Freeze Mode** for more information. 0 = Ignores the IMB internal FREEZE signal 1 = Finish any conversion in progress, then freeze
2:7	—	Reserved
8	SUPV	Supervisor/Unrestricted Data Space — Refer to **13.2.1.3 Supervisor/Unrestricted Address Space** and **Table 13-6** for more information. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted. 1 = All QADC64E registers and CCW/result tables are designated as supervisor-only data space.
9	MSTR	Master/Slave Operation — Refer to **13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks** for more information. 0 = Module is a slave, and QADCMCR EXTCLK should be used to select the conversion clock source. 1 = Module is the master, and QADCMCR EXTCLK should be left clear.
10	EXTCLK	External Clock Select — 0 = Internal conversion clock derived from IMB3 clock will be used for the converter and QADC Periodic/Interval Timer. 1 = Internal conversion clock derived from a master QADC module (see MSTR, bit 9). Refer to **13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks**.
11:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in QADC64E implementations that use hardware interrupt arbitration. These bits are not used on the MPC565 / MPC566.

13.2.1.1 Low Power Stop Mode

When the STOP bit in the QADCMCR is set, the QADC64E clock (QCLK) which clocks the A/D converter, is disabled and the analog circuitry is powered down . This results in a static, low power consumption, idle condition. The stop mode aborts any conver-

sion sequence in progress. Because the bias currents to the analog circuits are turned off in stop mode, the QADC64E requires some recovery time (T_{SR} in **APPENDIX E ELECTRICAL CHARACTERISTICS**) to stabilize the analog circuits after the stop enable bit is cleared.

In stop mode:

- BIU state machine and logic do not shut down
- The RAM is not reset and is not accessible
- The module configuration register (QADCMCR), the interrupt register (QADCINT), and the test register (QADCTEST) are fully accessible and are not reset
- The data direction register (DDRQA), port data register (PORTQA/PORTQB), and control register 0 (QACR0) are not reset and are read-only accessible
- Control register 1 (QACR1), control register 2 (QACR2), and the status registers (QASR0 and QASR1) are reset and are read-only accessible
- In addition, the periodic/interval timer is held in reset during stop mode

If the STOP bit is clear, stop mode is disabled.

13.2.1.2 Freeze Mode

Freeze mode occurs when the background debug mode is enabled in the device integration module and a breakpoint is encountered. This is indicated by the assertion of the internal FREEZE line on the IMB. The FRZ bit in the QADCMCR determines whether or not the QADC64E responds to an IMB internal FREEZE signal assertion. Freeze is very useful when debugging an application.

When the FRZ bit is set, the QADC64E finishes any conversion in progress then freezes. Depending on when the FRZ bit is asserted, there are three possible queue “freeze” scenarios.

- When a queue is not executing, the QADC64E freezes immediately
- When a queue is executing, the QADC64E completes the conversion in progress and then freezes
- If, during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC64E freezes immediately

When the QADC64E enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock, QCLK, is held in reset and the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not captured. The BIU remains active to allow IMB access to all QADC64E registers and RAM. Although the QADC64E saves a pointer to the next CCW in the current queue, the software can force the QADC64E to execute a different CCW by writing new queue operating modes for normal operation. The QADC64E looks at the queue operating

modes, the current queue pointer, and any pending trigger events to decide which CCW to execute when exiting freeze.

If the FRZ bit is clear, the internal FREEZE signal is ignored.

13.2.1.3 Supervisor/Unrestricted Address Space

The QADC64E memory map is divided into two segments: supervisor-only data space and assignable data space. Access to supervisor-only data space is permitted only when the software is operating in supervisor access mode. Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user data space accesses. The SUPV bit in the QADCMCR designates the assignable space as supervisor or unrestricted.

The following information applies to accesses to address space located within the module's 16-bit boundaries and where the module response causes the bus error. See **Table 13-6** for more information.

Table 13-6 QADC64E Bus Error Response

S/U Mode	SUPV Bit	Supervisor-Only Register	Supervisor/ Unrestricted Register	Reserved/ Unimplemented Register
U	0	QADC64E bus error	Valid access	QADC64E bus error
U	1	Master bus error	Master bus error	Master bus error
S	0	Valid access	Valid access	QADC64E bus error
S	1	Valid access	Valid access	QADC64E bus error

NOTES:
S/U = Supervisor/Unrestricted
QADC64E bus error = Caused by QADC64E
Master bus error = Caused by bus master
No bus error = Writes have no effect, reads will return zeroes.
Access to QADCTEST register will act as a reserved/unimplemented register unless in factory test mode

- Attempts to read a supervisor-only data space when not in the supervisor access mode and SUPV = 1, causes the bus master to assert a bus error condition. No data is returned. If SUPV = 0, the QADC64E asserts a bus error condition and no data is returned,
- Attempts to write supervisor-only when not in the supervisor access mode and SUPV = 1, causes the bus master to assert a bus error condition. No data is written. If SUPV = 0, the QADC64E asserts a bus error condition and the register is not written.
- Attempts to read unimplemented data space in the unrestricted access mode and SUPV = 1, causes the bus master to assert a bus error condition and no data is returned. In all other attempts to read unimplemented data space, the QADC64E causes a bus error condition and no data is returned.

- Attempts to write unimplemented data space in the unrestricted access mode and SUPV= 1, causes the bus master to assert a bus error condition and no data is written. In all other attempts to write unimplemented data space, the QADC64E causes a bus error condition and no data is written.
- Attempts to read assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes the bus master to assert a bus error condition and no data is returned.
- Attempts to write assignable data space in the unrestricted access mode when the space is programmed as supervisor space causes the bus master to assert a bus error condition and the register is not written.

The bus master indicates the supervisor and user space access with the function code bits (FC[2:0]) on the IMB. For privilege violations, refer to the MPC565 / MPC566 reference manual to determine the consequence of a bus error cycle termination.

The supervisor-only data space segment contains the QADC64E global registers, which include the QADCMCR, the QADCTEST, and the QADCINT. The supervisor/ unrestricted space designation for the CCW table, the result word table, and the remaining QADC64E registers are programmable.

13.2.1.4 Master/Slave Operation and Multi-Module Synchronous Clocks

Master/slave mode operation is controlled by two bits in the module configuration register. The first is QADCMCR MSTR field, which, when set makes the module the master and causes that module's conversion clock to be output on the slave clock input/output. If the QADCMCR MSTR bit is cleared, then that module is set up as a slave. When set as a slave, the module's QADCMCR EXTCLK may be set, causing the module to use the slave clock signal input as the conversion clock.

If a module is configured as a master, then the QADCMCR EXTCLK bit should be set low.

If a module is configured as a slave, then the QADCMCR EXTCCLK bit may be left clear, in which case the module will operate off of the internal conversion clock, or it may be set, in which case the module will operate off of the slave clock input/output.

13.2.2 QADC64E Interrupt Register

QADCINT specifies the priority level of QADC64E interrupt requests and the vector provided. The interrupt level for queue 1 and queue 2 may be different. The interrupt register is read/write accessible in supervisor data space only. The implemented interrupt register fields can be read and written, reserved bits read zero and writes have no effect. They are typically written once when the software initializes the QADC64E, and not changed afterwards.

QADCINT — MPC565 / MPC566 Interrupt Register

0x30 4804
0x30 4C04

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRL1					IRL2					RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-7 QADCMCR Bit Descriptions

Bit(s)	Name	Description
0:4	IRL1	Queue 1 Interrupt Request Level — The IRL1 field establishes the queue 1 interrupt request level. The 00000 state provides a level 0 interrupt while 11111 provides a level 31 interrupt. All interrupts are presented on the IMB. Interrupt level priority software determines which level has the highest priority request.
5:9	IRL2	Queue 2 Interrupt Request Level — The IRL2 field establishes the queue 2 interrupt request level. The 00000 state provides a level 0 interrupt while 11111 provides a level 31 interrupt. All interrupts are presented on the IMB. Interrupt level priority software determines which level has the highest priority request.
10:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in QADC64E implementations that use hardware interrupt arbitration. These bits are not used on the MPC565 / MPC566.

The QADC64E conditionally generates interrupts to the bus master via the IMB IRQ signals. When the QADC64E sets a status bit assigned to generate an interrupt, the QADC64E drives the IRQ bus. The value driven onto IRQ[7:0] represents the interrupt level assigned to the interrupt source. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 13-5** displays the interrupt levels on IRQ with ILBS.

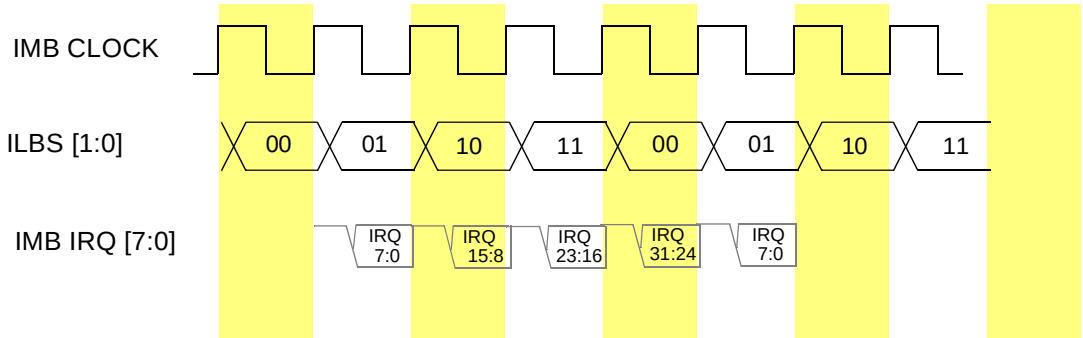


Figure 13-5 Interrupt Levels on IRQ with ILBS

13.2.3 Port Data Register

QADC64E ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB) in each QADC64E.

Port A pins are referred to as PQA[7:0] when used as a bidirectional 8-bit input/output port that may be used for general-purpose digital input signals or digital output signals. Port A of module A can also be used for analog inputs (AN[59:52]) and external multiplexer address outputs (MA[2:0]). Module B Port A pins are shared with AN[79:72] and MA[2:0].

Port B pins are referred to as PQB[7:0] when used as a bidirectional 8-bit input/output port that may be used for general-purpose digital input signals or digital output signals. Data for PQB[7:0] is accessed in the lower half of the Module A. Port B can also be used for non-multiplexed (AN[51:44]) and multiplexed PORTQB ANw, ANx, ANy, ANz) analog inputs. Module B Port B is shared with analog inputs AN[71:64].

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input. When the data direction bit specifies the pin to be an output, the content of the port data register is read.

PORTQA and PORTQB are not initialized by reset.

PORTQA — Port A Data Register **0x30 4806**
0x30 4C06

PORTQB — Port B Data Register **0x30 4807**
0x30 4C07

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
QADC64E_A ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN47	AN46	AN45	AN44
QADC64E_B ANALOG CHANNEL:															
AN79	AN78	AN77	AN76	AN75	AN74	AN73	AN72	AN71	AN70	AN69	AN68	AN67	AN66	AN65	AN64
QADC64E_A and QADC64E_B MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
QADC64E_A (Only) MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

13.2.4 Port Data Direction Register

The port data direction register (DDRQA and DDRQB) is associated with the port A and B digital I/O pins. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information. Any bit set to one in this register configures the corresponding pin as an output. Any bit cleared to zero in this register configures the corresponding pin

as an input. The software is responsible for ensuring that DDR bits are not set to one on pins used for analog inputs. When the DDR bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

Caution should be exercised when mixing digital and analog inputs. This should be isolated as much as possible. Rise and fall times should be as large as possible to minimize AC coupling effects.

There are two special cases to consider for the digital I/O port operation. When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PORTQA[2:0], the three multiplexed address (MA[2:0]) output pins. The MA[2:0] pins are forced to be digital outputs, regardless of the data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the multiplexed address latches which drive MA[2:0], regardless of the data direction setting.

DDRQA — Port A Data Direction Register **0x30 4808**
0x30 4C08

DDRQB — Port B Data Direction Register **0x30 4809**
0x30 4C09

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
DDQA 7	DDQA 6	DDQA 5	DDQA 4	DDQA 3	DDQA 2	DDQA 1	DDQA 0	DDQB 7	DDQB 6	DDQB 5	DDQB 4	DDQB 3	DDQB 2	DDQB 1	DDQB 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.2.5 Control Register 0

Control register 0 establishes the QCLK with prescaler parameter fields and defines whether external multiplexing is enabled. All of the implemented control register fields can be read or written, and reserved fields read zero and writes have no effect. They are typically written once when the software initializes the QADC64E, and not changed afterwards.

QACR0 — Control Register 0 **0x30 480A**
0x30 4C0A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MUX	RESERVED		TRG	RESERVED					PRESCALER						
RESET:															
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Table 13-8 QACR0 Bit Descriptions

Bit(s)	Name	Description
0	MUX	Externally Multiplexed Mode — The MUX bit allows the software to select the externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[0], MA[1] and MA[2] pins to be outputs. 0 = Internally multiplexed, 40 possible channels 1 = Externally multiplexed, up to 65 possible channels See **Table 13-4**
1:2	—	Reserved
3	TRG	Trigger Assignment — The TRG bit allows the software to assign the ETRIG[2:1] pins to queue 1 and queue 2. 0 = ETRIG[1] triggers queue 1, ETRIG[2] triggers queue 2 1 = ETRIG[1] triggers queue 2, ETRIG[2] triggers queue 1
4:8	—	Reserved
9:15	PRESCAL-ER	Prescaler Value — The PRESCALER determines the QCLK frequency. Refer **APPENDIX E ELECTRICAL CHARACTERISTICS** to for more information on QADC64E operating clock frequency (F_{QCLK}) values. To keep the QCLK within the specified range, the PRES-CALAR field selects the divisor -1 of the system clock to create the frequency of the QCLK, which can range from zero to 128 system clock cycles.

Table 13-9 displays the bits in PRESCALER field which enable a range of QCLK frequencies

Table 13-9 Prescaler F_{SYSCLK} Divide-by Values

Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	Fsysclk Div	Prescaler [6:0]	F_{SYSCLK} Div
0000000	2	0100000	33	1000000	65	1100000	97
0000001	2	0100001	34	1000001	66	1100001	98
0000010	3	0100010	35	1000010	67	1100010	99
0000011	4	0100011	36	1000011	68	1100011	100
0000100	5	0100100	37	1000100	69	1100100	101
0000101	6	0100101	38	1000101	70	1100101	102
0000110	7	0100110	39	1000110	71	1100110	103
0000111	8	0100111	40	1000111	72	1100111	104
0001000	9	0101000	41	1001000	73	1101000	105
0001001	10	0101001	42	1001001	74	1101001	106
0001010	11	0101010	43	1001010	75	1101010	107
0001011	12	0101011	44	1001011	76	1101011	108
0001100	13	0101100	45	1001100	77	1101100	109
0001101	14	0101101	46	1001101	78	1101101	110
0001110	15	0101110	47	1001110	79	1101110	111
0001111	16	0101111	48	1001111	80	1101111	112
0010000	17	0110000	49	1010000	81	1110000	113
0010001	18	0110001	50	1010001	82	1110001	114
0010010	19	0110010	51	1010010	83	1110010	115
0010011	20	0110011	52	1010011	84	1110011	116

Table 13-9 Prescaler F_{SYSCLK} Divide-by Values (Continued)

Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	F_{SYSCLK} Div	Prescaler [6:0]	Fsysclk Div	Prescaler [6:0]	F_{SYSCLK} Div
0010100	21	0110100	53	1010100	85	1110100	117
0010101	22	0110101	54	1010101	86	1110101	118
0010110	23	0110110	55	1010110	87	1110110	119
0010111	24	0110111	56	1010111	88	1110111	120
0011000	25	0111000	57	1011000	89	1111000	121
0011001	26	0111001	58	1011001	90	1111001	122
0011010	27	0111010	59	1011010	91	1111010	123
0011011	28	0111011	60	1011011	92	1111011	124
0011100	29	0111100	61	1011100	93	1111100	125
0011101	30	0111101	62	1011101	94	1111101	126
0011110	31	0111110	63	1011110	95	1111110	127
0011111	32	0111111	64	1011111	96	1111111	128

13.2.6 Control Register 1

Control register 1 is the mode control register for the operation of queue 1. The applications software defines the queue operating mode for the queue, and may enable a completion and/or pause interrupt. All of the control register fields are read/write data. However, the SSE1 bit always reads as zero unless the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64E, and not changed afterwards.

QACR1 — Control Register 1 **0x30 480C**
0x30 4C0C

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE1	PIE1	SSE1	MQ1					RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-10 QACR1 Bit Descriptions

Bit(s)	Name	Description
0	CIE1	Queue 1 Completion Interrupt Enable — CIE1 enables an interrupt upon completion of queue 1. The interrupt request is initiated when the conversion is complete for the CCW in queue 1. Refer to **Table 13-11** for more information. 0 = Disable the queue completion interrupt associated with queue 1 1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 1
1	PIE1	Queue 1 Pause Interrupt Enable — PIE1 enables an interrupt when queue 1 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set. Refer to **Table 13-11** for more information. 0 = Disable the pause interrupt associated with queue 1 1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 1 which has the pause bit set

Table 13-10 QACR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
2	SSE1	Queue 1 Single-Scan Enable Bit — SSE1 enables a single-scan of queue 1 to start after a trigger event occurs. The SSE1 bit may be set to a one during the same write cycle when the MQ1 bits are set for one of the single-scan queue operating modes. The single-scan enable bit can be written as a one or a zero, but is always read as a zero, unless a test mode is selected. The SSE1 bit enables a trigger event to initiate queue execution for any single-scan operation on queue 1. The QADC64E clears the SSE1 bit when the single-scan is complete. Refer to **Table 13-11** for more information. 0 = Trigger events are not accepted for single-scan modes 1 = Accept a trigger event to start queue 1 in a single-scan mode
3:7	MQ1	Queue 1 Operating Mode — The MQ1 field selects the queue operating mode for queue 1. **Table 13-11** shows the bits in the MQ1 field which enable different queue 1 operating mode
8:15	—	Reserved

Table 13-11 Queue 1 Operating Modes

MQ1[3:7]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE1)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	External gated single-scan mode (started with SSE1)
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
10110	Periodic timer timer continuous-scan mode: time = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}

Table 13-11 Queue 1 Operating Modes (Continued)

MQ1[3:7]	Operating Modes
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{1}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic l timer continuous-scan mode: time = QCLK period x 2^{17}
11111	External gated continuous-scan mode

13.2.7 Control Register 2

Control register 2 is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit, which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64E, and not changed afterwards.

QACR2 — Control Register 2 **0x30 480E**
0x30 4C0E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CIE2	PIE2	SSE2	MQ2					RE-SUME	BQ2						
RESET:															
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 13-12 QACR2 Bit Descriptions

Bit(s)	Name	Description
0	CIE2	Queue 2 Completion Software Interrupt Enable — CIE2 enables an interrupt upon completion of queue 2. The interrupt request is initiated when the conversion is complete for the CCW in queue 2. Refer to **Table 13-13** for more information. 0 = Disable the queue completion interrupt associated with queue 2 1 = Enable an interrupt after the conversion of the sample requested by the last CCW in queue 2
1	PIE2	Queue 2 Pause Software Interrupt Enable — PIE2 enables an interrupt when queue 2 enters the pause state. The interrupt request is initiated when conversion is complete for a CCW that has the pause bit set. Refer to **Table 13-13** for more information. 0 = Disable the pause interrupt associated with queue 2 1 = Enable an interrupt after the conversion of the sample requested by a CCW in queue 2 which has the pause bit set

Table 13-12 QACR2 Bit Descriptions (Continued)

Bit(s)	Name	Description
2	SSE2	Queue 2 Single-Scan Enable Bit — SSE2 enables a single-scan of queue 2 to start after a trigger event occurs. The SSE2 bit may be set to a one during the same write cycle when the MQ2 bits are set for one of the single-scan queue operating modes. The single-scan enable bit can be written as a one or a zero, but is always read as a zero, unless a test mode is selected. The SSE2 bit enables a trigger event to initiate queue execution for any single-scan operation on queue 2. The QADC64E clears the SSE2 bit when the single-scan is complete. Refer to **Table 13-13** for more information. 0 = Trigger events are not accepted for single-scan modes 1 = Accept a trigger event to start queue 2 in a single-scan mode
3:7	MQ2	Queue 2 Operating Mode — The MQ2 field selects the queue operating mode for queue 2. Refer to **Table 13-13** for more information.
8	RESUME	0 = After suspension, begin executing with the first CCW in queue 2 or the current sub-queue 1 = After suspension, begin executing with the aborted CCW in queue 2
9:15	BQ2	Beginning of queue 2 — The BQ2 field indicates the CCW location where queue 2 begins. To allow the length of queue 1 and queue 2 to vary, a programmable pointer identifies the CCW table location where queue 2 begins. The BQ2 field also serves as an end-of-queue condition for queue 1. Setting BQ2 beyond physical CCW table memory space allows queue 1 all 64 entries. Software defines the beginning of queue 2 by programming the BQ2 field in QACR2. BQ2 is usually programmed before or at the same time as the queue operating mode for queue 2 is selected. If BQ2 is 64 or greater, queue 2 has no entries, and the entire CCW table is dedicated to queue 1 and CCW63 is the end-of-queue 1. If BQ2 is zero, the entire CCW table is dedicated to queue 2. As a special case, when a queue operating mode for queue 1 is selected and a trigger event occurs for queue 1 with BQ2 set to zero, queue 1 execution is terminated after CCW0 is read. Conversions do not occur. The BQ2 pointer may be changed dynamically, to alternate between queue 2 scan sequences. A change in BQ2 after queue 2 has begun or if queue 2 has a trigger pending does not affect queue 2 until queue 2 is started again.For example, two scan sequences could be defined as follows: the first sequence starts at CCW10, with a pause after CCW11 and an EOQ programmed in CCW15; the second sequence starts at CCW16, with a pause after CCW17 and an EOQ programmed in CCW39. With BQ2 set to CCW10 and the continuous-scan mode selected, queue execution begins. When the pause is encountered in CCW11, a software interrupt routine can redefine BQ2 to be CCW16. Therefore, after the end-of-queue is recognized in CCW15, an internal retrigger event is generated and execution restarts at CCW16. When the pause software interrupt occurs again, software can change BQ2 back to CCW10. After the end-of-queue is recognized in CCW39, an internal retrigger event is created and execution now restarts at CCW10. If BQ2 is changed while queue 1 is active, the effect of BQ2 as an end-of-queue indication for queue 1 is immediate. However, beware of the risk of losing the end-of-queue 1 through moving BQ2. Recommend use of EOQ (chan63) to end queue 1. Refer to **Table 13-13** for more information.

Table 13-13 shows the bits in the MQ2 field which enable different queue 2 operating modes.

Table 13-13 Queue 2 Operating Modes

MQ2[3:7]	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	Reserved mode

NOTE

If BQ2 was assigned to the CCW that queue 1 is currently working on, then that conversion is completed before BQ2 takes effect.

Each time a CCW is read for queue 1, the CCW location is compared with the current value of the BQ2 pointer to detect a possible end-of-queue condition. For example, if BQ2 is changed to CCW3 while queue 1 is converting CCW2, queue 1 is terminated after the conversion is completed. However, if BQ2 is changed to CCW1 while queue 1 is converting CCW2, the QADC64E would not recognize a BQ2 end-of-queue condition until queue 1 execution reached CCW1 again, presumably on the next pass through the queue.

13.2.8 Status Registers

The status registers contains information about the state of each queue and the current A/D conversion. Except for the four flag bits (CF1, PF1, CF2, and PF2) and the two trigger overrun bits (TOR1 and TOR2), all of the status register fields contain read-only data. The four flag bits and the two trigger overrun bits are cleared by writing a zero to the bit after the bit was previously read as a one.

QASR0 — Status Register 0 **0x30 4810**
0x30 4C10

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-14 QASR0 Bit Descriptions

Bit(s)	Name	Description
0	CF1	Queue 1 Completion Flag — CF1 indicates that a queue 1 scan has been completed. The scan completion flag is set by the QADC64E when the input channel sample requested by the last CCW in queue 1 is converted, and the result is stored in the result table. The end-of-queue 1 is identified when execution is complete on the CCW in the location prior to that pointed to by BQ2, when the current CCW contains an end-of-queue code instead of a valid channel number, or when the currently completed CCW is in the last location of the CCW RAM. When CF1 is set and interrupts are enabled for that queue completion flag, the QADC64E asserts an interrupt request at the level specified by IRL1 in the interrupt register (QADCINT). The software reads the completion flag during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to the completion flag bit, when the bit was previously read as a one. Once set, only software or reset can clear CF1. CF1 is maintained by the QADC64E regardless of whether the corresponding interrupt is enabled. The software polls for CF1 bit to see if it is set. This allows the software to recognize that the QADC64E is finished with a queue 1 scan. The software acknowledges that it has detected the completion flag being set by writing a zero to the completion flag after the bit was read as a one.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
1	PF1	Queue 1 Pause Flag — PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC64E when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. Once PF1 is set, the queue enters the paused state and waits for a trigger event to allow queue execution to continue. However, if the CCW with the pause bit set is the last CCW in a queue, the queue execution is complete. The queue status becomes idle, not paused, and both the pause and completion flags are set. Another exception occurs in software controlled mode, where the PF1 can be set but queue 1 never enters the pause state since queue 1 continues without pausing. When PF1 is set and interrupts are enabled for the corresponding queue, the QADC64E asserts an interrupt request at the level specified by IRL1 in the interrupt register. The software may read PF1 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to PF1, when the bit was previously read as a one. Once set, only software or reset can clear PF1. In external gated single-scan and continuous-scan mode the definition of PF1 has been redefined. When the gate closes before the end-of-queue 1 is reached, PF1 becomes set to indicate that an incomplete scan has occurred.In single-scan mode, setting PF1 can be used to cause an interrupt and software can then determine if queue 1 should be enabled again. In either external gated mode, setting PF1 indicates that the results for queue 1 have not been collected during one scan (coherently). NOTE: If a pause in a CCW is encountered in external gated mode for either single-scan and continuous-scan mode, the pause flag **will not set**, and execution continues without pausing. This has allowed for the added definition of PF1 in the external gated modes. PF1 is maintained by the QADC64E regardless of whether the corresponding interrupts are enabled. The software may poll PF1 to find out when the QADC64E has reached a pause in scanning a queue.The software acknowledges that it has detected a pause flag being set by writing a zero to PF1 after the bit was last read as a one. 0 = queue 1 has not reached a pause (or gate has not closed before end-of-queue in gated mode) 1 = queue 1 has reached a pause (or gate closed before end-of-queue in gated mode) Refer to **Table 13-15** for a summary of pause response in all scan modes.
2	CF2	Queue 2 Completion Flag — CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC64E when the input channel sample requested by the last CCW in queue 2 is converted, and the result is stored in the result table. The end-of-queue 2 is identified when the current CCW contains an end-of-queue code instead of a valid channel number, or when the currently completed CCW is in the last location of the CCW RAM. When CF2 is set and interrupts are enabled for that queue completion flag, the QADC64E asserts an interrupt request at the level specified by IRL2 in the interrupt register (QADCINT). The software reads CF2 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to the CF2 bit, when the bit was previously read as a one. Once set, only software or reset can clear CF2. CF2 is maintained by the QADC64E regardless of whether the corresponding interrupts are enabled. The software polls for CF2 to see if it is set. This allows the software to recognize that the QADC64E is finished with a queue 2 scan. The software acknowledges that it has detected the completion flag being set by writing a zero to the completion flag after the bit was read as a one. Refer to **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
3	PF2	Queue 2 Pause Flag — PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC64E when the current queue 2 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. Once PF2 is set, the queue enters the paused state and waits for a trigger event to allow queue execution to continue. However, if the CCW with the pause bit set is the last CCW in a queue, the queue execution is complete. The queue status becomes idle, not paused, and both the pause and completion flags are set. Another exception occurs in software controlled mode, where the PF2 can be set but queue 2 never enters the pause state. When PF2 is set and interrupts are enabled for the corresponding queue, the QADC64E asserts an interrupt request at the level specified by IRL2 in the interrupt register. The software reads PF2 during an interrupt service routine to identify the interrupt request. The interrupt request is cleared when the software writes a zero to PF2, when the bit was previously read as a one. Once set, only software or reset can clear PF2. PF2 is maintained by the QADC64E regardless of whether the corresponding interrupts are enabled. The software may poll PF2 to find out when the QADC64E has reached a pause in scanning a queue. The software acknowledges that it has detected a pause flag being set by writing a zero to PF2 after the bit was last read as a one. 0 = queue 2 has not reached a pause 1 = queue 2 has reached a pause Refer to **Table 13-15** for a summary of pause response in all scan modes.
4	TOR1	Queue 1 Trigger Overrun — TOR1 indicates that an unexpected trigger event has occurred for queue 1. TOR1 can be set only while queue 1 is in the active state. A trigger event generated by a transition on the external trigger pin or by the periodic/interval timer may be captured as a trigger overrun. TOR1 cannot occur when the software initiated single-scan mode or the software initiated continuous-scan mode are selected. TOR1 occurs when a trigger event is received while a queue is executing and before the scan has completed or paused. TOR1 has no effect on the queue execution. After a trigger event has occurred for queue 1, and before the scan has completed or paused, additional queue 1 trigger events are not retained. Such trigger events are considered unexpected, and the QADC64E sets the TOR1 error status bit. An unexpected trigger event may be a system overrun situation, indicating a system loading mismatch. In external gated continuous-scan mode the definition of TOR1 has been redefined. In the case when queue 1 reaches an end-of-queue condition for the second time during an open gate, TOR1 becomes set. This is considered an overrun condition. In this case CF1 has been set for the first end-of-queue 1 condition and then TOR1 becomes set for the second end-of-queue 1 condition. For TOR1 to be set, software must not clear CF1 before the second end-of-queue 1. The software acknowledges that it has detected a trigger overrun being set by writing a zero to the trigger overrun, after the bit was read as a one. Once set, only software or reset can clear TOR1. 0 = No unexpected queue 1 trigger events have occurred 1 = At least one unexpected queue 1 trigger event has occurred (or queue 1 reaches an end-of-queue condition for the second time in gated mode) Refer to **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
5	TOR2	Queue 2 Trigger Overrun — TOR2 indicates that an unexpected trigger event has occurred for queue 2. TOR2 can be set when queue 2 is in the active, suspended, and trigger pending states. The TOR2 trigger overrun can only occur when using an external trigger mode or a periodic/interval timer mode. Trigger overruns cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected. TOR2 occurs when a trigger event is received while queue 2 is executing, suspended, or a trigger is pending. TOR2 has no effect on the queue execution. A trigger event that causes a trigger overrun is not retained since it is considered unexpected. An unexpected trigger event may be a system overrun situation, indicating a system loading mismatch. The software acknowledges that it has detected a trigger overrun being set by writing a zero to the trigger overrun, after the bit was read as a one. Once set, only software or reset can clear TOR2. 0 = No unexpected queue 2 trigger events have occurred 1 = At least one unexpected queue 2 trigger event has occurred Refer to **Table 13-15** for a summary of pause response in all scan modes.

Table 13-14 QASR0 Bit Descriptions (Continued)

Bit(s)	Name	Description
6:9	QS	Queue Status The 4-bit read-only QS field indicates the current condition of queue 1 and queue 2. The following are the five queue status conditions: Idle Active Paused Suspended Trigger pending The two most significant bits are associated primarily with queue 1, and the remaining two bits are associated with queue 2. Since the priority scheme between the two queues causes the status to be interlinked, the status bits are considered as one 4-bit field. **Table 13-16** shows the bits in the QS field and how they affect the status of queue 1 and queue 2. Refer to **13.5 Trigger and Queue Interaction Examples**, which shows the 4-bit queue status field transitions in typical situations.
10:15	CWP	Command Word Pointer — The CWP allows the software to know which CCW is executing at present, or was last completed. The command word pointer is a software read-only field, and write operations have no effect. The CWP allows software to monitor the progress of the QADC64E scan sequence. The CWP field is a CCW word pointer with a valid range of 0 to 63. When a queue enters the paused state, the CWP points to the CCW with the pause bit set. While in pause, the CWP value is maintained until a trigger event occurs on the same queue or the other queue. Usually, the CWP is updated a few clock cycles before the queue status field shows that the queue has become active. For example, software may read a CWP pointing to a CCW in queue 2, and the status field shows queue 1 paused, queue 2 trigger pending. When the QADC64E finishes the scan of the queue, the CWP points to the CCW where the end-of-queue condition was detected. Therefore, when the end-of-queue condition is a CCW with the EOQ code (channel 63 or 127), the CWP points to the CCW containing the EOQ. When the last CCW in a queue is in the last CCW table location (CCW63), and it does not contain the EOQ code, the end-of-queue is detected when the following CCW is read, so the CWP points to word CCW0. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, the CWP points to the same CCW as BQ2. During the stop mode, the CWP is reset to zero, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW.

Table 13-15 Pause Response

Scan Mode	Q Operation	PF Asserts?
External Trigger Single-scan	Pauses	Yes
External Trigger Continuous-scan	Pauses	Yes
Periodic/Interval Timer Trigger Single-scan	Pauses	Yes
Periodic/Interval Timer Continuous-scan	Pauses	Yes
Software Initiated Single-scan	Continues	Yes
Software Initiated Continuous-scan	Continues	Yes
External Gated Single-scan	Continues	No
External Gated Continuous-scan	Continues	No

Table 13-16 Queue Status

QS[9:6]	Queue 1/Queue 2 States
0000	queue 1 idle, queue 2 idle
0001	queue 1 idle, queue 2 paused
0010	queue 1 idle, queue 2 active
0011	queue 1 idle, queue 2 trigger pending
0100	queue 1 paused, queue 2 idle
0101	queue 1 paused, queue 2 paused
0110	queue 1 paused, queue 2 active
0111	queue 1 paused, queue 2 trigger pending
1000	queue 1 active, queue 2 idle
1001	queue 1 active, queue 2 paused
1010	queue 1 active, queue 2 suspended
1011	queue 1 active, queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

One or both queues may be in the idle state. When a queue is idle, CCWs are not being executed for that queue, the queue is not in the pause state, and there is not a trigger pending.

The idle state occurs when a queue is disabled, when a queue is in a reserved mode, or when a queue is in a valid queue operating mode awaiting a trigger event to initiate queue execution.

A queue is in the active state when a valid queue operating mode is selected, when the selected trigger event has occurred, or when the QADC64E is performing a conversion specified by a CCW from that queue.

Only one queue can be active at a time. Either or both queues can be in the paused state. A queue is paused when the previous CCW executed from that queue had the pause bit set. The QADC64E does not execute any CCWs from the paused queue until a trigger event occurs. Consequently, the QADC64E can service queue 2 while queue 1 is paused.

Only queue 2 can be in the suspended state. When a trigger event occurs on queue 1 while queue 2 is executing, the current queue 2 conversion is aborted. The queue 2 status is reported as suspended. Queue 2 transitions back to the active state when queue 1 becomes idle or paused.

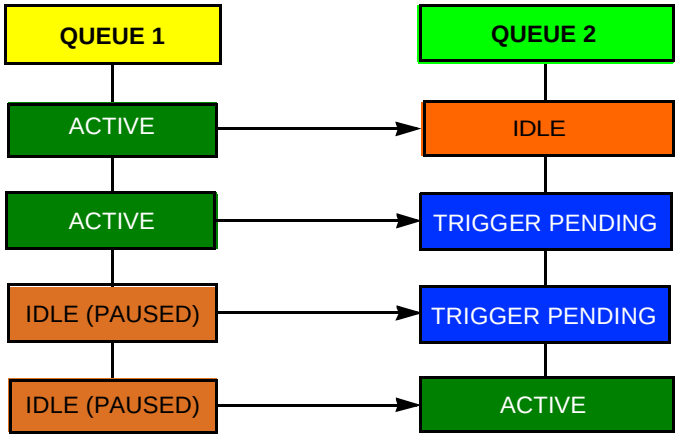
A trigger pending state is required since both queues cannot be active at the same time. The status of queue 2 is changed to trigger pending when a trigger event occurs for queue 2 while queue 1 is active. In the opposite case, when a trigger event occurs for queue 1 while queue 2 is active, queue 2 is aborted and the status is reported as

queue 1 active, queue 2 suspended. So due to the priority scheme, only queue 2 can be in the trigger pending state.

There are two transition cases which cause the queue 2 status to be trigger pending before queue 2 is shown to be in the active state. When queue 1 is active and there is a trigger pending on queue 2, after queue 1 completes or pauses, queue 2 continues to be in the trigger pending state for a few clock cycles. The following are fleeting status conditions:

- Queue 1 idle with queue 2 trigger pending
- Queue 1 paused with queue 2 trigger pending

Figure 13-6 displays the status conditions of the queue status field as the QADC64E goes through the transition from queue 1 active to queue 2 active.



QADC64E QUEUE STATUS

Figure 13-6 Queue Status Transition

The queue status field is affected by the stop mode. Since all of the analog logic and control registers are reset, the queue status field is reset to queue 1 idle, queue 2 idle.

During the freeze mode, the queue status field is not modified. The queue status field retains the status it held prior to freezing. As a result, the queue status can show queue 1 active, queue 2 idle, even though neither queue is being executed during freeze.

QASR1 — Status Register 1 **0x30 4812**
0x30 4C12

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED		CWPQ1						RESERVED		CWPQ2					
RESET:															
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Table 13-17 QASR1 Bit Descriptions

Bit(s)	Name	Description
0:1	—	Reserved
2:7	CWPQ1	Command Word Pointer for Q1 — CWPQ1 allows the software to know what CCW was last completed for queue 1. This field is a software read-only field, and write operations have no effect. CWPQ1 allows software to read the last executed CCW in queue 1, regardless of which queue is active. The CWPQ1 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ1 is updated when the conversion result is written. When the QADC64E finishes a conversion in queue 1, both the result register is written and the CWPQ1 are updated. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, CWP points to BQ2 while CWPQ1 points to the last CCW queue 1. During the stop mode, the CWPQ1 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWPQ1 is unchanged; it points to the last executed CCW in queue 1.
8:9	—	Reserved
10:15	CWPQ2	Command Word Pointer for Q2 — CWPQ2 allows the software to know what CCW was last completed for queue 2. This field is a software read-only field, and write operations have no effect. CWPQ2 allows software to read the last executed CCW in queue 2, regardless which queue is active. The CWPQ2 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ2 is updated when the conversion result is written. When the QADC64E finishes a conversion in queue 2, both the result register is written and the CWPQ2 are updated. During the stop mode, the CWPQ2 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW in queue 2.

13.2.9 Conversion Command Word Table

The conversion command word (CCW) table is a RAM, 64 words long on 16-bit address boundaries where 10-bits of each entry are implemented. A CCW can be programmed by the software to request a conversion of one analog input channel. The CCW table is written by software and is not modified by the QADC64E. Each CCW requests the conversion of an analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW. The ten implemented bits of the CCW word are read/write data, where they may be written when the software initializes the QADC64E. The remaining 6-bits are unimplemented so these read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry. The QADC64E provides 64 CCW table entries.

The beginning of queue 1 is the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, queue 2 is programmed to be in the disabled mode, and BQ2 is programmed to 64 or greater. To dedicate the entire CCW table to queue 2, queue 1 is programmed to be in the disabled mode, and BQ2 is specified as the first location in the CCW table

Figure 13-7 illustrates the operation of the queue structure.

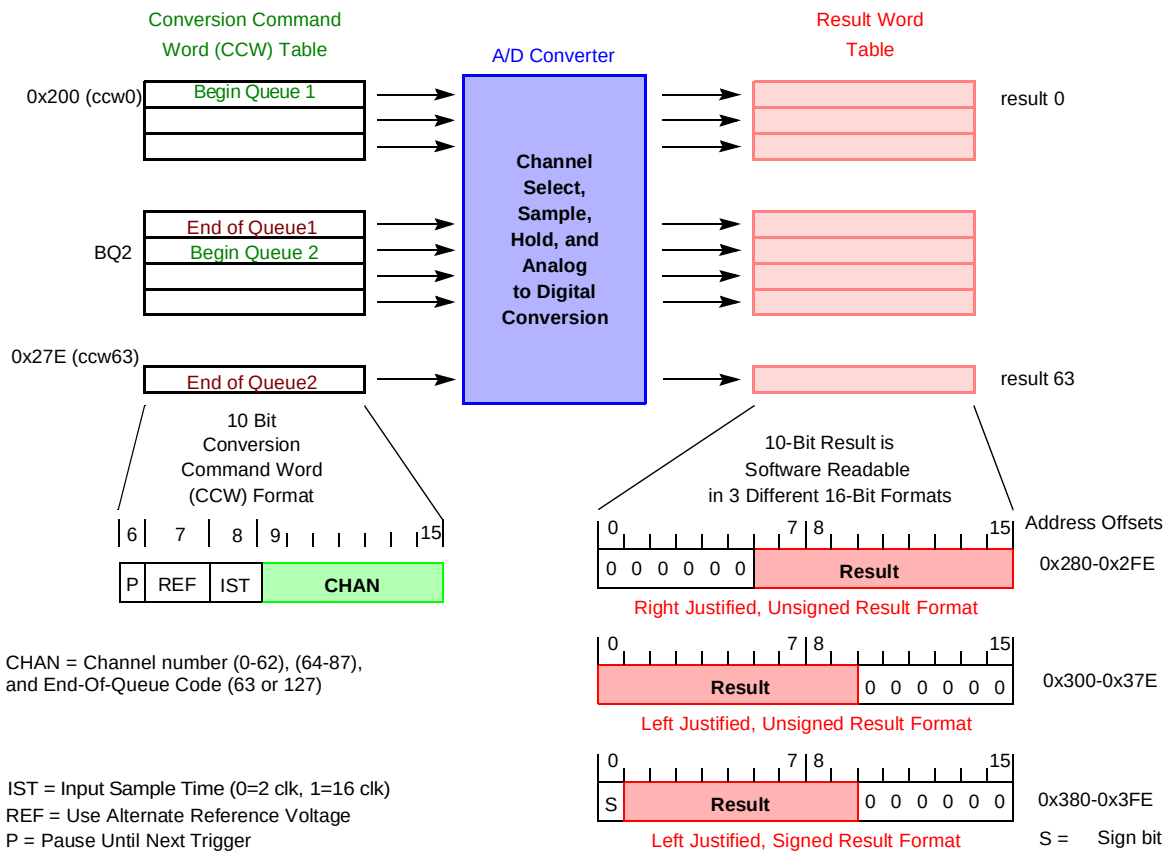


Figure 13-7 QADC64E Conversion Queue Operation

To prepare the QADC64E for a scan sequence, the software writes to the CCW table to specify the desired channel conversions. The software also establishes the criteria for initiating the queue execution by programming the queue operating mode. The queue operating mode determines what type of trigger event causes queue execution to begin. A “trigger event” is used to refer to any of the ways to cause the QADC64E to begin executing the CCWs in a queue or sub-queue. An “external trigger” is only one of the possible “trigger events.”

A scan sequence may be initiated by the following:

- A software command
- Expiration of the periodic/interval timer
- External trigger signal
- External gated signal (queue 1 only)

The software also specifies whether the QADC64E is to perform a single pass through the queue or is to scan continuously. When a single-scan mode is selected, the software selects the queue operating mode and sets the single-scan enable bit. When a

continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC64E completes each queue scan sequence.

During queue execution, the QADC64E reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample
- Resolution

During initial sample, a buffered version of the selected input channel is connected to the sample capacitor at the input of the sample buffer amplifier.

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of two or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC64E continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC64E stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, which may cause an interrupt to notify the software that the queue has reached the pause state. After the trigger event occurs, the paused state ends and the QADC64E continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

The following indicate the end-of-queue condition:

- The CCW channel field is programmed with 63 (0x3F) or 127 (0x7F) to specify the end of the queue
- The end-of-queue 1 is implied by the beginning of queue 2, which is specified in the BQ2 field in QACR2
- The physical end of the queue RAM space defines the end of either queue

When any of the end-of-queue conditions is recognized, a queue completion flag is set, and if enabled, an interrupt is issued to the software. The following situations prematurely terminate queue execution:

- Since queue 1 is higher in priority than queue 2, when a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and the queue 1 execution begins. When queue 1 execution is completed, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 sub-queue being executed when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit in QACR2 allows the software to select where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 queue and sub-queue CCWs, all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger

event rate for queue 1 can prohibit the completion of queue 2. If this occurs, the software may choose to begin execution of queue 2 with the aborted CCW entry.

- Software can change the queue operating mode to disabled mode. Any conversion in progress for that queue is aborted. Putting a queue into the disabled mode does not power down the converter.
- Software can change the queue operating mode to another valid mode. Any conversion in progress for that queue is aborted. The queue restarts at the beginning of the queue, once an appropriate trigger event occurs.
- For low power operation, software can set the stop mode bit to prepare the module for a loss of clocks. The QADC64E aborts any conversion in progress when the stop mode is entered.
- When the freeze enable bit is set by software and the IMB internal FREEZE line is asserted, the QADC64E freezes at the end of the conversion in progress. When internal FREEZE is negated, the QADC64E resumes queue execution beginning with the next CCW entry. Refer to **13.4.7 Configuration And Control using the IMB Interface** for more information.

CCW — Conversion Command Word Table **0x30 4A00 – 0x30 4A7F**
0x30 4E00 – 0x30 4E7F

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						P	REF	IST	CHAN[6:0]						
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 13-18 CCW Bit Descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	P	Pause — The pause bit allows software to create sub-queues within queue 1 and queue 2. The QADC64E performs the conversion specified by the CCW with the pause bit set, and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW. 0 = Do not enter the pause state after execution of the current CCW 1 = Enter the pause state after execution of the current CCW **NOTE:** The pause bit will not cause the queue to pause in the software controlled modes or external gated modes.
7	REF	Alternate Reference Enabled — Setting REF high in the CCW enables the use of an alternate reference. 0 = VRH is used as high reference 1 = AltRef pin is used as the high reference

Table 13-18 CCW Bit Descriptions (Continued)

Bit(s)	Name	Description
8	IST	Input Sample Time — The IST field allows software to specify the length of the sample window. Provision is made to vary the input sample time, through software control, to offer flexibility in the source impedance of the circuitry providing the QADC64E analog channel inputs. Longer sample times permit more accurate A/D conversions of signals with higher source impedances. The programmable sample time can also be used to increase the time interval between conversions to adjust the queue execution time or the sampling rate. **Table 13-19** shows the two selectable input sample times.
9:15	CHAN[6:0]	CHAN[6:0] — Channel Number — The CHAN field selects the input channel number. The software programs the channel field of the CCW with the channel number corresponding to the analog input pin to be sampled and converted. The analog input pin channel number assignments and the pin definitions vary depending on whether the multiplexed or non-multiplexed mode is used by the application. As far as the queue scanning operations are concerned, there is no distinction between an internally or externally multiplexed analog input. Refer to **13.1.5 External Multiplexing** for more information on external multiplexing. **Table 13-20** shows the channel number assignments

Table 13-19 Input Sample Times

IST	Input Sample Times
0	Input sample time = QCLK period x 2
1	Input sample time = QCLK period x 16

Table 13-20 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				Channel Number in CCW CHAN Field	
Port Pin Name	**Analog Pin Name**	**Other Functions / Descriptions**	**Pin Type**	**Binary**	**Decimal**
ANw/ A_PQB[0]	AN[00] to AN[07]	—	Input	0000000 to 0000111	0 to 7
ANx/A_PQB[1]	AN[08] to AN[15]	—	Input	0001000 to 0001111	8 to 15
ANy/A_PQB[2]	AN[16] to AN[23]	—	Input	0010000 to 0010111	16 to 23
ANz/A_PQB[3]	AN[24] to AN[31]	—	Input	0011000 to 0011111	24 to 31
—	RESERVED	—	—	0100000 to 0101001	32 to 41
—	RESERVED	—	—	0101010	42
—	RESERVED	—	—	0101011	43
A_PQB[0] A_PQB[1] A_PQB[2] A_PQB[3]	AN[44] AN[45] AN[46] AN[47]	ANw ANx ANy ANz	Input/Output Input/Output Input/Output Input/Output	0101100 0101101 0101110 0101111	44 45 46 47
A_PQB[4] A_PQB[5] A_PQB[6] A_PQB[7]	AN[48] AN[49] AN[50] AN[51]	— — — —	Input/Output Input/Output Input/Output Input/Output	0110000 0110001 0110010 0110011	48 49 50 51
A_PQA[0] A_PQA[1] A_PQA[2] A_PQA[3]	AN[52] AN[53] AN[54] AN[55]	MA[0] MA[1] MA[2] —	Input/Output Input/Output Input/Output Input/Output	0110100 0110101 0110110 0110111	52 53 54 55
A_PQA[4] A_PQA[5] A_PQA[6] A_PQA[7]	AN[56] AN[57] AN[58] AN[59]	— — — —	Input/Output Input/Output Input/Output Input/Output	0111000 0111001 0111010 0111011	56 57 58 59
VRL VRH/ALTREF[1] —	Low Ref High Ref —	— — (VRH – VRL)/2	Input Input —	0111100 0111101 0111110	60 61 62
—	—	End of Queue Code	—	0111111	63

Table 13-20 Multiplexed Channel Assignments and Pin Designations (Continued)

Multiplexed Input Pins				Channel Number in CCW CHAN Field	
Port Pin Name	**Analog Pin Name**	**Other Functions / Descriptions**	**Pin Type**	**Binary**	**Decimal**
B_PQB[0]	AN[64]	—	AMUX Input	1000000	64
B_PQB[1]	AN[65]	—	AMUX Input	1000001	65
B_PQB[2]	AN[66]	—	AMUX Input	1000010	66
B_PQB[3]	AN[67]	—	AMUX Input	1000011	67
B_PQB[4]	AN[68]	—	AMUX Input	1000100	68
B_PQB[5]	AN[69]	—	AMUX Input	1000101	69
B_PQB[6]	AN[70]	—	AMUX Input	1000110	70
B_PQB[7]	AN[71]	—	AMUX Input	1000111	71
B_PQA[0]	AN[72]	MA[0]	AMUX Input	1001000	72
B_PQA[1]	AN[73]	MA[1]	AMUX Input	1001001	73
B_PQA[2]	AN[74]	MA[2]	AMUX Input	1001010	74
B_PQA[3]	AN[75]	—	AMUX Input	1001011	75
B_PQA[4]	AN[76]	—	AMUX Input	1001100	76
B_PQA[5]	AN[77]	—	AMUX Input	1001101	77
B_PQA[6]	AN[78]	—	AMUX Input	1001110	78
B_PQA[7]	AN[79]	—	AMUX Input	1001111	79
—	AN[80]	—	AMUX Input	1010000	80
—	AN[81]	—	AMUX Input	1010001	81
—	AN[82]	—	AMUX Input	1010010	82
—	AN[83]	—	AMUX Input	1010011	83
—	AN[84]	—	AMUX Input	1010100	84
—	AN[85]	—	AMUX Input	1010101	85
—	AN[86]	—	AMUX Input	1010110	86
—	AN[87]	—	AMUX Input	1010111	87
—	RESERVED	—	—	1011000 to 1111110	88 to 126
—	—	End of Queue Code	—	1111111	127

NOTES:
1. Whichever is selected in the CCW.

RESERVED CHANNELS

The channel field is programmed for channel 63 or 127 to indicate the end of the queue. Channels 60 to 62 are special internal channels. When one of the special channels is selected, the sampling amplifier is not used. The value of V_{RL}, V_{RH}, or $(V_{RH} - V_{RL})/2$ is placed directly onto the converter. Also for the internal special channels, programming any input sample time other than two has no benefit except to lengthen the overall conversion time.

13.2.10 Result Word Table

The result word table is a RAM, 64 words long and 10 bits wide. An entry is written by the QADC64E after completing an analog conversion specified by the corresponding CCW table entry. Software can read or write the result word table, but in normal operation, the software reads the result word table to obtain analog conversions from the

QADC64E. Unimplemented bits are read as zeros, and write operations do not have any effect. See **Figure 13-7** for a diagram of the result word table

While there is only one result word table, the data can be accessed in three different data formats:

- Right justified in the 16-bit word, with zeros in the higher order unused bits
- Left justified, with the most significant bit inverted to form a sign bit, and zeros in the unused lower order bits
- Left justified, with zeros in the lower order unused bits

The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The data is routed onto the IMB according to the selected format. The address used to access the table determines the data alignment format. All write operations to the result word table are right justified.

RJURR — Right Justified, Unsigned Result Format **0x30 4A80 – 0x30 4AFF**
0x30 4E80 – 0x30 4EFF

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED						RESULT									
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

LJSRR — Left Justified, Signed Result Format **0x30 4B00 – 0x03 4B7F**
0x30 4F00 – 0x03 4F7F

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
S[1]	RESULT									RESERVED					
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

NOTES:
1. S = Sign bit.

LJURR — Left Justified, Unsigned Result Register **0x30 4B80 – 0x03 4BFF**
0x30 4F80 – 0x03 4FFF

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESULT										RESERVED					
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

The three result data formats are produced by routing the RAM bits onto the data bus. The software chooses among the three formats by reading the result at the memory address which produces the desired data alignment.

The result word table is read/write accessible by software. During normal operation, applications software only needs to read the result table. Write operations to the table may occur during test or debug breakpoint operation. When locations in the CCW table are not used by an application, software could use the corresponding locations in the result word table as scratch pad RAM, remembering that only 10 bits are implemented. The result alignment is only implemented for software read operations. Since write operations are not the normal use for the result registers, only one write data format is supported, which is right justified data.

NOTE

Some write operations, like bit manipulation, may not operate as expected because the hardware cannot access a true 16-bit value.

13.3 Analog Subsystem

This section describes the QADC64E analog subsystem, which includes the front-end analog multiplexer and analog-to-digital converter.

13.3.1 Analog-to-Digital Converter Operation

The analog subsystem consists of the path from the input pins to the A/D converter block. Signals from the queue control logic are fed to the multiplexer and state machine. The end of convert (EOC) signal and the successive-approximation register (SAR) are the result of the conversion. **Figure 13-8** shows a block diagram of the QADC64E analog subsystem.

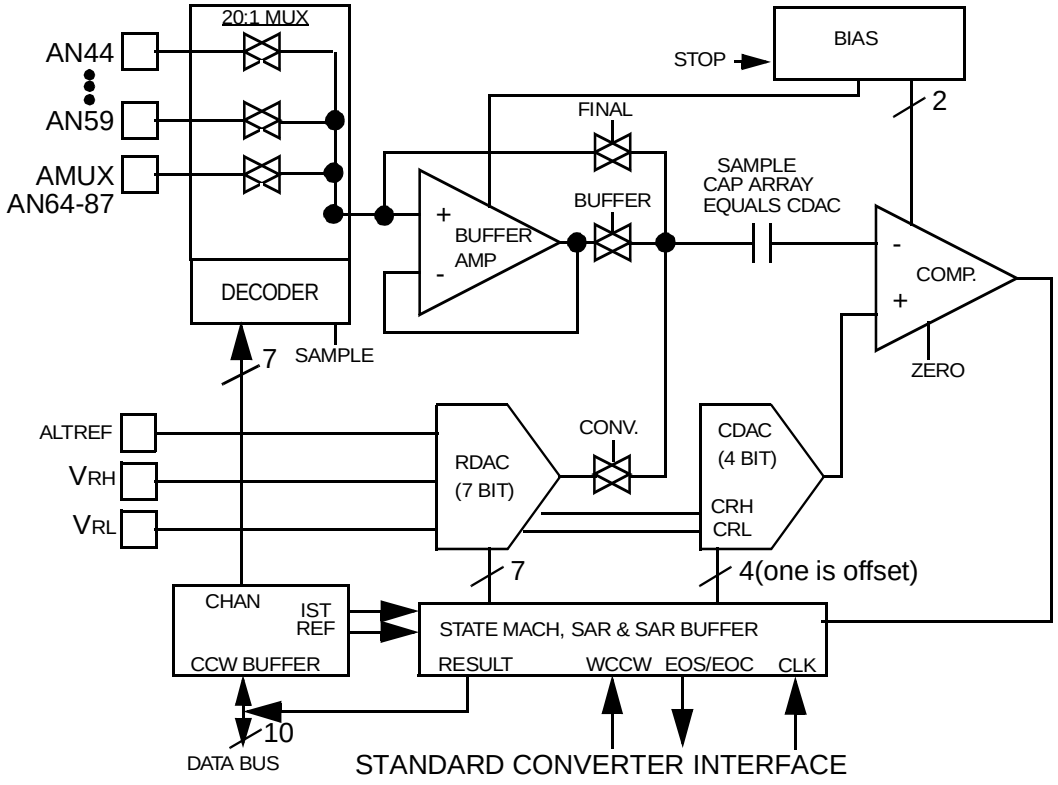


Figure 13-8 QADC64E Analog Subsystem Block Diagram

13.3.1.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is coupled through the buffer amplifier to the sample capacitor. This buffer is used to quickly reproduce its input signal on the sample capacitor and minimize charge sharing errors. During the final sampling period the amplifier is bypassed, and the multiplexer input charges the sample capacitor array directly for improved accuracy. During the resolution period, the voltage in the sample capacitor is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLK cycles. Final sample time can be two or sixteen QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is ten QCLK cycles.

Therefore, conversion time requires a minimum of 14 QCLK clocks (seven µs with a 2.0-MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 28 QCLKs or 14 µs (with a 2.0-MHz QCLK)

Figure 13-9 illustrates the timing for conversions.

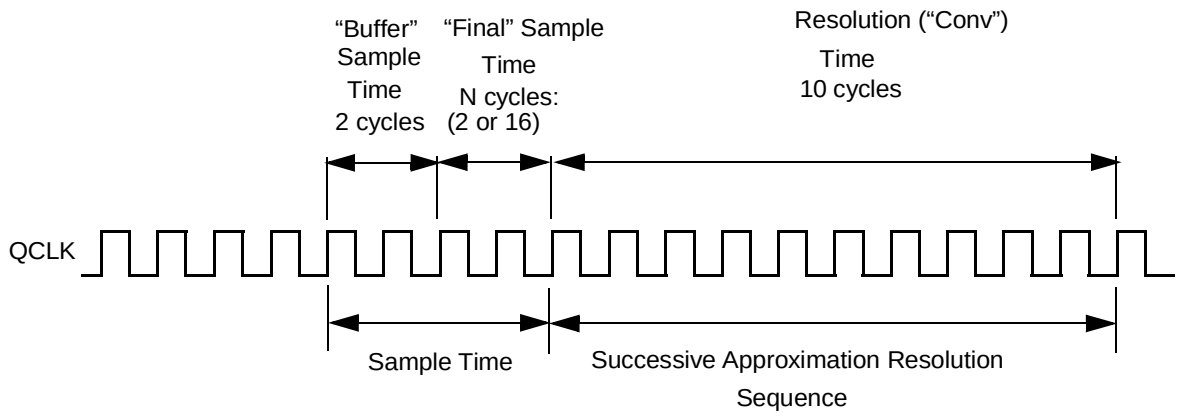


Figure 13-9 Conversion Timing

13.3.2 Channel Decode and Multiplexer

The internal multiplexer selects one of the 40 analog input pins for conversion. The selected input is connected to the sample buffer amplifier. The multiplexer also includes positive and negative stress protection circuitry, which prevents deselected channels from affecting the selected channel when current is injected into the deselected channels. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for specific current levels.

13.3.3 Sample Buffer Amplifier

The sample buffer is used to raise the effective input impedance of the A/D converter, so that external components (higher bandwidth or higher impedance) are less critical to accuracy. The input voltage is buffered onto the sample capacitor to reduce crosstalk between channels.

13.3.4 Digital to Analog Converter (DAC) Array

The digital to analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The reference voltages, V_{RH} and V_{RL}, are used by the DAC to perform ratiometric conversions. The DAC also converts the following three internal channels:

- V_{RH} — Reference voltage high
- V_{RL} — Reference voltage low
- $(V_{RH} – V_{RL})/2$ — Reference voltage

The DAC array serves to provide a mechanism for the successive approximation A/D conversion.

Resolution begins with the most significant bit (MSB) and works down to the least significant bit (LSB). The switching sequence is controlled by the comparator and successive-approximation register (SAR) logic.

- Sample Capacitor — The sample capacitor is employed to sample and hold the voltage to be converted.

13.3.5 Comparator

The comparator is used during the approximation process to sense whether the digitally selected arrangement of the DAC array produces a voltage level higher or lower than the sampled input. The comparator output feeds into the SAR which accumulates the A/D conversion result sequentially, beginning with the MSB.

13.3.6 Bias

The bias circuit is controlled by the STOP signal to power-up and power-down all the analog circuits.

13.3.7 Successive Approximation Register

The input of the successive approximation register (SAR) is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the 10 bits of the conversion result, the SAR data is transferred to the appropriate result location, where it may be read from the IMB by user software.

13.3.8 State Machine

The state machine receives the QCLK, RST, STOP, IST, CHAN[6:0], and START CONV. signals, from which it generates all timing to perform an A/D conversion. The start convert (start conv.) signal indicates to the A/D converter that the desired channel has been sent to the MUX. IST indicates the desired sample time. BYP indicates whether to bypass the sample amplifier. The end of conversion (EOC) signal, notifies the queue control logic that a result is available for storage in the result RAM.

13.4 Digital Subsystem

The digital control subsystem includes the control logic to sequence the conversion activity, the clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of the QADC64E conversions is the 64-entry CCW table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, sub queues can be created in the two queues. Each queue can be operated using one of several different scan modes. The scan modes for queue 1 and queue 2 are programmed in QACR1 and QACR2 (control registers 1 and 2). Once a queue has been started by a trigger event (any of the ways to cause the QADC64E to begin executing the CCWs

in a queue or sub-queue), the QADC64E performs a sequence of conversions and places the results in the result word table.

13.4.1 Queue Priority

Queue 1 has priority over queue 2 execution. The following cases show the conditions under which queue 1 asserts its priority:

- When a queue is not active, a trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
- When queue 1 is active and a trigger event occurs for queue 2, queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are captured as trigger overruns.
- When queue 2 is active and a trigger event occurs for queue 1, the current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while queue 2 is suspended are captured as trigger overruns. Once queue 1 reaches the completion or the paused state, queue 2 begins executing again. The programming of the RESUME bit in QACR2 determines which CCW is executed in queue 2. Refer to **13.2.7 Control Register 2** for more information.
- When simultaneous trigger events occur for queue 1 and queue 2, queue 1 begins execution and the queue 2 status is changed to trigger pending.

13.4.2 Sub-Queues That are Paused

The pause feature can be used to divide queue 1 and/or queue 2 into multiple sub-queues. A sub-queue is defined by setting the pause bit in the last CCW of the sub-queue.

Figure 13-10 shows the CCW format and an example of using pause to create sub-queues. Queue 1 is shown with four CCWs in each sub-queue and queue 2 has two CCWs in each sub-queue.

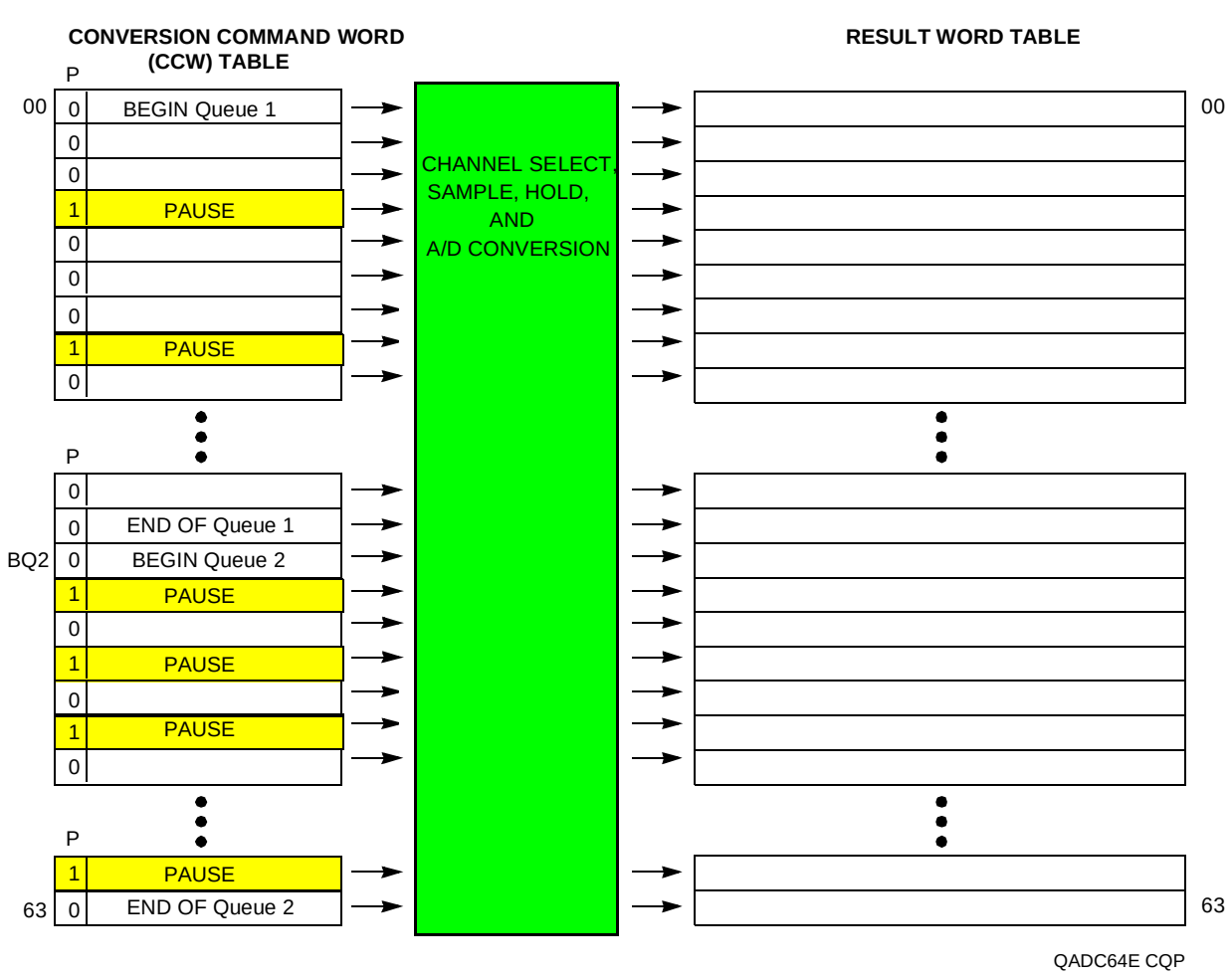


Figure 13-10 QADC64E Queue Operation With Pause

The queue operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the sub-queues within queue 1. Similarly, the queue operating mode for queue 2 determines the type of trigger event required to execute each of the sub-queues within queue 2.

EXAMPLE

When the external trigger rising edge continuous-scan mode is selected for queue 1, and there are six sub-queues within queue 1, a separate rising edge is required on the external trigger pin after every pause to begin the execution of each sub-queue (refer to **Figure 13-10**). Refer to **13.4.4 Scan Modes** for information on different scan modes.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each sub-queue. Once a sub-queue is initiated, each CCW is executed sequentially until the last CCW in the sub-queue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the

next sub-queue. A sub-queue cannot be executed a second time before the overall queue execution has been completed. Refer to **13.2.7 Control Register 2** for more information.

Trigger events which occur during the execution of a sub-queue are ignored, except that the trigger overrun flag is set. When a continuous-scan mode is selected, a trigger event occurring after the completion of the last sub-queue (after the queue completion flag is set), causes the execution to continue with the first sub-queue, starting with the first CCW in the queue.

When the QADC64E encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set and a pause software interrupt may optionally be issued. The status of the queue is shown to be paused, indicating completion of a sub-queue. The QADC64E then waits for another trigger event to again begin execution of the next sub-queue.

13.4.3 Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63 or 127, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set at the end of the CCW table (63 or 127) and a trigger event occurs on queue 2. Refer to **13.2.7 Control Register 2** for more information on BQ2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 is set beyond the end of the CCW table (64 – 127) and a trigger event occurs on queue 2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in the QADC64E behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC64E reads CCW0 and detects both end-of-queue conditions. The completion flag is set and queue 1 becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, since the end-of-queue condition is recognized, the com-

pletion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 or 127 (EOQ) code is in CCW6
- The pause is in CCW63
- During queue 1 operation, the pause bit is set in CCW20 and BQ2 points to CCW21

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC64E sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW10 and EOQ is programmed into CCW10
- During queue 1 operation, the pause bit set in CCW32, which is also BQ2

13.4.4 Scan Modes

The QADC64E queuing mechanism allows the application to utilize different requirements for automatically scanning input channels.

In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed. The possible modes are:

- Disabled and reserved mode
- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode
- Interval timer single-scan mode
- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode
- Periodic timer continuous-scan mode

The following paragraphs describe single-scan and continuous-scan operations.

13.4.4.1 Disabled Mode

When the disabled mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, wait states are not encountered for IMB accesses of the RAM. When both queues are disabled, it is safe to change the QCLK prescaler values.

13.4.4.2 Reserved Mode

Reserved mode allows for future mode definitions. When the reserved mode is selected, the queue is not active. It functions the same as disabled mode.

CAUTION

Do not use a reserved mode. Unspecified operations may result.

13.4.4.3 Single-Scan Modes

When the application software wants to execute a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected. By programming the MQ field in QACR1 or QACR2, the following modes can be selected:

- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode
- Interval timer single-scan mode

NOTE

Queue 2 cannot be programmed for external gated single-scan mode.

In all single-scan queue operating modes, the software must also enable the queue to begin execution by writing the single-scan enable bit to a one in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2 respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a one during the write cycle, which selects the single-scan queue operating mode. The single-scan enable bit is set through software, but will always read as a zero. Once set, writing the single-scan enable bit to zero has no effect. Only the QADC64E can clear the single-scan enable bit. The completion flag, completion interrupt, or queue status are used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC64E to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC64E resets the single-scan enable bit to zero. If the single-scan enable bit is written to a one or a zero by the software before the queue scan is complete, the queue is not affected. However, if the software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The conversion in progress is aborted and the new queue operating mode takes effect.

In the software-initiated single-scan mode, the writing of a one to the single-scan enable bit causes the QADC64E to internally generate a trigger event and the queue execution begins immediately. In the other single-scan queue operating modes, once

the single-scan enable bit is written, the selected trigger event must occur before the queue can start. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in an edge-sensitive external trigger mode or a periodic/interval timer mode.

In the periodic/interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After the queue execution is complete, the queue status is shown as idle. The software can restart the queue by setting the single-scan enable bit to a one. Queue execution begins with the first CCW in the queue.

13.4.4.4 Software Initiated Single-Scan Mode

Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting the software initiated single-scan mode, and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC64E immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC64E automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the software again sets the single-scan enable bit. While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused. The trigger overrun flag is never set while in the software initiated single-scan mode.

The software initiated single-scan mode is useful in the following applications:

- Allows software complete control of the queue execution
- Allows the software to easily alternate between several queue sequences.

13.4.4.5 External Trigger Single-Scan Mode

The external trigger single-scan mode is available on both queue 1 and queue 2. The software programs the polarity of the external trigger edge that is to be detected, either a rising or a falling edge. The software must enable the scan to occur by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC64E clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of the queue to be initiated by the next external trigger edge.

The external trigger single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though the software is not interested in data taken from every edge. The software can start the external trigger single-scan mode and get one set of data, and at a later time, start the queue again for the next set of samples.

When a pause bit is encountered during external trigger single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

13.4.4.6 External Gated Single-Scan Mode

The QADC64E provides external gating for queue 1 only. When external gated single-scan mode is selected, the input level on the associated external trigger pin enables and disables queue execution. The polarity of the external gated signal is fixed so only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Software must enable the scan to occur by setting the single-scan enable bit for queue 1. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC64E sets the completion flag (CF1) and clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. Software can read the CWPQ1 to determine the last valid conversion in the queue. Software must set the single-scan enable bit again and should clear the PF1 bit before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Since the condition of the gate is only sampled after each conversion during queue execution, closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

13.4.4.7 Periodic/Interval Timer Single-Scan Mode

Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128- to 128-Kbyte QCLK cycles in binary multiples. When the periodic/ interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR1 or QACR2 , the timer begins counting. When the time interval elapses, an internal trigger event is created to start the queue and the QADC64E begins execution with the first CCW.

The QADC64E automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval elapses again, and then queue execution continues. When the queue execution reaches an end-of-queue situation, the single-scan enable bit is cleared. Software may set the single-scan enable bit again, allowing another scan of the queue to be initiated by the periodic/interval timer.

The periodic/interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a

pause, or may be considered a trigger overrun. Once the queue execution is completed, the single-scan enable bit must be set again to enable the timer to count again.

Normally only one queue will be enabled for periodic/interval timer single-scan mode and the timer will reset at the end-of-queue. However, if both queues are enabled for either single-scan or continuous periodic/interval timer mode, the end-of-queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end-of-queue. See **13.4.6 Periodic/Interval Timer** for a definition of periodic/interval timer reset conditions.

The periodic/interval timer single-scan mode can be used in applications which need coherent results, for example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog pins
- When the interrupt rate in the periodic/interval timer continuous-scan mode would be too high
- In sensitive battery applications, where the single-scan mode uses less power than the software initiated continuous-scan mode

13.4.4.8 Continuous-Scan Modes

When the application software wants to execute multiple passes through a sequence of conversions defined by a queue, a continuous-scan queue operating mode is selected. By programming the MQ1 field in QACR1 or the MQ2 field in QACR2, the following software initiated modes can be selected:

- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode
- Periodic/interval timer continuous-scan mode

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of the software-initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic/interval timer continuous-scan mode.

After the queue execution is complete, the queue status is shown as idle. Since the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed to enable queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

Coherent samples are guaranteed. The time between consecutive conversions has been designed to be consistent. However, there is one exception. For queues that end with a CCW containing EOQ code (channel 63 or 127), the last queue conversion to the first queue conversion requires 1 additional CCW fetch cycle. Therefore continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion cannot be guaranteed since it is a function of clock synchronization, programmable trigger events, queue priorities, and so on.

13.4.4.9 Software Initiated Continuous-Scan Mode

When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC64E. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and then execution continues without pausing. When the end-of-queue is reached, another internal trigger event is generated, and queue execution begins again from the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is idle. The trigger overrun flag is never set while in the software-initiated continuous-scan mode.

The software initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The software can always read the result table to get the latest converted value for each channel. The channels scanned are kept up to date by the QADC64E without software involvement. Software can read a result value at any time.

The software initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When the software initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything, or for the slow-to-change analog channels. Interrupts are normally not used with the software initiated continuous-scan mode. Rather, the software reads the latest conversion result from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel.

13.4.4.10 External Trigger Continuous-Scan Mode

The QADC64E provides external trigger pins for both queues. When the external trigger software initiated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external trigger signal is programmable, so that the software can select a mode which begins queue execu-

tion on the rising or falling edge. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, the queue execution begins again automatically. Software initialization is not needed between trigger events.

When a pause bit is encountered in external trigger continuous-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, since interrupt response times vary.

13.4.4.11 External Gated Continuous-Scan Mode

The QADC64E provides external gating for queue 1 only. When external gated continuous-scan mode is selected, the input level on the associated external trigger pin enables and disables queue execution. The polarity of the external gated signal is fixed so a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, the queue execution automatically begins again from the beginning of the queue. Software initialization is not needed between trigger events. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

The purpose of external gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. It is up to the programmer to ensure that the queue is large enough so that a maximum gate open time will not reach an end-of-queue. However it is useful to take advantage of a smaller queue in the manner described in the next paragraph.

In the event that the queue completes before the gate closes, a completion flag will be set and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the completion flag is not cleared, when the queue completes a second time the trigger overrun flag will be set and the queue will roll-over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the current CCW completes execution of queue 1 stops and QADC64E sets the PF1 bit to indicate an incomplete queue. Software can read the CWPQ1 to determine the last valid conversion in the queue. In this mode, if the gate opens again, execution of queue 1 begins again. The start of queue 1 is always the first CCW in the CCW table.

Since the condition of the gate is only sampled after each conversion during queue execution, closing the gate for a period less than a conversion time interval does not guarantee the closure will be captured.

13.4.4.12 Periodic/Interval Timer Continuous-Scan Mode

The QADC64E includes a dedicated periodic/interval timer for initiating a scan sequence on queue 1 and/or queue 2. Software selects a programmable timer interval ranging from 128 to 128 Kbytes times the QCLK period in binary multiples. The QCLK period is prescaled down from the IMB MCU clock.

When a periodic/interval timer continuous-scan mode is selected for queue 1 and/or queue 2, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. Meanwhile, the QADC64E automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC64E waits for the periodic interval to expire again, then continues with the queue. Once end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in the queue.

The periodic/interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause or queue completion, or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between trigger events. Since both queues may be triggered by the periodic/interval timer, see **13.4.6 Periodic/Interval Timer** for a summary of periodic/interval timer reset conditions.

Software enables the completion interrupt when using the periodic/interval timer continuous-scan mode. When the interrupt occurs, the software knows that the periodically collected analog results have just been taken. The software can use the periodic interrupt to obtain non-analog inputs as well, such as contact closures, as part of a periodic look at all inputs.

13.4.5 QADC64E Clock (QCLK) Generation

Figure 13-11 is a block diagram of the clock subsystem. The QCLK provides the timing for the A/D converter state machine which controls the timing of the conversion. The QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (F_{QCLK}) must be within the tolerance specified in **APPENDIX E ELECTRICAL CHARACTERISTICS**.

Before using the QADC64E, the software must initialize the prescaler with values that put the QCLK within the specified range. Though most software applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

CAUTION

A change in the prescaler value while a conversion is in progress is likely to corrupt the result from any conversion in progress. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

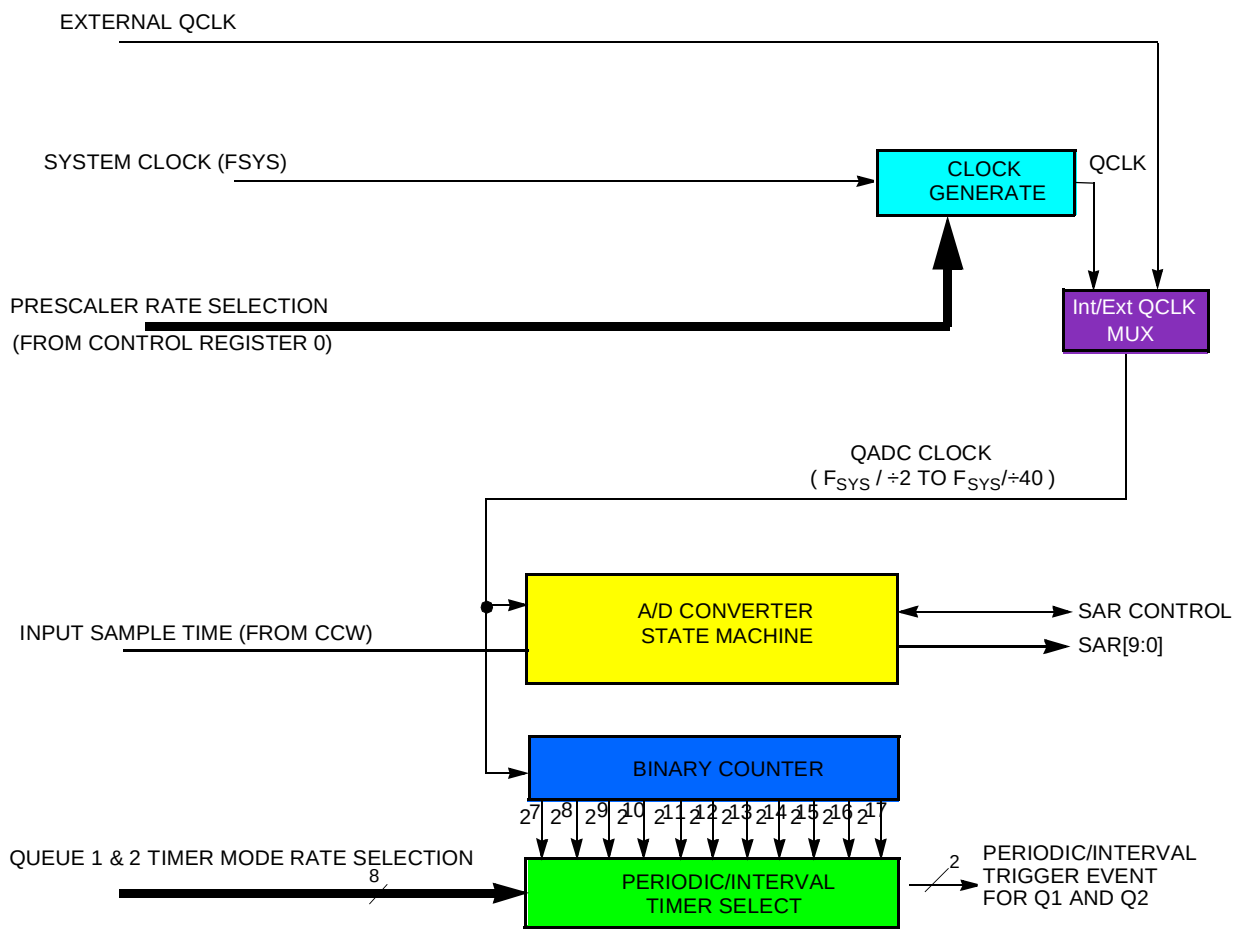


Figure 13-11 QADC64E Clock Subsystem Functions

To accommodate wide variations of the main MCU clock frequency (IMB system clock – F_{SYS}), QCLK is generated by a programmable prescaler which divides the MCU system clock. To allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC64E prescaler permits the frequency of QCLK to be software selectable. The software establishes the frequency of the QCLK waveform by setting the PRESCALAR field in QACR0. The frequency resulting from a value in the PRESCALAR field that is > 0, is calculated by the following formula:

$$F_{QCLK} = F_{SYSCLK} / (PRESCALAR + 1)$$

if the value in the PRESCALAR field is set to 0, the resulting QCLK frequency may be calculated to be:

$$F_{QCLK} = F_{SYSCLK} / 2$$

Table 13-21 QADC64E Clock Programmability

Control Register 0 Information			Input Sample Time (IST) =%00	
Example Number	Frequency	PRESCALER	QCLK (MHz)	Conversion Time (µs)
1	20 MHz	0x09	2.0	7.0
2	40 MHz	0x13	2.0	7.0
3	56 MHz	0x1B	2.0	7.0

The MCU system clock frequency is the basis of the QADC64E timing. The QADC64E requires that the system clock frequency be at least twice the QCLK frequency. The QCLK frequency is established by the prescaler parameter in QACR0.

13.4.6 Periodic/Interval Timer

The on-chip periodic/interval timer can be used to generate trigger events at a programmable interval, initiating execution of queue 1 and/or queue 2. The periodic/interval timer stays reset under the following conditions:

- Both queue 1 and queue 2 are programmed to any mode which does not use the periodic/interval timer
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

NOTE

Interval timer single-scan mode does not use the periodic/interval timer until the single-scan enable bit is set.

The following two conditions will cause a pulsed reset of the periodic/interval timer during use:

- A queue 1 operating mode change to a mode which uses the periodic/interval timer, even if queue 2 is already using the timer
- A queue 2 operating mode change to a mode which uses the periodic/interval timer, provided queue 1 is not in a mode which uses the periodic/interval timer
- Roll over of the timer

During the low power stop mode, the periodic timer is held in reset. Since low power stop mode causes QACR1 and QACR2 to be reset to zero, a valid periodic or interval timer mode must be written after stop mode is exited to release the timer from reset.

When the IMB internal FREEZE line is asserted and a periodic or interval timer mode is selected, the timer counter is reset after the conversion in progress completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, the freeze mode takes effect immediately, and the timer is held in reset. When the internal FREEZE line is negated, the timer counter

starts counting from the beginning. Refer to **13.4.7 Configuration And Control using the IMB Interface** for more information.

13.4.7 Configuration And Control using the IMB Interface

The QADC64E module communicates with other microcontroller modules via the IMB. The QADC64E bus interface unit (BIU) coordinates IMB activity with internal QADC64E bus activity. This section describes the operation of the BIU, IMB read/write accesses to QADC64E memory locations, module configuration, and general-purpose I/O operation.

13.4.7.1 QADC64E Bus Interface Unit

The BIU is designed to act as a slave device on the IMB. The BIU has the following functions: to respond with the appropriate bus cycle termination, and to supply IMB interface timing to all internal module signals.

BIU components consist of

- IMB buffers
- Address match and module select logic
- The BIU state machine
- Clock prescaler logic
- Data bus routing logic
- Interface to the internal module data bus

NOTE

Normal accesses from the IMB to the QADC64E require two clocks. However, if the CPU tries to access table locations while the QADC64E is accessing them, the QADC64E produces IMB wait states. From one to four IMB wait states may be inserted by the QADC64E in the process of reading and writing.

13.4.7.2 QADC64E Bus Accessing

The QADC64E supports 8-bit, 16-bit, and 32-bit data transfers, at even and odd addresses. Coherency of results read (ensuring that all results read were taken consecutively in one scan) is not guaranteed. For example, if a read of two consecutive 16-bit locations in a result area is made, the QADC64E could change one 16-bit location in the result area between the bus cycles. There is no holding register for the second 16-bit location. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. Depending on bus master protocol, these accesses could include misaligned and 32-bit accesses.

Figure 13-12 shows the three bus cycles which are implemented by the QADC64E. The following paragraphs describe how the three types of accesses are used, including misaligned 16-bit and 32-bit accesses.

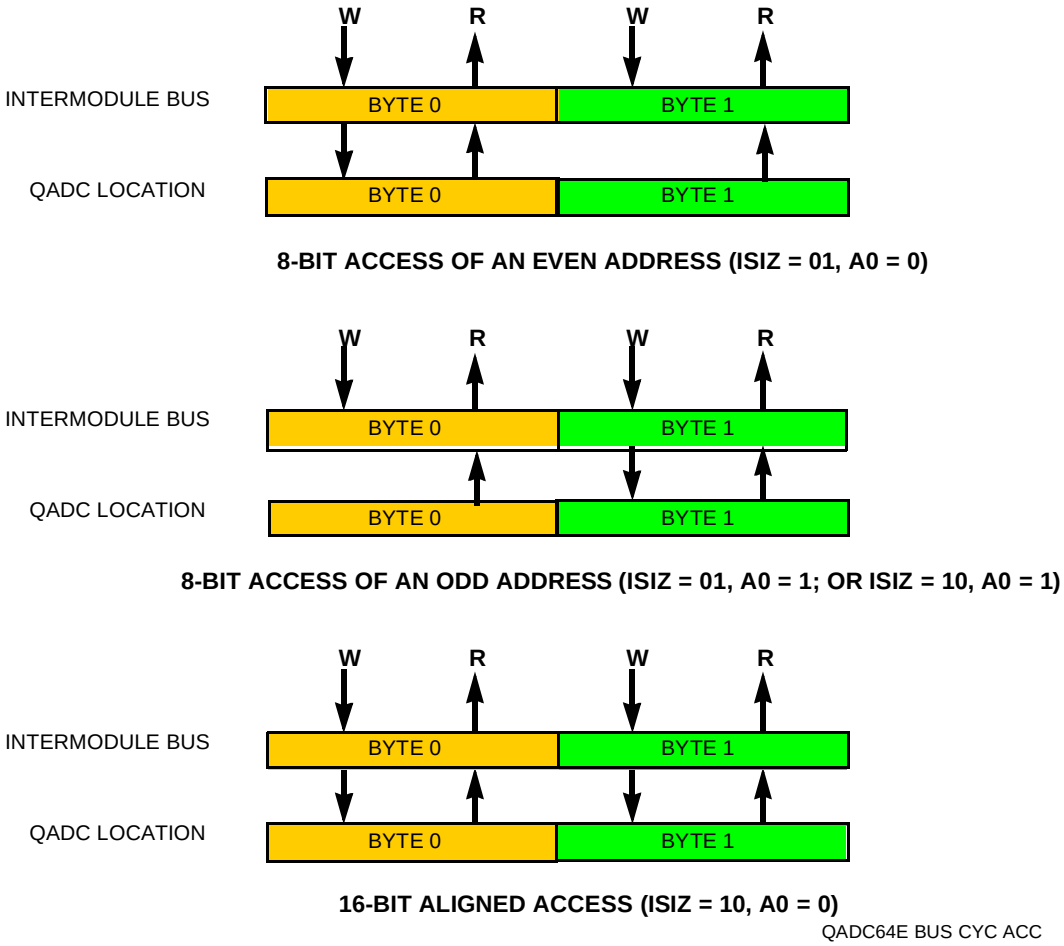


Figure 13-12 Bus Cycle Accesses

Byte access to an even address of a QADC64E location is shown in the top illustration of **Figure 13-12**. In the case of write cycles, byte1 of the register is not disturbed. In the case of a read cycle, the QADC64E provides both byte 0 and byte1.

Byte access to an odd address of a QADC64E location is shown in the center illustration of **Figure 13-12**. In the case of write cycles, byte 0 of the register is not disturbed. In the case of read cycles, the QADC64E provides both byte 0 and byte1.

16-bit accesses to an even address read or write byte 0 and byte 1 as shown in the lowest illustration of **Figure 13-12**. The full 16 bits of data is written to and read from the QADC64E location with each access.

16-bit accesses to an odd address require two bus cycles; one byte of two different 16-bit QADC64E locations is accessed. The first bus cycle is treated by the QADC64E as an 8-bit read or write of an odd address. The second cycle is an 8-bit read or write of an even address. The QADC64E address space is organized into 16-bit even address locations, so a 16-bit read or write of an odd address obtains or provides the lower half of one QADC64E location, and the upper half of the following QADC64E location.

32-bit accesses to an even address require two bus cycles to complete the access, and two full 16-bit QADC64E locations are accessed. The first bus cycle reads or writes the addressed 16-bit QADC64E location and the second cycle reads or writes the following 16-bit location.

32-bit accesses to an odd address require three bus cycles. Portions of three different QADC64E locations are accessed. The first bus cycle is treated by the QADC64E as an 8-bit access of an odd address, the second cycle is a 16-bit aligned access, and the third cycle is an 8-bit access of an even address. The QADC64E address space is organized into 16-bit even address locations, so a 32-bit read or write of an odd address provides the lower half of one QADC64E location, the full 16-bit content of the following QADC64E location, and the upper half of the third QADC64E location.

13.5 Trigger and Queue Interaction Examples

This section contains examples describing queue priority and conversion timing schemes.

13.5.1 Queue Priority Schemes

Since there are two conversion command queues and only one A/D converter, there is a priority scheme to determine which conversion is to occur. Each queue has a variety of trigger events that are intended to initiate conversions, and they can occur asynchronously in relation to each other and other conversions in progress. For example, a queue can be idle awaiting a trigger event, a trigger event can have occurred but the first conversion has not started, a conversion can be in progress, a pause condition can exist awaiting another trigger event to continue the queue, and so on.

The following paragraphs and figures outline the prioritizing criteria used to determine which conversion occurs in each overlap situation.

NOTE

The situations in **Figure 13-13** through **Figure 13-31** are labeled S1 through S19. In each diagram, time is shown increasing from left to right. The execution of queue 1 and queue 2 (Q1 and Q2) is shown as a string of rectangles representing the execution time of each CCW in the queue. In most of the situations, there are four CCWs (labeled C1 to C4) in both queue 1 and queue 2. In some of the situations, CCW C2 is presumed to have the pause bit set, to show the similarities of pause and end-of-queue as terminations of queue execution.

Trigger events are described in **Table 13-22**.

Table 13-22 Trigger Events

Trigger	Events
T1	Events that trigger queue 1 execution (external trigger, software initiated single-scan enable bit, or completion of the previous continuous loop)
T2	Events that trigger queue 2 execution (external trigger, software initiated single-scan enable bit, timer period/interval expired, or completion of the previous continuous loop)

When a trigger event causes a CCW execution in progress to be aborted, the aborted conversion is shown as a ragged end of a shortened CCW rectangle.

The situation diagrams also show when key status bits are set. **Table 13-23** describes the status bits.

Table 13-23 Status Bits

Bit	Function
CF Flag	Set when the end of the queue is reached
PF Flag	Set when a queue completes execution up through a pause bit
Trigger Overrun Error (TOR)	Set when a new trigger event occurs before the queue is finished serving the previous trigger event

Below the queue execution flows are three sets of blocks that show the status information that is made available to the software. The first two rows of status blocks show the condition of each queue as:

- Idle
- Active
- Pause
- Suspended (queue 2 only)
- Trigger pending

The third row of status blocks shows the 4-bit QS status register field that encodes the condition of the two queues. Two transition status cases, QS = 0011 and QS = 0111, are not shown because they exist only very briefly between stable status conditions.

The first three examples in **Figure 13-13** through **Figure 13-15** (S1, S2, and S3) show what happens when a new trigger event is recognized before the queue has completed servicing the previous trigger event on the same queue.

In situation S1 (**Figure 13-13**), one trigger event is being recognized on each queue while that queue is still working on the previously recognized trigger event. The trigger overrun error status bit is set, and otherwise, the premature trigger event is ignored. A

trigger event which occurs before the servicing of the previous trigger event is through does not disturb the queue execution in progress.

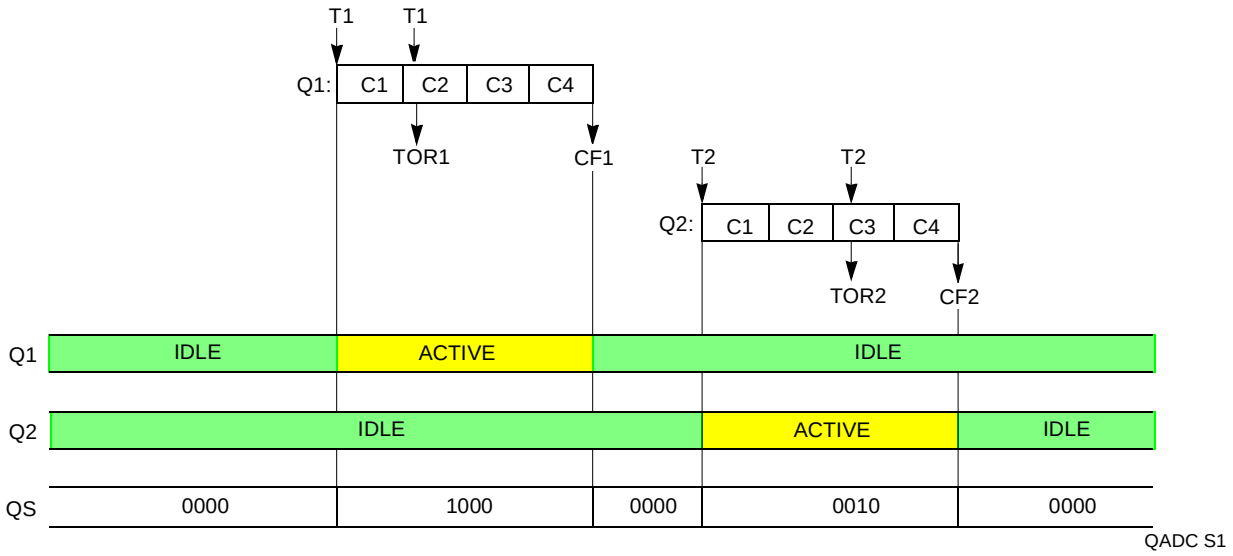


Figure 13-13 CCW Priority Situation 1

In situation S2 (**Figure 13-13**), more than one trigger event is recognized before servicing of a previous trigger event is complete, the trigger overrun bit is again set, but otherwise, the additional trigger events are ignored. After the queue is complete, the first newly detected trigger event causes queue execution to begin again. When the trigger event rate is high, a new trigger event can be seen very soon after completion of the previous queue, leaving software little time to retrieve the previous results. Also, when trigger events are occurring at a high rate for queue 1, the lower priority queue 2 channels may not get serviced at all.

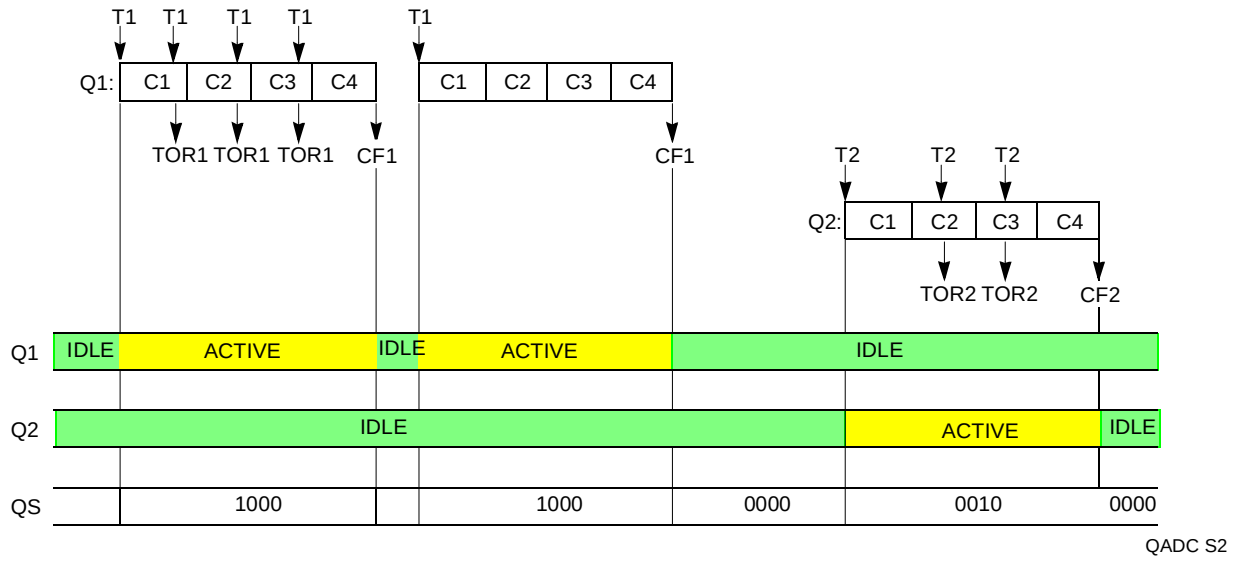


Figure 13-14 CCW Priority Situation 2

Situation S3 (**Figure 13-14**) shows that when the pause feature is in use, the trigger overrun error status bit is set the same way, and that queue execution continues unchanged.

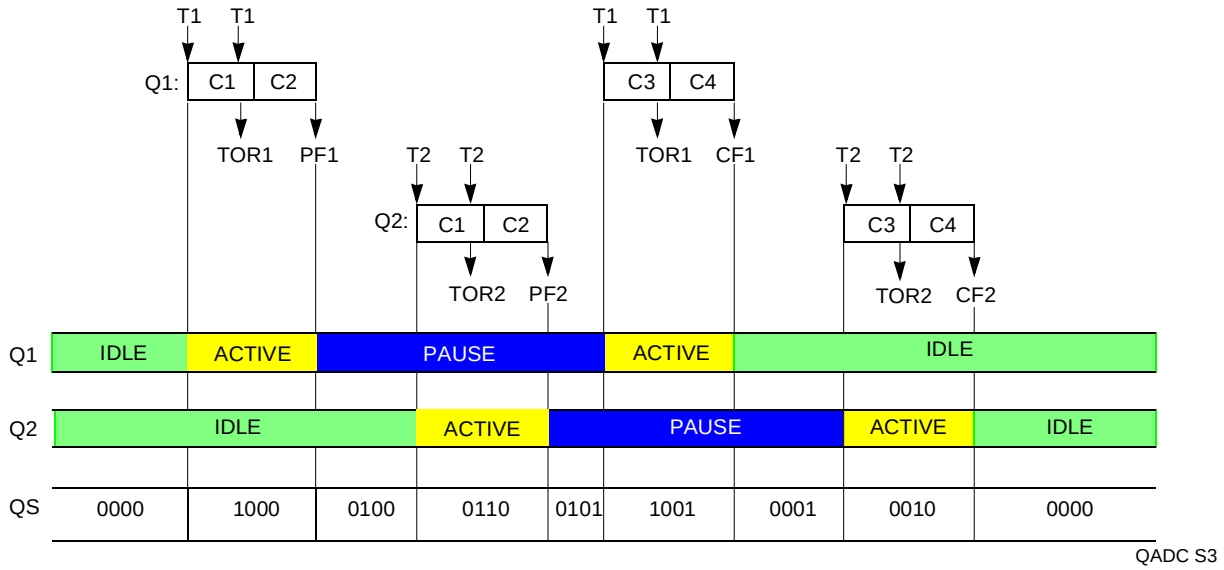


Figure 13-15 CCW Priority Situation 3

The next two situations consider trigger events that occur for the lower priority queue 2, while queue 1 is actively being serviced.

Situation S4 (**Figure 13-16**) shows that a queue 2 trigger event that is recognized while queue 1 is active is saved, and as soon as queue 1 is finished, queue 2 servicing begins.

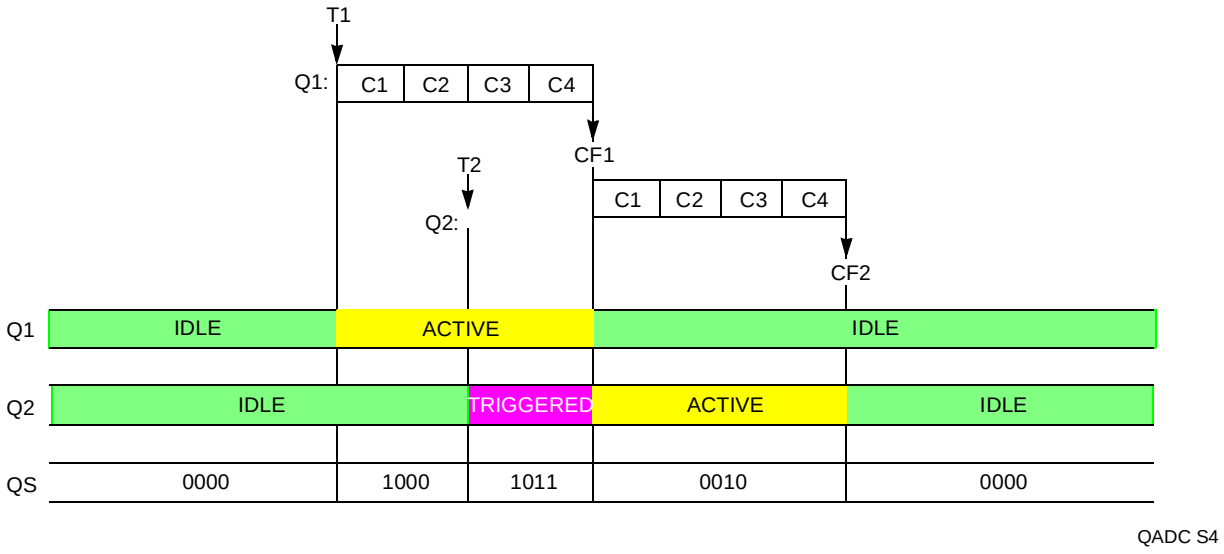


Figure 13-16 CCW Priority Situation 4

Situation S5 (**Figure 13-17**) shows that when multiple queue 2 trigger events are detected while queue 1 is busy, the trigger overrun error bit is set, but queue 1 execution is not disturbed. Situation S5 also shows that the effect of queue 2 trigger events during queue 1 execution is the same when the pause feature is in use in either queue.

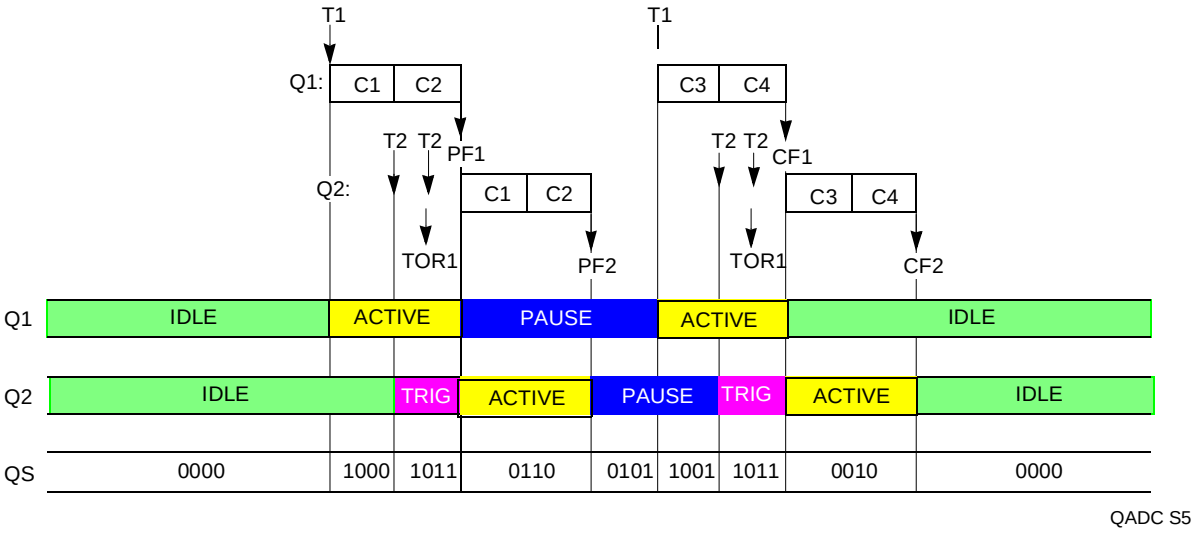


Figure 13-17 CCW Priority Situation 5

The remaining situations, S6 through S11, show the impact of a queue 1 trigger event occurring during queue 2 execution. Queue 1 is higher in priority the conversion taking place in queue 2 is aborted, so that there is not a variable latency time in responding to queue 1 trigger events.

In situation S6 (**Figure 13-18**), the conversion initiated by the second CCW in queue 2 is aborted just before the conversion is complete, so that queue 1 execution can begin. Queue 2 is considered suspended. After queue 1 is finished, queue 2 starts over with the first CCW, when the RES (resume) control bit is set to 0. Situation S7 (**Figure 13-19**) shows that when pause operation is not in use with queue 2, queue 2 suspension works the same way.

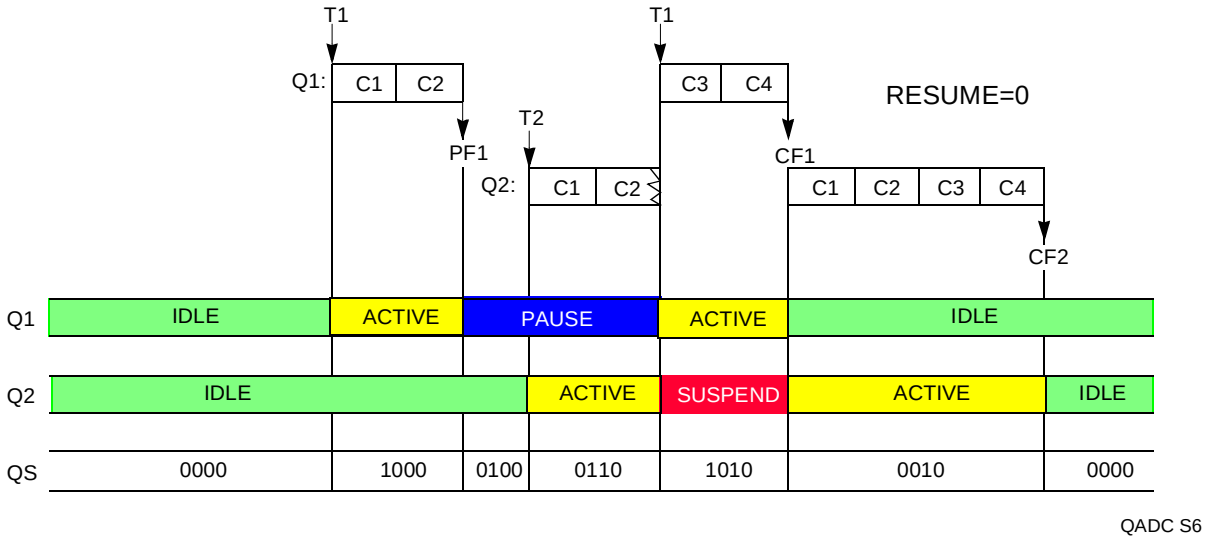


Figure 13-18 CCW Priority Situation 6

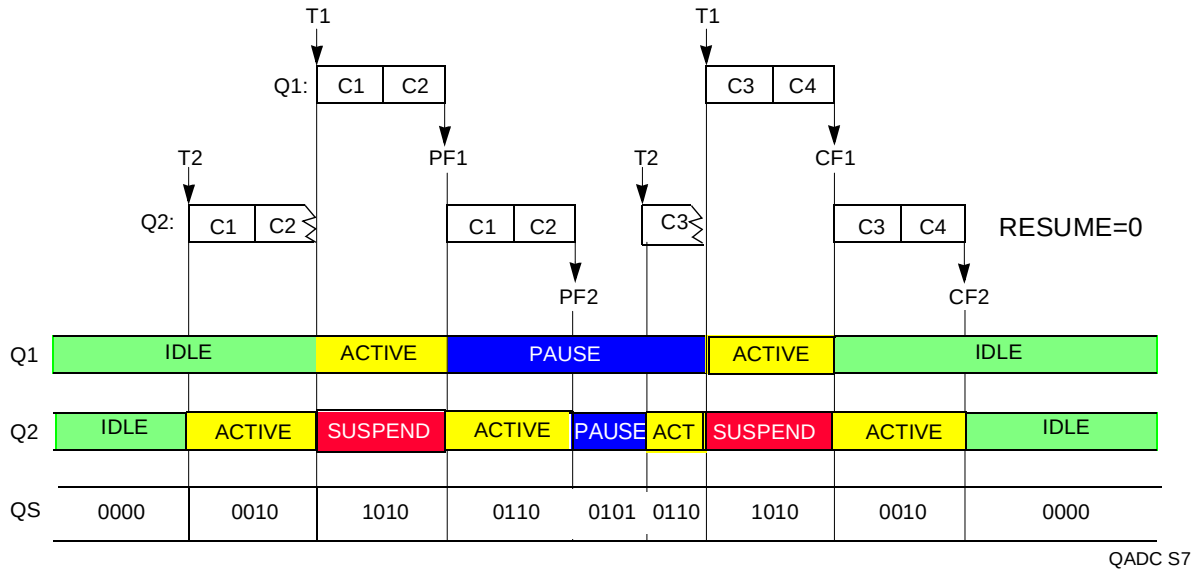


Figure 13-19 CCW Priority Situation 7

Situations S8 and S9 (**Figure 13-20** and **Figure 13-21**) repeat the same two situations with the resume bit set to a one. When the RES bit is set, following suspension, queue 2 resumes execution with the aborted CCW, not the first CCW in the queue.

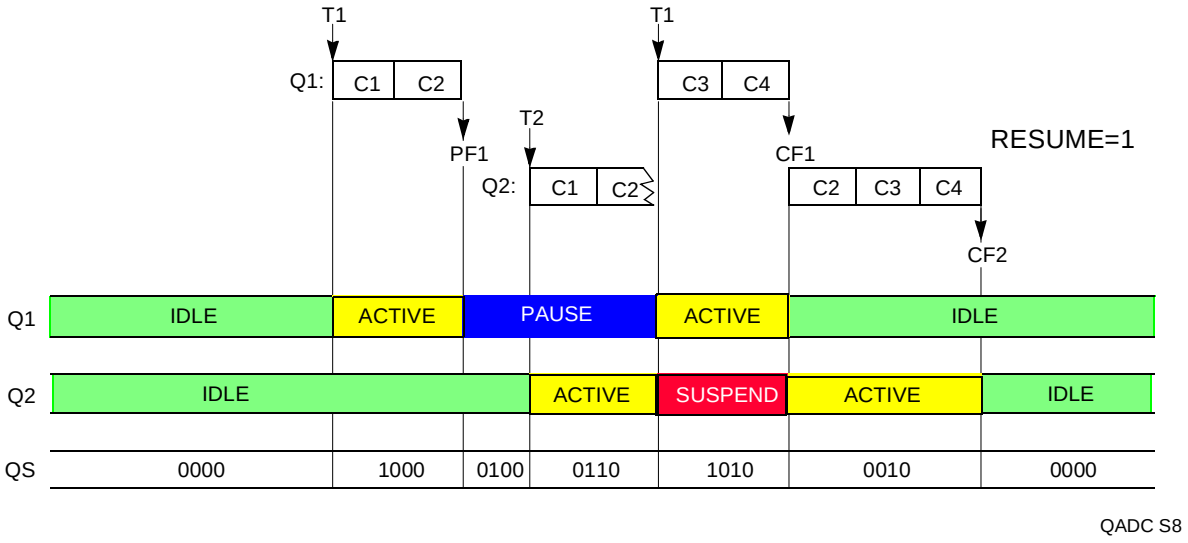


Figure 13-20 CCW Priority Situation 8

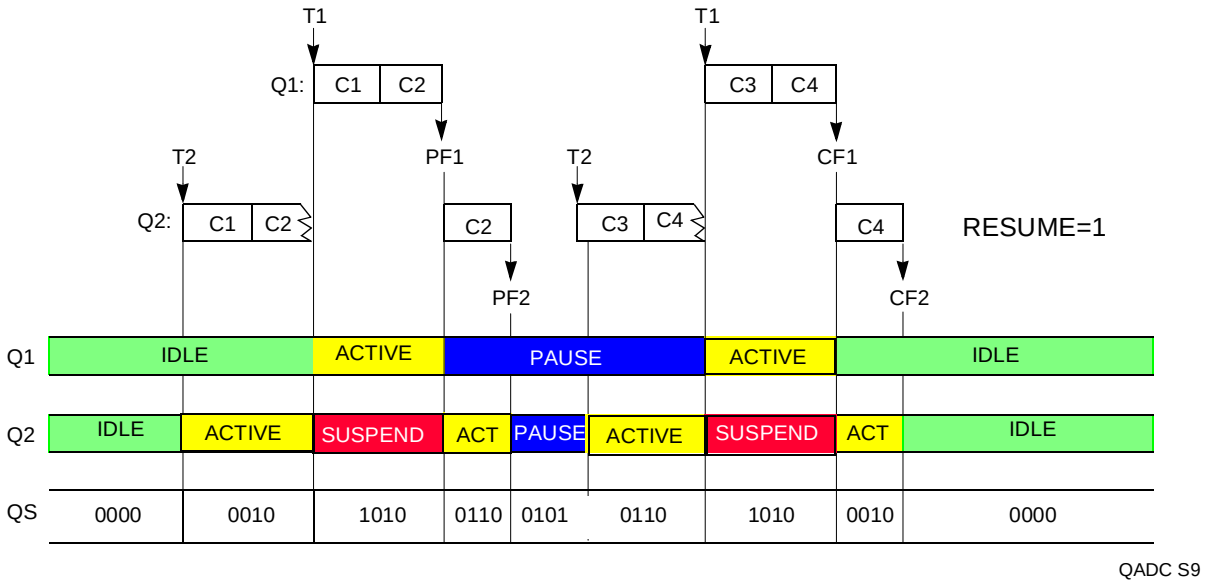


Figure 13-21 CCW Priority Situation 9

Situations S10 and S11 (Figures **Figure 13-22** and **Figure 13-23**) show that when an additional trigger event is detected for queue 2 while the queue is suspended, the trigger overrun error bit is set, the same as if queue 2 were being executed when a new trigger event occurs. Trigger overrun on queue 2 thus permits the software to know that queue 1 is taking up so much QADC64E time that queue 2 trigger events are being lost.

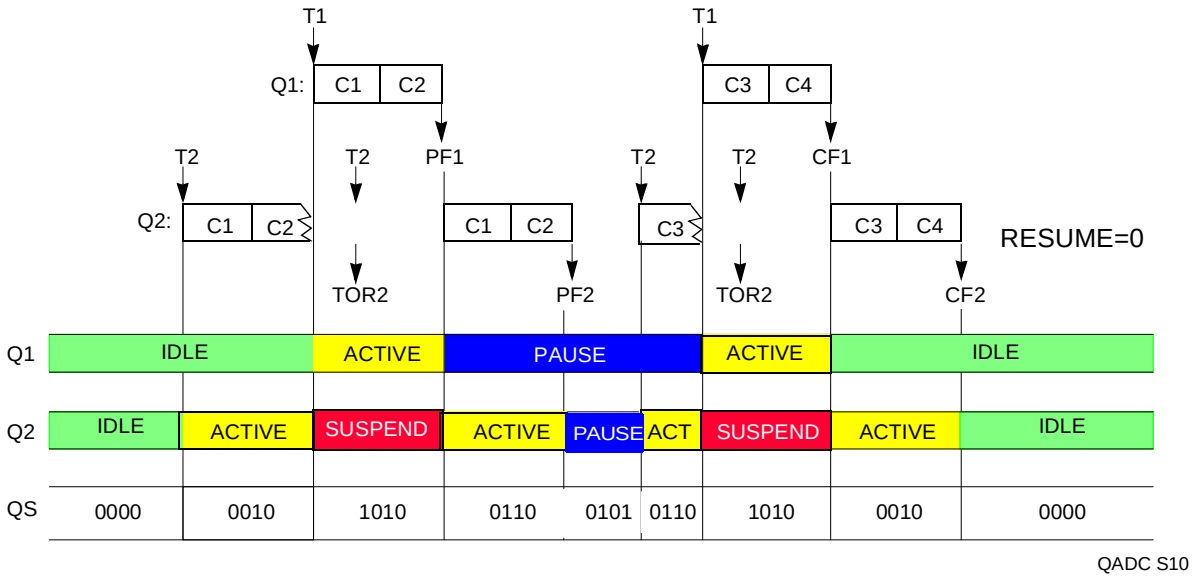


Figure 13-22 CCW Priority Situation 10

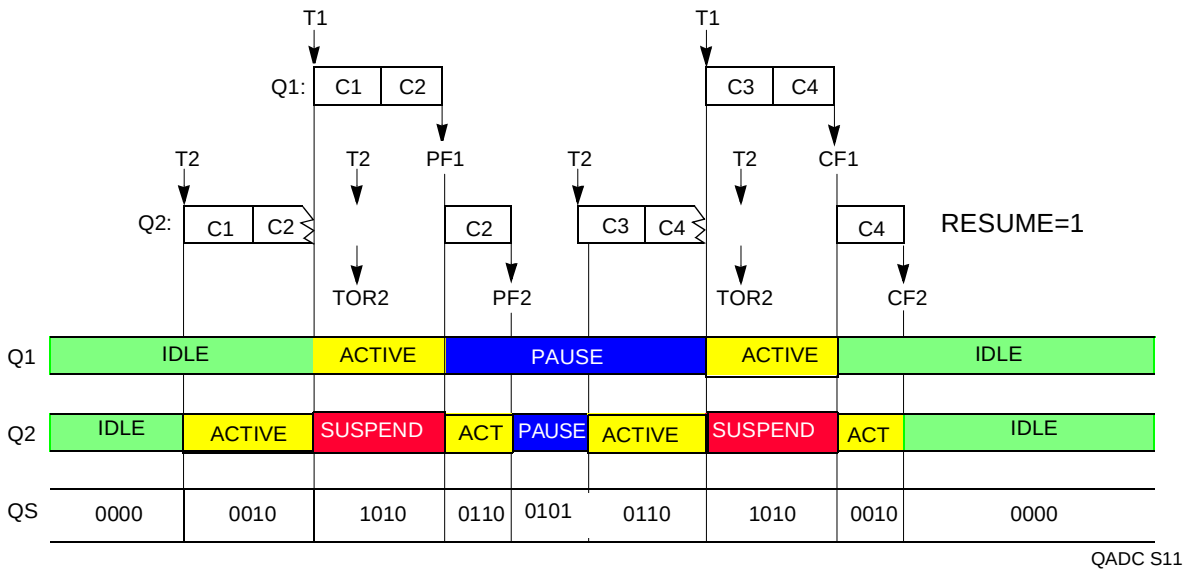


Figure 13-23 CCW Priority Situation 11

The above situations cover normal overlap conditions that arise with asynchronous trigger events on the two queues. An additional conflict to consider is that the freeze condition can arise while the QADC64E is actively executing CCWs. The conventional use for the freeze mode is for software/hardware debugging. When the CPU background debug mode is enabled and a breakpoint occurs, the freeze signal is issued, which can cause peripheral modules to stop operation. When freeze is detected, the QADC64E completes the conversion in progress, unlike queue 1 suspending queue 2. After the freeze condition is removed, the QADC64E continues queue execution with the next CCW in sequence.

Trigger events that occur during freeze are not captured. When a trigger event is pending for queue 2 before freeze begins, that trigger event is remembered when the freeze is passed. Similarly, when freeze occurs while queue 2 is suspended, after freeze, queue 2 resumes execution as soon as queue 1 is finished.

Situations 12 through 19 (**Figure 13-24** to **Figure 13-31**) show examples of all of the freeze situations.

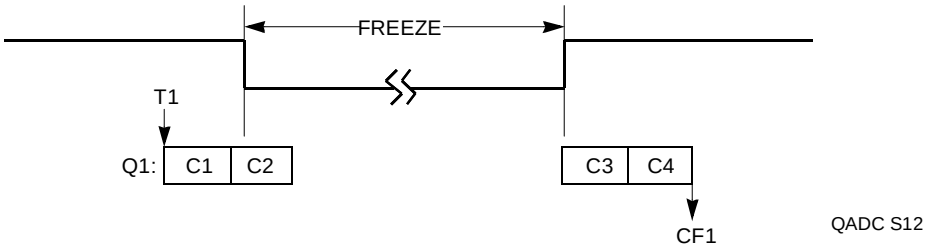


Figure 13-24 CCW Freeze Situation 12

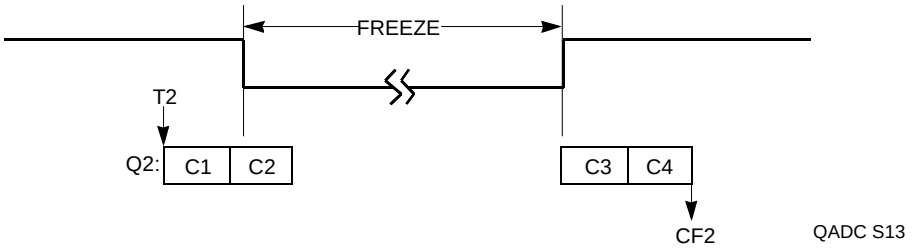


Figure 13-25 CCW Freeze Situation 13

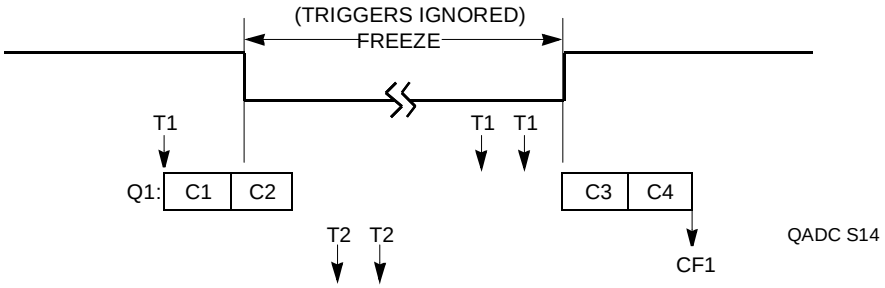


Figure 13-26 CCW Freeze Situation 14

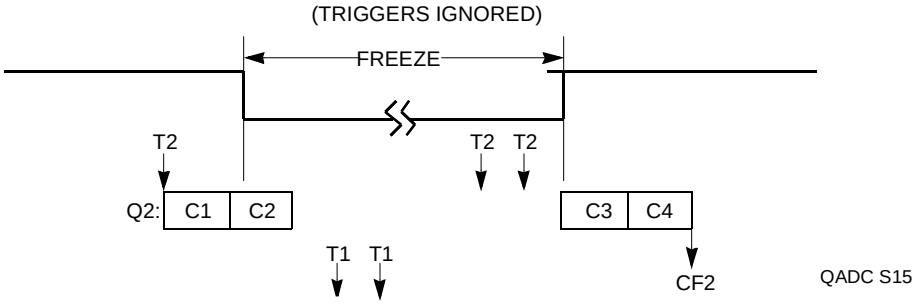


Figure 13-27 CCW Freeze Situation 15

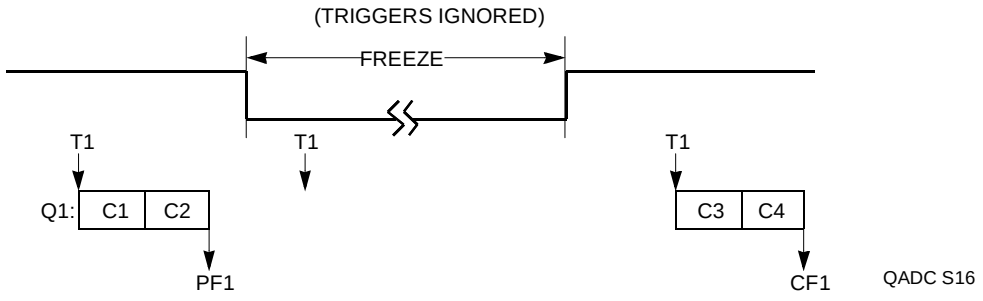


Figure 13-28 CCW Freeze Situation 16

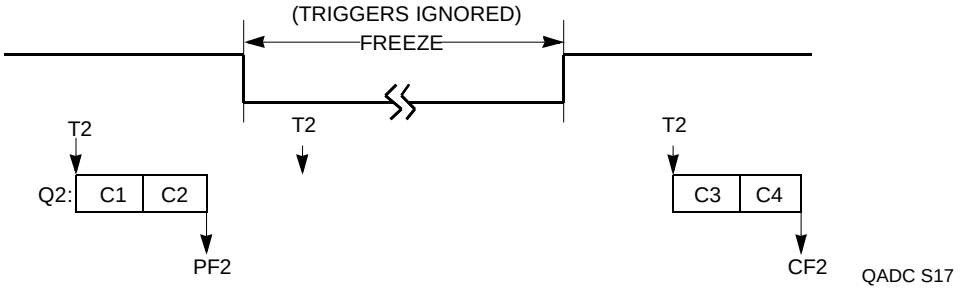


Figure 13-29 CCW Freeze Situation 17

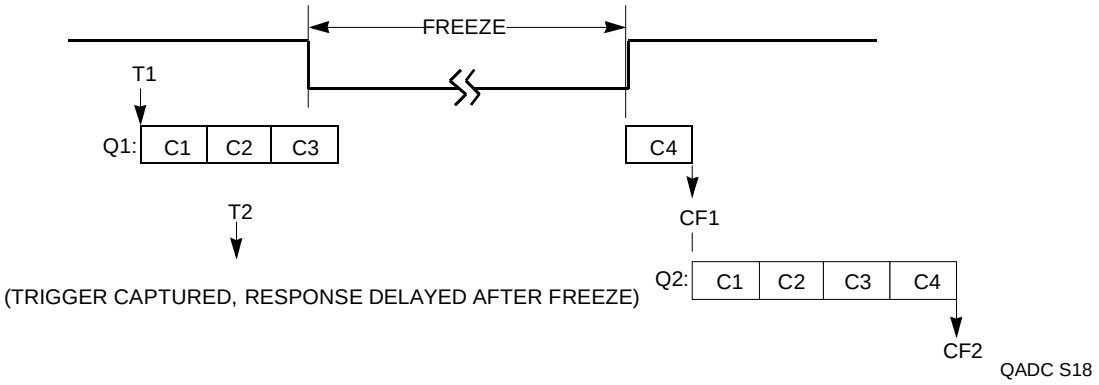


Figure 13-30 CCW Freeze Situation 18

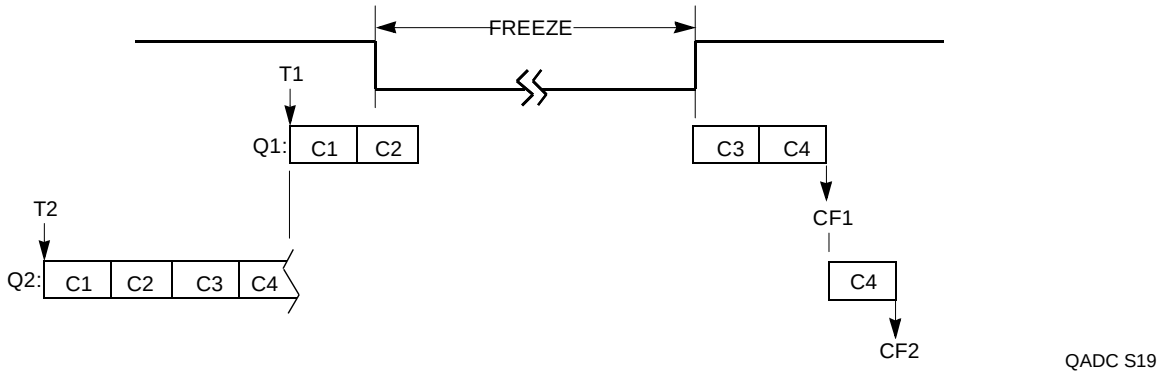


Figure 13-31 CCW Freeze Situation 19

13.5.2 Conversion Timing Schemes

This section contains some conversion timing examples. Example 1 below shows the timing for basic conversions where the following is assumed:

- Q1 begins with CCW0 and ends with CCW3
- CCW0 has pause bit set
- CCW1 does not have pause bit set
- External trigger rise-edge for Q1
- CCW4 = BQ2 and Q2 is disabled
- Q1 RES shows relative result register updates

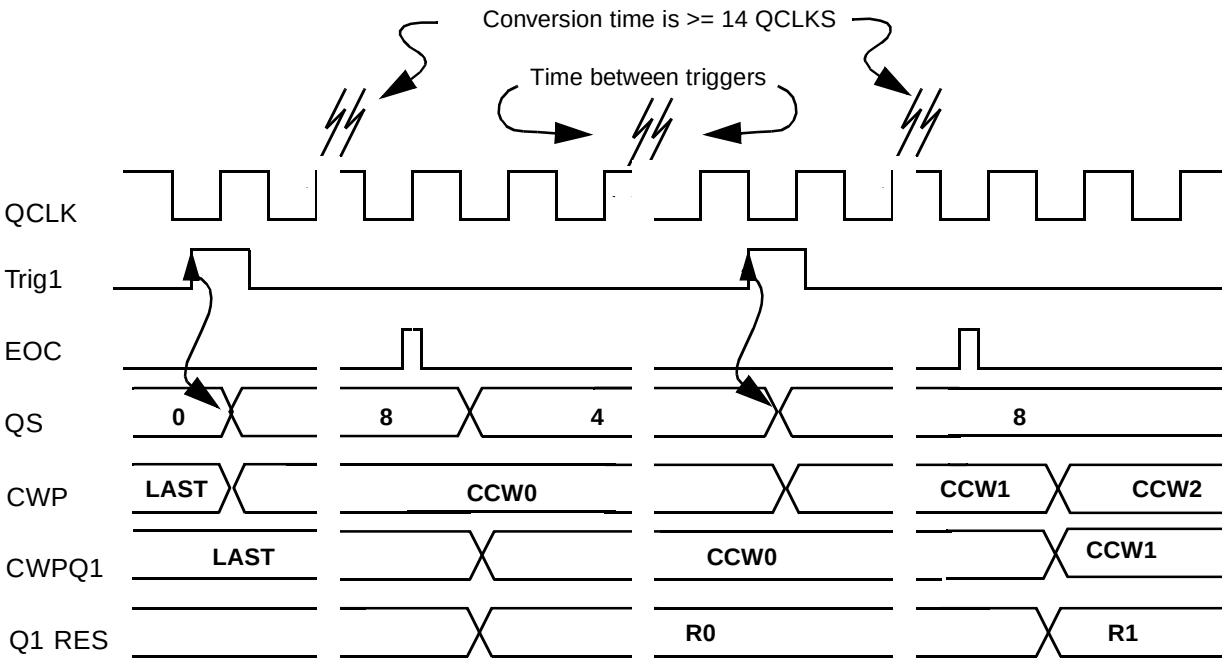


Figure 13-32 External Trigger Mode (Positive Edge) Timing With Pause

Recall QS = 0 => Queues disabled; QS = 8 => Q1 active, Q2 disabled; QS= 4 => Q1 paused, Q2 disabled.

A time separator was provided between the triggers and end of conversion (EOC). The relationship to QCLK displayed is not guaranteed.

CWPQ1 or CWPQ2 typically lag CWP and only match CWP when the associated queue is inactive. Another way to view CWPQ1 or CWPQ2 is that these registers update when EOC triggers the result register to be written.

When the pause bit is set (CCW0), please note that CWP does not increment until triggered. When the pause is not set (CCW1), the CWP increments with EOC.

The conversion results Q1 RES(x) show the result associated with CCW(x). So that R0 represents the result associated with CCW0.

Example 2 below shows the timing for conversions in gated mode single-scan with the same assumptions as example 1 except:

- No pause bits set in any CCW
- External trigger gated single-scan mode for Q1
- Single-scan bit is set

When the gate closes and opens again the conversions start with the first CCW in Q1.

When the gate closes the active conversion completes before the queue goes idle.

When Q1 completes both the CF1 bit sets and the SSE bit clears.

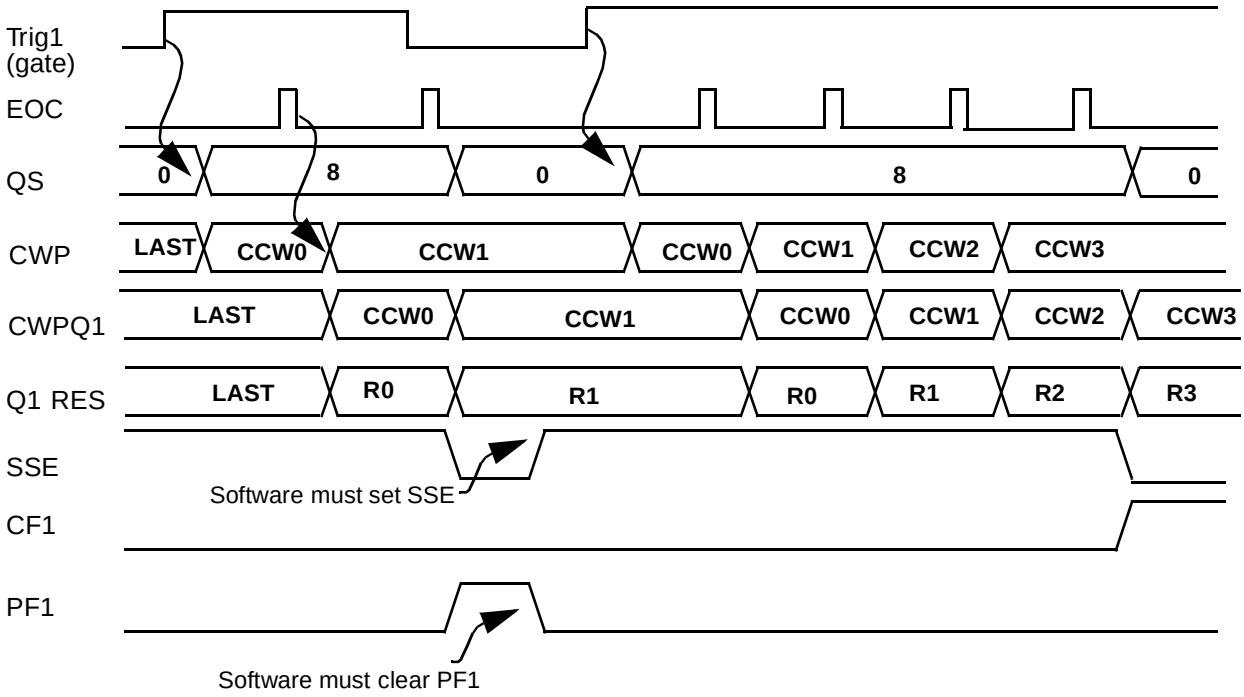


Figure 13-33 Gated Mode, Single-Scan Timing

Example 3 below shows the timing for conversions in gated continuous-scan mode with the same assumptions in the amended definition for the PF bit in this mode to reflect the condition that a gate closing occurred before the queue completed is a proposal under consideration at this time as example 2.

NOTE

At the end of Q1,the completion flag CF1 sets and the queue restarts. Also, note that if the queue starts a second time and completes, the trigger overrun flag TOR1 sets.

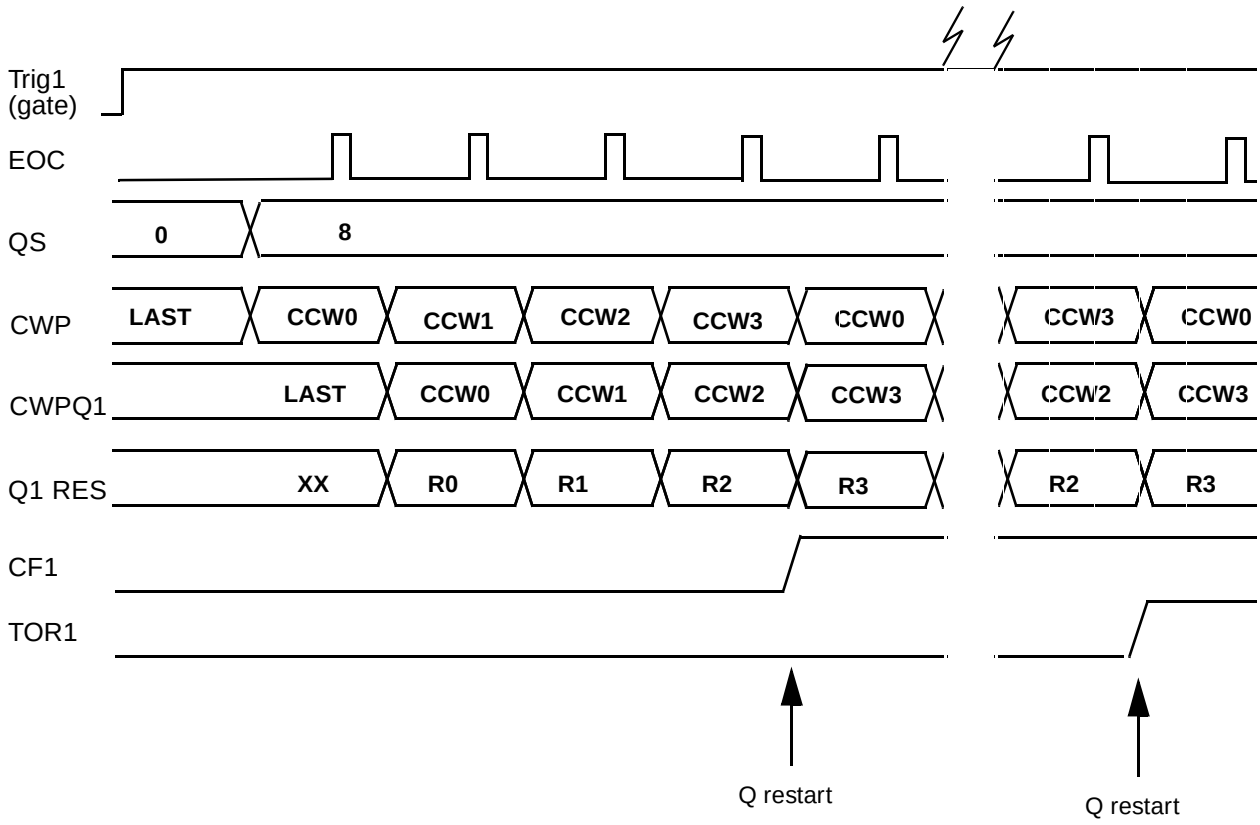


Figure 13-34 Gated Mode, Continuous Scan Timing

13.6 QADC64E Integration Requirements

The QADC64E requires accurate, noise-free input signals for proper operation. This section discusses the design of external circuitry to maximize QADC64E performance.

The QADC64E uses the external pins shown in **Figure 13-1.** There are 32 channel/ port pins and a total of 40 analog input channels. With external multiplexing MPC565 / MPC566 can support 65 analog inputs. 32 of the channel pins can also be used as general-purpose digital port pins. In addition, there are also two analog reference pins, and two analog submodule power shared by both QADC modules.

13.6.1 Port Digital Input/Output Pins

The sixteen port pins can be used as analog inputs, or as a bidirectional 16-bit digital input/output port.

Port A pins are referred to as PQA[7:0] when used as a bidirectional 8-bit digital input/ output port. These eight pins may be used for general-purpose digital input signals or push-pull digital output signals. Port B pins are referred to as PQB[7:0] and operate the same as Port A

Port A and B pins are connected to a digital input synchronizer during reads and may be used as general purpose digital inputs when the applied voltages meet high voltage input (V_{IH}) and low voltage input (V_{IL}) requirements. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information on voltage requirements.

Each port A or B pin is configured as an input or output by programming the port data direction register (DDRQA or DDRQB). The digital input signal states are read by the software in the upper half of the port data register when the port data direction register specifies that the pins are inputs. The digital data in the port data register is driven onto the port A or B pins when the corresponding bit in the port data direction register specifies output. Refer to **APPENDIX A INTERNAL MEMORY MAP** for more information. Since the outputs are configured as push-pull drivers, external pull-up provisions are not necessary when the output is used to drive another integrated circuit.

13.6.2 External Trigger Input Pins

The QADC64E uses two external trigger pins (ETRIG[2:1]). Each of the two input external trigger pins is associated with one of the scan queues, queue 1 or queue 2 The assignment of ETRIG[2:1] to a queue is made in the QACR0 register by the TRG bit. When TRG=0, ETRIG[1] triggers queue 1 and ETRIG[2] triggers queue 2. When TRG=1, ETRIG[1] triggers queue 2 and ETRIG[2] triggers queue 1.

13.6.3 Analog Power Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC64E module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information.

The analog supply pins (V_{DDA} and V_{SSA}) define the limits of the analog reference voltages (V_{RH} and V_{RL}) and of the analog multiplexer inputs. **Figure 13-35** is a diagram of the analog input circuitry.

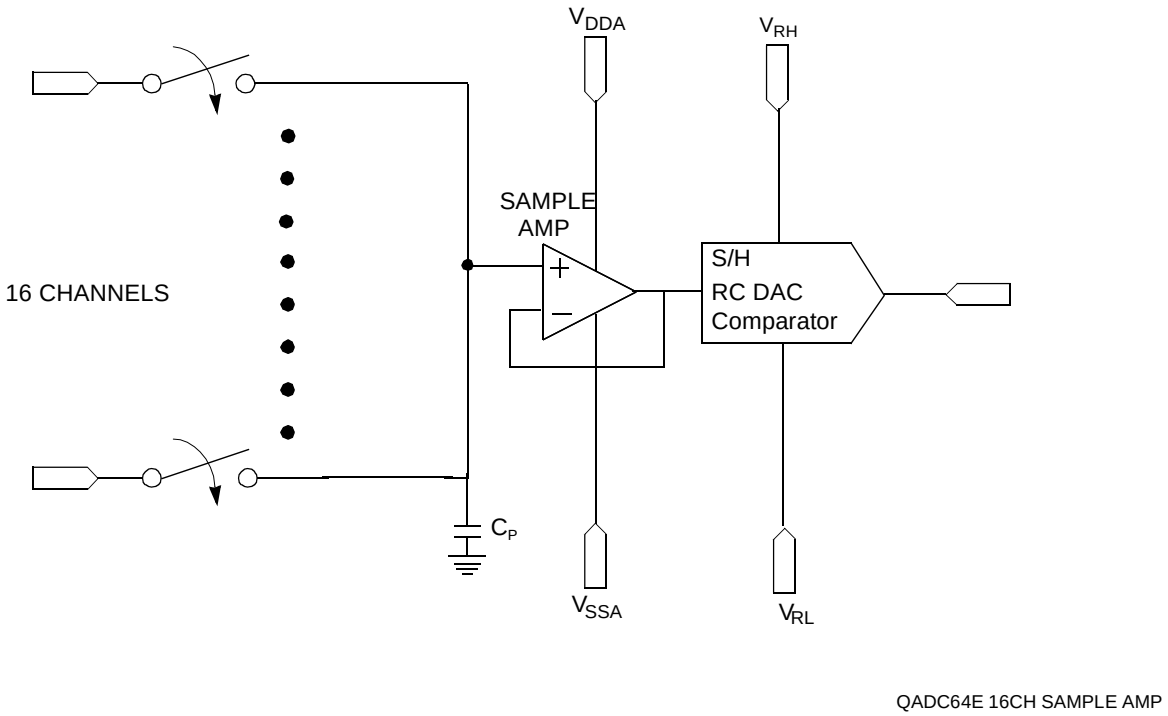


Figure 13-35 Equivalent Analog Input Circuitry

Since the sample amplifier is powered by V_{DDA} and V_{SSA}, it can accurately transfer input signal levels up to but not exceeding V_{DDA} and down to but not below $V_{SSA.}$ If the input signal is outside of this range, the output from the sample amplifier is clipped.

In addition, V_{RH} and V_{RL} must be within the range defined by V_{DDA} and $V_{SSA.}$ As long as V_{RH} is less than or equal to V_{DDA} and V_{RL} is greater than or equal to V_{SSA} and the sample amplifier has accurately transferred the input signal, resolution is ratiometric within the limits defined by V_{RL} and $V_{RH.}$ If V_{RH} is greater than V_{DDA}, the sample amplifier can never transfer a full-scale value. If V_{RL} is less than V_{SSA}, the sample amplifier can never transfer a zero value.

Figure 13-36 shows the results of reference voltages outside the range defined by V_{DDA} and V_{SSA}. At the top of the input signal range, V_{DDA} is 10 mV lower than $V_{RH.}$ This results in a maximum obtainable 10-bit conversion value of 0x3FE. At the bottom of the signal range, V_{SSA} is 15 mV higher than V_{RL}, resulting in a minimum obtainable 10-bit conversion value of three.

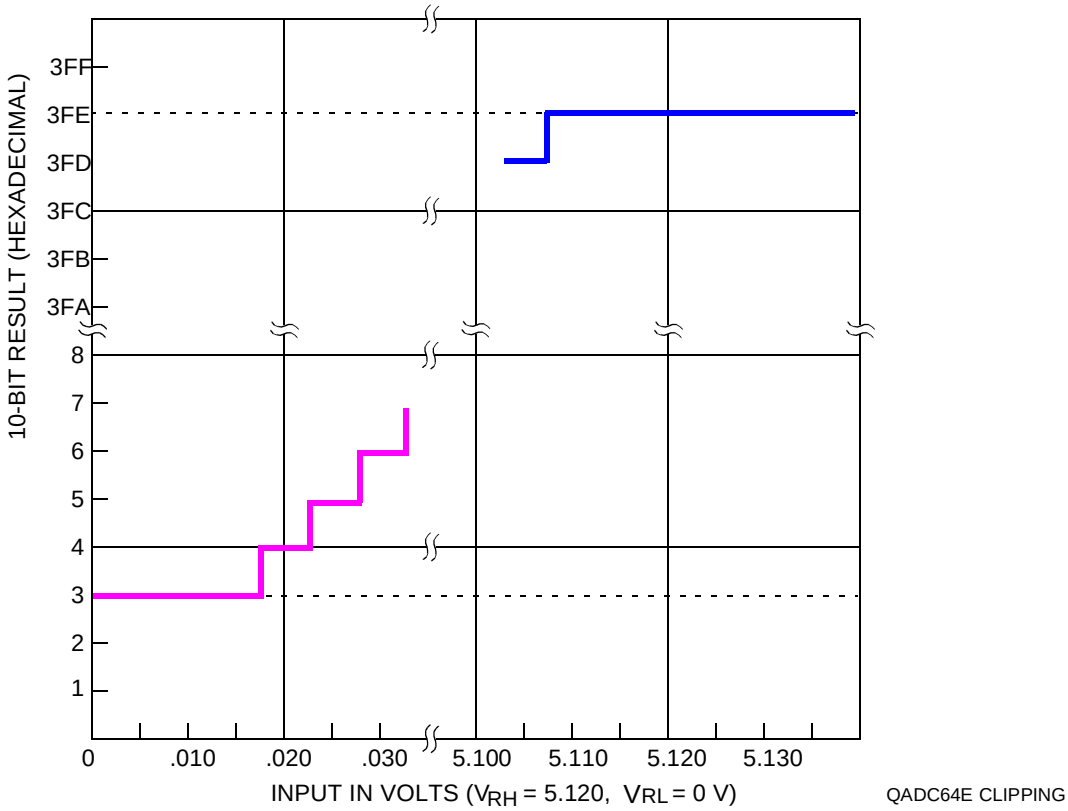


Figure 13-36 Errors Resulting from Clipping

13.6.3.1 Analog Supply Filtering and Grounding

Two important factors influencing performance in analog integrated circuits are supply filtering and grounding. Generally, digital circuits use bypass capacitors on every VDD/VSS pin pair. This applies to analog sub-modules also. The distribution of power and ground is equally important.

Analog supplies should be isolated from digital supplies as much as possible. This necessity stems from the higher performance requirements often associated with analog circuits. Therefore, deriving an analog supply from a local digital supply is not recommended. However, if for economic reasons digital and analog power are derived from a common regulator, filtering of the analog power is recommended in addition to the bypassing of the supplies already mentioned.

EXAMPLE

An RC low pass filter could be used to isolate the digital and analog supplies when generated by a common regulator. If multiple high precision analog circuits are locally employed (i.e., two A/D converters), the analog supplies should be isolated from each other as sharing supplies introduces the potential for interference between analog circuits.

Grounding is the most important factor influencing analog circuit performance in mixed signal systems (or in stand-alone analog systems). Close attention must be paid not to introduce additional sources of noise into the analog circuitry. Common sources of noise include ground loops, inductive coupling, and combining digital and analog grounds together inappropriately.

The problem of how and when to combine digital and analog grounds arises from the large transients which the digital ground must handle. If the digital ground is not able to handle the large transients, the current from the large transients can return to ground through the analog ground. It is the excess current overflowing into the analog ground which causes performance degradation by developing a differential voltage between the true analog ground and the microcontroller's ground pin. The end result is that the ground observed by the analog circuit is no longer true ground and often ends in skewed results.

Two similar approaches designed to improve or eliminate the problems associated with grounding excess transient currents involve star-point ground systems. One approach is to star-point the different grounds at the power supply origin, thus keeping the ground isolated. Refer to **Figure 13-37**.

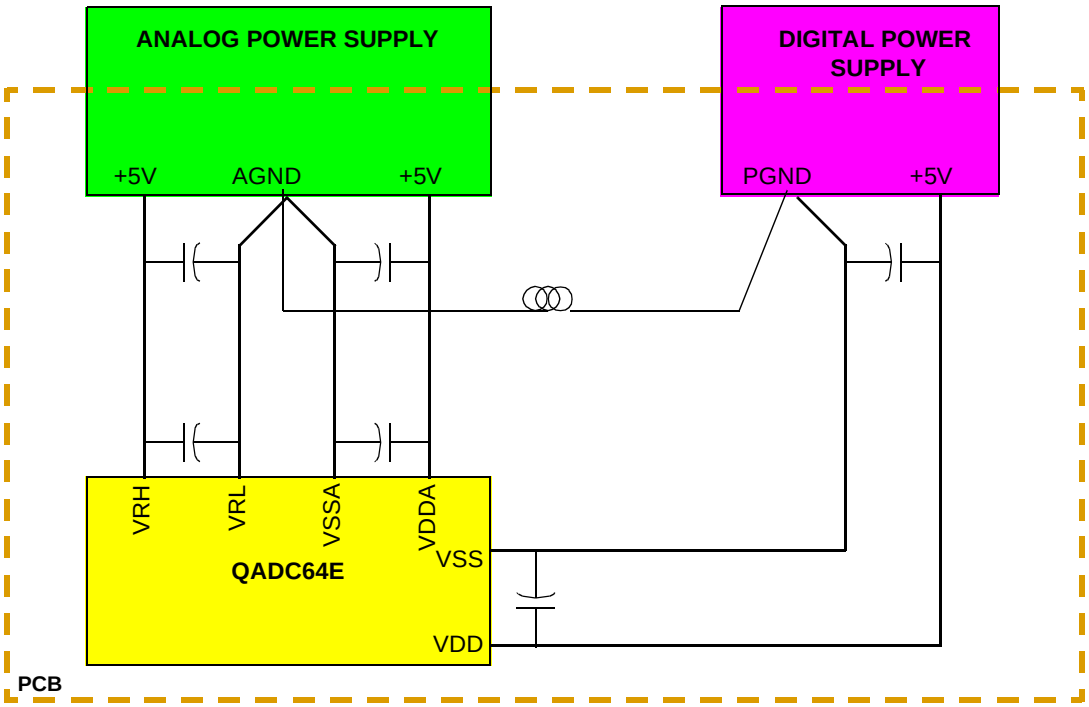


Figure 13-37 Star-Ground at the Point of Power Supply Origin

Another approach is to star-point the different grounds near the analog ground pin on the microcontroller by using small traces for connecting the non-analog grounds to the

analog ground. The small traces are meant only to accommodate DC differences, not AC transients.

NOTE

This star-point scheme still requires adequate grounding for digital and analog subsystems in addition to the star-point ground.

Other suggestions for PCB layout in which the QADC64E is employed include:

- Analog ground must be low impedance to all analog ground points in the circuit.
- Bypass capacitors should be as close to the power pins as possible.
- The analog ground should be isolated from the digital ground. This can be done by cutting a separate ground plane for the analog ground.
- Non-minimum traces should be utilized for connecting bypass capacitors and filters to their corresponding ground/power points.
- Distance for trace runs should be minimized where possible.

13.6.4 Analog Reference Pins

V_{RH} and V_{RL} are the dedicated input pins for the high and low reference voltages.Separating the reference inputs from the power supply pins allows for additional external filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy.

The AltRef pin may be selected through the CCW as the high reference for a conversion. This allows for the ability to “zoom” in on a portion of the convertible range with the full 10 bits. Refer to **Table 13-18**.

No A/D converter can be more accurate than its analog reference. Any noise in the reference can result in at least that much error in a conversion. The reference for the QADC64E, supplied by pins V_{RH}, AltRef, and V_{RL}, should be low-pass filtered from its source to obtain a noise-free, clean signal. In many cases, simple capacitive bypassing may sufficed. In extreme cases, inductors or ferrite beads may be necessary if noise or RF energy is present. Series resistance is not advisable since there is an effective DC current requirement from the reference voltage by the internal resistor string in the RC DAC array. External resistance may introduce error in this architecture under certain conditions. Any series devices in the filter network should contain a minimum amount of DC resistance.

13.6.5 Analog Input Pins

Analog inputs should have low AC impedance at the pins. Low AC impedance can be realized by placing a capacitor with good high frequency characteristics at the input pin of the part. Ideally, that capacitor should be as large as possible (within the practical range of capacitors that still have good high frequency characteristics). This capacitor has two effects:

- It helps attenuate any noise that may exist on the input.

- It sources charge during the sample period when the analog signal source is a high-impedance source.

Series resistance can be used with the capacitor on an input pin to implement a simple RC filter. The maximum level of filtering at the input pins is application dependent and is based on the bandpass characteristics required to accurately track the dynamic characteristics of an input. Simple RC filtering at the pin may be limited by the source impedance of the transducer or circuit supplying the analog signal to be measured. Refer to **13.6.5.3 Error Resulting from Leakage** for more information. In some cases, the size of the capacitor at the pin may be very small.

Figure 13-38 is a simplified model of an input channel. Refer to this model in the following discussion of the interaction between the external circuitry and the circuitry inside the QADC64E.

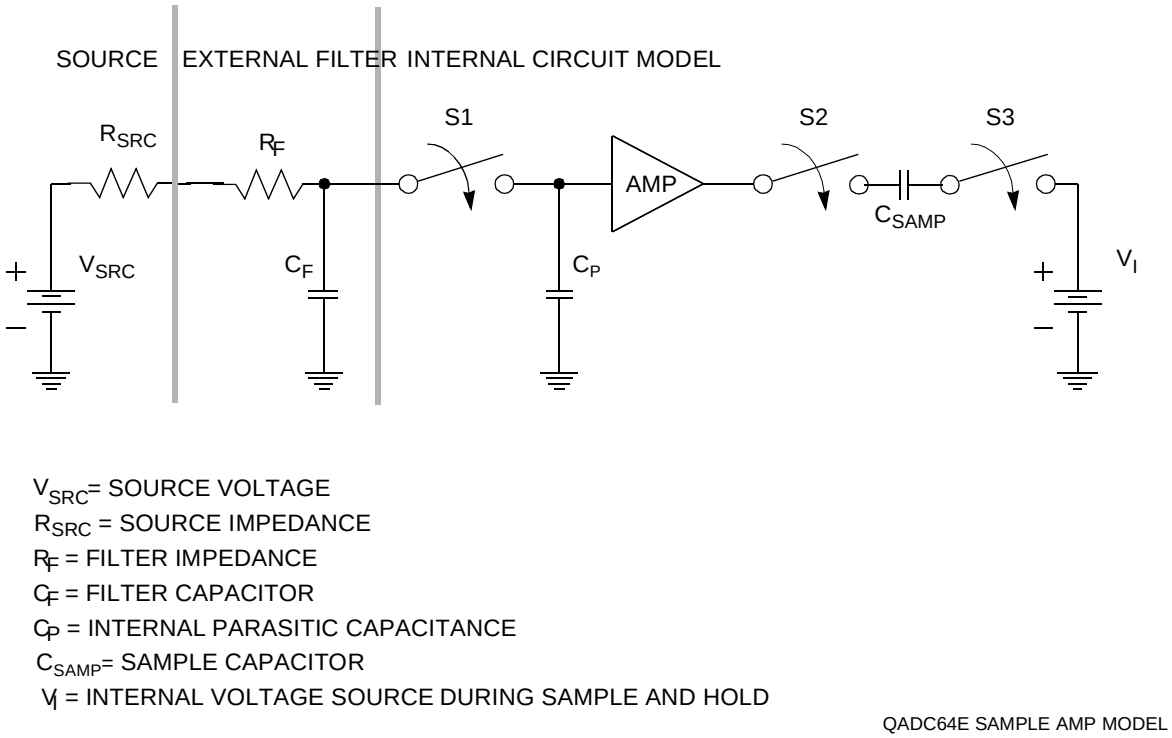


Figure 13-38 Electrical Model of an A/D Input Pin

In **Figure 13-38**, R_F, R_{SRC} and C_F comprise the external filter circuit. C_P is the internal parasitic capacitor. C_{SAMP} is the capacitor array used to sample and hold the input voltage. V_I is an internal voltage source used to provide charge to C_{SAMP} during sample phase.

The following paragraphs provide a simplified description of the interaction between the QADC64E and the external circuitry. This circuitry is assumed to be a simple RC low-pass filter passing a signal from a source to the QADC64E input pin. The following simplifying assumptions are made:

- The external capacitor is perfect (no leakage, no significant dielectric absorption characteristics, etc.)
- All parasitic capacitance associated with the input pin is included in the value of the external capacitor
- Inductance is ignored
- The “on” resistance of the internal switches is 0 Ω and the “off” resistance is infinite

13.6.5.1 Analog Input Considerations

The source impedance of the analog signal to be measured and any intermediate filtering should be considered whether external multiplexing is used or not. **Figure 13-39** shows the connection of eight typical analog signal sources to one QADC64E analog input pin through a separate multiplexer chip. Also, an example of an analog signal source connected directly to a QADC64E analog input channel is displayed.

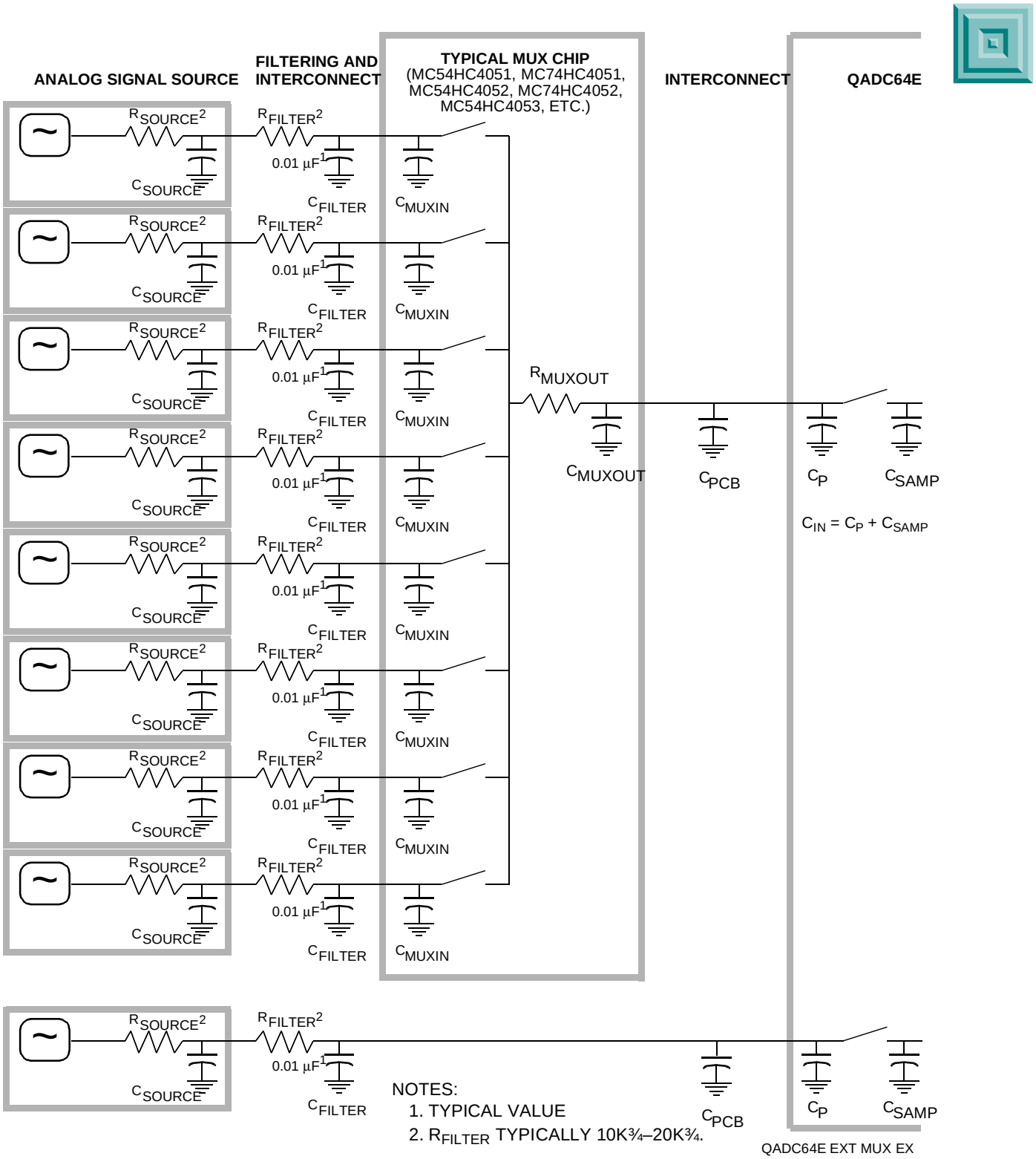


Figure 13-39 External Multiplexing of Analog Signal Sources

13.6.5.2 Settling Time for the External Circuit

The values for R_{SRC}, R_F and C_F in the external circuitry determine the length of time required to charge C_F to the source voltage level (V_{SRC}). At time t = 0, V_{SRC} changes in **Figure 13-38** while S1 is open, disconnecting the internal circuitry from the external circuitry. Assume that the initial voltage across C_F is zero. As C_F charges, the voltage across it is determined by the following equation, where t is the total charge time:

$$V_{CF} = V_{SRC}(1 - e^{-t/(R_F + R_{SRC})C_F})$$

As t approaches infinity, V_{CF} will equal V_{SRC}. (This assumes no internal leakage.) With 10-bit resolution, 1/2 of a count is equal to 1/2048 full-scale value. Assuming worst case (V_{SRC} = full scale), **Table 13-24** shows the required time for C_F to charge to within 1/2 of a count of the actual source voltage during 10-bit conversions. **Table 13-24** is based on the RC network in **Figure 13-38**.

NOTE

The following times are completely independent of the A/D converter architecture (assuming the QADC64E is not affecting the charging).

Table 13-24 External Circuit Settling Time to 1/2 LSB (10-Bit Conversions)

Filter Capacitor (CF)	Source Resistance (R_F + R_{SRC})			
	100 Ω	1 kΩ	10 kΩ	100 kΩ
1 µF	760 µs	7.6 ms	76 ms	760 ms
.1 µF	76 µs	760 µs	7.6 ms	76 ms
.01 µF	7.6 µs	76 µs	760 µs	7.6 ms
.001 µF	760 ns	7.6 µs	76 µs	760 µs
100 pF	76 ns	760 ns	7.6 µs	76 µs

The external circuit described in **Table 13-24** is a low-pass filter. A user interested in measuring an AC component of the external signal must take the characteristics of this filter into account.

13.6.5.3 Error Resulting from Leakage

A series resistor limits the current to a pin, therefore input leakage acting through a large source impedance can degrade A/D accuracy. The maximum input leakage current is specified in **APPENDIX E ELECTRICAL CHARACTERISTICS**. Input leakage is greater at higher operating temperatures. In the temperature range from 125° C to 50° C, the leakage current is halved for every 8 – 12° C reduction in temperature.

Assuming $V_{RH} - V_{RL}$ = 5.12 V, one count (assuming 10-bit resolution) corresponds to 5 mV of input voltage. A typical input leakage of 200 nA acting through 10 kΩ of external series resistance results in an error of 0.4 count (2.0 mV). If the source impedance is 100 kΩ and a typical leakage of 100 nA is present, an error of two counts (10 mV) is introduced.

In addition to internal junction leakage, external leakage (e.g., if external clamping diodes are used) and charge sharing effects with internal capacitors also contribute to the total leakage current. **Table 13-25** illustrates the effect of different levels of total leakage on accuracy for different values of source impedance. The error is listed in terms of 10-bit counts.

CAUTION

Leakage from the part below 200 nA is obtainable only within a limited temperature range.

Table 13-25 Error Resulting From Input Leakage (IOFF)

Source Impedance	Leakage Value (10-bit Conversions)			
	100 nA	200 nA	500 nA	1000 nA
1 kΩ	—	—	0.1 counts	0.2 counts
10 kΩ	0.2 counts	0.4 counts	1 counts	2 counts
100 kΩ	2 counts	4 count	10 counts	20 counts

13.6.5.4 Accommodating Positive/Negative Stress Conditions

Positive or negative stress refers to conditions which exceed nominally defined operating limits. Examples include applying a voltage exceeding the normal limit on an input (for example, voltages outside of the suggested supply/reference ranges) or causing currents into or out of the pin which exceed normal limits. QADC64E specific considerations are voltages greater than V_{DDA}, V_{RH} or less than V_{SSA} applied to an analog input which cause excessive currents into or out of the input. Refer **APPENDIX E ELECTRICAL CHARACTERISTICS** to for more information on exact magnitudes.

Either stress condition can potentially disrupt conversion results on neighboring inputs. Parasitic devices, associated with CMOS processes, can cause an immediate disruptive influence on neighboring pins. Common examples of parasitic devices are diodes to substrate and bipolar devices with the base terminal tied to substrate (V_{SSI}/V_{SSA} ground). Under stress conditions, current injected on an adjacent pin can cause errors on the selected channel by developing a voltage drop across the selected channel's impedances.

Figure 13-40 shows an active parasitic bipolar NPN transistor when an input pin is subjected to negative stress conditions. **Figure 13-41** shows positive stress conditions can activate a similar PNP transistor.

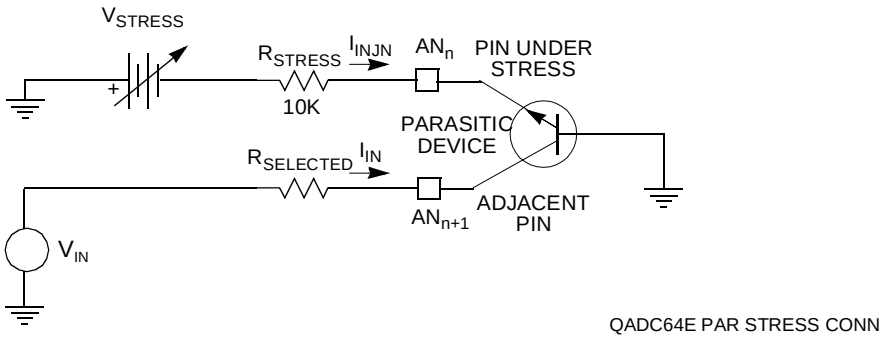


Figure 13-40 Input PIn Subjected to Negative Stress

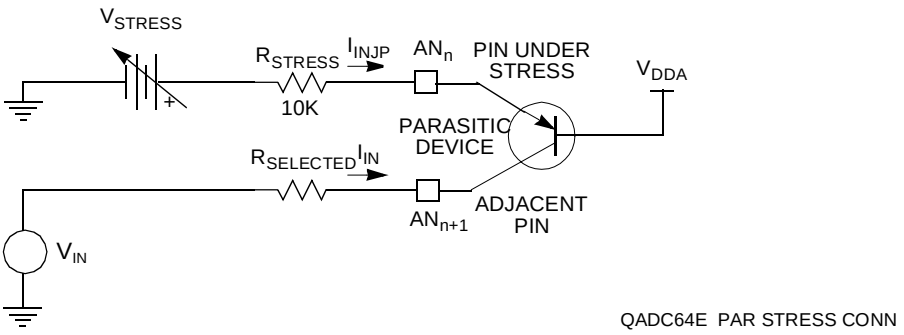


Figure 13-41 Input Pin Subjected to Positive Stress

The current into the pin (I_{INJN} or I_{INJP}) under negative or positive stress is determined by the following equations:

$$I_{INJN} = \frac{-(V_{STRESS} - V_{BE})}{R_{STRESS}}$$

$$I_{INJP} = \frac{V_{STRESS} - V_{EB} - V_{DDA}}{R_{STRESS}}$$

where:

V_{STRESS} = Adjustable voltage source

V_{EB} = Parasitic PNP emitter/base voltage (refer to $V_{NEGCLAMP}$ in **APPENDIX E ELECTRICAL CHARACTERISTICS**)

V_{BE} = Parasitic NPN base/emitter voltage
(refer to $V_{NEGCLAMP}$ in
APPENDIX E ELECTRICAL CHARACTERISTICS)

R_{STRESS} = Source impedance
(10-kΩ resistor in **Figure 13-40** and **Figure 13-41** on stressed channel)

$R_{SELECTED}$ = Source impedance on channel selected for conversion

The current into (I_{IN}) the neighboring pin is determined by the K_N (current coupling ratio) of the parasitic bipolar transistor ($K_N << 1$). The I_{IN} can be expressed by the following equation:

$$I_{IN} = - K_N * I_{INJ}$$

where I_{INJ} is either I_{INJN} or I_{INJP}.

A method for minimizing the impact of stress conditions on the QADC64E is to strategically allocate QADC64E inputs so that the lower accuracy inputs are adjacent to the inputs most likely to see stress conditions.

Also, suitable source impedances should be selected to meet design goals and minimize the effect of stress conditions.

SECTION 14
QUEUED SERIAL MULTI-CHANNEL MODULE

14.1 Overview

The MPC565 / MPC566 contains two queued serial multi-channel modules (QSMCM_A and QSMCM_B). The QSMCM provides three serial communication interfaces: the queued serial peripheral interface (QSPI) and two serial communications interfaces (SCI1 and SCI2). These submodules communicate with the CPU via a common slave bus interface unit (SBIU).

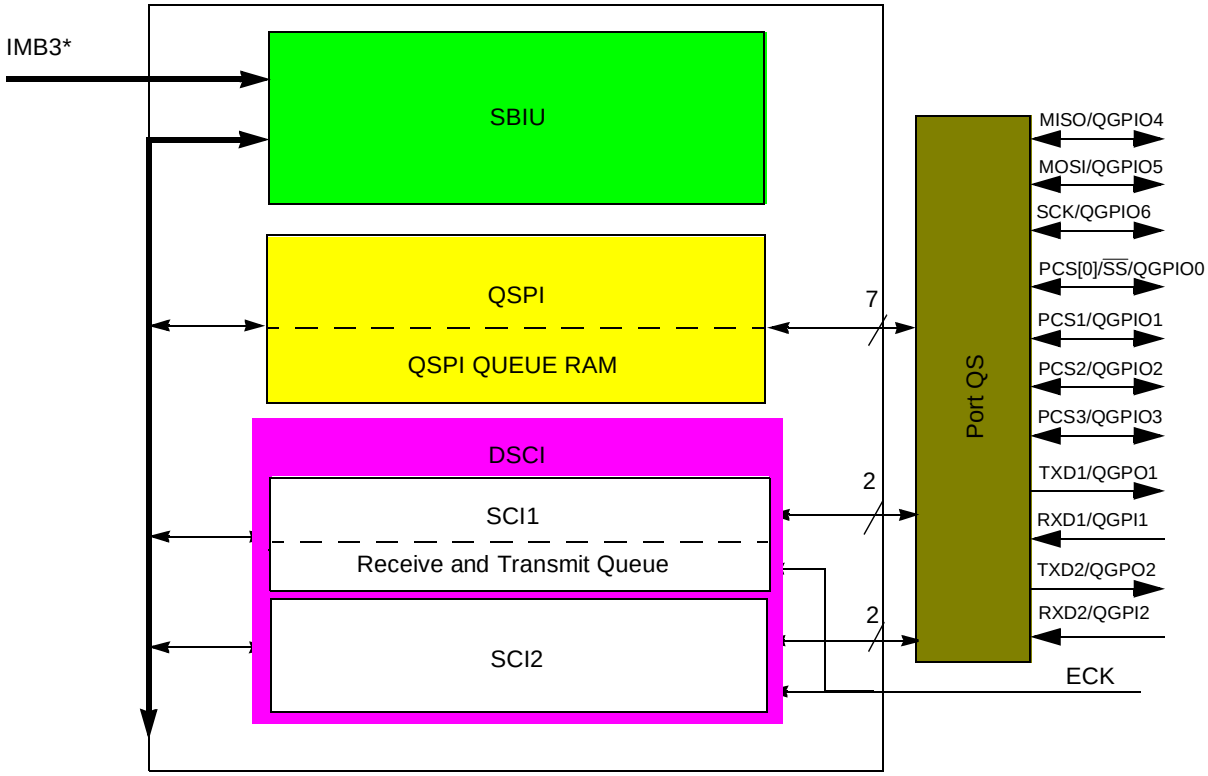
The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced from the original SPI in the QSMCM (queued serial module) to include a total of 160 bytes of queue RAM to accommodate more receive, transmit, and control information.

The duplicate, independent SCIs are full-duplex universal asynchronous receiver transmitter (UART) serial interface. The original QSM SCI is enhanced by the addition of an SCI, a common external baud clock source, receive and transmit buffers on one SCI.

The SBIU provides an interface between the QSMCM module and the intermodule bus (IMB3).

14.2 Block Diagram

Figure 14-1 depicts the major components of the QSMCM.



*Note: SBIU bus and interface to IMB3 are each 16 bits wide.

Figure 14-1 QSMCM Block Diagram

14.2.1 MPC565 / MPC566 QSMCM Details

The MPC565 / MPC566 has two QSMCM modules, QSMCM_A and QSMCM_B. QSMCM_A has identical function to the MPC555/MPC556. QSMCM_B has its RXD2and PCS[3] pins muxed with the DLCMD2 (J1850) module. The MUXing of the pins is controlled by the QPAPCS3 bit in QSMCM_B pin assignment register (PQSPAR_B), according to **Table 14-1** The MUXed pins default to DLCMD2 function at reset. Since the normal function of the PCS pins within the QSMCM require that this bit be written before the PCS pin is used, the MUXing appears transparent to both the QSMCM and the DLCMD2. However, only one of the modules, DLCMD2 or QSMCM can use the pins in a system.

Table 14-1 DLCMD2 / QSMCM_B SCI2 Pin MUX Control

QPAPCS3 Bit Value	QSMCM_B / DLCMD2 Pin Function
0	PCS[3]_J1850_TX_B pin assigned to J1850_TX. RXD2_J1850_RX_B pin assigned to J1850_RX. Pins are assigned to DLCMD2 (J1850_TX & J1850_RX)
1	PCS[3]_J1850_TX_B pin assigned to PCS[3]. RXD2_J1850_RX_B pin assigned to B_RXD2. Pins are assigned to QSMCM_B SCI2 (PCS[3]_B, RXD2_B)

Because of this MUXed function on QSMCM_B, the general purpose I/O functions are not available on the B_PCS[3] and B_RXD2 pins. **Table 14-2** shows the pin names for the modified pins.

Table 14-2 QSMCM Pins on QSMCM_A and QSMCM_B

QSMCM_A pins	QSMCM_B Pins	Different
PCS[0]/$\overline{SS}$/QGPIO[0]_A	PCS[0]/$\overline{SS}$/QGPIO[0]_B	
PCS[1]/QGPIO[1]_A	PCS[1]/QGPIO[1]_B	
PCS[2]/QGPIO[2]_A	PCS[2]/QGPIO[2]_B	
PCS[3]/QGPIO[3]_A	PCS[3]/J1850/TX_B	*
MISO/QGPIO[4]_A	MISO/QGPIO[4]_B	
MOSI/QGPIO[5]_A	MOSI/QGPIO[5]_B	
SCK/QGPIO[6]_A	SCK/QGPIO[6]_B	
—	ECK_B	
TXD1/QGPO[1]_A	TXD1/QGPO[1]_B	
TXD2/QGPO[2]_A	TXD2/QGPO[2]_B	
RXD1/QGPI[1]_A	RXD1/QGPI[1]_B	
RXD2/QGPI[2]_A	RXD2/J1850_RX_B	*

14.3 Signal Descriptions

The QSMCM has 12 external pins, as shown in **Figure 14-1**. Seven of the pins, if not in use for their submodule function, can be used as general-purpose I/O port pins. The RXDx and TXDx pins can alternately serve as general-purpose input-only and output-only signals, respectively. ECK is a dedicated clock pin.

For detailed descriptions of QSMCM signals, refer to **14.6 QSMCM Pin Control Registers**, **14.7.3 QSPI Pins**, and **14.8.6 SCI Pins**.

14.4 Memory Maps

The QSMCM memory maps, shown in **Table 14-3** and **Table 14-4**, includes the global registers, the QSPI and dual SCI control and status registers, and the QSPI RAM. The QSMCM memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **Figure 1-2** for a diagram of the MPC565 / MPC566 internal memory map.

Table 14-3 QSMCM_A and QSMCM_B Register Map

Access[1]	Address	MSB[2] 0	LSB 15
S	0x30 5000(A) 0x30 5400(B)	QSMCM Module Configuration Register (QSMCMMCR) See **Table 14-7** for bit descriptions.	
T	0x30 5002(A) 0x30 5402(B)	QSMCM Test Register (QTEST)	
S	0x30 5004(A) 0x30 5404(B)	Dual SCI Interrupt Level (QDSCI_IL) See **Table 14-8** for bit descriptions.	Reserved
S	0x30 5006(A) 0x30 5406(B)	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 14-9** for bit descriptions.
S/U	0x30 5008(A) 0x30 5408(B)	SCI1Control Register 0 (SCC1R0) See **Table 14-27** for bit descriptions.	
S/U	0x30 500A(A) 0x30 540A(B)	SCI1Control Register 1 (SCC1R1) See **Table 14-28** for bit descriptions.	
S/U	0x30 500C(A) 0x30 540C(B)	SCI1 Status Register (SC1SR) See **Table 14-29** for bit descriptions.	
S/U	0x30 500E(A) 0x30 540E(B)	SCI1 Data Register (SC1DR) See **Table 14-30** for bit descriptions.	
S/U	0x30 5010(A) 0x30 5410(B)	Reserved	
S/U	0x30 5012(A) 0x30 5412(B)	Reserved	
S/U	0x30 5014(A) 0x30 5414(B)	Reserved	QSMCM Port Q Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
S/U	0x30 5016(A) 0x30 5416(B)	QSMCM Pin Assignment Register (PQS-PAR) See **Table 14-14** for bit descriptions.	QSMCM Data Direction Register (DDRQS) See **Table 14-15** for bit descriptions.
S/U	0x30 5018(A) 0x30 5418(B)	QSPI Control Register 0 (SPCR0) See **Table 14-17** for bit descriptions.	
S/U	0x30 501A(A) 0x30 541A(B)	QSPI Control Register 1 (SPCR1) See **Table 14-19** for bit descriptions.	
S/U	0x30 501C(A) 0x30 541C(B)	QSPI Control Register 2 (SPCR2) See **Table 14-20** for bit descriptions.	
S/U	0x30 501E(A) 0x30 541E(B)	QSPI Control Register 3 (SPCR3) See **Table 14-21** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-22** for bit descriptions.
S/U	0x30 5020(A) 0x30 5420(B)	SCI2 Control Register 0 (SCC2R0)	
S/U	0x30 5022(A) 0x30 5422(B)	SCI2 Control Register 1 (SCC2R1)	
S/U	0x30 5024(A) 0x30 5424(B)	SCI2 Status Register (SC2SR)	
S/U	0x30 5026(A) 0x30 5426(B)	SCI2 Data Register (SC2DR)	
S/U	0x30 5028(A) 0x30 5428(B)	QSCI1 Control Register (QSCI1CR) See **Table 14-35** for bit descriptions.	
S/U	0x30 502A(A) 0x30 542A(B)	QSCI1 Status Register (QSCI1SR) See **Table 14-36** for bit descriptions.	

Table 14-3 QSMCM_A and QSMCM_B Register Map (Continued)

Access[1]	Address	MSB[2] 0	LSB 15
S/U	0x30 502C – 0x30 504A(A) 0x30 542C – 0x30 544A(B)	Transmit Queue Locations (SCTQ)	
S/U	0x30 504C – 0x30 506A(A) 0x30 544C – 0x30 546A(B)	Receive Queue Locations (SCRQ)	
S/U	0x30 506C – 0x30 513F(A)[3] 0x30 546C – 0x30 553F(B)	Reserved	
S/U	0x30 5140 – 0x30 517F(A) 0x30 5540 – 0x30 557F(B)	Receive Data RAM (REC.RAM)	
S/U	0x30 5180 – 0x30 51BF(A) 0x30 5580 – 0x30 55BF(B)	Transmit Data RAM (TRAN.RAM)	
S/U	0x30 51C0 – 0x30 51DF(A) 0x30 55C0 – 0x30 55DF(B)	Command RAM (COMD.RAM)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.
3. Note that QRAM offsets have been changed from the original (modular family) QSMCM.

The supervisor-only data space segment contains the QSMCM global registers. These registers define parameters needed by the QSMCM to integrate with the MCU. Access to these registers is permitted only when the CPU is operating in supervisor mode.

Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user accesses. The supervisor (SUPV) bit in the QSMCM module configuration register (QSMCMMCR) designates the assignable data space as either supervisor or unrestricted. If SUPV is set, then the space is designated as supervisor-only space. Access is then permitted only when the CPU is operating in supervisor mode. If SUPV is clear, both user and supervisor accesses are permitted. To clear SUPV, the CPU must be in supervisor mode.

The QSMCM assignable data space segment contains the control and status registers for the QSPI and SCI submodules, as well as the QSPI RAM. All registers and RAM can be accessed on byte (8-bit), half-word (16-bit), and word (32-bit) boundaries. Word accesses require two consecutive IMB3 bus cycles.

14.5 QSMCM Global Registers

The QSMCM global registers contain system parameters used by the QSPI and SCI submodules for interfacing to the CPU and the intermodule bus. The global registers are listed in **Table 14-4** and **Table 14-5**.

Table 14-4 QSMCM_A Global Registers

Access[1]	Address	MSB[2]	LSB
S	0x30 5000	QSMCM Module Configuration Register (QSMCMMCR_A) See **Table 14-7** for bit descriptions.	
T	0x30 5002	QSMCM Test Register (QTEST_A)	
S	0x30 5004	Dual SCI Interrupt Level (QDSCI_IL_A) See **Table 14-8** for bit descriptions.	Reserved
S	0x30 5006	Reserved	Queued SPI Interrupt Level (QSPI_IL_A) See **Table 14-9** for bit descriptions.

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers reside on 8-bit boundaries. 16-bit registers reside on 16-bit boundaries.

Table 14-5 QSMCM_B Global Registers

Access[1]	Address	MSB[2]	LSB
S	0x30 5400	QSMCM Module Configuration Register (QSMCMMCR_B) See **Table 14-7** for bit descriptions.	
T	0x30 5402	QSMCM Test Register (QTEST_B)	
S	0x30 5404	Dual SCI Interrupt Level (QDSCI_IL_B) See **Table 14-8** for bit descriptions.	Reserved
S	0x30 5406	Reserved	Queued SPI Interrupt Level (QSPI_IL_B) See **Table 14-9** for bit descriptions.

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers reside on 8-bit boundaries. 16-bit registers reside on 16-bit boundaries.

14.5.1 Low-Power Stop Operation

When the STOP bit in QSMCMMCR is set, the IMB3 clock input to the QSMCM is disabled and the module enters a low-power operating state. QSMCMMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable in low-power stop mode. However, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be written by the CPU and is cleared by reset.

System software must bring each submodule to an orderly stop before setting STOP to avoid data corruption. The SCI receiver and transmitter should be disabled after transfers in progress are complete. The QSPI can be halted by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set.

14.5.2 Freeze Operation

The FRZ1 bit in QSMCMMCR determines how the QSMCM responds when the IMB3 FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debug mode. Setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. FREEZE causes the SCI1 transmit queue to halt on the first transfer boundary following FREEZE assertion.

14.5.3 Access Protection

The SUPV bit in the QMCR defines the assignable QSMCM registers as either supervisor-only data space or unrestricted data space.

When the SUPV bit is set, all registers in the QSMCM are placed in supervisor-only space. For any access from within user mode, the IMB3 address acknowledge ($\overline{AACK}$) signal is asserted and a bus error is generated.

Because the QSMCM contains a mix of supervisor and user registers, $\overline{AACK}$ is asserted for either supervisor or user mode accesses, and the bus cycle remains internal. If a supervisor-only register is accessed in user mode, the module responds as if an access had been made to an unauthorized register location, and a bus error is generated.

14.5.4 QSMCM Interrupts

The interrupt structure of the IMB3 supports a total of 32 interrupt levels that are time multiplexed on the $\overline{IRQB}$[0:7] lines as seen in **Figure 14-2**.

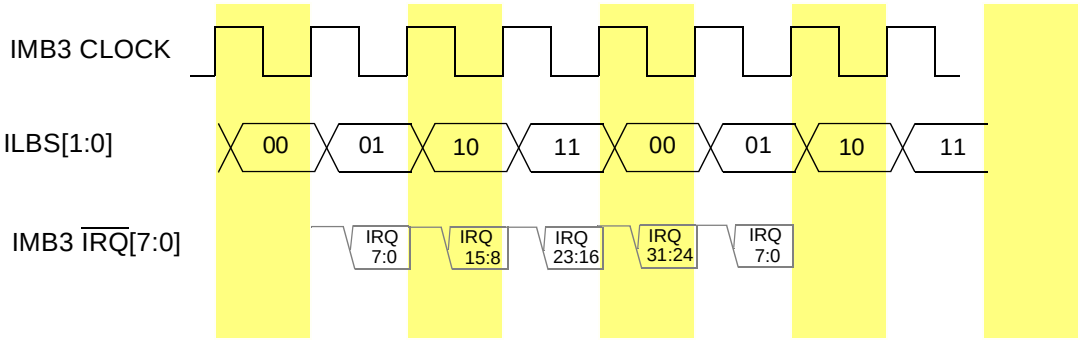


Figure 14-2 QSMCM Interrupt Levels

In this structure, all interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 14-6 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The QSMCM module is capable of generating one of the 32 possible interrupt levels on the IMB3. The levels that the interrupt will drive can be programmed into the interrupt request level (ILDSCI and ILQSPI) bits located in the interrupt configuration register (QDSCI_IL and QSPI_IL). This value determines which interrupt signal ($\overline{\text{IRQ}}$B[0:7]) is driven onto the bus during the programmed time slot. **Figure 14-3** shows a block diagram of the interrupt hardware.

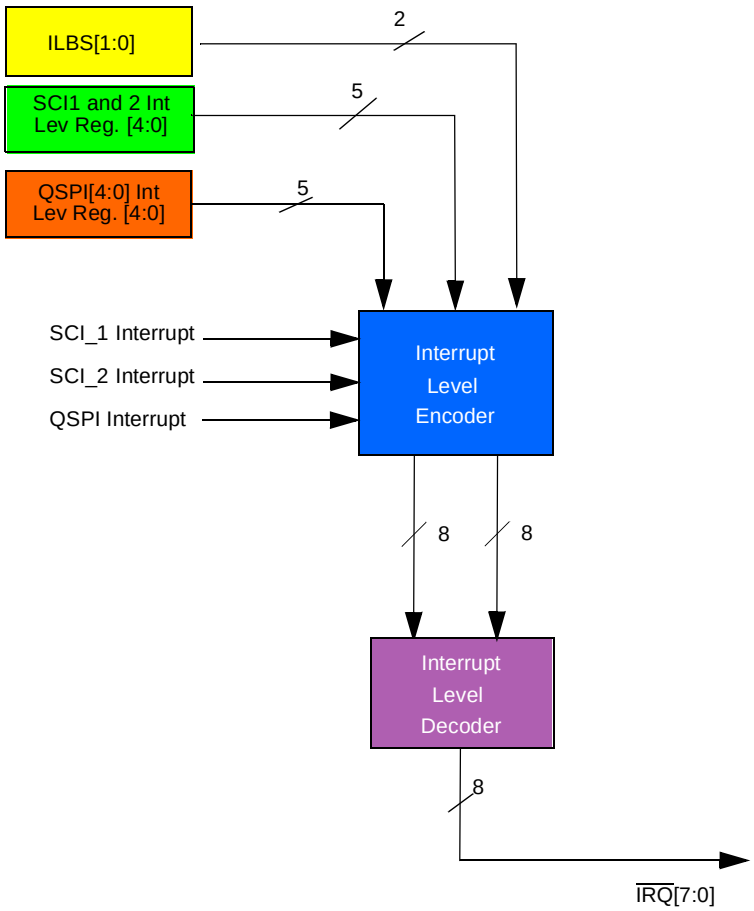


Figure 14-3 QSPI Interrupt Generation

14.5.5 QSMCM Configuration Register (QSMCMMCR)

The QSMCMMCR contains parameters for interfacing to the CPU and the intermodule bus. This register can be modified only when the CPU is in supervisor mode.

QSMCMMCR — QSMCM Configuration Register **0x30 5000**
0x30 5400

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ1	RESERVED						SUPV	RESERVED						
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 14-7 QSMCMMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Stop enable. Refer to **14.5.1 Low-Power Stop Operation**. 0 = Normal clock operation 1 = Internal clocks stopped
1	FRZ1	Freeze1 bit. Refer to **14.5.2 Freeze Operation**. 0 = Ignore the FREEZE signal 1 = Halt the QSMCM (on transfer boundary)
2:7	—	Reserved
8	SUPV	Supervisor / Unrestricted. Refer to **14.5.3 Access Protection**. 0 = Assigned registers are unrestricted (user access allowed) 1 = Assigned registers are restricted (only supervisor access allowed)
9:11	—	Reserved
12:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in QSM implementations that use hardware interrupt arbitration.

14.5.6 QSMCM Test Register (QTEST)

The QTEST register is used for factory testing of the MCU.

14.5.7 QSMCM Interrupt Level Registers (QDSCI_IL, QSPI_IL)

The QDSCI_ILI and QSPI_IL registers determine the interrupt level requested by the QSMCM. The two SCI submodules (DSCI) share a 5-bit interrupt level field, ILDSCI. The QSPI uses a separate field, ILQSPI. The level value is used to determine which interrupt is serviced first when two or more modules or external peripherals simultaneously request an interrupt. The user can select among 32 levels. This register can be accessed only when the CPU is in supervisor mode.

QDSCI_IL — QSM2 Dual SCI Interrupt Level Register **0x30 5004**
0x30 5404

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED			ILDSCI					RESERVED							
RESET:															
0	0	0	0	0	0	0	0								

Table 14-8 QDSCI_IL Bit Descriptions

Bit(s)	Name	Description
0:2	—	Reserved
3:7	ILDSCI	Interrupt level of SCIs 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)
8:15	—	Reserved

QSPI_IL — QSPI Interrupt Level Register **0x30 5006**
0x30 5406

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED											ILQSPI				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-9 QSPI_IL Bit Descriptions

Bit(s)	Name	Description
0:10	—	Reserved
11:15	ILQSPI	Interrupt level of SPI 00000 = lowest interrupt level request (level 0) 11111 = highest interrupt level request (level 31)

14.6 QSMCM Pin Control Registers

Table 14-10 lists the three QSMCM pin control registers.

Table 14-10 QSMCM Pin Control Registers

Address	Register
0x30 5014 0x30 5414	QSMCM Port Data Register (PORTQS) See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
0x30 5016 0x30 5416	PORTQS Pin Assignment Register (PQSPAR) See **Table 14-14** for bit descriptions.
0x30 5017 0x30 5417	PORTQS Data Direction Register (DDRQS) See **Table 14-14** for bit descriptions.

Each QSMCM uses 12 pins. Eleven of the pins (nine on QSMCM_B), when not being used by the serial sub-systems, form a parallel port on the MCU. (The ECK pin is a dedicated external clock source.)

The port QS pin assignment register (PQSPAR) governs the usage of QSPI pins. Clearing a bit assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. QPAPCS[3] (bit 1) of PQSPAR_B selects between the J1850_TX/J1850_RX and B_PCS[3]/B_RXD2.

PQSPAR does not affect operation of the SCI. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. When the SCIx transmitter or receiver is enabled, the associated TXDx or RXDx pin is assigned its SCI function.

The port QS data direction register (DDRQS) determines whether QSPI pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table 14-14** summarizes the effect of DDRQS bits on QSPI pin function.

DDRQS does not affect SCI pin function. TXDx pins are always outputs, and RXDx pins are always inputs, regardless of whether they are functioning as SCI pins or as PORTQS pins.

The port QS data register (PORTQS) latches I/O data. PORTQS writes drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, write the first data to PORTQS before configuring DDRQS.

Table 14-11 Effect of DDRQS on QSPI Pin Function

QSMCM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQS[0]	0	Serial data input to QSPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from QSPI
MOSI	Master	DDQS[1]	0	Disables data output
			1	Serial data output from QSPI
	Slave		0	Serial data input to QSPI
			1	Disables data input
SCK[1]	Master	DDQS[2]	—	Clock output from QSPI
	Slave		—	Clock input to QSPI
PCS[0]/$\overline{SS}$	Master	DDQS[3]	0	Assertion causes mode fault
			1	Chip-select output
	Slave		0	QSPI slave select input
			1	Disables slave select input
PCS[1:3]	Master	DDQS[4:6]	0	Disables chip-select output
			1	Chip-select output
	Slave		0	Inactive
			1	Inactive

NOTES:
1. SCK/QGPIO6 is a digital I/O pin unless the SPI is enabled (SPE set in SPCR1), in which case it becomes the QSPI serial clock SCK.

14.6.1 Port QS Data Register (PORTQS)

PORTQS determines the actual input or output value of a QSMCM port pin if the pin is defined as general-purpose input or output. All QSMCM pins except the ECK pin can be used as general-purpose input and/or output. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. Writes to this register affect the pins defined as outputs; reads of this register return the actual value of the pins.

PORTQS — Port QS Data Register **0x30 5014**
0x30 5414

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED				QDRXD 2	QDTXD 2	QDRXD 1	QDTXD 1	0	QDPCS 3	QDPCS 2	QDPCS 1	QDPCS 0	QDSCK	QDMO-SI	QDMI-SO
RESET:															
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

14.6.2 PORTQS Pin Assignment Register (PQSPAR)

PQSPAR determines which of the QSPI pins, with the exception of the SCK pin, are used by the QSPI submodule, and which pins are available for general-purpose I/O. Pins may be assigned on a pin-by-pin basis. If the QSPI is disabled, the SCK pin is automatically assigned its general-purpose I/O function (QGPIO6).

QSPI pins designated by PQSPAR as general-purpose I/O pins are controlled only by PQSDDR and PQSPDR; the QSPI has no effect on these pins. PQSPAR does not affect the operation of the SCI submodule.

Table 14-12 and **Table 14-13** summarizes the QSMCM pin functions.

Table 14-12 QSMCM_A Pin Functions

PORTQS Function	QSMCM Function
QGPI2	RXD2
QGPO2	TXD2
QGPI1	RXD1
QGPO1	TXD1
QGPIO6	SCK
QGPIO5	MOSI
QGPIO4	MISO
QGPIO3	PCS3
QGPIO2	PCS2
QGPIO1	PCS1
QGPIO0	PCS[0]

Table 14-13 QSMCM_B Pin Functions

PORTQS Function	QSMCM Function
J1850_RX	RXD2
QGPO2	TXD2
QGPI1	RXD1
QGPO1	TXD1
QGPIO6	SCK
QGPIO5	MOSI
QGPIO4	MISO

Table 14-13 QSMCM_B Pin Functions

PORTQS Function	QSMCM Function
J1850_TX	PCS3
QGPIO2	PCS2
QGPIO1	PCS1
QGPIO0	PCS[0]

PQSPAR — PORTQS Pin Assignment Register **0x30 5016**
0x30 5416

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	QPAPCS3	QPAPCS2	QPAPCS1	QPAPCS0	0	QPAMO-SI	QPAMI-SO	DDRQS*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-15**.

Table 14-14 PQSPAR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	QPAPCS3	0 = Pin is assigned QGPIO[3] in QSMCM_A or J1850_TX on QSMCM_B 1 = Pin is assigned PCS[3] function When J1850_TX is selected, B_RXD2 becomes J1850_RX
2	QPAPCS2	0 = Pin is assigned QGPIO[2] 1 = Pin is assigned PCS[2] function
3	QPAPCS1	0 = Pin is assigned QGPIO[3] 1 = Pin is assigned PCS[3]
4	QPAPCS0	0 = Pin is assigned QGPIO[0] 1 = Pin is assigned PCS[0] function
5	—	Reserved
6	QPAMOSI	0 = Pin is assigned QGPIO[5] 1 = Pin is assigned MOSI function
7	QPAMISO	0 = Pin is assigned QGPIO[4] 1 = Pin is assigned MISO function
8:15	DDRQS	PORSTQS data direction register. See **14.6.3 PORTQS Data Direction Register (DDRQS)**.

14.6.3 PORTQS Data Direction Register (DDRQS)

DDRQS assigns QSPI pin as an input or an output regardless of whether the QSPI submodule is enabled or disabled. All QSPI pins are configured during reset as general-purpose inputs.

This register does not affect SCI operation. The TXD1 and TXD2 remain output pins dedicated to the SCI submodules, and the RXD1, RXD2 and ECK pins remain input pins dedicated to the SCI submodules.

DDRQS — PORTQS Data Direction Register **0x30 5016**
0x30 5416

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PQSPAR*								0	QDDPCS3	QDDPCS2	QDDPCS1	QDDPCS0	QDDSCK	QDDMOSI	QDDMISO
RESET:															
								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 14-14**.

Table 14-15 DDRQS Bit Descriptions

Bit(s)	Name	Description
0:7	PQSPAR	PORTSQS pin assignment register. See **14.6.2 PORTQS Pin Assignment Register (PQSPAR)**.
8	—	Reserved
9	QDDPCS3	QSPI pin data direction for the pin PCS[3] 0 = Pin direction is input 1 = Pin direction is output
10	QDDPCS2	QSPI pin data direction for the pin PCS[2] 0 = Pin direction is input 1 = Pin direction is output
11	QDDPCS1	QSPI pin data direction for the pin PCS[1] 0 = Pin direction is input 1 = Pin direction is output
12	QDDPCS0	QSPI pin data direction for the pin PCS[0] 0 = Pin direction is input 1 = Pin direction is output
13	QDDSCK	QSPI pin data direction for the pin SCK 0 = Pin direction is input 1 = Pin direction is output
14	QPDMOSI	QSPI pin data direction for the pin MOSI 0 = Pin direction is input 1 = Pin direction is output
15	QPDMISO	QSPI pin data direction for the pin MISO 0 = Pin direction is input 1 = Pin direction is output

14.7 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) is used to communicate with external devices through a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The QSPI can perform full duplex three-wire or half duplex two-wire transfers. Several transfer rates, clocking, and interrupt-driven communication options are available. **Figure 14-4** is a block diagram of the QSPI.

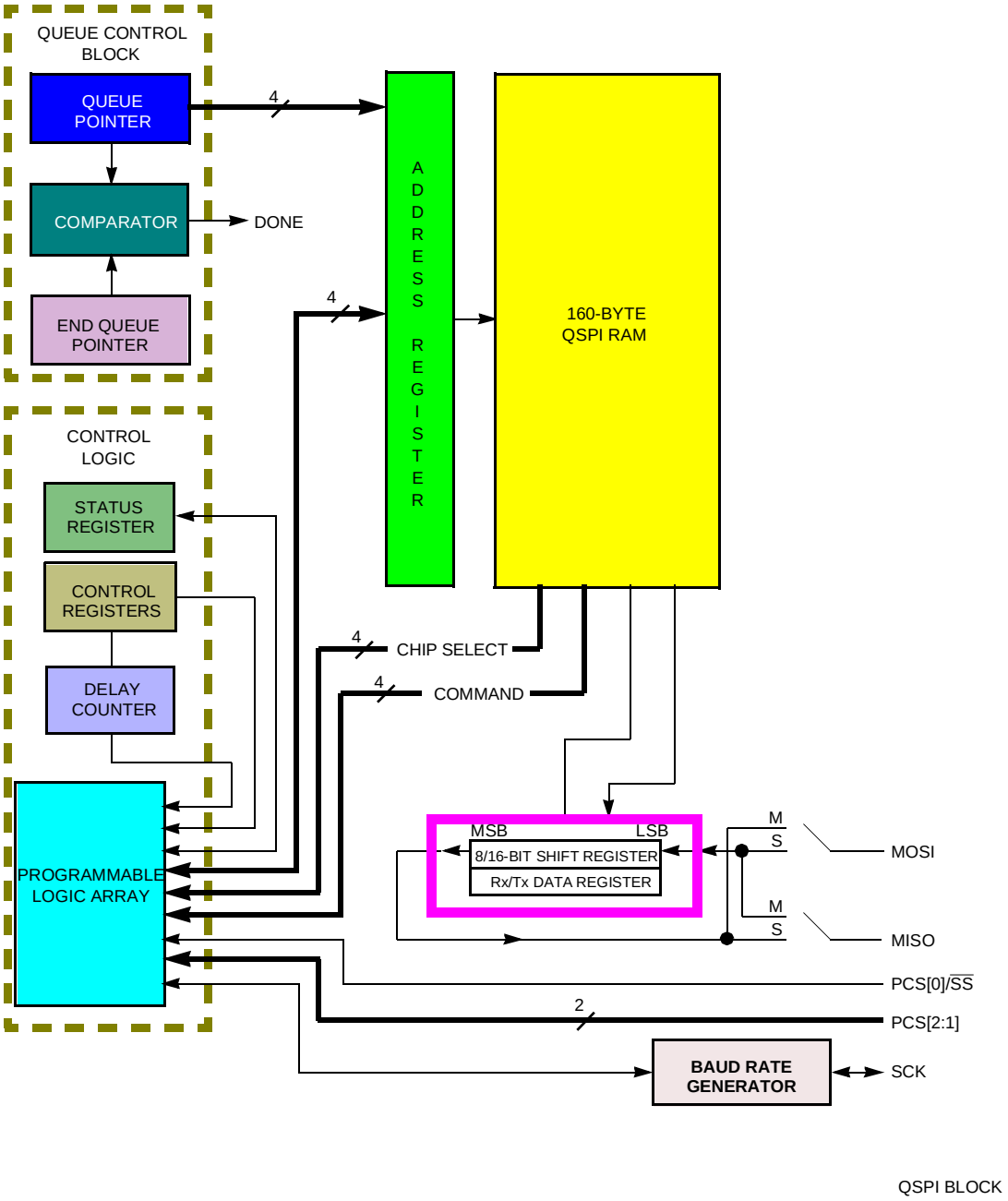


Figure 14-4 QSPI Block Diagram

Serial transfers of eight to 16 bits can be specified. Programmable transfer length simplifies interfacing to devices that require different data lengths.

An inter-transfer delay of approximately 0.8 to 204 µs (using a 40-MHz IMB3 clock) can be programmed. The default delay is 17 clocks (0.425 µs at 40 MHz). Programmable delay simplifies the interface to devices that require different delays between transfers.

A dedicated 160-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. This allows serial peripherals to be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 32 serial transfers without CPU intervention. Each queue entry contains all the information needed by the QSPI to independently complete one serial transfer.

A pointer identifies the queue location containing the data and command for the next serial transfer. Normally, the pointer address is incremented after each serial transfer, but the CPU can change the pointer value at any time. Support for multiple tasks can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. The chip-select signals simplify interfacing by reducing CPU intervention. If the chip-select signals are externally decoded, 16 independent select signals can be generated.

Wrap-around mode allows continuous execution of queued commands. In wrap-around mode, newly received data replaces previously received data in the receive RAM. Wrap-around mode can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows transfer of an uninterrupted bit stream. From eight to 512 bits can be transferred without CPU intervention. Longer transfers are possible, but minimal intervention is required to prevent loss of data. A standard delay of 17 IMB3 clocks (0.8 µs with a 40-MHz IMB3 clock) is inserted between the transfer of each queue entry.

14.7.1 QSPI Registers

The QSPI memory map, shown in **Table 14-16**, includes the QSMCM global and pin control registers, four QSPI control registers (SPCR[0:3]), the status register (SPSR), and the QSPI RAM. Registers and RAM can be read and written by the CPU. The memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **Figure 1-2** for a diagram of the MPC565 / MPC566 internal memory map.

Table 14-16 QSPI Register Map

Access[1]	Address	MSB[2]	LSB
S/U	0x30 5018(A) 0x30 5418(B)	QSPI Control Register 0 (SPCR0) See **Table 14-17** for bit descriptions.	
S/U	0x30 501A(A) 0x30 541A(B)	QSPI Control Register 1 (SPCR1) See **Table 14-19** for bit descriptions.	
S/U	0x30 501C(A) 0x30 541C(B)	QSPI Control Register 2 (SPCR2) See **Table 14-20** for bit descriptions.	
S/U	0x30 501E/ 0x30 501F(A) 0x30 541E/ 0x30 541F(B)	QSPI Control Register 3 (SPCR3) See **Table 14-21** for bit descriptions.	QSPI Status Register (SPSR) See **Table 14-22** for bit descriptions.
S/U	0x30 5140 – 0x30 517F(A) 0x30 5540 – 0x30 557F(B)	Receive Data RAM (32 half-words)	
S/U	0x30 5180 – 0x30 51BF(A) 0x30 5580 – 0x30 55BF(B)	Transmit Data RAM (32 half-words)	
S/U	0x30 51C0 – 0x30 51DF(A) 0x30 55C0 – 0x30 55DF(B)	Command RAM (32 bytes)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.

To ensure proper operation, set the QSPI enable bit (SPE) in SPCR1 only after initializing the other control registers. Setting this bit starts the QSPI.

Rewriting the same value to a control register does not affect QSPI operation with the exception of writing NEWQP in SPCR2. Rewriting the same value to these bits causes the RAM queue pointer to restart execution at the designated location.

Before changing control bits, the QSPI should be halted. Writing a different value into a control register other than SPCR2 while the QSPI is enabled may disrupt operation. SPCR2 is buffered, preventing any disruption of the current serial transfer. After the current serial transfer is completed, the new SPCR2 value becomes effective.

14.7.1.1 QSPI Control Register 0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU has read/write access to SPCR0, but the QSPI has read access only. SPCR0 must be initialized before QSPI operation begins. Writing a new value to SPCR0 while the QSPI is enabled disrupts operation.

SPCR0 — QSPI Control Register 0

0x30 5018
0x30 5418

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Table 14-17 SPCR0 Bit Descriptions

Bit(s)	Name	Description
0	MSTR	Master/slave mode select 0 = QSPI is a slave device and only responds to externally generated serial transfers. 1 = QSPI is the system master and can initiate transmission to external SPI devices.
1	WOMQ	Wired-OR mode for QSPI pins. This bit controls the QSPI pins regardless of whether they are used as general-purpose outputs or as QSPI outputs, and regardless of whether the QSPI is enabled or disabled. 0 = Pins designated for output by DDRQS operate in normal mode. 1 = Pins designated for output by DDRQS operate in open drain mode.
2:5	BITS	Bits per transfer. In master mode, when BITSE is set in a command RAM byte, BITS determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred regardless of the value in BITS. In slave mode, the BITS field always determines the number of bits the QSPI will receive during each transfer before storing the received data. Data transfers from 8 to 16 bits are supported. Illegal (reserved) values default to eight bits. **Table 14-18** shows the number of bits per transfer.
6	CPOL	Clock polarity. CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices. 0 = The inactive state of SCK is logic zero. 1 = The inactive state of SCK is logic one.
7	CPHA	Clock phase. CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. 0 = Data is captured on the leading edge of SCK and changed on the trailing edge of SCK. 1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK
8:15	SPBR	Serial clock baud rate. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU IMB3 clock. Baud rate is selected by writing a value from 2 to 255 into SPBR. The following equation determines the SCK baud rate: $\text{SCK Baud Rat} = \frac{f_{SYS}}{2 \times SPBR}$ Refer to **14.7.5.2 Baud Rate Selection** for more information.

Table 14-18 Bits Per Transfer

BITS[3:0]	Bits per Transfer
0000	16
0001 to 0111	Reserved (defaults to 8)
1000	8
1001	9
1010	10
1011	11
1100	12

Table 14-18 Bits Per Transfer (Continued)

BITS[3:0]	Bits per Transfer
1101	13
1110	14
1111	15

14.7.1.2 QSPI Control Register 1

SPCR1 enables the QSPI and specifies transfer delays. The CPU has read/write access to SPCR1, but the QSPI has read access only to all bits except SPE. SPCR1 must be written last during initialization because it contains SPE. The QSPI automatically clears this bit after it completes all serial transfers or when a mode fault occurs. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPCR1 — QSPI Control Register 1 **0x30 501A**
0x30 541A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPE	DSCKL							DTL							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 14-19 SPCR1 Bit Descriptions

Bit(s)	Name	Description
0	SPE	QSPI enable. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O. 1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.
1:7	DSCKL	Delay before SCK. When the DSCK bit is set in a command RAM byte, this field determines the length of the delay from PCS valid to SCK transition. The following equation determines the actual delay before SCK: $\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$ where DSCKL equals is in the range of 1 to 127. Refer to **14.7.5.3 Delay Before Transfer** for more information.
8:15	DTL	Length of delay after transfer. When the DT bit is set in a command RAM byte, this field determines the length of the delay after a serial transfer. The following equation is used to calculate the delay: $\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$ where DTL is in the range of 1 to 255. A zero value for DTL causes a delay-after-transfer value of $8192 \div f_{SYS}$ (204.8 µs with a 40-MHz IMB3 clock). Refer to **14.7.5.4 Delay After Transfer** for more information.

14.7.1.3 QSPI Control Register 2

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSPI has read access only. Writes to this register are buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not the buffer.

SPCR2 — QSPI Control Register 2 **0x30 501C**
0x30 541C

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPIFIE	WREN	WRTO	ENDQP					Reserved			NEWQP				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-20 SPCR2 Bit Descriptions

Bit(s)	Name	Description
0	SPIFIE	SPI finished interrupt enable. Refer to **14.7.4.2 QSPI Interrupts**. 0 = QSPI interrupts disabled 1 = QSPI interrupts enabled
1	WREN	Wrap enable. Refer to **14.7.5.7 Master Wraparound Mode**. 0 = Wraparound mode disabled. 1 = Wraparound mode enabled.
2	WRTO	Wrap to. When wraparound mode is enabled and after the end of queue has been reached, WRTO determines which address the QSPI executes next. The end of queue is determined by an address match with ENDQP. 0 = Wrap to pointer address 0x0 1 = Wrap to address in NEWQP
3:7	ENDQP	Ending queue pointer. This field determines the last absolute address in the queue to be completed by the QSPI. After completing each command, the QSPI compares the queue pointer value of the just-completed command with the value of ENDQP. If the two values match, the QSPI sets SPIF to indicate it has reached the end of the programmed queue. Refer to **14.7.4 QSPI Operation** for more information.
8:10	—	Reserved
11:15	NEWQP	New queue pointer value. This field contains the first QSPI queue address. Refer to **14.7.4 QSPI Operation** for more information.

14.7.1.4 QSPI Control Register 3

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enable, and the halt control bit. The CPU has read/write access to SPCR3, but the QSPI has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation.

SPCR3 — QSPI Control Register **0x30 501E**
0x30 541E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Reserved					LOOPQ	HMIE	HALT	SPSR*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 14-22**.

Table 14-21 SPCR3 Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5	LOOPQ	QSPI loop mode. LOOPQ controls feedback on the data serializer for testing. 0 = Feedback path disabled. 1 = Feedback path enabled.
6	HMIE	HALTA and MODF interrupt enable. HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR. 0 = HALTA and MODF interrupts disabled. 1 = HALTA and MODF interrupts enabled.
7	HALT	Halt QSPI. When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted. Refer to **14.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI operates normally. 1 = QSPI is halted for subsequent restart.
8:15	—	SPSR. See**Table 14-22** for bit descriptions.

14.7.1.5 QSPI Status Register

The SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in this register. To clear status flags, the CPU reads SPSR with the flags set and then writes the SPSR with zeros in the appropriate bits. Writes to CPTQP have no effect.

SPSR — QSPI Status Register **0x30 501E[1]**
0x30 541E[1]

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
SPCR3[2]								SPIF	MODF	HALTA	CPTQP				
								0	0	0	0	0	0	0	0

NOTES:
1. SPSR can be accessed as an 8-bit register at location 0x30 501F or 0x30 541F.
2. See bit descriptions in **Table 14-21**.

Table 14-22 SPSR* Bit Descriptions

Bit(s)	Name	Description
0:7	SPCR3	See bit descriptions in **Table 14-21**.
8	SPIF	QSPI finished flag. SPIF is set after execution of the command at the address in ENDQP in SPCR2. If wraparound mode is enabled (WREN = 1), the SPIF is set, after completion of the command defined by ENDQP, each time the QSPI cycles through the queue. 0 = QSPI is not finished 1 = QSPI is finished
9	MODF	Mode fault flag. The QSPI asserts MODF when the QSPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver. Refer to **14.7.8 Mode Fault** for more information. 0 = Normal operation 1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).
10	HALTA	Halt acknowledge flag. HALTA is set when the QSPI halts in response to setting the HALT bit in SPCR3. HALTA is also set when the IMB3 FREEZE signal is asserted, provided the FRZ1 bit in the QSMCMMCR is set. To prevent undefined operation, no modification should be made to any QSPI control registers or RAM while the QSPI is halted. If HMIE in SPCR3 is set the QSPI sends interrupt requests to the CPU when HALTA is asserted. 0 = QSPI is not halted. 1 = QSPI is halted
11:15	CPTQP	Completed queue pointer. CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value 0x0 or a pointer to the last command completed in the previous queue. If the QSPI is halted, CPTQP may be used to determine which commands have not been executed. The CPTQP may also be used to determine which locations in the receive data segment of the QSPI RAM contain valid received data.

14.7.2 QSPI RAM

The QSPI contains a 160-byte block of dual-ported static RAM that can be accessed by both the QSPI and the CPU. Because of this dual access capability, up to two wait states may be inserted into CPU access time if the QSPI is in operation.

The size and type of access of the QSPI RAM by the CPU affects the QSPI access time. The QSPI allows byte, half-word, and word accesses. Only word accesses of the RAM by the CPU are coherent because these accesses are an indivisible operation. If the CPU makes a coherent access of the QSPI RAM, the QSPI cannot access the QSPI RAM until the CPU is finished. However, a word or misaligned word access is not coherent because the CPU must break its access of the QSPI RAM into two parts, which allows the QSPI to access the QSPI RAM between the two accesses by the CPU.

The RAM is divided into three segments: receive data RAM, transmit data RAM, and command data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored for transmission to an external device. Command data defines transfer parameters. **Figure 14-5** shows RAM organization.

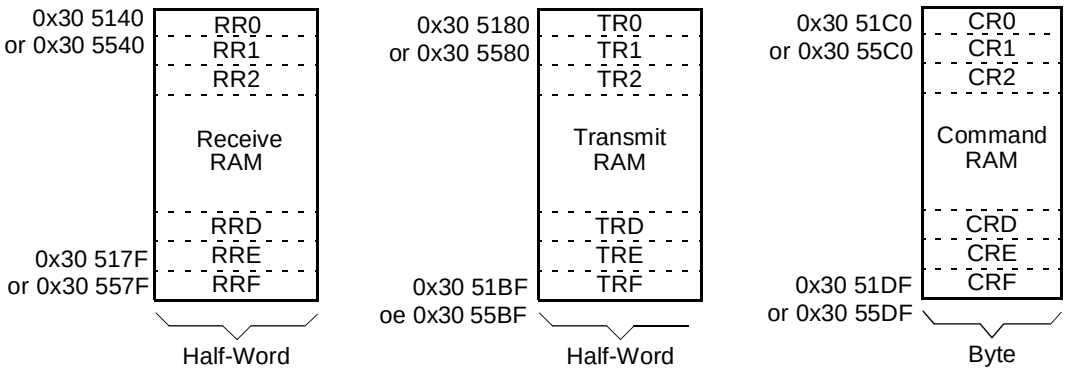


Figure 14-5 QSPI RAM

14.7.2.1 Receive RAM

Data received by the QSPI is stored in this segment, to be read by the CPU. Data stored in the receive RAM is right-justified, (i.e., the least significant bit is always in the right-most bit position within the word regardless of the serial transfer length). Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, half-word, or word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

14.7.2.2 Transmit RAM

Data that is to be transmitted by the QSPI is stored in this segment. The CPU normally writes one word of data into this segment for each queue command to be executed. If the corresponding peripheral, such as a serial input port, is used solely to input data, then this segment does not need to be initialized.

Data must be written to transmit RAM in a right-justified format. The QSPI cannot modify information in the transmit RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

14.7.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 32 bytes. Each byte is divided into two fields. The peripheral chip-select field, enables peripherals for transfer. The command control field provides transfer options.

A maximum of 32 commands can be in the queue. These bytes are assigned an address from 0x00 to 0x1F. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CR[0:F] — Command RAM **0x30 51C0 – 0x30 51DF**
0x30 55C0 – 0x30 55DF

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
Command Control				Peripheral Chip Select			

NOTES:
1. The PCS[0] bit represents the dual-function PCS[0]/$\overline{SS}$.

Table 14-23 Command RAM Bit Descriptions

Bit(s)	Name	Description
0	CONT	Continue 0 = Control of chip selects returned to PORTQS after transfer is complete. 1 = Peripheral chip selects remain asserted after transfer is complete.
1	BITSE	Bits per transfer enable 0 = Eight bits 1 = Number of bits set in BITS field of SPCR0.
2	DT	Delay after transfer 0 = Delay after transfer is $17 \div f_{SYS}$. 1 = SPCR1 DTL[7:0] specifies delay after transfer PCS valid to SCK.
3	DSCK	PCS to SCK Delay 0 = PCS valid to SCK delay is one-half SCK. 1 = SPCR1 DSCKL[6:0] specifies delay from PCS valid to SCK.
4:7	PCS[3:0]	Peripheral chip selects. Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select may be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided proper fanout is observed. PCS[0] shares a pin with the slave select ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

Refer to **14.7.5 Master Mode Operation** for more information on the command RAM.

14.7.3 QSPI Pins

Seven pins are associated with the QSPI. When not needed by the QSPI, they can be configured for general-purpose I/O. **Table 14-24** identifies the QSPI pins and their functions. Register DDRQS determines whether the pins are designated as input or output. The user must initialize DDRQS for the QSPI to function correctly.

Table 14-24 QSPI Pin Functions

Pin Names	Mnemonic	Mode	Function
Master in slave out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master out slave in	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial clock	SCK[1]	Master Slave	Clock output from QSPI clock Input to QSPI
Peripheral chip selects	PCS[1:3]	Master	Outputs select peripheral(s)
Peripheral chip select[2] Slave select[3]	PCS[0]/ $\overline{SS}$	Master Slave	Output selects peripheral(s) Input selects the QSPI
Slave select[4]	$\overline{SS}$	Master	May cause mode fault

NOTES:
1. All QSPI pins (except SCK) can be used as general-purpose I/O if they are not used by the QSPI while the QSPI is operating. SCK can only be used for general-purpose I/O if the QSPI is disabled.
2. An output (PCS[0]) when the QSPI is in master mode.
3. An input ($\overline{SS}$) when the QSPI is in slave mode.
4. An input ($\overline{SS}$) when the QSPI is in master mode; useful in multimaster systems.

14.7.4 QSPI Operation

The QSPI uses a dedicated 160-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments: 32 command control bytes, 64 transmit data bytes, and 64 receive data bytes.

Once the CPU has set up a queue of QSPI commands, written the transmit data segment with information to be sent, and enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating completion, and then either interrupts the CPU or waits for CPU intervention.

QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to each queue entry, 0x0 to 0x2F.

The CPU initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU or waits for intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

The internal pointer is initialized to the same value as NEWQP. During normal operation, the command pointed to by the internal pointer is executed, the value in the internal pointer is copied into CPTQP, the internal pointer is incremented, and then the

sequence repeats. Execution continues at the internal pointer address unless the NEWQP value is changed. After each command is executed, ENDQP and CPTQP are compared. When a match occurs, the SPIF flag is set and the QSPI stops and clears SPE, unless wraparound mode is enabled.

At reset, NEWQP is initialized to 0x0. When the QSPI is enabled, execution begins at queue address 0x0 unless another value has been written into NEWQP. ENDQP is initialized to 0x0 at reset but should be changed to the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When NEWQP changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to 0x0 transfers only the data in transmit RAM location 0x0.

14.7.4.1 Enabling, Disabling, and Halting the SPI

The SPE bit in the SPCR1 enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS[0]/$\overline{SS}$ pin to respond to the external initialization of a serial transfer.

When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access to all 160 bytes of the QSPI RAM. The QSPI can read-only the transmit data segment and the command control segment and can write-only the receive data segment of the QSPI RAM.

The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS[0]/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS[0]/$\overline{SS}$ is driven low externally.

Setting the HALT bit in SPCR3 stops the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is set, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip select pins with the value designated by the last command before the halt. If CONT was cleared, the QSPI drives the peripheral chip-select pins to the value in register PORTQS.

If HALT is set during the last command in the queue, the QSPI completes the last command, sets both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF or clear SPE. QSPI execution continues when the CPU clears HALT.

To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE can then be safely cleared, providing an orderly method of shutting down the QSPI quickly after the current serial transfer is completed. The CPU can disable the QSPI immediately by clearing SPE. However, loss of data from a current serial transfer may result and confuse an external SPI device.

14.7.4.2 QSPI Interrupts

The QSPI has three possible interrupt sources but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the interrupt cause must ascertained by reading the SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the interrupt source.

The SPIFIE bit in SPCR2 enables the QSPI to generate an interrupt request upon assertion of the SPIF status flag. Because it is buffered, the value written to SPIFIE applies only upon completion of the queue (the transfer of the entry indicated by ENDPQ). Thus, if a single sequence of queue entries is to be transferred (i.e., no WRAP), then SPIFIE should be set to the desired state before the first transfer.

If a sub-queue is to be used, the same CPU write that causes a branch to the sub-queue may enable or disable the SPIF interrupt for the sub-queue. The primary queue retains its own selected interrupt mode, either enabled or disabled.

The SPIF interrupt must be cleared by clearing SPIF. Subsequent interrupts may then be prevented by clearing SPIFIE. Clearing SPIFIE does not immediately clear an interrupt already caused by SPIF.

14.7.4.3 QSPI Flow

The QSPI operates in either master or slave mode. Master mode is used when the MCU initiates data transfers. Slave mode is used when an external device initiates transfers. Switching between these modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSMCM and QSPI registers must be initialized properly.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from the transmit RAM and received by the receive RAM.

In slave mode, operation proceeds in response to SS pin assertion by an external SPI bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multi-master operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

Figure 14-6 shows QSPI initialization. **Figure 14-7** through show QSPI master and slave operation. The CPU must initialize the QSMCM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation. The command queue must be written before the QSPI is enabled for master mode opera-

tion. Any data to be transmitted should be written into transmit RAM before the QSPI is enabled. During wraparound operation, data for subsequent transmissions can be written at any time.

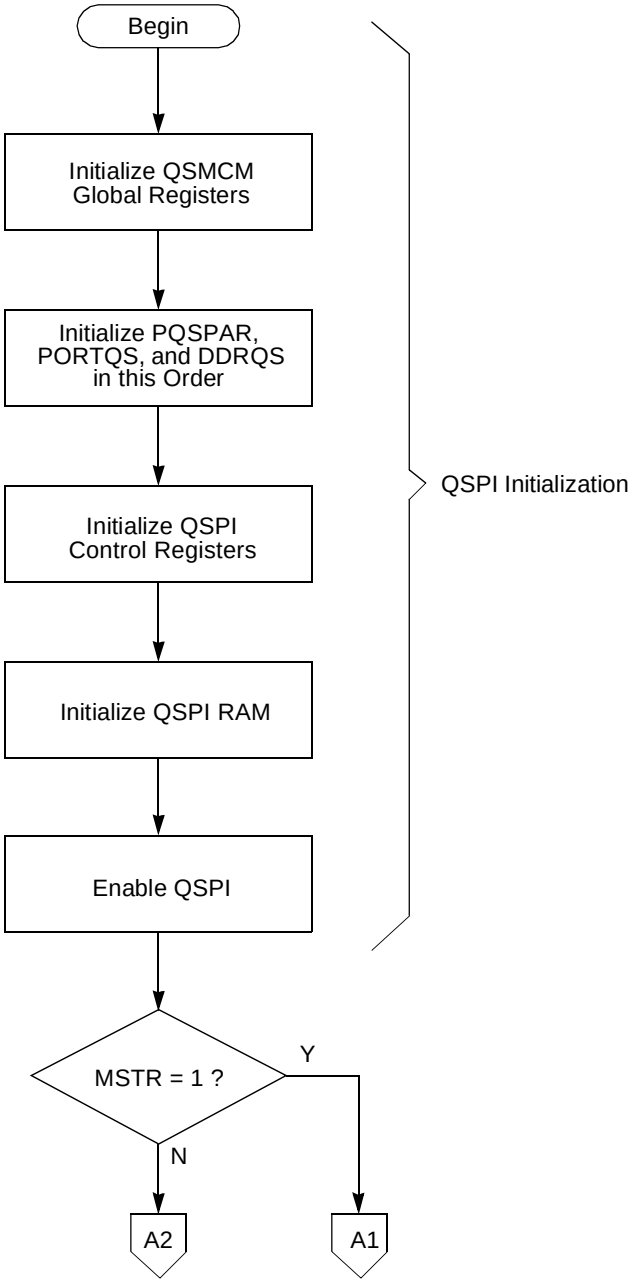


Figure 14-6 Flowchart of QSPI Initialization Operation

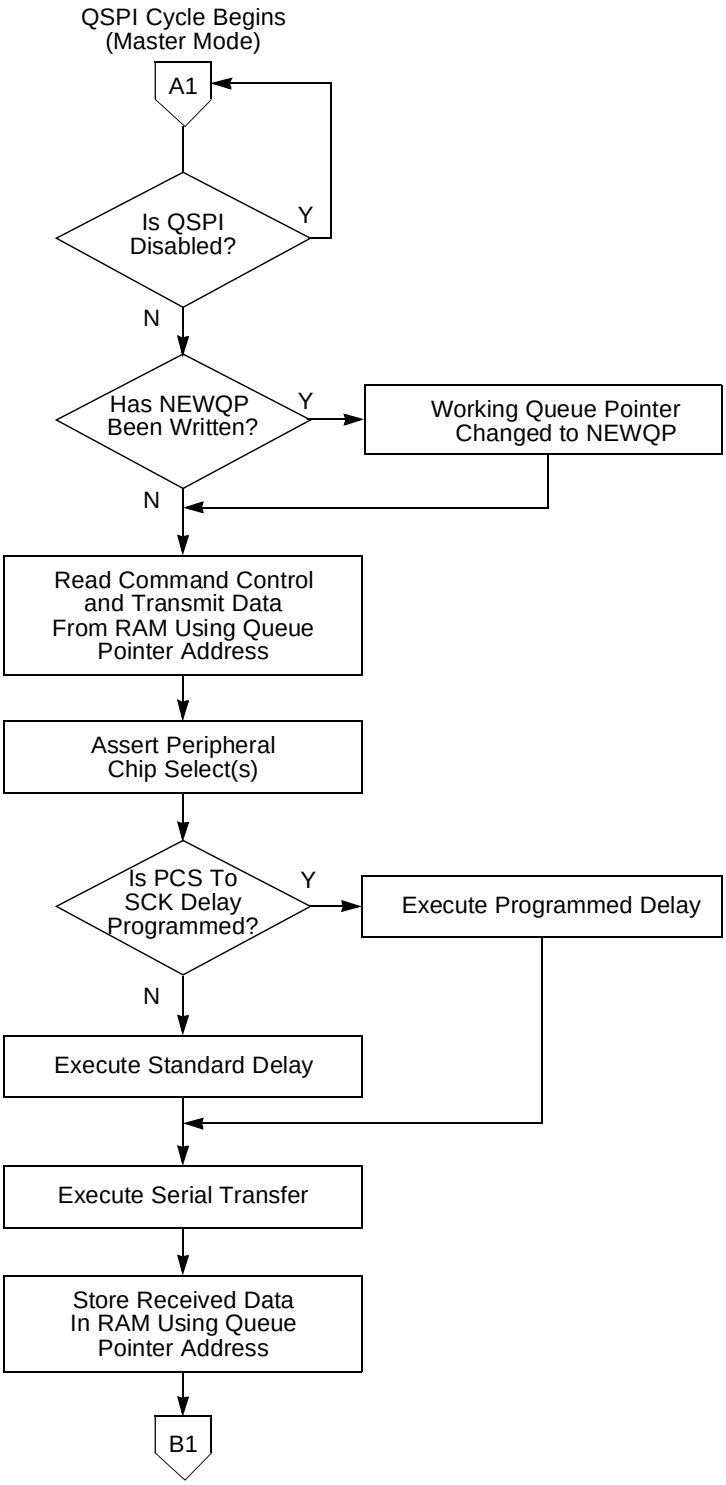
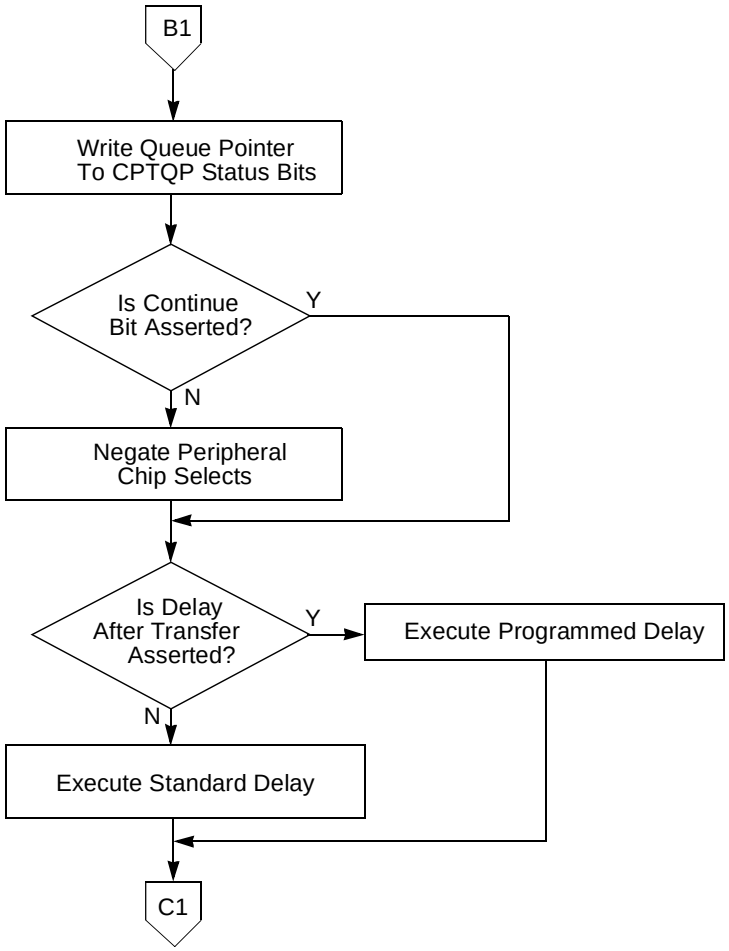
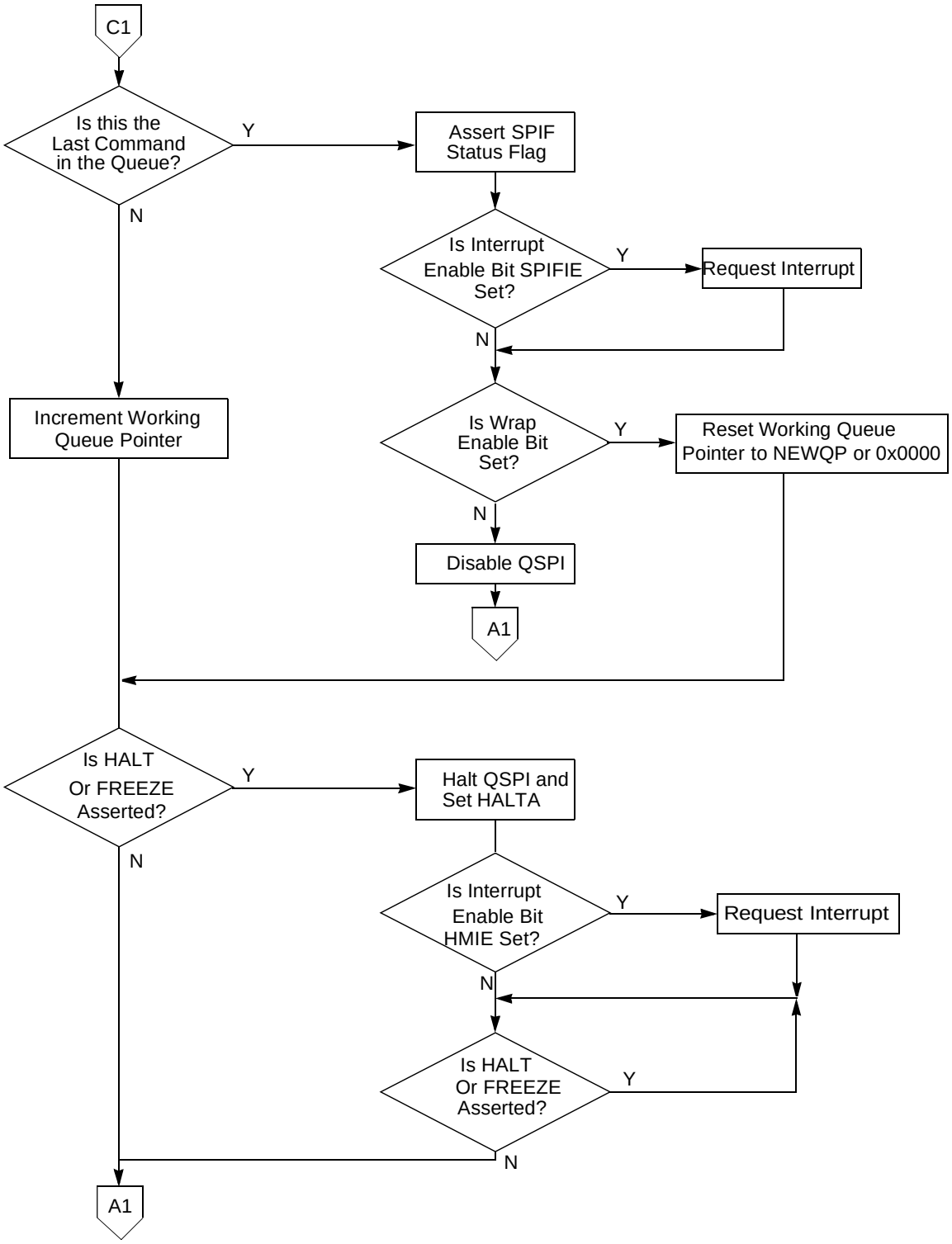


Figure 14-7 Flowchart of QSPI Master Operation (Part 1)



Flowchart of QSPI Master Operation (Part 2)



Flowchart of QSPI Master Operation (Part 3)

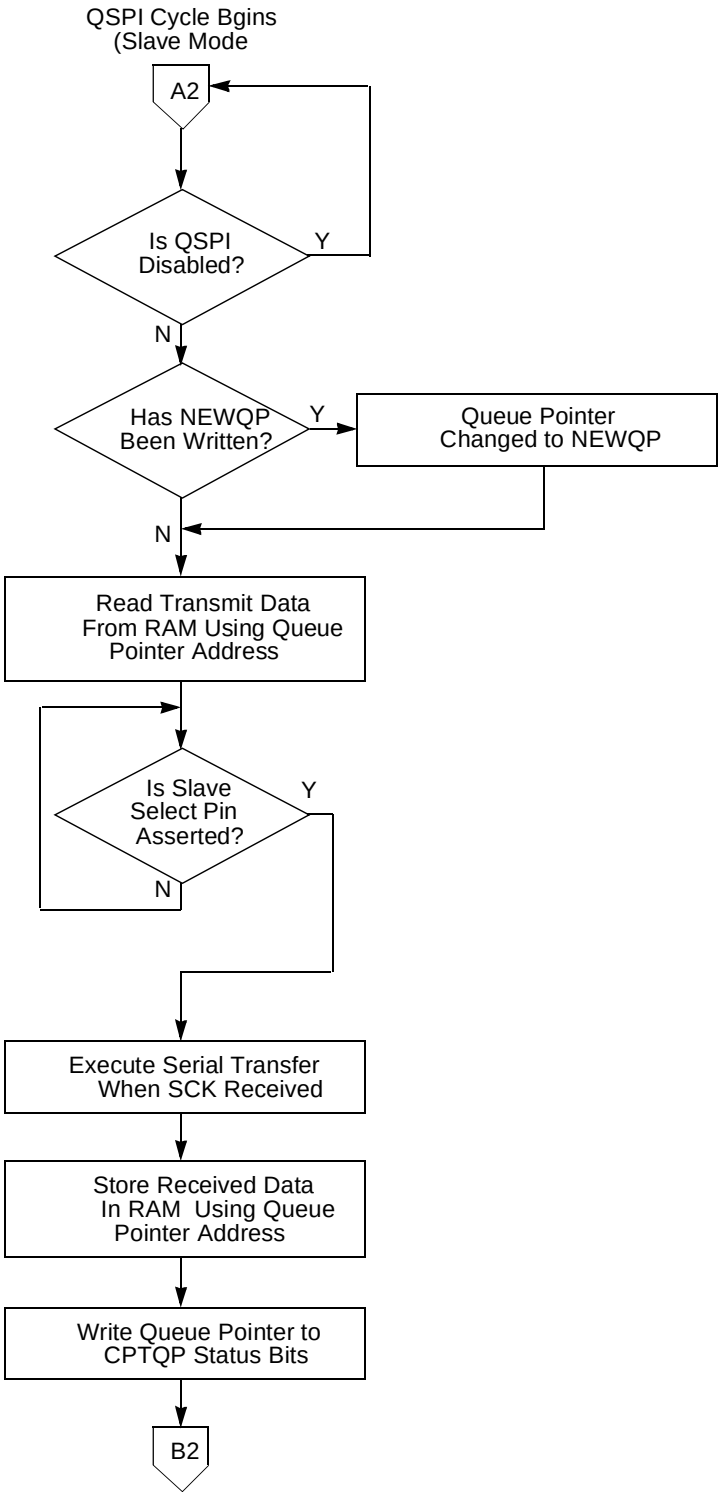
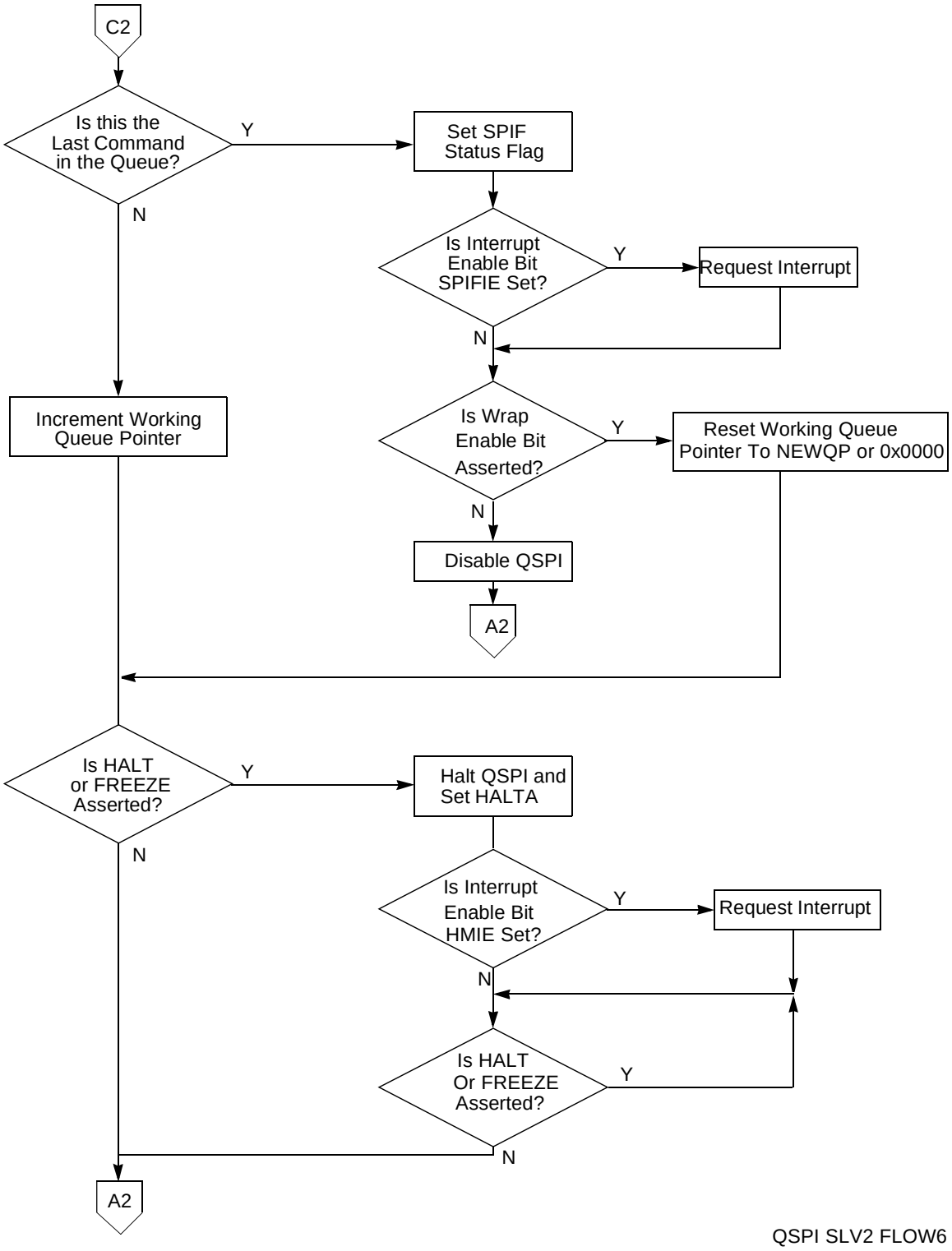


Figure 14-8 Flowchart of QSPI Slave Operation (Part 1)



Flowchart of QSPI Slave Operation (Part 2)

Normally, the SPI bus performs synchronous bidirectional transfers. The serial clock on the SPI bus master supplies the clock signal SCK to time the transfer of data. Four

possible combinations of clock phase and polarity can be specified by the CPHA and CPOL bits in SPCR0.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but can be set to any value from eight to sixteen bits by writing a value into the BITS field in SPCR0 and setting BITSE in command RAM.

Typically, SPI bus outputs are not open drain unless multiple SPI masters are in the system. If needed, the WOMQ bit in SPCR0 can be set to provide wired-OR, open drain outputs. An external pull-up resistor should be used on each output line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

14.7.5 Master Mode Operation

Setting the MSTR bit in SPCR0 selects master mode operation. In master mode, the QSPI can initiate serial transfers, but cannot respond to externally initiated transfers. When the slave select input of a device configured for master mode is asserted, a mode fault occurs.

Before QSPI operation begins, PQSPAR must be written to assign the necessary pins to the QSPI. The pins necessary for master mode operation are MISO, MOSI, SCK, and one or more of the chip-select pins. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. SCK is the serial clock output in master mode and must be assigned to the QSPI for proper operation.

The PORTQS data register must next be written with values that make the QGPIO[6]/SCK (bit 13 QDSCK of PORTQS) and QGPIO[3:0]/PCS[3:0] (bits 12:9 QDPCS[3:0] of PORTQS) outputs inactive when the QSPI completes a series of transfers. Pins allocated to the QSPI by PQSPAR are controlled by PORTQS when the QSPI is inactive. PORTQS I/O pins driven to states opposite those of the inactive QSPI signals can generate glitches that momentarily enable or partially clock a slave device.

For example, if a slave device operates with an inactive SCK state of logic one (CPOL = 1) and uses active low peripheral chip-select PCS[0], the QDSCK and QDPCS[0] bits in PORTQS must be set to 0b11. If QDSCK and QDPCS0 = 0b00, falling edges will appear on QGPIO[6]/SCK and GPIO[0]/PCS[0] as the QSPI relinquishes control of these pins and PORTQS drives them to logic zero from the inactive SCK and PCS[0] states of logic one.

Before master mode operation is initiated, QSMCM register DDRQS is written last to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0] as outputs. The MISO pin must be configured as an input.

After pins are assigned and configured, write appropriate data to the command queue. If data is to be transmitted, write the data to transmit RAM. Initialize the queue pointers as appropriate.

QSPI operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in a command RAM byte is set, PCS pins are continuously driven to specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

14.7.5.1 Clock Phase and Polarity

In master mode, data transfer is synchronized with the internally-generated serial clock SCK. Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

14.7.5.2 Baud Rate Selection

Baud rate is selected by writing a value from two to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU IMB3 clock.

The following expressions apply to the SCK baud rate:

$$\text{SCK Baud Rat} = \frac{f_{SYS}}{2 \times \text{SPBR}}$$

or

$$\text{SPBR} = \frac{f_{SYS}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state. At reset, the SCK baud rate is initialized to one eighth of the IMB3 clock frequency.

Table 14-25 provides some example SCK baud rates with a 40-MHz IMB3 clock.

Table 14-25 Example SCK Frequencies with a 40-MHz IMB3 clock

Division Ratio	SPBR Value	SCK Frequency
4	2	10.00 MHz
6	3	6.67 MHz
8	4	5.00 MHz
14	7	2.86 MHz
28	14	1.43 MHz
58	29	689 KHz
280	140	143 KHz
510	255	78.43 KHz

14.7.5.3 Delay Before Transfer

The DSCK bit in each command RAM byte inserts either a standard (DSCK = 0) or user-specified (DSCK = 1) delay from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the length of the user-defined delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$$

where DSCKL is in the range from 1 to 127.

NOTE

A zero value for DSCKL causes a delay of 128 IMB3 clocks, which equals 3.2 µs for a 40-MHz IMB3 clock. Because of design limits, a DSCKL value of one defaults to the same timing as a value of two.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the SCK period.

14.7.5.4 Delay After Transfer

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in each command RAM byte determines whether the standard delay period (DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$$

where DTL is in the range from one to 255.

A zero value for DTL causes a delay-after-transfer value of 8192 ÷ IMB3 clock frequency (204.8 µs with a 40-MHz IMB3 clock).

If DT is zero in a command RAM byte, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = \frac{17}{f_{SYS}}$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the IMB3 clock is operating at a slower rate, the delay between transfers must be increased proportionately.

14.7.5.5 Transfer Length

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value from eight (0b1000) to 16 (0b0000) bits, inclusive. Reserved values (from 0b0001 to 0b0111) default to eight bits. The programmed value must be written into the BITS field in SPCR0. The BITSE bit in each command RAM byte determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used.

14.7.5.6 Peripheral Chip Selects

Peripheral chip-select signals are used to select an external device for serial data transfer. Chip-select signals are asserted when a command in the queue is executed. Signals are asserted at a logic level corresponding to the value of the PCS[3:0] bits in each command byte. More than one chip-select signal can be asserted at a time, and more than one external device can be connected to each PCS pin, provided proper fanout is observed. PCS[0] shares a pin with the slave select SS signal, which initiates slave mode serial transfer. If SS is taken low when the QSPI is in master mode, a mode fault occurs.

To configure a peripheral chip select, set the appropriate bit in PQSPAR, then configure the chip-select pin as an output by setting the appropriate bit in DDRQS. The value of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If the base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared dur-

ing reset. If no new data is written to PORTQS before pin assignment and configuration as an output, the base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

14.7.5.7 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

Wraparound mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

14.7.6 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external SPI bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSMCM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO, MOSI, SCK, and PCS[0]/SS. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the serial clock input in slave mode and must be assigned to the QSPI for proper operation. Assertion of the active-low slave select signal SS initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS[0]/SS pins as inputs. The MISO pin must be configured as an output.

After pins are assigned and configured, write data to be transmitted into transmit RAM. Command RAM is not used in slave mode, and does not need to be initialized. Set the queue pointers, as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select PCS[0]/SS pin begins slave mode operation at the address indicated by NEWQP. Data that is received is stored at the pointer address in receive RAM. Data is simultaneously loaded into the data serializer from the pointer address in transmit RAM and transmitted. Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine upon which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI pin.

Because the command RAM is not used in slave mode, the CONT, BITSE, DT, DSCK, and peripheral chip-select bits have no effect. The PCS[0]/$\overline{SS}$ pin is used only as an input.

The SPBR, DT and DSCKL fields in SPCR0 and SPCR1 bits are not used in slave mode. The QSPI drives neither the clock nor the chip-select pins and thus cannot control clock rate or transfer delay.

Because the BITSE option is not available in slave mode, the BITS field in SPCR0 specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS[3:0] has been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads new transmit data from transmit RAM into the data serializer. The working queue pointer address is used the next time PCS[0]/$\overline{SS}$ is asserted, unless the RCPU writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If $\overline{SS}$ goes high before the number of bits specified by the BITS field is transferred, the QSPI resumes operation at the same pointer address the next time $\overline{SS}$ is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before $\overline{SS}$ is negated, pointers are incremented and operation continues.

The QSPI transmits as many bits as it receives at each queue address, until the BITS value is reached or $\overline{SS}$ is negated. $\overline{SS}$ does not need to go high between transfers as the QSPI transfers data until reaching the end of the queue, whether $\overline{SS}$ remains low or is toggled between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2. Slave wraparound operation is identical to master wraparound operation.

14.7.6.1 Description of Slave Operation

After reset, the QSMCM registers and the QSPI control registers must be initialized as described above. Although the command control segment is not used, the transmit and receive data segments may, depending upon the application, need to be initial-

ized. If meaningful data is to be sent out from the QSPI, the data to the transmit data should be written to the segment before enabling the QSPI.

If SPE is set and MSTR is not set, a low state on the slave select (PCS[0]/$\overline{SS}$) pin commences slave mode operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the address indicated by NEWQP. Data is transferred in response to an external slave clock input at the SCK pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The QSPI does not drive any of the four peripheral chip-selects as outputs. PCS[0]/$\overline{SS}$ is used as an input.

Although CONT cannot be used in slave mode, a provision is made to enable receipt of more than 16 data bits. While keeping the QSPI selected (PCS[0]/$\overline{SS}$ is held low), the QSPI stores the number of bits, designated by BITS, in the current receive data segment address, increments NEWQP, and continues storing the remaining bits (up to the BITS value) in the next receive data segment address.

As long as PCS[0]/$\overline{SS}$ remains low, the QSPI continues to store the incoming bit stream in sequential receive data segment addresses, until either the value in BITS is reached or the end-of-queue address is used with wraparound mode disabled.

When the end of the queue is reached, the SPIF flag is asserted, optionally causing an interrupt. If wraparound mode is disabled, any additional incoming bits are ignored.

If wraparound mode is enabled, storing continues at either address 0x0 or the address of NEWQP, depending on the WRTO value. When using this capability to receive a long incoming data stream, the proper delay between transfers must be used. The QSPI requires time, approximately 0.425 µs with a 40-MHz IMB3 clock, to prefetch the next transmit RAM entry for the next transfer. Therefore, a baud rate may selected that provides at least a 0.6-µs delay between successive transfers to ensure no loss of incoming data. If the IMB3 clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Because the BITSE option in the command control segment is no longer available, BITS sets the number of bits to be transferred for all transfers in the queue until the CPU changes the BITS value. As mentioned above, until PCS[0]/$\overline{SS}$ is negated (brought high), the QSPI continues to shift one bit for each pulse of SCK. If PCS[0]/$\overline{SS}$ is negated before the proper number of bits (according to BITS) is received, the next time the QSPI is selected it resumes storing bits in the same receive-data segment address where it left off. If more than 16 bits are transferred before negating the PCS[0]/$\overline{SS}$, the QSPI stores the number of bits indicated by BITS in the current receive data segment address, then increments the address and continues storing as described above.

NOTE

PCS[0]/$\overline{SS}$ does not necessarily have to be negated between transfers.

Once the proper number of bits (designated by BITS) are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the new transmit data from the transmit data segment into the data serializer. The internal working queue pointer address is used the next time PCS[0]/$\overline{SS}$ is asserted, unless the CPU writes to the NEWQP first.

The DT and DSCK command control bits are not used in slave mode. As a slave, the QSPI does not drive the clock line nor the chip-select lines and, therefore, does not generate a delay.

In slave mode, the QSPI shifts out the data in the transmit data segment. The transmit data is loaded into the data serializer (refer to **Figure 14-1**) for transmission. When the PCS[0]/$\overline{SS}$ pin is pulled low the MISO pin becomes active and the serializer then shifts the 16 bits of data out in sequence, most significant bit first, as clocked by the incoming SCK signal. The QSPI uses CPHA and CPOL to determine which incoming SCK edge the MOSI pin uses to latch incoming data, and which edge the MISO pin uses to drive the data out.

The QSPI transmits and receives data until reaching the end of the queue (defined as a match with the address in ENDQP), regardless of whether PCS[0]/$\overline{SS}$ remains selected or is toggled between serial transfers. Receiving the proper number of bits causes the received data to be stored. The QSPI always transmits as many bits as it receives at each queue address, until the BITS value is reached or PCS[0]/$\overline{SS}$ is negated.

14.7.7 Slave Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag, whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE bit can be found in **14.7.1.3 QSPI Control Register 2**.

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF, it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE.

As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR with SPIF asserted, followed by a write to SPSR with zero in SPIF (clear SPIF). Execution continues in wraparound mode even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer is incremented to the next address and the com-

mands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data located in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wraparound mode by clearing WREN. The next time end of the queue is reached, the QSPI sets SPIF, clears SPE, and stops; and, b) The CPU sets HALT. This second method halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended, as it causes the QSPI to abort a serial transfer in process.

14.7.8 Mode Fault

MODF is asserted by the QSPI when the QSPI is the serial master (MSTR = 1) and the slave select (PCS[0]/$\overline{SS}$) input pin is pulled low by an external driver. This is possible only if the PCS[0]/$\overline{SS}$ pin is configured as input by QDDR. This low input to $\overline{SS}$ is not a normal operating condition. It indicates that a multimaster system conflict may exist, that another MCU is requesting to become the SPI network master, or simply that the hardware is incorrectly affecting PCS[0]/$\overline{SS}$. SPE in SPCR1 is cleared, disabling the QSPI. The QSPI pins revert to control by QPDR. If MODF is set and HMIE in SPCR3 is asserted, the QSPI generates an interrupt to the CPU.

The CPU may clear MODF by reading SPSR with MODF asserted, followed by writing SPSR with a zero in MODF. After correcting the mode fault problem, the QSPI can be re-enabled by asserting SPE.

The PCS[0]/$\overline{SS}$ pin may be configured as a general-purpose output instead of input to the QSPI. This inhibits the mode fault checking function. In this case, MODF is not used by the QSPI.

14.8 Serial Communication Interface

The dual, independent, serial communication interface (DSCI) communicates with external devices through an asynchronous serial bus. The two SCI modules are functionally equivalent, except that the SCI1 also provides 16-deep queue capabilities for the transmit and receive operations. The SCIs are fully compatible with other Motorola SCI systems. The DSCI has all of the capabilities of previous SCI systems as well as several significant new features.

Figure 14-9 is a block diagram of the SCI transmitter. **Figure 14-10** is a block diagram of the SCI receiver.

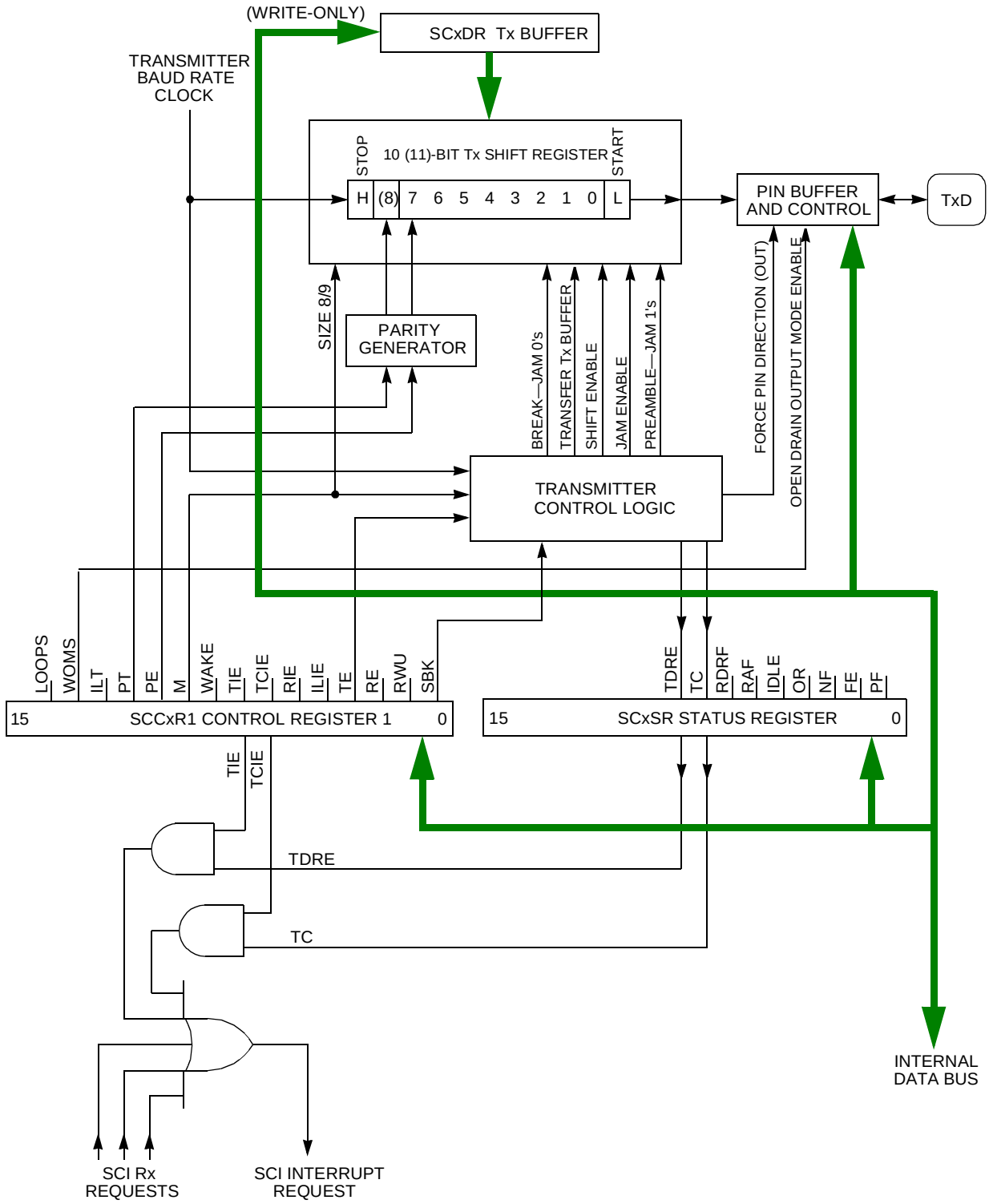


Figure 14-9 SCI Transmitter Block Diagram

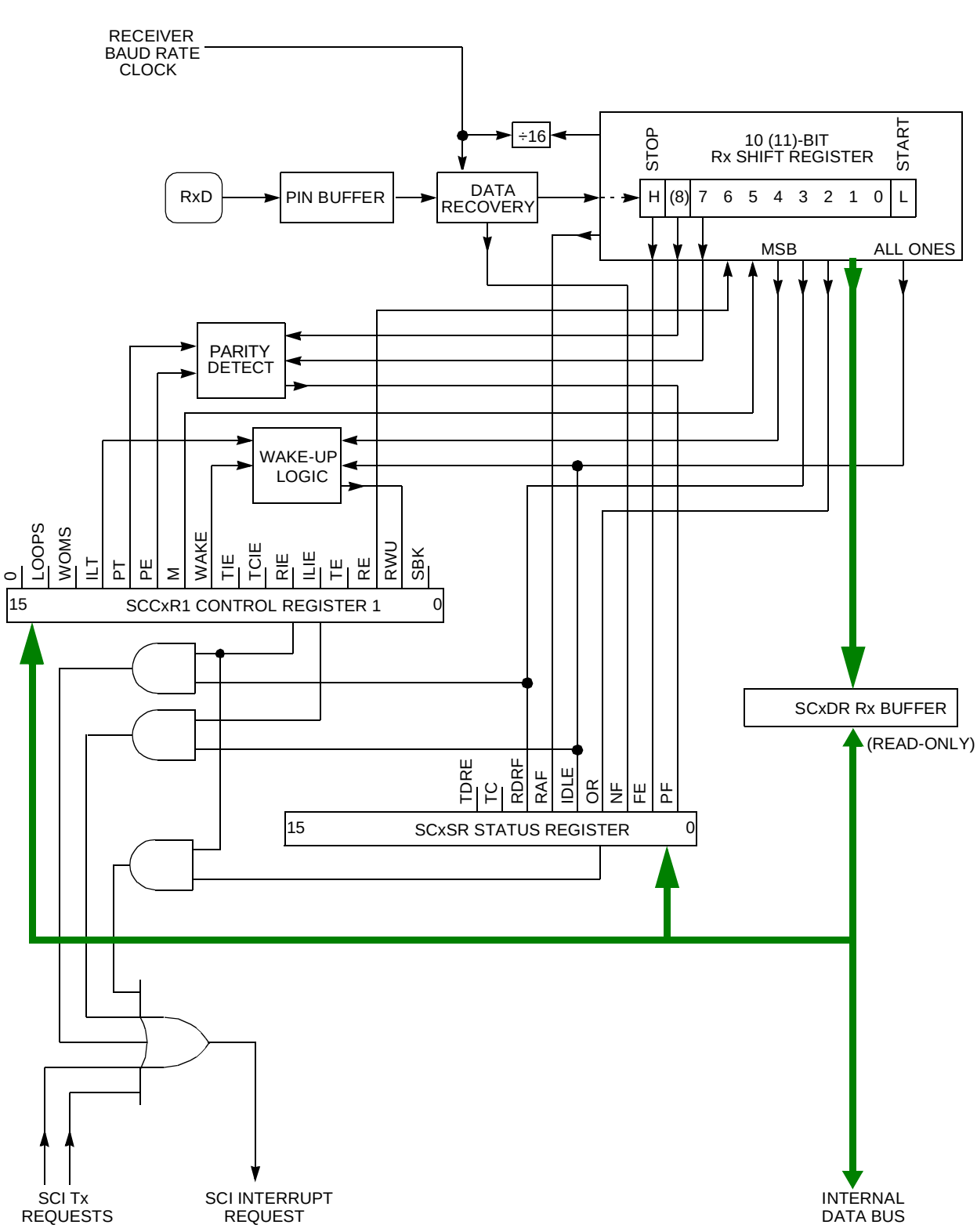


Figure 14-10 SCI Receiver Block Diagram

14.8.1 SCI Registers

The SCI programming model includes the QSMCM global and pin control registers and the DSCI registers.

The DSCI registers, listed in **Table 14-26**, consist of five control registers, three status registers, and 34 data registers. All registers may be read or written at any time by the CPU. Rewriting the same value to any DSCI register does not disrupt operation; however, writing a different value into a DSCI register when the DSCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in register SCxSR can be cleared at any time.

Table 14-26 SCI Registers

Address	Name	Usage
0x30 5008(A) 0x30 5408(B)	SCC1R0	SCI1 Control Register 0 See **Table 14-27** for bit descriptions.
0x30 500A(A) 0x30 540A(B)	SCC1R1	SCI1 Control Register 1 See **Table 14-28** for bit descriptions.
0x30 500C(A) 0x30 540C(B)	SC1SR	SCI1 Status Register See **Table 14-29** for bit descriptions.
0x30 500E(A) 0x30 540E(B) (non-queue mode only)	SC1DR	SCI1 Data Register Transmit Data Register (TDR1)* Receive Data Register (RDR1)* See **Table 14-30** for bit descriptions.
0x30 5020(A) 0x30 5420(B)	SCC2R0	SCI2 Control Register 0
0x30 5022(A) 0x30 5422(B)	SCC2R1	SCI2 Control Register 1
0x30 5024(A) 0x30 5424(B)	SC2SR	SCI2 Status Register
0x30 5026(A) 0x30 5426(B)	SC2DR	SCI2 Data Register Transmit Data Register (TDR2)* Receive Data Register (RDR2)*
0x30 5028(A) 0x30 5428(B)	QSCI1CR	QSCI1 Control Register Interrupts, wrap, queue size and enables for receive and transmit, QTPNT. See **Table 14-35** for bit descriptions.
0x30 502A(A) 0x30 542A(B)	QSCI1SR	QSCI1 Status Register OverRun error flag, queue status flags, QRPNT, and QPEND. See **Table 14-36** for bit descriptions.
0x30 502C — 0x30 504A(A) 0x30 542C — 0x30 544A(B)	QSCI1 Transmit Queue Memory Area	QSCI1 Transmit Queue Data locations (on half-word boundary)
0x30 504C-6A(A) 0x30 544C-6A(B)	QSCI1 Receive Queue Memory Area	QSCI1 Receive Queue Data locations (on half-word boundary)

*Reads access the RDRx; writes access the TDRx.

During SCIx initialization, two bits in the SCCxR1 should be written last: the transmitter enable (TE) and receiver enable (RE) bits, which enable SCIx. Registers SCCxR0 and SCCxR1 should both be initialized at the same time or before TE and RE are asserted.

A single half-word write to SCCxR1 can be used to initialize SCIx and enable the transmitter and receiver.

14.8.2 SCI Control Register 0

SCCxR0 contains the SCIx baud rate selection field and two bits controlling the clock source. The baud rate must be set before the SCI is enabled. The CPU can read and write SCCxR0 at any time.

Changing the value of SCCxR0 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR0 — SCI Control Register 0 **0x30 5008**
0x30 5408

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
OTHR	LNKBD	0	SCxBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 14-27 SCCxR0 Bit Descriptions

Bit(s)	Name	Description
0	OTHR	This bit is reserved and should always be programmed to 0.
1	LNKBD	This bit is reserved and should always be programmed to 0.
2	—	Reserved
3:15	SCxBR	SCI baud rate. The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCxBR disables the baud rate generator. Baud clock rate is calculated as follows: $\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$ where SCxBR is in the range of 1 to 8191. Refer to **14.8.7.3 Baud Clock** for more information.

14.8.3 SCI Control Register 1

SCCxR1 contains SCIx configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read or write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCCxR1 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR1 — SCI Control Register 1

0x30 500A, 0x30 5022
0x30 540A, 0x30 5422

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-28 SCCxR1 Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	LOOPS	Loop mode 0 = Normal SCI operation, no looping, feedback path disabled. 1 = SCI test operation, looping, feedback path enabled.
2	WOMS	Wired-OR mode for SCI Pins 0 = If configured as an output, TXD is a normal CMOS output. 1 = If configured as an output, TXD is an open drain output.
3	ILT	Idle-line detect type. Refer to **14.8.7.8 Idle-Line Detection**. 0 = Short idle-line detect (start count on first one). 1 = Long idle-line detect (start count on first one after stop bit(s)).
4	PT	Parity type. Refer to **14.8.7.4 Parity Checking**. 0 = Even parity. 1 = Odd parity.
5	PE	Parity enable. Refer to **14.8.7.4 Parity Checking**. 0 = SCI parity disabled. 1 = SCI parity enabled.
6	M	Mode select. Refer to **14.8.7.2 Serial Formats**. 0 = 10-bit SCI frame. 1 = 11-bit SCI frame.
7	WAKE	Wakeup by address mark. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = SCI receiver awakened by idle-line detection. 1 = SCI receiver awakened by address mark (last bit set).
8	TIE	Transmit interrupt enable 0 = SCI TDRE interrupts disabled. 1 = SCI TDRE interrupts enabled.
9	TCIE	Transmit complete interrupt enable 0 = SCI TC interrupts disabled. 1 = SCI TC interrupts enabled.
10	RIE	Receiver interrupt enable 0 = SCI RDRF and OR interrupts disabled. 1 = SCI RDRF and OR interrupts enabled.
11	ILIE	Idle-line interrupt enable 0 = SCI IDLE interrupts disabled. 1 = SCI IDLE interrupts enabled.
12	TE	Transmitter enable 0 = SCI transmitter disabled (TXD pin can be used as general-purpose output) 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).
13	RE	Receiver Enable 0 = SCI receiver disabled (RXD pin can be used as general-purpose input). 1 = SCI receiver enabled (RXD pin is dedicated to SCI receiver).

Table 14-28 SCCxR1 Bit Descriptions (Continued)

Bit(s)	Name	Description
14	RWU	Receiver wakeup. Refer to **14.8.7.9 Receiver Wake-Up**. 0 = Normal receiver operation (received data recognized). 1 = Wakeup mode enabled (received data ignored until receiver is awakened).
15	SBK	Send break 0 = Normal operation. 1 = Break frame(s) transmitted after completion of current frame.

14.8.4 SCI Status Register (SCxSR)

SCxSR contains flags that show SCI operating conditions. These flags are cleared either by SCIx hardware or by a read/write sequence. The sequence consists of reading the SCxSR (either the upper byte, lower byte, or the entire half-word) with a flag bit set, then reading (or writing, in the case of flags TDRE and TC) the SCxDR (either the lower byte or the half-word).

The contents of the two 16-bit registers SCxSR and SCxDR appear as upper and lower half-words, respectively, when the SCxSR is read into a 32-bit register. An upper byte access of SCxSR is meaningful only for reads. Note that a word read can simultaneously access both registers SCxSR and SCxDR. This action clears the receive status flag bits that were set at the time of the read, but does not clear the TDRE or TC flags. To clear TC, the SCxSR read must be followed by a write to register SCxDR (either the lower byte or the half-word). The TDRE flag in the status register is read-only.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits but before the CPU has read or written the SCxDR, the newly set status bit is not cleared. Instead, SCxSR must be read again with the bit set and SCxDR must be read or written before the status bit is cleared.

NOTE

None of the status bits are cleared by reading a status bit while it is set and then writing zero to that same bit. Instead, the procedure outlined above must be followed. Note further that reading either byte of SCxSR causes all 16 bits to be accessed, and any status bits already set in either byte are armed to clear on a subsequent read or write of SCxDR.

SCxSR — SCIx Status Register **0x30 500C, 0x30 5024**
0x30 540C, 0x30 5424

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED							TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:															
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 14-29 SCxSR Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	TDRE	Transmit data register empty. TDRE is set when the byte in TDRx is transferred to the transmit serial shifter. If this bit is zero, the transfer is yet to occur and a write to TDRx will overwrite the previous value. New data is not transmitted if TDRx is written without first clearing TDRE. 0 = Transmit data register still contains data to be sent to the transmit serial shifter. 1 = A new character can now be written to the transmit data register. For transmit queue operation, this bit should be ignored by software.
8	TC	Transmit complete. TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle-line), or queued breaks (logic zero). 0 = SCI transmitter is busy. 1 = SCI transmitter is idle. For transmit queue operation, TC is cleared when SCxSR is read with TC set, followed by a write to SCTQ[0:15].
9	RDRF	Receive data register full. RDRF is set when the contents of the receive serial shifter are transferred to register RDRx. If one or more errors are detected in the received word, the appropriate flag(s) (NF, FE, or PF) are set within the same clock cycle. 0 = Receive data register is empty or contains previously read data. 1 = Receive data register contains new data. For receiver queue operation, this bit should be ignored by software.
10	RAF	Receiver active flag. RAF indicates whether the receiver is busy. This flag is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters. 0 = SCI receiver is idle. 1 = SCI receiver is busy.
11	IDLE	Idle line detected. IDLE is set when the receiver detects an idle-line condition (reception of a minimum of 10 or 11 consecutive ones as specified by ILT in SCCxR1). This bit is not set by the idle-line condition when RWU in SCCxR1 is set. Once cleared, IDLE is not set again until after RDRF is set (after the line is active and becomes idle again). If a break is received, RDRF is set, allowing a subsequent idle line to be detected again. Under certain conditions, the IDLE flag may be set immediately following the negation of RE in SCCxR1. System designs should ensure this causes no detrimental effects. 0 = SCI receiver did not detect an idle-line condition. 1 = SCI receiver detected an idle-line condition. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15].
12	OR	Overrun error. OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDRx, and RDRx is already full (RDRF is still set). Data transfer is inhibited until OR is cleared. Previous data in RDRx remains valid, but additional data received during an overrun condition (including the byte that set OR) is lost. Note that whereas the other receiver status flags (NF, FE, and PF) reflect the status of data already transferred to RDRx, the OR flag reflects an operational condition that resulted in a loss of data to RDRx. 0 = RDRF is cleared before new data arrives. 1 = RDRF is not cleared before new data arrives.

Table 14-29 SCxSR Bit Descriptions (Continued)

Bit(s)	Name	Description
13	NF	Noise error flag. NF is set when the receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit(s). It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times for noise. If the three samples are not at the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with NF, an interrupt can be generated with RDRF, and the interrupt handler can check NF. 0 = No noise detected in the received data. 1 = Noise detected in the received data. For receiver queue operation NF is cleared when SCxSR is read with NF set, followed by a read of SCRQ[0:15].
14	FE	Framing error. FE is set when the receiver detects a zero where a stop bit (one) was expected. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. FE is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with FE, an interrupt can be generated with RDRF, and the interrupt handler can check FE. 0 = No framing error detected in the received data. 1 = Framing error or break detected in the received data.
15	PF	Parity error. PF is set when the receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with PF, an interrupt can be generated with RDRF, and the interrupt handler can check PF. 0 = No parity error detected in the received data. 1 = Parity error detected in the received data.

14.8.5 SCI Data Register (SCxDR)

The SCxDR consists of two data registers located at the same address. The receive data register (RDRx) is a read-only register that contains data received by the SCI serial interface. Data is shifted into the receive serial shifter and is transferred to RDRx. The transmit data register (TDRx) is a write-only register that contains data to be transmitted. Data is first written to TDRx, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

SCxDR — SCI Data Register **0x30 500E**
0x30 540E

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED							R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

Table 14-30 SCxDR Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7:15	R[8:0]/ T[8:0]	R[7:0]/T[7:0] contain either the eight data bits received when SCxDR is read, or the eight data bits to be transmitted when SCxDR is written. R8/T8 are used when the SCI is configured for nine-bit operation (M = 1). When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect. Accesses to the lower byte of SCxDR triggers the mechanism for clearing the status bits or for initiating transmissions whether byte, half-word, or word accesses are used.

14.8.6 SCI Pins

The RXD1 and RXD2 pins are the receive data pins for the SCI1 and SCI2, respectively. TXD1 and TXD2 are the transmit data pins for the two SCI modules. An external clock pin, ECK, is common to both SCIs. The pins and their functions are listed in **Table 14-31**.

Table 14-31 SCI Pin Functions

Pin Names	Mnemonic	Mode	Function
Receive Data	RXD1, RXD2	Receiver disabled Receiver enabled	General purpose input Serial data input to SCI
Transmit Data	TXD1, TXD2	Transmitter disabled Transmitter enabled	General purpose output Serial data output from SCI
External Clock	ECK	Receiver disabled Receiver enabled Transmitter disabled Transmitter enabled	Not used Alternate input source to baud Not used Alternate input source to baud

14.8.7 SCI Operation

The SCI can operate in polled or interrupt-driven mode. Status flags in SCxSR reflect SCI conditions regardless of the operating mode chosen. The TIE, TCIE, RIE, and ILIE bits in SCCxR1 enable interrupts for the conditions indicated by the TDRE, TC, RDRF, and IDLE bits in SCxSR, respectively.

14.8.7.1 Definition of Terms

- Bit-time — The time required to transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- Start bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time samples of logic one.
- Stop bit— One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle frame — A frame that consists of consecutive ones. An idle frame has no start bit.

- Break frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

14.8.7.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both 10-bit and 11-bit frames. The M bit in SCCxR1 specifies the number of bits per frame.

The most common data frame format for NRZ (non-return to zero) serial interfaces is one start bit, eight data bits (LSB first), and one stop bit (ten bits total). The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and 11-bit frames are shown in **Table 14-32**.

Table 14-32 Serial Frame Formats

10-bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

14.8.7.3 Baud Clock

The SCI baud rate is programmed by writing a 13-bit value to the SCxBR field in SCI control register zero (SCCxR0). The baud rate is derived from the MCU IMB3 clock by a modulus counter. Writing a value of zero to SCxBR[12:0] disables the baud rate generator. The baud rate is calculated as follows:

$$\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$$

or

$$\text{SCxBR} = \frac{f_{SYS}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCxBR is in the range {1, 2, 3, ..., 8191}.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receive time sampling clock with a frequency 16 times that of the SCI baud rate. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period.

Table 14-33 shows possible baud rates for a 40-MHz IMB3 clock. The maximum baud rate with this IMB3 clock speed is 1250 Kbaud.

Table 14-33 Examples of SCIx Baud Rates[1]

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCxBR
1,250,000.00	1,250,000.00	0.00	1
57,600.00	56,818.18	-1.36	22
38,400.00	37,878.79	-1.36	33
32,768.00	32,894.74	0.39	38
28,800.00	29,069.77	0.94	43
19,200.00	19,230.77	0.16	65
14,400.00	14,367.81	-0.22	87
9,600.00	9,615.38	0.16	130
4,800.00	4,807.69	0.16	260
2,400.00	2,399.23	-0.03	521
1,200.00	1,199.62	-0.03	1042
600.00	600.09	0.02	2083
300.00	299.98	-0.01	4167

NOTES:
1. These rates are based on a 40-MHz IMB3 clock.

14.8.7.4 Parity Checking

The PT bit in SCCxR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCxR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame (i.e., the bit preceding the stop bit) is used for the parity function. For transmitted data, a parity bit is generated. For received data, the parity bit is checked. When parity checking is enabled, the PF bit in the SCI status register (SCxSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 14-34** shows possible data and parity formats.

Table 14-34 Effect of Parity Checking on Data Size

M	PE	Result
0	0	8 data bits
0	1	7 data bits, 1 parity bit
1	0	9 data bits
1	1	8 data bits, 1 parity bit

14.8.7.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDRx while other data is shifted out. The TE bit in SCCxR1 enables (TE = 1) and disables (TE = 0) the transmitter.

The shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCxR1 determines whether TXD is an open drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function, regardless of whether the pin is used by the SCI or as a general-purpose output pin.

Data to be transmitted is written to SCxDR, then transferred to the serial shifter. Before writing to TDRx, the transmit data register empty (TDRE) flag in SCxSR should be checked. When TDRE = 0, the TDRx contains data that has not been transferred to the shifter. Writing to SCxDR again overwrites the data. If TDRE = 1, then TDRx is empty, and new data may be written to TDRx, clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new data frame is in TDRx (TDRE = 0), then the new data is transferred from TDRx to the transmit serial shifter and TDRE is set automatically. An interrupt may optionally be generated at this point.

The transmission complete (TC) flag in SCxSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCxSR while TC is set, then writing new data to SCxDR, or writing to SCTQ[0:15] for transmit queue operation.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The SBK bit in SCCxR1 is used to insert break frames in a transmission. A non-zero integer number of break frames are transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To ensure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

When TE is cleared, the transmitter is disabled after all pending idle, data, and break frames are transmitted. The TC flag is set, and control of the TXD pin reverts to PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output, then write a one to either QDTX1 or QDTX2 of the PORTQS register. See **14.6.1 Port QS Data Register**

(PORTQS). When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place non-listening receivers in wake-up mode between transmissions, or to signal a re-transmission by forcing an idle-line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter marks idle. Otherwise, normal transmission of the next sequence begins.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCxR1. Service routines can load the last data frame in a sequence into SCxDR, then terminate the transmission when a TDRE interrupt occurs.

Two SCI messages can be separated with minimum idle time by using a preamble of 10 bit-times (11 if a 9-bit data format is specified) of marks (logic ones). Follow these steps:

1. Write the last data frame of the first message to the TDRx
2. Wait for TDRE to go high, indicating that the last data frame is transferred to the transmit serial shifter
3. Clear TE and then set TE back to one. This queues the preamble to follow the stop bit of the current transmission immediately.
4. Write the first data frame of the second message to register TDRx

In this sequence, if the first data frame of the second message is not transferred to TDRx prior to the finish of the preamble transmission, then the transmit data line (TXDx pin) marks idle (logic one) until TDRx is written. In addition, if the last data frame of the first message finishes shifting out (including the stop bit) and TE is clear, TC goes high and transmission is considered complete. The TXDx pin reverts to being a general-purpose output pin.

14.8.7.6 Receiver Operation

The receiver can be divided into two segments. The first is the receiver bit processor logic that synchronizes to the asynchronous receive data and evaluates the logic sense of each bit in the serial stream. The second receiver segment controls the functional operation and the interface to the CPU including the conversion of the serial data stream to parallel access by the CPU.

Receiver Bit Processor — The receiver bit processor contains logic to synchronize the bit-time of the incoming data and to evaluate the logic sense of each bit. To accomplish this an RT clock, which is 16 times the baud rate, is used to sample each bit. Each bit-time can thus be divided into 16 time periods called RT1–RT16. The receiver looks for a possible start bit by watching for a high-to-low transition on the RXDx pin and by assigning the RT time labels appropriately.

When the receiver is enabled by writing RE in SCCxR1 to one, the receiver bit processor logic begins an asynchronous search for a start bit. The goal of this search is

to gain synchronization with a frame. The bit-time synchronization is done at the beginning of each frame so that small differences in the baud rate of the receiver and transmitter are not cumulative. SCIx also synchronizes on all one-to-zero transitions in the serial data stream, which makes SCIx tolerant to small frequency variations in the received data stream.

The sequence of events used by the receiver to find a start bit is listed below.

1. Sample RXDx input during each RT period and maintain these samples in a serial pipeline that is three RT periods deep.
2. If RXDx is low during this RT period, go to step 1.
3. If RXDx is high during this RT period, store sample and proceed to step 4.
4. If RXDx is low during this RT period, but not high for the previous three RT periods (which is noise only), set an internal working noise flag and go to step 1, since this transition was not a valid start bit transition.
5. If RXDx is low during this RT period and has been high for the previous three RT periods, call this period RT1, set RAF, and proceed to step 6.
6. Skip RT2 but place RT3 in the pipeline and proceed to step 7.
7. Skip RT4 and sample RT5. If both RT3 and RT5 are high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT5 in the pipeline and proceed to step 8.
8. Skip RT6 and sample RT7. If any two of RT3, RT5, or RT7 is high (RT1 was noise only), set an internal working noise flag. Go to step 3 and clear RAF. Otherwise, place RT7 in the pipeline and proceed to step 9.
9. A valid start bit is found and synchronization is achieved. From this point on until the end of the frame, the RT clock will increment starting over again with RT1 on each one-to-zero transition or each RT16. The beginning of a bit-time is thus defined as RT1 and the end of a bit-time as RT16.

Upon detection of a valid start bit, synchronization is established and is maintained through the reception of the last stop bit, after which the procedure starts all over again to search for a new valid start bit. During a frame's reception, SCIx resynchronizes the RT clock on any one-to-zero transitions.

Additional logic in the receiver bit processor determines the logic level of the received bit and implements an advanced noise-detection function. During each bit-time of a frame (including the start and stop bits), three logic-sense samples are taken at RT8, RT9, and RT10. The logic sense of the bit-time is decided by a majority vote of these three samples. This logic level is shifted into register RDRx for every bit except the start and stop bits.

If RT8, RT9, and RT10 do not all agree, an internal working noise flag is set. Additionally for the start bit, if RT3, RT5, and RT7 do not all agree, the internal working noise flag is set. If this flag is set for any of the bit-times in a frame, the NF flag in SCxSR is set concurrently with the RDRF flag in SCxSR when the data is transferred to register RDRx. The user must determine if the data received with NF set is valid. Noise on the RXDx pin does not necessarily corrupt all data.

The operation of the receiver bit processor is shown in **Figure 14-11**. This example demonstrates the search for a valid start bit and the synchronization procedure as outlined above. The possibilities of noise durations greater than one bit-time are not considered in this examples.

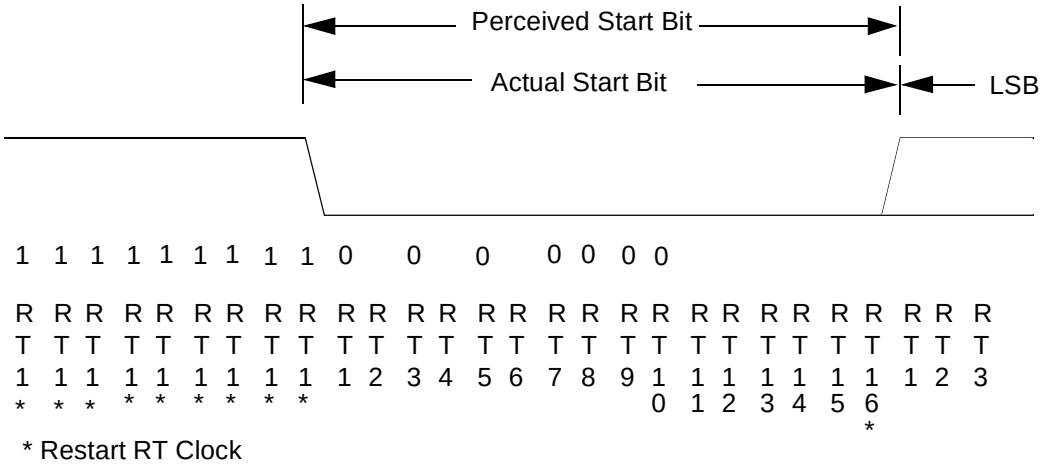


Figure 14-11 Start Search Example

14.8.7.7 Receiver Functional Operation

The RE bit in SCCxR1 enables (RE = 1) and disables (RE = 0) the receiver. The receiver contains a receive serial shifter and a parallel receive data register (RDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in the RDRx while other data is shifted in.

Receiver bit processor logic drives a state machine that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. A receive time clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the receive time clock with the incoming data stream. From this point on, data movement is synchronized with the MCU IMB3 clock. Operation of the receiver state machine is detailed in the ***Queued Serial Module Reference Manual*** **(QSMRM/AD)**.

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDRx. The receiver data register flag (RDRF) is set when the data is transferred.

The stop bit is always a logic one. If a logic zero is sensed during this bit-time, the FE flag in SCxSR is set. A framing error is usually caused by mismatched baud rates between the receiver and transmitter or by a significant burst of noise. Note that a

framing error is not always detected; the data in the expected stop bit-time may happen to be a logic one.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise flag (NF), the parity flag (PF), and the framing error (FE) flag in SCxSR are not set until data is transferred from the serial shifter to the RDRx.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in SCxSR is set. OR indicates that the RDRx needs to be serviced faster. When OR is set, the data in the RDRx is preserved, but the data in the serial shifter is lost.

When a completed frame is received into the RDRx, either the RDRF or OR flag is always set. If RIE in SCCxR1 is set, an interrupt results whenever RDRF is set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF, as appropriate. These receiver flags are never set with OR because the flags apply only to the data in the receive serial shifter. The receiver status flags do not have separate interrupt enables, since they are set simultaneously with RDRF and must be read at the same time as RDRF.

When the CPU reads SCxSR and SCxDR in sequence, it acquires status and data, and also clears the status flags. Reading SCxSR acquires status and arms the clearing mechanism. Reading SCxDR acquires data and clears SCxSR.

14.8.7.8 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronically and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCxR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle-line type (ILT) bit in SCCxR1 determines which type of detection is used. When an idle-line condition is detected, the IDLE flag in SCxSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle-line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, software overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the ILIE bit in SCCxR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCxSR and SCxDR in sequence. For

receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15]. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

14.8.7.9 Receiver Wake-Up

The receiver wake-up function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCxR1. While RWU is set, receiver status flags and interrupts are disabled. Although the software can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCxR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up allows a receiver to sleep until an idle line is detected. When an idle line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address mark wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

14.8.7.10 Internal Loop Mode

The LOOPS bit in SCCxR1 controls a feedback path in the data serial shifter. When LOOPS is set, the SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

14.9 SCI Queue Operation

14.9.1 Queue Operation of SCI1 for Transmit and Receive

The SCI1 serial module allows for queueing on transmit and receive data frames. In the standard mode, in which the queue is disabled, the SCI1 operates as previously defined (i.e., transmit and receive operations done via SC1DR). However, if the SCI1 queue feature is enabled (by setting the QTE and/or QRE bits within QSCI1CR) a set of 16 entry queues is allocated for the receive and/or transmit operation. Through software control the queue is capable of continuous receive and transfer operations within the SCI1 serial unit.

14.9.2 Queued SCI1 Status and Control Registers

The SCI1 queue uses the following registers:

- QSCI1 control register (QSCI1CR, address offset 0x28)
- QSCI1 status register (QSCI1SR, address offset 0x2A)

14.9.2.1 QSCI1 Control Register

QSCI1CR — QSCI1 Control Register **0x30 5028**
0x30 5428

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QTPNT				QTHFI	QBHFI	QTHEI	QBHEI	0	QTE	QRE	QTWE	QTSZ			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 14-35 QSCI1CR Bit Descriptions

Bit(s)	Name	Description
0:3	QTPNT	Queue transmit pointer. QTPNT is a 4-bit counter used to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This feature allows for ease of testability. This field is writable in test mode only; otherwise it is read-only.
4	QTHFI	Receiver queue top-half full interrupt. When set, QTHFI enables an SCI1 interrupt whenever the QTHF flag in QSCI1SR is set. The interrupt is blocked by negating QTHFI. This bit refers to the queue locations SCRQ[0:7]. 0 = QTHF interrupt inhibited 1 = Queue top-half full (QTHF) interrupt enabled
5	QBHFI	Receiver queue bottom-half full interrupt. When set, QBHFI enables an SCI1 interrupt whenever the QBHF flag in QSCI1SR is set. The interrupt is blocked by negating QBHFI. This bit refers to the queue locations SCRQ[8:15]. 0 = QBHF interrupt inhibited 1 = Queue bottom-half full (QBHF) interrupt enabled
6	QTHEI	Transmitter queue top-half empty interrupt. When set, QTHEI enables an SCI1 interrupt whenever the QTHE flag in QSCI1SR is set. The interrupt is blocked by negating QTHEI. This bit refers to the queue locations SCTQ[0:7]. 0 = QTHE interrupt inhibited 1 = Queue top-half empty (QTHE) interrupt enabled
7	QBHEI	Transmitter queue bottom-half empty interrupt. When set, QBHEI enables an SCI1 interrupt whenever the QBHE flag in QSCI1SR is set. The interrupt is blocked by negating QBHEI. This bit refers to the queue locations SCTQ[8:15]. 0 = QBHE interrupt inhibited 1 = Queue bottom-half empty (QBHE) interrupt enabled
8	—	Reserved
9	QTE	Queue transmit enable. When set, the transmit queue is enabled and the TDRE bit should be ignored by software. The TC bit is redefined to indicate when the entire queue is finished transmitting. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QTE. 0 = Transmit queue is disabled 1 = Transmit queue is enabled
10	QRE	Queue receive enable. When set, the receive queue is enabled and the RDRF bit should be ignored by software. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QRE. 0 = Receive queue is disabled 1 = Receive queue is enabled
11	QTWE	Queue transmit wrap enable. When set, the transmit queue is allowed to restart transmitting from the top of the queue after reaching the bottom of the queue. After each wrap of the queue, QTWE is cleared by hardware. 0 = Transmit queue wrap feature is disabled 1 = Transmit queue wrap feature is enabled
12:15	QTSZ	Queue transfer size. The QTSZ bits allow programming the number of data frames to be transmitted. From 1 (QTSZ = 0b0000) to 16 (QTSZ = 0b1111) data frames can be specified. QTSZ is loaded into QPEND initially or when a wrap occurs.

14.9.2.2 QSCI1 Status Register

QSCI1SR — QSCI1 Status Register **0x30 502A**
0x30 542A

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED			QOR	QTHF	QBHF	QTHE	QBHE	QRPNT				QPEND			
RESET:															
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Table 14-36 QSCI1SR Bit Descriptions

Bit(s)	Name	Description
0:2	—	Reserved
3	QOR	Receiver queue overrun error. The QOR is set when a new data frame is ready to be transferred from the SC1DR to the queue and the queue is already full (QTHF or QBHF are still set). Data transfer is inhibited until QOR is cleared. Previous data transferred to the queue remains valid. Additional data received during a queue overrun condition is not lost provided the receive queue is re-enabled before OR (SC1SR) is set. The OR flag is set when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost. 0 = The queue is empty before valid data is in the SC1DR 1 = The queue is not empty when valid data is in the SC1DR
4	QTHF	Receiver queue top-half full. QTHF is set when the receive queue locations SCRQ[0:7] are completely filled with new data received via the serial shifter. QTHF is cleared when register QSCI1SR is read with QTHF set, followed by a write of QTHF to zero. 0 = The queue locations SCRQ[0:7] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[0:7] are completely full of newly received data
5	QBHF	Receiver queue bottom-half full. QBHF is set when the receive queue locations SCRQ[8:15] are completely filled with new data received via the serial shifter. QBHF is cleared when register QSCI1SR is read with QBHF set, followed by a write of QBHF to zero. 0 = The queue locations SCRQ[8:15] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[8:15] are completely full of newly received data
6	QTHE	Transmitter queue top-half empty. QTHE is set when all the data frames in the transmit queue locations SCTQ[0:7] have been transferred to the transmit serial shifter. QTHE is cleared when register QSCI1SR is read with QTHE set, followed by a write of QTHE to zero. 0 = The queue locations SCTQ[0:7] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[0:7]
7	QBHE	Transmitter queue bottom-half empty. QBHE is set when all the data frames in the transmit queue locations SCTQ[8:15] has been transferred to the transmit serial shifter. QBHE is cleared when register QSCI1SR is read with QBHE set, followed by a write of QBHE to zero. 0 = The queue locations SCTQ[8:15] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[8:15]
8:11	QRPNT	Queue receive pointer. QRPNT is a 4-bit counter used to indicate the position where the next valid data frame will be stored within the receive queue. This field is writable in test mode only; otherwise it is read-only.
12:15	QPEND	Queue pending. QPEND is a 4-bit decrementer used to indicate the number of data frames in the queue that are awaiting transfer to the SC1DR. This field is writable in test mode only; otherwise it is read-only. From 1 (QPEND = 0b0000) to 16 (or done, QPEND = 1111) data frames can be specified.

14.9.3 QSCI1 Transmitter Block Diagram

The block diagram of the enhancements to the SCI transmitter is shown in **Figure 14-12**.

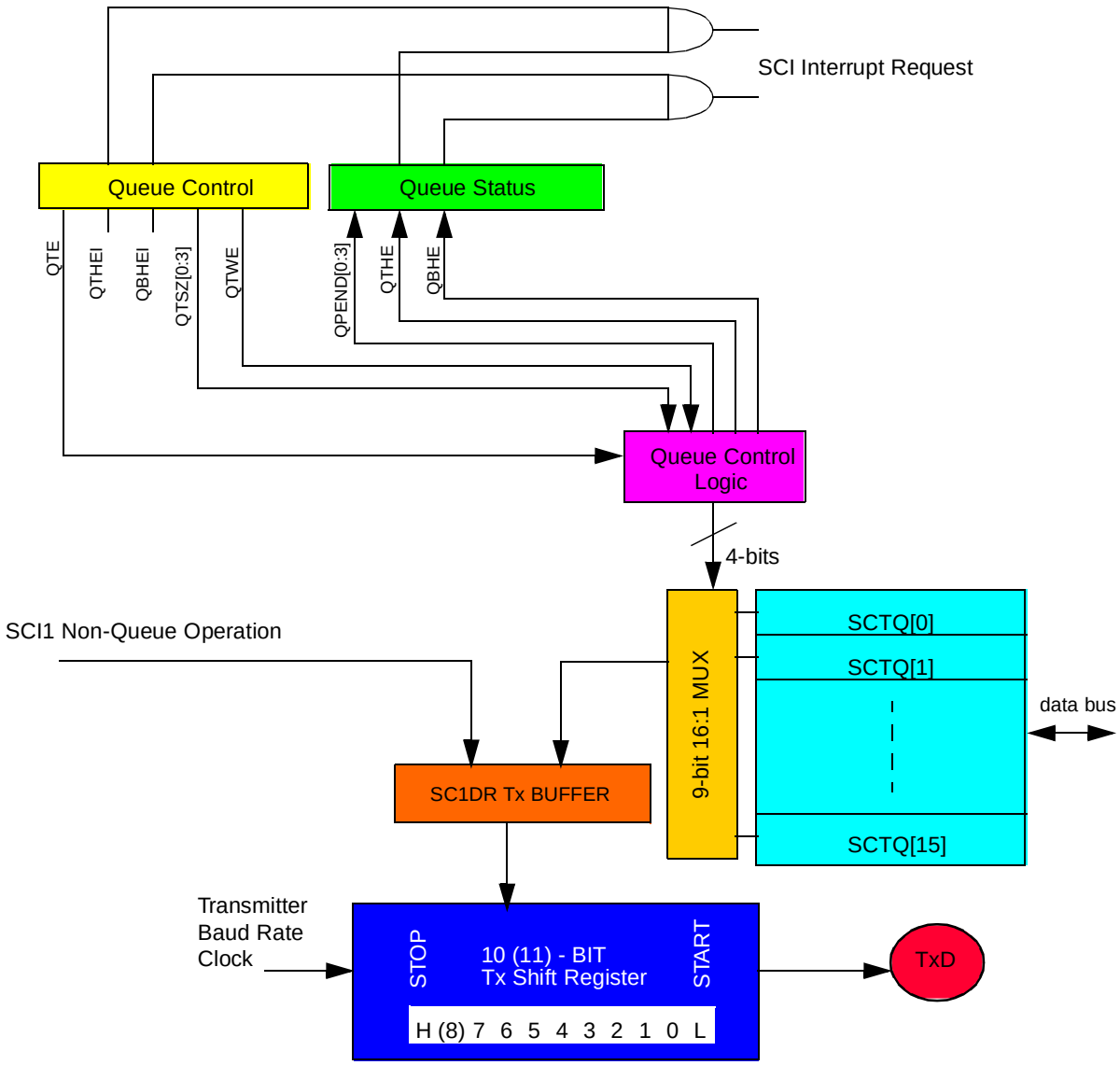


Figure 14-12 Queue Transmitter Block Enhancements

14.9.4 QSCI1 Additional Transmit Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue transmit enable (QTE) bit set by software. When enabled, (QTE = 1) the TDRE bit should be ignored by software and the TC bit is redefined (as described later).
- When the queue is disabled (QTE = 0), the SCI functions in single buffer transfer mode where the queue size is set to one (QTSZ = 0000), and TDRE and TC function as previously defined. Locations SCTQ[0:15] can be used as general purpose 9-bit registers. All other bits pertaining to the queue should be ignored by software.
- Programmable queue up to 16 transmits (SCTQ[0:15]) which may allow for infinite and continuous transmits.

- Available transmit wrap function to prevent message breaks for transmits greater than 16. This is achieved by the transmit wrap enable (QTWE) bit. When QTWE is set, the hardware is allowed to restart transmitting from the top of the queue (SCTQ[0]). After each wrap, QTWE is cleared by hardware.
  - Transmissions of more than 16 data frames must be performed in multiples of 16 (QTSZ = 0b1111) except for the last set of transmissions. For any single non-continuous transmissions of 16 or less or the last transmit set composed of 16 or fewer data frames, programming of QTSZ to the corresponding value of 16 or less where QTWE = 0 is allowed.
- Interrupt generation when the top half (SCTQ[0:7]) of the queue has been emptied (QTHE) and the bottom half (SCTQ[8:15]) of the queue has been emptied (QBHE). This may allow for uninterrupted and continuous transmits by indicating to the CPU that it can begin refilling the queue portion that is now emptied.
  - The QTHE bit is set by hardware when the top half is empty or the transmission has completed. The QTHE bit is cleared when the QSCI1SR is read with QTHE set, followed by a write of QTHE to zero.
  - The QBHE bit is set by hardware when the bottom half is empty or the transmission has completed. The QBHE bit is cleared when the QSCI1SR is read with QBHE set, followed by a write of QBHE to zero.
  - In order to implement the transmit queue, QTE must be set (QSCI1CR), TE must be set (SCC1R1), QTHE must be cleared (QSCI1SR), and TDRE must be set (SC1SR).
- Enable and disable options for the interrupts QTHE and QBHE as controlled by QTHEI and QBHEI respectfully.
- Programmable 4-bit register queue transmit size (QTSZ) for configuring the queue to any size up to 16 transfers at a time. This value may be rewritten after transmission has started to allow for the wrap feature.
- 4-bit status register to indicate the number of data transfers pending (QPEND). This register counts down to all 0's where the next count rolls over to all 1's. This counter is writable in test mode; otherwise it is read-only.
- 4-bit counter (QTPNT) is used as a pointer to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This counter is writable in test mode; otherwise it is read-only.
- A transmit complete (TC) bit re-defined when the queue is enabled (QTE = 1) to indicate when the entire queue (including when wrapped) is finished transmitting. This is indicated when QPEND = 1111 and the shifter has completed shifting data out. TC is cleared when the SCxSR is read with TC = 1 followed by a write to SCTQ[0:15]. If the queue is disabled (QTE = 0), the TC bit operates as originally designed.
- When the transmit queue is enabled (QTE = 1), writes to the transmit data register (SC1DR) have no effect.

14.9.5 QSCI1 Transmit Flow Chart Implementing the Queue

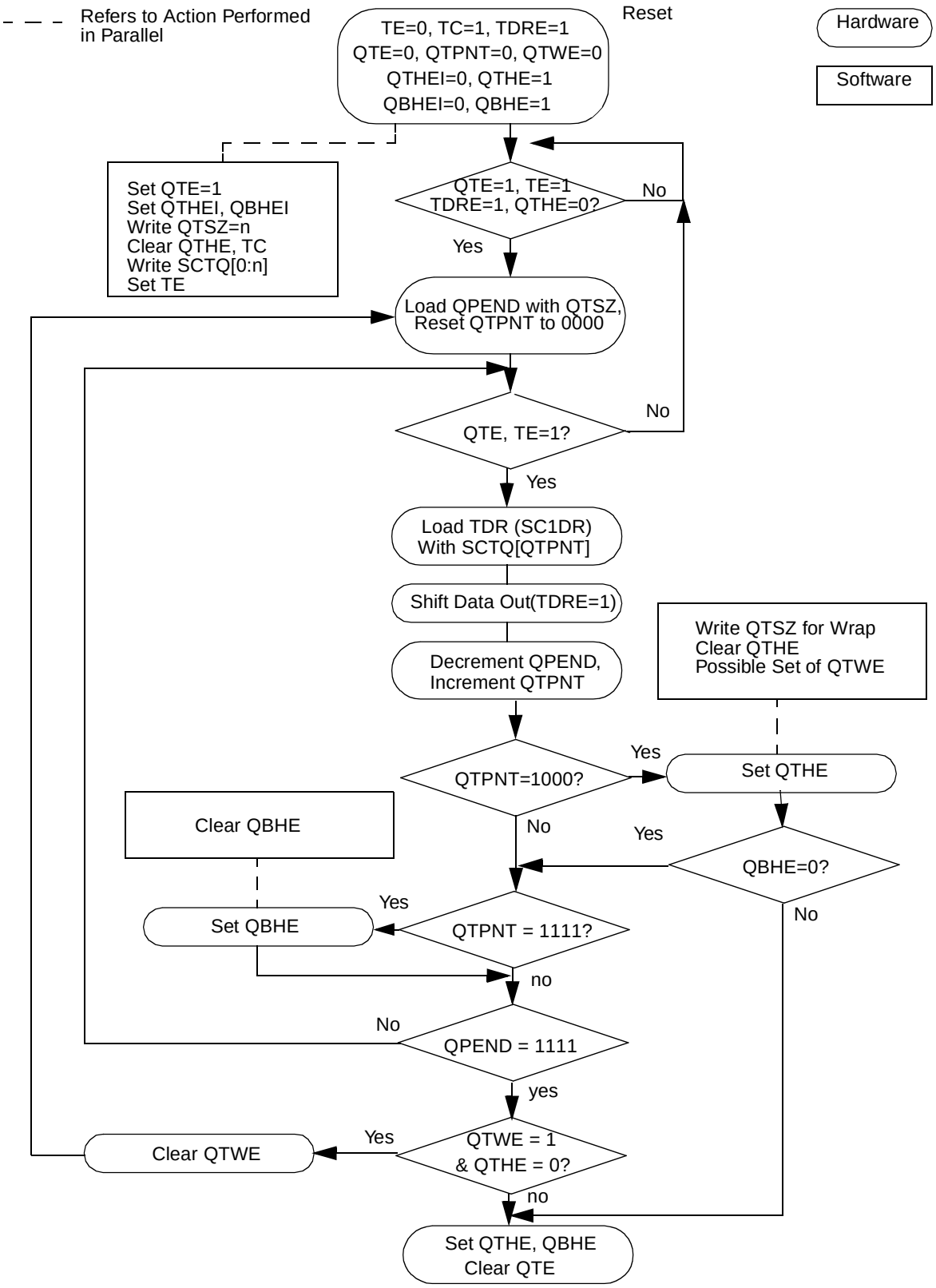


Figure 14-13 Queue Transmit Flow

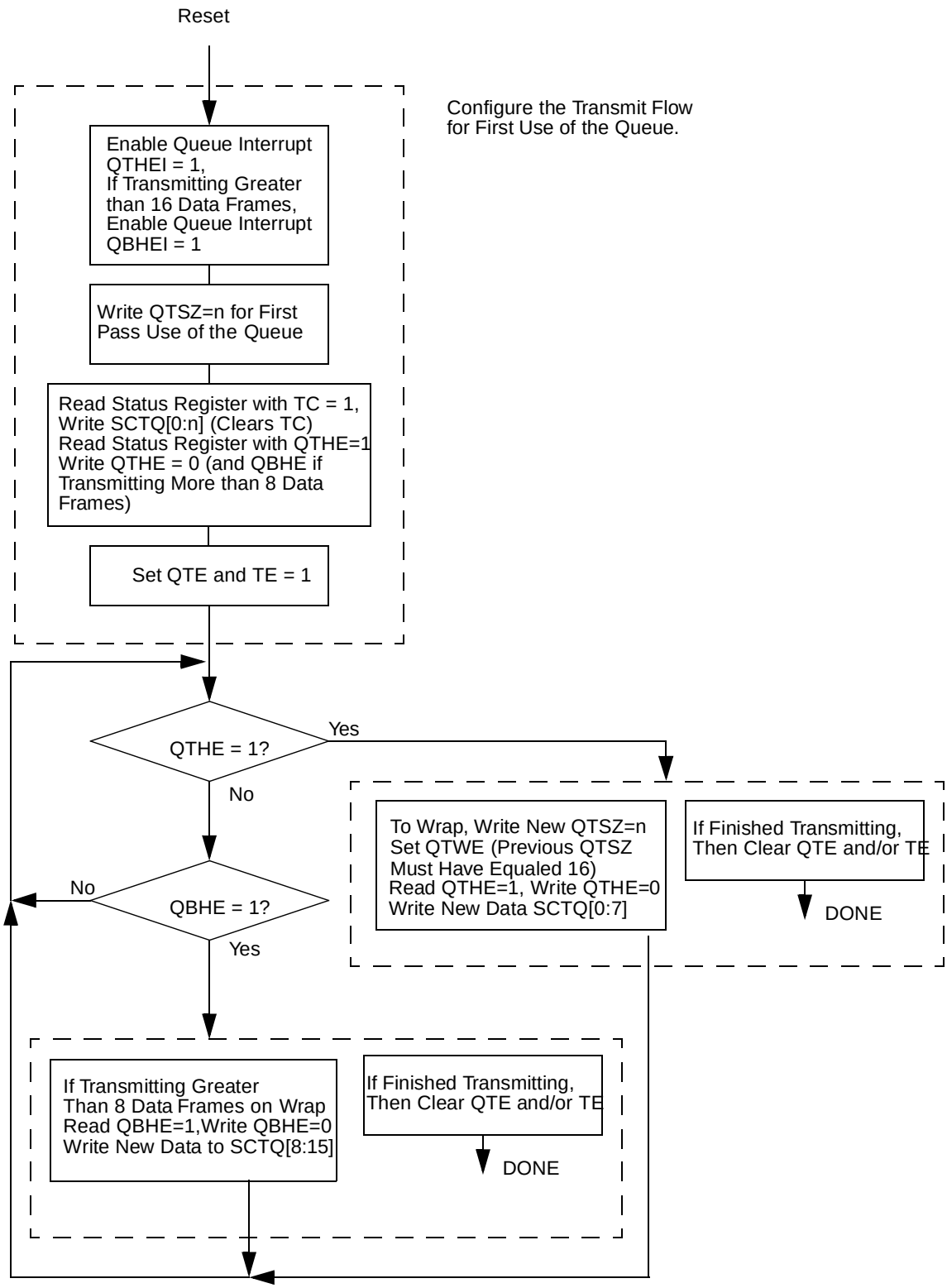


Figure 14-14 Queue Transmit Software Flow

14.9.6 Example QSCI1 Transmit for 17 Data Bytes

Figure 14-15 below shows a transmission of 17 data frames. The bold type indicates the current value for QTPNT and QPEND. The italic type indicates the action just performed by hardware. Regular type indicates the actions that should be performed by software before the next event.

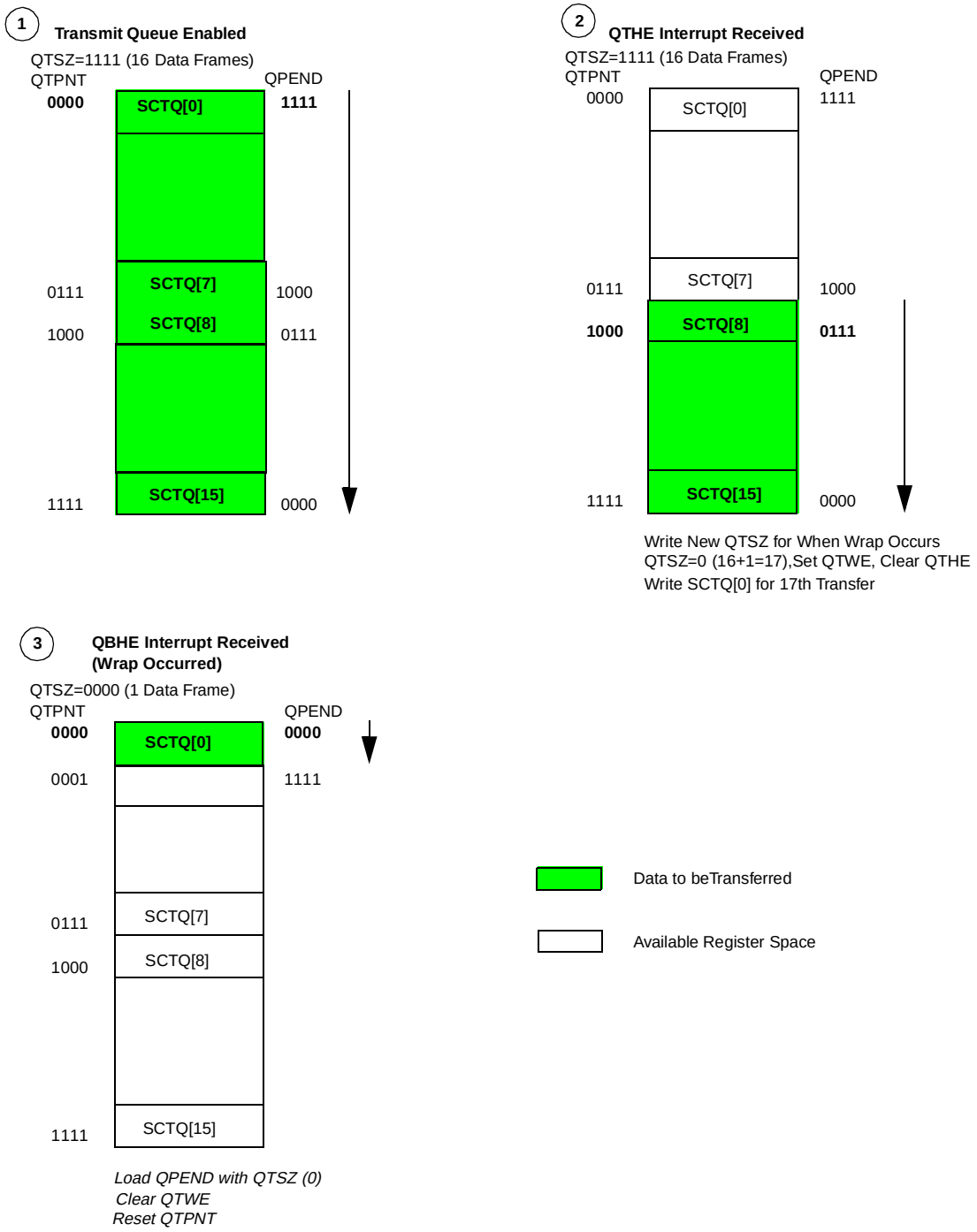


Figure 14-15 Queue Transmit Example for 17 Data Bytes

14.9.7 Example SCI Transmit for 25 Data Bytes

Figure 14-16 below is an example of a transmission of 25 data frames.

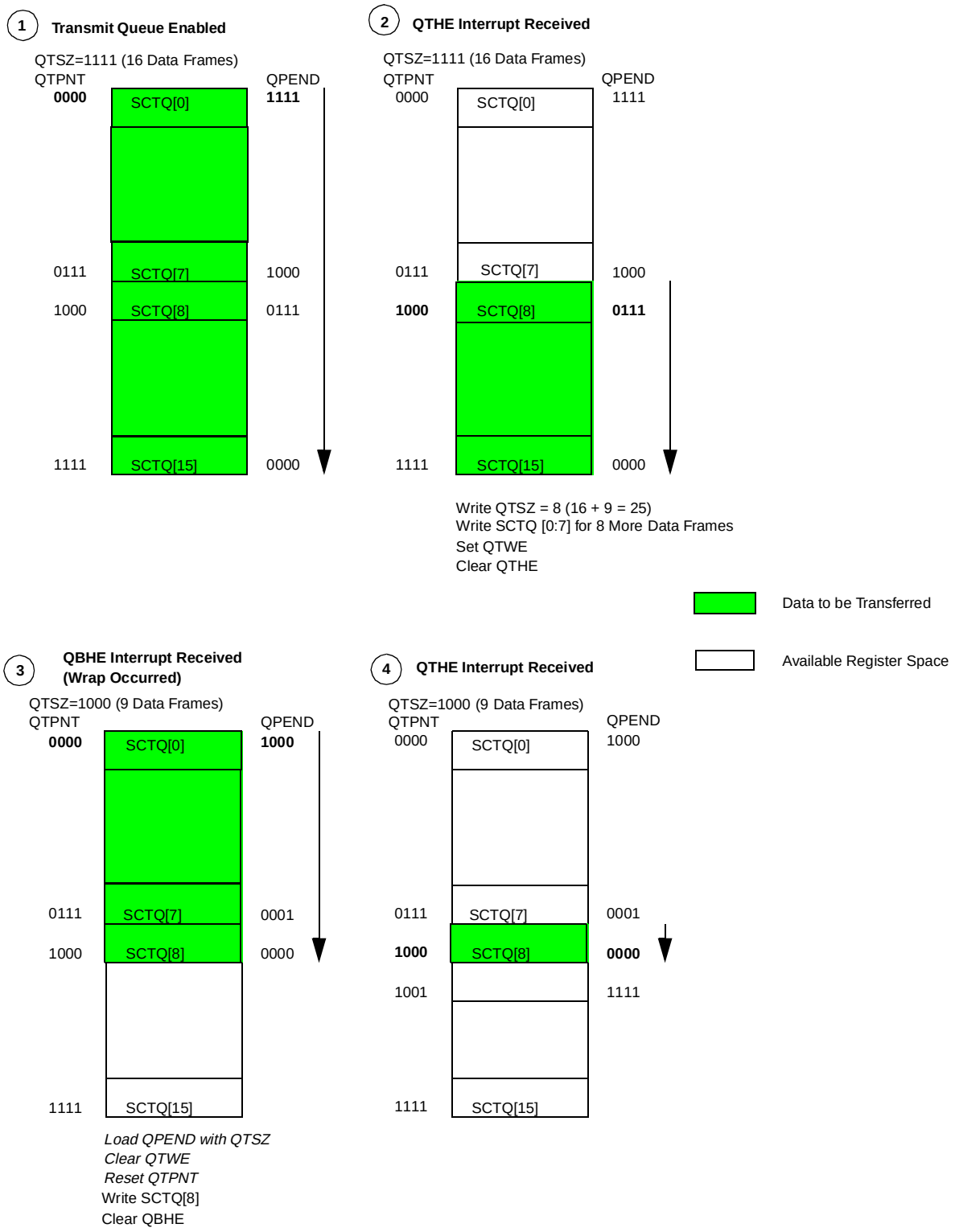


Figure 14-16 Queue Transmit Example for 25 Data Frames

14.9.8 QSCI1 Receiver Block Diagram

The block diagram of the enhancements to the SCI receiver is shown below in **Figure 14-17**.

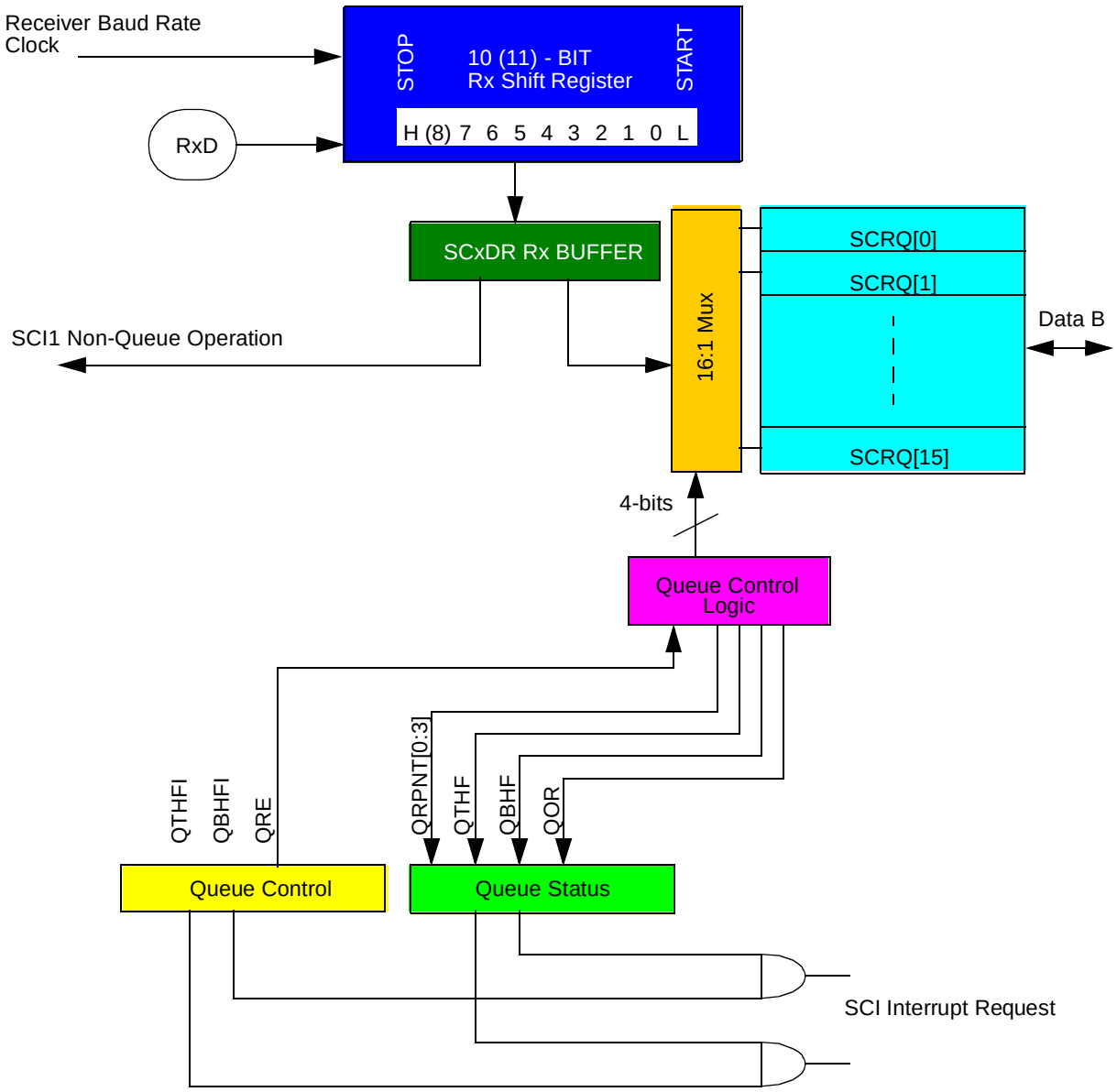


Figure 14-17 Queue Receiver Block Enhancements

14.9.9 QSCI1 Additional Receive Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue receiver enable (QRE) bit set by software. When the queue is enabled, software should ignore the RDRF bit.
- When the queue is disabled (QRE = 0), the SCI functions in single buffer receive mode (as originally designed) and RDRF and OR function as previously defined.

Locations SCRQ[0:15] can be used as general purpose 9-bit registers. Software should ignore all other bits pertaining to the queue.

- Only data that has no errors (FE and PF both false) is allowed into the queue. The status flags FE and PF, if set, reflect the status of data not allowed into the queue. The receive queue is disabled until the error flags are cleared via the original SCI mechanism and the queue is re-initialized. The pointer QRPNT indicates the queue location where the data frame would have been stored.
- Queue size capable to receive up to 16 data frames (SCRQ[0:15]) which may allow for infinite and continuous receives.
- Interrupt generation can occur when the top half (SCRQ[0:7]) of the queue has been filled (QTHF) and the bottom half (SCRQ[8:15]) of the queue has been filled (QBHF). This may allow for uninterrupted and continuous receives by indicating to the CPU to start reading the queue portion that is now full.
  - The QTHF bit is set by hardware when the top half is full. The QTHF bit is cleared when the SCxSR is read with QTHF set, followed by a write of QTHF to zero.
  - The QBHF bit is set by hardware when the bottom half is full. The QBHF bit is cleared when the SCxSR is read with QBHF set, followed by a write of QBHF to zero.
- In order to implement the receive queue, the following conditions must be met: QRE must be set (QSCI1CR); RE must be set (SCC1R1); QOR and QTHF must be cleared (QSCI1SR); and OR, PF, and FE must be cleared (SC1SR).
- Enable and disable options for the interrupts QTHF and QBHF as controlled by the QTHFI and QBHFI, respectfully.
- 4-bit counter (QRPNT) is used as a pointer to indicate where the next valid data frame will be stored.
- A queue overrun error flag (QOR) to indicate when the queue is already full when another data frame is ready to be stored into the queue (similar to the OR bit in single buffer mode). The QOR bit can be set for QTHF = 1 or QBHF = 1, depending on where the store is being attempted.
- The queue can be exited when an idle line is used to indicate when a group of serial transmissions is finished. This can be achieved by using the ILIE bit to enable the interrupt when the IDLE flag is set. The CPU can then clear QRE and/or RE allowing the receiver queue to be exited.
- For receiver queue operation, IDLE is cleared when SC1SR is read with IDLE set, followed by a read of SCRQ[0:15].
- For receiver queue operation, NF is cleared when the SC1SR is read with NF set, followed by a read of SCRQ[0:15]. When noise occurs, the data is loaded into the receive queue, and operation continues unaffected. However, it may not be possible to determine which data frame in the receive queue caused the noise flag to be asserted.
- The queue is successfully filled (16 data frames) if error flags (FE and PF) are clear, QTHF and QBHF are set, and QRPNT is reset to all zeroes.
- QOR indicates that a new data frame has been received in the data register

(SC1DR), but it cannot be placed into the receive queue due to either the QTHF or QBHF flag being set (QSCI1SR). Under this condition, the receive queue is disabled (QRE = 0). Software may service the receive queue and clear the appropriate flag (QTHF, QBHF). Data is not lost provided that the receive queue is re-enabled before OR (SC1SR) is set, which occurs when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost.

14.9.10 QSCI1 Receive Flow Chart Implementing The Queue

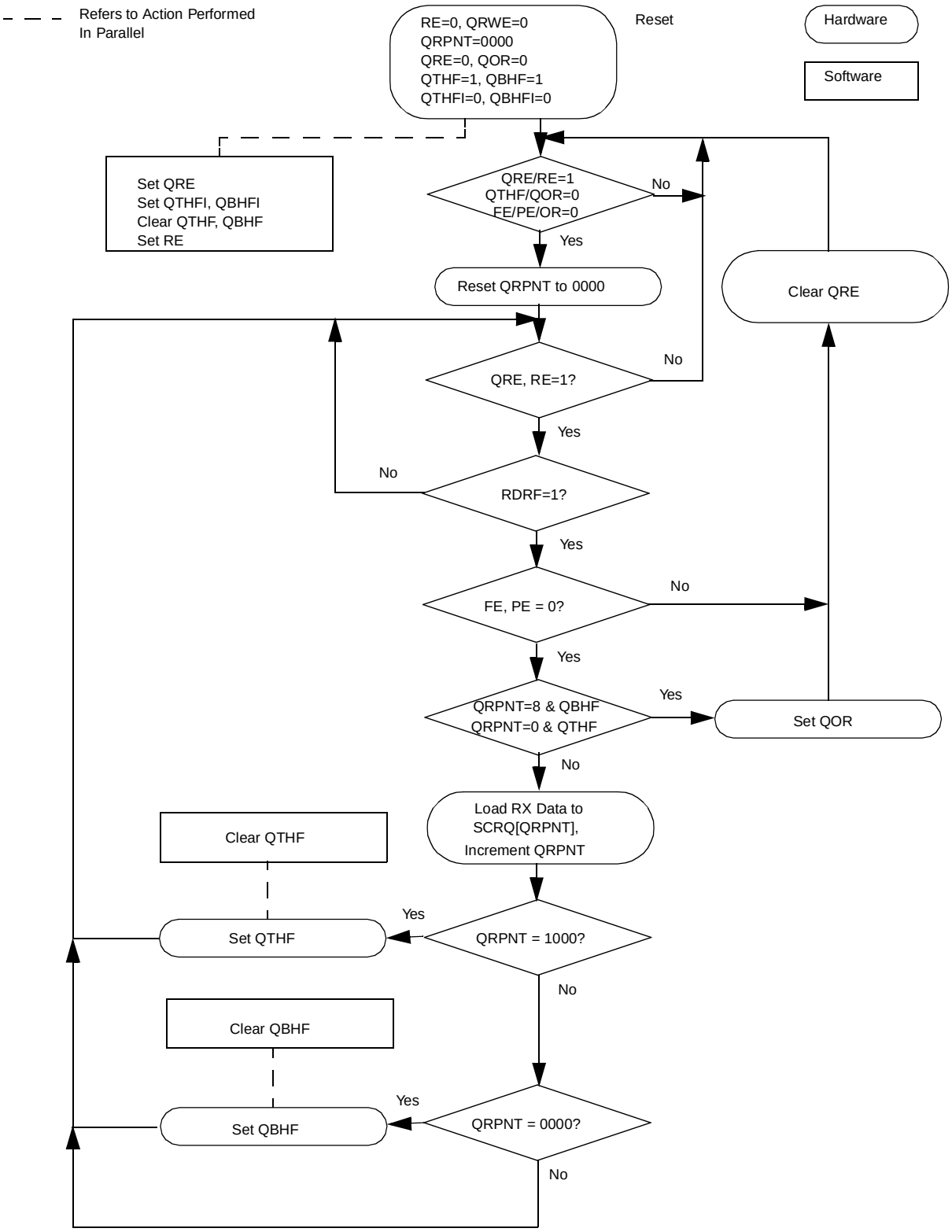


Figure 14-18 Queue Receive Flow

14.9.11 QSCI1 Receive Queue Software Flow Chart

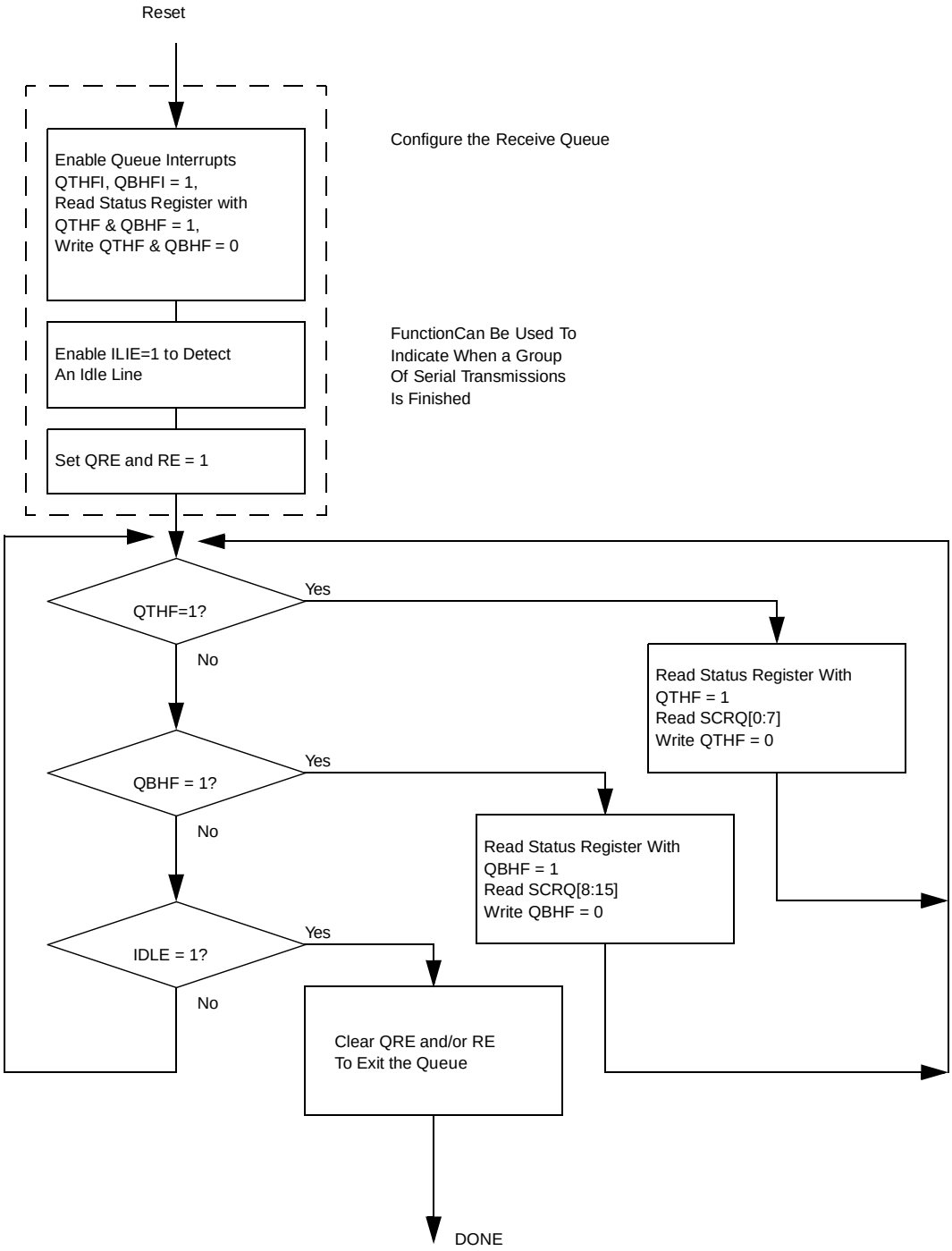


Figure 14-19 Queue Receive Software Flow

14.9.12 Example QSCI1 Receive Operation of 17 Data Frames

Figure 14-20 shows an example receive operation of 17 data frames. The bold type indicates the current value for the QRPNT. Action of the queue may be followed by starting at the top of the figure and going left to right and then down the page.

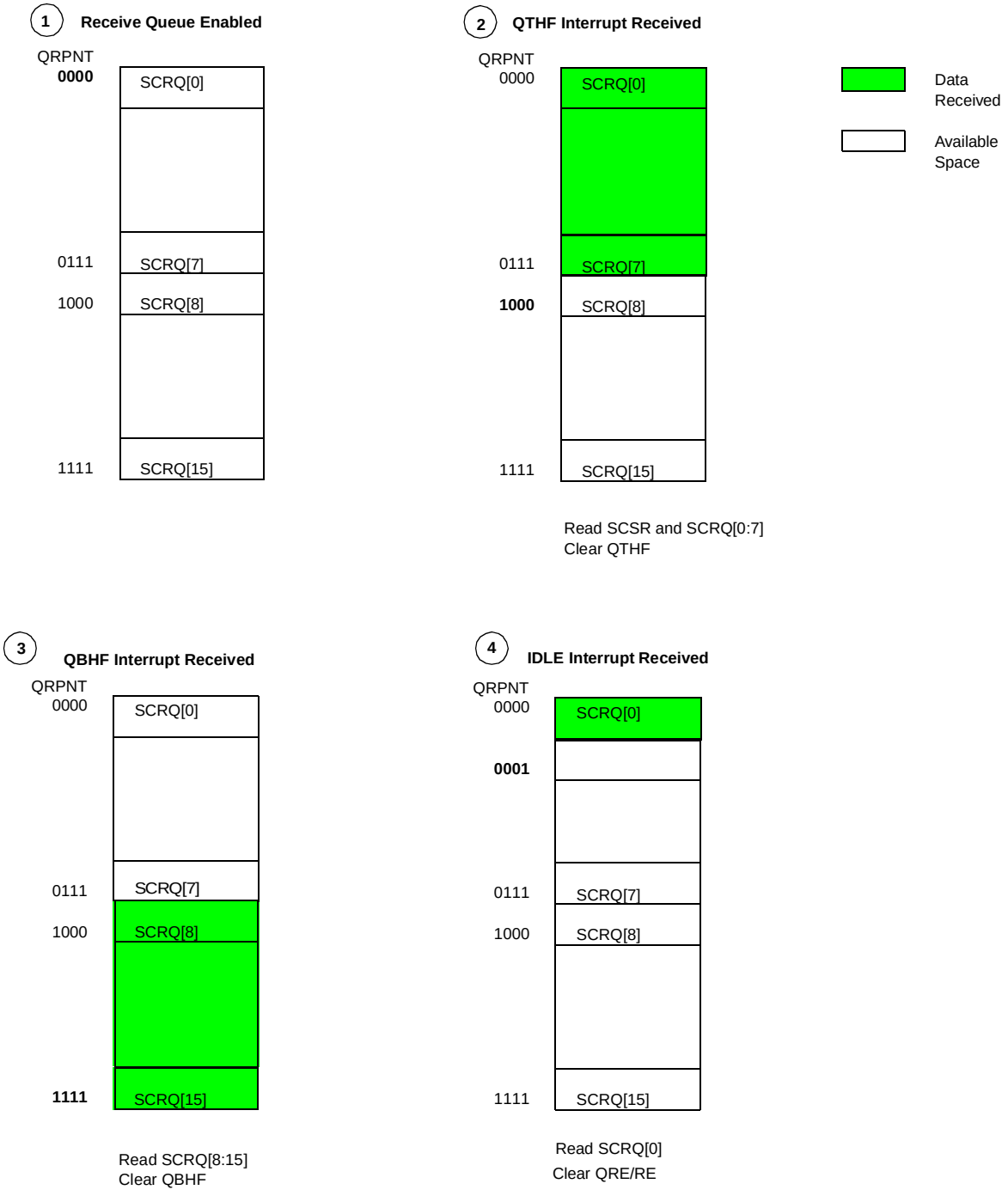


Figure 14-20 Queue Receive Example for 17 Data Bytes

SECTION 15
DATA LINK CONTROLLER MODULE (DLCMD2)

15.1 Scope

This section contains data for the data link controller digital module (DLCMD2). This module is based on the IMB DLCMD module, but has several feature enhancements including those required for the IMB3. The primary purpose of this document is to form the foundation for the functionality and features of the module. This module is designed in a modular structure and is fully compatible with the intermodule bus version 3.

The DLCMD2 is designed with sufficient flexibility to accommodate feature mixes such as byte or symbol-level message buffering.

15.2 Features

The DLCMD2 is essentially the digital portion of a Class B serial data link controller. A separate transceiver is required. The DLCMD2 will provide the following features:

- SAE J1850 compatible
- GM Class 2 compatible
- 10.4 Kbytes/s VPW bit format
- Handles all network protocol functions (access, arbitration, error detection)
- Parallel 16-bit accesses
- All registers are individually addressable
- Polling and IMB3 interrupt generation with vector lookup available
- Transmit buffer first byte can be loaded without a command byte
- Message buffering on transmit and receive
- Eight-bit hardware CRC generation and checking
- No on-board oscillator (uses system clock)
- DLCMD2 logic is clocked from IMB3
- No analog (transceiver) circuit
- Interface to the external transceiver
- Transmit and receive block mode supported
- Transmit and receive 4X mode supported
- Two extra 1-bits sent if lose arbitration on a byte boundary
- Transmitter underrunning indication added to status register
- IMB3 full feature support with option plug selection
- Software programmable prescaler to support two 128-MHz system clock range
- Programmable receiver input polarity
- Programmable normalization bit format
- Digitally filtered receiver
- Power conserving sleep mode with wakeup from bus activity and no loss of data
- IFR type 1, 2 and 3 supported

- Auto retry for loss of arbitration and errors
- Symbol timing control and pre-scaler register
- Symbol timing data register (SDATA)
- Write access to symbol timing parameter table through SDATA

15.3 1.4 Background

The DLCMD2 is an evolution of an earlier module found on IMB MCUs.The analog function, or transceiver, necessary to interface to the J1850 bus will be applied external to the module, and will not be built into the DLCMD2.

15.4 1.5 Applicable Documents

- SAE J1850 ***Class B Data Communications Network Interface***

15.5 1.6 General Requirements

This module is an integrated module for inclusion on-board an IMB3 MCU. Refer to **Table 15-1**.

Table 15-1 DLCMD2 Requirements

Operating Temperature	IC Package	Clock Speed[1]	Quiescent Current Draw
–40 to +125 °C	NA	2.00 Mhz (Min) to 128.00 MHz (Max)[2]	50 µA LPSTOP, 200 µA idle

NOTES:

1. No additional clock part needed for the DLCMD2. The DLCMD2 will adapt to the CPU (IMB3) clocking using software programmable prescaler. DLCMD2 must correctly operate over any clock jitter present within the integrated device.
2. Limited to 56 MHz (Max) on the MPC565 / MPC566.

15.6 Logic Description

The data link controller module (DLCMD2) provides access to an external serial communication multiplex bus, operating according to the SAE J1850 protocol.

This section describes the features, functions, and operation, of the DLCMD2 used as one of several nodes in a vehicle multiplex and/or diagnostic wiring network. All control, status, and message bytes (head of FIFOs only) are accessible as memory mapped registers within the MCU.

The DLCMD2 module supports the SAE J1850 protocol. It retains the maximum throughput performance for single-chip applications including full J1850 message-level buffering but does not provide an internal transceiver.

15.6.1 Block Diagram

A block diagram of the DLCMD2 is shown in **Figure 15-1**.

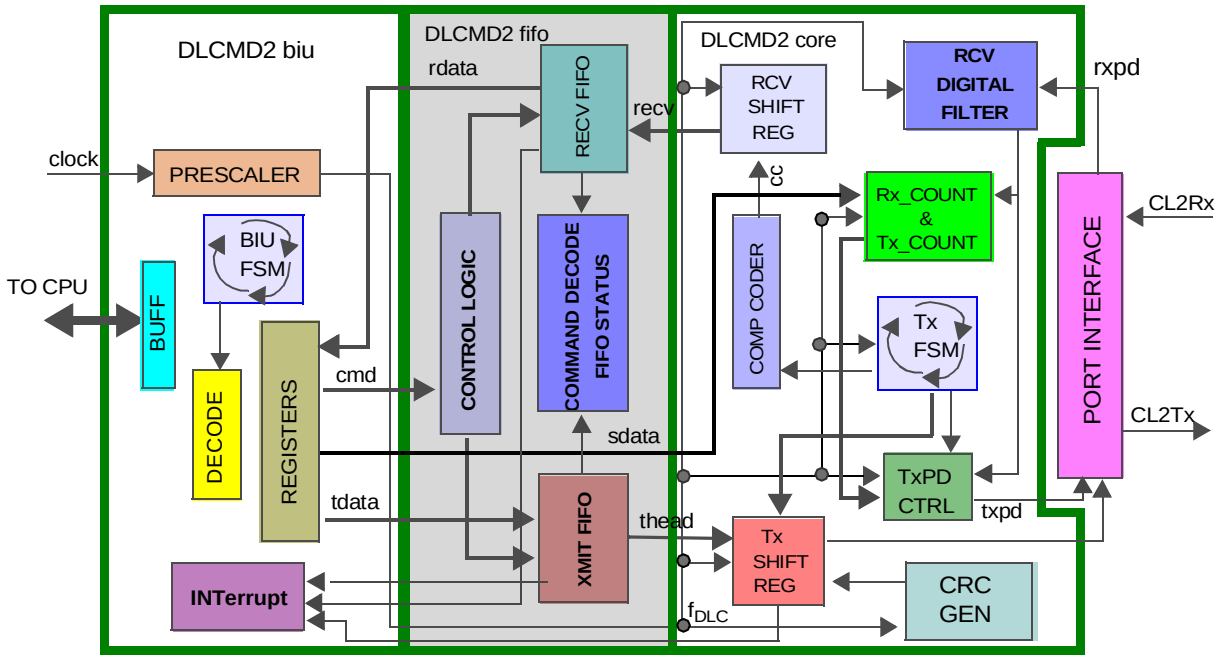


Figure 15-1 DLCMD2 Block Diagram

15.6.2 DLCMD2 Operation

The sections that follow describe the operation of the DLCMD2.

15.6.2.1 General

The DLCMD2 handles J1850 messages with minimal CPU servicing. The MCU will typically transfer complete messages into the DLCMD2 for transmission on the J1850 data link, and is interrupted only when a complete message is received off of the J1850 data link. Internal buffers of 20 bytes on the receive side and 11 bytes on the transmit side allow full message length operations (maximum 12 bytes normal mode, including a one-byte CRC). The DLCMD2 handles all arbitration and error detection duties internally.

The Class 2 data link has been defined as the GM implementation of the SAE J1850 automotive communications protocol. The Class 2 bus is a 10.4 Kbytes/s carrier sense multiple access with collision resolution (CSMA/CR) communications bus. CSMA/CR operates by arbitrating ownership of the bus on a bit-by-bit basis. The major advantage of CSMA/CR is that no message is lost in a collision. If a node determines that it needs to send a message while another message is in progress, it must wait until the link is idle. When one node begins to transmit (after bus idle), all nodes desiring to transmit are obligated to begin their transmission at the same time. The rule of arbitration is that any node that transmits a 1-bit when another node transmits a 0-bit stops transmitting on the bus immediately. This is called a zero-dominant bus. See **15.7 Signals Over-**

view for 1-bit and 0-bit definitions. All nodes are obligated to receive all bits of every message on the bus, even while transmitting. Arbitration continues to the end of a message, if necessary, resulting in a properly transmitted message even if arbitration were to continue to the end of the message. When the bus becomes idle, the node(s) which previously lost arbitration will re-transmit its message if the TxFIFO was not cleared or overwritten with another message.

15.6.2.2 Logic Section Description and Relation to Transceiver

The logic section includes the IMB3 interface, J1850 waveform generation and timing logic, buffers for data (transmit and receive), error detection code generation and checking, configuration logic, control logic, status logic, and arbitration logic.

The nondestructive contention protocol of the J1850 bus requires that there be an active and a passive state of the bus. The bus is in the active or driven state when one or more of the connected transceivers is active and passive when all transceivers are inactive. This is a logical wired OR arrangement.

The function of the transceiver is to drive the bus active in response to a signal from the DLCMD2 logic and to detect the state of the bus for the DLCMD2 logic handler. The transceiver establishes and reliably detects the state of the bus within a limited period of time. It does so in the presence of conducted and induced noise and without creating radio interference. Operation of the transceiver is constrained by available power and the need to tolerate a number of abnormal conditions.

The J1850 bus is intended to work in a relatively noisy environment. The main source of low frequency noise is ground offset between the nodes. Proper operation is assured with any combination of ground offsets up to a maximum differential of two volts at any frequency. Induced noise tends to be short duration pulses. The protocol handler includes a digital filter to remove these pulses. Additional filtering is not needed in the receiver, which responds quickly and avoids stretching large amplitude pulses.

15.6.2.3 DLCMD2 Transmit/Receive Operation

A standard data exchange is composed of one data byte and (in some cases) one command byte going from CPU to DLCMD2, or one status byte and one data byte going from DLCMD2 to CPU. The use of the byte written to the DLCMD2 is specified in the command byte that accompanies it or deduced from current DLCMD2 state. The command byte also contains instructions for the DLCMD2 regarding the receive first in/first out (FIFO) buffer, transmit actions, resetting the transmitter, and sending a break signal. The status byte that the DLCMD2 sends to the CPU contains information on the status of the receive FIFO buffer, the status of the transmit FIFO buffer, the condition of the bus, and the type of accompanying data. The data accompanying the status byte can be data received off of the J1850 bus, a completion code which contains information about a received message, or nothing.

Figure 15-2 shows the DLCMD2 transmit/receive operation.

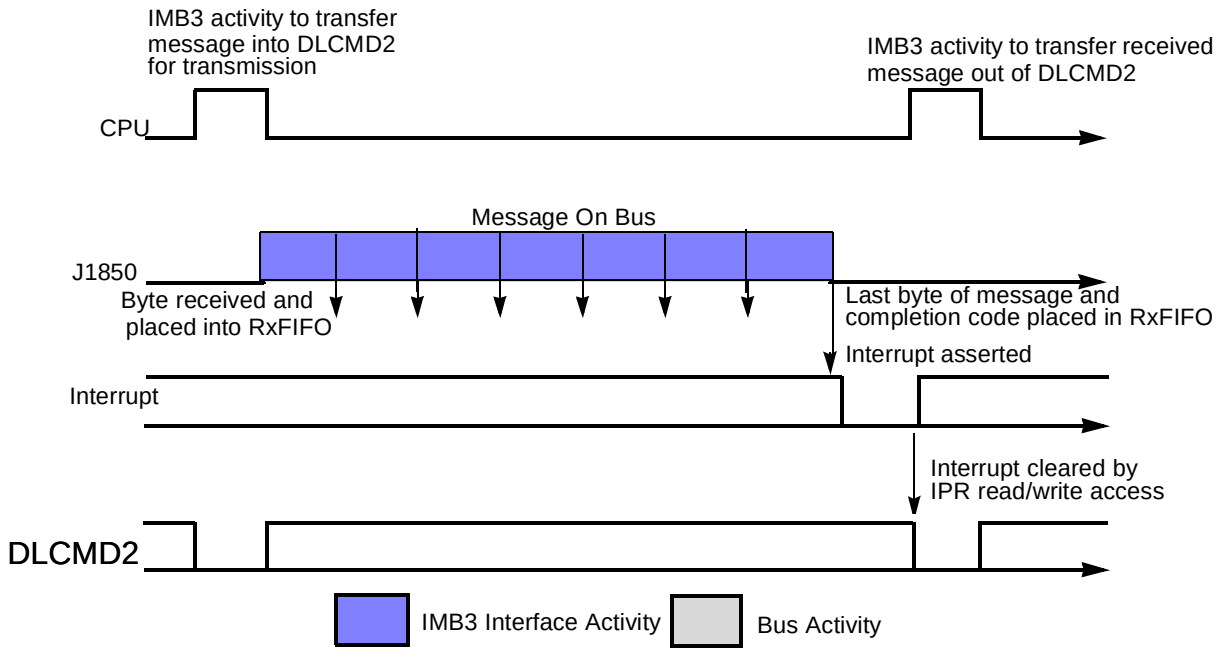


Figure 15-2 Typical Transmit/Receive Operation

All messages received off of the J1850 bus will have their start bit removed and the CRC replaced with a completion code. All other bytes of the message are placed in order, most significant bit (MSB) first, in the receive FIFO buffer.

The DLCMD2 requests servicing by requesting an interrupt. Interrupts may be selected to operate in two different modes, or be disabled. Typically, the DLCMD2 will only interrupt the CPU when a complete message has been received. The CPU then will service the DLCMD2 and remove the message. When the CPU desires to send a message on the J1850 bus, it will select the DLCMD2, and transfer a complete message (without start bit or CRC).

The DLCMD2 allows the CPU to continually write message bytes to be transmitted without intervening command bytes; only the last byte must be accompanied by a command byte indicating “last byte.” User code must read both status and receive data in pairs. The DLCMD2 supports aligned word writes and reads of certain locations.

The DLCMD2 can be programmed by the CPU to enter a power conserving sleep mode as soon as bus traffic stops. If interrupts are enabled, the DLCMD2 will wake-up its internal circuitry and interrupt the host when activity on the bus is sensed. The DLCMD2 will be able to correctly receive the first message that wakes it up in sleep mode.

Error conditions and transmit status, such as lost arbitration, are sent to the CPU either in the status byte or in the completion code that is placed in the receive FIFO immediately after each received message.

15.6.2.4 Message Transmission

As described in the preceding section, the DLCMD2 is loaded with a message from the CPU for transmission. The DLCMD2 will then add a start bit to the outgoing data, and contend for a message slot on the J1850 bus. The transmit buffer in the DLCMD2 is 11 bytes long, to allow complete messages to be transferred to the DLCMD2 for transmission. Information about which byte the CPU is transferring is sent to the DLCMD2 as part of the control information sent with each byte transferred. When the DLCMD2 has transmitted all of the message bytes, it will automatically append a CRC to the end of the message.

The DLCMD2 will automatically retry a transmission if it lost arbitration. The auto retry feature causes the DLCMD2 to signal to the CPU (via the status byte) that the transmit FIFO is full until the message is successfully sent. As soon as the CPU transfers the last byte of a message, the DLCMD2 will indicate that the transmit FIFO buffer (TxFIFO) is full. Once successfully sent, the DLCMD2 will signal the CPU that the TxFIFO is empty and hence ready for the next message. Each time the message loses arbitration, the completion code for that received message will indicate that the transmitter attempted transmission, and lost arbitration. As soon as a transmit slot on the bus becomes open, the DLCMD2 will automatically attempt to retransmit the message. If there were any errors during the transmission of the message the auto retry feature will cause the message to be retransmitted. The auto retry feature can be terminated by the CPU through the command byte. This causes the DLCMD2 to finish its current transmit activity, and then clear the transmit buffer. If there is no transmit activity when the auto retry is disabled and the DLCMD2 previously attempted to transmit, the DLCMD2 will immediately clear the transmit buffer.

15.6.2.5 Message Reception

Receiving information off of the J1850 bus occurs in much the same manner as sending data.

NOTE

By definition, every message a DLCMD2 sends on the data link is also received by its own receiver as if the transmission had been initiated by a different node.

The DLCMD2 will automatically remove the start bit from the message, and check the CRC for errors. If a CRC error occurs, it is flagged in the message completion information (completion code) that takes the place of the CRC byte in the receive FIFO buffer (RxFIFO). The DLCMD2 will interrupt the CPU to signal that a complete message has been received, or when the RxFIFO is approaching full. When the CPU starts the transfer process, the status byte from the DLCMD2 indicates the data's presence, and the amount of data left in the RxFIFO. When a DLCMD2 loses arbitration to another node, it will continue to receive the remainder of the message. The completion

code will indicate that the DLCMD2 contended for a transmit slot, and lost arbitration to the received message. The CPU does not need to resend the message if the auto retry feature is enabled.

The timing of the transmit waveform is re-synchronized on each edge as received off of the bus.

Provisions have been made for immediate in-message reply to allow a path for compatibility with other J1850 implementations. In-frame response (IFR) requires a byte-by-byte interrupt mode and careful CPU attention to accomplish. IFR is described in detail in a later section.

A break/reset waveform on the bus is shaped such that it will win arbitration against any currently transmitting message. When the DLCMD2 senses that such a waveform has occurred on the bus, it will stop transmitting its current message, reset its transmitter (clear the data link controller module (DLCMD2) TxFIFO), and set a bit in the completion code and request an interrupt to indicate to the CPU that such a condition has occurred. Since the break signal always wins arbitration, any in-progress messages will simply lose arbitration, and the DLCMD2 will treat the in-progress received message as complete. If a break occurs while there are no in progress messages, a completion code indicating that a break has occurred will be placed in the RxFIFO and a CPU interrupt generated. The break/reset waveform is sent by a DLCMD2 combination in the command byte. This waveform is sent immediately upon the command byte being latched in. If there is a current transmission, it will be stopped, and the break waveform sent. A break will also take the DLCMD2 out of 4X mode.

15.6.2.6 Sleep Mode

The CPU may put the DLCMD2 in sleep mode by setting the STOP bit in the MCR. Setting this bit will tell the DLCMD2 to halt its internal clocks, immediately after any currently in progress messages are completed.

Interrupts to the host on bus activity can be disabled by configuring the DLCMD2 with the ILR register. Normal use of the sleep feature will have interrupts to the host enabled, so that the host will not miss any messages on the data link. If interrupts are disabled, and then the DLCMD2 is put to sleep, the only way to wake up the DLCMD2 is by the CPU clearing the STOP bit.

15.6.2.7 Debug Mode

This mode is entered from the reset state or from the run state by asserting or deasserting the appropriate signals. See **15.8.5 DLCMD2 DEBUG** for details.

15.6.2.8 4X Speed Mode

The DLCMD2 has the ability to transfer large amounts of data in a 4X speed mode under special conditions such as memory loading at the vehicle factory, and diagnostic responses. There is a bit in the MCR to control this feature. The 4X speed mode affects only the bit timing section of the DLCMD2, including the digital filter. A break will reset

any listening nodes out of 4X speed mode. The 4X speed mode is not for use during normal operation.

To use 4X mode there must be coordination of all nodes on the network. This mode will not work properly at the network level unless ALL nodes are transmitting in 4X mode. Certain nodes may elect not to take part in 4X communications; these nodes may listen but must not transmit.

Notification of entrance into the special 4X mode is communicated to all nodes with a regular speed message indicating the bus protocol speed switch to 4X mode. A BREAK received will automatically take the DLCMD2 out of 4X mode.

15.6.2.9 Block Mode

The DLCMD2 has the ability to receive a message of unlimited length, provided the CPU reads bytes out of the RxFIFO before it overflows.

If the TxFIFO was filled with no last byte indicated, the DLCMD2 will start sending that message in terminate auto-retry mode; the status will indicate “TxFIFO almost full” as soon as the first byte is sent, and “TxFIFO contains some data bytes” as soon as the second byte is sent. As new data is written to the TxFIFO, the status will accurately reflect the condition of the TxFIFO until a “last byte” is written. When a “last byte” command has been sent to the DLCMD2, the TxFIFO will indicate “TxFIFO full” until the transmission is finished. The DLCMD2 will send an infinite length message if properly handled by the CPU.

15.6.2.10 Error Detection

The DLCMD2 uses a digital filter and the cyclical redundancy check (CRC) byte to detect errors.

The digital filter eliminates short duration noise spikes and transition noise from the incoming waveform. It is a “hysteresis” type filter with a time constant of approximately 8 µs, depending on the IMB3 clock frequency. The step response of the filter is a step function delayed by ±8 µs. It can be described by example with a 2-MHz clock (T_{DLC} = 0.5 µs) and a 4 bit up/down counter. The counter counts up for every oscillator clock pulse when the input is in the active state and down when the input is in the passive state. The counter clamps at 0 and 15. The output is defined by **Table 15-2**.

Table 15-2 Digital Filter Output

Count	Output
0	0
1 – 14	Unchanged
15	1

This filter will cause a receive delay of 16-17 times T_{DLC} in addition to the delay in the transmitter and receiver analog interface circuitry. This delay’s only variation is due to the tolerance on the CPU’s oscillator.

In simple terms, the effect of the filter is that a low or high level on the bus is not recognized unless it is longer than about 8 µs.

The CRC byte is used by the receiver to determine if any errors have occurred during transmission. CRC generation uses the divisor polynomial: $X^8 + X^4 + X^3 + X^2 + 1$. The transmitted CRC is generated by the receiver by initially setting the remainder polynomial to all ones, serially processing the first byte and then all remaining bytes of the message, and appending the one's complement of the remainder to the end of the transmitted message. The receiver uses the same divisor polynomial to process all received message bits including the CRC but excluding the start bit. If the transmission is received correctly, at the completion of the message reception, the remainder polynomial will be: $X^7 + X^6 + X^2$ (0b11000100 or 0xC4).

This CRC code will detect all single and 2 bit errors and all 8 bit burst errors (i.e., any number of errors within a single 8-bit span). Severe noise will normally be detected separately as a bit timing error.

15.6.2.11 Arbitration

The J1850 bus is classified as a carrier sense multiple access with collision resolution (CSMA/CR). This type of bus operates by arbitrating ownership of the network on a bit-by-bit basis. The major advantage of CSMA/CR is that no message is lost in a collision. If a node determines that it needs to send a message while another message is in progress, it must wait until the link is idle. When one node begins to transmit (after bus idle), all nodes desiring to transmit are obligated to begin their transmission at the same time. The rule of arbitration is that any node that transmits a 1-bit when another node transmits a 0-bit stops transmitting on the bus immediately. This is called a 0 dominant bus. To prevent noise from corrupting the bus, arbitration is also lost if a 1-bit is detected when a 0-bit was transmitted. All nodes are obligated to receive all bits of every message on the bus, even while transmitting. Arbitration continues to the end of a message, if necessary. If an opposite bit is detected, transmission is immediately stopped unless it occurs on the 8th bit of a byte. In this case the DLCMD2 will automatically append two extra 1-bits and then stop transmitting.

NOTE

Two extra bits must be transmitted due to the fact that the eighth bit of a byte is an active, high level on the J1850 bus. Therefore the first extra bit will be a passive, low level, and one more bit is needed in the active, high level so that after the falling edge of this bit the bus will be in the passive state.

These two extra bits will be arbitrated normally and thus will not interfere with another message.

15.6.2.12 Timebase Generation

The generation of time intervals within the DLCMD2 module takes into account the variations of MCU family, oscillator frequency, and physical interface delays that may occur. The frequency F_{IMB3} is sent to the DLCMD2 where it is further divided by "n" (set

by through the **15.11.5 Symbol Timing Control and Pre-Scaler Register (SCTL)**), such that the main DLCMD2 operating frequency (F_{DLC}) is approximately 2.00 MHz, depending on IMB3 clock frequency.)

The DLCMD2 J1850 bit timings are derived from the f_{DLC} time base and a stored table of VPW symbol values. The table of VPW symbol values is generated via the **15.11.6 Symbol Timing Data Register (SDATA)**.

NOTE

The F_{DLC} signal defines the fundamental resolution of the DLCMD2 module. All bit timings within the DLCMD2 are based upon integer multiples of the fundamental resolution.

Should a physical interface exhibit an unusually large delay, the length of the J1850 transmit symbol values stored in the DLCMD2 symbol table may be reduced pro rata to compensate.

The VPW symbol length table is determined after the symbol lengths have been set via the transceiver REXT bias resistor selection. The REXT resistor values are chosen so as to minimize the radio frequency interference (RFI) from the J1850 bus by inputting a 10.4-kHz square wave into the transmitter and subsequently out on the J1850 bus. These biasing resistors will affect the length of the VPW symbols to some degree due to their effect on the corners of the bus signal that is output by the transmitter.

15.6.2.13 Receive and Transmit Message Buffers

The RxFIFO and TxFIFO are 20 and 11 bytes in length respectively, to allow buffering of a complete message.

The TxFIFO must be able to differentiate between three types of data:

1. Message data byte
2. First byte in message
3. Last byte in message

The auto retry feature recirculates the bytes of a message in the TxFIFO until the message is successfully sent, at which time the FIFO's contents are flushed. When auto retry is disabled, the FIFO will complete an in progress transmission, if any, and then flush the contents of the FIFO. If the node is not transmitting, the FIFO will be flushed immediately. If the auto retry feature is disabled as a message is being loaded into theDLCMD2, the DLCMD2 will try to transmit the message once and then clear the transmit FIFO.

Received bytes will be placed into the RxFIFO as soon as they are completely received off of the bus. When an EOD has occurred on the bus, a completion code will be inserted into the RxFIFO after the last received byte of the message. The CRC byte will be checked by the logic and discarded.

15.6.2.14 Bus Waveforms Generation

The DLCMD2 supports Huntzicker encoding. Each symbol generated by the DLCMD2 will be synchronized with the latest edge seen on the bus. Errors due to oscillator tolerance and ground offsets will not accumulate through the message in this manner. Synchronizing in this manner does require that the bit timing unit account for all known delays. The transmit timing will have a very narrow window due to oscillator tolerance and variation in the known delay only. The receive timing will have much wider windows due to the uncertainty in determining edge position resulting from ground offsets, oscillator tolerance, and delay time variation. In either case, transmit or receive, the timing will be specified as beginning when the DLCMD2 senses a transition, to when the DLCMD2 causes or senses the next transition.

15.6.2.15 Huntzicker Encoding

The information contained in this section describes the bit timing section of the logic on the DLCMD2. The timing of VPW (Huntzicker) waveforms requires knowledge of the fixed delays in the transceiver and the logic section. The J1850 bus is a single wire ground referenced bus. This configuration has two important consequences for the bit timing section. In order to reduce the radiated emissions of the bus, each edge must be slew rate limited, and have its corners shaped. To not adversely affect the corner shaping, the specification must not place limits that force the corners. The other consequence is due to the ground offset requirement for the bus. This requirement dictates a minimal voltage swing necessary to operate in the presence of ground offset. The combination of the two factors gives rise to an uncertainty in both when the receiver (of a receiving node) detects a given transition, and when the transmitter (through its own receiver) detects the same transition.

VPW encoding defines one edge for each symbol. A symbol is composed of a period of time (at a particular state of the bus) and the edge that follows that period. The point of reference for the time period is the trip point that the receiver uses to recognize the preceding transition on the bus. Three independent variables are used to describe the waveform generated. These are the times from the trip point to each of the following transitions threshold levels, and the time between these threshold levels. The corners of the waveform fall outside of the “slew rate” time requirement, and may bargain for time and voltage more freely.

The following symbol limits are consistent with $T_{t,max}$ = 16 µs and an oscillator tolerance of 2%. T_{nom} is the nominal symbol time with no oscillator error and the receiver detecting the transition at $T_{t,max/2}$. T_{R1} to T_{R2} is the required acceptance range while T_{R1TYP} to T_{R2TYP} is a typical acceptance range with a 2% guard band plus a small margin. T_{R1TYP} to T_{R2TYP} in **Table 15-3** represent the receiver windows. T_{x1} and T_{x2} in **Table 15-4** represent transmitter windows consistent with a 2% oscillator tolerance and 3 µs for all other variations in the transmit path.

Table 15-3 Receive Windows

Symbol [1, 2, 3]	T_{rnom}	T_{rmin}	T_{rmax}	T_{r1}	T_{r2}	T_{r1typ}	T_{r2typ}	Units
Short 1/0	64	53	75	37	91	34	96	µs
Long 1/0	128	116	141	100	157	96	163	µs
SOF/EOD	200	186	214	170	230	163	239	µs
EOF	280	265	—	249	—	239	320[4]	µs

1. All waveforms less than 8 µs will be filtered out by the digital filter, and will not be seen as an error.
2. Break is an active symbol that will be transmitted as at least 239 µs in length.
3. All window times include digital as well as analog signal delays.
4. All transmitters are armed as soon as they detect the EOF. The end of the guard band on the EOF serves as a arming point for all transmitters. This is the point that all nodes must have recognized an EOF.

Table 15-4 Transmit Windows

Symbol [1, 2, 3]	T_{xnom}	T_{xmin}	T_{xmax}	Units
Short 1/0	64	60	68	µs
Long 1/0	128	122	134	µs
SOF/EOD	200	193	207	µs
EOF	280	271	289	µs

The symbol waveforms seen on the bus have two important characteristics:

Each transition of the transmitted bus signal, as initiated by CL2Tx (LOTI), is slew rate limited and has its corners rounded (wave shaped) so that the nominal rise or fall time is about 16 µs to reduce the radiated emissions of the bus. This wave shaping is disabled when 4X mode is enabled.

The received bus signal needs only a minimal voltage swing around the receiver's nominal trip point voltage for proper detection. The point of reference for the time period is the trip point voltage (V_T) a receiver uses to recognize a transition on the bus and produce the CL2Rx (LITO) signal.

The CL2Rx signal is digitally filtered with an approximate 8-µs delay at the 10.4-kHz bus rate (2 µs in 4X mode). Since a high or low level input to the filter must last longer than the filter delay time in order to appear at the filter output, noise pulses shorter than this are eliminated.

When a single node is transmitting, the symbol time period between successive transitions is controlled completely by the transmitter's transmit symbol timing logic. When two or more nodes are contending for the bus the start point for an active to passive state transition is determined by the node with the slowest clock rate and the start point for an inactive to active state transition is determined by the node with the fastest clock rate, assuming that both nodes are transmitting the same symbol. The symbol width as controlled by the transmitting node's CL2Tx signal, can range from T_{XMIN} to T_{XMAX}. The receiver's acceptance time window range (T_{RMIN} to T_{RMAX}) is much broader to allow all widths to be classified into defined symbols.

15.6.3 TData Link Controller Module (DLCMD2)

The time windows are not affected by multiple nodes trying to transmit at the same time during arbitration. This is because one node effectively dominates each transition (the first node to leave the passive state or the last node to leave the active state). Although the fastest or slowest node dominates a particular transition, the arbitration scheme assures that the highest priority message always wins.

J1850 bus transmitter output and input signal waveforms are shown in **Figure 15-3**.

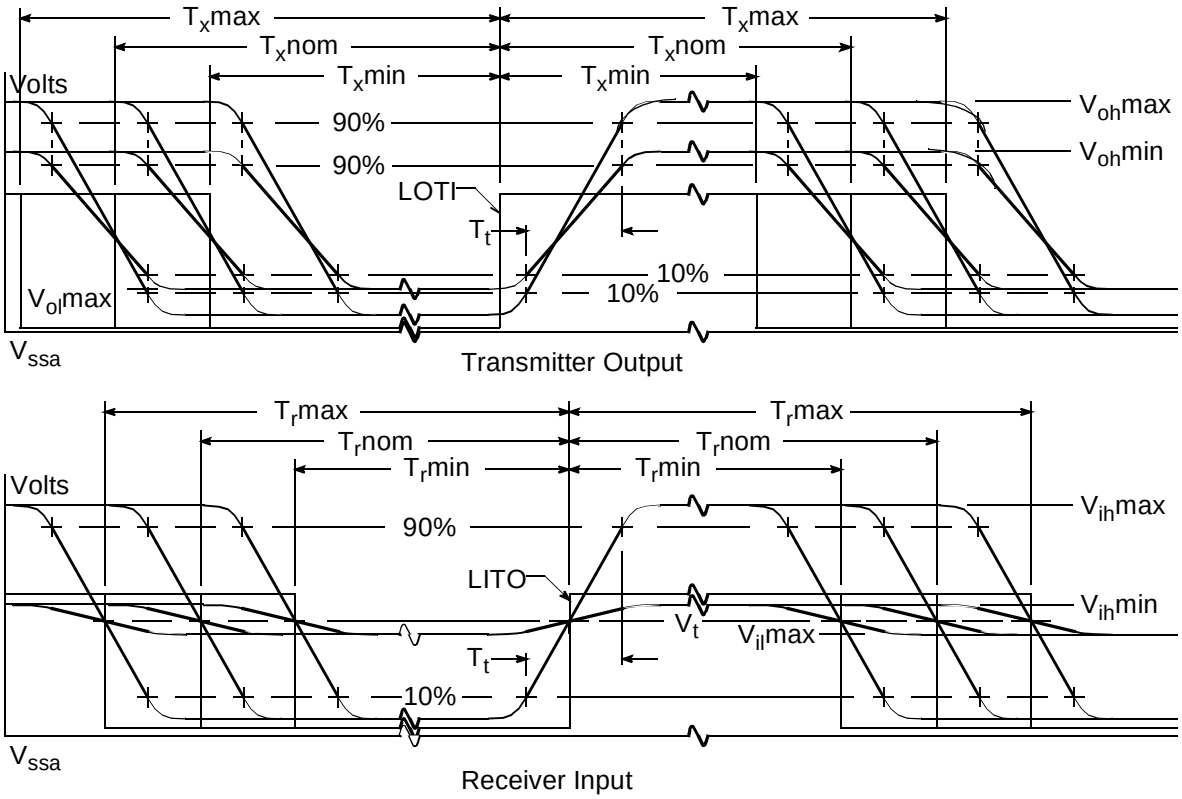


Figure 15-3 VPW Signal Waveforms

NOTES:
1. T_{XMAX} is maximum symbol transmission time.
2. T_{XNOM} is nominal symbol transmission time.
3. T_{XMIN} is minimum symbol transmission time.
4. T_{RMAX} is maximum symbol receive window time.
5. T_{RNOM} is nominal symbol receive window time.
6. T_{RMIN} is minimum symbol receive window time.

15.7 Signals Overview

This section provides an overview of DLCMD2 signals.

15.7.1 J1850 Bus Waveforms

The DLCMD2 module must be able to generate and recognize the set of Huntzicker waveforms described in the following sections. See **Figure 15-4**. Additionally:

- Each symbol is represented by the time between two consecutive transitions
- There is one transition per symbol and one symbol per transition
- There are both active and passive symbols that are used alternately
- A longer active symbol will dominate a shorter one
- A shorter passive symbol will dominate a longer one

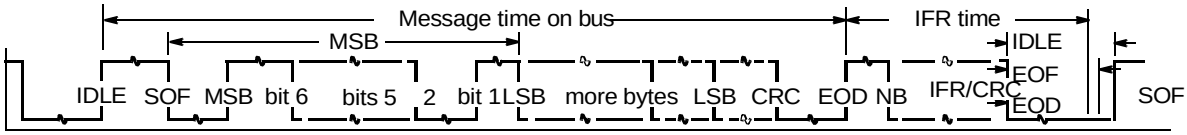


Figure 15-4 Huntzicker Waveform Message

The following sections show the nominal time duration, in microseconds (µs), of the VPW message symbols generated by the DLCMD2 as they appear on the J1850 bus when operating at the normal bus speed. When the DLCMD2 is operating at the high bus speed all 4X symbol times are one fourth that shown, except for “Break”, which will be transmitted the same length in 1X or 4X mode.

15.7.1.1 Start of Frame (SOF)

This active symbol appears at the start of every message when a transmitter drives the bus high to start a message.

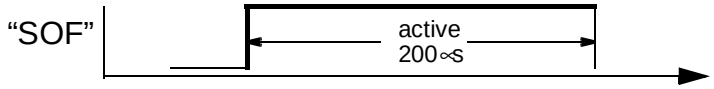


Figure 15-5 Start of Frame Symbol

15.7.1.2 Data Bits

Each data bit is represented by the time between two consecutive transitions. There are both passive and active bit states that are used alternately. The “0” bit is the dominant bit in arbitration.

15.7.1.3 “0” bit

The two dominant “0” bit waveforms are:

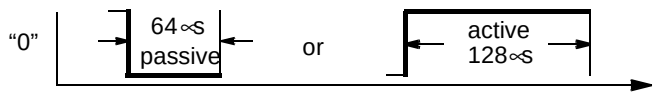


Figure 15-6 Passive "0" and Active "0"

15.7.1.4 "1" bit

The two "1" bit waveforms are:

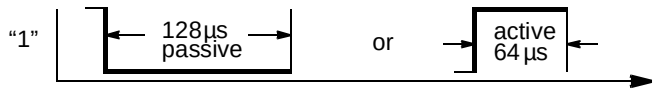


Figure 15-7 Passive "1" and Active "1'

15.7.1.5 End of Data (EOD)

This passive symbol appears after the first CRC byte only in the "request in-frame data" message. It ends when the responding transmitter sends its normalization/format bit prior to the start of the first in-frame response byte. If no node responds, this passive symbol will stretch into an "end of frame" symbol.

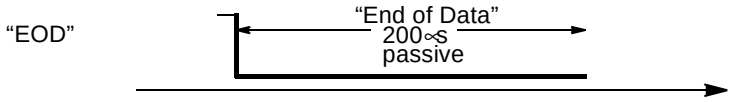


Figure 15-8 End of Data Symbol

15.7.1.6 Normalization Bit

The normalization bit symbol's duration is the same as an active "1" or "0" bit time. The format of this bit, whether it is a "1" or "0"," can be selected by the normalization bit format select (NBFS) bit in the **15.11.5 Symbol Timing Control and Pre-Scaler Register (SCTL)**. J1850 protocol encourages the use of a "0" when the in-frame response (IFR) ends with a CRC byte and a "1" when the IFR does not end with a CRC byte.

15.7.1.7 End of Frame (EOF)

This passive symbol appears at the end of every message. It is at least 280 µs long. If the bus remains passive until 320 µs, the bus is idle and a transmitter may begin transmitting. If a transmitter desiring bus access detects a rising edge on the bus

between 280 µs and 320 µs (due to clock mismatch between nodes) it may join in and arbitrate.

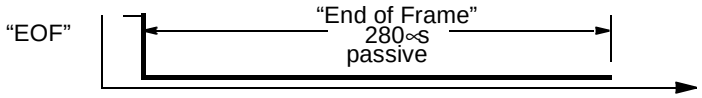


Figure 15-9 End of Frame Symbol

15.7.1.8 Break

The active "Break" signal causes any other transmitting module to stop transmitting immediately because it loses arbitration. It is at least 239 µs long.

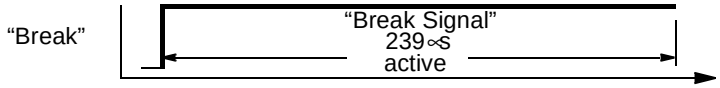


Figure 15-10 Break Symbol Controller Module (DLCMD2)

15.7.2 General Symbol Transmission

The J1850 transmitter will drive the bus to active state and expect that the external RC network will pull the bus back down to the passive state, which is relative since there may be a difference of base ground potential between J1850 nodes in the vehicle. The transmitter is responding to feedback from the receiver in order to know precisely when to switch the transmitter on or off. There is a set of basic transmit timing windows for transmitted symbols within the logic section of the DLCMD2 but if the receiver detects the state of the bus as changing early, the transmitter will also change to that level unless it had not intended to transmit that symbol whereby arbitration is lost and transmission will cease immediately. Thus, all J1850 devices on the J1850 bus synchronize to each other's clock and ground mismatches. Remember that there is only one train of symbols appearing on the bus. The individual symbols are pulled high and released low by various transmitters but the end result is one waveform. It just may be seen differently by the devices due to clock, ground, and power supply variations.

15.7.3 General Symbol Reception

An external transceiver passes unfiltered bus status information to the DLCMD2's Rx pin. Internal to the DLCMD2, the digital 1 or 0 is clocked through a digital delay filter for 16 ticks of its internal frequency clock (a delay of 8 µs at normal (2-MHz F_{DLC}) speed before the filter output changes state. High and low levels on the J1850 bus are timed in the logic section and compared to a set of received symbol threshold windows. Every received high or low level is translated into one of the symbols in the above sections or is flagged as a bit timing error.

The bus is “idle” when the output of the DLCMD2’s digital filter has been in the passive state for 320 µs.

15.7.4 Support For External Transceiver

As shown in **Figure 15-11**, the DLCMD2 will be designed to use an external (IC) transceiver. Along with the CL2Tx and CL2Rx signals, a 4XEN signal will be provided at the periphery of the DLCMD2 to disable transceiver waveshaping during DLCMD2 4X mode.

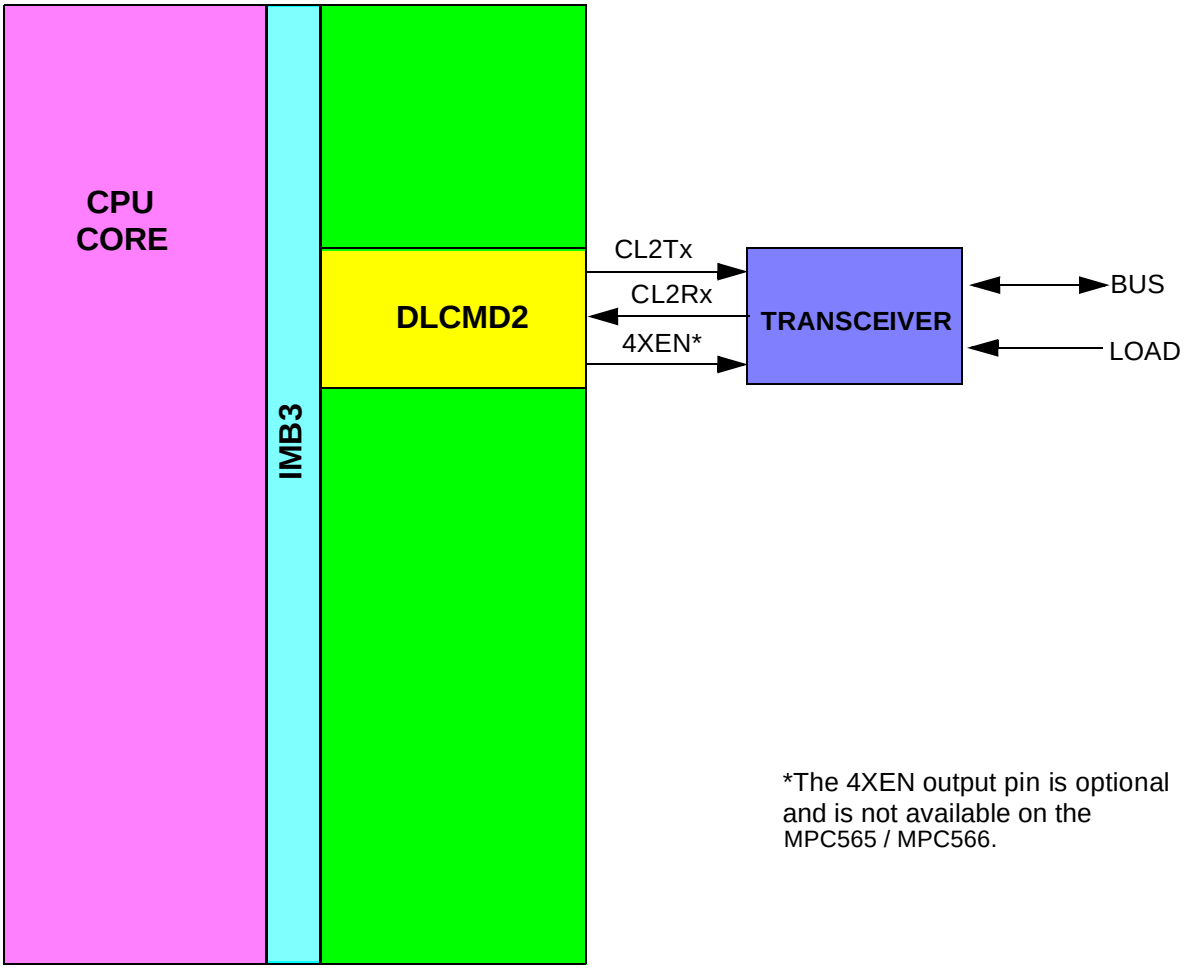


Figure 15-11 Support For External Transceiver

15.8 Operating Modes

This section describes DLCMD2 operating modes. The DLCMD2 has five main modes of operation which interact with the power supplies, pins, and the rest of the MCU. Refer to **Figure 15-12**.

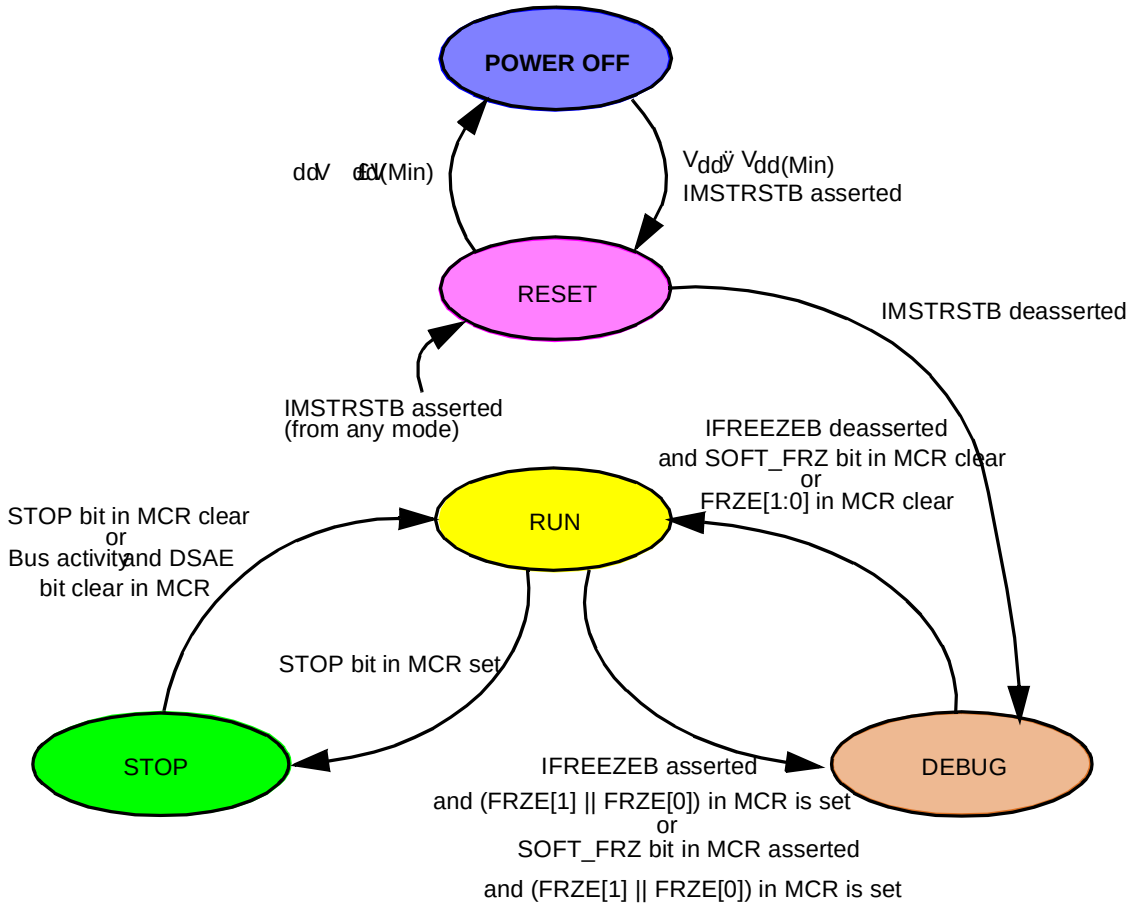


Figure 15-12 DLCMD2 Operation Modes

15.8.1 Power Off

This mode is entered from the reset mode whenever the DLCMD2 supply voltage V_{DD} drops below its minimum specified value for the DLCMD2 to guarantee operation. This implies that the DLCMD2 must be placed in the reset mode before being powered down. In this mode, the pin input and output specifications are not guaranteed.

15.8.2 Reset

This mode is entered from the power off mode whenever the DLCMD2 supply voltage V_{DD} rises above its minimum specified value and IMSTRSTB is asserted. This implies that IMSTRSTB must be asserted while powering up the DLCMD2 or an unknown

state will be entered and correct operation cannot be guaranteed. It is also entered from any other mode on the falling edge of CLOCK after IMSRSTB is asserted.

In this mode V_{DD} is supplied to the internal circuits, which are held in their reset state and the internal DLCMD2 system clock is running. Registers will assume their reset condition. Outputs are held in their programmed reset state, inputs and network activity are ignored.

15.8.3 Run

This mode is entered from the debug mode after all MCU reset sources are no longer asserted. It is entered from the DLCMD2 STOP mode whenever a message is successfully received or the CPU has accessed the DLCMD2 and negated the STOP bit (if previously set).

It is entered from the DLCMD2 LPSTOP mode whenever network activity is sensed although messages will not be received properly until the clocks have stabilized and the CPU is also in the run mode.

In this mode, normal network operation takes place. The user should ensure that all DLCMD2 transmissions have ceased before exiting this mode.

15.8.4 DLCMD2 STOP and LPSTOP

15.8.4.1 DLCMD2 STOP mode

This mode is automatically entered from the run mode whenever the CPU executes a STOP instruction. The IMB3 clocks continue to run, but the CPU clock is stopped.

In this mode, the DLCMD2 internal clocks continue to run and the module will await a valid network message. If a valid network message is successfully received, a CPU interrupt request will be generated (if interrupts are enabled).

Controller module (DLCMD2) DLCMD2 power is only conserved in this mode if the STOP bit in the MCR is set, stopping the DLCMD2 clocks.

15.8.4.2 DLCMD2 LPSTOP mode

This power conserving mode is automatically entered from the run mode whenever the CPU executes a LPSTOP instruction.

In this mode, the DLCMD2 internal clocks are stopped and the module will await any J1850 activity (including noise). If network activity is sensed, then a CPU interrupt request will be generated, restarting both the IMB and DLCMD2 internal clocks.

15.8.5 DLCMD2 DEBUG

This is a special debug mode entered by asserting the SOFT_FRZ bit in the MCR register, or by asserting IMB3 IFREEZEB line. For both, activating the DLCMD2 debug mode is qualified by the FRZE[1:0] in MCR register.

Upon exiting the reset state the SOFT_FRZ bit and FRZE[1:0] bits are set in the MCR register. Hence, reset mode is always followed by the debug mode.

Once this mode is set, the following occurs:

- The pre-scaler divider is stopped, thus halting all related activities.
- Any activity on the J1850 bus will be ignored. The DLCMD2 ignores the CL2Rx input pin and drives CL2Tx to the passive state.
- The CPU can read and write into most of DLCMD2 registers except for otherwise noted in the register description section.
- The NOT_RDY and FREEZ_ACK bits in MCR register are set.
- After asserting the debug mode configuration bits, the FREEZ_ACK bit will be set in MCR register, before executing any other action to the DLCMD2; otherwise the DLCMD2 may operate in an unpredictable way.

Exiting the debug mode is done in one of the following ways:

- Both, IMB3 freeze and SOFT_FRZ bits are negated.
- CPU negates the FRZE bit.
- Once debug mode is exited, the DLCMD2 is ready to transmit/receive normally on the J1850 bus.

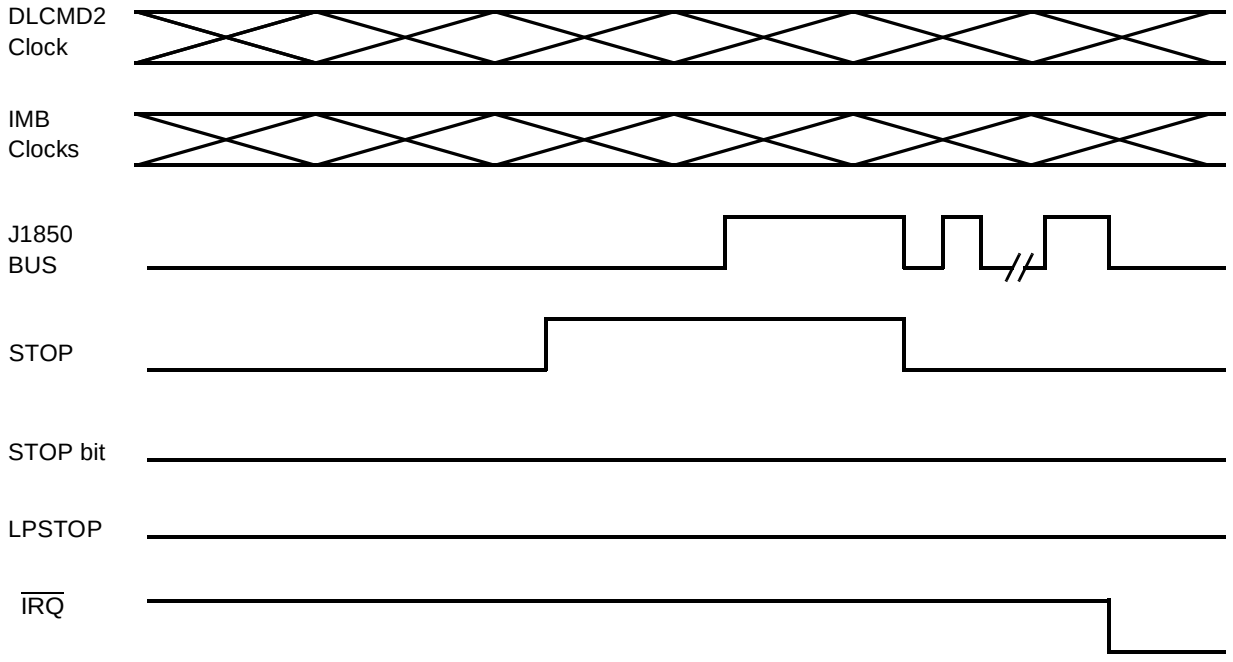


Figure 15-13 STOP Power Mode (No STOP Bit Set)

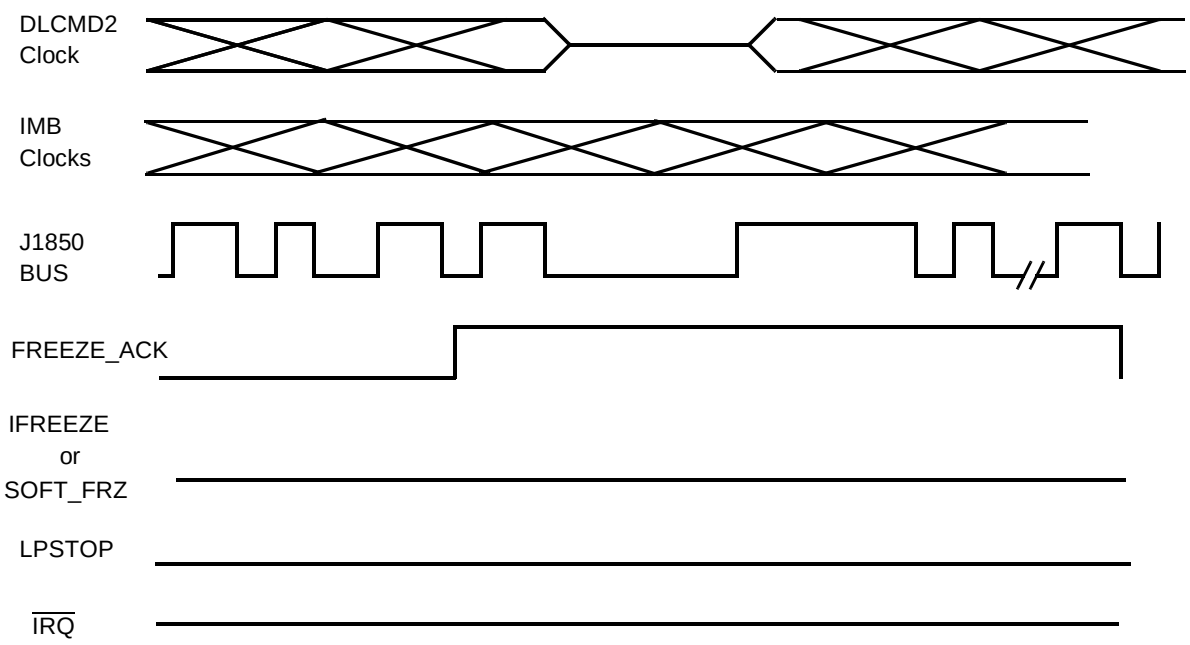


Figure 15-14 DEBUG Power Mode (IFEEZEB or SOFT_FRZ)

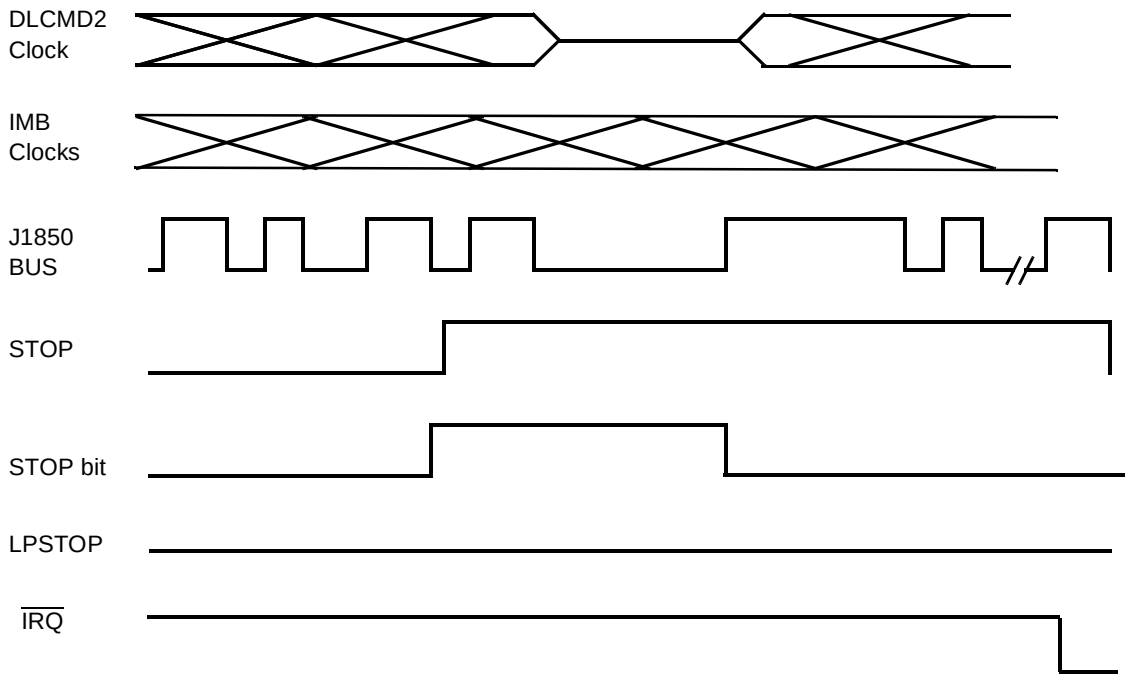


Figure 15-15 STOP Power Mode (STOP Bit Set)

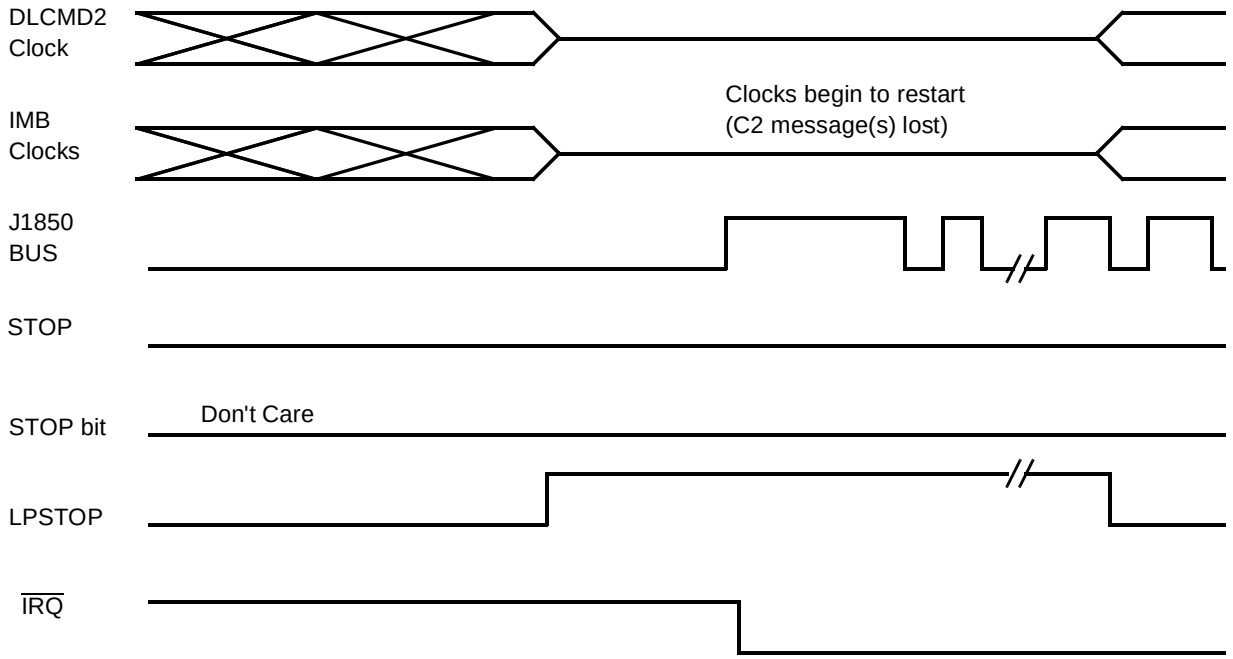


Figure 15-16 LPSTOP Power Mode

15.9 CPU Interface

This section covers DLCMD2 interfaces to the CPU.

15.9.1 Parallel Interface Requirements

This section defines the de-multiplexed parallel interface protocol used.

The logical operations done in a DLCMD2 module service routine are as follows:

1. Read status byte from DLCMD2 module
2. Read received data from DLCMD2 module if required
3. Write command byte to DLCMD2 module if required
4. Write transmit data to DLCMD2 module

These four bytes are a complete exchange of information. However, these bytes are not all necessary or required during a CPU/DLCMD2 data link controller module (DLCMD2) module transfer. All possible unique CPU/DLCMD2 module transfers are shown in **Table 15-5**. All CPU/DLCMD2 module transfers will be made up of combinations of one or more of these building blocks. Pop and push refer to automatic (without command byte) flushing and loading of data bytes out of and into the DLCMD2 module.

A transmit data byte need not immediately follow the command. The transmit data byte must be the next byte transferred to the DLCMD2 but could be sent any time later. The command will not be acted upon until this next byte is sent to the DLCMD2.

Table 15-5 CPU/DLCMD2 Transfers

Acceptable Read/Write Combinations	Auto Pop?	Auto Push?	First Byte Flag Set?	Word Read/ Write Allowed?
1. Read status byte from DLCMD2[1]	No	No	No	No
2. Write command byte to DLCMD2	—	No	No	Yes
Write transmit or dummy data to DLCMD2 module	—	No	Maybe	Yes
If no data accompanies, then command byte causes no action	—	—	—	—
3. Read status byte from DLCMD2	No	—	—	Yes
Read received data byte from DLCMD2	Yes	—	—	Yes
Read status byte from DLCMD2	No	—	—	Yes
Read received data byte from DLCMD2	Yes	—	—	—
(Until entire message received)	—	—	—	—
4. Write transmit data byte to DLCMD2	—	Yes[2]	Yes	No
Write transmit data byte to DLCMD2	—	Yes	No	No
Write command byte to DLCMD2 (load as last byte)	—	No	No	Yes
Write transmit data to DLCMD2 (last byte)	—	No	No	Yes

NOTES:
1. Minimal transfer
2. If TxFIFO empty

In numbers 2-4, word reads and writes may be done to read a status byte and a received data byte, or write a command byte and a transmit data byte with one instruction. If number 2 was for loading a single byte for transmit the command byte would specify load as first and last byte (no auto push, no first byte flag set). Number 3 would be for reading an entire message from the DLCMD2 module when there is no transmit data to be sent to the DLCMD2 module. Number 4 is a quick way to load an entire message into the DLCMD2 module when there is no data to read from the DLCMD2 module.

Auto pop is the default for reading from the RxFIFO. This means that a read of a data byte from the RxFIFO causes the current byte in the FIFO to be automatically flushed. Auto pop can not be disabled. Auto push is the automatic loading of a data byte into the TxFIFO via a write to the DLCMD2 module. This is the default when there was no preceding command byte. There is no auto push if a command byte is sent before the data byte; and the command byte must specify what to do with the following data byte.

The action(s) called out in the command byte will be acted upon the moment the transmit (or dummy) data byte is written into the DLCMD2 module.

15.9.2 Reset Operation

When master reset is asserted, the DLCMD2 module will be held in an off state. System power should be up and stable when master reset is negated.

After toggling the reset line, the CPU writes command and any configuration bits (via MCR) into the DLCMD2 module to initialize it.

15.10 Operational Information

The following sections will be included in the DLCMD2 application document and are mostly redundant information to previous sections.

15.10.1 Initialization

After power up and/or reset the DLCMD2 should be configured via writes to the MCR, ILR, IVR, SDATA, and SCTL registers.

NOTE

Interrupts are disabled at power up and reset and must be enabled if a polled method of servicing the DLCMD2 is not used.

15.10.1.1 Step 1 — Initialize MCR

Begin initialization of the configuration bits by writing the SUPV bit in the MCR. This will determine what types of accesses are to be allowed (user and supervisor or just supervisor) to DLCMD2 registers for the rest of the initialization process and during normal operation. Care should be taken not to clear the SOFT_FRZ or FRZ[1:0] bits while writing the SUPV bit. This precaution will ensure the DLCMD2 remains in the debug mode until initialization is complete.

15.10.1.2 Step 2 — Initialize ILR and IVR registers if Interrupts Employed

15.10.1.3 Step 3 — Initialize SCTL and SDATA registers

15.10.1.4 Step 4 — Enable DLCMD2 by exiting DEBUG mode

15.10.2 Transmitting a Message

The DLCMD2 is loaded with a message from the CPU for transmission. The DLCMD2 will then add a SOF bit to the outgoing data, and contend for a message slot on the J1850 bus. The TxFIFO in the DLCMD2 is 11 bytes long, to allow complete (except block) messages to be transferred to the DLCMD2 by the CPU for transmission. Information about which byte the CPU is transferring is sent to the DLCMD2 as part of the control information sent with each byte transferred and is optional with the DLCMD2 except for the last byte. When the DLCMD2 has transmitted all of the message bytes, it will automatically append a CRC to the end of the outgoing message.

The DLCMD2 will automatically retry a transmission if it lost arbitration or any errors were detected. The auto retry feature causes the DLCMD2 to indicate that the TxFIFO is full until the message is successfully sent except where no last byte was indicated. As soon as the CPU transfers a “last byte” of a message to the DLCMD2, or fills the eleventh position of the TxFIFO, the DLCMD2 will indicate that the TxFIFO is full. In the case of filling the TxFIFO with no “last byte” indicated (block mode) the status will

say “TxFIFO almost full” after a byte has been sent so that the data link controller module (DLCMD2) next byte of the block message can be loaded by the CPU. Once successfully sent, the DLCMD2 will indicate that the TxFIFO is empty and hence ready for the next message. Each time the message loses arbitration, the completion code for that received message will indicate to the transmitter that it attempted transmission, and lost arbitration. The auto retry feature can be terminated by the CPU through the command byte. This causes the DLCMD2 to finish its current transmit activity, and then clear the TxFIFO. If there had not been any transmit activity when the auto retry is disabled, the DLCMD2 will attempt to transmit the message once and then clear the TxFIFO.

A break waveform on the bus is shaped such that it will win arbitration against any currently transmitting message. When the DLCMD2 senses that such a waveform has occurred on the bus, it will stop transmitting its current message, reset its transmitter (clear the TxFIFO), and set a bit in the completion code and assert an interrupt to indicate to the CPU that such a condition has occurred. Since the break signal wins arbitration all of the time, any in progress messages will have simply lost arbitration, and the DLCMD2 will treat the in progress received message as complete. If a break occurs while there are no in progress messages, a completion code indicating that a break has occurred will be placed in the RxFIFO and a CPU interrupt generated. The break/reset waveform is sent by a DLCMD2 by setting a bit combination in the command byte. This waveform is sent immediately upon the command byte being latched in. If there is a current transmission, it will be stopped, and the break waveform sent. Break automatically takes all nodes on the network out of 4X speed mode.

NOTE

An SOF from a node in regular mode will be seen as a break by any nodes in 4X mode.

15.10.3 Receiving a Message

The status byte that the CPU reads from the DLCMD2 contains information on the status of the RxFIFO, the status of the TxFIFO, the condition of the bus, and the type of accompanying data byte. The data accompanying the status byte can be data received off of the J1850 busa completion code which contains information about a received message, or nothing.

All messages received off of the J1850 bus will have their start bit removed and error detection code replaced with a completion code. All other bytes of the message are placed in order, MSB first, in the RxFIFO.

Receiving information off of the J1850 bus occurs in much the same manner as sending data.

NOTE

By definition, every message a DLCMD2 sends on the data link is also received by its own receiver as if the transmission had been initiated by a different node.

The DLCMD2 will automatically remove the SOF bit from the message, and check the CRC for errors. If a CRC error occurs, it is flagged in the message completion information (completion code) that takes the place of the CRC byte in the RxFIFO. The DLCMD2 will interrupt the CPU to signal that a complete message has been received, or when the RxFIFO is approaching full. When the CPU starts the transfer process, the status byte from the DLCMD2 indicates the data's presence, and something about the amount of data left in the RxFIFO. Conditions necessary for interrupting the CPU are selectable with a write to the interrupt level register (ILR). When a DLCMD2 loses arbitration to another node, it will continue to receive the remainder of the message. The completion code will indicate that the DLCMD2 contended for a transmit slot, and lost arbitration to the received message. The CPU does not need to resend the message if the auto retry feature is enabled.

15.10.4 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the DLCMD2 does allow for a special "block mode" of operation for the receiver. As far as the DLCMD2 is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Data link controller module (DLCMD2) Class 2 convention requires that another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

Block mode may be combined with 4X mode.

15.10.5 Transmitting a Message in Block Mode

If the TxFIFO was filled with no last byte indicated, the DLCMD2 will start sending that message in terminate auto-retry mode; the status will indicate "TxFIFO almost full" as soon as the first byte is sent, and "TxFIFO contains some data bytes" as soon as the second byte is sent. As new data is written to the TxFIFO, the status will accurately reflect the condition of the TxFIFO until a "last byte" is written. When a "last byte" command has been sent to the DLCMD2, the TxFIFO will indicate "TxFIFO full" until the transmission is finished. The DLCMD2 will send an infinite length message if properly handled by the CPU.

Block mode transmit may be combined with 4X mode.

15.10.6 Receiving a Message in 4X Mode

Although not a part of the SAE J1850 protocol, the DLCMD2 does allow for a special "4X mode" of operation for the receive. 4X mode is entered in software by setting the 4X bit in the MCR register. This bit is cleared automatically by a BREAK symbol reception, or may be manually cleared in software.

15.10.7 Transmitting a Message in 4x Mode

If the the 4X mode bit is set in the MCR register, the DLCMD2 will use a different set of VPW timing numbers to set the widths of the J1850 bus symbols when transmitting. This bit is cleared automatically by a BREAK symbol reception, or it may be cleared manually.

15.11 Register Descriptions

The following section provides register descriptions for the DLCMD2. In the following paragraphs, the top line lists the bit number in the register. The second line contains the mnemonic for the bit. The values shown under the mnemonic are the values of those register bits after IMSTRSTB is asserted. If ISYSRSTB affects the bits differently than IMSTRSTB, this fact is discussed in the paragraphs following the chart.

The DLCMD2 registers hold, in some cases, bit locations marked as “reserved”. These bits will always read as logic ‘0’ and writes to these bits are ignored.

The register decode map is fixed and begins at the first address of the module base address. **Table 15-6** shows the registers associated with the DLCMD2 module and their relative offset from the base address. Four of the registers are in supervisor-only data space and the remainder are in assignable data space.

Table 15-6 DLCMD2 Memory Map

Access	Offset	15	0	R/$\overline{W}$	Reset
S	0x30 0080	Module Configuration Register (MCR)		R/$\overline{W}$	0x6780
S	0x30 0082	RESERVED		—	—
S	0x30 0084	Interrupt Pending Register (IPR)		R/$\overline{W}$	0x0000
S	0x30 0086/ 0x30 0087	Interupt Level Register (ILR)	Interrupt Vector Register (IVR)	R/$\overline{W}$, RO (bit2~bit0)	0x000F
S/U	0x30 0088	Symbol Timing Control and Pre-scaler Register (SCTL)		R/$\overline{W}$	0x0000
S/U	0x30 008A	Symbol Timing Data Register (SDATA)		WO	0x00XX
S/U	0x30 008C/ 0x30 008D	Command Register (CMD)	Transmit Data Register (TDATA)	R/$\overline{W}$-WO for (TDATA)	0x00XX
S/U	0x30 008E/ 0x30 008F	Status Register (STAT)	Receive Data Register (RDATA)	RO	0x00XX

15.11.1 Module Configuration Register (MCR)

MCR— Module Configuration Register **0x30 0080**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FR[1:0]		DSAE	4XMD	SOFT	NOT_	FREE	SUPV	RESERVED		STOP	RESERVED			
RESET:															
0			0	0	1	1	1	1	0	0	0	0	0	0	0

Table 15-7 MCR Bit Descriptions

Bit(s)	Name	Description
15	STOPSTP[3:0]	Stop System Clock — The STOP bit, if asserted, will stop the system clock within the DLCMD2 module except for the IMB bus interface (BIU). The module's BIU must continue to operate to allow the CPU access to the module's registers (except for LPSTOP). The system clock is stopped on the low phase of ICLOCK. Once the bus becomes idle, the system clock will remain stopped until the STOP bit is negated by the CPU, or a reset occurs, or an edge from the J1850 bus passes through the digital filter (if DSAE not set).
14:13	FR[1:0]	Freeze — There exists two bits in the DLCMD2 module control register to determine the action to be taken when the Freeze signal of the IMB is asserted. **Table 15-8** defines the freeze bit field.
12	DSAE	Disable STOP Mode Automatic Exit — When asserted, the DSAE bit will prevent J1850 activity from causing the DLCMD2 to exit STOP mode, and will prevent INTACL2-type interrupts from being asserted. (When DSAE is negated, the DLCMD2 will automatically restart its internal clocks from STOP mode upon sensing any J1850 activity, and INTACL2-type interrupts will be allowed (although they must still be explicitly enabled by INTACL2E).) The CPU must negate the STOP bit to exit DLCMD2 STOP mode and restart the DLCMD2 clocks, since J1850 activity will not take the DLCMD2 out of STOP mode. When cleared, any J1850 activity that passes through the digital filter will take the DLCMD2 out of STOP mode and clear the STOP bit (if set).
11	4XMD	4X Mode — When the 4XMD bit is asserted, the DLCMD2 will use 4X mode bit timings rather than the normal J1850 symbol timings. Note that normal waveshaping by the analog transceiver (whether on-chip or off-chip) must be disabled for the DLCMD2/transceiver combination to transmit properly in 4X mode. This bit is automatically reset upon receipt of a BREAK symbol.
10	SOFT_FRZ	Software Freeze — Assertion of this bit has the same effect as the assertion of the IFREEZEB signal on the IMB3, as described in the description of IFREEZEB/SOFT_FRZ mode and FRZE bits. However, it does not require that the IFREEZBE signal be asserted in order to enter debug mode. This bit is initialized to 1 (debug mode).The CPU clears it after initializing the control registers. No transmissions or receptions are performed by the DLCMD2 before this bit is cleared. For detailed description of the debug mode, refer to **15.8.5 DLCMD2 DEBUG**. 0 = No DLCMD2 internal request to enter Debug Mode. 1 = Enter Debug mode if (FRZE[1] || FRZE[0])
9	NOT_RDY	NOT_RDY — This bit indicates that the DLCMD2 is either in STOP or in DEBUG states. This bit is read-only. Whenever one of these two modes is asserted, this bit is set once the DLCMD2 has entered the corresponding mode. It is negated once the DLCMD2 has exited these modes. For more details refer to **15.8.4 DLCMD2 STOP and LPSTOP** and LPSTOP and **15.8.5 DLCMD2 DEBUG**.
8	FREEZ_ ACK	DLCMD2 Disabled — DLCMD2 is in debug mode and its pre-scaler is stopped. This bit is read-only. The value of this bit is one when the DLCMD2's pre-scaler is stopped, and 0 when debug mode is negated and the pre-scaler is running again. When the DLCMD2 enters debug mode it sets the FREEZ_ACK bit. The CPU can poll this bit to know if the DLCMD2 entered debug mode. If debug mode is negated then this bit is negated once the DLCMD2's pre-scaler is running.

Table 15-7 MCR Bit Descriptions (Continued)

Bit(s)	Name	Description
7	SUPV	Supervisor/User Data Space — The SUPV bit affects the SCTL/SDATA, CMD/TDATA and STAT/RDATA registers, the only registers in the DLCMD2 currently defined as supervisor/unrestricted access. If SUPV is asserted (all DLCMD2 registers are to be treated as supervisor only) bit 2 of the function code (FC2) must be asserted during module address decoding to allow the supervisor/unrestricted access registers to respond to accesses in supervisor data space. If the SUPV bit is cleared (SCTL/SDATA, CMD/TDATA and STAT/RDATA accesses are to be treated as unrestricted) FC2 is ignored during module address decoding, allowing supervisor/unrestricted access registers to respond to accesses in supervisor data or user data space.
6:5	—	Reserved.
4	STOP_ACK	DLCMD2 is STOPPED — DLCMD2 is in STOP mode and its main clocks are stopped. This bit is read-only. The value of this bit is '1' when the DLCMD2 enters STOP Mode and its clocks are stopped, (see **15.8.4 DLCMD2 STOP and LPSTOP**) and '0' when stop mode is negated and the DLCMD2's clocks are running again. When the DLCMD2 enters STOP mode and shuts its clocks off, it sets the STOP_ACK bit. The CPU can poll this bit to know if the DLCMD2 entered stop mode (e.g., stopped its clocks). If stop mode bit is negated, then this bit is negated once the DLCMD2's clocks are running.
3:0	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in DLCMD2 implementations that use hardware interrupt arbitration.

Table 15-8 Freeze Bit Field Description

FRZ[1]	FRZ[0]	Result
0	0	Ignore FREEZE
0	1	Freeze on DLCMD2 internal F_{DLC} high state. (CTWO is in high state when IFREEZEB is asserted, or F_{DLC} enters its high state sometime after IFREEZEB is asserted.)
1	0	Freeze on receipt of next bit. (DLCMD2 internal RIT signal is asserted IFREEZEB is asserted, or RBIT becomes asserted sometime after IFREEZEB is asserted.)
1	1	Freeze immediately

Read restrictions: supervisor only, bits[6:5] are reserved but can be read as zeroes.

Write restrictions: supervisor only, bits[6:5] are reserved.

15.11.2 Interrupt Pending Register (IPR)

IPR — Interrupt Pending Register **0x30 0084**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED											IPR4	IPR3	IPR2	IPR1	IPR0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-9 Interrupt Pending (IPR) Bit Descriptions

Bit(s)	Name	Description
15:5	—	Reserved.
4:0	IPR[0:4]	Interrupt pending. Indicates that an interrupt service request has been made by the module interrupt logic. An interrupt request to the IMB3 is generated whenever the interrupt asserting condition is met. A bit in the IP register (indicating the application logic interrupt request is detected) can be set in any order to generate the IMB3 interrupt request. Once set, the IP bits remain set until the IP bit is cleared by software, or reset. To clear an IP bit, the bit must be first read as a 1 and then the bit must be written to a 0. IP bits which are 0 when the IP register is read are unaffected by the write operation. Also, if the IP logic detects another application logic interrupt request after the IP bit was read as a 1 and before a 0 is written to clear it, the IP bit cannot be cleared until the IP bit is again read as a 1 and then written to a 0. Bits in this register are set by the application logic request and cannot be written to a one by software (writing 1 to the IP register have no effect). Only writes of 0 are valid, when permitted, to clear the IP bit(s). IPR[0] : R1STBYTE interrupt pending sets IPR[0]. IPR[1] : RCCODE interrupt pending sets IPR[1]. IPR[2] : R13BYTE interrupt pending sets IPR[2]. IPR[3] : THLFMTY interrupt pending sets IPR[3]. IPR[4] : ACL2 interrupt pending sets IPR[4].

Read restrictions: supervisor only, bit[15:5] are reserved but can be read as zeros.

Write restrictions: supervisor only, bit[15:5] are reserved.

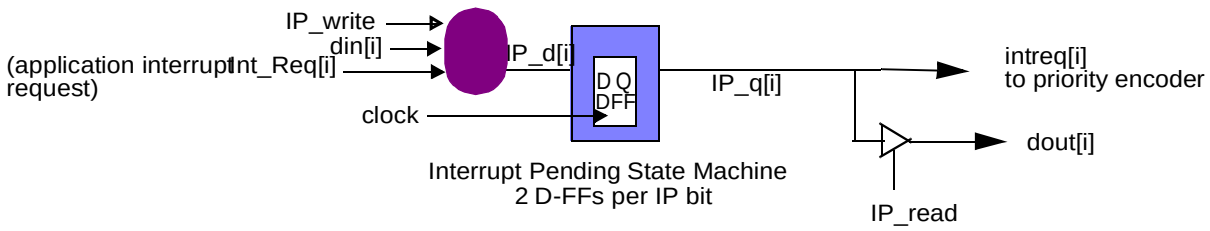


Figure 15-17 Interrupt Request Logic Path

15.11.3 Interrupt Level Register (ILR)

ILR — Interrupt Level Register **0x39 0086**

MSB 7	6	5	4	3	2	1	LSB 0
INTMODE	INTACL2E	RESERVED	ILBS[1:0]		ILR[2:0]		
RESET:							
0	0	0	0	0	0	0	0

Table 15-10 Interrupt Level (ILR) Bit Descriptions

Bit(s)	Name	Description
15	INTMODE	Interrupt Mode — When the INTMODE bit is asserted, the DLCMD2 will assert an interrupt when a single byte is received into an empty RxFIFO. When cleared, only standard interrupts are enabled. **Table 15-11** defines interrupt levels.
14	INTAC L2E	Interrupt Any Bus activity Enable — When the INTACL2E bit is set, the DLCMD2 will assert an interrupt when any network activity (including noise) is detected. This bit must be set for the DLCMD2 to wake up the processor from LPSTOP. Although the detected activity may be only noise, neither IMB3 nor module clocks can run in LPSTOP, therefore the DLCMD2 cannot distinguish noise from valid network activity, and must wake up the processor to restart the clocks necessary for J1850 message reception. Normally, this bit would be set by the CPU just before going into LPSTOP, and is cleared once the ACL2 interrupt condition is latched.
13	—	Reserved.
12:11	ILBS[1:0]	Interrupt Level Byte Select — These two bits indicate during which time slot the DLCMD2 should drive its interrupt. For more details, refer to **15.8.4 DLCMD2 STOP and LPSTOP** and **15.8.5 DLCMD2 DEBUG** for IRQ_PLUG=1.
10:8	ILR	Interrupt Request Level — The interrupt request level field contains the priority level of the DLCMD2 interrupts for the CPU. Level seven for this field indicates a nonmaskable interrupt, while level zero indicates that interrupts have been disabled. The interrupt request level field is initialized to zero during reset which prevents the module from generating an interrupt until this register has been initialized. **NOTE:** Level zero corresponds to IRQ[0] which is not recognized at the system level, hence the interrupts are treated as disabled. The interrupt request level field, therefore acts as master enable for the interrupts.

Read restrictions: Supervisor only, bit[5] is reserved but can be read as zero.

Write restrictions: Supervisor only, bit[5] is reserved.

Table 15-11 Interrupt Levels

ILBS [1:0]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

15.11.4 Interrupt Vector Register (IVR)

IVR — Interrupt Vector Register **0x30 0087**

MSB 7	6	5	4	3	2	1	LSB 0
IVR[7:0]							
RESET:							
0	0	0	0	0	0	0	0

Table 15-12 Interrupt Vector (IVR) Bit Descriptions

Bit(s)	Name	Description
7:0	IVR[7:0]	Interrupt Vector Field — The interrupt vector register holds the offset into the exception vector table, and is what is driven by the DLCMD2 in response to an IMB IACK cycle. IVR[7:3] are programmable. IVR[2:0] are read-only bits and encoded from the highest priority of any currently active interrupt sources per **Table 15-12**. The IVR is not used by the MPC565 / MPC566.

Read restrictions: Supervisor only.
Write restrictions: Supervisor only.

15.11.5 Symbol Timing Control and Pre-Scaler Register (SCTL)

SCTL — Symbol Timing Control and Pre-Scaler Register **0x30 0088**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED	NBFS	RxPOL	RESERVED			LCK	SEL	RESERVED		PS5	PS4	PS3	PS2	PS1	PS0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 15-13 Symbol Timing Control and Pre-Scaler Bit Descriptions

Bit(s)	Name	Description
15	—	Reserved.
14	NBFS	This bit controls the format of normalization bit (NB). SAE J1850 strongly encourages the use of an active long, ‘0’, for in-frame responses containing CRC and active short, ‘1’, for in-frame response without CRC. Once the LCK is set, the writes to NBFS are disabled. 0 = NBFS, NB that is received or transmitted is a ‘1’ when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘0’ when the response part of an in-frame response (IFR) does not end with a CRC byte. 1 = NBFS, NB that is received or transmitted is a ‘0’ when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a ‘1’ when the response part of an in-frame response (IFR) does not end with a CRC byte.
13	RxPOL	Receive Pin Polarity (Bit 13) — The receive pin polarity bit is used to select the polarity of incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding back to the digital receive pin. Once the LCK is set, the writes to RxPOL are disabled. 0 = RxPOL, select normal/true polarity; true non-inverted signal from J1850 bus, (i.e., the external transceiver does not invert the receive signal). 1 = RxPOL, select inverted polarity, where external transceiver inverts the receive signal.
12:10	—	Reserved.

Table 15-13 Symbol Timing Control and Pre-Scaler Bit Descriptions (Continued)

Bit(s)	Name	Description
9	LCK	Write Lock on Symbol Timing Parameter Table (Bit 9) — LCK disables writes to the symbol timing parameter table through SDATA. Once LCK is set, only reset will clear the bit (unless in test mode). 0 = LCK, enables the writes to the symbol timing parameter table through SDATA register. 1 = LCK, disables the writes to SEL, NBFS, RxPOL, PS5 -PS0 in SCTL register and the symbol timing parameter table through SDATA register. Once LCK is set, only reset will clear this bit (unless in test mode).
8	SEL	Select Normal or 4X Bit-timing Parameter Table (Bit 8) — There are two sets of parameters used for DLMCD2 bit-timing. These two sets of parameters are accessible through the SDATA register. There are twelve parameters for normal bit-timing (four for receiving and eight for transmitting) and twelve parameters for 4X bit-timing (four for receiving and eight for transmitting). These bits may be written only when LCK is cleared. Once the LCK is set, the writes to SEL are disabled and SEL is cleared. 0 = SEL, allows access (write) to the normal mode parameters. 1 = SEL, allows access (write) to the 4X mode parameters.
7:6	—	Reserved.
5:0	PS[5]-PS[0]	DLCMD2 Pre-Scaler Rate Select (Bit [5:0]) Set the system clock divisor necessary to achieve the DLCMD2 internal bit-rate clock. The frequency should be as close to 2 MHz as possible. These bits may be written only when LCK is cleared. Once the LCK is set, the writes to PS5-PS0 are disabled. The value programmed into PS5-PS0 bits is dependent on the chosen IMB3 system clock frequency per **Table 15-14**.

Read restrictions: Supervisor/Unrestricted access, bit[15], bits[12:10], bits[7:6] are reserved but can be read as zeroes.

Write restrictions: Supervisor/Unrestricted access, bit[15], bits[12:10], bits7:6] are reserved.

NOTE

The PS[5]-PS[0] is always loaded with “desired divisor” -1 and comes out of reset programmed for a divided by one clock rate (PS[5]-PS[0] = 0x00).

Table 15-14 DLCMD2 Pre-Scaler Rate Selection

IMB3 Bus Clock Frequency	PS5-PS0	Division	F_{DLC}
f_{CLOCK} = 2.000 MHz	0x00	1	2.000 MHz
f_{CLOCK} = 15.000 MHz	0x06	7	2.143 MHz
f_{CLOCK} = 16.000 MHz	0x07	8	2.0 MHz
f_{CLOCK} = 25.000 MHz	0x0B	12	2.083 MHz
f_{CLOCK} = 26.000 MHz	0x0C	13	2.0 MHz
f_{CLOCK} = 40.000 MHz	0x13	20	2.0 MHz
f_{CLOCK} = 45.000 MHz	0x15	22	2.045 MHz
f_{CLOCK} = 66.000 MHz	0x20	33	2.0 MHz
f_{CLOCK} = 128.000 MHz	0X3F	64	2.0 MHz

15.11.6 Symbol Timing Data Register (SDATA)

Refer to **Figure 15-18** for SDATA symbol timing table access read restrictions: supervisor/unrestricted access, the entire register will be read as zeroes.

SDATA— Symbol Timing Data Register **0x30 008A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					SDATA[10:0]										
RESET:															
0	0	0	0	0	0	0	0								

Table 15-15 Symbol Timing Data (SDATA) Bit Descriptions

Bit(s)	Name	Description
15:11	—	Reserved.
10:0	SDATA	Symbol Timing Data Register — The bit-timing of J1850 symbols is written into the twenty-four entries of the parameter table through the SDATA register. An internal pointer along with the SEL bit is used to select which parameter is accessed (write) through the SDATA register. This pointer is incremented when SDATA is written and cleared when the SCTL register is accessed. Refer to **Figure 15-18** for a block diagram and **Table 15-16** for the parameter table.

Write restrictions: supervisor/unrestricted access when LCK in SCTL register is clear. Bit [15:11] are reserved.

S10-S0 — symbol timing data register controller module (DLCMD2)

The parameter table lists the cycle counts for four receive symbols and eight transmit symbols in normal and 4X modes. The user must calculate the cycle counts for each symbol based on the desired value, round trip delay, digital filter delay, and bus frequency. The calculated cycle counts must be entered into the parameter table through the **15.11.6 Symbol Timing Data Register (SDATA)**. To calculate the cycle count for receive symbols, multiply F_{DLC} by the desired symbol time (in µs) and round to the nearest integer. To calculate the cycle count for transmit symbols, subtract the round trip delay for the transceiver from the desired symbol time (in µs), multiply by F_{DLC}, round to the nearest integer, and subtract the cycle count of the digital filter (16 cycles in normal mode, four cycles in 4X mode). As an example, let the transceiver round trip delay be 16 µs, the desired symbol be 64 µs, and F_{DLC} be 2.083 MHz. The cycle count would be:

cycle count = ((64 µs - 16 µs) x 2.083 cycles / µs) - 16 cycles = 84 cycles

After the cycle count has been computed, it must be converted into binary format and entered into the respective parameter through the **15.11.6 Symbol Timing Data Register (SDATA)**.

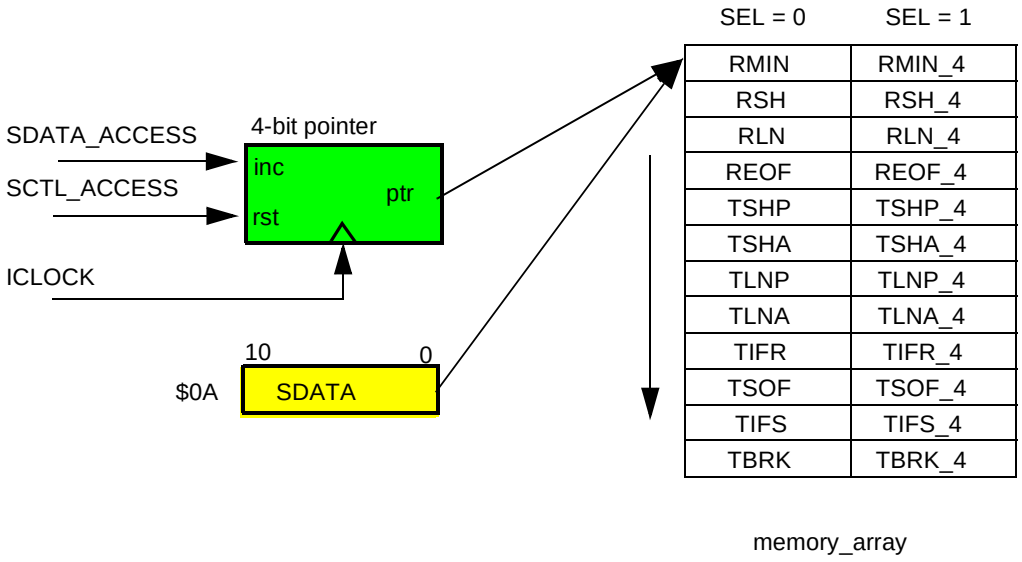


Figure 15-18 DLCMD2 SDATA Block Diagram

Table 15-16 Timing Parameter Table

Programming Sequence	Symbol Description	Symbol (SEL = 0)	Symbol Timing (µs)	Symbol (SEL = 1)	Symbol Timing 4X (µs)
1	Minimum receive symbol time	RMIN	34	RMIN_4	8
2	Maximum receive short pulse time	RSH	96	RSH_4	24
3	Maximum receive long pulse time	RLN	163	RLN_4	41
4	Minimum end-of-frame time	REOF	239	REOF_4	60
5	Target transmit short passive symbol time	TSHP	64	TSHP_4	16
6	Target transmit short active symbol time	TSHA	64	TSHA_4	16
7	Target transmit long passive symbol time	TLNP	128	TLNP_4	32
8	Target transmit long active symbol time	TLNA	128	TLNA_4	32
9	Target end-of-data time before IFR begins	TIFR	200	TIFR_4	50
10	Target transmit start-of-frame symbol time	TSOF	200	TSOF_4	50
11	Target inter-frame separation time before DLCMD2 is ready for new message	TIFS	320	TIFS_4	80
12	Target transmit break symbol time	TBRK	800	TBRK_4	800

15.11.7 Transmit Command Register (CMD)

Command and configuration of the DLCMD2 module is accomplished through a command byte that accompanies every data byte sent from the CPU to the DLCMD2 module.

NOTE

The command byte is the first byte transferred to the DLCMD2 module, and must be followed by a byte (data or dummy) which causes the command to be acted on (this can be done in an aligned word write or two separate byte writes).

CMD— Transmit Command Register 0x30 008C

MSB 7	6	5	4	3	2	2	LSB 0
CMD[7:0]							
RESET:							
0	0	0	0	0	0	0	0

Table 15-17 Transmit Command Register (CMD) Bit Descriptions

Bit(s)	Name	Description
7:0	CMD	Transmit Command Register — The command byte can be broken in three fields. Bits 7:5 describe general commands. Bits 4:3 describe the type and destination of the accompanying byte associated with the command. Bit 1:0 control the receive FIFO. Refer to **Table 15-18** for general command descriptions. Refer to **Table 15-19** for type and destination of accompanying byte descriptions. Refer to **Table 15-20** for RxFIFO command descriptions.

Read restrictions: Reads will return 0x00.
Write restrictions: Supervisor/Unrestricted access.

Table 15-18 General Commands

CMD 7	CMD6	CMD5	Description
0	0	0	Do nothing — DLCMD2 will not perform any actions defined in this field.
0	0	1	Reserved — Results in “do nothing.”
0	1	0	Send Break signal — When this command is latched in, the DLCMD2 will immediately send a break signal on the bus, regardless of its current transmit or receive status.
0	1	1	Send on EOD with CRC (Transmit Single or Multiple Byte IFR with CRC) 1. Must be accompanied by a data byte that is flagged as “1st byte” or “1st and last” byte. The remaining IFR bytes may not be sent to the DLCMD2 with the “Send on EOD” combination set. 2. This command causes the DLCMD2 to go into Terminate Auto Retry (TAR) mode automatically. It also causes a reset of the transmitter and TxFIFO before the response byte is loaded. If the DLCMD2 is transmitting and loses arbitration, an IFR can be loaded in response to the message winning arbitration. 3. If this command is latched in while the bus is idle or between EOD and EOF, the IFR command/data bytes will be ignored. If valid command/data bytes follow the invalid IFR command/data bytes, the DLCMD2 will attempt to transmit the bytes as a normal message once. 4. If a message is in progress and the EOD symbol has not been detected when this command is latched in, the DLCMD2 will send the IFR if there were no receiver errors detected. If receiver errors were detected, the IFR will be lost and the DLCMD2 will not be in TAR mode. 5. If the remaining IFR bytes, if any are required, are not placed into the TxFIFO before they are needed, then an underrun will occur. This will cause the CRC to be inverted in the IFR. The second byte must be loaded before the falling edge of the Normalization Bit. The next IFR byte must be loaded before the last bit of the current byte is transmitted. Since there is only one responder transmitting an IFR, the inverted CRC will cause the DLCMD2 to corrupt its own message. Other nodes will receive this as a CRC error. 6. If this DLCMD2 is the one transmitting, send on EOD will cause a bit timing or incomplete byte error and a reset of the TxFIFO. Both the original message and the IFR will be lost. 7. Cannot be sent in response to an IFR without CRC.
1	0	0	Terminate Auto Retry (TAR) 1. If this command is latched in while there is not a complete message in the TxFIFO, then when that message is completely loaded the DLCMD2 will attempt to transmit that message only once. After this transmit attempt, the TxFIFO will be empty. 2. If this command is latched in while the DLC is transmitting, then the DLCMD2 will finish its transmit activity (successful transmission, or lose arbitration) and then clear the TxFIFO. 3. If a complete message is in the TxFIFO, but not yet transmitting when this command is latched in, and: a. The message has just lost arbitration and is waiting for the next slot for retransmission, the DLCMD2 will attempt transmission one more time. b. The message has not tried to transmit yet (loaded while a message was on the bus). The DLCMD2 will try to transmit once, and then reset the TxFIFO. c. If the TxFIFO is full, and there is no last byte indicated (message of more than 11 bytes), then an automatic TAR is executed.

Table 15-18 General Commands (Continued)

CMD 7	CMD6	CMD5	Description
1	0	1	Send on EOD without CRC (Transmit Single or Multiple Byte IFR w/o CRC) 1. Same as 0 1 1 but a CRC will not be transmitted. 2. This response cannot be sent after a previous IFR without CRC.
1	1	0	Send on EOD with auto re-try (Transmit Single Byte IFR with auto re-try) 1. If a loss of arbitration occurs when the DLCMD2 attempts to transmit and after the IFR byte winning arbitration completes transmission, the DLCMD2 will again attempt to transmit. The DLCMD2 will continue transmission attempts until an error is detected on the bus. 2. The TxFIFO will not be empty until the message is successfully sent.
1	1	1	Abort Transmission Now (Reset Transmitter) 1. This command causes the transmitter, including the TxFIFO to be reset immediately. If a message is in progress, it will be terminated immediately. 2. The first byte of a new message may accompany this command in the same 2-byte host-DLCMD2 transfer.

Table 15-19 Type and Destination of Accompanying Byte Commands

CMD 4	CMD 3	CMD 2	Description
0	0	0	Do Not Load as Transmit Data 1. The DLCMD2 will not perform any actions defined in this field.
0	0	1	Load as Transmit Data 1. If the TxFIFO is indicating full (complete message already in TxFIFO), then the byte is not loaded and is lost.
0	1	0	Reserved
0	1	1	Load as Last Byte of Transmit Data 1. If there is not a “1st byte” at the head of the TxFIFO, the byte is not loaded and is lost unless a block transfer is in progress. 2. If the TxFIFO is empty, the byte is not loaded and is lost. 3. If the TxFIFO is indicating full (complete message already in TxFIFO), then the byte is not loaded and is lost.
1	0	0	Reserved
1	0	1	Load as First Byte of Message 1. If there is already a first byte in the TxFIFO, the byte is loaded, and will cause a transmit underrun error during transmission of this message. The first “first byte” and subsequent bytes will transmit correctly until the second “first byte” is encountered and this byte is not transmitted and is lost. 2. If the status of the TxFIFO is full (complete message already loaded, or partial message < 11 bytes), then the data byte is not loaded, and is lost.
1	1	0	Reserved
1	1	1	Load as First and Last Byte of Message 1. If the TxFIFO is full, then the byte is not loaded and is lost.

15.11.8 Transmit Data Register (TDATA)

Table 15-20 RxFIFO Commands

CMD 1	CMD 0	Description
0	0	Do Nothing
0	1	Reserved (Do nothing)
1	0	Flush Byte 1. If there is nothing in the RxFIFO, then no action is taken even if a byte arrives before the DLCMD2 access is complete. 2. If there are any bytes in the RxFIFO, then the first byte (at the head of the RxFIFO) is flushed. 3. Flush byte commands will not queue up.
1	1	Flush Current Message[1] 1. If there is nothing in the RxFIFO when this command is latched in, the next message received will be flushed except for the completion code. Only the completion code will cause an interrupt. 2. If there is a partial message in the RxFIFO (still being received off of the data link), all remaining bytes of the message will be flushed, except for the completion code, when it is formed. Only the completion code will cause an interrupt. 3. If there is a complete message in the RxFIFO (message bytes and completion code), all bytes of the message will be flushed except for the completion code. 4. If there is completion code at the head of the RxFIFO, no action is taken when this command is latched in.

NOTE:
1. Flush message commands cannot be queued up. Only one dump command is ever active. A Flush mes-sage command can not be stopped after being issued. In cases 2 and 3, the RxFIFO status may be invalid during a flush message operation. It could take a few þs (or even longer if there is no completion code in the RxFIFO yet) for the status to correctly indicate the condition of the RxFIFO. In cases 1 and 2, the RxFIFO status invalid time varies with how much of the message is yet to be received, (i.e., the completion code must “leak down” to the head of the RxFIFO before the RxFIFO is “valid” again). The RxFIFO is not necessarily empty after a flush message command.

15.11.9 TxFIFO Command Load Sequences

Table 15-21 shows various load sequences for the CMD and TDATA registers. The table indicates whether the sequence is valid or invalid and how the DLCMD2 will handle each case.

Table 15-21 Command Load Sequences

Load Sequence	DLCMD2 operation
1) Write “Load as First Byte” and data byte as 16-bit write. 2) Write “Load as Transmit Data Byte” and data byte as 16-bit write. 3) Write “Load as Last Byte” and data byte as 16-bit write.	All three command and data bytes are pushed into the TxFIFO. DLCMD2 transmits message correctly. “Load as Last Byte” command and last data byte can be written with two separate 8-bit writes.
1) Write data bytes as 8-bit writes. 2) Write “Load as Last Byte” and last data byte as 16-bit write.	First data byte treated as “first byte.” “Load as Last Byte” and last data byte is pushed into the TxFIFO. DLCMD2 transmits message correctly. “Load as Last Byte” command and last data byte can be written with two separate 8-bit writes.
1) Write “Send on EOD” as 8-bit write. 2) Write data bytes as 8-bit writes. 3) Write “Load as Last Byte” and last byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD” as 8-bit write. 2) Write “Load as First Byte” and data byte as 16-bit write. 3) Write data bytes as 8-bit writes. 4) Write “Load as Last Byte” and data byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD,” “Load as First Byte,” and data byte as 16-bit write. 2) Write data bytes as 8-bit writes. 3) Write “Load as Last Byte” and last data byte as 16-bit write.	Command and data word is pushed into the TxFIFO. Data bytes are pushed into the TxFIFO. Command and data byte is pushed into the TxFIFO. DLCMD2 transmits an IFR correctly.
1) Write “Load as First Byte” as 8-bit write. 2) Write “Send on EOD” as 8-bit write. 3) Write data bytes as 8-bit writes. 4) Write “Load as Last Byte” and data byte as 16-bit write.	No IFR transmission occurs. “Send on EOD” command will be ignored and first data byte will be treated as “first byte.” If EOF is received, the DLCMD2 will transmit the bytes as a normal message.
1) Write “Send on EOD,” “Load as First Byte,” and data byte as 16-bit write. 2) Write “Send on EOD,” “Load as transmit data byte,” and data byte as 16-bit write. 3) Write “Send on EOD,” Load as Last Byte,” and data byte as 16-bit write.	No IFR transmission occurs. Data bytes other than the first byte of an IFR cannot be written with “Send on EOD.” All data bytes are ignored.

NOTE

Any load sequence other than sequences 1-5 will not result in DLCMD2 transmission activity.

15.11.10 Transmit Data Register (TDATA)

TDATA— Transmit Data Register **0x30 008D**

MSB 7	6	5	4	3	2	1	LSB 0
TDATA[7:0]							
RESET:							
0	0	0	0	0	0	0	0

Table 15-22 Transmit Data Register (TDATA) Bit Descriptions

Bit(s)	Name	Description
7:0	TDATA	Transmit Data Register — The transmit data register (TDATA[7:0]) is used to load data into the TxFIFO. Data is automatically pushed into the TxFIFO on every write to the TDATA register. See **Figure 15-19**.

Read restrictions: Reads will return 0x00.

Write restrictions: Supervisor/Unrestricted access.

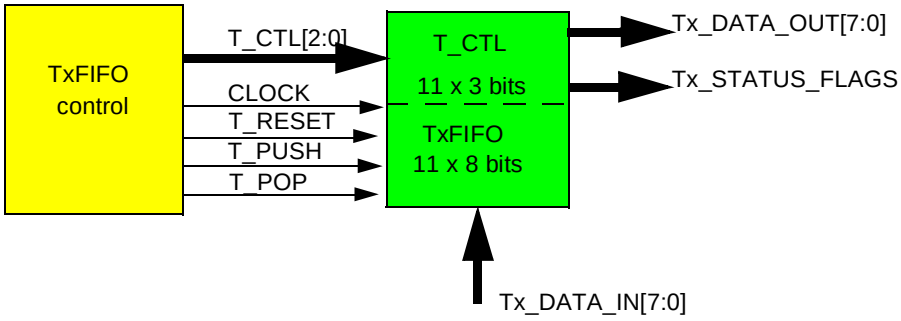


Figure 15-19 TxFIFO Block Diagram

15.11.11 Receive Status Register (STAT)

Each byte of data that the DLCMD2 module has received from the J1850 bus will be transferred to the CPU along with a status byte, which relays information on the condition of the DLCMD2 module and the data that has been received.

NOTE

Commands that are in progress may not be reflected in the status byte until sufficient time has elapsed for that command to be performed. The status must be read with each received data byte on the DLCMD2 module, either in an aligned word read, or two separate byte reads.

STAT— Receive Status Register **0x30 008E**

MSB 7	6	5	4	3	2	1	LSB 0
STAT[7:0]							
RESET:							
0	0	0	0	0	0	0	0

Table 15-23 Receive Status Register (STAT) Bit Descriptions

Bit(s)	Name	Description
7:0	STAT	Receive Status Register — The status byte can be broken in five fields. Bits 7:5 describe the status of the RxFIFO. Bit 4 indicates whether the bus is idle. Bit 3 indicates whether the bus is shorted to ground. Bit 2 indicates whether the TxFIFO is underrunning. Bit 1:0 describes the status of the TxFIFO. Refer to **Table 15-21** for RxFIFO status descriptions. Refer to **Table 15-24** for data link idle status descriptions. Refer to **Table 15-25** for transmitter shorted status descriptions. Refer to **Table 15-26** for TxFIFO underrunning status descriptions. Refer to **Table 15-27** for TxFIFO status descriptions.

Read Restrictions: Supervisor/Unrestricted access.

Write Restrictions: Supervisor/Unrestricted access, unaffected by writes.

Table 15-24 RxFIFO Status

CMD 7	CMD 6	CMD 5	Description
0	0	0	RxFIFO Invalid or Empty. 1. Read of accompanying data byte is not valid. 2. RxFIFO invalid (flush message command in progress).
0	0	1	RxFIFO Contains More Than One Byte. 1. RxFIFO has between 2-12 bytes of data at the DLCMD2 access and no completion code.
0	1	0	RxFIFO Contains a Completion Code. 1. Indicates the presence of a completion code in the RxFIFO (not at head of RxFIFO) at the DLCMD2 access. There is no other data also in the RsFIFO.
0	1	1	Receive Data Byte in Postion 13 and No Completion Code in RxFIFO. 1. Position 13 in the RxFIFO is filled and no completion code is present.
1	0	0	RxFIFO Contains Exactly One Byte. 1. RxFIFO contains exactly one data byte (not a completion code).
1	0	1	Completion Code at Head of RxFIFO, More Bytes Available 1. RxFIFO has a completion code at the head, additional bytes from an in-progress message and no other completion codes.
1	1	0	Completion Code at Head of RxFIFO, Another Complete Message Available. 1. RxFIFO has a completion code at the head and another completed message (completion code presetn) in the RxFIFO.
1	1	1	Completion Code on at Head of FIFO. 1. RxFIFO has a completion code at the head and it is the only byte in the RxFIFO.

Table 15-25 Data Link Idle Status

STAT 4	Description
0	Data Link is Busy 1. A high level on the bus has been detected for more than 8 µs, or an EOF symbol is bing timed by the receive logic. Once bit 4 is a zero because of message activity it will not become one until an EOF symbol has been received. Noise on an idle line could cause the idle bit to change state as the idle bit is un-filtered. The order of events after a transmission is: EOD —> comp. code pushed —> interupt —> EOF (idle).
1	Data Link is Idle. 1. This is the point that a transmitter may begin accessing the bus.

Table 15-26 Transmitter Shorted Status

STAT 3	Description
0	Data Link is Not Shorted to Ground
1	Data Link is Shorted to Ground[1] 1. When a hardware fault prevents the active level from being sensed after the transmitter has driven the bus for 60 µs the “data link shorted” bit is set. The only way to clear this error and try again is to reset the transmitter with the DLCMD2 Com-mand byte or a master reset of the system reset pin.

NOTE:

1. If the problem is a momentary short on the data link, it is appropriate to try reloading and transmitting the mes-sage again. If the problem is a defective line receiver then a retry could corrupt another mes-sage. The CPU must be able to handle this condition and execute a graceful recovery routine. A short to 12V will be sensed as a “data link is busy” condition. If the short to high is longer than the minimum time for a BREAK signal a completion code indicating a BREAK will be pushed into the RxFIFO after the short goes away.

Table 15-27 TxFIFO Underrunning Status

STAT 2	Description
0	The DLCMD2 TxFIFO in Not Underrunning
1	The DLCMD2 TxFIFO is Underrunning (TxFIFO is Half Empty, No Last Byte Indicated) 1. Cleared automatically following a read of the status register. 2. Actual TxFIFO underrun event (if it occurs) will be indicated in the completion code.

Table 15-28 TxFIFO Status

STAT 1	STAT 0	Description
0	0	TxFIFO is Empty
0	1	TxFIFO Contains Some Data Bytes. 1. TxFIFO contains between one and nine data bytes.

Table 15-28 TxFIFO Status

STAT 1	STAT 0	Description
1	0	TxFIFO Almost Full. 1. TxFIFO contains 10 bytes (one byte left empty).
1	1	TxFIFO Full. 1. TxFIFO contains a valid “last byte” or it contains 11 bytes and no “last byte” (see Block Mode section).

15.11.12 Receive Data Register (RDATA)

RDATA— Receive Data Register **0x30 008F**

MSB 7	6	5	4	3	2	1	LSB 0
RDATA[7:0]							
RESET:							
0	0	0	0	0	0	0	0

Table 15-29 Receive Data Register (RDATA) Bit Descriptions

Bit(s)	Name	Description
7:0	RDATA	Receive Data Register — The receive data register is used to read data from the RxFIFO. Data is popped from the RxFIFO on every read of the RDATA register. See **Figure 15-20**.

Read Restrictions: Supervisor/Unrestricted access.
Write Restrictions: Supervisor/Unrestricted access, unaffected by writes.

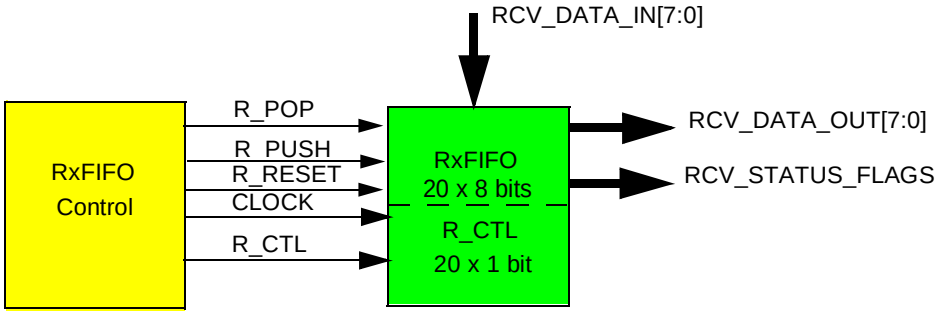


Figure 15-20 RxFIFO Block Diagram

15.11.13 Completion Code

When the message that is being received is complete, that is, after EOD has elapsed since the last transition on the bus, the DLCMD2 will prepare a completion code, which is a one byte descriptor with information about the message. If an error (CRC error, incomplete byte, bit timing error, underrun, and loss of arbitration) has occurred, the DLCMD2 will push a completion code into the RxFIFO after EOD has elapsed. If a BREAK symbol has been received, the DLCMD2 will push a completion code into the RxFIFO as soon as the bus is passive. Refer to **Figure 15-21**. This byte will be queued

and transferred to the host in the same manner that a data byte is. The status byte will flag its presence.

NOTE

Any activity on the bus (i.e., noise pulses greater than 8 µs, messages, truncated messages, bad messages, etc.) will cause a completion code to be pushed into the RxFIFO after the bus is idle for an EOD length of time. In the case of noise, the completion code will not be pushed until the noise train is over for at least an EOD period of time and there will be only one completion code for all of the noise.

	7	6	5	4	3	2	1	0
	Errors in Receive Message	RxFIFO Overrun	Transmitter Actions		IFR	IFR w/CRC	Error Code	
RESET	0	0	0	0	0	0	0	0

Figure 15-21 Completion Code Byte Bit Definitions

Refer to **Table 15-30** for receive error descriptions. Refer to **Table 15-31** for RxFIFO overrun descriptions. Refer to **Table 15-32** for transmitter action descriptions. Refer to **Table 15-27** for TxFIFO underrunning status descriptions. Refer to **Table 15-32** for IFR bit descriptions. Refer to **Table 15-33** for IFR with/without a CRC bit descriptions. Refer to **Table 15-34** for error code descriptions.

Table 15-30 Receive Error Status

Bit 7	Description
0	No Error Detected
1	Error(s) Detected 1. When set, certain errors occurred in the reception of this message (see description for bits 1 and 0). If not set, then bits 1 and 0 are also 0.

Table 15-31 RxFIFO Overrun Status

Bit 6	Description
0	No Overrun
1	Overrun 1. When set, a receiver RxFIFO overrun has occurred. It will be set as soon as the 19th data byte has been placed in the FIFO (20th position is always completion code). The host has not read out the DLCMD2 regularly enough to prevent the incoming message bytes from being lost. Previously received data bytes remain in the RxFIFO. The host must make a decision to disregard the partial message and perhaps transmit a request for retransmission of the message. If this bit is set no other bits in the completion code may be taken as valid.

Table 15-32 Transmitter Action Status

Bit 5	Bit 4	Description
0	0	Transmitter Not Involved 1. The transmitter did not attempt to contend against this message.
0	1	Transmitter Underrun 1. This message was not completed (not set if transmitter lost arbitration).
1	0	Transmitter Lost Arbitration 1.Transmitter contended against this message unsuccessfully or a RxFIFO overrun occurred (arbitration was not necessarily lost nor transmission stopped).
1	1	Transmitter Successful 1. Transmitter contended for this slot successfully (message transmitted successfully). Message originated from this node.

Table 15-33 IFR Bit Status

Bit 3	Description
0	Message not an IFR.
1	IFR 1. The preceding bytes that this completion code is associated with was an in-frame response.

Table 15-34 IFR With/Without a CRC Bit Status

Bit 2	Description
0	IFR Without CRC. 1. This in-frame response does not contain a CRC.
1	IFR With CRC. 1. The preceding byte(s) that this completion code is associated with was an in-frame response with a CRC.

Table 15-35 Error Code Status[1, 2, 3]

Bit 1	Bit 0	Description
0	0	CRC Error. 1. A CRC error was detected in the reception of this message or a receiver overrun occurred.
0	1	Incomplete Byte Received. 1. An incorrect mod 8 count was detected in the reception of this message. A complete byte was not received.
1	0	Bit Timing Error. 1. A bit timing error ocurred in this message. Will occur with a mismatch of internal clock divide relative to other nodes. The receiver immediately places this completion code after the reception of this error. May happen if a break occurs after the bus has been low for only 8 to 34 µs. In this case there will be two completion codes, the first for the bit error, and the second for the break. See **Table 15-30** through **Table 15-34** for othr bit timing errors.
1	1	Break Symbol Received. 1. A break was detected on the bus, or the bus was shorted high for more than 239 µs and has recovered. a. If the receive buffer has 19 bytes in it and a break is received, the completion code will indicate on overflow but no break. b. If the receive buffer has 20 bytes (full) and a break is received, thre will be no new completion code pushed and no indication of the break.

NOTE:
1. These error codes are used in conjunction with the error flag to report detected errors. Errors are reported based on the order of precedence. Only the highest precedence error be reported.
2. The following list is in precedence order, highest first:
   1. Bread condition occurred.
   2. Bit timing error detected.
   3. Incomplete byte (incorrect mod 8 count).
   4. CRC error detected.
3. If any of these error conditions is noted, any pending EOD response message (in-frame response) will not be sent.

15.11.14 Bus Errors

The following figures describe various bit timing/ invalid symbol errors.

NOTE

TXP signals shown are what is expected if the error had not occurred and not necessarily what is on the bus during these error conditions.

Figure 15-22 shows an active symbol received during SOF transmission that is greater than the 8–µs digital filter and less than the minimum SOF symbol time. This symbol will be flagged as an invalid symbol and the corresponding completion code will be 0x82.

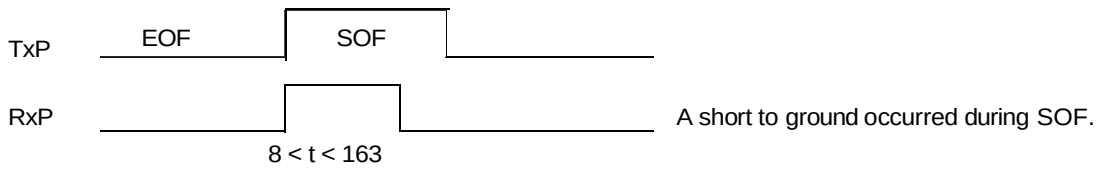


Figure 15-22 SOF Symbol Too Short

Figure 15-23 shows bit timing errors on short bits. This can be caused by noise, a bus short to ground (or V_{DD} for passive bits), clock mismatches, etc. The corresponding completion code will be 0x82. If for the second case, the bus remains high for more than 239 µs and then goes low, a completion code of 0x83 will follow.

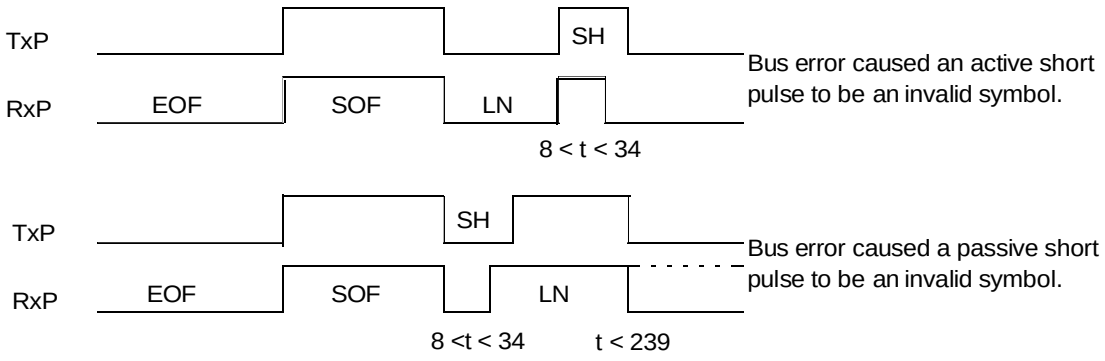


Figure 15-23 Figure 1-23 Short Bit Too Short

Figure 15-24 shows bit timing errors on long bits. This can be caused by noise, a bus short to V_{DD} (or ground for passive bits), clock mismatches, etc. For an active bit, if the bus goes low between 163 and 239 µs, the corresponding completion code will be 0x82. If the bus goes low after the minimum BREAK time, a completion code of 0x83 will follow. For a passive bit, if the bus goes high between163 and 239 µs, the completion code will be 0x82. If the bus remains high for more than 239 µs and then goes low, a completion code of 0x83 will follow.

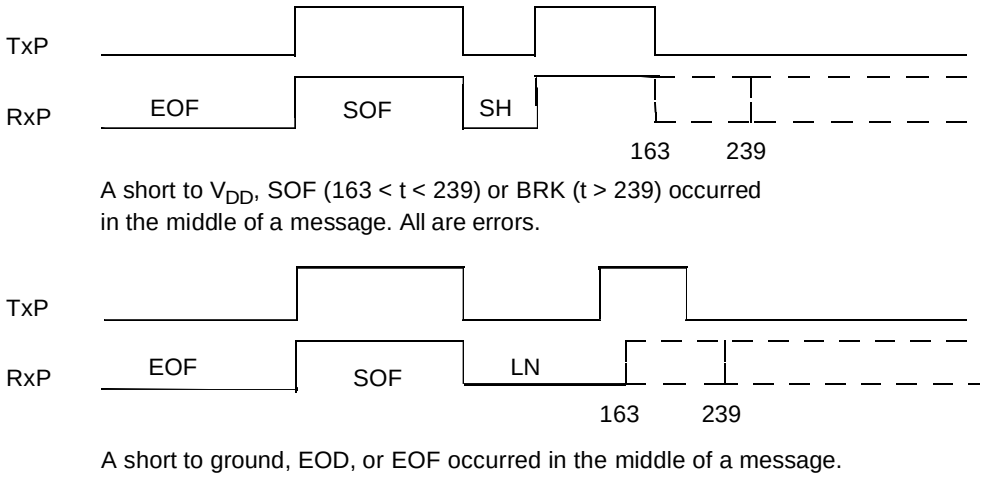


Figure 15-24 Long Bit Too Long

15.11.15 Data Link Controller Module (DLCMD2)

Figure 15-25 shows bit timing errors on an EOD and EOF. This can be caused by noise, a bus short to V_{DD}, clock mismatches, etc. In these cases, the completion code will be 0x82.

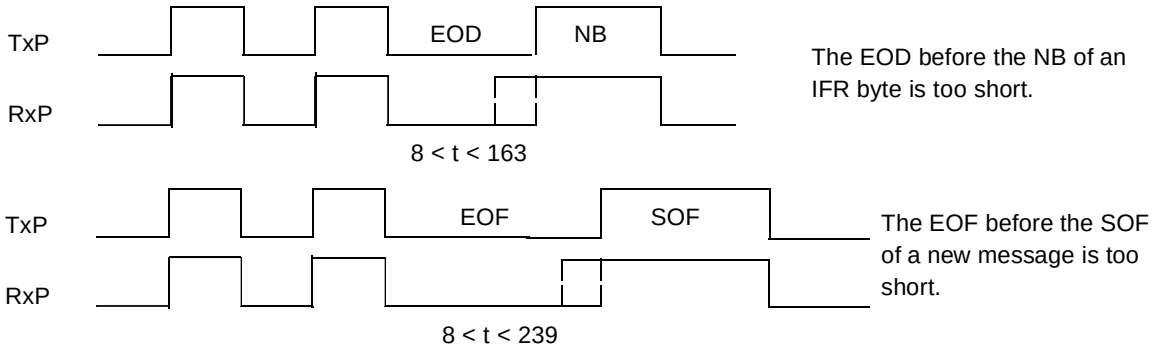


Figure 15-25 EOD and EOF Too Short

Figure 15-26 shows a bit timing error on a normalization bit. This can be caused by noise, a bus short to ground, clock mismatches, etc. The corresponding completion code will be 0x82.

Figure 15-26 Normalization Bit Too Short

Figure 15-27 shows a bit timing error on a normalization bit. This can be caused by noise, a bus short to V_{DD}, clock mismatches, etc. The corresponding completion code will be 0x82. If the bus remains high for more than 239 µs and then goes low, a completion code of 0x83 will follow.

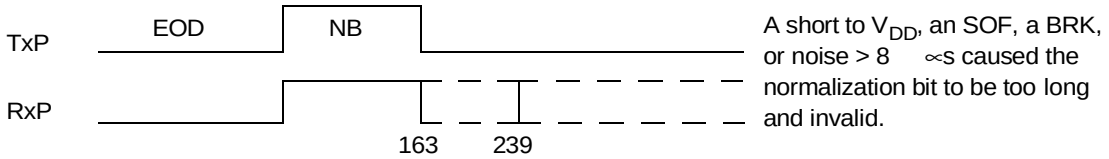


Figure 15-27 Normalization Bit Too Long

Other errors include CRC, incomplete byte, and BREAK. A CRC error occurs when the received message is corrupted by noise, delays, clock mismatches, etc. An incomplete byte error occurs when a received message ends on a non-byte boundary. This error also occurs when two extra 1's are transmitted after arbitration is lost on the last bit of a byte. Reception of a BREAK symbol is considered an error. This error occurs when the bus is held high for more than 239 µs and then goes low.

15.12 Mask Programmable Bus Error (BERR) Functionality

This DLCMD2 supports the BERR behavior according to IMB3 specification.

15.12.1 BERR_PLUG = 0

The DLCMD2 never asserts BERR signal in the IMB3.

15.12.2 BERR_PLUG = 1

The DLCMD2 will terminate the bus cycle with BERR in the following cases:

- Access to reserved 16-bit register within the DLCMD2's memory map.
- User access to supervisor-only registers when other registers in the DLCMD2's memory map are user-accessible (MCR SUPV bit is not set).
- Access to test register (TCR) when not in test mode.
- Writes to read-only registers.

15.13 Interrupt

This section describes DLCMD2 interrupt operation.

15.13.1 DLCMD2 Interrupts

In the default mode, interrupts may be requested due to one or more of four conditions becoming true. An optionally configured mode adds one more condition which will generate an interrupt. Interrupts may also be disabled.

If conditions which generate an interrupt occur while the DLCMD2 is being accessed, they will not be requested until the access is complete. Assuming interrupts are enabled, the default set of conditions that will cause the DLCMD2 module to request an interrupt are:

1. Completion code is placed in RxFIFO, DLCMD2 not accessed.
   a. An EOD has been received on the J1850 bus.
   b. bAdditional byte is received when the RxFIFO is full. This condition will cause the completion code to be pushed (in position 20 of the RxFIFO). The received byte and the rest of the message are ignored.
   c. DLCMD2 receives break. The reception of the break symbol will cause the completion code to be pushed into RxFIFO. If the break is received in the middle of DLCMD2 transmitting, it will stop transmitting, reset the transmitter, and clear the TxFIFO.
   d. DLCMD2 detects a bit-timing error. The bit-timing error will cause the completion code to be pushed into RxFIFO.
2. The RxFIFO has 12 bytes in it and the 13th byte is received, DLCMD2 not accessed. Completion code may or may not be present in RxFIFO. This differs from the 13th byte status indication which occurs only if there is no completion code in the RxFIFO.
   a. The status byte will only reflect this interrupt if there are no completion codes in the RxFIFO
   b. This interrupt generally means that the DLCMD2 is being neglected by the host.
3. A transmit operation is in progress, and there is no last byte to the message in the buffer and the buffer becomes half empty (six bytes left to transmit, five bytes in TxFIFO and the sixth byte is popped off to the transmit shift register), DLCMD2 not accessed.
4. The DLCMD2 is waking up on the rising edge of data link activity when it was previously in sleep mode.
   a. Any J1850 bus edge will wake up the DLCMD2. Will get a bit error indication if the DLCMD2 does not see at least 34 µs of the SOF.

By setting bit 15 in the ILR register in **15.11.3 Interrupt Level Register (ILR)**, one more condition capable of generating an interrupt is added:

5. A byte has been received into an empty RxFIFO, DLCMD2 not accessed.

Table 15-35 shows DLCMD2 interrupt operations. Refer to **Table 15-12** for IVR[2:0] Encoding.

Table 15-36 Interrupt Operations

Interrupt	Conditions Necessary to Enable/Re-Enable Interrupts	Conditions for Interrupt Assertion	Conditions to Clear This Interrupt
Wake-up	Enter low power mode	DLCMD2 module is in sleep mode. Positive going edge on bus > V_{ih} is sensed on bus.	Read IPR with bit 3 set and write IPR[3] “0” to clear
Transmitter half full (6 bytes)	1. “Load a byte into TxFIFO” with a fifth byte position filled, — OR — 2. Load in data bytes so that the sixth position in TxFIFO is occupied.	A transmit operation is in progress, no last byte indicated to the message in TxFIFO (*block mode*), TxFIFO becomes half empty, and transmit interrupt is enabled (*last byte not pushed into TxFIFO*)	Read IPR with bit 3 set and write IPR[3] “0” to clear
13th byte received	The only way to insure that this interruptis re-enabled is to complete empty out the Rx FIFO: 1. If the 13th byte is occupeid and the 14th isn’t a “flush byte” command will re-enable. — OR — 2. If the 13th byte is occupied and the 14th is not, and there is no completion code at the head of the RsFIFO, a “flush message” command will re-enable. — OR — 3. The 13th byte becomes unoccupied.	RxFIFO receives 13th byte, (*Completion may or may not be present in RxFIFO, refer to* **15.13.1 DLCMD2 Interrupts**)	Read IPR with bit 2 set and write IPR[2] “0” to clear.
EOD sensed on bus	Always enabled.	A completion code is placed onto the RxFIFO.	Read IPR with bit 1 set and write IPR[1] “0” to clear.
First byte received in an empty RxFIFO	1. If the first positon is filled, and the second position is empty, and a “flush byte” command is issued. — OR — 2. The first position becomes empty.	RxFIFO empty, first byte received, INTMODE in ILR must be set, and flush interrupt enabled (flush message except completion code, completion code will cauase EOD interrupt)	Read IPR with bit 0 set and write IPR[0] “0” to clear.

15.13.2 Interrupt Structure

This interrupt structure for the IMB3 supports a total of 32 interrupt levels that are time multiplexed on the IRQB[7:0] lines as seen in **Table 15-37**. In this structure, all interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. (However, each group of levels actually occur, one system clock cycle after the associated IMB3 ILBS signal is asserted.) The ILBS[1:0] signals indicate which group of eight are being driven on the

interrupt request lines. See **Table 15-37** and the **12.4 Interrupt Operation** for further details.

Table 15-37 Interrupt Levels

ILBS[1:0]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The DLCMD2 module is capable of generating one of the 32 possible interrupt levels on the IMB3. The level that the DLCMD2 will drive can be programmed into the interrupt request level bits located in the interrupt level register (ILR[2:0] bit field — bits [2:0] in the ILR register). The two bits, ILBS[1:0] in the ILR register (bits [4:3]) determine on which slot the DLCMD2 should drive its interrupt signal (one of IRQB[7:1]. **Figure 15-28** shows the timing of slot multiplexing on the IMB3.

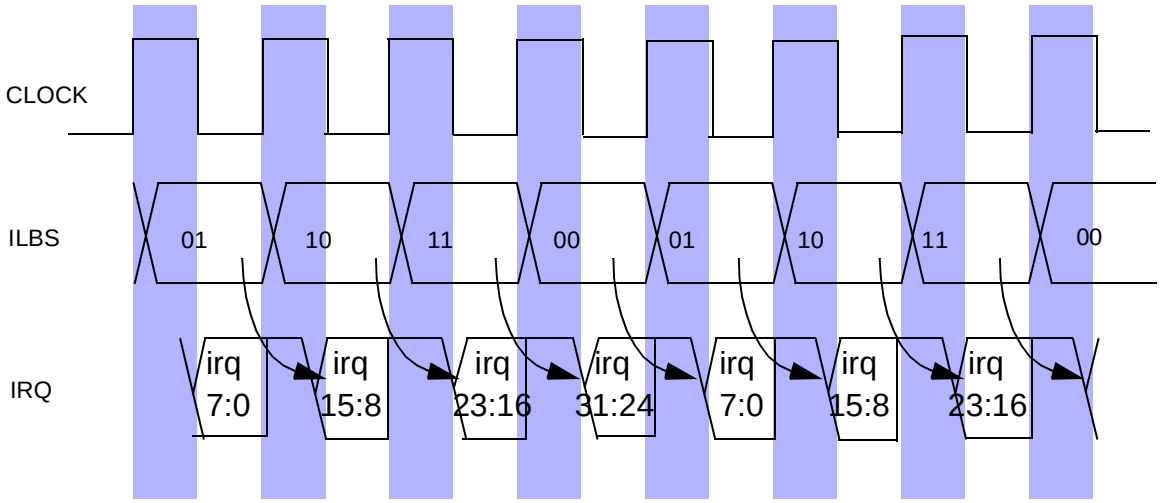


Figure 15-28 Interrupt Request Multiplex Timing

This interrupt generation scheme is shown in **Figure 15-29**. Even though the IACK cycles are not generated in this interrupt scheme, the lower three bits of IVR register are read-only bits and will be updated immediately by the DLCMD2 to indicate the source of interrupts.

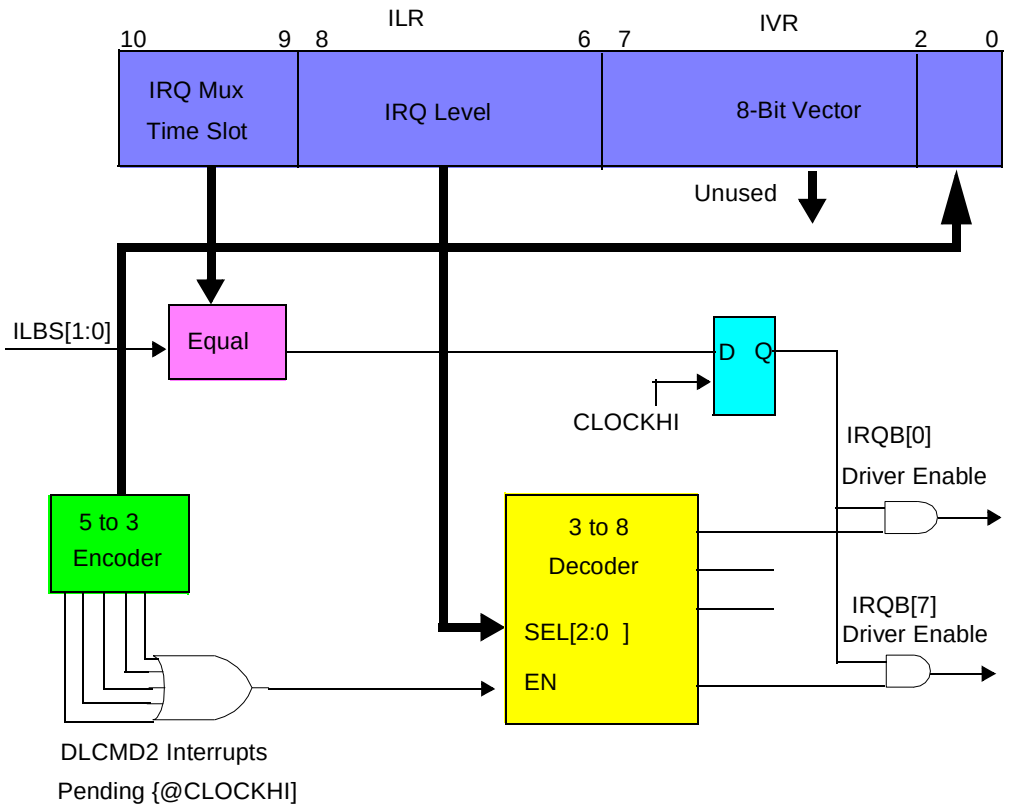


Figure 15-29 DLCMD2 Interrupt Vector Generation(IRQ_PLUG = 1)

15.14 In-Frame Response

This section describes how the DLCMD2 uses an in-frame response (IFR) in message transmission.

15.14.1 IFR Operation

The IFR may be sent by loading up a message for transmission with a “send on EOD” bit combination set in the command byte accompanying the first byte. The setting of the EOD bit combination will automatically next byte as part of, or the complete, response.

A message on the bus is determined to be ready for a response when an “end of data” (EOD) waveform has occurred on the bus. If a reply message initiates immediately following the EOD, then this will be an IFR. If no response is sent when EOD occurs, then an EOF will appear on the bus, signifying normal end of message without response. The EOF waveform is an extension of the EOD waveform. The reply will start with an active “1” or “0”, called the normalization bit, and may also have a CRC transmitted with it. The DLCMD2 is capable of arbitrating against and receiving all types of IFRs. Whether or not the received IFR contains a CRC or not is signalled through the use of the NBFS bit in the **15.11.5 Symbol Timing Control and Pre-Scaler Register**

(SCTL). IFRs are arbitrated just like any other message if more than one node wants to send a response. Single byte IFR from multiple responders (type 2) will retransmit if arbitration is lost. If arbitration is lost on the last bit of a type 2 IFR byte, two extra “1” bits will not be sent onto the bus since the IFR will retransmit. Upon beginning transmission of a single byte IFR from a single responder (type 1) or multiple byte IFR from a single responder (type 3), the DLCMD2 will enter TAR mode and not retransmit the IFR if arbitration is lost. If arbitration is lost on the last bit of a type 3 IFR byte, two extra “1” bits will be sent onto the bus.

This response message is sent only if:

1. There were no errors of any type in the original message requiring response. The only exception to this is that if an IFR is loaded after the transmitter lost arbitration to another message, the IFR will be sent at EOD.
2. IFR was loaded with a command byte that indicated either: “first byte” or “first and last” byte of a message.
3. This IFR was loaded into the DLCMD2 after the start of the message was detected, and t_{resp} before the EOD was recognized.

When the DLCMD2 receives the command byte and data byte signaling an IFR, it will start transmitting the response after EOD even if there is no last byte in the TxFIFO. The DLCMD2 will start the response by automatically transmitting an active phase “0” or a “1” depending on how the NBFS bit is programmed.

It is left to the host to remember that any messages that were in the transmit buffer at the time the IFR was loaded have been flushed, and must be reloaded by the host for transmission. This is true even if the IFR was not successfully loaded or transmitted.

Completion code bit 3 will inform the host of whether or not an IFR has occurred. If an IFR has occurred, the RxFIFO will contain the initial message, with a completion code, and the response, with its completion code. Each completion code identifies the message associated with it as a normal message, or a response. The only way that the CPU can tell that such a sequence has occurred on the bus is by reading the completion code.

If any errors occur during the reception of an IFR:

1. A completion code indicating an error occurred during reception is pushed into the RxFIFO.
2. Anything left of the IFR is not received off the bus.

If bit 3 of the completion code is set, then bit 2 will indicate whether the response had a CRC field or not. If bit 2 is set, then the response had a CRC field. This is important for the CPU to determine the meaning of the response.

A DLCMD2 may send an IFR to an IFR with CRC but no response may follow an IFR without CRC.

For the system to enter a mode that allows an IFR, some coordination of nodes is required. The node sending an IFR must select the appropriate interrupt configuration

(interrupt on first byte), and be ready to transfer a response into the DLCMD2, signalling through the command byte that this is an IFR.

15.14.2 IFR Abort Conditions

Table 15-38 shows conditions in which IFRs are aborted.he IFR was l

Table 15-38 IFR Aborted Conditions

IFR Transmission Aborted If	Actions Taken
Host underran transmitter (no “last byte” indicated in the TxFIFO at EOD, and host did not supply a last byte in time for transmission).	DLCMD2 inverts the CRC field on the truncated IFR.
IFR loses contention to another IFR.	DLCMD2 stops transmitting multiple byte IFRs, and flushes the TxFIFO immediately. DLCMD2 can be configured to auto-retry or stop transmission of single byte IFRs upon loss of arbitration.
Error detected in the received message.	DLCMD2 resets the transmitter and TxFIFO, no IFR is sent.
Error in IFR during transmission.	IFR is stopped, transmitter and TxFIFO are reset.
IFR not loaded correctly. 1. Bus idle when IFR was loaded (loaded before any message is on the bus). 2. IFR loaded less than t_{resp} before EOD is sensed on bus (loaded too late). 3. IFR not loaded with “first byte” indicated. 4. IFR bit set in command byte, but no data is loaded at all.	DLCMD2 resets the transmitter and TxFIFO, no IFR is sent.

15.14.3 IFR Types

The DLCMD2 supports types 1, 2 and 3 IFRs with arbitration and auto-retransmission. See **Figure 15-30** for a description of each type of IFR. The following sections describe how the DLCMD2 can be configured to transmit the three types of IFRs.

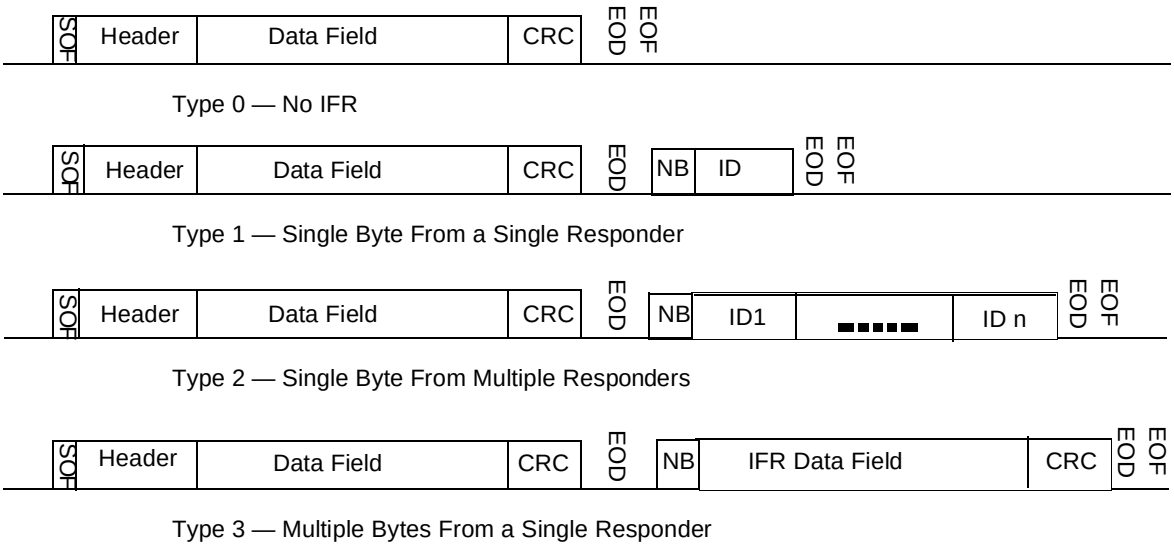


Figure 15-30 Types of IFR

15.14.3.1 Type 1 IFR

A type 1 IFR is a single byte IFR without CRC transmitted by a single responder. If arbitration is lost, the DLCMD2 will not attempt to re-transmit the IFR byte in the TxFIFO. The TxFIFO will be cleared and the IFR byte will be lost.

The DLCMD2 can be configured to transmit a type 1 IFR by writing an IFR byte into the TDATA register and a command byte of $BC indicating “send on EOD without CRC” and “load as first and last byte.”

15.14.3.2 1.Type 2 IFR

A type 2 IFR results from multiple responders transmitting a single byte IFR onto the bus. Since only one IFR will win arbitration, the other responders will re-transmit their IFRs as soon as the IFR winning arbitration finishes. The IFR byte in the TxFIFO will not be cleared until transmission is successful or an error is detected on the bus.

The DLCMD2 can be configured to transmit a type 2 IFR by writing an IFR byte into the TDATA register and a command byte of $DC indicating “send on EOD with auto re-try” and “load as first and last byte.”

15.14.3.3 Type 3 IFR

A type 3 IFR is a multiple byte IFR with or without CRC transmitted by a single responder. If arbitration is lost, the DLCMD2 will not attempt to re-transmit the IFR bytes in the TxFIFO. The TxFIFO will be cleared and the IFR bytes will be lost.

The DLCMD2 can be configured to transmit a type 3 IFR with CRC by writing the first IFR byte into the TDATA register and a command byte of $74 indicating “send on EOD with CRC” and “load as first byte.”

The DLCMD2 can also be configured to transmit a type 3 IFR without CRC by writing the first byte into the TDATA register and a command byte of $B4 indicating “send on EOD without CRC” and “load as first byte.”troller Module (DLCMD2)

15.15 System Overview

Typical usage of the DLCMD2 in a microcomputer system is shown in **Figure 15-31**.

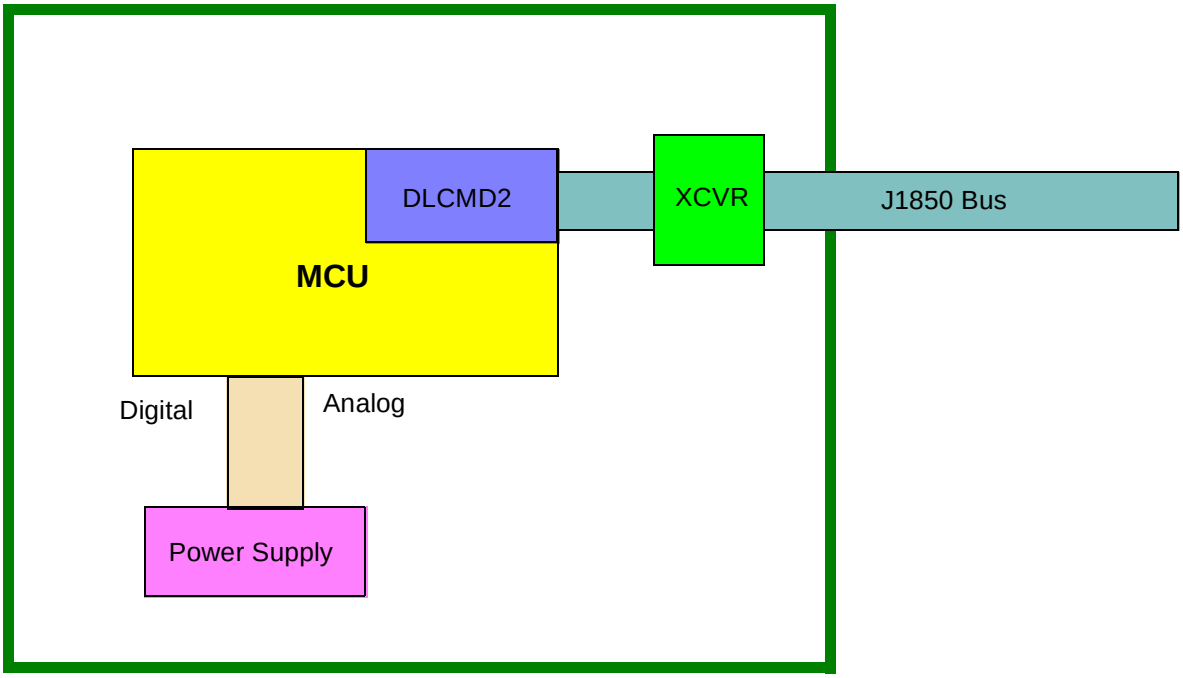


Figure 15-31 J1850 Node

15.16 Test Operation

15.16.1 Test Configuration Register (TCR)

The test configuration register controls various test modes which are used during manufacturing, and are not intended to be used in a normal application. The test configuration register can be read or written only in test mode.

When read from supervisor data space in non-test mode, the test register reads as 0x0000. When being written to supervisor data space in non-test mode, the test register ignores the write.

15.17 Module I/O Signals

15.17.1 Signal Descriptions

This section provides information on external signal functions.

15.17.2 External Connections

Figure 15-32 shows DLCMD2 external connections.

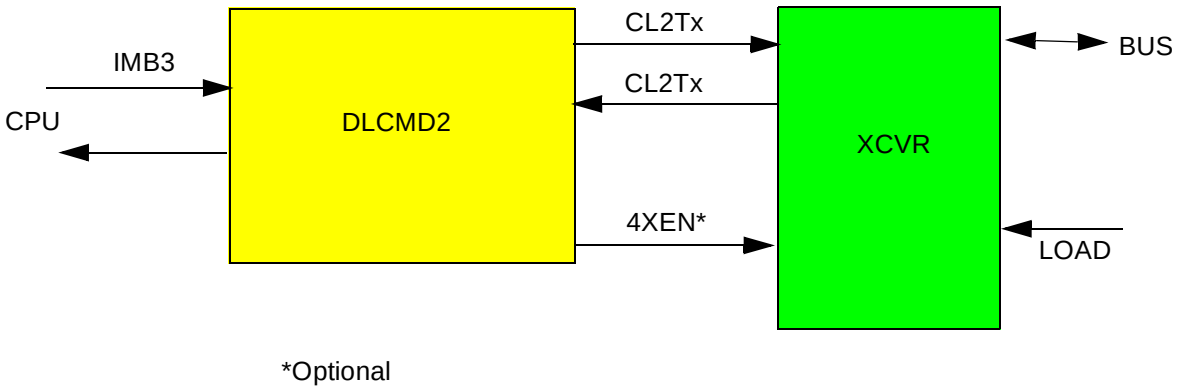


Figure 15-32 DLCMD2 External Connections

15.17.3 Signal Functions

Table 15-39 summarizes DLCMD2 external signals.

Table 15-39 Signal Names

Module Signal Name (External)	Input / Output	Description
CL2Tx	O	DLCMD2 digital output to transceiver
CL2Rx	I	DLCMD2 digital input from transceiver
4XEN[1]	O	4X transmit enable output to transceiver

NOTES:
1. This signal is not available on the MPC565 / MPC566.

15.17.3.1 CL2Tx

This pin is a logic level output. A logic “0” output drives the BUS output to 0 VDC (external pull down resistor to ground) and a logic “1” output produces a high voltage at the bus output. An internal 200 kΩ to ground in the XCVR guarantees a logic “0” input when this pin is open circuit (the bus output is tri-stated).

15.17.3.2 CL2Rx

This is a CMOS compatible input used to get receiver data to the microprocessor. There is a minimum of .1 VDC of hysteresis between the bus high and low (and vice versa) transition points.

15.17.3.3 4XEN

This is an optional pin is not available on the MPC565 / MPC566 and is used to select whether waveshaping for the J1850 output is enabled or disabled. A logic “0” shall enable waveshaping and a logic “1” shall disable waveshaping. An internal 200 kΩ to ground in the XCVR guarantees a logic “0” input when this pin is open circuit.

SECTION 16
CAN 2.0B CONTROLLER MODULE

The MPC565 / MPC566 contains three CAN 2.0B controller modules (TouCAN). Each TouCAN is a communication controller that implements the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (one Mbit/sec), short distance, priority based protocol that can run over a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The TouCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B. The third TouCAN has its pins muxed with MIOS14 GPIO pins. These pins are configured as GPIO inputs at reset and must be changed to TouCAN pins in the MIOS before enabling the TouCAN.

Each TouCAN module contains 16 message buffers, which are used for transmit and receive functions. It also contains message filters, which are used to qualify the received message IDs when comparing them to the receive buffer identifiers. **Figure 16-1** shows a block diagram of a TouCAN module.

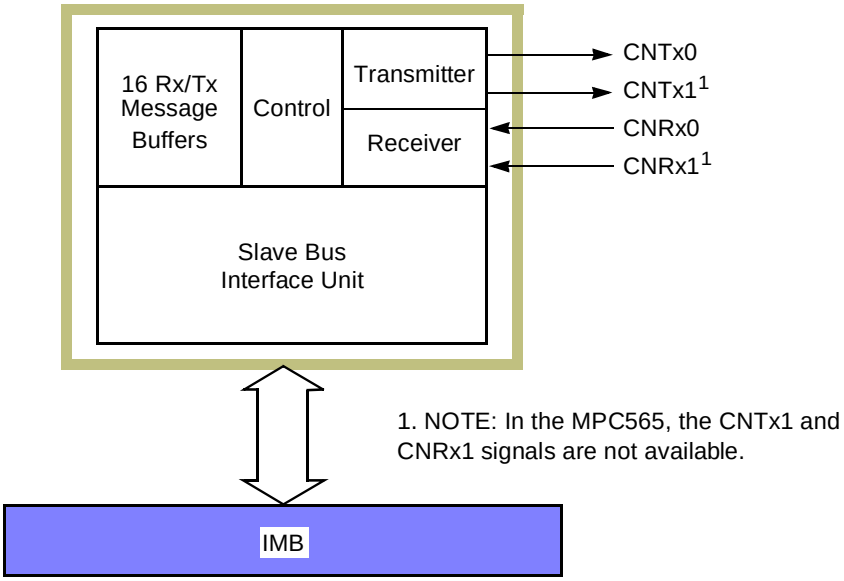


Figure 16-1 TouCAN Block Diagram

16.1 Features

Each TouCAN module provides these features:

- Full implementation of CAN protocol specification, version 2.0 A/B
  — Standard data and remote frames (up to 109 bits long)

— Extended data and remote frames (up to 127 bits long)
— Zero to eight bytes data length
— Programmable bit rate up to one Mbit/sec

- 16 Rx/Tx message buffers of 0-8 bytes data length
- Content-related addressing
- No read/write semaphores required
- Three programmable mask registers: global (for message buffers zero through 13), special for message buffer 14, and special for message buffer 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- “Time stamp”, based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity
- Outputs have open drain drivers
- Can be used to implement recommended practices SAE J1939 and SAE J2284
- Can also be used to implement popular industrial automation standards such as DeviceNet™ and Smart Distributed System
- Motorola IMB-family modular architecture

16.2 External Pins

The TouCAN module interface to the CAN bus consists of four pins: CANTx0 and CANTx1, which transmit serial data, and CANRx0 and CANRx1, which receive serial In the MPC565 / MPC566, the CNTx1 and CNRx1 signals are not available.

NOTE

In the MPC565 / MPC566, the CNTx1 and CNRx1 signals are not available.

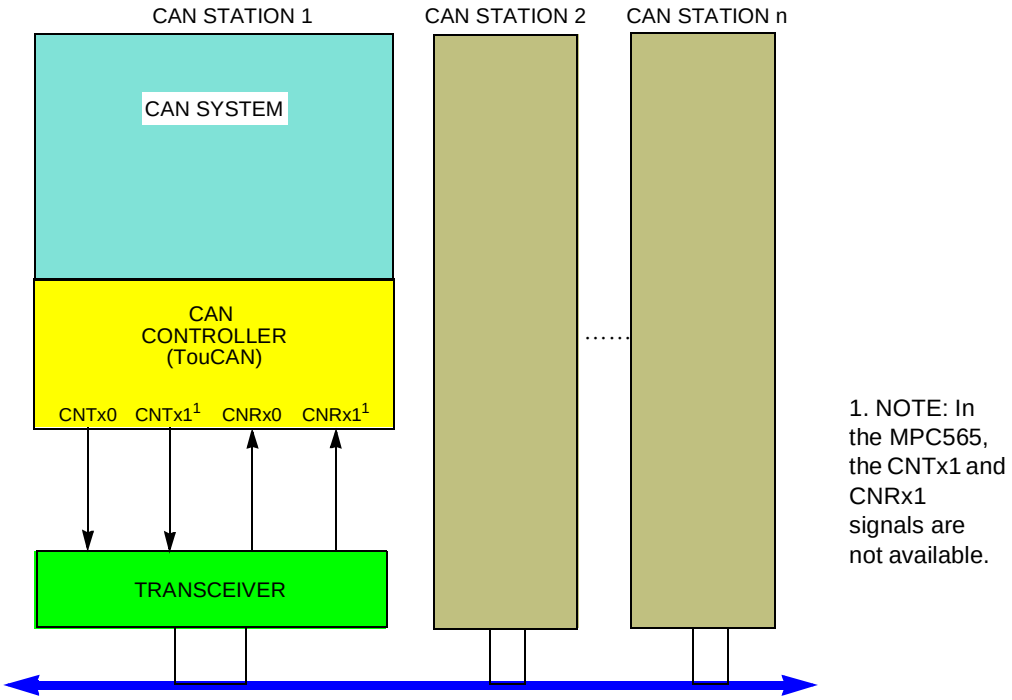


Figure 16-2 Typical CAN Network

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the TouCAN caused by a defective CAN bus or a defective CAN station.

16.3 TouCAN Architecture

The TouCAN module uses a flexible design that allows each of its 16 message buffers to be designated either a transmit (Tx) buffer or a receive (Rx) buffer. In addition, to reduce the CPU overhead required for message handling, each message buffer is assigned an interrupt flag bit to indicate that the transmission or reception completed successfully.

16.3.1 Tx/Rx Message Buffer Structure

Figure 16-3 displays the extended (29-bit) ID message buffer structure. **Figure 16-4** displays the standard (11-bit) ID message buffer structure.

	MSB 0 – 7	8 – 11	12 – 15				LSB 15	
0x0	TIME STAMP	CODE	LENGTH					CONTROL/STATUS
0x2	ID[28-18]		SRR	IDE	ID[17-15]			ID_HIGH
0x4	ID[14-0]						RTR	ID_LOW
0x6	DATA BYTE 0		DATA BYTE 1					
0x8	DATA BYTE 2		DATA BYTE 3					
0xA	DATA BYTE 4		DATA BYTE 5					
0xC	DATA BYTE 6		DATA BYTE 7					
0xE	RESERVED							

Figure 16-3 Extended ID Message Buffer Structure

	MSB 0 – 7	8 – 11	12 – 15				LSB 15	
0x0	TIME STAMP	CODE	LENGTH					CONTROL/STATUS
0x2	ID[28:18]		RTR	0	0	0	0	ID_HIGH
0x4	16-BIT TIME STAMP							ID_LOW
0x6	DATA BYTE 0		DATA BYTE 1					
0x8	DATA BYTE 2		DATA BYTE 3					
0xA	DATA BYTE 4		DATA BYTE 5					
0xC	DATA BYTE 6		DATA BYTE 7					
0xE	RESERVED							

Figure 16-4 Standard ID Message Buffer Structure

16.3.1.1 Common Fields for Extended and Standard Format Frames

Table 16-1 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 16-1 Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to **Table 16-2** and **Table 16-3**.
Rx Length	Length (in bytes) of the Rx data stored in offset 0x6 through 0xD of the buffer.This field is written by the TouCAN module, copied from the DLC (data length code) field of the received frame.
Tx Length	Length (in bytes) of the data to be transmitted, located in offset 0x6 through 0xD of the buffer. This field is written by the CPU and is used as the DLC field value. If RTR (remote transmission request) = 1, the frame is a remote frame and will be transmitted without data field, regardless of the value in Tx length.
Data	This field can store up to eight data bytes for a frame. For Rx frames, the data is stored as it is received from the bus. For Tx frames, the CPU provides the data to be transmitted within the frame.
Reserved	The CPU controls access to this word entry field (16 bits).

Table 16-2 Message Buffer Codes for Receive Buffers

Rx Code Before Rx New Frame	Description	Rx Code After Rx New Frame	Comment
0b0000	NOT ACTIVE — message buffer is not active.	—	—
0b0100	EMPTY — message buffer is active and empty.	0b0010	—
0b0010	FULL — message buffer is full.	0b0110	If a CPU read occurs before the new frame, new receive code is 0010.
0b0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0b0XY1[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0b0010	An empty buffer was filled (XY was 10).
		0b0110	A full/overrun buffer was filled (Y was 1).

NOTES:
1. For Tx message buffers, upon read, the BUSY bit should be ignored.

Table 16-3 Message Buffer Codes for Transmit Buffers

RTR	Initial Tx Code	Description	Code After Successful Transmission
x	0b1000	Message buffer not ready for transmit.	—
0	0b1100	Data frame to be transmitted once, unconditionally.	0b1000
1	0b1100	Remote frame to be transmitted once, and message buffer becomes an Rx message buffer for data frames.	0b0100
0	0b1010[1]	Data frame to be transmitted only as a response to a remote frame, always.	0b1010
0	0b1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame, always.	0b1010

NOTES:
1. When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

16.3.1.2 Fields for Extended Format Frames

Table 16-4 describes the message buffer fields used only for extended identifier format frames.

Table 16-4 Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one for Tx buffers. It will be stored as received on the CAN bus for Rx buffers.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 = Data Frame, 1 = Remote Frame.

16.3.1.3 Fields for Standard Format Frames

Table 16-5 describes the message buffer fields used only for standard identifier format frames.

Table 16-5 Standard Format Frames

Field	Description
16-bit Time Stamp	The ID LOW word, which is not needed for standard format, is used in a standard format buffer to store the 16-bit value of the free-running timer which is captured at the beginning of the identifier field of the frame on the CAN bus.
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the TouCAN.
RTR	This bit is located in the ID HIGH word of the message buffer; 0 = data frame, 1 = remote frame.
RTR/SRR Bit Treatment	If the TouCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the TouCAN transmits this bit as a zero and is receives it as a one, a bit error is indicated. If the TouCAN transmits a value and receives a matching response, a successful bit transmission is indicated.

16.3.1.4 Serial Message Buffers

To allow double buffering of messages, the TouCAN has two shadow buffers called serial message buffers. The TouCAN uses these two buffers for buffering both received messages and messages to be transmitted. Only one serial message buffer is active at a time, and its function depends upon the operation of the TouCAN at that time. At no time is there access to or visibility of these two buffers.

16.3.1.5 Message Buffer Activation/Deactivation Mechanism

Each message buffer must be activated once it is configured it for the desired operation. A buffer is activated by writing the appropriate code to the control/status word for that buffer. Once the buffer is activated, it will begin participating in the normal transmit and receive processes.

A buffer is deactivated by writing the appropriate deactivation code to the control/status word for that buffer. A buffer is typically deactivated when to reconfigure the buffer (for example to change the buffer's function from Rx to Tx or Tx to Rx). The buffer should also be deactivated before changing a receive buffer's message identifier or before loading a new message to be transmitted into a transmit buffer.

For more details on activation and deactivation of message buffers and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.1.6 Message Buffer Lock/Release/Busy Mechanism

In addition to the activation/deactivation mechanism, the TouCAN also uses a lock/release/busy mechanism to ensure data coherency during the receive process. The mechanism includes a lock status for each message buffer and uses the two serial message buffers to facilitate frame transfers within the TouCAN.

Reading the control/status word of a receive message buffer triggers the lock for that buffer. While locked, a received message cannot be transferred into that buffer from one of the serial message buffers.

If a message transfer between the message buffer and a serial message buffer is in progress when the control/status word is read, the BUSY status is indicated in the code field, and the lock is not activated.

The user can release the lock on a message buffer in one of two ways. Reading the control/status word of another message buffer locks that buffer, releasing the previously locked buffer. A global release can also be performed on any locked message buffer by reading the free-running timer.

Once a lock is released, any message transfers between a serial message buffer and a message buffer that were delayed due to that buffer being locked will take place. For more details on the message buffer locking mechanism, and the effects on message buffer operation, refer to **16.4 TouCAN Operation**.

16.3.2 Receive Mask Registers

The receive mask registers are used as acceptance masks for received frame IDs. The following masks are defined:

- A global mask, used for receive buffers 0-13
- Two separate masks for buffers 14 and 15

The value of the mask registers should not be changed during normal operation. If the mask register data is changed after the masked identifier of a received message is matched to a locked message buffer, that message will be transferred into that mes-

sage buffer once it is unlocked, regardless of whether that message's masked identifier still matches the receive buffer identifier. **Table 16-6** shows mask bit values.

Table 16-6 Receive Mask Register Bit Values

Mask Bit	Values
0	The corresponding incoming ID bit is "don't care"
1	The corresponding ID bit is checked against the incoming ID bit to see if a match exists

Table 16-7 shows mask examples for normal and extended messages. Refer to **16.7 Programmer's Model** for more information on Rx mask registers.

Table 16-7 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	1 1 1 1 1 1 1 1 0 0 0	0	—	—
MB3	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB4	0 0 0 0 0 0 1 1 1 1 1	0	—	—
MB5	0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB14	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
Rx Global Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	—
Rx Message In	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
	1 1 1 1 1 1 1 1 0 0 1	0	—	2[2]
	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	__[3]
	0 1 1 1 1 1 1 1 0 0 0	0	—	__[4]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	__[5]
Rx 14 Mask	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	—
Rx Message In	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	__[6]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

16.3.3 Bit Timing

The TouCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers one and two (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow configuration of the bit timing parameters. The prescaler divide register (PRESDIV) allows selection of the ratio used to derive the S-clock from the system clock. The time quanta clock operates at the S-clock frequency. **Table 16-8** provides examples of sys-

tem clock, CAN bit rate, and S-clock bit timing parameters. Refer to **16.7 Programmer's Model** for more information on the bit timing registers.

Table 16-8 Example System Clock, CAN Bit Rate and S-Clock Frequencies

System Clock Frequency (MHz)	CAN Bit Rate (MHz)	Possible S-Clock Frequency (MHz)	Possible Number of Time Quanta/Bit	PRESDIV Value + 1
25	1	25	25	1
20	1	10, 20	10, 20	2, 1
16	1	8, 16	8, 16	2, 1
25	0.125	1, 1.25, 2.5	8,10, 20	25, 20,10
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8
16	0.125	1, 2	8,16	16, 8

16.3.3.1 Configuring the TouCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single system clock per one time quantum, then the PSEG2 field in CANCTRL2 register must not be programmed to zero.
- If the programmed PRESDIV value results in a single system clock per one time quantum, then the information processing time (IPT) equals three time quanta; otherwise it equals two time quanta. If PSEG2 equals two, then the TouCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than 10 system clock periods per CAN bit time and the CAN bus loading is 100%, then any time the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the TouCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The TouCAN bit time must be programmed to be greater than or equal to nine system clocks, or correct operation is not guaranteed.

16.3.4 Error Counters

The TouCAN has two error counters, the transmit (Tx) error counter and the receive (Rx) error counter. Refer to **16.7 Programmer's Model** for more information on error counters.The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the TouCAN. Each counter has the following features:

- Eight-bit up/down-counter
- Increment by eight (Rx error counter also increments by one)
- Decrement by one
- Avoid decrement when equal to zero
- Rx error counter reset to a value between 119 and 127 inclusive, when the Tou-

CAN transitions from error passive to error active
- Following reset, both counters reset to zero
- Detect values for error passive, bus off and error active transitions
- Cascade usage of Tx error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read-only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the Tx error counter or Rx error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the Tx error counter or Rx error counter decrements to a value less than or equal to 127 while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the Tx error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the Tx error counter is reset to zero.
- If the TouCAN is in the bus off state, the Tx error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the Tx error counter is first reset to zero, and then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the Tx error counter is incremented by one and the internal counter is reset to zero. When the Tx error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero but does not affect the Tx error counter value.
- If only one node is operating in a system, the Tx error counter is incremented with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors never cause the TouCAN to change from the error passive state to the bus off state.
- If the Rx error counter increments to a value greater than 127, it stops incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

16.3.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp is stored in the time stamp entry of the receive message buffer at the time the message is written into

that buffer. For a message being transmitted, the time stamp entry is written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed.

16.4 TouCAN Operation

The basic operation of the TouCAN can be divided into three areas:

- Reset and initialization of the module
- Transmit message handling
- Receive message handling

Example sequences for performing each of these processes is given in the following paragraphs.

16.4.1 TouCAN Reset

The TouCAN can be reset in two ways:

- Hard reset, using one of the IMB3 reset lines
- Soft reset, using the SOFTRST bit in the module configuration register

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

16.4.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes
   a. Initialize the transmit and receive pin modes in control register 0 (CANCTRL0)
   b. Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2])
   c. Select the S-clock rate by programming the PRESDIV register
   d. Select the internal arbitration mode (LBUF bit in CANCTRL1)
2. Initialize message buffers
   a. The control/status word of all message buffers must be written either as an active or inactive message buffer.

b. All other entries in each message buffer should be initialized as required

3. Initialize mask registers for acceptance mask as needed
4. Initialize TouCAN interrupt handler
   a. Initialize the interrupt configuration register (CANICR) with a specific request level
   b. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt
5. Negate the HALT bit in the module configuration register
   a. At this point, the TouCAN attempts to synchronize with the CAN bus

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

16.4.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. This involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare or change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = 0b1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active Tx code, Tx length)

NOTE

Steps one and four are mandatory to ensure data coherency.

Once an active transmit code is written to a transmit message buffer, that buffer begins participating in an internal arbitration process as soon as the receiver senses that the CAN bus is free, or at the inter-frame space. If there are multiple messages awaiting transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

The TouCAN transmits no more than eight data bytes, even if the transmit length contains a value greater than eight.

At the end of a successful transmission, the value of the free-running timer (which was captured at the beginning of the identifier field on the CAN bus), is written into the time stamp field in the message buffer. The code field in the control/status word of the message buffer is updated and a status flag is set in the IFLAG register.

16.4.3.1 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the transmit message buffer is deactivated while a message is being transferred from it to a serial message buffer, the message is not transmitted.

If the transmit message buffer is deactivated after the message is transferred to the serial message buffer, the message is transmitted, but no interrupt is requested, and the transmit code is not updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

16.4.3.2 Reception of Transmitted Frames

The TouCAN receives a frame it has transmitted if an empty message buffer with a matching identifier exists.

16.4.4 Receive Process

During the receive process, the following events occur:

- The user configures the message buffers for reception
- The TouCAN transfers received messages from the serial message buffers to the receive message buffers with matching IDs
- The user retrieves these messages

The user should prepare or change a message buffer for frame reception by executing the following steps.

1. Write the control/status word to hold the receive buffer inactive (code = 0b0000)
2. Write the ID_HIGH and ID_LOW words
3. Write the control/status word to mark the receive message buffer as active and empty

NOTE

Steps one and three are mandatory for data coherency.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time

the TouCAN receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one. If a match is detected, the following actions occur:

1. The frame is transferred to the first (lowest entry) matching receive message buffer
2. The value of the free-running timer (captured at the beginning of the identifier field on the CAN bus) is written into the time stamp field in the message buffer
3. The ID field, data field, and Rx length field are stored
4. The code field is updated
5. The status flag is set in the IFLAG register

The user should read a received frame from its message buffer in the following order:

1. Control/status word (mandatory, as it activates the internal lock for this buffer)
2. ID (optional, since it is needed only if a mask was used)
3. Data field word(s)
4. Free-running timer (optional, as it releases the internal lock)

If the free running timer is not read, that message buffer remains locked until the read process starts for another message buffer. Only a single message buffer is locked at a time. When a received message is read, the only mandatory read operation is that of the control/status word. This ensures data coherency.

If the BUSY bit is set in the message buffer code, the CPU should defer accessing that buffer until this bit is negated. Refer to **Table 16-2**.

NOTE

The user should check the status of a message buffer by reading the status flag in the IFLAG register and not by reading the control/status word code field for that message buffer. This prevents the buffer from being locked inadvertently.

Because the received identifier field is always stored in the matching receive message buffer, the contents of the identifier field in a receive message buffer may change if one or more of the ID bits are masked.

16.4.4.1 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data must never be written into a receive message buffer. If this occurs while a message is being transferred from a serial message buffer, the control/status word will reflect a full or overrun condition, but no interrupt is requested.

16.4.4.2 Locking and Releasing Message Buffers

The lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the the lock/release/busy mechanism affects TouCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame which matches the message buffer cannot be written into this message buffer while it is locked.
2. To release a locked message buffer, the CPU either locks another message buffer by reading its control/status word or globally releases any locked message buffer by reading the free-running timer.
3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame is not immediately transferred into that message buffer, but remains in the serial message buffer. There is no indication when this occurs.
4. When a locked message buffer is released, if a frame with a matching identifier exists within the serial message buffer, then this frame is transferred to the matching message buffer.
5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication when this occurs.
6. If the control/status word of a receive message buffer is read while a frame is being transferred from a serial message buffer, the BUSY code is indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word does not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should be followed by a read of the control/status word of another buffer, or by a read of the free-running timer, to ensure that the locked buffer is unlocked.

16.4.5 Remote Frames

The remote frame is a message frame that is transmitted to request a data frame. The TouCAN can be configured to transmit a data frame automatically in response to a remote frame, or to transmit a remote frame and then wait for the responding data frame to be received.

To transmit a remote frame, a message buffer is intialized as a transmit message buffer with the RTR bit set to one. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer, with the same ID as the remote frame that was transmitted.

When the TouCAN receives a remote frame, it compares the remote frame ID to the IDs of all transmit message buffers programmed with a code of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit

in the matching transmit message buffer is set, the TouCAN transmits a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

16.4.6 Overload Frames

The TouCAN does not initiate overload frame transmissions unless it detects the following conditions on the CAN bus:

- A dominant bit in the first or second bit of intermission
- A dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames
- A dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter

16.5 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto power save mode

16.5.1 Debug Mode

Debug mode is entered when the FRZ1 bit in CANMCR is set and one of the following events occurs:

- The HALT bit in the CANMCR is set; or
- The IMB3 FREEZE line is asserted

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. Once this happens, the following events occur:

- The TouCAN stops transmitting or receiving frames
- The prescaler is disabled, thus halting all CAN bus communication
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set
- The CPU is allowed to read and write the error counter registers

After engaging one of the mechanisms to place the TouCAN in debug mode, the FRZACK bit will be set before accessing any other registers in the TouCAN; otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

Once debug mode is exited, the TouCAN resynchronizes with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

16.5.2 Low-Power Stop Mode

Before entering low-power stop mode, the TouCAN waits for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The TouCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Then the following events occur:

- The TouCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register
- The TouCAN ignores its Rx pins and drives its Tx pins as recessive
- The TouCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set

To exit low-power stop mode:

- Reset the TouCAN either by asserting one of the IMB3 reset lines or by asserting the SOFTRST bit CANMCR
- Clear the STOP bit in CANMCR
- The TouCAN module can optionally exit low-power stop mode via the self wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the TouCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the TouCAN clears the STOP bit in CANMCR and its clocks begin running.

When the TouCAN is in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event generates an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self wake mechanism is activated, the TouCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit.) It will not arbitrate for the CAN bus at this time.
- If the STOP bit is set while the TouCAN is in the bus off state, then the TouCAN enters low-power stop mode and stops counting recessive bit times. The count continues when STOP is cleared.
- To place the TouCAN in low-power stop mode with the self wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, and then wait for the TouCAN to set the STOPACK bit.
- To take the TouCAN out of low-power stop mode when the self wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, and then wait for the TouCAN to clear the STOPACK bit.

- The SELFWAKE bit should not be set after the TouCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the TouCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the TouCAN awakes from low-power stop mode via the self wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the TouCAN in low-power stop mode.
- If the TouCAN is in debug mode when the STOP bit is set, the TouCAN assumes that debug mode should be exited. As a result, it tries to synchronize with the CAN bus, and only then does it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the TouCAN does not come out of reset in low-power stop mode. The basic TouCAN initialization procedure should be executed before placing the module in low-power stop mode. (Refer to **16.4.2 TouCAN Initialization**.)
- If the TouCAN is in low-power stop mode with the self wake mechanism engaged and is operating with a single system clock per time quantum, there can be extreme cases in which TouCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. TouCAN synchronization is shifted one time quantum from the wake-up event. This shift lasts until the next recessive-to-dominant edge, which resynchronizes the TouCAN to be in conformance with the CAN protocol. The same holds true when the TouCAN is in auto power save mode and awakens on a recessive to dominant edge.

16.5.3 Auto Power Save Mode

Auto power save mode enables normal operation with optimized power savings. Once the auto power save (APS) bit in CANMCR is set, the TouCAN looks for a set of conditions in which there is no need for the clocks to be running. If these conditions are met, the TouCAN stops its clocks, thus saving power. The following conditions activate auto power save mode.

- No Rx/Tx frame in progress
- No transfer of Rx/Tx frames to and from a serial message buffer, and no Tx frame awaiting transmission in any message buffer
- No CPU access to the TouCAN module
- The TouCAN is not in debug mode, low-power stop mode, or the bus off state

While its clocks are stopped, if the TouCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The TouCAN then continues to monitor these conditions and stops or restarts its clocks accordingly.

16.6 Interrupts

The TouCAN can generate one interrupt level on the IMB. This level is programmed into the priority level bits in the interrupt configuration register (CANICR). This value determines which interrupt signal is driven onto the bus when an interrupt is requested.

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between transmit and receive interrupts for a particular buffer. Each of the buffers is assigned a bit in the IFLAG register. An IFLAG bit is set when the corresponding buffer completes a successful transmission/reception. An IFLAG bit is cleared when the CPU reads IFLAG while the associated bit is set, and then writes it back as zero (and no new event of the same type occurs between the read and the write actions).

The other three interrupt sources (bus off, error and wake up) act in the same way, and have flag bits located in the error and status register (ESTAT). The bus off and error interrupt mask bits (BOFFMSK and ERRMSK) are located in CANCTRL0, and the wake up interrupt mask bit (WAKEMSK) is located in the module configuration register. Refer to **16.7 Programmer's Model** for more information on these registers.

The TouCAN module is capable of generating one of the 32 possible interrupt levels on the IMB3. The 32 interrupt levels are time multiplexed on the IMB3 IRQ[0:7] lines. All interrupt sources place their asserted level on a time multiplexed bus during four different time slots, with eight levels communicated per slot. The ILBS[0:1] signals indicate which group of eight are being driven on the interrupt request lines.

Table 16-9 Interrupt Levels

ILBS[0:1]	Levels
00	0:7
01	8:15
10	16:23
11	24:31

The level that the TouCAN will drive onto IRQ[7:0] is programmed in the three interrupt request level (IRL) bits located in the interrupt configuration register. The two ILBS bits in the ICR register determine on which slot the TouCAN should drive its interrupt signal. Under the control of ILBS, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 16-5** displays the interrupt levels on IRQ with ILBS.

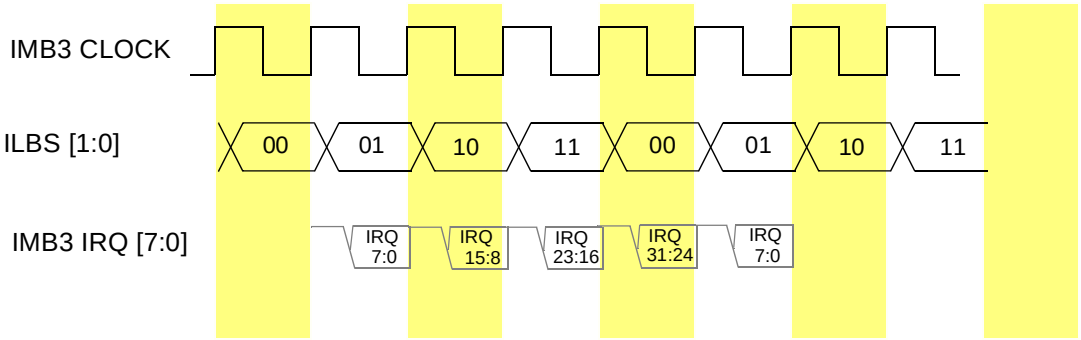


Figure 16-5 Interrupt levels on IRQ with ILBS

16.7 Programmer's Model

Table 16-10 shows the TouCAN address map. The lowercase "x" appended to each register name represents "A", "B" or "C" for the TouCAN_A, TouCAN_B or TouCAN_C module, respectively. Refer to **Figure 1-2** to locate each TouCAN module in the MPC565 / MPC566 address map.

The column labeled "Access" indicates the privilege level at which the CPU must be operating to access the register. A designation of "S" indicates that supervisor mode is required. A designation of "S/U" indicates that the register can be programmed for either supervisor mode access or unrestricted access.

The address space for each TouCAN module is split, with 128 bytes starting at the base address, and an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures. Of the lower 128 bytes, some are not used. Registers with bits marked as "reserved" should always be written as logic 0.

Typically, the TouCAN control registers are programmed during system initialization, before the TouCAN becomes synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the TouCAN module. This is done by setting the HALT bit in the TouCAN module configuration register (CANMCR). The TouCAN responds by asserting the CANMCR NOTRDY bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

NOTE

The TouCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Table 16-10 TouCAN Register Map

Access	Address	MSB 0	LSB 15
S	0x30 7080(A) 0x30 7480(B) 0x30 7880(C)	TouCAN Module Configuration Register (TCNMCR_x) See **Table 16-11** for bit descriptions.	
S	0x30 7082(A) 0x30 7482(B) 0x30 7882(C)	TouCAN Test Register (TTR_x)	
S	0x30 7084(A) 0x30 7484(B) 0x30 7884(C)	TouCAN Interrupt Register (TINTR_x)	
S/U	0x30 7086(A) 0x30 7486(B) 0x30 7886(C)	Control Register 0 (CANCTRL0_x) See **Table 16-13** and **Table 16-16** for bit descriptions.	Control Register 1 (CANCTRL1_x)
S/U	0x30 7088(A) 0x30 7488(B) 0x30 7888(C)	Control and Prescaler Divider Register (PRESDIV_x) See **Table 16-17** and **Table 16-18** for bit descriptions.	Control Register 2 (CTRL2_x)
S/U	0x30 708A(A) 0x30 748A(B) 0x30 788A(C)	Free-Running Timer Register (TIMER_x) See **Table 16-19** for bit descriptions.	
—	0x30 708C – 0x30 708E(A)) 0x30 748C – 0x30 748E(B) 0x30 788C – 0x30 788E(C)	Reserved	
S/U	0x30 7090(A) 0x30 7490(B) 0x30 7890(C)	Receive Global Mask – High (RxGMASKHI_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7092(A) 0x30 7492(B) 0x30 7892(C)	Receive Global Mask – Low (RxGMASKLO_x) See **Table 16-20** for bit descriptions.	
S/U	0x30 7094(A) 0x30 7494(B) 0x30 7894(C)	Receive Buffer 14 Mask – High (Rx14MASKHI_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7096(A) 0x30 7496(B) 0x30 7896(C)	Receive Buffer 14 Mask – Low (Rx14MASKLO_x) See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0x30 7098(A) 0x30 7498(B) 0x30 7898(C)	Receive Buffer 15 FMask – High (Rx15MASKHI_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
S/U	0x30 709A(A) 0x30 749A(B) 0x30 789A(C)	Receive Buffer 15 Mask – Low (Rx15MASKLO_x) See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	
—	0x30 709C – 0x30 709E(A) 0x30 749C– 0x30 749E(B) 0x30 789C – 0x30 789E(C)	Reserved	
S/U	0x30 70A0(A) 0x30 74A0(B) 0x30 78A0(C)	Error and Status Register (ESTAT_x) See **Table 16-21** for bit descriptions.	
S/U	0x30 70A2(A) 0x30 74A2(B) 0x30 78A2(C)	Interrupt Masks (IMASK_x) See **Table 16-24** for bit descriptions.	

Table 16-10 TouCAN Register Map (Continued)

Access	Address	MSB 0	LSB 15
S/U	0x30 70A4(A) 0x30 74A4(B) 0x30 78A4(C)	Interrupt Flags (IFLAG_x) See **Table 16-25** for bit descriptions.	
S/U	0x30 70A6(A) 0x30 74A6(B) 0x30 78A6(C)	Receive Error Counter (RxECTR_x) See **Table 16-26** for bit descriptions.	Transmit Error Counter (TxECTR_x)

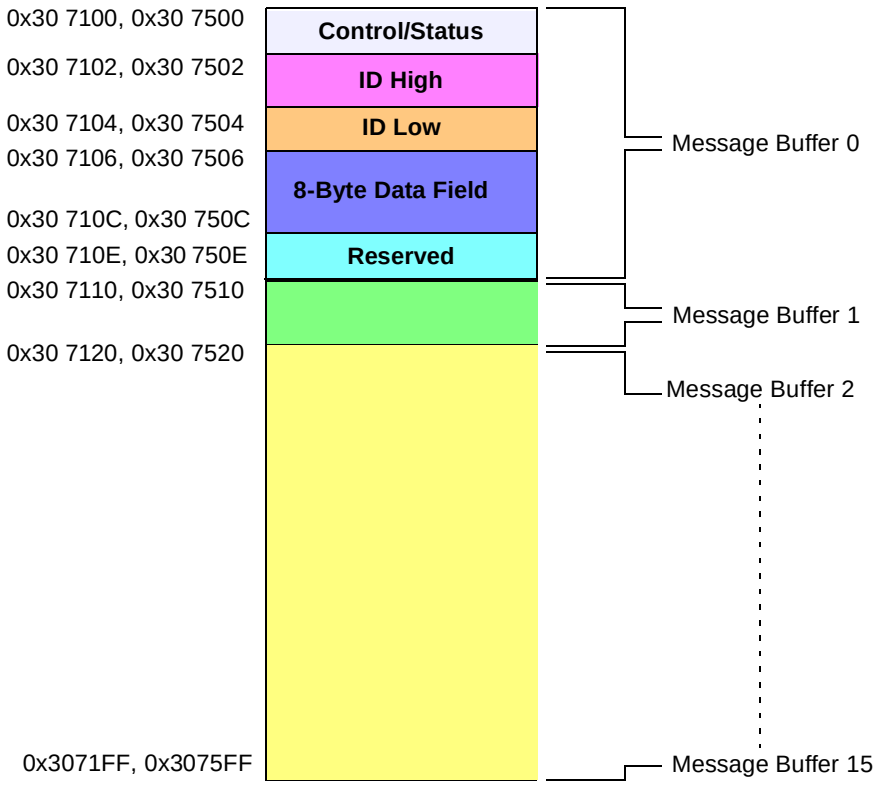


Figure 16-6 TouCAN Message Buffer Memory Map

16.7.1 TouCAN Module Configuration Register

TCNMCR — TouCAN Module Configuration Register **0x30 7080**
0x30 7480
0x30 7880

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	FRZ	NOT USED	HALT	NOT RDY	WAKE MSK	SOFT RST	FRZ ACK	SUPV	SELF WAKE	APS	STOP ACK	RESERVED			
RESET:															
0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Table 16-11 TCNMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the TouCAN, if the SELFWAKE bit is set. 0 = Enable TouCAN clocks 1 = Disable TouCAN clocks
1	FRZ	FREEZE assertion response. When FRZ = 1, the TouCAN can enter debug mode when the IMB3 FREEZE line is asserted or the HALT bit is set. Clearing this bit field causes the TouCAN to exit debug mode. Refer to **16.5.1 Debug Mode** for more information. 0 = TouCAN ignores the IMB3 FREEZE signal and the HALT bit in the module configuration register. 1 = TouCAN module enabled to enter debug mode.
2	—	Reserved
3	HALT	Halt TouCAN S-Clock. Setting the HALT bit has the same effect as assertion of the IMB3 FREEZE signal on the TouCAN without requiring that FREEZE be asserted. This bit is set to one after reset. It should be cleared after initializing the message buffers and control registers. TouCAN message buffer receive and transmit functions are inactive until this bit is cleared. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 = The TouCAN operates normally 1 = TouCAN enters debug mode if FRZ = 1
4	NOTRDY	TouCAN not ready. This bit indicates that the TouCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the TouCAN enters low-power stop mode or debug mode. It is cleared once the TouCAN exits either mode, either by synchronization to the CAN bus or by the self wake mechanism. 0 = TouCAN has exited low-power stop mode or debug mode. 1 = TouCAN is in low-power stop mode or debug mode.
5	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 = Wake up interrupt is disabled 1 = Wake up interrupt is enabled
6	SOFTRST	Soft reset. When this bit is asserted, the TouCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal TouCAN circuitry to completely reset before executing another access to CANMCR. The TouCAN clears this bit once the internal reset cycle is completed. 0 = Soft reset cycle completed 1 = Soft reset cycle initiated
7	FRZACK	TouCAN disable. When the TouCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the TouCAN has entered debug mode. When debug mode is exited, this bit is negated once the TouCAN prescaler is enabled. This is a read-only bit. 0 = The TouCAN has exited debug mode and the prescaler is enabled 1 = The TouCAN has entered debug mode, and the prescaler is disabled
8	SUPV	Supervisor/user data space. The SUPV bit places the TouCAN registers in either supervisor or user data space. 0 = Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode 1 = Registers with access controlled by the SUPV bit are restricted to supervisor mode

Table 16-11 TCNMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
9	SELFWAKE	Self wake enable. This bit allows the TouCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the TouCAN enters low-power stop mode, the TouCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the TouCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to **16.5.2 Low-Power Stop Mode** for more information on entry into and exit from low-power stop mode. 0 = Self wake disabled 1 = Self wake enabled
10	APS	Auto power save. The APS bit allows the TouCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 = Auto power save mode disabled; clocks run normally 1 = Auto power save mode enabled; clocks stop and restart as needed
11	STOPACK	Stop acknowledge. When the TouCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the TouCAN has entered low-power stop mode. When the TouCAN exits low-power stop mode, the STOPACK bit is cleared once the TouCAN's clocks are running. 0 = The TouCAN is not in low-power stop mode and its clocks are running 1 = The TouCAN has entered low-power stop mode and its clocks are stopped
12:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TouCAN implementations that use hardware interrupt arbitration.

16.7.2 TouCAN Test Configuration Register

CANTCR — TouCAN Test Configuration Register **0x30 7002, 0x30 7402, 0x30 7802**

This register is used for factory test only.

16.7.3 TouCAN Interrupt Configuration Register

CANICR — TouCAN Interrupt Configuration Register **0x30 7084**
0x30 7484
0x30 7884

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RESERVED					IRL			ILBS		RESERVED					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 16-12 CANICR Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	IRL	Interrupt request level. When the TouCAN generates an interrupt request, this field determines which of the interrupt request signals is asserted.

Table 16-12 CANICR Bit Descriptions (Continued)

Bit(s)	Name	Description
8:9	ILBS	Interrupt level byte select. This field selects one of four time-multiplexed slots during which the interrupt request is asserted. The ILBS and IRL fields together select one of 32 effective interrupt levels. 00 = Levels 0 to7 01 = Levels 8 to 15 10 = Levels 16 to 23 11 = Levels 24 to 31
10:15	—	Reserved

16.7.4 Control Register 0

CANCTRL0 — Control Register 0

0x30 7086
0x30 7486
0x30 7886

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BOFF MSK	ERR MSK	RESERVED		RxMOD		TxMODE		CANCTRL1							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-13 CANCTRL0 Bit Descriptions

Bit(s)	Name	Description
0	BOFFMSK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 = Bus off interrupt disabled 1 = Bus off interrupt enabled
1	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 = Error interrupt disabled 1 = Error interrupt enabled
2:3	—	
4:5	RxMODE	Receive pin configuration control. These bits control the configuration of the CANRx0 and CANRx1 pins. Refer to the **Table 16-14**.
6:7	TxMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTx0 and CANTx1 pins. Refer to **Table 16-15**.
8:15	CANCTRL1	See **Table 16-16**.

Table 16-14 Rx MODE[1:0] Configuration

Pin	Rx1	Rx0	Receive Pin Configuration
CNRx1[1]	0	X	A logic zero on the CNRx1 pin is interpreted as a dominant bit; a logic one on the CNRx1 pin is interpreted as a recessive bit
	1	X	A logic one on the CNRx1 pin is interpreted as a dominant bit; a logic zero on the CNRx1 pin is interpreted as a recessive bit

Table 16-14 Rx MODE[1:0] Configuration

Pin	Rx1	Rx0	Receive Pin Configuration
CNRx0	X	0	A logic zero on the CNRx0 pin is interpreted as a dominant bit; a logic one on the CNRx0 pin is interpreted as a recessive bit
	X	1	A logic one on the CNRx0 pin is interpreted as a dominant bit; a logic zeroon the CNRx0 pin is interpreted as a recessive bit

NOTES:
1. The CNRx1 signal is not available on the MPC565.

Table 16-15 Transmit Pin Configuration

TxMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CNTx0 = 0, CNTx1 = 1 is a dominant level)[2]
01	Full CMOS; negative polarity (CNTx0 = 1, CNTx1 = 0 is a dominant level)
1X	Open drain[3]; positive polarity

NOTES:
1. Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.
2. The CNRx1 signal is not available on the MPC565.
3. Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CNTx0 and CNTx1 pins are disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both Tx mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

16.7.5 Control Register 1

CANCTRL1 — Control Register 1 **0x30 7086 0x30 7486 0x30 7886**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CANCTRL0								SAMP	Reserved	TSYNC	LBUF	0D	PROPSE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-16 CANCTRL1 Bit Descriptions

Bit(s)	Name	Description
0:7	CANCTRL0	See **Table 16-13**
8	SAMP	Sampling mode. The SAMP bit determines whether the TouCAN module will sample each received bit one time or three times to determine its value. 0 = One sample, taken at the end of phase buffer segment one, is used to determine the value of the received bit. 1 = Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock.
9	—	Reserved
10	TSYNC	Timer synchronize mode. The TSYNC bit enables the mechanism that resets the free-running timer each time a message is received in message buffer zero. This feature provides the means to synchronize multiple TouCAN stations with a special “SYNC” message (global network time). 0 = Timer synchronization disabled. 1 = Timer synchronization enabled. Note: there can be a bit clock skew of four to five counts between different TouCAN modules that are using this feature on the same network.
11	LBUF	Lowest buffer transmitted first. The LBUF bit defines the transmit-first scheme. 0 = Message buffer with lowest ID is transmitted first. 1 = Lowest numbered buffer is transmitted first.
12	—	Reserved
13:15	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are zero to seven. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

16.7.6 Prescaler Divide Register

PRESDIV — Prescaler Divide Register **0x30 7088**
0x30 7488
0x30 7888

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								CANCTRL2							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-17 PRESDIV Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the system clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $$\text{S-clock} = \frac{f_{SYS}}{\text{PRESDIV} + 1}$$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the system clock. The valid programmed values are 0 through 255.
8:15	CANCTRL2	See **Table 16-18**.

16.7.7 Control Register 2

CANCTRL2 — Control Register 2

0x30 7088
0x30 7488
0x30 7888

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-18 CANCTRL2 Bit Descriptions

Bit(s)	Name	Description
0:7	PRESDIV	See **Table 16-17**.
8:9	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are zero through three. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
10:12	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment one in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
13:15	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment two in the bit time. The valid programmed values are zero through seven. The length of phase buffer segment two is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

16.7.8 Free Running Timer

TIMER — Free Running Timer Register **0x30 708A**
0x30 748A
0x30 788A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
TIMER															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-19 TIMER Bit Descriptions

Bit(s)	Name	Description
0:15	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the TouCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

16.7.9 Receive Global Mask Registers

RxGMSKHI — Receive Global Mask Register High **0x30 7090, 0x30 7490 0x30 7890**
RxGMSKLO — Receive Global Mask Register Low **0x30 7092, 0x30 7492,0x30 7892**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 16-20 RxGMSKHI, RxGMSKLO Bit Descriptions

Bit(s)	Name	Description
0:31	MIDx	The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14 and 15, which have their own specific mask registers. Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and zero) are always zero, regardless of any write to these bits. The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

16.7.10 Receive Buffer 14 Mask Registers

Rx14MSKHI — Receive Buffer 14 Mask Register High
0x30 7094, 0x30 7494, 0x30 7894

Rx14MSKLO — Receive Buffer 14 Mask Register Low
0x30 7096, 0x30 7496, 0x30 7894

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

16.7.11 Receive Buffer 15 Mask Registers

Rx15MSKHI — Receive Buffer 15 Mask Register High
0x30 7098, 0x30 7498, 0x30 7894

Rx15MSKLO — Receive Buffer 15 Mask Register Low
0x30 709A, 0x30 749A, 0x30 789A

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.

16.7.12 Error and Status Register

ESTAT — Error and Status Register **0x30 70A0**
0x30 74A0
0x30 78A0

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BITER		ACK ERR	CRC ERR	FORM ERR	STUFF ERR	Tx WARN	Rx WARN	IDLE	Tx/Rx	FCS		0	BOFF INT	ERR INT	WAKE INT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register reflects various error conditions, general status, and has the enable bits for three of the TouCAN interrupt sources. The reported error conditions are those

which have occurred since the last time the register was read. A read clears these bits to zero.

Table 16-21 ESTAT Bit Descriptions

Bit(s)	Name	Description
0:1	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. Refer to **Table 16-22**. **NOTE**: The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
2	ACKERR	Acknowledge error. The ACKERR bit indicates whether an acknowledgment has been correctly received for a transmitted message. 0 = No ACK error was detected since the last read of this register 1 = An ACK error was detected since the last read of this register
3	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 0 = No CRC error was detected since the last read of this register 1 = A CRC error was detected since the last read of this register
4	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 = No format error was detected since the last read of this register 1 = A format error was detected since the last read of this register
5	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 0 = No bit stuffing error was detected since the last read of this register 1 = A bit stuffing error was detected since the last read of this register
6	TxWARN	Transmit error status flag. The TxWARN status flag reflects the status of the TouCAN transmit error counter. 0 = Transmit error counter $<$ 96 1 = Transmit error counter $\geq$ 96
7	RxWARN	Receiver error status flag. The RxWARN status flag reflects the status of the TouCAN receive error counter. 0 = Receive error counter $<$ 96 1 = Receive error counter $\geq$ 96
8	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 = The CAN bus is not idle 1 = The CAN bus is idle
9	Tx/$\overline{\text{Rx}}$	Transmit/receive status. The Tx/Rx bit indicates when the TouCAN module is transmitting or receiving a message. Tx/Rx has no meaning when IDLE = 1. 0 = The TouCAN is receiving a message if IDLE = 0 1 = The TouCAN is transmitting a message if IDLE = 0
10:11	FCS	Fault confinement state. The FCS[1:0] field describes the state of the TouCAN. Refer to **Table 16-23**. If the SOFTRST bit in CANMCR is asserted while the TouCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the TouCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to **16.3.4 Error Counters** for more information on entry into and exit from the various fault confinement states.
12	—	Reserved
13	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the TouCAN enters the bus off state. 0 = No bus off interrupt requested 1 = When the TouCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.

Table 16-21 ESTAT Bit Descriptions (Continued)

Bit(s)	Name	Description
14	ERRINT	Error Interrupt. The ERRINT bit is used to request an interrupt when the TouCAN detects a transmit or receive error. 0 = No error interrupt request 1 = If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated. To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.
15	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the TouCAN module is in low-power stop mode. 0 = No wake interrupt requested 1 = When the TouCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

Table 16-22 Transmit Bit Error Status

BITERR[1:0]	Bit Error Status
00	No transmit bit error
01	At least one bit sent as dominant was received as recessive
10	At least one bit sent as recessive was received as dominant
11	Not used

Table 16-23 Fault Confinement State Encoding

FCS[1:0]	Bus State
00	Error active
01	Error passive
1X	Bus off

16.7.13 Interrupt Mask Register

IMASK — Interrupt Mask Register **0x30 70A2, 0x30 74A2, 0x30 78A2**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IMASKH								IMASKL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-24 IMASK Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IMASKH, IMASKL	IMASK contains two 8-bit fields, IMASKH and IMASKL. IMASK can be accessed with a 16-bit read or write, and IMASKH and IMASKL can be accessed with byte reads or writes. IMASK contains one interrupt mask bit per buffer. It allows the CPU to designate which buffers will generate interrupts after successful transmission/reception. Setting a bit in IMASK enables interrupt requests for the corresponding message buffer.

16.7.14 Interrupt Flag Register

IFLAG — Interrupt Flag Register

0x30 70A4
0x30 74A4
0x30 78A4

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IFLAGH								IFLAGL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-25 IFLAG Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	IFLAGH, IFLAGL	IFLAG contains two 8-bit fields, IFLAGH and IFLAGL. IFLAG can be accessed with a 16-bit read or write, and IFLAGH and IFLAGL can be accessed with byte reads or writes. IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a zero. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only.

16.7.15 Error Counters

RxECTR — Receive Error Counter
TxECTR — Transmit Error Counter

0x30 70A6, 0x30 74A6, 0x30 78A6

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
RXECTR								TXECTR							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16-26 RxECTR, TxECTR Bit Descriptions

Bit(s)	Name	Description
0:7, 8:15	RXECTR, TXECTR	Both counters are read only, except when the TouCAN is in test or debug mode.

SECTION 17
MODULAR I/O SYSTEM (MIOS14)

The intent of the following section is to describe in detail the configuration of the MIOS14. The MIOS14 is a MIOS derivative.

The modular I/O subsystem (MIOS) is an integral module of Motorola's CISC and RISC modular embedded controller families.

Members of these families are generally composed of a processor and slave modules that are connected together through a bus that is referred to, throughout this document, as the "peripheral bus".

A generic block diagram of such a configuration is shown on **Figure 17-1**.

17.1 Overview

The MIOS is, in itself, modular and is composed of submodules, making it easily configurable for different kinds of applications. It offers an expandable library of simple-to-use yet flexible I/O and timer functions such as:

- I/O ports
- Counters
- Input captures, output compares
- Pulse and period measurements
- Pulse width modulation
- Real-time clock

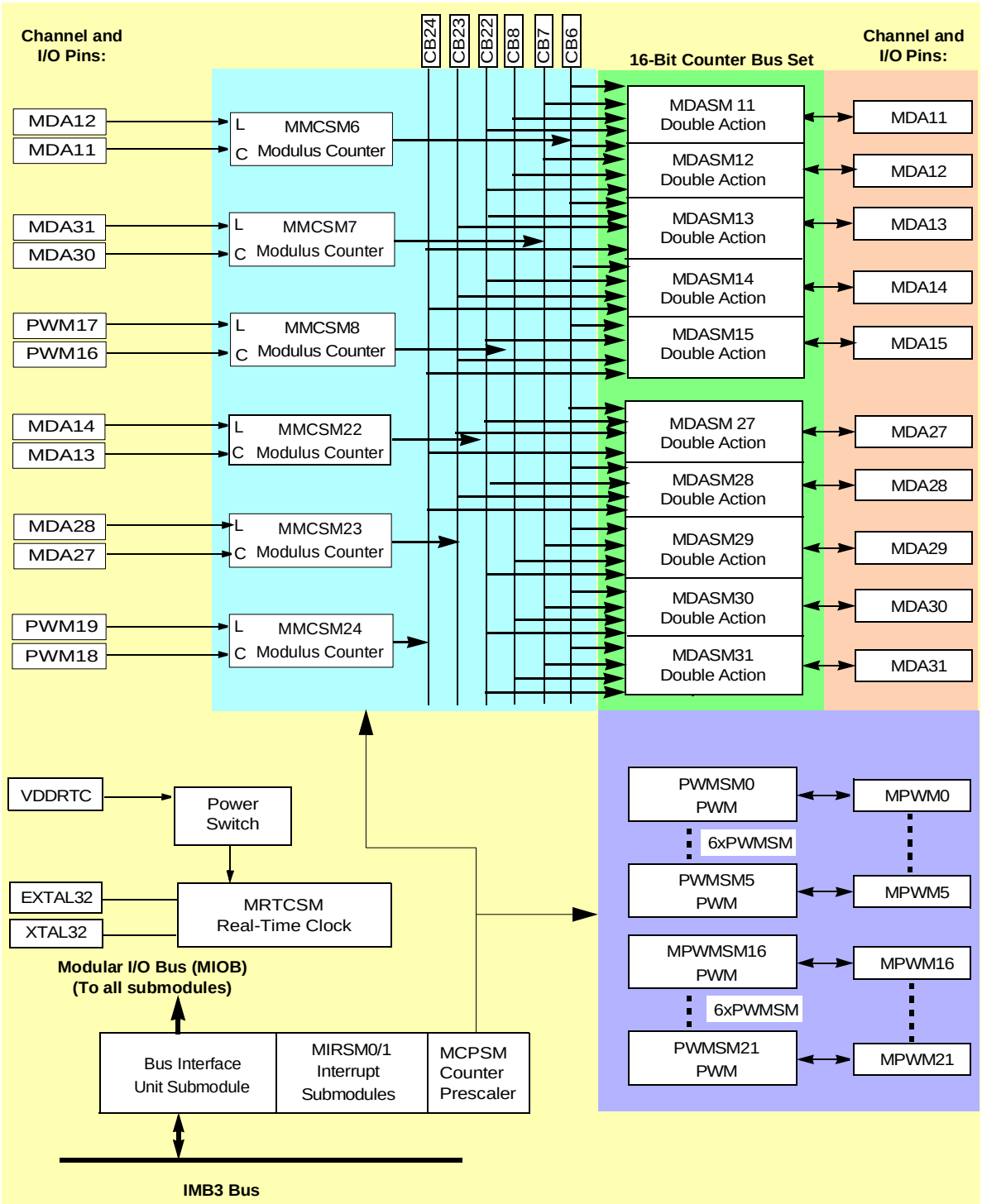


Figure 17-1 MPC565 / MPC566 MIOS14 Block Diagram

17.1.1 Documentation Conventions

The following conventions are applied throughout this document.

17.1.1.1 Terminology Definitions:

Peripheral Bus — A bus that is used to connect together the different modules, the CPU and the integration module of a given chip. This bus can be for example the IMB3.

Bus Cycle — A single transfer of data across the bus using the established protocols and timing. For any MIOS, a bus cycle consists of a single byte or a 16-bit transfer.

Master — The module that initiates a bus request and controls the bus cycle with the slave module.

Slave — The module that responds to a master's bus transaction.

Transaction — The transfer of data between two modules (a master and a slave) utilizing the internal auxiliary bus and its established protocol. A single transaction may require multiple bus cycles (for example a 32-bit transfer).

Asserted — A signal is driven to its active or true state, irrespective of that state being represented by a high or low voltage level.

Negated — A signal is driven to its inactive or false state, irrespective of that state being represented by a high or low voltage level.

Tic — A clock tic is defined as the single high or low state of a clock signal. For a 40-MHz clock, a tic has a duration of 12.5 ns.

Clock — Two tics: a full period. For a 40-MHz clock, a clock has a duration of 25 ns.

BIU — Bus interface unit: a submodule responsible for interfacing between the peripheral bus and the module.

Block — A group of four 16-bit registers.

$\mathbf{f_{SYS}}$ — Frequency of the chip clock used by the MIOS.

17.1.2 Bit and Byte Ordering

Throughout this document:

- The least significant bit of a register (LSB) is called bit-0
- The least significant byte of an MIOS14 register is at an odd address

17.1.2.1 Supported Data Sizes

The following data sizes are supported within the MIOS14:

- 32-bit words
- 16-bit words
- bytes (8 bits)

17.1.2.2 MIOS14 Port Size:

The MIOS14 port size is 16 bits.

17.1.3 Submodule Numbering Convention

By definition, a block is a group of four 16-bit registers. Each of the blocks within the MIOS14 addressing range is assigned a block number. The first block is located at the base address of the MIOS14. The blocks are numbered sequentially starting from zero.

Every submodule instantiation is also assigned a number. The number of a given submodule is the block number of the first block of this submodule.

This does not apply to the MBISM, the MCPSM and the RQSMs:

- The MBISM and the MCPSM are unique in the MIOS14 and does not need a number
- The RQSMs are numbered incrementally starting from zero

17.1.4 MIOS14 Module/Submodule Addressing

17.1.4.1 MIOS14 Base Address

The MIOS14 base address is 0x306000, and is referred to as “MIOS14 base address” in this document. The MIOS14 addressable range is 4 Kbytes.

17.1.4.2 Submodule Base Addresses

The base address of a given implemented submodule within the MIOS14 is the sum of the base address of the MIOS14 and the submodule number multiplied by eight. (This does not apply to the MBISM nor to the MCPSM and RQSMs).

17.1.4.3 Bus Error Support

A bus error signal has to be generated when accessing unimplemented/reserved 16-bit registers and privilege violations.

The MIOS14 should return a bus error under the following conditions:

- Access to unimplemented 16-bit registers within the decoded register block boundary (no bits in register)
- User access to supervisor registers
- Access to test registers when not in test mode (TSTMOD negated)
- Writes to read-only registers

17.1.4.4 Wait States

The MIOS14 does not generate any wait states.

17.1.4.5 Unimplemented Locations

All MIOS14 unimplemented locations, within the addressable range, will return a logic 0 when accessed, and will generate a BERR signal.

17.1.4.6 Unimplemented Bits

All MIOS14 registers unused bits should return a 0 when accessed.

17.2 MIOS14 Naming Convention

17.2.1 Submodule Naming Convention

An instantiated submodule is assigned a name made of its acronym followed by its submodule number.

EXAMPLE

If submodule number 18 was an MPWMSM it would be named MPWMSM18.

17.2.2 MIOS14 Pin Naming Convention

The MIOS14 input and output pin names are composed of five fields according to the following convention:

- “M”
- <submodule short_prefix>
- <submodule number>
- <pin attribute suffix> (optional)
- <bit number> (optional)

The pin prefix and suffix for the different MIOS14 submodules are as follows:

- MMCSM:
  submodule short_prefix: “MC”
  pin attribute suffix: C for the clock pin
  pin attribute suffix: L for the load pin
  For example an MMCSM placed as submodule number n would have its corresponding input clock pin named MMCnC and its input load pin named MMCnL.

- MDASM:
  submodule short_prefix: “DA”
  pin attribute suffix: none
  For example a MDASM placed as submodule number n would have its corresponding channel I/O pin named MDAn

- MPWMSM:
  submodule short_prefix: “PWM”
  pin attribute suffix: none
  For example a MPWMSM placed as submodule number n would have its corresponding channel I/O pin named MPWMn

- MPIOSM:
  submodule short_prefix: “PIO”
  pin attribute suffix: B
  For example a MPIOSM placed as submodule number n would have its corresponding I/O pins named MPIOnB0 to MPIOnB15 for bit 0 to bit 15, respectively.

In the MIOS14, some pins are multiplexed between submodules. This is shown in **Table 17-1** with using the same pin names for the inputs or outputs which are connected together.

Table 17-1 MIOS14 Configuration Description

Sub-Module Type	Block Number	Connected to:				RQSM Number	RQSM Bit Position	Base Address Offset	Pin Function	Input Pin Name	Output Pin Name	Alternate Pin Function
		CBA	CBB	CBC	CBD							
		BSL0=0 BSL1=0	BSL0=1 BSL1=0	BSL0=0 BSL1=1	BSL0=1 BSL1=1							
PWMSM	0					0	0	0x30 6000	PWM, I/O	MPWM0	MPWM0	
PWMSM	1					0	1	0x30 6008	PWM, I/O	MPWM1	MPWM1	
PWMSM	2					0	2	0x30 6010	PWM, I/O	MPWM2	MPWM2	
PWMSM	3					0	3	0x30 6018	PWM, I/O	MPWM3	MPWM3	
PWMSM	4					0	4	0x30 6020	PWM, I/O	MPWM4	MPWM4	
PWMSM	5					0	5	0x30 6028	PWM, I/O	MPWM5	MPWM5	
MMCSM	6	CB6				0	6	0x30 6030	Clock In	MDA11		
									Load In	MDA12		
MMCSM	7			CB7		0	7	0x30 6038	Clock In	MDA30		
									Load In	MDA31		
MMCSM	8				CB8	0	8	0x30 6040	Clock In	MPWM16		
									Load In	MPWM17		
Reserved	9-9											
RTCSM	10					0	10	0x30 6050	Osc. Input	EXTAL32		
									Osc. Output	XTAL32		
									Power Sup-ply	VRTC		
MDASM	11	CB6	CB22	CB7	CB8	0	11	0x30 6058	Channel I/O	MDA11	MDA11	
MDASM	12	CB6	CB22	CB7	CB8	0	12	0x30 6060	Channel I/O	MDA12	MDA12	
MDASM	13	CB6	CB22	CB23	CB24	0	13	0x30 6068	Channel I/O	MDA13	MDA13	
MDASM	14	CB6	CB22	CB23	CB24	0	14	0x30 6070	Channel I/O	MDA14	MDA14	
MDASM	15	CB6	CB22	CB23	CB24	0	15	0x30 6078	Channel I/O	MDA15	MDA15	
PWMSM	16					1	0	0x30 6080	PWM, I/O	MPWM16	MPWM16	
PWMSM	17					1	1	0x30 6088	PWM, I/O	MPWM17	MPWM17	
PWMSM	18					1	2	0x30 6090	PWM, I/O	MPWM18	MPWM18	
PWMSM	19					1	3	0x30 6098	PWM, I/O	MPWM19	MPWM19	
PWMSM	20					1	4	0x30 60A0	PWM, I/O	MPWM20	MPWM20	

Table 17-1 MIOS14 Configuration Description (Continued)

Sub-Module Type	Block Number	Connected to:				RQSM Number	RQSM Bit Position	Base Address Offset	Pin Function	Input Pin Name	Output Pin Name	Alternate Pin Function
		CBA	CBB	CBC	CBD							
		BSL0=0 BSL1=0	BSL0=1 BSL1=0	BSL0=0 BSL1=1	BSL0=1 BSL1=1							
PWMSM	21					1	5	0x30 60A8	PWM, I/O	MPWM21	MPWM21	
MMCSM	22		CB22			1	6	0x30 60B0	Clock In	MDA13		
									Load In	MDA14		
MMCSM	23			CB23		1	7	0x30 60B8	Clock In	MDA27		
									Load In	MDA28		
MMCSM	24				CB24	1	8	0x30 60C0	Clock In	MPWM18		
									Load In	MPWM19		
Reserved	25-26											
MDASM	27	CB6	CB22	CB23	CB24	1	11	0x30 60D8	Channel I/O	MDA27	MDA27	
MDASM	28	CB6	CB22	CB23	CB24	1	12	0x30 60E0	Channel I/O	MDA28	MDA28	
MDASM	29	CB6	CB22	CB7	CB8	1	13	0x30 60E8	Channel I/O	MDA29	MDA29	
MDASM	30	CB6	CB22	CB7	CB8	1	14	0x30 60F0	Channel I/O	MDA30	MDA30	
MDASM	31	CB6	CB22	CB7	CB8	1	15	0x30 60F8	Channel I/O	MDA31	MDA31	
PIOSM	32							0x30 6100	GPIO	MPIO32B0	MPIO32B0	VF[0]
									GPIO	MPIO32B1	MPIO32B1	VF[1]
									GPIO	MPIO32B2	MPIO32B2	VF[2]
									GPIO	MPIO32B3	MPIO32B3	VFLS[0]
									GPIO	MPIO32B4	MPIO32B4	VFLS[1]
									GPIO	MPIO32B5	MPIO32B5	MPWM4
									GPIO	MPIO32B6	MPIO32B6	MPWM5
									GPIO	MPIO32B7	MPIO32B7	MDO[7]
									GPIO	MPIO32B8	MPIO32B8	MDO[6]
									GPIO	MPIO32B9	MPIO32B9	MDO[5]
									GPIO	MPIO32B10	MPIO32B10	MDO[4]
									GPIO	MPIO32B11	MPIO32B11	MPWM20
									GPIO	MPIO32B12	MPIO32B12	MPWM21
									GPIO	MPIO32B13	MPIO32B13	C_CNTX0
									GPIO	MPIO32B14	MPIO32B14	C_CNRX0
									GPIO	MPIO32B15	MPIO32B15	32KCLKOUT
Reserved	33-255											
MBISM	256							0x30 6800				
Reserved	257-257											
MCPSM	258							0x30 6810				
Reserved	259-383											
MIRSM0	384-391							0x30 6C00				
MIRSM1	392-399							0x30 6C40				
Reserved	400-511											

17.3 MIOS14 Functional Overview

This section gives an overview of the MIOS14, a description of the timing function concepts, an overview description of the MIOS14 submodules and examples of how to implement functions with the MIOS14.

17.3.1 MIOS14 Configuration

The MIOS14 is composed of the following submodules:

- One MIOS bus interface submodule (MBISM)
- One MIOS counter prescaler submodule (MCPSM)
- Six MIOS modulus counter submodules (MMCSM)
- 10 MIOS double action submodules (MDASM)
- 12 MIOS pulse-width modulation submodules (MPWMSM)
- One MIOS 16-bit parallel port I/O submodule (MPIOSM)
- Two request submodules (RQSM)
- One MIOS real time clock submodule (MRTCSM)

Figure 17-1 shows the block diagram of the MIOS14 configuration.

17.4 MIOS14 Feature List

The basic features of the MIOS14 are as follows:

- Modular architecture at the silicon implementation level
- Disable capability in each submodule to allow power saving when its function is not needed
- Six 16-bit counter bus to allow action submodules to use counter data
- When not used for timing functions, every channel pin can be used as a port pin: I/O, output only or input only, depending on the channel function.
- Submodules' pin status bits reflect the status of the pin
- MIOS counter prescaler submodule (MCPSM):
  - Centralized counter clock generator
  - Programmable 4-bit modulus down-counter
  - Wide range of possible division ratios: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16
  - Count inhibit under software control
- MIOS modulus counter submodule (MMCSM):
  - Programmable 16-bit modulus up-counter with built-in programmable 8-bit prescaler clocked by MCPSM output.

- Maximum increment frequency of the counter:
  - Clocked by the internal MCPSM output: f_{SYS} / 2
  - Clocked by the external pin: f_{SYS} / 4
- Flag setting and possible interrupt generation on overflow of the up-counter
- Time counter on internal clock with interrupt capability after a pre-determined time
- Optional pin usable as an external event counter (pulse accumulator) with overflow and interrupt capability after a pre-determined number of external events.
- Usable as a regular free-running up-counter
- Capable of driving a dedicated 16-bit counter bus to provide timing information to action submodules (the value driven is the contents of the 16-bit up-counter register)
- Optional pin to externally force a load to the counter with modulus value

• MIOS double action submodule (MDASM):
- Versatile 16-bit dual action unit allowing two events to occur before software intervention is required
- Six software selectable modes allowing the MDASM to perform pulse width and period measurements, PWM generation, single input capture and output compare operations as well as port functions
- Software selection of one of the six possible 16-bit counter buses used for timing operations
- Flag setting and possible interrupt generation after MDASM action completion
- Software selection of output pulse polarity
- Software selection of totem-pole or open-drain output
- Software readable output pin status
- Possible use of pin as I/O port when MDASM function is not needed

• MIOS pulse width modulation submodule (MPWMSM):
- Output pulse width modulated (PWM) signal generation with no software involvement
- Built-in 8-bit programmable prescaler clocked by the MCPSM

- PWM period and pulse width values provided by software:
  - Double-buffered for glitch-free period and pulse width changes
  - Two-cycle minimum output period/pulse-width increment (50 ns @ 40 MHz)
  - 50% duty-cycle output maximum frequency: 10 MHz
  - Up to 16 bits output pulse width resolution
  - Wide range of periods:
    - 16 bits of resolution: period range from 3.27 ms (with 50-ns steps) to 6.71 s (with 102.4 µs steps)
    - Eight bits of resolution: period range from 12.8 µs (with 50 ns steps) to 26.2 ms (with 102.4-µs steps)
  - Wide range of frequencies:
    - Maximum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-2 prescaler selection: 305 Hz (3.27 ms)
    - Minimum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-4096 prescaler selection: 0.15 Hz (6.7 s)
    - Maximum output frequency at f_{SYS} = 40 MHz with eight bits of resolution and divide-by-2 prescaler selection: 78125 Hz (12.8 µs)
    - Minimum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-4096 prescaler selection: 38.14 Hz (8.2 ms)
- Programmable duty cycle from 0% to 100%
- Possible interrupt generation after every period
- Software selectable output pulse polarity
- Software readable output pin status
- Possible use of pin as I/O port when PWM function is not needed

- MIOS 16-bit parallel port I/O submodule (MPIOSM):
  - Up to 16 parallel I/O pins per MPIOSM
  - Uses four 16-bit registers in the address space, one for data and one for direction and two reserved

- Simple data direction register (DDR) concept for selection of pin direction

- MIOS real time clock submodule (MRTCSM):
  - Programmable 47-bit free-running ripple counter for minimum power consumption split into a 15-bit prescaler and a 32-bit second counter
  - Buffering of the 47-bit free-running counter to guaranty 32-bit and 47-bit coherent accesses on the 16-bit MIOB bus
  - Possibility of suppressing 15-bit prescaler update
  - Software shutdown of the dedicated low power oscillator to maintain battery shelf life
  - Flag setting and possible interrupt generation according to eight software selectable rates
  - Automatic hardware power supply selection through dedicated power switch
  - Customized to use low cost standard 32.768-KHz crystal for internal clocking of the 32-bit counter at 1Hz
  - Software accessible precision of 2^{-15} second and unique time indication in seconds over a span of 136 years (2^{32} seconds)
  - Eliminates risk of time inaccuracy due to interrupt overruns appearing in systems with software accumulated time

17.5 MIOS14 Pin Count

The MIOS14 requires 40 pins. The usage of these pins is shown in **Figure 17-2**.

NOTE

Three “non-I/O” pins are dedicated to the real-time clock standby power supply and oscillator.

17.6 Submodule Overview

17.6.1 MIOS Bus Interface Submodule (MBISM)

The MIOS bus interface submodule (MBISM) allows all the MIOS14 submodules to communicate to the peripheral bus master via the modular I/O bus (MIOB).

17.6.2 MIOS Modulus Counter Submodule (MMCSM)

The MIOS modulus counter submodule (MMCSM) contains a programmable 8-bit prescaler, a 16-bit up counter associated with a modulus latch that gives the flexibility of recycling the counter at a count other than 64-Kbyte clock cycles, a clock source

selector, software writable control registers and software readable status bits. When the 16-bit up counter overflows an interrupt flag is generated.

The state of the modulus latch is transferred to the counter under the following three conditions:

1. When an overflow occurs
2. When an appropriate transition occurs on the external load pin
3. When the program writes to the counter register. In this case, the value is first written into the modulus latch and immediately transferred to the counter.

Software can also write a value to the modulus latch for later loading into the counter with one of the two first criteria.

The current state of the 16-bit counter is the primary output of the MIOS modulus counter submodule. The 16-bit counter always drives a value onto a dedicated 16-bit counter bus.

A software control register selects whether the clock input to the counter is one of the prescaler output or the corresponding input pin. The polarity of the external input pin is also programmable. Clocking of the counter may also be inhibited under software control.

17.6.3 MIOS Double Action Submodule (MDASM)

The MIOS double action submodule (MDASM) provides two consecutive 16-bit input captures or two consecutive 16-bit output compare functions that can occur automatically without software involvement. The input edge detector is programmable to cause the capture function to occur on the desired edges. The output flip-flop is set by one of the output compare and is reset by the other one. In either the input capture modes or the output compare modes, an optional interrupt is available to the software. Software selection is provided to select which of the incoming 16-bit counter buses is used for the input capture or the output compare

The MDASM has six different software selectable modes:

- Disable mode
- Pulse width measurement
- Period measurement
- Input capture mode
- Single pulse generation
- Continuous pulse width generation

The MDASM has three data registers that are accessible to the software from the various modes. For some of the modes, two of the registers are cascaded together to provide double buffering. The value in one register is transferred to another register

automatically at the correct time so that the minimum pulse (measurement or generation) is just one 16-bit counter bus count.

17.6.4 MIOS Pulse Width Modulation Submodule (MPWMSM)

The purpose of the MIOS pulse width modulation submodule (MPWMSM) is to create a variable pulse width output signal at a wide range of frequencies, independent of other MIOS14 output signals. The MPWMSM includes its own 8-bit prescaler and counter, and thus does not use the MIOS14 16-bit counter buses.

The MPWMSM pulse width can vary from 0.0%t to 100.0%, with up to 16 bits of resolution. The finest output resolution is the MCU system clock time divided by two (for a f_{SYS} of 40.0 MHz, the finest output pulse width resolution is 50 ns). With the full 16 bits of resolution and the overall prescaler divide ratio varying from divide-by-2 to divide-by-4096, the period of the MPWMSM output can range from 3.2 8ms to 6. 7s (assuming a f_{SYS} of 40 MHz). By reducing the counting value, the output signal period can be reduced. The period can be as fast as 205 µs (4.882 KHz) with 12 bits of resolution, as fast as 12.8 µs (78.125 KHz) with eight bits of resolution, and as fast as 3.2 µs (312.500 KHz) with 6 bits of resolution (still assuming a f_{SYS} of 40 MHz and a first stage prescaler divide-by-2 clock selection).

17.6.5 MIOS 16-bit Parallel Port I/O Submodule (MPIOSM)

An MIOS 16-bit parallel port I/O submodule can handle up to 16 input/output pins. It contains two 16-bit registers: the data register (DR) and the data direction register (DDR). Each pin of the MPIOSM may be programmed as an input or an output under software control. The direction of the pins is determined by the state of the corresponding bits in the DDR.

17.6.6 MIOS Real-Time Clock Submodule (MRTCSM)

The purpose of the MRTCSM is to provide a real-time clock function independent of other MIOS14 submodules. This time counter is driven by a dedicated low frequency oscillator (32.768 KHz) for low power consumption. The MRTCSM contains a 15-bit prescaler and a 32-bit free-running counter, from which seconds, minutes, hours and days can be determined. The MRTCSM also has internal interrupt generation capability.

The MRTCSM may be sustained on a separate power supply, VDDRTC, for battery backup. The power switch, associated with the VDDRTC power pin, controls which supply (V_{DD} or VDDRTC) powers the MRTCSM.

17.6.7 Timing Function Examples

The versatility of the MIOS timer architecture is based on multiple counters and capture/compare channel units interconnected on 16-bit counter buses. Rather than present block diagrams of each submodule, this section includes some typical application examples to show how the submodules can be interconnected to form timing functions. The diagrams used to illustrate these examples show only the blocks utilized for that function.

To illustrate the timing range of the MIOS in different applications, many of the following paragraphs include time intervals quoted in microseconds and seconds. The assumptions used are that f_{SYS} is at 40 MHz with minimum prescaling (50-ns cycle) and with the maximum prescaling (102-µs cycle). For other f_{SYS} clock cycle rates and prescaler choices, the times mentioned in these paragraphs scale appropriately.

17.6.7.1 Single-Edge Input Capture

The MIOS single action submodule (MSASM) has an input capture register to latch the current state of a 16-bit counter bus when an external input edge is detected. The MSASM is software programmable to latch on the rising or falling edge of the input signal. The software also selects one of the 16-bit counter buses, each originating at a counter submodule. The software can also enable an interrupt to occur when the input edge is detected to notify the software that new edge capture information is available.

Figure 17-2 shows an example of an MIOS modulus counter submodule (MMCSM) for an MIOS single action submodule (MSASM) configured for input capturing. To measure the period of an incoming signal, the software reads and saves the latched value in register A for one edge, then when the next edge arrives, the software subtracts the new captured value in register A from the previously saved value to obtain the period interval. The maximum period that can be measured is the worst case software response time to a newly captured value.

The software measures the width of a pulse in a very similar way, the only difference is that after each edge, the edge detector is reprogrammed to trigger on the next opposite edge.

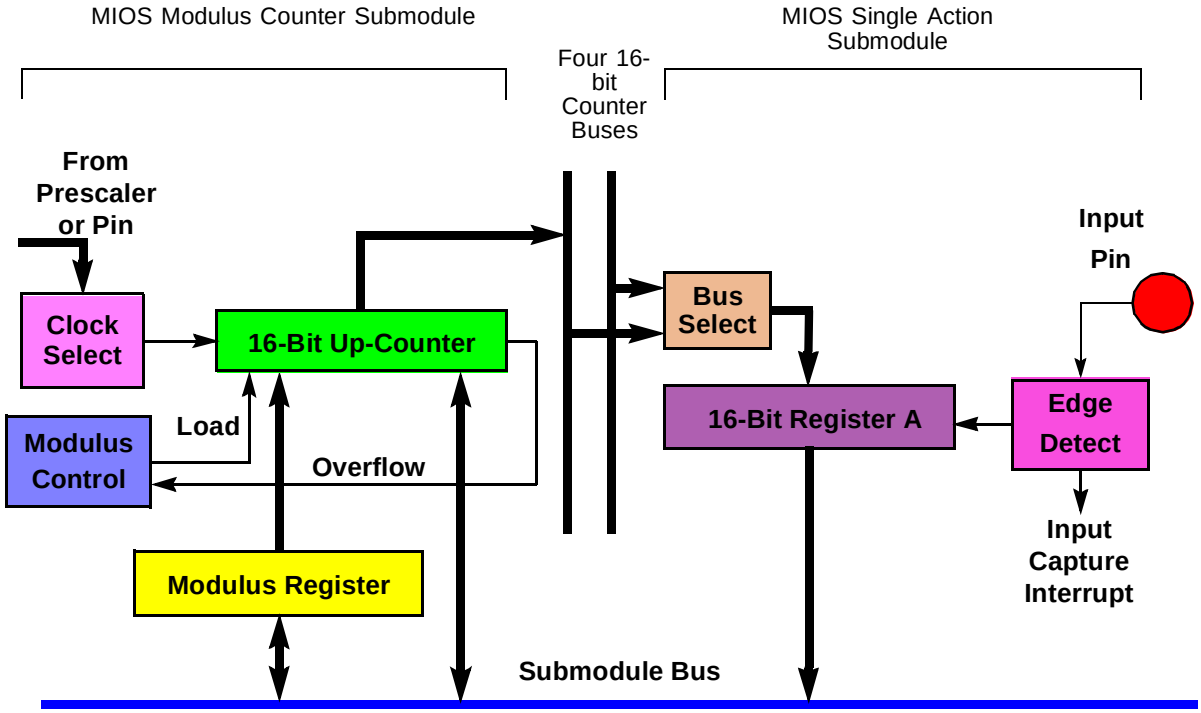


Figure 17-2 Example: Single-Edge Input Capture

17.6.7.2 Input Double-Edge Pulse Width Measurement

To measure the width of an input pulse, the MIOS double action submodule (MDASM) has two capture registers so that only one interrupt is needed after the second edge. The software can read both edge samples and subtract them to get the pulse width. The leading edge sample is double latched so that the software has the time of one full period of the input signal to read the samples to be sure that nothing is lost. Depending on the prescaler divide ratio, pulses width from 50 ns to 6.71 s can be measured.

NOTE

A software option is provided to also generate an interrupt after the first edge.

In the example shown in **Figure 17-2**, a counter submodule is used as the time-base for a MDASM configured in the input pulse width measurement mode. When the leading edge (programmed for being either rising or falling) of the input signal occurs, the state of the 16-bit counter bus is saved in register B1. When the trailing edge occurs, the 16-bit counter bus is latched into register A, and the content of register B1 is transferred to register B2. This operation leaves register B1 free for the next leading edge to occur, as soon as on the next clock cycle. When enabled, an interrupt is provided after the trailing edge, to notify the software that pulse width measurement data is available for a new pulse. After the trailing edge, the software has one cycle time of the input signal to obtain the values for each edge. When software attention is not needed for every pulse, the interrupt can be disabled — the software can at any time read registers A and B2 coherently (using a 32-bit read instruction) to get the latest edge measurements. Since the measurement resolution is 16 bits, signals with pulse duty cycles from 0.0015% to 99.9985% can be measured. The software work is less than half that needed with a timer that requires the software to read one edge and save the value, and then wait for the second edge.

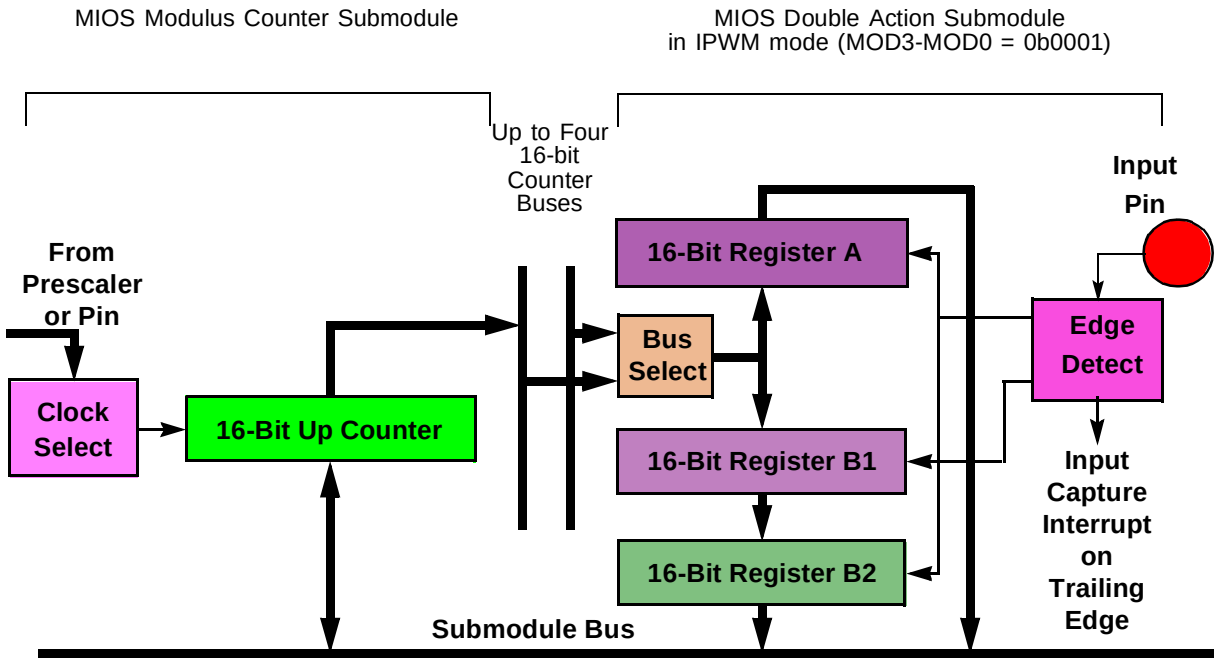


Figure 17-3 Example: Double-Edge Pulse Width Measurement

17.6.7.3 Input Double-Edge Period Measurement

Two samples are also available to the software from a MIOS double action submodule for period measurement. The software can read the previous and the current edge samples and subtract them. As with pulse width measurement, the software can be sure of not missing samples by insuring that the interrupt response time is faster than the fastest input period. Alternately, when the software is just interested in the latest period measurement, one 32-bit coherent read instruction can get both the current and the previous samples. Depending on the prescaler divide ratio, period times can be measured from 50 ns to 6.71 s.

Figure 17-3 shows an MMCSM and an MDASM combination as an example of period measurement. The software designates whether the rising or falling edge of the input signal is to be used for the measurements. When the edge is detected, the state of the 16-bit counter bus is stored in register A, and the content of register B1 is transferred into register B2. After register B2 is safely latched, the content of register A is transferred to register B1. This procedure gives the software coherent current and previous samples in registers A and B2 at all times. An interrupt is available for the cases where the software needs to be aware of each new sample.

NOTE

A software option is provided to also generate an interrupt after the first edge.

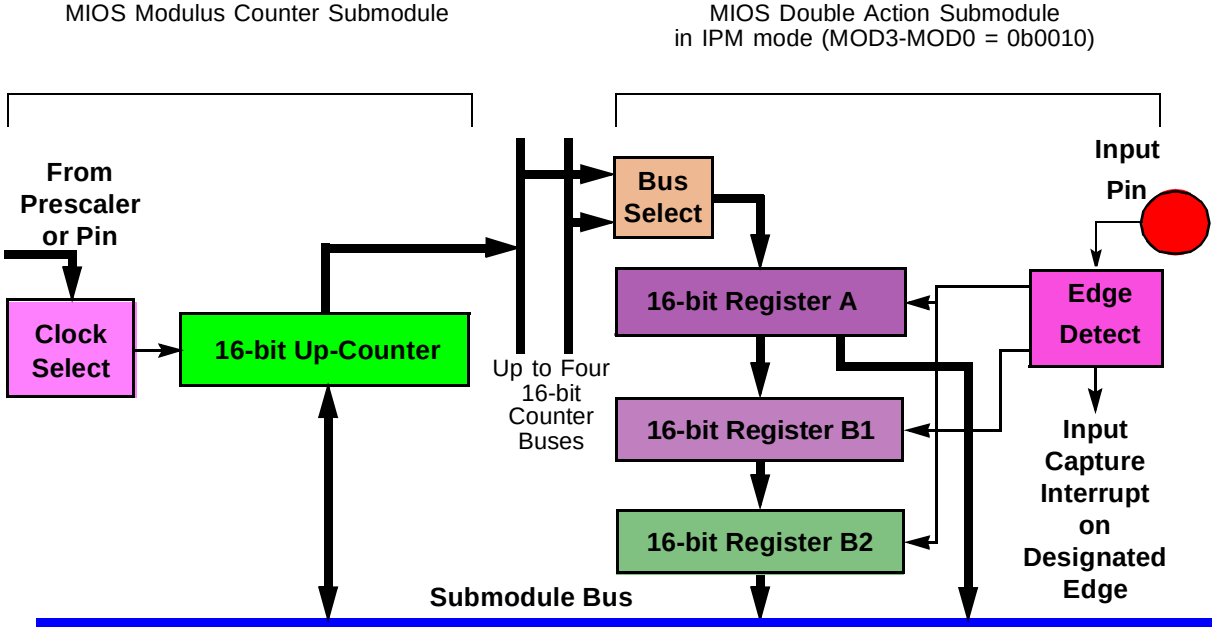


Figure 17-4 Example: Double-Edge Period Measurement

17.6.7.4 Single-Edge Output Compare

To create one output edge, the software can use an MSASM channel. The software provides a compare value in a register and the MSASM compares that value to the incrementing value seen on one of the 16-bit counter buses. When a comparison is detected, the state of the output pin is changed.

The example shown in **Figure 17-4** uses a counter submodule with one channel of an MSASM to create an output signal. The software can read the current state of the counter submodule. That, or some other criteria, is used to determine the time-base value when the output is to change. The software thus writes the compare value into register A. In the MSASM control register, the software establishes whether the output flip-flop is to toggle to the opposite state, or is to go to a high or a low level. The output compare interrupt is typically used to notify the software that the previous compare is complete and the MSASM is available for a new compare value in register A.

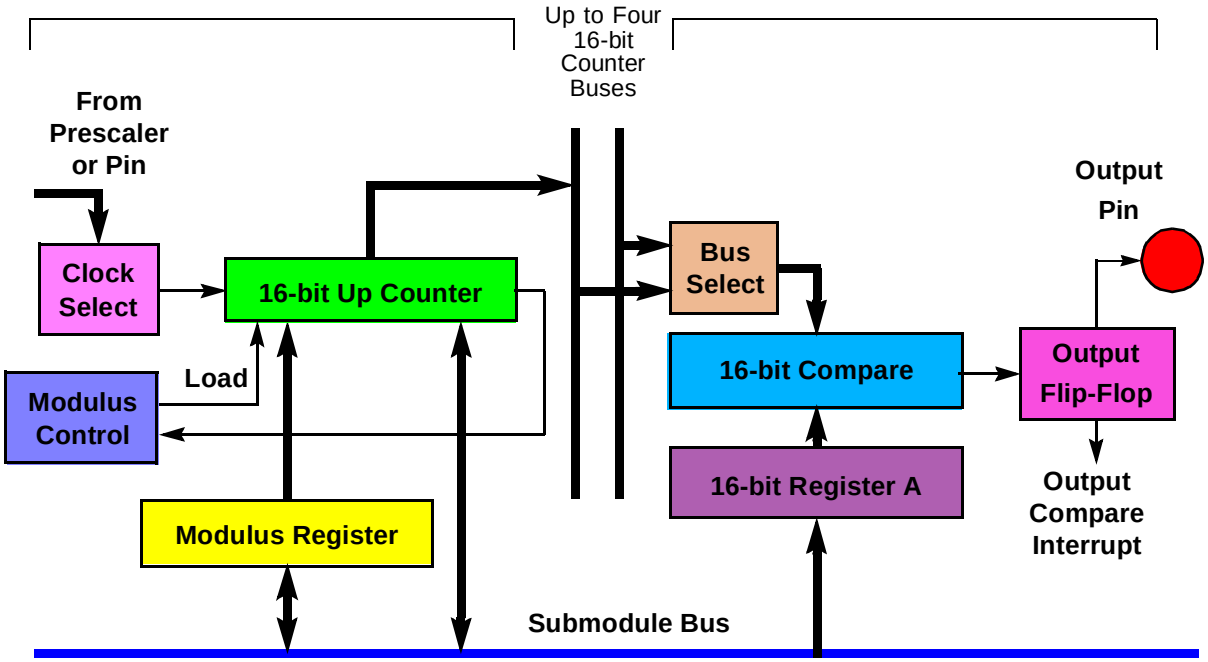


Figure 17-5 MIOS Example: Single-Edge Output Compare

17.6.7.5 Double-Edge Single Output Pulse Generation

Software can initialize the MIOS to generate both the rising and the falling edge of an output pulse. With a MDASM, pulses as narrow as 50 ns can be generated since software action is not needed between the edges. Pulses as long as 6.71 s can be generated. When an interrupt is desired, it can be selected to occur on every edge or only after the second edge.

Figure 17-5 shows how a counter submodule and a MDASM can be used to generate both edges of a single output pulse. The software puts the compare value for one edge in register A and the other one in register B2. The MDASM automatically creates both edges, and the pulse can be selected by software to be a high-going or a low-going. After the trailing edge, the MDASM stops to await further commands from the software.

NOTE

A single-edge output can be generated by writing to only one register.

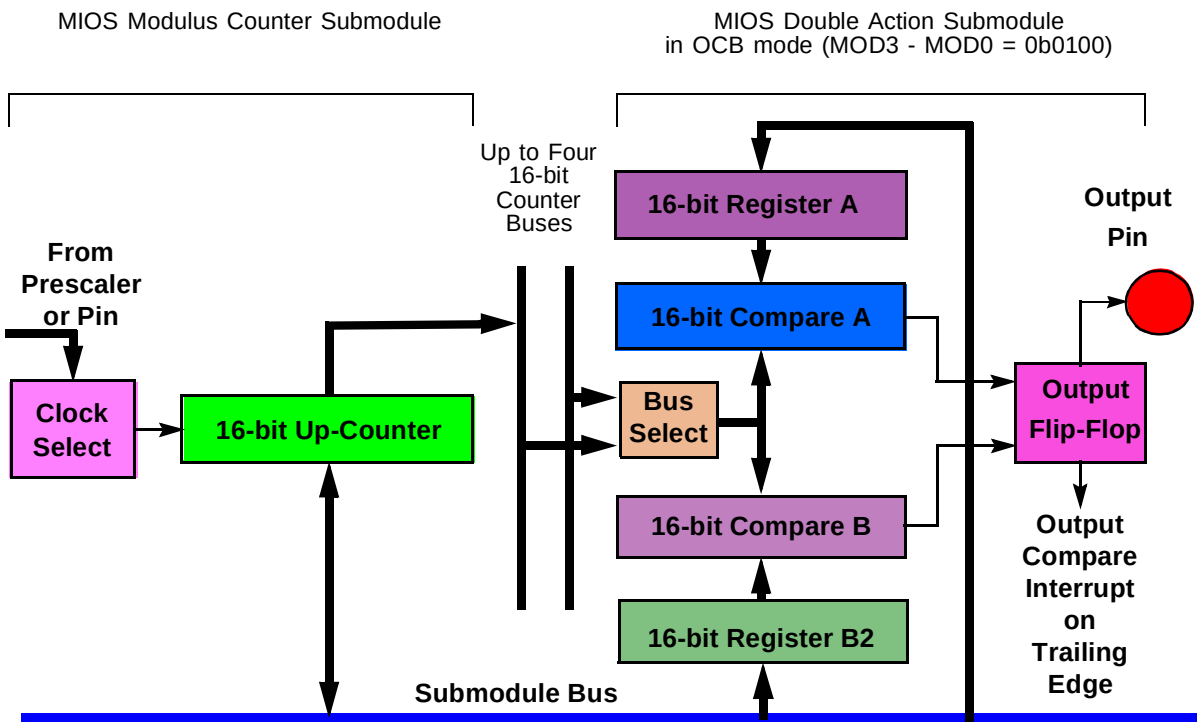


Figure 17-6 MIOS Example: Double-Edge Output Compare

17.6.7.6 Output Pulse Width Modulation With MDASM

Output waveforms can be generated with any duty cycle without software involvement. The software sets up a MDASM with the compare times for the rising and falling edges, and they are automatically repeated. The software does not need to respond to interrupts to generate continuous pulses. The period may be selected as the period of a MIOS modulus counter submodule used in free-running mode time-base, times a binary multiplier selected in the MDASM. Multiple PWM outputs can be created from multiple MDASMs and share one counter submodule, provided that the periods of all of the output signals are a binary multiple of the time-base, and that the counter submodule is operating in a free-running mode. Each MDASM has a software selectable “don’t care” on high-order bits of the time-base comparison so that the period of one output can be a binary multiple of another signal. Masking the time-base serves to multiply the period of the time-base by a binary number to form the period of the output waveform. The duty cycle can vary from one cycle to 64-Kbyte cycles. The frequency can range from 0.6 Hz to 156 KHz, though the resolution decreases at the higher frequencies to as low as seven bits. The generation of output square wave signals is of course the special case where the high and low time are equal.

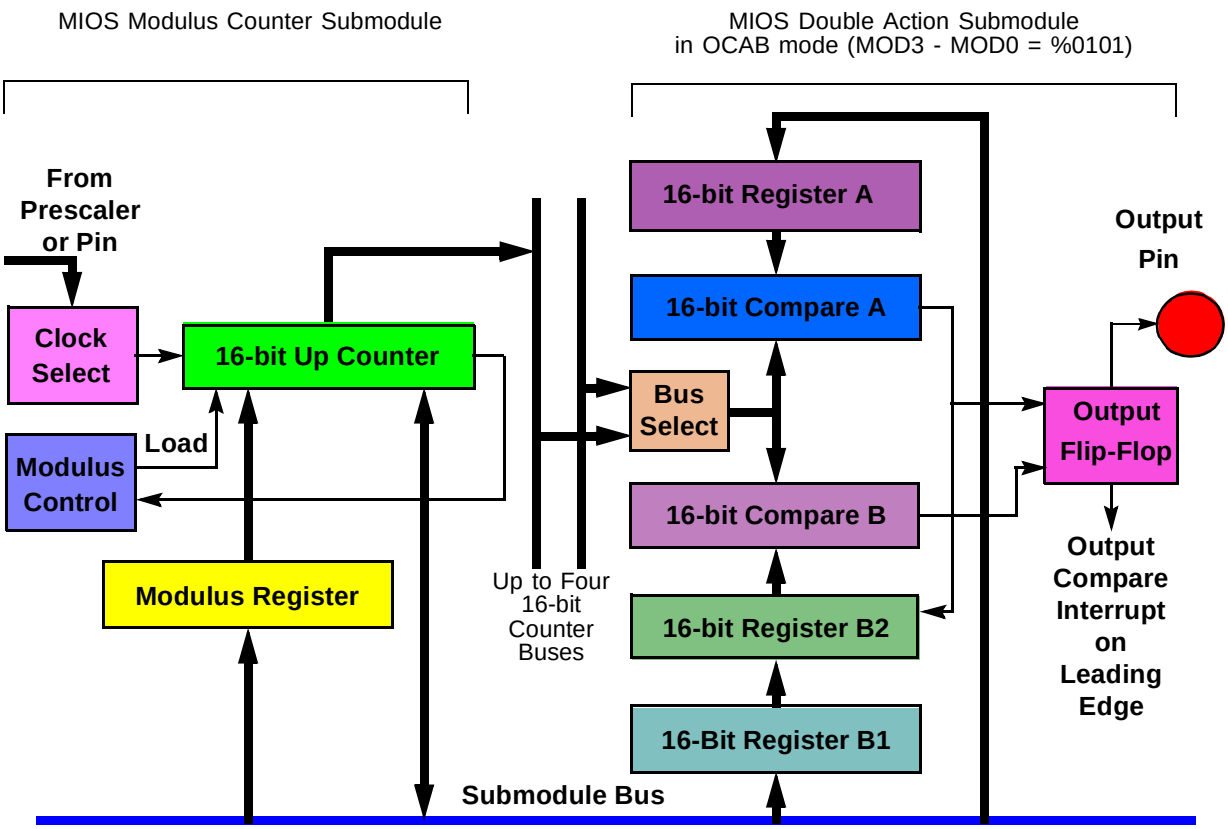


Figure 17-7 MIOS Example: Pulse Width Modulation Output

When an MMCSM is used to drive the time-base, the modulus value is the period of the output PWM signal. **Figure 17-7** shows such an example. The polarity of the leading edge of an output waveform is programmable for a rising or a falling edge. The software selects the period of the output signal by programming the MMCSM with a modulus value. The leading edge compare value is written into register A by software, and the trailing edge time is written into register B1. When the leading edge value is reached, the content of register B1 is transferred to register B2, to form the next trailing edge value. Subsequent changes to the output pulse width are made by writing a new time into register B1. Updates to the pulse width are always synchronized to the leading edge of the waveform.

It is typical to use the pulse width modulation mode of the MDASM without interrupts, though an interrupt can be enabled to occur on the leading edge. When the output is an unchanging repetitive waveform, the MDASM continuously generates the signal without any software intervention. When the software needs to change the pulse width, a new trailing edge time is written to the MDASM. The output is changed on the next full pulse. When the software needs to change the output at a regular rate, such as an acceleration curve, the leading edge interrupt gives the software one period time to update the new trailing edge time.

17.6.7.7 Input Pulse Accumulation

Counting the number of pulses on an input signal is another capability of the MIOS14. Pulse accumulation uses an MMCSM. Since the counters in the counter submodules are software accessible, pulse accumulation does not require the use of an action submodule. The pulse accumulation can operate continuously, interrupting only on binary overflow of the 16-bit counter. When an MMCSM is used, an interrupt can instead be created when the pulse accumulation reaches a preprogrammed value. To do that, the complement of the value is put in the modulus register and the interrupt occurs when the counter overflows.

17.7 MIOS14 I/O Ports

Each pin of each submodule can be used as an input, output or I/O port:

Submodule:	Number:	Type:
MPIOSM	16	I/O
MMCSM	2	Input
MDASM	1	I/O
MPWMSM	1	I/O

17.7.1 Using and Clearing Flag Bits

To clear any flag bit in the MIOS14, the software must first read the concerned flag as a one, then write a zero to it. These two steps do not have to be done on consecutive instructions. Writing a one to a flag bit has no effect.

NOTE

The flag clearing mechanism works only when no flag setting event occurs between the read and write operations. When a flag setting event occurs between the read and write operations, the flag bit is not cleared.

17.8 MIOS14 Bus System Description

In addition to its connection to the peripheral bus, there is an internal bus system within the MIOS14, called the modular I/O bus (MIOB). The MIOB is a bus that makes communications possible between any submodule and the peripheral bus master through the MBISM.

This MIOB is described in the following sections.

17.8.1 Modular I/O Bus Description

The MIOB is divided into three dedicated buses:

- The read/write and control bus
- The request bus
- The counter bus set

17.8.2 Read/Write and Control Bus Description

The read/write and control bus (RWCB) allows read and write data transfers to and from any I/O submodule through the MBISM. It includes signals for data and addresses as well as control signals. The control signals allow 16-bit simple synchronous single master accesses, and supports fast or slow masters accesses.

17.8.3 Request Bus Description

The request bus (RQB) provides interrupt requests signals along with I/O submodule identification and priority information to the MBISM.

NOTE

That some submodules do not generate interrupts and are therefore independent of the RQB.

17.8.4 Counter Bus Set Description

The counter bus set (CBS) is a set of six 16-bit counter buses. The CBS makes it possible to transfer information between submodules. Typically, counter submodules drive the CBS, while action submodules process the data on these buses.

NOTE

Some submodules are self contained and therefore independent of the counter bus set.

17.9 MIOS14 Address Map

The address space of the MIOS14 consist of 4Kbytes starting at the base address of the module (0x306000). The overall address map organization is shown in **Figure 17-1**.

The basic memory map in shown on **Figure 17-8**.

NOTE

If a supervisor privilege address space is accessed in user mode, the module will return a bus error.

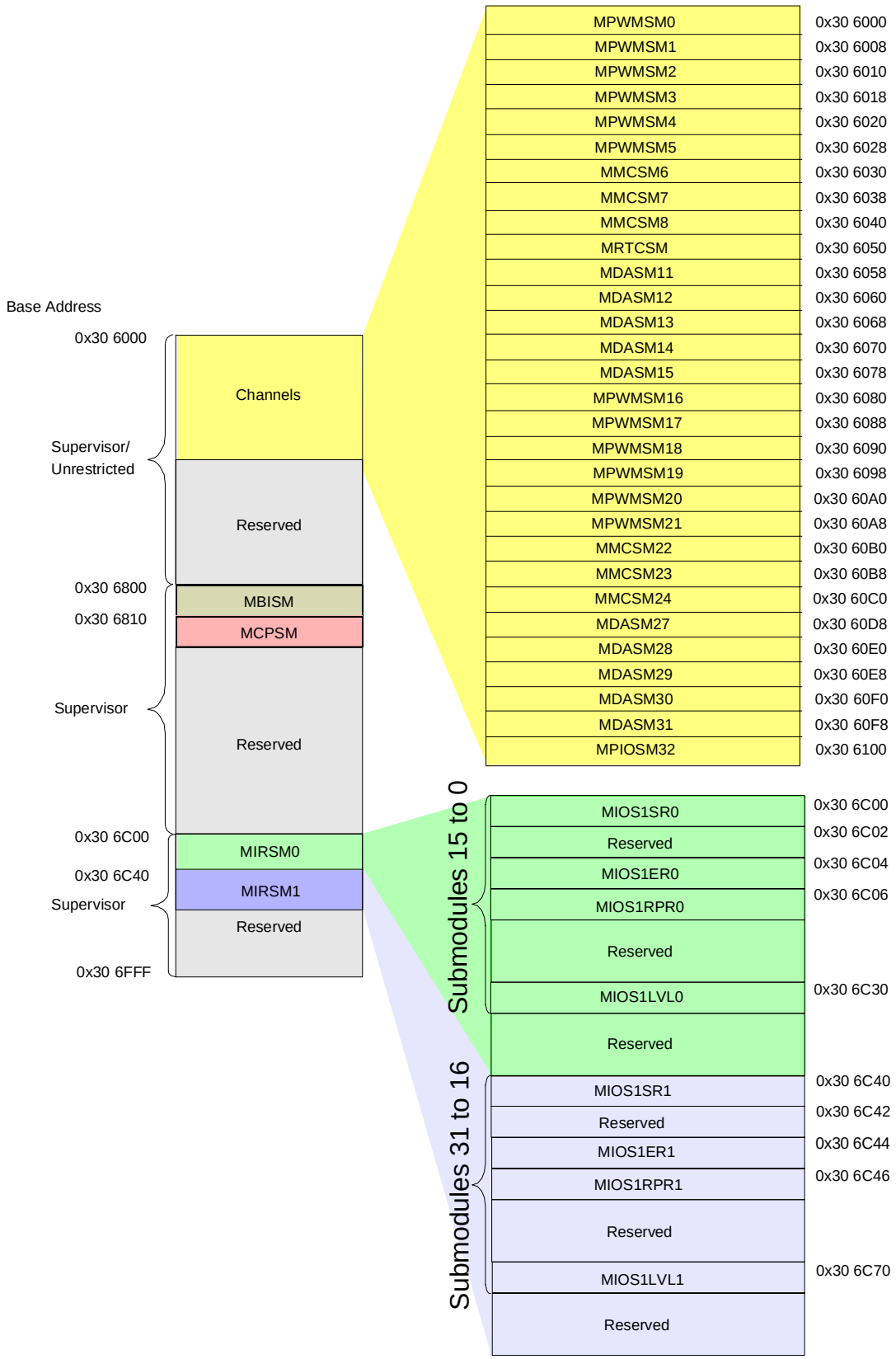


Figure 17-8 MIOS14 Memory Map

17.9.1 Interrupts

This section describes the interrupt functions of the MIOS14 and its submodules and how these interrupts are passed to the CPU via the Peripheral bus. Interrupt requests from the MIOS14 are treated as exceptions by the CPU and are dealt with by the CPU's exception processing routines. For a more detailed description of exception processing in the relevant microprocessors, please refer **SECTION 3 CENTRAL PROCESSING UNIT** and to the ***RCPU Reference Manual*** **(RCPURM/AD)**

17.9.2 MIOS14 Interrupt Structure

The MIOS14 and its submodules are capable of generating interrupts on different levels to be transmitted to the CPU via the peripheral bus. Inside the MIOS14, all the information required for requesting and servicing the interrupts are treated in two different sections:

- The interrupt control section (ICS)
- The request submodules (RQSM)

The request submodule gathers in service request flags from each group of up-to 16 submodules and transfers those requests to the MIOS14 interrupt control section (ICS). **Figure 17-9** shows a block diagram of the whole interrupt architecture.

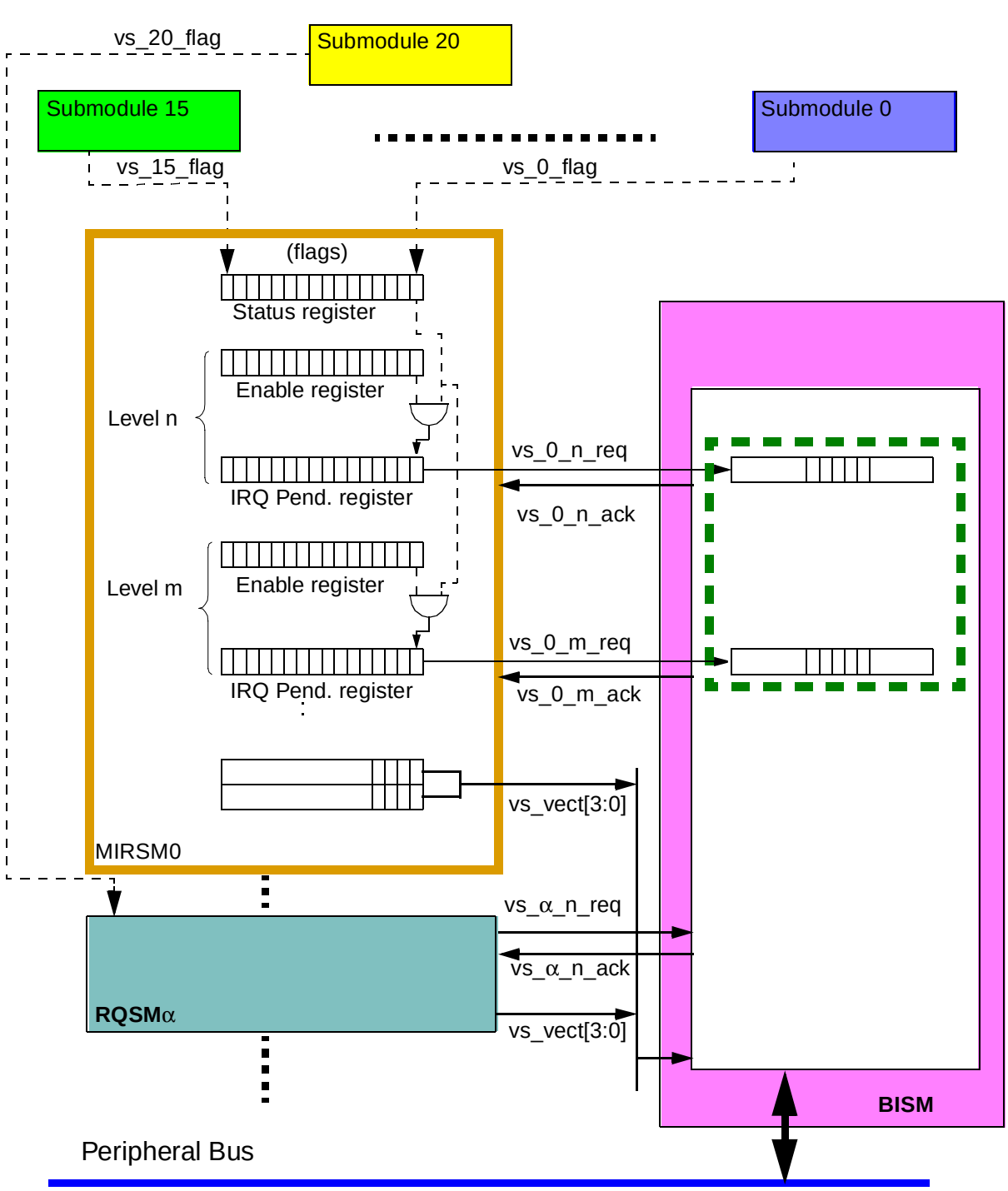


Figure 17-9 MIOS Interrupt Structure

17.9.3 Request Submodule (RQSM)

Each submodule that is capable of generating an interrupt can assert a flag line when an event occurs. In the maximum MIOS configuration, there could be up to 128 flag lines. Each RQSM serves up to 16 submodules. A particular MIOS configuration includes as many RQSM as are needed (maximum of eight).

Each RQSM includes:

- One 16-bit status register (for the flags)
- One 16-bit enable register for each implemented level
- One 16-bit IRQ pending register for each implemented level

One bit position in each of the above registers is associated with one submodule.

NOTE

If a submodule in a group of 16 cannot generate interrupts, then its corresponding flag bit in the status register is inactive and is read as zero.

When an event occurs in a submodule that activates a flag line, the corresponding flag bit in the status register is set. The status register is read/write, but a flag bit can be reset only if it has previously been read as a one. Writing a “one” to a flag bit has no effect. When the software intends to clear only one flag bit within a status register, the software must write an all-ones 16-bit value except for the bit position to be cleared which is a zero.

The enable register is initialized by the software to indicate whether each interrupt request is enabled for the levels defined in the ICS.

NOTE

In the case of multiple requests levels implementation in the same MIOS, it is possible to enable interrupts at more than one different levels for the same submodule. It is the responsibility of the software to manage this.

Each bit in the IRQ pending register is the result of a logical “AND” between the corresponding bits in the status and in the enable registers. If a flag bit is set and the level enable bit is also set, then the IRQ pending bit is set, and the information is transferred to the interrupt control section that is in charge of sending the corresponding level to the CPU. The IRQ pending register is read only.

NOTE

When the enable bit is not set for a particular submodule, the corresponding status register bit is still set when the corresponding flag is set. This allows the traditional software approach of polling the flag bits to see which ones are set. The status register makes flag polling easy, since up to 16 flag bits are contained in one register.

The submodule number of an interrupting source defines the corresponding RQSM number and the bit position in the status registers. To find the RQSM number and bit position of an interrupting source, proceed as follow:

1. Divide the interrupting submodule number by 16
2. The integer result of the division gives the RQSM number

3. The reminder of the division gives the bit position

17.9.4 MIRSM0 Registers

17.9.4.1 MIOS14SR0 Interrupt Status Register

MIOS14SR0 — Interrupt Status Register **0x30 6C00**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
FLG15	FLG14	FLG13	FLG12	FLG11	FLG10	RSVD	FLG8	FLG7	FLG6	FLG5	FLG4	FLG3	FLG2	FLG1	FLG0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-2 MIOS14SR0 Bit Descriptions

Bit(s)	Name	Description
15:11	FLG15:11	Flag Bits — MDASM flag bits [15:11]
10	FLG10	Flag Bit — RTCSM flag bit [10]
9	—	Reserved
8:6	FLG8:6	Flag Bits — MMCSM flag bits [8:6]
5:0	FLG5:0	Flag Bits — PWMSM flag bits [5:0]

This register contains the flag bits that are raised when the submodules generate an interrupt. Each bit corresponds to a given submodule.

17.9.4.2 MIOS14ER0 Interrupt Enable Registers

MIOS14ER0 — Interrupt Enable Register **0x30 6C04**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
EN15	EN14	EN13	EN12	EN11	EN10	RSVD	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
SRESET:															
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Table 17-3 MIOS14ER0 Bit Descriptions

Bit(s)	Name	Description
0:4	EN15:11	Flag Bits — MDASM flag bits [15:11]
5	EN10	Flag Bit — RTCSM flag bit [10]
6	—	Reserved
8:6	EN8:6	Flag Bits — MMCSM flag bits [8:6]
5:0	EN5:0	Flag Bits — PWMSM flag bits [5:0]

This register contains the interrupt enable bits for the submodules. Each bit corresponds to a given submodule.

17.9.4.3 MIOS14RPR0 Request Pending Register

MIOS14RPR0 — Request Pending Register **0x30 6C06**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
IRP15	IRP14	IRP13	IRP12	IRP11	IRP10	RSVD	IRP8	IRP7	IRP6	IRP5	IRP4	IRP3	IRP2	IRP1	IRP0
SRESET:															
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Table 17-4 MIOS14PR0 Bit Descriptions

Bit(s)	Name	Description
0:4	IRP15:11	Flag Bits — MDASM flag bits [15:11]
5	IRP10	Flag Bit — RTCSM flag bit [10]
6	—	Reserved
7:9	IRP8:6	Flag Bits — MMCSM flag bits [8:6]
10:15	IRP5:0	Flag Bits — PWMSM flag bits [5:0]

This register is a read only register that contains the interrupt pending bits for the sub-modules. Each bit corresponds to a given submodule. When one of these bits is set, it means that a submodule raised its flag and the corresponding enable was set.

As this register is read only, a write to this register has no other effect than generating a bus error if the bus error option is selected.

17.9.5 MIRSM1 Registers

17.9.5.1 MIOS14SR1 Interrupt Status Register

MIOS14SR1 — Interrupt Status Register **0x30 6C40**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
FLG31	FLG30	FLG29	FLG28	FLG27	RESERVED		FLG24	FLG23	FLG22	FLG21	FLG20	FLG19	FLG18	FLG17	FLG16
SRESET:															
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Table 17-5 MIOS14SR1 Bit Descriptions

Bit(s)	Name	Description
0:4	FLG31:27	Flag Bits — MDASM flag bits [31:27]
5:6	—	Reserved
7:10	FLGL24:22	Flag Bit — MMCSM flag bit [24:22]
11:15	FLG21:16	Flag Bits — PWMSM flag bits [5:0]

This register contains the flag bits that are raised when the submodules generate an interrupt. Each bit corresponds to a given submodule.

17.9.5.2 MIOS14ER1 Interrupt Enable Registers

MIOS14ER1 — Interrupt Enable Register **0x30 6C44**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
EN31	EN30	EN29	EN28	EN27	RESERVED		EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-6 MIOS14ER1 Bit Descriptions

Bit(s)	Name	Description
15:11	EN31:27	Flag Bits — MDASM flag bits [15:11]
10:9	—	Reserved
8:6	EN24:22	Flag Bits — MMCSM flag bits [8:6]
5:0	EN21:16	Flag Bits — PWMSM flag bits [5:0]

This register contains the interrupt enable bits for the submodules. Each bit corresponds to a given submodule.

17.9.5.3 MIOS14RPR1 Request Pending Register

MIOS14RPR1 — Request Pending Register **0x30 6C46**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
IRP31	IRP30	IRP29	IRP28	IRP27	RESERVED		IRP24	IRP23	IRP22	IRP21	IRP20	IRP19	IRP18	IRP17	IRP16
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-7 MIOS14PR1 Bit Descriptions

Bit(s)	Name	Description
15:11	IRP31:27	Flag Bits — MDASM flag bits [15:11]
10:9	—	Reserved
8:6	IRP24:22	Flag Bits — MMCSM flag bits [8:6]
5:0	IRP21:16	Flag Bits — PWMSM flag bits [5:0]

This register is a read only register that contains the interrupt pending bits for the submodules. Each bit corresponds to a given submodule. When one of these bits is set, it means that a submodule raised its flag and the corresponding enable was set.

As this register is read only, a write to this register has no other effect than generating a bus error if the bus error option is selected.

17.9.6 MBISM Interrupt Registers

17.9.6.1 MIOS14LVL0 Register

MIOS14LVL0 — Interrupt Level 0 Register **0x30 6C30**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					LVL2	LVL1	LVL0	TM1	TM0	RESERVED					
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-8 MIOS14LVL0 Bit Descriptions

Bit(s)	Name	Description
15:11	—	Reserved
10:8	LVL2 — LVL1	Interrupt Level 2-1 — Represent one of eight possible levels.
7:6	TMM1 — TM0	Only used in the case of non-vectored interrupt to determine the mulitplexed time slot. Read/write unused bits in the case of vectored interrupt.
5:0	—	Reserved

This register contains the interrupt level that apply to the submodules number 15 to 0.

17.9.6.2 MIOS14LVL1 Register

MIOS14LVL1 — Interrupt Level 1 Register **0x30 6C70**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					LVL2	LVL1	LVL0	TM1	TM0	RESERVED					
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-9 MIOS14LVL1 Bit Descriptions

Bit(s)	Name	Description
15:11	—	Reserved
10:8	LVL2 — LVL1	Interrupt Level 2-1 — Represent one of eight possible levels.
7:6	TMM1 — TM0	Only used in the case of non-vectored interrupt to determine the mulitplexed time slot. Read/write unused bits in the case of vectored interrupt.
5:0	—	Reserved

This register contains the interrupt level that apply to the submodules number 31 to 16.

17.9.7 Interrupt Control Section (ICS)

The function of the interrupt control section is to deliver the level and the vector, if the vectored interrupt is concerned, to the CPU. An interrupt control section adapts the

characteristics of the MIOB request bus to the characteristics of the interrupt structure of the peripheral bus.

17.9.7.1 Non-Vectored Interrupt

This ICS mode is used when the MIOS is connected to a processor that does not run an IACK cycle.

When at least one of the bits is set in an IRQ pending register, the ICS receives a signal. This signal is the result of a logical “OR” between all the bits of the IRQ pending register.

Each signal received from the IRQ pending register is associated with an interrupt level register within the ICS. This level is coded on five bits in this register: three bits represent one of eight levels and the two other ones represent the four time multiplex slots. According to this level, the ICS sets the correct IRQ[7:0] lines with the correct ILBS[1:0] time multiplex lines on the peripheral bus. The CPU is then informed of which of the 32 interrupt levels is requested.

Based on the interrupt level requested, the software must determine which submodule requested the interrupt. The software may use a find-first-one type of instruction to determine, in the concerned RQSM, which bit is set. The CPU can then serve the requested interrupt.

17.10 Bus Interface Submodule (MBISM)

17.10.1 MBISM Overview Description

The MIOS bus interface submodule (MBISM) is used as an interface between the MIOB (modular I/O bus) and the peripheral bus. It allows the CPU to communicate with the MIOS14 submodules.

17.10.2 MBISM Registers

All MBISM registers are supervisor access only.

17.10.2.1 MBISM Registers Organization

Table 17-10 MBISM Registers

	MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0x30 6800	MIOS14 Test and Pin Control Register (MIOS14TPCR)															
0x03 6802	MIOS14 Vector Register (MIOS14VECT)															
0x03 6804	MIOS14 Module-Version Number Register (MIOS14VNR)															
0x03 6806	MIOS14 Module Control Register (MIOS14MCR)															
0x03 6808	Reserved															
0x03 680A	Reserved															
0x03 680C	Reserved															
0x03 680E	Reserved															

17.10.2.2 MIOS14 Test and Pin Control Register (MIOS14TPCR)

MIOS14TPCR — Test and Pin Control Register **0x30 6800**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
TEST	RESERVED							VMUX[7:5]			PWM	RESERVED	CCAN	VF	VFLS
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-11 MIOS14TPCR Bit Descriptions

Bit(s)	Name	Description
15	—	Test — This bit is used for MIOS14 factory testing and should always be programmed to a 0.
14:8	—	Reserved
7:5	VMUX7 — VMUX5	Pin Multiplex — These bits are unused on MPC565 / MPC566.
4	PWM	PWM Pin Multiplex — This bit controls the function of the following pins: mpwm20_mpio32b11, mpwm21_mpio32b12, mpwm4_mpio32b5, mpwm5_mpio32b6. 1 = PWM function is selected (mpwm4, mpwm5, mpwm20, mpwm21) 0 = MIOS General Purpose I/O is selected (mpio32b5, mpio32b6, mpio32b11, mpio32b12)
3	—	This bit should be set to 0.
2	CCAN	TOUCAN_C Pin Multiplex — This bit controls the function of the following pins: c_cntx0_mpio32b13 c_cnrx0_mpio32b14 1 = TOUCAN_C function is selected (c_cntx0, c_cnrx0) 0 = MIOS General Purpose I/O is selected (mpio32b13, mpio32b14)
1	VF	VF Pin Multiplex — This bit controls the function of the VF pins (vf0_mpio32b0, vf1_mpio32b1, vf2_mpio32b2) 1 = VF function is selected (vf[0:2]) 0 = MIOS General Purpose I/O is selected (mpio32b0, mpio32b1, mpio32b2)
0	VFLS	VFLS Pin Multiplex — This bit controls the function of the VFLS pins (vfls0_mpio32b3, vfls1_mpio32b4) 1 = VFLS function is selected (vfls[0:1]) 0 = MIOS General Purpose I/O is selected (mpio32b0, mpio32b1, mpio32b2)

This register is used for MIOS14 factory testing and to specify the pin usage. Note that this register does not control the pin multiplexing of the MIOS GPIO pins which are shared with the READI MDO function (mdo_4_mpio32b10, mdo_5_mpio32b9, mdo_6_mpio32b8, mdo_7_mpio32b7). These pins are controlled by the READI module.

17.10.2.3 MIOS14 Vector Register (MIOS14VECT)

MIOS14VECT— Vector Register **0x30 6802**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED								VECT	VECT	VECT	VECT	RESERVED			
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-12 MIOS14VECT Bit Descriptions

Bit(s)	Name	Description
15:8	—	Reserved
7:4	VECT7 — VECT5	Interrupt Vectors MSBs — These bits are not used on the MPC565 / MPC566.
3:0	—	Reserved

17.10.2.4 MIOS14 Module-Version Number Register (MIOS14VNR)

This register is read only, and contains the hard-coded values of the version and module number. These values are different for each implementation and allow a tracking of module numbers and versionning.

MIOS14VNR — Module-Version Number Register **0x30 6804**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MN7	MN6	MN5	MN4	MN3	MN2	MN1	MN0	VN7	VN6	VN5	VN4	VN3	VN2	VN1	VN0
SRESET:															
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0

Table 17-13 MIOS14VNR Bit Descriptions

Bit(s)	Name	Description
15:8	MN7 — MN0	Module Number — These bits represent the module number. In the MPC565 / MPC566 the module number is 0xE.
7:0	VN7 — VN0	Version Number — These bits represent the version number of the described module.This has a value of 0x01 in the K85H mask set of the MPC565 / MPC566 and may be incremented in the future.

17.10.2.5 MIOS14 Module Configuration Register (MIOS14MCR)

The MIOS14MCR register is a collection of read/write stop, freeze, reset and supervisor bits, as well as interrupt arbitration number bits. These bits are detailed in **Table 17-14**.

MIOS14MCR— Module Configuration Register **0x30 6806**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	RSVD	FRZ	RST	RESERVED				SUPV	RESERVED						
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-14 MIOS14MCR Bit Descriptions

Bit(s)	Name	Description
15	STOP	Stop enable — The STOP bit, while asserted, activates the MIOB freeze signal regardless of the state of the IMB3 FREEZE signal. The MIOB freeze signal is further validated in some submodules with internal freeze enable bits in order for the submodule to be stopped. The MBISM continues to operate to allow the CPU access to the submodule's registers. The MIOB freeze signal remains active until reset or until the STOP bit is written to zero by the CPU (via the IMB3). The STOP bit is cleared by reset. 0 = Allows MIOS14 operation. 1 = Selectively stops MIOS14 operation.
14	—	Reserved
13	FRZ	Freeze enable — The FRZ bit, while asserted, activates the MIOB freeze signal only when the IMB3 FREEZE signal is active. The MIOB freeze signal is further validated in some submodules with internal freeze enable bits in order for the submodule to be frozen. The MBISM continues to operate to allow the CPU access to the submodule's registers. The MIOB freeze signal remains active until the FRZ bit is written to zero or the IMB3 FREEZE signal is negated. The FRZ bit is cleared by reset. 0 = Ignores the FREEZE signal on the IMB3, allows MIOS14 operation. 1 = Selectively stops MIOS14 operation when the FREEZE signal appears on the IMB3.
12	RST	Module reset — The RST bit is always read as 0 and can be written to 1. When the RST bit is written to 1, the MBISM activates the vs_rst signal on the MIOB. This completely stops the operation of the MIOS14 and reset all the values in the submodules registers that are affected by reset. This bit provides a way of resetting the complete MIOS14 module regardless of the reset state of the CPU. The RST bit is cleared by reset. 0 = Writing a 0 to RST has no effect. 1 = Reset the MIOS14 submodules.
11:8	—	Reserved
7	SUPV	Supervisor data space selector — The SUPV bit tells if the address space from (0x30 6000) to (0x30 67FF) in the MIOS14 is accessed at the supervisor privilege level (See the address map on **Figure 17-1**). When cleared, these addresses are accessed at the unrestricted privilege level. 0 = Unrestricted Data Space. 1 = Supervisor Data Space. The SUPV bit is cleared by reset.
6:0	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TPU3 implementations that use hardware interrupt arbitration.

17.11 Counter Prescaler Submodule (MCPSM)

17.11.1 MCPSM Overview Description

The MIOS counter prescaler submodule (MCPSM) divides the MIOS14 clock (f_{SYS}) to generate the counter clock. It is designed to provide all the submodules with the same division of the main MIOS14 clock (division of f_{SYS}). It uses a 4-bit modulus counter. The clock signal is prescaled by loading the value of the clock prescaler register into

the prescaler counter every time it overflows. This allows all prescaling factors between two and 16. Counting is enabled by asserting the PREN bit in the control register. The counter can be stopped at any time by negating this bit, thereby stopping all submodules using the output of the MCPSM (counter clock). A block diagram of the MCPSM is given in **Figure 17-10**.

The following sections describe the MCPSM in detail.

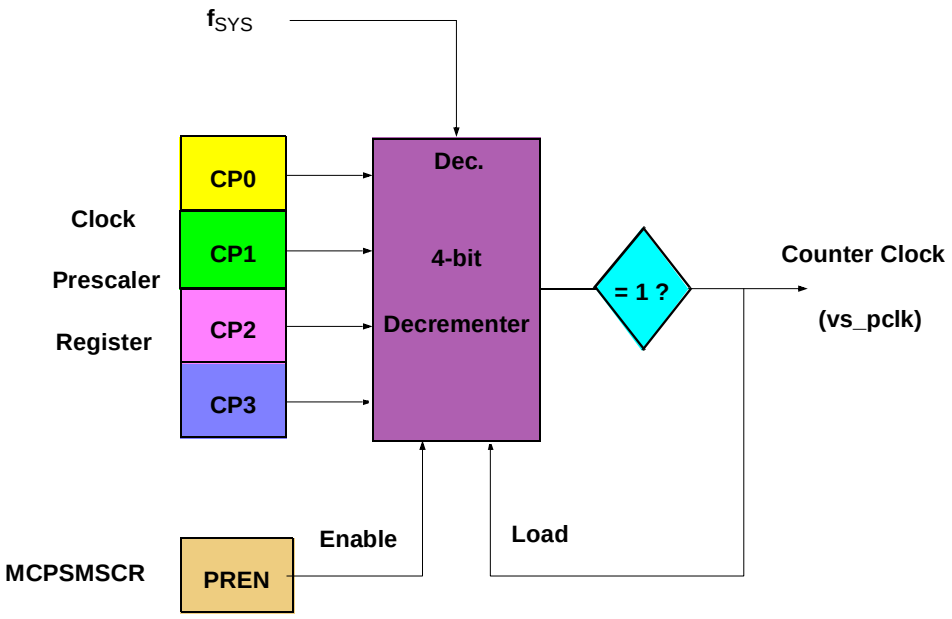


Figure 17-10 MCPSM Block Diagram

17.11.1.1 MCPSM features

- Centralized counter clock generator
- Programmable 4-bit modulus down-counter
- Wide range of possible division ratios:
  2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 or 16
- Count inhibit under software control

17.11.1.2 MCPSM Pin Functions

The MCPSM has no associated external pins.

17.11.1.3 Modular I/O Bus (MIOB) Interface

- The MCPSM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MCPSM registers, and to control the MCPSM in the different possible situations.
- The MIOS counter prescaler submodule does not use any 16-bit counter bus.

• The MIOS counter prescaler submodule does not use the request bus.

17.11.2 Effect of RESET on MCPSM

When the RESET signal is asserted, all the bits in the MCPSM status and control register are cleared.

NOTE

The MCPSM is still disabled after the RESET signal is negated and counting must be explicitly enabled by asserting the PREN bit.

17.11.3 MCPSM Registers

The privilege level to access to the MCPSM registers is supervisor only.

17.11.3.1 MCPSM Registers Organization

MCPSM— Register Organization

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0x30 6810	Reserved														
0x30 6812	Reserved														
0x30 6814	Reserved														
0x30 6816	MCPSM Status/Control Register (MCPSMSCR)														

17.11.3.2 MCPSMSCR — MCPSM Status/Control Register

MCPSMSCR — MCPSM Status/Control Register **0x30 6816**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PREN	FREN	RESERVED										PSL3	PSL2	PSL1	PSL0
SRESET:															
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Table 17-15 MCPSMSCR Bit Descriptions

Bit(s)	Name	Description
0	PREN	Prescaler enable bit — This active high read/write control bit enables the MCPSM counter. The PREN bit is cleared by reset. 1 = MCPSM counter enabled. 0 = MCPSM counter disabled.
1	FREN	Freeze bit — This active high read/write control bit when set make possible a freeze of the MCPSM counter if the MIOB freeze line is activated: note that this line is active when the MIOS14MCR STOP bit is set or when the MIOS14MCR FREN bit and the IMB3 FREEZE line are set. When the MCPSM is frozen, its stops counting. Then when the FREN bit is reset or when the freeze condition on the MIOB is negated, the counter restart from where it was before to freeze. The FREN bit is cleared by reset. 1 = MCPSM counter frozen if MIOB freeze active. 0 = MCPSM counter not frozen.

Table 17-15 MCPSMSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
2:11	—	Reserved
12:15	PSL3 — PSL0	Clock prescaler — This 4-bit read/write data register stores the modulus value for loading into the clock prescaler. The new value is loaded into the counter on the next time the counter equals one or when disabled (PREN bit at “0”).

Table 17-16 Clock Prescaler Setting

PSL3-PSL0 Value	Divide ratio
0x0	16
0x1	No counter clock output (vs_pclk signal is set to 0)
0x2	2
0x3	3
...	...
0xE	14
0xF	15

Note that if the binary value %0001 is entered in PSL3 — PSL0, the output signal is stuck at zero, no clock is output.

17.12 Modulus Counter Submodule (MMCSM)

In this section the values taken by the bits in the registers are given according to the following rule:

- 0 -> reset (negated)
- 1 -> set (asserted)
- u -> undefined
- - -> unaffected

17.12.1 MMCSM Overview Description

The MMCSM is a versatile counter submodule capable of performing complex counting and timing functions, including modulus counting, in a wide range of applications. The MMCSM may also be configured as an event counter, allowing the overflow flag to be set after a predefined number of events (internal clocks or external events), or as a time source for other submodules.

NOTE

The MMCSM can also operate as a free running counter by loading the modulus value of zero.

The main components of the MMCSM are an 8-bit prescaler counter, an 8-bit prescaler register, a 16-bit up-counter register, a 16-bit modulus latch register, counter loading and interrupt flag generation logic.

The contents of the modulus latch register is transferred to the counter under the following three conditions:

1. When an overflow occurs
2. When an appropriate transition occurs on the external load pin
3. When the program writes to the counter register. In this case, the value is first written into the modulus register and immediately transferred to the counter.

Software can also write a value to the modulus register for later loading into the counter with one of the two first criteria.

A software control register selects whether the clock input to the counter is one of the prescaler outputs or the corresponding input pin. The polarity of the external input pin is also programmable.

The following sections describe the MMCSM in detail. A block diagram of the MMCSM is shown in **Figure 17-11**.

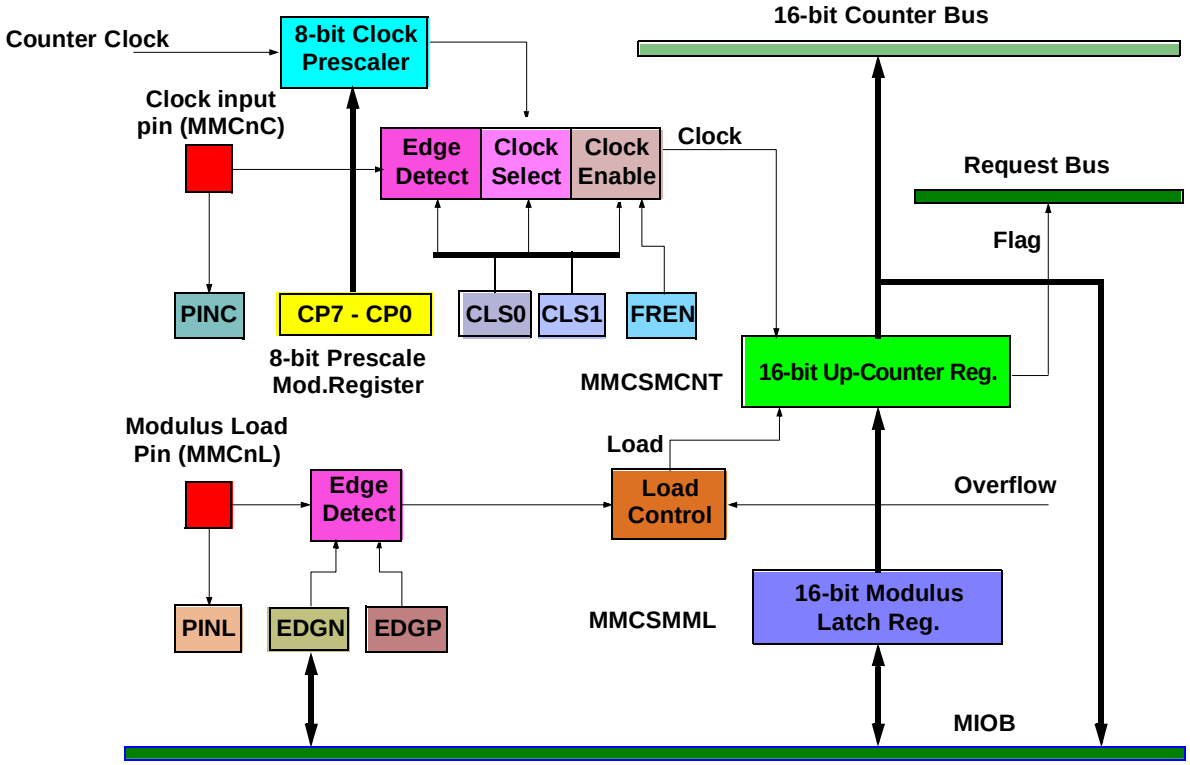


Figure 17-11 MMCSM Block Diagram

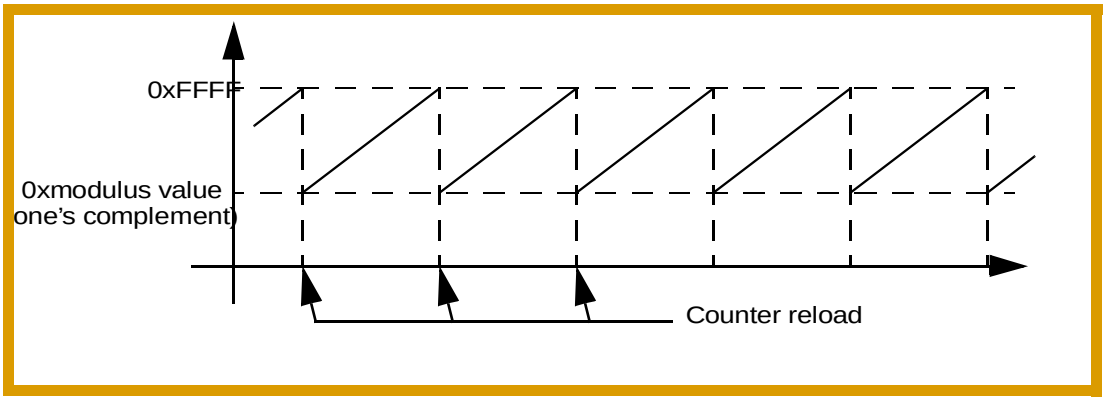


Figure 17-12 MMCSM Modulus Up-Counter

17.12.2 MMCSM features

- Programmable 16-bit modulus up-counter with a built-in programmable 8-bit prescaler clocked by MCPSM
- Maximum increment frequency of the counter:
  - clocked by the internal MCPSM output: f_{SYS} / 2
  - clocked by the external pin: f_{SYS} / 4
- Flag setting and possible interrupt generation on overflow of the up-counter register
- Time counter on internal clock with interrupt capability after a pre-determined time
- External event counter (pulse accumulator) with overflow and interrupt capability after a pre-determined number of external events
- Usable as a regular free-running up-counter
- Capable of driving a dedicated 16-bit counter bus to provide timing information to action submodules (the value driven is the contents of the 16-bit up-counter register)
- Optional pin for counting external events
- Optional pin to externally force a load of the modulus counter

17.12.2.1 MMCSM Pin Functions

The MMCSM has two dedicated external pins.

An external modulus load pin (MMCnL) allows the modulus value stored in the modulus latch register (MMCSMML) to be loaded into the up-counter register (MMCSMCNT) at any time. Both rising and falling edges of the load signal may be used, according to the EDGEP and EDGEN bit settings in the MMCSMSCR.

An external event clock pin (MMCnC) can be selected as the clock source for the up-counter register (MMCSMCNT) by setting the appropriate bits (CLS0 and CLS1) in the status/control register (MMCSMSCR). Either rising or falling edge may be used according to the setting of these bits.

When the external clock source is selected, the MMCSM is in the event counter mode. The counter can simply counts the number of events occurring on the input pin. Alternatively, the MMCSM can be programmed to generate an interrupt when a predefined number of events have been counted; this is done by presetting the counter with the *complement value of the desired number of events*.

17.12.3 MMCSM Prescaler

The built-in prescaler consists of a 8-bit modulus counter, clocked by the MCPSM output. It is loaded with an 8-bit value every time the counter overflows or whenever the prescaler output is selected as the clock source. This 8-bit value is stored in the MMCSMSCR bits CP7 — CP0. The prescaler overflow signal is used to clock the MMCSM up-counter. This allows the MMCSMCNT to be incremented at the MCPSM output frequency divided by a value between one and 256.

17.12.4 Modular I/O Bus (MIOB) Interface

- The MMCSM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MMCSM registers, and to control the MMCSM in the different possible situations.
- The MMCSM drives a dedicated 16-bit counter bus with the value currently in the up-counter register
- The MMCSM uses the request bus to transmit the FLAG line to the request submodule (RQSM). A flag is set when an overflow has occurred in the up-counter register.

17.12.5 Effect of RESET on MMCSM

When the RESET signal is asserted, only the FREN, EDGP, EDGN, CLS1 and CLS0 bits in the MMCSMSCR are cleared. The clock prescaler (CP7-CP0), PINC and PINL bits in the same register are not cleared.

- The PINC and PINL bits in the MMCSMSCR always reflect the state of the appropriate external pins.
- The MMCSM is disabled after reset and must be explicitly enabled by selecting a clock source using the CLS1-CLS0 bits.

The MMCSMCNT and the MMCSMML, together with the clock prescaler register bits, must be initialized by software, since they are undefined after a hardware reset. A modulus value must be written to the MMCSMCNT (which also writes into the MMCSMML) before the MMCSMSCR is written to. The latter access initializes the clock prescaler.

17.12.6 MMCSM Registers

The privilege level to access to the MMCSM registers depends on the MIOS14MCR SUPV bit. The privilege level is unrestricted after reset and can be change to supervisor by software.

17.12.7 MMCSM Register Organization

Table 17-17 MMCSM Address Map

Address	Register
MMCSM6	
0x30 6030	MMCSM6 Up-Counter Register (MMCSMCNT) See **Table 17-17** for bit descriptions.
0x30 6032	MMCSM6 Modulus Latch Register (MMCSMML) See **Table 17-19** for bit descriptions.
0x30 6034	MMCSM6 Status/Control Register Duplicated (MMCSMSCRD) See **17.12.8.3 MMCSM Status/Control Register (MMCSMSCR)** for bit descriptions.
0x30 6036	MMCSM6 Status/Control Register (MMCSMSCR). See **Table 17-20** for bit descriptions.
MMCSM7	
0x30 6038	MMCSM7 Up-Counter Register (MMCSMCNT)
0x30 603A	MMCSM7 Modulus Latch Register (MMCSMML)
0x30 603C	MMCSM7 Status/Control Register Duplicated (MMCSMSCRD)
0x30 603E	MMCSM7 Status/Control Register (MMCSMSCR)
MMCSM8	
0x30 6040	MMCSM8 Up-Counter Register (MMCSMCNT)
0x30 6042	MMCSM8 Modulus Latch Register (MMCSMML)
0x30 6044	MMCSM8 Status/Control Register Duplicated (MMCSMSCRD)
0x30 6046	MMCSM8 Status/Control Register (MMCSMSCR)
MMCSM22	
0x30 60B0	MMCSM22 Up-Counter Register (MMCSMCNT)
0x30 60B2	MMCSM22 Modulus Latch Register (MMCSMML)
0x30 60B4	MMCSM22 Status/Control Register Duplicated (MMCSMSCRD)
0x30 60B6	MMCSM22 Status/Control Register (MMCSMSCR)
MMCSM23	
0x30 60B8	MMCSM23 Up-Counter Register (MMCSMCNT)
0x30 60BA	MMCSM23 Modulus Latch Register (MMCSMML)
0x30 60BC	MMCSM23 Status/Control Register Duplicated (MMCSMSCRD)
0x30 60BE	MMCSM23 Status/Control Register (MMCSMSCR)

Table 17-17 MMCSM Address Map (Continued)

Address	Register
MMCSM24	
0x30 60C0	MMCSM24 Up-Counter Register (MMCSMCNT)
0x30 60C2	MMCSM24 Modulus Latch Register (MMCSMML)
0x30 60C4	MMCSM24 Status/Control Register Duplicated (MMCSMSCRD)
0x30 60C6	MMCSM24 Status/Control Register (MMCSMSCR)

17.12.8 MMCSM Up-Counter Register (MMCSMCNT)

MMCSMCNT — MMCSM Up-Counter Register

0x30 6030
0x30 6038
0x30 6040
0x30 60B0
0x30 60B8
0x30 60C0

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CNT15	CNT14	CNT13	CNT12	CNT11	CNT10	CNT9	CNT8	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2	CNT1	CNT0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-18 MMCSMCNT Bit Descriptions

Bit(s)	Name	Description
0:15	CNT15 — CNT0	Counter value — These bits are read/write data bits representing the 16-bit value of the up-counter. It contains the value that is driven onto the 16-bit counter bus. **NOTE:** Writing to MMCSMCNT simultaneously writes to MMCSMML.

17.12.8.1 MMCSM Modulus Latch Register (MMCSMML)

MMCSMML — MMCSM Modulus Latch Register

0x30 6032
0x30 603A
0x30 6042
0x30 60B2
0x30 60BA
0x30 60C2

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
ML15	ML14	ML13	ML12	ML11	ML10	ML9	ML8	ML7	ML6	ML5	ML4	ML3	ML2	ML1	ML0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-19 MMCSMML Bit Descriptions

Bit(s)	Name	Description
0:15	ML15 — ML0	Modulus latches — These bits are read/write data bits containing the 16-bit modulus value to be loaded into the up-counter. The value loaded in this register must be the one's complement of the desired modulus count. The up-counter increments from this one's complement value up to 0xFFFF to get the correct number of steps before an overflow is generated to reload the modulus value into the up-counter.

17.12.8.2 MMCSMSCRD — MMCSM Status/Control Register (Duplicated)

The MMCSMSCRD and the MMCSMSCR are the same registers accessed at two different addresses.

Reading or writing to one of these two addresses has exactly the same effect.

The duplication of the SCR register allows coherent 32-bit accesses when using a RCPU.

WARNING

The user should not write directly to the address of the MMCSM-SCRD. This register's address may be reserved for future use and should not be accessed by the software to ensure future software compatibility.

17.12.8.3 MMCSM Status/Control Register (MMCSMSCR)

The status/control register (SCR) is a collection of read-only pin status bits, read/write control bits and an 8-bit read/write data register, as detailed below.

MMCSMSCR — MMCSM Status/Control Register **0x30 6036**
0x30 603E
0x30 6046
0x30 60B6
0x30 60BE
0x30 60C6

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PINC	PINL	FREN	EDGN	EDGP	CLS1	CLS0	RSVD	CP7	CP6	CP5	CP4	CP3	CP2	CP1	CP0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-20 MMCSMSCR Bit Descriptions

Bit(s)	Name	Description
0	PINC	Clock input pin status bit — This read-only status bit reflects the logic state of the clock input pin MMCnC.
1	PINL	Modulus load input pin status bit — This read-only status bit reflects the logic state of the modulus load pin MMCnL.

Table 17-20 MMCSMSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
2	FREN	Freeze enable — This active high read/write control bit enables the MMCSM to recognize the MIOB freeze signal.
3	EDGN	Modulus load falling-edge sensitivity — This active high read/write control bit sets falling-edge sensitivity for the MMCnL pin, such that a high-to-low transition causes a load of the MMCSMCNT.
4	EDGP	Modulus load rising-edge sensitivity This active high read/write control bit sets rising-edge sensitivity for the MMCnL pin, such that a low-to-high transition causes a load of the MMCSMCNT. See **Table 17-21** for details about edge sensitivity.
5:6	CLS1 — CLS0	Clock select — These read/write control bits select the clock source for the modulus counter. Either the rising edge or falling edge of the clock signal on the MMCnC pin may be selected, as well as, the internal MMCSM prescaler output or disable mode (no clock source). See **Table 17-22** for details about the clock selection.
7	—	Reserved
8:15	CP7 — CP0	Clock prescaler — This 8-bit read/write data register stores the modulus value for loading into the built-in 8-bit clock prescaler. The new value is loaded into the prescaler counter on the next counter overflow, or upon setting the CLS1 — CLS0 bits for selecting the clock prescaler as the clock source. **Table 17-23** gives the clock divide ratio according to the CP7..CP0 values:

Table 17-21 MMCSMCNT Edge Sensitivity

EDGN	EDGP	Edge Sensitivity
1	1	MMCSMCNT load on rising and falling edges
1	0	MMCSMCNT load on falling edges
0	1	MMCSMCNT load on rising edges
0	0	None (disabled)

Table 17-22 MMCSMCNT Clock Signal

CLS1	CLS0	Clocking Selected
1	1	MMCSM clock prescaler
1	0	Clock pin rising-edge
0	1	Clock pin falling-edge
0	0	None (disable)

Table 17-23 Prescaler Values

Prescaler Value (CP7..CP0 in Hex)	MIOS14 Prescaler Clock Divided By:
FF	1
FE	2
FD	3
FC	4

Table 17-23 Prescaler Values

Prescaler Value (CP7..CP0 in Hex)	MIOS14 Prescaler Clock Divided By:
FB	5
FA	6
F9	7
F8	8
.....	.....
02	254 (2^8 -2)
01	255 (2^8 -1)
00	256 (2^8)

17.12.9 Double Action Submodule (MDASM)

In this section the values taken by the bits in the registers are given according to the following rule:

- 0 -> reset (negated))
- 1 -> set (asserted)
- u -> undefined
- - -> unaffected

17.12.10 MDASM Overview Description

The MIOS double action submodule (MDASM) is a function included in the MIOS library. It is a versatile 16-bit dual action submodule capable of performing two event operations before software intervention is required. It can perform two event operations such as PWM generation and measurement, input capture, output compare, etc.

The MDASM is composed of two timing channels (A and B), an output flip-flop, an input edge detector and some control logic. All control and status bits are contained in the MDASM status and control register.

The following sections describe the MDASM in detail. A block diagram of the MDASM is shown in **Figure 17-13**.

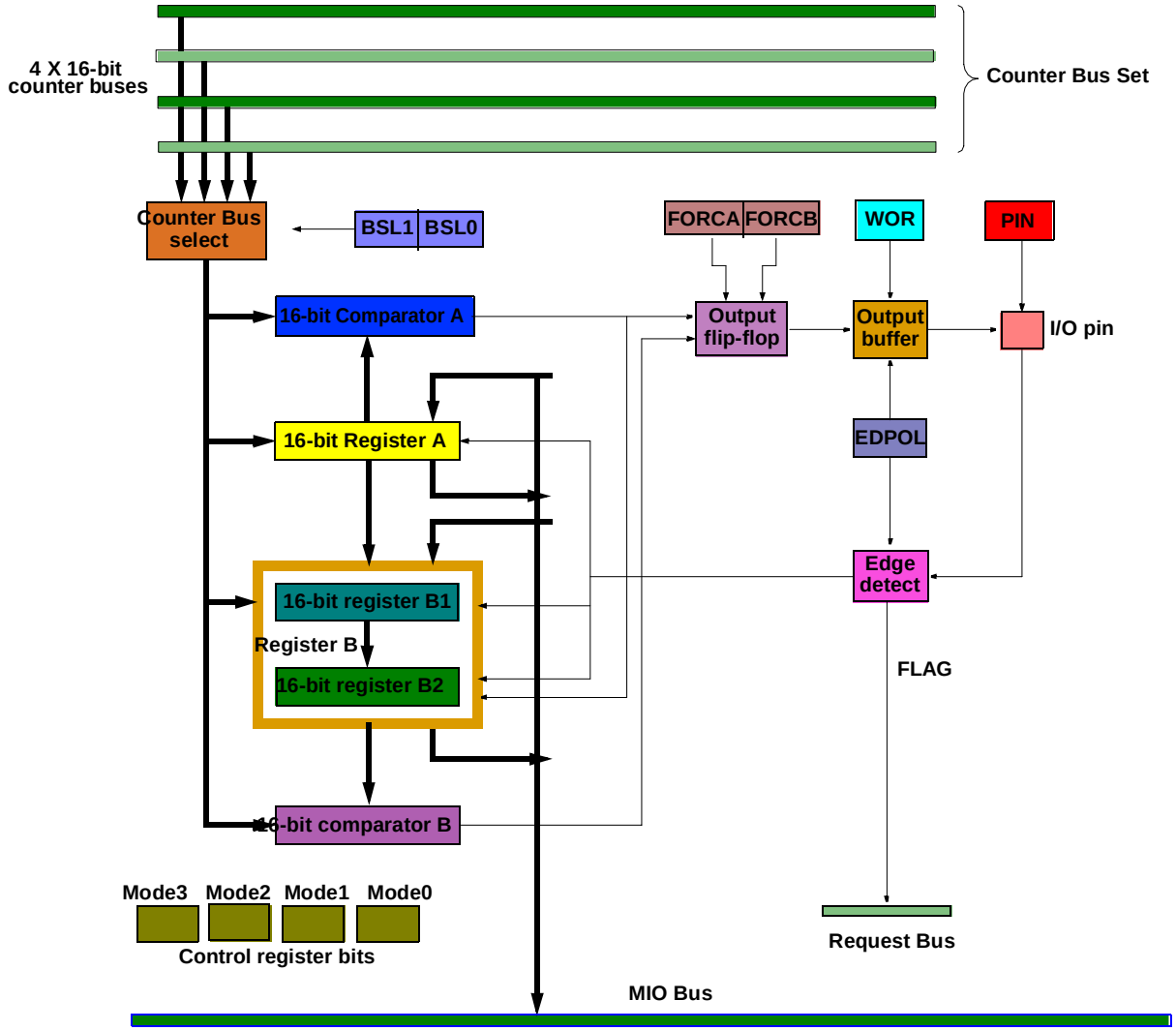


Figure 17-13 MDASM Block Diagram

17.12.11 MDASM Features

- Versatile 16-bit dual action unit allowing up to two events to occur before software intervention is required
- Six software selectable modes allowing the MDASM to perform pulse width and period measurements, PWM generation, single input capture and output compare operations as well as port functions
- Software selection of one of the four possible 16-bit counter buses used for timing operations
- Flag setting and possible interrupt generation after MDASM action completion
- Software selection of output pulse polarity
- Software selection of totem-pole or open-drain output
- Software readable output pin status

17.12.11.1 MDASM Pin Functions

The MDASM has one dedicated external pin. This pin is used in input or in output depending on the selected mode. When in input, it allows the MDASM to perform input capture, input pulse width measurement and input period measurement. When in output, it allows output compare, single shot output pulse, single output compare and output port bit operations as well as output pulse width modulation.

NOTE

In disable mode, the pin becomes a high impedance input and the input level on this pin is reflected by the state of the PIN bit in the MDASMSCR register.

17.12.12 MDASM Description

The MDASM contains two timing channels A and B associated with the same input/output pin. The dual action submodule is so called because its timing channel configuration allows two events (input capture or output compare) to occur before software intervention is required.

Six operating modes allow the software to use the MDASM's input capture and output compare functions to perform pulse width measurement, period measurement, single pulse generation and continuous pulse width generation, as well as standard input capture and output compare. The MDASM can also work as a single I/O pin. See **Table 17-24** for details.

Channel A comprises one 16-bit data register and one 16-bit comparator. Channel B also consists of one 16-bit data register and one 16-bit comparator, however, internally, channel B has two data registers B1 and B2, and the operating mode determines which register is accessed by the software:

- In the input modes (IPWM, IPM and IC), registers A and B2 are used to hold the captured values; in these modes, the B1 register is used as a temporary latch for channel B.
- In the output compare modes (OCB and OCAB), registers A and B2 are used to define the output pulse; register B1 is not used in these modes.
- In the output pulse width modulation mode (OPWM), registers A and B1 are used as primary registers and hidden register B2 is used as a double buffer for channel B.

Register contents are always transferred automatically at the correct time so that the minimum pulse (measurement or generation) is just one 16-bit counter bus count. The A and B data registers are always read/write registers, accessible via the MIOB.

In the input modes, the edge detect circuitry triggers a capture whenever a rising or falling edge (as defined by the EDPOL bit) is applied to the input pin. The signal on the input pin is Schmitt triggered and synchronized with the MIOS CLOCK.

In the disable mode (DIS) and in the input modes, the PIN bit reflects the state present on the input pin (after being Schmitt triggered and synchronized). In the output modes the PIN bit reflects the value present on the output flip-flop.

The output flip-flop is used in output modes to hold the logic level applied to the output pin.

The 16-bit counter bus selector is common to all input and output functions; it connects the MDASM to one of the four 16-bit counter buses available to that submodule instance and is controlled in software by the 16-bit counter bus selector bits BSL0 and BSL1 in the MDASMSCR register.

17.12.13 MDASM Modes of Operation

The mode of operation of the MDASM is determined by the mode select bits MODE[3:0] in the MDASMSCR register (see **Table 17-24**).

Table 17-24 MDASM Modes Of Operation

MODE[3:0]	Mode	Description of Mode
0000	DIS	Disabled — Input pin is high impedance; PIN gives state of the input pin.
0001	IPWM	Input pulse width measurement — Capture on the leading edge and the trailing edge of an input pulse.
0010	IPM	Input period measurement — Capture two consecutive rising/falling edges.
0011	IC	Input capture — Capture when the designated edge is detected.
0100	OCB	Output compare, flag line activated on B compare — Generate leading and trailing edges of an output pulse.
0101	OCAB	Output compare, flag line activated on A and B compare — Generate leading and trailing edges of an output pulse.
1xxx	OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15 or 16 bits of resolution.

To avoid spurious interrupts, and to make sure that the FLAG line is activated according to the newly selected mode, the following sequence of operations should be adopted when changing mode:

1. Disable MDASM interrupts (by resetting the enable bit in the relevant RQSM)
2. Change mode (via disable mode)
3. Reset the corresponding FLAG bit in the relevant RQSM
4. Re-enable MDASM interrupts (if desired)

NOTE

When changing between output modes, it is not necessary to follow this procedure, as in these modes the FLAG bit merely indicates to the software that the compare value can be updated. However

changing modes without passing via the disable mode does not guarantee the subsequent functionality.

17.12.13.1 Disable (DIS) Mode

The disable mode is selected by setting MODE[3:0] to 0000.

In this mode, all input capture and output compare functions of the MDASM are disabled and the FLAG line is maintained inactive, but the input port pin function remains available. The associated pin becomes a high impedance input and the input level on this pin is reflected by the state of the PIN bit in the MDASMSCR register. All control bits remain accessible, allowing the software to prepare for future mode selection. Data registers A and B are accessible at consecutive addresses. Writing to data register B stores the same value in registers B1 and B2.

WARNING

When changing modes, it is imperative to go through the DIS mode in order to reset the MDASM's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results, and to flags being set incorrectly.

17.12.13.2 Input Pulse Width Measurement (IPWM) Mode

IPWM mode is selected by setting MODE[3:0] to 0001.

This mode allows the width of a positive or negative pulse to be determined by capturing the leading edge of the pulse on channel B and the trailing edge of the pulse on channel A; successive captures are done on consecutive edges of opposite polarity. The edge sensitivity is selected by the EDPOL bit in the MDASMSCR register.

This mode also allows the software to determine the logic level on the input pin at any time by reading the PIN bit in the MDASMSCR register.

The channel A input capture function remains disabled until the first leading edge triggers the first input capture on channel B. When this leading edge is detected, the count value of the 16-bit counter bus selected by the BSL[1:0] bits is latched in the 16-bit data register B1; the FLAG line is not activated. When the next trailing edge is detected, the count value of the 16-bit counter bus is latched into the 16-bit data register A and, at the same time, the FLAG line is activated and the contents of register B1 are transferred to register B2. Reading data register B returns the value in register B2. If subsequent input capture events occur while the FLAG bit is set in the corresponding RQSM, data registers A and B will be updated with the latest captured values and the FLAG line will remain active.

If a 32-bit coherent operation is in progress when the trailing edge is detected, the transfer from B1 to B2 is deferred until the coherent operation is completed. Operation of the MDASM then continues on channels B and A as previously described.

The input pulse width is calculated by subtracting the value in data register B from the value in data register A.

Figure 17-14 provides an example of how the MDASM can be used for input pulse width measurement.

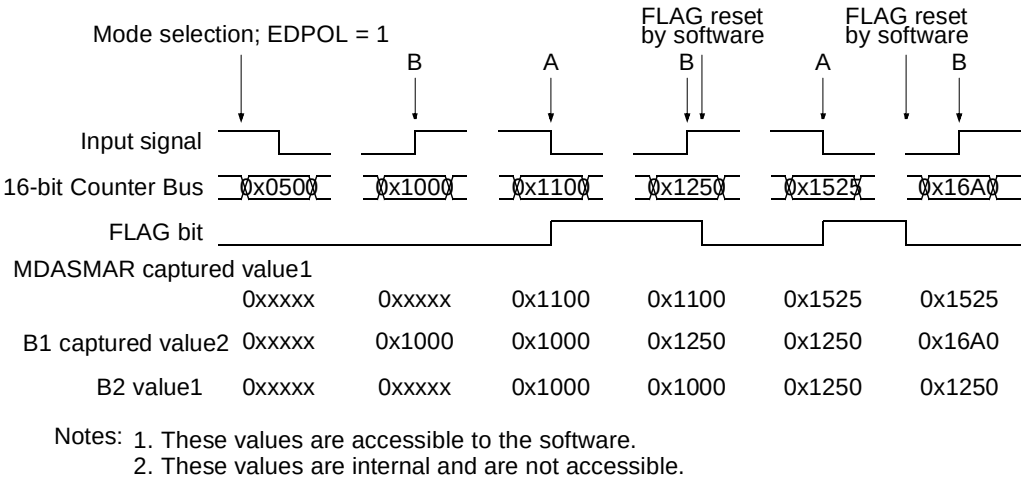


Figure 17-14 Input Pulse Width Measurement Example

17.12.14 Input Period Measurement (IPM) Mode

IPM mode is selected by setting MODE[3:0] to 0010.

This mode allows the period of an input signal to be determined by capturing two consecutive rising edges or two consecutive falling edges; successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the MDASMSCR register.

This mode also allows the software to determine the logic level on the input pin at any time by reading the PIN bit in the MDASMSCR register.

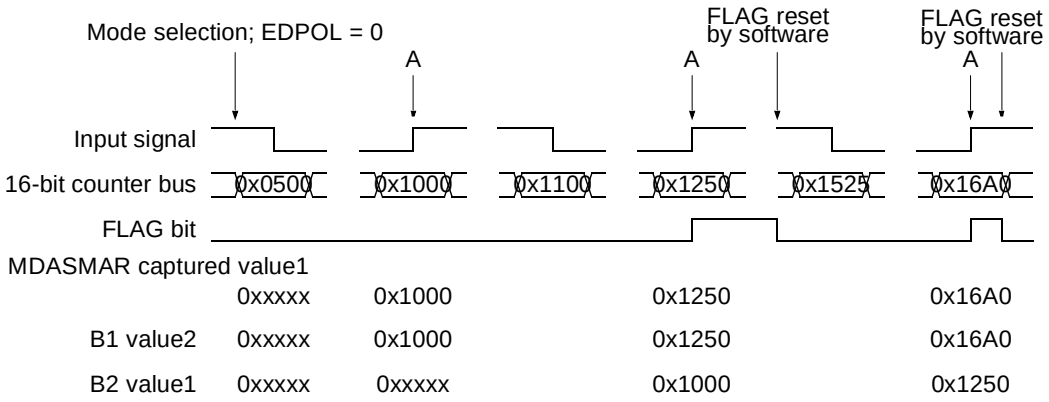
When the first edge having the selected polarity is detected, the 16-bit counter bus value is latched into the 16-bit data register A, the data in register B1 is transferred to data register B2 and finally the data in register A is transferred to register B1. On this first capture the FLAG line is not activated, and the value in register B2 is meaningless. On the second and subsequent captures, the FLAG line is activated when the data in register A is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into data register A, the data in register B1 is transferred to data register B2, the FLAG line is activated to signify that the beginning and end points of a complete period have been captured, and finally the data in register A is transferred to register B1. This sequence of events is repeated for each subsequent capture. Reading data register B returns the value in register B2.

If a 32-bit coherent operation is in progress when an edge (except for the first edge) is detected, the transfer of data from B1 to B2 is deferred until the coherent operation is completed. At any time, the input level present on the input pin can be read on the PIN bit.

The input pulse period is calculated by subtracting the value in data register B from the value in data register A.

Figure 17-15 provides an example of how the MDASM can be used for input period measurement.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 17-15 Input Period Measurement Example

17.12.15 Input Capture (IC) Mode

IC mode is selected by setting MODE[3:0] to 0011.

This mode is identical to the input period measurement mode (IPM) described above, with the exception that the FLAG line is also activated at the occurrence of the first detected edge of the selected polarity. In this mode the MDASM functions as a standard input capture function. In this case the value latched in channel B can be ignored.

Figure 17-16 provides an example of how the MDASM can be used for input capture.

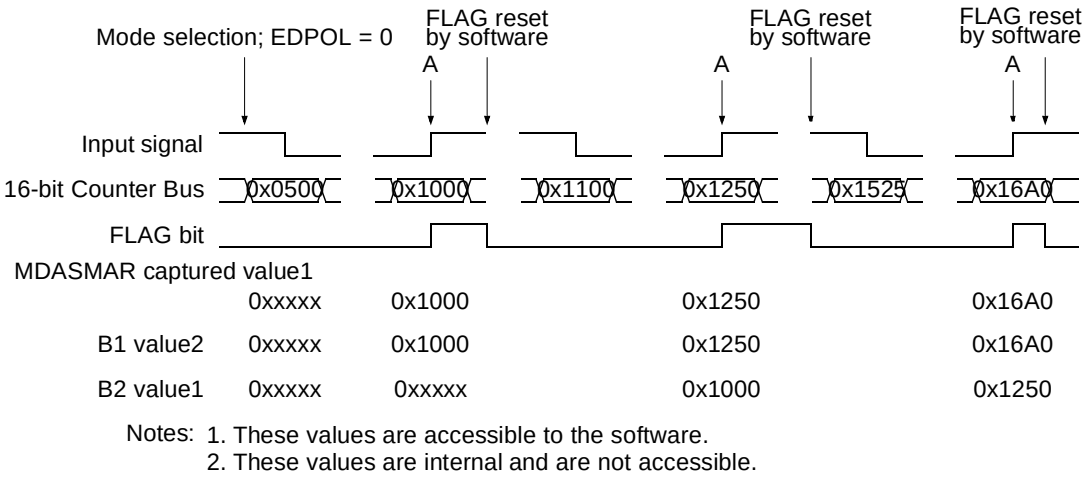


Figure 17-16 MDASM Input Capture Example

17.12.15.1 Output Compare (OCB and OCAB) Modes

Output compare mode (either OCA or OCB) is selected by setting MODE[3:0] to 010x. The MODE0 bit controls the activation criteria for the FLAG line, (i.e., when a compare occurs only on channel B or when a compare occurs on either channel).

This mode allows the MDASM to perform four different output functions:

- Single-shot output pulse (two edges), with FLAG line activated on the second edge
- Single-shot output pulse (two edges), with FLAG line activated on both edges
- Single-shot output transition (one edge)
- Output port pin, with output compare function disabled

In this mode the leading and trailing edges of variable width output pulses are generated by calculated output compare events occurring on channels A and B, respectively. OC mode may also be used to perform a single output compare function, or may be used as an output port bit.

In this mode, channel B is accessed via register B2. A write to register B2 writes the same value to register B1 even though the contents of B1 are not used in this mode. Both channels work together to generate one ‘single shot’ output pulse signal. Channel A defines the leading edge of the output pulse, while channel B defines the trailing edge of the pulse. FLAG line activation can be done when a match occurs on channel B only or when a compare occurs on either channel (as defined by the MODE0 bit in the MDASMSCR register).

When this mode is first selected, (i.e., coming from disable mode, both comparators are disabled). Each comparator is enabled by writing to its data register; it remains enabled until the next successful comparison is made on that channel, whereupon it

is disabled. The values stored in registers A and B are compared with the count value on the selected 16-bit counter bus when their corresponding comparators are enabled.

The output flip-flop is set when a match occurs on channel A. The output flip-flop is reset when a match occurs on channel B. The polarity of the output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the PIN bit.

If subsequent enabled output compares occur on channels A and B, the output pulses continue to be output, regardless of the state of the FLAG bit.

At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to a comparison on channel A or B, respectively.

NOTE

The FLAG line is not affected by these ‘force’ operations.

Totem pole or open-drain output circuit configurations can be selected using the WOR bit in the MDASMSCR register.

NOTE 1

If both channels are loaded with the same value, the output flip-flop provides a logic zero level output and the flag bit is still set on the match.

NOTE 2

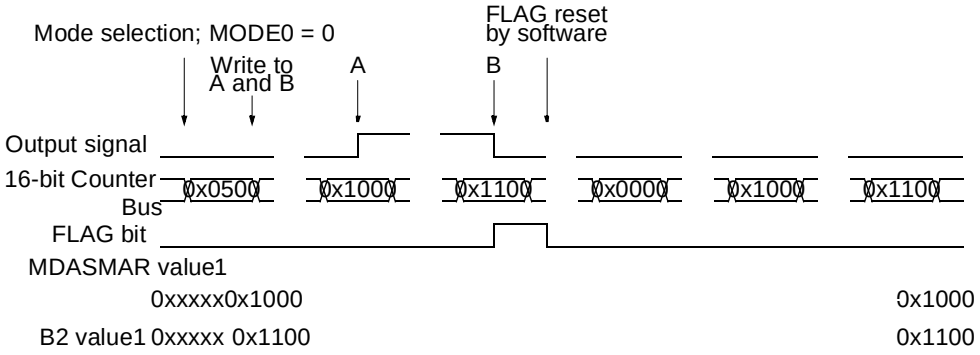
16-bit counter bus compare only occurs when the 16-bit counter bus is updated.

17.12.15.2 Single Shot Output Pulse Operation

The single shot output pulse operation is selected by writing the leading edge value of the desired pulse to data register A and the trailing edge value to data register B. A single pulse will be output at the desired time, thereby disabling the comparators until new values are written to the data registers. To generate a single shot output pulse, the OCB mode should be used to only generate a flag on the B match.

In this mode, registers A and B2 are accessible to the user software (at consecutive addresses).

Figure 17-17 provides an example of how the MDASM can be used to generate a single output pulse.



Note: 1. These values are accessible to the software.

Figure 17-17 Single Shot Output Pulse Example

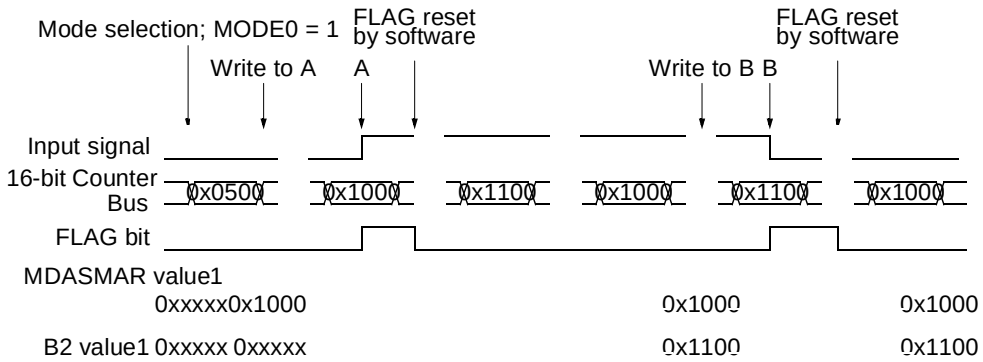
17.12.15.3 Single Output Compare operation

The single output compare operation is selected by writing to only one of the two data registers (A or B), thus enabling only one of the comparators. Following the first successful match on the enabled channel, the output level is fixed and remains at the same level indefinitely with no further software intervention being required. To generate a single output compare, the OCAB mode should be used to generate a flag on both the A and the B match.

NOTE

In this mode, registers A and B2 are accessible to the user software (at consecutive addresses).

Figure 17-18 provides an example of how the MDASM can be used to perform a single output compare.



Note: 1. These values are accessible to the software.

Figure 17-18 Single Shot Output Transition Example

17.12.15.4 Output Port Bit operation

The output port bit operation is selected by leaving both channels disabled, (i.e., by writing to neither register A nor B). The EDPOL bit alone controls the output value. The same result can be achieved by keeping EDPOL at zero and using the FORCA and FORCB bits to obtain the desired output level.

17.12.15.5 Output Pulse Width Modulation (OPWM) mode

OPWM mode is selected by setting MODE[3:0] to 1xxx. The MODE[2:0] bits allow some of the comparator bits to be masked.

This mode allows pulse width modulated output waveforms to be generated, with eight selectable frequencies. Frequencies are only relevant as such if the counter bus is driven by a counter as a time reference. Both channels (A and B) are used to generate one PWM output signal on the MDASM pin.

Channel B is accessed via register B1. Register B2 is not accessible. Channels A and B define respectively the leading and trailing edges of the PWM output pulse. The value in register B1 is transferred to register B2 each time a match occurs on either channel A or B.

NOTE

A FORCA or FORCB does not cause a transfer from B1 to B2.

The value loaded in register A is compared with the value on the 16-bit counter bus each time the counter bus is updated. When a match on A occurs, the FLAG line is activated and the output flip-flop is set. The value loaded in register B2 is compared with the value on the 16-bit counter bus each time the counter bus is updated. When a match occurs on B, the output flip-flop is reset.

NOTE

If both channels are loaded with the same value, when a simultaneous match on A and B occurs, the submodule behaves as if a simple match on B had occurred except for the FLAG line which is activated. The output flip-flop is reset and the value in register B1 is transferred to register B2 on the match.

The polarity of the PWM output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the PIN bit.

If subsequent compares occur on channels A and B, the PWM pulses continue to be output, regardless of the state of the FLAG bit.

At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to a comparison on A or B respectively. Note that the FLAG line is not activated by the FORCA and FORCB operations.

WARNING

Data registers A and B must be loaded with the values needed to produce the desired PWM output pulse.

NOTE

16-bit counter bus compare only occurs when the 16-bit counter bus is updated.

Figure 17-19 provides an example of how the MDASM can be used for pulse width modulation.

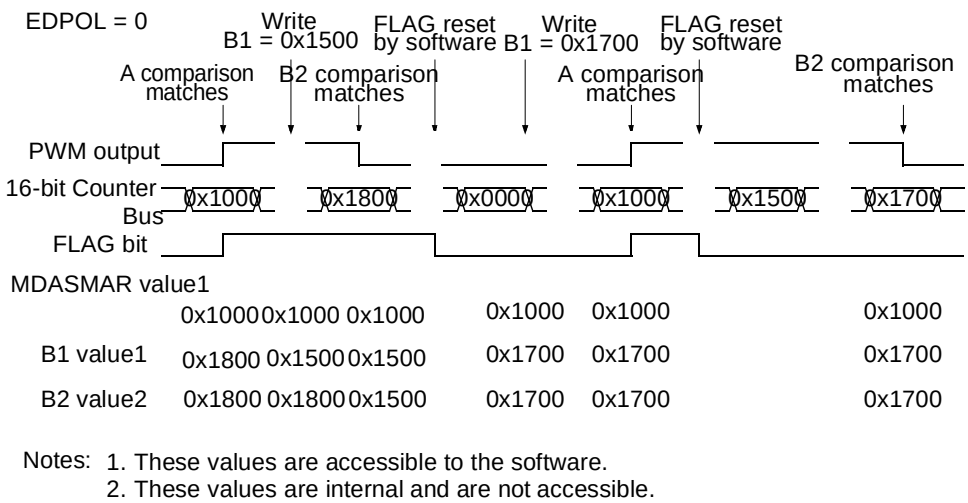


Figure 17-19 MDASM Output Pulse Width Modulation Example

To generate PWM output pulses of different frequencies, the 16-bit comparator can have some of its bits masked. This is controlled by bits MODE2, MODE1 and MODE0. The frequency of the PWM output (f_{PWM}) is given by the following equation (assuming the MDASM is connected to a 16-bit counter bus used as time reference and f_{SYS} is the frequency of the MIOS CLOCK):

$$f_{PWM} = \frac{f_{SYS}}{N_{MCPSM} \bullet N_{COUNTER} \bullet N_{MDASM}} \qquad [1]$$

where:

- N_{MCPSM} is the overall MCPSM clock divide ratio (2, 3, 4,...,16).
- $N_{COUNTER}$ is the divide ratio of the prescaler of the counter (used as a time reference) that drives the 16-bit counter bus.
- N_{MDASM} is the maximum count reachable by the counter when using n bits of resolution (this count is equal to 2^n).

A few examples of frequencies and resolutions that can be obtained are shown in **Table 17-25**.

Table 17-25 MDASM PWM Example Output Frequencies/Resolutions at $f_{[SYS]}$ = 40 MHz

Resolution (bits)	N_{MCPSM}	$N_{COUNTER}$	N_{MDASM}	PWM output frequency (Hz)[1]
16	16	256	65536	0.15
16	2	1	65536	305.17
15	16	256	32768	0.29
15	2	1	32768	610.35
14	16	256	16384	0.59
14	2	1	16384	1 220.70
13	16	256	8192	1.19
13	2	1	8192	2 441.41
12	16	256	4096	2.38
12	2	1	4096	4 882.81
11	16	256	2048	4.77
11	2	1	2048	9 765.63
9	16	256	512	19.07
9	2	1	512	39 062.50
7	16	256	128	76.29
7	2	1	128	156 250

NOTES:
1. This table is valid only if the MDASM is connected to an MMCSM operating as a free-running counter.

When using 16 bits of resolution on the comparator (MODE[2:0] = 000), the output can vary from a 0% duty cycle up to a duty cycle of 65535/65536. In this case it is not possible to have a 100% duty cycle. In cases where 16-bit resolution is not needed, it is possible to have a duty cycle ranging from 0% to 100%. Setting bit 15 of the value stored in register B to one results in the output being ‘always set’. Clearing bit 15 (to zero) allows normal comparisons to occur and the normal output waveform is obtained. Changes to and from the 100% duty cycle are done synchronously on an A or B match, as are all other width changes.

In the OPWM mode, the WOR bit selects whether the output is totem pole driven or open-drain.

17.12.16 Modular I/O Bus (MIOB) Interface

- The MDASM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MDASM registers, and to control the MDASM in the different possible situations.
- The MDASM is connected to four 16-bit counter buses available to that submod-

ule instance, so that the MDASM can select by software which one to use.

- The MDASM uses the request bus to transmit the FLAG line to the request submodule (RQSM).

17.12.17 Effect of RESET on MDASM

When the RESET signal is asserted, the MDASM registers are reset according to the values specified in **17.12.18 MDASM Registers**.

17.12.18 MDASM Registers

The privilege level to access the MDASM registers depends on the MIOS14MCR SUPV bit. The privilege level is unrestricted after reset and can be changed to supervisor by software.

17.12.18.1 MDASM Registers Organization

The MDASM register map comprises four 16-bit register locations. As shown in below, the register block contains four MDASM registers. Note that the MDASMSCRD is the duplication of the MDASMSCR. This is done to allow 32-bit aligned accesses.

WARNING

Do not write directly to the address of the MDASMSCRD. This register's address may be reserved for future use and should not be accessed by the software to ensure future software compatibility.

All unused bits return zero when read by the software. All register addresses in this section are specified as offsets from the base address of the MDASM.

Table 17-26 MDASM Address Map

Address	Register
MDASM11	
0x30 6058	MDASM11 Data A Register (MDASMAR) See **17.12.18.2 MDASM DataA (MDASMAR) Register Bits** for bit descriptions.
0x30 605A	MDASM11 Data B Register (MDASMBR) See **17.12.18.3 MDASM DataB (MDASMBR) Register Bits** for bit descriptions.
0x30 605C	MDASM11 Status/Control Register Duplicated (MDASMSCRD) See **Table 17-29** for bit descriptions.
0x30 605E	MDASM11 Status/Control Register (MDASMSCR) See **Table 17-29** for bit descriptions.
MDASM12	
0x30 6060	MDASM12 Data A Register (MDASMAR)
0x30 6062	MDASM12 Data B Register (MDASMBR)
0x30 6064	MDASM12 Status/Control Register Duplicated (MDASMSCRD)
0x30 6066	MDASM12 Status/Control Register (MDASMSCR)

Table 17-26 MDASM Address Map (Continued)

Address	Register
MDASM13	
0x30 6068	MDASM13 Data A Register (MDASMAR)
0x30 606A	MDASM13 Data B Register (MDASMBR)
0x30 606C	MDASM13 Status/Control Register Duplicated (MDASMSCRD)
0x30 606E	MDASM13 Status/Control Register (MDASMSCR)
MDASM14	
0x30 6070	MDASM14 Data A Register (MDASMAR)
0x30 6072	MDASM14 Data B Register (MDASMBR)
0x30 6074	MDASM14 Status/Control Register Duplicated (MDASMSCRD)
0x30 6076	MDASM14 Status/Control Register (MDASMSCR)
MDASM15	
0x30 6078	MDASM15 Data A Register (MDASMAR)
0x30 607A	MDASM15 Data B Register (MDASMBR)
0x30 607C	MDASM15 Status/Control Register Duplicated (MDASMSCRD)
0x30 607E	MDASM15 Status/Control Register (MDASMSCR)
MDASM27	
0x30 60D8	MDASM27 Data A Register (MDASMAR)
0x30 60DA	MDASM27 Data B Register (MDASMBR)
0x30 60DC	MDASM27 Status/Control Register Duplicated (MDASMSCRD)
0x30 60DE	MDASM27 Status/Control Register (MDASMSCR)
MDASM28	
0x30 60E0	MDASM28 Data A Register (MDASMAR)
0x30 60E2	MDASM28 Data B Register (MDASMBR)
0x30 60E4	MDASM28 Status/Control Register Duplicated (MDASMSCRD)
0x30 60E6	MDASM28 Status/Control Register (MDASMSCR)
MDASM29	
0x30 60E8	MDASM29 Data A Register (MDASMAR)
0x30 60EA	MDASM29 Data B Register (MDASMBR)
0x30 60EC	MDASM29 Status/Control Register Duplicated (MDASMSCRD)
0x30 60EE	MDASM29 Status/Control Register (MDASMSCR)
MDASM30	
0x30 60F0	MDASM30 Data A Register (MDASMAR)
0x30 60F2	MDASM30 Data B Register (MDASMBR)
0x30 60F4	MDASM30 Status/Control Register Duplicated (MDASMSCRD)
0x30 60F6	MDASM30 Status/Control Register (MDASMSCR)
MDASM31	
0x30 60F8	MDASM31 Data A Register (MDASMAR)
0x30 60FA	MDASM31 Data B Register (MDASMBR)
0x30 60FC	MDASM31 Status/Control Register Duplicated (MDASMSCRD)
0x30 60FE	MDASM31 Status/Control Register (MDASMSCR)

17.12.18.2 MDASM DataA (MDASMAR) Register Bits

MDASMAR — MDASM DataA Register

0x30 6058
0x30 6060
0x30 6068
0x30 6070
0x30 6078
0x30 60D8
0x30 60E0
0x30 60E8
0x30 60F0
0x30 60F8

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-27 MDASMAR Bit Descriptions

Bit(s)	Name	Description
0:15	AR15 — AR0	MDASMAR is the data register associated with channel A; its use varies with the different modes of operation: In the DIS mode, MDASMAR can be accessed to prepare a value for a subsequent mode selection. In the IPWM mode, MDASMAR contains the captured value corresponding to the trailing edge of the measured pulse. In the IPM and IC modes, MDASMAR contains the captured value corresponding to the most recently detected dedicated edge (rising or falling edge). In the OCB and OCAB modes, MDASMAR is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to MDASMAR in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison. In the OPWM mode, MDASMAR is loaded with the value corresponding to the leading edge of the PWM pulse to be generated. **NOTE:** In IC, IPM, or IPWM mode, when a read to register A or B occurs at the same time as a counter bus capture into that register and the counter bus is changing value, then the counter bus capture to that register is delayed.

17.12.18.3 MDASM DataB (MDASMBR) Register Bits

MDASMBR — MDASM DataB Register

0x30 605A
0x30 6062
0x30 606A
0x30 6072
0x30 607A
0x30 60DA
0x30 60E2
0x30 60EA
0x30 60F2
0x30 60FA

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-28 MDASMBR Bit Descriptions

Bit(s)	Name	Description
0:15	BR15 — BR0	MDASMBR is the data register associated with channel B; its use varies with the different modes of operation. Writing to register B always writes to B1 and, depending on the mode selected, sometimes to B2. Reading register B either reads B1 or B2 depending on the mode selected. In the DIS mode, MDASMBR can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value when register B1 is written. In the IPWM mode, MDASMBR contains the captured value corresponding to the leading edge of the measured pulse. In this mode, register B2 is accessed; buffer register B1 is hidden and is not readable. In the IPM and IC modes, MDASMBR contains the captured value corresponding to the previously dedicated edge (rising or falling edge). In this mode, register B2 is accessed; buffer register B1 is hidden and is not readable. In the OCB and OCAB modes, MDASMBR is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to MDASMBR in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed; buffer register B1 is hidden and is not readable. In the OPWM mode, MDASMBR is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed; buffer register B2 is hidden and cannot be accessed. NOTE In IC, IPM, or IPWM mode, when a read to register A or B occurs at the same time as a counter bus capture into that register and the counter bus is changing value, then the counter bus capture to that register is delayed.

17.12.19 MDASMSCRD — MDASM Status/Control Register (Duplicated)

The MDASMSCRD and the MDASMSCR are the same registers accessed at two different addresses.

Reading or writing to one of these two addresses has exactly the same effect.

WARNING

The user should not write directly to the address of the MDASMSCRD. This register's address may be reserved for future use and should not be accessed by the software to ensure future software compatibility.

The duplication of the SCR register allows coherent 32-bit accesses when using an RCPU.

17.12.20 MDASMSCR — MDASM Status/Control Register

The status and control register gathers a read only bit reflecting the status of the MDASM pin as well as read/write bits related to its control and configuration.

The pin input status bit reflects the status of the corresponding pin when in input mode. When in output mode, the PIN bit only reflects the status of the output flip-flop .

MDASMSCR — MDASM Status/Control Register

0x30 605E
0x30 6066
0x30 606E
0x30 6076
0x30 607E
0x30 60DE
0x30 60E6
0x30 60EE
0x30 60F6
0x30 60FE

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIN	WOR	FREN	RESERV	EDPOL	FORCA	FORCB	RESERVED		BSL1	BSL0	RESERV	MOD3	MOD2	MOD1	MOD0
SRESET:															
—	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-29 MDASMSCR Bit Descriptions

Bit(s)	Name	Description
0	PIN	
1	WOR	Wired-OR bit — In the DIS, IPWM, IPM and IC modes, the WOR bit is not used; reading this bit returns the value that was previously written. In the OCB, OCAB and OPWM modes, the WOR bit selects whether the output buffer is configured for open-drain or totem pole operation. 1 = Output buffer is open-drain. 0 = Output buffer is totem pole. The WOR bit is cleared by reset.
2	FREN	Freeze enable bit — This active high read/write control bit enables the MDASM to recognize the MIOB freeze signal. 1 = The MDASM is frozen if the MIOB freeze line is active. 0 = The MDASM is not frozen even if the MIOB freeze line is active. The FREN is cleared by reset.
3	—	Reserved
4	EDPOL	Polarity bit — In the DIS mode, this bit is not used; reading it returns the last value written. In the IPWM mode, this bit is used to select the capture edge sensitivity of channels A and B. 1 = Channel A captures on a falling edge. Channel B captures on a rising edge. 0 = Channel A captures on a rising edge. Channel B captures on a falling edge. In the IPM and IC modes, the EDPOL bit is used to select the input capture edge sensitivity of channel A. 1 = Channel A captures on a falling edge. 0 = Channel A captures on a rising edge. In the OCB, OCAB and OPWM modes, the EDPOL bit is used to select the voltage level on the output pin. 1 = The complement of the output flip-flop logic level appears on the output pin: a match on channel A resets the output pin; a match on channel B sets the output pin. 0 = The output flip-flop logic level appears on the output pin: a match on channel A sets the output pin, a match on channel B resets the output pin. The EDPOL bit is cleared by reset.
5	FORCA	Force A bit — In the OCB, OCAB and OPWM modes, the FORCA bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel A (except that the FLAG line is not activated). Writing a one to FORCA sets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, the FORCA bit is not used and writing to it has no effect. FORCA is cleared by reset and is always read as zero. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.
6	FORCB	Force B bit — In the OCB, OCAB and OPWM modes, the FORCB bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel B (except that the FLAG line is not activated). Writing a one to FORCB resets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, the FORCB bit is not used and writing to it has no effect. FORCB is cleared by reset and is always read as zero. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop.
7:8	—	Reserved
9:10	BSL1 — BSL0	Bus select bits — These bits are used to select which of the four 16-bit counter buses is used by the MDASM. **NOTE:** Unconnected counter buses inputs are grounded.

Table 17-29 MDASMSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
11	—	Reserved
12:15	MOD3 — MOD0	Mode select bits — The four mode select bits select the mode of operation of the MDASM. To avoid spurious interrupts, it is recommended that MDASM interrupts are disabled before changing the operating mode. The mode select bits are cleared by reset. **NOTE:** The reserved modes should not be set; if these modes are set, the MDASM behavior is undefined.

17.13 Pulse Width Modulation Submodule (MPWMSM)

In this section the values taken by the bits in the registers are given according to the following rule:

- 0 -> reset (negated))
- 1 -> set (asserted)
- u -> undefined
- - -> unaffected

17.13.1 MPWMSM Overview Description

The MIOS pulse width modulation submodule (MPWMSM) is a function included in the MIOS library. It allows pulse width modulated signals to be generated over a wide range of frequencies, independently of other MIOS output signals and with no software intervention. The output pulse width can vary from 0% to 100%. The minimum pulse width is twice the minimum MIOS CLOCK period (i.e., the minimum pulse width is 50 ns when f_{SYS} is 40 MHz). The MWPMSM can run in a double-buffered mode, to avoid spurious update.

The following sections describe the MPWMSM in detail. A block diagram of the MPWMSM is shown in **Figure 17-20**.

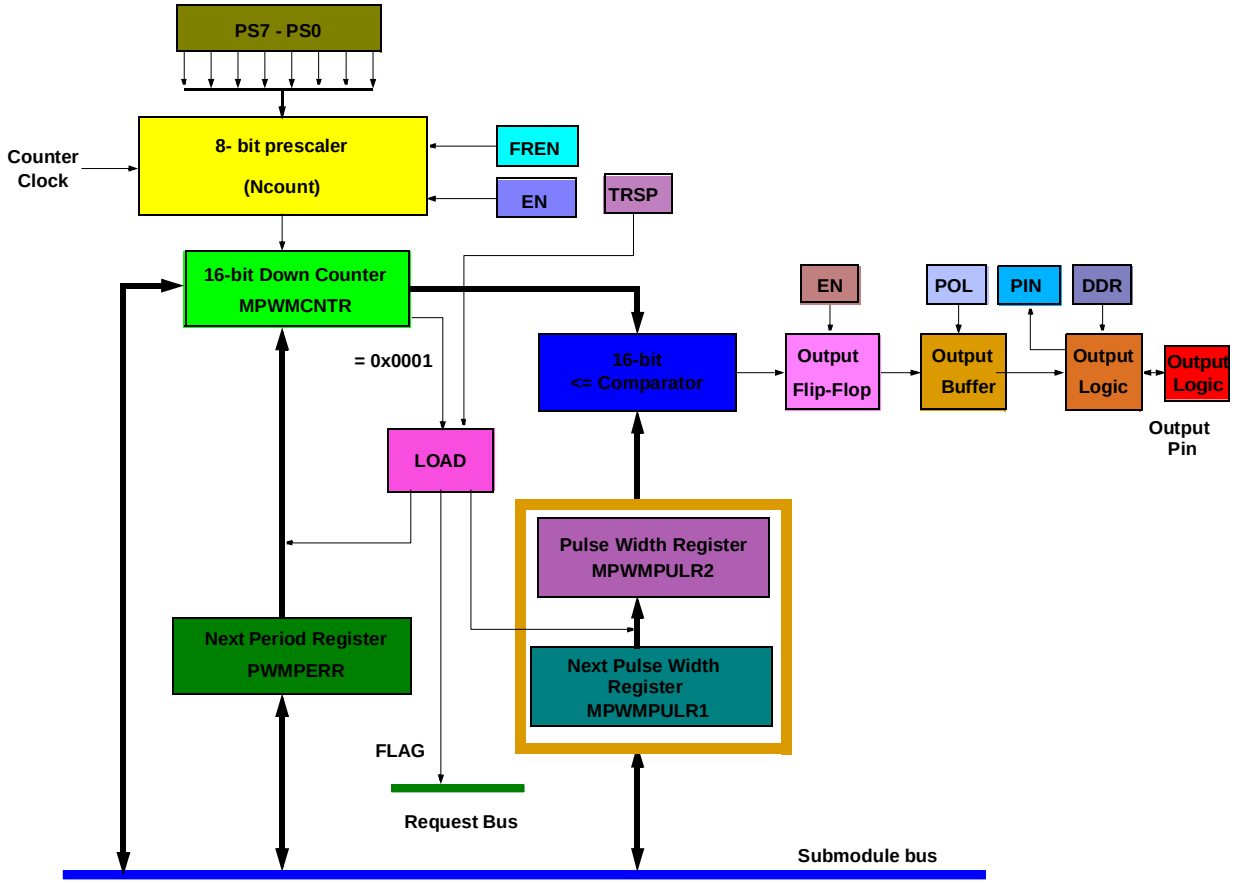


Figure 17-20 MPWMSM Block Diagram

17.13.2 MPWMSM Terminology

Bits of resolution — The MPWMSM contains a 16-bit modulus down-counter that counts from the desired loaded value to 0x0001. The term “bits of resolution” is used in this document to indicate the size of the equivalent free running binary counter. To cover the worst case, the number of bits is rounded to the lower number. For example, if the counter is preset with a value between 128 and 255, it is said to have seven bits of resolution. If it is preset with a value between 256 and 511, it is said to have eight bits of resolution, and so on.

Resolution — The term “resolution” is used in this document to define the minimum MPWMSM output increment in time units.

17.13.2.1 MPWMSM Features

— Output pulse width modulated (PWM) signal generation with no software intervention
— Built-in 8-bit programmable prescaler clocked by the MCPSM
— PWM period and pulse width values provided by software:

- Double-buffered for glitch-free period and pulse width changes
- Minimum output period/pulse-width increment: 50 ns (assuming f_{SYS} = 40 MHz)
- Maximum 50% duty-cycle output frequency: 10 MHz (assuming f_{SYS} = 40 MHz)
- Up to 16 bits of resolution for the output pulse width
- Wide range of periods
  - — 16 bits of resolution: period range from 3.27 ms (with 50-ns steps) to 6.71 s (with 102.4 µs steps)
  - — Eight bits of resolution: period range from 12.8 µs (with 50-ns steps) to 26.2 ms (with 102.4-µs steps)
- Wide range of frequencies
  - — Maximum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-2 prescaler selection: 305 Hz (3.27 ms.)
  - — Minimum output frequency at f_{SYS} = 40 MHz with 16 bits of resolution and divide-by-4096 prescaler selection: 0.15 Hz (6. 7s.)
  - — Maximum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-2 prescaler selection: 78125 Hz (12.8 µs.)
  - — Minimum output frequency at f_{SYS} = 40 MHz with 8 bits of resolution and divide-by-4096 prescaler selection: 38.15 Hz (26. 2ms.)

— Programmable duty cycle from 0% to 100%
— Possible interrupt generation at start of every period
— Software selectable output pulse polarity
— Software readable output pin status
— Possible use of pin as I/O port when PWM function is not needed

17.13.3 MPWMSM Description

The purpose of the MPWMSM is to create a variable pulse width output signal at a wide range of frequencies, independently of other MIOS output signals. The MPWMSM includes its own counter, and thus does not use the MIOS counter bus set. However the MPWMSM uses the prescaled clock bus that originates in the MIOS counter prescaler submodule (MCPSM) (See **17.13.3 MPWMSM Description**). The MPWMSM pulse width can vary from 0.0 percent to 100.0 percent, with up to 16 bits of resolution. The finest output resolution is the MIOS CLOCK period multiplied by two (for a MIOS CLOCK with f_{SYS} = 40 MHz, the finest output pulse width resolution is 50 ns). With the full 16 bits of resolution and the MCPSM set to divide by two, the period of the output signal can range from 3.276 ms to 6.7 1s (assuming f_{SYS} = 40 MHz).

By reducing the amount of bits of resolution, the output signal period can be reduced. For example, the period can be as fast as 204.8 µs (4882 Hz) with 12 bits of resolution, as fast as 12.8 µs (78.125 KHz) with eight bits of resolution, and as fast as 3.2 µs (312.5 KHz) with six bits of resolution (still assuming a f_{SYS} = 40 MHz and the MCPSM set to divide by two).

The MPWMSM is composed of:

- An output flip-flop with output buffer and polarity control
- An input/output pin with data direction control
- An 8-bit prescaler and clock selection logic
- A 16-bit down-counter (MPWMCNTR)
- A register to hold the next period values (MPWMPERR)
- Two registers to hold the current and next pulse width values (MPWMPULR)
- A less-than or equal comparator
- A status and control register (MPWMSCR)

17.13.3.1 Clock Selection

The MPWMSM contains an 8-bit prescaler clocked by the output signal from the MIOS counter prescaler submodule ($f_{SYS}/2$ to $f_{SYS}/16$). The MPWMSM clock selector allows the choice, by software, of one of 256 divide ratios which give to the MPWMSM a large choice of frequencies available for the down-counter. The MPWMSM down-counter is thus capable of counting with a clock frequency ranging from $f_{SYS}/2$ to $f_{SYS}/4096$.

Switching the MPWMSM from disable to enable will reload the value of CP7-CP0 into the 8-bit prescaler counter.

17.13.3.2 Counter

A 16-bit down-counter in the MPWMSM provides the time reference for the output signal. The counter is software writable. When writing to the counter (i.e., at the MPWMCNTR address), it also writes to the MPWMPERR register. When in transparent mode (TRSP = 1), writing to the MPWMPERR will also write to the counter. The down-counter is readable at anytime. The value loaded in the down-counter corresponds to the period of the output signal.

When the MPWMSM is enabled, the counter begins counting. As long as it is enabled, the counter counts down freely. The counter counts at the rate established by the prescaler. When the count down reaches 0x0001, the load operation is executed and the value in the MPWMPERR register is loaded in the MPWMCNTR register, (i.e., the counter). Then the counter restarts to count down from that value.

17.13.3.3 Period Register

The period section is composed of a 16-bit data register (MPWMPERR). The software establishes the period of the output signal in register MPWMPERR.

When the MPWMSM is running in transparent mode, the period value in register MPWMPERR is immediately transferred to the counter on a write to the MPWMPERR.

When the MPWMSM is running in double-buffered mode, the period value in register MPWMPERR can be changed at any time without affecting the current period of the

output signal. The new value MPWMPERR will be transferred to the counter only when the counter reaches the value of 0x0001 and generates a load signal.

Period values of 0x0000, 0x0001 and 0x0002 are MPWMSM special cases:

- The value 0x0000 in the period register, causes the counter to act like a free running counter. This condition creates a period of 65536 PWM clock periods.
- The value 0x0001 in the period register will always cause a period match to occur and the counter will never decrement below 0x0001. This condition is defined as a period of “1” PWM clock count. The output flip-flop is always set unless the value in the MPWMPULR = 0x0000, when the output flip-flop is always reset. Refer to **17.13.3.5 0% and 100% Duty Cycles** for details about 0% and 100% duty cycles.
- Writing value 0x0002 in the period register causes a period match to occur every two clock periods. The counter decrements from 0x0002 to 0x0001, and then it is initialized back to 0x0002. This condition is defined as a period of “2” clock count. Note that the value 0x0002 loaded in the period register and a value of 0x0001 in the pulse width register is the condition to obtain the maximum possible output frequency for a given clock period.

The relationship between the output frequency obtained (F_{PWMO}) and the MIOS CLOCK frequency (f_{SYS}), the MCPSM clock divide ratio (N_{MCPSM}), the counter divide ratio (N_{MPWMSM}) and the value loaded in the counter ($V_{COUNTER}$) is given by the following equation:

$$F_{PWMO} = \frac{F_{SYS}}{N_{MCPSM} \bullet N_{MPWMSM} \bullet V_{COUNTER}}$$

17.13.3.4 Pulse Width Registers

The pulse width section is composed of two 16-bit data registers (MPWMPULR1 and MPWMPULR2). Only register MPWMPULR1 is accessible by software. The software establishes the pulse width of the MPWMSM output signal in register MPWMPULR1. Register MPWMPULR2 is used as a double buffer of register MPWMPULR1.

When the MPWMSM is running in transparent mode, the pulse width value in register MPWMPULR1 is immediately transferred in the register MPWMPULR2 so that the new value takes effect immediately.

NOTE

When the MPWMSM is in disable mode, writing to MPWMPULR1 will write automatically to MPWMPULR2.

When the MPWMSM is not running in double-buffered mode, the pulse width value in register MPWMPULR1 can be changed at any time without affecting the current pulse width of the output signal. The new value MPWMPULR1 will be transferred to MPWMPULR2 only when the down-counter reaches the value of 0x0001.

When the counter first reaches the value in register MPWMPULR2, the output flip-flop is set. The output is reset when the counter reaches 0x0001. The pulse width match starts the width of the output signal, it does not affect the counter. Register MPWMPULR1 is software readable and writable at any time. The MPWMSM does not modify the content of register MPWMPULR1.

The PWM output pulse width can be as wide as one period minus one MPWMSM clock count: (i.e., MPWMPULR2 = MPWMPERR — [one MPWMSM clock count]). At the other end of the pulse width range, register MPWMPULR2 can contain 0x0001 to create a pulse width of one PWM clock count.

EXAMPLE

With 0x00FF in the counter and 0x0002 in MPWMPULR2, the period is 255 PWM clock count and the pulse width is two PWM clock counts.

For a given system clock frequency, with a given counter divide ratio and clock selection divide ratio, the output pulse width is given by the following equation:

$$\text{Pulse_Width} = \frac{N_{MCPSM} \bullet N_{MPWMSM} \bullet V_{MPWMB2}}{F_{SYS}}$$

where V_{MPWMB2} is the value in the register B2

In such conditions, the minimum output pulse width that can be obtained is given by:

$$\text{Minimum_Pulse_Width} = \frac{N_{MCPSM} \bullet N_{MPWMSM}}{F_{SYS}}$$

and the maximum pulse width by:

$$\text{Maximum_Pulse_Width} = \frac{N_{MCPSM} \bullet N_{MPWMSM} \bullet (2^{\text{Bit_of_Resolution}} - 1)}{F_{SYS}}$$

17.13.3.5 0% and 100% Duty Cycles

The 0% and 100% duty cycles are special cases to give flexibility to the software to create a full range of outputs. The “always set” and “always clear” conditions of the output flip-flop are established by the value in register MPWMPULR2. These boundary conditions are generated by software, just like another pulse. When the PWM output is being used to generate an analog level, the 0% and 100% represent the full scale values.

The 0% output is created with a 0x0000 in register MPWMPULR2, which prevents the output flip-flop from ever being set.

The 100% output is created when the content of register MPWMPULR2 is equal to or greater than the content of register MPWMPERR. Thus, the width register match

occurs on counter reload. The state sequencer provides the timing to ensure that the first appearance of a 100% value in register MPWMPULR2 causes a glitchless always-set condition of the output flip-flop when TRSP = ‘0’.

NOTE 1

Even if the output is forced to 100%, the 16-bit up counter continues its counting and that output changes to or from the 100% value are done synchronously to the selected period.

NOTE 2

When a PWM output period is selected to be 65536 PWM clocks by loading 0x0000 in the period register, it is not possible to have an 100% duty cycle output signal. In this case, the maximum duty cycle available is of 65535/65536.

17.13.3.6 Tables

Table 17-30 and **Table 17-31** summarize the frequency and minimum pulse width values that can be obtained respectively with divide-by-1 and divide-by-256 MPWMSM clock prescaler options, while using a MIOS CLOCK frequency of 40 MHz, and for each MCPSM clock divide ratios.

Table 17-30 PWM Pulse/Frequency Ranges (in Hz) Using /1 Option (40 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
50 ns/2	305	610	1220	2441	4882	9765	19.5K	39 K	78 K	156 K	312 K	625 K	1250 K	2500 K	5000 K	10000 K
75 ns/3	203	407	814	1628	3255	6510	13K	26 K	52 K	104 K	208 K	416 K	833 K	1666 K	3333 K	6666 K
100 ns/4	152	305	610	1220	2441	4882	9765	19.5 K	39 K	78 K	156 K	312 K	625 K	1250 K	2500 K	5000K
125 ns/5	122	244	488	976	1953	3906	7812	15.6 K	31.3 K	62.5 K	125 K	250 K	500 K	1000 K	2000 K	4000K
150 ns/6	101	203	407	814	1628	3255	6510	13 K	26 K	52 K	104 K	208 K	416 K	833 K	1666 K	3333K
175 ns/7	87.2	174	348	697	1395	2790	5580	11.1 K	22.3 K	44.6 K	89.3 K	178 K	357 K	714 K	1428 K	2857K
200 ns/8	76.3	152	305	610	1220	2441	4882	9765	19.5 K	39 K	78 K	156 K	312 K	625 K	1250 K	2500K
225 ns/9	67.8	135	271	542	1085	2170	4340	8680	17.3 K	34.7 K	69.4 K	138 K	277 K	555 K	1111 K	2222K
250 ns/10	61	122	244	488	976	1953	3906	7812	15.6 K	31.3 K	62.5 K	125 K	250 K	500 K	1000 K	2000K
275 ns/11	55.5	111	222	443	887	1775	3551	7102	14.2 K	28.4 K	56.8 K	113 K	227 K	454 K	909 K	1818 K
300 ns/12	50.8	101	203	407	814	1628	3255	6510	13 K	26 K	52 K	104 K	208 K	416 K	833 K	1666 K
325 ns/13	46.9	93.9	187	375	751	1502	3004	6009	12 K	24 K	48 K	96.1 K	192 K	384 K	769 K	1538 K
350 ns/14	43.6	87.2	174	348	697	1395	2790	5580	11.1 K	22.3 K	44.6 K	89.3 K	178 K	357 K	714 K	1428K
375 ns/15	40.7	81.4	162	325	651	1302	2604	5208	10.4 K	20.8 K	41.6 K	83.3 K	166 K	333 K	666 K	1333 K
400 ns/16	38.1	76.3	152	305	610	1220	2441	4882	9765	19.5 K	39 K	78 K	156 K	312 K	625 K	1250 K

17.13.3.7 MPWMSM Status and Control Register (SCR)

One register is used to initialize the MPWMSM and monitor its operation. Control bits are included to allow the software to enable the PWM generator, establish the output signal polarity, select the counter clock rate and set the glitch-free mode. A status bit is included to allow the software to read the state of the output pin.

Table 17-31 PWM Pulse/frequency Ranges (in Hz) Using /256 Option (40 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
12.8 µs/512	1.192	2.384	4.768	9.536	19.07	38.14	76.29	152.5	305.1	610.3	1220	2441	4882	9765	19.5 K	39 K
19.2 µs/768	0.794	1.589	3.178	6.357	12.71	25.43	50.86	101.7	203.4	406.9	813.8	1627	3255	6510	13 K	26 K
25.6 µs/1024	0.596	1.192	2.384	4.768	9.536	19.07	38.14	76.29	152.5	305.1	610.3	1220	2441	4882	9765	19.5 K
32 µs/1280	0.476	0.953	1.907	3.814	7.629	15.24	30.51	61.03	122	244.1	488.2	976.5	1953	3906	7812	15.6 K
38.4 µs/1536	0.397	0.795	1.589	3.179	6.358	12.71	25.43	50.86	101.7	203.5	406.9	813.8	1627	3255	6510	13 K
44.8 µs/1792	0.34	0.681	1.362	2.724	5.449	10.89	21.80	43.59	87.19	174.4	348.8	697.5	1395	2790	5580	11.1 K
51.2 µs/2048	0.298	0.596	1.192	2.384	4.768	9.536	19.07	38.14	76.29	152.5	305.1	610.3	1220	2441	4882	9765
57.6 µs/2304	0.264	0.529	1.059	2.119	4.238	8.477	16.95	33.90	67.81	135.6	271.2	542.5	1085	2170	4340	8680
64 µs/2560	0.238	0.476	0.953	1.907	3.814	7.629	15.24	30.51	61.03	122	244.1	488.2	976.5	1953	3906	7812
70.4 µs/2816	0.216	0.433	0.867	1.734	3.468	6.936	13.87	27.74	55.48	110.9	221.9	443.9	887.8	1775	3551	7102
76.8 µs/3072	0.198	0.397	0.795	1.589	3.179	6.358	12.71	25.43	50.86	101.7	203.5	406.9	813.8	1627	3255	6510
83.2 µs/3328	0.183	0.366	0.733	1.467	2.934	5.869	11.74	23.47	46.95	93.9	187.8	375.6	751.2	1502	3004	6009
89.6 µs/3584	0.170	0.340	0.681	1.362	2.724	5.449	10.89	21.80	43.59	87.19	174.4	348.8	697.5	1395	2790	5580
96 µs/3840	0.159	0.318	0.636	1.271	2.543	5.086	10.17	20.34	40.69	81.38	162.8	325.5	651	1302	2604	5208

17.13.3.8 MPWMSM Interrupt

A valid MPWMSM interrupt is recognized when a pulse occurs on the flag line to set the flag bit and the interrupt enable bit is set for the corresponding level in the request submodule (Refer to **17.9.1 Interrupts** for details about interrupts). A set flag pulse is generated at the start of every period.

The flag bit is a status bit which indicates, when set, that the output period has started and that registers MPWMPERR and MPWMPULR1 are available for updates when in double-buffered mode. The level of the resulting interrupt is determined in the request submodule.

17.13.3.9 MPWMSM Port Functions

The MPWMSM has one dedicated I/O external pin.

The output flip-flop is the basic output of the MPWMSM. Except when the pulse width is at 100% or 0%, the output flip-flop is reset at the beginning of each period and is set at the beginning of the designated pulse width until the end of the period. As a software option, the polarity of the signal presented to the output pin may be the state of the output flip-flop or the inverse of the output flip-flop.

The MPWMSM is connected to an external, input/output pin. When in the disabled mode, the POL bit (polarity) and the DDR bit (data direction) in the SCR register allow the MPWMSM to be used as an I/O port.

17.13.3.10 MPWMSM Data Coherency

Byte accesses to MPWMPULR and MPWMPERR are supported, but are not recommended as the transfer from the primary registers to the secondary registers are done as a 16-bit word transfer.

For most MPWMSM operations, 16-bit accesses are sufficient and long word accesses (32-bit) are treated as two 16-bit accesses, with one exception — a long word write to the period/pulse width registers. In this case, if the long word write takes place within the PWM period, there is no visible effect on the output signal and the new values stored in MPWMPERR and MPWMPULR are ready to be loaded into the buffer registers at the start of the next period. If, however, the long word write coincides with the end of the period, then the transfer of values from the primary to the secondary registers is delayed until the end of the next period; during this period the previous values are used for the period and width This feature enables updates of the period and pulse-width values without getting erroneous pulses.

17.13.4 MIO Bus (MIOB) Interface

- The MPWMSM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MPWMSM registers, and to control the MPWMSM in the different possible situations.
- The MPWMSM is not using any of the 16-bit counter buses
- The MPWMSM uses the request bus to transmit to the request submodule

17.13.5 Effect of RESET on MPWMSM

The MPWMSM is affected by reset according to what is described in the section related to register description.

The MPWMPERR, MPWMPULR, and MPWMCNTR registers, together with the clock prescaler register bits, must be initialized by software, since they are undefined after hardware reset.

A value must be written to the MPWMCNTR (which writes the same value into the MPWMPERR) and a pulse width value written to MPWMPULR, before the MPWMSCR is written to. The latter access initializes the clock prescaler.

17.13.6 MPWMSM Registers

The privilege level to access to the MPWMSM registers depends on the MIOS14MCR SUPV bit. The privilege level is unrestricted after reset and can be change to supervisor by software.

17.13.6.1 MPWMSM Registers Organization

The MPWMSM register map comprises four 16-bit register locations, as shown below. All unused bits return zero when read by the software. All register addresses in this section are specified as offsets from the base address of the MPWMSM.

Table 17-32 MPWMSM Address Map

Address	Register
MPWMSM0	
0x30 6000	MPWMSM0 Period Register (MPWMSMPERR) See **Table 17-33** for bit descriptions.
0x30 6002	MPWMSM0 Pulse Register (MPWMSMPULR) See **Table 17-34** for bit descriptions.
0x30 6004	MPWMSM0 Count Register (MPWMSMCNTR) See **Table 17-35** for bit descriptions.
0x30 6006	MPWMSM0 Status/Control Register (MPWMSMSCR) See **Table 17-36** for bit descriptions.
MPWMSM1	
0x30 6008	MPWMSM1 Period Register (MPWMSMPERR)
0x30 600A	MPWMSM1 Pulse Register (MPWMSMPULR)
0x30 600C	MPWMSM1 Count Register (MPWMSMCNTR)
0x30 600E	MPWMSM1 Status/Control Register (MPWMSMSCR)
MPWMSM2	
0x30 6010	MPWMSM2 Period Register (MPWMSMPERR)
0x30 6012	MPWMSM2 Pulse Register (MPWMSMPULR)
0x30 6014	MPWMSM2 Count Register (MPWMSMCNTR)
0x30 6016	MPWMSM2 Status/Control Register (MPWMSMSCR)
MPWMSM3	
0x30 6018	MPWMSM3 Period Register (MPWMSMPERR)
0x30 601A	MPWMSM3 Pulse Register (MPWMSMPULR)
0x30 601C	MPWMSM3 Count Register (MPWMSMCNTR)
0x30 601E	MPWMSM3 Status/Control Register (MPWMSMSCR)
MPWMSM4	
0x30 6020	MPWMSM4 Period Register (MPWMSMPERR)
0x30 6022	MPWMSM4 Pulse Register (MPWMSMPULR)
0x30 6024	MPWMSM4 Count Register (MPWMSMCNTR)
0x30 6026	MPWMSM4 Status/Control Register (MPWMSMSCR)
MPWMSM5	
0x30 6028	MPWMSM5 Period Register (MPWMSMPERR)
0x30 602A	MPWMSM5 Pulse Register (MPWMSMPULR)
0x30 602C	MPWMSM5 Count Register (MPWMSMCNTR)
0x30 602E	MPWMSM5 Status/Control Register (MPWMSMSCR)
MPWMSM16	
0x30 6080	MPWMSM16 Period Register (MPWMSMPERR)
0x30 6082	MPWMSM16 Pulse Register (MPWMSMPULR)
0x30 6084	MPWMSM16 Count Register (MPWMSMCNTR)
0x30 6086	MPWMSM16 Status/Control Register (MPWMSMSCR)

Table 17-32 MPWMSM Address Map (Continued)

Address	Register
MPWMSM17	
0x30 6088	MPWMSM17 Period Register (MPWMSMPERR)
0x30 608A	MPWMSM17 Pulse Register (MPWMSMPULR)
0x30 608C	MPWMSM17 Count Register (MPWMSMCNTR)
0x30 608E	MPWMSM17 Status/Control Register (MPWMSMSCR)
MPWMSM18	
0x30 6090	MPWMSM18 Period Register (MPWMSMPERR)
0x30 6092	MPWMSM18 Pulse Register (MPWMSMPULR)
0x30 6094	MPWMSM18 Count Register (MPWMSMCNTR)
0x30 6096	MPWMSM18 Status/Control Register (MPWMSMSCR)
MPWMSM19	
0x30 6098	MPWMSM19 Period Register (MPWMSMPERR)
0x30 609A	MPWMSM19 Pulse Register (MPWMSMPULR)
0x30 609C	MPWMSM19 Count Register (MPWMSMCNTR)
0x30 609E	MPWMSM19 Status/Control Register (MPWMSMSCR)
MPWMSM20	
0x30 60A0	MPWMSM20 Period Register (MPWMSMPERR)
0x30 60A2	MPWMSM20 Pulse Register (MPWMSMPULR)
0x30 60A4	MPWMSM20 Count Register (MPWMSMCNTR)
0x30 60A6	MPWMSM20 Status/Control Register (MPWMSMSCR)
MPWMSM21	
0x30 60A8	MPWMSM21 Period Register (MPWMSMPERR)
0x30 60AA	MPWMSM21 Pulse Register (MPWMSMPULR)
0x30 60AC	MPWMSM21 Count Register (MPWMSMCNTR)
0x30 60AE	MPWMSM21 Status/Control Register (MPWMSMSCR)

17.13.6.2 MPWMPERR — MPWMSM Period Register

MPWMPERR address: MPWMSM base address

The period register contains the binary value corresponding to the period to be generated.

MPWMPERR — MPWMSM Period Register

0x30 6000
0x30 6008
0x30 6010
0x30 6018
0x30 6020
0x30 6028
0x30 6080
0x30 6088
0x30 6090
0x30 6098
0x30 60A0
0x30 60A8

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PER[15:0]															
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-33 MPWMPERR Bit Descriptions

Bit(s)	Name	Description
0:15	PER15 — PER0	Period. These bits contain the binary value corresponding to the period to be generated.

17.13.6.3 MPWMPULR — MPWMSM Pulse Width Register

The pulse width register contains the binary value of the pulse width to be generated.

MPWMPULR — MPWMSM Pulse Width Register

0x30 6002
0x30 600A
0x30 6012
0x30 601A
0x30 6022
0x30 602A
0x30 6082
0x30 608A
0x30 6092
0x30 609A
0x30 60A2
0x30 60AA

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PUL15	PUL14	PUL13	PUL12	PUL11	PUL10	PUL9	PUL8	PUL7	PUL6	PUL5	PUL4	PUL3	PUL2	PUL1	PUL0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-34 MPWMPULR Bit Descriptions

Bit(s)	Name	Description
0:15	PUL15 — PUL0	Pulse width. These bits contain the binary value of the pulse width to be generated.

17.13.6.4 MPWMCNTR — MPWMSM Counter Register

The counter register reflects the actual value of the MPWMSM counter.

This register is writable only through the period register (PWMPERR). Writes to the counter register will write the same value to the period register.

MPWMCNTR — MPWMSM Counter Register

0x30 6004
0x30 600C
0x30 6014
0x30 601C
0x30 6024
0x30 602C
0x30 6084
0x30 608C
0x30 6094
0x30 609C
0x30 60A4
0x30 60AC

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CNT[15:0]															
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-35 MPWMCNTR Bit Descriptions

Bit(s)	Name	Description
0:15	CNT[15:0]	Counter. These bits reflect the actual value of the MPWMSM counter.

17.13.6.5 MPWMSCR — MPWMSM Status/Control Register

The status and control register gathers read only bits reflecting the status of the MPWMSM pin as well as read/write bits related to its control and configuration.

MPWMSCR — MPWMSM Status/Control Register

0x30 6006
0x30 600E
0x30 6016
0x30 601E
0x30 6026
0x30 602E
0x30 6086
0x30 608E
0x30 6096
0x30 609E
0x30 60A6
0x30 60AE

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PIN	DDR	FREN	TRSP	POL	EN	RESERVED		CP7	CP6	CP5	CP4	CP3	CP2	CP1	CP0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-36 MPWMSCR Bit Descriptions

Bit(s)	Name	Description
0	PIN	Pin input status bit — The PIN bit reflects the state present on the MPWMSM pin. The software can thus monitor the signal on the pin. The PIN bit is a read-only bit. Writing to the PIN bit has no effect.
1	DDR	Data direction register — The DDR bit indicates the direction for the pin when the PWM function is not used (disable mode). 0 = pin is in input. 1 = pin is in output. The DDR bit is cleared by reset. **Table 17-37** lists the different uses for the polarity (POL) bit, the enable (EN) bit and the data direction register (DDR) bit.
2	FREN	Freeze enable bit — This active high read/write control bit enables the MPWMSM to recognize the freeze signal on the MIOB. 0 = MPWMSM not frozen even if the MIOB freeze line is active. 1 = MPWMSM frozen if the MIOB freeze line is active. The FREN is cleared by reset.
3	TRSP	Transparent mode — The TRSP bit indicates that the MPWMSM is in transparent mode. In transparent mode, when the software writes to either the MPWMPERR or MPWMPULR1 register the value written is immediately transferred to the counter or register MPWMPULR2 respectively. 0 = Double-buffered mode. 1 = Transparent mode. The TRSP bit is cleared by reset.
4	POL	Output polarity control bit — The POL bit works in conjunction with the EN bit and controls whether the MPWMSM drives the pin with the direct or the inverted value of the output flip-flop. **Table 17-37** lists the different uses for the polarity (POL) bit, the enable (EN) bit and the data direction register (DDR) bit.

Table 17-36 MPWMSCR Bit Descriptions (Continued)

Bit(s)	Name	Description
5	EN	Enable PWM signal generation — The EN bit defines whether the MPWMSM generates a PWM signal or is used as an I/O channel: 0 = PWM generation disabled (pin can be used as I/O). 1 = PWM generation enabled (the pin is in output mode). Each time the submodule is enabled, the value of CP7-CP0 is loaded into the prescaler. The EN bit is cleared by reset.
6:7	—	Reserved
8:15	CP7 — CP0	Clock prescaler — This 8-bit read/write data register stores the modulus value for loading into the built-in 8-bit clock prescaler. The value loaded defines the divide ratio for the signal that clocks the MPWMSM. The new value is loaded into the prescaler counter on the prescaler counter overflow, or upon the EN bit of the MPWMSCR being set. **Table 17-38** gives the clock divide ratio according to the CP7..CP0 values:

Table 17-37 PWMSM Output Pin Polarity Selection

Control Bits			Pin Direction	Pin State	Periodic Edge	Variable Edge	Optional Interrupt On
POL	EN	DDR					
0	0	0	Input	INPUT	—	—	—
0	0	1	Output	Always Low	—	—	—
0	1	—	Output	High Pulse	Falling Edge	Rising Edge	Falling Edge
1	0	0	Input	INPUT	—	—	—
1	0	1	Output	Always High	—	—	—
1	1	—	Output	Low Pulse	Rising Edge	Falling Edge	Rising Edge

Table 17-38 Prescaler Values

Prescaler Value (CP7..CP0 in Hex)	MCPSM Divide Ratio:
FF	1
FE	2
FD	3
FC	4
FB	5
FA	6
F9	7
F8	8
.....	.....
02	254 (2^8 -2)
01	255 (2^8 -1)
00	256 (2^8)

17.14 Parallel Port I/O Submodule (MPIOSM)

In this section the values taken by the bits in the registers are given according to the following rule:

- 0 -> reset (negated))
- 1 -> set (asserted)
- u -> undefined
- - -> unaffected

17.14.1 MPIOSM Overview Description

The MIOS parallel port I/O submodule (MPIOSM) is a function included in the MIOS library in order to provide the required port I/O capability. The MPIOSM can operate without the involvement of other MIOS submodules.

Each implemented MPIOSM provides I/O capability for up to 16 pins. Any number of required MPIOSM could be implemented as long as the total number of submodules do not exceed the limit described in **17.1.4 MIOS14 Module/Submodule Addressing**.

The following sections describe the MPIOSM in detail. A block diagram of one bit of the MPIOSM is shown in **Figure 17-21**. The MPIOSM contains 16 such blocks.

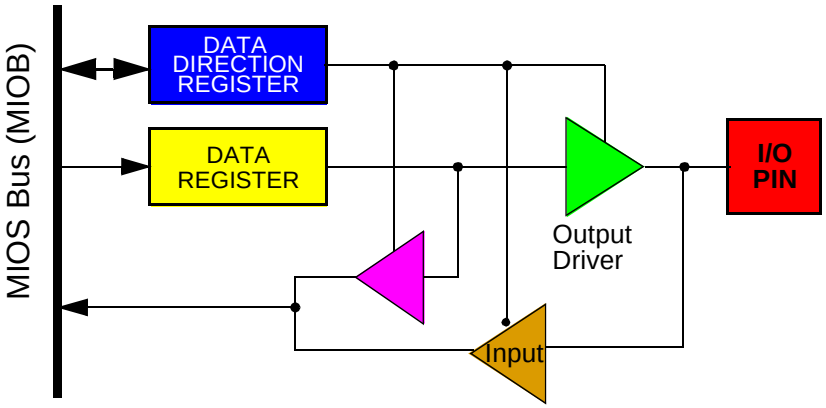


Figure 17-21 MPIOSM 1-Bit Block Diagram

17.14.2 MPIOSM features

- A submodule of the MIOS library
- Uses two 16-bit registers in the address space
- Up to 16 bidirectional parallel input/output pins
- Simple DDR (data direction register) concept for pin direction selection

17.14.3 MPIOSM Pin Functions

Table 17-39 shows the MPIOSM I/O pin functions according to the setting of the DDR when writing to or reading from the DR.

Table 17-39 MPIOSM I/O Pin Function

Operation Performed	DDR	I/O Pin Function
Write	0	The I/O pin is in input mode. Data is written into the DR.
Write	1	Data is written into the DR and output to the I/O pin.
Read	0	The I/O pin is in input mode. The state of the I/O pin is read.
Read	1	The I/O pin is in an output mode. The DR is read.

17.14.4 MPIOSM Description

17.14.4.1 MPIOSM Port function

A MIOS parallel port I/O submodule can handle up to 16 input/output pins. The number of I/O pins is determined at the time of silicon implementation.

The MPIOSM has two 16-bit registers: the data register (DR) and the data direction register (DDR). Each pin of the MPIOSM may be programmed as an input or an output, determined by the state of the corresponding bit in the DDR.

The data direction register can be written to or read by the processor. During the programmed output state, a read of the data register reads the value of the output data latch and not the I/O pin. See **Figure 17-21** and **Table 17-39**.

During reset, all MPIOSM pins are configured as inputs. The contents of the data register are undefined after reset.

As a general practice, it is recommended to write a value in the data register before configuring its corresponding I/O pin as an output.

17.14.4.2 Non-Bonded MPIOSM Pads

A non-bonded MPIOSM pad reads ‘0’ when it is configured as an input. When configured as an output, it indicates the current state of the output data latch.

17.14.5 Modular I/O Bus (MIOB) Interface

- The MPIOSM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MPIOSM registers, and to control the MPIOSM in the different possible situations.
- The MPIOSM does not use the counter bus set and is therefore not connected to it.
- The MPIOSM does not generate any interrupts and is therefore not connected to this bus.

17.14.6 Effect of RESET on MPIOSM

When the RESET signal is asserted, all the DDR bits are cleared. The data bits are undefined after reset.

17.14.7 MPIOSM Testing

No special test logic has been implemented in this submodule. To be flexible while selecting the number of implemented pins, the test patterns are implemented in a bit per bit modular fashion.

17.14.8 MPIOSM Registers

The privilege level to access to the MPIOSM registers depends on the MIOS14MCR SUPV bit. The privilege level is unrestricted after reset and can be change to supervisor by software.

17.14.9 MPIOSM Register Organization

MPIOSM— Register Organization

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0x30 6100	MPIOSM Data Register (MPIOSMDR)														
0x30 6102	MPIOSM Data Direction Register (MPIOSMDDR)														
0x30 6104	Reserved														
0x30 6106	Reserved														

17.14.9.1 MPIOSMDR — MPIOSM Data Register

MPIOSMDR — MPIOSM Data Register **0x30 6100**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
Data15	Data14	Data13	Data12	Data11	Data10	Data9	Data8	Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0
SRESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Table 17-40 MPIOSMDR Bit Descriptions

Bit(s)	Name	Description
0:15	DATA15 — DATA0	These bits are read/write data bits that define the value to be driven to the pad in output mode, for each implemented I/O pin of the MPIOSM

17.14.9.2 MPIOSMDDR — MPIOSM Data Direction Register

MPIOSMDDR — MPIOSM Data Direction Register **0x30 6102**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
DDR15	DDR14	DDR13	DDR12	DDR11	DDR10	DDR9	DDR8	DDR7	DDR6	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0
SRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 17-41 MPIOSMDDR Bit Descriptions

Bit(s)	Name	Description
0:15	DDR15 — DDR0	These bits are read/write data bits that define the data direction status for each implemented I/O pin of the MPIOSM 0 = corresponding pin is input. 1 = corresponding pin is output.

17.15 Real-Time Clock Submodule (MRTCSM)

In this section the values taken by the bits in the registers are given according to the following rule:

- 0 → reset (negated)

- 1 → set (asserted)
- u → undefined
- - → unaffected

17.15.1 MRTCSM Overview Description

The MIOS real-time clock submodule (MRTCSM) is a function included in the MIOS library. It is a software-programmable counter suitable for keeping track of the time of the day, maintaining calendar information or timestamping incoming system events.

The purpose of the MRTCSM is to provide a real time function independently of other MIOS submodules, which may be sustained on a separate standby power supply. The MRTCSM was designed to minimize current drained from battery when in standby.

This time counter is driven by a dedicated 32.768-KHz low-power oscillator. The core of the MRTCSM is a 47-bit counter chain, split as a 15-bit prescaler and a 32-bit free-running counter. Seconds, minutes, hours and days can be derived by software from the 32-bit counter. The MRTCSM can maintain a unique one-second count over a period of approximately 136 years. The prescaler provides additional sub-second information for precise timestamping.

The MRTCSM has interrupt generation capability for one of eight delays ranging from one second to 2^{23} seconds (approximately three months).

17.15.1.1 MRTCSM Terminology

In this section, the following terminology is used:

Update — The term “update” indicates the buffers are updated from the counter and/or prescaler.

Transfer — The term “transfer” is used to indicate a data transfer from the buffer to the counter and prescaler.

17.15.1.2 MRTCSM Features

- Programmable 47-bit free-running ripple counter for minimum power consumption split into a 15-bit prescaler and a 32-bit second counter
- Buffering of the 47-bit free-running counter to guaranty 32-bit and 47-bit coherent accesses on the 16-bit MIOB bus
- Possibility of suppressing 15-bit prescaler update
- Software shutdown of the dedicated low power oscillator to maintain battery shelf life

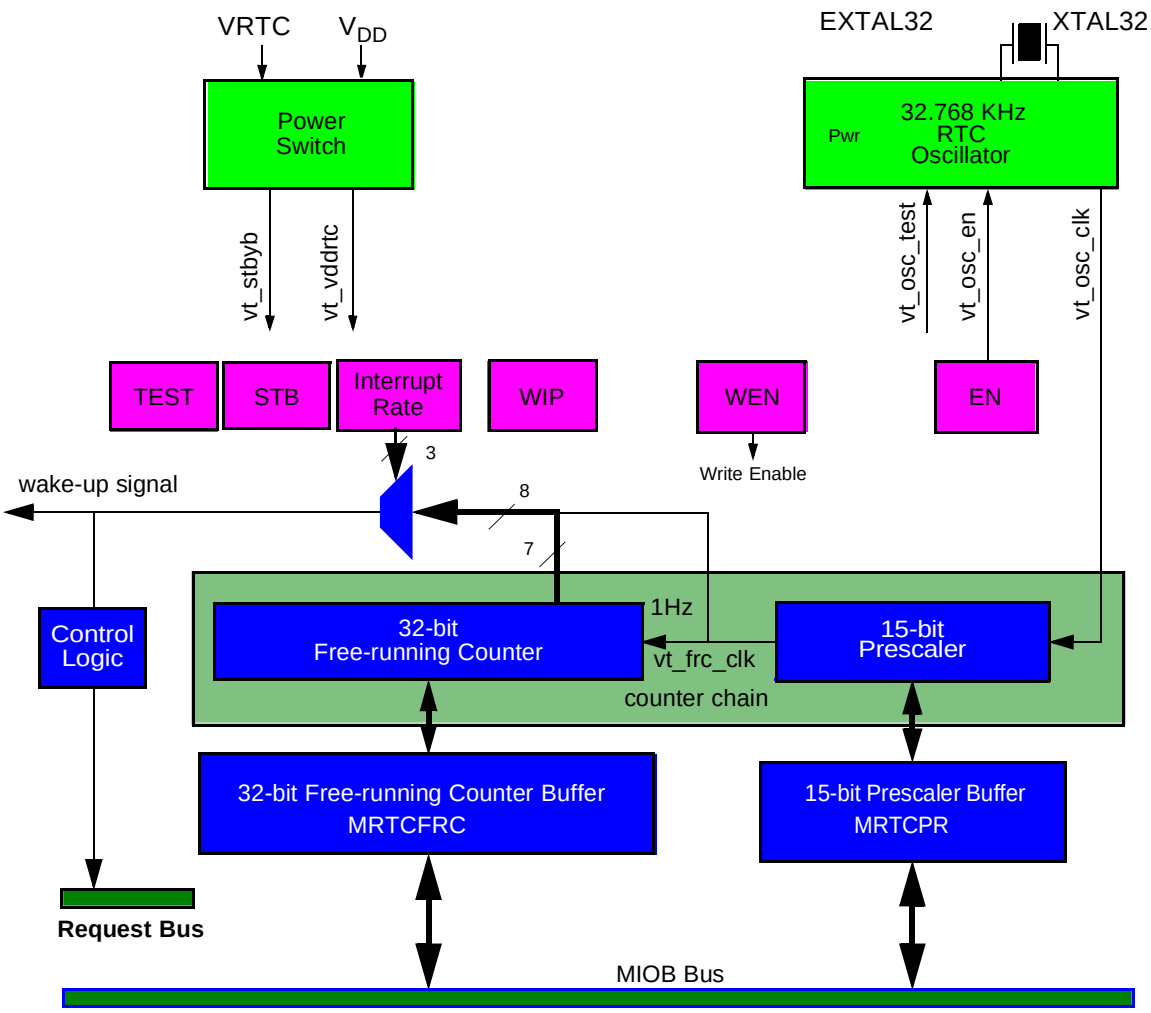


Figure 17-22 MRTCSM Block Diagram

- Flag setting and possible interrupt generation according to eight software selectable rates
- Wake-up signal generation for chip level usage
- Automatic hardware power supply selection through dedicated power switch
- Customized to use low cost standard 32.768-KHz crystal for internal clocking of the 32-bit counter at one Hz
- Software accessible precision of 2^{-15} second and unique time indication in seconds over a span of 136 years (2^{32} seconds)
- Eliminates risk of time inaccuracy due to interrupt overruns appearing in systems with software accumulated time

17.15.1.3 MRTCSM Pad Functions

The MRTCSM has no dedicated pads. However its dedicated power switch needs two pads for the 32.768-KHz crystal while its dedicated power switch needs one.

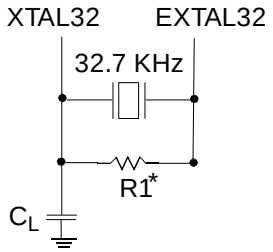
17.15.2 MRTCSM Description

This section describes the MRCTSM and its dedicated oscillator and power switch. Refer to the specifications of these two external modules for more information. Refer to **Figure 17-22** for a block diagram of the MRTCSM.

17.15.2.1 Oscillator

The basic time base for the MRTCSM is a 32.768-KHz dedicated low power oscillator. This oscillator uses an external crystal connected between the XTAL32 and EXTAL32 pads as a reference frequency source. The dedicated 32.768-KHz oscillator is in the chip periphery with the pads for the 32.768-KHz crystal. Having a dedicated oscillator with its uninterruptable power supply allows the counter chain (15-bit prescaler and 32-bit free-running counter) to run while the rest of the chip is powered down. The 32.768-KHz clock signal which is output by the oscillator is called vt_osc_clk. The frequency of the oscillator is called OSC_FREQ. The EXTAL32 pin has an internal load capacitor. Therefore an external load capacitor is not require on EXTAL32. The C_L on the XTAL32 side of the crystal should be according to the crystal manufacturer, typically 12 pF. The MIOS RTC oscillator circuit is shown in **Figure 17-23**.

The enable bit (EN) in the MRTCSM control register (MRTCSCR) can disable the oscillator and counter chain for maximum power saving. When enabled, the oscillator supplies the clock to the counter chain.



*Resistor is not currently required on the board but room should be left on the board for its addition in the future.

Figure 17-23 MIOS RTC Oscillator Circuit

17.15.2.2 Standby Supply and Power Switch

The MRTCSM power switch selects either the main power V_{DD} or dedicated standby power V_{RTC} as power supply (vt_vddrtc) for the MRTCSM. The power switch also generates the internal vt_stbyb signal which indicates in the MRTCSM which of the two supplies is selected:

- vt_stbyb = 0, standby mode, V_{RTC} supply selected

- vt_stbyb = 1, normal mode, V_{DD} supply selected

Normal mode is selected when V_{DD} is greater than V_{RTC}. The MRTCSM is supplied by V_{DD} and all the MRTCSM functions are allowed. Read and write operations to the MRTCSM registers comply to the description given in **17.15.1 MRTCSM Overview Description**.

The standby mode is selected automatically by hardware when the main V_{DD} supply becomes lower than the VRTC supply. While in standby mode, the MRTCSM is supplied by the VRTC supply. The counter chain continues to count the time from the dedicated 32.768-KHz oscillator. However, the buffers of the counter chain are no longer updated. If, despite being in standby mode, a buffer read or write operation occurs, a bus error is returned. Write operations to any of the MRTCSM registers have no effect.

The loss of primary power V_{DD} does not cause the counter chain to be affected as long as the VRTC power pad is above its minimum operating voltage. Only the loss of primary and standby power cause the content of the counter chain to be lost.

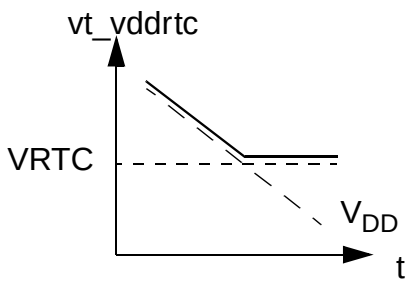


Figure 17-24 Selection of vt_vddrtc

17.15.2.3 Counter Chain

The MRTCSM avoids the software burden of servicing periodic interrupts to count time of day and calendar information. The software needs an initialization procedure that determines the preset state of the counter chain. After initialization, the software can calculate the current time and the current date from the contents of the counter chain. The software fetches the contents of the counter chain either regularly, with a periodic interrupt or as a low priority task in the application's software loop, or whenever needed. In any case, software response time problems during intensive CPU usage will not affect the counter chain accuracy.

When the EN bit in MRTCSCR is set, the MRTCSM oscillator and the counter chain are running. The counter chain is clocked with every rising edge of vt_osc_clk. Refer to **Figure 17-25** for timing details.

17.15.2.4 15-Bit Prescaler

The prescaler is a 15-bit presetable ripple counter designed to require minimum power consumption. The input clock is vt_osc_clk. The output is the vt_frc_clk clock signal used by the 32-bit free-running counter and runs at one Hz.

The 15-bit prescaler is double buffered in order to synchronize the available data and to allow easy coherent register accesses. The 15-bit prescaler can only be accessed through the 15-bit prescaler buffer (MRTCPR)

17.15.2.5 32-Bit Free-running Counter

The MRTCSM has a 32-bit presetable binary free-running counter that increments every second. The input to this counter is vt_frc_clk which comes directly from the 15-bit prescaler. The counter chain overflows approximately every 136 years.

The 32-bit free-running counter is double buffered in order to synchronize the available data and to allow easy coherent register accesses. The 32-bit free-running counter can only be accessed through the 32-bit free-running counter buffer (MRTCFRC).

17.15.2.6 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers

The 15-bit prescaler buffer (MRTCPR) and 32-bit free-running counter buffer (MRTC-FRC) shown in **Figure 17-22** serve two purposes:

- Synchronize the counter chain signals (vt_osc_clk and vt_frc_clk) to the peripheral bus system clock
- Provide coherent access to 47-bit or 32-bit data on a 16-bit data bus

MRTCFRC and MRTCPR can be read at any time. They must be written in conjunction and can only be written if the write enable bit (WEN) is set. Refer to **17.15.1.2 MRTCSM Features** for details concerning write operations to the buffers.

When doing regular byte or word read operations, the coherency of subsequent accesses is not guaranteed. Specific hardware is implemented that allows coherent accesses by using long word operations. Refer to **17.12.20 MDASMSCR — MDASM Status/Control Register** for the description of coherent accesses.

17.15.3 Modes of Operation

The following subsections describe how MRTCPR and MRTCFRC are updated and how they should be accessed by software.

17.15.3.1 Enabling the MRTCSM

The EN enable bit of the MRTCSCR register selects whether the oscillator and counter chain are running or not. The MRTCSM can be disabled (EN = 0) by software for maximum power saving (e.g., to maintain battery shelf life).

When the MRTCSM is disabled, writing to MRTCPR and MRTCFRC may give unpredictable results.

17.15.3.2 15-Bit Prescaler and 32-Bit Free-Running Counter Buffer Updates

When the MRTCSM is enabled (EN = 1), MRTCPR and MRTCFRC are updated at the OSC_FREQ rate. The timing for updating the buffers is shown in **Figure 17-25**. The 47-bit counter chain is incremented at every rising edge of vt_osc_clk. A pulse is generated after the falling edge of vt_osc_clk to synchronize transfers and updates of the buffers to/from the counter chain. In this document, this synchronized pulse is called vt_sync.

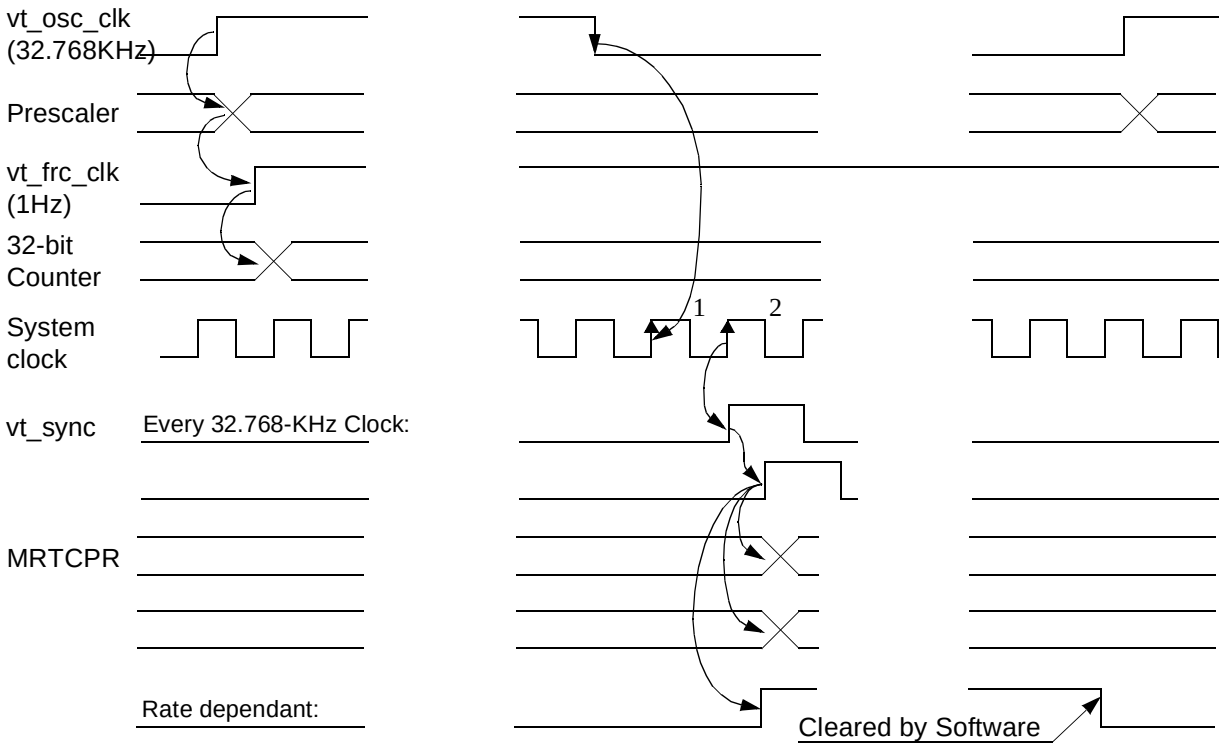


Figure 17-25 15-Bit Prescaler and 32-Bit Free-Running Counter Buffer Updates

In standby mode (vt_stbyb = 0), the update of MRTCPR and MRTCFRC is stopped to reduce power consumption. Access to all the MRTCSM registers is then inhibited. When exiting standby mode, the update of MRTCPR and MRTCFRC is reestablished at the regular OSC_FREQ rate starting on the next vt_sync.

In some particular conditions, the update of MRTCPR can be disabled. Refer to **17.15.2.2 Standby Supply and Power Switch** for more details.

17.15.3.3 Read of 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers

The addresses of the different MRTCSM registers are given in **17.15.9 MRTCSM Registers**. Byte or word accesses to MRTCPR, MRTCFRCH and MRTCFRCL can be performed at any time. However, the coherency between reads is not guaranteed

when using byte or word accesses since these buffers may be updated in between read operations.

In order to guarantee coherent reads, the following procedure should be followed:

1. Execute a long word read of MRTCFRC (at address MRTCFRCH)
2. Execute a word read of MRTCPR (optional)

If vt_sync occurs while a long word read operation of MRTCFRC is in progress, the update of the buffer is deferred until the end of the long word read.

Long word reads to MRTCFRC disable the update of MRTCPR until the next word read of MRTCPR. Once a 15-bit prescaler buffer read operation is performed, MRTCPR updates resume at the OSC_FREQ rate, starting with the next vt_sync.

17.15.3.4 Write to 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers

Write operations to the buffers have no effect if WEN is reset.

In order to write a new value coherently to the complete counter chain, the following sequence must be performed:

1. Set the WEN bit in the MRTCSCR register
2. Execute a long word write to MRTCFRC (at address MRTCFRCH), or two word writes to MRTCFRCH and MRTCFRCL
3. Execute a word write to MRTCPR

When the WEN bit is set, the update of MRTCPR and MRTCFRC is stopped. When MRTCPR is written to, the WEN bit is reset and the buffers become read only. The written values will then be transferred to the counter chain at the next vt_sync pulse. Following this sequence, the buffers will be updated from the counter chain at their regular OSC_FREQ rate, as described in **17.15.3.2 15-Bit Prescaler and 32-Bit Free-Running Counter Buffer Updates**. The option of disabling the update of MRTCPR from the counter chain is not affected by the write operation.

Once WEN is set, write MRTCFRC before MRTCPR. Not doing so transfers an old value of MRTCFRC to the counter chain.

When a word write to MRTCPR happens after setting the WEN bit, the WIP read only bit in the MRTCSCR register is set. This bit is reset once the transfer has occurred. The WIP bit indicates to the programmer that a write procedure is in progress

While doing a coherent write, once the prescaler buffer has been written to, the WEN bit cannot be set again until the full contents of MRTCPR and MRTCFRC have been transferred to the counter chain. Trying to set the WEN bit before this operation is completed has no effect.

If a reset appears while doing a coherent write or after a coherent write before the next vt_sync, the write operation is aborted and the buffers are updated as usual from the counter chain at the next vt_sync pulse.

17.15.4 MRTCSM Interrupt

The MRTCSM is capable of generating interrupts at a fixed time interval. The value of the interrupt rate is determined by the MRTCSCR bits IR2, IR1 and IR0. These three bits control which bit in the counter chain will be monitored for a positive edge. Therefore, the interrupt rates are powers of two of the vt_frc_clk period. The interrupt flag is updated synchronously with vt_sync.

The possible rates which can be selected are shown in **Table 17-46**.

Interrupts are not generated when writing a new value to the counter chain.

When exiting low power mode with the peripheral bus clock stopped or standby mode, buffers contain unpredictable values until the next vt_sync and the periodic interrupt flag is undetermined until the second vt_sync. The programmer should consider clearing the corresponding RQSM interrupt enable bit before entering these low power or standby modes to avoid possible interrupts when exiting these modes.

17.15.5 Chip Wake-Up Feature

In the case of main power supply shut-down, the chip clocks are stopped. A wake-up signal is available at the MIOS14 periphery that allows the MRTCSM to wake up the chip periodically. This signal reflects the status of one of the bits of the counter chain. The period is programmable by setting IR2, IR1 and IR0 appropriately, and is the same as the interrupt signal period (refer to **Table 17-46**).

EXAMPLE

This feature can be used to get the chip out of the low power mode. Refer to the chip specification for details about the use of the wake-up signal.

17.15.6 Modular I/O Bus (MIOB) Interface

- The MRTCSM is connected to all the signals in the read/write and control bus, to allow data transfer from and to the MRTCSM registers, and to control the MRTCSM
- The MRTCSM does not use the counter bus set
- The MRTCSM uses the request bus to transmit the FLAG line to its request submodule (RQSM)

17.15.6.1 Low Power Mode — Peripheral Bus Clock Running

As long as the peripheral bus clock is running, the MRTCSM is running normally.

17.15.6.2 Low Power Mode — Peripheral Bus Clock Stopped

If the peripheral bus clock is stopped, the update of the prescaler and counter buffers is not possible. The interrupt flag is also not updated and consequently cannot be used to exit this low power mode. However, the operation of the wake-up signal, the dedi-

cated 32.768-KHz oscillator and the counter chain are not disturbed by this mode. The wake-up signal can still be used to exit this mode.

As soon as this low power mode is exited, the update of MRTCPR and MRTCFRC is done at their regular OSC_FREQ rate, starting on the next vt_sync. Until this update occurs, the buffers contain unpredictable values. Refer to **17.15.4 MRTCSM Interrupt** for a description of the effect of this mode on the interrupt flag.

When a coherent write operation is performed, it is mandatory to wait for the vt_sync following the completion of this operation before entering this low power mode.

17.15.7 Effect of Standby Mode on MRTCSM

When in standby mode, the MRTCSM is supplied by the VRTC power supply. The counter chain continues to count clocked by the 32.768-KHz oscillator. However, the update of MRTCPR and MRTCFRC is stopped. The interrupt flag is also not updated and consequently cannot be used to exit this mode.

In order to prevent loss of data in a run away situation during power-up and power-down sequences, the access to all the MRTCSM registers is inhibited. It is recommended to use an external LVI circuit asserting the RESET pin when the main V_{DD} supply is below the minimum specified value.

When exiting the standby mode, the update of MRTCPR and MRTCFRC is done at the regular OSC_FREQ rate, starting on the next vt_sync. Until this update occurs, the buffers contain unpredictable values. Refer to **17.15.4 MRTCSM Interrupt** for a description of the effect of this mode on the interrupt flag.

The standby mode should not be entered during a coherent write operation or while WIP is set. If this happens, the correct transfer to the counter chain is not guaranteed.

The prescaler update power saving feature described in **17.15.3.3 Read of 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers** is not affected by standby mode.

17.15.8 Effect of RESET on MRTCSM

When the MIOB reset is asserted, the access to all the MRTCSM registers is blocked. The operation of the MRTCSM oscillator, counter chain and buffer updates is not affected by reset. The interrupt rate selection and the prescaler update power saving feature are not affected by reset. Only the WEN, TEST, STB and WIP bits of the MRTCSCR register are affected by reset. Refer to **17.15.9 MRTCSM Registers** for more details.

Reset should not occur during a coherent write operation or while WIP is set. If this happens, the transfer operation is not guaranteed.

Since the interrupt rate and the power saving feature are not affected by the reset, initialize these options when the device is powered for the first time.

On a power-on reset, after a standby mode, buffers contain unpredictable values and the periodic interrupt flag is undetermined until the second vt_sync. However, the interrupt enable will have been cleared by reset in the corresponding RQSM.

17.15.9 MRTCSM Registers

The privilege level to access the MRTCSM registers depends on the MIOS14MCR SUPV bit. This privilege level is reset to user and can be changed to supervisor by software.

17.15.10 MRTCSM Register Organization

The MRTCSM register map comprises four 16-bit register locations.

All unused bits return zero when read by the software. All register addresses in this section are specified as offsets from the base address of the MRTCSM.

MRTCSM— Register Organization

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
0x30 6050	MRTCSM Free-running counter buffer high register (MRTCFRCH)														
0x30 6052	MRTCSM Free-running counter buffer low register (MRTCFRCL)														
0x30 6054	MRTCSM Prescaler buffer register (MRTCPR)														
0x30 6056	MRTCSM status and control register (MRTCSCR)														

17.15.10.1 MRTCSM Free-Running Counter High Buffer (MRTCFRCH) Register Bits

MRTCSMFRCH — MRTCSM 32-Bit Counter High Buffer Register **0x30 6050**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
SRESET:															
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 17-42 MRTCSMFRCH Bit Descriptions

Bit(s)	Name	Description
0:15	CH15 — CH0	MRTCFRCH is the data register associated with the 32-bit free-running counter high buffer. It contains the synchronized high word value of the 32-bit free-running counter or the value to be loaded into the high word 32-bit free-running counter. The MRTCFRCH register is not affected by reset.

17.15.10.2 MRTCSM Free-Running Counter Low Buffer (MRTCFRCL) Register Bits

MRTCSMFRCL — MRTCSM 32-Bit Counter Low Buffer Register **0x30 6052**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
CL15	CL14	CL13	CL12	CL11	CL10	CL9	CL8	CL7	CL6	CL5	CL4	CL3	CL2	CL1	CL0
SRESET:															
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 17-43 MRTCSMFRCL Bit Descriptions

Bit(s)	Name	Description
0:15	CL15 — CL0	MRTCFRCL is the data register associated with the 32-bit free-running counter low buffer. It contains the synchronized low word value of the 32-bit free-running counter or the value to be loaded into the low word 32-bit free-running counter. The MRTCFRCL register is not affected by reset.

17.15.10.3 MRTCSM Prescaler Counter Buffer (MRTCPR) Register Bits

MRTCPR — MRTCSM Prescaler Counter Buffer Register **0x30 6054**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	0
SRESET:															
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 17-44 MRTCPR Bit Descriptions

Bit(s)	Name	Description
0:15	PR15 — PR0	MRTCPR is the data register associated with the prescaler buffer. It contains the synchronized value of the 15-bit prescaler or the value to be loaded into the 15-bit prescaler. The MRTCPR register is not affected by reset.

17.15.10.4 MRTCSMSCR — MRTCSM Status/Control Register

The status and control register gathers read/write bits related to its control and configuration.

MRTCSMSCR — MRTCSM Status/Control Register **0x30 6056**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
WIP	RESERVED			WEN	EN	RESERVED		TEST	STB	RESERVED			IR2	IR1	IR0
SRESET:															
0	0	0	0	0	—	0	0	0	0	0	0	0	—	—	—

Table 17-45 MRTCSMSCR Bit Descriptions

Bit(s)	Name	Description
0	WIP	Write in progress status bit — This read only bit indicates that a transfer from the buffers to the counter chain is in progress. This bit is set when writing to MRTCPR during the coherent write procedure and is reset once the transfer to the counter chain has occurred. 0 = No transfer from buffers to counter chain in progress. 1 = Transfer from buffers to counter chain in progress. The WIP bit is cleared by reset.
1:3	—	Reserved
4	WEN	Write enable control bit — This active high control bit allows the counter chain to be written to. MRTCPR and MRTCFRC are read only registers and regular write operations to these registers have no effect. When the WEN bit is set, it enables writing to the counter chain buffers. **17.15.3.4 Write to 15-Bit Prescaler and 32-Bit Free-Running Counter Buffers** describes the coherent write procedure to be used. **NOTE:** The WEN bit cannot be set while the WIP bit is set. The WEN bit is cleared by reset.
5	EN	Enable control bit — This active high bit enables real time clock operation. It selects whether the MRTCSM is running or not. 0 = MRTCSM is not running. Oscillator, counter chain and associated logic is not running, thus completely disabling the MRTCSM for maximum power saving. 1 = MRTCSM is running, all MRTCSM functions are enabled. The EN bit is not affected by reset and is undefined after the first power-up of the MRTCSM. **17.15.3.1 Enabling the MRTCSM** describes the enabling of the MRTCSM. **NOTE:** The EN bit cannot be modified while the WIP bit is set.
6:7	—	Reserved
9	TEST	This bit is reserved for factory testing only and should never be writen to one. The TEST bit is cleared by reset.
10	STB	This bit is reserved for factory testing only and must always be zero. The STB bit is cleared by reset.
10:12	—	Reserved
13:15	IR[2:0]	Interrupt rate control bits — The three interrupt rate control bits select the rate of the timer interrupt. Refer to **Table 17-46** to determine the rate of the real time clock interrupt.

Table 17-46 Interrupt Rate Selection

MRTCSM Control Register Bits			**Monitored Counter Chain Bit #N**	**MRTCSM Interrupt Rate When vt_osc_clk = 32.768KHz (2N/32768)**
IR2	**IR1**	**IR0**		
0	0	0	15 (output of prescaler)	1 second
0	0	1	21	64 seconds = 1.1 minutes
0	1	0	25	1024 seconds = 17.1 minutes
0	1	1	27	4096 seconds = 1.1 hours
1	0	0	31	65536 second = 18.2 hours
1	0	1	34	524288 seconds = 6.1 days
1	1	0	36	2097152 seconds = 24.3 days
1	1	1	38	8388608 seconds = 3.2 months

The interrupt rate control bits are unaffected by reset and unknown after the first power-up of the MRTCSM.

SECTION 18
TIME PROCESSOR UNIT 3

The time processor unit 3 (TPU3), an enhanced version of the original TPU, is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU3 is fully compatible to the TPU2. Operating simultaneously with the CPU, the three TPU3 modules process micro-instructions, schedules and processes real-time hardware events, performs input and output, and accesses shared data without CPU intervention. Consequently, for each timer event, the CPU setup and service times are minimized or eliminated.

The MPC565 / MPC566 contains three independent TPU3s. **Figure 18-1** is a simplified block diagram of a single TPU3.

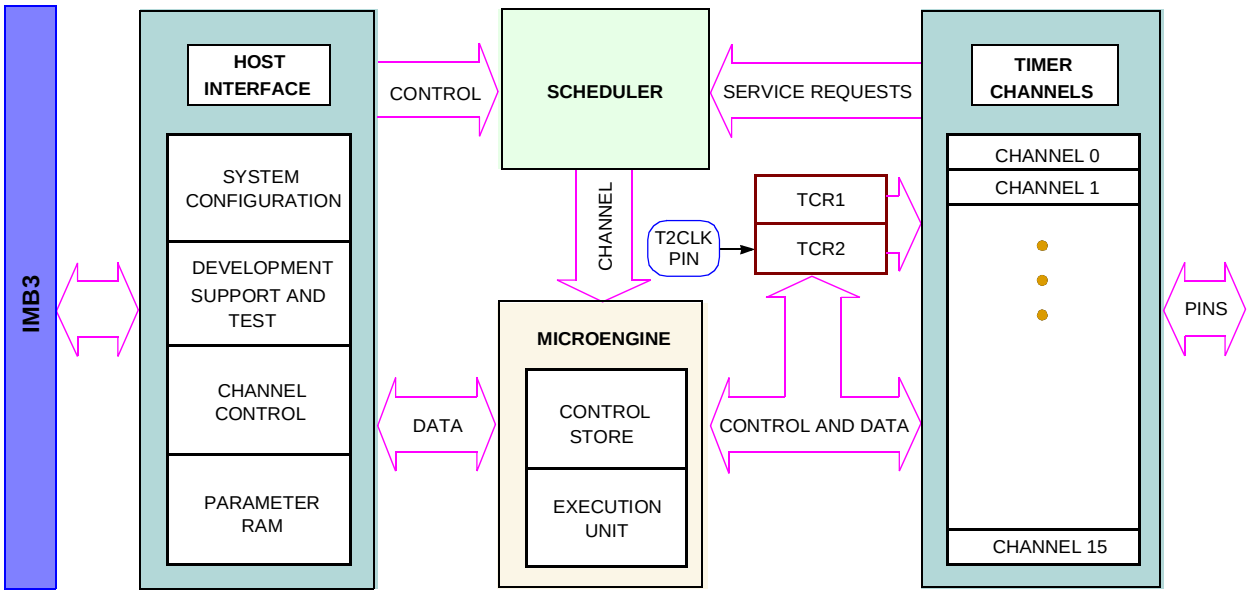


Figure 18-1 TPU3 Block Diagram

18.1 Overview

The TPU3 can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require CPU interrupt service.

The microcode ROM TPU3 functions that are available in the MPC565 / MPC566 are described in **APPENDIX D TPU ROM FUNCTIONS**.

18.2 TPU3 Components

The TPU3 consists of two 16-bit time bases, 16 independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-ported parameter RAM is used to pass parameters between the module and the CPU.

18.2.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the CPU via bit fields in the TPU3 module configuration register (TPUMCR) and TPU module configuration register two (TPUMCR2). Timer count registers TCR1 and TCR2 provide access to the current counter values. TCR1 and TCR2 can be read by TPU microcode but are not directly available to the CPU. The TCR1 clock is always derived from the system clock. The TCR2 clock can be derived from the system clock or from an external input via theT2CLK clock pin. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

18.2.2 Timer Channels

The TPU3 has 16 independent channels, each connected to an MCU pin. The channels have identical hardware and are functionally equivalent in operation. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

18.2.3 Scheduler

When a service request is received, the scheduler determines which TPU3 channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

18.2.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the CPU. Microcode can also be executed from the dual-port RAM (DPTRAM) module instead of the control store. The DPTRAM allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **18.3.6 Emulation Support** for more information.

18.2.5 Host Interface

The host interface registers allow communication between the CPU and the TPU3, both before and during execution of a time function. The registers are accessible from the IMB through the TPU3 bus interface unit. Refer to **18.4 Programming Model** for register bit/field definitions and address mapping.

18.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Channels zero through 15 each have eight parameters. The parameter RAM address map in **18.4.18 TPU3 Parameter RAM** shows how parameter words are organized in memory.

The CPU specifies function parameters by writing to the appropriate RAM address. The TPU3 reads the RAM to determine channel operation. The TPU3 can also store information to be read by the CPU in the parameter RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this manual. Refer to the ***TPU Reference Manual*** **(TPURM/AD)**, included in the ***TPU Literature Package*** **(TPULITPAK/D)** for more information.

18.3 TPU Operation

All TPU3 functions are related to one of the two 16-bit time bases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU3 can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneous match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

18.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. The time needed to respond to and service an event is determined by which channels and the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU3 performance in a given application. Latency can be closely estimated. For more information, refer to the ***TPU Reference Manual*** **(TPURM/AD)**.

18.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU3 channels contain identical hardware and are functionally equivalent in oper-

ation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

18.3.3 Interchannel Communication

The autonomy of the TPU3 is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

18.3.4 Programmable Channel Service Priority

The TPU3 provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

18.3.5 Coherency

For data to be coherent, all available portions of the data must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

18.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU3 provides emulation capability that allows the development of new time functions. Emulation mode is entered by setting the EMU bit in TPUMCR. In emulation mode, an auxiliary bus connection is made between the DPTRAM and the TPU3, and access to DPTRAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, DPTFLASH module access timing remains consistent with access timing of the TPU microcode ROM control store.

To support changing TPU application requirements, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Motorola Programming Note**,** ***Using the TPU Function Library and TPU Emulation Mode*** (**TPUPN00/D)** for information about developing custom functions and accessing the

TPU function library. Refer to the Motorola ***TPU Literature Package*** **(TPULITPAK/D)** for more information about specific functions.

18.3.7 TPU3 Interrupts

Each of the TPU3 channels can generate an interrupt service request. Interrupts for each channel must be enabled by writing to the appropriate control bit in the channel interrupt enable register (CIER). The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions set the flags. Setting a flag bit causes the TPU3 to make an interrupt service request if the corresponding channel interrupt enable bit is set.

The TPU3 can generate one of 32 possible interrupt request levels on the IMB3. The value driven onto $\overline{\text{IRQ}}$[7:0] represents the interrupt level programmed in the IRL field of the TPU interrupt configuration register (TICR). Under the control of the ILBS bits in the ICR, each interrupt request level is driven during the time multiplexed bus during one of four different time slots, with eight levels communicated per time slot. No hardware priority is assigned to interrupts. Furthermore, if more than one source on a module requests an interrupt at the same level, the system software must assign a priority to each source requesting at that level. **Figure 18-2** displays the interrupt level scheme.

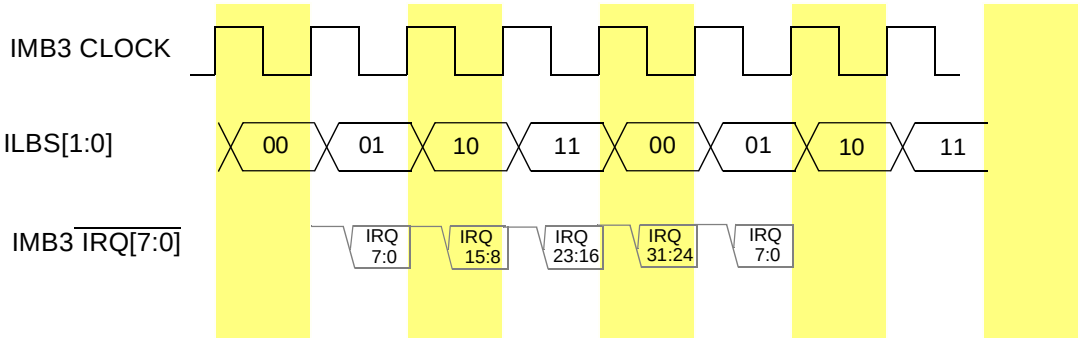


Figure 18-2 TPU3 Interrupt Levels

18.3.8 Prescaler Control for TCR1

Timer count register 1 (TCR1) is clocked from the output of a prescaler. The following fields control TCR1:

- The PSCK and TCR1P fields in TPUMCR
- The DIV2 field in TPUMCR2
- The EPSCKE and EPSCK fields in TPUMCR3.

The rate at which TCR1 is incremented is determined as follows:

- The user selects either the standard prescaler (by clearing the enhanced prescaler enable bit, EPSCKE, in TPUMCR3) or the enhanced prescaler (by setting EPSCKE).

— If the standard prescaler is selected (EPSCKE = 0), the the PSCK bit determines whether the standard prescaler divides the system clock input by 32 (PSCK = 0) or four (PSCK = 1)

— If the enhanced prescaler is selected (EPSCKE = 1), the EPSCK bits select a value by which the system clock is divided. The lowest frequency for TCR1 clock is system clock divided by 64x8. The highest frequency for TCR1 clock is system clock divided by two (2x1). See **Table 18-1**.

Table 18-1 Enhanced TCR1 Prescaler Divide Values

EPSCK Value	Divide System Clock By
0x00	2
0x01	4
0x02	6
0x03	8
0x04, 0x05,...0x1d	10,12,...60
0x1e	62
0x1f	64

— The output of either the standard prescaler or the enhanced prescaler is then divided by 1, 2, 4, or 8, depending on the value of the TCR1P field in the TPUMCR.

Table 18-2 TCR1 Prescaler Values

TCR1P Value	Divide by
0b00	1
0b01	2
0b10	4
0b11	8

— If the DIV2 bit is one, the TCR1 counter increments at a rate of the internal clock divided by two. If DIV2 is zero, the TCR1 increment rate is defined by the output of the TCR1 prescaler (which, in turn, takes as input the output of either the standard or enhanced prescaler).

Figure 18-3 shows a diagram of the TCR1 prescaler control block.

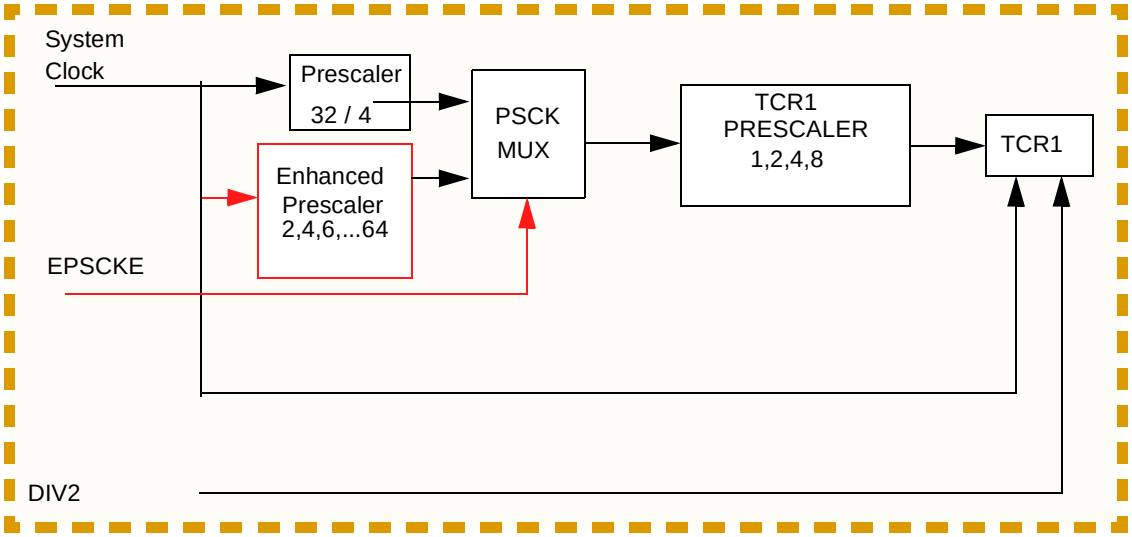


Figure 18-3 TCR1 Prescaler Control

18.3.9 Prescaler Control for TCR2

Timer count register 2 (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit and the T2CSL (TCR2 counter clock edge) bit in TPUMCR determine T2CR2 pin functions. Refer to **Table 18-3**.

Table 18-3 TCR2 Counter Clock Source

T2CSL	T2CG	TCR2 Clock
0	0	Rise transition T2CLK
0	1	Gated system clock
1	0	Fall transition T2CLK
1	1	Rise and fall transition T2CLK

The function of the T2CG bit is shown in **Figure 18-4**.

When T2CG is set, the external T2CLK pin functions as a gate of the DIV8 clock (the TPU3 system clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

The TCR2PSCK2 bit in TPUMCR3 determines whether the clock source is divided by two before it is fed into the TCR2 prescaler. The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to

resolve down to the TPU3 system clock divided by eight. **Figure 18-4** illustrates the TCR2 pre-divider and pre-scaler control.

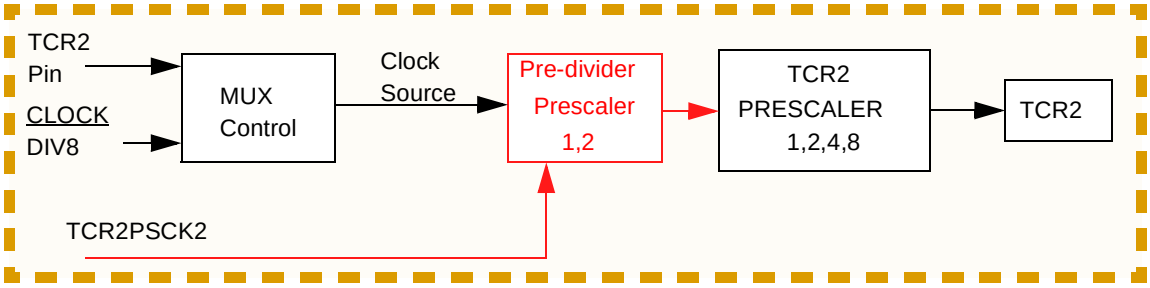


Figure 18-4 TCR2 Prescaler Control

Table 18-4 is a summary of prescaler output (assuming a divide-by-one value for the pre-divider prescaler.

Table 18-4 TCR2 Prescaler Control

TCR2 Value	Internal Clock Divide Ratio		External Clock Divide Ratio	
	TCR2PSCK2 = 0	TCR2PSCK2 = 1	TCR2PSCK2 = 0	TCR2PSCK2 = 1
0b00	8	8	1	1
0b01	16	24	2	3
0b10	32	56	4	7
0b11	64	120	8	15

18.4 Programming Model

The TPU3 memory map contains three groups of registers:

- System configuration registers
- Channel control and status registers
- Development support and test verification registers

All registers except the channel interrupt status register (CISR) must be read or written by means of half-word (16-bit) or word (32-bit) accesses. The address space of the TPU3 memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

Table 18-5 shows the TPU3 address map.

Table 18-5 TPU3 Register Map

	MSB 0 LSB 15
Address	**Register**
0x30 4000(A) 0x30 4400(B) 0x30 5C00(C)	TPU3 Module Configuration Register (TPUMCR) See **Table 18-6** for bit descriptions.
0x30 4002(A) 0x30 4402(B) 0x30 5C02(C)	TPU3 Test Configuration Register (TCR)
0x30 4004(A) 0x30 4404(B) 0x30 5C04(C)	Development Support Control Register (DSCR) See **Table 18-7** for bit descriptions.
0x30 4006(A) 0x30 4406(B) 0x30 5C06(C)	Development Support Status Register (DSSR) See **Table 18-8** for bit descriptions.
0x30 4008(A) 0x30 4408(B) 0x30 5C08(C)	TPU3 Interrupt Configuration Register (TICR) See **Table 18-9** for bit descriptions.
0x30 400A(A) 0x30 440A(B) 0x30 5C0A(C)	Channel Interrupt Enable Register (CIER) See **Table 18-10** for bit descriptions.
0x30 400C(A) 0x30 440C(B) 0x30 5C0C(C)	Channel Function Selection Register 0 (CFSR0) See **Table 18-11** for bit descriptions.
0x30 400E(A) 0x30 440E(B) 0x30 5C0E(C)	Channel Function Selection Register 1 (CFSR1) See **Table 18-11** for bit descriptions.
0x30 4010(A) 0x30 4410(B) 0x30 5C10(C)	Channel Function Selection Register 2 (CFSR2) See **Table 18-11** for bit descriptions.
0x30 4012(A) 0x30 4412(B) 0x30 5C12(C)	Channel Function Selection Register 3 (CFSR3) See **Table 18-11** for bit descriptions.
0x30 4014(A) 0x30 4414(B) 0x30 5C14(C)	Host Sequence Register 0 (HSQR0) See **Table 18-12** for bit descriptions.
0x30 4016(A) 0x30 4416(B) 0x30 4416(C)	Host Sequence Register 1 (HSQR1) See **Table 18-12** for bit descriptions.
0x30 4018(A) 0x30 4418(B) 0x30 5C18(C)	Host Service Request Register 0 (HSRR0) See **Table 18-13** for bit descriptions.
0x30 401A(A) 0x30 441A(B) 0x30 5C1A(C)	Host Service Request Register 1 (HSRR1) See **Table 18-13** for bit descriptions.
0x30 401C(A) 0x30 441C(B) 0x30 5C1C(C)	Channel Priority Register 0 (CPR0) See **Table 18-14** for bit descriptions.
0x30 401E(A) 0x30 441E(B) 0x30 5C1E(C)	Channel Priority Register 1 (CPR1) See **Table 18-14** for bit descriptions.

Table 18-5 TPU3 Register Map (Continued)

	MSB 0
Address	**Register**
0x30 4020(A) 0x30 4420(B) 0x30 5C20(C)	Channel Interrupt Status Register (CISR) See **Table 18-16** for bit descriptions.
0x30 4022(A) 0x30 4422(B) 0x30 5C22(C)	Link Register (LR)
0x30 4024(A) 0x30 4424(B) 0x30 5C24(C)	Service Grant Latch Register (SGLR)
0x30 4026(A) 0x30 4426(B) 0x30 5C26(C)	Decoded Channel Number Register (DCNR)
0x30 4028(A) 0x30 4428(B) 0x30 5C28(C)	TPU Module Configuration Register 2 (TPUMCR2) See **Table 18-17** for bit descriptions.
0x30 402A(A) 0x30 442A(B) 0x30 5C2A(C)	TPU Module Configuration 3 (TPUMCR3) See **Table 18-20** for bit descriptions.
0x30 402C(A) 0x30 442C(B) 0x30 5C2C(C)	Internal Scan Data Register (ISDR)
0x30 402E(A) 0x30 442E(B) 0x30 5C2E(C)	Internal Scan Control Register (ISCR)
0x30 4100 – 0x30 410F(A) 0x30 4500 – 0x30 450F(B) 0x30 5D00– 0x30 5D0F(C)	Channel 0 Parameter Registers
0x30 4110 – 0x30 411F(A) 0x30 4510 – 0x30 451F(B) 0x30 5D10 – 0x30 5D1F(C)	Channel 1 Parameter Registers
0x30 4120 – 0x30 412F(A) 0x30 4520 – 0x30 452F(B) 0x30 5D20 – 0x30 5D2F(C)	Channel 2 Parameter Registers
0x30 4130 – 0x30 413F(A) 0x30 4530 – 0x30 453F(B) 0x30 5D30 – 0x30 5D3F(C)	Channel 3 Parameter Registers
0x30 4140 – 0x30 414F(A) 0x30 4540 – 0x30 454F(B) 0x30 5D40 – 0x30 5D4F(C)	Channel 4 Parameter Registers
0x30 4150 – 0x30 415F(A) 0x30 4550 – 0x30 455F(B) 0x30 5D50 – 0x30 5D5F(C)	Channel 5 Parameter Registers
0x30 4160 – 0x30 416F(A) 0x30 4560 – 0x30 456F(B) 0x30 5D60 – 0x30 5D6F(C)	Channel 6 Parameter Registers
0x30 4170 – 0x30 417F(A) 0x30 4570 – 0x30 457F(B) 0x30 5D70 – 0x30 5D7F(C)	Channel 7 Parameter Registers

Table 18-5 TPU3 Register Map (Continued)

	MSB 0
Address	**Register**
0x30 4180 – 0x30 418F(A) 0x30 4580 – 0x30 458F(B) 0x30 5D80 – 0x30 5D8F(C)	Channel 8 Parameter Registers
0x30 4190 – 0x30 419F(A) 0x30 4590 – 0x30 459F(B) 0x30 5D90 – 0x30 5D9F(C)	Channel 9 Parameter Registers
0x30 41A0 – 0x30 41AF(A) 0x30 45A0 – 0x30 45AF(B) 0x30 5DA0 – 0x30 5DAF(C)	Channel 10 Parameter Registers
0x30 41B0 – 0x30 41BF(A) 0x30 45B0 – 0x30 45BF(B) 0x30 5DB0 – 0x30 5DBF(C)	Channel 11 Parameter Registers
0x30 41C0 – 0x30 41CF(A) 0x30 45C0 – 0x30 45CF(B) 0x30 5CC0 – 0x30 5CCF(C)	Channel 12 Parameter Registers
0x30 41D0 – 0x30 41DF(A) 0x30 45D0 – 0x30 45DF(B) 0x30 5DD0 – 0x30 5DDF(C)	Channel 13 Parameter Registers
0x30 41E0 – 0x30 41EF(A) 0x30 45E0 – 0x30 45EF(B) 0x30 5CD0 – 0x30 5DEF(C)	Channel 14 Parameter Registers
0x30 41F0 – 0x30 41FF(A) 0x30 45F0 – 0x30 45FF(B) 0x30 5DF0 – 0x30 5DFF(C)	Channel 15 Parameter Registers

18.4.1 TPU Module Configuration Register

TPUMCR — TPU Module Configuration Register **0x30 4000**
0x30 4400
0x30 5C00

MSB LSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	TPU3	T2CSL	RESERVED			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 18-6 TPUMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Low-power stop mode enable. If the STOP bit in TPUMCR is set, the TPU3 shuts down its internal clocks, shutting down the internal microengine. TCR1 and TCR2 cease to increment and retain the last value before the stop condition was entered. The TPU3 asserts the stop flag (STF) in TPUMCR to indicate that it has stopped. 0 = Enable TPU3 clocks 1 = Disable TPU3 clocks
1:2	TCR1P	Timer count register 1 prescaler control. TCR1 is clocked from the output of a prescaler. The prescaler divides its input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **18.3.8 Prescaler Control for TCR1** for more information.
3:4	TCR2P	Timer count register 2 prescaler control. TCR2 is clocked from the output of a prescaler. The prescaler divides this input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **18.3.9 Prescaler Control for TCR2** for more information.
5	EMU	Emulation control. In emulation mode, the TPU3 executes microinstructions from DPTRAM exclusively. Access to the DPTRAM via the IMB3 is blocked, and the DPTRAM is dedicated for use by the TPU3. After reset, this bit can be written only once. 0 = TPU3 and DPTRAM operate normally 1 = TPU3 and DPTRAM operate in emulation mode
6	T2CG	TCR2 clock/gate control 0 = TCR2 pin used as clock source for TCR2 1 = TCR2 pin used as gate of DIV8 clock for TCR2 Refer to **18.3.9 Prescaler Control for TCR2** for more information.
7	STF	Stop flag. 0 = TPU3 is operating normally 1 = TPU3 is stopped (STOP bit has been set)
8	SUPV	Supervisor data space 0 = Assignable registers are accessible from user or supervisor privilege level 1 = Assignable registers are accessible from supervisor privilege level only
9	PSCK	Standard prescaler clock. Note that this bit has no effect if the extended prescaler is selected (EPSCKE = 1). 0 = $f_{SYS} \div 32$ is input to TCR1 prescaler, if standard prescaler is selected 1 = $f_{SYS} \div 4$ is input to TCR1 prescaler, if standard prescaler is selected
10	TPU3	TPU3 enable. The TPU3 enable bit provides compatibility with the TPU. If running TPU code on the TPU3, the microcode size should not be greater than 2 Kbytes and the TPU3 enable bit should be cleared to zero. The TPU3 enable bit is write-once after reset. The reset value is one, meaning that the TPU3 will operate in TPU3 mode. 0 = TPU mode; zero is the TPU reset value 1 = TPU3 mode; one is the TPU3 reset value **NOTE**: The programmer should not change this value unless necessary when developing custom TPU microcode.
11	T2CSL	TCR2 counter clock edge. This bit and the T2CG control bit determine the clock source for TCR2. Refer to **18.3.9 Prescaler Control for TCR2** for details.
12:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TPU3 implementations that use hardware interrupt arbitration.

18.4.2 TPU3 Test Configuration Register

TCR — TPU3 Test Configuration Register **0x30 4002, 0x30 4402, 0x30 5C02**

Used for factory test only.

18.4.3 Development Support Control Register

DSCR — Development Support Control Register **0x30 4004**
0x30 4404
0x30 5C04

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HOT4	RESERVED				BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
RESET:															
0					0	0	0	0	0	0	0	0	0	0	0

Table 18-7 DSCR Bit Descriptions

Bit(s)	Name	Description
0	HOT4	Hang on T4 0 = Exit wait on T4 state caused by assertion of HOT4 1 = Enter wait on T4 state
1:4	—	Reserved
5	BLC	Branch latch control 0 = Latch conditions into branch condition register before exiting halted state 1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period
6	CLKS	Stop clocks (to TCRs) 0 = Do not stop TCRs 1 = Stop TCRs during the halted state
7:8	FRZ	FREEZE assertion response. The FRZ bits specify the TPU microengine response to the IMB3 FREEZE signal 00 = Ignore freeze 01 = Reserved 10 = Freeze at end of current microcycle 11 = Freeze at next time-slot boundary
9	CCL	Channel conditions latch. CCL controls the latching of channel conditions (MRL and TDL) when the CHAN register is written. 0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction 1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction
10	BP	µPC breakpoint enable 0 = Breakpoint not enabled 1 = Break if µPC equals µPC breakpoint register
11	BC	Channel breakpoint enable 0 = Breakpoint not enabled 1 = Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode
12	BH	Host service breakpoint enable 0 = Breakpoint not enabled 1 = Break if host service latch is asserted at beginning of state
13	BL	Link service breakpoint enable 0 = Breakpoint not enabled 1 = Break if link service latch is asserted at beginning of state
14	BM	MRL breakpoint enable 0 = Breakpoint not enabled 1 = Break if MRL is asserted at beginning of state
15	BT	TDL breakpoint enable 0 = Breakpoint not enabled 1 = Break if TDL is asserted at beginning of state

18.4.4 Development Support Status Register

DSSR — Development Support Status Register

0x30 4006
0x30 4406
0x30 5C04

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED								BKPT	PCBK	CHBK	SRBK	TPUF	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-8 DSSR Bit Descriptions

Bit(s)	Name	Description
0:7	—	Reserved
8	BKPT	Breakpoint asserted flag. If an internal breakpoint caused the TPU3 to enter the halted state, the TPU3 asserts the BKPT signal on the IMB and sets the BKPT flag. BKPT remains set until the TPU3 recognizes a breakpoint acknowledge cycle, or until the IMB FREEZE signal is asserted.
9	PCBK	µPC breakpoint flag. PCBK is asserted if a breakpoint occurs because of a µPC (microprogram counter) register match with the µPC breakpoint register. PCBK is negated when the BKPT flag is cleared.
10	CHBK	Channel register breakpoint flag. CHBK is asserted if a breakpoint occurs because of a CHAN register match with the CHAN register breakpoint register. CHBK is negated when the BKPT flag is cleared.
11	SRBK	Service request breakpoint flag. SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is cleared.
12	TPUF	TPU3 FREEZE flag. TPUF is set whenever the TPU3 is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU3 exits the halted state because of FREEZE being negated.
13:15	—	Reserved

18.4.5 TPU3 Interrupt Configuration Register

TICR — TPU3 Interrupt Configuration Register

0x30 4008
0x30 4408
0x30 5C08

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED					CIRL			ILBS		RESERVED					
RESET:															
					0	0	0	0	0	0	0				

Table 18-9 TICR Bit Descriptions

Bit(s)	Name	Description
0:4	—	Reserved
5:7	CIRL	Channel interrupt request level. This three-bit field specifies the interrupt request level for all channels. This field is used in conjunction with the ILBS field to determine the request level of TPU3 interrupts.
8:9	ILBS	Interrupt level byte select. This field and the CIRL field determine the level of TPU3 interrupt requests. 00 = $\overline{\text{IRQ}}$[0:7] selected 01 = $\overline{\text{IRQ}}$[8:15] selected 10 = $\overline{\text{IRQ}}$[16:23] selected 11 = $\overline{\text{IRQ}}$[24:31] selected
10:15	—	Reserved. Note that bits 10:11 represent channel interrupt base vector (CIBV) bits in some TPU3 implementations.

18.4.6 Channel Interrupt Enable Register

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU3 channels to request interrupt service. Setting the appropriate bit in the register enables a channel to make an interrupt service request; clearing a bit disables the interrupt.

CIER — Channel Interrupt Enable Register **0x30 400A**
0x30 440A
0x30 5C0A

MSB LSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-10 CIER Bit Descriptions

Bit(s)	Name	Description
0:15	CH[15:0]	Channel interrupt enable/disable 0 = Channel interrupts disabled 1 = Channel interrupts enabled **NOTE:** The MSB (bit 0 in big-endian mode) represents CH15, and the LSB (bit 15 in big-endian mode) represents CH0.

18.4.7 Channel Function Select Registers

Encoded 4-bit fields within the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions will be provided in a subsequent draft of this document.

CFSR0 — Channel Function Select Register 0

0x30 400C
0x30 440C
0x30 5C0C

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15				CH 14				CH 13				CH 12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1

0x30 400E
0x30 440E
0x30 5C0E

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 11				CH 10				CH 9				CH 8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2

0x30 4010
0x30 4410
0x30 5C10

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7				CH 6				CH 5				CH 4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3

0x30 4012
0x30 4412
0x30 5C12

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 3				CH 2				CH 1				CH 0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-11 CFSRx Bit Descriptions

Name	Description
CH[15:0]	Encoded time function for each channel. Encoded four-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

18.4.8 Host Sequence Registers

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Meanings of host sequence bits and host service request bits for predefined time functions will be provided in a subsequent draft of this document.

HSQR0 — Host Sequence Register 0 **0x30 4014 0x30 4414 0x30 5C14**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSQR1 — Host Sequence Register 1 **0x30 4016 0x30 4416 0x30 5C16**

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-12 HSQRx Bit Descriptions

Name	Description
CH[15:0]	Encoded host sequence. The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

18.4.9 Host Service Request Registers

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits is determined by time function microcode. Refer to the ***TPU Reference Manual* (TPURM/AD**) and the Motorola ***TPU Literature Package* (TPULITPAK/D)** for more information.

HSRR0 — Host Service Request Register 0

0x30 4018
0x30 4418
0x30 5C18

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSRR1 — Host Service Request Register 1

0x30 401A
0x30 441A
0x30 5C1A

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-13 HSSRx Bit Descriptions

Name	Description
CH[15:0]	Encoded type of host service. The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. A host service request field cleared to 0b00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three non-zero states. The CPU must monitor the host service request register until the TPU3 clears the service request to 0b00 before any parameters are changed or a new service request is issued to the channel.

18.4.10 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel.

CPR0 — Channel Priority Register 0

0x30 401C
0x30 441C
0x30 5C1C

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1

0x30 401E
0x30 441E
0x30 5C1E

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-14 CPRx Bit Descriptions

Name	Description
CH[15:0]	Encoded channel priority levels. **Table 18-15** indicates the number of time slots guaranteed for each channel priority encoding.

Table 18-15 Channel Priorities

CHx[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

18.4.11 Channel Interrupt Status Register

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU3 to make an interrupt service request if the corresponding CIER bit is set. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU3 register that can be accessed on a byte basis.

CISR — Channel Interrupt Status Register

0x30 4020
0x30 4420
0x30 5C20

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-16 CISR Bit Descriptions

Bit(s)	Name	Description
0:15	CH[15:0]	Channel interrupt status 0 = Channel interrupt not asserted 1 = Channel interrupt asserted

18.4.12 Link Register

LR — Link Register **0x30 4022, 0x30 4422, 0x30 5C22**

Used for factory test only.

18.4.13 Service Grant Latch Register

SGLR — Service Grant Latch Register **0x30 4024, 0x30 4424, 0x30 5C24**

Used for factory test only.

18.4.14 Decoded Channel Number Register

DCNR — Decoded Channel Number Register **0x30 4026, 0x30 4426, 0x30 5C26**

Used for factory test only.

18.4.15 TPU3 Module Configuration Register 2

TPUMCR2 — TPU Module Configuration Register 2 **0x30 4028**
0x30 4428
0x30 5C28

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED							DIV2	SOFT RST	ETBANK		FPSCK			T2CF	DTPU
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 18-17 TPUMCR2 Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	DIV2	Divide by 2 control. When asserted, the DIV2 bit, along with the TCR1P bit and the PSCK bit in the TPUMCR, determines the rate of the TCR1 counter in the TPU3. If set, the TCR1 counter increments at a rate of two system clocks. If negated, TCR1 increments at the rate determined by control bits in the TCR1P and PSCK fields. 0 = TCR1 increments at rate determined by control bits in the TCR1P and PSCK fields of the TPUMCR register 1 = Causes TCR1 counter to increment at a rate of the system clock divided by two
8	SOFT RST	Soft reset. The TPU3 performs an internal reset when both the SOFT RST bit in the TPUMCR2 and the STOP bit in TPUMCR are set. The CPU must write zero to the SOFT RST bit to bring the TPU3 out of reset. The SOFT RST bit must be asserted for at least nine clocks. 0 = Normal operation 1 = Puts TPU3 in reset until bit is cleared **NOTE**: Do not attempt to access any other TPU3 registers when this bit is asserted. When this bit is asserted, it is the only accessible bit in the register.

Table 18-17 TPUMCR2 Bit Descriptions (Continued)

Bit(s)	Name	Description
9:10	ETBANK	Entry table bank select. This field determines the bank where the microcoded entry table is situated. After reset, this field is 0b00. This control bit field is write once after reset. ETBANK is used when the microcode contains entry tables not located in the default bank 0. To execute the ROM functions on this MCU, ETBANK[1:0] must be 00. Refer to **Table 18-18**. **NOTE**: This field should not be modified by the programmer unless necessary because of custom microcode.
11:13	FPSCK	Filter prescaler clock. The filter prescaler clock control bit field determines the ratio between system clock frequency and minimum detectable pulses. The reset value of these bits is zero, defining the filter clock as four system clocks. Refer to **Table 18-19**.
14	T2CF	T2CLK pin filter control. When asserted, the T2CLK input pin is filtered with the same filter clock that is supplied to the channels. This control bit is write once after reset. 0 = Uses fixed four-clock filter 1 = T2CLK input pin filtered with same filter clock that is supplied to the channels
15	DTPU	Disable TPU3 pins. When the disable TPU3 control pin is asserted, pin TP15 is configured as an input disable pin. When the TP15 pin value is zero, all TPU3 output pins are three-stated, regardless of the pins function. The input is not synchronized. This control bit is write once after reset. 0 = TP15 functions as normal TPU3 channel 1 = TP15 pin configured as output disable pin. When TP15 pin is low, all TPU3 output pins are in a high-impedance state, regardless of the pin function.

Table 18-18 Entry Table Bank Location

ETBANK	Bank
00	0
01	1
10	2
11	3

Table 18-19 System Clock Frequency/Minimum Guaranteed Detected Pulse

Filter Control	Divide By	20 MHz	33 MHz	40 MHz
000	4	200 ns	121 ns	100 ns
001	8	400 ns	242 ns	200 ns
010	16	800 ns	485 ns	400 ns
011	32	1.6 µs	970 ns	800 ns
100	64	3.2 µs	1.94 µs	1.60 µs
101	128	6.4 µs	3.88 µs	3.20 µs
110	256	12.8 µs	7.76 µs	6.40 µs
111	512	25.6 µs	15.51 µs	12.80 µs

18.4.16 TPU Module Configuration Register 3

TPUMCR3 — TPU Module Configuration Register 3

0x30 402A
0x30 442A
0x30 5C2A

MSB															LSB
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RESERVED							PWOD	TCR2 PCK2	EPSCKE	RSVD	EPSCK				
RESET:															
					0	0	0	0	0	0	0				

Table 18-20 TPUMCR3 Bit Descriptions

Bit(s)	Name	Description
0:6	—	Reserved
7	PWOD	Prescaler write-once disable bit. The PWOD bit does not lock the EPSCK field and the EPSCKE bit. 0 = Prescaler fields in MCR are write-once 1 = Prescaler fields in MCR can be written anytime
8	TCR2PSCK2	TCR2 prescaler 2 0 = Prescaler clock source is divided by one. 1 = Prescaler clock is divided. See divider definitions in **Table 18-4**.
9	EPSCKE	Enhanced pre-scaler enable 0 = Disable enhanced prescaler (use standard prescaler) 1 = Enable enhanced prescaler. System clock will be divided by the value in EPSCK field.
10	—	Reserved
11:15	EPSCK	Enhanced prescaler value that will be loaded into the enhanced prescaler counter. Prescaler value = (EPSCK + 1) x 2. Refer to **18.3.8 Prescaler Control for TCR1** for details.

18.4.17 TPU3 Test Registers

The following TPU3 registers are used for factory test only:

- Internal scan data register (ISDR, address offset 0x30 402C, 0x30 442C, and 0x30 5C2C)
- Internal scan control register (ISCR, address offset 0x30 402E, 0x30 442E, 0x30 5C2E)

18.4.18 TPU3 Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 15 have eight parameters. The parameter registers constitute a shared work space for communication between the CPU and the TPU3. The TPU3 can only access data in the parameter RAM. Refer to **Table 18-21**.

Table 18-21 Parameter RAM Address Offset Map

Channel Number	Parameter							
	0	1	2	3	4	5	6	7
0	0x30 4100(A) 0x30 4500(B) 0x30 5D00(C)	0x30 4102A(A) 0x30 4502(B) 0x30 5D02(C)	0x30 4104(A) 0x30 4504(B) 0x30 5D04(C)	0x30 4106(A) 0x30 4506(B) 0x30 5D06(C)	0x30 4108(A) 0x30 4508(B) 0x30 5D08(C)	0x30 410A(A) 0x30 450A(B) 0x30 5D0A(C)	0x30 410C(A) 0x30 450C(B) 0x30 5D0C(C)	0x30 410E(A) 0x30 450E(B) 0x30 5D03(C)
1	0x30 4110(A) 0x30 4510(B) 0x30 5D10(C)	0x30 4112(A) 0x30 4512(B) 0x30 5D12(C)	0x30 4114(A) 0x30 4514(B) 0x30 5D14(C)	0x30 4116(A) 0x30 4516(B) 0x30 5D16(C)	0x30 4118(A) 0x30 4518(B) 0x30 5D18(C)	0x30 411A(A) 0x30 451A(B) 0x30 5D1A(C)	0x30 411C(A) 0x30 451C(B) 0x30 5D1C(C)	0x30 411E(A) 0x30 451E(B) 0x30 5D1E(C)
2	0x30 4120(A) 0x30 4520(B) 0x30 5D20(C)	0x30 4122(A) 0x30 4522(B) 0x30 5D23(C)	0x30 4124(A) 0x30 4524(B) 0x30 5D24(C)	0x30 4126(A) 0x30 4526(B) 0x30 5D26(C)	0x30 4128(A) 0x30 4528(B) 0x30 5D28(C)	0x30 412A(A) 0x30 452A(B) 0x30 5D2A(C)	0x30 412C(A) 0x30 452C(B) 0x30 5D2C(C)	0x30 412E(A) 0x30 452E(B) 0x30 5D2E(C)
3	0x30 4130(A) 0x30 4530(B) 0x30 5D30(C)	0x30 4132(A) 0x30 4532(B) 0x30 5D32(C)	0x30 4134(A) 0x30 4534(B) 0x30 5D34(C)	0x30 4136(A) 0x30 4536(B) 0x30 5D36(C)	0x30 4138(A) 0x30 4538(B) 0x30 5D38(C)	0x30 413A(A) 0x30 453A(B) 0x30 5D3A(C)	0x30 413C(A) 0x30 453C(B) 0x30 5D3C(C)	0x30 413E(A) 0x30 453E(B) 0x30 5D3E(C)
4	0x30 4140(A) 0x30 4540(B) 0x30 5D40(C)	0x30 4142(A) 0x30 4542(B) 0x30 5D42(C)	0x30 4144(A) 0x30 4544(B) 0x30 5D44(C)	0x30 4146(A) 0x30 4546(B) 0x30 5D46(C)	0x30 4148(A) 0x30 4548(B) 0x30 5D48(C)	0x30 414A(A) 0x30 454A(B) 0x30 5D4A(C)	0x30 414C(A) 0x30 454C(B) 0x30 5D4C(C)	0x30 414E(A) 0x30 454E(B) 0x30 5D4E(C)
5	0x30 4150(A) 0x30 4550(B) 0x30 5D50(C)	0x30 4152(A) 0x30 4552(B) 0x30 5D52(C)	0x30 4154(A) 0x30 4554(B) 0x30 5D54(C)	0x30 4156(A) 0x30 4556(B) 0x30 5D56(C)	0x30 4158(A) 0x30 4558(B) 0x30 5D58(C)	0x30 415A(A) 0x30 455A(B) 0x30 5D5A(C)	0x30 415C(A) 0x30 455C(B) 0x30 5D5C(C)	0x30 415E(A) 0x30 455E(B) 0x30 5D5E(C)
6	0x30 4160(A) 0x30 4560(B) 0x30 5D60(C)	0x30 4162(A) 0x30 4562(B) 0x30 5D62(C)	0x30 4164(A) 0x30 4564(B) 0x30 5D64(C)	0x30 4166(A) 0x30 4566(B) 0x30 5D66(C)	0x30 4168(A) 0x30 4568(B) 0x30 5D68(C)	0x30 416A(A) 0x30 456A(B) 0x30 5D6A(C)	0x30 416C(A) 0x30 456C(B) 0x30 5D6C(C)	0x30 416E(A) 0x30 456E(B) 0x30 5D6E(C)
7	0x30 4170(A) 0x30 4570(B) 0x30 5D70(C)	0x30 4172(A) 0x30 4572(B) 0x30 5D72(C)	0x30 4174(A) 0x30 4574(B) 0x30 5D74(C)	0x30 4176(A) 0x30 4576(B) 0x30 5D76(C)	0x30 4178(A) 0x30 4578(B) 0x30 5D78(C)	0x30 417A(A) 0x30 457A(B) 0x30 5D7A(C)	0x30 417C(A) 0x30 457C(B) 0x30 5D7C(C)	0x30 417E(A) 0x30 457E(B) 0x30 5D7E(C)
8	0x30 4180(A) 0x30 4580(B) 0x30 5D80(C)	0x30 4182(A) 0x30 4582(B) 0x30 5D82(C)	0x30 4184(A) 0x30 4585(B) 0x30 5D84(C)	0x30 4186(A) 0x30 4586(B) 0x30 5D86(C)	0x30 4188(A) 0x30 4588(B) 0x30 5D88(C)	0x30 418A(A) 0x30 458A(B) 0x30 5D8A(C)	0x30 418C(A) 0x30 458C(B) 0x30 5D8C(C)	0x30 418E(A) 0x30 458E(B) 0x30 5D8E(C)
9	0x30 4190(A) 0x30 4590(B) 0x30 5D90(C)	0x30 4192(A) 0x30 4592(B) 0x30 5D92(C)	0x30 4194(A) 0x30 4594(B) 0x30 5D94(C)	0x30 4196(A) 0x30 4596(B) 0x30 5D96(C)	0x30 4198(A) 0x30 4598(B) 0x30 5D98(C)	0x30 419A(A) 0x30 459A(B) 0x30 5D9A(C)	0x30 419C(A) 0x30 459C(B) 0x30 5D9C(C)	0x30 419E(A) 0x30 459E(B) 0x30 5D9E(C)
10	0x30 41A0(A) 0x30 45A0(B) 0x30 5DA0(C)	0x30 41A2(A) 0x30 45A2(B) 0x30 5DA2(C)	0x30 41A4(A) 0x30 45A4(B) 0x30 5DA4(C)	0x30 41A6(A) 0x30 45A6(B) 0x30 5DA6(C)	0x30 41A8(A) 0x30 45A8(B) 0x30 5DA8(C)	0x30 41AA(A) 0x30 45AA(B) 0x30 5DAA(C)	0x30 41AC(A) 0x30 45AC(B) 0x30 5DAC(C)	0x30 41AE(A) 0x30 45AE(B) 0x30 5DAE(C)
11	0x30 41B0(A) 0x30 45B0(B) 0x30 5DB0(C)	0x30 41B2(A) 0x30 45B2(B) 0x30 5DB2(C)	0x30 41B4(A) 0x30 45B4(B) 0x30 5DB4(C)	0x30 41B6(A) 0x30 45B6(B) 0x30 5DB6(C)	0x30 41B8(A) 0x30 45B8(B) 0x30 5DB8(C)	0x30 41BA(A) 0x30 45BA(B) 0x30 5DBA(C)	0x30 41BC(A) 0x30 45BC(B) 0x30 5DBC(C)	0x30 41BE(A) 0x30 45BE(B) 0x30 5DBE(C)
12	0x30 41C0(A) 0x30 45C0(B) 0x30 5DC0(C)	0x30 41C2(A) 0x30 45C2(B) 0x30 5DC2(C)	0x30 41C4(A) 0x30 45C4(B) 0x30 5DC4(C)	0x30 41C6(A) 0x30 45C6(B) 0x30 5DC6(C)	0x30 41C8(A) 0x30 45C8(B) 0x30 5DC8(C)	0x30 41CA(A) 0x30 45CA(B) 0x30 5DCA(C)	0x30 41CC(A) 0x30 45CC(B) 0x30 5DCC(C)	0x30 41CE(A) 0x30 45CE(B) 0x30 5DCE(C)
13	0x30 41D0(A) 0x30 45D0(B) 0x30 5DD0(C)	0x30 41D2(A) 0x30 45D2(B) 0x30 5DD2(C)	0x30 41D4(A) 0x30 45D4(B) 0x30 5DD4(C)	0x30 41D6(A) 0x30 45D6(B) 0x30 5DD6(C)	0x30 41D8(A) 0x30 45D8(B) 0x30 5DD8(C)	0x30 41DA(A) 0x30 45DA(B) 0x30 5DDA(C)	0x30 41DC(A) 0x30 45DC(B) 0x30 5DDC(C)	0x30 41DE(A) 0x30 45DE(B) 0x30 5DDE(C)
14	0x30 41E0(A) 0x30 45E0(B) 0x30 5DE0(C)	0x30 41E2(A) 0x30 45E2(B) 0x30 5DE2(C)	0x30 41E4(A) 0x30 45E4(B) 0x30 5DE4(C)	0x30 41E6(A) 0x30 45E6(B) 0x30 5DE6(C)	0x30 41E8(A) 0x30 45E8(B) 0x30 5DE8(C)	0x30 41EA(A) 0x30 45EA(B) 0x30 5DEA(C)	0x30 41EC(A) 0x30 45EC(B) 0x30 5DEC(C)	0x30 41EE(A) 0x30 45EE(B) 0x30 5DEE(C)
15	0x30 41F0(A) 0x30 45F0(B) 0x30 5DF0(C)	0x30 41F2(A) 0x30 45F2(B) 0x30 5DF2(C)	0x30 41F4(A) 0x30 45F4(B) 0x30 5DF4(C)	0x30 41F6(A) 0x30 45F6(B) 0x30 5DF6(C)	0x30 41F8(A) 0x30 45F8(B) 0x30 5DF8(C)	0x30 41FA(A) 0x30 45FA(B) 0x30 5DFA(C)	0x30 41FC(A) 0x30 45FC(B) 0x30 5DFC(C)	0x30 41FE(A) 0x30 45FE(B) 0x30 5DFE(C)

18.5 Time Functions

Descriptions of the MPC565 / MPC566 pre-programmed time functions are shown in **APPENDIX D TPU ROM FUNCTIONS**.

SECTION 19
DUAL-PORT TPU RAM (DPTRAM)

The dual-port RAM module with TPU microcode storage support (DPTRAM) consists of a control register block and a 4- or 6-Kbyte array of static RAM, which can be used either as a microcode storage for TPU or as a general-purpose memory. The MPC565 / MPC566 has two DPTRAM modules. One module (DPTRAM_AB) has a 6-Kbyte array and serves two TPU3 modules (A and B). The other module (DPTRAM_C) has a 4-Kbyte array and serves a third TPU3 module (C).

The DPTRAM module acts as a common memory on the IMB3 and allows the transfer of data to the two TPU3 modules. Therefore, the DPTRAM interface includes an IMB3 bus interface and two TPU3 interfaces. When the RAM is being used in microcode mode, the array is only accessible to the TPU3 via a separate local bus, and not via the IMB3.

In the MPC565 / MPC566, the DPTRAM base address register (RAMBAR) for each DPTRAM must be set to a particular value to fit into the IMB memory map of the part. The DPTRAM_AB RAMBAR must be programmed to 0xFFA0 and the DPTRAM_C RAMBAR must be programmed to 0xFF90.

The dual-port TPU3 RAM (DPTRAM) is intended to serve as fast, two-clock access, general-purpose RAM memory for the MCU. When used as general-purpose RAM, this module is accessed via the MCU's internal bus.

The DPTRAM module is powered by V_{DDL} in normal operation. The entire array may be used as standby RAM if standby power is supplied via the $V_{DDSRAM3}$ pin of the MCU. $V_{DDSRAM3}$ must be supplied by an external source. The DPTRAM powers the array both during run and in standby operation.

The DPTRAM may also be used as the microcode control store for up to two TPU3 modules when placed in a special emulation mode. In this mode the DPTRAM array may only be accessed by either or both of the TPU3 units simultaneously via separate emulation buses, and not via the IMB3.

The DPTRAM contains a multiple input signature calculator (MISC) in order to provide RAM data corruption checking. The MISC reads each RAM address and generates a 32-bit data-dependent signature. This signature can then be checked by the host.

The DPTRAM supports soft defects detection (SDD).

NOTE

The RCPU can not perform instruction fetches from any module on the IMB (including the DPTRAM). Only data accesses are permitted.

19.1 Features

- Six Kbytes of static RAM and four Kbytes of static RAM
- Only accessible by the CPU if neither TPU3 is in emulation mode
- Low-power stop operation
  - — Entered by setting the STOP bit in the DPTMCR
  - — Applies only to IMB3 accesses and not to accesses from either TPU3 interface
- TPU microcode mode
  - — The DPTRAM array acts as a microcode storage for the TPU module. This provides a means executing TPU code out of DPTRAM instead of programming it in the TPU ROM.
- Includes built in check logic which scans the array contents and calculates the RAM signature
- IMB3 bus interface
- Two TPU3 interface units
- Byte, half-word or word accessible

19.2 DPTRAM Configuration and Block Diagram

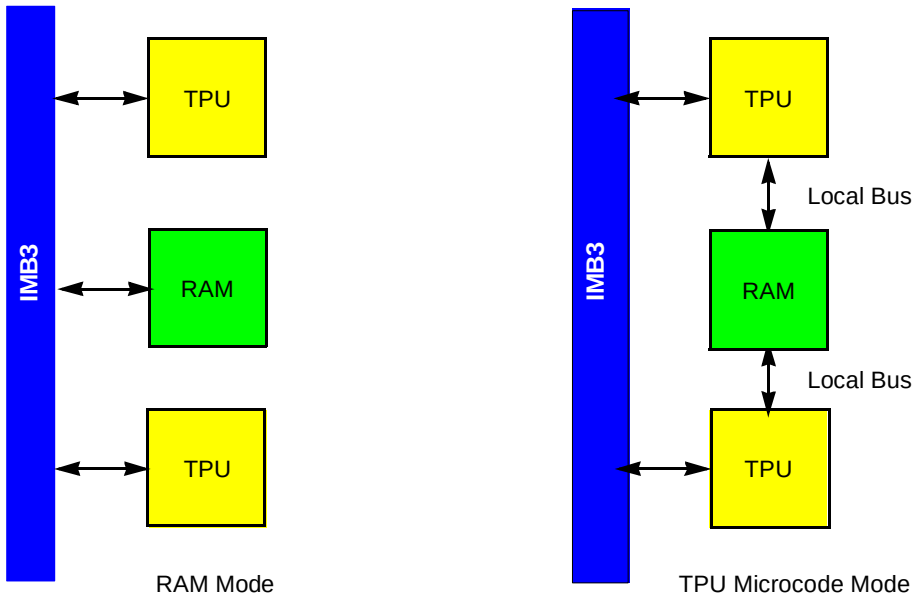


Figure 19-1 DPTRAM Configuration

19.3 Programming Model

The DPTRAM module consists of two separately addressable sections. The first is a set of memory-mapped control and status registers used for configuration (DPTMCR, RAMBAR, MISRH, MISRL, MISCNT) and testing (DPTTCR) of the DPTRAM array. The second section is the array itself.

All DPTRAM module control and status registers are located in supervisor data space. User reads or writes of these will result in a bus error.

When the TPU3 is using the RAM array for microcode control store, none of these control registers have any effect on the operation of the RAM array.

All addresses within the 64-byte control block will respond when accessed properly. Unimplemented addresses will return zeros for read accesses. Likewise, unimplemented bits within registers will return zero when read and will not be affected by write operations.

Table 19-1 shows the DPTRAM control and status registers. The addresses shown are offsets from the base address for the module. Refer to **Figure 1-2** to locate the DPTRAM control block in the MPC565 / MPC566 address map.

Table 19-1 DPTRAM Register Map

(AB) = Six Kbytes and (C) = Four Kbytes

R/W Access	Address	Register	Reset Value
Supv R/W	0x30 0000(AB) 0x30 0040(C)	DPT RAM Module Configuration Register (DPTRMCR) See **Table 19-2** for bit descriptions.	0x0100
Test	0x30 0002(AB) 0x30 0042(C)	Test Configuration Register (DPTTCR)	0x0000
Supv R/W	0x30 0004(AB) 0x30 0044(C)	RAM Base Address Register (RAMBAR) See **Table 19-3** for bit descriptions.	0x0001
Supv Read Only	0x30 0006(AB) 0x30 0046(C)	Multiple Input Signature Register High (MISRH) See **19.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0x30 0008(AB) 0x30 0048(C)	Multiple Input Signature Register Low (MISRL) See **19.3.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0x30 000A(AB) 0x30 004A(C)	Multiple Input Signature Counter (MISCNT) See **19.3.5 MISC Counter (MISCNT)** for bit descriptions.	Last memory address

The DPTRAM_AB array occupies a 6-Kbyte block. In the MPC565 / MPC566, the array must be located at the address 0x30 2000. The DPTRAM_C array occupies a 4-Kbyte block and the array must be located at the address 0x30 1000. Refer to **Figure 1-2** and **Figure 19-2**.

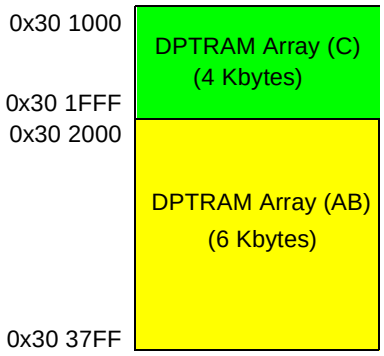


Figure 19-2 DPTRAM Memory Map

19.3.1 DPTRAM Module Configuration Register (DPTMCR)

This register defines the basic configuration of the DPTRAM module. The DPTMCR contains bits to configure the DPT RAM module for stop operation and for proper access rights to the array. The register also contains the MISC control bits.

DPTMCR — DPT Module Configuration Register **0x30 0000**
0x30 0040

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
STOP	NOT USED				MISF	MIS-EN	RASP	RESERVED							
RESET:															
0					0	0	1	0	0	0	0	0	0	0	0

Table 19-2 DPTMCR Bit Settings

Bit(s)	Name	Description
0	STOP	Low power stop (sleep) mode 0 = DPTRAM clocks running 1 = DPTRAM clocks shut down Only the STOP bit in the DPTMCR may be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior. Refer to **19.4.4 Stop Operation** for more information.
1:4	—	Reserved
5	MISF	Multiple input signature flag. MISF is readable at any time. This flag bit should be polled by the host to determine if the MISC has completed reading the RAM. If MISF is set, the host should read the MISRH and MISRL registers to obtain the RAM signature. 0 = First signature not ready 1 = MISC has read entire RAM. Signature is latched in MISRH and MISRL and is ready to be read.
6	MISEN	Multiple input signature enable. MISEN is readable and writable at any time. The MISC will only operate when this bit is set and the MPC565 / MPC566 is in TPU3 emulation mode. When enabled, the MISC will continuously cycle through the RAM addresses, reading each and adding the contents to the MISR. In order to save power, the MISC can be disabled by clearing the MISEN bit. 0 = MISC disabled 1 = MISC enabled
7	RASP	Ram area supervisor/user program/data. The RAM array may be placed in supervisor or unrestricted Space. When placed in supervisor space, (RASP = 1), only a supervisor may access the array. If a supervisor program is accessing the array, normal read/write operation will occur. If a user program is attempting to access the array, the access will be ignored and the address may be decoded externally. 0 = Both supervisor and user access to RAM allowed 1 = Supervisor access only to RAM allowed
8:15	—	Reserved. These bits are used for the IARB (interrupt arbitration ID) field in TPU3 implementations that use hardware interrupt arbitration.

19.3.2 DPTRAM Test Register

RAMTST — Test Register **0x30 0002, 0x30 0042**

RAMTST is used only during factory testing of the MCU.

19.3.3 Ram Base Address Register (RAMBAR)

The RAMBAR register is used to specify the 16 MSBs of the starting DPTRAM array location in the memory map. In the MPC565 / MPC566, this register must be programed to the value 0xFFA0.

This register can be written only once after a reset. This prevents runaway software from inadvertently re-mapping the array. Since the locking mechanism is triggered by the first write after reset, the base address of the array should be written in a single operation. Writing only one half of the register will prevent the other half from being written.

Soft reset has no effect on this register.

RAMBAR — RAM Array Base Address Register **0x30 0004**
0x30 0044

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	RESERVED			RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 19-3 RAMBAR Bit Settings

Bit(s)	Name	Description
0:10	A[8:19]	RAM array base address. These bits specify the 11 high-order bits (address lines ADDR[8:18] in little-endian notation) of the 24-bit base address of the RAM array. This allows the array to be placed on a 8-Kbyte boundary anywhere in the memory map. Do not overlap the RAM array memory map with other modules on the chip. On the MPC565 / MPC566 the value 0xFFA0 must be used for DPTRAM_6 Kbyte and 0xFF90 must be used for DPTRAM_4 Kbyte.
12:14	—	Reserved. (Bit 12 represent A[20] in DPTRAM implementations that require it.)
15	RAMDS	RAM disabled. RAMDS is a read-only status bit. The RAM array is disabled after a master reset since the RAMBAR register may be incorrect. When the array is disabled, it will not respond to any addresses on the IMB3. Access to the RAM control register block is not affected when the array is disabled. RAMDS is cleared by the DPTRAM module when a base address is written to the array address field of RAMBAR. RAMDS = 0: RAM enabled RAMDS = 1: RAM disabled

19.3.4 MISR High (MISRH) and MISR Low (MISRL)

The MISRH and MISRL together contain the 32-bit RAM signature calculated by the MISC. These registers are read-only and should be read by the host when the MISF

bit in the MCR is set. Note that the naming of the D[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode results in the reset of both MISRH and MISRL

MISRH — Multiple Input Signature Register High **0x30 0006**
0x30 0046

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISRL — Multiple Input Signature Register Low **0x30 0008**
0x30 0048

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

19.3.5 MISC Counter (MISCNT)

The MISCNT contains the address of the current MISC memory access. This registers is read-only. Note that the naming of the A[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode or clearing the MISEN bit in the DPTMCR results in the reset of this register.

MISCNT — MISC Counter **0x30 000A**
0x30 004A

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	LSB 15
A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	RESERVED		
RESET:															
Last Memory Address															

19.4 Operation

The DPTRAM module has several modes of operation. The following sections describe DPTRAM operation in each of these modes.

19.4.1 Normal Operation

In normal operation, read or write accesses of 8, 16, or 32 bits are supported. Also, in normal operation, neither TPU3 accesses the array, nor do they have any effect on the operation of the DPTRAM module.

19.4.2 Standby Operation

The DPTRAM array uses a separate power supply $V_{DDSRAM3}$ to provide power to the DPTRAM array during a power-down phase.

In order to guarantee valid DPTRAM data during power-down, external low voltage inhibit circuitry (external to the MCU) must be designed to force the RESET pin of the MCU into the active state before V_{DDL} drops below its normal limit. This is necessary to inhibit spurious writes to the DPTRAM during power-down.

19.4.3 Reset Operation

When a synchronous reset occurs, a bus master is allowed to complete the current access. Thus a write bus cycle (byte or half word) that is in progress when a synchronous reset occurs will be completed without error. Once a write already in progress has been completed, further writes to the RAM array are inhibited.

NOTE

A word (32-bit) write will be completed coherently only if the reset occurs during the second (16-bit) write bus cycle. If reset occurs during the first write bus cycle, only the first half word will be written to the RAM array and the second write will not be allowed to occur. In this case, the word data contained in the DPTRAM will not be coherent. The first half word will contain the most significant half of the new word information and the second half word will contain the least significant half of the old word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the RAM array are not guaranteed. (Refer to **SECTION 7 RESET** for a description of MPC565 / MPC566 reset sources, operation, control, and status.)

Reset will also reconfigure some of the fields and bits in the DPTRAM control registers to their default reset state. See the description of the control registers to determine the effect of reset on these registers.

19.4.4 Stop Operation

Setting the STOP control bit in the DPTMCR causes the module to enter its lowest power-consuming state. The DPTMCR can still be written to allow the STOP control bit to be cleared.

In stop mode, the DPTRAM array cannot be read or written. All data in the array is retained. The BIU continues to operate to allow the CPU to access the STOP bit in the DPTMCR. The system clock remains stopped until the STOP bit is cleared or the DPTRAM module is reset.

The STOP bit is initialized to logical zero during reset. Only the STOP bit in the DPTMCR can be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior.

NOTE

The STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior.

Switching to VDDSRAM3 occurs if V_{DDL} drops below its specified value when the RAM module is in stop mode.

The DPTRAM_AB will not enter stop mode if either or both of the TP1EMM or TP2EMM signals are asserted, indicating TPU3 emulation mode.

19.4.5 Freeze Operation

The FREEZE line on the IMB3 has no effect on the DPTRAM module. When the freeze line is set, the DPTRAM module will operate in its current mode of operation. If the DPTRAM module is not disabled, (RAMDS = 0), it may be accessed via the IMB3. If the DPTRAM array is being used by the TPU in emulation mode, the DPTRAM will still be able to be accessed by the TPU microengine.

19.4.6 TPU3 Emulation Mode Operation

To emulate TPU3 time functions, store the microinstructions required for all time functions to be used, in the RAM array. This must be done with the DPTRAM in its normal operating mode and accessible from the IMB3. After the time functions are stored in the array, place one or both of the TPU3 units in emulation mode. The RAM array is then controlled by the TPU3 units and disconnected from the IMB3.

To use the DPTRAM for microcode accesses, set the EMU bit in the corresponding TPU3 module configuration register. Through the auxiliary buses, the TPU3 units can access word instructions simultaneously at a rate of up to 56 MHz.

When a RAM array is being used by one or two of the TPU3 units, all accesses via the IMB3 are disabled. The control registers have no effect on the RAM array.

The contents of the RAM are validated using a multiple input signature calculator (MISC). MISC reads of the RAM are performed only when the MPC565 / MPC566 is in emulation mode and the MISC is enabled (MISEN = 1 in the DPTMCR).

Refer to **18.3.6 Emulation Support** for more information in TPU3 and DPTRAM operation in emulation mode.

19.5 Multiple Input Signature Calculator (MISC)

The integrity of the RAM data is ensured through the use of a MISC. The RAM data is read in reverse address order and a unique 32-bit signature is generated based on the output of these reads. MISC reads are performed when one of the TPU3 modules does not request back-to-back accesses to the RAM provided that the MISEN bit in the DPTMCR is set.

The MISC generates the DPTRAM signature based on the following polynomial:

$$G(x) = 1 + x + x^2 + x^{22} + x^{31}$$

After the entire RAM has been read and a signature has been calculated, the MISC sets the MISF bit in the MPC565 / MPC566 MCR. The host should poll this bit and enter a handling routine when the bit is found to be set.

The signature should be then read from the MISRH and MISRL registers and the host determines if it matches the predetermined signature.

The MISRH and MISRL registers are updated each time the MISC completes reading the entire RAM regardless of whether or not the previous signature has been read or not. This ensures that the host reads the most recently generated signature.

The MISC can be disabled by clearing the MISEN bit in the DPTMCR.

NOTE

The reset state of the DPTMCR MISEN is disabled.

SECTION 20
CDR3 FLASH (UC3F) EEPROM

20.1 Introduction

The U-bus CDR3 EEPROM module (UC3F) is designed for use in embedded microcontroller (MCU) applications targeted for high speed read performance and high density byte count requirements. The UC3F array uses a single transistor flash bit cell and is configured for a module of 512 Kbytes (524,288 bytes) of non-volatile memory (NVM). The UC3F array is divided into 64-Kbyte (65,536-byte) array blocks. The UC3F array size is created by using eight array blocks. Two blocks of the UC3F array memory map may be subdivided into two smaller blocks, a 16-Kbyte (16,384-byte) block and a 16-Kbyte block or a 16-Kbyte block and a 48-Kbyte (49,152-byte) block. The microcontroller's (MCU's) total flash requirements is distributed between two UC3F EEPROM modules on MPC565 / MPC566.

The primary function of the UC3F EEPROM module is to serve as electrically programmable and erasable NVM to store program instructions and/or data. It is a class of non-volatile solid state silicon memory device consisting of an array of isolated elements, an electrical means for selectively adding and removing charge to the elements, and a means of selectively sensing the stored charge in the elements. When power is removed from the device, the stored charge of the isolated elements will be retained.

The UC3F EEPROM module is arranged into two major sections as shown in **Figure 20-1**. The first section is the UC3F array used to store system program and data. The second section is the memory interface (MI) that controls operation of the UC3F array. The MI also serves as the interface between the UC3F array and a bus interface unit (BIU) which connects the UC3F array to the U-bus.

NOTE

If the flash arrays are disabled in the IMMR register (FLEN=0), then neither the UC3F array or the UC3F control registers are accessible.

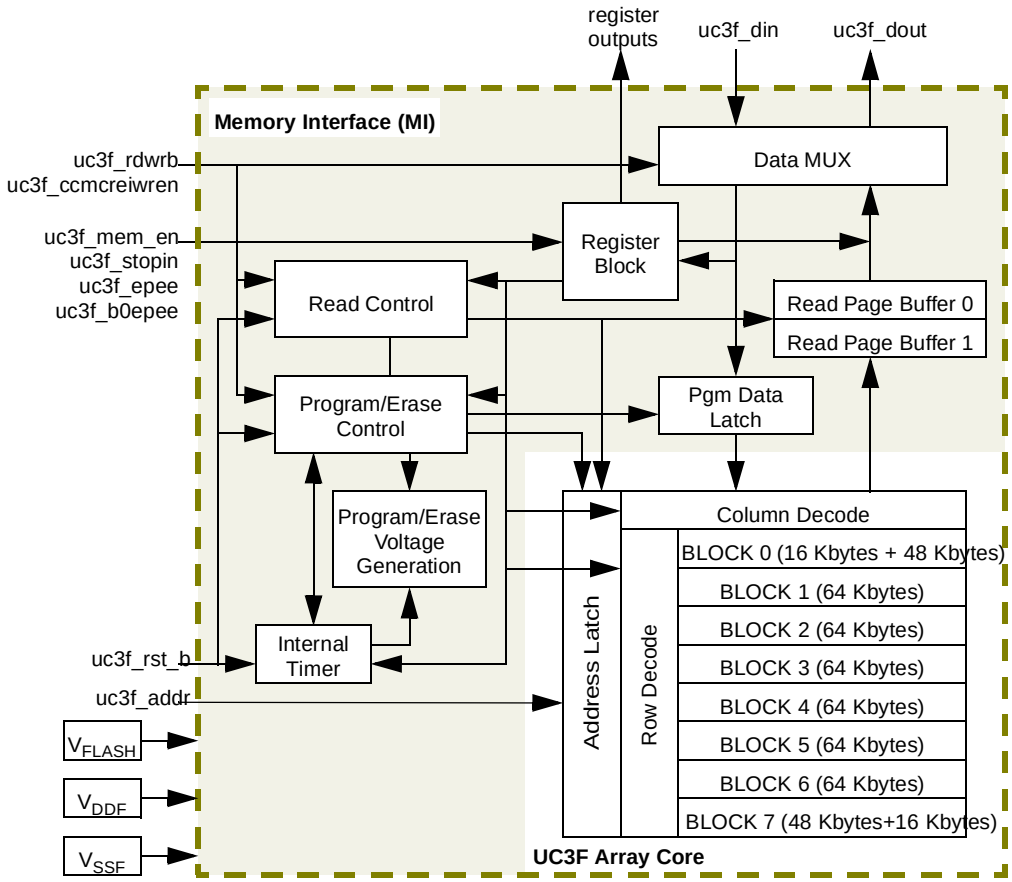


Figure 20-1 Block Diagram for a 512-Kbyte UC3F Module Configuration

The UC3F EEPROM module array is divided into array blocks to allow for independent erase, address attributes restrictions, and protection from program and erase for each array block. The size of a large array block in the UC3F module is fixed at 64 Kbytes. The size of a subdivided large block becomes the original large array block size minus 16 Kbytes, (64 Kbytes – 16 Kbytes = 48 Kbytes. The size of the small block, which is the remainder of the large block, is always 16 Kbytes. The total UC3F EEPROM array is distributed into eight large blocks, two of which contain small blocks. Information is transferred to the UC3F EEPROM by long-word (64 bits), word (32 bits), half-word (16 bits), or byte (eight bits).

To improve system performance, each array read access retrieves 32 bytes of information. These 32 bytes may be copied into one of two read page buffers aligned to the low order addresses. The two read page buffers are independently updated by page management logic contained in the BIU which interfaces to the UC3F EEPROM module.

To prevent unnecessary page accesses from the array, the UC3F memory interface (MI) shall monitor the incoming address to determine if the required information is in one of the two read page buffers. This strategy allows the UC3F array to have an off

page access and an on page access. In normal operation, write accesses to the UC3F array are not recognized except during program and erase operations.

The UC3F EEPROM uses an embedded hardware algorithm to program and erase the UC3F array. Special control logic is included to guard against accidental program or erase by requiring a specific series of read and write accesses to the UC3F control registers. Optional external inputs provide a hardware protection mechanism to prevent accidental program and erase of UC3F array blocks. The hardware algorithm automatically performs all necessary applications of high voltage pulses and verify reads of the UC3F array to ensure that all bits are programmed and erased with sufficient margin to guarantee data integrity and reliability.

20.1.1 Features of the CDR3 FLASH EEPROM (UC3F)

- High density single transistor flash bit cell
- -40 to 125° C ambient temperature operating range
- 2.5-V to 2.7-V V_{DDF} operating range and 4.75-V to 5.25-V V_{FLASH} operating range
- Shadow information stored in special flash NVM shadow locations
- Two 16-Kbyte small blocks per module
- 512 Kbytes per module using 64-Kbyte blocks
  - — One Mbyte total
- Array block restriction control for small and large blocks
  - — Erase by array block(s)
  - — Array protection for program and erase operations
  - — Array block assignment of supervisor or supervisor/user space
  - — Array block assignment of data or instruction/data space
- Internal 64-bit data path architecture
- Page mode read
  - — Retains two independent read page buffers
  - — Read page size of 32 bytes (eight words).
- Word (32-bit) programming
- Embedded hardware program and erase algorithm
  - — Uses internal oscillator to time program and erase pulses. Pulses are timed independently of system clock frequency
  - — Automatically performs margin reads
- Optional external flash program or erase enable inputs for block 0 or entire flash array
  - — Flash program or erase enables are selected from one of three sources: a pin external to the device, a register bit external to the UC3F but internal to the MCU, or always enabled
- Low power disable via an external signal or UC3F register bit
- Censor mode for flash memory array access restriction with a user bypass for unrestricted array access

20.1.2 Glossary of Terms Used

Array block — A 64-Kbyte UC3F array subdivision.

Array small block — The independently erasable 16-Kbyte subdivision of a 64-Kbyte array block.

Array host block — An array block containing a small block.

Array residual block — The 16-Kbyte or 48-Kbyte portion of a host block not contained in the small block.

BIU — Bus interface unit which controls access of the UC3F module through a standard system bus interface.

UC3F — The CDR3 flash EEPROM module.

CAM — Non-volatile content addressible memory cell used as a nonvolatile storage bit. CAMs are used for CENSOR bits.

Erase interlock write — A write to any UC3F array address after initializing the erase sequence.

IM — Integration module is a bus master which serves as the system bus arbitration unit on the microcontroller.

I**nitialize program/erase sequence** — The write to the high voltage control register that changes the SES bit from a 0 to a 1.

Mask option — A customer configurable option that is specified during the design process.

MCU — Microcontroller unit.

MI — Memory interface controls operation of the UC3F array.

Off page read — Array read operation that retrieves 32 bytes of information from the flash memory array and transfers the address selected eight bytes of the 32 bytes to the data out port.

On page read — Array read operation that accesses eight bytes of information stored in one of the read page buffers.

Programming write — A word write to a UC3F array address to transfer information into a program data latch. The UC3F EEPROM accepts programming writes after initializing the program sequence until the EHV bit is changed from a 0 to a 1.

Read page buffer — 32-byte block of information that is read from the UC3F array. This information is aligned to a 32-byte boundary within the UC3F array. Each UC3F module has two read page buffers.

Shadow information — The shadow provides 512 bytes of additional UC3F storage that is used for the reset configuration word and for additional user data. These loca-

tions may be accessed by setting the SIE bit in the module configuration register and accessing the UC3F array. The shadow information is always in the lowest numbered array block of the UC3F module.

SIU — System integration unit.

20.2 UC3F Interface

The UC3F module contains a memory interface (MI) and an array core. The MI controls access of the array core and register block in the UC3F module.

The interface signals to the UC3F module consist of address inputs, data inputs, data outputs, a simple set of control signals for read and write operations, a set of register selects, and a set of register outputs which are used by the BIU. Three required supply pins power the module: V_{DDF}, V_{SSF}, and V_{FLASH}.

The UC3F module is a fully asynchronous module and does not require a clock input for operation. All required clocks are generated internally using an internal oscillator, external test clock input, or internal delay circuits.

20.2.1 External Interface

The UC3F EEPROM module uses external pins to provide power supplies. These pins are listed in **Table 20-1**.

Table 20-1 UC3F External Interface Signals

Mnemonic	I/O Type	Description	Comments
V_{DDF}	Power Pin	UC3F power supply	To reduce noise in the read path no other circuits should be connected to the UC3F V_{DDF} supply pin.This V_{DD} pin must be isolated from all other V_{DD} pins inside the device. The specified voltage range during operation is 2.6 V ± 0.1 V.
V_{SSF}	Ground Pin	UC3F ground	To reduce noise in the read path no other circuits should be connected to the UC3F V_{SSF} supply. This V_{SS} pin must be isolated from all other V_{SS} pins inside the device.
VFLASH	Power Pin	UC3F 5-V power supply	VFLASH provides a 5-V supply to the UC3F module which is used for read, program, and erase operations. VFLASH must be in the range of 4.75 V to 5.25 V (5 V ± 5%) during operation.

Table 20-1 UC3F External Interface Signals (Continued)

Mnemonic	I/O Type	Description	Comments
EPEE	External Program/ Erase	EPEE pin status	The EPEE bit monitors the state of the external program/erase enable (EPEE) input. The UC3F module samples the EPEE input when EHV is asserted and holds that sampled state until EHV is negated.
B0EPEE	Block 0 External Program/ Erase	Block 0 EPEE pin status	The B0EM bit monitors the state of the Block 0 EPEE, B0EPEE, input. The UC3F module samples the B0EPEE input when EHV is asserted and holds that sampled value until EHV is negated.If B0EM = 1 when EHV is asserted, high voltage operations such as program or erase are enabled for either small block 0 or the lowest numbered block of the UC3F array regardless of the state of EPEE. If B0EM = 0 when EHV is asserted, high voltage operations are disabled for small block 0 or the lowest numbered block of the UC3F array regardless of the state of EPEE.

20.3 Programmer's Model

The UC3F EEPROM module consists of a control register block, an addressable shadow row implemented in flash, and an addressable main flash memory array. The control registers are used to configure, program, erase and exercise the UC3F shadow row and flash array.

20.3.0.1 UC3F EEPROM Module Control Register Addressing

The UC3F module control registers are selected with individual register selects generated from the BIU. As such, each flash module that is designed using the UC3F EEPROM module may uniquely define the addressing of the control register block.

Table 20-2 UC3F Register Programmer's Model

Address	Register
0x2F C800	Module A Configuration (UC3FMCR_A)
0x2F C804	High Voltage Control (UC3FCTL_A)
0x2F C808	Extended Module Configuration (UC3FMCRE_A)
0x2F C80C	Reserved
0x2F C840	Module A Configuration (UC3FMCR_B)
0x2F C844	High Voltage Control (UC3FCTL_B)
0x2F C848	Extended Module Configuration (UC3FMCRE_B)
0x2F C84C	Reserved

20.3.1 UC3F EEPROM Control Registers

The control registers are used to control UC3F EEPROM module operation. On reset, the registers are loaded with default reset information. Several bits of the UC3F control registers are special flash NVM registers which retain their state when power is

removed from the UC3F EEPROM. These special NVM registers are identified in the individual register field and control bit descriptions.

20.3.2 UC3F EEPROM Configuration Register (UC3FMCR)

The UC3F module configuration register is used to configure the operation and access restrictions of the UC3F array and shadow row.

UC3FMCR — UC3F EEPROM Configuration Register **0x2F C800**
0x2F C840

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
STOP	LOCK	Reserved	FIC	SIE	ACCESS	CENSOR		SUPV							
HRESET:															
0	1	0	0	0	0			1	1	1	1	1	1	1	1

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
DATA								PROTECT							
HRESET:															
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Reset State Determined by NVM Registers

Table 20-3 UC3FMCR Bit Descriptions

Bit(s)	Name	Description
0	STOP	Array stop control. When uc3f_stopin = 1 during the negation of uc3f_rst_b, the reset state of STOP is 1 and the C3F array wakes up in a disabled low power mode. Likewise, when uc3f_stopin = 0, the reset state of STOP is 0 and the UC3F array wakes up in a read mode. Writes to the STOP bit have no effect while in program or erase operation (SES = 1). The STOP bit is always readable whenever the registers are enabled. When STOP is set to 1, the UC3F array is disabled, and internal circuits are switched into a low power state. The STOP bit may be used to implement low power standby modes or power management schemes. At least the UC3FMCR remains readable and writable when STOP = 1 so that the STOP bit may be deasserted. Attempts to program or erase the array while STOP = 1 have no effect. SES cannot be set to 1 when STOP = 1. When STOP = 0, the UC3F array is enabled for accesses. In addition, all registers which were disabled with STOP = 1 are now enabled. A STOP recovery time of 1 µs is required for biases in the UC3F array to reach their appropriate levels to resume normal operation. Operations to the UC3F array should be delayed for at least 1µs after clearing the STOP bit. 0 = UC3F array is enabled 1 = UC3F array is disabled
1	LOCK[1]	Lock control. The default reset state of $\overline{\text{LOCK}}$ is 1. $\overline{\text{LOCK}}$ can be written to 0 once after reset when CSC = 0 to allow protection of the write-locked register bits after initialization. When the $\overline{\text{LOCK}}$ control bit in the UC3FMCR Register is written to a 0, the write-locked register bits: BPB, FIC, SUPV, SBSUPV[0:1], DATA, SBDATA, PROTECT, SBPROTECT, and SBEN[0:1] are locked. Writes to these bits while $\overline{\text{LOCK}}$ = 0 will have no effect. Once the $\overline{\text{LOCK}}$ bit has been asserted ($\overline{\text{LOCK}}$ = 0) in normal operating mode, the write-lock can only be disabled again by a reset. When the device is in background debug mode and CSC = 0, the $\overline{\text{LOCK}}$ bit may be written from a 0 to a 1. **WARNING**: If the lock protection mechanism is enabled ($\overline{\text{LOCK}}$ = 0) before PROTECT and SBPROTECT are cleared, the device must use background debug mode to program or erase the UC3F EEPROM.
2	—	Reserved
3	FIC	Force information censorship. The default reset state of FIC is normal censorship operation (FIC = 0). The FIC bit is write protected by the $\overline{\text{LOCK}}$ bit and the CSC bit. Writes will have no effect if $\overline{\text{LOCK}}$ = 0 or CSC = 1. Once FIC is set (FIC = 1), it can not be cleared except by a reset. The FIC bit can be read whenever the registers are enabled. The FIC bit is a censorship emulation mode used to aid in the development of custom techniques for controlling the ACCESS bit without setting CENSOR[0:1] to the information censorship state. Using FIC to force information censorship allows testing of the hardware and software for setting ACCESS without setting CENSOR[0:1] = 11 and risk permanently setting the device into an unusable information censorship state. 0 = normal uc3f censorship operation 1 = forces the uc3f into information censorship model
4	SIE	Shadow information enable. The default reset state of SIE is 0. The SIE bit is write protected in program operation (SES = 1 and PE = 0). The SIE bit can be read whenever the registers are enabled. When SIE = 1, normal array accesses are disabled, and the two shadow information rows are enabled. Array accesses are directed to the shadow row while SIE = 1. When an array location is read in this mode, only the lower 6 address bits are used to select which 64 bits of the 512-byte shadow row are read. The upper address bits are not used for shadow row decoding. The read page buffer address monitor shall be reset whenever SIE is modified making the next UC3F array access an off page access. 0 = normal array access 1 = disables normal array access and selects the shadow information rows

Table 20-3 UC3FMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
5	ACCESS	Enable uncensored access. A censored access to the UC3F EEPROM is any access where the device is in the censored mode (uc3f_censor_b = 0). The default reset state is ACCESS is a 0 so that FIC and CENSOR[0:1] control the level of censorship to the UC3F EEPROM array. All accesses to the UC3F EEPROM array shall be allowed if ACCESS = 1. ACCESS can be read whenever the registers are enabled. ACCESS provides a method to bypass the UC3F EEPROM module censorship. 0 = censored uc3f array access allowed only if the c3f censorship is no censorship 1 = allows all uc3f array access
6:7	CENSOR	Censor accesses. The CENSOR[0:1] bits are implemented using nonvolatile register bits or CAM cells. The reset state of CENSOR[0:1] is user defined by the contents stored in the NVM register bits. CENSOR is not writable but the NVM register's data can be set or cleared to the desired reset state. Reading CENSOR while setting or clearing with the high voltage applied (CSC = 1 and HVS = 1) will return 0's. 00 = cleared censorship, UC3F array access allowed only if device is in uncensored mode 01 = no censorship, all UC3F array accesses allowed 10 = no censorship, all UC3F array accesses allowed 11 = information censorship, UC3F array access allowed only if device is in uncensored mode
8:15	SUPV	Supervisor space. The SUPV bits are used to assign supervisor space restrictions for each block of the UC3F array. The index for the SUPV bit field is used to determine block assignment. For example, SUPV[0] is used for the supervisor space assignment of array block 0, while SUPV[4] is used for array block 4 Supervisor space assignment. Array block M is mapped into supervisor address space when SUPV[M] = 1, and only supervisor accesses are allowed to array block M. If SUPV[M] = 0, then array block M is mapped into unrestricted address space which allows both supervisor and user accesses to array block M. The SUPV bits are not actually used in the UC3F EEPROM module but are used by the BIU to determine access restrictions to UC3F array on a blockwise basis. The block addresses are decoded in the BIU to determine which array block is selected, and the selected block's SUPV bit is compared with the address space attributes to determine validity of an array access. When the small block function is enabled, the enabled small block portion of an array block is not controlled by the SUPV bit corresponding to the array block containing that small block. This particular small block is controlled by the appropriate SBSUPV bit while the remainder of that array block is controlled by its SUPV bit. 0 = array block m is placed in unrestricted address space 1 = array block m is placed in supervisor address space

Table 20-3 UC3FMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
16:23	DATA	Data space. The DATA bits are write protected by $\overline{\text{LOCK}}$ and CSC. Writes to DATA have no effect if $\overline{\text{LOCK}}$ = 0 or CSC = 1. The DATA bits may be read whenever the registers are enabled. Each array block of the UC3F EEPROM may be mapped into data or data and instruction address space. When array block M is mapped into data address space (DATA[M] = 1), only data accesses will be allowed. When array block M is mapped into both Data and Instruction address space (DATA[M] = 0), both data and instruction accesses will be allowed. The DATA bits are not actually used in the UC3F EEPROM module but are used by the BIU to determine access restrictions to UC3F array on a blockwise basis. The block addresses are decoded in the BIU to determine which array block is selected, and the selected block's DATA bit is compared with the address space attributes to determine validity of an array access. When the small block function is enabled, the enabled small block portion of an array block is not controlled by the DATA bit corresponding to the array block containing that small block. This particular small block is controlled by the appropriate SBDATA bit while the remainder of that array block is controlled by its DATA bit. 0 = array block m is placed in both data and instruction address spaces 1 = array block m is placed in data address space
24:31	PROTECT	Block protect. Each array block of the UC3F EEPROM can be individually protected from program or erase operation. The contents of array block M are protected from program or erase by setting PROTECT[M] = 1. The UC3F will perform all program and erase interlocks and complete the program or erase sequence, but the program and erase voltages are not applied to locations within the protected array block(s), blocks whose corresponding PROTECT bit is set to 1. By setting PROTECT[M] = 0, array block M is enabled for program and erase operation, and its contents may be altered by programming or erasing. When the small block function is enabled, the enabled small block portion of an array block is not controlled by the PROTECT bit corresponding to the array block containing that small block. This particular small block is controlled by the appropriate SBPROTECT bit while the remainder of that array block is controlled by its PROTECT bit. 0 = array block m is unprotected 1 = array block m is protected

NOTES:
1. Note that the LOCK bit is in a different bit location on the MPC565 / MPC566 than in the MPC555/556.

20.3.3 UC3F EEPROM Extended Configuration Register (UC3FMCRE)

The UC3FMCRE is an extended module configuration register used for configuring the small block functions. In addition, 16 bits of the UC3FMCRE are used to provide a source for module identification.

UC3FMCRE— UC3F EEPROM Extended Configuration Register **0x2F C804**
0x2F C844

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SBEN		SBSUPV		SBDATA		SBPROTECT		Reserved		BIU					
HRESET:															
0	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
MEMSIZ			BLK	MAP	SBLKL		FLASHID								
HRESET:															
1	0	1	1	0	X[1]	X	0	0	0	0	0	0	0	0	0

NOTES:
1. SBLKL = 0b10 for module A and SBLKL = 0b01 for module B.

Table 20-4 UC3FMCRE Bit Descriptions

Bit(s)	Name	Description
0:1	SBEN	Small block enable. When SBEN[M]=0, the corresponding small block M behaves logically as if the small block is still part of the larger host block. In addition, the small block protect bit (SBPROTECT[M]), the small block supervisor bit (SBSUPV[M]), the small block data bit (SBDATA[M]), and the small block BLOCK bit (SBBLOCK[M]) corresponding to small block M have no effect. The corresponding small block is controlled by the same protect, supervisor, data, and block bits that control its host block. When SBEN[M] = 1, the corresponding small block M can be programmed and erased independently of its host block. The corresponding small block protect bit, the small block supervisor bit, the small block data bit, and the small block bit are enabled by SBEN. For example: when SBEN[0] = 0, Small Block 0 (16 Kbytes) and the residual block (16 Kbytes or 48 Kbytes) contained in the host block of Small Block 0 are programmed and erased as if the two blocks are one large array block (32 Kbytes or 64 Kbytes). When SBEN[0] = 1, small block 0 and the residual block contained in the host block of small block 0 behave as two separate blocks, i.e. small block 0 and the residual block in small block 0's host block can be programmed and erased independently of each other. 0 = small block M behaves as part of the host block 1 = small block M functions independent of host block
2:3	SBSUPV	Small block supervisor space. Each small array block of the UC3F EEPROM may be mapped into supervisor or unrestricted address space. When small array block M is mapped into supervisor address space, SBSUPV[M] = 1, only supervisor accesses are allowed. When small block M is mapped to unrestricted address space, SBSUPV[M] = 0, both supervisor and user accesses are allowed. If SBEN[M] = 0, the corresponding small block M is logically part of the host block and SBSUPV[M] has no effect. Instead, the corresponding SUPV[M] bit will be used to determine if the small block is mapped to Supervisor or Unrestricted address space. Like the SUPV[0:7] bits, SBSUPV are not actually used in the UC3F EEPROM module but are used by the BIU to determine access restrictions to the UC3F array. Block addresses are decoded in the BIU to determine which small array block is selected, and the selected small block's SBSUPV bit is compared with the address space attributes to determine validity of an array access. 0 = small block M is placed in unrestricted address space 1 = small block M is placed in supervisor address space

Table 20-4 UC3FMCRE Bit Descriptions (Continued)

Bit(s)	Name	Description
4:5	SBDATA	Small block data space. Each small array block of the UC3F EEPROM may be mapped into data or both data and instruction address space. When a small array block is mapped into data address space, SBDATA[M] = 1, only data accesses will be allowed. When a small array block is mapped into both data and instruction address space, SBDATA[M] = 0, both data and instruction accesses will be allowed. If SBEN[M] = 0, the corresponding small block M is logically part of the host block and SBDATA[M] has no effect. Instead, the corresponding DATA[M] bit will be used to determine if the small block is mapped to Data or to both Data and Instruction address space. Like the DATA bits, SBDATA are not actually used in the UC3F EEPROM module but are used by the BIU to determine access restrictions to the UC3F array. Block addresses are decoded in the BIU to determine which small array block is selected, and the selected small block's SBDATA bit is compared with the address space attributes to determine validity of an array access. 0 = small block M is place in both Data and Instruction address spaces 1 = small block M is place in Data address space
6:7	SBPRO-TECT	Small block protect. Each small block of the UC3F EEPROM can be individually protected from program or erase operation. The UC3F will perform all program and erase interlocks and even complete the program or erase sequence, but the program and erase voltages are not applied to locations within the protected small block(s). 0 = small block M is unprotected 1 = small block M is protected
8:9	—	Reserved
10:15	BIU	BIU configuration bits. These register bits are reserved for BIU functionality and are strictly outputs from the UC3F EEPROM.
16:18	MEMSIZ	Memory size. The MEMSIZ field is used to indicate the UC3F array size. The MEMSIZ bits are read only and writes have no effect. 0b000 = UC3F array is 64 Kbytes 0b001 = UC3F array is 128 Kbytes 0b010 = UC3F array is 192 Kbytes 0b011 = UC3F array is 256 Kbytes 0b100 = unused 0b101 = UC3F array is 512 Kbytes 0b110 = unused 0b111 = unused Both modules on the MPC565 / MPC566 are 512 Kbytes.
19	BLK	Block size. The BLK bit is used to indicate the array block size used in the UC3F array. The BLK bit is read only and writes have no effect. 0 = array block size is 32 Kbytes 1 = block size is 64 Kbytes
20	MAP	Array address mapping. The MAP bit is used to indicate the UC3F array address mapping within a 2^N address space. The MAP bit is read only and writes have no effect. The MAP bit is more useful when the UC3F array is a non-2^N size. When MAP = 0, the UC3F array is mapped to the bottom (starting at address 0) of the 2^N space in which the array resides. For modules with 2^N array sizes, the MAP bit is always set to 0. When MAP = 1, the UC3F array is mapped to the top (ending at address all $F's) of the 2^N space in which the array resides. The 192KB and 384KB modules are the only two allowed configurations of the UC3F module which have non-2^N array sizes. If MAP = 1 for a 192-Kbyte or 384-Kbyte array, the UC3F array is mapped starting at the respective 64KB offset from the beginning of a 256-Kbyte address space or 128-Kbyte offset from the beginning of a 512-Kbyte address space. 0 = UC3F array is mapped to top of 2^N address space 1 = UC3F array is mapped to bottom of 2^N address space

Table 20-4 UC3FMCRE Bit Descriptions (Continued)

Bit(s)	Name	Description
21:22	SBLKL	Small block location code. There are three possible locations for the small blocks: 1) a small block may be placed in the lowest numbered host block and the highest numbered host blocks, 2) a small block may be placed in the lowest numbered host block and the second lowest numbered host block, and 3) a small block may be placed in the second highest numbered host block and the highest numbered host block. 00 = unused 01 = small blocks are part of the two highest numbered blocks of the UC3F array 10 = small blocks are part of the two lowest numbered blocks of the UC3F array 11 = small blocks are part of the lowest and highest numbered blocks of the UC3F array
23:31	FLASHID	Flash module identification code. The FLASHID value is assigned by Motorola and used internally for tracking purposes. The FLASHID field is read only and writes have no effect.

20.3.4 UC3F EEPROM High Voltage Control Register (UC3FCTL)

The UC3F EEPROM high voltage control register is used to control the program and erase operations of the UC3F EEPROM module.

UC3FCTL— UC3F EEPROM High Voltage Control Register **0x2F C808**
0x2F C848

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HVS	PEGOOD	PEFI	EPEE	B0EM	Reserved									SBBLOCK	
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
BLOCK								0	CSC	Reserved		HSUS	PE	SES	EHV
HRESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 20-5 UC3FCTL Bit Descriptions

Bit(s)	Name	Description
0	HVS	High voltage status. The HVS bit is for status only, and writes to HVS have no effect. During a program or erase operation, HVS is set (HVS = 1) to indicate when high voltage operations are in progress. The HVS bit will negate itself when the program or erase operation completes successfully, EHV negates during program or erase to terminate the program/erase operation, HSUS is asserted to suspend the program/erase operation, resetting the module, or the internal hardware program/erase controller times out. 0 = no program or erase of the UC3F array or shadow information or CENSOR bits in progress 1 = program or erase of the UC3F array or shadow information or CENSOR bits in progress
1	PEGOOD	Program/erase operation result. The PEGOOD bit is for status only. At the completion of a program or erase operation using the embedded hardware algorithm, the hardware algorithm will change the state of the PEGOOD bit to reflect whether or not the program or erase operation was successful. The PEGOOD bit is only valid after the hardware program erase algorithm has cleared HVS. PEGOOD is reset when either EHV is asserted or SES is cleared. See **Figure 20-2** for a timing diagram of when PEGOOD is valid. 0 = program or erase operation failed 1 = program or erase operation was successful
2	PEFI	Program/erase fail indicator. The PEFI bit is a status qualifier for the PEGOOD bit and is valid for the same times that PEGOOD is valid. In the event of an erase failure which returns PEGOOD = 0, the PEFI bit provides diagnostic information for the cause of the erase failure. If PEFI = 0, the erase failure occurred during the preprogramming step of the erase operation. If PEFI = 1, the erase failure occurred during the actual erase or APDE steps of the erase operation. In the event of a program failure which returns PEGOOD = 0, the PEFI bit indicates a program failure by reading as a 0. The PEFI bit should never return a 1 for a program failure. NOTE: The PEFI bit is meaningful only while PEGOOD is valid and PEGOOD = 0. PEFI is valid after HVS negates and prior to the assertion of EHV or negation of SES. 0 = program operation failed if PEGOOD = 0 1 = erase operation failed if PEGOOD = 0
3	EPEE	EPEE pin status. The EPEE bit monitors the state of the external program/erase enable (EPEE) input. The UC3F module samples the EPEE input when EHV is asserted and holds that sampled state until EHV is negated. 0 = high voltage operations are not possible 1 = high voltage operations are possible
4	B0EM	Block 0 EPEE pin status. The B0EM bit monitors the state of the Block 0 EPEE, B0EPEE, input. The UC3F module samples the B0EPEE input when EHV is asserted and holds that sampled value until EHV is negated.If B0EM = 1 when EHV is asserted, high voltage operations such as program or erase are enabled for either small block 0 or the lowest numbered block of the UC3F array regardless of the state of EPEE. If B0EM = 0 when EHV is asserted, high voltage operations are disabled for small block 0 or the lowest numbered block of the UC3F array regardless of the state of EPEE. 0 = high voltage operations are not possible for block 0 or lowest numbered block 1 = high voltage operations are possible for block 0 or lowest numbered block.
5:13	—	**Reserved**
14:15	SBBLOCK	Small block block program and erase select. The SBBLOCK bits are write-protected by the SES bit. SBBLOCK selects the UC3F EEPROM small array blocks for program and erase operation. When programming, only those blocks intended to be enabled for programming should have their corresponding BLOCK[M] or SBBLOCK[M] bit set. 0 = small block M is not selected for program or erase 1 = small Block M is selected for program or erase

Table 20-5 UC3FCTL Bit Descriptions (Continued)

Bit(s)	Name	Description
16:23	BLOCK	Block program and erase select. The BLOCK bits are write protected by the SES bit. BLOCK selects the UC3F EEPROM array blocks for program and erase operation. All the blocks may be selected for program or erase operation at once. The UC3F EEPROM configuration along with BLOCK determine which array blocks that may be programmed. The UC3F EEPROM array blocks that are enabled to be programmed by the program operation are the blocks whose corresponding BLOCK bit is set to 1. For example, if array blocks 2 and 5 are enabled for programming, BLOCK[2] and BLOCK[5] must be set to 1 while BLOCK[0], BLOCK[1], BLOCK[3], BLOCK[4], BLOCK[6], and BLOCK[7] are set to 0. The UC3F EEPROM configuration along with BLOCK determine the blocks that will be erased simultaneously. All array blocks whose corresponding BLOCK bits are set will be erased during the erase operation. For example, if BLOCK = 00100111, then array blocks 2, 5, 6, and 7 get erased when an erase operation is performed. 0 = array block M is not selected for program or erase 1 = array block M is selected for program or erase
24	—	Reserved
25	CSC	Censor set or clear. The CSC bit is write protected by the SES bit. CSC configures the UC3F EEPROM for setting or clearing CENSOR. If CSC = 1 then CENSOR is configured for setting if PE = 0 or clearing if PE = 1. When the CSC bit is set, the following bits in the UC3FMCR register are write-locked: $\overline{\text{LOCK}}$, FIC, ACCESS, SUPV, DATA, and PROTECT. 0 = configure for normal operation 1 = configure to set or clear the CENSOR bits
26:27	—	Reserved
28	HSUS	Program/erase suspend. Setting the HSUS bit during an embedded hardware algorithm program or erase operation will force the UC3F EEPROM to suspend the current program or erase. The UC3F EEPROM will maintain all information necessary to resume the suspended operation. Array reads are possible while HSUS = 1. However, array reads must be done to locations that are not being affected by the program/erase operation that is currently being suspended. The UC3F EEPROM will NOT prevent read accesses to those locations. Reads to those locations will result in UNKNOWN data. Writes to the HSUS bit only have effect while EHV = 1. The HSUS bit is write locked by EHV = 0. 0 = hardware program/erase behaves normally 1 = any current hardware program/erase is suspended
29	PE	Program or erase select. The PE bit is write protected by the SES bit. PE configures the UC3F EEPROM for programming or erasing. When PE = 0, the array is configured for programming and if SES = 1 the SIE bit will be write locked. When PE = 1, the array is configured for erasing and SES will not write lock the SIE bit. 0 = configure for program operation 1 = configure for erase operation

Table 20-5 UC3FCTL Bit Descriptions (Continued)

Bit(s)	Name	Description
30	SES	Start-end program or erase sequence. The SES bit is write protected by the STOP, HVS, and EHV bits. The SES bit is used to signal the start and end of a program or erase sequence. At the start of a program or erase sequence, SES is set (written to a 1). This will lock STOP, PROTEC, SBPROTECT, BLOCK, SBBLOCK, SBEN, CSC, and PE. If PE = 0 and SES = 1, SIE will be write locked. At this point, the UC3F EEPROM is ready to receive either the programming writes or the erase interlock write. NOTE: The erase interlock write is a write to any UC3F EEPROM array location after SES is set and PE = 1. If PE = 0 and SES = 1, writes to the UC3F array are programming writes. The first programming write sets the address of the location to be programmed, and the data written is captured into the program data latch for programming into the UC3F array. All programming writes after the first programming write update the program data latch but do not change the address to be programmed. At the end of the program or erase operation, the SES bit must be cleared (written to a 0) to return to normal operation and release the STOP, PROTECT, SBPROTECT, BLOCK, SBBLOCK, CSC, SBEN, and PE bits. 0 = UC3F EEPROM not configured for program or erase operation 1 = configure UC3F EEPROM for program or erase operation
31	EHV	Enable high voltage. EHV can be asserted only after the SES bit has been asserted and a valid programming write(s) or erase hardware interlock write has occurred. If an attempt is made to assert EHV when SES is negated, or if a valid programming write or erase hardware interlock write has not occurred since SES was asserted, EHV will remain negated. The external program or erase enable pin (EPEE) and EHV are used to control the application of the program or erase voltage to the UC3F EEPROM module. High voltage operations to the UC3F EEPROM array, special shadow locations or FLASH NVM registers can occur only if EHV = 1 and EPEE = 1. Only after the correct hardware and software interlocks have been applied to the UC3F EEPROM can EHV be set. Once EHV is set, SES cannot be changed and attempts to read the array will not be acknowledged. Clearing EHV during a program or erase operation will safely terminate the high voltage operation. If EHV is cleared while using the embedded hardware program/erase algorithm, the program/erase routine will abort the operation and exit normally. 0 = program or erase pulse disabled 1 = program or erase pulse enabled

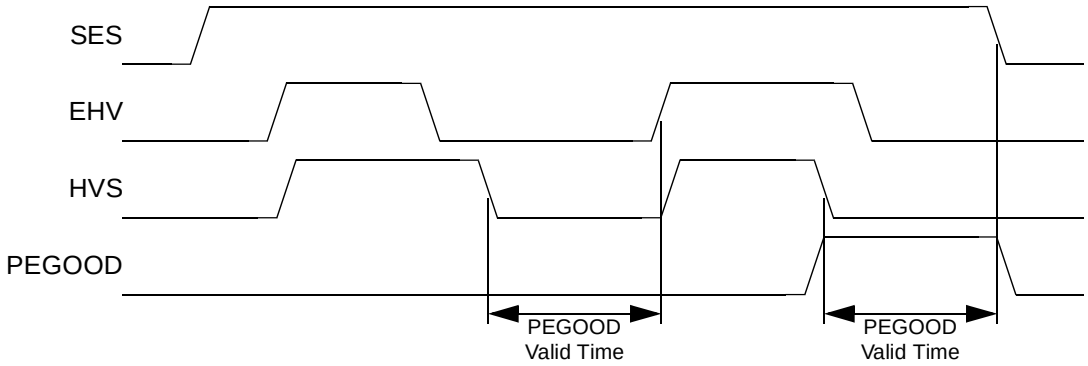


Figure 20-2 PEGOOD Valid Time

20.3.5 UC3F EEPROM Array Addressing

The mapping of the array in the MCU is determined by the address decoder in the BIU. The UC3F array is divided into a maximum of eight blocks, 64 Kbytes in size, which may be independently erased. Two blocks are host to a 16-Kbyte small block.

Seventeen bits of address are used to decode locations in the UC3F array. The read control logic in the UC3F EEPROM module decodes the upper 14 bits of that address to determine if the desired data is currently stored in one of the two read page buffers. If the data is already present in one of the two read page buffers, a read operation is not completed to the UC3F array core, and 64 bits of data are transferred from the appropriate read page buffer to the BIU. This type of array read access is an on-page read.

In the event that the read control logic determines that the desired data is not contained within one of the read page buffers, a read access to the UC3F array core is completed and 32 bytes of data are transferred from the array core. Only the addressed 64 bits of data will be transferred to the BIU. This type of array read access is an off-page read. The BIU contains logic to implement the read page buffer update and replacement scheme to transfer the 32 bytes of data into the appropriate read page buffer. If the read page update and replacement scheme contains a random access mode that does not update the read page buffers, the 32 bytes of data retrieved from the UC3F array core will not be transferred into either read page buffer. The BIU is expected to contain page update logic for controlling the updating of the read page buffers.

Write accesses to the UC3F array have no effect except during program and erase operation.

20.3.6 UC3F EEPROM Shadow Row

The UC3F EEPROM module contains a special shadow row that is used to hold reset configuration data and user data. See figure **Figure 20-3**.

The shadow row is accessed by setting SIE = 1 and performing normal array accesses. Upon transitioning SIE (a 1-to-0 or 0-to-1 transition), the read page match decode circuit is reset so that the next array access is an off-page access.

The shadow row contains 512 bytes which are addressed for read accesses using the low order row and read page addresses.

The shadow row is implemented in the lowest numbered block of the array. In the case of a UC3F array configuration which also has a small block in the lowest numbered block of the array, the shadow row is contained in the small block. If SBEN[0] = 1 in this array configuration, the shadow row is treated as part of small block 0. SBPROTECT[0] and SBBLOCK[0] are used to control program and erase operation of the shadow row. If SBEN[0] = 0 in this array configuration, the shadow row is treated as part of the host block. The corresponding PROTECT and BLOCK bits are used to control program and erase operation of the shadow row.

NOTE

A shadow row can not be read if the program code is executing from the same module of the MPC565 / MPC566. (i.e., If a program is executing from one flash module), it can only read the shadow row of the other flash module, for example a program running from module UC3F_A can read the shadow row of the UC3F_B module.

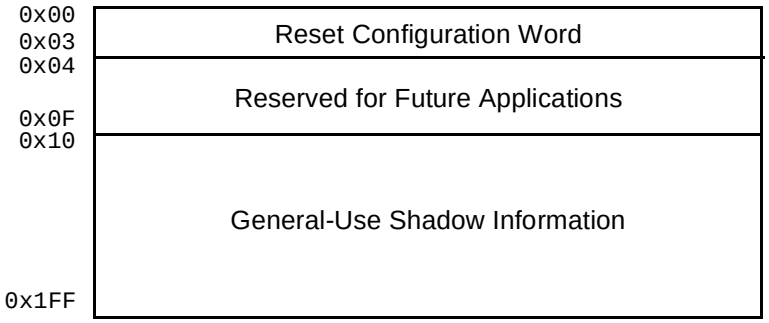


Figure 20-3 Shadow Information

20.3.6.1 Reset Configuration Word (UC3FCFIG)

The UC3F EEPROM reset configuration word is implemented in the first word (ADDR[23:29] = 0x00) of the special shadow locations. The reset configuration word along with the rest of the shadow information words is located in supervisor data address space. The purpose of the reset configuration word is to provide the system with an alternative internal source for the reset configuration.

Note that with the exception of bit 20, the bits in the UC3FCFIG are identical to those in the USIU hard reset configuration word.

UC3FCFIG — Hard Reset Configuration Word

MSB

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	IP	BDRV	BDIS	BPS[0:1]		Reserved			DBGC[0:1]		DBPC	ATWC	EBDF[0:1]		Re-served
Reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRPM	SC		ETRE	$\overline{HC}$	EN_COMP[1]	EXC_COMP[1]	Re-served	OERC		RESERVED		ISB			DME
Reset															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. Available only on the MPC566.

Table 20-6 RCW Bit Descriptions

Bit(s)	Name	Description
0	EARB	External Arbitration — Refer to **9.5.6 Arbitration Phase** for a detailed description of Bus arbitration. The default value is that internal arbitration hardware is used. 0 = Internal arbitration is performed 1 = External arbitration is assumed
1	IP	Initial Interrupt Prefix — This bit defines the initial value of the MSR_{IP} immediately after reset. The MSR_{IP} bit defines the Interrupt Table location. If IP is zero then the MSR_{IP} initial value is zero, If the IP is one, then the MSR_{IP} initial value is one. Default value is zero. See **Table 3-12** for more information. 0 = MSR[IP] = 0 after reset 1 = MSR[IP] = 1 after reset
2	BDRV	Bus Pins Drive Strength — This bit determines the bus pins (address, data and control) driving capability to be either full or reduced drive. The bus default drive strength is full; Upon default, it also effects the CLKOUT drive strength to be full. See **Table 6-7** for more information. BDRV controls the default state of COM[1] in the SIUMCR. 0 = Full drive 1 = Reduced drive
3	BDIS	Boot Disable — If the BDIS bit is set, then memory controller is not activated after reset. If it is cleared then the memory controller bank 0 is active immediately after reset such that it matches any addresses. If a write to the OR0 register occurs after reset this bit definition is ignored. The default value is that the memory controller is enabled to control the boot with the $\overline{CS}$0 pin. See **10.4 Global (Boot) Chip-Select Operation** for more information. 0 = Memory controller bank 0 is active and matches all addresses immediately after reset 1 = Memory controller is not activated after reset.
4:5	BPS	Boot Port Size — This field defines the port size of the boot device on reset (BR0[PS]). If a write to the OR0 register occurs after reset this field definition is ignored. See**Table 10-3** and **Table 10-7** for more information. 00 = 32-bit port (default) 01 = 8-bit port 10 = 16-bit port 11 = Reserved
6:8	—	Reserved. This bit must not be high in the reset configuration word.
9:10	DBGC[0:1]	Debug Pins Configuration — See **6.14.1.1 SIU Module Configuration Register (SIUMCR)** for this field definition. The default value is that these pins function as: VFLS[0:1], $\overline{BI}$, $\overline{BR}$, $\overline{BG}$ and $\overline{BB}$. See **Table 6-8**.
11	DBPC	Debug Port Pins Configuration — The default value is that these pins function as the debug port pins for development support. See **Table 6-9**.
12	ATWC	Address Type Write Enable Configuration — The default value is that these pins function as $\overline{WE}$ pins. 0 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as $\overline{WE}$[0:3]/$\overline{BE}$[0:3] 1 = $\overline{WE}$[0:3]/$\overline{BE}$[0:3]/AT[0:3] functions as AT[0:3] See **Table 6-7**.
13:14	EBDF	External Bus Division Factor — This field defines the initial value of the external bus frequency. The default value is that CLKOUT frequency is equal to that of the internal clock (no division). See **Table 8-9**.
15	—	Reserved. This bit must not be high in the reset configuration word.
16	PRPM	Peripheral Mode Enable — This bit determines if the chip is in peripheral mode. A detailed description is in **Table 6-14** The default value is no peripheral mode enabled.
17:18	SC	Single Chip Select — This field defines the mode of the MPC565 / MPC566. 00 = Extended chip, 32 bits data 01 = Extended chip, 16 bits data 10 = Single chip and show cycles (address) 11 = Single chip See **Table 6-11**.

Table 20-6 RCW Bit Descriptions (Continued)

Bit(s)	Name	Description
19	ETRE	Exception Table Relocation Enable — This field defines whether the Exception Table Relocation feature in the BBC is enabled or disabled; The default state for this field is disabled. For more details, see **Table 4-4**.
20	$\overline{HC}$	$\overline{\text{Has Configuration}}$ — During reset the $\overline{HC}$ bit ("has configuration," bit 20) and the USIU configure the UC3F EEPROM module to provide UC3FCFIG. If $\overline{HC}$ = 0 and the USIU requests internal configuration during reset the reset configuration word will be provided by UC3FCFIG Unlike the MPC555, the flash reset configuration word can come from either the UC3F_A or the UC3F_B module. See **7.5.2 Hard Reset Configuration Word** for additional information..
21	EN_ COMP[1]	Enable Compression — This bit enables the operation of the MPC566 with compressed code. The default state is disabled. See **Table 4-4**.
22	EXC_ COMP[1]	Exception Compression — This bit determines the operation of the MPC566 with exceptions. If this bit is set, than the MPC566 assumes that ALL the exception routines are in compressed code. The default indicates the exceptions are all non-compressed. See **Table 4-4**.
23	—	Reserved. This bit must not be high in the reset configuration word.
24:25	OERC	Other Exceptions Relocation Control — These bits effect only if ETRE was enabled. Relocation offset: 00 = Offset 0 01 = Offset 64 KB 10 = Offset 512 KB 11 = Offset to 0x003 FE000 (SRAM start address) See **Table 4-2**.
26:27	—	Reserved
28:30	ISB	Internal Space Base Select — This field defines the initial value of the ISB field in the IMMR register. A detailed description is in **Table 6-13**. The default state is that the internal memory map is mapped to start at address 0x0000_0000 hex. This bit must not be high in the reset configuration word.
31	DME	Dual Mapping Enable — This bit determines whether Dual mapping of the internal flash is enabled. For a detailed description refer to **Table 10-9**. The default state is that dual mapping is disabled. 0 = Dual mapping disabled 1 = Dual mapping enabled

NOTES:
1. This bit is available only on the MPC556.

The default reset state of the UC3FCFIG after an erase operation of the UC3F module A block 0 is no configuration word available ($\overline{HC}$ = 1).

20.3.7 UC3F EEPROM 512-Kbyte Array Configuration

The MPC565 / MPC566 contains only the 512-Kbyte array configuration of the UC3F module, as shown in **Figure 20-4**. The blue shaded blocks in the array configuration diagram indicate the location of the shadow row.

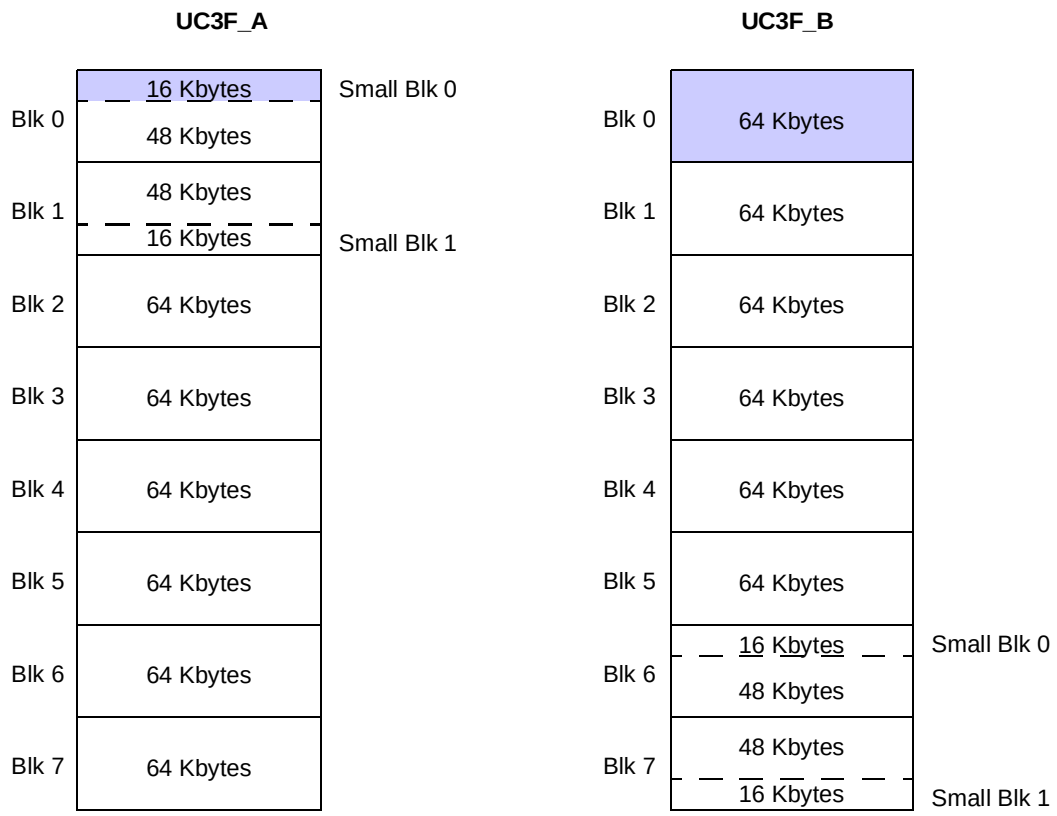


Figure 20-4 512-Kbyte Array Configurations

20.4 Operation

The following sections describe the operation of the UC3F EEPROM during various operational modes. The primary function of the UC3F EEPROM module is to serve as electrically erasable and programmable non-volatile memory for embedded application in microcontrollers.

20.4.1 Reset

The device shall signal a reset to the UC3F EEPROM by asserting the reset signal. A reset is the highest priority operation for the UC3F EEPROM and terminates all other operations. The UC3F EEPROM module uses reset to initialize register bits to their default reset value. If the UC3F EEPROM is in program or erase operation (EHV = 1 and SES = 1) and a reset is issued, the module will perform the needed interlocks to disable the high voltage without damage to the high voltage circuits. Reset terminates any other mode of operation and forces the UC3F EEPROM module to a state ready to receive accesses.

Upon power up and power down periods, it is assumed that the reset signal is asserted to prevent accidental program/erase disturb of the UC3F array.

20.4.2 Register Read and Write Operation

The UC3F EEPROM control registers are accessible for read or write operation at all times while the device is powered up and enabled except during reset.

20.4.3 Array Read Operation

The UC3F EEPROM array is available for read operation under most conditions while the device is powered up. Reads of the array are not allowed during reset, when in information or cleared censorship with ACCESS = 0, while the UC3F EEPROM is disabled (see section **20.4.10 Disabled** for more information on disabling the UC3F EEPROM), while the UC3F EEPROM is in STOP mode (see section **20.4.9 Stop Operation** for more information on STOP mode), or while high voltage is applied to the array during program and erase operation (HVS = 1 or EHV = 1 and not suspended). The UC3F array may be configured into a page mode access memory by setting the BPB bit in the UC3FMCR register, see **20.3.2 UC3F EEPROM Configuration Register (UC3FMCR)**.

When in page mode operation, the address of an incoming read access is compared to the address for which data is currently held in the read page buffers. If the data corresponding to the read address is currently held in one of the two read page buffers, the data is fetched from the appropriate read page buffer. A data fetch from a read page buffer is an on-page read operation **20.4.3.1 Array On-Page Read Operation**. If the data is not contained in one of the read page buffers, 32 bytes of information is fetched from the UC3F array, and the addressed data is driven onto the data bus. A data fetch from the UC3F array is an off-page read operation.

NOTE

After setting/clearing HSUS, reset, programming writes, erase interlock write, setting EHV, clearing SES or setting/clearing SIE, the page buffers may not contain valid information and the UC3F page match logic negates uc3f_onpage[0:1] to force an off page read before an on page read can be accomplished to ensure data coherency.

For information regarding how the two read page buffers in the UC3F EEPROM are associated to array blocks, refer to **20.3.5 UC3F EEPROM Array Addressing**.

The UC3F module is configured as a page mode memory. The UC3F module uses an internal address comparator to monitor incoming addresses to determine if the addressed information is stored in a read page buffer. When the address comparator determines that the requested information is not stored in a read page buffer, an array off-page read operation retrieves 32 bytes of data from the flash array and transfers the addressed data to the data bus.

In the MPC565 / MPC566, each UC3F module contains two 32-byte read page buffers. On module UC3F_A, these read page buffers contain the two most recently loaded off-page accessed instruction fetches. On module UC3F_B, one buffer is dedicated to the most recently accessed instruction fetches and the other read page buffer is dedicated to the most recently loaded data access.

20.4.3.1 Array On-Page Read Operation

An internal address comparator is used to determine if addressed information is stored in a read page buffer. If the address of a read access matches data contained in a read page buffer, that addressed data is transferred from the read page buffer to the data bus. An off-page read access to transfer data from the flash array to the data bus is not performed in this case.

20.4.4 Shadow Row Select Read Operation

The normal array is accessed when the SIE register bit in the UC3FMCR = 0. When SIE = 1, reads to the array access the shadow information row.

20.4.5 Array Program/Erase Interlock Write Operation

The only valid writes to the UC3F array are program or erase interlock writes. In the case of program interlock writes, the address of the write determines the location to be programmed while the data written is transferred to the program data latches to be programmed into the array. Address and data written during an erase interlock write is a “don’t care” and is not stored anywhere.

20.4.6 High Voltage Operations

There are two fundamental high voltage operations, program and erase. Program changes a UC3F array bitcell from a logic 1 state to a logic 0 state and is a selective operation performed on up to 32 bits at a time. Erase changes a UC3F array bitcell from a logic 0 state to a logic 1 state and is a bulk operation performed on one block or multiple blocks of the UC3F array.

20.4.6.1 Overview of Program/Erase Operation

The embedded hardware program/erase algorithm relies on an internal state controller to perform the program and erase sequences. The embedded hardware algorithm uses an internal oscillator to control the high voltage pulse duration and hardware control logic. The embedded hardware algorithm is also responsible for performing all margin reads and applying high voltage pulses to ensure each bit is programmed or erased with sufficient margin. Upon successful program or erase operation, the program/erase hardware control logic terminates the program or erase operation with a pass status (PEGOOD = 1). The program/erase control logic will time out in the event that the maximum program or erase time is exceeded and return a fail status (PEGOOD = 0).

20.4.7 Programming

To modify the charge stored in an isolated element of the UC3F bit from a logic 1 state to a logic 0 state, a programming operation is required. This programming operation shall apply the required voltages to change the charge state of the selected bits without changing the logic state of any other bits in the UC3F array. The program operation cannot change the logic 0 state to a logic 1 state; this transition must be done by the erase operation. Programming uses a program data latch to store the data to be pro-

grammed and an address latch to store the word address to be programmed. The UC3F Array may be programmed by byte (8 bits), half-word (16 bits), or word (32 bits).

Blocks of the UC3F EEPROM that are protected (PROTECT[M] = 1, SBEN[N] = 1 and SBPROTECT[N] = 1) will not be programmed. Also, if EPEE = 0, no programming voltages will be applied to the array. If B0EPEE = 0, no programming voltages will be applied to block 0 or small block 0 depending on the state of SBEN[0] and the configuration of the array.

In the event of UC3F module configurations which do not contain a block 0, the lowest numbered block is protected by B0EPEE. Also, if the lowest numbered block does not host small block 0, then no small blocks can be protected by B0EPEE.

20.4.7.1 Program Sequence

The UC3F EEPROM module requires a sequence of writes to the high voltage control register (UC3FCTL) and to the program data latch in order to enable the high voltage to the array or shadow information for program operation. The required hardware program sequence follows.

1. Write PROTECT[0:7] and SBPROTECT[0:1] to disable protection on blocks to be programmed.
2. Write BLOCK[0:7] and SBBLOCK[0:1] to select the array blocks to be programmed, SES = 1 and PE = 0 in the UC3FCTL register.

NOTE

BLOCK[0:7] and SBBLOCK[0:1] in conjunction with SBEN[0:1] determine which blocks/small blocks in the array are enabled for programming operation. Just because a BLOCK or SBBLOCK bit is enabled (set to 1), no programming can occur in the corresponding block/small block unless the programming operation specifically targets an address location within that block/small block to program. If BLOCK or SBBLOCK is not set to 1, no address locations in that corresponding block or small block can be programmed.

3. Programming write — A successful write to the array location to be programmed. This write shall update the program data latch with the information to be programmed. In addition, the address, uc3f_addr[0:16], of the first programming write is latched in the UC3F memory interface block. All access of the array after the first write shall be to the same address regardless of the address provided. Thus the locations accessed after the first programming write are limited to the location to be programmed. The last write to the program data latch shall be saved for programming.

NOTE

If a byte of the program data latch has not received a programming write, no programming voltages will be applied to the corresponding byte in the array. Once EHV has been set, writes to the program data latch are disabled until EHV is cleared to 0.

4. Write EHV = 1 in the UC3FCTL register.

NOTE

The values of the EPEE and B0EPEE inputs are latched with the assertion of EHV to determine the array protection state for the program operation. It is assumed that the EPEE and B0EPEE inputs are setup prior to the assertion of EHV.

5. Read the UC3FCTL register until HVS = 0.
6. Read the UC3FCTL, confirm PEGOOD = 1.
7. Write EHV = 0.

WARNING

Writing EHV = 0 before HVS = 0 causes the current program sequence to ABORT. The location for which the program sequence was aborted may not have been programmed with sufficient margins. The block containing that location must be erased and reprogrammed before that block of the UC3F array may be used reliably.

8. If more information needs to be programmed go to step 3.
9. Write SES = 0 in the UC3FCTL register.

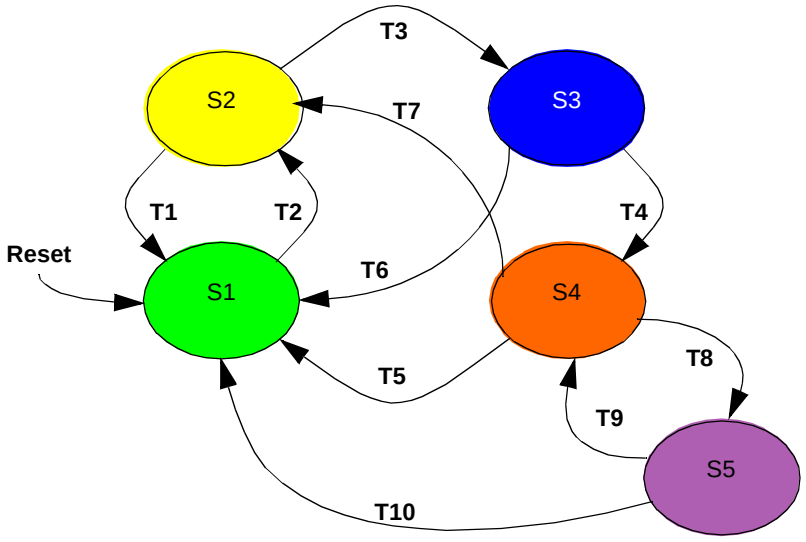


Figure 20-5 Program State Diagram

Table 20-7 Program Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses. The block protect information can be modified.	S2	T2	Write PE = 0, SES = 1.
S2	First Program Hardware Interlock Write: Normal read operation still occurs. The array will accept programming writes. Accesses to the registers are normal register accesses. A write to UC3FCTL can not change EHV at this time. If the write is to a register no data will be stored in the program data latch and the UC3F shall remain in state S2.	S1	T1	Write SES = 0 or a reset.
		S3	T3	Hardware Interlock A successful write to any UC3F array location. This programming write will latch the selected word of data into the program data latch and the address shall be latched to select the location that will be programmed. Once a bit has been written then it shall remain in the program data latch until another write over-writes that data or a write of SES = 0. If the write is to a register no data will be stored in the program data latch and the UC3F shall remain in state S2.
S3	Expanded Program Hardware Interlock Operation: Programming writes are accepted so that data may be programmed. These writes may be to any UC3F array location. The location to be programmed is determined from the address initially written to on the first program interlock write. The program data latch may be updated on any program interlock writes which occur in this state. Accesses to the registers are normal register accesses. A write to UC3FCTL can change EHV. If the write is to a register no data will be stored in the program data latch.	S1	T6	Write SES = 0 or a reset.
		S4	T4	Write EHV = 1.
S4	Program Operation: High voltage is applied to the array or shadow information to program the UC3F bit cells, and program margin reads are automatically performed by the internal program control logic. No further programming writes will be accepted. During programming, the array will not respond to any access. Accesses to the registers are allowed. A write to UC3FCTL can change EHV or HSUS only.	S1	T5	Reset.
		S2	T7	Write EHV = 0.
		S5	T8	Write HSUS = 1 or disable the UC3F module.
S5	Program Suspend Operation: The program operation is suspended to either read the array or disable the module. Once HVS reads as a 0, the program operation is suspended. Normal reads to the array can be performed if the module is enabled; read accesses to the location being programmed returns indeterminate data.	S1	T10	Reset.
		S4	T9	Write HSUS = 0 or re-enable the UC3F module.

20.4.7.2 Program Shadow Information

Programming the shadow information uses the same procedure as programming the array except that SIE must be set to a 1 prior to initiating the programming sequence. Only the lower most addresses are used to encode which words get programmed in the shadow row. the shadow information is physically located in lowest numbered block and will also be located in small block 0 if the lowest numbered block hosts a small block in the implemented configuration.

20.4.7.3 Program Suspend

The program operation may be suspended to allow read accesses to the array. Setting the HSUS bit in the UC3FCTL to a 1 while PE = 0, EHV = 1, and HVS = 1 forces the array into a program suspend state. The deassertion of the HVS bit (HVS = 0) signifies that the program operation has been successfully suspended. The HVS bit should negate within 10µs of asserting the HSUS bit.

While in program suspend mode, normal read accesses may be performed to the UC3F array or shadow information words. Reads to the array location targeted for program return indeterminate data since only a partial programming operation may have been performed.

The program operation may be resumed by setting HSUS = 0.

NOTE

Repeated suspending of a program operation to fetch array contents may extend the program operation. The internal program hardware may only resume the program operation at predefined steps of the internal program hardware sequence; interrupting the program operation on a high frequency basis may cause the internal program hardware to delay completion of the current step and delay advancement to the next step of the internal program hardware sequence.

20.4.8 Erasing

To modify the charge stored in an isolated element of the UC3F bit from a logic 0 state to a logic 1 state, an erase operation is required. In the UC3F EEPROM, erase is a bulk operation that shall affect the stored charge of all the isolated elements in an array block. To make the UC3F module block-erasable, the array is divided into blocks that are physically isolated from each other. Each of the array blocks may be erased in isolation or in any combination. The UC3F array block size is fixed for all blocks in the module at 64 Kbytes and the module is comprised of eight blocks. Two of these blocks may be further subdivided into two small blocks. Array blocks of the UC3F EEPROM that are protected (PROTECT[M] = 1 or (SBEN[M] = 1 & SBPROTECT[M] = 1)) will not be erased. Also, if EPEE = 0 or B0EPEE = 0, no erase voltages will be applied to the array or the block corresponding to block 0 (or lowest numbered block if no block 0 exists) or small block 0 (assuming that the lowest numbered block contains small block 0) if SBEN[0] = 1.

The embedded program/erase algorithm first preprograms all bits in blocks selected for erase prior to actually erasing the selected blocks.

The array blocks selected for erase operation are determined by BLOCK[0:7], SBBLOCK[0:1] in conjunction with SBEN[0:1], and the array configuration. If multiple blocks are selected for erase, the embedded erase hardware algorithm serially erases each array block until all of the selected blocks are erased. For instance, if BLOCK[0:7] = 0x78 and SBEN[0:1] = 0b00, then blocks 1, 2, 3, and 4 are selected for erase. The embedded erase hardware algorithm first erases block 1 and then erases block 2 followed by blocks 3 and 4. The total erase time for this example is the block erase time, T_{ERASE}, multiplied by four since four blocks are erased. In addition, the preprogramming time to program all locations in blocks 1, 2, 3, and 4 to a “0” state needs to be considered when determining the total erase time. The preprogramming time is dependent on the data already stored in the flash array before beginning the erase operation.

20.4.8.1 Erase Sequence

The UC3F EEPROM module requires a sequence of writes to the high voltage control register (UC3FCTL) and an erase interlock write in order to enable high voltage to the array and shadow information for erase operation. The required hardware algorithm erase sequence follows.

1. Write PROTECT[0:7] and SBPROTECT[0:1] to disable protect for the blocks to be erased.
2. Write BLOCK[0:7] and SBBLOCK[0:1] to select the blocks to be erased, PE = 1 and SES = 1 in the UC3FCTL register.

NOTE

BLOCK[0:7] and SBBLOCK[0:1] in conjunction with SBEN[0:1] determine which blocks are selected for erase. Blocks whose BLOCK bits or enabled small blocks whose SBBLOCK bits are set (equal to 1) get erased when an erase operation is performed.

3. Execute an erase interlock write to any UC3F array location.
4. Write EHV = 1 in the UC3FCTL register.

NOTE

The values of the EPEE and B0EPEE inputs are latched with the assertion of EHV to determine the array protection state for the erase operation. It is assumed that the EPEE and B0EPEE inputs are setup prior to the assertion of EHV.

5. Read the UC3FCTL register until HVS = 0.

WARNING

Writing EHV = 0 before HVS = 0 causes the current erase sequence to ABORT. All blocks being erased must go through another erase sequence before the UC3F EEPROM can be used reliably.

6. Read the UC3FCTL register. Confirm PEGOOD =1.
7. Write EHV = 0 in the UC3FCTL register.
8. Write SES =0 in the UC3FCTL register.

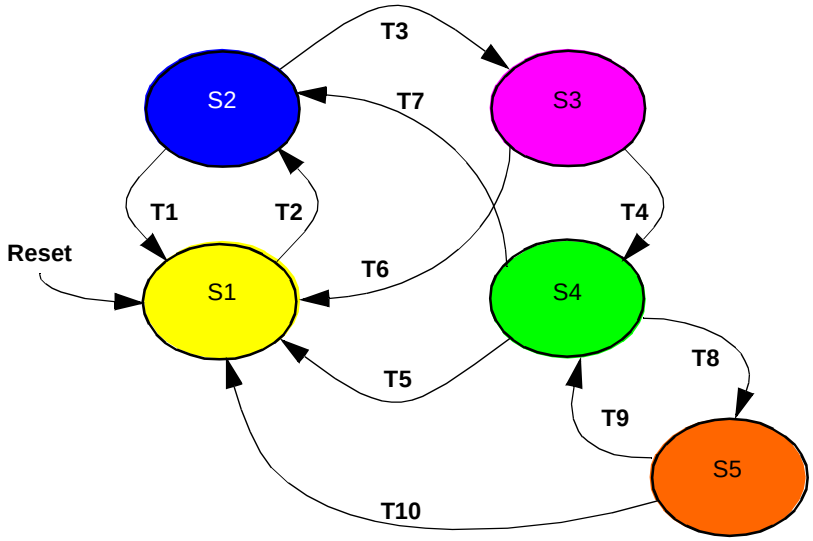


Figure 20-6 Erase State Diagram

Table 20-8 Erase Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses. The Block protect information can be modified.	S2	T2	Write PE = 1, SES = 1.
S2	Erase Hardware Interlock Write: Normal read operation still occurs. The UC3F will accept the erase hardware interlock write. This write may be to any UC3F array location. Accesses to the registers are normal register accesses. A write to UC3FCTL can not set EHV at this time. A write to the register is not an erase hardware interlock write and the UC3F shall remain in state S2.	S1	T1	Write SES = 0 or a reset.
		S3	T3	Hardware Interlock A successful write to any UC3F array location is the erase interlock write. If the write is to a register the erase hardware interlock write has not been done and the UC3F shall remain in state S2.
S3	High voltage write enable Accesses to the registers are normal register accesses. A write to UC3FCTL can change SES or EHV.	S1	T6	Write SES = 0 or a reset.
		S4	T4	Write EHV = 1.

Table 20-8 Erase Interlock State Descriptions (Continued)

State	Mode	Next State	Transition Requirement	
S4	Erase Operation: High voltage is applied to the array blocks to erase the UC3F bit cells, and erase margin reads are automatically performed by the embedded erase control logic. During erase the array will not respond to any address. Accesses to the registers are allowed. A write to UC3FCTL can change EHV or HSUS only.	S1	T5	Reset.
		S2	T7	Write EHV = 0.
		S5	T8	Write HSUS = 1 or disable the UC3F module.
S5	Erase Suspend Operation: The erase operation is suspended to either read the array or disable the module. Once HVS reads as a 0, the erase operation is suspended. Normal reads to the array can be performed if the module is enabled; read accesses to locations in blocks being erased return indeterminate data.	S1	T10	Reset.
		S4	T9	Write HSUS = 0 or re-enable the UC3F module.

20.4.8.2 Erasing Shadow Information Words

The shadow information words are erased with either the lowest numbered block or small block 0, depending on the array configuration and the state of SBEN[0]. If the lowest numbered block in the array does not host a small block, then the shadow information words are erased with the lowest numbered block. If the lowest numbered block hosts a small block, then the shadow information words may get erased with small block 0. If SBEN[0] = 0 for this array configuration, then the shadow information words get erased with the lowest numbered block. If SBEN[0] = 1 for this same array configuration, then the shadow information words get erased with small block 0 only.

20.4.8.3 Erase Suspend

The erase operation may be suspended to allow read accesses to the array. Setting the HSUS bit in the UC3FCTL to a 1 while EHV=1 and HVS=1 forces the array into an erase suspend state. The deassertion of the HVS bit (HVS = 0) signifies that the erase operation has been successfully suspended. The HVS bit should negate within 10 ms of asserting the HSUS bit.

While in erase suspend mode, normal read accesses may be performed to the UC3F array or shadow information words. Reads to the array block or blocks targeted for erase return indeterminate data since only a partial erase operation has been performed.

The erase operation may be resumed by setting HSUS = 0.

NOTE

Repeated suspending of an erase operation to fetch array contents may severely extend the erase operation. The internal erase hardware may only resume the erase operation at predefined steps of the internal erase hardware sequence; interrupting the erase operation on a high frequency basis may cause the internal erase hardware to delay completion of the current step and delay advancement to the next step of the internal erase hardware sequence.

20.4.9 Stop Operation

The UC3F EEPROM goes into a low power operation, or stop operation, while STOP = 1. When the STOP bit is set, only the control registers can be accessed on the UC3F EEPROM module. The UC3F EEPROM array may not be programmed, erased or read while STOP = 1.

With STOP = 1, the UC3F module enters a low power state by shutting down internal timers and bias generators. A stop recovery time of 1 µs is required when clearing the STOP bit to exit stop operation. The BIU should allow 1 µs following the negation of the STOP bit so that internal bias generators used by the array may recover to normal levels prior to initiating any UC3F array accesses.

NOTE

The UC3F cannot be stopped while the array is being programmed or erased since the STOP bit is write locked by SES = 1.

20.4.10 Disabled

Both UC3F modules can be disabled by clearing the FLEN bit in the IMMR register (see **6.14 System Configuration and Protection Registers**). While disabled, the UC3F module is completely shut down. The register block and array are not accessible in this mode, and all circuits which draw any DC power are disabled to eliminate power consumption. In addition, each individual module can be disabled by setting the STOP bit in the UC3FMCR register (see **20.3.2 UC3F EEPROM Configuration Register (UC3FMCR)**).

If the UC3F module is disabled while programming or erasing, the HSUS bit in the UC3FCTL register is asserted (HSUS = 1) to suspend the current program or erase operation. When the UC3F module is re-enabled, the suspended program or erase operation may be resumed by writing the HSUS bit to a 0.

NOTE

While there should be no harmful side effects resulting from disabling the UC3F module while in program or erase operation, it is not recommended that program or erase operation be suspended in this manner.

When disabled, the power used by the UC3F is reduced to leakage levels, see Power Supplies Section.

When not disabled, the UC3F module is enabled for accesses. Like recovering from stop operation (STOP = 1), there is a recovery time of one µs for internal biases to recover to operating levels.

20.4.11 Censored Accesses and Non-Censored Accesses

The UC3F EEPROM has a censorship mechanism which provides for several censorship levels. The censorship mechanism is used to increase restrictions in accessing flash data. Four bits in UC3FMCR are used to configure the UC3F censorship level. These bits are:

ACCESS	Enables a UC3F EEPROM to bypass the censorship.
FIC	Overrides CENSOR[0:1] to force information censorship.
CENSOR[0:1]	Determine the censorship level of the UC3F.

The device has two relevant modes used by the UC3F EEPROM to select the type of censorship. The first mode, which is uncensored mode, provides no censorship. In uncensored mode the ACCESS and CENSOR[0:1] bits are irrelevant. The second mode, censored mode, enables the UC3F EEPROM to exercise censorship based on the state of ACCESS, FIC, and CENSOR[0:1]. The device shall authenticate between uncensored mode and censored mode. In censored mode, a UC3F EEPROM may disallow accesses to the array. If censored mode is entered by any means then the UC3F EEPROM will exercise censorship according to the following.

ACCESS	FIC	CENSOR[0:1]	
0	0	11	Information censorship, No UC3F array accesses allowed.
0	0	01 or 10	No censorship, UC3F array accesses allowed.
0	0	00	Cleared censorship, No UC3F array accesses allowed.
0	1	XX	Emulated censorship, UC3F array accesses not allowed.
1	X	XX	No censorship, UC3F array accesses allowed.

While the device remains in the uncensored mode, ACCESS may be set to allow the device to enter censored mode and still access the UC3F array. ACCESS may not be set while the device is in censored mode but may be cleared.

Device Mode	Censored							Uncensored			
ACCESS	0					1		0		1	
FIC	0			1		0	1	0	1	0	1
CENSOR[0:1]	00	01 or 10	11	00, 01 or 10	11	00, 01, 10 or 11					
UC3F EEPROM Status	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11

Indicates that the UC3F array can not be accessed.

1.	ACCESS can not be changed. FIC can be set. UC3F array can not be accessed. CENSOR[0:1] can be set. CENSOR[0:1] can not be cleared.
2.	ACCESS can not be changed. FIC can be set. UC3F array can be accessed. CENSOR[0:1] can be set. CENSOR[0:1] can be cleared.
3.	ACCESS can not be changed. FIC can be set. UC3F array can not be accessed. CENSOR[0:1] can not be cleared unless uc3f_ccmcreiwren = 1.
4.	ACCESS can not be changed. FIC can not be changed. UC3F array can not be accessed. CENSOR[0:1] can be set. CENSOR[0:1] can not be cleared unless uc3f_ccmcreiwren = 1.
5.	ACCESS can not be changed. FIC can not be changed. UC3F array can not be accessed. CENSOR[0:1] can not be cleared unless uc3f_ccmcreiwren = 1.
6.	ACCESS can be cleared. FIC can be set. UC3F array can be accessed. CENSOR[0:1] can be changed.
7.	ACCESS can be cleared. FIC can not be changed. UC3F array can be accessed. CENSOR[0:1] can be changed.
8.	ACCESS can be changed. FIC can be set. UC3F array can be accessed. CENSOR[0:1] can be changed.
9.	ACCESS can be changed. FIC can not be changed. UC3F array can be accessed. CENSOR[0:1] can not be changed unless uc3f_ccmcreiwren = 1.
10.	ACCESS can be changed. FIC can be set. UC3F array can be accessed. CENSOR[0:1] can be changed.
11.	ACCESS can be changed. FIC can not be changed. UC3F array can be accessed. CENSOR[0:1] can be changed.

The only way CENSOR[0:1] can be changed is by setting or clearing the FLASH NVM fuses. In the information censorship state, CENSOR[0:1] must be cleared to the cleared censorship state before CENSOR[0:1] can be put into the no censorship state. While clearing CENSOR[0:1] the entire UC3F array is erased. Thus the information stored in the UC3F array is made invalid while clearing CENSOR[0:1].

20.4.11.1 Setting and Clearing Censor

The value of each bit in CENSOR[0:1] is determined by the state of an NVM CAM cell. The NVM CAM cell is not writable but instead may be set or cleared. Reading CENSOR[0:1] while setting or clearing with the high voltage applied (CSC = 1 and EHV = 1) will return 0's.

20.4.11.2 Setting Censor

The set operation changes the state in an NVM CAM cell from a zero to a one. This set operation can be done without changing the contents of the UC3F array. The required sequence to set one or both of the bits in CENSOR[0:1] follows.

1. Write CSC = 1, PE = 0 and SES = 1 in the UC3FCTL register
2. Write a 1 to the CENSOR bit(s) to be set
3. Write EHV = 1 in the UC3FCTL register
4. Read the UC3FCTL register until HVS = 0
5. Read the UC3FCTL register. Confirm PEGOOD = 1
6. Write EHV = 0 in the UC3FCTL register
7. Write SES = 0 and CSC = 0

20.4.11.3 Clearing Censor

The clear operation changes the state of the CENSOR[0:1] bits from a one to a zero by erasing the CAM cells. This clear operation can be done only while erasing the entire UC3F array and shadow information. The required sequence to clear CENSOR follows.

Clear CENSOR[0:1]

1. Write PROTECT[0:7] = 0x00 to enable the entire array for erase. If SBEN[M] = 1, then SBPROTECT[M] must also be set to 0.
2. Write BLOCK[0:7] = 0xFF, CSC = 1, PE = 1 and SES = 1 in the UC3FCTL register. If SBEN[M] = 1, then SBBLOCK[M] must also be set to 1.
3. Do an erase interlock write.

The erase interlock write is normally defined as a write to any valid array location and is subject to any censorship conditions which might apply.

When the UC3F module interface signal, uc3f_ccmcreiwren, is maintained in a logic 1 state, a write to the UC3FMCR also serves as an erase interlock write for the clear CENSOR operation, in addition to a write to any valid array location. When uc3f_ccmcreiwren = 1, the CENSOR[0:1] bits may always be cleared in the UC3F EEPROM status states #3, #4, #5, #9, and #11, from the above table, in addition to those states (state #1 is already cleared) where writes to the array are valid interlock writes.

When uc3f_ccmcreiwren = 0, only a write to any valid array location serves as the erase interlock write for the clear CENSOR operation.

The erase interlock write is only valid if all blocks of the array are selected for erase and not protected. BLOCK[0:7] and SBBLOCK[0:1] set to 1 in addition to PROTECT[0:7] and SBPROTECT[0:1] set to 0 are required to validate the erase interlock write during the clear censor operation.

4. Write EHV = 1 in the UC3FCTL register.

NOTE

The values of the EPEE and B0EPEE inputs are latched with the assertion of EHV to determine the array protection state for the clear censor operation. It is assumed that the EPEE and B0EPEE inputs are setup prior to the assertion of EHV. If EPEE and B0EPEE are not enabled for erase, the CENSOR[0:1] bits may not be cleared.

5. Read the UC3FCTL register until HVS = 0.
6. Read the UC3FCTL register. Confirm PEGOOD = 1.
7. Write EHV = 0 in the UC3FCTL register.
8. Write SES = 0 and CSC = 0.

20.4.11.4 Switching The UC3F EEPROM Censorship

There are three levels of censorship that CENSOR[0:1] can select. These are: cleared censorship, no censorship (two states) and information censorship. These three levels, state values, transitions and level of censorship are shown in the following diagram.

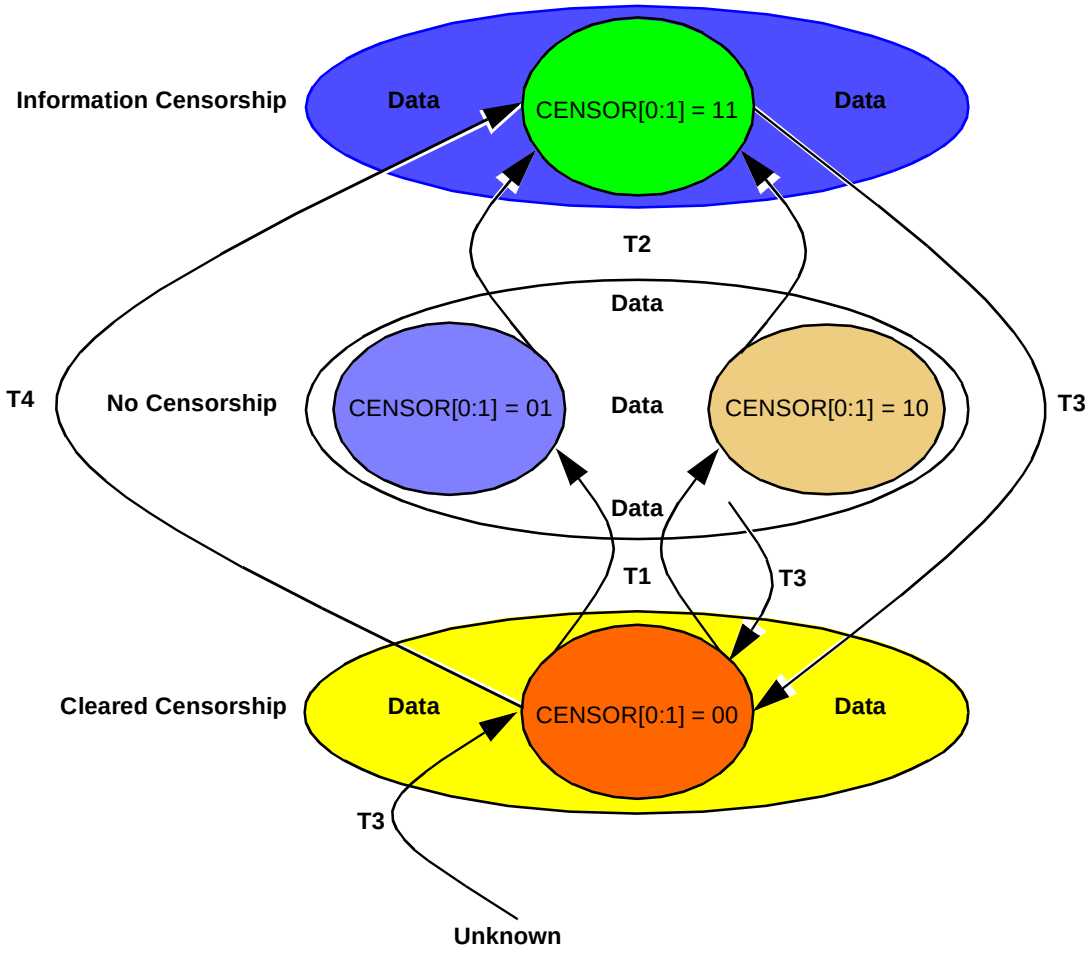


Figure 20-7 Censorship States and Transitions

CENSOR[0:1] transitions are listed as follows:

1. Cleared censorship to no censorship, T1
   Set CENSOR[0] or CENSOR[1].
2. No censorship to information censorship, T2
   Set CENSOR[0] and CENSOR[1].
3. Information censorship, no censorship or unknown to cleared censorship, T3
   Clear CENSOR[0:1]. This is done only while the entire UC3F array is erased.
4. Cleared censorship to information censorship, T4
   Set both CENSOR[0] and CENSOR[1].

20.4.12 Background Debug Mode or Freeze Operation

While in background debug mode, the UC3F should respond normally to accesses except that $\overline{\text{LOCK}}$ is writable.

SECTION 21
CALRAM OPERATION

21.1 Definitions and Acronyms

- CALRAM — The module name for the calibration RAM
- L2U — The module name for the L-bus to U-bus interface unit
- USIU / SIU — Unified system integration unit / system integration unit
- READI — The module name for real-time embedded application development interface (READI)
- BIU — Bus interface unit
- Array — Array indicates the RAM array; this is sometimes used to distinguish from CALRAM internal registers.
- Reserved — Memory locations and register bit fields that are unimplemented. This term is used interchangeably with unimplemented register bit fields.
- Bus access — Synonymous with bus cycle and bus transaction. Refers to the sequence of events on a bus and are defined by a bus protocol.
- RCPU — RISC CPU
- MSR — Machine status register is a register which defines the state of the processor. This register resides in the RCPU core.
- PPC register — PowerPC^{TM} special purpose register (SPR) defined by the PowerPC^{TM} architecture
- Byte — Eight bits of data form a byte
- Half-word — 16 bits of data form a half-word
- Word —32 bits of data form a word
- Calibration — Calibration tuning in engine control is a means of optimization of parameters in real time. The parameters reside in the flash.
- RAM overlay — During calibration mode, it is necessary to change the data within the calibration data region. The calibration data resides in the flash, which makes it very difficult to change this data in normal operation. The concept of RAM overlay is that the portion(s) of CALRAM map to calibration data regions of the flash address space. Hence, when overlay is enabled, data for these regions are driven by the CALRAM rather than by the flash.
- One-cycle — Equivalent to one clock period (1/system clock frequency)
- Two-cycle — Equivalent to two clock periods (2/system clock frequency)
- Freeze mode — Special mode used during debug
- Supervisor/user — L-bus signal indicating whether an access is a supervisor (privileged access) or user access
- Data/instruction — L-bus signal indicating whether data read or instruction fetch
- Machine check/data storage interrupt — Type of exceptions that can be generated due to an access causing an error. See the ***RCPU Reference Manual* (RCPURM/AD)** for details.
- Hex address — Addresses in hexadecimal numbers unless otherwise stated

21.1.1 Key Feature List

Standard RAM features are listed below:

- One-clock accesses
- Byte, half-word (16-bits), or word (32-bit) read/write accesses
- Each 8-Kbyte block has individual protection control bits.
- Two-cycle access for power savings
- Low power standby operation for data retention

Special overlay features are:

- Eight overlay regions; each can be programmed to be four, 16, 32, 64, 128, 256, or 512 bytes long
- Each overlay region size can be forced to be of 4 bytes long
- Data driven from the CALRAM module for overly access has the same timing as the data that would have come from the U-bus flash
- Overlay is for data read from the U-bus flash space and does not affect instruction fetches from the flash
- Overlay block is naturally aligned
  — For example, 128-byte block is 128-byte aligned
- Normal access to overlaid portion of CALRAM array can be made to generate an error (machine check) if so configured

21.2 CALRAM Introduction

The calibration static random access memory (CALRAM) module provides the micro-controller unit (MCU) with a general purpose memory which may be read from or written to as either bytes, half-words, or words. In addition to this, a portion of the CAL-RAM called the overlay region can be used for calibration. Calibration in this context is defined as overlaying portions of the U-bus flash with a portion of the CALRAM array. During normal flash access, the PowerPC core reads data from U-bus flash (through L-bus and L2U), as shown in **Figure 21-1**. During calibration access, instead of flash providing the data, the overlay regions of CALRAM provide the data to the PowerPC core.

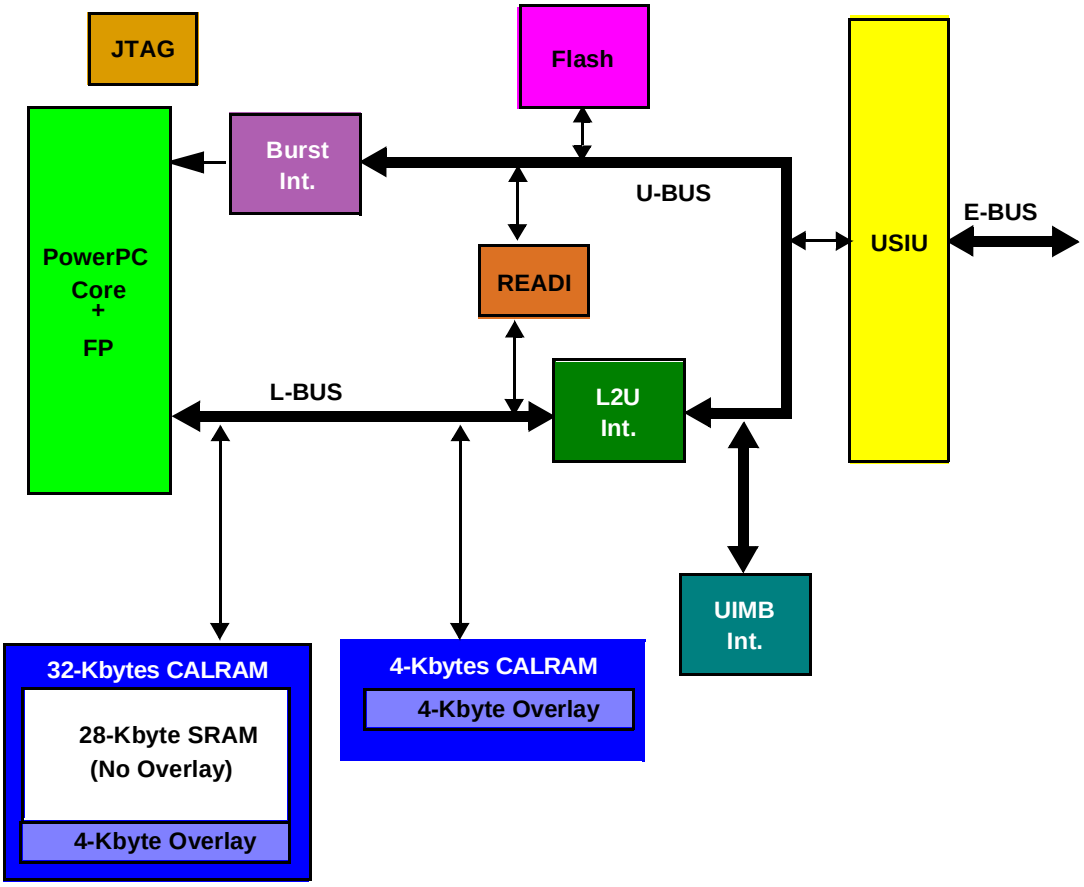


Figure 21-1 System Block Diagram

The MPC565 / MPC566 chip internal memory map is shown in **Figure 21-2**.

The internal memory block can reside in one of eight possible 4-Mbyte memory spaces. This 32-Mbyte (8 x 4-Mbyte) memory blocks starts at address 0x0000 0000. There is a user-programmable register (internal memory space (ISB) bits in internal memory map register (IMMR)) to configure the internal memory map to one of the eight possible locations. All addresses shown in this chapter are the default 4-Mbyte address from 0x00 0000 to 0x3F FFFF.

The CALRAM module is divided into two sections.

- Control section:
  — Includes all the registers in the CALRAM module
- Array sub-region:
  — Contains memory arrays

The MPC565 / MPC566 contains two CALRAM modules — one is a 32-Kbyte memory (at address 0x3F 8000 – 0x3F FFFF) and another is a 4-Kbyte memory (at address 0x3F 7000 – 0x3F 7FFF) as shown in **Figure 21-1** in and at address spaces shown in

Figure 21-2. In addition, each module is assigned 16 32-bit register address spaces — 12 registers are implemented and four registers are unimplemented. The 12 registers are-- one module configuration register (CRAMMCR), one register reserved for factory test, eight region base address (CRAM_RBAx) registers, one overlay configuration register (CRAMOVLCR), and one ownership trace register (CALRAM_OTR) to support a separate module called READI. Refer to **SECTION 23 READI MODULE**.

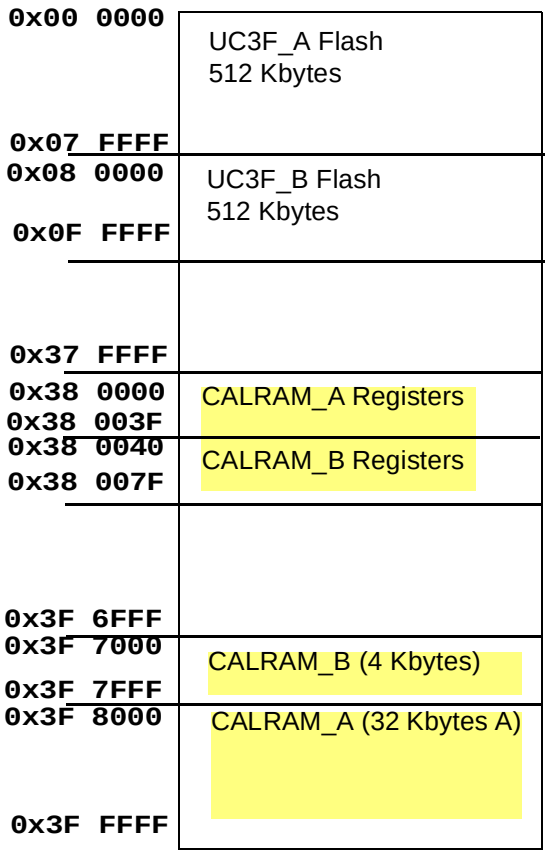


Figure 21-2 MPC565 / MPC566 Memory Map with CALRAM Address Ranges

When the normal chip power (VDD) is off, portions of the CALRAM array can be powered by separate power supply sources (VDDSRAM1/VDDSRAM2) as shown in **Figure 21-3**, thus allowing the data to be retained.

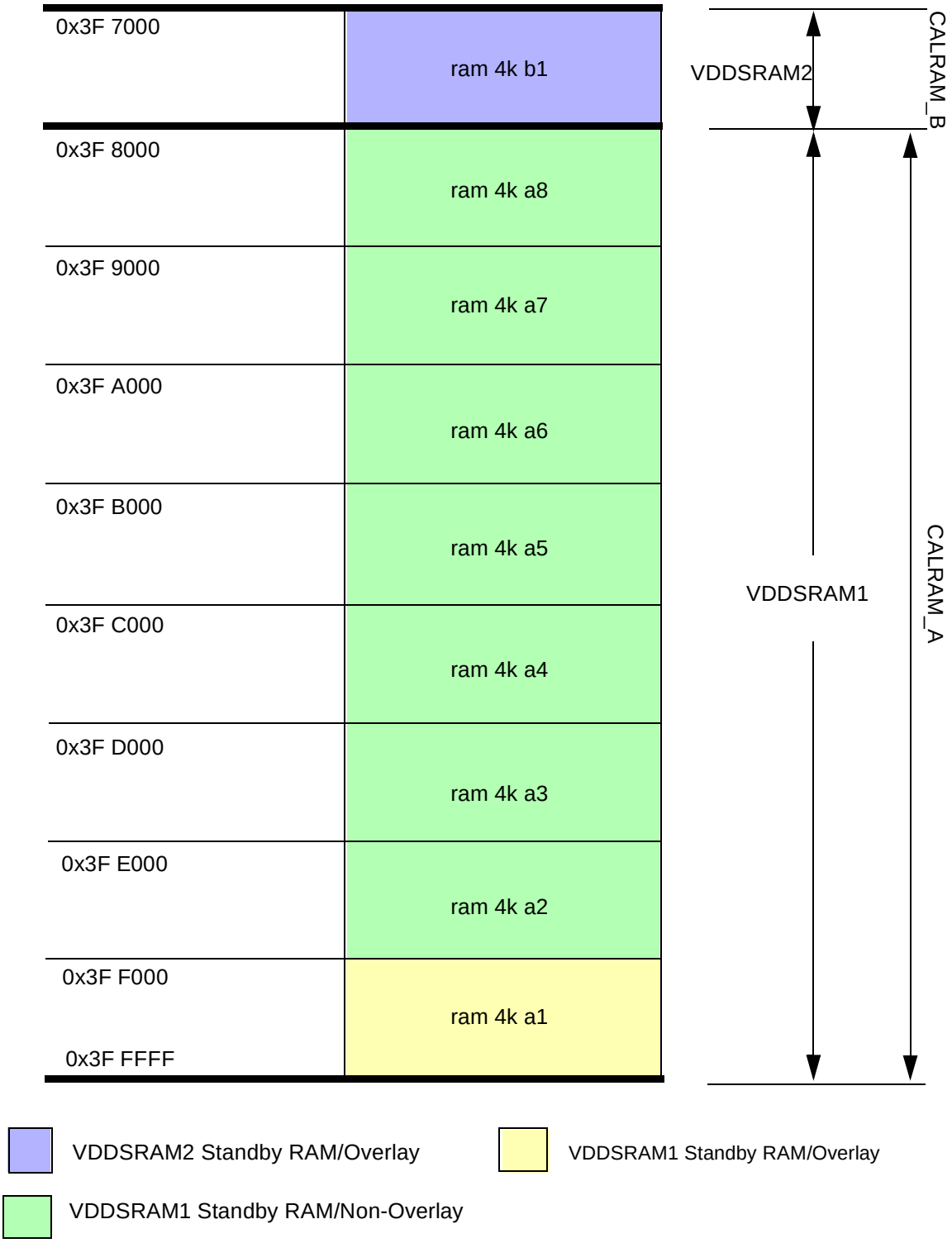


Figure 21-3 Standby Power Supply Configuration for CALRAM Array

21.3 Modes of Operation

The CALRAM module has the following modes of operation:

- Reset
- One-cycle
- Two-cycle
- Standby
- Stop
- Overlay

21.3.1 Reset

Reset configures the CALRAM module and resets some of the bits in the CALRAM registers to their default reset state. Some register bits are unaffected by reset. See section **21.4 Register Definitions**.

21.3.2 One-Cycle Mode

The CALRAM registers and array may be accessed for reads or writes as byte, (aligned) half-word, or word. This mode is the default mode of operation and, as the name suggests, the access time to the array and the internal registers for reads and writes is one cycle. Thus the one-cycle mode is used for high performance although it consumes more power than the two cycle mode.

21.3.2.1 CALRAM Access/Privilege Violations

Each 8-Kbyte CALRAM array can be assigned read-only, data-only, or supervisor-only privilege if data relocate (DR) bit in MSR is set. All CALRAM registers are assigned supervisor-only and data-only privilege. A privilege violation causes an error. See section **21.4.1 CALRAM Module Configuration Register (CRAMMCR)**.

Each CALRAM module can have up to 32 Kbytes of array. If less than 32 Kbytes of array are implemented in a CALRAM module, an attempt to access unimplemented portion of array causes an error. For example, if only four Kbytes of CALRAM_B implemented in MPC565 / MPC566, access to the remaining 28-Kbyte region of CALRAM_B causes an error; that is, any accesses to the array from 0x3F 0000 to 0x3F 6FFF end in an error. Each CALRAM module is allocated 16 register spaces among which only twelve registers are implemented. An attempt to access any of the four unimplemented reserved registers causes an error and returns 0's on the data bus for a read access.

If an error condition occurs due to privilege violation or due to an attempt to access unimplemented portion of array or register space then the type of the error generated depends on whether the access generating the error was initiated by the RCPU core or by a non-RCPU bus master. If the error causing access was initiated by the RCPU core, a data storage interrupt (DSI) is generated. If the access was initiated by a non-RCPU bus master, a machine check exception is generated. Also, a write access that generates an error does not corrupt the data in an array or a register. Similarly, a read access that generates an error does not drive the data on the L-bus from the array or the register, instead it drives 0's. Also, aborted accesses maintains data integrity —

aborted writes do not corrupt data in register/array, aborted reads do not drive the requested data on L-bus.

21.3.3 Two-Cycle Mode

In this mode, the CALRAM module takes two cycles to complete an access and the consumes less power than in one-cycle mode. It follows the normal one-cycle mode operation except that the accesses are completed one cycle later. This mode is selected by setting the 2CY bit in the CRAMMCR register.

21.3.4 Standby Operation / Keep-Alive Power

The registers and control logic for the CALRAM module is powered by VDD. The memory array(s) is also supplied by VDD during normal operation but when the VDD is off portions of CALRAM array is backed up by switched sources (VDDSRAM1/ VDDSRAM2) which are also known as standby power.

In the MPC565 / MPC566, when the VDD is off, the CALRAM arrays from 0x3F 8000 to 0x3F FFFF (32 Kbytes of CALRAM module A) and from 0x3F 7000 to 0x3F 7FFF (all four Kbytes of CALRAM module B) are powered by VDDSRAM1 and VDDSRAM2 respectively as shown in **Figure 21-3**.

21.3.5 Stop Operation

The low power stop mode for this module is entered by setting the disable bit (DIS) in the CRAMMCR register. Reads from and writes to the array during this mode will generate an error.

When the disable bit (DIS) is cleared, the module returns to normal function.

21.3.6 Overlay Mode Operation

For a microcontroller used as a controller for an engine (or other electromechanical device), various parameters stored in the flash memory may need to be changed in order to properly tune (calibrate) the engine. Since flash memory may not be readily programmed during normal operation of an embedded controller, this problem is solved by overlaying portions of CALRAM array onto the flash memory residing in U-bus. By allowing CALRAM module to overlay portions of flash memory, parameters normally stored in the flash may be tweaked and changed by development tool during normal operation and prior to programming a final version of the flash memory with more precise parameters.

The overlay is to read only data and does not affect instruction fetches from the flash. The data for any L-bus address which falls in the overlay region of the U-bus flash will be driven by the CALRAM on the L-bus. The CALRAM also indicates to the L2U to block the data from the flash to be driven on the L-bus. As far as the RCPU core is concerned, the timing of data coming from the CALRAM appears to be the same as that from the flash.

21.3.6.1 Overlay Mode Configuration

Each CALRAM module contains eight overlay regions, each of which is 512 bytes long as shown in **Figure 21-4**. All overlay regions of a module are contiguous **Figure 21-4** and each starts at the least significant address of the region and can increment all the way up to 512 bytes as shown in **Figure 21-5**. As described in section **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)**, CRAM_RBAx registers allow the programming of the base addresses RBA[11:29] of the U-bus flash regions and the RGN_SIZE[0:4] to be overlaid. Note that each region can also be individually disabled by writing 0000 to RGN_SIZE[0:3]. If the programmed base address is not naturally aligned with respect to the RGN_SIZE field, the least significant bits of the base address fields can be considered 0's in order to make the starting address naturally aligned. In an RBA register, RGN_SIZE[0:3] ={0101} select the size to be 128 bytes, and even if CRAM_RBAx [25:29] are not all 0's, they will be considered as 0's so that the address becomes 128-byte naturally aligned.

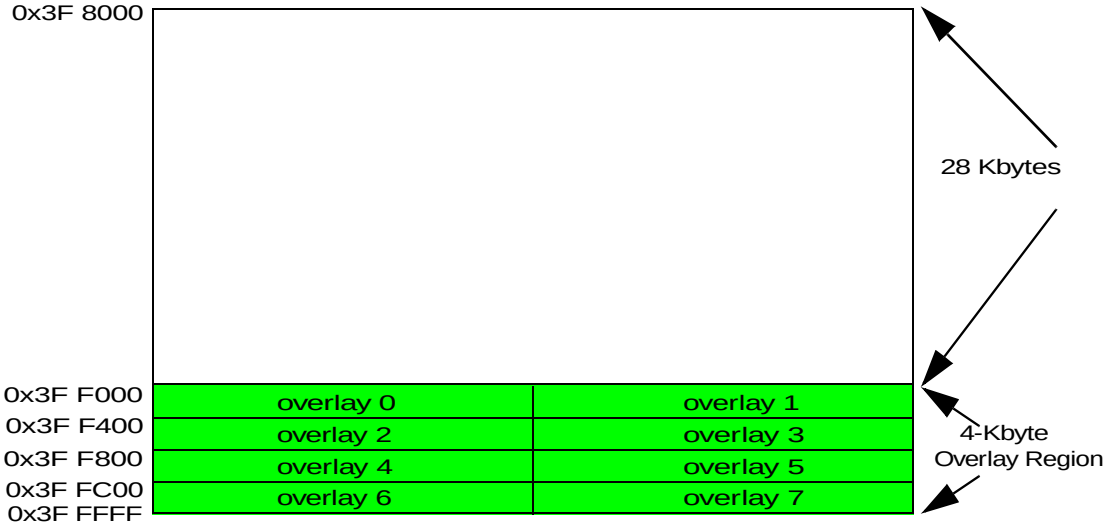


Figure 21-4 CALRAM_A array

When programming the CRAM_RBAx registers, the CALRAM can be put in overlay mode by setting the OVL bit in the CALRAM overlay configuration register (CRAMOVL) as described in section **21.4.3 CALRAM Overlay Configuration Register (CRAMOVLCR)**. For example, **Figure 21-5** shows that overlay regions 0, 4, and 5 have their entire region of 512 bytes mapped to regions in the flash as specified by CRAM_RBA0, CRAM_RBA4, and CRAM_RBA5. Overlay region 1 is partially mapped to a region in flash as specified by the CRAM_RBA1. For example, if the region size of 256 bytes is selected for overlay region 1, then the enabled portion of the overlay region 1 occupies array from 0x3F F200 to 0x3F F2FF. The rest of overlay region 1 from 0x3F F300 to 0x3F F3FF is available for normal (non-overlay) array access. Overlay regions 2, 3, 6, and 7 are disabled for overlay and hence can be used, in their entirety, for normal (non-overlay) array accesses.

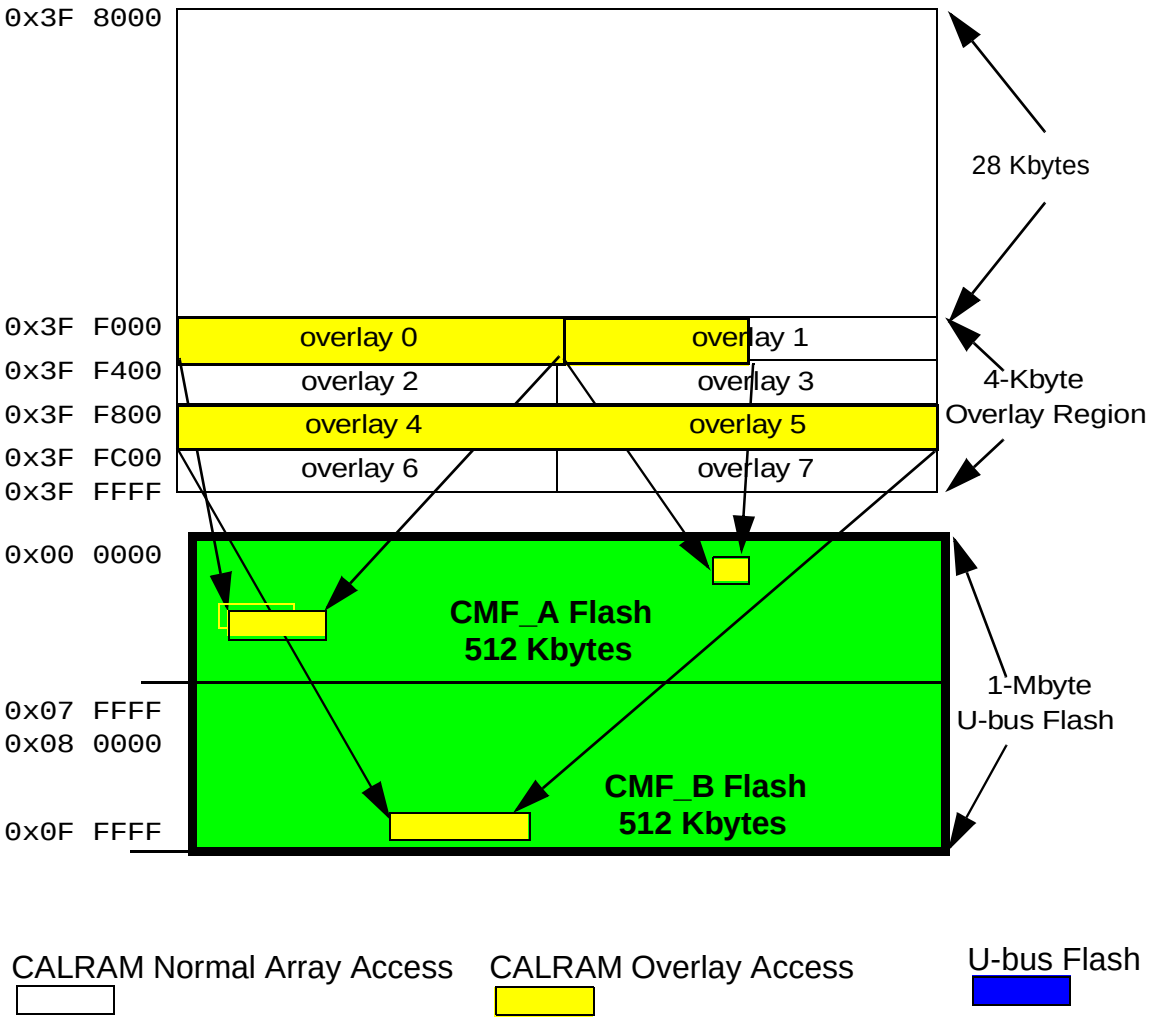


Figure 21-5 CALRAM_A Module Overlay Map of Flash (CLPS = 0)

Figure 21-6, illustrates the address spaces occupied by the two CALRAM modules available in MPC565 / MPC566.

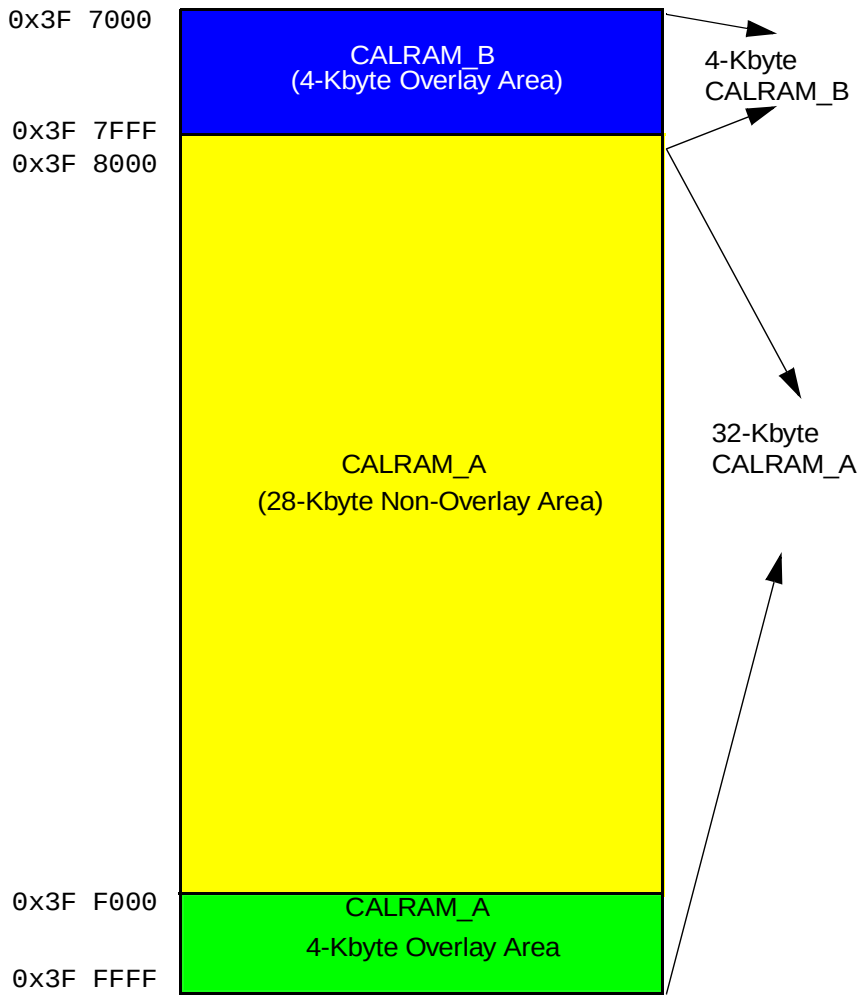


Figure 21-6 CALRAM_B and CALRAM_A Address Map (When CLPS = 0)

If the CLPS bit in OVLCR register is set, then each of the eight region size is forced to be 4 bytes long as shown in **Figure 21-7**,regardless of the value programmed in the RGN_SIZE field. These 32 bytes occupy contiguous address space in CALRAM_A, for example, from 0x3F FFE0 to 0x3F FFFF. The remainder (4 Kbytes – 32 bytes) is not only available for normal array access but also contiguous with a 28-Kbyte non-overlay array.

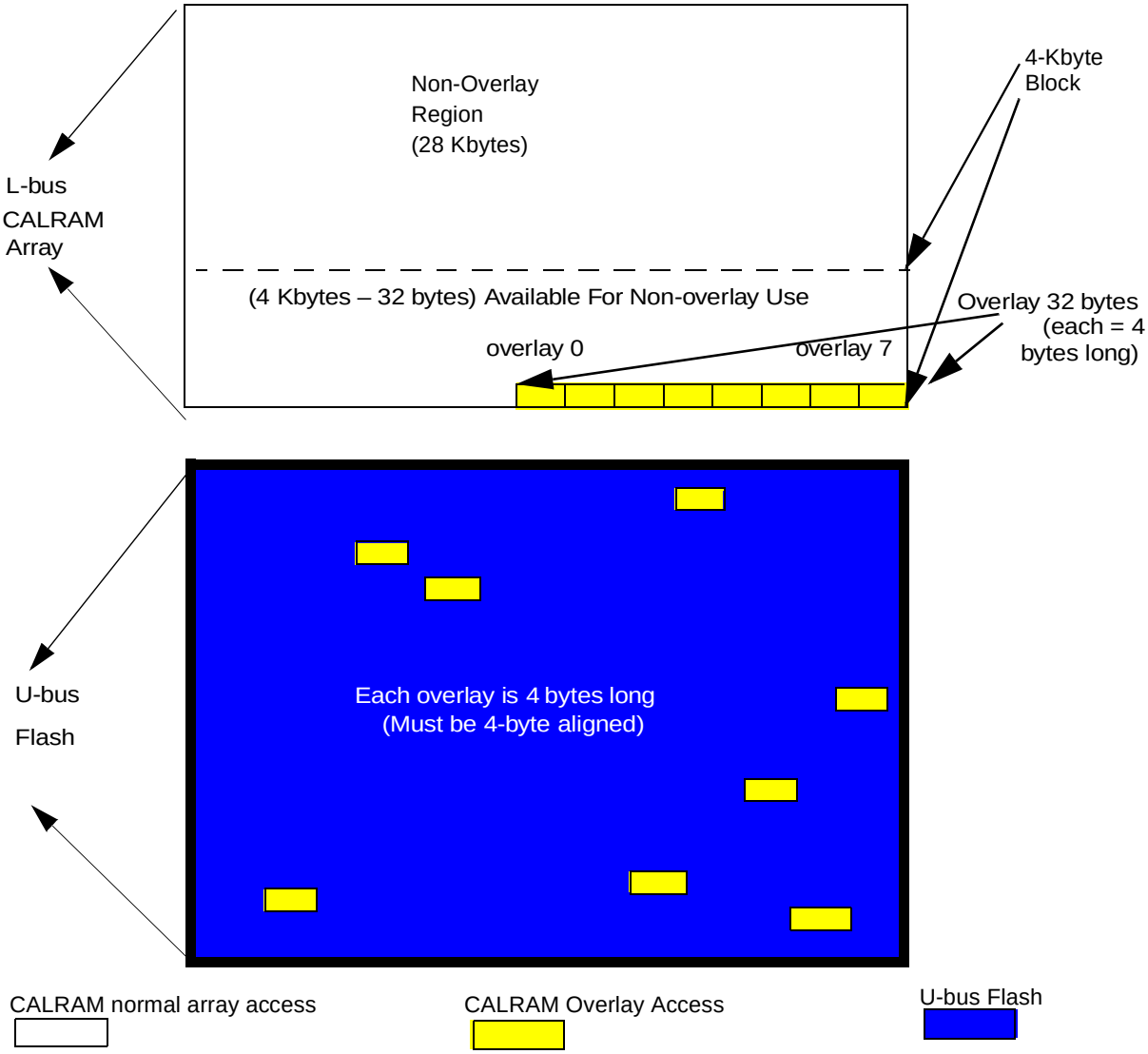


Figure 21-7 CALRAM Module Overlay Map of FLASH (CLPS = 1)

Figure 21-8 shows the overlay regions when the CLPS bit is set for both CALRAM_A and CALRAM_B in MPC565 / MPC566.

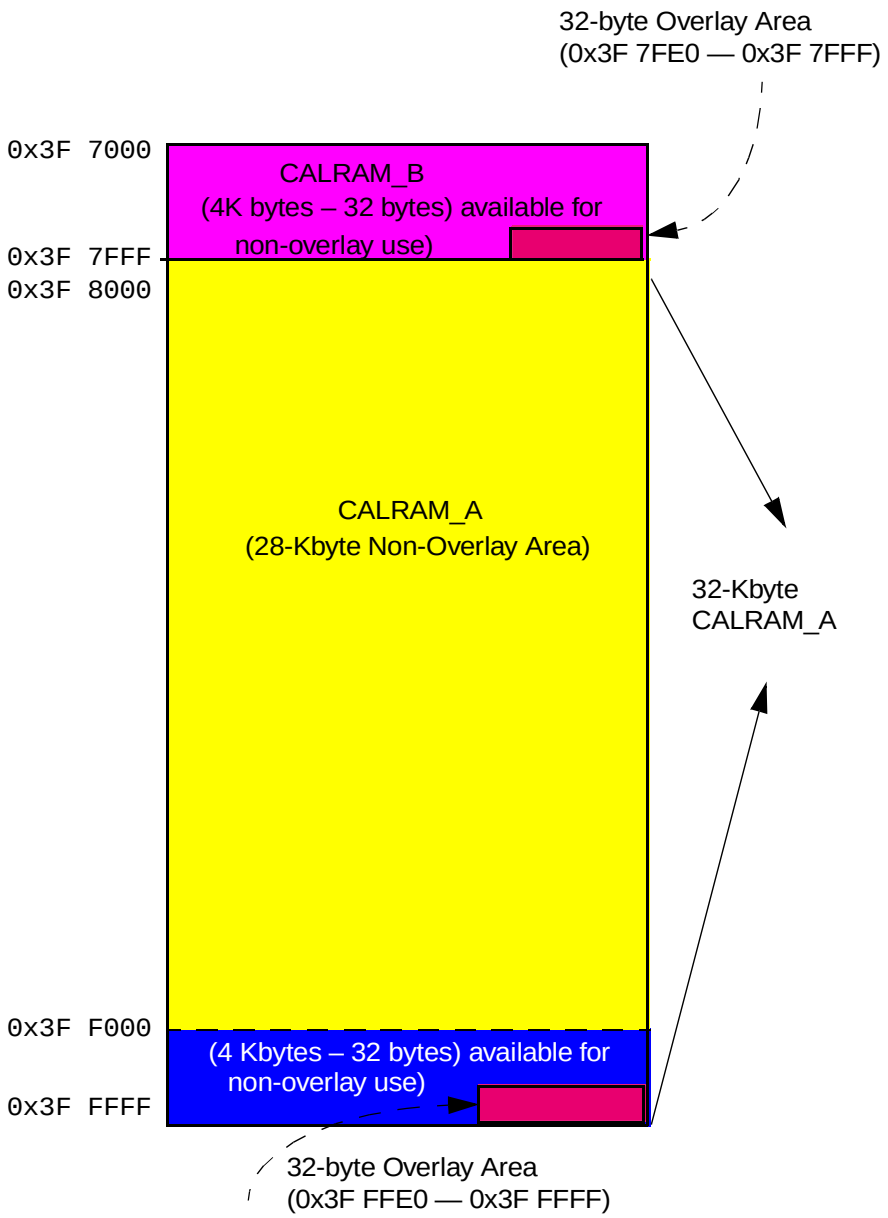


Figure 21-8 CALRAM_B and CALRAM_A Address Map (When CLPS = 1 for Both CALRAM_A and CALRAM_B)

NOTE

The values programmed in the RBAx registers are unaffected by reset. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for details. On reset, the it is not necessary to have to reprogram RABx registers. In such cases, the calibration mode can be re-entered simply by setting the OVL bit.

21.3.6.2 Priority of Overlay Regions

When the address matches to more than one enabled portion of the overlay region, the effective region is the region with the highest priority. Priority is determined by the region number; the highest priority assigned to the lowest region number. In an MCU with multiple CALRAM modules, the highest priority is assigned to the one in lowest address space (highest address).

EXAMPLE

In MPC565 / MPC566, region 0 of CALRAM_A has the higher priority and region 7 of CALRAM_B has the lowest priority as shown in **Table 21-1**. Also, region 7 of CALRAM_A has higher priority than region 0 of CALRAM_B.

The benefit from this priority feature is that by storing the parameters in eight overlay regions, it overlays all eight regions onto the same 512-byte flash region, and enables the overlay feature. Upon observing system performance with a set of parameters, the next set of parameters can be selected by simply disabling the highest priority region. This “observing and disabling the highest priority region” loop can continue until all regions are disabled. This allows moving from one set of parameters to another with minimal amount of reprograming efforts. If there are two CALRAM modules in a chip like in MPC565 / MPC566, then 16 sets of parameters can be run using this technique.

Table 21-1 Priorities of Overlay Regions

Module/Region Number	Priority
CALRAM_A/region 0	highest
CALRAM_A/region 1	.
CALRAM_A/region 2	.
CALRAM_A/region 3	.
CALRAM_A/region 4	.
CALRAM_A/region 5	.
CALRAM_A/region 6	.
CALRAM_A/region 7	.
CALRAM_B/region 0	.
CALRAM_B/region 1	.
CALRAM_B/region 2	.
CALRAM_B/region 3	.
CALRAM_B/region 4	.
CALRAM_B/region 5	.
CALRAM_B/region 6	.
CALRAM_B/region 7	lowest

21.3.6.3 Normal (Non-Overlay) Access to Overlay Regions

If overlay is enabled then any normal L-bus array access which falls within any of the eight enabled overlay regions, a machine check exception is generated if the option bit DERR is set in CRAMOVLCR register; otherwise, the access terminates normally

without asserting data error. The L-bus write accesses cause the data to be written regardless of whether DERR bit is set or not.

EXAMPLE

If overlay region 1 is programmed such that it is enabled and its region size is 256 bytes, then any L-bus access to address in the range of 0x3F F200 – 0x3F F2FF generates machine check exception if the DERR bit is set in CRAMOVLCR register. The other portion of region 1 from 0x3F F300 to 0x3F F3FF can be used as normal (non-overlay) array.

21.3.6.4 Calibration Write Cycle Flow

Write accesses to the overlaid U-bus flash regions are ignored completely by the CALRAM module.

21.4 Register Definitions

The following section describes the CALRAM programmer's model. The CALRAM has one register for configuring the CALRAM array and one register dedicated to factory test. In addition, there are eight 32-bit region base address registers for calibration purposes and a 32-bit overlay configuration register. The region base address registers hold the base address for the flash region and region size that need to be overlaid by the CALRAM. The overlay configuration register provides three bits (OVL, DERR, and CLSP) that are needed for overlay configuration. The CALRAM ownership trace register (CRAM_OTR) is provided to support a separate module called a READI module. Access to all CALRAM registers requires the bus master to be in supervisor data mode. On a privilege violation, the register is not accessed and the access generates an error.

Table 21-2 shows the register address map for MPC565 / MPC566.

Table 21-2 CALRAM A and B Control Registers

Address	Register
CALRAM_A	
0x38 0000	CRAMMCR_A
0x38 0004	for factory test
0x38 0008	CRAM_RBA0_A
0x38 000C	CRAM_RBA1_A
0x38 0010	CRAM_RBA2_A
0x38 0014	CRAM_RBA3_A
0x38 0018	CRAM_RBA4_A
0x38 001C	CRAM_RBA5_A
0x38 0020	CRAM_RBA6_A
0x38 0024	CRAM_RBA7_A
0x38 0028	CRAMOVLCR_A
0x38 002C	CRAMOTR_A
0x38 0030	Reserved
0x38 0034	Reserved
0x38 0038	Reserved
0x38 003C	Reserved
CALRAM_B	
0x38 0040	CRAMMCR_B
0x38 0044	for factory test
0x38 0048	CRAM_RBA0_B
0x38 004C	CRAM_RBA1_B
0x38 0050	CRAM_RBA2_B
0x38 0054	CRAM_RBA3_B
0x38 0058	CRAM_RBA4_B
0x38 005C	CRAM_RBA5_B
0x38 0060	CRAM_RBA6_B
0x38 0064	CRAM_RBA7_B
0x38 0068	CRAMOVLCR_B
0x38 006C	CRAMOTR_B
0x38 0070	Reserved
0x38 0074	Reserved
0x38 0078	Reserved
0x38 007C	Reserved

Any unimplemented bits in CALRAM registers return 0's on a read and writes to these bits are ignored.

21.4.1 CALRAM Module Configuration Register (CRAMMCR)

The module configuration register (CRAMMCR) contains bits that allow the CALRAM to be configured for normal RAM accesses.

CRAMMCR — CALRAM Module Configuration Register

0x38 8040
0x38 8000

LSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	MSB 15
LCK	DIS	2CY	Reserved												
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB 16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	MSB 31
Reserved				R0	D0	S0	R1	D1	S1	R2	D2	S2	R3	D3	S3
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

A brief description of each bit is provided in **Table 21-3**:

Table 21-3 CRAMMCR Bit Descriptions

Bit(s)	Name	Description
0	LCK	Write protection — This bit is designed to lock out writes to the CRAMMCR. While LCK = 0 the register can be written repeatedly without restriction. If LCK = 1, the register does not accept writes (i.e., the value of the register remains unchanged, but the cycle terminates normally.) In normal mode, this bit can only be set once and can only be cleared by reset 0 = writes to the CRAMMCR are unrestricted 1 = writes to the CRAMMCR are ignored In freeze mode, only the LCK bit may be written to zero if it was previously set.
1	DIS	Array disable — When set, this bit disables the CALRAM array. In this mode, all reads and writes to the CALRAM array are ignored and a bus error is generated. The CALRAM responds to register access while DIS = 1. This is a low power mode for the module, since all internal functions will be disabled. The module can be re-enabled by writing the DIS bit back to a zeroReset will also re-enable the module. 0 = CALRAM module array access is enabled 1 = CALRAM module array access is disabled
2	2CY	Two cycle mode — When set, this bit puts the CALRAM into a two cycle access mode operation for CALRAM register accesses as well as array accesses. This mode provides power savings by using the first cycle to decode any L-bus access for an address match to where the array resides. 0 = CALRAM module in one-cycle operation 1 = CALRAM module in two-cycle operation
3:19	—	Reserved

Table 21-3 CRAMMCR Bit Descriptions (Continued)

Bit(s)	Name	Description
20	R0	Read-only/read-write privilege — If the data relocate (DR) bit is set in Machine Status Register (MSR in RCPU) and R0 is also set, then write accesses are terminated with an error. If DR bit is 0, both reads and writes to the array block is allowed regardless of the value programmed in R0. This bit controls the highest 8-Kbyte block (lowest address) of CALRAM in the associated array. Likewise, R1, R2, and R3 control three other 8-Kbyte blocks in the same manner. See **Table 21-4** for control bit address ranges. R0 = 0 and DR = 0 readable and writable (array 8-Kbyte block) R0 = 0 and DR = 1 readable and writable (array 8-Kbyte block) R0 = 0 and DR = 0 readable and writable (array 8-Kbyte block) R0 = 1 and DR = 1 read only (array 8-Kbyte block)
21	D0	Data-only/data-instruction privilege (Data type assignment) — If the data relocate (DR) bit is set in Machine Status Register (MSR) and D0 is also set, then any access attempting to fetch an instruction from the array block generates an error. If DR bit is 0, both data read and instruction fetch from the array block is allowed, regardless of the value programmed in D0. This bit controls the highest 8-Kbyte block (lowest address) of CALRAM in the associated array. Likewise, D1, D2, and D3 control three other 8-Kbyte blocks in the same manner. See **Table 21-4** for control bit address ranges. D0 = 0 and DR = 0 data and/or Instruction (array 8-Kbyte block) D0 = 0 and DR = 1 data and/or Instruction (array 8-Kbyte block) D0 = 0 and DR = 0 data and/or Instruction (array 8-Kbyte block) D0 = 1 and DR = 1 data only (array 8-Kbyte block)
22	S0	Supervisor-only/supervisor-user privilege (Space assignment) — If the data relocate (DR) bit is set in Machine Status Register (MSR) and S0 is also set, then any access to the array block by a user program generates an error. If DR bit is 0, both user and supervisor program can access the array block, regardless of the value programmed in S0. The CALRAM array may be placed in supervisor or unrestricted space. This bit controls the highest 8-Kbyte block (lowest address) of CALRAM in the associated array. Likewise, S1, S2, and S3 control other three blocks in the same manner. See **Table 21-4** for control bit address ranges. S0 = 0 and DR = 0 both user and supervisor access allowed (array 8-Kbyte block) S0 = 0 and DR = 1 both user and supervisor access allowed (array 8-Kbyte block) S0 = 0 and DR = 0 both user and supervisor access allowed (array 8-Kbyte block) S0 = 1 and DR = 1 only supervisor access allowed (array 8-Kbyte block)
23	R1	Same as R0 except for address ranges shown on **Table 21-4**.
24	D1	Same as D0 except for address ranges shown on **Table 21-4**.
25	S1	Same as S0 except for address ranges shown on **Table 21-4**.
26	R2	Same as R0 except for address ranges shown on **Table 21-4**.
27	D2	Same as D0 except for address ranges shown on **Table 21-4**.
28	S2	Same as S0 except for address ranges shown on **Table 21-4**.
29	R3	Same as R0 except for address ranges shown on **Table 21-4**.
30	D3	Same as D0 except for address ranges shown on **Table 21-4**.
31	S3	Same as S0 except for address ranges shown on **Table 21-4**.

Table 21-4 CRAMMCR Privilege Bit Assignment for 8-Kbyte Array Blocks

Bit Selection	Address Block (Relative)
R0, D0, and S0	0xXXXX 0000 – 0xXXXX 1FFF
R1, D1, and S1	0xXXXX 2000 – 0xXXXX 3FFF
R2, D2, and S2	0xXXXX 4000 – 0xXXXX 5FFF
R3, D3, and S3	0xXXXX 6000 – 0xXXXX 7FFF

Since only a 4-Kbyte array is implemented in CALRAM_B module in MPC565 / MPC566, note that R3, D3, and S3 provide the privilege bits for the array from 0x3F 7000 to 0x3F 7FFF.

21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)

The region base address register defines the base address of a region on the U-bus flash memory space that will be overlaid by a portion of the CALRAM memory space and the region size. Since eight such regions in the flash can be overlaid by the CALRAM, eight such registers (x = 0, 1, 2, ., 7) are provided.

The CRAM_RBAx[11:29] provides the base address (starting address) of the of the U-bus flash region to be overlaid and the CRAM_RBAx[0:3] provides size corresponding to the region. See **Table 21-6** for details. The RGN_SIZE[0] is reserved and should never be programmed to a one. Since MPC565 / MPC566 has only one Mbyte of flash, CRAM_RBAx[11] should never be programmed to a one. Also, note that if CLPS bit in CRAMOVLCR is set, each of the eight sizes are forced to be four bytes, regardless of the value programmed in RGN_SIZE[0:3] field. See **21.4.3 CALRAM Overlay Configuration Register (CRAMOVLCR)** for details.

The implemented bits of CRAM_RBAx bits are, indeed, unaffected by reset (hard reset). The diagram below shows one such registers, CRAM_RBA0 which provides the base address of overlay region 0.

CRAM_RBA0-7 — CALRAM Region Base Address Register **0x38 0048 — 0x38 0064**
0x38 0008 — 0x38 0024

LSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	MSB 15
RGN_SIZE				RESERVED							RBA				
HARD RESET:															
X[1]	X	X	X	0	0	0	0	0	0	0	X	X	X	X	X

LSB 16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	MSB 31
RBA														RESERVED	
HARD RESET:															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0

NOTES:
1. Reset (hard reset) does not affect the values of these bits.

Table 21-5 CRAMMCR Bit Descriptions

Bit(s)	Name	Description
0:3	RGN_ SIZE	These bits define the size of the overlay region. See **Table 21-5** for sizes.
4:10	—	Reserved
11:29	RBA	The Region Base Address defines the starting address of the memory to be overlayed.
30:31	—	Reserved

Table 21-6 RGN_SIZE Encoding

RGN_SIZE	Number of Overlay Bytes
0000	Overlay block disabled
0001	Overlay block is 4 bytes
0010	Overlay block is 16 bytes
0011	Overlay block is 32 bytes
0100	Overlay block is 64 bytes
0101	Overlay block is 128 bytes
0110	Overlay block is 256 bytes
0111	Overlay block is 512 bytes
1000	Reserved
1001	Reserved
1010	Reserved
1011	Reserved

Table 21-6 RGN_SIZE Encoding

RGN_SIZE	Number of Overlay Bytes
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

NOTE

The overlay size of eight bytes cannot be selected.

21.4.3 CALRAM Overlay Configuration Register (CRAMOVLCR)

CRAMOVLCR — CALRAM Overlay Configuration Register **0x38 0068**
0x38 0028

LSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	MSB 15
OVL	DERR	CLPS	Reserved												
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB 16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	MSB 31
Reserved															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21-7 CRAMOVLCR Bit Descriptions

Bit(s)	Name	Description
0	OVL	Overlay enable — When set, the CALRAM overlay mode operation is enabled. In this mode CALRAM allows eight programmable sections (four to 512 bytes) of the on-chip U-bus flash memory module to be overlaid by sections of the CALRAM. 0 = CALRAM module overlay is disabled 1 = CALRAM module overlay is enabled
1	DERR	Data error 0 = CALRAM module will not generate machine check exception due to normal L-bus array access to the enabled portion overlay region even if overlay is enabled 1 = CALRAM module will generate machine check exception due to normal L-bus array access to the enabled portion of overlay region even if overlay is enabled
2	CLPS	Collapse the total overlay region from 4 Kbytes to 32 bytes; that is, the size is forced to be four bytes for each for the eight regions regardless of the values programmed in CRAM_RBAx[0:3]; these bits are also referred to as RGN_SIZE[0:3]. 0 = Overlay region of four Kbytes; region size as specified by CRAM_RBAx[0:3]. 1 = Overlay region of 32 bytes; each region size is four bytes long regardless of the valuesin CRAM_RBAx[0:3].
3:31	—	Reserved

21.4.4 CALRAM Ownership Trace Register (CRAMOTR)

This register is provided to support a separate module called READI. Refer to **SECTION 23 READI MODULE**.

The reads from this register will return 0's.

NOTE

CRAMOTR_A is also defined as READI_OTR. See **23.2.1.1 User Mapped Register**. CRAMOTR_B is not used on the MPC565 / MPC566.

CRAMOTR — CALRAM Ownership Trace Register **READI_OTR 0x38 006C**
CRAMOTR_B 0x38 002C

LSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	MSB 15
Ownership Trace Register															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LSB 16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	MSB 31
Ownership Trace Register															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECTION 22
DEVELOPMENT SUPPORT

22.1 Overview

The visibility and controllability requirements of emulators and bus analyzers are in opposition to the trend of modern microcomputers and microprocessors where many bus cycles are directed to internal resources and are not visible externally.

In order to enhance the development tool visibility and controllability, some of the development support functions are implemented in silicon. These functions include program flow tracking, internal watchpoint, breakpoint generation, and emulation while in debug mode.

This section covers program flow tracking support, breakpoint/watchpoint support, development system interface support (debug mode) and software monitor debugger support. These features allow efficiency in debugging systems based on the MPC565 / MPC566.

22.2 Program Flow Tracking

The mechanism described below allows tracking of program instruction flow with almost no performance degradation. The information provided may be compressed and captured externally and then parsed by a post-processing program using the microarchitecture defined below.

The program instructions flow is visible on the external bus when the MPC565 / MPC566 is programmed to operate in serial mode and show all fetch cycles on the external bus. This mode is selected by programming the ISCT_SER (instruction fetch show cycle control) field in the I-bus support control register (ICTRL), as shown in **Table 22-21**. In this mode, the processor is fetch serialized, and all internal fetch cycles appear on the external bus. Processor performance is, therefore, much lower than when working in regular mode.

These features, together with the fact that most fetch cycles are performed internally (e.g., from the I-cache), increase performance but make it very difficult to provide the real program trace.

In order to reconstruct a program trace, the program code and the following additional information from the MCU are needed:

- A description of the last fetched instruction (stall, sequential, branch not taken, branch direct taken, branch indirect taken, exception taken)
- The addresses of the targets of all indirect flow change. Indirect flow changes include all branches using the link and count registers as the target address, all exceptions, and **rfi**, **mtmsr and mtspr** (to some registers) because they may cause a context switch.
- The number of instructions canceled each clock

Instructions are fetched sequentially until branches (direct or indirect) or exceptions appear in the program flow or some stall in execution causes the machine not to fetch the next address. Instructions may be architecturally executed, or they may be canceled in some stage of the machine pipeline.

The following sections define how this information is generated and how it should be used to reconstruct the program trace. The issue of data compression that could reduce the amount of memory needed by the debug system is also mentioned.

22.2.1 Program Trace Cycle

To allow visibility of the events happening in the machine a few dedicated pins are used and a special bus cycle attribute, program trace cycle, is defined.

The program trace cycle attribute is attached to all fetch cycles resulting from indirect flow changes. When program trace recording is needed, make sure these cycles are visible on the external bus.

The **VSYNC** indication, when asserted, forces all fetch cycles marked with the program trace cycle attribute to be visible on the external bus even if their data is found in one of the internal devices. To enable the external hardware to properly synchronize with the internal activity of the CPU, the assertion and negation of **VSYNC** forces the machine to synchronize. The first fetch after this synchronization is marked as a program trace cycle and is visible on the external bus. For more information on the activity of the external hardware during program trace refer to **22.2.4 The External Hardware**.

In order to keep the pin count of the chip as low as possible, **VSYNC** is not implemented as one of the chip's external pins. It is asserted and negated using the serial interface implemented in the development port. For more information on this interface refer to **22.5 Development Port**

Forcing the CPU to show all fetch cycles marked with the program trace cycle attribute can be done either by asserting the **VSYNC** pin (as mentioned above) or by programming the fetch show cycle bits in the instruction support control register, ICTRL. For more information refer to **22.2.5 Instruction Fetch Show Cycle Control**

When the **VSYNC** indication is asserted, all fetch cycles marked with the program trace cycle attribute are made visible on the external bus. These cycles can generate regular bus cycles (address phase and data phase) when the instructions reside only in one of the external devices. Or, they can generate address-only cycles when the instructions reside in one of the internal devices (internal memory, etc.).

When **VSYNC** is asserted, some performance degradation is expected due to the additional external bus cycles. However, since this performance degradation is expected to be very small, it is possible to program the machine to *show all indirect flow changes*. In this way, the machine will always perform the additional external bus cycles and maintain exactly the same behavior both when **VSYNC** is asserted and when it is negated. For more information refer to **22.7.6 I-Bus Support Control Register**.

The status pins are divided into two groups and one special case listed below:

22.2.1.1 Instruction Queue Status Pins — VF [0:2]

Instruction queue status pins denote the type of the last fetched instruction or how many instructions were flushed from the instruction queue. These status pins are used for both functions because queue flushes only happen in clocks that there is no fetch type information to be reported.

Possible instruction types are defined in **Table 22-1**.

Table 22-1 VF Pins Instruction Encodings

VF[0:2]	Instruction Type	VF Next Clock Will Hold
000	None	More instruction type information
001	Sequential	More instruction type information
010	Branch (direct or indirect) **not** taken	More instruction type information
011	VSYNC was asserted/negated and therefore the next instruction will be marked with the indirect change-of-flow attribute	More instruction type information
100	Exception taken — the target will be marked with the indirect change-of-flow attribute	Queue flush information[1]
101	Branch indirect taken, **rfi**, **mtmsr**, **isync** and in some cases **mtspr** to CMPA-F, ICTRL, ECR, or DER — the target will be marked with the indirect change-of-flow attribute[2]	Queue flush information[1]
110	Branch direct taken	Queue flush information[1]
111	Branch (direct or indirect) **not** taken	Queue flush information[1]

NOTES:
1. Unless next clock VF=111. See below.
2. The sequential instructions listed here affect the machine in a manner similar to indirect branch instructions. Refer to **22.2.3 Sequential Instructions Marked as Indirect Branch**.

Table 22-2 shows VF[0:2] encodings for instruction queue flush information.

Table 22-2 VF Pins Queue Flush Encodings

VF[0:2]	Queue Flush Information
000	0 instructions flushed from instruction queue
001	1 instruction flushed from instruction queue
010	2 instructions flushed from instruction queue
011	3 instructions flushed from instruction queue
100	4 instructions flushed from instruction queue
101	5 instructions flushed from instruction queue
110	Reserved
111	Instruction type information[1]

NOTES:
1. Refer to **Table 22-1**.

22.2.1.2 History Buffer Flushes Status Pins— VFLS [0..1]

The history buffer flushes status pins denote how many instructions are flushed from the history buffer this clock due to an exception. **Table 22-3** shows VFLS encodings.

Table 22-3 VFLS Pin Encodings

VFLS[0:1]	History Buffer Flush Information
00	0 instructions flushed from history queue
01	1 instruction flushed from history queue
10	2 instructions flushed from history queue
11	Used for debug mode indication (FREEZE). Program trace external hardware should ignore this setting.

22.2.1.3 Queue Flush Information Special Case

There is one special case when although queue flush information is expected on the VF pins, (according to the last value on the VF pins), regular instruction type information is reported. The only instruction type information that can appear in this case is VF = 111, branch (direct or indirect) NOT taken. Since the maximum queue flushes possible is five, it is easy to identify this special case.

22.2.2 Program Trace when in Debug Mode

When entering debug mode an *interrupt/exception taken* is reported on the VF pins, (VF = 100) and a cycle marked with the program trace cycle is made visible externally.

When the CPU is in debug mode, the VF pins equal ‘000’ and the VFLS pins equal ‘11’. For more information on debug mode refer to **22.4 Development System Interface**

If **VSYNC** is asserted/negated while the CPU is in debug mode, this information is reported as the first VF pins report when the CPU returns to regular mode. If **VSYNC** was not changed while in debug mode. the first VF pins report will be of an indirect branch taken (VF = 101), suitable for the **rfi** instruction that is being issued. In both cases the first instruction fetch after debug mode is marked with the program trace cycle attribute and therefore is visible externally.

22.2.3 Sequential Instructions Marked as Indirect Branch

There are cases when non-branch (sequential) instructions may effect the machine in a manner similar to indirect branch instructions. These instructions include **rfi**, **mtmsr**, **isync** and **mtspr** to CMPA-F, ICTRL, ECR and DER.

These instructions are marked by the CPU as indirect branch instructions (VF = 101) and the following instruction address is marked with the same program trace cycle attribute as if it were an indirect branch target. Therefore, when one of these special instructions is detected in the CPU, the address of the following instruction is visible externally. In this way the reconstructing software is able to evaluate correctly the effect of these instructions.

22.2.4 The External Hardware

When program trace is needed, the external hardware needs to sample the status pins (VF and VFLS) each clock cycle and the address of all cycles marked with the program trace cycle attribute.

Program trace can be used in various ways. Below are two examples of how program trace can be used:

- **Back trace —** Back trace is useful when a record of the program trace *before* some event occurred is needed. An example of such an event is some system failure.
  In case back trace is needed the external hardware should start sampling the status pins (VF and VFLS) and the address of all cycles marked with the program trace cycle attribute immediately when reset is negated. If *show cycles* is programmed out of reset to *show all,* all cycles marked with program trace cycle attribute are visible on the external bus. **VSYNC** should be asserted sometime after reset and negated when the programmed event occurs. If *no show* is programmed for *show cycles*, make sure **VSYNC** is asserted before the Instruction show cycles programming is changed from *show all*.
  Note that in case the timing of the programmed event is unknown it is possible to use cyclic buffers.
  After **VSYNC** is negated the trace buffer will contain the program flow trace of the program executed before the programmed event occurred.
- **Window trace —** Window trace is useful when a record of the program trace between two events is needed. In case window trace is needed the **VSYNC** pin should be asserted between these two events.
  After the **VSYNC** pin is negated the trace buffer will contain information describing the program trace of the program executed *between* the two events.

22.2.4.1 Synchronizing the Trace Window to the CPU Internal Events

The assertion/negation of **VSYNC** is done using the serial interface implemented in the development port. In order to synchronize the assertion/negation of **VSYNC** to an internal event of the CPU, it is possible to use the internal breakpoints together with debug mode. This method is available only when debug mode is enabled. For more information on debug mode refer to **22.4 Development System Interface**

The following is an example of steps that enable synchronization of the trace window to the CPU internal events:

1. Enter debug mode, either immediately out of reset or using the debug mode request
2. Program the hardware to break on the event that marks the start of the trace window using the control registers defined in **22.3 Watchpoints and Breakpoints Support**
3. Enable debug mode entry for the programmed breakpoint in the debug enable register (DER). See **22.7.12 Debug Enable Register (DER)**)
4. Return to the regular code run (see **22.4.1.6 Exiting Debug Mode**)
5. The hardware generates a breakpoint when the programmed event is detected

and the machine enters debug mode (see **22.4.1.2 Entering Debug Mode**)

6. Program the hardware to break on the event that marks the end of the trace window
7. Assert **VSYNC**
8. Return to the regular code run. The first report on the VF pins is a **VSYNC** (VF = 011).
9. The external hardware starts sampling the program trace information upon the report on the VF pins of **VSYNC**
10. The hardware generates a breakpoint when the programmed event is detected and the machine enters debug mode
11. Negate **VSYNC**
12. Return to the regular code run (issue an **rfi**). The first report on the VF pins is a **VSYNC** (VF = 011)
13. The external hardware stops sampling the program trace information upon the report on the VF pins of **VSYNC**

22.2.4.2 Detecting the Trace Window Start Address

When using *back trace*, latching the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle, should start immediately after the negation of reset. The start address is the first address in the program trace cycle buffer.

When using *window trace*, latching the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle, should start immediately after the first **VSYNC** is reported on the VF pins. The start address of the trace window should be calculated according to first two VF pins reports.

Assuming that VF1 and VF2 are the two first VF pins reports and T1 and T2 are the two addresses of the first two cycles marked with the program trace cycle attribute that were latched in the trace buffer, use the following table to calculate the trace window start address.

Table 22-4 Detecting the Trace Buffer Start Point

VF1	VF2	Starting point	Description
011 **VSYNC**	001 sequential	T1	**VSYNC** asserted followed by a sequential instruction. The start address is T1
011 **VSYNC**	110 branch direct taken	T1 - 4 + offset (T1 - 4)	**VSYNC** asserted followed by a taken direct branch. The start address is the target of the direct branch
011 **VSYNC**	101 branch indirect taken	T2	**VSYNC** asserted followed by a taken indirect branch. The start address is the target of the indirect branch

22.2.4.3 Detecting the Assertion/Negation of VSYNC

Since the VF pins are used for reporting both instruction type information and queue flush information, the external hardware must take special care when trying to detect the assertion/negation of **VSYNC**. When VF = 011 it is a **VSYNC** assertion/negation

report only if the previous VF pins value was one of the following values: 000, 001, or 010.

22.2.4.4 Detecting the Trace Window End Address

The information on the status pins that describes the last fetched instruction and the last queue/history buffer flushes, changes every clock. Cycles marked as program trace cycle are generated on the external bus only when possible (when the SIU wins the arbitration over the external bus). Therefore, there is some delay between the information reported on the status pins that a cycle marked as program trace cycle will be performed on the external bus and the actual time that this cycle can be detected on the external bus.

When **VSYNC** is negated (through the serial interface of the development port), the CPU delays the report of the of the assertion/negation of **VSYNC** on the VF pins (VF = 011) until all addresses marked with the program trace cycle attribute were visible externally. Therefore, the external hardware should stop sampling the value of the status pins (VF and VFLS), and the address of the cycles marked as program trace cycle immediately after the **VSYNC** report on the VF pins.

The last two instructions reported on the VF pins are not always valid. Therefore at the last stage of the reconstruction software, the last two instructions should be ignored.

22.2.4.5 Compress

In order to store all the information generated on the pins during program trace (five bits per clock + 30 bits per show cycle) a large memory buffer may be needed. However, since this information includes events that were canceled, compression can be very effective. External hardware can be added to eliminate all canceled instructions and report only on branches (taken and not taken), indirect flow change, and the number of sequential instructions after the last flow change.

22.2.5 Instruction Fetch Show Cycle Control

Instruction fetch show cycles are controlled by the bits in the ICTRL and the state of **VSYNC**. The following table defines the level of fetch show cycles generated by the CPU. For information on the fetch show cycles control bits refer to **Table 22-5**

Table 22-5 Fetch Show Cycles Control

VSYNC	ISCTL Instruction Fetch Show Cycle Control Bits	Show Cycles Generated
X	00	All fetch cycles
X	01	All change of flow (direct & indirect)
X	10	All indirect change of flow
0	11	No show cycles are performed
1	11	All indirect change of flow

NOTE

A cycle marked with the program trace cycle attribute is generated for any change in the **VSYNC** state (assertion or negation).

22.3 Watchpoints and Breakpoints Support

Watchpoints, when detected, are reported to the external world on dedicated pins but do not change the timing and the flow of the machine. Breakpoints, when detected, force the machine to branch to the appropriate exception handler. The CPU supports internal watchpoints, internal breakpoints, and external breakpoints.

Internal watchpoints are generated when a user programmable set of conditions are met. Internal breakpoints can be programmed to be generated either as an immediate result of the assertion of one of the internal watchpoints, or after an internal watchpoint is asserted for a user programmable times. Programming a certain internal watchpoint to generate an internal breakpoint can be done either in software, by setting the corresponding software trap enable bit, or on the fly using the serial interface implemented in the development port to set the corresponding development port trap enable bit.

External breakpoints can be generated by any of the peripherals of the system, including those found on the MPC565 / MPC566 or externally, and also by an external development system. Peripherals found on the external bus use the serial interface of the development port to assert the external breakpoint.

In the CPU, as in other RISC processors, saving/restoring machine state on the stack during exception handling, is done mostly in software. When the software is in the middle of saving/restoring machine state, the MSRRI bit is cleared. Exceptions that occur and that are handled by the CPU when the MSRRI bit is clear result in a non-restartable machine state. For more information refer to **3.15.4 Interrupts**

In general, breakpoints are recognized in the CPU is only when the MSRRI bit is set, which guarantees machine restartability after a breakpoint. In this working mode breakpoints are said to be *masked*. There are cases when it is desired to enable breakpoints even when the MSRRI bit is clear, with the possible risk of causing a non-restartable machine state. Therefore internal breakpoints have also a programmable *non-masked* mode, and an external development system can also choose to assert a *non-maskable* external breakpoint.

Watchpoints are not *masked* and therefore always reported on the external pins, regardless of the value of the MSRRI bit. The counters, although counting watchpoints, are part of the internal breakpoints logic and therefore are not decremented when the CPU is operating in the masked mode and the MSRRI bit is clear.

The following figure illustrates the watchpoints and breakpoints support of the CPU.

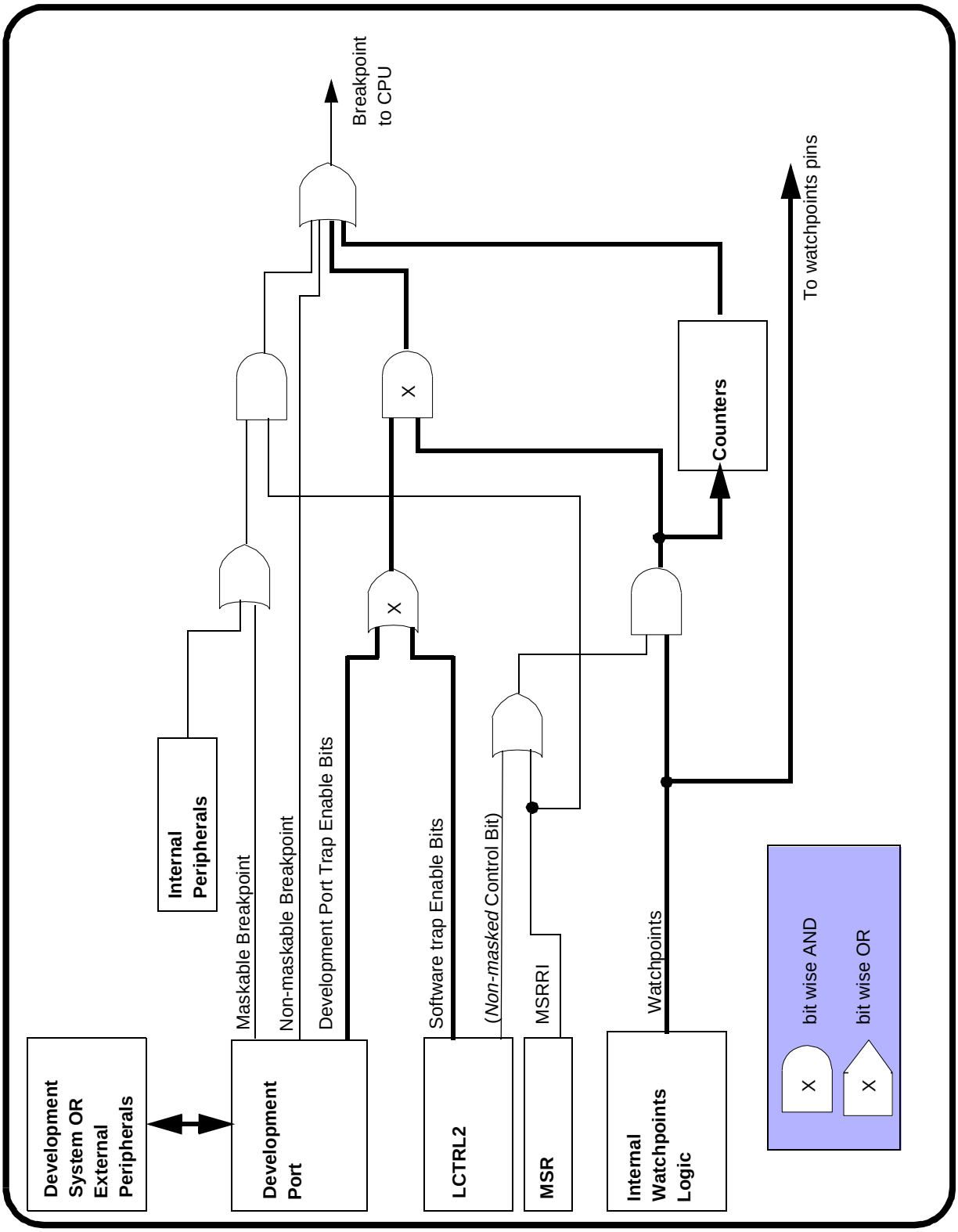


Figure 22-1 Watchpoints and Breakpoint Support in the CPU

22.3.1 Internal Watchpoints and Breakpoints

This section describes the internal breakpoints and watchpoints support of the CPU. For information on external breakpoints support refer to **22.4 Development System Interface**

Internal breakpoint and watchpoint support is based on eight comparators comparing information on instruction and load/store cycles, two counters, and two AND-OR logic structures. The comparators perform compare on the Instruction address (I-address), on the load/store address (L-address) and on the load/store data (L-data).

The comparators are able to detect the following conditions: equal, not equal, greater than, less than (greater than or equal and less than or equal are easily obtained from these four conditions, for more information refer to **22.3.1.6 Generating Six Compare Types**). Using the AND-OR logic structures “in range” and “out of range” detections (on address and on data) are supported. Using the counters, it is possible to program a breakpoint to be recognized after an event was detected a predefined number of times.

The L-data comparators can operate on fix point data of load or store. When operating on fix point data the L-data comparators are able to perform compare on bytes, half-words and words and can treat numbers either as signed or as unsigned values.

The comparators generate match events. The match events enter the instruction AND-OR logic where the instruction watchpoints and breakpoint are generated. The instruction watchpoints, when asserted, may generate the instruction breakpoint. Two of them may decrement one of the counters. If one of the instruction watchpoints expires in a counter that is counting, the instruction breakpoint is asserted.

The instruction watchpoints and the load/store match events (address and data) enter the load/store AND-OR logic where the load/store watchpoints and breakpoint are generated. The load/store watchpoints, when asserted, may generate the load/store breakpoint or they may decrement one of the counters. When a counter that is counting one of the load/store watchpoints expires, the load/store breakpoint is asserted.

The READI module provides watchpoint messaging using GEPDIS standard (version 1.0) defined public messages. The watchpoint status signals from the RCPU are snooped, and when watchpoints occur, a message is sent to the pin output formatter to be messaged out (the general message queue is bypassed to prevent watchpoint messages from being cancelled in the event of a queue overflow). The Watchpoint message has the second highest priority. Refer to **23.3.2.1 Message Priority** for further details on message priorities. The watchpoint message contains the watchpoint code which indicates all the unique watchpoints have occurred since the last watchpoint message. If duplicate watchpoints occur before the watchpoint message is sent out, a watchpoint overrun message is generated. The watchpoint source field will indicate which watchpoints occurred.

Watchpoints progress in the machine and are reported on retirement. Internal breakpoints progress in the machine until they reach the top of the history buffer when the machine branches to the breakpoint exception routine.

In order to enable the use of the breakpoint features without adding restrictions on the software, the address of the load/store cycle that generated the load/store breakpoint

is not stored in the DAR (data address register), like other load/store type exceptions. In case of a load/store breakpoint, the address of the load/store cycle that generated the breakpoint is stored in an implementation-dependent register called the BAR (breakpoint address register).

Key features of internal watchpoint and breakpoint support are:

- Four I-address comparators (each supports equal, not equal, greater than, less than)
- Two L-address comparators (each supports equal, not equal, greater than, less than) including least significant bits masking according to the size of the bus cycle for the byte and half-word working modes. Refer to **22.3.1.2 Byte and Half-Word Working Modes**
- Two L-data comparators (each supports equal, not equal, greater than, less than) including byte, half-word and word operating modes and four byte mask bits for each comparator. Can be used for fix point data. Match is detected only on the valid part of the data bus (according to the cycle's size and the two address least significant bits).
- *No internal breakpoint/watchpoint matching support for unaligned words and half-words*
- The L-data comparators can be programmed to treat fix point numbers as signed values or as unsigned values
- Combine comparator pairs to detect in and out of range conditions (including either signed or unsigned values on the L-data)
- A programmable AND-OR logic structure between the four instruction comparators results with five outputs, four instruction watchpoints and one instruction breakpoint
- A programmable AND-OR logic structure between the four instruction watchpoints and the four load/store comparators results with three outputs, two load/store watchpoints and one load/store breakpoint
- Five watchpoint pins, three for the instruction and two for the load/store
- Two dedicated 16-bit down counters. Each can be programmed to count either an instruction watchpoint or an load/store watchpoint. Only *architecturally executed* events are counted, (count up is performed in case of recovery).
- On the fly trap enable programming of the different internal breakpoints using the serial interface of the development port (refer to **22.5 Development Port**). Software control is also available.
- Watchpoints do not change the timing of the machine
- Internal breakpoints and watchpoints are detected on the instruction during instruction fetch
- Internal breakpoints and watchpoints are detected on the load/store during load/store bus cycles
- Both instruction and load/store breakpoints and watchpoints are handled and reported on retirement. Breakpoints and watchpoints on recovered instructions (as a result of exceptions, interrupts or miss prediction) are not reported and do not change the timing of the machine.
- Instructions with instruction breakpoints are not executed. The machine branches to the breakpoint exception routine BEFORE it executes the instruction.
- Instructions with load/store breakpoints are executed. The machine branches to

the breakpoint exception routine AFTER it executes the instruction. The address of the access is placed in the BAR (breakpoint address register).

- Load/store multiple and string instructions with load/store breakpoints first finish execution (all of it) and then the machine branches to the breakpoint exception routine.
- Load/store data compare is done on the load/store, AFTER swap in store accesses and BEFORE swap in load accesses (as the data appears on the bus).
- Internal breakpoints may operate either in *masked* mode or in *non-masked* mode.
- Both “go to x” and “continue” working modes are supported for the instruction breakpoints.

22.3.1.1 Restrictions

There are cases when the same watchpoint can be detected more than once during the execution of a single instruction, e.g. a load/store watchpoint is detected on more than one transfer when executing a load/store multiple/string or a load/store watchpoint is detected on more than one byte when working in byte mode. In all these cases only one watchpoint of the same type is reported for a single instruction. Similarly, only one watchpoint of the same type can be counted in the counters for a single instruction.

Since watchpoint events are reported upon the retirement of the instruction that caused the event, and more than one instruction can retire from the machine in one clock, consequent events may be reported in the same clock. Moreover the same event, if detected on more than one instruction (e.g., tight loops, range detection), in some cases will be reported only once. Note that the internal counters count correctly in these cases.

Do not put a breakpoint on an **mtspr ICTRL** instruction. When a breakpoint is set on an **mtspr ICTRL** Rx instruction and the value of bit 28 (IFM) is one, the result will be unpredictable. A breakpoint can be taken or not on the instruction and the value of the IFM bit can be either zero or one. Also, do not put a breakpoint on an mtspr ICTRL Rx instruction when Rx contains one in bit 28.

22.3.1.2 Byte and Half-Word Working Modes

The CPU watchpoints and breakpoints support enables detection of matches on bytes and half-words even when accessed using a load/store instruction of larger data widths, for example when loading a table of bytes using a series of load word instructions. In order to use this feature, program the byte mask for each of the L-data comparators and to write the needed match value to the correct half-word of the data comparator when working in half-word mode and to the correct bytes of the data comparator when working in byte mode.

Since bytes and half-words can be accessed using a larger data width instruction, it is impossible to predict the exact value of the L-address lines when the requested byte/half-word is accessed, (e.g., if the matched byte is byte two of the word and it is accessed using a load word instruction), the L-address value will be of the word (byte zero). Therefore, the CPU masks the two least-significant bits of the L-address com-

parators whenever a word access is performed and the least-significant bit whenever a half-word access is performed.

Address range is supported only when aligned according to the access size. (See examples)

22.3.1.3 Examples

- A fully supported scenario:
  Looking for:
  Data size: Byte
  Address: 0x00000003
  Data value: greater than 0x07 and less than 0x0c
  Programming options:
  One L-address comparator = 0x00000003 and program for equal
  One L-data comparator = 0x00000007 and program for greater than
  One L-data comparator = 0x0000000c and program for less than
  Both byte masks = 0xe
  Both L-data comparators program to byte mode
  Result: The event will be correctly detected regardless of the load/store instruction the compiler chooses for this access
- A fully supported scenario:
  Looking for:
  Data size: half-word
  Address: greater than 0x00000000 and less than 0x0000000c
  Data value: greater than 0x4e204e20 and less than 0x9c409c40
  Programming option:
  One L-address comparator = 0x00000000 and program for greater than
  One L-address comparator = 0x0000000c and program for less than
  One L-data comparator = 0x4e204e20 and program for greater than
  One L-data comparator = 0x9c409c40 and program for less than
  Both byte masks = 0x0
  Both L-data comparators program to half-word mode
  Result: The event will be correctly detected as long as the compiler does not use a load/store instruction with data size of byte.
- A **partially** supported scenario:
  Looking for:
  Data size: half-word
  Address: greater than or equal 0x00000002 and less than 0x0000000e
  Data value: greater than 0x4e204e20 and less than 0x9c409c40
  Programming option:
  One L-address comparator = 0x00000001 and program for greater than
  One L-address comparator = 0x0000000e and program for less than
  One L-data comparator = 0x4e204e20 and program for greater than
  One L-data comparator = 0x9c409c40 and program for less than
  Both byte masks = 0x0
  Both L-data comparators program to half-word mode or to word mode

Result: The event will be correctly detected if the compiler chooses a load/store instruction with data size of half-word. If the compiler chooses load/store instructions with data size greater than half-word (word, multiple), there might be some false detections.

These can be ignored only by the software that handles the breakpoints. The following figure illustrates this **partially** supported scenario.

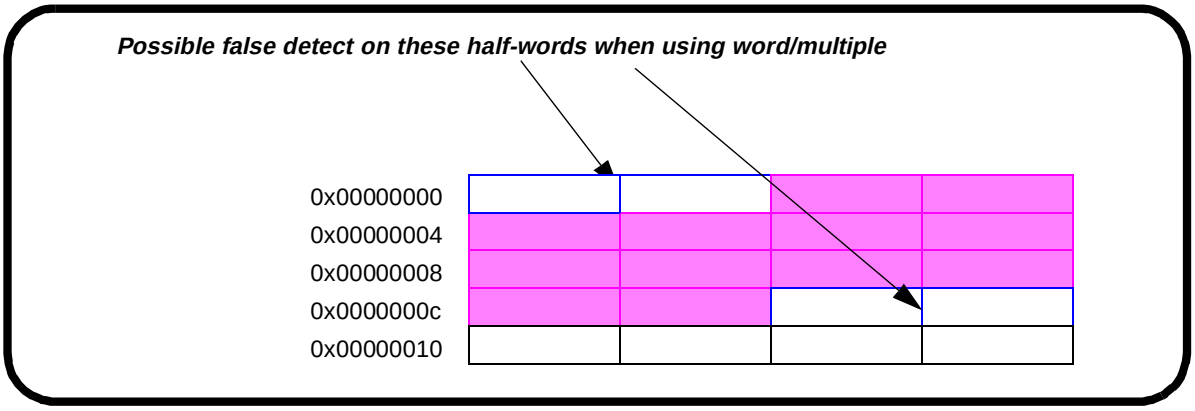


Figure 22-2 Partially Supported Watchpoint/Breakpoint Example

22.3.1.4 Context Dependent Filter

The CPU can be programmed to either recognize internal breakpoints only when the recoverable interrupt bit in the MSR is set (*masked* mode) or it can be programmed to always recognize internal breakpoints (*non-masked* mode).

When the CPU is programmed to recognize internal breakpoints only when MSRRI = 1, it is possible to debug all parts of the code except when the machine status save/restore registers (SRR0 and SRR1), DAR (data address register) and DSISR (data storage interrupt status register) are busy and, therefore, MSRRI = 0, (in the prologues and epilogues of interrupt/exception handlers).

When the CPU is programmed always to recognize internal breakpoints, it is possible to debug all parts of the code. However, if an internal breakpoint is recognized when MSRRI = 0 (SRR0 and SRR1 are busy), the machine enters into a non-restartable state. For more information refer to **3.15.4 Interrupts**

When working in the *masked* mode, all internal breakpoints detected when MSRRI = 0 are lost. Watchpoints detected in this case are not counted by the debug counters. Watchpoints detected are always reported on the external pins, regardless of the value of the MSRRI bit.

Out of reset, the CPU is in *masked* mode. Programming the CPU to be in *non-masked* mode is done by setting the BRKNOMSK bit in the LCTRL2 register. Refer to **22.7.8 L-Bus Support Control Register 2** The BRKNOMSK bit controls all internal breakpoints (I-breakpoints and L-breakpoints).

22.3.1.5 Ignore First Match

In order to facilitate the debugger utilities “continue” and “go from x”, the ignore first match option is supported for instruction breakpoints. When an instruction breakpoint is first enabled (as a result of the first write to the instruction support control register or as a result of the assertion of the MSRRI bit when operating in the *masked* mode), the first instruction will not cause an instruction breakpoint if the ignore first match (IFM) bit in the instruction support control register (ICTRL) is set (used for “continue”).

When the IFM bit is clear, every matched instruction can cause an instruction breakpoint (used for “go from x”). This bit is set by the software and cleared by the hardware after the first instruction breakpoint match is ignored. Load/store breakpoints and all counter generated breakpoints (instruction and load/store) are not affected by this mode.

22.3.1.6 Generating Six Compare Types

Using the four compare types mentioned above (equal, not equal, greater than, less than) it is possible to generate also two more compare types: greater than or equal and less than or equal.

- Generating the greater than or equal compare type can be done by using the greater than compare type and programming the comparator to the needed value minus 1.
- Generating the less than or equal compare type can be done by using the less than compare type and programming the comparator to the needed value plus 1.

This method does not work for the following boundary cases:

- Less than or equal of the largest unsigned number (1111...1)
- Greater than or equal of the smallest unsigned number (0000...0)
- Less than or equal of the maximum positive number when in signed mode (0111...1)
- Greater than or equal of the maximum negative number when in signed mode (1000...)

These boundary cases need no special support because they all mean ‘always true’ and can be programmed using the ignore option of the load/store watchpoint programming (refer to **22.3 Watchpoints and Breakpoints Support**).

22.3.2 Instruction Support

There are four instruction address comparators A,B,C, and D. Each is 30 bits long, generating two output signals: equal and less than. These signals are used to generate one of the following four events: equal, not equal, greater than, less than.

The instruction watchpoints and breakpoint are generated using these events and according to user programming. Note that using the OR option enables “out of range” detect.

Table 22-6 Instruction Watchpoints Programming Options

Name	Description	Programming options
IWP0	First instruction watchpoint	Comparator A Comparators (A&B)
IWP1	Second instruction watchpoint	Comparator B Comparator (A | B)
IWP2	Third instruction watchpoint	Comparator C Comparators (C&D)
IWP3	Fourth instruction watchpoint	Comparator D Comparator (C | D)

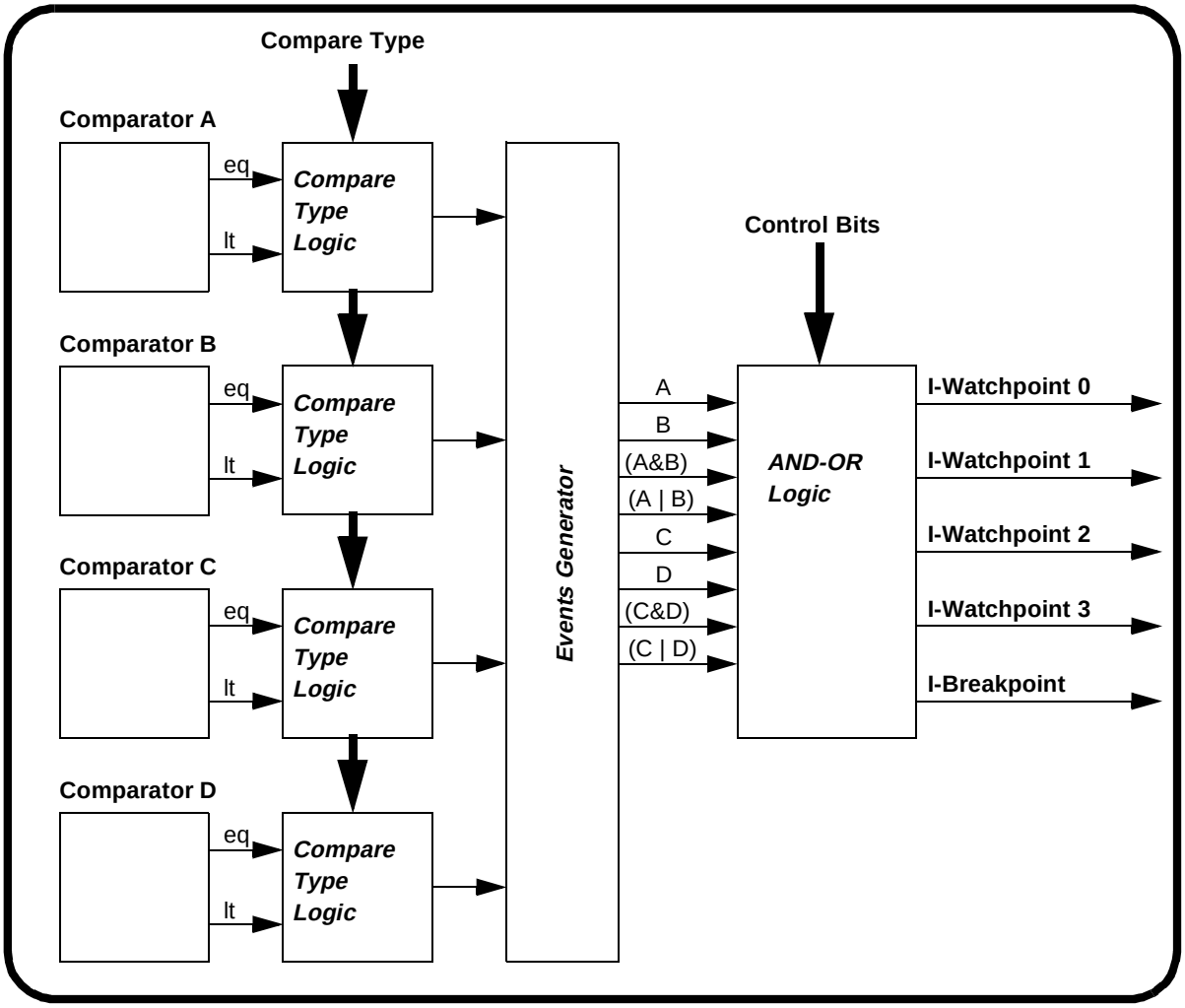


Figure 22-3 Instruction Support General Structure

22.3.2.1 Load/Store Support

There are two load/store address comparators E, and F. Each compares the 32 address bits and the cycle's attributes (read/write). The two least-significant bits are

masked (ignored) whenever a word is accessed and the least-significant bit is masked whenever a half-word is accessed. (For more information refer to **22.3.1.2 Byte and Half-Word Working Modes**). Each comparator generates two output signals: equal and less than. These signals are used to generate one of the following four events (one from each comparator): equal, not equal, greater than, less than.

There are two load/store data comparators (comparators G,H) each is 32 bits wide and can be programmed to treat numbers either as signed values or as unsigned values. Each data comparator operates as four independent byte comparators. Each byte comparator has a mask bit and generates two output signals: equal and less than, if the mask bit is not set. Therefore, each 32 bit comparator has eight output signals.

These signals are used to generate the “equal and less than” signals according to the compare size programmed (byte, half-word, word). When operating in byte mode all signals are significant, when operating in half-word mode only four signals from each 32 bit comparator are significant. When operating in word mode only two signals from each 32 bit comparator are significant.

From the *new* “equal and less than” signals and according to the compare type programmed one of the following four match events are generated: equal, not equal, greater than, less than. Therefore, from the two 32-bit comparators eight match indications are generated: Gmatch[0:3], Hmatch[0:3].

According to the lower bits of the address and the size of the cycle, only match indications that were detected on bytes that have valid information are validated, the rest are negated. Note that if the cycle executed has a smaller size than the compare size (e.g., a byte access when the compare size is word or half-word) no match indication will be asserted.

Using the match indication signals four load/store data events are generated in the following way.

Table 22-7 Load/Store Data Events

Event Name	Event Function[1]
G	(Gmatch0 | Gmatch1 | Gmatch2 | Gmatch3)
H	(Hmatch0 | Hmatch1 | Hmatch2 | Hmatch3)
(G&H)	((Gmatch0 & Hmatch0) | (Gmatch1 & Hmatch1) | (Gmatch2 & Hmatch2) | (Gmatch3 & Hmatch3))
(G | H)	((Gmatch0 | Hmatch0) | (Gmatch1 | Hmatch1) | (Gmatch2 | Hmatch2) | (Gmatch3 | Hmatch3))

NOTES:
1. ‘&’ denotes a logical AND, ‘|’ denotes a logical OR

The four load/store data events together with the match events of the load/store address comparators and the instruction watchpoints are used to generate the load/store watchpoints and breakpoint according to the programming.

Table 22-8 Load/Store Watchpoints Programming Options

Name	Description	Instruction Events Programming Options	L-address Events Programming Options	L-data Events Programming Options
LWP0	First Load/store watch-point	IWP0, IWP1, IWP2, IWP3, ignore instruction events	Comparator E Comparator F Comparators (E&F) Comparators (E | F) ignore L-addr events	Comparator G Comparator H Comparators (G&H) Comparators (G | H) ignore L-data events
LWP1	Second Load/store watch-point	IWP0, IWP1, IWP2, IWP3, ignore instruction events	Comparator E Comparator F Comparators (E&F) Comparators (E | F) ignore l-addr events	Comparator G Comparator H Comparators (G&H) Comparators (G | H) ignore L-data events

Note that when programming the load/store watchpoints to ignore L-addr events and L-data events, it does not reduce the load/store watchpoints detection logic to be instruction watchpoint detection logic since the instruction must be a load/store instruction for the load/store watchpoint event to trigger.

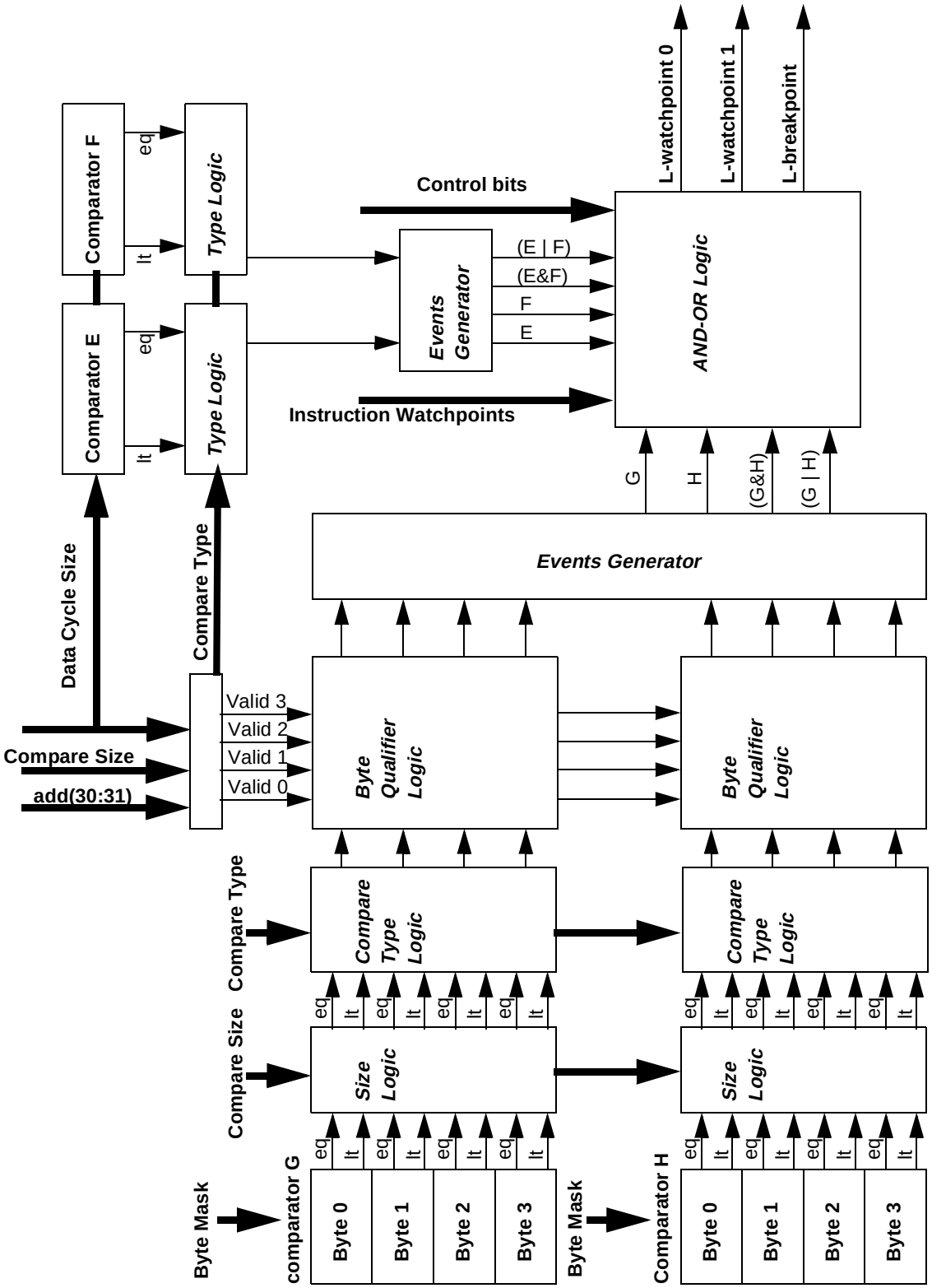


Figure 22-4 Load/Store Support General Structure

22.3.3 Watchpoint Counters

There are two 16-bit watchpoint counters. Each counter is able to count one of the instruction watchpoints or one of the load/store watchpoints. Both generate the corresponding breakpoint when they reach ZERO.

When working in the *masked* mode, the counters do **not** count watchpoints detected when MSRRI = 0. See **22.3.1.4 Context Dependent Filter**

The counters value when counting watchpoints programmed on the actual instructions that alter the counters, are not predictable. Reading values from the counters when they are active, must be synchronized by inserting a sync instruction before the actual read is performed.

NOTE

When programmed to count instruction watchpoints, the last instruction which decrements the counter to ZERO is treated like any other instruction breakpoint in the sense that it is not executed and the machine branches to the breakpoint exception routine BEFORE it executes this instruction. As a side effect of this behavior, the value of the counter inside the breakpoint exception routine equals ONE and not ZERO as might be expected.

When programmed to count load/store watchpoints, the last instruction which decrements the counter to ZERO is treated like any other load/store breakpoint in the sense that it is executed and the machine branches to the breakpoint exception routine AFTER it executes this instruction. Therefore, the value of the counter inside the breakpoint exception routine equals ZERO.

22.3.3.1 Trap Enable Programming

The trap enable bits can be programmed by regular software (only if MSRPR = 0) using THE **mtspr** instruction or “on the fly” using the special development port interface. For more information refer to section **22.5.6.5 Development Port Serial Communications — Trap Enable Mode**.

The value used by the breakpoints generation logic is the bit wise **OR** of the software trap enable bits, (the bits written using the **mtspr**) and the development port trap enable bits (the bits serially shifted using the development port).

All bits, the software trap enable bits and the development port trap enable bits, can be read from ICTRL and the LCTRL2 using **mfspr**. For the exact bits placement refer to **22.7.6 I-Bus Support Control Register** and to **22.7.8 L-Bus Support Control Register 2**

22.4 Development System Interface

When debugging an existing system, it is sometimes desirable to be able to do so without the need to insert any changes in the existing system. In some cases it is not

desired, or even impossible, to add load to the lines connected to the existing system. The development system interface of the CPU supports such a configuration.

The development system interface of the CPU uses a dedicated serial port (the development port) and, therefore, does not need any of the regular system interfaces. Controlling the activity of the system from the development port is done when the CPU is in the debug mode. The development port is a relatively economical interface (three pins) that allows the development system to operate in a lower frequency than the frequency of the CPU. Note that it is also possible to debug the CPU using monitor debugger software, for more information refer to **22.6 Software Monitor Debugger Support**.

Debug mode is a state where the CPU fetches all instructions from the development port. In addition, when in debug mode, data can be read from the development port and written to the development port. This allows memory and registers to be read and modified by a development tool (emulator) connected to the development port.

For protection purposes, two possible working modes are defined: debug mode enable and debug mode disable. These working modes are selected only during reset. For more information refer to **22.4.1.1 Debug Mode Enable vs. Debug Mode Disable**

The user can work in debug mode starting from reset or the CPU can be programmed to enter debug mode as a result of a predefined list of events. These events include all possible interrupts and exceptions in the CPU system, including the internal breakpoints, together with two levels of development port requests (*masked* and *non-masked*) and one peripheral breakpoint request that can be generated by any one of the peripherals of the system (including internal and external modules). Each event can be programmed either to be treated as a regular interrupt that causes the machine to branch to its interrupt vector, or to be treated as a special interrupt that causes debug mode entry.

When in debug mode an **rfi** instruction will return the machine to its regular work mode.

The relationship between the debug mode logic to the rest of the CPU chip is shown in the following figure.

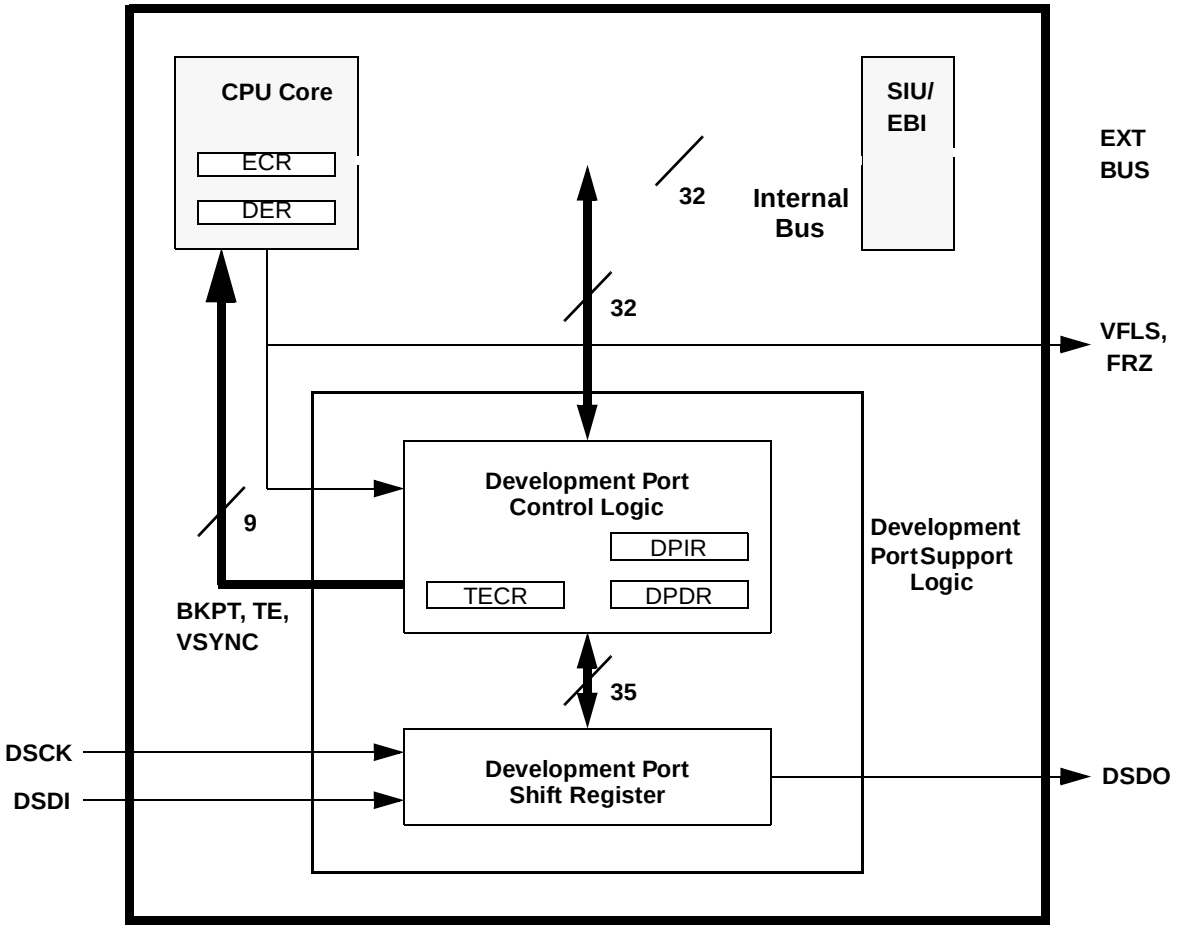


Figure 22-5 Functional Diagram of MPC565 / MPC566 Debug Mode Support

The development port provides a full duplex serial interface for communications between the internal development support logic of the CPU and an external development tool. The development port can operate in two working modes: the trap enable mode and the debug mode.

The trap enable mode is used in order to shift into the CPU internal development support logic the following control signals:

1. Instruction trap enable bits, used for on the fly programming of the instruction breakpoint
2. Load/store trap enable bits, used for on the fly programming of the load/store breakpoint
3. Non-maskable breakpoint, used to assert the non-maskable external breakpoint
4. Maskable breakpoint, used to assert the maskable external breakpoint
5. **VSYNC**, used to assert and negate **VSYNC**

In debug mode the development port controls also the debug mode features of the CPU. For more information **22.5 Development Port**

22.4.1 Debug Mode Support

The debug mode of the CPU provides the development system with the following basic functions:

- Gives an ability to control the execution of the processor and maintain control on it under all circumstances. The development port is able to force the CPU to enter to the debug mode even when external interrupts are disabled.
- It is possible to enter debug mode immediately out of reset thus allowing debugging of a ROM-less system.
- It is possible to selectively define, using an enable register, the events that will cause the machine to enter into the debug mode.
- When in debug mode detect the reason upon which the machine entered debug mode by reading a cause register.
- Entering into the debug mode in all regular cases is restartable in the sense that that it is possible to continue to run the regular program from the location where it entered the debug mode.
- When in debug mode all instructions are fetched from the development port but load/store accesses are performed on the real system memory.
- Data Register of the development port is accessed using **mtspr** and **mfspr** instructions via special load/store cycles. (This feature together with the last one enables easy memory dump & load).
- Upon entering debug mode, the processor gets into the privileged state (MSRPR = 0). This allows execution of any instruction, and access to any storage location.
- An OR signal of all exception cause register (ECR) bits (ECR_OR) enables the development port to detect pending events while already in debug mode. An example is the ability of the development port to detect a debug mode access to a non existing memory space.

The following figure illustrates the debug mode logic implemented in the CPU.

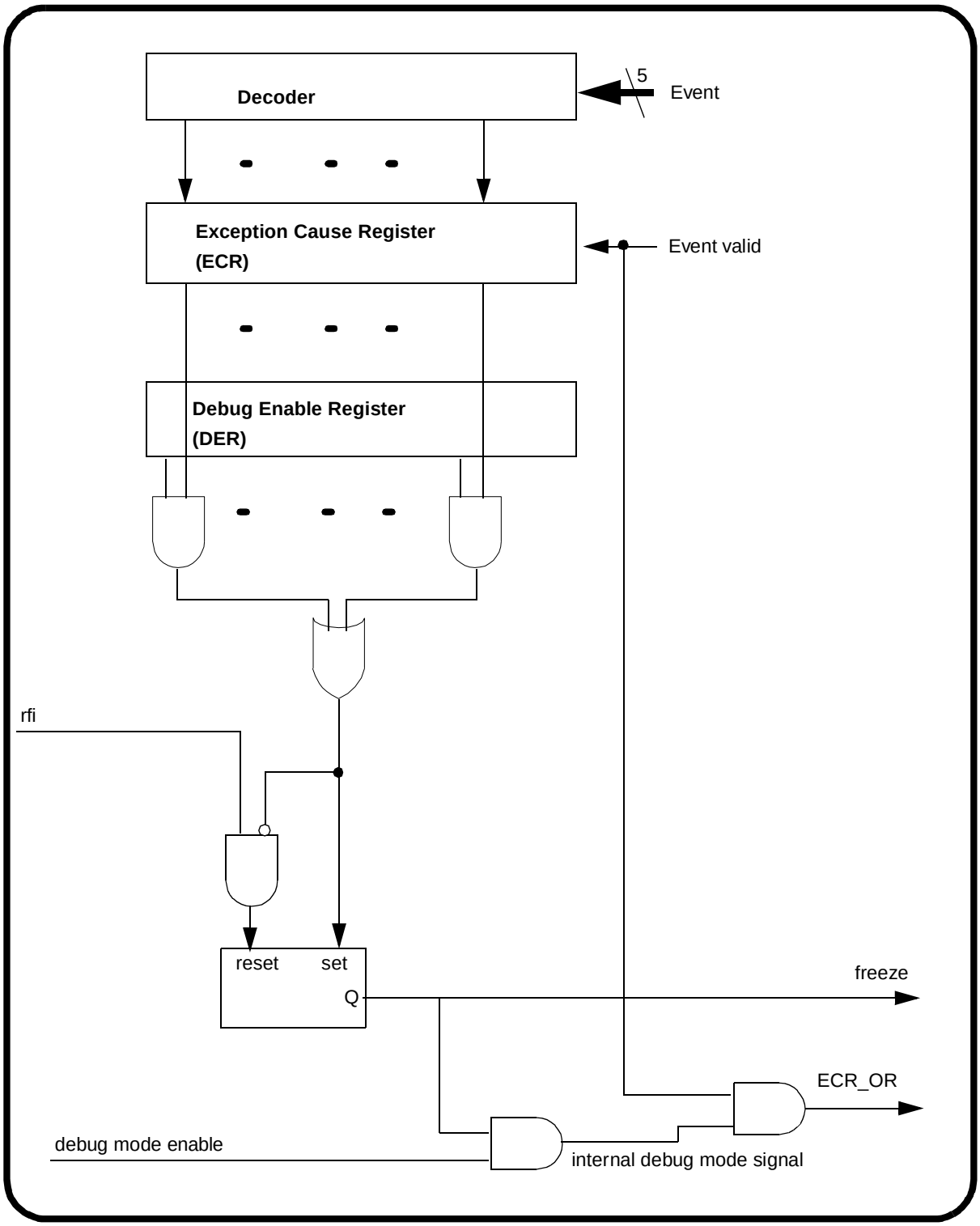


Figure 22-6 Debug Mode Logic

22.4.1.1 Debug Mode Enable vs. Debug Mode Disable

For protection purposes two possible working modes are defined: debug mode enable and debug mode disable. These working modes are selected only during reset.

Debug mode is enabled by asserting the DSCK pin during reset. The state of this pin is sampled three clocks before the negation of $\overline{\text{SRESET}}$.

NOTE

Since $\overline{\text{SRESET}}$ negation is done by an external pull up resistor any reference here to $\overline{\text{SRESET}}$ negation time refers to the time the MPC565 / MPC566 releases $\overline{\text{SRESET}}$. If the actual negation is slow due to large resistor, set up time for the debug port signals should be set accordingly.

If the DSCK pin is sampled negated, debug mode is disabled until a subsequent reset when the DSCK pin is sampled in the asserted state. When debug mode is disabled the internal watchpoint/breakpoint hardware will still be operational and may be used by a software monitor program for debugging purposes.

When working in debug mode disable, all development support registers (see list in **Table 22-14**) are accessible to the supervisor code (MSRPR = 0) and can be used by a monitor debugger software. However, the processor *never* enters debug mode and, therefore, the exception cause register (ECR) and the debug enable register (DER) are used only for asserting and negating the freeze signal. For more information on the software monitor debugger support refer to **22.6 Software Monitor Debugger Support**.

When working in debug mode enable, all development support registers are accessible *only* when the CPU is in debug mode. Therefore, even supervisor code that may be still under debug cannot prevent the CPU from entering debug mode. The development system has full control of all development support features of the CPU through the development port. Refer to **Table 22-16**

22.4.1.2 Entering Debug Mode

Entering debug mode can be a result of a number of events. All events have a programmable enable bit to selectively decide which events result in debug mode entry and which in regular interrupt handling.

Entering debug mode is also possible immediately out of reset, thus allowing the debugging of even a ROM-less system. Using this feature is possible by special programming of the development port during reset. If the DSCK pin continues to be asserted following $\overline{\text{SRESET}}$ negation (after enabling debug mode) the processor will take a breakpoint exception and go directly to debug mode instead of fetching the reset vector. To avoid entering debug mode following reset, the DSCK pin must be negated no later than seven clock cycles after $\overline{\text{SRESET}}$ negates. In this case, the processor will jump to the reset vector and begin normal execution. When entering debug mode immediately after reset, bit 31 (development port interrupt) of the exception cause register (ECR) is set.

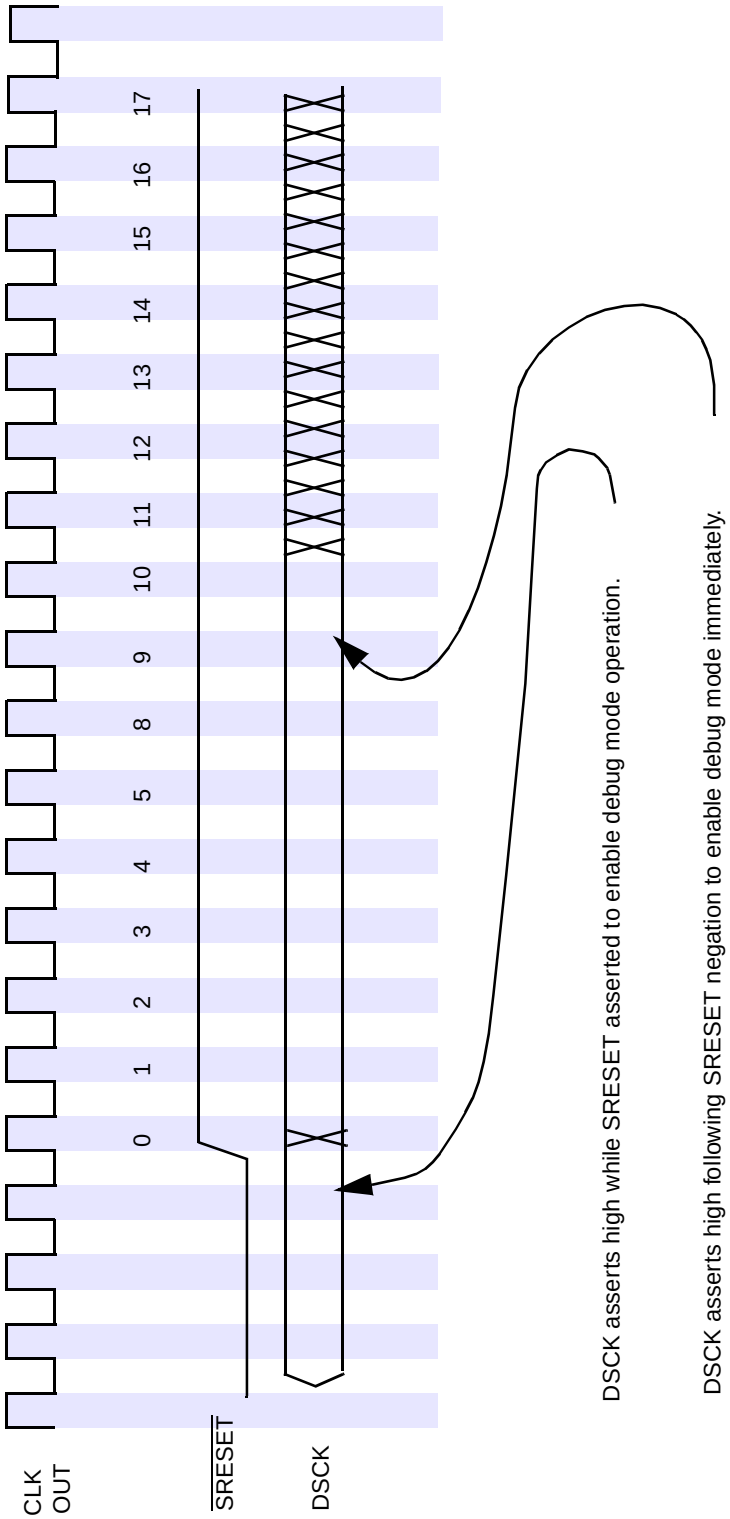


Figure 22-7 Debug Mode Reset Configuration

When debug mode is disabled all events result in regular interrupt handling.

The internal freeze signal is asserted whenever an enabled event occurs, regardless if debug mode is enabled or disabled. The internal freeze signal is connected to all relevant internal modules. These modules can be programmed to stop all operations in response to the assertion of the freeze signal. Refer to **22.6.1 Freeze Indication**.

The freeze indication is negated when exiting debug mode. Refer to **22.4.1.6 Exiting Debug Mode**

The following list contains the events that can cause the CPU to enter debug mode. Each event results in debug mode entry if debug mode is enabled and the corresponding enable bit is set. The reset values of the enable bits allow, in most cases, the use of the debug mode features without the need to program the debug enable register (DER). For more information refer to **22.7.12 Debug Enable Register (DER)**.

- NMI exception as a result of the assertion of the IRQ0_B pin. For more information refer to **3.15.4.1 System Reset Interrupt**
- Check stop. Refer to **22.4.1.3 The Check Stop State and Debug Mode**
- Machine check exception
- Implementation specific instruction protection error
- Implementation specific data protection error
- External interrupt, recognized when MSREE = 1
- Alignment interrupt
- Program interrupt
- Floating point unavailable exception
- Floating point assist exception
- Decrementer exception, recognized when MSREE = 1
- System call exception
- Trace, asserted when in single trace mode or when in branch trace mode (refer to **3.15.4.10 Trace Interrupt**)
- Implementation dependent software emulation exception
- Instruction breakpoint, when breakpoints are *masked* (BRKNOMSK bit in the LCTRL2 is clear) recognized only when MSRRI = 1, when breakpoints are not *mask*ed (BRKNOMSK bit in the LCTRL2 is set) always recognized
- Load/store breakpoint, when breakpoints are *masked* (BRKNOMSK bit in the LCTRL2 is cleared) recognized only when MSRRI = 1, when breakpoints are not *masked* (BRKNOMSK bit in the LCTRL2 is set) always recognized
- Peripherals breakpoint, from the development port, internal and external modules. are recognized only when MSRRI = 1.
- Development port non-maskable interrupt, as a result of a debug station request. Useful in some catastrophic events like an endless loop when MSRRI = 0. As a result of this event the machine may enter a non-restartable state, for more information refer to **3.15.4 Interrupts**.

The processor enters into the debug mode state when at least one of the bits in the exception cause register (ECR) is set, the corresponding bit in the debug enable register (DER) is enabled and debug mode is enabled. When debug mode is enabled and an enabled event occurs, the processor waits until its pipeline is empty and then starts fetching the next instructions from the development port. For information on the exact value of machine status save/restore registers (SRR0 and SRR1) refer to **3.15.4 Interrupts**

When the processor is in debug mode the freeze indication is asserted thus allowing any peripheral that is programmed to do so to stop. The fact that the CPU is in debug mode is also broadcast to the external world using the value b11 on the VF**LS** pins.

NOTE

The freeze signal can be asserted by software when debug mode is disabled.

The development port should read the value of the exception cause register (ECR) in order to get the cause of the debug mode entry. *Reading the exception cause register (ECR) clears all its bits*.

22.4.1.3 The Check Stop State and Debug Mode

The CPU enters the check stop state if the machine check interrupt is disabled (MSRME = 0) and a machine check interrupt is detected. However, if a machine check interrupt is detected when MSRME = 0, debug mode is enabled and the check stop enable bit in the debug enable register (DER) is set, the CPU enters debug mode rather then the check stop state.

The different actions taken by the CPU when a machine check interrupt is detected are shown in the following table.

Table 22-9 The Check Stop State and Debug Mode

MSR_{ME}	Debug Mode Enable	CHSTPE[1]	MCIE[2]	Action Performed by the CPU when Detecting a Machine Check Interrupt	Exception Cause Register (ECR) Value
0	0	X	X	Enter the check stop state	0x20000000
1	0	X	X	Branch to the machine check interrupt	0x10000000
0	1	0	X	Enter the check stop state	0x20000000
0	1	1	X	Enter Debug Mode	0x20000000
1	1	X	0	Branch to the machine check interrupt	0x10000000
1	1	X	1	Enter Debug Mode	0x10000000

NOTES:
1. Check stop enable bit in the debug enable register (DER)
2. Machine check interrupt enable bit in the debug enable register (DER)

22.4.1.4 Saving Machine State upon Entering Debug Mode

If entering debug mode was as a result of any load/store type exception, and therefore the DAR (data address register) and DSISR (data storage interrupt status register)

have some significant value, these two registers must be saved before any other operation is performed. Failing to save these registers may result in loss of their value in case of another load/store type exception inside the development software.

Since exceptions are treated differently when in debug mode (refer to **22.4.1.5 Running in Debug Mode**), there is no need to save machine status save/restore zero register (SRR0) and machine status save/restore one register (SRR1).

22.4.1.5 Running in Debug Mode

When running in debug mode all fetch cycles access the development port regardless of the actual address of the cycle. All load/store cycles access the real memory system according to the cycle's address. The data register of the development port is mapped as a special control register therefore it is accessed using **mtspr** and **mfspr** instructions via special load/store cycles (refer to **22.7.13 Development Port Data Register (DPDR)**).

Exceptions are treated differently when running in debug mode. When already in debug mode, upon recognition of an exception, the exception cause register (ECR) is updated according to the event that caused the exception, a special error indication (ECR_OR) is asserted for one clock cycle to report to the development port that an exception occurred and execution continues in debug mode without any change in SRR0 and SRR1. ECR_OR is asserted before the next fetch occurs to allow the development system to detect the excepting instruction.

Not all exceptions are recognized when in debug mode. Breakpoints and watchpoints are not generated by the hardware when in debug mode (regardless of the value of MSRRI). Upon entering debug mode MSREE is cleared by the hardware thus forcing the hardware to ignore external and decrementer interrupts.

WARNING

Setting the MSREE bit while in debug mode, (by the debug software), is strictly forbidden. The reason for this restriction is that the external interrupt event is a level signal, and since the CPU only reports exceptions while in debug mode but do not treat them, the CPU does not clear the MSREE bit and, therefore, this event, if enabled, is recognized again every clock cycle.

When the ECR_OR signal is asserted the development station should investigate the exception cause register (ECR) in order to find out the event that caused the exception.

Since the values in SRR0 and SRR1 do not change if an exception is recognized while already in debug mode, they only change once when entering debug mode, saving them when entering debug mode is not necessary.

22.4.1.6 Exiting Debug Mode

The **rfi** instruction is used to exit from debug mode in order to return to the normal processor operation and to negate the freeze indication. The development system may

monitor the freeze status to make sure the MPC565 / MPC566 is out of debug mode. It is the responsibility of the software to read the exception cause register (ECR) before performing the **rfi**. Failing to do so will force the CPU to immediately re-enter to debug mode and to re-assert the freeze indication in case an asserted bit in the interrupt cause register (ECR) has a corresponding enable bit set in the debug enable register (DER).

22.5 Development Port

The development port provides a full duplex serial interface for communications between the internal development support logic including debug mode and an external development tool.

The relationship of the development support logic to the rest of the CPU chip is shown in **Figure 22-5**. The development port support logic is shown as a separate block for clarity. It is implemented as part of the SIU module.

22.5.1 Development Port Pins

The following development port pin functions are provided:

1. Development serial clock (DSCK)
2. Development serial data in (DSDI)
3. Development serial data out (DSDO)

22.5.2 Development Serial Clock

The development serial clock (DSCK) is used to shift data into and out of the development port shift register. At the same time, the new most significant bit of the shift register is presented at the DSDO pin. In all further discussions references to the DSCK signal imply the internal synchronized value of the clock. The DSCK input must be driven either high or low at all times and not allowed to float. A typical target environment would pull this input low with a resistor.

The clock may be implemented as a free running clock or as gated clock. As discussed in section **22.5.6.5 Development Port Serial Communications — Trap Enable Mode** and section **22.5.6.8 Development Port Serial Communications — Debug Mode**, the shifting of data is controlled by ready and start signals so the clock does not need to be gated with the serial transmissions.

The DSCK pin is also used at reset to enable debug mode and immediately following reset to optionally cause immediate entry into debug mode following reset.

22.5.3 Development Serial Data In

Data to be transferred into the development port shift register is presented at the development serial data in (DSDI) pin by external logic. To be sure that the correct value is used internally. When driven asynchronous (synchronous) with the system clock, the data presented to DSDI must be stable a setup time before the rising edge of DSCK (CLKOUT) and a hold time after the rising edge of DSCK (CLKOUT).

The DSDI pin is also used at reset to control the overall chip configuration mode and to determine the development port clock mode. See section **22.5.6.4 Development Port Serial Communications — Clock Mode Selection** for more information.

22.5.4 Development Serial Data Out

The debug mode logic shifts data out of the development port shift register using the development serial data out (DSDO) pin. All transitions on DSDO are synchronous with DSCK or CLKOUT depending on the clock mode. Data will be valid a setup time before the rising edge of the clock and will remain valid a hold time after the rising edge of the clock.

Refer to **Table 22-12** for DSDO data meaning.

22.5.5 Freeze Signal

The freeze indication means that the processor is in debug mode (i.e., normal processor execution of user code is frozen). On the MPC565 / MPC566, the freeze state can be indicated by three different pins. The FRZ signal is generated synchronously with the system clock. This indication may be used to halt any off-chip device while in debug mode as well as a handshake means between the debug tool and the debug port. The internal freeze status can also be monitored through status in the data shifted out of the debug port.

22.5.5.1 SGPIO6/FRZ/$\overline{\text{PTR}}$ Pin

The SGPIO6/FRZ/$\overline{\text{PTR}}$ pin powers up as the $\overline{\text{PTR}}$ function and its function is controlled by the GPC bits in the SIUMCR.

22.5.5.2 IWP[0:1]/VFLS[0:1] Pins

The IWP[0:1]/VFLS[0:1] pins power up as the VFLS[0:1] function and their function can be changed via the DBGC bits in the SIUMCR. They can also be set via the reset configuration word (See **7.5.2 Hard Reset Configuration Word**). The FRZ state is indicated by the value b11 on the VFLS[0:1] pins.

22.5.5.3 VFLS[0:1]_MPIO32B[3:4] Pins

The VFLS[0:1]_MPIO32B[3:4] Pins power up as the MPIO32B[3:4] function and their function can be changed via the VFLS bit in the MIOS1TPCR register. The FRZ state is indicated by the value b11 on the VFLS[0:1] pins.

22.5.6 Development Port Registers

The development port consists logically of the three registers: development port instruction register (DPIR), development port data register (DPDR), and trap enable control register (TECR). These registers are physically implemented as two registers, development port shift register and trap enable control register. The development port shift register acts as both the DPIR and DPDR depending on the operation being performed. It is also used as a temporary holding register for data to be stored into the TECR. These registers are discussed below in more detail.

22.5.6.1 Development Port Shift Register

The development port shift register is a 35-bit shift register. Instructions and data are shifted into it serially from DSDI using DSCK (or CLKOUT depending on the debug port clock mode, refer to **22.5.6.4 Development Port Serial Communications — Clock Mode Selection**) as the shift clock. These instructions or data are then transferred in parallel to the CPU, the trap enable control register (TECR). When the processor enters debug mode it fetches instructions from the DPIR which causes an access to the development port shift register. These instructions are serially loaded into the shift register from DSDI using DSCK (or CLKOUT) as the shift clock. In a similar way, data is transferred to the CPU by moving it into the shift register which the processor reads as the result of executing a “move from special purpose register DPDR” instruction. Data is also parallel-loaded into the development port shift register from the CPU by executing a “move to special purpose register DPDR” instruction. It is then shifted out serially to DSDO using DSCK (or CLKOUT) as the shift clock.

22.5.6.2 Trap Enable Control Register

The trap enable control register is a 9-bit register that is loaded from the development port shift register. The contents of the control register are used to drive the six trap enable signals, the two breakpoint signals, and the VSYNC signal to the CPU. The “transfer data to trap enable control register” commands will cause the appropriate bits to be transferred to the control register.

The trap enable control register is not accessed by the CPU, but instead supplies signals to the CPU. The trap enable bits, VSYNC bit, and the breakpoint bits of this register are loaded from the development port shift register as the result of trap enable mode transmissions. The trap enable bits are reflected in ICTRL and LCTRL2 special registers. See **22.7.6 I-Bus Support Control Register** and **22.7.8 L-Bus Support Control Register 2**.

22.5.6.3 Development Port Registers Decode

The development port shift register is selected when the CPU accesses DPIR or DPDR. Accesses to these two special purpose registers occur in debug mode and appear on the internal bus as an address and the assertion of an address attribute signal indicating that a special purpose register is being accessed. The DPIR register is read by the CPU to fetch all instructions when in debug mode and the DPDR register is read and written to transfer data between the CPU and external development tools. The DPIR and DPDR are pseudo registers. Decoding either of these registers will cause the development port shift register to be accessed. The debug mode logic knows whether the CPU is fetching instructions or reading or writing data. If what the CPU is expecting and what the register receives from the serial port do not match (instruction instead of data) the mismatch is used to signal a sequence error to the external development tool.

22.5.6.4 Development Port Serial Communications — Clock Mode Selection

All of the serial transmissions are clock transmissions and are therefore synchronous communications. However, the transmission clock may be either synchronous or

asynchronous with the system clock (CLKOUT). The development port allows the selection of two methods for clocking the serial transmissions. The first method allows the transmission to occur without being externally synchronized with CLKOUT, in this mode a serial clock DSCK must be supplied to the MPC565 / MPC566. The other communication method requires a data to be externally synchronized with CLKOUT.

The first clock mode is called “asynchronous clock” since the input clock (DSCK) is asynchronous with CLKOUT. To be sure that data on DSDI is sampled correctly, transitions on DSDI must occur a setup time ahead and a hold time after the rising edge of DSCK. This clock mode allows communications with the port from a development tool which does not have access to the CLKOUT signal or where the CLKOUT signal has been delayed or skewed. Refer to the timing diagram in **Figure 22-8**

The second clock mode is called “synchronous self clock”. It does not require an input clock. Instead the port is timed by the system clock. The DSDI input is required to meet setup and hold time requirements with respect to CLKOUT rising edge. The data rate for this mode is always the same as the system clock. Refer to the timing diagram in **Figure 22-9**.

The selection of clock or self clock mode is made at reset. The state of the DSDI input is latched eight clocks after $\overline{\text{SRESET}}$ negates. If it is latched low, asynchronous clock mode is enabled. If it is latched high then synchronous self clock mode is enabled.

Since DSDI is used to select the development port clocking scheme, it is necessary to prevent any transitions on DSDI during this time from being recognized as the start of a serial transmission. The port will not begin scanning for the start bit of a serial transmission until 16 clocks after the negation of $\overline{\text{SRESET}}$. If DSDI is asserted 16 clocks after $\overline{\text{SRESET}}$ negation, the port will wait until DSDI is negated to begin scanning for the start bit.

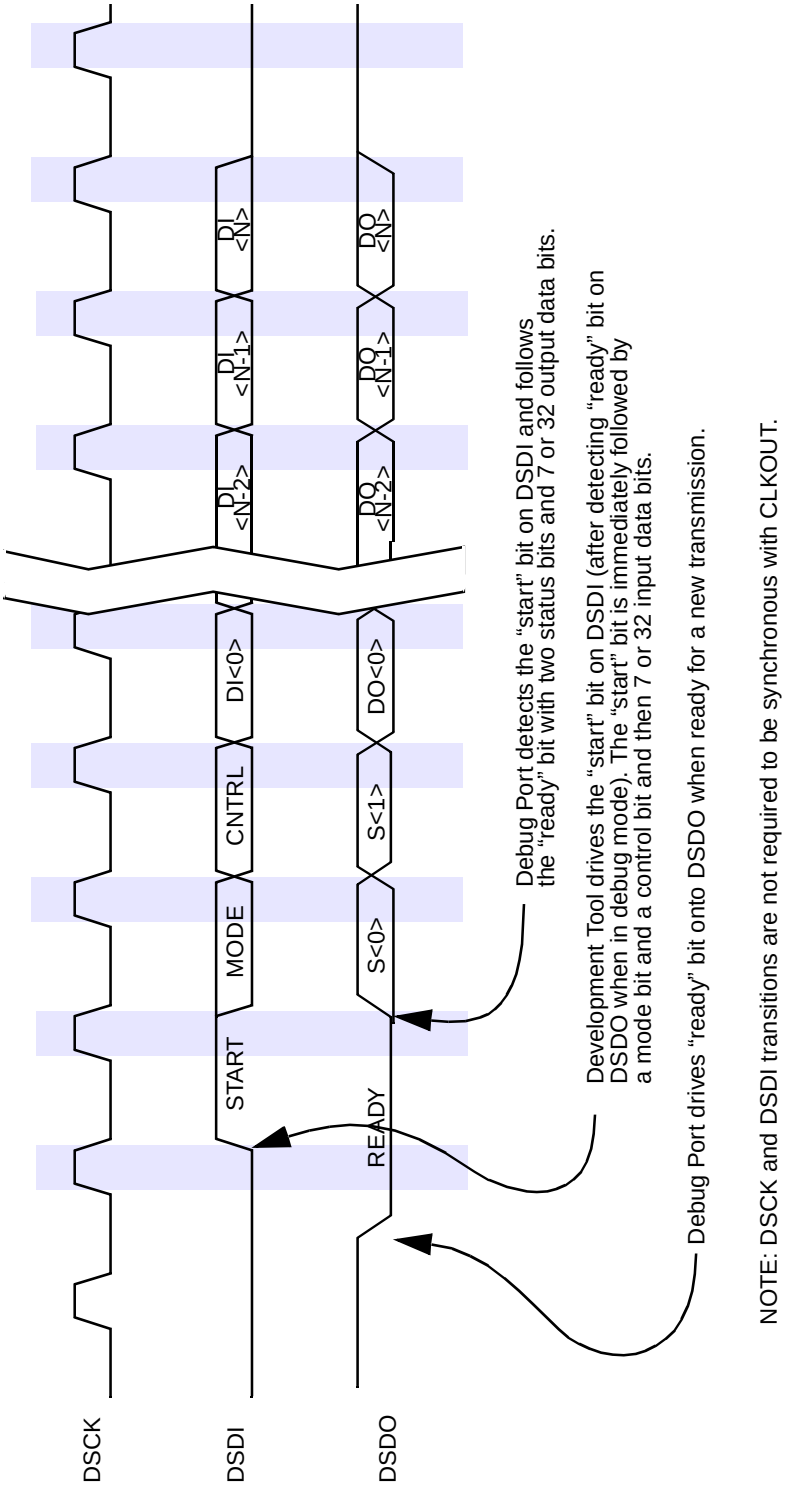


Figure 22-8 Asynchronous Clock Serial Communications

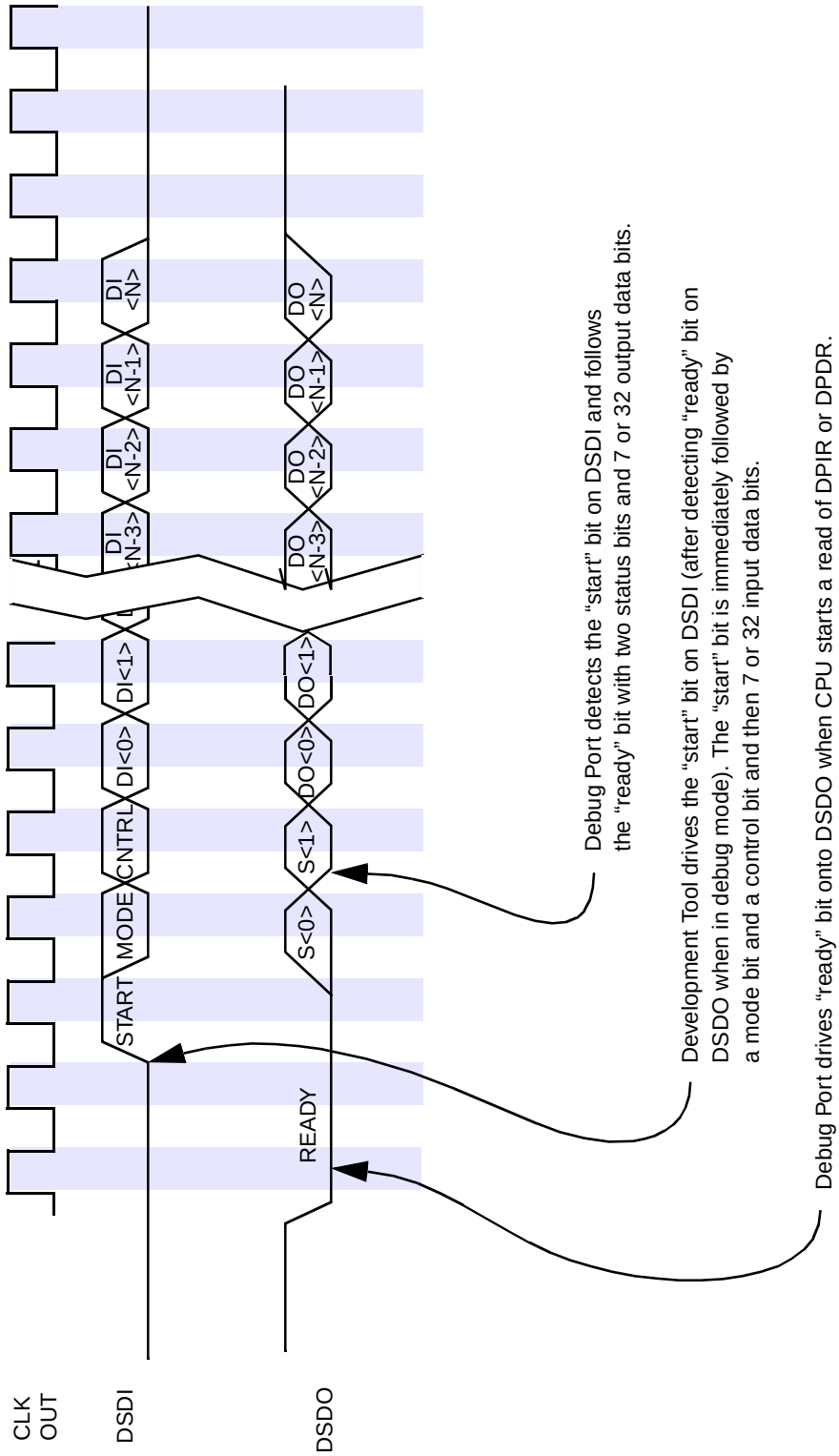


Figure 22-9 Synchronous Self Clock Serial Communication

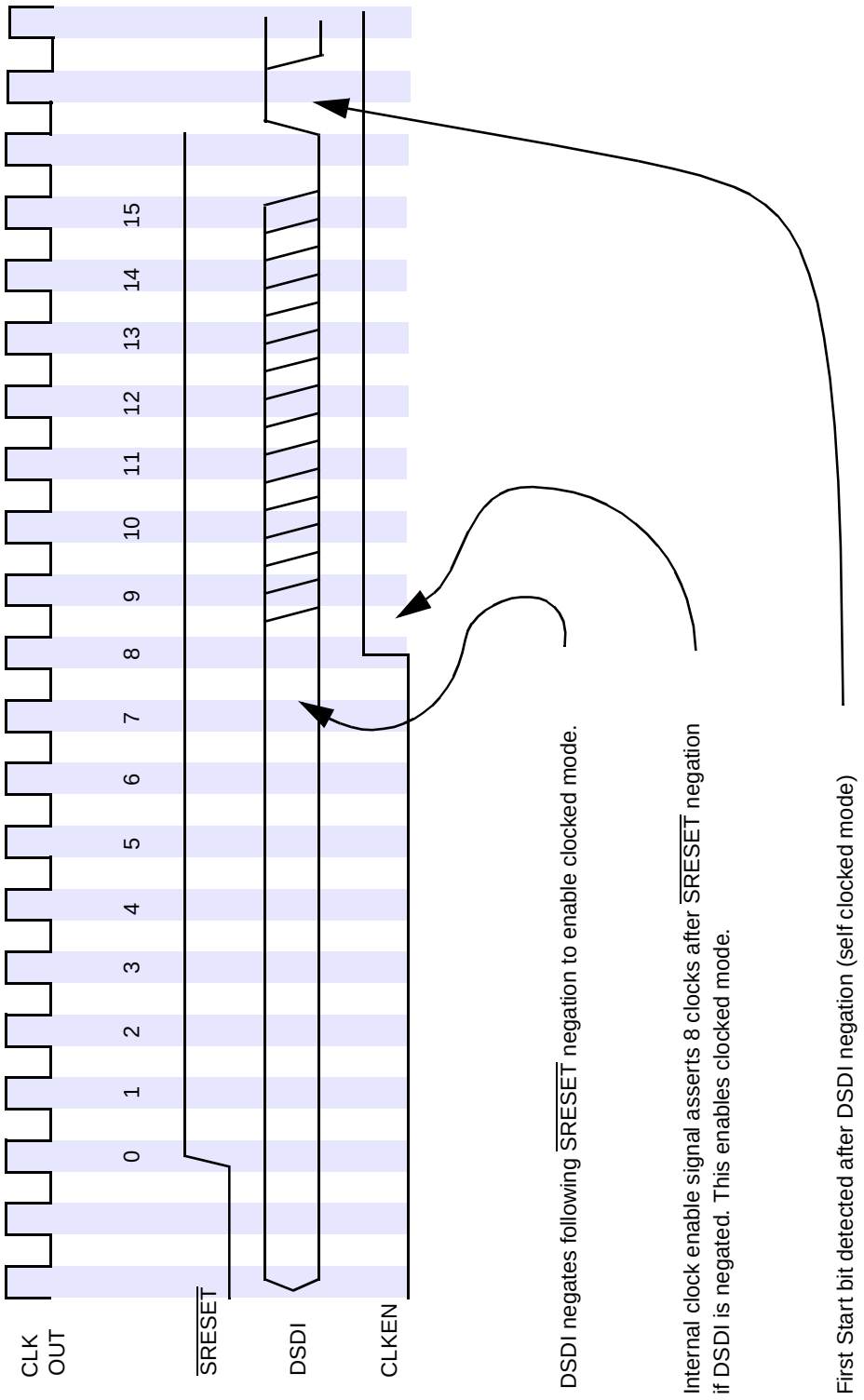


Figure 22-10 Enabling Clock Mode Following Reset

22.5.6.5 Development Port Serial Communications — Trap Enable Mode

When in not in debug mode the development port starts communications by setting DSDO (the MSB of the 35-bit development port shift register) low to indicate that all activity related to the previous transmission are complete and that a new transmission may begin. The start of a serial transmission from an external development tool to the development port is signaled by a start bit. A mode bit in the transmission defines the transmission as either a trap enable mode transmission or a debug mode transmission. If the mode bit is set the transmission will only be 10 bits long and only seven data bits will be shifted into the shift register. These seven bits will be latched into the TECR. A control bit determines whether the data is latched into the trap enable and VSYNC bits of the TECR or into the breakpoints bits of the TECR.

22.5.6.6 Serial Data into Development Port — Trap Enable Mode

The development port shift register is 35 bits wide but trap enable mode transmissions only use the start/ready bit, a mode/status bit, a control/status bit, and the seven least significant data bits. The encoding of data shifted into the development port shift register (through the DSDI pin) is shown in **Table 22-10** and **Table 22-11** below:

Table 22-10 Trap Enable Data Shifted into Development Port Shift Register

Start	Mode	Control	1st	2nd	3rd	4th	1st	2nd	VSYNC	Function
			- - - - - - Instruction- - - - - -				- - Data- -			
			Watchpoint Trap Enables							
1	1	0	0 = disabled; 1 = enabled							Transfer Data to Trap Enable Control Register

Table 22-11 Debug Port Command Shifted Into Development Port Shift Registe r

Start	Mode	Control	Extended Opcode		Major Opcode	Function
1	1	1	x	x	00000	NOP
					00001	Hard Reset request
					00010	Soft Reset request
			0	x	00011	Reserved
			1	0	00011	End Download procedure
			1	1	00011	Start Download procedure
			x	x	00100... 11110	Reserved
			x	0	11111	Negate Maskable breakpoint.
			x	1	11111	Assert Maskable breakpoint.
			0	x	11111	Negate Non Maskable breakpoint.
			1	x	11111	Assert Non Maskable breakpoint.

The watchpoint trap enables and VSYNC functions are described in section **22.3 Watchpoints and Breakpoints Support** and section **22.2 Program Flow Tracking**.

The debug port command function allows the development tool to either assert or negate breakpoint requests, reset the processor, activate or deactivate the fast download procedure.

22.5.6.7 Serial Data Out of Development Port — Trap Enable Mode

In trap enable mode the only response out of the development port is “sequencing error.”

Data that can come out of the development port is shown in **Table 22-12**. “Valid data from CPU” and “CPU interrupt” status cannot occur in trap enable mode.

Table 22-12 Status / Data Shifted Out of Development Port Shift Register

Ready	Status [0:1]		Data			Function
			Bit 0	Bit 1	Bits 2:31 or 2:6 — (Depending on Input Mode)	
(0)	0	0	Data			Valid Data from CPU
(0)	0	1	Freeze status[1]	Download Procedure in progress[2]	1’s	Sequencing Error
(0)	1	0			1’s	CPU Interrupt
(0)	1	1			1’s	Null

NOTES:
1. The “Freeze” status is set to (1) when the CPU is in debug mode and to (0) otherwise.
2. The “Download Procedure in progress” status is asserted (0) when Debug port in the Download procedure and is negated (1) otherwise.

When not in debug mode the sequencing error encoding indicates that the transmission from the external development tool was a debug mode transmission. When a sequencing error occurs the development port will ignore the data shifted in while the sequencing error was shifting out. It will be treated as a NOP function.

Finally, the null output encoding is used to indicate that the previous transmission did not have any associated errors.

When not in debug mode, ready will be asserted at the end of each transmission. If debug mode is not enabled and transmission errors can be guaranteed not to occur, the status output is not needed.

22.5.6.8 Development Port Serial Communications — Debug Mode

When in debug mode the development port starts communications by setting DSDO low to indicate that the CPU is trying to read an instruction from DPIR or data from DPDR. When the CPU writes data to the port to be shifted out the ready bit is not set. The port waits for the CPU to read the next instruction before asserting ready. This allows duplex operation of the serial port while allowing the port to control all transmissions from the external development tool. After detecting this ready status the external development tool begins the transmission to the development port with a start bit (logic high) on the DSDI pin.

22.5.6.9 Serial Data Into Development Port

In debug mode the 35 bits of the development port shift register are interpreted as a start/ready bit, a mode/status bit, a control/status bit, and 32 bits of data. All instructions and data for the CPU are transmitted with the mode bit cleared indicating a 32-bit data field. The encoding of data shifted into the development port shift register (through the DSDI pin) is shown below in **Table 22-13**

Table 22-13 Debug Instructions / Data Shifted Into Development Port Shift Register

Start	Mode	Control	Instruction / Data (32 Bits)		Function
			Bits 0:6	Bits 7:31	
1	0	0	CPU Instruction		Transfer Instruction to CPU
1	0	1	CPU Data		Transfer Data to CPU
1	1	0	Trap enable[1]	Not exist	Transfer data to Trap Enable Control Register
1	1	1	0011111	Not exist	Negate breakpoint requests to the CPU.
1	1	1	0	Not exist	nop

NOTES:
1. Refer to **Table 22-10**

Data values in the last two functions other than those specified are reserved.

All transmissions from the debug port on DSDO begin with a “0” or “ready” bit. This indicates that the CPU is trying to read an instruction or data from the port. The external development tool must wait until it sees DSDO go low to begin sending the next transmission.

The control bit differentiates between instructions and data and allows the development port to detect that an instruction was entered when the CPU was expecting data and vice versa. If this occurs a sequence error indication is shifted out in the next serial transmission.

The trap enable function allows the development tool to transfer data to the trap enable control register.

The debug port command function allows the development tool to either negate breakpoint requests, reset the processor, activate or deactivate the fast down load procedure.

The NOP function provides a null operation for use when there is data or a response to be shifted out of the data register and the appropriate next instruction or command will be determined by the value of the response or data shifted out.

22.5.6.10 Serial Data Out of Development Port

The encoding of data shifted out of the development port shift register in debug mode (through the DSDO pin) is the same as for trap enable mode and is shown in **Table 22-12**.

Valid data encoding is used when data has been transferred from the CPU to the development port shift register. This is the result of an instruction to move the contents of a general purpose register to the debug port data register (DPDR). The valid data encoding has the highest priority of all status outputs and will be reported even if an interrupt occurs at the same time. Since it is not possible for a sequencing error to occur and also have valid data there is no priority conflict with the sequencing error status. Also, any interrupt that is recognized at the same time that there is valid data is not related to the execution of an instruction. Therefore, a valid data status will be output and the interrupt status will be saved for the next transmission.

The sequencing error encoding indicates that the inputs from the external development tool are not what the development port and/or the CPU was expecting. Two cases could cause this error:

1. The processor was trying to read instructions and there was data shifted into the development port, or
2. The processor was trying to read data and there was instruction shifted into the development port. The port will terminate the read cycle with a bus error.

This bus error will cause the CPU to signal that an interrupt (exception) occurred. Since a status of sequencing error has a higher priority than exception, the port will report the sequencing error first, and the CPU interrupt on the next transmission. The development port will ignore the command, instruction, or data shifted in while the sequencing error or CPU interrupt is shifted out. The next transmission after all error status is reported to the port should be a new instruction, trap enable or command (possibly the one that was in progress when the sequencing error occurred).

The interrupt-occurred encoding is used to indicate that the CPU encountered an interrupt during the execution of the previous instruction in debug mode. Interrupts may occur as the result of instruction execution (such as unimplemented opcode or arithmetic error), because of a memory access fault, or from an unmasked external interrupt. When an interrupt occurs the development port will ignore the command, instruction, or data shifted in while the interrupt encoding was shifting out. The next transmission to the port should be a new instruction, trap enable or debug port command.

Finally, the null encoding is used to indicate that no data has been transferred from the CPU to the development port shift register.

22.5.6.11 Fast Download Procedure

The download procedure is used to download a block of data from the debug tool into system memory. This procedure can be accomplished by repeating the following sequence of transactions from the development tool to the debug port for the number of data words to be down loaded:

```
INIT:    Save RX, RY
         RY <- Memory Block address- 4

         •••

repeat:  mfspr       RX, DPDR
         DATA word to be moved to memory
         stwu        RX, 0x4(RY)
until here


         •••

         Restore RX,RY
```

Figure 22-11 Download Procedure Code Example

For large blocks of data this sequence may take significant time to complete. The “fast download procedure” of the debug port may be used to reduce this time. This time reduction is achieved by eliminating the need to transfer the instructions in the loop to the debug port. The only transactions needed are those required to transfer the data to be placed in system memory. **Figure 22-12** and **Figure 22-13** illustrate the time benefit of the “fast download procedure”.

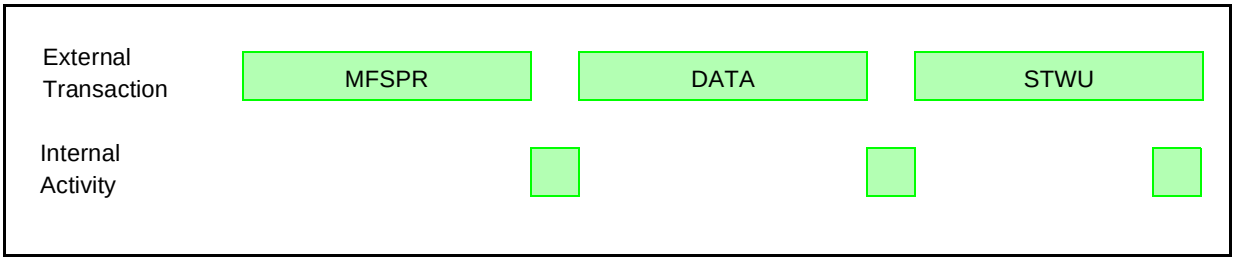


Figure 22-12 Slow Download Procedure Loop

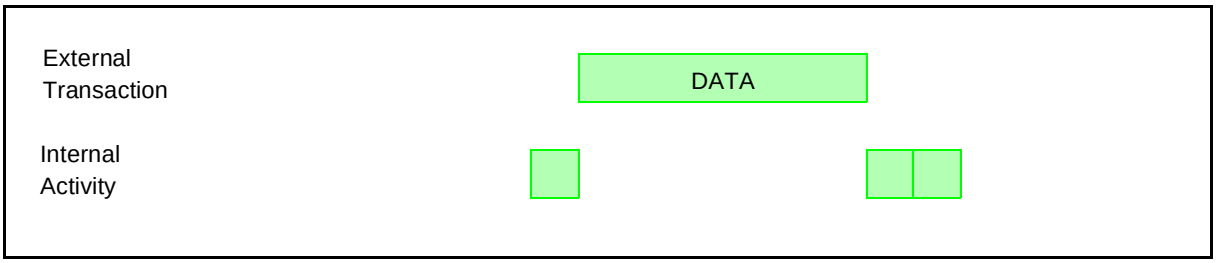


Figure 22-13 Fast Download Procedure Loop

The sequence of the instructions used in the “fast download procedure” is the one illustrated in **Figure 22-11** with RX = r31 and RY = r30. This sequence is repeated infinitely until the “end download procedure” command is issued to the debug port.

Note that, the internal general purpose register 31 is used for temporary storage data value. Before beginning the “fast download procedure” by the “start download procedure command”, The value of the first memory block address, – 4, must be written to the general purpose register 30.

To end a download procedure, an “end download procedure” command should be issued to the debug port, and then, additional DATA transaction should be sent by the development tool. This data word will NOT be placed into the system memory, but it is needed to stop the procedure gracefully.

22.6 Software Monitor Debugger Support

When in debug mode disable, a software monitor debugger can make use of all of the development support features defined in the CPU. When debug mode is disabled all events result in regular interrupt handling, i.e. the processor resumes execution in the corresponding interrupt handler. The exception cause register (ECR) and the debug enable register (DER) only influence the assertion and negation of the freeze signal.

22.6.1 Freeze Indication

The internal freeze signal is connected to all relevant internal modules. These modules can be programmed to stop all operations in response to the assertion of the freeze signal. In order to enable a software monitor debugger to broadcast the fact that the debug software is now executed, it is possible to assert and negate the internal freeze signal also when debug mode is disabled.

The assertion and negation of the freeze signal when in debug mode disable is controlled by the exception cause register (ECR) and the debug enable register (DER) as described in **Figure 22-6**. In order to assert the freeze signal the software needs to program the relevant bits in the debug enable register (DER). In order to negate the freeze line the software needs to read the exception cause register (ECR) in order to clear it and perform an **rfi** instruction.

If the exception cause register (ECR) is not cleared before the **rfi** is performed the freeze signal is not negated. Therefore it is possible to nest inside a software monitor debugger without affecting the value of the freeze line although **rfi** may be performed a few times. Only before the last **rfi** the software needs to clear the exception cause register (ECR).

The above mechanism enables the software to accurately control the assertion and the negation of the freeze signal.

22.7 Development Support Registers

Table 22-14 lists the registers used for development support. The registers are accessed with the **mtspr** and **mfspr** instructions.

Table 22-14 Development Support Programming Model

SPR Number (Decimal)	Name
144	Comparator A Value Register (CMPA) See **Table 22-17** for bit descriptions.
145	Comparator B Value Register (CMPB) See **Table 22-17** for bit descriptions.
146	Comparator C Value Register (CMPC) See **Table 22-17** for bit descriptions.
147	Comparator D Value Register (CMPD) See **Table 22-17** for bit descriptions.
148	Exception Cause Register (ECR) See **Table 22-27** for bit descriptions.
149	Debug Enable Register (DER) See **Table 22-28** for bit descriptions.
150	Breakpoint Counter A Value and Control Register (COUNTA) See **Table 22-25** for bit descriptions.
151	Breakpoint Counter B Value and Control Register (COUNTB) See **Table 22-26** for bit descriptions.
152	Comparator E Value Register (CMPE) See **Table 22-18** for bit descriptions.
153	Comparator F Value Register (CMPF) See **Table 22-18** for bit descriptions.
154	Comparator G Value Register (CMPG) See **Table 22-20** for bit descriptions.
155	Comparator H Value Register (CMPH) See **Table 22-20** for bit descriptions.
156	L-Bus Support Control Register 1 (LCTRL1) See **Table 22-23** for bit descriptions.
157	L-Bus Support Control Register 2 (LCTRL2) See **Table 22-24** for bit descriptions.
158	I-Bus Support Control Register (ICTRL) See **Table 22-21** for bit descriptions.
159	Breakpoint Address Register (BAR) See **Table 22-19** for bit descriptions.
630	Development Port Data Register (DPDR) See **22.7.13 Development Port Data Register (DPDR)** for bit descriptions.

22.7.1 Register Protection

Table 22-15 and **Table 22-16** summarize protection features of development support registers during read and write accesses, respectively.

Table 22-15 Development Support Registers Read Access Protection

MSR[PR]	Debug Mode Enable	In Debug Mode	Result
0	0	X	Read is performed. ECR is cleared when read. Reading DPDR yields indeterminate data.
0	1	0	Read is performed. ECR is *not* cleared when read. Reading DPDR yields indeterminate data.
0	1	1	Read is performed. ECR is cleared when read.
1	X	X	Program exception is generated. Read is not performed. ECR is *not* cleared when read.

Table 22-16 Development Support Registers Write Access Protection

MSR[PR]	Debug Mode Enable	In Debug Mode	Result
0	0	X	Write is performed. Write to ECR is ignored. Writing to DPDR is ignored.
0	1	0	Write is *not* performed. Writing to DPDR is ignored.
0	1	1	Write is performed. Write to ECR is ignored.
1	X	X	Write is *not* performed. Program exception is generated.

22.7.2 Comparator A–D Value Registers (CMPA–CMPD)

CMPA–CMPD — Comparator A–D Value Register **SPR 144 – SPR 147**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CMPAD															

RESET: UNAFFECTED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPAD														RESERVED	

RESET: UNAFFECTED

Table 22-17 CMPA-CMPD Bit Descriptions

Bits	Mnemonic	Description
0:29	CMPAD	Address bits to be compared
30:31	—	Reserved

These registers are unaffected by reset.

22.7.3 Comparator E–F Value Registers

CMPE–CMPF — Comparator E–F Value Registers **SPR 152, 153**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPEF																															

RESET: UNAFFECTED

Table 22-18 CMPE-CMPF Bit Descriptions

Bits	Mnemonic	Description
0:31	CMPV	Address bits to be compared

These registers are unaffected by reset.

22.7.4 Breakpoint Address Register (BAR)

BAR — Breakpoint Address Register **SPR 159**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPEF																															

RESET: UNAFFECTED

Table 22-19 BAR Bit Descriptions

Bits	Mnemonic	Description
0:31	BARV[0:31]	The address of the load/store cycle that generated the breakpoint

22.7.5 Comparator G–H Value Registers (CMPG–CMPH)

CMPG–CMPH — Comparator G–H Value Registers **SPR 154, 155**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CMPGH																															

RESET: UNAFFECTED

Table 22-20 CMPG-CMPH Bit Descriptions

Bits	Mnemonic	Description
0:31	CMPGH	Data bits to be compared

These registers are unaffected by reset.

22.7.6 I-Bus Support Control Register

ICTRL — I-Bus Support Control Register **SPR 158**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CTA			CTB			CTC			CTD			IWP0		IWP1	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
IWP2		IWP3		SIWP0 EN	SIWP1 EN	SIWP2 EN	SIWP3 EN	DIWP0 EN	DIWP 1 EN	DIWP 2 EN	DIWP 3 EN	IIFM	ISCT_SER*		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

*Changing the instruction show cycle programming starts to take effect only from the second instruction after the actual **mtspr** to ICTRL.

If the processor aborts a fetch of the target of a direct branch (due to an exception), the target is not always visible on the external pins. Program trace is not affected by this phenomenon.

Table 22-21 ICTRL Bit Descriptions

Bits	Mnemonic	Description	Function
0:2	CTA	Compare type of comparator A	0xx = not active (reset value) 100 = equal 101 = less than 110 = greater than 111 = not equal
3:5	CTB	Compare type of comparator B	
6:8	CTC	Compare type of comparator C	
9:11	CTD	Compare type of comparator D	
12:13	IWP0	I-bus 1st watchpoint programming	0x = not active (reset value) 10 = match from comparator A 11 = match from comparators (A&B)
14:15	W1	I-bus 2nd watchpoint programming	0x = not active (reset value) 10 = match from comparator B 11 = match from comparators (A | B)
16:17	IWP2	I-bus 3rd watchpoint programming	0x = not active (reset value) 10 = match from comparator C 11 = match from comparators (C&D)
18:19	IWP3	I-bus 4th watchpoint programming	0x = not active (reset value) 10 = match from comparator D 11 = match from comparators (C | D)
20	SIWP0EN	Software trap enable selection of the 1st I-bus watchpoint	0 = trap disabled (reset value) 1 = trap enabled
21	SIWP1EN	Software trap enable selection of the 2nd I-bus watchpoint	
22	SIWP2EN	Software trap enable selection of the 3rd I-bus watchpoint	
23	SIWP3EN	Software trap enable selection of the 4th I-bus watchpoint	

Table 22-21 ICTRL Bit Descriptions (Continued)

Bits	Mnemonic	Description	Function
24	DIWP0EN	Development port trap enable selection of the 1st I-bus watchpoint (read only bit)	0 = trap disabled (reset value) 1 = trap enabled
25	DIWP1EN	Development port trap enable selection of the 2nd I-bus watchpoint (read only bit)	
26	DIWP2EN	Development port trap enable selection of the 3rd I-bus watchpoint (read only bit)	
27	DIWP3EN	Development port trap enable selection of the 4th I-bus watchpoint (read only bit)	
28	IIFM	Ignore first match, only for I-bus breakpoints	0 = Do not ignore first match, used for “go to x” (reset value) 1 = Ignore first match (used for “continue”)
29:31	ISCT_SER	Instruction fetch show cycle and RCPU serialize control	These bits control serialization and instruction fetch show cycles. See **Table 22-5** for the bit definitions. **NOTE**: Changing the instruction show cycle programming starts to take effect only from the second instruction after the actual **mtspr** to ICTRL.

Table 22-22 ISCT_SER Bit Descriptions

Serialize (SER)	ISCTL	Functions Selected
0	00	RCPU is fully serialized and show cycle will be performed for all fetched instructions (reset value)[1]
0	01	RCPU is fully serialized and show cycle will be performed for all changes in the program flow
0	10	RCPU is fully serialized and show cycle will be performed for all indirect changes in the program flow
0	11	RCPU is fully serialized and no show cycles will be performed for fetched instructions
1	00	Illegal. This mode should not be selected.
1	01	RCPU is not serialized (normal mode) and show cycle will be performed for all changes in the program flow
1	10	RCPU is not serialized (normal mode) and show cycle will be performed for all indirect changes in the program flow
1	11	RCPU is not serialized (normal mode) and no show cycles will be performed for fetched instructions

NOTES:

1. In compression mode, the MPC566 can not perform a show cycles of all instructions. Only changes of flow or all indirect changes of flow are supported. If the ISCT_SER bits are set to 0b000, the RCPU will show changes of flow only (i.e., treat it as if ISCT_SER were set to 0b001).

22.7.7 L-Bus Support Control Register 1

LCTRL1 — L-Bus Support Control Register 1 **SPR 156**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CTE			CTF			CTG			CTH			CRWE		CRWF	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CSG		CSH		SUSG	SUSH	CGBMSK				CHBMSK				UNUSED	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-23 LCTRL1 Bit Descriptions

Bits	Mnemonic	Description	Function
0:2	CTE	Compare type, comparator E	0xx = not active (reset value) 100 = equal 101 = less than 110 = greater than 111 = not equal
3:5	CTF	Compare type, comparator F	
6:8	CTG	Compare type, comparator G	
9:11	CTH	Compare type, comparator H	
12:13	CRWE	Select match on read/write of comparator E	0X = don't care (reset value) 10 = match on read 11 = match on write
14:15	CRWF	Select match on read/write of comparator F	
16:17	CSG	Compare size, comparator G	00 = reserved 01 = word 10 = half word 11 = byte (Must be programmed to word for floating point compares)
18:19	CSH	Compare size, comparator H	
20	SUSG	Signed/unsigned operating mode for comparator G	0 = unsigned 1 = signed (Must be programmed to signed for floating point compares)
21	SUSH	Signed/unsigned operating mode for comparator H	
22:25	CGBMSK	Byte mask for 1st L-data comparator	0000 = all bytes are **not** masked 0001 = the last byte of the word is masked . . . 1111 = all bytes are masked
26:29	CHBMSK	Byte mask for 2nd L-data comparator	
30:31	—	Reserved	—

LCTRL1 is cleared following reset.

22.7.8 L-Bus Support Control Register 2

LCTRL2 — L-Bus Support Control Register 2 **SPR 157**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LW0EN	LW0IA		LW0 IADC	LW0LA		LW0 LADC	LW0LD		LW0 LDDC	LW1EN	LW1IA		LW1 IADC	LW1LA	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LW1 LADC	LW1LD		LW1 LDDC	BRK NOMSK	RESERVED							DLW0 EN	DLW1 EN	SLW0 EN	SLW1 EN
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-24 LCTRL2 Bit Descriptions

Bits	Mnemonic	Description	Function
0	LW0EN	1st L-bus watchpoint enable bit	0 = watchpoint not enabled (reset value) 1 = watchpoint enabled
1:2	LW0IA	1st L-bus watchpoint I-addr watchpoint selection	00 = first I-bus watchpoint 01 = second I-bus watchpoint 10 = third I-bus watchpoint 11 = fourth I-bus watchpoint
3	LW0IADC	1st L-bus watchpoint care/don't care I-addr events	0 = Don't care 1 = Care
4:5	LW0LA	1st L-bus watchpoint L-addr events selection	00 = match from comparator E 01 = match from comparator F 10 = match from comparators (E&F) 11 = match from comparators (E | F)
6	LW0LADC	1st L-bus watchpoint care/don't care L-addr events	0 = Don't care 1 = Care
7:8	LW0LD	1st L-bus watchpoint L-data events selection	00 = match from comparator G 01 = match from comparator H 10 = match from comparators (G&H) 11 = match from comparators (G | H)
9	LW0LDDC	1st L-bus watchpoint care/don't care L-data events	0 = Don't care 1 = Care
10	LW1EN	2nd L-bus watchpoint enable bit	0 = watchpoint not enabled (reset value) 1 = watchpoint enabled
11:12	LW1IA	2nd L-bus watchpoint I-addr watchpoint selection	00 = first I-bus watchpoint 01 = second I-bus watchpoint 10 = third I-bus watchpoint 11 = fourth I-bus watchpoint
13	LW1IADC	2nd L-bus watchpoint care/don't care I-addr events	0 = Don't care 1 = Care
14:15	LW1LA	2nd L-bus watchpoint L-addr events selection	00 = match from comparator E 01 = match from comparator F 10 = match from comparators (E&F) 11 = match from comparators (E | F)

Table 22-24 LCTRL2 Bit Descriptions (Continued)

Bits	Mnemonic	Description	Function
16	LW1LADC	2nd L-bus watchpoint care/don't care L-addr events	0 = Don't care 1 = Care
17:18	LW1LD	2nd L-bus watchpoint L-data events selection	00 = match from comparator G 01 = match from comparator H 10 = match from comparators (G&H) 11 = match from comparator (G | H)
19	LW1LDDC	2nd L-bus watchpoint care/don't care L-data events	0 = Don't care 1 = Care
20	BRKNOMSK	Internal breakpoints non-mask bit	0 = masked mode; breakpoints are recognized only when MSR[RI]=1 (reset value) 1 = non-masked mode; breakpoints are always recognized
21:27	—	Reserved	—
28	DLW0EN	Development port trap enable selection of the 1st L-bus watch-point (read only bit)	0 = trap disabled (reset value) 1 = trap enabled
29	DLW1EN	Development port trap enable selection of the 2nd L-bus watchpoint (read only bit)	
30	SLW0EN	Software trap enable selection of the 1st L-bus watchpoint	
31	SLW1EN	Software trap enable selection of the 2nd L-bus watchpoint	

LCTRL2 is cleared following reset.

For each watchpoint, three control register fields (LWxIA, LWxLA, LWxLD) must be programmed. For a watchpoint to be asserted, all three conditions must be detected.

22.7.9 Breakpoint Counter A Value and Control Register

COUNTA — Breakpoint Counter A Value and Control Register **SPR 150**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTV															

RESET: UNAFFECTED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED														CNTC	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-25 Breakpoint Counter A Value and Control Register (COUNTA)

Bit(s)	Name	Description
0:15	CNTV	Counter preset value
16:29	—	Reserved
30:31	CNTC	Counter source select 00 = not active (reset value) 01 = I-bus first watchpoint 10 =L-bus first watchpoint 11 = Reserved

COUNTA[16:31] are cleared following reset; COUNTA[0:15] are unaffected by reset.

22.7.10 Breakpoint Counter B Value and Control Register

COUNTB — Breakpoint Counter B Value and Control Register **SPR 151**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CNTV															

RESET: UNAFFECTED

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RESERVED														CNTC	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-26 Breakpoint Counter B Value and Control Register (COUNTB)

Bit(s)	Name	Description
0:15	CNTV	Counter preset value
16:29	—	Reserved
30:31	CNTC	Counter source select 00 = not active (reset value) 01 = I-bus second watchpoint 10 = L-bus second watchpoint 11 = Reserved

COUNTB[16:31] are cleared following reset; COUNTB[0:15] are unaffected by reset.

22.7.11 Exception Cause Register (ECR)

The ECR indicates the cause of entry into debug mode. All bits are set by the hardware and cleared when the register is read when debug mode is disabled, or if the processor is in debug mode. Attempts to write to this register are ignored. When the hardware sets a bit in this register, debug mode is entered only if debug mode is enabled and the corresponding mask bit in the DER is set.

All bits are cleared to zero following reset.

ECR — Exception Cause Register **SPR 148**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RST	CHSTP	MCE	RESERVED		EXTI	ALE	PRE	FPUVE	DECE	RESERVED		SYSE	TR	FPASE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	SEE	RE-SERVED	ITLBER	RE-SERVED	DTLBER	RESERVED						LBRK	IBRK	EBRKD	DPI
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 22-27 ECR Bit Descriptions

Bit(s)	Name	Description
0	—	Reserved
1	RST	Reset interrupt bit. This bit is set when the system reset pin is asserted.
2	CHSTP	Checkstop bit. Set when the processor enters checkstop state.
3	MCE	Machine check interrupt bit. Set when a machine check exception (other than one caused by a data storage or instruction storage error) is asserted.
4:5	—	Reserved
6	EXTI	External interrupt bit. Set when the external interrupt is asserted.
7	ALE	Alignment exception bit. Set when the alignment exception is asserted.
8	PRE	Program exception bit. Set when the program exception is asserted.
9	FPUVE	Floating point unavailable exception bit. Set when the program exception is asserted.
10	DECE	Decrementer exception bit. Set when the decrementer exception is asserted.
11:12	—	Reserved
13	SYSE	System call exception bit. Set when the system call exception is asserted.
14	TR	Trace exception bit. Set when in single-step mode or when in branch trace mode.
15	FPASE	Floating point assist exception bit. Set when the floating point assist exception occurs.
16	—	Reserved
17	SEE	Software emulation exception. Set when the software emulation exception is asserted.
18	—	Reserved
19	ITLBER	Implementation specific instruction protection error This bit is set as a result of an instruction protection error. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
20	—	Reserved
21	DTLBER	Implementation specific data protection error This bit is set as a result of an data protection error. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
22:27	—	Reserved
28	LBRK	L-bus breakpoint exception bit. This bit is set as a result of the assertion of a load/store break-point. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.

Table 22-27 ECR Bit Descriptions (Continued)

Bit(s)	Name	Description
29	IBRK	I-bus breakpoint exception bit. This bit is set as a result of the assertion of an Instruction breakpoint. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
30	EBRK	External breakpoint exception bit. Set when an external breakpoint is asserted (by an on-chip IMB or L-bus module, or by an external device or development system through the development port). This bit is set as a result of the assertion of an external breakpoint. Results in debug mode entry if debug mode is enabled and the corresponding enable bit is set.
31	DPI	Development port interrupt bit. Set by the development port as a result of a debug station non-maskable request or when debug mode is entered immediately out of reset.

22.7.12 Debug Enable Register (DER)

This register enables selectively masking the events that may cause the processor to enter into debug mode.

DER — Debug Enable Register **SPR 149**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RSTE	CHSTPE	MCEE	RESERVED		EXTIE	ALEE	PREE	FPU-VEE	DECEE	RESERVED		SYSEE	TRE	FPASE
RESET:															
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
RE-SERVED	SEEE	RE-SERVED	ITL-BERE	RE-SERVED	DTL-BERE	RESERVED						LBRKE	IBRKE	EBRKE	DPIE
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 22-28 DER Bit Descriptions

Bit(s)	Name	Description
0:1	—	Reserved
1	RSTE	Reset enable 0 = Debug entry is disabled (reset value) 1 = Debug entry is enabled
2	CHSTPE	Checkstop enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
3	MCEE	Machine check exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
4:5	—	Reserved
6	EXTIE	External interrupt enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled

Table 22-28 DER Bit Descriptions (Continued)

Bit(s)	Name	Description
7	ALEE	Alignment exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
8	PREE	Program exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
9	FPUVEE	Floating point unavailable exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
10	DECEE	Decrementer exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
11:12	—	Reserved
13	SYSEE	System call exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
14	TRE	Trace exception enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
15	FPASEE	Floating point assist exception enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
16	—	Reserved
17	SEEE	Software emulation exception enable bit 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
18	—	Reserved
19	ITLBERE	Implementation specific instruction protection error enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
20	—	Reserved
21	DTLBERE	Implementation specific data protection error enable bit. 0 = Debug mode entry disabled (reset value) 1 = Debug mode entry enabled
22:27	—	Reserved
28	LBRKE	Load/store breakpoint enable bit. 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
29	IBRKE	Instruction breakpoint interrupt enable bit. 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
30	EBRKE	External breakpoint interrupt enable bit (development port, internal or external modules). 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)
31	DPIE	Development port interrupt enable bit 0 = Debug mode entry disabled 1 = Debug mode entry enabled (reset value)

22.7.13 Development Port Data Register (DPDR)

This 32-bit special purpose register physically resides in the development port logic. It is used for data interchange between the core and the development system. An access to this register is initiated using **mtspr** and **mfspr** (SPR 630) and implemented using a special bus cycle on the internal bus.

SECTION 23
READI MODULE

23.1 Overview

The READI module provides real-time development capabilities for RCPU-based MCUs in compliance with the IEEE-ISTO 5001 - 1999. This module provides development support capabilities for MCUs in single chip mode, without requiring address and data pins for internal visibility.

The auxiliary port, along with RCPU development features (such as background debug mode and watchpoints) supports all software and hardware development in single chip mode. The auxiliary port, along with (on-chip) calibration RAM, allows calibration variable acquisition and calibration constant tuning in single chip mode, for automotive powertrain development systems.

NOTE

In this section the bit numbering in the register definitions of tool mapped registers follows the Nexus IEEE-ISTO 5001 - 1999 bit numbering convention of MSB = Bit 31 and LSB = Bit 0, unlike the PowerPC standard MSB = Bit 0 and LSB = Bit 31. The bit description tables list the PowerPC bit numbering and Nexus bit numbering.

23.1.1 General Description

The READI module interfaces to the RCPU processor and internal buses to provide development support as per the IEEE-ISTO 5001 - 1999. The development features supported are program trace, data trace, watchpoint trace, ownership trace, run-time access to the MCU's internal memory map, and access to RCPU internal registers during halt, via the auxiliary port.

23.1.2 Feature Summary List

The READI module is compliant with Class 3 of the IEEE-ISTO 5001 - 1999. The following features are implemented:

- Program trace via branch trace messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data trace via data write messaging (DWM) and data read messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources. Data trace also allows for calibration variable acquisition in automotive powertrain development systems.
  — Two data trace windows with programmable address range and access at-

tributes. Data trace windowing reduces the requirements on the auxiliary port bandwidth by constraining the number of trace locations.

- Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted to indicate when a new process/task is activated, allowing development tools to trace process/task flow.
- Run-time access to on-chip memory map and PowerPC™ special purpose registers (SPRs) via the READI read/write access protocol. This feature supports accesses for runtime internal visibility, calibration constant acquisition and tuning, and external rapid prototyping for powertrain automotive development systems.
- Watchpoint messaging via the auxiliary port
- Nine or 16 full-duplex auxiliary pin interface for medium and high visibility throughput
  - — One of two modes selected during reset: full port mode (FPM) and reduced port mode (RPM).
    - FPM comprises 16 pins and RPM comprises nine pins
  - — Auxiliary output port
    - One MCKO (message clock out) pin
    - Two or eight MDO (message data out) pins
    - One $\overline{\text{MSEO}}$ (message start/end out) pin
  - — Auxiliary input port
    - One $\overline{\text{MCKI}}$ (message clock in) pin
    - One or two MDI (message data in) pins
    - One $\overline{\text{MSEI}}$ (message start/end in) pin
    - One $\overline{\text{EVTI}}$ (event in) pin
    - One $\overline{\text{RSTI}}$ (reset in) pin
- All features configurable and controllable via the auxiliary port
- Security features for production environment
- Support of existing RCPU development access protocol via the auxiliary port
- READI module can be reset independent of system reset
- Parametrics:
  - — Two bits are downloaded per clock in full port mode. For example, with input clock running at 28 MHz, this translates to a download rate of 56 Mbits/s.
  - — One bit is downloaded per clock in reduced port mode. For example, with input clock running at 28 MHz, this translates to a download rate of 28 Mbits/s.
  - — Eight bits are uploaded per clock in full port mode. For example, with system clock running at 56 MHz, this translates to a upload rate of 448 Mbits/s.
  - — Two bits are uploaded per clock in reduced port mode. For example, with system clock running at 56 MHz, this translates to a upload rate of 112 Mbits/s.

23.1.3 Functional Block Diagram

The functional block diagram of the READI module is shown in **Figure 23-1**.

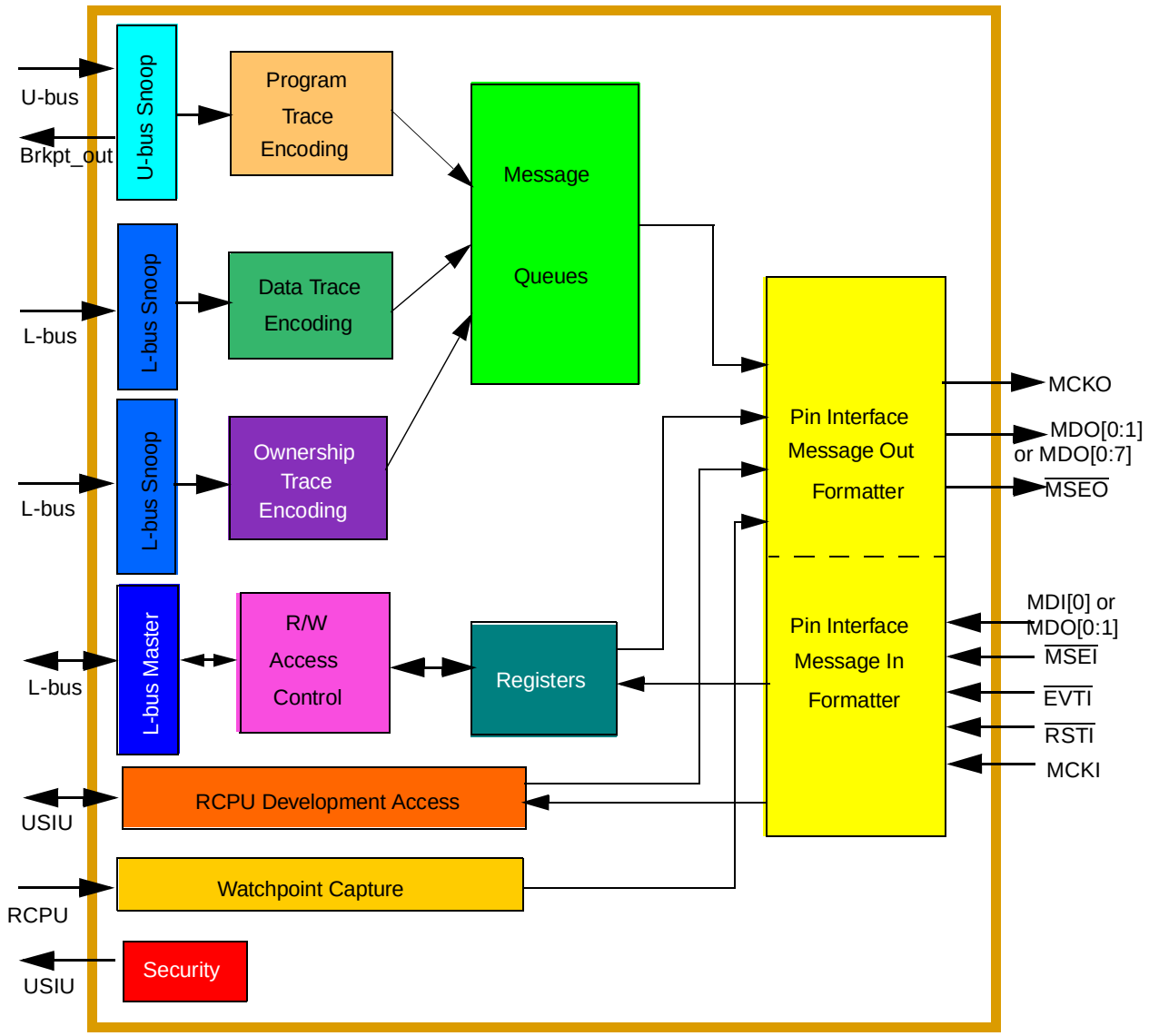


Figure 23-1 READI Functional Block Diagram

23.1.4 Modes of Operation

The various operating modes of the READI module are:

1. Reset
2. Secure
3. Normal
4. Disabled

23.1.4.1 READI Reset Configuration

The READI reset configuration information is received via $\overline{\text{EVTI}}$ and MDI[0] to enable or disable the READI module, and select the port size. $\overline{\text{EVTI}}$ and MDI[0] are sampled synchronously at the negation of $\overline{\text{RSTI}}$. Reset configuration information must be valid on $\overline{\text{EVTI}}$ and MDI[0] at least four clocks prior to the negation of $\overline{\text{RSTI}}$.

The following table describes the READI reset configuration options.

Table 23-1 READI Reset Configuration Options

$\overline{\text{EVTI}}$	MDI [0]	Configuration
1	X	Module Disabled. All outputs three-stated.
0	1	Module Enabled, Full Port Configuration 2 MDI, 8 MDO
0	0	Module Enabled, Reduced Port Configuration 1 MDI, 2 MDO

If $\overline{\text{EVTI}}$ is asserted at negation of $\overline{\text{RSTI}}$, the READI module will be enabled.

The READI module will three-state the auxiliary output port when $\overline{\text{RSTI}}$ is asserted.

23.1.4.2 Security

Security is provided via the UC3F censorship mechanism. If a UC3F array is in censored mode, reads or writes to the UC3F will not be allowed (RCPU will not be able to fetch instructions from the UC3F) once any of the following cases are detected:

- Program trace and/or data trace are enabled
- Read/write access is attempted (can be to any address location)
- RCPU development access is enabled.

23.1.4.3 Disabled

If $\overline{\text{EVTI}}$ is negated at negation of $\overline{\text{RSTI}}$, the READI module will be disabled. No trace output will be provided, and output auxiliary port will be three-stated. Any message sent by the tool is ignored.

23.1.5 Parametrics

With 16 message queues, throughput numbers were calculated for the following benchmark codes [assuming full port mode]:

- For an example benchmark which had 10.9% direct branches, 2.5% indirect branches, 10.4% data writes, and 19.3% data reads, approximately 20% of total data trace accesses will be traced.
- For another example benchmark which had 9.8% direct branches, 2.8% indirect branches, 6.6% data writes, and 18.3% data reads, approximately 27% of total data trace accesses will be traced.

For reduced port mode, the data trace feature should not be used, or used sparingly, so as not to cause queue overruns.

23.1.6 Programmer's Model

The READI registers do not follow the recommendations of the IEEE-ISTO 5001 - 1999, but are loosely based on the 0.9 release of the standard.

READI registers are classified into two categories: user mapped register and tool mapped registers.

User mapped register (memory mapped register):

- Ownership trace register

Tool mapped registers (registers which can be accessed only through the development tool and are not memory mapped):

- Device ID register
- Development control register
- User base address register
- Read/write access register
- Upload/download information register
- Data trace attributes register 1
- Data trace attributes register 2

23.1.7 Messages

The READI module implements messaging via the auxiliary port according to the IEEE-ISTO 5001 - 1999. Messaging will be implemented via transfer codes (TCODEs) on the auxiliary port. The TCODE number for the message identifies the transfer format (the number and/or size of packets to be transferred) and the purpose of each packet.

Public messages outlined in **Table 23-2** are supported by READI.

Table 23-2 Public Messages

TCODE Number	Message Name
1	Device ID
2	Ownership Trace Message
3	Program Trace — Direct Branch Message
4	Program Trace — Indirect Branch Message
5	Data Trace — Data Write Message
6	Data Trace — Data Read Message
8	Error Message
10	Program Trace Correction
11	Program Trace — Direct Branch Synchronization Message
12	Program Trace — Indirect Branch Synchronization Message

Table 23-2 Public Messages (Continued)

TCODE Number	Message Name
13	Data Trace — Data Write Synchronization Message
14	Data Trace — Data Read Synchronization Message
15	Watchpoint Message
16	Auxiliary Access — Device Ready for Upload/Download Message
17	Auxiliary Access — Upload Request (Tool Requests Information) Message
18	Auxiliary Access — Download Request (Tool Provides Information) Message
19	Auxiliary Access — Upload/Download Information (Device/Tool Provides Information) Message

Table 23-3 Vendor-Defined Messages

TCODE Number	Message Name
56	RCPU Development Access — DSDI Data (Tool Provides Information) Message
57	RCPU Development Access — DSDO Data (Device Provides Information) Message
58	RCPU Development Access — BDM Status (Device Provides Information) Message
59	Program Trace — Indirect Branch Message With Compressed Code
60	Program Trace — Direct Branch Synchronization Message With Compressed Code
61	Program Trace — Indirect Branch Synchronization Message With Compressed Code

Vendor-defined messages outlined in **Table 23-3** are also supported by READI.

23.1.8 Terms and Definitions

Table 23-4 Terms and Definitions

Term	Description
Auxiliary Port	Refers to IEEE-ISTO 5001 auxiliary port.
Branch Trace Messaging (BTM)	External visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
BDM	Background Debug Mode.
Compressed Code Mode	Current instruction stream is fetching compressed code.
Calibration Constants	Performance related constants which must be tuned for automotive powertrain and disk drive applications.
Calibration Variables	Intermediate calculations which must be visible during the calibration or tuning process to enable accurate tuning of calibration constants.
Data Read Message (DRM)	External visibility of data reads to internal memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to internal memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. May include DRM and/or DWM.
Download	Tool sends information to the device

Table 23-4 Terms and Definitions (Continued)

Term	Description
Field	Number of bits representing single piece of information
FPM	Full Port Mode. This is the default port mode for READI.
IEEE-ISTO 5001	IEEE-ISTO 5001, formerly known as Global Embedded Processor Debug Interface Standard. World Wide Web documentation at ***http://www.nexus-standard.org***.
Halt	RCPU is in freeze state (typically in debug mode)
Instruction Fetch	The process of reading the instruction data received from the instruction memory.
Instruction Issue	The process of driving valid instruction bits inside the processor. The instruction is decoded by each execution unit, and the appropriate execution unit prepares to execute the instruction during the next clock cycle.
Instruction Taken	An instruction is taken after it has been issued and recognized by the appropriate execution unit. All resources to perform the instruction are ready, and the processor begins to execute it.
Instruction Retire	Completion of the instruction issue, execution and writeback stages. An instruction is ready to be retired if it completes without generating an exception and all instructions ahead of it in history buffer have completed without generating an exception.
ICTRL	Instruction bus Support Control Register.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements e.g. DRM and DWM.
RCPU	PowerPC based processor used in the Motorola MPC500 family of microcontrollers.
READI	Real time Embedded Applications Development Interface.
READI pins	Refers to IEEE-ISTO 5001 auxiliary port.
RPM	Reduced Port Mode. This is the alternate port mode for READI.
run-time	RCPU is executing program code in normal mode
Sequential Instruction	Any instruction other than a flow-control instruction or **isync**.
Snooping	Monitoring addresses driven by a bus master to detect the need for coherency actions.
Standard	The phrase “according to the standard” implies according the IEEE-ISTO 5001 - 1999.
Superfield	One or more message “fields” delimited by $\overline{\text{MSEO}}$/$\overline{\text{MSEI}}$ assertion/negation. The information transmitted between “start-message” and “end-packet” states.
Show Cycle	An internal access (e.g., to an internal memory) reflected on the external bus using a special cycle (marked with a dedicated transfer code). For an internal memory “hit,” an address-only bus cycle is generated; for an internal memory “miss,” a complete bus cycle is generated.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
TCK_DSCK	Multiplexed pin: JTAG Clock or Development Port Clock.
TDI_DSDI	Multiplexed pin: JTAG Data In or Development Port Serial Data In.
TDO_DSDO	Multiplexed pin: JTAG Data Out or Development Port Serial Data Out.
Upload	Device sends information to the tool.
VSYNC	Internal RCPU signal
VF	Internal RCPU signal which indicates instruction queue status.
VFLS	Internal RCPU signal which indicates history buffer flush status.

23.2 Programmer's Model

This section describes the READI programmer's model. READI resources can only be configured and accessed via the auxiliary port.

The READI registers do not follow the recommendations of the IEEE-ISTO 5001 - 1999, but are loosely based on the 0.9 release of the standard.

23.2.1 Register Map

READI registers are accessible via the *auxiliary* port. They can be classified into two categories: user mapped registers and tool mapped registers.

23.2.1.1 User Mapped Register

The operating system writes the ID for the current task/process in the OTR register. **Table 23-5** shows the location of the register bits. Their functions are explained below.

The current task/process (CTP) field is updated by the operating system software to provide task/process ID information. The OT register can only be accessed by supervisor data attributes. Only CPU writes to this register will be transmitted out. This register is not accessible via the auxiliary port download request message.

READI_OTR — READI Ownership Trace Register **0x38 002C**

MSB 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Current Task Process (CTP)															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	LSB 31
Current Task Process (CTP)															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-5 READI_OTR Bit Descriptions

Bit(s)	Name	Description
31:0	CTP	READI ownership trace register, write only.

23.2.1.2 Tool Mapped Registers

Table 23-6 defines READI registers which are not memory mapped and can only be accessed through the development tool. Their corresponding access opcodes are also defined

Table 23-6 Tool Mapped Register Space

Access Opcode	Register	Access Type
0x08 (8)	Device ID Register (DID)	Read Only
0x0A (10)	Development Control Register (DC)	Read/Write
0x0D (13)	User Base Address Register (UBA)	Read Only
0x0F (15)	Read/Write Access Register (RWA)	Read/Write
0x10 (16)	Upload/Download Information Register (UDI)	Read/Write
0x14 (20)	Data Trace Attributes Register 1 (DTA1)	Read/Write
0x15 (21)	Data Trace Attributes Register 2 (DTA2)	Read/Write

23.2.1.3 Device ID (DID) Register

Accessing the DID register provides key attributes to the development tool concerning the MCU. This information is also transmitted via the auxiliary output port upon exit of READI reset ($\overline{\text{RSTI}}$), if $\overline{\text{EVTI}}$ is asserted at $\overline{\text{RSTI}}$ negation. **Table 23-7** gives the bit descriptions.

READI_DID — READI Device ID Register **0x08**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV				MDC						PN					
HRESET:															
—	—	—	—	0	0	0	0	1	0	0	0	0	0	1	1

LSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PN				MID											RE-SERVED
HRESET:															
0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1

Table 23-7 READI_DID Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0:3	31:28	REV	READI version number. This field contains the revision level of the device.
4:9	27:22	MDC	READI Manufacturer Design Center. This field identifies the manufacturer's design center. The MPC565 has a value of 0x02.
10:19	21:12	PN	READI Part Number. This part number identification field. The MPC565 is 0x033.
20:30	11:1	MID	READI Manufacturer ID. This field identifies the manufacturer of the device, Motorola's ID is 0x1C.
31	0	—	Reserved

23.2.1.4 Development Control (DC) Register

The DC register is used for basic development control of the READI module. **Table 23-8** shows the location of register bits.

READI_DC — READI Deveopment Control (DC) Register 0x0A

MSB 7	6	5	4	3	2	1	LSB 0
DOR	DME	DPA	TM			EC	
HARD RESET:							
0	0	0	0	0	0	0	0

Table 23-8 READI_DC Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0	7	DOR[1]	READI Debug Mode Entry Out-of-reset Field can be configured to enable or disable debug mode entry out of reset. 0 =Debug Mode Not Entered Out-of-Reset 1 = Debug Mode Entered Out-of-Reset
1	6	DME[1]	READI Debug Mode Enable Field can be configured to enable or disable debug mode. 0 = Debug Mode Disabled 1 = Debug Mode Enabled
2	5	DPA[1]	READI RCPU Development Access Field can be configured to access the RCPU Development features via the READI module or via the debug pins (TCK_DSCK, TDI_DSDI and TDO_DSDO). The default DPA setting configures the RCPU development features to be accessed via debug pins. 0 = RCPU Development Access Disabled through READI, BDM pins enabled 1 = RCPU Development Access Enabled through READI

Table 23-8 READI_DC Bit Descriptions (Continued)

RCPU Bit(s)	Nexus Bit(s)	Name	Description
3:5	4:2	TM	READI Trace Mode Field can be configured to enable BTM, DTM and OTM. Any or all types of trace may be enabled. 000 = No Trace 1xx = BTM Branch Trace Messaging Enabled x1x = DTM Data Trace Messaging Enabled xx1 = OTM Ownership Trace Messaging Enabled
6:7	1:0	EC	READI $\overline{\text{EVTI}}$ Control Field can be configured for synchronization and breakpoint generation. If the EC is equal to 0b00, asserting $\overline{\text{EVTI}}$ will cause the next program and data trace message to be a synchronization message (providing program and data trace are enabled). If the EC field is equal to 0b01, a breakpoint will be generated. If the field is configured to one of the reserved states, its action reverts to that of the default state. **NOTE**: The $\overline{\text{EVTI}}$ pin is level sensitive when EC is configured for breakpoint generation. This implies that as long as $\overline{\text{EVTI}}$ assertion is continued (with EC set to 0b01), the READI module will continue requesting a breakpoint. The user must detect breakpoint generation and negate the $\overline{\text{EVTI}}$ pin appropriately. 00 = $\overline{\text{EVTI}}$ for program and data trace synchronization 01 = $\overline{\text{EVTI}}$ for breakpoint generation 1x = Reserved

NOTES:

1. The DOR, DME, and DPA fields in the DC register can only be modified when system reset is asserted, or reset (to default state) when the READI module is reset by the assertion of $\overline{\text{RSTI}}$.

23.2.1.5 RCPU Development Access Modes

Table 23-9 describes the allowed modes for RCPU development access.

Table 23-9 RCPU Development Access Modes

DPA	DME	DOR	RCPU Development Access through READI
0	x	x	No RCPU Development Access via READI. RCPU access is done through BDM pins.
1	0	x	Non-debug mode access of RCPU development through READI.
1	1	0	Debug mode is enabled through READI (RCPU is still in normal mode, out of reset)
1	1	1	Debug mode is enabled through READI and entered out-of-reset. Debug mode entry causes RCPU to halt.

23.2.1.6 User Base Address (UBA) Register

The UBA register defines the memory map address for the OT register. **Table 23-10** gives a description of the register bits.

READI_UBA — READI User Base Address Register **0x0D**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UBA															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
UBA															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0

Table 23-10 READI_UBA Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0:31	31:0	UBA	The user base address (UBA) field defines the memory map address for the OT register. The MPC565 user base address is 0x38002C. The UBA register is read-only by the development tool.

23.2.1.7 Read/Write Access (RWA) Register

The RWA register provides DMA-like access to memory mapped locations, PowerPC special purpose registers, and READI tool mapped registers. **Table 23-11** shows the location of register bits.

READI_RWA — READI Read/Write Access Register **0x0F**

MSB 79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
SC	RWAD														
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
RWAD										RW	SZ		WD		
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
WD															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WD													PRV		MAP
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CNT															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-11 READI_RWA Read/Write Access Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0	79	SC	The start complete (SC) field is set when a read or write access is initiated. The device will clear the SC bit once the read or write access completes. During a block access, if the SC bit is reset, the access will terminate. 0 = Access complete 1 = Start access
1:25	78:54	RWAD	Read/write address (RWAD) bits are used to identify the address of internal memory-mapped resources to be accessed, or the lowest address (i.e., lowest unsigned value) for a block move (CNT > 0). The address range for a block move is from RWAD to RWAD + CNT. **NOTE:** The RWD field of the UDI register is shared with the WD field of the RWA register.
26	53	RW	The read/write (RW) field can be configured to allow selection of a read or a write access. 0 = Read access 1 = Write access
27:28	52:51	SZ	The word size (SZ) field can be configured to allow 32-bit, 16-bit, or 8-bit read/write accesses. If the field is configured to one of the reserved states, its action reverts to that of the default state. 00 = 32-bit 01 = 16-bit 10 = 8=bit 11 = Reserved
29:60	50:19	WD	Write data (WD) bits contain the data to be written. For a read access, the data stored is a don't care.
61:62	18:17	PRV	The Privilege Attribute Field can be configured to select different read/write access attributes. 00 = User Data 01 = User Instruction 10 = Supervisor Data 11 = Supervisor Instruction
63	16	MAP	The Map Select Field can be configured to allow access to multiple memory maps. The primary processor memory map (MAP equal to 0b0) is designated as the default. The secondary memory map (MAP equal to 0b1) can be set to select the PowerPC special purpose registers. 0 = Primary memory map 1 = Secondary memory map (PPC Special Purpose Registers)
64:79	15:0	CNT	The Access Count Field can be configured to indicate the number of accesses of word size (defined in SZ field). The CNT value is used to increment the specified address in the RWAD field for block read/write accesses. For a single read/write access, the CNT value should equal to 0x0000. A 64-Kbyte block read/write access can be performed by configuring the CNT bits as 0xFFFF. If a user wants to terminate a block read or write access which has not completed, the CNT bits should be reset.

23.2.1.8 Upload/Download Information (UDI) Register

The UDI register is used to store the data to be written for block write access, and the data read for read (single and block) accesses. **Table 23-12** gives a description of the register bits.

READI_UDI — READI Upload/Download Information Register **0x10**

MSB 33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
RWD																
HARD RESET:																
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RWD															ERR	DV
	HARD RESET:															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-12 READI_UDI Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0:31	33:2	RWD	The Read/Write Data Field is used to store data for read accesses and block write accesses. It can contain three sizes of data. Refer to **Table 23-13**, **Table 23-14** and **Table 23-15** for details.
32	1	ERR	The Error Field is used to determine the status of the read or write access. Refer to **Table 23-13** and **Table 23-14** for details. 0 = Read or write access has not been completed.. 1 = Read or write access has completed. **NOTE**: The ERR field is read-only.
33	0	DV	The Data Valid Field is used to determine the status of the read or write access. Refer to **Table 23-13** and **Table 23-14** for details. 0 = No error has occurred. 1 = Access error occurred. **NOTE**: The DV field is read-only.

Table 23-13 Read Access Status

ERR	DV	Status
0	0	Read access has not yet completed
0	1	Read access has completed and no access error occurred
1	0	Access error occurred
1	1	Not allowed

Table 23-14 Write Access Status

ERR	DV	Status
0	0	Write access has completed and no access error occurred
1	0	Write access error occurred (Error Message sent out)
0	1	Write access has not yet completed
1	1	Not allowed

Table 23-15 RWD Field Configuration

						LSB
8 bit	Reserved – Read as Zeros			LS Byte	ERR	DV
16 bit	Reserved – Read as Zeros		MS Byte	LS Byte	ERR	DV
32 bit	MS Byte			LS Byte	ERR	DV

NOTE

The RWD field of the UDI register is shared with the WD field of the RWA register.

23.2.1.9 Data Trace Attributes 1 and 2 (DTA1 and DTA2) Registers

The DTA1 and DTA2 registers allow data trace messaging (DTM) to be restricted to reads, writes or both for a user programmable address range. Two DTA registers allow two address ranges to be selected for DTM. Refer to **Table 23-16** for register bit descriptions.

READI_DTA 1 — READI Data Trace Attributes 1 Register **0x14**
READI_DTA 2 — READI Data Trace Attributes 2 Register **0x15**

MSB 47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
DTEA															
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTEA							DTSA								
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
DTSA														TA	
HARD RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 23-16 READI_DTA 1 AND 2 Bit Descriptions

RCPU Bit(s)	Nexus Bit(s)	Name	Description
0:22	47:25	DTEA[1]	The Read/Write End Field defines the end address for the address range. Refer to **Table 23-30**.
23:45	24:2	DTSA[1]	The Read/Write Start Field defines the starting address for the address range. Refer to **Table 23-30**.
46:47	1:0	TA	The Read/Write Trace Field can be configured to allow enabling or disabling data read and/or data write traces. 00 = Disable data read and data write trace x1 = Enable data read trace 1x = Enable data write trace

NOTES:
1. Data trace range start and end addresses must be word-aligned.

Table 23-17 Data Trace Values

Programmed Values	Range Selected
DTSA < DTEA	DTSA → ← DTEA
DTSA > DTEA	Invalid Range
DTSA = DTEA	Word at DTSA

NOTE

There is no way to distinguish between off-core PowerPC special purpose register (SPR) map and normal memory map accesses via the defined address range control. If data trace ranges are set up such that the off-core PowerPC SPR map falls within active ranges, then accesses to these off-core PowerPC SPRs will be traced, and the messages will not be distinguishable from accesses to normal memory map space. Off-core PowerPC SPRs typically exist in the 8- to 16-Kbyte lowest memory block (0x2000 – 0x3FF0). If data or peripherals are mapped to this space, load/stores to PowerPC SPRs will be indistinguishable from data or peripheral accesses.

23.2.2 Accessing Memory Mapped Locations Via the Auxiliary Port

The control and status information is accessed via the four auxiliary access public messages: device ready for upload/download, upload request (tool requests information), download request (tool provides information), and upload/download information (device/tool provides information).

To write control or status to memory mapped locations the following sequence would be required.

1. The tool confirms that the device is ready (so as to not cancel an ongoing read

write access). The tool transmits the download request public message (TCODE 18) which contains write attributes, write data, and target address.

2. The tool waits for device ready for upload/download (TCODE 16) message before initiating next access.

To read control or status from memory mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains read attributes and target address.
2. When device reads data it transmits upload/download information message (TCODE 19) containing read data. Device is now ready for next access.

For a block write to memory mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains block write attributes, first write data, and target address.
2. The tool waits for device ready for upload/download message (TCODE 16). When it is transmitted by device, tool transmits upload/download information message (TCODE 19) containing next write data. This step is repeated until all data is written

For a block read from memory mapped locations the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request public message (TCODE 18) which contains block read attributes and target address.
2. The tool waits for upload/download information message (TCODE 19) from device, which contains read data. This step is repeated until all data is read.

Refer to **23.6 Read/Write Access** for more details on read/write access protocol.

23.2.3 Accessing READI Tool Mapped Registers Via the Auxiliary Port

To write control or status data to READI tool mapped registers the following sequence would be required.

1. The tool confirms that the device is ready. The tool transmits the download request message (TCODE 18) which contains write data, and register opcode.
2. The tool waits for device ready for upload/download message (TCODE 16) before initiating next access.

To read control or status from READI tool mapped registers the following sequence would be required

1. The tool confirms that the device is ready. The tool transmits the upload request message (TCODE 17) which contains the target opcode.

2. When device reads data it transmits upload/download information message (TCODE 19) containing read data. Device is now ready for next access.

Refer to **23.6 Read/Write Access** for more details on read/write access protocol.

23.2.4 Partial Register Updates

Registers may be updated via the auxiliary port using the download request message with the message containing only **N** (where **N** is less than register width) most-significant bits of the register. In such cases the bits not transmitted will be reset to 0b0. The bits transmitted will be aligned such that the last bit transmitted will be the most significant bit of the register. Therefore a message size that is divisible by the input port size should be transmitted.

23.2.5 Programming Considerations

The following programing guidelines are recommended for users of the READI features.

23.2.5.1 Program Trace Guidelines

Program trace via BTM is not supported during BDM.

For program trace synchronization to work, the ICTRL (refer to **Table 23-21**) register must be programmed such that show cycle will be performed for all changes in the program flow (ISCTL field = 01).

NOTE

The user must program the ICTRL for change of flow show cycles early in the reset vector, before any branches, otherwise trace is not guaranteed.

If BDM is enabled, the ICTRL register cannot be modified through the program and can only be modified through RCPU development access.

It is also recommended that the USIU be programmed to ignore instruction show cycles (so as to not impact U-bus performance).

To correctly trace program execution using BTM, the READI module must be enabled prior to release of system reset. If the READI module is enabled ($\overline{\text{EVTI}}$ asserted, $\overline{\text{RSTI}}$ negated) after the RCPU has started execution of the program, the trace cannot be guaranteed.

Refer to **Figure 23-2** for further details.

23.2.5.2 Compressed Code Mode Guidelines

The ICTRL register must be programmed such that a show cycle will be performed for all changes in the program flow (ISCTL field = 01). (See **Table 22-21**.)

The BBC must be enabled to display data on instruction show cycles. BBCMCR[DECOMP_SC_EN] (refer to **4.8.2.1 BBC Module Configuration Register**

BBCMCR) must be set when decompression is enabled. This will allow READI to track the compressed code. BBCMCR[DECOMP_SC_EN] should not be set if there is no intention to use compressed code, as it will degrade U-bus performance.

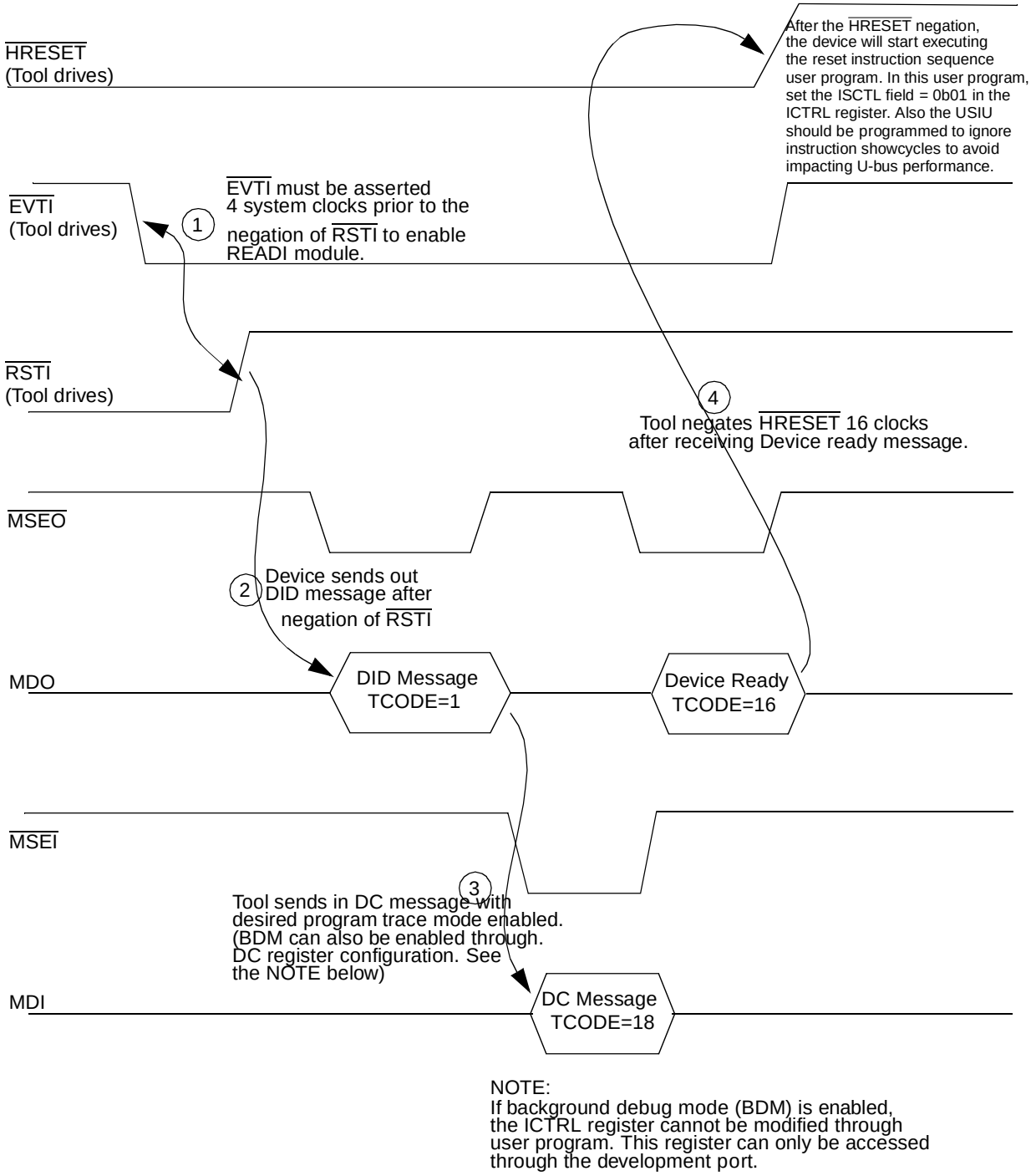


Figure 23-2 Enabling Program Trace out of System Reset

23.2.5.3 Reset Sequence Guidelines

Refer to **23.3.3 READI Reset Configuration** for further details.

23.2.5.4 BDM Guidelines

Refer to **23.3.3.1 Reset Configuration for Debug Mode** for further details.

23.3 Pin Interface

This section details information regarding the READI pins and pin protocol.

23.3.1 Functional Description

The READI pin interface provides the function of transmitting messages from the message queues to the external tools. The pin interface also provides the control for timing and logic for formatting the messages.

23.3.1.1 Pins Implemented

The READI module implements one MCKO, MCKI, $\overline{\text{EVTI}}$, $\overline{\text{RSTI}}$, $\overline{\text{MSEO}}$ and $\overline{\text{MSEI}}$ pin. It also implements 1 or 2 MDI and 2 or 8 MDO pins. The input pins are synchronized to MCKI input clock and the output pins are synchronized to free running MCKO output clock.The READI pin definition is outlined in **Table 23-18**.

NOTE

MCKI clock frequency has to be less than or equal to one half of MCKO clock frequency.

Table 23-18 Description of READI Pins

IEEE-ISTO 5001 Pin Name	Input/ Output	Description of Pin
MCKO	Output	Message Clock-Out (MCKO) is a free-running output clock to development tools for timing of MDO and $\overline{\text{MSEO}}$ pin functions. MCKO is the same as the MCU system clock.
MDO[7:0] or MDO[1:0]	Output	Message Data Out (MDO[7:0] or MDO[1:0]) are output pins used for uploading OTM, BTM, DTM, and Read/Write Accesses. External latching of MDO will occur on rising edge of MCKO. 8 pins are implemented.
$\overline{\text{MSEO}}$	Output	Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin which indicates when a message on the MDO pins has started, when a variable length packet has ended, and when the message has ended. 1 $\overline{\text{MSEO}}$ pin is implemented. External latching of $\overline{\text{MSEO}}$ will occur on rising edge of MCKO.
MCKI	Input	Message Clock-In (MCKI) is a input clock from development tools for timing of MDI and $\overline{\text{MSEI}}$ pin functions. MCKI frequency has to be less than or equal to one half of MCKO frequency.
MDI[1:0] or MDI[0]	Input	Message Data In (MDI[1:0] or MDI[0]) are input pins used for downloading configuration information, writes to user resources, etc. Internal latching of MDI will occur on rising edge of MCKI. 2 pins are implemented on the MPC565.

Table 23-18 Description of READI Pins (Continued)

IEEE-ISTO 5001 Pin Name	Input/ Output	Description of Pin
$\overline{\text{MSEI}}$	Input	Message Start/End In ($\overline{\text{MSEI}}$) is an input pin which indicates when a message on the MDI pins has started, when a variable length packet has ended, and when the message has ended. 1 $\overline{\text{MSEI}}$ pin is implemented. Internal latching of $\overline{\text{MSEI}}$ will occur on rising edge of MCKI.
$\overline{\text{EVTI}}$	Input	Event In ($\overline{\text{EVTI}}$) — The $\overline{\text{EVTI}}$ pin is level sensitive when configured for breakpoint generation, otherwise it is edge sensitive.
$\overline{\text{RSTI}}$	Input	Reset In ($\overline{\text{RSTI}}$).

23.3.2 Functional Block Diagram

Figure 23-3 depicts the functional block diagram of the pin interface.

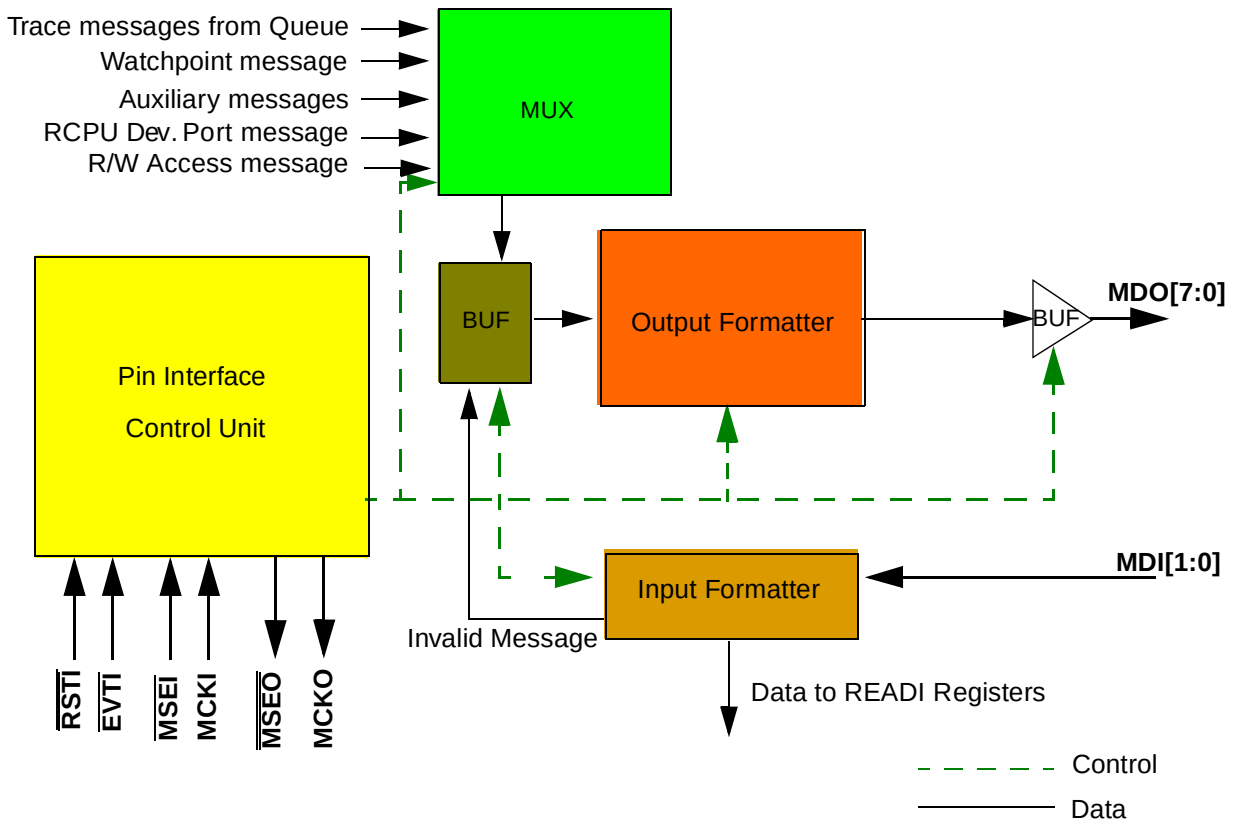


Figure 23-3 Functional Diagram of Pin Interface

The pin interface is responsible for handshaking with the message queue and registers. It is also responsible for requesting new messages from the message queue. A message is always requested from the message queue if the message queue is not empty, the message buffer is available and a higher priority message is not requesting to be transmitted. The rate at which data is removed from the queue depends on the average message length, the number of MDO pins, and the MCKO clocking rate.

23.3.2.1 Message Priority

Message formatting is performed in the pin interface block. The following priority scheme is implemented for messages sent to the pin output formatter block, with 1 being the highest priority and 5 being the lowest priority:

1. Invalid message
2. READI register access handshakes (device ready/download information)
3. Watchpoint messages
4. Read/write access message
5. RCPU development access message
6. Queued messages (program trace, data trace, and ownership trace)

23.3.2.2 Pin Protocol

The protocol for the MCU receiving and transmitting messages via the auxiliary pins will be accomplished with the $\overline{\text{MSEI}}$ and $\overline{\text{MSEO}}$ pin functions respectively. The $\overline{\text{MSEI}}$ pin will provide the protocol for the MCU receiving messages, and the $\overline{\text{MSEO}}$ pin will provide the protocol for the MCU transmitting messages.

The $\overline{\text{MSEI}}$/$\overline{\text{MSEO}}$ protocol is illustrated in **Table 23-19**.

Table 23-19 $\overline{\text{MSEI}}$/$\overline{\text{MSEO}}$ protocol

Operation	$\overline{\text{MSEO}}$/$\overline{\text{MSEI}}$ state
Idle	“1”s at all clocks
Start	Two “1”s followed by one “0”
Active	“0”s at all clocks during transmission of a message
End of variable length packet	“0” followed by “1”
End of packet and message	“0” followed by two or more “1”s

$\overline{\text{MSEI}}$/$\overline{\text{MSEO}}$ are used to signal the end of variable-length packets and messages. They are not required to indicate end of fixed length packets. $\overline{\text{MSEI}}$/$\overline{\text{MSEO}}$ are sampled on the rising edge of MCKI and MCKO respectively.

Fixed width fields can be concatenated before variable length fields without regard to the individual fields starting or ending at message ***N*** bit boundaries. Variable width fields must end at message ***N*** bit boundaries (where ***N*** is MDI/MDO pins).

Figure 23-4 shows the basic relation between the MDO and $\overline{\text{MSEO}}$ pins, and packet structure. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

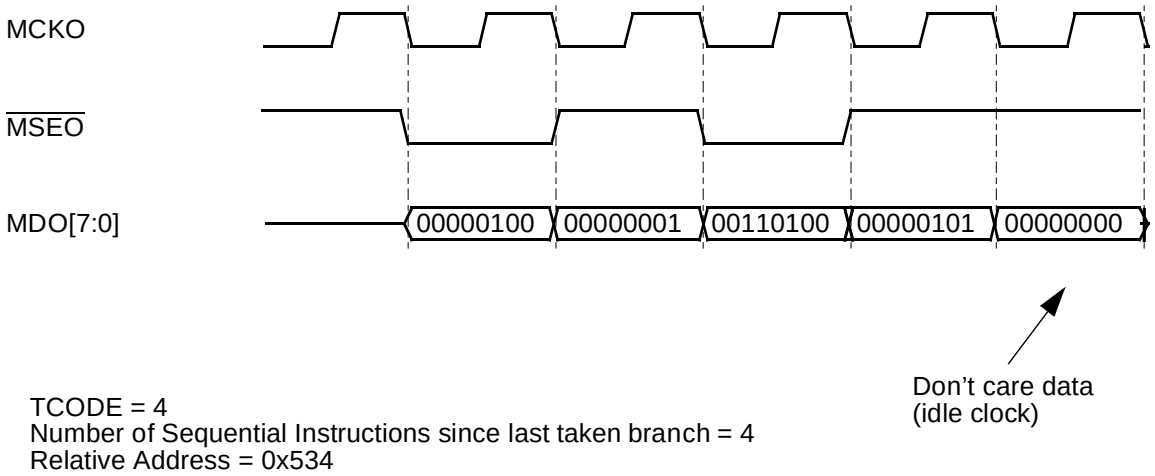


Figure 23-4 Auxiliary Pin Packet Structure for Program Trace Indirect Branch Message

Figure 23-5 illustrates the state diagram for $\overline{\text{MSEI}}$/$\overline{\text{MSEO}}$ transfers.

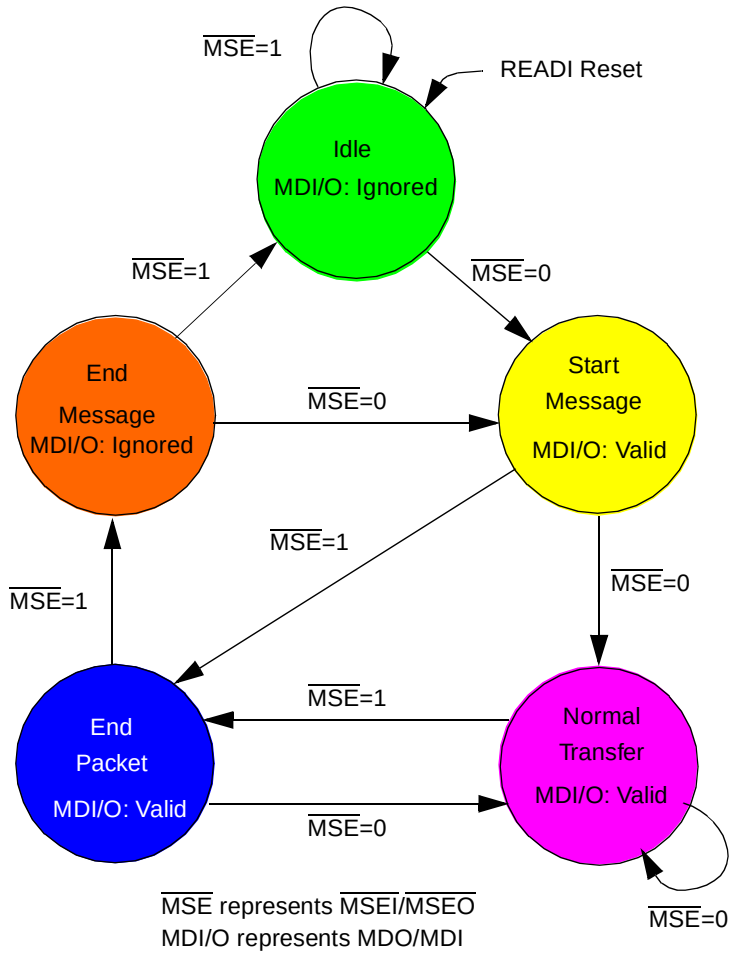


Figure 23-5 $\overline{\text{MSEI}}/\overline{\text{MSEO}}$ Transfers

NOTE

In the “End Message” state data on the MDI/MDO pins is ignored.

23.3.2.3 Messages

Public messages outlined in **Table 23-2** are supported by READI.

Table 23-20 Public Messages Supported

Message Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Device ID	6	6	Fixed	TCODE number = 1	From Device
	32	32	Fixed	Device ID information	
Ownership Trace Message	6	6	Fixed	TCODE number = 2	From Device
	32	32	Fixed	Task/Process ID tag	
Program Trace — Direct Branch Message	6	6	Fixed	TCODE number = 3	From Device
	1	8	Variable	number of sequential instructions executed since last taken branch	
Program Trace — Indirect Branch Message	6	6	Fixed	TCODE number = 4	From Device
	1	8	Variable	number of sequential instructions executed since last taken branch	
	1	23	Variable	unique portion of the target address for taken branches and exceptions	
Data Trace — Data Write Message	6	6	Fixed	TCODE number = 5	From Device
	1	25	Variable	unique portion of the data write address	
	8	32	Variable	data write value (8, 16, 32 bits)	
Data Trace — Data Read Message	6	6	Fixed	TCODE number = 6	From Device
	1	25	Variable	unique portion of the data read address	
	8	32	Variable	data read value (8, 16, 32 bits)	
Error Message	6	6	Fixed	TCODE number = 8	From Device
	5	5	Fixed	error code	
Program Trace Correction Message	6	6	Fixed	TCODE number = 10 (0xA)	From Device
	1	8	Variable	correcting the number of instructions in the trace	
Program Trace — Direct Branch Synchronization Message	6	6	Fixed	TCODE number = 11 (0xB)	From Device
	1	1	Variable	number of program trace messages cancelled	
	1	23	Variable	full target address	
Program Trace — Indirect Branch Synchronization Message	6	6	Fixed	TCODE number = 12 (0xC)	From Device
	1	1	Variable	number of program trace messages cancelled	
	1	23	Variable	full target address	
Data Trace — Data Write Synchronization Message	6	6	Fixed	TCODE number = 13 (0xD)	From Device
	1	1	Variable	number of messages canceled	
	1	25	Variable	full target address	
	8	32	Variable	data write value (8, 16, 32 bits)	

Table 23-20 Public Messages Supported (Continued)

Message Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Data Trace — Data Read Synchronization Message	6	6	Fixed	TCODE number = 14 (0xE)	From Device
	1	1	Variable	number of messages canceled	
	1	25	Variable	full target address	
	8	32	Variable	data read value (8, 16, 32 bits)	
Watchpoint Message	6	6	Fixed	TCODE number = 15 (0xF)	From Device
	6	6	Fixed	number indicating watchpoint source	
Auxiliary Access — Device Ready for Upload/Download Message	6	6	Fixed	TCODE number = 16 (0x10)	From Device
Auxiliary Access — Upload Request Message	6	6	Fixed	TCODE number = 17 (0x11)	From Tool
	8	8	Fixed	opcode to enable selected configuration, status or data upload from MCU	
Auxiliary Access — Download Request Message	6	6	Fixed	TCODE number = 18 (0x12)	From Tool
	8	8	Fixed	opcode to enable selected configuration or data download to MCU	
	8	80	Variable	Depending upon opcode selected for download, information to be downloaded to device will vary.	
Auxiliary Access — Upload/Download Information Message	6	6	Fixed	TCODE number = 19 (0x13)	From Device / Tool
	8	80	Variable	1). For an access, depending on word size selected (SZ field in RWA register), variable-length packets of information (10, 18, or 34bits) will be uploaded/downloaded from/to device. 2). Depending upon opcode selected for upload from internal READI registers, information to be uploaded to the device will vary.	

Table 23-21 describes the error code encoding for error messages.

Table 23-21 Error Codes

Error Code	Description
00000	Ownership trace overrun (Not Used)
00001	Program trace overrun (Not Used)
00010	Data trace overrun (Not Used)
00011	Read/write access error
00100	Invalid message
00101	Invalid Access Opcode
00110	Watchpoint overrun
00111	Program/Data/Ownership trace overrun
01000-10111	Reserved
11000-11111	Vendor Defined

Vendor-defined messages outlined in **Table 23-22** are also supported by READI.

Table 23-22 Vendor-Defined Messages Supported

TCODE Name	Minimum Packet Size (bits)	Maximum Packet Size (bits)	Packet Type	Packet Description	Direction
Dev Port Access — DSDI Data Message	6	6	Fixed	TCODE number = 56 (0x38)	From Tool
	10	35	Variable	BDM Development Serial Data In (DSDI)	
Dev Port Access — DSDO Data Message	6	6	Fixed	TCODE number = 57 (0x39)	From Device
	10	35	Variable	BDM Development Serial Data Out (DSDO)	
Dev Port Access — BDM Status Message	6	6	Fixed	TCODE number = 58 (0x3A)	From Device
	1	1	Fixed	BDM status	
Program Trace — Indirect Branch Message With Compressed Code	6	6	Fixed	TCODE number = 59 (0x3B)	From Device
	1	8	Variable	Number of sequential instructions executed since last taken branch	
	6	6	Fixed	Bit address	
	1	23	Variable	Unique portion of the target address for taken branches and exceptions (compressed code)	
Program Trace — Direct Branch Synchronization Message With Compressed Code	6	6	Fixed	TCODE number = 60 (0x38C	From Device
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	
Program Trace — Indirect Branch Synchronization Message With Compressed Code	6	6	Fixed	TCODE number = 61 (0x3D)	From Device
	6	6	Fixed	Bit address	
	1	23	Variable	Current instruction address	

23.3.2.4 Message Formats

Message formatting is performed in the pin interface block. Raw messages read from the message queue are independent of the number of MDO pins implemented.

Table 23-23 shows the various message formats that the pin interface formatter has to encounter.

NOTE

For variable length fields, the transmitted size of the field is determined as the bits from the least significant bit to the most significant non-zero valued bit, (i.e., most significant 0 value bits are not transmitted).

Table 23-23 Message Field Sizes

Message	TCODE	Field # 1	Field # 2	Field # 3	Max Size[1]	Min. Size[2]
Device ID	1	Fixed = 32	NA	NA	38 bits	38 bits
Ownership Trace Message	2	Fixed = 32	NA	NA	38 bits	38 bits
Program Trace — Direct Branch Message	3	Variable Max = 8 Min = 1	NA	NA	14 bits	7 bits
Program Trace — Indirect Branch Message	4	Variable Max = 8 Min = 1	Variable Max = 23 Min = 1	NA	37 bits	8 bits
Data Trace — Data Write Message	5	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	NA	63 bits	15 bits
Data Trace — Data Read Message	6	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	NA	63 bits	15 bits
Error Message	8	Fixed = 5	NA	NA	11 bits	11 bits
Program Trace Correction Message	10	Variable Max = 8 Min = 1	NA	NA	14 bits	7 bits
Program Trace — Direct Branch Synchronization Message	11	Variable Max = 1 Min = 1	Variable Max = 23 Min = 1	NA	30 bits	8 bits
Program Trace — Indirect Branch Synchronization Message	12	Variable Max = 1 Min = 1	Variable Max = 23 Min = 1	NA	30 bits	8 bits
Data Trace — Data Write Synchronization Message	13	Variable Max = 1 Min = 1	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	64 bits	16 bits
Data Trace — Data Read Synchronization Message	14	Variable Max = 1 Min = 1	Variable Max = 25 Min = 1	Variable Max = 32 Min = 8	64 bits	16 bits

Table 23-23 Message Field Sizes

Message	TCODE	Field # 1	Field # 2	Field # 3	Max Size[1]	Min. Size[2]
Watchpoint Message	15	Fixed = 6	NA	NA	12 bits	12 bits
Auxiliary Access — Device Ready for Upload/Download Message	16	NA	NA	NA	6 bits	6 bits
Auxiliary Access — Upload Request (Tool requests information) Message	17	Fixed = 8	NA	NA	14 bits	14 bits
Auxiliary Access — Download Request (Tool provides Information) Message	18	Fixed = 8	Variable Max = 80 Min = 8	NA	94 bits	22 bits
Auxiliary Access — Upload/Download Information (Device/ Tool provides Information) Message	19	Variable Max = 80 Min = 8	NA	NA	86 bits	14 bits
Dev Port Access — DSDI Data (Tool Provides Information) Message	56	Variable Max = 35 Min = 10	NA	NA	41 bits	16 bits
Dev Port Access — DSDO Data (Device Provides Information) Message	57	Variable Max = 35 Min = 10	NA	NA	41 bits	16 bits
Dev Port Access — BDM Status (Device Provides Information) Message	58	Fixed = 1	NA	NA	7 bits	7 bits
Program Trace — Indirect Branch Message With Compressed Code	59	Variable Max = 8 Min = 1	Fixed = 6	NA	43 bits	14 bits
Program Trace — Direct Branch Synchronization Message With Compressed Code	60	Fixed = 6	Variable Max = 23 Min = 1	NA	35 bits	13 bits
Program Trace— Indirect Branch Synchronization Message With Compressed Code	61	Fixed = 6	Variable Max = 23 Min = 1	NA	35 bits	13 bits

NOTES:

1. Maximum information size. The actual number of bits transmitted is dependant on the number of MDO pins.
2. Minimum information size. The actual number of bits transmitted is dependent on the number of MDO pins.

The maximum message length is 94 bits. The maximum number of fields is 3, excluding the TCODE itself.

The double edges in **Table 23-23** indicate the required $\overline{\text{MSEO}}$/$\overline{\text{MSEI}}$ assertion or negation indicating the start of message or end of message respectively.

The shaded edges in **Table 23-23** indicate how information can be packed into super-fields delimited via $\overline{\text{MSEO}}$/$\overline{\text{MSEI}}$ assertion followed by $\overline{\text{MSEO}}$/$\overline{\text{MSEI}}$ negation.

23.3.2.5 Rules of Messages

- A variable sized field within a message must end on a port boundary.
- A variable sized field may start within a port boundary only when following a fixed length packet.
- Super-fields must end on a port boundary (2-, 4-, or 8-bit boundaries depending on whether the device receives or sends messages, and the port size configured).
- When a variable length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.
- A data field within a data trace message must be eight, 16, or 32 bits in length.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information. Within a field, the lowest significant bits are shifted out first. **Figure 23-6** shows the transmission sequence of a message which is made up of a TCODE, a variable length field, and finally a super-field consisting of a fixed length and a variable length field. The only exception to this rule is the dev port access messages. See **23.10.1 RCPU Development Access Messaging** for further details.

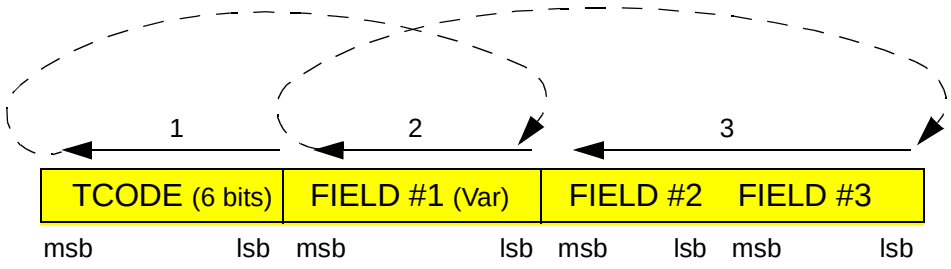


Figure 23-6 Transmission Sequence of messages.

23.3.2.6 Examples

The following are examples of branch trace messages.

Table 23-24 illustrates an example of how the indirect branch public message is transmitted.

NOTE 1

The example uses a four-bit output port. Note also that T0, I0, and A0 are the least significant bits where:

Tx = TCODE number (fixed)
Ix = Number of sequential instructions (variable)
Ax = Unique portion of the address (variable)

NOTE 2

During clock 7, the tool should ignore data on MDO pins.

Table 23-24 Indirect Branch Message

Clock	MDO[3:0] 3	2	1	0	$\overline{\text{MSEO}}$ 1	Idle
0	X	X	X	X	1	Idle (or end of last message)
1	T3	T2	T1	T0	0	Start Message
2	I1	I0	T5	T4	0	Normal Transfer
3	I5	I4	I3	I2	0	Normal Transfer
4	0	0	I7	I6	1	End Packet
5	A3	A2	A1	A0	0	Normal Transfer
6	A7	A6	A5	A4	1	End Packet
7	0	0	0	0	1	End Message
8	T3	T2	T1	T0	0	Start Message

Table 23-24 is an example of the minimum transmission of any message containing a variable length field (three clocks).

NOTE 1

The example uses a 4-bit output port. Note also that T0, and I0 are the least significant bits where:

Tx = TCODE number (fixed)
Ix = Number of sequential instructions (variable)

NOTE 2

During clock 3, the tool should ignore data on MDO pins.

Table 23-25 Direct Branch Message

Clock	MDO[3:0] 3	2	1	0	$\overline{\text{MSEO}}$ 1	Idle
1	T3	T2	T1	T0	0	Start Message
2	I1	I0	T5	T4	1	End Packet
3	0	0	0	0	1	End Message

23.3.2.7 Non-Temporal Ordering of Transmitted Messages

Trace messages sent out may not be in the sequence they actually occurred. The traces are monitored on the internal buses and these traces are captured as they occur and are sent out in the order they were captured and processed.

BTMs are in sequence and DTMs are in sequence, however, temporal order of DTMs interleaved with BTMs may not be accurate with regard to logical flow of code.

23.3.3 READI Reset Configuration

The READI reset configuration information is received via $\overline{\text{EVTI}}$ and MDI[0] to enable or disable the READI module and select the port size. $\overline{\text{EVTI}}$ and MDI[0] are sampled synchronously at the negation of $\overline{\text{RSTI}}$. Reset configuration information must be valid on $\overline{\text{EVTI}}$ and MDI[0] at least four clocks prior to the negation of $\overline{\text{RSTI}}$.

If $\overline{\text{EVTI}}$ is sampled asserted at negation of $\overline{\text{RSTI}}$, the READI module will be enabled. This is illustrated in **Figure 23-7**.

READI control and status information will be reset and the auxiliary output port will be three-stated, when $\overline{\text{RSTI}}$ is asserted. System reset will not reset the READI control and status information and not three-state the auxiliary output port.

Port size configuration is selected via the value of MDI[0] at the negation of $\overline{\text{RSTI}}$. **Table 23-26** describes the READI reset configuration options.

Table 23-26 READI Reset Configuration Options

$\overline{\text{EVTI}}$	MDI [0]	Configuration
1	X	Module Disabled. All outputs three-stated.
0	1	Module Enabled, Default Port Configuration 2 MDI, 8 MDO
0	0	Module Enabled, Alternate Port Configuration 1 MDI, 2 MDO

$\overline{\text{RSTI}}$ has a pull-down resistor in the pads. If the auxiliary port is not connected to a tool, READI module will be in reset state and not drive the auxiliary output port.

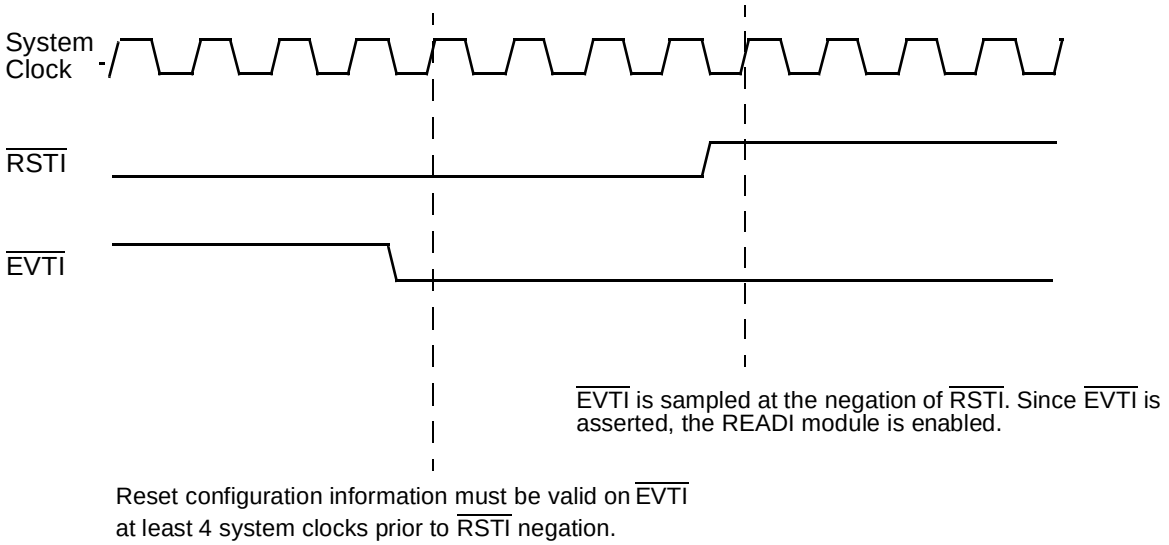


Figure 23-7 READI Module Enabled

23.3.3.1 Reset Configuration for Debug Mode

To enable RCPU development access via the READI pins, the reset sequence outlined below should be used:

- Assert READI reset ($\overline{\text{RSTI}}$), event-in ($\overline{\text{EVTI}}$) and system reset ($\overline{\text{HRESET}}$)
- Negate $\overline{\text{RSTI}}$
- Upon negation of $\overline{\text{RSTI}}$, tool should configure the DOR, DME, and DPA fields in the DC register to desired setting.
- Tool negates $\overline{\text{HRESET}}$ at least 16 system clocks after receiving the device ready message

Refer to **Figure 23-68** for further details.

23.3.3.2 Reset Configuration for Non-Debug Mode

Refer to **23.3.3.1 Reset Configuration for Debug Mode** for details on reset configuration for non-debug mode. The only difference between non-debug mode reset configuration and debug mode reset configuration is the values of the DOR, DME, and DPA fields in the DC register.

23.3.3.3 Secure Mode

Refer to **23.1.4.2 Security** for further details.

23.3.3.4 Disabled Mode

If $\overline{\text{EVTI}}$ is negated at negation of $\overline{\text{RSTI}}$, the READI module will be disabled. No trace output will be provided, and output auxiliary pins will be three-stated. This is illustrated in **Figure 23-8**.

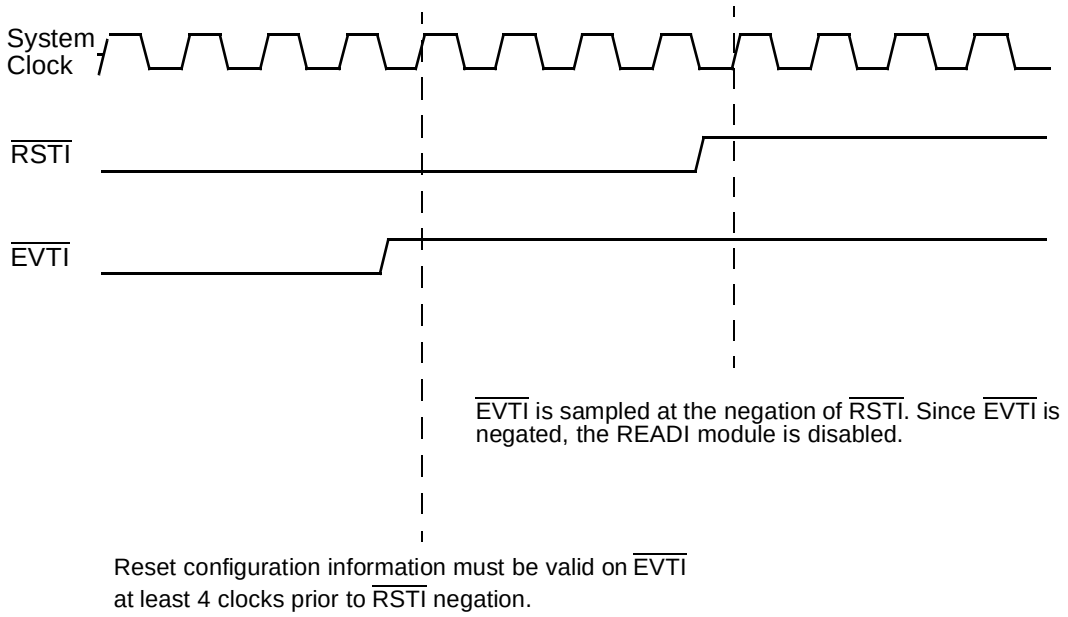


Figure 23-8 READI Module Disabled

23.3.3.5 Guidelines For Transmitting Input Messages

- An error message is sent out when an invalid TCODEis detected by the pin input formatter. Refer to **23.6.8.2 Invalid Message** for further details.
- An error message is sent out when an invalid access opcode is detected in auxiliary input messages by the pin input formatter. Refer to **23.6.8.3 Invalid Access Opcode** for further details.
- If the TCODE is valid, then READI will expect that the correct number of packets have been received and no further checking will be performed. If the number of packets received by READI is not correct, READI response is not defined, unless the message is a download request message (refer to **23.2.4 Partial Register Updates** for further details).

23.4 Program Trace

This section details the program trace mechanism supported by READI for the RCPU. Program trace is implemented via branch trace messaging (BTM) as per the IEEE-ISTO 5001 - 1999 definition.

23.4.1 Branch Trace Messaging

Branch trace messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception. Direct (or indirect) branches not taken are counted as sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many se-

quential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address.

- For some mispredicted branches and exception occurrences, program trace correction messages correct the number of instructions since last taken branch as transmitted in prior BTM message.

23.4.1.1 RCPU Instructions that Cause BTM Messages

The following RCPU instructions, when executed, cause indirect branch messages to be encoded:

1. Taken branch relative to link or counter registers
2. Context switching sequential Instructions
3. Exception taken (error/interrupts)

The following RCPU instruction, when executed, causes direct branch messages to be encoded:

4. Taken direct branch instructions

23.4.2 Review of RCPU Instruction Execution

Branch trace messaging for the RCPU is accomplished by snooping the U-bus address, data and program trace signals, and RCPU VF and VFLS signals, and performing algorithms necessary to generate the trace messages.

23.4.2.1 RCPU Pipeline and Execution Model

The RCPU instruction sequencer provides the central control for data flow between execution units and data files. It also implements the instruction pipeline to fetches and issues instructions to the execution units, and to control the pipeline. **Figure 23-9** depicts the RCPU instruction flow.

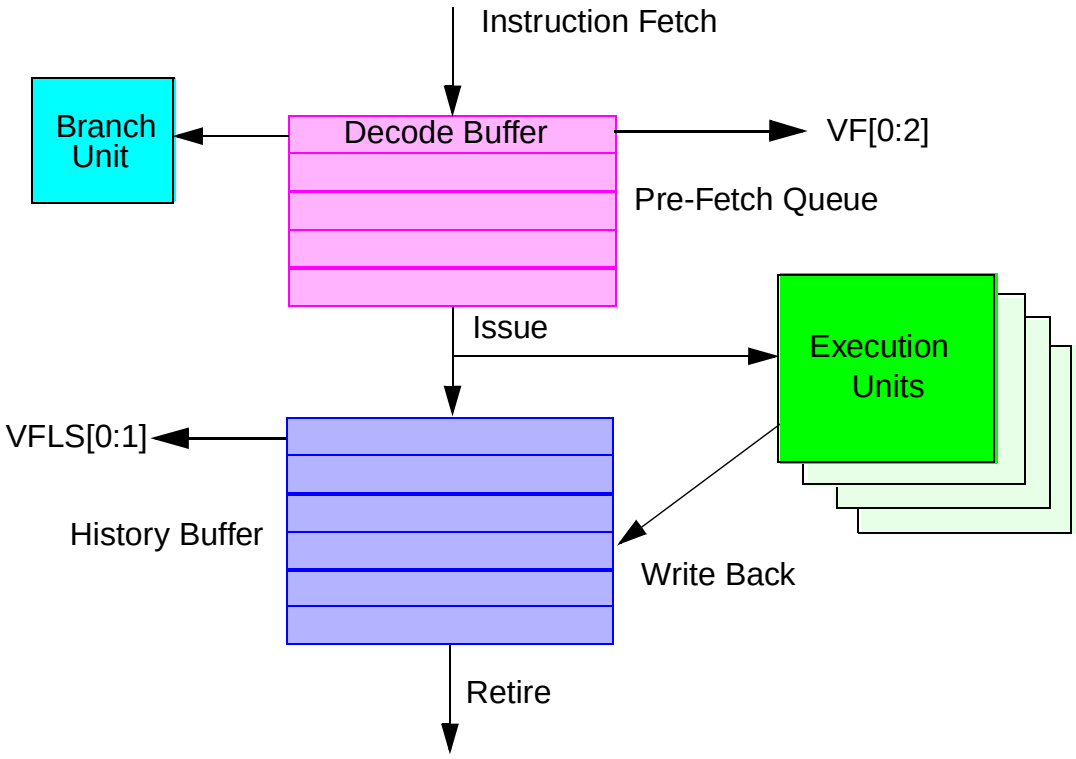


Figure 23-9 RCPU Instruction Flow

A generalized execution flow for the sequencer is as follows. For more details refer to the ***RCPU Reference Manual*** **(RCPURM/AD)**.

1. Fetch instructions into the prefetch queue [4+1 instructions deep].
2. Decode instructions fetched into the prefetch queue. Signal instruction type based on the decode through the VF signals.
   — If the decoded branch instruction is unconditional, instructions subsequent to the branch instruction in the prefetch queue are flushed. Fetching of instructions is started from the new target address.
   — If the decoded branch instruction is conditional and predicted not taken, the branch instruction is treated as a sequential instruction. Instructions are not flushed from the prefetch queue.
   — If the decoded branch instruction is conditional and predicted taken, instructions subsequent to the branch instruction in the prefetch queue are flushed. Fetching of instructions is started from the new target address.
3. Issue decoded instructions at the top of the prefetch queue to the execution units, and load processor status into history buffer [six instructions deep].
4. If the conditional branch instruction which was predicted taken, is subsequently discovered to be mispredicted (condition not met), fetched instructions (from the new target address specified by the branch instruction) are flushed. The

cancelled branch and the number of flushed instructions are signalled through the VF signals. Now fetching of instructions is started from the old address (before the branch was predicted as taken).

5. An instruction is ‘retired’ when the execution unit completes execution (without an exception) of the issued instruction (write-back completed) and the instruction is at the top of the history buffer queue; (i.e., all previously issued instructions completed without exception).
   — Issued instructions may complete execution out of order, but are always retired from the history buffer in order.
6. When an instruction-caused exception is recognized, instructions that appear earlier in the instruction stream are required to complete before the exception is taken. An instruction is said to have “completed” when the results of that instruction’s execution has been committed to the appropriate registers (i.e., following the writeback stage).
   — Exception conditions may be recognized out of order, but they are handled strictly in order.
7. Instructions that appear after the exception-causing instruction which have not been executed are flushed from the prefetch queue. The number of instructions flushed is signalled via the VF signals. Instructions that appear after the exception-causing instruction which have been executed are flushed from the history buffer. The number of instructions flushed from the history buffer are signaled via the VFLS signals.
8. When exceptions occur, information about the state of the processor is saved to certain registers, and the processor begins execution at an address predetermined for each exception.

As can be seen from the above flow, the following issues arise, when tracking the execution of the RCPU:

1. Not all fetched (decoded) instructions are executed.
2. Not all executed (issued and completed) instructions are retired.
3. Taken branches may be cancelled due to misprediction or exceptions.
4. Executed instructions (sequential or branches) which are subsequent to the exception-causing instruction will be flushed from the history buffer due to exception.

The trace information that is generated by RCPU is ideally suited for off-line software oriented processing, where there is the ability to trace back and recover from flushed queues.

23.4.2.2 RCPU Branch Trace Indicators

The RCPU supports program trace by providing information on the status of the instruction decode, instruction queue flush and history buffer flush. For more information, consult the section on development support in the ***RCPU Reference Manual* (RCPURM/AD)**. The branch trace indicators from the RCPU are the VF and the VFLS signals.

The VF signals indicate the type for the current instruction decoded and the outcomes of change-of-flow instructions. They also indicate the number of instructions flushed from the instruction prefetch queue on change of flow. This indication is in the clock following the indication of change of flow. For details on VF instruction type encoding and VF queue flush Information refer to **Table 23-27** and **Table 23-28** respectively.

Table 23-27 VF Instruction Type Encoding

VF	Instruction Type	Next VF Encoding
000	None	Instruction Type
001	Sequential	
010	Branch (direct or indirect) NOT taken	
011	VSYNC was asserted/negated and therefore the next instruction will be marked with program trace cycle attribute	
100	Interrupt/Exception taken – the target instruction fetch will be marked with program trace cycle attribute.	Queue Flush Information
101	Branch Indirect taken, ***rfi***, ***mtmsr***, ***isync*** and in some cases ***mtspr*** – the target instruction fetch will be marked with program trace cycle attribute.	
110	Branch direct taken	
111	Branch (direct or indirect) NOT taken (misprediction)	

The RCPU VF signals indicate (instruction pre-fetch) queue flush information in the clock following a taken change-of-flow indication. This encoding is as follows.

Table 23-28 VF Queue Flush Information

VF	Instruction Type	Next VF Encoding
000	0 Instructions flushed from instruction queue	Instruction Type
001	1 Instructions flushed from instruction queue	
010	2 Instructions flushed from instruction queue	
011	3 Instructions flushed from instruction queue	
100	4 Instructions flushed from instruction queue	
101	5 Instructions flushed from instruction queue	
110	Reserved	
111	Branch (direct or indirect) NOT taken (mis-prediction)	Queue Flush Information

NOTE

Encoding VF equal to 0b111 for the queue flush information indicates that the predicted branch (indicated in the previous clock by appropriate VF encoding) is cancelled, and that further flush information follows in the next clock.

The VFLS signals track the history buffer flushes. This aids the READI module to determine how many instructions actually executed and retired. Refer to **Table 23-29** for details on VFLS encoding.

Table 23-29 VFLS History Buffer Flush Encoding

VFLS	History Buffer Flush Information
00	0 Instructions flushed from History Buffer
01	1 Instructions flushed from History Buffer
10	2 Instructions flushed from History Buffer
11	Ignored for Program Trace

NOTE

The history buffer has a queue depth of six instructions, but can flush (restore processor state) up to two instructions at a time.

Using the RCPU program trace information available via the VF and the VFLS signals, and snooping the U-bus for instruction fetches, the READI module generates program trace messages to track the change of flow for the RCPU.

23.4.3 BTM Message Formats

BTM messages are of five types — direct, indirect, correction, error, and synchronization.

23.4.3.1 Direct Branch Messages

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode.

The program trace direct branch message has the following format:

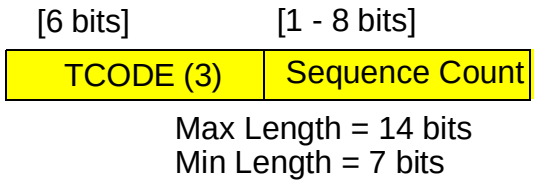


Figure 23-10 Direct Branch Message Format

23.4.3.2 Indirect Branch Messages

Indirect branches include interrupts, exceptions, and all taken branches whose destination is determined at run time. For the RCPU, certain sequential instructions are tagged with the indirect change-of-flow attribute because these instruction affect the machine in a similar manner to true indirect change-of-flow instructions. These instruc-

tions are the ***rfi***, ***isync***, ***mtmsr*** and certain ***mtspr*** (to CMPA – CMPF, ICTRL, ECR and DER)

The program trace indirect branch message has the following format:

[6 bits]	[1 - 8 bits]	[1 - 23 bits]
TCODE (4)	Sequence Count	Relative Address

Max Length = 37 bits
Min Length = 8 bits

Figure 23-11 Indirect Branch Message Format

For compressed code support, six additional bits indicate the starting bit address within the word of the compressed instruction.

The program trace indirect branch with compressed code message has the following format.

[6 bits]	[1 - 8 bits]	[6 bits]	[1 - 23 bits]
TCODE (59)	Sequence Count	Bit Address	Relative Address

Max Length = 40 bits
Min Length = 14 bits

Figure 23-12 Indirect Branch Message Format with Compressed Code

23.4.3.3 Program Trace Correction Message

In case of a mispredicted branch or an exception, a program trace correction message may also be sent indicating a number which corrects the number of instructions (not messages) in the trace.

In the case of a synchronizing branch trace message getting corrected due to an exception or misprediction, the next branch trace message will be a synchronizing message.

Table 23-30 illustrates an example of a program trace correction message in case of a mispredicted branch. **Table 23-31** illustrates an example of a program trace correction message in case of an exception.

NOTE 1

In case of an exception, the sequential instruction count is reset to 0, after the program trace correction message is sent.

NOTE 2

In case of an mispredicted branch, the correction count is always 1 and the sequential instruction count is reset to 1 (to denote the not-taken branch as a sequential instruction), after the program trace correction message is sent. This is because a mispredicted branch is considered to be a sequential instruction.

Table 23-30 Program Trace Correction Due to a Mispredicted Branch

Time	Processor State	Message sent
1	Sequential Instruction	
2	Sequential Instruction	
3	Sequential Instruction	
4	Sequential Instruction	
5	Sequential Instruction	
6	Direct Branch Instruction	Direct Branch Message TCODE = 3 Number of sequential instructions executed since last taken branch = 5
7	Sequential Instruction	
8	Sequential Instruction	
9	Sequential Instruction	
10	Sequential Instruction	
11	Indirect Branch Instruction (mispredicted taken)	Indirect Branch Message TCODE = 4 Number of sequential instructions executed since last taken branch = 4 Unique portion of the target address
12	Sequential Instruction	
13	Sequential Instruction	
14	Sequential Instruction	
15	Branch Correction	Program Trace Correction Message TCODE = 10 Number of instructions to rewind from trace = 1
16	Sequential Instruction	
17	Sequential Instruction	

Table 23-30 Program Trace Correction Due to a Mispredicted Branch (Continued)

Time	Processor State	Message sent
18	Indirect Branch Instruction (predicted taken)	Indirect Branch Message TCODE = 4 Number of sequential instructions executed since last valid taken branch = 3 Unique portion of the target address
At Time 11, the Indirect Branch is mispredicted taken. At Time 15, branch correction occurs due to the mispredicted branch which was taken at Time 11. A program trace correction message is sent out correcting the number of instructions in the trace (1). Sequential instruction which occurred at Time 12, 13, and 14 respectively are not included in the correction count because the tool is not aware that they occurred (they were not transmitted out). At Time 18, the Indirect Branch Message indicates that 3 sequential instructions were executed since trace correction (this includes the mispredicted branch instruction which is considered to be a sequential instruction).		

Table 23-31 Program Trace Correction Due to an Exception

Time	Processor State	Message sent
1	Sequential Instruction	
2	Sequential Instruction	
3	Sequential Instruction	
4	Direct Branch Instruction	Direct Branch Message TCODE = 3 Number of sequential instructions executed since last taken branch = 3
5	Sequential Instruction	
6	Sequential Instruction	
7	Sequential Instruction	
8	Sequential Instruction	
9	Indirect Branch Instruction	Indirect Branch Message TCODE = 4 Number of sequential instructions executed since last taken branch = 4 Unique portion of the target address
10	Sequential Instruction	
11	Sequential Instruction	
12	Indirect Branch Instruction	Indirect Branch Message TCODE = 4 Number of sequential instructions executed since last taken branch = 2 Unique portion of the target address

Table 23-31 Program Trace Correction Due to an Exception (Continued)

Time	Processor State	Message sent
13	Sequential Instruction	
14	Exception due to instruction at Time 8	Program Trace Correction Message TCODE = 10 Number of instructions to rewind from trace = 5
16	Indirect Branch Instruction	Indirect Branch Message TCODE = 4 Number of sequential instructions executed since last taken branch = 0 Unique portion of the target address
At Time 8, the Sequential instruction that causes an exception is issued. At Time 14, the instruction issued at Time 8 causes an exception. A Program Trace Correction Message is sent out correcting the number of instructions in the trace (5). The sequential instruction that occurred at Time 13 is not included in the correction count because the tool is not aware that it occurred (it was not transmitted out). Note: The sequential instruction at Time 8 did not retire and is included in the correction number.		

The program trace correction message has the following format:

[6 bits]	[1 - 8 bits]
TCODE (10)	Number of **instructions** to rewind from trace

Max Length = 14 bits
Min Length = 7 bits

Figure 23-13 Program Trace Correction Message Format

23.4.3.4 Error Message (Queue Overflow)

A program/data/ownership trace overrun error occurs when a trace message cannot be queued due to the queue being full, provided program trace is enabled.

The overrun error causes the message queue to be flushed, and an error message to be queued. The error code within the error message indicates that a program/data/ownership trace overrun error has occurred. The next BTM will be a synchronization message.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00111)

Length = 11 bits

Figure 23-14 Error Message (Queue Overflow) Format

23.4.3.5 Program Trace Synchronization Messages

A program trace synchronization message is transmitted via the auxiliary port (provided program trace is enabled) for the following conditions:

- Initial program trace message upon exit of any system reset will be a synchronization message.
- Upon exit of sleep, deep-sleep and low power down mode, the first BTM will be a synchronization message.
- Initial program trace message upon exit of background debug mode. Upon exiting BDM, the next BTM will be a synchronization message.
- When BTM is enabled, the first BTM will be a synchronization message.
- After 255 program trace messages have been queued without synchronization, the next BTM will be a synchronization message.
- Upon assertion of an event In ($\overline{\text{EVTI}}$) pin. If the READI module is not disabled, an $\overline{\text{EVTI}}$ assertion will cause the next BTM to be a synchronization message (provided the EC field is 0b00 in the DC register).
- Upon occurrence of a watchpoint, the next BTM will be a synchronization message (provided program trace is enabled).
- Occurrence of queue overrun. A program trace overrun error occurs when a trace message cannot be queued due to the queue being full. This causes the message queue to be flushed, and an error message is placed as the first message in the queue. The error code within the Error message will indicate that program/data/ownership trace overrun has occurred. The next BTM will be a synchronization message.
- Sequential instruction count overflow. When the sequential instruction counter reaches its maximum count (up to 256 sequential instructions may be executed), the next BTM will be a program trace synchronization message.The sequential instruction counter is reset.
- Upon entering or exiting code compression mode, the next BTM will be a synchronization message.
- The next change-of-flow instruction fetch following VSYNC will be a synchronization message.

Program trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with program trace regularly. Synchronization messages provide a reference address for subsequent BTMs, in which only the unique portion of the program trace address is transmitted.

NOTE

For program trace synchronization to work, the ICTRL register must be programmed such that show cycle will be performed for all changes in the program flow (ISCTL field = 01).

It is also recommended that the USIU be programmed to ignore instruction show cycles (so as to not impact U-bus performance). Synchronization will only occur at changes in program flow boundaries, and cannot be forced by the READI module. Synchronizations on errors, overflows, as well as periodic synchronizations will not be deterministic to the nearest instruction, but to the next taken change in program flow. The start of program trace (enabled via any means) will be also deferred to the next change in program flow.

Program trace synchronization messages are of the following types:

- Direct branch with compressed code
- Indirect branch with compressed code
- Direct branch
- Indirect branch

23.4.3.6 Direct Branch Synchronization Message

The program trace direct branch synchronization message has the following format:

[6 bits]	[1 bit]	[1 – 23 bits]
TCODE (11)	Messages Cancelled (0b0)	Full target address

Max Length = 30 bits
Min Length = 8 bits

Figure 23-15 Direct Branch Synchronization Message Format

23.4.3.7 Indirect Branch Synchronization Message

The program trace indirect branch synchronization message has the following format:

[6 bits]	[1 bit]	[1 – 23 bits]
TCODE (12)	Messages Cancelled (0b0)	Full target address

Max Length = 30 bits
Min Length = 8 bits

Figure 23-16 Indirect Branch Synchronization Message Format

23.4.3.8 Direct Branch Synchronization Message With Compressed Code

For compressed code support, six additional bits indicate the starting bit address within the word of the compressed instruction. The program trace direct branch synchronization with compressed code message has the following format:

[6 bits]	[6 bits]	[1 – 23 bits]
TCODE (60)	Bit address	Full target address

Max Length = 32 bits
Min Length = 13 bits

Figure 23-17 Direct Branch Synchronization Message Format With Compressed Code

23.4.3.9 Indirect Branch Synchronization Message with Compressed Code

For compressed code support, six additional bits indicate the starting bit address within the word of the compressed instruction. The program trace indirect branch synchronization with compressed code message has the following format:

[6 bits]	[6 bits]	[1 – 23 bits]
TCODE (61)	Bit address	Full target address

Max Length = 35 bits
Min Length = 13bits

Figure 23-18 Indirect Branch Synchronization Message Format With Compressed Code

23.4.3.10 Relative Addressing

The relative address feature is compliant with the IEEE-ISTO 5001 - 1999 recommendations, and is designed to reduce the number of bits transmitted for addresses of indirect branch messages.

The address transmitted is relative to the address of the previous branch trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant ‘1’ in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address. **Figure 23-19** shows how a relative address is generated and how it can be used to recreate the original address.

Previous Address (A1) = 0x0003 FC01, New Address (A2) = 0x0003 F365

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Recreation:

A1 ⊕ M1 = A2

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
⊕
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 23-19 Relative Address Generation and Re-creation

23.4.4 BTM Operation

23.4.4.1 BTM Capture and Encoding Algorithm

BTM is accomplished by capturing instruction fetch information from the U-bus and instruction execution information from the RCPU (VF and VFLS signals), and combining them to generate program trace messages.

23.4.4.2 Instruction Fetch Snooping

Instruction fetches are snooped on the U-bus. There is a one-to-one correspondence between instruction fetches marked with the U-bus program trace attribute and the indication of RCPU VF signal (only 3, 4, 5, and 6) between two synchronization events.

Since U-bus program trace attribute occurs after the indication of VF, it is latched and paired with the nearest (previous) unpaired VF (3, 4, 5, and 6) indication to determine the instruction address.

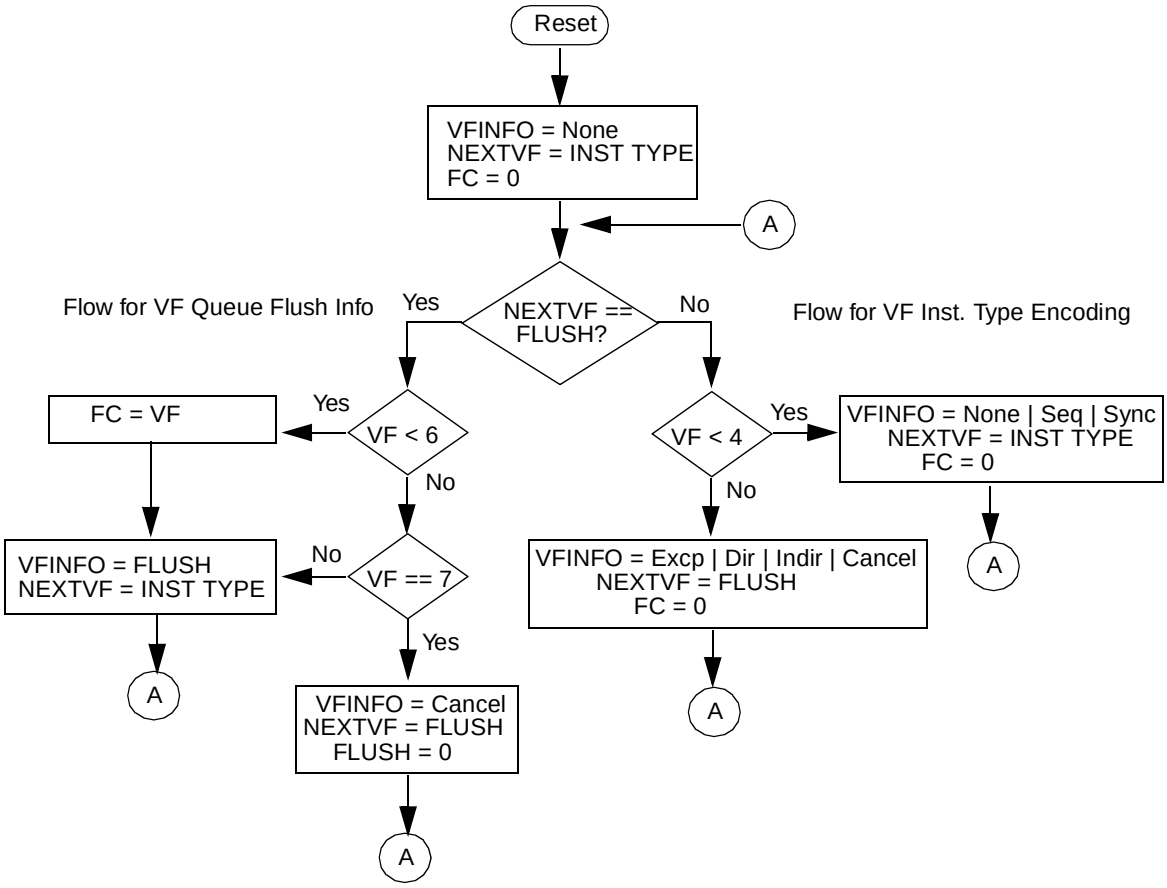
For all other VF indications, except 3, 4, 5, and 6, it is not possible to determine the instruction address.

23.4.4.3 Instruction Execution Tracking

Instruction execution tracking is performed by capturing the RCPU VF and VFLS signals, and decoding them to infer the state of the processor. The RCPU VF signals indicate two classifications of information:

- The current instruction type which is being loaded into the RCPU instruction queue. For further details refer to the ***RCPU Reference Manual* (RCPURM/AD)**.
- The number of instructions which are currently being flushed from the RCPU instruction queue. For further details refer to the ***RCPU Reference Manual* (RCPURM/AD)**.

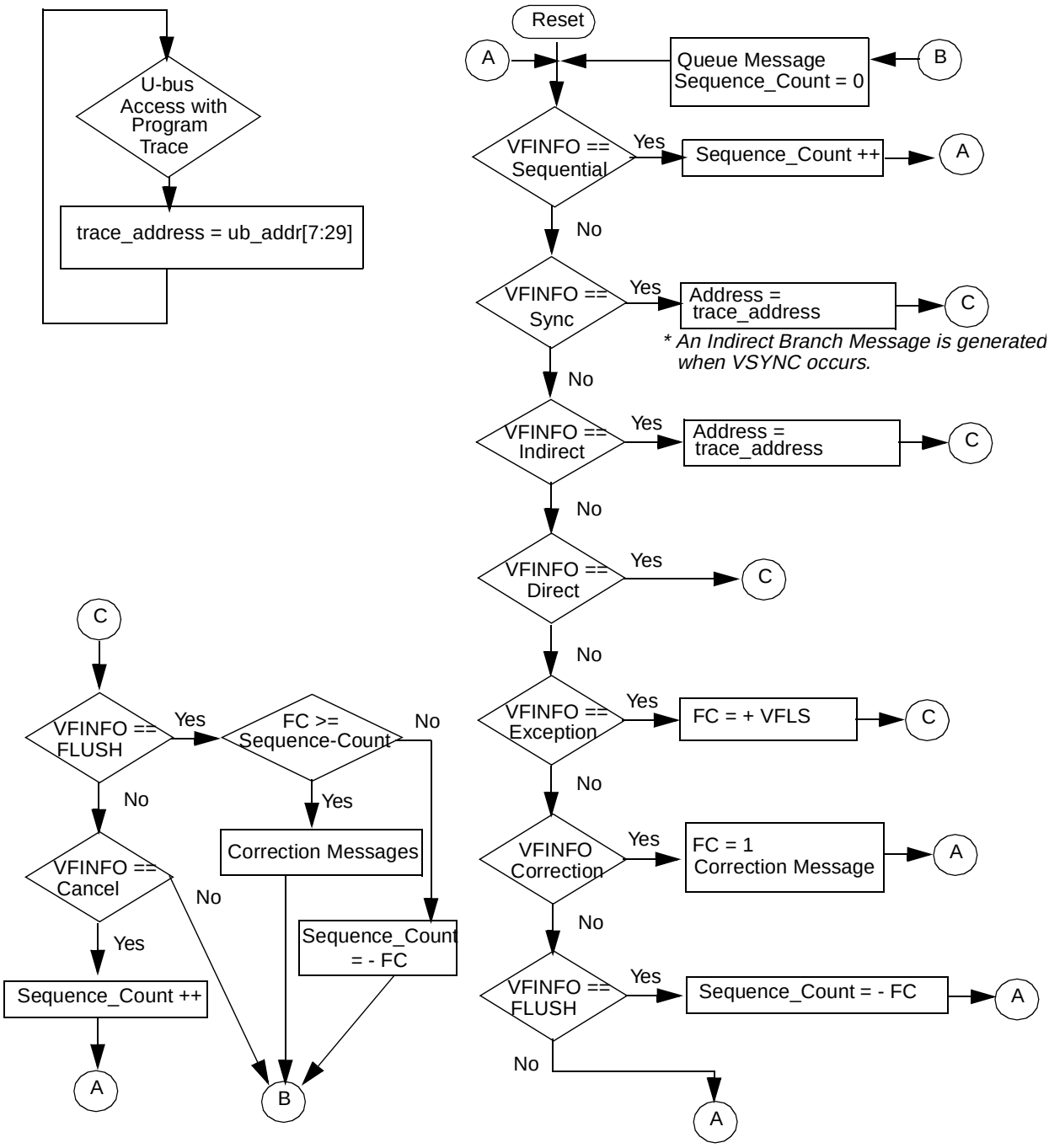
The algorithm to detect the information that is being indicated on the VF signals is depicted in **Figure 23-20**.



VFINFO is the type of information the RCPU VF signals indicate during this cycle
NEXTVF is the type of information the RCPU VF signals will indicate in the next cycle
FC is the number of instructions flushed from the prefetch queue

Figure 23-20 VF State Decoding

Based on the determination of the VF state decoding, **Figure 23-21** depicts the sequence for generating and queueing program trace messages.



VFINFO is the type of information the RCPU VF signals indicate during this cycle
NEXTVF is the type of information the RCPU VF signals will indicate in the next cycle
FLUSH is a state of RCPU VF signals.
VFLS indicates history buffer flush.
FC indicates the number of instructions to be flushed.

Figure 23-21 BTM Encoding Flow

NOTE

In **Figure 23-21** some operations require multiple cycles. The RCPU VFLS signals may take up to three clocks to indicate the full status of the history buffer flush (two instructions are flushed per clock).

For compressed code support, refer to **Figure 23-22** for further details.

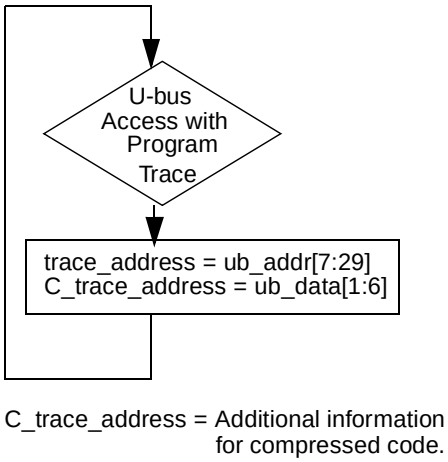


Figure 23-22 BTM Encoding Flow — Compressed Code Support

23.4.4.4 Instruction Flush Cases

The various conditions under which the RCPU may signal instruction flushes of the RCPU prefetch queue or RCPU history buffer are:

1. A taken branch (direct, indirect, interrupt or exception) will cause the instruction prefetch queue (which contains instructions from the now old stream) to be flushed, and fetching will start from the branch target stream. The sequential instruction count will be updated to reflect this.
2. A mispredicted branch will cause instructions fetched from the new stream to be flushed, and fetching will resume from the old stream. It will also require a program trace message to be cancelled and the trace to be corrected.
3. An exception can cause cancellation of multiple taken branches which may require cancelling multiple program trace messages.

23.4.5 BTM Queueing

READI implements a queue 16 message deep for program trace, data trace, and ownership trace messages. Messages that enter the queue are transmitted via the output auxiliary port in the order in which they are queued.

If multiple trace messages need to be queued at the same time, program trace messages will have the highest priority unless the data trace buffers are full, in which case the data trace messages are given temporary higher priority than the program trace messages.

23.4.6 Timing Diagram

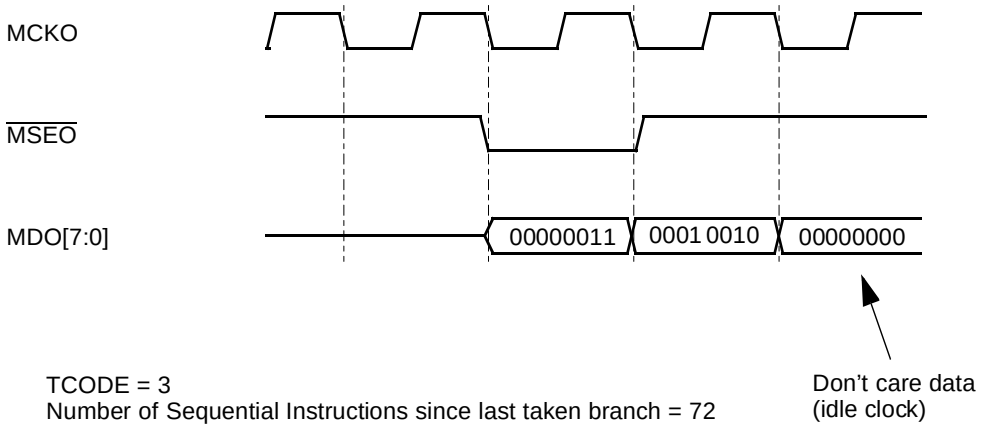


Figure 23-23 Direct Branch Message

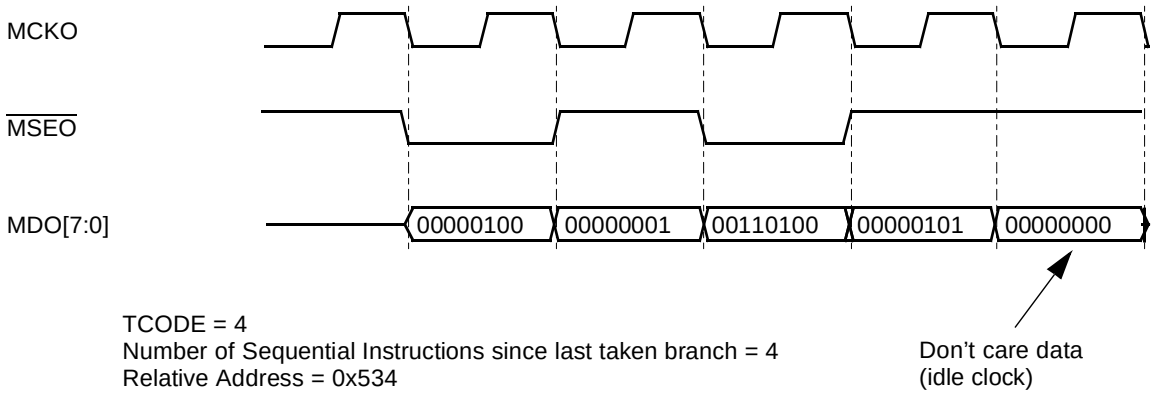


Figure 23-24 Indirect Branch Message

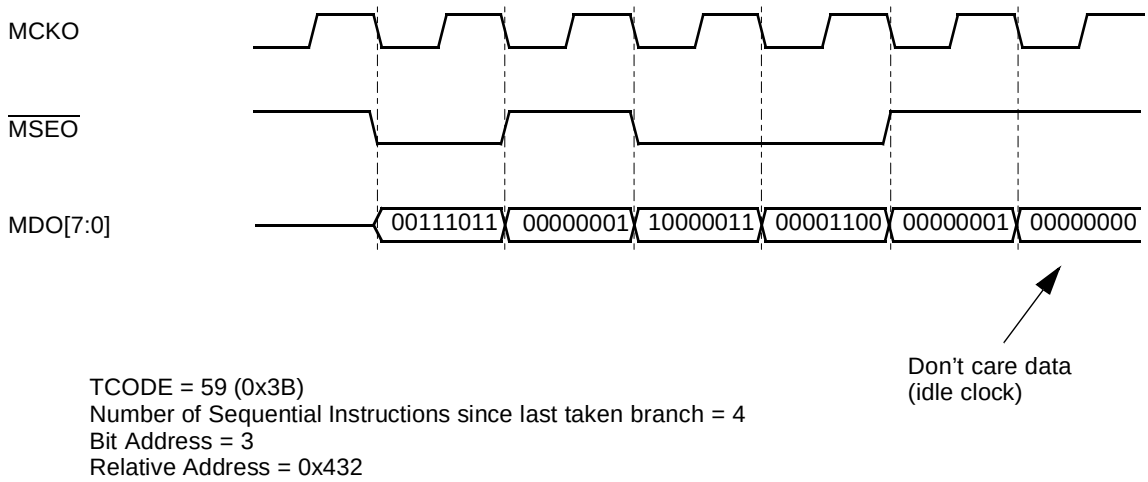


Figure 23-25 Indirect Branch Message With Compressed Code

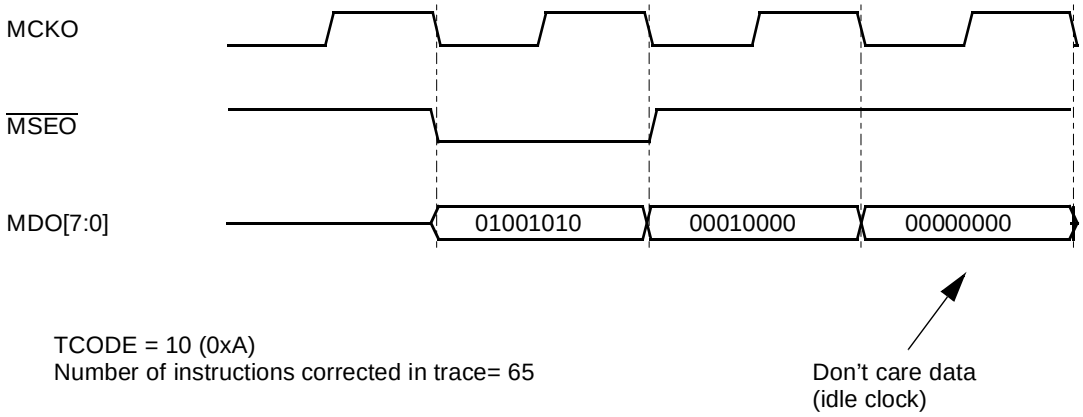


Figure 23-26 Program Trace Correction Message

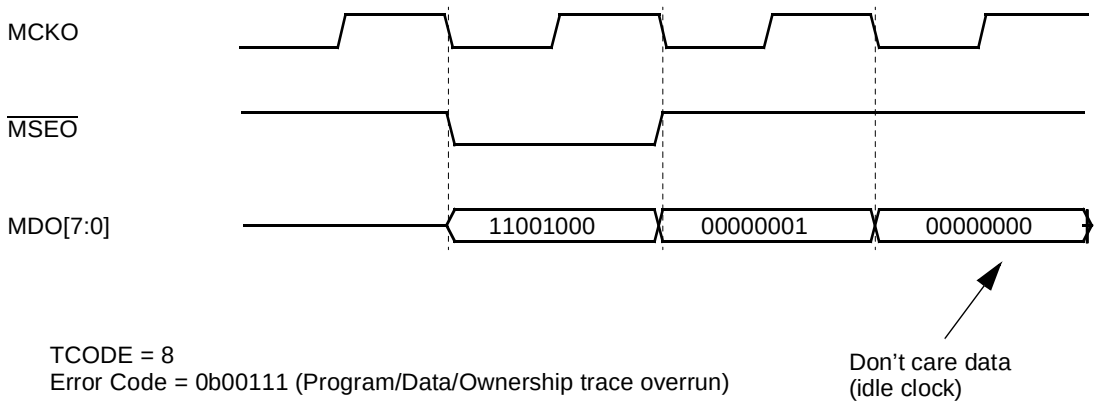


Figure 23-27 Error Message (Program/Data/Ownership Trace Overrun)

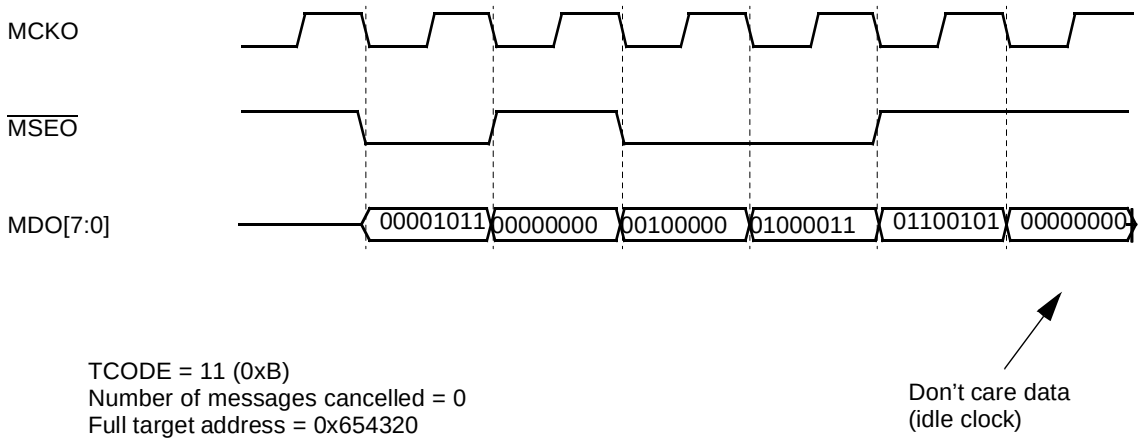


Figure 23-28 Direct Branch Synchronization Message

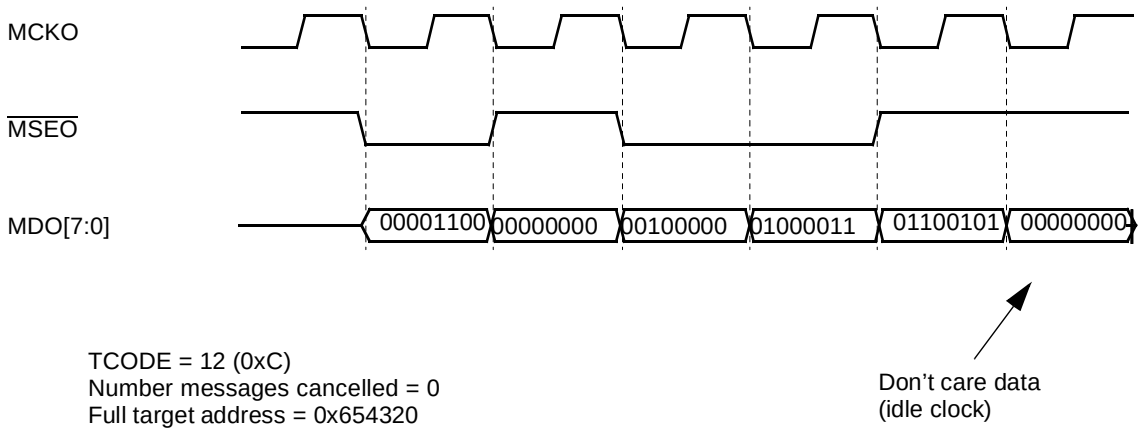


Figure 23-29 Indirect Branch Synchronization Message

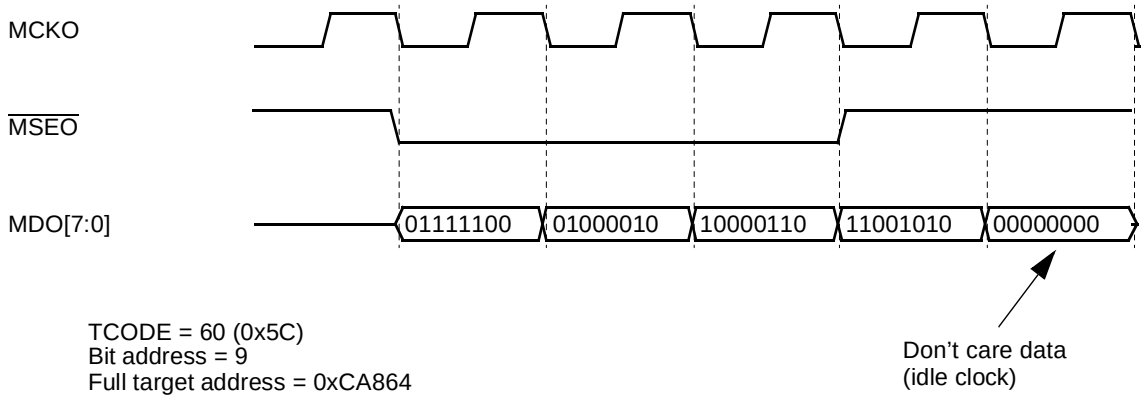


Figure 23-30 Direct Branch Synchronization Message With Compressed Code

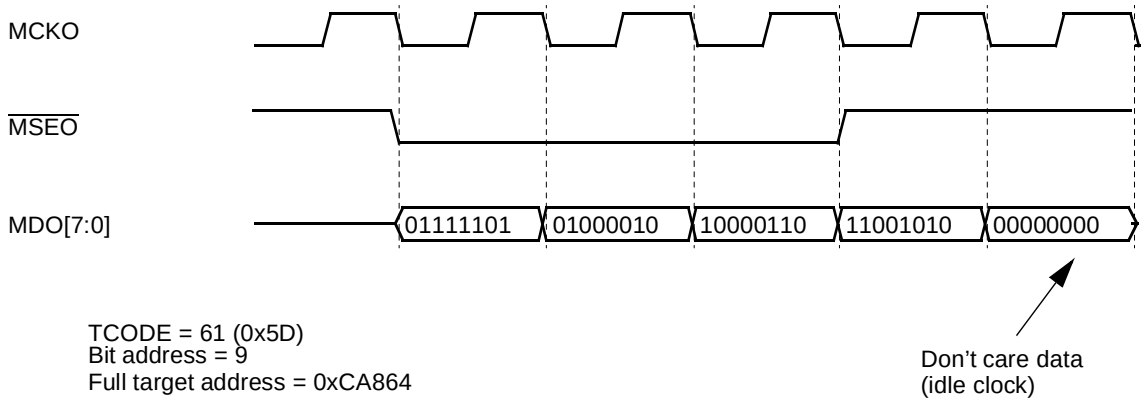


Figure 23-31 Indirect Branch Synchronization Message With Compressed Code

23.4.7 Program Trace Guidelines

Refer to **23.2.5.1 Program Trace Guidelines** for further details.

23.5 Data Trace

This section details the data trace mechanism supported by READI. Data trace is implemented via data write messaging (DWM) and data read messaging (DRM), as per the IEEE-ISTO 5001 - 1999.

23.5.1 Data Trace for the Load/Store Bus (L-bus)

The L-bus allows the RCPU to perform loads and stores, and the L2U to read and write the L-bus resources. Snooping for data trace on the L-bus requires the READI module to handle the full range of L-bus cycles. This includes various cases of pipelining and aborted cycles.

Data trace requires snooping the L-bus cycles, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The READI module traces all data accesses that meet the selected range and attributes. This includes all RCPU initiated accesses and all L-bus accesses.

L-bus data cycles can have data sizes of 8, 16 or 32 bits.The READI module supports all three data sizes.

23.5.2 Data Trace Message Formats

Data trace messages are of five types:

- Data write
- Data read
- Data write synchronization

- Data read synchronization
- Error message

23.5.2.1 Data Write Message

The data write message contains the data write value and the address of the target location, relative to the previous data trace message.

The data write message has the following format:

[6 bits]	[1 to 25 bits]	[8, 16, or 32 bits]
TCODE (5)	Relative Address	Data Value

Max Length = 63 bits
Min Length = 15 bits

Figure 23-32 Data Write Message Format

23.5.2.2 Data Read Message

The data read message contains the data read value and the address of the target location, relative to the previous data trace message.

The data read message has the following format:

[6 bits]	[1 to 25 bits]	[8, 16, or 32 bits]
TCODE (6)	Relative Address	Data Value

Max Length = 63 bits
Min Length = 15 bits

Figure 23-33 Data Read Message Format

23.5.2.3 Data Trace Synchronization Messages

A data trace synchronization message shall be transmitted via the auxiliary port (provided data trace is enabled) for the following conditions:

- Initial data trace message upon exit of any system reset will be a synchronization message.
- Upon exit of sleep, deep-sleep and low power down mode, the first data trace message will be a synchronization message.
- Initial data trace message upon exit of background debug mode. Upon exiting BDM, the next data trace message will be a synchronization message.
- When data trace is enabled, the first data trace message will be a synchronization message.

- After 255 data trace messages have been queued without synchronization, the next data trace message will be a synchronization message.
- Upon assertion of an event In ($\overline{EVTI}$) pin. If the READI module is not disabled at reset, when $\overline{EVTI}$ asserts, if the EC field is 0b00 in the DC register, the next data trace message will be a synchronization message.
- Upon occurrence of a watchpoint, the next data trace message will be a synchronization message.
- Occurrence of queue overrun. A data trace overrun error occurs when a trace message cannot be queued due to the queue being full (provided data trace is enabled). This causes the message queue to be flushed, and an error message is placed as the first message in the queue. The error code within the Error message indicates that program/data/ownership trace overrun has occurred. The next data trace message will be a synchronization message.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted.

Data trace synchronization messages are of two types:

- Data write
- Data read

23.5.2.4 Data Write Synchronization Message

The data write synchronization message has the following format:

[6 bits]	[1 bit]	[1 to 25 bits]	[8, 16, or 32 bits]
TCODE (13)	Messages Cancelled (0b0)	Full target address	Data Value

Max Length = 64 bits
Min Length = 16 bits

Figure 23-34 Data Write Synchronization Message Format

23.5.2.5 Data Read Synchronization Messaging

The data read synchronization message has the following format:

[6 bits]	[1 bit]	[1 to 25 bits]	[8, 16, or 32 bits]
TCODE (14)	Messages Cancelled (0b0)	Full target address	Data Value

Max Length = 64 bits
Min Length = 16 bits

Figure 23-35 Data Read Synchronization Message Format

23.5.2.6 Relative Addressing

Refer to **23.5.2.6 Relative Addressing** for further details.

23.5.3 Error Message (Queue Overflow)

A program/data/ownership trace overrun error occurs when a trace message cannot be queued due to the queue being full, provided data trace is enabled.

The overrun error causes the message queue to be flushed, and a error message to be queued. The error code within the error message indicates that a program/data/ownership trace overrun error has occurred. The next DTM will be a synchronization message.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00111)

Length = 11 bits

Figure 23-36 Error Message (Queue Overflow) Format

23.5.4 Data Trace Operation

Data trace is performed by snooping the L-bus for read or write cycles. Data trace functions are enabled by setting the appropriate fields in the DC register and the DTA registers. For details on field configuration, refer to **23.2.1.4 Development Control (DC) Register** and **23.2.1.9 Data Trace Attributes 1 and 2 (DTA1 and DTA2) Registers** respectively.

Data trace flow is depicted in **Figure 23-37**.

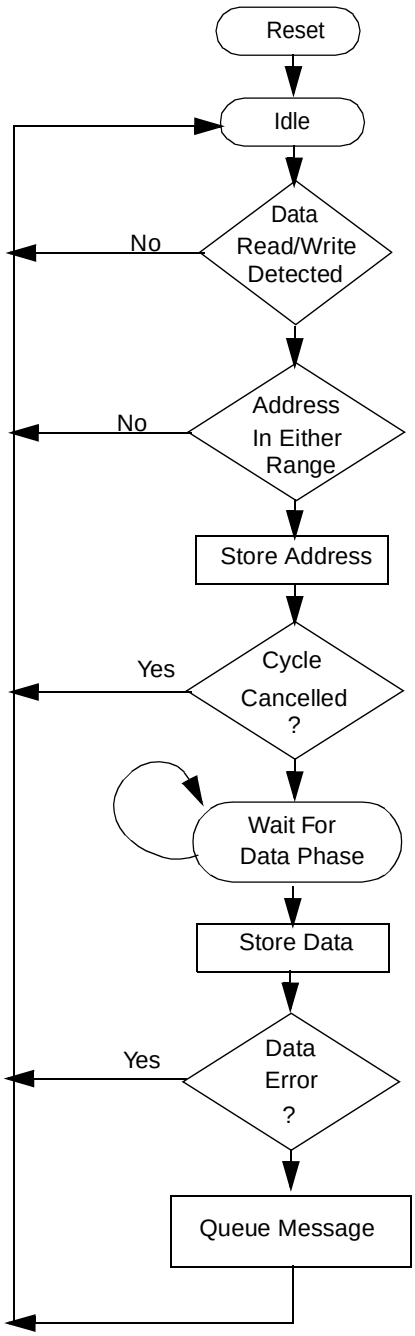


Figure 23-37 Data Trace Flow Diagram for Non-Pipelined Access

23.5.5 Data Trace Windowing

Data trace windowing is achieved via the address range within the DTEA and the DTSA fields of the DTA registers. All L-bus accesses which fall within these two address ranges, provided the address ranges are enabled in either DTA register, are candidates to be transmitted. Data read and/or data write trace may be enabled via the TA field of the data trace attributes registers (DTA).

NOTE 1

Data trace ranges are word aligned. Therefore, the address range fields (DTEA and DTSA) of the DTA registers are only 23 bits wide and, as such, should be assigned by the tool with the 23 most significant bits of the intended 25-bit range address.

NOTE 2

The off-core PowerPC special purpose register (SPR) map cannot be distinguished from the normal memory map accesses via the defined address range control. If data trace ranges are set up such that the off-core PowerPC SPR map falls within active ranges, then accesses to these off-core PowerPC SPRs will be traced, and the messages will not be distinguishable from accesses to normal memory map space. Off-core PowerPC SPRs typically exist in the 8-Kbyte – 16-Kbyte lowest memory block (0x2000 - 0x3FF0). If data or peripherals are mapped to this space, load/stores to PowerPC SPRs will be indistinguishable from data or peripheral accesses.

23.5.6 Special L-bus Cases

Special L-bus cases are handled as described in **Table 23-32**.

Table 23-32 Special L-bus Case Handling

Special Case	Action
L-bus Cycle Aborted	Cycle ignored
L-bus Cycle with data error	Message discarded
L-bus Cycle terminated due to address error	Cycle ignored
L-bus Cycle completed without error	Cycle captured and transmitted
L-bus Cycle initiated by READI (Read/Write Access)	Cycle ignored
L-bus Cycle is an instruction fetch	Cycle ignored
Data Storage Interrupt	Cycle ignored
System Rese	Cycle ignored

23.5.7 Data Trace Queuing

READI implements a queue 16 messages deep for queuing program trace, data trace, and ownership trace messages. Messages that enter the queue are transmitted via the output auxiliary port in the order in which they are queued.

NOTE

If multiple trace messages need to be queued at the same time, program trace messages have a higher priority for queue entry than data trace messages, unless the data trace buffers are full, in which case the data trace messages are given temporary higher priority than the program trace messages.

23.5.8 Throughput and Latency

23.5.8.1 Assumptions for Throughput Analysis

- All accesses are data trace only
- 56-MHz operation
- Output pins are always free (not in middle of transmission) when requested
- Relative Address field for data trace messages is 20 bits
- Data field for data trace messages is 32 bits
- One idle clock between data trace messages

23.5.8.2 Throughput Calculations

The data (read or write) trace message is 58 bits (6 [TCODE] + 20 [Relative address] + 32 [Data]).

Data trace messages are transmitted out via the MDO pins. Hence it will take eight clocks (58 bits/8 MDO pins) to send a message. There will be one idle clock before the next data trace message can be sent.

At 56 MHz, it will take 161ns ((8+1) x 17.8) to transmit the message.

Therefore, the average number of data trace messages that can be transmitted out is 6.2 million (1/161ns) per second, or 24.8 million bytes of read/write data per second.

23.5.9 Timing Diagrams

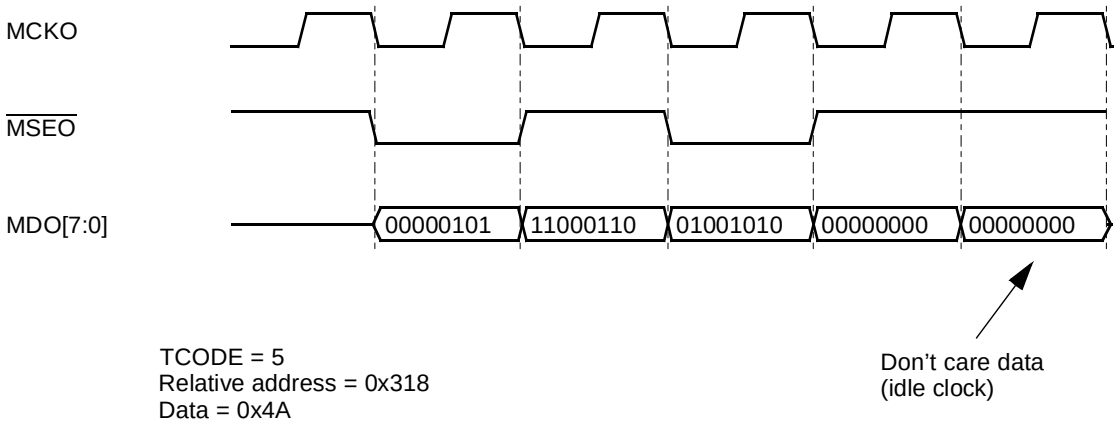


Figure 23-38 Date Write Message

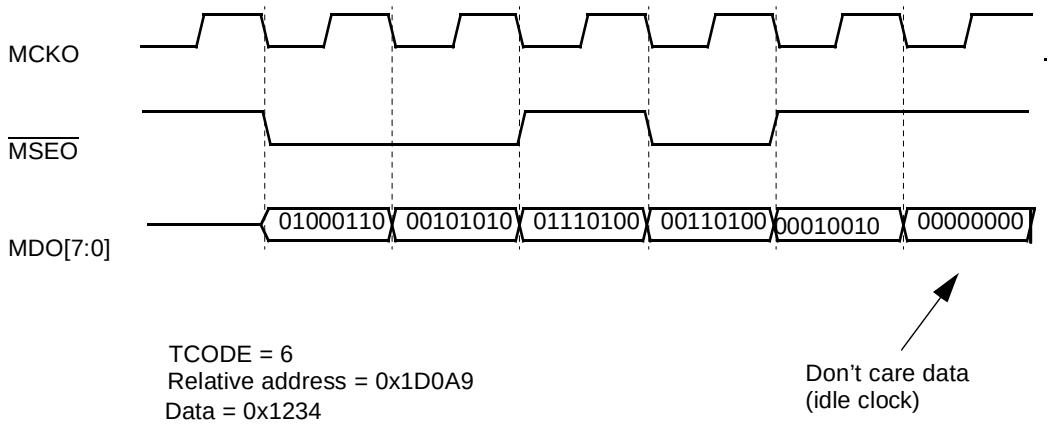


Figure 23-39 Data Read Message

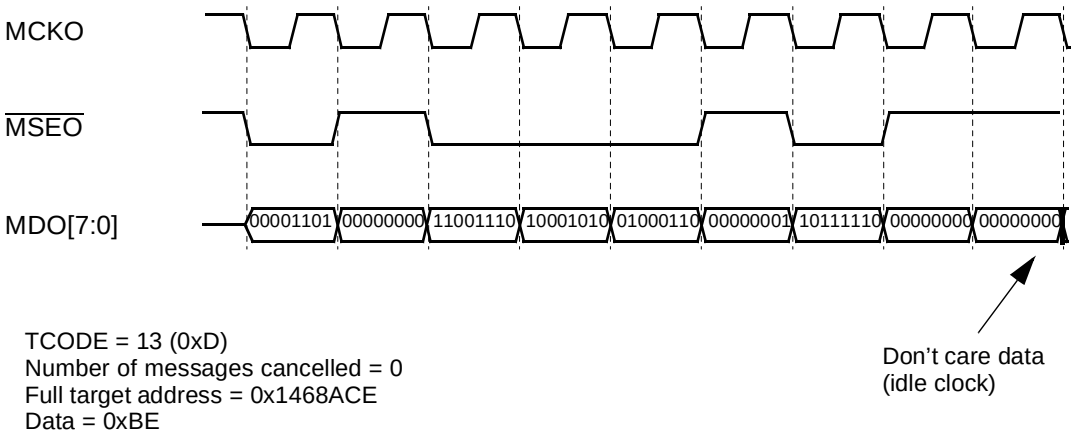


Figure 23-40 Data Write Synchronization Message

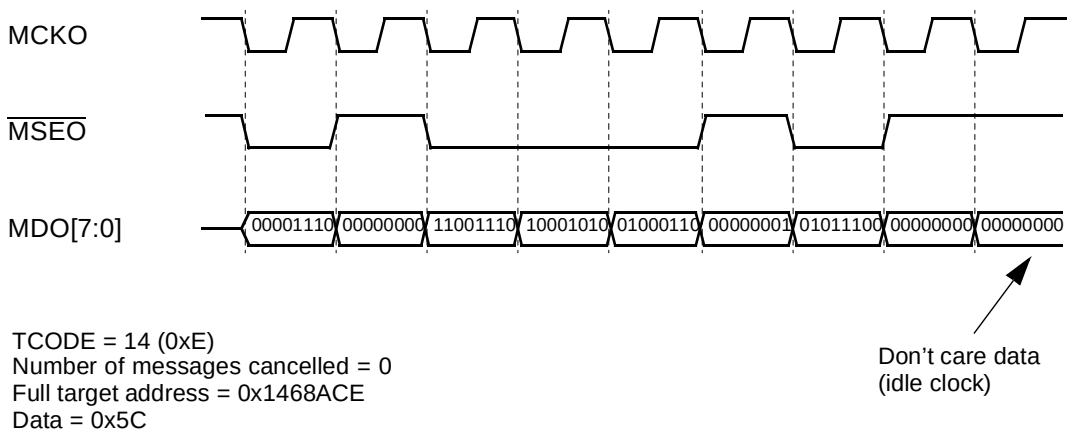


Figure 23-41 Data Read Synchronization Message

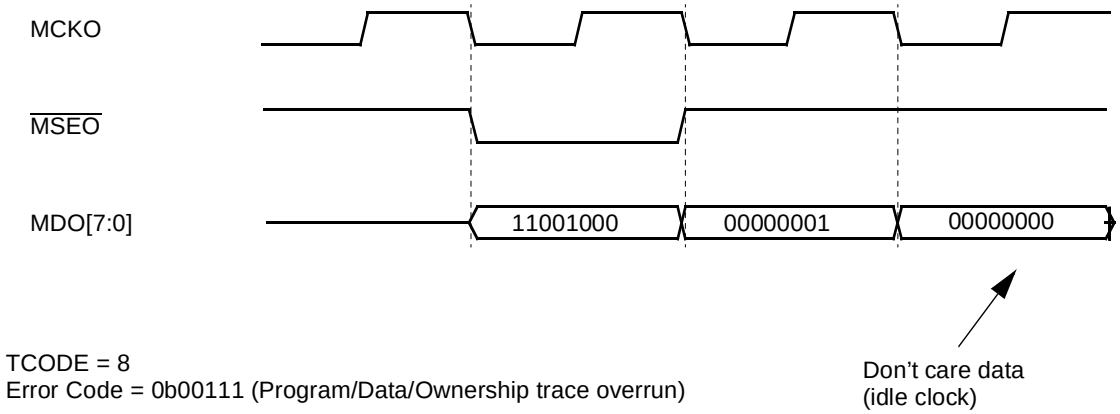


Figure 23-42 Error Message (Program/Data/Ownership Trace Overrun)

23.6 Read/Write Access

The read/write access feature allows access to internal memory mapped space via the auxiliary port. Read/write mechanism supports single and block, reads and writes.

23.6.1 Functional Description

The READI module is capable of bus mastership on the L-bus and for setting up and reading data and status.

All accesses are setup and initiated to the read/write access register (RWA) and upload/download information register (UDI) via the four auxiliary access public messages: device ready for upload/download, upload request (tool requests information),

download request (tool provides information), upload/download information (device/ tool provides information).

Read/write access features are enabled by setting the appropriate fields in the RWA register. For details on field configuration, refer to **23.2.1.7 Read/Write Access (RWA) Register**.

The functional flow for read/write access to memory mapped locations and PowerPC registers is depicted in **Figure 23-43**.

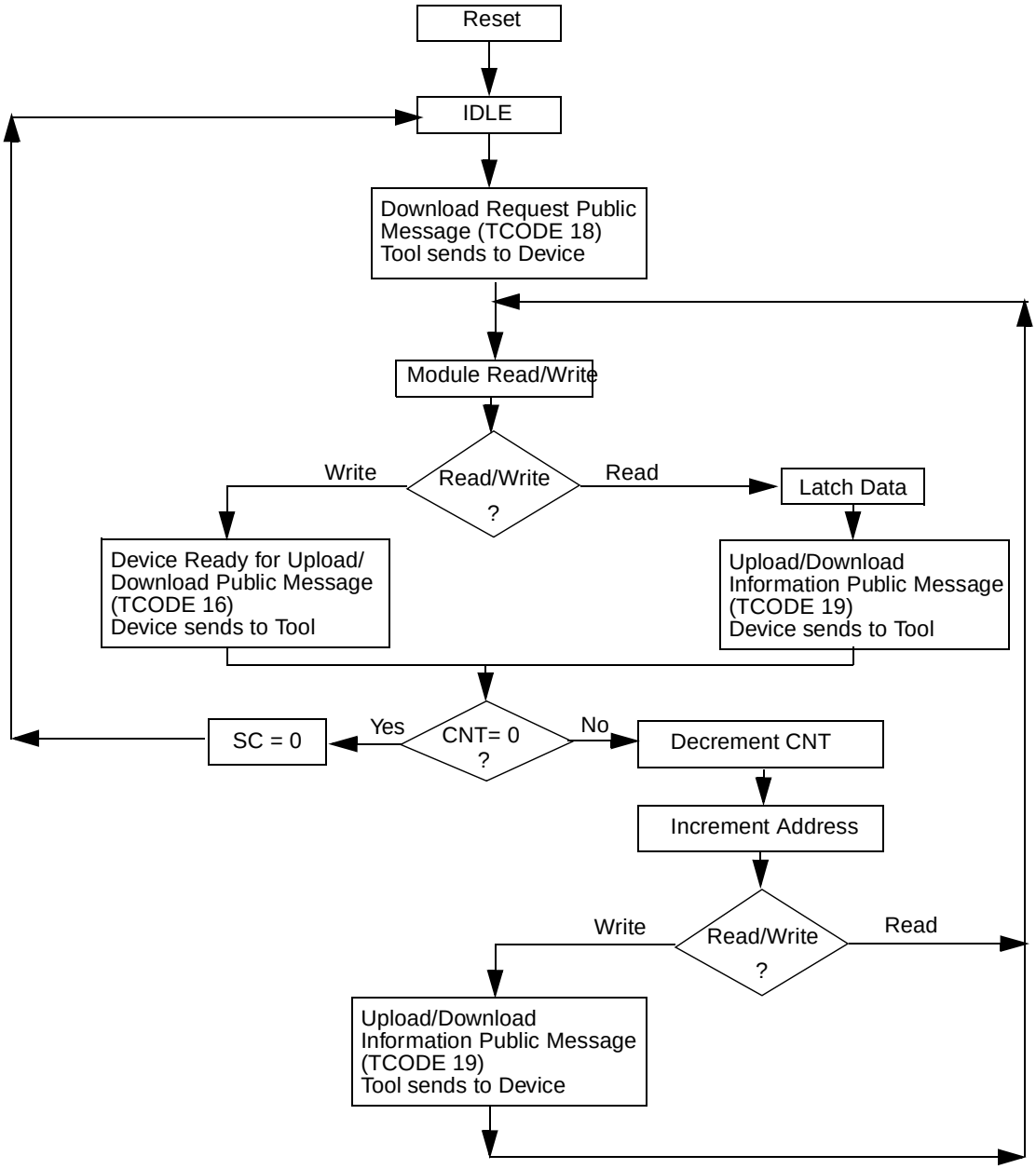


Figure 23-43 Read/Write Access Flow Diagram

23.6.2 Write Operation to Memory Mapped Locations and PowerPC Registers

23.6.2.1 Single Write Operation

For a single write access to memory mapped locations and PowerPC registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting download request public message (TCODE=18).
2. The download request public message contains:
   a. TCODE(18)
   b. Access opcode 0xF which signals that subsequent data needs to be stored in the RWA register.
   c. Configure the RWA register fields as follows:
      - Start/complete (1 to indicate start access) -> SC
      - Read/write address (write address) -> RWAD
      - Read/write (1 to indicate a write access) -> RW
      - Word size (32 bits, 16 bits, 8 bits) -> SZ
      - Write data (write data) -> WD
      - Privilege (user data/instruction, supervisor data/instruction) -> PRV
      - Map select (select memory map, 00 or 01) -> MAP
      - Access Count (0 to indicate single access) -> CNT
3. After completion of the write operation, the device ready for upload/download public message (TCODE=16) is transmitted to the tool indicating that the device is ready for next access.
4. The SC bit is cleared to indicate that the write access is complete.

23.6.2.2 Block Write Operation

For a block write access to memory mapped locations and PowerPC registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting download request public message (TCODE = 18).
2. The download request public message contains:
   a. TCODE(18)
   b. Access opcode 0xF which signals that subsequent data needs to be stored in the RWA register.
   c. Configure the RWA register fields as follows
      - Start/complete (1 to indicate start access) -> SC
      - Read/write address (starting write address of block) -> RWAD
      - Read/write (1 to indicate a write access) -> RW
      - Word size (32 bits, 16 bits, 8 bits) -> SZ
      - Write data (write data) -> WD
      - Privilege (user data/instruction, supervisor data/instruction) -> PRV
      - Map select (select memory map, 00 or 01) -> MAP

- Access count (non zero number to indicate size of block access) -> CNT

3. After completion of this write operation, the device ready for upload/download public message (TCODE = 16) is transmitted to the tool indicating that the device is ready for next access.
4. The specified address (stored in RWAD field) is incremented to the next word size and the number in the CNT field is decremented. The SC field is not cleared.
5. The tool transmits the next upload/download information public message (TCODE = 19).
6. The upload/download information public message contains:
   a. TCODE(19)
   b. Write data (write data -> UDI)
7. After the completion of this write operation, the device ready for upload/download public message (TCODE = 16) is transmitted to the tool indicating that the device is ready for next access.
8. The specified address (in RWAD field) is incremented to the next word size and the number in the CNT field is decremented. The SC field is not cleared.
9. Steps 5 through 8 are repeated until the count value in the CNT field of RWA register equals zero. The SC bit is cleared to indicate end of the block write access.

NOTE

For downloading write data to the device for block write operation, the download request public message (TCODE = 18) should not be used to write subsequent data to the UDI register. Data written to the UDI register (via download request message, TCODE 18) is not used by the device for any read/write operation.

23.6.3 Read Operation to Memory Mapped Locations and PowerPC Registers

23.6.3.1 Single Read Operation

For a single read access to memory mapped locations and PowerPC registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting download request public message (TCODE = 18).
2. The download request public message contains:
   a. TCODE(18)
   b. Access opcode 0xF which signals that subsequent data needs to be stored in the RWA register.
   c. Configure the RWA fields as follows:
      - Start/complete (1 to indicate start access) -> SC
      - Read/write address (read address) -> RWAD

- Read/write (0 to indicate a read access) -> RW
- Word size (32 bits, 16 bits, 8 bits) -> SZ
- Write data (0xXXXXXXXX-> WD [don't care])
- Privilege (user data/instruction, supervisor data/instruction) > PRV
- Map select (select memory map, 00 or 01) -> MAP
- Access count (0 to indicate single access) -> CNT

3. Data read from the specified address is stored in the UDI register.
4. Once the read access is completed, the upload/download information public message (TCODE = 19) is transmitted to the tool along with the data read from the UDI register. This message also indicates that the device is ready for next access.
5. The SC field in the RWA register is cleared.

23.6.3.2 Block Read Operation

For a block read access to memory mapped locations and PowerPC registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting download request public message (TCODE = 18).
2. The download request public message contains:
   a. TCODE(18)
   b. Access opcode 0xF which signals that subsequent data needs to be stored in the RWA register.
   c. Configure the RWA fields as follows:
      - Start/complete (1 to indicate start access) -> SC
      - Read/write address (starting read address of block) -> RWAD
      - Read/write (0 to indicate a read access) -> RW
      - Word size (32 bits, 16 bits, 8 bits) -> SZ
      - Write data (0xXXXXXXXX-> WD [don't care])
      - Privilege (user data/instruction, supervisor data/instruction) > PRV
      - Map select (select memory map, 00 or 01) -> MAP
      - Access count (non-zero number to indicate block access) -> CNT
3. Data read from the specified address is stored in the UDI register.
4. After the completion of this read operation, the upload/download information public message (TCODE=19) is transmitted to the tool along with the data read from the UDI register. This message also indicates that the device is ready to perform the next read operation.
5. The specified address (in RWAD field) is incremented to the next word size and the number in the CNT field is decremented. The SC field is not cleared.
6. The data read from the new address is stored in the UDI register.
7. Steps 4 through 7 are repeated until the count value in the CNT field of RWA register equals zero. The SC bit is cleared to indicate end of the block read access.

23.6.4 Read/Write Access to Internal READI Registers

23.6.4.1 Write Operation

For a write access to internal READI registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting download request public message (TCODE = 18).
2. The download request public message contains:
   a. TCODE(18)
   b. Access opcode, which specifies the register where data needs to be written, (e.g., access opcode 0x14 indicates that DTA1 register is the target register).
   c. Data to be written to the register.
3. After the data has been written to the targeted register, the device ready for upload/download public message (TCODE = 16) is transmitted to the tool indicating that the device is ready for next access.

23.6.4.2 Read Operation

For a read access to internal READI registers, the following sequence of operations need to be performed via the auxiliary port:

1. The tool confirms that the device is ready before transmitting upload request public message (TCODE = 17).
2. The upload request public message contains:
   a. TCODE(17)
   b. Access opcode, which specifies the register where data needs to be read from, (e.g., access opcode 0x14 indicates that DTA1 register is the target register).
3. The upload/download information public message (TCODE =19) is transmitted to the tool along with the data read from the targeted register indicating that the device is ready for next access.

23.6.5 Error Handling

The READI module handles the various error conditions in the manner shown in the following sections.

23.6.5.1 Access Alignment

The READI module will force address alignment based on the word size field (SZ) value. If the SZ field indicates word (32-bit) access, the least significant two bits of the read/write address field (RWAD) are ignored. If the SZ field indicates half-word (16-bit) access, the least significant bit of the read/write address field (RWAD) is ignored.

23.6.5.2 L-bus Address Error

An address error occurs on the L-bus when the address phase of a cycle is not completed normally. This could occur because of address not being valid or the address map not being valid. In such cases:

1. The access is terminated without retrying.
2. The SC bit of the RWA is reset. Block accesses do not continue.
3. The error message (TCODE = 8) is transmitted (error code 0b00011).

23.6.5.3 L-bus Data Error

L-bus data error is signalled due to:

- L-bus data phase error.
- U-bus address phase error (for a L-bus to U-bus cycle).
- U-bus data phase error (for a L-bus to U-bus cycle).

L-bus data error conditions are signalled along with the transfer acknowledge for the access. L-bus data error conditions may occur because of privilege violations, access to protected memory, etc. In such cases, for a read access, the ERR bit of the UDI is set, and the DV bit in the UDI is reset at the termination of the access. For a write access, an error public message (TCODE = 8) is transmitted (error code 0b00011).

23.6.6 Exception Sequences

The following cases are defined for sequences of the read/write protocol that differ from those described in the above sections:

1. If the SC bit is set to start READI read/write accesses, without valid values in the RWAD, then an L-bus address error may occur, which is handled as described above.
2. If a block access is in progress with all the cycles not yet completed, and the RWA is written to again, (with or without modifications), then the block access is terminated at the boundary of the nearest completed access. The resulting data is discarded and not written to the UDI. If a new access has been programmed in the RWA register, then that access will start once the controller has recovered.
3. When a block access is in progress with all the cycles not yet completed, writing the SC bit to 0 in RWA register will terminate the block access and device will send out device ready for upload/download message.
4. If a any type (single/block) of access is in progress with the cycles not yet completed, and system reset occurs, the device will send out an error message. The access will be terminated and the SC bit will be reset.
5. If any type of (single/block) of access is requested while system is in reset, the device will send out an error message. The access will not be started and the SC bit will be reset.

23.6.7 Secure Mode

For details refer to **23.1.4.2 Security**.

23.6.8 Error Messages

23.6.8.1 Read/Write Access Error

An error message is sent out when an L-bus access error or data error on a write access occurs. The error code within the error message indicates that an L-bus address or L-bus data error occurred. For other error handling, see **23.6.5 Error Handling**.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00011)

Length = 11 bits

Figure 23-44 Error Message (Read/Write Access Error) Format

23.6.8.2 Invalid Message

An error message is sent out when an invalid message is received by READI. The error code within the error message indicates that an invalid TCODE was detected in the auxiliary input messages by the pin input formatter.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00100)

Length = 11 bits

Figure 23-45 Error Message (Invalid Message) Format

NOTE

If the TCODE is valid, then READI will expect that the correct number of packets have been received and no further checking will be performed. If the number of packets received by READI is not correct, READI response is not deterministic.

23.6.8.3 Invalid Access Opcode

An error message is sent out when an invalid access opcode is received by READI. The error code within the error message indicates that an invalid access opcode was detected in the auxiliary input messages by the pin input formatter.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00101)

Length = 11 bits

Figure 23-46 Error Message (Invalid Access Opcode) Format

23.6.9 Faster Read/Write Accesses With Default Attributes

Read/write access throughput may be increased by taking advantage of the default settings of the RWA register, and truncating the least significant zero bits of the download request message. For example, to read a word from the default memory map, with default attributes, a download request message that selects the RWA register, and transmits the SC, RWAD, RW fields only is sufficient. This message will contain 41 bits instead of the 94 bits for writing the full contents of the RWA register. See **Table 23-11** and **23.2.4 Partial Register Updates** for RWAR and partial register update details respectively.

NOTE

The last data bit transmitted in the download request message (TCODE 18) will always be the MSB of the register referenced by the opcode (SC field in the case of the RWA register).

23.6.10 Throughput and Latency

Throughput analysis has been performed for various read/write access cases such as single write, block write, single byte read, single word read, block byte read, block word read accesses to memory mapped locations. Data is presented for the two cases when the RWA register is written partially and completely.

23.6.10.1 Assumptions for Throughput Analysis

- All accesses are single read accesses only.
- MCKI running at 28 MHz.
- MCKO running at 56 MHz.
- 56 MHz internal operation.
- Five clock internal L-bus access (read)
- Output pins always free (not in middle of transmission) when requested.
- One idle clock between read messages.
- No delay from tool in responding — tool keeps up with READI port.

Table 23-33 Throughput Comparison for FPM and RPM MDO/MDI Configurations

Access Type	Reduced Port Mode 2 MDO / 1 MDI pins		Full Port Mode 8 MDO / 2 MDI pins	
	Full RWAR Update	Partial RWAR Update	Full RWAR Update	Partial RWAR Update
Single Write Access to memory mapped location – Word and Byte access (In Million Messages Per Second)	0.28	0.35	0.53	0.65
Single Read Access to memory mapped location – Word access (In Million Messages Per Second)	0.25	0.51	0.52	1.05
Single Read Access to memory mapped location – Byte access (In Million Messages Per Second)	0.27	0.56	0.53	1.05
Block Write Access to memory mapped locations – 64-Kbyte block (Word and Byte) write access (In 64-Kbyte Block Writes Per Second)	9	9	17	17
Block Read Access to memory mapped locations – 64-Kbyte block (Word) read access (In 64-Kbyte Block Writes Per Second)	32	32	77	77
Block Read Access to memory mapped locations – 64-Kbyte block (Byte) read access (In 64-Kbyte Block Writes Per Second)	61	61	95	95

23.7 Timing Diagrams

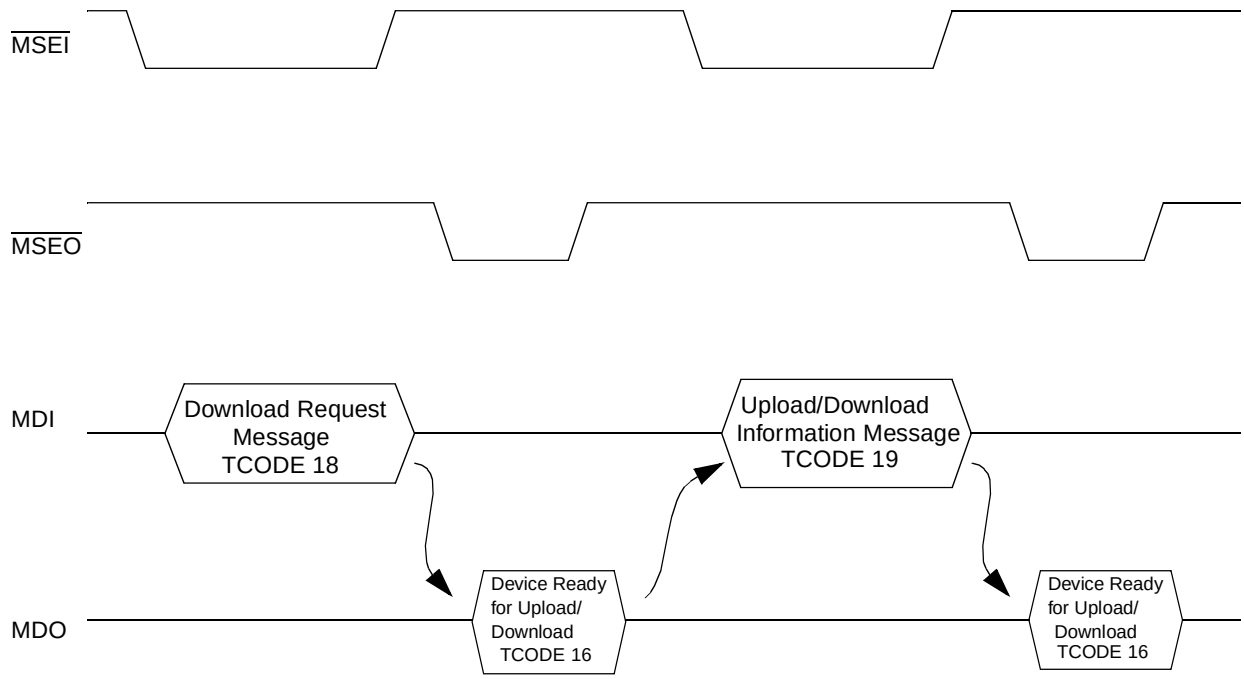


Figure 23-47 Block Write Access

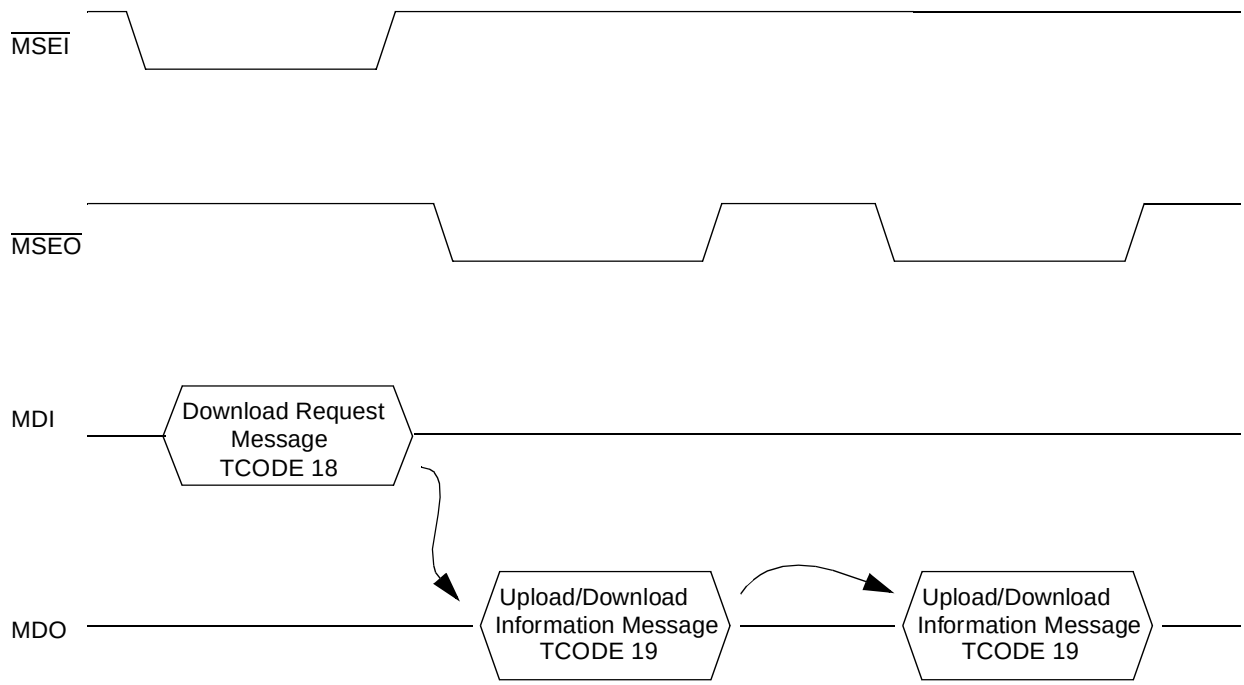


Figure 23-48 Block Read Access

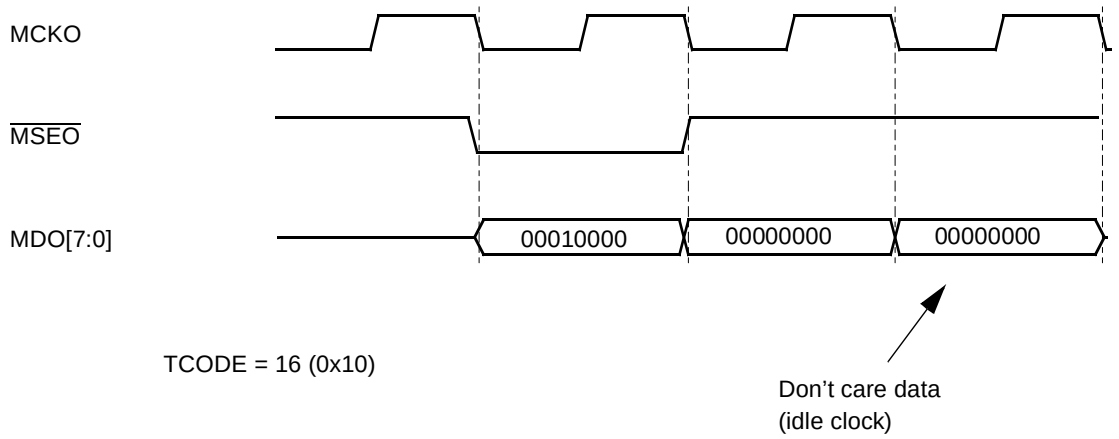


Figure 23-49 Device Ready for Upload/Download Request Message

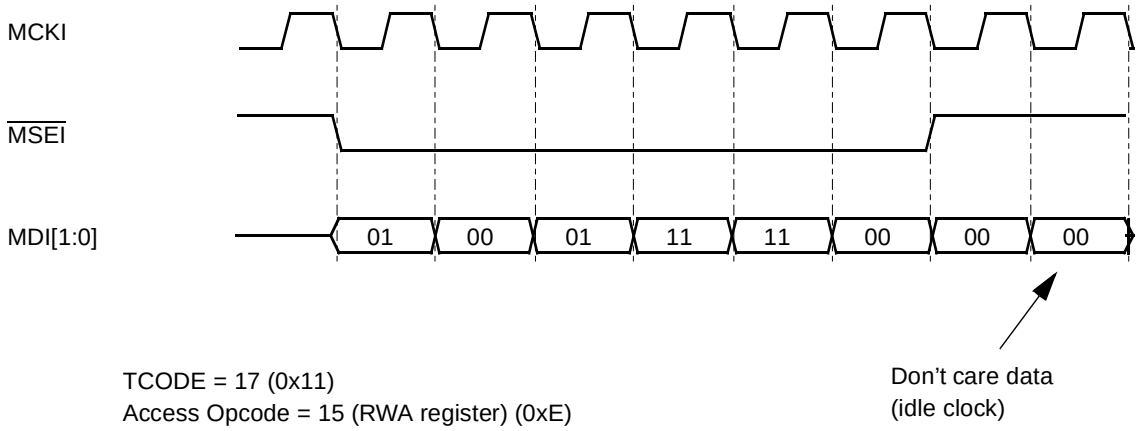
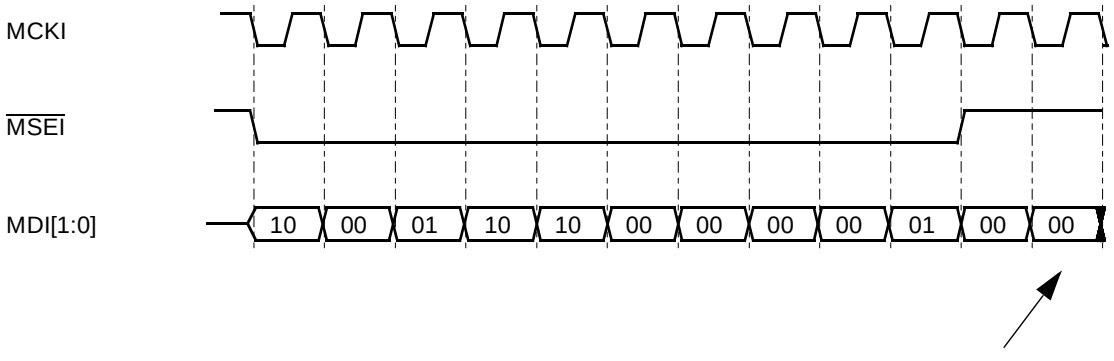


Figure 23-50 Upload Request Message

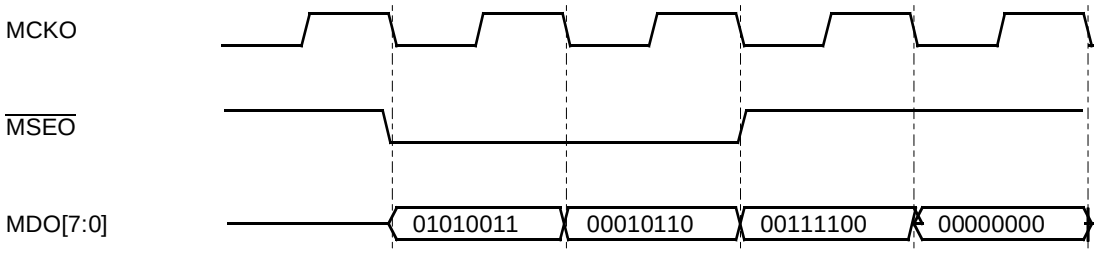


TCODE = 18 (0x12)
Access Opcode = 10 (DC register) (0xA)
Data written to DC register:

EC = 0b00
TM = 0b100
DPA = 0b0
DME = 0b0
DOR = 0b0

Don’t care data
(idle clock)

Figure 23-51 Download Request Message



TCODE = 19 (0x13)
DV =1
ERR = 0
Data Read = 0x3C16 (16 bit read access)

Don’t care data
(idle clock)

Figure 23-52 Upload/Download Information Message

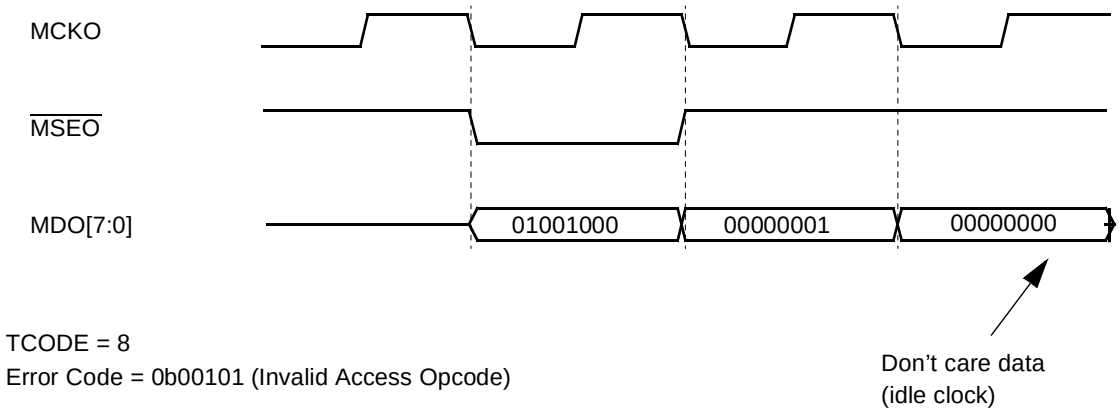


Figure 23-53 Error Message (Invalid Access Opcode)

23.8 Watchpoint Support

This section details the watchpoint support features of the READI module.

The READI module provides watchpoint messaging via the auxiliary port, as defined by the IEEE-ISTO 5001 - 1999.

READI is not compliant with all the breakpoint/watchpoint requirements defined in the IEEE-ISTO 5001 standard. Watchpoint trigger and breakpoint/watchpoint control registers are not implemented.

Watchpoint setting via READI can only be done using the BDM protocol.

23.8.1 Watchpoint Messaging

The READI module provides watchpoint messaging using IEEE-ISTO 5001 - 1999 defined public messages. The watchpoint status signals from the RCPU are snooped, and when watchpoints occur, a message is sent to the pin output formatter to be messaged out (the general message queue is bypassed to prevent watchpoint messages from being cancelled in the event of a queue overflow). The watchpoint message has the second highest priority. Refer to **23.3.2.1 Message Priority** for further details on message priorities. The watchpoint message contains the watchpoint code which indicates all the unique watchpoints have occurred since the last watchpoint message. If duplicate watchpoints occur before the watchpoint message is sent out, a watchpoint overrun message is generated. The watchpoint source field will indicate which watchpoints occurred.

The watchpoint message has the following format:

[6 bits]	[6 bits]
TCODE (15)	Watchpoint Source

Length = 12 bits

Figure 23-54 Watchpoint Message Format

23.8.1.1 Watchpoint Source Field

The Watchpoint Source field is outlined in **Table 23-34**.

Table 23-34 Watchpoint Source

Watchpoint Source	Description
0bXXXXX1	First L-bus watchpoint (LW0)
0bXXXX1X	Second L-bus watchpoint (LW1)
0bXXX1XX	First I-bus watchpoint (IW0)
0bXX1XXX	Second I-bus watchpoint (IW1)
0bX1XXXX	Third I-bus watchpoint (IW2)
0b1XXXXX	Fourth I-bus watchpoint (IW3)

23.8.2 Error Message (Watchpoint Overrun)

A watchpoint overrun error occurs when the same watchpoint occurs multiple times before the first occurrence of that watchpoint has been messaged out. The watchpoint message (which has information of all the watchpoints that occurred prior to the detection of the same watchpoint occurring multiple times) will be sent before the error message can be sent.

The overrun error causes further watchpoint occurrences to be ignored, until the error message has been sent. The error code within the error message indicates that a watchpoint overrun error has occurred.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00110)

Length = 11 bits

Figure 23-55 Error Message (Watchpoint Overrun) Format

23.8.3 Synchronization

Upon occurrence of a watchpoint, the next program and data trace message will be a synchronization message (provided program and data trace are enabled).

23.8.4 Timing Diagrams

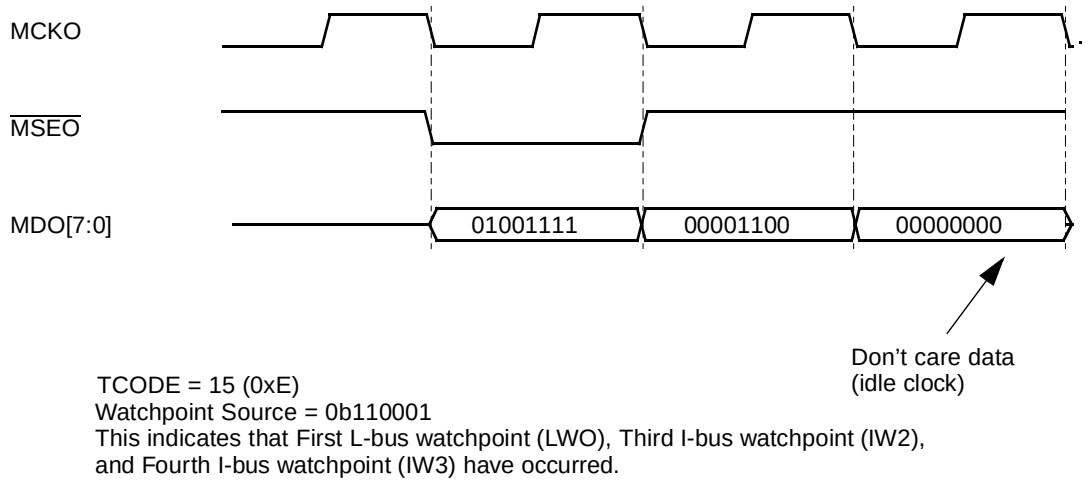


Figure 23-56 Watchpoint Message

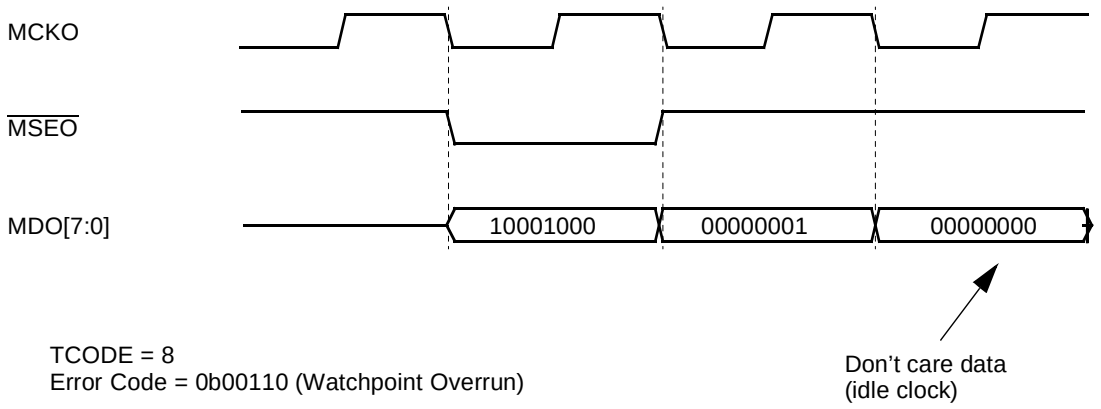


Figure 23-57 Error Message (Watchpoint Overrun)

23.9 Ownership Trace

This section details the ownership trace support features of the READI module.

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the

highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

23.9.1 Ownership Trace Messaging

Ownership trace information is messaged via the auxiliary port using an ownership trace message (OTM). The ownership trace register (OT), which can be accessed via auxiliary port, is updated by the operating system software to provide task/process ID information. When new information is updated in the register by the embedded processor, it is messaged out via the auxiliary port, allowing development tools to trace ownership flow.

Ownership trace information is messaged out in the following format:

[6 bits]	[32 bits]
TCODE (2)	Task/Process ID Tag

Length = 38 bits

Figure 23-58 Ownership Trace Message Format

23.9.2 Error Message (Queue Overflow)

A program/data/ownership trace overrun error occurs when a trace message cannot be queued due to the queue being full, provided ownership trace is enabled.

The overrun error causes the message queue to be flushed, and a error message to be queued. The error code within the error message indicates that a program/data/ownership trace overrun error has occurred.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00111)

Length = 11 bits

Figure 23-59 Error Message Format

23.9.2.1 OTM Flow

Ownership trace messages are generated when the operating system (privileged supervisor task) writes to the memory mapped ownership trace register.

The following flow describes the OTM process.

1. The OT register is a memory mapped register, whose address is located in the UBA. The OT register address can be read from the UBA register by the IEEE-ISTO 5001 tool.
2. Only privileged writes (byte/half word or word) initiated by the RCPU to the OT register that terminate normally are valid. The data value (word) written into the register is formed into the ownership trace message that is queued to be transmitted.
3. OT register reads and non-privileged OT register writes, or writes initiated by any source other than the RCPU, do not cause ownership trace messages to be transmitted by the READI module.

23.9.2.2 OTM Queueing

READI implements a 16-message-deep queue for program trace, data trace, and ownership trace messages. Messages that enter the queue are transmitted via the output auxiliary port in the order in which they are queued.

NOTE

If multiple trace messages need to be queued at the same time, ownership trace messages will have the lowest priority.

23.9.3 Timing Diagram

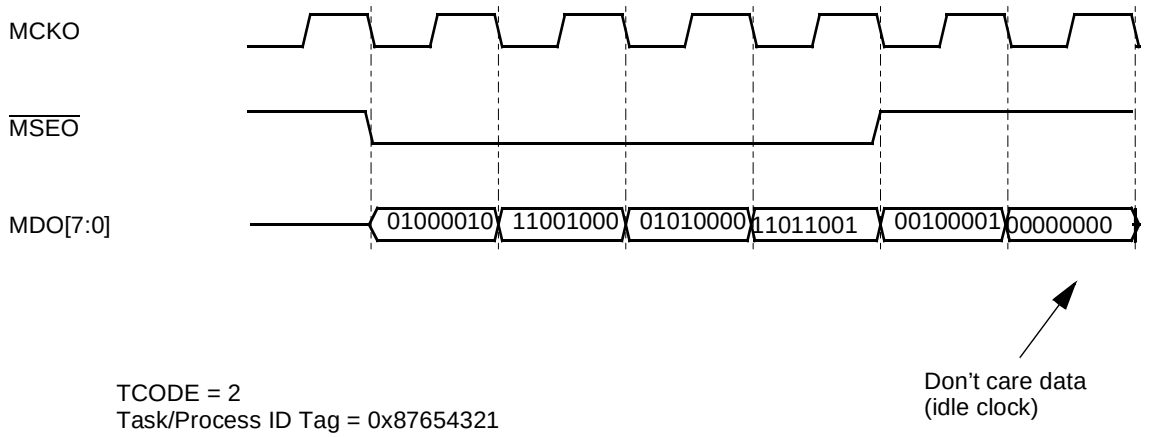


Figure 23-60 Ownership Trace Message

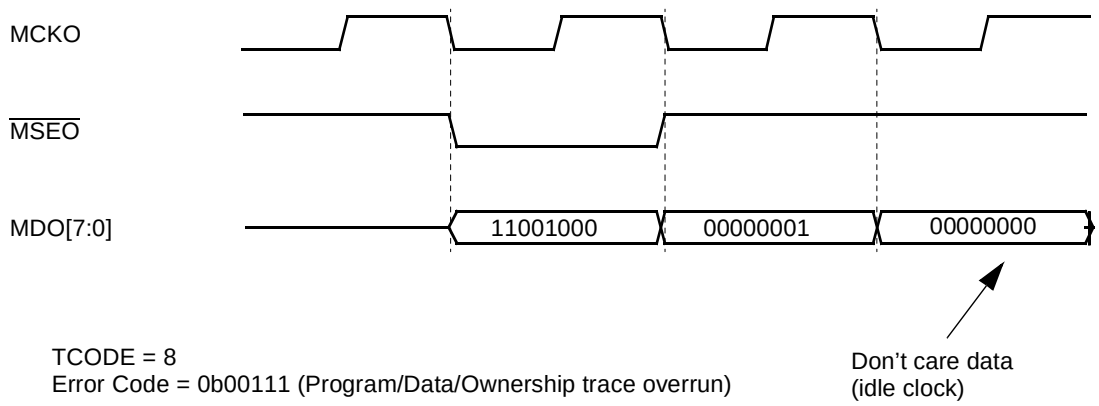


Figure 23-61 Error Message (Program/Data/Ownership Trace Overrun)

23.10 RCPU Development Access

This section details RCPU development access support features of the READI module.

The READI development port provides a full duplex serial interface for accessing existing RCPU user register and development features including BDM (background debug mode).

For further details on RCPU development function and usage, consult **SECTION 22 DEVELOPMENT SUPPORT** or the Development Port chapter of the ***RCPU Reference Manual* (RCPURM/AD)**.

RCPU development access can be achieved either via the READI pins or the BDM pins on the MCU. The access method is determined during READI module configuration. **Figure 23-62** shows how READI and BDM pins are multiplexed for RCPU development access.

When the READI module is configured for RCPU development access, IEEE-ISTO 5001 compliant vendor-defined messages are used for transmission of data in and out of the MCU.

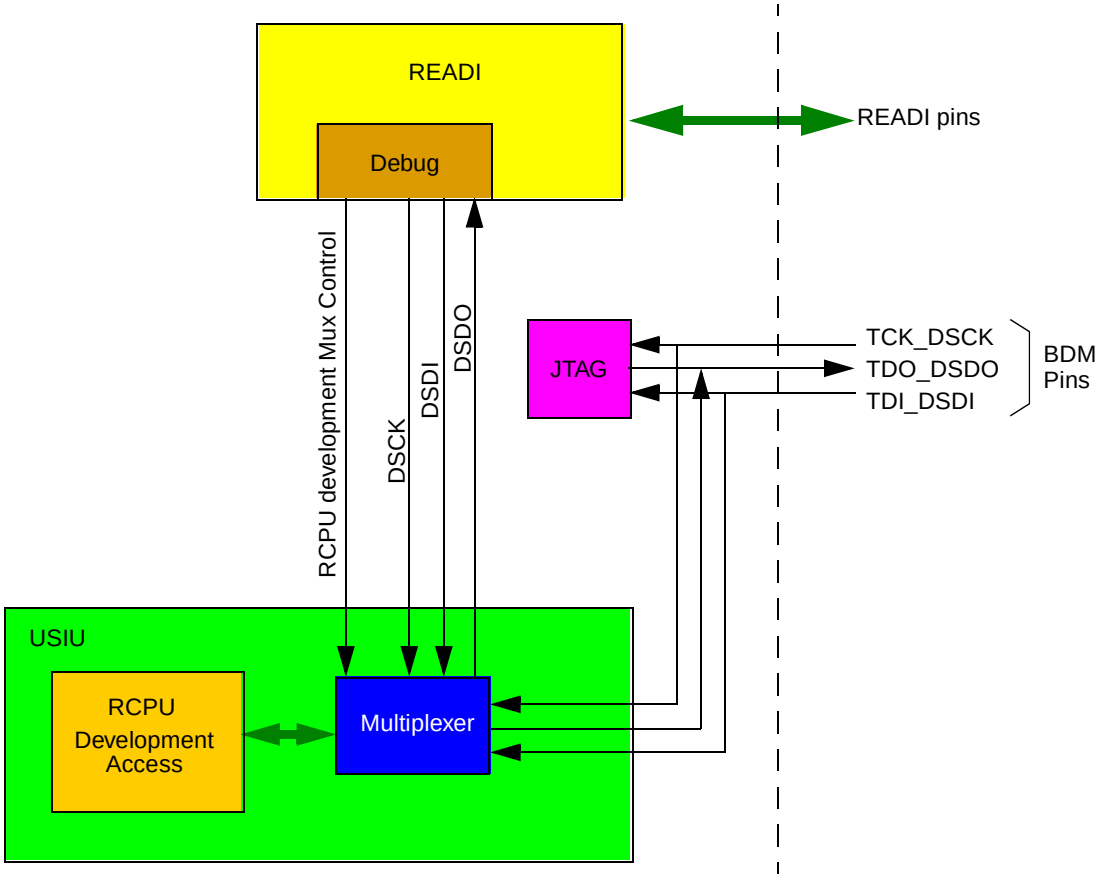


Figure 23-62 RCPU Development Access Multiplexing Between READI and BDM Pins

23.10.1 RCPU Development Access Messaging

The following RCPU development access messages are used for handshaking between the device and the tool — DSDI data message, DSDO data message and BDM status message.

23.10.1.1 DSDI Message

The DSDI message is used by the tool to download information to the RCPU.

The DSDI data field has a 3-bit status header followed by 7 or 32 bits of data/instruction, depending on the RCPU development port mode.

The DSDI message has the following format:

[6 bits]	[10 or 35 bits]
TCODE (56)	DSDI data

Max Length = 41 bits
Min Length = 16 bits

Figure 23-63 DSDI Message Format

NOTE

When sending in a DSDI data message, the DSDI data should contain the control and status bits (*start, mode, control*), followed by the 7 or 32-bit CPU instruction/data or trap enable, **MSBit first**. See **Figure 23-69** for DSDI data message transmission sequence.

23.10.1.2 DSDO Message

The DSDO message is used by the device to upload information from the RCPU.

The DSDO data field has a 3-bit status header followed by 7 or 32 bits of data/instruction, depending on the RCPU development port mode.

The three status bits in the DSDO data indicates if the device is ready to receive the next message from the tool.

The DSDO message has the following format:

[6 bits]	[10 or 35 bits]
TCODE (57)	DSDO data

Max Length = 41 bits
Min Length = 16 bits

Figure 23-64 DSDO Message Format

NOTE

The DSDO data received will contain the control and status bits and data from the CPU, **MSbit first**. See **Figure 23-69** for DSDO data message transmission sequence.

23.10.1.3 BDM Status Message

BDM status message is generated by the device to let the tool know about the status of debug mode.

BDM status message (with BDM status field equal to 0b1) is sent when the RCPU is in debug mode and the device is ready to receive debug mode messages.

BDM status message (with BDM status field equal to 0b0) is sent out when the device exits BDM mode and RCPU is in normal operating mode.

The BDM status message has the following format:

[6 bits]	[1 bit]
TCODE (58)	BDM Status

Length = 7 bits

Figure 23-65 BDM Status Message Format

23.10.1.4 Error Message (Invalid Message)

An error message is sent out when an invalid message is received by READI. The error code within the error message indicates that an invalid TCODE was detected in the auxiliary input messages by the pin input formatter.

The error message has the following format:

[6 bits]	[5 bits]
TCODE (8)	Error Code (0b00100)

Length = 11 bits

Figure 23-66 Error Message (Invalid Message) Format

23.10.2 RCPU Development Access Operation

The RCPU development access can be achieved either via the READI pins or the BDM pins. The default access is via the BDM pins.

To enable RCPU development access via the READI pins, the tool has to configure the DC register during the READI reset ($\overline{\text{RSTI}}$). Once the READI module takes the control of RCPU development access, the protocol for transmission of development serial data in (DSDI) and out (DSDO) is performed through the IEEE-ISTO 5001-1999 compliant vendor-defined messages.

After enabling RCPU development access via the READI pins, the READI module can enable debug mode and enter debug mode. When debug mode is enabled and entered, READI sends a BDM status message (BDM status field equal to 0b1) to the development tool indicating that the RCPU has entered debug mode and is now expecting instructions from the READI pins.

The development tool then uses the DSDI Data Message to send in the serial transmission data to READI. Data is transmitted to the tool using the DSDO data message.

This process continues until the RCPU exits debug mode and READI sends the BDM status message (BDM status field equal to 0b0) indicating debug mode exit.

NOTE

Only after the DSDO data message is sent out should another DSDI data message be sent in.

Synchronous self-clocked mode is selected by READI for RCPU development access. In this mode, the internal transmission between READI and the USIU is performed at system frequency.

When the RCPU is in debug mode, program trace is not allowed. If program trace is enabled, a program trace synchronization message is generated when debug mode exits.

When the RCPU is in debug mode, data trace and R/W access are allowed.

The flow chart in **Figure 23-67** shows RCPU development access configuration via READI. The modes of RCPU development access via READI are described below. Allowed modes are also summarized in **Table 23-9** of **23.2.1.5 RCPU Development Access Modes**.

23.10.2.1 Enabling RCPU Development Access Via READI Pins

Reset sequencing is done by the tool to initialize the READI pins and registers by asserting $\overline{\text{RSTI}}$ (the device sends out the device ID message after the $\overline{\text{RSTI}}$ negation). System reset is held by the tool until the READI module is reset and initialized with desired RCPU development access setting.

For RCPU development access to be enabled via the READI pins, the tool has to configure the DC register (DPA field equal to 0b1) after the negation of $\overline{\text{RSTI}}$, but before the negation of system reset. System reset should only be negated at least 16 system clocks after the DC register has been configured.

If the DC register is not configured such that READI module has control of the RCPU development access signals before the negation of the system reset, then RCPU development access is via debug mode pins. This is the default setting (DPA=0b0, DME=0bX, DOR=0bX in DC register).

NOTE

The READI module will ignore any incoming DSDI data messages when the module is not configured for RCPU development access.

23.10.2.2 Enabling Background Debug Mode (BDM) via READI Pins

After RCPU development access has been enabled via the READI pins, debug mode is enabled by setting bits in the DC register (DPA=0b1, DME=0b1, DOR=0b0).

23.10.2.3 Entering Background Debug Mode (BDM) Via READI Pins

There are three ways to enter debug mode (provided debug mode has been enabled):

1. Enter debug mode (halted state) out-of-system reset through READI module configuration. This is displayed in **Figure 23-68**.
2. Enter debug mode by downloading breakpoint instructions through RCPU development access when in non-debug (running) mode.
3. Enter debug mode if an exception or interrupt occurs.

When entering debug mode following an exception/breakpoint, the RCPU signals VFLS[0:1] are equal to 0b11. This causes READI to send a BDM status message to the tool indicating that the RCPU has entered debug mode and is now expecting instructions from the READI pins.

Debug mode enabling through READI and entering debug mode out of system reset is done by setting the following bits in the DC register (DPA=0b1, DME=0b1, DOR=0b1) during system reset. Debug mode entry causes RCPU to halt.

23.10.2.4 Non-Debug Mode Access of RCPU Development Access

The RCPU development access can be also be used while the RCPU is not halted (in debug mode). This feature is used to send in breakpoints or synchronization events to the RCPU. Please refer to the ***RCPU Reference Manual* (RCPURM/AD)** for further details.

Non-debug mode access of RCPU development can be achieved by configuring the READI module to take control of RCPU development access during module configuration of the DC register (DPA=0b1, DME=0b0, DOR=0bx).

23.10.2.5 RCPU Development Access Flow Diagram

Figure 23-67 has flow diagram describing how the RCPU development access can be achieved via READI pins.

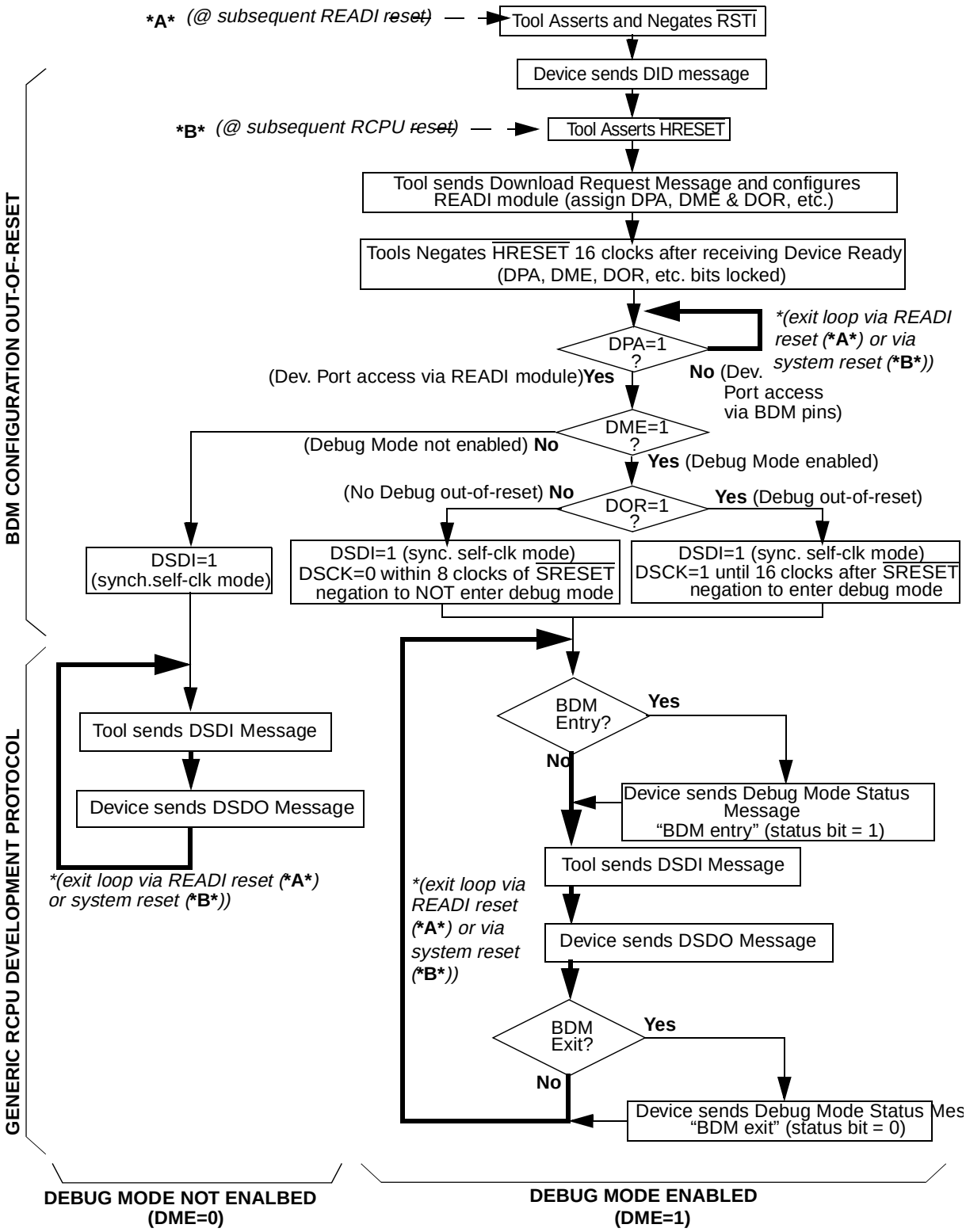


Figure 23-67 RCPU Development Access Flow Diagram

23.10.3 Throughput

The tool can send a DSDI data message into device upon the receipt of a DSDO data message as soon as the tool decodes the first two status bits of the DSDO data message just received and confirms valid data from the RCPU.

An example throughput analysis is performed with the following assumptions:

- READI configuration of RCPU development access and debug mode is already entered through READI
- The module is configured for reduced port mode
- MCKI running at 28 MHz
- MCKO running at 56 MHz
- 56-MHz internal operation
- READI auxiliary input and output pins are free (not in middle of transmission)
- No delay from tool in responding — tool keeps up with READI port
- Tool reads the complete DSDO data message before shifting in DSDI data message
- 10 clocks estimated to format and encode/decode DSDI data and DSDO data messages within READI

The DSDI Data message is 41 bits (six bits of TCODE and 35 bits of DSDI data data). It takes 41 clocks (41 bits / 1 MDI pins) to shift in the DSDI data message. It is estimated that READI will take approximately 10 clocks to decode the DSDI data message. After the message has been decoded, READI will take 35 clocks to serially shift in the 35 bits of DSDI data to the RCPU development port. Hence, it takes a total of 86 clocks (41 + 10 + 35) to decode and shift in DSDI data from the tool to the RCPU development port.

At 28 MHz, it translates to 3079 ns (35.8 x 81) to decode and shift in DSDI data to RCPU development port

As DSDI bits are shifted into the RCPU development register, DSDO bits are shifted out from the same RCPU development register (DPDR) and these are captured by READI.

It is estimated that READI will take approximately 10 clocks to encode the DSDO data. The DSDO message is 41 bits (6 bits of TCODE and 35 bits of DSDO data). It will take 21 clocks (41 bits / 2 MDO pins) for READI to transmit this message. Hence, it will take a total of 31 clocks (10 + 21) to encode the DSDO data message and shift out the DSDO data message to the tool.

At 56 MHz, it will take 552 ns (17.8 x 31) to encode and shift out DSDO data to the tool.

Thus, it will take 3631 ns (3079 + 552) for one complete DSDI data and DSDO data messaging cycle.

23.10.4 Timing Diagrams

Figure 23-68 shows the timing diagram of RCPU development access and entering debug mode out-of-system reset through READI.

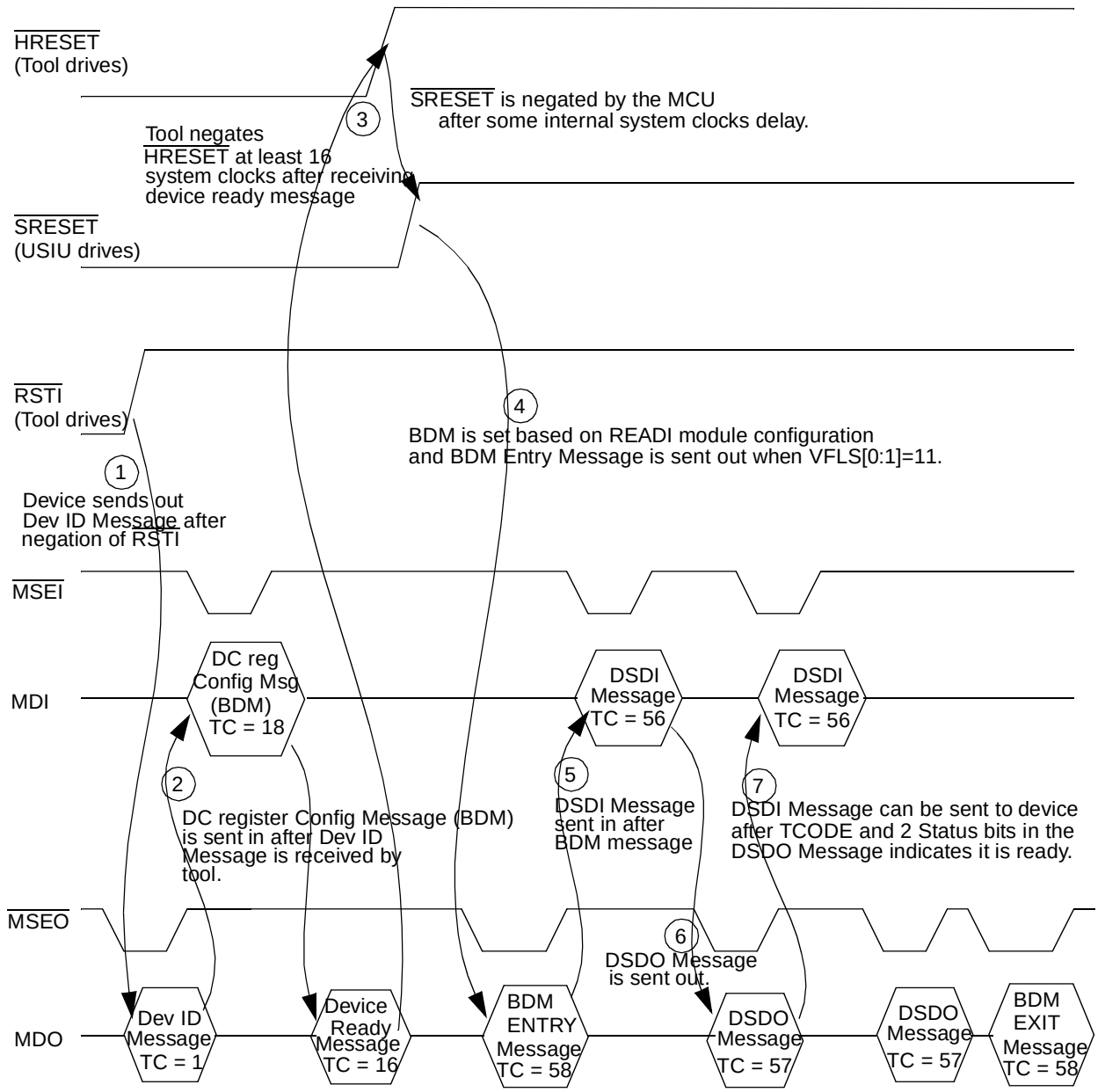


Figure 23-68 RCPU Development Access Timing Diagram — Debug Mode Entry Out-of-Reset

Figure 23-69 shows the transmission sequence of DSDI/DSDO data messages.

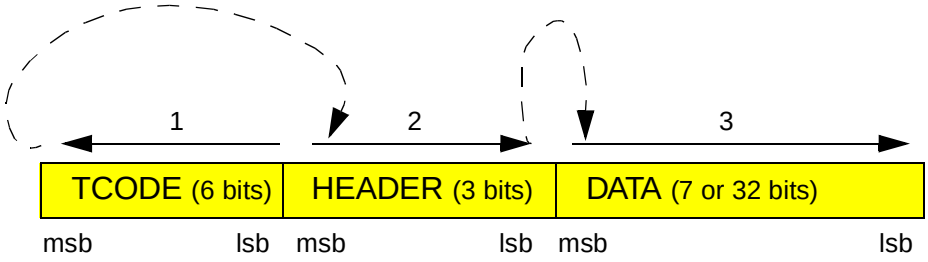


Figure 23-69 Transmission Sequence of DSD*x* Data Messages

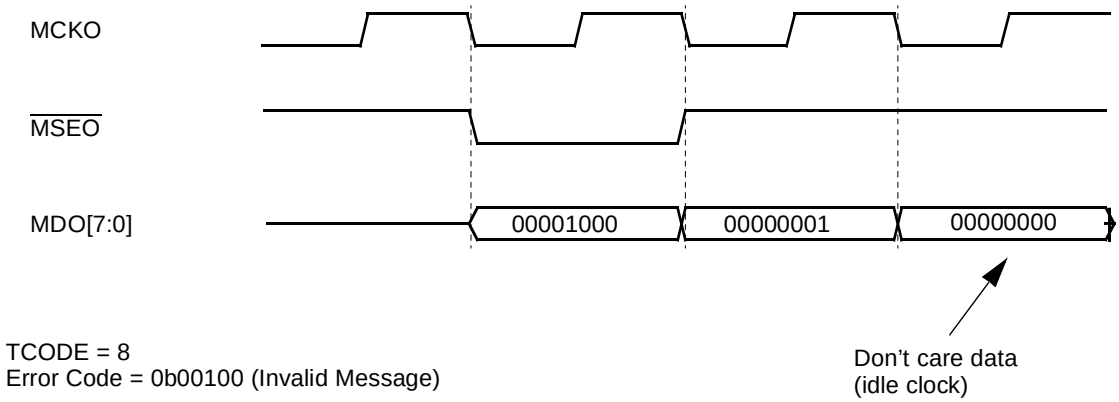


Figure 23-70 Error Message (Invalid Message)

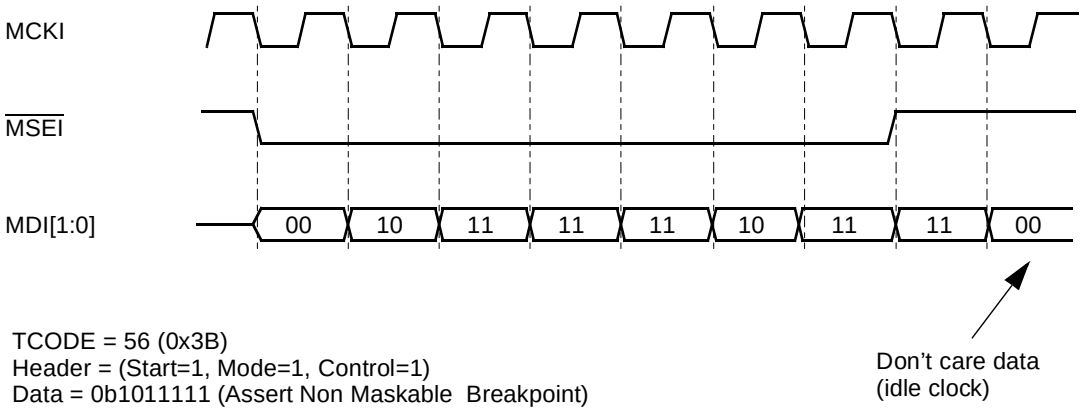


Figure 23-71 DSDI Data Message (Assert Non-Maskable Breakpoint)

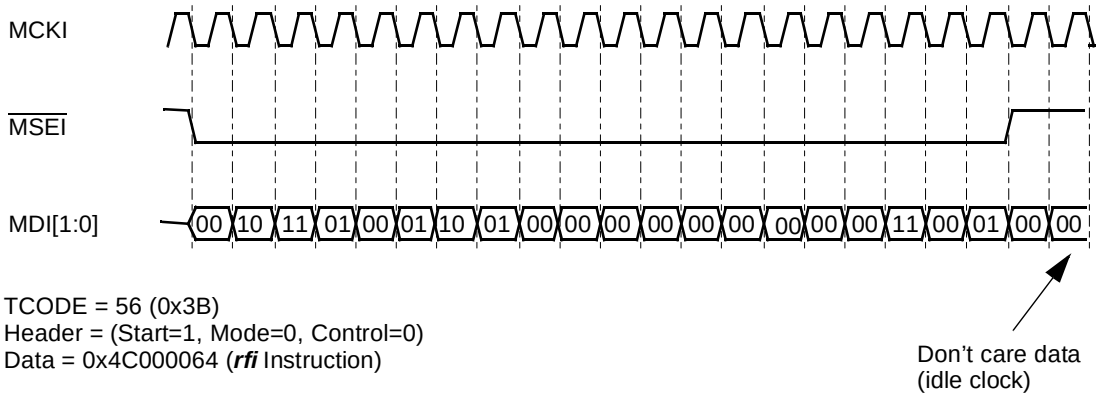


Figure 23-72 DSDI Data Message (CPU Instruction — *rfi*)

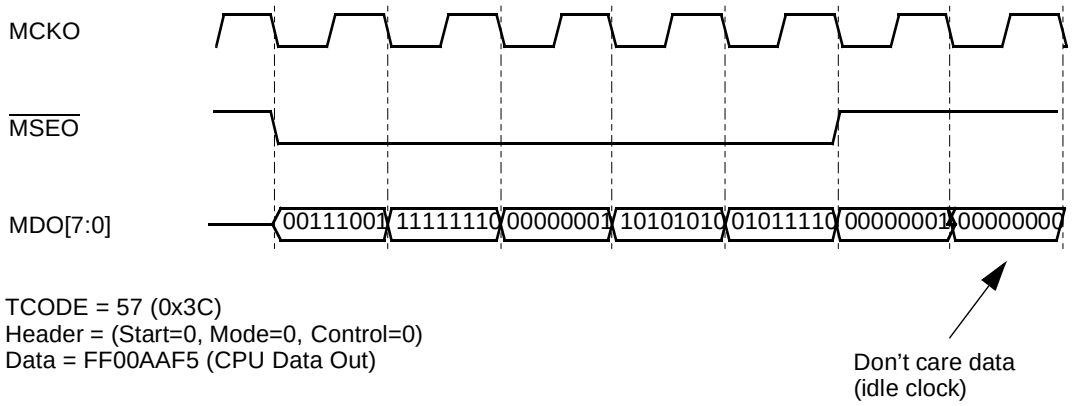


Figure 23-73 DSDO Data Message (CPU Data Out)

23.11 Power Management

This section details the power management features of the READI module.

The READI module is a development interface, and is not expected to function under normal (non-development) conditions. Therefore power management is required to reduce and minimize power consumption during normal operation of the part.

23.11.1 Functional Description

The following are the candidates for power management:

Table 23-35 Power Management Mechanism Overview

Feature	Power Saving Mechanism
Disabled Mode	If $\overline{\text{EVTI}}$ is negated at negation of $\overline{\text{RSTI}}$, the READI module will be disabled. No trace output will be provided, and output auxiliary port will be three-stated.
Sleep, Deep-Sleep and Low Power-Down Mode	All outputs will be held static.
READI Reset ($\overline{\text{RSTI}}$)	Output auxiliary pins will be three-stated.

23.11.2 Low Power Modes

When the MCU is in sleep, deep-sleep or low power-down mode, all internal clocks on the MCU are shut down, including the MCKO. The $\overline{\text{MSEO}}$ pin will be held negated.

Low power mode entry for the MCU will be held off until the READI module has transmitted all existing messages (in the queues and transmit buffers). During this time, input messages from the development tool are ignored.

Upon restoration of clocks in normal mode, program and data traces will be synchronized, if enabled.

23.12 Application Notes

This section describes application notes for READI.

23.12.1 Automotive Calibration

READI has features that can be used for calibration tasks during automotive system development. The basic needs for automotive development tools are

1. To acquire, while running an engine or vehicle, crank shaft synchronous data relating to calibration factors as they are being used or modified during high speed transient events, with acceptable impact to the system under development.
2. To acquire, while running an engine or vehicle, time synchronous data relating to calibration factors as they are being used or modified during high speed transient events, with acceptable impact to the system under development.
3. To coherently modify table(s) of calibration constant with the engine control unit is running an engine or vehicle.

23.12.1.1 Calibration Variable Acquisition

Calibration variable acquisition refers to the first two points listed in the basic needs for automotive development tools. READI supports calibration variable acquisition via data trace (read and write) messaging and read/write access.

Requirements for measurement:

- Time synchronous as well as engine synchronous
- Selection of values must be flexible

- Data to be measured (minimum)
  — 29 values engine speed synchronous
  — 45 values 10 ms synchronous
  — 45 values 100 ms synchronous
- Minimal interrupt load
- Minimal impact to system

23.12.2 Calibration Variables Located In Contiguous Memory Locations

23.12.2.1 Data Read Messaging

Calibration variables located in two contiguous memory (RAM) locations can be acquired using data read messaging. The software routines must read these variables for them to be traced and messaged via the READI port.

READI queue sizes and READI port throughput will require few clocks between the individual ‘reads’ of the variables, otherwise information will be lost.

DRM is accomplished by snooping the L-bus hence there is no impact to the RCPU performance.

23.12.2.2 Read/Write Access

In case the throughput requirements for data traces are not acceptable, block read/write access should be used to load the variables from the calibration memory.

This method may be more preferable because it is more likely to be synchronized and coherent data for multiple task scenarios.

The disadvantage of using read/write access is the impact it has on RCPU performance. Block word read accesses can be done on the L-bus every 13 clocks. If the block word read accesses happen to be overlapped with a block access via the RCPU, the maximum intrusion will be 7.7%.

23.12.3 Calibration Variables Not Located in Contiguous Memory Locations

23.12.3.1 Data Write Messaging

Option A:

If calibration variables are not located in contiguous memory locations, then a contiguous section of the RAM can be reserved for variable acquisition tracing. At engine/time synchronous points, the software can write these contiguous locations with the values of the variables. This activity will be traced and messaged out data write messaging.

This method requires a ‘shadow’ location for each word being monitored.

A load and a store will be required for each variable to update the shadow locations, which are traced.

READI queue sizes and IEEE-ISTO 5001 port throughput will require few clocks between the individual ‘writes’ of the variables. This could be achieved by adding a few ‘nop’ instructions.

Option B:

Another possibility is to allocate a (single/few) word(s) in the RAM that are configured to be traced by the READI module. Then at engine/time synchronous points, the software will write the values of a SERIES of variables to a corresponding location. This activity will be traced and messaged out data write messaging.

This method requires a ‘shadow’ location, for each SERIES (of related) variables or channels, in the RAM.

A load and a store will be required for each variable to update the shadow locations, which are traced. The 1st write to the location could be a byte which indicates the channel number. The calibration tool then will not be required to know the address translation for channel information.

The advantage of using this option is that fewer locations need to be reserved in memory for measurement and the data trace messages are smaller, since a whole table (45 values) may be written to the same location.

The disadvantage of using this option is that the calibration tool needs to be aware of which series a particular location represents, and the order in which the values are written.

READI queue sizes and READI port throughput will require few clocks between the individual ‘writes’ of the variables. This could be achieved by adding a few ‘nop’ instructions.

23.12.3.2 Read/Write Access

In case the throughput requirements for block data traces are not acceptable, read/write access should be used to load the variables from the calibration memory.

Option A: Block Read Access

This method calls for setting aside some memory (~ 1-Kbyte RAM locations) for calibration variable measurement. When the software tasks that modify the variables are completed, they will copy the variables into contiguous locations in this reserved space. Then the tasks will indicate to the external tool that the variables are valid by any of various means (write keyword to special address, watchpoint etc.). The calibration tool will then proceed to read the variables corresponding to the channel completed using the block read access feature.

This method may be more preferable because it is more likely to be synchronized and coherent data for multiple task scenarios.

The disadvantage of using block read access is the impact it has on RCPU performance. Block word read accesses can be done on the L-bus every 13 clocks. If the

block word read accesses happen to be overlapped with a block access via the RCPU, the maximum intrusion will be 7.7%.

Option B: Single Read Access

Single read accesses can also be used to acquire calibration variables which are not located in contiguous memory locations.

Single read accesses also have an the impact on RCPU performance. Single word read accesses can be done on the L-bus every 74 clocks. If single word read accesses happen to be overlapped with a block access via the RCPU, the maximum intrusion will be 1.3%. This is significantly better when compared to block read accesses (7.7%).

The disadvantage of using single read access the data trace throughput. For single read accesses, 0.41 million words can be messaged out via the READI port, whereas for block read accesses, approximately 4 million words can be messaged out via the READI port.

23.12.3.3 Calibration Constant Tuning

Calibration constant tuning refers to the third basic requirement for automotive development tools. READI supports calibration constant tuning via the read/write access feature.

The read/write access protocol provides run-time access to MCU registers and memory map. READI implements the run-time read/write access feature via two MDI pins and four MDO pins.

Read/write accesses can be either single or block write accesses. If the calibration constants that need to be modified are located in contiguous memory locations, the block write access feature should be used.

The disadvantage of using read/write access is the impact it has on RCPU performance. Block write accesses can be done on the L-bus every 26 clocks. If the block write happens to be overlapped with a block access via the RCPU, the maximum intrusion will be 3.9%.

23.12.4 UC3F Programming Guideline Flowchart via READI Read/Write Access

The user can use READI features such as read/write access or RCPU development access to program the UC3F. For details on how the UC3F can be programmed via READI read/write access, refer to **Figure 23-74** and **Figure 23-75**.

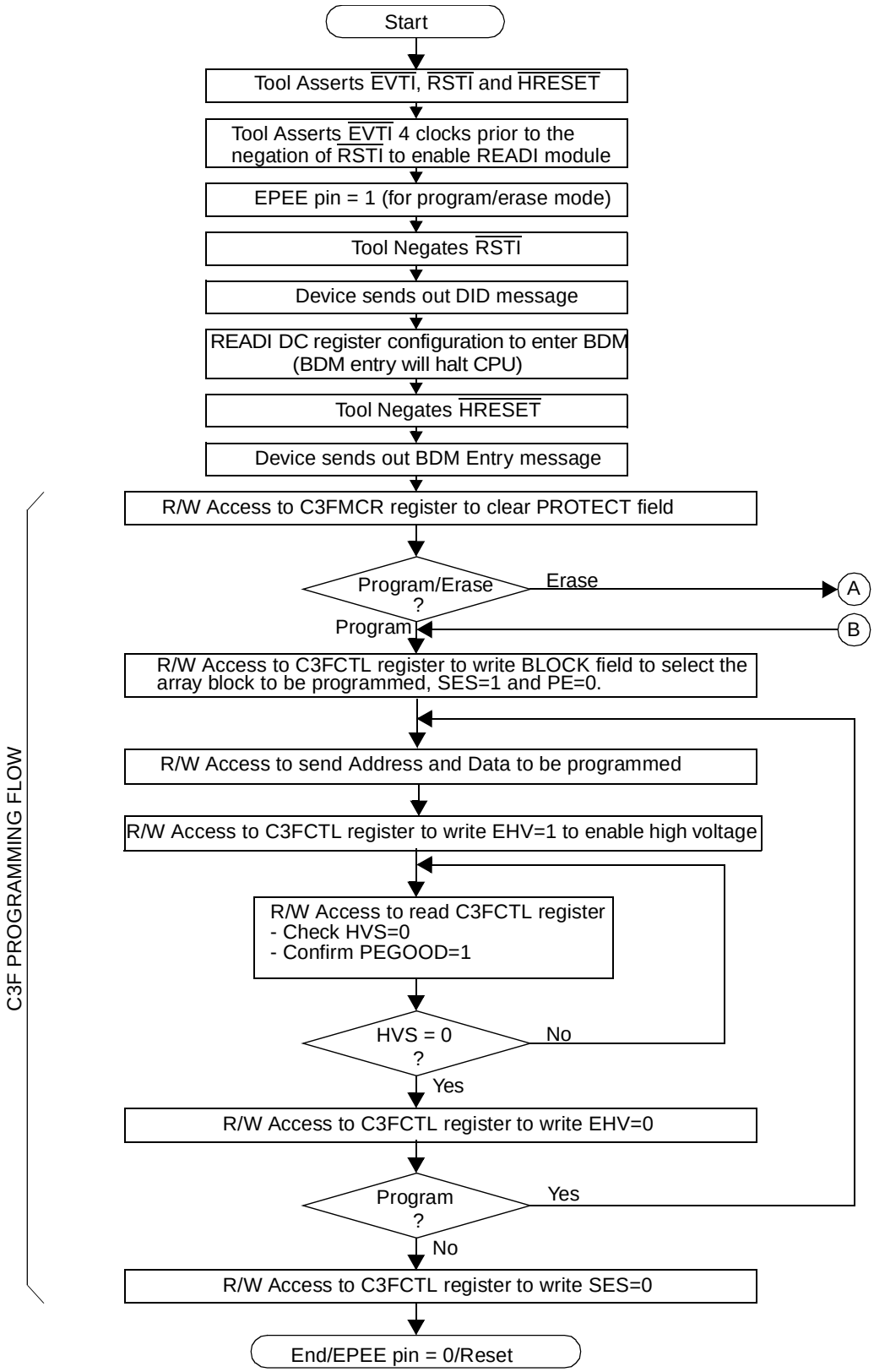


Figure 23-74 Programming UC3F Flash via READI R/W Access (Sheet 1 of 2)

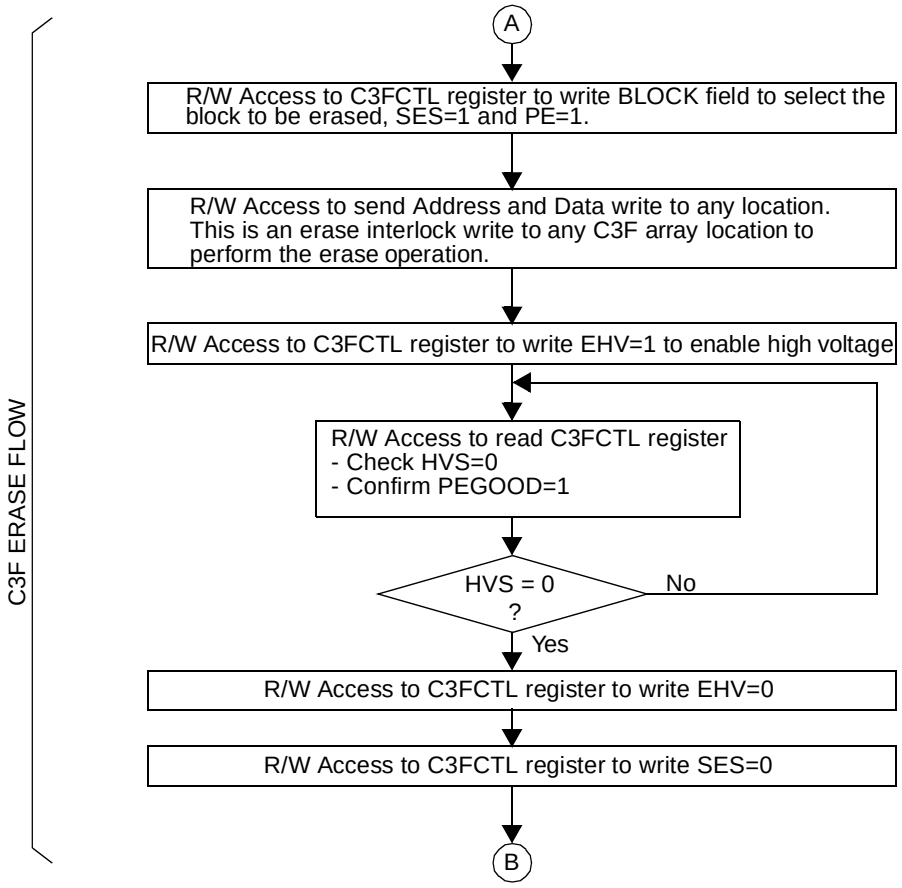


Figure 23-75 Programming UC3F Flash via READI R/W Access (Sheet 2 of 2)

SECTION 24
IEEE 1149.1-COMPLIANT INTERFACE (JTAG)

24.1 IEEE 1149.1 Test Access Port (TAP) and Joint Test Action Group (JTAG)

The chip design includes user-accessible test logic that is compatible with *the IEEE 1149.1-1994 Standard Test Access Port and Boundary Scan Architecture*.The implementation supports circuit-board test strategies based on this standard. An overview of the pins requirement on JTAG is shown:

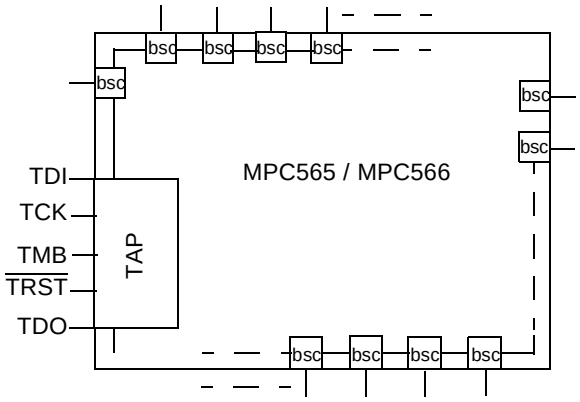


Figure 24-1 Pin Requirement on JTAG

24.2 IEEE 1149.1 Test Access Port

The MPC565 / MPC566 provides a dedicated user-accessible test access port (TAP) that is compatible with the *IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture* in all but 2 areas listed below. Problems associated with testing high density circuit boards have led to development of this proposed standard under the sponsorship of the Test Technology Committee of IEEE and the Joint Test Action Group (JTAG). The MPC565 / MPC566 implementation supports circuit-board test strategies based on this standard.

IEEE1149.1 Compatibility Exceptions

- The MPC565 enters JTAG mode by going through a standard device reset sequence with a specific configuration word applied to the databus. Once JTAG has been entered, the MPC565 remains in JTAG mode until another reset sequence is applied to exit JTAG mode, or the device is powered down.
- The JTAG output port, "tdo_dsdo" is configured with a weak pullup when its output drivers are disabled, rather than being tri-stated.

The TAP consists of five dedicated signal pins, a 16-state TAP controller, and two test data registers. A boundary scan register links all device signal pins into a single shift register. The test logic implemented utilizes static logic design. The MPC565 / MPC566 implementation provides the capability to:

1. Perform boundary scan operations to test circuit-board electrical continuity.
2. Bypass the MPC565 / MPC566 for a given circuit-board test by effectively reducing the boundary scan register to a single cell.
3. Sample the MPC565 / MPC566 system pins during operation and transparently shift out the result in the boundary scan register.
4. Disable the output drive to pins during circuit-board testing.

NOTE

Certain precautions must be observed to ensure that the IEEE 1149.-like test logic does not interfere with nontest operation. See **24.3.1 Non-Scan Chain Operation** for details.

24.2.1 Overview

An overview of the MPC565 / MPC566 scan chain implementation is shown in . The MPC565 / MPC566 implementation includes a TAP controller, a 4-bit instruction register, and two test registers (a one-bit bypass register and a 520-bit boundary scan register). This implementation includes a dedicated TAP consisting of the following signals:

- TCK — a test clock input to synchronize the test logic. (with an internal pull-down resistor)
- TMS — a test mode select input (with an internal pullup resistor) that is sampled on the rising edge of TCK to sequence the TAP controller's state machine.
- TDI — a test data input (with an internal pullup resistor) that is sampled on the rising edge of TCK.
- TDO — a three-state test data output that is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK. (This pin also has a weak pull-up that is active when output drivers are disabled, except during a HI-Z instruction).
- $\overline{\text{TRST}}$ — an asynchronous reset with an internal pull-up resistor that provides initialization of the TAP controller and other logic required by the standard.

NOTE

JTAG mode does not provide access to the internal MPC565 / MPC566 circuitry. It allows access only to the input or output pad (periphery) circuitry

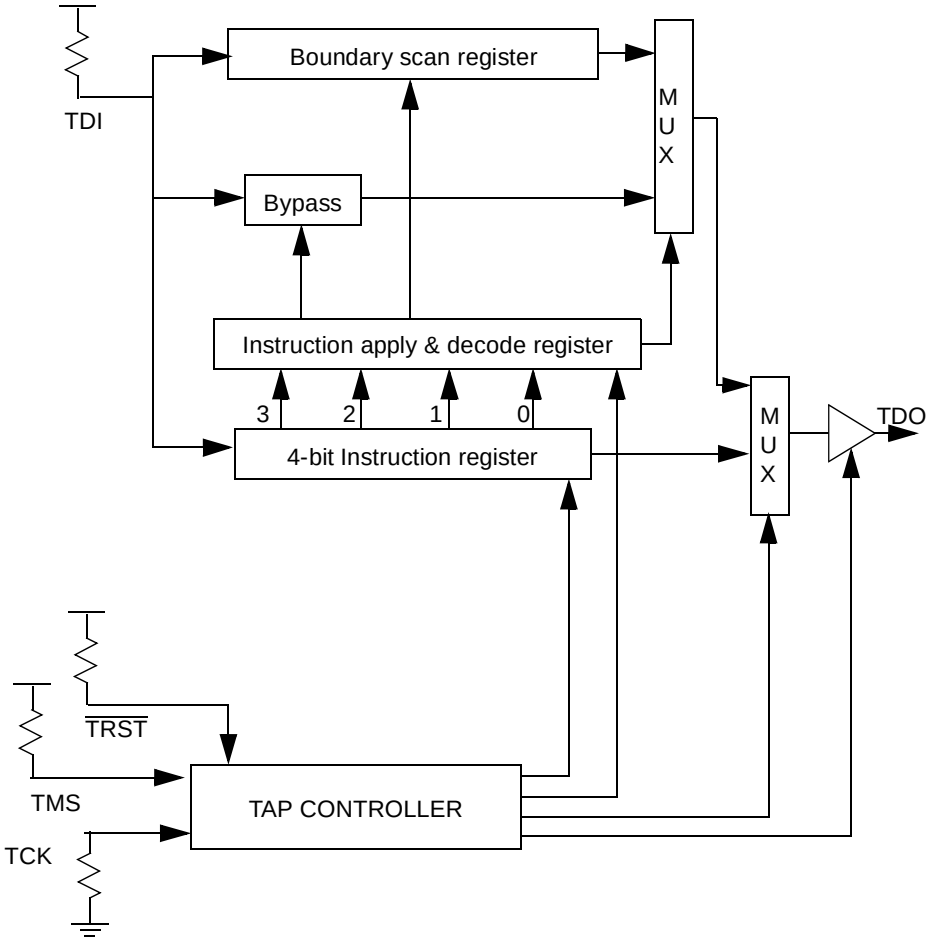


Figure 24-2 Test Logic Block Diagram

24.2.1.1 TAP CONTROLLER

The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operation of the JTAG logic. The state machine is shown in . The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of the TCK signal.

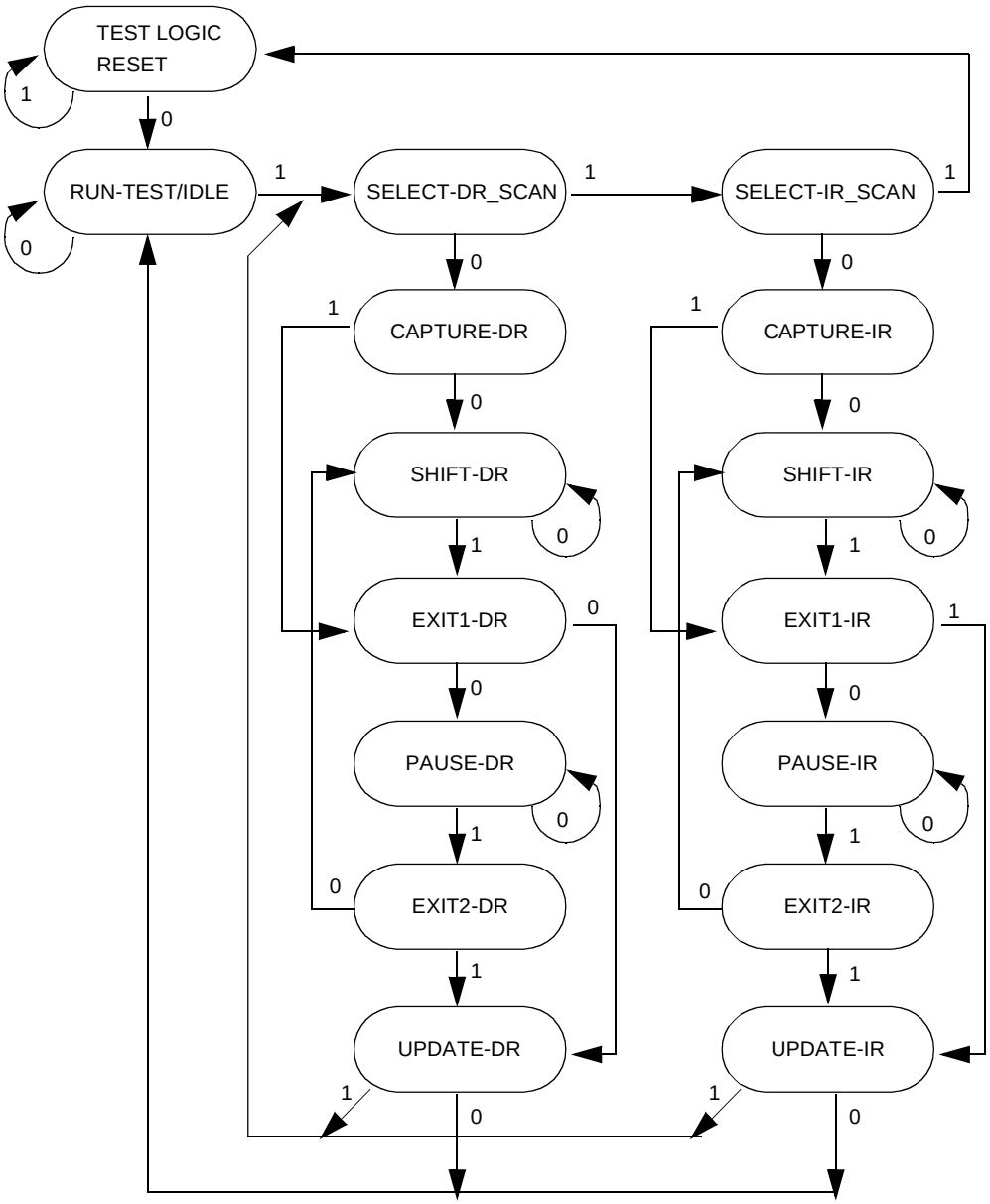


Figure 24-3 TAP Controller State Machine

24.2.1.2 Boundary Scan Register

The MPC565 / MPC566 scan chain implementation has a 520-bit boundary scan register. This register contains bits for most device signals, clock pins and associated control signals. The XTAL, EXTAL and XFC pins are associated with analog signals and are not included in the boundary scan register. The $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, and $\overline{\text{SRESET}}$ pins are also excluded from the boundary scan register.

The 520-bit boundary scan register can be connected between TDI and TDO by selecting the EXTEST or SAMPLE/PRELOAD instructions. This register is used to capturing signal pin data on the input pins, forcing fixed values on the output signal

pins, and selecting the direction and drive characteristics (a logic value or high impedance) of the bidirectional and three-state signal pins.

The key to using the boundary scan register is knowing the boundary scan bit order and the pins that are associated with them. **Table 24-1** shows the bit order starting from the TDO output and going to the TDI input.

Table 24-1 Boundary Scan Bit Definition

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
0	BC_2	*	internal	1					
1	BC_2	mdo_2	output2	1				O	26v
2	BC_2	*	internal	1					
3	BC_2	mdo_3	output2	1				O	26v
4	BC_2	*	internal	1					
5	BC_2	mseo_b	output2	1				O	26v
6	BC_2	*	internal	1					
7	BC_2	iwp0_vfls0	output2	1				O	26v
8	BC_2	*	internal	1					
9	BC_2	iwp1_vfls1	output2	1				O	26v
10	BC_2	*	controlr	0					
11	BC_7	addr_sgpioa(16)	bidir	0	10	0	Z	I/O	26v5vs
12	BC_2	*	controlr	0					
13	BC_7	addr_sgpioa(17)	bidir	0	12	0	Z	I/O	26v5vs
14	BC_2	*	controlr	0					
15	BC_7	sgpioc6_frz_ptr_b	bidir	0	14	0	Z	I/O	26v5vs
16	BC_2	*	controlr	0					
17	BC_7	addr_sgpioa(8)	bidir	0	16	0	Z	I/O	26v5vs
18	BC_2	*	controlr	0					
19	BC_7	addr_sgpioa(18)	bidir	0	18	0	Z	I/O	26v5vs
20	BC_2	*	controlr	0					
21	BC_7	addr_sgpioa(19)	bidir	0	20	0	Z	I/O	26v5vs
22	BC_2	*	controlr	0					
23	BC_7	addr_sgpioa(9)	bidir	0	22	0	Z	I/O	26v5vs
24	BC_2	*	controlr	0					
25	BC_7	addr_sgpioa(10)	bidir	0	24	0	Z	I/O	26v5vs
26	BC_2	*	controlr	0					

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
27	BC_7	addr_sgpioa(20)	bidir	0	26	0	Z	I/O	26v6vs
28	BC_2	*	controlr	0					
29	BC_7	addr_sgpioa(21)	bidir	0	28	0	Z	I/O	26v5vs
30	BC_2	*	controlr	0					
31	BC_7	addr_sgpioa(11)	bidir	0	30	0	Z	I/O	26v5vs
32	BC_2	*	controlr	0					
33	BC_7	addr_sgpioa(12)	bidir	0	32	0	Z	I/O	26v5vs
34	BC_2	*	controlr	0					
35	BC_7	addr_sgpioa(22)	bidir	0	34	0	Z	I/O	26v5vs
36	BC_2	*	controlr	0					
37	BC_7	addr_sgpioa(23)	bidir	0	36	0	Z	I/O	26v5vs
38	BC_2	*	controlr	0					
39	BC_7	addr_sgpioa(13)	bidir	0	38	0	Z	I/O	26v5vs
40	BC_2	*	controlr	0					
41	BC_7	addr_sgpioa(24)	bidir	0	40	0	Z	I/O	26v5vs
42	BC_2	*	controlr	0					
43	BC_7	addr_sgpioa(25)	bidir	0	42	0	Z	I/O	26v5vs
44	BC_2	*	controlr	0					
45	BC_7	addr_sgpioa(14)	bidir	0	44	0	Z	I/O	26v5vs
46	BC_2	*	controlr	0					
47	BC_7	addr_sgpioa(15)	bidir	0	46	0	Z	I/O	26v5vs
48	BC_2	*	controlr	0					
49	BC_7	addr_sgpioa(30)	bidir	0	48	0	Z	I/O	26v5vs
50	BC_2	*	controlr	0					
51	BC_7	addr_sgpioa(26)	bidir	0	50	0	Z	I/O	26v5vs
52	BC_2	*	controlr	0					
53	BC_7	addr_sgpioa(27)	bidir	0	52	0	Z	I/O	26v5vs
54	BC_2	*	controlr	0					
55	BC_7	addr_sgpioa(31)	bidir	0	54	0	Z	I/O	26v5vs
56	BC_2	*	controlr	0					
57	BC_7	addr_sgpioa(28)	bidir	0	56	0	Z	I/O	26v5vs
58	BC_2	*	controlr	0					

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
59	BC_7	addr_sgpioa(29)	bidir	0	58	0	Z	I/O	26v5vs
60	BC_2	*	controlr	0					
61	BC_7	data_sgpiod(0)	bidir	0	60	0	Z	I/O	26v5vs
62	BC_2	*	controlr	0					
63	BC_7	data_sgpiod(29)	bidir	0	62	0	Z	I/O	26v5vs
64	BC_2	*	controlr	0					
65	BC_7	data_sgpiod(1)	bidir	0	64	0	Z	I/O	26v5vs
66	BC_2	*	controlr	0					
67	BC_7	data_sgpiod(2)	bidir	0	66	0	Z	I/O	26v5vs
68	BC_2	*	controlr	0					
69	BC_7	data_sgpiod(3)	bidir	0	68	0	Z	I/O	26v5vs
70	BC_2	*	controlr	0					
71	BC_7	data_sgpiod(27)	bidir	0	70	0	Z	I/O	26v5vs
72	BC_2	*	controlr	0					
73	BC_7	data_sgpiod(4)	bidir	0	72	0	Z	I/O	26v5vs
74	BC_2	*	controlr	0					
75	BC_7	data_sgpiod(28)	bidir	0	74	0	Z	I/O	26v5vs
76	BC_2	*	controlr	0					
77	BC_7	data_sgpiod(31)	bidir	0	76	0	Z	I/O	26v5vs
78	BC_2	*	controlr	0					
79	BC_7	data_sgpiod(5)	bidir	0	78	0	Z	I/O	26v5vs
80	BC_2	*	controlr	0					
81	BC_7	data_sgpiod(6)	bidir	0	80	0	Z	I/O	26v5vs
82	BC_2	*	controlr	0					
83	BC_7	data_sgpiod(30)	bidir	0	82	0	Z	I/O	26v5vs
84	BC_2	*	controlr	0					
85	BC_7	data_sgpiod(7)	bidir	0	84	0	Z	I/O	26v5vs
86	BC_2	*	controlr	0					
87	BC_7	data_sgpiod(25)	bidir	0	86	0	Z	I/O	26v5vs
88	BC_2	*	controlr	0					
89	BC_7	data_sgpiod(8)	bidir	0	88	0	Z	I/O	26v5vs
90	BC_2	*	controlr	0					

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
91	BC_7	data_sgpiod(24)	bidir	0	90	0	Z	I/O	26v5vs
92	BC_2	*	controlr	0					
93	BC_7	data_sgpiod(9)	bidir	0	92	0	Z	I/O	26v5vs
94	BC_2	*	controlr	0					
95	BC_7	data_sgpiod(10)	bidir	0	94	0	Z	I/O	26v5vs
96	BC_2	*	controlr	0					
97	BC_7	data_sgpiod(26)	bidir	0	96	0	Z	I/O	26v5vs
98	BC_2	*	controlr	0					
99	BC_7	data_sgpiod(22)	bidir	0	98	0	Z	I/O	26v5vs
100	BC_2	*	controlr	0					
101	BC_7	data_sgpiod(11)	bidir	0	100	0	Z	I/O	26v5vs
102	BC_2	*	controlr	0					
103	BC_7	data_sgpiod(12)	bidir	0	102	0	Z	I/O	26v5vs
104	BC_2	*	controlr	0					
105	BC_7	data_sgpiod(13)	bidir	0	104	0	Z	I/O	26v5vs
106	BC_2	*	controlr	0					
107	BC_7	data_sgpiod(20)	bidir	0	106	0	Z	I/O	26v5vs
108	BC_2	*	controlr	0					
109	BC_7	data_sgpiod(14)	bidir	0	108	0	Z	I/O	26v5vs
110	BC_2	*	controlr	0					
111	BC_7	data_sgpiod(23)	bidir	0	110	0	Z	I/O	26v5vs
112	BC_2	*	controlr	0					
113	BC_7	data_sgpiod(15)	bidir	0	112	0	Z	I/O	26v5vs
114	BC_2	*	controlr	0					
115	BC_7	data_sgpiod(16)	bidir	0	114	0	Z	I/O	26v5vs
116	BC_2	*	controlr	0					
117	BC_7	data_sgpiod(21)	bidir	0	116	0	Z	I/O	26v5vs
118	BC_2	*	controlr	0					
119	BC_7	data_sgpiod(17)	bidir	0	118	0	Z	I/O	26v5vs
120	BC_2	*	controlr	0					
121	BC_7	data_sgpiod(18)	bidir	0	120	0	Z	I/O	26v5vs
122	BC_2	*	controlr	0					

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
123	BC_7	data_sgpiod(19)	bidir	0	122	0	Z	I/O	26v5vs
124	BC_2	*	controlr	0					
125	BC_7	irq3_b_kr_b_retry_b_sgpioc3	bidir	0	124	0	Z	I/O	26v5vs
126	BC_2	*	controlr	0					
127	BC_7	irq4_b_at2_sgpioc4	bidir	0	126	0	Z	I/O	26v5vs
128	BC_2	*	controlr	0					
129	BC_7	irq1_b_rsv_b_sgpioc1	bidir	0	128	0	Z	I/O	26v5vs
130	BC_2	*	controlr	0					
131	BC_7	sgpioc7_irqout_b_lwp0	bidir	0	130	0	Z	I/O	26v5vs
132	BC_2	*	controlr	0					
133	BC_7	bb_b_vf2_iwp3	bidir	0	132	0	Z	I/O	26v
134	BC_2	*	controlr	0					
135	BC_7	bg_b_vf0_lwp1	bidir	0	134	0	Z	I/O	26v
136	BC_2	*	controlr	0					
137	BC_7	br_b_vf1_iwp2	bidir	0	136	0	Z	I/O	26v
138	BC_2	*	controlr	0					
139	BC_7	rd_wr_b	bidir	0	138	0	Z	I/O	26v
140	BC_2	*	internal	1					
141	BC_2	oe_b	output2	1				O	26v
142	BC_2	*	controlr	0					
143	BC_7	tea_b	bidir	0	142	0	Z	O	26v
144	BC_2	*	controlr	0					
145	BC_7	irq2_b_cr_b_sgpioc2	bidir	0	144	0	Z	I/O	26v5vs
146	BC_2	*	controlr	0					
147	BC_7	irq0_b_sgpioc0	bidir	0	146	0	Z	I/O	26v5vs
148	BC_2	*	internal	1					
149	BC_2	we_b_at(0)	output2	1				O	26v
150	BC_2	*	internal	1					
151	BC_2	we_b_at(1)	output2	1				O	26v

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
152	BC_2	*	internal	1					
153	BC_2	we_b_at(2)	output2	1				O	26v
154	BC_2	*	internal	1					
155	BC_2	we_b_at(3)	output2	1				O	26v
156	BC_2	*	internal	1					
157	BC_2	cs0_b	output2	1				O	26v
158	BC_2	*	internal	1					
159	BC_2	cs1_b	output2	1				O	26v
160	BC_2	*	internal	1					
161	BC_2	cs2_b	output2	1				O	26v
162	BC_2	*	internal	1					
163	BC_2	cs3_b	output2	1				O	26v
164	BC_2	*	controlr	0					
165	BC_7	burst_b	bidir	0	164	0	Z	I/O	26v
166	BC_2	*	controlr	0					
167	BC_7	bi_b_sts_b	bidir	0	166	0	Z	I/O	26v
168	BC_2	*	controlr	0					
169	BC_7	tsiz0	bidir	0	168	0	Z	I/O	26v
170	BC_2	*	controlr	0					
171	BC_7	tsiz1	bidir	0	170	0	Z	I/O	26v
172	BC_2	*	controlr	0					
173	BC_7	ts_b	bidir	0	172	0	Z	I/O	26v
174	BC_2	*	controlr	0					
175	BC_7	ta_b	bidir	0	174	0	Z	I/O	26v
176	BC_2	*	controlr	0					
177	BC_7	bdip_b	bidir	0	176	0	Z	I/O	26v
178	BC_2	*	internal	0					
179	BC_4	b0epee	input	X				I	26v
180	BC_2	*	internal	0					
181	BC_4	epee	input	X				I	26v
182	BC_2	*	internal	1					
183	BC_2	clkout	output2	1				I/O	26vf

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
184	BC_2	*	internal	1					
185	BC_2	engclk_buclk	output2	1				O	26vs5vr
186	BC_2	*	controlr	0					
187	BC_7	irq5_b_sgpioc5_m odck1	bidir	0	186	0	Z	I/O	26v
188	BC_2	*	controlr	0					
189	BC_7	irq6_b_modck2	bidir	0	188	0	Z	I	26v
190	BC_2	*	controlr	0					
191	BC_7	irq7_b_modck3	bidir	0	190	0	Z	I	26v
192	BC_2	*	controlr	0					
193	BC_7	rstconf_b_texp	bidir	0	192	0	Z	I/O	26v
194	BC_4	extclk	input	X				I	extclk
195	BC_2	*	controlr	0					
196	BC_7	a_cnrx0	bidir	0	195	0	Z	I	5vsa
197	BC_2	*	internal	1					
198	BC_2	a_cntx0	output2	1				O	5vfa
199	BC_2	*	controlr	0					
200	BC_7	a_pcs0_ss_b_qgpi o0	bidir	0	199	0	Z	I/O	5vfa
201	BC_2	*	internal	0					
202	BC_4	a_eck	input	X				I	5vfa
203	BC_2	*	controlr	0					
204	BC_7	a_pcs1_qgpio1	bidir	0	203	0	Z	I/O	5vfa
205	BC_2	*	internal	1					
206	BC_2	a_txd2_qgpo2	output2	1				O	5vfa
207	BC_2	*	controlr	0					
208	BC_7	a_pcs2_qgpio2	bidir	0	207	0	Z	I/O	5vfa
209	BC_4	a_rxd2_qgpi2	input	X				I	5vido
210	BC_4	b_rxd1_qgpi1	input	X				I	5vido
211	BC_4	a_rxd1_qgpi1	input	X				I	5vido
212	BC_2	*	controlr	0					
213	BC_7	a_mosi_qgpio5	bidir	0	212	0	Z	I/O	5vh
214	BC_2	*	controlr	0					

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
215	BC_7	a_pcs3_qgpio3	bidir	0	214	0	Z	I/O	5vfa
216	BC_2	*	controlr	0					
217	BC_7	a_miso_qgpio4	bidir	0	216	0	Z	I/O	5vh
218	BC_2	*	controlr	0					
219	BC_7	a_sck_qgpio6	bidir	0	218	0	Z	I/O	5vh
220	BC_2	*	controlr	0					
221	BC_7	b_pcs2_qgpio2	bidir	0	220	0	Z	I/O	5vfa
222	BC_2	*	internal	0					
223	BC_4	b_rxd2_j1850_rx	input	X				I	5vsa
224	BC_2	*	internal	1					
225	BC_2	a_txd1_qgpo1	output2	1				O	5vsa
226	BC_2	*	internal	1					
227	BC_2	b_txd2_qgpo2	output2	1				O	5vsa
228	BC_2	*	internal	1					
229	BC_2	b_txd1_qgpo1	output2	1				O	5vsa
230	BC_2	*	internal	0					
231	BC_4	b_eck	input	X				I	5vsa
232	BC_2	*	controlr	0					
233	BC_7	b_sck_qgpio6	bidir	0	232	0	Z	I/O	5vh
234	BC_2	*	controlr	0					
235	BC_7	b_mosi_qgpio5	bidir	0	234	0	Z	I/O	5vh
236	BC_2	*	controlr	0					
237	BC_7	b_miso_qgpio4	bidir	0	236	0	Z	I/O	5vh
238	BC_2	*	controlr	0					
239	BC_7	b_pcs3_j1850_tx	bidir	0	238	0	Z	O	5vfa
240	BC_2	*	controlr	0					
241	BC_7	b_pcs1_qgpio1	bidir	0	240	0	Z	I/O	5vfa
242	BC_2	*	controlr	0					
243	BC_7	b_pcs0_ss_b_qgpio0	bidir	0	242	0	Z	I/O	5vfa
244	BC_2	*	controlr	0					
245	BC_7	vfls1_mpio32b4	bidir	0	244	0	Z	I/O	26v5vs

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
246	BC_2	*	controlr	0					
247	BC_7	vfls0_mpio32b3	bidir	0	246	0	Z	I/O	26v5vs
248	BC_2	*	controlr	0					
249	BC_7	vf2_mpio32b2	bidir	0	248	0	Z	I/O	26v5vs
250	BC_2	*	controlr	0					
251	BC_7	vf1_mpio32b1	bidir	0	250	0	Z	I/O	26v5vs
252	BC_2	*	controlr	0					
253	BC_7	vf0_mpio32b0	bidir	0	252	0	Z	I/O	26v5vs
254	BC_2	*	controlr	0					
255	BC_7	mpwm4_mpio32b5	bidir	0	254	0	Z	I/O	5vsa
256	BC_2	*	controlr	0					
257	BC_7	mpwm19	bidir	0	256	0	Z	I/O	5vsa
258	BC_2	*	controlr	0					
259	BC_7	mpio32b15	bidir	0	258	0	Z	I/O	5vsa
260	BC_2	*	controlr	0					
261	BC_7	c_cnrx0_mpio32b14	bidir	0	260	0	Z	I/O	5vsa
262	BC_2	*	controlr	0					
263	BC_7	c_cntx0_mpio32b13	bidir	0	262	0	Z	I/O	5vfa
264	BC_2	*	controlr	0					
265	BC_7	mpwm21_mpio32b12	bidir	0	264	0	Z	I/O	5vsa
266	BC_2	*	controlr	0					
267	BC_7	mpwm20_mpio32b11	bidir	0	266	0	Z	I/O	5vsa
268	BC_2	*	controlr	0					
269	BC_7	mda15	bidir	0	268	0	Z	I/O	5vsa
270	BC_2	*	controlr	0					
271	BC_7	mda14	bidir	0	270	0	Z	I/O	5vsa
272	BC_2	*	controlr	0					
273	BC_7	mpwm16	bidir	0	272	0	Z	I/O	5vsa
274	BC_2	*	controlr	0					
275	BC_7	mpwm3	bidir	0	274	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
276	BC_2	*	controlr	0					
277	BC_7	mpwm2	bidir	0	276	0	Z	I/O	5vsa
278	BC_2	*	controlr	0					
279	BC_7	mpwm1	bidir	0	278	0	Z	I/O	5vsa
280	BC_2	*	controlr	0					
281	BC_7	mpwm0	bidir	0	280	0	Z	I/O	5vsa
282	BC_2	*	controlr	0					
283	BC_7	mda31	bidir	0	282	0	Z	I/O	5vsa
284	BC_2	*	controlr	0					
285	BC_7	mda30	bidir	0	284	0	Z	I/O	5vsa
286	BC_2	*	controlr	0					
287	BC_7	mda29	bidir	0	286	0	Z	I/O	5vsa
288	BC_2	*	controlr	0					
289	BC_7	mda28	bidir	0	288	0	Z	I/O	5vsa
290	BC_2	*	controlr	0					
291	BC_7	mda27	bidir	0	290	0	Z	I/O	5vsa
292	BC_2	*	controlr	0					
293	BC_7	mda13	bidir	0	292	0	Z	I/O	5vsa
294	BC_2	*	controlr	0					
295	BC_7	mda12	bidir	0	294	0	Z	I/O	5vsa
296	BC_2	*	controlr	0					
297	BC_7	mda11	bidir	0	296	0	Z	I/O	5vsa
298	BC_2	*	controlr	0					
299	BC_7	mpwm18	bidir	0	298	0	Z	I/O	5vsa
300	BC_2	*	controlr	0					
301	BC_7	mpwm17	bidir	0	300	0	Z	I/O	5vsa
302	BC_2	*	controlr	0					
303	BC_7	b_t2clk	bidir	0	302	0	Z	I/O	5vsa
304	BC_2	*	controlr	0					
305	BC_7	b_tpuch1	bidir	0	304	0	Z	I/O	5vsa
306	BC_2	*	controlr	0					
307	BC_7	b_tpuch0	bidir	0	306	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
308	BC_2	*	controlr	0					
309	BC_7	mpwm5_mpio32b6	bidir	0	308	0	Z	I/O	5vsa
310	BC_2	*	controlr	0					
311	BC_7	b_tpuch2	bidir	0	310	0	Z	I/O	5vsa
312	BC_2	*	controlr	0					
313	BC_7	b_tpuch15	bidir	0	312	0	Z	I/O	5vsa
314	BC_2	*	controlr	0					
315	BC_7	b_tpuch14	bidir	0	314	0	Z	I/O	5vsa
316	BC_2	*	controlr	0					
317	BC_7	b_tpuch13	bidir	0	316	0	Z	I/O	5vsa
318	BC_2	*	controlr	0					
319	BC_7	b_tpuch12	bidir	0	318	0	Z	I/O	5vsa
320	BC_2	*	controlr	0					
321	BC_7	b_tpuch11	bidir	0	320	0	Z	I/O	5vsa
322	BC_2	*	controlr	0					
323	BC_7	b_tpuch10	bidir	0	322	0	Z	I/O	5vsa
324	BC_2	*	controlr	0					
325	BC_7	b_tpuch9	bidir	0	324	0	Z	I/O	5vsa
326	BC_2	*	controlr	0					
327	BC_7	b_tpuch8	bidir	0	326	0	Z	I/O	5vsa
328	BC_2	*	controlr	0					
329	BC_7	b_tpuch7	bidir	0	328	0	Z	I/O	5vsa
330	BC_2	*	controlr	0					
331	BC_7	b_tpuch6	bidir	0	330	0	Z	I/O	5vsa
332	BC_2	*	controlr	0					
333	BC_7	b_tpuch5	bidir	0	332	0	Z	I/O	5vsa
334	BC_2	*	controlr	0					
335	BC_7	b_tpuch4	bidir	0	334	0	Z	I/O	5vsa
336	BC_2	*	controlr	0					
337	BC_7	b_tpuch3	bidir	0	336	0	Z	I/O	5vsa
338	BC_2	*	controlr	0					
339	BC_7	a_tpuch1	bidir	0	338	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
340	BC_2	*	controlr	0					
341	BC_7	a_tpuch0	bidir	0	340	0	Z	I/O	5vsa
342	BC_2	*	controlr	0					
343	BC_7	a_t2clk	bidir	0	342	0	Z	I/O	5vsa
344	BC_2	*	controlr	0					
345	BC_7	a_tpuch15	bidir	0	344	0	Z	I/O	5vsa
346	BC_2	*	controlr	0					
347	BC_7	a_tpuch14	bidir	0	346	0	Z	I/O	5vsa
348	BC_2	*	controlr	0					
349	BC_7	a_tpuch13	bidir	0	348	0	Z	I/O	5vsa
350	BC_2	*	controlr	0					
351	BC_7	a_tpuch12	bidir	0	350	0	Z	I/O	5vsa
352	BC_2	*	controlr	0					
353	BC_7	a_tpuch11	bidir	0	352	0	Z	I/O	5vsa
354	BC_2	*	controlr						
355	BC_7	a_tpuch10	bidir	0	354	0	Z	I/O	5vsa
356	BC_2	*	controlr	0					
357	BC_7	a_tpuch9	bidir	0	356	0	Z	I/O	5vsa
358	BC_2	*	controlr	0					
359	BC_7	a_tpuch8	bidir	0	358	0	Z	I/O	5vsa
360	BC_2	*	controlr	0					
361	BC_7	a_tpuch7	bidir	0	360	0	Z	I/O	5vsa
362	BC_2	*	controlr	0					
363	BC_7	a_tpuch6	bidir	0	362	0	Z	I/O	5vsa
364	BC_2	*	controlr	0					
365	BC_7	a_tpuch5	bidir	0	364	0	Z	I/O	5vsa
366	BC_2	*	controlr	0					
367	BC_7	a_tpuch4	bidir	0	366	0	Z	I/O	5vsa
368	BC_2	*	controlr	0					
369	BC_7	a_tpuch3	bidir	0	368	0	Z	I/O	5vsa
370	BC_2	*	controlr	0					
371	BC_7	a_tpuch2	bidir	0	370	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
372	BC_2	*	internal	0					
373	BC_4	etrig1	input	X				I	5vsa
374	BC_2	*	internal	0					
375	BC_4	etrig2	input	X				I	5vsa
376	BC_2	*	controlr	0					
377	BC_7	an64_b_pqb0	bidir	0	376	0	Z	I/O	5vsa
378	BC_2	*	controlr	0					
379	BC_7	an65_b_pqb1	bidir	0	378	0	Z	I/O	5vsa
380	BC_2	*	controlr	0					
381	BC_7	an66_b_pqb2	bidir	0	380	0	Z	I/O	5vsa
382	BC_2	*	controlr	0					
383	BC_7	an67_b_pqb3	bidir	0	382	0	Z	I/O	5vsa
384	BC_2	*	controlr	0					
385	BC_7	an68_b_pqb4	bidir	0	384	0	Z	I/O	5vsa
386	BC_2	*	controlr	0					
387	BC_7	an69_b_pqb5	bidir	0	386	0	Z	I/O	5vsa
388	BC_2	*	controlr	0					
389	BC_7	an70_b_pqb6	bidir	0	388	0	Z	I/O	5vsa
390	BC_2	*	controlr	0					
391	BC_7	an71_b_pqb7	bidir	0	390	0	Z	I/O	5vsa
392	BC_2	*	controlr	0					
393	BC_7	an72_b_ma0_pqa0	bidir	0	392	0	Z	I/O	5vsa
394	BC_2	*	controlr	0					
395	BC_7	an73_b_ma1_pqa1	bidir	0	394	0	Z	I/O	5vsa
396	BC_2	*	controlr	0					
397	BC_7	an74_b_ma2_pqa2	bidir	0	396	0	Z	I/O	5vsa
398	BC_2	*	controlr	0					
399	BC_7	an75_b_pqa3	bidir	0	398	0	Z	I/O	5vsa
400	BC_2	*	controlr	0					
401	BC_7	an76_b_pqa4	bidir	0	400	0	Z	I/O	5vsa
402	BC_2	*	controlr	0					
403	BC_7	an77_b_pqa5	bidir	0	402	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
404	BC_2	*	controlr	0					
405	BC_7	an78_b_pqa6	bidir	0	404	0	Z	I/O	5vsa
406	BC_2	*	controlr	0					
407	BC_7	an79_b_pqa7	bidir	0	406	0	Z	I/O	5vsa
408	BC_2	*	controlr	0					
409	BC_7	an59_a_pqa7	bidir	0	408	0	Z	I/O	5vsa
410	BC_2	*	controlr	0					
411	BC_7	an58_a_pqa6	bidir	0	410	0	Z	I/O	5vsa
412	BC_2	*	controlr	0					
413	BC_7	an57_a_pqa5	bidir	0	412	0	Z	I/O	5vsa
414	BC_2	*	controlr	0					
415	BC_7	an56_a_pqa4	bidir	0	414	0	Z	I/O	5vsa
416	BC_2	*	controlr	0					
417	BC_7	an55_a_pqa3	bidir	0	416	0	Z	I/O	5vsa
418	BC_2	*	controlr	0					
419	BC_7	an54_a_ma2_pqa2	bidir	0	418	0	Z	I/O	5vsa
420	BC_2	*	controlr	0					
421	BC_7	an53_a_ma1_pqa1	bidir	0	420	0	Z	I/O	5vsa
422	BC_2	*	controlr	0					
423	BC_7	an52_a_ma0_pqa0	bidir	0	422	0	Z	I/O	5vsa
424	BC_2	*	controlr	0					
425	BC_7	an51_a_pqb7	bidir	0	424	0	Z	I/O	5vsa
426	BC_2	*	controlr	0					
427	BC_7	an50_a_pqb6	bidir	0	426	0	Z	I/O	5vsa
428	BC_2	*	controlr	0					
429	BC_7	an49_a_pqb5	bidir	0	428	0	Z	I/O	5vsa
430	BC_2	*	controlr	0					
431	BC_7	an48_a_pqb4	bidir	0	430	0	Z	I/O	5vsa
432	BC_2	*	controlr	0					
433	BC_7	an47_anz_a_pqb3	bidir	0	432	0	Z	I/O	5vsa
434	BC_2	*	controlr	0					
435	BC_7	an46_any_a_pqb2	bidir	0	434	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
436	BC_2	*	internal	0					
437	BC_4	an80	input	X				I	5vsa
438	BC_2	*	internal	0					
439	BC_4	an81	input	X				I	5vsa
440	BC_2	*	internal	0					
441	BC_4	an82	input	X				I	5vsa
442	BC_2	*	internal	0					
443	BC_4	an83	input	X				I	5vsa
444	BC_2	*	internal	0					
445	BC_4	an84	input	X				I	5vsa
446	BC_2	*	internal	0					
447	BC_4	an85	input	X				I	5vsa
448	BC_2	*	internal	0					
449	BC_4	an86	input	X				I	5vsa
450	BC_2	*	internal	0					
451	BC_4	an87	input	X				I	5vsa
452	BC_2	*	controlr	0					
453	BC_7	an45_anx_a_pqb1	bidir	0	452	0	Z	I/O	5vsa
454	BC_2	*	controlr	0					
455	BC_7	an44_anw_a_pqb0	bidir	0	454	0	Z	I/O	5vsa
456	BC_2	*	controlr	0					
457	BC_7	b_cnrx0	bidir	0	456	0	Z	I	5vsa
458	BC_2	*	internal	1					
459	BC_2	b_cntx0	output2	1				O	5vfa
460	BC_2	*	controlr	0					
461	BC_7	c_t2clk	bidir	0	460	0	Z	I/O	5vsa
462	BC_2	*	controlr	0					
463	BC_7	c_tpuch15	bidir	0	462	0	Z	I/O	5vsa
464	BC_2	*	controlr	0					
465	BC_7	c_tpuch14	bidir	0	464	0	Z	I/O	5vsa
466	BC_2	*	controlr	0					
467	BC_7	c_tpuch13	bidir	0	466	0	Z	I/O	5vsa

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
468	BC_2	*	controlr	0					
469	BC_7	c_tpuch12	bidir	0	468	0	Z	I/O	5vsa
470	BC_2	*	controlr	0					
471	BC_7	c_tpuch11	bidir	0	470	0	Z	I/O	5vsa
472	BC_2	*	controlr	0					
473	BC_7	c_tpuch10	bidir	0	472	0	Z	I/O	5vsa
474	BC_2	*	controlr	0					
475	BC_7	c_tpuch9	bidir	0	474	0	Z	I/O	5vsa
476	BC_2	*	controlr	0					
477	BC_7	c_tpuch8	bidir	0	476	0	Z	I/O	5vsa
478	BC_2	*	controlr	0					
479	BC_7	c_tpuch7	bidir	0	478	0	Z	I/O	5vsa
480	BC_2	*	controlr	0					
481	BC_7	c_tpuch6	bidir	0	480	0	Z	I/O	5vsa
482	BC_2	*	controlr	0					
483	BC_7	c_tpuch5	bidir	0	482	0	Z	I/O	5vsa
484	BC_2	*	controlr	0					
485	BC_7	c_tpuch4	bidir	0	484	0	Z	I/O	5vsa
486	BC_2	*	controlr	0					
487	BC_7	c_tpuch3	bidir	0	486	0	Z	I/O	5vsa
488	BC_2	*	controlr	0					
489	BC_7	c_tpuch2	bidir	0	488	0	Z	I/O	5vsa
490	BC_2	*	controlr	0					
491	BC_7	c_tpuch1	bidir	0	490	0	Z	I/O	5vsa
492	BC_2	*	controlr	0					
493	BC_7	c_tpuch0	bidir	0	492	0	Z	I/O	5vsa
494	BC_2	*	controlr						
495	BC_7	mcki	bidir	0	494	0	Z	I	26v
496	BC_2	*	controlr	0					
497	BC_7	mdi_0	bidir	0	496	0	Z	I	26v
498	BC_2	*	controlr	0					
499	BC_7	mdi_1	bidir	0	498	0	Z	I	26v

Table 24-1 Boundary Scan Bit Definition (Continued)

BSDL Bit	Cell Type	Port Name	BSDL Function	saf. val.	cnt. cell	dis. val.	rslt	Pin Function	Pad Type
500	BC_2	*	controlr	0					
501	BC_7	msei_b	bidir	0	500	0	Z	I	26v
502	BC_2	*	internal	1					
503	BC_2	mdo_1	output2	1				O	26v
504	BC_2	*	internal	1					
505	BC_2	mdo_0	output2	1				O	26v
506	BC_2	*	internal	1					
507	BC_2	mcko	output2	1				I/O	26v
508	BC_2	*	controlr	0					
509	BC_7	mdo_7_mpio32b7	bidir	0	508	0	Z	I/O	26v5vs
510	BC_2	*	controlr	0					
511	BC_7	mdo_6_mpio32b8	bidir	0	510	0	Z	I/O	26v5vs
512	BC_2	*	controlr	0					
513	BC_7	mdo_5_mpio32b9	bidir	0	512	0	Z	I/O	26v5vs
514	BC_2	*	controlr	0					
515	BC_7	mdo_4_mpio32b10	bidir	0	514	0	Z	I/O	26v5vs
516	BC_2	*	internal	0					
517	BC_4	*[rsti_b force to 0]	internal	0				I	26v
518	BC_2	*	controlr	0					
519	BC_7	evti_b	bidir	0	518	0	Z	I	26v

Notes:

1. Bi-state outputs (Pin Function = O) such as mdo_2, and mdo_3, are incorporated with general I/O pads hard-wired to keep output enable always on in system mode. The JTAG Control cell, indicated by the next lower bsdl bit in the chain, is configured as an “internal” only cell to be held at a “1” value (always driving out) during JTAG testing.
2. Some input-only cells made with generic I/O pads are configured with “internal” control cells to keep them always in input mode, such as epee, b0epee, and input pins that may be attached to analog references. Other input-only cells are configured as bidirectional for JTAG testing, to give the board-level ATPG tools the flexability to use the pad as an input or output, depending on the network of other devices that the pin is connected too. If it is desired to restrict these pins to only act as receivers during JTAG mode, then these JTAG bsdl entries can be converted as shown in the example below:

3. This description allows ATPG tools to use a pin as a driver or receiver:

188	BC_2	*	controlr	0					
189	BC_7	irq6_b_modck2	bidir	0	188	0	Z	I	26v

4. A modification to restrict ATPG tools to use a functional input-only pin as an input receiver only:

188	BC_2	*	internal	0					
189	BC_4	irq6_b_modck2	input	X				I	26v

5. The $\overline{\text{PORESET}}$, $\overline{\text{HRESET}}$, and $\overline{\text{SRESET}}$ pins are not part of the JTAG boundary scan chain. These pins are used in the reset configuration to enter JTAG. Board-level connections to them will not be testable with the EXTEST and CLAMP instructions. They do respond to the HI-Z JTAG instruction for parametric testing purposes.
6. The XTAL, EXTAL, and XFC pins are associated with analog signals and are excluded from the boundary scan chain.
7. The READI module reset pin, rsti_b, (bsdl pin 517) is in the JTAG boundary scan chain, but must be kept at a “0” level during JTAG testing, (except for Hi-Z testing), due to system interactions. It is classified as a “linkage” pin, and its data and control cells are configured to advise ATPG tools to drive a “0” value in during JTAG testing.
8. Pad type naming conventions:
   - 26 V – 2.6 V
   - 5 V – 5 V
   - s – slow
   - f – fast
   - h – high drive
   - a – analog input
   - i – input only
   - d – has direct connection to the pad (may be used for module test)
   - r – resized cell instance
9. Column Descriptions:
   - Columns 1 through 8 are entries from the boundary-scan description from the BSDL file. The columns and formats for each of these entries are defined in the *IEEE Std. 1149.1b-1994 Supplement to the IEEE Std. 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture* document. Descriptions of these columns are described below:
   - Column 1: Defines the bit’s ordinal position in the boundary scan register. The shift register cell nearest TDO (i.e., first to be shifted in) is defined as bit 0; the last bit to be shifted in is 519.
   - Column 2: References one of the three standard JTAG Cell Types (BC_4, BC_2, and BC_7) that are used for this JTAG cell in the MPC565 / MPC566. See the *IEEE Std. 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture* document for further description of these standard cell types.

- Column 3: Lists the pin name (also called the PortID) for all pin-related cells. For JTAG control cells or data cells that have been designated as “internal”, an asterisk, is shown in this column.
- Column 4: Lists the BSDL pin function.
- Column 5: The “safe bit” column specifies the value that should be loaded into the capture (and update) flip-flop of a given cell when board-level test generation software might otherwise choose a value randomly.
- Column 6: The “control cell” column identifies the cell number of the control cell that is associated with this data cell, and can disable its output.
- Column 7: The “disable value” column gives the value that must be scanned into the control cell identified by the previous “control cell” (column 6) to disable the port named by the relevant portID.
- Column 8: The “disable result” column identifies a given signal value of the PortID if that signal can be disabled. The values shown specifies the condition of the driver of that signal when it is disabled.
- Column 9: The “pin function” column indicates the normal system pin directionality. (– Input Only Pin, O – Output Only Pin, I/O – Bidirectional I/O pin)
- Column 10: The pad type column describes relevant characteristics about each pad type. See the Pad Type Keys in Note 5 above.

24.2.2 Instruction Register

The MPC565 / MPC566 JTAG implementation includes the public instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS), and also supports the CLAMP instruction. One additional public instruction (HI-Z) provides the capability for disabling all device output drivers. The MPC565 / MPC566 includes a 4-bit instruction register without parity consisting of a shift register with four parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The four bits are used to decode the five unique instructions listed in.

Table 24-2 Instruction Decoding

Code				
B3	B2	B1	B0	Instruction
0	0	0	0	EXTEST
0	0	0	1	SAMPLE/PRELOAD
0	X	1	X	BYPASS
0	1	0	0	HI-Z
0	1	0	1	CLAMP and BYPASS

NOTE

B0 (LSB) is shifted first.

The parallel output of the instruction register is reset to all ones in the test-logic-reset controller state.

NOTE

This preset state is equivalent to the BYPASS instruction.

During the capture-IR controller state, the parallel inputs to the instruction shift register are loaded with the CLAMP command code.

24.2.2.1 EXTEST

The external test (EXTEST) instruction selects the 520-bit boundary scan register. EXTEST also asserts internal reset for the MPC565 / MPC566 system logic to force a predictable beginning internal state while performing external boundary scan operations.

By using the TAP, the register is capable of:

a. scanning user-defined values into the output buffers

b. capturing values presented to input pins

c. controlling the output drive of three-state output or bidirectional pins

24.2.2.2 SAMPLE/PRELOAD

The SAMPLE/PRELOAD instruction initializes the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction. The SAMPLE/PRELOAD instruction also provides a means to obtain a snapshot of system data and control signals.

NOTE

Since there is no internal synchronization between the scan chain clock (TCK) and the system clock (CLKOUT), there must be provision of some form of external synchronization to achieve meaningful results.

24.2.2.3 BYPASS

The BYPASS instruction selects the single-bit bypass register as shown in **Figure 24-4**. This creates a shift register path from TDI to the bypass register and, finally, to TDO, circumventing the 520-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the MPC565 / MPC566 becomes the device under test.

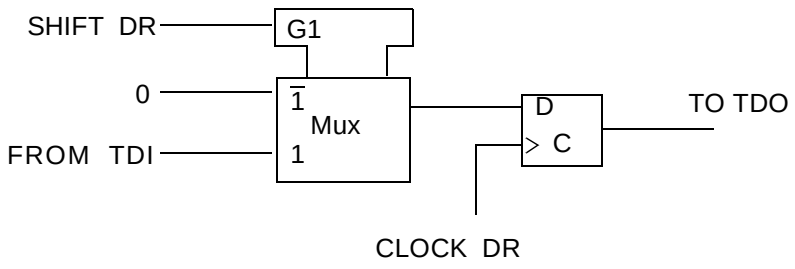


Figure 24-4 Bypass Register

When the bypass register is selected by the current instruction, the shift register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

24.2.2.4 CLAMP

The CLAMP instruction selects the single-bit bypass register as shown in **Figure 24-4**, and the state of all signals driven from system output pins is completely defined by the data previously shifted into the boundary scan register (for example, using the SAMPLE/PRELOAD instruction).

24.2.3 HI-Z

The HI-Z instruction is provided as a manufacturer's optional public instruction to prevent having to backdrive the output pins during circuit-board testing. When HI-Z is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register.

24.3 MPC565 / MPC566 Restrictions

The control afforded by the output enable signals using the boundary scan register and the EXTEST instruction requires a compatible circuit-board test environment to avoid device-destructive configurations. The user must avoid situations in which the MPC565 / MPC566 output drivers are enabled into actively driven networks.

The MPC565 / MPC566 features a low-power stop mode. The interaction of the scan chain interface with low-power stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller in the test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to V_{DD} or ground.

3. The TMS, TDI, $\overline{\text{TRST}}$ pins include on-chip pullup resistors. In low-power stop mode, these three pins should remain either unconnected or connected to V_{DD} to achieve minimal power consumption. Note that for proper reset of the scan chain test logic, the best approach is to pull active $\overline{\text{TRST}}$ at power on reset (PORESET).

24.3.1 Non-Scan Chain Operation

In non-scan chain operation, there are two constraints. First, the TCK input does not include an internal pullup resistor and should not be left unconnected to preclude mid-level inputs. The second constraint is to ensure that the scan chain test logic is kept transparent to the system logic by forcing TAP into the test-logic-reset controller state, using either of two methods. Connecting pin $\overline{\text{TRST}}$ to logic 0 (or one of the reset pins), or TMS must be sampled as a logic one for five consecutive TCK rising edges. If then TMS either remains unconnected or is connected to V_{DD}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

24.3.2 Motorola MPC565 / MPC566 BSDL Description

```
entity spanishoak is
 generic(PHYSICAL_PIN_MAP: string := “DIE”);
port(
a_cnrx0: inout bit;
 a_cntx0: buffer bit;
 a_eck: in bit;
 a_miso_qgpio4: inout bit;
 a_mosi_qgpio5: inout bit;
 a_pcs0_ss_b_qgpio0: inout bit;
 a_pcs1_qgpio1: inout bit;
 a_pcs2_qgpio2: inout bit;
 a_pcs3_qgpio3: inout bit;
 a_rxd1_qgpi1: in bit;
 a_rxd2_qgpi2: in bit;
 a_sck_qgpio6: inout bit;
 a_t2clk: inout bit;
 a_tpuch0: inout bit;
 a_tpuch1: inout bit;
 a_tpuch10: inout bit;
 a_tpuch11: inout bit;
 a_tpuch12: inout bit;
 a_tpuch13: inout bit;
 a_tpuch14: inout bit;
 a_tpuch15: inout bit;
 a_tpuch2: inout bit;
 a_tpuch3: inout bit;
 a_tpuch4: inout bit;
 a_tpuch5: inout bit;
 a_tpuch6: inout bit;
```

```
a_tpuch7: inout bit;
a_tpuch8: inout bit;
a_tpuch9: inout bit;
a_txd1_qgpo1: buffer bit;
a_txd2_qgpo2: buffer bit;
addr_sgpioa: inout bit_vector(8 to 31);
an44_anw_a_pqb0: inout bit;
an45_anx_a_pqb1: inout bit;
an46_any_a_pqb2: inout bit;
an47_anz_a_pqb3: inout bit;
an48_a_pqb4: inout bit;
an49_a_pqb5: inout bit;
an50_a_pqb6: inout bit;
an51_a_pqb7: inout bit;
an52_a_ma0_pqa0: inout bit;
an53_a_ma1_pqa1: inout bit;
an54_a_ma2_pqa2: inout bit;
an55_a_pqa3: inout bit;
an56_a_pqa4: inout bit;
an57_a_pqa5: inout bit;
an58_a_pqa6: inout bit;
an59_a_pqa7: inout bit;
an64_b_pqb0: inout bit;
an65_b_pqb1: inout bit;
an66_b_pqb2: inout bit;
an67_b_pqb3: inout bit;
an68_b_pqb4: inout bit;
an69_b_pqb5: inout bit;
an70_b_pqb6: inout bit;
an71_b_pqb7: inout bit;
```

```
an72_b_ma0_pqa0: inout bit;
an73_b_ma1_pqa1: inout bit;
an74_b_ma2_pqa2: inout bit;
an75_b_pqa3: inout bit;
an76_b_pqa4: inout bit;
an77_b_pqa5: inout bit;
an78_b_pqa6: inout bit;
an79_b_pqa7: inout bit;
an80: in bit;
an81: in bit;
an82: in bit;
an83: in bit;
an84: in bit;
an85: in bit;
an86: in bit;
an87: in bit;
b0epee: in bit;
b_cnrx0: inout bit;
b_cntx0: buffer bit;
b_eck: in bit;
b_miso_qgpio4: inout bit;
b_mosi_qgpio5: inout bit;
b_pcs0_ss_b_qgpio0: inout bit;
b_pcs1_qgpio1: inout bit;
b_pcs2_qgpio2: inout bit;
b_pcs3_j1850_tx: inout bit;
b_rxd1_qgpi1: in bit;
b_rxd2_j1850_rx: in bit;
b_sck_qgpio6: inout bit;
b_t2clk: inout bit;
```

```
b_tpuch0: inout bit;
b_tpuch1: inout bit;
b_tpuch10: inout bit;
b_tpuch11: inout bit;
b_tpuch12: inout bit;
b_tpuch13: inout bit;
b_tpuch14: inout bit;
b_tpuch15: inout bit;
b_tpuch2: inout bit;
b_tpuch3: inout bit;
b_tpuch4: inout bit;
b_tpuch5: inout bit;
b_tpuch6: inout bit;
b_tpuch7: inout bit;
b_tpuch8: inout bit;
b_tpuch9: inout bit;
b_txd1_qgpo1: buffer bit;
b_txd2_qgpo2: buffer bit;
bb_b_vf2_iwp3: inout bit;
bdip_b: inout bit;
bg_b_vf0_lwp1: inout bit;
bi_b_sts_b: inout bit;
br_b_vf1_iwp2: inout bit;
burst_b: inout bit;
c_cnrx0_mpio32b14: inout bit;
c_cntx0_mpio32b13: inout bit;
c_t2clk: inout bit;
c_tpuch0: inout bit;
c_tpuch1: inout bit;
c_tpuch10: inout bit;
```

```
c_tpuch11: inout bit;
c_tpuch12: inout bit;
c_tpuch13: inout bit;
c_tpuch14: inout bit;
c_tpuch15: inout bit;
c_tpuch2: inout bit;
c_tpuch3: inout bit;
c_tpuch4: inout bit;
c_tpuch5: inout bit;
c_tpuch6: inout bit;
c_tpuch7: inout bit;
c_tpuch8: inout bit;
c_tpuch9: inout bit;
clkout: buffer bit;
clockout1: linkage bit;
cs0_b: buffer bit;
cs1_b: buffer bit;
cs2_b: buffer bit;
cs3_b: buffer bit;
d_corner: linkage bit;
data_sgpiod: inout bit_vector(0 to 31);
engclk_buclk: buffer bit;
epee: in bit;
etrig1: in bit;
etrig2: in bit;
evti_b: inout bit;
extal: linkage bit;
extclk: in bit;
hreset_b: linkage bit;
irq0_b_sgpioc0: inout bit;
```

```
irq1_b_rsv_b_sgpioc1: inout bit;
irq2_b_cr_b_sgpioc2: inout bit;
irq3_b_kr_b_retry_b_sgpioc3: inout bit;
irq4_b_at2_sgpioc4: inout bit;
irq5_b_sgpioc5_modck1: inout bit;
irq6_b_modck2: inout bit;
irq7_b_modck3: inout bit;
iwp0_vfls0: buffer bit;
iwp1_vfls1: buffer bit;
kapwr: linkage bit;
l_corner: linkage bit;
mcki: inout bit;
mcko: buffer bit;
mda11: inout bit;
mda12: inout bit;
mda13: inout bit;
mda14: inout bit;
mda15: inout bit;
mda27: inout bit;
mda28: inout bit;
mda29: inout bit;
mda30: inout bit;
mda31: inout bit;
mdi_0: inout bit;
mdi_1: inout bit;
mdo_0: buffer bit;
mdo_1: buffer bit;
mdo_2: buffer bit;
mdo_3: buffer bit;
mdo_4_mpio32b10: inout bit;
```

```
mdo_5_mpio32b9: inout bit;
mdo_6_mpio32b8: inout bit;
mdo_7_mpio32b7: inout bit;
mpio32b15: inout bit;
mpwm0: inout bit;
mpwm1: inout bit;
mpwm16: inout bit;
mpwm17: inout bit;
mpwm18: inout bit;
mpwm19: inout bit;
mpwm2: inout bit;
mpwm20_mpio32b11: inout bit;
mpwm21_mpio32b12: inout bit;
mpwm3: inout bit;
mpwm4_mpio32b5: inout bit;
mpwm5_mpio32b6: inout bit;
mrtc_extal32: linkage bit;
mrtc_xtal32: linkage bit;
msei_b: inout bit;
mseo_b: buffer bit;
nvddl: linkage bit_vector(1 to 19);
oe_b: buffer bit;
poreset_b: linkage bit;
qvddl: linkage bit_vector(1 to 12);
r_corner: linkage bit;
rd_wr_b: inout bit;
rstconf_b_texp: inout bit;
rsti_b: linkage bit;
sgpioc6_frz_ptr_b: inout bit;
sgpioc7_irqout_b_lwp0: inout bit;
```

```
sreset_b: linkage bit;
ta_b: inout bit;
tck_dsck: in bit;
tdi_dsdi: in bit;
tdo_dsdo: out bit;
tea_b: inout bit;
tms: in bit;
trst_b: in bit;
ts_b: inout bit;
tsiz0: inout bit;
tsiz1: inout bit;
u_corner: linkage bit;
vdd: linkage bit_vector(1 to 6);
vdda: linkage bit;
vddf: linkage bit;
vddh: linkage bit_vector(1 to 6);
vddsram1: linkage bit;
vddsram2: linkage bit;
vddsram3: linkage bit;
vddsyn: linkage bit;
vddsyn32: linkage bit;
vf0_mpio32b0: inout bit;
vf1_mpio32b1: inout bit;
vf2_mpio32b2: inout bit;
vflash: linkage bit;
vfls0_mpio32b3: inout bit;
vfls1_mpio32b4: inout bit;
vrh: linkage bit;
vrhaltref: linkage bit;
vrl: linkage bit;
```

```
vss: linkage bit_vector(1 to 28);
vssa: linkage bit;
vssf: linkage bit;
vsssyn: linkage bit;
vsssyn32: linkage bit;
we_b_at: buffer bit_vector(0 to 3);
xfc: linkage bit;
xtal: linkage bit);
  use STD_1149_1_1994.all;
   attribute COMPONENT_CONFORMANCE of SpanishOak: entity is
      “STD_1149_1_1993”;   -- complies with Std. 1149.1a-1993
 attribute PIN_MAP of SpanishOak: entity is PHYSICAL_PIN_MAP;
constant DIE: PIN_MAP_STRING :=

“a_cnrx0:1,” &
“a_cntx0:2,” &
“a_eck:3,” &
“a_miso_qgpio4:4,” &
“a_mosi_qgpio5:5,” &
“a_pcs0_ss_b_qgpio0:6,” &
“a_pcs1_qgpio1:7,” &
“a_pcs2_qgpio2:8,” &
“a_pcs3_qgpio3:9,” &
“a_rxd1_qgpi1:10,” &
“a_rxd2_qgpi2:11,” &
“a_sck_qgpio6:12,” &
“a_t2clk:13,” &
“a_tpuch0:14,” &
“a_tpuch1:15,” &
“a_tpuch10:16,” &
```

```
 “a_tpuch11:17,” &

 “a_tpuch12:18,” &

 “a_tpuch13:19,” &

 “a_tpuch14:20,” &

 “a_tpuch15:21,” &

 “a_tpuch2:22,” &

 “a_tpuch3:23,” &

 “a_tpuch4:24,” &

 “a_tpuch5:25,” &

 “a_tpuch6:26,” &

 “a_tpuch7:27,” &

 “a_tpuch8:28,” &

 “a_tpuch9:29,” &

 “a_txd1_qgpo1:30,” &

 “a_txd2_qgpo2:31,” &

“addr_sgpioa:(32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,5
0,51,52,53,54,55),” &

 “an44_anw_a_pqb0:56,” &

 “an45_anx_a_pqb1:57,” &

 “an46_any_a_pqb2:58,” &

 “an47_anz_a_pqb3:59,” &

 “an48_a_pqb4:60,” &

 “an49_a_pqb5:61,” &

 “an50_a_pqb6:62,” &

 “an51_a_pqb7:63,” &

 “an52_a_ma0_pqa0:64,” &

 “an53_a_ma1_pqa1:65,” &

 “an54_a_ma2_pqa2:66,” &

 “an55_a_pqa3:67,” &

 “an56_a_pqa4:68,” &
```

```
“an57_a_pqa5:69,” &
“an58_a_pqa6:70,” &
“an59_a_pqa7:71,” &
“an64_b_pqb0:72,” &
“an65_b_pqb1:73,” &
“an66_b_pqb2:74,” &
“an67_b_pqb3:75,” &
“an68_b_pqb4:76,” &
“an69_b_pqb5:77,” &
“an70_b_pqb6:78,” &
“an71_b_pqb7:79,” &
“an72_b_ma0_pqa0:80,” &
“an73_b_ma1_pqa1:81,” &
“an74_b_ma2_pqa2:82,” &
“an75_b_pqa3:83,” &
“an76_b_pqa4:84,” &
“an77_b_pqa5:85,” &
“an78_b_pqa6:86,” &
“an79_b_pqa7:87,” &
“an80:88,” &
“an81:89,” &
“an82:90,” &
“an83:91,” &
“an84:92,” &
“an85:93,” &
“an86:94,” &
“an87:95,” &
“b0epee:96,” &
“b_cnrx0:97,” &
“b_cntx0:98,” &
```

```
“b_eck:99,” &
“b_miso_qgpio4:100,” &
“b_mosi_qgpio5:101,” &
“b_pcs0_ss_b_qgpio0:102,” &
“b_pcs1_qgpio1:103,” &
“b_pcs2_qgpio2:104,” &
“b_pcs3_j1850_tx:105,” &
“b_rxd1_qgpi1:106,” &
“b_rxd2_j1850_rx:107,” &
“b_sck_qgpio6:108,” &
“b_t2clk:109,” &
“b_tpuch0:110,” &
“b_tpuch1:111,” &
“b_tpuch10:112,” &
“b_tpuch11:113,” &
“b_tpuch12:114,” &
“b_tpuch13:115,” &
“b_tpuch14:116,” &
“b_tpuch15:117,” &
“b_tpuch2:118,” &
“b_tpuch3:119,” &
“b_tpuch4:120,” &
“b_tpuch5:121,” &
“b_tpuch6:122,” &
“b_tpuch7:123,” &
“b_tpuch8:124,” &
“b_tpuch9:125,” &
“b_txd1_qgpo1:126,” &
“b_txd2_qgpo2:127,” &
“bb_b_vf2_iwp3:128,” &
```

```
“bdip_b:129,” &
“bg_b_vf0_lwp1:130,” &
“bi_b_sts_b:131,” &
“br_b_vf1_iwp2:132,” &
“burst_b:133,” &
“c_cnrx0_mpio32b14:134,” &
“c_cntx0_mpio32b13:135,” &
“c_t2clk:136,” &
“c_tpuch0:137,” &
“c_tpuch1:138,” &
“c_tpuch10:139,” &
“c_tpuch11:140,” &
“c_tpuch12:141,” &
“c_tpuch13:142,” &
“c_tpuch14:143,” &
“c_tpuch15:144,” &
“c_tpuch2:145,” &
“c_tpuch3:146,” &
“c_tpuch4:147,” &
“c_tpuch5:148,” &
“c_tpuch6:149,” &
“c_tpuch7:150,” &
“c_tpuch8:151,” &
“c_tpuch9:152,” &
“clkout:153,” &
“clockout1:154,” &
“cs0_b:155,” &
“cs1_b:156,” &
“cs2_b:157,” &
“cs3_b:158,” &
```

```
 “d_corner:159,” &

“data_sgpiod:(160,161,162,163,164,165,166,167,168,169,170,171,17,173,
174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,1
91),” &

 “engclk_buclk:192,” &

 “epee:193,” &

 “etrig1:194,” &

 “etrig2:195,” &

 “evti_b:196,” &

 “extal:197,” &

 “extclk:198,” &

 “hreset_b:199,” &

 “irq0_b_sgpioc0:200,” &

 “irq1_b_rsv_b_sgpioc1:201,” &

 “irq2_b_cr_b_sgpioc2:202,” &

 “irq3_b_kr_b_retry_b_sgpioc3:203,” &

 “irq4_b_at2_sgpioc4:204,” &

 “irq5_b_sgpioc5_modck1:205,” &

 “irq6_b_modck2:206,” &

 “irq7_b_modck3:207,” &

 “iwp0_vfls0:208,” &

 “iwp1_vfls1:209,” &

 “kapwr:210,” &

 “l_corner:211,” &

 “mcki:212,” &

 “mcko:213,” &

 “mda11:214,” &

 “mda12:215,” &

 “mda13:216,” &

 “mda14:217,” &

 “mda15:218,” &
```

```
“mda27:219,” &
“mda28:220,” &
“mda29:221,” &
“mda30:222,” &
“mda31:223,” &
“mdi_0:224,” &
“mdi_1:225,” &
“mdo_0:226,” &
“mdo_1:227,” &
“mdo_2:228,” &
“mdo_3:229,” &
“mdo_4_mpio32b10:230,” &
“mdo_5_mpio32b9:231,” &
“mdo_6_mpio32b8:232,” &
“mdo_7_mpio32b7:233,” &
“mpio32b15:234,” &
“mpwm0:235,” &
“mpwm1:236,” &
“mpwm16:237,” &
“mpwm17:238,” &
“mpwm18:239,” &
“mpwm19:240,” &
“mpwm2:241,” &
“mpwm20_mpio32b11:242,” &
“mpwm21_mpio32b12:243,” &
“mpwm3:244,” &
“mpwm4_mpio32b5:245,” &
“mpwm5_mpio32b6:246,” &
“mrtc_extal32:247,” &
“mrtc_xtal32:248,” &
```

```
 “msei_b:249,” &

 “mseo_b:250,” &

“nvddl:(251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,2
66,267,268,269),” &

 “oe_b:270,” &

 “poreset_b:271,” &

 “qvddl:(272,273,274,275,276,277,278,279,280,281,282,283),” &

 “r_corner:284,” &

 “rd_wr_b:285,” &

 “rstconf_b_texp:286,” &

 “rsti_b:287,” &

 “sgpioc6_frz_ptr_b:288,” &

 “sgpioc7_irqout_b_lwp0:289,” &

 “sreset_b:290,” &

 “ta_b:291,” &

 “tck_dsck:292,” &

 “tdi_dsdi:293,” &

 “tdo_dsdo:294,” &

 “tea_b:295,” &

 “tms:296,” &

 “trst_b:297,” &

 “ts_b:298,” &

 “tsiz0:299,” &

 “tsiz1:300,” &

 “u_corner:301,” &

 “vdd:(302,303,304,305,306,307),” &

 “vdda:308,” &

 “vddf:309,” &

 “vddh:(310,311,312,313,314,315),” &

 “vddsram1:316,” &
```

```
"vddsram2:317," &
"vddsram3:318," &
"vddsyn:319," &
"vddsyn32:320," &
"vf0_mpio32b0:321," &
"vf1_mpio32b1:322," &
"vf2_mpio32b2:323," &
"vflash:324," &
"vfls0_mpio32b3:325," &
"vfls1_mpio32b4:326," &
"vrh:327," &
"vrhaltref:328," &
"vrl:329," &

"vss:(330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345
,346,347,348,349,350,351,352,353,354,355,356,357)," &
"vssa:358," &
"vssf:359," &
"vsssyn:360," &
"vsssyn32:361," &
"we_b_at:(362,363,364,365)," &
"xfc:366," &
"xtal:367";

attribute TAP_SCAN_IN    of tdi_dsdi:  signal is true;
attribute TAP_SCAN_MODE  of tms:  signal is true;
attribute TAP_SCAN_OUT   of tdo_dsdo:  signal is true;
attribute TAP_SCAN_RESET of trst_b:  signal is true;
attribute TAP_SCAN_CLOCK of tck_dsck:  signal is (20.0e6, BOTH);

attribute INSTRUCTION_LENGTH of SpanishOak: entity is 4;
```

```
attribute INSTRUCTION_OPCODE of SpanishOak: entity is
   “EXTEST   (0000),”  &
   “SAMPLE   (0001),”  &
   “CLAMP    (0101),”  &
   “HIGHZ    (0100),”  &
   “BYPASS   (0111, 0010, 0110, 0011)”;

attribute INSTRUCTION_CAPTURE of SpanishOak: entity is “0101”;

attribute BOUNDARY_LENGTH of SpanishOak: entity is 520;

attribute BOUNDARY_REGISTER of SpanishOak: entity is
--  numcellport functionsafe[ccell disval rslt]
“0( BC_2,* ,internal, 1), “ &
  “1( BC_2,mdo_2,output2 , 1), “ &

“2( BC_2,* ,internal, 1),“ &
“3 ( BC_2,mdo_3,output2 , 1),“ &

“4 ( BC_2,* ,internal, 1), “ &
“5 ( BC_2,mseo_b,output2 , 1), “ &

“6 ( BC_2,* ,internal, 1),“ &
“7 ( BC_2,iwp0_vfls0,output2 , 1),“ &

“8 ( BC_2,* ,internal, 1),“ &
“9( BC_2,iwp1_vfls1,output2 , 1),“ &

“10 ( BC_2,*,controlr, 0),“ &
“11( BC_7,addr_sgpioa(16),bidir, 0, 10, 0, Z),“ &
```

```
“12 ( BC_2,* ,controlr, 0),“ &
“13 ( BC_7, addr_sgpioa(17),bidir, 0, 12, 0, Z),“ &
“14 ( BC_2,* ,controlr, 0),   “ &
“15 ( BC_7,sgpioc6_frz_ptr_b,bidir, 0, 14, 0, Z),“ &
“16 ( BC_2,  * ,controlr, 0),   “ &
“17 ( BC_7, addr_sgpioa(8),bidir, 0, 16, 0, Z),“ &

“18 ( BC_2,* ,controlr, 0),“ &
“19 ( BC_7,addr_sgpioa(18),bidir, 0, 18, 0, Z),“ &

“20 ( BC_2,* ,controlr, 0),“ &
“21 ( BC_7,addr_sgpioa(19),bidir, 0, 20, 0, Z),“ &

“22 ( BC_2,* ,controlr, 0),   “ &
“23 ( BC_7,addr_sgpioa(9),bidir, 0, 22, 0, Z),“ &

“24 ( BC_2,* ,controlr, 0),   “ &
“25 ( BC_7,addr_sgpioa(10),bidir, 0, 24, 0, Z),“ &

“26 ( BC_2,* ,controlr, 0),   “ &
“27 ( BC_7,addr_sgpioa(20),bidir, 0, 26, 0, Z),“ &

“28 ( BC_2,* ,controlr, 0),   “ &
“29 ( BC_7,addr_sgpioa(21),bidir, 0, 28, 0, Z),“ &

“30 ( BC_2,* ,controlr, 0),   “ &
“31 ( BC_7,addr_sgpioa(11),bidir, 0, 30, 0, Z),“ &

“32 ( BC_2,* ,controlr, 0),   “ &
```

```
“33 ( BC_7,  addr_sgpioa(12),bidir, 0, 32, 0, Z),“ &

“34 ( BC_2,* ,controlr, 0),   “ &
“35 ( BC_7,  addr_sgpioa(22),bidir, 0, 34, 0, Z),“ &

“36 ( BC_2,* ,controlr, 0),   “ &
“37 ( BC_7,  addr_sgpioa(23),bidir, 0, 36, 0, Z),“ &

“38 ( BC_2,* ,controlr, 0),   “ &
“39 ( BC_7,addr_sgpioa(13),bidir, 0, 38, 0, Z),“ &

“40 ( BC_2,* ,controlr, 0),   “ &
“41 ( BC_7,  addr_sgpioa(24),bidir, 0, 40, 0, Z),“ &

“42 ( BC_2,  * ,controlr, 0),   “ &
“43 ( BC_7,  addr_sgpioa(25),bidir, 0, 42, 0, Z),“ &

“44( BC_2,  * ,controlr, 0),   “ &
“45( BC_7,  addr_sgpioa(14),bidir, 0, 44, 0, Z),“ &

“46 ( BC_2,  *,controlr, 0),   “ &
“47 ( BC_7,  addr_sgpioa(15),bidir, 0, 46, 0, Z),“ &

“48 ( BC_2,  *,controlr,0),   “ &
“49 ( BC_7,  addr_sgpioa(30),bidir, 0, 48, 0, Z),“ &

“50 ( BC_2,  * ,controlr, 0),   “ &
“51 ( BC_7,  addr_sgpioa(26),bidir, 0, 50, 0, Z),“ &

“52 ( BC_2,  * ,controlr, 0),   “ &
```

```
“53 ( BC_7,  addr_sgpioa(27),bidir, 0, 52, 0, Z)“ &

“54 ( BC_2,  * ,controlr, 0),   “ &
“55 ( BC_7,  addr_sgpioa(31),bidir, 0, 54, 0, Z),“ &

“56 ( BC_2,  * ,controlr, 0),   “ &
“57 ( BC_7,  addr_sgpioa(28),bidir, 0, 56, 0, Z),“ &

“58 ( BC_2,  * ,controlr, 0),   “ &
“59 ( BC_7,  addr_sgpioa(29),bidir, 0 ,58, 0, Z),“ &

“60 ( BC_2,  * ,controlr, 0),   “ &
“61 ( BC_7,  data_sgpiod(0),bidir, 0, 60, 0, Z),“ &

“62 ( BC_2,  * ,controlr, 0),   “ &
“63 ( BC_7,  data_sgpiod(29),bidir, 0, 62, 0, Z),“ &

“64 ( BC_2,  * ,controlr, 0),   “ &
“65 ( BC_7,  data_sgpiod(1),bidir, 0, 64, 0, Z),“ &

“66 ( BC_2,  * ,controlr, 0),   “ &
“67 ( BC_7,  data_sgpiod(2),bidir, 0, 66, 0, Z),“ &

“68 ( BC_2,  *,controlr, 0),   “ &
“69 ( BC_7,  data_sgpiod(3),bidir, 0, 68, 0, Z),“ &

“70 ( BC_2,  * ,controlr, 0),   “ &
“71 ( BC_7,  data_sgpiod(27),bidir, 0, 70, 0, Z),“ &

“72 ( BC_2,  * ,controlr, 0),   “ &
```

```
“73 ( BC_7,  data_sgpiod(4),bidir, 0, 72, 0, Z),“ &

“74 ( BC_2,  * ,controlr, 0),   “ &
“75 ( BC_7,  data_sgpiod(28),bidir, 0, 74, 0, Z),“ &

“76 ( BC_2,  * ,controlr, 0),   “ &
“77 ( BC_7,  data_sgpiod(31),bidir, 0, 76, 0, Z),“ &

“78( BC_2,*,controlr, 0),   “ &
“79( BC_7, data_sgpiod(5),bidir, 0 78, 0, Z),“ &

“80 ( BC_2,  * ,controlr, 0),   “ &
“81 ( BC_7,  data_sgpiod(6),bidir, 0, 80, 0, Z),“ &

“82 ( BC_2,  * ,controlr, 0),   “ &
“83 ( BC_7,  data_sgpiod(30),bidir, 0, 82, 0, Z),“ &

“84 ( BC_2,  * ,controlr, 0),   “ &
“85 ( BC_7,  data_sgpiod(7),bidir, 0, 84 0 Z),“ &

“86 ( BC_2,  * ,controlr,0),   “ &
“87 ( BC_7,  data_sgpiod(25),bidir, 0, 86, 0, Z),“ &

“88 ( BC_2,  * ,controlr, 0),   “ &
“89 ( BC_7,  data_sgpiod(8),bidir, 0, 88, 0, Z),“ &

“90 ( BC_2,  * ,controlr, 0),   “ &
“91 ( BC_7,  data_sgpiod(24),bidir, 0, 90, 0, Z),“ &

“92 ( BC_2,  * ,controlr, 0),   “ &
```

```
“93 ( BC_7,  data_sgpiod(9),bidir, 0, 92, 0, Z) “ &

“94 ( BC_2,  * ,controlr, 0),   “ &
“95 ( BC_7,  data_sgpiod(10),bidir, 0, 94, 0, Z),“ &

“96 ( BC_2,  * ,controlr, 0),   “ &
“97 ( BC_7,  data_sgpiod(26),bidir, 0, 96, 0, Z),“ &

“98 ( BC_2,  * ,controlr, 0),   “ &
“99 ( BC_7,  data_sgpiod(22),bidir, 0, 98, 0, Z),“ &

“100 ( BC_2,  * ,controlr, 0),   “ &
“101 ( BC_7, data_sgpiod(11),bidir, 0, 100, 0, Z),“ &

“102 ( BC_2,  * ,controlr, 0),   “ &
“103 ( BC_7, data_sgpiod(12),bidir, 0, 102, 0, Z),“ &

“104 ( BC_2,  * ,controlr, 0),   “ &
“105 ( BC_7, data_sgpiod(13),bidir, 0, 104, 0, Z),“ &

“106 ( BC_2,  * ,controlr, 0),   “ &
“107 ( BC_7, data_sgpiod(20),bidir, 0, 106, 0, Z),“ &

“108 ( BC_2,  * ,controlr, 0),   “ &
“109 ( BC_7, data_sgpiod(14),bidir, 0, 108, 0, Z),“ &

“110 ( BC_2,  * ,controlr, 0),   “ &
“111 ( BC_7, data_sgpiod(23),bidir, 0, 110, 0, Z),“ &

“112 ( BC_2,  * ,controlr, 0),   “ &
```

```
“113 ( BC_7, data_sgpiod(15),bidir, 0, 112, 0, Z),“ &

“114 ( BC_2,  * ,controlr, 0),   “ &
“115 ( BC_7, data_sgpiod(16),bidir, 0, 114, 0, Z),“ &

“116 ( BC_2,  * ,controlr, 0),   “ &
“117 ( BC_7,  data_sgpiod(21),bidir, 0, 116, 0, Z),“ &

“118 ( BC_2,  * controlr, 0),   “ &
“119 ( BC_7,  data_sgpiod(17),bidir, 0, 118, 0, Z),“ &

“120 ( BC_2,  * ,controlr, 0),   “ &
“121 ( BC_7,  data_sgpiod(18),bidir, 0, 120, 0, Z),“ &

“122 ( BC_2,  * ,controlr, 0),   “ &
“123 ( BC_7,  data_sgpiod(19),bidir, 0, 122, 0, Z),“ &

“124 ( BC_2, * ,controlr, 0),“ &
“125 (BC_7,irq3_b_kr_b_retry_b_sgpioc3,bidir,0,124,0,Z),“&

“126 ( BC_2,  * ,controlr, 0),“ &
“127 ( BC_7,irq4_b_at2_sgpioc4,bidir, 0, 126, 0, Z),“ &

“128 ( BC_2,  * ,controlr, 0),   “ &
“129 ( BC_7,irq1_b_rsv_b_sgpioc1,bidir, 0, 128, 0, Z),“ &

“130 ( BC_2,  * ,controlr, 0),   “ &
“131 ( BC_7,sgpioc7_irqout_b_lwp0,bidir, 0, 130, 0, Z),“ &

“132 ( BC_2,  * ,controlr, 0),   “ &
```

```
“133 ( BC_7,  bb_b_vf2_iwp3,bidir, 0, 132, 0, Z),“ &

“134 ( BC_2,  * ,controlr, 0),   “ &
“135 ( BC_7,  bg_b_vf0_lwp,bidir, 0, 134, 0, Z),“ &

“136 ( BC_2,  * ,controlr, 0),   “ &
“137 ( BC_7,  br_b_vf1_iwp2,bidir, 0, 136, 0, Z),“ &

“138 ( BC_2,  * ,controlr, 0),   “ &
“139 ( BC_7,  rd_wr_b,bidir, 0, 138, 0, Z),“ &

“140 ( BC_2,  * ,internal, 1),“ &
“141 ( BC_2,  oe_b,output2, 1),“ &

“142 ( BC_2,  * ,controlr, 0),“ &
“143 ( BC_7,  tea_b,bidir, 0, 142, 0, Z),“ &

“144 ( BC_2,  * ,controlr, 0),“ &
“145 ( BC_7, irq2_b_cr_b_sgpioc2,bidir, 0, 144, 0, Z),“ &

“146 ( BC_2,  * ,controlr, 0),“ &
“147 ( BC_7,  irq0_b_sgpioc0,bidir, 0, 146, 0, Z),“ &

“148 ( BC_2,  * ,internal, 1),“ &
“149 ( BC_2,  we_b_at(0),output2, 1),“ &

“150 ( BC_2,  * ,internal, 1),   “ &
“151 ( BC_2,  we_b_at(1),output2, 1),“ &

“152 ( BC_2,  * ,internal, 1),   “ &
```

```
“153 ( BC_2,  we_b_at(2),output2, 1),“ &

“154 ( BC_2,  * ,internal, 1),   “ &
“155 ( BC_2,  we_b_at(3),output2, 1),“ &

“156 ( BC_2,  * ,internal, 1),   “ &
“157 ( BC_2,  cs0_b,output2, 1),“ &
“158 ( BC_2,  *,internal, 1),“ &
“159 ( BC_2,  cs1_b,output2 , 1),“ &

“160 ( BC_2,  *,internal, 1),“ &
“161 ( BC_2,  cs2_b,output2 , 1),“ &

“162 ( BC_2,  *,internal, 1),“ &
“163 ( BC_2,  cs3_b,output2 , 1),“ &

“164 ( BC_2,  *,controlr, 0),“ &
“165 ( BC_7,  burst_b,bidir, 0, 164, 0, Z),“ &

“166 ( BC_2,  *,controlr, 0),“ &
“167 ( BC_7,  bi_b_sts_b,bidir, 0, 166, 0, Z),“ &

“168 ( BC_2,  *,controlr, 0),“ &
“169 ( BC_7,  tsiz0,bidir, 0, 168, 0, Z),“ &

“170( BC_2,  *,controlr, 0),“ &
“171( BC_7,  tsiz1,bidir, 0, 170, 0, Z),“ &

“172 ( BC_2,  *,controlr, 0),“ &
“173 ( BC_7,  ts_b,bidir, 0, 172, 0, Z),“ &
```

```
“174 ( BC_2,  *,controlr, 0),“ &
“175 ( BC_7,  ta_b,bidir, 0, 174, 0, Z),“ &

“176 ( BC_2,  *,controlr, 0),“ &
“177 ( BC_7,  bdip_b,bidir, 0, 176, 0, Z),“ &

“178 ( BC_2,  *,internal, 0), “ &
“179 ( BC_4,  b0epee,input, X), “ &

“180 ( BC_2,  *,internal, 0), “ &
“181 ( BC_4,  epee,input, X), “ &

“182 ( BC_2,  *,internal, 1),“ &
“183 ( BC_2,  clkout,output2, 1), “ &

“184 ( BC_2,  *,internal, 1), “ &
“185 ( BC_2,  engclk_buclk,output2 , 1), “ &

“186 ( BC_2,  *,controlr, 0), “ &
“187 ( BC_7,irq5_b_sgpioc5_modck1,bidir, 0, 186, 0, Z),“ &

“188 ( BC_2,  *,controlr, 0),   “ &
“189 ( BC_7,  irq6_b_modck2,bidir, 0, 188, 0, Z),“ &

“190 ( BC_2,  *,controlr, 0), “ &
“191 ( BC_7,  irq7_b_modck3,bidir, 0, 190, 0, Z),“ &

“192 ( BC_2,  * ,controlr, 0) “ &
“193 ( BC_7,  rstconf_b_texp,bidir, 0, 192, 0, Z),“ &
```

```
"194 ( BC_4,  extclk,input, X)," &

"195 ( BC_2,  *,controlr, 0)," &
"196 ( BC_7,  a_cnrx0,bidir, 0, 195, 0, Z)," &

"197 ( BC_2,  * ,internal, 1)," &
"198 ( BC_2,  a_cntx0,output2 , 1)," &

"199 ( BC_2,  * ,controlr, 0)," &
"200 ( BC_7, a_pcs0_ss_b_qgpio0,bidir, 0, 199, 0, Z)," &

"201 ( BC_2,  * ,internal, 0)," &
"202 ( BC_4,  a_eck ,input, X),   " &

"203 ( BC_2,  * ,controlr, 0)," &
"204 ( BC_7,  a_pcs1_qgpio1,bidir, 0, 203, 0, Z)," &

"205 ( BC_2,  * ,internal, 1)," &
"206 ( BC_2,  a_txd2_qgpo2,output2 , 1)," &

"207 ( BC_2,  * ,controlr, 0)," &
"208 ( BC_7,  a_pcs2_qgpio2,bidir, 0, 207, 0, Z)," &

"209 ( BC_4,  a_rxd2_qgpi2,input, X)," &

"210 ( BC_4,  b_rxd1_qgpi1,input, X)," &
"211 ( BC_4,  a_rxd1_qgpi1,input, X)," &

"212 ( BC_2,  * ,controlr, 0),"
```

```
“213 ( BC_7,  a_mosi_qgpio5,bidir, 0, 212, 0, Z),“ &

“214 ( BC_2,  * ,controlr, 0),“ &
“215 ( BC_7,  a_pcs3_qgpio3,bidir, 0, 214, 0, Z),“ &

“216 ( BC_2,  * ,controlr, 0),“ &
“217 ( BC_7,  a_miso_qgpio4,bidir, 0, 216, 0, Z),“ &

“218 ( BC_2,  * ,controlr, 0),“ &
“219 ( BC_7,  a_sck_qgpio6,bidir, 0, 218, 0, Z),“ &

“220 ( BC_2,  * ,controlr, 0),“ &
“221 ( BC_7,  b_pcs2_qgpio2,bidir, 0, 220, 0, Z),“ &

“222 ( BC_2,  * ,internal, 0),“ &
“223 ( BC_4,  b_rxd2_j1850_rx,input, X),“ &

“224 ( BC_2,  * ,internal, 1),“ &
“225 ( BC_2,  a_txd1_qgpo1,output2 , 1),“ &

“226 ( BC_2,  * ,internal, 1),“ &
“227 ( BC_2,  b_txd2_qgpo2,output2, 1),“ &

“228 ( BC_2,  * ,internal, 1),“ &
“229 ( BC_2,  b_txd1_qgpo1,output2 , 1),“ &

“230 ( BC_2,  * ,internal, 0),“ &
“231 ( BC_4,  b_eck,input, X),   “ &

“232 ( BC_2,  * ,controlr, 0),“ &
```

```
“233 ( BC_7,  b_sck_qgpio6,bidir, 0, 232, 0, Z),“ &

“234 ( BC_2,  * ,controlr, 0),   “ &
“235 ( BC_7,  b_mosi_qgpio5,bidir, 0, 234, 0, Z),“ &

“236 ( BC_2,  * ,controlr, 0),   “ &
“237 ( BC_7,  b_miso_qgpio4,bidir, 0, 236, 0, Z),“ &

“238 ( BC_2,  * ,controlr, 0),   “ &
“239 ( BC_7,  b_pcs3_j1850_tx,bidir, 0, 238, 0, Z),“ &

“240 ( BC_2,  * ,controlr, 0),   “ &
“241 ( BC_7, b_pcs1_qgpio1,bidir, 0, 240, 0, Z),“ &

“242 ( BC_2,  * ,controlr, 0),   “ &
“243 ( BC_7,  b_pcs0_ss_b_qgpio0,bidir, 0, 242, 0, Z),“ &

“244 ( BC_2,  * ,controlr, 0),   “ &
“245 ( BC_7,  vfls1_mpio32b4,bidir, 0, 244, 0, Z),“ &

“246 ( BC_2,  * ,controlr, 0),   “ &
“247 ( BC_7,  vfls0_mpio32b3,bidir, 0, 246, 0, Z),“ &

“248 ( BC_2,  * ,controlr, 0),   “ &
“249 ( BC_7,  vf2_mpio32b2,bidir, 0, 248, 0, Z),“ &

“250 ( BC_2,  * ,controlr, 0),   “ &
“251 ( BC_7,  vf1_mpio32b1,bidir, 0, 250, 0, Z),“ &

“252 ( BC_2,  * ,controlr, 0),   “ &
```

```
“253 ( BC_7,  vf0_mpio32b0,bidir, 0, 252, 0, Z),“ &

“254 ( BC_2,  * ,controlr, 0),“ &
“255 ( BC_7,  mpwm4_mpio32b5,bidir, 0, 254, 0, Z),“ &

“256 ( BC_2,  * ,controlr, 0),   “ &
“257 ( BC_7,  mpwm19,bidir, 0, 256, 0, Z),“ &

“258 ( BC_2,  * ,controlr, 0),“ &
“259 ( BC_7,  mpio32b15,bidir, 0, 258, 0, Z),“ &

“260 ( BC_2,  * ,controlr, 0),   “ &
“261 ( BC_7, c_cnrx0_mpio32b14,bidir, 0, 260, 0, Z),“ &

“262 ( BC_2,* ,controlr, 0),   “ &
“263 ( BC_7, c_cntx0_mpio32b1,bidir, 0, 262, 0, Z),“ &

“264 ( BC_2,* ,controlr, 0),   “ &
“265 ( BC_7, mpwm21_mpio32b12,bidir, 0, 264, 0, Z),“ &

“266 ( BC_2,  * ,controlr, 0),   “ &
“267 ( BC_7, mpwm20_mpio32b11,bidir, 0, 266, 0, Z),“ &

“268 ( BC_2,  * ,controlr, 0),   “ &
“269 ( BC_7,  mda15 ,bidir, 0, 268, 0, Z),“ &

“270 ( BC_2,  * ,controlr, 0),   “ &
“271 ( BC_7,  mda14,bidir, 0, 270, 0, Z),“ &

“272 ( BC_2,  * ,controlr, 0),   “ &
```

```
“273 ( BC_7,  mpwm16,bidir, 0, 272, 0, Z), “ &

“274 ( BC_2,  * ,controlr, 0),   “ &
“275 ( BC_7,  mpwm3,bidir, 0, 274, 0, Z),“ &

“276 ( BC_2,  * ,controlr, 0),   “ &
“277 ( BC_7,  mpwm2,bidir, 0, 276, 0, Z),“ &

“278 ( BC_2,  * ,controlr, 0),   “ &
“279 ( BC_7,  mpwm1,bidir, 0, 278, 0, Z),“ &

“280 ( BC_2,  * ,controlr, 0),   “ &
“281 ( BC_7,  mpwm0,bidir, 0, 280, 0, Z),“ &

“282 ( BC_2,  * ,controlr, 0),   “ &
“283 ( BC_7,  mda31,bidir, 0, 282, 0, Z),“ &

“284 ( BC_2,  * ,controlr, 0),   “ &
“285 ( BC_7,  mda30,bidir, 0, 284, 0, Z),“ &

“286 ( BC_2,  * ,controlr, 0),   “ &
“287 ( BC_7,  mda29,bidir, 0, 286, 0, Z),“ &

“288 ( BC_2,  * ,controlr, 0),   “ &
“289 ( BC_7,  mda28,bidir, 0, 288, 0, Z),“ &

“290 ( BC_2,  * ,controlr, 0),   “ &
“291 ( BC_7,  mda27,bidir, 0 290, 0, Z),“ &

“292 ( BC_2,  * ,controlr, 0),   “ &
```

```
“293 ( BC_7,  mda13,bidir, 0, 292, 0, Z),“ &

“294 ( BC_2,  * ,controlr, 0),   “ &
“295 ( BC_7,  mda12,bidir, 0, 294, 0, Z),“ &

“296 ( BC_2,  * ,controlr, 0),   “ &
“297 ( BC_7,  mda11,bidir, 0, 296, 0, Z),“ &

“298 ( BC_2,  * ,controlr, 0),   “ &
“299 ( BC_7,  mpwm18,bidir, 0, 298, 0, Z),“ &

“300 ( BC_2,  * ,controlr, 0),   “ &
“301 ( BC_7,  mpwm17,bidir, 0, 300, 0, Z)“ &

“302 ( BC_2,  * ,controlr, 0),   “ &
“303 ( BC_7,  b_t2clk,bidir, 0, 302, 0, Z),“ &

“304 ( BC_2,  * ,controlr, 0),   “ &
“305 ( BC_7,  b_tpuch1 ,bidir, 0, 304, 0, Z),“ &

“306 ( BC_2,  * ,controlr, 0),   “ &
“307 ( BC_7,  b_tpuch0,bidir, 0, 306, 0, Z),“ &

“308 ( BC_2,  * ,controlr, 0),   “ &
“309 ( BC_7,  mpwm5_mpio32b6,bidir, 0, 308, 0, Z),“ &

“310 ( BC_2,  * ,controlr, 0),   “ &
“311 ( BC_7,  b_tpuch2,bidir, 0, 310, 0, Z),“ &

“312 ( BC_2,  * ,controlr, 0),   “ &
```

```
“313 ( BC_7,  b_tpuch15,bidir, 0, 312, 0, Z),“ &

“314 ( BC_2,  * ,controlr, 0),   “ &
“315 ( BC_7,  b_tpuch14,bidir, 0, 314, 0, Z),“ &

“316 ( BC_2,  * ,controlr, 0),   “ &
“317 ( BC_7,  b_tpuch13,bidir, 0, 316, 0, Z),“ &

“318 ( BC_2,  * ,controlr, 0),   “ &
“319 ( BC_7,  b_tpuch12,bidir, 0, 318, 0, Z),“ &

“320 ( BC_2,  * ,controlr, 0),   “ &
“321 ( BC_7,  b_tpuch11,bidir, 0, 320, 0, Z),“ &

“322 ( BC_2,  * ,controlr, 0),   “ &
“323 ( BC_7,  b_tpuch10,bidir, 0, 322, 0, Z),“ &

“324 ( BC_2,  * ,controlr, 0),   “ &
“325 ( BC_7,  b_tpuch9,bidir, 0, 324, 0, Z),“ &

“326 ( BC_2,  * ,controlr, 0),   “ &
“327 ( BC_7,  b_tpuch8,bidir, 0, 326, 0, Z),“ &

“328 ( BC_2,  * ,controlr, 0),   “ &
“329 ( BC_7,  b_tpuch7,bidir, 0, 328, 0, Z),“ &

“330 ( BC_2,  * ,controlr, 0),   “ &
“331 ( BC_7,  b_tpuch6,bidir, 0, 330, 0, Z),“ &

“332 ( BC_2,  * ,controlr, 0),   “ &
```

```
“333 ( BC_7,  b_tpuch5,bidir, 0, 332, 0, Z),“ &

“334 ( BC_2,  * ,controlr, 0),   “ &
“335 ( BC_7,  b_tpuch4,bidir, 0, 334, 0, Z),“ &

“336 ( BC_2,  * ,controlr, 0),   “ &
“337 ( BC_7,  b_tpuch3,bidir, 0, 336, 0, Z),“ &

“338 ( BC_2,  * ,controlr, 0),   “ &
“339 ( BC_7,  a_tpuch1,bidir, 0, 338, 0, Z),“ &

“340 ( BC_2,  * ,controlr, 0),   “ &
“341 ( BC_7,  a_tpuch0,bidir, 0, 340, 0, Z),“ &

“342 ( BC_2,  * ,controlr, 0),   “ &
“343 ( BC_7,  a_t2cl,bidir, 0, 342, 0, Z),“ &

“344 ( BC_2,  * ,controlr, 0),   “ &
“345 ( BC_7, a_tpuch15,bidir, 0, 344, 0, Z),“ &

“346 ( BC_2,  * ,controlr, 0),   “ &
“347( BC_7,  a_tpuch14,bidir, 0, 346, 0, Z),“ &

“348 ( BC_2,  * ,controlr, 0),   “ &
“349 ( BC_7,  a_tpuch13,bidir, 0, 348, 0, Z),“ &

“351 ( BC_7,  a_tpuch12,bidir, 0, 350, 0, Z),“ &

“352 ( BC_2,  * ,controlr, 0),   “ &
“353 ( BC_7,  a_tpuch11,bidir, 0, 352, 0, Z),“ &
```

```
“354 ( BC_2,  * ,controlr, 0),   “ &
“355 ( BC_7,  a_tpuch10,bidir, 0, 354, 0, Z),“ &

“356 ( BC_2,  * ,controlr, 0),   “ &
“357 ( BC_7,  a_tpuch9 ,bidir, 0, 356, 0, Z),“ &

“358 ( BC_2,  * ,controlr, 0),   “ &
“359 ( BC_7,  a_tpuch8,bidir, 0, 358, 0, Z),“ &

“360 ( BC_2,  * ,controlr, 0),   “ &
“361 ( BC_7,  a_tpuch7,bidir, 0, 360, 0, Z),“ &

“362 ( BC_2,  * ,controlr, 0),   “ &
“363 ( BC_7,  a_tpuch6,bidir, 0, 362, 0, Z),“ &

“364 ( BC_2,  * ,controlr, 0),   “ &
“365 ( BC_7,  a_tpuch5,bidir, 0, 364, 0, Z),“ &

“366 ( BC_2,  * ,controlr, 0),   “ &
“367 ( BC_7,  a_tpuch4,bidir, 0, 366, 0, Z),“ &

“368 ( BC_2,  * ,controlr, 0),   “ &
“369 ( BC_7,  a_tpuch3,bidir, 0, 368, 0, Z),“ &

“370 ( BC_2,  * ,controlr, 0),   “ &
“371 ( BC_7,  a_tpuch2,bidir, 0, 370, 0, Z),“ &

“372 ( BC_2,  * ,internal, 0),   “ &
“373 ( BC_4,  etrig1,input, X), “ &
```

```
“374 ( BC_2,  * ,internal, 0),   “ &
“375 ( BC_4,  etrig2,input, X),   “ &

“376 ( BC_2,  * ,controlr, 0),   “ &
“377 ( BC_7,  an64_b_pqb0,bidir, 0, 376, 0, Z),“ &

“378 ( BC_2,  * ,controlr, 0),   “ &
“379 ( BC_7,  an65_b_pqb1,bidir, 0, 378, 0, Z)“ &

“380 ( BC_2,  * ,controlr, 0),   “ &
“381 ( BC_7,  an66_b_pqb2,bidir, 0, 380, 0, Z),“ &

“382 ( BC_2,  * ,controlr, 0),   “ &
“383 ( BC_7,  an67_b_pqb3,bidir, 0, 382, 0, Z),“ &

“384 ( BC_2,  * ,controlr, 0),   “ &
“385 ( BC_7,  an68_b_pqb4,bidir, 0, 384, 0, Z),“ &

“386 ( BC_2,  * ,controlr, 0),   “ &
“387 ( BC_7,  an69_b_pqb5,bidir, 0, 386, 0, Z),“ &

“388 ( BC_2,  * ,controlr, 0),   “ &
“389 ( BC_7,  an70_b_pqb6,bidir, 0, 388, 0, Z),“ &

“390 ( BC_2,  * ,controlr, 0),   “ &
“391 ( BC_7,  an71_b_pqb7,bidir, 0, 390, 0, Z),“ &

“392 ( BC_2,  * ,controlr, 0),   “ &
“393 ( BC_7,  an72_b_ma0_pqa0,bidir, 0, 392, 0, Z),“ &
```

```
“394 ( BC_2,  * ,controlr, 0),   “ &
“395 ( BC_7,  an73_b_ma1_pqa1,bidir, 0, 394, 0, Z),“ &

“396 ( BC_2,  * ,controlr, 0),“ &
“397 ( BC_7,  an74_b_ma2_pqa2,bidir, 0, 396, 0, Z),“ &

“398 ( BC_2, * ,controlr, 0),   “ &
“399 ( BC_7,  an75_b_pqa3,bidir, 0, 398, 0, Z),“ &

“400 ( BC_2,  * ,controlr, 0),   “ &
“401 ( BC_7,  an76_b_pqa4,bidir, 0, 400, 0, Z),“ &

“402 ( BC_2,  * ,controlr, 0),   “ &
“403 ( BC_7,  an77_b_pqa5,bidir, 0, 402, 0, Z),“ &

“404 ( BC_2,  * ,controlr, 0),   “ &
“405 ( BC_7,  an78_b_pqa6,bidir, 0, 404, 0, Z),“ &

“406 ( BC_2,  * ,controlr, 0), “ &
“407 ( BC_7,  an79_b_pqa7,bidir, 0, 406, 0, Z),“ &

“408 ( BC_2,  * ,controlr, 0), “ &
“409 ( BC_7, an59_a_pqa7,bidir, 0, 408, 0, Z),“ &

“410 ( BC_2,  * ,controlr, 0),   “ &
“411 ( BC_7,  an58_a_pqa6,bidir, 0, 410, 0, Z),“ &

“412 ( BC_2,  * ,controlr, 0),   “ &
“413 ( BC_7,  an57_a_pqa5,bidir, 0, 412, 0, Z),“ &
```

```
“414 ( BC_2,  * ,controlr, 0),   “ &
“415 ( BC_7,  an56_a_pqa4,bidir, 0, 414, 0, Z),“ &

“416 ( BC_2,  * ,controlr, 0),“ &
“417 ( BC_7,  an55_a_pqa3,bidir, 0, 416, 0, Z),“ &

“418 ( BC_2,  * ,controlr, 0), “ &
“419 ( BC_7,  an54_a_ma2_pqa2,bidir, 0, 418, 0, Z),“ &

“420 ( BC_2,  * ,controlr, 0),   “ &
“421 ( BC_7,  an53_a_ma1_pqa1,bidir, 0, 420, 0, Z),“ &

“422 ( BC_2,  * ,controlr, 0),   “ &
“423 ( BC_7,  an52_a_ma0_pqa0,bidir, 0, 422, 0, Z),“ &

“424 ( BC_2,  * ,controlr, 0),   “ &
“425 ( BC_7,  an51_a_pqb7,bidir, 0, 424, 0, Z),“ &

“426 ( BC_2,  * ,controlr, 0),   “ &
“427 ( BC_7,  an50_a_pqb6,bidir, 0, 426, 0, Z),“ &

“428 ( BC_2,  * ,controlr, 0),   “ &
“429 ( BC_7,  an49_a_pqb5,bidir, 0, 428, 0, Z),“ &

“430 ( BC_2,  * ,controlr, 0),   “ &
“431 ( BC_7,  an48_a_pqb4,bidir, 0, 430, 0, Z),“ &

“432 ( BC_2,  * ,controlr, 0),   “ &
“433 ( BC_7,  an47_anz_a_pqb3,bidir, 0, 432, 0, Z),“ &
```

```
“434 ( BC_2,  * ,controlr, 0),   “ &
“435 ( BC_7,  an46_any_a_pqb2,bidir, 0, 434, 0, Z),“ &

“436 ( BC_2,  * ,internal, 0),   “ &
“437 ( BC_4,  an80,input, X),   “ &

“438 ( BC_2,  * ,internal, 0),   “ &
“439 ( BC_4,  an81,input, X),   “ &

“440 ( BC_2,  * ,internal, 0),   “ &
“441 ( BC_4,  an82,input, X),   “ &

“442 ( BC_2,  * ,internal, 0),   “ &
“443 ( BC_4,  an83,input, X),   “ &

“444 ( BC_2,  * ,internal, 0),   “ &
“445 ( BC_4,  an84,input, X),   “ &

“446 ( BC_2,  * ,internal, 0),   “ &
“447 ( BC_4,  an85,input, X),   “ &

“448 ( BC_2,  * ,internal, 0),   “ &
“449 ( BC_4,  an86,input, X),   “ &

“450 ( BC_2,  * ,internal, 0),   “ &
“451 ( BC_4,  an87,input, X),   “ &

“452 ( BC_2,  * ,controlr, 0),   “ &
“453 ( BC_7,  an45_anx_a_pqb1,bidir, 0, 452, 0, Z),“ &
```

```
“454 ( BC_2,  * ,controlr, 0),   “ &
“455 ( BC_7,  an44_anw_a_pqb0,bidir, 0, 454, 0, Z),“ &

“456 ( BC_2,  * ,controlr, 0),   “ &
“457 ( BC_7,  b_cnrx0,bidir, 0, 456, 0, Z),“ &

“458 ( BC_2,  * ,internal, 1),   “ &
“459 ( BC_2,  b_cntx0,output2, 1),   “ &

“460 ( BC_2,  * ,controlr, 0),   “ &
“461 ( BC_7,  c_t2clk,bidir, 0, 460, 0, Z),“ &

“462 ( BC_2,  * ,controlr, 0),   “ &
“463 ( BC_7,  c_tpuch15,bidir, 0, 462, 0, Z),“ &

“464 ( BC_2,  * ,controlr, 0),   “ &
“465 ( BC_7,  c_tpuch14,bidir, 0, 464, 0, Z),“ &

“466 ( BC_2,  * ,controlr, 0),   “ &
“467 ( BC_7,  c_tpuch13,bidir, 0, 466, 0, Z),“ &

“468 ( BC_2,  * ,controlr, 0),   “ &
“469 ( BC_7,  c_tpuch12,bidir, 0, 468, 0, Z),“ &

“470 ( BC_2,  * ,controlr, 0),   “ &
“471 ( BC_7,  c_tpuch11,bidir, 0, 470, 0, Z),“ &

“472 ( BC_2,  * ,controlr, 0),   “ &
“473 ( BC_7,  c_tpuch10,bidir, 0, 472, 0, Z),“ &
```

```
“474 ( BC_2,  * ,controlr, 0),   “ &
“475 ( BC_7,  c_tpuch9,bidir, 0, 474, 0, Z),“ &

“476 ( BC_2,  * ,controlr, 0),   “ &
“477 ( BC_7,  c_tpuch8,bidir, 0, 476, 0, Z),“ &

“478 ( BC_2,  * ,controlr, 0),   “ &
“479 ( BC_7,  c_tpuch7,bidir, 0, 478, 0, Z),“ &

“480 ( BC_2,  * ,controlr, 0),   “ &
“481 ( BC_7,  c_tpuch6,bidir, 0, 480, 0, Z),“ &

“482 ( BC_2,  * ,controlr, 0),   “ &
“483 ( BC_7,  c_tpuch5,bidir, 0, 482, 0, Z),“ &

“484 ( BC_2,  * ,controlr, 0),   “ &
“485 ( BC_7,  c_tpuch4,bidir, 0, 484, 0, Z),“

“486 ( BC_2,  * , controlr, 0),   “ &
“487 ( BC_7,  c_tpuch3,bidir, 0, 486, 0, Z),“ &

“488 ( BC_2,  * ,controlr, 0),   “ &
“489 ( BC_7,  c_tpuch2,bidir, 0, 488, 0, Z),“ &

“490 ( BC_2,  * ,controlr, 0),   “ &
“491 ( BC_7,  c_tpuch1,bidir, 0, 490, 0, Z),“ &

“492 ( BC_2,  * ,controlr, 0),   “ &
“493 ( BC_7,  c_tpuch0,bidir, 0, 492, 0, Z),“ &
```

```
“494 ( BC_2,  * ,controlr, 0),   “ &
“495 ( BC_7,  mcki,bidir, 0, 494, 0, Z),“ &

“496 ( BC_2,  * ,controlr, 0),   “ &
“497 ( BC_7,  mdi_0,bidir, 0, 496, 0, Z),“ &

“498 ( BC_2,  * ,controlr, 0),   “ &
“499 ( BC_7,  mdi_1,bidir, 0, 498, 0, Z),“ &

“500 ( BC_2,  * ,controlr, 0),   “ &
“501 ( BC_7,  msei_b,bidir, 0, 500, 0, Z),“ &

“502 ( BC_2,  * ,internal, 1),   “ &
“503 ( BC_2,  mdo_1,output2 , 1),   “ &

“504 ( BC_2,  * ,internal, 1),   “ &
“505 ( BC_2,  mdo_0,output2 , 1),   “ &

“506 ( BC_2,  * ,internal, 1),   “ &
“507 ( BC_2,  mcko,output2 , 1),   “ &

“508 ( BC_2,  * ,controlr, 0),   “ &
“509 ( BC_7,  mdo_7_mpio32b7,bidir, 0, 508 0 Z),“ &

“510 ( BC_2,  * ,controlr, 0),   “ &
“511 ( BC_7,  mdo_6_mpio32b8,bidir, 0, 510, 0, Z),“ &

“512 ( BC_2,  * ,controlr, 0),   “ &
“513 ( BC_7,  mdo_5_mpio32b9,bidir, 0, 512, 0, Z),“ &
```

```
“514 ( BC_2,  * ,controlr, 0),   “ &
“515 ( BC_7,  mdo_4_mpio32b10,bidir, 0, 514, 0, Z),“ &

“516 ( BC_2,* ,controlr, 0),“ &
“517 ( BC_4, * ,internal, 0),“ &

“518 ( BC_2,  * ,controlr, 0),   “ &
“519 ( BC_7, evti_b,bidir, 0, 518, 0, Z)“;

end spanishoak;
```

APPENDIX A
INTERNAL MEMORY MAP

The tables below use the following notations.

In the Access column: S = Supervisor Access Only, U = User Access, T = Test Access

In the Reset column: S = $\overline{\text{SRESET}}$, H = $\overline{\text{HRESET}}$, M = $\overline{\text{Module Reset}}$,
POR = $\overline{\text{Power-On Reset}}$, U = Unchanged, X = Unknown

The codes in the Reset column indicate which reset has an effect on register values.

INDEX of MEMORY MAP TABLES

Table A-1 SPR (Special Purpose Registers)

Table A-2 UC3F Flash Array

Table A-3 DECRAM SRAM Array

Table A-4 BBC (Burst Buffer Controller Module)

Table A-5 USIU (Unified System Interface Unit)

Table A-6 CDR3 Flash Control Registers EEPROM (UC3F)

Table A-7 DPTRAM AB and C Control Registers

Table A-8 DLCMD2 (Data Link Controller Module)

Table A-9 DPTRAM Memory Arrays

Table A-10 Time Processor Unit 3 A and B (TPU3 A and B)

Table A-11 QADC64E A and B (Queued Analog-to-Digital Converter)

Table A-12 QSMCM A and B (Queued Serial Multi-Channel Module)

Table A-13 Time Processor Unit 3 C (TPU3_C) (See Table A-10 for Bit Descriptions)

Table A-14 MIOS14 (Modular Input/Output Subsystem)

Table A-15 TouCAN A, B and C (CAN 2.0B Controller)

Table A-16 UIMB (U-Bus to IMB Bus Interface)

Table A-17 CALRAM_A and CALRAM_B Control Registers

Table A-18 CALRAM_B and CALRAM_A Array

Table A-1 SPR (Special Purpose Registers)

Address	Access	Symbol	Register	Size	Reset
CR	U	CR	Condition State Register See **3.7.4 Condition Register (CR)** for bit descriptions	32	—
FPSCR	U	FPSCR	Floating-Point Status and Control Register See **Table 3-5** for bit descriptions	32	—
MSR	U	MSR	Machine State Register See **Table 3-12** for bit descriptions	32	—
SPR 1	U	XER	Integer Exception Register See **Table 3-10** for bit descriptions	32	—
SPR 8	U	LR	Link Register See **3.7.6 Link Register (LR)** for bit descrip-tions	32	—
SPR 9	U	CTR	Count Register See **3.7.7 Count Register (CTR)** for bit de-scriptions	32	—
SPR 18	U	DSISR	DAE/Source Instruction Service Register See **3.9.2 DAE/Source Instruction Service Register (DSISR)** for bit descriptions	32	—
SPR 19	U	DAR	Data Address Register See **3.9.3 Data Address Register (DAR)** for bit descriptions	32	—
SPR 22	S	DEC	Decrementer Register. See **3.9.5 Decrementer Register (DEC)** for more information.	32	POR, H
SPR 26	U	SRR0	Machine Status Save/Restore Register 0 See **3.9.6 Machine Status Save/Restore Register 0 (SRR0)** for bit descriptions	32	—
SPR 27	U	SRR1	Machine Status Save/Restore Register1 See **3.9.7 Machine Status Save/Restore Register 1 (SRR1)** for bit descriptions	32	—
SPR 80	U	EIE	External Interrupt Enable See **3.9.10.1 EIE, EID, and NRI Special-Pur-pose Registers** for bit descriptions.	32	—
SPR 81	U	EID	External Interrupt Disable See **3.9.10.1 EIE, EID, and NRI Special-Pur-pose Registers** for bit descriptions.	32	—
SPR 82	U	NRI	Non-Recoverable Interrupt Register See **3.9.10.1 EIE, EID, and NRI Special-Pur-pose Registers** for bit descriptions	32	—
SPR 144 — SPR 147	—	CMPA — CMPD	Comparator A-D Value Register. See **Table 22-17** for bit descriptions.	32	—
SPR 148	—	ECR	Exception Cause Regiser. See **Table 22-27** for bit descriptions.	32	—
SPR 149	—	DER	Debug Enable Regiser. See **Table 22-28** for bit descriptions.	32	—
SPR 150	—	COUNTA	Breakpoint Counter A Value and Control Regiser. See **Table 22-25** for bit descriptions.	32	—
SPR 151	—	COUNTB	Breakpoint Counter B Value and Control Regiser. See **Table 22-26** for bit descriptions.	32	—

Table A-1 SPR (Special Purpose Registers) (Continued)

Address	Access	Symbol	Register	Size	Reset
SPR 152 — SPR 153	—	CMPE — CMPF	Comparator E-F Value Register. See **Table 22-18** for bit descriptions.	32	—
SPR 154 — SPR 155	—	CMPG — CMPH	Comparator G-H Value Register. See **Table 22-20** for bit descriptions.	32	—
SPR 156	—	LCTRL1	L-bus Support Control Regiser 1. See **Table 22-23** for bit descriptions.	32	—
SPR 157	—	LCTRL2	L-bus Support Control Regiser 2. See **Table 22-24** for bit descriptions.	32	—
SPR 158	—	ICTRL	I-bus Support Control Regiser. See **Table 22-21** for bit descriptions.	32	—
SPR 159	—	BAR	Breakpoint Address Register. See **Table 22-19** for bit descriptions.	32	—
SPR 268, 269	U	TBL/TBU	Time Base Register See **Table 3-11** and **6.14.4.2 Time Base SPRs** for bit descriptions	32	POR, H
SPR 272 — SPR 275	U	SPRG0 — SPRG3	General Special-Purpose Registers 0-3 See **Table 3-15** for bit descriptions	32	—
SPR 284, 285	U	TBL/TBU	Time Base (Write Only) Register See **Table 3-14** and **6.14.4.2 Time Base SPRs** for bit descriptions	32	POR, H
SPR 287	U	PVR	Processor Version Register See **Table 3-14** for bit descriptions	32	—
SPR 1022	U	FPECR	Floating-Point Exception Cause Register See **Table 3-18** for bit descriptions	32	—
SPR 528	S	MI_GRA	MI Global Region Attribute Register. See **Table 4-8** for bit descriptions.	32	—
SPR 529	S	EIBADR	External Interrupt Relocation Table Base Address Register. See **Table 4-9** for bit descriptions.	32	—
SPR 536	S	L2U_GRA	L2U Global Region Attribute Register. See **Table 11-10** for bit descriptions.	32	—
SPR 560	S	BBCMCR	BBC Module Configuration Register. See **Table 4-4** for bit descriptions.	32	—
SPR 568	S	L2U_MCR	L2U Module Configuration Register. See **Table 11-7** for bit descriptions.	32	—
SPR 630	S	DPDR	Development Port Data Register See **22.5.6 Development Port Registers** for bit descriptions.	32	—
SPR 638	S	IMMR	Internal Memory Mapping Register. See **Table 6-13** for bit descriptions.	32	POR, H
SPR 784 – 787	S	MI_RBAx	MI Region x Base Address Register. See **Table 4-5** for bit descriptions.	32	—
SPR 792 – 795	S	L2U_RBAx	L2U Region x Base Address Register. See **Table 11-8** for bit descriptions.	32	—
SPR 816 – 819	S	MI_RAx	MI Region x Attribute Register. See **Table 4-6** for bit descriptions.	32	—
SPR 824 – 827	S	L2U_RAx	L2U Region x Attribute Register. See **Table 11-9** for bit descriptions.	32	—
SPR 1022	S	FPECR	Floating-Point Exception Cause Register See **3.9.10.2 Floating-Point Exception Cause Register (FPECR)** for bit descriptions.	32	—

Table A-2 UC3F Flash Array

Address	Access	Symbol	Register	Size	Reset
C3F_A					
0x00 0000 — 0x07 FFFF	U,S	UC3F_A	UC3F Flash Array A.	32	—
C3F_B					
0x08 0000 — 0x0F FFFF	U,S	UC3F_B	UC3F Flash Array B.	32	—

Table A-3 DECRAM SRAM Array

Address	Access	Symbol	Register	Size	Reset
0x2F 8000 — 0x2F 8FFF	U,S	DECRAM	DECRAM SRAM	32	—

Table A-4 BBC (Burst Buffer Controller Module)

Address	Access	Symbol	Register	Size	Reset
0x2F A004	U	DCCR1	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A008	U	DCCR2	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A00C	U	DCCR3	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	—	—
0x2F A010	U	DCCR4	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	16	S
0x2F A014	U	DCCR5	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	S
0x2F A018	U	DCCR6	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	S
0x2F A01C	U	DCCR7	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A020	U	DCCR8	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A024	U	DCCR9	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A028	U	DCCR10	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A02C	U	DCCR11	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A030	U	DCCR12	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A034	U	DCCR13	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A038	U	DCCR14	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H
0x2F A03C	U	DCCR15	Decompressor Class Configuration Register. See **Table 4-10** for bit descriptions.	32	H

Table A-5 USIU (Unified System Interface Unit)

Address	Access	Symbol	Register	Size	Reset
0x2F C000	U[1]	SIUMCR	SIU Module Configuration Register. See **Table 6-7** for bit descriptions.	32	H
0x2F C004	U[2]	SYPCR	System Protection Control Register. See **Table 6-15** for bit descriptions.	32	H
0x2F C008	—	—	Reserved	—	—
0x2F C00E	U, write only	SWSR	Software Service Register. See **Table 6-16** for bit descriptions.	16	S
0x2F C010	U	SIPEND	Interrupt Pending Register. See **6.14.2.1 SIU Interrupt Pending Register** for bit descriptions.	32	S
0x2F C014	U	SIMASK	Interrupt Mask Register. See **SIMASK is a 32 bit read/write register. Each bit in the register corresponds to an interrupt request bit in the SIPEND register.** for bit descriptions.	32	S
0x2F C018	U	SIEL	Interrupt Edge Level Mask. See **6.14.2.7 SIU Interrupt Edge Level Register (SIEL)** for bit descriptions.	32	H
0x2F C01C	U, read only	SIVEC	Interrupt Vector. See **6.14.2.8 SIU Interrupt Vector Register** for bit descriptions.	32	—
0x2F C020	U	TESR	Transfer Error Status Register. See **Table 6-17** for bit descriptions.	32	S
0x2F C024	U	SGPIODT1	USIU General-Purpose I/O Data Register 1. See **Table 6-23** for bit descriptions.	32	H
0x2F C028	U	SGPIODT2	USIU General-Purpose I/O Data Register 2. See **Table 6-24** for bit descriptions.	32	H
0x2F C02C	U	SGPIOCR	USIU General-Purpose I/O Control Register. See **Table 6-25** for bit descriptions.	32	H
0x2F C030	U	EMCR	External Master Mode Control Register. See **Table 6-14** for bit descriptions.	32	H
0x2F C03C	U	PDMCR	Pads Module Configuration Register. See **Table 2-3** for bit descriptions.	32	H
0x2F C040 — 0x2F C044	U	SIPEND2 — SIPEND3	Interrupt Pending Registers 2 and 3. See **6.14.2.1 SIU Interrupt Pending Register** for bit descriptions.	32	S
0x2F C048 — 0x2F C04C	U	SIMASK2 — SIMASK3	Interrupt Mask Register and Interrupt Mask Registers 2 and 3. See **6.14.2.9 Interrupt In-Service Registers** for bit descriptions.	32	S
0x2F C050 — 0x2F C054	U	SISR2 — SISR3	SISR2and SISR3 Registers. See **6.14.2.9 Interrupt In-Service Registers** for bit descriptions.	32	S
0x2F C0FC — 0x2F C0FF	—	—	Reserved	—	—
Memory Controller Registers					
0x2F C100	U	BR0	Base Register 0. See **Table 10-7** for bit descriptions.	32	H
0x2F C104	U	OR0	Option Register 0. See **Table 10-8** for bit descriptions.	32	H

Table A-5 USIU (Unified System Interface Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x2F C108	U	BR1	Base Register 1. See **Table 10-7** for bit descriptions.	32	H
0x2F C10C	U	OR1	Option Register 1. See **Table 10-8** for bit descriptions.	32	H
0x2F C110	U	BR2	Base Register 2. See **Table 10-7** for bit descriptions.	32	H
0x2F C114	U	OR2	Option Register 2. See **Table 10-8** for bit descriptions.	32	H
0x2F C118	U	BR3	Base Register 3. See **Table 10-7** for bit descriptions.	32	H
0x2F C11C	U	OR3	Option Register 3. See **Table 10-8** for bit descriptions.	32	H
0x2F C120 – 0x2F C13C	—	—	Reserved	—	—
0x2F C140	U	DMBR	Dual-Mapping Base Register. See **Table 10-9** for bit descriptions.	32	H
0x2F C144	U	DMOR	Dual-Mapping Option Register. See **Table 10-10** for bit descriptions.	32	H
0x2F C148 – 0x2F C174	—	—	Reserved	—	—
0x2F C178	U	MSTAT	Memory Status. See **Table 10-6** for bit descriptions.	16	H
System Integration Timers					
0x2F C200	U[3]	TBSCR	Time Base Status and Control. See **Table 6-18** for bit descriptions.	16	H
0x2F C204	U[3]	TBREF0	Time Base Reference 0. See **6.14.4.3 Time Base Reference Registers** for bit descriptions.	32	U
0x2F C208	U[3]	TBREF1	Time Base Reference 1. See **6.14.4.3 Time Base Reference Registers** for bit descriptions.	32	U
0x2F C20C – 0x2F C21C	—	—	Reserved	—	—
0x2F C220	U[4]	RTCSC	Real-Time Clock Status and Control. See **Table 6-19** for bit descriptions.	16	H
0x2F C224	U[4]	RTC	Real-Time Clock. See **6.14.4.6 Real-Time Clock Register (RTC)** for bit descriptions.	32	U
0x2F C228	T[4]	RTSEC	Real-Time Alarm Seconds, Reserved.	32	—
0x2F C22C	U[4]	RTCAL	Real-Time Alarm. See **6.14.4.7 Real-Time Clock Alarm Register (RTCAL)** for bit descriptions.	32	U
0x2F C230 – 0x2F C23C	—	—	Reserved	—	—
0x2F C240	U[3]	PISCR	PIT Status and Control. See **Table 6-20** for bit descriptions.	16	H
0x2F C244	U[3]	PITC	PIT Count. See **Table 6-21** for bit descriptions.	32 (half reserved)	U

Table A-5 USIU (Unified System Interface Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x2F C248	U, read only	PITR	PIT Register. See **Table 6-22** for bit descriptions.	32 (half reserved)	U
0x2F C24C – 0x2F C27C	—	—	Reserved	—	—
Clocks and Reset					
0x2F C280	U[2]	SCCR	System Clock Control Register. See **Table 8-9** for bit descriptions.	32	H
0x2F C284	U[3,5,6]	PLPRCR	PLL Low Power and Reset Control Register. See **Table 8-11** for bit descriptions.	32	H
0x2F C288	U[3]	RSR	Reset Status Register. See **Table 7-3** for bit descriptions.	16	POR
0x2F C28C	U	COLIR	Change of Lock Interrupt Register. See **Table 8-12** for bit descriptions.	16	U
0x2F C290	U	VSRCR	VDDSRAM1 Control Register. See **Table 8-13** for bit descriptions.	16	U
0x2F C294 – 0x2F C2FC	—	—	Reserved	—	—
System Integration Timer Keys					
0x2F C300	U	TBSCRK	Time Base Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C304	U	TBREF0K	Time Base Reference 0 Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C308	U	TBREF1K	Time Base Reference 1 Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C30C	U	TBK	Time Base and Decrementer Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C310 – 0x2F C31C	—	—	Reserved	—	—
0x2F C320	U	RTCSCK	Real-Time Clock Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C324	U	RTCK	Real-Time Clock Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C328	U	RTSECK	Real-Time Alarm Seconds Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C32C	U	RTCALK	Real-Time Alarm Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C330 – 0x2F C33C	—	—	Reserved	—	—
0x2F C340	U	PISCRIK	PIT Status and Control Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C344	U	PITCK	PIT Count Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C348 – 0x2F C37C	—	—	Reserved	—	—
Clocks and Reset Keys					
0x2F C380	U	SCCRK	System Clock Control Key. See **Table 8-8** for bit descriptions.	32	POR

Table A-5 USIU (Unified System Interface Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x2F C384	U	PLPRCRK	PLL Low-Power and Reset Control Register Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C388	U	RSRK	Reset Status Register Key. See **Table 8-8** for bit descriptions.	32	POR
0x2F C38C – 0x2F C3FC	—	—	Reserved	—	—

NOTES:
1. Entire register is locked if bit 15 (DLK) is set.
2. Write once after power on reset (POR).
3. Must use the key register to unlock if it has been locked by a key register, see **8.9.3.2 Keep-Alive Power Registers Lock Mechanism**.
4. Locked after Power on Reset (POR). A write of 0x55CCAA33 must performed to the key register to unlock. See **8.9.3.2 Keep-Alive Power Registers Lock Mechanism**.
5. Can have bits 0:11 (MF bits) write-protected by setting bit 4 (MFPDL) in the SCCR register to 1. Bit 21 (CSRC) and bits 22:23 (LPM) can be locked by setting bit 5 (LPML) of the SCCR register to 1.
6. Bit 24 (CSR) is write-once after soft reset.

Table A-6 CDR3 Flash Control Registers EEPROM (UC3F)

Address	Access	Symbol	Register	Size	Reset
C3F_A					
0x2F C800	U, S	C3FMCR_A	C3F EEPROM Configuration Register. See **Table 20-3** for bit descriptions.	32	POR, H
0x2F C804	U, S	C3FMCRE_A	C3F EEPROM Extended Configuration Register. See **Table 20-4** for bit descriptions.	32	POR, H
0x2F C808	U, S	C3FCTL_A	C3F EEPROM High Voltage Control Register. See **Table 20-5** for bit descriptions.	32	POR, H
C3F_B					
0x2F C840	U, S	C3FMCR_B	C3F EEPROM Configuration Register	32	POR, H
0x2F C844	U, S	C3FMCRE_B	C3F EEPROM Extended Configuration Register	32	POR, H
0x2F C848	U, S	C3FCTL_B	C3F EEPROM High Voltage Control Register	32	POR, H

Table A-7 DPTRAM AB and C Control Registers

Address	Access	Symbol	Register	Size	Reset
DPTRAM_AB Control					
0x30 0000	U, S[1]	DPTMCR_AB	DPTRAM Module Configuration Register. See **Table 19-2** for bit descriptions.	16	S
0x30 0002	S	DPTTCR_AB	Test Configuration Register.	16	S
0x30 0004	S	RAMBAR_AB	RAM Array Base Address Register. See **Table 19-3** for bit descriptions.	16	S
0x30 0006	S	MISRH_AB	Multiple Input Signature Register High.	16	S
0x30 0008	S	MISRL_AB	Multiple Input Signature Register Low.	16	S
0x30 000A	S	MISCNT_AB	MISC Counter Register.	16	S
DPTRAM_C Control					
0x30 0040	U, S[2]	DPTMCR_C	DPTRAM Module Configuration Register. See **Table 19-2** for bit descriptions.	16	S
0x30 0042	S	DPTTCR_C	Test Configuration Register.	16	S
0x30 0044	S	RAMBAR_C	RAM Array Base Address Register. See **Table 19-3** for bit descriptions.	16	S
0x30 0046	S	MISRH_C	Multiple Input Signature Register High.	16	S
0x30 0048	S	MISRL_C	Multiple Input Signature Register Low.	16	S
0x30 004A	S	MISCNT_C	MISC Counter Register.	16	S

NOTES:
1. Access to the DPTRAM_AB array through the IMB3 bus is disabled once bit 5 (EMU) of either TPUMCR_A or TPUMCR_B is set.
2. Access to the DPTRAM_C array through the IMB3 bus is disabled once bit 5 (EMU) of TPUMCR_C is set.

Table A-8 DLCMD2 (Data Link Controller Module)

Address	Access	Symbol	Register	Size	Reset
0x30 0080	S	MCR	Module Configuration Register. See **Table 15-7** for bit descriptions.	16	S, M
0x30 0084	S	IPR	Interrupt Pending Register. See **Table 15-9** for bit descriptions.	16	S, M
0x30 0086 — 0x30 0087	S	ILR — IVR	Interrupt Level Register and Interrupt Vector Register. See **Table 15-10** and **Table 15-12** for bit descriptions.	16	S, M
0x30 0088	S/U	SCTL	Symbol Timing Control and Pre-Scaler Register. See **Table 15-13** for bit descriptions.	16	S, M
0x30 008A	S/U	SDATA	Symbol Timing Control and Pre-Scaler Register. See **Table 15-15** for bit descriptions.	16	S, M
0x30 008C — 0x30 008D	S/U	CMD — TDATA	Command Register and Transmit Data Register. See **Table 15-17** and **Table 15-22** for bit descriptions.	16	S, M
0x30 008E — 0x30 008F	S/U	STAT — RDATA	Status Register and Receive Data Register. See **Table 15-23** and **Table 15-29** for bit descriptions.	16	S, M

Table A-9 DPTRAM Memory Arrays

DPTRAM_C Memory Array					
0x30 1000 — 0x30 1FFF	U, S[1]	DPRAM_C	DPTRAM Module Configuration Register. See **Table 19-2** for bit descriptions.	16	—
DPTRAM_AB Memory Array					
0x30 2000 — 0x30 37FF	U, S[2]	DPTRAM_AB	DPTRAM Module Configuration Register. See **Table 19-2** for bit descriptions.	16	—

NOTES:
1. Access to the DPTRAM_C array through the IMB3 bus is disabled once bit 5 (EMU) of TPUMCR_C is set.
2. Access to the DPTRAM_AB array through the IMB3 bus is disabled once bit 5 (EMU) of either TPUMCR_A or TPUMCR_B is set.

Table A-10 Time Processor Unit 3 A and B (TPU3 A and B)

Address	Access	Symbol	Register	Size	Reset
TPU_A (Note: Bit descriptions apply to TPU_B and TPU_C as well)					
0x30 4000	S[1]	TPUMCR_A	TPU3_A Module Configuration Register. See **Table 18-6** for bit descriptions.	16 only	S, M
0x30 4002	T	TCR_A	TPU3_A Test Configuration Register.	16	S, M
0x30 4004	S	DSCR_A	TPU3_A Development Support Control Register. See **Table 18-7** for bit descriptions.	16[2]	S, M
0x30 4006	S	DSSR_A	TPU3_A Development Support Status Register. See **Table 18-8** for bit descriptions.	16[2]	S, M
0x30 4008	S	TICR_A	TPU3_A Interrupt Configuration Register. See **Table 18-9** for bit descriptions.	16[2]	S, M
0x30 400A	S	CIER_A	TPU3_A Channel Interrupt Enable Register. See **Table 18-10** for bit descriptions.	16[2]	S, M
0x30 400C	S	CFSR0_A	TPU3_A Channel Function Selection Register 0. See **Table 18-11** for bit descriptions.	16[2]	S, M
0x30 400E	S	CFSR1_A	TPU3_A Channel Function Selection Register 1. See **Table 18-11** for bit descriptions.	16[2]	S, M
0x30 4010	S	CFSR2_A	TPU3_A Channel Function Selection Register 2. See **Table 18-11** for bit descriptions.	16[2]	S, M
0x30 4012	S	CFSR3_A	TPU_A Channel Function Selection Register 3. See **Table 18-11** for bit descriptions.	16[2]	S, M
0x30 4014	S/U[3]	HSQR0_A	TPU_A Host Sequence Register 0. See **Table 18-12** for bit descriptions.	16[2]	S, M
0x30 4016	S/U[3]	HSQR1_A	TPU_A Host Sequence Register 1. See **Table 18-12** for bit descriptions.	16[2]	S, M
0x30 4018	S/U[3]	HSRR0_A	TPU_A Host Service Request Register 0. See **Table 18-13** for bit descriptions.	16[2]	S, M
0x30 401A	S/U[3]	HSRR1_A	TPU_A Host Service Request Register 1. See **Table 18-13** for bit descriptions.	16[2]	S, M
0x30 401C	S	CPR0_A	TPU_A Channel Priority Register 0. See **Table 18-14** for bit descriptions.	16[2]	S, M
0x30 401E	S	CPR1_A	TPU_A Channel Priority Register 1. See **Table 18-14** for bit descriptions.	16[2]	S, M
0x30 4020	S	CISR_A	TPU_A Channel Interrupt Status Register. See **Table 18-16** for bit descriptions.	16	S, M

Table A-10 Time Processor Unit 3 A and B (TPU3 A and B) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 4022	T	LR_A	TPU_A Link Register[4]	16[2]	S, M
0x30 4024	T	SGLR_A	TPU_A Service Grant Latch Register[4]	16[2]	S, M
0x30 4026	T	DCNR_A	TPU_A Decoded Channel Number Register[4]	16[2]	S, M
0x30 4028	S[5]	TPUMCR2_A	TPU_A Module Configuration Register 2. See **Table 18-17** for bit descriptions.	16[2]	S, M
0x30 402A	S	TPUMCR3_A	TPU_A Module Configuration Register 3. See **Table 18-20** for bit descriptions.	16[2]	S, M
0x30 402C	T	ISDR_A	TPU_A Internal Scan Data Register	16, 32[2]	—
0x30 402E	T	ISCR_A	TPU_A Internal Scan Control Register	16, 32[2]	—
0x30 4100 – 0x30 410F	S/U[3]	—	TPU_A Channel 0 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4110 – 0x30 411F	S/U[3]	—	TPU_A Channel 1 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4120 – 0x30 412F	S/U[3]	—	TPU_A Channel 2 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4130 – 0x30 413F	S/U[3]	—	TPU_A Channel 3 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4140 – 0x30 414F	S/U[3]	—	TPU_A Channel 4 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4150 – 0x30 415F	S/U[3]	—	TPU_A Channel 5 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4160 – 0x30 416F	S/U[3]	—	TPU_A Channel 6 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4170 – 0x30 417F	S/U[3]	—	TPU_A Channel 7 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4180 – 0x30 418F	S/U[3]	—	TPU_A Channel 8 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 4190 – 0x30 419F	S/U[3]	—	TPU_A Channel 9 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 41A0 – 0x30 41AF	S/U[3]	—	TPU_A Channel 10 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 41B0 – 0x30 41BF	S/U[3]	—	TPU_A Channel 11 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 41C0 – 0x30 41CF	S/U[3]	—	TPU_A Channel 11 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—

Table A-10 Time Processor Unit 3 A and B (TPU3 A and B) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 41D0 – 0x30 41DF	S/U[3]	—	TPU_A Channel 11 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 41E0 – 0x30 41EF	S/U[3]	—	TPU_A Channel 14 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
0x30 41F0 – 0x30 41FF	S/U[3]	—	TPU_A Channel 15 Parameter Registers. See **18.4.18 TPU3 Parameter RAM** for more information.	16, 32[2]	—
TPU_B					
0x30 4400[1]	S[1]	TPUMCR_B	TPU3_B Module Configuration Register	16 only	S, M
0x30 4402	T	TCR_B	TPU3_B Test Configuration Register	16	S, M
0x30 4404	S	DSCR_B	TPU3_B Development Support Control Register	16[2]	S, M
0x30 4406	S	DSSR_B	TPU3_B Development Support Status Register	16[2]	S, M
0x30 4408	S	TICR_B	TPU3_B Interrupt Configuration Register	16[2]	S, M
0x30 440A	S	CIER_B	TPU3_B Channel Interrupt Enable Register	16[2]	S, M
0x30 440C	S	CFSR0_B	TPU3_B Channel Function Selection Register 0	16[2]	S, M
0x30 440E	S	CFSR1_B	TPU3_B Channel Function Selection Register 1	16[2]	S, M
0x30 4410	S	CFSR2_B	TPU3_B Channel Function Selection Register 2	16[2]	S, M
0x30 4412	S	CFSR3_B	TPU_B Channel Function Selection Register 3	16[2]	S, M
0x30 4414	S/U[3]	HSQR0_B	TPU_B Host Sequence Register 0	16[2]	S, M
0x30 4416	S/U[3]	HSQR1_B	TPU_B Host Sequence Register 1	16[2]	S, M
0x30 4418	S/U[3]	HSRR0_B	TPU_B Host Service Request Register 0	16[2]	S, M
0x30 441A	S/U[3]	HSRR1_B	TPU_B Host Service Request Register 1	16[2]	S, M
0x30 441C	S	CPR0_B	TPU_B Channel Priority Register 0	16[2]	S, M
0x30 441E	S	CPR1_B	TPU_B Channel Priority Register 1	16[2]	S, M
0x30 4420	S	CISR_B	TPU_B Channel Interrupt Status Register	16	S, M
0x30 4422	T	LR_B	TPU_B Link Register	16[2]	S, M
0x30 4424	T	SGLR_B	TPU_B Service Grant Latch Register	16[2]	S, M
0x30 4426	T	DCNR_B	TPU_B Decoded Channel Number Register	16[2]	S, M
0x30 4428	S[4]	TPUMCR2_B	TPU_B Module Configuration Register 2	16[2]	S, M
0x30 442A	S	TPUMCR3_B	TPU_B Module Configuration Register 3	16, 32[2]	S, M
0x30 442C	T	ISDR_B	TPU_B Internal Scan Data Register	16, 32[2]	—
0x30 442E	T	ISCR_B	TPU_B Internal Scan Control Register	16, 32[2]	—
0x30 4500 – 0x30 450E	S/U[3]	—	TPU_B Channel 0 Parameter Registers	16, 32[2]	—
0x30 4510 – 0x30 451E	S/U[3]	—	TPU_B Channel 1 Parameter Registers	16, 32[2]	—
0x30 4520 – 0x30 452E	S/U[3]	—	TPU_B Channel 2 Parameter Registers	16, 32[2]	—

Table A-10 Time Processor Unit 3 A and B (TPU3 A and B) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 4530 – 0x30 453E	S/U[3]	—	TPU_B Channel 3 Parameter Registers	16, 32[2]	—
0x30 4540 – 0x30 454E	S/U[3]	—	TPU_B Channel 4 Parameter Registers	16, 32[2]	—
0x30 4550 – 0x30 455E	S/U[3]	—	TPU_B Channel 5 Parameter Registers	16, 32[2]	—
0x30 4560 – 0x30 456E	S/U[3]	—	TPU_B Channel 6 Parameter Registers	16, 32[2]	—
0x30 4570 – 0x30 457E	S/U[3]	—	TPU_B Channel 7 Parameter Registers	16, 32[2]	—
0x30 4580 – 0x30 458E	S/U[3]	—	TPU_B Channel 8 Parameter Registers	16, 32[2]	—
0x30 4590 – 0x30 459E	S/U[3]	—	TPU_B Channel 9 Parameter Registers	16, 32[2]	—
0x30 45A0 – 0x30 45AE	S/U[3]	—	TPU_B Channel 10 Parameter Registers	16, 32[2]	—
0x30 45B0 – 0x30 45BF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	—
0x30 45C0 – 0x30 45CF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	—
0x30 45D0 – 0x30 45DF	S/U[3]	—	TPU_B Channel 11 Parameter Registers	16, 32[2]	—
0x30 45E0 – 0x30 45EF	S/U[3]	—	TPU_B Channel 14 Parameter Registers	16, 32[2]	—
0x30 45F0 – 0x30 45FF	S/U[3]	—	TPU_B Channel 15 Parameter Registers	16[2]	—

NOTES:

1. Bit 10 (TPU3) and bit 11 (T2CSL) are write-once. Bits 1:2 (TCR1P) and bits 3:4 (TCR2P) are write-once if PWOD is not set in the TPUMCR3 register. This register cannot be accessed with a 32-bit read. It can only be accessed with an 8- or 16-bit read.
2. Some TPU registers can only be read or written with 16- or 32-bit accesses. 8-bit accesses are not allowed.
3. S/U = Supervisor accessible only if SUPV = 1 or unrestricted if SUPV = 0. Unrestricted registers allow both user and supervisor access. The SUPV bit is in the TPUMCR register.
4. TPU code development (Debug) register
5. Bits 9:10 (ETBANK), 14 (T2CF), and 15 (DTPU) are write-once.

Table A-11 QADC64E A and B (Queued Analog-to-Digital Converter)

Address	Access	Symbol	Register	Size	Reset
QADC_A (Note: Bit descriptions apply to QADC_B as well)					
0x30 4800	S	QADC64MCR_A	QADC64 Module Configuration Register. See **Table 13-5** for bit descriptions.	16	S
0x30 4802	S	QADC64TST	QADC64 Test Register.	16	S
0x30 4804	S	QADC64INT_A	Interrupt Register. See **13.2.2 QADC64E Interrupt Register** for bit descriptions.	16	S
0x30 4806	S/U	PORTQA_A/ PORTQB_A	Port A and Port B Data. See **Table 13-21** for bit descriptions.	16	U

Table A-11 QADC64E A and B (Queued Analog-to-Digital Converter) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 4808	S/U	DDRQA_A/ DDRQB_A	Port A Data and Port B Direction Register. See **Table 13-21** for bit descriptions.	16	S
0x30 480A	S/U	QACR0_A	QADC64 Control Register 0. See **Table 13-8** for bit descriptions.	16	S
0x30 480C	S/U[1]	QACR1_A	QADC64 Control Register 1. See **Table 13-10** for bit descriptions.	16	S
0x30 480E	S/U[1]	QACR2_A	QADC64 Control Register 2. See **Table 13-12** for bit descriptions.	16	S
0x30 4810	S/U	QASR0_A	QADC64 Status Register 0. See **Table 13-8** for bit descriptions.	16	S
0x30 4812	S/U	QASR1_A	QADC64 Status Register 1. See **Table 13-16** for bit descriptions.	16	S
0x30 4814 – 0x30 49FE	—	—	Reserved	—	—
0x30 4A00 – 0x30 4A7E	S/U	CCW_A	Conversion Command Word Table. See **Table 13-18** for bit descriptions.	16	U
0x30 4A80 – 0x30 4AFE	S/U	RJURR_A	Result Word Table Right-Justified, Unsigned Result Register. See **13.2.10 Result Word Table** for bit descriptions.	16	X
0x30 4B00 – 0x30 4B7E	S/U	LJSRR_A	Result Word Table Left-Justified, Signed Result Register. See **13.2.10 Result Word Table** for bit descriptions.	16	X
0x30 4B80 – 0x30 4BFE	S/U	LJURR_A	Result Word Table Left-Justified, Unsigned Result Register. See **13.2.10 Result Word Table** for bit descriptions.	16	X
QADC_B					
0x30 4C00	S	QADC64MCR_B	QADC64 Module Configuration Register	16	S
0x30 4C02	T	QADC64TEST_B	QADC64 Test Register	16	—
0x30 4C04	S	QADC64INT_B	Interrupt Register	16	S
0x30 4C06	S/U	PORTQA_B/ PORTQB_B	Port A and Port B Data	16	U
0x30 4C08	S/U	DDRQA_B/ DDRQB_B	Port A Data and Port B Direction Register	16	S
0x30 4C0A	S/U	QACR0_B	QADC64 Control Register 0	16	S
0x30 4C0C	S/U[1]	QACR1_B	QADC64 Control Register 1	16	S
0x30 4C0E	S/U[1]	QACR2_B	QADC64 Control Register 2	16	S
0x30 4C10	S/U	QASR0_B	QADC64 Status Register 0	16	S
0x30 4C12	S/U	QASR1_B	QADC64 Status Register 1	16	S
0x30 4C14 – 0x30 4DFE	—	—	Reserved	—	—
0x30 4E00 – 0x30 4E7E	S/U	CCW_B	Conversion Command Word Table	16	U
0x30 4E80 – 0x30 4EFE	S/U	RJURR_B	Result Word Table. Right-Justified, Unsigned Result Register.	16	X

Table A-11 QADC64E A and B (Queued Analog-to-Digital Converter) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 4F00 – 0x30 4F7E	S/U	LJSRR_B	Result Word Table. Left-Justified, Signed Result Register.	16	X
0x30 4F80 – 0x30 4FFE	S/U	LJURR_B	Result Word Table. Left-Justified, Unsigned Result Register.	16	X

NOTES:
1. Bit 3 (SSEx) is readable in test mode only.

Table A-12 QSMCM A and B (Queued Serial Multi-Channel Module)

Address	Access	Symbol	Register	Size	Reset
QSMCM_A **(Note: Bit descriptions apply to QSMCM_B as well)**					
0x30 5000	S	QSMCMMCR_A	QSMCM Module Configuration Register. See **Table 14-7** for bit descriptions.	16	S
0x30 5002	T	QTEST_A	QSMCM Test Register	16	S
0x30 5004	S	QDSCI_IL_A	Dual SCI Interrupt Level. See **Table 14-8** for bit descriptions.	16	S
0x30 5006	S	QSPI_IL_A	Queued SPI Interrupt Level. See **Table 14-9** for bit descriptions.	16	S
0x30 5008	S/U	SCC1R0_A	SCI1Control Register 0. See **Table 14-27** for bit descriptions.	16	S
0x30 500A	S/U	SCC1R1_A	SCI1Control Register 1. See **Table 14-28** for bit descriptions.	16	S
0x30 500C	S/U	SC1SR_A	SCI1 Status Register. See **Table 14-29** for bit descriptions.	16	S
0x30 500E	S/U	SC1DR_A	SCI1 Data Register. See **Table 14-30** for bit descriptions.	16	S
0x30 5010 — 0x30 5012	—	—	Reserved	—	—
0x30 5014	S/U	PORTQS_A	QSMCM Port QS Data Register. See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.	16	S
0x30 5016	S/U	PQSPAR/ DDRQST_A	QSMCM Port QS PIn Assignment Register/ QSMCM Port QS Data Direction Register. See **Table 14-14** for bit descriptions.	16	S
0x30 5018	S/U	SPCR0_A	QSPI Control Register 0. See **Table 14-17** for bit descriptions.	16	S
0x30 501A	S/U	SPCR1_A	QSPI Control Register 1. See **Table 14-19** for bit descriptions.	16	S
0x30 501C	S/U	SPCR2_A	QSPI Control Register 2. See **Table 14-20** for bit descriptions.	16	S
0x30 501E	S/U	SPCR3_A	QSPI Control Register 3. See **Table 14-21** for bit descriptions.	8	S
0x30 501F	S/U	SPSR_A	QSPI Status Register 3. See **Table 14-22** for bit descriptions.	8	S
0x30 5020	S/U	SCC2R0_A	SCI2 Control Register 0. See **Table 14-27** for bit descriptions.	16	S
0x30 5022	S/U	SCC2R1_A	SCI2 Control Register 1. See **Table 14-28** for bit descriptions.	16	S

Table A-12 QSMCM A and B (Queued Serial Multi-Channel Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 5024	S/U	SC2SR_A	SCI2 Status Register. See **Table 14-29** for bit descriptions.	16	S
0x30 5026	S/U	SC2DR_A	SCI2 Data Register. See **Table 14-30** for bit descriptions.	16	S
0x30 5028	S/U[1]	QSCI1CR_A	QSCI1 Control Register. See **Table 14-35** for bit descriptions.	16	S
0x30 502A	S/U[2]	QSCI1SR_A	QSCI1 Status Register. See **Table 14-36** for bit descriptions.	16	S
0x30 502C – 0x30 504A	S/U	SCTQ_A	Transmit Queue Locations	16	S
0x30 504C – 0x30 506A	S/U	SCRQ_A	Receive Queue Locations	16	S
0x30 506C – 0x30 513F	—	—	Reserved	—	—
0x30 5140 – 0x30 517F	S/U	RECRAM_A	Receive Data RAM	16	S
0x30 5180 – 0x30 51BF	S/U	TRAN.RAM_A	Transmit Data RAM	16	S
0x30 51C0 – 0x30 51DF	S/U	COMD.RAM_A	Command RAM	16	S
QSMCM_B					
0x30 5400	S	QSMCMMCR_B	QSMCM Module Configuration Register. See **Table 14-7** for bit descriptions.	16	S
0x30 5402	T	QTEST_B	QSMCM Test Register	16	S
0x30 5404	S	QDSCI_IL_B	Dual SCI Interrupt Level. See **Table 14-8** for bit descriptions.	16	S
0x30 5406	S	QSPI_IL_B	Queued SPI Interrupt Level. See **Table 14-9** for bit descriptions.	16	S
0x30 5408	S/U	SCC1R0_B	SCI1Control Register 0. See **Table 14-27** for bit descriptions.	16	S
0x30 540A	S/U	SCC1R1_B	SCI1Control Register 1. See **Table 14-28** for bit descriptions.	16	S
0x30 540C	S/U	SC1SR_B	SCI1 Status Register. See **Table 14-29** for bit descriptions.	16	S
0x30 540E	S/U	SC1DR_B	SCI1 Data Register. See **Table 14-30** for bit descriptions.	16	S
0x30 5410 — 0x30 5412	—	—	Reserved	—	—
0x30 5414	S/U	PORTQS_B	QSMCM Port QS Data Register. See **14.6.1 Port QS Data Register (PORTQS)** for bit descriptions.	16	S
0x30 5416	S/U	PQSPAR/ DDRQST_B	QSMCM Port QS PIn Assignment Register/ QSMCM Port QS Data Direction Register. See **Table 14-14** for bit descriptions.	16	S
0x30 5418	S/U	SPCR0_B	QSPI Control Register 0. See **Table 14-17** for bit descriptions.	16	S
0x30 541A	S/U	SPCR1_B	QSPI Control Register 1. See **Table 14-19** for bit descriptions.	16	S

Table A-12 QSMCM A and B (Queued Serial Multi-Channel Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 541C	S/U	SPCR2_B	QSPI Control Register 2. See **Table 14-20** for bit descriptions.	16	S
0x30 541E	S/U	SPCR3_B	QSPI Control Register 3. See **Table 14-21** for bit descriptions.	8	S
0x30 541F	S/U	SPSR_B	QSPI Status Register 3. See **Table 14-22** for bit descriptions.	8	S
0x30 5420	S/U	SCC2R0_B	SCI2 Control Register 0	16	S
0x30 5422	S/U	SCC2R1_B	SCI2 Control Register 1	16	S
0x30 5424	S/U	SC2SR_B	SCI2 Status Register	16	S
0x30 5426	S/U	SC2DR_B	SCI2 Data Register	16	S
0x30 5428	S/U[3]	QSCI1CR_B	QSCI1 Control Register. See **Table 14-35** for bit descriptions.	16	S
0x30 542A	S/U[4]	QSCI1SR_B	QSCI1 Status Register. See **Table 14-36** for bit descriptions.	16	S
0x30 542C – 0x30 544A	S/U	SCTQ_B	Transmit Queue Locations	16	S
0x30 544C – 0x30 546A	S/U	SCRQ_B	Receive Queue Locations	16	S
0x30 546C – 0x30 553F	—	—	Reserved	—	—
0x30 5540 – 0x30 557F	S/U	RECRAM_B	Receive Data RAM	16	S
0x30 5580 – 0x30 55BF	S/U	TRAN.RAM_B	Transmit Data RAM	16	S
0x30 55C0 – 0x30 55DF	S/U	COMD.RAM_B	Command RAM	16	S

NOTES:
1. Bits 0–3 writeable only in test mode, otherwise read only.
2. Bits 3–11 writeable only in test mode, otherwise read only.
3. Bits 0–3 writeable only in test mode, otherwise read only.
4. Bits 3–11 writeable only in test mode, otherwise read only.

Table A-13 Time Processor Unit 3 C (TPU3_C)
(See Table A-10 for Bit Descriptions)

Address	Access	Symbol	Register	Size	Reset
TPU_C					
0x30 5C00[1]	S[1]	TPUMCR_C	TPU3_C Module Configuration Register	16 only	S, M
0x30 5C02	T	TCR_C	TPU3_C Test Configuration Register	16	S, M
0x30 5C04	S	DSCR_C	TPU3_C Development Support Control Register	16[2]	S, M
0x30 5C06	S	DSSR_C	TPU3_C Development Support Status Register	16[2]	S, M
0x30 5C08	S	TICR_C	TPU3_C Interrupt Configuration Register	16[2]	S, M
0x30 5C0A	S	CIER_C	TPU3_C Channel Interrupt Enable Register	16[2]	S, M

Table A-13 Time Processor Unit 3 C (TPU3_C) (Continued)
(See Table A-10 for Bit Descriptions) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 5C0C	S	CFSR0_C	TPU3_C Channel Function Selection Register 0	16[2]	S, M
0x30 5C0E	S	CFSR1_C	TPU3_C Channel Function Selection Register 1	16[2]	S, M
0x30 5C10	S	CFSR2_C	TPU3_C Channel Function Selection Register 2	16[2]	S, M
0x30 5C12	S	CFSR3_C	TPU_C Channel Function Selection Register 3	16[2]	S, M
0x30 5C14	S/U[3]	HSQR0_C	TPU_C Host Sequence Register 0	16[2]	S, M
0x30 5C16	S/U[3]	HSQR1_C	TPU_C Host Sequence Register 1	16[2]	S, M
0x30 5C18	S/U[3]	HSRR0_C	TPU_C Host Service Request Register 0	16[2]	S, M
0x30 5C1A	S/U[3]	HSRR1_C	TPU_C Host Service Request Register 1	16[2]	S, M
0x30 5CC	S	CPR0_C	TPU_C Channel Priority Register 0	16[2]	S, M
0x30 5C1E	S	CPR1_C	TPU_C Channel Priority Register 1	16[2]	S, M
0x30 5C20	S	CISR_C	TPU_C Channel Interrupt Status Register	16	S, M
0x30 5C22	T	LR_C	TPU_C Link Register	16[2]	S, M
0x30 5C24	T	SGLR_C	TPU_C Service Grant Latch Register	16[2]	S, M
0x30 5C26	T	DCNR_C	TPU_C Decoded Channel Number Register	16[2]	S, M
0x30 5C28	S[4]	TPUMCR2_C	TPU_C Module Configuration Register 2	16[2]	S, M
0x30 5C2A	S	TPUMCR3_C	TPU_C Module Configuration Register 3	16, 32[2]	S, M
0x30 5C2C	T	ISDR_C	TPU_C Internal Scan Data Register	16, 32[2]	
0x30 5C2E	T	ISCR_C	TPU_C Internal Scan Control Register	16, 32[2]	
0x30 5C00 – 0x30 5C0E	S/U[3]	—	TPU_C Channel 0 Parameter Registers	16, 32[2]	
0x30 5C10 – 0x30 5C1E	S/U[3]	—	TPU_C Channel 1 Parameter Registers	16, 32[2]	
0x30 5C20 – 0x30 5C2E	S/U[3]	—	TPU_C Channel 2 Parameter Registers	16, 32[2]	
0x30 5C30 – 0x30 5C3E	S/U[3]	—	TPU_C Channel 3 Parameter Registers	16, 32[2]	
0x30 5C40 – 0x30 5C4E	S/U[3]	—	TPU_C Channel 4 Parameter Registers	16, 32[2]	
0x30 5C50 – 0x30 5C5E	S/U[3]	—	TPU_C Channel 5 Parameter Registers	16, 32[2]	

Table A-13 Time Processor Unit 3 C (TPU3_C) (Continued)
(See Table A-10 for Bit Descriptions) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 5C60 – 0x30 5C6E	S/U[3]	—	TPU_C Channel 6 Parameter Registers	16, 32[2]	
0x30 5C70 – 0x30 5C7E	S/U[3]	—	TPU_C Channel 7 Parameter Registers	16, 32[2]	
0x30 5C80 – 0x30 5C8E	S/U[3]	—	TPU_C Channel 8 Parameter Registers	16, 32[2]	
0x30 5C90 – 0x30 5C9E	S/U[3]	—	TPU_C Channel 9 Parameter Registers	16, 32[2]	
0x30 5CA0 – 0x30 5CAE	S/U[3]	—	TPU_C Channel 10 Parameter Registers	16, 32[2]	
0x30 5CB0 – 0x30 5CBF	S/U[3]	—	TPU_C Channel 11 Parameter Registers	16, 32[2]	
0x30 5CC0 – 0x30 5CCF	S/U[3]	—	TPU_C Channel 11 Parameter Registers	16, 32[2]	
0x30 5CD0 – 0x30 5CDF	S/U[3]	—	TPU_C Channel 11 Parameter Registers	16, 32[2]	
0x30 5CE0 – 0x30 5CEF	S/U[3]	—	TPU_C Channel 14 Parameter Registers	16, 32[2]	
0x30 5CF0 – 0x30 5CFF	S/U[3]	—	TPU_C Channel 15 Parameter Registers	16[2]	

NOTES:
1. Bit 10 (TPU3) and bit 11 (T2CSL) are write-once. Bits 1:2 (TCR1P) and bits 3:4 (TCR2P) are write-once if PWOD is not set in the TPUMCR3 register. This register cannot be accessed with a 32-bit read. It can only be accessed with an 8- or 16-bit read.
2. Some TPU registers can only be read or written with 16- or 32-bit accesses. 8-bit accesses are not allowed.
3. S/U = Supervisor accessible only if SUPV = 1 or unrestricted if SUPV = 0. Unrestricted registers allow both user and supervisor access. The SUPV bit is in the TPUMCR register.
4. Bits 9:10 (ETBANK), 14 (T2CF), and 15 (DTPU) are write-once.

Table A-14 MIOS14 (Modular Input/Output Subsystem)

Address	Access	Symbol	Register	Size	Reset
0x30 6000	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S[1]
0x30 6002	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 6004	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 6006	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6008	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 600A	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 600C	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S

Table A-14 MIOS14 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 600E	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6010	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 6012	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 6014	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 6016	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6018	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 601A	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 601C	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 601E	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6020	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 6022	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 6024	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 6026	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6028	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 602A	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 602C	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 602E	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6030	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X
0x30 6032	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 6034	S/U	MMCSMSCRD	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 6036	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 6038	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X
0x30 603A	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 603E	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 6040	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X

Table A-14 MIOS14 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 6042	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 6046	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 6050	S/U	MRTCSMFRCH	MRTCSM 32-Bit Counter High Buffer Register. See **Table 17-42** for bit descriptions.	16	U
0x30 6052	S/U	MRTCSMFRCL	MRTCSM 32-Bit Counter Low Buffer Register. See **Table 17-43** for bit descriptions.	16	U
0x30 6054	S/U	MRTCPR	MRTCSM Prescaler Counter Buffer Register. See **Table 17-44** for bit descriptions.	16	U
0x30 6056	S/U	MRTCSM	MRTCSM Status/Control Register. See **Table 17-45** for bit descriptions.	16	S
0x30 6058	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 605A	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6060	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6062	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6062	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6066	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 6068	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 606A	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 606E	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 6070	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6072	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6076	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 6078	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 607A	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 607E	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 6080	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 6082	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 6084	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S

Table A-14 MIOS14 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 6086	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6088	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 608A	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 608C	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 608E	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6090	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 6092	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 6094	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 6096	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 6098	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 609A	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 609C	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 609E	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 60A0	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 60A2	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 60A4	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 60A6	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 60A8	S/U	MPWMPERR	MPWMSM Period Register. See **Table 17-33** for bit descriptions.	16	S
0x30 60AA	S/U	MPWMPULR	MPWMSM Pulse Width Register. See **Table 17-34** for bit descriptions.	16	S
0x30 60AC	S/U	MPWMCNTR	MPWMSM Counter Register. See **Table 17-35** for bit descriptions.	16	S
0x30 60AE	S/U	MPWMSCR	MPWMSM Status/Control Register. See **Table 17-36** for bit descriptions.	16	S
0x30 60B0	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X
0x30 60B2	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 60B6	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 60B8	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X

Table A-14 MIOS14 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 60BA	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 60BE	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 60C0	S/U	MMCSMCNT	MMCSM Up-Counter Register. See **Table 17-18** for bit descriptions.	16	X
0x30 60C2	S/U	MMCSMML	MMCSM Modulus Latch Register. See **Table 17-19** for bit descriptions.	16	S
0x30 60C6	S/U	MMCSMSCR	MMCSM Status/Control Register. See **Table 17-20** for bit descriptions.	16	S
0x30 60DA	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60D8	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60DE	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 60E0	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60E2	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60E6	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 60E8	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60EA	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60EE	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 60F0	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 6F2	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60F6	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 60F8	S/U	MDASMAR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60FA	S/U	MDASMBR	MDASM DataA Register. See **Table 17-27** for bit descriptions.	16	S
0x30 60FE	S/U	MDASMSCR	MDASM Status/Control Register. See **Table 17-29** for bit descriptions.	16	S
0x30 6100	S/U	MPIOSMDR	MPIOSM Data Register. See **Table 17-40** for bit descriptions.	16	S
0x30 6102	S/U	MPIOSMDDR	MPIOSM Data Direction Register. See **Table 17-41** for bit descriptions.	16	S
0x30 6800	S/U	MIOS14TPCR	MIOS14 Test and Pin Control Register. See **Table 17-11** for bit descriptions.	16	X
0x30 6802	S/U	MIOS14VECT	MIOS14 Vector Register. See **Table 17-12** for bit descriptions.	16	X
0x30 6804	S/U	MIOS14VNR	MIOS14 Vector Register. See **Table 17-13** for bit descriptions.	16	S

Table A-14 MIOS14 (Modular Input/Output Subsystem) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 6806	S/U	MIOS14MCR	MIOS14 Module Configuration Register. See **Table 17-14** for bit descriptions.	16	X
0x30 6816	S/U	MCPSMSCR	MCPSM Status/Control Register. See **Table 17-15** for bit descriptions.	16	X
0x30 6C00	S/U	MIOS14SR0	MIOS14 Interrupt Statis Register. See **Table 17-2** for bit descriptions.	16	X
0x30 6C04	S/U	MIOS14ER0	MIOS14 Interrupt Enable Register. See **Table 17-3** for bit descriptions.	16	X
0x30 6C06	S/U	MIOS14RPR0	MIOS14 Request Pending Register. See **Table 17-4** for bit descriptions.	16	S
0x30 6C30	S/U	MIOS14LVL0	MIOS14 Interrupt Level 0 Register. See **Table 17-8** for bit descriptions.	16	S
0x30 6C40	S/U	MIOS14SR1	MIOS14 Interrupt Status Register. See **Table 17-5** for bit descriptions.	16	X
0x30 6C44	S/U	MIOSER1	MIOS14 Interrupt Enable Register. See **Table 17-6** for bit descriptions.	16	X
0x30 6C46	S/U	MIOS14RPR1	MIOS14 Request Pending Register. See **Table 17-7** for bit descriptions.	16	X
0x30 6C70	S/U	MIOS14LVL1	MIOS14 Interrupt Level 1 Register. See **Table 17-9** for bit descriptions.	16	X

NOTES:
1. Only bits WEN, TEST, STB, and WIP affected by reset.

Table A-15 TouCAN A, B and C (CAN 2.0B Controller)

Address	Access	Symbol	Register	Size	Reset
TouCAN_A (Note: Bit descriptions apply to TouCAN_B and TouCAN_C as well)					
0x30 7080	S	TCNMCR_A	TouCAN_A Module Configuration Register. See **Table 16-11** for bit descriptions.	16	S
0x30 7082	T	TTR_A	TouCAN_A Test Register	16	S
0x30 7084	S	CANICR_A	TouCAN_A Interrupt Configuration Register. See **Table 16-12** for bit descriptions.	16	S
0x30 7086	S/U	CANCTRL0_A/ CANCTRL1_A	TouCAN_A Control Register 0/ TouCAN_A Control Register 1. See **Table 16-13** and **Table 16-16** for bit descriptions.	16	S
0x30 7088	S/U	PRESDIV_A/ CTRL2_A	TouCAN_A Control and Prescaler Divider Register/ TouCAN_A Control Register 2. See **Table 16-17** and **Table 16-18** for bit descriptions.	16	S
0x30 708A	S/U	TIMER_A	TouCAN_A Free-Running Timer Register. See **Table 16-19** for bit descriptions.	16	S
0x30 708C — 0x30 708E	—	—	Reserved	—	—
0x30 7090	S/U	RXGMASKHI_A	TouCAN_A Receive Global Mask High. See **Table 16-20** for bit descriptions.	32	S
0x30 7092	S/U	RXGMASKLO_A	TouCAN_A Receive Global Mask Low. See **Table 16-20** for bit descriptions.	32	S

Table A-15 TouCAN A, B and C (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 7094	S/U	RX14MASKHI_A	TouCAN_A Receive Buffer 14 Mask High. See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	32	S
0x30 7096	S/U	RX14MASKLO_A	TouCAN_A Receive Buffer 14 Mask Low. See **16.7.10 Receive Buffer 14 Mask Registers** for bit descriptions.	32	S
0x30 7098	S/U	RX15MASKHI_A	TouCAN_A Receive Buffer 15 Mask High. See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	32	S
0x30 709A	S/U	RX15MASKLO_A	TouCAN_A Receive Buffer 15 Mask Low. See **16.7.11 Receive Buffer 15 Mask Registers** for bit descriptions.	32	S
0x30 709C — 0x30 709E	—	—	Reserved	—	—
0x30 70A0	S/U	ESTAT_A	TouCAN_A Error and Status Register. See **Table 16-21** for bit descriptions.	16	S
0x30 70A2	S/U	IMASK_A	TouCAN_A Interrupt Masks. See **Table 16-24** for bit descriptions.	16	S
0x30 70A4	S/U	IFLAG_A	TouCAN_A Interrupt Flags. See **Table 16-25** for bit descriptions.	16	S
0x30 70A6	S/U	RXECTR_A/ TXECTR_A	TouCAN_A Receive Error Counter/ TouCAN_A Transmit Error Counter. See **Table 16-26** for bit descriptions.	16	S
0x30 7100 — 0x30 710F	S/U	MBUFF0_A[1]	TouCAN_A Message Buffer 0. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7110 — 0x30 711F	S/U	MBUFF1_A[1]	TouCAN_A Message Buffer 1. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7120 — 0x30 712F	S/U	MBUFF2_A[1]	TouCAN_A Message Buffer 2. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7130 — 0x30 713F	S/U	MBUFF3_A[1]	TouCAN_A Message Buffer 3. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7140 — 0x30 714F	S/U	MBUFF4_A[1]	TouCAN_A Message Buffer 4. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7150 — 0x30 715F	S/U	MBUFF5_A[1]	TouCAN_A Message Buffer 5. See **Figure 16-3** and **Figure 16-3** for message buffer definitions.	—	U
0x30 7160 — 0x30 716F	S/U	MBUFF6_A[1]	TouCAN_A Message Buffer 6. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x307170 — 0x30717F	S/U	MBUFF7_A[1]	TouCAN_A Message Buffer 7. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 7180 — 0x30 718F	S/U	MBUFF8_A[1]	TouCAN_A Message Buffer 8. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U

Table A-15 TouCAN A, B and C (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 7190 — 0x30 719F	S/U	MBUFF9_A[1]	TouCAN_A Message Buffer 9. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71A0 — 0x30 71AF	S/U	MBUFF10_A[1]	TouCAN_A Message Buffer 10. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71B0 — 0x30 71BF	S/U	MBUFF11_A[1]	TouCAN_A Message Buffer 11. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71C0 — 0x30 71CF	S/U	MBUFF12_A[1]	TouCAN_A Message Buffer 12. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71D0 — 0x30 71DF	S/U	MBUFF13_A[1]	TouCAN_A Message Buffer 13. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71E0 — 0x30 71EF	S/U	MBUFF14_A[1]	TouCAN_A Message Buffer 14. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
0x30 71F0 — 0x30 71FF	S/U	MBUFF15_A[1]	TouCAN_A Message Buffer 15. See **Figure 16-3** and **Figure 16-4** for message buffer definitions.	—	U
TouCAN_B					
0x30 7480	S	TCNMCR_B	TouCAN_B Module Configuration Register	16	S
0x30 7482	T	TTR_B	TouCAN_B Test Register	16	S
0x30 7484	S	CANICR_B	TouCAN_B Interrupt Configuration Register	16	S
0x30 7486	S/U	CANCTRL0_B/ CANCTRL1_B	TouCAN_B Control Register 0/ TouCAN_B Control Register 1	16	S
0x30 7488	S/U	PRESDIV_B/ CTRL2_B	TouCAN_B Control and Prescaler Divider Register/ TouCAN_B Control Register 2	16	S
0x30 748A	S/U	TIMER_B	TouCAN_B Free-Running Timer Register		S
0x30 748C — 0x30 748E	—	—	Reserved	—	—
0x30 7490	S/U	RXGMASKHI_B	TouCAN_B Receive Global Mask High	32	S
0x30 7492	S/U	RXGMASKLO_B	TouCAN_B Receive Global Mask Low	32	S
0x30 7494	S/U	RX14MASKHI_B	TouCAN_B Receive Buffer 14 Mask High	32	S
0x30 7496	S/U	RX14MASKLO_B	TouCAN_B Receive Buffer 14 Mask Low	3	S
0x30 7498	S/U	RX15MASKHI_B	TouCAN_B Receive Buffer 15 Mask High	32	S
0x30 749A	S/U	RX15MASKLO_B	TouCAN_B Receive Buffer 15 Mask Low	32	S
0x30 749C — 0x30 749E	—	—	Reserved	—	—
0x30 74A0	S/U	ESTAT_B	TouCAN_B Error and Status Register	16	S
0x30 74A2	S/U	IMASK_B	TouCAN_B Interrupt Masks	16	S
0x30 74A4	S/U	IFLAG_B	TouCAN_B Interrupt Flags	16	S
0x30 74A6	S/U	RXECTR_B/ TXECTR_B	TouCAN_B Receive Error Counter/ TouCAN_B Transmit Error Counter	16	S
0x30 7500 — 0x30 750F	S/U	MBUFF0_B[1]	TouCAN_B Message Buffer 0.	—	U
0x30 7510 — 0x30 751F	S/U	MBUFF1_B[1]	TouCAN_B Message Buffer 1.	—	U

Table A-15 TouCAN A, B and C (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 7520 — 0x30 752F	S/U	MBUFF2_B[1]	TouCAN_B Message Buffer 2.	—	U
0x30 7530 — 0x30 753F	S/U	MBUFF3_B[1]	TouCAN_B Message Buffer 3.	—	U
0x30 7540 — 0x30 754F	S/U	MBUFF4_B[1]	TouCAN_B Message Buffer 4.	—	U
0x30 7550 — 0x30 755F	S/U	MBUFF5_B[1]	TouCAN_B Message Buffer 5.	—	U
0x30 7560 — 0x30 756F	S/U	MBUFF6_B[1]	TouCAN_B Message Buffer 6.	—	U
0x30 7570 — 0x30 757F	S/U	MBUFF7_B[1]	TouCAN_B Message Buffer 7.	—	U
0x30 7580 — 0x30 758F	S/U	MBUFF8_B[1]	TouCAN_B Message Buffer 8.	—	U
0x30 7590 — 0x30 759F	S/U	MBUFF9_B[1]	TouCAN_B Message Buffer 9.	—	U
0x30 75A0 — 0x30 75AF	S/U	MBUFF10_B[1]	TouCAN_B Message Buffer 10.	—	U
0x30 75B0 — 0x30 75BF	S/U	MBUFF11_B[1]	TouCAN_B Message Buffer 11.	—	U
0x30 75C0 — 0x30 75CF	S/U	MBUFF12_B[1]	TouCAN_B Message Buffer 12.	—	U
0x30 75D0 — 0x30 75DF	S/U	MBUFF13_B[1]	TouCAN_B Message Buffer 13.	—	U
0x30 75E0 — 0x30 75EF	S/U	MBUFF14_B[1]	TouCAN_B Message Buffer 14.	—	U
0x30 75F0 — 0x30 75FF	S/U	MBUFF15_B[1]	TouCAN_B Message Buffer 15.	—	U
TouCAN_C					
0x30 7880	S	TCNMCR_C	TouCAN_C Module Configuration Register	16	S
0x30 7882	T	TTR_C	TouCAN_C Test Register	16	S
0x30 7884	S	CANICR_C	TouCAN_C Interrupt Configuration Register	16	S
0x30 7886	S/U	CANCTRL0_C/ CANCTRL1_C	TouCAN_C Control Register 0/ TouCAN_C Control Register 1	16	S
0x30 7888	S/U	PRESDIV_C/ CTRL2_C	TouCAN_C Control and Prescaler Divider Register/ TouCAN_C Control Register 2	16	S
0x30 788A	S/U	TIMER_C	TouCAN_C Free-Running Timer Register		S
0x30 788C — 0x30 788E	—	—	Reserved	—	—
0x30 7890	S/U	RXGMASKHI_C	TouCAN_C Receive Global Mask High	32	S
0x30 7892	S/U	RXGMASKLO_C	TouCAN_C Receive Global Mask Low	32	S
0x30 7894	S/U	RX14MASKHI_C	TouCAN_C Receive Buffer 14 Mask High	32	S
0x30 7896	S/U	RX14MASKLO_C	TouCAN_C Receive Buffer 14 Mask Low	32	S
0x30 7898	S/U	RX15MASKHI_C	TouCAN_C Receive Buffer 15 Mask High	32	S
0x30 789A	S/U	RX15MASKLO_C	TouCAN_C Receive Buffer 15 Mask Low	32	S
0x30 789C — 0x30 789E	—	—	Reserved	—	—
0x30 78A0	S/U	ESTAT_C	TouCAN_C Error and Status Register	16	S

Table A-15 TouCAN A, B and C (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0x30 78A2	S/U	IMASK_C	TouCAN_C Interrupt Masks	16	S
0x30 78A4	S/U	IFLAG_C	TouCAN_C Interrupt Flags	16	S
0x30 78A6	S/U	RXECTR_C/ TXECTR_C	TouCAN_C Receive Error Counter/ TouCAN_C Transmit Error Counter	16	S
0x30 7500 — 0x30 750F	S/U	MBUFF0_C[1]	TouCAN_C Message Buffer 0.	—	U
0x30 7510 — 0x30 751F	S/U	MBUFF1_C[1]	TouCAN_B Message Buffer 1.	—	U
0x30 7520 — 0x30 752F	S/U	MBUFF2_C[1]	TouCAN_C Message Buffer 2.	—	U
0x30 7530 — 0x30 753F	S/U	MBUFF3_C[1]	TouCAN_C Message Buffer 3.	—	U
0x30 7540 — 0x30 754F	S/U	MBUFF4_C[1]	TouCAN_C Message Buffer 4.	—	U
0x30 7550 — 0x30 755F	S/U	MBUFF5_C[1]	TouCAN_C Message Buffer 5.	—	U
0x30 7560 — 0x30 756F	S/U	MBUFF6_C[1]	TouCAN_C Message Buffer 6.	—	U
0x30 7570 — 0x30 757F	S/U	MBUFF7_C[1]	TouCAN_C Message Buffer 7.	—	U
0x30 7580 — 0x30 758F	S/U	MBUFF8_C[1]	TouCAN_C Message Buffer 8.	—	U
0x30 7590 — 0x30 759F	S/U	MBUFF9_C[1]	TouCAN_C Message Buffer 9.	—	U
0x30 75A0 — 0x30 75AF	S/U	MBUFF10_C[1]	TouCAN_C Message Buffer 10.	—	U
0x30 75B0 — 0x30 75BF	S/U	MBUFF11_C[1]	TouCAN_C Message Buffer 11.	—	U
0x30 75C0 — 0x30 75CF	S/U	MBUFF12_C[1]	TouCAN_C Message Buffer 12.	—	U
0x30 75D0 — 0x30 75DF	S/U	MBUFF13_C[1]	TouCAN_C Message Buffer 13.	—	U
0x30 75E0 — 0x30 75EF	S/U	MBUFF14_C[1]	TouCAN_C Message Buffer 14.	—	U
0x30 75F0 — 0x30 75FF	S/U	MBUFF15_C[1]	TouCAN_C Message Buffer 15.	—	U

NOTES:

1. The last word of each of the the MBUFF arrays (address 0x....E) is reserved and may cause a RCPU exception if read.

Table A-16 UIMB (U-Bus to IMB Bus Interface)

Address	Access	Symbol	Register	Size	Reset
0x30 7F80	S[1]	UMCR	UIMB Module Configuration Register. See **Table 12-6** for bit descriptions.	32	H
0x30 7F84 — 0x30 7F8C	—	—	Reserved	32	H
0x30 7F90	S/T	UTSTCREG	UIMB Test Control Register. Reserved	32	H
0x30 7F94 — 0x30 7F9C	—	—	Reserved	32	H
0x30 7FA0	S	UIPEND	Pending Interrupt Request Register. See **12.5.3 Pending Interrupt Request Register (UIPEND)** and **Table 12-7** for bit descriptions.	32	H

NOTES:
1. S = Supervisor mode only, T = Test mode only

Table A-17 CALRAM_A and CALRAM_B Control Registers

Address	Access	Symbol	Register	Size	Reset
CALRAM_A					
0x38 0000	S	CRAMMCR_A	CALRAM_A Module Configuration Register. See **Table 21-3** for bit descriptions.	32	S
0x38 0004	S	CRAMTST_A	CALRAM_A Test Register.	32	S
0x38 0008	S	CRAM_RBA0_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 000C	S	CRAM_RBA1_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0010	S	CRAM_RBA2_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0014	S	CRAM_RBA3_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0018	S	CRAM_RBA4_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 001C	S	CRAM_RBA5_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0020	S	CRAM_RBA6_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S

Table A-17 CALRAM_A and CALRAM_B Control Registers (Continued)

Address	Access	Symbol	Register	Size	Reset
0x38 0024	S	CRAM_RBA7_A	CALRAM_A Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0028	S	CRAM_OLVCR_A	CALRAM_A Overlay Configuration Register. See **Table 21-7** for bit descriptions.	32	S
0x38 002C	S[1]	READI_OTR	READI Ownership Trace Register. See **23.2.1.1 User Mapped Register** for more information.	32	S
CALRAM_B					
0x38 0040	S	CRAMMCR_B	CALRAM_B Module Configuration Register. See **Table 21-3** for bit descriptions.	32	S
0x38 0044	S	CRAMTST_B	CALRAM_B Test Register. See **Table 21-3** for bit descriptions.	32	S
0x38 0048	S	CRAM_RBA0_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 004C	S	CRAM_RBA1_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0050	S	CRAM_RBA2_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0054	S	CRAM_RBA3_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0058	S	CRAM_RBA4_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 005C	S	CRAM_RBA5_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0060	S	CRAM_RBA6_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	15	S
0x38 0064	S	CRAM_RBA7_B	CALRAM_B Region Base Address Register. See **21.4.2 CALRAM Region Base Address Registers (CRAM_RBAx)** for more information.	32	S
0x38 0068	S	CRAM_OLVCR_B	CALRAM_A Overlay Configuration Register. See **Table 21-7** for bit descriptions.	32	S

Table A-17 CALRAM_A and CALRAM_B Control Registers (Continued)

Address	Access	Symbol	Register	Size	Reset
0x38 006C	S[1], [2]	CRAM_OTR_B	CALRAM_B Ownership Trace Register. See **21.4.4 CALRAM Ownership Trace Register (CRAMOTR)** for more information.	32	S

NOTES:
1. This register is write only.
2. This register is not used on the MPC565.

Table A-18 CALRAM_B and CALRAM_A Array

Address	Access	Symbol	Register	Size	Reset
CALRAM_B					
0x3F 7000 — 0x3F 7FFF	U,S	CRAM_B	CALRAM Array B.	4 Kbytes	—
CALRAM_A					
0x3F 8000 — 0x3F FFFF	U,S	CRAM_A	CALRAM Array A.	32 Kbytes	—

APPENDIX B
REGISTER GENERAL INDEX

–A–

Associated registers 10-4

–B–

BAR (breakpoint address register) 22-45
BBCMCR BBC module configuration register 4-25
BR0 (BR3 - memory controller base registers 0 - 3) 10-29
Breakpoint address register (BAR) 22-45
Breakpoint counter A value and control register (COUNTA) 22-50
Breakpoint counter B value and control register (COUNTB) 22-51

–C–

CALRAM_OTR CALRAM ownership trace register 21-21
CANCTRL0 (control register 0) 16-25
CANCTRL1 (control register 1) 16-26
CANCTRL2 (control register 2) 16-28
CANICR (TouCAN interrupt configuration register) 16-24
CANMCR (TouCAN module configuration register) 16-22
CFSR0 (TPU3 channel function select register 0) 18-17
CFSR1 (TPU3 channel function select register 1) 18-17
CFSR2 (TPU3 channel function select register 2) 18-17
CFSR3 (TPU3 channel function select register 3) 18-17
CIER (TPU3 channel interrupt enable register) 18-16
CISR (TPU3 channel interrupt status register) 18-20
CMPA-CMPD (comparator A-D value register) 22-44
CMPE-CMPF (comparator E-F value registers) 22-45
CMPG-CMPH (comparator G-H value registers) 22-45
COLIR
    change of lock interrupt register 8-36
COLIR (change of lock register) 8-36
Comparator A-D value registers (CMPA-CMPD) 22-44
Comparator E-F value registers (CMPE-CMPF) 22-45
Comparator G-H value registers (CMPG-CMPH) 22-45
control register (QACR1) 13-18
COUNTA (breakpoint counter A value and control register) 22-50
COUNTB (breakpoint counter B value and control register) 22-51
CPR0 (TPU3 channel priority register 0) 18-19
CPR1 (TPU3 channel priority register 1) 18-20
CR (condition register) 3-16
CRAM_RBAx CALRAM region base address register 21-18, 21-19
CRAMMCR CALRAM module configuration register 6-36, 8-30, 21-15, 21-16
CRAMOVL CALRAM overlay configuration register 21-20
CTR (count register) 3-19

–D–

DAR (data address register) 3-22
DCCR0-DCCR15 decompressor class configuration registers 4-32
DDRQS (PORTQS data direction register) 14-14

Debug enable register (DER) 22-53
DEC (decrementer register) 3-24, 6-39
Decrementer register (DEC) 6-39
DER (debug enable register) 22-53
Development port data register (DPDR) 22-55
DLCMD2 command register (CMD) 15-35, 15-36
DLCMD2 interrupt level register (ILR) 15-31, 15-32
DLCMD2 interrupt pending register (IPR) 15-30
DLCMD2 interrupt vector register (IVR) 15-32
DLCMD2 module configuration register (MCR) 15-27
DLCMD2 receive data register (RDATA) 15-34, 15-44
DLCMD2 status register (STAT) 15-41
DLCMD2 symbol timing control and pre-scaler register (SCTL) 15-32
DLCMD2 test configuration register (TCR) 15-58
DLCMD2 transmit data register (TDATA) 15-40
DMBR (dual mapping base register) 10-33
DMOR (dual mapping option register) 10-34
DPTMCR (DPTRAM module configuration register) 19-4
DPTRAM
    module configuration register (DPTMCR) 19-4
    ram base address register (RAMBAR) 19-5
    test register 19-5
DSCR (TPU3 development support control register) 18-13
DSISR (dae/source instruction service register) 3-22
DSSR (TPU3 development support status register) 18-15
Dual mapping base register (DMBR) 10-33
Dual mapping option register 10-34

–E–

ECR (exception cause register) 22-52
EIBADR external interrupt relocation table base address register 4-31
EMCR (external master control register) 6-29
ESTAT (error and status register) 16-30
Exception cause register (ECR) 22-51
External master control register (EMCR) 6-29

–F–

FASRAM module test register (FTEST) 17-42
FPRs - (floating-point registers) 3-12
FPSCR (floating-point status and control register) 3-13

–G–

General-Purpose I/O registers 6-45
GPRs (general-purpose registers) 3-12

–H–

HSQR0 (TPU3 host sequence register 0) 18-18
HSQR1 (TPU3 host sequence register 1) 18-18
HSSR0 (TPU3 host service request register 0) 18-19
HSSR1 (TPU3 host service request register 1) 18-19

–I–

I-Bus support control register (ICTRL) 22-46
ICTRL (i-bus support control register) 22-46
IFLAG (interrupt flag register) 16-33
IMASK (interrupt mask register) 16-32
IMMR (internal memory mapping register) 6-28

Internal memory map register 6-28

–K–

Keep alive power registers lock mechanism 8-24

–L–

L2U
global region attribute register (L2U_GRA) 11-16
module configuration register (L2U_MCR) 11-13
region attribute registers (L2U_RAx) 11-15
region base address registers (L2U_RBAx) 11-14
L2U_GRA (L2U global region attribute register) 11-16
L2U_MCR (L2U module configuration register) 11-14
L2U_RAx (L2U region X attribute register) 11-15
L2U_RBAx (L2U region x base address register) 11-14
L-Bus support control register 1 (LCTRL1) 22-48
L-Bus support control register 2 (LCTRL2) 22-49
LCTRL1 (l-bus support control register 1) 22-48
LCTRL2 (l-bus support control register 2) 22-49
LR (link register) 3-19

–M–

MCPSMCR MCPSM status/control register 17-36
MDASMAR MDASM DataA register 17-60
MDASMBR MDASM DataB register 17-61
MDASMSCR MDASM status/control register 17-62
Memory controller base registers (BR0 - BR3) 10-29
Memory controller option registers (OR0 - OR3) 10-31
Memory controller status registers (MSTAT) 10-29
MI_GRA Global region attribute register 4-29
MI_GRA Global regionattribute register 4-29
MIOS14ER0 interrupt enable register 17-27
MIOS14ER1 interrupt enable register 17-29
MIOS14LVL0 interrupt register 17-30
MIOS14LVL1 register 17-30
MIOS14MCR module configuration register 17-33, 17-34
MIOS14RPR0 request pending register 17-28
MIOS14RPR1 request pending register 17-29
MIOS14SR0 interrupt status register 17-27, 23-8
MIOS14SR1 interrupt status register 17-28
MIOS14TPCR test and pin control register 17-32
MIOS14VECT vector register 17-33
MIOS14VNR module-version number register 17-33
MISCNT (MISC counter) 19-6, 20-7, 20-10, 20-13
MISRH (multiple input signature register high) 19-6
MISRL (multiple input signature register low) 19-6
MMCSMCNT MMCSM up-counter register 17-42
MMCSMMML MMCSM modulus latch register 17-42
MMCSMSCR MMCSM status/control register 17-43
MPIOSMDDR MPIOSM data direction register 17-82
MPIOSMDR MPIOSM data register 17-82
MPWMCNTR MPWMSM counter regiser 17-76
MPWMPERR MPWMSM period regiser 17-74, 17-75
MPWMPULR MPWMSM pulse width regiser 17-75
MPWMSCR MPWMSM status/control regiser 17-76, 17-77
MRTCPR MRTCSM prescaler counter buffer register 17-93
MRTCSMFRCH MRTCSM 32-bit counter high buffer register 17-92
MRTCSMFRCL MRTCSM 32-bit counter low buffer register 17-93

MRTCSMSCR MRTCSM status/control register 17-93
MSR (machine state register) 3-20
MSTAT (memory controller status register) 10-29

–O–

OR0 (OR3 - memory controller option registers 0 - 3) 10-31

–P–

PDMCR pads module configuration register 2-3, 2-4
PDMCR2 pads module configuration register 2-5
Periodic interrupt status and control register (PISCR) 6-43
Periodic interrupt timer count register (PITC) 6-43
Periodic interrupt timer register (PITR) 6-44
PISCR (periodic interrupt status and control register) 6-43
PITC (periodic interrupt timer count) 6-43
PITR (periodic interrupt timer register) 6-44
PLL,
    low power, and reset control register (PLPRCR) 8-33
PLPRCR (PLL, low power, and reset control register) 8-34
port data direction registers 13-15
port data registers 13-15
PORTQS (port QS data register) 14-12
PQSPAR (PORTQS pin assignment register) 14-13
PRESDIV (prescaler divide register) 16-27
PVR (processor version register) 3-26

–Q–

QADC64E control register 0 (QACR0) 13-16
QADC64E control register 1 (QACR1) 13-18
QADC64E control register 2 (QACR2) 13-20
QADC64E conversion command word table (CCW) 13-33
QADC64E left justified, signed result format (LJSRR) 13-37
QADC64E left justified, unsigned result format LJURR) 13-37
QADC64E port A data direction registers (DDRQA) 13-16
QADC64E port A data register (PORTQA) 13-15
QADC64E port B data direction register (DDRQB) 13-16
QADC64E port B data registers (PORTQB) 13-15
QADC64E right justified, unsigned result format (RJURR) 13-37
QADC64E status register 0 (QASR0) 13-23
QADC64E status register 1 (QASR1) 13-29
QADCINT-QADC64E interrupt register (with IACK(C)/with ILBS(D) QADCINT 13-13, 13-14
QDSCI_IL (QSM2 dual SCI interrupt level register) 14-9
QSCI1CR (QSCI1 control register) 14-60
QSCI1SR (QSCI1 status register) 14-62
QSMCM
    configuration register (QMCMMCR) 14-9
    interrupt level registers (QDSCI_IL, QSPI_IL) 14-9
    port QS data register (PORTQS) 14-11
    PORTQS data direction register (DDRQS) 14-13
    PORTQS pin assignment register (PQSPAR) 14-12
    QSCI1 control register (QSCI1CR) 14-60
    QSCI1 status register (QSCI1SR) 14-62
    QSPI command RAM (CRx) 14-23
    QSPI control register 0 (SPCR0) 14-17
    QSPI control register 1 (SPCR1) 14-19
    QSPI control register 2 (SPCR2) 14-20
    QSPI control register 3 (SPCR3) 14-20
    QSPI registers 14-16

QSPI status register (SPSR) 14-21
queued SCI1 status and control registers 14-60
SCI control register 0 (SCCxR0) 14-46
SCI control register 1 (SCCxR1) 14-46
SCI data register (SCxDR) 14-50
SCI registers 14-45
SCI status register (SCxSR) 14-48
test register (QTEST) 14-9
QSMCMMCR (QSMCM module configuration register) 14-9
QSPI_IL (QSPI interrupt level register) 14-10

–R–

RAMBAR (ram array base address register) 19-5
RCPU
additional implementation-specific registers 3-27
condition register (CR) 3-15
condition register CR0 field definition 3-16
condition register CR1 field definition 3-16
condition register crn field - compare instruction 3-17
count register (CTR) 3-19
dae/source instruction service register (DSISR) 3-22
data address register (DAR) 3-22
decrementer register (DEC) 3-23
EIE, EID, and NRI special-purpose registers 3-26
floating-point exception cause register (FPECR) 3-26
floating-point registers (FPRs) 3-12
floating-point status and control register (FPSCR) 3-12
general special-purpose registers (SPRG0-SPRG3) 3-25
general-purpose registers (GPRs) 3-12
implementation-specific special-purpose registers 3-26
integer exception register (XER) 3-17
link register (LR) 3-18
machine state register (MSR) 3-20
machine status save/restore register 0 (SRR0) 3-24
machine status save/restore register 1 (SRR1) 3-24
powerpc OEA register set 3-20
powerpc UISA register set 3-11
powerpc VEA register set - time base 3-19
processor version register (PVR) 3-25
READI data trace attributes 1 and 2 register 23-16
READI development control register 23-10
READI device ID register 23-9
READI DID 23-9, 23-10
READI DTA 1 and 2 23-16
READI read/write access register 23-13
READI RWA 23-13
READI UBA 23-11, 23-12, 23-15
READI user base accress register 23-11, 23-12, 23-15
READI user base address register 23-11, 23-12, 23-15
READI_OTR READI ownership trace register 23-8
Real-Time clock alarm register (RTCAL) 6-42
Real-Time clock register (RTC) 6-42
Real-Time clock status and control register (RTCSC) 6-41
Region attribute registers (0 - 3) 4-27
Region base address registers (0 - 3) 4-27
Register descriptions
SIUMCR SIU module configuration register 6-25
Reset status register (RSR) 7-5
RSR (reset status register) 7-5
RTC (real time clock alarm register) 6-42

RTC (real time clock register) 6-42
RTCSC (real time clock status and control register) 6-41
RXECTR (receive error counter) 16-33
RXGMSKHI (receive global mask register high) 16-29

–S–

SCCxR0 (QSMCM SCI control register 0) 14-46
SCCxR1 (QSMCM SCI control register 1) 14-47
SCDR (QSMCM SCI data register) 14-50
SCxSR (QSMCM SCIx status register) 14-48
SGPIO
    control register (SGPIOCR) 6-46
    data register 1 (SGPIODT1) 6-45
    data register 2 (SGPIODT2) 6-45
SGPIOCR (SGPIO control register) 6-46
SGPIODT1 (SGPIO data register 1) 6-45
SGPIODT2 (SGPIO data register 2) 6-45
SIEL (SIU interrupt edge level register) 6-34
SIMASK (SIU interrupt mask register) 6-32, 6-33
SIPEND (SIU interrupt pending register) 6-31, 6-32
SIU
    interrupt edge level register (SIEL) 6-33
    interrupt mask register (SIMASK) 6-32
    interrupt registers 6-30
    interrupt vector register (SIVEC) 6-34
SIUMCR SIU module configuration register 6-25
SIVEC (SIU interrupt vector) 6-34
Software service register (SWSR) 6-37
SPCR0 (QSPI control register 0) 14-18
SPCR1 (QSPI control register 1) 14-19
SPCR2 (QSPI control register 2) 14-20
SPCR3 (QSPI control register) 14-21
SPRG0-SPRG3 (general special-purpose registers 0-3) 3-25
SPSR (QSPI status register) 14-21
SRR0 (machine status save/restore register 0) 3-24
SRR1 (machine status save/restore register 1) 3-25
status register (QASR) 13-23
SWSR (software service register) 6-38
SYPCR (system protection control register) 6-37
System clock control register (SCCR) 8-29
System configuration and protection registers 6-24
System configuration registers 6-25
System protection control register (SYPCR) 6-36
System protection registers 6-36
System timer registers 6-39

–T–

TB (time base) 3-19, 3-23, 6-40
TBREF0 (time base reference register 0) 6-40
TBREF1 (time base reference register 1) 6-40
TBSCR (time base control and status register) 6-41
TESR (transfer error status register) 6-38
TICR (TPU3 interrupt configuration register) 18-15
Time base control and status register (TBSCR) 6-40
Time base reference registers (TBREF0) 6-40
TIMER (free running timer register) 16-29
TouCAN
    control register 0 (CANCTRL0) 16-25
    control register 1 (CANCTRL1) 16-26

control register 2 (CANCTRL2) 16-28
error and status register (ESTAT) 16-30
interrupt configuration register (CANICR) 16-24
interrupt flag register (IFLAG) 16-33
interrupt mask register (IMASK) 16-32
module configuration register (CANMCR) 16-22
prescaler divide register (PRESDIV) 16-27
receive buffer 14 mask registers 16-30
receive buffer 15 mask registers 16-30
receive global mask registers (RXGMSKHI) 16-29
receive mask registers 16-7
test configuration register 16-24
TPU3
channel function select registers (CFSRx) 18-16
channel interrupt enable register (CIER) 18-16
channel interrupt status register (CISR) 18-20
channel priority registers (CPRx) 18-19
decoded channel number register (DCNR) 18-21
development support control register (DSCR) 18-13
development support status register (DSSR) 18-15
host sequence registers (HSQRx) 18-18
host service request registers (HSSRx) 18-18
interrupt configuration register (TICR) 18-15
link register (LR) 18-21
module configuration register (TPUMCR) 18-11
module configuration register 2 (TPUMCR2) 18-21
module configuration register 3 (TPUMCR3) 18-23
service grant latch register (SGLR) 18-21
test configuration register (TCR) 18-13
test registers (ISDR, ISCR) 18-23
TPUMCR (TPU3 module configuration register) 18-11
TPUMCR2 (TPU3 module configuration register 2) 18-21
TPUMCR3 (TPU3 module configuration register 3) 18-23
Transfer error status register (TESR) 6-38

–U–

UC3FCFIG (hard reset configuration word) 20-18, 20-19
UIMB
module configuration register (UMCR) 12-7
pending interrupt request register (UIPEND) 12-8
test control register (UTSTCREG) 12-8
UIPEND (UIMB pending interrupt reqiuest register) 12-8
UMCR (UIMB module configuration register) 12-7

–V–

VDDSRM
sensor register (VSRMSR) 8-36
VSRMSR (VDDSRM control register) 8-37

–X–

XER (integer exception register) 3-18

APPENDIX C
REGISTER DIAGRAM INDEX

–B–

BAR (breakpoint address register) 22-45
BBCMCR BBC module configuration register 4-25
BR0 (BR3 - memory controller base registers 0 - 3) 10-29

–C–

CALRLAM_OTR CALRAM ownership trace register 21-21
CANCTRL0 (control register 0) 16-25
CANCTRL1 (control register 1) 16-26
CANCTRL2 (control register 2) 16-28
CANICR (TouCAN interrupt configuration register) 16-24
CANMCR (TouCAN module configuration register) 16-22
CFSR0 (TPU3 channel function select register 0) 18-17
CFSR1 (TPU3 channel function select register 1) 18-17
CFSR2 (TPU3 channel function select register 2) 18-17
CFSR3 (TPU3 channel function select register 3) 18-17
CIER (TPU3 channel interrupt enable register) 18-16
CISR (TPU3 channel interrupt status register) 18-20
CMPA-CMPD (comparator A-D value register) 22-44
CMPE-CMPF (comparator E-F value registers) 22-45
CMPG-CMPH (comparator G-H value registers) 22-45
COLIR (change of lock interrupt register) 8-36
COUNTA (breakpoint counter A value and control register) 22-50
COUNTB (breakpoint counter B value and control register) 22-51
CPR0 (TPU3 channel priority register 0) 18-19
CPR1 (TPU3 channel priority register 1) 18-20
CR (condition register) 3-16
CRAM_RBAx CALRAM region base address register 21-19
CRAMMCR CALRAM module configuration register 6-36, 8-30, 21-16
CRAMOVL CALRAM overlay configuration register 21-20
CTR (count register) 3-19

–D–

DAR (data address register) 3-22
DDRQS (PORTQS data direction register) 14-14
DEC (decrementer register) 3-24, 6-39
DER (debug enable register) 22-53
DLCMD2 command regiser (CMD) 15-36
DLCMD2 interrupt level regiser (ILR) 15-31
DLCMD2 interrupt level register (ILR) 15-32
DLCMD2 interrupt pending regiser (IPR) 15-30
DLCMD2 Module configuration regiser (MCR) 15-27
DLCMD2 receive data regiser (RDATA) 15-34, 15-44
DLCMD2 status regiser (STAT) 15-41
DLCMD2 transmit data regiser (TDATA) 15-40
DMBR (dual mapping base register) 10-33
DMOR (dual mapping option register) 10-34
DPTMCR (DPTRAM module configuration register) 19-4
DSCR (TPU3 development support control register) 18-13
DSISR (dae/source instruction service register) 3-22

DSSR (TPU3 development support status register) 18-15

–E–

ECR (exception cause register) 22-52
EIBADR external interrupt relocation table base address register 4-31
EMCR (external master control register) 6-29
ESTAT (error and status register) 16-30

–F–

FASRAM module test register (FTEST) 17-42
FPRs - (floating-point registers) 3-12
FPSCR (floating-point status and control register) 3-13

–G–

GPRs (general-purpose registers) 3-12

–H–

HSQR0 (TPU3 host sequence register 0) 18-18
HSQR1 (TPU3 host sequence register 1) 18-18
HSSR0 (TPU3 host service request register 0) 18-19
HSSR1 (TPU3 host service request register 1) 18-19

–I–

ICTRL (i-bus support control register) 22-46
IFLAG (interrupt flag register) 16-33
IMASK (interrupt mask register) 16-32
IMMR (internal memory mapping register) 6-28

–L–

L2U_GRA (L2U global region attribute register) 11-16
L2U_MCR (L2U module configuration register) 11-14
L2U_RAx (L2U region X attribute register) 11-15
L2U_RBAx (L2U region x base address register) 11-14
LCTRL1 (l-bus support control register 1) 22-48
LCTRL2 (l-bus support control register 2) 22-49
LR (link register) 3-19

–M–

MCPSMSCR MCPSM status/control register 17-36
MDASMAR MDASM DataA register 17-60
MDASMBR MDASM DataB register 17-61
MDASMSCR MDASM status/control register 17-62
MI_GRA Global region attribute register 4-29
MIOS14ER1interrupt enable register 17-29
MIOS14LVL0 interrupt register 17-30
MIOS14LVL1 register 17-30
MIOS14MCR module configuration register 17-34
MIOS14RPR0 request pending register 17-28
MIOS14RPR1 request pending register 17-29
MIOS14SR0 interrupt status register 17-27
MIOS14SR1interrupt status register 17-28
MIOS14TPCR test and pin control register 17-32
MIOS14VEC vector register 17-33
MIOS14VNR module-version number register 17-33
MISCNT (MISC counter) 19-6, 20-7, 20-10, 20-13
MISRH (multiple input signature register high) 19-6

MISRL (multiple input signature register low) 19-6
MMCSMMML MMCSM modulus latch register 17-42
MMCSMSCR MMCSMstatus/control register 17-43
MPIOSMDDR MPIOSM data direction regiser 17-82
MPIOSMDR MPIOSM data regiser 17-82
MPWMCNTR MPWMSM counter register 17-76
MPWMPERR MPWMSM period register 17-75
MPWMPULR MPWMSM pulse width register 17-75
MPWMSCR MPWMSM status/control register 17-77
MRTCPR MRTCSM prescaler counter buffer register 17-93
MRTCSMFRCH MRTCSM32-bit counter high buffer register 17-92
MRTCSMFRCL MRTCSM32-bit counter low buffer register 17-93
MRTCSMSCR MRTCSM status/control register 17-93
MSR (machine state register) 3-20
MSTAT (memory controller status register) 10-29

–O–

OR0 (OR3 - memory controller option registers 0 - 3) 10-31

–P–

PDMCR pads module configuration register 2-4
PDMCR2 pads module configuration register 2-5
PISCR (periodic interrupt status and control register) 6-43
PITC (periodic interrupt timer count) 6-43
PITR (periodic interrupt timer register) 6-44
PLPRCR (PLL, low power, and reset control register) 8-34
PORTQS (port QS data register) 14-12
PQSPAR (PORTQS pin assignment register) 14-13
PRESDIV (prescaler divide register) 16-27
PVR (processor version register) 3-26

–Q–

QADC64E control register 0 (QACR0) 13-16
QADC64E control register 1 (QACR1) 13-18
QADC64E control register 2 (QACR2) 13-20
QADC64E conversion command word table register (CCW) 13-33
QADC64E leflt justified, unsigned result format register LJURR) 13-37
QADC64E left justified, signed result format register (LJSRR) 13-37
QADC64E port A data direction registers (DDRQA) 13-16
QADC64E port A data register (PORTQA) 13-15
QADC64E port B data direction register (DDRQB) 13-16
QADC64E port B data registers (PORTQB) 13-15
QADC64E right justified, unsigned result format register (RJURR) 13-37
QADC64E status register 0 (QASR0) 13-23
QADC64E status register 1(QASR1) 13-29
QADCINT-QADC64E interrupt register (with IACK(C)/with ILBS(D) QADCINT 13-14
QDSCI_IL (QSM2 dual SCI interrupt level register) 14-9
QSCI1CR (QSCI1 control register) 14-60
QSCI1SR (QSCI1 status register) 14-62
QSMCMMCR (QSMCM module configuration register) 14-9
QSPI_IL (QSPI interrupt level register) 14-10

–R–

RAMBAR (ram array base address register) 19-5
READI DC 23-10
READI DID 23-9
READI DTA 1 and 2 23-16
READI RWA 23-13

READI UBA 23-12, 23-15
READI_OTR READI ownership trace register 23-8
Region base address registers (0 - 3) 4-27
Regionattribute registers (0 - 3) 4-27
RSR (reset status register) 7-5
RTC (real time clock) 6-42
RTCAL (real time clock alarm) 6-42
RTCSC (real time clock status and control register) 6-41
RXECTR (receive error counter) 16-33
RXGMSKHI (receive global mask register high) 16-29

–S–

SCCxR0 (QSMCM SCI control register 0) 14-46
SCCxR1 (QSMCM SCI control register 1) 14-47
SCDR (QSMCM SCI data register) 14-50
SCxSR (QSMCM SCIx status register) 14-48
SGPIOCR (SGPIO control register) 6-46
SGPIODT1 (SGPIO data register 1) 6-45
SGPIODT2 (SGPIO data register 2) 6-45
SIEL (SIU interrupt edge level register) 6-34
SIMASK (SIU interrupt mask register) 6-32, 6-33
SIPEND (SIU interrupt pending register) 6-31, 6-32
SIVEC (SIU interrupt vector) 6-34
SPCR0 (QSPI control register 0) 14-18
SPCR1 (QSPI control register 1) 14-19
SPCR2 (QSPI control register 2) 14-20
SPCR3 (QSPI control register) 14-21
SPRG0-SPRG3 (general special-purpose registers 0-3) 3-25
SPSR (QSPI status register) 14-21
SRR0 (machine status save/restore register 0) 3-24
SRR1 (machine status save/restore register 1) 3-25
SWSR (software service register) 6-38
SYPCR (system protection control register) 6-37

–T–

TB (time base) 3-19, 3-23, 6-40
TBREF0 (time base reference register 0) 6-40
TBREF1 (time base reference register 1) 6-40
TBSCR (time base control and status register) 6-41
TESR (transfer error status register) 6-38
TICR (TPU3 interrupt configuration register) 18-15
TIMER (free running timer register) 16-29
TPUMCR (TPU3 module configuration register) 18-11
TPUMCR2 (TPU3 module configuration register 2) 18-21
TPUMCR3 (TPU3 module configuration register 3) 18-23

–U–

UC3FCFIG (hard reset configuration word) 20-19
UIPEND (UIMB pending interrupt reqiuest register) 12-8
UMCR (UIMB module configuration register) 12-7

–V–

VSRMSR (VDDSRM control register) 8-37

–X–

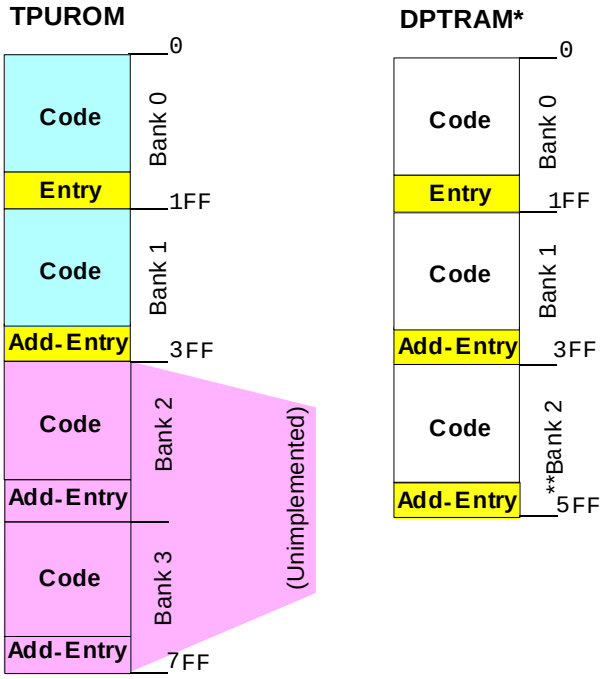
XER (integer exception register) 3-18

APPENDIX D
TPU ROM FUNCTIONS

The following pages provide brief descriptions of the pre-programmed functions in the TPU3. For detailed descriptions, refer to the programming note for the individual function. The Motorola **TPU Literature Pack, TPULITPAK/D**, provides a list of available programming notes.

D.1 Overview

The TPU3 contains four Kbytes of microcode ROM. This appendix defines the functions that are in the standard ROM on the MPC565 / MPC566. The TPU3 can have up to eight Kbytes of memory and a maximum of four entry tables (see **Figure D-1**).



*The DPTRAM is located at 0x30 2000 or 0x30 1000 until it is switched to emulation mode. In emulation mode, the DPTRAM is accessible by the TPUs only.
**DPTRAM_C does not have bank 2 of memory.

Figure D-1 TPU3 Memory Map

The TPU3 can address up to eight Kbytes of memory at any one time. It has four Kbytes of internal ROM, located in banks zero and one, and six Kbytes of dual ported SRAM (DPTRAM), located in banks zero, one and two. As only one type of memory

can be used at a time, the TPU3 must either use the internal ROM or the SRAM. Functions from both memory types cannot be used in conjunction.

A new feature of the TPU3 microcode ROM is the existence of two entry tables in the four Kbytes of internal ROM. Each entry table has a set of sixteen functions allowing definition of which of the two tables the TPU3 will be able to access. Only one table can be used at a time and functions from the two entry tables cannot be mixed. The default entry table is located in Bank 0. This table is identical to the standard microcode ROM in the TPU2, so any CPU code written for the TPU2 will work unchanged on the TPU3. The TPU2 and TPU3 ROMs are different than the original TPU ROM. The functions in the default entry table in Bank 0 are listed in **Table D-1**.

Table D-1 Bank 0 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	MCPWM	Multi-Channel Pulse Width Modulation
6	FQD	Fast Quadrature Decode
5	PPWA	Period/Pulse Width Accumulator
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	SPWM	Synchronized Pulse Width Modulation
0	SIOP	Serial Input/Output Port

The functions in the entry table in Bank 1 are listed in **Table D-2**.

Table D-2 Bank 1 Functions

Function Number	Function Nickname	Function Name
0xF	PTA	Programmable Time Accumulator
0xE	QOM	Queued Output Match
0xD	TSM	Table Stepper Motor
0xC	FQM	Frequency Measurement
0xB	UART	Universal Asynchronous Receiver/Transmitter
0xA	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	Reserved	
6	FQD	Fast Quadrature Decode
5	ID	Identification
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	RWTPIN	Read/Write Timers and Pin
0	SIOP	Serial Input/output Port

The functions in the Bank 1 entry table are identical to the bank zero entry table functions with three exceptions. Function 1, SPWM, has been replaced by RWTPIN. This is a function that allows a read and write to the TPU timebases and corresponding pin. Function 5, PPWA, is now an identification function in the second table. The microcode ROM revision number is provided by this function. Finally, Function 7, MCPWM, has been removed and left open for future use.

The CPU selects which entry table to use by setting the ETBANK field in the TPUMCR2 register. This register is write once after reset. Although one entry table is specified at start-up, it is possible, in some cases, to use functions from both tables without resetting the microcontroller. A customer may, for example, wish to use the ID function from bank one to verify the TPU microcode version but then use the MCPWM function from bank zero. As a customer will typically only run the ID function during system configuration, and not again after that, the bank one entry table can be changed to the bank zero entry table using the soft reset feature of the TPU3. The procedure should be:

1. Set ETBANK field in TPUMCR2 to 0b01 to select the entry table in Bank 1
2. Run the ID function
3. Stop the TPU3 by setting the STOP bit in the TPUMCR to one
4. Reset the TPU3 by setting the SOFTRST bit in the TPUMCR2 register
5. Wait at least nine clocks
6. Clear the SOFTRST bit in the TPUMCR2 register

The TPU3 stays in reset until the CPU clears the SOFTRST bit. After the SOFTRST bit has been cleared the TPU3 will be reset and the entry table in Bank 0 will be selected by default. To select the bank zero entry table write 0b00 to the ETBANK field

in TPUMCR 2. It is good practice to always initialize any write once register to ensure an incorrect value is not accidentally written.

The descriptions below document the functions listed in **Table D-1** (Bank 0) and **Table D-2** (Bank 1) of the TPU3 ROM module.

D.2 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The period accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request.

From one to 255 period measurements can be accumulated before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability. See Motorola TPU Progamming Note ***Programmable Time Accumulator TPU Function (PTA),*** **(TPUPN06/D)**.

Figure D-2 shows all of the host interface areas for the PTA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PTA Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – High Time Accumulate 01 – Low Time Accumulate 10 – Period Accumulate, Rising 11 – Period Accumulate, Falling	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Not Used 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	MAX_COUNT	PERIOD_COUNT
0x30YYW4	LAST_TIME	
0x30YYW6	ACCUM	
0x30YYW8	HW	
0x30YYWA	LW	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-2 PTA Parameters

D.3 Queued Output Match TPU Function (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin. See Motorola TPU Progamming Note ***Queued Output Match TPU Function (QOM),*** **(TPUPN01/D)**.

Figure D-3 shows all of the host interface areas for the QOM function. The bit encodings shown in **Table D-3** describe the corresponding fields in parameter RAM.

Table D-3 QOM Bit Encoding

A	Timebase Selection
0	Use TCR1 as Timebase
1	Use TCR2 as Timebase

	Edge Selection
0	Falling Edge at Match
1	Rising Edge at Match

B:C	Reference for First Match
00	Immediate TCR Value
01	Last Event Time
10	Value Pointed to by REF_ADDR
11	Last Event Time

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	COM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Single-Shot Mode 01 – Loop Mode 10 – Continuous Mode 11 – Continuous Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize, No Pin Change 10 – Initialize, Pin Low 11 – Initialize, Pin High	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8	7 6 5 4 3 2 1	0
0x30YYW0	REF_ADDR	B	LAST_OFF_ADDR	A
0x30YYW2	LOOP_CNT	(LAST_MATCH_TM)	OFF_PTR	C
0x30YYW4	OFFSET_1			;
0x30YYW6	OFFSET_2			;
0x30YYW8	OFFSET_3			;
0x30YYWA	OFFSET_4			;
0x30YYWC	OFFSET_5*			;
0x30YYWE	OFFSET_6*			;
0x30YY(W=1)0	OFFSET_7*			;
0x30YY(W=1)2	OFFSET_8*			;
	: :			:
0x30YY(W=1)14	OFFSET_14*			;

*Not Available On All Channels.

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-3 QOM Parameters

D.4 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing full defininition of the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. See Motorola TPU Progamming Note ***Table Stepper Motor TPU Function (TSM),*** **(TPUPN04/D)**.

Figure D-4 and **Figure D-5** show all of the host interface areas for the TSM function when operating in master and slave mode, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	TSM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	x0 – Rotate Pin_Sequence Once Between Steps x1 – Split Mode Acceleration Table 1x – Rotate Pin_Sequence Once Between Steps 1x – Rotate Pin_Sequence Twice Between Steps	0x30YY14 – 0x30YY16
1 0	Host Service Reques	00 – No Host Service (Reset Condition) 01 – Initialize, Pin Low 10 – Initialize, Pin High 11 – Move Request (Master Only)	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	DESIRED_POSITION															
0x30YYW2	CURRENT_POSITION															
0x30YYW4	TABLE_SIZE					CHANNEL_COUNTER			TABLE_INDEX							
0x30YYW6	SLEW_PERIOD															S
0x30YYW8	START_PERIOD															A
0x30YYWA	PIN_SEQUENCE															
0x30YYWC																
0x30YYWE																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-4 TSM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	TSM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	x0 – Rotate Pin_Sequence Once Between Steps x1 – Split Mode Acceleration Table 1x – Rotate Pin_Sequence Once Between Steps 1x – Rotate Pin_Sequence Twice Between Steps	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Initialize, Pin Low 10 – Initialize, Pin High 11 – Move Request (Master Only)	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YY(W+1)0	ACCEL_RATIO_2	ACCEL_RATIO_1
0x30YY(W+1)2	ACCEL_RATIO_4	ACCEL_RATIO_3
0x30YY(W+1)4	ACCEL_RATIO_6	ACCEL_RATIO_5
0x30YY(W+1)6	ACCEL_RATIO_8	ACCEL_RATIO_7
0x30YY(W+1)8	ACCEL_RATIO_10	ACCEL_RATIO_9
0x30YY(W+1)A	ACCEL_RATIO_12	ACCEL_RATIO_11
0x30YY(W+1)C*	ACCEL_RATIO_14*	ACCEL_RATIO_13*
	:	:
0x30YY(W+3)A*	ACCEL_RATIO_36*	ACCEL_RATIO_35*

*Optional Additional Parameters not Available in all cases.
Refer to Motorola Programming Note TPUPN04/D for details

= Written By CPU
= Written by CPU and TPU
W = Channel Number
= Written By TPU
= Unused Parameters
YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

Figure D-5 TSM Parameters — Slave Mode

D.5 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement. Measurement of slow pulses with noise rejection can be made with PTA. See Motorola TPU Progamming Note ***Frequent Measurement TPU Function (FQM),* (TPUPN03/D)**.

Figure D-6 shows all of the host interface areas for the FQM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Begin with Falling Edge, Single-Shot Mode 01 – Begin with Falling Edge, Continuous Mode 10 – Begin with Rising Edge, Single-Shot Mode 11 – Begin with Rising Edge, Continuous Mode	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize 11 –Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	
0x30YYW2	
0x30YYW4	CHANEL_CONTROL
0x30YYW6	WINDOW_SIZE
0x30YYW8	PULSE_COUNT
0x30YYWA	IN_WINDOW_ACCUMULATOR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Primary Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-6 FQM Parameters

D.6 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from one to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bidirectional UART channels running in excess of 9600 baud could be implemented on the TPU. See Motorola TPU Progamming Note ***Asynchronous Serial Interface TPU Function (UART),* (TPUPN07/D)**.

Figure D-7 and **Figure D-8** show all of the host interface areas for the UART function in transmitting and receiving modes, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	UART Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – No Parity 01 – No Parity 10 – Even Parity 11 – Odd Parity	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Transmit 11 – Receive	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	PARITY_TEMP
0x30YYW2	MATCH_RATE
0x30YYW4	TDRE | TRANSMIT_DATA_REG
0x30YYW6	DATA_SIZE
0x30YYW8	ACTUAL_BIT_COUNT
0x30YYWA	SHIFT_REGISTER
0x30YYWC	
0x30YYWA	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-7 UART Transmitter Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	UART Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – No Parity 01 – No Parity 10 – Even Parity 11 – Odd Parity	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Transmit 11 – Receive	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	PARITY_TEMP															
0x30YYW2	MATCH_RATE															
0x30YYW4	PE	RE	TRANSMIT_DATA_REG													
0x30YYW6	DATA_SIZE															
0x30YYW8	ACTUAL_BIT_COUNT															
0x30YYWA	SHIFT_REGISTER															
0x30YYWC																
0x30YYWA																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-8 UART Receiver Parameters

D.7 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels. See Motorola TPU Progamming Note ***New Input Capture/Input Transition Counter TPU Function (NITC),*** **(TPUPN08/D)**.

Figure D-9 shows all of the host interface areas for the NITC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	NITC Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Single-Shot Mode, No Links 01 – Continuous Mode, No Links 10 – Single-Shot Mode, Links 11 – Continuous Mode, Links	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize TCR Mode 10 – Initialize Parameter Mode 11 – Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12	11 10 9	8 7 6 5 4 3 2 1	0
0x30YYW0			CHANNEL_CONTROL	
0x30YYW2	START_LINK_CHANNEL	LINK_CHANNEL_COUNT	PARAM_ADDR	0
0x30YYW4	MAX_COUNT			
0x30YYW6	TRANS_COUNT			
0x30YYW8	FINAL_TRANS_TIME			
0x30YYWA	LAST_TRANS_TIME			
0x30YYWC				
0x30YYWE				

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-9 NITC Parameters

D.8 Multiphase Motor Commutation (COMM)

The COMM function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds. See Motorola TPU Progamming Note ***Multiphase Motor Commutation TPU Function (COMM),* (TPUPN09/D)**.

Figure D-10 and **Figure D-11** show all of the host interface areas for the COMM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Sensorless Match Update Mode 01 – Sensorless Match Update Mode 10 – Sensorless Link Update Mode 11 – Sensorled Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize or Force State 11 – Initialize or Force Immediate State Test	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		START_LINK_ CHANNEL	COUNTER_ADDR
0x30YYW2	NO_OF_STATES		STATE_NO
0x30YYW4	OFFSET		
0x30YYW6	UPDATE_PERIO		
0x30YYW8	UPPER		
0x30YYWA	LOWER		
0x30YYWC			
0x30YYWE			

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-10 COMM Parameters (Part 1 of 2)

	CONTROL BITS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YY(W + 1)0	LENGTH						STATE		0	PIN_CONFIG						
0x30YY(W + 1)2	LENGTH						STATE		1	PIN_CONFIG						
0x30YY(W + 1)4	LENGTH						STATE		2	PIN_CONFIG						
0x30YY(W + 1)6	LENGTH						STATE		3	PIN_CONFIG						
0x30YY(W + 1)8	LENGTH						STATE		4	PIN_CONFIG						
0x30YY(W + 1)A	LENGTH						STATE		5	PIN_CONFIG						
0x30YY(W + 1)C	LENGTH						STATE		6*	PIN_CONFIG						
0x30YY(W + 1)E	LENGTH						STATE		7*	PIN_CONFIG						
0x30YY(W + 2)0	LENGTH						STATE		8*	PIN_CONFIG						
0x30YY(W + 2)2	LENGTH						STATE		9*	PIN_CONFIG						
•								•								
•								•								
•								•								
0x30YY(W + 3)A	LENGTH						STATE		21*	PIN_CONFIG						

*Not Available on all Channels

= Written By CPU

= Written by CPU and TPU

W = Master Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-11 COMM Parameters (Part 2 of 2)

D.9 Hall Effect Decode (HALLD)

The HALLD function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding “option” switches. See Motorola TPU Progamming Note ***Hall Effect Decode TPU Function (HALLD),*** **(TPUPN10/D)**.

Figure D-12 shows all of the host interface areas for the HALLD function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Channel A 01 – Channel B 10 – Channel B 11 – Channel C (3-Channel Mode Only)	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Initialize, 2-Channel Mode 11 – Initialize, 3-Channel Mode	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	x – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		
0x30YYW2		
0x30YYW4		
0x30YYW6	DIRECTION (1)	
0x30YYW8		STATE_NO_ADDR (2)
0x30YYWA	PINSTATE	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

NOTES:
1. Channel A Only.
2. One Channel Only (Channel B in 2-Channel Mode, Channel C in 3-Channel Mode.

Figure D-12 HALLD Parameters

D.10 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge-aligned mode uses n + 1 TPU channels for n PWMs; center-aligned mode uses 2n + 1 channels. Center-aligned mode allows a user-defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications. See Motorola TPU Progamming Note ***Multichannel PWM TPU Function (MCPWM),*** **(TPUPN05/D)**.

Figure D-13 through **Figure D-18** show the host interface areas for the MCPWM function in each mode.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center-Aligned Mode 10 – Slave B Type Center-Aligned Mode 11 – Slave C Type Center-Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize as Slave (Inverted) 10 – Initialize, as Slave (Normal) 11 – Initialize as Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	IRQ_RATE	PERIOD_COUNT
0x30YYW4	LAST_RISE_TIME	
0x30YYW6	LAST_FALL_TIME	
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-13 MCPWM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	PERIOD
0x30YYW2	HIGH_TIME
0x30YYW4	
0x30YYW6	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR
0x30YYWA	FALL_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-14 MCPWM Parameters — Slave Edge-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1A
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	NXT_B_RISE_TIME	
0x30YYW4	NXT_B_FALL_TIME	
0x30YYW6	DEAD_TIME	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	HIGH_TIME
0x30YYW2	CURRENT_HIGH_TIME
0x30YYW4	TEMP_STORAGE
0x30YYW6	
0x30YYW8	B_FALL_TIME_PTR
0x30YYWA	B_RISE_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0A
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY0C – 0x30YY12
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY14 – 0x30YY16
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY18 – 0x30YY1A
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY1C – 0x30YY1E
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	PERIOD	
0x30YYW2	NXT_B_RISE_TIME	
0x30YYW4	NXT_B_FALL_TIME	
0x30YYW6	DEAD_TIME	HIGH_TIME_PTR
0x30YYW8	RISE_TIME_PTR	
0x30YYWA	FALL_TIME_PTR	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	MCPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	00 – Edge-Aligned Mode 01 – Slave A Type Center Aligned Mode 10 – Slave B Type Center Aligned Mode 11 – Slave C Type Center Aligned Mode	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Host Service (Reset Condition) 01 – Initialize As Slave (Inverted) 10 – Initialize As Slave (Normal) 11 – Initialize As Master	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	HIGH_TIME
0x30YYW2	CURRENT_HIGH_TIME
0x30YYW4	TEMP_STORAGE
0x30YYW6	
0x30YYW8	B_FALL_TIME_PTR
0x30YYWA	B_RISE_TIME_PTR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-18 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

D.11 Fast Quadrature Decode TPU Function (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free-running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function. See Motorola TPU Progamming Note ***Fast Quadrature Decode TPU Function (FQD),*** **(TPUPN02/D)**.

Figure D-19 and **Figure D-20** show the host interface areas for the FQD function for primary and secondary channels, respectively.

CONTROL BITS

Bits	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Primary Channel (Normal Mode) 01 – Secondary Channel (Normal Mode) 10 – Primary Channel (Fast Mode) 11 – Secondary Channel (Fast Mode)	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Read TCR1 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
	Channel Interrupt Enable	x – Not Used	0x30YY0A
0	Channel Interrupt Status	xx – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	EDGE_TIME
0x30YYW2	POSITION_COUNT
0x30YYW4	TCR1_VALUE
0x30YYW6	CHAN_PINSTATE
0x30YYW8	CORR_PINSTATE_ADDR
0x30YYWA	EDGE_TIME_LSB_ADDR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-19 FQD Parameters — Primary Channel

	NAME	CONTROL BITS OPTIONS	ADDRESSES
0 3 2 1 0	Channel Function Select	xxxx – FQD Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence Bits	00 – Primary Channel (Normal Mode) 01 – Secondary Channel (Normal Mode) 10 – Primary Channel (Fast Mode) 11 – Secondary Channel (Fast Mode)	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – Not Used 10 – Read TCR1 11 – Initialize	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
	Channel Interrupt Enable	x – Not Used	0x30YY0A
0	Channel Interrupt Status	xx – Not Used	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0x30YYW0	
0x30YYW2	
0x30YYW4	TCR1_VALUE
0x30YYW6	CHAN_PINSTATE
0x30YYW8	CORR_PINSTATE_ADDR
0x30YYWA	EDGE_TIME_LSB_ADDR
0x30YYWC	
0x30YYWE	

= Written By CPU

= Written by CPU and TPU

W = Channel Number

= Written By TPU

= Unused Parameters

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

Figure D-20 FQD Parameters — Secondary Channel

D.12 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator (PPWA) algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from one to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation.

Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From one to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (one to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter.

By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels. See Motorola TPU Progamming Note ***Period/Pulse-Width Accumulator TPU Function (PPWA),*** **(TPUPN11/D)**.

Figure D-21 shows the host interface areas and parameter RAM for the PPWA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PPWA Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Accumulate 24-Bit Periods, No Links 01 – Accumulate 16-Bit Periods, Links 10 – Accumulate 24-Bit Pulse Widths, No Links 11 – Accumulate 16-Bit Pulse Widths, Links	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Not Used 10 – Initialize 11 – Not Used	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	START_LINK_CHANNEL	LINK_CHANNEL_COUNT	CHANNEL_CONTROL
0x30YYW2	MAX_COUNT		PERIOD_COUNT
0x30YYW4	LAST_ACCUM		
0x30YYW6	ACCUM		
0x30YYW8	ACCUM_RATE		PPWA_UB
0x30YYWA	PPWA_LW		
0x30YYWC			
0x30YYWE			

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

NOTES:

1. The TPU does not check the value of LINK_CHANNEL_COUNT. If this parameter is NOT >0 and ≤ 8, results are unpredictible.
2. MAX_COUNT may be written at any time by the host CPU, but if the value written is ≤ PERIOD_COUNT, a period or pulse-width accumulation is terminated. If this happens, the number of periods over which the accumulation is done will not correspond to MAX_COUNT.

Figure D-21 PPWA Parameters

D.13 Output Compare (OC)

The output compare (OC) function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time
2. At a programmable delay time from a user-specified time
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} \times \text{Ratio}$$

where RATIO is a parameter supplied.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins. See Motorola TPU Progamming Note ***Output Compare TPU Function (OC),*** **(TPUPN12/D)**.

Figure D-22 shows the host interface areas and parameter RAM for the OC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	OC Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Channel Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	0x – Matches and Pulses Scheduled x1 – Only Read TCR1, TCR2	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service Request 01 – Host-Initiated Pulse 10 – Not Used 11 – Initialize, Continuous Pulses	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1	0
0x30YYW0		CHANNEL_CONTROL	
0x30YYW2	OFFSET		
0x30YYW4	RATIO	REF_ADDR1	0
0x30YYW6	REF_ADDR2	0 REF_ADDR3	0
0x30YYW8	REF_TIME		
0x30YYWA	ACTUAL_MATCH_TIME		
0x30YYWC	TCR1		
0x30YYWE	TCR2		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-22 OC Parameters

D.14 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation (PWM) waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin. See Motorola TPU Progamming Note ***Pulse-Width Modulation TPU Function (PWM),*** **(TPUPN17/D)**.

Figure D-23 shows the host interface areas and parameter RAM for the PWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	PWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	xx – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Immediate Update of PWM 10 – Initialize 11 – Not Used	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	OLDRIS	
0x30YYW4	PWMHI (1, 3)	
0x30YYW6	PWMPER (2, 3)	
0x30YYW8	PWMRIS	
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C for TPU_C.

= Written By TPU

= Unused Parameters

Figure D-23 PWM Parameters

D.15 Discrete Input/Output (DIO)

The DIO function allows a TPU channel to be used as a digital I/O pin.

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter:

1. when a transition occurs
2. when the CPU makes a request, or
3. when a rate specified in another parameter is matched

When a pin is used as a discrete output, it is set high or low only upon request by the CPU. See Motorola TPU Progamming Note ***Discrete Input/Output TPU Function (DIO),*** **(TPUPN18/D)**.

Figure D-24 shows the host interface areas for the DIO function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	DIO Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Update on Transition 01 – Update at Match Rate 10 – Update on HSR 11 11 – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – Not Used 01 – Drive Pin High 10 – Drive Pin Low 11 – Initialize	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
0x30YYW0		CHANNEL_CONTROL
0x30YYW2	PIN_LEVEL	
0x30YYW4	MATCH_RATE	
0x30YYW6		
0x30YYW8		
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-24 DIO Parameters

D.16 Synchronized Pulse-Width Modulation (SPWM)

The SPWM function generates a pulse-width modulated waveform (PWM). The CPU can change the period or high time of the waveform at any time. Three different operating modes allow the function to maintain complex timing relationships between channels without CPU intervention.

The SPWM output waveform duty cycle excludes 0% and 100%. If a PWM does not need to maintain a time relationship to another PWM, the PWM function should be used instead. See Motorola TPU Progamming Note ***Synchronized Pulse-Width Modulation TPU Function (SPWM),*** **(TPUPN19/D)**.

Figure D-25 and **Figure D-26** show all of the host interface areas for the SPWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	SPWM Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Mode 0 01 – Mode 1 10 – Mode 2 11 – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Bits	00 – No Host Service Request 01 – Not Used 10 – Initialize 11 – Immediate Update (Mode 1)	0x30YY18 – 0x30YY1A
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	0x30YY0A
0	Interrupt Status		0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	LASTRISE	CHANNEL_CONTROL
0x30YYW2	NEXTRISE	
0x30YYW4	HIGH_TIME	
0x30YYW6	PERIOD	
0x30YYW8		REF_ADDR1
0x30YYWA	DELAY	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-25 SPWM Parameters, Part 1 of 2

PARAMETER RAM (MODE 1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	LASTRISE								CHANNEL_CONTROL							
0x30YYW2	NEXTRISE															
0x30YYW4	HIGH_TIME															
0x30YYW6	DELAY															
0x30YYW8	REF_ADDR1								REF_ADDR2							
0x30YYWA	REF_VALUE															
0x30YYWC																
0x30YYWE																

PARAMETER RAM (MODE 2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30YYW0	LASTRISE								CHANNEL_CONTROL							
0x30YYW2	NEXTRISE															
0x30YYW4	HIGH_TIME															
0x30YYW6	PERIOD															
0x30YYW8	START_LINK_ CHANNEL					LINK_CHANNEL _COUNT			REF_ADDR1							
0x30YYWA	DELAY															
0x30YYWC																
0x30YYWE																

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-26 SPWM Parameters, Part 2 of 2

D.17 Read / Write Timers and Pin TPU Function (RWTPIN)

The RWTPIN TPU function enables the CPU to read both the TCR1 and TCR2 timer counters via locations in PRAM and selectively load TCR1 or TCR2 with a CPU supplied value contained in PRAM. The function also allows control of the pin state and direction of the RWTPIN channel.

A pin state parameter is maintained in PRAM and is updated upon every service request. It can contain a value of the current pinstate whether the pin is programmed as an input or output.

The function also receives links, and upon receipt will read the two TCRs into PRAM, updating the pinstate parameter. A maskable interrupt request to the CPU is generated.

The CPU can control the channel pin, the channel pin and the TCRs, or just the TCRs. To control the pin only, the ‘read TCR’ option is used and the values returned ignored. Controlling the TCRs without effect on the pin allows this function to be run on a TPU channel whose pin is being controlled by a different function running on another channel (e.g., a slave stepper motor channel). See Motorola TPU Progamming Note ***Using The TPU Function Library And TPU Emulation Mode,*** **(TPUPN00/D)**.

Figure D-27 shows all of the host interface areas for the PTA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	Read/Write Timers and Pin Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	XX – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Action 01 – Read TCRs and read/write pin 10 – Write TCR1, read TCRs and read/write pin 11 – Write TCR2, read TCRs and read/write pin	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0		
0x30YYW2		
0x30YYW4		CHANNEL_CONTROL
0x30YYW6	CHANNEL_PIN_STATE	
0x30YYW8	TCR1_VALUE	
0x30YYWA	TCR2_VALUE	
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

= Written By TPU

= Unused Parameters

YY = 41 For TPU_A 44 For TPU_B and 5C For TPU_C

Figure D-27 RWTPIN Parameters

D.18 ID TPU Function (ID)

This is a simple function that returns the version of the TPU ROM on the current device.

Figure D-28 shows all of the host interface areas for the ID function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	Identification Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Host Sequence	XX – Not Used	0x30YY14 – 0x30YY16
1 0	Host Service Request	00 – No Action 01 – Read Read TPU ROM version 10 – Not Used 11 – Not Used	0x30YY18 – 0x30YY1A
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
0	Channel Interrupt Enable	0 – Channel Interrupts Disabled 1 – Channel Interrupts Enabled	0x30YY0A
0	Channel Interrupt Status	0 – Channel Interrupt Not Asserted 1 – Channel Interrupt Asserted	0x30YY20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	TPU3_ID	ROM_REVISION
0x30YYW2		
0x30YYW4		
0x30YYW6		
0x30YYW8		
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

YY = 41 For TPU_A 44 For TPU_B and 5C For TPU_C

= Written By TPU

= Unused Parameters

Figure D-28 ID Parameters

D.19 Serial Input/Output Port (SIOP)

The serial input/output port (SIOP) TPU function uses two or three TPU channels to form a uni- or bidirectional synchronous serial port that can be used to communicate with a wide variety of devices. Features such as baud rate and transfer size are user programmable. The function can also produce a clock-only, when it uses just one channel.

The SIOP TPU function has been designed to closely resemble the SIOP hardware port found on some Motorola MCUs and can be used to add serial capabilities to a device without a serial port, or extend the capabilities of one with a hardware synchronous port.

SIOP operates in master mode (i.e., the TPU always generates the clock) and the following features are programmable:

1. Choice of clock-only (one channel), clock + transmit (two channels), clock + receive (two channels) or clock + transmit + receive (three channels) operating modes
2. Baud rate period is freely programmable over a 15-bit range of TCR1 counts
3. Selection of MSB or LSB first shift direction
4. Variable transfer size from one to 16 bits
5. Clock polarity is programmable

When a transfer of data is complete the SIOP function notifies the host CPU by issuing an interrupt request. The arrangement of the multiple SIOP channels is fixed: the data-out channel is the channel above the clock channel and the data-in channel is the channel below the clock channel. In clock-only or uni-directional mode, the unused TPU channels are free to run other TPU functions. Two possible SIOP configurations are show in **Figure D-29**.

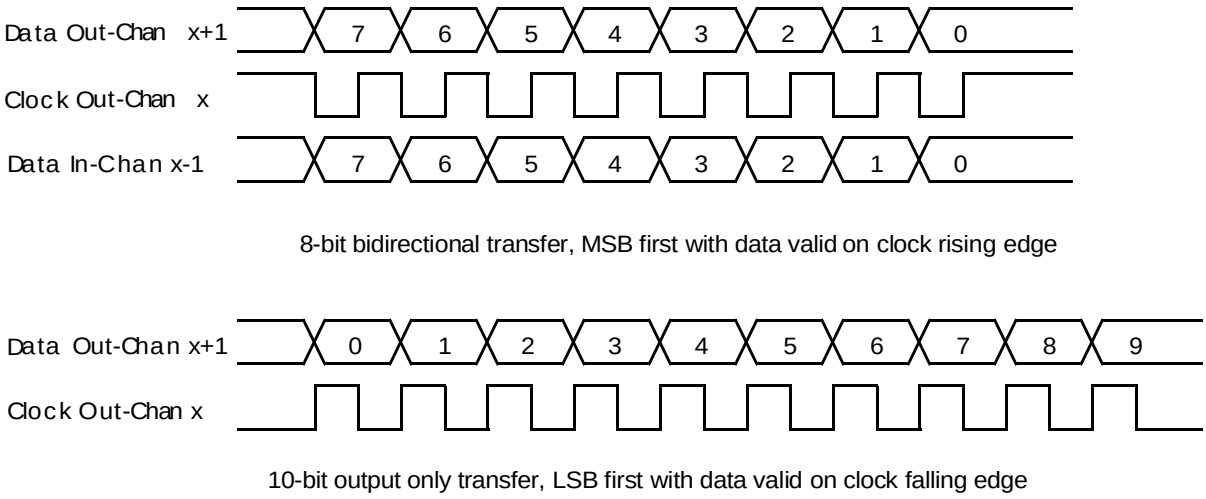


Figure D-29 Two Possible SIOP Configurations

D.19.1 Parameters

Figure D-30 shows the host interface areas and parameter RAM for the SIOP function. The following sections describe these parameters.

NOTE

Only the clock channel requires any programming. The data in and out channels are entirely under TPU microcode control.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	Channel Function Select	SIOP Function Number (Assigned During Microcode Assembly)	0x30YY0C – 0x30YY12
1 0	Channel Priority	00 – Disabled 01 – Low Priority 10 – Medium Priority 11 – High Priority	0x30YY1C – 0x30YY1E
1 0	Host Sequence Bits	00 – Clock Channel Active Only, No Data Transfer 01 – D_{OUT} Channels Active, No Data Receive 10 – Clock and D_{IN} Channels Active, No Data Transmit 11 – Full Bidirectional Tansmit and Receive	0x30YY14 – 0x30YY16 0x30YY18 – 0x30YY1A
1 0	Host Service Bits	00 – No Host Service (Reset Condition) 01 – No Action 10 – No Action 11 – Initialize Clock Channel and Start Transfer	0x30YY0A 0x30YY20
0	Interrupt Enable	0 – Interrupt Not Asserted 1 – Interrupt Asserted	**ADDRESSES**
0			0x30YY0C – 0x30YY12

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0x30YYW0	S	CHANNEL_CONTROL
0x30YYW2	HALF-PERIOD	
0x30YYW4		BIT_COUNT
0x30YYW6		XFER_SIZE
0x30YYW8	DATA	
0x30YYWA		
0x30YYWC		
0x30YYWE		

= Written By CPU

= Written by CPU and TPU

W = Channel Number

= Written By TPU

= Unused Parameters

YY = 41 For TPU_A, 44 For TPU_B and 5C For TPU_C

Figure D-30 SIOP Parameters

D.19.1.1 CHAN_CONTROL

This 9-bit CPU-written parameter is used to set up the clock polarity for the SIOP data transfer. The valid values for CHAN_CONTROL for this function are given in the table

below. CHAN_CONTROL must be written by the host prior to issuing the host service request (HSR) to initialize the function.

Table D-4 SIOP Function Valid CHAN_Control Options

CHAN_CONTROL[1] 8 7 6 5 4 3 2 1 0	Resulting Action
0 1 0 0 0 1 1 0 1	Data valid on clock Falling edge.
0 1 0 0 0 1 1 1 0	Data valid on clock Rising edge.

NOTES:
1. Other values of CHAN_CONTROL may result in indeterminate operation.

D.19.1.2 BIT_D

BIT_D is a CPU written bit that determines the direction of shift of the SIOP data. If BIT_D is zero then SIOP_DATA is right shifted (LSB first). If BIT_D is one then SIOP_DATA is left shifted (MSB first).

D.19.1.3 HALF_PERIOD

This CPU-written parameter defines the baud rate of the SIOP function. The value contained in HALF_PERIOD is the number of TCR1 counts for a half SIOP clock period (e.g., for a 50 baud rate, with a TCR1 period of 240 ns, the value [(1/50)/2]/240 ns = 42 should be written to HALF_PERIOD. The range for HALF_PERIOD is 1 to 0x8000, although the minimum value in practice will be limited by other system conditions. See the notes in **D.19.1.6 SIOP_DATA** for information on use and performance of SIOP function.

D.19.1.4 BIT_COUNT

This parameter is used by the TPU to count down the number bits remaining while a transfer is in progress. During the SIOP initialization state, BIT_COUNT is loaded with the value contained in XFER_SIZE. It is then decremented as the data is transferred and when it reaches zero, the transfer is complete and the TPU issues an interrupt request to the CPU.

D.19.1.5 XFER_SIZE

This CPU-written parameter determines the number of bits that make up a data transfer. During initialization, XFER_SIZE is copied into BIT_COUNT. XFER_SIZE is shown as a 5-bit parameter to match the maximum size of 16 bits in SIOP_DATA, although the TPU uses the whole word location. For normal use, XFER_SIZE should be in the 1- to 16-bit range.

D.19.1.6 SIOP_DATA

This parameter is the data register for all SIOP transfers. Data is shifted out of one end of SIOP_DATA and shifted in at the other end, the shift direction being determined by the value of BIT_D. In output-only mode, zero will be shifted into SIOP_DATA and in input-only mode, the data shifted out is ignored. In clock-only mode SIOP_DATA is still shifted.

NOTE 1

No ‘justifying’ of SIOP_DATA is performed by the TPU, (e.g., if an 8-bit bidirectional transfer is made, shifting LSB first, then the bottom byte of SIOP_DATA will be shifted out and the input data will be shifted into the upper byte of SIOP_DATA).

NOTE 2

SIOP_DATA is not buffered. The CPU should only access it between completion of one transfer and the start of the next.

D.19.2 Host CPU Initialization of the SIOP Function

The CPU initializes the SIOP function by:

1. Disabling the channel by clearing the two channel priority bits
2. Selecting the SIOP function on the channel by writing the assigned SIOP function number to the function select bits
3. Writing CHAN_CONTROL in the clock channel parameter RAM
4. Writing HALF_PERIOD, BIT_D and XFER_SIZE in the clock channel parameter RAM to determine the speed, shift direction and size of the transfer
5. Writing SIOP_DATA if the data output is to be used
6. Selecting the required operating mode via the two host sequence bits
7. Issuing a host service request type 0b11
8. Enabling service by assigning H, M or L priority to the clock channel via the two channel priority bits

The TPU then starts the data transfer, and issues an interrupt request when the transfer is complete.

Once the function has been initialized, the CPU only needs to write SIOP_DATA with the new data and issue a HSR 0b11 to initiate a new transfer. In input-only or clock-only modes, just the HSR 0b11 is required.

D.19.3 SIOP Function Performance

Like all TPU functions, the performance limit of the SIOP function in a given application is dependent to some extent on the service time (latency) associated with other active TPU channels. This is due to the operational nature of the scheduler. Where two channels are being used for a uni-directional system, and no other TPU channels are active, the maximum baud rate is approximately 230 at a bus speed of 16.77 MHz. A three-channel bidirectional system under the same conditions has a maximum baud rate of approximately 200. When more TPU channels are active, these performance figures will be degraded, however, the scheduler assures that the worst case latency in any TPU application can be closely approximated. It is recommended that the guidelines given in the TPU reference manual be used along with the information given in the SIOP state timing table to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

Table D-5 SIOP State Timing[1]

State Number and Name	Max. CPU Clock Cycles	Number of RAM Accesses by TPU
S1 SIOP_INIT		
HSQ = X0	28	7
X1	38	7
S2 DATA_OUT		
HSQ = X0	14	4
X1	24	4
S3 DATA_IN		
HSQ = 0X	14	4
1X	28	6

NOTES:
1. Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks).

D.19.3.1 XFER_SIZE Greater Than 16

XFER_SIZE is normally programmed to be in the 1- to 16-bit range to match the size of SIOP_DATA, and has thus been shown as a 5-bit value in the host interface diagram. However, the TPU actually uses all 16 bits of the XFER_SIZE parameter when loading BIT_COUNT. In some unusual circumstances this can be used to an advantage. If an input device is producing a data stream of greater than 16 bits then manipulation of XFER_SIZE will allow selective capturing of the data. In clock-only mode, the extended XFER_SIZE can be used to generate up to 0xFFFF clocks.

D.19.3.2 Data Positioning

As stated above, no ‘justifying’ of the data position in SIOP_DATA is performed by the TPU. This means that in the case of a byte transfer, the data output will be sourced from one byte and the data input will shift into the other byte. This rule holds for all data size options except 16 bits when the full SIOP_DATA register is used for both data output and input.

D.19.3.3 Data Timing

In the example given in **Figure D-31**, the data output transitions are shown as being completely synchronous with the relevant clock edge and it is assumed that the data input is latched exactly on the opposite clock edge. This is the simplest way to show the examples, but is not strictly true. Since the TPU is a multi-tasking system, and the data channels are manipulated directly by microcode software while servicing the clock edge, there is a finite delay between the relevant clock edge and the data-out being valid or the data-in being latched. This delay is equivalent to the latency in servicing the clock channel due to other TPU activity and is shown as ‘Td’ in the timing diagram. Td is the delay between the clock edge and the next output data being valid and also the delay between the opposite clock edge and the input data being read. For the vast majority of applications, the delay Td will not present a problem and can be ignored. Only for a system which heavily loads the TPU should the worst case latency be calculated for the SIOP clock channel + actual SIOP service time (= Td) and ensure that the baud rate is chosen such that HALF_PERIOD - Td is not less that the minimum

setup time of the receiving device. A transmitting device must also hold data valid for a minimum time of Td after the clock.

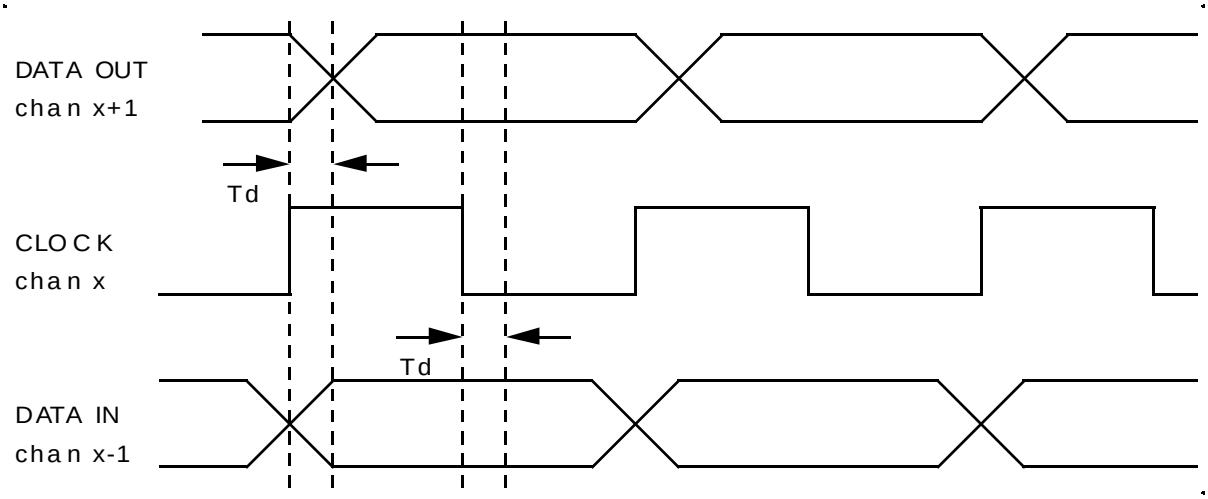


Figure D-31 SIOP Function Data Transition Example

APPENDIX E
ELECTRICAL CHARACTERISTICS

This section contains detailed information on power considerations, DC/AC electrical characteristics, and AC timing characteristics of the MPC565 / MPC566.

The MPC565 / MPC566 is designed to operate at 40 MHz, or optionally up to 56 MHz. The electrical characteristics are PRELIMINARY and are from previous designs, design simulations, or initial evaluation. These characteristics may not be fully tested or guaranteed at this early stage of the product life cycle, however, for production silicon these characteristics will be met. Finalized characteristics will be published after complete characterization and device qualifications have been completed.

E.1 Absolute Maximum Ratings (VSS = 0V)

Table E-1 Absolute Maximum Ratings

	Rating	Symbol	Min. Value	Max. Value	Unit
1	2.6-V Supply Voltage[1]	V_{DDL}	-0.3	3.0	V
2	Flash Supply Voltages[2]	V_{FLASH}	-0.3	5.6	V
3	Flash Core Voltage[1]	V_{DDF}	-0.3	3.0	V
4	Oscillator, keep-alive Reg. Supply Voltage[1]	KAPWR	-0.3	3.0	V
5	SRAM Supply Voltage[1]	V_{DDSRAM}[1, 2, 3]	-0.3	3.0	V
6	Clock Synthesizer Supply Voltage[1]	V_{DDSYN}	-0.3	3.0	V
7	32-KHz RTC Oscillator Supply Voltage	V_{DDRTC}	-0.3	3.0	V
8	QADC Supply Voltage[3]	V_{DDA}	-0.3	5.6	V
9	5-V Supply Voltage	V_{DDH}	-0.3	5.6	V
10	DC Input Voltages[4,5]	V_{IN}	V_{SS}-0.3	5.6	V
11	Reference V_{RH}, with reference to V_{RL}	V_{RH}	-0.3	5.6	V
12	Reference ALTREF, with reference to V_{RL}	V_{ARH}	-0.3	5.6	V
13	V_{SS} Differential Voltage	$V_{SS} – V_{SSA}$	-0.1	0.1	V
15	V_{REF} Differential Voltage	$V_{RH} – V_{RL}$	-5.6	5.6	V
16	V_{RL} to V_{SSA} Differential Voltage	$V_{RL} – V_{SSA}$	-5.6	0.3	V
17	Maximum Input Current per pin [6, 7, 8]	I_{MA}	- 25	25	µA
18	QADC Maximum Input Current per Pin	I_{MAX}	-25	25	µA
19	Operating Temperature Range – Ambient (Packaged)	T_A	-40 (T_L)	+125 (T_H)	°C
20	Operating Temperature Range – Solder Ball (Packaged any perimeter solder ball)[9]	T_{SB}	-40 (T_L)	+135 (T_H)	°C
21	Operating Temperature Range (Die Form)	T_J	-40	+150	°C
22	Storage Temperature Range	T_{STG}	-55	+150	°C
23	Maximum Solder Temperature[10]	T_{SDR}	—	220	°C
24	Moisture Sensitivity Level[11]	MSL	—	3	—

NOTES:

1. For internal digital supply of V_{DDL} = 2.6-V typical.
2. During operation the value of V_{FLASH} must be 5.0 V ± 5%.
3. V_{DDA}=5.0 V ±5%.
4. All 2.6-V input-only pins are 5-V tolerant.
5. Note that long term reliability may be compromised if 2.6-V output drivers drive a node which has been previously pulled to >3.45 V by an external component.
6. Maximum continuous current on I/O pins provided the overall power dissipation is below the power dissipation of the package. Proper operation is not guaranteed at this condition.
7. Condition applies to one pin at a time.
8. Transitions within the limit do not affect device reliability or cause permanent damage. Exceeding limit may cause permanent conversion error on stressed channels and on unstressed channels.
9. Maximum operating temperature on any solder ball in outer four rows of solder balls on the package. These rows are referred to as “Perimeter Balls” to distinguish them from the thermal balls in the center of the package.
10. Solder profile per CDF-AEC-Q100, current revision.
11. Moisture sensitivity per JEDEC test method A112.

Functional operating conditions are given in **E.6 DC Electrical Characteristics**. Ab-

solute maximum ratings are stress ratings only, and functional operation at the maxima is not guaranteed. Stress beyond those listed may affect device reliability or cause permanent damage to the device.

This device contains circuitry protecting against damage due to high static voltage or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

E.2 Package

The MPC565 / MPC566 is available in two forms, packaged and bumped die. The package is a 388-ball PBGA having a 1.0 mm ball pitch, Motorola case outline 1164-01 (See **Figure 2-5**).

E.3 EMI Characteristics

E.3.1 Reference Documents

The documents referenced for the EMC testing of MPC565 / MPC566 are listed below.

1. The SAE J1752/3 Issued 1995-03
2. The ***MPC555/MPC556UM/AD Users Manual***

E.3.2 Definitions and Acronyms

EMC – Electromagnetic Compatibility

EMI – Electromagnetic Interference

TEM Cell – Transverse Electromagnetic Mode cell

E.3.3 EMI Testing Specifications

1. Scan range: 150 KHz – 1000 MHz
2. Operating Frequency: 56 MHz
3. Operating Voltages: 2.6 V, 5.0 V
4. Max spikes: TBD dBuV
5. I/O port waveforms: Per J1752/3
6. Temperature: 25 °C

E.4 Thermal Characteristics

Table E-2 Thermal Characteristics

Characteristic	Symbol	Value	Unit
BGA Package Thermal Resistance, Junction to Ambient – Natural Convection	$R_{\theta JA}$	TBD[1,2]	°C/W
BGA Package Thermal Resistance, Junction to Ambient – Four layer (2s2p) board, natural convection	$R_{\theta MA}$	TBD[3,4]	°C/W
BGA Package Thermal Resistance, Junction to Board	$R_{\theta JB}$	TBD[5]	°C/W
BGA Package Thermal Resistance, Junction to Case (top)	$R_{\theta JC}$	TBD[6]	°C/W
BGA Package Thermal Resistance, Junction to Package Top, Natural Convection	Ψ_{JC}	TBD[7]	°C/W

NOTES:

1. Junction temperature is a function of on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and the board thermal resistance.
2. Per SEMI G38-87 and JESD51-2 with the board horizontal.
3. Junction temperature is a function of on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and the board thermal resistance.
4. Per JESD51-6 with the board horizontal.
5. Thermal resistance between the die and the printed circuit board (Four layer (2s2p) board, natural convection).
6. Indicates the thermal resistance between the die and the case top surface as measured by the cold plate method (MIL SPEC-883 Method 1012.1) with the cold plate temperature used for the case temperature.
7. Thermal characterization parameter indicating the temperature difference between package top and the junction temperature per EIA/JESD51-2.

An estimation of the chip junction temperature, T_J, in °C can be obtained from the equation:

$$T_J = T_A + (R_{\theta JA} \times P_D)$$

where:

T_A = ambient temperature (°C)

$R_{\theta JA}$ = package junction to ambient resistance (°C/W)

P_D = power dissipation in package

The junction to ambient thermal resistance is an industry standard value which provides a quick and easy estimation of thermal performance. Unfortunately, the answer is only an estimate; test cases have demonstrated that errors of a factor of two are possible. As a result, more detailed thermal characterization is supplied.

Historically, the thermal resistance has frequently been expressed as the sum of a junction to case thermal resistance and a case to ambient thermal resistance:

$$R_{\theta JA} = R_{\theta JC} + R_{\theta CA}$$

where:

$R_{\theta JA}$ = junction to ambient thermal resistance (°C/W)

$R_{\theta JC}$ = junction to case thermal resistance (°C/W)

$R_{\theta JA}$ = case to ambient thermal resistance (°C/W)

$R_{\theta JC}$ is device related and cannot be influenced. The user controls the thermal environment to change the case to ambient thermal resistance, $R_{\theta CA}$. For instance, the air flow can be changed around the device, add a heat sink, change the mounting arrangement on printed circuit board, or change the thermal dissipation on the printed circuit board surrounding the device. This description is most useful for ceramic packages with heat sinks where about 90% of the heat flow is through the case to the heat sink to ambient. For most packages, a better model is required.

The simplest thermal model of a package which has demonstrated reasonable accuracy (about 20%) is a two resistor model consisting of a junction to board and a junction to case thermal resistance. The junction to case covers the situation where a heat sink will be used or where a substantial amount of heat is dissipated from the top of the package. The junction to board thermal resistance describes the thermal performance when most of the heat is conducted to the printed circuit board. It has been observed that the thermal performance of most plastic packages and especially PBGA packages is strongly dependent on the board. temperature.

If the board temperature is known, an estimate of the junction temperature in the environment can be made using the following equation:

$$T_J = T_B + (R_{\theta JB} \times P_D)$$

where:

T_B = board temperature (°C)

$R_{\theta JB}$ = package junction to board resistance (°C/W)

P_D = power dissipation in package (W)

If the board temperature is known and the heat loss from the package case to the air can be ignored, acceptable predictions of junction temperature can be made. For this method to work, the board and board mounting must be similar to the test board used to determine the junction to board thermal resistance, namely a 2s2p (board with a power and a ground plane) and vias attaching the thermal balls to the ground plane.

When the board temperature is not known, a thermal simulation of the application is needed. The simple two-resistor model can be used with the thermal simulation of the application (2), or a more accurate and complex model of the package can be used in the thermal simulation. Consultation on the creation of the complex model is available.

To determine the junction temperature of the device in the application after prototypes are available, the thermal characterization parameter (Ψ_{JT}) can be used to determine

the junction temperature with a measurement of the temperature at the top center of the package case using the following equation:

$$T_J = T_T + (\Psi_{JA} \times P_D)$$

where:

T_T = thermocouple temperature on top of package (°C)

$R_{\theta JA}$ = thermal characterization parameter

P_D = power dissipation in package

The thermal characterization parameter is measured per JESD51-2 specification published by JEDEC using a 40 gauge type-T thermocouple epoxied to the top center of the package case. The thermocouple should be positioned so that the thermocouple junction rests on the package. A small amount of epoxy is placed over the thermocouple junction and over about one mm of wire extending from the junction. The thermocouple wire is placed flat against the package case to avoid measurement errors caused by cooling effects of the thermocouple wire.

E.4.1 Thermal References

Semiconductor Equipment and Materials International
805 East Middlefield Road
Mountain View, CA 94043
(415) 964-5111

MIL-SPEC and EIA/JESD (JEDEC) specifications are available from Global Engineering Documents at 800-854-7179 or 303-397-7956.

JEDEC specifications are available on the WEB at ***http://www.jedec.org***

1. C.E. Triplett and B. Joiner, "*An Experimental Characterization of a 272 PBGA Within an Automotive Engine Controller Module*," Proceedings of SemiTherm, San Diego, 1998, pp. 47-54.
2. B. Joiner and V. Adams, "*Measurement and Simulation of Junction to Board Thermal Resistance and Its Application in Thermal Modeling*," Proceedings of SemiTherm, San Diego, 1999, pp. 212-220.

E.5 ESD Protection

Table E-3 ESD Protection

Characteristics	Symbol	Value	Units
ESD for Human Body Model (HBM)[1]		2000	V
HBM Circuit Description	R1	1500	W
	C	100	pF
ESD for Machine Model (MM)		200	V
MM Circuit Description	R1	0	W
	C	200	pF
Number of pulses per pin[2] Positive pulses (MM) Negative pulses (MM) Positive pulses (HBM) Negative pulses (HBM)	 — — — —	 3 3 1 1	—
Interval of Pulses	—	1	S

NOTES:

1. All ESD testing is in conformity with CDF-AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits.
2. A device will be defined as a failure if after exposure to ESD pulses the device no longer meets the device specification requirements. Complete DC parametric and functional testing shall be performed per applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

E.6 DC Electrical Characteristics

(V_{DDL} = 2.6 V ± 0.1 V, V_{DDH} = 5.0 V ± 0.25 V, $T_A = T_L$ to T_H, $T_B = T_L$ to T_H)

Table E-4 DC Electrical Characteristics

	Characteristic	Symbol	Min	Max	Unit
1	2.6-V only Input High Voltage [1] except DATA[0:31] and EXTCLK	VIH2.6	2.0	V_{DDH} + 0.3	V
1a	2.6-V Input High Voltage EXTCLK	VIHC	1.6	V_{DDH} + 0.3	V
2	DATA[0:31] Precharge Voltage [2]	VDATAPC		3.45	V
3	5-V Input only High Voltage [3]	VIH5	0.7 * V_{DDH}	V_{DDH} + 0.3	V
4	5-V Input High Voltage (QADC PQA, PQB)	VIHA5	0.7 * V_{DDH}	V_{DDA} + 0.3	V
5	MUXed 2.6-V/ 5-V pins (GPIO muxed with Addr and Data) 2.6-V Input High Voltage Addr., Data 5-V Input High Voltage (GPIO)	 VIH2.6M VIH5M	 2.0 0.7 * V_{DDH}	 V_{DDH} + 0.3 V_{DDH} + 0.3	 V V
6	2.6-V Input Low Voltage Except EXTCLK	VIL2.6	V_{SS} – 0.3	0.8	V
7	2.6-V Input Low Voltage EXTCLK	VIL2.6C	V_{SS} – 0.3	0.4	V

Table E-4 DC Electrical Characteristics (Continued)

	Characteristic	Symbol	Min	Max	Unit
8	5-V Input Low Voltage	VIL5	V_{SS} – 0.3	0.48 * V_{DDH}	V
9	5-V Input Low Voltage (QADC PQA, PQB)	VILA5	V_{SSA} – 0.3	0.48 * V_{DDH}	V
10	MUXed 2.6-V/ 5-V pins (GPIO muxed with Addr, Data) 2.6-V Input Low Voltage (Addr., Data) 5-V Input Low Voltage (GPIO)	 VIL2.6M VIL5M	 V_{SS} – 0.3 V_{SS} – 0.3	 0.8 0.48 * V_{DDH}	 V V
11	QADC Analog Input Voltage[4] Note: Assumes $V_{DDA} \geq V_{DDH}$	VINDC	V_{SSH} – 0.3	V_{DDH} + 0.3	V
12	2.6-V Weak Pull-up/down Current [5] pull-up @ VIL2.6, pull-down @ VIH2.6	$I_{ACT2.6V}$	20	130	µA
13	5-V Weak Pull-up/down Current[4] pull-up @ VIL5, pull-down @ VIH5	I_{ACT5V}	20	130	µA
14	2.6-V Input Leakage Current[4] pull-up/down inactive – measured @rails	$I_{INACT2.6V}$	—	2.5	µA
15	5V Input Leakage Current[4] pull-up/down inactive – measured @rails	$I_{INACT5V}$	—	2.5	µA
16	QADC64 Input Current, Channel Off [6] PQA, PQB	I_{OFF}	-200 -200	200 200	nA nA
17	2.6-V Output High Voltage V_{DD} = V_{DDL} (IOH = -2mA)	VOH2.6	2.3	—	V
18	5-V Output High Voltage V_{DD} = V_{DDH} (IOH= -2mA) All 5-V only outputs except TPU.	VOH5	V_{DDH} – 0.7	—	V
19	5-V Output High Voltage V_{DD} = V_{DDH} (IOH= -5mA) For TPU pins Only	VOHTP5	V_{DDH} – 0.65	—	V
20	MUXed 2.6-V/ 5-V pins (GPIO MUXed with Addr, Data) 2.6-V Output High Voltage (IOH = -2mA) 5-V Output High Voltage (IOH = -2mA)	 VOH2.6M VOH5M	 2.3 V_{DDH} – 0.7	—	 V V
21	2.6-V Output Low voltage V_{DD} = V_{DDL} (IOL = 3.2mA)	VOL2.6	—	0.5	V
22	5-V Output Low voltage V_{DD} = V_{DDH} (IOL = 2mA) All 5-V only outputs except TPU	VOL5	—	0.45	V
23	5-V Output Low voltage V_{DD} = V_{DDH} -TPU pins Only IOL = 2mA IOL = 10mA	VOLTP5	—	 0.45 1.0	V
24	MUXed 2.6-V/ 5-V pins (GPIO MUXed with Addr, Data) 2.6-V Output Low Voltage (IOL = 3.2mA) 5-V Output Low Voltage (IOL = 2mA)	 VOL2.6M VOL5M		 0.5 0.45	 V V
25	Output Low Current (@ VOL= 0.4 V)	IOL	2.0	—	mA
26	Output High Current (@ VOH= 2.3 V)	IOH	2.0	—	mA
27	CLKOUT Load Capacitance – SCCR COM & CQDS COM[0:1]= 0b01, CQDS = 0b1 COM[0:1]= 0b01 CQDS = 0b0 COM[0:1]= 0b00 CQDS = 0bx	C_{CLK}	—	 25 50 100	 pF pF pF
29	Capacitance for Input, Output, and Bidirectional Pins: Vin = 0 V, f = 1 MHz (except QADC)	Cin	—	7	pF
30	Load Capacitance for bus pins only [7] COM[0:1] of SCCR = 0b11 COM[0:1] of SCCR = 0b10	CL	—	 25 45	pF

Table E-4 DC Electrical Characteristics (Continued)

	Characteristic	Symbol	Min	Max	Unit
31	Total Input Capacitance PQA Not Sampling PQB Not Sampling	 C_{IN}	 — —	 15 15	pF
32	Hysteresis (Only IRQ, TPU, MIOS, GPIO, QADC (Digital inputs) and RESET) [8]	VH	0.5	—	V
33	Operating Current (2.6-V supplies) @ 40 MHz[9] V_{DDL}/V_{DDI} KAPWR (Crystal Frequency: 20 MHz) $V_{DDSRAM1}$ ($V_{DDSRAM1}$ + $V_{DDSRAM2}$) $V_{DDSRAM3}$ V_{DDSYN} V_{DDRTC} (Crystal Frequency: 32 KHz) V_{DDF} (Read, program, or erase) V_{DDF} (two-module program or erase) $V_{DDFSTOP}$ $V_{DDFDISABLED}$	 I_{DDL} I_{DDKAP} $I_{DDSRAM1}$ $I_{DDSRAM3}$ I_{DDSYN} I_{DDRTC} I_{DDF} $I_{DDFPROG2}$ $I_{DDFSTOP}$ $I_{DDFDISB}$		 200 8 10 30 8 10 60 120 10 100	mA
34	Operating Current (5-V supplies)@ 40 MHz[10] V_{DDH} V_{DDA}[11] $V_{FLASHF5}$ (Program or Erase) $V_{FLASHF5READ}$ $V_{FLASHF5}$ (Stopped) $V_{FLASHF5}$ (Disabled)	 I_{DDH5} I_{DDA} I_{DDF5} I_{DDF5R} SI_{DDF5} SI_{DDF5D}		 21 5 75 7 1 100	mA
35	Operating Current (2.6-V supplies)@ 56 MHz[12] V_{DDL}/V_{DDI} KAPWR $V_{DDSRAM1}$ ($V_{DDSRAM1}$ + $V_{DDSRAM2}$) $V_{DDSRAM3}$ V_{DDSYN} (Crystal Frequency: 20 MHz) V_{DDRTC} (Crystal Frequency: 32 KHz) V_{DDF} (Read, program, or erase) V_{DDF} (2 module program or erase) $V_{DDF\ STOP}$ $V_{DDF\ DISABLED}$	 I_{DDL} I_{DDKAP} $I_{DDSRAM1}$ $I_{DDSRAM3}$ I_{DDSYN} I_{DDRTC} I_{DDF} $I_{DDFPROG2}$ $I_{DDFSTOP}$ $I_{DDFDISB}$		 285 8 10 30 10 10 60 120 10 100	mA
36	Operating Current (5-V supplies)@ 56 MHz[13] V_{DDH} V_{DDA}[14] $V_{FLASHF5}$ (Program or Erase) $V_{FLASHF5READ}$ $V_{FLASHF5}$ (Stopped) $V_{FLASHF5}$ (Disabled)	 I_{DDH5} I_{DDA} I_{DDF5} I_{DDF5R} SI_{DDF5} SI_{DDF5D}		 25 5.0 75 10 1 100	 mA mA mA mA mA mA
37	QADC64 Low Power Stop Mode (V_{DDA})	I_{DDA}		10	µA
38	Low Power Current (V_{DDL}+V_{DDI}+V_{DDF}) @56 MHz DOZE, Active PLL and Active Clocks SLEEP, Active PLL with Clocks off DEEP SLEEP, PLL and Clocks off	 I_{DDDZ} I_{DDSLP} $I_{DDDPSLP}$		 TBD[15] TBD[15] TBD[15]	 mA mA µA

Table E-4 DC Electrical Characteristics (Continued)

	Characteristic	Symbol	Min	Max	Unit
39	V_{DDL}, V_{DDI}, $V_{DDF,}$ Operating Voltage	V_{DDL}, $V_{DDI,}$ $V_{DDF,}$	2.5	2.7	V
40	V_{FLASH} Flash Operating/Programming Voltage	V_{FLASH}	4.75	5.25	V
41	Oscillator, Keep-Alive Registers Operating Voltage[16,17]	KAPWR	V_{DDL}-0.2 V	V_{DDL}+0.2 V	V
42	SRAM Operating Voltage[18]	V_{DDSRAM}	V_{DDL}-0.2 V	V_{DDL}+ 0.2 V	V
43	V_{DDH} Operating Voltage	V_{DDH}	4.75	5.25	V
44	QADC Operating Voltage	V_{DDA}	4.75	5.25	V
45	Clock Synthesizer Operating Voltage[17]	V_{DDSYN}	V_{DDL} – 0.2 V	V_{DDL} + 0.2 V	V
46	Real Time Clock Operating Voltage[17]	V_{DDRTC}	V_{DDL} – 0.2 V	V_{DDL}+ 0.2 V	V
47	V_{SS} Differential Voltage	V_{SS} – V_{SSA}	-100	100	mV
48	QADC64 Reference Voltage Low[19]	V_{RL}	V_{SSA}	V_{SSA} + 0.1	V
49	QADC64 Reference Voltage High[19]	V_{RH}	3.0	V_{DDA}	V
50	QADC64 V_{REF} Differential Voltage	V_{RH} – V_{RL}	3.0	5.25	V
51	QADC64 Reference Supply Current, DC QADC64 Reference Supply Current, Transient	I_{REF} I_{REF}	— —	500 4.0	µA mA
52	QADC64 ALT Reference Voltage	V_{ARH}	.25 * V_{DDA}	.75 * V_{DDA}	V
53	Standby Supply Current[4] KAPWR only V_{DDRTC} RAM Standby Current (CALRAM) $V_{DDSRAM1}$ RAM Standby Current (CALRAM) $V_{DDSRAM2}$ RAM Standby Current (DPTRAM)$V_{DDSRAM3}$ Measured @ 2.7 V	ISB_{KAPWR} ISB_{RTC} ISB_{SRAM1} ISB_{SRAM2} ISB_{SRAM3}		5 50 250 100 100	mA µA µA µA µA
53a	RAM Standby Voltage for Data Retention Applies to $V_{DDSRAM1}$ $V_{DDSRAM2}$ and $V_{DDSRAM3}$ (power-down Mode)[4] Specified V_{DD} applied (V_{DD} = V_{SS})	V_{STBY}	1.4[20, 21]	2.7	V
54	DC Injection Current per Pin GPIO, TPU, MIOS, QSM, EPEE and 5 V [4, 22]	I_{IC5}	-1.0	1.0	mA
55	DC Injection Current per Pin 2.6 V [4, 23, 24]	I_{IC3}	-1.0	1.0	mA
56	QADC64 Disruptive Input Current [25]	I_{NA}	- 3	3	mA
57	Power Dissipation – 56 MHz 40 MHz	PD		1.12 0.8	W W

NOTES:

1. This characteristic is for 2.6-V output and 5-V input friendly pins.
2. VDATAPC is the max voltage the data pins can have been precharged to by an external device when the MPC565 / MPC566 data pins turn on as outputs. The 3.45-V max for VDATAPC is to allow the data pins to be driven from an external memory running at a higher voltage. Note that if the data pins are precharged to higher than V_{DDL}, then the 50-pF max. load characteristic must be observed.
3. This characteristic is for 5-V output and 5-V input pins.
4. Within this range, no significant injection will be seen. See QADC64 Disruptive Input Current (I_{NA}).
5. After characterization this value may be improved.
6. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each eight to 12 °C, in the ambient temperature range of 50 to 125 °C.
7. All bus pins support two drive strengths capabilities, 25 pF and 50 pF. Current drive is less at the 25-pF capacitive load. Both modes achieve 40-MHz (or, optionally, 56-MHz) timing.

8. Only IRQ, TPU, MIOS, GPIO, QADC (when digital inputs) and RESET pins have hysteresis, thus there is no hysteresis specification required for all other pins
9. Values to be characterized. Current consumption values will be updated as information becomes available. Initial values are only estimates based on predicted capacitive differences between CDR1 and CDR3 as well as actual CDR1 measurements.
10. Values to be characterized. Current measured at maximum system clock frequency with QADC active.
11. Current measured at maximum system clock frequency with QADC active.
12. All power consumption specifications assume 50-pF loads and running a typical application. Of the power consumption of some modules could go up is they are exercised heavier, but the power consumption of other modules would decrease.
13. Values to be characterized. Current measured at maximum system clock frequency with QADC active.
14. Current measured at maximum system clock frequency with QADC active.
15. Pending characterization.
16. KAPWR and V_{DDSRAM} are powered-up prior to any other supply.
17. This parameter is periodically sampled rather than 100% tested
18. KAPWR and V_{DDSRAM} are powered-up prior to any other supply.
19. To obtain full-range results, $V_{SSA} \leq V_{RL} \leq V_{INDC} \leq V_{RH} \leq V_{DDA}$
20. The voltage at which the LVSRS bits in the VSRAMCR register will be set ranges from TBD to TBD volts. V_{STBY} should remain above TBD volts to prevent LVSRS bits to be set.
21. This parameter is guaranteed by design.
22. All injection current is transferred to the V_{DDH}. An external load is required to dissipate this current to maintain the power supply within the specified voltage range.
23. Total injection current for all I/O pins on the chip must not exceed 20 µA. Exceeding this limit can cause disruption of normal operation.
24. Current refers to two QADC64 modules operating simultaneously.
25. Below disruptive current conditions, the channel being stressed has conversion values of 0x3FF for analog inputs greater than V_{RH} and 0x000 for values less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.

E.7 Oscillator and PLL Electrical Characteristics

Table E-5 Oscillator and PLL

PIN	Characteristic	Symbol	Min	Typical	Max	Unit
1	Oscillator Startup time (for typical crystal capacitive load) 4-MHz crystal 20-MHz crystal	 OSCstart4 OSCstart20			 10 10	 mS mS
2	PLL Lock Time	T_{LOCK}			500[1]	Input Clocks
3	PLL Operating Range	F_{VCOOUT}	30		112	MHz
4	Crystal Operating Range, MODCK=0b010,0b110 MODCK[1:3] = 0b001, 0b011, 0b100, 0b101, 0b111)	$F_{CRYSTAL}$	2 15		5 25	MHz MHz
5	PLL Jitter PLL Jitter (averaged over 10 µs)	F_{JIT} F_{JIT10}	-1% -0.3%		+1% +0.3%	—
6	Limp Mode Clock Out Frequency	—	**3**[2]	7	**12**[1]	MHz
7	Oscillator Bias Current (XTAL) 4 MHz 20 MHz	I_{BIAS}	**TBD**[3] **TBD**[2]		— —	 mA mA
8	Oscillator Drive (XTAL)	Iosc	**TBD**[2]		—	mA
9	Oscillator Bias Resistor	R_{OSC}	**TBD**[2]	1	**TBD**[2]	MΩ

NOTES:
1. Assumes stable power and oscillator.
2. Estimated value, real values to be characterized and updated.
3. Values to be characterized.

E.8 Flash Electrical Characteristic

Table E-6 Array Program and Erase Characteristics

(V_{DDF} = 2.6 V ± 0.1 V, V_{FLASH} = 5.0 V ± 0.25 V, $T_A = T_L$ to T_H, $T_B = T_L$ to T_H)

Symbol	Meaning	Value		
		Minimum	Typical[1]	Maximum
T_{ERASE}	Block or Module Erase Time[2]		0.7 s	15 s
T_{PROG}	Word Programming Time[3]		15 µs	300 µs

NOTES:
1. Typical program and erase times assume nominal supply values and 25 °C.
2. Erase time specification does not include pre-programming operation
3. Word size is.32 bits.

Table E-7 CENSOR Cell Program and Erase Characteristics

(V_{DDF} = 2.6 V ± 0.1 V, V_{FLASH} = 5.0 V ± 0.25 V, $T_A = T_L$ to T_H, $T_B = T_L$ to T_H)

Symbol	Meaning	Value		
		Minimum	Typical[1]	Maximum
T_{CLEAR}	CENSOR Bit Clear Time[2]		TBD	
T_{SET}	CENSOR bit Set Time		TBD	

NOTES:
1. Typical set and clear times assume nominal supply values and 25C.
2. Clear time specification does not include pre-set operation.

Table E-8 Module Life

Symbol	Meaning	Value
Array P/E Cycles[1]	Maximum number of Program/Erase cycles per block to guarantee data retention.	1,000
CENSOR Set/Clear Cycles[2]	Minimum number of Program/Erase cycles per bit before failure.	100
Array and CENSOR Data Retention	Minimum data retention at 85 °C.	Min 10 years

NOTES:
1. A Program/Erase cycle is defined as switching the bits from 1 to 0 to 1.
2. A CENSOR Set/Clear cycle is defined as switching the bits from 1 to 0 to 1.

E.9 Power-Up/Down Sequencing

This section applies to only CDR3 devices such as MPC565 / MPC566.

The supply symbols used in this section are described in **Table E-9**.

Table E-9 MPC565 / MPC566 Power Supply Pin Groups

Symbol	Types of Power Pins
V_{DDHI} (High Voltage Supply Group)	Supply to the 5-V pads for output driver (V_{DDH})
	Supply to the analog (QADC64E) circuitry (V_{DDA})
	High voltage supply to the flash module (V_{FLASH})
V_{DDLO} (Low Voltage Supply Pins)	Supply to low voltage pad drivers (QVDDL, NVDDL)
	Supply to all low voltage internal logic (V_{DD})
	Supply to low voltage flash circuitry (V_{DDF})
	Supply to system PLL and other circuitry required for keep-alive functions (KAPWR, V_{DDSYN})
V_{DDKA} (Low Voltage Keep-Alive Supply Pins[1]	Supply to SRAM arrays only ($V_{DDSRAM1}$, $V_{DDSRAM2}$, $V_{DDSRAM3}$)
	Supply to low power counter and 32 KHz crystal (V_{DDRTC})

NOTES:
1. Any supply in the V_{DDKA} group can be powered with the V_{DDLO} if the function which it supplies is not required during “Keep-alive”.

There are two power-up/down options. Choosing which one is required for an application will depend upon circuitry connected to 2.6-V compliant pins and dual 2.6-V/5-V compliant pins. power-up/down option A is required if 2.6-V compliant pins and dual 2.6-V/5-V compliant pins are connected to the 5-V supply with a pull-up resistor or driven by 5-V logic during power-up/down. In applications for which this scenario is not true the power-up/down option B may be implemented. Option B is less stringent and easier to ensure over a variety of applications.

Refer to **Table 2-1** for a list of 2.6 V and dual 2.6V/5 V compliant pins.

Power consumption during power-up/down sequencing can not be specified prior to evaluation and characterization of production silicon. It is Motorola’s intention to keep the power consumption during power-up/down sequencing below the operating power consumption when following these guidelines.

E.9.1 Power-Up/Down Option A

The Option A power-up sequence (excluding V_{DDKA}) is

1. $V_{DDHI} \leftarrow V_{DDLO} + 3.1$ V (V_{DDHI} cannot lead V_{DDLO} by more than 3.1 V)
2. $V_{DDHI} > V_{DDLO} - 0.5$ V (V_{DDHI} cannot lag V_{DDLO} by more than 0.5 V)

Requirement ‘1’ is due to gate-to-drain stress limits for transistors in the pads of 2.6-V compliant pins and dual 2.6-V/5-V compliant pins. Damage can occur if gate-to-drain voltage potential is greater than 3.1 V. This is only a concern at power-up/down. Requirement ‘2’ is due to ESD diodes in the pad logic for dual 2.6-V/5-V compliant pins. The diodes are forward biased when V_{DDLO} is greater than V_{DDHI} and will start to conduct current.

Figure E-1 illustrates the power-up sequence if no keep-alive supply is required. **Figure E-2** illustrates the power-up sequence if a keep-alive supply is required. The keep-

alive supply should be powered-up at the same instant or before both the high voltage and low voltage supplies are powered-up.

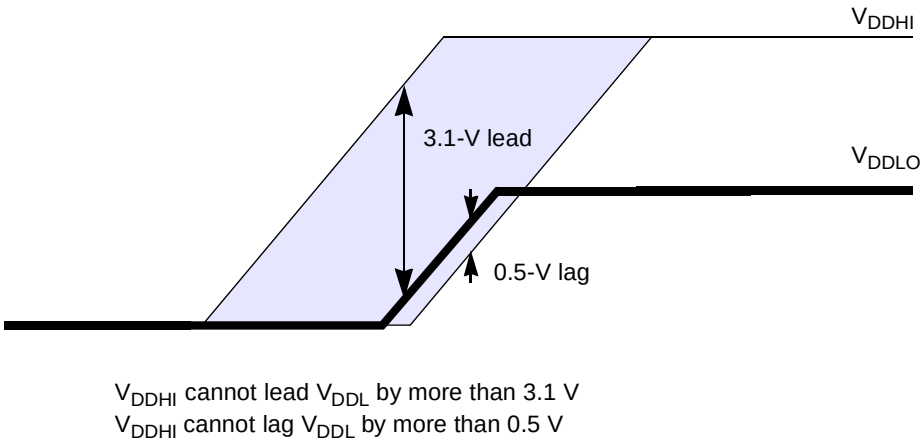


Figure E-1 Option A Power-Up Sequence Without Keep-Alive Supply

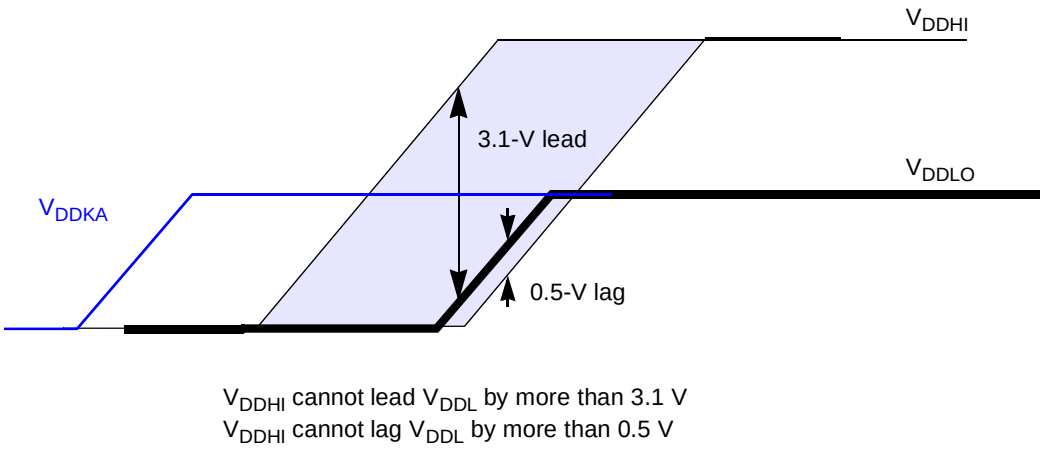


Figure E-2 Option A Power-Up Sequence With Keep-Alive Supply

The option A power-down sequence (excluding V_{DDKA}) is

1. $V_{DDHI} \leftarrow V_{DDLO}$ + 3.1 V (V_{DDHI} cannot lag V_{DDL} by more than 3.1 V)
2. V_{DDLO} crossover should occur at 0.5 V or less

Figure E-3 illustrates the power-down sequence if no keep-alive supply is required. **Figure E-4** illustrates the power-down sequence if a keep-alive supply is required.

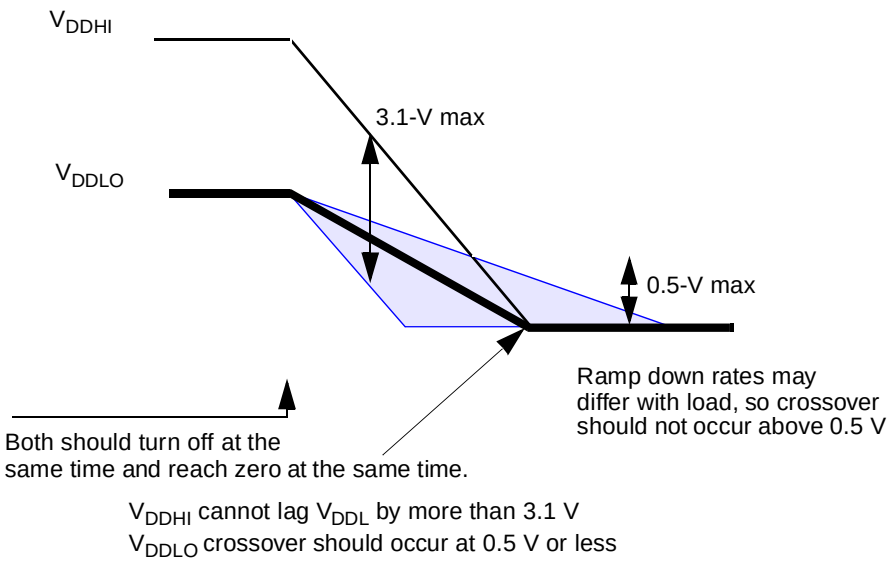


Figure E-3 Option A Power-Down Sequence Without Keep-Alive Supply

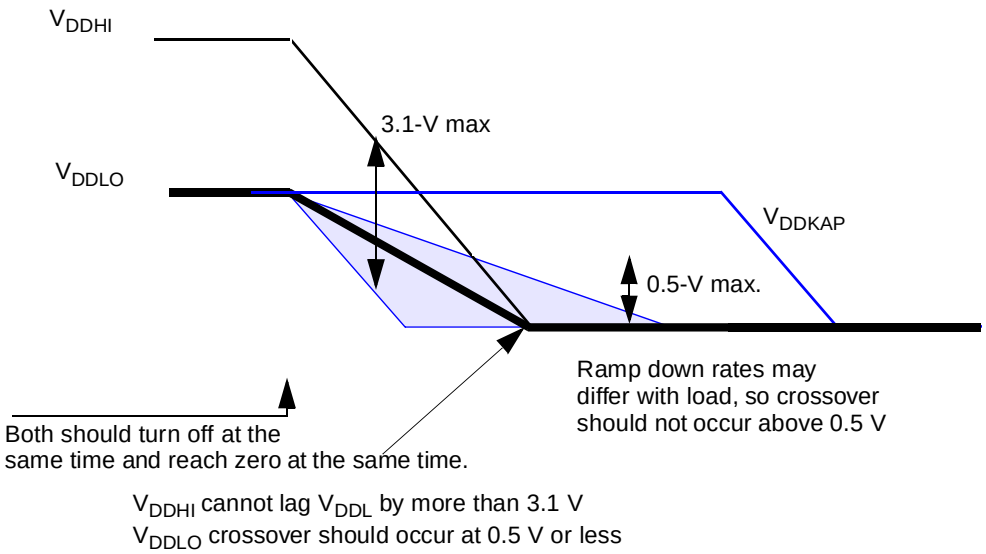


Figure E-4 Option A Power-Down Sequence With Keep-Alive Supply

E.9.2 Power-Up/Down Option B

A less stringent power-up sequence may be implemented if 2.6-V compliant pins and dual 2.6-V/5-V compliant pins are NOT connected to the 5-V supply with a pull-up resistor or driven by 5-V logic during power-up/down.

The option B power-up sequence (excluding V_{DDKA}) is:

$$V_{DDHI} > V_{DDLO} - 0.5\ V\ (V_{DDHI}\ \text{cannot lag}\ V_{DDLO}\ \text{by more than 0.5 V})$$

Thus the V_{DDHI} supply group can be fully powered-up prior to power-up of the V_{DDLO} supply group, with no adverse affects to the device.

The requirement that V_{DDHI} cannot lag V_{DDLO} by more than 0.5 V is due to ESD diodes in the pad logic for dual 2.6-V/5-V compliant pins. The diodes are forward biased when V_{DDLO} is greater than V_{DDHI} and will start to conduct current.

Figure E-5 illustrates the power-up sequence if no keep-alive supply is required. **Figure E-6** illustrates the power-up sequence if a keep-alive supply is required. The keep-alive supply should be powered-up at the same instant or before both the high voltage and low voltage supplies are powered-up.

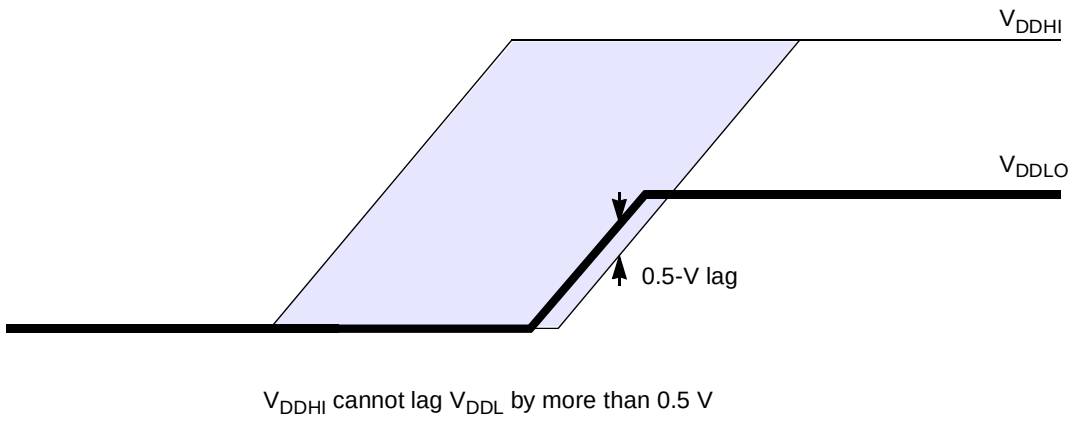


Figure E-5 Option B Power-Up Sequence Without Keep-Alive Supply

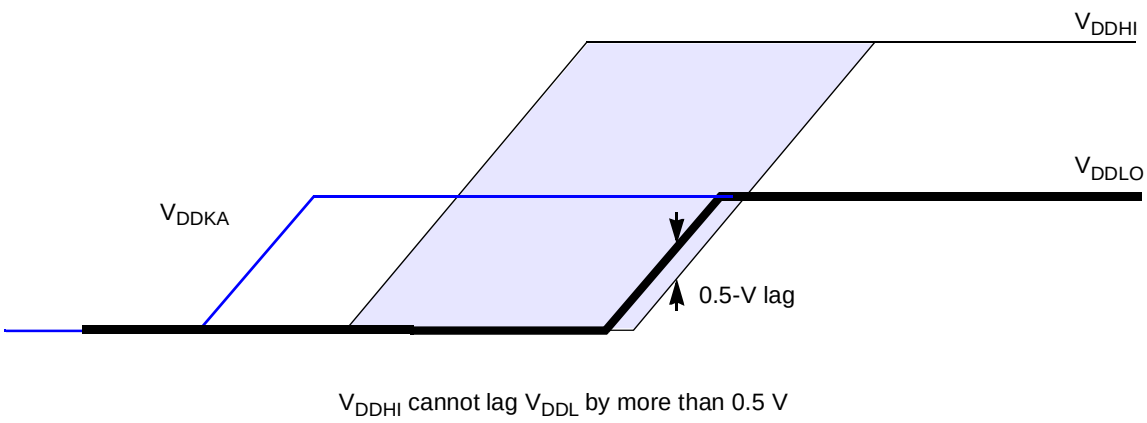


Figure E-6 Option B Power-Up Sequence With Keep-Alive Supply

The option B power-down sequence (excluding V_{DDKA}) is:

V_{DDLO} crossover should occur at 0.5 V or less

Thus the V_{DDLO} supply group can be fully powered-down prior to power-down of the V_{DDHI} supply group, with no adverse affects to the device.

For power-down, the low voltage supply should come down before the high voltage supply, although with varying loads, the high voltage may actually get ahead.

Figure E-7 illustrates the power-down sequence if no keep-alive supply is required. **Figure E-8** illustrates the power-down sequence if a keep-alive supply is required.

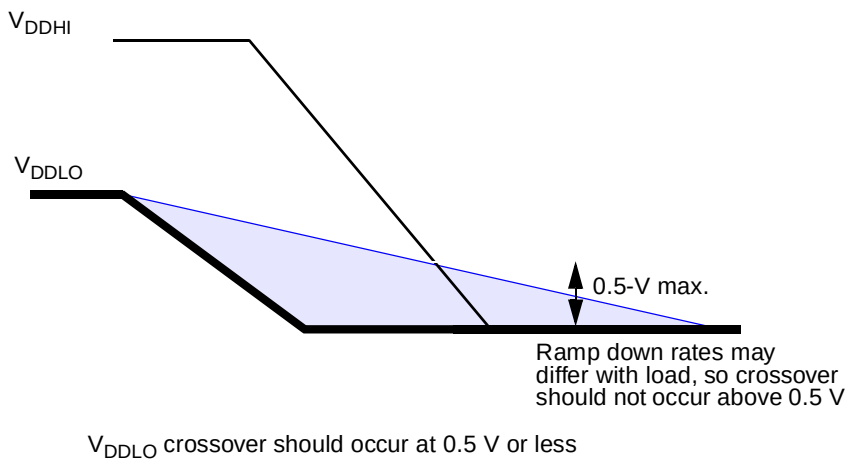


Figure E-7 Option B Power-Down Sequence Without Keep-Alive Supply

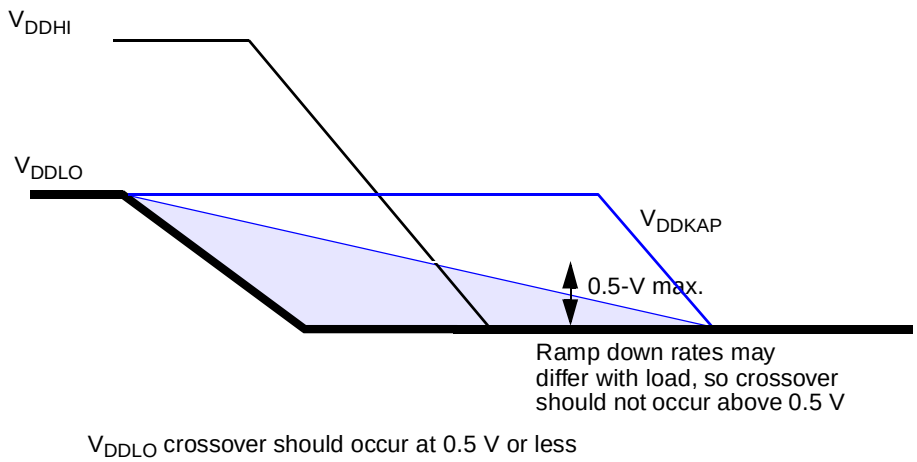


Figure E-8 Option B Power-Down Sequence with Keep-Alive Supply

E.10 Issues Regarding Power Sequence

E.10.1 Flash Issue

The $\overline{\text{PORESET}}$ signal must be asserted during power-up before the low voltage supply is above 0.5 V. If the $\overline{\text{PORESET}}$ signal is not asserted before this condition, there is a possibility of disturbing the programmed state of the flash.

E.10.2 Keep-Alive RAM

This issue should be addressed on all MPC5XX devices in both CDR1 and CDR3.

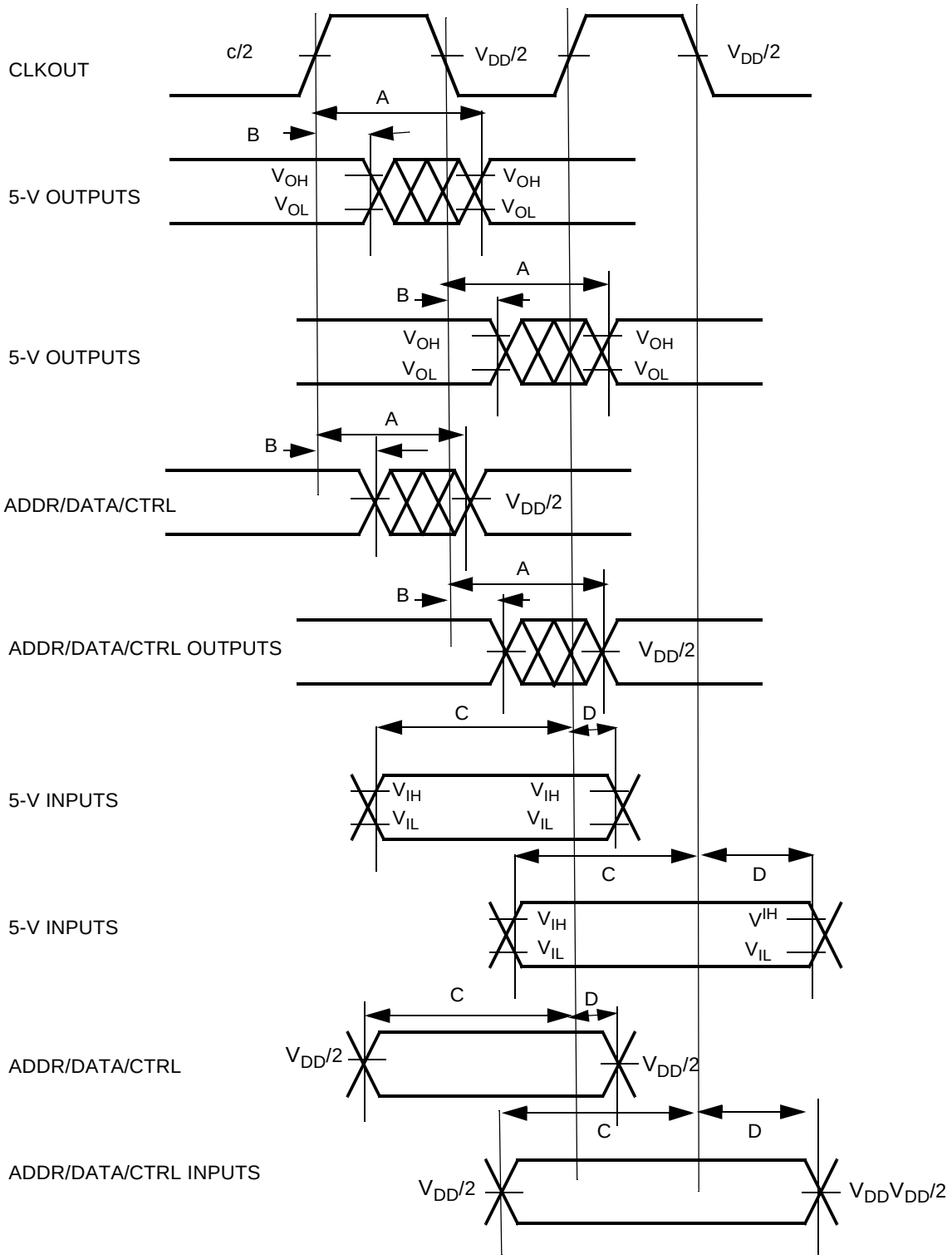
$\overline{\text{PORESET}}$ or $\overline{\text{HRESET}}$ must be asserted during power-down prior to any supply dropping out of specified operating conditions.

An additional constraint is placed on $\overline{\text{PORESET}}$ assertion since it is an asynchronous input. To assure that the assertion of $\overline{\text{PORESET}}$ does not potentially cause stores to keep-alive RAM to be corrupted (store single or store multiple) or non-coherent (store multiple), either of the following solutions is recommended:

1. Assert $\overline{\text{HRESET}}$ at least 0.5 µs prior when $\overline{\text{PORESET}}$ is asserted.
2. Assert $\overline{\text{IRQ0}}$ (non-maskable interrupt) at least 0.5 µs prior when $\overline{\text{PORESET}}$ is asserted. The service routine for $\overline{\text{IRQ0}}$ should not perform any writes to keep-alive RAM.

The amount of delay which should be added to $\overline{\text{PORESET}}$ assertion is dependent upon the frequency of operation and the maximum number of store multiples executed that are required to be coherent. If store multiples of more that 28 registers are needed and if the frequency of operation is lower that 56 MHz, the delay added to $\overline{\text{PORESET}}$ assertion will need to be greater than 0.5 µs.

E.11 Timing



A. Maximum Output Delay Specification
B. Minimum Output Hold Time
C. Minimum input Setup Time Specification
D. Minimum input Hold Time Specification

Figure E-9 Generic Timing Examples

Table E-10 Bus Operation Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
1	CLKOUT Period	TC	25		17.86		ns
1a	ENGCLK Frequency 5 V – EECLK = 01 2. 6 V – EECLK = 00			 10 40		 10 56	MHz
2	Clock pulse width low		12.5 –2%	12.5 + 2%	8.93 –2%	8.93 + 2%	ns
3	Clock pulse width high		12.5 – 2%	12.5 + 2%	8.93 – 2%	8.93 + 2%	ns
4	CLKOUT rise time ABUS/DBUS rise time			3.5 3.0		3.5 3.0	ns
5	CLKOUT fall time ABUS/DBUS rise time			3.5 3.0		3.5 3.0	ns
6	Circuit Parameter	TCC	7		5		ns
7	CLKOUT to Signal Invalid (Hold Time) A[0:31] RD/$\overline{\text{WR}}$ $\overline{\text{BURST}}$ D[0:31]	0.25TC – 1.25	5		5		ns
7a	CLKOUT to Signal Invalid: (Hold Time) TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] $\overline{\text{BDIP}}$ PTR $\overline{\text{RETRY}}$	0.25TC – 1.25	5		5		ns
7b	CLKOUT to Signal Invalid (Hold Time)[2] $\overline{\text{BR}}$ $\overline{\text{BG}}$ FRZ VFLS[0:1] VF[0:2] IWP(0:2] LWP[0:1] $\overline{\text{STS}}$[3]	0.25TC –- 1.25	5		5		ns
7c	Slave mode CLKOUT to Signal Invalid D[0:31]	**0.25TC+TCC**	7		6.5		ns
8	CLKOUT to Signal Valid A[0:31] RD/$\overline{\text{WR}}$ BURST D[0:31][4]	0.25TC + TCC	6.25	13	4.5	10	ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
8a	CLKOUT to Signal Valid TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] $\overline{\text{BDIP}}$ PTR $\overline{\text{RETRY}}$	0.25TC + TCC	6.25	13	4.5	9.5	ns
8b	CLKOUT to Signal Valid[2] $\overline{\text{BR}}$ $\overline{\text{BG}}$ VFLS[0:1] VF[0:2] IWP[0:2] FRZ LWP[0:1] $\overline{\text{STS}}$ valid.	0.25TC + TCC	6.25	13	4.5	9.5	ns
8c	Slave Mode CLKOUT to Signal Valid D[0:31]	0.25TC + TCC + 4		14		14	ns
9	CLKOUT to High Z A[0:31] RD/$\overline{\text{WR}}$ BURST D[0:31] TSIZ[0:1] $\overline{\text{RSV}}$ AT[0:3] PTR $\overline{\text{RETRY}}$	0.25TC + TCC	6.25	13	4.5	9.5	ns
10	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ assertion	0.25TC + TCC	6.25	13	4.5	9.5	ns
10a	CLKOUT to $\overline{\text{TA}}$, $\overline{\text{BI}}$ assertion (when driven by the Memory Controller)	—		10		10	ns
10b	CLKOUT to $\overline{\text{RETRY}}$ assertion (when driven by the Memory Controller)	—		10		10	ns
11	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ negation	0.25TC + TCC	6.25	13	4.5	9.5	ns
11a	CLKOUT to $\overline{\text{TA}}$, $\overline{\text{BI}}$ negation (when driven by the Memory Controller)	—		11		11	ns
11b	CLKOUT to $\overline{\text{RETRY}}$ negation (when driven by the Memory Controller)	—		11		11	ns
12	CLKOUT to $\overline{\text{TS}}$, $\overline{\text{BB}}$ High Z	0.25TC +14	6.25	20	4.5	16	ns
12a	CLKOUT to $\overline{\text{TA}}$, $\overline{\text{BI}}$ High Z (when driven by the Memory Controller)			15		15	ns
13	CLKOUT to $\overline{\text{TEA}}$ assertion	—		11		11	ns
14	CLKOUT to $\overline{\text{TEA}}$ High Z			15		15	ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
15	Input Valid to CLKOUT (Setup Time) $\overline{TA}$ $\overline{TEA}$ $\overline{BI}$[3]		12		8.5		ns
15a	Input Valid to CLKOUT (Setup Time) $\overline{KR}$ $\overline{CR}$ $\overline{RETRY}$		10		7.25		ns
15b	Input Valid to CLKOUT (Setup Time) BB $\overline{BG}$ $\overline{BR}$[2]		7		5		ns
16	CLKOUT to Signal Invalid (Hold Time) $\overline{TA}$ $\overline{TEA}$ $\overline{BI}$ BB $\overline{BG}$ $\overline{BR}$[2, 3]		2		2		ns
16a	CLKOUT to Signal Invalid (Hold Time) $\overline{RETRY}$ $\overline{KR}$ $\overline{CR}$		2		2		ns
17	Signal Valid to CLKOUT Rising Edge (Setup Time) D[0:31][4]		6		6		ns
18	CLKOUT Rising Edge to Signal Invalid (Hold Time) D[0:31][4]		1		.71		ns
19	CLKOUT Rising Edge to $\overline{CS}$ asserted -GPCM- ACS = 00	0.25TC + TCC + 1	6.25	14	5.5	10.5	ns
19a	CLKOUT Falling Edge to $\overline{CS}$ asserted -GPCM- ACS = 10, TRLX = 0 or 1	TCC + 1		8		6	ns
19b	CLKOUT Falling Edge to $\overline{CS}$ asserted -GPCM- ACS = 11, TRLX = 0 or 1	0.25TC + TCC + 1	6.25	14	5.5	10.5	ns
19c	CLKOUT Falling Edge to $\overline{CS}$ asserted -GPCM- ACS = 11, TRLX = 0, EBDF = 1	**0.375TC+TCC+1**	6.25	17	6.69	12.69	ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
20	CLKOUT Rising Edge to $\overline{CS}$ negated -GPCM- Read Access or Write access when CSNT = 0 or write access when CSNT = 1 and ACS = 00	TCC + 1	1	8	1	6	ns
21	A[0:31] to $\overline{CS}$ asserted -GPCM- ACS = 10, TRLX = 0		3		2.25		ns
21a	A[0:31] to $\overline{CS}$ asserted -GPCM- ACS = 11, TRLX = 0		8		5.71		ns
22	CLKOUT Rising Edge to $\overline{OE}$, $\overline{WE[0:3]}$ asserted			8		5.71	ns
23	CLKOUT Rising Edge to $\overline{OE}$ n egated		1	8	0.71	5.71	ns
24	A[0:31] to $\overline{CS}$ asserted -GPCM- ACS = 10, TRLX = 1		23		16.42		ns
24a	A[0:31] to $\overline{CS}$ asserted -GPCM- ACS = 11, TRLX = 1		28		20		ns
25	CLKOUT Rising Edge to $\overline{WE[0:3]}$ negated -GPCM-write access CSNT = ‘0‘			6		4.5	ns
25a	CLKOUT Falling Edge to $\overline{WE[0:3]}$ negated -GPCM-write access TRLX = ‘0’ or ‘1’, CSNT = ‘1, EBDF = 0’.	0.25TC + TCC + 1	6.25	14	5.5	10.5	ns
25b	CLKOUT Falling Edge to $\overline{CS}$ negated -GPCM-write access TRLX = ‘0’ or ‘1’, CSNT = ‘1’, ACS = ‘10’ or ACS=’11’, EBDF = 0	0.25TC + TCC + 1	6.25	14	5.5	10.5	ns
25c	CLKOUT Falling Edge to $\overline{WE[0:3]}$ negated -GPCM-write access TRLX = ‘0’, CSNT = ‘1, EBDF = 1’.	0.375TC + TCC + 1	6.25	17	5.5	12.69	ns
25d	CLKOUT Falling Edge to $\overline{CS}$ negated -GPCM-write access TRLX = ‘0’, CSNT = ‘1’, ACS = ‘10’ or ACS=’11’, EBDF = 1	0.375TC + TCC + 1	6.25	17	6.25	17	ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
26	$\overline{WE[0:3]}$ negated to D[0:31] High Z -GPCM- write access, CSNT = '0'		3		2.25		ns
26a	$\overline{WE[0:3]}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '0', CSNT = '1', EBDF = 0		8		5.71		ns
26b	$\overline{CS}$ negated to D[0:31], High Z -GPCM- write access, ACS = '00', TRLX = '0' & CSNT = '0'		3		2.25		ns
26c	$\overline{CS}$ negated to D[0:31], High Z -GPCM- write access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0		8		5.71		ns
26d	$\overline{WE[0:3]}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', EBDF = 0		28		20		ns
26e	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 0		28		20		ns
26f	$\overline{WE[0:3]}$ negated to D[0:31] HighZ -GPCM- write access, TRLX = '0', CSNT = '1', EBDF = 1		5		3.75		ns
26g	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '0', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1		5		3.75		ns
26h	$\overline{WE}$[0:3] negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', EBDF = 1		24		17.25		ns
26i	$\overline{CS}$ negated to D[0:31] High Z -GPCM- write access, TRLX = '1', CSNT = '1', ACS = '10' or ACS='11', EBDF = 1		24		17.25		ns
27	$\overline{CS}$, $\overline{WE[0:3]}$ negated to A[0:31] invalid -GPCM- write access[5]		3		2.25		ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
27a	$\overline{WE[0:3]}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='0', CSNT = '1'. $\overline{CS}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='0', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 0		8		5.71		ns
27b	$\overline{WE[0:3]}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='1', CSNT = '1'. $\overline{CS}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='1', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 0		28		20		ns
27c	$\overline{WE[0:3]}$ negated to A[0:31] invalid -GPCM- write access, TRLX='0', CSNT = '1'. $\overline{CS}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='0', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 1		4		3		ns
27d	$\overline{WE[0:3]}$ negated to A[0:31] Invalid -GPCM- write access, TRLX='1', CSNT = '1'. CS negated to A[0:31] Invalid -GPCM- write access, TRLX='1', CSNT = '1', ACS = 10,ACS = ='11', EBDF = 1		24		17.25		ns
28	A[0:31], TSIZ[0:1], RD/$\overline{WR}$, $\overline{BURST}$, $\overline{BDIP}$ valid to CLKOUT Rising Edge. (Slave mode Setup Time)		11		8		ns
28a	Slave Mode D[0:31] valid to CLKOUT Rising Edge		7		7		ns

Table E-10 Bus Operation Timing (Continued)

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz[1]		Unit
			Min	Max	Min	Max	
29	$\overline{TS}$ valid to CLKOUT Rising Edge (Setup Time)		7		5		ns
30	CLKOUT Rising Edge to $\overline{TS}$ Valid (Hold Time).		2		2		ns

NOTES:

1. 56-MHz operation is available as an option. Some parts (without the 56-MHz option) will operate at a maximum frequency of 40 MHz.
2. The timing for $\overline{BR}$ output is relevant when the MPC565 / MPC566 is selected to work with external bus arbiter. The timing for $\overline{BG}$ output is relevant when the MPC565 / MPC566 is selected to work with internal bus arbiter.
3. The setup times required for $\overline{TA}$, $\overline{TEA}$, and $\overline{BI}$ are relevant only when they are supplied by the external device (and not the memory controller).
4. The maximum value of spec 8 for DATA[0:31] pins must be extended by 1.1 ns if the pins have been pre-charged to greater than V_{DDL}. This is the case if an external slave device on the bus is running at the max. value of VDATAPC. This is currently specified at 3.45 V. The 1.1 ns addition to spec 8 reflectls the expected timing degradation for 3.45 V.
5. The timing 27 refers to $\overline{CS}$ when ACS = ‘00’ and to $\overline{WE}$[0:3] when CSNT = ‘0’.

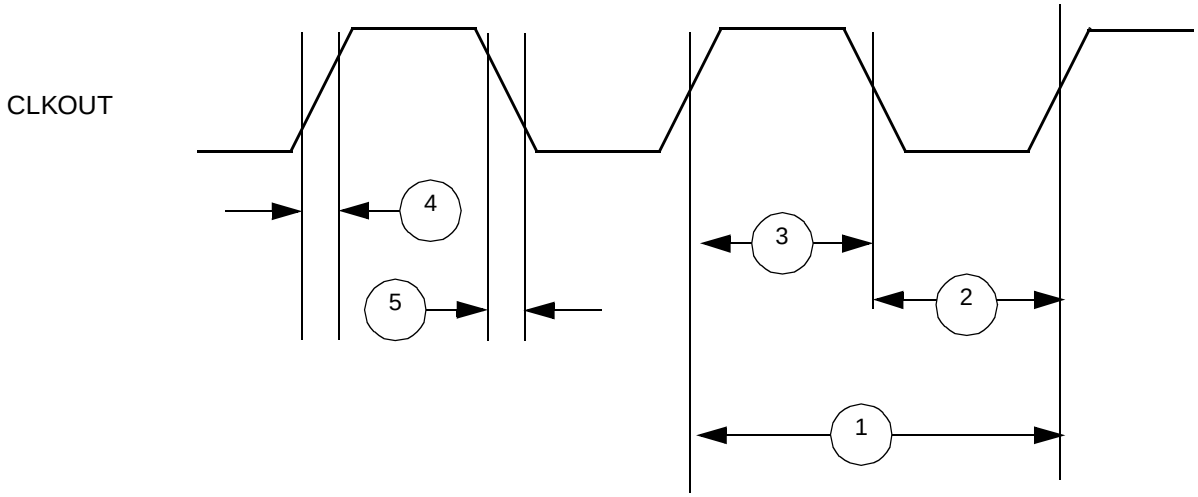


Figure E-10 CLKOUT Pin Timing

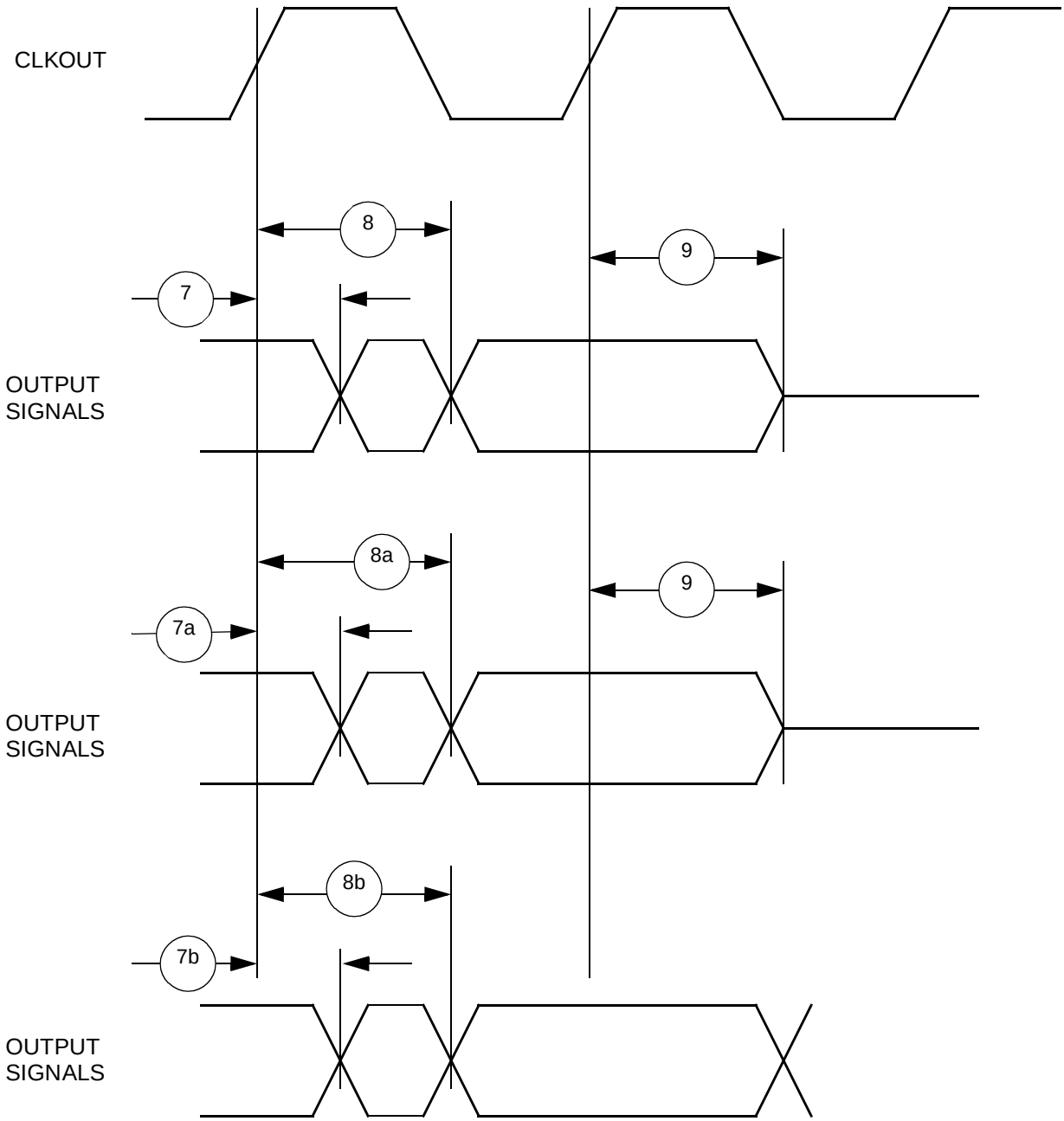


Figure E-11 Synchronous Output Signals Timing

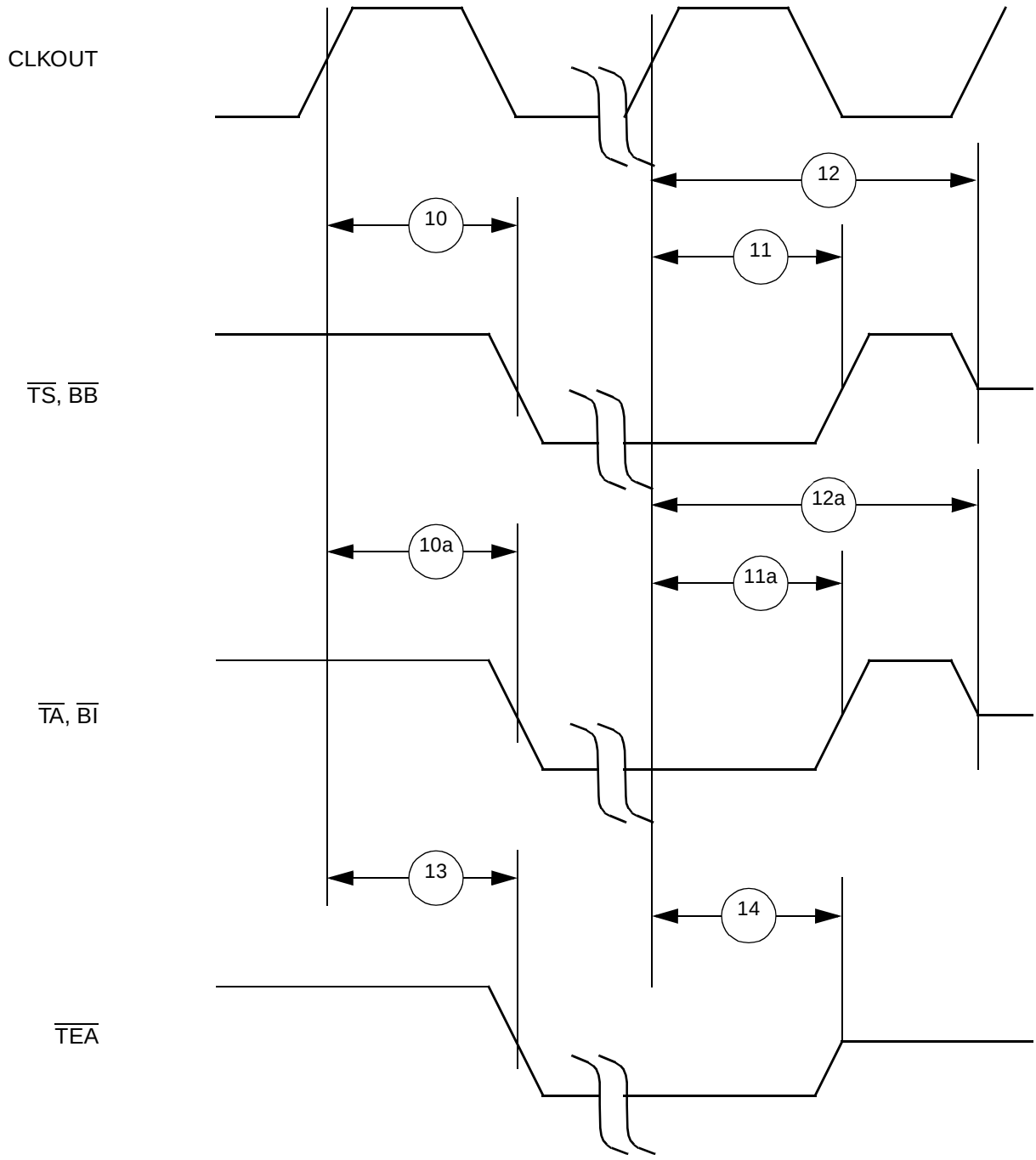


Figure E-12 Synchronous Active Pull-Up And Open Drain Outputs Signals Timing

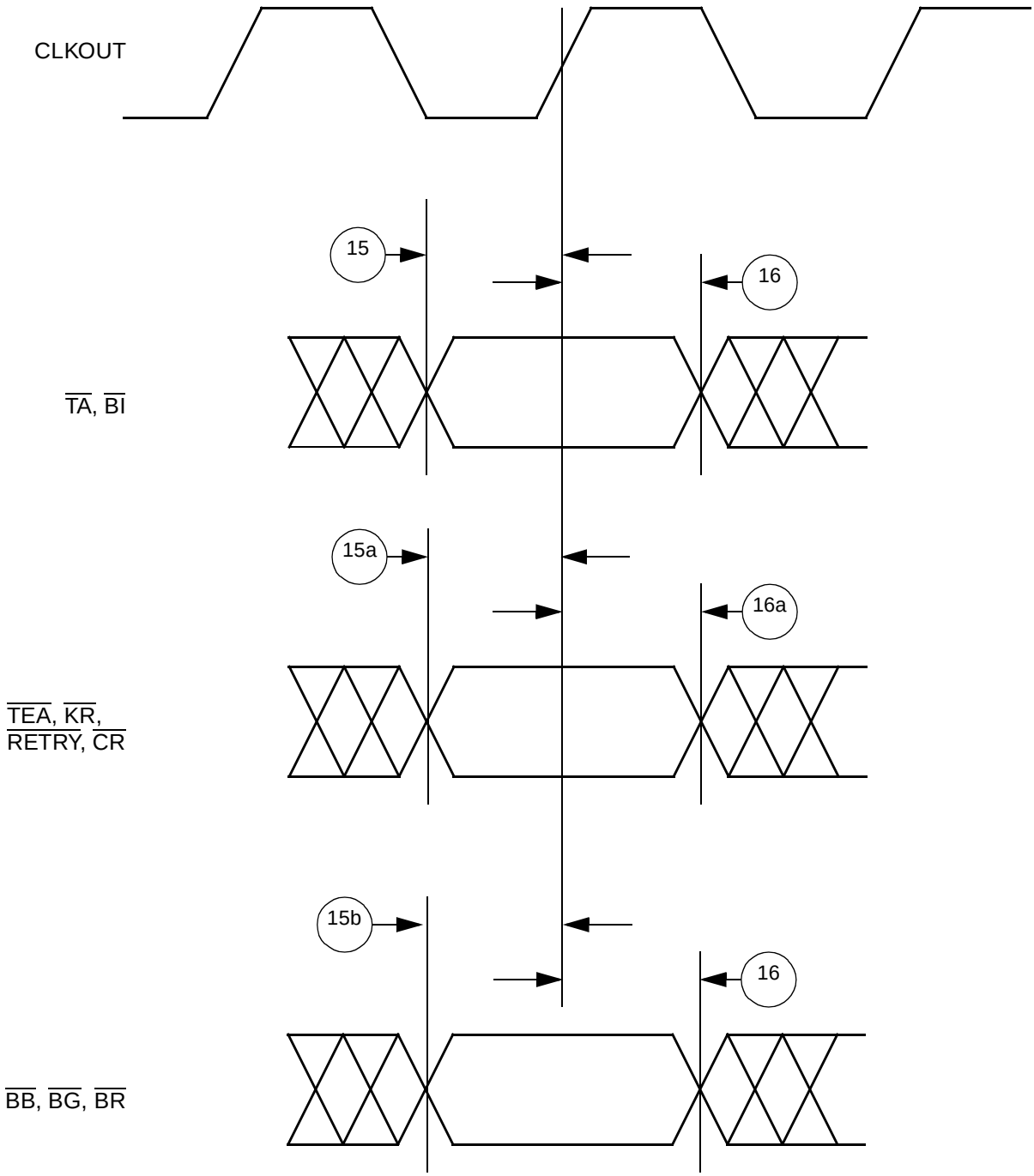


Figure E-13 Synchronous Input Signals Timing

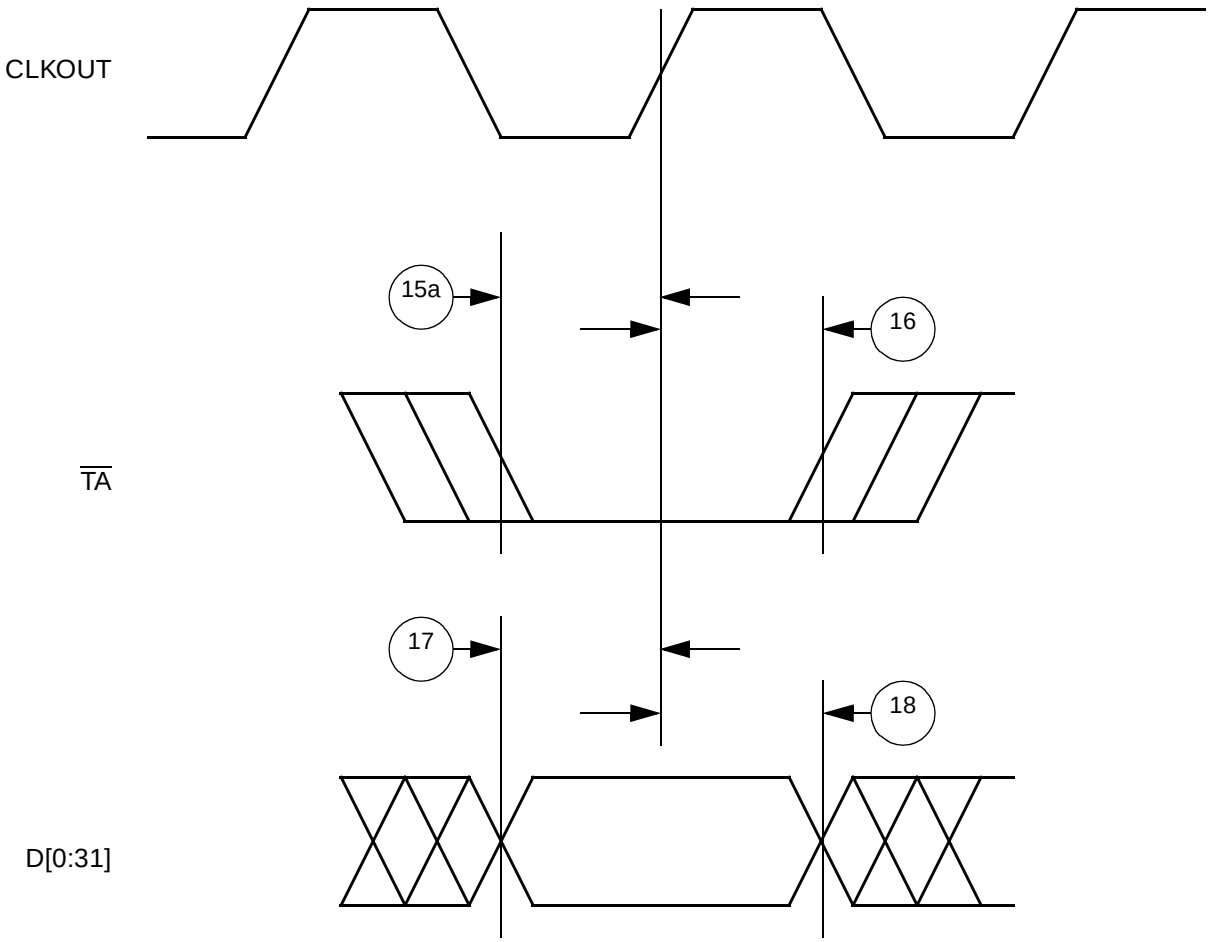


Figure E-14 Input Data Timing In Normal Case

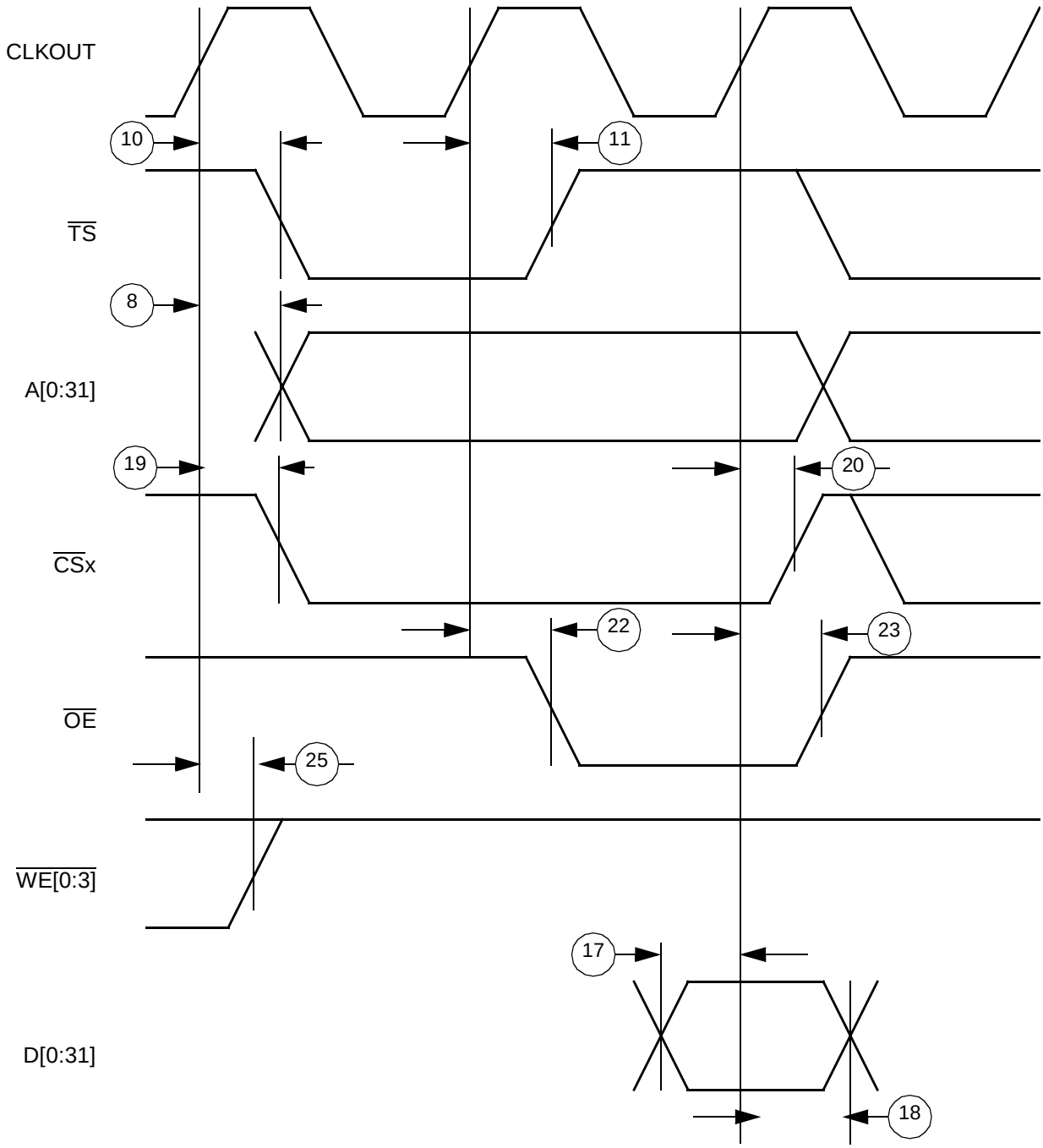


Figure E-15 External Bus Read Timing (GPCM Controlled – ACS = ‘00’)

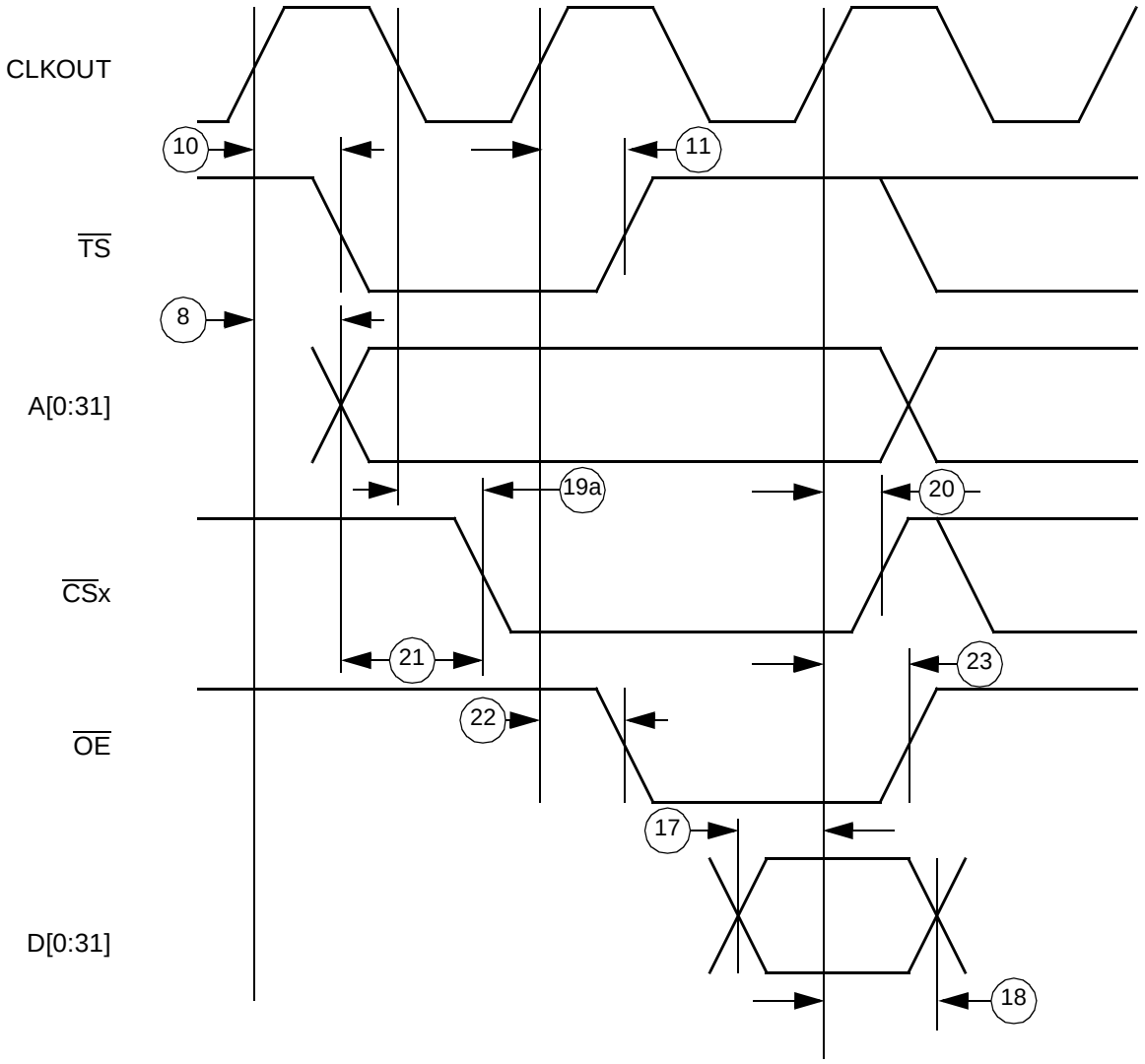


Figure E-16 External Bus Read Timing
(GPCM Controlled – TRLX = ‘0’ ACS = ‘10’)

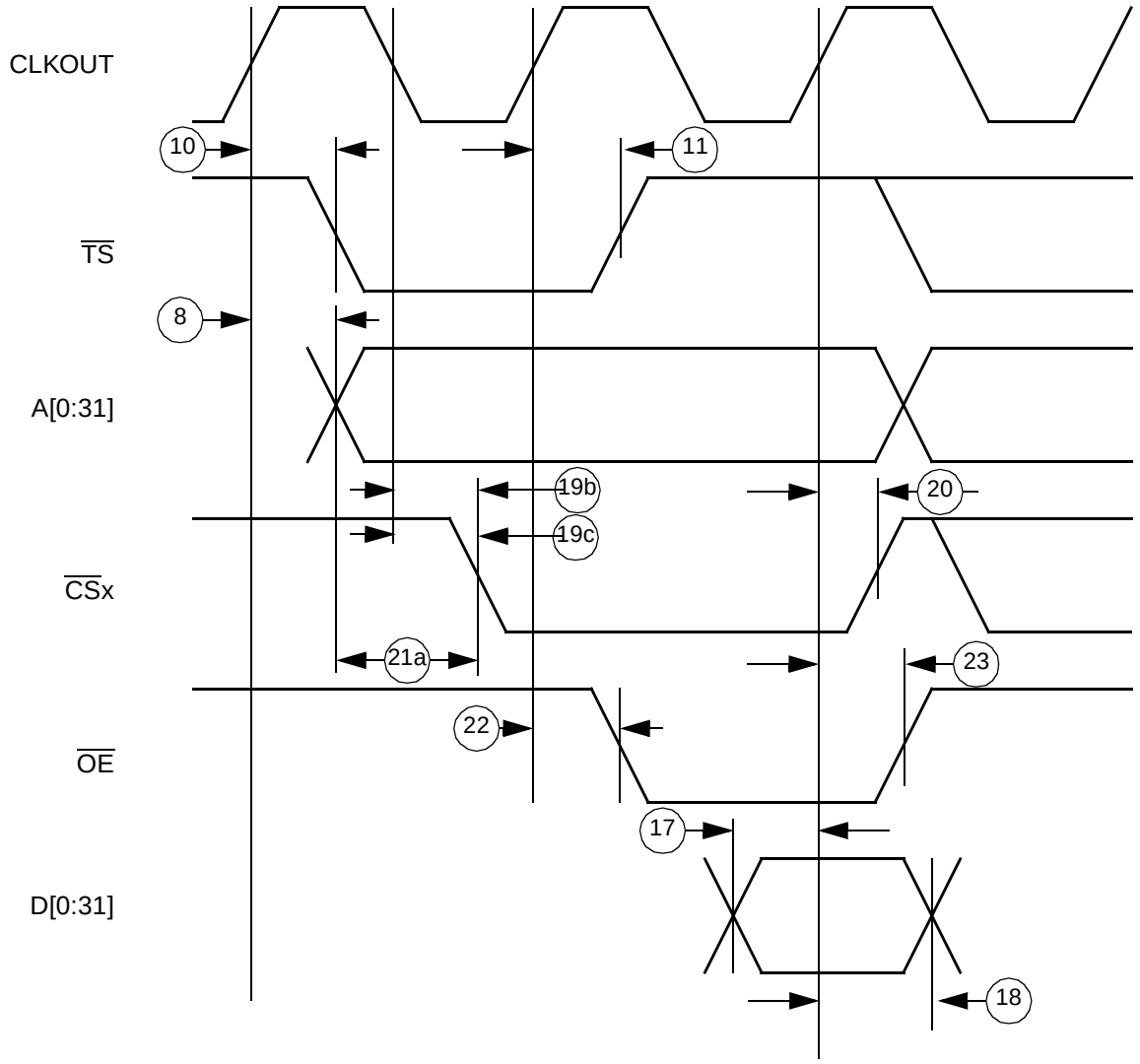


Figure E-17 External Bus Read Timing
(GPCM Controlled – TRLX = ‘0’ ACS = ‘11’)

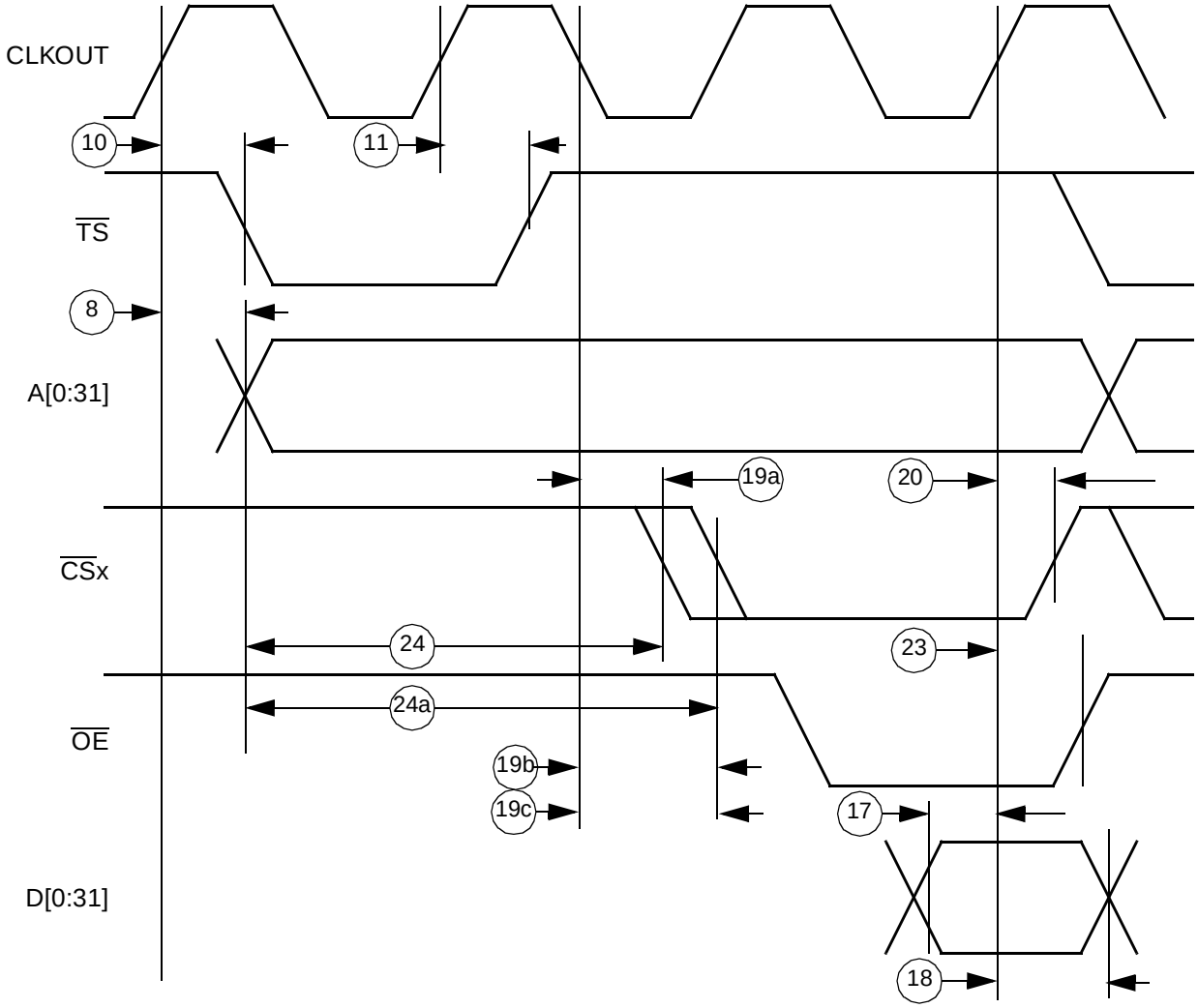


Figure E-18 External Bus Read Timing
(GPCM Controlled – TRLX = ‘1’, ACS = ‘10’, ACS = ‘11’)

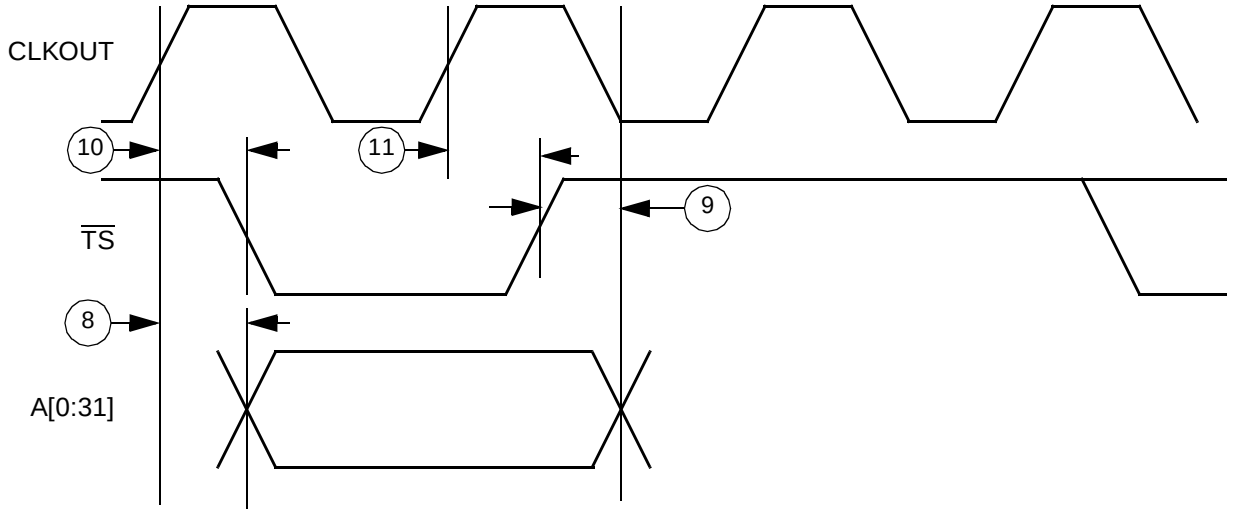


Figure E-19 Address Show Cycle Bus Timing

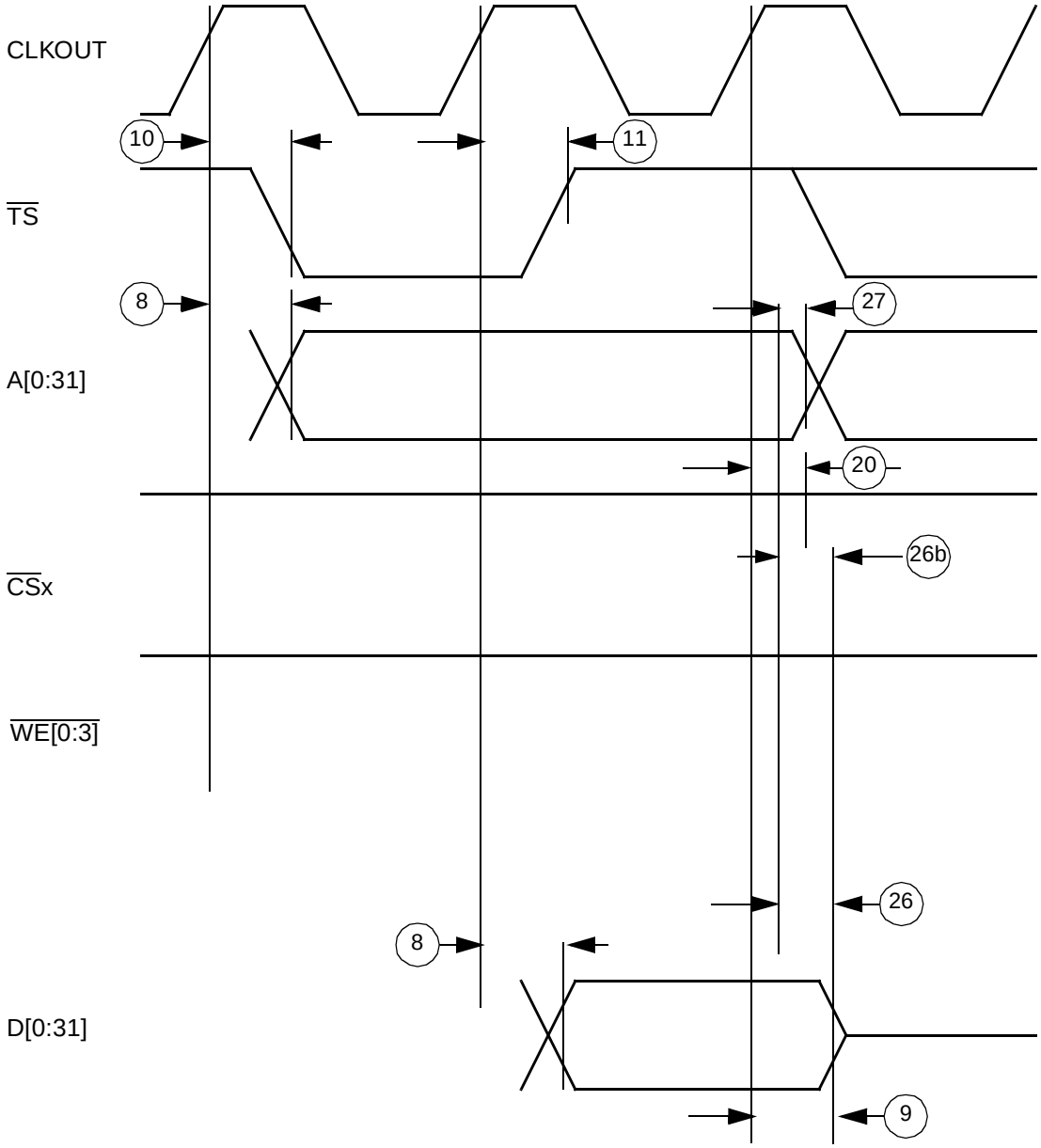


Figure E-20 Address and Data Show Cycle Bus Timing

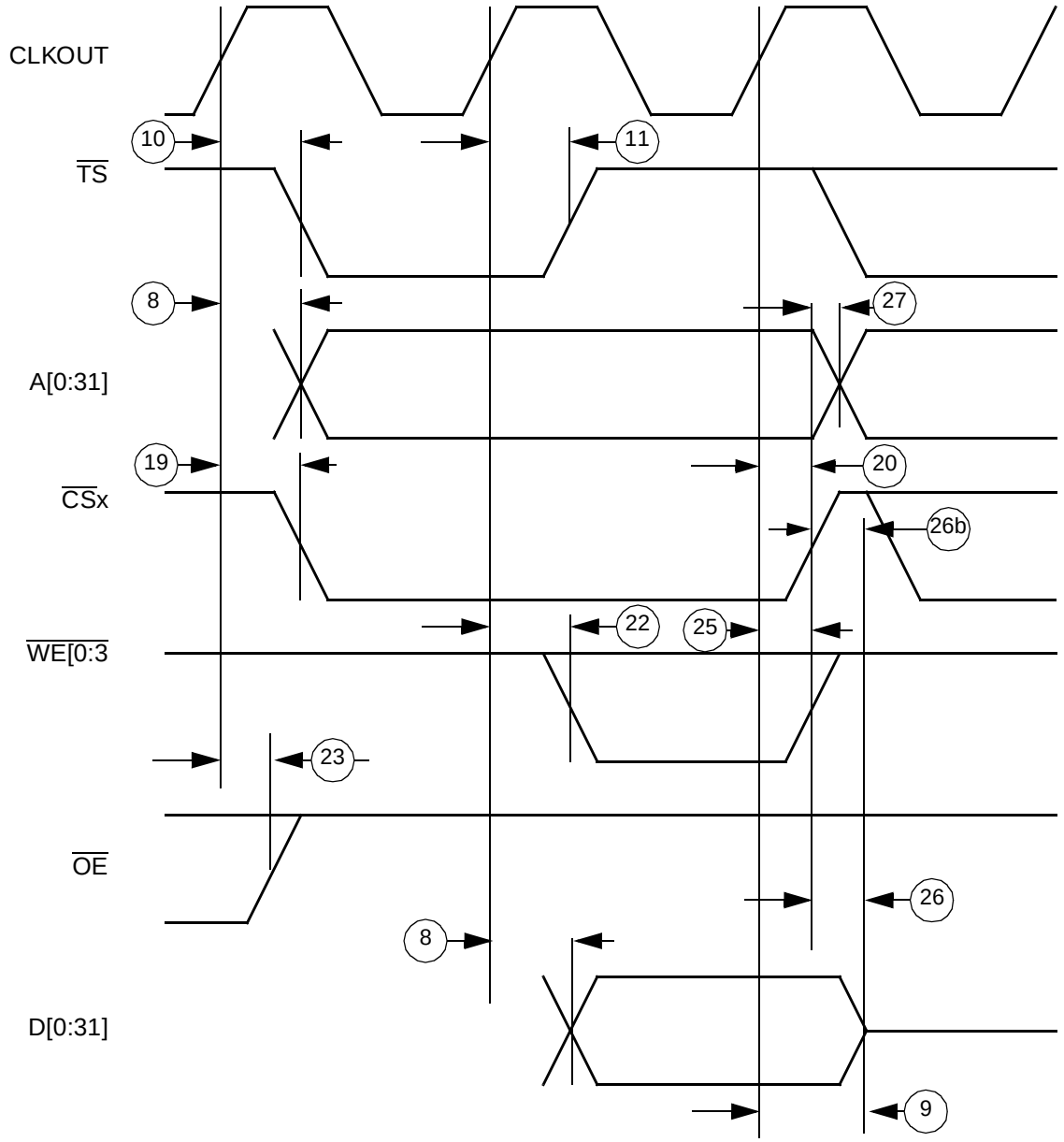


Figure E-21 External Bus Write Timing
(GPCM Controlled – TRLX = ‘0’, CSNT = ‘0’)

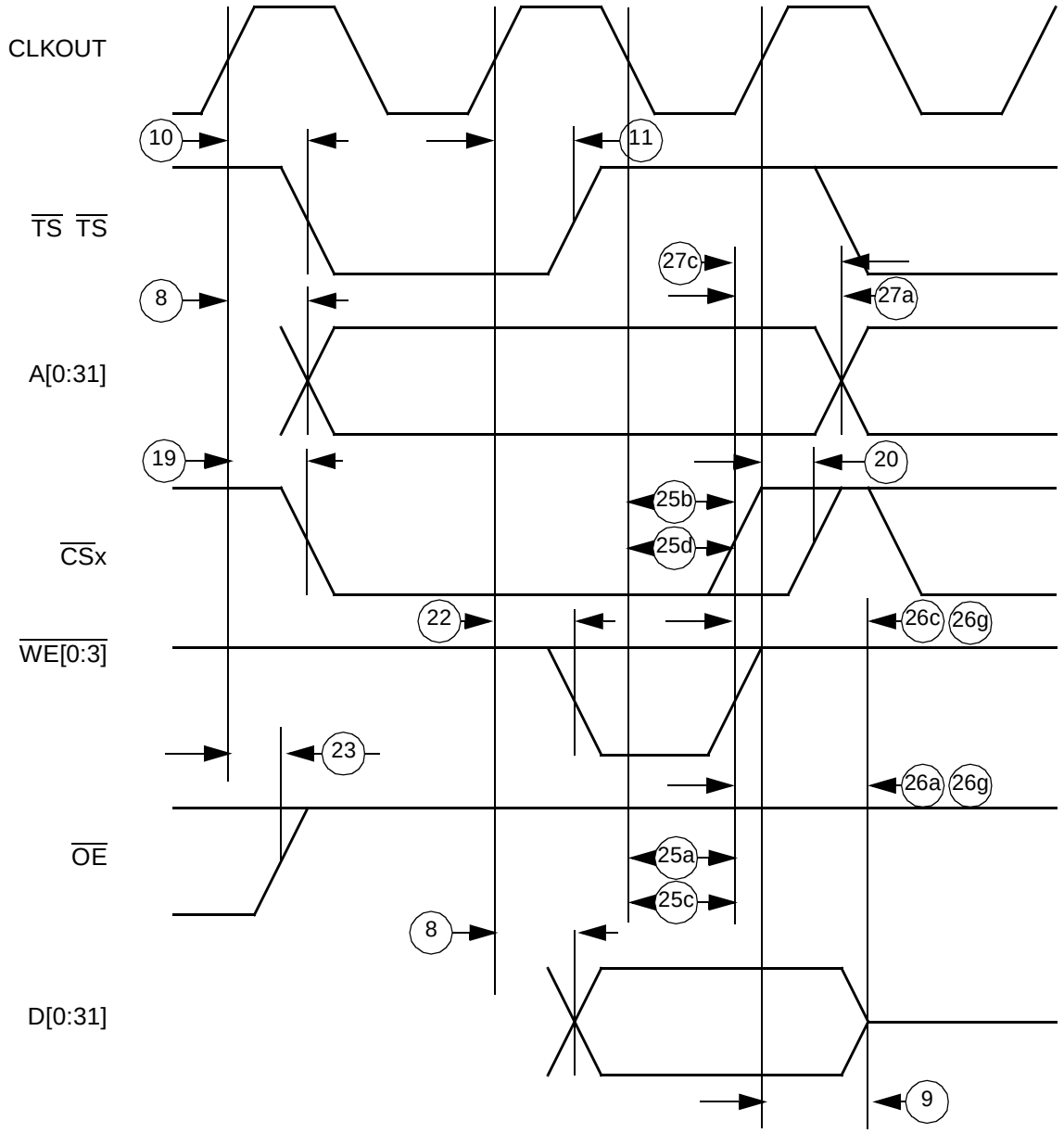


Figure E-22 External Bus Write Timing
(GPCM Controlled – TRLX = ‘0’, CSNT = ‘1’)

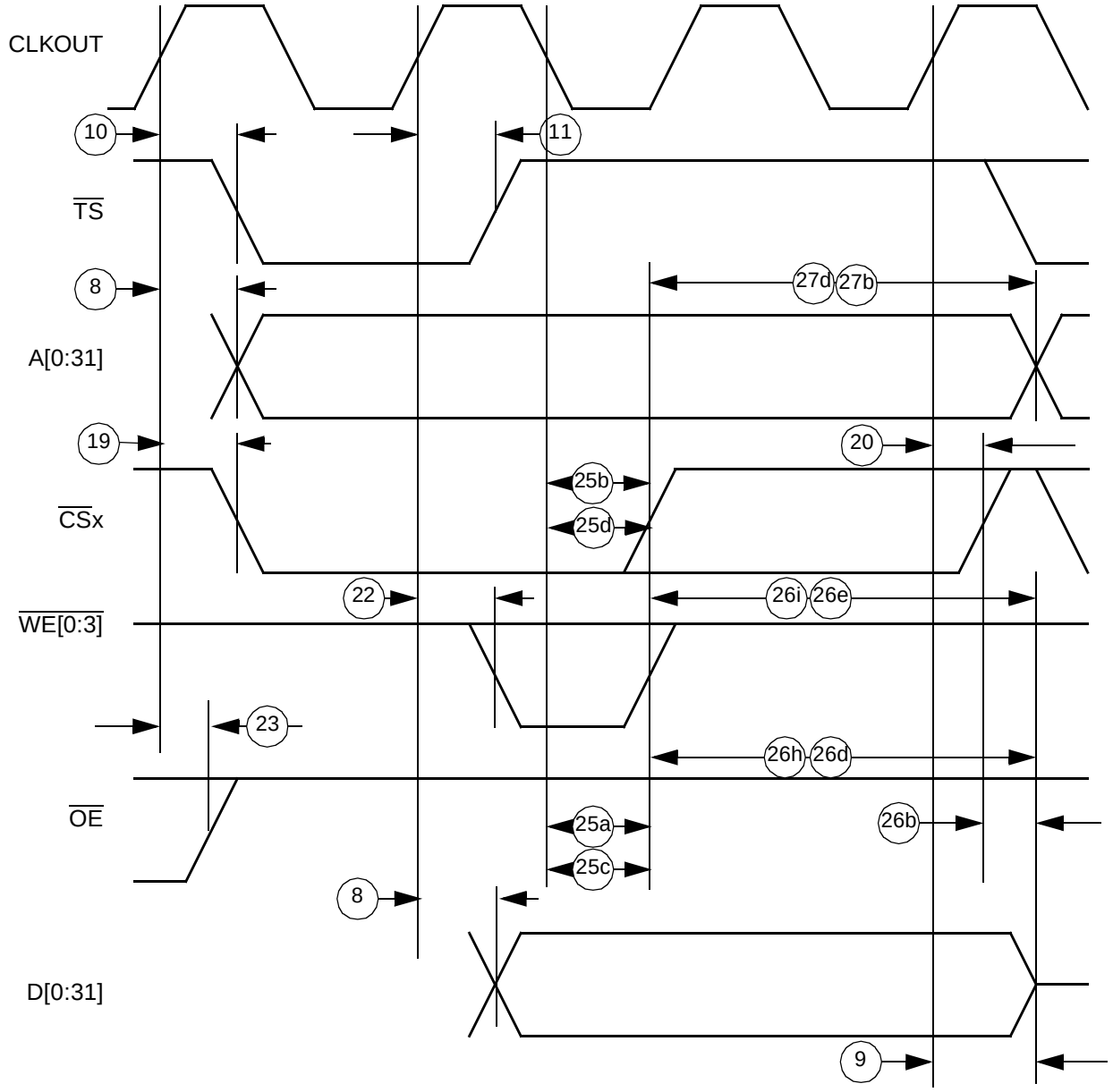


Figure E-23 External Bus Write Timing
(GPCM Controlled – TRLX = ‘1’, CSNT = ‘1’)

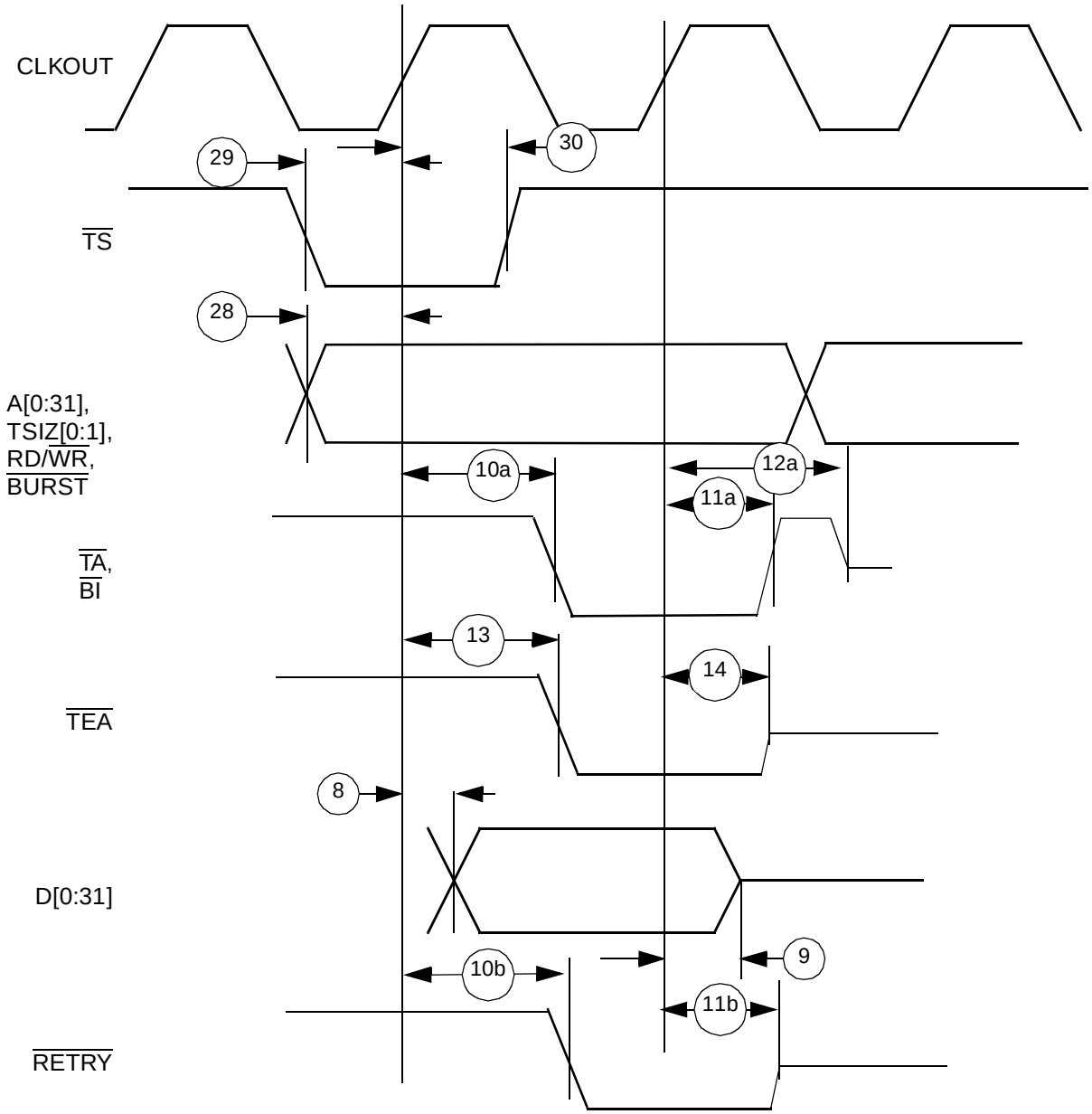


Figure E-24 External Master Read From Internal Registers Timing

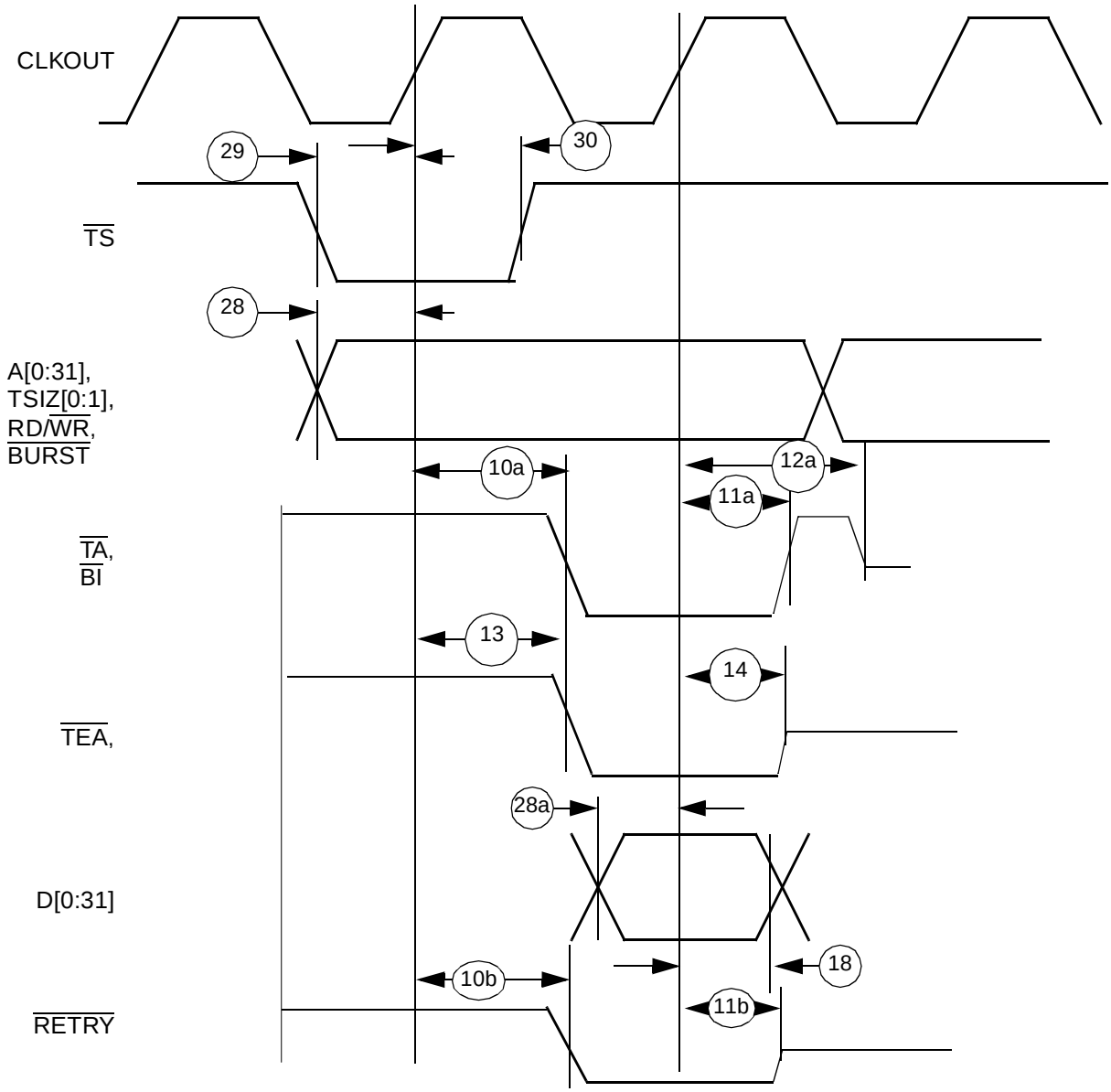


Figure E-25 External Master Write To Internal Registers Timing

Table E-11 Interrupt Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz		Unit
			Min	Max	Min	Max	
31	$\overline{IRQ}$x valid to CLKOUT rising edge (Setup Time)[1]		10		10		ns
32	$\overline{IRQ}$x hold time after CLKOUT[1]		2		2		ns
33	$\overline{IRQ}$x Pulse width Low		3		3		ns
34	$\overline{IRQ}$x Pulse width High		3		3		ns
35	$\overline{IRQ}$x Edge to Edge time	4 * TC	100		100		ns

NOTES:

1. The timings 31 and 32 describe the testing conditions under which the $\overline{IRQ}$ lines are tested when being defined as level sensitive. The $\overline{IRQ}$ lines are synchronized internally and do not have to be asserted or negated with reference to the CLKOUT.

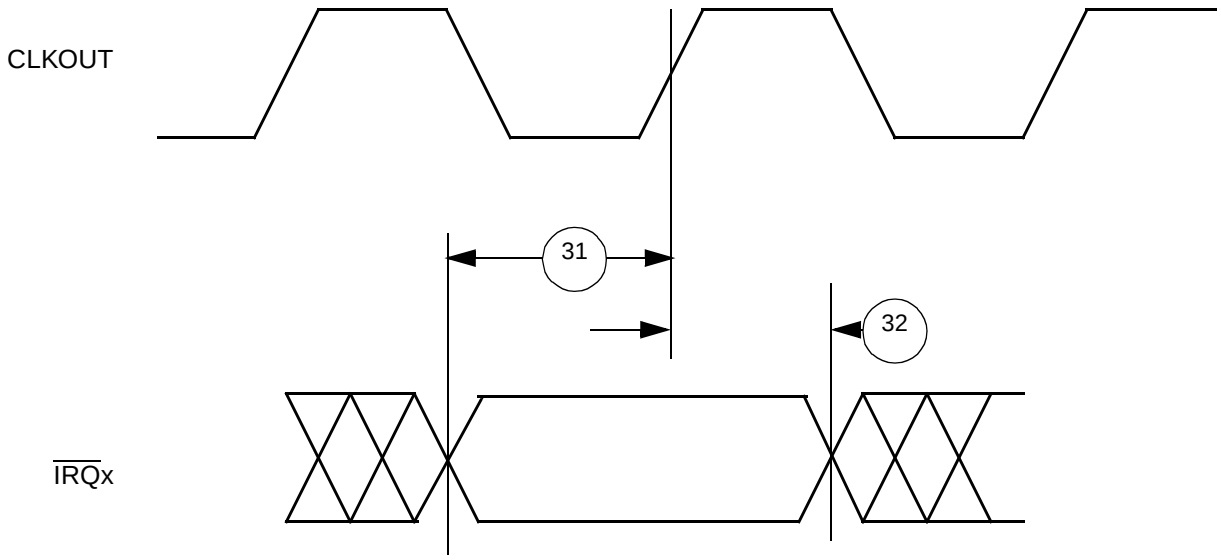


Figure E-26 Interrupt Detection Timing for External Level Sensitive Lines

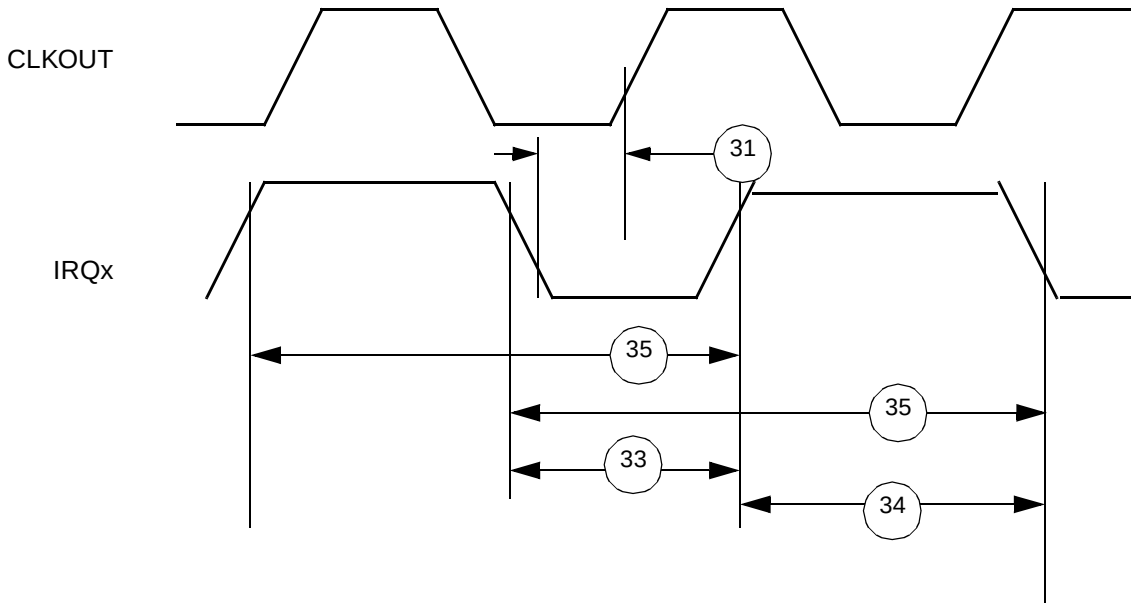


Figure E-27 Interrupt Detection Timing for External Edge Sensitive Lines

E.12 Debug Port Timing

Table E-12 Debug Port timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz		Unit
			Min	Max	Min	Max	
36	DSCK Cycle Time		60	—	TBD	—	ns
37	DSCK Clock Pulse Width		25	—	TBD	—	ns
38	DSCK Rise and Fall Times		0	3	0	3	ns
39	DSDI Input Data Setup Time		15	—	TBD	—	ns
40	DSDI Data Hold Time		5	—	TBD	—	ns
41	DSCK low to DSDO Data Valid		0	18	0	18	ns
42	DSCK low to DSDO Invalid		0	—	0	—	ns

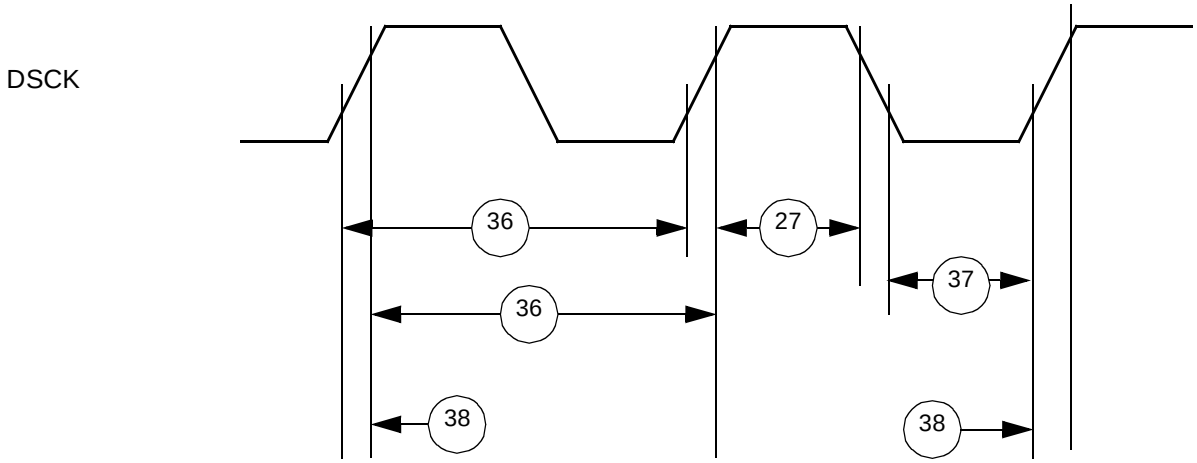


Figure E-28 Debug Port Clock Input Timing

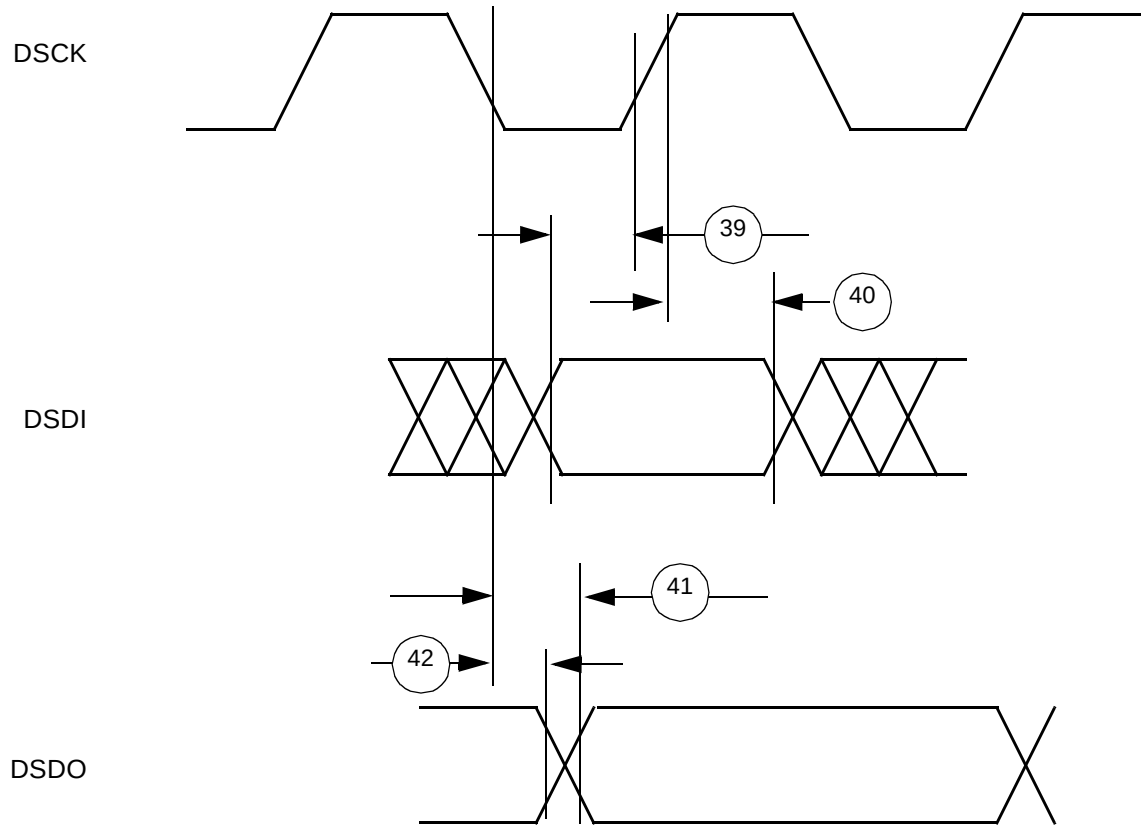


Figure E-29 Debug Port Timings

E.13 Pin Electrical Characteristics

E.13.1 AC Electrical Characteristics

The AC electrical characteristics (56 MHz) are described in the following tables and figures

Table E-13 AC Electrical Characteristics

Number	Characteristic	Min	Max	Unit
1	MCKO Cycle Time (**Tco**)	17.9	—	ns
2	MCKO Duty Cycle	40	60	%
3	Output Rise and Fall Times	0	3	ns
4	MCKO low to MDO Data Valid	(-0.10)Tco	(0.20) Tco	ns
5	MCKI Cycle Time (**Tci**)	35.6	—	ns
6	MCKI Duty Cycle	40	60	%
7	Input Rise and Fall Times	0	3	ns
8	MDI, $\overline{\text{EVTI}}$, $\overline{\text{MSEI}}$ Setup Time	(0.20)Tci	—	ns
9	MDI Hold TIme	(0.10)Tci	—	ns
10	$\overline{\text{RSTI}}$ Pulse Width	(4.0) Tco	—	ns
11	MCKO low to $\overline{\text{MSEO}}$ Valid	(-0.10)Tco	(0.20) Tco	ns
12	$\overline{\text{EVTI}}$ Pulse Width	(4.0) Tco	—	ns
13	$\overline{\text{EVTI}}$ to $\overline{\text{RSTI}}$ Setup (at reset only)	(4.0) System Clock	—	ns
14	$\overline{\text{EVTI}}$ to $\overline{\text{RSTI}}$ Hold (at reset only)	(4.0) System Clock	—	ns

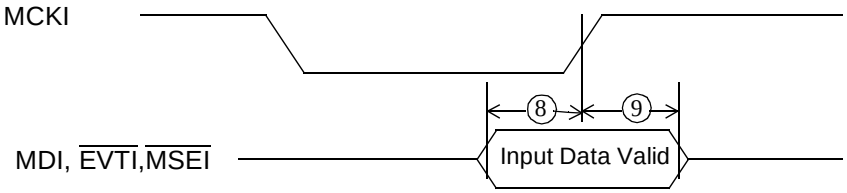


Figure E-30 Auxiliary Port Data Input Timing Diagram

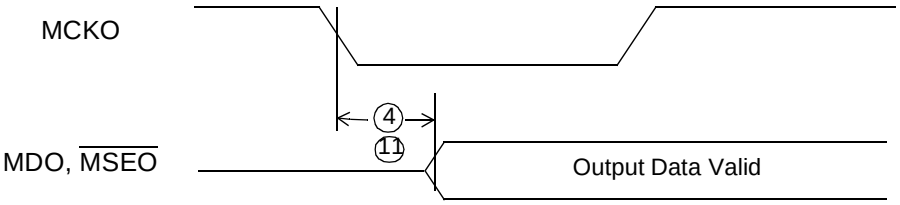


Figure E-31 Auxiliary Port Data Output Timing Diagram

MDO and $\overline{\text{MSEO}}$ data is held valid until the next MCKO low transition.

When $\overline{\text{RSTI}}$ is asserted, $\overline{\text{EVTI}}$ is used to enable or disable the auxiliary port. Because MCKO probably is not active at this point, the timing must be based on the system clock. Since the system clock is not realized on the connector, its value must be known by the tool.

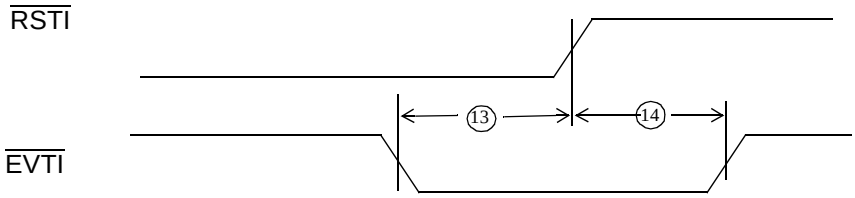


Figure E-32 Enable Auxiliary From $\overline{\text{RSTI}}$

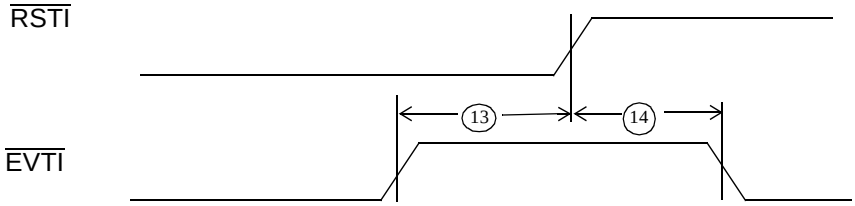


Figure E-33 Disable Auxiliary From $\overline{\text{RSTI}}$

E.14 RESET Timing

Table E-14 RESET Timing

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz		Unit
			Min	Max	Min	Max	
43	CLKOUT to $\overline{\text{HRESET}}$ high impedance			20		20	ns
44	CLKOUT to $\overline{\text{SRESET}}$ high impedance			20		20	ns
45	$\overline{\text{RSTCONF}}$ pulse width	17 * TC	425		302		ns
46	Configuration Data to $\overline{\text{HRESET}}$ rising edge Setup Time	15 * TC + TCC	382		272		ns
47	Configuration Data to $\overline{\text{RSTCONF}}$ rising edge set up time	15 * TC + TCC	382		272		ns
48	Configuration Data hold time after $\overline{\text{RSTCONF}}$ negation		0		0		ns
49	Configuration Data hold time after $\overline{\text{HRESET}}$ negation		0		0		ns
49a	$\overline{\text{RSTCONF}}$ hold time after $\overline{\text{HRESET}}$ negation[1]		50		35		
50	$\overline{\text{HRESET}}$ and $\overline{\text{RSTCONF}}$ asserted to Data out drive		25		25		ns
51	$\overline{\text{RSTCONF}}$ negated to Data out high impedance		25		25		ns
52	CLKOUT of last rising edge before chip tristates $\overline{\text{HRESET}}$ to Data out high impedance		25		25		ns
53	DSDI, DSCK set up	3 * TC	75		55		ns
54	DSDI, DSCK hold time		0		0		ns
55	$\overline{\text{SRESET}}$ negated to CLKOUT rising edge for DSDI and DSCK sample	8 * TC	200		142		ns
55a	$\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$, $\overline{\text{PORESET}}$ pulse width [2]		100		100		ns

NOTES:

1. Weak pull-ups and pull-downs used for Reset timing will comply with the 130 µA mode select current outlined in **E.6 DC Electrical Characteristics** The system requires two clocks of hold time on $\overline{\text{RSTCONF}}$/TEXP after negation of $\overline{\text{HRESET}}$. The simplest way to insure meeting this requirement in systems that require the use of the TEXP function, is to connect $\overline{\text{RSTCONF}}$/TEXP to $\overline{\text{SRESET}}$.
2. $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{PORESET}}$ have a glitch detector to ensure that spikes less than 20 ns are rejected. The internal $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{PORESET}}$ will assert only if these signals are asserted for more than 100 ns

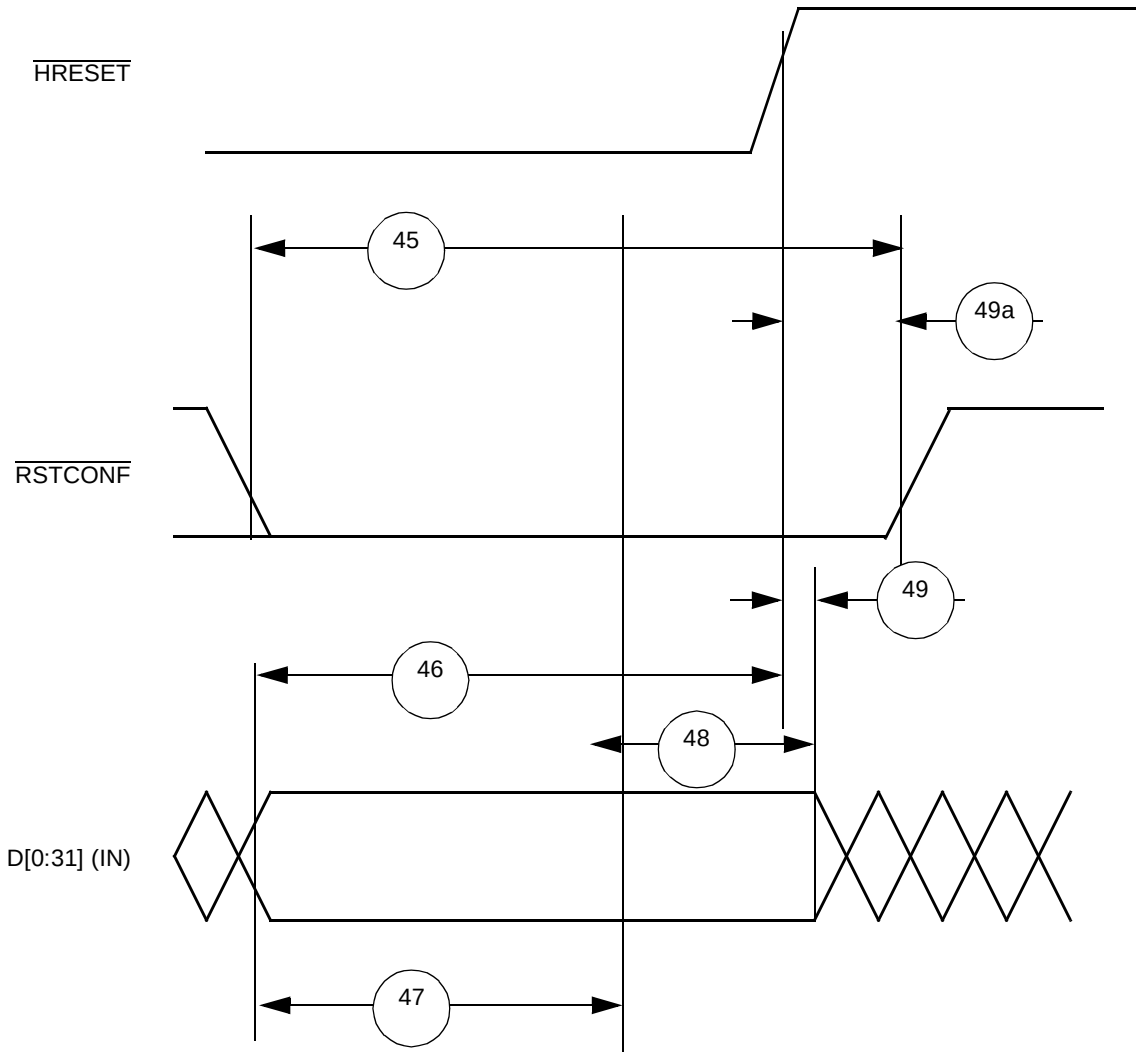


Figure E-34 Reset Timing – Configuration from Data Bus

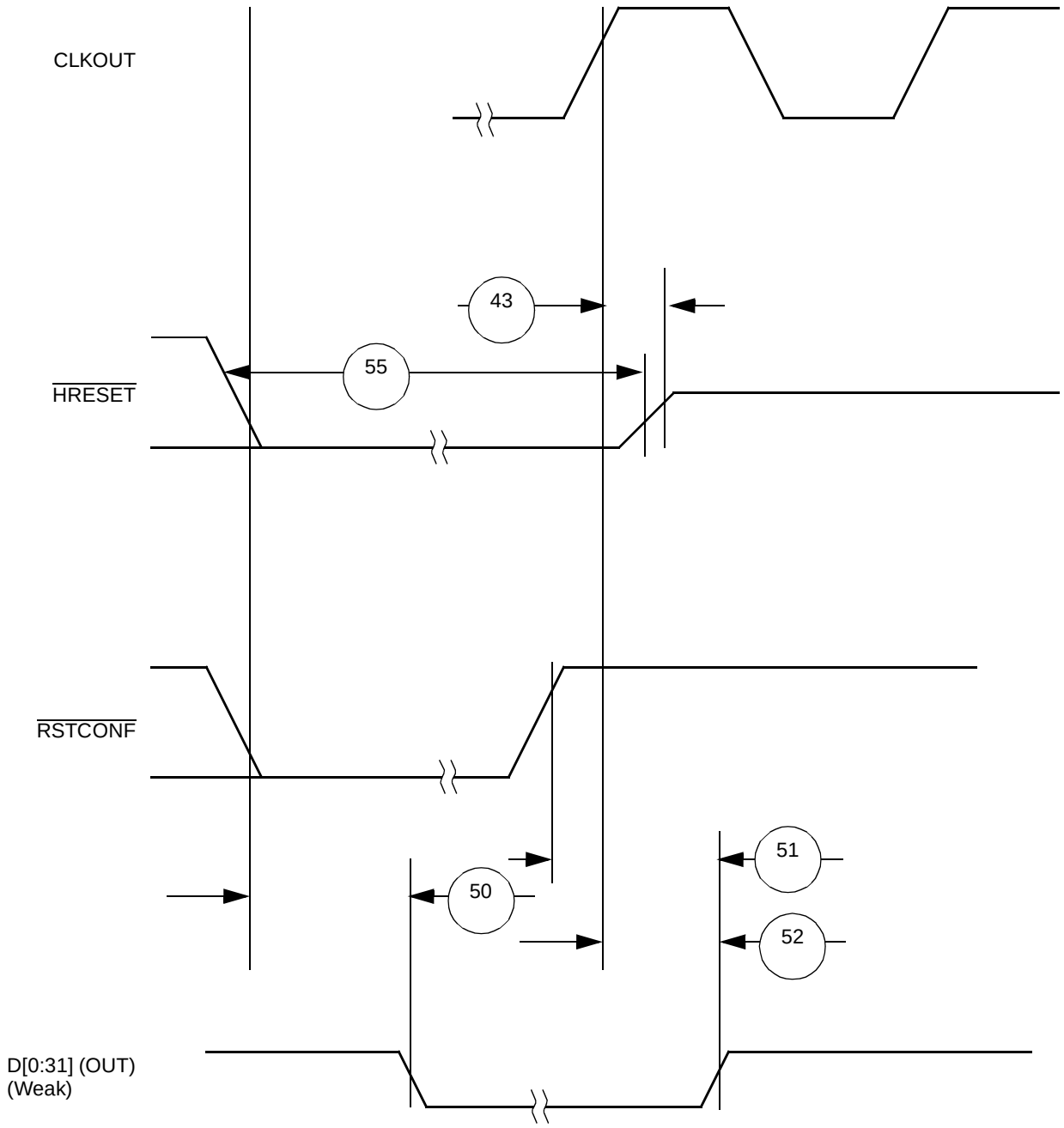


Figure E-35 Reset Timing – Data Bus Weak Drive During Configuration

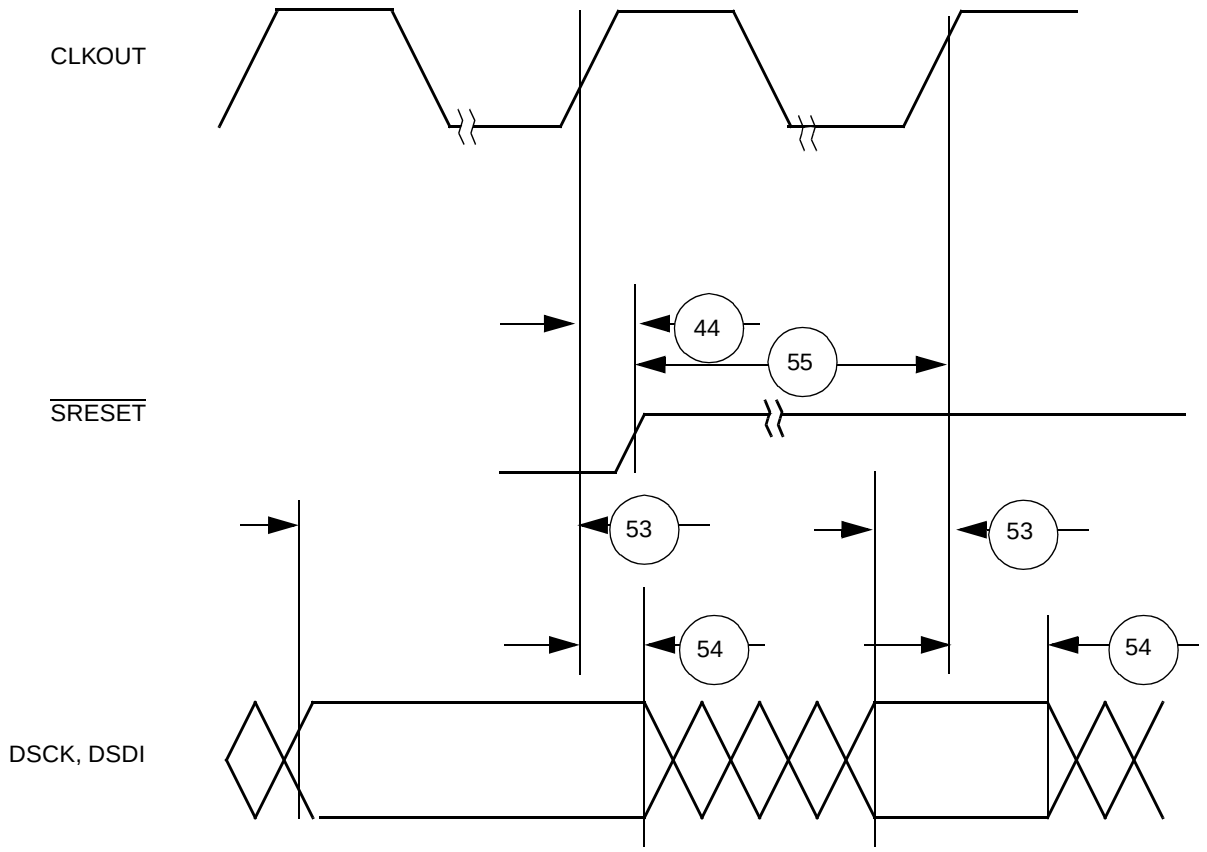


Figure E-36 Reset Timing – Debug Port Configuration

E.15 IEEE 1149.1 Electrical Characteristics

Table E-15 JTAG timing[1]

($T_A = T_L$ to T_H)

	Characteristic	Expression	40 MHz		56 MHz		Unit
			Min	Max	Min	Max	
56	TCK Cycle Time		—	100	—	100	ns
57	TCK Clock Pulse Width Measured at $V_{DD}/2$		—	40	—	40	ns
58	TCK Rise and Fall Times		0	10	0	10	ns
59	TMS, TDI Data Setup Time		5		5		ns
60	TMS, TDI Data Hold Time		25		25		ns
61	TCK Low to TDO Data Valid			20		20	ns
62	TCK Low to TDO Data Invalid		0		0		ns
63	TCK Low to TDO High Impedance			20		20	ns
64	$\overline{\text{TRST}}$ Assert Time		100		100		ns
65	$\overline{\text{TRST}}$ Setup Time to TCK Low		40		40		ns
66	TCK Falling Edge to Output Valid			50		50	ns
67	TCK Falling Edge to Output Valid out of High Impedance			50		50	ns
68	TCK Falling Edge to Output High Impedance			50		50	ns
69	Boundary Scan Input Valid to TCK Rising Edge		50		50		ns
70	TCK Rising Edge to Boundary Scan Input Invalid		50		50		ns

NOTES:
1. JTAG timing is only tested at 10 MHz

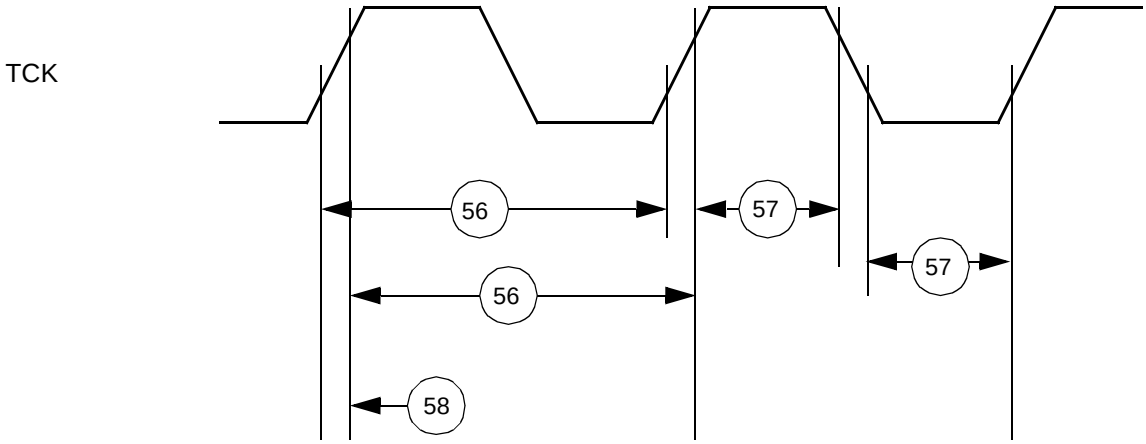


Figure E-37 JTAG Test Clock Input Timing

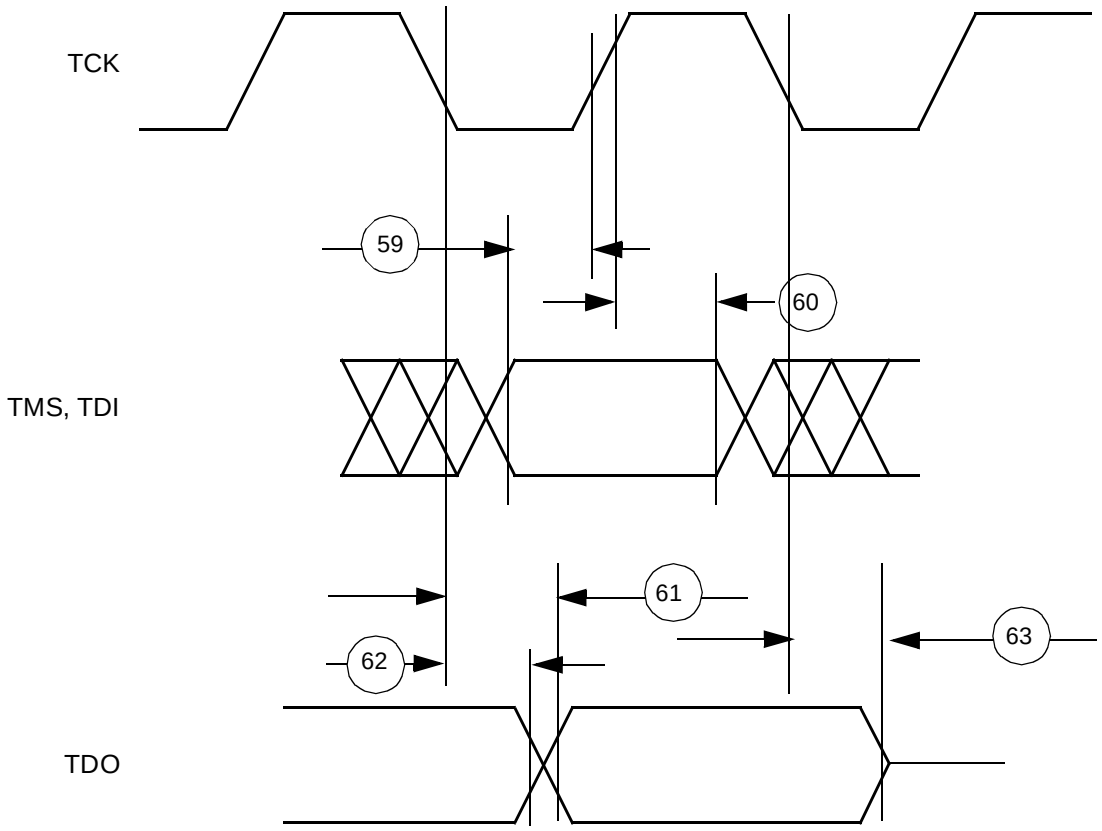


Figure E-38 JTAG Test Access Port Timing Diagram

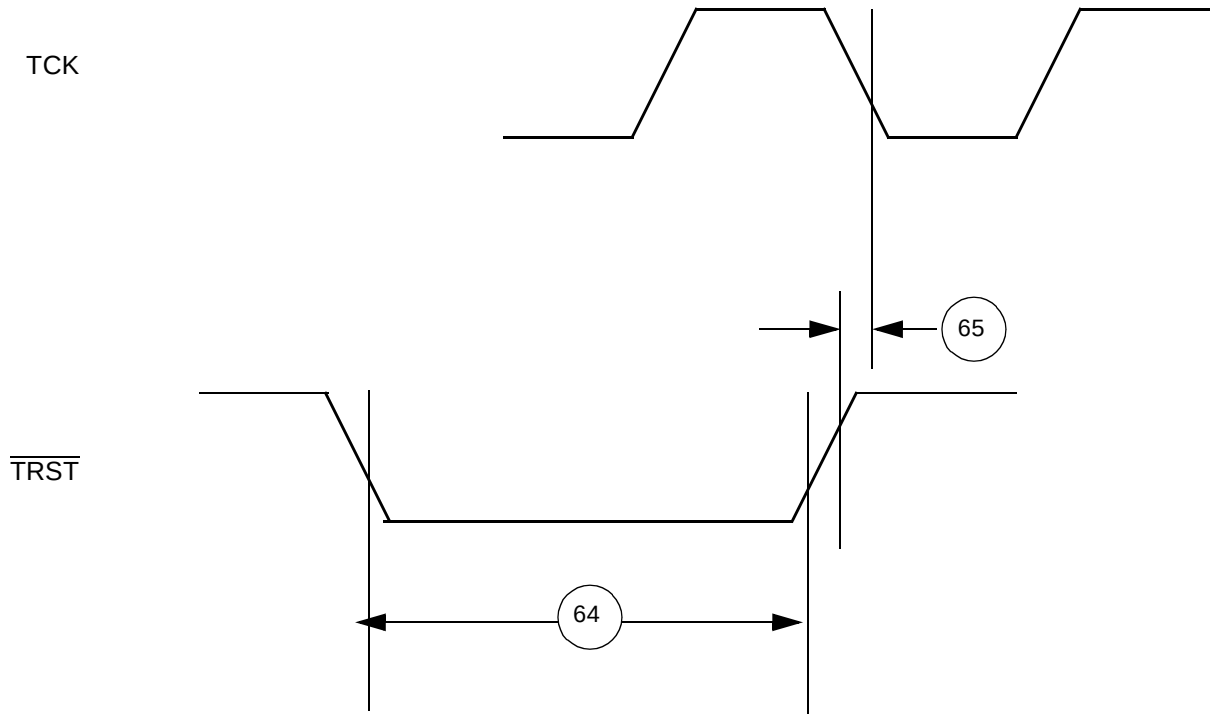


Figure E-39 JTAG $\overline{\text{TRST}}$ Timing Diagram

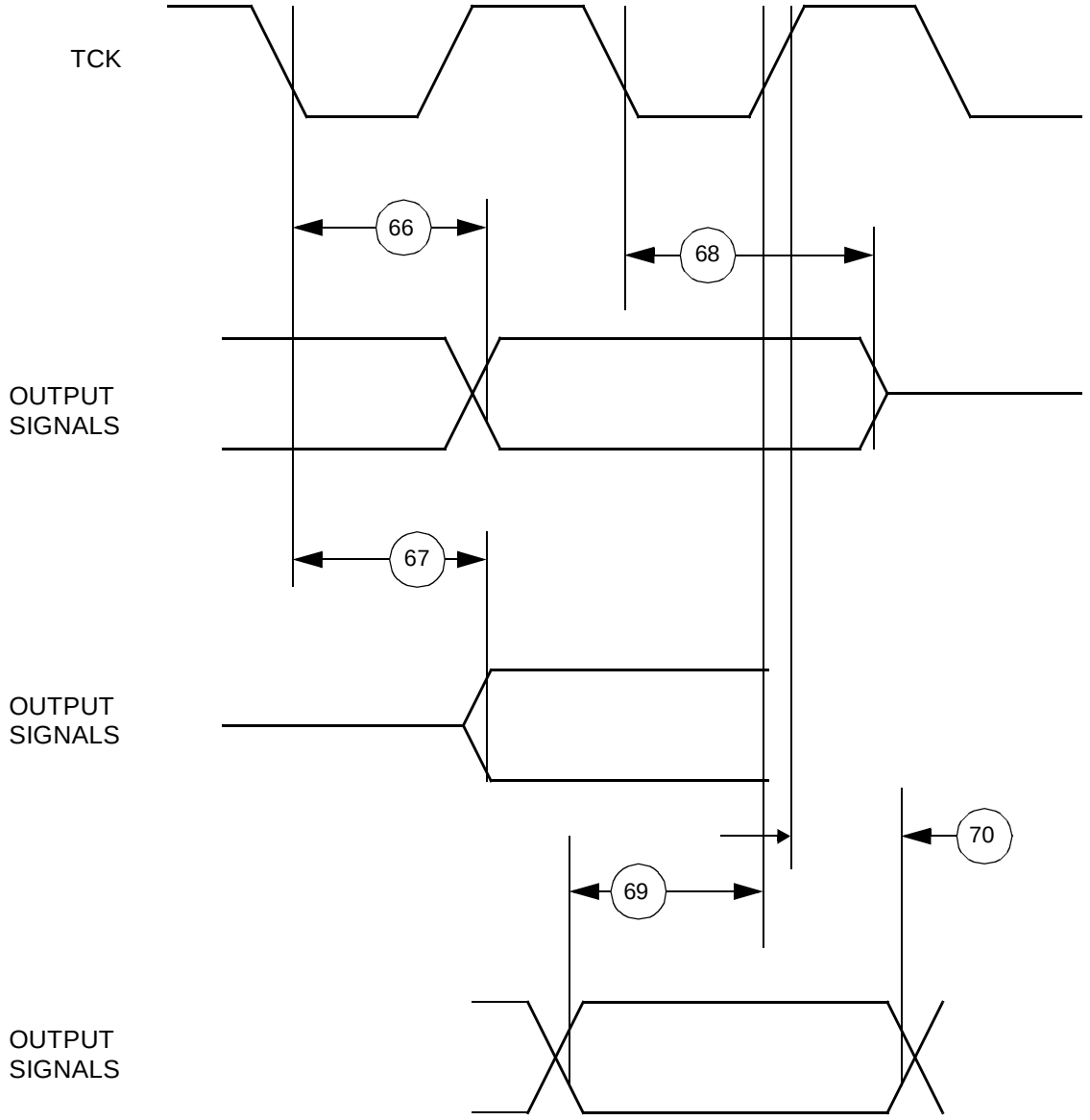


Figure E-40 Boundary Scan (JTAG) Timing Diagram

E.16 QADC64E Electrical Characteristics

Table E-16 QADC64E Conversion Characteristics (Enhanced Mode Operation)

($T_A = T_L$ to T_H)

Num	Parameter	Symbol	Min	Max	Units
97	QADC Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	3.0	MHz
98	Conversion Cycles[2]	CC	14	20	QCLK cycles
99	Conversion Time[3] F_{QCLK} = 2.8MHz[1,4] Min = CCW/IST =1b0 Max = CCW/IST =1b1	T_{CONV}	7.0	 7.25	 µs µs
100	Stop Mode Recovery Time	T_{SR}	—	10	µs
101	Resolution[5]	—	5	—	mV
102	Absolute (total unadjusted) error[6, 7, 8, 9] F_{QCLK} = 3.0MHz[3], 2 clock input sample time	AE	- 2	2	Counts
103	DC Disruptive Input Injection Current[10, 11, 12, 13, 14]	I_{INJ}[15] I_{INJ}[16]	- 3 -1	3 1	mA mA
104	Current Coupling Ratio[17] PQA PQB	K	 — —	 $8x10^{-5}$ $8x10^{-5}$	
105	Incremental Error due to injection current All channels have same 10kΩ < Rs <100kΩ Channel under test has Rs=10kΩ, I_{INJ}=±3mA	E_{INJ}		 ±1.0 ±1.0	 Counts Counts
106	Source impedance at input [18]	R_S	—	100	kΩ
107	Incremental Capacitance during Sampling [19]	C_{SAMP}	—	5	pF

NOTES:

1. Conversion characteristics vary with F_{QCLK} rate. Reduced conversion accuracy occurs at max F_{QCLK} rate.
2. The number of conversion cycles is dependent on the IST bits in the CCW register.
3. Assumes that f_{SYS} = 56 MHz
4. Assumes F_{QCLK} = 2.80 MHz, with clock prescaler values of: 20 * CCW: BYP=0b0
5. At $V_{RH} – V_{RL}$ = 5.12 V, one count = 5 mV.
6. Accuracy tested and guaranteed at $V_{RH} – V_{RL}$ = 5.0 V ± 0.5 V
7. This parameter is periodically sampled rather than 100% tested.
8. Absolute error includes 1/2 count (~2.5 mV) of inherent quantization error and circuit (differential, integral, and offset) error. Specification assumes that adequate low-pass filtering is present on analog input pins — capacitive filter with 0.01 µF to 0.1 µF capacitor between analog input and analog ground, typical source isolation impedance of 10 KΩ.
9. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals may affect the conversion accuracy of other channels.
10. Below disruptive current conditions, the channel being stressed has conversion values of 0x3FF for analog inputs greater than V_{RH} and 0x000 for values less than V_{RL}. This assumes that $V_{RH} \leftarrow V_{DDA}$ and $V_{RL} \leftarrow V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
11. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
12. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using $V_{POSCLAMP}$ = (the lower of V_{DDA} or V_{DDH}) + 0.3 V and $V_{NEGCLAMP}$ = – 0.3 V, then use the larger of the calculated values. The diode drop voltage is a function of current and varies approximately 0.4 to 0.8 V over temperature
13. This parameter is periodically sampled rather 100% tested.
14. Derate linearly to 0.3 mA if VDDH - VDDA = 1 V. This specification is preliminary and may change after further characterization.

15. Condition applies to two adjacent pins.
16. Condition applies to all analog channels.
17. Current Coupling Ratio, K, is defined as the ratio of the output current, I_{OUT}, measured on the pin under test to the injection current, I_{INJ}, when both adjacent pins are overstressed with the specified injection current. $K = I_{OUT}/ I_{INj}$ The input voltage error on the channel under test is calculated as $Verr = I_{INJ} * K * R_S$.
18. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance. Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{ERRJ}): $V_{ERRJ} = R_S * I_{OFF}$ where I_{OFF} is a function of operating temperature. Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the filtering capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance
19. For a maximum sampling error of the input voltage ← 1LSB, then the external filter capacitor, $C_f \geq 1024 * C_{SAMP}$. The value of C_{SAMP} in the new design may be reduced.

E.17 QSMCM Electrical Characteristics

Table E-17 QSPI Timing

($T_A = T_L$ to T_H, 50 pF load on all QSPI pins unless otherwise noted)

Num	Function	Symbol	Min	Max	Unit
108	Operating Frequency[1] Master Slave	f_{OP}	DC DC	$f_{SYS}/4$ $f_{SYS}/4$	Hz Hz
109	Cycle Time Master Slave	t_{QCYC}	4*TC 4*TC	510 * TC[2] —	ns ns
110	Enable Lead Time Master Slave	t_{LEAD}	2*TC 2*TC	128 * TC —	ns ns
111	Enable Lag Time Master Slave	t_{LAG}	— 2*TC	SCK/2 —	ns ns
112	Clock (SCK) High or Low Time Master Slave[3]	t_{SW}	2*TC– 60 2*TC– n	255 * TC —	ns ns
113	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{TD}	17*TC 13*TC	8192 * TC -	ns ns
114	Data Setup Time (Inputs) Master Slave	t_{SU}	30 20	- -	ns ns
115	Data Hold Time (Inputs) Master Slave	t_{HI}	0 20	- -	ns ns
116	Slave Access Time	t_A	—	TC	ns
117	Slave MISO Disable Time	t_{DIS}	—	2 * TC	ns
118	Data Valid (after SCK Edge) Master Slave	t_V	— —	50 50	ns ns
119	Data Hold Time (Outputs) Master Slave	t_{HO}	0 0	— —	ns ns

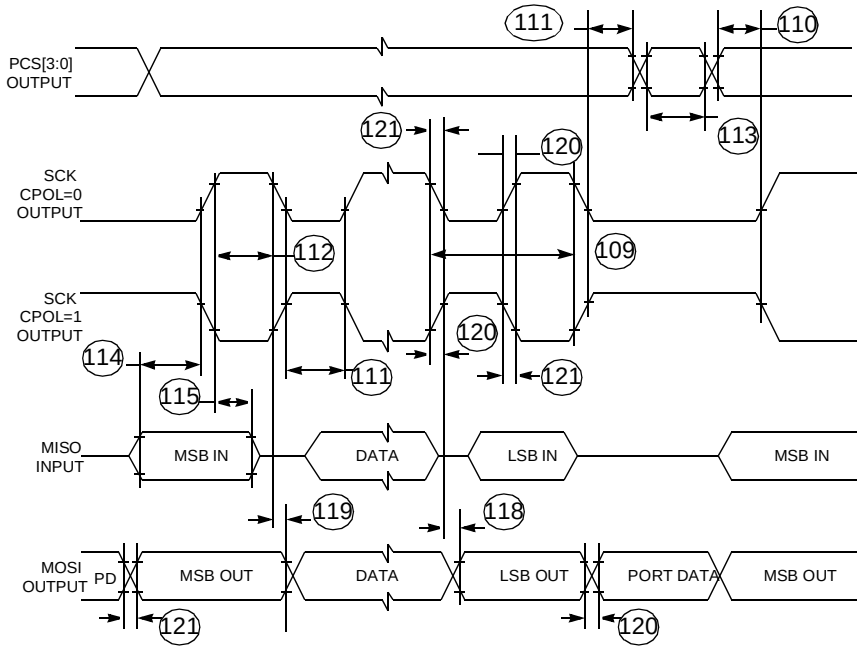
Table E-17 QSPI Timing (Continued)

($T_A = T_L$ to T_H, 50 pF load on all QSPI pins unless otherwise noted)

Num	Function	Symbol	Min	Max	Unit
120	Rise Time Input Output – up to 50 pF, SLRC bit of PDMCR = “0” up to 50 pF, SLRC bit of PDMCR = “1” up to 200 pF, SLRC bit of PDMCR = “1”	 t_{rI} t_{RO}	— — — —	 1 200 8 21	 µs ns ns ns
121	Fall Time Input Output – up to 50 pF, SLRC bit of PDMCR = “0” up to 50 pF, SLRC bit of PDMCR = “1” up to 200 pF, SLRC bit of PDMCR = “1”	 t_{FI} t_{FO}	— — — —	 1 200 8 21	 µs ns ns ns

NOTES:

1. All AC timing is tested to the 3-V levels outlined in **E.6 DC Electrical Characteristics**
2. TC is defined to be the clock period.
3. For high time, n = External SCK rise time; for low time, n = External SCK fall time.



QSPI MAST CPHA0

Figure E-41 QSPI Timing – Master, CPHA = 0

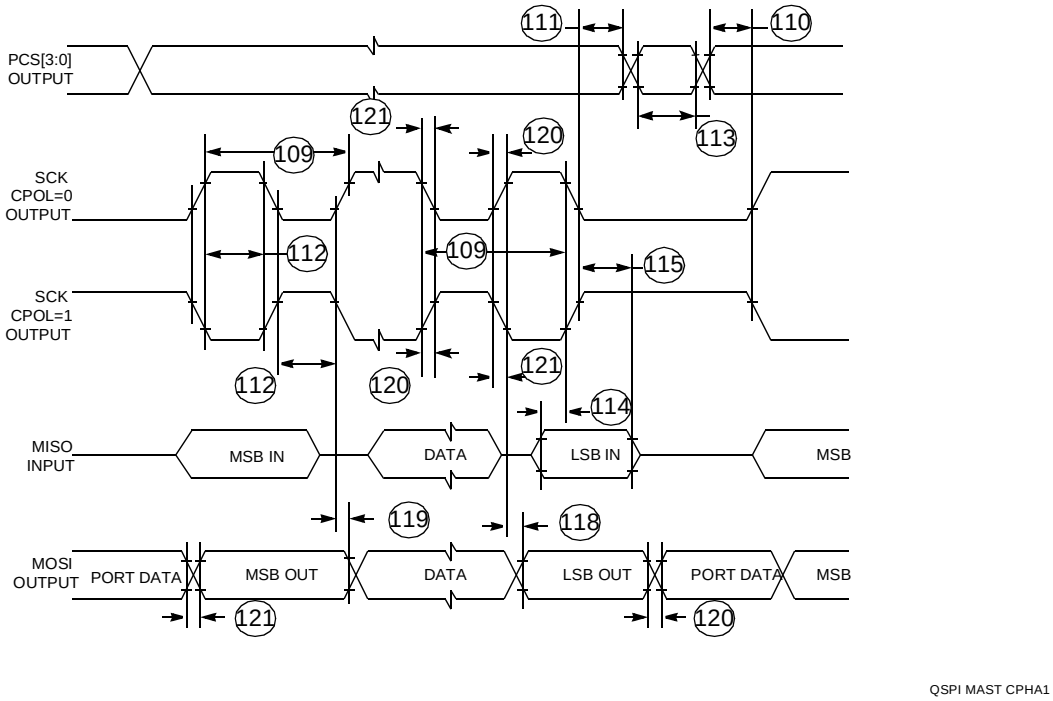


Figure E-42 QSPI Timing – Master, CPHA = 1

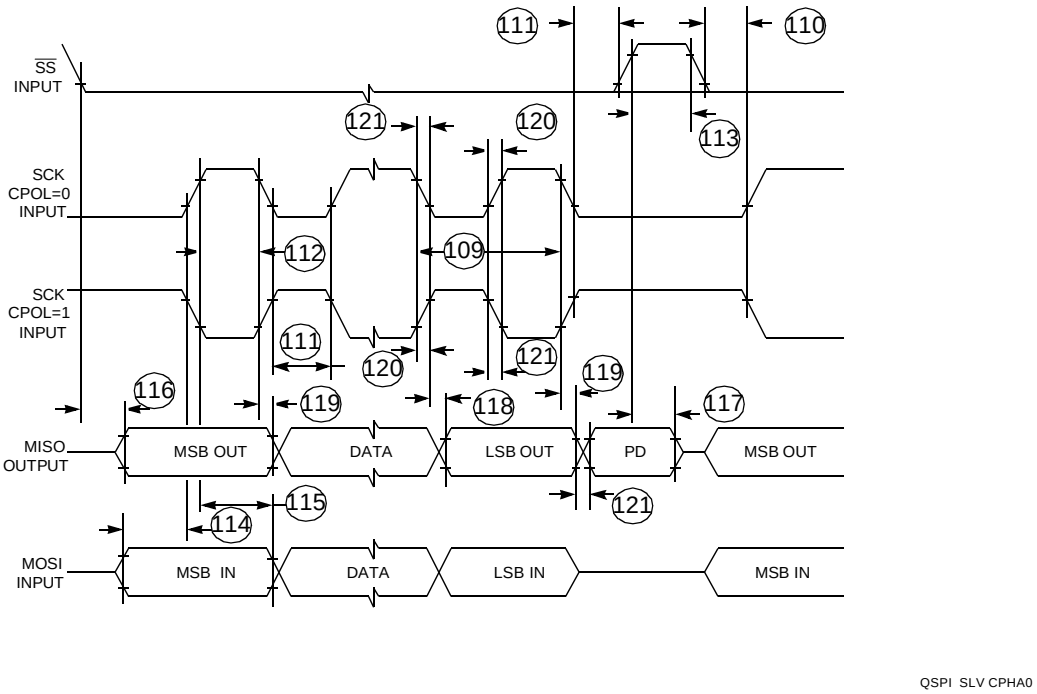


Figure E-43 QSPI Timing – Slave, CPHA = 0

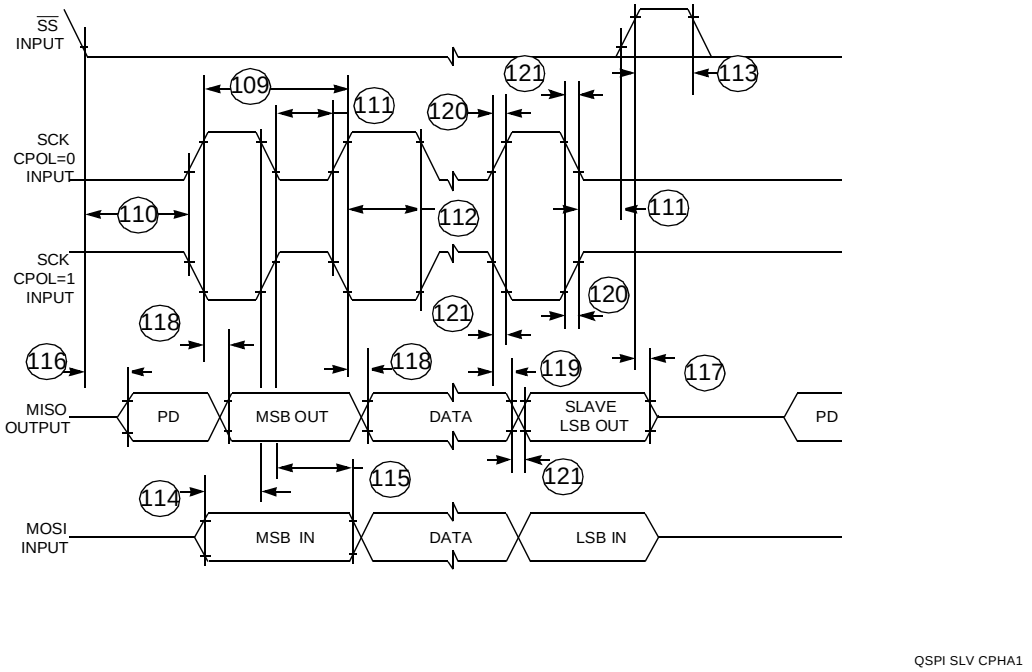


Figure E-44 QSPI Timing – Slave, CPHA = 1

E.18 GPIO Electrical Characteristics

Table E-18 GPIO Timing

($T_A = T_L$ to T_H)

GPIO applies to all pins used as GPIO: SGPIOA[8:31], SQPIOD[0:31], SGPIOC[0:7], QGPIO[0:6], QGPO[1:2], MPIO[0:15], A_PQA[0:7], B_PQA[0:7], A_PQB[0:7], B_PQB[0:7]

Num	Rating	Symbol	Min	Max	Unit
122	Rise Time Input Output[1] (SLR0 of PDMCR = 0), 50 pF to 200 pF Load Output[1] (SLR0 of PDMCR = 0), 20 nF Load[2] Output (SLR0 of PDMCR = 1), 50 pF Load	 t_{RI} t_{RO}	 — 90 2000 3	 1 600 7500 25	 µs ns ns ns
123	Fall Time Input Output[1] (SLR0 of PDMCR = 0), 50 pF to 200 pF Load Output[1] (SLR0 of PDMCR = 0), 20 nF Load[2] Output (SLR0 of PDMCR = 1), 50 pF Load	 t_{FI} t_{FO}	 — 90 2000	 1 600 7500 25	 µs ns ns ns

NOTES:
1. This parameter is tested during initial characterization and is not tested in production.
2. Care should be taken to insure that the total power dissipation of the device remain below the absolute maximum rating under this condition. See **Table E-1**.

E.19 TPU3 Electrical Characteristics

Table E-19 TPU3 Timing

($T_A = T_L$ to T_H)

Num	Rating	Symbol	Min	Max	Unit
124	Slew Rate of TPU Output Channel Valid[1,2] (SLR0 of PDMCR = 0, 50 pF to 200 pF load) (SLR0 of PDMCR = 0, 20 nF load)[3] (SLR0 of PDMCR = 1, 50 pF load)	t_{CHTOV}	92 2000 3	650 7550 25	ns ns ns
125	CLKOUT High to TPU Output Channel Hold	t_{CHTOH}	0	15	ns
126	TPU Input Channel Pulse Width[4]	t_{TIPW}	4	—	t_{cyc}

NOTES:
1. AC timing is shown with respect to 10% V_{DD} & 90% V_{DD} levels.
2. Timing not valid for external T2CLK input.
3. Care should be taken to insure that the total power dissipation of the device remain below the absolute maximum rating under this condition. See **Table E-1**.
4. t_{CYC} is defined as the CLKOUT Period.

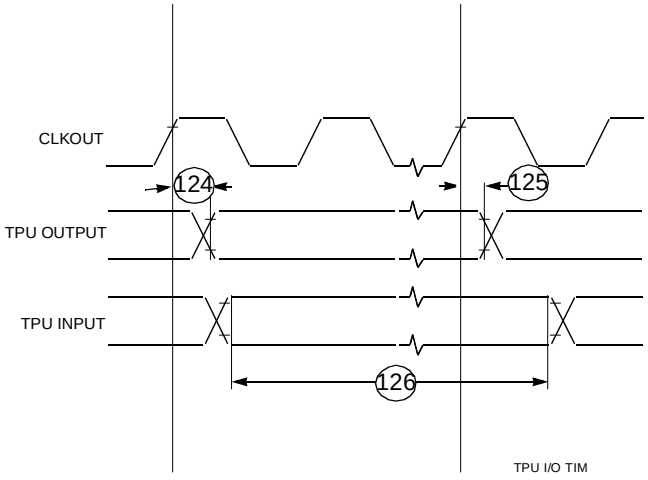


Figure E-45 TPU3 Timing

E.20 TouCAN Electrical Characteristics

Table E-20 TouCAN Timing[1]

($T_A = T_L$ to T_H)

Num	Rating	Symbol	Min	Max	Unit
127	CNTX0 (Delay from ICLOCK)	t_{CNTX0}		19	ns
128	CNRX0 (Set-Up to ICLOCK rise)	t_{CNRX0}		0	ns
127	Rise Time Input Output – 50 pF load, SLRC bit of PDMCR = “0” 200 pF load, SLRC bit of PDMCR = “0” 50 pF, SLRC bit of PDMCR = “1”	 t_{RI} t_{RO}		 1 50 100 25	 µs ns ns ns
128	Fall Time Input Output– 50 pF load, SLRC bit of PDMCR = “0” 200 pF load, SLRC bit of PDMCR = “0” 50 pF, SLRC bit of PDMCR = “1”	 t_{FI} t_{FO}		 1 50 100 25	 µs ns ns ns
	Serial Pins (Maximum frequency)	t_F	1	—	MHz

NOTES:
1. AC timing is shown is tested to the 3-V levels outlined in **E.6 DC Electrical Characteristics**.

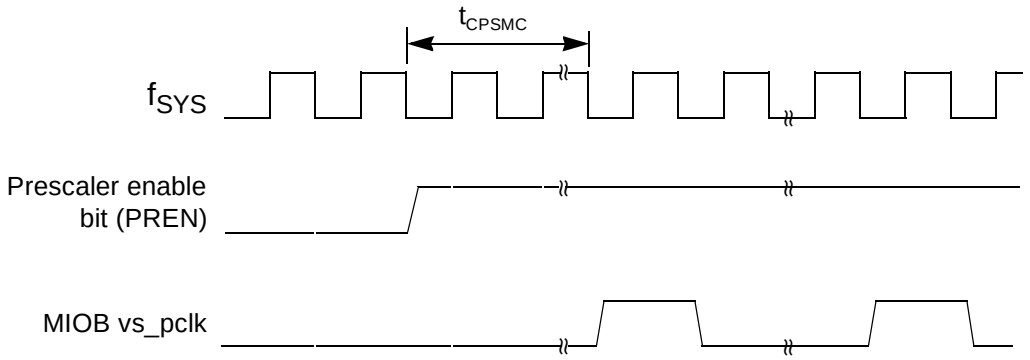
E.21 MIOS Timing Characteristics

All MIOS output pins are slew rate controlled. Slew rate control circuitry adds 90 ns as minimum to the output timing and 650 ns as a maximum. This slew rate is from 10% V_{DD} to 90% V_{DD}, an additional 100 ns should be added for total 0 to V_{DD} slew rate.

Table E-21 MCPSM Timing Characteristics

Characteristic	Symbol	Delay	Unit
MCPSM enable to vs_pclk pulse[1]	t_{CPSMC}	(MCPSMSCR_PSL[3:0]) -1	System Clock Cycles

NOTES:
1. The MCPSM clock prescaler value (MCPSMSCR_PSL[3:0]) should be written to the MCPSMSCR (MCPSM Status/Control Register) before rewriting the MCPSMSCR to set the enable bit (MCPSMSCR_PREN). If this is not done the prescaler will start with the old value in the MCPSMSCR_PSL[3:0] before reloading the new value into the counter.
   Note: after reset MCPSMSCR_PSL[3:0] is set to 0b0000.
   Note: vs_pclk is the MIOS prescaler clock which is distributed to all the counter (e.g., MPWMSM and MMCSM) submodules.



Note 1: f_{SYS} is the internal system clock for the IMB3 bus.
Note 2: The numbers associated with the f_{SYS} ticks refer to the IMB3 internal state.
Note 3: vs_pclk is the MIOS prescaler clock which is distributed around the MIOS to counter modules such as the MMCSM and MPWMSM.

Figure E-46 MCPSM Enable to vs_pclk Pulse Timing Diagram

E.21.1 MPWMSM Timing Characteristics

NOTE

All delays are in system clock periods.

Table E-22 MPWMSM Timing Characteristics

Characteristic	Symbol	Min	Max
PWMSM output resolution	t_{PWMR}	— [1]	2.0 [2]
PWM output pulse[3]	t_{PWMO}	2.0	—
MPWMI input pin to MPWMSCR_PIN status set	t_{PIN}	1	2
CPSM enable to output set[4]	t_{PWMP}	(MPWMPERR - MPWMPULR + 1) * (256 - MPWMSCR_CP) * MCPSMSCR_PSL + 1	
MPWMSM Enable to output set (MIN)[5]	t_{PWME}	(MPWMPERR - MPWMPULR) * (256 - MPWMSCR_CP) * MCPSMSCR_PSL + 3 + (255 - MPWMSCR_CP) * MCPSMSCR_PSL[6]	
MPWMSM Enable to output set (MAX)[5]	t_{PWME}	t_{PWME}(MIN) + MCPSMSCR_PSL - 1[6]	
Interrupt Flag to output pin reset (period start)[7]	t_{FLGP}	(256 - MPWMSCR_CP) * MCPSMSCR_PSL - 1[6]	

NOTES:
1. Minimum output resolution depends on MPWMSM and MCPSM prescaler settings.
2. Maximum resolution is obtained by setting CPSMPSL[3:0] =0x2 and MPWMSCR_CP[7:0] =0xFF.
3. Excluding the case where the output is always “0”.
4. With MPWMSM enabled before enabling the MCPSM. Please also see note *1 on the MCPSM timing information.
5. The exact timing from MPWMSM enable to the pin being set depends on the timing of the register write and the MCPSM vs_pclk.
6. When MCPSMSCR_PSL = 0x0000, this gives a prescale value of 16 and it is 16 which should be used in these calculations. When MCPSMSCR_PSL = 0x0001, the CPSM is inactive.
7. Note: the interrupt is set before the output pin is reset (Signifying the start of a new period).

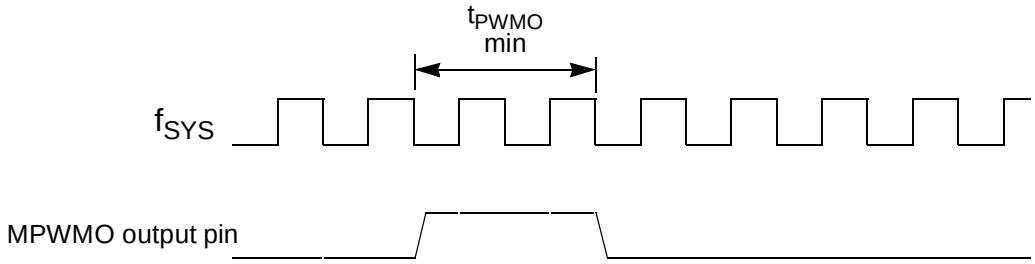


Figure E-47 MPWMSM Minimum Output Pulse Example Timing Diagram

NOTE

f_{SYS} is the internal system clock for the IMB3 bus.

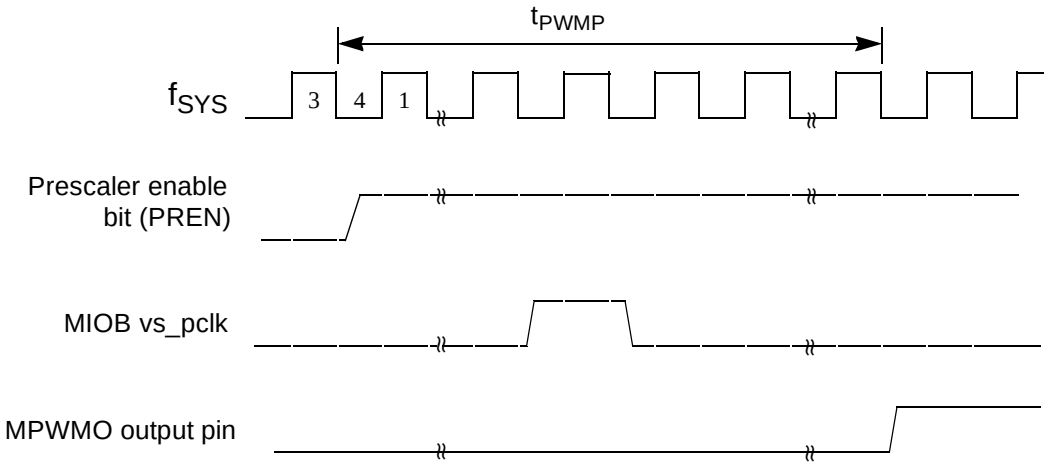


Figure E-48 MCPSM Enable To MPWMO Output Pin Rising Edge Timing Diagram

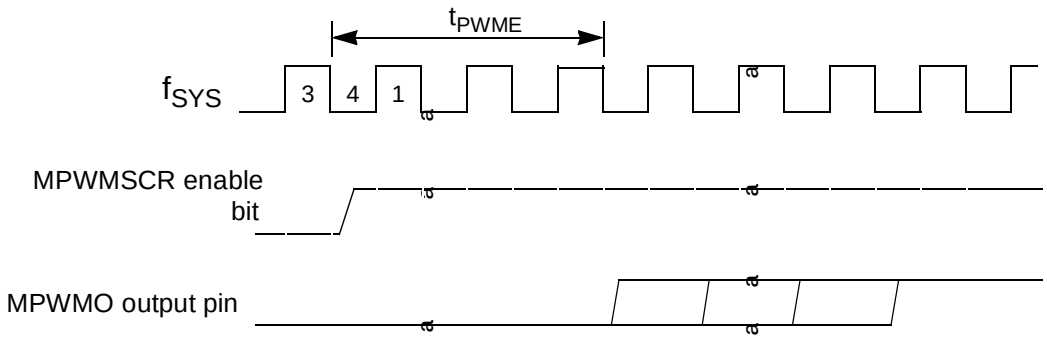


Figure E-49 MPWMSM Enable To MPWMO Output Pin Rising Edge Timing Diagram

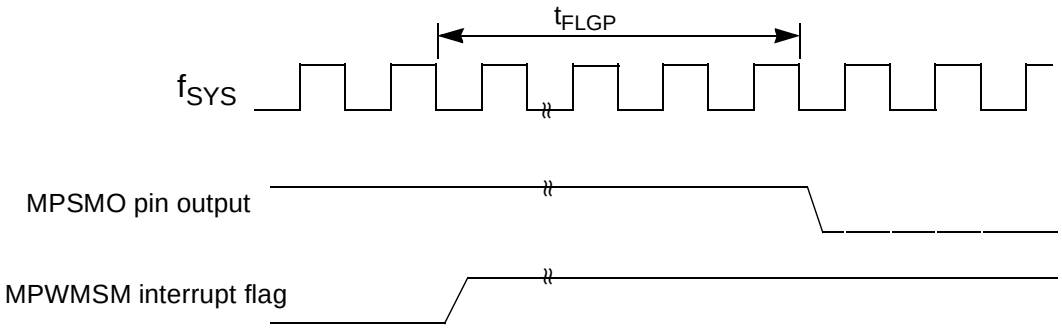


Figure E-50 MPWMSM Interrupt Flag To MPWMO Output Pin Falling Edge Timing Diagram

E.21.2 MMCSM Timing Characteristics

Table E-23 MMCSM Timing Characteristics

Note: All delays are in system clock periods.

Characteristic	Symbol	Min	Max
MMCSM input pin period	t_{PPER}	4	—
MMCSM pin low time	t_{PLO}	2	—
MMCSM pin high time	t_{PHI}	2	—
clock pin to counter bus increment.	t_{PCCB}	1	2
load pin to new counter bus value	t_{PLCB}	1	2
clock pin to PINC delay	t_{PINC}	1	2
load pin to PINL delay	t_{PINL}	1	2
Counter bus resolution	t_{CBR}	—[1]	2 [2]
Counter bus overflow reload to interrupt flag	t_{CBFLG}	1	
MCPSM enable to counter bus increment.	t_{MCMP}	(256 - MMCSMSCR_CP) * MCPSMSCR_PSL + 2	
MMCSM Enable to counter bus increment (MIN) [3]	t_{MCME}	4 + MCPSMSCR_PSL* (255 - MMCSMSCR_CP)[3]	
MMCSM Enable to counter bus increment (MAX)[3]	t_{MCME}	4 + MCPSMSCR_PSL * (255 - MMCSMSCR_CP) + (MCPSMSCR_PSL - 1)[3]	

NOTES:
1. Minimum output resolution depends on MMCSM and MCPSM prescaler settings.
2. Maximum resolution is obtained by setting CPSMPSL[3:0] =0x2 and MMCSMSCR_CP[7:0] =0xFF.
3. The exact timing from MMCSM enable to the pin being set depends on the timing of the MMCSMSCR register write and the MCPSM vs_pclk. The MMCSM enable is taken to mean the MMCSMSCR_CLS[1:0] being written to 2‘b11.

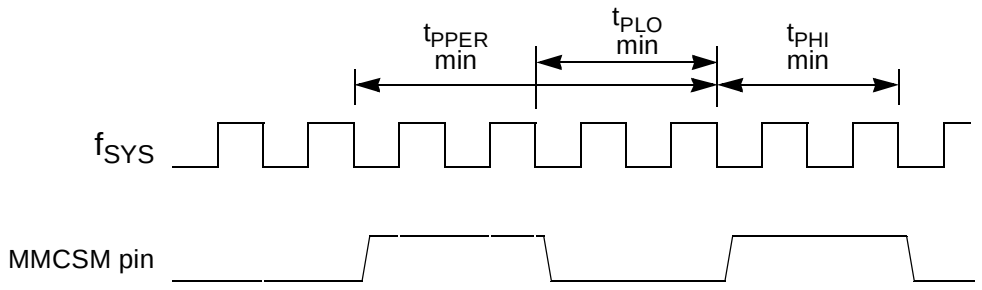


Figure E-51 MMCSM Minimum Input Pin (Either Load Or Clock) Timing Diagram

NOTE

f_{SYS} is the internal system clock for the IMB3 bus.

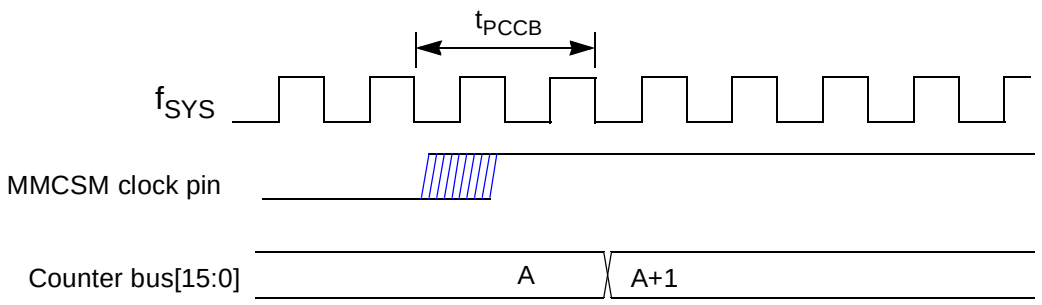


Figure E-52 MMCSM Clock Pin To Counter Bus Increment Timing Diagram

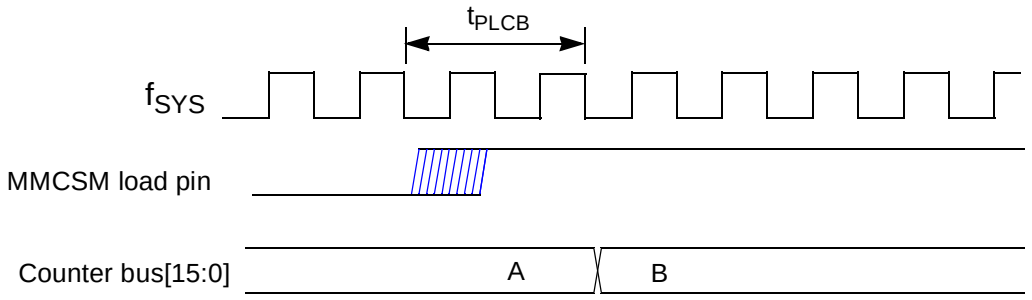


Figure E-53 MMCSM Load Pin To Counter Bus Reload Timing Diagram

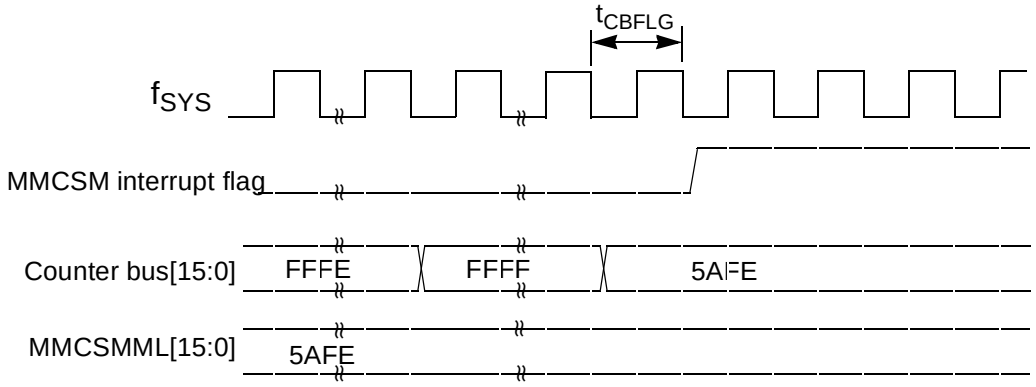


Table E-24 MMCSM Counter Bus Reload To Interrupt Flag Setting Timing Diagram

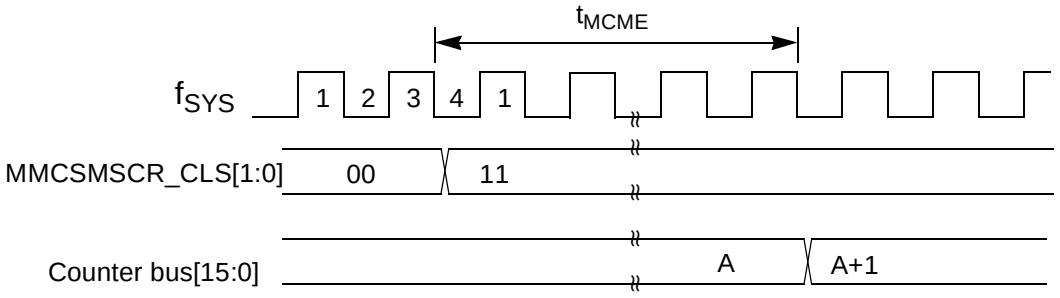


Figure E-54 MMCSM Prescaler Clock Select To Counter Bus Increment Timing Diagram

E.21.3 MDASM Timing Characteristics

Table E-25 MDASM Timing Characteristics

Note: All delays are in system clock periods.

Characteristics	Symbol	Min	Max
Input Modes: (IPWM, IPM, IC, DIS)			
MDASM input pin period	t_{PPER}	4	—
MDASM pin low time	t_{PLO}	2	—
MDASM pin high time	t_{PHI}	2	—
Input capture resolution	t_{CAPR}	—	2
Input pin to Counter Bus capture delay	t_{PCAP}	1	3[1]
Input pin to interrupt flag delay	t_{PFLG}	2	3
Input pin to PIN delay	t_{PIN}	1	2
Counter bus resolution	t_{CBR}	—	2[2]
Output Modes: (OC, OPWM)			
Output pulse width[3]	t_{PULW}	2	—
Compare resolution[3]	t_{COMR}	—	2[2]
Counter Bus to pin change	t_{CBP}	3	
Counter Bus to interrupt flag set.	t_{CBFLG}	3	

NOTES:

1. If the counter bus capture occurs when the counter bus is changing then the capture is delayed one cycle. In situations where the counter bus is stable when the input capture occurs the t_{PCAP} has a maximum delay of two cycles (the one-cycle uncertainty is due to the synchronizer).
2. Maximum resolution is obtained by setting CPSMPSL[3:0] =0x2 and MDASMSCR_CP[7:0] =0xFF.
3. Maximum output resolution and pulse width depends on counter (e.g., MMCSM) and MCPSM prescaler settings.

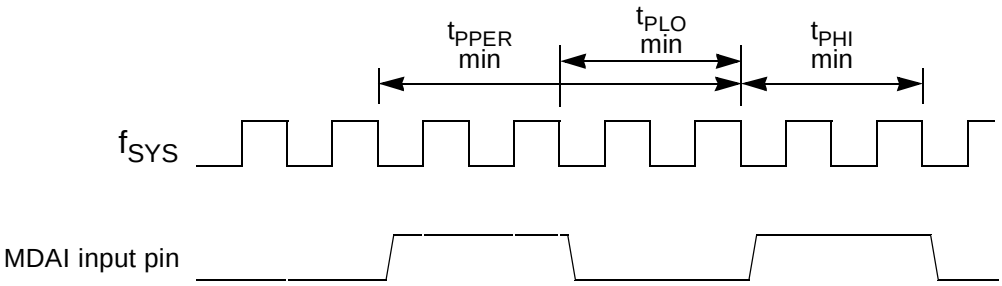


Figure E-55 MDASM Minimum Input Pin Timing Diagram

NOTE

f_{SYS} is the internal system clock for the IMB3 bus.

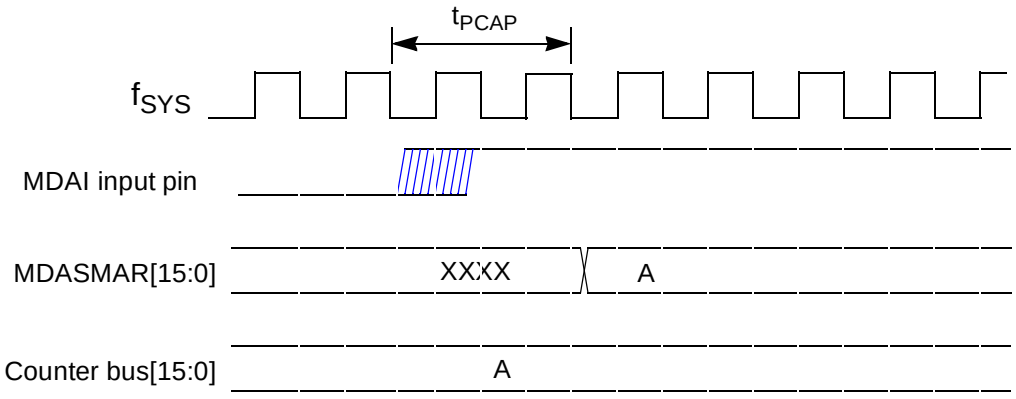


Figure E-56 MDASM Input Pin To Counter Bus Capture Timing Diagram

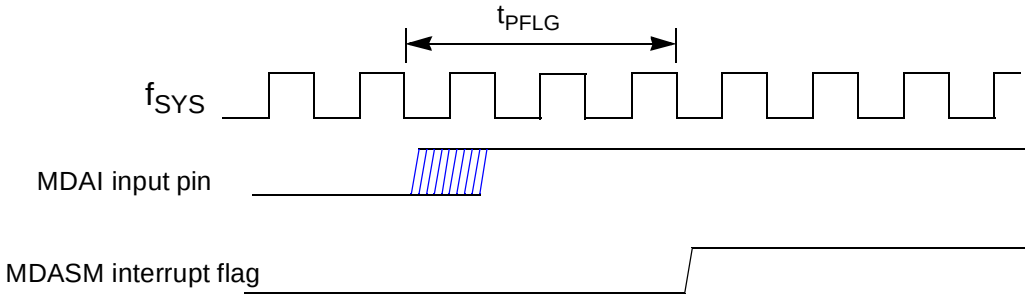


Figure E-57 MDASM Input Pin to MDASM Interrupt Flag Timing Diagram

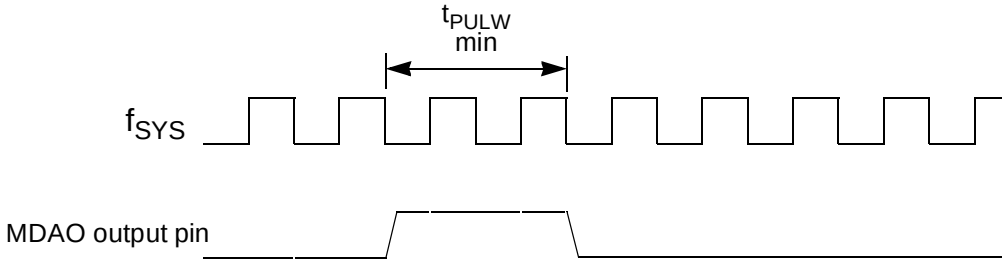


Figure E-58 MDASM Minimum Output Pulse Width Timing Diagram

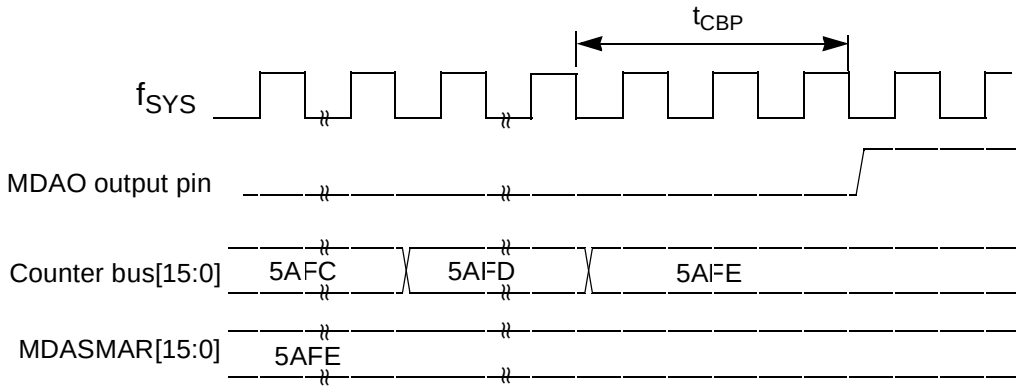


Figure E-59 Counter Bus to MDASM Output Pin Change Timing Diagram

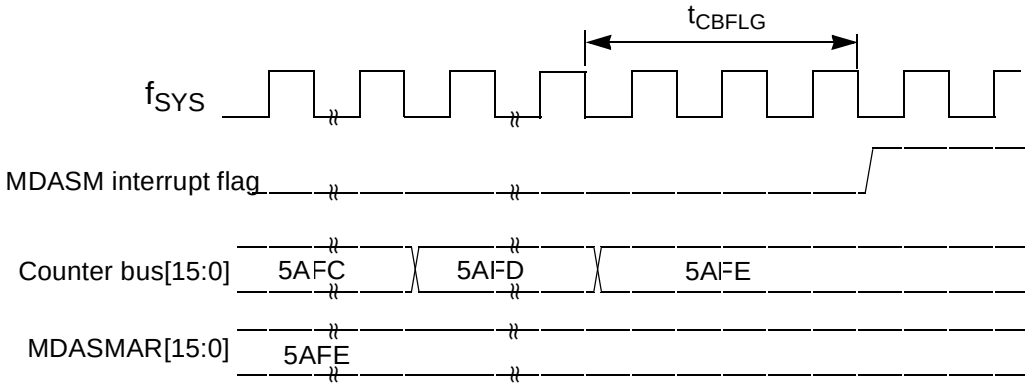


Figure E-60 Counter Bus to MDASM Interrupt Flag Setting Timing Diagram

E.22 MSASM Timing Characteristics

Table E-26 MSASM Timing Characteristics

Note: All delays are in system clock periods.

Characteristics	Symbol	Min	Max
Input Modes: (IC, IOP)			
MSASM input pin period	t_{PPER}	4[1]	—
MSASM pin low time	t_{PLO}	2	—
MSASM pin high time	t_{PHI}	2	—
Input capture resolution	t_{CAPR}	—	2
Input pin to Counter Bus capture delay	t_{PCAP}	1	3 [2]
Input pin to interrupt flag delay	t_{PFLG}	3	5
Input pin to PIN delay	t_{PIN}	1	2
Counter bus resolution	t_{CBR}	—	2 [3]
Output Modes: (OC, OCT)			
Output pulse width[4]	t_{PULW}	2	—
Compare resolution[4]	t_{COMR}	—	2 [4]
Counter Bus to pin change	t_{CBP}	2	
Counter Bus to interrupt flag set.	t_{CBFLG}	3	

NOTES:

1. The minimum input pin period depends on how fast the CPU services the MSASM. The MSASM can respond to a pin changing every four clock periods but the CPU will miss some of the captured values.
2. If the counter bus capture occurs when the counter bus is changing then the capture is delayed one cycle. In situations where the Counter Bus is stable when the input capture occurs the t_{PCAP} has a maximum delay of two cycles (the one-cycle uncertainty is due to the synchronizer).
3. Maximum resolution is obtained by setting CPSMPSL[3:0] =%2 and MSASMSCR_CP[7:0] =%FF.
4. Maximum output resolution and pulse width depends on counter (e.g., MMCSM) and MCPSM prescaler settings.

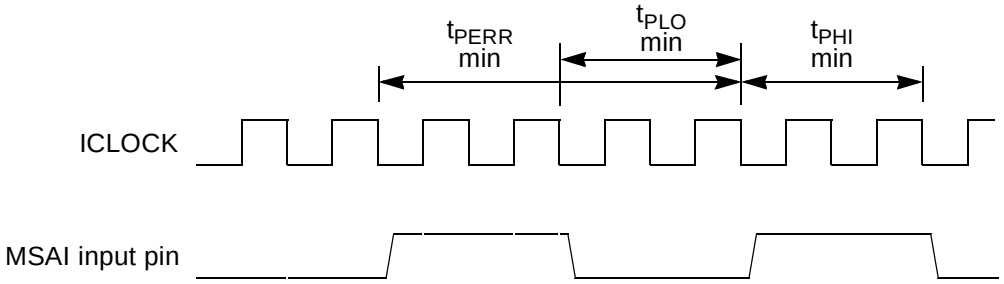


Figure E-61 MSASM Minimum Input Pin Timing Diagram

NOTE

ICLOCK is the internal system clock for the IMB3 bus.

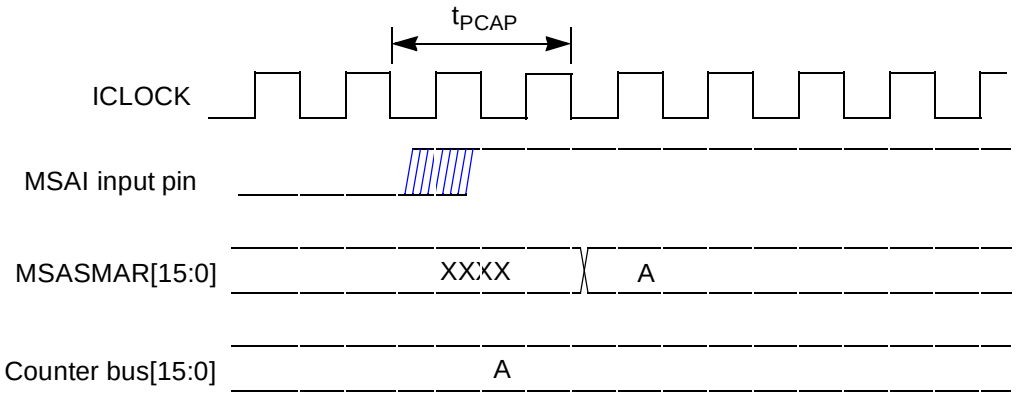


Figure E-62 MSASM Input Pin To Counter Bus Capture Timing Diagram

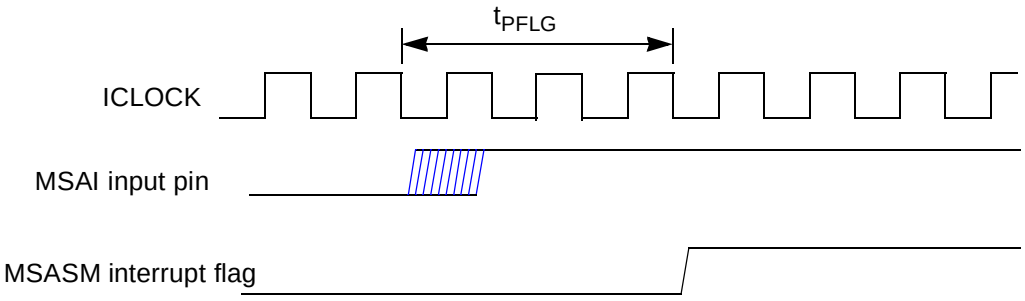


Figure E-63 MSASM Input Pin to MSASM Interrupt Flag Timing Diagram

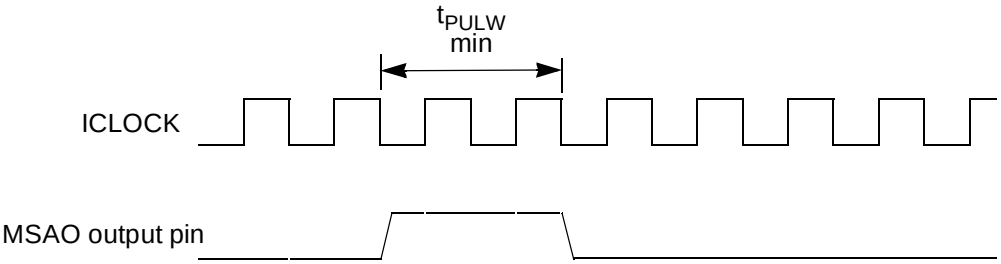


Figure E-64 MSASM Minimum Output Pulse Width Timing Diagram

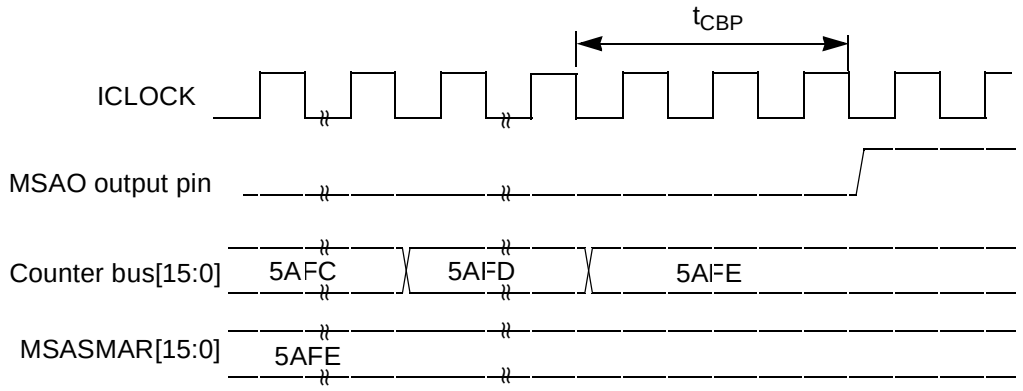


Figure E-65 Counter Bus to MSASM Output Pin Change Timing Diagram

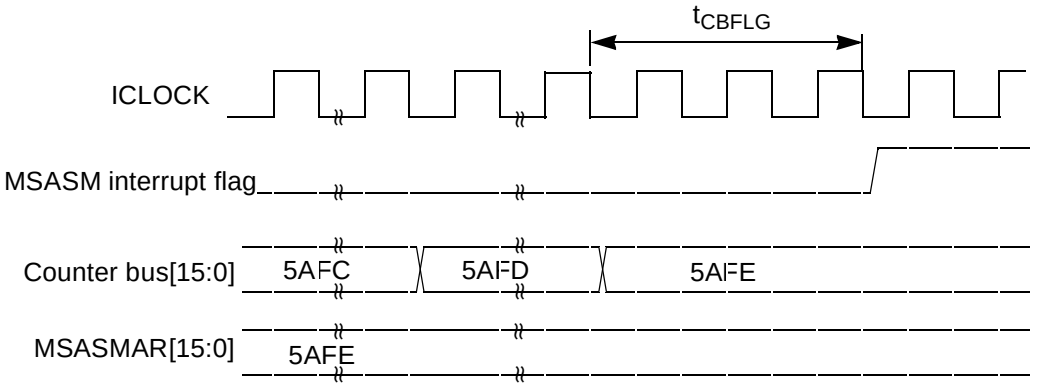


Figure E-66 Counter Bus to MSASM Interrupt Flag Setting Timing Diagram

E.22.1 MPIOSM Timing Characteristics

Table E-27 MPIOSM Timing Characteristics

Note: All delays are in system clock periods.

Characteristic	Symbol	Min	Max
Input Mode			
MPIOSM input pin period	t_{PPER}	—[1]	—
MPIOSM pin low time	t_{PLO}	—[1]	—
MPIOSM pin high time	t_{PHI}	—[1]	—
Input pin to MPIOSM_DR delay	t_{PDR}	0	1
Output mode			
Output pulse width [2]	t_{PULW}	—[2]	—

NOTES:

1. The minimum input pin period, pin low and pin high times depend on the rate at which the MPIOSM_DR register is polled.
2. The minimum output pulse width depends on how quickly the CPU updates the value inside the MIOPSM_DR register.

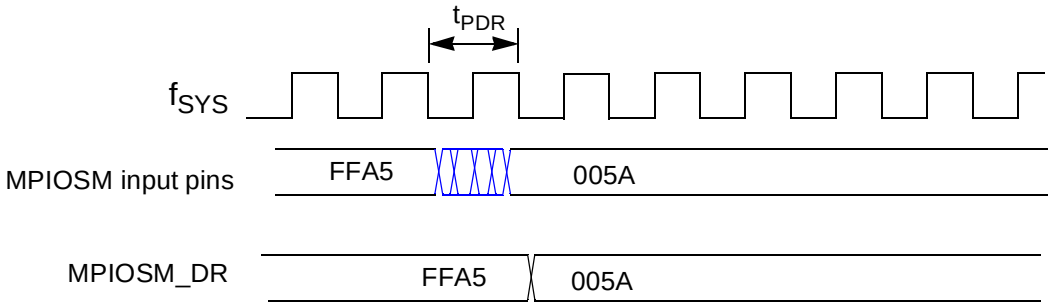


Figure E-67 MPIOSM input pin to MPIOSM_DR (Data Register) Timing Diagram

NOTE

f_{SYS} is the internal system clock for the IMB3 bus.

INDEX

–A–

A(0
31), 9-4
Accesses
clock requirements 13-55
ACKERR 16-31
Acknowledge error (ACKERR) 16-31
Address
-mark wakeup 14-59
Address space 13-12, 13-13
address type (AT0-AT3), 9-37
ALE 22-52
ALEE 22-54
alignment exception, 3-46
ALU–BFU 3-5
AN 13-8
Analog
front-end multiplexer 13-40
input
considerations 13-78
pins 13-76
power pins 13-72
reference pins 13-76
section contents 13-4
submodule block diagram 13-39
supply
filtering and grounding 13-74
pins 13-72
to digital converter operation 13-38
arbitration, 9-30
AT(0
3), 9-4
atomic update primitives, 3-42
atomic, 9-31

–B–

BAR 3-53
Base ID mask bits 16-30
Baud
clock 14-52
BB, 9-7
BDIP, 9-5
BE bit 3-21
Beginning of queue 2 (BQ2) 13-21
BG, 9-7
BI, 9-6
Binary
divider 13-52
-weighted capacitors 13-40
Bit stuff error (STUFFERR) 16-31
BITERR 16-31
BITS 14-18
Bits per transfer
enable (BITSE) 14-24
field (BITS) 14-18
BITSE 14-24, 14-39
Bit-time 14-51
BIU 13-4, 13-55
BIUSM
BIUTEST — BIUSM test configuration register 17-32, 17-33
selecting the time base bus 17-32, 17-33
BIUTEST — BIUSM test configuration register 17-32, 17-33
BKPT (TPU asserted) 18-15
BLC 18-14
BOFFINT 16-31
Boundary conditions 13-44
BPU 3-5
BQ2 13-21, 13-44
BR, 9-7
Branch
prediction 3-5
processing unit 3-5
trace enable 3-21
Branch latch control (BLC) 18-14
Branch processing unit 3-5
Break frame 14-52
Breakpoint
asserted flag (BKPT) 18-15
flag (PCBK) 18-15
Breakpoint counter A value and control register 22-50
Breakpoint counter B value and control register 22-51
BRKNOMSK 22-50
burst indicator (BURST), 9-36
burst inhibit (BI), 9-39
burst read cycle (illustration), 9-18
burst transfer, 9-15
burst write cycle (illustration), 9-23
BURST, 9-4
Bus
monitor 6-18
off interrupt
(BOFFINT) 16-31
bus busy (BB), 9-32
bus exception control cycles, 9-44
bus grant (BG), 9-32
bus interface
bus control signals, 9-2
bus operation
address transfer phase related signals, 9-35
arbitration phase, 9-30

basic transfer protocol, 9-8
burst mechanism, 9-16
burst transfer, 9-15
bus exception control cycles, 9-44
single beat transfer
single beat read flow, 9-8
single beat write flow, 9-8, 9-11
single beat transfer, 9-8
storage reservation, 9-41
termination signals, 9-39
bus operations, 9-7
bus transfer signals, 9-1
features, 9-1
signal descriptions, 9-3
Bus interface unit (BIU) 13-4
components 13-55
bus request (BR), 9-31
bus signals (illustration), 9-3
BUSY 16-5, 16-15
BYP 13-33
BYTES field 3-18

–C–

C bit 3-14
CA bit 3-18
cache control instructions, 3-43
CAN2.0B
system 16-3
CANCTRL0 16-25
CANCTRL1 16-26
CANCTRL2 16-28
CANICR 16-24
Carry 3-18
CCL 18-14
CCW 13-4, 13-30
C_F 13-80
CF1 13-23
CF2 13-24
CFSR 18-16
CGBMSK 22-48
CH 18-16, 18-19, 18-20
CHAN 13-34
CHANNEL 18-17
Channel
assignments (multiplexed) 13-35
conditions latch (CCL) 18-14
interrupt
enable
/disable field (CH) 18-16
request level (CIRL) 18-16
status (CH) 18-20
number (CHAN) 13-34
orthogonality 18-3
priority registers 18-19
register breakpoint flag (CHBK) 18-15
Charge sharing 13-81
CHBK 18-15
CHBMSK 22-48
checkstop state, 3-45
CHSTP bit 22-52
CHSTPE 22-53
CIE1 13-18
CIE2 13-20
CIER 18-16, 18-20
CIRL 18-16
CISR 18-8, 18-20
class, instruction, 3-39
CLKOUT to TA, BI assertion (when driven by the Memory Controller) E-22
CLKOUT, 9-7
CLKS 18-14
Clock
block diagram 13-53
frequency 13-53
generation 13-52
phase (CPHA) 14-18
polarity (CPOL) 14-18
requirements 13-55
CMPA–CMPD 22-44
CMPE–CMPF 22-45
CMPG–CMPH 22-45
CNRX/TX pins 16-2
CNTC 22-51
CNTV 22-51
Code 16-5
Coherency 13-50, 13-55, 18-4
COMM D-18
Command
RAM 14-23
Command word pointer (CWP) 13-27, 13-30
Comparator 13-41
Comparator A–D value registers 22-44
Comparator E–F value registers 22-45
Comparator G–H value registers 22-45
Compare instructions 3-17
Compare size 22-48
Compare type 22-46, 22-48
Completed queue pointer (CPTQP) 14-22
Condition register 3-15, 3-17
CONT 14-24
contention, 9-36
Continue (CONT) 14-24
Continuous transfer mode 14-16
Control registers
0 (QACR0) 13-9, 13-16
1 (QACR1) 13-9
2 (QACR2) 13-9
QADC64E control register 1 (QACR1) 13-18
QADC64E control register 2 (QACR2) 13-20
controlling termination of a bus cycle for a bus error, 9-44
Conversion
command word table (CCW) 13-4, 13-30, 13-41
cycle times 13-39
stages 13-32
Count register 3-19
COUNTA 22-50
COUNTB 22-51
CPHA 14-18, 14-35
CPOL 14-18, 14-35
CPR 18-19

CPTQP 14-22, 14-25
CPU
    wait states 13-55
CR 3-5, 3-15, 3-19
    and compare instructions 3-17
CR, 9-5
CR0 field 3-16
CR1 field 3-16
CRCERR 16-31
CRWE 22-48
CRWF 22-48
CSG 22-48
CSH 22-48
CTA 22-46
CTB 22-46
CTC 22-46
CTD 22-46
CTE 22-48
CTF 22-48
CTG 22-48
CTH 22-48
CTR 3-5
CWP 13-27, 13-30
Cyclic redundancy check error (CRCERR) 16-31

–D–

D(0
    31), 9-6
DAC 13-4
DAE/source instruction service register 3-22
DAR 3-22, 3-46, 3-52, 3-53
DAR, 3-46, 3-52
Data
    field for RX/TX frames (TOUCAN) 16-5
    frame 14-51
Data address register 3-22
data storage interrupt, 3-46
DCNR 18-21
DDRQA 13-15
DDRQS 14-10, 14-34, 14-38
Debug enable register 22-53
debug mode disable, 3-45
DEC 3-23
DECE 22-52
DECEE 22-54
Decrementer
    register 3-23
Delay
    after transfer (DT) 14-24, 14-36
    before SCK (DSCKL) 14-19
DER 22-53
Development Port
    trap enable selection 22-47
Digital
    control section
        contents 13-4, ??–13-38, 13-41–??
    input/output port (PQA) 13-71
    to analog converter (DAC) 13-4, 13-40
DIO D-38
Disable TPU2 pins field (DTPU) 18-22
Disabled mode 13-46
Discrete input/output (DIO) D-38
DIV2 18-21
DIV8 clock 18-7
Divide by two control field (DIV2) 18-21
DIW0EN 22-47
DIW1EN 22-47
DIW2EN 22-47
DIW3EN 22-47
DLW0EN 22-50
DLW1EN 22-50
Double
    -buffered 14-53, 14-57
DPI 22-53
DPTRAM 19-5
DSCK 14-24
DSCKL 14-19
DSCR 18-13
DSISR 3-22, 3-46, 3-52, 3-53
DSSR 18-15
DT 14-24
DTL 14-19
DTPU 18-22

–E–

EA 3-33
EBRK 22-53
ECR 22-51
EE bit 3-21, 3-26
Effective address 3-33
EID 3-26
EIE 3-26
eieio, 3-43
ELE bit 3-21
EMPTY 16-5
EMU 18-4, 18-12
Emulation
    control (EMU) 18-12
    support 18-4
Encoded
    one of three channel priority levels (CH) 18-20
    time function for each channel (CHANNEL) 18-17
    type of host service (CH) 18-19
Ending queue pointer (ENDQP) 14-20
End-of-
    frame (EOF) 16-16
End-of-queue condition 13-32
ENDQP 14-20, 14-25
Entry
    table bank select field (ETBANK) 18-22
EOF 16-16
EOQ 13-44
EP bit 3-21
ERRINT 16-32
Error
    conditions 14-58
    counters 16-9
    interrupt (ERRINT) 16-32
    resulting from leakage 13-80
ESTAT 16-30

ETBANK 18-22
ETRIG 13-72, 13-76
Event timing 18-3
Exception cause register 22-51
Exception prefix 3-21
Exceptions 3-34
    classes 3-34
    little endian mode 3-21
    ordered 3-34
    precise 3-35
    unordered 3-35
    vector table 3-35, 3-36
Execution units 3-4
Extended message format 16-1
    frames 16-5
External
    interrupt
        disable 3-26
        enable 3-26
    leakage 13-81
    trigger continuous-scan mode 13-50, 13-51
    trigger pins 13-72
    trigger single-scan mode 13-47, 13-48
External interrupt
    enable 3-21, 3-26
EXTI 22-52
EXTIE 22-53

–F–

Fast quadrature decode TPU function (FQD) D-29
Fault confinement state (FCS) 16-10, 16-31
FCS 16-10, 16-31
FE 14-50, 14-58
FE bits 3-21, 3-22
FE flag 3-14
features
    bus interface, 9-1
Fetch serialized 22-1
FEX bit 3-14
FG bit 3-14
FI bit 3-14
Final sample time 13-39
FL bit 3-14
Floating-point
    available 3-21
    condition code 3-14
    enabled exception summary 3-14
    equal or zero 3-14
    exception mode 3-21, 3-22
    exception summary 3-14
    fraction inexact 3-14
    fraction rounded 3-14
    greater than or positive 3-14
    inexact exception 3-14
        enable 3-15
    invalid operation exception
        enable 3-15
        for ×*0 3-14
        for ×/× 3-14
        for ×-× 3-14
        for 0/0 3-14
        for invalid compare 3-14
        for invalid integer convert 3-15
        for invalid square root 3-15
        for SNaN 3-14
        for software request 3-15
        summary 3-14
    less than or negative 3-14
    overflow exception 3-14
        enable 3-15
    registers 3-12
    result class descriptor 3-14
    result flags 3-14
    rounding control 3-15
    status and control register 3-12
    underflow exception 3-14
    unit 3-5
    unordered or NaN 3-14
    zero divide exception 3-14
        enable 3-15
floating-point unavailable interrupt, 3-47
FORMERR 16-31
FP bit 3-21
FPCC bit 3-14
FPRF field 3-14
FPRs 3-12
FPSCK 18-22
FPSCR 3-12
FPU 3-5
FPUVE 22-52
FPUVEE 22-54
F_{QCLK} 13-52
FQD D-29
FQM D-11
FR 3-14
Frame 14-51
    size 14-58
Frames
    overload 16-16
    remote 16-15
Framing error (FE) flag 14-50, 14-58
FREEZ ACK 16-16
FREEZE
    assertion response (FRZ)
        QSM 14-7
        TPU 18-14
Freeze
    enable (FRZ) 13-10
    mode 13-11
FREEZE (internal signal) 13-12, 13-33
Frequency measurement (FQM) D-11
    parameters D-13
FRZ 13-10, 13-11, 16-11, 18-14
FRZACK 16-11
FU bit 3-14
FULL 16-5
Function
    library for TPU 18-4
FX bit 3-14

–G–

General SPRs 3-25
General-purpose registers (GPRs) 3-12
Global registers 13-9

–H–

Hall effect decode (HALLD) D-20
HALLD D-20
HALT 14-21, 16-11
Halt
    acknowledge flag (HALTA) 14-22
    QSPI (HALT) 14-21
HALTA 14-22
HALTA and MODF Interrupt Enable (HMIE) 14-42
HALTA/MODF interrupt enable (HMIE) bit 14-21
Hang on T4 (HOT4) 18-14
HMIE 14-21
HOT4 18-14
HSQR 18-18
HSSR 18-18

–I–

IBRK 22-53
I-bus
    watchpoint programming 22-46
I-bus support
    control register 22-46
ICTRL 22-46
ID
    Extended (IDE) field 16-6
    HIGH field 16-6
    LOW field 16-6
IDE 16-6
Identifier (ID) 16-1
    bit field 16-6
IDLE 14-49, 14-58, 16-31
Idle
    CAN status (IDLE) 16-31
    frame 14-51
    -line
        detect type (ILT) 14-47
        detected (IDLE) 14-49, 14-58
        detection process 14-58
        interrupt enable (ILIE) 14-47, 14-58
        type (ILT) bit 14-58
IFLAG 16-33
Ignore first match 22-47
IIFM 22-47
ILIE 14-47, 14-58
illegal and reserved instructions, 3-39
ILSCI 14-10
ILT 14-47, 14-58
IMASK 16-32
IMB 13-52
implementation dependent software emulation interrupt, 3-49
implementation specific data TLB error interrupt, 3-51
implementation specific debug interrupt, 3-52
implementation specific instruction TLB error interrupt, 3-50
IMUL–IDIV 3-5
Information processing time (IPT) 16-9
Initial sample time 13-39
Input
    sample time (IST) 13-34
Instruction
    pipeline 3-36
    sequencer 3-3
    set summary 3-29
    timing 3-36
Instruction fetch
    show cycle control 22-1
instruction storage interrupt, 3-46
instructions
    cache control, 3-43
    storage control, 3-44
instructions, partially executed, 3-53
Integer exception register 3-17
Integer unit 3-5
Interchannel communication 18-4
Intermission 16-16
Interrupt
    register (QADCINT) 13-9, 13-13
Interrupt Level of SCI (ILSCI) 14-10
Interrupts
    TOUCAN 16-18
    TPU 18-5
interrupts, 3-44
Inter-transfer delay 14-15
Interval timer single-scan mode 13-48
invalid and preferred instructions, 3-39
IPT 16-9
IRQ 18-5
ISCTL 22-1
IST 13-34
isync, 3-43
IU 3-5
IW 22-46

–K–

KR/RETRY, 9-5

–L–

LBRK 22-52
LBUF 16-27
L-bus support
    control register 1 22-48
    control register 2 22-49
LCTRL1 22-48
LCTRL2 22-49
LE bit 3-22
Least significant bit (LSB) 13-41
Length of delay after transfer (DTL) 14-19
Link register 3-18
Little endian mode 3-22
Load/store unit 3-5, 3-6
Lock

/release/busy mechanism 16-15
Loop
mode
(LOOPS) 14-47
LOOPQ 14-21
LOOPS 14-47
Low power stop (LPSTOP)
QSM 14-6
Lowest buffer transmitted first (LBUF) 16-27
Low-power
stop mode enable (STOP)
TPU 18-12
LR 3-5, 3-18, 18-21
LSB 13-41
LSU 3-5, 3-6
LW0EN 22-49
LW0IA 22-49
LW0IADC 22-49
LW0LA 22-49
LW0LADC 22-49
LW0LD 22-49
LW0LDDC 22-49
LW1EN 22-49
LW1IA 22-49
LW1IADC 22-49
LW1LA 22-49
LW1LADC 22-50
LW1LD 22-50
LW1LDDC 22-50

–M–

M 14-47, 14-52
MA 13-16
Machine
check enable 3-21
state register 3-20
status save/restore register 0 3-24
status save/restore register 1 3-24
machine check interrupt, 3-45
Machine status save/restore register 1 3-24
Mask
examples for normal/extended messages 16-8
registers (RX) 16-7
Master
/slave mode select (MSTR) 14-18
master
external
arbitration phase, 9-30
MCE 22-52
MCEE 22-53
MCPWM D-22
ME bit 3-21
Message
buffer
address map 16-22
code for RX/TX buffers 16-5
deactivation 16-13
structure 16-3
format error (FORMERR) 16-31
MISO 14-34, 14-38
MMCSMCNT MMCSM up-counter register 17-42
MMCSMMML MMCSM modulus latch register 17-42
Mode
fault flag (MODF) 14-22, 14-27
select (M) 14-47
Mode Fault Flag (MODF) 14-42
Modes
disabled 13-46
freeze 13-11
reserved 13-46
scan. See Scan modes
stop 13-10
MODF 14-22, 14-27, 14-42
module configuration register (QADC64E) 13-10
Module configuration register (QADCMCR) 13-9
module configuration register (QADCMCR) 13-10
Modulus
counter 14-52
MOSI 14-34, 14-38
Most significant bit (MSB) 13-41
MQ1 13-19
MQ2 13-21
MSB 13-41
MSR 3-20, 3-46, 3-48, 3-49, 3-50, 3-51, 3-52, 3-53
MSR, 3-45
MSTR 14-18
Multichannel pulse-width modulation (MCPWM) D-22
parameters
slave channel A
non-inverted center aligned mode D-41
slave edge-aligned mode D-24
Multimaster operation 14-27
Multiphase motor commutation (COMM) D-18
parameters D-19
Multiple end-of-queue 13-44
Multiplexed
analog inputs 13-8
MUX 13-16

–N–

Negative stress 13-81
New
queue pointer value (NEWQP) 14-20
New input capture/transistion counter (NITC) D-16
parameters D-17
NEWQP 14-20, 14-25
NF 14-50, 14-58
NI bit 3-15
NITC D-16
Noise
error flag (NF) 14-50
errors 14-58
flag (NF) 14-58
Non-IEEE floating-point operation 3-15
nonoptional instructions, 3-39
Non-recoverable interrupt 3-26
NOT ACTIVE 16-5
Not ready (NOTRDY) 16-20
NOTRDY 16-16, 16-20
NRI 3-26

–O–

OC D-34
OE bit 3-15
OP0, 9-28
OP1, 9-28
OP2, 9-28
OP3, 9-28
Open drain drivers 13-72
operand placement (effects), 3-42
operand representation (illustration), 9-28
optional instructions, 3-39
OR 14-49
Ordered exceptions 3-34
Output compare (OC) D-34
OV (overflow) bit 3-18
Overload frames 16-16
OVERRUN 16-5
Overrun error (OR) 14-49
OX bit 3-14

–P–

P 13-33
Parity
  (PF) flag 14-58
  checking 14-53
  enable (PE) 14-47
  error (PF) bit 14-50
  errors 14-58
  type (PT) 14-47
  type (PT) bit 14-53
Pause (P) 13-33, 13-42
PCBK 18-15
PCS 14-24
  to SCK delay (DSCK) 14-24
PCS0/SS 14-39
PCS3-PCS0/SS 14-42
PE 14-47
Period
  /pulse-width accumulator (PPWA) D-32
Periodic interrupt
  timer 6-21
Periodic/interval timer 13-54
  continuous-scan mode 13-52
Peripheral
  chip-selects (PCS) 14-24, 14-37
Peripheral Chip-Select 3-0/Slave Select (PCS3-PCSO/SS) 14-42
PF 14-50, 14-58
PF1 13-24
PF2 13-25
Phase buffer segment 1/2 (PSEG1/2) bit field 16-28
phase-lock loop, 9-7
PIE1 13-18
PIE2 13-20
PIT 6-21
PLL, 9-7
Pointer 14-16
Port
  A data register (PORTQA) 13-15
  B data register (PORTQB) 13-15
  data direction register (DDRQA) 13-15
port size device interfaces (illustration), 9-29
port width, 9-1
PORTQA 13-15
PORTQB 13-15
PORTQS 14-11
Positive stress 13-81
PowerPC standards
  PowerPC Operating Environment Architecture (Book 3)
    branch processor, 3-43
    fixed-point processor
      special purpose registers, 3-44
    fixed-point processor, 3-44
    interrupts, 3-44
    optional facilities and instructions, 3-54
    storage control instructions, 3-44
    timer facilities, 3-54
  PowerPC Operating Environment Architecture (Book 3), 3-43
  PowerPC User Instruction Set Architecture (Book 1)
    branch instructions, 3-39
    branch processor, 3-39
    computation modes, 3-38
    exceptions, 3-39
    fixed point-processor, 3-40
    instruction classes, 3-39
    load/store processor, 3-41
    reserved fields, 3-38
  PowerPC User Instruction Set Architecture (Book 1), 3-38
  PowerPC Virtual Environment Architecture (Book 2)
    operand placement effects, 3-42
    storage control instructions, 3-43
    timebase, 3-43
  PowerPC Virtual Environment Architecture (Book 2), 3-42
PowerPC User Instruction Set Architecture
  Book 1
    instruction fetching, 3-39
PPWA D-32
PQA 13-15, 13-71
PQB 13-15
PQSPAR 14-10, 14-34, 14-38
PR bit 3-7, 3-21
PRE 22-52
Precise exceptions 3-35
PREE 22-54
Prescaler 13-53
  clock
    (PSCK) 18-12
  clock high time (PSH) 13-17
  control
    for TCR1 18-5
    for TCR2 18-7
  divide
    factor field 16-28
    register (PRESDIV) 16-8, 16-27
PRESDIV (bit field) 16-28

PRESDIV (register) 16-8, 16-9, 16-27
Privilege level 3-7, 3-21
Processor version register 3-25
Programmable
  channel service priority 18-4
  transfer length 14-15
Programmable time accumulator (PTA) D-4
  parameters D-5, D-44, D-46
Propagation segment time (PROPSEG) 16-27
PROPSEG 16-11, 16-27
PSCK 18-12
PSEG1 16-28
PSEG2 16-9, 16-11, 16-28
PSEGS1 16-11
PSH 13-17
PT 14-47, 14-53
PTA D-4
PTR, 9-4, 9-37
Pulse-width modulation (PWM) D-36
  parameters D-37
PVR 3-25
PWM D-36

–Q–

QACR0 13-9, 13-16
QACR1 13-9, 13-18
QACR2 13-9, 13-20
QADC64E right justified, unsigned result word table (RJURR) 13-37
QADCINT 13-9, 13-13
QADCMCR 13-9, 13-10
QADCTEST 13-9
QASR 13-9, 13-23
QCLK 13-52
  frequency 13-52
QDDR 14-13, 14-42
QILR 14-9
QOM D-6
QPAR 14-12
QPDR 14-11, 14-42
QS 13-27
QSM
  pin function 14-11
  QSPI 14-14
    operating modes 14-27
    operation 14-25
    RAM 14-22
  registers
    pin control registers 14-10
      port QS
        data direction register (DDRQS) 14-10
        data register (PORTQS) 14-11
    QSPI
      control register 0 (SPCR0) 14-17
      control register 1 (SPCR1) 14-19
      control register 2 (SPCR2) 14-20
      control register 3 (SPCR3) 14-20
      status register (SPSR) 14-20
    SCI
      control register 0 (SCCR0) 14-46
      control register 1 (SCCR1) 14-46
      data register (SCDR) 14-50
      status register (SCSR) 14-48
  SCI 14-42
    operation 14-51
    pins 14-51
    registers 14-45
QSM Data Direction Register (QDDR) 14-13, 14-42
QSM Interrupt Level Register (QILR) 14-9
QSM Pin Assignment Register (QPAR) 14-12
QSM Port Data Register (QPDR) 14-11, 14-42
QSMCMMCR bit settings 14-9
QSPI 14-14
  block diagram 14-15
  enable (SPE) 14-19
  finished flag (SPIF) 14-22
  initialization operation 14-28
  loop mode (LOOPQ) 14-21
  master operation flow 14-29
  operating modes 14-27
    master mode 14-27, 14-34
      wraparound mode 14-38
    slave mode 14-27, 14-38
  operation 14-25
  peripheral chip-selects 14-37
  RAM 14-22, 14-23
    command RAM 14-23
    receive RAM 14-23
    transmit RAM 14-23
QSPI Enable (SPE) 14-42
QSPI Status Register (SPSR) 14-42
Queue 13-41
  1 completion flag (CF1) 13-23
  1 completion interrupt enable (CIE1) 13-18
  1 operating mode (MQ1) 13-19
  1 pause flag (PF1) 13-24
  1 pause interrupt enable (PIE1) 13-18
  1 single-scan enable bit (SSE1) 13-19
  1 trigger overrun (TOR1) 13-25
  2 completion flag (CF2) 13-24
  2 completion software interrupt enable (CIE2) 13-20
  2 operating mode (MQ2) 13-21
  2 pause flag (PF2) 13-25
  2 pause software interrupt enable (PIE2) 13-20
  2 single-scan enable bit (SSE2) 13-21
  2 trigger overrun (TOR2) 13-26
  pointers
    completed queue pointer (CPTQP) 14-25
    end queue pointer (ENDQP) 14-25
    new queue pointer (NEWQP) 14-25
  priority 13-42
  priority schemes 13-57, 13-68
  status (QS) 13-27
Queued
  serial
    peripheral interface (QSPI) 14-14
Queued output match TPU function (QOM) D-6

–R–

RAF 14-49

RD/WR, 9-4
RDRF 14-49, 14-57
RE 14-45, 14-47, 14-57
RE bit 3-22, 3-26
read cycle, data bus requirements, 9-30
read/write (RD/WR), 9-36
Receive
  data
    register full (RDRF) 14-49
  error status flag (RXWARN) 16-31
  RAM 14-23
  time sample clock (RT) 14-52, 14-57
Receiver
  active (RAF) 14-49
  data register (RDRF) flag 14-57
  enable (RE) 14-47, 14-57
  interrupt enable (RIE) 14-47
  wakeup (RWU) 14-48, 14-59
Receiver Enable (RE) 14-45
Reception of transmitted frames 16-13
Recoverable exception 3-22, 3-26
Register diagrams
  CMD (DLCMD2 command register) 15-36
  ILR (DLCMD2 interrupt level register) 15-31
  IPR (DLCMD2 interrupt pending register) 15-30
  IVR (DLCMD2 interrupt vector register) 15-32
  MCR (DLCMD2 module configuration register) 15-27
  MDASMARAR MDASM DataA register 17-60
  MDASMARBR MDASM DataB register 17-61
  MDASMSCR MDASM status/control register 17-62
  MMCSMCNT MMCSM up-counter register 17-42
  MMCSMMML MMCSM modulus latch register 17-42
  MMCSMSCR MMCSM status/control register 17-43
  module configuration (QADCMCR) 13-10
  MPIOSMDDR MPIOSM data direction register 17-82
  MPIOSMDR MPIOSM data register 17-82
  MPTCPR MRTCSM prescaler counter buffer register 17-93
  MPTCSM MRTCSM status/control register 17-93
  MPTCSMFRCH MRTCSM 32-bit counter high buffer register 17-92
  MPTCSMFRCL MRTCSM 32-bit counter low buffer register 17-93
  MPWMCNTR MPWMSM counter register 17-76
  MPWMPERR MPWMSM period register 17-75
  MPWMPULR MPWMSM pulse width register 17-75
  MPWMSCR MPWMSM status/control register 17-77
  RDATA (DLCMD2 receive data register) 15-34, 15-44
  SCTL (DLCMD2 sysmbol timing control and pre-scaler register) 15-32
  STAT (DLCMD2 status register) 15-41
  TDATA (DLCMD2 transmit data register) 15-40
Registers
  BBCMCR BBC module configuration register 4-25
  CALRAM_OTR CALRAM ownership trace register 21-21
  CMD (DLCMD2 command register) 15-36
  CMPA–CMPD 22-44
  CMPE–CMPF 22-45
  CMPG–CMPH 22-45
  conversion command word table (CCW) 13-33
  COUNTA 22-50
  COUNTB 22-51
  CRAM_RBAx CALRAM region base address register 21-19
  CRAMMCR CALRAM module configuration register 6-36, 7-11, 8-30, 20-19, 21-16
  CRAMOVL CALRAM overlay configuration register 21-20
  DCCR0-DCCR15 decompressor class configuration registers 4-32
  DER 22-53
  ECR 22-51
  EIBADR external interrupt relocation table base address register 4-31
  ICTRL 22-46
  ILR (DLCMD2 interrupt level register) 15-31
  interrupt (QADCINT) 13-13
  IPR (DLCMD2 interrupt pending register) 15-30
  IVR (DLCMD2 interrupt vector register) 15-32
  LCTRL1 22-48
  LCTRL2 22-49
  left justified, unsigned result format (LJURR) 13-37
  MCPSM register organization 17-36, 17-82, 17-92
  MCPSMCR MCPSM status/control register 17-36
  MCR (DLCMD2 module configuration register) 15-27
  MDASMAR MDASM DataA Register 17-60
  MDASMBR MDASM DataB Register 17-61
  MDASMSCR MDASM status/control register 17-62
  MI_GRA global region atribute register 4-29
  MI_RA 0-3 region attribute register 4-27
  MI_RBA 0-3 region base address register 4-27
  MIOS14ER0 interrupt enable register 17-27
  MIOS14LVL0 interrupt register 17-30
  MIOS14LVL1 register 17-30, 17-31
  MIOS14MCR module configuration register 17-34
  MIOS14RER1 interrupt enable register 17-29
  MIOS14RPR0 request pending register 17-28
  MIOS14RPR1 request pending register 17-29
  MIOS14RSR1 interrupt status register 17-28
  MIOS14SR0 interrupt status register 17-27
  MIOS14TPCR test and pin control register 17-32
  MIOS14VECT vector register 17-33
  MIOS14VNR module-version number register 17-33
  MMCSMCNT MMCSM up-counter register 17-42
  MMCSMMML MMCSM modulus latch register 17-42
  MMCSMSCR MMCSM status/control register 17-43
  module configuration (QADC64E) 13-10
  MPIOSMDDR MPIOSM data direction register 17-82
  MPIOSMDR MPIOSM data register 17-82
  MPWMCNTR MPWMSM counter register 17-76
  MPWMPERR MPWMSM period register 17-75
  MPWMPULR MPWMSM pulse width register 17-75
  MPWMSCR MPWMSM status/control register 17-77
  MRTCPR MRTCSM prescaler counter buffer register 17-93
  MRTCSM MRTCSM status/controlr register 17-93
  MRTCSMFRCH MRTCSM 32-bit counter high buffer register 17-92

MRTCSMFRCL MRTCSM 32-bit counter low buffer register 17-93
PDMCR pads module configuration register 2-4
PDMCR2 pads module configuration register 2-5
port A data register (PORTQA) 13-15
port B data register (PORTQB) 13-15
port data direction (DDRQA) 13-15
PORTQB port B data registerr 13-15
QACR0 control register 13-16
QADC64E control register 1 (QACR1) 13-18
QADC64E control register 2 (QACR2) 13-20
QADC64E left justified, signed result format (LJSRR) 13-37
QADC64E module configuration register 13-10
QADC64E PORTQA port A data register 13-15, 13-16
QADC64E PORTQB port B data register 13-15, 13-16
QADC64E right justified, unsigned result format (RJURR) 13-37
RDATA (DLCMD2 receive data register) 15-34, 15-44
SCTL (DLCMD2 sysmbol timing control and pre-scaler register) 15-32
STAT (DLCMD2 status register) 15-41
status (QASR) 13-23
status (QASR0) 13-23
status (QASR1) 13-29
supervisor level 3-20
TDATA (DLCMD2 transmit data register) 15-40
test (RAMTST) 19-5
user level 3-11
registers
special purpose
added registers, 3-44
unsupported registers, 3-44
special purpose, 3-44
Remote
frames 16-15
transmission request (RTR) 16-5, 16-6
reservation protocol for a multi-level (local) bus, 9-43
Reserved
mode 13-46
Reset
status register 7-5
Resistor-divider chain 13-40
Resolution time 13-39
Result word table 13-4, 13-36, 13-42
Resynchronization jump width (RJW) bit field 16-28
RETRY, 9-44
R_F 13-80
RIE 14-47
RJURR 13-37
RJW 16-11, 16-28
RN field 3-15
RSR 7-5
RSV, 9-37
RT 14-57
RTR 16-5, 16-6, 16-15
RWU 14-48, 14-59
RX
Length 16-5
RX14MSKHI 16-30
RX14MSKLO 16-30
RX15MSKHI 16-30
RX15MSKLO 16-30
RXECTR 16-33
RXGMSKHI 16-29
RXGMSKLO 16-29
RXWARN 16-31

–S–

S0 14-9
SAMP 16-27
Sample amplifier bypass (BYP) 13-33
Sampling mode (SAMP) 16-27
SAR 13-41
SBK 14-48, 14-54
Scan modes
continuous-scan modes
external trigger 13-50, 13-51
periodic timer continuous-scan mode 13-52
software initiated 13-50
single-scan modes
external trigger 13-47, 13-48
interval timer 13-48
software initiated 13-47
SCBR 14-46
SCCR0 14-46
SCCR1 14-46
SCDR 14-50
SCI 14-34, 14-42
baud
clock 14-52
rate (SCBR) 14-46
equation 14-46
idle-line detection 14-58
internal loop 14-59
operation 14-51
parity checking 14-53
pins 14-51
receiver
block diagram 14-44
operation 14-57
wakeup 14-59
registers 14-45
SCCR0 14-45
SCCR1 14-45
SCI Baud Rates 14-53
SCI SUBMODULE 14-13
SCSR 14-45
transmitter
block diagram 14-43
operation 14-53
SCI Control Register 0 (SCCR0) 14-45
SCI Control Register 1 (SCCR1) 14-45
SCI Status Register (SCSR) 14-45
SCK 14-12, 14-33, 14-38
actual delay before SCK (equation) 14-36
baud rate (equation) 14-35

S-clock 16-8
SCSR 14-48
SE bit 3-21
SEE 22-52
Send break (SBK) 14-48, 14-54
Sequencer, instruction 3-3
Serial
    clock baud rate (SPBR) 14-18
    communication interface (SCI) 14-42
    formats 14-52
    mode (M) bit 14-52
    shifter 14-54
Serial Clock (SCK) 14-12
Serialization
    fetch 22-1
Service
    request breakpoint flag (SRBK) 18-15
SGLR 18-21
Simplified mnemonics 3-33
Single-step trace enable 3-21
SIU signals, 9-4
SIW0EN 22-46
SIW1EN 22-46
SIW2EN 22-46
SIW3EN 22-46
Slave Select (SS) 14-42
Slave select signal (SS) 14-38, 14-39
SLW0EN 22-50
SLW1EN 22-50
snooping external bus activity, 3-42
SO bit 3-18
SOF 16-9
Soft reset control field (SOFT_RST) 18-21
SOFT_RST 18-21
SOFTRST 16-11
Software initiated
    continuous-scan mode 13-50
    single-scan mode 13-47
Software trap enable selection 22-46
SPBR 14-18
SPCR0 14-17
SPCR1 14-19
SPCR2 14-20
SPCR3 14-20
SPE 14-19, 14-42
Special-purpose registers, general 3-25
SPI
    finished interrupt enable (SPIFIE) 14-20
SPIF 14-22
SPIFIE 14-20
SPRG0–SPRG3 3-25
SPRGs 3-25
SPRs
    general 3-25
SPSR 14-20, 14-42
SPWM D-40
SRAM
    supervisor space only 14-9
SRBK 18-15
SRR 16-6
SRR0 3-24, 3-45, 3-48, 3-49, 3-50, 3-51, 3-52, 3-53
SRR1 3-24, 3-45, 3-46, 3-48, 3-49, 3-50, 3-51, 3-52, 3-53
SS 14-42
SS 14-38, 14-39
SSE1 13-19
SSE2 13-21
Standard
    message format 16-1
        frames 16-5
Star-point ground system 13-75
Start
    bit (beginning of data frame) 14-51
    -of-frame (SOF) symbol 16-9
State machine 13-52, 14-57
Status register (QASR) 13-9, 13-23
STF 18-12
STOP 13-10, 16-17, 18-12
Stop
    clocks to TCRs (CLKS) 18-14
    enable (STOP) 13-10
    enable (STOP) bit
        QSM 14-6
        TOUCAN 16-17
        TPU 18-12
    flag (STF) 18-12
    mode 13-10
    SCI end of data frame bit 14-51
storage control instructions, 3-44
storage reservation, 9-41
Stress conditions 13-81
STS, 9-5
STUFFERR 16-31
Subqueue 13-42
Substitute remote request (SRR) 16-6
Successive approximation register (SAR) 13-41
Summary overflow 3-18
Supervisor
    /Unrestricted data space (SUPV) 13-10
    /unrestricted data space (SUPV)
        TPU 18-12
Supervisor mode 3-20
    and SRAM 14-9
SUPV 13-10, 13-12
SUSG 22-48
SUSH 22-48
Synchronized pulse-width modulation (SPWM) D-40
SYSE 22-52
SYSEE 22-54
system clock output, 9-7
system reset interrupt, 3-44

–T–

T1 13-58
T2 13-58
T2CFILTER 18-22
T2CG 18-7, 18-12
T2CLK pin filter control (T2CFILTER) 18-22
T2CSL 18-12
TA, 9-6
Table stepper motor (TSM) D-8

TB 3-19
TBL 3-20, 3-23
TBRS1, TBRS0 - bits in BIUMCR 17-32, 17-33
TBU 3-20, 3-23
TC 14-49, 14-54
TCIE 14-47, 14-55
TCNMCR 16-22
TCR 18-13
TCR1P 18-12
TCR2 clock/gate control (T2CG) 18-12
TDRE 14-49
TE 14-45, 14-47
TEA, 9-6
termination signals, 9-39
Test register (QADCTEST) 13-9
test register (RAMTST) 19-5
TICR 18-15, 18-23
TIE 14-47, 14-55
Time
  quanta clock 16-8
  stamp 16-5, 16-10
Time base 3-19
time base bus
  selecting 17-32, 17-33
timebase, 3-43
TIMER 16-29
Timer
  count register
    1 prescaler control (TCR1P) 18-12
  synchronize mode (TSYNC) 16-27
Timing, instruction 3-36
TOR 13-58
TOR1 13-25
TOR2 13-26
TOUCAN
  address
    map 16-20
  bit timing configuration 16-8
    operation 16-9
  external pins 16-2
  initialization sequence 16-11
  interrupts 16-18
  message buffer address map 16-22
  operation 16-3
  receive process 16-13
  registers
    control register 0 (CANCTRL0) 16-25
    control register 1 (CTRL1) 16-8
    control register 1(CANCTRL1) 16-26
    control register 2 (CANCTRL2) 16-28
    control register 2 (CTRL2) 16-8
    error and status register (ESTAT) 16-30
    free running timer register (TIMER) 16-29
    interrupt
      configuration register (CANICR) 16-24
      flag register (IFLAG) 16-33
      mask register (IMASK) 16-32
    module configuration register (TCNMCR) 16-22
    receive
      buffer 14 mask registers (RX14MSKHI/LO) 16-30
      buffer 15 mask registers (RX15MSKHI/LO) 16-30
      global mask registers (RXGMSKLO/HI 16-29
    RX/TX error counter registers (RXECTR/TXEC-TR) 16-33
    test configuration register (CANTCR) 16-24
  special operating modes 16-16
    auto power save mode 16-18
    debug mode 16-16
    low-power stop mode 16-17
  transmit process 16-12
TPU
  address map 18-8
  components 18-2
  FREEZE flag (TPUF) 18-15
  function library 18-4
  host interface 18-2
  interrupts 18-5
  microengine 18-2
  operation 18-3
    coherency 18-4
    emulation support 18-4
    event timing 18-3
    interchannel communication 18-4
    programmable channel service priority 18-4
  parameter RAM 18-2, 18-23
    address map 18-24
  registers
    channel
      function select registers (CFSR) 18-16
      interrupt
        enable register (CIER) 18-5, 18-16
        status register (CISR) 18-5, 18-20
      priority registers (CPR) 18-19
    decoded channel number register (DCNR) 18-21
    development
      support control register (DSCR) 18-13
      support status register (DSSR) 18-15
    host
      sequence registers (HSQR) 18-18
      service request registers (HSSR) 18-18
    link register (LR) 18-21
    module configuration register (TPUMCR) 18-11
    service grant latch register (SGLR) 18-21
    test configuration register (TCR) 18-13
    TPU interrupt configuration register (TICR) 18-15, 18-23
  scheduler 18-2
  time
    bases 18-2
  timer channels 18-2
TPU Reference Manual 18-3, 18-18
TPU2
  module configuration register 2 (TPUMCR2) 18-21
TPUF 18-15
TPUMCR 18-11
TPUMCR2 18-21
TR 22-52

trace interrupt, 3-47
transaction (bus), 9-8
Transfer
    length options 14-37
    time 13-39
transfer acknowledge (TA), 9-39
transfer error acknowledge (TEA), 9-39
transfer size (TSIZ), 9-37
transfer start (TS), 9-36
transfers, alignment and packaging, 9-28
transfers, burst-inhibited, 9-16
transfers, termination signals, 9-39
Transmission
    complete
        (TC) flag 14-54
        interrupt enable (TCIE) 14-55
Transmit
    /receive status (TX/RX) 16-31
    bit error (BITERR) 16-31
    complete
        bit (TC) 14-49
        interrupt enable (TCIE) 14-47
    data
        register empty (TDRE) flag 14-49
    error status flag (TXWARN) 16-31
    interrupt enable (TIE) 14-47, 14-55
    pin configuration control (TXMODE) 16-25
    RAM 14-23
Transmitter Enable (TE) 14-45
Transmitter enable (TE) 14-47, 14-53
TRE 22-54
Trigger
    event 13-31, 13-57
    overrun error (TOR) 13-58
TS, 9-5
TSIZ(0
    1), 9-4
TSIZ0, 9-1
TSIZ1, 9-1
TSM D-8
T_{SR} 13-11
TSYNC 16-27
TX
    Length 16-5
TX/RX 16-31
TXECTR 16-33
TXMODE 16-25
TXWARN 16-31

–U–

UART D-13
UISA register set 3-11
Universal asynchronous receiver/transmitter (UART) D-13
    parameters
        receiver parameters D-15
        transmitter parameters D-14
Unordered exceptions 3-35
User level registers 3-11
Using the TPU Function Library and TPU Emulation Mode 18-4
UX bit 3-14

–V–

V_{CF} 13-80
V_{DDA} 13-72, 13-76
VE bit 3-15
Vector table, exception 3-36
Vector table, exceptions 3-35
V_{IH} 13-72
V_{IL} 13-72
Voltage
    inputs 13-72
    reference pins 13-76
V_{RH} 13-36, 13-40, 13-76, 13-80
V_{RL} 13-40, 13-76, 13-80
V_{SRC} 13-80
V_{SSA} 13-72, 13-76
VX bit 3-14
VXCVI bit 3-15
VXIDI 3-14
VXIMZ bit 3-14
VXISI 3-14
VXSNAN 3-14
VXSOFT bit 3-15
VXSQRT bit 3-15
VXVC bit 3-14
VXZDZ bit 3-14

–W–

WAKE 14-47, 14-59
Wake interrupt (WAKEINT) 16-32
WAKEINT 16-17, 16-32
WAKEMSK 16-17
Wakeup
    address mark (WAKE) 14-47, 14-59
Wired-OR
    mode
        for QSPI pins (WOMQ) 14-18
        for SCI pins (WOMS) 14-47, 14-54
WOMQ 14-18
WOMS 14-47, 14-54
Wrap
    enable (WREN) 14-20
    to (WRTO) 14-20
Wraparound mode 14-16
    master 14-38
WREN 14-20
write cycle data bus contents, 9-30
WRTO 14-20

–X–

XE bit 3-15
XER 3-17
XX bit 3-14

–Z–

ZE bit 3-15

ZX bit 3-14