SIM
SYSTEM INTEGRATION MODULE
REFERENCE MANUAL

TABLE OF CONTENTS

Paragraph **Title** **Page**

SECTION 1INTRODUCTION

SECTION 2 SIGNAL AND PIN DESCRIPTIONS

SECTION 3 SYSTEM CONFIGURATION AND PROTECTION

SECTION 4 SYSTEM CLOCK

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 5 EXTERNAL BUS INTERFACE

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 6 INTERRUPTS

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

TABLE OF CONTENTS
(Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

PREFACE

This manual describes the capabilities, operation, and functions of the system integration module (SIM), an integral module of Motorola's family of modular microcontrollers. 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, including the SIM reference manual, that provide detailed information about module operation and applications. Refer to Motorola publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation.

The following conventions are used throughout the manual.

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.

A signal that is **asserted** is in its 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.

A signal that is **negated** is in its inactive 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.

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

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.

SECTION 1INTRODUCTION

The system integration module (SIM) is a module on many Motorola 16- and 32-bit modular microcontroller units (MCUs). SIM-based MCUs contain a SIM, a CPU, and some combination of communication, timing, and memory modules. The different modules perform the following tasks:

- The SIM supplies a clock signal to the rest of the microcontroller, provides system protection features, and manages the external bus. In addition, the SIM provides on-chip chip-select signals and (if the pins are not being used for their alternate functions) I/O ports.
- The CPU contains the microcode to process the instructions in its instruction set. The CPU also works with the SIM to support exception processing (including processing of interrupts and reset requests), system initialization, special CPU bus cycles (including breakpoint-acknowledge cycles), input/output, and separate supervisor and user privilege levels.
- To understand the SIM, it is necessary to be familiar with the microcontroller's CPU. Use this reference manual in conjunction with the appropriate CPU reference manual. The CPU16 and CPU32 are the CPUs currently used with the SIM. **1.3 CPU-Specific Differences Affecting SIM Operation** summarizes the differences between the CPU32-based SIM and the CPU16-based SIM.
- Communication and timing modules include an analog-to-digital converter (ADC), time-processing unit (TPU), general-purpose timer (GPT), queued serial module (QSM), and multichannel communications interface (MCCI). These modules are present in different combinations on different MCUs.
- Memory modules include standby RAM, ROM, EEPROM, Flash EEPROM, and standby RAM with TPU-emulation capabilities (TPURAM). These modules are present in different combinations on different MCUs.

The different modules on an MCU communicate with one another and with external components via the intermodule bus (IMB), a standardized bus developed to facilitate design of modular microcontrollers. The IMB supports 24 address and 16 data lines.

NOTE

On CPU16-based MCUs, external address lines ADDR[23:20] follow the state of ADDR19.

The SIM consists of the following functional blocks:

- The system configuration and protection block controls configuration parameters and provides bus and software watchdog monitors. In addition, it provides a periodic interrupt generator to support execution of time-critical control routines.
- The system clock generates clock signals used by the SIM, other IMB modules, and external devices.
- The external bus interface handles the transfer of information between IMB modules and external address space.

- The chip-select block provides 12 chip-select signals. Each chip-select signal has an associated base register and option register that contain the programmable characteristics of that chip select. A data port, port C, is available for discrete output on pins not being used for their chip-select function or alternate function as address or function code lines.
- Two data ports, port E and port F, are available for general-purpose input and output if not required for their alternate function. A port data register, data direction register, and pin assignment register are associated with each port.
- The system test block incorporates hardware necessary for testing the MCU. Its use in normal applications is not supported.

NOTE

Some SIM-based MCUs have a reduced pin set due to pin limitations. Some of the chip-select and data port pins described in this manual may not be present on these MCUs. Refer to the user's manual for the particular MCU for a list of the available pins on that device. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** for additional information.

Figure 1-1 is a block diagram of the SIM.

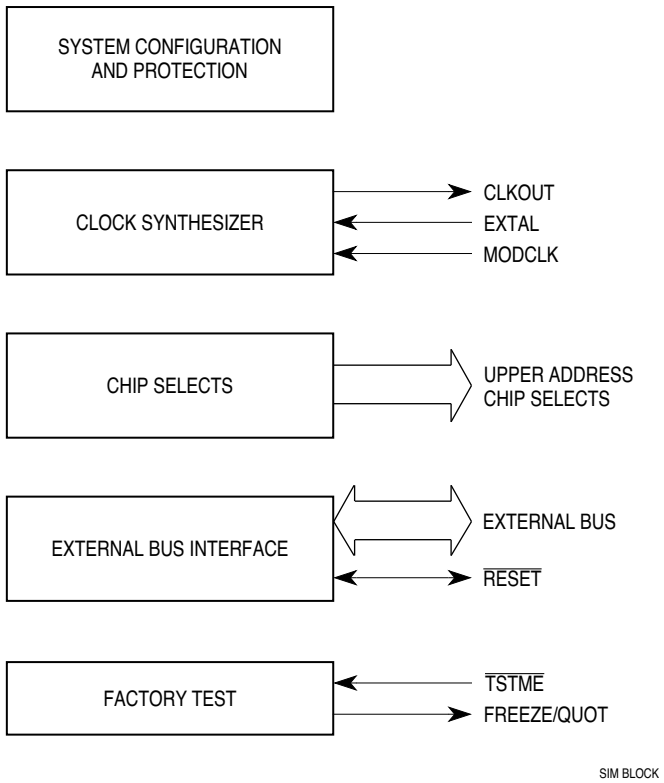


Figure 1-1 System Integration Module Block Diagram

Figure 1-2 shows the input and output signals associated with each functional block of the SIM. These signals are described more fully in **SECTION 2 SIGNAL AND PIN DESCRIPTIONS** and in subsequent sections of the manual.

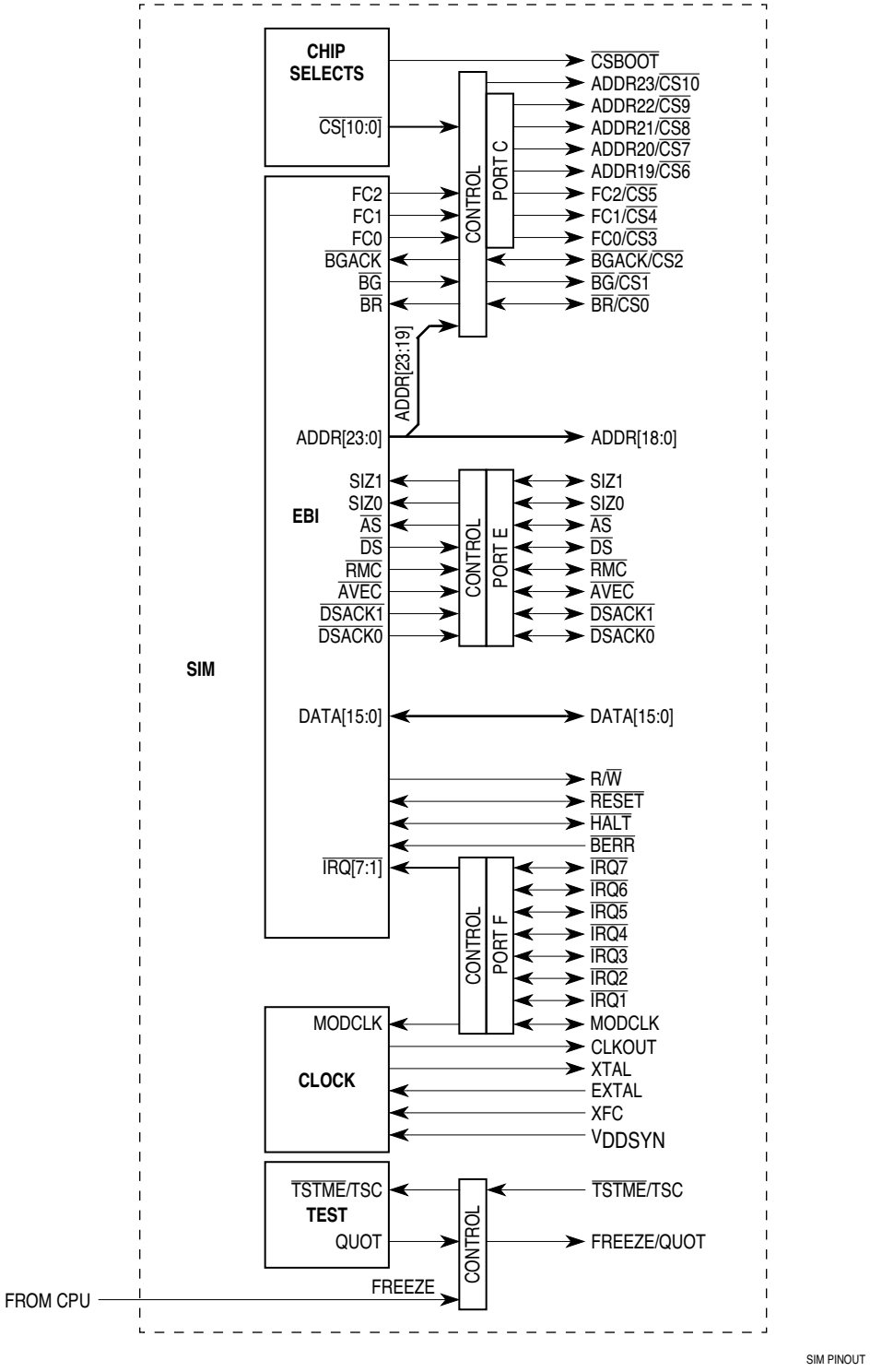


Figure 1-2 SIM Input and Output Signals

1.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in the SIM module configuration register (SIMCR) determines where the control register block is located in the system memory map. When MM is equal to zero, register addresses range from $7FF000 to $7FFFFF. When MM = 1, register addresses range from $FFF000 to $FFFFFF.

CAUTION

On CPU16-based MCUs, ADDR[23:20] follow the logic state of ADDR19 unless externally driven. If MM is cleared on these MCUs, the SIM maps IMB modules into an address space which is inaccessible to the CPU. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit can be written once. Initialization software should make certain it remains set by writing a one to it.

Table 1-1 is the SIM register 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 access is required; a designation of “S/U” indicates that the register can be programmed to the desired privilege level. Refer to **3.1 Module Configuration and Testing** for information on assigning privilege levels.

NOTE

CPU16-based MCUs do not support separate supervisor and user privilege levels. The CPU16 always operates at the supervisor privilege level.

Table 1-1 provides SIM register addresses relative to the SIM base address. In this table, the four high-order nibbles of each address are listed as $####. Refer to the user's manual for the specific MCU for the exact location of these registers. Remember that the MSB is determined by the MM bit.

Table 1-1 SIM Address Map

Access	Address	15 8	7 0
S	$####00	SIM CONFIGURATION REGISTER (SIMCR)	
S	$####02	SIM TEST REGISTER (SIMTR)	
S	$####04	SYNTHESIZER CONTROL REGISTER (SYNCR)	
S	$####06	UNUSED	RESET STATUS REGISTER (RSR)
S	$####08	SYSTEM TEST REGISTER E (SIMTRE)	
S	$####0A	UNUSED	UNUSED
S	$####0C	UNUSED	UNUSED
S	$####0E	UNUSED	UNUSED
S/U	$####10	UNUSED	PORT E DATA (PORTE0)
S/U	$####12	UNUSED	PORT E DATA (PORTE1)
S/U	$####14	UNUSED	PORT E DATA DIRECTION (DDRE)
S	$####16	UNUSED	PORT E PIN ASSIGNMENT (PEPAR)
S/U	$####18	UNUSED	PORT F DATA (PORTF0)
S/U	$####1A	UNUSED	PORT F DATA (PORTF1)
S/U	$####1C	UNUSED	PORT F DATA DIRECTION (DDRF)
S	$####1E	UNUSED	PORT F PIN ASSIGNMENT (PFPAR)
S	$####20	UNUSED	SYSTEM PROTECTION CONTROL (SYPCR)
S	$####22	PERIODIC INTERRUPT CONTROL REGISTER (PICR)	
S	$####24	PERIODIC INTERRUPT TIMING REGISTER (PITR)	
S	$####26	UNUSED	SOFTWARE SERVICE (SWSR)
S	$####28	UNUSED	UNUSED
S	$####2A	UNUSED	UNUSED
S	$####2C	UNUSED	UNUSED
S	$####2E	UNUSED	UNUSED
S	$####30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$####32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$####34	TEST MODULE SHIFT COUNT (TSTSC)	
S	$####36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$####38	TEST MODULE CONTROL (CREG)	
S/U	$####3A	TEST MODULE DISTRIBUTED (DREG)	
S	$####3C	UNUSED	UNUSED
S	$####3E	UNUSED	UNUSED
S/U	$####40	UNUSED	PORT C DATA (PORTC)
S/U	$####42	UNUSED	UNUSED
S	$####44	CHIP-SELECT PIN ASSIGNMENT REGISTER (CSPAR0)	
S	$####46	CHIP-SELECT PIN ASSIGNMENT REGISTER (CSPAR1)	
S	$####48	CHIP-SELECT BASE ADDRESS REGISTER BOOT (CSBARBT)	
S	$####4A	CHIP-SELECT OPTION REGISTER BOOT (CSORBT)	
S	$####4C	CHIP-SELECT BASE ADDRESS REGISTER 0 (CSBAR0)	
S	$####4E	CHIP-SELECT OPTION REGISTER 0 (CSOR0)	
S	$####50	CHIP-SELECT BASE ADDRESS REGISTER 1 (CSBAR1)	
S	$####52	CHIP-SELECT OPTION REGISTER 1 (CSOR1)	
S	$####54	CHIP-SELECT BASE ADDRESS REGISTER 2 (CSBAR2)	
S	$####56	CHIP-SELECT OPTION REGISTER 2 (CSOR2)	
S	$####58	CHIP-SELECT BASE ADDRESS REGISTER 3 (CSBAR3)	
S	$####5A	CHIP-SELECT OPTION REGISTER 3 (CSOR3)	

Table 1-1 SIM Address Map

Access	Address	15 8	7 0
S	$####5C	CHIP-SELECT BASE ADDRESS REGISTER 4 (CSBAR4)	
S	$####5E	CHIP-SELECT OPTION REGISTER 4 (CSOR4)	
S	$####60	CHIP-SELECT BASE ADDRESS REGISTER 5 (CSBAR5)	
S	$####62	CHIP-SELECT OPTION REGISTER 5 (CSOR5)	
S	$####64	CHIP-SELECT BASE ADDRESS REGISTER 6 (CSBAR6)	
S	$####66	CHIP-SELECT OPTION REGISTER 6 (CSOR6)	
S	$####68	CHIP-SELECT BASE ADDRESS REGISTER 7 (CSBAR7)	
S	$####6A	CHIP-SELECT OPTION REGISTER 7 (CSOR7)	
S	$####6C	CHIP-SELECT BASE ADDRESS REGISTER 8 (CSBAR8)	
S	$####6E	CHIP-SELECT OPTION REGISTER 8 (CSOR8)	
S	$####70	CHIP-SELECT BASE ADDRESS REGISTER 9 (CSBAR9)	
S	$####72	CHIP-SELECT OPTION REGISTER 9 (CSOR9)	
S	$####74	CHIP-SELECT BASE ADDRESS REGISTER 10 (CSBAR10)	
S	$####76	CHIP-SELECT OPTION REGISTER 10 (CSOR10)	
	$####78	UNUSED	UNUSED
	$####7A	UNUSED	UNUSED
	$####7C	UNUSED	UNUSED
	$####7E	UNUSED	UNUSED

1.2 Reset Mode Selection

The following information is a concise reference to one aspect of system reset. System reset is a complex operation. To understand SIM operation during and after reset, refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION**.

The logic states of certain data bus pins during reset determine the function of SIM pins that can be assigned to more than one function. In addition, the state of the DATA11 pin determines whether test mode is enabled, the state of the MODCLK pin determines the source of the system clock, and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. **Table 1-2** is a summary of reset mode selection options.

Table 1-2 SIM Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS}}$[7:6] $\overline{\text{CS}}$[8:6] $\overline{\text{CS}}$[9:6] $\overline{\text{CS}}$[10:6]	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZE}}$	PORTE
DATA9	$\overline{\text{IRQ}}$[7:1] MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

1.3 CPU-Specific Differences Affecting SIM Operation

This reference manual can be used with Motorola modular MCUs that contain a system integration module, regardless of whether the chip contains a CPU16 or CPU32. Certain aspects of SIM operation, however, vary according to which CPU is present. Whenever a feature being discussed is CPU-dependent, this manual refers the reader to the appropriate CPU reference manual or to the user's manual for a specific MCU.

The major differences between the CPU16 and CPU32 that affect SIM operation are summarized in **Table 1-3**.

Table 1-3 CPU Differences Affecting SIM Operation

Feature	CPU16 Operation	CPU32 Operation
Address Bus	ADDR[23:20] follow state of ADDR19	All 24 lines are operational
Module Mapping	MM bit in SIMCR must equal 1	MM bit in SIMCR can equal 0 or 1
Exception Vector Table	Each vector (except reset vector) is one word; vector table is 512 bytes and is not relocatable (there is no vector base register)	Each vector (except reset vector) is two words; vector table is 1024 bytes and is relocatable (base address is stored in vector base register)
Privilege Levels	CPU always operates at supervisor privilege level; FC2 always = 1	CPU supports supervisor and user privilege levels; FC2 is active according to privilege level
Memory Partitions	Memory partitioned into 16 banks of 64 Kbytes; register extension fields sup-port bank switching; separate 1-Mbyte program and data spaces available	Memory is fully accessible as 16-Mbyte program and data spaces
$\overline{\text{RMC}}$ pin, port E	Pin may or may not be connected for I/O (PE3); $\overline{\text{RMC}}$ function not supported for indivisible read-modify-write operations	Pin connected, port E fully operational, $\overline{\text{RMC}}$ asserted during indivisible read-modify-write operations
Misaligned Transfers	Misaligned transfers are allowed	Misaligned transfers are not supported
Retry Operation	Retry operation not supported	Assertion of $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ results in retry sequence

SECTION 2 SIGNAL AND PIN DESCRIPTIONS

The tables in this section summarize functional characteristics of SIM pins. For a more detailed discussion of a particular signal, refer to the section of this manual that discusses the SIM function or submodule involved. Refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION** for details on pin state during and after system reset.

NOTE

On MCUs with a reduced pin-count SIM, some of the pins described in this section may not be available. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** for additional information. Refer to the user's manual for the particular MCU for a list of the pins on the chip.

2.1 Pin Characteristics

Table 2-1 shows types of output drivers.

Table 2-1 SIM Output Driver Types

Type	Description
A	Output-only signals that are always driven; no external pull-up required
Aw	Type A output with weak P-channel pull-up during reset
B	Output that includes circuitry to pull up output before high-impedance state is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while the pin is in the high-impedance state. Assertion of TSC always places these pins in a high-impedance state; in addition, many bus control outputs are placed in a high-impedance state when bus is granted to an external master. Refer to **5.10 Bus Arbitration** for details.
Bo	Type B output that can be operated in an open-drain mode

Table 2-2 shows all inputs and outputs. Digital inputs and outputs use CMOS logic levels. An entry in the Discrete I/O column indicates that a pin can also be used for general-purpose input, output, or both. I/O port designation is given when it applies. For details on the states of bus control signals when the bus is granted to an external master, refer to **5.10 Bus Arbitration**.

Table 2-2 SIM Pin Characteristics

Pin Mnemonic	Output Driver	Input Synchronized	Input Hysteresis	Discrete I/O	Port Designation
ADDR23/$\overline{CS10}$/ECLK	A	Y	N	O	—
ADDR[22:20]/$\overline{CS}$[9:7][1]	A	Y	N	O	PC[6:4]
ADDR19/$\overline{CS6}$	A	Y	N	O	PC3
ADDR[18:0]	A	Y	N	—	—
$\overline{AS}$	B	Y	N	I/O	PE5
$\overline{AVEC}$[1]	B	Y	N	I/O	PE2
$\overline{BERR}$	B	Y	N	—	—
$\overline{BG}$/$\overline{CS1}$	B	—	—	—	—
$\overline{BGACK}$/$\overline{CS2}$	B	Y	N	—	—
$\overline{BR}$/$\overline{CS0}$	B	Y	N	—	—
CLKOUT[3]	A	—	—	—	—
$\overline{CSBOOT}$	B	—	—	—	—
DATA[15:0][2]	Aw	Y	N	—	—
$\overline{DS}$	B	Y	N	I/O	PE4
$\overline{DSACK1}$	B	Y	N	I/O	PE1
$\overline{DSACK0}$[1]	B	Y	N	I/O	PE0
EXTAL[3]	—	—	Special	—	—
FC[2:0]/$\overline{CS}$[5:3]	A	Y	N	O	PC[2:0]
$\overline{HALT}$[1]	Bo	Y	N	—	—
$\overline{IRQ}$[7:6]	B	Y	Y	I/O	PF[7:6]
$\overline{IRQ}$[5:1][1]	B	Y	Y	I/O	PF[5:1]
MODCLK[2]	B	Y	N	I/O	PF0
R/$\overline{W}$	A	Y	N	—	—
$\overline{RESET}$	Bo	Y	Y	—	—
$\overline{RMC}$[1]	B	Y	N	I/O	PE3
SIZ[1:0]	B	Y	N	I/O	PE[7:6]
$\overline{TSTME}$/TSC	—	Y	Y	—	—
XFC[3]	—	—	—	—	—
XTAL[3]	—	—	—	—	—

NOTES
1. These pins may not be implemented on certain MCUs.
2. DATA[15:0] are synchronized during reset only. MODCLK is synchronized only when used as an input port pin.
3. CLKOUT, EXTAL, XFC, and XTAL are clock reference connections.

2.2 Signal Descriptions

The following tables are a quick reference to SIM signals. **Table 2-3** shows signal name, type, and active state. **Table 2-4** describes signal functions. Both tables are sorted alphabetically by mnemonic. Since MCU pins often have multiple functions, more than one description may apply to a pin.

Table 2-3 SIM Signal Characteristics

Signal Name	Signal Type	Active State
ADDR[23:0]	Bus	—
$\overline{AS}$	Output	0
$\overline{AVEC}$	Input	0
$\overline{BERR}$	Input	0
$\overline{BG}$	Output	0
$\overline{BGACK}$	Input	0
$\overline{BR}$	Input	0
CLKOUT	Output	—
$\overline{CS[10:0]}$	Output	0
$\overline{CSBOOT}$	Output	0
DATA[15:0]	Bus	—
$\overline{DS}$	Output	0
$\overline{DSACK[1:0]}$	Input	0
EXTAL	Input	—
FC[2:0]	Output	—
FREEZE	Output	1
$\overline{HALT}$	Input/Output	0
$\overline{IRQ[7:1]}$	Input	0
MODCLK	Input	—
PC[6:0]	Output	(Port)
PE[7:0]	Input/Output	(Port)
PF[7:0]	Input/Output	(Port)
R/$\overline{W}$	Output	1/0
$\overline{RESET}$	Input/Output	0
$\overline{RMC}$	Input/Output	0
SIZ[1:0]	Output	—
TSC	Input	—
$\overline{TSTME}$	Input	0
XFC	Input	—
XTAL	Output	—

Table 2-4 SIM Signal Function

Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus
Address Strobe	$\overline{AS}$	Indicates that a valid address is on the address bus
Autovector	$\overline{AVEC}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{BERR}$	Indicates that a bus error has occurred
Bus Grant	$\overline{BG}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{BGACK}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{BKPT}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{BR}$	Indicates that an external device requires bus mastership
Chip Selects	$\overline{CS}$[10:0]	Select external devices at programmed addresses
Boot Chip Select	$\overline{CSBOOT}$	Chip select for external boot start-up ROM
System Clockout	CLKOUT	System clock output
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{DS}$	During a read cycle, indicates that an external device should place valid data on the data bus. During a write cycle, indicates that valid data is on the data bus.
Halt	$\overline{HALT}$	Suspend external bus activity
Interrupt Request Level	$\overline{IRQ}$[7:1]	Provides an interrupt priority level to the CPU
Data and Size Acknowledge	$\overline{DSACK}$[1:0]	Provide asynchronous data transfers and dynamic bus sizing
Reset	$\overline{RESET}$	System reset
Test Mode Enable	$\overline{TSTME}$	Hardware enable for SIM test mode
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
Function Codes	FC[2:0]	Identify processor state and current address space
Clock Mode Select	MODCLK	Selects the source and type of system clock
Port C	PC[6:0]	SIM digital output port signals
Port E	PE[7:0]	SIM digital I/O port signals
Port F	PF[7:0]	SIM digital I/O port signals
Read-Modify-Write	$\overline{RMC}$	Indicates indivisible read-modify-write cycle (CPU32 test-and-set instruction)
Read/Write	R/$\overline{W}$	Indicates the direction of data transfer on the bus
Size	SIZ[1:0]	Indicates the number of bytes to be transferred during a bus cycle
Three-State Control	TSC	Places all output drivers in a high-impedance state
External Filter Capacitor	XFC	Connection for external phase-locked loop filter capacitor

SECTION 3 SYSTEM CONFIGURATION AND PROTECTION

The system configuration and protection submodule controls MCU configuration and testing, monitors internal activity, monitors reset status, and provides periodic interrupt generation. Providing these functions on chip reduces the number of external components in a complete control system.

For a discussion of the reset status register, refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION**. Other aspects of system configuration and protection are discussed in the following paragraphs. **Figure 3-1** is a block diagram of the submodule.

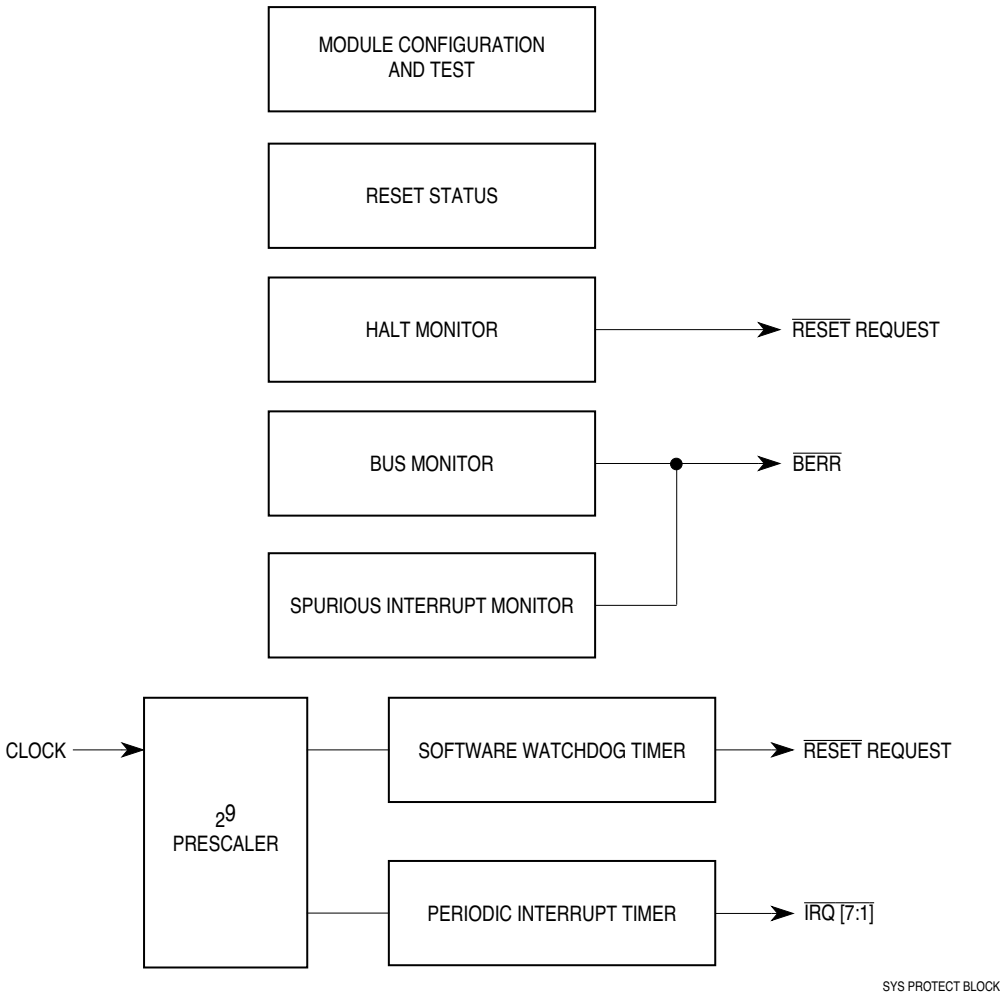


Figure 3-1 System Configuration and Protection

3.1 Module Configuration and Testing

The SIM configuration register (SIMCR) governs the operation of the system protection block and other aspects of system operation. System protection is discussed in detail later in this section. The following paragraphs describe the other aspects of system configuration controlled by the SIMCR.

3.1.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in the SIM module configuration register (SIMCR) determines where the control register block is located in the system memory map. When MM is equal to zero, register addresses range from $7FF000 to $7FFFFF. When MM = 1, register addresses range from $FFF000 to $FFFFFF.

CAUTION

On CPU16-based MCUs, ADDR[23:20] follow the logic state of ADDR19. On these MCUs, if MM is cleared, the SIM maps MCU modules into an address space that is inaccessible to the CPU. Modules remain inaccessible until reset occurs. The reset state of MM is one, but the bit can be written once. Initialization software for these MCUs should make certain MM remains set by writing a one to it.

3.1.2 Privilege Levels

To protect system resources, the processor in CPU32-based MCUs can operate at either the user or supervisor privilege level. On CPU32-based MCUs, access to most SIM registers is permitted only when the CPU is operating at the supervisor privilege level. The remaining SIM registers are programmable to permit supervisor access only or to permit access when the CPU is operating at either the supervisor or user privilege level.

If the SUPV bit in the SIMCR is set, access to SIM registers is permitted only when the CPU is operating at the supervisor level. If SUPV is cleared, then access to certain SIM registers is permitted when the CPU is operating at either the supervisor or user privilege level. The SIM address map (**Table 1-1**) indicates which registers are programmable to allow access from either privilege level.

CPU16-based MCUs, which do not support the user privilege level, always operate at the supervisor level, so that SIM (and all MCU) registers are always accessible. The state of the SUPV bit has no meaning on these MCUs.

3.1.3 Response to FREEZE Assertion

When the CPU enters background debugging mode, it suspends instruction execution and asserts the internal FREEZE signal. The CPU enters background debugging mode if a breakpoint occurs while background mode is enabled. Refer to the appropriate CPU manual for a discussion of background debugging mode.

Two bits in the SIMCR determine how the SIM responds to FREEZE assertion. Setting the freeze bus monitor (FRZBM) bit in the SIMCR disables the bus monitor when

FREEZE is asserted. Setting the freeze software watchdog (FRZSW) bit disables the software watchdog and the periodic interrupt timer when FREEZE is asserted. If these bits are cleared when FREEZE is asserted, the bus monitor, software watchdog, and periodic interrupt timer continue to operate normally.

FREEZE assertion has no affect on the halt monitor or spurious interrupt monitor: they continue to operate normally.

3.1.4 Interrupt Arbitration Priority

Each module that can generate interrupts, including the SIM, has an IARB (interrupt arbitration number) field in its module configuration register. Each IARB field has a different value. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level.

The reset value of IARB in the SIMCR is $F. This prevents SIM interrupts from being discarded. Initialization software must set the IARB field to a lower value in the range $F (highest priority) to $1 (lowest priority) if lower priority interrupts are to be arbitrated.

Refer to **SECTION 6 INTERRUPTS** for additional information.

3.1.5 Factory Test Mode

The internal IMB can be slaved to an external master for direct module testing. This mode is reserved for factory testing. Factory test mode is enabled by holding DATA11 low during reset. The factory test (slave) mode enabled (SLVEN) bit is a read-only bit that shows the reset state of DATA11.

3.1.6 SIM Configuration Register

The SIM configuration register (SIMCRS) controls system configuration. It can be read or written at any time. (Refer to the discussion of the MM bit, however, for certain restrictions.)

SIMCR — Module Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN		SUPV	MM	0	0	IARB			
RESET:															
0	0	0	0	DATA 11	0	0	0	1	1	0	0	1	1	1	1

EXOFF — External Clock Off

0 = The CLKOUT pin is driven from an internal clock source.
1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable

0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.
1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts during software debugging.

FRZBM — Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the bus monitor continues to operate.

1 = When FREEZE is asserted, the bus monitor is disabled.

SLVEN — Factory Test (Slave) Mode Enabled

0 = IMB is not available to an external tester.

1 = An external tester has direct access to the IMB.

SHEN[1:0] — Show Cycle Enable

This field determines what the external bus interface does with the external bus during internal transfer operations. Refer to **5.11 Show Cycles** for more information.

SHEN	Action
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

SUPV — Supervisor/Unrestricted Data Space

0 = Registers with access controlled by this bit are unrestricted

1 = Registers with access controlled by this bit are restricted to supervisor access only.

MM — Module Mapping

0 = Internal modules are addressed from $7FF000 — $7FFFFF.

1 = Internal modules are addressed from $FFF000 — $FFFFFF.

This bit can be written only once. Subsequent attempts to change this bit are ignored. Address space $7FF000 – $7FFFFF is inaccessible to the CPU16. On CPU16-based microcontrollers, MM must always be set. Initialization software for these MCUs should make certain MM remains set (its reset state) by writing a one to it.

IARB[3:0] — Interrupt Arbitration Field

IARB determines SIM interrupt arbitration priority. The reset value is $F (highest priority), to prevent SIM interrupts from being discarded during initialization. Refer to **3.1.4 Interrupt Arbitration Priority** and **SECTION 6 INTERRUPTS** for additional information.

3.1.7 SIM Test Registers

SIMTRE — System Integration Test Register (ECLK) **$####08**

The SIMTRE is used for factory test only.

SIMTR — System Integration Test Register **$####02**

SIMTR is used for factory test only.

3.2 Internal Bus Monitor

The internal bus monitor checks for excessively long response times during normal bus cycles (those terminated by $\overline{\text{DSACK}}$, or $\overline{\text{AVEC}}$ during autovector cycles). The monitor asserts internal $\overline{\text{BERR}}$ when response time is excessive. Refer to **SECTION 5 EXTERNAL BUS INTERFACE** for a discussion of external bus cycles and $\overline{\text{DSACK}}$ signals. Refer to **SECTION 6 INTERRUPTS** for a discussion of $\overline{\text{AVEC}}$ signals during interrupt acknowledge cycles.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. Maximum allowable response time can be selected by setting the bus monitor timing (BMT) field in the system protection control register (SYPCR). **Table 3-1** shows possible periods.

Table 3-1 Bus Monitor Period

BMT	Bus Monitor Time-out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU initiates a bus cycle. The BME bit in the SYPCR enables the internal bus monitor for internal to external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented, and the internal to external bus monitor option must be disabled.

Refer to **3.8 System Protection Registers** for a register diagram of the SYPCR.

3.3 Halt Monitor

The halt monitor responds to an assertion of $\overline{\text{HALT}}$ on the internal bus. Refer to **5.9 Bus Error Processing** for more information. Halt monitor reset can be inhibited by the halt monitor enable (HME) bit in the SYPCR. Refer to **3.8 System Protection Registers** for a register diagram of the SYPCR.

3.4 Spurious Interrupt Monitor

During interrupt exception processing, the CPU normally acknowledges an interrupt request, arbitrates among various sources of interrupt, recognizes the highest priority source, and then acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{\text{BERR}}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{\text{BERR}}$ causes the CPU to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled. Refer to **SECTION 6 INTERRUPTS** for a comprehensive discussion of interrupts and to **5.9 Bus Error Processing** for a discussion of bus error exceptions.

3.5 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in the SYPCR. When enabled, the watchdog requires that a service sequence be written to the software service register (SWSR) on a periodic basis. If servicing does not take place, the watchdog times out and asserts the reset signal.

Perform a software watchdog service sequence as follows:

- Write $55 to the SWSR.
- Write $AA to the SWSR.

Both writes must occur in the order listed prior to time-out, but any number of instructions can be executed between the two writes.

Watchdog clock rate is affected by the software watchdog prescale (SWP) and software watchdog timing (SWT) fields in the SYPCR.

SWP determines system clock prescaling for the watchdog timer. Either no prescaling or prescaling by a factor of 512 can be selected. During reset, the state of the MODCLK pin determines the value of SWP, as shown in **Table 3-2**. System software can change the value of SWP.

Table 3-2 MODCLK Pin and SWP Bit During Reset

MODCLK	SWP	Watchdog Prescaling
0 (External Clock)	1	÷ 512
1 (Internal Clock)	0	÷ 1

The SWT field, in conjunction with the SWT bit, selects the divide ratio used to establish software watchdog time-out period. Time-out period is given by the following equations.

Time-out Period = 1/(EXTAL Frequency/Divide Ratio)

or

Time-out Period = Divide Ratio/EXTAL Frequency

Table 3-3 gives the ratio for each combination of SWP and SWT bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period can take effect.

Table 3-3 Software Watchdog Ratio

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

Figure 3-2 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

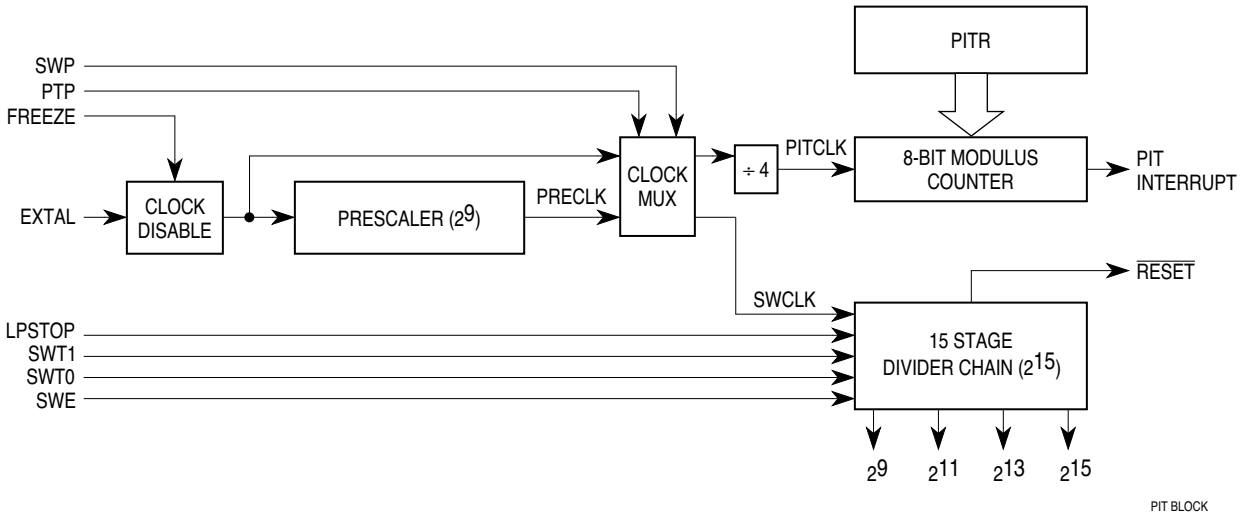


Figure 3-2 Periodic Interrupt Timer and Software Watchdog Timer

3.6 Periodic Interrupt Timer

The periodic interrupt timer allows a user to generate interrupts of specific priority at predetermined intervals. This capability is often used to schedule control system tasks that must be performed within time constraints. The timer consists of a prescaler, a modulus counter, and registers that determine interrupt timing and priority and vector assignment.

3.6.1 Prescaler and Modulus Counter

The periodic interrupt modulus counter is clocked by a signal derived from the buffered crystal oscillator (EXTAL) input pin unless an external frequency source is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the watchdog timer. Either no prescaling or prescaling by a factor of 512 can be selected. During reset, the state of the MODCLK pin (which also determines the source of the system clock) determines the value of PTP, as shown in **Table 3-4**. System software can change the PTP value.

Table 3-4 MODCLK Pin and PTP Bit During Reset

MODCLK	PTP	Watchdog Prescaling
0 (External Clock)	1	÷ 512
1 (Internal Clock)	0	÷ 1

Either clock signal (EXTAL or EXTAL ÷ 512) is divided by four before driving the modulus counter (PITCLK). The modulus counter is initialized by writing a value to the periodic timer modulus (PITM) field in the PITR. A zero value turns off the periodic timer.

When the modulus counter value reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM and counting repeats. If a new value is written to the PITM field, it is loaded into the modulus counter when the current count is completed.

Use the following equation to calculate timer period with a 32.768-kHz reference frequency (with a 4.194-MHz reference, first divide the EXTAL frequency by 128):

PIT Period = (PIT Modulus)(Prescaler value)(4)/EXTAL Frequency

For example, when a 32.768-kHz crystal reference frequency is being used to generate the system clock and the PIT prescaler is disabled (PTP = 0), PIT period is determined as follows:

PIT Period = (PIT Modulus)(1)(4)/32768
= PIT Modulus/8192

This results in a PIT period ranging from 122 μs when PITM = $01 to 31.128 ms when PITM = $FF.

With the same 32.768-kHz reference crystal and the prescaler enabled (PTP = 1),

PIT Period = (PIT Modulus)(512)(4)/32768
= PIT Modulus/16

This results in a PIT period ranging from 62.5 ms when PITM = $01 to 15.94 s when PITM = $FF.

For fast calculation of periodic timer period using a 32.768-kHz crystal reference frequency, the following equations can be used. With prescaler disabled,

PIT Period = (PIT Modulus)(122 μs)

With prescaler enabled,

PIT Period = (PIT Modulus)(62.5 ms)

To use the periodic interrupt timer as a real-time clock interrupt, rearrange the periodic timer period equation to solve for the desired count value. For a timer period of one second, for example, with the prescaler enabled,

PIT Modulus = (PIT Period)(EXTAL)(Prescaler)(4)
= (1)(32768)/(512)(4)
= 16

Therefore, the PITR should be loaded with a value of $10 (16 decimal), with the prescaler enabled, to generate interrupts at a 1-s rate.

3.6.2 Interrupt Priority and Vectoring

Interrupt priority and vectoring are determined by the values of the periodic interrupt request level (PIRQL) and periodic interrupt vector (PIV) fields in the periodic interrupt control register (PICR).

The content of PIRQL is compared to the CPU interrupt priority mask to determine whether the interrupt is recognized. **Table 3-5** shows the priority of PIRQL values. Due to SIM hardware prioritization, a PIT interrupt is serviced before an external interrupt request of the same priority. The periodic timer continues to run when the interrupt is disabled.

Table 3-5 Periodic Interrupt Priority

PIRQL	Priority Level
000	Periodic Interrupt Disabled
001	Interrupt Priority Level 1
010	Interrupt Priority Level 2
011	Interrupt Priority Level 3
100	Interrupt Priority Level 4
101	Interrupt Priority Level 5
110	Interrupt Priority Level 6
111	Interrupt Priority Level 7

The PIV field contains the periodic interrupt vector. The vector is placed on the IMB when an interrupt request is made. The method for calculating the vector address from the vector number depends on the CPU. Refer to the appropriate CPU reference manual or to the user manual for the particular MCU for this information. Refer to **SECTION 6 INTERRUPTS** of this manual for additional details concerning interrupt exception processing.

Reset value of the PIV field is $0F, which generates the uninitialized interrupt vector.

3.7 Low-Power Stop Operation

When the CPU executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, and internal clocks are disabled according to the state of the STSIM bit in the SYNCR. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low-power stop.

During low power stop, the clock input to the software watchdog timer is disabled, and the timer stops. The software watchdog begins to run again on the first rising clock edge after low power stop ends. The watchdog is not reset by low power stop; a service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction. It continues to run at the same frequency as EXTAL during LPSTOP. A periodic timer interrupt can bring the MCU out of low power stop if it has a higher priority than the interrupt mask value stored in the clock control logic when low power stop is initiated. To stop the periodic interrupt timer, the PITR must be loaded with a zero value before the LPSTOP instruction is executed.

3.8 System Protection Registers

The following registers are involved in system protection: the SIMCR, the SWSR, the PICR, the PITR, and the SYPCR. A register diagram of the SIMCR is provided in **3.1 Module Configuration and Testing**. Register diagrams of the other registers are provided in the following paragraphs.

3.8.1 Software Service Register (SWSR)

When the software watchdog is enabled, a service sequence must be written to the SWSR within a specific interval. When read, the SWSR returns all zeros. The register is shown with the read value.

SWSR — Software Service Register **$####26**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	0	0	0	0	0	0	0
RESET															
								0	0	0	0	0	0	0	0

3.8.2 Periodic Interrupt Control Register (PICR)

The PICR contains the interrupt request level and vector number for the periodic interrupt timer. PICR[10:0] can be read or written at any time. PICR[15:11] are unimplemented and always return zero.

PICR — Periodic Interrupt Control Register **$####22**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQP			PIB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

PIRQL[2:0] — Periodic Interrupt Request Level
This field determines the priority of periodic interrupt requests.

PIV[7:0] — Periodic Interrupt Vector
The bits of this field contain the periodic interrupt vector number supplied by the SIM when the CPU acknowledges an interrupt request.

3.8.3 Periodic Interrupt Timer Register (PITR)

The PITR contains the count value for the periodic timer. This register can be read or written at any time.

PITR — Periodic Interrupt Timer Register **$####24**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

PTP — Periodic Timer Prescaler Control
0 = Periodic timer clock not prescaled
1 = Periodic timer clock prescaled by a value of 512

PITM[7:0] — Periodic Interrupt Timing Modulus

This is the 8-bit timing modulus used to determine periodic interrupt rate. Use the following expression to calculate timer period.

PIT Period = [(PIT Modulus)(Prescaler value)(4)]/EXTAL Frequency

3.8.4 System Protection Register (SYPCR)

The system protection control register controls system monitor functions, software watchdog clock prescaling, and bus monitor timing. This register can be written only once following power-on or external reset, but can be read at any time.

SYPCR — System Protection Control Register **$####20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		HME	BME	BMT	
RESET:								1		0	0	0	0	0	0

SWE — Software Watchdog Enable

0 = Software watchdog disabled

1 = Software watchdog enabled

SWP — Software Watchdog Prescale

0 = Software watchdog clock not prescaled

1 = Software watchdog clock prescaled by 512

SWT[1:0] — Software Watchdog Timing

This field selects software watchdog time-out period.

Software Watchdog Ratio

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

HME — Halt Monitor Enable

0 = Disable halt monitor function

1 = Enable halt monitor function

BME — Bus Monitor External Enable

0 = Disable bus monitor function for an internal-to-external bus cycle.

1 = Enable bus monitor function for an internal-to-external bus cycle.

BMT[1:0] — Bus Monitor Timing

This field selects bus monitor time-out period.

Bus Monitor Period

BMT	Bus Monitor Time-Out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

SECTION 4 SYSTEM CLOCK

The system clock provides timing signals for the IMB modules and for an external peripheral bus. Because the MCU is a fully static design, register and memory contents are not affected when the clock rate changes. An internal phase-locked loop can synthesize the clock from a reference frequency, or the clock signal can be input from an external source.

4.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines the source of the system clock. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from the reference frequency. The reference signal is the EXTAL input to the phase-locked loop. (Refer to **4.2 Clock Synthesizer Operation**.) Frequency control bits in the clock synthesizer control register (SYNCR) determine the operating frequency.

When MODCLK is held low during reset and an external clock signal is applied to the EXTAL pin, the clock synthesizer is bypassed. SYNCR frequency control bits have no effect in this case.

CAUTION

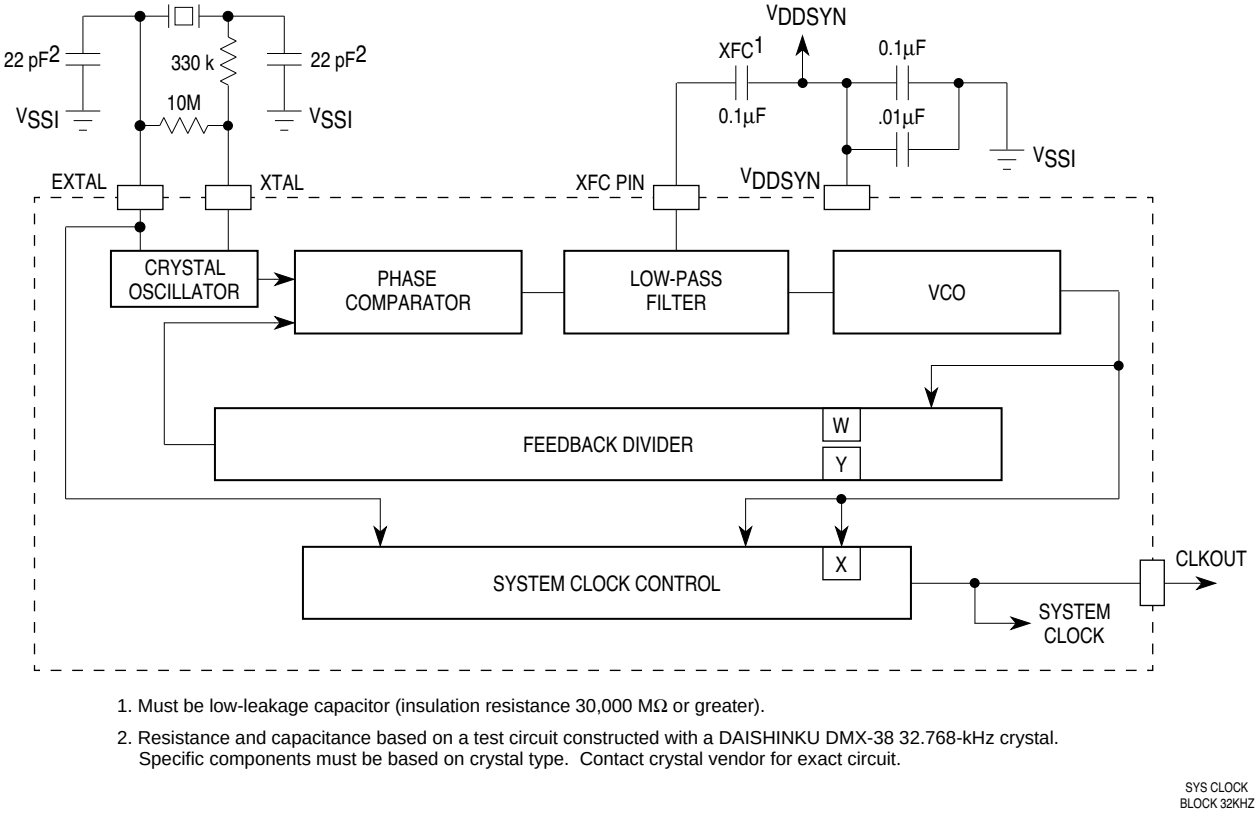
When using the MODCLK pin for I/O (PF0), be sure the external circuitry is designed so that during reset MODCLK is held at the logic level that selects the desired clock mode. After reset, the PF0 bit in the port F data register can be assigned the desired value for I/O.

4.1.1 Internal Phase-Locked Loop

To generate a clock signal with the phase-locked loop (PLL), connect a clock reference to the EXTAL pin and hold MODCLK high during reset. The clock reference can be created by connecting a crystal circuit across the EXTAL and XTAL pins or by connecting an external reference to the EXTAL pin. In the latter case, the XTAL pin must be left floating.

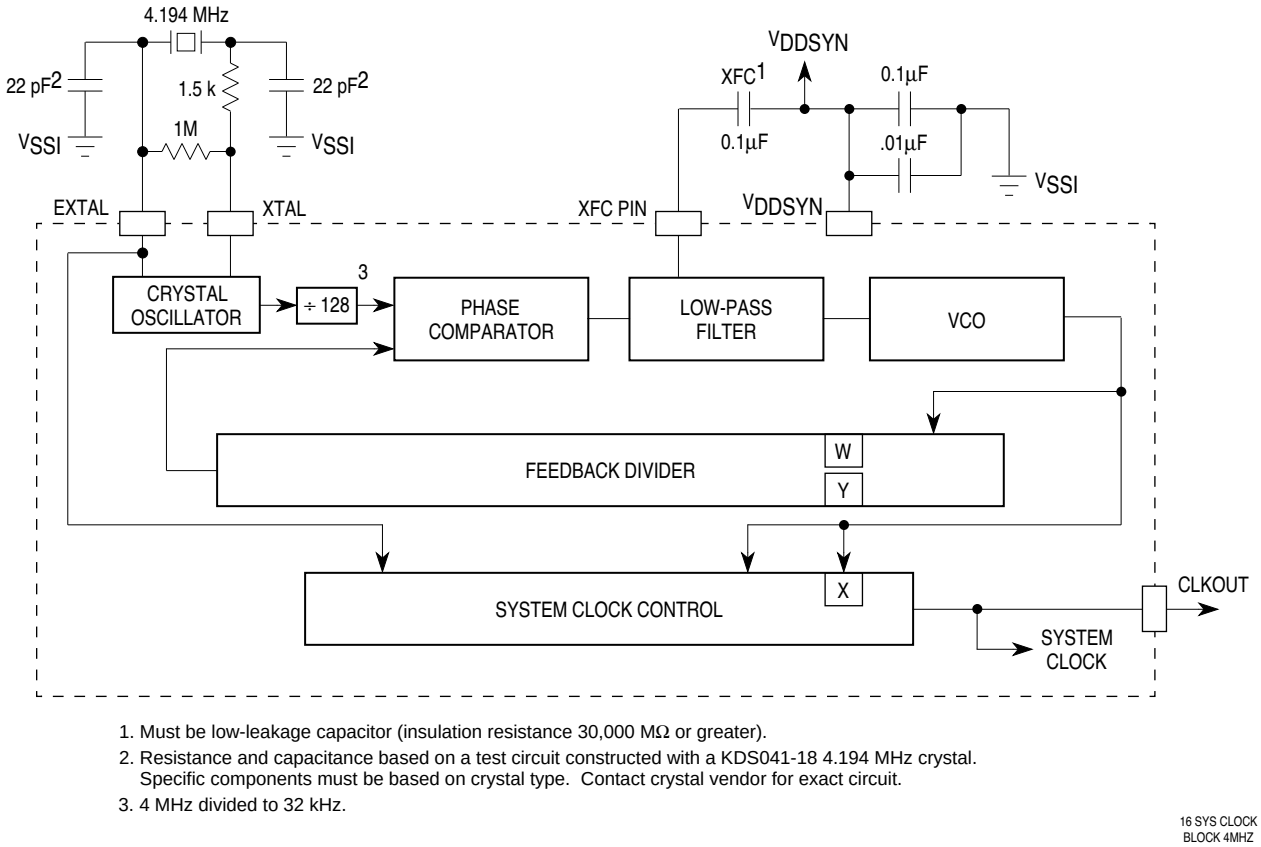
The frequency of the external reference depends on the MCU. Refer to the user's manual for the particular MCU for the range of reference frequencies that can be used with the part. Possible frequency ranges include 25 to 50 kHz and 3.2 to 6.4 MHz. For frequencies in the latter range, the reference frequency is divided internally by 128 before it is supplied to the PLL. (Refer to **4.2 Clock Synthesizer Operation**.) With a 4.194-MHz reference frequency, a signal of 32.768 kHz is provided to the PLL. Clock synthesizer specifications in **Table A-1** of **APPENDIX A ELECTRICAL CHARACTERISTICS** are based upon a 32.768-kHz or 4.194-MHz reference frequency.

Figure 4-1 is a block diagram of the clock submodule and external circuitry with a Daishinku DMX-38 32.768-kHz crystal providing the reference signal. **Figure 4-2** shows the clock submodule with a Daishinku KDS041 4.194-MHz crystal. Resistor and capacitor values depend on the crystal type. Refer to vendor documentation for recommended values. In addition, refer to **4.3.2 Crystal Tune-up Procedure**.



1. Must be low-leakage capacitor (insulation resistance 30,000 MΩ or greater).
2. Resistance and capacitance based on a test circuit constructed with a DAISHINKU DMX-38 32.768-kHz crystal. Specific components must be based on crystal type. Contact crystal vendor for exact circuit.

Figure 4-1 System Clock with 32.768-kHz Reference Crystal



1. Must be low-leakage capacitor (insulation resistance 30,000 MΩ or greater).
2. Resistance and capacitance based on a test circuit constructed with a KDS041-18 4.194 MHz crystal. Specific components must be based on crystal type. Contact crystal vendor for exact circuit.
3. 4 MHz divided to 32 kHz.

16 SYS CLOCK BLOCK 4MHZ

Figure 4-2 System Clock with 4.194-MHz Reference Crystal

4.1.2 External Clock Signal

To use an external clock source with the MCU, apply the clock signal to the EXTAL pin, and leave the XTAL pin floating. Hold the MODCLK pin low during reset, signaling the MCU to bypass the frequency synthesizer and VCO. SYNCR frequency control bits have no effect in this case.

When an external system clock signal is applied, the duty cycle of the input is critical, especially at operating frequencies close to the maximum. The relationship between the duty cycle of an external clock signal and the clock signal period is expressed as follows:

$$\text{Minimum external clock period} = \frac{\text{minimum external clock high/low time}}{50\% - \text{percentage variation of external clock input duty cycle}}$$

Refer to **Table A-3** in **APPENDIX A ELECTRICAL CHARACTERISTICS** for the minimum external clock high/low time. The external system clock signal frequency must be less than or equal to the maximum specified system clock frequency.

4.2 Clock Synthesizer Operation

When MODCLK is held high during reset, a voltage-controlled oscillator (VCO) in the phase-locked loop generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is the reference signal. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is equal to reference frequency (or reference frequency divided by 128, if the reference frequency is a 4.194-MHz crystal). Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever a comparator input changes, the synthesizer must relock. Lock status is shown by the SLOCK bit in the SYNCR. The MCU does not come out of the reset state until the synthesizer locks.

4.3 External Circuit Design

The following paragraphs discuss design issues relating to the oscillator circuit and external filter circuit.

4.3.1 Conditioning the XTAL and EXTAL Pins

As in all crystal oscillator designs, all leads should be kept as short as possible. It is also good practice to route V_{SSI} paths as shown in **Figure 4-3**. These paths isolate the oscillator input from the output and the oscillator from adjacent circuitry, only adding capacitance in parallel capacitors C1 and C2.

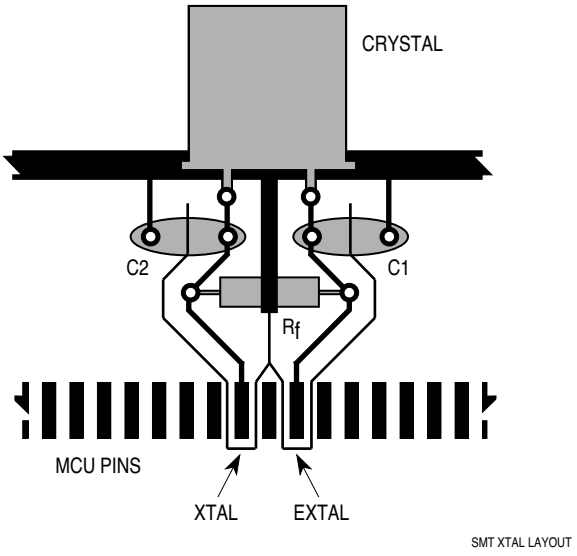


Figure 4-3 Crystal Layout Example

4.3.2 Crystal Tune-up Procedure

The following tune-up procedure applies to crystals with frequencies below 1 MHz. At higher crystal frequencies, because R_S (the resistance between the external oscillator and the XTAL pin) is normally 0 Ω, this procedure is not needed. For more specific tuning instructions, contact the crystal manufacturer.

The value of R_S can be determined experimentally by using the final PCB and an MCU of the exact type that will be used in the final application. The MCU need not have the final software in place. Because of the number of variables involved, use components with the exact properties of those that will be used in production. For example, do not use a ceramic-packaged MCU prototype for the experiment when a plastic-packaged MCU will be used in production. An emulator version of the part will also have slightly different electrical properties from the masked ROM version of the same part.

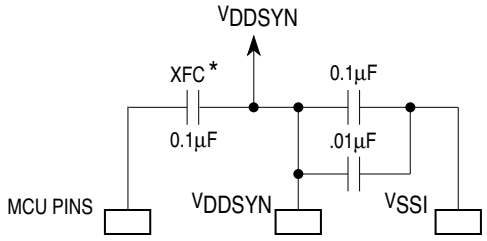
To determine the optimum value for R_S, observe the operating current (I_{DDSYN}) of the MCU as a function of R_S. The MCU should have only V_{DDSYN} powered throughout this procedure because operating current variations when the MCU is running are much greater than the current variations due to varying R_S. Normally, a dip in current will occur. This dip is not sharp as in many LC circuits, but is rather very broad. As the shape of this curve suggests, the exact value of R_S is not critical.

Finally, verify that the maximum operating supply voltage does not overdrive the crystal. Observe the output frequency as a function of V_{DDSYN} at the buffered CLKOUT output with the device powered up in reset. Under proper operating conditions, the frequency should increase a few parts per million as supply voltage increases. If the crystal is overdriven, an increase in supply voltage will cause a decrease in frequency, or the frequency will become unstable. If frequency problems arise, supply voltage must be decreased, or the values of R_S and the two capacitors should be increased to reduce the crystal drive.

4.3.3 Conditioning the XFC, V_{DDSYN}, and V_{SSI} Pins

The XFC, V_{DDSYN}, and V_{SSI} input lines should be made as free of noise as possible. Noise on these lines will cause frequency shifts in CLKOUT. Guard ring the XFC line with V_{DDSYN}, and guard ring V_{DDSYN} with V_{SSI}, wherever possible. The XFC filter capacitor and the V_{DDSYN} bypass capacitors should be kept as close to the XFC and V_{DDSYN} pins as possible, with no digital logic coupling to either XFC or V_{DDSYN}. The ground for the V_{DDSYN} bypass capacitors should be tied directly to the V_{SSI} ground plane. If possible, route V_{DDSYN} and V_{SSI} as separate supply runs or planes. V_{DDSYN} may require an inductive or resistive filter to control supply noise.

Figure 4-4 shows the external circuit for the XFC and V_{DDSYN} pins.



* Must be low-leakage capacitor (insulation resistance 30,000 MΩ or greater).

XFC VDDSYN CONN

Figure 4-4 Conditioning the XFC and V_{DDSYN} Pins

A V_{DDSYN} resistive filter would consist of a 100- to 500-ohm resistor from V_{DD} to V_{DDSYN} and a 0.1-µF bypass capacitor from V_{DDSYN} to V_{SSI}. The proper values for the resistor and capacitor can be determined by examining the frequency of the V_{DDSYN} noise. The RC time constant needs to be large enough to filter the supply noise. An inductive filter would replace the resistor with an inductor.

The low-pass filter requires an external low-leakage capacitor (e.g., polypropylene), typically 0.1 µF with an insulation resistance specification of 30,000 MΩ or greater, connected between the XFC and V_{DDSYN} pins.

4.4 System Clock Frequency Control

When the clock synthesizer is used, bits [15:8] of the synthesizer control register (SYNCR) determine the operating frequency. The W bit controls a prescaler tap in the synthesizer feedback loop. The Y field determines the modulus for a modulo down counter. Y contains a value from 0 to 63; input to the modulo counter is divided by a value of Y + 1. Both W and Y affect the value of the feedback divider input to the VCO. Consequently, changing either of these values results in a time delay before the VCO locks in to the new frequency.

The SYNCR X bit controls a divide-by-two prescaler that is not in the synthesizer feedback loop. Setting X doubles the clock speed without changing the VCO speed. Consequently, there is no VCO relock delay.

In order for the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified in **Table A-1** of **APPENDIX A ELECTRICAL CHARACTERISTICS**. As **Table 4-1** and **Table 4-2** indicate, with either a 32.768-kHz or a 4.194-MHz reference frequency, W and X must not both be set when the count modulus is greater than Y = %001111.

4.4.1 Frequency Control with a Reference Frequency of 25–50 kHz

With a reference frequency between 25 and 50 kHz (such as a 32.768-kHz crystal), clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}[4(Y + 1)(2^{2W + X})]$$

The internal VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

The reset state of SYNCR ($3F00) produces a modulus-64 count. The W and X bits are both zero, so that system frequency is 256 times the reference frequency.

4.4.2 Frequency Control with a Reference Frequency of 3.2 – 6.4 MHz

With a reference frequency between 3.2 and 6.4 MHz (such as a 4.194-MHz crystal), the reference frequency is divided by 128 before it is passed to the PLL system. The system clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = \frac{F_{REFERENCE}}{128}[4(Y + 1)(2^{2W + X})]$$

The internal VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

The reset state of SYNCR ($3F00) produces a modulus-64 count.

4.4.3 Avoiding Frequency Overshoot

When the W and Y fields in the SYNCR are changed to increase the operating frequency, a frequency overshoot of up to 30% can occur. This overshoot can be avoided by following these procedures:

1. Determine the values for the W and Y fields which will result in the desired frequency when the X bit is set.
2. With the X bit *cleared*, write these values for the W and Y fields to the SYNCR.
3. After the VCO locks, *set* the X bit in the SYNCR. This changes the clock frequency to the desired frequency.

For example, follow these procedures to change the clock frequency from 8 to 13 MHz after reset using a 32.768-kHz crystal:

1. Determine the values for the W and Y fields: W = %0, Y = %110100
2. Clear the X bit.
3. Write the values to the SYNCR: W = %0, Y = %110100
4. When the VCO locks, set the X bit.

4.4.4 Frequency Control Tables

Table 4-1 shows how to compute system clock frequency for various combinations of SYNCR bits. **Table 4-2** shows actual clock frequencies for the same combinations of SYNCR bits. The frequencies in **Table 4-2** are valid for typical values of 32.768-kHz and 4.194-MHz crystal references.

Table 4-1 Clock Control Multipliers

To obtain the clock frequency, find the counter modulus in the leftmost column, then multiply the reference frequency by the value in the appropriate prescaler cell. (For 4.194-MHz reference crystals, first divide by 128.) Shaded cells contain values that exceed specifications.

Modulus	Prescalers			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
000000	4	8	16	32
000001	8	16	32	64
000010	12	24	48	96
000011	16	32	64	128
000100	20	40	80	160
000101	24	48	96	192
000110	28	56	112	224
000111	32	64	128	256
001000	36	72	144	288
001001	40	80	160	320
001010	44	88	176	352
001011	48	96	192	384
001100	52	104	208	416
001101	56	112	224	448
001110	60	120	240	480
001111	64	128	256	512
010000	68	136	272	544
010001	72	144	288	576
010010	76	152	304	608
010011	80	160	320	640
010100	84	168	336	672
010101	88	176	352	704
010110	92	184	368	736
010111	96	192	384	768
011000	100	200	400	800
011001	104	208	416	832
011010	108	216	432	864
011011	112	224	448	896
011100	116	232	464	928
011101	120	240	480	960
011110	124	248	496	992
011111	128	256	512	1024
100000	132	264	528	1056
100001	136	272	544	1088
100010	140	280	560	1120
100011	144	288	576	1152
100100	148	296	592	1184
100101	152	304	608	1216
100110	156	312	624	1248
100111	160	320	640	1280
101000	164	328	656	1312
101001	168	336	672	1344
101010	172	344	688	1376
101011	176	352	704	1408

Table 4-1 Clock Control Multipliers (Continued)

To obtain the clock frequency, find the counter modulus in the leftmost column, then multiply the reference frequency by the value in the appropriate prescaler cell. (For 4.194-MHz reference crystals, first divide by 128.) Shaded cells contain values that exceed specifications.

Modulus	Prescalers			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
101100	180	360	720	1440
101101	184	368	736	1472
101110	188	376	752	1504
101111	192	384	768	1536
110000	196	392	784	1568
110001	200	400	800	1600
110010	204	408	816	1632
110011	208	416	832	1664
110100	212	424	848	1696
110101	216	432	864	1728
110110	220	440	880	1760
110111	224	448	896	1792
111000	228	456	912	1824
111001	232	464	928	1856
111010	236	472	944	1888
111011	240	480	960	1920
111100	244	488	976	1952
111101	248	496	992	1984
111110	252	504	1008	2016
111111	256	512	1024	2048

j

Table 4-2 System Frequencies from Typical 32.768-kHz or 4.194-MHz Reference

To obtain the clock frequency, find the counter modulus in the leftmost column, then look in the appropriate prescaler cell. Values are in kilohertz. Shaded cells represent values that exceed maximum system frequency specification.

Modulus	Prescaler			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
000000	131 kHz	262 kHz	524 kHz	1049 kHz
000001	262	524	1049	2097
000010	393	786	1573	3146
000011	524	1049	2097	4194
000100	655	1311	2621	5243
000101	786	1573	3146	6291
000110	918	1835	3670	7340
000111	1049	2097	4194	8389
001000	1180	2359	4719	9437
001001	1311	2621	5243	10486
001010	1442	2884	5767	11534
001011	1573	3146	6291	12583
001100	1704	3408	6816	13631
001101	1835	3670	7340	14680
001110	1966	3932	7864	15729
001111	2097	4194	8389	16777
010000	2228	4456	8913	17826
010001	2359	4719	9437	18874
010010	2490	4981	9961	19923
010011	2621	5243	10486	20972
010100	2753	5505	11010	22020
010101	2884	5767	11534	23069
010110	3015	6029	12059	24117
010111	3146	6291	12583	25166
011000	3277	6554	13107	26214
011001	3408	6816	13631	27263
011010	3539	7078	14156	28312
011011	3670	7340	14680	29360
011100	3801	7602	15204	30409
011101	3932	7864	15729	31457
011110	4063	8126	16253	32506
011111	4194	8389	16777	33554
100000	4325 kHz	8651 kHz	17302 kHz	34603 kHz
100001	4456	8913	17826	35652
100010	4588	9175	18350	36700
100011	4719	9437	18874	37749
100100	4850	9699	19399	38797
100101	4981	9961	19923	39846
100110	5112	10224	20447	40894
100111	5243	10486	20972	41943
101000	5374	10748	21496	42992
101001	5505	11010	22020	44040
101010	5636	11272	22544	45089

Table 4-2 System Frequencies from Typical 32.768-kHz or 4.194-MHz Reference (Continued)

To obtain the clock frequency, find the counter modulus in the leftmost column, then look in the appropriate prescaler cell. Values are in kilohertz. Shaded cells represent values that exceed maximum system frequency specification.

Modulus	Prescaler			
Y	[W:X] = 00	[W:X] = 01	[W:X] = 10	[W:X] = 11
101011	5767	11534	23069	46137
101100	5898	11796	23593	47186
101101	6029	12059	24117	48234
101110	6160	12321	24642	49283
101111	6291	12583	25166	50332
110000	6423	12845	25690	51380
110001	6554	13107	26214	52428
110010	6685	13369	26739	53477
110011	6816	13631	27263	54526
110100	6947	13894	27787	55575
110101	7078	14156	28312	56623
110110	7209	14418	28836	57672
110111	7340	14680	29360	58720
111000	7471	14942	2988	59769
111001	7602	15204	30409	60817
111010	7733	15466	30933	61866
111011	7864	15729	31457	62915
111100	7995	15991	31982	63963
111101	8126	16253	32506	65011
111110	8258	16515	33030	66060
111111	8389	16777	33554	67109

4.5 External Bus Clock

The state of the external clock division bit (EDIV) in the SYNCR determines the clock rate for the external bus clock signal (ECLK) available on pin ADDR23. ECLK is a bus clock for MC6800 devices and peripherals. ECLK frequency can be set to the system clock frequency divided by eight or system clock frequency divided by sixteen. The clock is enabled by the $\overline{CS10}$ field in chip select pin assignment register 1 (CSPAR1). The operation of the external bus clock during low-power stop is described below. Refer to **SECTION 7 CHIP SELECTS** for more information on the external bus clock.

4.6 Low-Power Stop Operation

To reduce overall microcontroller power consumption to a minimum, the CPU can execute the LPSTOP instruction, which causes the SIM to turn off the system clock to most of the MCU.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SIM brings the MCU out of low-power operation when either an interrupt of higher priority than the stored mask or a reset occurs. Refer to **5.8.2 LPSTOP Broadcast Cycle** for more information.

During low-power stop, SIM clock control logic and the SIM clock signal (SIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ pins are clocked by SIMCLK. The SIM can also continue to generate the CLKOUT signal while in low-power mode.

The STSIM (stop mode SIM clock) and STEXT (stop mode external clock) bits in the SYNCR determine clock operation during low-power stop. **Table 4-3** summarizes the sources of SIMCLK and CLKOUT for different combinations of clock mode, STSIM, and STEXT during low-power stop and normal operation. MODCLK value is the logic level on the MODCLK pin during the last reset prior to LPSTOP execution. Any clock in the off state is held low.

Table 4-3 Clock Control

Mode	Pins		SYNCR Bits		Clock Source	
LPSTOP	MODCLK	EXTAL	STSIM	STEXT	SIMCLK	CLKOUT
No	0	External Clock	X	X	External Clock	External Clock
Yes	0	External Clock	0	0	External Clock	Off
Yes	0	External Clock	0	1	External Clock	External Clock
Yes	0	External Clock	1	0	External Clock	Off
Yes	0	External Clock	1	1	External Clock	External Clock
No	1	Crystal/ Reference	X	X	VCO	VCO
Yes	1	Crystal/ Reference	0	0	Crystal/ Reference	Off
Yes	1	Crystal/ Reference	0	1	Crystal/ Reference	Crystal/ Reference
Yes	1	Crystal/ Reference	1	0	VCO	Off
Yes	1	Crystal/ Reference	1	1	VCO	VCO

4.7 Loss of Reference Signal

The state of the reset enable (RSTEN) bit in the SYNCR determines what happens when clock logic detects a reference failure.

When RSTEN is cleared (the default state out of reset), the clock synthesizer is forced into an operating condition referred to as limp mode. Limp mode frequency varies from device to device, but maximum limp frequency does not exceed one half maximum system when X = 0, or maximum system clock frequency when X = 1.

When RSTEN is set, the SIM generates a reset when clock logic detects a reference failure. Refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION** for more information on reset procedures.

4.8 Clock Synthesizer Control Register (SYNCR)

The SYNCR determines system clock operating frequency and mode of operation. Bits with reset states labeled “U” are unaffected by reset.

SYNCR — Clock Synthesizer Control Register **$####04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	U	U	0	0	0

W — Frequency Control (VCO)
- 0 = Base VCO frequency
- 1 = VCO frequency multiplied by four

Refer to **4.4 System Clock Frequency Control** for additional information.

X — Frequency Control Bit (Prescale)
- 0 = Base system clock frequency
- 1 = System clock frequency multiplied by two

Refer to **4.4 System Clock Frequency Control** for additional information.

Y[5:0] — Frequency Control (Counter)

The Y field is the initial value for the modulus 64 down counter in the synthesizer feedback loop. Values range from 0 to 63. Refer to **4.4 System Clock Frequency Control** for additional information.

EDIV — ECLK Divide Rate
- 0 = ECLK is system clock divided by 8.
- 1 = ECLK is system clock divided by 16.

Refer to **4.5 External Bus Clock** for additional information.

SLIMP — Limp Mode Status
- 0 = MCU is operating normally.
- 1 = Loss of reference signal — MCU operating in limp mode.

Refer to **4.7 Loss of Reference Signal** for additional information.

SLOCK — Synthesizer Lock
- 0 = VCO is enabled, but has not locked.
- 1 = VCO has locked on the desired frequency or system clock is external.

RSTEN — Reset Enable
- 0 = Loss of clock causes the MCU to operate in limp mode.
- 1 = Loss of clock causes system reset.

Refer to **4.7 Loss of Reference Signal** for additional information.

STSIM — Stop Mode SIM Clock
- 0 = SIM clock driven by the external reference signal and the VCO is turned off during low-power stop.
- 1 = SIM clock driven by VCO during low-power stop.

This bit has an effect only if the PLL is configured to supply the clock signal (MODCLK held high during reset). **4.6 Low-Power Stop Operation** has additional information.

STEXT — Stop Mode External Clock

0 = CLKOUT held low during low-power stop.

1 = CLKOUT driven from SIM clock during low-power stop.

Refer to **4.6 Low-Power Stop Operation** for additional information.

SECTION 5 EXTERNAL BUS INTERFACE

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. The external bus has 24 address lines and 16 data lines.

NOTE

On 16-bit MCUs, ADDR[19:0] are normal address outputs, and ADDR[23:20] follow the output state of ADDR19.

A three-line handshaking interface performs external bus arbitration. The interface supports byte, word, and long-word transfers. The EBI performs dynamic sizing for data accesses.

The maximum number of bits transferred during an access is referred to as port width. Widths of eight and sixteen bits can be accessed by means of asynchronous bus cycles controlled by the size (SIZ0 and SIZ1) and the data and size acknowledge ($\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$) signals. Multiple bus cycles may be required for a dynamically-sized transfer. Refer to **5.3 Dynamic Bus Sizing** for more information on data alignment and port width. **5.7 System Interfacing Examples** provides example system configurations for different port widths.

NOTE

Some MCUs with reduced pin-count SIMs do not include a $\overline{\text{DSACK0}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** for details on handshaking with these MCUs.

The SIM contains 12 chip-select circuits that can simplify the interface to memory and peripherals. These chip selects are described in **SECTION 7 CHIP SELECTS**. The discussion of the EBI in this section is useful both as a background for understanding chip-select operation and as a guide to interfacing to external devices without the use of SIM chip selects.

NOTE

MCUs with reduced pin count SIMs may contain fewer than 12 chip-select pins. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** and to the user's manual for the particular MCU for details.

5.1 Bus Signal Descriptions

The address bus provides addressing information to external devices. The data bus transfers 8-bit and 16-bit data between the MCU and external devices. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data.

Control signals indicate the availability of an address on the bus ($\overline{AS}$), validity of data on the bus ($\overline{DS}$), the address space where the bus cycle is to occur (FC[2:0]), the size of the transfer (SIZ[1:0]), and the type of cycle (R/$\overline{W}$). External devices decode these signals and respond by transferring data and terminating the bus cycle.

5.1.1 Address Bus

Bus signals ADDR[23:0] define the address of the most significant byte to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{AS}$ is asserted.

5.1.2 Address Strobe

Address strobe ($\overline{AS}$) is a timing signal that indicates the validity of an address on the address bus and of many control signals. It is asserted one-half clock after the beginning of a bus cycle.

5.1.3 Data Bus

Bus signals DATA[15:0] comprise a bidirectional, nonmultiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer 8 or 16 bits of data in one bus cycle. During a read cycle, the MCU latches the data on the last falling clock edge of the bus cycle. During a write cycle, the MCU places the data on the data bus one half clock cycle after $\overline{AS}$ is asserted.

5.1.4 Data Strobe

For a read cycle, the MCU asserts data strobe ($\overline{DS}$) to signal an external device to place data on the bus. $\overline{DS}$ is asserted at the same time as $\overline{AS}$ during a read cycle. For a write cycle, $\overline{DS}$ signals an external device that data on the bus is valid. The MCU asserts $\overline{DS}$ one full clock cycle after the assertion of $\overline{AS}$ during a write cycle.

5.1.5 Read/Write Signal

The read/write (R/$\overline{W}$) signal determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and is valid while $\overline{AS}$ is asserted. R/$\overline{W}$ changes state only when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for two consecutive write cycles.

5.1.6 Size Signals

The size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe ($\overline{AS}$) is asserted. Refer to **5.3 Dynamic Bus Sizing** for information on SIZ1 and SIZ0 encoding and on using the SIZ0 and SIZ1 pins in dynamic bus allocation.

5.1.7 Function Codes

The CPU generates function code signals FC[2:0]. The function codes can be considered address extensions that designate which of eight external address spaces is accessed during a bus cycle. Refer to **5.6 Function Codes and Memory Usage** for information on function code signal encoding and usage.

5.1.8 Data and Size Acknowledge Signals

During normal (asynchronous) bus transfers, external devices assert one of the data and size acknowledge signals ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$) to indicate port width to the MCU. During a read cycle, these signals also tell the MCU to terminate the bus cycle and to latch data. During a write cycle, they indicate that an external device has successfully stored data and that the cycle may terminate. Refer to **5.3 Dynamic Bus Sizing** for information on $\overline{\text{DSACK}}$ encoding and dynamic bus sizing.

$\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ can also be supplied internally by chip-select logic. Refer to **SECTION 7 CHIP SELECTS** for more information on internally generated $\overline{\text{DSACK}}$[1:0] signals.

The designation $\overline{\text{DSACK}}$ is used in this manual as a generic reference to one or both of these signals.

NOTE

Some MCUs with reduced pin-count SIMs do not include a $\overline{\text{DSACK0}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** for details on handshaking with these MCUs.

5.1.9 Bus Error Signal

The bus error ($\overline{\text{BERR}}$) signal is asserted in the absence of $\overline{\text{DSACK}}$ assertion to indicate a bus error condition. It can also be asserted in conjunction with $\overline{\text{DSACK}}$ to indicate a bus error condition, provided it meets the appropriate timing requirements. Refer to **5.9 Bus Error Processing** for more information.

An external bus master must provide its own $\overline{\text{BERR}}$ generation and drive the $\overline{\text{BERR}}$ pin. Refer to **5.10 Bus Arbitration** for more information.

5.1.10 Halt Signal

An external device can assert the halt signal ($\overline{\text{HALT}}$) to cause single bus cycle operation. Additionally, on certain MCUs $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ can be asserted simultaneously to indicate a retry termination. **5.9 Bus Error Processing** provides additional information on the $\overline{\text{HALT}}$ signal.

5.1.11 Autovector Signal

The autovector signal ($\overline{\text{AVEC}}$) can be used to terminate external interrupt acknowledge cycles resulting from interrupts from external $\overline{\text{IRQ}}$ pins. Assertion of $\overline{\text{AVEC}}$ causes the CPU to generate vector numbers to locate an interrupt handler routine. Refer to **SECTION 6 INTERRUPTS** for more information. for external interrupt requests can also be supplied internally by chip-select logic. Refer to **SECTION 7 CHIP SELECTS** for more information.

5.2 External Bus Cycle Overview

Data transfer operations with external devices consist of one or more external bus cycles. This subsection describes individual bus cycles during each of these transfer operations. **5.3 Dynamic Bus Sizing** explains the use of the $\overline{\text{DSACK}}$[1:0] and SIZ[1:0] signals and the placement of operands on the data bus. **5.5 Operand Transfer Cases** provides additional details of bus cycle operation for each combination of operand size and port width.

External bus cycles are normally asynchronous, using handshaking between the MCU and external peripherals to indicate transfer size and the availability of data. These accesses typically require a minimum of three system clock cycles. (Two-clock accesses are possible with the chip select fast termination option.) Internal microcontroller modules, in contrast, are typically accessed in two system clock cycles.

The SIM contains 12 chip-select circuits. These on-chip circuits decode EBI address lines and control signals (R/$\overline{\text{W}}$, SIZ[1:0], $\overline{\text{AS}}$, $\overline{\text{DS}}$, and FC[2:0]), reducing the need for external glue logic. Chip-select circuits can also generate the following types of external bus cycles:

- Bus cycles with internally generated $\overline{\text{DSACK}}$ signals.
- Fast termination (two-clock) cycles.
- Bus cycles that are synchronous to the ECLK signal, rather than asynchronous. (Asynchronous cycles are always terminated with $\overline{\text{DSACK}}$ signals.)

These features are discussed in **SECTION 7 CHIP SELECTS**. The paragraphs that follow describe asynchronous external bus cycles that require a minimum of three clock cycles, decode address and control signals externally, and use externally generated $\overline{\text{DSACK}}$ signals.

Bus cycles normally occur in user or supervisor space. Bus cycles that occur in CPU space, which follow most of the protocol of regular external bus cycles, are described in **5.8 CPU Space Cycles**. The interrupt acknowledge cycle, a type of CPU space cycle, is described in **SECTION 6 INTERRUPTS**.

5.2.1 Bus Cycle Operation

To initiate a transfer, the MCU asserts the appropriate address and SIZ[1:0] signals. The SIZ[1:0] signals and ADDR0 are externally decoded to select the active portion of the data bus. (Refer to **5.3 Dynamic Bus Sizing**.) The remaining address lines are decoded to select the peripheral and address within the peripheral.

When $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ are valid, a peripheral device either places data on the bus during a read cycle or latches data from the bus during a write cycle, then asserts the appropriate $\overline{\text{DSACK}}$ signal to indicate port size.

$\overline{\text{DSACK}}$ signals can be asserted before the data from a peripheral device is valid on a read cycle. To ensure that valid data is latched into the MCU, a maximum period between $\overline{\text{DSACK}}$ assertion and $\overline{\text{DS}}$ assertion is specified. (Refer to **Table A-3** in **APPENDIX A ELECTRICAL CHARACTERISTICS**.)

There is no specified maximum for the period between $\overline{AS}$ assertion and $\overline{DSACK}$ assertion. Although the MCU can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{DSACK}$, the MCU inserts wait cycles in clock period increments until either $\overline{DSACK}$ signal goes low.

NOTE

The SIM bus monitor asserts $\overline{BERR}$ when response time exceeds a predetermined limit. Bus monitor period is determined by the BMT field in the system protection control register (SYPCR). The bus monitor cannot be disabled; maximum monitor period is 64 system clock cycles.

If no peripheral responds to an access or if an access is invalid, external logic should assert the $\overline{BERR}$ signal to abort the bus cycle. If $\overline{BERR}$ or bus termination signals are not asserted within the specified bus monitor time-out period, the bus monitor, if enabled for internal to external bus cycles, terminates the cycle.

5.2.2 Synchronization to CLKOUT

All external asynchronous input signals must be synchronized to the MCU clock before being acted upon. The CLKOUT (system clock output) signal enables external devices to synchronize $\overline{DSACK}$ and other signals with the MCU system clock.

For all inputs, the MCU latches the level of the input during a sample window around the falling edge of CLKOUT. (Refer to **Figure 5-1**.) 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 MCU is not predictable; however, the MCU always resolves the latched level to a logic high or low before using it.

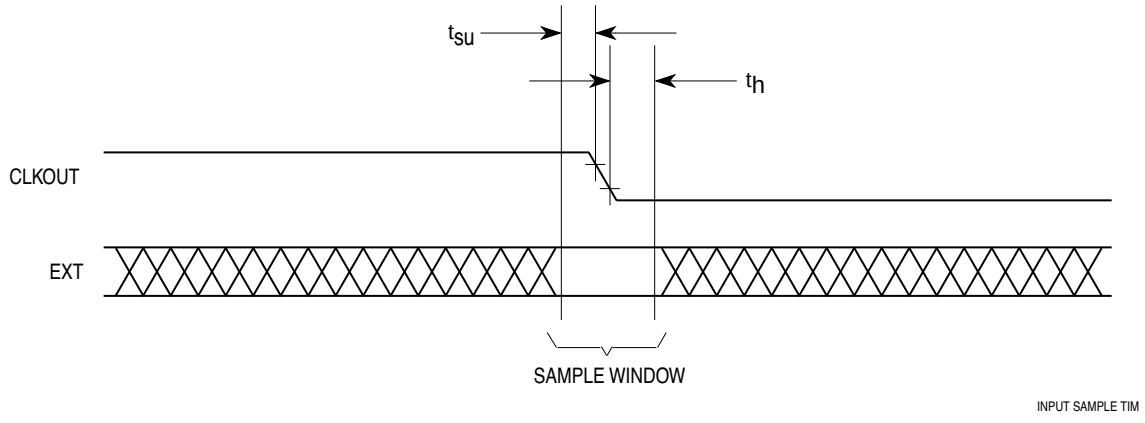


Figure 5-1 Input Sample Window

When the specifications for asynchronous setup and hold time are met, the MCU is guaranteed to recognize the appropriate signal on a specific edge of the CLKOUT signal. For a read cycle, when assertion of $\overline{\text{DSACK}}$ is recognized on a particular falling edge of the clock, valid data is latched into the MCU on the next falling clock edge, provided that the data meets the data setup time.

When a system asserts $\overline{\text{DSACK}}$ for the required window around the falling edge of state 2 (see **5.4.1 Read Cycles** and **5.4.2 Write Cycles**) and obeys the bus protocol by maintaining $\overline{\text{DSACK}}$ until and throughout the clock edge that negates $\overline{\text{AS}}$, no wait states are inserted. The bus cycle runs at the maximum speed of three clocks per cycle.

5.3 Dynamic Bus Sizing

Dynamic bus sizing allows the MCU to move byte, word, or long-word data to either an 8-bit or 16-bit memory or peripheral system.

The MCU and target device use the SIZ[1:0], $\overline{\text{DSACK}}$[1:0], and ADDR0 signals to indicate the operand and port size and operand alignment (even or odd address). Based on this information, the MCU places the data on or reads the data from the appropriate byte or bytes of the data bus.

5.3.1 Size Signal Encoding

When the MCU starts to perform an access to an external device, it uses the SIZ[1:0] pins to inform the external device of the size of the intended data transfer. **Table 5-1** shows the encodings for the size signals.

Table 5-1 Size Signal Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	3 Byte
0	0	Long Word

If a transfer operation requires more than one bus cycle, the MCU automatically updates the SIZ[1:0] pins at the start of each cycle to reflect the number of remaining bytes to be transferred. For example, a word write to an 8-bit port requires two bus cycles. During the first cycle, the MCU drives 1 and 0 on the SIZ1 and SIZ0 pins, respectively. During the second cycle, the MCU drives 0 and 1 on these pins.

5.3.2 Data and Size Acknowledge Signal Encoding

The external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{\text{DSACK}}$ inputs. If the processor attempts to write a word to an 8-bit port, for example, the $\overline{\text{DSACK}}$ pins inform the processor that the port is only 8 bits wide, and the processor initiates a second bus cycle to write the remaining byte.

Table 5-2 shows $\overline{\text{DSACK}}$ encodings.

Table 5-2 $\overline{\text{DSACK}}$ Signal Encodings

$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	Result
1	1	Insert Wait States — in Current Bus Cycle
1	0	Complete Cycle — Data Bus Port Size is 8 Bits
0	1	Complete Cycle — Data Bus Port Size is 16 Bits
0	0	Reserved (Defaults to 16-bit Port)

NOTE

Some MCUs with reduced pin count SIMs do not contain a $\overline{\text{DSACK0}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** for details on handshaking with these MCUs.

Chip-select logic can generate data and size acknowledge signals for an external device. Refer to **SECTION 7 CHIP SELECTS** for details on generating $\overline{\text{DSACK}}$ signals with chip selects.

5.3.3 Operand Alignment

Operand alignment on the data bus is determined by the ADDR0, SIZ[1:0], and $\overline{\text{DSACK[1:0]}}$ signals. To understand operand alignment more fully, refer to the individual cases described in **5.5 Operand Transfer Cases**. The following paragraphs summarize the procedure the EBI follows for aligning operands.

The EBI dynamically determines the port size of the target device during each bus cycle. The EBI begins each bus cycle by assuming a 16-bit port and then determines the actual state of affairs based on the $\overline{\text{DSACK}}$ signals returned by the target device.

During a write cycle, the EBI routes the bytes of the operand to the bytes of the data bus so that both 8- and 16-bit devices can retrieve the data. The EBI signals the location of the data to the target device through the SIZ[1:0] and ADDR0 pins.

This scheme implies that in some cases both bytes of the data bus are copies of the same byte of the operand and that in some cases one of the bytes of the data bus will be unused by the target device.

During a read cycle, the EBI uses the encoding of the $\overline{\text{DSACK[1:0]}}$ pins to determine the location of the valid data on the data bus. This method implies that in some cases, the EBI must reroute a byte of data internally to the operand latch, and that sometimes a byte of data is ignored.

Table 5-3 indicates the location of valid data for each combination of operand size, port size, and even or odd address for read and write cycles.

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must be connected to data bus bits [15:0], and an 8-bit port must be connected to data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand via a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word on a following bus cycle.

5.3.4 Misaligned Operands

For external bus cycles, the basic operand size of both the CPU16 and CPU32 processors is 16 bits. An operand is misaligned when it overlaps a word boundary. When ADDR0 = 0, indicating an even address, the address is on a word and byte boundary. When ADDR0 = 1, indicating an odd address, the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address.

NOTE

The CPU16 can perform misaligned word transfers. This capability makes it compatible with the M68HC11 CPU. The CPU16 treats misaligned long-word transfers as two misaligned word transfers. The CPU32, however, does not support misaligned word transfers. Refer to the user's manual for the specific MCU or CPU for additional information.

5.4 Data Transfer Operations

The following paragraphs provide detailed descriptions of read and write bus cycles. Following these descriptions is a discussion of the CPU32 indivisible read-modify-write operation, which consists of one or more read cycles followed by one or more write cycles.

5.4.1 Read Cycles

During a read operation, the MCU transfers data from an external memory or peripheral device. A read operation consists of one or more read cycles. If the instruction specifies a long-word or word operation, the MCU attempts to read two bytes per cycle. For a byte operation, the MCU reads one byte. The portion of the data bus from which each byte is read depends on operand size, peripheral address, and peripheral port size. Refer to **5.3 Dynamic Bus Sizing** for more information. **Figure 5-2** is a flow chart of a word read cycle.

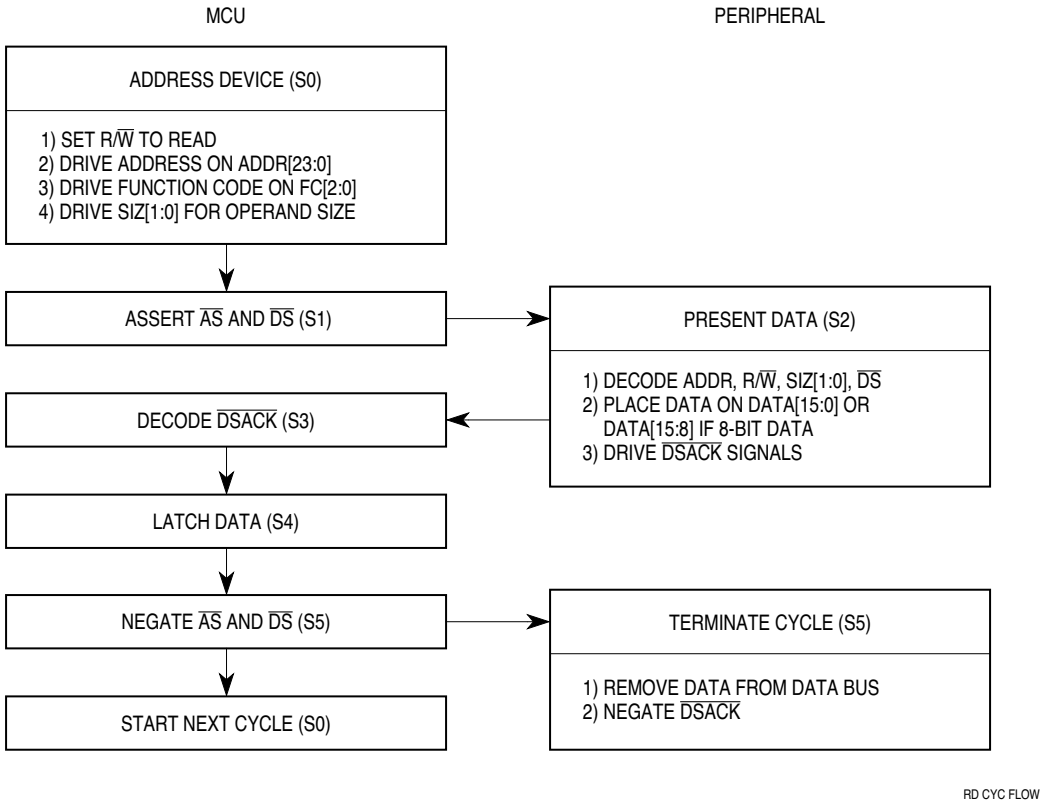


Figure 5-2 Read Cycle Flowchart

A read cycle consists of the following states. The designation “state” refers to the logic level of the clock signal, and does not correspond to any implemented machine state. A clock cycle consists of two successive states.

State 0 (S0) — The read cycle starts. The MCU places an address on ADDR[23:0] and function codes on FC[2:0]. (On CPU16-based MCUs, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one.) The MCU drives R/$\overline{W}$ high for a read cycle. SIZ[1:0] become valid, indicating the number of bytes to be read.

State 1 (S1) — The MCU asserts $\overline{AS}$, indicating that the address on the address bus is valid. The MCU also asserts $\overline{DS}$, signaling the peripheral to place data on the bus.

State 2 (S2) — External logic decodes ADDR[23:0], FC[1:0], R/$\overline{W}$, SIZ[1:0], and $\overline{DS}$. One or both of DATA[15:8] and DATA[7:0] are selected, and the responding device places data on that portion of the bus. Concurrently, the device asserts the appropriate $\overline{DSACK}$ signals. For the MCU to be guaranteed to latch the data in minimum cycle time, at least one $\overline{DSACK}$ signal must be recognized as asserted by the end of S2. (That is, it must meet the input setup time requirement preceding the falling edge of S2.) For wait states to be guaranteed to be inserted, both $\overline{DSACK1}$ and $\overline{DSACK0}$ must remain negated throughout the asynchronous input setup and hold times at the end of S2.

State 3 (S3) — When a change in one or both of the $\overline{\text{DSACK}}$ signals has been recognized, the MCU latches data from the bus on the next falling edge of the clock (S4), and the cycle terminates (S5). If neither $\overline{\text{DSACK}}$ signal is recognized as asserted by the start of S3, the MCU inserts wait states instead of proceeding to S4 and S5. While wait states are added, the MCU continues to sample the $\overline{\text{DSACK}}$ signals on falling edges of the clock until a change in one or more is recognized. In effect, S3 and S4 repeat until a change in the $\overline{\text{DSACK}}$ signals is detected.

State 4 (S4) — If a change in the $\overline{\text{DSACK}}$ signals is detected by the beginning of S3, the MCU latches data on the falling edge at the end of S4. If not, S3 and S4 repeat until a change in the $\overline{\text{DSACK}}$ signals is detected.

State 5 (S5) — The MCU negates $\overline{\text{AS}}$ and $\overline{\text{DS}}$, but holds the address valid to provide address hold time for memory systems. R/$\overline{\text{W}}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must maintain data and assert the $\overline{\text{DSACK}}$ signals until it detects the negation of either $\overline{\text{AS}}$ or $\overline{\text{DS}}$. The external device must remove the data and negate $\overline{\text{DSACK}}$ within approximately one clock period after sensing the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$. Signals that remain asserted beyond this limit can be prematurely detected during the next bus cycle.

Figure 5-3 is a timing diagram of a read cycle.

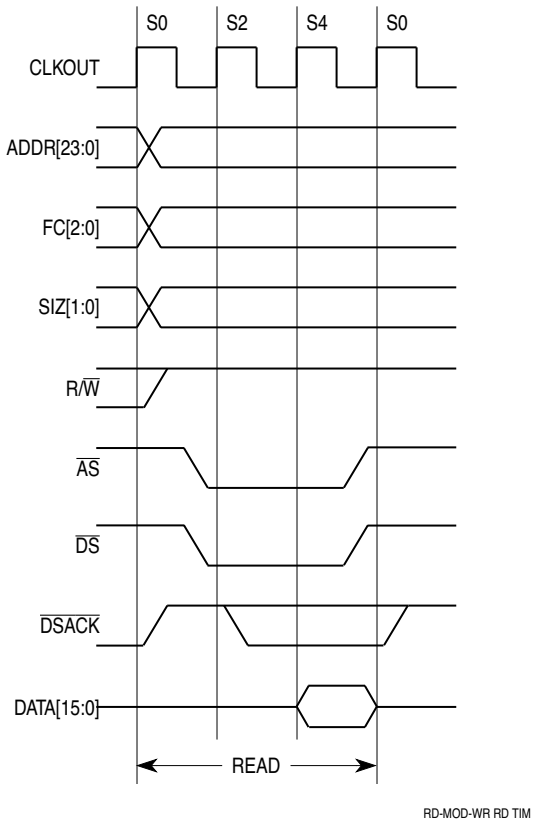


Figure 5-3 Read Cycle Timing Diagram

5.4.2 Write Cycles

During a write operation, the MCU transfers data to an external memory or peripheral device. A write operation consists of one or more write cycles. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes per cycle. For a byte operation, the MCU writes one byte. The portion of the data bus to which each byte is written depends on operand size, peripheral address, and peripheral port size. Refer to **5.3 Dynamic Bus Sizing** for more information. **Figure 5-4** is a flow chart of a write cycle for a word transfer.

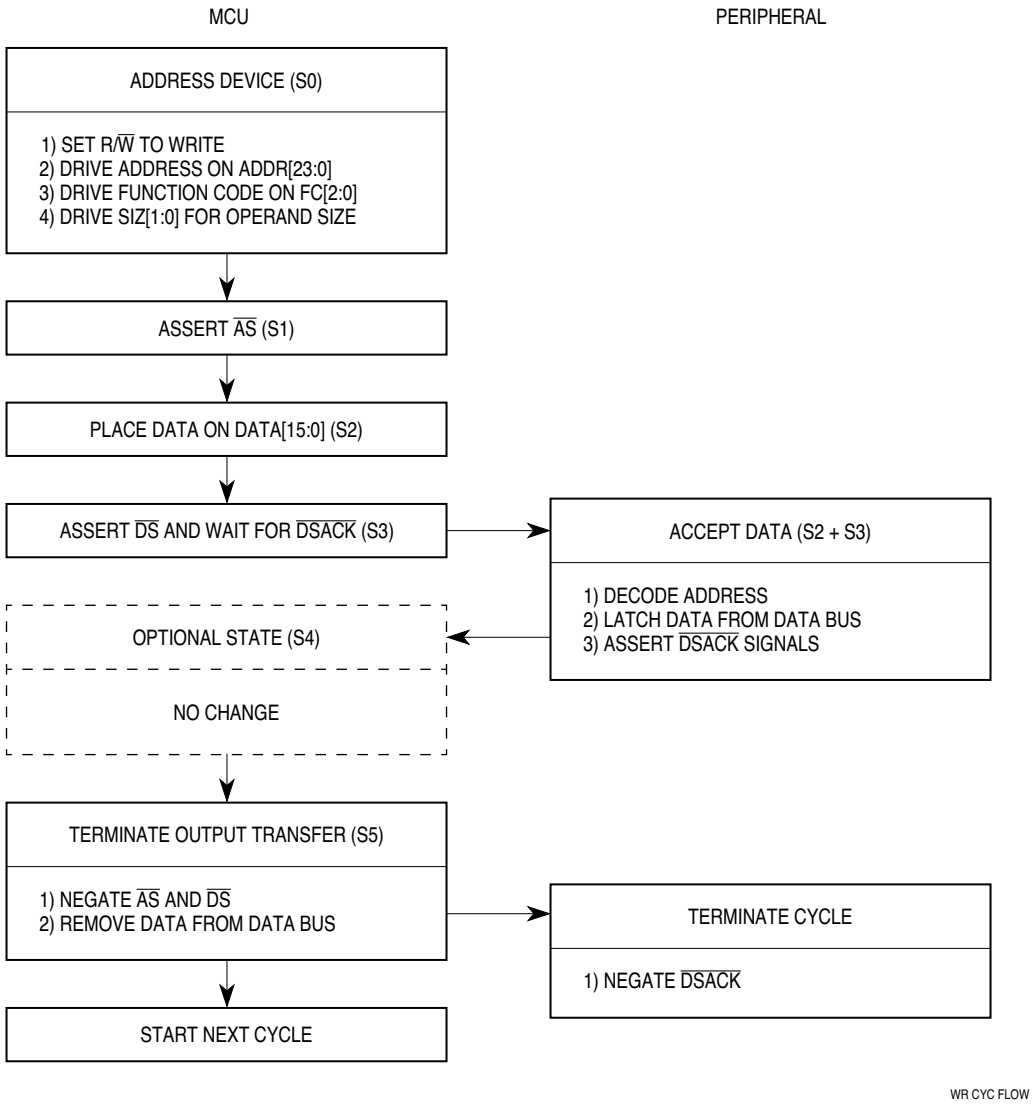


Figure 5-4 Write Cycle Flow Chart

State 0 (S0) — The MCU places an address on ADDR[23:0] and function codes on FC[2:0]. (On CPU16-based MCUs, ADDR[23:20] always follow the state of ADDR19, and FC2 is always equal to one.) The MCU drives R/$\overline{W}$ low for a write cycle. SIZ[1:0] become valid, indicating the number of bytes to be written.

State 1 (S1) — The MCU asserts $\overline{AS}$, indicating that the address on the address bus is valid.

State 2 (S2) — The MCU places the data on DATA[15:0], then begins to sample the $\overline{DSACK}$ signals. External logic decodes the address lines, FC[1:0], R/$\overline{W}$, SIZ[1:0], and $\overline{AS}$. One or both of DATA[15:8] and DATA[7:0] are selected, and appropriate $\overline{DSACK}$ signals are asserted. For the MCU to be guaranteed to latch the data in minimum cycle time, the MCU must recognize a change in at least one $\overline{DSACK}$ signal by the end of S2 (that is, the $\overline{DSACK}$ signal must meet the input setup and hold time requirements). For wait states to be guaranteed to be inserted, both $\overline{DSACK1}$ and $\overline{DSACK0}$ must remain negated throughout the asynchronous input setup and hold times at the end of S2.

State 3 (S3) — The MCU asserts $\overline{DS}$ to indicate that data is stable on the data bus, and the selected peripheral latches the data. When a change in one or both of the $\overline{DSACK}$ signals has already been recognized, S4 elapses, and the cycle terminates during S5. If neither $\overline{DSACK}$ signal changes state by the start of S3, the MCU inserts wait states instead of proceeding to S4 and S5. While wait states are added, the MCU continues to sample the $\overline{DSACK}$ signals on falling edges of the clock until a change in one or more is recognized. In effect, S3 repeats until a change in the $\overline{DSACK}$ signals is detected.

State 4 (S4) — The MCU issues no new control signals during S4.

State 5 (S5) — The MCU negates $\overline{AS}$ and $\overline{DS}$, but holds the address and data valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must assert the $\overline{DSACK}$ signals until it detects the negation of either $\overline{AS}$ or $\overline{DS}$. It must negate $\overline{DSACK}$ within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. Signals that remain asserted beyond this limit can be prematurely detected during the next bus cycle.

Figure 5-5 is a timing diagram of a write cycle.

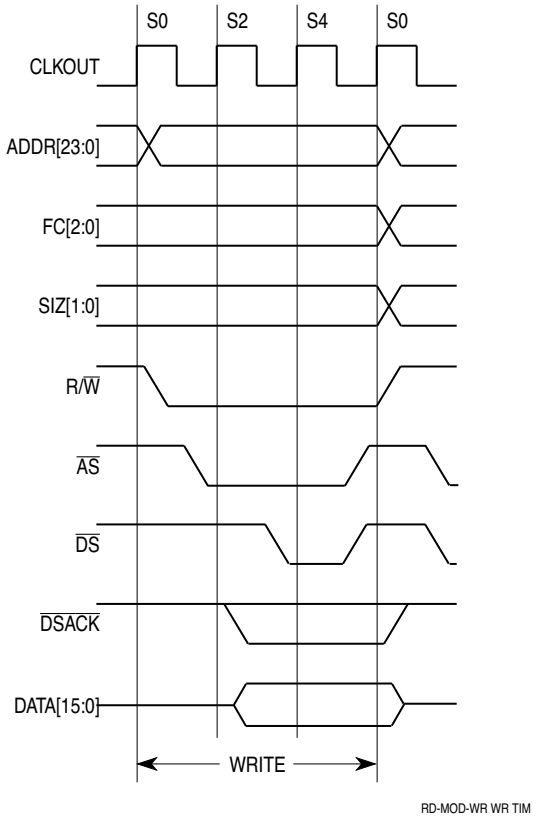


Figure 5-5 Write Cycle Timing Diagram

5.4.3 Indivisible Read-Modify-Write Sequence

The indivisible read-modify-write sequence provides semaphore capabilities for multiprocessor systems. This sequence performs a read, conditionally modifies the data in the arithmetic logic unit, and may write the data out to memory. During the entire read-modify-write sequence, the MCU asserts the $\overline{\text{RMC}}$ signal to indicate that an indivisible operation is occurring. The MCU does not issue a bus grant ($\overline{\text{BG}}$) signal in response to a bus request ($\overline{\text{BR}}$) signal during this operation.

The CPU32 test-and-set (TAS) instruction is the only instruction that generates an indivisible read-modify-write memory cycle. Refer to the *CPU32 Reference Manual* (CPU32RM/AD) for information on this instruction.

NOTE

The CPU16 does not support the $\overline{\text{RMC}}$ signal or the TAS instruction.

Figure 5-6 is an example of a functional timing diagram of a read-modify-write instruction specified in terms of clock periods.

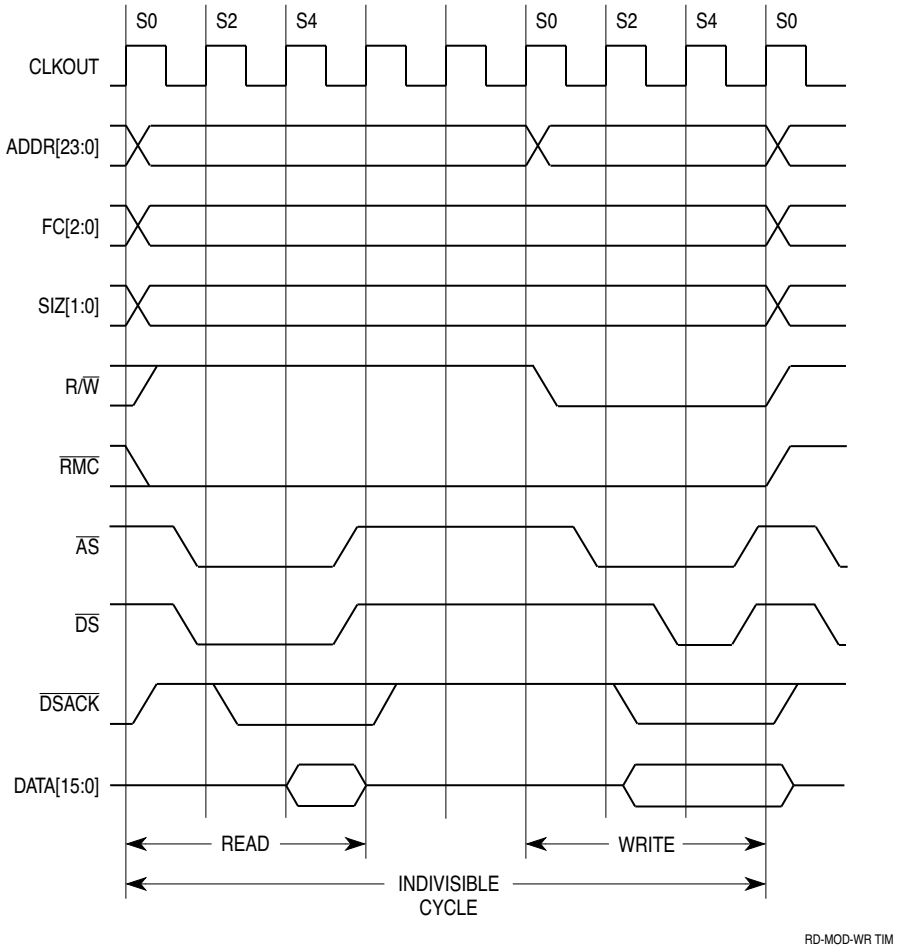


Figure 5-6 Read-Modify-Write Timing

The read-modify-write sequence consists of one or more read cycles, followed by idle states, followed by one or more write cycles. The read and write cycles are similar to those previously described. The differences are summarized as follows:

Read Cycles. If more than one read cycle is required to read the operand, S0–S5 are repeated for each read cycle. In S0, the MCU asserts $\overline{RMC}$ to identify a read-modify-write cycle. When finished reading in S5, the MCU holds the address, R/$\overline{W}$, and FC[2:0] valid in preparation for the write portion of the cycle.

Idle States. The MCU does not assert any new control signals during the idle states between the read and write cycles, but it may internally begin the modify portion of the cycle at this time. If a write cycle is required, the R/$\overline{W}$ signal continues to signal a read operation until state 0 of the write cycle to prevent bus conflicts with the preceding read portion of the cycle.

Write Cycle. The write cycle is omitted if it is not required. If more than one write cycle is required, S0–S5 are repeated for each write cycle. During S0, the MCU drives R/$\overline{W}$ low for a write cycle. Depending on the write operation to be performed, the address

lines may change during S0. Function code and size signals do not change.

5.5 Operand Transfer Cases

Table 5-3 summarizes operand alignment for various types of transfers. Subsequent subsections discuss each allowable transfer case in detail.

In **Table 5-3**, operand bytes are designated as shown in **Figure 5-7**. OP0 – OP3 represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

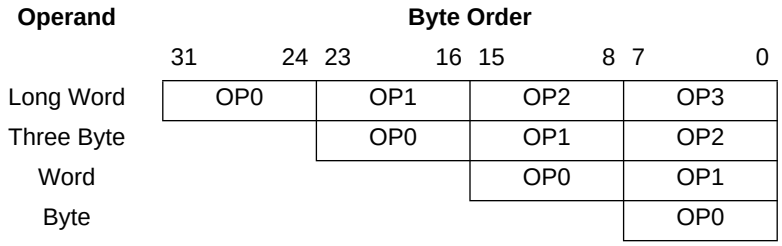


Figure 5-7 Operand Byte Order

In **Table 5-3**, an X in a column means that the state of the signal has no effect. Blank entries in the data bus columns represent bytes of the data bus that the CPU ignores during read cycles. Operands in parentheses are placed on the data bus but ignored by the peripheral during write cycles. The “Next Cycle” column indicates the number of the transfer case for the next bus cycle of the transfer operation. A blank in the “Next Cycle” column indicates that the transfer is complete.

Table 5-3 Operand Transfer Cases

Num	Transfer Case	SIZ[1:0]	ADDR0	$\overline{\text{DSACK}}$[1:0]	Read DATA [15:8]	Cycles DATA [7:0]	Write DATA [15:8]	Cycles DATA [7:0]	Next Cycle
1	Byte to 8-bit Port (Even/Odd)	01	X	10	OP0	—	OP0	OP0	—
2	Byte to 16-bit Port (Even)	01	0	01	OP0	—	OP0	(OP0)	—
3	Byte to 16-bit Port (Odd)	01	1	01	—	OP0	(OP0)	OP0	—
4	Word to 8-bit Port (Aligned)	10	0	10	OP0	—	OP0	(OP1)	1
5	Word to 8-bit Port (Misaligned)[1]	10	1	10	OP0	—	OP0	(OP0)	1
6	Word to 16-bit Port (Aligned)	10	0	01	OP0	OP1	OP0	OP1	—
7	Word to 16-bit Port (Misaligned)[1]	10	1	01	—	OP0	(OP0)	OP0	2
8	Long Word to 8-bit Port (Aligned)	00	0	10	OP0	—	OP0	(OP1)	13
9	Long Word to 8-bit Port (Misaligned)[1,3]	10	1	10	OP0	—	OP0	(OP0)	12
10	Long Word to 16-bit Port (Aligned)	00	0	01	OP0	OP1	OP0	OP1	6
11	Long Word to 16-bit Port (Misaligned)[1,3]	10	1	01	—	OP0	(OP0)	OP0	2
12	3 Byte to 8-bit Port (Aligned)[2]	11	0	10	OP0	—	OP0	(OP1)	5
13	3 Byte to 8-bit Port (Misaligned) [2]	11	1	10	OP0	—	OP0	(OP0)	4

NOTES:
1. The CPU32 does not support misaligned transfers.
2. Three-byte transfer cases occur only as a result of a long word to byte transfer.
3. The CPU16 treats misaligned long-word transfers as two misaligned-word transfers.

5.5.1 Byte Operand to 8-Bit Port

For an eight-bit port, consecutive bytes of data can be read from and written to consecutive byte addresses in the memory system. To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand. Since bytes of data can be written to either odd or even addresses, ADDR0 can be either zero or one. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

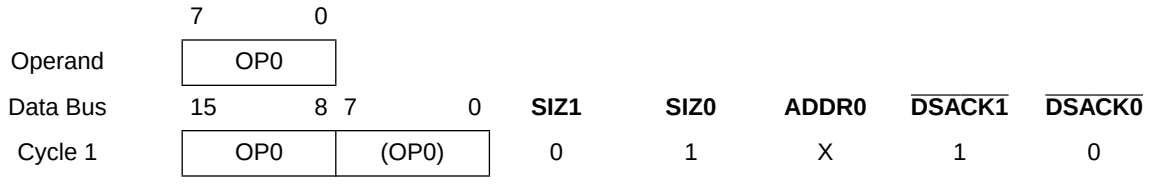


Figure 5-8 Byte Operand to 8-Bit Port

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0].

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral determines the operand size and transfers the data from the upper byte of the data bus to the specified address, then asserts $\overline{\text{DSACK0}}$ to terminate the bus cycle.

5.5.2 Byte Operand to 16-Bit Port, Even (ADDR0 = 0)

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

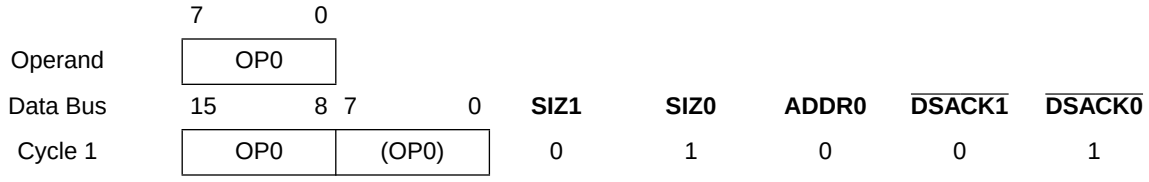


Figure 5-9 Byte Operand to 16-Bit Port, Even (ADDR0 = 0)

For a read operation, the 16-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK1}}$. The MCU reads the data from DATA[15:8] and ignores DATA[7:0].

For a write operation, the MCU drives OP0 onto both bytes of the data bus. The peripheral determines operand size and transfers the data from the upper byte of the data bus to the specified address, then asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

In order to read or write to individual bytes of a 16-bit memory, the memory must consist of 8-bit banks with individual chip selects. Refer to **5.7 System Interfacing Examples** for a description of such a configuration.

5.5.3 Byte Operand to 16-Bit Port, Odd (ADDR0 = 1)

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a single-byte operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

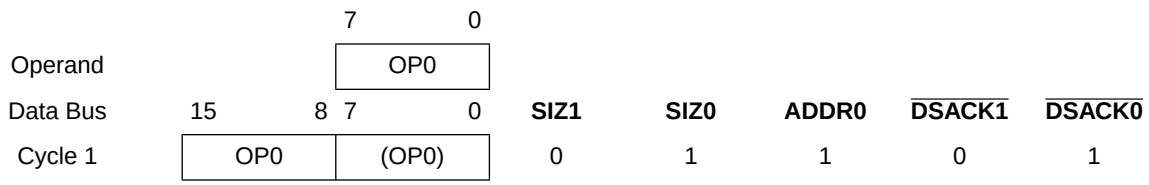


Figure 5-10 Byte Operand to 16-Bit Port, Odd (ADDR0 = 1)

For a read operation, the 16-bit peripheral responds to the address and size signals by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$. The MCU then reads the data from DATA[7:0] and ignores DATA[15:8].

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral determines operand size and transfers OP0 from the lower byte of the data bus to the specified address, then asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

In order to read or write to individual bytes of a 16-bit memory, the memory must be divided into 8-bit banks with individual chip selects. Refer to **5.7 System Interfacing Examples** for a description of such a configuration.

5.5.4 Word Operand to 8-Bit Port, Aligned

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

	15 8	7 0					
Operand	OP0	OP1					
Data Bus	15 8	7 0	**SIZ1**	**SIZ0**	**ADDR0**	**$\overline{\text{DSACK1}}$**	**$\overline{\text{DSACK0}}$**
Cycle 1	OP0	(OP1)	1	0	0	1	0
Cycle 2	OP1	(OP1)	0	1	1	1	0

Figure 5-11 Word Operand to 8-Bit Port, Aligned

For a read operation, the 8-bit peripheral responds to the address and size signals by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0], then decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus for the second cycle of the transfer (a byte read of an 8-bit port).

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA [7:0]. The 8-bit peripheral transfers OP0 from DATA[15:8] to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte has been transferred. The MCU then decrements the transfer size counter, increments the address, and transfers OP1 to bits [15:8] of the data bus for the second cycle of the transfer (a byte write to an 8-bit port).

For both reads and writes, refer to **5.5.1 Byte Operand to 8-Bit Port** for details on the second cycle of the data transfer.

5.5.5 Word Operand to 8-Bit Port, Misaligned

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

NOTE

The CPU32 does not support misaligned operand transfers.

	15 8	7 0	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Operand		OP0					
	OP1						
Data Bus	15 8	7 0					
Cycle 1	OP0	(OP0)	1	0	1	1	0
Cycle 2	OP1	(OP1)	0	1	0	1	0

Figure 5-12 Word Operand to 8-Bit Port, Misaligned

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads the upper operand byte from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus for the second cycle of the transfer (a byte read of an 8-bit port).

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The 8-bit peripheral transfers OP0 to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte of word data has been received. The MCU then decrements the transfer size counter, increments the address, and transfers OP1 to the upper byte of the data bus for the second cycle of the transfer (a byte write to an 8-bit port).

For both reads and writes, refer to **5.5.1 Byte Operand to 8-Bit Port** for details on the second cycle of the data transfer.

5.5.6 Word Operand to 16-Bit Port, Aligned

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

	15 8	7 0	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
Operand	OP0	OP1					
Data Bus	15 8	7 0					
Cycle 1	OP0	OP1	1	0	0	0	1

Figure 5-13 Word Operand to 16-Bit Port, Aligned

For a read operation, the peripheral responds by placing OP0 on DATA[15:8] and OP1 on DATA[7:0], then asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads DATA[15:0] and terminates the cycle.

For a write operation, the MCU drives the word operand on DATA[15:0]. The peripheral device then reads the entire operand from DATA[15:0] and asserts $\overline{\text{DSACK1}}$ to terminate the bus cycle.

5.5.7 Word Operand to 16-Bit Port, Misaligned

To initiate a transfer, the MCU places the desired address on the address bus and drives the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

NOTE

The CPU32 does not support transfers of misaligned operands.

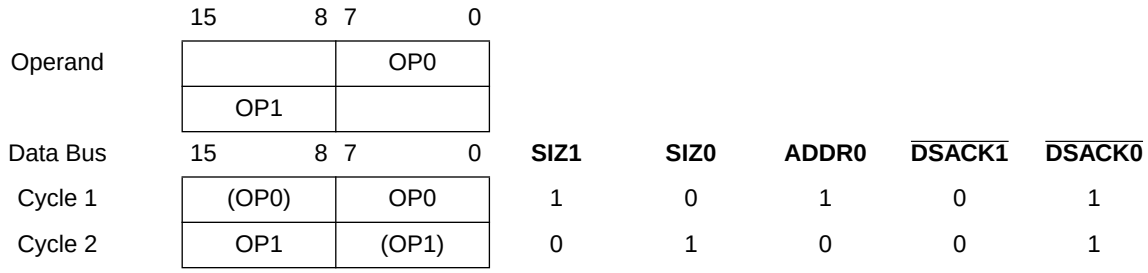


Figure 5-14 Word Operand to 16-Bit Port, Misaligned

For a read operation, the peripheral responds by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads OP0 from DATA[7:0], decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus for the second cycle of the transfer (a byte read of a 16-bit port).

For a write operation, the MCU drives OP0 on both bytes of the data bus. The peripheral device reads OP0 from DATA[7:0] and asserts $\overline{\text{DSACK1}}$. The MCU decrements the transfer size counter, increments the address, and places OP1 on both bytes of the data bus for the second cycle of the transfer (a byte write to a 16-bit port).

For both reads and writes, refer to **5.5.2 Byte Operand to 16-Bit Port, Even (ADDR0 = 0)** for details on the second cycle of the data transfer.

5.5.8 Long-Word Operand to 8-Bit Port, Aligned

The MCU drives the address bus with the desired address and the size pins to indicate a long word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

	15 8	7 0					
Operand	OP0	OP1					
	OP1	OP3					
Data Bus	15 8	7 0	**SIZ1**	**SIZ0**	**ADDR0**	**$\overline{\text{DSACK1}}$**	**$\overline{\text{DSACK0}}$**
Cycle 1	OP0	(OP1)	0	0	0	1	0
Cycle 2	OP1	(OP1)	1	1	1	1	0
Cycle 3	OP2	(OP2)	1	0	0	1	0
Cycle 4	OP3	(OP3)	0	1	1	1	0

Figure 5-15 Long-Word Operand to 8-Bit Port, Aligned

For a read operation, the peripheral places OP0 on DATA[15:8] and asserts $\overline{\text{DSACK0}}$ to indicate an 8-bit port. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus for the second cycle of the transfer (a three-byte read of an 8-bit port). The process is repeated for OP2 in the third cycle (aligned word to 8-bit port transfer) and OP3 in the fourth cycle (byte to 8-bit port transfer).

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The peripheral device then reads only OP0 from DATA[15:8] and asserts $\overline{\text{DSACK0}}$ to indicate an 8-bit port. The MCU then decrements the transfer size counter, increments the address, and writes OP1 to DATA[15:8] during the second cycle (a three-byte to 8-bit port transfer). The process is repeated for OP2 in the third cycle (aligned word to 8-bit port transfer) and OP3 in the fourth cycle (byte to 8-bit port transfer).

Figure 5-16 and **Figure 5-17** are timing diagrams for long-word read and write operations, respectively, to an eight-bit port.

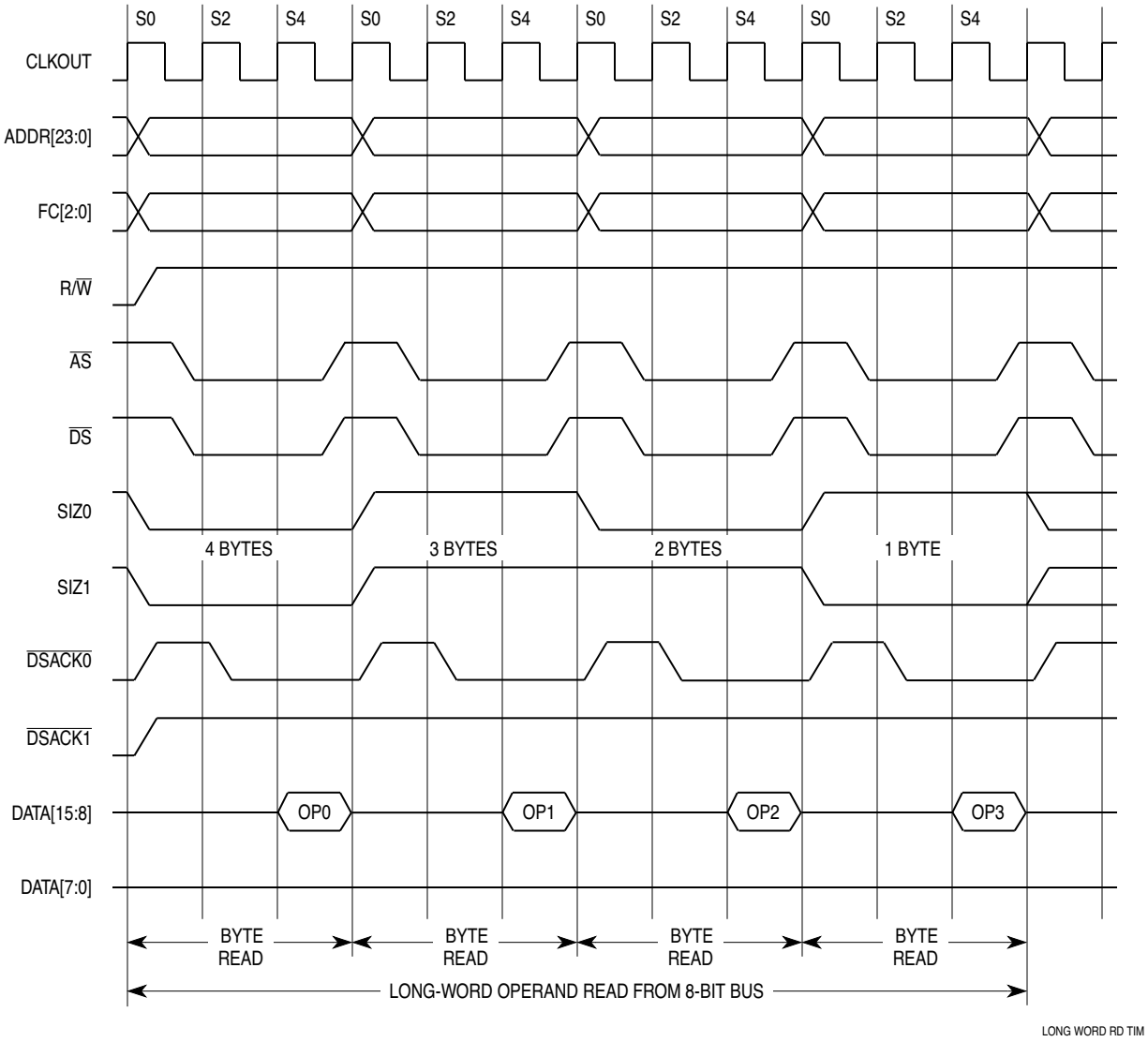


Figure 5-16 Timing of a Long-Word Read of an 8-Bit Port

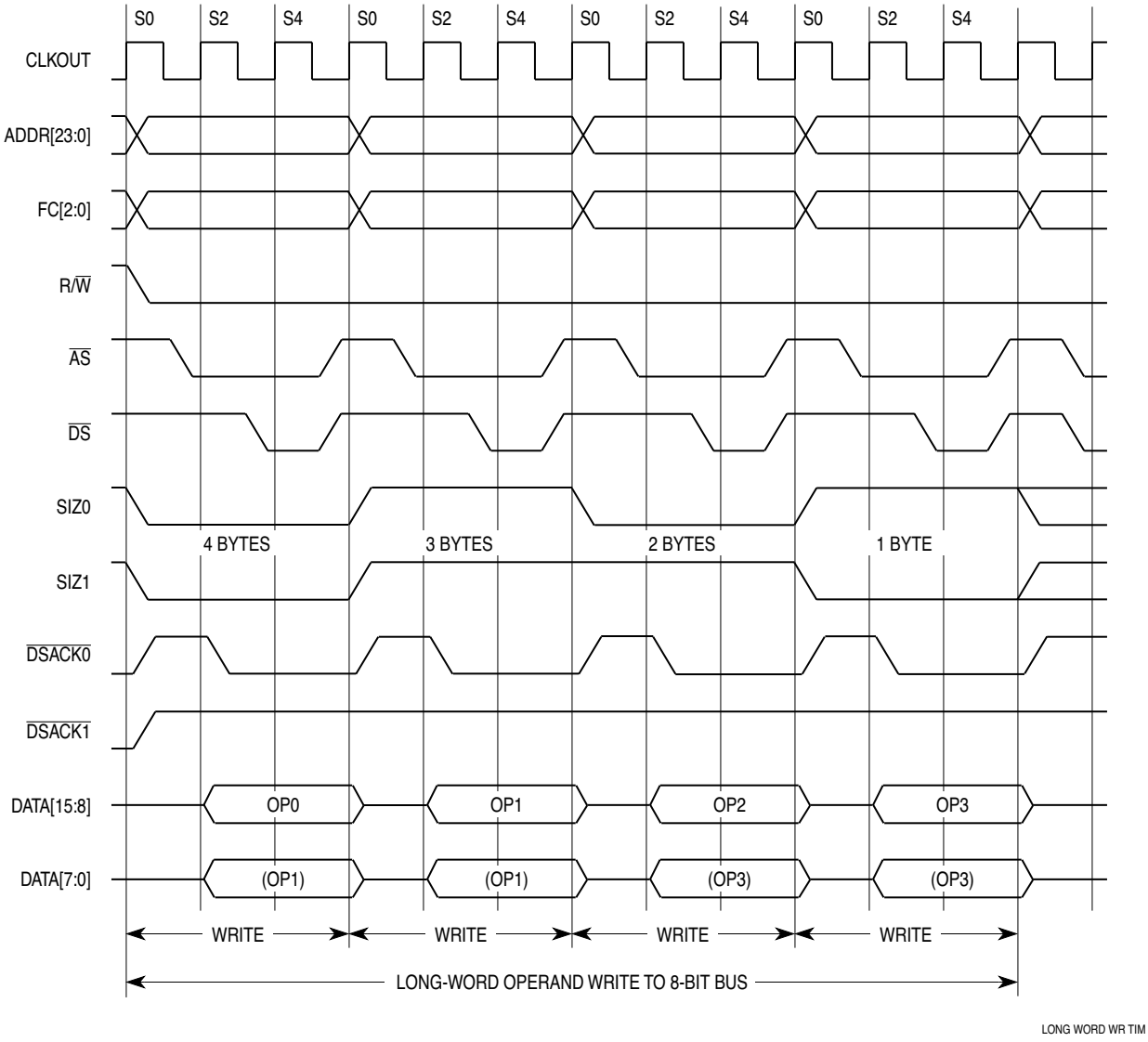


Figure 5-17 Timing of a Long-Word Write to an 8-Bit Port

5.5.9 Long-Word Operand to 8-Bit Port, Misaligned

The CPU16 treats misaligned long words as two misaligned words. The MCU drives the address bus with the desired address and the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{W}$ pins to appropriate values.

NOTE

The CPU32 does not support transfers of misaligned operands.

	15 8	7 0					
Operand		OP0					
	OP1	OP2					
	OP3						
Data Bus	15 8	7 0	**SIZ1**	**SIZ0**	**ADDR0**	**$\overline{\text{DSACK1}}$**	**$\overline{\text{DSACK0}}$**
Cycle 1	OP0	(OP0)	1	0	1	1	0
Cycle 2	OP1	(OP1)	0	1	0	1	0
Cycle 3	OP2	(OP2)	1	0	1	1	0
Cycle 4	OP3	(OP3)	0	1	0	1	0

Figure 5-18 Long-Word Operand to 8-Bit Port, Misaligned

For a read operation, the 8-bit peripheral responds by placing OP0 on DATA[15:8] and asserting $\overline{\text{DSACK0}}$. The MCU reads OP0 from DATA[15:8] and ignores DATA[7:0]. The MCU then decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus during the second cycle (a byte read of an 8-bit port). When the second cycle is finished, one misaligned word has been read. The MCU then reads a second misaligned word (OP2 and OP3) from the 8-bit port during the third and fourth cycles.

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA [7:0]. The 8-bit peripheral transfers OP0 to the specified address, then asserts $\overline{\text{DSACK0}}$ to indicate that the first byte of word data has been received. The MCU then decrements the transfer size counter, increments the address, and places OP1 on the upper half of the data bus for the second cycle of the transfer (a byte write to an 8-bit port). After this second cycle, one misaligned word has been written. The MCU then writes a second misaligned word (OP2 and OP3) to the 8-bit port during the third and fourth cycles of the transfer.

5.5.10 Long-Word Operand to 16-Bit Port, Aligned

The MCU drives the address bus with the desired address and drives the size pins to indicate a long-word operand. The MCU also drives the function code and R/$\overline{\text{W}}$ pins to appropriate values.

	15 8	7 0					
Operand	OP0	OP1					
	OP2	OP3					
Data Bus	15 8	7 0	**SIZ1**	**SIZ0**	**ADDR0**	**$\overline{\text{DSACK1}}$**	**$\overline{\text{DSACK0}}$**
Cycle 1	OP0	(OP0)	0	0	0	0	1
Cycle 2	OP1	(OP1)	1	0	0	0	1

Figure 5-19 Long-Word Operand to 16-Bit Port, Aligned

For a read operation, the 16-bit peripheral responds by placing OP0 on DATA[15:8] and OP1 on DATA[7:0], then asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. The MCU reads OP0 and OP1 from DATA[15:0]. The MCU increments the address bus by two, drives SIZ1 to 1 and SIZ0 to 0, and waits for the peripheral to place OP2 and OP3 on the data bus during the second cycle of the transfer (an aligned word read of a 16-bit port).

For a write operation, the MCU drives OP0 on DATA[15:8] and OP1 on DATA[7:0]. The peripheral device reads OP0 and OP1 from DATA[15:0] and asserts $\overline{\text{DSACK1}}$ to indicate a 16-bit port. The MCU increments the address bus by two, drives SIZ1 to 1 and SIZ0 to 0, and places OP2 and OP3 on the data bus during the second cycle of the transfer (an aligned word write to a 16-bit port).

Figure 5-20 is a timing diagram for a long-word read or write operation involving a 16-bit port.

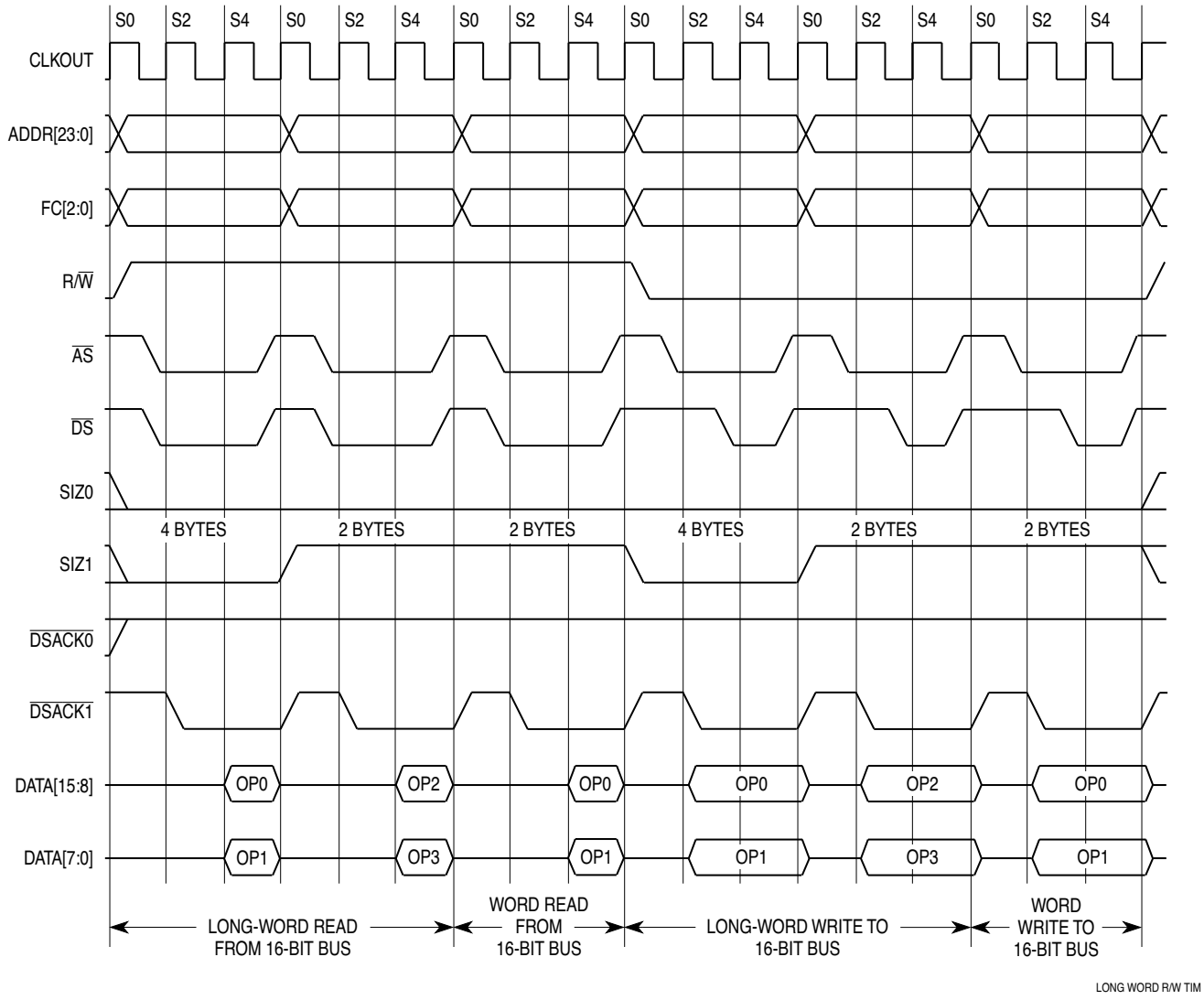


Figure 5-20 Timing of Long-Word Read or Write, 16-Bit Port

5.5.11 Long-Word Operand to 16-Bit Port, Misaligned

The CPU16 treats misaligned long-word transfers as two misaligned word transfers. The MCU drives the address bus with the desired address and drives the size pins to indicate a word operand. The MCU also drives the function code and R/$\overline{W}$ pins to appropriate values.

NOTE

The CPU32 does not support transfers of misaligned operands.

	15 8	7 0					
Operand		OP0					
	OP1	OP2					
	OP3						
Data Bus	15 8	7 0	**SIZ1**	**SIZ0**	**ADDR0**	**$\overline{\text{DSACK1}}$**	**$\overline{\text{DSACK0}}$**
Cycle 1	(OP0)	OP0	1	0	1	0	1
Cycle 2	OP1	(OP1)	0	1	0	0	1
Cycle 3	(OP2)	OP2	1	0	1	0	1
Cycle 4	OP3	(OP3)	0	1	0	0	1

Figure 5-21 Long-Word Operand to 16-Bit Port, Misaligned

For a read operation, the peripheral responds by placing OP0 on DATA[7:0] and asserting $\overline{\text{DSACK1}}$ to indicate a 16-bit port. When $\overline{\text{DSACK1}}$ is asserted, the MCU reads OP0 from DATA[7:0], decrements the transfer size counter, increments the address, and waits for the peripheral to place OP1 on the upper byte of the data bus during the second cycle of the transfer (a byte read of an 8-bit port). When the second cycle is finished, one misaligned word has been read. The MCU then reads a second misaligned word (OP2 and OP3) from the 16-bit port during the third and fourth cycles of the transfer.

For a write operation, the MCU first drives OP0 on DATA[7:0] and duplicates it on DATA[15:8]. The peripheral device reads OP0 from DATA[7:0] and asserts $\overline{\text{DSACK1}}$. The MCU decrements the transfer size counter, increments the address, and places OP1 on both bytes of the data bus during the second cycle of the transfer. When the second cycle is finished, one misaligned word has been written. The MCU then writes a second misaligned word (OP2 and OP3) to the 16-bit port during the third and fourth cycles of the transfer.

5.6 Function Codes and Memory Usage

The CPU generates function code output signals FC[2:0] to indicate the type of activity occurring on the data or address bus. These signals can be considered address extensions that can be externally decoded to determine which of eight external address spaces is accessed during a bus cycle.

Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{\text{AS}}$ is asserted.

Table 5-4 shows address space encoding.

Table 5-4 Address Space Encoding

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User Data Space
0	1	0	User Program Space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor Data Space
1	1	0	Supervisor Program Space
1	1	1	CPU Space

The supervisor bit in the status register determines whether the CPU is operating in supervisor or user mode. Addressing mode and the instruction being executed determine whether a memory access is to program or data space.

NOTE

Because the CPU16 always operates in supervisor mode (FC2 = 1), it does not use address spaces 0 to 3.

The SIM chip select circuits can be programmed to respond to the type of memory access: CPU space, user space, supervisor space, or user/supervisor space. Refer to **SECTION 7 CHIP SELECTS** for more information.

5.7 System Interfacing Examples

This section provides examples of interfacing the MCU to 8- and 16-bit memory devices. For purposes of illustration, the examples that follow decode EBI signals externally. In practice, it is more efficient to use SIM chip selects for most applications. For configurations using SIM chip selects, refer to **7.11 Interfacing Example with Chip Selects**.

5.7.1 Connecting an 8-Bit Device to the MCU

When connecting an 8-bit memory device or peripheral to the MCU, connect the upper eight bits of the data bus (DATA[15:8]) to the eight data lines of the external device. Connect high-order address lines to the external chip select and lower-order address lines to the corresponding address lines of the memory or peripheral. **Figure 5-22** shows the basic configuration.

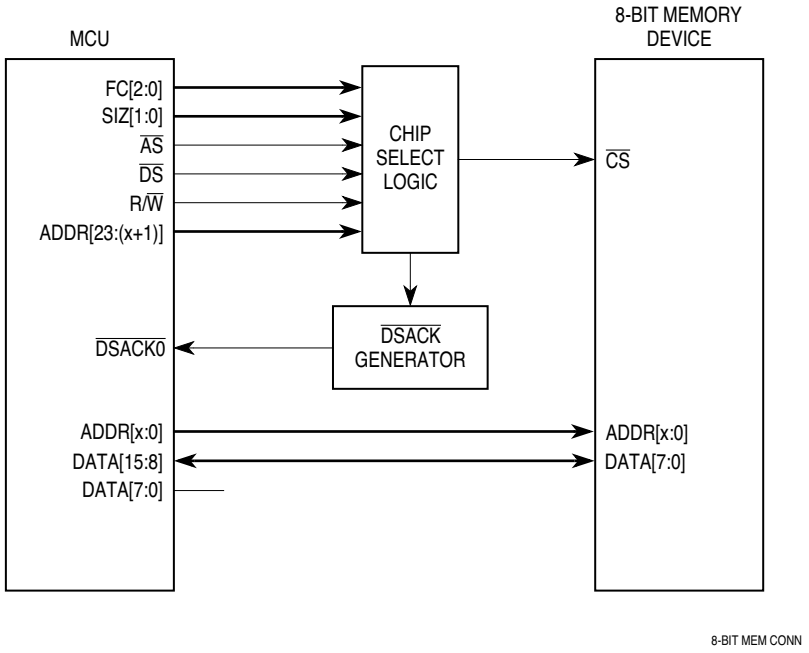


Figure 5-22 Connecting an 8-Bit Memory Device

5.7.2 Connecting a 16-Bit Memory Device to the MCU

When connecting a 16-bit memory system to the MCU, connect DATA[15:0] of the MCU to the sixteen data lines of the memory. Since the memory is arranged in words (16 bits) rather than bytes, the address bus is incremented by two bytes with each successive access. To accommodate this, connect ADDR1 of the MCU to ADDR0 of the memory, ADDR2 of the MCU to ADDR1 of the memory, and so on. Do not connect ADDR0 of the MCU to the address bus of the memory system.

This method precludes writes to individual bytes: all memory accesses are 16 bits wide. To be able to write to individual bytes of the memory, construct the memory system as outlined in **5.7.3 Connecting Two 8-bit Memory Devices to the MCU**, or use chip selects as explained in **7.11 Interfacing Example with Chip Selects**.

Figure 5-23 illustrates connecting a 16-bit memory device to the MCU.

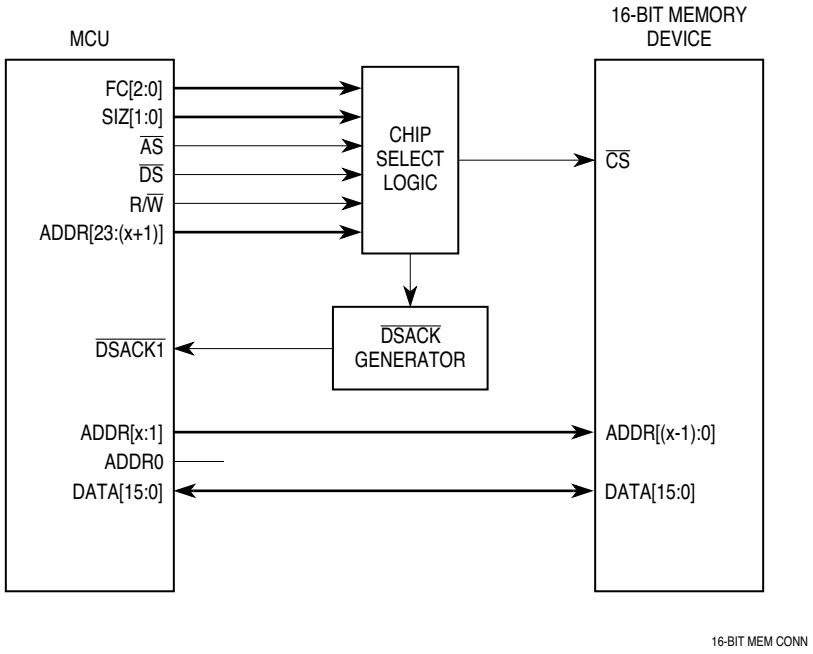


Figure 5-23 Connecting a 16-Bit Memory Device

5.7.3 Connecting Two 8-bit Memory Devices to the MCU

A 16-bit memory can be implemented using two 8-bit banks. This configuration allows both byte and word memory accesses. To implement this method, individual chip selects must be used for each bank of memory. The upper bank of memory is selected when ADDR0 = 1 and the lower bank is selected when ADDR0 = 0. ADDR0 is thus connected to the chip selects, rather than to the address lines of the memory. **Figure 5-24** illustrates this configuration.

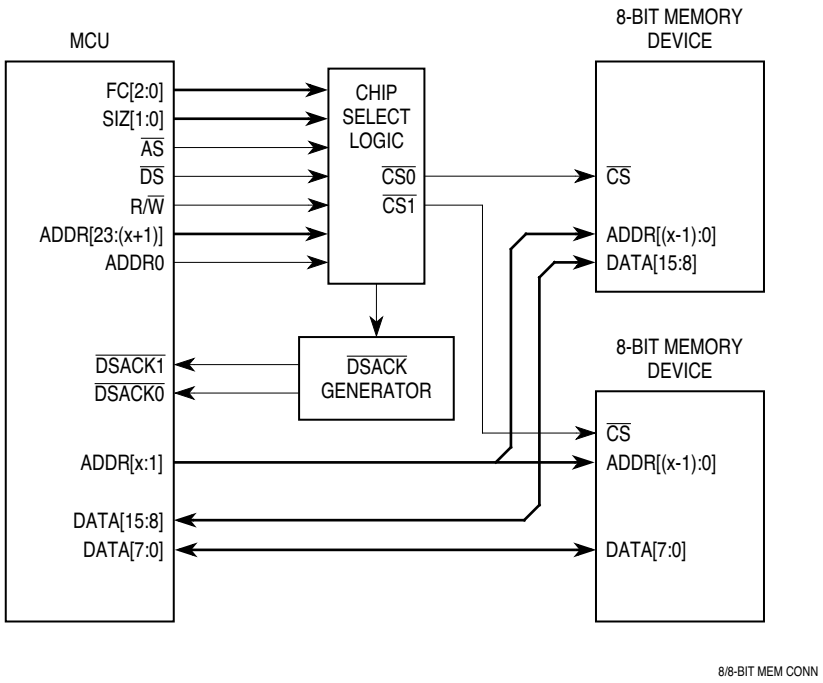


Figure 5-24 Connecting Two 8-bit Memory Devices

The SIM chip-select circuitry can be programmed to individually select upper and lower bytes of a 16-bit memory system using two 8-bit banks. Refer to **7.11 Interfacing Example with Chip Selects** for an example of such a configuration.

5.8 CPU Space Cycles

Function code signals FC[2:0] designate which of eight external address spaces is accessed during a bus cycle. Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Refer to **5.6 Function Codes and Memory Usage** for more information on function codes.

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. The three encodings are shown in **Figure 5-25**. These encodings represent breakpoint acknowledge (type $0) cycles, low power stop broadcast (Type $3) cycles, and interrupt acknowledge (type $F) cycles. Type 0 and type 3 cycles are discussed in the following paragraphs. Refer to **SECTION 6 INTERRUPTS** for a comprehensive discussion of interrupt acknowledge bus cycles.

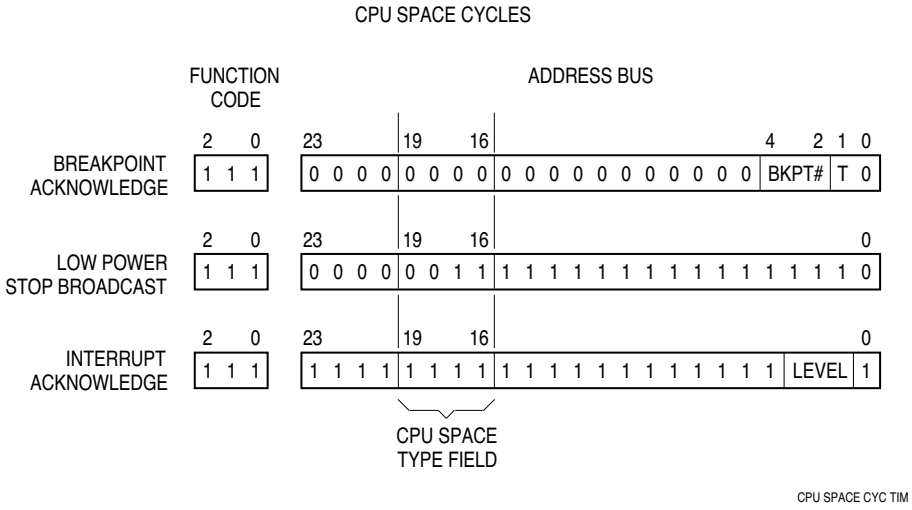


Figure 5-25 CPU Space Address Encoding

5.8.1 Breakpoint Acknowledge Cycle

Breakpoints are used to stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with the background debugging mode. The following paragraphs discuss breakpoint processing when background debugging mode is not enabled. Refer to the appropriate CPU reference manual for information on exception processing and background debugging mode.

On CPU32-based MCUs, both hardware and software can initiate breakpoints; CPU16-based MCUs support hardware-initiated breakpoints only. Refer to the appropriate CPU reference manual for details. The following paragraphs discuss both types of breakpoints.

5.8.1.1 Software Breakpoints

The CPU32 BKPT instruction allows the user to insert breakpoints through software. The CPU responds to this instruction by initiating a breakpoint-acknowledge read cycle in CPU space. It places the breakpoint acknowledge (%0000) code in ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) in ADDR[4:2], and %0 (indicating a software breakpoint) in ADDR1.

NOTE

The CPU16 does not support the BKPT instruction.

The external breakpoint circuitry decodes the function code and address lines and responds by either asserting $\overline{\text{BERR}}$ or placing an instruction word on the data bus and asserting $\overline{\text{DSACK}}$.

If the bus cycle is terminated by $\overline{\text{DSACK}}$, the CPU32 reads the instruction on the data bus and inserts the instruction into the pipeline. (For eight-bit ports, this instruction fetch may require two read cycles.)

If the bus cycle is terminated by $\overline{\text{BERR}}$, the CPU32 then performs illegal-instruction exception processing: it acquires the number of the illegal-instruction exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address.

5.8.1.2 Hardware Breakpoints

A hardware breakpoint is initiated by assertion of the $\overline{\text{BKPT}}$ input. The CPU responds by initiating a breakpoint-acknowledge read cycle in CPU space. It places $00001E on the address bus. (The breakpoint acknowledge code of %0000 is placed in ADDR[19:16], the breakpoint number value of %111 is placed in ADDR[4:2], and ADDR1 is set to one, indicating a hardware breakpoint.)

With CPU16-based MCUs, the external breakpoint circuitry decodes the function code and address lines, places an instruction word on the data bus, and asserts either $\overline{\text{BERR}}$ or $\overline{\text{DSACK}}$. The CPU16 then performs hardware breakpoint exception processing: it acquires the number of the hardware breakpoint exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address.

With CPU32-based MCUs, the external breakpoint circuitry decodes the function code and address lines, places an instruction word on the data bus, and asserts $\overline{\text{BERR}}$. The CPU32 then performs hardware breakpoint exception processing: it acquires the number of the hardware breakpoint exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address. If the external device asserts $\overline{\text{DSACK}}$ rather than $\overline{\text{BERR}}$, the CPU32 ignores the breakpoint and continues processing.

Refer to the appropriate CPU reference manual for additional details.

When $\overline{\text{BKPT}}$ assertion is synchronized with an instruction prefetch, processing of the breakpoint exception occurs at the end of that instruction. The prefetched instruction is “tagged” with the breakpoint when it enters the instruction pipeline, and the breakpoint exception occurs after the instruction executes. If the pipeline is flushed before the tagged instruction is executed, no breakpoint occurs. When $\overline{\text{BKPT}}$ assertion is synchronized with an operand fetch, exception processing occurs at the end of the instruction during which $\overline{\text{BKPT}}$ is latched.

Breakpoint operation flow for CPU32-based MCUs is shown in **Figure 5-26**; **Figure 5-27** shows breakpoint flow for CPU16-based MCUs. **Figure 5-28** is the timing diagram for the breakpoint acknowledge cycle for the CPU32 BKPT instruction. **Figure 5-29** shows the timing (for both CPU16- and CPU32-based MCUs) for breakpoint acknowledge cycles initiated by a low signal on the $\overline{\text{BKPT}}$ pin.

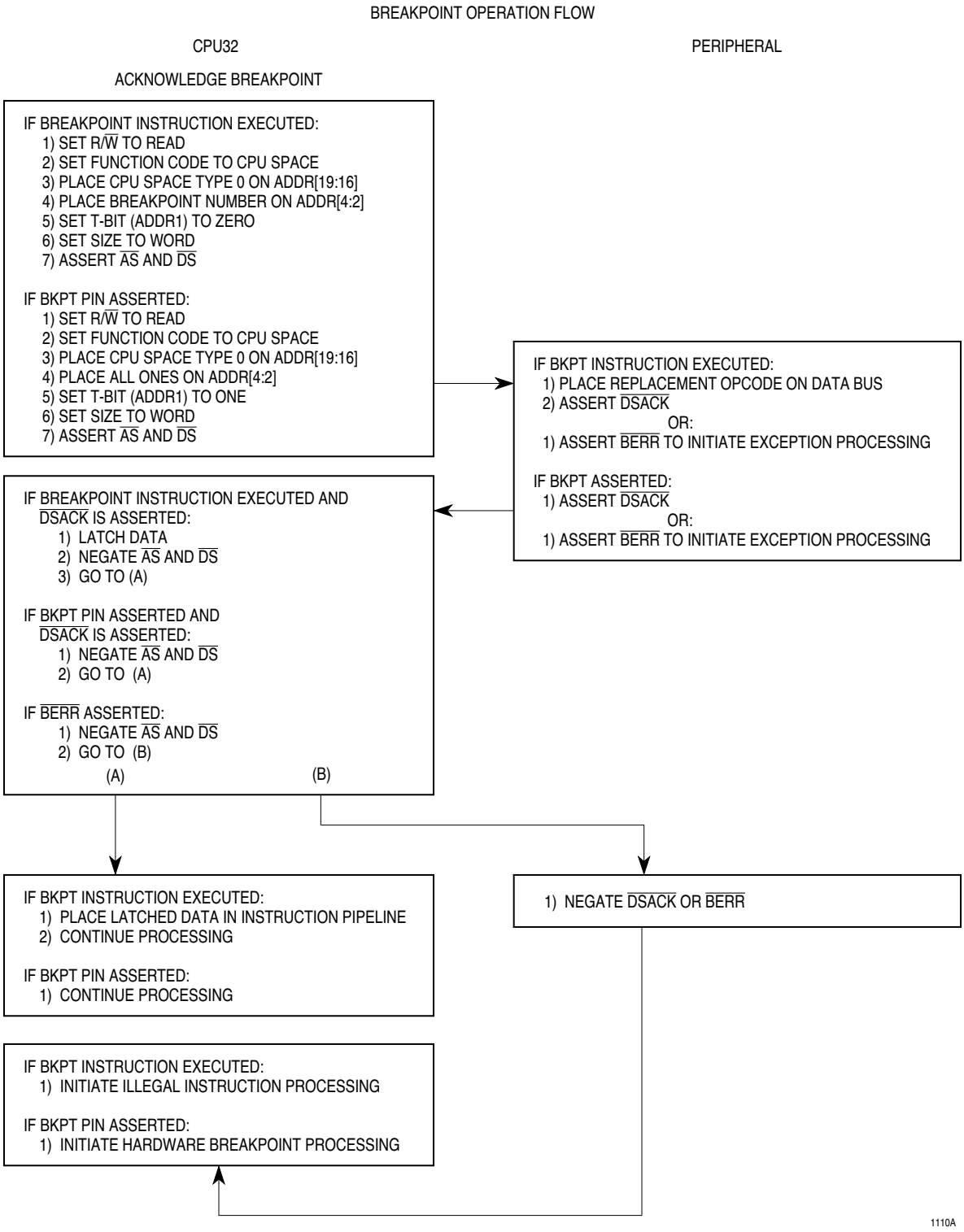


Figure 5-26 CPU32 Breakpoint Operation Flow

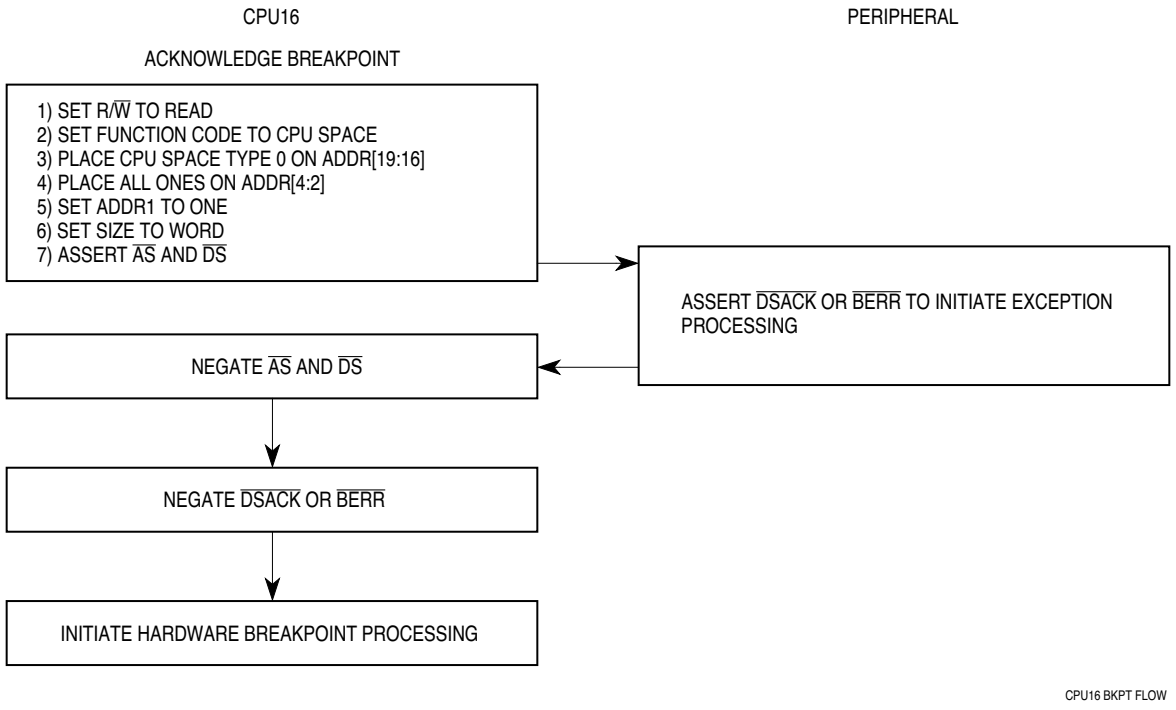


Figure 5-27 CPU16 Breakpoint Operation Flow

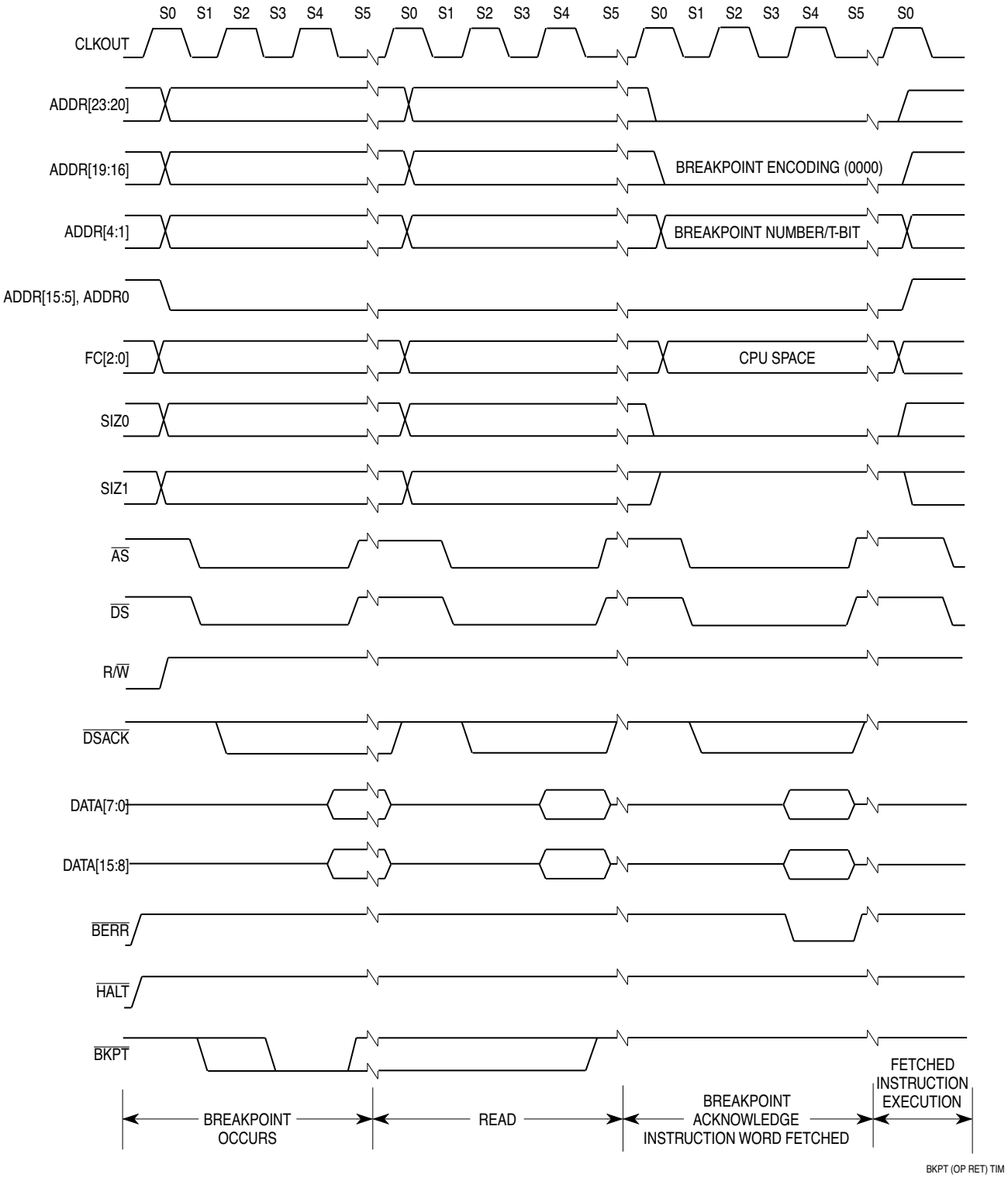


Figure 5-28 Breakpoint Acknowledge Cycle Timing — Opcode Returned (CPU32 Only)

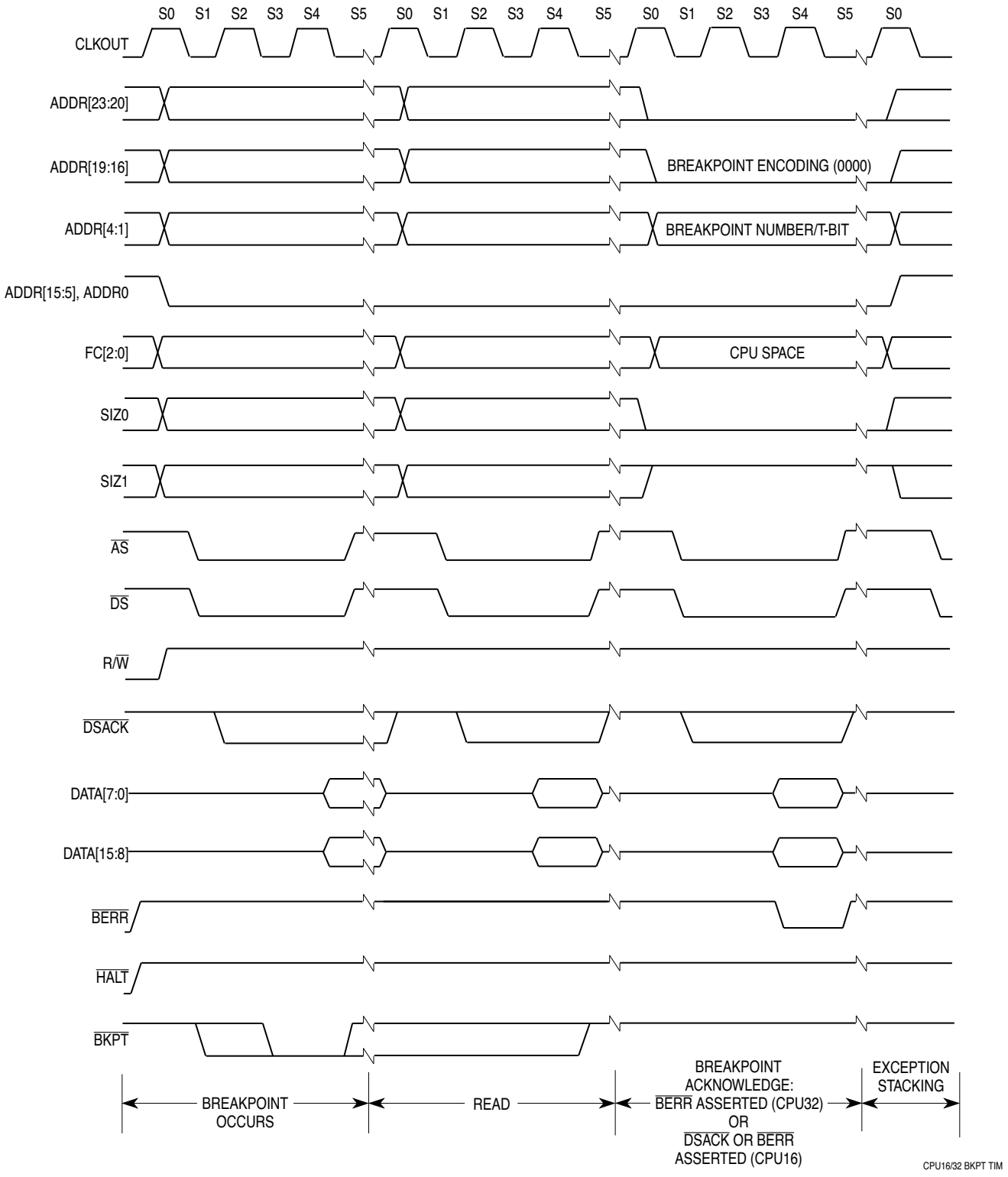


Figure 5-29 Breakpoint Acknowledge Cycle Timing — Exception Signaled

5.8.2 LPSTOP Broadcast Cycle

When the CPU executes the LPSTOP instruction, an LPSTOP broadcast cycle is generated. During an LPSTOP broadcast cycle, the CPU performs a CPU space write to address $3FFFE. This write puts a copy of the interrupt mask value in the clock control logic. The mask is encoded on the data bus as shown in **Figure 5-30**. The CPU space cycle is shown externally (if the bus is available) as an indication to external devices that the MCU is going into low power stop mode. The SIM provides an internally generated $\overline{\text{DSACK}}$ response to this cycle. The timing of this bus cycle is the same as for a fast write cycle.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	IP MASK		

Figure 5-30 LPSTOP Interrupt Mask Level

The SIM brings the MCU out of low-power mode when either an interrupt of higher priority than the stored mask or a reset occurs. Refer to the appropriate CPU reference manual for more information.

5.9 Bus Error Processing

An external device or a chip-select circuit must assert at least one of the $\overline{\text{DSACK}}$[1:0] signals or the $\overline{\text{AVEC}}$ signal to terminate a bus cycle normally. Bus error processing occurs when bus cycles are not terminated in the expected manner. The internal bus monitor can be used to generate $\overline{\text{BERR}}$ internally, causing a bus error exception to be taken. Bus cycles can also be terminated by assertion of the external $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ signal, or by assertion of the two signals simultaneously.

Acceptable bus cycle termination sequences are summarized as follows. The case numbers refer to **Table 3-5**, which indicates the results of each type of bus cycle termination.

Normal Termination

$\overline{\text{DSACK}}$ is asserted; $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ remain negated (case 1).

Halt Termination

$\overline{\text{HALT}}$ is asserted at the same time, or before $\overline{\text{DSACK}}$, and $\overline{\text{BERR}}$ remains negated (case 2).

Bus Error Termination

$\overline{\text{BERR}}$ is asserted in lieu of, at the same time as, or before $\overline{\text{DSACK}}$ (case 3), or after $\overline{\text{DSACK}}$ (case 4), and $\overline{\text{HALT}}$ remains negated; $\overline{\text{BERR}}$ is negated at the same time or after $\overline{\text{DSACK}}$.

Retry Termination

$\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are asserted in lieu of, at the same time as, or before $\overline{\text{DSACK}}$ (case 5) or after $\overline{\text{DSACK}}$ (case 6); $\overline{\text{BERR}}$ is negated at the same time or after $\overline{\text{DSACK}}$; $\overline{\text{HALT}}$ may be negated at the same time or after $\overline{\text{BERR}}$.

NOTE

On CPU16-based MCUs, assertion of $\overline{\text{BERR}}$ results in a bus error exception regardless of the state of the $\overline{\text{HALT}}$ signal. These CPUs do not support the retry termination sequence. Refer to the appropriate CPU manual or to the user manual for the specific device for details.

Table 5-5 shows various combinations of control signal sequences and the resulting bus cycle terminations.

Table 5-5 $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ Assertion Results

Case Number	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S NA X	Normal termination.
2	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA A/S	S NA S	Halt termination: normal cycle terminate and halt. Continue when $\overline{\text{HALT}}$ is negated.
3	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA/A A NA	X S X	Bus error termination: terminate and take bus error exception, possibly deferred.
4	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A A NA	X S NA	Bus error termination: terminate and take bus error exception, possibly deferred.
5	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA/A A A/S	X S S	Retry termination: terminate and retry when $\overline{\text{HALT}}$ is negated. (The result is a bus error termination, rather than retry termination, on CPU16-based MCUs.)
6	$\overline{\text{DSACK}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	X A A	Retry termination: terminate and retry when $\overline{\text{HALT}}$ is negated. (The result is a bus error termination, rather than retry termination, on CPU16-based MCUs.)

NOTES:
N= The number of current even bus state (S2, S4, etc.).
A= Signal is asserted in this bus state.
NA= Signal is not asserted in this state.
X= Don't care.
S= Signal was asserted in previous state and remains asserted in this state.

To properly control termination of a bus cycle for a retry or a bus error condition, $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ must be asserted and negated with the rising edge of the MCU clock. This ensures that when two signals are asserted simultaneously, the required setup time and hold time for both of them are met for the same falling edge of the MCU clock. (Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for timing requirements.) External circuitry that provides these signals must be designed with these constraints in mind, or else the internal bus monitor must be used.

$\overline{\text{DSACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ may be negated after $\overline{\text{AS}}$ is negated.

WARNING

If $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ remain asserted into S2 of the next bus cycle, that cycle may be terminated prematurely.

5.9.1 Bus Error Exceptions

The CPU treats bus errors as a type of exception. Bus error exception processing begins when the CPU detects assertion of the IMB $\overline{\text{BERR}}$ signal (by the internal bus monitor or an external source) while the $\overline{\text{HALT}}$ signal remains negated.

NOTE

On CPU16-based MCUs, assertion of $\overline{\text{BERR}}$ results in a bus error exception regardless of the state of the $\overline{\text{HALT}}$ signal.

$\overline{\text{BERR}}$ takes precedence over $\overline{\text{DSACK}}$, provided it meets the timing constraints described in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

WARNING

If $\overline{\text{BERR}}$ does not meet these constraints, it may cause unpredictable operation of the MCU. If $\overline{\text{BERR}}$ remains asserted into the next bus cycle, it may cause incorrect operation of that cycle.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU at the end of the bus cycle in which it was asserted. Since bus cycles can overlap instruction boundaries, bus error exception processing may not occur at the end of the instruction in which the bus cycle begins. Timing of $\overline{\text{BERR}}$ detection and acknowledgment depends on several factors:

- Which bus cycle of an instruction is terminated by assertion of $\overline{\text{BERR}}$.
- The number of bus cycles in the instruction during which $\overline{\text{BERR}}$ is asserted.
- The number of bus cycles in the instruction following the instruction in which $\overline{\text{BERR}}$ is asserted.
- Whether $\overline{\text{BERR}}$ is asserted during a program space access or a data space access.

Because of these factors, it is impossible to predict precisely how long after occurrence of a bus error the bus error exception will be processed.

CAUTION

The external bus interface does not latch data when an external bus cycle is terminated by a bus error. When this occurs during an instruction prefetch, the IMB precharge state (bus pulled high, or $FF) is latched into the CPU instruction register, with indeterminate results.

The instruction prefetch mechanism requests instruction words from the bus controller before it is ready to execute them. If a bus error occurs on an instruction fetch, the CPU does not take the exception until it attempts to use that instruction word. Should an in-

tervening instruction cause a branch, or should a task switch occur, the bus error exception does not occur.

When the MCU recognizes a bus error condition, it terminates the current bus cycle in the normal way. **Figure 5-31** shows the timing of a bus error for the case in which $\overline{\text{DSACK}}$ is not asserted. **Figure 5-32** shows the timing for a bus error that is asserted after $\overline{\text{DSACK}}$. Exceptions are taken in both cases. Refer to the appropriate CPU reference manual for details of bus error exception processing.

When $\overline{\text{BERR}}$ is asserted after $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$ must be asserted within the time specified (refer to **APPENDIX A ELECTRICAL CHARACTERISTICS**) for purely asynchronous operation, or it must be asserted and remain stable during the sample window around the next falling edge of the clock after $\overline{\text{DSACK}}$ is recognized.

WARNING

If $\overline{\text{BERR}}$ is not stable at the specified time, the MCU may exhibit erratic behavior.

When $\overline{\text{BERR}}$ is asserted after $\overline{\text{DSACK}}$, data may be present on the bus but not be valid. This sequence may be used by systems that have memory error detection and correction logic and by external cache memories.

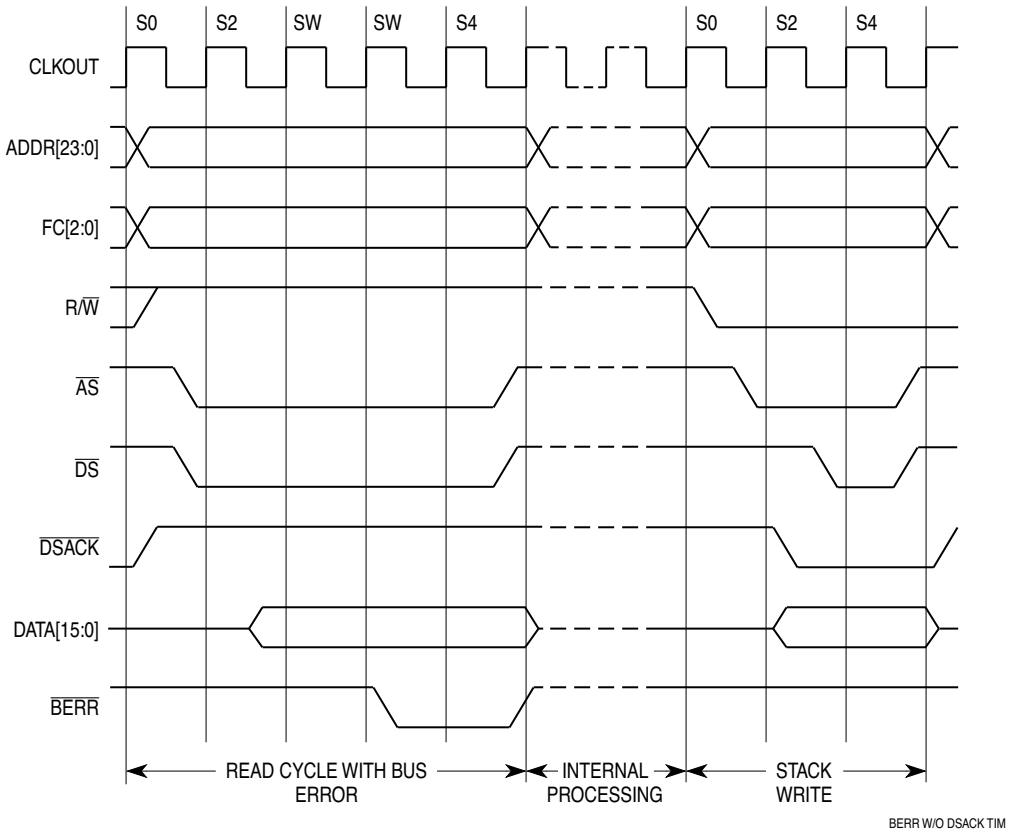


Figure 5-31 Bus Error Without $\overline{\text{DSACK}}$

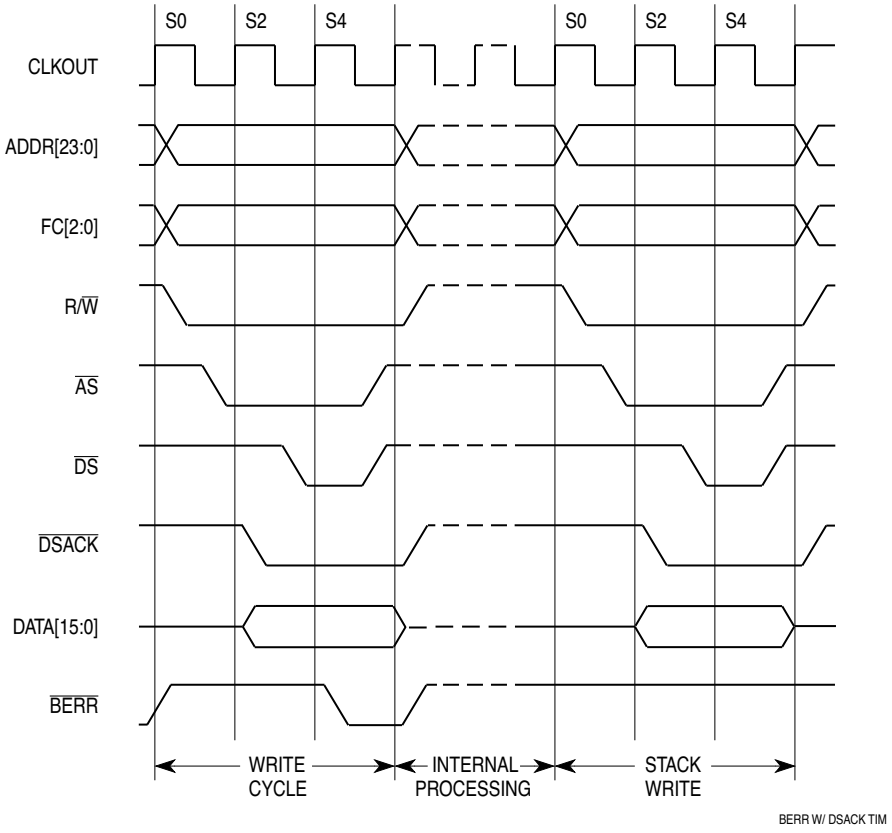


Figure 5-32 Late Bus Error with DSACK

5.9.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. However, a special case of bus error, called double bus fault, can abort exception processing.

BERR assertion is not detected until an instruction is complete. The BERR latch is cleared by the first instruction of the BERR exception handler. Double bus faults may occur under the following conditions, depending on the CPU:

- Bus error exception processing begins and a second BERR is detected before the first instruction of the first exception handler is executed.
- One or more bus errors occur before the first instruction after a reset exception is executed.
- A bus error occurs while the CPU is loading information from a bus error stack frame during a return from exception (RTE) instruction (CPU32-based MCUs only).

Multiple bus errors within a single instruction which can generate multiple bus cycles cause a single bus error exception after the instruction has executed.

Immediately after assertion of a second $\overline{\text{BERR}}$ signal, the MCU halts and drives the $\overline{\text{HALT}}$ line low. Only a reset can restart a halted MCU. However, bus arbitration can still occur. (Refer to **5.10 Bus Arbitration**.) A bus error or address error that occurs after exception processing has completed (during the execution of the exception handler routine, or later) does not cause a double bus fault. A bus cycle that is retried does not constitute a bus error or cause a double bus fault. The MCU continues to retry the same bus cycle as long as the external hardware requests it.

5.9.3 Retry Operation

When an external device asserts $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ during a bus cycle, the MCU enters the retry sequence, shown in **Figure 5-33**.

NOTE

On CPU16-based MCUs, assertion of $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ results in a bus error exception, rather than a retry sequence. The retry sequence is not available with these CPUs. Refer to the appropriate CPU manual or to the user's manual for the specific device for details.

A delayed retry (**Figure 5-34**), similar to the delayed bus error signal described previously, can also occur. The MCU terminates the bus cycle, places the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals in their inactive state, and does not begin another bus cycle until the $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ signals are negated by external logic. After a synchronization delay, the MCU retries the previous cycle using the same address, function codes, data (for a write), and control signals. The $\overline{\text{BERR}}$ signal should be negated before S2 of the read cycle to ensure correct operation of the retried cycle.

If $\overline{\text{BR}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ are all asserted on the same cycle, the EBI will enter the rerun sequence but first relinquishes the bus to an external master. Once the external master returns the bus and negates $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, the EBI runs the previous bus cycle. This feature allows an external device to correct the problem that caused the bus error (e.g., swap in a new page of memory) and then try the bus cycle again.

The MCU retries any read or write cycle of an indivisible read-modify-write operation separately; $\overline{\text{RMC}}$ remains asserted during the entire retry sequence. The MCU will not relinquish the bus while $\overline{\text{RMC}}$ is asserted. Any device that requires the MCU to give up the bus and retry a bus cycle during a read-modify-write cycle must assert $\overline{\text{BERR}}$ and $\overline{\text{BR}}$ only ($\overline{\text{HALT}}$ must remain negated). The bus error handler software should examine the read-modify-write bit in the special status word and take the appropriate action to resolve this type of fault when it occurs. (Refer to the CPU32 reference manual for details on the special status word. The CPU16 does not use this word.)

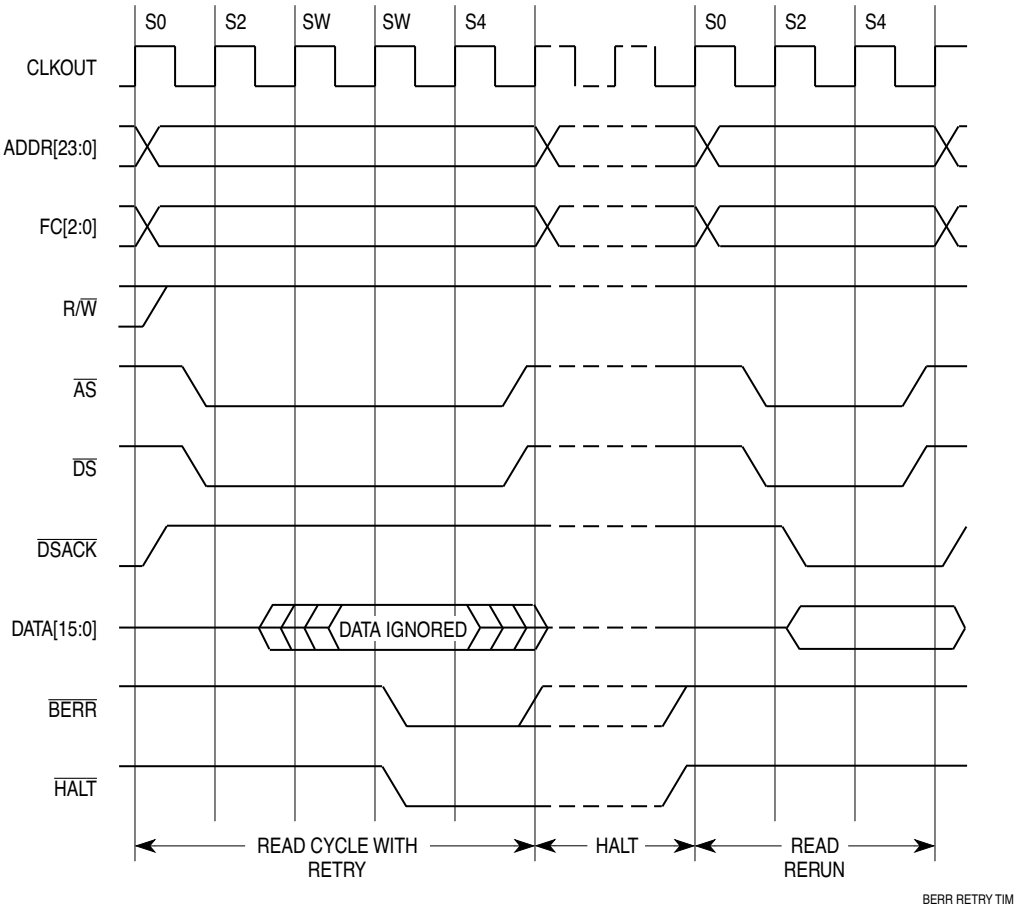


Figure 5-33 Retry Sequence

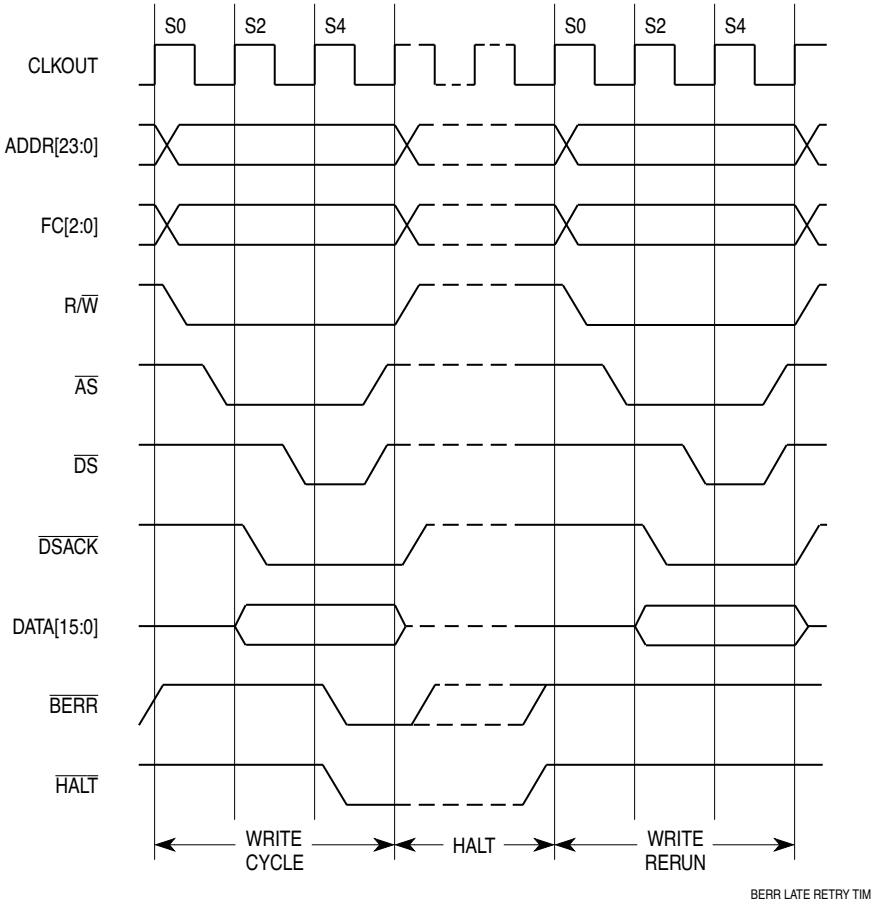


Figure 5-34 Late Retry Sequence

5.9.4 Halt Operation

When HALT is asserted (and BERR is not asserted, on MCUs that support the retry sequence), the MCU halts external bus activity after negation of DSACK. The MCU may complete the current word transfer in progress. For a long-word to byte transfer, this could be after S2 or S4. For a word to byte transfer, activity ceases after S2 (refer to **Figure 5-35**).

Negating and reasserting HALT in accordance with timing requirements provides single-step (bus cycle to bus cycle) operation. The HALT signal affects external bus cycles only, so that a program which does not use the external bus can continue executing. During dynamically-sized 8-bit transfers, external bus activity may not stop at the next cycle boundary.

Occurrence of a bus error while HALT is asserted causes the CPU to either process a bus error exception (on CPU16-based MCUs) or initiate a retry sequence (on CPU32-based MCUs). In the latter case, retry cycles must be anticipated while debugging in single-cycle mode. In dynamically sized 8-bit transfers, external bus activity may not stop at the next cycle boundary.

When the MCU completes a bus cycle while the $\overline{HALT}$ signal is asserted, the data bus goes to high-impedance state and the $\overline{AS}$ and $\overline{DS}$ signals are driven to their inactive states. Address, function code, size, and read/write signals remain in the same state.

The halt operation has no effect on bus arbitration. (Refer to **5.10 Bus Arbitration**.) However, when external bus arbitration occurs while the MCU is halted, address and control signals go to a high-impedance state. If $\overline{HALT}$ is still asserted when the MCU regains control of the bus, address, function code, size, and read/write signals revert to the previous driven states. The MCU cannot service interrupt requests while halted.

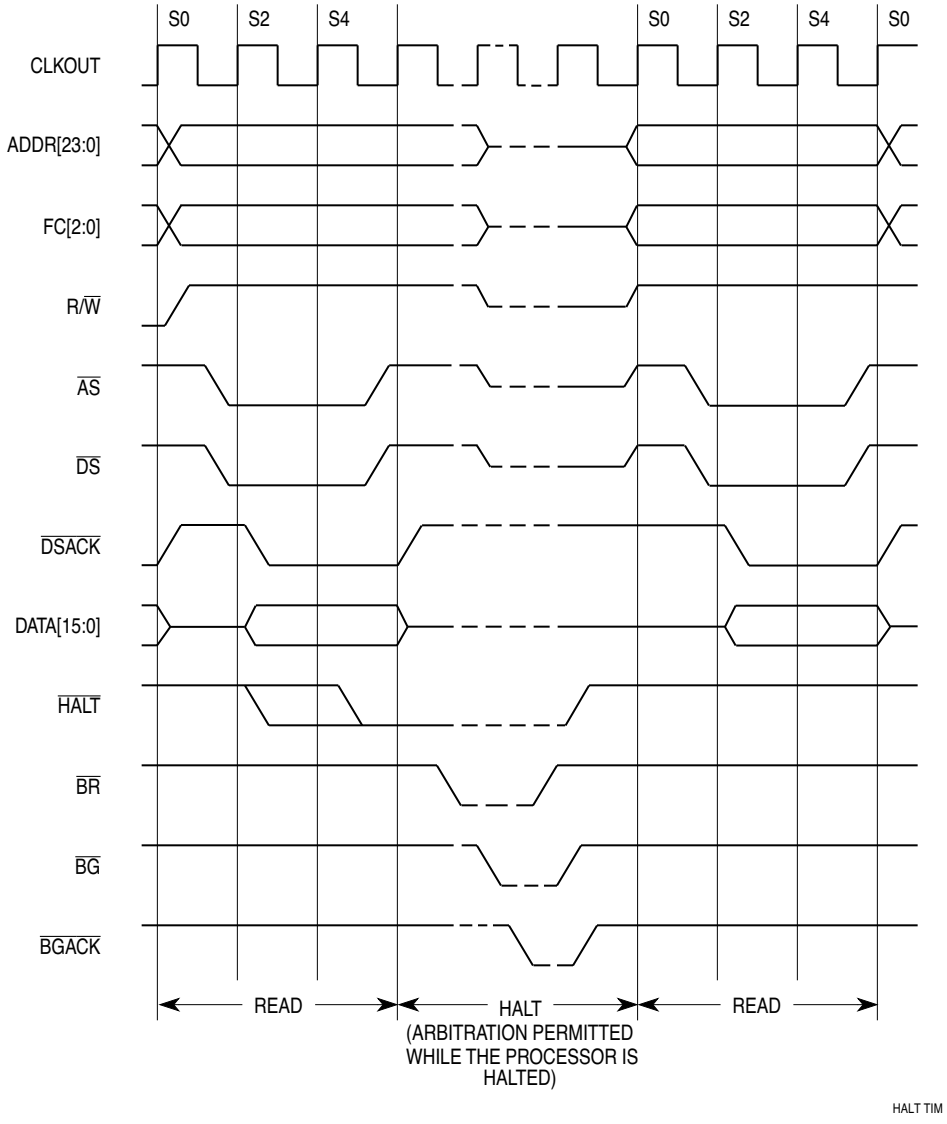


Figure 5-35 $\overline{HALT}$ Timing

5.10 Bus Arbitration

MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, during $\overline{\text{HALT}}$ assertion, and when the CPU has halted due to a double bus fault.

The bus controller in the MCU manages bus arbitration signals so that the MCU has the lowest priority. Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first.

External devices that need to obtain the bus must assert bus arbitration signals in the following sequence:

- An external device asserts the bus request signal ($\overline{\text{BR}}$).
- The MCU asserts the bus grant signal ($\overline{\text{BG}}$) to indicate that the bus is available.
- An external device asserts the bus grant acknowledge ($\overline{\text{BGACK}}$) signal to indicate that it has assumed bus mastership.

$\overline{\text{BR}}$ may be issued any time during a bus cycle or between cycles. $\overline{\text{BG}}$ is asserted in response to $\overline{\text{BR}}$. To guarantee operand coherency, $\overline{\text{BG}}$ is only asserted at the end of operand transfer. Additionally, $\overline{\text{BG}}$ is not asserted until the end of an indivisible read-modify-write operation (when $\overline{\text{RMC}}$ is negated), on CPU32-based MCUs. (CPU16-based MCUs do not provide an $\overline{\text{RMC}}$ signal.)

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives $\overline{\text{BG}}$. An external device must assert $\overline{\text{BGACK}}$ when it assumes mastership and must maintain $\overline{\text{BGACK}}$ assertion as long as it is bus master.

Two conditions must be met for an external device to assume bus mastership. The device must receive $\overline{\text{BG}}$ through the arbitration process, and $\overline{\text{BGACK}}$ must be inactive, indicating that no other bus master is active. This technique allows processing of bus requests during data transfer cycles.

$\overline{\text{BG}}$ is negated a few clock cycles after an external device asserts $\overline{\text{BGACK}}$. However, if bus requests are still pending after $\overline{\text{BG}}$ is negated, the MCU asserts $\overline{\text{BG}}$ again within a few clock cycles. This additional $\overline{\text{BG}}$ assertion allows external arbitration circuitry to select the next bus master before the current master has released the bus.

Figure 5-36 is a flow chart of bus arbitration for a single device.

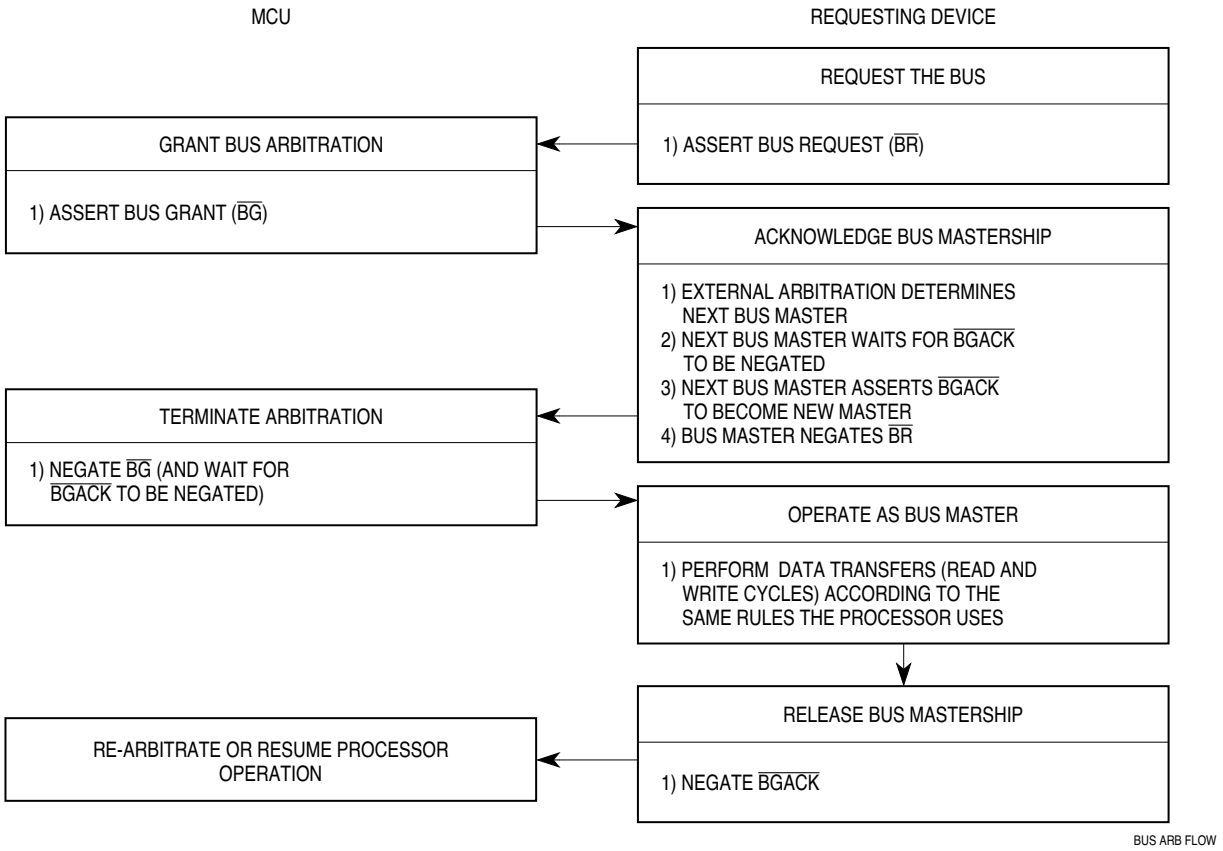


Figure 5-36 Bus Arbitration Flow Chart for Single Request

5.10.1 Bus Request

External devices capable of becoming bus masters request the bus by asserting the $\overline{BR}$ signal. In a system with a number of devices capable of bus mastership, a wired-OR connection can connect the bus request line from each device to the MCU. After it has completed the current operand transfer, the MCU asserts $\overline{BG}$, then releases the bus when $\overline{BGACK}$; is asserted.

If no acknowledge signal is received, the MCU remains bus master. This prevents interference with ordinary processing if the arbitration circuitry responds to noise or if an external device negates a request after mastership has been granted.

5.10.2 Bus Grant

The bus arbitration protocol supports operand coherency. When an operand transfer requires multiple bus cycles, the MCU does not release the bus until the entire transfer is complete. The assertion of bus grant is subject to the following constraints:

- The minimum time for $\overline{BG}$ assertion after $\overline{BR}$ assertion depends on internal synchronization (as specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**).
- During an external transfer, the MCU does not assert $\overline{BG}$ until after the last cycle of the transfer (determined by SIZ[1:0] and $\overline{DSACK}$[1:0] signals).

- During an external transfer, the MCU does not assert $\overline{BG}$ as long as $\overline{RMC}$ is asserted.
- When SHEN bits are both set and the CPU is making internal accesses, the MCU does not assert $\overline{BG}$ until the CPU finishes the internal transfers.

When a device is granted the bus and asserts $\overline{BGACK}$, it must also negate $\overline{BR}$. If $\overline{BR}$ remains asserted after assertion of $\overline{BGACK}$, the MCU assumes that another device is requesting the bus and prepares to assert $\overline{BG}$ again.

Externally, the $\overline{BG}$ signal can be routed through a daisy-chained network or a priority-encoded network. The MCU is not affected by the method of arbitration as long as the protocol is obeyed.

5.10.3 Bus Grant Acknowledge

When bus protocols are obeyed, a device becomes the active bus master when it asserts $\overline{BGACK}$. An external device cannot request and be granted the bus while another device is the active bus master. A device remains the bus master until it negates $\overline{BGACK}$. $\overline{BGACK}$ must not be negated until all required bus cycles are completed.

NOTE

On MCUs with a reduced pin-count SIM, the $\overline{BGACK}$ pin may not be supported. Refer to **SECTION 10 REDUCED PIN-COUNT SIM** and to the user's manual for the particular MCU for additional information.

When a device receives the bus and asserts $\overline{BGACK}$, it must also negate $\overline{BR}$. If $\overline{BR}$ remains asserted after $\overline{BGACK}$ assertion, the MCU assumes that another device is requesting the bus and prepares to issue another $\overline{BG}$.

Since external devices have priority, the MCU cannot regain control of the external bus until all pending external bus requests have been satisfied.

5.10.4 Bus Arbitration Pin State

Table 5-6 indicates the states of bus control pins when an external master has control of the bus. For pins with multiple functions, **Table 5-6** applies if the pin is configured for its bus control function, shown in the second column. Output bus control signals not used when an external master has control of the bus are placed in a disabled, high-impedance state, and inactive signals are placed in an inactive state.

Table 5-6 Bus Arbitration Pin State

Pin Mnemonic	Bus Control Signal	Pin Direction	Pin State when Bus Granted Away
ADDR23/CS10/ECLK	ADDR23	Output	High Impedance
ADDR[22:19]/CS[9:6]/PC[6:3]	ADDR[22:19]	Output	High Impedance
ADDR[18:0]	ADDR[18:0]	Output	High Impedance
AS/PE5	AS	Output	High Impedance
AVEC/PE2	AVEC	Input	Inactive
BERR	BERR	Input	Inactive
BG/CS1	BG	Output	Active
BGACK/CS2	BGACK	Input	Active
BR/CS0	BR	Input	Active
DATA[15:0]	DATA[15:0]	Depends on R/W	High Impedance
DS/PE4	DS	Output	High Impedance
DSACK[1:0]/PE[1:0]	DSACK[1:0]	Input	Inactive
FC[2:0]/CS[5:3]/[PC[2:0]	FC[2:0]	Output	High Impedance
R/W	R/W	Output	High Impedance
RMC/PE3	RMC	Output	High Impedance
SIZ[1:0]/PE[7:6]	SIZ[1:0]	Output	High Impedance

5.10.5 Bus Arbitration Control

The bus arbitration control unit in the MCU is implemented with a finite-state machine. All asynchronous inputs to the MCU are internally synchronized in a maximum of two CLKOUT cycles. **Figure 5-37** is the bus arbitration state diagram. Input signals labeled R and A are internal versions of the bus request and bus grant acknowledge signals that are internally synchronized to the system clock. The bus grant output is labeled G and the internal high-impedance control signal is labeled T. If T is true, the address, data, and control buses are placed in the high-impedance state after the next rising edge following the negation of AS and RMC.

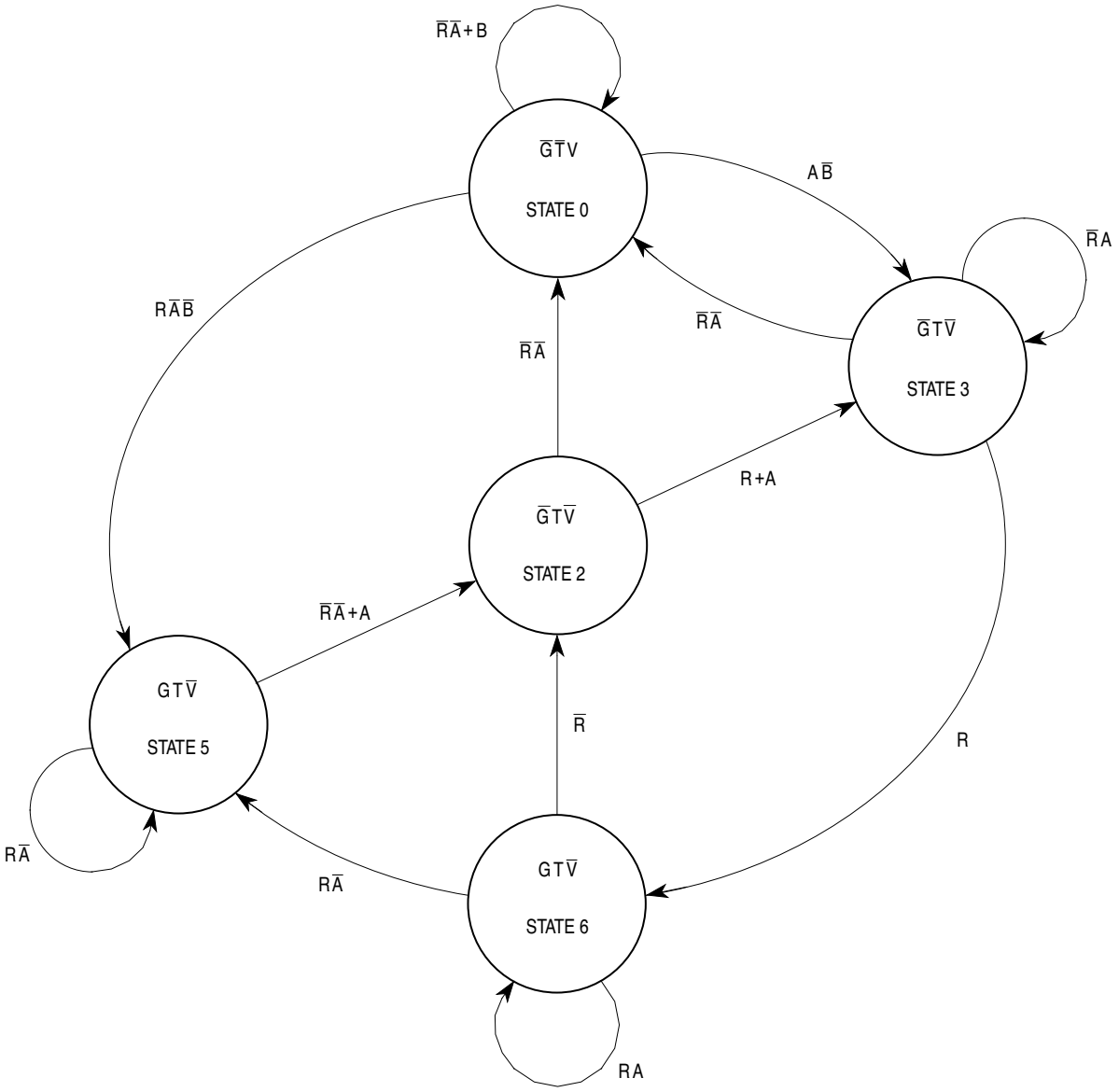


Figure 5-37 Bus Arbitration State Diagram

State changes occur on the next rising edge of CLKOUT after the internal signal is valid. The $\overline{BG}$ signal transitions on the falling edge of the clock after a state is reached during which G changes. The bus control signals (controlled by T) are driven by the MCU immediately following a state change, when bus mastership is returned to the MCU. State 0, in which G and T are both negated, is the state of the bus arbiter while the MCU is bus master. Request R and acknowledge A keep the arbiter in state 0 as long as they are both negated.

5.10.6 Factory Test (Slave) Mode Arbitration

This mode is used for factory production testing of internal modules. It is not supported for customer use, due to abnormal operating conditions that result. Factory test mode is enabled by holding DATA11 low during reset. In this mode, when $\overline{BG}$ is asserted, the MCU is slaved to an external tester that has full access to all internal registers.

5.11 Show Cycles

The MCU normally performs internal data transfers without affecting the external bus, but it is possible to "show" these transfers during debugging. $\overline{AS}$ is not asserted externally during show cycles.

Show cycles are controlled by the SHEN field in the SIMCR. (Refer to **3.1.6 SIM Configuration Register**.) This field is cleared by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but $\overline{AS}$ and $\overline{DS}$ are not asserted externally and external data bus pins are in high-impedance state during internal accesses.

When show cycles are enabled, $\overline{DS}$ is asserted externally during internal cycles, and internal data is driven out on the external data bus during writes. Since internal cycles normally continue to run when the external bus is granted away, one SHEN encoding halts internal bus activity while there is an external master.

SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SIM causes the value of the byte that is written to be driven out on both bytes of the data bus.

State 0 (S0) — Address and function codes become valid, R/$\overline{W}$ is driven to indicate a show read or write cycle, and the size pins indicate the number of bytes to transfer. During a read, the addressed peripheral drives the data bus, and the user must take care to avoid bus conflicts.

State 41 (S41) — $\overline{DS}$ is asserted to indicate that address information is valid.

State 42 (S42) — No change. The bus controller remains in S42 until the internal read cycle is complete.

State 43 (S43) — $\overline{DS}$ is negated to indicate that show data is valid on the next falling edge of CLKOUT. External data bus drivers are enabled so that data becomes valid on the external bus as soon as it is available on the internal bus.

State 0 (S0) — ADDR[23:0], FC[2:0], R/$\overline{W}$, and SIZ[1:0] pins change state to begin the next cycle. Data from the preceding cycle is valid through S0.

SECTION 6 INTERRUPTS

Interrupt recognition and servicing involve complex interaction between the system integration module, the central processing unit, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests from external devices. Refer to **SECTION 7 CHIP SELECTS** for more information.

The CPU handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Each exception has an assigned vector that points to an associated handler routine. During exception processing, the CPU fetches the vector assigned to the exception and executes the exception routine to which the vector points. Refer to the appropriate CPU manual for additional information on exception processing.

6.1 Sources of Interrupt

External devices or microcontroller modules can assert interrupt request signals. The SIM includes two sources of interrupt requests: the periodic interrupt timer and the external bus interface. The external bus interface routes external interrupt requests (i.e., requests received from the interrupt request pins $\overline{IRQ}[7:1]$) to the CPU. During interrupt arbitration (see **6.3 Interrupt Arbitration**), the CPU treats external interrupt requests as though they come from the SIM.

NOTE

MCUs with a reduced pin-count SIM may not have all the interrupt-request pins mentioned above. Refer to the user's manual for the specific MCU for details.

When the CPU receives simultaneous interrupt requests of the same level (see **6.2 Interrupt Level and Recognition**) from an interrupt request pin and the PIT, the PIT is given priority. The interrupt from the interrupt request pin remains pending until the next allowable interrupt time.

6.2 Interrupt Level and Recognition

Each of the interrupt request signals $\overline{IRQ}[7:1]$ corresponds to an interrupt priority level. $\overline{IRQ1}$ has the lowest priority and $\overline{IRQ7}$ the highest. For periodic timer interrupts, the PIRQ field in the periodic interrupt control register (PICR) determines the priority level. A priority level of 0 in the PICR means that PIT interrupts are inactive.

Interrupt recognition is based on the interrupt priority level and the interrupt priority mask value. The interrupt priority mask consists of three bits in the CPU status register (on CPU32-based MCUs) or condition code register (on CPU16-based MCUs). Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value from being recognized and processed.

$\overline{\text{IRQ7}}$, however, is always recognized, even if the mask value is %111. If simultaneous interrupt requests of different levels are made, the CPU recognizes the higher-level request.

$\overline{\text{IRQ[7:1]}}$ are active-low level-sensitive inputs. The level on the pin must remain asserted until an interrupt acknowledge cycle corresponding to that level is detected.

$\overline{\text{IRQ7}}$ is transition-sensitive as well as level-sensitive: a level-7 interrupt is not detected unless a falling edge transition is detected on the $\overline{\text{IRQ7}}$ line. This prevents redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted as well as each time the priority mask changes from %111 to a lower number while $\overline{\text{IRQ7}}$ is asserted.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis: to be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request with a priority equal to or lower than the current IP mask value is made, the CPU does not recognize the occurrence of the request.

6.3 Interrupt Arbitration

When the CPU detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it places the interrupt request level on the address bus and initiates a CPU space read cycle. The request level serves two purposes: it is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle pertains to them, and it is latched into the interrupt priority mask field in the CPU status register (on CPU32-based MCUs) or condition code register (on CPU16-based MCUs) in order to mask lower-priority interrupts during exception processing.

Modules that have requested interrupt service decode the interrupt priority mask value placed on the address bus at the beginning of the interrupt acknowledge cycle. If a module's request is at the specified priority mask level, it enters interrupt arbitration.

Arbitration between simultaneous requests of the same level is performed by means of serial contention between module interrupt arbitration (IARBI) field bit values. Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU to process a spurious interrupt exception when an interrupt from that module is recognized.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values in order to implement an arbitration scheme.

WARNING

Do not assign the same level and arbitration priority to more than one module. When two or more IARB fields have the same nonzero value, the CPU interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source is requesting service. This point is important for two reasons: the EBI does not transfer the CPU interrupt acknowledge cycle to the external bus unless the SIM wins contention, and failure to contend causes the interrupt acknowledge bus cycle to be terminated early, by a bus error.

When arbitration is complete, the module that wins contention must place an interrupt vector number on the data bus and terminate the bus cycle by asserting $\overline{\text{DSACK}}$. Alternately, an external device that wins arbitration can assert the autovector ($\overline{\text{AVEC}}$) signal to request that the CPU supply a vector number. The process is described in **6.4 Interrupt Acknowledge Bus Cycles**.

6.4 Interrupt Acknowledge Bus Cycles

The MCU acknowledges an interrupt request by performing a read cycle in CPU space to obtain the interrupt vector number. The interrupt acknowledge cycle differs from the read cycle described in **5.4.1 Read Cycles** in the following ways:

- FC[2:0] are set to %111, the CPU space encoding.
- ADDR[19:16] (the CPU space type field) are set to %1111, the interrupt acknowledge encoding.
- ADDR[3:1] are set to the interrupt request level.
- All remaining address bits are set.
- SIZ[1:0] and R/$\overline{\text{W}}$ are driven to indicate a single-byte read cycle.

Figure 6-1 shows the encoding of the address and function code lines for an interrupt acknowledge read cycle.

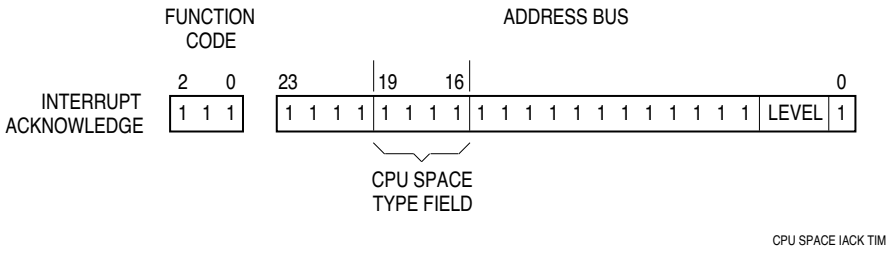


Figure 6-1 Interrupt Acknowledge Read Cycles

Interrupting devices must decode ADDR[3:1] to determine which device or devices enter interrupt arbitration. The responding devices must also decode SIZ[1:0] for dynamic bus allocation.

An interrupt acknowledge cycle is completed in one of three ways:

- When arbitration is complete, the module or device that wins contention places an interrupt vector number on the data bus and terminates the bus cycle with the appropriate $\overline{\text{DSACK}}$ signal.
- When arbitration is complete, an external device that wins contention asserts the autovector ($\overline{\text{AVEC}}$) signal.
- If no device or module enters interrupt arbitration, or if the device winning arbitration does not respond in time, the EBI bus monitor, if enabled, asserts the bus error signal ($\overline{\text{BERR}}$), and a spurious interrupt exception is taken.

Chip-select logic can be programmed to decode the interrupt acknowledge bus cycle, generate an interrupt acknowledge signal to the external device, and generate a $\overline{\text{DSACK}}$ response. Alternately, the chip-select circuit can be programmed to generate an $\overline{\text{AVEC}}$ response. Refer to **SECTION 7 CHIP SELECTS** for more information.

Figure 6-2 is a flow chart of the interrupt acknowledge cycle.

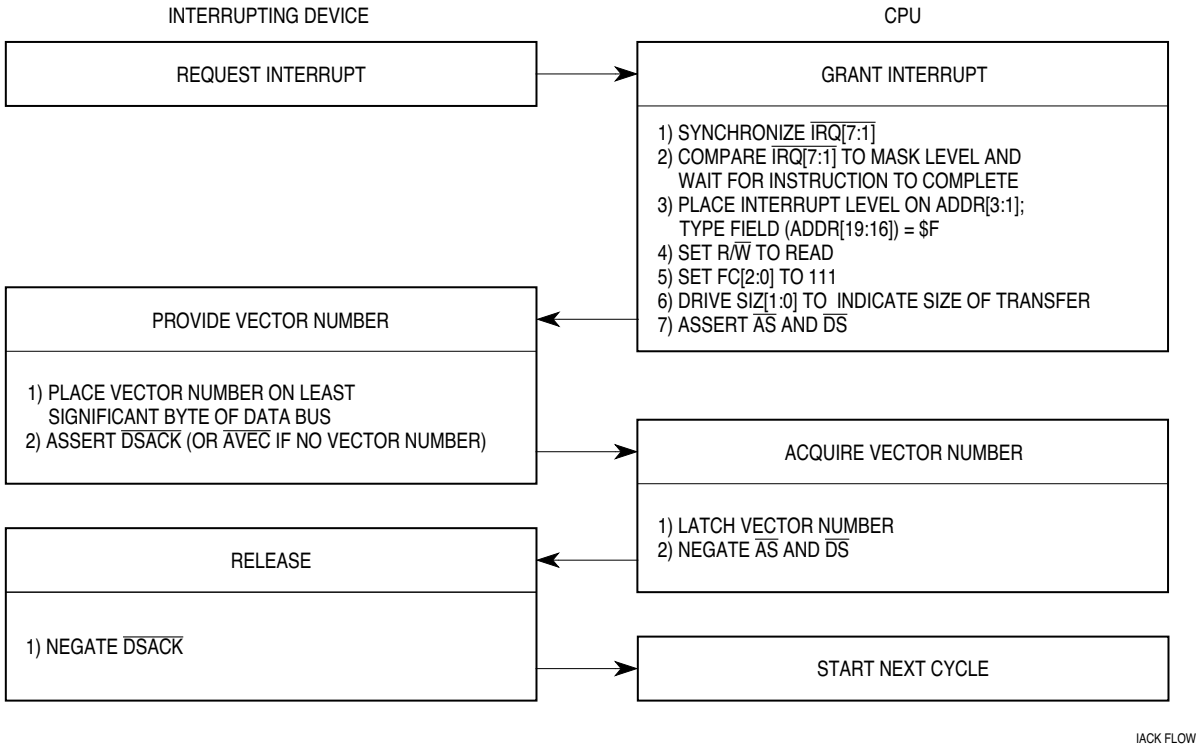


Figure 6-2 Interrupt Acknowledge Cycle Flowchart

6.4.1 Bus Cycle Terminated by DSACK Signals

If an external device with a vector number register wins arbitration, the device places the vector number on the data bus and asserts the appropriate DSACK signal to terminate the interrupt acknowledge cycle. The device must place its vector number on the least significant byte of its data port. A device with an 8-bit port must drive the vector number on DATA[15:8]; a device with a 16-bit port must drive the vector number on DATA[7:0].

Figure 6-3 shows the timing for an interrupt acknowledge cycle terminated with DSACK.

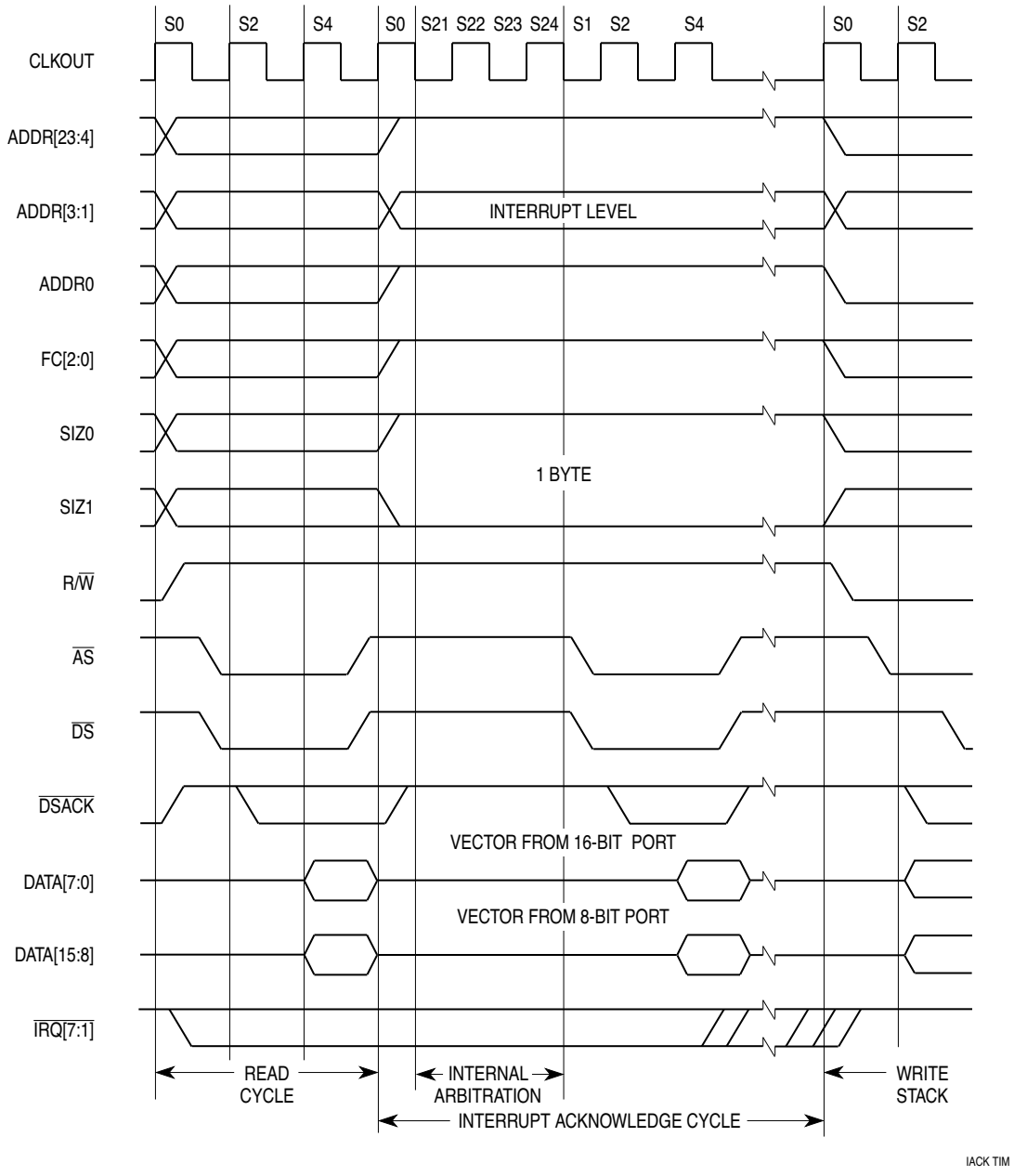


Figure 6-3 Interrupt Acknowledge Cycle Timing

Figure 6-4 indicates possible pin connections and external logic connecting the SIM and an interrupting external device that provides a vector number. The design shown in **Figure 6-4** can be simplified by using SIM chip selects to decode the function code and address lines and supply the interrupt acknowledge signal. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles**.

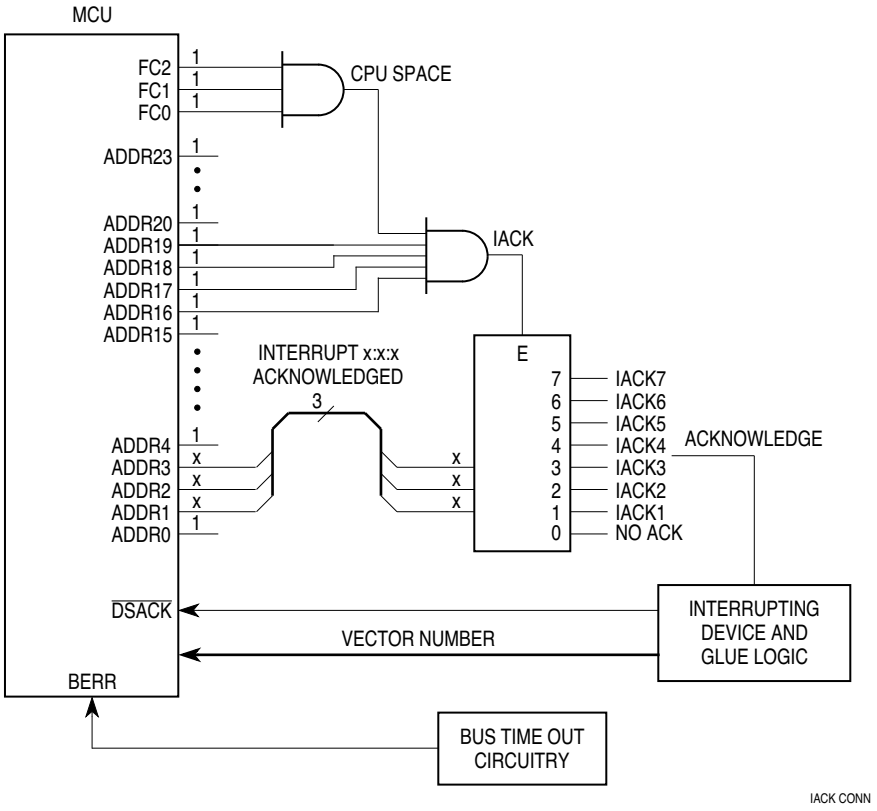


Figure 6-4 External Connections for Interrupt Processing

6.4.2 Bus Cycle Terminated by $\overline{\text{AVEC}}$ Signal

An external interrupting device requests an automatically generated vector (autovector) by asserting the $\overline{\text{AVEC}}$ signal to terminate an interrupt acknowledge cycle. $\overline{\text{DSACK}}$ signals must not be asserted during an interrupt acknowledge cycle terminated by $\overline{\text{AVEC}}$. If the $\overline{\text{AVEC}}$ pin is permanently wired low (asserted), the CPU generates an autovector whenever an interrupt (of any priority, from any external source) is acknowledged.

When $\overline{\text{AVEC}}$ is asserted, the CPU ignores the state of the data bus and generates a vector number. Refer to the appropriate CPU manual for information on determining the vector number and vector address. Seven autovectors are available, one for each of the seven interrupt request signals.

Figure 6-5 shows the timing for an autovector operation.

Chip-select logic can be programmed to decode this bus cycle and generate an internal $\overline{\text{AVEC}}$ response when an external interrupt request is made. The interrupting device does not have to respond in this case. Chip-select logic is typically used to generate an internal autovector signal when the corresponding chip-select pin is used for an alternate function or for general-purpose output. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for more information.

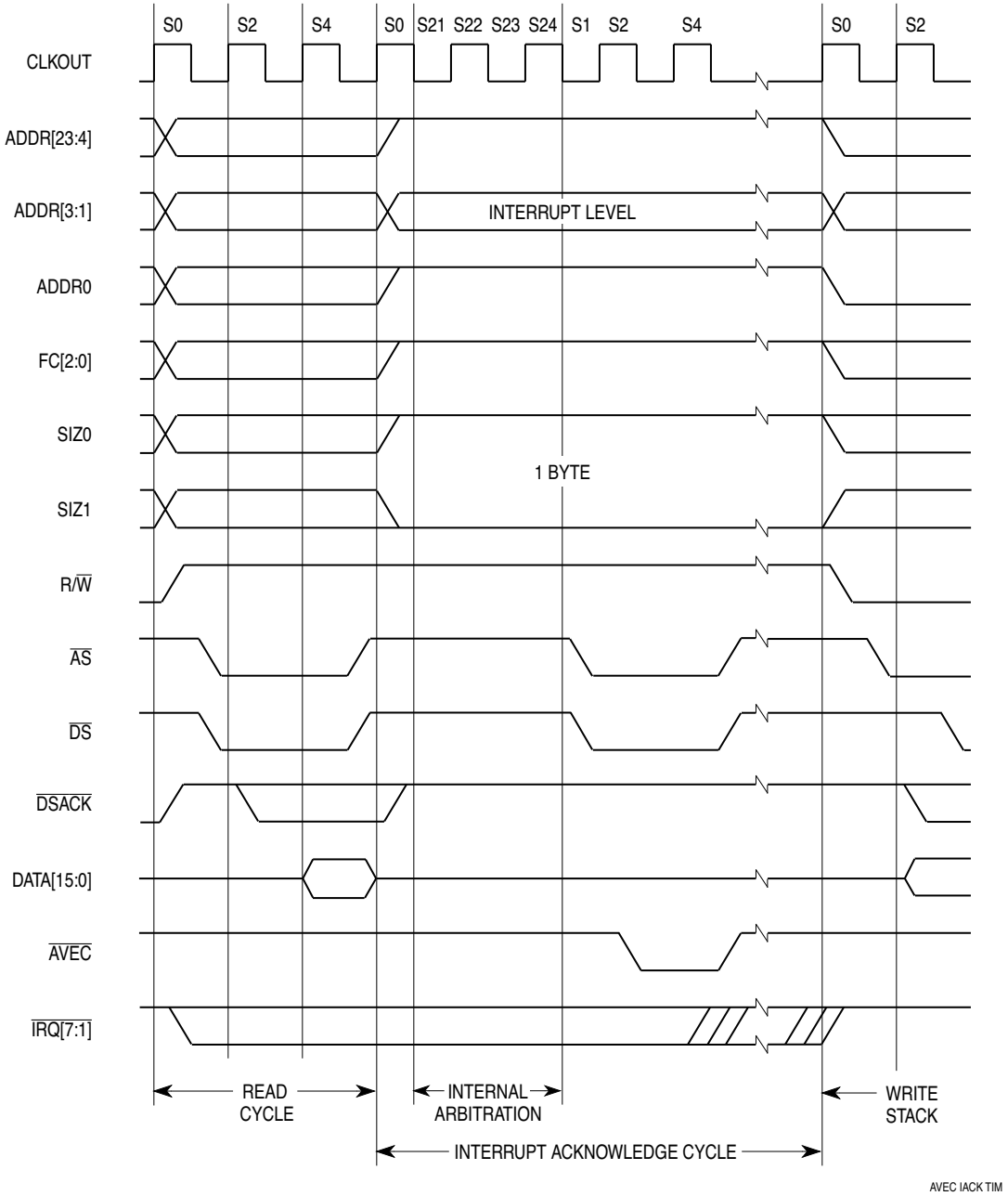


Figure 6-5 Autovector Timing

6.4.3 Spurious Interrupt Cycle

When an interrupt request is made, but no IARB field value is asserted in response to the interrupt acknowledge cycle, the spurious interrupt monitor asserts the $\overline{\text{BERR}}$ signal internally to prevent vector acquisition. When a responding device does not terminate an interrupt acknowledge cycle with $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$, the bus monitor asserts $\overline{\text{BERR}}$ internally. The CPU automatically generates the spurious interrupt vector number ($F) in both cases.

6.5 Interrupt Processing Summary

A summary of the entire interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.
B. The processor state is stacked. On CPU16-based MCUs, the PK extension field in the condition code register is cleared. On CPU32-based MCUs, the S bit in the status register is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing.
C. The interrupt acknowledge cycle begins:
    1. FC[2:0] are driven to %111 (CPU space) encoding.
    2. The address bus is driven as follows: ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.
    3. The request level is latched from the address bus into the interrupt priority mask field in the status or condition code register.
D. Modules or external peripherals that have requested interrupt service decode the priority value in ADDR[3:1]. Each module or device with a request level equal to the value in ADDR[3:1] enters interrupt arbitration. When there is no arbitration, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.
E. After arbitration, the interrupt acknowledge cycle is completed in one of three ways:
    1. The interrupt source that wins arbitration supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU acquires the vector number.
    2. The $\overline{\text{AVEC}}$ signal is asserted (the external interrupt source can assert the signal, or the pin can be tied low), and the CPU generates an autovector number corresponding to the interrupt priority level.
    3. The bus monitor or external device asserts $\overline{\text{BERR}}$, and the CPU generates the spurious interrupt vector number.
F. The vector number is converted to a vector address.
G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

SECTION 7 CHIP SELECTS

Typical microcontrollers require additional hardware to provide external chip-select and address decode signals. The SIM includes 12 programmable chip-select circuits that can provide 2-clock-cycle (fast termination) to 16-clock-cycle (13-wait-state) access to external memory and peripherals. Address block sizes of 2 Kbytes to 1 Mbyte (on CPU32-based MCUs) or 2 Kbytes to 512 Kbytes (on CPU16-based MCUs) can be selected.

NOTE

MCUs with reduced pin-count SIMs may not have all 12 chip-select pins available. Consult the user's manual for the particular MCU for details.

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Chip select logic can generate $\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ signals internally. Each signal can also be synchronized with the ECLK signal available as a programming option on ADDR23.

Chip-select registers include two global pin assignment registers, a base address register and option register for each chip-select circuit, and a data register. The pin assignment registers assign pins individually for chip-select operation, discrete output, or alternate functions. The pin data register controls the state of pins programmed as discrete outputs. The base address registers define the base address and block size to which the chip select responds. The option registers determine timing of and conditions for assertion of chip-select signals.

7.11 Interfacing Example with Chip Selects provides a diagram of a basic system that uses chip selects. **Figure 7-1** is a functional diagram of a single chip-select circuit.

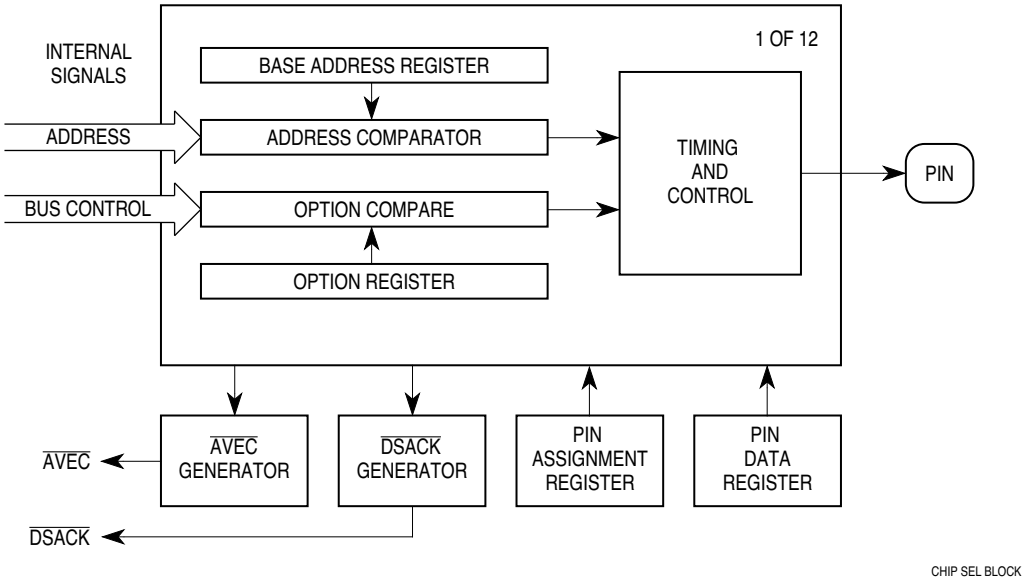


Figure 7-1 Chip-Select Circuit Block Diagram

7.1 Chip-Select Options

Chip-select option registers determine timing of and conditions for assertion of chip-select signals. Constraints set by fields in the option register and in the base address register must all be satisfied in order to assert a chip-select signal, and to provide $\overline{\text{DSACK}}$ or autovector support.

The following fields in the option registers specify the conditions for assertion of the chip-select signal. These conditions must all be satisfied before chip-select logic will respond:

- The BYTE field determines whether to assert the chip select for an upper-byte access, lower-byte access, both, or neither (disabled).
- The R/$\overline{\text{W}}$ field specifies whether to assert the chip select during a read cycle, write cycle, or both.
- The SPACE field specifies whether to assert the chip select during a user-space access, supervisor-space access, user or supervisor access, or CPU-space access.

The following fields specify chip-select assertion timing:

- $\overline{\text{DSACK}}$ specifies the number of wait states to insert before the chip-select circuit asserts $\overline{\text{DSACK}}$, or specifies that the external device must provide the $\overline{\text{DSACK}}$ signal.
- STRB specifies whether chip select assertion is synchronous with $\overline{\text{AS}}$ or $\overline{\text{DS}}$. (This bit applies only when MODE = 0.)
- MODE determines whether the chip-select cycle is synchronous with ECLK or emulates an asynchronous external bus cycle.

The following fields determine chip-select operation during interrupt acknowledge cycles:

- $\overline{\text{AVEC}}$ determines whether to generate $\overline{\text{AVEC}}$ internally when match conditions specified in SPACE and IPL fields and base address register are satisfied.
- IPL specifies the interrupt priority level to which the chip select responds.
- SPACE must be set to %00 (CPU space) for the chip-select circuit to respond to interrupt acknowledge cycles.

Table 7-1 is a summary of option register functions.

Table 7-1 Option Register Function Summary

MODE	BYTE	R/$\overline{\text{W}}$	STRB	$\overline{\text{DSACK}}$	SPACE	IPL	$\overline{\text{AVEC}}$
0 = Async.*	00 = Disable	00 = Rsvd	0 = $\overline{\text{AS}}$	0000 = 0 wait states	00 = CPU	000 = All*	0 = Off*
1 = Sync.	01 = Lower	01 = Read	1 = $\overline{\text{DS}}$	0001 = 1 wait state	01 = User	001 = Level 1	1 = On
	10 = Upper	10 = Write		0010 = 2 wait states	10 = Supv	010 = Level 2	
	*11 = Both	11 = Both		0011 = 3 wait states	11 = S/U*	011 = Level 3	
				0100 = 4 wait states		100 = Level 4	
				0101 = 5 wait states		101 = Level 5	
				0110 = 6 wait states		110 = Level 6	
				0111 = 7 wait states		111 = Level 7	
				1000 = 8 wait states			
				1001 = 9 wait states			
				1010 = 10 wait states			
				1011 = 11 wait states			
				1100 = 12 wait states			
				1101 = 13 wait states			
				1110 = Fast terminate			
				1111 = External			

*Use this value when function is not required for chip-select operation.

For additional information on the MODE and STRB fields, refer to **7.5 Chip-Select Timing**. For more information on the BYTE field, refer to **7.6 Chip Selects and Dynamic Bus Sizing**. For information on the $\overline{\text{DSACK}}$ field, see **7.5 Chip-Select Timing** and **7.7 Fast Termination Cycles**. For details on the IPL, $\overline{\text{AVEC}}$, and SPACE fields, refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles**. Diagrams of the chip-select option registers are provided in **7.10 Chip-Select Register Diagrams**.

7.2 Chip-Select Base Addresses

Each chip select has an associated base address register. The base address register specifies the base address and block size of the memory or peripheral enabled by the chip select. A base address is the lowest address in the block of addresses enabled by a chip select. Block size is the extent of the address block above the base address.

Block sizes of 2 Kbytes to 1 Mbyte can be selected. Address blocks for separate chip-select functions can overlap.

NOTE

On CPU16-based MCUs, because the logic states of ADDR[23:20] follow that of ADDR19, a 1-Mbyte block size encoding is not supported. In addition, on these MCUs be sure that the ADDR[23:20] bits in the base address register have the same value as ADDR19, to conform with the logic states of the corresponding address bus pins.

The BLKSZ field determines which bits in the base address field are compared to corresponding bits on the address bus during an access. Provided other constraints determined by option register fields are also satisfied, when a match occurs, the associated chip-select signal is asserted. **Table 7-2** shows BLKSZ encoding.

Table 7-2 Block Size Encoding

BLKSZ[2:0]	Block Size	Address Lines Compared
000	2 K	ADDR[23:11]
001	8 K	ADDR[23:13]
010	16 K	ADDR[23:14]
011	64 K	ADDR[23:16]
100	128 K	ADDR[23:17]
101	256 K	ADDR[23:18]
110	512 K	ADDR[23:19]
111	1 M *	ADDR[23:20]

*Maximum block size = 512 Kbytes on CPU16-based MCUs, in which ADDR[23:20] = ADDR19

The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of block size. Base address register diagrams show how base register bits correspond to address lines. (See **7.10 Chip-Select Register Diagrams**.)

After reset, the MCU fetches initialization vectors from word addresses beginning at memory location $000000. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A memory device containing initialization can be automatically enabled by $\overline{\text{CSBOOT}}$ after a reset. The block size field in CSBARBT has a reset value of 512 Kbytes. Refer to **7.9 Chip-Select Reset Operation** for more information.

When programming a chip-select circuit to respond to interrupt acknowledge cycles, program the base address field (bits [15:3]) in the base address register to all ones. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for more information.

7.3 Pin Assignments and Discrete Output

The chip-select pin assignment registers (CSPAR1–0) contain 12 two-bit fields ($\overline{\text{CS}}$[10:0] and $\overline{\text{CSBOOT}}$) that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 7-3**.

Table 7-3 Chip-Select Pin Functions

16-Bit Chip Select	8-Bit Chip Select	Alternate Function	Discrete Output or ECLK
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—
$\overline{\text{CS0}}$	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CS1}}$	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
$\overline{\text{CS2}}$	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
$\overline{\text{CS3}}$	$\overline{\text{CS3}}$	FC0	PC0
$\overline{\text{CS4}}$	$\overline{\text{CS4}}$	FC1	PC1
$\overline{\text{CS5}}$	$\overline{\text{CS5}}$	FC2	PC2
$\overline{\text{CS6}}$	$\overline{\text{CS6}}$	ADDR19	PC3
$\overline{\text{CS7}}$	$\overline{\text{CS7}}$	ADDR20	PC4
$\overline{\text{CS8}}$	$\overline{\text{CS8}}$	ADDR21	PC5
$\overline{\text{CS9}}$	$\overline{\text{CS9}}$	ADDR22	PC6
$\overline{\text{CS10}}$	$\overline{\text{CS10}}$	ADDR23	ECLK

Table 7-4 shows pin assignment field encoding. Pins that have no discrete output or ECLK function do not use the %00 encoding.

Table 7-4 Pin Assignment Field Encoding

Bit Field	Description
00	Discrete Output or ECLK
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

Port size is involved in dynamic bus sizing and determines which $\overline{\text{DSACK}}$ signal the chip-select circuit asserts during a bus cycle. Refer to **7.6 Chip Selects and Dynamic Bus Sizing** for more information.

A pin programmed as a discrete output drives an external signal to the value specified in the port C data register. No discrete output function is available on pins $\overline{\text{CSBOOT}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, or $\overline{\text{BGACK}}$. ADDR23 provides ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ internally on an address and control signal match. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

Refer to **7.9 Chip-Select Reset Operation** for information on pin assignments at reset.

7.4 Chip-Select Operation

Pins come out of reset assigned to their chip select function. Before a chip select (other than $\overline{\text{CSBOOT}}$) can respond to a memory access, however, its option register and base address register must be programmed. The BYTE fields in the option registers for $\overline{\text{CS}}$[10:0] are cleared during reset. These fields must be set to nonzero values to

enable the associated chip selects. However, $\overline{\text{CSBOOT}}$ is made active out of reset so that it can be used as a chip select for the initialization memory. (See **7.9 Chip-Select Reset Operation** for information on the initial states of the $\overline{\text{CSBOOT}}$ base address and option registers.)

Disabling the chip selects prevents chip-select signal assertion, even when all other constraints are satisfied. The associated pin is driven high, and internal chip-select logic cannot assert associated signals, such as $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$, internally.

Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of the reset signal. (Refer to **7.9 Chip-Select Reset Operation** for more information.)

When the MCU makes a memory access, each enabled chip-select circuit compares the following items:

1. Appropriate ADDR bits to the base address field in the base address register.
2. Function code signals to the SPACE field in the option register.
3. Read/write status to R/$\overline{\text{W}}$ field in the option register.
4. ADDR0 or SIZ bits to the BYTE field in the option register (16-bit ports only).
5. Priority of the interrupt being acknowledged (ADDR[3:1]) to the IPL field in the option register (when the access is an interrupt acknowledge cycle).

When a match occurs, the chip-select signal is asserted. The signal is asserted at the same time as) or $\overline{\text{DS}}$ assertion if MODE = 0 in the chip-select option register; chip-select assertion is synchronized with ECLK if MODE = 1. Chip-select signals are active low.

If a chip-select function is given the same address as an internal microcontroller module or an internal memory array, access to the internal module or array has priority. The chip-select signal is not asserted, and no external bus cycle occurs.

Figure 7-2 is a flow diagram for the assertion of chip select.

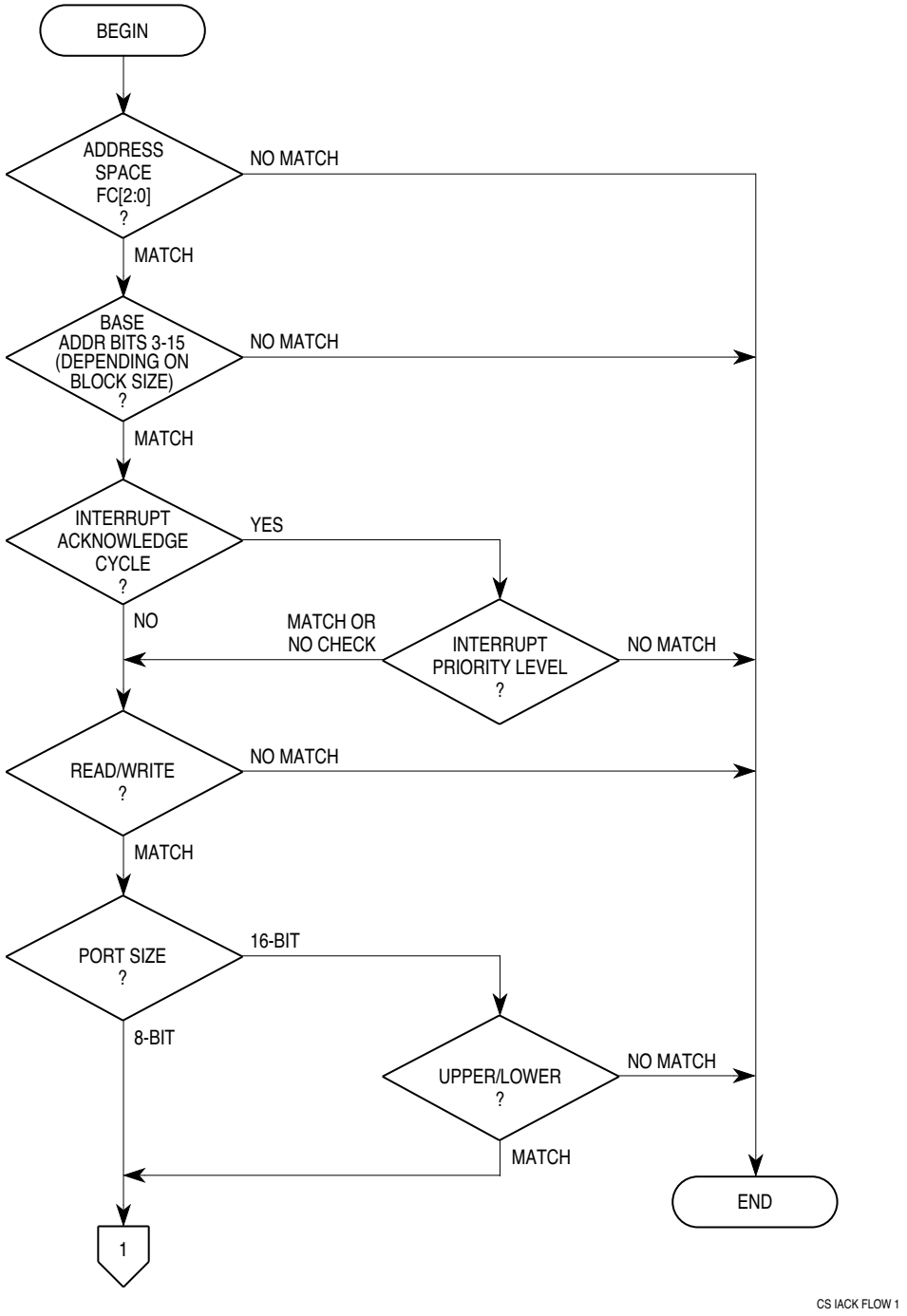


Figure 7-2 Flow Diagram for Chip Select (Sheet 1 of 3)

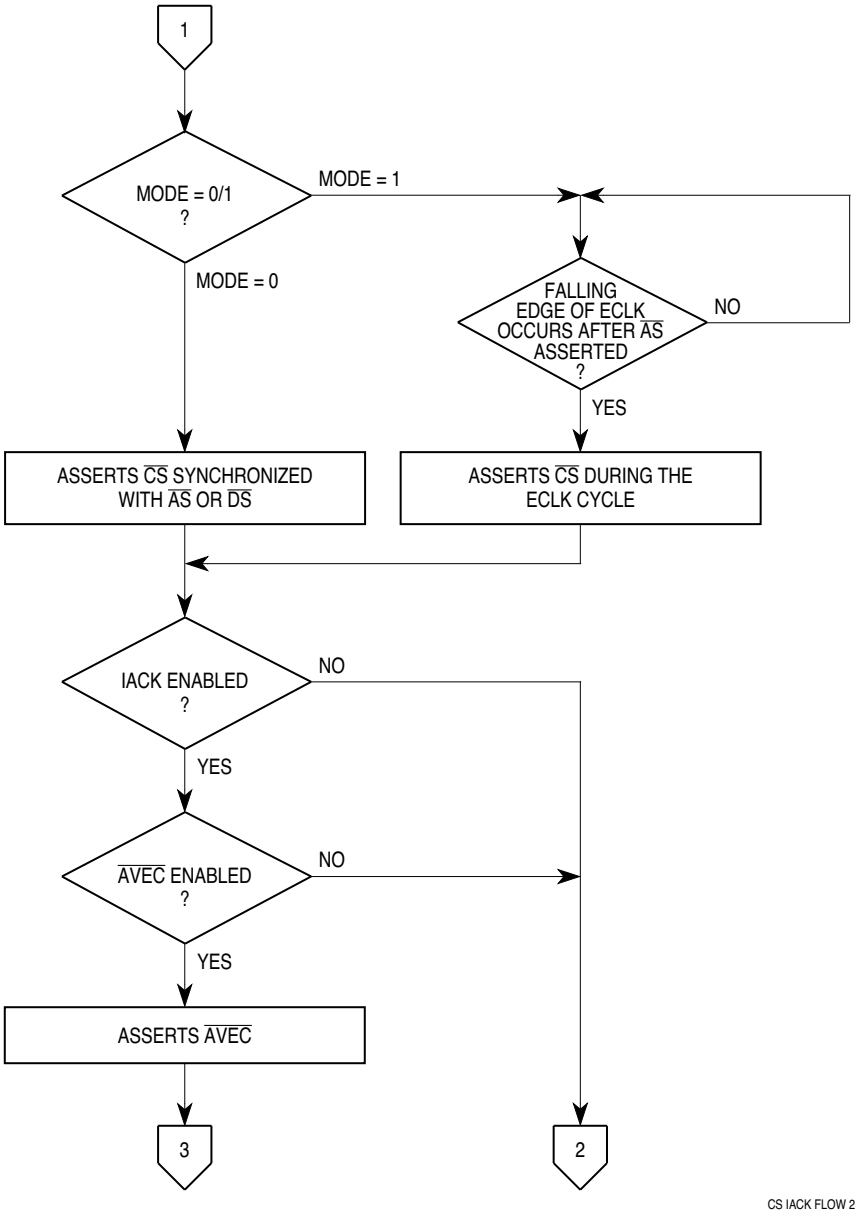


Figure 7-2 Flow Diagram for Chip Select (Sheet 2 of 3)

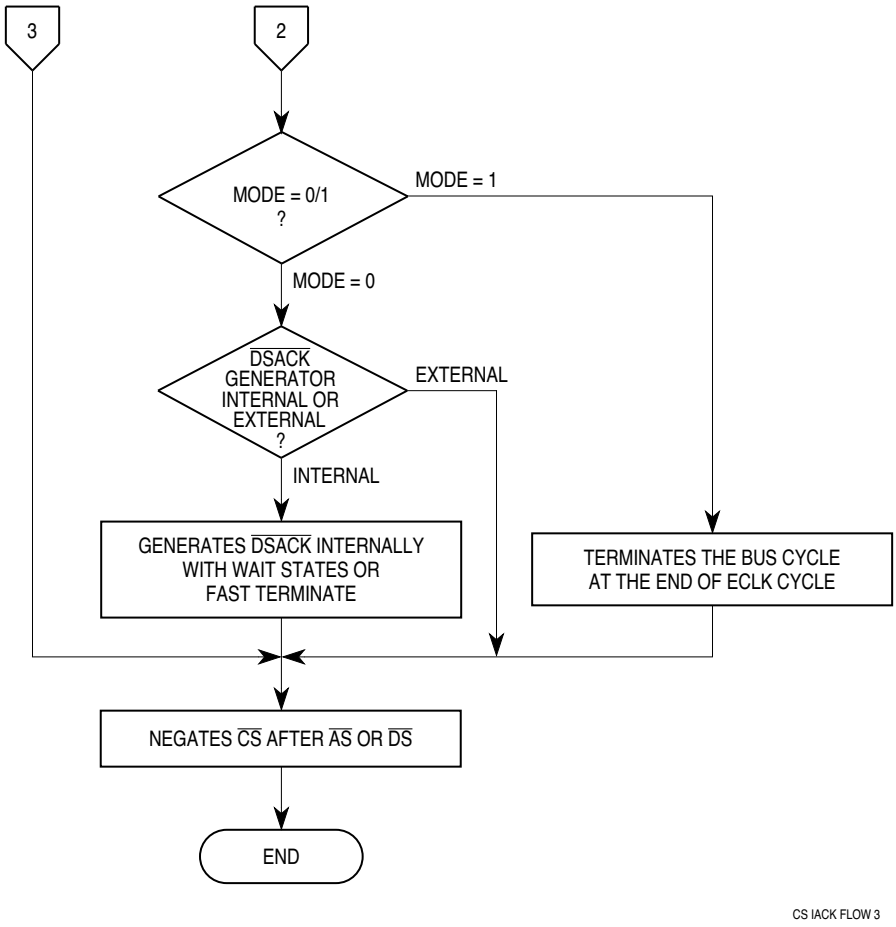


Figure 7-2 Flow Diagram for Chip Select (Sheet 3 of 3)

7.5 Chip-Select Timing

The MODE bit in the chip-select option register determines whether chip-select operation emulates asynchronous bus operation or is synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23. (Refer to **SECTION 4 SYSTEM CLOCK** for more information on ECLK.) When the MODE bit is programmed to emulate asynchronous bus operation, the $\overline{\text{DSACK}}$ and STRB fields further define chip-select timing.

7.5.1 Synchronization with $\overline{\text{AS}}$ or $\overline{\text{DS}}$

When MODE = 0 in the associated chip-select option register, chip-select operation emulates asynchronous external bus operation. The chip-select signal is asserted at the same time as $\overline{\text{AS}}$ or $\overline{\text{DS}}$, depending on the value in the STRB field in the option register. As in an asynchronous bus cycle, the chip-select cycle must be terminated by a data and size acknowledge ($\overline{\text{DSACK}}$) signal or by an autovector ($\overline{\text{AVEC}}$) signal.

The value of the $\overline{\text{DSACK}}$ field in the associated chip-select option register determines whether $\overline{\text{DSACK}}$ is generated internally. If it is, the $\overline{\text{DSACK}}$ field determines the number of wait states inserted before internal $\overline{\text{DSACK}}$ assertion.

A wait state has a duration of one clock cycle. The wait states are inserted beginning with S3 of the external bus cycle. An encoding of zero wait states corresponds to a three-clock-cycle bus.

Fast termination encoding corresponds to a two-clock-cycle bus access. MCU modules typically respond at this rate; the fast termination encoding is used to access fast external devices. With fast termination encoding, the bus cycle can be terminated at S3. (Refer to **7.7 Fast Termination Cycles**.)

Cycles are terminated by the first $\overline{\text{DSACK}}$ signal that occurs. If an external $\overline{\text{DSACK}}$ signal occurs during internal wait state generation, the bus cycle terminates immediately. If the externally generated acknowledge option is selected, the MCU waits indefinitely for external $\overline{\text{DSACK}}$ assertion.

If multiple chip selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all of the associated $\overline{\text{DSACK}}$ fields should be programmed for the same number of wait states. (Alternately, all but one of the associated $\overline{\text{DSACK}}$ fields can be programmed for external $\overline{\text{DSACK}}$ generation, and the remaining $\overline{\text{DSACK}}$ field to the desired number of wait states.) This prevents a conflict on the internal bus when the wait states are loaded into the $\overline{\text{DSACK}}$ counter shared by all chip selects.

7.5.2 Synchronization with ECLK

When MODE = 1 in the associated chip-select option register, chip-select assertion is synchronized to the MCU ECLK output. When a match condition occurs, the chip-select circuit signals the EBI that an ECLK cycle is pending. When the EBI determines that bus timing constraints are satisfied, the chip-select signal is asserted. Transfers of word and long-word data to an 8-bit port are performed consecutively, without insertion of additional ECLK cycles. The bus monitor time-out period must be longer than the number of clock cycles required for two ECLK cycles. (Refer to **SECTION 3 SYSTEM CONFIGURATION AND PROTECTION** for more information.) Because chip-select cycles synchronized to ECLK are not terminated by data and size acknowledge signals, the $\overline{\text{DSACK}}$ field has no effect in this mode. The $\overline{\text{AVEC}}$ bit must not be used, since autovector response timing can vary due to ECLK synchronization with the internal system clock.

7.6 Chip Selects and Dynamic Bus Sizing

Chip-select logic works with the external bus interface to perform dynamic bus sizing. Pin assignment fields (in the pin assignment registers) assign port sizes of 8 or 16 bits to the devices assigned to the chip selects. (Refer to **7.3 Pin Assignments and Discrete Output** for pin assignment field encoding.) Port size assignment determines which signal the chip-select logic asserts ($\overline{\text{DSACK1}}$ or $\overline{\text{DSACK0}}$) after the specified number of wait states elapse during a chip-select access.

Chip select logic also decodes the internal SIZ[1:0] signals to determine which byte or bytes of the data bus to use during a data transfer. In addition, for 16-bit ports, the BYTE field (in the chip-select option register) determines whether the chip select is asserted for upper byte accesses, lower byte accesses, or both. **Table 7-5** shows BYTE field encoding.

Table 7-5 BYTE Field Encoding

BYTE[1:0]	Meaning
00	Disable
01	Assert when ADDR = 0 (lower byte)
10	Assert when ADDR = 1 (upper byte)
11	Assert when ADDR = 0 or 1 (either byte)

BYTE field encoding options are used to generate chip-select signals for word and single-byte transfers to 16-bit ports. For example, two chip-select lines can be used to select 8-bit banks in a 16-bit memory. To do this, program two chip-select base address registers with the same base address, then set up the individual lines for byte access. Program both option registers identically except for the BYTE fields: use the upper byte option for one line and the lower byte option for the other.

Refer to **7.11 Interfacing Example with Chip Selects** for an illustration of dynamic bus sizing using chip selects.

7.7 Fast Termination Cycles

With an external device that has a fast access time, the chip-select circuit fast-termination option can provide a two-cycle external bus transfer. Select this option by assigning a value of %1110 to the field in the appropriate chip-select option register. Fast termination cycles are only available in conjunction with chip selects.

If multiple chip selects are to be used to select the same device that can support fast termination, and match conditions can occur simultaneously, select the fast-termination option in the $\overline{\text{DSACK}}$ field of each associated chip-select option register. Alternately, select fast termination in one of the $\overline{\text{DSACK}}$ fields for external termination.

Fast termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the CPU asserts an address and the SIZ[1:0] signals. When $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ are valid, a peripheral device places data on the bus during a read cycle or latches data from the bus during a write cycle. At the appropriate time, chip-select logic asserts data and size acknowledge signals. Two clock states (S2 and S3) that are normally required for external handshaking are eliminated during fast termination cycles.

To use the fast-termination option, an external device should be fast enough to have data ready, within the specified setup time, by the falling edge of S4.

When using the fast-termination option, data strobe is asserted only in a read cycle and not in a write cycle. The STRB field in the chip-select option register used must be programmed to address strobe in order to assert the chip select during a fast-termination write cycle.

Figure 7-3 shows the $\overline{\text{DSACK}}$ timing for a read cycle with two wait states, followed by a fast-termination read cycle and a fast-termination write cycle.

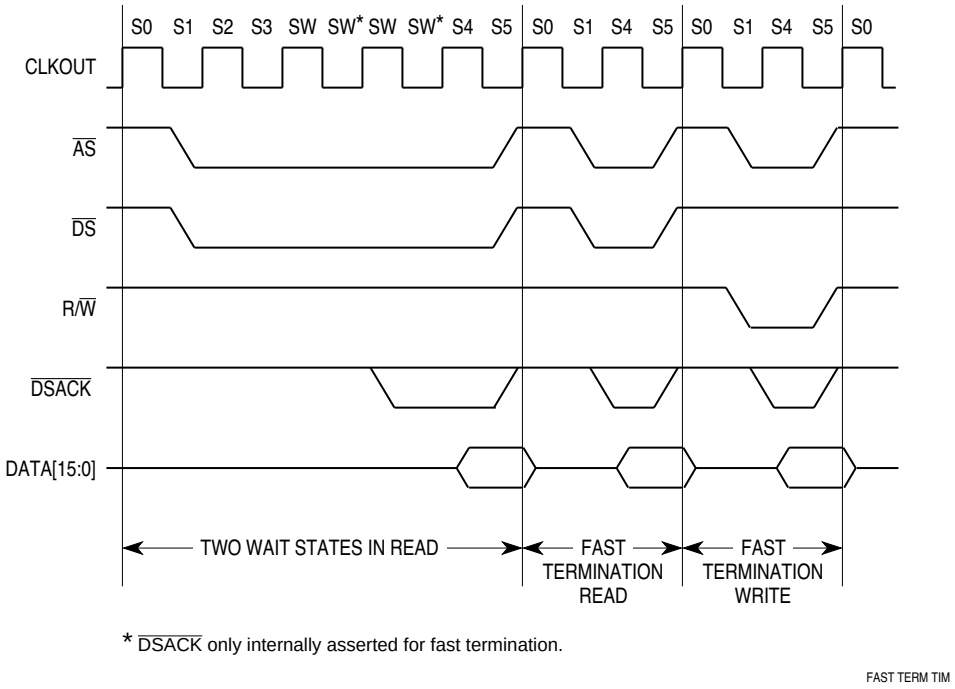


Figure 7-3 Fast-Termination Timing

7.7.1 Fast-Termination Read Cycle

A fast-termination read cycle takes place in much the same way as a regular read cycle, except that the clock states for external handshaking are omitted.

State 0 (S0) — The read cycle starts. The CPU places an address on ADDR[23:0] and function codes on FC[2:0]. The CPU drives R/$\overline{W}$ high for a read cycle. Size signals SIZ[1:0]; become valid, indicating the number of bytes to be read.

State 1 (S1) — The CPU asserts $\overline{AS}$ indicating that the address on the address bus is valid. The CPU also asserts $\overline{DS}$, indicating to external devices that data can be placed on the data bus. SIM chip-select logic decodes the appropriate address lines, FC[1:0], R/$\overline{W}$, and SIZ[1:0]. One or both of DATA[15:8] and DATA[7:0] are selected, and the responding device places data on that portion of the bus.

State 4 (S4) — Appropriate internal $\overline{DSACK}$ signals are generated and the CPU latches data on the falling edge of S4.

State 5 (S5) — The CPU negates $\overline{AS}$ and $\overline{DS}$, but holds the address valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must maintain data until either $\overline{AS}$ or $\overline{DS}$ is negated. It must remove the data within approximately one clock period after sensing the negation of $\overline{AS}$ or $\overline{DS}$. Signals that remain asserted beyond this limit can be prematurely detected during the next bus cycle.

7.7.2 Fast-Termination Write Cycle

A fast-termination write cycle takes place in much the same way as a regular write cycle, except that the clock states for external handshaking are omitted.

State 0 (S0) — The CPU places an address on ADDR[23:0] and function codes on FC[2:0]. The CPU drives R/$\overline{W}$ low for a write cycle. Size signals SIZ[1:0] become valid, indicating the number of bytes to be written.

State 1 (S1) — The CPU asserts $\overline{AS}$, indicating that the address on the address bus is valid. SIM chip-select logic decodes the appropriate address lines, FC[1:0], R/$\overline{W}$, SIZ[1:0], and $\overline{AS}$.

State 4 (S4) — Data driven onto DATA[15:0] becomes valid, and the selected peripheral latches the data. Appropriate internal $\overline{DSACK}$ signals are generated.

State 5 (S5) — The MCU negates $\overline{AS}$ but holds the address and data valid to provide address hold time for memory systems. R/$\overline{W}$, SIZ[1:0], and FC[2:0] also remain valid throughout S5.

7.8 Using Chip Selects in Interrupt Acknowledge Cycles

Chip-select circuits can be programmed to respond during interrupt acknowledge cycles initiated by assertion of an external $\overline{IRQ}$ pin. Any chip-select circuit can be programmed so that the chip-select pin is asserted during an interrupt acknowledge cycle when match conditions are met. Alternately, the chip-select circuit can be programmed to generate autovector ($\overline{AVEC}$) signals internally.

To configure a chip select to respond during an interrupt acknowledge cycle, bits [15:3] of the base register must be set to all ones to match ADDR[23:11], since the address is compared to an address generated by the CPU. (See **Figure 7-4**.) In the chip-select option register, set the SPACE field to %00 for CPU space, and set the R/$\overline{W}$ field to read only.

During an interrupt acknowledge cycle, the interrupt priority on ADDR[3:1] is compared to the value of IPL in the chip-select option register. If the values are the same (and other option register constraints are satisfied), a chip select signal is asserted. Encoding %000 causes a chip-select signal to be asserted regardless of the interrupt level on ADDR[3:1], provided all other constraints are met.

Figure 7-4 shows CPU space encoding for an interrupt acknowledge cycle. FC[2:0] are set to %111, designating CPU space access. ADDR[3:1] indicate interrupt priority, and the space type field (ADDR[19:16]) is set to %1111, the interrupt acknowledge code. The rest of the address lines are set to one.

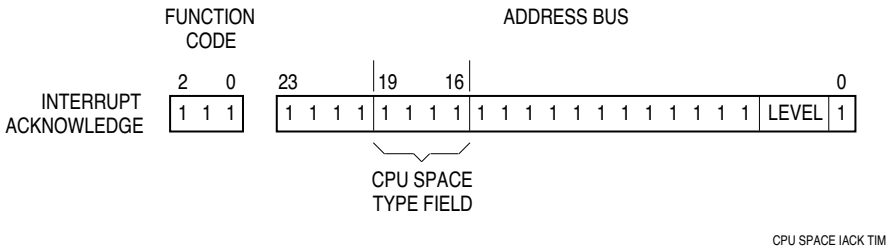


Figure 7-4 CPU Space Encoding for Interrupt Acknowledge Cycles

When the chip-select base and option registers are programmed to respond during an interrupt acknowledge cycle, they must not be programmed to select an external device for reading or writing. Normal data accesses occur in supervisor or user data space, but interrupt acknowledge cycles occur in CPU space. To select the device for reading or writing, a separate chip select is needed from the one programmed to respond during interrupt acknowledge cycles. If a chip-select circuit is used for $\overline{\text{AVEC}}$ support, however, the associated pin can still be used for discrete output or its alternate function.

NOTE

If chip-select base and option registers are programmed to generate an $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signal internally in response to a given interrupt level, and an internal module generates an interrupt request of that level, the internal module will supply an internal $\overline{\text{DSACK}}$ signal to terminate the interrupt acknowledge cycle. The chip-select circuit generates $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals only in response to interrupt requests from external $\overline{\text{IRQ}}$ pins.

7.8.1 Using a Chip-Select Pin as an Interrupt Acknowledge Signal

Follow these steps to use a chip-select pin as an interrupt acknowledge signal.

1. Configure the pin as a chip select to an 8- or 16-bit port in the appropriate chip-select pin assignment register.
2. In the base address register, program the base address field (bits [15:3]) to all ones. Program the block size to no more than 64 Kbytes so that the address comparator checks address lines ADDR[19:16] against the corresponding bits in the base address register. (The CPU places the CPU space type on ADDR[19:16].)
3. Program the chip-select options register as follows:
   - Program MODE to zero to emulate asynchronous bus cycles.
   - Set the R/$\overline{\text{W}}$ field to read only. An interrupt acknowledge cycle is performed as a read cycle.
   - Program the BYTE field to lower byte when using a 16-bit port, since the external vector for a 16-bit port is fetched from the lower byte. Program the BYTE field to upper byte when using an 8-bit port.

- Program STRB to synchronize with $\overline{AS}$.
- Program the $\overline{DSACK}$ field to the desired number of wait states. Select the %1111 option if the external device will generate $\overline{DSACK}$ signals.
- Program IPL to respond to the desired interrupt request level, or to %000 to respond to all request levels.
- Program the $\overline{AVEC}$ bit to 0 to disable autovector generation. (Generating an autovector signal with chip selects is described in the following subsection.)

7.8.2 Generating an Autovector Signal with a Chip-Select Circuit

If the $\overline{AVEC}$ bit in the chip-select option register is set to one, chip-select circuitry generates an internal automatic vector signal in response to an interrupt acknowledge cycle initiated by an $\overline{IRQ}$ pin, provided the match conditions in the base address and option registers are met. The $\overline{AVEC}$ signal causes the CPU to use a predetermined set of vectors to service the interrupt.

If the $\overline{AVEC}$ bit is set to zero, the device requesting interrupt service must either assert the $\overline{AVEC}$ pin or supply the vector and terminate the cycle by asserting $\overline{DSACK}$.

The $\overline{AVEC}$ bit must not be set when the MODE bit is set (chip select assertion synchronized to ECLK), since autovector response timing can vary due to ECLK synchronization.

To generate an autovector signal with a chip-select circuit, follow these steps:

1. For most applications, program the appropriate chip-select pin assignment register to configure the pin for either discrete output or its alternate function. This prevents the pin from being asserted during interrupt acknowledge cycles.
2. In the base address register, program the base address field (bits [15:3]) to all ones. Program the block size to no more than 64 Kbytes so that the address comparator checks address lines ADDR[19:16] against the corresponding bits in the base address register. (The CPU places the CPU space type on ADDR[19:16].)
3. Program the chip-select options register as follows:
   - Program MODE to zero to emulate asynchronous bus cycles.
   - Set the R/$\overline{W}$ field to read/write.
   - Program the BYTE field to both bytes.
   - Program STRB to synchronize with $\overline{AS}$.
   - Program the $\overline{DSACK}$ field to any value. When the $\overline{AVEC}$ bit is set, fast termination is automatically selected.
   - Program IPL to respond to the desired interrupt request level, or to %000 to respond to all request levels.
   - Program the $\overline{AVEC}$ bit to 1 to enable autovector generation.

7.9 Chip-Select Reset Operation

Chip-select pin assignment at reset and the reset values in the base and option registers are discussed in the following paragraphs.

7.9.1 Pin Assignment

Out of reset, chip-select pin functions are determined by the logic levels on pins DATA[7:0]. Data pins have weak internal pull-up drivers, but external devices can hold the pins low. Refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION** for suggestions for holding pins low during reset and to **APPENDIX A ELECTRICAL CHARACTERISTICS** for drive requirements.

The least significant bit of each of the 2-bit CS[10:0] pin assignment fields in CSPAR0 and CSPAR1 has a reset value of one. The reset value of the most significant bit of each field is determined by the states of DATA[7:1] during reset. An encoding of %11 configures the pin as a chip select for a 16-bit port. An encoding of %01 selects the alternate function for the pin.

DATA[2:0] determine the reset setting of CSPAR0, which controls pins $\overline{\text{CS}}$[5:0] and $\overline{\text{CSBOOT}}$. (See **Table 7-6**.)

Table 7-6 Reset Pin Function of $\overline{\text{CS}}$[5:0], $\overline{\text{CSBOOT}}$

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2

DATA[7:3] determine the reset setting of CSPAR1, which controls pins $\overline{\text{CS}}$[10:6]. If an external device pulls one of these data pins low, the associated chip-select pin and lower-numbered pins controlled by CSPAR1 are configured as address pins coming out of reset. For example, if DATA6 is pulled low during reset, pins $\overline{\text{CS}}$[9:6]/ADDR[22:19] are configured as address lines. **Table 7-7** summarizes the reset operation of pins controlled by CSPAR1.

Table 7-7 Reset Pin Function of $\overline{\text{CS}}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{\text{CS10}}$/ ADDR23	$\overline{\text{CS9}}$/ ADDR22	$\overline{\text{CS8}}$/ ADDR21	$\overline{\text{CS7}}$/ ADDR20	$\overline{\text{CS6}}$/ ADDR19
1	1	1	1	1	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	$\overline{\text{CS6}}$
1	1	1	1	0	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	ADDR19
1	1	1	0	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{\text{CS10}}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

7.9.2 $\overline{\text{CS}}$[10:0] Base and Option Registers

Base address and option registers for $\overline{\text{CS}}$[10:0] have reset values of all zeros. This means that the chip selects are disabled out of reset. Assigning a nonzero value in the BYTE field of the option register enables the associated chip select.

7.9.3 $\overline{\text{CSBOOT}}$ Base Address and Option Registers

The $\overline{\text{CSBOOT}}$ assignment field in CSPAR0 is configured differently from the other pin assignment fields. The MSB (bit 1) of the $\overline{\text{CSBOOT}}$ assignment field in CSPAR0, has a reset value of one. This enables the $\overline{\text{CSBOOT}}$ signal to select a boot ROM containing initialization firmware. The LSB value, determined by the logic level of DATA0 during reset, selects boot ROM port size. When DATA0 is held low, port size is 8 bits. When DATA0 is held high (either by the weak internal pull-up driver or by an external pull-up device), port size is 16 bits.

After reset, the MCU fetches initialization vectors beginning at word address $000000. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A ROM device containing a reset vector beginning at its base address can be enabled by $\overline{\text{CSBOOT}}$ after a reset.

Table 7-8 shows the reset values in the base and option registers for $\overline{\text{CSBOOT}}$.

Table 7-8 $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values

Field	Reset Value
Base Address	$000000
Block Size	1 Mbyte*
Timing Mode	Emulates Asynchronous Bus Cycles
Upper/Lower Byte	Both Bytes
Read/Write	Read/Write
Strobe ($\overline{\text{AS}}$/$\overline{\text{DS}}$)	$\overline{\text{AS}}$
$\overline{\text{DSACK}}$	13 Wait States
Address Space	Supervisor/User Space
IPL	Any Level
Autovector	Interrupt Vector Externally

*Default block size is effectively 512 Kbyte on CPU16-based MCUs since values of ADDR[23:20] follow ADDR19.

7.10 Chip-Select Register Diagrams

Chip-select registers include two pin assignment registers, two base address registers, twelve option registers, and a discrete output data register.

7.10.1 Chip-Select Pin Assignment Registers

The pin assignment registers contain 12 two-bit fields ($\overline{\text{CS}}$[10:0] and $\overline{\text{CSBOOT}}$) that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown below. Pins that have no discrete output function do not use the %00 encoding.

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

NOTE

On MCUs with a reduced pin-count SIM, some chip-select pins may not be available. Refer to the user's manual for the particular MCU for details.

Pin assignments at reset are determined by the states of the data bus pins indicated in the following register diagrams. Refer to **7.9 Chip-Select Reset Operation** for additional information on pin assignments at reset.

CSPAR0 — Chip-Select Pin Assignment Register 0 **$####44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CSPA0[6]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		$\overline{\text{CSBOOT}}$	
RESET:															
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains seven two-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect.

Table 7-9 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	$\overline{\text{CS4}}$	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
CSPA0[1]	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—	—

CSPAR1 — Chip-Select Pin Assignment Register 1 **$####46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CSPA1[4]		CSPA1[3]		CSPA1[2]		CSPA1[1]		CSPA1[0]	
RESET:															
0	0	0	0	0	0	DATA7	1	DATA6	1	DATA5	1	DATA4	1	DATA3	1

CSPAR1contains five two-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

Table 7-10 CSPAR1 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA1[4]	$\overline{\text{CS10}}$	ADDR23	ECLK
CSPA1[3]	$\overline{\text{CS9}}$	ADDR22	PC6
CSPA1[2]	$\overline{\text{CS8}}$	ADDR21	PC5
CSPA1[1]	$\overline{\text{CS7}}$	ADDR20	PC4
CSPA1[0]	$\overline{\text{CS6}}$	ADDR19	PC3

7.10.2 Chip-Select Base Address Registers

CSBARBT contains the base address for selection of a bootstrap peripheral memory device. Bit and field definition for CSBARBT and CSBAR[10:0] are the same, but reset block sizes differ. Refer to **7.2 Chip-Select Base Addresses** for more information.

CSBARBT — Chip-Select Base Address Register Boot ROM **$####48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR[10:0] — Chip-Select Base Address Registers **$####4C–$####74**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADDR[15:3] — Base Address Field

This field sets the starting address of a particular address space.

BLKSZ — Block Size Field

This field determines the size of the block above the base address that is enabled by the chip select.

7.10.3 Chip-Select Option Registers

Option register fields determine timing of and conditions for assertion of chip-select signals. CSORBT contains parameters that support bootstrap operations from peripheral memory devices. Bit and field definitions for CSORBT and CSOR[10:0] are the same, but their reset settings differ.

CSORBT — Chip Select Option Register Boot ROM **$####4A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			$\overline{AVEC}$
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

CSOR[10:0] — Chip Select Option Registers **$####4E–$####76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			$\overline{AVEC}$
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MODE — Timing Mode

The MODE bit determines whether chip-select operation emulates asynchronous bus operation or is synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23. Refer to **7.5 Chip-Select Timing** for additional information.

0 = Emulate asynchronous bus operation
1 = Synchronize chip-select assertion to ECLK

BYTE — Upper/Lower Byte Option

This field enables or disables the chip-select circuit and, for 16-bit ports, determines which combinations of size and ADDR0 pins will cause the chip select to be asserted. Refer to **7.6 Chip Selects and Dynamic Bus Sizing** for more information.

00 = Disable
01 = Lower byte
10 = Upper byte
11 = Both Bytes

R/$\overline{W}$ — Read/Write

This field causes a chip select to be asserted only for a read, only for a write, or for both read and write.

00 = Reserved
01 = Read only
10 = Write only
11 = Read/Write

STRB — Address Strobe/Data Strobe

The STRB bit controls the timing of a chip-select assertion in asynchronous mode. This bit has no effect in synchronous mode.

0 = Synchronize chip select assertion with address strobe
1 = Synchronize chip select assertion with data strobe

$\overline{DSACK}$ — Data and Size Acknowledge

This field specifies the source of $\overline{DSACK}$ when chip-select cycles emulate asynchronous bus cycles, and controls wait state insertion. Refer to **7.6 Chip Selects and Dynamic Bus Sizing** for details.

SPACE — Address Space Select

The SPACE field determines the address space in which a chip select is asserted. An access must have the space type represented by SPACE encoding in order for a chip-select signal to be asserted. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

00 = CPU space
01 = User space
10 = Supervisor space
11 = Supervisor or user space

IPL — Interrupt Priority Level

This field selects the interrupt level when a chip select is used for interrupt acknowledge. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

000 = Any Level
001 = Level 1
010 = Level 2
011 = Level 3
100 = Level 4
101 = Level 5
110 = Level 6
111 = Level 7

$\overline{\text{AVEC}}$ — Autovector Enable

This field specifies whether to generate an internal $\overline{\text{AVEC}}$ signal during an interrupt acknowledge cycle initiated by assertion of an $\overline{\text{IRQ}}$ pin when match conditions are met. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

0 = Disable $\overline{\text{AVEC}}$ generation
1 = Enable $\overline{\text{AVEC}}$ generation

7.10.4 Port C Data Register (PORTC)

The port C data register latches data for pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. PC[6:0] correspond to CS[9:3]. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect, and a read returns zero. Refer to **7.3 Pin Assignments and Discrete Output** for more information on this register.

PORTC — Port C Data Register **$####40**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:								1	1	1	1	1	1	1	1

7.11 Interfacing Example with Chip Selects

Figure 7-5 shows a system configuration in which SIM chip-select pins are connected to a boot ROM module and a 16-bit memory consisting of two banks of 8-bit memory. The following paragraphs discuss connecting the pins and programming the base and option registers for the connected chip selects.

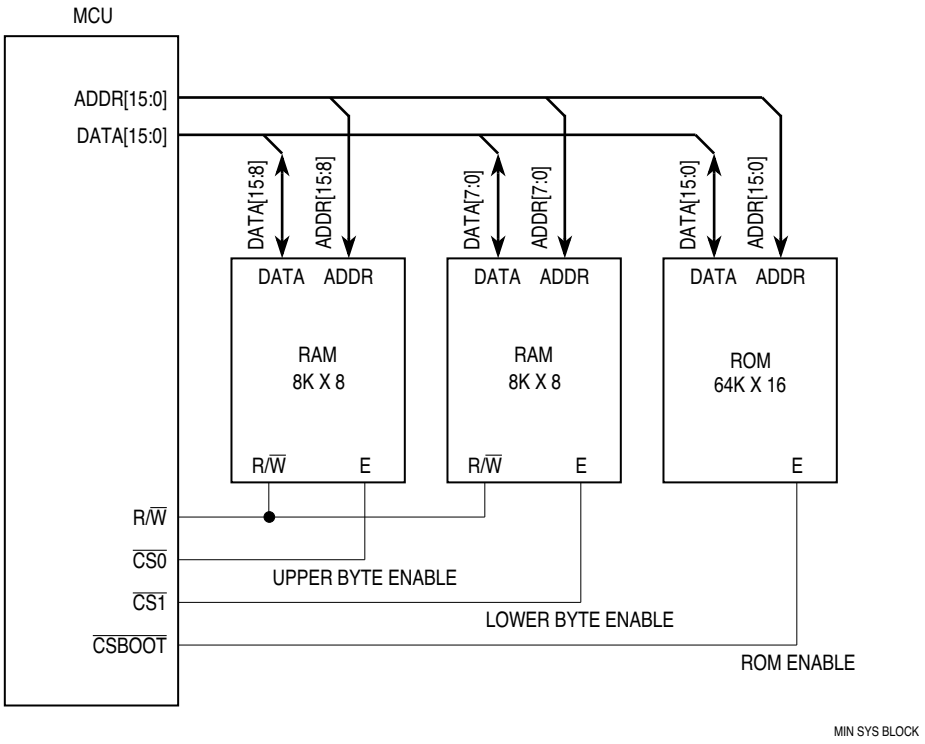


Figure 7-5 System Configuration with Chip Selects

7.11.1 Configuring the RAM Chip Selects

The following paragraphs describe the configuration of the RAM chip selects in the example configuration (**Figure 7-5**).

7.11.1.1 Pin Connections

The upper and lower memory banks are connected to $\overline{CS0}$ and $\overline{CS1}$, respectively. ADDR[13:1] are connected to ADDR[12:0] of each memory bank. ADDR0 of the MCU is not connected to the memory chips; instead, the chip-select logic in each circuit uses the value of ADDR0 on the internal address bus and the value of the BYTE field in the associated chip-select option register to determine whether a match occurs.

No $\overline{DSACK}$ lines are connected, since the chip selects are configured to generate $\overline{DSACK}$ signals internally. No function code lines are connected; in this example, chip-select logic is used to specify supervisor/user space.

7.11.1.2 Base Address Registers

The base address registers are programmed as follows:

CSBAR0 = $1001
CSBAR1 = $1001

This selects a block size of 16 Kbytes starting at address $100000.

7.11.1.3 Option Registers

The option registers (CSOR0 and CSOR1) are programmed as follows:

MODE = %0. This causes chip select operation to emulate asynchronous bus operation. (Bus cycles are terminated with $\overline{\text{DSACK}}$.)

BYTE = %10 in CSOR0 and BYTE = %01 in CSOR1. This assigns the memory bank connected to $\overline{\text{CS0}}$ as the upper byte and the bank connected to $\overline{\text{CS1}}$ as the lower byte.

R/$\overline{\text{W}}$ = %11. This configures the memory for both reads and writes.

STRB = %0. This causes the chip select to be asserted with $\overline{\text{AS}}$.

DSACK = %0000. This causes $\overline{\text{DSACK}}$ signals to be generated internally by the SIM chip-select circuitry, with zero wait states inserted.

SPACE = %11. This selects the memory for both supervisor and user access.

IPL, AVEC = %0000. Since the chip selects are not being used during interrupt acknowledge cycles, the interrupt fields are set to zeros.

7.11.2 Configuring the Boot ROM Chip Select

The following paragraphs describe the configuration of the boot ROM chip select in **Figure 7-5**.

7.11.2.1 Pin Connections

The boot ROM chip is connected to $\overline{\text{CSBOOT}}$. ADDR[16:1] are connected to ADDR[15:0] of the ROM chip. ADDR0 of the MCU is not connected to the ROM in this example.

As with the RAM chips, no $\overline{\text{DSACK}}$ lines are connected, since the chip selects are configured to generate $\overline{\text{DSACK}}$ signals internally. No function code lines are connected; in this example, chip-select logic is used to specify supervisor/user space.

7.11.2.2 Base Address Register

CSBARBT comes out of reset with a base address of $000000 and a block size of 1 Mbyte. The register is reassigned this value:

CSBARBT = $0003

This selects a block size of 128 Kbytes starting at address $000000.

7.11.2.3 Option Registers

The option register (CSORBT) is reprogrammed as follows:

MODE = %0. This causes chip select operation to emulate asynchronous bus operation. (Bus cycles are terminated with $\overline{\text{DSACK}}$.)

BYTE = %11. This assigns the ROM to be a 16-bit port.

R/$\overline{W}$ = %01. This configures the memory as read only.

STRB = %0. This causes the chip select to be asserted with $\overline{AS}$.

DSACK = %0010. This causes $\overline{DSACK}$ signals to be generated internally by the SIM chip-select circuitry, with two wait states inserted.

SPACE = %11. This selects the memory for both supervisor and user access.

IPL, AVEC = %0000. For read/write cycles, the interrupt fields are set to zeros.

SECTION 8 RESET AND SYSTEM INITIALIZATION

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SIM determines whether a reset is valid, synchronizes the reset if necessary to the completion of the current bus cycle, asserts the appropriate internal signals, performs basic system configuration and boot ROM selection based on hardware mode-select inputs, and then passes control to the CPU.

The CPU handles a reset as the highest-priority exception. Each exception has an assigned vector that points to an associated handler routine. During exception processing, the CPU fetches the vector assigned to the exception and executes the exception routine to which the vector points.

Exception vectors are stored in a vector table. Out of reset, the table is located beginning at address $000000. The reset vector occupies the first four words of the vector table. The $\overline{\text{CSBOOT}}$ chip-select signal, which responds to memory accesses starting at $000000 coming out of reset, can be used to select the boot ROM chip with the system initialization routine.

The size of the vector table, the size of each exception vector, and whether the table can be relocated depend on the CPU. Refer to the appropriate CPU reference manual for additional information on exception vectors and exception processing.

8.1 Reset Operation

Sources of reset include external reset, power-on reset, software watchdog, double bus fault, loss of crystal, and system (the CPU32 RESET instruction). The reset status register (RSR) contains a status bit for every reset source in the SIM.

Reset control logic determines the cause of reset, synchronizes reset assertion if necessary to the completion of the current bus cycle, and asserts the appropriate reset lines.

Reset control logic can drive four different internal signals:

- EXTRST (external reset) drives the external reset pin.
- CLKRST (clock reset) resets the clock module.
- MSTRST (master reset) goes to all other internal circuits.
- SYSRST (system reset) indicates to internal circuits that the CPU has executed a RESET instruction.

Figure 8-1 indicates the different sources of reset and the reset lines that the reset control logic asserts for each type of reset request.

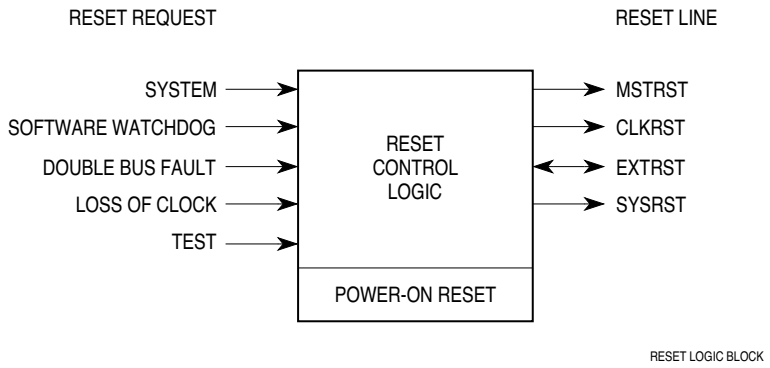


Figure 8-1 Reset Block Diagram

All resets are gated by CLKOUT. Resets are classified as synchronous or asynchronous. An asynchronous reset can occur on any CLKOUT edge. Reset sources that cause an asynchronous reset usually indicate a catastrophic failure; thus the reset control logic responds by asserting reset to the system immediately. (A system reset, however, caused by the CPU32 RESET instruction, is asynchronous but does not indicate any type of catastrophic failure; see **8.2 Sources of Reset** for more information.)

A synchronous reset occurs at the end of a bus cycle. For synchronous resets, only single-byte or aligned-word writes on the IMB are guaranteed to be completed without data corruption. Long-word writes, misaligned operand writes, and read cycles are not guaranteed to be completed. External writes are also guaranteed to be completed, provided the external configuration logic on the data bus is conditioned by R/$\overline{W}$ as shown in **Figure 8-4** later in this section.

The internal bus monitor is automatically enabled whenever the reset control logic must synchronize reset to the end of the bus cycle. When a bus cycle does not terminate normally, the bus monitor terminates it based on the length of time programmed in the BMT field of the system protection control register. Refer to **3.2 Internal Bus Monitor** for additional information.

8.2 Sources of Reset

Sources of reset include external reset requests, the power-on reset circuit, the software watchdog monitor, the double bus fault monitor, the loss-of-crystal circuitry, and the CPU32 RESET instruction (system reset). The reset status register (RSR) contains a status bit for every reset source in the EBI.

NOTE

The RESET instruction is a CPU32 instruction; the CPU16 does not support it.

Table 8-1 is a summary of reset sources.

Table 8-1 Reset Sources

Type	Source	Timing	Cause	Lines Asserted by Reset Controller		
External	External	Synch	External Signal	MSTRST	CLKRST	EXTRST
Power Up	EBI	Asynch	V_{DD}	MSTRST	CLKRST	EXTRST
Software Watchdog	SW Monitor	Asynch	Time Out	MSTRST	CLKRST	EXTRST
Double Bus Fault	DBF Monitor	Asynch	Double Bus Fault	MSTRST	CLKRST	EXTRST
Loss of Clock	Clock	Synch	Loss of Reference	MSTRST	CLKRST	EXTRST
Test	Test	Synch	Test Mode	MSTRST	—	EXTRST
System	CPU32	Asynch	RESET Instruction	—	—	EXTRST

8.2.1 External Reset

When the EXT bit in the RSR is set, the most recent was caused by an external device asserting $\overline{\text{RESET}}$. External resets are synchronous.

NOTE

Since this pin is bidirectional, a conflict exists when the CPU32 executes a RESET instruction and an external device asserts the $\overline{\text{RESET}}$ line. On CPU32-based MCUs, to guarantee that an external reset is recognized by the EBI, $\overline{\text{RESET}}$ must be held for at least 520 cycles so that it overlaps the 512 cycles of the CPU32 RESET instruction.

8.2.2 Power-On Reset

When the POW bit in the RSR is set, the most recent reset state was caused by the power-on reset circuit in the reset controller. A power-on reset is asynchronous. Refer to **8.4 Power-On Reset** for more information.

8.2.3 Software Watchdog Reset

When the SW bit in the RSR is set, the most recent reset was caused by the software watchdog circuit in the system protection module. A software watchdog reset indicates that the CPU is no longer executing the desired code. A software watchdog reset is asynchronous.

8.2.4 Double Bus Fault Reset

When the DBF bit in the RSR is set, the most recent reset was caused by a double bus fault detected by the system protection module. This type of reset is asynchronous.

8.2.5 Loss of Clock Reset

When the LOC bit in the RSR is set, the most recent reset was caused by a loss of clock reference signal. This reset condition can exist only if the RSTEN bit in the synthesizer control register (SYNCR) is set and the VCO is being used. This type of reset is synchronous.

8.2.6 System Reset

When the SYS bit in the RSR is set, the most recent reset was caused by the CPU32 RESET instruction. (The CPU16 does not support this instruction.) This type of reset does not load a reset vector or affect CPU registers or SIM configuration registers, but does assert the $\overline{\text{RESET}}$ line, thus resetting external devices and internal modules other than the CPU. (Not all internal modules respond to system reset. Refer to the reference manuals for the individual modules to determine how they respond to this instruction.) System reset thus allows software to reset the system to a known state and then continue processing with the next instruction.

Since the CPU32 is in control during a RESET instruction, it is not normally necessary to read the RSR to determine the source of reset. The SYS bit is provided, however, for the sake of completeness.

8.2.7 Test Module Reset

When the TST bit in the RSR is set, the most recent reset was caused by the test submodule. This condition occurs during system testing only.

8.2.8 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, more than one bit in RSR may be set. The reset status register is updated by the reset control logic when the $\overline{\text{RESET}}$ signal is released. This register can be read at any time. A write has no effect.

RSR — Reset Status Register **$####06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								EXT	POW	SW	DBF	0	LOC	SYS	TST

EXT — External Reset
Reset was caused by an external signal.

POW — Power-On Reset
Reset was caused by the power-on reset circuit.

SW — Software Watchdog Reset
Reset was caused by the software watchdog circuit.

DBF — Double Bus Fault Reset
Reset was caused by a double bus fault.

LOC — Loss of Clock Reset
Reset was caused by loss of clock reference signal.

SYS — System Reset
Reset was caused by the CPU32 RESET instruction. The CPU16 does not support this instruction.

TST — Test Submodule Reset
Reset was caused by the test submodule. This bit is set during system test only.

8.3 Reset Control Flow

The following paragraphs describe reset control flow after the SIM receives a reset request. Refer to **8.4 Power-On Reset** for additional details of reset timing during power-on reset. **Figure 8-2** is a reset control flow diagram.

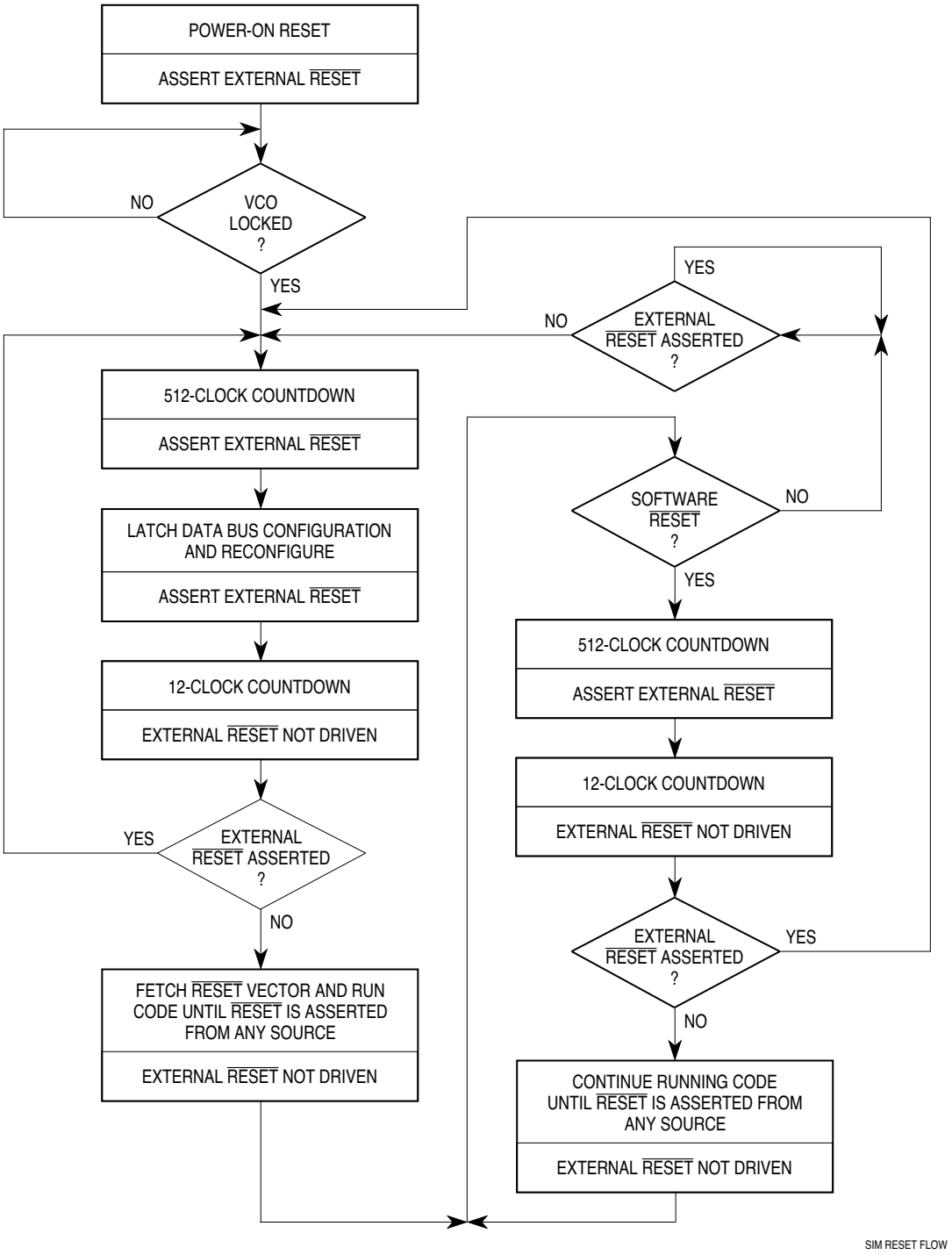


Figure 8-2 Reset Control Flow

8.3.1 $\overline{\text{RESET}}$ Assertion by an External Device

When an external device requests reset by asserting $\overline{\text{RESET}}$ for at least four CLKOUT cycles, reset control logic clocks the signal into an internal latch. The control logic drives the $\overline{\text{RESET}}$ pin low for an additional 512 CLKOUT cycles after it detects that the $\overline{\text{RESET}}$ signal is no longer being externally driven, to guarantee this length of reset to the entire system.

After 512 cycles have elapsed, allowing the weak pull-up devices on the data bus configuration pins to pull the pins up to logic level one, the $\overline{\text{RESET}}$ pin goes to a disabled (high-impedance) state for 10 cycles. At the end of this 10-cycle period, the data bus pin configuration is latched, and the state of the $\overline{\text{RESET}}$ pin is tested. If the pin state is logic level one (negated), reset exception processing begins. If, however, the pin state is logic level zero (asserted), the reset control logic drives the pin low for another 512 cycles, on the assumption that an external device is driving the pin low. At the end of this period, the pin again goes to a high-impedance state for 10 cycles, and then is tested again. The process repeats until $\overline{\text{RESET}}$ goes high.

Refer to parameters 77 and 78 in **Table A-3** in **APPENDIX A ELECTRICAL CHARACTERISTICS** for additional timing details regarding $\overline{\text{RESET}}$ assertion and negation.

8.3.2 Internal Reset Request

When reset is requested by any source other than an external device driving the $\overline{\text{RESET}}$ pin low, the reset control logic asserts $\overline{\text{RESET}}$ for a minimum of 512 cycles, allowing the weak pull-up devices on the data bus configuration pins to pull the pins up to logic level one. If the reset signal is still asserted at the end of 512 cycles, the control logic continues to assert $\overline{\text{RESET}}$ until the internal reset signal is negated.

8.4 Power-On Reset

When the SIM clock synthesizer is used to generate the system clock, power-on reset involves special circumstances related to the application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin V_{DDSYN} in order for the MCU to operate. The following discussion assumes that V_{DDSYN} is applied before and during reset, minimizing crystal start-up time. When this is not the case, start-up time includes crystal cold start-up time in addition to the delays listed in the following paragraphs. Crystal start-up time without V_{DDSYN} applied is affected by specific crystal parameters and by oscillator circuit design.

8.4.1 SIM Operation During Power-On Reset

During power-on reset, an internal circuit in the SIM drives the IMB internal (MSTRST) and external (EXTRST) reset lines. The power-on reset circuit releases the internal reset line as V_{DD} ramps up to the minimum operating voltage (refer to **Table A-4** in **APPENDIX A ELECTRICAL CHARACTERISTICS**), and SIM pins are initialized to the values shown in **Table 8-3**. When V_{DD} reaches the minimum operating voltage, the clock synthesizer VCO begins operation. Clock frequency ramps up to the specified limp mode frequency (F_{LIMP}). The external $\overline{\text{RESET}}$ signal remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

Figure 8-3 is a timing diagram of power-on reset. It shows the relationships between $\overline{\text{RESET}}$, V_{DD}, and bus signals.

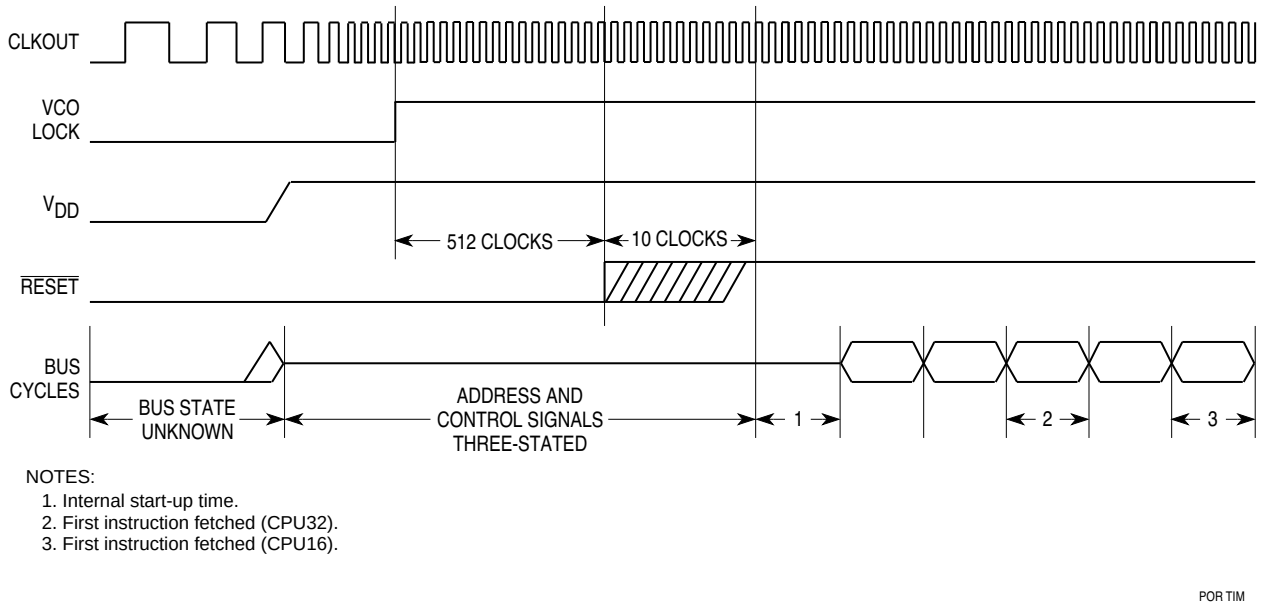


Figure 8-3 Power-On Reset Timing

8.4.2 Other Modules During Power-On Reset

The clock synthesizer in the SIM provides clock signals to other MCU modules. After the clock is running and the internal reset signal (MSTRST) is asserted for at least four clock cycles, these modules are reset. V_{DD} ramp time and VCO frequency ramp time determine how long these four cycles take. Worst case is approximately 15 milliseconds.

During this period, input/output and output-only port pins on modules other than the SIM may be in an indeterminate state. Input/output pins on these modules may be in output mode for a short time, which may create a conflict with external input drive logic. If a known state on input/output or output-only pins is required before this 15-ms period, external reset control logic must condition these lines. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

Input-only pins can be placed in a known state by means of external pull-up resistors.

8.5 Use of the Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in a disabled, high-impedance state. The signal must remain asserted for approximately 10 clock cycles in order for drivers to change state.

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the 10 cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as approximately 10 clock pulses have been applied to the EXTAL pin.

NOTE

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

8.6 Operating Configuration out of Reset

The logic states of certain data bus pins during reset determine SIM operating configuration. In addition, the state of the MODCLK pin determines the system clock source, and the state of the $\overline{\text{BKPT}}$ pin determines what happens during subsequent breakpoint assertions. **Table 8-2** is a summary of reset mode selection options. Subsequent paragraphs explain these options in detail.

Table 8-2 Reset Mode Selection

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA1	$\overline{\text{CS0}}$ $\overline{\text{CS1}}$ $\overline{\text{CS2}}$	$\overline{\text{BR}}$ $\overline{\text{BG}}$ $\overline{\text{BGACK}}$
DATA2	$\overline{\text{CS3}}$ $\overline{\text{CS4}}$ $\overline{\text{CS5}}$	FC0 FC1 FC2
DATA3 DATA4 DATA5 DATA6 DATA7	$\overline{\text{CS6}}$ $\overline{\text{CS}}$[7:6] $\overline{\text{CS}}$[8:6] $\overline{\text{CS}}$[9:6] $\overline{\text{CS}}$[10:6]	ADDR19 ADDR[20:19] ADDR[21:19] ADDR[22:19] ADDR[23:19]
DATA8	$\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, $\overline{\text{SIZE}}$	PORTE
DATA9	$\overline{\text{IRQ}}$[7:1] MODCLK	PORTF
DATA11	Test Mode Disabled	Test Mode Enabled
MODCLK	VCO = System Clock	EXTAL = System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled

8.6.1 Data Bus Mode Selection

All data lines have weak internal pull-up drivers during reset. When pins are held high by the internal drivers, the MCU uses a default operating configuration. However, specific lines can be held low externally to achieve an alternate configuration.

NOTE

External bus loading can overcome the weak internal pull-up drivers on data bus lines and hold pins low during reset.

DATA0 determines the function of the boot ROM chip-select signal $\overline{\text{CSBOOT}}$. Unlike other chip-select signals, $\overline{\text{CSBOOT}}$ is active at the release of reset. During reset exception processing, the MCU fetches initialization vectors beginning at address $000000 in supervisor program space. After a reset, an external memory device containing the reset vector can be enabled by $\overline{\text{CSBOOT}}$, which is assigned at reset to a memory block beginning at $000000. The logic level of DATA0 during reset selects boot ROM port size. When DATA0 is held low, port size is 8 bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to **7.9 Chip-Select Reset Operation** for more information.

DATA1 and DATA2 determine the functions of $\overline{\text{CS}}$[2:0] and $\overline{\text{CS}}$[5:3], respectively. DATA[7:3] determine the functions of an associated chip select and all lower-numbered chip-selects down through $\overline{\text{CS6}}$. For example, if DATA5 is pulled low during reset, CS[8:6] are assigned alternate function as ADDR[21:19], and CS[10:9] remain chip selects. (On some MCUs, ADDR[23:20] follow the state of ADDR19, and DATA[7:4] have limited use.) Refer to **7.9 Chip-Select Reset Operation** for more information.

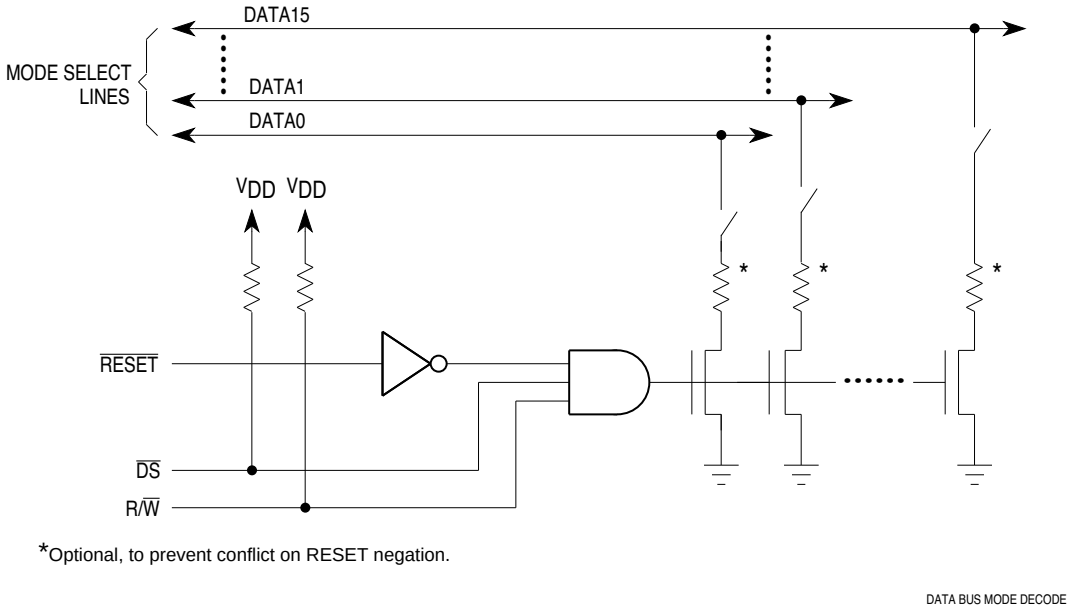
DATA8 determines the function of the $\overline{\text{DSACK0}}$, $\overline{\text{DSACK1}}$, $\overline{\text{AVEC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, and SIZ[1:0] pins. If DATA8 is held low during reset, these pins are used for discrete I/O (port E).

DATA9 determines the function of interrupt request pins ($\overline{\text{IRQ}}$[7:0]) and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are used for discrete I/O (port F).

DATA11 determines whether the SIM operates in test mode out of reset. This capability is used for factory testing of the MCU.

8.6.2 Holding Data Bus Pins Low at Reset

To avoid conflicts on the data bus during reset, use an active device to hold data bus lines low. The data bus configuration logic must be released prior to the first bus cycle after reset in order to prevent conflict with external memory devices. The first bus cycle occurs 14 CLKOUT cycles after $\overline{\text{RESET}}$ is released. If external mode selection logic causes a conflict with external memory devices, an isolation resistor on the driven lines may be required. **Figure 8-4** shows a recommended method for conditioning data bus pins that are held low at reset.



*Optional, to prevent conflict on RESET negation.

Figure 8-4 Data Bus Signal Conditioning

If conflicts on the data bus during reset are not a concern, a diode in series with a 1-kΩ resistor can be used in place of an active device to hold a data bus pin low.

8.6.3 Clock Mode Selection

The state of the clock mode (MODCLK) pin during reset determines which clock source the MCU uses. When MODCLK is held high during reset, the clock signal is generated from a reference frequency. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. Refer to **SECTION 4 SYSTEM CLOCK** for more information.

NOTE

If the MODCLK pin is also used as a parallel port pin, make certain that bus loading does not overcome the weak internal pull-up driver during reset and cause inadvertent clock mode selection.

8.6.4 Breakpoint Mode Selection

The MCU uses internal and external breakpoint ($\overline{BKPT}$) signals. During reset exception processing, at the release of the $\overline{RESET}$ signal, the CPU samples these signals to determine how to handle breakpoints.

If either $\overline{BKPT}$ signal is at logic level zero when sampled, an internal BDM flag is set, and the CPU enters background debugging mode whenever either $\overline{BKPT}$ input is subsequently asserted.

If both $\overline{BKPT}$ inputs are at logic level one when sampled, breakpoint exception processing begins whenever either $\overline{BKPT}$ signal is subsequently asserted.

Refer to the appropriate CPU manual for more information on background debugging mode and exceptions. Refer to **5.8 CPU Space Cycles** for information concerning breakpoint acknowledge bus cycles.

8.7 Pin State During Reset

While the MCU is held in reset, the data bus pins are configured as inputs. Function code pins are driven high. Bus control pins $\overline{AS}$, $\overline{DS}$, SIZ[1:0], and $\overline{RMC}$ are driven high if configured for their bus control function (see **Table 8-3**). Any bus control pins configured for I/O (ports E and F) are configured as inputs when the SIM comes out of reset.

NOTE

Pins that are not used should either be configured as outputs or (if configured as inputs) pulled to the appropriate inactive state. This decreases additional I_{DD} caused by digital inputs floating near mid-supply level.

After $\overline{RESET}$ is released, mode selection occurs, and reset exception processing begins. Pins configured as inputs during reset become active high-impedance loads after $\overline{RESET}$ is released. Inputs must be driven to the desired active state; pull-up or pull-down circuitry may be necessary. Pins configured as outputs begin to function after $\overline{RESET}$ is released.

Table 8-3 is a summary of SIM pin states during reset.

Table 8-3 SIM Pin Reset States

Mnemonic	Pin State While RESET Asserted	Pin State After RESET Released			
		Default Function		Alternate Function	
		Pin State	Pin Function	Pin State	Pin Function
CS10/ADDR23	1	CS10	1	ADDR23	Unknown
CS[9:6]/ADDR[22:19]/ PC[6:3]	1	CS[9:6]	1	ADDR[22:19]	Unknown
ADDR[18:0]	High-Z Output	ADDR[18:0]	Unknown	ADDR[18:0]	Unknown
AS/PE5	See Note 1	AS	Output	PE5	Input
AVEC/PE2	Inactive Input	AVEC	Input	PE2	Input
BERR	Inactive Input	BERR	Input	BERR	Input
CS1/BG	1	CS1	1	BG	1
CS2/BGACK	1	CS2	1	BGACK	Input
CS0/BR	1	CS0	1	BR	Input
CLKOUT	Output	CLKOUT	Output	CLKOUT	Output
CSBOOT	1	CSBOOT	0	CSBOOT	0
DATA[15:0]	Mode Select	DATA[15:0]	Input	DATA[15:0]	Input
DS/PE4	See Note 1	DS	Output	PE4	Input
DSACK0/PE0	Inactive Input	DSACK0	Input	PE0	Input
DSACK1/PE1	Inactive Input	DSACK1	Input	PE1	Input
CS[5:3]/FC[2:0]/ PC[2:0]	1	CS[5:3]	1	FC[2:0]	Unknown
HALT	Inactive Input	HALT	Input	HALT	Input
IRQ[7:1]/PF[7:1]	Inactive Input	IRQ[7:1]	Input	PF[7:1]	Input
MODCLK/PF0	Mode Select	MODCLK	Input	PF0	Input
R/W	High-Z Output	R/W	Output	R/W	Output
RESET	Asserted	RESET	Input	RESET	Input
RMC/PE3	See Note 1	RMC	Output	PE3	Input
SIZ[1:0]/PE[7:6]	See Note 1	SIZ[1:0]	Unknown	PE[7:6]	Input
TSTME/TSC	Mode Select	TSC	Input	TSC	Input

NOTES:

1. The state of these bus control/port E pins during reset depends on the state of DATA8. If DATA8 is high during reset (the default), these pins are driven high during reset and come out of reset as bus control outputs. If DATA8 is held low during reset, these pins are inactive, high-impedance inputs during reset and come out of reset as port E inputs.

8.8 SIM Registers Out of Reset

Table 8-4 summarizes the reset values of bits and fields in the SIM registers. Bits not included in the table are unimplemented and have reset values of zero.

Table 8-4 SIM Registers Out of Reset

Register	Bits	Name	Value	Meaning
SIMCR	15	EXOFF	0	CLKOUT enabled
	14	FRZSW	1	Disable watchdog and PIT when FREEZE asserted
	13	FRZBM	1	Disable bus monitor when FREEZE asserted
	11	SLVEN	$\overline{\text{DATA11}}$	Slave mode enabled if DATA11 low
	9:8	SHEN	00	Show cycles disabled
	7	SUPV	1	Supervisor access only
	6	MM	1	Module registers begin at $FFF000
	3:0	IARB	1111	SIM interrupts have highest priority
SYPCR	7	SWE	1	Software watchdog enabled
	6	SWP	$\overline{\text{MODCLK}}$	Software watchdog prescaled by 512 if MODCLK low
	5:4	SWT	00	Software watchdog time-out set to minimum
	3	HME	0	Disable halt monitor
	2	BME	0	Disable internal-to-external bus monitor
	1:0	BMT	00	Bus monitor time-out = 64 system clocks
SYNCR	15:8	W,X,Y	00111111	System clock frequency = Source frequency $\sum$ 256
	7	EDIV	0	ECLK = System clock $\geq$ 8
	4	SLIMP	Unchanged	Limp mode flag unaffected by reset
	3	SLOCK	Unchanged	Synthesizer lock flag unaffected by reset
	2	RSTEN	0	Loss of crystal causes limp mode (not reset)
	1	STSIM	0	If LPSTOP, disable system clock
	0	STEXT	0	If LPSTOP, disable external clock
PICR	10:8	PIRQL	000	Periodic interrupt disabled
	7:0	PIV	00001111	Vector number = $F (uninitialized interrupt vector)
PITR	8	PTP	$\overline{\text{MODCLK}}$	If MODCLK is low, PTP = 1 (PIT prescaled by 512) If MODCLK is high, PTP = 0 (PIT not prescaled)
	7:0	PITM	00000000	Periodic timer disabled
PORTE	7:0	PE	Unchanged	Port E data register is unaffected by reset
DDRE	7:0	DDE	00000000	Port E pins are configured for input
PEPAR	7:0	PEPA	$\overline{\text{DATA8}}$	If DATA8 = 0, pins configured for G/P I/O If DATA8 = 1, pins configured for bus control
PORTF	7:0	PF	Unchanged	Port F data register is unaffected by reset
DDRF	7:0	DDF	00000000	Port F pins are configured for input
PFPAR	7:0	PFPA	$\overline{\text{DATA9}}$	If DATA9 = 0, pins configured for G/P I/O If DATA9 = 1, pins configured for interrupt request (MODCLK for PF0)
PORTC	6:0	PC	1111111	Port C data register bits [6:0] set to 1
CSPAR1	9:0			**See Table 7-7**
CSPAR0	13:0			**See Table 7-6**
CSBARBT	15:3	ADDR[23:11]	$0	Base address = 0
	2:0	BLKSZ	111	Block size = 1 Mbyte
CSBAR[10:0]	15:3	ADDR[23:11]	$0	Base address = 0
	2:0	BLKSZ	000	Block size = 2 Kbytes

Table 8-4 SIM Registers Out of Reset (Continued)

Register	Bits	Name	Value	Meaning
CSORBT	15	MODE	0	Asynchronous mode
	14:13	BYTE	11	Both bytes
	12:11	R/$\overline{W}$	11	Assert chip select for both reads and writes
	10	STRB	0	Chip select synchronized with $\overline{AS}$
	9:6	$\overline{DSACK}$	1101	13 wait states
	5:4	SPACE	11	Supervisor/user space
	3:1	IPL	000	Assert chip select on any level interrupt
	0	$\overline{AVEC}$	0	Autovector disabled
CSOR[10:0]				Disabled (BYTE field = 0)

8.9 System Initialization

During system initialization, a hardware initialization sequence occurs. Following hardware initialization, the MCU performs the initialization routine fetched from the exception vector table. This initialization routine normally defines constants, addresses, and other data values needed by the program.

The following steps summarize procedures for initializing the SIM. Initialize the SIM before initializing other MCU modules.

1. Program the SIMCR to set the SIM arbitration level, base address of the module registers, and privilege level of assignable SIM registers, and to enable or disable the external clock, the software watchdog and bus monitor, and show cycles.
2. Program the SYNCR to set the system clock frequency, ECLK divide rate, to enable or disable loss-of-crystal reset, and to assign STOP mode clock operation.
3. Program the SYPCR to enable or disable the software watchdog, halt monitor, and bus monitor, and to assign the timing for the software watchdog and bus monitor.
4. Program the PICR and PITR to set the periodic interrupt request level and establish the timing for the periodic timer.
5. Program the chip-select pin assignment, base address, and options registers to assign pin function, initialize chip selects, and assign base addresses and options.
6. Initialize the port E and port F data registers. Then program the port E and port F data direction and pin assignment registers, if necessary, to change the reset values. (Initializing the data registers first guarantees that the desired logic level is output on the pins that are configured as outputs.)

SECTION 9GENERAL-PURPOSE I/O

Sixteen of the SIM pins can be configured for general-purpose discrete input and output. Although these pins are organized into two ports, port E and port F, function assignment is by individual pin. Pin assignment registers, data direction registers, and data registers are used to implement discrete I/O.

In addition to the sixteen pins in ports E and F, the seven pins in port C can be configured as discrete outputs. Port C is discussed in **SECTION 7 CHIP SELECTS**. The following paragraphs describe ports E and F.

NOTE

On MCUs with a reduced pin-count SIM, some of the pins that comprise ports E and F may not be available. Refer to the user's manual for the particular MCU for a list of the pins on the chip.

9.1 Pin Assignment Registers

Bits in the port E and port F pin assignment registers (PEPAR and PFPAR) control the functions of the pins in each port. Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one configures the corresponding pin for its alternate function.

NOTE

On CPU16-based MCUs, the $\overline{\text{RMC}}$ signal is not implemented, and the $\overline{\text{RMC}}$/PE3 pin may be left unconnected. When the pin is left unconnected, PEPA3 always returns one when read, and writes have no effect.

The states of the DATA8 and DATA9 pins control the reset state of PEPAR and PFPAR, respectively. When DATA8 is high during reset, PEPAR is set to $FF, defining all port E pins to be bus control signals. When DATA8 is low during reset, PEPAR is set to $00, defining all port E pins to be I/O pins. In a similar fashion, DATA9 determines the reset state of PFPAR.

PEPAR — Port E Pin Assignment Register **$####16**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:								DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be a bus control signal.

Table 9-1 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$*
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

*On CPU16-based MCUs, when PEPA3 is set, the PE3 pin, if connected, goes to logic level one. The CPU16 does not support the $\overline{RMC}$ function for this pin.

PFPAR — Port F Pin Assignment Register **$####1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:								DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK.

Table 9-2 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{IRQ7}$
PFPA6	PF6	$\overline{IRQ6}$
PFPA5	PF5	$\overline{IRQ5}$
PFPA4	PF4	$\overline{IRQ4}$
PFPA3	PF3	$\overline{IRQ3}$
PFPA2	PF2	$\overline{IRQ2}$
PFPA1	PF1	$\overline{IRQ1}$
PFPA0	PF0	MODCLK

9.2 Data Direction Registers

Bits in the port E and port F data direction registers (DDRE and DDRF) control the direction of the pin drivers when the pins are configured for I/O. Any bit in a register set to one configures the corresponding pin as an output. Any bit in a register cleared to zero configures the corresponding pin as an input. These registers can be read or written at any time. In MCUs in which the $\overline{RMC}$ pin is not implemented, DDE3 always returns one when read, and writes have no effect.

DDRE — Port E Data Direction Register **$####14**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:								0	0	0	0	0	0	0	0

DDRF — Port F Data Direction Register **$####1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:								0	0	0	0	0	0	0	0

9.3 Data Registers

A write to the port E and port F data registers (PORTE and PORTF) is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of a data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Both data registers can be accessed in two locations. They can be read or written at any time.

PORTE and PORTF are unaffected by reset.

PORTE0, PORTE1 — Port E Data Register **$####10, $####12**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:								U	U	U	U	U	U	U	U

NOTE

On CPU16-based MCUs, the $\overline{\text{RMC}}$ signal is not implemented, and PE3 may not be connected to a pin. On CPU16-based MCUs, the value read from PE3 depends on the particular MCU. Refer to the appropriate MCU user's manual. Writes to PE3 have no effect when the $\overline{\text{RMC}}$/PE3 pin is not available.

PORTF0, PORTF1 — Port F Data Register **$####18, $####1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:								U	U	U	U	U	U	U	U

SECTION 10 REDUCED PIN-COUNT SIM

Some Motorola MCUs contain a version of the SIM with a reduced pin count. MCUs with a reduced pin-count SIM (RPSIM) always contain the SIM pins that are most basic to MCU operation; some SIM pins, however, are not included on these chips. The set of pins that are not connected depends on the MCU, but it is always a subset of the pins listed in this section. This section lists the pins that may be omitted from RPSIM-based MCUs and discusses how the omission of these pins affects SIM operation.

10.1 Optional RPSIM Pins

Table 10-1 lists the pins that are optionally not included on MCUs with a reduced pin-count SIM. RPSIM-based MCUs may still have some of these pins available, depending on the requirements of the particular microcontroller. Refer to the user's manual for the specific MCU for a list of the pins on a given chip.

Table 10-1 Optional RPSIM Pins

Pin Mnemonic	Description	Port Designation
ADDR[22:20]/$\overline{\text{CS}}$[9:7]	Address Bus/Chip Selects	PC[6:4]
$\overline{\text{AVEC}}$	Autovector	PE2
$\overline{\text{CSBOOT}}$	Boot Chip Select	—
$\overline{\text{DSACK0}}$	Data and Size Acknowledge	PE0
$\overline{\text{HALT}}$	Halt	—
$\overline{\text{IRQ}}$[5:1]	Interrupt Request Level	PF[5:1]
$\overline{\text{RMC}}$	Read-Modify-Write Cycle	PE3

10.2 Address Bus/Chip Select Pins

Up to three address bus/chip select pins (ADDR[22:20]/$\overline{\text{CS}}$[9:7]/PC[6:4]) are optionally not included on MCUs with a reduced pin-count SIM. On MCUs that lack these pins, ADDR[22:20] follow the logic level of ADDR19, and the chip-select and discrete output functions for these pins are unavailable.

10.3 Data Size and Acknowledge Pins

On MCUs with both $\overline{\text{DSACK}}$[1:0] pins, an external device asserts the appropriate $\overline{\text{DSACK}}$ signal to indicate port size and the availability of data. On some MCUs with a reduced pin-count SIM, no $\overline{\text{DSACK0}}$ pin is provided. With these MCUs, an external device indicates the availability of data by asserting $\overline{\text{DSACK1}}$, regardless of port size. All accesses thus appear to be word accesses.

When connecting an 8-bit device with an 8-bit port to an MCU with no $\overline{\text{DSACK0}}$ pin, connect the peripheral to DATA[15:8] as usual. The peripheral asserts $\overline{\text{DSACK1}}$, rather than $\overline{\text{DSACK0}}$, to indicate completion of the cycle, and the read or write operation terminates after the first bus cycle.

Connecting an 8-bit device with a 16-bit port (e.g., a 16-bit timer) to an MCU with no $\overline{\text{DSACK0}}$ pin requires one of the following work-arounds.

For CPU16-based products, connect the peripheral to DATA[15:8]. Use long word load and store instructions (LDED and STED) and arrange the upper bytes of accumulators E and D into a word.

For CPU32-based products, use the MOVEP (move peripheral data) instruction to move data to and from even addresses. Refer to the CPU32 reference manual for details.

10.4 $\overline{\text{RMC}}$ Pin

The CPU32 asserts the $\overline{\text{RMC}}$ signal to indicate that the current bus cycle is part of an indivisible read-modify-write instruction. The CPU16 does not support this signal. On CPU16-based MCUs, the pin may or may not be available for discrete I/O (PE3). If the pin is not available, a read of PE3 will always return either zero or one, depending on the particular MCU. Refer to the user's manual for the particular CPU16-based MCU to determine how PE3 is implemented.

APPENDIX AELECTRICAL CHARACTERISTICS

Table A-1 Clock Control Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc
32.768 kHz or 4.194 MHz reference)

Characteristic	Symbol	Min	Max	Unit
PLL Reference Frequency Range	f_{ref}	25	50	kHz
System Frequency[1] On-Chip PLL System Frequency External Clock Operation	f_{sys}	dc 0.131 dc	16.78 16.78 16.78	MHz
PLL Lock Time[2]	t_{lpll}	—	20	ms
Limp Mode Clock Frequency[3] SYNCR X bit = 0 SYNCR X bit = 1	f_{limp}	 — —	 f_{sys} max/2 f_{sys} max	MHz
CLKOUT Stability[4, 5] Short term Long term	C_{stab}	 –1.0 –0.5	 1.0 0.5	%

NOTES:

1. All internal registers retain data at 0 Hz.
2. Assumes that stable V_{DDSYN} is applied, that an external filter capacitor with a value of 0.1 µF is attached to the XFC pin, and that the crystal oscillator is stable. Lock time is measured from power-up to $\overline{\text{RESET}}$ release. This specification also applies to the period required for PLL lock after changing the W and Y frequency control bits in the synthesizer control register (SYNCR) while the PLL is running, and to the period required for the clock to lock after LPSTOP.
3. Determined by the internal reference voltage applied to the on-chip VCO. The X bit in SYNCR controls a divide by two prescaler on the system clock output.
4. Short-term CLKOUT stability is the average deviation from programmed frequency measured over a 2 µs interval at maximum f_{sys}. Long-term CLKOUT stability is the average deviation from programmed frequency measured over a 1 ms interval at maximum f_{sys}. Stability is measured with a stable external clock input applied — variation in crystal oscillator frequency is additive to this figure.
5. This parameter is periodically sampled rather than 100% tested.

Table A-2 DC Characteristics

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	0.7 (V_{DD})	V_{DD} + 0.3	V
Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.2 (V_{DD})	V
Input Hysteresis[1]	V_{HYS}	0.5	—	V
Input Leakage Current[2] V_{in} = V_{DD} or V_{SS} Input-only pins	I_{in}	–2.5	2.5	µA
High Impedance (Off-State) Leakage Current[2] V_{in} = V_{DD} or V_{SS} All input/output and output pins	I_{OZ}	 –2.5	 2.5	µA
CMOS Output High Voltage[2, 3] I_{OH} = –10.0 µA Group 1, 2 input/output and all output pins	V_{OH}	V_{DD} – 0.2	—	V
CMOS Output Low Voltage[2] I_{OL} = 10.0 µA Group 1, 2 input/output and all output pins	V_{OL}	—	0.2	V
Output High Voltage[2, 3] I_{OH} = –0.8 mA Group 1, 2 input/output and all output pins	V_{OH}	V_{DD} – 0.8	—	V
Output Low Voltage[2] I_{OL} = 1.6 mA Group 1 I/O Pins, CLKOUT, FREEZE/QUOT I_{OL} = 5.3 mA Group 2 I/O Pins, $\overline{CSBOOT}$, $\overline{BG}$/$\overline{CS}$ I_{OL} = 12 mA Group 3	V_{OL}	 — — —	 0.4 0.4 0.4	V
Three State Control Input High Voltage	V_{IHTSC}	1.6 (V_{DD})	9.1	V
Data Bus Mode Select Pull-up Current[5] V_{in} = V_{IL} DATA[15:0] V_{in} = V_{IH} DATA[15:0]	I_{MSP}	 — –15	 –120 —	µA
V_{DD} Supply Current[6] RUN [4] LPSTOP, 32.768 kHz or 4.194 MHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	 I_{DD} S_{IDD} S_{IDD}	 — — —	 124 350 5	 mA µA mA
Clock Synthesizer Operating Voltage	V_{DDSYN}	4.5	5.5	V
V_{DDSYN} Supply Current[6] 32.768 kHz or 4.194 MHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 32.768 kHz or 4.194 MHz crystal, VCO off (STSIM = 0) 32.768 kHz or 4.194 MHz crystal, V_{DD} powered down	 I_{DDSYN} I_{DDSYN} S_{IDDSYN} I_{DDSYN}	 — — — —	 1 5 150 100	 mA mA µA µA
Power Dissipation[7]	P_D	—	690	mW
Input Capacitance[2, 8] All input-only pins All input/output pins	C_{in}	 — —	 10 20	pF
Load Capacitance[2] Group 1 I/O Pins and CLKOUT, FREEZE/QUOT Group 2 I/O Pins and $\overline{CSBOOT}$, $\overline{BG}$/$\overline{CS}$ Group 3 I/O pins	C_L	 — — —	 90 100 130	pF

NOTES:

1. Applies to:
   Port D [7:0]
   Port E [7:3]
   Port F [7:0]
   $\overline{\text{TSTME}}$/TSC, $\overline{\text{BKPT}}$, $\overline{\text{RESET}}$
2. Input-Only Pins:$\overline{\text{TSTME}}$/TSC, $\overline{\text{BKPT}}$
   Output-Only Pins: $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}$/$\overline{\text{CS}}$, CLKOUT, FREEZE/QUOT
   Input/Output Pins:
   Group 1: DATA[15:0]
   Group 2: Port C (ADDR23/ECLK, ADDR[22:19]/$\overline{\text{CS}}$[9:6], FC[2:0]/$\overline{\text{CS}}$[5:3])
   Port E ($\overline{\text{DSACK}}$[1:0], $\overline{\text{AVEC}}$, $\overline{\text{RMC}}$, $\overline{\text{DS}}$, $\overline{\text{AS}}$, SIZ[1:0])
   Port F ($\overline{\text{IRQ}}$[7:1], MODCLK)
   ADDR[18:0], R/$\overline{\text{W}}$, $\overline{\text{BERR}}$, $\overline{\text{BR}}$/$\overline{\text{CS0}}$, $\overline{\text{BGACK}}$/$\overline{\text{CS2}}$
   Group 3: $\overline{\text{HALT}}$, $\overline{\text{RESET}}$
3. Does not apply to $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ because they are open drain pins.
4. Current measured at system clock frequency of 16.78 MHz.
5. Use of an active pulldown device is recommended.
6. Total operating current is the sum of the appropriate V_{DD} supply and V_{DDSYN} supply current.
7. Power dissipation measured with system clock frequency of 16.78 MHz. Power dissipation is calculated using the following expression:

$$P_{D} = \text{Maximum } V_{DD} (I_{DDSYN} + I_{DD})$$

8. Input capacitance is periodically sampled rather than 100% tested.

Table A-3 AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc)

Num	Characteristic	Symbol	Min	Max	Unit
F1[2]	Frequency of Operation (32.768 kHz or 4.194 MHz crystal)	f	0.13	16.78	MHz
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B[3]	External Clock Input Period	t_{Xcyc}	59.6	—	ns
2, 3	Clock Pulse Width	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B, 3B[3]	External Clock Input High/Low Time	t_{XCHL}	29.8	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	5	ns
4A, 5A	Rise and Fall Time — All Outputs except CLKOUT	t_{rf}	—	8	ns
4B, 5B	External Clock Rise and Fall Time	t_{XCrf}	—	5	ns
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
7[14]	Clock High to Address, Data, FC, SIZE, $\overline{RMC}$ High Impedance	t_{CHAZx}	0	59	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
9A[4]	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)	t_{STSA}	–15	15	ns
11[14]	Address, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
14B	$\overline{AS}$, $\overline{CS}$ Width Asserted (Fast Write Cycle)	t_{SWDW}	40	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	59	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High	t_{SNRN}	15	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	15	—	ns
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	70	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$	t_{DVASN}	15	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	15	—	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	80	ns
29[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6, 7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
30[6]	CLKOUT Low to Data In Invalid (Fast Cycle Hold)	t_{CLDI}	15	—	ns
30A[6]	CLKOUT Low to Data In High Impedance	t_{CLDH}	—	90	ns
31[8]	$\overline{DSACK}$ Asserted to Data In Valid	t_{DADI}	—	50	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35[9,14]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39	$\overline{BG}$ Width Negated	t_{GH}	2	—	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns

Table A-3 AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc)

47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48[10]	$\overline{DSACK}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	28	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns
56	$\overline{RESET}$ Pulse Width (Reset Instruction)	t_{HRPW}	512	—	t_{cyc}
57	$\overline{BERR}$ Negated to $\overline{HALT}$ Negated (Rerun)	t_{BNHN}	0	—	ns
70	Clock Low to Data Bus Driven (Show)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	10	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[11]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[12]	t_{RSTR}	—	10	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. Minimum system clock frequency is four times the crystal frequency, subject to specified limits.
3. Minimum external clock high and low times are based on a 50% duty cycle. The minimum allowable t_{XCYC} period will be reduced when the duty cycle of the external clock signal varies. The relationship between external clock input duty cycle and minimum t_{XCYC} is expressed:
   Minimum t_{XCYC} period = minimum t_{XCHL} / (50% – external clock input duty cycle tolerance).
   To achieve maximum operating frequency (f_{sys}) while using an external clock input, adjust clock input duty cycle to obtain a 50% duty cycle on CLKOUT.
4. Specification 9A is the worst-case skew between $\overline{AS}$ and $\overline{DS}$ or $\overline{CS}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{AS}$ and $\overline{DS}$ to fall outside the limits shown in specification 9.
5. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
6. These hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
8. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{DSACK}$ low to data setup time (specification 31) and $\overline{DSACK}$ low to $\overline{BERR}$ low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time (specification 27A) for the following clock cycle.
9. To ensure coherency during every operand transfer, $\overline{BG}$ will not be asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete and $\overline{RMC}$ is negated.
10. In the absence of $\overline{DSACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
11. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SIM drives low for 512 t_{cyc}.
12. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
13. Address access time = (2.5 + WS) t_{cyc} – t_{CHAV} – t_{DICL}
    Chip select access time = (2 + WS) t_{cyc} – t_{CLSA} – t_{DICL}
    Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = –1.
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

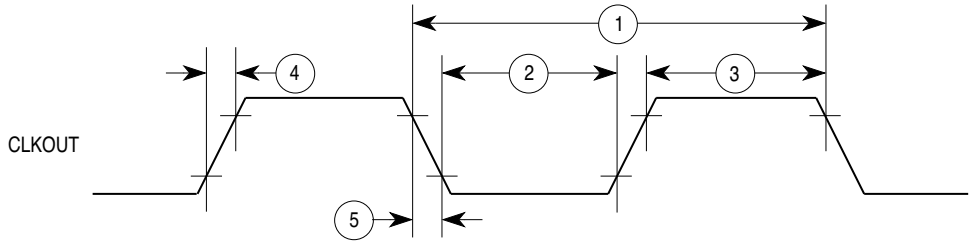
Table A-4 ECLK Bus Timing

(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc)

Num	Characteristic	Symbol	Min	Max	Unit
E1[2]	ECLK Low to Address Valid	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ delay)	t_{ECSD}	—	120	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	100	—	ns
E6	Read Data Setup Time	t_{EDSR}	30	—	ns
E7	Read Data Hold Time	t_{EDHR}	15	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	115	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14[3]	Address Access Time (Read)	t_{EACC}	386	—	ns
E15[4]	Chip Select Access Time (Read)	t_{EACS}	326	—	ns
E16	Address Setup Time	t_{EAS}	—	1/2	t_{cyc}

NOTES:

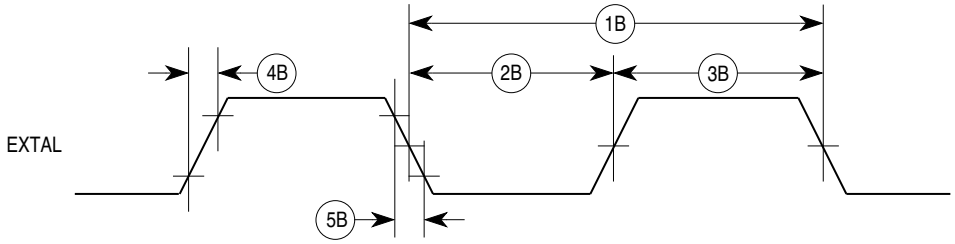
1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When the previous bus cycle is not an ECLK cycle, the address may be valid before ECLK goes low.
3. Address access time = $t_{Ecyc} - t_{EAD} - t_{EDSR}$
4. Chip select access time = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$



NOTE: Timing shown with respect to 20% and 70% V_{DD}.

68300 CLKOUT TIM

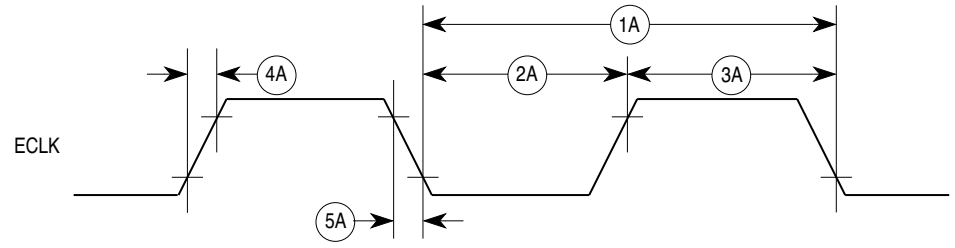
Figure A-1 CLKOUT Output Timing Diagram



NOTE: Timing shown with respect to 20% and 70% V_{DD}.
Pulse width shown with respect to 50% V_{DD}.

68300 EXT CLK INPUT TIM

Figure A-2 External Clock Input Timing Diagram



NOTE: Timing shown with respect to 20% and 70% V_{DD}.

68300 ECLK OUTPUT TIM

Figure A-3 ECLK Output Timing Diagram

Table A-5 Key to Figures A-1, A-2, A-3

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B[3]	External Clock Input Period	t_{Xcyc}	59.6	—	ns
2, 3	Clock Pulse Width	t_{CW}	24	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
2B,3B[3]	External Clock Input High/Low Time	t_{XCHL}	29.8	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	5	ns
4A, 5A	ECLK Rise and Fall Time	t_{Erf}	—	8	ns
4B, 5B	External Clock Rise and Fall Time	t_{XCrf}	—	5	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
3. Minimum external clock high and low times are based on a 50% duty cycle. The minimum allowable t_{XCYC} period will be reduced when the duty cycle of the external clock signal varies. The relationship between external clock input duty cycle and minimum t_{XCYC} is expressed:
   Minimum t_{XCYC} period = minimum t_{XCHL} / (50% – external clock input duty cycle tolerance).
   To achieve maximum operating frequency (f_{sys}) while using an external clock input, adjust clock input duty cycle to obtain a 50% duty cycle on CLKOUT.

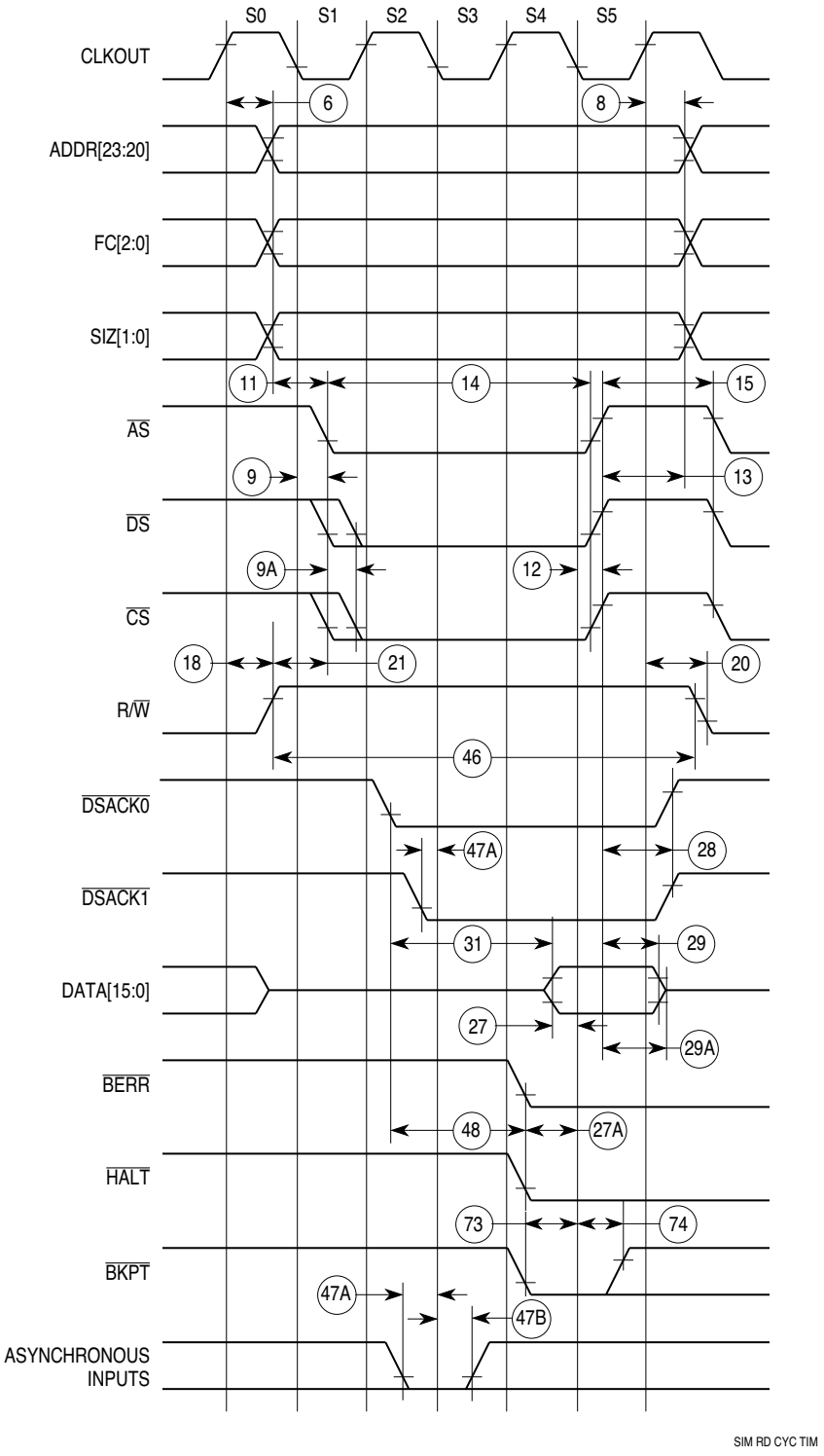


Figure A-4 Read Cycle Timing Diagram

Table A-6 Key to Figure A-4

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
9A[4]	$\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)	t_{STSA}	–15	15	ns
11[14]	$A_{DD}R$, FC, SIZ, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	100	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	15	—	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	80	ns
29[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6, 7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
31[8]	$\overline{DSACK}$ Asserted to Data In Valid	t_{DADI}	—	50	ns
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48[10]	$\overline{DSACK}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	t_{DABA}	—	30	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
4. This is the worst-case skew between $\overline{AS}$ and $\overline{DS}$ or $\overline{CS}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{AS}$ and $\overline{DS}$ to fall outside the limits shown in specification 9.
5. If multiple chip selects are used, $\overline{CS}$ width negated (#15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip selects does not apply to chip selects used for ECLK cycles.
6. These hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on synchronous reads. The user is free to use either hold time.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
8. If the asynchronous setup time (#47A) requirements are satisfied, the $\overline{DSACK}$ low to data setup time (#31) and $\overline{DSACK}$ low to $\overline{BERR}$ low setup time (#48) can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time (#27A) for the following clock cycle.
10. In the absence of $\overline{DSACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (47A).
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

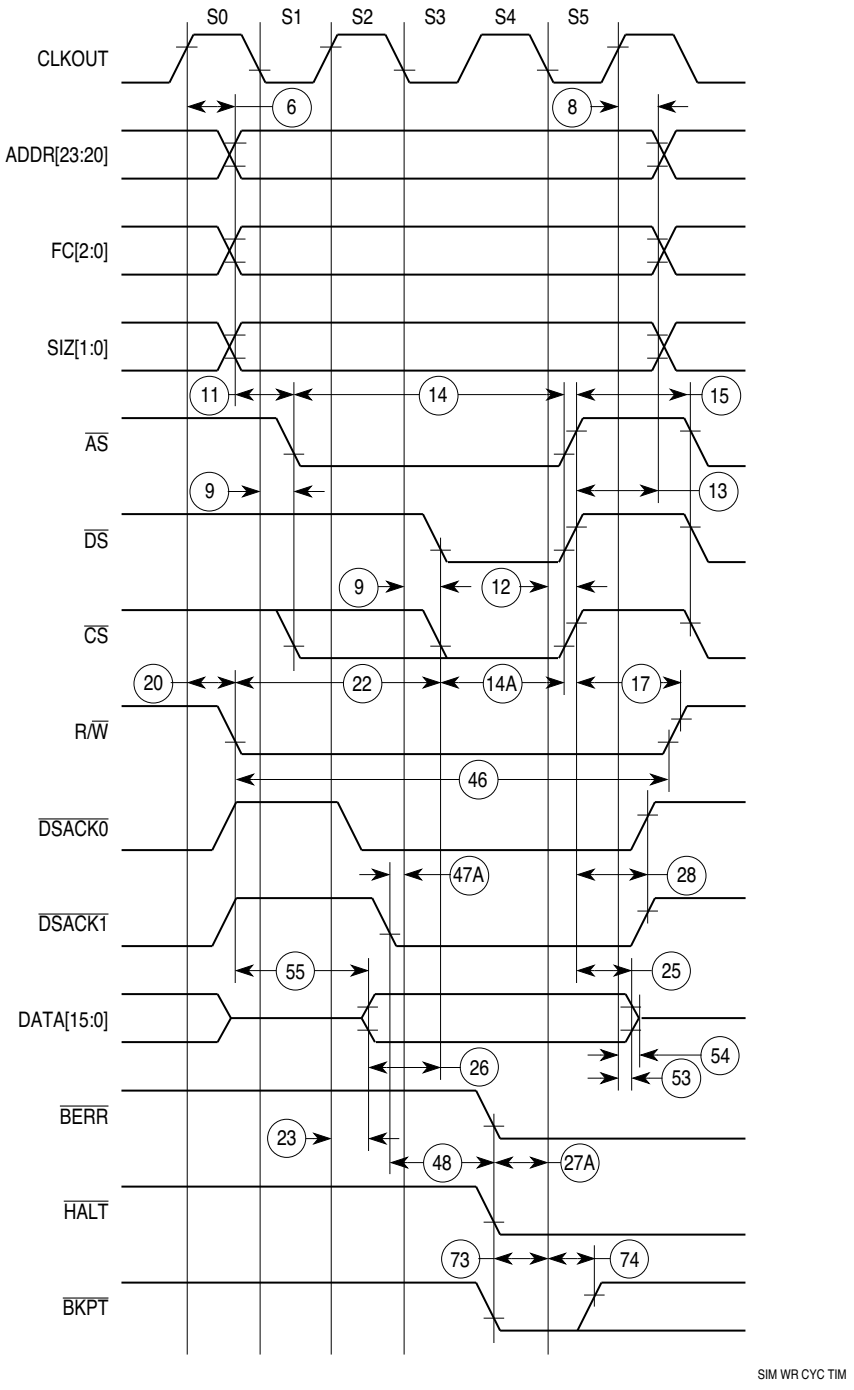


Figure A-5 Write Cycle Timing Diagram

Table A-7 Key to Figure A-5

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
11[14]	Address, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted Write	t_{SWAW}	45	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High	t_{SNRN}	15	—	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	70	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	15	—	ns
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated	t_{SNDN}	0	80	ns
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns
48[10]	$\overline{DSACK}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted	t_{DABA}	—	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	28	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
10. In the absence of $\overline{DSACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

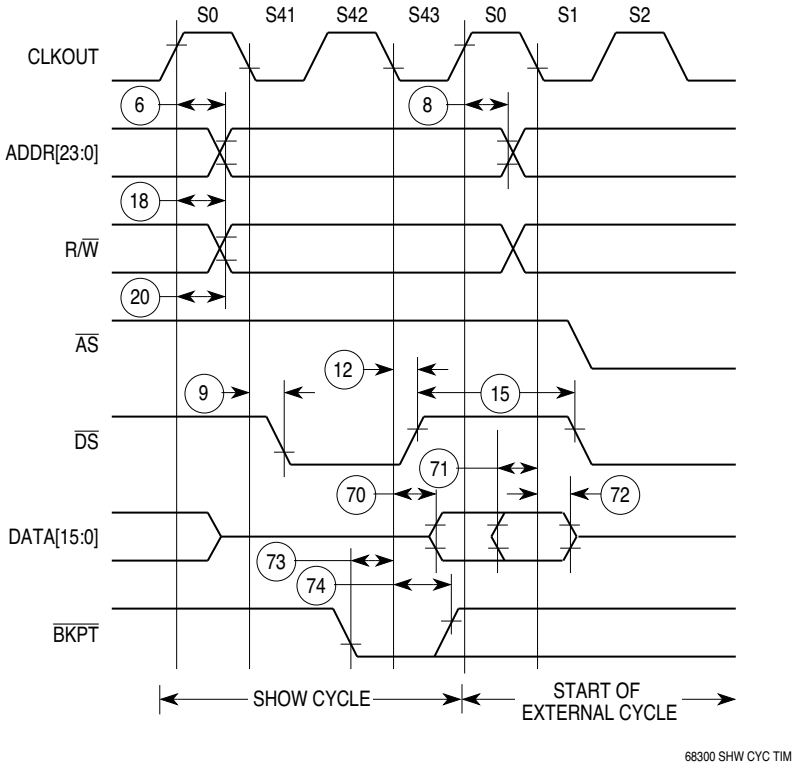


Figure A-6 Show Cycle Timing Diagram

Table A-8 Key to Figure A-6

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, RMC Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, RMC Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to AS, DS, CS Asserted	t_{CLSA}	2	25	ns
12	Clock Low to AS, DS, CS Negated	t_{CLSN}	2	29	ns
15[5]	AS, DS, CS Width Negated	t_{SN}	40	—	ns
18	Clock High to R/W High	t_{CHRH}	0	29	ns
20	Clock High to R/W Low	t_{CHRL}	0	29	ns
70	Clock Low to Data Bus Driven (Show)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	10	—	ns
73	BKPT Input Setup Time	t_{BKST}	15	—	ns
74	BKPT Input Hold Time	t_{BKHT}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. If multiple chip selects are used, CS width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The CS width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
14. RMC signal is not supported on CPU16-based MCUs.

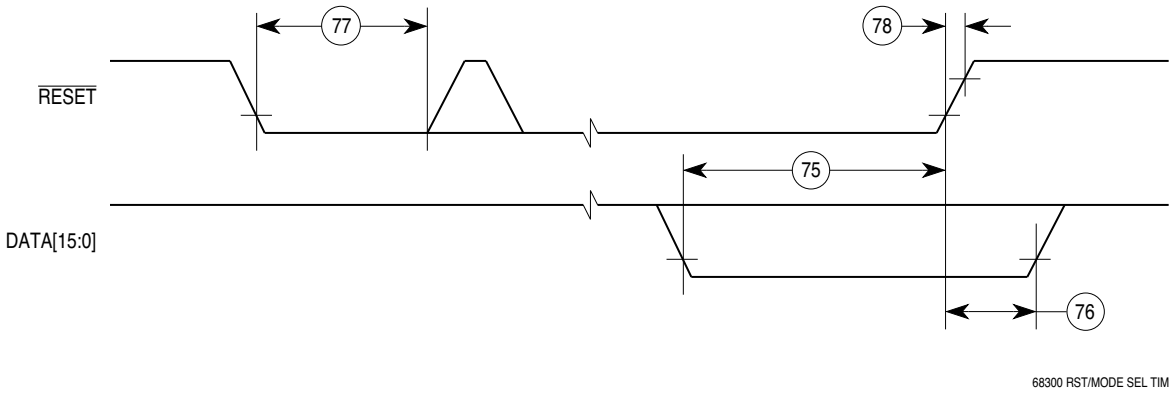


Figure A-7 Reset and Mode Select Timing Diagram

Table A-9 Key to Figure A-7

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
75	Mode Select Setup Time	t_{MSS}	20	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{\text{RESET}}$ Assertion Time[11]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{\text{RESET}}$ Rise Time[12]	t_{RSTR}	—	10	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.

11. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SIM drives low for 512 t_{cyc}.

12. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.

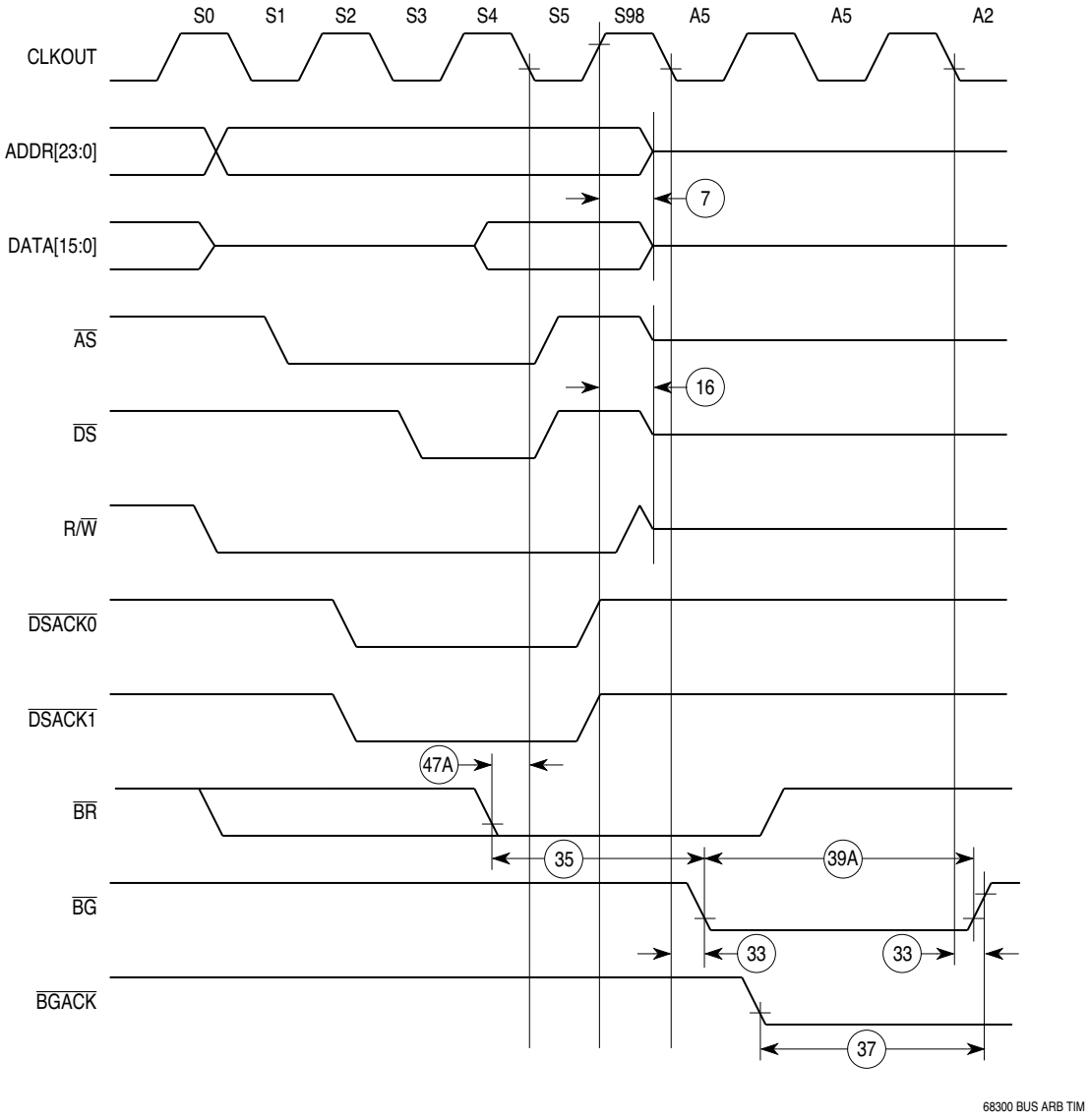


Figure A-8 Bus Arbitration Timing Diagram — Active Bus Case

Table A-10 Key to Figure A-8

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
7[14]	Clock High to Address, Data, FC, SIZE, $\overline{RMC}$ High Impedance	t_{CHAZx}	0	59	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	59	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBA}	—	29	ns
35[9]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
9. To ensure coherency during every operand transfer, $\overline{BG}$ will not be asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete and $\overline{RMC}$ is negated.
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

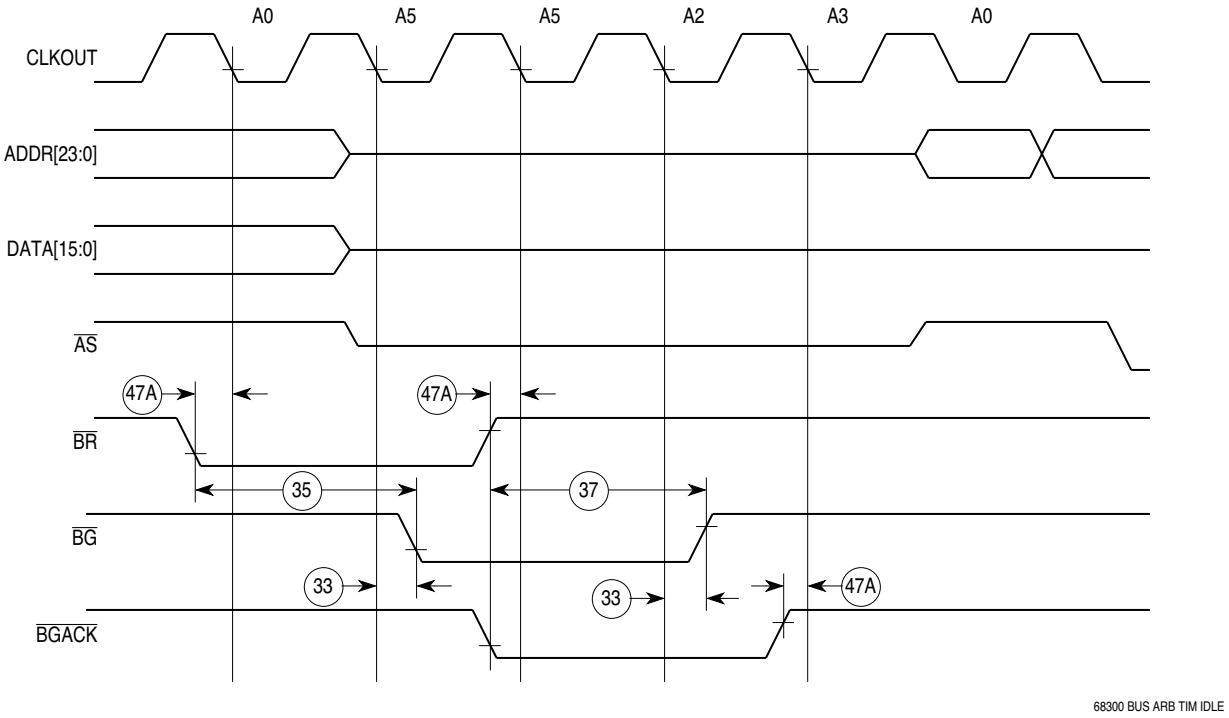


Figure A-9 Bus Arbitration Timing Diagram — Idle Bus Case

Table A-11 Key to Figure A-9

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBA}	—	29	ns
35[9]	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	t_{AIST}	5	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
9. To ensure coherency during every operand transfer, $\overline{BG}$ will not be asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete and $\overline{RMC}$ is negated.

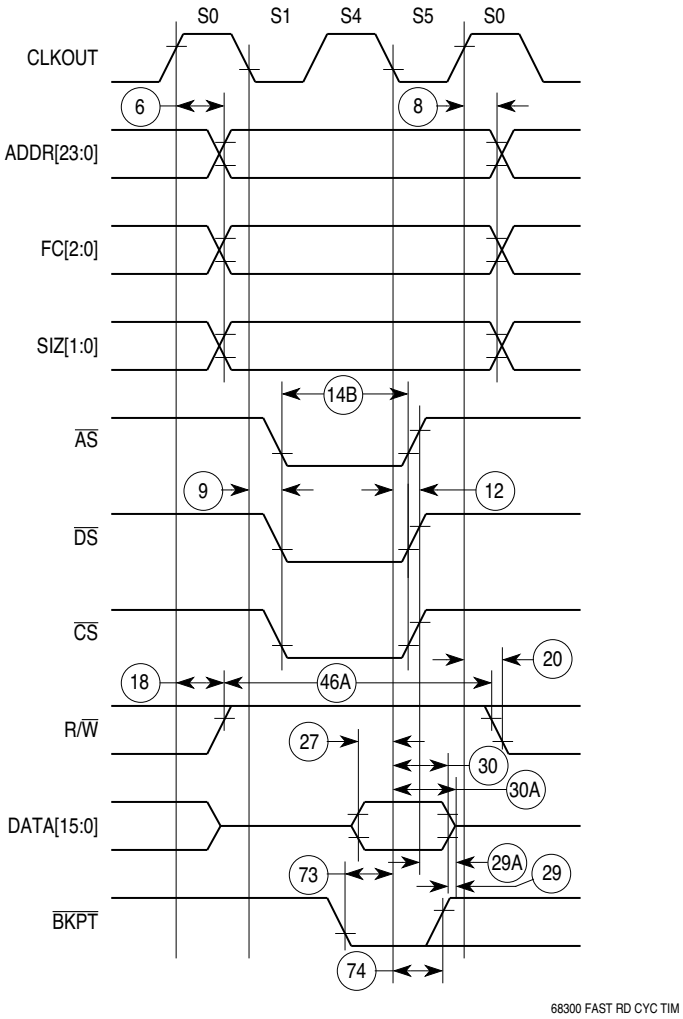


Figure A-10 Fast Termination Read Cycle Timing Diagram

Table A-12 Key to Figure A-10

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWDW}	40	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
30[6]	CLKOUT Low to Data In Invalid	t_{CLDI}	15	—	ns
30A[6]	CLKOUT Low to Data In High Impedance	t_{CLDH}	—	90	ns
46A	R/$\overline{W}$ Width Asserted	t_{RWAS}	90	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
6. These hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on synchronous reads. The user is free to use either hold time.
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

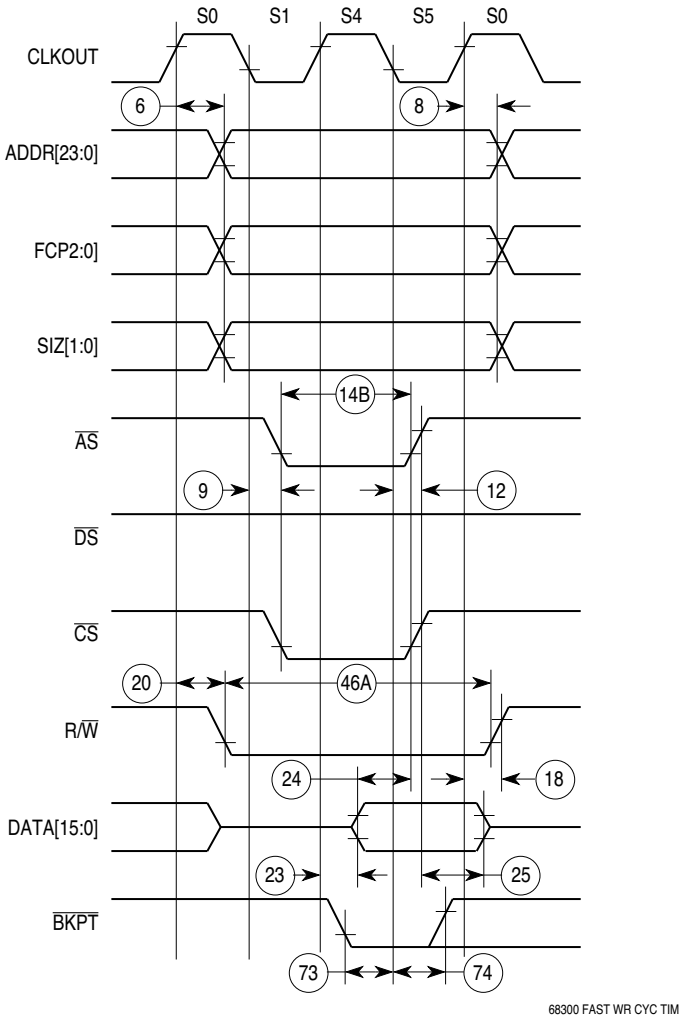


Figure A-11 Fast Termination Write Cycle Timing Diagram

Table A-13 Key to Figure A-11

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWDW}	40	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$	t_{DVASN}	15	—	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
46A	R/$\overline{W}$ Width Asserted	t_{RWAS}	90	—	ns
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	10	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.

14. $\overline{RMC}$ signal is not supported on CPU32-based MCUs.

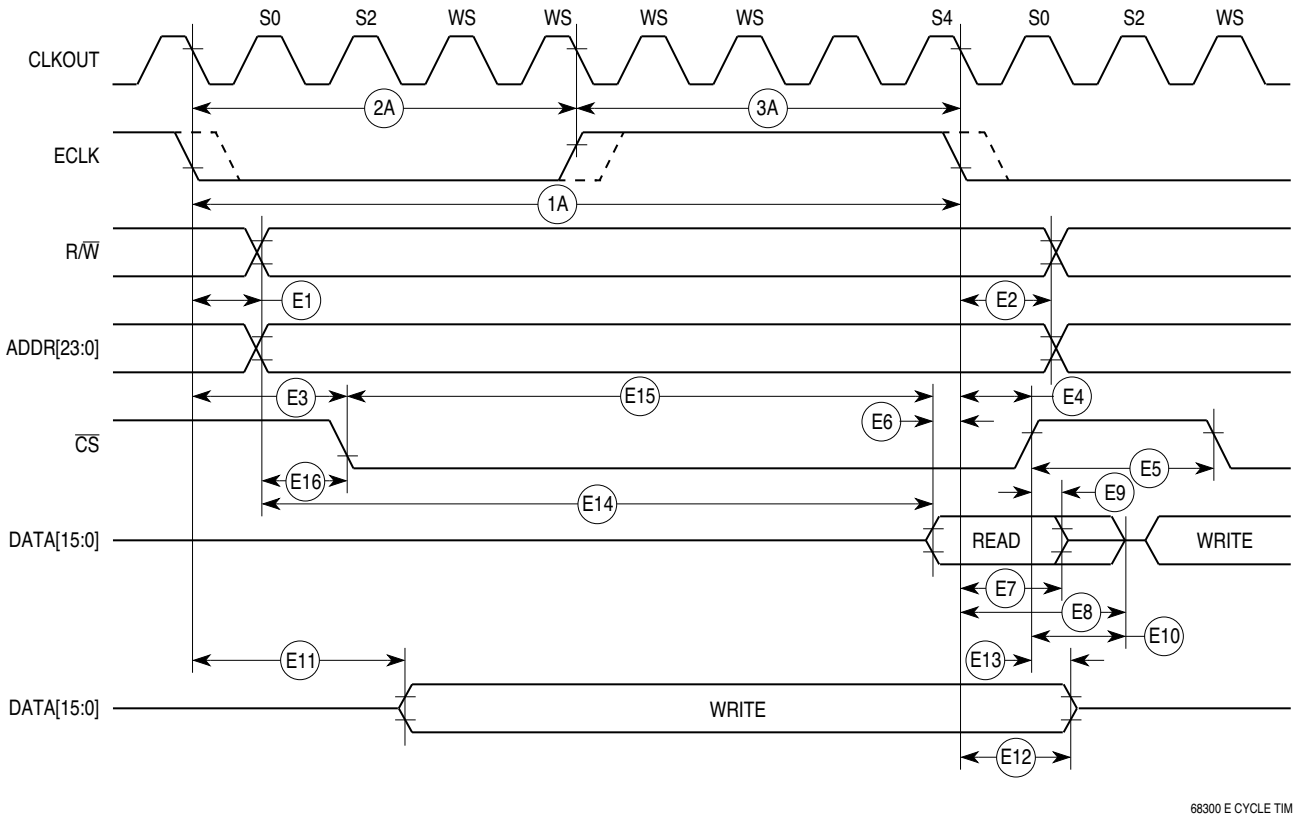


Figure A-12 ECLK Timing Diagram

Table A-14 Key to Figure A-12

(Abstracted from Tables A-3 and A-4; see tables for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
1A	ECLK Period	t_{Ecyc}	476	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	236	—	ns
4A, 5A	Rise and Fall Time — All Outputs except CLKOUT	t_{Erf}	—	8	ns
E1[2]	ECLK Low to Address and R/$\overline{W}$ Valid	t_{EAD}	—	60	ns
E2	ECLK Low to Address and R/$\overline{W}$ Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ delay)	t_{ECSD}	—	120	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	100	—	ns
E6	Read Data Setup Time	t_{EDSR}	30	—	ns
E7	Read Data Hold Time	t_{EDHR}	15	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	115	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14[3]	Address Access Time (Read)	t_{EACC}	386	—	ns
E15[4]	Chip Select Access Time (Read)	t_{EACS}	326	—	ns
E16	Address Setup Time	t_{EAS}	—	1/2	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When the previous bus cycle is not a synchronous ECLK bus cycle, the address may be valid before ECLK goes low.
3. Address access time = $t_{Ecyc} - t_{EAD} - t_{EDSR}$
4. Chip select access time = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$

14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

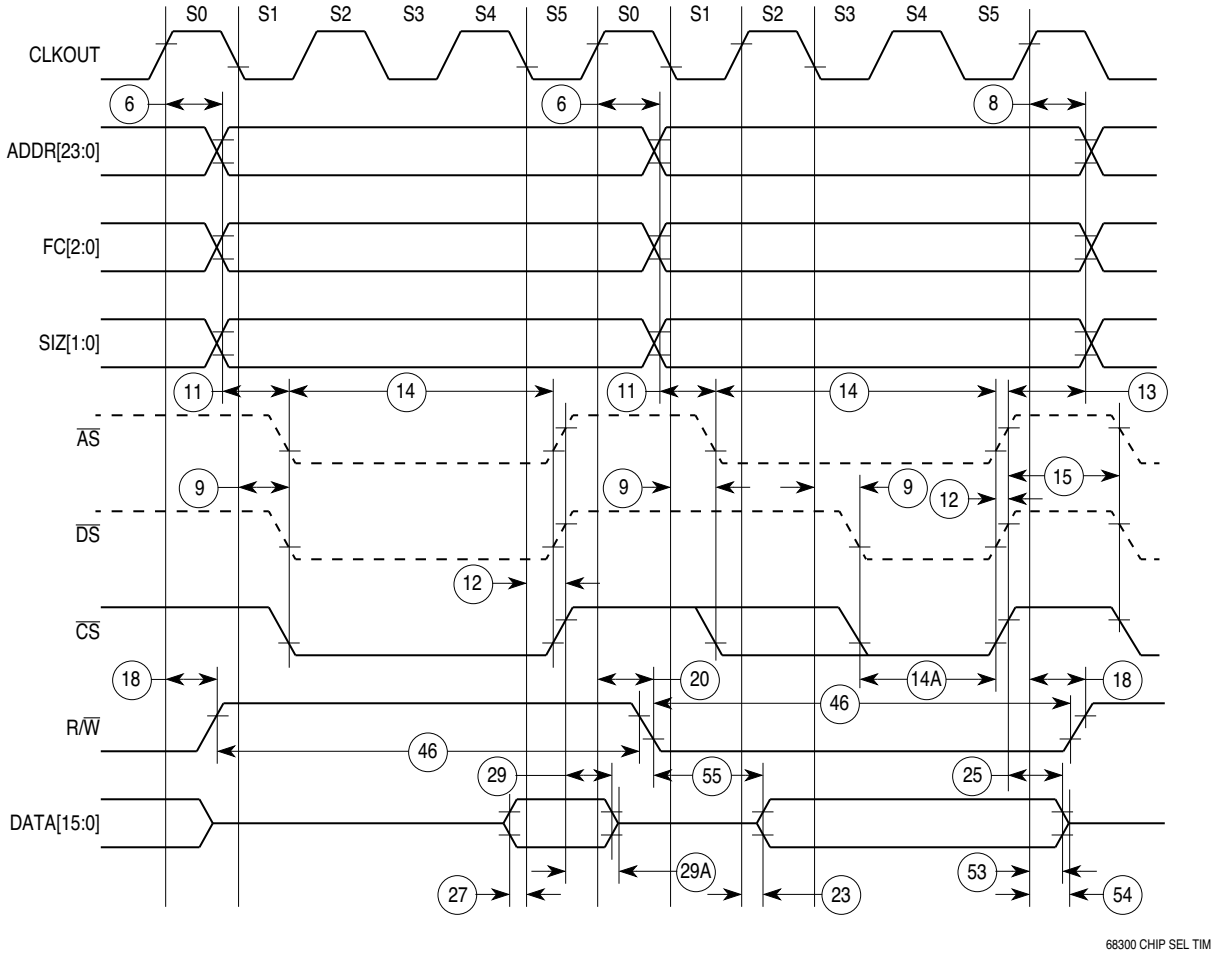


Figure A-13 Chip Select Timing Diagram

Table A-15 Key to Figure A-13

(Abstracted from Table A-3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Valid	t_{CHAV}	0	29	ns
8[14]	Clock High to Address, FC, SIZE, $\overline{RMC}$ Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	25	ns
11[14]	Address, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ Asserted	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{AS}$, $\overline{CS}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted Write	t_{SWAW}	45	—	ns
15[5]	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated	t_{SN}	40	—	ns
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	29	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
29[6]	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)	t_{SNDI}	0	—	ns
29A[6, 7]	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance	t_{SHDI}	—	55	ns
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	28	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
5. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
6. These hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on synchronous reads. The user is free to use either hold time.
7. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
14. $\overline{RMC}$ signal is not supported on CPU16-based MCUs.

APPENDIX BMEMORY MAP AND REGISTERS

B.1 SIM Memory Map

Table B-1 SIM Address Map

Access	Address	15 8	7 0
S	####00	SIM CONFIGURATION REGISTER (SIMCR)	
S	####02	SIM TEST REGISTER (SIMTR)	
S	####04	SYNTHESIZER CONTROL REGISTER (SYNCR)	
S	####06	UNUSED	RESET STATUS REGISTER (RSR)
S	####08	SYSTEM TEST REGISTER E (SIMTRE)	
S	####0A	UNUSED	UNUSED
S	####0C	UNUSED	UNUSED
S	####0E	UNUSED	UNUSED
S/U	####10	UNUSED	PORT E DATA (PORTE0)
S/U	####12	UNUSED	PORT E DATA (PORTE1)
S/U	####14	UNUSED	PORT E DATA DIRECTION (DDRE)
S	####16	UNUSED	PORT E PIN ASSIGNMENT (PEPAR)
S/U	####18	UNUSED	PORT F DATA (PORTF0)
S/U	####1A	UNUSED	PORT F DATA (PORTF1)
S/U	####1C	UNUSED	PORT F DATA DIRECTION (DDRF)
S	####1E	UNUSED	PORT F PIN ASSIGNMENT (PFPAR)
S	####20	UNUSED	SYSTEM PROTECTION CONTROL (SYPCR)
S	####22	PERIODIC INTERRUPT CONTROL REGISTER (PICR)	
S	####24	PERIODIC INTERRUPT TIMING REGISTER (PITR)	
S	####26	UNUSED	SOFTWARE SERVICE (SWSR)
S	####28	UNUSED	UNUSED
S	####2A	UNUSED	UNUSED
S	####2C	UNUSED	UNUSED
S	####2E	UNUSED	UNUSED
S	####30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	####32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	####34	TEST MODULE SHIFT COUNT (TSTSC)	
S	####36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	####38	TEST MODULE CONTROL (CREG)	
S/U	####3A	TEST MODULE DISTRIBUTED (DREG)	
S	####3C	UNUSED	UNUSED
S	####3E	UNUSED	UNUSED
S/U	####40	UNUSED	PORT C DATA (PORTC)
S/U	####42	UNUSED	UNUSED
S	####44	CHIP-SELECT PIN ASSIGNMENT REGISTER (CSPAR0)	
S	####46	CHIP-SELECT PIN ASSIGNMENT REGISTER (CSPAR1)	
S	####48	CHIP-SELECT BASE ADDRESS REGISTER BOOT (CSBARBT)	
S	####4A	CHIP-SELECT OPTION REGISTER BOOT (CSORBT)	
S	####4C	CHIP-SELECT BASE ADDRESS REGISTER 0 (CSBAR0)	

Table B-1 SIM Address Map (Continued)

Access	Address	15 8	7 0
S	####4E	CHIP-SELECT OPTION REGISTER 0 (CSOR0)	
S	####50	CHIP-SELECT BASE ADDRESS REGISTER 1 (CSBAR1)	
S	####52	CHIP-SELECT OPTION REGISTER 1 (CSOR1)	
S	####54	CHIP-SELECT BASE ADDRESS REGISTER 2 (CSBAR2)	
S	####56	CHIP-SELECT OPTION REGISTER 2 (CSOR2)	
S	####58	CHIP-SELECT BASE ADDRESS REGISTER 3 (CSBAR3)	
S	####5A	CHIP-SELECT OPTION REGISTER 3 (CSOR3)	
S	####5C	CHIP-SELECT BASE ADDRESS REGISTER 4 (CSBAR4)	
S	####5E	CHIP-SELECT OPTION REGISTER 4 (CSOR4)	
S	####60	CHIP-SELECT BASE ADDRESS REGISTER 5 (CSBAR5)	
S	####62	CHIP-SELECT OPTION REGISTER 5 (CSOR5)	
S	####64	CHIP-SELECT BASE ADDRESS REGISTER 6 (CSBAR6)	
S	####66	CHIP-SELECT OPTION REGISTER 6 (CSOR6)	
S	####68	CHIP-SELECT BASE ADDRESS REGISTER 7 (CSBAR7)	
S	####6A	CHIP-SELECT OPTION REGISTER 7 (CSOR7)	
S	####6C	CHIP-SELECT BASE ADDRESS REGISTER 8 (CSBAR8)	
S	####6E	CHIP-SELECT OPTION REGISTER 8 (CSOR8)	
S	####70	CHIP-SELECT BASE ADDRESS REGISTER 9 (CSBAR9)	
S	####72	CHIP-SELECT OPTION REGISTER 9 (CSOR9)	
S	####74	CHIP-SELECT BASE ADDRESS REGISTER 10 (CSBAR10)	
S	####76	CHIP-SELECT OPTION REGISTER 10 (CSOR10)	
	####78	UNUSED	UNUSED
	####7A	UNUSED	UNUSED
	####7C	UNUSED	UNUSED
	####7E	UNUSED	UNUSED

B.2 SIM Registers

SIMCR — Module Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN		SUPV	MM	0	0	IARB			
RESET:															
0	0	0	0	DATA 11	0	0	0	1	1	0	0	1	1	1	1

EXOFF — External Clock Off

0 = The CLKOUT pin is driven from an internal clock source.

1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable

0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.

1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts during software debugging.

FRZBM — Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the bus monitor continues to operate.

1 = When FREEZE is asserted, the bus monitor is disabled.

SLVEN — Factory Test (Slave) Mode Enabled

0 = IMB is not available to an external tester.

1 = An external tester has direct access to the IMB.

SHEN[1:0] — Show Cycle Enable

This field determines what the external bus interface does with the external bus during internal transfer operations. Refer to **5.11 Show Cycles** for more information.

SHEN	Action
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

SUPV — Supervisor/Unrestricted Data Space

0 = Registers with access controlled by this bit are unrestricted

1 = Registers with access controlled by this bit are restricted to supervisor access only.

MM — Module Mapping

0 = Internal modules are addressed from $7FF000 — $7FFFFF.

1 = Internal modules are addressed from $FFF000 — $FFFFFF.

This bit can be written only once. Subsequent attempts to change this bit are ignored. Address space $7FF000 – $7FFFFF is inaccessible to the CPU16. On CPU16-based microcontrollers, MM must always be set. Initialization software for these MCUs should make certain MM remains set (its reset state) by writing a one to it.

IARB[3:0] — Interrupt Arbitration Field

IARB determines SIM interrupt arbitration priority. The reset value is $F (highest priority), to prevent SIM interrupts from being discarded during initialization. Refer to **3.1.4 Interrupt Arbitration Priority** and **SECTION 6 INTERRUPTS** for additional information.

SIMTR — System Integration Test Register **$####02**

SIMTR is used for factory test only.

SYNCR — Clock Synthesizer Control Register **$####04**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	U	U	0	0	0

W — Frequency Control (VCO)

0 = Base VCO frequency.
1 = VCO frequency multiplied by four.

Refer to **4.4 System Clock Frequency Control** for additional information.

X — Frequency Control Bit (Prescale)

0 = Base system clock frequency.
1 = System clock frequency multiplied by two.

Refer to **4.4 System Clock Frequency Control** for additional information.

Y[5:0] — Frequency Control (Counter)

The Y field is the initial value for the modulus 64 down counter in the synthesizer feedback loop. Values range from 0 to 63. Refer to **4.4 System Clock Frequency Control** for additional information.

EDIV — ECLK Divide Rate

0 = ECLK is system clock divided by 8.
1 = ECLK is system clock divided by 16.

Refer to **4.5 External Bus Clock** for additional information.

SLIMP — Limp Mode Status

0 = MCU is operating normally.
1 = Loss of reference signal — MCU operating in limp mode.

Refer to **4.7 Loss of Reference Signal** for additional information.

SLOCK — Synthesizer Lock

0 = VCO is enabled, but has not locked.
1 = VCO has locked on the desired frequency or system clock is external.

RSTEN — Reset Enable

0 = Loss of clock causes the MCU to operate in limp mode.
1 = Loss of clock causes system reset.

Refer to **4.7 Loss of Reference Signal** for additional information.

STSIM — Stop Mode SIM Clock

0 = SIM clock driven by the external reference signal and the VCO is turned off during low-power stop.

1 = SIM clock driven by VCO during low-power stop.

This bit has an effect only if the PLL is configured to supply the clock signal (MODCLK held high during reset). Refer to **4.6 Low-Power Stop Operation** for additional information.

STEXT — Stop Mode External Clock

0 = CLKOUT held low during low-power stop.

1 = CLKOUT driven from SIM clock during low-power stop.

Refer to **4.6 Low-Power Stop Operation** for additional information.

RSR — Reset Status Register **$####06**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								EXT	POW	SW	DBF	0	LOC	SYS	TST

EXT — External Reset

Reset was caused by an external signal.

POW — Power-On Reset

Reset was caused by the power-on reset circuit.

SW — Software Watchdog Reset

Reset was caused by the software watchdog circuit.

DBF — Double Bus Fault Reset

Reset was caused by a double bus fault.

LOC — Loss of Clock Reset

Reset was caused by loss of clock reference signal.

SYS — System Reset

Reset was caused by the CPU32 RESET instruction. The CPU16 does not support this instruction.

TST — Test Submodule Reset

Reset was caused by the test submodule. This bit is set during system test only.

SIMTRE — System Integration Test Register (ECLK) **$####08**

The SIMTRE is used for factory test only.

PORTE0, PORTE1 — Port E Data Register **$####10, $####12**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:								U	U	U	U	U	U	U	U

DDRE — Port E Data Direction Register **$####14**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:								0	0	0	0	0	0	0	0

PEPAR — Port E Pin Assignment Register **$####16**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:								DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be a bus control signal.

Table B-2 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$*
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

*On CPU16-based MCUs, when PEPA3 is set, the PE3 pin, if connected, goes to logic level one. The CPU16 does not support the $\overline{RMC}$ function for this pin.

PORTF0, PORTF1 — Port F Data Register **$####18, $####1A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:								U	U	U	U	U	U	U	U

DDRF — Port F Data Direction Register **$####1C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:								0	0	0	0	0	0	0	0

PFPAR — Port F Pin Assignment Register **$####1E**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								PFPA7	PFPA6	PFPA5	PFPA4	PFPA3	PFPA2	PFPA1	PFPA0
RESET:								DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9

Any bit cleared to zero defines the corresponding pin to be an I/O pin. Any bit set to one defines the corresponding pin to be an interrupt request signal or MODCLK.

Table B-3 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA7	PF7	$\overline{\text{IRQ7}}$
PFPA6	PF6	$\overline{\text{IRQ6}}$
PFPA5	PF5	$\overline{\text{IRQ5}}$
PFPA4	PF4	$\overline{\text{IRQ4}}$
PFPA3	PF3	$\overline{\text{IRQ3}}$
PFPA2	PF2	$\overline{\text{IRQ2}}$
PFPA1	PF1	$\overline{\text{IRQ1}}$
PFPA0	PF0	$\overline{\text{MODCLK}}$

SYPCR — System Protection Control Register **$####20**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		HME	BME	BMT	
RESET:								1		0	0	0	0	0	0

SWE — Software Watchdog Enable
0 = Software watchdog disabled
1 = Software watchdog enabled

SWP — Software Watchdog Prescale
0 = Software watchdog clock not prescaled
1 = Software watchdog clock prescaled by 512

SWT[1:0] — Software Watchdog Timing
This field selects software watchdog time-out period.

Software Watchdog Ratio

SWP	SWT	Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

HME — Halt Monitor Enable
0 = Disable halt monitor function
1 = Enable halt monitor function

BME — Bus Monitor External Enable

0 = Disable bus monitor function for an internal-to-external bus cycle.
1 = Enable bus monitor function for an internal-to-external bus cycle.

BMT[1:0] — Bus Monitor Timing

This field selects bus monitor time-out period.

Bus Monitor Period

BMT	Bus Monitor Time-out Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

PICR — Periodic Interrupt Control Register **$####22**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL			PIV							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

PIRQL[2:0] — Periodic Interrupt Request Level

This field determines the priority of periodic interrupt requests.

PIV[7:0] — Periodic Interrupt Vector

The bits of this field contain the periodic interrupt vector number supplied by the SIM when the CPU acknowledges an interrupt request.

PITR — Periodic Interrupt Timer Register **$####24**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PTP	PITM							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MODCLK}}$	0	0	0	0	0	0	0	0

PTP — Periodic Timer Prescaler Control

1 = Periodic timer clock prescaled by a value of 512
0 = Periodic timer clock not prescaled

PITM[7:0] — Periodic Interrupt Timing Modulus

This is the 8-bit timing modulus used to determine periodic interrupt rate. Use the following expression to calculate timer period.

PIT Period = [(PIT Modulus)(Prescaler value)(4)]/EXTAL Frequency

SWSR — Software Service Register **$####26**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	0	0	0	0	0	0	0
RESET:															
								0	0	0	0	0	0	0	0

PORTC — Port C Data Register **$####40**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:															
								1	1	1	1	1	1	1	1

CSPAR0 — Chip Select Pin Assignment Register 0 **$####44**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CSPA0[6]		CSPA0[5]		CSPA0[4]		CSPA0[3]		CSPA0[2]		CSPA0[1]		$\overline{\text{CSBOOT}}$	
RESET:															
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains seven two-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect.

Table B-4 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	$\overline{\text{CS4}}$	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CS2}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CS1}}$	$\overline{\text{BG}}$	—
CSPA0[1]	$\overline{\text{CS0}}$	$\overline{\text{BR}}$	—
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$	—	—

CSPAR1 — Chip Select Pin Assignment Register 1 **$####46**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	CSPA1[4]		CSPA1[3]		CSPA1[2]		CSPA1[1]		CSPA1[0]	
RESET:															
0	0	0	0	0	0	DATA7	1	DATA6	1	DATA5	1	DATA4	1	DATA3	1

CSPAR1 contains five two-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

Table B-5 CSPAR1 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA1[4]	$\overline{\text{CS10}}$	ADDR23	ECLK
CSPA1[3]	$\overline{\text{CS9}}$	ADDR22	PC6
CSPA1[2]	$\overline{\text{CS8}}$	ADDR21	PC5
CSPA1[1]	$\overline{\text{CS7}}$	ADDR20	PC4
CSPA1[0]	$\overline{\text{CS6}}$	ADDR19	PC3

Table B-6 Pin Assignment Encodings

Bit Field	Description
00	Discrete Output
01	Alternate Function
10	Chip Select (8-Bit Port)
11	Chip Select (16-Bit Port)

CSBARBT — Chip Select Base Address Register Boot ROM **$####48**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR[10:0] — Chip Select Base Address Registers **$####4C–$####74**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADDR[15:3] — Base Address Field

This field sets the starting address of a particular address space.

BLKSZ — Block Size Field

This field determines the size of the block above the base address that is enabled by the chip select.

CSORBT — Chip Select Option Register Boot ROM **$####4A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

CSOR[10:0] — Chip Select Option Registers **$####4E–$####76**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE	BYTE		R/$\overline{W}$		STRB	$\overline{DSACK}$				SPACE		IPL			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MODE — Timing Mode

The MODE bit determines whether chip-select operation emulates asynchronous bus operation or is synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23. Refer to **7.5 Chip-Select Timing** for additional information.

0 = Emulate asynchronous bus operation

1 = Synchronize chip-select assertion to ECLK

BYTE — Upper/Lower Byte Option

This field enables or disables the chip-select circuit and, for 16-bit ports, determines which combinations of size and ADDR0 pins will cause the chip select to be asserted. Refer to **7.6 Chip Selects and Dynamic Bus Sizing** for more information.

00 = Disable
01 = Lower byte
10 = Upper byte
11 = Both Bytes

R/$\overline{W}$ — Read/Write

This field causes a chip select to be asserted only for a read, only for a write, or for both read and write.

00 = Reserved
01 = Read only
10 = Write only
11 = Read/Write

STRB — Address Strobe/Data Strobe

The STRB bit controls the timing of a chip-select assertion in asynchronous mode. This bit has no effect in synchronous mode.

0 = Synchronize chip select assertion with address strobe
1 = Synchronize chip select assertion with data strobe

$\overline{DSACK}$ — Data and Size Acknowledge

This field specifies the source of $\overline{DSACK}$ when chip-select cycles emulate asynchronous bus cycles, and controls wait state insertion. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for details.

SPACE — Address Space Select

The SPACE field determines the address space in which a chip select is asserted. An access must have the space type represented by SPACE encoding in order for a chip-select signal to be asserted. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

00 = CPU space
01 = User space
10 = Supervisor space
11 = Supervisor or user space

IPL — Interrupt Priority Level

This field selects the interrupt level when a chip select is used for interrupt acknowledge. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

000 = Any Level
001 = Level 1
010 = Level 2
011 = Level 3
100 = Level 4
101 = Level 5
110 = Level 6
111 = Level 7

$\overline{\text{AVEC}}$ — Autovector Enable

This field specifies whether to generate an internal $\overline{\text{AVEC}}$ signal during an interrupt acknowledge cycle initiated by assertion of an $\overline{\text{IRQ}}$ pin when match conditions are met. Refer to **7.8 Using Chip Selects in Interrupt Acknowledge Cycles** for additional information.

0 = Disable $\overline{\text{AVEC}}$ generation

1 = Enable $\overline{\text{AVEC}}$ generation

INDEX

–A–

–B–

–C–

–D–

–E–

–F–

–G–

–H–

–I–

–L–

–M–

–O–

–P–

–R–

–S–

–T–

–U–

–V–

–W–

–X–

–Y–