Motorola Microcontroller Family

SCIM

SINGLE-CHIP 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**

SECTION 7 CHIP SELECTS

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 8 RESET AND SYSTEM INITIALIZATION

SECTION 9GENERAL-PURPOSE I/O

TABLE OF CONTENTS
(Continued)

Paragraph **Title** **Page**

SECTION 10 REDUCED PIN-COUNT SCIM

APPENDIX A ELECTRICAL CHARACTERISTICS

APPENDIX BREGISTER SUMMARY

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 single-chip integration module (SCIM), 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 this 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 single-chip integration module (SCIM) is a module on many Motorola 16- and 32-bit modular microcontroller units (MCUs). SCIM-based MCUs contain a SCIM, a CPU, and some combination of memory, input/output, timer, and additional modules. The modules perform the following tasks:

- The SCIM supplies a clock signal to the rest of the microcontroller, provides system protection features, and manages the external bus. In addition, the SCIM 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 hardware components for processing instructions and data. The CPU also works with the SCIM 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 SCIM, it is therefore 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 SCIM. **1.3 CPU-Specific Differences Affecting SCIM Operation** summarizes the differences between the CPU32-based SCIM and the CPU16-based SCIM.
- 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.
- Input/output, timer, and additional 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.

The different modules on an MCU communicate with one another and with external components via the intermodule bus (IMB), a standardized internal bus developed to facilitate design of modular microcontrollers. The IMB supports 24 address and 16 data lines. The SCIM external bus, when fully expanded, supports 24 address and 16 data lines as well. Refer to **1.2 Bus Configuration and Reset Mode Selection** for additional information.

NOTE

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

The SCIM 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 SCIM, 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 nine chip-select signals. Each chip-select signal has an associated base register and option register that contain the programmable characteristics of that chip select. Two additional chip selects allow external emulation of on-chip SCIM I/O ports and ROM arrays. 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.
- The system test block incorporates hardware necessary for testing the MCU. Its use in normal applications is not supported.

When the external data bus is fully expanded, two data ports, port E and port F, are available for general-purpose input and output if not required for their alternate function. When the external data bus is partially expanded, port H is available in addition to ports E and F. When the SCIM is configured for single-chip operation, six data ports are available for general-purpose I/O: ports A, B, E, F, G, and H. A port data register, data direction register, and pin assignment register are associated with each port.

The SCIM works with the CPU to support exception processing, including reset and interrupt processing. Refer to **SECTION 6 INTERRUPTS** and **SECTION 8 RESET AND SYSTEM INITIALIZATION** for additional information.

NOTE

> Some SCIM-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 the device. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** for a discussion of SCIM operation with a reduced pin count.

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

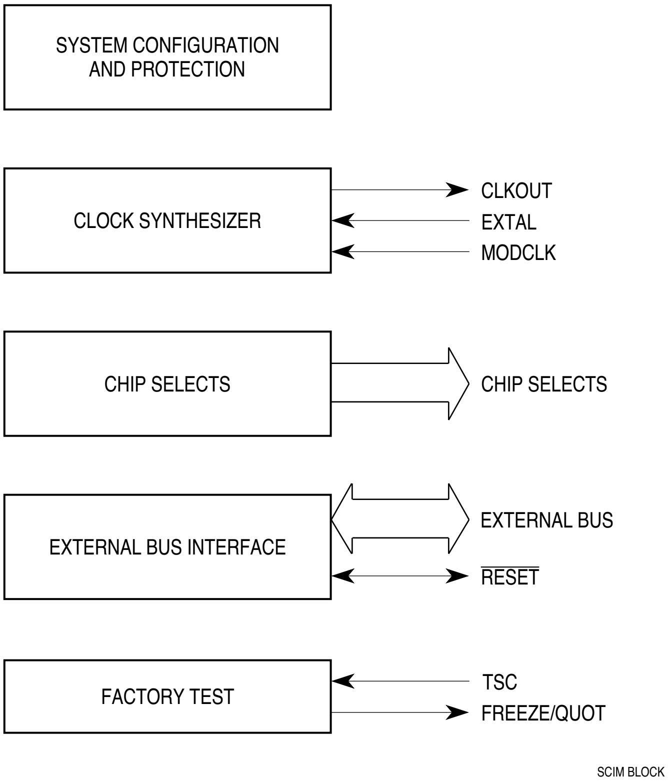


Figure 1-1 Single-Chip Integration Module Block Diagram

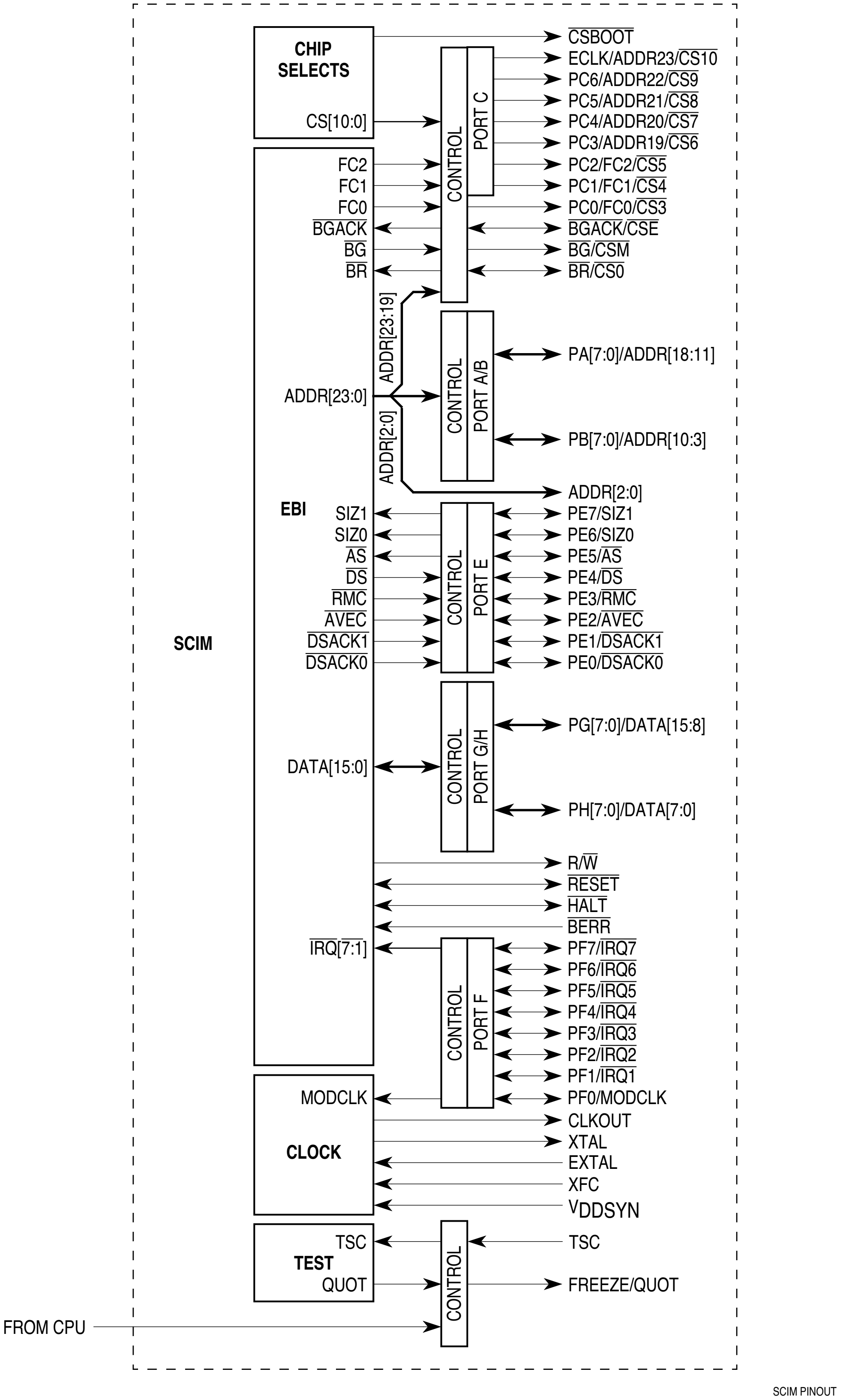


Figure 1-2 SCIM Input and Output Signals

1.1 SCIM Address Map

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 SCIM configuration register (SCIMCR) determines where the control register block is located in the system memory map. When MM = 0, 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 SCIM 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 SCIM address map. The column labeled “Access” indicates the privilege level at which the CPU must be operating to access the register. A designation of “S” indicates that supervisor 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 SCIM register addresses relative to the SCIM 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 SCIM Address Map

Access	Address	15 8	7 0
S	$####00	SCIM CONFIGURATION REGISTER (SCIMCR)	
S	$####02	MODULE TEST (SCIMTR)	
S	$####04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
S	$####06	UNUSED	RESET STATUS REGISTER (RSR)
S	$####08	MODULE TEST E (SCIMTRE)	
S/U	$####0A	PORT A DATA (PORTA)	PORT B DATA (PORTB)
S/U	$####0C	PORT G DATA (PORTG)	PORT H DATA (PORTH)
S/U	$####0E	PORT G DATA DIRECTION (DDRG)	PORT H DATA DIRECTION (DDRH)
S/U	$####10	UNUSED	PORT E DATA (PORTE0)
S/U	$####12	UNUSED	PORT E DATA (PORTE1)
S/U	$####14	PORT A/B DATA DIRECTION (DDRAB)	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 (PICR)	
S	$####24	PERIODIC INTERRUPT TIMING (PITR)	
S	$####26	UNUSED	SOFTWARE SERVICE (SWSR)
S	$####28	UNUSED	PORT F EDGE DETECT FLAGS (PORTFE)
S	$####2A	UNUSED	PORT F EDGE DETECT INTERRUPT VECTOR (PFIVR)
S	$####2C	UNUSED	PORT F EDGE DETECT INTERRUPT LEVEL (PFLVR)
S/U	$####2E	UNUSED	
S	$####30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$####32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$####34	TEST MODULE SHIFT COUNT A (TSTSCA)	TEST MODULE SHIFT COUNT B (TSTSCB)
S	$####36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$####38	TEST MODULE CONTROL (CREG)	
S/U	$####3A	TEST MODULE DISTRIBUTED REGISTER (DREG)	
S/U	$####3C	UNUSED	UNUSED
S/U	$####3E	UNUSED	UNUSED
S/U	$####40	UNUSED	PORT C DATA (PORTC)
S/U	$####42	UNUSED	UNUSED
S	$####44	CHIP-SELECT PIN ASSIGNMENT 0 (CSPAR0)	
S	$####46	CHIP-SELECT PIN ASSIGNMENT 1 (CSPAR1)	
S	$####48	CHIP-SELECT BASE ADDRESS BOOT (CSBARBT)	
S	$####4A	CHIP-SELECT OPTION BOOT (CSORBT)	
S	$####4C	CHIP-SELECT BASE 0 (CSBAR0)	
S	$####4E	CHIP-SELECT OPTION 0 (CSOR0)	
S	$####50	UNUSED	
S	$####52	UNUSED	
S	$####54	UNUSED	

Table 1-1 SCIM Address Map (Continued)

Access	Address	15 8 7 0
S	$####56	UNUSED
S	$####58	CHIP-SELECT BASE 3 (CSBAR3)
S	$####5A	CHIP-SELECT OPTION 3 (CSOR3)
S	$####5C	UNUSED
S	$####5E	UNUSED
S	$####60	CHIP-SELECT BASE 5 (CSBAR5)
S	$####62	CHIP-SELECT OPTION 5 (CSOR5)
S	$####64	CHIP-SELECT BASE 6 (CSBAR6)
S	$####66	CHIP-SELECT OPTION 6 (CSOR6)
S	$####68	CHIP-SELECT BASE 7 (CSBAR7)
S	$####6A	CHIP-SELECT OPTION 7 (CSOR7)
S	$####6C	CHIP-SELECT BASE 8 (CSBAR8)
S	$####6E	CHIP-SELECT OPTION 8 (CSOR8)
S	$####70	CHIP-SELECT BASE 9 (CSBAR9)
S	$####72	CHIP-SELECT OPTION 9 (CSOR9)
S	$####74	CHIP-SELECT BASE 10 (CSBAR10)
S	$####76	CHIP-SELECT OPTION 10 (CSOR10)
	$####78	UNUSED
	$####7A	UNUSED
	$####7C	UNUSED
	$####7E	UNUSED

1.2 Bus Configuration and Reset Mode Selection

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

The logic state of certain pins during reset determine SCIM operating configuration. During reset, the SCIM reads pin configuration from DATA[11:0], internal module configuration from DATA[15:12], and basic operating information from $\overline{\text{BERR}}$, MODCLK, and $\overline{\text{BKPT}}$.

The data bus pins are normally pulled high internally during reset, causing the MCU to default to a specific configuration. The user must drive the desired pins low during reset to achieve alternate configurations. $\overline{\text{BERR}}$, MODCLK, and $\overline{\text{BKPT}}$ do not contain internal pull-ups and must be driven high or low to achieve the desired configuration.

Basic operating options include system clock selection, background mode disable/enable, and external bus configuration. The MCU can be configured as a fully-expanded MCU with an MC68020-style 24-bit external address bus and 16-bit data bus with chip selects, a single-chip MCU with no external address and data bus, or a partially-expanded MCU with a 24-bit address bus and an 8-bit external data bus.

Table 1-2 summarizes basic configuration options.

Table 1-2 Basic Configuration Options

Select Pin	Pin Pulled High During Reset	Pin Pulled Low During Reset
MODCLK	Synthesized System Clock	External System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled
$\overline{\text{BERR}}$	Expanded Mode	Single-Chip Mode
DATA1 (if $\overline{\text{BERR}}$ = 1)	8-Bit Expanded Mode	16-Bit Expanded Mode

Many SCIM pins, including data and address bus pins, have multiple functions. The reset functions of these pins coming out of reset depend on the external bus configuration selected during reset.

In expanded modes, the values of DATA[11:0] during reset determine the function of pins with multiple functions. The functions of some pins can be changed subsequently by writing to the appropriate pin assignment register. Data bus pins have internal pull-ups and must be pulled low to achieve the desired alternate configuration. DATA[15:12] may also be used during reset to configure modules other than the SCIM. Refer to the user's manual for the particular MCU for more information.

External bus configuration determines which address and data bus lines are used and which general-purpose I/O ports are available. ADDR[18:3] serve as pins for ports A and B when the MCU is operating in single-chip mode. DATA[7:0] serve as port H pins in partially expanded and single-chip modes, and DATA[15:8] serve as port G pins during single-chip operation. **Table 1-3** shows the three possible address and data bus pin configurations.

Table 1-3 Address and Data Bus Configuration Options

Bus Configuration	Mode-Select Pins[1]		Address Bus/Data Bus/Port Distribution		
	$\overline{\text{BERR}}$	DATA1	Address Bus Pins [18:3]	Data Bus Pins [15:0]	I/O Ports
16-Bit Expanded	1	0	ADDR[18:3]	DATA[15:0]	—
8-Bit Expanded	1	1	ADDR[18:3]	DATA[15:8]	DATA[7:0] = Port H
Single Chip	0	X	None	None	ADDR[18:11] = Port A ADDR[10:3] = Port B DATA[15:8] = Port G DATA[7:0] = Port H

NOTES:

1. DATA1 has a weak pull-up that is enabled during reset. $\overline{\text{BERR}}$ does not have a weak pull-up and must be driven to the desired state during reset.

1.3 CPU-Specific Differences Affecting SCIM Operation

This reference manual can be used with Motorola modular MCUs that contain a single-chip integration module, regardless of whether the chip contains a CPU16 or CPU32. Certain aspects of SCIM 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 SCIM operation are summarized in **Table 1-4**.

Table 1-4 CPU Differences Affecting SCIM 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 SCIMCR must equal 1	MM bit in SCIMCR can equal 0 or 1
Exception Vector Table	Each vector (except reset vector) is one word; vector table is 512 bytes and cannot be relocated (there is no vector base register)	Each vector (except reset vector) is two words; vector table is 1024 bytes and can be relocated (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 $\overline{\text{HALT}}$ is equivalent to assertion of $\overline{\text{BERR}}$ alone	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 SCIM signals and pins. For a more detailed discussion of a particular signal, refer to the section of this manual that discusses the SCIM 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 SCIM, some of the pins described in this section may not be available. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** for additional information. Refer to the user's manual for the particular MCU for a list of the pins implemented on the chip.

2.1 Pin Characteristics

Table 2-1 describes the different types of output drivers on SCIM pins. **Table 2-2** lists the output driver type and other characteristics of each SCIM pin.

Table 2-1 SCIM 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 the bus is granted to an external master. Refer to **5.1 Bus Signal Descriptions** 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. For pins that can be configured for general-purpose input or output, the “Discrete I/O” column indicates whether the pin can be used for both input and output or for output only.

Table 2-2 SCIM Pin Characteristics

Pin Mnemonic	Discrete I/O	Output Driver	Input Synchronized	Input Hysteresis
ADDR23/$\overline{\text{CS10}}$/ECLK	O	A	—	N
ADDR[22:20]/$\overline{\text{CS[9:7]}}$/ PC[6:4][1]	O	A	—	N
ADDR19/$\overline{\text{CS6}}$/PC3	O	A	—	N
ADDR[18:11]/PA[7:0]	I/O	A	Y	Y
ADDR[10:3]/PB[7:0]	I/O	A	Y	Y
ADDR[2:0]	—	A	Y	N
$\overline{\text{AS}}$/PE5	I/O	B	Y	Y
$\overline{\text{AVEC}}$/PE2[1]	I/O	B	Y	N
$\overline{\text{BERR}}$	—	B	Y	N
$\overline{\text{BG}}$/$\overline{\text{CSM}}$	—	B	—	—
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$	—	B	Y	N
$\overline{\text{BR}}$/$\overline{\text{CS0}}$	—	B	Y	N
CLKOUT	—	A	—	—
$\overline{\text{CSBOOT}}$[1]	—	B	—	—
DATA[15:8]/PG[7:0][2]	I/O	Aw	Y	Y
DATA[7:0]/PH[7:0][2]	I/O	Aw	Y	Y
$\overline{\text{DS}}$/PE4	I/O	B	Y	Y
$\overline{\text{DSACK1}}$/PE1	I/O	B	Y	N
$\overline{\text{DSACK0}}$/PE0[1]	I/O	B	Y	N
EXTAL[3]	—	—	—	Special
FC2/$\overline{\text{CS5}}$/PC2	O	A	—	N
FC1/PC1	O	A	—	N
FC0/$\overline{\text{CS3}}$/PC0	O	A	—	N
$\overline{\text{HALT}}$[1]	—	Bo	Y	N
$\overline{\text{IRQ[7:6]}}$/PF[7:6]	I/O	B	Y	Y
$\overline{\text{IRQ[5:1]}}$/PF[5:1][1]	I/O	B	Y	Y
MODCLK/PF0[2]	I/O	B	Y	Y
R/$\overline{\text{W}}$	—	A	Y	N
$\overline{\text{RESET}}$	—	Bo	Y	Y
$\overline{\text{RMC}}$/PE3[1]	I/O	B	Y	Y
SIZ[1:0]/PE[7:6]	I/O	B	Y	Y
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. EXTAL, XFC, and XTAL are clock reference connections.

2.2 Signal Descriptions

The following tables are a quick reference to SCIM 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 SCIM Signal Characteristics

Signal Name	Signal Type	Active State
ADDR[23:0]	Bus	—
$\overline{\text{AS}}$	Output	0
$\overline{\text{AVEC}}$	Input	0
$\overline{\text{BERR}}$	Input	0
$\overline{\text{BG}}$	Output	0
$\overline{\text{BGACK}}$	Input	0
$\overline{\text{BR}}$	Input	0
CLKOUT	Output	—
$\overline{\text{CS[10:5]}}$, $\overline{\text{CS3}}$, $\overline{\text{CS0}}$	Output	0
$\overline{\text{CSBOOT}}$	Output	0
$\overline{\text{CSE}}$	Output	0
$\overline{\text{CSM}}$	Output	0
DATA[15:0]	Bus	—
$\overline{\text{DS}}$	Output	0
$\overline{\text{DSACK[1:0]}}$	Input	0
ECLK	Output	—
EXTAL	Input	—
FC[2:0]	Output	—
FREEZE	Output	1
$\overline{\text{HALT}}$	Input/Output	0
$\overline{\text{IRQ[7:1]}}$	Input	0
MODCLK	Input	—
PA[7:0]	Input/Output	(Port)
PB[7:0]	Input/Output	(Port)
PC[6:0]	Output	(Port)
PE[7:0]	Input/Output	(Port)
PF[7:0]	Input/Output	(Port)
PG[7:0]	Input/Output	(Port)
PH[7:0]	Input/Output	(Port)
R/$\overline{\text{W}}$	Output	1/0
$\overline{\text{RESET}}$	Input/Output	0
$\overline{\text{RMC}}$	Output	0
SIZ[1:0]	Output	—
TSC	Input	1
XFC	Input	—
XTAL	Output	—

Table 2-4 SCIM Signal Function

Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus
Address Strobe	$\overline{\text{AS}}$	Indicates that a valid address is on the address bus
Autovector	$\overline{\text{AVEC}}$	Requests an automatic vector during interrupt acknowledge
Bus Error	$\overline{\text{BERR}}$	Indicates that a bus error has occurred
Bus Grant	$\overline{\text{BG}}$	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	$\overline{\text{BGACK}}$	Indicates that an external device has assumed bus mastership
Breakpoint	$\overline{\text{BKPT}}$	Signals a hardware breakpoint to the CPU
Bus Request	$\overline{\text{BR}}$	Indicates that an external device requires bus mastership
System Clockout	CLKOUT	System clock output
Chip Selects	$\overline{\text{CS[10:5]}}$, $\overline{\text{CS3}}$, $\overline{\text{CS0}}$	Select external devices at programmed addresses
Boot Chip Select	$\overline{\text{CSBOOT}}$	Chip select for external boot start-up ROM
Emulator Chip Select	$\overline{\text{CSE}}$	Chip select for external port emulator
Module Chip Select	$\overline{\text{CSM}}$	Chip select for external ROM emulator
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	$\overline{\text{DS}}$	During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus.
Data and Size Acknowledge	$\overline{\text{DSACK[1:0]}}$	Provide asynchronous data transfers and dynamic bus sizing
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
Halt	$\overline{\text{HALT}}$	Suspend external bus activity
Interrupt Request Level	$\overline{\text{IRQ[7:1]}}$	Provides an interrupt priority level to the CPU
Clock Mode Select	MODCLK	Selects the source of system clock
Port A	PA[7:0]	SCIM digital I/O port signals
Port B	PB[7:0]	SCIM digital I/O port signals
Port C	PC[6:0]	SCIM digital output port signals
Port E	PE[7:0]	SCIM digital I/O port signals
Port F	PF[7:0]	SCIM digital I/O port signals
Port G	PG[7:0]	SCIM digital I/O port signals
Port H	PH[7:0]	SCIM digital I/O port signals
Reset	$\overline{\text{RESET}}$	System reset
Read-Modify-Write	$\overline{\text{RMC}}$	Indicates indivisible read-modify-write cycle (CPU32 test-and-set instruction)
Read/Write	R/$\overline{\text{W}}$	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.

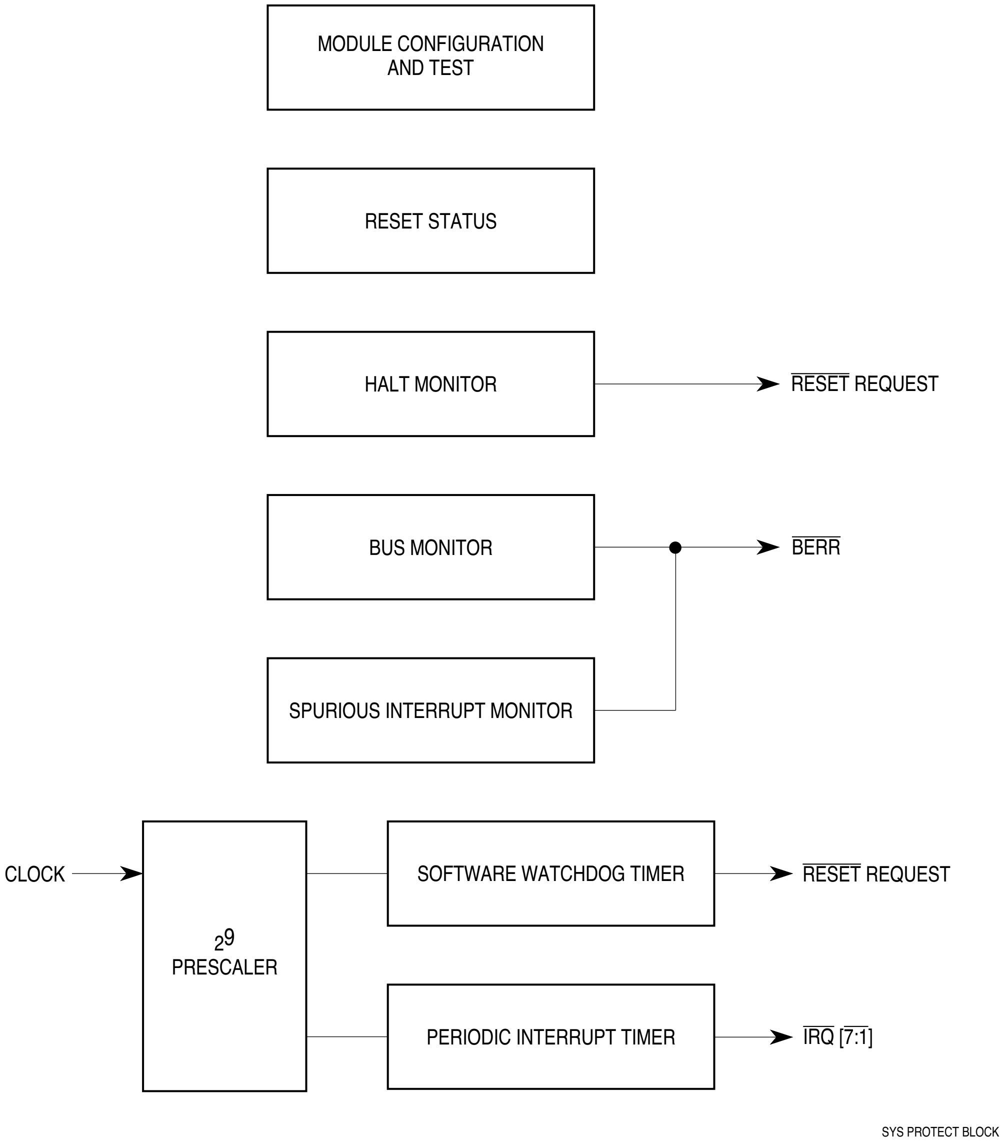

MODULE CONFIGURATION AND TEST
RESET STATUS
HALT MONITOR
$\overline{\text{RESET}}$ REQUEST
BUS MONITOR
$\overline{\text{BERR}}$
SPURIOUS INTERRUPT MONITOR
CLOCK
2^9 PRESCALER
SOFTWARE WATCHDOG TIMER
$\overline{\text{RESET}}$ REQUEST
PERIODIC INTERRUPT TIMER
$\overline{\text{IRQ}}$ [7:1]
SYS PROTECT BLOCK

Figure 3-1 System Configuration and Protection

3.1 Module Configuration and Testing

The SCIM configuration register (SCIMCR) 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 SCIMCR.

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 SCIM configuration register (SCIMCR) 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 is equal to one, 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 SCIM 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 CPU16-based 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 SCIM registers is permitted only when the CPU is operating at the supervisor privilege level. The remaining SCIM 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 SCIMCR is set, access to SCIM registers is permitted only when the CPU is operating at the supervisor level. If SUPV is cleared, then access to certain SCIM registers is permitted when the CPU is operating at either the supervisor or user privilege level. The SCIM 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 SCIM (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 SCIMCR determine how the SCIM responds to FREEZE assertion. Setting the freeze bus monitor (FRZBM) bit in the SCIMCR 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 SCIM, has an IARB (interrupt arbitration number) field in its module configuration register. Each IARB field must have 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 SCIMCR is $F. This prevents SCIM 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 Single-Chip Operation Support

The SCIMCR contains three bits that support single-chip operation. Setting the CPU development support disable bit (CPUD) disables (places in a high-impedance state) the instruction tracking pins whenever the FREEZE signal is not asserted. The instruction tracking pins on CPU32-based MCUs are IPIPE and IFETCH; on CPU16-based MCUs the pins are IPIPE0 and IPIPE1. When CPUD is cleared to zero, the instruction tracking pins operate normally.

Setting the address bus disable bit (ABD) disables ADDR[2:0] by placing the pins in a high-impedance state. During single-chip operation, the ADDR[23:3] pins are configured for discrete output or input/output, and ADDR[2:0] should normally be disabled.

Setting the R/$\overline{\text{W}}$ disable bit (RWD) disables the R/$\overline{\text{W}}$ pin. This pin is not normally used during single-chip operation.

The reset state of each of these three bits is one if $\overline{\text{BERR}}$ is held low during reset (configuring the MCU for single-chip operation) or zero if $\overline{\text{BERR}}$ is held high during reset.

3.1.6 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 whether the IMB is available to an external tester.

CAUTION

> When DATA11 is held low during reset, the bus is granted to an external master as soon as the $\overline{\text{BR}}$ pin is asserted. The user must be sure DATA11 is left high during reset to prevent this from occurring.

3.1.7 SCIM Configuration Register

The SCIM configuration register (SCIMCR) controls system configuration. It can be read or written at any time, except for the MM bit, which can only be written once.

SCIMR — SCIM Service Register **$####26**

15							8	7	6	5	4	3	2	1	0
EXOFF	FRZSW	FRZBM	CPUD	SLVEN	0	SHEN		SUPV	MM	ABD	RWD	IARB			
RESET:															
0	0	0	$\overline{\text{BERR}}$	$\overline{\text{DATA11}}$	0	0	0	1	1	$\overline{\text{BERR}}$	$\overline{\text{BERR}}$	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.

CPUD — CPU Development Support Disable

0 = Instruction pipeline signals available
1 = Instruction pipeline pins placed in high-impedance state unless a breakpoint occurs.

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

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.

SLVEN is a read-only status bit. Slave mode is used for factory testing. Reset state is the complement of DATA11 during reset for fully expanded operation.

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.

CAUTION

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.

ABD — Address bus disable

0 = ADDR[2:0] signals are available.
1 = ADDR[2:0] pins are disabled (placed in a high-impedance state).

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

RWD — R/$\overline{\text{W}}$ Pin Disable

0 = R/$\overline{\text{W}}$ signal is available.
1 = R/$\overline{\text{W}}$ signal is disabled (placed in a high-impedance state).

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

IARB[3:0] — Interrupt Arbitration Number

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

3.1.8 SCIM Test Registers

SCIMTRE — Single-Chip Integration Module Test Register (ECLK) **$####08**

The SCIMTRE is used for factory test only.

SCIMTR — Single-Chip Integration Module Test Register **$####02**

SCIMTR 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 external 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.

On MCUs designed to use a 32.768-kHz crystal, the software watchdog clock rate is derived from the frequency on the EXTAL pin. The SWP (software watchdog prescale) bit and the SWT (software watchdog timing) field in the SYPCR control the software watchdog clock rate. The SWP bit determines whether or not the EXTAL frequency is prescaled by 512. The SWT field determines whether the resulting signal (either EXTAL or EXTAL ÷ 512) is then divided by 2^9, 2^{11}, 2^{13}, or 2^{15}. The SWP bit and the SWT field thus determine the number (called the software watchdog divide ratio) by which the EXTAL frequency is divided to generate the software watchdog clock.

On MCUs designed to use a 4.194-MHz crystal, the frequency on the EXTAL pin is first divided by 128, producing a 32.768-kHz clock. This signal is then divided by the value determined by the SWP bit and the SWT field, as with MCUs that use a 32.768-kHz crystal.

Software watchdog prescaling and divide ratios are discussed in greater detail in the following subsections.

3.5.1 Software Watchdog Prescaling

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 (32.768-kHz Crystal)	Watchdog Prescaling (4.194-MHz Crystal)
0 (External Clock)	1	EXTAL ÷ 512	(EXTAL ÷ 128) ÷ 512
1 (Internal Clock)	0	EXTAL ÷ 1	(EXTAL ÷128)÷ 1

3.5.2 Software Watchdog Divide Ratio

The SWT field, in conjunction with the SWP bit, selects the divide ratio used to establish the software watchdog time-out period. **Table 3-3** gives the divide 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 Divide Ratio

SWP	SWT	Divide 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}

With a 32.768-kHz reference crystal, time-out period is computed from the EXTAL frequency and the divide ratio as follows:

$$\text{Time-out Period} = 1/(\text{EXTAL Frequency}/\text{Divide Ratio})$$

or

$$\text{Time-out Period} = \text{Divide Ratio}/\text{EXTAL Frequency}$$

With a 4.194-MHz reference crystal, the EXTAL frequency is first divided by 128, resulting in the following equations:

$$\text{Time-out Period} = 1/(\text{EXTAL Frequency}/\text{Divide Ratio} * 128)$$

or

$$\text{Time-out Period} = \text{Divide Ratio} * 128/\text{EXTAL Frequency}$$

Figure 3-2 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer with a 32.768-kHz external reference frequency. For MCUs that use a crystal reference with a 4.194-MHz frequency, the EXTAL frequency is divided by 128 before it is sent to the prescaler.

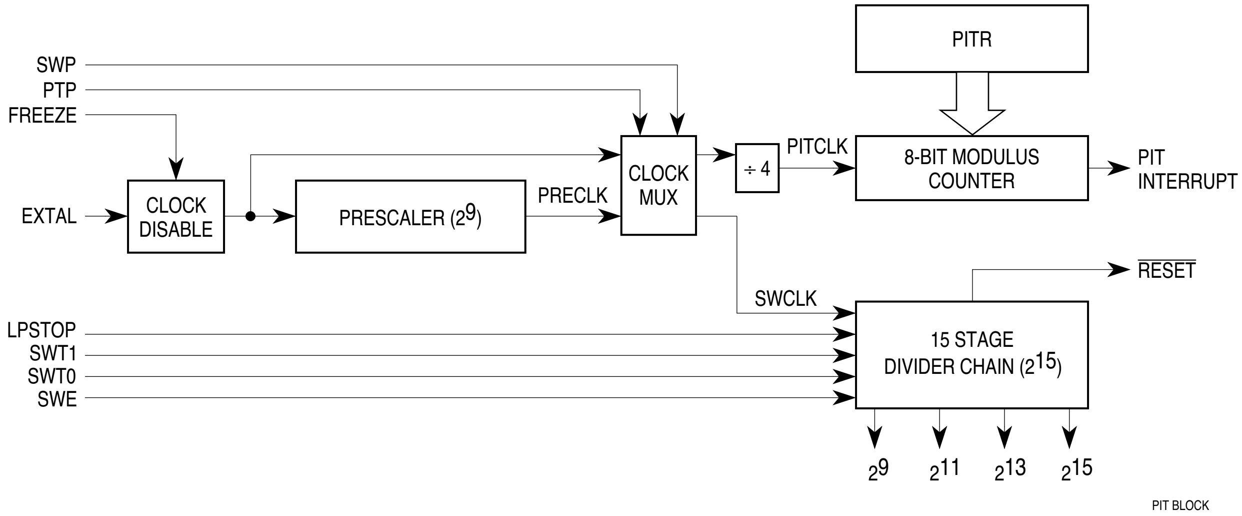


Figure 3-2 Periodic Interrupt Timer and Software Watchdog Timer

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

On MCUs designed to use a 32.768-kHz crystal, the periodic interrupt modulus counter is clocked by a signal derived from the buffered crystal oscillator (EXTAL) input pin. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines whether or not the EXTAL frequency is prescaled by 512.

With a 32.768-kHz reference frequency, either EXTAL or EXTAL divided by 512 (depending on the value of PTP) 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.

On MCUs designed to use a 4.194-MHz crystal, the frequency on the EXTAL pin is first divided by 128, producing a signal of approximately 32.768 kHz. Then either this value or this value divided by 512 (depending on the value of PTP) is divided by four before driving the modulus counter. In other words, after the EXTAL signal is divided by 128, the operation of the prescaler and modulus counter is the same as for an MCU with a 32.768-kHz crystal.

3.6.1.1 Pit Prescaling

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	SWP	PIT Prescaling (32.768-kHz Crystal)	PIT Prescaling (4.194-MHz Crystal)
0 (External Clock)	1	EXTAL ÷ 512	(EXTAL ÷ 128) ÷ 512
1 (Internal Clock)	0	EXTAL ÷ 1	(EXTAL ÷128)÷ 1

3.6.1.2 Pit Period with a 32.768-kHz Reference Frequency

With a 32.768-kHz reference frequency, use the following equation to calculate timer period.

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:

$$\begin{aligned}\text{PIT Period} &= \text{(PIT Modulus)(1)(4)/32768}\\ &= \text{PIT Modulus/8192}\end{aligned}$$

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),

$$\begin{aligned}\text{PIT Period} &= \text{(PIT Modulus)(512)(4)/32768}\\ &= \text{PIT Modulus/16}\end{aligned}$$

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,

$$\text{PIT Period} = \text{(PIT Modulus)}(122\ \mu\text{s})$$

With prescaler enabled,

$$\text{PIT Period} = \text{(PIT Modulus)(62.5 ms)}$$

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.

3.6.1.3 Pit Period with a 4.194-MHz Reference Frequency

Use the following equation to calculate timer period with a 4.194-MHz reference frequency.

PIT Period = (PIT Modulus)(Prescaler Value)(4)(128)/EXTAL Frequency

3.6.2 Pit 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 SCIM 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 STSCIM 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. LPSTOP does not reset the watchdog; 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 (or EXTAL ≥ 128 with a 4.194-MHz crystal reference) 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.

During low-power stop, internal control logic disables the necessary input buffers and forces input stages to a controlled state to prevent the inputs from floating and causing excessive current. Output pins remain in the state they were in before the MCU entered low-power stop, unless otherwise indicated in the reference manual for the MCU module that uses the pins.

3.8 System Protection Registers

The following registers are involved in system protection: the SCIMCR, the SWSR, the PICR, the PITR, and the SYPCR. A register diagram of the SCIMCR 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	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		8	7							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 SCIM 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							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 Control 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							8	7	6	5	4	3	2	1	0
NOT USED								SWE	SWP	SWT		HME	BME	BMT	
RESET:															
								1	$\overline{\text{MODCLK}}$	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.

Table 3-6 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.

Table 3-7 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 MCU operation is fully static, 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. Frequency control bits in the clock synthesizer control register (SYNCR) determine the operating frequency. Refer to **4.2 Clock Synthesizer Operation**, **4.3 System Clock Frequency Control**, and **4.4 External Circuit Design** for information on various aspects of clock synthesizer operation.

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. Refer to **4.5 External Clock Signal Input** for additional information on applying an external system clock signal.

CAUTION

> When using the MODCLK/PF0 pin for I/O, 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.

4.2 Clock Synthesizer Operation

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. For the PLL to operate, voltages must be applied to both the V_{DDSYN} pin and the other V_{DD} pins.

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. Frequency control bits in the SYNCR determine the operation of the divider. (Refer to **4.3 System Clock Frequency Control**.) The divider controls the frequency of one input to a phase comparator. The other phase comparator input is the reference signal. Once the synthesizer locks, 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.

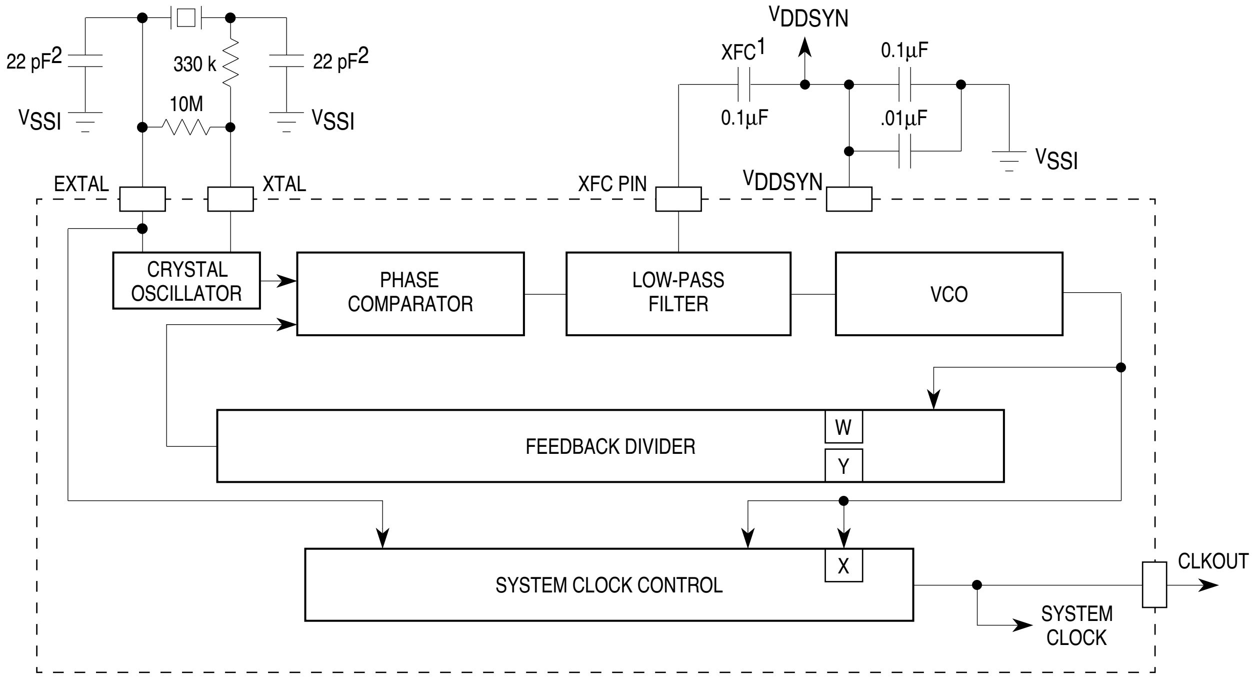
The synthesizer locks when VCO frequency is equal to the output of the divider/ counter. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever a comparator input changes, the synthesizer must relock. Lock status is shown by the SLOCK bit in the SYNCR. The MCU does not come out of the reset state until the synthesizer locks.

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. 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 **4.4.2 Crystal Tune-Up Procedure**.

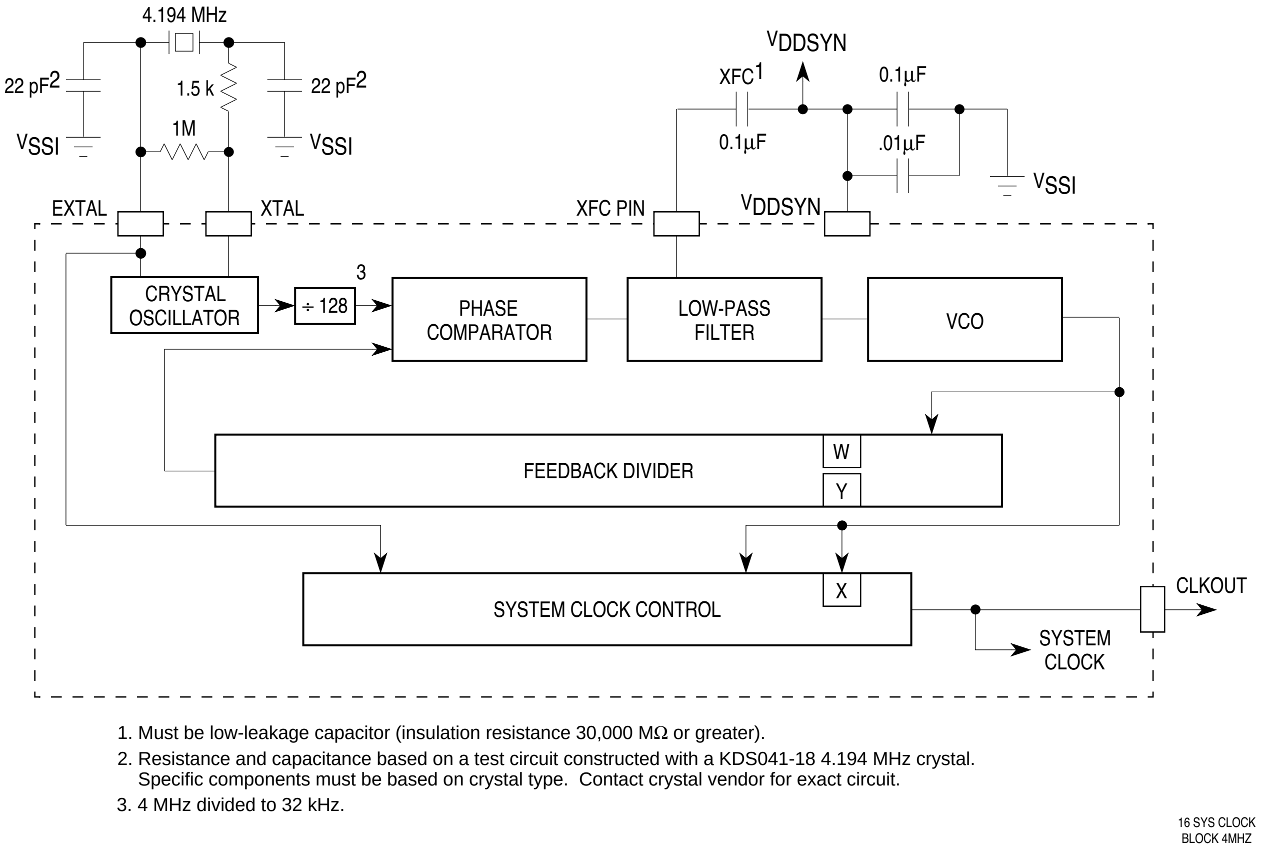
CAUTION

When providing samples to a crystal vendor for analysis, inform the vendor that MODCLK must be driven high during reset, enabling the PLL. The gain of the crystal driver circuitry depends on whether the PLL is enabled. If the vendor fails to enable the PLL, incorrect crystal component recommendations may result.



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.

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

4.3 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 X bit in the SYNCR controls a divide-by-two prescaler that is not in the synthesizer feedback loop. Setting X doubles the system clock speed without changing the VCO speed. Consequently, there is no VCO re-lock 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**.

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

With a reference frequency between 25 and 50 kHz (typically, 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 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 (W = %0, X = %0, Y = %111111). The system clock frequency at reset is thus 256 times the reference frequency. With a 32.768-kHz frequency reference crystal, system clock frequency at reset equals 8.39 MHz. Setting the X bit in the SYNCR after reset doubles the system clock frequency to 16.78 MHz.

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

With a reference frequency between 3.2 and 6.4 MHz (typically, 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 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 (W = %0, X = %0, Y = %111111). The system clock frequency at reset is thus twice the reference frequency. With a 4.194-MHz frequency reference crystal, system clock frequency at reset equals 8.39 MHz. Setting the X bit in the SYNCR after reset doubles the system clock frequency to 16.78 MHz.

4.3.3 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 new 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.

4.3.4 Frequency Control Tables

Table 4-1 shows how to compute the system clock frequency generated by a PLL designed to use a 32.768-kHz reference crystal, given various combinations of SYNCR bits. **Table 4-2** shows how to compute the clock frequency generated by a PLL designed to use a 4.194-MHz reference crystal. **Table 4-3** shows actual clock frequencies for the same combinations of SYNCR bits. The frequencies in **Table 4-3** are valid for typical values of 32.768-kHz and 4.194-MHz crystal references.

Table 4-1 Clock Control Multipliers for a 32.768-kHz Clock Source

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

Table 4-1 Clock Control Multipliers for a 32.768-kHz Clock Source

Modulus	**Prescalers**			
Y	**[W:X] = 00**	**[W:X] = 01**	**[W:X] = 10**	**[W:X] = 11**
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
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

Table 4-2 Clock Control Multipliers for a 4.194 MHz Clock Source

To obtain clock frequency, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell. Refer to **Table A-1** in **APPENDIX A ELECTRICAL CHARACTERISTICS** for maximum allowable clock rate (F_{SYS}). Refer to user’s manual for the particular MCU to determine appropriate clock source frequency.

Modulus	Prescalers			
Y	**[W:X] = 00**	**[W:X] = 01**	**[W:X] = 10**	**[W:X] = 11**
000000	.03125	.0625	.125	.25
000001	.0625	.125	.25	.5
000010	.09375	.1875	.375	.75
000011	.125	.25	.5	1
000100	.15625	.3125	.625	1.25
000101	.1875	.375	.75	1.5
000110	.21875	.4375	.875	1.75
000111	.25	.5	1	2
001000	.21825	.5625	1.125	2.25
001001	.3125	.625	1.25	2.5
001010	.34375	.6875	1.375	2.75
001011	.375	.75	1.5	3
001100	.40625	.8125	1.625	3.25
001101	.4375	.875	1.75	3.5
001110	.46875	.9375	1.875	3.75
001111	.5	1	2	4
010000	.53125	1.0625	2.125	4.25
010001	.5625	1.125	2.25	4.5
010010	.59375	1.1875	2.375	4.75
010011	.625	1.25	2.5	5
010100	.65625	1.3125	2.625	5.25
010101	.6875	1.375	2.75	5.5
010110	.71875	1.4375	2.875	5.75
010111	.75	1.5	3	6
011000	.78125	1.5625	3.125	6.25
011001	.8125	1.625	3.25	6.5
011010	.84375	1.6875	3.375	6.75
011011	.875	1.75	3.5	7
011100	.90625	1.8125	3.625	7.25
011101	.9375	1.875	3.75	7.5
011110	.96875	1.9375	3.875	7.75
011111	1	2	4	8
100000	1.03125	2.0625	4.125	8.25
100001	1.0625	2.125	4.25	8.5
100010	1.09375	2.1875	4.375	8.75
100011	1.125	2.25	4.5	9
100100	1.15625	2.3125	4.675	9.25
100101	1.1875	2.375	4.75	9.5
100110	1.21875	2.4375	4.875	9.75
100111	1.25	2.5	5	10
101000	1.28125	2.5625	5.125	10.25
101001	1.3125	2.625	5.25	10.5
101010	1.34375	2.6875	5.375	10.75
101011	1.375	2.75	5.5	11

Table 4-2 Clock Control Multipliers for a 4.194 MHz Clock Source

To obtain clock frequency, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell. Refer to **Table A-1** in **APPENDIX A ELECTRICAL CHARACTERISTICS** for maximum allowable clock rate (F_{sys}). Refer to user's manual for the particular MCU to determine appropriate clock source frequency.

Modulus	**Prescalers**			
Y	**[W:X] = 00**	**[W:X] = 01**	**[W:X] = 10**	**[W:X] = 11**
101100	1.40625	2.8125	5.625	11.25
101101	1.4375	2.875	5.75	11.5
101110	1.46875	2.9375	5.875	11.75
101111	1.5	3	6	12
110000	1.53125	3.0625	6.125	12.25
110001	1.5625	3.125	6.25	12.5
110010	1.59375	3.1875	6.375	12.75
110011	1.625	3.25	6.5	13
110100	1.65625	3.3125	6.625	13.25
110101	1.6875	3.375	6.75	13.5
110110	1.71875	3.4375	6.875	13.75
110111	1.75	3.5	7	14
111000	1.78125	3.5625	7.125	14.25
111001	1.8125	3.625	7.25	14.5
111010	1.84375	3.6875	7.375	14.75
111011	1.875	3.75	7.5	15
111100	1.90625	3.8125	7.625	15.25
111101	1.9375	3.875	7.75	15.5
111110	1.96875	3.9375	7.875	15.75
111111	2	4	8	16

Table 4-3 System Frequencies from Typical 32.768-kHz or 4.194-MHz Reference

To obtain clock frequency, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell. Shaded cells represent values that exceed maximum system frequency specification (F_{sys} in **Table A-1** of **APPENDIX A ELECTRICAL CHARACTERISTICS**).

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
101011	5767	11534	23069	46137

Table 4-3 System Frequencies from Typical 32.768-kHz or 4.194-MHz Reference

To obtain clock frequency, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell. Shaded cells represent values that exceed maximum system frequency specification (F_{sys} in **Table A-1** of **APPENDIX A ELECTRICAL CHARACTERISTICS**).

Modulus	Prescaler			
Y	**[W:X] = 00**	**[W:X] = 01**	**[W:X] = 10**	**[W:X] = 11**
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

4.4 External Circuit Design

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

4.4.1 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.

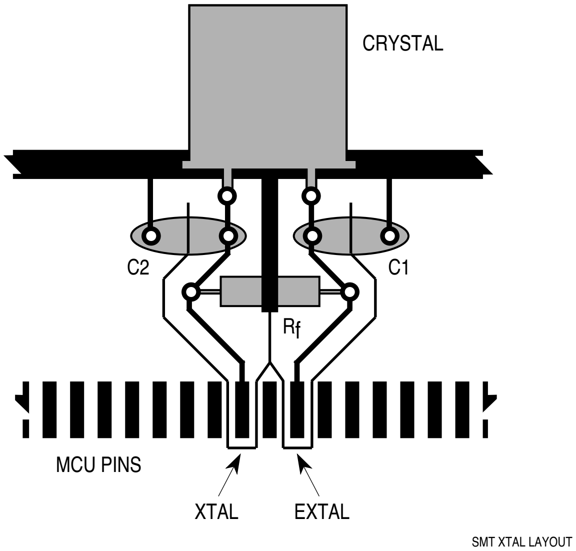


Figure 4-3 Crystal Layout Example

4.4.2 Crystal Tune-Up Procedure

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

CAUTION

> When providing samples to a crystal vendor for analysis, inform the vendor that MODCLK must be driven high during reset, enabling the PLL. The gain of the crystal driver circuitry depends on whether the PLL is enabled. If the vendor fails to enable the PLL, incorrect crystal component recommendations may result.

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.

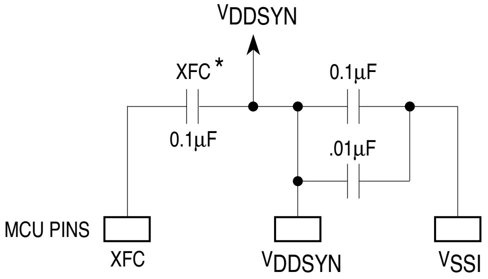
To determine the optimum value for R_S, observe the operating current (I_{DDSYN}) of the MCU as a function of R_S. The MCU should have only V_{DDSYN} powered throughout this procedure because operating current variations when the MCU is running are much greater than the current variations due to varying 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.

4.4.3 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 should consist of a low-ohm resistor (approximately 100 to 500 Ω, depending on V_{DDSYN} current and system noise) 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, typically 0.1 µF with an insulation resistance specification of 30,000 MΩ or greater, connected between the XFC and V_{DDSYN} pins. External sources of leakage from XFC may affect stability. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for limitations.

4.5 External Clock Signal Input

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.

4.6 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 M6800 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 $\overline{\mathrm{CS10}}$ field in chip select pin assignment register 1 (CSPAR1). Refer to **SECTION 7 CHIP SELECTS** for more information on the external bus clock. In addition, the operation of the external bus clock during low-power stop is described in **4.7 System Clock and Low-Power Stop**.

4.7 System Clock and Low-Power Stop

To reduce overall microcontroller power consumption to a minimum, the CPU can execute the LPSTOP instruction, which causes the SCIM to turn off the system clock to most of the MCU. During low-power stop, SCIM clock control logic and the SCIM clock signal (SCIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{\mathrm{RESET}}$ and $\overline{\mathrm{IRQ}}$ pins are clocked by SCIMCLK. The SCIM can also continue to generate the CLKOUT signal while in low-power mode.

The STSCIM (stop mode SCIM clock) and STEXT (stop mode external clock) bits in the SYNCR determine clock operation during low-power stop. **Table 4-4** summarizes the sources of SCIMCLK and CLKOUT for different combinations of clock mode, STSCIM, and STEXT during low-power stop and normal operation. The 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 4-4 Clock Control

Mode	Pins		SYNCR Bits		Clock Source	
LPSTOP	**MODCLK**	**EXTAL**	**STSCIM**	**STEXT**	**SCIMCLK**	**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

For additional information on low-power stop, refer to **3.7 Low-Power Stop Operation**. Refer to **5.8.2 LPSTOP Broadcast Cycle** for more information on the LPSTOP bus cycle.

4.8 Loss of Clock

The SCIM can detect the loss of the external clock input signal (EXTAL), regardless of whether it is used as an external clock source (MODCLK held low at reset) or as the PLL reference clock (MODCLK held high at reset). Two bits in the SYNCR determine how the SCIM responds to the loss of a clock signal. The loss-of-clock oscillator disable (LOSCD) bit enables (LOSCD = 0) or disables (LOSCD = 1) loss-of-clock detection. The reset enable (RSTEN) bit determines how the SCIM responds when it detects the loss of a clock signal. The LOSCD bit must be cleared for the RSTEN bit to have any effect.

When the RSTEN bit is set and a loss of clock is detected, the SCIM resets the MCU. If RSTEN is cleared when loss of clock is detected, an alternate clock is provided as the system clock, allowing the MCU to continue operating at a low frequency. The alternate clock is the output of an RC oscillator which is also used as the time base for the loss-of-clock detector. When edges are again detected on the crystal or external clock input, normal system clock operation resumes on the next falling edge of the alternate clock. Maximum switching time is thus determined by the alternate clock rate.

An MCU using the alternate clock as the system clock is said to be operating in limp mode. The limp mode status bit (SLIMP) in the SYNCR indicates whether the MCU is running in limp mode.

Loss of clock is recognized during low-power stop operation as well as normal operation, provided the LOSCD bit is cleared. During low-power operation (as during normal operation), when loss of clock is detected, the MCU either operates in limp mode or is reset, depending on the value of the RSTEN bit.

The following paragraphs explain loss-of-clock operation when the PLL is generating the clock signal and when an external clock is provided. In either case, the LOSCD bit must be cleared in order for loss of clock to be detected.

4.8.1 Loss of Reference Frequency

When the PLL is generating the system clock (MODCLK held high during reset), the RSTEN bit is cleared, and the SCIM detects a loss of reference signal, the alternate clock becomes the system clock. In addition, a nominal voltage is applied to the VCO. This voltage allows the PLL to ramp up quickly if the reference signal is recovered while the SCIM is operating in limp mode. If the reference signal is recovered, the PLL switches back to normal operation.

Figure 4-5 illustrates the loss and recovery of a reference signal to the PLL.

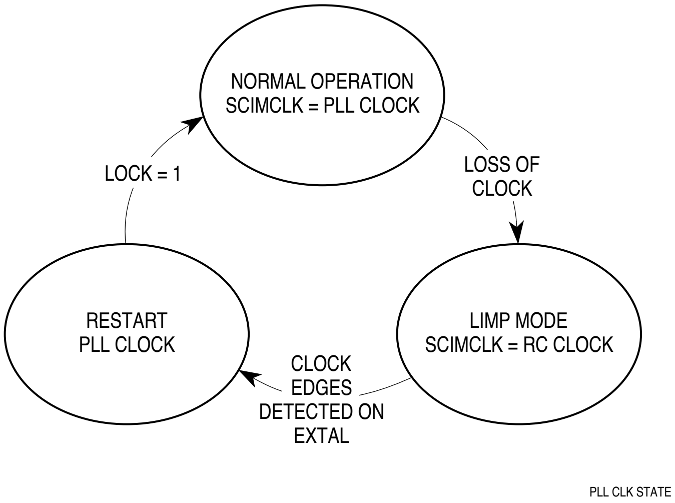


Figure 4-5 Loss of Reference Frequency

If the reference signal is lost while the VCO is ramping up, the PLL re-locks at 0 Hz, the loss-of-clock detector is triggered, and the system clock is switched back to the alternate clock.

When the RSTEN bit is set and the SCIM detects a loss of reference signal, the SCIM generates a synchronous reset. Refer to **4.8.3 Loss-of-Clock Reset**.

Notice that the LOSCD bit must be cleared for the SCIM to detect the loss of a reference signal.

4.8.2 Loss of External Clock

When the SCIM is configured for an external clock source (MODCLK held low during reset), the RSTEN bit is cleared, and a loss of clock is detected, the alternate clock becomes the system clock until the external clock input re-starts. **Figure 4-6** illustrates the loss and recovery of an external clock signal.

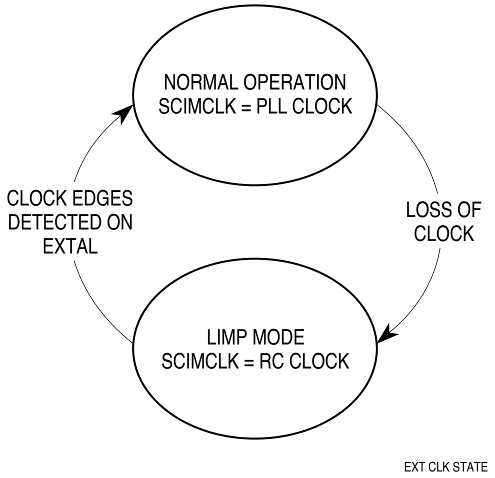


Figure 4-6 Loss of External Clock Signal

Notice that the LOSCD bit must be cleared for the SCIM to detect the loss of a clock signal.

4.8.3 Loss-of-Clock Reset

When the RSTEN bit is set and the SCIM detects the loss of a clock signal, the SCIM generates a synchronous reset. For loss of clock to be detected, the LOSCD bit must be cleared. This bit is automatically cleared at reset, ensuring that a clock signal will be present during reset: if the system clock has stopped, the SCIM detects the condition and switches to the alternate clock. This ensures that $\overline{\text{RESET}}$ will be accepted.

Refer to **SECTION 8 RESET AND SYSTEM INITIALIZATION** for more information on reset procedures.

4.9 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					8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	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 **4.3 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 **4.3 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 **4.3 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 **4.6 External Bus Clock** for additional information.

LOSCD — Loss-of-Clock Oscillator Disable

0 = Enable the loss-of-clock oscillator.
1 = Disable the loss-of-clock oscillator.

SLIMP — Limp Mode Status

0 = MCU is operating normally.
1 = Loss of reference signal — MCU operating in limp mode.

Refer to **4.8 Loss of Clock** 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 **4.8 Loss of Clock** for additional information.

STSCIM — Stop Mode SCIM Clock

0 = SCIM clock driven by the external reference signal and the VCO is turned off during low-power stop.
1 = SCIM 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 **4.7 System Clock and Low-Power Stop** for additional information.

STEXT — Stop Mode External Clock

0 = CLKOUT held low during low-power stop.
1 = CLKOUT driven from SCIM clock during low-power stop.

Refer to **4.7 System Clock and Low-Power Stop** 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 when fully expanded, or 24 address lines and 8 data lines when partially expanded. The external bus is not available when the SCIM is operating in single-chip mode.

Where not otherwise noted, this section describes the operation of the external bus interface with a fully expanded bus. For details on operation with a partially expanded bus, refer to **1.2 Bus Configuration and Reset Mode Selection** and to **5.3.5 Bus Sizing with a Partially Expanded Bus**.

NOTE

> On CPU16-based MCUs, ADDR[19:0] are normal address outputs, and ADDR[23:20] follow the output state of ADDR19. In addition, certain SCIM-based MCUs have fewer than 24 address lines. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** and to the user's manual for the particular MCU for details.

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. In fully expanded mode, 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. In partially expanded mode, only 8-bit transfers are supported. 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 with External Chip Selects** provides example system configurations for different port widths.

NOTE

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

The SCIM contains up to nine chip-select circuits that can simplify the interface to memory and peripherals. (MCUs with a reduced pin-count SCIM may contain fewer than nine chip-select circuits. Refer to the appropriate MCU user's manual for details.) 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 SCIM chip selects.

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 ($\overline{\text{AS}}$), validity of data on the bus ($\overline{\text{DS}}$), 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/$\overline{\text{W}}$). 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 $\overline{\text{AS}}$ is asserted.

5.1.2 Address Strobe

Address strobe ($\overline{\text{AS}}$) 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 $\overline{\text{AS}}$ is asserted.

5.1.4 Data Strobe

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

5.1.5 Read/Write Signal

The read/write (R/$\overline{\text{W}}$) 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 $\overline{\text{AS}}$ is asserted. R/$\overline{\text{W}}$ 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 ($\overline{\text{AS}}$) is asserted. Refer to **5.3 Dynamic Bus Sizing** for information on SIZ[1:0] encoding and on using these 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** and **5.4 Data Transfer Operations** 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 SCIMs do not include a $\overline{\text{DSACK0}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** 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.

NOTE

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

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. $\overline{\text{AVEC}}$ for external interrupt requests can also be supplied internally by chip-select logic. Refer to **SECTION 7 CHIP SELECTS** for more information.

NOTE

Some MCUs with reduced pin-count SCIMs do not include an $\overline{\text{AVEC}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** for details on handshaking with these MCUs.

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 SCIM contains nine 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 $\overline{\text{AS}}$ assertion and $\overline{\text{DSACK}}$ assertion. Although the MCU can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{\text{DSACK}}$, the MCU inserts wait cycles in clock period increments until either $\overline{\text{DSACK}}$ signal goes low.

NOTE

The SCIM bus monitor asserts $\overline{\text{BERR}}$ internally 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 $\overline{\text{BERR}}$ signal to abort the bus cycle. If $\overline{\text{BERR}}$ 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 $\overline{\text{DSACK}}$ 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.

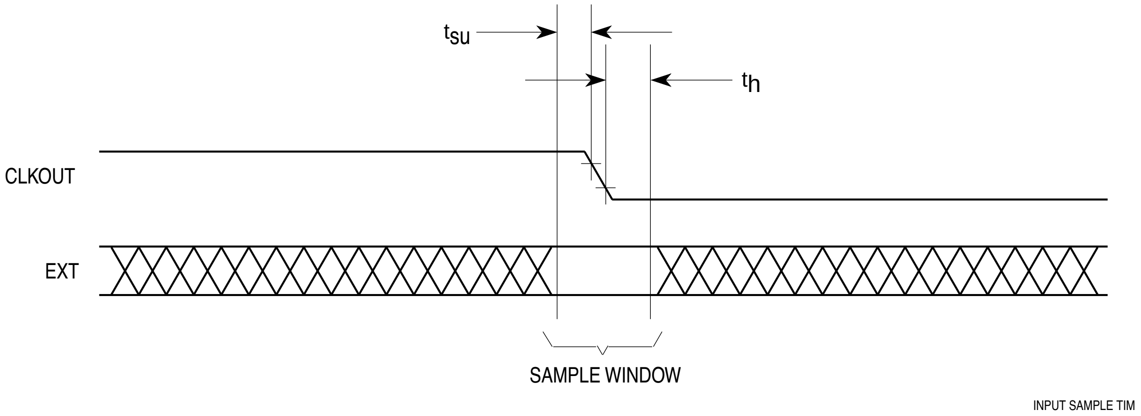


Figure 5-1 Input Sample Window

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 one and zero on the SIZ1 and SIZ0 pins, respectively. During the second cycle, the MCU drives zero and one 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 DSACK inputs. If the processor attempts to write a word to an 8-bit port, for example, the DSACK pins inform the processor that the port is only eight bits wide, and the processor initiates a second bus cycle to write the remaining byte.

Table 5-2 shows DSACK encodings.

Table 5-2 DSACK Signal Encodings

DSACK1	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 SCIMs do not contain a DSACK0 pin. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** 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 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 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 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 the target device will not use one of the bytes of the data bus.

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 re-route a byte of data internally to the operand latch, and that in some cases, 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[15:0], and an 8-bit port must be connected to DATA[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.

5.3.5 Bus Sizing with a Partially Expanded Bus

When the SCIM is operating with an 8-bit data bus, data is always transferred on DATA[15:8]. The MCU asserts the appropriate address and SIZ[1:0] signals and, during a write cycle, places the most significant byte of data on DATA[15:8]. External logic must decode the appropriate address and control lines, place the data on DATA[15:8] during a read cycle or read the data from DATA[15:8] during a write cycle, and assert $\overline{\text{DSACK0}}$, indicating an 8-bit port.

NOTE

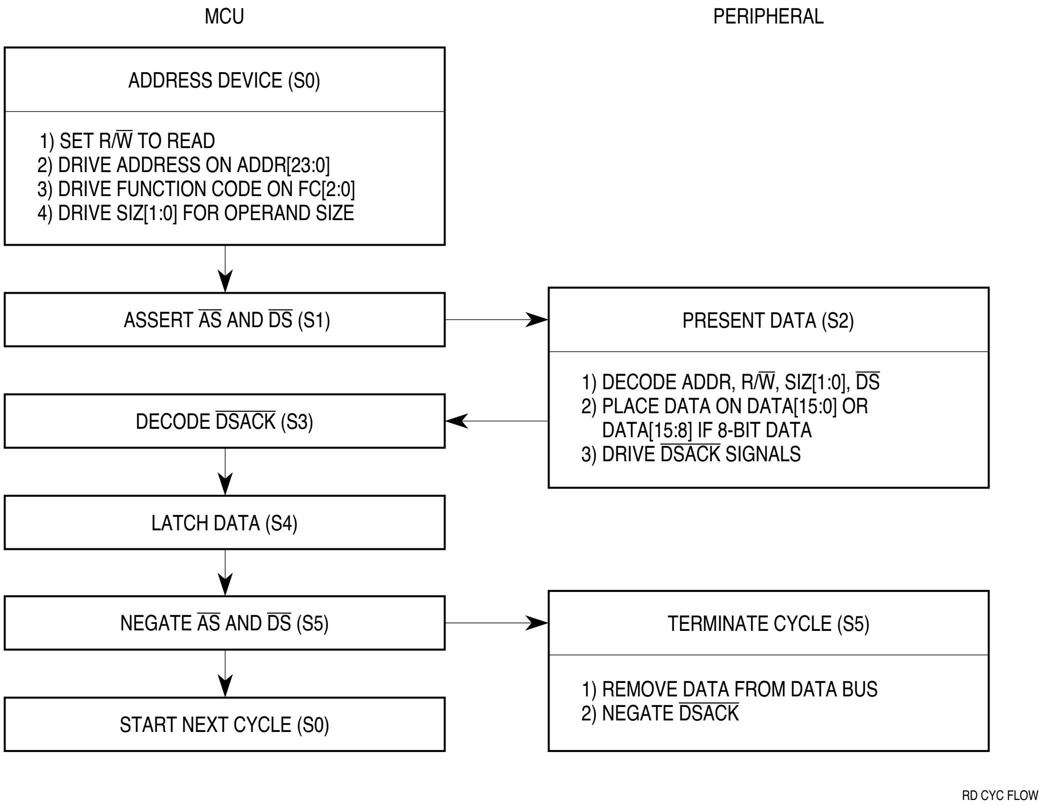
Some MCUs with reduced pin-count SCIMs do not include a $\overline{\text{DSACK0}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** for details on handshaking with these MCUs.

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 indivisible read-modify-write operation (not supported on CPU16-based MCUs), 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 flowchart of a word read cycle.



MCU
PERIPHERAL
ADDRESS DEVICE (S0)
1) SET R/$\overline{\text{W}}$ TO READ
2) DRIVE ADDRESS ON ADDR[23:0]
3) DRIVE FUNCTION CODE ON FC[2:0]
4) DRIVE SIZ[1:0] FOR OPERAND SIZE
ASSERT $\overline{\text{AS}}$ AND $\overline{\text{DS}}$ (S1)
PRESENT DATA (S2)
1) DECODE ADDR, R/$\overline{\text{W}}$, SIZ[1:0], $\overline{\text{DS}}$
2) PLACE DATA ON DATA[15:0] OR DATA[15:8] IF 8-BIT DATA
3) DRIVE $\overline{\text{DSACK}}$ SIGNALS
DECODE $\overline{\text{DSACK}}$ (S3)
LATCH DATA (S4)
NEGATE $\overline{\text{AS}}$ AND $\overline{\text{DS}}$ (S5)
TERMINATE CYCLE (S5)
1) REMOVE DATA FROM DATA BUS
2) NEGATE $\overline{\text{DSACK}}$
START NEXT CYCLE (S0)
RD CYC FLOW

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/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 AS, indicating that the address on the address bus is valid. The MCU also asserts DS, signaling the peripheral to place data on the bus.

State 2 (S2) — External logic decodes ADDR[23:0], FC[2:0] as needed, R/W, SIZ[1:0], and DS. If the addressed location is a 16-bit port, the responding device places data on the appropriate byte or bytes of the data bus and asserts DSACK1. If the addressed location is an 8-bit port (or if the MCU is operating with a partially-expanded data bus), the responding device places data on DATA[15:8] and asserts DSACK0.

To guarantee that data is latched in the minimum cycle time, at least one 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.) To guarantee that wait states are inserted, both DSACK1 and 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 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 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 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 DSACK signals is detected.

State 4 (S4) — If a change in the 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 DSACK signals is detected.

State 5 (S5) — The MCU negates AS and DS, but holds the address valid to provide address hold time for memory systems. R/W, SIZ[1:0], and FC[2:0] also remain valid throughout S5. The external device must maintain data and assert the DSACK signals until it detects the negation of either AS or DS. The external device must remove the data and negate DSACK within approximately one clock period after sensing the negation of AS or 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.

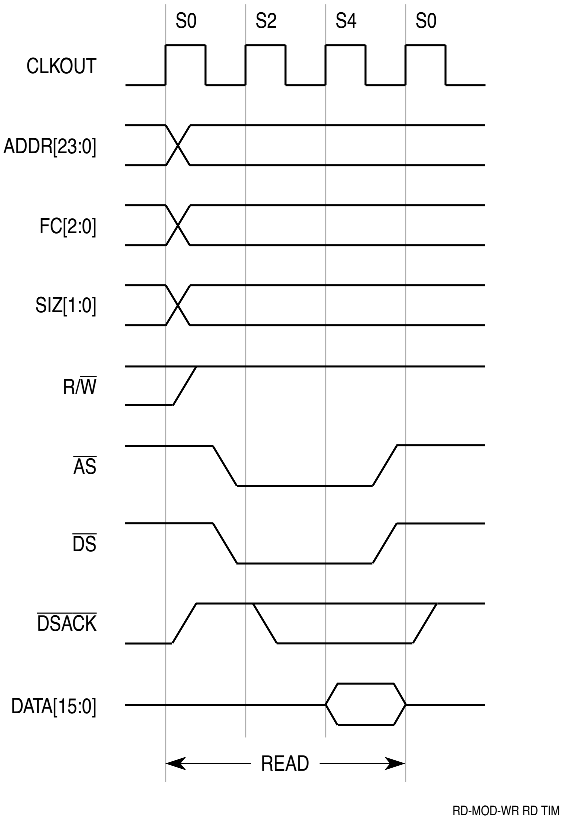


Figure 5-3 Read Cycle Timing Diagram

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 flowchart of a write cycle for a word transfer.

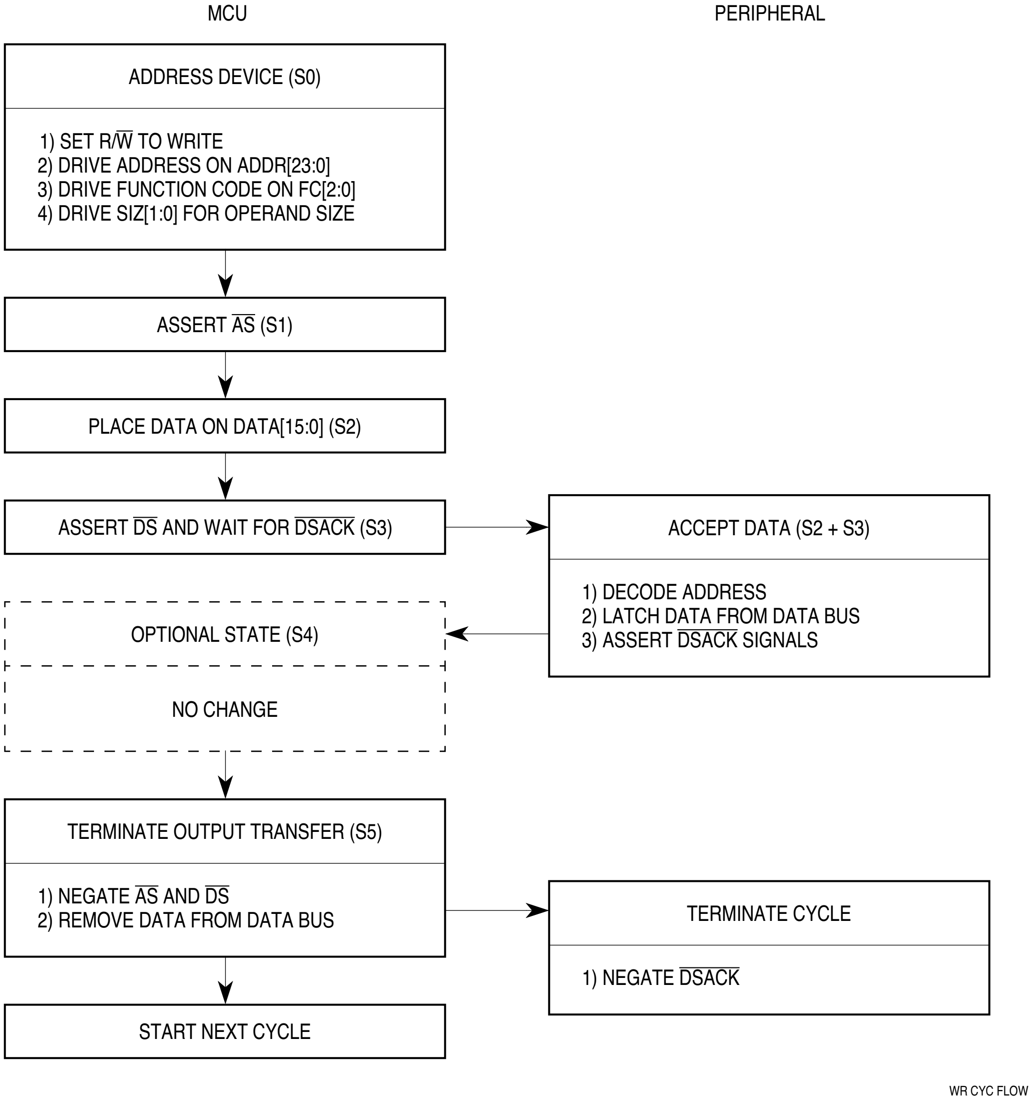

MCU
PERIPHERAL
ADDRESS DEVICE (S0)
1) SET R/$\overline{W}$ TO WRITE
2) DRIVE ADDRESS ON ADDR[23:0]
3) DRIVE FUNCTION CODE ON FC[2:0]
4) DRIVE SIZ[1:0] FOR OPERAND SIZE
ASSERT $\overline{AS}$ (S1)
PLACE DATA ON DATA[15:0] (S2)
ASSERT $\overline{DS}$ AND WAIT FOR $\overline{DSACK}$ (S3)
ACCEPT DATA (S2 + S3)
1) DECODE ADDRESS
2) LATCH DATA FROM DATA BUS
3) ASSERT $\overline{DSACK}$ SIGNALS
OPTIONAL STATE (S4)
NO CHANGE
TERMINATE OUTPUT TRANSFER (S5)
1) NEGATE $\overline{AS}$ AND $\overline{DS}$
2) REMOVE DATA FROM DATA BUS
TERMINATE CYCLE
1) NEGATE $\overline{DSACK}$
START NEXT CYCLE
WR CYC FLOW

Figure 5-4 Write Cycle Flowchart

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[2:0] if necessary, R/$\overline{W}$, SIZ[1:0], and $\overline{AS}$. If the addressed location is a 16-bit port, the responding device retrieves the data from the appropriate byte or bytes of the data bus and asserts $\overline{DSACK1}$. If the addressed location is an 8-bit port (or if the MCU is operating with a partially-expanded data bus), the responding device retrieves the data on DATA[15:8] and asserts $\overline{\text{DSACK0}}$.

To guarantee that data is latched in the minimum cycle time, the MCU must recognize a change in at least one $\overline{\text{DSACK}}$ signal by the end of S2 (that is, the $\overline{\text{DSACK}}$ signal must meet the input setup and hold time requirements). To guarantee that wait states are inserted, both $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ must remain negated throughout the asynchronous input setup and hold times at the end of S2.

State 3 (S3) — The MCU asserts $\overline{\text{DS}}$ 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 $\overline{\text{DSACK}}$ signals has already been recognized, S4 elapses, and the cycle terminates during S5. If neither $\overline{\text{DSACK}}$ 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 $\overline{\text{DSACK}}$ 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 $\overline{\text{DSACK}}$ signals is detected.

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

State 5 (S5) — The MCU negates $\overline{\text{AS}}$ and $\overline{\text{DS}}$, but holds the address and data 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 assert the $\overline{\text{DSACK}}$ signals until it detects the negation of either $\overline{\text{AS}}$ or $\overline{\text{DS}}$. It must negate 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 5-5 is a timing diagram of a write cycle.

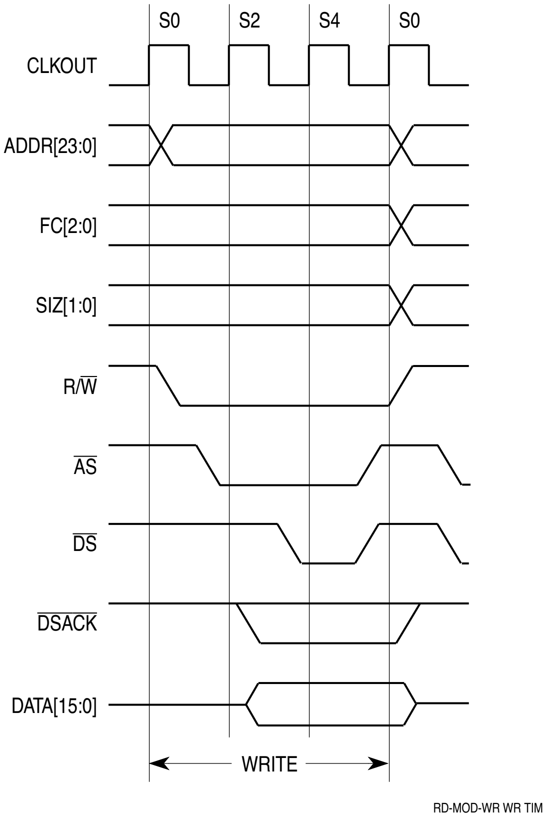


Figure 5-5 Write Cycle Timing Diagram

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.

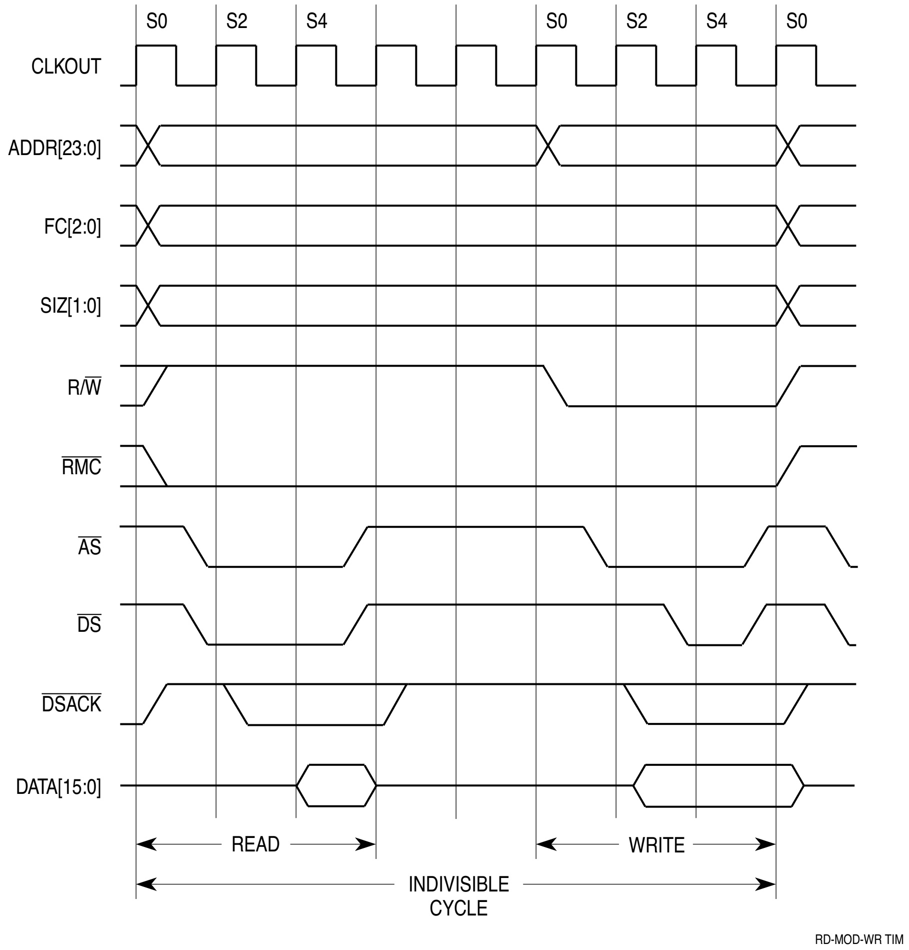


Figure 5-6 Read-Modify-Write Timing

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 $\overline{\text{RMC}}$ to identify a read-modify-write cycle. When finished reading in S5, the MCU holds the address, R/$\overline{\text{W}}$, 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/$\overline{\text{W}}$ 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/$\overline{\text{W}}$ 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.

NOTE

> The discussion in this section assumes that both $\overline{\text{DSACK}}$ pins are present on the chip. For MCUs without a $\overline{\text{DSACK0}}$ pin, refer to **SECTION 10 REDUCED PIN-COUNT SCIM**.

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.

Operand	**Byte Order**			
	31 24	23 16	15 8	7 0
Long Word	OP0	OP1	OP2	OP3
Three Byte		OP0	OP1	OP2
Word			OP0	OP1
Byte				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 Cycles		Write Cycles		Next Cycle
					DATA [15:8]	DATA [7:0]	DATA [15:8]	DATA [7:0]	
1	Byte to 8-Bit Port (Even/ Odd)	01	X	10	OP0	—	OP0	(OP0)	—
2	Byte to 16-Bit Port (Even)[1]	01	0	01	OP0	—	OP0	(OP0)	—
3	Byte to 16-Bit Port (Odd)[1]	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)[2]	10	1	10	OP0	—	OP0	(OP0)	1
6	Word to 16-Bit Port (Aligned)[1]	10	0	01	OP0	OP1	OP0	OP1	—
7	Word to 16-Bit Port (Misaligned)[1,2]	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)[2,3]	10	1	10	OP0	—	OP0	(OP0)	12
10	Long Word to 16-Bit Port (Aligned)[1]	00	0	01	OP0	OP1	OP0	OP1	6
11	Long Word to 16-Bit Port (Misaligned)[1,2,3]	10	1	01	—	OP0	(OP0)	OP0	2
12	3 Byte to 8-Bit Port (Aligned)[3]	11	0	10	OP0	—	OP0	(OP1)	5
13	3 Byte to 8-Bit Port (Misaligned) [4]	11	1	10	OP0	—	OP0	(OP0)	4

NOTES:

1. Transfers involving 16-bit ports are supported only in fully-expanded (16-bit data bus) mode.
2. The CPU32 does not support misaligned transfers.
3. The CPU16 treats misaligned long-word transfers as two misaligned-word transfers
4. Three-byte transfer cases occur only as a result of a long word to byte transfer.

5.5.1 Byte Operand to 8-Bit Port

For an 8-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.

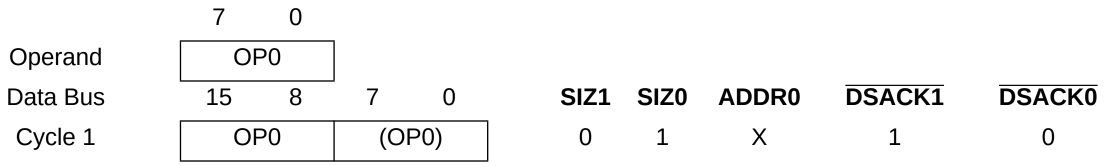


Figure 5-8 Byte Operand to 8-Bit Port

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.

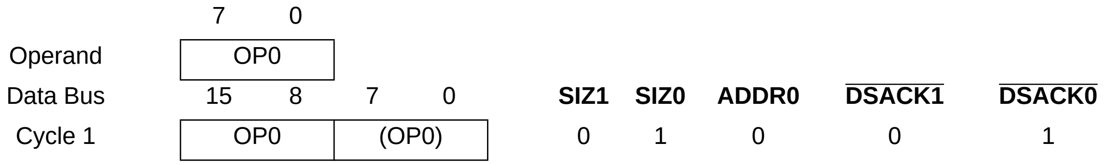


Figure 5-9 Byte Operand to 16-Bit Port, Even (ADDR0 = 0)

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 with External Chip Selects** 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.

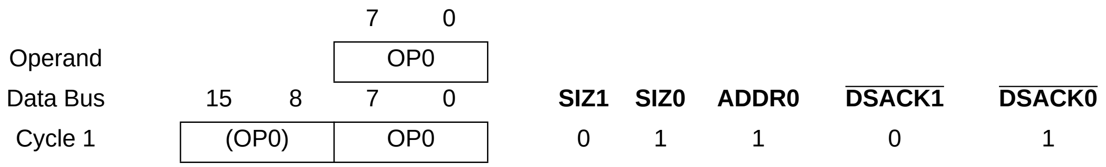


Figure 5-10 Byte Operand to 16-Bit Port, Odd (ADDR0 = 1)

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 with External Chip Selects** 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[15:0] 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					
Operand		OP0					
	OP1						
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

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[15:0] 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					
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	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.

	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)	OP0	1	0	1	0	1
Cycle 2	OP1	(OP1)	0	1	0	0	1

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
	OP2	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, involving an 8-bit port.

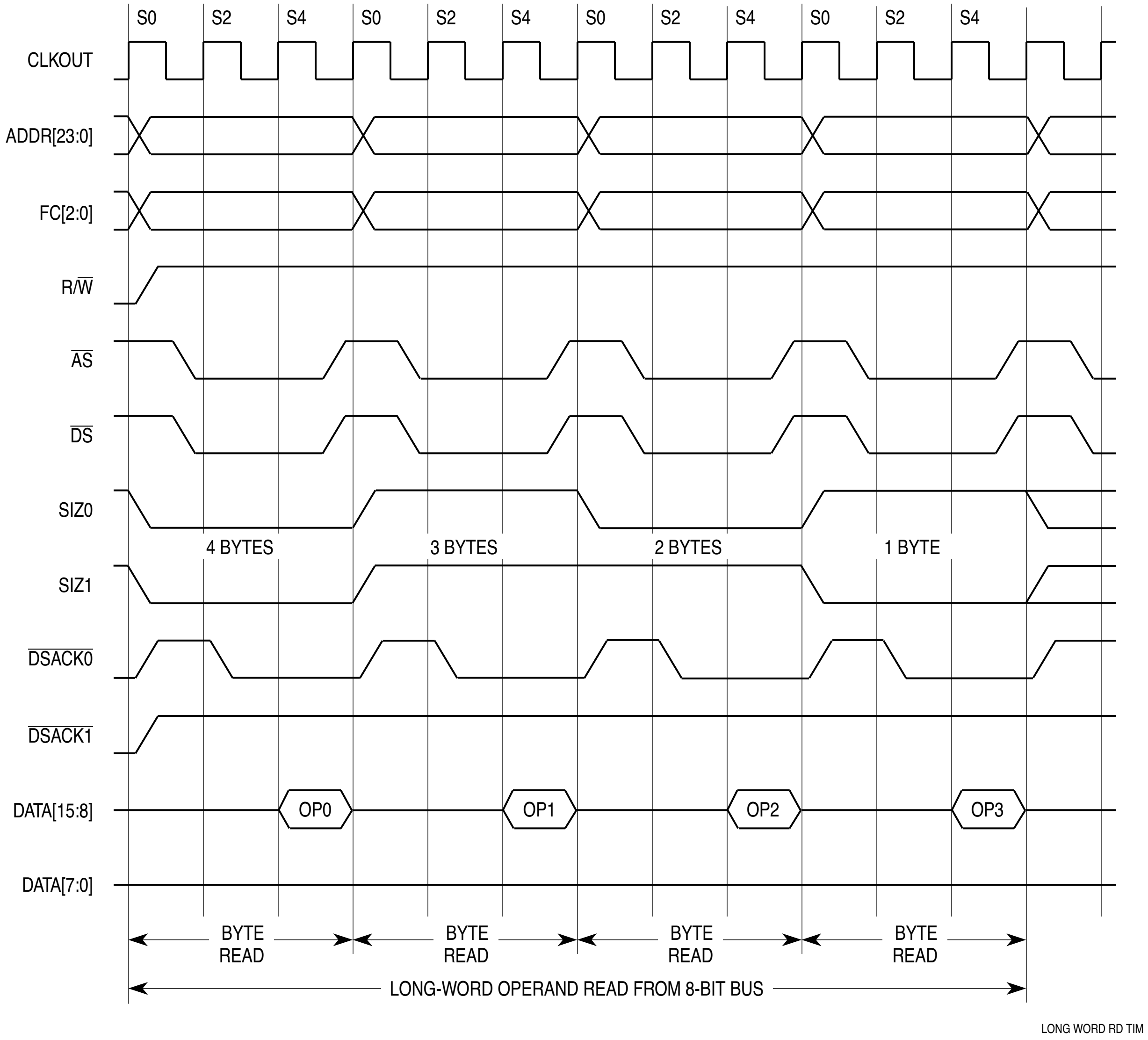


Figure 5-16 Timing of a Long-Word Read of an 8-Bit Port

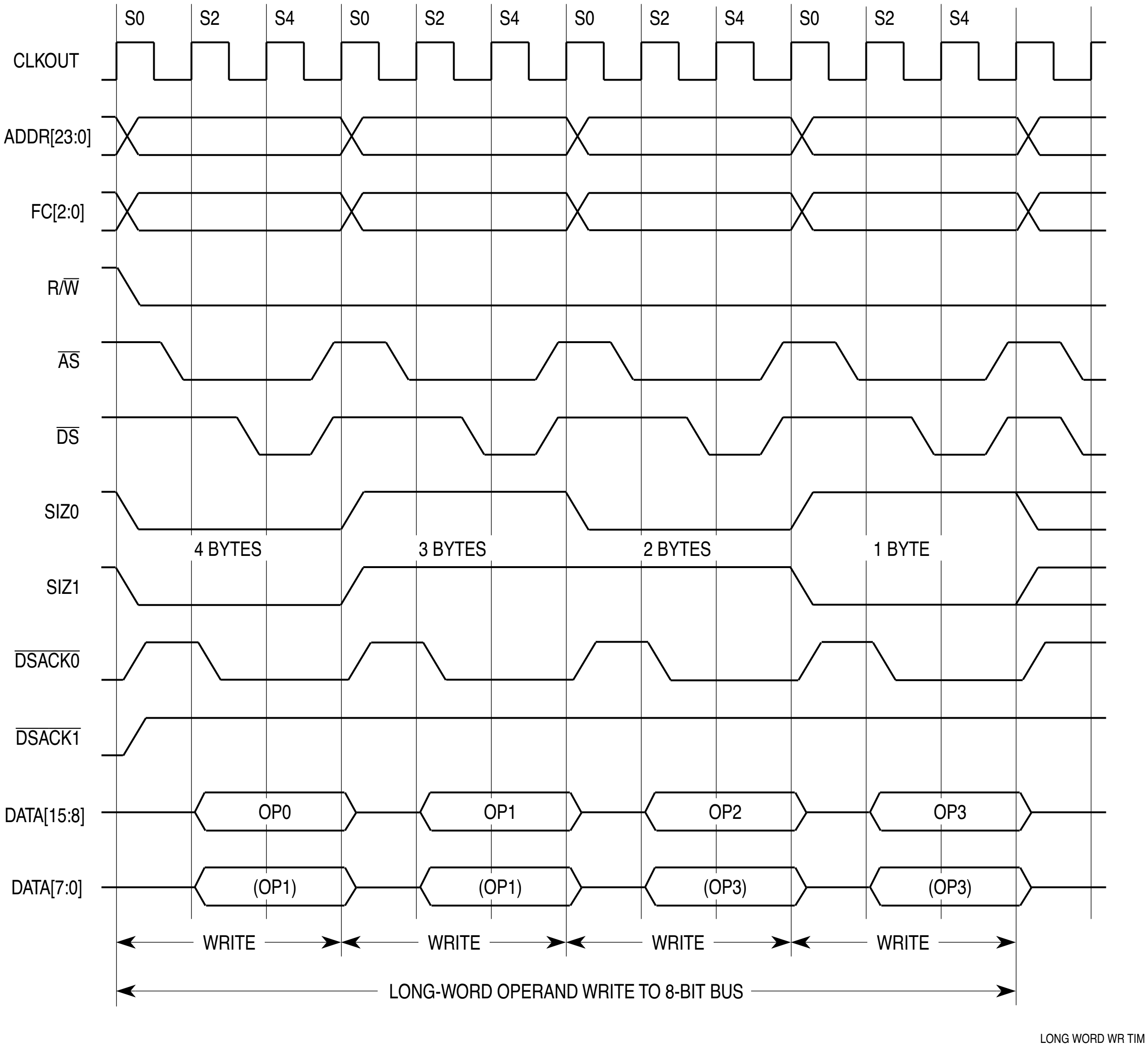


Figure 5-17 Timing of a Long-Word Write to an 8-Bit Port

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/W̄ 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	OP1	0	0	0	0	1
Cycle 2	OP2	OP2	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 one and SIZ0 to zero, 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 one and SIZ0 to zero, 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.

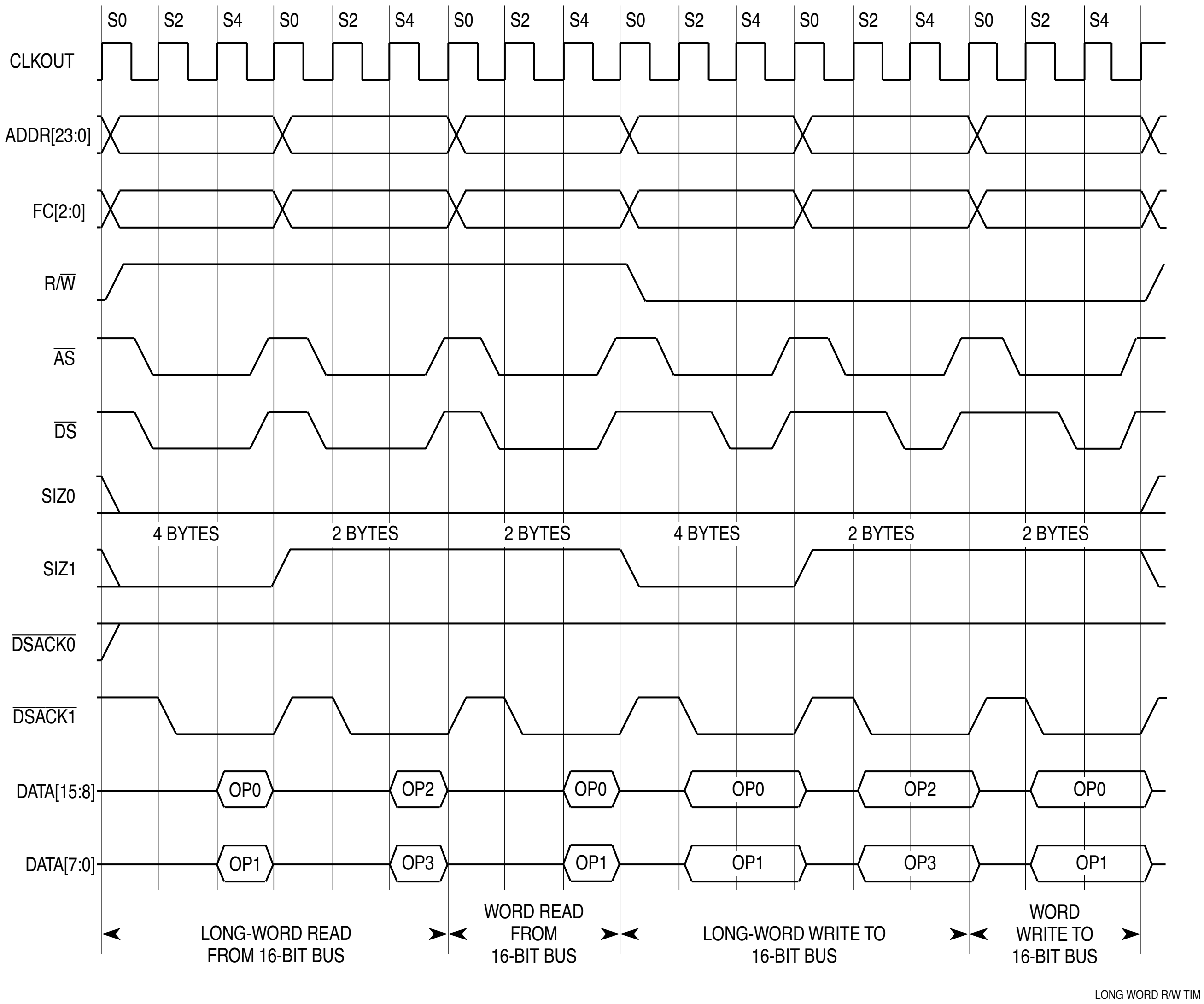


Figure 5-20 Timing of Long-Word Read or Write, 16-Bit Port

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

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

The SCIM 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 with External Chip Selects

This section provides examples of interfacing the MCU, with a fully-expanded external bus (and the $\overline{\text{DSACK0}}$ pin present on the chip), 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 SCIM chip selects for most applications. For configurations using SCIM chip selects, refer to **7.12 Interfacing Example with SCIM Chip Selects**.

5.7.1 Connecting an 8-Bit Memory 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.

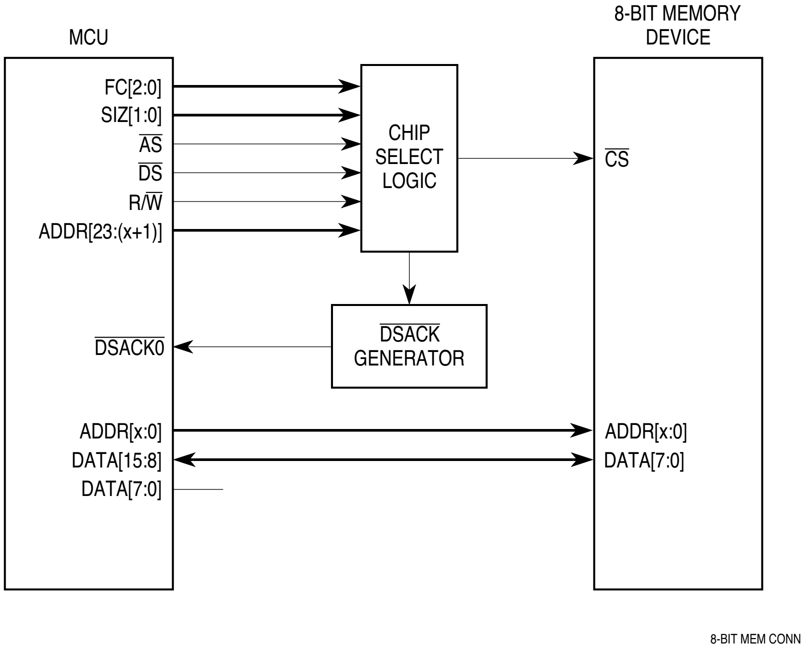


Figure 5-22 Connecting an 8-Bit Memory Device

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 16 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 **7.12 Interfacing Example with SCIM Chip Selects**.

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

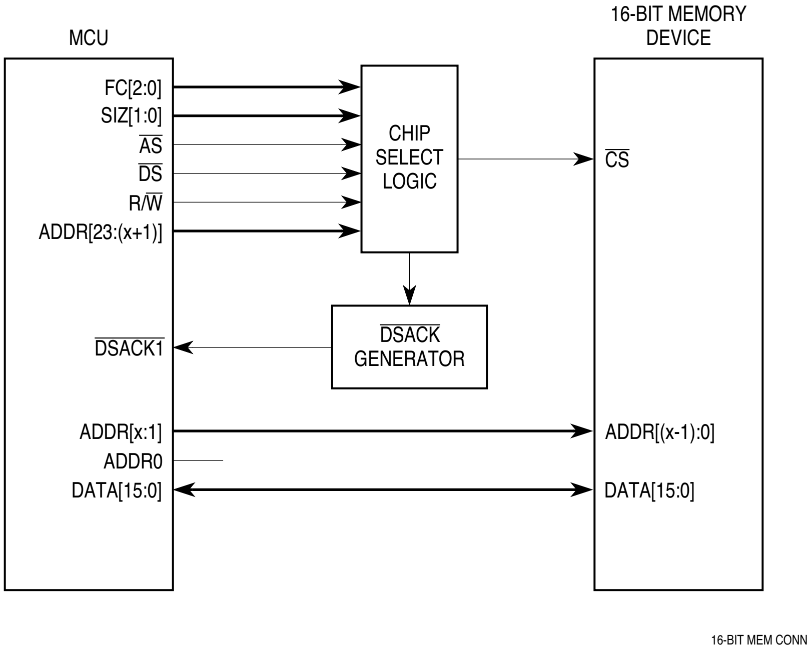


Figure 5-23 Connecting a 16-Bit Memory Device

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.

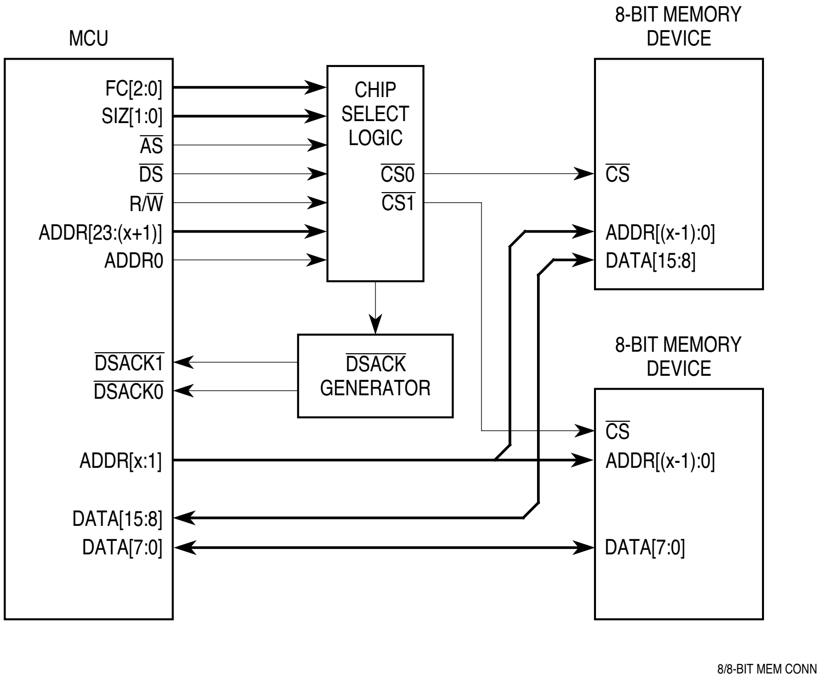


Figure 5-24 Connecting Two 8-Bit Memory Devices

The SCIM 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 **7.12 Interfacing Example with SCIM 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.

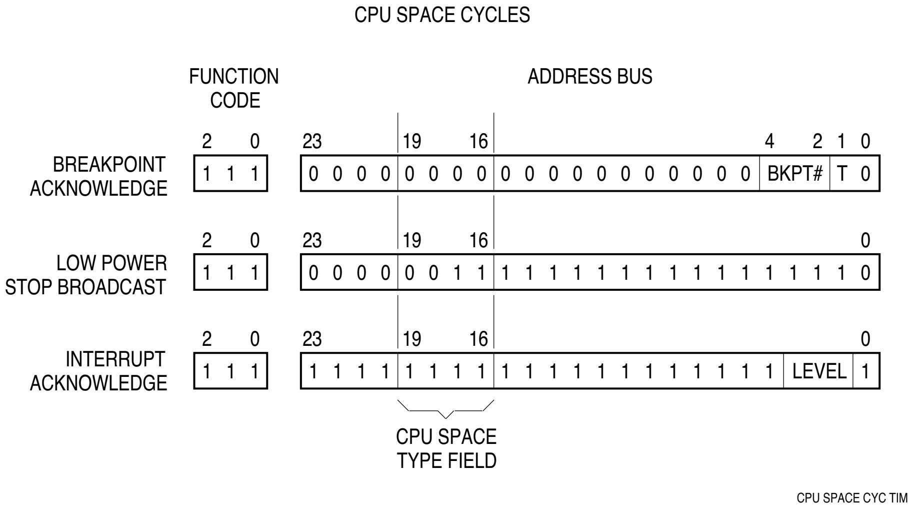


Figure 5-25 CPU Space Address Encoding

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 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 8-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

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. 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.

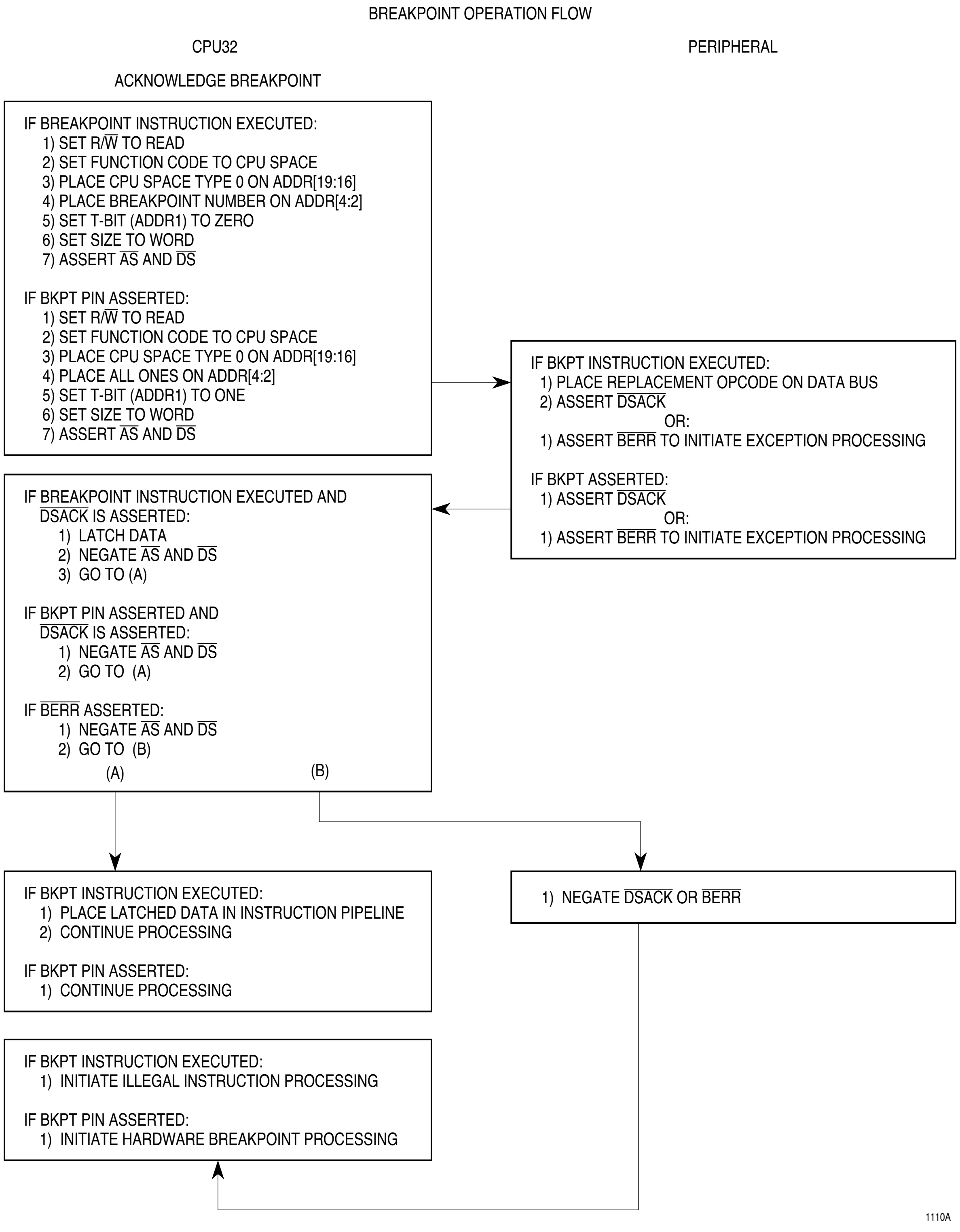

BREAKPOINT OPERATION FLOW
CPU32
PERIPHERAL
ACKNOWLEDGE BREAKPOINT
IF BREAKPOINT INSTRUCTION EXECUTED:
1) SET R/$\overline{W}$ TO READ
2) SET FUNCTION CODE TO CPU SPACE
3) PLACE CPU SPACE TYPE 0 ON ADDR[19:16]
4) PLACE BREAKPOINT NUMBER ON ADDR[4:2]
5) SET T-BIT (ADDR1) TO ZERO
6) SET SIZE TO WORD
7) ASSERT $\overline{AS}$ AND $\overline{DS}$
IF BKPT PIN ASSERTED:
1) SET R/$\overline{W}$ TO READ
2) SET FUNCTION CODE TO CPU SPACE
3) PLACE CPU SPACE TYPE 0 ON ADDR[19:16]
4) PLACE ALL ONES ON ADDR[4:2]
5) SET T-BIT (ADDR1) TO ONE
6) SET SIZE TO WORD
7) ASSERT $\overline{AS}$ AND $\overline{DS}$
IF BKPT INSTRUCTION EXECUTED:
1) PLACE REPLACEMENT OPCODE ON DATA BUS
2) ASSERT $\overline{DSACK}$
OR:
1) ASSERT $\overline{BERR}$ TO INITIATE EXCEPTION PROCESSING
IF BKPT ASSERTED:
1) ASSERT $\overline{DSACK}$
OR:
1) ASSERT $\overline{BERR}$ TO INITIATE EXCEPTION PROCESSING
IF BREAKPOINT INSTRUCTION EXECUTED AND
$\overline{DSACK}$ IS ASSERTED:
1) LATCH DATA
2) NEGATE $\overline{AS}$ AND $\overline{DS}$
3) GO TO (A)
IF BKPT PIN ASSERTED AND
$\overline{DSACK}$ IS ASSERTED:
1) NEGATE $\overline{AS}$ AND $\overline{DS}$
2) GO TO (A)
IF $\overline{BERR}$ ASSERTED:
1) NEGATE $\overline{AS}$ AND $\overline{DS}$
2) GO TO (B)
(A)
(B)
IF BKPT INSTRUCTION EXECUTED:
1) PLACE LATCHED DATA IN INSTRUCTION PIPELINE
2) CONTINUE PROCESSING
IF BKPT PIN ASSERTED:
1) CONTINUE PROCESSING
1) NEGATE $\overline{DSACK}$ OR $\overline{BERR}$
IF BKPT INSTRUCTION EXECUTED:
1) INITIATE ILLEGAL INSTRUCTION PROCESSING
IF BKPT PIN ASSERTED:
1) INITIATE HARDWARE BREAKPOINT PROCESSING
1110A

Figure 5-26 CPU32 Breakpoint Operation Flow

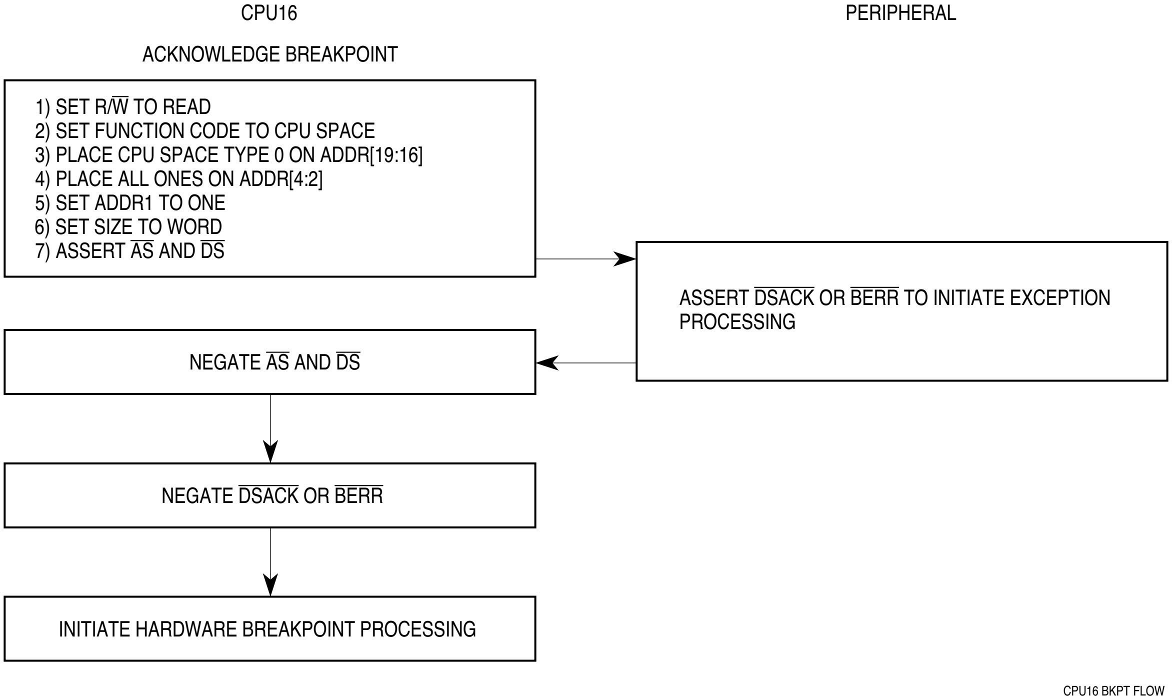

CPU16
PERIPHERAL
ACKNOWLEDGE BREAKPOINT
1) SET R/W̄ TO READ
2) SET FUNCTION CODE TO CPU SPACE
3) PLACE CPU SPACE TYPE 0 ON ADDR[19:16]
4) PLACE ALL ONES ON ADDR[4:2]
5) SET ADDR1 TO ONE
6) SET SIZE TO WORD
7) ASSERT A̅S̅ AND D̅S̅
ASSERT D̅S̅A̅C̅K̅ OR B̅E̅R̅R̅ TO INITIATE EXCEPTION PROCESSING
NEGATE A̅S̅ AND D̅S̅
NEGATE D̅S̅A̅C̅K̅ OR B̅E̅R̅R̅
INITIATE HARDWARE BREAKPOINT PROCESSING
CPU16 BKPT FLOW

Figure 5-27 CPU16 Breakpoint Operation Flow

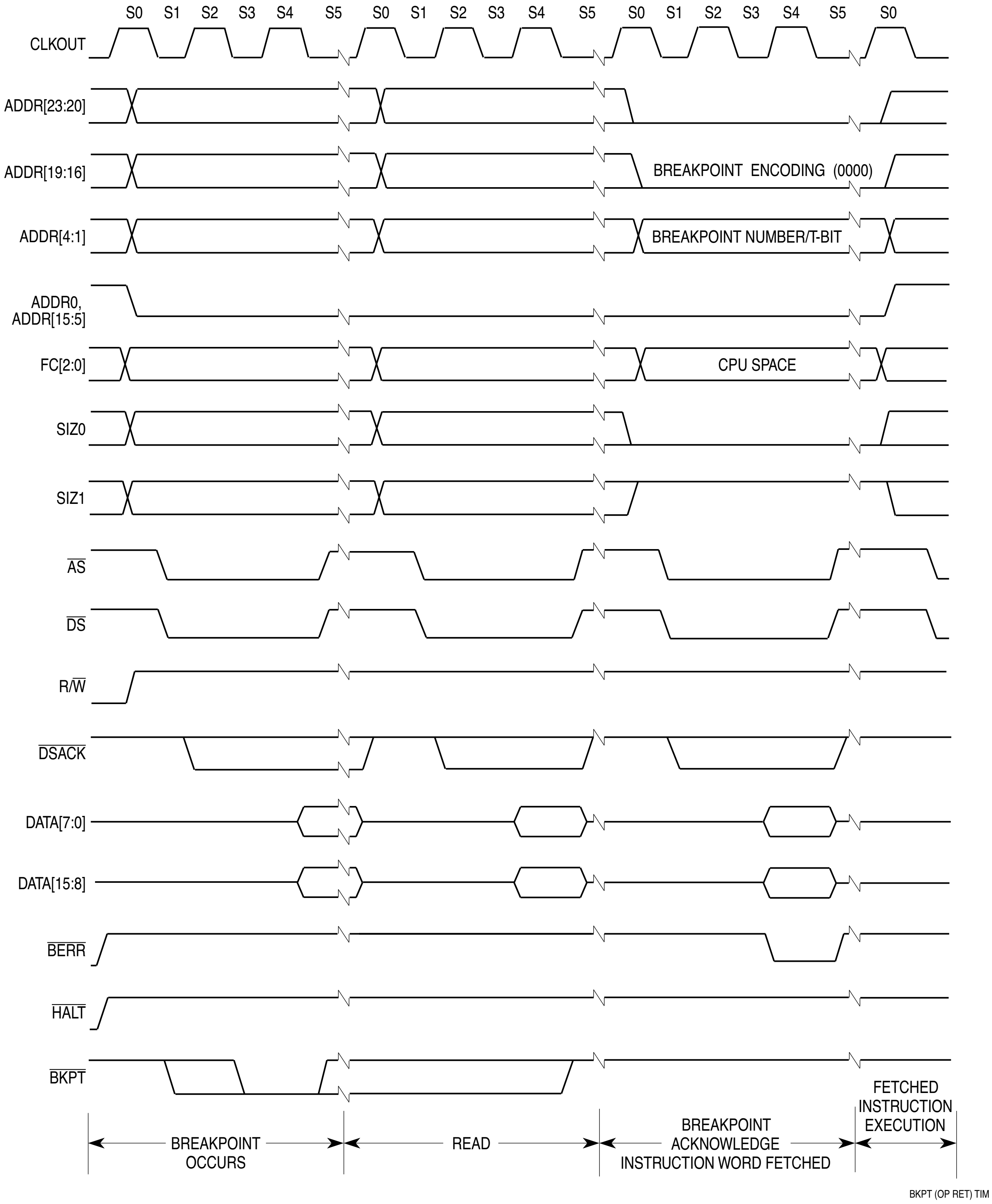


Figure 5-28 Breakpoint Acknowledge Cycle Timing — Opcode Returned (CPU32 Only)

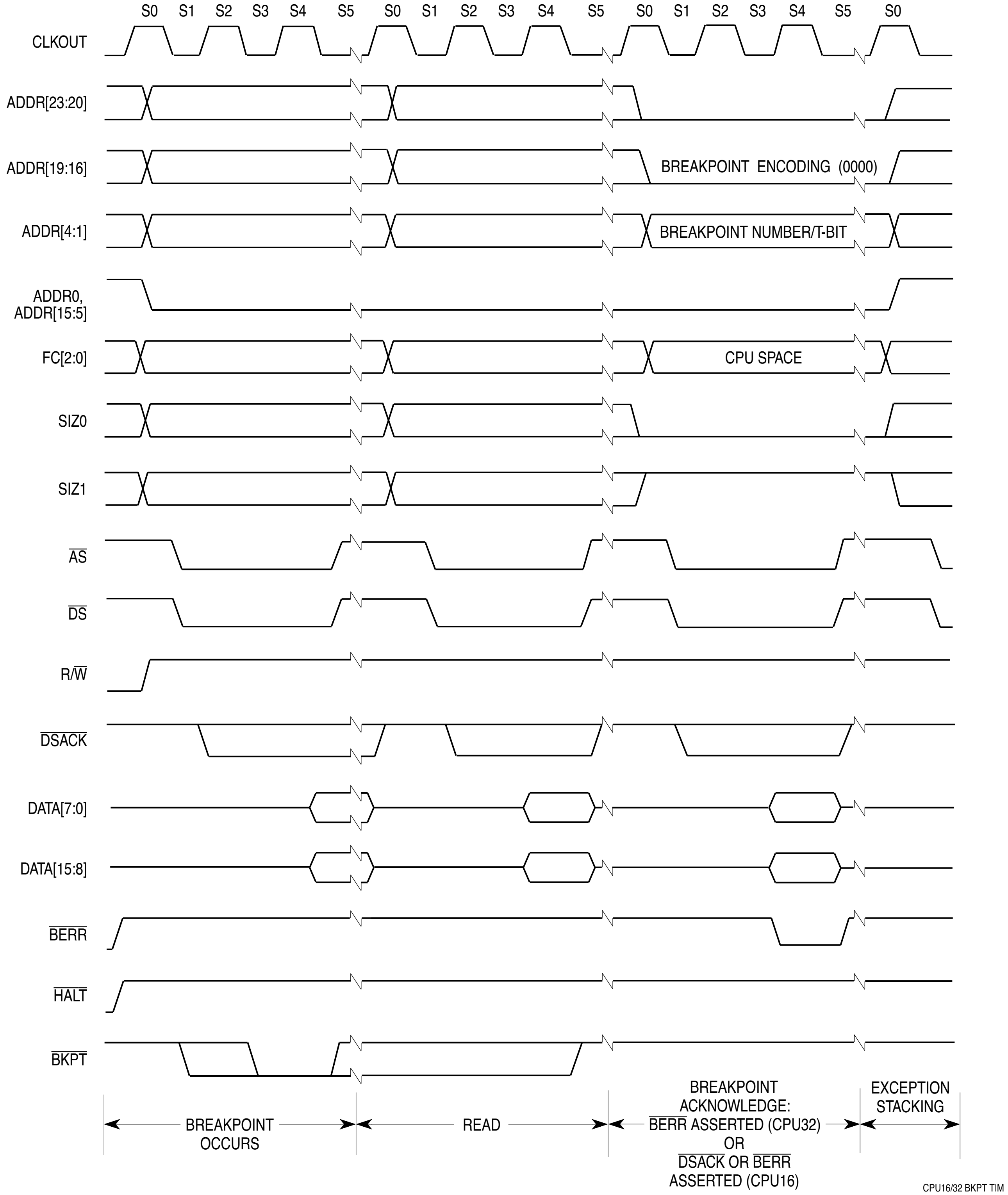


Figure 5-29 Breakpoint Acknowledge Cycle Timing — Exception Signaled

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 SCIM provides an internally generated 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 SCIM brings the MCU out of low-power mode when either an interrupt of higher priority (including an interrupt from the periodic interrupt timer) 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 DSACK[1:0] signals or the 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 BERR internally, causing a bus error exception to be taken. Bus cycles can also be terminated by assertion of the external BERR or 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 5-5**, which indicates the results of each type of bus cycle termination.

Normal Termination

DSACK is asserted; BERR and HALT remain negated (case 1).

Halt Termination

HALT is asserted at the same time, or before DSACK, and BERR remains negated (case 2).

Bus Error Termination

BERR is asserted in lieu of, at the same time as, or before DSACK (case 3), or after DSACK (case 4), and HALT remains negated; BERR is negated at the same time or after 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. If an intervening instruction causes a branch, or if a task switch occurs, 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.

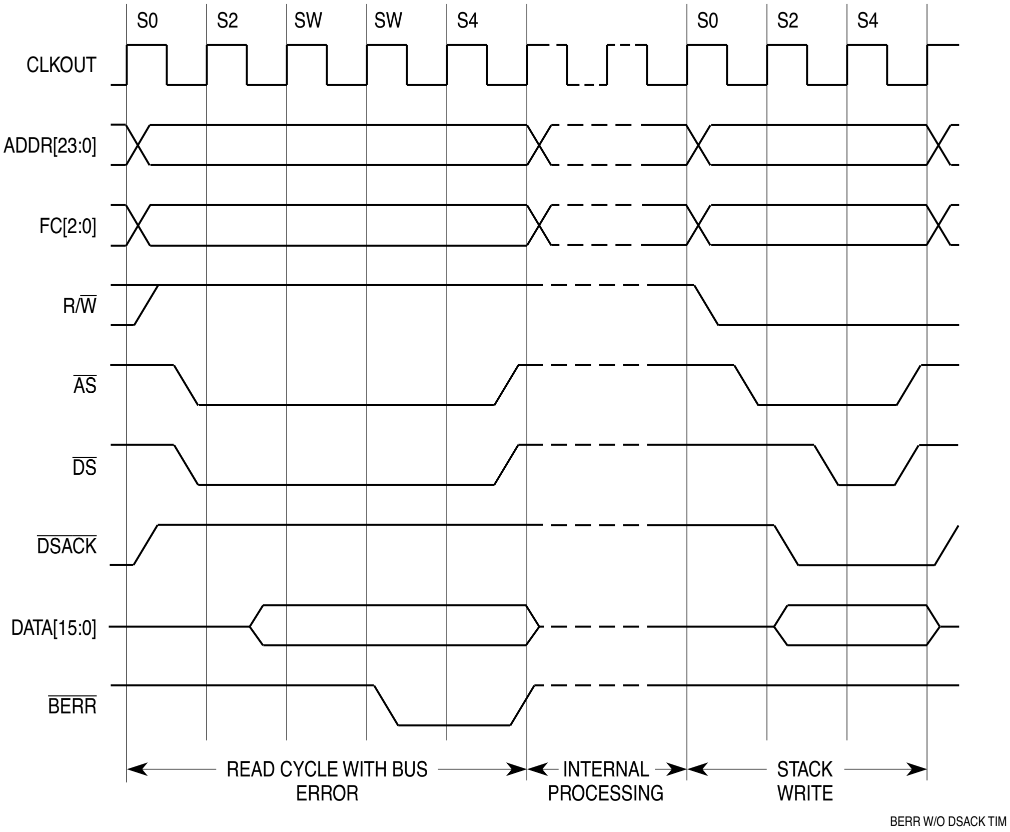


Figure 5-31 Bus Error Without $\overline{\text{DSACK}}$

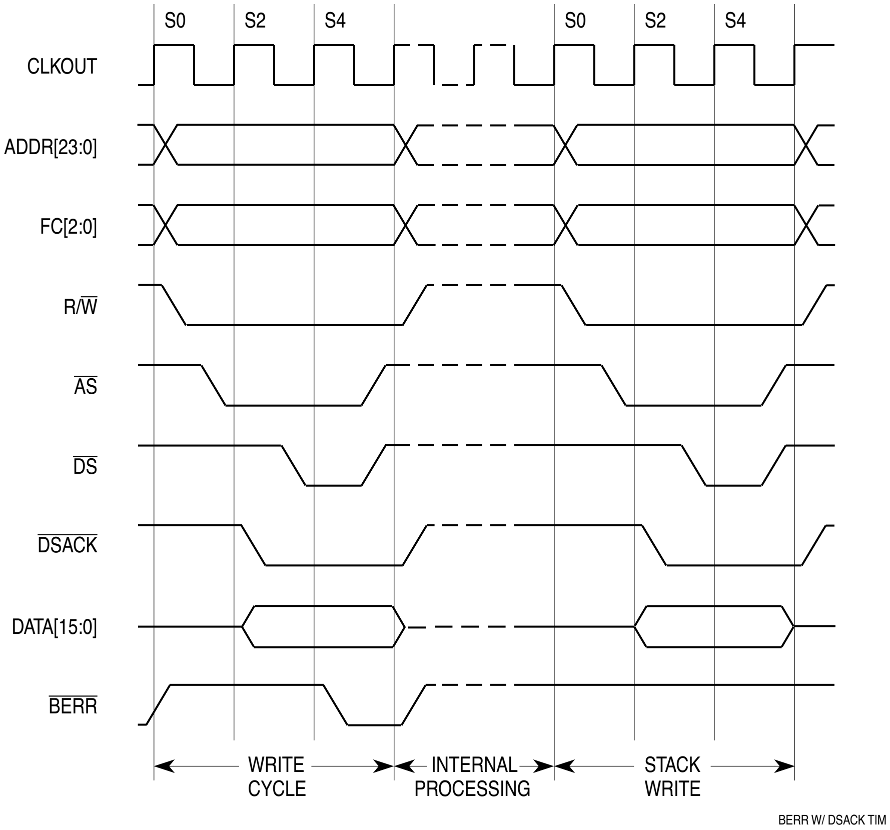


Figure 5-32 Late Bus Error with DSACK

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 BERR signal, the MCU halts and drives the 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 BERR and HALT during a bus cycle, the MCU enters the retry sequence, shown in **Figure 5-33**.

NOTE

> On CPU16-based MCUs, assertion of BERR and 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 AS and DS signals in their inactive state, and does not begin another bus cycle until the BERR and 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 BERR signal should be negated before S2 of the read cycle to ensure correct operation of the retried cycle.

If BR, BERR, and HALT are all asserted on the same cycle, the EBI enters the rerun sequence but first relinquishes the bus to an external master. Once the external master returns the bus and negates BERR and 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; RMC remains asserted during the entire retry sequence. The MCU will not relinquish the bus while 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 BERR and BR only (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.)

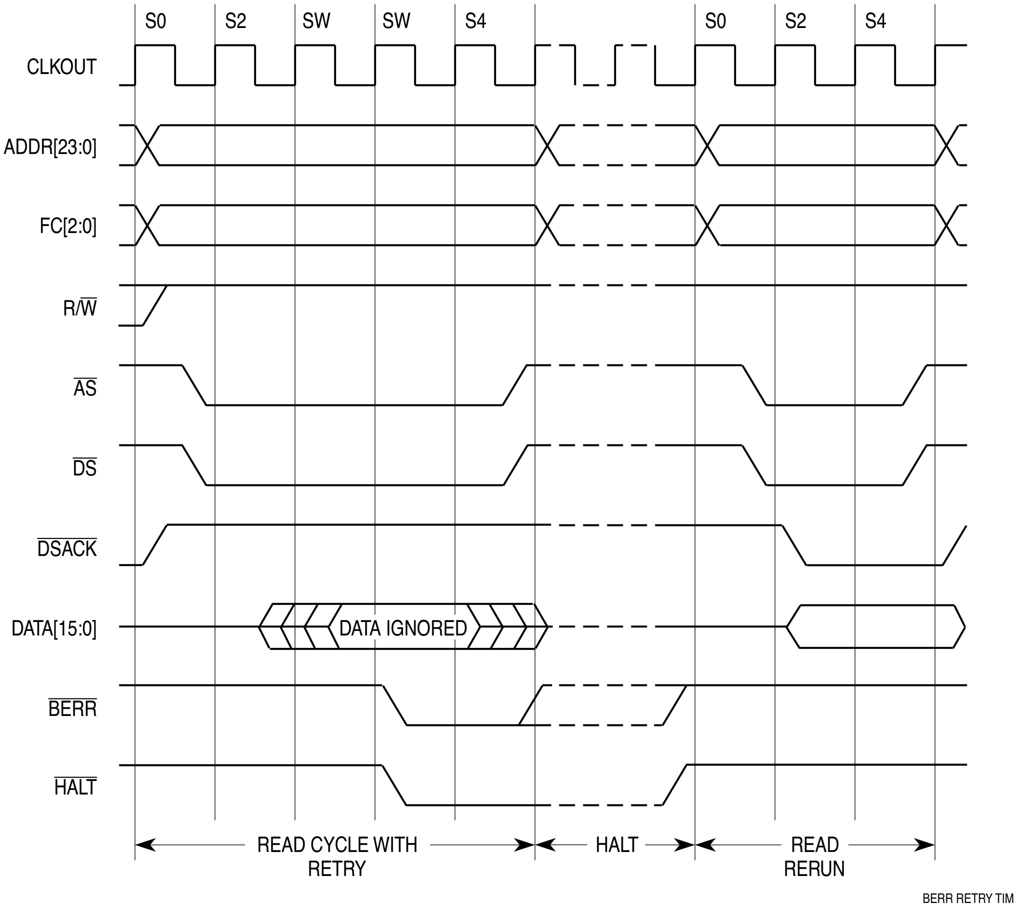


Figure 5-33 Retry Sequence

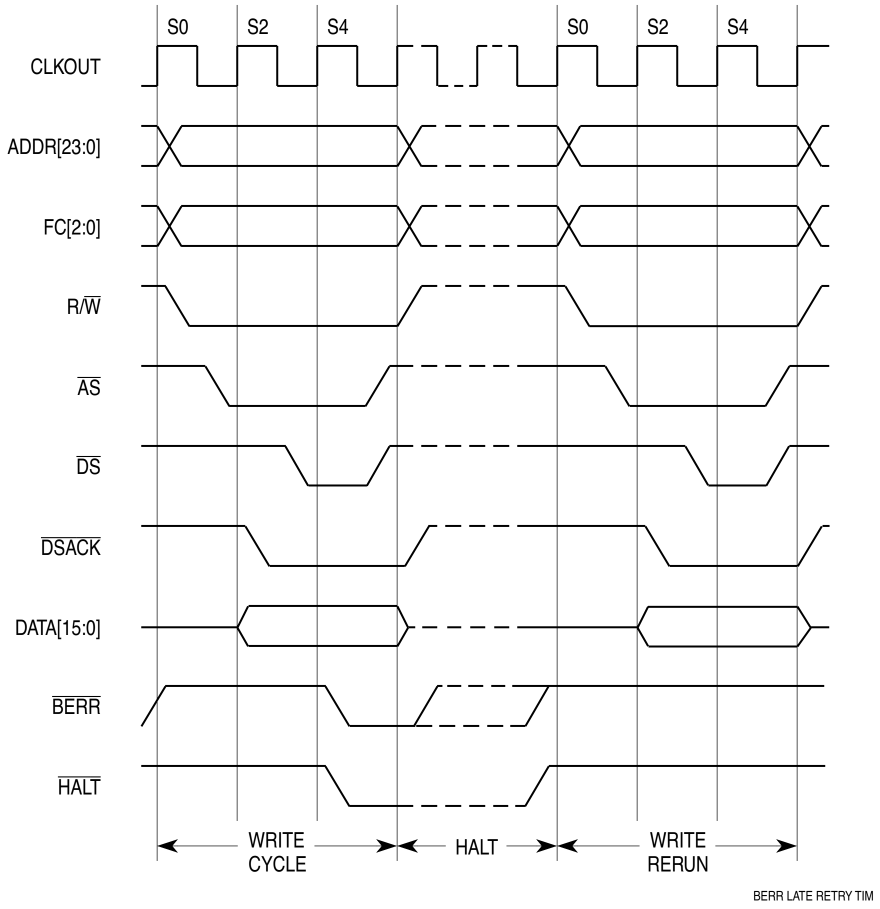


Figure 5-34 Late Retry Sequence

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 $\overline{\text{HALT}}$ signal is asserted, the data bus goes to high-impedance state and the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ 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 $\overline{\text{HALT}}$ 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.

NOTE

Some MCUs with reduced pin-count SCIMs do not include a $\overline{\text{HALT}}$ pin. Refer to **SECTION 10 REDUCED PIN-COUNT SCIM** and to the user's manual for the particular MCU for additional information.

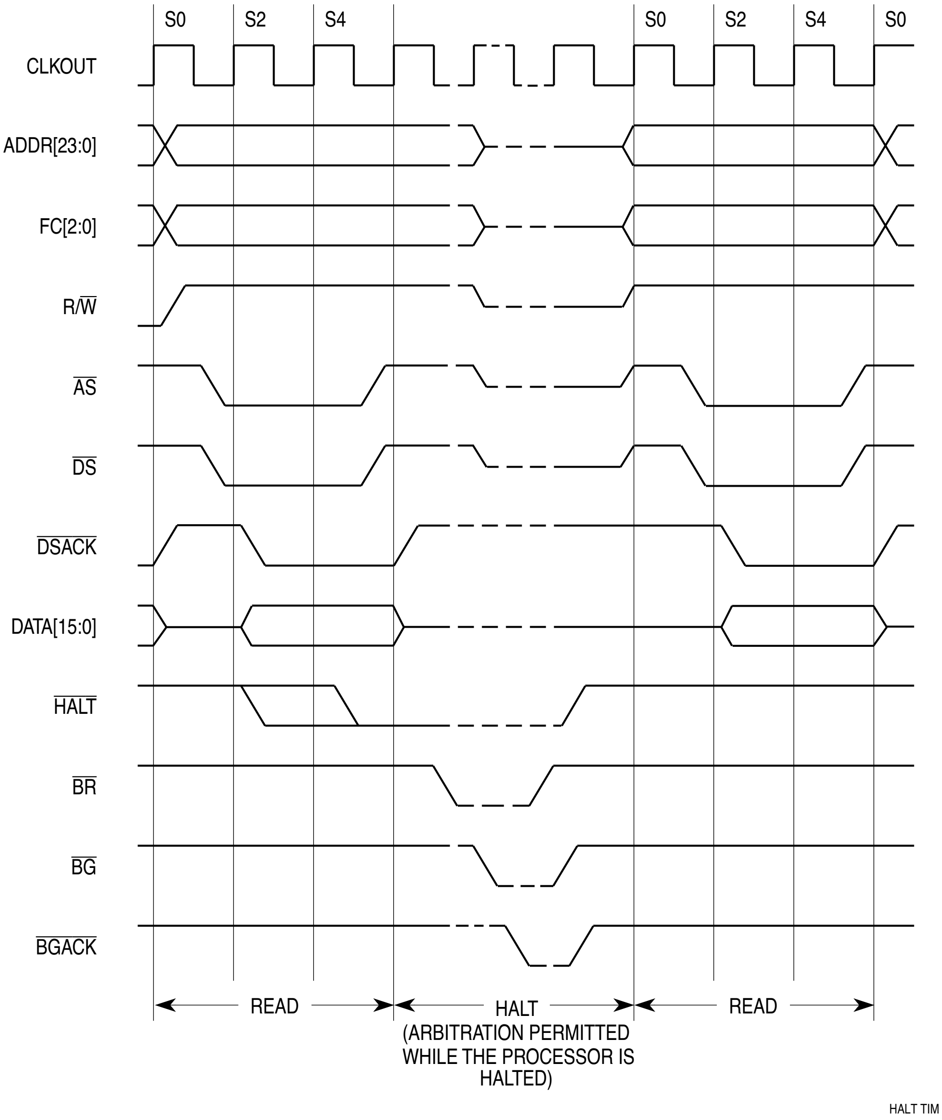


Figure 5-35 $\overline{\text{HALT}}$ Timing

5.10 Bus Arbitration

MCU bus design provides for a single bus master at any one time. When the MCU is not configured for single-chip operation, 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 flowchart of bus arbitration for a single device.

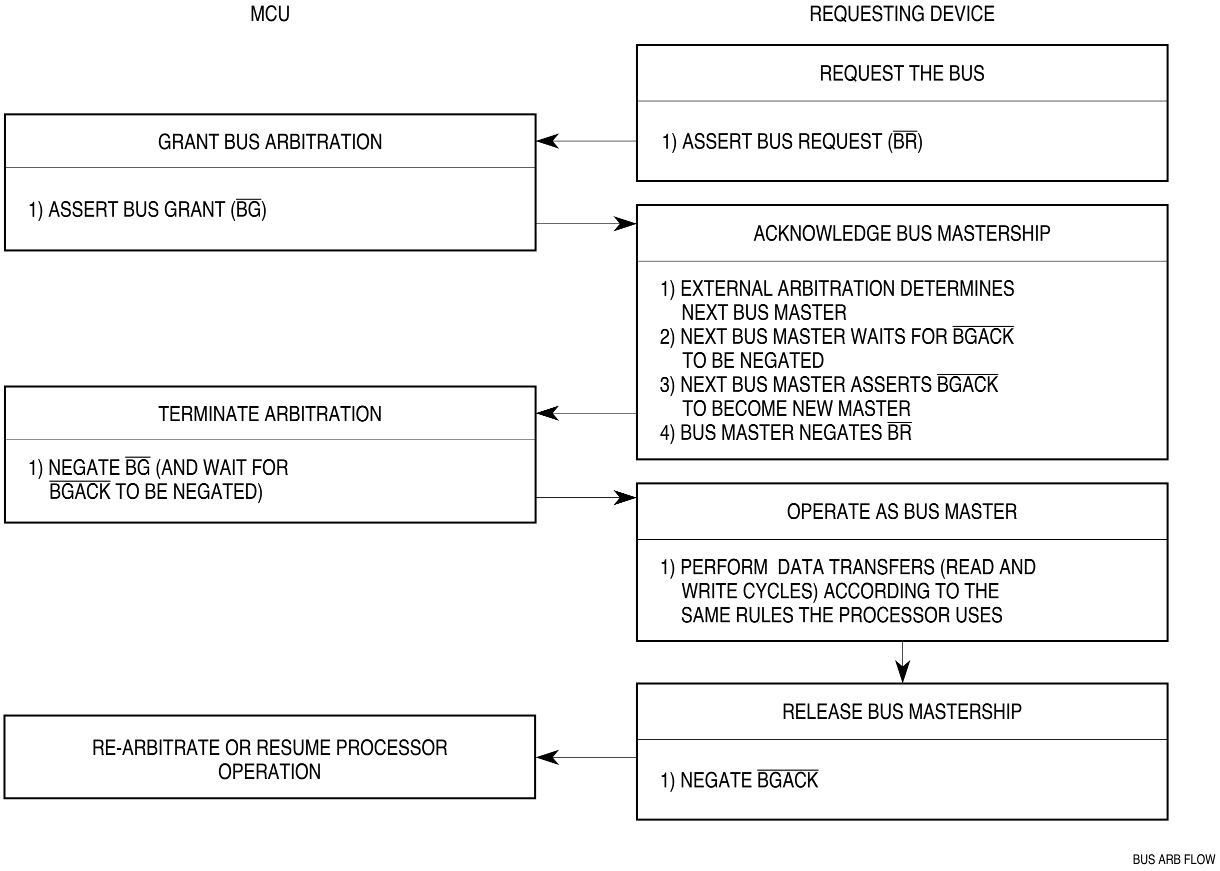

MCU
REQUESTING DEVICE
REQUEST THE BUS
1) ASSERT BUS REQUEST ($\overline{\text{BR}}$)
GRANT BUS ARBITRATION
1) ASSERT BUS GRANT ($\overline{\text{BG}}$)
ACKNOWLEDGE BUS MASTERSHIP
1) EXTERNAL ARBITRATION DETERMINES NEXT BUS MASTER
2) NEXT BUS MASTER WAITS FOR $\overline{\text{BGACK}}$ TO BE NEGATED
3) NEXT BUS MASTER ASSERTS $\overline{\text{BGACK}}$ TO BECOME NEW MASTER
4) BUS MASTER NEGATES $\overline{\text{BR}}$
TERMINATE ARBITRATION
1) NEGATE $\overline{\text{BG}}$ (AND WAIT FOR $\overline{\text{BGACK}}$ TO BE NEGATED)
OPERATE AS BUS MASTER
1) PERFORM DATA TRANSFERS (READ AND WRITE CYCLES) ACCORDING TO THE SAME RULES THE PROCESSOR USES
RELEASE BUS MASTERSHIP
1) NEGATE $\overline{\text{BGACK}}$
RE-ARBITRATE OR RESUME PROCESSOR OPERATION
BUS ARB FLOW

Figure 5-36 Bus Arbitration Flowchart for Single Request

5.10.1 Bus Request

External devices capable of becoming bus masters request the bus by asserting the $\overline{\text{BR}}$ 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 $\overline{\text{BG}}$, then releases the bus when $\overline{\text{BGACK}}$ 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 BG assertion after BR assertion depends on internal synchronization (as specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**).
- During an external transfer, the MCU does not assert BG until after the last cycle of the transfer (determined by SIZ[1:0] and DSACK[1:0] signals).
- During an external transfer, the MCU does not assert BG as long as RMC is asserted.
- When SHEN bits are both set and the CPU is making internal accesses, the MCU does not assert BG until the CPU finishes the internal transfers.

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

Externally, the 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

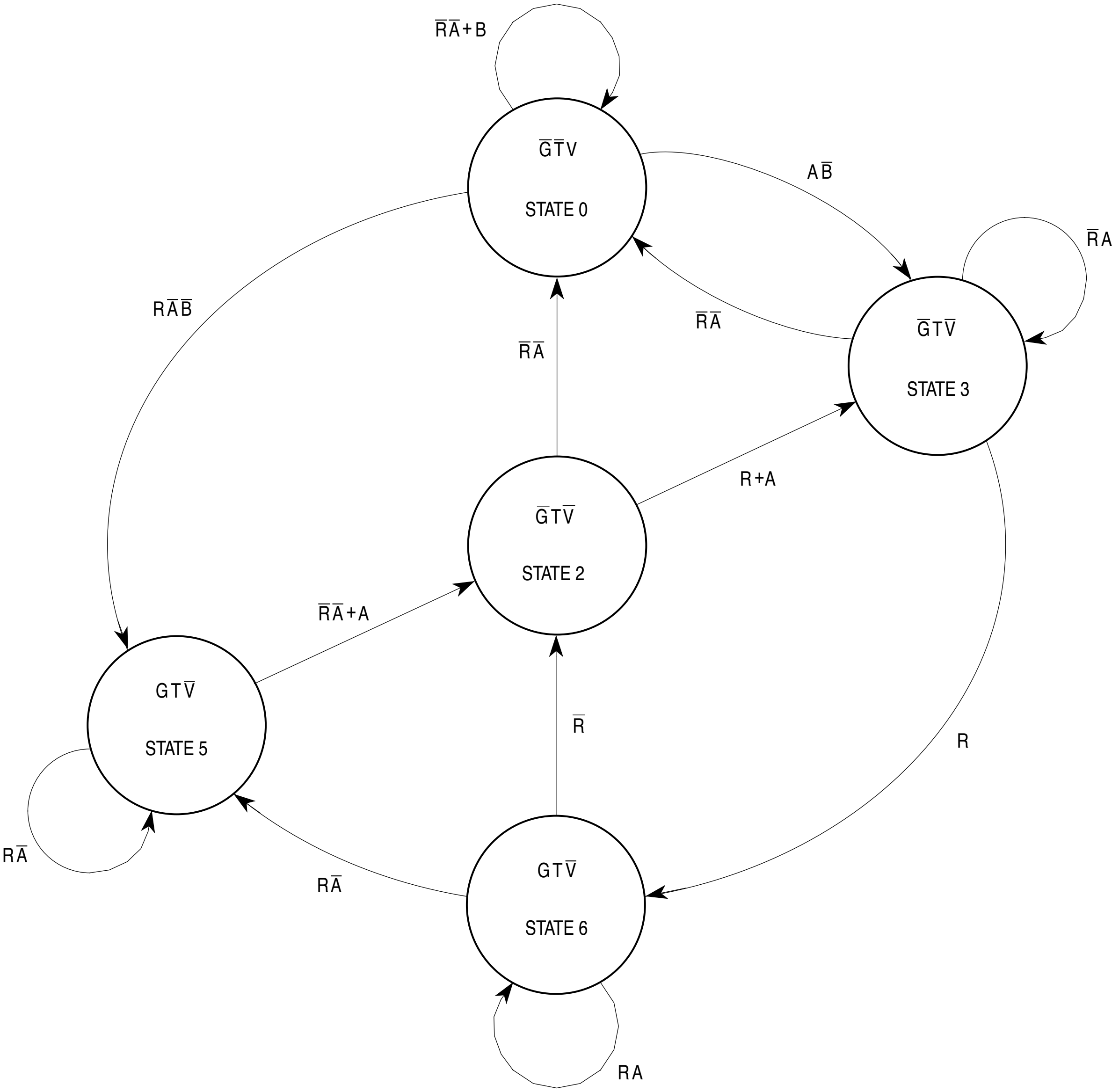
When bus protocols are obeyed, a device becomes the active bus master when it asserts BGACK. An external device cannot request and be granted the bus while another device is the active bus master. A device remains the bus master until it negates BGACK. BGACK must not be negated until all required bus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

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


$\overline{R}\,\overline{A}+B$
$\overline{G}\,\overline{T}\,V$
STATE 0
$A\overline{B}$
$\overline{R}A$
$\overline{R}\,\overline{A}$
$R\overline{A}\,\overline{B}$
$\overline{R}\,\overline{A}$
$\overline{G}\,T\,\overline{V}$
STATE 3
$R+A$
$\overline{G}\,T\,\overline{V}$
STATE 2
$\overline{R}\,\overline{A}+A$
$G\,T\,\overline{V}$
STATE 5
$\overline{R}$
R
$R\overline{A}$
$R\overline{A}$
$G\,T\,\overline{V}$
STATE 6
RA

R - BUS REQUEST	G - BUS GRANT
A - BUS GRANT ACKNOWLEDGE	T - THREE-STATE SIGNAL TO BUS CONTROL
B - BUS CYCLE IN PROGRESS	V - BUS AVAILABLE TO BUS CONTROL

NOTE: All figures are shown in positive logic (active high) regardless of their active state.

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 changes 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.

5.10.5 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 $\overline{\mathrm{BG}}$ 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, provided the MCU is not configured for single-chip operation. $\overline{\mathrm{AS}}$ is not asserted externally during show cycles.

Show cycles are controlled by the SHEN field in the SCIMCR. (Refer to **3.1.7 SCIM 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 $\overline{\mathrm{AS}}$ and $\overline{\mathrm{DS}}$ are not asserted externally and external data bus pins are in high-impedance state during internal accesses.

When show cycles are enabled, $\overline{\mathrm{DS}}$ 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 SCIM 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/$\overline{\mathrm{W}}$ 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) — $\overline{\mathrm{DS}}$ 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) — $\overline{\mathrm{DS}}$ 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[32:0], FC[2:0], R/$\overline{\mathrm{W}}$, 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 single-chip 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 SCIM includes three sources of interrupt requests: the periodic interrupt timer, port F edge-detect logic, and the external bus interface. The external bus interface routes external interrupt requests (i.e., requests received via the interrupt request pins $\overline{\text{IRQ[7:1]}}$ to the CPU. During interrupt arbitration (see **6.3 Interrupt Arbitration**), the CPU treats external interrupt requests as though they come from the SCIM.

NOTE

MCUs with a reduced pin-count SCIM 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 more than one SCIM source, the PIT is given highest priority, followed by the $\overline{\text{IRQ}}$ pins, and then the port F edge-detect logic. The interrupt from the lower-priority source remains pending until the next allowable interrupt time.

6.2 Interrupt Level and Recognition

Each of the interrupt request signals $\overline{\text{IRQ[7:1]}}$ corresponds to an interrupt priority level. $\overline{\text{IRQ1}}$ has the lowest priority and $\overline{\text{IRQ7}}$ the highest. For periodic timer interrupts, the PIRQ field in the periodic interrupt control register (PICR) determines the priority level. The port F edge-detect interrupt level register (PFLVR) contains the priority level of port F edge-detect interrupts. A priority level of zero in the PICR or PFLVR means that PIT or port F edge detect interrupts, respectively, 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 (IARB) field bit values. Each module that can make an interrupt service request, including the SCIM, 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 SCIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SCIM 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 SCIM 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:

A. FC[2:0] are set to %111, the CPU space encoding.
B. ADDR[19:16] (the CPU space type field) are set to %1111, the interrupt acknowledge encoding.
C. ADDR[3:1] are set to the interrupt request level.
D. All remaining address bits are set.
E. 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.

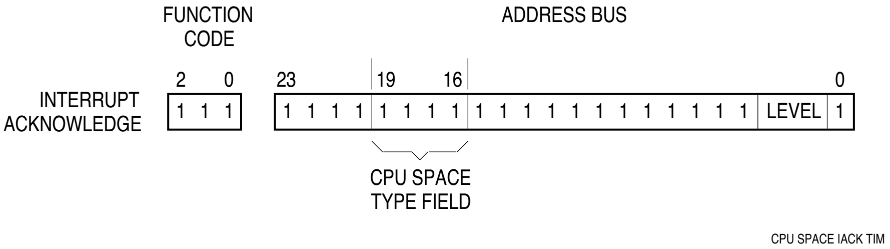


Figure 6-1 Interrupt-Acknowledge Read Cycles

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 flowchart of the interrupt-acknowledge cycle.

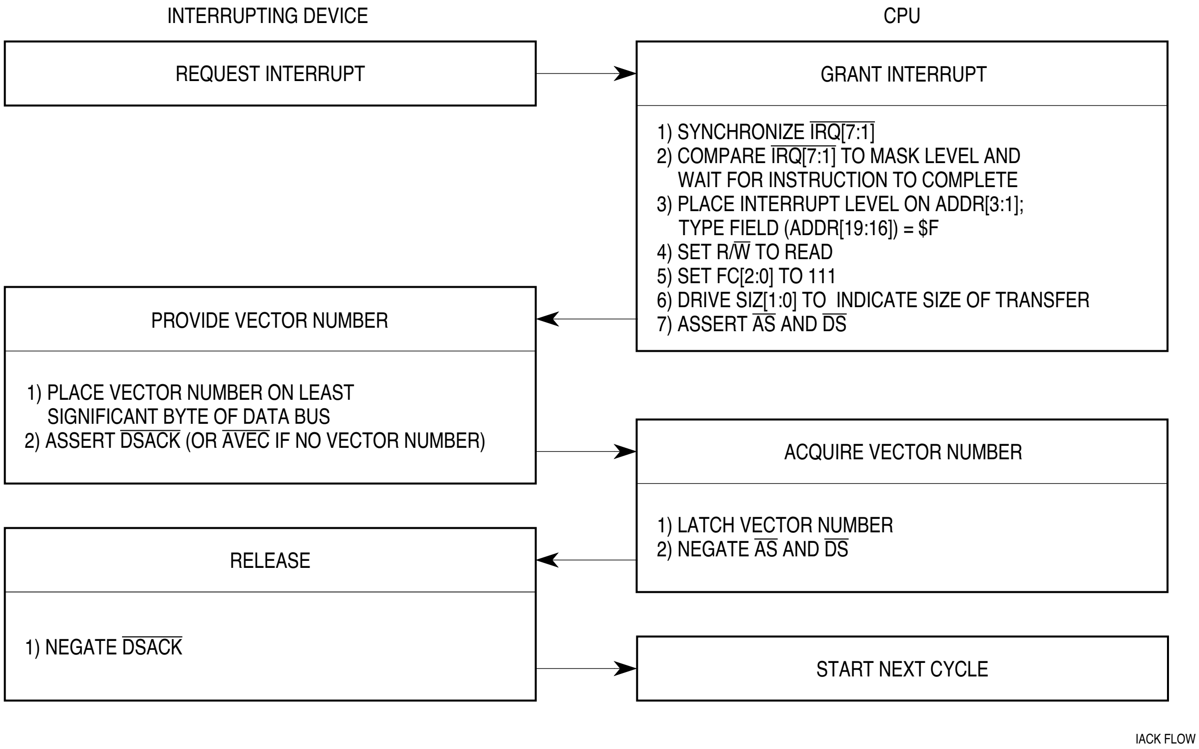

INTERRUPTING DEVICE
CPU
REQUEST INTERRUPT
GRANT INTERRUPT
1) SYNCHRONIZE IRQ[7:1]
2) COMPARE IRQ[7:1] TO MASK LEVEL AND WAIT FOR INSTRUCTION TO COMPLETE
3) PLACE INTERRUPT LEVEL ON ADDR[3:1]; TYPE FIELD (ADDR[19:16]) = $F
4) SET R/W TO READ
5) SET FC[2:0] TO 111
6) DRIVE SIZ[1:0] TO INDICATE SIZE OF TRANSFER
7) ASSERT AS AND DS
PROVIDE VECTOR NUMBER
1) PLACE VECTOR NUMBER ON LEAST SIGNIFICANT BYTE OF DATA BUS
2) ASSERT DSACK (OR AVEC IF NO VECTOR NUMBER)
ACQUIRE VECTOR NUMBER
1) LATCH VECTOR NUMBER
2) NEGATE AS AND DS
RELEASE
1) NEGATE DSACK
START NEXT CYCLE
IACK FLOW

Figure 6-2 Interrupt-Acknowledge Cycle Flowchart

6.4.1 Bus Cycle Terminated by $\overline{\text{DSACK}}$ 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 $\overline{\text{DSACK}}$ 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 $\overline{\text{DSACK}}$.

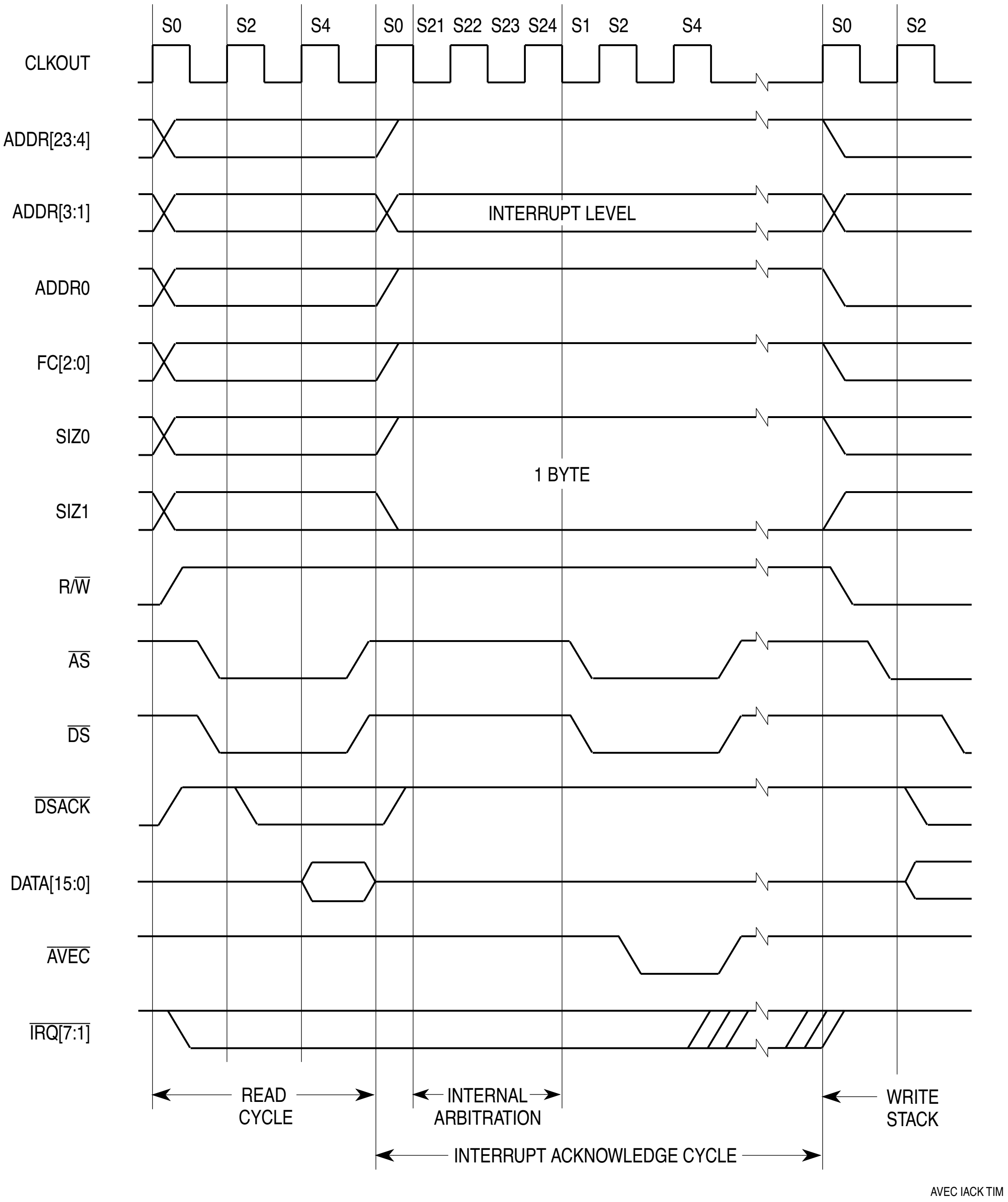


Figure 6-3 Interrupt-Acknowledge Cycle Timing

Figure 6-4 indicates possible pin connections and external logic connecting the SCIM and an interrupting external device that provides a vector number. The design shown in **Figure 6-4** can be simplified by using SCIM 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**.

NOTE

If the MCU has a reduced pin-count SCIM without a $\overline{\text{DSACK0}}$ pin, devices with 8-bit ports must use SCIM chip selects to assert the $\overline{\text{DSACK0}}$ signal internally during interrupt-acknowledge cycles.

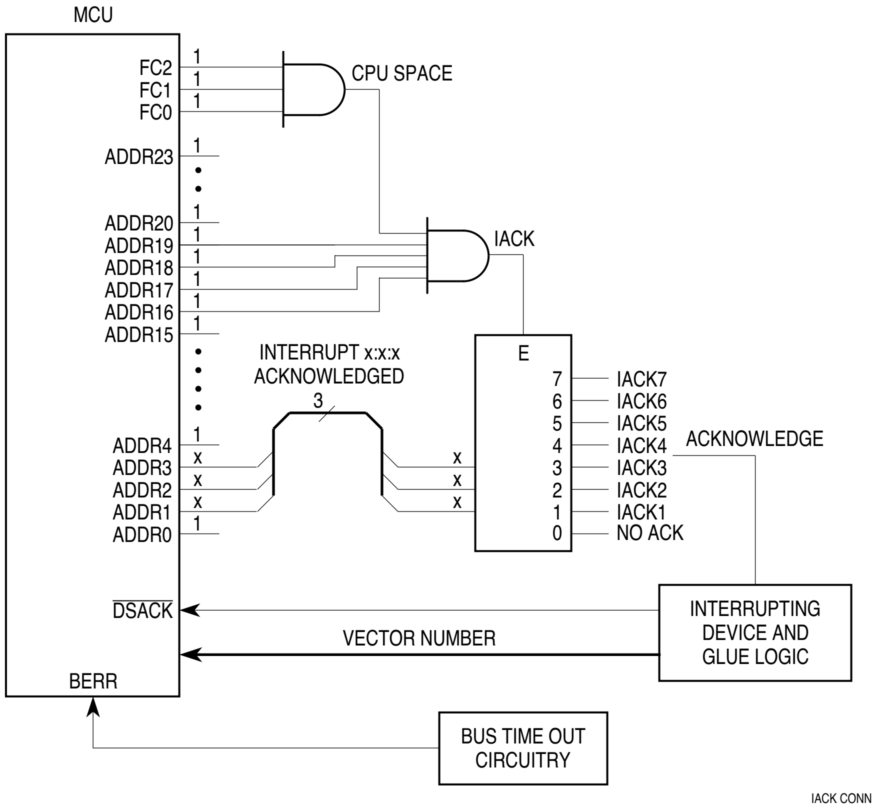


Figure 6-4 External Connections for External Interrupt Processing

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.

NOTE

> If the MCU has a reduced pin-count SCIM without an $\overline{\text{AVEC}}$ pin, devices with 8-bit ports must use SCIM chip selects to assert the $\overline{\text{AVEC}}$ signal internally during interrupt-acknowledge cycles.

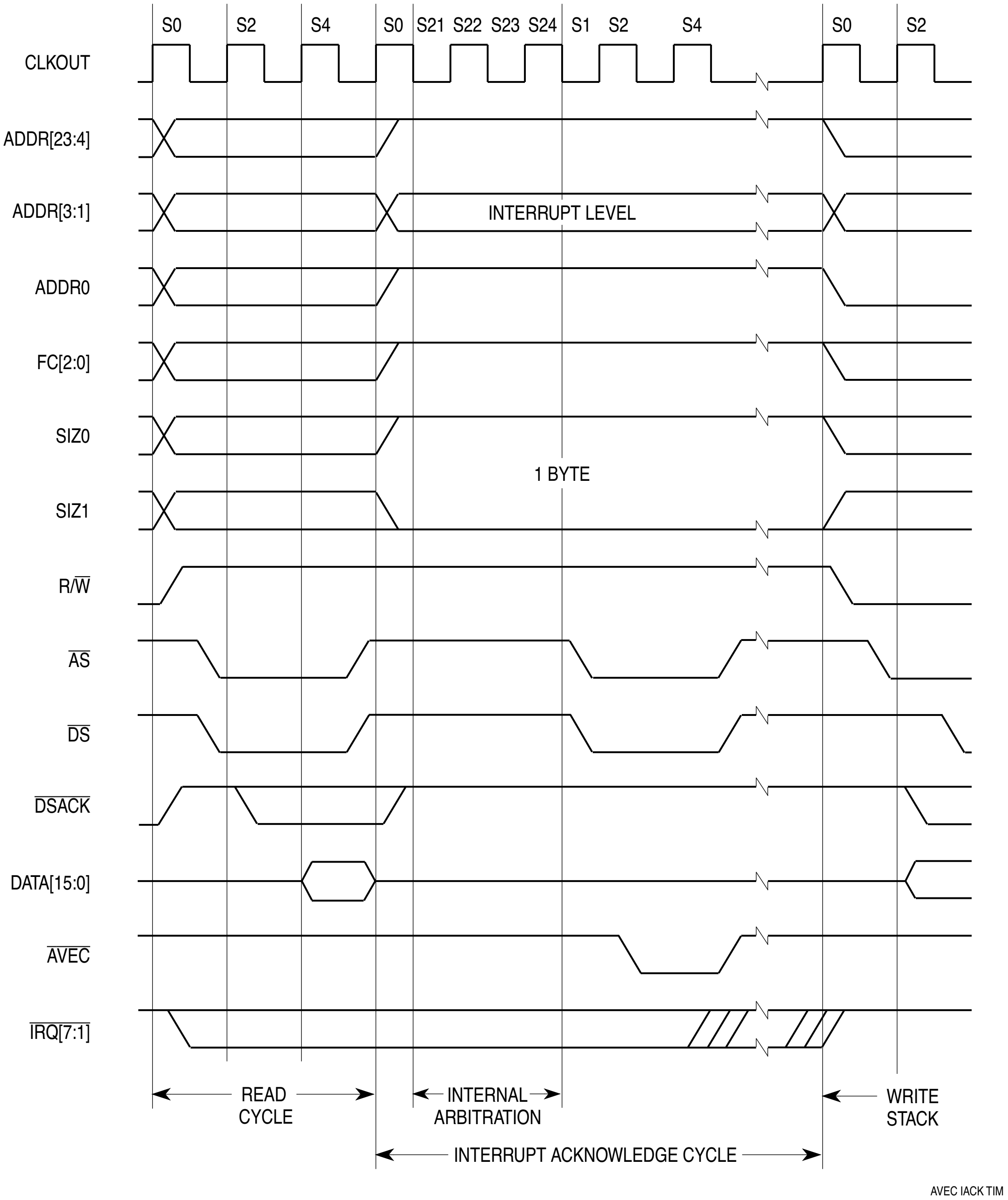


Figure 6-5 Autovector Timing

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 SCIM includes nine 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 SCIMs may not have all nine chip-select pins available. Consult the user's manual for the particular MCU for details.

In addition to the nine general-purpose chip-select circuits, the SCIM provides two special chip selects to support emulation of on-chip ROM and port I/O. Refer to **7.9 Emulation-Support Chip Selects** for additional information.

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.

7.12 Interfacing Example with SCIM 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.

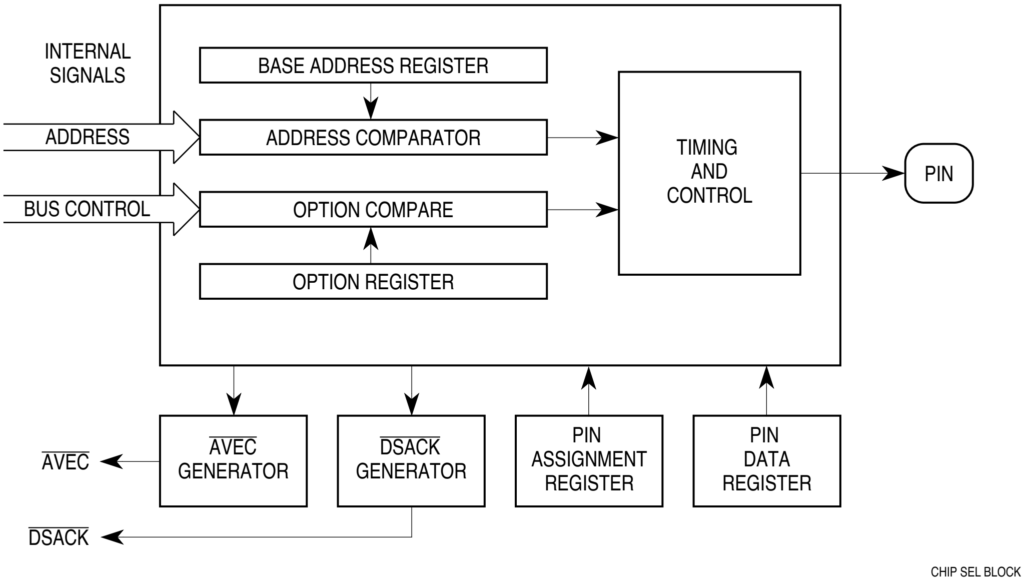


Figure 7-1 Chip-Select Circuit Block Diagram

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.
- When the SPACE field is not programmed for CPU space, the IPL field specifies whether to assert the chip select during accesses to program space, data space, or both.

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.
- SPACE must be set to%00 (CPU space) for the chip-select circuit to respond to interrupt-acknowledge cycles.
- When SPACE is set to %00, IPL specifies the interrupt priority level to which the chip select responds.

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		001 = Level 1	1 = On
	10 = Upper	10 = Write		0010 = 2 wait states		010 = Level 2	
	*11 = Both	11 = Both		0011 = 3 wait states		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	01 = User	000 = Data/Prog	
				1001 = 9 wait states	10 = Supv	001 = Data Sp	
				1010 = 10 wait states	11 = S/U*	010 = Prog Sp	
				1011 = 11 wait states		011 = Reserved	
				1100 = 12 wait states		100 = Reserved	
				1101 = 13 wait states		101 = Data Sp	
				1110 = Fast terminate		110 = Prog Sp	
				1111 = External		111 = Reserved	

*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**, **7.6 Chip Selects and Dynamic Bus Sizing**, 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 **7.11 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 **7.11 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 the vector table and initialization routine 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 **7.10 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 and CSPAR0) contain 2-bit fields 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

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

Table 7-4 shows the 2-bit field encodings in CSPAR0 and CSPAR1 that assign functions to the pins listed in **Table 7-3**. Pins that have no discrete output or ECLK function do not use the %00 encoding.

Table 7-4 Pin Assignment Field Encoding

Encoding	Description
%00	Discrete Output or ECLK (Assert $\overline{\text{DSACK0}}$ internally on match)
%01	Alternate Function (Assert $\overline{\text{DSACK1}}$ internally on match)
%10	Chip Select, 8-Bit Port (Assert $\overline{\text{DSACK0}}$ internally on match)
%11	Chip Select, 16-Bit Port (Assert $\overline{\text{DSACK1}}$ internally on match)

Port size is involved in dynamic bus sizing and determines which $\overline{\text{DSACK}}$ signal the chip-select circuit asserts during a bus cycle. 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 **7.6 Chip Selects and Dynamic Bus Sizing** and **7.8 Using Chip Selects in Interrupt-Acknowledge Cycles** for additional 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.

Refer to **7.10 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 $\overline{\text{CS[10:5]}}$, $\overline{\text{CS3}}$, and $\overline{\text{CS1}}$ 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 **7.10 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.

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, and to the IPL field in the option register, if the SPACE field encoding is not for CPU space.
3. Read/write status to R/$\overline{\text{W}}$ field in the option register.
4. ADDR0 or SIZ signals 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 $\overline{\text{AS}}$ 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.

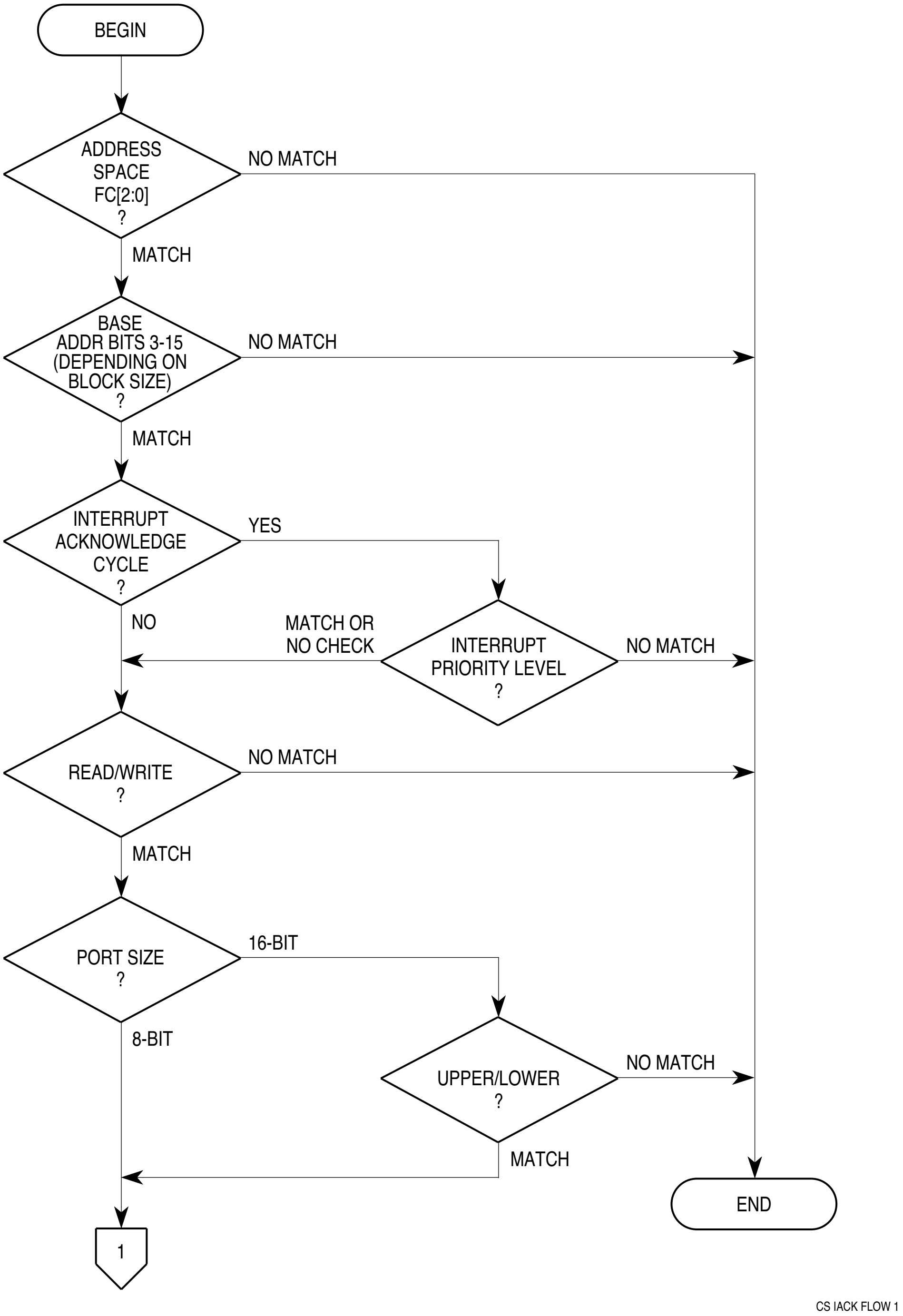


Figure 7-2 Flow Diagram for Chip Select (sheet 1 of 3)

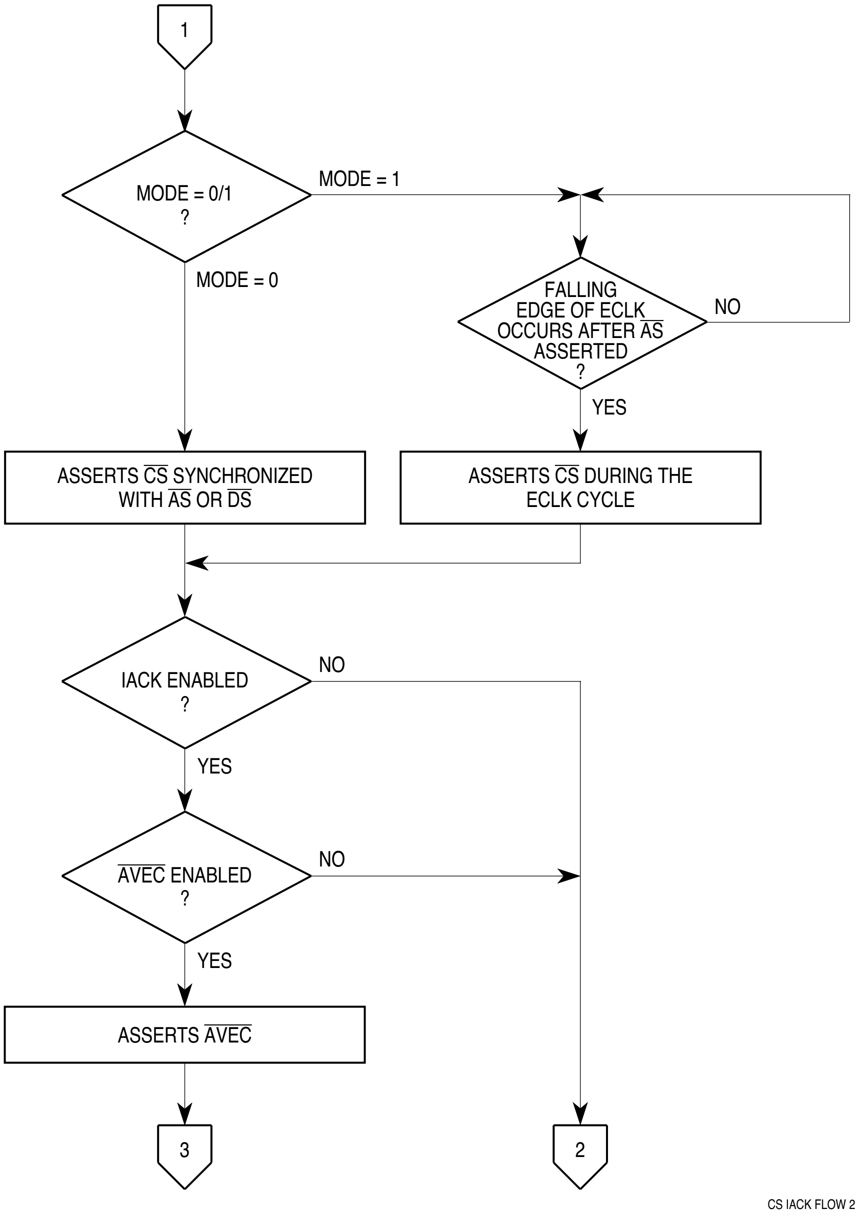

1
MODE = 0/1 ?
MODE = 1
MODE = 0
FALLING EDGE OF ECLK OCCURS AFTER AS ASSERTED ?
NO
YES
ASSERTS CS SYNCHRONIZED WITH AS OR DS
ASSERTS CS DURING THE ECLK CYCLE
IACK ENABLED ?
NO
YES
AVEC ENABLED ?
NO
YES
ASSERTS AVEC
3
2
CS IACK FLOW 2

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

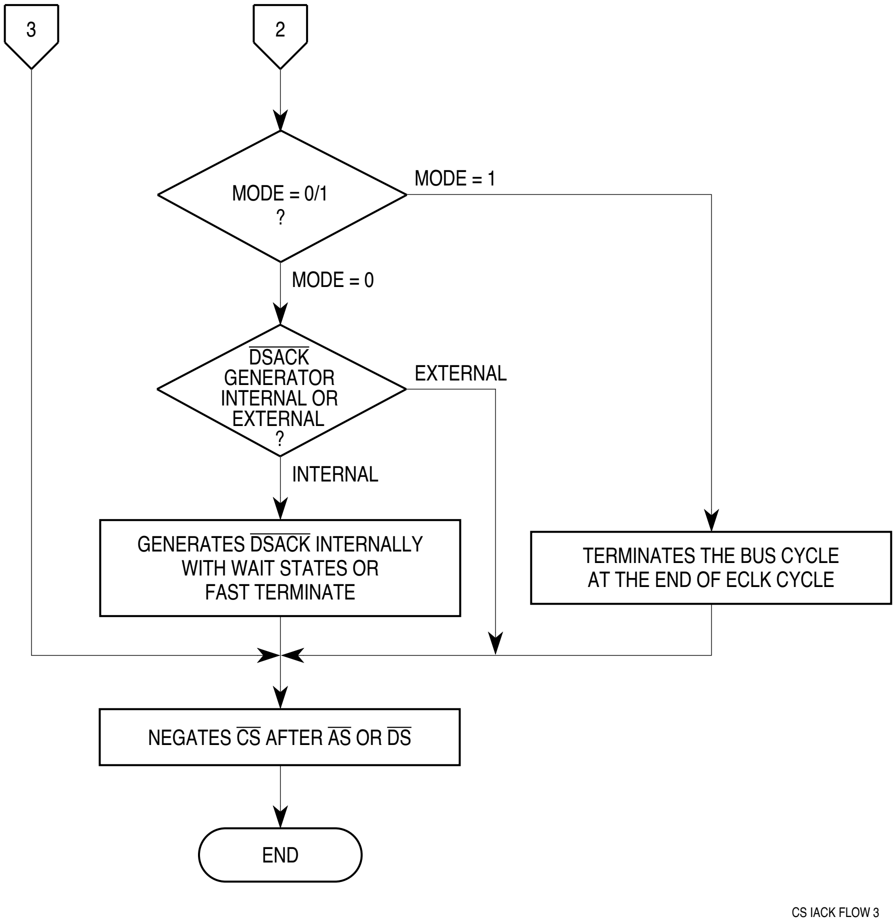


Figure 7-2 Flow Diagram for Chip Select (sheet 3 of 3)

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 DSACK and STRB fields further define chip-select timing.

7.5.1 Synchronization with AS or 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 AS or 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 (DSACK) signal or by an autovector (AVEC) signal.

7.5.2 Wait States and $\overline{\text{DSACK}}$ Generation

The value of the $\overline{\text{DSACK}}$ field in the associated chip-select option register specifies the number of wait states to insert before generating $\overline{\text{DSACK0}}$ or $\overline{\text{DSACK1}}$ internally. An encoding of %1111 in this field specifies that the external device will provide the $\overline{\text{DSACK}}$ signal.

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.

Refer to **7.6 Chip Selects and Dynamic Bus Sizing** for information on assigning port size in order to generate the appropriate internal signal ($\overline{\text{DSACK0}}$ or $\overline{\text{DSACK1}}$) to terminate a bus cycle.

7.5.3 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. Two-bit pin assignment fields in CSPAR0 and CSPAR1 assign port sizes of eight or sixteen bits to the chip-select circuits. Port size assignment determines which signal the chip-select logic asserts ($\overline{\text{DSACK0}}$ or $\overline{\text{DSACK1}}$) after the specified number of wait states elapse during a chip-select access. An encoding of %10 causes $\overline{\text{DSACK0}}$ to be asserted, indicating an 8-bit port. An encoding of %11 causes $\overline{\text{DSACK1}}$ to be asserted, indicating a 16-bit port.

If the chip-select pin is programmed for discrete output (encoding %00) or alternate function (encoding %01), the associated chip-select registers can still be programmed to generate $\overline{\text{DSACK}}$ internally. In this case, the low-order bit in the pin-assignment field determines port size. The discrete output encoding (%00) causes $\overline{\text{DSACK0}}$ to be generated when match conditions are met, and the alternate function encoding (%01) causes $\overline{\text{DSACK1}}$ to be generated.

NOTE

MCUs with a reduced pin-count SCIM and no $\overline{\text{DSACK0}}$ pin still have the internal $\overline{\text{DSACK0}}$ signal available. These MCUs can follow the procedures in this subsection to assert the appropriate $\overline{\text{DSACK}}$ signal using chip-select logic.

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 **7.12 Interfacing Example with SCIM 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 DSACK field in the appropriate chip-select option register. Fast termination cycles are only available in conjunction with chip selects.

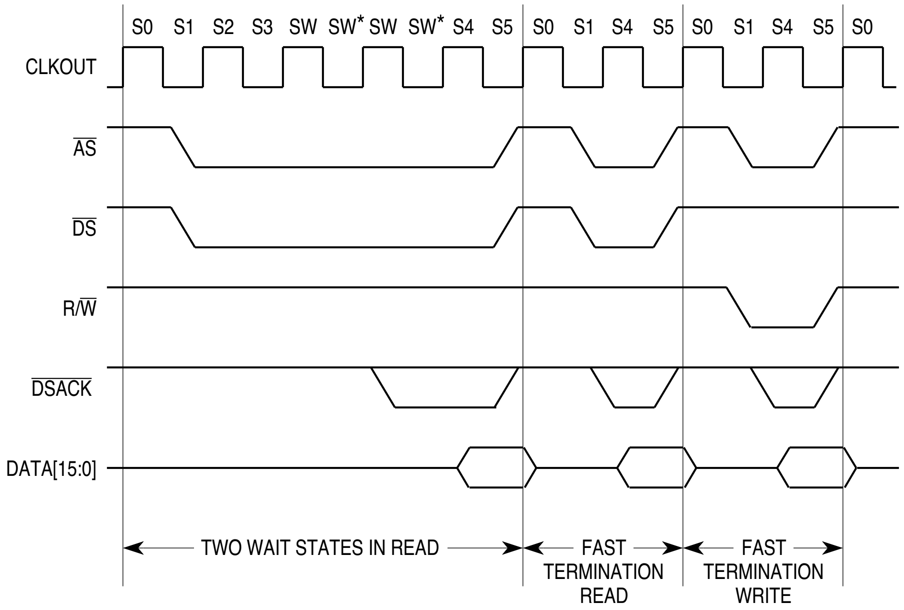
If multiple chip selects are to be used to select the same device that can support fast termination, and match conditions can occur simultaneously, program the DSACK field in each associated chip-select option register for fast termination. Alternately, program one DSACK field for fast termination and the remaining 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 AS, DS, and R/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 DSACK timing for a read cycle with two wait states, followed by a fast-termination read cycle and a fast-termination write cycle.



* $\overline{\text{DSACK}}$ only internally asserted for fast termination.

FAST TERM TIM

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 in state 0. The CPU places an address on ADDR[23:0] and places function codes on FC[2:0]. The CPU drives R/$\overline{\text{W}}$ 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 $\overline{\text{AS}}$ indicating that the address on the address bus is valid. The CPU also asserts $\overline{\text{DS}}$, indicating to external devices that data can be placed on the data bus. SCIM chip-select logic decodes the appropriate address lines, FC[1:0], R/$\overline{\text{W}}$, 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 $\overline{\text{DSACK}}$ signals are generated and the CPU latches data on the falling edge of S4.

State 5 (S5) — The CPU 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 until either $\overline{\text{AS}}$ or $\overline{\text{DS}}$ is negated. It must remove the data 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.

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/$\overline{\text{W}}$ 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 $\overline{\text{AS}}$, indicating that the address on the address bus is valid. SCIM chip-select logic decodes the appropriate address lines, FC[1:0], R/$\overline{\text{W}}$, SIZ[1:0], and $\overline{\text{AS}}$.

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

State 5 (S5) — The MCU negates $\overline{\text{AS}}$ but holds the address and data 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.

7.8 Using Chip Selects in Interrupt-Acknowledge Cycles

Chip-select circuits can be programmed to respond during interrupt-acknowledge cycles initiated by an external device asserting an $\overline{\text{IRQ}}$ 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 ($\overline{\text{AVEC}}$) signals internally. For example, if the ADDR0/$\overline{\text{CS10}}$/ECLK pin is configured to generate ECLK, the associated chip-select circuit can be used to generate $\overline{\text{AVEC}}$ internally when match conditions are met.

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/$\overline{\text{W}}$ 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.

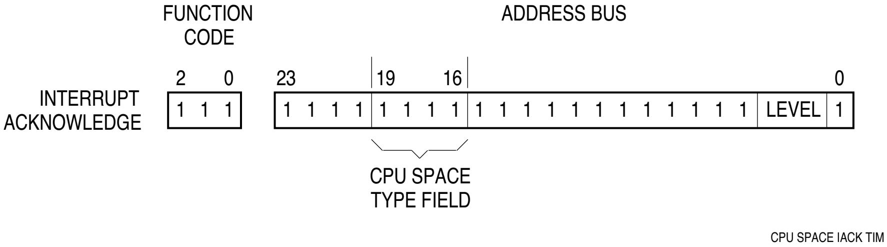


Figure 7-4 CPU Space Encoding for Interrupt-Acknowledge Cycles

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 $\overline{\text{AS}}$.

- Program the $\overline{\text{DSACK}}$ field to the desired number of wait states. Select the %1111 option if the external device will generate $\overline{\text{DSACK}}$ signals.
- Program IPL to respond to the desired interrupt request level, or to %000 to respond to all request levels.
- Program the $\overline{\text{AVEC}}$ bit to zero 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 $\overline{\text{AVEC}}$ 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 $\overline{\text{IRQ}}$ pin, provided the match conditions in the base address and option registers are met. The $\overline{\text{AVEC}}$ signal causes the CPU to use a predetermined set of vectors to service the interrupt.

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

The $\overline{\text{AVEC}}$ 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], corresponding to ADDR[23:11]) 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 %1111, the CPU interrupt-acknowledge space type encoding, 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 both bytes.
    - Program STRB to synchronize with $\overline{\text{AS}}$.
    - Program the $\overline{\text{DSACK}}$ field to any value. When the $\overline{\text{AVEC}}$ 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 $\overline{\text{AVEC}}$ bit to one to enable autovector generation.

7.9 Emulation-Support Chip Selects

The SCIM contains logic that can be used to replace on-chip ports externally. It also contains special support logic to allow external emulation of internal ROM. This emulation support allows system development of a single-chip application in expanded mode.

Two special chip selects, $\overline{\text{CSE}}$ and $\overline{\text{CSM}}$, support external emulation of on-chip ports and on-chip ROM, respectively. Emulator mode is a special type of 16-bit expanded operation. It is entered by holding DATA10 low, $\overline{\text{BERR}}$ high, and DATA1 low during reset.

In emulator mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. An emulator chip select ($\overline{\text{CSE}}$) is asserted whenever one of the externally-mapped registers is addressed. In addition, on MCUs with a masked ROM module, the memory chip select signal $\overline{\text{CSM}}$ is asserted whenever a valid access to an address assigned to the masked ROM array is made.

Chip select pin assignment register 0 (CSPAR0) assigns the $\overline{\text{CSE}}$/$\overline{\text{BGACK}}$ and $\overline{\text{CSM}}$/$\overline{\text{BG}}$ pins for chip-select or alternate function. $\overline{\text{CSE}}$ and $\overline{\text{CSM}}$ do not have associated base and option registers.

7.9.1 Port Emulation

In emulator mode, an emulator chip select ($\overline{\text{CSE}}$) is asserted whenever a port A, B, E, G, or H data or data direction register or the port E pin assignment register is addressed during a read or write cycle. The SCIM does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. The port C data register and port F data, data direction, and pin assignment registers are accessible normally in emulator mode.

$\overline{\text{CSE}}$ is asserted on the falling edge of $\overline{\text{AS}}$. The SCIM generates an internal $\overline{\text{DSACK}}$ signal without inserting any wait states, corresponding to a three-clock-cycle access.

7.9.2 ROM Emulation

In emulator mode, on MCUs with a masked ROM module, memory chip select signal $\overline{\text{CSM}}$ is asserted whenever a valid access to an address assigned to the masked ROM array is made. The ROM module does not acknowledge IMB accesses while in emulation mode. This causes the SCIM to run an external bus cycle for each access. The ROM module generates an internal $\overline{\text{DSACK}}$ signal after inserting the number of wait states specified by the WAIT field in the MRMCR. See the discussion of the masked ROM module in the user's manual for the particular MCU for more information.

7.10 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.

7.10.1 Pin Assignment

Out of reset, chip-select pin functions are determined by the external bus configuration selected. If fully expanded operation is selected, pin functions are further defined by the logic levels on pins on DATA[7:2] and DATA0. 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.

When single-chip operation is selected, the chip-select pins are configured for discrete output (port C); no chip selects are available. When partially-expanded operation is selected, chip-select pins are always configured for their chip-select function out of reset. When fully-expanded operation is selected, the logic levels on DATA[7:2] and DATA0 determine pin function out of reset and the reset values of CSPAR0 and CSPAR1, as shown in **Table 7-6** and **Table 7-7**.

Table 7-6 summarizes the reset function of pins controlled by CSPAR0.

Table 7-6 Reset Function of Pins
Controlled by CSPAR0 in Fully Expanded Mode

Mode Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
DATA2	$\overline{\text{CS0}}$ $\overline{\text{CS3}}$ $\overline{\text{CS5}}$	$\overline{\text{BR}}$ FC0 FC2
DATA10	$\overline{\text{BG}}$ $\overline{\text{BGACK}}$	$\overline{\text{CSM}}$ $\overline{\text{CSE}}$

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 Function of Pins
Controlled by CSPAR1 in Fully Expanded Mode

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**	**$\overline{\text{CS7}}$/ ADDR23**	**$\overline{\text{CS6}}$/ ADDR23**
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

7.10.2 Chip-Select Base and Option Registers

Base address and option registers for $\overline{\text{CS[10:5]}}$, $\overline{\text{CS3}}$, and $\overline{\text{CS0}}$ 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.

7.10.3 $\overline{\text{CSBOOT}}$ Base 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 an external 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	Data or Program Space
Autovector	Interrupt Vector Externally

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

7.11 Chip-Select Register Diagrams

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

7.11.1 Chip-Select Pin Assignment Registers

The pin assignment registers contain 2-bit fields 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 or ECLK function do not use the %00 encoding.

Table 7-9 Pin Assignment Field Encoding

Encoding	Description
%00	Discrete Output or ECLK (Assert $\overline{\text{DSACK0}}$ internally on match)
%01	Alternate Function (Assert $\overline{\text{DSACK1}}$ internally on match)
%10	Chip Select, 8-Bit Port (Assert $\overline{\text{DSACK0}}$ internally on match)
%11	Chip Select, 16-Bit Port (Assert $\overline{\text{DSACK1}}$ internally on match)

NOTE

On MCUs with a reduced pin-count SCIM, 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 **7.10 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:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains six 2-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 7-10 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	—	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CSE}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CSM}}$	$\overline{\text{BG}}$	—
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:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	DATA7	1	DATA[7:6]	1	DATA[7:5]	1	DATA[7:4]	1	DATA[7:3]	1

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. Each pin is assigned one of the functions shown in **Table 7-11**. **Table 7-7** explains how pin function is assigned at reset. CSPAR1[15:10] are not used. These bits always return zero when read; writes have no effect.

Table 7-11 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

7.11.2 Chip-Select Base Address Registers

CSBARBT contains the base address for selection of a bootstrap peripheral memory device. Bit and field definitions for CSBARBT are the same as for the other chip-select base address registers, 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		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[10:5] — Chip-Select Base Address Registers **$####60–$####74**
CSBAR3 **$####58**
CSBAR0 **$####4C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2		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.

7.11.3 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 are the same as for the other chip-select option registers, but their reset settings differ.

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

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

CSOR[10:5] — Chip-Select Option Registers **$####62–$####76**
CSOR3 **$####5A**
CSOR0 **$####4E**

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

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

This field specifies the source of $\overline{\text{DSACK}}$ when chip-select cycles emulate asynchronous bus cycles, and controls wait state insertion. Refer to **7.5.2 Wait States and DSACK Generation** and **7.6 Chip Selects and Dynamic Bus Sizing** for additional information.

Table 7-12 $\overline{\text{DSACK}}$ Encoding

Encoding	Description
%0000	0 Wait States
%0001	1 Wait State
%0010	2 Wait States
%0011	3 Wait States
%0100	4 Wait States
%0101	5 Wait States
%0110	6 Wait States
%0111	7 Wait States
%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 Termination
%1111	External $\overline{\text{DSACK}}$

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 in interrupt-acknowledge cycles (SPACE = 00). During user- or supervisor-space cycles, this field selects program or data space. Refer to **7.8 Using Chip Selects in Interrupt-Acknowledge Cycles** for additional information.

Table 7-13 IPL Encoding

IPL	Interrupt Level (SPACE = 00)	Data/Program Space (SPACE = 01, 10, 11)
000	All	Data or Program
001	Level 1	Data Space
010	Level 2	Program Space
011	Level 3	Reserved
100	Level 4	Reserved
101	Level 5	Data Space
110	Level 6	Program Space
111	Level 7	Reserved

$\overline{\text{AVEC}}$ — 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

7.11.4 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. 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 8	7	6	5	4	3	2	1	0
NOT USED	0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:								
	0	1	1	1	1	1	1	1

7.12 Interfacing Example with SCIM Chip Selects

Figure 7-5 shows a system configuration in which SCIM 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 associated chip-select base and option registers.

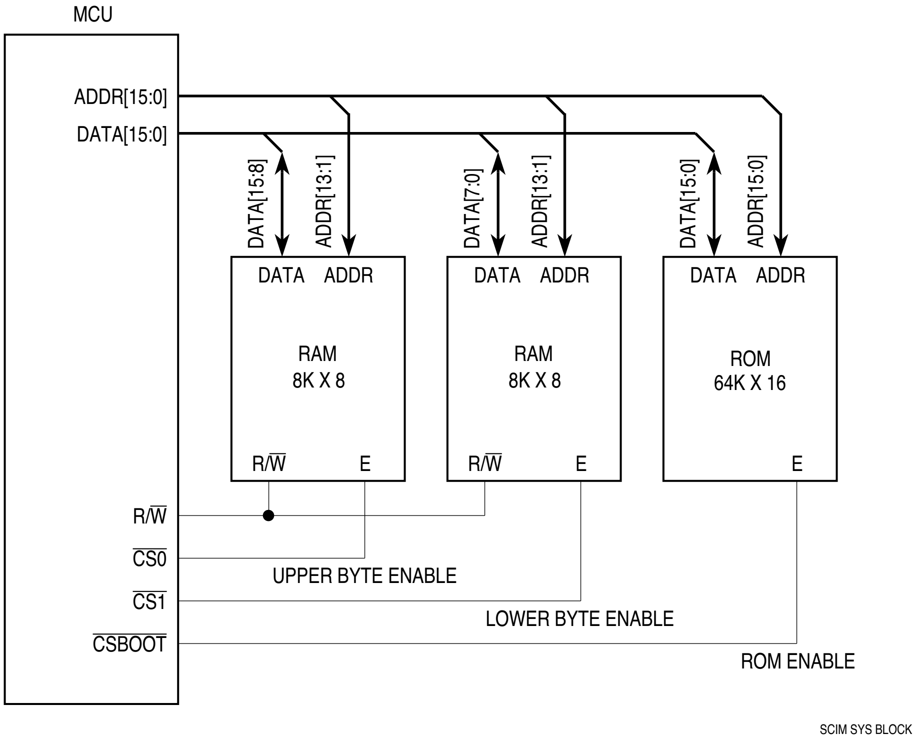


Figure 7-5 System Configuration with Chip Selects

7.12.1 Configuring the RAM Chip Selects

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

7.12.1.1 Pin Connections

The upper and lower memory banks are connected to $\overline{\text{CS0}}$ and $\overline{\text{CS3}}$ 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. Since separate chip selects are used for the upper and lower memory banks, byte accesses as well as word and long-word accesses are supported. (If the memory banks shared a common chip select, and $\overline{\text{CS1}}$ of the MCU were connected to $\overline{\text{CS0}}$ of the memory chips, byte accesses would not be possible.)

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.

7.12.1.2 Base Address Registers

The base address registers are programmed as follows:

CSBAR0 = $1001
CSBAR3 = $1001

This selects a block size of 8 Kbytes starting at address $100000.

7.12.1.3 Option Registers

The option registers (CSOR0 and CSOR3) 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 CSOR3. This assigns the memory bank connected to $\overline{\text{CS0}}$ as the upper byte and the bank connected to $\overline{\text{CS3}}$ 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 SCIM chip-select circuitry, with zero wait states inserted.

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

IPL = %000. The chip select is enabled during both data- and program-space accesses.

AVEC = %0000. Since the chip selects are not being used during interrupt-acknowledge cycles, this field is cleared to zero.

7.12.2 Configuring the Boot ROM Chip Select

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

7.12.2.1 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.

7.12.2.2 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 64 Kbytes starting at address $000000.

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

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

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

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

IPL = %000. The chip select is enabled during both data- and program-space accesses.

AVEC = %0000. For read/write cycles, this field is cleared to zero.

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 SCIM 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 vector table and 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 SCIM.

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.

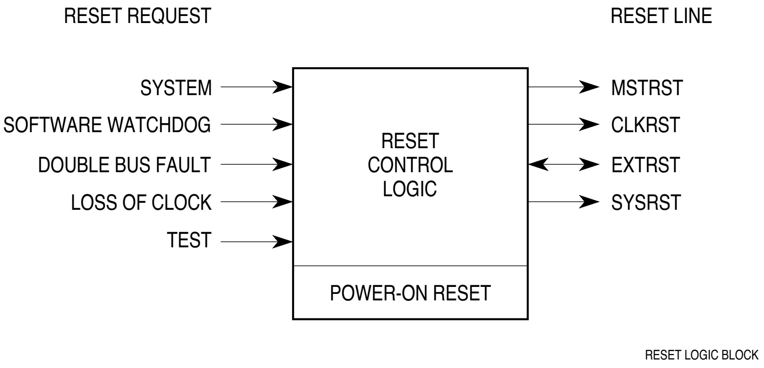


Figure 8-1 Reset Block Diagram

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, which is 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
$\overline{\text{HALT}}$	Halt Monitor	Asynch	Internal $\overline{\text{HALT}}$ Assertion (e.g., 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 reset 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 $\overline{\text{HALT}}$ Reset

When the HLT bit in the RSR is set, the most recent reset was caused by the halt monitor in the system protection module. This type of reset results when the HME (halt monitor enable) bit in the SYPCR is set and the halt monitor detects assertion of $\overline{\text{HALT}}$ on the internal bus (caused by a double bus fault). 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 the loss of a clock signal. This reset condition can exist only if the RSTEN bit in the synthesizer control register (SYNCR) is set and the LOSCD bit is cleared. 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 SCIM 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 the 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	8	7	6	5	4	3	2	1	0
NOT USED		EXT	POW	SW	HLT	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.

HLT — Halt Monitor Reset

Reset was caused by the system protection module halt monitor.

LOC — Loss-of-Clock Reset

Reset was caused by loss of a system clock 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 SCIM 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. Notice in **Figure 8-2** that the SCIM counts down 512 CLKOUT cycles during all types of reset but first waits for the VCO to lock only during power-on reset.

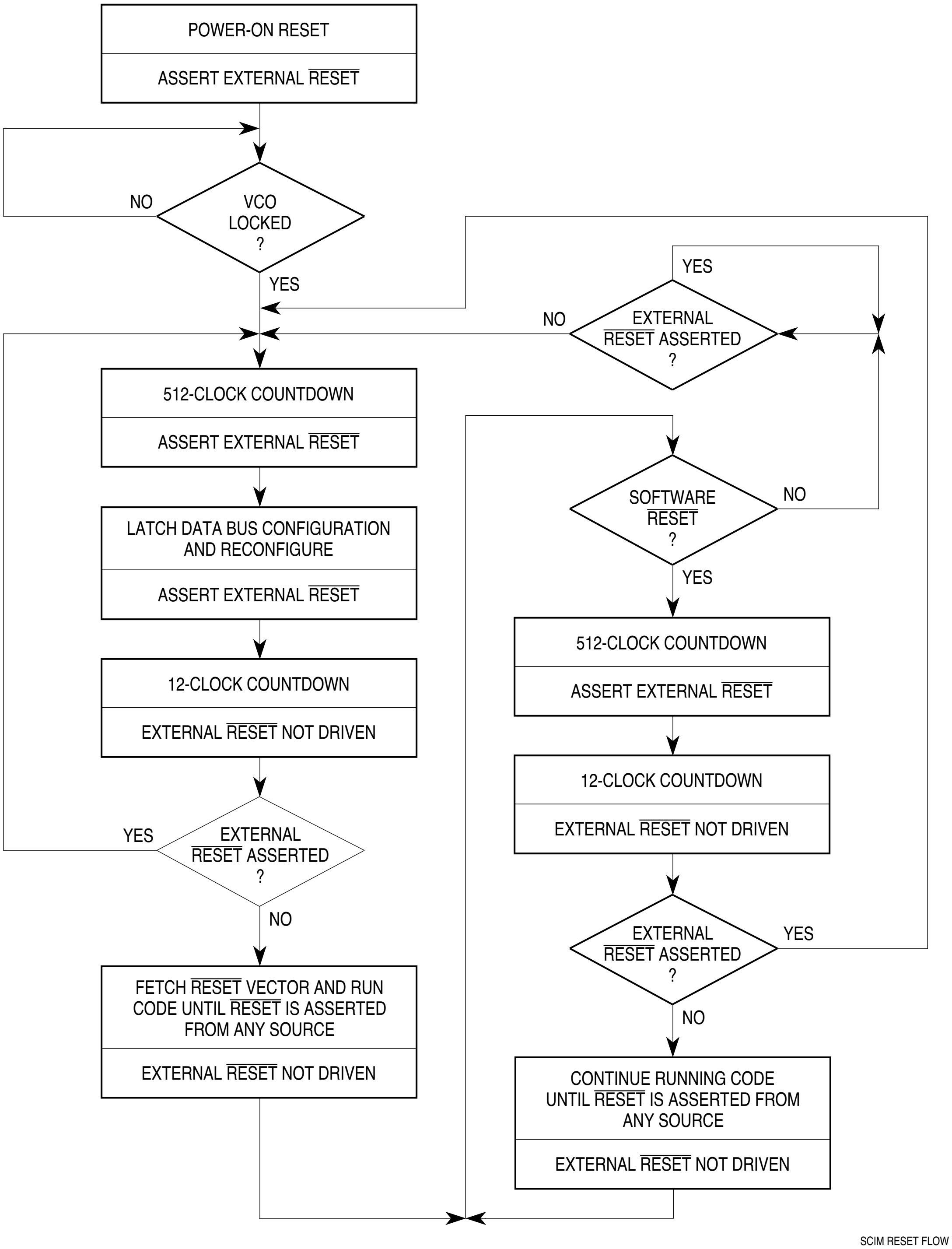

POWER-ON RESET
ASSERT EXTERNAL RESET
NO
VCO LOCKED ?
YES
YES
NO
EXTERNAL RESET ASSERTED ?
512-CLOCK COUNTDOWN
ASSERT EXTERNAL RESET
LATCH DATA BUS CONFIGURATION AND RECONFIGURE
ASSERT EXTERNAL RESET
12-CLOCK COUNTDOWN
EXTERNAL RESET NOT DRIVEN
YES
EXTERNAL RESET ASSERTED ?
NO
FETCH RESET VECTOR AND RUN CODE UNTIL RESET IS ASSERTED FROM ANY SOURCE
EXTERNAL RESET NOT DRIVEN
SOFTWARE RESET ?
NO
YES
512-CLOCK COUNTDOWN
ASSERT EXTERNAL RESET
12-CLOCK COUNTDOWN
EXTERNAL RESET NOT DRIVEN
EXTERNAL RESET ASSERTED ?
YES
NO
CONTINUE RUNNING CODE UNTIL RESET IS ASSERTED FROM ANY SOURCE
EXTERNAL RESET NOT DRIVEN
SCIM RESET FLOW

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 SCIM 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 the clock source, voltage must be applied to the 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 SCIM Operation During Power-On Reset

During power-on reset, an internal circuit in the SCIM 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 SCIM pins are initialized to the values shown in **Table 8-3**. When V_{DD} reaches the minimum operating voltage, the R/C oscillator (alternate clock) begins operation and serves as the system clock until edges are detected on the EXTAL pin. (Refer to **4.8 Loss of Clock** for more information.) 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.

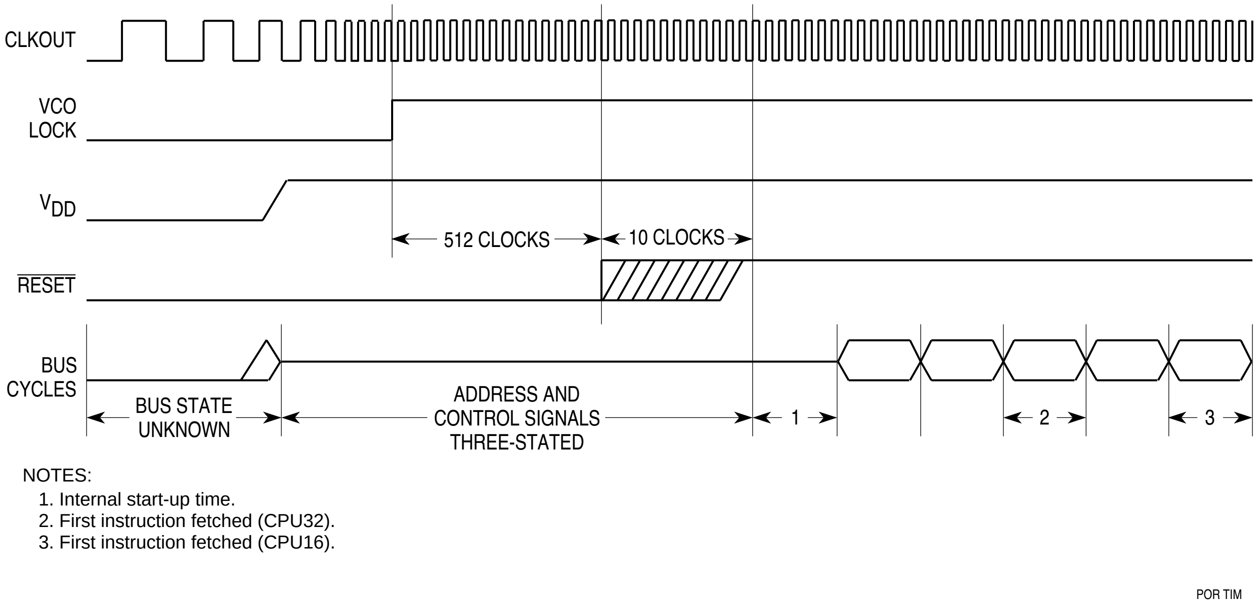

CLKOUT
VCO LOCK
VDD
RESET
BUS CYCLES
512 CLOCKS
10 CLOCKS
BUS STATE UNKNOWN
ADDRESS AND CONTROL SIGNALS THREE-STATED
1
2
3
NOTES:
1. Internal start-up time.
2. First instruction fetched (CPU32).
3. First instruction fetched (CPU16).
POR TIM

Figure 8-3 Power-On Reset Timing

8.4.2 Other Modules During Power-On Reset

The clock synthesizer in the SCIM 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 SCIM 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 an inactive, high-impedance state. The TSC, an active-high input, does not have an internal pull-down and must be tied low when not in use.

TSC must remain asserted for ten clock cycles for drivers to change state. During power-up reset, when the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 ms 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 ten 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 pins during reset determine SCIM operating configuration. During reset, the SCIM reads pin configuration from DATA[11:0], internal module configuration from DATA[15:12], and basic operating information from $\overline{\text{BERR}}$, MODCLK, and $\overline{\text{BKPT}}$. The data pins are normally pulled high internally during reset, causing the MCU to default to a specific configuration. The user must drive the desired data pins low during reset to achieve alternate configurations. $\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and MODCLK do not have internal pull-ups and must be driven to the desired state during reset.

Basic operating options include system clock selection, background mode disable/enable, and external bus configuration. The following external bus configurations are supported:

- Fully-expanded operation with an M68020-style 24-bit address bus and 16-bit data bus with chip selects
- Single-chip operation with no external address or data bus
- Partially-expanded operation with a 24-bit address bus and an 8-bit external data bus.

Table 8-2 summarizes basic configuration options.

Table 8-2 Basic Configuration Options

Select Pin	Pin High at Reset	Pin Low at Reset
MODCLK	Synthesized System Clock	External System Clock
$\overline{\text{BKPT}}$	Background Mode Disabled	Background Mode Enabled
$\overline{\text{BERR}}$	Expanded Mode	Single-Chip Mode
DATA1 (if $\overline{\text{BERR}}$ = 1)	8-Bit Expanded Mode	16-Bit Expanded Mode

When $\overline{\text{BERR}}$ is high during reset, the MCU is configured for partially or fully expanded operation. DATA1 is then decoded to select 8- or 16-bit data bus operation, DATA8 is decoded to configure pins for bus control or port E operation, and DATA9 is decoded to configure pins for interrupt requests or port F operation. If DATA1 is held low at reset, selecting 16-bit data bus operation, DATA11, DATA[7:2] and DATA0 are also decoded. The following subsections explain the process in greater detail.

8.6.1 Address and Data Bus Pin Functions

External bus configuration determines whether certain address and data pins are used for those functions or for general-purpose I/O. ADDR[18:3] serve as pins for ports A and B when the MCU is operating in single-chip mode. DATA[7:0] serve as port H pins in partially expanded and single-chip modes, and DATA[15:8] serve as port G pins during single-chip operation. **Table 8-3** summarizes bus and port configuration.

Table 8-3 Bus and I/O Port Pin Functions

	Mode-Select Pins		**Address Bus/Data Bus/Port Distribution**		
Bus Configuration	**$\overline{\text{BERR}}$**	**DATA1**	**Address Bus Pins [18:3]**	**Data Bus Pins [15:0]**	**I/O Ports**
16-Bit Expanded	1	0	ADDR[18:3]	DATA[15:0]	—
8-Bit Expanded	1	1	ADDR[18:3]	DATA[15:8]	DATA[7:0] = Port H
Single Chip	0	X	None	None	ADDR[18:11] = Port A ADDR[10:3] = Port B DATA[15:8] = Port G DATA[7:0] = Port H

ADDR[2:0] are normally placed in a high-impedance state in single-chip mode and function as normal address bus pins in the expanded modes. Refer to the discussion of the ABD bit in **3.1.7 SCIM Configuration Register** for more information.

The ADDR[23:19] pins can also be used as chip selects or discrete output pins, depending on the external bus configuration selected at reset. The following paragraphs summarize pin configuration options for each external bus configuration.

8.6.2 Pin Configuration for 16-Bit Data Bus Operation

16-bit data bus operation is selected when $\overline{\text{BERR}}$ = 1 and DATA1 = 0 during reset. In this configuration, pins ADDR[18:3] and DATA[15:0] are configured as address and data pins, respectively. The alternate functions for these pins as ports A, B, G, and H are unavailable. ADDR[23:20] can be configured as chip selects or address bus pins, as shown in **Table 8-4**. ADDR[2:0] are configured as address bus pins.

DATA0 determines the port size 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. 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 **7.10 Chip-Select Reset Operation** for more information.

DATA2 determines the functions of $\overline{\text{CS0}}$/BR, $\overline{\text{CS3}}$/FC0, and $\overline{\text{CS5}}$/FC2.

DATA[7:3] determine the functions of ADDR[23:19]/$\overline{\text{CS[10:6]}}$. One of these data bus pins pulled low selects the 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 configured as address bus signals ADDR[21:19], and CS[10:9] are configured as chip selects. (On CPU16-based MCUs, ADDR[23:20] follow the state of ADDR19, and DATA[7:4] have limited use.) Refer to **7.10 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).

DATA10 assigns the BGACK/$\overline{\text{CSE}}$ and BG/$\overline{\text{CSM}}$ pins as either bus grant signals or special chip selects. Refer to **8.6.5 Reset Configuration for Masked ROM and SCIM Port Emulation** for additional information.

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

Table 8-4 summarizes pin function options for 16-bit data bus operation.

Table 8-4 16-Bit Data Bus Mode Reset Pin Configuration

Pin(s) Affected	Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
$\overline{\text{CSBOOT}}$	DATA0	$\overline{\text{CSBOOT}}$ 16-Bit	$\overline{\text{CSBOOT}}$ 8-Bit
$\overline{\text{BR}}$/$\overline{\text{CS0}}$ FC0/$\overline{\text{CS3}}$ FC1/PC1 FC2/$\overline{\text{CS5}}$/PC2	DATA2	$\overline{\text{CS0}}$ $\overline{\text{CS3}}$ $\overline{\text{FC1}}$ $\overline{\text{CS5}}$	BR FC0 FC1 FC2
ADDR19/$\overline{\text{CS6}}$/PC3 ADDR20/$\overline{\text{CS7}}$/PC4 ADDR21/$\overline{\text{CS8}}$/PC5 ADDR22/$\overline{\text{CS9}}$/PC6 ADDR23/$\overline{\text{CS10}}$/ECLK	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]
$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1 $\overline{\text{AVEC}}$/PE2 PE3 $\overline{\text{DS}}$/PE4 $\overline{\text{AS}}$/PE5 $\overline{\text{SIZ0}}$/PE6 $\overline{\text{SIZ1}}$/PE7	DATA8	$\overline{\text{DSACK0}}$ $\overline{\text{DSACK1}}$ $\overline{\text{AVEC}}$ PE3 $\overline{\text{DS}}$ $\overline{\text{AS}}$ $\overline{\text{SIZ0}}$ $\overline{\text{SIZ1}}$	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
MODCLK/PF0 $\overline{\text{IRQ[7:1]}}$/PF[7:1]	DATA9	MODCLK $\overline{\text{IRQ[7:1]}}$	PF0 PF[7:1]
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$ $\overline{\text{BG}}$/$\overline{\text{CSM}}$	DATA10	$\overline{\text{BGACK}}$ $\overline{\text{BG}}$	$\overline{\text{CSE}}$[1] $\overline{\text{CSM}}$[2]
DATA11	DATA11	Slave Mode Disabled[3]	Slave Mode Enabled[3]

NOTES:

1. $\overline{\text{CSE}}$ is enabled when DATA10 and DATA1 = 0 during reset.
2. $\overline{\text{CSM}}$ is enabled when DATA13, DATA10 and DATA1 = 0 during reset.
3. Slave mode is used for factory testing only and is available only in fully-expanded mode.

8.6.3 Pin Configuration for 8-Bit Data Bus Operation

The SCIM uses an 8-bit data bus when $\overline{\text{BERR}}$ = 1 and DATA1 = 1 during reset. In this configuration, pins DATA[7:0] are configured as port H, an 8-bit I/O port. Pins DATA[15:8] are configured as data bus pins, and ADDR[18:3] are configured as address bus pins. The alternate functions for these address and data bus pins as ports A, B, and G are unavailable. ADDR[23:19]/CS[10:6] are configured as chip selects out of reset. (Writing the appropriate value to CSPAR1 can reassign some or all of these pins as address signals.) ADDR[2:0] are configured as address bus pins. Emulator mode is always disabled.

DATA8 determines the function of the DSACK0, DSACK1, AVEC, DS, 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 (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).

Table 8-5 summarizes pin function selection for 8-bit data bus operation.

Table 8-5 8-Bit Data Bus Mode Reset Pin Configuration

Pin(s) Affected	Select Pin	Default Function (Pin Left High)	Alternate Function (Pin Pulled Low)
CSBOOT	N/A[1]	CSBOOT 8-Bit	CSBOOT 8-Bit
BR/CS0 FC0/CS3/PC0 FC1/PC1 FC2/CS5/PC2	N/A[1]	CS0 CS3 FC1 CS5	CS0 CS3 FC1 CS5
ADDR19/CS6/PC3 ADDR20/CS7/PC4 ADDR21/CS8/PC5 ADDR22/CS9/PC6 ADDR23/CS10/ECLK	N/A[1]	CS[10:6]	CS[10:6]
DSACK0/PE0 DSACK1/PE1 AVEC/PE2 PE3 DS/PE4 AS/PE5 SIZ0/PE6 SIZ1/PE7	DATA8	DSACK0 DSACK1 AVEC PE3 DS AS SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
MODCLK/PF0 IRQ[7:1]/PF[7:1]	DATA9	MODCLK IRQ[7:1]	PF0 PF[7:1]
BGACK/CSE BG/CSM	N/A[1]	BGACK BG	BGACK BG

NOTES:

1. These pins have only one reset configuration in 8-bit expanded mode.

8.6.4 Pin Configuration for Single-Chip Operation

Single-chip operation is selected when BERR = 0 during reset. In single-chip configuration, pins DATA[15:0] are configured as two 8-bit I/O ports, ports G and H. ADDR[18:3] are configured as two 8-bit I/O ports, ports A and B. There is no external data bus path. Expanded mode configuration options are not available: I/O ports A, B, C, E, F, G, and H are always selected. ADDR[2:0] come out of reset in a high-impedance state. After reset, clearing the ABD bit in the SCIM configuration register enables these pins, and leaving the bit set (its single-chip reset state) leaves the pins in a disabled (high-impedance) state. Refer to the discussion of the ABD bit in **3.1.7 SCIM Configuration Register** for more information.

CAUTION

$\overline{\text{BERR}}$ should be continually tied low while the MCU is operating in single-chip mode. This way, if the CPU inadvertently attempts to access a memory location that is not implemented within the MCU, the resulting external bus cycle will be terminated with $\overline{\text{BERR}}$.If $\overline{\text{BERR}}$ is not tied low and the CPU attempts an access to a location outside the MCU, the results are unpredictable.

Table 8-6 summarizes SCIM pin function during single-chip operation.

Table 8-6 Single-Chip Mode Reset Pin Configuration

Pin(s) Affected	Function
$\overline{\text{CSBOOT}}$	$\overline{\text{CSBOOT}}$ 16-Bit
ADDR[18:10]	PA[7:0]
ADDR[9:3]	PB[7:0]
BR/$\overline{\text{CS0}}$	$\overline{\text{CS0}}$
FC0/$\overline{\text{CS3}}$/PC0 FC1/PC1 FC2/$\overline{\text{CS5}}$/PC2 ADDR19/$\overline{\text{CS6}}$/PC3 ADDR20/$\overline{\text{CS7}}$/PC4 ADDR21/$\overline{\text{CS8}}$/PC5 ADDR22/$\overline{\text{CS9}}$/PC6	PC[6:0]
ADDR23/$\overline{\text{CS10}}$/ECLK	—
$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1 $\overline{\text{AVEC}}$/PE2 PE3 $\overline{\text{DS}}$/PE4 $\overline{\text{AS}}$/PE5 $\overline{\text{SIZ0}}$/PE6 $\overline{\text{SIZ1}}$/PE7	PE[7:0]
MODCLK/PF0 $\overline{\text{IRQ[7:1]}}$/PF[7:1]	PF0 PF[7:1]
DATA[15:8]	PG[7:0]
DATA[7:0]	PH[7:0]
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$ $\overline{\text{BG}}$/$\overline{\text{CSM}}$	$\overline{\text{BGACK}}$ $\overline{\text{BG}}$

8.6.5 Reset Configuration for Masked ROM and SCIM Port Emulation

Emulator mode is a special type of 16-bit expanded bus operation. It is entered by holding DATA10 low, $\overline{\text{BERR}}$ high, and DATA1 low during reset. In emulator mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally, and $\overline{\text{CSE}}$ is asserted whenever an access to one of these locations is made. In addition, memory chip select signal $\overline{\text{CSM}}$ is asserted whenever a valid access to an address assigned to the masked ROM array is made. Refer to **7.9 Emulation-Support Chip Selects** for more information.

8.6.6 Holding Data Bus Pins Low At Reset

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. 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. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for internal pull-up specifications.

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 ten 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.

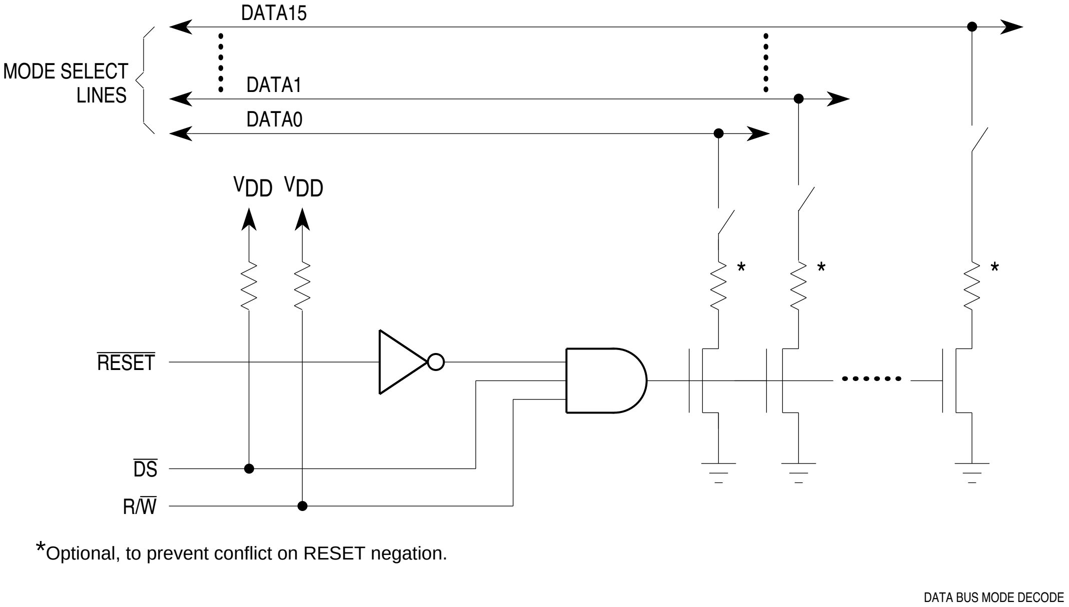


Figure 8-4 Data Bus Signal Conditioning

8.6.7 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, it should be driven to the desired reset state with an active device, so that bus loading does not result in incorrect clock mode selection.

8.6.8 Breakpoint Mode Selection

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

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

If both $\overline{\text{BKPT}}$ inputs are at logic level one when sampled, breakpoint exception processing begins whenever either $\overline{\text{BKPT}}$ 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

A pin driver can be in an active, inactive, or disabled (high-impedance) state while reset occurs. During power-on reset, pin state is subject to the constraints discussed in **8.4 Power-On 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} current caused by digital inputs floating near mid-supply level.

While the MCU is held in reset, the data bus pins are configured as inputs, and the address bus, function code, and bus control pins ($\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, SIZ[1:0], and $\overline{\text{RMC}}$) are driven high. Any address bus, data bus, or bus control pins configured for I/O (ports A, B, G, H, E, and F) are configured as inputs when the SCIM comes out of reset.

After $\overline{\text{RESET}}$ is released, mode selection occurs, and reset exception processing begins. Pins configured as inputs during reset become active high-impedance loads after $\overline{\text{RESET}}$ 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 $\overline{\text{RESET}}$ is released.

Table 8-7 is a summary of SCIM pin states during reset. The function of many SCIM pins out of reset depends on the bus configuration selected and the reset state of the relevant data bus mode select pin. Pin state out of reset is shown for each pin function where appropriate. The pin function shown on the left is the default function for the 16-bit expanded bus configuration.

Table 8-7 SCIM Pin Reset States

Mnemonic	State While $\overline{\text{RESET}}$ Asserted	Pin State After $\overline{\text{RESET}}$ Released[1]			
		Pin Function	Pin State	Pin Function	Pin State
$\overline{\text{CS10}}$/ADDR23	1	$\overline{\text{CS10}}$	1	ADDR23	Output
$\overline{\text{CS[9:6]}}$/ ADDR[22:19]/ PC[6:3]	1	$\overline{\text{CS[9:6]}}$	1	ADDR[22:19]	Output
ADDR[18:0]	High-Z Output	ADDR[18:0]	Output	ADDR[18:0]	Output
$\overline{\text{AS}}$/PE5	See Note 2	$\overline{\text{AS}}$	Output	PE5	Input
$\overline{\text{AVEC}}$/PE2	Inactive Input	$\overline{\text{AVEC}}$	Input	PE2	Input
$\overline{\text{BERR}}$	Mode Select	$\overline{\text{BERR}}$	Input	$\overline{\text{BERR}}$	Input
$\overline{\text{CSM}}$/$\overline{\text{BG}}$	1	$\overline{\text{CSM}}$	1	$\overline{\text{BG}}$	1
$\overline{\text{CSE}}$/$\overline{\text{BGACK}}$	1	$\overline{\text{CSE}}$	1	$\overline{\text{BGACK}}$	Input
$\overline{\text{CS0}}$/$\overline{\text{BR}}$	1	$\overline{\text{CS0}}$	1	$\overline{\text{BR}}$	Input
CLKOUT	Output	CLKOUT	Output	CLKOUT	Output
$\overline{\text{CSBOOT}}$	1	$\overline{\text{CSBOOT}}$	0[3]	$\overline{\text{CSBOOT}}$	0[3]
DATA[15:0]	Mode Select	DATA[15:0]	Input	DATA[15:0]	Input
$\overline{\text{DS}}$/PE4	Inactive Input	$\overline{\text{DS}}$	Output	PE4	Input
$\overline{\text{DSACK0}}$/PE0	See Note 2	$\overline{\text{DSACK0}}$	Input	PE0	Input
$\overline{\text{DSACK1}}$/PE1	Inactive Input	$\overline{\text{DSACK1}}$	Input	PE1	Input
$\overline{\text{CS5}}$/FC2/PC2	1	$\overline{\text{CS5}}$	1	FC2	Output
FC[2:0]/PC[2:0]	1	FC1	1	FC1	Output
$\overline{\text{CS3}}$/FC[2:0]/PC[2:0]	1	$\overline{\text{CS3}}$	1	FC0	Output
$\overline{\text{HALT}}$	Inactive Input	$\overline{\text{HALT}}$	Input	$\overline{\text{HALT}}$	Input
$\overline{\text{IRQ[7:1]}}$/PF[7:1]	Inactive Input	$\overline{\text{IRQ[7:1]}}$	Input	PF[7:1]	Input
MODCLK/PF0	Mode Select	MODCLK	Input	PF0	Input
R/$\overline{\text{W}}$	High-Z Output	R/$\overline{\text{W}}$	Output	R/$\overline{\text{W}}$	Output
$\overline{\text{RESET}}$	Asserted	$\overline{\text{RESET}}$	Input	$\overline{\text{RESET}}$	Input
$\overline{\text{RMC}}$/PE3	See Note 2	$\overline{\text{RMC}}$	1[4]	PE3	Input
SIZ[1:0]/PE[7:6]	See Note 2	SIZ[1:0]	Output	PE[7:6]	Input
TSC	Mode Select	TSC	Input	TSC	Input

NOTES

1. Pin states are generally valid until the first instruction is executed.
2. 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.
3. Asserted while address and bus control pins satisfy conditions in the chip-select base address and option registers.
4. On CPU-16 based MCUs, the CPU does not drive $\overline{\text{RMC}}$, and the state of the pin depends on mode configuration. If the pin is configured for its $\overline{\text{RMC}}$ function (DATA8 held high during reset), the pin is an output and is driven to logic level one. If the pin is configured for its PE3 function (DATA8 held low during reset), the pin is an input coming out of reset.

8.8 SCIM Registers Out of Reset

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

Table 8-8 SCIM Registers Out of Reset

Register	Bits	Name	Value	Meaning
SCIMCR	15	EXOFF	0	CLKOUT enabled
	14	FRZSW	1	Disable watchdog and PIT when FREEZE asserted
	13	FRZBM	1	Disable bus monitor when FREEZE asserted
	12	CPUD	$\overline{\text{BERR}}$	CPU development support disable
	11	SLVEN	$\overline{\text{DATA11}}$[1]	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
	5	ABD	$\overline{\text{BERR}}$	Disable (place in high-impedance state) address bus pins ADDR[2:0]
	4	RWD	$\overline{\text{BERR}}$	Disable (place in high-impedance state) R/$\overline{\text{W}}$ pin
	[3:0]	IARB	1111	SCIM 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 $*$ 256
	7	EDIV	0	ECLK = System clock ≥ 8
	5	LOSCD	0	Loss-of-clock oscillator disable
	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	STSCIM	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	$\overline{\text{MODCLK}}$[1]	If MODCLK low (external clock), PIT prescaled If MODCLK high (VCO clock), PIT not prescaled
	[7:0]	PITM	00000000	Periodic timer disabled
PORTA	[7:0]	PA	Unchanged	PORTA unaffected by reset
PORTB	[7:0]	PB	Unchanged	PORTB unaffected by reset
DDRAB	[1:0]	DDA, DDB	00	Ports A and B configured as input ports
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	$\overline{\text{DATA8}}$[1]	If DATA8 low, pins configured for G/P I/O If DATA8 high, 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	$\overline{\text{DATA9}}$[1] (expanded mode) 0 (single chip)	If DATA9 = 0 or if single-chip mode, pins configured for G/P I/O without edge detect If DATA9 = 1 in expanded mode, pins configured for interrupt request (MODCLK for PF0)

Table 8-8 SCIM Registers Out of Reset (Continued)

Register	Bits	Name	Value	Meaning
PORTFE	[7:0]	EF	0	Edge-detect flags cleared at reset
PFIVR	[7:0]	PFIV	00001111	Uninitialized interrupt vector
PFLVR	[2:0]	PFLV	000	Disable interrupts
PORTG	[7:0]	PG	Unchanged	Port G pins unaffected by reset
DDRG	[7:0]	DDG	00000000	Port G pins are inputs out of reset
PORTH	[7:0]	PH	Unchanged	Port H pins are unaffected by reset
DDRH	[7:0]	DDH	0000000	Port H pins are inputs out of reset
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[10:5] CSBAR3, CSBAR0	[15:3]	ADDR[23:11]	$0	Base address = 0
	[2:0]	BLKSZ	000	Block size = 2 Kbytes
CSORBT	15	MODE	0	Asynchronous mode
	[14:13]	BYTE	11	Both bytes
	[12:11]	R/$\overline{\text{W}}$	11	Assert chip select for both reads and writes
	10	STRB	0	Chip select synchronized with $\overline{\text{AS}}$
	[9:6]	$\overline{\text{DSACK}}$	1101	13 wait states
	[5:4]	SPACE	11	Supervisor/user space
	[3:1]	IPL	000	Assert chip select on any level interrupt
	0	$\overline{\text{AVEC}}$	0	Autovector disabled
CSBAR[10:5], CSBAR3, CSBAR0	Disabled (BYTE field = 0)			

NOTES:

1. Overbars above active-high signals indicate the complement of the logic level on the pin.

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 SCIM. Initialize the SCIM before initializing other MCU modules.

1. Program the SCIMCR to set the SCIM arbitration level, base address of the module registers, and privilege level of assignable SCIM 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. If applicable, 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 SCIM port data registers. Then program the port 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

The SCIM contains six general-purpose input/output ports: ports A, B, E, F, G, and H. (Port C, an output-only port, is included in the discussion of chip selects.) Ports A, B, and G are available in single-chip mode only, and port H is available in single-chip or 8-bit expanded modes only. Ports E, F, G, and H have an associated data direction register to configure each pin as input or output. Ports A and B share a data direction register that configures each port as input or output. Ports E and F have associated pin assignment registers that configure each pin for digital I/O or an alternate function. Port F has an edge-detect flag register that indicates whether a transition has occurred on any of its pins.

The following table shows the shared functions of the general-purpose I/O ports and the external bus configurations in which the I/O ports are available.

Table 9-1 General-Purpose I/O Ports

Port	Shared Function	Bus Configurations in which Port is Available
A	ADDR[18:11]	Single Chip
B	ADDR[10:3]	Single Chip
E	Bus Control	All
F	$\overline{\text{IRQ[7:1]}}$/MODCLK	All
G	DATA[15:8]	Single Chip
H	DATA[7:0]	Single Chip, 8-Bit Expanded

Accesses to port A, B, E, G, and H data and data direction registers and port E pin assignment register require three clock cycles, to ensure timing compatibility with external port replacement logic. Accesses to port F registers require two clock accesses. Port registers are byte-addressable and are grouped to allow coherent word access to port data register pairs A-B and G-H, as well as word-aligned long word coherency of A-B-G-H port data registers.

If emulator mode is enabled, accesses to ports A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally and cause the emulator chip-select signal ($\overline{\text{CSE}}$) to be asserted. The SCIM does not respond to these accesses, allowing external logic, such as a Motorola port replacement unit, to respond. Port F registers, however, remain accessible.

A write to the port A, B, E, F, G, or H data register is stored in the internal data latch, and if any port pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port 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.

9.1 Ports A and B

Ports A and B are available in single-chip mode only. One data direction register controls data direction for both ports. Port A and B registers can be read or written to anytime the MCU is not in emulator mode.

PORTA — Port A Data Register **$####0A**
PORTB — Port B Data Register **$####0B**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

DDRAB — Port A/B Data Direction Register **$####14**

15	14	13	12	11	10	9	8	7	0
0	0	0	0	0	0	DDA	DDB	DDRE (Port E Data Direction)	
RESET:									
0	0	0	0	0	0	0	0		

DDA and DDB control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

9.2 Port E

Port E can be made available in all external bus configurations. The states of $\overline{\text{BERR}}$ and DATA8 during reset control whether the port E pins are used as bus control signals or discrete I/O lines.

If the MCU is in emulator mode, the SCIM does not respond to accesses to the port E data, data direction, or pin assignment registers (PORTE, DDRE, PEPAR). This allows external port replacement logic to respond to these signals, while giving an emulator access to the bus control signals.

PORTE0, PORTE1— Port E Data Register **$####10, $####12**

15	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

PORTE is a single register that can be accessed in two locations. It can be read or written at any time the MCU is not in emulator mode.

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

15 8	7	6	5	4	3	2	1	0
DDRAB (Port A/B Data Direction Register)	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:								
	0	0	0	0	0	0	0	0

The bits in this register control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. This register can be read or written anytime the MCU is not in emulator mode.

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

15 8	7	6	5	4	3	2	1	0
NOT USED	PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET (Expanded-bus configuration):								
	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8
RESET (Single-chip configuration):								
	0	0	0	0	0	0	0	0

The bits in PEPAR control the function of each port E pin. Any bit set to one defines the corresponding pin to be a bus control signal, with the function shown in **Table 9-2**. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

Table 9-2 Port E Pin Assignments

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

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

$\overline{\text{BERR}}$ and DATA8 control the state of the PEPAR following reset. If either $\overline{\text{BERR}}$ or DATA8 is low during reset, the PEPAR is set to $00, defining all port E pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA8 are both high during reset, the register is set to $FF, and all port E pins are defined to be bus control signals.

9.3 Port F

Port F consists of eight I/O pins, a data register, a data direction register, a pin assignment register, an edge-detect flag register, an edge-detect interrupt vector register, an edge-detect interrupt level register, and associated control logic. Port F pins can be configured as interrupt request inputs, edge-detect inputs or outputs, or discrete inputs or outputs. Port F edge-detect logic can detect rising- or falling-edge transitions.

The port F pin assignment register assigns each pin for rising edge detection, falling edge detection, I/O without edge detection, or for external interrupt requests. The port F data direction register (DDRF) assigns each pin configured for I/O (with or without edge detection) as an input or output. The port F data register (PORTF0, PORTF1) returns the logic level of input pins when read and drives the specified logic level onto output pins when written to.

The port F edge-detect flag register (PORTFE) indicates when the proper transition has occurred on a port F pin that is configured as an input or output edge-detect pin. Optionally, an interrupt can be generated whenever the proper transition has been detected. When the interrupt is serviced, the interrupt service routine can read PORTFE to determine which pin transition caused the interrupt.

To enable interrupt detection for port F edge-detect pins, assign an interrupt priority level greater than zero to the port F interrupt level register (PFLVR), and write the interrupt vector number to the port F interrupt vector register (PFIVR).

Port F edge-detect interrupts have the lowest arbitration priority in the SCIM: an interrupt request of the same level from either the periodic interrupt timer or a port F pin configured as an interrupt request ($\overline{\mathrm{IRQ}[7:1]}$) has priority over an edge-detect interrupt request from the port F control logic.

Depending on the state of $\overline{\mathrm{BERR}}$ and DATA9 during reset, port F pins come out of reset as either interrupt request inputs or discrete inputs. These registers are unaffected by the CPU32 RESET instruction.

Figure 9-1 is a block diagram of port F pins, registers, and control logic.

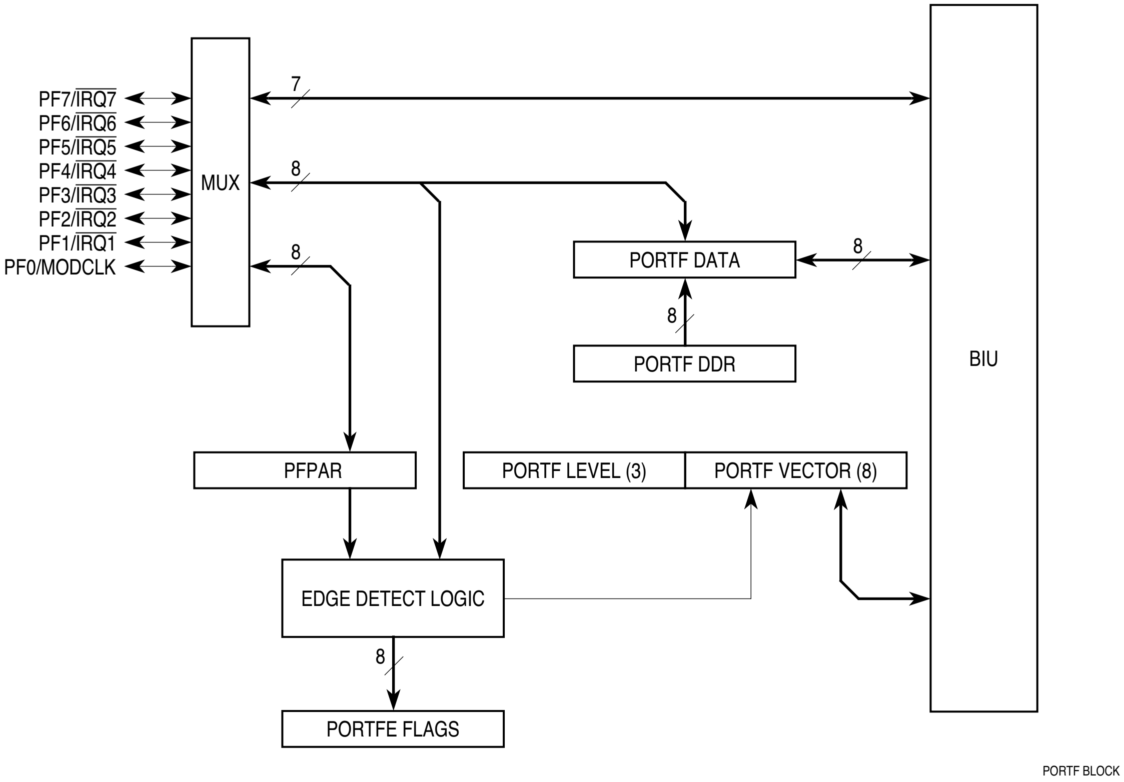


Figure 9-1 Port F Block Diagram

PORTF0, PORTF1 — Port F Data Register **$####18, $####1A**

15 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

A write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of PORTF returns the value on a pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data register.

Port F is a single register that can be accessed in two locations. It can be read or written at any time, including when the MCU is in emulator mode.

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

15 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

The bits in the DDRF control the direction of port F pin drivers when the pins are configured for I/O. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding pin as an input.

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

15	8	7	6	5	4	3	2	1	0
NOT USED		PFPA3		PFPA2		PFPA1		PFPA0	
RESET (Expanded-bus configuration):									
		DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
RESET (Single-chip configuration):									
		0	0	0	0	0	0	0	0

The fields in the PFPAR determine the functions of pairs of port F pins as shown in **Table 9-3** and **Table 9-4**. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register. If either $\overline{\text{BERR}}$ or DATA9 is low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins except MODCLK/PF0 to be interrupt signals.

Table 9-3 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ[7:6]}}$
PFPA2	PF[5:4]	$\overline{\text{IRQ[5:4]}}$
PFPA1	PF[3:2]	$\overline{\text{IRQ[3:2]}}$
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, MODCLK*

*MODCLK signal is recognized only during reset

Table 9-4 PFPAR Pin Encodings

PFPA Bits	Port F Signal
00	I/O pin without edge detection
01	Rising edge detection
10	Falling edge detection
11	Interrupt request (or MODCLK)

PORTFE — Port F Edge-Detect Flag Register $####28

15	8	7	6	5	4	3	2	1	0
NOT USED		EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
RESET:									
		0	0	0	0	0	0	0	0

When a pin is configured in the PFPAR to detect either a rising or falling edge, and such an edge is detected, the corresponding bit in the PORTFE is set. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state.

PFIVR — Port F Edge-Detect Interrupt Vector Register **$####2A**

15	8	7	6	5	4	3	2	1	0
NOT USED		PFIV7	PFIV6	PFIV5	PFIV4	PFIV3	PFIV2	PFIV1	PFIV0
RESET:									
		0	0	0	0	1	1	1	1

PFIVR determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIV[7:0] to the value pointing to the appropriate interrupt vector. Refer to the appropriate CPU reference manual for interrupt vector assignments.

PFLVR — Port F Edge-Detect Interrupt Level Register **$####2C**

15	8	7	6	5	4	3	2	1	0
NOT USED		0	0	0	0	0	PFLV2	PFLV1	PFLV0
RESET:									
		0	0	0	0	0	0	0	0

PFLVR determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts within the SCIM with the same interrupt request level enter interrupt arbitration, the port F edge-detect interrupt has the lowest arbitration priority.

9.4 Port G

Port G is available during single-chip operation only. During single-chip operation, these pins are always configured for general-purpose I/O. There is no pin assignment register associated with port G.

PORTG — Port G Data Register **$####0C**

15	14	13	12	11	10	9	8	7	0
PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PORTH (Port H Data Register)	
RESET:									
U	U	U	U	U	U	U	U		

PORTG can be read or written anytime the MCU is not in emulator mode.

DDRG — Port G Data Direction Register **$####0E**

15	14	13	12	11	10	9	8	7	0
DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDRH (Port H Data Direction Register)	
RESET:									
0	0	0	0	0	0	0	0		

The bits in the DDRG control the direction of the pin drivers when the pins are configured as I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

9.5 Port H

Port H is available during single-chip and partially expanded operation only. The function of these pins is determined by external bus configuration; there is no pin assignment register associated with this port.

PORTH — Port H Data Register **$####0D**

15	8	7	6	5	4	3	2	1	0
PORTG (Port G Data Register)		PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
RESET:									
		U	U	U	U	U	U	U	U

PORTH can be read or written anytime the MCU is not in emulator mode. Reset has no effect.

DDRH — Port H Data Direction Register **$####0E**

15	8	7	6	5	4	3	2	1	0
DDRG (Port G Data Direction Register)		DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0
RESET:									
		0	0	0	0	0	0	0	0

The bits in the DDRH control the direction of the pin drivers when the pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

SECTION 10 REDUCED PIN-COUNT SCIM

Some Motorola MCUs contain a version of the SCIM with a reduced pin count. These MCUs always contain the SCIM pins that are most basic to MCU operation. Some SCIM 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 SCIM-based MCUs and discusses MCU operation with a reduced pin-count SCIM.

10.1 Optional SCIM Pins

Table 10-1 lists the pins that may or may not be included on MCUs with a reduced pin-count SCIM. These MCUs may still have some of the optional 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 that chip.

Table 10-1 Optional SCIM Pins

Pin Mnemonic	**Description**	**Port Designation**
ADDR[22:20]/CS[9:7]	Address Bus/Chip Selects	PC[6:4]
AVEC	Autovector	PE2
CSBOOT	Boot Chip Select	—
DSACK0	Data and Size Acknowledge	PE0
HALT	Halt	—
IRQ[5:1]	Interrupt Request Level	PF[5:1]
RMC	Read-Modify-Write Cycle	PE3

10.2 DSACK0/PE0 Pin

On MCUs with both DSACK[1:0] pins, an external device asserts the appropriate DSACK signal to indicate port size and the availability of data. On some MCUs with a reduced pin-count SCIM, no DSACK0 pin is provided. With these MCUs, an external device indicates the availability of data by asserting DSACK1, regardless of port size. All accesses thus appear to be word accesses.

SECTION 5 EXTERNAL BUS INTERFACE explains the operation of the external bus. **10.2.1 Data Transfers Involving a 16-Bit Port** and **10.2.2 Data Transfers Involving an 8-Bit Port** explain how this operation is modified on MCUs with no DSACK0 pin.

NOTE

Use of SCIM chip selects simplifies system interfacing. When SCIM chip selects are used to generate DSACK signals internally, external DSACK generation is unnecessary. For applications that might require external DSACK generation, however, the following paragraphs describe the procedures to follow when no DSACK0 pin is available.

When the $\overline{\text{DSACK0}}$/PE0 pin is configured for discrete I/O in the port E pin assignment register and PE0 is configured as an input in the port E data direction register, a read of PE0 in the port E data register returns one.

10.2.1 Data Transfers Involving a 16-Bit Port

Data transfers involving a 16-bit port proceed in the same way on SCIM-based MCUs with a reduced pin count as on SCIM-based MCUs with a full pin count. (Refer to **5.5 Operand Transfer Cases**.) The MCU asserts the appropriate control and address lines and places the data (during a write cycle) or expects the data (during a read cycle) on the appropriate byte or bytes of the data bus. The external device asserts $\overline{\text{DSACK1}}$ to signal a 16-bit port size. To transfer data to or from a 16-bit port, the external data bus must be fully expanded.

Table 10-2 summarizes operand transfer cases for MCUs without a $\overline{\text{DSACK0}}$ pin. Since the external device cannot assert the $\overline{\text{DSACK0}}$ pin to signal an 8-bit port, only transfers involving 16-bit ports are included.

Table 10-2 Operand Transfer Cases with Reduced Pin-Count SCIM

Num	Transfer Case	SIZ[1:0]	ADDR0	Read Cycles		Write Cycles		Next Cycle
				DATA [15:8]	DATA [7:0]	DATA [15:8]	DATA [7:0]	
1	Byte to 16-bit Port (Even)[1]	01	0	OP0	—	OP0	(OP0)	—
2	Byte to 16-bit Port (Odd)[1]	01	1	—	OP0	(OP0)	OP0	—
3	Word to 16-bit Port (Aligned)[1]	10	0	OP0	OP1	OP0	OP1	—
4	Word to 16-bit Port (Misaligned)[1,2]	10	1	—	OP0	(OP0)	OP0	1
5	Long Word to 16-bit Port (Aligned)[1]	00	0	OP0	OP1	OP0	OP1	3
6	Long Word to 16-bit Port (Misaligned)[1,2,3]	10	1	—	OP0	(OP0)	OP0	1

NOTES:

1. Transfers involving 16-bit ports are supported only in fully-expanded (16-bit data bus) mode.
2. The CPU32 does not support misaligned transfers.
3. The CPU16 treats misaligned long-word transfers as two misaligned-word transfers.

10.2.2 Data Transfers Involving an 8-Bit Port

To transfer data between an 8-bit port and an MCU with no $\overline{\text{DSACK0}}$ pin, use of SCIM chip selects is strongly recommended. Use the following procedures only if using SCIM chip selects is not possible.

10.2.2.1 Byte Transfers Involving an 8-Bit Port

To transfer data between an MCU with no $\overline{\text{DSACK0}}$ pin and an 8-bit register in an 8-bit device, 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.

This procedure works only for even-numbered byte addresses, however. Since the external device asserts $\overline{\text{DSACK1}}$, which normally signals a 16-bit port, the MCU expects data from an 8-bit port with an odd address to be placed on DATA[7:0]. These pins, however, are not connected to the external device.

10.2.2.2 Word Transfers Involving an 8-Bit Port

A word transfer involving an MCU with no $\overline{\mathrm{DSACK0}}$ pin and an 8-bit port (for example, a read of a 16-bit timer register value in an 8-bit device) is performed without difficulty using SCIM chip selects. To perform the transfer without using SCIM chip selects requires using one of the following work-arounds.

For CPU16-based products, connect the peripheral to DATA[15:8]. To write a word to an 8-bit port, store the upper byte of the word in the upper byte of accumulator E and the lower byte of the word in the upper byte of accumulator D, then issue the STED (store concatenated E and D) instruction. To read a word from an 8-bit port, issue the LDED (load concatenated E and D) instruction, then arrange the upper bytes of accumulators E and D into a word. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for additional details.

For CPU32-based products, connect the peripheral to DATA[15:8]. Use the MOVEP (move peripheral data) instruction to move a word or long-word operand between a data register and alternate even bytes within the address space. Refer to the *CPU32 Reference Manual* (CPU32RM/AD) for details.

CAUTION

For both CPU16- and CPU32-based MCUs with no $\overline{\mathrm{DSACK0}}$ pin, a word read of an 8-bit port places the logic level of the DATA[7:0] pins onto the lower half of the internal data bus, regardless of how the pins are being used.

10.3 ADDR[22:20]/$\overline{\mathrm{CS[9:7]}}$/PC[6:4] Pins

As many as three address bus/chip select pins (ADDR[22:20]/$\overline{\mathrm{CS[9:7]}}$/PC[6:4]) may not be used on MCUs with a reduced pin-count SCIM. On MCUs that lack these pins, use the remaining address lines to access external devices. Alternately, use SCIM chip selects to automatically decode the upper address lines. Refer to **SECTION 7 CHIP SELECTS** for details.

If $\overline{\mathrm{CS[9:7]}}$ are unavailable, use the remaining chip selects ($\overline{\mathrm{CS10}}$, $\overline{\mathrm{CS[6:5]}}$, $\overline{\mathrm{CS3}}$, and $\overline{\mathrm{CS0}}$). In addition, the chip select registers for $\overline{\mathrm{CS[9:7]}}$ can still be programmed to assert $\overline{\mathrm{DSACK}}$ or $\overline{\mathrm{AVEC}}$ internally during interrupt-acknowledge cycles, even if the external pins are not available.

If the ADDR[22:20]/$\overline{\mathrm{CS[9:7]}}$/PC[6:4] pins are not available, then the discrete output function for PC[6:4] is not supported.

10.4 $\overline{\mathrm{RMC}}$ Pin

The CPU32 asserts the $\overline{\mathrm{RMC}}$ 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.

10.5 $\overline{\text{AVEC}}$/PE2 Pin

The $\overline{\text{AVEC}}$ input signal requests that the SCIM generate the interrupt vector internally during an interrupt-acknowledge cycle. On MCUs that do not have an $\overline{\text{AVEC}}$ pin, SCIM chip-select circuits can be used to assert the $\overline{\text{AVEC}}$ signal internally.

When PE2 is configured as an input in the port E data direction register, a read of PE2 in the port E data register returns one.

10.6 $\overline{\text{CSBOOT}}$

The $\overline{\text{CSBOOT}}$ signal selects a boot ROM chip containing a reset vector and an initialization program. On MCUs that do not have a $\overline{\text{CSBOOT}}$ pin, if the reset vector and initialization program are located off chip, external logic can decode the address lines to map boot ROM to external address space starting at \$000000. At reset, the CPU reads the reset vector beginning at that address.

10.7 $\overline{\text{IRQ[5:1]}}$/PF[5:1]

The $\overline{\text{IRQ[5:1]}}$ inputs signal the MCU that an external device is requesting interrupt service. The interrupt request level corresponds to the number of the $\overline{\text{IRQ}}$ pin. On MCUs without $\overline{\text{IRQ[5:1]}}$, the interrupting device must assert the $\overline{\text{IRQ7}}$ or $\overline{\text{IRQ6}}$ pin to request interrupt service. If the interrupt level of the external device (seven or six) is the same as that of an internal module, the IARB fields in the respective module configuration registers assign priorities among the modules.

When any of the $\overline{\text{IRQ[5:1]}}$/PF[5:1] pins are configured for input in the port F data direction register, reads of the associated bits in the port E data register return ones.

10.8 $\overline{\text{HALT}}$ Pin

The bidirectional $\overline{\text{HALT}}$ signal is asserted to suspend external bus activity, to request a retry when used with $\overline{\text{BERR}}$, or for single-step operation. As an output, $\overline{\text{HALT}}$ indicates a double bus fault by the CPU. On MCUs without a $\overline{\text{HALT}}$ pin, these functions are not available.

APPENDIX A ELECTRICAL CHARACTERISTICS

This appendix contains electrical specification tables and reference timing diagrams. Each timing diagram has an associated key table made up of parameters abstracted from the specification tables. Pertinent notes have been included in the key tables.

Table A-1 Clock Control Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ±10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H,
32.768 kHz or 4.194 MHz reference)

Num	Characteristic	Symbol	Min	Max	Unit
1	PLL Reference Frequency Range	f_{ref}			
	4.194 MHz Reference		3.2	4.2	MHz
	32.768 kHz Reference		25	50	kHz
2	System Frequency[1]		dc	16.78	
	On-Chip PLL Frequency	f_{sys}	0.131	16.78	MHz
	External Clock Operation		dc	16.78	
3	PLL Lock Time[2,3,4,5]	t_{lpll}	—	20	ms
4	Limp Mode Clock Frequency[6]	f_{limp}			MHz
	SYNCR X bit = 0		—	f_{sys} max /2	
	SYNCR X bit = 1		—	f_{sys} max	
5	CLKOUT Stability[2,3,4,7]				%
	Short term (5 µs interval)	C_{stab}	–0.5	0.5	
	Long term (500 µs interval)		–0.05	0.05	

NOTES:

1. All internal registers retain data at 0 Hz.
2. This parameter is periodically sampled rather than 100% tested.
3. Assumes that a low-leakage external filter capacitor with a value of 0.1 µF is attached to the XFC pin. Total external resistance from the XFC pin due to external leakage sources must be greater than 15 MΩ to guarantee this specification.
4. Proper layout procedures must be followed to achieve specifications.
5. Assumes that stable V_{DDSYN} is applied, and that the crystal oscillator is stable. Lock time is measured from the time V_{DD} and V_{DDSYN} are valid 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.
6. Determined by the default frequency of the on-chip RC oscillator. The X bit in SYNCR controls a divide by two scaler on the system clock output.
7. Stability is the average deviation from the programmed frequency measured over the specified interval at maximum f_{sys}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via V_{DDSYN} and V_{SS} and variation in crystal oscillator frequency increase the C_{stab} percentage for a given interval.

Table A-2 DC Characteristics

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
1	Input High Voltage	V_{IH}	0.7 (V_{DD})	V_{DD} + 0.3	V
2	Input Low Voltage	V_{IL}	V_{SS} – 0.3	0.2 (V_{DD})	V
3	Input Hysteresis[1,6,7]	V_{HYS}	0.5	—	V
4	Input Leakage Current[2] $V_{in} = V_{DD}$ or V_{SS} Input-only pins	I_{in}	–2.5	2.5	µA
5	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
6	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
7	CMOS Output Low Voltage[2] I_{OL} = 10.0 µA Group 1,2 input/output and all output pins	V_{OL}	—	0.2	V
8	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
9	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, $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}/\overline{\text{CS}}$ I_{OL} = 12 mA Group 3	V_{OL}	 — — —	 0.4 0.4 0.4	V
10	Data Bus Mode Select Pull-Up Current[4] $V_{in} = V_{IL}$ DATA[15:0] $V_{in} = V_{IH}$ DATA[15:0]	I_{MSP}	 — –15	 –120 —	µA
11	Clock Synthesizer Operating Voltage	V_{DDSYN}	4.5	5.5	V
12	V_{DDSYN} Supply Current VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, VCO off (STSCIM = 0) Crystal, V_{DD} powered down	I_{DDSYN}	 — — — —	 2 7 2 2	 mA mA mA mA
13	Input Capacitance[2,6] All input-only pins All input/output pins	C_{in}	— —	10 20	pF
14	Load Capacitance[2] Group 1 I/O Pins and CLKOUT, FREEZE/QUOT Group 2 I/O Pins and $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}/\overline{\text{CS}}$ Group 3 I/O pins	C_L	 — — —	 90 100 130	 pF

NOTES:

1. Parameter applies to the following pins:
   Port A: PA[7:0]/ADDR[18:11]
   Port B: PB[7:0]/ADDR[10:3]
   Port E: PE[7:6]/SIZ[1:0], PE5/$\overline{\text{AS}}$, PE4/$\overline{\text{DS}}$, PE3/$\overline{\text{RMC}}$
   Port F: PF[7:1]/$\overline{\text{IRQ[7:1]}}$, PF0/MODCLK
   Port G: PG[7:0]/DATA[15:8]
   Port H: PH[7:0]/DATA[7:0]
   Other: $\overline{\text{BKPT}}$/DSCLK, $\overline{\text{RESET}}$, TSC

2. Input-Only Pins: $\overline{\text{BKPT}}$/DSCLK, EXTAL, TSC
   Output-Only Pins: ADDR[2:0], $\overline{\text{CSBOOT}}$, $\overline{\text{BG}}$/$\overline{\text{CS1}}$, CLKOUT, FREEZE/QUOT
   Input/Output Pins:
   Group 1:Port G: PG[7:0]/DATA[15:8]
     Port H: PH[7:0]/DATA[7:0]
   Group 2:Port A: PA[7:0]/ADDR[18:11]
     Port B: PB[7:0]/ADDR[10:3]
     Port C: PC[6:3]/ADDR[22:19]/$\overline{\text{CS[9:6]}}$, PC2/FC2/$\overline{\text{CS5}}$, PC1/FC1, PC0/FC0/$\overline{\text{CS3}}$
     Port E: PE[7:6]/SIZ[1:0], PE5/$\overline{\text{AS}}$, PE4/$\overline{\text{DS}}$, PE3/$\overline{\text{RMC}}$
     Port F: PF[7:1]/$\overline{\text{IRQ[7:1]}}$, PF0/MODCLK
     Other: ADDR23/$\overline{\text{CS10}}$/ECLK, 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 $\overline{\text{HALT}}$ or $\overline{\text{RESET}}$.

5. Use of an active pulldown device is recommended.

6. This parameter is periodically sampled rather than 100% tested.

7. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

Table A-3 AC Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation (32.768 kHz or 4.194 MHz crystal)[2]	f	0.13	16.78	MHz
1	Clock Period	t_{cyc}	59.6	—	ns
1A	ECLK Period	t_{Ecyc}	476	—	ns
1B	External Clock Input Period[3]	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	External Clock Input High/Low Time[3]	t_{XCHL}	29.8	—	ns
4, 5	CLKOUT 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 Input Rise and Fall Time[4]	t_{XCrf}	—	5	ns
6	Clock High to ADDR, FC, SIZE, $\overline{\mathrm{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
7	Clock High to ADDR, DATA, FC, SIZE, $\overline{\mathrm{RMC}}$ High Impedance[14]	t_{CHAZx}	0	59	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\mathrm{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Asserted	t_{CLSA}	2	24	ns
9A	$\overline{\mathrm{AS}}$ to $\overline{\mathrm{DS}}$ or $\overline{\mathrm{CS}}$ Asserted (Read)[5]	t_{STSA}	–15	15	ns
11	ADDR, FC, SIZE, $\overline{\mathrm{RMC}}$ Valid to $\overline{\mathrm{AS}}$, $\overline{\mathrm{CS}}$, (and $\overline{\mathrm{DS}}$ Read) Asserted[14]	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{\mathrm{AS}}$, $\overline{\mathrm{CS}}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
14B	$\overline{\mathrm{AS}}$, $\overline{\mathrm{CS}}$ Width Asserted (Fast Cycle)	t_{SWDW}	40	—	ns
15	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Width Negated[6]	t_{SN}	40	—	ns
16	Clock High to $\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, R/$\overline{\mathrm{W}}$ High Impedance	t_{CHSZ}	—	59	ns
17	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated to R/$\overline{\mathrm{W}}$ High	t_{SNRN}	15	—	ns
18	Clock High to R/$\overline{\mathrm{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\mathrm{W}}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{\mathrm{W}}$ High to $\overline{\mathrm{AS}}$, $\overline{\mathrm{CS}}$ Asserted	t_{RAAA}	15	—	ns
22	R/$\overline{\mathrm{W}}$ Low to $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ 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 $\overline{\mathrm{AS}}$, $\overline{\mathrm{CS}}$	t_{DVASN}	15	—	ns
25	$\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Asserted (Write)	t_{DVSA}	15	—	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
27A	Late $\overline{\mathrm{BERR}}$, $\overline{\mathrm{HALT}}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to $\overline{\mathrm{DSACK[1:0]}}$, $\overline{\mathrm{BERR}}$, $\overline{\mathrm{HALT}}$, $\overline{\mathrm{AVEC}}$ Negated	t_{SNDN}	0	80	ns
29	$\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{\mathrm{DS}}$, $\overline{\mathrm{CS}}$ Negated to Data In High Impedance[7,8]	t_{SHDI}	—	55	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7]	t_{CLDI}	15	—	ns
30A	CLKOUT Low to Data In High Impedance[7]	t_{CLDH}	—	90	ns
31	$\overline{\mathrm{DSACK[1:0]}}$ Asserted to Data In Valid[9]	t_{DADI}	—	50	ns
33	Clock Low to $\overline{\mathrm{BG}}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35	$\overline{\mathrm{BR}}$ Asserted to $\overline{\mathrm{BG}}$ Asserted ($\overline{\mathrm{RMC}}$ not asserted)[10,14]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{\mathrm{BGACK}}$ Asserted to $\overline{\mathrm{BG}}$ Negated	t_{GAGN}	1	2	t_{cyc}
39	$\overline{\mathrm{BG}}$ Width Negated	t_{GH}	2	—	t_{cyc}
39A	$\overline{\mathrm{BG}}$ Width Asserted	t_{GA}	1	—	t_{cyc}
46	R/$\overline{\mathrm{W}}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
46A	R/$\overline{\mathrm{W}}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns
47A	Asynchronous Input Setup Time $\overline{\mathrm{BR}}$, $\overline{\mathrm{BGACK}}$, $\overline{\mathrm{DSACK[1:0]}}$, $\overline{\mathrm{BERR}}$, $\overline{\mathrm{AVEC}}$, $\overline{\mathrm{HALT}}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns

Table A-3 AC Timing (Continued)

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
48	$\overline{\text{DSACK[1:0]}}$ Asserted to $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted[11]	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/$\overline{\text{W}}$ Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns
70	Clock Low to Data Bus Driven (Show Cycle)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show Cycle)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show Cycle)	t_{SCLDH}	10	—	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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	$\overline{\text{RESET}}$ Assertion Time[12]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{\text{RESET}}$ Rise Time[13]	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. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is 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).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal — if transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.
5. Specification 9A is the worst-case skew between $\overline{\text{AS}}$ and $\overline{\text{DS}}$ or $\overline{\text{CS}}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{\text{AS}}$ and $\overline{\text{DS}}$ to fall outside the limits shown in specification 9.
6. If multiple chip selects are used, $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{\text{DSACK[1:0]}}$ low to data setup time (specification 31) and $\overline{\text{DSACK[1:0]}}$ low to $\overline{\text{BERR}}$ 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. $\overline{\text{BERR}}$ must satisfy only the late $\overline{\text{BERR}}$ low to clock low setup time (specification 27A) for the following clock cycle.
10. To ensure coherency during every operand transfer, $\overline{\text{BG}}$ is not asserted in response to $\overline{\text{BR}}$ until after all cycles of the current operand transfer are complete.
11. In the absence of $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$ is an asynchronous input using the asynchronous setup time (specification 47A).
12. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SCIM drives $\overline{\text{RESET}}$ low for 512 t_{cyc}.
13. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.
15. 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.

Table A-4 ECLK Bus Timing

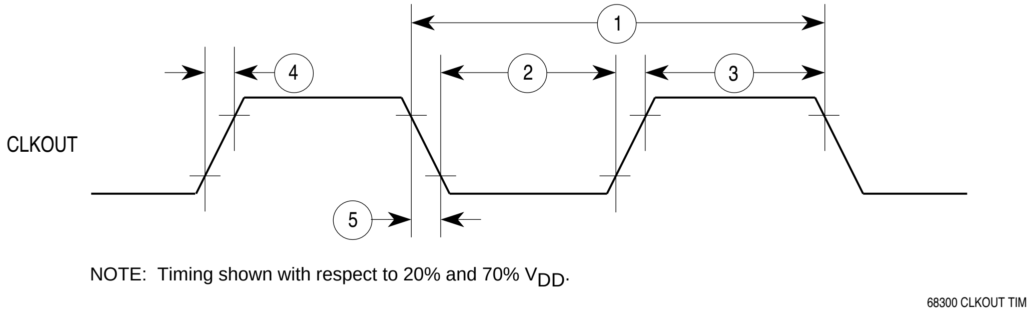
(V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
E1	ECLK Low to Address Valid[2]	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{\text{CS}}$ Valid ($\overline{\text{CS}}$ delay)	t_{ECSD}	—	150	ns
E4	ECLK Low to $\overline{\text{CS}}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{\text{CS}}$ Negated Width	t_{ECSN}	30	—	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}	—	60	ns
E9	$\overline{\text{CS}}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{\text{CS}}$ 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	$\overline{\text{CS}}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14	Address Access Time (Read)[3]	t_{EACC}	386	—	ns
E15	Chip Select Access Time (Read)[4]	t_{EACS}	296	—	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 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$

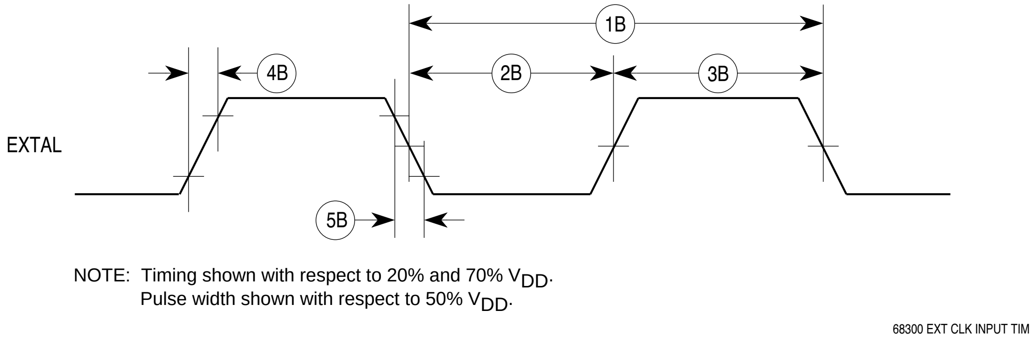
A.1 Timing Diagrams



NOTE: Timing shown with respect to 20% and 70% V_{DD}.

68300 CLKOUT TIM

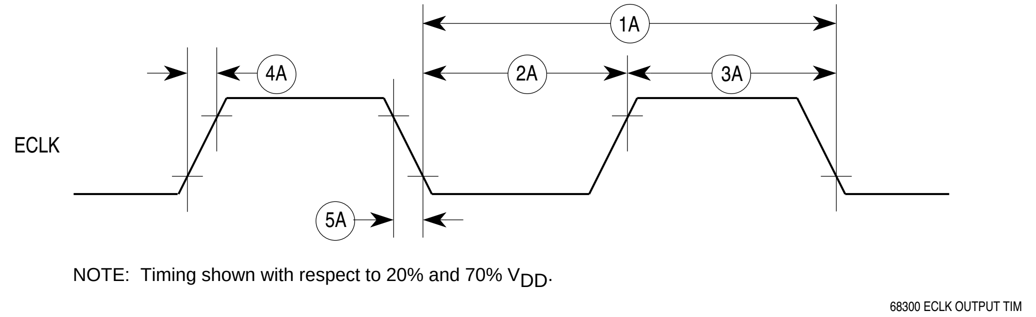
Figure A-1 CLKOUT Output Timing Diagram



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	External Clock Input Period[3]	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	External Clock Input High/Low Time[3]	t_{XCHL}	29.8	—	ns
4, 5	CLKOUT 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 Input Rise and Fall Time[4]	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. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is 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).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal — if transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.

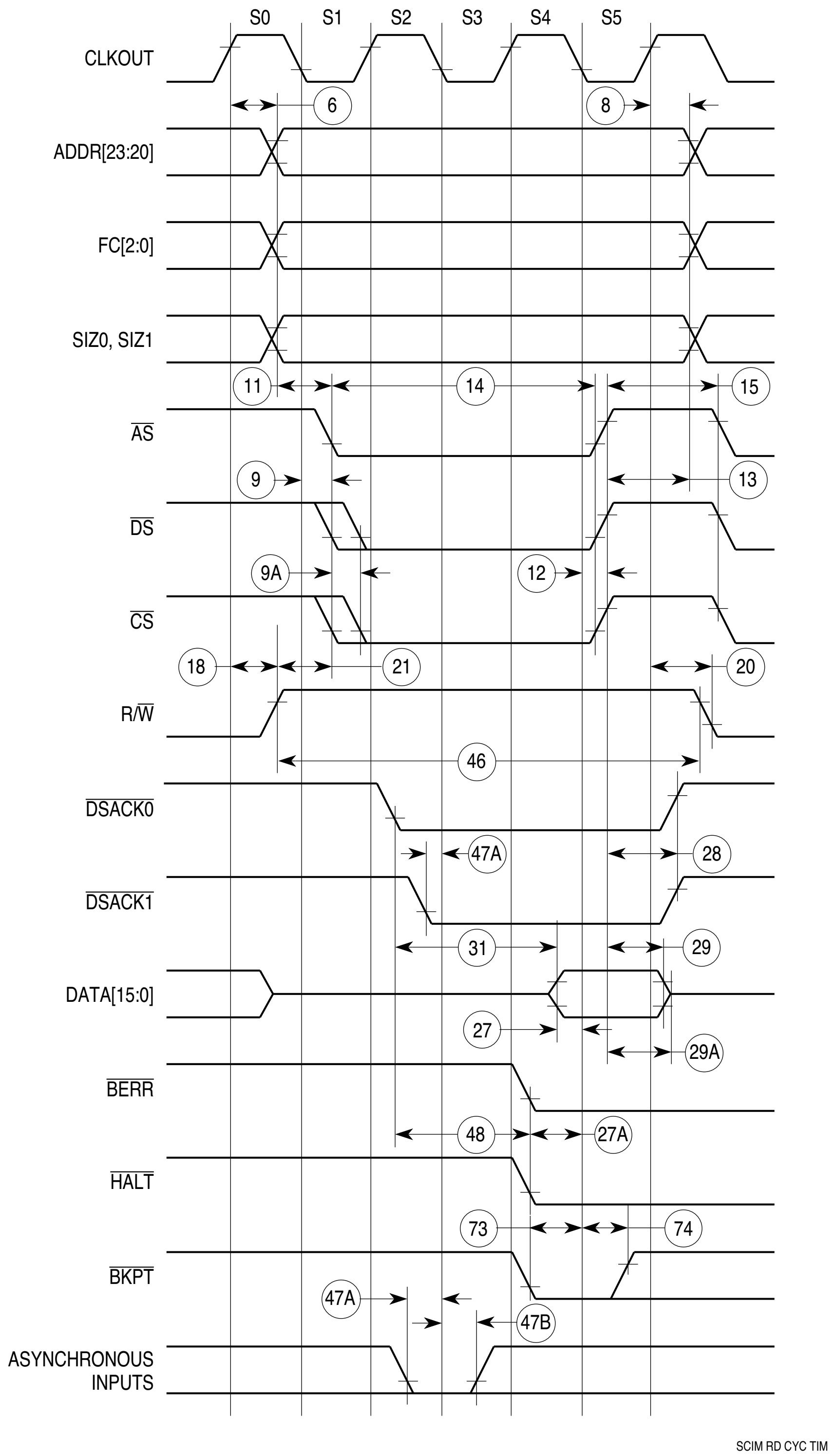


Figure A-4 Read Cycle Timing Diagram

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	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
9A	$\overline{\text{AS}}$ to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ Asserted (Read)[5]	t_{STSA}	–15	15	ns
11	ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid to $\overline{\text{AS}}$, $\overline{\text{CS}}$, (and $\overline{\text{DS}}$ Read) Asserted[14]	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{\text{AS}}$, $\overline{\text{CS}}$ Width Asserted	t_{SWA}	100	—	ns
15	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Negated[6]	t_{SN}	40	—	ns
18	Clock High to R/$\overline{\text{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\text{W}}$ Low	t_{CHRL}	0	29	ns
21	R/$\overline{\text{W}}$ High to $\overline{\text{AS}}$, $\overline{\text{CS}}$ Asserted	t_{RAAA}	15	—	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
27A	Late $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{\text{AS}}$, $\overline{\text{DS}}$ Negated to $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$, $\overline{\text{HALT}}$, $\overline{\text{AVEC}}$ Negated	t_{SNDN}	0	80	ns
29	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In High Impedance[7,8]	t_{SHDI}	—	55	ns
31	$\overline{\text{DSACK[1:0]}}$ Asserted to Data In Valid[9]	t_{DADI}	—	50	ns
46	R/$\overline{\text{W}}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{\text{BR}}$, $\overline{\text{BG}}$, $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$, $\overline{\text{AVEC}}$, $\overline{\text{HALT}}$	t_{AIST}	5	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	15	—	ns
48	$\overline{\text{DSACK[1:0]}}$ Asserted to $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted[11]	t_{DABA}	—	30	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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. Specification 9A is the worst-case skew between $\overline{\text{AS}}$ and $\overline{\text{DS}}$ or $\overline{\text{CS}}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{\text{AS}}$ and $\overline{\text{DS}}$ to fall outside the limits shown in specification 9.
6. If multiple chip selects are used, $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ on asynchronous reads and with respect to CLKOUT on fast reads. The user is free to use either hold time.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{\text{DSACK[1:0]}}$ low to data setup time (specification 31) and $\overline{\text{DSACK[1:0]}}$ low to $\overline{\text{BERR}}$ 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. $\overline{\text{BERR}}$ must satisfy only the late $\overline{\text{BERR}}$ low to clock low setup time (specification 27A) for the following clock cycle.
11. In the absence of $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$ is an asynchronous input — use specification 47A.
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

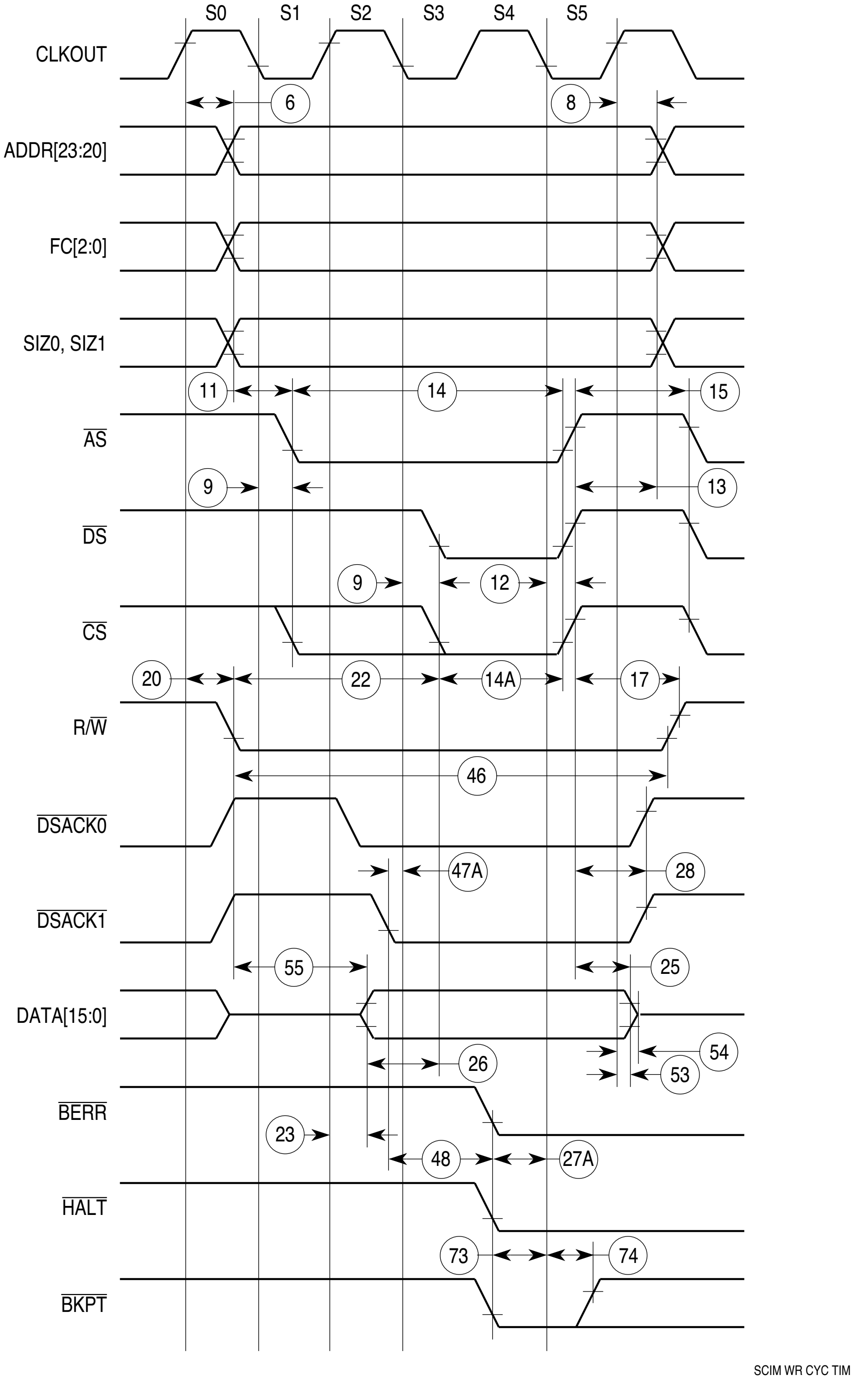


Figure A-5 Write Cycle Timing Diagram

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	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
11	ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid to $\overline{\text{AS}}$, $\overline{\text{CS}}$, (and $\overline{\text{DS}}$ Read) Asserted[14]	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{\text{AS}}$, $\overline{\text{CS}}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
15	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Negated[6]	t_{SN}	40	—	ns
17	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to R/$\overline{\text{W}}$ High	t_{SNRN}	15	—	ns
20	Clock High to R/$\overline{\text{W}}$ Low	t_{CHRL}	0	29	ns
22	R/$\overline{\text{W}}$ Low to $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted (Write)	t_{RASA}	70	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
25	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
26	Data Out Valid to $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted (Write)	t_{DVSA}	15	—	ns
27A	Late $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	20	—	ns
28	$\overline{\text{AS}}$, $\overline{\text{DS}}$ Negated to $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$, $\overline{\text{HALT}}$, $\overline{\text{AVEC}}$ Negated	t_{SNDN}	0	80	ns
46	R/$\overline{\text{W}}$ Width Asserted (Write or Read)	t_{RWA}	150	—	ns
47A	Asynchronous Input Setup Time $\overline{\text{BR}}$, $\overline{\text{BGACK}}$, $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$, $\overline{\text{AVEC}}$, $\overline{\text{HALT}}$	t_{AIST}	5	—	ns
48	$\overline{\text{DSACK[1:0]}}$ Asserted to $\overline{\text{BERR}}$, $\overline{\text{HALT}}$ Asserted[11]	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/$\overline{\text{W}}$ Asserted to Data Bus Impedance Change	t_{RADC}	40	—	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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. If multiple chip selects are used, $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
11. In the absence of $\overline{\text{DSACK[1:0]}}$, $\overline{\text{BERR}}$ is an asynchronous input using the asynchronous setup time (specification 47A).
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

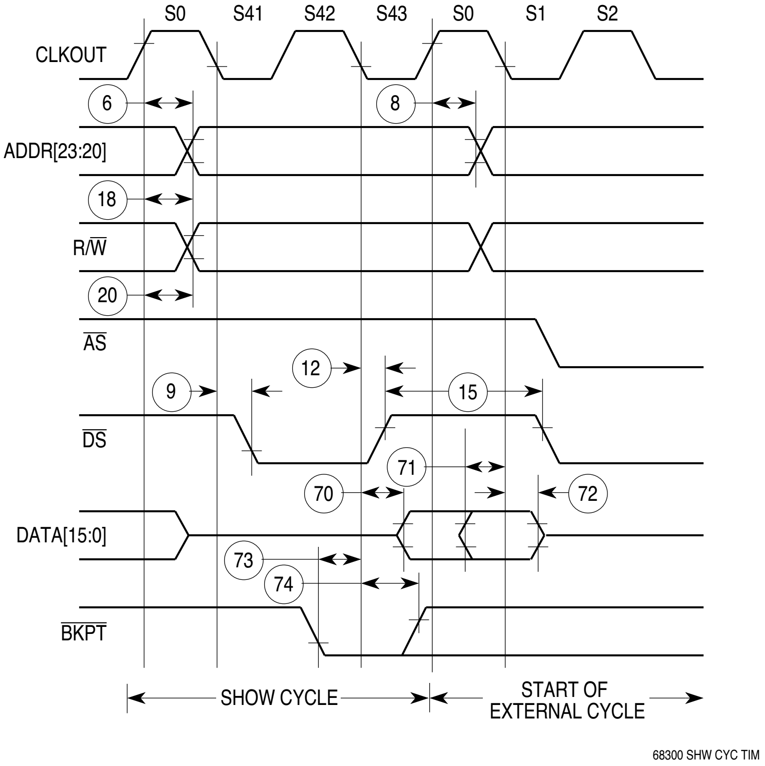


Figure A-6 Show Cycle Timing Diagram

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	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
15	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Negated[6]	t_{SN}	40	—	ns
18	Clock High to R/$\overline{\text{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\text{W}}$ Low	t_{CHRL}	0	29	ns
70	Clock Low to Data Bus Driven (Show Cycle)	t_{SCLDD}	0	29	ns
71	Data Setup Time to Clock Low (Show Cycle)	t_{SCLDS}	15	—	ns
72	Data Hold from Clock Low (Show Cycle)	t_{SCLDH}	10	—	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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. If multiple chip selects are used, $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

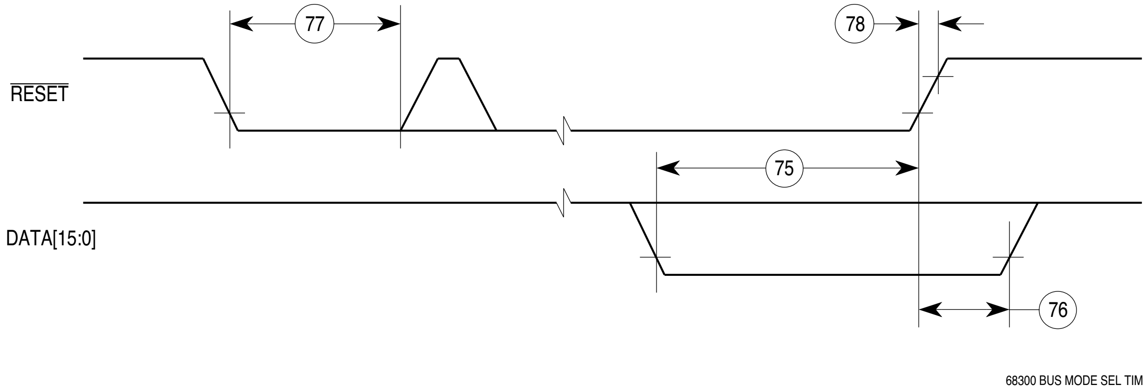


Figure A-7 Reset and Mode Select Timing Diagram

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[12]	t_{RSTA}	4	—	t_{cyc}
78	$\overline{\text{RESET}}$ Rise Time[13]	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.
12. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the SCIM drives $\overline{\text{RESET}}$ low for 512 t_{cyc}.
13. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.

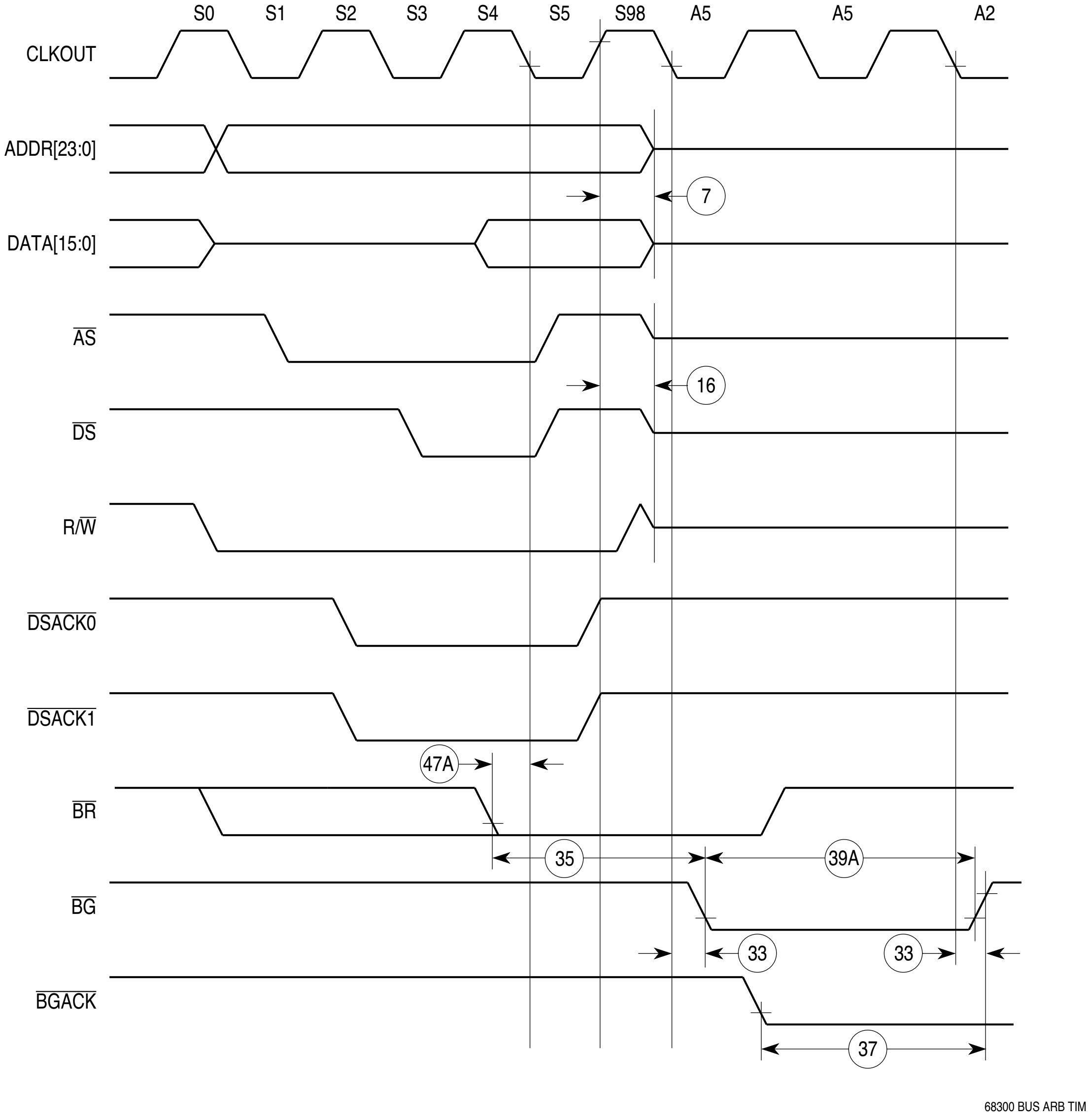


Figure A-8 Bus Arbitration Timing Diagram — Active Bus Case

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	Clock High to ADDR, DATA, FC, SIZE, $\overline{\mathrm{RMC}}$ High Impedance[14]	t_{CHAZx}	0	59	ns
16	Clock High to $\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$, R/$\overline{\mathrm{W}}$ High Impedance	t_{CHSZ}	—	59	ns
33	Clock Low to $\overline{\mathrm{BG}}$ Asserted/Negated	t_{CLBAN}	—	29	ns
35	$\overline{\mathrm{BR}}$ Asserted to $\overline{\mathrm{BG}}$ Asserted ($\overline{\mathrm{RMC}}$ not asserted)[10,14]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{\mathrm{BGACK}}$ Asserted to $\overline{\mathrm{BG}}$ Negated	t_{GAGN}	1	2	t_{cyc}
39A	$\overline{\mathrm{BG}}$ Width Asserted	t_{GA}	1	—	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{\mathrm{BR}}$, $\overline{\mathrm{BGACK}}$, $\overline{\mathrm{DSACK[1:0]}}$, $\overline{\mathrm{BERR}}$, $\overline{\mathrm{AVEC}}$, $\overline{\mathrm{HALT}}$	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.
10. To ensure coherency during every operand transfer, $\overline{\mathrm{BG}}$ is not asserted in response to $\overline{\mathrm{BR}}$ until after all cycles of the current operand transfer are complete.
14. $\overline{\mathrm{RMC}}$ signal is not used in M68HC16 devices.

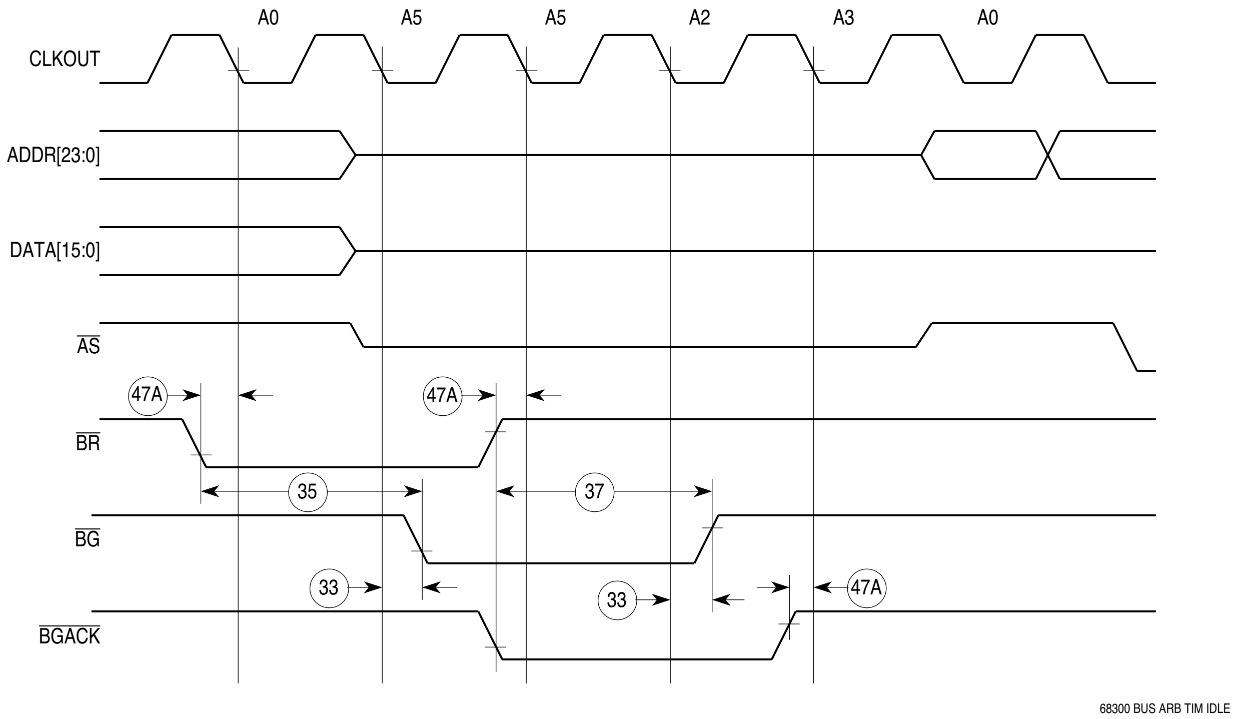


Figure A-9 Bus Arbitration Timing Diagram — Idle Bus Case

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_{CLBAN}	—	29	ns
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ not asserted)[10,14]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$	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.
10. To ensure coherency during every operand transfer, $\overline{BG}$ is not asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete.
14. $\overline{RMC}$ signal is not used in M68HC16 devices.

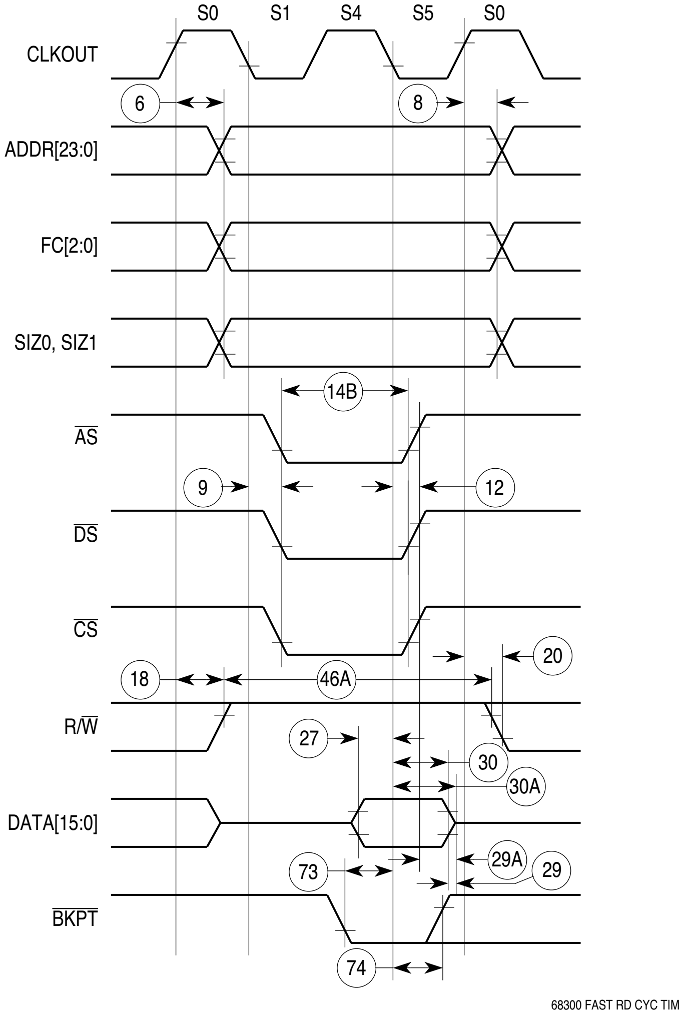


Figure A-10 Fast Termination Read Cycle Timing Diagram

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	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{\text{AS}}$, $\overline{\text{CS}}$ Width Asserted (Fast Cycle)	t_{SWDW}	40	—	ns
18	Clock High to R/$\overline{\text{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\text{W}}$ Low	t_{CHRL}	0	29	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	5	—	ns
29	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In High Impedance[7,8]	t_{SHDI}	—	55	ns
30	CLKOUT Low to Data In Invalid (Fast Hold)[7]	t_{CLDI}	15	—	ns
30A	CLKOUT Low to Data In High Impedance[7]	t_{CLDH}	—	90	ns
46A	R/$\overline{\text{W}}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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.
7. Hold times are specified with respect to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

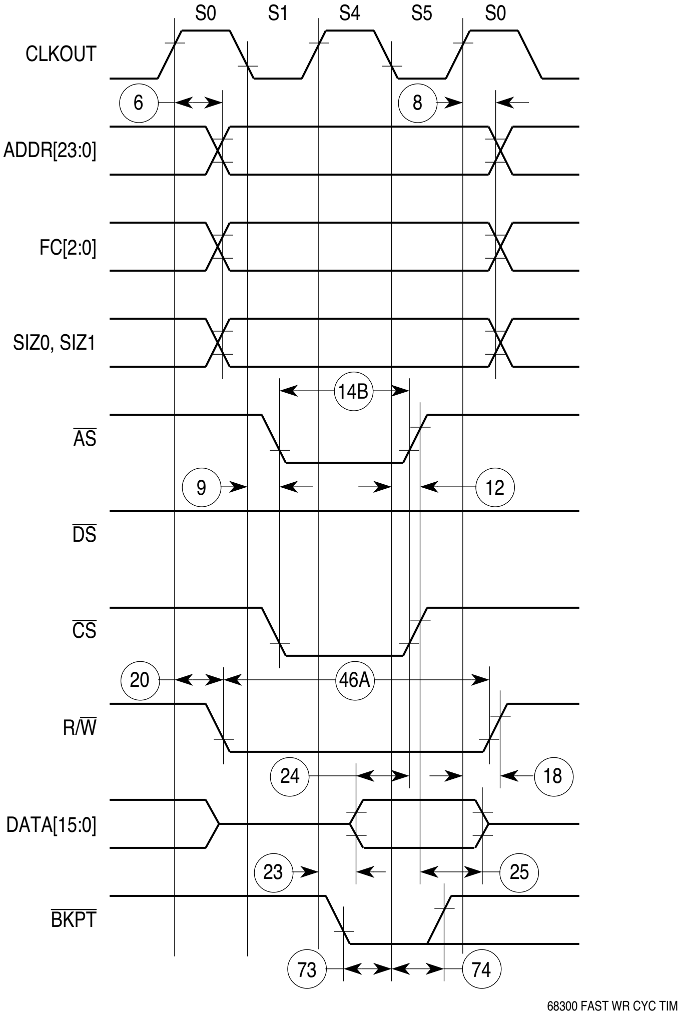


Figure A-11 Fast Termination Write Cycle Timing Diagram

Table A-13 Key to Figure A–11

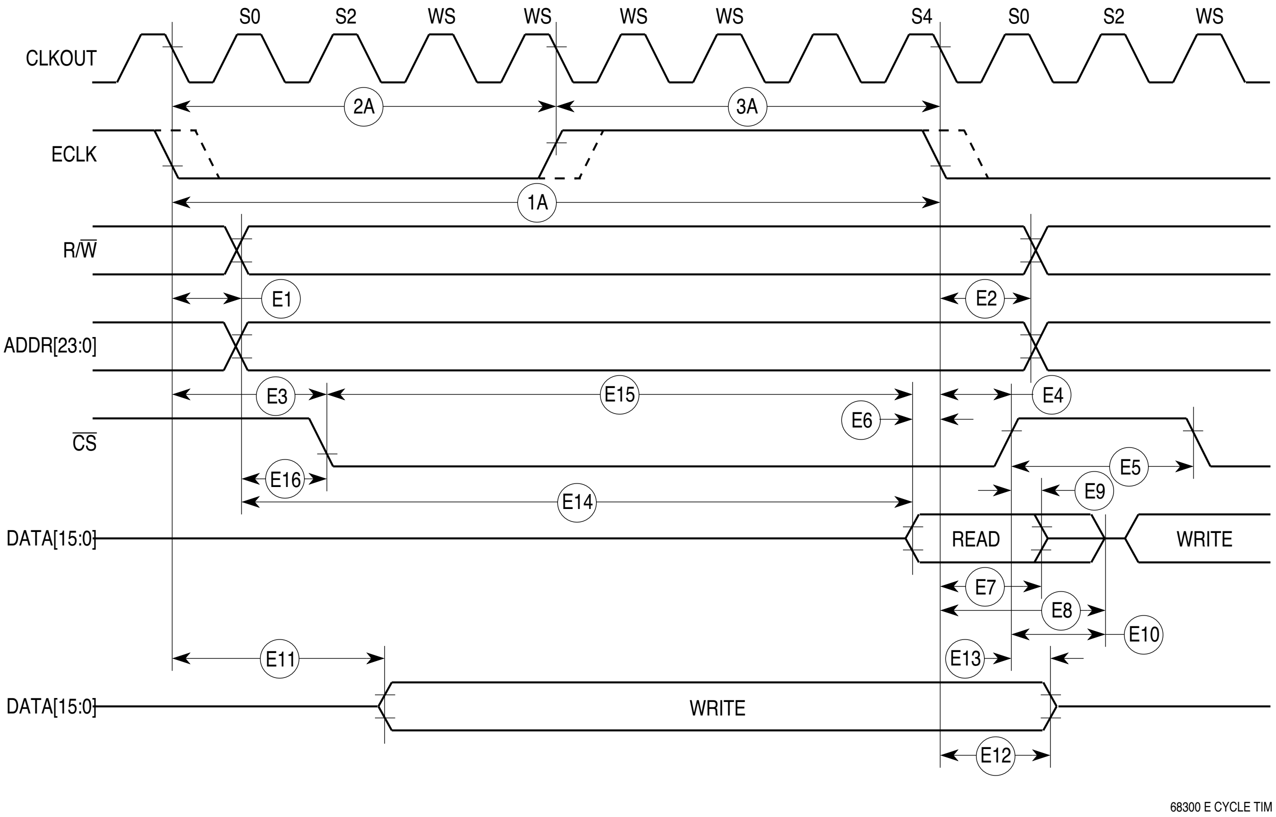
(Abstracted from Table A–3; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
14B	$\overline{\text{AS}}$, $\overline{\text{CS}}$ Width Asserted (Fast Cycle)	t_{SWDW}	40	—	ns
18	Clock High to R/$\overline{\text{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\text{W}}$ 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 $\overline{\text{AS}}$, $\overline{\text{CS}}$	t_{DVASN}	15	—	ns
25	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
46A	R/$\overline{\text{W}}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	90	—	ns
73	$\overline{\text{BKPT}}$ Input Setup Time	t_{BKST}	15	—	ns
74	$\overline{\text{BKPT}}$ 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. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.



NOTE: Shown with ECLK = system clock/8 — EDIV bit in clock synthesizer control register (SYNCR) = 0.

Figure A-12 ECLK Timing Diagram

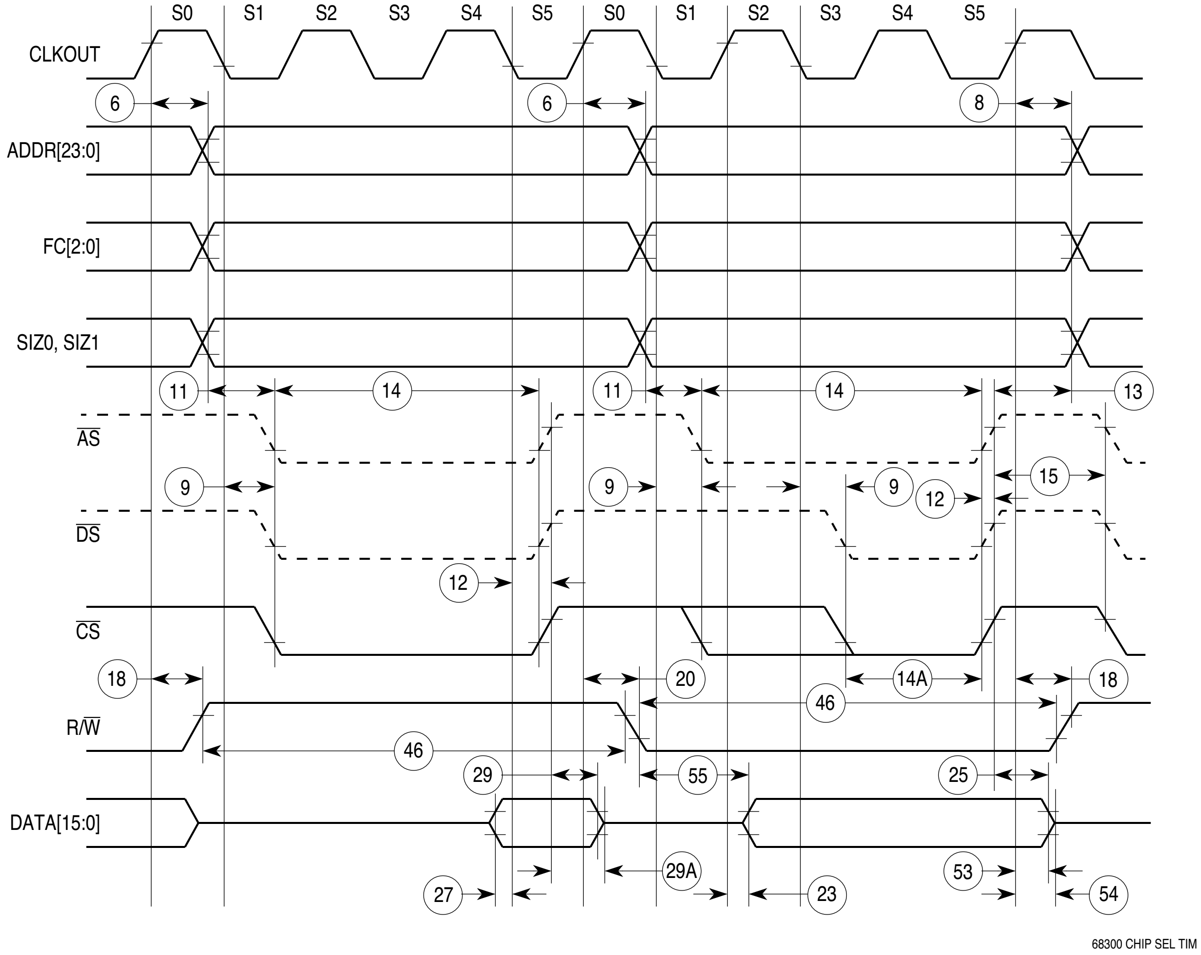
Table A-14 Key to Figure A–12

(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_{rf}	—	8	ns
E1	ECLK Low to ADDR and R/$\overline{W}$ Valid[2]	t_{EAD}	—	60	ns
E2	ECLK Low to ADDR and R/$\overline{W}$ Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ delay)	t_{ECSD}	—	150	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	30	—	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}	—	60	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	Address Access Time (Read)[3]	t_{EACC}	386	—	ns
E15	Chip Select Access Time (Read)[4]	t_{EACS}	296	—	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 previous bus cycle is not an 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$



NOTE: $\overline{\text{AS}}$ and $\overline{\text{DS}}$ timing shown for reference only.

Figure A-13 Chip Select Timing Diagram

Table A-15 Key to Figure A–13

(Abstracted from Table A–6; see table for complete notes)

Num	Characteristic	Symbol	Min	Max	Units
6	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid[14]	t_{CHAV}	0	29	ns
8	Clock High to ADDR, FC, SIZE, $\overline{\text{RMC}}$ Invalid[14]	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Asserted	t_{CLSA}	2	24	ns
11	ADDR, FC, SIZE, $\overline{\text{RMC}}$ Valid to $\overline{\text{AS}}$, $\overline{\text{CS}}$, (and $\overline{\text{DS}}$ Read) Asserted[14]	t_{AVSA}	15	—	ns
12	Clock Low to $\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated	t_{CLSN}	2	29	ns
13	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to ADDR, FC, SIZE Invalid (Address Hold)	t_{SNAI}	15	—	ns
14	$\overline{\text{AS}}$, $\overline{\text{CS}}$ Width Asserted	t_{SWA}	100	—	ns
14A	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Asserted (Write)	t_{SWAW}	45	—	ns
15	$\overline{\text{AS}}$, $\overline{\text{DS}}$, $\overline{\text{CS}}$ Width Negated[6]	t_{SN}	40	—	ns
18	Clock High to R/$\overline{\text{W}}$ High	t_{CHRH}	0	29	ns
20	Clock High to R/$\overline{\text{W}}$ Low	t_{CHRL}	0	29	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	29	ns
25	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	15	—	ns
29	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In Invalid (Data In Hold)[7]	t_{SNDI}	0	—	ns
29A	$\overline{\text{DS}}$, $\overline{\text{CS}}$ Negated to Data In High Impedance[7,8]	t_{SHDI}	—	55	ns
46	R/$\overline{\text{W}}$ 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/$\overline{\text{W}}$ 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.
6. If multiple chip selects are used, $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to $\overline{\text{DS}}$ or $\overline{\text{CS}}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
8. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
14. $\overline{\text{RMC}}$ signal is not used in M68HC16 devices.

APPENDIX BREGISTER SUMMARY

B.1 SCIM Address Map

Table B-1 SCIM Address Map

Access	Address	15 8	7 0
S	$####00	SCIM CONFIGURATION REGISTER (SCIMCR)	
S	$####02	MODULE TEST (SCIMTR)	
S	$####04	CLOCK SYNTHESIZER CONTROL (SYNCR)	
S	$####06	UNUSED	RESET STATUS REGISTER (RSR)
S	$####08	MODULE TEST E (SCIMTRE)	
S/U	$####0A	PORT A DATA (PORTA)	PORT B DATA (PORTB)
S/U	$####0C	PORT G DATA (PORTG)	PORT H DATA (PORTH)
S/U	$####0E	PORT G DATA DIRECTION (DDRG)	PORT H DATA DIRECTION (DDRH)
S/U	$####10	UNUSED	PORT E DATA (PORTE0)
S/U	$####12	UNUSED	PORT E DATA (PORTE1)
S/U	$####14	PORT A/B DATA DIRECTION (DDRAB)	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 (PICR)	
S	$####24	PERIODIC INTERRUPT TIMING (PITR)	
S	$####26	UNUSED	SOFTWARE SERVICE (SWSR)
S	$####28	UNUSED	PORT F EDGE DETECT FLAGS (PORTFE)
S	$####2A	UNUSED	PORT F EDGE DETECT INTERRUPT VECTOR (PFIVR)
S	$####2C	UNUSED	PORT F EDGE DETECT INTERRUPT LEVEL (PFLVR)
S/U	$####2E	UNUSED	
S	$####30	TEST MODULE MASTER SHIFT A (TSTMSRA)	
S	$####32	TEST MODULE MASTER SHIFT B (TSTMSRB)	
S	$####34	TEST MODULE SHIFT COUNT A (TSTSCA)	TEST MODULE SHIFT COUNT B (TSTSCB)
S	$####36	TEST MODULE REPETITION COUNTER (TSTRC)	
S	$####38	TEST MODULE CONTROL (CREG)	
S/U	$####3A	TEST MODULE DISTRIBUTED REGISTER (DREG)	
S/U	$####3C	UNUSED	UNUSED
S/U	$####3E	UNUSED	UNUSED
S/U	$####40	UNUSED	PORT C DATA (PORTC)
S/U	$####42	UNUSED	UNUSED
S	$####44	CHIP-SELECT PIN ASSIGNMENT 0 (CSPAR0)	

Table B-1 SCIM Address Map

Access	Address	15 8 7 0
S	$####46	CHIP-SELECT PIN ASSIGNMENT 1 (CSPAR1)
S	$####48	CHIP-SELECT BASE ADDRESS BOOT (CSBARBT)
S	$####4A	CHIP-SELECT OPTION BOOT (CSORBT)
S	$####4C	CHIP-SELECT BASE 0 (CSBAR0)
S	$####4E	CHIP-SELECT OPTION 0 (CSOR0)
S	$####50	UNUSED
S	$####52	UNUSED
S	$####54	UNUSED
S	$####56	UNUSED
S	$####58	CHIP-SELECT BASE 3 (CSBAR3)
S	$####5A	CHIP-SELECT OPTION 3 (CSOR3)
S	$####5C	UNUSED
S	$####5E	UNUSED
S	$####60	CHIP-SELECT BASE 5 (CSBAR5)
S	$####62	CHIP-SELECT OPTION 5 (CSOR5)
S	$####64	CHIP-SELECT BASE 6 (CSBAR6)
S	$####66	CHIP-SELECT OPTION 6 (CSOR6)
S	$####68	CHIP-SELECT BASE 7 (CSBAR7)
S	$####6A	CHIP-SELECT OPTION 7 (CSOR7)
S	$####6C	CHIP-SELECT BASE 8 (CSBAR8)
S	$####6E	CHIP-SELECT OPTION 8 (CSOR8)
S	$####70	CHIP-SELECT BASE 9 (CSBAR9)
S	$####72	CHIP-SELECT OPTION 9 (CSOR9)
S	$####74	CHIP-SELECT BASE 10 (CSBAR10)
S	$####76	CHIP-SELECT OPTION 10 (CSOR10)
	$####78	UNUSED
	$####7A	UNUSED
	$####7C	UNUSED
	$####7E	UNUSED

B.2 SCIM Registers

SCIMCR — SCIM Configuration Register **$####00**

15	14	13	12	11	10	9	8	7	6	5	4	3			0
EXOFF	FRZSW	FRZBM	CPUD	SLVEN	0	SHEN		SUPV	MM	ABD	RWD	IARB			
RESET:															
0	0	0	$\overline{\text{BERR}}$	$\overline{\text{DATA11}}$	0	0	0	1	1	$\overline{\text{BERR}}$	$\overline{\text{BERR}}$	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.

CPUD — CPU Development Support Disable
- 0 = Instruction pipeline signals available
- 1 = Instruction pipeline pins placed in high-impedance state unless a breakpoint occurs.

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

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.

SLVEN is a read-only status bit. Slave mode is used for factory testing. Reset state is the complement of DATA11 during reset for fully expanded operation.

SHEN[1:0] — Show Cycle Enable

This field determines what the external bus interface does with the external bus during internal transfer operations.

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.

CAUTION

> 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.

ABD — Address Bus Disable

0 = ADDR[2:0] signals are available.
1 = ADDR[2:0] pins are disabled (placed in a high-impedance state).

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

RWD — R/$\overline{\text{W}}$ Pin Disable

0 = R/$\overline{\text{W}}$ signal is available.
1 = R/$\overline{\text{W}}$ signal is disabled (placed in a high-impedance state).

The reset state is one if $\overline{\text{BERR}}$ is held low during reset, configuring the MCU for single-chip operation, or zero if $\overline{\text{BERR}}$ is held high during reset.

IARB[3:0] — Interrupt Arbitration Number

IARB determines SCIM interrupt arbitration priority. The reset value is \$F (highest priority), to prevent SCIM interrupts from being discarded during initialization.

SCIMTR — Single-Chip Integration Module Test Register **\$####02**

SCIMTR is used for factory test only.

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

15	14	13					8	7	6	5	4	3	2	1	0
W	X	Y						EDIV	0	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	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.

X — Frequency Control Bit (Prescale)

0 = Base system clock frequency
1 = System clock frequency multiplied by two

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.

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

LOSCD — Loss-of-Clock Oscillator Disable
0 = Enable the loss-of-clock oscillator.
1 = Disable the loss-of-clock oscillator.

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

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.

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

STEXT — Stop Mode External Clock
0 = CLKOUT held low during low-power stop.
1 = CLKOUT driven from SCIM clock during low-power stop.

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

15–8	7	6	5	4	3	2	1	0
NOT USED	EXT	POW	SW	HLT	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.

HLT — Halt Monitor Reset
Reset was caused by the system protection module halt monitor.

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.

SCIMTRE — Single-Chip Integration Module Test Register (ECLK) **$####08**

The SCIMTRE is used for factory test only.

PORTA — Port A Data Register **$####0A**
PORTB — Port B Data Register **$####0B**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Ports A and B are available in single-chip mode only. PORTA and PORTB can be read or written any time the MCU is not in emulator mode.

DDRAB — Port A/B Data Direction Register **$####14**

15	14	13	12	11	10	9	8	7							0
0	0	0	0	0	0	DDA	DDB	DDRE (Port E Data Direction)							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDA and DDB control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

PORTG — Port G Data Register **$####0C**
PORTH — Port H Data Register **$####0D**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

PORTG and PORTH can be read or written any time the MCU is not in emulator mode. Resets have no effect.

DDRG — Port G Data Direction Register **$####0E**

DDRH — Port H Data Direction Register **$####0F**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bits in DDRG and DDRH control the direction of the pin drivers when the pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input.

PORTE0, PORTE1 — Port E Data Register **$####10, $####12**

	15	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

PORTE is a single register that can be accessed in two locations. It can be read or written anytime the MCU is not in emulator mode.

DDRAB — Port A/B Data Direction Register **$####14**

DDRE — Port E Data Direction Register **$####15**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	DDA	DDB	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDA and DDB control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

The bits in DDRE controls the directions of the pin drivers when the pins are configured for I/O. Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. This register can be read or written anytime the MCU is not in emulator mode.

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

	15	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	$\overline{\text{AS}}$
PEPA4	PE4	$\overline{\text{DS}}$
PEPA3	PE3	$\overline{\text{RMC}}$*
PEPA2	PE2	$\overline{\text{AVEC}}$
PEPA1	PE1	$\overline{\text{DSACK1}}$
PEPA0	PE0	$\overline{\text{DSACK0}}$

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

PORTF0, PORTF1 — Port F Data Register **$####18, $####1A**

15	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

A write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of PORTF returns the value on a pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data register.

Port F is a single register that can be accessed in two locations. It can be read or written at any time, including when the MCU is in emulator mode.

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

15	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

The bits in the DDRF control the direction of port F pin drivers when the pins are configured for I/O. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding pin as an input.

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

15	8	7	6	5	4	3	2	1	0
NOT USED		PFPA3		PFPA2		PFPA1		PFPA0	
RESET (Expanded-bus configuration):									
		DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
RESET (Single-chip configuration):									
		0	0	0	0	0	0	0	0

The fields in the PFPAR determine the functions of pairs of port F pins as shown in **Table B-3** and **Table B-4**. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register.

If either $\overline{\text{BERR}}$ or DATA9 is low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins except MODCLK/PF0 to be interrupt signals.

Table B-3 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ[7:6]}}$
PFPA2	PF[5:4]	$\overline{\text{IRQ[5:4]}}$
PFPA1	PF[3:2]	$\overline{\text{IRQ[3:2]}}$
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, MODCLK*

*MODCLK signal is recognized only during reset

Table B-4 PFPAR Pin Encodings

PFPA Bits	Port F Signal
00	I/O pin without edge detection
01	Rising edge detection
10	Falling edge detection
11	Interrupt request (or MODCLK)

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

15	8	7	6	5	4	3	2	1	0
NOT USED		SWE	SWP	SWT		HME	BME	BMT	
RESET:									
		1	$\overline{\text{MODCLK}}$	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.

Table B-5 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.

Table B-6 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		8	7							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 SCIM when the CPU acknowledges an interrupt request.

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

15	14	13	12	11	10	9	8	7							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	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

PORTFE — Port F Edge-Detect Flag Register $####28

15	8	7	6	5	4	3	2	1	0
NOT USED		EF7	EF6	EF5	EF4	EF3	EF2	EF1	EF0
RESET:									
		0	0	0	0	0	0	0	0

When a pin is configured in the PFPAR to detect either a rising or falling edge, and such an edge is detected, the corresponding bit in the PORTFE is set. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state.

PFIVR — Port F Edge-Detect Interrupt Vector Register $####2A

15	8	7	6	5	4	3	2	1	0
NOT USED		PFIV7	PFIV6	PFIV5	PFIV4	PFIV3	PFIV2	PFIV1	PFIV0
RESET:									
		0	0	0	0	1	1	1	1

The PFIVR determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIV[7:0] to the value pointing to the appropriate interrupt vector. Refer to the appropriate CPU reference manual for interrupt vector assignments.

PFLVR — Port F Edge-Detect Interrupt Level Register $####2C

15	8	7	6	5	4	3	2	1	0
NOT USED		0	0	0	0	0	PFLV2	PFLV1	PFLV0
RESET:									
		0	0	0	0	0	0	0	0

PFLVR determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts within the SCIM with the same interrupt request level enter interrupt arbitration, the port F edge-detect interrupt has the lowest arbitration priority.

PORTC — Port C Data Register $####40

15	8	7	6	5	4	3	2	1	0
NOT USED		0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:									
		0	1	1	1	1	1	1	1

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.

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]		CSPA05		CSPA04		CSPA03		CSPA02		CSPA01		$\overline{\text{CSBOOT}}$	
RESET:															
0	0	DATA2	1	DATA2	1	DATA2	1	DATA1	1	DATA1	1	DATA1	1	1	DATA0

CSPAR0 contains six 2-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-7 CSPAR0 Pin Assignments

CSPAR0 Field	Chip Select Signal	Alternate Signal	Discrete Output
CSPA0[6]	$\overline{\text{CS5}}$	FC2	PC2
CSPA0[5]	—	FC1	PC1
CSPA0[4]	$\overline{\text{CS3}}$	FC0	PC0
CSPA0[3]	$\overline{\text{CSE}}$	$\overline{\text{BGACK}}$	—
CSPA0[2]	$\overline{\text{CSM}}$	$\overline{\text{BG}}$	—
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	DATA[7:6]	1	DATA[7:5]	1	DATA[7:4]	1	DATA[7:3]	1

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. Each pin is assigned one of the functions shown in **Table B-6**. **Table B-7** explains how pin function is assigned at reset. CSPAR1[15:10] are not used. These bits always return zero when read; writes have no effect.

Table B-8 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-9 Chip-Select 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		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[10:5] — Chip Select Base Address Registers **$####60–$####74**
CSBAR3 **$####58**
CSBAR0 **$####4C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2		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			6	5	4	3		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[10:5] — Chip Select Option Registers **$####62–$####76**
CSOR3 **$####5A**
CSOR0 **$####4E**

15	14	13	12	11	10	9			6	5	4	3		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.

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.

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

R/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

DSACK — Data and Size Acknowledge

This field specifies the source of DSACK when chip-select cycles emulate asynchronous bus cycles, and controls wait state insertion. Refer to **7.5.2 Wait States and DSACK Generation** and **7.6 Chip Selects and Dynamic Bus Sizing** for additional information.

Table B-10 DSACK Encoding

Encoding	Description
%0000	0 Wait States
%0001	1 Wait State
%0010	2 Wait States
%0011	3 Wait States
%0100	4 Wait States
%0101	5 Wait States
%0110	6 Wait States
%0111	7 Wait States
%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 Termination
%1111	External DSACK

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.

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 in interrupt-acknowledge cycles (SPACE = 00). During user- or supervisor-space cycles, this field selects program or data space.

Table B-11 IPL Encoding

IPL	Interrupt Level (SPACE = 00)	Data/Program Space (SPACE = 01, 10, 11)
000	All	Data or Program
001	Level 1	Data Space
010	Level 2	Program Space
011	Level 3	Reserved
100	Level 4	Reserved
101	Level 5	Data Space
110	Level 6	Program Space
111	Level 7	Reserved

$\overline{\text{AVEC}}$ — 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.

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