SECTION 4
SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)

4.1 Overview

MC68F375 contains the single chip integration module 2 (SCIM2E). The SCIM2E consists of several submodules:

- The system configuration block controls MCU configuration and operating mode.
- The system clock generates clock signals used by the SCIM2E, other IMB modules, and external devices. Circuitry is included to detect loss of the phase-locked loop (PLL) reference frequency and to control clock operation in low power stop mode.
- The system protection block provides bus and software watchdog monitors. It also incorporates a periodic interrupt generator that supports execution of time-critical control routines.
- The external bus interface (EBI) handles the transfer of information between the CPU32 and external address space. Ports A, B, E, F, G, and H comprise the EBI and may be used for discrete I/O subject to the MCU's operating mode.
- The chip-select block provides nine general-purpose chip-select signals and two emulation support chip-select signals. Each general-purpose chip select has associated base address and option registers that control the programmable characteristics of the chip select. Chip-select pins can also be used as general-purpose output port C.

The MC68F375 SCIM2E includes improvements to the regular SCIM. This enhanced SCIM includes improvements to the clock synthesizer. These changes are defined in detail in **4.3.2 Clock Synthesizer Submodule**. Please refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for more details on the characteristics of this module.

The SCIM2E has three basic operating modes:

- 16-bit expanded mode
- 8-bit expanded mode
- Single-chip mode

Operating mode is determined by the logic states of specific MCU pins during reset. Refer to **4.7.8 Operating Configuration Out of Reset** for more detailed information on MCU operating modes.

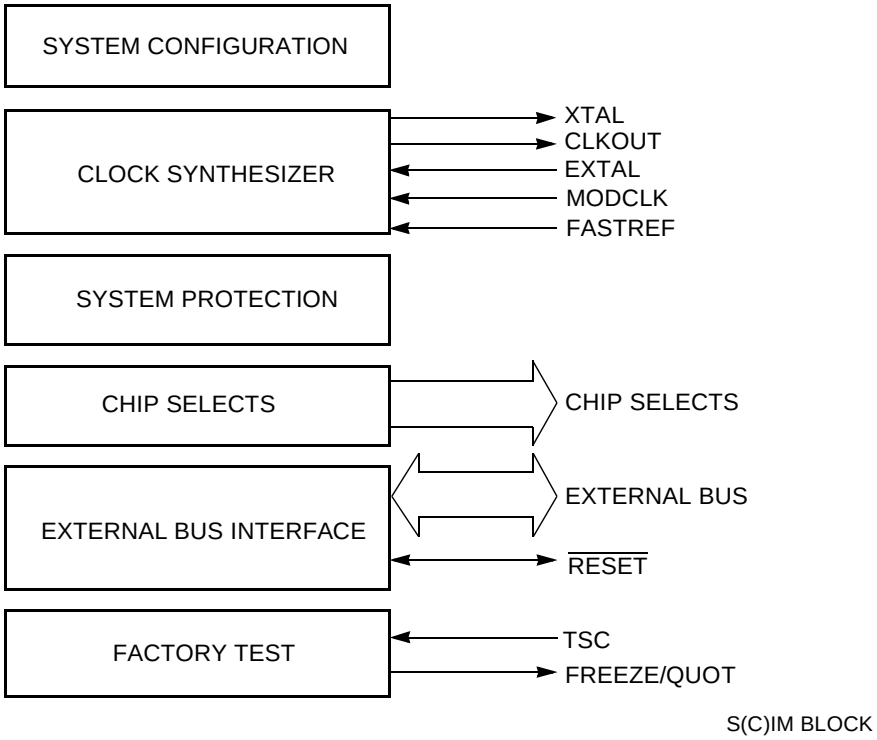


Figure 4-1 SCIM2E Block Diagram

4.2 System Configuration

The MCU can operate as a stand-alone device in single-chip mode, or it can operate with the support of external memory and/or peripheral devices in the 16-bit or 8-bit expanded modes. System configuration is determined by asserting MCU pins during reset and by setting bits in the SCIM2E configuration register (SCIMMCR).

4.2.1 SCIM Module Configuration Register

SCIMMCR — SCIM Module Configuration Register **0xYF FA00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
EXOFF	FRZSW	FRZBM	CPUD[1]	0[2]	SLOWE	SHEN		SUPV	MM	ABD[1]	RWD[1]	IARB			
RESET:															
0	1	1	*	0	0	0	0	1	1	*	*	1	1	1	1

NOTES:
1. Reset state is mode-dependent. Refer to the following bit descriptions.
2. Ensure that initialization software does not change this value (it should always read zero).

Table 4-1 SCIMMCR Bit Descriptions

Bit(s)	Name	Description
15	EXOFF	External clock off. 0 = The CLKOUT pin is driven during normal operation. 1 = The CLKOUT pin is placed in a high-impedance state.
14	FRZSW	Freeze software enable. Enables or disables the software watchdog and periodic interrupt timer during background debug mode when FREEZE is asserted. 0 = Enables the software watchdog and periodic interrupt timer when FREEZE is asserted. 1 = Disables the software watchdog and periodic interrupt timer when FREEZE is asserted.
13	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.
12	CPUD	CPU development support disable. CPUD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. 0 = Instruction pipeline signals available on pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$. 1 = Pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ placed in high-impedance state unless a breakpoint occurs.
11	—	Reserved
10	SLOWE	Slow mode enable. Control bit which forces pins on the chip to operate in fast mode regardless of how they are set up from the controlling module. Slow mode is enabled by setting this bit. 0 = Pins setup by the controlling module to operate in slow mode will operate in fast mode. 1 = Pins will operate at the normal speed controlled by the module.
9:8	SHEN	Show cycle enable. The SHEN field determines how the external bus is driven during internal transfer operations. A show cycle allows internal transfers to be monitored externally. **Table 4-3** indicates whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external devices must not be selected during show cycles.
7	SUPV	Supervisor/user data space. The SUPV bit places the SCIM2E global registers in either supervisor or user data space. 0 = Registers access controlled by the SUPV bit accessible in either supervisor or user mode. 1 = Registers access controlled by the SUPV bit restricted to supervisor access only.
6	MM	Module mapping 0 = Internal modules are addressed from 0x7FF000 – 0x7FFFFF. 1 = Internal modules are addressed from 0xFFF000 – 0xFFFFFF.
5	ABD	Address Bus Disable. ABD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. ABD can be written only once after reset. 0 = Pins ADDR[2:0] operate normally. 1 = Pins ADDR[2:0] are disabled.
4	RWD	Read/write disable. RWD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. RWD can be written only once after reset. 0 = R/$\overline{\text{W}}$ signal operates normally 1 = R/$\overline{\text{W}}$ signal placed in high-impedance state.
3:0	IARB	Each module that can generate interrupts, including the SCIM2E, has an IARB field. Each IARB field can be assigned a value from 0x0 to 0xF. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level. The reset value of the SCIM2 IARB field is 0xF, the highest priority. This prevents SCIM2 interrupts from being discarded during system initialization.

The SCIMMCR register controls the system configuration. SCIMMCR can be read or written at any time, except for the module mapping (MM) bit, which can only be written once after reset, and the reserved bit, which is read-only. Write has no effect.

4.2.2 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 SCIMMCR determines where the control register block is located in the system memory map. When MM = 0, register addresses range from 0x7FF000 to 0x7FFFFF; when MM = 1, register addresses range from 0xFFF000 to 0xFFFFFF.

4.2.3 Interrupt Arbitration

Each module that can request interrupts has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention will take place whenever an interrupt request is acknowledged, even when there is only a single request pending. For an interrupt to be serviced, the appropriate IARB field must have a non-zero value. If an interrupt request from a module with an IARB field value of 0b0000 is recognized, the CPU32 will start to process the interrupt. The CPU will attempt to run and IACK cycle. Because the IARB values of the interrupting module is 0b0000, the module cannot cause the termination of the IACK cycle. In this case, the IACK cycle can only be terminated by an external DSACK, a software watchdog timeout or a bus error. If the IACK cycle is terminated by BERR, a spurious interrupt exception is taken.

Because the SCIM2E routes external interrupt requests to the CPU32, the SCIM2E IARB field value is used for arbitration between interrupts of the same priority. The reset value of IARB for the SCIM2E is 0b1111, and the reset IARB value for all other modules is 0b0000. This prevents SCIM2E interrupts from being discarded during initialization. Refer to **4.8 Interrupts** for a discussion of interrupt arbitration.

4.2.4 Noise Reduction in Single-Chip Mode

Four bits in SCIMMCR control pins that can be disabled in single-chip mode to reduce MCU noise emissions. The characteristics of these control bits are listed in **Table 4-2**. Except for EXOFF, these bits disable their associated pins when the MCU is configured for single-chip mode ($\overline{BERR}$ = 0 during reset).

Table 4-2 SCIMMCR Noise Control Bits

Bit Mnemonic	Position in SCIMMCR	Function	Reset State
EXOFF	15	Disables CLKOUT when set to one.	0
CPUD	12	Disables $\overline{IPIPE}$/DSO and $\overline{IFETCH}$/DSI pins when set to one.	Inverted state of the $\overline{BERR}$ pin
ABD	5	Disables ADDR[2:0] when set to one.	
RWD	4	Disables R/$\overline{W}$ when set to one.	

EXOFF disables the CLKOUT external clock output pin by placing the pin in a high-impedance state. CLKOUT is enabled at power-up unless explicitly disabled by writing a zero to EXOFF.

CPUD disables the IPIPE/DSO and IFETCH/DSI instruction tracking pins by placing them in a high-impedance state when the MCU is not in background debug mode (BDM). When the MCU enters BDM and FREEZE is asserted, IPIPE/DSO and IFETCH/DSI become active and serve as the BDM serial I/O lines.

ABD and RWD disable the ADDR[2:0] and R/W pins by placing them in a high-impedance state. These pins should be disabled because they cannot be used for discrete I/O and have no use in single-chip mode.

4.2.5 Show Internal Cycles

A show cycle allows internal bus transfers to be monitored externally. The SHEN field in SCIMMCR determines what the external data bus interface does during internal transfer operations. **Table 4-3** shows whether data is driven externally, and whether external bus arbitration can occur. The external address bus is always driven. Refer to **4.6.6.1 Show Cycles** for more information.

Table 4-3 Show Cycle Enable Bits

SHEN[1:0]	Effect
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

4.2.6 FREEZE Assertion Response

When the CPU32 enters background debug mode, the IMB FREEZE signal is asserted. The FRZ[1:0] bits in SCIMMCR control the behavior of the software watchdog, periodic interrupt timer, and bus monitor in response to FREEZE assertion. By default, these protection mechanisms are disabled in BDM; they can be selectively enabled by the FRZ[1:0] bits as shown in **Table 4-4**.

Table 4-4 Effects of FREEZE Assertion

FRZ[1:0]		Disabled Elements
0	0	None
0	1	Bus monitor
1	0	Software watchdog and periodic interrupt timer
1	1	Both

4.3 System Clock

The system clock in the SCIM2E provides timing signals for IMB modules and the external bus interface. MC68F375 MCUs are fully static MCU designs; register and memory contents are not affected by clock rate changes. System hardware and software support clock rate changes during operation.

CAUTION

The SCIM2E system clock is different from the SCIM. Do not refer to SCIM documentation for SCIM2E clock information.

4.3.1 System Clock Sources

The system clock signal can be generated from one of three sources. An internal phase-locked loop (PLL) can synthesize the clock from either a slow or fast reference. The clock signal can also be input directly from an external 2X frequency source.

The slow reference is typically a 32.768 KHz crystal; the fast reference is typically a 4 to 8 MHz crystal. The slow and fast references may be provided by sources other than a crystal. Keep these clock sources in mind while reading the rest of this section.

The system clock source is determined upon $\overline{\text{RESET}}$ assertion by the state of the V_{DDSYN}/MODCLK and FASTREF/PF0 pins. In addition to selecting the system clock source, these pins govern the functionality of the W, X, and Y bits in the synthesizer control register (SYNCR) and the equation that determines the MCU operating frequency. **Table 4-5** summarizes system clock source information. For more information, see **4.3.6 Clock Synthesizer Control Register**.

Table 4-5 System Clock Sources

V_{DDSYN}/ MODCLK	FASTREF/ PF0	Clock Mode	SYNCR W/X/Y Bit Assignments and Reset Values	MCU Operating Frequency Equation
0	X[1]	External Clock	W : has no effect X=0b1 : Divider Y[2:0] = 0b000 : Divider	$f_{sys} = \frac{\frac{f_{ref}}{2}}{(2-X)(2^{Y})}$, for $Y \leq 7$
1	0	Slow Reference	W = 0b0 : Multiplier X = 0b0 : Divider Y[5:0] = 0b11111 : Multiplier	$f_{sys} = 4f_{ref}(Y+1)(2^{(2W+X)})$
1	1	Fast Reference	W[2:0] = 0b011 : Multiplier X = 0b0 : Divider Y[2:0] = 0b111 : Divider	$f_{sys} = \frac{f_{ref}(W+1)}{(2-X)(2^{Y})}$, for $Y \leq 7$

NOTES:
1. In external clock mode, the FASTREF/PF0 pin has no effect on clock operation.

The parameter "f_{ref}" refers to the frequency of the clock source connected to the EXTAL pin. The parameter "f_{SYS}" refers to the operating frequency of the MCU and has a defined relationship to f_{ref} that depends on the clock mode selected during reset.

4.3.2 Clock Synthesizer Submodule

The MC68F375 contains an improved version of the clock synthesizer subsystem. The new architecture accommodates both slow or fast crystal references, see **4.3.1 System Clock Sources**. Range of operation and power consumption were taken into

consideration when this new architecture was defined. Compatibility with the previous architecture has been retained when possible.

In general, the improvements fall into four basic categories as follows:

- Configurable PLL for optimization of divider chain based on mode of operation;
- Improved loss-of-clock circuitry based on an independent RC oscillator;
- Improved lock detect circuitry;
- Improved noise immunity by the addition of a V_{SSSYN} pin.

There are three modes for generating the system clock for the MCU. The system clock may be driven directly into the EXTAL pin (external clock mode), it may be generated on-chip by a phase locked loop (PLL) frequency synthesizer using either a slow or fast reference mode. For modes using the PLL, a lock detect circuit detects that the PLL is on frequency and sets a register flag. The PLL mode is determined at reset and behaves according to **Table 4-5**. The source of the system clock is determined at reset by the state of the FASTREF/PF0 and V_{DDSYN}/MODCK pins. To enable external clock mode, the V_{DDSYN}/MODCK pin must be tied directly to V_{SS} at all times

The MC68F375's clock architecture supports the three modes of operation by optionally reconfiguring the number and location of the W bit and Y bit divider stages. In slow reference mode, one W bit and six Y bits are located in the PLL feedback path, enabling frequency multiplication by a factor of up to 2048. The X bit is located in the VCO clock output path to enable dividing the system clock frequency by two without disturbing the VCO and thus requiring re-lock. In fast reference mode, three W bits are located in the PLL feedback path, enabling frequency multiplication by a factor from 1 to 8. Three Y bits and the X bit are located in the VCO clock output path to provide the ability to slow the system clock without disturbing the PLL. In external clock mode, three Y bits and the X bit are located between the EXTAL input and the system clock, to allow slowing the clock for reduced power consumption. Refer to **Figure 4-4**, **Figure 4-3**, and **Figure 4-2** for block diagrams of the architecture in these modes. The reset value of the W, X and Y bits are determined by the clock mode as shown in **Table 4-6**.

NOTE

The crystal oscillator and frequency synthesizer circuits are powered from a separate power pin pair (V_{DDSYN} and V_{SSSYN}) to allow the oscillator to continue to run when the rest of the chip is powered down. This allows avoidance of crystal start-up time. Separate supplies also help improve noise immunity.

The filter for both PLL modes consists of resistor R1 connected in series with capacitor C1. This combination is connected between V_{DDSYN} and XFC. A second capacitor, C2, is also connected between V_{DDSYN} and XFC.

The following sections describe the clock sub-module in detail. One rule applies to all modes of operation — the actual VCO core frequency (before being divided down by X and/or Y bits) must stay at or below the maximum allowable system clock frequency. When changing frequencies, ensure that the values written to the W, X, and Y bits do not select VCO core frequencies above that value. If a system clock frequency is to be

changed with multiple writes to SYNCR, the write sequences should select lower VCO core frequencies first, and the higher VCO core frequencies last unless it is certain the write sequence will not result in VCO core frequencies above the maximum allowable system clock frequency.

4.3.3 Slow Reference Mode

In slow reference mode, the system clock is generated by the PLL typically from a 32.768-KHz reference. The frequency of the system clock is controlled by programming the X, Y, and W bits according to **Table 4-6**.

The W bit is in the feedback path of the VCO. When clear, this bit multiplies the reference frequency by two, and when set, by eight. This bit is clear at reset.

The Y bit divider is a 6-bit modulo counter which can multiply the reference input frequency by up to 256, providing a large number of programmable system clock frequencies. The Y bits are all set to one at reset, providing the highest frequency system clock for a given combination of X and W bits.

The X bit is in the output path of the VCO. It may be used to divide the system clock by two when clear (and pass it without dividing when set). At reset, this bit is cleared to 0.

In slow reference mode, the W and Y bits are both in the feedback path of the PLL. Changing the value of these bits requires the PLL to relock (with some delay) at the new frequency. Changing the X bit, however, will change the system clock frequency without having to wait for the PLL to relock.

Note that for slow reference mode, the crystal is not constrained to be 32.768 KHz but must be in the range of 25 KHz to 50 KHz to ensure that the on-chip crystal oscillator will work.

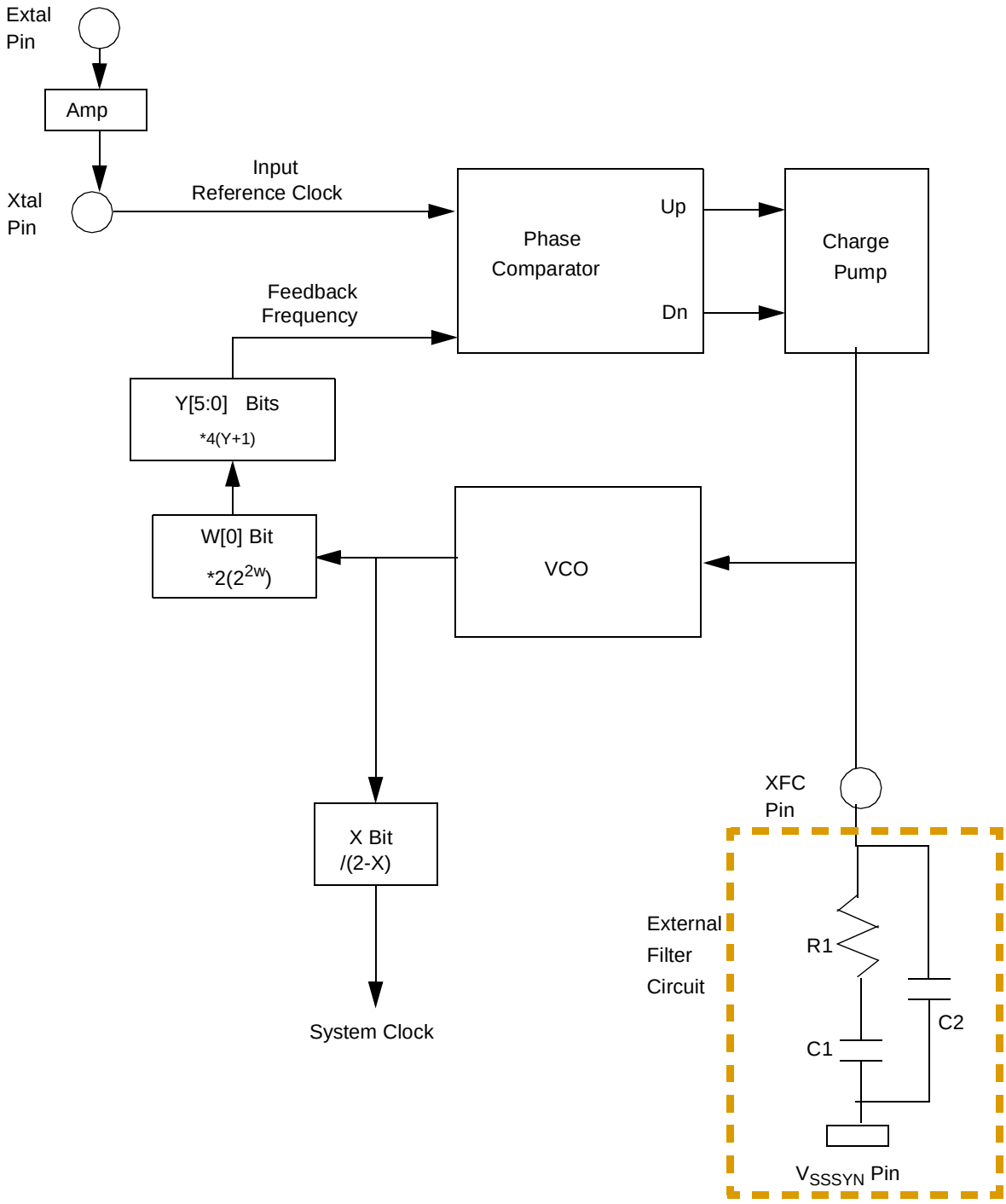


Figure 4-2 Slow Reference Mode

Figure 4-2 depicts the architecture of the system clock generation circuitry when in this mode. **Table 4-6** gives the range of system clock frequencies which may be generated using a 32.768-KHz crystal in this mode. The frequency representing the reset configuration is shaded. Configurations of the PLL in which the system clock or the

VCO core frequency can exceed maximum frequency specification are not supported and are left blank, as a reminder, in the table for frequencies above 25 MHz. Select operating frequencies from the table which do not violate the maximum system clock frequency specification.

Table 4-6 CLKOUT Frequency: Slow Reference; 32.768 KHz Reference

Y (bits)	X=0, W=0[1]	X=1, W=0[2]	X=0, W=1[1]	X=1, W=1[2]	Y (bits)	X=0, W=0[1]	X=1, W=0[2]	X=0, W=1[1]	X=1, W=1[2]
0=000000	131,072	262,144	524,288	1,048,576	32=100000	4,325,376	8,650,752	17,301,504	34,603,008
1=000001	262,144	524,288	1,048,576	2,097,152	33=100001	4,456,448	8,912,896	17,825,792	35,651,584
2=000010	393,216	786,432	1,572,864	3,145,728	34=100010	4,587,520	9,175,040	18,350,080	36,700,160
3=000011	524,288	1,048,576	2,097,152	4,194,304	35=100011	4,718,592	9,437,184	18,874,368	37,748,736
4=000100	655,360	1,310,720	2,621,440	5,242,880	36=100100	4,849,664	9,699,328	19,398,656	38,797,312
5=000101	786,432	1,572,864	3,145,728	6,291,456	37=100101	4,980,736	9,961,472	19,922,944	39,845,888
6=000110	917,504	1,835,008	3,670,016	7,340,032	38=100110	5,111,808	10,223,616	20,447,232	40,894,464
7=000111	1,048,576	2,097,152	4,194,304	8,388,608	39=100111	5,242,880	10,485,760	20,971,520	41,943,040
8=001000	1,179,648	2,359,296	4,718,592	9,437,184	40=101000	5,373,952	10,747,904	21,495,808	42,991,616
9=001001	1,310,720	2,621,440	5,242,880	10,485,760	41=101001	5,505,024	11,010,048	22,020,096	44,040,192
10=001010	1,441,792	2,883,584	5,767,168	11,534,336	42=101010	5,636,096	11,272,192	22,544,384	45,088,768
11=001011	1,572,864	3,145,728	6,291,456	12,582,912	43=101011	5,767,168	11,534,336	23,068,672	46,137,344
12=001100	1,703,936	3,407,872	6,815,744	13,631,488	44=101100	5,898,240	11,796,480	23,592,960	47,185,920
13=001101	1,835,008	3,670,016	7,340,032	14,680,064	45=101101	6,029,312	12,058,624	24,117,248	48,234,496
14=001110	1,966,080	3,932,160	7,864,320	15,728,640	46=101110	6,160,384	12,320,768	24,641,536	49,283,072
15=001111	2,097,152	4,194,304	8,388,608	16,777,216	47=101111	6,291,456	12,582,912	25,165,824	50,331,648
16=010000	2,228,224	4,456,448	8,912,896	17,825,792	48=110000	6,422,528	12,845,056	25,690,112	51,380,224
17=010001	2,359,296	4,718,592	9,437,184	18,874,368	49=110001	6,553,600	13,107,200	26,214,400	52,428,800
18=010010	2,490,368	4,980,736	9,961,472	19,922,944	50=110010	6,684,672	13,369,344	26,738,688	53,477,376
19=010011	2,621,440	5,242,880	10,485,760	20,971,520	51=110011	6,815,744	13,631,488	27,262,976	54,525,952
20=010100	2,752,512	5,505,024	11,010,048	22,020,096	52=110100	6,946,816	13,893,632	27,787,264	55,574,528
21=010101	2,883,584	5,767,168	11,534,336	23,068,672	53=110101	7,077,888	14,155,776	28,311,552	56,623,104
22=010110	3,014,656	6,029,312	12,058,624	24,117,248	54=110110	7,208,960	14,417,920	28,835,840	57,671,680
23=010111	3,145,728	6,291,456	12,582,912	25,165,824	55=110111	7,340,032	14,680,064	29,360,128	58,720,256
24=011000	3,276,800	6,553,600	13,107,200	26,214,400	56=111000	7,471,104	14,942,208	29,884,416	59,768,832
25=011001	3,407,872	6,815,744	13,631,488	27,262,976	57=111001	7,602,176	15,204,352	30,408,704	60,817,408
26=011010	3,538,944	7,077,888	14,155,776	28,311,552	58=111010	7,733,248	15,466,496	30,932,992	61,865,984
27=011011	3,670,016	7,340,032	14,680,064	29,360,128	59=111011	7,864,320	15,728,640	31,457,280	62,914,560
28=011100	3,801,088	7,602,176	15,204,352	30,408,704	60=111100	7,995,392	15,990,784	31,981,568	63,963,136
29=011101	3,932,160	7,864,320	15,728,640	31,457,280	61=111101	8,126,464	16,252,928	32,505,856	65,011,712
30=011110	4,063,232	8,126,464	16,252,928	32,505,856	62=111110	8,257,536	16,515,072	33,030,144	66,060,288
31=011111	4,194,304	8,388,608	16,777,216	33,554,432	63=111111	8,388,608	16,777,216	33,554,432	67,108,864

NOTES:

1. VCO core frequency = 4 times the value in the table.
2. VCO core frequency = 2 times the value in the table. VCO core must not exceed 2x f_{SYS}.

4.3.4 Fast Reference Mode

In fast reference mode, the system clock is generated by the PLL from a reference frequency much higher than that used in slow reference mode (e.g., four MHz). At reset, the system clock frequency will be equal to twice the reference frequency. This is accomplished by configuring the W bits to multiply by four, and setting the X bit to divide by two for a net result of multiplying by two. The frequency may be multiplied using the W bits or divided using the X and Y bits to generate the desired system clock frequency. This mode improves stability over the slow reference mode and, like slow reference mode, provides a 50% duty cycle system clock regardless of the reference duty cycle.

The W bits are in the feedback path of the VCO. They can be programmed to multiply the reference frequency by a factor from one to eight. These bits come out of reset as 0b11, so that the system clock frequency is four times the reference clock frequency. After reset, the W bits can be changed to multiply the reference to the desired system clock frequency. Changing of the W bits will result in PLL unlocking for the PLL lock time specified.

The X bit on the VCO output is used to divide the VCO frequency by two or one. This bit comes out of reset clear (divide by two) so that the system clock frequency is actually one half of what it is multiplied to by the W bits. This bit may be set to increase the system frequency to twice that of the frequency when this bit is clear.

The Y bit divider is a six-stage divider chain in the clock output path, whose output tap is controlled by the three Y register bits. It can divide the output of the VCO down by powers of two to as much as 64, providing a large number of programmable frequencies in this mode.

NOTE

Setting Y to 7 in this mode has the same effect as setting it to 6; the maximum divisor is 2^6. Dividing the PLL output clock with the Y bits reduces the system clock frequency thereby conserving power. Since the X and Y bits are not in the PLL feedback path, the PLL will not have to relock to the new target frequency when they are changed. The Y bits are cleared to all zeros at reset. **Figure 4-3** is a block diagram of the fast reference mode architecture.

Table 4-7 gives example values of the system clock frequency in fast reference mode using a 4.0-MHz reference. The frequency representing the reset configuration is shaded in this table. This frequency ends up being twice the reference frequency when the X bit is cleared, or 8.0 MHz with a 4.0-MHz reference clock. The X bit can then be set to select a system frequency of four times the reference frequency, or 16.0 MHz with a 4.0-MHz reference clock with no PLL re-lock time requirements. Combinations of programmed values for the W, X, and Y bits which would exceed maximum system or VCO core frequencies are not supported and are left blank, as a reminder, in the table for frequencies above 33 MHz.

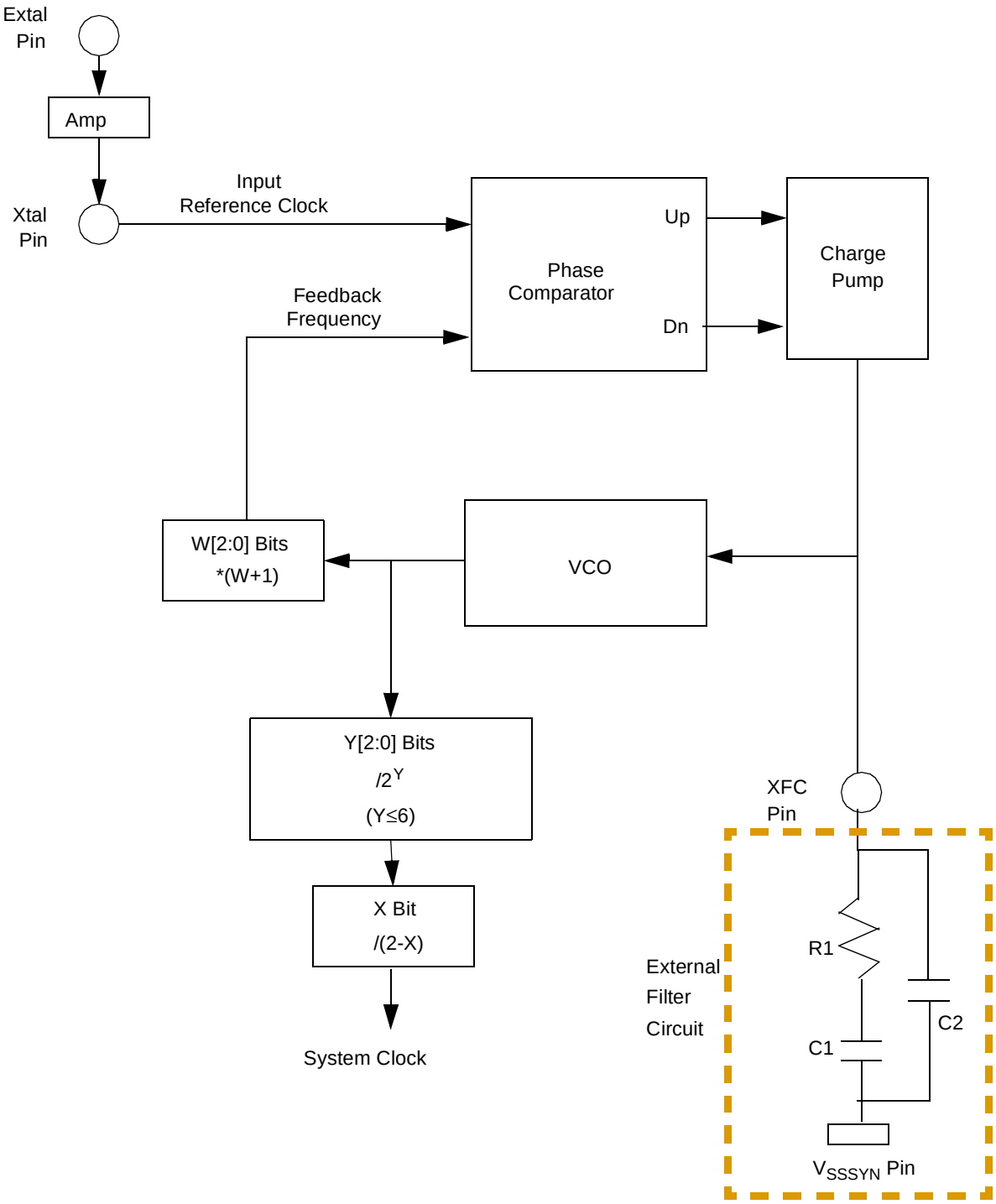


Figure 4-3 Fast Reference Mode

Table 4-7 CLKOUT In Fast Reference Mode with 4.0 MHz Reference

Y X=0	W=000	W=001	W=010	W=011	W=100	W=101	W=110	W=111
0=000	2,000,000	4,000,000	6,000,000	8,000,000	10,000,000	12,000,000	14,000,000	16,000,000
1=001	1,000,000	2,000,000	3,000,000	4,000,000	5,000,000	6,000,000	7,000,000	8,000,000
2=010	500,000	1,000,000	1,500,000	2,000,000	2,500,000	3,000,000	3,500,000	4,000,000
3=011	250,000	500,000	750,000	1,000,000	1,250,000	1,500,000	1,750,000	2,000,000
4=100	125,000	250,000	375,000	500,000	625,000	750,000	875,000	1,000,000
5=101	62,500	125,000	187,500	250,000	312,500	375,000	437,500	500,000
6=110	31,250	62,500	93,750	125,000	156,250	187,500	218,750	250,000
7=111	31,250	62,500	93,750	125,000	156,250	187,500	218,750	250,000

Y X=1	W=000	W=001	W=010	W=011	W=100	W=101	W=110	W=111
0=000	4,000,000	8,000,000	12,000,000	16,000,000	20,000,000	24,000,000	28,000,000	32,000,000
1=001	2,000,000	4,000,000	6,000,000	8,000,000	10,000,000	12,000,000	14,000,000	16,000,000
2=010	1,000,000	2,000,000	3,000,000	4,000,000	5,000,000	6,000,000	7,000,000	8,000,000
3=011	500,000	1,000,000	1,500,000	2,000,000	2,500,000	3,000,000	3,500,000	4,000,000
4=100	250,000	500,000	750,000	1,000,000	1,250,000	1,500,000	1,750,000	2,000,000
5=101	125,000	250,000	375,000	500,000	625,000	750,000	375,000	1,000,000
6=110	62,500	125,000	187,500	250,000	312,500	375,000	187,500	500,000
7=111	62,500	125,000	187,500	250,000	312,500	375,000	187,500	500,000

4.3.5 External Clock Mode

In external clock mode, the clock source, which should be 2x the desired system frequency, must be driven onto the EXTAL pin. This clock is used to generate the system clock directly (the VCO is turned off). At reset, the system clock frequency is one-half the external clock frequency. If this frequency is the the maximum specified system clock frequency, it must not violate strict minimum duty cycle requirements. A block diagram of external clock mode is show in **Figure 4-4**.

In this mode, the six-stage Y divider and the one-stage X divider are placed in the clock output path such that the input clock may be divided down by as much as 128 to produce the system clock. When this is done, it is not necessary to meet the input duty cycle restrictions. The Y bit divider is a six-stage divider chain whose output tap is controlled by the three Y register bits. The X bit divider is a single-stage divider which is bypassed when X is set to 1. X is 1 and Y is 0 after reset, so that the system clock is the same as the external clock.

NOTE

Setting Y to 7 has the same effect as setting it to 6; the maximum divisor is 2^6. The X and Y bit dividers are in the output clock path. Therefore, changing the X or Y bits in this mode causes the frequency to change without a delay.

When the MC68F375 is configured in external clock mode, the VCO will be turned off to save power. Changing the unused W bits will have no effect.

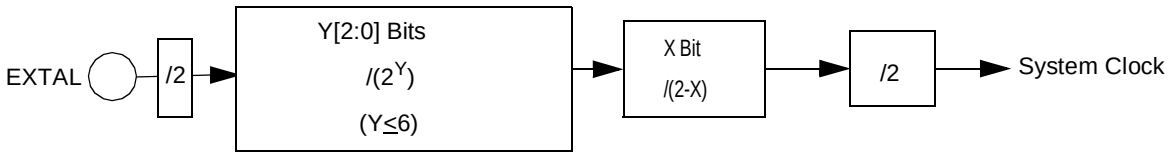


Figure 4-4 External Clock Mode

4.3.6 Clock Synthesizer Control Register

The synthesizer control register (SYNCR) is readable and writable in supervisor mode. The encoding and default value of the upper byte of SYNCR depend on the clock mode selected at reset.

For slow reference mode, the default value is 0x3F which corresponds to an operating frequency of 8.38 MHz for a 32.768-KHz crystal. For fast reference mode, the default value of the upper byte is 0x30, which corresponds to a 2-to-1 match of the reference frequency. For external clock mode, the default value of the upper byte is 0x80, which corresponds to an operating frequency equal to the input frequency on EXTAL. The default value is forced into the SYNCR on reset, along with the default values of the other bits in the register (all zeros).

The encodings and default reset states of the bits in SYNCR in the three clock modes are shown below.

SYNCR — Synthesizer Control Register, Slow Reference Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	W	Y						EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0

SYNCR — Synthesizer Control Register, Fast Reference Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	W[2:0]			—	Y			EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0

SYNCR — Synthesizer Control Register, External Clock Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	Reserved				Y			EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

4.3.6.1 Frequency control Bits (X,W,Y)

Bits [15:8] of the SYNCR control the multiplication or division factors of the synthesizer. X bit [15] controls a one-bit divider which drives the system clock in all modes. When X is set, the divider is bypassed; when clear, the system clock is divided by two. The W bits and the Y bits have different field lengths and functions depending on the clock mode. In slow reference mode, bit [14] is the single W bit, and bits [13:8] are the six Y bits, and both fields are used to multiply the reference frequency. In fast reference mode, bits [14:12] are the three W bits, which are used to multiply the reference frequency. Bits [10:8] are the three Y bits, which are used to divide the PLL output frequency. In external clock mode, bits [14:11] are unused, and bits [10:8] are the three Y bits which are used to divide the input clock frequency. Refer to **Table 4-6** and **Table 4-7** for system frequencies available in common configurations.

4.3.6.2 E Clock Divide Rate (EDIV)

The E clock that goes to the chip select section is driven from a divider circuit off of the same clock source that drives the external clock. This allows turning off or leaving on the E clock in LPSTOP mode using the STEXT bit. When EDIV=0, E is the system clock divided by eight. When EDIV=1, E is the system clock divided by 16. EDIV is cleared to zero by reset.

4.3.6.3 Loss of Clock Oscillator Disable (LOSCD)

An internal oscillator is used in the detection of loss of clock. This oscillator can be disabled by setting this bit. See **4.3.7.5 Loss Of Clock Detect Circuit (LOC)** for details of this feature. When LOSCD = 1, the loss of clock oscillator is disabled. When LOSCD = 0, the loss of clock oscillator is enabled. This bit is cleared to 0 on reset.

4.3.6.4 Limp Mode (SLIMP)

This read only status bit indicates whether the loss of crystal detect logic has detected a loss of system clock. If a loss of clock is detected, the synthesizer will use an internal RC oscillator to derive the system clock and enter limp mode, allowing the MCU to continue to run even without an external clock.

SLIMP=0 indicates that the system clock is being provided normally, either by the PLL or by an external clock from the EXTAL input. SLIMP=1 indicates that a loss of system clock has been detected, and the system clock is being provided from the loss of crystal oscillator reference. See **4.3.7.5 Loss Of Clock Detect Circuit (LOC)**. The limp clock is approximately 16 KHz.

4.3.6.5 Synthesizer Lock (SLOCK)

This read only status bit gives an indication of when the PLL is locked in at the specified frequency. Synthesizer lock occurs when the filter circuit switches from the wide bandwidth to the narrow bandwidth mode (see **4.3.7.2 Phase Comparator and Filter**).

SLOCK=0 is an indication that the PLL is enabled and is not yet locked into the narrow bandwidth mode. If SLOCK=1, it indicates that either the PLL is disabled (system clock is driven in directly), or the PLL is locked. This bit is used by the power on reset circuit to determine whether the clock is stable or not.

4.3.6.6 Reset Enable (RSTEN)

This bit dictates what action to take when the loss of clock logic detects a loss of system clock. If RSTEN=1, a loss of clock will cause a system reset. After completing a normal reset sequence, the part will exit reset and run normally in limp mode. If RSTEN is set while the MC68F375 is currently in limp mode, it will enter reset immediately. If RSTEN=0, when a loss of clock occurs, the part will continue to run normally in limp mode. This bit is cleared to 0 by reset.

4.3.6.7 Low Power Stop Mode SCIM2 Clock (STSCIM)

This bit determines what happens to the SCIM2 clock when the CPU executes the LPSTOP instruction. When STSCIM=0, the SCIM2 clock comes from the crystal oscillator circuit and the VCO is turned off to save power. When STSCIM=1, the SCIM2 clock is driven from the VCO. STSCIM is cleared to 0 by reset.

4.3.6.8 Low Power Stop Mode External Clock (STEXT)

This bit determines what happens to the external clock pin when the CPU executes the LPSTOP instruction. When STEXT=0, no external clock will be driven in LPSTOP mode. When STEXT=1, the external clock is driven from the SCIM2 clock as governed by the STSCIM bit. STEXT is cleared to 0 by reset.

4.3.7 Clock Circuits Operation

For the following discussion, refer to **Figure 4-2**, **Figure 4-3**, and **Figure 4-4**.

4.3.7.1 Synthesizer Circuit

The clocks for the MCU can be derived from an external crystal reference frequency which is multiplied using the phase locked loop, frequency synthesizer circuit as described below.

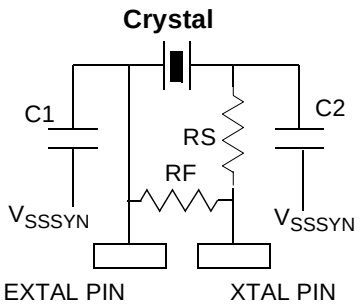
4.3.7.2 Phase Comparator and Filter

The output of the crystal oscillator is compared with the output of the divider chain to determine the frequency relationship of the two signals. The result of this compare is then filtered and used to control the voltage controlled oscillator (VCO).

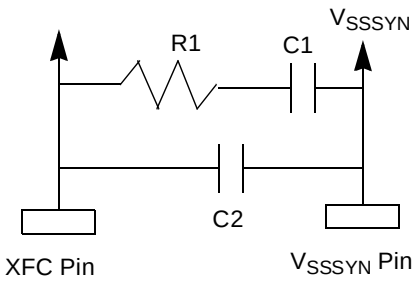
The PLL loop filter has two bandwidths which are automatically selected. When the PLL is first enabled, the wide bandwidth mode is used which enables the PLL frequency to ramp up quickly. Then when the output frequency is near the desired frequency, the filter is switched to the narrow bandwidth to make the final frequency more stable. The PLL requires an external filter network on XFC. **Figure 4-5** shows the suggested values of bypass and PLL external capacitors.

For slow reference mode with a multiplication factor of N=512 (ex: 32.768 KHz reference for a 16.78 MHz f_{SYS}) the values are: R1 = 18 K; C1 = 0.1 µF; and C2 = 3300 pF. For fast reference mode with a multiplication factor of N=4 (ex: 4.194 MHz reference and 16.78 MHz f_{SYS}) the values are: R1 = 20 K; C1 = 1000 pF; and C2 = 100 pF. Reference frequencies and f_{SYS} values very different from those stated may require different filter values (first order affect is multiplication factor N, based on W and Y bits in the SYCNR register). Leakage from the XFC pin must not be greater than that of a 15-mΩ resistor to meet jitter specifications. If the PLL is not enabled, then the XFC filter is not required and the pin may be left unconnected.

The VCO will oscillate at a frequency dictated by the voltage coming out of the filter circuit. To save system power, the VCO is shut off when it is not needed.



Slow Reference Mode (32.768 KHz): RF=1-10 mΩ
RS=330 KΩ, C1=C2=22 pF
Fast Reference Mode (4.194 MHz):RF=1-10 mΩ
RS=1.5 KΩ, C1=C2=27 pF
Fast Reference Mode (5.438 MHz):RF=1-10 mΩ
RS=400 Ω, C1=C2=10 pF
Location of RS can be inboard (as shown here) or outboard of the crystal.
NOTE: Actual circuit should be reviewed with crystal manufacturer.



Slow Reference Mode (32.768 KHz):

$$R1 = R_0 \cdot \sqrt{(Y+1) \cdot 2^{(2W+1)}}; R_0 = 475.2,$$

C1=0.1 µF, C2=3300 pF

Fast Reference Mode (4.194 - 5.438 MHz):

$$R1 = R_0 \cdot \sqrt{(W+1)}; R_0 = 1197,$$

C1=4000 pF, C2=330 pF

Figure 4-5 Crystal Oscillator and External Capacitor Configuration

4.3.7.3 Lock Detect Circuit

The clock generator subsystem on the MC68F375 also includes an improved lock detect circuit. This lock detect does not depend on frequency over-shoot as did the original circuit. It is also more precise than the original circuit. Basic operation is based on two counters and a “frequency match” requirement. When the VCO feedback frequency ramps to within approximately 3.5% of the reference frequency, a count-down time-out is initiated. At the end of that period, the PLL is considered locked and the SLOCK bit is set in the SYNCR register. This same logic will reset the SLOCK bit if the VCO feedback frequency drifts outside of approximately ± 3.5% range. The actual trip points for going into and out of lock depend on the phase relationship of the reference frequency and the VCO feedback frequency.

4.3.7.4 Clock Control Circuit

The clock control circuit generates the following system clock signals:

- EXTCLK is the external system clock which is driven out on the CLKOUT pin. This signal is also used to generate the E clock which is used by the chip selects.
- ICLOCK is the internal module system clock. This clock is used by all of the internal modules of the MCU except for the SCIM2. This clock is stopped in LPSTOP mode.
- SCLOCK is the SCIM2 module’s primary clock. This clock is used by all sections of the SCIM2 except those that must continue to operate in LPSTOP. This clock is stopped in LPSTOP mode.
- SCIMCLK: This is the SCIM2’s secondary system clock which is used by sections of the SCIM2 which continue to operate in LPSTOP mode, such as timers in the system protection block.

4.3.7.5 Loss Of Clock Detect Circuit (LOC)

The loss of clock (LOC) feature is designed to detect the condition in which the SCIM2 SCIMCLK system clock falls in frequency to a range between 20 KHz and 150 Hz. The LOC circuit uses an independent, free-running RC oscillator as a time base to monitor the system clock. The LOC circuit is used as a system protection feature to monitor the operation of the crystal reference for the PLL, or the presence of an external clock signal.

4.3.8 Basic Operation

The SCIM2 internal system clock, SCIMCLK, is monitored by a loss-of-clock subsystem. SCIMCLK remains running in LPSTOP at the PLL frequency if STSCIM=1, or at the crystal frequency if STSCIM=0, or at the external clock frequency if the part is in External Clock mode. The LOC detector should always be triggered if SCIMCLK falls below 150 Hz, and should never be triggered if SCIMCLK is running above 20 KHz. This specified range provides a sufficiently large window of uncertainty to compensate for variations in the RC oscillator frequency due to processing and operating conditions.

The LOC detector operates in either PLL or external clock modes. When it is triggered, limp mode is entered, the SLIMP bit is set, and an alternate clock is provided as the system clock until edges are detected on the crystal/external clock input. The alternate clock is the output of an RC oscillator which is also used as the time-base for the LOC detector. All clock switching is done synchronously, such that no short pulses, or glitches, are caused on the system clock.

4.3.8.1 POR Characteristics

When the power-on-reset logic detects the application of power or the return of power after a power loss situation, the internal RC oscillator is selected as the system clock source. The RC oscillator begins to operate at relatively low levels of VDD and typically several milliseconds before the crystal oscillator/VCO will begin to function. This provides system clocks as soon as possible, allowing I/O pins and modules to reach defined reset states as soon as possible. After the crystal oscillator and VCO start-up is recognized by the LOC logic, a few more RC clock edges followed by a few VCO clock edges will switch the system clock source control logic over from the RC oscillator to the crystal oscillator/VCO subsystem. Exact switching time depends on the RC oscillator frequency and the crystal/VCO clock frequency.

4.3.8.2 External Clock Operating Mode

If SCIMCLK stops in external clock mode, the LOC detect triggers, and the system clock is switched to the alternate clock until the external clock input restarts.

4.3.8.3 PLL Operating Mode

When the PLL is being used to provide the system clock and the crystal reference input stops, the synthesized clock frequency will drop below the LOC threshold. When this occurs, the system clock source is switched to the alternate clock until the crystal reference restarts. If and when the crystal restarts, the clock source will switch back to the PLL, in unlocked mode. The PLL will then relock to the reference frequency.

4.3.8.4 RSTEN Bit Operation

The RSTEN bit is used to put the part in reset if loss of clock is detected. The reset sequence clears the RSTEN bit, and when the sequence finishes, the part exits reset and runs in limp mode. Setting the RSTEN bit while in limp mode will cause reset immediately. This will allow for a continuously reset the part as long as it is in limp mode, by setting the RSTEN bit after exiting reset.

4.3.8.5 Reset Conditions

To save power, the RC oscillator can be disabled by setting the LOSCD bit in the SYNCR register. In this case, the ability to detect loss of clock is disabled. However, if the RESET pin is driven low, the RC oscillator is forced on to provide a system clock, ensuring that external RESET will be recognized even in a DC state. For more information, see **4.7 Reset**.

In order to reset the ports to their post reset state listed in **Table 4-8**, an internal clock and the reset signal must be present. The clocks are generated with the SCIM2E voltage controlled oscillator (VCO). The VCO is biased to operate at approximately eight KHz whenever the crystal oscillator is not detected. This feature causes the VCO to start before the crystal oscillator. (See **APPENDIX E ELECTRICAL CHARACTERISTICS** for the exact frequencies.)

Table 4-8 Port Reset Condition

Port	State of Pins after Reset
A[1]	Input
B	Input
G	Input
H	Input
E	Input
F	Input
C	Output (PC[6:2, 0]) are driven high
PQS0, PQS1, PQS2	Input
PQA, PQB	Input
TPU3	Input

NOTES:
1. Each port requires approximately 4 clocks to assume their post reset state. The VCO startup time is no more than 15 msec after VDD reaches minimum value.

Only single byte or aligned word writes on the IMB to the RAM module will be guaranteed to complete without data corruption for synchronous resets. A long-word write, a misaligned operand write, a write to a peripheral module other than the RAM or a read cycle are not guaranteed. External writes are also guaranteed to complete, provided the external configuration logic on the data bus is conditioned by R/W. Asynchronous reset sources usually indicate a catastrophic failure and require the reset control logic to assert reset to the system immediately.

4.3.8.6 Low Power Operation

Low power operation is initiated by the CPU32. To reduce power consumption selectively, the CPU32 can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU32 can execute the LPSTOP instruction which causes the SCIM2E to turn off the system clock.

A loss of clock will be recognized while the part is in low power stop, unless the RC oscillator is disabled. If it is disabled, external RESET will re-enable it so that RESET will be recognized. If a loss of clock occurs in LPSTOP mode and RSTEN=0, the part will continue to operate normally on the alternate clock. LPSTOP can then be exited normally, either by an interrupt request or by external RESET. If RSTEN=1 in LPSTOP mode, the loss of clock event will cause reset. For more information, see **4.4.9 Low Power Stop Mode**.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SCIM2E brings the MCU out of low power stop mode when one of the following exceptions occur:

- RESET
- Trace
- SCIM2E interrupt of higher priority than the stored interrupt mask

Refer to **4.4.9 Low Power Stop Mode** and **4.6.4.2 LPSTOP Broadcast Cycle** for more information.

During low power stop mode, unless the system clock signal is supplied by an external source and that source is removed, the SCIM2E clock control logic and the SCIM2E clock signal (SCIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ pins are clocked by SCIMCLK. The SCIM2E can also continue to generate the CLKOUT signal while in low power stop mode.

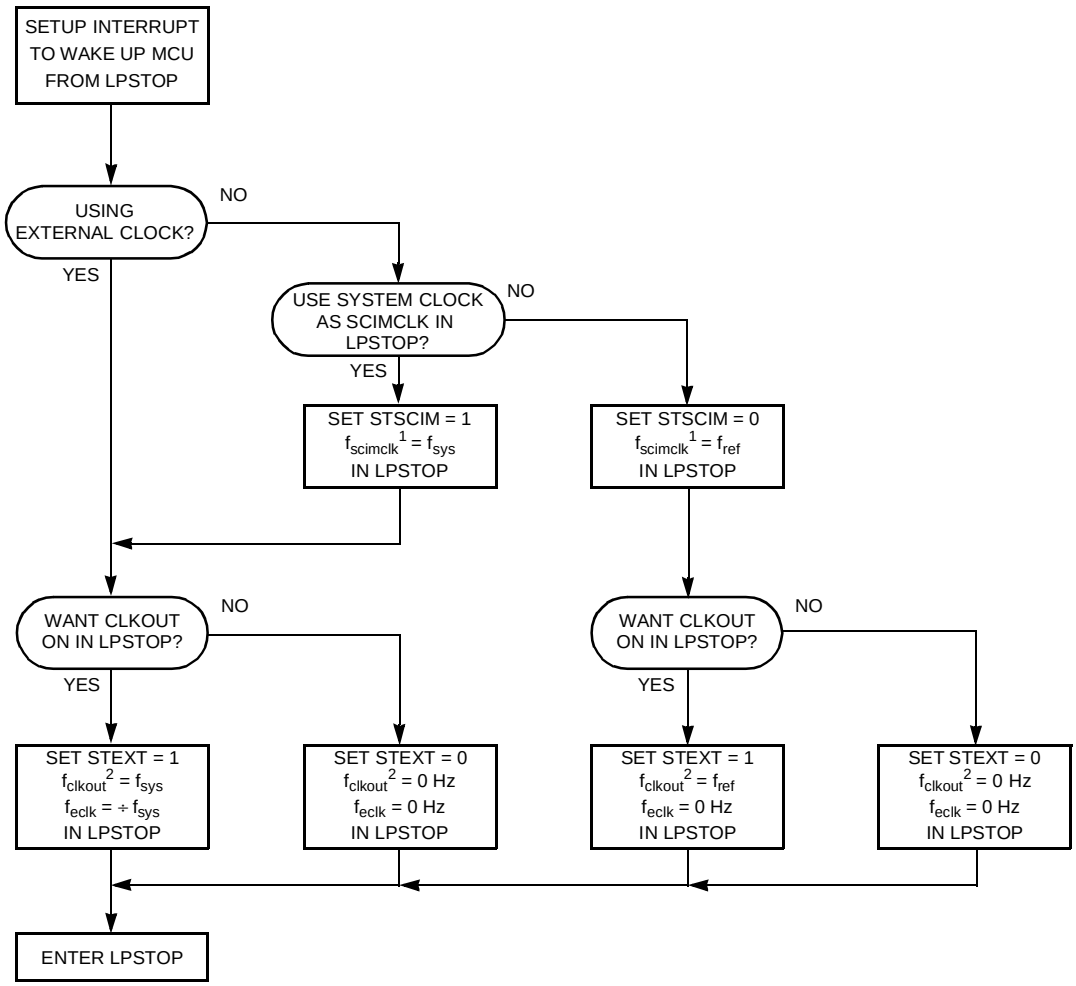
During low power stop mode, the address bus and data bus continue to drive the LPSTOP broadcast cycle, and bus control signals are negated. I/O pins configured as outputs continue to hold their previous state; I/O pins configured as inputs will remain in a high impedance state.

The STSCIM and STEXT bits in SYNCR determine clock operation during low power stop mode.

The flow chart shown in **Figure 4-6** summarizes the effects of the STSCIM and STEXT bits when the MCU enters low power stop mode. Any clock in the off state is held low. If the synthesizer VCO is turned off during low power stop mode, PLL relock delay will occur when the MCU exits LPSTOP mode and the VCO is re-enabled.

NOTE

In LPSTOP mode, the crystal oscillator is not disabled and will continue to run.



1. THE SCIMCLK IS USED BY THE PIT, $\overline{IRQ}$, AND INPUT BLOCKS OF THE SCIM2E.
2. CLKOUT CONTROL DURING LPSTOP IS OVERRIDDEN BY THE EXOFF BIT IN SCIMMCR. IF EXOFF = 1, THE CLKOUT PIN IS ALWAYS IN A HIGH IMPEDANCE STATE AND STEXT HAS NO EFFECT IN LPSTOP. IF EXOFF = 0, CLKOUT IS CONTROLLED BY STEXT IN LPSTOP.

LPSTOPFLOW

Figure 4-6 LPSTOP Flowchart

4.3.8.7 Loss of Reference Signal

The SCIM2E includes circuitry to detect a loss of the synthesizer f_{ref} signal and to force reset or to allow continued operation from an alternate clock source. The LOSCD, SLIMP, and RSTEN bits in SYNCR control and report the behavior of the clock synthesizer when f_{ref} loss is detected.

In the SCIM2E, f_{ref} is compared to the output of an independent free-running RC oscillator to detect failure. The loss of clock detector should always be triggered when f_{ref} falls below 150 Hz and should never be triggered when f_{ref} is above 20 KHz. This range provides a window of uncertainty sufficiently large enough to compensate for

variations in the output of the RC oscillator due to processing and/or operating conditions.

The loss of clock detector can be disabled by writing a one to the loss of clock oscillator bit (LOSCD). This disables the free-running RC oscillator and prevents f_{ref} loss from being detected. The reset state of LOSCD is zero which enables the RC oscillator and loss of clock detector.

The reset enable bit (RSTEN) determines how the MCU will process a loss of clock detection. The default state out of reset for RSTEN is zero. This forces the clock synthesizer into the limp mode operating state. In limp mode, the RC oscillator used by the loss of clock detector provides the system clock. Limp mode frequency varies from device-to-device but does not exceed one-half the maximum system clock frequency.

When set to one, RSTEN allows the clock synthesizer to reset the MCU when the loss of clock detector triggers. After powering-up from a loss of clock reset, the MCU will set the LOC bit in the reset status register (RSR) and begin operation in limp mode.

The limp status bit (SLIMP) in SYNCR indicates that f_{ref} has failed and that the MCU has entered limp mode. SLIMP will remain set until normal f_{ref} operation is restored.

4.4 System Protection

The system protection block reports reset status information, monitors internal bus activity, and provides periodic interrupt generation. **Figure 4-7** is a block diagram of the submodule.

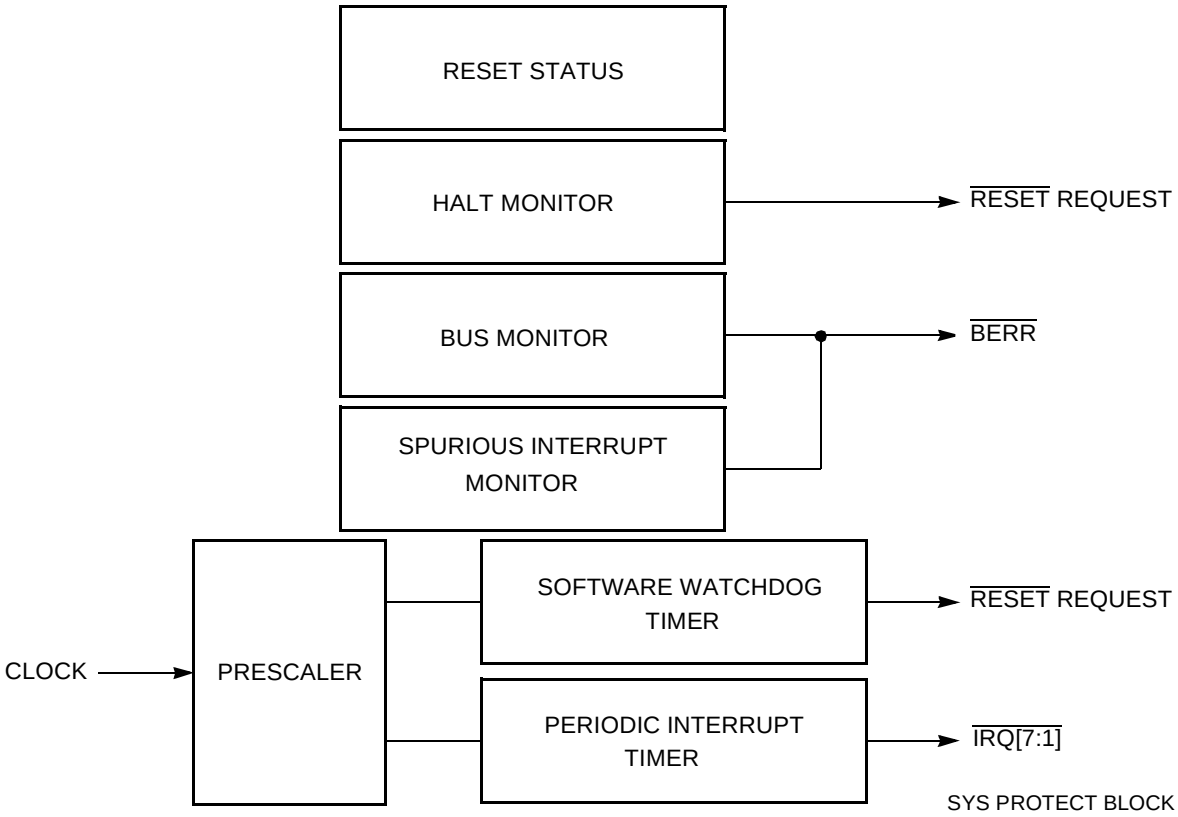


Figure 4-7 System Protection

4.4.1 System Protection Control Register

SYPCR — System Protection Control Register **0xYF FA21**

7	6	5	4	3	2	1	LSB 0
SWE	SWP	SWT[1:0]		HME	BME	BMT[1:0]	
1	MODCLK	0	0	0	0	0	0

Table 4-9 SYPCR Bit Descriptions

Bit(s)	Name	Description
7	SWE	Software watchdog enable 0 = Software watchdog is disabled. 1 = Software watchdog is enabled.
6	SWP	Software watchdog prescaler. This bit controls the value of the software watchdog prescaler. The reset value of SWP is the complement of the state of the MODCLK pin during reset. 0 = Software watchdog clock is not prescaled. 1 = Software watchdog clock is prescaled by 512.
5:4	SWT	Software watchdog timing. This field selects the divide ratio used to establish the software watchdog timeout period. Refer to **Table 4-12**.

Table 4-9 SYPCR Bit Descriptions (Continued)

Bit(s)	Name	Description
3	HME	Halt monitor enable 0 = Halt monitor is disabled. 1 = Halt monitor is enabled.
2	BME	Bus monitor external enable 0 = Disable bus monitor for external bus cycles. 1 = Enable bus monitor for external bus cycles.
1:0	BMT	BMT[1:0] — Bus Monitor Timing. This field selects the bus monitor timeout period. Refer to **Table 4-10**.

4.4.2 Reset Status

The reset status register (RSR) latches MCU status during reset. Refer to **4.7.4 Reset Status Register** for more information.

4.4.3 Bus Monitor

The internal bus monitor checks data size acknowledge ($\overline{\text{DSACK}}$) or autovector ($\overline{\text{AVEC}}$) signal response times during normal bus cycles. The monitor asserts the internal bus error ($\overline{\text{BERR}}$) signal when the response time is excessively long.

$\overline{\text{DSACK}}$ 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[1:0]) field in the system protection control register (SYPCR). **Table 4-10** shows the periods allowed.

Table 4-10 Bus Monitor Period

BMT[1:0]	Bus Monitor Timeout 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 CPU32 initiates a bus cycle. The BME bit in 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.

When monitoring transfers to an 8-bit port, the bus monitor does not reset until both byte accesses of a word transfer are completed. Monitor timeout period must be at least twice the number of clocks that a single-byte access requires.

4.4.4 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus when a double bus fault occurs. A flag in the reset status register (RSR) will indicate when the last reset was caused by the halt monitor. Halt monitor reset can be inhibited by

the halt monitor (HME) enable bit in SYPCR. Refer to **4.6.5.2 Double Bus Faults** for more information.

4.4.5 Spurious Interrupt Monitor

During interrupt exception processing, the CPU32 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 CPU32 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled.

Refer to **4.8 Interrupts** for further information. For detailed information about interrupt exception processing, refer to **4.8.1 Interrupt Exception Processing**.

4.4.6 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in 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 $\overline{\text{RESET}}$ signal.

Each time the service sequence is written, the software watchdog timer restarts. The sequence to restart the software watchdog requires the following steps:

- Write 0x55 to SWSR
- Write 0xAA to SWSR

Both writes must occur before timeout in the order listed. Any number of instructions can be executed between the two writes.

The clock rate of the watchdog timer is affected by clock mode, the software watchdog prescale (SWP) bit, and the software watchdog timing (SWT[1:0]) field in SYPCR. In slow reference mode and external clock mode, f_{ref} or $f_{ref} \div 512$ can be used to clock the watchdog timer. The options in fast reference mode are $f_{ref} \div 128$ or $(f_{ref} \div 128) \div 512$. In all cases, the divide-by-512 option is selected when SWP = 1.

The value of SWP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 4-11**. System software can change SWP value.

Table 4-11 SWP Reset States

V_{DDSYN}/MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

SWT[1:0] selects the divide ratio used to establish the software watchdog timeout period.

The following equation calculates the timeout period in slow reference mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

The following equation calculates the timeout period in fast reference mode:

$$\text{Timeout Period} = \frac{(128)(\text{Divide Ratio Specified by SWP and SWT[1:0]})}{f_{ref}}$$

The following equation calculates the timeout period in external clock mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

Table 4-12 shows the divide ratio for each combination of the SWP and SWT[1:0] bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period can take effect.

Table 4-12 Software Watchdog Divide Ratio

SWP	SWT[1:0]	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}

Figure 4-8 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

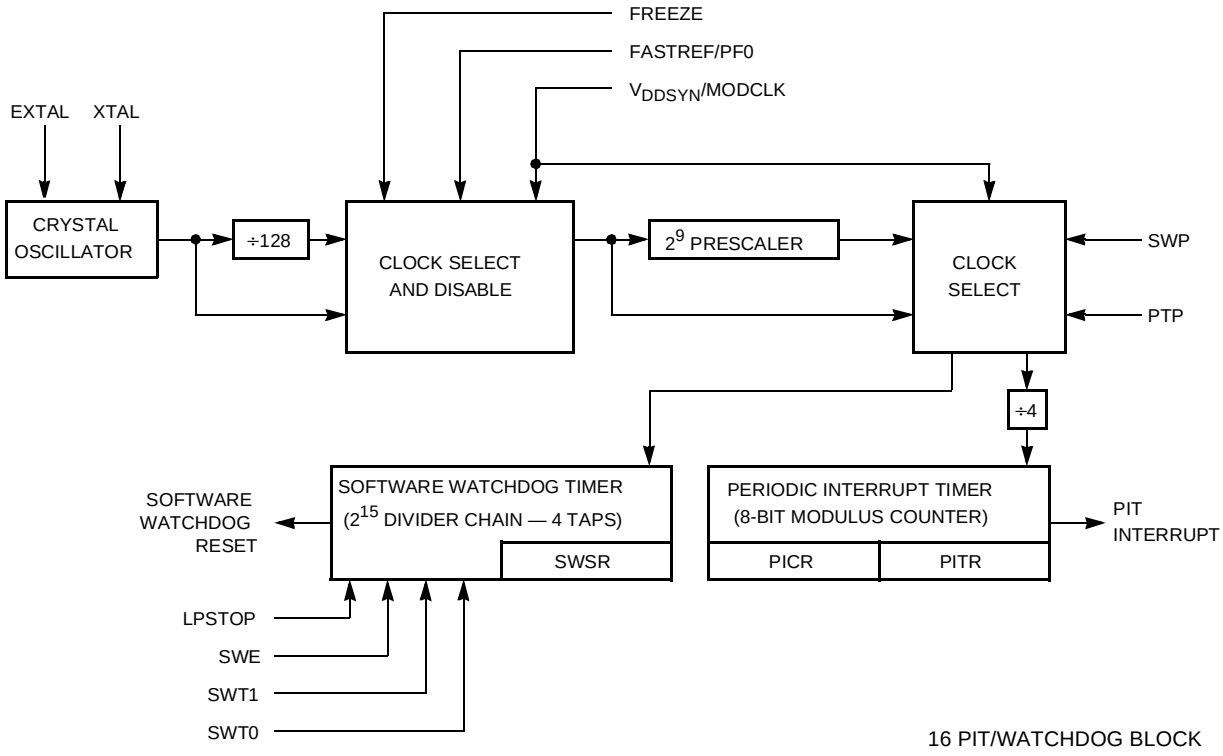


Figure 4-8 Periodic Interrupt Timer and Software Watchdog Timer

4.4.6.1 Software Watchdog Service Register

SWSR — Software Watchdog Service Register[1] **0xYF FA27**

7	6	5	4	3	2	1	LSB 0
SWSR[7:0]							
0	0	0	0	0	0	0	0

NOTES:
1. This register is shown with a read value.

This register can be read or written at any time. Bits [15:8] are reserved and will always read zero.

4.4.7 Periodic Interrupt Timer

The periodic interrupt timer (PIT) allows the generation of 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 PIT consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. Refer to **4.8.1 Interrupt Exception Processing** for further information about interrupt exception processing.

The periodic interrupt timer modulus counter is clocked by one of two signals. When the PLL is enabled, f_{ref} is used in slow reference mode and $f_{ref} \div 128$ is used in fast reference mode. When the PLL is disabled, f_{ref} is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the periodic interrupt timer. One of two options, either no prescaling, or prescaling by a factor of 512, can be selected. The value of PTP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 4-13**. System software can change PTP value.

Table 4-13 PTP Reset States

V_{DDSYN}/MODCLK	PTP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

Either clock signal selected by PTP is divided by four before driving the modulus counter. The modulus counter is initialized by writing a value to the periodic interrupt timer modulus (PITM[7:0]) field in PITR. A zero value turns off the PIT. When the modulus counter reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM[7:0] and counting repeats. If a new value is written to PITR, it is loaded into the modulus counter when the current count is completed.

The following equation calculates the PIT period in slow reference mode:

$$\text{PIT Period} = \frac{(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in fast reference mode:

$$\text{PIT Period} = \frac{(128)(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in external clock mode:

$$\text{PIT Period} = \frac{(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

4.4.8 Interrupt Priority and Vectoring for the Periodic Interrupt Timer

Interrupt priority and vectoring for the PIT are determined by the values of the periodic interrupt request level (PIRQL[2:0]) and periodic interrupt vector (PIV[7:0]) fields in the periodic interrupt control register (PICR).

The PIRQL field is compared to the CPU32 interrupt priority mask to determine whether the interrupt is recognized. **Table 4-14** shows PIRQL[2:0] priority values.

Because of SCIM2E 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 4-14 Periodic Interrupt Priority

PIRQL[2:0]	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 vector number is used to calculate the address of the appropriate vector in the exception vector table. The reset value of the PIV field is 0x0F, which corresponds to the uninitialized interrupt exception vector.

4.4.9 Low Power Stop Mode

When the CPU32 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSCIM bit in SYNCR, and the MCU enters low power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low power stop.

During low power stop mode, 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 the MCU exits low power stop mode. The watchdog is not reset when entering low power stop mode. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITM[7:0] must be loaded with zero before entering LPSTOP. A PIT interrupt, or an external interrupt request, can bring the MCU out of low power stop mode if it has a higher priority than the interrupt mask value stored in the clock control logic when low power stop mode is entered. LPSTOP can be terminated by a reset.

4.4.9.1 Periodic Interrupt Control Register

PICR sets the interrupt level and vector number for the periodic interrupt timer (PIT). Bits [10:0] can be read or written at any time. Bits [15:11] are reserved and always read zero.

PICR — Periodic Interrupt Control Register **0xYF FA22**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	PIRQL[2:0]			PIV[7:0]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 4-15 PICR Bit Descriptions

Bit(s)	Name	Description
15:11	—	**Reserved**
10:8	PIRQL	Periodic interrupt request level. This field determines the priority of periodic interrupt requests. A value of 0b000 disables PIT interrupts.
7:0	PIV	Periodic interrupt vector. This field specifies the periodic interrupt vector number supplied by the SCIM2 when the CPU32 acknowledges an interrupt request.

4.4.9.2 Periodic Interrupt Timer Register

PITR — Periodic Interrupt Timer Register **0xYF FA24**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	0	PTP	PITM[7:0]							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MOD-CLK}}$	0	0	0	0	0	0	0	0

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

Table 4-16 PITR Bit Descriptions

Bit(s)	Name	Description
15:9	—	Reserved
8	PTP	Periodic timer prescaler 0 = Periodic timer clock not prescaled. 1 = Periodic timer clock prescaled by a value of 512.
7:0	PITM	Periodic interrupt timing modulus. This field determines the periodic interrupt rate.

4.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 4-9** shows a basic system with external memory and peripherals.

The external bus has 24 address lines and 16 data lines. The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Port width is the maximum number of bits accepted or provided by the external memory system during a bus transfer. Widths of eight and 16 bits are accessed through the use of asynchronous cycles controlled by the size (SIZ1 and SIZ0) and

data size acknowledge (DSACK1 and DSACK0) pins. Multiple bus cycles may be required for dynamically sized transfers.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic is synchronized with EBI transfers. Refer to **4.9 Chip Selects** for more information.

4.5.1 Bus Control Signals

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 beginning of each bus cycle, the address space, the size of the transfer, and the type of cycle. External devices can decode these signals and respond to transfer data and terminate the bus cycle. The EBI can operate in an asynchronous mode for any port width.

4.5.1.1 Address Bus

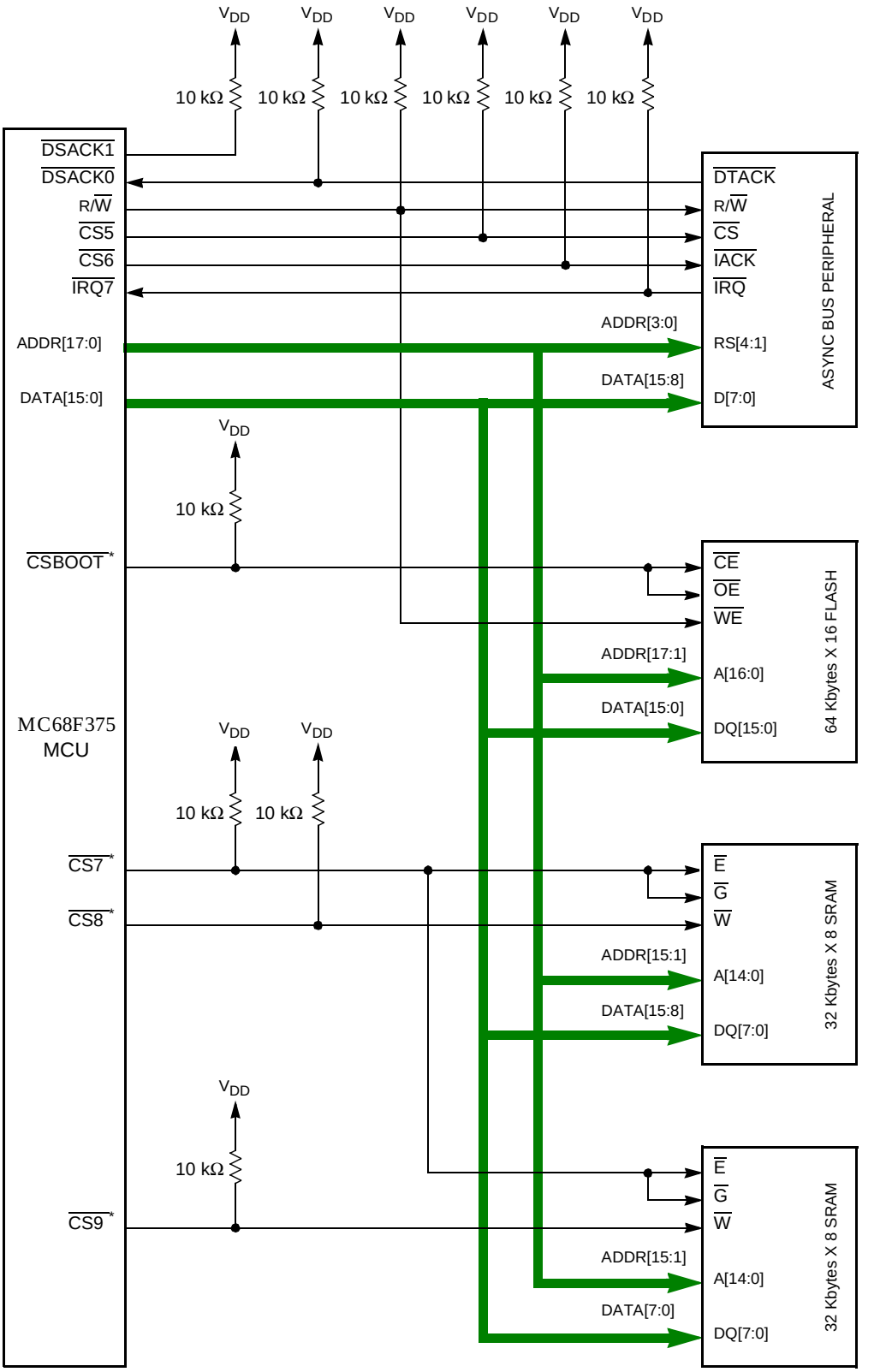
Bus signals ADDR[19:0] define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while AS is asserted.

4.5.1.2 Address Strobe

Address strobe (AS) is a timing signal that indicates the validity of an address on the address bus as well as that of many control signals.

4.5.1.3 Data Bus

DATA[15:0] form a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer eight or 16 bits of data in one bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size.



* THESE CHIP-SELECT LINES ARE CONFIGURED FOR 16-BIT PORT OPERATION IN THIS EXAMPLE.

F396 SCIM2E BUS

Figure 4-9 MCU Basic System

4.5.1.4 Data Strobe

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

4.5.1.5 Read/Write Signal

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

4.5.1.6 Size Signals

Size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while $\overline{AS}$ is asserted. **Table 4-17** shows SIZ0 and SIZ1 encoding.

Table 4-17 Size Signal Encoding

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

4.5.1.7 Function Codes

The CPU generates function code 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. Only supervisor data, supervisor program, user data, user program, and CPU spaces are defined.

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 $\overline{AS}$ is asserted. **Table 4-18** shows address space encoding.

Table 4-18 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

4.5.1.8 Data Size Acknowledge Signals

During normal bus transfers, external devices can assert the data size acknowledge signals ($\overline{\text{DSACK}}$[1:0]) to indicate port width to the MCU. During a read cycle, these signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can terminate. $\overline{\text{DSACK}}$[1:0] can also be supplied internally by chip-select logic. Refer to **4.9 Chip Selects** for more information.

4.5.1.9 Bus Error Signal

The bus error signal ($\overline{\text{BERR}}$) can be asserted by an external source when a bus cycle is not properly terminated by $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ assertion. 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 **4.6.5 Bus Exception Control Cycles** for more information.

The internal bus monitor can generate the $\overline{\text{BERR}}$ signal for excessively long internal-to-external transfers. In systems with an external bus master, the SCIM2E bus monitor must be disabled and external logic must be provided to drive the $\overline{\text{BERR}}$ pin, because the internal $\overline{\text{BERR}}$ monitor has no information about transfers initiated by an external bus master. Refer to **4.6.6 External Bus Arbitration** for more information.

4.5.1.10 Halt Signal

The halt signal ($\overline{\text{HALT}}$) can be asserted by an external device for debugging purposes to cause single bus cycle operation or (in combination with $\overline{\text{BERR}}$) a retry of a bus cycle in error. The $\overline{\text{HALT}}$ signal affects external bus cycles only. As a result, a program not requiring use of the external bus may continue executing, unaffected by the $\overline{\text{HALT}}$ signal. When the MCU completes a bus cycle with the $\overline{\text{HALT}}$ signal asserted, DATA[15:0] is placed in a high-impedance state and $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are driven inactive; the address, function code, size, and read/write signals remain in the same state. The MCU does not service interrupt requests while it is halted. Refer to **4.6.5 Bus Exception Control Cycles** for further information.

4.5.1.11 Autovector Signal

The autovector signal (AVEC) can be used to terminate interrupt acknowledgment cycles for external interrupts only. Assertion of AVEC causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If AVEC is continuously asserted, autovectors are generated for all external interrupt requests. AVEC is ignored during all other bus cycles. Refer to **4.8 Interrupts** for more information. AVEC for external interrupt requests can also be supplied internally by chip-select logic. Refer to **4.9 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **4.6.6 External Bus Arbitration** for more information.

4.5.2 Dynamic Bus Sizing

The MCU dynamically interprets the port size of an addressed device during each bus cycle, allowing operand transfers to or from 8-bit and 16-bit ports.

During a bus transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the DSACK inputs, as shown in **Table 4-19**. Chip-select logic can generate data size acknowledge signals for an external device. Refer to **4.9 Chip Selects** for more information.

Table 4-19 Effect of DSACK Signals

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

If the CPU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the first 16 bits of valid data and then runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the DSACK signals to indicate the port width. For instance, a 16-bit external device always returns DSACK for a 16-bit port (regardless of whether the bus cycle is a byte or word operation).

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 reside on data bus bits [15:0], and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins.

Operand bytes are designated as shown in **Figure 4-10**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes

of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

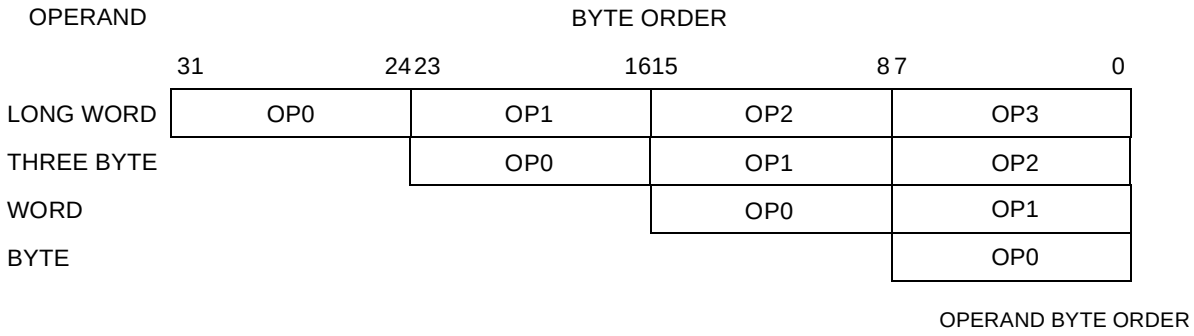


Figure 4-10 Operand Byte Order

4.5.3 Operand Alignment

The EBI data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the SIZ[1:0] and ADDR0 outputs. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During a bus transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

4.5.4 Misaligned Operands

The CPU32 uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (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.

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 through a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word is transferred on a following bus cycle.

The CPU32 does not support misaligned word transfers. An attempt to do so will result in an “address error” exception.

4.5.5 Operand Transfer Cases

Table 4-20 shows how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle. **Table 4-20** also shows

to what states $\overline{\text{DSACK}}$[1:0] must be driven — either by a chip select or by external circuitry — to terminate the given bus cycle.

Table 4-20 Operand Alignment

Current Cycle	Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	DATA [15:8]	DATA [7:0]	Next Cycle
1	Byte to 8-bit port (even)	0	1	0	1	0	OP0	(OP0)[1]	—
2	Byte to 8-bit port (odd)	0	1	1	1	0	OP0	(OP0)	—
3	Byte to 16-bit port (even)	0	1	0	0	1	OP0	(OP0)	—
4	Byte to 16-bit port (odd)	0	1	1	0	1	(OP0)	OP0	—
5	Word to 8-bit port (aligned)	1	0	0	1	0	OP0	(OP1)	2
6	Word to 8-bit port (misaligned	1	0	1	1	0	OP0	(OP0)	1
7	Word to 16-bit port (aligned)	1	0	0	0	1	OP0	OP1	—
8	Word to 16-bit port (misaligned)	1	0	1	0	1	(OP0)	OP0	3
9	Long-word to 8-bit port (aligned)	0	0	0	1	0	OP0	(OP1)	13
10	Long-word to 8-bit port (misaligned)[2]	1	0	1	1	0	OP0	(OP0)	1
11	Long-word to 16-bit port (aligned)	0	0	0	0	1	OP0	OP1	7
12	Long-word to 16-bit port (misaligned)[2]	1	0	1	0	1	(OP0)	OP0	3
13	Three byte to 8-bit port[3]	1	1	1	1	0	OP0	(OP0)	5

NOTES:
1. Operands in parentheses are ignored by the CPU32 during read cycles.
2. The CPU32 does not support misaligned operand transfers.
3. Three byte transfer cases occur only as a result of an aligned long word to 8-bit port transfer.

4.6 Bus Operation

Internal microcontroller modules are typically accessed in two system clock cycles. Regular external bus cycles use handshaking between the MCU and external peripherals to manage transfer size and data. These accesses take a minimum of three system clock cycles, with no wait states. During regular cycles, wait states can be inserted as needed by bus control logic. Refer to **4.6.2 Regular Bus Cycle** for more information.

Fast-termination cycles, which are two clock external accesses with no wait states, use chip-select logic to generate handshaking signals internally. Refer to **4.6.3 Fast Termination Cycles** and **4.9 Chip Selects** for more information. Bus control signal timing, as well as chip-select signal timing, is specified in **APPENDIX E ELECTRICAL CHARACTERISTICS**. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for more information about each type of bus cycle.

4.6.1 Synchronization to CLKOUT

External devices connected to the MCU bus can operate at a clock frequency different from the frequencies of the MCU as long as the external devices satisfy the interface signal timing constraints. Although bus cycles are classified as asynchronous, they are interpreted relative to the MCU system clock output (CLKOUT).

Descriptions are made in terms of individual system clock states, labelled {S0, S1, S2,..., SN}. 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. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information.

Bus cycles terminated by $\overline{\text{DSACK}}$ assertion normally require a minimum of three CLKOUT cycles. To support systems that use CLKOUT to generate $\overline{\text{DSACK}}$ and other inputs, asynchronous input setup time and asynchronous input hold times are specified. When these specifications are met, the MCU is guaranteed to recognize the appropriate signal on a specific edge of the CLKOUT signal.

4.6.2 Regular Bus Cycle

The following paragraphs contain a discussion of cycles that use external bus control logic. Refer to **4.6.3 Fast Termination Cycles** for more information.

To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. The SIZ signals and ADDR0 are externally decoded to select the active portion of the data bus. Refer to **4.5.2 Dynamic Bus Sizing** for more information. When $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle), then asserts a $\overline{\text{DSACK}}$[1:0] combination to indicate the port size.

The $\overline{\text{DSACK}}$[1:0] signals can be asserted before the data from a peripheral device is valid on a read cycle. To ensure valid data is latched by the MCU, a maximum period between MCU assertion of $\overline{\text{DS}}$ and supplied assertion of $\overline{\text{DSACK}}$[1:0] is specified.

There is no specified maximum for the period between MCU assertion of $\overline{\text{AS}}$ and supplied assertion of $\overline{\text{DSACK}}$[1:0]. 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 $\overline{\text{DSACK1}}$ or $\overline{\text{DSACK0}}$ goes low.

If the $\overline{\text{DSACK}}$ bus termination signals remain unasserted, the MCU will continue to insert wait states, and the bus cycle will never end. If no peripheral responds to an access, or if an access is invalid, external logic should assert the $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ signals to abort the bus cycle (when $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the CPU32 acts as though only $\overline{\text{BERR}}$ is asserted). When enabled, the SCIM2E bus monitor asserts $\overline{\text{BERR}}$ when $\overline{\text{DSACK}}$ response time exceeds a predetermined limit. The bus monitor timeout period is determined by the BMT[1:0] field in SYPCR. The maximum bus monitor timeout period is 64 system clock cycles.

4.6.2.1 Read Cycle

During a read cycle, the MCU transfers data from an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to read two bytes at once. 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.

Figure 4-11 is a flow chart of a word read cycle. Refer to **4.5.2 Dynamic Bus Sizing**, **4.5.4 Misaligned Operands**, and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

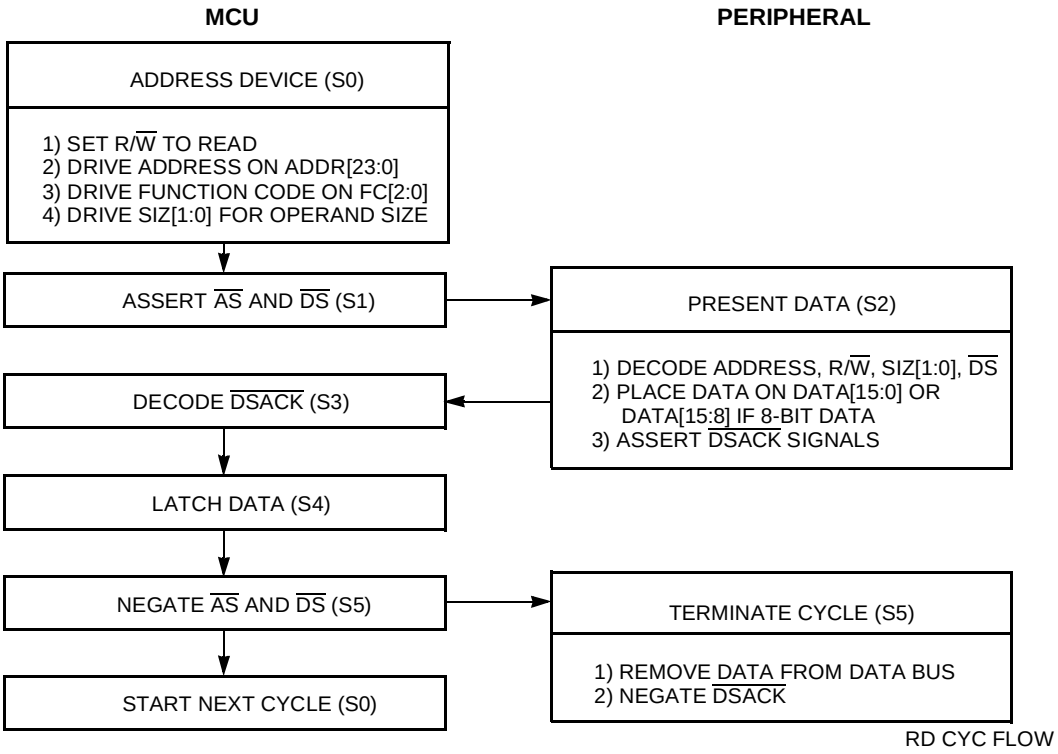


Figure 4-11 Word Read Cycle Flowchart

4.6.2.2 Write Cycle

During a write cycle, the MCU transfers data to an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes at once. For a byte operation, the MCU writes one byte. The portion of the data bus upon which each byte is written depends on operand size, peripheral address, and peripheral port size.

Figure 4-12 is a flow chart of a write-cycle. Refer to **4.5.2 Dynamic Bus Sizing**, **4.5.4 Misaligned Operands**, and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

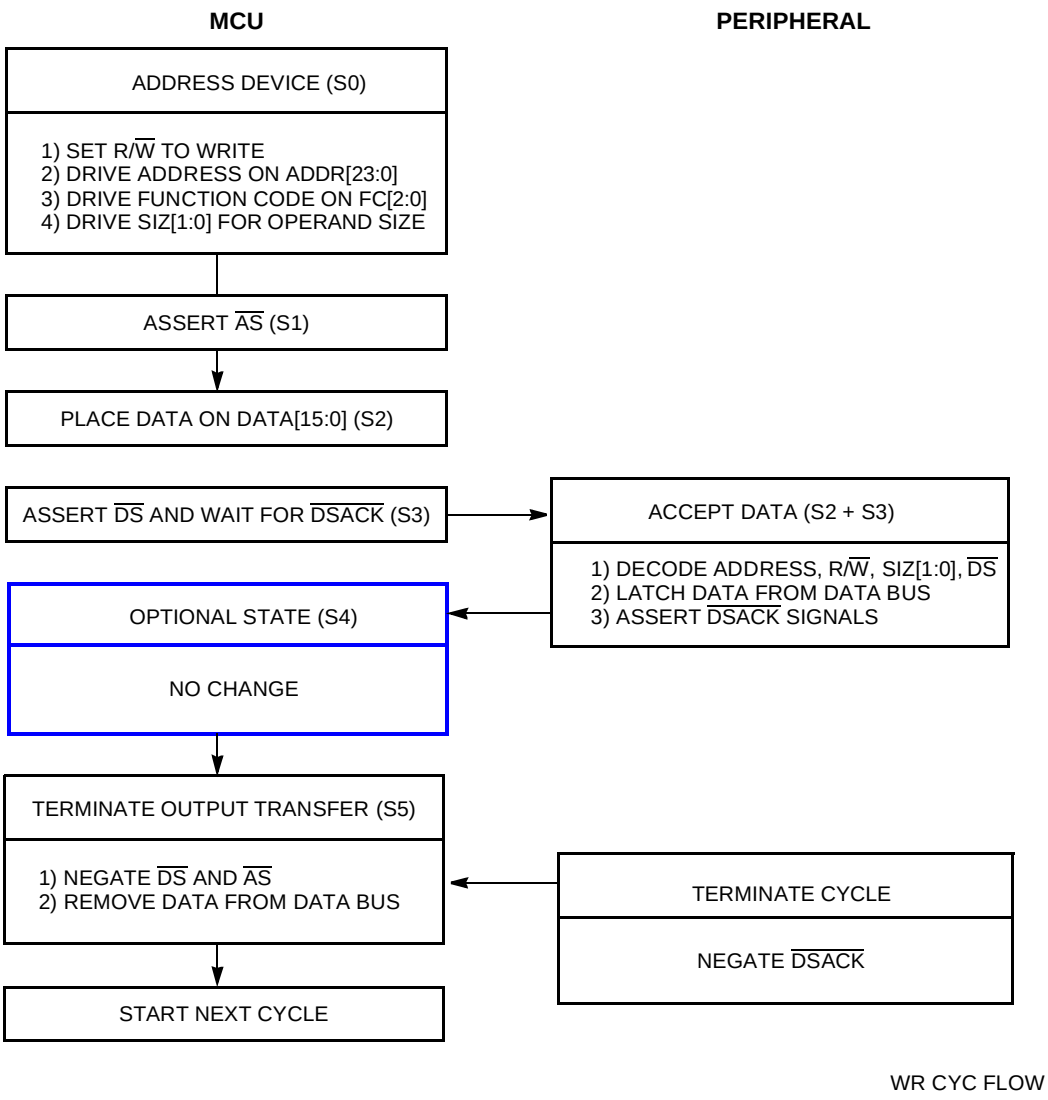


Figure 4-12 Write Cycle Flowchart

4.6.3 Fast Termination Cycles

When an external device can meet fast access timing, the fast termination option of SCIM2E chip selects can provide a two-cycle external bus transfer. Because the chip-select circuits are driven from the system clock, bus cycle termination is inherently synchronized with the system clock.

If multiple chip selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all MODE, STRB, and associated DSACK fields must be programmed to the same value. This prevents a conflict on the internal bus when the wait states are loaded into the DSACK counter shared by all chip-selects.

Fast termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. When AS, DS, and R/W are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle). At the appropriate time, chip-select logic asserts the DSACK[1:0] signals.

The DSACK field in the chip-select option registers determine whether internally generated DSACK or externally generated DSACK is used. For fast termination cycles, the fast termination encoding (0b1110) must be used. Refer to **4.6.3 Fast Termination Cycles** for information about fast termination setup.

The external DSACK lines are always active, regardless of the setting of the DSACK field in the chip-select option registers. Thus, an external DSACK can always terminate a bus cycle. Holding a DSACK line low will cause essentially all external bus cycles to be three-cycle (zero wait states) accesses unless the chip-select option register specifies fast termination accesses.

To use fast termination, an external device must be fast enough to have data ready within the specified setup time (for example, by the falling edge of S4). Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for information about fast termination timing.

When a fast termination cycle is issued, DS is asserted for reads but not for writes. The STRB field in the chip-select option register used must be programmed with the address strobe encoding to assert the chip-select signal for a fast termination write.

4.6.4 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 as CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid only while AS is asserted. Refer to **4.5.1.7 Function Codes** for more information on codes and encoding.

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. Three encodings are used by the MCU, as shown in **Figure 4-13**. These encodings represent breakpoint acknowledge (type 0x0) cycles, low power stop broadcast (type 0x3) cycles, and interrupt acknowledge (type 0xF) cycles. Type 0x0 and type 0x3 cycles are discussed in the following paragraphs. Refer to **4.8 Interrupts** for information about interrupt acknowledge bus cycles.

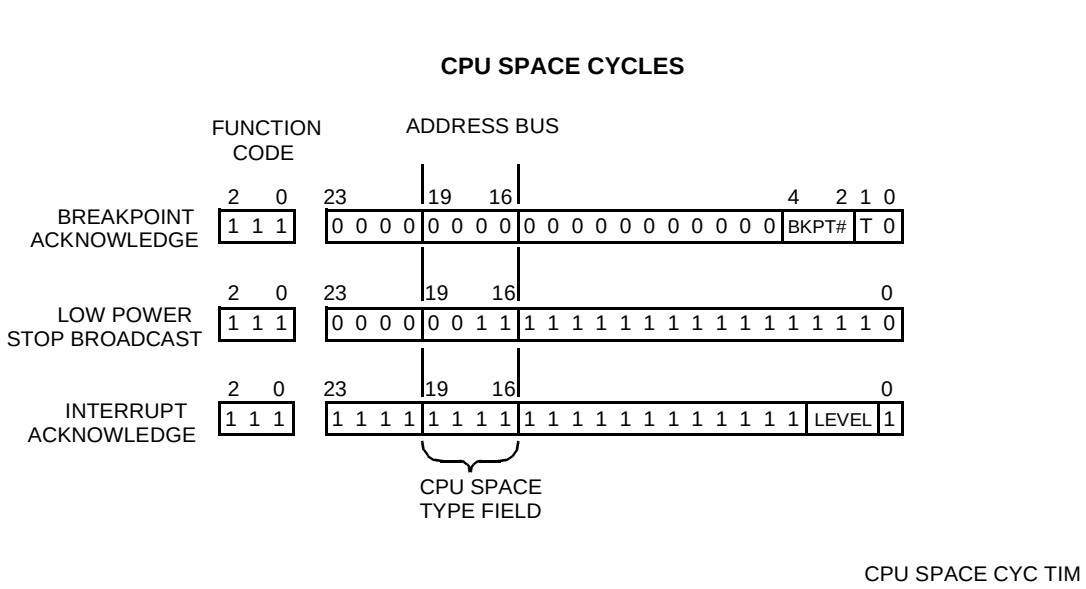


Figure 4-13 CPU Space Address Encoding

4.6.4.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with background debug mode. On the MC68F375 microcontroller, both hardware and software can initiate breakpoints.

The CPU32 BKPT instruction allows breakpoints to be inserted through software. The CPU32 responds to this instruction by initiating a breakpoint acknowledge read cycle in CPU space. It places the breakpoint acknowledge (0b0000) code on ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) on ADDR[4:2], and 0b0 (indicating a software breakpoint) on ADDR1.

External breakpoint circuitry must decode the function code and address lines and responds either by 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 performs illegal instruction exception processing. The CPU32 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.

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. The CPU32 responds by initiating a breakpoint acknowledge read cycle in CPU space. The CPU32 places the breakpoint acknowledge code of 0b0000 on ADDR[19:16], the breakpoint number value of 0b111 on ADDR[4:2], and ADDR1 is set to 0b1, indicating a hardware breakpoint.

External breakpoint circuitry must decode the function code and address lines, place an instruction word on the data bus, and assert $\overline{\text{BERR}}$. 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.

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. 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.

Refer to the ***CPU32 Reference Manual* (CPU32RM/AD)** and the ***SCIM Reference Manual* (SCIMRM/AD)** for additional information. Breakpoint operation flow for the CPU32 is shown in **Figure 4-14**.

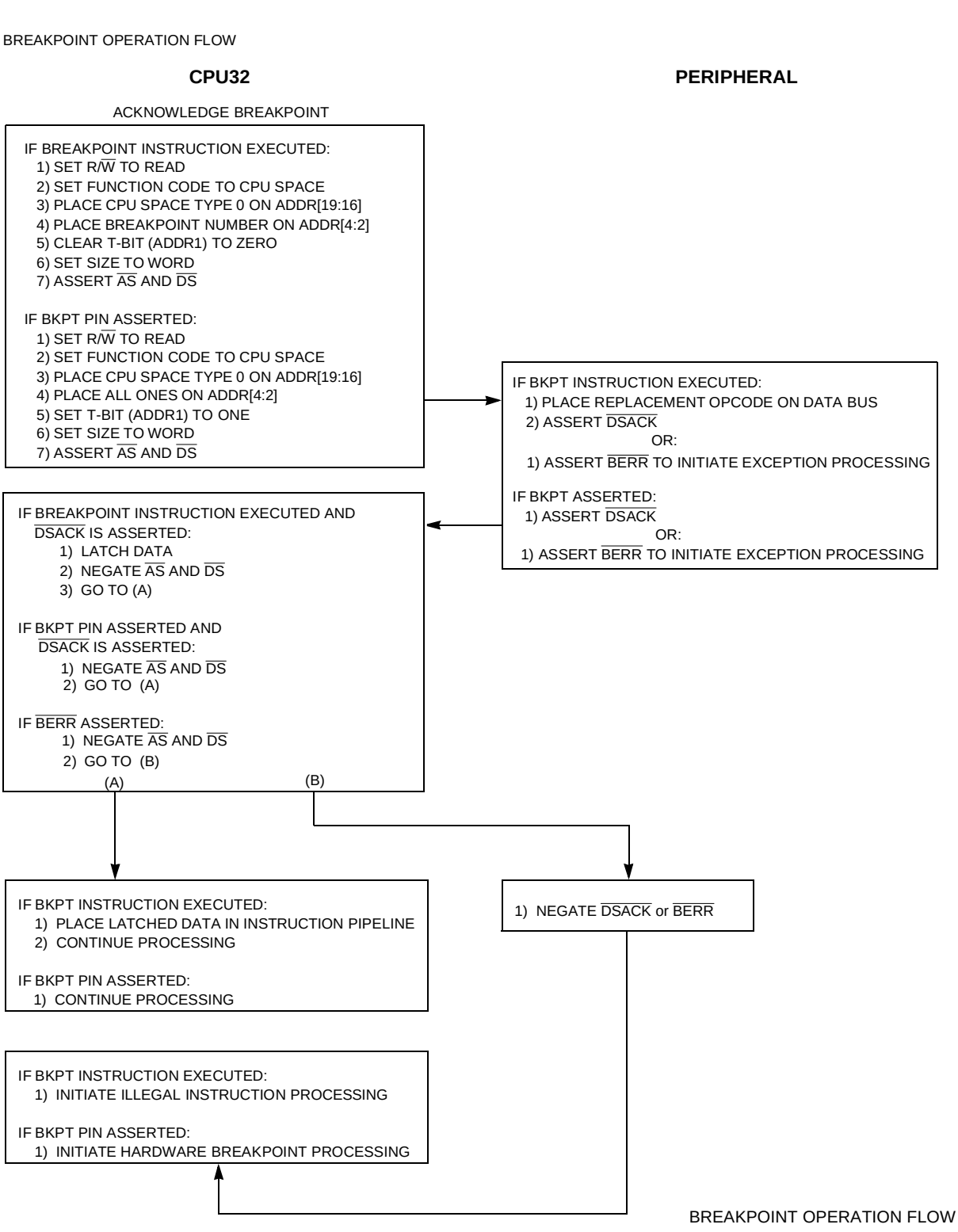


Figure 4-14 Breakpoint Operation Flowchart

4.6.4.2 LPSTOP Broadcast Cycle

Low power stop mode is initiated by the CPU32. Individual modules can be stopped by setting the STOP bits in each module configuration register. The SCIM2E can turn off system clocks after execution of the LPSTOP instruction. When the CPU32 executes LPSTOP, a low power stop broadcast cycle is generated. The SCIM2E brings the MCU out of low power mode when either a reset or an interrupt of higher priority than the interrupt mask level in the CPU32 condition code register occurs.

Refer to **4.3.8.6 Low Power Operation** and **SECTION 3 CENTRAL PROCESSOR UNIT** for more information.

During an LPSTOP broadcast cycle, the CPU32 performs a CPU space write to address 0x3FFFE. 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 4-15**.

The LPSTOP 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 SCIM2E provides an internally generated $\overline{\text{DSACK}}$ response to this cycle. The timing of this bus cycle is the same as for a fast termination write cycle. If the bus is not available (arbitrated away), the LPSTOP broadcast cycle is not shown externally.

NOTE

$\overline{\text{BERR}}$ assertion during the LPSTOP broadcast cycle is ignored.

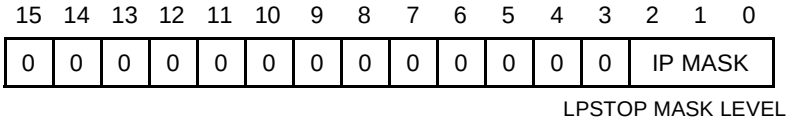


Figure 4-15 LPSTOP Interrupt Mask Encoding on DATA[15:0]

4.6.5 Bus Exception Control Cycles

An external device or a chip-select circuit must assert at least one of the $\overline{\text{DSACK}}$[1:0] signals or the $\overline{\text{AVEC}}$ signal to terminate a bus cycle normally. Bus exception control cycles are used when bus cycles are not terminated in the expected manner.

Acceptable bus cycle termination sequences are summarized as follows. The case numbers refer to **Table 4-21**, which indicates the results of each type of bus cycle termination.

- Normal Termination
  - — $\overline{\text{DSACK}}$ is asserted; $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ remain negated (case 1).
- Halt Termination
  - — $\overline{\text{HALT}}$ is asserted at the same time or before $\overline{\text{DSACK}}$, and $\overline{\text{BERR}}$ remains negated (case 2).
- Bus Error Termination
  - — $\overline{\text{BERR}}$ is asserted in lieu of, at the same time as, or before $\overline{\text{DSACK}}$ (case 3),

or after DSACK (case 4), and HALT remains negated; BERR is negated at the same time or after DSACK.

- Retry Termination
  - HALT and BERR are asserted in lieu of, at the same time as, or before DSACK (case 5) or after DSACK (case 6); BERR is negated at the same time or after DSACK; HALT may be negated at the same time or after BERR.

Table 4-21 shows various combinations of control signal sequences and the resulting bus cycle terminations.

Table 4-21 DSACK, BERR, and HALT Assertion Results

Case Number	Control Signal	Asserted on Rising Edge of State		Result
		N[1]	N + 2	
1	DSACK BERR HALT	A[2] NA[3] NA	S[4] NA X[5]	Normal termination.
2	DSACK BERR HALT	A NA A/S	S NA S	Halt termination: normal cycle terminate and halt. Continue when HALT is negated.
3	DSACK BERR HALT	NA/A A NA	X S X	Bus error termination: terminate and take bus error exception, possibly deferred.
4	DSACK BERR HALT	A A NA	X S NA	Bus error termination: terminate and take bus error exception, possibly deferred.
5	DSACK BERR HALT	NA/A A A/S	X S S	Retry termination: terminate and retry when HALT is negated.
6	DSACK BERR HALT	A NA NA	X A A	Retry termination: terminate and retry when HALT is negated.

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

To control termination of a bus cycle for a retry or a bus error condition properly, DSACK, BERR, and HALT must be asserted and negated with the rising edge of CLKOUT. 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 E 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.

DSACK, BERR, and HALT may be negated after 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.

4.6.5.1 Bus Errors

The CPU32 treats bus errors as a type of exception. Bus error exception processing begins when the CPU32 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.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU32 at the end of the bus cycle in which it was asserted. Because 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/acknowledge is dependent upon 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 is 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 0xFFFF) is latched into the CPU32 instruction register, with indeterminate results.

4.6.5.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. Refer to **3.9 Exception Processing** for more information. However, a special case of bus error, called double bus fault, can abort exception processing.

$\overline{\text{BERR}}$ assertion is not detected until an instruction is complete. The $\overline{\text{BERR}}$ latch is cleared by the first instruction of the $\overline{\text{BERR}}$ exception handler. Double bus fault occurs in three ways:

1. When bus error exception processing begins and a second $\overline{\text{BERR}}$ is detected before the first instruction of the exception handler is executed.
2. When one or more bus errors occur before the first instruction after a reset exception is executed.
3. A bus error occurs while the CPU32 is loading information from a bus error

stack frame during a return from exception (RTE) instruction.

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

Immediately after assertion of a second $\overline{\text{BERR}}$, the MCU halts and drives the $\overline{\text{HALT}}$ line low. Only a reset can restart a halted MCU. However, bus arbitration can still occur. Refer to **4.6.6 External Bus Arbitration** for more information. A bus error or address error that occurs after exception processing has been completed (during the execution of the exception handler routine, or later) does not cause a double bus fault. The MCU continues to retry the same bus cycle as long as the external hardware requests it.

4.6.5.3 Retry Operation

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

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

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

4.6.5.4 Halt Operation

When $\overline{\text{HALT}}$ is asserted while $\overline{\text{BERR}}$ is not asserted, the MCU halts external bus activity after negation of $\overline{\text{DSACK}}$. 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.

Negating and reasserting $\overline{\text{HALT}}$ according to timing requirements provides single-step (bus cycle to bus cycle) operation. The $\overline{\text{HALT}}$ signal affects external bus cycles only, so that a program that 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 $\overline{\text{HALT}}$ is asserted causes the CPU32 to initiate a retry sequence.

When the MCU completes a bus cycle while the $\overline{\text{HALT}}$ signal is asserted, the data bus goes into a 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. However, when external bus arbitration occurs while the MCU is halted, address and control signals go into 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.

4.6.6 External Bus Arbitration

The MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, $\overline{\text{HALT}}$ assertion, and when the CPU32 has halted due to a double bus fault.

The MCU bus controller manages bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert bus arbitration signals in the sequences described in the following paragraphs.

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. The protocol sequence for assuming bus mastership from the MCU is:

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

$\overline{\text{BR}}$ can be asserted 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.

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 the processing of bus requests during data transfer cycles.

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

Refer to **Figure 4-16** which shows bus arbitration for a single device. The flow chart shows $\overline{BR}$ negated at the same time $\overline{BGACK}$ is asserted. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for more information on bus arbitration.

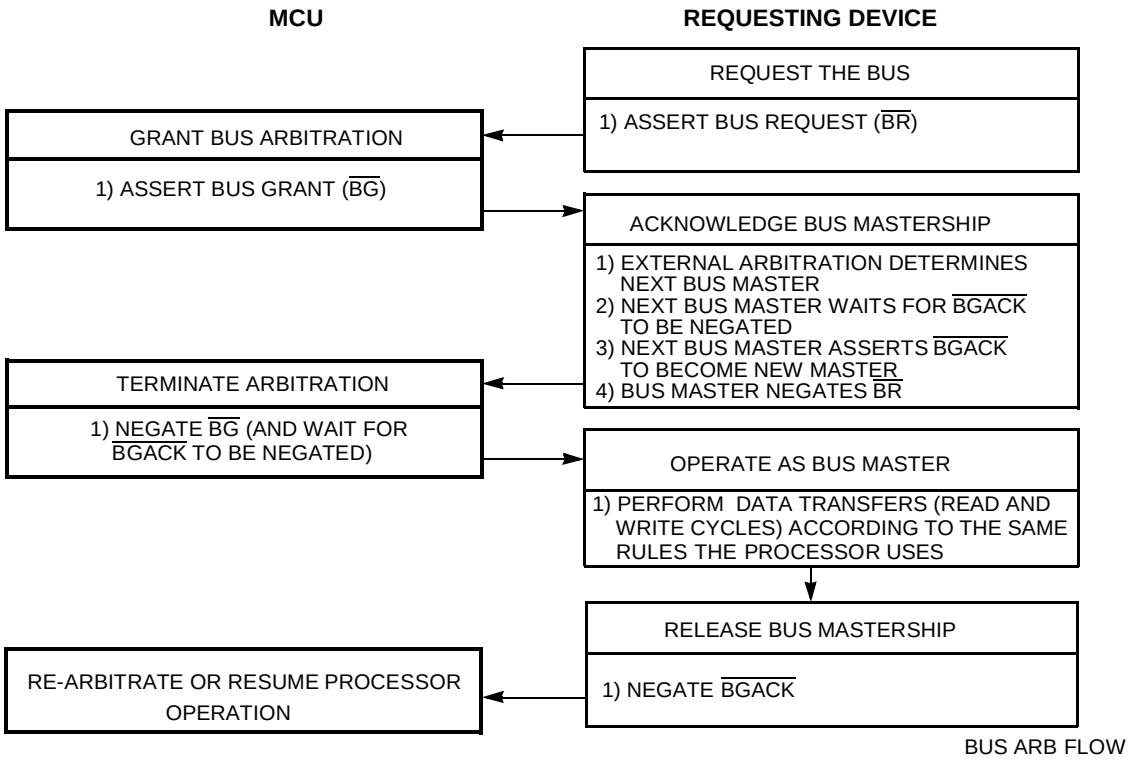


Figure 4-16 Bus Arbitration Flowchart for Single Request

4.6.6.1 Show Cycles

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

Show cycles are controlled by SHEN[1:0] in SCIMMCR. This field set to 0b00 by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but $\overline{AS}$ and $\overline{DS}$ are not asserted externally and external data bus pins are in a high-impedance state during internal accesses. Refer to **4.2.5 Show Internal Cycles** and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

When show cycles are enabled, $\overline{DS}$ is asserted externally during internal cycles, and internal data is driven out on the external data bus. Because internal cycles normally continue to run when the external bus is granted, one SHEN[1:0] encoding halts internal bus activity while there is an external master.

The 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 SCIM2E drives the value of the byte that is written to both halves of the data bus.

4.7 Reset

Reset occurs when an active low logic level on the $\overline{RESET}$ pin is clocked into the SCIM2E. The $\overline{RESET}$ input is synchronized to the system clock. If there is no clock when $\overline{RESET}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{RESET}$ is asserted.

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SCIM2E determines whether a reset is valid, asserts control signals, performs basic system configuration and boot memory selection based on hardware mode-select inputs, then passes control to the CPU32.

4.7.1 SCIM2E Reset Control Logic

The SCIM2E reset control logic differs somewhat from the SCIM. Do not use information in SCIM documentation for the SCIM2E. As with the SCIM, the asserted state of the external reset pin is ‘0’. The released state is ‘1’. The external circuit must make some provision to pull the reset pin high (usually, just a pullup resistor and a capacitor to ground) when the SCIM2’s reset controller releases reset. The specified maximum time in which external circuit must release the reset pin (the input signal must be at or above VIH) has been extended by approximately 180 clock cycles (22.5 µs at eight MHz for slow reference clock mode, 45 µs at four MHz for fast reference clock mode, for more information about clock modes, see **4.3 System Clock**.

For the SCIM, reset must rise to VIH within 10 clocks (1.25 µs at eight MHz). For applications that meet the current requirements of the SCIM’s reset timing, no apparent change in reset timing will occur on the SCIM2E. Reset vector fetch and code execution for the SCIM2E will begin after the 10th clock as it does on the SCIM. Applications that do not pull reset high by the end of the first 10 clocks are effectively given one more chance to have properly released reset by the 190th clock (180 clocks after the end of the initial 10-clock period). If reset has not been released by this time, reset is re-asserted by the SCIM2’s reset controller for 512 clocks, reset configuration is again latched from the data bus, and the sequence is repeated.

4.7.1.1 SCIM2E Reset Control Flow

The reset control flow for SCIM2 is depicted in **Figure 4-17**. This is the same as the original flow with one exception: After the external system is given the 10 clocks to pull reset high, the reset pin is sampled. If it is not high, the external system is given one more opportunity to pull it high. This time period is 180 clocks long. At the end of this second period, if the external system has negated reset, code execution begins. Otherwise, the reset controller loops back into the 512-clock hard reset sequence after waiting for the reset pin to go high. The boxes that have been added to the original reset control flow are shaded for easy identification.

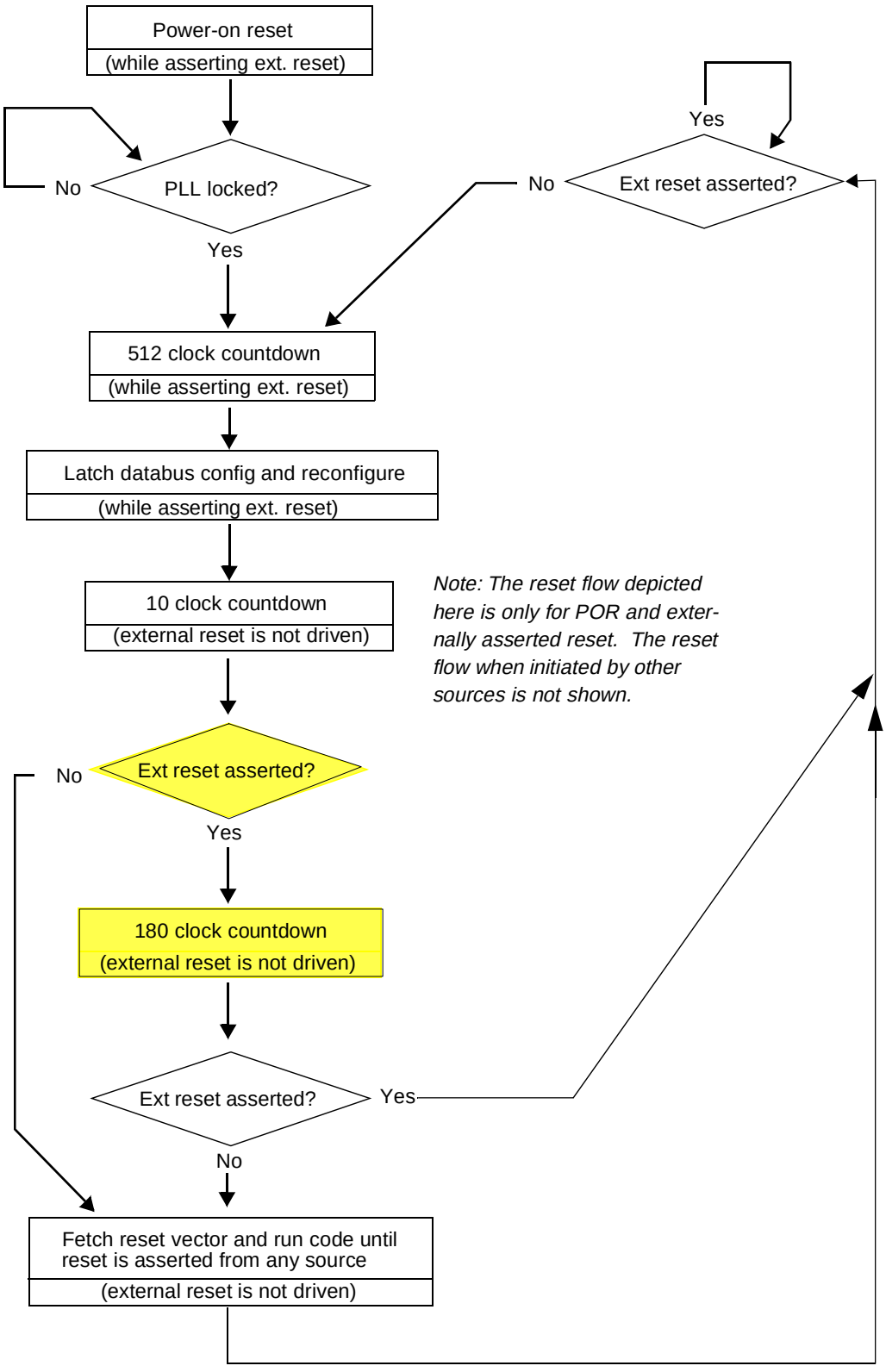


Figure 4-17 SCIM2 Reset Control Flow

4.7.2 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the exception vector table. The exception vector table consists of 256 four-byte vectors and occupies 1024 bytes of address space. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

Reset is the highest-priority CPU32 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine.

NOTE

External circuitry is required to disable external bus configuration logic until $\overline{DS}$ and R/$\overline{W}$ are negated to ensure that bus cycles in progress at the time $\overline{RESET}$ is asserted complete correctly.

4.7.3 Reset Source Summary

SCIM2E reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control logic. All resets are gated by CLKOUT. Asynchronous resets can occur on any clock edge and are assumed to be catastrophic. Synchronous resets are timed to occur at the end of bus cycles. When a synchronous reset is detected, the SCIM2E bus monitor is automatically enabled. If the bus cycle during which a synchronous reset is detected does not terminate normally, the bus monitor will terminate the cycle and allow the reset to proceed. **Table 4-22** is a summary of reset sources.

Table 4-22 Reset Source Summary

Type	Source	Timing
External	Assertion of $\overline{RESET}$ pin	Synchronous
Power on	Rising voltage on V_{DD}	Asynchronous
Software watchdog	Timeout of software watchdog	Asynchronous
Halt	Halt monitor (e.g. double bus fault)	Asynchronous
Loss of clock	Reference failure caught by loss of clock detector	Synchronous
Test	Test submodule	Synchronous

Internal byte and aligned word write cycles are guaranteed valid for synchronous resets. External writes will also complete uncorrupted, provided the data bus is conditioned with a circuit that incorporates RESET, such as that shown in **Figure 4-19**.

4.7.4 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, more than one bit in RSR may be set. The reset status register is updated by the reset control logic when RESET is released.

RSR — Reset Status Register **0xYF FA06**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Reserved								EXT	POW	SW	HLT	0	Re-served	SYS	TST

Table 4-23 RSR Bit Descriptions

Bit(s)	Name	Description
15:8	—	Reserved
7	EXT	External reset. Reset caused by the RESET pin.
6	POW	Power-up reset. Reset caused by the power-up reset circuit.
5	SW	Software watchdog reset. Reset caused by the software watchdog circuit.
4	HLT	Halt monitor reset. Reset caused by the halt monitor.
3:2	—	Reserved
1	SYS	System reset. Reset caused by a RESET instruction.
0	TST	Test submodule reset. Reset caused by the test submodule. Used during factory test reserved operating mode only.

This register can be read at any time; a write has no effect. Bits [15:8] are reserved and always read zero.

4.7.5 Reset Timing

When an external device asserts the RESET pin for at least four clock cycles, the signal will be latched and held internally until completion of the current bus cycle. Any further processing of the reset exception is then delayed until the SCIM2E reset control logic detects that the RESET pin is no longer being externally driven. Two clock cycles will elapse (during which time the pullup resistor on RESET will pull the pin high) while the reset control logic switches the RESET pin from an input to an output. RESET will then be driven low for 512 clock cycles.

If a synchronous internal reset is detected (e.g., from the loss of clock detector or the test submodule), the reset control logic will wait for bus cycle completion and then drive RESET low for 512 clock cycles. An asynchronous internal reset (e.g., from the halt monitor or the software watchdog) will immediately drive RESET low for 512 clock cycles without waiting for the current bus cycle to complete.

After the 512-clock cycle assertion of the RESET pin, the processing flow for both internal and external resets is the same. The SCIM2E reset control logic will release the RESET pin and read configuration information from BERR, BKPT, and DATA[15:0]. Refer to **4.7.8 Operating Configuration Out of Reset** for more information.

Ten clock cycles will elapse to allow the pull-up resistor on RESET to pull the pin high. The reset control logic will then sample the RESET pin. If the pin is high, the reset control logic will release the external bus interface (EBI) and allow the reset vector to be fetched. If RESET is still low, 180 clock cycles will elapse, and the reset control logic will sample the pin again. As above, if RESET is high, processing will resume and the reset vector will be fetched.

If RESET still has not risen to logic one, the reset control logic will begin the external reset sequence as described at the beginning of this section. Further reset exception processing will not proceed until RESET is sampled at logic one after the 10-clock cycle or 180-clock cycle delays described above. **Figure 4-18** depicts the reset sequence for the SCIM2E.

4.7.6 Power-On Reset

Power-on reset (POR) operation involves special circumstances related to the application of system power and, if the PLL is used, clock synthesizer power. V_{DD} ramp time affects pin state during reset. Slow V_{DD} ramp times can leave MCU pins in an indeterminate state longer than is desired or is tolerable in some applications.

When the PLL is used to generate the MCU system clock, oscillator start up time also determines how long MCU pins remain in an indeterminate state. Immediate application of V_{DDSYN}/MODCLK power and careful attention to crystal specifications and oscillator circuit design play an important role in minimizing start up time.

Grounding V_{DDSYN}/MODCLK places the MCU in external clock mode, initially operating at the frequency input on the EXTAL pin. In this case, any events requiring clock cycles during POR will occur as quickly as those clock cycles are input on the EXTAL pin.

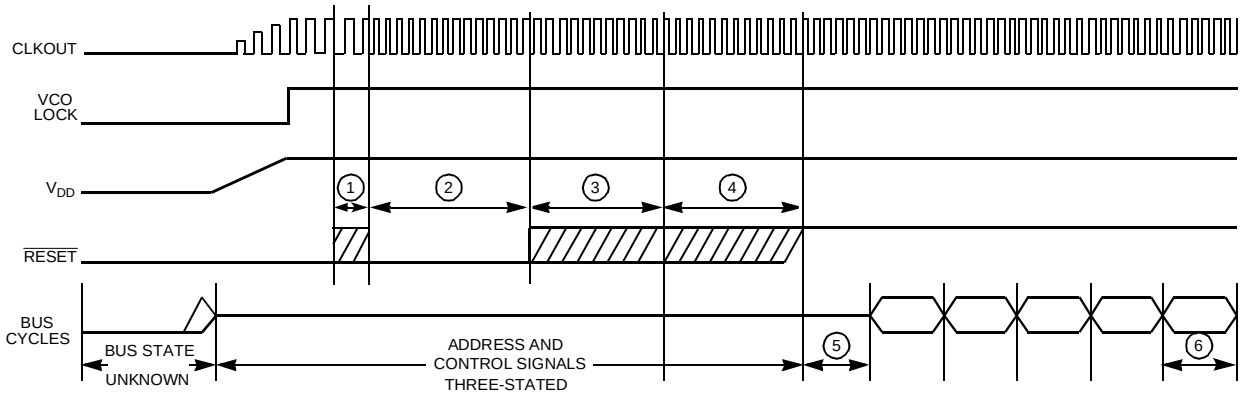
Power-on reset activates a circuit in the SCIM2E that asserts the internal and external RESET lines. As V_{DD} ramps up to the minimum operating voltage, the PLL (if enabled) begins to generate the system clock and the internal RESET line is negated. This initializes SCIM2E pins the values shown in **Table 4-24**.

At this point, POR will proceed no further until the PLL locks at two or 256 times f_{ref} in fast or slow reference modes, respectively. Reset exception processing will then continue as outlined in **4.7.5 Reset Timing.** In external clock mode, the PLL is disabled which permits normal reset exception processing as soon as the internal RESET line is negated.

The SCIM2E propagates RESET and the system clock to all other MCU modules. Once the clock is running and internal RESET is asserted for at least four clock cycles, these modules reset. V_{DD} and PLL ramp up times determine how long these four clock

cycles take. Worst case occurs in slow reference mode and is approximately 15 milliseconds. During this period, MCU pins may be in an indeterminate state. While pull-up resistors may be used on input only pins, active logic will be required to condition input/output or output only pins.

Figure 4-18 depicts the timing of the power-on reset sequence.



NOTES:
1. TWO CLOCK CYCLE DELAY REQUIRED BY RESET CONTROL LOGIC TO SWITCH $\overline{\text{RESET}}$ PIN FROM INPUT TO OUTPUT.
2. RESET CONTROL LOGIC DRIVES THE $\overline{\text{RESET}}$ PIN LOW FOR 512 CLOCK CYCLES TO GUARANTEE THIS LENGTH OF RESET TO THE ENTIRE SYSTEM.
3. TEN CLOCK CYCLE DELAY DURING WHICH THE $\overline{\text{RESET}}$ PIN IS NO LONGER DRIVEN AND IS ALLOWED TO RISE TO A LOGIC ONE. OPERATING CONFIGURATION IS LATCHED FROM DATA[15:0] AND $\overline{\text{BERR}}$ WHEN THIS DELAY BEGINS.
4. ADDITIONAL 180 CLOCK DELAY PROVIDED BY RESET CONTROL LOGIC TO ALLOW THE RESET PIN TO RISE TO A LOGIC ONE IF IT DID NOT DO SO DURING THE PRIOR 10 CLOCK CYCLES.
5. INTERNAL START-UP TIME.
6. FIRST INSTRUCTION FETCHED.

SCIM2E POR TIM

Figure 4-18 Power-On Reset

4.7.7 Pin State During Reset

It is important to keep the distinction between pin function and pin electrical state clear. Although control register values and mode select inputs determine pin function, a pin driver can be active, inactive, or in a high impedance state when reset occurs. During power-on reset, pin state is subject to the constraints discussed in **4.7.6 Power-On Reset**.

NOTE

Pins that are not used should either be configured as outputs (if possible) or as inputs and be pulled to an appropriate inactive state. This decreases unnecessary current consumption caused by digital inputs floating near mid-supply level.

4.7.7.1 Reset States of SCIM2E Pins

Generally, while $\overline{\text{RESET}}$ is asserted, SCIM2E pins either go into a high-impedance state or are driven to their inactive states. Operating mode selection occurs when

$\overline{\text{RESET}}$ is released. Mode select inputs are driven to the appropriate states at this time. Use an active circuit, such as that shown in **Figure 4-19**, for this purpose. **Table 4-24** shows the state of SCIM2E pins during reset.

Table 4-24 SCIM2E Pin States During Reset

Pin(s)	Pin State During $\overline{\text{RESET}}$
ADDR[2:0]	High-Z
ADDR[10:3]/PB[7:0]	High-Z
ADDR[18:11]/PA]7:0]	High-Z
ADDR[22:19]/$\overline{\text{CS}}$[9:6]/PC[6:3]	V_{DD}
ADDR23/$\overline{\text{CS10}}$/ECLK	V_{DD}
$\overline{\text{AS}}$/PE5	High-Z
$\overline{\text{AVEC}}$/PE2	High-Z
$\overline{\text{BERR}}$	Mode Select Input
$\overline{\text{BG}}$/$\overline{\text{CSM}}$	V_{DD}
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$	V_{DD}
$\overline{\text{BR}}$/$\overline{\text{CS0}}$	V_{DD}
CLKOUT	Output
$\overline{\text{CSBOOT}}$	V_{DD}
DATA[7:0]/PH[7:0]	Mode Select Inputs
DATA[15:8]/PG[7:0]	Mode Select Inputs
$\overline{\text{DS}}$/PE4	High-Z
$\overline{\text{DSACK0}}$/PE0	High-Z
$\overline{\text{DSACK1}}$/PE1	High-Z
FASTREF/PF0	Mode Select Input
FC0/$\overline{\text{CS3}}$/PC0	V_{DD}
FC1/PC1	V_{DD}
FC2/$\overline{\text{CS3}}$/PC2	V_{DD}
$\overline{\text{HALT}}$	High-Z
$\overline{\text{IRQ}}$[7:1]/PF[7:1]	High-Z
PE3	High-Z
R/$\overline{\text{W}}$	High-Z
$\overline{\text{RESET}}$	Asserted
SIZ[1:0]/PE[7:6]	High-Z
TSC	Three State Enable Input

4.7.7.2 Reset States of Pins Assigned to Other MCU Modules

As a rule, module pins that can be configured for general purpose I/O go into a high-impedance state during reset. However, during power-on reset, module port pins may be in an indeterminate state for a short period of time. Refer to **4.7.6 Power-On Reset** for more information.

4.7.8 Operating Configuration Out of Reset

When $\overline{\text{RESET}}$ is released, the SCIM2E acquires setup information from several MCU pins. Individually or in groups, these pins control the four basic areas of MCU configuration outlined in **Table 4-25**.

Table 4-25 Pins Associated with Basic Configuration Options

Option	Controlling Pins
Background Debug Mode Disable/Enable	$\overline{\text{BKPT}}$
Flash/ROM Module Disable/Enable	DATA[15:12]
Operating Mode Selection	$\overline{\text{BERR}}$, DATA1
SCIM2E I/O Port Configuration	DATA[11:2], DATA0

Clock mode is not listed in **Table 4-25** because it is not selected when $\overline{\text{RESET}}$ is released. Instead, it is latched immediately from V_{DDSYN}/MODCLK and FASTREF/PF0 upon assertion of $\overline{\text{RESET}}$. Refer to **4.3.1 System Clock Sources** for more information.

4.7.8.1 Operating Mode Selection

The logic states of $\overline{\text{BERR}}$ and DATA1 determine MCU operating mode when $\overline{\text{RESET}}$ is released. Care should be taken to guarantee that $\overline{\text{BERR}}$ is driven to a known state during reset. Unlike DATA1 which has a weak pull-up resistor, no conditioning circuitry is present on the $\overline{\text{BERR}}$ pin. If $\overline{\text{BERR}}$ is allowed to float during reset, improper mode determination may occur. Operating mode selection is summarized in **Table 4-26**.

The configuration of the SCIM2E EBI is dependent on operating mode. ADDR[18:3] serve as general purpose I/O ports A and B when the MCU is running in single-chip mode. DATA[7:0] serve as general purpose I/O port H in the single-chip and 8-bit expanded modes, and DATA[15:8] serve as general purpose I/O port G in single-chip mode.

Table 4-26 Mode Configuration During Reset

Pin(s) Affected	Mode Select Pin	Effect of Mode Pin High During RESET	Effect of Mode Pin Low During RESET	Default for 8-Bit Data Bus Mode 11	Default Single Chip Mode 01, 00
A[18:3] D[15:0]	BERR	Expanded	Single Chip	—	A[18:3] = PortA, B D[15:0] = Port G, H
—	MODCK	VCO= System Clock	EXTAL= System Clock	Decode MODCK	Decode MODCK
—	BKPT	BGND Mode Disabled	BGND Mode Enabled	Decode BKPT	Decode BKPT
CSBOOT	D0	16-bit	8-bit	8-bit	8-bit
D[7:0]	D1	8-Bit Data Bus	16-Bit Data Bus	D[7:0] = Port H	Ignored
BR/CS0 FC0/CS3/PC0 FC1/PC1 FC2/CS5/PC2	D2	CS0 CS3 FC1 CS5	BR FC0 FC1 FC2	CS0 CS3 FC1 CS5	CS0 PC0 PC1 PC2
A19/CS6/PC3 A20/CS7/PC4 A21/CS8/PC5 A22/CS9/PC6 A23/CS10/E	D3-D7 D4-D7 D5-D7 D6–D7 D7	CS6 CS7 CS8 CS9 CS10	A19 A20 A21 A22 A23	CS6 CS7 CS8 CS9 CS10	PC3 PC4 PC5 PC6 CS10
DSACK0/PE0 DSACK1/PE1 AVEC/PE2 RMC/PE3 DS/PE4 AS/PE5 SIZ0/PE6 SIZ1/PE7	D8	DSACK0 DSACK1 AVEC RMC DS AS SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7	— (Decode D8) — — — — — — —	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
FASTREF/PF0 IRQ1/PF1 IRQ2/PF2 IRQ3/PF3 IRQ4/PF4 IRQ5/PF5 IRQ6/PF6 IRQ7/PF7	D9	MODCK IRQ1 IRQ2 IRQ3 IRQ4 IRQ5 IRQ6 IRQ7	PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7	— (Decode D9) — — — — — — —	PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7
BGACK/CSE BG/CSM	D10 D11 D12-15				
	D10	BGACK/BG Emulator mode disabled	CSE/CSM Emulator mode enabled	BGACK/BG Emulator mode disabled	BGACK/BG Emulator mode disabled
	D11	Slave mode disabled	Slave mode enabled	Slave mode disabled	Slave mode disabled
	D12	Not used			
	D13	CMFI and ROM EMUL enable (high = enabled, low = disabled)			
	D14	ROM STOP (high = enabled, low = disabled)			
	D15	CMFI STOP (high = enabled, low = disabled)			

In single-chip mode, the default setting of the address bus disable (ABD) bit places ADDR[2:0] in a high-impedance state to reduce noise emissions. ADDR[2:0] function

as normal address bus pins in expanded operating modes. Refer to **4.2.4 Noise Reduction in Single-Chip Mode** and **APPENDIX A INTERNAL MEMORY MAP** for information on the address bus disable bit (ABD).

The ADDR[23:19] pins have multiple functions as high-order address lines, chip selects, or discrete outputs and are configured differently depending on operating mode selection and data bus conditioning when $\overline{\text{RESET}}$ is released. The following paragraphs contain a summary of pin configuration options for each external bus configuration.

4.7.8.2 Data Bus Mode Selection

DATA[15:0] have weak internal pull-up devices. When pins are held high by the internal pull-ups, the MCU uses a default operating configuration. Specific lines can be held low externally during reset to achieve alternate configurations.

NOTE

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

Use an active device to properly configure data bus lines while $\overline{\text{RESET}}$ is low. Data bus configuration logic must release the bus before the first bus cycle after reset 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 of this type, an isolation resistor on the driven lines may be required. **Figure 4-19** shows a recommended method for conditioning the data bus mode select signals.

The mode configuration drivers are conditioned with R/$\overline{\text{W}}$ and $\overline{\text{DS}}$ to prevent conflicts between external devices and the MCU when $\overline{\text{RESET}}$ is asserted. If $\overline{\text{RESET}}$ is asserted during an external write cycle, R/$\overline{\text{W}}$ conditioning (as shown in **Figure 4-19**) prevents corruption of the data during the write. Similarly, $\overline{\text{DS}}$ conditions the mode configuration drivers so that external reads are not corrupted when $\overline{\text{RESET}}$ is asserted during an external read cycle.

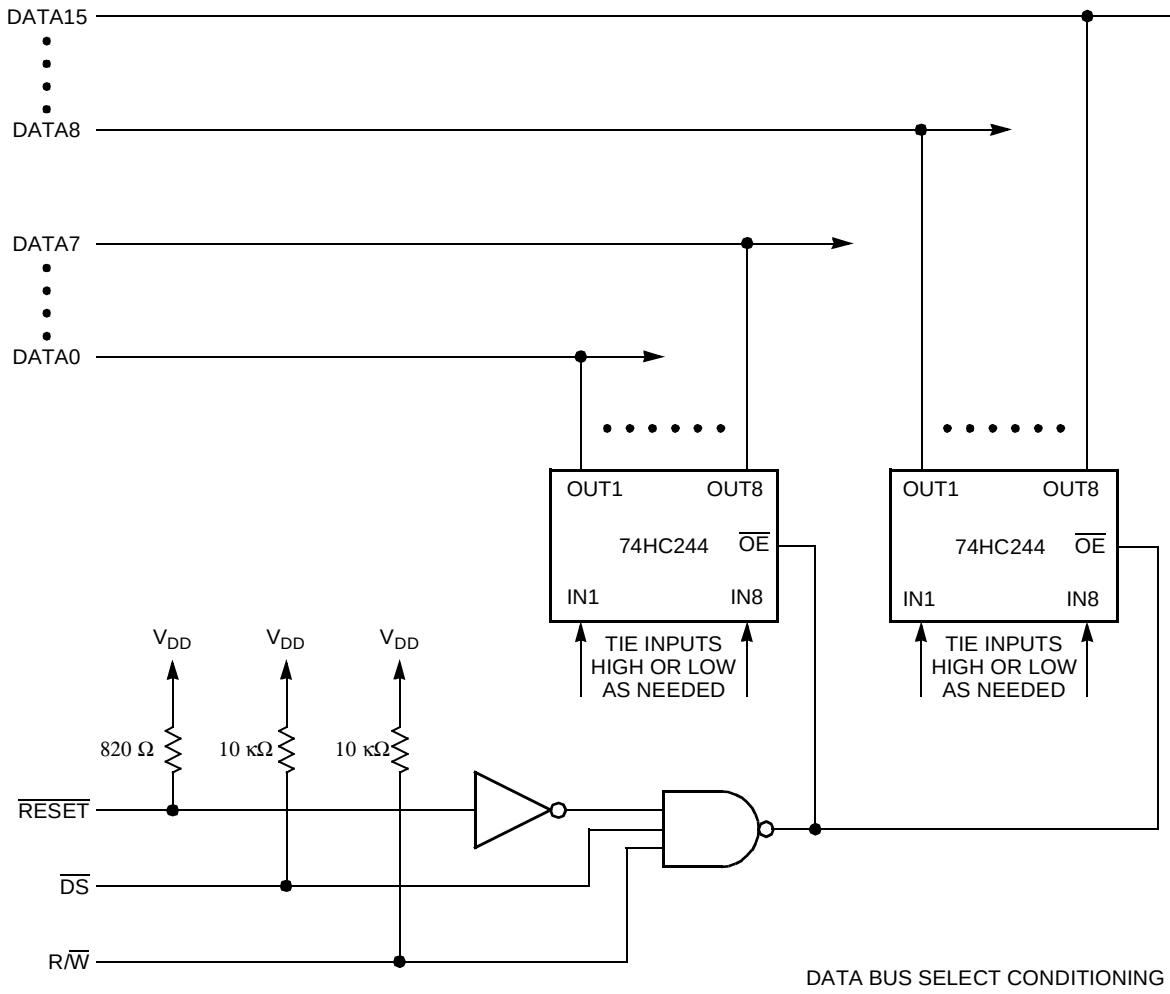


Figure 4-19 Preferred Circuit for Data Bus Mode Select Conditioning

Alternate methods can be used for driving data bus pins low during reset. **Figure 4-20** shows two of these options.

NOTE

These simpler circuits do not offer the protection from potential memory corruption during $\overline{\text{RESET}}$ assertion as does the circuit shown in **Figure 4-19**.

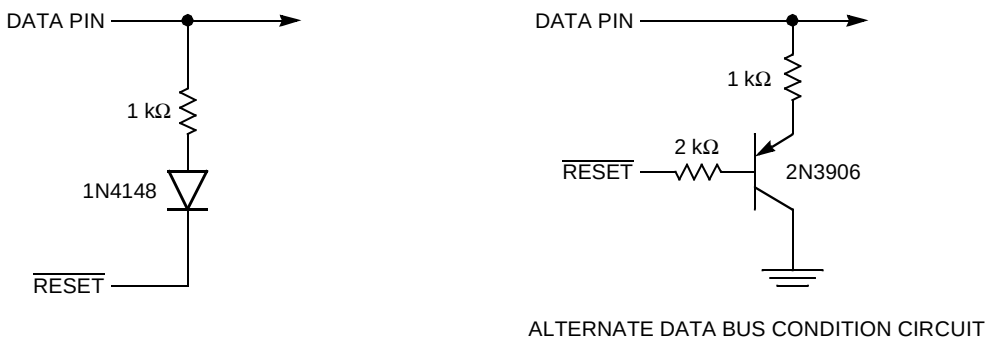


Figure 4-20 Alternate Circuit for Data Bus Mode Select Conditioning

In the simpler of these two circuits, a resistor is connected in series with a diode from the data bus pin to the RESET line. A bipolar transistor can be used for the same purpose, but an additional current limiting resistor must be connected between the base of the transistor and the RESET pin. If a MOSFET is substituted for the bipolar transistor, only the 1 KΩ isolation resistor is required.

4.7.8.3 Single-Chip Mode

When BERR = 0 at the release of RESET, single-chip operation is selected. BERR must return to a logic 1 before the first bus cycle after reset to insure proper device operation. The external bus interface is essentially disabled in single-chip mode, and SCIM2E pins generally serve as discrete inputs and outputs. The behavior of specific pin groups is discussed in the following paragraphs.

NOTE

The paragraphs that follow describe the behavior of SCIM2E pins in single-chip mode only. Sections that follow cover 16-bit and 8-bit expanded modes.

ADDR[2:0] have no discrete I/O function in single-chip mode. These pins are placed in a high-impedance state at power-on but can be enabled by clearing the ABD bit in SCIMMCR.

ADDR[18:11] become port A input/output pins PA[7:0], and ADDR[10:3] become port B input/output pins PB[7:0]. Each port is configurable entirely as inputs or outputs on a per port basis by the DDA and DDB bits in the port A/B data direction register (DDRAB).

Special attention should be paid to chip-select pins in single-chip mode. While each chip-select base address register and option register is active and may be programmed as desired, a match condition will not assert the corresponding pin. For this reason, chip selects should be used expressly to provide autovector termination of interrupt acknowledge cycles generated in response to assertion of the IRQ[7:1] pins.

Because match conditions do not result in chip-select assertion, the 0b10 (8-bit port) and 0b11 (16-bit port) encodings of the pin assignment fields in CSPAR0 and CSPAR1 serve only to drive pins so configured high at all times. Consequently, any chip select may provide autovector termination, even if its pin assignment field in CSPAR0 or CSPAR1 is programmed with the 0b00 (discrete output) or 0b01 (alternate function) encoding.

The first chip select that should be used for autovector termination in single-chip mode is $\overline{\text{CSBOOT}}$; it has no discrete output or alternate function capability. Although typically not needed in single-chip mode, the SCIM2E bus arbitration feature may be used when the $\overline{\text{BR}}$/$\overline{\text{CS0}}$, $\overline{\text{BG}}$/$\overline{\text{CSM}}$, and $\overline{\text{BGACK}}$/$\overline{\text{CSE}}$ pins are configured for their alternate functions. Of these three pins, only $\overline{\text{BR}}$/$\overline{\text{CS0}}$ can provide autovector termination. The $\overline{\text{CSM}}$ and $\overline{\text{CSE}}$ chip selects function in emulation mode only and are not user programmable.

CSPAR0 initially configures the SCIM2E function code pins FC[2:0] as port C discrete outputs PC[2:0]. Each pin may, however, still operate as a function code output, and when configured as such, will be driven during appropriate bus cycles. The chip-select functions of FC0/$\overline{\text{CS3}}$/PC0 and FC2/$\overline{\text{CS5}}$/PC2 may also be used for autovector termination in the fashion described above.

ADDR[22:19]/$\overline{\text{CS[9:6]}}$/PC[6:3] are initially configured as port C discrete outputs PC[6:3] by chip-select pin assignment register 1 (CSPAR1). Each pin may, however, still operate as an address line, and when configured as such, will be driven during appropriate bus cycles.

CSPAR1 initially configures ADDR23/$\overline{\text{CS10}}$/ECLK as a 16-bit chip select (0b11 pin assignment field encoding) to drive the pin to its inactive state. ADDR23/ $\overline{\text{CS10}}$/ECLK has no discrete output function. When 0b00 is programmed into its pin assignment field in CSPAR1, ADDR23/$\overline{\text{CS10}}$/ECLK will drive the M6800 bus E clock signal.

Just as the chip-select, function code, and bus arbitration signals associated with port C can be made active in single-chip mode, so too can the bus control signals associated with port E. While initially configured as discrete I/O by the port E pin assignment register (PEPAR), any port E bus control signal can be made active and will be driven or accept input during appropriate bus cycles.

Port F pins will initially be configured for discrete I/O in single-chip mode but can otherwise serve as interrupt request lines or edge-detect I/O pins with optional interrupt capability. Because no external bus is available in single-chip mode, interrupt requests from port F pins configured as interrupts (as opposed to interrupt requests from the port F edge-detect logic) must have their interrupt acknowledge cycles terminated by autovector.

In single-chip mode, the data bus is disabled at all times. DATA[15:8] become port G input/output pins PG[7:0], and DATA[7:0] become port H input/output pins PH[7:0]. Port G and H pins are configurable as inputs or outputs on a per pin basis.

Like ADDR[2:0] (which can be disabled by setting the ABD bit in SCIMMCR), the R/W line and instruction tracking pins (IPIPE/DSO and IFETCH/DSI) can be disabled by setting RWD and CPUD bits in SCIMMCR, respectively.

4.7.8.4 Fully (16-bit) Expanded Mode

Operation in 16-bit expanded mode is selected when BERR = 1 and DATA1 = 0 at the release of RESET. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/PB[7:0] become part of the address bus. Likewise, DATA[15:8]/PG[7:0] and DATA[7:0]/PH[7:0] become the data bus. The ABD, RWD, and CPUD bits in SCIMMCR are clear, enabling ADDR[2:0], R/W, and the instruction tracking pins (IPIPE/DSO and IFETCH/DSI), respectively. Ports A, B, G, and H are unavailable in 16-bit expanded mode. The initial configuration of all other SCIM2E pins is controlled by DATA[11:2] and DATA0 and is outlined in **Table 4-27** below.

Table 4-27 Fully (16-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
DATA0	CSBOOT	16-bit CSBOOT	8-bit CSBOOT
DATA2	BR/CS0 FC0/CS3/PC0 FC1/PC1 FC2/CS5/PC2	CS0 CS3 FC1 CS5	BR FC0 FC1 FC2
DATA[7:3]	ADDR23/CS10/ECLK ADDR[22:19]/CS[9:6]/PC[6:3]	See **Table 4-28**	
DATA8	DSACK0/PE0 DSACK1/PE1 AVEC/PE2 RMC/PE3 DS/PE4 AS/PE5 SIZ0/PE6 SIZ1/PE7	DSACK0 DSACK1 AVEC RMC DS AS SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 IRQ[7:1]/PF[7:1]	FASTREF[1] IRQ[7:1]	PF0 PF[7:1]
DATA10[2]	BGACK/CSE BG/CSM	BGACK BG	CSE CSM[3]
DATA11[4]	External Bus Interface	Normal Operation	Factory Test

NOTES:

1. The FASTREF function is used only at reset and serves no purpose during normal operation.
2. If DATA1 and DATA10 are low at the rising edge of RESET, the SCIM2E will operate in emulation mode, a special variation of 16-bit expanded mode.
3. For CSM to be active, the SCIM2E must be configured for emulation mode, as described above, and any on-chip masked ROM modules must be disabled by driving their associated data bus pins low at the rising edge of RESET. At present, only masked ROM modules support memory emulation by means of the CSM chip select. CSM will not assert on MCUs with flash EEPROM modules.
4. DATA11 must be high at the rising edge of RESET for normal MCU operation.

DATA[7:3] select in a contiguous fashion whether ADDR[23:19]/$\overline{CS}$[10:6] serve as high-order address lines or chip selects. **Table 4-28** shows the reset correspondence between these pins.

Table 4-28 Reset Pin Function of $\overline{CS}$[10:6]

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

DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. **Table 4-29** shows which modules on the MC68F375 are affected by these pins.

Table 4-29 Reset Configuration for MC68F375 Memory Modules

Select Pin	State of Select Pin at Rising Edge of $\overline{RESET}$	Memory Modules Affected
DATA10, DATA13	0	CMFI/ROM emulation mode
	1	CMFI/ROM normal mode
DATA14[1]	0	8-Kbyte Masked ROM array disabled All four 32-Kbyte FLASH arrays disabled
	1	8-Kbyte Masked ROM array enabled All four 32-Kbyte FLASH arrays enabled
DATA15[2]	0	All four 32-Kbyte CMFI FLASH blocks disabled
	1	All four 32-Kbyte CMFI FLASH blocks enabled

NOTES:
1. The ROM can be disabled if the STOP shadow bit is programmed to one or if DATA14 is low at the rising edge of $\overline{RESET}$.
2. The CMFI array is disabled if its STOP shadow bit is programmed to one or if DATA15 is low at the rising edge of $\overline{RESET}$.

Operation in 8-bit expanded mode is selected when $\overline{BERR}$ = 1 and DATA1 = 1 at the release of $\overline{RESET}$. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/ PB[7:0] become part of the address bus, and only DATA[15:8]/PG[7:0] are used for the data bus. The ABD, RWD, and CPUD bits in SCIMMCR are clear, enabling ADDR[2:0], R/$\overline{W}$, and the instruction tracking pins ($\overline{IPIPE}$/DSO and $\overline{IFETCH}$/DSI), respectively. Ports A, B, and G are unavailable in 8-bit expanded mode, and DATA[7:0]/PH[7:0] serve as port H discrete I/O pins only. All remaining SCIM2E pins are configured as shown in **Table 4-30.**

Table 4-30 Partially (8-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s) or Module(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
NA[1]	$\overline{\text{CSBOOT}}$	8-bit $\overline{\text{CSBOOT}}$	
NA[1]	$\overline{\text{BR}}$/$\overline{\text{CS0}}$ FC0/$\overline{\text{CS3}}$/PC0 FC1/PC1 FC2/$\overline{\text{CS5}}$/PC2	$\overline{\text{CS0}}$ $\overline{\text{CS3}}$ FC1 $\overline{\text{CS5}}$	
NA[1]	ADDR23/$\overline{\text{CS10}}$/ECLK ADDR[22:19]/$\overline{\text{CS}}$[9:6]/PC[6:3]	$\overline{\text{CS}}$[10:6]	
DATA8	$\overline{\text{DSACK0}}$/PE0 $\overline{\text{DSACK1}}$/PE1 $\overline{\text{AVEC}}$/PE2 $\overline{\text{RMC}}$/PE3 $\overline{\text{DS}}$/PE4 $\overline{\text{AS}}$/PE5 SIZ0/PE6 SIZ1/PE7	$\overline{\text{DSACK0}}$ $\overline{\text{DSACK1}}$ $\overline{\text{AVEC}}$ $\overline{\text{RMC}}$ $\overline{\text{DS}}$ $\overline{\text{AS}}$ SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 $\overline{\text{IRQ}}$[7:1]/PF[7:1]	FASTREF[2] $\overline{\text{IRQ}}$[7:1]	PF0 PF[7:1]
DATA10	$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$[3] $\overline{\text{BG}}$/$\overline{\text{CSM}}$[3]	$\overline{\text{BGACK}}$ $\overline{\text{BG}}$	

NOTES:
1. Because DATA[7:0] are unavailable in 8-bit expanded mode, these pins default to the reset configurations noted.
2. The FASTREF function is used only at reset and serves no purpose during normal operation.
3. The $\overline{\text{CSE}}$ and $\overline{\text{CSM}}$ emulation chip selects do not function in 8-bit expanded mode.

Just as in 16-bit expanded mode, DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. Refer to **Table 4-29** above for which modules on the MC68F375 are affected by these pins.

4.7.8.5 Breakpoint Mode Selection

Background debug mode (BDM) is enabled when the breakpoint ($\overline{\text{BKPT}}$) pin is sampled at logic zero at the release of $\overline{\text{RESET}}$. Subsequent assertion of the $\overline{\text{BKPT}}$ pin or the internal breakpoint signal (for instance, execution of the CPU32 BGND instruction) will place the CPU32 in BDM.

If $\overline{\text{BKPT}}$ is sampled at logic one at the rising edge of $\overline{\text{RESET}}$, BDM is disabled. Assertion of the $\overline{\text{BKPT}}$ pin or execution of the BKPT instruction will result in normal breakpoint exception processing. Execution of the BGND instruction will cause an illegal instruction exception to be taken.

BDM remains enabled until the next system reset. $\overline{\text{BKPT}}$ is relatched and synchronized on each rising transition of $\overline{\text{RESET}}$ and must be held low for at least two clock cycles prior to $\overline{\text{RESET}}$ negation for BDM to be enabled. $\overline{\text{BKPT}}$ assertion logic must be designed with special care. If $\overline{\text{BKPT}}$ assertion extends into the first bus cycle following the release of $\overline{\text{RESET}}$, the bus cycle could inadvertently be tagged with a breakpoint.

Refer to **3.10.2 Background Debug Mode** and the ***CPU32 Reference Manual* (CPU32RM/AD)** for more information on background debug mode. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** and **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information concerning $\overline{\text{BKPT}}$ signal timing.

4.7.8.6 Emulation Mode Selection

The SCIM2E contains logic that can be used to replace on-chip ports externally. The SCIM2E also contains special support logic that allows external emulation of internal ROM. These emulation support features enable the development of a single-chip application in expanded mode.

Emulation 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 emulation mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. The port C data, port F data and data direction registers, and port F pin assignment register are accessible normally in emulation mode.

The port emulation chip select ($\overline{\text{CSE}}$) is asserted whenever any of the externally mapped registers are addressed. The signal is asserted on the falling edge of $\overline{\text{AS}}$. The SCIM2E does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. Accesses to externally mapped registers require three clock cycles.

CMFI and ROM emulation is enabled by holding $\overline{\text{BERR}}$ high and by holding low DATA1, DATA10, and DATA13 when $\overline{\text{RESET}}$ is released. While CMFI and ROM emulation mode is enabled, the emulation memory chip-select signal ($\overline{\text{CSM}}$) is asserted whenever an access to an address assigned to the masked ROM module or CMFI module is made.

CMFI and ROM modules do not acknowledge IMB accesses while in emulation mode. This causes the SCIM2E to run an external bus cycle for each access. The bus cycle is terminated by the module, however, ensuring consistent timing.

4.7.9 Use of the Three-State Control Pin

Asserting the three-state control (TSC) input to a logic 0 causes the MCU to place all output drivers in a disabled, high-impedance state. TSC must remain asserted for approximately ten clock cycles in order for drivers to change state.

When the SCIM2E clock synthesizer is used, PLL 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, pins go high-impedance as soon after TSC assertion as approximately 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.

4.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SCIM2E, the CPU32, 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. Refer to **4.9 Chip Selects** for more information.

4.8.1 Interrupt Exception Processing

The CPU32 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in a vector table located from 0x00000 to 0x001FF. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

4.8.2 Interrupt Priority and Recognition

The CPU32 provides seven levels of interrupt priority (1–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the condition code register (CCR).

The IP field consists of CCR bits [7:5]. Binary values 0b000 to 0b111 provide eight priority masks. Each mask prevents an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized. When the IP field contains 0b000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

There are seven interrupt request signals ($\overline{\text{IRQ[7:1]}}$) with corresponding external pins that can be asserted by microcontroller modules or external devices. Simultaneous requests of different priorities can be made. Internal assertion of an interrupt request line does not affect the state of the corresponding MCU pin.

External interrupt requests are routed to the CPU32 via the EBI and SCIM2E interrupt control logic. All requests for interrupt service are treated as if they come from internal modules. The CPU32 treats external interrupt requests as if they come from the SCIM2E.

The $\overline{\text{IRQ[6:1]}}$ pins are active-low level-sensitive inputs. The $\overline{\text{IRQ7}}$ pin is an active-low transition-sensitive input; it requires both an edge and a voltage level to be valid.

$\overline{IRQ}$[6:1] are maskable. $\overline{IRQ7}$ is non-maskable. The $\overline{IRQ7}$ input is transition sensitive to prevent redundant servicing and stack overflow. A non-maskable interrupt is generated each time $\overline{IRQ7}$ is asserted and each time the CCR is written while $\overline{IRQ7}$ is asserted. A write to the CCR re-arms the $\overline{IRQ7}$ detection circuitry; consequently, any write to the CCR while $\overline{IRQ7}$ is asserted, even one that sets the IP field to 0b111, will generate a new $\overline{IRQ7}$ interrupt.

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 cycles. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when processing of higher priority exceptions is complete.

The CPU32 does not latch the priority of pending interrupt requests. If an interrupt source of higher priority makes a request while a lower priority request is pending, the higher priority request will be serviced. If an interrupt request with a priority less than or equal to the current IP mask value is made, the CPU32 will not recognize the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU32 will recognize the higher priority request.

4.8.3 Interrupt Acknowledge and Arbitration

When the CPU32 detects one or more interrupt requests of a priority higher than the IP mask value, a read cycle from address 0b11111111111111111111: [IP]:1 in CPU space is executed. Refer to **4.5.1.7 Function Codes** and **4.6.4 CPU Space Cycles** for more information.

The CPU space read cycle performs two functions. It places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value is decoded by modules or external devices that have requested interrupt service to determine whether the current interrupt acknowledge (IACK) cycle pertains to them. It is also latched into the IP field to mask lower priority interrupts during exception processing.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the IACK cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can request interrupt service, including the SCIM2E, has an IARB field in its module configuration register. To implement an arbitration scheme, each module that can request interrupt service must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities can range from 0b0001 (lowest) to 0b1111 (highest). If the CPU32 recognizes an interrupt request from a module that has an IARB field value of 0b0000, a spurious interrupt exception will be processed.

Because the EBI manages external interrupt requests, the SCIM2E IARB field value is used for arbitration between internal and external interrupt requests of the same priority. The reset value of IARB for the SCIM2E is 0b1111, and the reset value of IARB for all other modules is 0b0000. As noted above, initialization software must assign different values to each IARB field to implement an arbitration scheme.

NOTE

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same non-zero value, the CPU32 will interpret multiple vector numbers simultaneously with unpredictable consequences.

Although arbitration is intended to deal with simultaneous interrupt requests of the same priority level, it always take place, even when a single source is requesting service. This is important for two reasons: the EBI does not transfer the IACK cycle to the external bus unless the SCIM2E wins contention, and failure to contend causes the IACK cycle to be terminated early by bus error.

When arbitration is complete, the winning module must place a vector number on the data bus and terminate the IACK cycle with $\overline{\text{DSACK}}$. In the case of external interrupt requests, the IACK cycle is transferred to the external bus. The device requesting interrupt service must decode the mask value then respond with a vector number and generate data and size acknowledge ($\overline{\text{DSACK}}$) termination signals, or it must assert $\overline{\text{AVEC}}$ to request an autovector. If the device does not respond in time, the SCIM2E bus monitor, if enabled, will assert the internal $\overline{\text{BERR}}$ signal and a spurious interrupt exception will be taken.

Chip-select logic can also be used to generate internal $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to external interrupt requests. Chip-select address match logic functions only after the SCIM2E has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2E will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$. Refer to **4.9.4.1 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination** for more information.

As stated above, all interrupt requests from internal modules have their associated IACK cycles terminated by $\overline{\text{DSACK}}$. For this reason, user vectors (instead of autovectors) must always be used for interrupts generated by internal modules.

For periodic timer interrupts, the PIRQL[2:0] field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQL[2:0] value of 0b000 disables PIT interrupts. By hardware convention, PIT interrupts are serviced before external interrupt service requests or port F edge-detect interrupts requests of the same priority. External interrupt requests are serviced before requests from the port F edge-detect logic, as well. Refer to **4.4.7 Periodic Interrupt Timer** and **4.10.4 Port F** for more information.

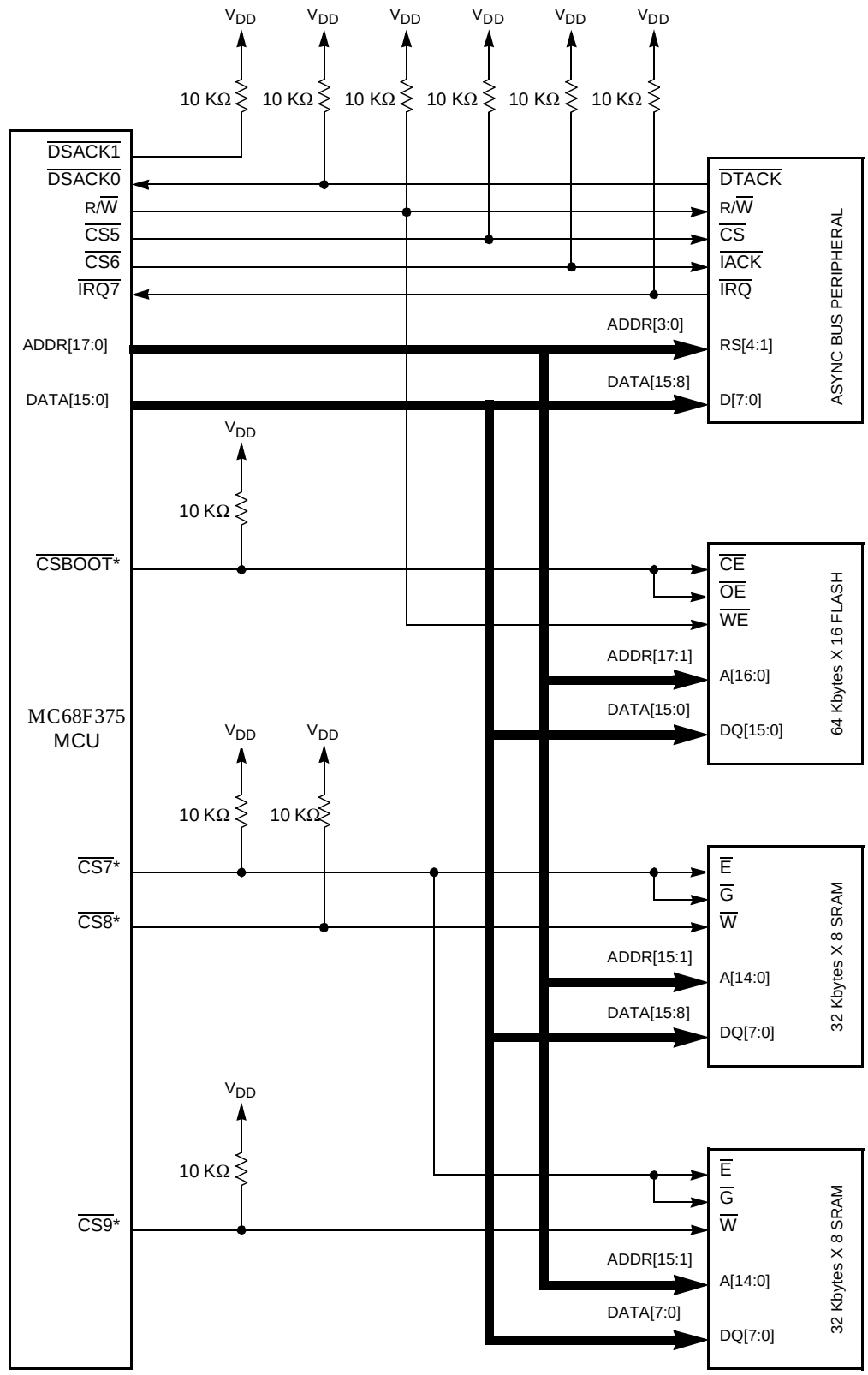
4.8.4 Interrupt Processing Summary

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

1. The CPU32 finishes higher priority exception processing or reaches an instruction boundary.
2. Processor state is stacked.
3. The interrupt acknowledge cycle begins:
   a. FC[2:0] are driven to 0b111 (CPU space) encoding.
   b. The address bus is driven as follows. ADDR[23:20] = 0b1111 ADDR[19:16] = 0b1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = 0b111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = 0b1.
   c. Request priority is latched into the CCR IP field from the address bus.
4. 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.
5. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
   a. When there is no contention (responding modules have IARB = 0b0000), the internal bus monitor, if enabled, asserts $\overline{\text{BERR}}$, and the CPU32 generates the spurious interrupt vector number.
   b. The interrupt source that wins arbitration supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
   c. The $\overline{\text{AVEC}}$ signal is asserted either by the external device requesting interrupt service ($\overline{\text{AVEC}}$ can be tied low if all external interrupts are to use autovectors) or by an appropriately programmed SCIM2E chip select, and the CPU32 generates an autovector number corresponding to the interrupt priority.
   d. The bus monitor or external device asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.
6. The vector number is converted to a vector address.
7. The content of the vector address is loaded into the PC and the processor transfers control to the exception handler routine.

4.9 Chip Selects

Typical microcontrollers require additional hardware to provide chip-select signals for external devices. The SCIM2E includes nine programmable chip-select circuits that can provide from 2- to 16-clock cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. **Figure 4-21** is a diagram of a basic system that uses chip selects.



1. THESE CHIP-SELECT LINES ARE CONFIGURED FOR 16-BIT PORT OPERATION IN THIS EXAMPLE.

F396 SCIM2E B

Figure 4-21 Basic MCU System

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 also generate DSACK and AVEC signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Chip-select signals are active low. If a chip-select function is given the same address as a microcontroller module or an internal memory array, an access to that address goes to the module or array, and the chip-select signal is not asserted. The external address and data buses do not reflect the internal access.

All chip-select signals except CSBOOT are disabled after the release of RESET, and cannot be asserted until the R/W[1:0] and BYTE[1:0] fields in the corresponding option register are programmed to non-zero values. CSBOOT is automatically enabled out of reset in 8-bit and 16-bit expanded modes. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of RESET. Refer to **4.7.8.2 Data Bus Mode Selection** for more information. **Figure 4-22** is a functional diagram of a single chip-select circuit.

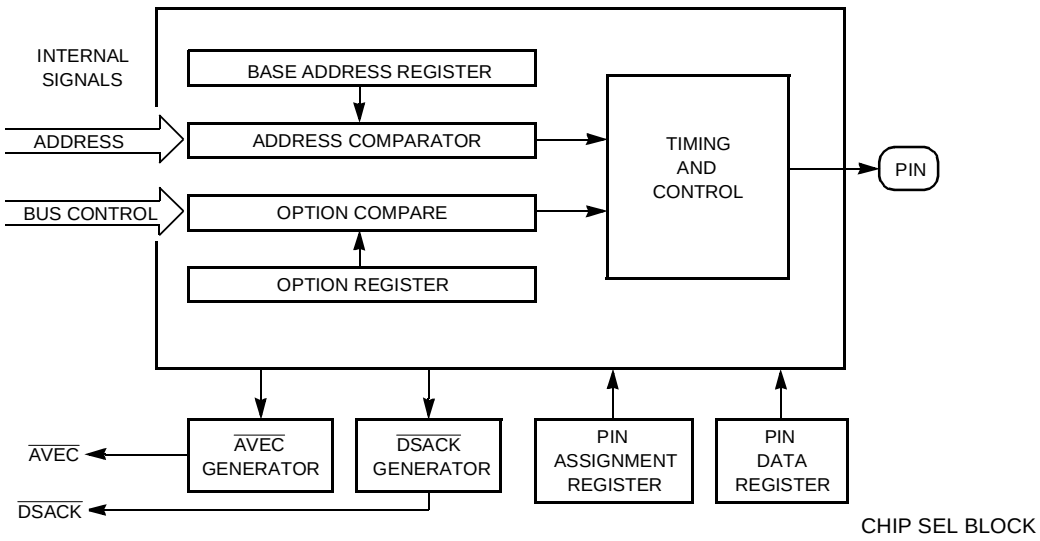


Figure 4-22 Chip-Select Circuit Block Diagram

4.9.1 Chip-Select Pin Assignment Register

The pin assignment registers contain twelve 2-bit fields that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 4-31** and **Table 4-32**.

CSPAR0 — Chip-Select Pin Assignment Register 0 0xYF FA44

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	CS5PA[1:0]		CS4PA[1:0]		CS3PA[1:0]		CSEPA[1:0]		CSMPA[1:0]		CS0PA[1:0]		CSBTPA[1:0]	
RESET:															
0	0	DATA2[1]	1	DATA2[1]	1	DATA2[1]	1	DATA10[1]	1	DATA10[1]	1	DATA2[1]	1	1	DATA0[1]

NOTES:
1. The default state of this bit is taken from the listed bit of the data bus during reset.

This register contains seven 2-bit fields that determine the function of corresponding chip-select pins. Bits [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. The alternate functions can be enabled by data bus mode selection during reset. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table 4-31 shows CSPAR0 pin assignments.

Table 4-31 CSPAR0 Pin Assignments

CSPAR0 Field[1]	Chip-Select Signal	Alternate Signal	Discrete Output
CS5PA[1:0]	CS5	FC2	PC2
CS4PA[1:0]	—	FC1	PC1
CS3PA[1:0]	CS3	FC0	PC0
CSEPA[1:0]	CSE	$\overline{\text{BGACK}}$	—
CSMPA[1:0]	CSM	$\overline{\text{BG}}$	—
CS0PA[1:0]	CS0	$\overline{\text{BR}}$	—
CSBTPA[1:0]	$\overline{\text{CSBOOT}}$	—	—

NOTES:
1. See **Table 4-34** for pin assignment field encoding.

CSPAR1 — Chip-Select Pin Assignment Register 1 0xYF FA46

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	CS10PA[1:0]		CS9PA[1:0]		CS8PA[1:0]		CS7PA[1:0]		CS6PA[1:0]	
RESET:															
0	0	0	0	0	0	DATA 7[1]	1	DATA 6[1]	1	DATA 5[1]	1	DATA 4[1]	1	DATA 3[1]	1

NOTES:
1. Refer to **Table 4-28** for CSPAR1 reset state information.

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. Bits [15:10] are reserved. These bits always read zero; writes have no effect. **Table 4-32** shows CSPAR1 pin assignments, including alternate functions that can be enabled by data bus mode selection during reset.

Table 4-32 CSPAR1 Pin Assignments

CSPAR1 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS10PA[1:0]	$\overline{CS10}$	ADDR23	ECLK
CS9PA[1:0]	$\overline{CS9}$	ADDR22	PC6
CS8PA[1:0]	$\overline{CS8}$	ADDR21	PC5
CS7PA[1:0]	$\overline{CS7}$	ADDR20	PC4
CS6PA[1:0]	$\overline{CS6}$	ADDR19	PC3

The reset state of DATA[7:3] determines whether pins controlled by CSPAR1 are initially configured as high-order address lines or chip-selects. **Table 4-28** shows the correspondence between DATA[7:3] and the reset configuration of $\overline{CS}$[10:6]/ ADDR[23:19]. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table 4-33 Reset Pin Function of $\overline{CS}$[10:6]

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

Table 4-34 shows pin assignment field encoding. Pins that have no discrete output function must not be programmed with the 0b00 encoding as this will configure the pin for its alternate function. For instance, programming CS0PA[1:0] to 0b00 will configure $\overline{CS0}$/$\overline{BR}$ for the bus request ($\overline{BR}$) function.

Table 4-34 Pin Assignment Field Encoding

CSxPA[1:0]	Description
00	Discrete output
01	Alternate function
10	Chip-select (8-bit port)
11	Chip-select (16-bit port)

Port size determines the way in which bus transfers to an external address are allocated. A port size of eight bits or sixteen bits can be selected when a pin is assigned as a chip select. Port size and transfer size affect how the chip-select signal is asserted. Refer to **4.9.3 Chip-Select Option Registers** for more information.

From the release of reset, chip-select pin functions are determined by logic levels on certain data bus pins. The data bus pins have weak internal pull-up devices but can be held low by external logic. This allows a pin's 16-bit chip-select function (data bus pin(s) held high) or its alternate function (data bus pin(s) held low) to be selected at the release of RESET. Refer to **4.7.8.2 Data Bus Mode Selection** for more information.

The CSBOOT signal is enabled out of reset. The state of DATA0 during reset determines what port width CSBOOT uses. If DATA0 is held high, 16-bit port size is selected. If DATA0 is held low, 8-bit port size is selected. In 8-bit expanded mode, the state of DATA0 is ignored, and CSBOOT is configured for 8-bit operation.

A pin programmed as a discrete output will drive the value specified in the port C data register. No discrete output function is available for the CSBOOT, CS0/BR, CSM/BG,and CSE/BGACK pins. ADDR23 provides the ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate DSACK or AVEC (to terminate IACK cycles generated in response to external interrupt requests) internally on an address and control signal match.

4.9.1.1 Port C Data Register

The port C data register (PORTC) latches data for port C pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. Port C bit 7 is not used. Writing to this bit has no effect, and it always reads zero.

PORTC — Port C Data Register **0xYF FA41**

7	6	5	4	3	2	1	LSB 0
0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:							
0	1	1	1	1	1	1	1

PORTC latches data for chip-select pins configured as discrete outputs.

4.9.2 Chip-Select Base Address Registers

Each chip select has an associated base address register, CSBAR[0], [3] and [5:10]. 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 size is determined by the value contained in BLKSZ[2:0]. Multiple chip selects assigned to the same block of addresses must have the same number of wait states.

BLKSZ[2:0] 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 4-35** shows BLKSZ[2:0] encoding.

Table 4-35 Block Size Encoding

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

The chip-select address compare logic uses only the most significant bits to match an address within a block. For this reason, the value of the base address must be an integer multiple of the block size.

After reset, the CPU32 fetches initialization values from addresses 0x00000 to 0x00007 in program space. To support bootstrap operation from reset, the chip-select boot base address register (CSBARBT) defaults to 0x0007 which specifies a base address of 0x000000 and a block size of 512 Kbytes. This allows a memory device containing the reset vector and startup routine to automatically be selected by CSBOOT. Refer to **4.9.4.2 Chip-Select Reset Operation** for more information.

Bit and field definitions for CSBARBT and CSBAR[0], [3] and [5:10] are the same, but reset block sizes differ. For the bit definitions of CSBARBT and CSBAR[0], [3] and [5:10] registers, see **Table 4-36**.

CSBARBT — Chip-Select Base Address Register Boot **0xYF FA48**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR0 — Chip-Select Base Address Registers **0xYF FA4C**
CSBAR3 **0xYF FA58**
CSBAR5 **0xYF FA60**
CSBAR6 **0xYF FA64**
CSBAR7 **0xYF FA68**
CSBAR8 **0xYF FA6C**
CSBAR9 **0xYF FA70**
CSBAR10 **0xYF FA74**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-36 CSBARBT/CSBAR Bit Descriptions

Bit(s)	Name	Description
15:3	ADDR[23:11]	Chip-select base address. This field sets the starting address of a particular chip select's address space. 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 an integer multiple of the block size.
2:0	BLKSZ[2:0]	Block size field. This field determines the size of the block that is enabled by the chip select. **Table 4-35** shows bit encoding for the base address registers block size field.

4.9.3 Chip-Select Option Registers

Fields in the chip-select option registers determine the timing of and conditions for assertion of chip-select signals. For a chip select to assert and to provide $\overline{\text{DSACK}}$ or autovector termination, other constraints set by fields in its option register and base address register must also be satisfied. The following paragraphs summarize option register functions.

CSORBT — Chip-Select Option Register Boot **0xYF FA4A**

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

CSOR0 — Chip-Select Option Registers **0xYF FA4E**
CSOR3 **0xYF FA5A**
CSOR5 **0xYF FA62**
CSOR6 **0xYF FA66**
CSOR7 **0xYF FA6A**
CSOR8 **0xYF FA6E**
CSOR9 **0xYF FA72**
CSOR10 **0xYF FA76**

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

CSORBT and CSOR[0], [3] and [5:10] contain parameters that support operations from external memory devices. Bit and field definitions for CSORBT and CSOR[0], [3] and [5:10] are the same.

Table 4-37 CSOR Bit Descriptions

Bit(s)	Name	Description
15	MODE	Asynchronous/Synchronous Mode. In asynchronous mode, chip-select assertion is synchronized with $\overline{AS}$ and $\overline{DS}$. 0 = Asynchronous mode is selected. 1 = Synchronous mode is selected, and used with ECLK peripherals.
14:13	BYTE	Upper/lower byte option. This field is used only when the chip-select 16-bit port option is selected in the pin assignment register. This allows the usage of two external 8-bit memory devices to be concatenated to form a 16-bit memory. 00 = Disable 01 = Lower byte 10 = Upper byte 11 = Both bytes
12:11	R/$\overline{W}$	Read/write. This field causes a chip select to be asserted only for a read, only for a write, or for both reads and writes. 00 = Disable 01 = Read only 10 = Write only 11 = Read/Write
10	STRB	Address strobe/data strobe. This bit controls the timing for assertion of a chip select in asynchronous mode only. Selecting address strobe causes the chip select to be asserted synchronized with address strobe. Selecting data strobe causes the chip select to be asserted synchronized with data strobe. Data strobe timing is used to create a write strobe when needed. 0 = Address strobe 1 = Data strobe
9:6	$\overline{DSACK}$	Data strobe acknowledge. This field specifies the source of $\overline{DSACK}$ in asynchronous mode as internally generated or externally supplied. It also allows adjust bus timing adjustment with internal $\overline{DSACK}$ generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. **Table 4-38** shows the $\overline{DSACK}$[3:0] field encoding. The fast termination encoding (0b1110) effectively corresponds to –1 wait states.

Table 4-37 CSOR Bit Descriptions (Continued)

Bit(s)	Name	Description
5:4	SPACE	Address space select. Use this option field to select an address space for the chip-select logic. The CPU32 normally operates in supervisor or user space, but interrupt acknowledge cycles must take place in CPU space 00 = CPU Space 01 = User Space 10 = Supervisor Space 11 = Supervisor/User Space
3:1	IPL	Interrupt priority level. When SPACE[1:0] is set for CPU space (0b00), chip-select logic can be used as an interrupt acknowledge strobe for an external device. During an interrupt acknowledge cycle, the interrupt priority level is driven on address lines ADDR[3:1] is then compared to the value in IPL[2:0]. If the values match, an interrupt acknowledge strobe will be generated on the particular chip-select pin, provided other option register conditions are met. **Table 4-39** shows IPL[2:0] field encoding.
0	$\overline{\text{AVEC}}$	Autovector enable. This field selects one of two methods of acquiring an interrupt vector during an interrupt acknowledge cycle. This field is not applicable when SPACE[1:0] = 0b00. 0 = External interrupt vector enabled 1 = Autovector enabled

Table 4-38 $\overline{\text{DSACK}}$ Field Encoding

$\overline{\text{DSACK}}$[3:0]	Clock Cycles Required Per Access	Wait States Inserted Per Access
0000	3	0
0001	4	1
0010	5	2
0011	6	3
0100	7	4
0101	8	5
0110	9	6
0111	10	7
1000	11	8
1001	12	9
1010	13	10
1011	14	11
1100	15	12
1101	16	13
1110	2	–1 (Fast termination)
1111	—	External $\overline{\text{DSACK}}$

Table 4-39 Interrupt Priority Level Field Encoding

IPL[2:0]	Interrupt Priority Level
000	Any Level[1]
001	1
010	2
011	3
100	4
101	5
110	6
111	7

NOTES:
1. Any level means that chip select is asserted regardless of the level of the interrupt acknowledge cycle.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE[1:0] = 0b00) and the $\overline{\text{AVEC}}$ field is set to one, the chip select automatically generates $\overline{\text{AVEC}}$ and completes the interrupt acknowledge cycle. If the $\overline{\text{AVEC}}$ bit = 0, then the vector must be supplied by the requesting external device to complete the IACK read cycle.

The BYTE field controls the data placement conditions under which a particular chip select asserts. This is a different function from that of the chip-select pin assignment registers which determine if transfers controlled by a particular chip select are fundamentally eight or sixteen bits in length. Instead, BYTE[1:0] specifies whether the chip select will assert for data placed on the lower half, upper half, or both halves of the data bus.

When a chip select is configured for 8-bit port operation, only DATA[15:8] are used. Consequently, any BYTE field value other than 0b00 will permit signal assertion when all other match conditions are met.

When a chip select is configured for 16-bit port operation, BYTE[1:0] determines which combinations of ADDR0 and SIZ0 will result in chip-select assertion. A chip select configured for both bytes (0b11) will assert (assuming all other conditions are met) regardless of the states of ADDR0 and SIZ0. A chip select configured for upper byte (0b10) will assert only when ADDR0 = 0 (even addresses). A chip select configured for lower byte (0b01) must assert on all accesses to odd addresses (ADDR0 = 1) and on word accesses to even addresses (ADDR0 = 0 and SIZ0 = 1). When the boolean expression $\overline{\text{ADDR0}}$ • SIZ0 is false, lower byte chip-select assertion will occur. In all cases, the routing of information onto the data bus by the EBI data multiplexer is controlled by ADDR0 and SIZ0.

When SPACE[1:0] = 0b00 (CPU space), IPL[2:0] specifies the interrupt priority that must be matched when chip-select logic is used to terminate IACK cycles generated in response to external requests for interrupt service. When SPACE[1:0] is set to 0b00 (CPU space), ADDR[3:1] is compared to the IPL field at the beginning of an IACK cycle. If these values are the same (and other option register constraints are satisfied),

the specified chip select will be asserted. This field only affects the response of chip-select logic to IACK cycles and does not affect interrupt recognition by the CPU32. Setting IPL[2:0] to 0b000 when SPACE[1:0] = 0b00 will cause chip-select assertion regardless of the IACK cycle priority, provided other option register conditions are met. When SPACE[1:0] = 0b01, 0b10, or 0b11, the IPL field specifies whether chip-select assertion should occur during accesses to data space, program space, or both.

4.9.4 Chip-Select Operation

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

- Function code signals FC[2:0] to the SPACE field, and to the IP field if the SPACE field is not programmed for CPU space.
- Appropriate address bus bits to base address field.
- The R/$\overline{W}$ signal to the R/$\overline{W}$ field.
- ADDR0 and/or SIZ to the BYTE field (only chip selects configured for 16-bit operation).
- Priority of the interrupt being acknowledged (ADDR[3:1]) to the IPL field when the access is an interrupt acknowledge cycle and the SPACE field is programmed for CPU space.

When a match occurs, the chip-select signal is asserted. Assertion occurs at the same time as $\overline{AS}$ or $\overline{DS}$ assertion in asynchronous mode. Assertion is synchronized with ECLK in synchronous mode. In asynchronous mode, the $\overline{DSACK}$ field specifies internal or external $\overline{DSACK}$ assertion and the number of wait states inserted if internal $\overline{DSACK}$ assertion is selected.

The number of wait states needed by an external device is determined by its access time. Normally, wait states are inserted into the bus cycle during state S3 until a peripheral asserts $\overline{DSACK}$. If a peripheral does not generate $\overline{DSACK}$, internal $\overline{DSACK}$ generation must be selected and a predetermined number of wait states can be programmed into the chip-select option register.

4.9.4.1 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination

Ordinary bus cycles are issued in supervisor or user space. Interrupt acknowledge bus cycles are issued in CPU space. Refer to **4.6.4 CPU Space Cycles** and **4.8 Interrupts** for more information. The SCIM2E chip selects operate identically in each type of space, but base address and option registers must be properly programmed for each type of external bus cycle.

During a CPU space cycle, bits [15:3] of the appropriate base register must be configured to match ADDR[23:11], as the address is compared to an address generated by the CPU.

Figure 4-23 shows CPU space encoding for an interrupt acknowledge cycle. FC[2:0] drive 0b111, designating a CPU space access. ADDR[3:1] denote interrupt priority, and the space type field (ADDR[19:16]) is set to 0b1111, the interrupt acknowledge code. The rest of the address lines are set to one.

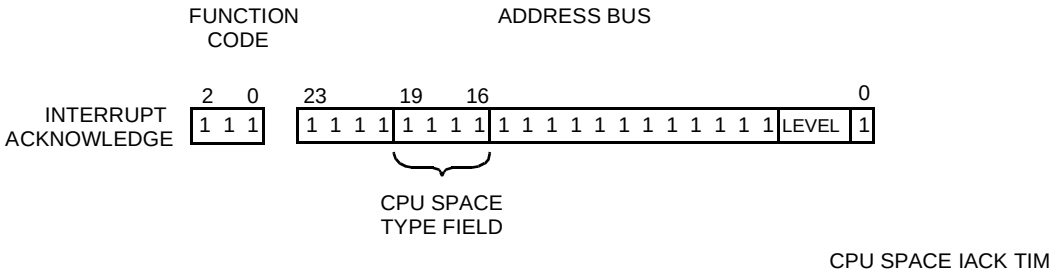


Figure 4-23 CPU Space Encoding for Interrupt Acknowledge

Chip-select address match logic functions only after the SCIM2E has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2E will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$.

Use the procedure that follows to configure a chip select to provide IACK cycle termination.

1. Program the base address field to all ones.
2. Program block size to no more than 64 Kbytes, so that the address comparator checks ADDR[19:16] against the corresponding bits in the base address register. (The CPU space bus cycle type is placed on ADDR[19:16]).
3. Set the R/$\overline{\text{W}}$ field to read only. An interrupt acknowledge cycle is performed as a read in CPU space.
4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector must be generated, either by asserting the $\overline{\text{AVEC}}$ pin or by having the chip select assert $\overline{\text{AVEC}}$ internally. The latter is accomplished by setting the chip-select option register $\overline{\text{AVEC}}$ bit. This terminates the bus cycle.

4.9.4.2 Chip-Select Reset Operation

The LSB of each of the 2-bit pin assignment fields in CSPAR0 and CSPAR1 has a reset value of one. The reset values of the MSBs of each field are determined by the states of DATA[7:1] during reset. Weak internal pull-up devices condition each of the data lines so that chip-select operation is selected by default out of reset. Excessive bus loading can overcome the internal pull-up devices, resulting in inadvertent configuration out of reset. Use external pull-up resistors or active devices to avoid this.

The base address fields in chip-select base address registers CSBAR[0,3, 5:10], CSBAR3, and CSBAR0 and chip-select option registers CSOR[10:5], CSOR3, and CSOR0 have the reset values shown in **Table 4-40**. The BYTE and R/$\overline{\text{W}}$ fields of each option register have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

Table 4-40 Chip-Select Base and Option Register Reset Values

Fields	Reset Values
Base address	0x000000
Block size	2 Kbytes
Async/sync Mode	Asynchronous mode
Upper/lower byte	Disabled
Read/write	Disabled
$\overline{AS}$/$\overline{DS}$	$\overline{AS}$
$\overline{DSACK}$	No wait states
Address space	CPU space
IPL	Any level
Autovector	External interrupt vector

Following reset, the MCU fetches initialization values from the reset vector, beginning at 0x000000 in supervisor program space. The $\overline{CSBOOT}$ chip-select signal is enabled and can select an external boot device mapped to a base address of 0x000000.

The MSB of the CSBTPA field in CSPAR0 has a reset value of one, so that chip-select function is selected by default out of reset. The BYTE field in chip-select option register CSORBT has a reset value of “both bytes” so that the select signal is enabled out of reset. The LSB of the $\overline{CSBOOT}$ field, determined by the logic level of DATA0 during reset, selects the boot ROM port size. When DATA0 is held low during reset, a port size of eight bits is selected. When DATA0 is held high during reset, a port size of 16 bits is selected. DATA0 has a weak internal pull-up device, so that a 16-bit port is selected by default out of reset. As mentioned above, the internal pull-up device can be overcome by bus loading effects. To ensure a particular configuration out of reset, use a pull-up resistor or an active device to place DATA0 in a known state during reset.

The base address field in the boot chip-select base address register CSBARBT has a reset value of all zeros, so that when the initial access to address 0x000000 is made, an address match occurs, and the $\overline{CSBOOT}$ signal is asserted. The block size field in CSBARBT has a reset value of 0b111 (one Mbyte on CPU32-based MCUs and 512 Kbytes on CPU16-based MCUs). **Table 4-41** shows $\overline{CSBOOT}$ reset values.

Table 4-41 $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values

Fields	Reset Values
Base address	0x000000
Block size	1 Mbyte
Async/sync mode	Asynchronous mode
Upper/lower byte	Both bytes
Read/write	Read/write
$\overline{\text{AS}}$/$\overline{\text{DS}}$	$\overline{\text{AS}}$
$\overline{\text{DSACK}}$	13 wait states
Address space	Supervisor space
IPL	Any level
Autovector	External vector externally

4.10 General-Purpose Input/Output

The SCIM2E has six general-purpose input/output ports: A, B, E, F, G, and H. (Port C, an output-only port, is included under 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 and 8-bit expanded modes only. Ports E, F, G, and H have associated data direction registers to configure each port pin as an input or output. Ports A and B share a data direction register that configures each port entirely as inputs or outputs. Ports E and F have associated pin assignment registers that allow the digital I/O or alternate function of each port pin to be selected. Port F has an edge-detect flag register that indicates whether a transition has occurred on any of its pins.

Table 4-42 shows the shared functions of the general-purpose I/O ports and the modes in which they are available.

Table 4-42 General-Purpose I/O Ports

Port	Shared Function	Modes
A	ADDR[18:11]	Single-chip
B	ADDR[10:3]	Single-chip
E	Bus control signals	All
F	$\overline{\text{IRQ}}$[7:1]/FASTREF	All
G	DATA[15:8]	Single-chip
H	DATA[7:0]	Single-chip, 8-bit expanded

Access to the port A, B, E, G, and H data and data direction registers, and the port E pin assignment register require three clock cycles to ensure timing compatibility with external port replacement logic. Accesses to the port F registers require two clock cycles. 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 the port A-B-G-H data registers.

If emulation mode is enabled, accesses to the port A, B, F, G, and H data and data direction registers and the port E pin assignment register are mapped externally, and cause the $\overline{CSE}$ port emulation chip select to be asserted. The SCIM2E does not respond to these accesses, but allows external logic, such as the Motorola MC68HC33 port replacement unit (PRU), to respond. Accesses to the port F registers will still be handled by the SCIM2E.

A write to the port A, B, E, F, G, or H data register is stored in the port's internal data latch. 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 data latch.

4.10.1 Ports A and B

Ports A and B are available in single-chip mode only. One data direction register controls the data direction for both ports. The SCIM2E will respond to port A and B registers at any time the MCU is not in emulation mode.

The port A/B data direction bits (DDA and DDB) control the direction of the pin drivers for ports A and B, respectively. Setting DDA or DDB to one configures all corresponding port pins as outputs. Clearing DDA or DDB to zero configures all corresponding port pins as inputs.

4.10.2 Port A and B Data Registers

PORTA — Port A Data Register **0xYF FA0A**
PORTB — Port B Data Register **0xYF FA0B**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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.

4.10.3 Port E

Port E can be made available in all operating modes. The state of $\overline{BERR}$ and DATA8 at the release of $\overline{RESET}$ controls whether the port E pins are initially configured as bus control signals or discrete I/O lines.

If the MCU is in emulation mode, accesses to the port E data, data direction, and pin assignment registers (PORTE, DDRE, and PEPAR) are mapped externally. This allows port replacement logic to be supplied externally, giving an emulator access to the bus control signals.

4.10.3.1 Port E Data Register

PORTE0 — Port E0 Data Register **0xYF FA11**
PORTE1 — Port E1 Data Register **0xYF FA13**

MSB 7	6	5	4	3	2	1	LSB 0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:							
U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this 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. Bits [15:8] are reserved and will always read zero.

4.10.3.2 Port E Data Direction Register

DDRAB — Port A/B Data Direction Register **0xYF FA14**
DDRE — Port E Data Direction Register **0xYF FA15**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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

The port E data direction register controls the direction of the port E pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

The port A/B data direction register controls the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB to one configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs. Bits [15:10] are reserved and will always read zero.

4.10.3.3 Port E Pin Assignment Register

PEPAR — Port E Pin Assignment **0xYF FA17**

MSB 7	6	5	4	3	2	1	LSB 0
PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:							
DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

This register determines the function of port E pins. Setting a bit assigns the corresponding pin to a bus control signal; clearing a bit assigns the pin to I/O port E. Bits

[15:8] are reserved and will always read zero. **Table 4-43** displays port E pin assignments.

Table 4-43 Port E Pin Assignments

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

4.10.4 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. **Figure 4-24** is a block diagram of port F pins, registers, and control logic.

Port F pins can be configured as interrupt request inputs, edge-detect input/outputs, or discrete input/outputs. When port F pins are configured for edge detection, and a priority level is specified in the port F edge-detect interrupt level register (PFLVR), the port F control logic will generate an interrupt request when the specified edge is detected. Interrupt vector assignment is made by writing a value to the port F edge-detect interrupt vector register (PFIVR). Edge-detect interrupts have the lowest arbitration priority in the SCIM2E.

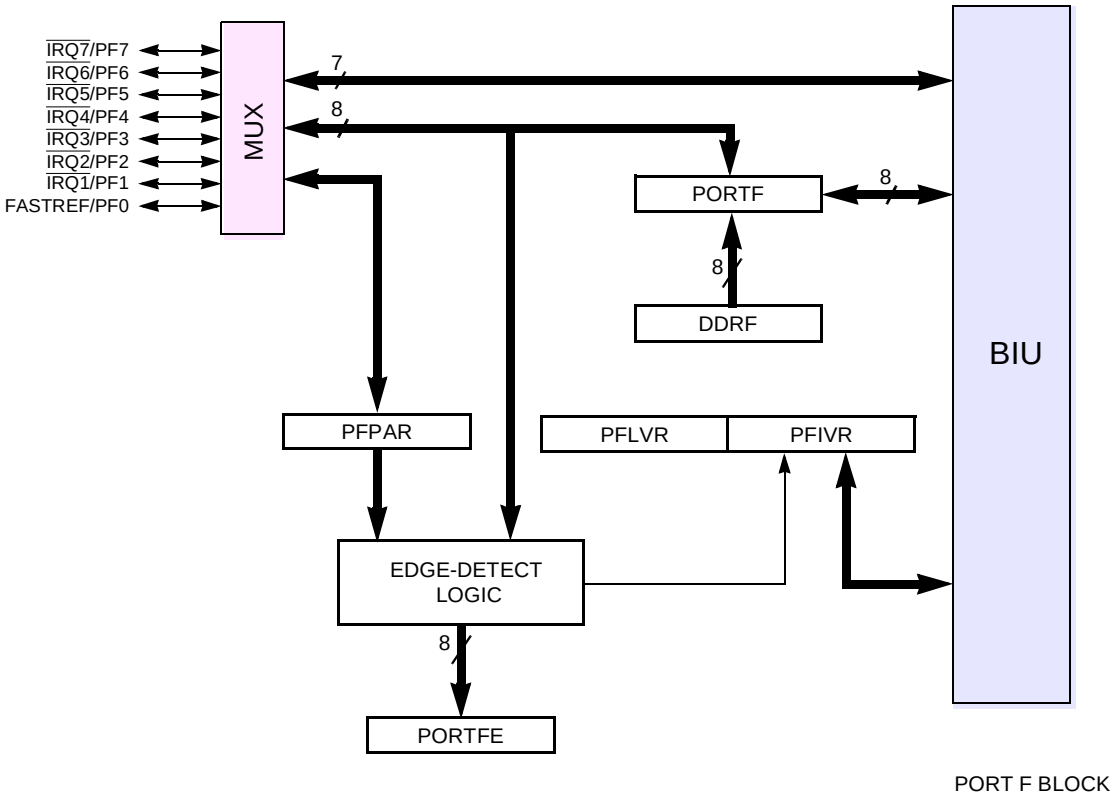


Figure 4-24 Port F Block Diagram

4.10.4.1 Port F Data Register

PORTF0 — Port F Data Register 0 **0xYF FA19**
PORTF1 — Port F Data Register 1 **0xYF FA1B**

MSB 7	6	5	4	3	2	1	LSB 0
PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:							
U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this 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. Bits [15:8] are reserved and will always read zero.

4.10.4.2 Port F Data Direction Register

DDRF — Port F Data Direction Register **0xYF FA1D**

MSB 7	6	5	4	3	2	1	LSB 0
DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:							
0	0	0	0	0	0	0	0

This register controls the direction of the port F pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time. Bits [15:8] are reserved and will always read zero.

4.10.4.3 Port F Pin Assignment Register

PFPAR — Port F Pin Assignment Register **0xYF FA1F**

MSB 7	6	5	4	3	2	1	LSB 0
PFPA3		PFPA2		PFPA1		PFPA0	
RESET							
8- AND 16-BIT EXPANDED MODES							
DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
SINGLE-CHIP MODE							
0	0	0	0	0	0	0	0

This register determines the function of port F pins. Setting a bit assigns the corresponding pin to a control signal; clearing a bit assigns the pin to port F. Bits [15:8] are reserved and will always read zero. **Table 4-44** shows port F pin assignments. **Table 4-45** shows PFPAR pin functions.

Table 4-44 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]	$\overline{\text{IRQ1}}$, FASTREF

Table 4-45 PFPAR Pin Functions

PFPAx[1:0]	Port F Signal
00	I/O pin without edge detect
01	Rising edge detect
10	Falling edge detect
11	Interrupt request

4.10.4.4 Port F Edge-Detect Flag Register

PORTFE — Port F Edge-Detect Flag Register **0xYF FA29**

MSB 7	6	5	4	3	2	1	LSB 0
PEF7	PEF6	PEF5	PEF4	PEF3	PEF2	PEF1	PEF0
RESET:							
0	0	0	0	0	0	0	0

When the corresponding pin is configured for edge detection, a PORTFE bit is set if an edge is detected. 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. Bits [15:8] are reserved and will always read zero.

4.10.4.5 Port F Edge-Detect Interrupt Vector

PFIVR — Port F Edge-Detect Interrupt Vector Register **0xYF FA2B**

MSB 7	6	5	4	3	2	1	LSB 0
PFIVR[7:0]							
RESET:							
0	0	0	0	0	0	0	0

This register determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIVR[7:0] to the value pointing to the appropriate interrupt vector. Bits [15:8] are reserved and will always read zero.

4.10.4.6 Port F Edge-Detect Interrupt Level

PFLVR — Port F Edge-Detect Interrupt Level Register **0xYF FA2D**

MSB 7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	PFLV[2:0]		
RESET:							
0	0	0	0	0	0	0	0

This register determines the priority level of the port F edge-detect interrupt. The reset value is 0x00, indicating that the interrupt is disabled. When several sources of interrupts from the SCIM are arbitrating for the same level, the port F edge-detect interrupt has the lowest arbitration priority. Bits [15:8] are reserved and will always read zero.

4.10.5 Port G

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

The SCIM2E will respond to port G data register (PORTG) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

4.10.5.1 Port G and H Data Registers

PORTG — Port G Data Register **0xYF FA0C**
PORTH — Port H Data Register **0xYF FA0D**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

These port data registers can be read or written any time the MCU is not in emulation mode. Reset has no effect.

4.10.5.2 Port G and H Data Direction Registers

DDRG — Port G Data Direction Register **0xYF FA0E**
DDRH— Port H Data Direction Register **0xYF FA0F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 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 this register control the direction of the port pin drivers when pins are configured as I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

4.10.6 Port H

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

The SCIM2E will respond to port H data register (PORTH) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

The port H data direction register (DDRH) controls the direction of the pin drivers when port H pins are configured for I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.