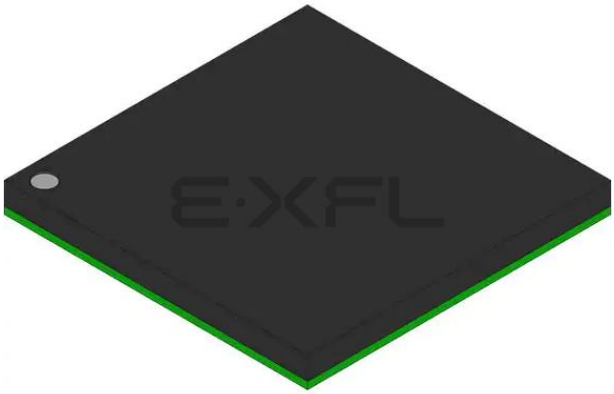

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	Coldfire V2
Core Size	32-Bit Single-Core
Speed	66MHz
Connectivity	EBI/EMI, Ethernet, I²C, SPI, UART/USART, USB
Peripherals	DMA, WDT
Number of I/O	32
Program Memory Size	16KB (4K x 32)
Program Memory Type	ROM
EEPROM Size	-
RAM Size	1K x 32
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	196-LBGA
Supplier Device Package	196-MAPBGA (15x15)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mcf5272cvf66

List of Figures (Continued)

Table xiv. PLIC Module Memory Map (continued)

MBAR Offset	Register Name	Old Mnemonic	New Mnemonic
0x0350	Port0-1 GCI/IDL Configuration Register	PLCR0 PLCR1	**P0CR** **P1CR**
0x0354	Port2-3 GCI/IDL Configuration Register	PLCR2 PLCR3	**P2CR** **P3CR**
0x0358	Port0-1 Interrupt Configuration Register	PLICR0 PLICR1	**P0ICR** **P1ICR**
0x035C	Port2-3 Interrupt Configuration Register	PLICR2 PLICR3	**P2ICR** **P3ICR**
0x0360	Port0-1 GCI Monitor RX	PLGMR0 PLGMR1	**P0GMR** **P1GMR**
0x0364	Port2-3 GCI Monitor RX	PLGMR2 PLGMR3	**P2GMR** **P3GMR**
0x0368	Port0-1 GCI Monitor TX	PLGMT0 PLGMT1	**P0GMT** **P1GMT**
0x036C	Port2-3 GCI Monitor TX	PLGMT2 PLGMT3	**P2GMT** **P3GMT**
0x0370	GCI Monitor TX Status GCI Monitor TX abort	PLGMTS PLGMTA	**PGMTS** **PGMTA**
0x0374	Port0-3 GCI C/I RX	PLGCIR0 PLGCIR1 PLGCIR2 PLGCIR3	**P0GCIR** **P1GCIR** **P2GCIR** **P3GCIR**
0x0378	Port0-3 GCI C/I TX	PLGCIT0 PLGCIT1 PLGCIT2 PLGCIT3	**P0GCIT** **P1GCIT** **P2GCIT** **P3GCIT**
0x037C	GCI C/I TX Status	PGCITSR	No change
0x0384	Port0-1 Periodic Status	PLPSR0 PLPSR1	**P0PSR** **P1PSR**
0x0388	Port2-3 Periodic Status	PLPSR2 PLPSR3	**P2PSR** **P3PSR**
0x038C	Aperiodic Interrupt Status Register; Loop back Control	PLASR PLLCR	**PASR** **PLCR**
0x0392	D Channel Request	PLDRQ	**PDRQR**
0x0394	Port0-1 Sync Delay	PLSD0 PLSD1	**P0SDR** **P1SDR**
0x0398	Port2-3 Sync Delay	PLSD2 PLSD3	**P2SDR** **P3SDR**
0x039C	Clock Select	PLCKSEL	**PCSR**

Table xvi. USB Module Memory Map (continued)

MBAR Offset	Register Name	Old Mnemonic	New Mnemonic
0x1013	USB Function Address Register	USBFAR	**FAR**
0x1014	USB Alternate Setting Register	USBASR	**ASR**
0x1018	USB Device Request Data1 Register	USBDRR1	**DRR1**
0x101C	USB Device Request Data2 Register	USBDRR2	**DRR2**
0x1022	USB Specification Number Register	USBSPECR	**SPECR**
0x1026	USB Endpoint 0 Status Register	USBEPSR0	**EP0SR**
0x1028	USB Endpoint 0 IN Config Register	USBEPICFG0	**IEP0CFG**
0x102C	USB Endpoint 0 OUT Config Register	USBEPOCFG0	**OEP0CFG**
0x1030	USB Endpoint 1 Configuration Register	USBEPCFG1	**EP1CFG**
0x1034	USB Endpoint 2 Configuration Register	USBEPCFG2	**EP2CFG**
0x1038	USB Endpoint 3 Configuration Register	USBEPCFG3	**EP3CFG**
0x103C	USB Endpoint 4 Configuration Register	USBEPCFG4	**EP4CFG**
0x1040	USB Endpoint 5 Configuration Register	USBEPCFG5	**EP5CFG**
0x1044	USB Endpoint 6 Configuration Register	USBEPCFG6	**EP6CFG**
0x1048	USB Endpoint 7 Configuration Register	USBEPCFG7	**EP7CFG**
0x104C	USB Endpoint 0 Control Register	USBEPCTL0	**EP0CTL**
0x1052	USB Endpoint 1 Control Register	USBEPCTL1	**EP1CTL**
0x1056	USB Endpoint 2 Control Register	USBEPCTL2	**EP2CTL**
0x105A	USB Endpoint 3 Control Register	USBEPCTL3	**EP3CTL**
0x105E	USB Endpoint 4 Control Register	USBEPCTL4	**EP4CTL**
0x1062	USB Endpoint 5 Control Register	USBEPCTL5	**EP5CTL**
0x1066	USB Endpoint 6 Control Register	USBEPCTL6	**EP6CTL**
0x106A	USB Endpoint 7 Control Register	USBEPCTL7	**EP7CTL**
0x106C	USB General/Endpoint 0 Interrupt Status Register	USBEPISR0	**EP0ISR**
0x1072	USB Endpoint 1 Interrupt Status Register	USBEPISR1	**EP1ISR**
0x1076	USB Endpoint 2 Interrupt Status Register	USBEPISR2	**EP2ISR**
0x107A	USB Endpoint 3 Interrupt Status Register	USBEPISR3	**EP3ISR**
0x107E	USB Endpoint 4 Interrupt Status Register	USBEPISR4	**EP4ISR**
0x1082	USB Endpoint 5 Interrupt Status Register	USBEPISR5	**EP5ISR**
0x1086	USB Endpoint 6 Interrupt Status Register	USBEPISR6	**EP6ISR**
0x108A	USB Endpoint 7 Interrupt Status Register	USBEPISR7	**EP7ISR**
0x108C	USB Endpoint 0 Interrupt Mask Register	USBEPIMR0	**EP0IMR**

Table 2-21. MCF5272 Exceptions (continued)

Exception	Description
Illegal Instruction	On Version 2 ColdFire implementations, only some illegal opcodes (0x0000 and 0x4AFC) are decoded and generate an illegal instruction exception. Additionally, attempting to execute an illegal line A or line F opcode generates unique exception types: vectors 10 and 11, respectively. If any other nonsupported opcode is executed, the resulting operation is undefined. ColdFire processors do not provide illegal instruction detection on extension words of any instruction, including MOVEC. Attempting to execute an instruction with an illegal extension word causes undefined results.
Divide by Zero	Attempted division by zero causes an exception (vector 5, offset = 0x014) except when the PC points to the faulting instruction (DIVU, DIVS, REMU, REMS).
Privilege Violation	Caused by attempted execution of a supervisor mode instruction while in user mode. The *ColdFire Programmer's Reference Manual* lists supervisor- and user-mode instructions.
Trace Exception	ColdFire processors provide instruction-by-instruction tracing. While the processor is in trace mode (SR[T] = 1), instruction completion signals a trace exception. This allows a debugger to monitor program execution. The only exception to this definition is the STOP instruction. If the processor is in trace mode, the instruction before the STOP executes and then generates a trace exception. In the exception stack frame, the PC points to the STOP opcode. When the trace handler is exited, the STOP instruction is executed, loading the SR with the immediate operand from the instruction. The processor then generates a trace exception. The PC in the exception stack frame points to the instruction after STOP, and the SR reflects the just-loaded value. If the processor is not in trace mode and executes a STOP instruction where the immediate operand sets the trace bit in the SR, hardware loads the SR and generates a trace exception. The PC in the exception stack frame points to the instruction after STOP, and the SR reflects the just-loaded value. Because ColdFire processors do not support hardware stacking of multiple exceptions, it is the responsibility of the operating system to check for trace mode after processing other exception types. As an example, consider a TRAP instruction executing in trace mode. The processor initiates the TRAP exception and passes control to the corresponding handler. If the system requires that a trace exception be processed, the TRAP exception handler must check for this condition (SR[15] in the exception stack frame asserted) and pass control to the trace handler before returning from the original exception.
Debug Interrupt	Caused by a hardware breakpoint register trigger. Rather than generating an IACK cycle, the processor internally calculates the vector number (12). Additionally, the M bit and the interrupt priority mask fields of the SR are unaffected by the interrupt. See Section 2.2.2.1, "Status Register (SR)."
RTE and Format Error Exceptions	When an RTE instruction executes, the processor first examines the 4-bit format field to validate the frame type. For a ColdFire processor, any attempted execution of an RTE where the format is not equal to {4,5,6,7} generates a format error. The exception stack frame for the format error is created without disturbing the original exception frame and the stacked PC points to RTE.The selection of the format value provides limited debug support for porting code from M68000 applications. On M68000 Family processors, the SR was at the top of the stack. Bit 30 of the longword addressed by the system stack pointer is typically zero; so, attempting an RTE using this old format generates a format error on a ColdFire processor. If the format field defines a valid type, the processor does the following: 1 Reloads the SR operand. 2 Fetches the second longword operand. 3 Adjusts the stack pointer by adding the format value to the auto-incremented address after the first longword fetch. 4 Transfers control to the instruction address defined by the second longword operand in the stack frame.
TRAP	Executing TRAP always forces an exception and is useful for implementing system calls. The trap instruction may be used to change from user to supervisor mode.

5.4.5 Data Breakpoint/Mask Registers (DBR, DBMR)

The data breakpoint register (DBR), Figure 5-8, specify data patterns used as part of the trigger into debug mode. DBR bits are masked by setting corresponding DBMR bits, as defined in TDR.

	31 … 0
Field	Data (DBR); Mask (DBMR)
Reset	Uninitialized
R/W	DBR is accessible in supervisor mode as debug control register 0x0E, using the WDEBUG instruction and through the BDM port using the RDMREG and WDMREG commands. DBMR is accessible in supervisor mode as debug control register 0x0F,using the WDEBUG instruction and via the BDM port using the WDMREG command.
DRc[4–0]	0x0E (DBR), 0x0F (DBMR)

Figure 5-8. Data Breakpoint/Mask Registers (DBR and DBMR)

Table 5-9 describes DBR fields.

Table 5-9. DBR Field Descriptions

Bits	Name	Description
31–0	Data	Data breakpoint value. Contains the value to be compared with the data value from the processor's local bus as a breakpoint trigger.

Table 5-10 describes DBMR fields.

Table 5-10. DBMR Field Descriptions

Bits	Name	Description
31–0	Mask	Data breakpoint mask. The 32-bit mask for the data breakpoint trigger. Clearing a DBR bit allows the corresponding DBR bit to be compared to the appropriate bit of the processor's local data bus. Setting a DBMR bit causes that bit to be ignored.

The DBR supports both aligned and misaligned references. Table 5-11 shows relationships between processor address, access size, and location within the 32-bit data bus.

Table 5-11. Access Size and Operand Data Location

A[1:0]	Access Size	Operand Location
00	Byte	D[31:24]
01	Byte	D[23:16]
10	Byte	D[15:8]
11	Byte	D[7:0]
0x	Word	D[31:16]
1x	Word	D[15:0]
xx	Longword	D[31:0]

Command Sequence:

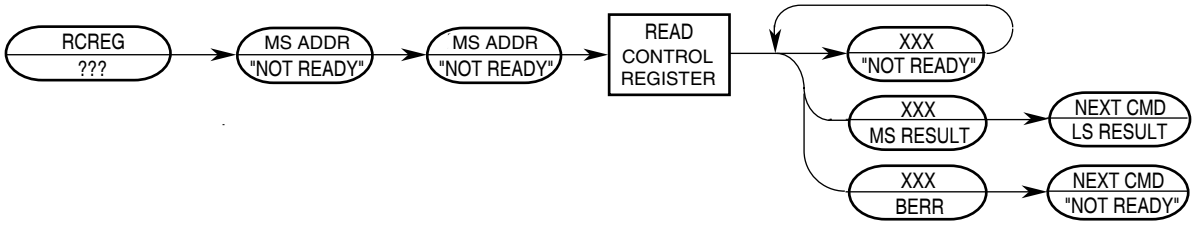

Figure 5-34. RCREG Command Sequence

Operand Data: The only operand is the 32-bit Rc control register select field.

Result Data: Control register contents are returned as a longword, most-significant word first. The implemented portion of registers smaller than 32 bits is guaranteed correct; other bits are undefined.

5.5.3.3.10 Write Control Register (WCREG)

The operand (longword) data is written to the specified control register. The write alters all 32 register bits.

Command/Result Formats:

	15–12	11–8	7–4	3–0
Command	0x2	0x8	0x8	0x0
	0x0	0x0	0x0	0x0
	0x0	Rc		
Result	D[31:16]			
	D[15:0]			

Figure 5-35. WCREG Command/Result Formats

Command Sequence:

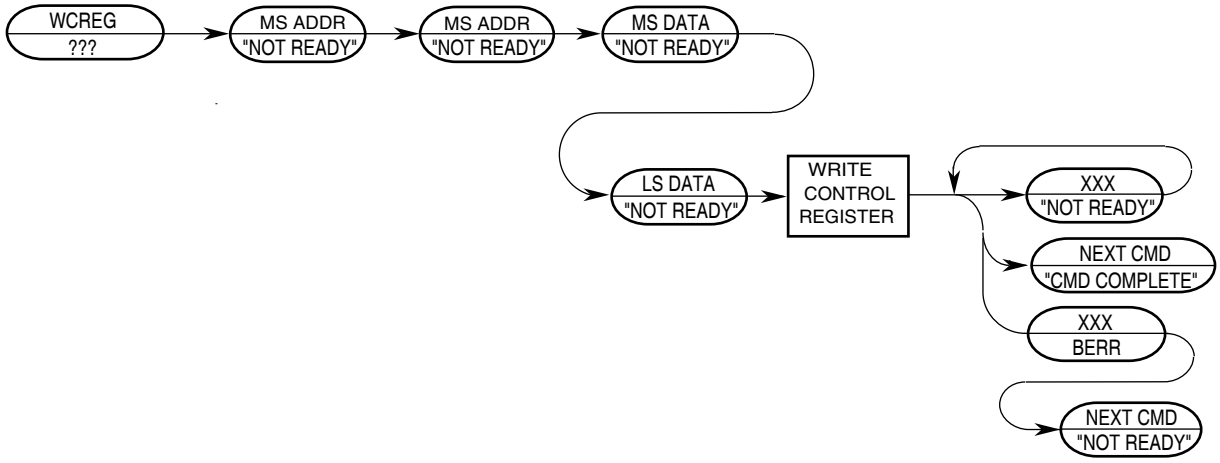

Figure 5-36. WCREG Command Sequence

8.2.1 Chip Select Base Registers (CSBR0–CSBR7)

The CSBRs, Figure 8-1, provide a model internal bus cycle against which to match actual bus cycles to determine whether a specific chip select should assert. A bus cycle in a specific chip select register causes the assertion of the corresponding external chip select.

	31 12	11 10	9 8	7	6 5	4 2	1	0
Field	BA	EBI	BW	SUPER	TT	TM	CTM	ENABLE
Reset	CSBR0: 0x0000_0x01[1]; CSBR1: 0x0000_1300; CSBR2: 0x0000_2300; CSBR3: 0x0000_3300; CSBR4: 0x0000_4300; CSBR5: 0x0000_5300; CSBR6: 0x0000_6300; CSBR7: 0x0000_7700							
R/W	R/W							
Addr	0x040 (CSBR0); 0x048 (CSBR1); 0x050 (CSBR2); 0x058 (CSBR3); 0x060 (CSBR4); 0x068 (CSBR5); 0x070 (CSBR6); 0x078 (CSBR7)							

Figure 8-1. Chip Select Base Registers (CSBR*n*)

Table 8-2 describes CSBR*n* fields.

Table 8-2. CSBR*n* Field Descriptions

Bits	Name	Description
31–12	BA	Base address. The starting address of the memory space covered by the chip select. BA is compared with bits 31–12 of the access to determine whether the current bus cycle is intended for this chip select. Any combination of BA bits can be masked in the associated CSOR.
11–10	EBI	External bus interface modes. These modes are used to multiplex outputs and determine timing of the appropriate bus interface module onto the device pins. 00 16-/32-bit SRAM/ROM. For 16-/32-bit wide memory devices with byte strobe inputs. CSBR0[EBI] = 00 at reset. Affects all chip selects. 01 SDRAM. One physical bank of SDRAM consisting of 16–256 Mbit devices. CSOR7[WS] must be set to 0x1F. Affects only $\overline{CS7}$/$\overline{SDCS}$. 10 Reserved 11 Use SRAM/ROM timing for 8-bit devices without byte strobe inputs.
9–8	BW	Bus width. Determines data bus size of the memory-mapped resource for all chip selects except $\overline{CS0}$. It is assumed that boot code for the processor is accessed through the global chip select $\overline{CS0}$, so the initial bus width for this chip select must be configured at reset. QSPI_CS0/BUSW0 and QSPI_CLK/BUSW1 are used to program the bus width for $\overline{CS0}$ at reset. 00 Longword (32 bits) 01 Byte (8 bits) 10 Word (16 bits) 11 Cache line (32 bits)
7	SUPER	Supervisor mode. 0 Bus cycle may be in user or supervisor mode (neglecting conditions imposed by setting CTM). 1 The chip select asserts a match only if the transfer modifier indicates a supervisor mode access. A user access matching BA causes an access error. SUPER, CTM, TT, and TM are used to restrict bus access. For example, if TT and TM indicate a user data access and SUPER and CTM are both set, no accesses can occur.
6–5	TT	Transfer type. TT and TM may be used to further qualify the address match. If CTM is set, TT and TM must match the access types for the chip select to assert. See the description of TM.

9.5.2 SDRAM Timing Register (SDTR)

The SDTR is used to configure SDRAM controller refresh counters for the type of SDRAM devices used and the number of clocks required for each type of SDRAM access. The reset value is 0x2115. For lower CPU clock frequencies, precharge and activate times can be reduced to eliminate up to 2 clock cycles from the read and write accesses. Consult the data sheets of the SDRAMs being considered.

	15–10	9–8	7–6	5–4	3–2	1–0
Write	RTP	RC	—	RP	RCD	CLT
Reset	0010_00	01	00	01	01	01
R/W	R/W					
Addr	MBAR + 0x0186					

Figure 9-4. SDRAM Timing Register (SDTR)

Table 9-8 describes SDTR fields.

Table 9-8. SDTR Field Descriptions

Bits	Name	Description
15–10	RTP	Refresh timer prescaler. Determines the number of clock cycles x 16 between refreshes. The following table describes different recommended prescaler settings for different clock frequencies including a margin of 1.2 μS. Recommended values are as follows: **RTP** / **15.6 μs = 1/f*RTP*16** / **System Clock** 111101 / 61 / 66 MHz 101011 / 43 / 48 MHz 011101 / 29 / 33 MHz 010110 / 22 / 25 MHz 000100 / 4 / 5 MHz (emulator)
9–8	RC	Refresh count. Indicates the number of clock cycles spent in refresh state (RC + 5). Refresh occurs during the first of these clock cycles; the rest of the time is the delay that must occur before the SDRAM is ready to do anything else. 00 5 cycles 01 6 cycles (default) 10 7 cycles 11 8 cycles
7–6	—	Reserved, should be cleared.
5–4	RP	Precharge time. Specifies number of clock cycles taken for a precharge (RP + 1). 00 1 cycle 01 2 cycles (default) 10 3 cycles 11 4 cycles

12.3.2 Register Descriptions

The following are detailed register diagrams and field descriptions.

12.3.2.1 USB Frame Number Register (FNR)

Once every 1 ms, the USB host issues a start-of-frame packet containing a frame number. The FNR register captures this frame number. Figure 12-3 shows the FNR.

	15	11	10	0
Field	—		FRM	
Reset	0000_0000_0000_0000			
R/W	Read			
Addr	MBAR + 0x1002			

Figure 12-3. USB Frame Number Register (FNR)

Table 12-3 describes FNR fields.

Table 12-3. FNR Field Descriptions

Bits	Name	Description
15–11	—	Reserved, should be cleared.
10–0	FRM	USB frame number. Contains the current USB frame number. FRM is updated with the new frame number each time the device successfully receives a start-of-frame (SOF) packet from the USB host. The new frame number is contained in the SOF packet.

12.3.2.2 USB Frame Number Match Register (FNMR)

FNMR is used to signal, via the FRM_MAT interrupt, when a particular frame number is issued by the USB host. To avoid a false match during intermediate states of a byte write operation to the FNMR, Figure 12-4, byte accesses to this register are not supported and cause an access error.

	15	11	10	0
Field	—		FRM_MAT	
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	MBAR + 0x1006			

Figure 12-4. USB Frame Number Match Register (FNMR)

Table 12-4 describes FNMR fields.

Table 12-4. FNMR Field Descriptions

Bits	Name	Description
15–11	—	Reserved, should be cleared
10–0	FRM_MAT	Frame number match value. Contains the USB frame number match value. When the FNR value equals the value in the register, a FRM_MAT interrupt is generated.

12.4 Software Architecture and Application Notes

This section describes architecture and applications.

12.4.1 USB Module Initialization

At power-up, the USB module should be initialized within 20 ms if the USB port is connected. This time corresponds to the 10 ms reset signaling and 10 ms reset recovery time when a device is detected. The following steps are necessary to initialize the USB module for operation:

1. Clear EP0CTL[CFG_RAM_VAL,USB_EN].
2. Load the configuration RAM with the descriptors from external ROM. The starting configuration RAM address is MBAR + 0x1400.
3. Select the internal or external USB transceiver in EPC0TL.
4. Write IEP0CFG and OEP0CFG to initialize the EP0 IN and OUT FIFOs.
5. Initialize the EP0 interrupt vectors.
6. Clear all interrupt bits in EP0ISR.
7. Enable the desired EP0 interrupt sources in EP0IMR.
8. Set EP0CTL[USB_EN, CFG_RAM_VALID].

12.4.2 USB Configuration and Interface Changes

Although the USB module handles the SET_CONFIGURATION and SET_INTERFACE requests, the user is still required to perform some initialization when the configuration or alternate settings change. A configuration or alternate setting change is signaled by the DEV_CFG interrupt. The following steps are required to service the DEV_CFG interrupt.

1. Read EPSR0 and ASR to determine the current configuration and alternate settings.
2. Read the endpoint descriptors for the current configuration and alternate settings to determine the endpoint type and maximum packet size for all of the active endpoints.
3. Write EP*n*CFG for all active endpoints to initialize the FIFOs.
4. Initialize the interrupt vectors for the active endpoints.
5. Clear all interrupt bits in the EP*n*ISR registers for all active endpoints.
6. Enable the desired interrupt sources in the EP*n*IMR registers.
7. Clear the DEV_CFG interrupt bit to allow the USB module to access the FIFOs.

Note that only the active, that is enabled, B- and D-channel receive and transmit registers need to be read and written. B and D channels which are not active need not have their receive and transmit registers read and written.

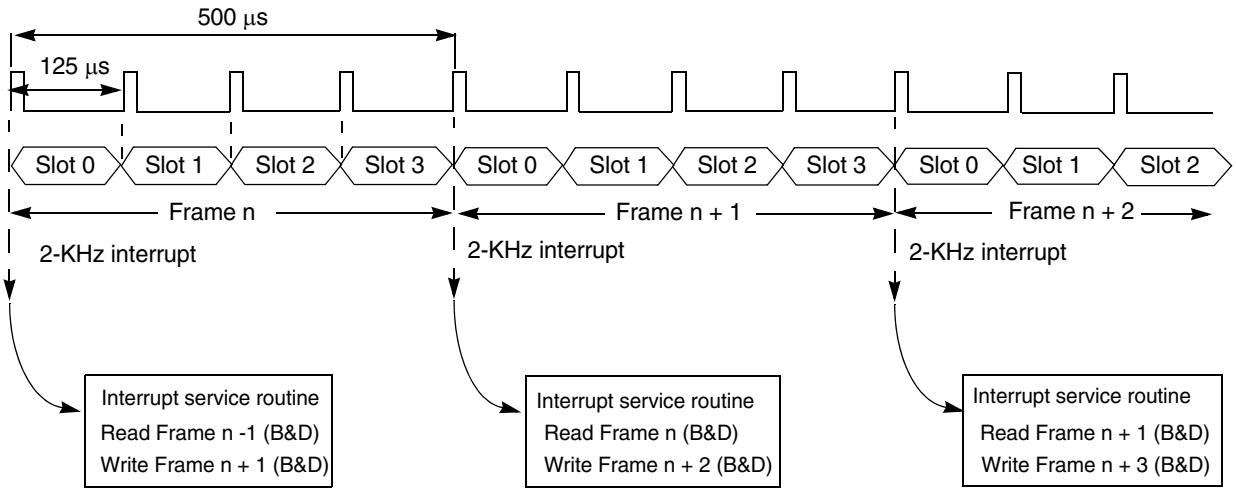

Figure 13-10. Periodic Frame Interrupt

It should be clear from Figure 13-10 that due to the double buffering through the PLIC shadow register, frame (n) is written to the PLIC transmit register during the interrupt service routine of the previous frame, frame (n-1). Similarly on the receive side, frame (n) is read from the PLIC receive register during the interrupt service routine of the following frame, frame (n + 2). Figure 13-10 shows that the minimum delay through the PLIC, when not in loopback mode, is two 2-KHz frames, or 1 mS.

13.2.5.2 GCI Aperiodic Status Interrupt

The aperiodic status interrupt is an interrupt which is driven by a number of conditions. The CPU services this interrupt by reading the aperiodic status register, ASR, and by reading or writing the relevant C/I or monitor channel register or registers which have generated this interrupt. Once read, the interrupt is cleared. Each port and individual interrupts within each port is maskable. The following conditions for each of the ports can trigger this interrupt:

- Monitor channel receive: ASR defines which port or ports have generated a monitor channel receive interrupt. The interrupt service routine must then read the appropriate GMR register or registers to clear the monitor channel receive interrupt.
- Monitor channel transmit: ASR defines which port or ports have generated a monitor channel transmit interrupt. The interrupt service routine must then read the appropriate GMT register or registers to clear the monitor channel transmit interrupt.
- C/I channel receive: ASR defines which port or ports have generated a C/I channel receive interrupt. The interrupt service routine must then read the appropriate GCIR register or registers to clear the C/I channel receive interrupt.
- C/I channel transmit: ASR defines which port or ports have generated a C/I channel transmit interrupt. The interrupt service routine must then read the appropriate GCIT register or registers to clear the C/I channel transmit interrupt.

13.5.15 GCI Monitor Channel Transmit Status Register (PGMTS)

All bits in this register are read only and are cleared on hardware or software reset.

The PGMTS register contains the monitor channel status bits for each of the four transmit ports on the MCF5272.

	7	6	5	4	3	2	1	0
Field	ACK3	ACK2	ACK1	ACK0	AB3	AB2	AB1	AB0
Reset	0000_0000							
R/W	Read Only							
Addr	MBAR + 0x371							

Figure 13-27. GCI Monitor Channel Transmit Status Register (PGMTS)

Table 13-10. PGMTS Field Descriptions

Bits	Name	Description
7	ACK3	Acknowledge, port 3. 0 Default reset value. 1 Indicates to the CPU that the GCI controller has transmitted the previous monitor channel information. Automatically cleared by the CPU reading the register. The clearing of this bit by reading this register also clears the aperiodic GMT interrupt.
6	ACK2	Acknowledge, port 2. See ACK3.
5	ACK1	Acknowledge, port 1. See ACK3.
4	ACK0	Acknowledge, port 0. See ACK3.
3	AB3	Abort, port 3. 0 Default reset value. 1 Indicates to the CPU that the GCI controller has aborted the current message. This bit is automatically cleared by the CPU reading the register. When the GCI controller sets this bit, it also clears the AR bit in the PGMTA register, the ACK bit in the GMTS register, and the L and R bits in the P*n*GMT register.
2	AB2	Abort, port 2. See AB3.
1	AB1	Abort, port 1. See AB3.
0	AB0	Abort, port 0. See AB3.

13.5.17 GCI C/I Channel Transmit Registers (P0GCIT–P3GCIT)

All bits in these registers are read/write and are cleared on hardware or software reset.

The P*n*GCIT registers are 8-bit registers containing the monitor channel bits to be transmitted for each of the four ports on the MCF5272.

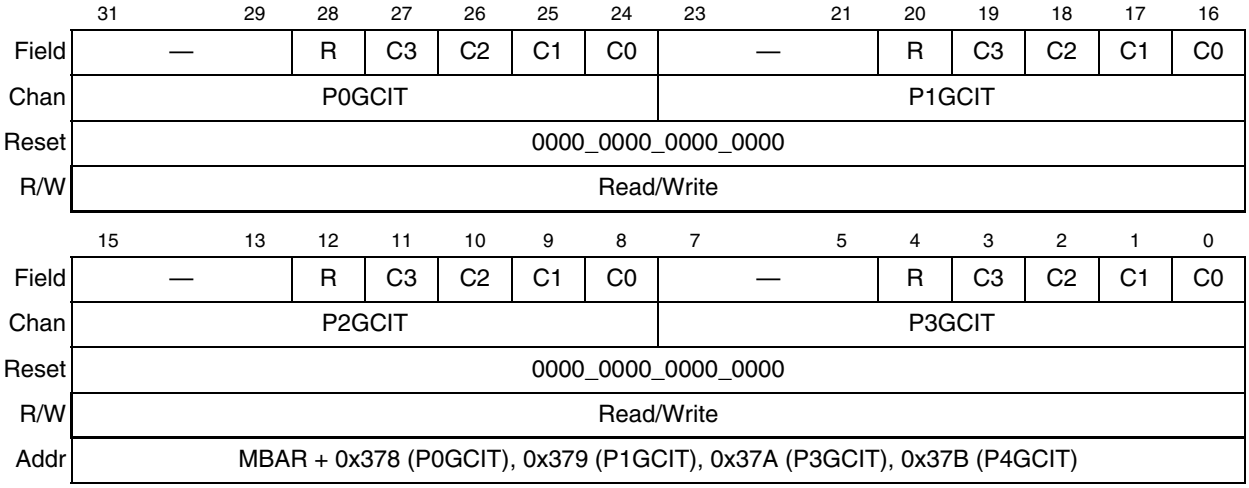

Figure 13-29. GCI C/I Channel Transmit Registers (P0GCIT–P3GCIT)

Table 13-12. P0GCIT–P3GCIT Field Descriptions

Bits	Name	Description
31–29, 23–21, 15–13, 7–5	—	Reserved, should be cleared.
28, 20, 12, 4	R	Ready. This bit is set, by the CPU to indicate to the C/I channel controller that data is ready for transmission. The transition of this bit from a 0 to a 1 starts the C/I state machine which responds with the ACK bit once transmission of two successive C/I words is complete. This bit is automatically cleared by the GCI controller when it generates a transmit acknowledge (ACK bit in PGCITSR register). The clearing of this bit by reading this register also clears the aperiodic GCT interrupt.
27–24, 19–16, 11–8, 3–0	C3–C0	C/I bits. The CPU writes C/I data to be transmitted, on the GCI or SCIT channel 0, into these positions. The CPU must ensure that this data is not overwritten before it has been transmitted the required minimum amount of times, that is, so any change is detected and confirmed by a receiver. A maskable interrupt is generated when this data has been successfully transmitted

14.5.4 QSPI Interrupt Register (QIR)

Figure 14-8 shows the QSPI interrupt register.

	15	14	13	12	11	10	9	8	7–4	3	2	1	0
Field	WCEFB	ABRTB	—	ABRTL	WCEFE	ABRTE	—	SPIFE	—	WCEF	ABRT	—	SPIF
Reset	0000_0000_0000_0000												
R/W	R/W												
Address	MBAR + 0x00AC												

Figure 14-8. QSPI Interrupt Register (QIR)

Table 14-6 describes QIR fields.

Table 14-6. QIR Field Descriptions

Bits	Name	Description
15	WCEFB	Write collision access error enable. A write collision occurs during a data transfer when the RAM entry containing the command currently being executed is written to by the CPU with the QDR. When this bit is asserted, the write access to QDR results in an access error.
14	ABRTB	Abort access error enable. An abort occurs when QDLYR[SPE] is cleared during a transfer. When set, an attempt to clear QDLYR[SPE] during a transfer results in an access error.
13	—	Reserved, should be cleared.
12	ABRTL	Abort lock-out. When set, QDLYR[SPE] cannot be cleared by writing to the QDLYR. QDLYR[SPE] is only cleared by the QSPI when a transfer completes.
11	WCEFE	Write collision interrupt enable. Interrupt enable for WCEF. Setting this bit enables the interrupt, and clearing it disables the interrupt.
10	ABRTE	Abort interrupt enable. Interrupt enable for ABRT flag. Setting this bit enables the interrupt, and clearing it disables the interrupt.
9	—	Reserved, should be cleared.
8	SPIFE	QSPI finished interrupt enable. Interrupt enable for SPIF. Setting this bit enables the interrupt, and clearing it disables the interrupt.
7–4	—	Reserved, should be cleared.
3	WCEF	Write collision error flag. Indicates that an attempt has been made to write to the RAM entry that is currently being executed. Writing a 1 to this bit clears it and writing 0 has no effect.
2	ABRT	Abort flag. Indicates that QDLYR[SPE] has been cleared by the user writing to the QDLYR rather than by completion of the command queue by the QSPI. Writing a 1 to this bit clears it and writing 0 has no effect.
1	—	Reserved, should be cleared.
0	SPIF	QSPI finished flag. Asserted when the QSPI has completed all the commands in the queue. Set on completion of the command pointed to by QWR[ENDQP], and on completion of the current command after assertion of QWR[HALT]. In wraparound mode, this bit is set every time the command pointed to by QWR[ENDQP] is completed. Writing a 1 to this bit clears it and writing 0 has no effect.

The command and data RAM in the QSPI is indirectly accessible with QDR and QAR as 48 separate locations that comprise 16 words of transmit data, 16 words of receive data and 16 bytes of commands.

A write to QDR causes data to be written to the RAM entry specified by QAR[ADDR]. This also causes the value in QAR to increment.

Chapter 15
Timer Module

This chapter describes configuration and operation of the four general-purpose timer modules, timer 0, 1, 2 and 3.

15.1 Overview

The timer module has four identical general-purpose 16-bit timers and a software watchdog timer, described in Chapter 6, “System Integration Module (SIM).”

Each general-purpose timer consists of a timer mode register (TMR*n*), a timer capture register (TCAP*n*), a 16-bit timer counter (TCN*n*), a timer reference register (TRR*n*), and a timer event register (TER*n*). The TMRs contain the prescaler value programmed by the user. The four timer units have the following features:

- Maximum period of 4 seconds at 66 MHz
- 15-nS resolution at 66 MHz
- Programmable sources for the clock input, including external clock
- Input capture capability with programmable trigger edge on input pins
- Output compare with programmable mode for the output pins
- Free run and restart modes

15.2 Timer Operation

The timer units consist of four identical, independent 16-bit timers, timers 0–3. For timers 0 and 1, the clock input to the prescaler is selected from the main clock (divided by 1 or 16) or from the corresponding timer input (TIN0 or TIN1) pin. For timers 2 and 3, the clock input to the prescaler can be selected only from the main clock (divided by 1 or 16). TIN is internally synchronized to the internal clock, with a synchronization delay between two and three main clocks. The clock input source is selected by the CLK bits of the corresponding timer mode register (TMR0–TMR3). The prescaler is programmed to divide the clock input by values from 1 to 256. The output of the prescaler is used as an input to the 16-bit counter.

The maximum timer resolution is one system clock cycle (15 nS at 66 MHz). The maximum period (the reference value is all ones) is 268,435,456 cycles = $2^4 \times 2^8 \times 2^{16}$ (4 seconds at 66 MHz).

The timer can be configured to count until a reference is reached at which point it can either start a new time count immediately or continue to run. The free run/restart bit, TMR*n*[FRR], selects each mode. Upon reaching the reference value, the TER0 or TER1 bit is set, and an interrupt is issued if the output reference interrupt enable bit, TMR[ORI], is set.

Table 19-1. Signal Descriptions Sorted by Function (Sheet 7 of 8)

Configured by (see notes)[1]	Pin Functions				Description	Map BGA Pin	I/O	Drive (mA)	Cpf
	0 (Reset)	1	2	3					
	PST3	—	—	—	Internal processor status 3	D3	O	4	30
	PWM_OUT0	—	—	—	PWM output compare 0	N5	O	4	30
	QSPI_CLK/ BUSW1	—	—	—	QSPI serial clock/CS0 bus width bit 1	L5	O	4	30
	QSPI_CS0/ BUSW0	—	—	—	QSPI peripheral chip select 0/$\overline{CS0}$ bus width bit 0	M5	O	2	30
	QSPI_Din	—	—	—	QSPI data input	P4	I		
	QSPI_Dout/ WSEL	—	—	—	QSPI data output/Bus width selection	N4	I/O	4	30
	R/$\overline{W}$	—	—	—	Read/Write	P14	O	10	30
	RAS0	—	—	—	SDRAM row select strobe	A10	O	10	30
	RSTI	—	—	—	Device reset	M12	I		
	RSTO	—	—	—	Reset output strobe	F4	O	4	30
	SDBA0	—	—	—	SDRAM bank 0 select	J14	O	10	30
	SDBA1	—	—	—	SDRAM bank 1 select	H12	O	10	30
	SDCLK	—	—	—	SDRAM (bus) clock, Same frequency as CPU clock	E14	O	10	30
	SDCLKE	—	—	—	SDRAM clock enable	D13	O	10	30
	$\overline{SDCS}$/ $\overline{CS7}$	—	—	—	SDRAM chip select/$\overline{CS7}$	B10	O	10	30
	SDWE	—	—	—	SDRAM write enable	B9	O	10	30
	BYPASS	—	—	—	Bypass internal test mode	M13	O	4	30
MTMOD[4]	TCK	PSTCLK	—	—	JTAG test clock in/ BDM PSTCLK output	C4	I/O	4	30
MTMOD[4]	TDI	DSI	—	—	JTAG test data in/BDM data in	A4	I		
MTMOD[4]	TDO	DSO	—	—	JTAG test data out /BDM data out	D5	O	4	30
	TEA	—	—	—	BDM debug transfer error acknowledge	A3	I		
	TEST	—	—	—	Device test mode enable	E6	I		
	TIN0	—	—	—	Timer 0 input	L6	I		
MTMOD[4]	TMS	BKPT	—	—	JTAG test mode/BDM select breakpoint input	B4	I		

19.6 Bus Control Signals

This section describes bus control signals.

19.6.1 Output Enable/Read ($\overline{OE}$/RD)

The output enable/read signal ($\overline{OE}$/RD) defines the data transfer direction for the data bus D[31:0] for accesses to SRAM, ROM or external peripherals. A low (logic zero) level indicates a read cycle while a high (logic one) indicates a write cycle.

This signal is normally connected to the $\overline{OE}$ pins of external SRAM, ROM, or FLASH.

19.6.2 Byte Strobes ($\overline{BS}$[3:0])

The byte strobes ($\overline{BS}$[3:0]) define the flow of data on the data bus. During SRAM and peripheral accesses, these outputs indicate that data is to be latched or driven onto a byte of the data when driven low. $\overline{BS}n$ signals are asserted only to the memory bytes used during a read or write access.

$\overline{BS}n$ signals are asserted during accesses to on-chip peripherals but not to on-chip SRAM, cache, or ROM. During SDRAM accesses, these signals indicate a byte transfer between SDRAM and the MCF5272 when driven high.

For SRAM or FLASH devices, $\overline{BS}$[3:0] outputs should be connected to individual byte strobe signals.

For SDRAM devices, $\overline{BS}$[3:0] should be connected to individual SDRAM DQM signals. Note that most SDRAMs associate DQM3 with the MSB, in which case $\overline{BS3}$ should be connected to the SDRAM's DQM3 input.

Table 19-3. Byte Strobe Operation for 32-Bit Data Bus

$\overline{BS3}$	$\overline{BS2}$	$\overline{BS1}$	$\overline{BS0}$	Access Type	Access Size	Data Located On
1	1	1	1	None	None	—
1	1	1	0	FLASH/SRAM	Byte	D[31:24]
1	1	0	1	FLASH/SRAM		D[23:16]
1	0	1	1	FLASH/SRAM		D[15:8]
0	1	1	1	FLASH/SRAM		D[7:0]
1	1	0	0	FLASH/SRAM	Word	D[31:16]
0	0	1	1	FLASH/SRAM		D[15:0]
0	0	0	0	FLASH/SRAM	Longword	D[31:0]
1	1	1	0	SDRAM	Byte	D[7:0]
1	1	0	1	SDRAM		D[15:8]
1	0	1	1	SDRAM		D[23:16]
0	1	1	1	SDRAM		D[31:24]
1	1	0	0	SDRAM	Word	D[15:0]
0	0	1	1	SDRAM		D[31:16]
0	0	0	0	SDRAM	Longword	D[31:0]

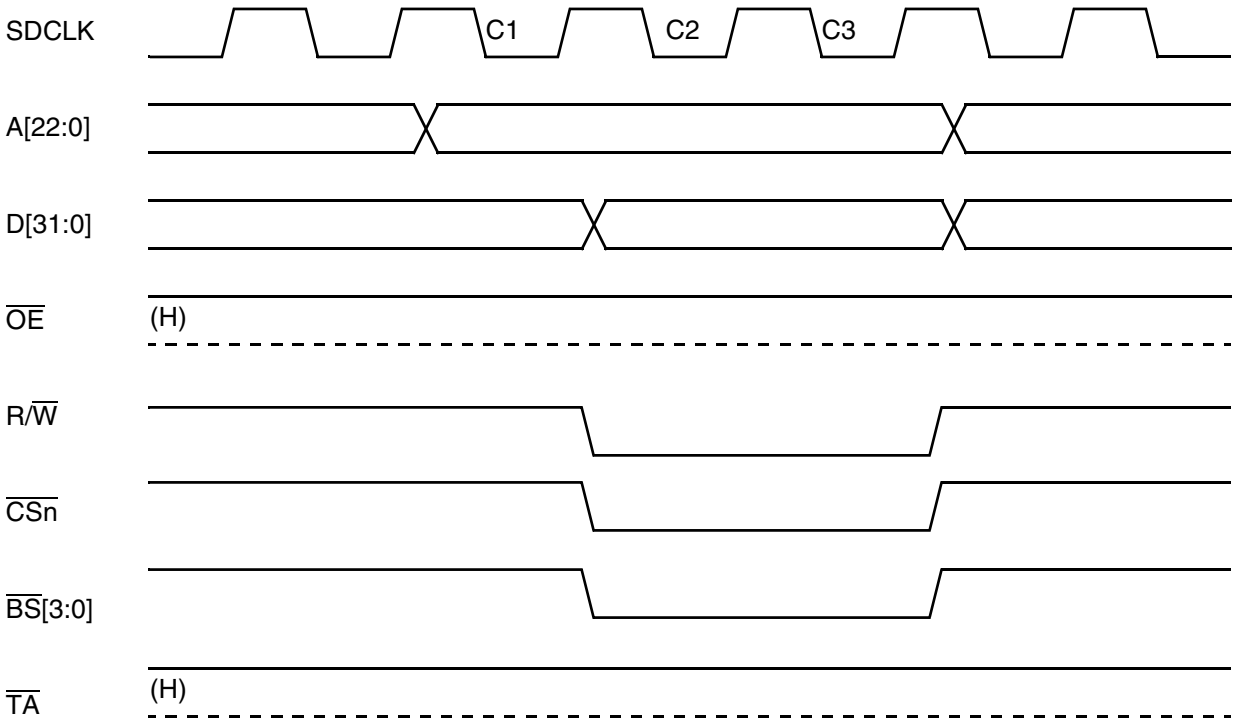

Figure 20-6. Longword Write with Address Setup; EBI = 00; 32-Bit Port; Internal Termination

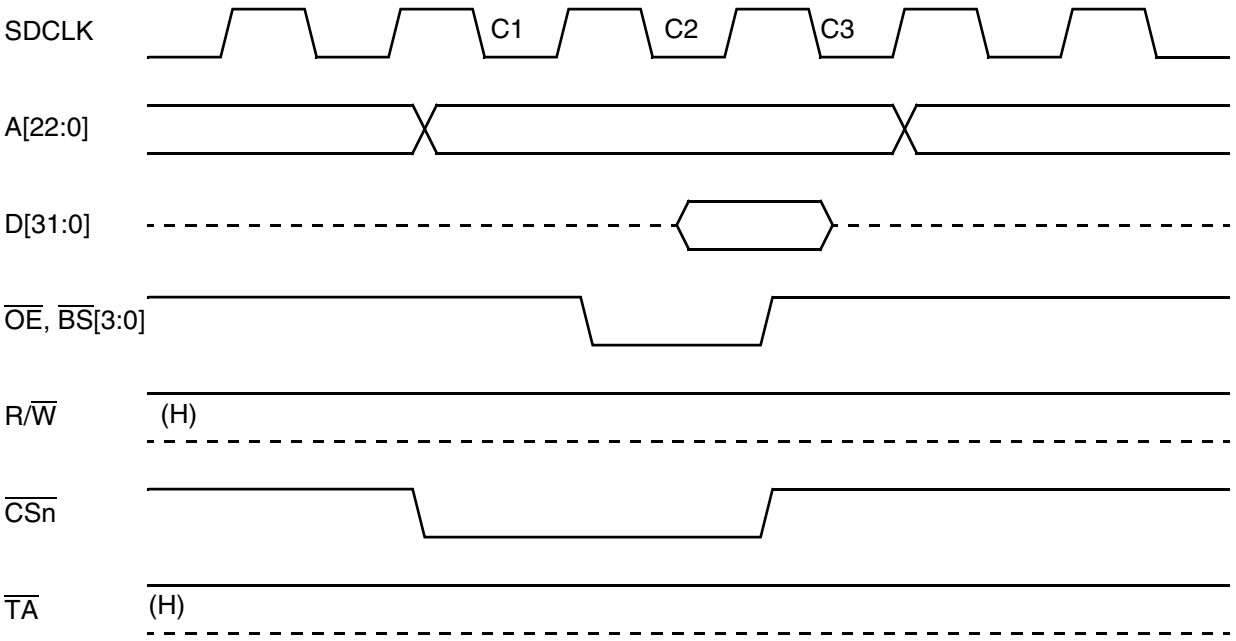

Figure 20-7. Longword Read with Address Hold; EBI = 00; 32-Bit Port; Internal Termination

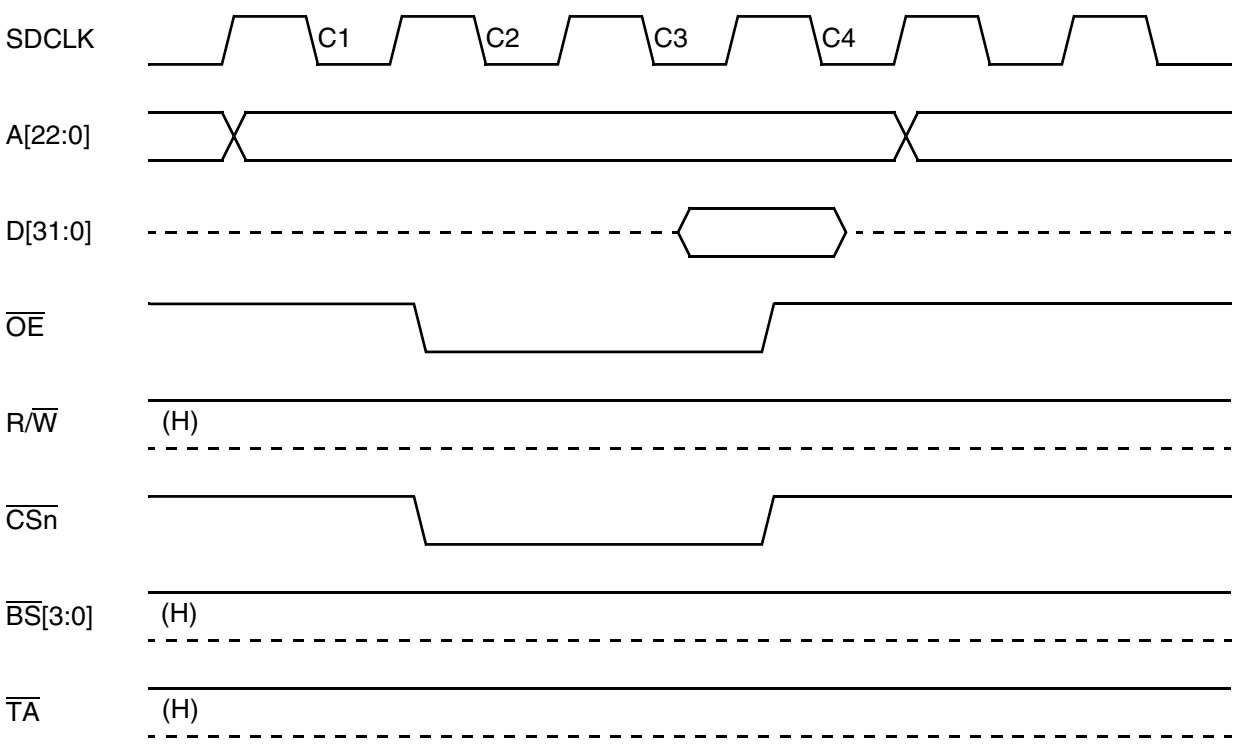

Figure 20-16. Longword Read with Address Setup and Address Hold; EBI = 11; 32-Bit Port, Internal Termination

Table A-7. QSPI Module Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x00A0	QSPI Mode Register (QMR)		Reserved	
0x00A4	QSPI Delay Register (QDLYR)		Reserved	
0x00A8	QSPI Wrap Register (QWR)		Reserved	
0x00AC	QSPI Interrupt Register (QIR)		Reserved	
0x00B0	QSPI Address Register (QAR)		Reserved	
0x00B4	QSPI Data Register (QDR)		Reserved	

Table A-8. PWM Module Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x00C0	PWM Control Register 1 (PWCR1)	Reserved		
0x00C4	PWM Control Register 2 (PWCR2)	Reserved		
0x00C8	PWM Control Register 3 (PWCR3)	Reserved		
0x00D0	PWM Pulse-Width Register 1 (PWWD1)	Reserved		
0x00D4	PWM Pulse-Width Register 2 (PWWD2)	Reserved		
0x00D8	PWM Pulse-Width Register 3 (PWWD3)	Reserved		

Table A-9. DMA Module Memory Map

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x00E0	DMA Mode Register (DCMR)			
0x00E6			DMA Interrupt Register (DCIR)	
0x00E8	DMA Byte Count Register (DBCR)			
0x00EC	DMA Source Address Register (DSAR)			
0x00F0	DMA Destination Address Register (DDAR)			