Chapter 6
SIM Overview

This chapter provides detailed operation information regarding the system integration module (SIM). It describes the SIM programming model, bus arbitration, and system-protection functions for the MCF5307.

6.1 Features

The SIM, shown in Figure 6-1, provides overall control of the bus and serves as the interface between the ColdFire core processor complex and the internal peripheral devices.

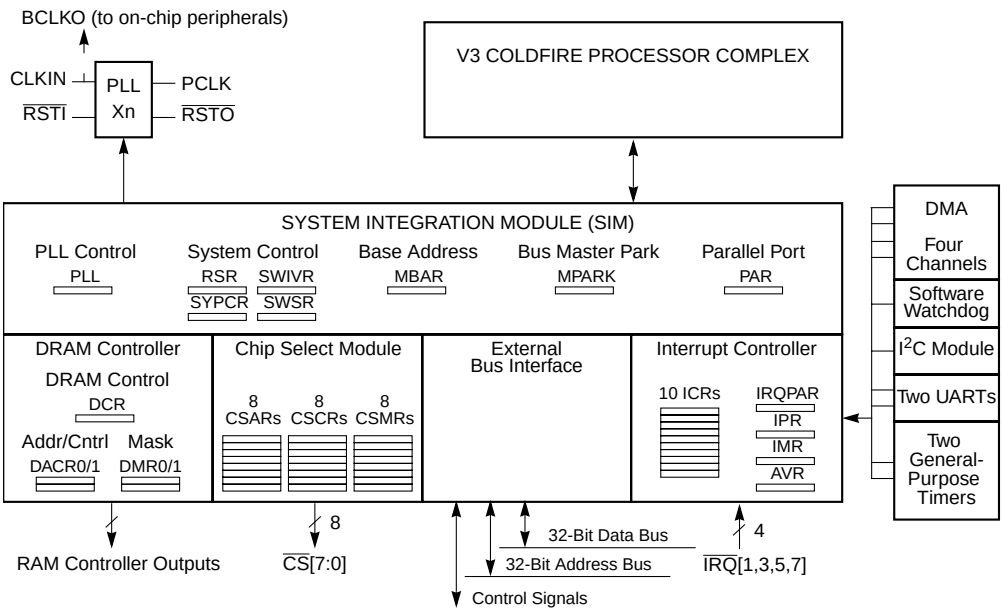

Figure 6-1. SIM Block Diagram

The following is a list of the key SIM features:

- Module base address register (MBAR)
  - — Base address location of all internal peripherals and SIM resources
  - — Address space masking to internal peripherals and SIM resources
- Phase-locked loop (PLL) clock control register (PLLCR) for CPU STOP instruction
  - — Control for turning off clocks to core and interrupt levels that turn clocks back on

  Chapter 7, “Phase-Locked Loop (PLL).”
- Interrupt controller
  - — Programmable interrupt level (1–7) for internal peripheral interrupts
  - — Programmable priority level (0–3) within each interrupt level
  - — Four external interrupts; one set to interrupt level 7; three others programmable to two interrupt levels

  See Chapter 9, “Interrupt Controller.”
- Chip select module
  - — Eight independent, user-programmable chip-select signals ($\overline{CS}$[7:0]) that can interface with SRAM, PROM, EPROM, EEPROM, Flash, and peripherals
  - — Address masking for 64-Kbyte to 4-Gbyte memory block sizes
  - — Programmable wait states and port sizes
  - — External master access to chip selects

  See Chapter 10, “Chip-Select Module.”
- System protection and reset status
  - — Reset status indicating the cause of last reset
  - — Software watchdog timer with programmable secondary bus monitor

  See Section 6.2.4, “Software Watchdog Timer.”
- Pin assignment register (PAR) configures the parallel port. See Section 6.2.9, “Pin Assignment Register (PAR).”
- Bus arbitration
  - — Default bus master park register (MPARK) controls internal and external bus arbitration and enables display of internal accesses on the external bus for debugging
  - — Supports several arbitration algorithms

  See Section 6.2.10, “Bus Arbitration Control.”

6.2 Programming Model

The following sections describe the registers incorporated into the SIM.

6.2.1 SIM Register Memory Map

Table 6-1 shows the memory map for the SIM registers. The internal registers in the SIM are memory-mapped registers offset from the MBAR address pointer defined in MBAR[BA]. This supervisor-level register is described in Section 6.2.2, “Module Base Address Register (MBAR).” Because SIM registers depend on the base address defined in MBAR[BA], MBAR must be programmed before SIM registers can be accessed.

NOTE:

Although external masters cannot access the MCF5307’s on-chip memories or MBAR, they can access any of the SIM memory map and peripheral registers, such as those belonging to the interrupt controller, chip-select module, UARTs, timers, DMA, and I^2C.

Table 6-1. SIM Registers

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x000	Reset status register (RSR) [p. 6-5]	System protection control register (SYPCR) [p. 6-8]	Software watchdog interrupt vector register (SWIVR) [p. 6-9]	Software watchdog service register (SWSR) [p. 6-9]
0x004	Pin assignment register (PAR) [p. 6-10]		Interrupt port assignment register (IRQPAR) [p. 9-7]	Reserved
0x008	PLL control (PLLCR) [p. 7-3]	Reserved		
0x00C	Default bus master park register (MPARK) [p. 6-11]	Reserved		
0x010–0x03C	Reserved			
Interrupt Controller Registers [p. 9-2]				
0x040	Interrupt pending register (IPR) [p. 9-6]			
0x044	Interrupt mask register (IMR) [p. 9-6]			
0x048	Reserved			Autovector register (AVR) [p. 9-5]
Interrupt Control Registers (ICRs) [p. 9-3]				

Table 6-1. SIM Registers (Continued)

MBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x04C	Software watchdog timer (ICR0) [p. 9-3]	Timer0 (ICR1) [p. 9-3]	Timer1 (ICR2) [p. 9-3]	I^2C (ICR3) [p. 9-3]
0x050	UART0 (ICR4) [p. 9-3]	UART1 (ICR5) [p. 9-3]	DMA0 (ICR6) [p. 9-3]	DMA1 (ICR7) [p. 9-3]
0x054	DMA2 (ICR8) [p. 9-3]	DMA3 (ICR9) [p. 9-3]	Reserved	

6.2.2 Module Base Address Register (MBAR)

The supervisor-level MBAR, Figure 6-2, specifies the base address and allowable access types for all internal peripherals. It is written with a MOVEC instruction using the CPU address 0xC0F. (See the *ColdFire Family Programmer's Reference Manual*.) MBAR can be read or written through the debug module as a read/write register, as described in Chapter 5, "Debug Support." Only the debug module can read MBAR.

The valid bit, MBAR[V], is cleared at system reset to prevent incorrect references before MBAR is written; other MBAR bits are uninitialized at reset. To access internal peripherals, write MBAR with the appropriate base address (BA) and set MBAR[V] after system reset.

All internal peripheral registers occupy a single relocatable memory block along 4-Kbyte boundaries. If MBAR[V] is set, MBAR[BA] is compared to the upper 20 bits of the full 32-bit internal address to determine if an internal peripheral is being accessed. MBAR masks specific address spaces using the address space fields. Attempts to access a masked address space generate an external bus access.

Addresses hitting overlapping memory spaces take the following priority:

1. MBAR
2. SRAM and caches
3. Chip select

NOTE:

The MBAR region must be mapped to non-cacheable space.

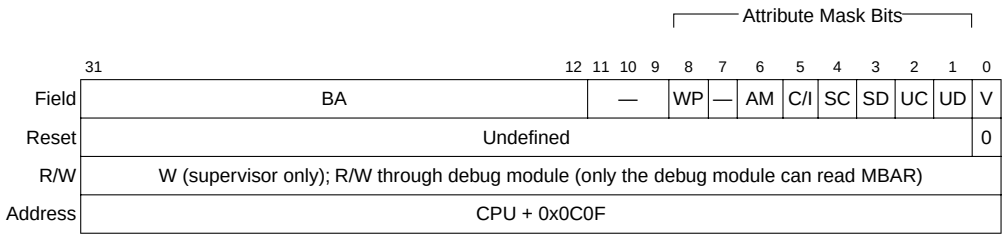

Figure 6-2. Module Base Address Register (MBAR)

Table 6-2 describes MBAR fields.

Table 6-2. MBAR Field Descriptions

Bits	Field	Description
31–12	BA	Base address. Defines the base address for a 4-Kbyte address range.
11–9	—	Reserved, should be cleared.
8	WP	Write protect. Mask bit for write cycles in the MBAR-mapped register address range. 0 Module address range is read/write. 1 Module address range is read only.
7	—	Reserved, should be cleared.
6	AM	Alternate master mask. When AM = 0 and an alternate master (external master or DMA) accesses MBAR-mapped registers, MBAR[SC,SD,UC,UD] are ignored in address decoding. These fields mask address space, placing the MBAR-mapped register in a specific address space or spaces.
5	C/I	Mask CPU space and interrupt acknowledge cycles. 0 Activates the corresponding MBAR-mapped register 1 Regular external bus access
4	SC	Setting masks supervisor code space in MBAR address range
3	SD	Setting masks supervisor data space in MBAR address range
2	UC	Setting masks user code space in MBAR address range
1	UD	Setting masks user data space in MBAR address range
0	V	Valid. Determines whether MBAR settings are valid. 0 MBAR contents are invalid. 1 MBAR contents are valid.

The following example shows how to set the MBAR to location 0x1000_0000 using the D0 register. Setting MBAR[V] validates the MBAR location. This example assumes all accesses are valid:

```
move.1 #0x10000001,DO
movec DO,MBAR
```

6.2.3 Reset Status Register (RSR)

The reset status register (RSR), Figure 6-3, contains two status bits, HRST and SWTR. Reset control logic sets one of the bits depending on whether the last reset was caused by an external device asserting $\overline{\text{RSTI}}$ (HRST = 1) or by the software watchdog timer (SWTR = 1). Only one RSR bit can be set at any time. If a reset occurs, reset control logic sets only the bit that indicates the cause of reset.

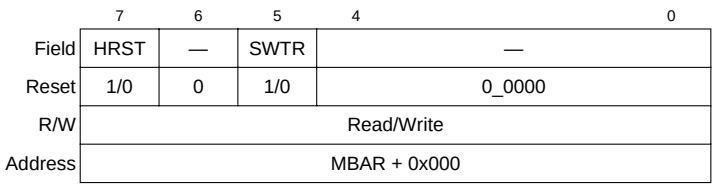

Figure 6-3. Reset Status Register (RSR)

Table 6-3 describes RSR fields.

Table 6-3. RSR Field Descriptions

Bits	Name	Description
7	HRST	Hardware or system reset 1 An external device driving $\overline{\text{RSTI}}$ caused the last reset. Assertion of reset by an external device causes the core processor to take a reset exception. All registers in internal peripherals and the SIM are reset.
6	—	Reserved, should be cleared.
5	SWTR	Software watchdog timer reset 1 The last reset was caused by the software watchdog timer. If SYPCR[SWRI] = 1 and the software watchdog timer times out, a hardware reset occurs.
4–0	—	Reserved, should be cleared.

6.2.4 Software Watchdog Timer

The software watchdog timer prevents system lockup should the software become trapped in loops with no controlled exit. The software watchdog timer can be enabled or disabled through SYPCR[SWE]. If enabled, the watchdog timer requires the periodic execution of a software watchdog servicing sequence. If this periodic servicing action does not occur, the timer times out, resulting in a watchdog timer $\overline{\text{IRQ}}$ or hardware reset with $\overline{\text{RSTO}}$ driven low, as programmed by SYPCR[SWRI].

If the timer times out and the software watchdog transfer acknowledge enable bit (SYPCR[SWTA]) is set, a watchdog timer $\overline{\text{IRQ}}$ is asserted. Note that the software watchdog timer IACK cycle cannot be autovectored.

If a software watchdog timer IACK cycle has not occurred after another timeout, SWT $\overline{\text{TA}}$ is asserted in an attempt to terminate the bus cycle and allow the IACK cycle to proceed. The setting of SYPCR[SWTAVAL] indicates that the watchdog timer $\overline{\text{TA}}$ was asserted. Figure 6-4 shows termination of a locked bus.

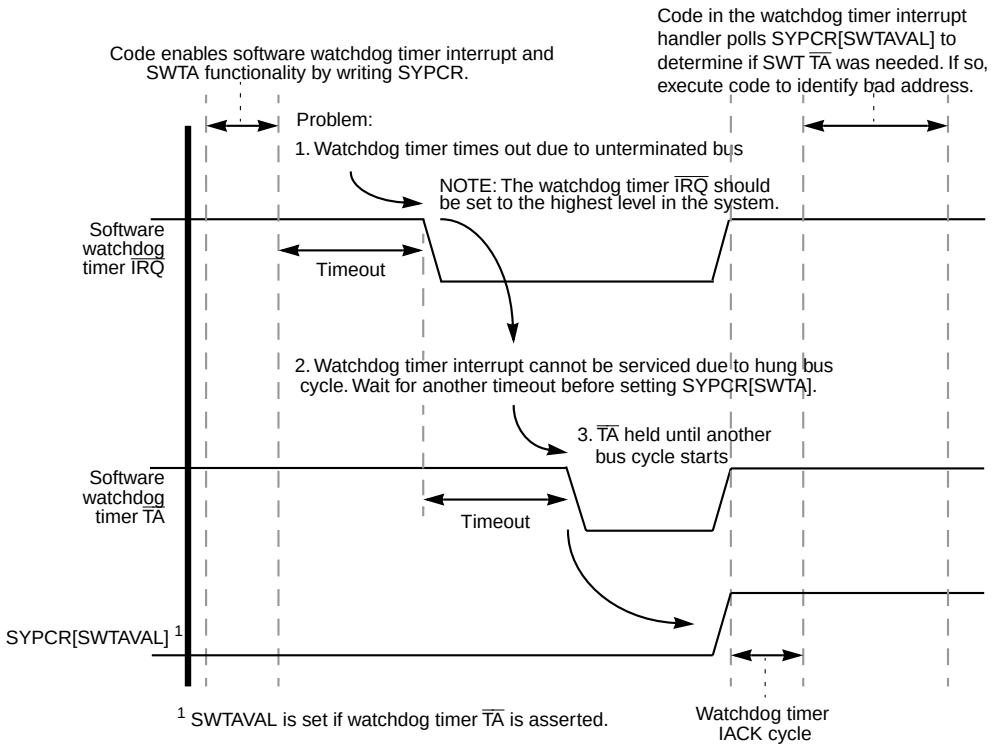

Figure 6-4. MCF5307 Embedded System Recovery from Unterminated Access

When the watchdog timer times out and SYPCR[SWRI] is programmed for a software reset, an internal reset is asserted and RSR[SWTR] is set.

To prevent the watchdog timer from interrupting or resetting, the SWSR must be serviced by performing the following sequence:

1. Write 0x55 to SWSR.
2. Write 0xAA to the SWSR.

Both writes must occur in order before the timeout, but any number of instructions or SWSR accesses can be executed between the two writes. This order allows interrupts and exceptions to occur, if necessary, between the two writes.

Caution should be exercised when changing SYPCR values after the software watchdog timer has been enabled with the setting of SYPCR[SWE], because it is difficult to determine the state of the watchdog timer while it is running. The countdown value is constantly compared with the timeout period specified by SYPCR[SWP,SWT]. Therefore, altering SWP and SWT improperly causes unpredictable processor behavior. The following steps must be taken to change SWP or SWT:

1. Disable the software watchdog timer by clearing SYPCR[SWE].
2. Reset the counter by writing 0x55 and then 0xAA to SWSR.
3. Update SYPCR[SWT,SWP].
4. Reenable the watchdog timer by setting SYPCR[SWE]. This can be done in step 3.

6.2.5 System Protection Control Register (SYPCR)

The SYPCR, Figure 6-5, controls the software watchdog timer, timeout periods, and software watchdog timer transfer acknowledge. The SYPCR can be read at any time, but can be written only if a software watchdog timer $\overline{IRQ}$ is not pending. At system reset, the software watchdog timer is disabled.

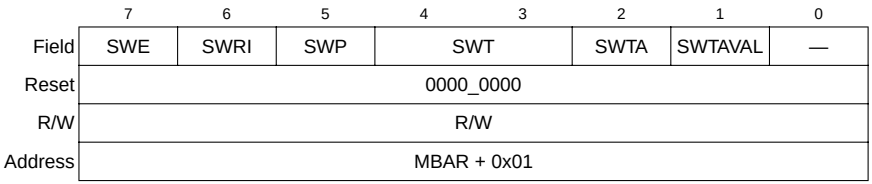

Figure 6-5. System Protection Control Register (SYPCR)

Table 6-4 describes SYPCR fields.

Table 6-4. SYPCR Field Descriptions

Bits	Name	Description
7	SWE	Software watchdog timer enable 0 Software watchdog timer disabled 1 Software watchdog timer enabled
6	SWRI	Software watchdog reset/interrupt select 0 If a timeout occurs, the watchdog timer generates an interrupt to the core processor at the level programmed into ICR0[IL]. 1 The software watchdog timer causes soft reset to be asserted for all modules of the part except for the PLL (reset mode selects, such as PP_RESET_SEL or chip-select settings, should not change).
5	SWP	Software watchdog prescaler. This bit interacts with SYPCR[SWT]. 0 Software watchdog timer clock not prescaled. 1 Software watchdog timer clock prescaled by 8192.
4–3	SWT	Software watchdog timing delay. SWT and SWP select the timeout period for the watchdog timer. At system reset, the software watchdog timer is set to the minimum timeout period. SWP = 0 / SWP = 1 00 2^9/system frequency / 00 2^{22}/system frequency 01 2^{11}/system frequency / 01 2^{24}/system frequency 10 2^{13}/system frequency / 10 2^{26}/system frequency 11 2^{15}/system frequency / 11 2^{28}/system frequency Note that if SWP and SWT are modified to select a new software timeout, the software service sequence must be performed (0x55 followed by 0xAA written to the SWSR) before the new timeout period takes effect.

Table 6-4. SYPCR Field Descriptions (Continued)

Bits	Name	Description
2	SWTA	Software watchdog transfer acknowledge enable 0 SWTA transfer acknowledge disabled 1 SWTA asserts transfer acknowledge enabled. After one timeout period of the unacknowledged assertion of the software watchdog timer interrupt, the software watchdog transfer acknowledge asserts, which allows the watchdog timer to terminate a bus cycle and allow the IACK to occur.
1	SWTAVAL	Software watchdog transfer acknowledge valid 0 SWTA transfer acknowledge has not occurred. 1 SWTA transfer acknowledge has occurred. Write a 1 to clear this flag bit.

6.2.6 Software Watchdog Interrupt Vector Register (SWIVR)

The SWIVR, shown in Figure 6-6, contains the 8-bit interrupt vector (SWIV) that the SIM returns during an interrupt-acknowledge cycle in response to a software watchdog timer-generated interrupt. SWIVR is set to the uninitialized vector 0x0F at system reset.

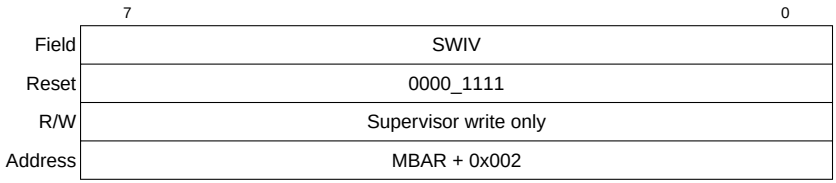

Figure 6-6. Software Watchdog Interrupt Vector Register (SWIVR)

Note that the software watchdog interrupt cannot be autovectored.

6.2.7 Software Watchdog Service Register (SWSR)

The SWSR, shown in Figure 6-7, is where the software watchdog timer servicing sequence should be written. To prevent a watchdog timer timeout, the software service sequence must be performed (0x55 followed by 0xAA written to the SWSR). Both writes must be performed in order before the timeout, but any number of instructions or accesses to the SWSR can be executed between the two writes. If the timer has timed out, writing to SWSR does not cancel the interrupt (that is, IPR[SWT] remains set). The interrupt is cancelled (and SWT is cleared) automatically when the IACK cycle is run.

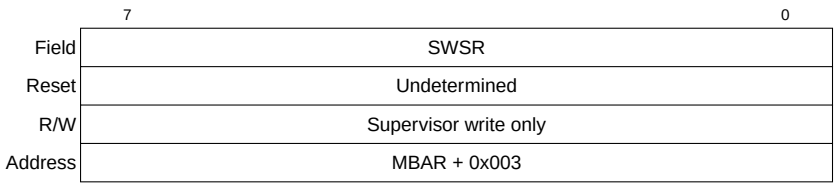

Figure 6-7. Software Watchdog Service Register (SWSR)

6.2.8 PLL Clock Control for CPU STOP Instruction

The SIM contains the PLL clock control register, which is described in detail in Section 7.2.4, “PLL Control Register (PLLCR).” PLLCR[ENBSTOP,PLLIPL] are significant to the operation of the SIM, and are described as follows:

- PLLCR[ENBSTOP] must be set for the ColdFire CPU STOP instruction to be acknowledged. This bit is cleared at reset and must be set for the MCF5307 to enter low-power modes. The CPU STOP instruction stops only clocks to the core processor. All internal modules remain clocked and can generate interrupts to restart the ColdFire core. For example, the on-chip timer can be used to interrupt the processor after a given timer countdown.
- PLLCR[PLLIPL] determines the minimum level at which an interrupt (decoded as an interrupt priority level or IPL) must occur to awaken the PLL. The PLL then turns clocks back on to the core processor and interrupt exception processing takes place. Table 6-5 describes PLLIPL settings to be compared against the interrupt ranges that awaken the core processor from a CPU STOP instruction.

Table 6-5. PLLIPL Settings

PLLIPL	Description
000	Any interrupts can wake core
001	Interrupts 2–7
010	Interrupts 3–7
011	Interrupts 4–7
100	Interrupts 5–7
101	Interrupts 6–7
110	Interrupt 7 only
111	No interrupts can wake core

6.2.9 Pin Assignment Register (PAR)

The pin assignment register (PAR), Figure 6-8, allows the selection of pin assignments.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	PAR15	PAR14	PAR13	PAR12	PAR11	PAR10	PAR9	PAR8	PAR7	PAR6	PAR5	PAR4	PAR3	PAR2	PAR1	PAR0
PARn = 0	PP15	PP14	PP13	PP12	PP11	PP10	PP9	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
PARn = 1	A31	A30	A29	A28	A27	A26	A25	A24	$\overline{\text{TIP}}$	$\overline{\text{DREQ0}}$	$\overline{\text{DREQ1}}$	TM2	TM1	TM0	TT1	TT0
Reset	Determined by driving D4/ADDR_CONFIG with a 1 or 0 when $\overline{\text{RSTI}}$ negates. The system is configured as PP[15:0] if D4 is low; otherwise alternate pin functions selected by PAR = 1 are used.															
R/W	R/W															
Address	Address MBAR + 0x004															

Figure 6-8. Pin Assignment Register (PAR)

6.2.10 Bus Arbitration Control

This section describes the bus arbitration register and the four arbitration schemes.

6.2.10.1 Default Bus Master Park Register (MPARK)

The MPARK, shown in Figure 6-9, determines the default bus master arbitration between internal transfers (core and DMA module) and between internal and external transfers to internal resources. This arbitration is needed because external masters can access internal registers within the MCF5307 peripherals.

	7	6	5	4	3	2	0
Field	PARK		IARBCTRL	EARBCTRL	SHOWDATA	—	BCR24BIT
Reset	0000_0000						
R/W	R/W						
Address	MBAR + 0x0C						

Figure 6-9. Default Bus Master Register (MPARK)

Table 6-6 describes MPARK bits.

Table 6-6. MPARK Field Descriptions

Bits	Name	Description
7–6	PARK	Park. Indicates the arbitration priority of internal transfers among MCF5307 resources. 00 Round-robin between DMA and ColdFire core 01 Park on master ColdFire core 10 Park on master DMA module 11 Park on current master Use of this field is described in detail in Section 6.2.10.1.1, “Arbitration for Internally Generated Transfers (MPARK[PARK]).”
5	IARBCTRL	Internal bus arbitration control. Controls external device access to the MCF5307 internal bus. 0 Arbitration disabled (single-master system) 1 Arbitration enabled. IARBCTRL must be set if external masters are using internal resources like the DRAM controller or chip selects. Use of this bit depends on whether the system has single or multiple masters, as follows: • In a single-master system, IARBCTRL should stay cleared, disabling internal arbitration by external masters. In this scenario, MPARK[PARK] applies only to priority of internal masters over one another. Note that the internal DMA (master 3) has priority over the ColdFire core (master 2), if internal DMA bandwidth is at its maximum (BWC = 000). • In multiple master systems that expect to use internal resources like the DRAM controller or chip selects, internal arbitration should be enabled. The external master defaults to the highest priority internal master anytime $\overline{BG}$ is negated.
4	EARBCTRL	External bus arbitration control. Enables internal register memory space to external bus arbitration. Internal registers are those accessed at offsets to the MBAR. These include the SIM, DMA, chip selects, timers, UARTs, I^2C, and parallel port registers. These registers do not include the MBAR; only the core can access the MBAR. 0 Arbitration disabled 1 Arbitration enabled The use of this field is described in detail in Section 6.2.10.1.2, “Arbitration between Internal and External Masters for Accessing Internal Resources.”

Table 6-6. MPARK Field Descriptions (Continued)

Bits	Name	Description
3	SHOWDATA	Enable internal register data bus to be driven on external bus. EARBCTRL must be set for this function to work. Section 6.2.10.1.2, “Arbitration between Internal and External Masters for Accessing Internal Resources,” describes the proper use of SHOWDATA. 0 Do not drive internal register data bus values to external bus. 1 Drive internal register data bus values to external bus.
2–1	—	Reserved, should be cleared.
0	BCR24BIT	Controls the BCR and address mapping for DMA. Allows the BCR to be used as a 24-bit register. Chapter 12, “DMA Controller Module,” describes the BCRs. 0 DMA BCRs function as 16-bit counters. 1 DMA BCRs function as 24-bit counters.

6.2.10.1.1 Arbitration for Internally Generated Transfers (MPARK[PARK])

MPARK[PARK] prioritizes internal transfers, which can be initiated by the core and the on-chip DMA module, which contains all four DMA channels. Priority among the four DMA channels in the module is determined by the BWC bits in their respective DMA control registers (see Chapter 12, “DMA Controller Module”).

The four arbitration schemes for internally generated transfers are described as follows:

- Round-robin scheme (PARK = 00)—Figure 6-10 shows round-robin arbitration between the core and DMA module. Bus mastership alternates between the core and DMA module.

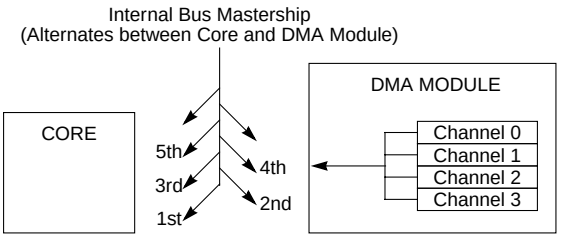

Figure 6-10. Round Robin Arbitration (PARK = 00)

The DMA module presents only the highest-priority DMA request, and bus mastership alternates between the core and DMA channel as long as both are requesting bus mastership. Section 12.5.4.1, “External Request and Acknowledge Operation,” includes a timing diagram showing a lower-priority DMA transfer.

When the processor is initialized, the core has first priority. If DMA channels 0 and 1 (both set to BWC = 010) assert an internal bus request during a core-generated bus transfer, DMA channel 0 would gain bus mastership next. However, if the core requests the bus during this DMA transfer, bus mastership returns to the core rather than being granted to DMA channel 1.

Note that the internal DMA has higher priority than the core if the internal DMA has its bandwidth BWC bits set to 000 (maximum bandwidth).

- Park on master core priority (PARK = 01)—The core retains bus mastership as long as it needs it. After it negates its internal bus request, the core does not have to rearbitrate for the bus unless the DMA module has requested the bus when it is idle. The DMA module can be granted bus mastership only when the core is not asserting its bus request. See Figure 6-11.

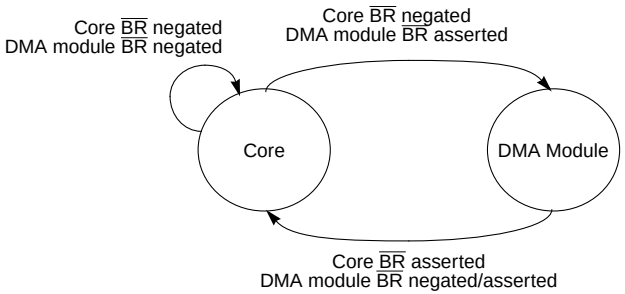

Figure 6-11. Park on Master Core Priority (PARK = 01)

- Park on master DMA priority (PARK = 10)—The DMA module retains bus mastership as long as it needs it. After it negates its internal bus request, the DMA module does not have to rearbitrate for the bus unless the core has requested the bus when it is idle. The core can be granted bus mastership only when the DMA module is not asserting its bus request. See Figure 6-12.

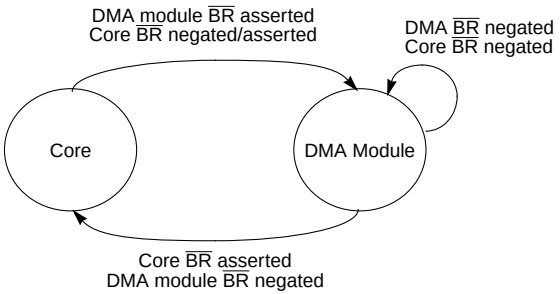

Figure 6-12. Park on DMA Module Priority (PARK = 10)

- Park on current master priority (PARK = 11)—The current bus master retains mastership as long as it needs the bus. The other device can become the bus master only when the bus is idle. For example, if the core is bus master out of reset, it retains mastership as long as it needs the bus. It loses mastership only when it negates its bus request signal and the DMA asserts its internal bus request signal. At this point the DMA module is the bus master, and retains bus mastership as long as it needs the bus. See Figure 6-13.

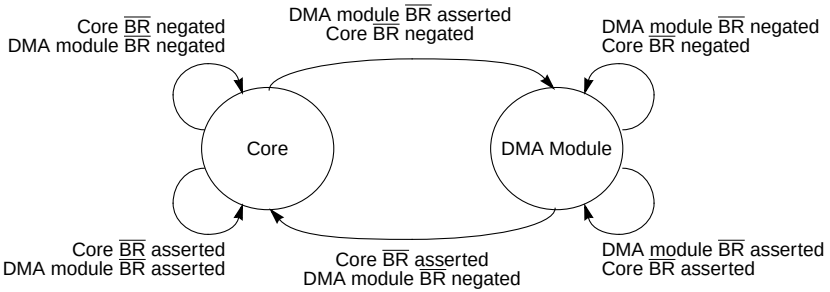

Figure 6-13. Park on Current Master Priority (PARK = 01)

6.2.10.1.2 Arbitration between Internal and External Masters for Accessing Internal Resources

If an external device is programmed to access internal MCF5307 resources (EARBCTRL = 1), the external device can gain bus mastership only when $\overline{BG}$ is negated. This means neither the core nor the DMA controller can access the external bus until the external device asserts $\overline{BG}$. After the external master finishes its bus transfer and asserts $\overline{BG}$, the core has priority on the next available bus cycle regardless of the value of PARK. Thus if the core asserts its internal bus request on this first bus cycle, it executes a bus cycle even if PARK indicates the DMA should have priority. Then, after the bus transfer, the PARK scheme returns to programmed functioning and the DMA is given bus mastership.

NOTE:

In all arbitration modes, if $\overline{BG}$ is negated, the external master interface has highest priority. In this case, the ColdFire core has second-highest priority, until the internal bus grant is asserted.

- In a single-master system, the setting of EARBCTRL does not affect arbitration performance. Typically, $\overline{BG}$ is tied low and the MCF5307 always owns the external bus and internal register transfers are already shown on the external bus. In a system where MCF5307 is the only master, this bit may remain cleared.

  If the system needs external visibility of the data bus values during internal register transfers for system debugging, both EARBCTRL and SHOWDATA must be set.

  Note that when an internal register transfer is driven externally, $\overline{TA}$ becomes an output, which is asserted (normally an input) to prevent external devices and

memories from responding to internal register transfers that go to the external bus. The $\overline{AS}$ signal and all chip-select-related strobe signals are not asserted.

Do not immediately follow a cycle in which SHOWDATA is set with a cycle using fast termination.

- In multiple-master systems, disabling arbitration with EARBCTRL allows performance improvement because internal register bus transfer cycles do not interfere with the external bus.

  Having internal transfers go external may affect performance in two ways:

  — If the internal device does not control the bus immediately, the core stalls until it wins arbitration of the external bus.

  — If the core wins arbitration instantly, it may kick the external master off of the external bus unnecessarily for a transfer that did not need the external bus. For debug, where this performance penalty is not a concern, setting EARBCTRL and SHOWDATA provides external visibility of the internal bus cycles.