Chapter 4

Peripherals

In addition to on-chip memory, the TMS320C62x and TMS320C67x devices also contain peripherals for communication with off-chip memory, coprocessors, host processors, and serial devices. All of these peripherals are briefly described here, but each ’C6x device has only a specific subset of them. See the data sheet for your particular device to determine the peripherals present.

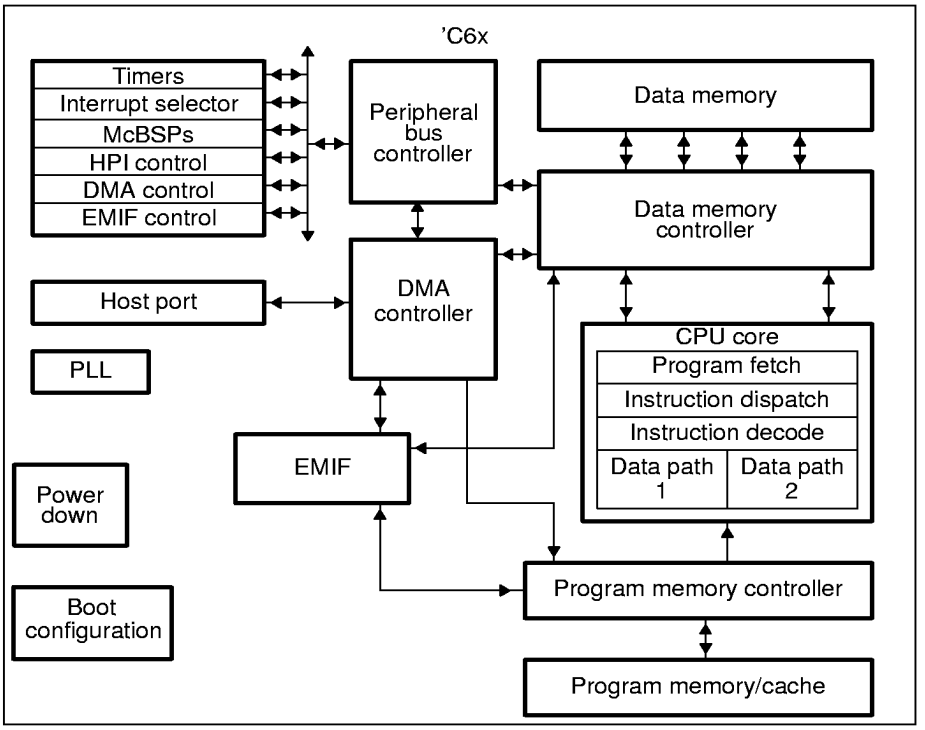
Topic **Page**

Peripherals for the ’C6x devices include:

- Direct memory access (DMA) controller
- Host-port interface (HPI)
- External memory interface (EMIF)
- Boot configuration
- Multichannel buffered serial ports (McBSPs)
- Interrupt Selector
- 32-bit timers
- Power-down logic

Figure 4–1 shows the block diagram for peripherals for the ’C6x devices.

Figure 4–1. TMS320C6x Block Diagram



4.1 Direct Memory Access (DMA)

The direct memory access (DMA) controller transfers data between regions in the memory map without intervention by the CPU. The DMA allows movement of data to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation. The DMA has four independently programmable channels allowing four different contexts for DMA operation. In addition a fifth (auxiliary) channel allows the DMA to service requests from the host-port interface (HPI). In discussing DMA operations several terms are important:

- **Read transfer:** The DMA reads the data element from a source location in memory.
- **Write transfer:** The DMA writes the data element that was read during a read transfer to its destination location in memory.
- **Element transfer:** The combined read and write transfer for a single data element.
- **Frame transfer:** Each DMA channel has an independently programmable number of elements per frame. In completing a frame transfer, the DMA moves all elements in a single frame.
- **Block transfer:** Each DMA channel also has an independently programmable number of frames per block. In completing a block transfer, the DMA moves all frames it has been programmed to move.

The DMA has the following features:

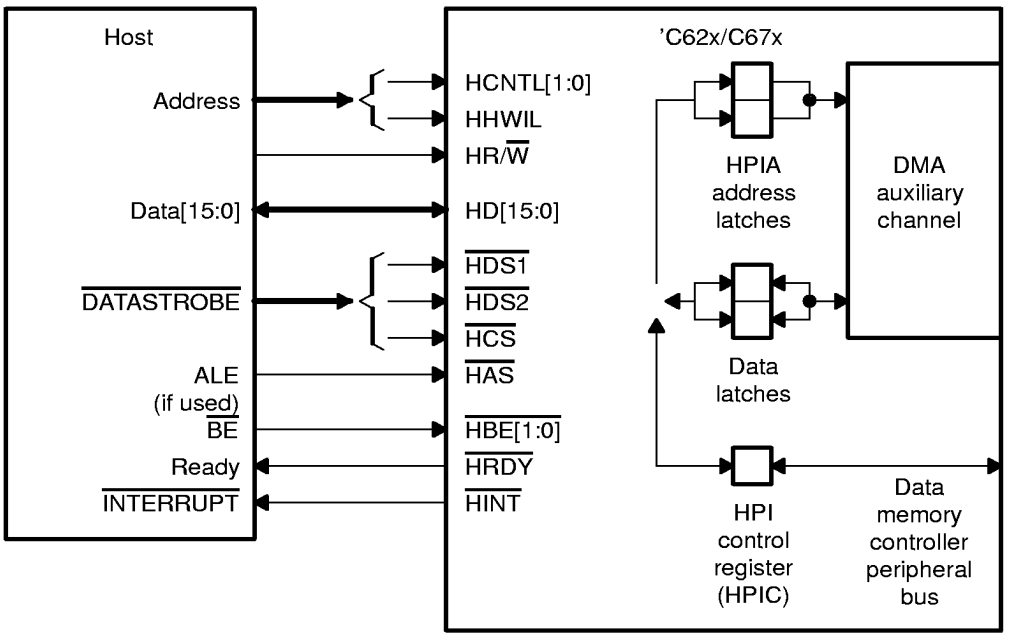
- **Background operation:** The DMA operates independently of the CPU.
- **High throughput:** Elements can be transferred at the CPU clock rate.
- **Four channels:** The DMA can keep track of the contexts of four independent block transfers.
- **Auxiliary channel:** This channel allows the host port to make requests into the CPU's memory space. This chapter discusses how the auxiliary channel requests are prioritized relative to other channels and the CPU. Detailed explanation of how it is used in conjunction with a peripheral is found in that peripheral's documentation.
- **Split operation:** A single channel may be used to simultaneously perform both the receive and transmit element transfers to or from two peripherals and memory, effectively acting like two DMAs.
- **Multiframe transfer:** Each block transfer can consist of multiple frames of a programmable size.

- **Programmable priority:** Each channel has independently programmable priorities versus the CPU.
- **Programmable address generation:** Each channel's source and destination address registers can have configurable indexes for each read and write transfer. The address may remain constant, increment, decrement, or be adjusted by a programmable value. The programmable value allows a distinct index for the last transfer in a frame and for the preceding transfers. This feature is used for multichannel sorting.
- **Full-address 32-bit address range:** The DMA can access any region in the memory map:
  - The on-chip data memory.
  - The on-chip program memory when mapped into memory space rather than being utilized as cache.
  - The on-chip peripherals.
  - The external memory interface (EMIF).
- **Programmable width transfers:** Each channel can be independently configured to transfer either bytes, 16-bit halfwords, or 32-bit words.
- **Autoinitialization:** Once a block transfer is complete, a DMA channel may automatically reinitialize itself for the next block transfer.
- **Event synchronization:** Each read, write, or frame transfer may be initiated by selected events.
- **Interrupt generation:** On completion of each frame transfer or of an entire block transfer, as well as on various error conditions, each DMA channel may send an interrupt to the CPU.

4.2 Host-port Interface (HPI)

The host-port interface (HPI) is a 16-bit-wide parallel port through which a host processor can directly access the CPU's memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals. Connectivity to the CPU's memory space is provided through the DMA controller. Dedicated address and data registers not accessible to the CPU connect the HPI to the DMA auxiliary channel, which connects the HPI to the CPU's memory space. The HPI and the CPU can access the HPI control register (HPIC). The host can access the host address register (HPIA) and the host data register (HPID) as well, as the HPIC, using the external data and interface control signals. Figure 4–2 is a simplified diagram of the interface between the host and the 'C62x/C67x HPI.

Figure 4–2. Host-port Interface (HPI) Block Diagram



The HPI provides 32-bit data to the CPU with an economical 16-bit external interface by automatically combining successive 16-bit transfers. When the host device transfers data through HPID, the DMA auxiliary channel accesses the CPU's address space. The HPI supports high speed consecutive host accesses.

The 16-bit data bus, HD[15:0], exchanges information with the host. Because of the 32-bit word structure of the chip architecture, all transfers with a host consist of two consecutive 16-bit halfwords. On host data (HPID) write accesses, the $\overline{\text{HBE}}$[1:0] byte enables select the bytes in a 32-bit word to be written. For HPIA, HPIC, and HPID read accesses the byte enables are not used. The dedicated HHWIL pin indicates whether the first or second halfword is being transferred. An internal control register bit determines whether the first or second halfword is placed into the most significant halfword of a word.

The two data strobes ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$), the read/write select (HR/$\overline{\text{W}}$), and the address strobe ($\overline{\text{HAS}}$) enable the HPI to interface to a variety of industry-standard host devices with little or no additional logic required. The HPI can easily interface to hosts with multiplexed or dedicated address/data bus, a data strobe and a read/write strobe, or two separate strobes for read and write.

The host can access HPID with an optional automatic address increment of HPIA. This feature facilitates reading or writing to sequential word locations.

The HPI ready pin ($\overline{\text{HRDY}}$) allows insertion of host wait states. Wait states may be necessary depending on the latency of the memory accessed via the HPI, as well as on the rate of host access.

For more information on the host-port interface, see the *TMS320C6201/C6701 Peripherals Reference Guide*.

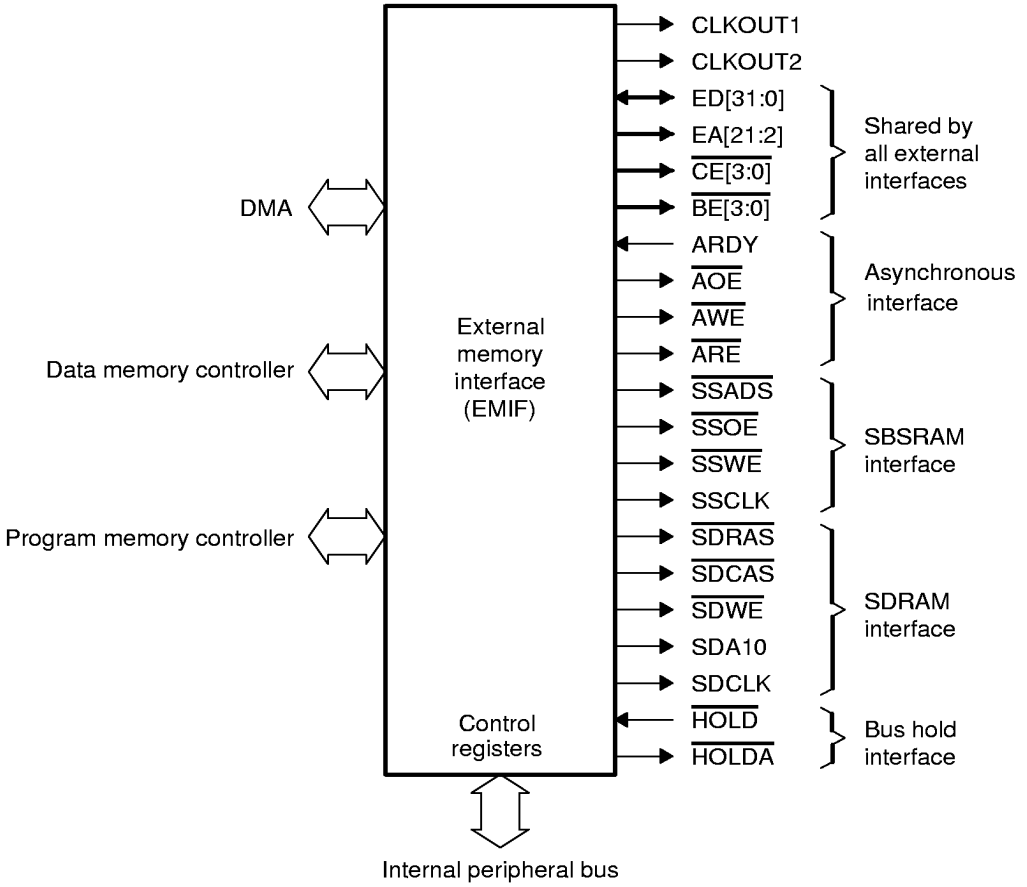
4.3 External Memory Interface (EMIF)

The external memory interface (EMIF) supports a glueless interface to several external devices, including:

- Synchronous burst SRAM (SBSRAM) running at the full rate and at half the rate of the CPU clock
- Synchronous DRAM (SDRAM) running at half the CPU clock rate
- Asynchronous devices, including asynchronous SRAM, ROM, and FIFOs. The EMIF provides highly programmable timing to these interfaces.
- An external shared-memory device

A block diagram of the EMIF is shown in Figure 4–3.

Figure 4–3. External Memory Interface (EMIF) Block Diagram



4.3.1 EMIF Registers

Control of the EMIF and the memory interfaces it supports is maintained through a set of memory-mapped registers within the EMIF. A write to any EMIF register will not finish until all pending EMIF accesses that use the register have finished. The memory-mapped registers are shown in Table 4–1.

Table 4–1. External Memory Interface (EMIF) Memory-Mapped Registers

Byte Address	Name
0180 0000h	EMIF global control
0180 0004h	EMIF $\overline{\text{CE1}}$ space control
0180 0008h	EMIF $\overline{\text{CE0}}$ space control
0180 000Ch	Reserved
0180 0010h	EMIF $\overline{\text{CE2}}$ space control
0180 0014h	EMIF $\overline{\text{CE3}}$ space control
0180 0018h	EMIF SDRAM control
0180 001Ch	EMIF SDRAM refresh period

The four $\overline{\text{CE}}$ space control registers correspond to the four $\overline{\text{CE}}$ spaces supported by the EMIF. The MTYPE field in each register identifies the memory type for the corresponding $\overline{\text{CE}}$ space. If MTYPE selects SDRAM or SBSRAM, the remaining fields in the register do not apply.

The SDRAM control register controls SDRAM parameters for all $\overline{\text{CE}}$ spaces that specify an SDRAM memory.

The SDRAM refresh period register controls the refresh period for SDRAM in terms of CLKOUT2 cycles (half the CPU clock rate). Optionally, the refresh period register can send an interrupt to the CPU. Thus, this register can be used as a general-purpose timer if SDRAM is not used by the system. The counter value can be read by the CPU.

4.3.2 SDRAM Interface

The EMIF supports 2-bank 16M-bit and 4-bank 64M-bit SDRAM, which offers system designers an interface to high-speed and high-density memory. Figure 4–4 illustrates the EMIF to SDRAM interface. Table 4–2 lists the suported memory configurations. The EA pins starting from pin13 connect to the SDRAM address pins starting at pin 11. The symbol m is 0 for a 16M bit interface and 2 for 64M bit interface.

Figure 4–4. EMIF to SDRAM Interface

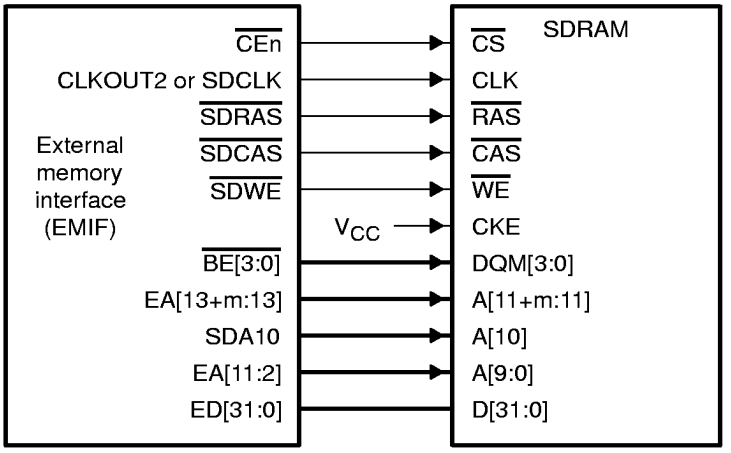


Table 4–2. SDRAM Memory Population

SDRAM Size	SDRAM Banks	SDRAM Width	Devices per $\overline{CE}$ Space	Memory Size per $\overline{CE}$ Space
16M bits	2	16 bits	2	4M bytes
16M bits	2	8 bits	4	8M bytes
64M bits	4	16 bits	2	16M bytes

During an SDRAM read, the selected bank is activated with the row address during the ACTV command. The EMIF uses a $\overline{CAS}$ latency of 3 and a burst length of 1. During read cycles, the 3-cycle latency causes data to appear three cycles after the corresponding column address.

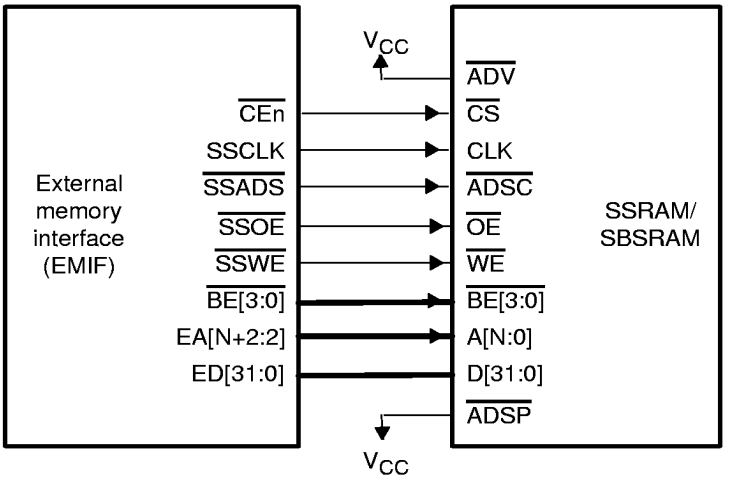
All SDRAM writes have a burst length of 1. The bank is activated with the row address during the ACTV command. There is no latency associated with writes, so data is output on the same cycle that the column address is received. Writes to invalid bytes are disabled via the appropriate DQM inputs. This feature allows for byte and halfword writes.

4.3.3 SBSRAM Interface

The EMIF interfaces directly to industry-standard synchronous burst SRAMs, see Figure 4–5. This memory interface allows a high-speed memory interface without some of the limitations of SDRAM. Since SBSRAM are SRAM devices, random accesses are possible during burst reads or writes. The SBSRAM interface can run at either the CPU clock speed or at half of this rate.

The four SBSRAM control pins are latched by the SBSRAM on the rising SSCLK edge to determine the current operation. These signals are valid only if the chip select line for the SBSRAM is low.

Figure 4–5. EMIF to SBSRAM Interface



4.3.4 Asynchronous Interface

The asynchronous interface offers configurable cycle types, which can be used to interface to a variety of memory and peripheral types, including SRAM, EPROM, and Flash memory, as well as FPGA and ASIC devices.

The following three figures show interfaces to SRAM (Figure 4–6), to FIFOs (Figure 4–7), and to ROM (Figure 4–8). Although ROM can be interfaced at any of the $\overline{CE}$ spaces, it is often used at $\overline{CE1}$, because only that space can be configured for widths smaller than 32 bits.

Figure 4–6. EMIF to SRAM Interface

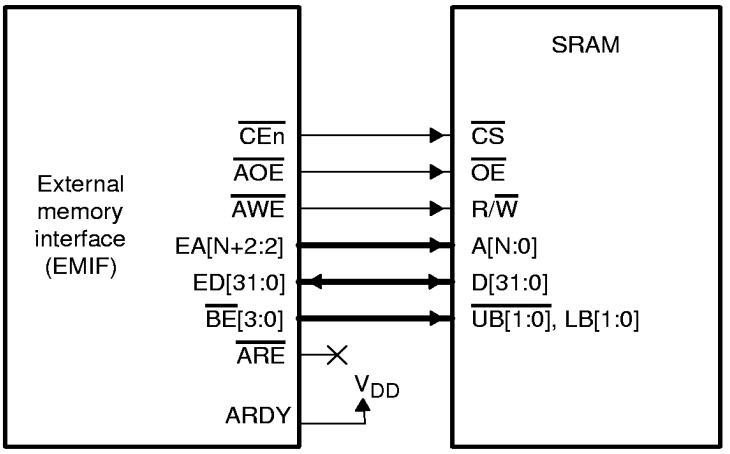


Figure 4–7. EMIF to FIFO Interface

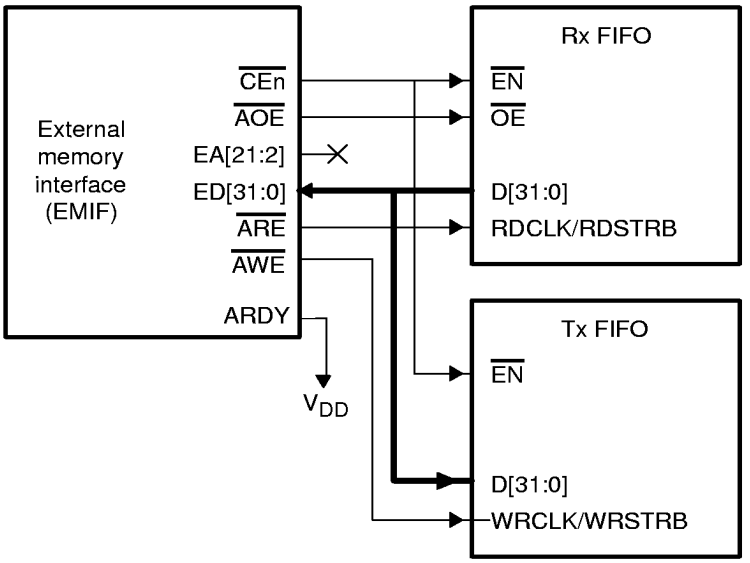
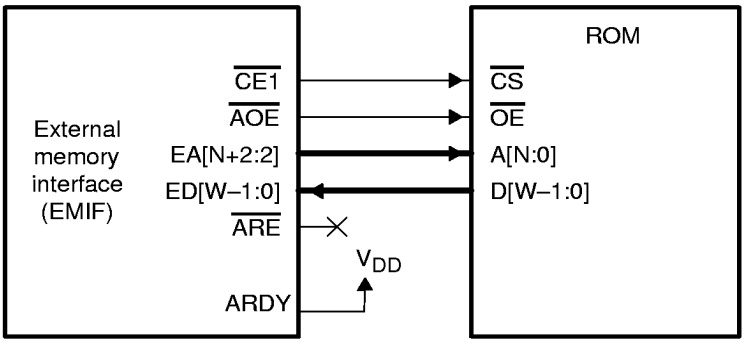


Figure 4–8. EMIF to ROM Interface



The EMIF supports 8-, 16-, and 32-bit-wide ROMs. In Figure 4–8 the W in ED to D lines denotes the number of data bits of the ROM. The access modes are selected by the MTYPE field in the EMIF $\overline{CE}$ space control register. In reading data from these narrow-width memory spaces, the EMIF packs multiple reads into one 32-bit-wide value. This mode is primarily intended for word access to 8-bit and 16-bit ROM devices. Thus, the following restrictions apply:

- Read operations always read 32 bits, regardless of the access size or the memory width.
- The address is shifted up appropriately to provide the correct address to the narrow memory. The shift amount is 1 for 16-bit ROM and 2 for 8-bit ROM. Thus, the high address bits are shifted out and accesses wrap around if that $\overline{CE}$ space spans the entire EA bus.
- The EMIF always reads the lower addresses first and packs these into the LSBytes. The EMIF packs subsequent accesses into the higher order bytes. Thus, the expected packing format in ROM is always little-endian regardless of the value of the LENDIAN bit.

In 8-bit ROM mode, the address is left shifted by 2 to create a byte address on EA to access byte-wide ROM. The EMIF always packs four consecutive bytes aligned on a 4-byte (byte address = 4N) boundary into a word access. The bytes are fetched in the address order: 4N, 4N + 1, 4N + 2, 4N + 3. Bytes are packed into the 32-bit word in the following little-endian order, from MSByte to LSByte: 4N + 3, 4N + 2, 4N + 1, 4N.

In 16-bit ROM mode, the address is left shifted by 1 to create a halfword address on EA to access 16-bit-wide ROM. The EMIF always packs two consecutive halfwords aligned on a 4-byte (byte address = 4N) boundary into a word access. The halfwords are fetched in the address order: 4N, 4N + 2. Halfwords are packed into the 32-bit word in the following little-endian order, from most significant halfword to least significant halfword: 4N + 2, 4N.

4.4 Boot Configuration Logic

The ’C62x and ’C67x provide a variety of boot configurations for proper device initialization. These configurations determine what actions the ’C62x/C67x performs after device reset to prepare for initialization. These boot configurations, which are set by external input pins, determine:

- The memory map the device selects. The memory map determines whether internal or external memory is mapped at address 0.
- The type of external memory at address 0 (if external memory is mapped at address 0)
- The boot process used to initialize the memory at address 0 before the CPU is allowed to run

4.4.1 Device Reset

The external device reset is the active-low $\overline{\text{RESET}}$ signal. While $\overline{\text{RESET}}$ is low, the device is held in reset. During this period the device is initialized to the prescribed reset state. All 3-state outputs are placed into the high-impedance state. All other outputs are returned to their default state. $\overline{\text{RESET}}$ is latched with the device CLKIN signal, as well as with the CPU clock. Thus, $\overline{\text{RESET}}$ has minimum low time in terms of CLKIN as well as CPU clock (CLKOUT1) cycles. The precise timing requirements for device reset are described in the data sheet for each particular device. The rising edge of $\overline{\text{RESET}}$ starts the processor running with the prescribed boot configuration.

4.4.2 Boot Configuration

External pins BOOTMODE[4:0] determine the boot configuration. The values of BOOTMODE[4:0] are latched with the rising edge of $\overline{\text{RESET}}$. These signals must be valid for the proper setup and hold times. See the data sheet for specific timing requirements.

Three types of boot processes are available:

- **No boot process(direct-execution startup):** The CPU simply starts running from the memory located at address 0. When this memory location resides on SDRAM, the CPU is held until SDRAM initialization finishes.
- **ROM boot process:** The memory located at the beginning of external memory space $\overline{CE1}$ (see Figure 3–1 on page 3-2) is copied to address 0 by the DMA controller. Although the boot process begins when the device is released from external reset, this transfer occurs while the CPU is held in reset internally. The amount of memory copied is 16K words of 32 bits each. This process lets you choose the width of the ROM. In this case, the EMIF can automatically assemble consecutive 8-bit bytes or 16-bit halfwords to form the 32-bit instruction words to be moved. These values are expected to be stored in little-endian format in the external memory, which is typically a ROM device.
- **HPI boot process:** In the HPI (host-port interface) boot process, the CPU is held in reset while the remainder of the device is released from reset. During this period, an external host can initialize the CPU's memory space as necessary through the HPI, including external memory configuration registers. Once the necessary external memory has been configured, the host can access any external sections it needs to complete initialization. Once the host is finished with all necessary initialization, the host writes a 1 to the DSPINT bit in the HPI control register (HPIC). This write causes an active transition on the DSPINT signal. In turn, this transition causes the boot configuration logic to remove the CPU from its reset state. The CPU then begins running from address 0. The DSPINT condition is not latched by the CPU, because it occurs while the CPU is still in reset. Also, DSPINT wakes up the CPU from internal reset only if the HPI boot process is selected.

4.5 Multichannel Buffered Serial Port (McBSP)

The ’C62x/C67x multichannel buffered serial port (McBSP) is based on the standard serial port interface found on the TMS320C2x, ’C2xx, ’C5x, and ’C54x devices. The standard serial port interface provides:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- External shift clock generation or an internal programmable frequency shift clock

In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP and ST–BUS compliant devices
  - IOM-2 compliant devices
  - AC97 compliant devices
  - IIS compliant devices
  - SPI™ devices
- Multichannel transmit and receive of up to 128 channels.
- A wider selection of data sizes including 8, 12, 16, 20, 24, or 32 bits

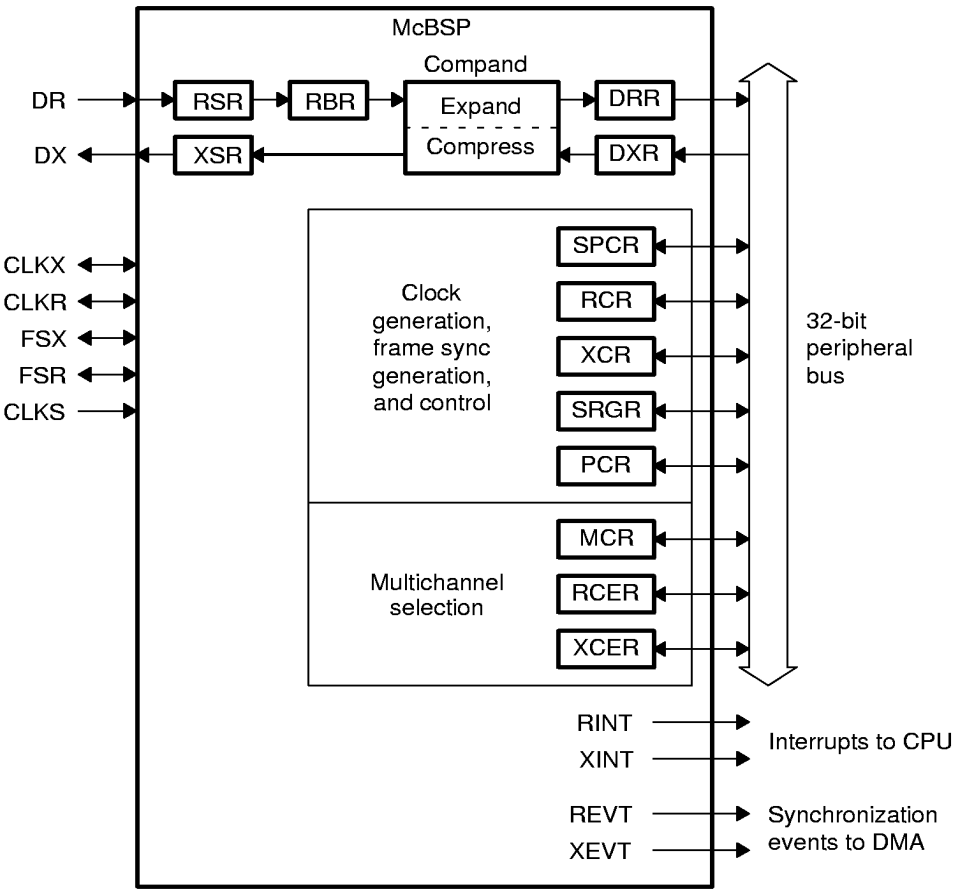
Note:

Data sizes are referred to as *word* or *serial word* throughout this document. Therefore, when *word* is used, it can be 8, 12, 16, 20, 24, or 32 bits, in contrast to the true definition of word as being 32 bits.

- μ-Law and A-Law companding
- 8-bit data transfers with LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

The McBSP consists of a data path and control path. Seven pins connect the control and data paths to external devices as shown in Figure 4–9.

Figure 4–9. Multichannel Buffered Serial Port (McBSP) Internal Block Diagram



Data is communicated to devices interfacing to the McBSP via the data transmit (DX) pin for transmit and the data receive (DR) pin for receive. Control information in the form of clocking and frame synchronization is communicated via CLKX, CLKR, FSX, and FSR. The peripheral device communicates to the McBSP via 32-bit-wide control registers accessible via the internal peripheral bus. The CPU or DMA controller reads the received data from the data receive register (DRR) and writes the data to be transmitted to the data transmit register (DXR). Data written to the DXR is shifted out to DX via the transmit shift register (XSR). Similarly, receive data on the DR pin is shifted into the receive shift register (RSR) and copied into the receive buffer register (RBR). RBR is then copied to DRR, which can be read by the CPU or the DMA controller. This

allows internal data movement and external data communications simultaneously. The remaining registers accessible to the CPU configure the control mechanism of the McBSP. These registers are listed in Table 4–3. The control block consists of internal clock generation, frame synchronization signal generation, control for both of these, and multichannel selection. This control block sends notification of important events to the CPU and the DMA controller via four signals as shown in Table 4–4.

Table 4–3. Multichannel Buffered Serial Port (McBSP) Registers

Abbreviation	Register Name
RBR	McBSP receive buffer register
RSR	McBSP receive shift register
XSR	McBSP transmit shift register
DRR	McBSP data receive register
DXR	McBSP data transmit register
SPCR	McBSP serial port control register
RCR	McBSP receive control register
XCR	McBSP transmit control register
SRGR	McBSP sample rate generator register
MCR	McBSP multichannel register
RCER	McBSP receive channel enable register
XCER	McBSP transmit channel enable register
PCR	McBSP pin control register

Table 4–4. Multichannel Buffered Serial Port (McBSP) CPU Interrupts and DMA Event Synchronization

Interrupt Name	Description
RINT	Receive interrupt to CPU
XINT	Transmit Interrupt to CPU
REVT	Receive synchronization event to DMA controller
XEVT	Transmit synchronization event to DMA controller

4.6 Timers

The ’C62x/C67x has two 32-bit general-purpose timers that you can use to:

- Time events
- Count events
- Generate pulses
- Interrupt the CPU
- Send synchronization events to the DMA controller

The timer has two signaling modes and can be clocked by an internal or an external source. The timer has an input pin (TINP) and an output pin (TOUT). The TINP can be used as a general-purpose input, and the TOUT pin can be used for a general-purpose output.

With an internal clock, the timer can signal an external A/D converter to start a conversion, or it can trigger the DMA controller to begin a data transfer. With an external clock, for example, the timer can count external events and interrupt the CPU after a specified number of events.

4.7 Interrupt Selector

The ’C62x/C67x peripheral set produces 16 interrupt sources. The CPU has 12 interrupts available for use. The interrupt selector allows you to choose which 12 of the 16 your system needs to use. The interrupt selector also allows you to effectively change the polarity of external interrupt inputs.

Table 4–5 lists the available interrupts.

Table 4–5. Peripheral Interrupts

Interrupt Selection Number	Interrupt Abbreviation	Interrupt Description
0000b	DSPINT	Host port host to DSP interrupt
0001b	TINT0	Timer 0 interrupt
0010b	TINT1	Timer 1 interrupt
0011b	SD_INT	EMIF SDRAM timer interrupt
0100b	EXT_INT4	External interrupt pin 4
0101b	EXT_INT5	External interrupt pin 5
0110b	EXT_INT6	External interrupt pin 6
0111b	EXT_INT7	External interrupt pin 7
1000b	DMA_INT0	DMA channel 0 interrupt
1001b	DMA_INT1	DMA channel 1 interrupt
1010b	DMA_INT2	DMA channel 2 interrupt
1011b	DMA_INT3	DMA channel 3 interrupt
1100b	XINT0	McBSP 0 transmit interrupt
1101b	RINT0	McBSP 0 receive Interrupt
1110b	XINT1	McBSP 1 transmit interrupt
1111b	RINT1	McBSP 1 receive interrupt

4.8 Power-Down Logic

Most of the operating power of CMOS logic is dissipated during circuit switching from one logic state to another. By preventing some or all of the chip's logic from switching, significant power savings can be realized without losing any data or operational context. Power-down mode PD1 blocks the internal clock inputs at the boundary of the CPU, preventing most of its logic from switching. PD1 effectively shuts down the CPU. Additional power savings are accomplished in power-down mode PD2, in which the entire on-chip clock structure (including multiple buffers) is halted at the output of the PLL. Power-down mode PD3 shuts down the entire internal clock tree (like PD2) and also disconnects the external clock source (CLKIN) from reaching the PLL. Wake-up from PD3 takes longer then wake-up from PD2 because the PLL needs to be relocked, just as it does following power up.

PD2 and PD3 assert the $\overline{PD}$ pin for external recognition of these two power-down modes. In addition to power-down modes, the IDLE instruction provides lower CPU power consumption by executing multiple NOPs. The IDLE instruction terminates only upon servicing an interrupt.