# Manual Update Sheet

**DATE: January 31, 2003**

**Document Being Updated:** ***TMS320C6000 Peripherals Reference Guide***

**Literature Number Being Updated: SPRU190D**

This Manual Update Sheet (SPRZ122C) describes changes for the *TMS320C6000 Peripherals Reference Guide* (SPRU190D).

Updates within paragraphs, figures, and tables appear in a **bold typeface**.

| **Page:** | **Change or Add:** |
|---|---|
| 1–9 | Add a C6416 column and a VCP/TCP coprocessors row, change the footnote in Table 1–2: |

*Table 1–2. TMS320C6000 Peripherals*

| Peripheral | C6201 | C6202(B) C6203(B) | C6204 | C6205 | C621x | C6414 | C6415 | C6416 | C6701 | C671x |
|---|---|---|---|---|---|---|---|---|---|---|
| Direct memory access (DMA) controller | Y | Y | Y | Y | N | N | N | N | Y | N |
| Enhanced direct memory access (EDMA) controller | N | N | N | N | Y | Y | Y | Y | N | Y |
| Host-port interface (HPI) | Y | N | N | N | Y | Y | Y† | Y† | Y | Y |
| Expansion bus (XBUS) | N | Y | Y | N | N | N | N | N | N | N |
| PCI | N | N | N | Y | N | N | Y† | Y† | N | N |
| External memory interface (EMIF) | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 |
| Boot configuration | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| Multichannel buffered serial ports (McBSPs) | 2 | 3 | 2 | 2 | 2 | 3 | 3† | 3† | 2 | 2 |
| UTOPIA | N | N | N | N | N | N | Y† | Y† | N | N |
| Interrupt selector | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| 32-bit timers | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 2 | 2 |
| Power-down logic | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |

**Page:** **Change or Add:**

| Peripheral | C6201 | C6202(B) C6203(B) | C6204 | C6205 | C621x | C6414 | C6415 | C6416 | C6701 | C671x |
|---|---|---|---|---|---|---|---|---|---|---|
| GPIO peripheral | N | N | N | N | N | Y | Y† | Y† | N | N |
| VCP/TCP coprocessors | N | N | N | N | N | N | N | Y | N | N |

† The C6415/C6416 peripheral set is selected at device reset. For details, see *Chapter 11, Boot Modes and Configuration*, and the specific device datasheet.

2–3 Change the last paragraph in section 2.2:

Table 2–1 and Table 2–2 compare the internal memory and cache configurations available on the current TMS320C6x0x devices. Figure 2–2 shows a block diagram of the connections between the C6201/C6204/C6205/C6701 CPU, PMEMC, and memory blocks. Figure 2–3 shows a block diagram of the connections between the CPU, PMEMC, and memory blocks in the C6202/C6202B/C6203(B). **For C6202(B)/C6203(B), there are two program memory controllers, PMEM1 and PMEM0. The PMEM1 controller handles all accesses to program memory block 1 (SRAM and cache), as well as all cache operations and external accesses. The PMEM0 controller always accesses program memory block 0 (SRAM only).** The addresses shown in Figure 2–2 and Figure 2–3 are for operation in memory map mode 1.

2–12 Add to the end of the paragraph in section 2.2.6:

While the CPU is executing from external memory, IPRAM block 1 can not be accessed using the DMA. The PMEM1 memory controller is used by the CPU to fetch instructions from the EMIF, therefore while performing a fetch from external memory, DMA access to PMEM1 is limited.

2–12 Add a new section 2.2.7:

### 2.2.7 Illegal Access to Program Memory

An access to a section of memory that does not return a ready indication is not allowed. Possible requestors are: CPU program fetches, CPU loads and stores, programmed DMA channels or HPI/PCI/XBUS host mastering of the DMA through the auxiliary DMA. This type of access can create a stall indefinitely. When a requestor has created a program memory stall, other requestors are unable to access this program memory space. For C6202/C6203, if an access generates a program memory block 0 stall, other requestors may still access program memory block 1 and vice versa.

| Page: | Change or Add: |
|---|---|
| 2–24 | Add a new section 2.4.8. The subsequent sections are renumbered accordingly: |

### 2.4.8 Illegal Access to Data Memory

An access to a section of memory that does not return a ready indication is not allowed. Possible requestors are: CPU program fetches, CPU loads and stores, programmed DMA channels or HPI/PCI/XBUS host mastering of the DMA through the auxiliary DMA. This type of access can create a stall indefinitely. When a requestor has created a data memory stall, other requestors are unable to access this data memory space.

3–all Chapter 3: TMS320C621x/C671x/C64x Two-Level Internal Memory. This chapter has been revised and divided into two new documents: *TMS320C621x/C671x Two-Level Internal Memory* (SPRU609) and *TMS320C64x Two-Level Internal Memory* (SPRU610). Updates to Chapter 3 that are not yet applied in those two new documents are documented in this manual update sheet.

3–11 Change the first paragraph in section 3.3.3:

The L2 operates in four operation modes, depending on the state of the CCFG register. **CPU may only perform read/write access to L2 addresses which are mapped as SRAM. Undefined operation may occur if CPU reads/writes from/to L2 addresses acting as cache.** Figure 3–6 shows the division of the L2 SRAM into mapped memory space and cache for each TMS320C621x/C671x L2 Mode. It also shows how the memory configuration for the L2 affects the proportion of cache and SRAM.

3–18 Change the title in section 3.4.4:

### 3.4.4 L1D Memory Banking Structure

3–33 Delete the last sentence of section 3.6.1: Since the L1D and L2 could be incoherent due to write hits in the L1D, the user should perform an L1D invalidation to force any dirty L1D data into the L2.

3–35 Change the paragraph in section 3.6.3:

Figure 3–26 shows four L1D misses when the L2 segment is configured as cache. The pipeline signals are explained in Figure 3–27. In this scenario, the CPU requests data in clock cycle 0 for read1 and read 2. In clock cycle 1 the data is looked for in L1D. The data is not present in L1D so in cycle 2 a miss is recorded for both read1 and read2. Also in cycle 1 the CPU requests the data for read3 and read4. In cycle 3, there is an L2 request for the data for read1 and a miss is recorded for both read3 and read4. **In cycles 4, 7, and 9 there are L2 requests for the data for read2, read3, and read4, respectively.** In cycles 9 and 10, the data for read1 is found in L2 and placed in L1. In cycles 11 and 12, the data for read2 is found in L2 and placed in L1. In cycle 13, the data from read1 and read2 is placed in the register file. Also in cycle 13 and in cycle 14 the data from read3 is found in L2 and placed in L1. In cycles 15 and 16 the data from read4 is found in L2 and placed in L1. In cycle 17, the data from read3 and read4 is placed in the register file. For these four misses, the CPU was stalled for a total of 14 clock cycles. This averages 4.67 cycles instead of 8 cycles for a single miss.

| Page: | Change or Add: |
|---|---|

3–39 Change the paragraph below Table 3–10:

The reset value of the L2MODE field is 000b, thus the L2 RAM is configured as mapped SRAM at reset to support data boot-loading. Any L2 RAM that is configured as cache is no longer in the memory map. For example, in L2 mode 010b the address range from **000F 0000h to 000F FFFFh** is no longer available in the TMS320C64x memory map. The associativity of the L2 cache RAM is a function of the L2 Mode on the C671x and C621x but stays at four-way for the C64x architecture. On C621x/C671x each ¼ of SRAM added in the cache increases the associativity by one line per set. To ensure coherency and data integrity on an L2 mode switch, the user must perform a series of operations.

3–40 Change the second paragraph in section 3.7.2:

The L2 SRAM is made up of four 64-bit-wide memory banks **on the C621x/C671x, and eight 64-bit-wide memory banks on the C64x.** Since the L1P data bus is 256-bits wide, any L1P request that occurs at the same time as an L1D or EDMA request will cause a bank collision and, therefore, a stall.

3–41 Add section 3.7.2.1, Figure 3–29, and Table 3–12. The subsequent figures and tables are renumbered accordingly:

#### 3.7.2.1 L2 Write Hits vs. EDMA Priority (C64x only)

C64x devices (revision 1.1 and later) incorporate a register to give EDMA accesses a temporary boost in priority so that they can meet real-time needs. This priority boost only applies when competing with write data from the CPU that misses in L1D, but hits in L2 cache or L2 SRAM. The EDMA weight register (EDMAWEIGHT) lets you control how often this priority boost is given. When EDMA priority is raised, it is allowed to complete one access before priority is returned to the CPU data. The EDMAWEIGHT is shown in Figure 3–29 and summarized in Table 3–12.

*Figure 3–29. TMS320C64x EDMA Weight Register (EDMAWEIGHT)*

| 31 … 2 | 1 … 0 |
|---|---|
| Reserved | EDMAWEIGHT |
| R,+0 | RW,+01 |

*Table 3–12. TMS320C64x EDMA Weight Register (EDMAWEIGHT) Field Description*

| Field | Description |
|---|---|
| EDMAWEIGHT | Allows EDMA priority raised over L1D–L2 writes once every N CPU cycles.<br>EDMAWEIGHT=00: L1D 100%, EDMA 0%, EDMA never gets priority over L1D.<br>EDMAWEIGHT=01: L1D 94%, EDMA 6%, EDMA gets priority every 16 cycles (default).<br>EDMAWEIGHT=10: L1D 80%, EDMA 20%, EDMA gets priority every 4 cycles.<br>EDMAWEIGHT=11: L1D 50%, EDMA 50%, EDMA gets priority every other cycle. |

3–41 Add section 3.7.3. The subsequent sections are renumbered accordingly.

### 3.7.3 Data Endianness

The data endianness of C621x/C671x/C64x is the same as C620x/C670x. See section 2.4.8, *Data Endianness*, for details.

| **Page:** | **Change or Add:** |
|---|---|

3–41 Change the paragraph in section 3.7.3:

When an L2 location is enabled as a cache, the operation is similar to the L1D cache. On a read request to the L2 the data is sent to the requestor if a hit occurs. If the data is not in the L2 the requestor is stalled and the Least Recently Used(LRU) line is allocated for the new data. If the allocated line contains valid data the L1D is snooped. The L1D must be snooped even if an L1P miss supplied the L2 miss address because the evicted L2 line could be cached in the L1D. If the L1D returns data both the matching L1D line and evicted L2 line are invalidated, otherwise only the evicted L2 line is invalidated. Both the L2 and L1D caches must be invalidated on an L1D match to maintain coherency between the caches. If the L1D returns dirty data or if the evicted L2 line contains dirty data that data is evicted to the external memory and the required data is requested from the Enhanced DMA. The L2 is a load through cache, thus **when servicing L1/L2 misses, data is stored in both L1 and L2 simultaneously. To minimize the CPU stall time, the L2 will fetch misses so that the data needed by L1 is returned first, followed by the rest of the L2 line. When the requested L1 data is available from the EDMA, the L2 will immediately forward it to L1 and unstall the CPU, and then wait for the remainder of the L2 line from the EDMA.**

4–4 Change the paragraph before Figure 4–1:

Figure 4–1 shows the TMS320C6000 block diagram with the DMA-related components shaded. **Table 4–1** summarizes the differences between the DMAs in different C6000 devices.

4–11 Change the bit name of bits 18–16 in Figure 4–3:

*Figure 4–3. DMA Channel Secondary Control Register (SECCTL)*

| 31–22 | 21 | 20 | 19 | 18–16 |
|---|---|---|---|---|
| Reserved | WSPOL* | RSPOL* | FSIG* | **DMAC EN** |
| R, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +000 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WSYNC CLR | WSYNC STAT | RSYNC CLR | RSYNC STAT | WDROP IE | WDROP COND | RDROP IE | RDROP COND | BLOCK IE | BLOCK COND | LAST IE | LAST COND | FRAME IE | FRAME COND | SX IE | SX COND |
| RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +1 | RW, +0 | RW,+0 | RW,+0 | RW, +0 | RW, +0 | RW,+0 | RW,+0 |

**Note:** * WSPOL, RSPOL, and FSIG bit fields are not available to the C6201 and C6701 devices. These bitfields are R+0 on the C6201 and C6701 devices.

4–12 Change the bit numbers (No.) of WSYNC CLR and DMAC EN in Table 4–5:

*Table 4–5. DMA Channel Secondary Control Register (SECCTL) Field Descriptions (Continued)*

| No. | Field | Description | Section |
|---|---|---|---|
| **15** | WSYNC CLR | Write synchronization status clear:<br>Read as 0 write 1 to clear write synchronization status. | 4.6.1 |
| **18 to 16** | DMAC EN | DMA action complete pins reflect status and condition. | 4.12 |

| Page: | Change or Add: |
|---|---|
| 4–20 | Add a paragraph after the introduction paragraph in section 4.6.1: |

> Care must be taken if software is used to poll and clear the status/conditions in the SECCTL register during a synchronized DMA transfer. To avoid inadvertently setting an extra RSYNC/WSYNC event during a synchronized DMA transfer, users should only write zeros to the STAT and CLR fields.

4–30 Change the last paragraph in section 4.8.1:

**When a DMA channel is operating in split mode, only one element count and one frame count are used for both the transmit and receive transfers. The end of frame or end of block is set following the last transfer. When the channel operating in split mode is servicing a McBSP, this will normally be the last receive transfer because the transmit transfers will normally run ahead of the receive transfers. The transfer counters will be modified after the transmit transfer, so that if autoinitialization is enabled, the transfer counters may indicate that another transfer has begun before the receive portion of the split-mode transfer has completed.** For split-channel operation to work properly, both the RSYNC and WSYNC fields must be set to non-zero synchronization events. Also, frame synchronization must be disabled in split-channel operation.

4–31 Change the last paragraph in section 4.8.1:

The above sequence is maintained for all transfers. However, the transmit transfers do not have to wait for all previous receive element transfers to finish before proceeding. Therefore, it is possible for the transmit stream to get ahead of the receive stream. The DMA channel transfer counter decrements (or reinitialize) after the associated transmit transfer finishes. However, reinitialization of the source address register occurs after all transmit element transfers finish. This configuration works as long as transmit transfers **do not exceed eight or more transfers** ahead of the receive transfers. If the transmit transfers do get ahead of the receive transfers, transmit element transfers are stopped, possibly causing synchronization events to be missed. For cases in which receive or transmit element transfers are within seven or less transfers of the other, the DMA channel maintains this information as internal status.

4–33 Change the Description of CH PRI (bits 3 to 0) in Table 4–9:

*Table 4–9. DMA Auxiliary Control Register (AUXCTL) Field Descriptions*

| No. | Field | Description |
|---|---|---|
| 4 | AUXPRI | Auxiliary channel priority mode<br><br>AUXPRI = 0: CPU priority<br>AUXPRI = 1: DMA priority |
| 3 to 0 | CH PRI | **DMA channel priority**<br><br>CH PRI = 0000b: fixed channel priority mode auxiliary channel highest priority<br>CH PRI = 0001b: fixed channel priority mode auxiliary channel 2nd-highest priority<br>CH PRI = 0010b: fixed channel priority mode auxiliary channel 3rd-highest priority<br>CH PRI = 0011b: fixed channel priority mode auxiliary channel 4th-highest priority<br>CH PRI = 0100b: fixed channel priority mode auxiliary channel lowest priority<br>CH PRI = other, reserved |

| Page: | Change or Add: |
|---|---|
| 4–34 | Change the paragraph in section 4.9.2: |

A higher priority channel gains control of the DMA controller from a lower priority channel once it has received the necessary read synchronization. In switching channels, the current channel allows all data from requested reads to be completed. The DMA controller determines which higher priority channel gains control of the DMA controller read operation. That channel then starts its read operation. Simultaneously, write transfers from the previous channel are allowed to finish. **The write transfer must complete before the higher priority channel will be able to start its transfer. Arbitration of the higher priority channel will occur as soon as the write from the lower priority channel completes. For example, if the lower priority channel's write is blocked by the CPU, the higher priority channel will not be able to start until the CPU releases the contending resource and the write is able to complete. This occurs even if the higher priority channel is accessing a different resource.** See Chapter 5, *DMA and CPU Data Access Performance,* for more detail.

4–42 Add before the last paragraph in section 4.11.2.2:

As with the shared FIFO DMA, a higher priority DMA channel can still be stalled by a lower priority channel if the lower priority channel is unable to complete its write to a resource. Arbitration of the higher priority channel will occur as soon as the write from the DMA completes.

5–13 Add a new section 5.2.7:

### 5.2.7 DMA Port Crossing

The DMA has 4–6 master ports, all of which are listed below:

- Data Memory
- Program Memory Block 0
- Program Memory Block 1 (on C6202/C6203 only)
- XBUS I/O (on C6202/C6203/C6204 only)
- EMIF
- Internal Peripheral Bus (peripheral control registers including McBSP data registers)

The DMA auxiliary port is a slave port and should be considered a requestor much like a programmed DMA channel.

DMA accesses/bursts are not permitted to cross a port boundary. See section 11.3, *Memory Map*, for a listing of C620x/C670x memory map and port boundaries.

**Page:** **Change or Add:**

6–4 Add a row to Table 6–1:

*Table 6–1. Differences in TMS320C6000 EDMAs*

| Features | Supported on Device | Described in Section |
|---|---|---|
| EDMA rate | Runs at CPU rate on C621x/C671x; runs at half of CPU rate on C64x | 6.1 |

6–13 Change the second paragraph in section 6.5:

The contents of the 2K byte PaRAM, shown in Table 6–3 comprises:

- For C621x/C671x, there are 16 transfer parameter entries for the 16 EDMA events. For C64x, there are 64 transfer parameter entries for the 64 EDMA events. Each entry is six words or 24 bytes. **These areas can also serve as reload/link parameters.**
- **Remaining parameter entries (69 entries for C621x/C671x, and 21 entries for C64x) serve as additional parameter sets used for linking transfers.** Each set or entry is 24 bytes.
- 8 bytes of unused RAM that can be used as scratch pad area. Note that a part or entire EDMA RAM can be used as a scratch pad RAM provided this area corresponding to an event(s) is disabled. It is the user’s responsibility to provide the transfer parameters when the event is eventually enabled.

| **Page:** | **Change or Add:** |
|---|---|
| 6–14 | Change Table 6–3: |

*Table 6–3. EDMA Parameter RAM Contents*

| Address | C621x/C671x | C64x |
|---|---|---|
| 01A0 0000h to 01A0 0017h | Parameters for event 0 (6 words) | |
| 01A0 0018h to 01A0 002Fh | Parameter for event 1 (6 words) | |
| 01A0 0030h to 01A0 0047h | Parameters for event 2 (6 words) | |
| 01A0 0048h to 01A0 005Fh | Parameters for event 3 (6 words) | |
| 01A0 0060h to 01A0 0077h | Parameters for event 4 (6 words) | |
| 01A0 0078h to 01A0 008Fh | Parameters for event 5 (6 words) | |
| 01A0 0090h to 01A0 00A7h | Parameters for event 6 (6 words) | |
| 01A0 00A8h to 01A0 00BFh | Parameters for event 7 (6 words) | |
| 01A0 00C0h to 01A0 00D7h | Parameters for event 8 (6 words) | |
| 01A0 00D8h to 01A0 00EFh | Parameters for event 9 (6 words) | |
| 01A0 00F0h to 01A0 0107h | Parameters for event 10 (6 words) | |
| 01A0 0108h to 01A0 011Fh | Parameters for event 11 (6 words) | |
| 01A0 0120h to 01A0 0137h | Parameters for event 12 (6 words) | |
| 01A0 0138h to 01A0 014Fh | Parameters for event 13 (6 words) | |
| 01A0 0150h to 01A0 0167h | Parameters for event 14 (6 words) | |
| 01A0 0168h to 01A0 017Fh | Parameters for event 15 (6 words) | |
| 01A0 0180h to 01A0 0197h | Additional reload/link entry (6 words) | Parameters for event 16 (6 words) |
| 01A0 0198h to 01A0 01AFh | Additional reload/link entry (6 words) | Parameters for event 17 (6 words) |
| ... | Additional reload/link entry (6 words) | ... |
| ... | Additional reload/link entry (6 words) | ... |
| 01A0 05D0h to 01A0 05E7h | Additional reload/link entry (6 words) | Parameters for event 62 (6 words) |
| 01A0 05E8h to 01A0 05FFh | Additional reload/link entry (6 words) | Parameters for event 63 (6 words) |
| 01A0 0600h to 01A0 0617h | Additional reload/link entry (6 words) | |
| 01A0 0618h to 01A0 062Fh | Additional reload/link entry (6words) | |
| … | … | |
| 01A0 07E0h to 01A0 07F7h | Additional reload/link entry (6 words) | |
| 01A0 07F8h to 01A0 07FFh | Scratch pad area (2 words) | |

| **Page:** | **Change or Add:** |
|---|---|

6–16 Change all read/write fields and the footnote, add a footnote in Figure 6–9:

*Figure 6–9. Options (OPT) Bit–Fields*

| 31–29 | 28–27 | 26 | 25–24 | 23 | 22–21 | 20 | 19–16 |
|---|---|---|---|---|---|---|---|
| PRI | ESIZE | 2DS | SUM | 2DD | DUM | TCINT | TCC |
| RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** |

| 15 | 14–13 | 12 | 11 | 10–5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| rsvd‡ | TCCM† | ATCINT† | rsvd‡ | ATCC† | rsvd‡ | PDTS† | PDTD† | LINK | FS |
| **RW,+x** | RW,**+x** | RW,**+x** | **RW,+x** | RW,**+x** | **RW,+x** | RW,**+x** | RW,**+x** | RW,**+x** | RW,**+x** |

† Applies to C64x only. On C621x/C671x, **you should always write 0 to these reserved fields.**
‡ **You should always write 0 to the reserved fields.**

6–17 Change the Description of ESIZE (bits 28–27) in Table 6–5:

*Table 6–5. EDMA Channel Options Parameter (OPT) Description (C621x/C671x/C64x)*

| Bit No. | Field | Description | Section |
|---|---|---|---|
| 28–27 | ESIZE | Element size | 6.9 |
| | | ESIZE=00b; **32-bit word, or 64-bit doubleword (on certain C64x transfers only. See section 6.9.1)** | |
| | | ESIZE=01b; 16-bit half-word | |
| | | ESIZE=10b; 8-bit byte | |
| | | ESIZE=11b; reserved | |

6–20 Change the second paragraph in section 6.6.5:

Element index provides an address offset **(in bytes)** to the next element in a frame. Element index is used *only* for 1D transfers. This is because 2D transfers do not allow spacing between elements, and hence the term ‘array’ is used to define a group of contiguous elements. Frame/array index provides an offset **(in bytes)** to the next frame/array in a block.

6–20 Change the title and the first paragraph in section 6.6.7:

### 6.6.7 EDMA Performance Considerations on the C621x/C671x

The EDMA controller provides a mechanism to link EDMA transfers. This is analogous to the autoinitialization feature in the DMA. **When LINK=1 in the EDMA options parameter,** the 16-bit link address specified in the EDMA parameter RAM specifies the lower 16-bit address in the parameter RAM from which the EDMA loads/reloads the parameters of the next event in the chain. Since the entire EDMA parameter RAM is located in the 01A0 xxxxh area, only the lower 16-bit address matters.

| Page: | Change or Add: |
|---|---|

6–24, 25 Change events on EDMA channel numbers 28, 29, 30, 31, and 53 in Table 6–8*:*

*Table 6–8. EDMA Channel Synchronization Events – TMS320C64x*

| EDMA Channel Number | Event Acronym | Event Description |
|---|---|---|
| 28 | VCPREVT | VCP receive interrupt |
| 29 | VCPXEVT | VCP transmit interrupt |
| 30 | TCPREVT | TCP receive interrupt |
| 31 | TCPXEVT | TCP transmit interrupt |
| 53 | GPINT13 | GPIO event 13 |

6–32 Delete the last paragraph in section 6.9: When transferring a burst of elements to or from a 64 bit wide peripheral (e.g. L2 or EMIFA), 64-bit elements are transferred regardless of the ESIZE programmed. This allows the EDMA to maximize the available bandwidth.

6–32 Change the second paragraph in section 6.9.1:

When transferring a burst of elements to or from a 64-bit-wide peripheral (for example, L2 SRAM or EMIFA), 64-bit elements are transferred to maximize the available bandwidth if the element size is 32-bit word (ESIZE = 00b).

Care must be taken when performing a fixed-mode access (SUM or DUM = fixed) to peripherals that have 64-bit data paths to/from the EDMA. These include L2 SRAM, EMIFA (C64x only), and TCP/VCP (C64x only).

If the EDMA is setup with the following parameters:

- ❑ Element size is 32-bit word (ESIZE = 00b)
- ❑ Fixed address mode (SUM or DUM = 00b in the options parameter)
- ❑ Transfer/synchronization type is array-/frame-/block-synchronized (not element-synchronized, see section 6.10)
- ❑ Element count is greater than 1 (ELECNT > 1).
- ❑ Either the source or destination bus width is 64 bits.

Then the programmer must ensure that the following conditions are true:

- ❑ Element count (ELECNT) must be a multiple of 2.
- ❑ Frame/Array index field must be a multiple of 2.

Operation is undefined if the above conditions are not met.

6–32 Change the last paragraph in section 6.9.1:

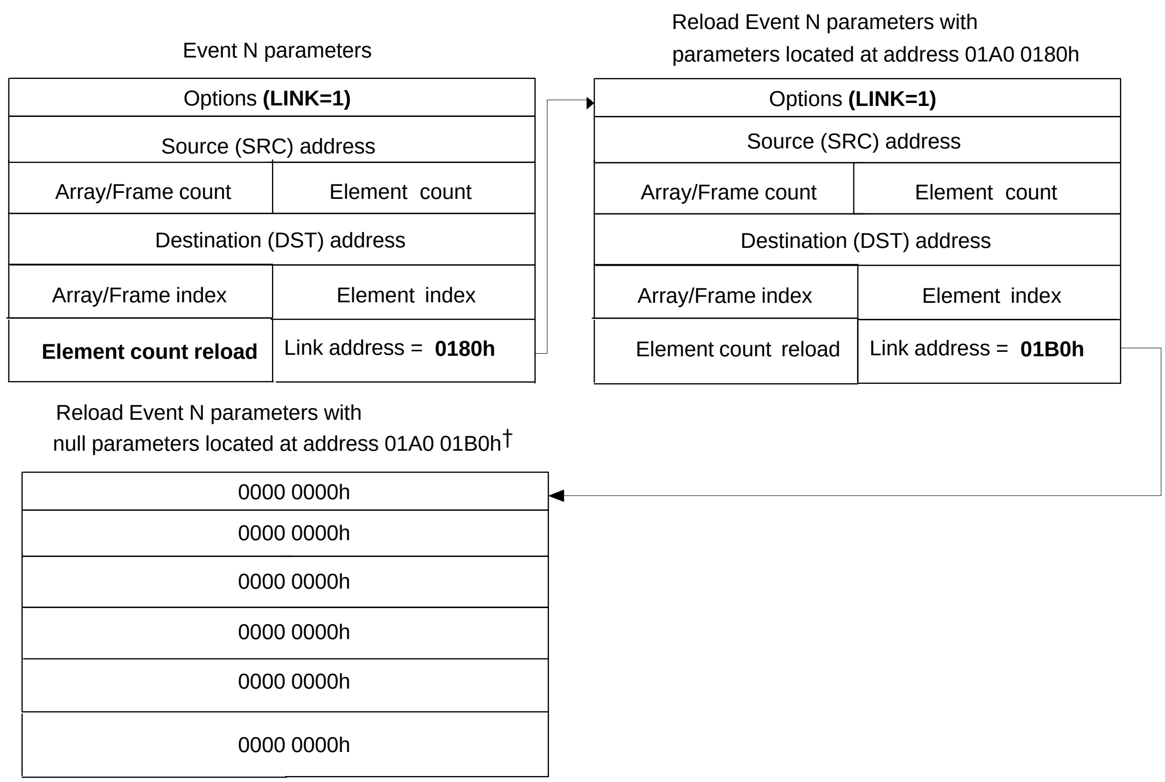
For a write to a 64-bit-wide data bus with the above conditions, **both word 0 and word 1 of the fixed doubleword address are updated.** For example, **under the above conditions** a write to the L2 SRAM address 0x00000000 updates both word 0 (at address 0x00000000) and word 1 (at address 0x00000004) with the new data.

| Page: | Change or Add: |
|---|---|
| 6–34 | Change the paragraph in section 6.10.1: |

There is a special condition for reloading the element count for element synchronized (FS = 0) 1D transfers. In this case the address is updated by element size or element/frame index depending on SUM/DUM fields. See the first row in Table 6–11. Therefore, the EDMA controller keeps track of the element count to update the address. When an element sync event occurs at the end of a frame (ELECNT = 1), the EDMA controller sends off the transfer request, and reloads the ELECNT from the element count reload field in the parameter RAM. This element count reload occurs when element count is 1 and the frame count is nonzero. **When configuring transfers where ELERLD will be used, ELERLD must be set to a nonzero value, or the transfer will hang.** For all other types of transfers, the 16-bit element count reload field is not used because the address generation hardware (transparent to users) tracks the address directly.

6–39 Change “Elementary count reload” to “Element count reload” in Figure 6–16:

*Figure 6–16. Linked EDMA Transfer*




† See section 6.13 for details on null parameters

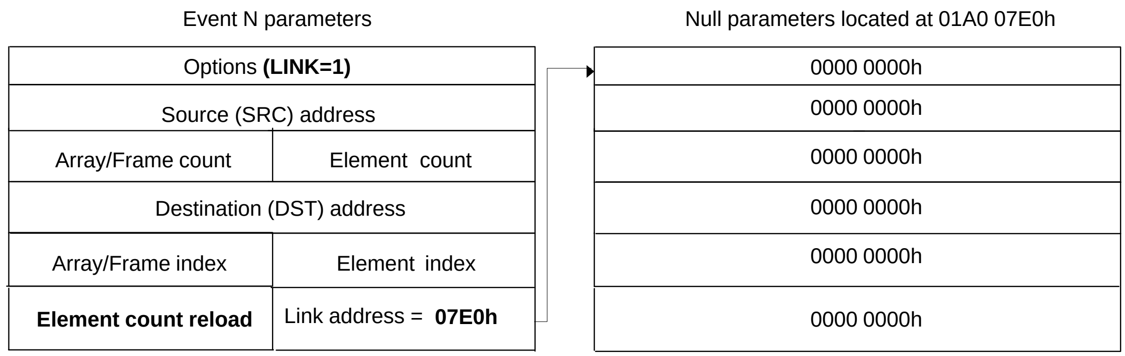
| Page: | Change or Add: |
|---|---|

6–40 Change the paragraph in section 6.12:

The link address is evaluated only if LINK is equal to 1 *and* only after the event parameters have been exhausted. An event's parameters are exhausted when the EDMA controller has completed the transfer associated with the request. Table 6–13 shows the channel completion conditions when the linking of parameters is performed. There is virtually no limit to the length of linked transfers. **The last transfer parameter entry should have its LINK = 1 to link to a NULL parameter set so that the linked transfer stops after the last transfer. See section 6.13 for details.**

6–41 Change "Elementary count reload" to "Element count reload" in Figure 6–17:

*Figure 6–17. Terminating EDMA Transfers*




6–45 Change the TCC in Options column in Table 6–14:

*Table 6–14. Transfer Complete Code (TCC) to EDMA Interrupt Mapping*

| TCC in Options (TCINT=1) | CIPR Bits Set | TCC in Options (TCINT=1) | CIPR Bits Set |
|---|---|---|---|
| 0000b | CIP0 | 1000b | CIP8 |
| 0001b | CIP1 | **1001b** | CIP9 |
| 0010b | CIP2 | **1010b** | CIP10 |
| 0011b | CIP3 | **1011b** | CIP11 |
| 0100b | CIP4 | **1100b** | CIP12 |
| 0101b | CIP5 | 1101b | CIP13 |
| 0110b | CIP6 | 1110b | CIP14 |
| 0111b | CIP7 | 1111b | CIP15 |

**Page:** **Change or Add:**

6–45 Change the TCC in Options column in Table 6–15:

*Table 6–15. C64x Transfer Complete Code (TCC) to EDMA Interrupt Mapping*

| **TCC in Options (TCINT=1)** | **CIPRL Bits Set†** | **TCC in Options (TCINT=1)** | **CIPRH Bits Set†** |
|---|---|---|---|
| 00 0000b | CIP0 | 10 0000b | CIP32 |
| 00 0001b | CIP1 | 10 0001b | CIP33 |
| 00 0010b | CIP2 | 10 0010b | CIP34 |
| 00 0011b | CIP3 | 10 0011b | CIP35 |
| 00 0100b | CIP4 | **10 0100b** | CIP36 |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| 01 1110b | CIP30 | 11 1110b | CIP62 |
| 01 1111b | CIP31 | 11 1111b | CIP63 |

6–48 Change the second paragraph in section 6.15.1:

To enable the EDMA controller to chain channels by way of a single event, the TCINT bit must be set to ‘1’. Additionally, the relevant bit in the channel chain enable register (CCER) in Figure 6–20 should be set to trigger off the next channel transfer specified by TCC. Since events 8 to 11 are the only EDMA channels that support chaining, only these bits are implemented in CCER. Reading unused bits returns **the corresponding bits in the EER** and writing to them has no effect. Therefore, one can still specify a TCC value between 8 and 11, and need not necessarily initiate the transfer on channels 8-11. However, the event is still captured in the ER[11:8] even if the corresponding bit in CCER is disabled. This allows selective enabling and disabling of these four specific events.

6–48 Change the paragraph in section 6.15.2:

The C64x EDMA transfer chaining is an expansion of the C621x/C671x transfer chaining. Any of the 64 transfer completion codes of the C64x EDMA can be used to trigger another channel transfer. The user-specified transfer complete code is expanded to a 6-bit value TCCM:TCC. The 4 bits in the TCC field (bits 19 to 16) of the options parameter are the least significant bits of the transfer complete code, while the new TCCM bit fields are the most significant bits of the transfer complete code. For example, if the transfer complete code (TCCM:TCC) is 010001b (i.e. TCCM = 01, TCC = 0001b) and CCERL[17] = 1 is specified for EDMA channel 4, the completion of the channel 4 transfer will initiate the next transfer specified by EDMA channel 17, provided that the channel 4 TCINT = 1. **Unlike the C621x/C671x, the event bits on the C64x are captured in the ER *only* if the corresponding bits in CCER are enabled.**

**Page:** **Change or Add:**

6–55 Change the C64x Requesters column in Table 6–16:

*Table 6-16. Programmable Priority Levels for Data Requests*

| PRI(31:29) | C621x/C671x Priority Level | C621x/C671x Requestors | C64x Priority Level | C64x Requesters |
|---|---|---|---|---|
| 000b | Level0; urgent priority | L2 controller | Level0; urgent priority | L2 controller, EDMA, QDMA**, HPI, and PCI** |
| 001b | Level1; high priority | EDMA, QDMA and/or HPI | Level1; high priority | L2 controller, EDMA, QDMA**, HPI, and PCI** |
| 010b | Level2; low priority | EDMA, QDMA | Level 2; medium priority | L2 controller, EDMA, QDMA**, HPI, and PCI** |
| 011b | Reserved | Reserved | Level 3, low priority | L2 controller, EDMA, QDMA**, HPI, and PCI** |
| 100b –111b | Reserved | Reserved | Reserved | Reserved |

6–55 Change the paragraph in section 6.17.1:

The priority queue status register (PQSR) shown in Figure 6–24 (C621x/C671x) and Figure 6–25 (C64x) indicates whether the transfer request queue is empty on the priority level queues. Status bits PQ in the PQSR provide the status of the queues. A '1' in the PQ bit indicates that there are no requests pending in the respective priority level queue. **For C621x/C671x**, if PQSR[0] is '1', this means all L2 requests for data movement have been completed and there are no requests pending in the priority level 0 queue. **For C64x, if PQSR[0] is '1', this means all requests for data movement from requestors programmed for priority level 0 have been completed.**

6–56 Add a paragraph after the section header in section 6.17.2:

### 6.17.2 Transfer Request Queue Length

Care should be taken to not overload any priority queue, as overloading any one queue can adversely affect all queues. When a transfer is submitted to a queue that is full, the EDMA controller stalls until room in the queue is available. While stalled, the EDMA controller does not process any other events, including those events that submit requests on a different priority queue. Events are still captured in the ER and processed when the EDMA controller is released.

| **Page:** | **Change or Add:** |
|---|---|
| 6–57 | Change Table 6–18: |

*Table 6–18. Transfer Request Queues (C64x)†*

| Queue | Priority Level (PRI) | Total Queue Length (fixed) | Requester | Default Queue Length | Register to Program Queue Length |
|---|---|---|---|---|---|
| Q0 | 0; urgent priority | 16 | L2 controller and QDMA | 6 | L2ALLOC0 |
| | | | EDMA | 2 | PQAR0 |
| | | | HPI/PCI | 0 | TRCTL/TRCTL |
| Q1 | 1; high priority | 16 | L2 controller and QDMA | 2 | L2ALLOC1 |
| | | | EDMA | 6 | PQAR1 |
| | | | HPI/PCI | 0 | TRCTL/TRCTL |
| Q2 | 2; medium priority | 16 | L2 controller and QDMA | 2 | L2ALLOC2 |
| | | | EDMA | 2 | PQAR2 |
| | | | HPI/PCI | 4 | TRCTL/TRCTL |
| Q3 | 3; low priority | 16 | L2 controller and QDMA | 2 | L2ALLOC3 |
| | | | EDMA | 6 | PQAR3 |
| | | | HPI/PCI | 0 | TRCTL/TRCTL |

† L2 controller and QDMA share one queue allocation. L2ALLOCx register controls this queue allocation length.
HPI and PCI share one queue allocation.
TRCTL register in the HPI module controls the queue allocation length of HPI requests.
TRCTL register in the PCI module controls the queue allocation length of PCI requests.

6–58 Change Figures 6-26, 6-27, 6-28, and 6-29:

*Figure 6–26. Priority Queue Allocation Register 0 (PQAR0) (C64x only)*

| 31 – **4** | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| rsvd | **rsvd** | PQA2 | PQA1 | PQA0 |
| R,+0 | **RW,+0** | RW,+0 | RW,+1 | RW,+0 |

*Figure 6–27. Priority Queue Allocation Register 1 (PQAR1) (C64x only)*

| 31 – **4** | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| rsvd | **rsvd** | PQA2 | PQA1 | PQA0 |
| R,+0 | **RW,+0** | RW,+1 | RW,+1 | RW,+0 |

*Figure 6–28. Priority Queue Allocation Register 2 (PQAR2) (C64x only)*

| 31 – **4** | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| rsvd | **rsvd** | PQA2 | PQA1 | PQA0 |
| R,+0 | **RW,+0** | RW,+0 | RW,+1 | RW,+0 |

**Page:** **Change or Add:**

*Figure 6–29. Priority Queue Allocation Register 3 (PQAR3) (C64x only)*

| 31–4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| rsvd | **rsvd** | PQA2 | PQA1 | PQA0 |
| R,+0 | **RW,+0** | RW,+1 | RW,+1 | RW,+0 |

6–59 Change the first bullet in section 6.18:

- **EDMA stalls** occur when the EDMA submits another request to a priority level queue that is already full.

6–59 Add the following text after the bullets in section 6.18:

EDMA bandwidth is fully utilized when performing burst transfer, which is obtained if and only if the EDMA transfer is configured as:

- Transfer/synchronization type is array-/frame-/block-synchronized transfer (not element-synchronized, see section 6.10).
- Element size is 32-bit (ESIZE = 00b)
- Addressing mode is increment, decrement, or fixed (SUM or DUM = 00/01/10b in the options parameter)

The EDMA will perform single element transfers for all transfers not meeting all of the above conditions, which will utilize only a portion of the bandwidth available.

For burst transfer types described above, the burst length is dictated by the 1D component of the transfer, which is specified by the ELECNT field. For array- or frame-synchronized transfer, the 1D component of the transfer is the amount of data that gets transferred per synchronization event. For block-synchronized transfers, the complete 2D transfer is transferred per synchronization event; however, burst transfers are only performed for the 1D component. If the 1D length (ELECNT) is programmed to a small value, the performance will reduce accordingly and in the worst case (ELECNT = 1), the performance will be identical to the performance described for single element transfers.

6–61 Change Figure 6–32:

*Figure 6–32. QDMA Options Register (QDMA_OPT, QDMA_S_OPT)*

| 31 29 | 28 27 | 26 | 25 24 | 23 | 22 21 | 20 | 19 16 | 15 | 14 13 | 12 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PRI | ESIZE | 2DS | SUM | 2DD | DUM | TCINT | TCC | Rsvd | **TCCM** | Rsvd | FS |
| W, +0 | W, +0 | W, +0 | W, +0 | W, +0 | W, +0 | W, +0 | W, +0 | W,+0 | W,+0 | W,+0 | W,+0 |

**Notes:** 1) TCCM applies to C64x only. For C621x/C671x, this bit is Reserved W,+0.
2) Register QDMA_OPT is read/writable. Pseudo-register QDMA_S_OPT is write only.

| Page: | Change or Add: |
|---|---|
| 6–62 | Change the first paragraph in section 6.19.5: |
| | The QDMA has several stalling conditions. Once a write has been performed to one of the pseudo-registers (resulting in a pending QDMA transfer request), future writes to the QDMA registers are stalled until the transfer request is sent. Normally this will occur for 2–3 EDMA cycles, as this is how long it takes to submit a transfer. **Stalls are not generally seen by the CPU, because writes to QDMA registers occur via the L1D write buffer. Future writes to the buffer may eventually fill it up and stall the CPU from subsequent reads/ writes.** |
| 6–62 | Change the third paragraph in section 6.19.5: |
| | Similar to the EDMA channels, QDMA can have programmable priority in the lower levels as described in section 6.17. The PRI bit-field in the QDMA_OPT register specifies the priority level of the QDMA. **On the C621x/C671x, level 0 (urgent priority) is reserved for L2 cache accesses; thus, QDMA requests with level 0 or reserved values will be discarded.** |
| 6–66 | Change the first paragraph in section 6.21.1.1: |
| | For TMS320C64x, cache servicing requests can be made on any priority levels as specified in the P bits in the CCFG register. For read requests, the cache controller always requests an L2 line in two bursts of 64 bytes each, requesting the “missed” portion of the line first. For write requests, as a result of flush/clean operations or eviction, the cache controller transfers one complete L2 line in two bursts of **64 bytes** each. |
| 6–66 | Change the paragraph in section 6.21.1.2: |
| | The HPI/PCI automatically generates transfer requests to service host activity. **For C621x/C671x,** these transfer request submissions are submitted only with a high priority and are invisible to the user. **For C64x, by default HPI/PCI transfer requests are submitted with medium priority, but request priority can be programmed to any of the four priority levels by setting the PRI field in the TRCTL register to the appropriate value.** The HPI/PCI submits a transfer request for a single element read or write for fixed mode host accesses and a transfer request for a short data burst for autoincrement transfers. The burst size is always for eight or fewer elements. See section 6.17 for available HPI transfer request priority. |
| 6–66 | Change the paragraph in section 6.21.1.3: |
| | The EDMA channel transfers can be submitted with **urgent (C64x only),** high, medium (C64x only), or low priority; with the recommendation that high priority be reserved for short bursts and single element transfers and low priority be used for longer (background) block moves. It is also recommended that transfers be divided between the priority levels when applicable, as this helps to maximize the device performance. |
| 6–73 | Change the fourth paragraph in section 6.22.3: |
| | For this example it is assumed that the 16-bit data is located in external RAM, beginning at address 0xA0000000 (CE2). The QDMA is used to bring four frames of 1k half-words from their locations in RAM to internal data memory beginning at **0x00002000**. The index value required is ELEIDX = F x S = 4 x 2 = 8. |

**Page:** **Change or Add:**

7–3 Add a paragraph below Figure 7–2 in section 7.1:

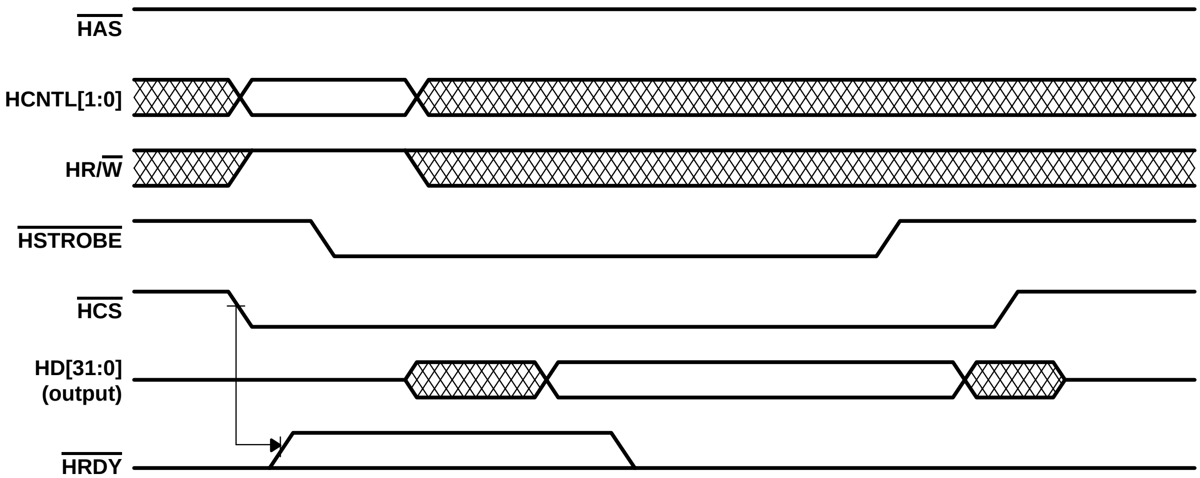
> Through the HPI, an external host is capable of accessing the entire DSP memory map *except* the following:
>
> - L2 control registers (C6x1x only)
> - Interrupt selector registers
> - Emulation logic

7–21 Add a paragraph at the end of section 7.4.3:

When performing reads with autoincrement, the C64x differs slightly from the C621x/C671x in that the C64x will not indicate ready ($\overline{\text{HRDY}}$ low) until the internal read buffer has filled with the 16-word prefetch. Thus, accesses to slow regions of memory, such as internal peripheral registers or slow external memory, might take a significant amount of time. For best performance, accesses to these regions should be done in fixed mode unless multiple words are desired.

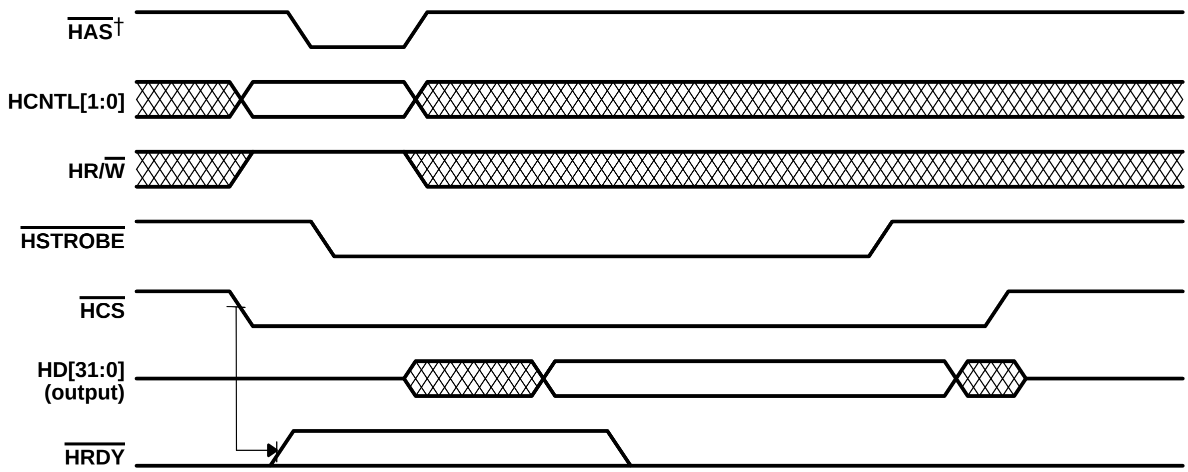
7–22 Change Figure 7–11 (delete $\overline{\text{HRDY}}$ case 2 waveform):

*Figures 7–11. HPI32 Read Timing ($\overline{\text{HAS}}$ Not Used, Tied High) for C64x only*



| Page: | Change or Add: |
|---|---|
| 7–22 | Change Figure 7–12 (delete $\overline{\text{HRDY}}$ case 2 waveform): |

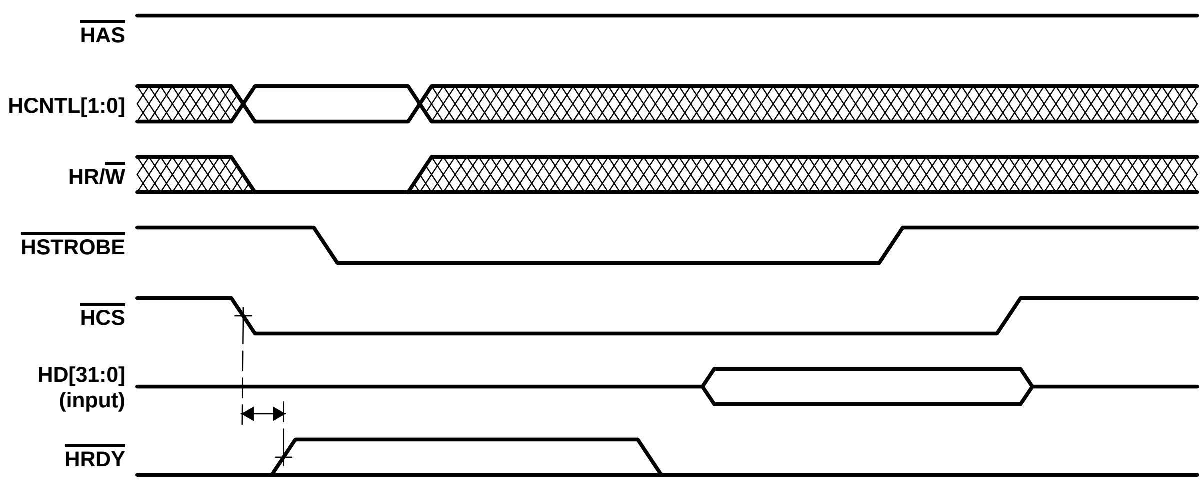
*Figures 7–12. HPI32 Read Timing ($\overline{\text{HAS}}$ Used) for C64x only*




† For correct operation, strobe the $\overline{\text{HAS}}$ signal only once per $\overline{\text{HSTROBE}}$ cycle.

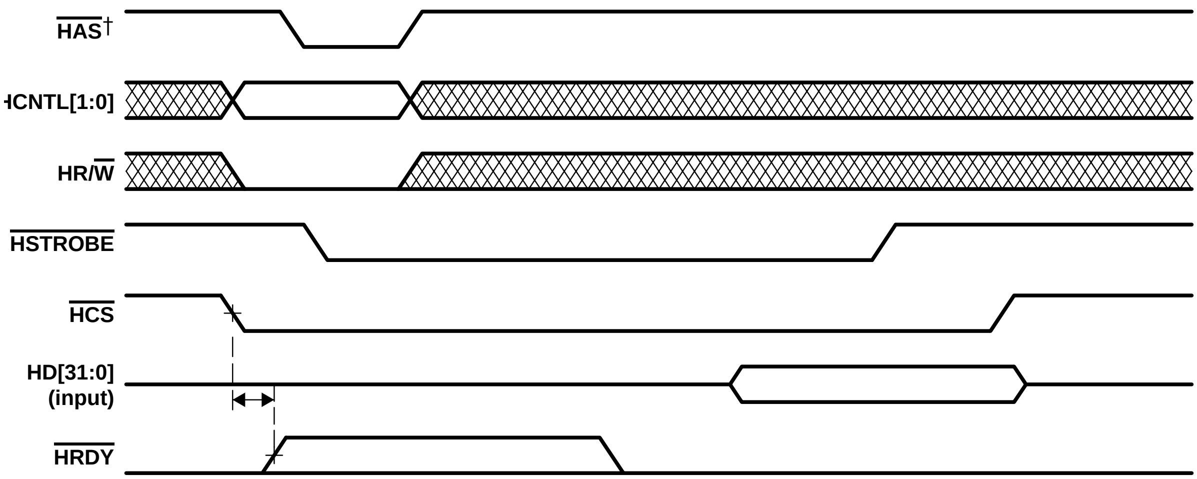
| | |
|---|---|
| 7–23 | Change the HR/$\overline{\text{W}}$ and $\overline{\text{HRDY}}$ waveforms in Figure 7–13: |

*Figure 7–13. HPI Write Timing ($\overline{\text{HAS}}$ Not Used, Tied High) for C64x only*



| Page: | Change or Add: |
|---|---|
| 7–23 | Change the HR/$\overline{\text{W}}$ and $\overline{\text{HRDY}}$ waveforms in Figure 7–14: |

*Figure 7–14. HPI Write Timing ($\overline{\text{HAS}}$ Used) for C64x only*




† For correct operation, strobe the $\overline{\text{HAS}}$ signal only once per $\overline{\text{HSTROBE}}$ cycle.

7–24 Add a row to Table 7–7:

*Table 7–7. HPI Registers for C64x*

| Register Abbreviation | Register Name | Host Read/Write Access | CPU Read/Write Access | CPU Read/Write (Hex Byte Address) |
|---|---|---|---|---|
| TRCTL | TR control | – | RW | 018A 0000h |

7–25 Change the first paragraph in section 7.5.1:

The HPIA contains the address of the memory accessed by the HPI at which the current access occurs. **This address is a 32-bit word address with all 32-bits readable/writable. The two LSBs always function as 0, regardless of the value read from their location.** The C62x/C67x HPIA register is only accessible by the host. It is not mapped to the DSP memory.

7–25 Change the second paragraph in section 7.5.1:

The C64x HPIA register is accessible by both the host and the CPU. Furthermore, the HPIA register is separated into two registers internally: the HPI address write register (HPIAW), and the HPI address read register (HPIAR). By separating the HPIA into HPIAW and HPIAR internally, the CPU can update the read and write memory address independently to allow the host to perform read and write to different address ranges. **When reading HPIA from the CPU, the value returned corresponds to the address currently being used by the HPI and DMA to transfer data inside the DSP. It is not the address for the current transfer at the external pins. Thus, reading HPIA does not indicate the status of a transfer, and should not be relied upon to do so.**

| **Page:** | **Change or Add:** |
|---|---|

7–25 Change the paragraph in section 7.5.2:

The HPIC register, shown in Figure 7–15 **and Figure 7–16** and summarized in Table 7–8, is normally the first register accessed to set configuration bits and initialize the interface. **From the host's view,** the HPIC is organized as a 32-bit register **with two identical halves, meaning the** high halfword and low halfword contents are the same. **On a host write, both halfwords must be identical, except when writing the DSPINT bits in HPI16 mode (see section 7.5.4). In HPI16 mode when setting DSPINT = 1, the host must only write '1' to the lower 16-bit halfword or upper 16-bit halfword, but not both. In HPI32 mode, the upper and lower halfwords must always be identical. From the C6000 (CPU) view, the HPIC is a 32-bit register with only 16-bits of useful data.** Only CPU writes to the lower halfword affect HPIC value and HPI operation.

7–25 Add a second paragraph in section 7.5.2:

On C64x, the HWOB bit is writable by the CPU. Therefore, care must be taken when writing to the HPIC, in order not to write an undesired value to HWOB.

7–26 Change the title, all read/write fields, and the footnote in Figure 7–15:

*Figure 7–15. HPIC Register—Host Reference View*

| 31 | 30 24 | 23 | 22 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| rsvd† | rsvd | rsvd† | rsvd | FETCH | HRDY | HINT | DSPINT | HWOB |
| HRW,+0 | HR,+0 | HRW,+0 | HR,+0 | HRW,+0 | HR,+1 | HRW,+0 | HRW,+0 | HRW,+0 (C62x/C67x)<br>HRW,+0 (C64x) |

| 15 | 14 8 | 7 | 6 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| rsvd† | rsvd | rsvd† | rsvd | FETCH | HRDY | HINT | DSPINT | HWOB |
| HRW,+0 | HR,+0 | HRW,+0 | HR,+0 | HRW,+0 | HR,+1 | HR,+0 | HRW,+0 | HRW,+0 (C62x/C67x)<br>HRW,+0 (C64x) |

† For C62x/C67x, bits 7, 15, 23, 31 are read-only; **HR,+0**. For C64x, bits 7, 15, 23, and 31 are writable fields and must be written with 0. Otherwise, operation is undefined.

7–26 Add a new Figure 7–16. The subsequent figures are renumbered accordingly:

*Figure 7–16. HPIC Register—C6000 Reference View*

| 31 16 |
|---|
| reserved |
| CR,+0 |

| 15 | 14 8 | 7 | 6 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| rsvd | rsvd | rsvd | rsvd | FETCH | HRDY | HINT | DSPINT | HWOB |
| CRW,+0 | CR,+0 | CRW,+0 | CR,+0 | CR,+0 | CR,+1 | CRW,+0 | CRW,+0 | CR,+0 |

| **Page:** | **Change or Add:** |
|---|---|
| 7–26 | Add a new section 7.5.3, Figure 7–17, and Table 7–9. The subsequent sections, figures, and tables are renumbered accordingly: |

### 7.5.3 TR Control Register (TRCTL) (C64x only)

The TR control register (TRCTL) controls how the HPI submits its requests to the EDMA subsystem. The TRCTL is shown in Figure 7–17 and summarized in Table 7–9.

*Figure 7–17. TR Control Register (TRCTL)*

| 31–9 | 8 | 7–6 | 5–4 | 3–0 |
|---|---|---|---|---|
| Reserved | TRSTALL | Reserved | PRI | PALLOC |
| RW,+0 | RW,+0 | RW,+x | RW,+10 | RW,+0100 |

*Table 7–9. TR Control Register (TRCTL) Bit Descriptions*

| Bit | Description | Section |
|---|---|---|
| PALLOC | Controls the total number of outstanding requests that can be submitted by the HPI to the EDMA | 7.5.3 |
| PRI | Controls the priority queue level that HPI requests are submitted to. | 7.5.3 |
| TRSTALL | Forces the HPI to stall all HPI requests to EDMA<br>This bit allows safe changing of the PALLOC and PRI fields.<br>TRSTALL=0: Allows HPI requests to be submitted to EDMA<br>TRSTALL=1: Halts the creation of new HPI requests to EDMA | 7.5.3 |

To safely change the PALLOC or PRI bits in TRCTL, the TRSTALL bit needs to be used to ensure a proper transition. The following procedure must be followed to change the PALLOC or PRI bits:

1) Set the TRSTALL bit to 1 to stop the HPI from submitting TR requests on the current PRI level. In the same write, the desired new PALLOC and PRI fields may be specified.
2) Clear all EDMA event enables (EER) corresponding to both old and new PRI levels to stop EDMA from submitting TR requests on both PRI levels. Do not manually submit additional events via the EDMA.
3) Do not submit new QDMA requests on either old or new PRI level.
4) Stop L2 cache misses on either old or new PRI level. This can be done by forcing program execution or data accesses in internal memory. Another way is to have the CPU executing a tight loop that does not cause additional cache misses.
5) Poll the appropriate PQ bits in PQSR until both queues are empty (see section 6.17.1).
6) Clear the TRSTALL bit to 0 to allow the HPI to continue normal operation.

Requestors are halted on the old HPI PRI level so that memory ordering can be preserved. In this case, all pending requests corresponding to the old PRI level must be let to complete before HPI is released from stall state.

| Page: | Change or Add: |
|---|---|

Requestors are halted on the new PRI level to ensure that at no time can the sum of all requestor allocations exceed the queue length. By halting all requestors at a given level, the user can be free to modify the queue allocation counters of each requestor.

7–27 Change the paragraph in section 7.5.4:

**The host can interrupt the CPU by writing to one of the DSPINT bits in the HPIC. In order for the CPU to receive DSPINT correctly, the host must only write one but not both of the DSPINT bits in HPIC register. The DSPINT bit is tied directly to the internal DSPINT signal. By writing DSPINT = 1 when DSPINT = 0, the host causes a low-to-high transition on the DSPINT signal.** If the user programs the selection of the DSPINT interrupt with interrupt selector, the CPU detects the transition of DSPINT as an interrupt condition. Unlike a host write, a CPU write of DSPINT = 1 when DSPINT = 0 has no effect. The CPU can clear the DSPINT bits by writing a 1 to DSPINT when DSPINT = 1. **Writing DSPINT = 0 (in HPIC) via the host or the CPU does not affect either the DSPINT bit or signal in any case.**

7–27 Change the first paragraph in section 7.5.5:

The CPU can send an active interrupt condition on the $\overline{\text{HINT}}$ signal by writing to the HINT bit in the HPIC. The HINT bit is inverted and tied directly to the $\overline{\text{HINT}}$ pin. The CPU can set $\overline{\text{HINT}}$ active by writing HINT = 1. The host can clear the $\overline{\text{HINT}}$ to inactive by writing a 1 to HINT. **Writing HINT = 0 (in HPIC) via the host or the CPU does not affect either the HINT bit or the $\overline{\text{HINT}}$ signal.**

7–32 Change the Value During Access column in Table 7–14:

*Table 7–14. Data Read Access in Fixed Address Mode for HPI32*

| | Value During Access | | | | Value After Access | | |
|---|---|---|---|---|---|---|---|
| **Event** | **HD** | **HR/$\overline{\text{W}}$** | **HCNTL[1:0]** | **$\overline{\text{HRDY}}$** | **HPIC** | **HPIA** | **HPID** |
| Host reads HPIC Data not ready | ???????? | 1 | **00** | 1 | 00000000 | 80001234 | ???????? |
| Host writes HPID Data ready | 789ABCDE | **0** | 11 | 0 | 00080008 | 80001234 | 789ABCDE |

**Note:** The “?” in this table indicate the value is unknown.

7–35 Change the title of Tables 7–18 and 7–19:

*Table 7–18 16-Bit Data Write Access to HPI in Fixed Address Mode: HWOB = 1†*

*Table 7–19. 16-Bit Data Write Access to HPI in Fixed Address Mode: HWOB = 0†*

| **Page:** | **Change or Add:** |
|---|---|

7–35 Add a new table after Table 7–19. The subsequent tables are renumbered accordingly:

*Table 7–20. 32-Bit Data Write Access to HPI in Fixed Address Mode: HWOB = 1*†

| | Value During Access | | | | | | Value After Access | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Event** | **HD** | **$\overline{\text{HBE}}$[1:0]** | **HR/$\overline{\text{W}}$** | **HCNTL[1:0]** | **$\overline{\text{HRDY}}$** | **HHWIL** | **HPIC** | **HPIA** | **HPID** | **Location 80001234** |
| Host writes HPID 1st halfword | 5566 | 00 | 0 | 11 | 1 | 0 | 00010001 | 80001234 | ???????? | 00000000 |
| Waiting for previous access to complete | | | | | | | | | | |
| Host writes HPID 1st halfword | 5566 | 00 | 0 | 11 | 0 | 0 | 00090009 | 80001234 | ????5566 | 00000000 |
| Host writes HPID 2nd halfword | wxyz | 00 | 0 | 11 | 0 | 1 | 00090009 | 80001234 | wxyz5566 | 00000000 |
| Waiting for access to complete | ???? | ?? | ? | ?? | 1 | ? | 00010001 | 80001234 | wxyz5566 | wxyz5566 |

† For C620x/C670x HPI, wxyz represents a “don’t care” value on the HD pins. The $\overline{\text{HBE}}$[1:0] value indicates that only 16-bit is transferred. For C621x/C671x and C64x HPI, however, wxyz should be 0000 on the HD pins. The entire 32-bit word is transferred.

**Note:** The “?” in this table indicate the value is unknown.

7–41 Change the paragraph in section 7.7:

All C621x/C671x HPI transfers are placed in the high priority transfer queue, Q1. **All C64x HPI transfers can be programmed to any of the four priority levels, with the medium priority level set as default**. Refer to section 6.17, *Resource Arbitration and Priority Processing*, for details on transfer priority.

8–6 Add a footnote in Table 8–2 at the XHOLD and XHOLDA signals:

*Table 8–2. Signal State for Disabled Host Port*

| **XBUS Signal** | **I/O Port Mode (I/O/Z)** | **External Connection** |
|---|---|---|
| XHOLD† | I/O/Z | Pull down |
| XHOLDA† | I/O/Z | Pull down |

† Internal arbitration should be enabled, such that the DSP is the master of the bus when not using the host port. See section 8.6 for more details.

| **Page:** | **Change or Add:** |
|---|---|

8–9 Change the Description of XFRAT in Table 8–5:

*Table 8–5. Expansion Bus Global Control Register (XBGC) Field Description*

| Field | Description | Section |
|---|---|---|
| XFRAT | FIFO clock rate<br>XFRAT = 00: XFCLK = 1/8 CPU clock rate<br>XFRAT = 01: XFCLK = 1/6 CPU clock rate<br>XFRAT = 10: XFCLK = 1/4 CPU clock rate<br>XFRAT = 11: XFCLK = 1/2 CPU clock rate<br>The FIFO clock setting cannot be changed while a DMA request to XCE space is active.<br>**The XFCLK should be disabled before changing XFRAT field. There is no delay required between enabling/disabling XFCLK and changing the XFRAT field.** | 8.4.2 |

8–12 Add a paragraph in section 8.4:

Figure 8–5 illustrates how to interface four 8-bit FIFOs to the I/O port (memory map for this case is described in Table 8–8). Figure 8–6 is an example of an interface between two 16-bit FIFOs and the I/O port.

**The $\overline{\text{XOE}}$, $\overline{\text{XRE}}$, $\overline{\text{XWE}}$, and $\overline{\text{XCEn}}$ signals are not tri-stated while the DSP releases control of the XBUS.**

8–14 Add a paragraph below the Notes in section 8.4.1:

An access to a section of memory that does not return a ready indication is not allowed. This includes accesses to XBUS I/O asynchronous spaces with XRDY pulled inactive or left floating on the device. Possible requestors are: programmed DMA channels or HPI/PCI/XBUS host mastering via the auxiliary DMA. This type of access can create a stall indefinitely.

8–14 Delete the third paragraph in section 8.4.2: The $\overline{\text{XOE}}$, $\overline{\text{XRE}}$, $\overline{\text{XWE}}$, and $\overline{\text{XCEn}}$ signals are not tri-stated while the DSP releases control of the XBUS.

8–24 Change the last paragraph in section 8.5.1.2:

This register is used when the host port operates either in synchronous or asynchronous mode. The DSP does not have access to the XBISA register content. Burst transfers in the synchronous host-port mode are always expected to occur with autoincrement (AINC bit should be **cleared to 0**). In autoincrement mode (AINC = 0), operation is undefined if an external host attempts to access the last 2 word locations in the **internal program/data RAM. This is because the DSP tries to prefetch data from reserved locations. Operation is also undefined if an external host attempts to cross a block boundary in a single DMA transfer. See Chapter 2 for details.**

| Page: | Change or Add: |
|---|---|
| 8–26 | Change the Descriptions of DSPINT and INTSRC in Table 8–16: |

*Table 8–16. Expansion Bus Host Port Interface Control Register (XBHC) Description*

| Field | Description |
|---|---|
| DSPINT | The expansion bus to DSP interrupt (set either by the external host or the completion of a master transfer) is cleared when this bit is set. The DSPINT bit must be manually cleared before another one can be set. |
| INTSRC | The XBUS host port interrupt can be caused either by DSPINT bit or by XFRCT counter. The INTSRC selects interrupt source between DSPINT and XFRCT counter.<br>INTSRC=0: **interrupt source is DSPINT bit of the XBISA. When a zero is written to INTSRC, the DSPINT of the XBISA is copied to the DSPINT bit of the XBHC.**<br>INTSRC=1: interrupt is generated at the completion of the master transfer initiated by writing to the START bit-field. |

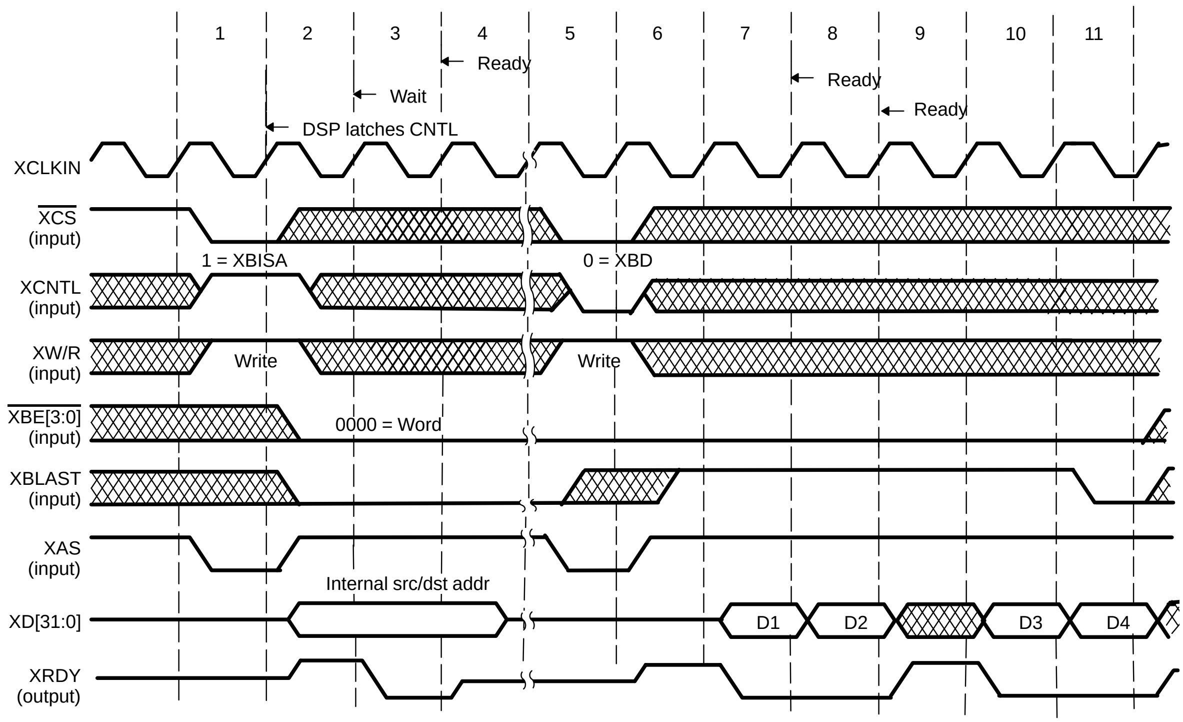
8–36 Add a paragraph before the third paragraph in section 8.5.2.2:

Initial access made to the expansion bus in host slave mode should be done in the order indicated above. After reset, the first access from the host should be an XBISA write followed by an XBD read/write. Undefined operation may occur if an XBISA read or an XBD read/write occurs before an XBISA write.

To read/write from the DSP memory space, the host must follow the following sequence:

| Page: | Change or Add: |
|---|---|
| 8–38 | Change Figure 8–22 to show the XRDY signal in high impedance (*not* high) in clock cycle four and five. The DSP should start driving the XRDY signal during the sixth clock cycle. |

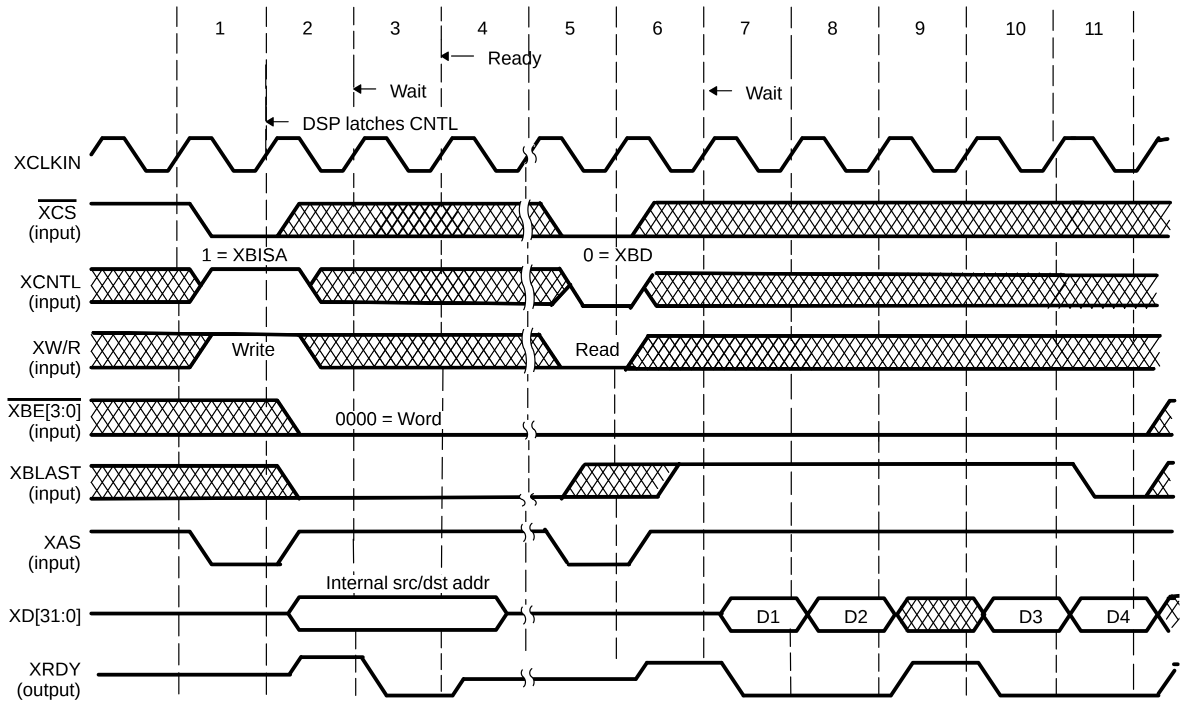
*Figure 8–22. Expansion Bus Master Writes a Burst of Data to the DSP*



| **Page:** | **Change or Add:** |
|---|---|

8–40 Change Figure 8–23 to show the XRDY signal in high impedance (*not* high) in clock cycle four and five. The DSP should start driving the XRDY signal during the sixth clock cycle.

*Figure 8–23. The Bus Master Reads a Burst of Data From the DSP*




8–43 Add a paragraph before the third paragraph in section 8.5.3:

> Initial access made to the expansion bus in host slave mode should be done in the order indicated above. After reset, the first access from the host should be an XBISA write followed by an XBD read/write. Undefined operation may occur if an XBISA read or an XBD read/write occurs before an XBISA write.
>
> If the XBUS host port is configured to operate in asynchronous mode, the $\overline{\text{XCS}}$ signal is used for four purposes:

8–43 Change the fourth paragraph in section 8.5.3:

The XRDY signal of the DSP functions differently than the C6201 HPI READY signal. The XRDY signal indicates normally not ready condition (active low READY signal is internally OR-ed with $\overline{\text{XCS}}$ signal in order to obtain XRDY). **XRDY should be polled during reads/writes to/from the XBISA or XBD.**

| Page: | Change or Add: |
|---|---|
| 8–44 | Add a new section 8.5.4: |

### 8.5.4 Special Circumstance of XBUS Host Memory Accesses

When the XBUS host port executes a read from the DSPs memory space, it does so by performing burst prefetches of 3 words. This results in the DMA auxiliary channel reading 3 higher-word addresses that you may not have explicitly requested. This occurs only under the following situations:

- ❑ An external master performs an autoincremented read from the XBUS configured for host slave mode (both synchronous or asynchronous).
- ❑ The XBUS configured as the master in synchronous host port mode writes from the DSP to the external space via the XBUS.

The accesses described above can cause the following undesired operations:

1) Accesses to undesired CE spaces:
   - When reading the top 3 words of EMIF CE0, the resulting prefetches can cause an inadvertent access to CE1 that may cause an undesired read to a device or a stall if the inadvertent access is to an asynchronous memory space with ARDY left floating or pulled inactive (not-ready).
   - The above example also applies to CE2 with the resulting prefetches possibly causing an inadvertent access to CE3 (see section 11.3, *Memory Map*).

   Associated design tip: If not using ARDY or XRDY, always pull to the ready state to avoid stalls. If you always want to detect bad software setups, always pull to the not-ready state to detect system stalls.

2) Unintended port crossings or illegal accesses to a reserved location:
   - When reading the top 3 words of EMIF CE1, the access can cross into either program memory (PMEM) block 0 when in Map 0 or to the internal peripheral bus (PBus) region storing EMIF control registers when in Map 1. This is an illegal port crossing.
   - When reading the top 3 words of EMIF CE3, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of PMEM block 0, the access can cross into PMEM block 1. This is an illegal port crossing.
   - When reading the top 3 words of PMEM block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of data memory (DMEM) block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading anywhere in the PBus space, you may prefetch ahead to three undesired control registers. This can cause an illegal access when accessing a reserved register address. If the register access has side effects (like reading the McBSP DRR, clearing RRDY), then you may inadvertently cause these side effects.

**Page:** **Change or Add:**

Note: A restriction does NOT exist when crossing between DMEM block 0 and block 1 because they both use the same DMA port.

Associated design tips:

- When reading internal peripheral registers:
  - For reads from an external master, use fixed-mode addressing. As a broader statement, it is good practice to use fixed mode also when writing to peripheral registers as sometimes there are gaps between them.
  - Do not use the XBUS host port configured in synchronous master mode to directly copy peripheral register values to external slaves. This is an atypical operation. If you must do so, copy the register data to an internal space with the CPU and then copy those internal locations to external space.
- When reading the top 3 locations of an EMIF CEx, internal program block or DMEM block 1, use fixed-mode addressing. Note this procedure does not have to be followed when accessing the top 3 words of DMEM block 0, this is because DMEM block 0 and block 1 are in the same DMA port.

| **Page:** | **Change or Add:** |
|---|---|
| 9–2 | Change the following PCI features in section 9.1: |

- ❑ Conforms to power management interface specification revision1.1 **(C6205 only)**
- ❑ DSP power control via software **(C6205 only)**
- ❑ Peripheral power control via software **(C6205 only)**
- ❑ Software-controlled assertion of PME from D0, D1, D2, $D3_{hot}$ **(C6205 only)**
- ❑ Hardware-controlled assertion of PME on power wakeup active from $D3_{cold}$. Optional hardware-controlled assertion of PME from D0, D1, D2, $D3_{hot}$. **(C6205 only)**
- ❑ Supports D0, D1, D2, $D3_{hot}$, $D3_{cold}$ power management modes **(C6205 only)**
- ❑ Implements PCI power management control status register “sticky” bits from logic powered by $3.3V_{aux}$ **(C6205 only)**

9–2 Delete the following PCI feature in section 9.1:

- ❑ 5-V or 3.3-V input signaling, 3.3-V output signaling

9–5 Add a row to Table 9–1:

*Table 9–1. Differences Between the C62x/C67x and C64x PCI*

| **Features** | **C62x/C67x PCI** | **C64x PCI** | **Section** |
|---|---|---|---|
| FIFO depth | 8 words | 16 words | N/A |

9–12 Change the Description of INTSRC in Table 9–5:

*Table 9–5 Host Status Register (HSR) Bit Field Description*

| **Bits** | **Name** | **Reset Source** | **Description** |
|---|---|---|---|
| 0 | INTSRC | $\overline{\text{PRST}}$ | PCI IRQ source active since last HSR clear. This bit, when 1, indicates that the DSP asserted the $\overline{\text{PINTA}}$ interrupt by writing the INTREQ bit in the RSTSRC register, and the INTAM bit in the HSR was a 0.<br>This bit can be cleared by the PCI Host by writing a 1 to this bit. This will also negate the $\overline{\text{PINTA}}$ signal.<br>Reads<br>INTSRC = 0: $\overline{\text{PINTA}}$ was not asserted after last clear<br>INTSRC = 1: $\overline{\text{PINTA}}$ was asserted after last clear<br>Writes<br>INTSRC = 0: no affect<br>INTSRC = 1: deassert $\overline{\text{PINTA}}$. **Note that this does not enable future interrupts. INTRST in RSTSRC must also be set to allow future interrupts. See section 9.10.3.** |

| **Page:** | **Change or Add:** |
|---|---|
| 9–14 | Add a new section 9.3.3, Figure 9–7, and Table 9–8. The subsequent sections, figures, and tables are renumbered accordingly: |

### 9.3.3 TR Control Register (TRCTL) (C64x only)

The TR control register (TRCTL) controls how the PCI submits its requests to the EDMA subsystem. The TRCTL is shown in Figure 9–7 and summarized in Table 9–8.

*Figure 9–7. TR Control Register (TRCTL)*

| 31–9 | 8 | 7–6 | 5–4 | 3–0 |
|---|---|---|---|---|
| Reserved | TRSTALL | Reserved | PRI | PALLOC |
| RW,+0 | RW,+0 | RW,+x | RW,+10 | RW,+0100 |

*Table 9–8. TR Control Register (TRCTL) Bit Descriptions*

| Bit | Description | Section |
|---|---|---|
| PALLOC | Controls the total number of outstanding requests that can be submitted by the PCI to the EDMA | 9.3.3 |
| PRI | Controls the priority queue level that PCI requests are submitted to. | 9.3.3 |
| TRSTALL | Forces the PCI to stall all PCI requests to EDMA<br>This bit allows safe changing of the PALLOC and PRI fields.<br>TRSTALL=0: Allows PCI requests to be submitted to EDMA<br>TRSTALL=1: Halts the creation of new PCI requests to EDMA | 9.3.3 |

To safely change the PALLOC or PRI bits in TRCTL, the TRSTALL bit needs to be used to ensure a proper transition. The following procedure must be followed to change the PALLOC or PRI bits:

1) Set the TRSTALL bit to 1 to stop the PCI from submitting TR requests on the current PRI level. In the same write, the desired new PALLOC and PRI fields may be specified.
2) Clear all EDMA event enables (EER) corresponding to both old and new PRI levels to stop EDMA from submitting TR requests on both PRI levels. Do not manually submit additional events via the EDMA.
3) Do not submit new QDMA requests on either old or new PRI level.
4) Stop L2 cache misses on either old or new PRI level. This can be done by forcing program execution or data accesses in internal memory. Another way is to have the CPU executing a tight loop that does not cause additional cache misses.
5) Poll the appropriate PQ bits in PQSR until both queues are empty (see section 6.17.1).
6) Clear the TRSTALL bit to 0 to allow the PCI to continue normal operation.

Requestors are halted on the old PCI PRI level so that memory ordering can be preserved. In this case, all pending requests corresponding to the old PRI level must be let to complete before PCI is released from stall state.

| Page: | Change or Add: |
|---|---|

Requestors are halted on the new PRI level to ensure that at no time can the sum of all requestor allocations exceed the queue length. By halting all requestors at a given level, the user can be free to modify the queue allocation counters of each requestor.

9–15 Add a TRCTL row and footnote to Table 9–8:

*Table 9–8. PCI Memory-Mapped Peripheral Registers*

| DSP Data Space Address | | Register Accessed | | |
|---|---|---|---|---|
| **C62x/C67x** | **C64x** | **Reads** | **Writes** | **Section** |
| 01A8 0008h | 01C2 0008h | EECTL | EECTL | 9.13.4 |
| — | **01C3 0000h** | **TRCTL**‡ | **TRCTL**‡ | **9.3.3** |

† HALT register applies to C62x/C67x only.
‡ **TRCTL register applies to C64x only.**

9–23 Change the paragraph in section 9.9.1:

The DSP master address register (DSPMA) contains the DSP's address for the location of destination data for DSP master reads, or the address location of source data for DSP master writes. The register also contains bits to control the address modification. **DSPMA is doubleword aligned on C64x and word aligned on C6205.** The DSPMA is shown in Figure 9–9 and summarized in Table 9–9.

9–23 Change the Description of AINC (bit 1) in Table 9–9:

*Table 9–9. DSP Master Address Register (DSPMA) Bit Field Description*

| Bits | Name | Reset Source | Description |
|---|---|---|---|
| 1 | AINC | $\overline{\text{RESET}}$<br>WARM | Autoincrement mode of DSP master address<br>AINC = 0: Autoincrement of ADDRMA enabled<br>AINC = 1: ADDRMA will not autoincrement<br>**Autoincrement only affects the lower 24 bits of DSPMA. As a result, autoincrement does not cross 16MB boundaries and will wrap around if incrementing past the boundary.** |
| 31:2 | ADDRMA | $\overline{\text{RESET}}$<br>WARM | DSP's word address for PCI master transactions |

9–24 Change the paragraph in section 9.9.2:

**The PCI master address register (PCIMA) contains the PCI word address.** For DSP master reads, PCIMA contains the source address. For DSP master writes, the PCIMA contains the destination address. The PCIMA is shown in Figure 9–10 and summarized in Table 9–10.

| **Page:** | **Change or Add:** |
|---|---|

9–24 Change the Description of ADDRMA in Table 9–10:

*Table 9–10. PCI Master Address Register (PCIMA) Bit Field Description*

| Bits | Name | Reset Source | Description |
|---|---|---|---|
| 31:2 | ADDRMA | $\overline{\text{RESET}}$ WARM | **PCI word address for PCI master transactions.** |

9–25 Change Figure 9–11. Add a footnote.

*Figure 9–11. PCI Master Control Register (PCIMC)*

| 31–16 | 15–**4** | **3** | **2**–0 |
|---|---|---|---|
| CNT | Reserved | **Rsvd**† | START |
| RW, +0000h | R, +0 | **RW, +0** | **RW, +000** |

† This reserved bit must always be written with zero. Writing 1 to this bit results in undefined operation.

9–25 Change the Bits and the Description of START in Table 9–11:

*Table 9–11. PCI Master Control Register (PCIMC) Bit Field Description*

| Bits | Name | Reset Source | Description |
|---|---|---|---|
| **2:0** | START | $\overline{\text{RESET}}$ WARM $\overline{\text{PRST}}$ | Start the read or write master transaction<br>START = **000b**: Transaction not started/flush current transaction<br>START = **001b**: Start a master write transaction<br>START = **010b**: Start a master read transaction to prefetchable memory<br>START = **011b**: Start a master read transaction to nonprefetchable memory<br>**START = 100b: Start a configuration write**<br>**START = 101b: Start a configuration read**<br>**START = 110b: Start an I/O write**<br>**START = 111b: Start an I/O read**<br><br>START will return to **000b** when the transaction is complete.<br>**The START field must not be written/changed during active master transfer.**<br>If the PCI bus is reset during a transfer, the transfer will stop and the FIFOs will be flushed. (A CPU interrupt can be generated on a $\overline{\text{PRST}}$ transition.)<br>START will only get set if bits 31:16 ≠ 0000h |
| 31:16 | CNT | $\overline{\text{RESET}}$ WARM | Transfer Count . It specifies the number of bytes to transfer |

| Page: | Change or Add: |
|---|---|
| 9–28 | Change the third paragraph in section 9.9.8: |

For DSP master writes, the ADDRMA field in the DSPMA contains the **word-aligned (C62x/C67x) or doubleword-aligned (C64x) source (DSP) address.** If AINC = 0 in the DSPMA, the source address is autoincremented by 4 bytes after each internal data transfer. **The PCIMA contains the word-aligned destination address (PCI address).** An internal register keeps track of the PCI master address.

9–29 Change the third through sixth paragraphs in section 9.9.9:

For DSP master reads, the PCIMA contains the external PCI slave source address. The ADDRMA field in the DSPMA contains the **word-aligned (C62x/C67x) or doubleword-aligned (C64x)** destination address (DSP address). If AINC = 0 in the DSPMA, the destination address is autoincremented by 4 bytes after each internal data transfer.

A **master memory read** is initiated by enabling the START bits in the PCIMC. The PCI port performs a PCI bus request. Once a PCI bus request is granted, a PCI bus cycle is initiated. The type of cycle initiated depends on the number of bytes to be transferred and the cache line size. The following **master memory read** commands are supported:

- ❏ Memory read
- ❏ Memory read multiple
- ❏ Memory read line

The user can initiate two types of **memory reads,** based on the START bits in the PCIMC. Prefetchable reads **(START = 010b)** use the memory read multiple and memory read line commands for transfers greater than one word. A memory read command is used for transfers of one word.

Nonprefetchable **memory reads (START = 011b)** always use a memory read command. A transfer size of N words is broken up into N one-word read cycles on the PCI bus. Users should read from prefetchable memory whenever possible.

9–30 Add a new section 9.9.10:

### 9.9.10 DSP As System Host

In systems where the DSP is the host for the PCI bus, an external mechanism is required to enable the DSP's master bit (bit 2 of PCI command register). This bit is not accessible by the CPU, and the DSP cannot generate any cycles until the master bit is set. A CPLD or FPGA can be programmed to do the configuration write necessary to set the bit. Once this bit is set, the DSP is capable of generating the configuration cycles necessary to configure a PCI bus. However, the DSP is not capable of configuring itself. If this is necessary and there is no other device on the bus capable of performing the configuration (such as another DSP), then the external mechanism used to set the master bit must also configure the rest of the PCI configuration registers.

| Page: | Change or Add: |
|---|---|

9–34 Change Figure 9–17. Add a footnote.

*Figure 9–17. PCI interrupt Enable Register (PCIIEN)*

| 31 **13** | **12** | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsvd | **Rsvd**† | $\overline{\text{PRST}}$ | **Rsvd**† | EERDY | CFG ERR | CFG DONE | MASTER OK | PWRHL | PWRLH | HOSTSW | PCI MASTER | PCI TARGET | PWR MGMT |
| R, +0 | **RW,+0** | RW, +0 | **RW, +0** | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +0 | RW, +1 | RW, +0 | RW, +0 | RW, +0 |

† These reserved bits must always be written with zeros. Writing 1 to these bits result in undefined operation.

9–36 Change the Description of INTREQ (bit 3) and INTRST (bit 4) in Table 9–18:

*Table 9–18. DSP Reset Source/Status Register (RSTSRC) Bit Field Description*

| Bits | Name | Reset Source | Description |
|---|---|---|---|
| 3 | INTREQ | $\overline{\text{RESET}}$ WARM | Request a DSP-to-PCI interrupt when written with a 1. Causes assertion of $\overline{\text{PINTA}}$ if the INTAM bit in **HSR is 0.** Writes of 0 have no effect. Always reads as 0. |
| 4 | INTRST | $\overline{\text{RESET}}$ WARM | When a 1 is written to this bit, $\overline{\text{PINTA}}$ is deasserted **and the interrupt logic is reset to enable future interrupts. This bit must be asserted before another host interrupt can be generated.** Writes of 0 have no effect. Always reads as 0. |

9–41 Change the third paragraph in section 9.13:

For C62x/C67x, the state of the boot configuration pins EESZ[2:0] at power-on reset determines if a serial EEPROM is present, and if so, what size. See Chapter 11, *Boot Modes and Configuration*, for details on EESZ[2:0]. Table 9–20 summarizes the EEPROM sizes supported by the C62x/C67x. The C64x only supports 4K **bits** EERPOM, and EESZ[2:0] does not exist.

9–41 Change the column header in Table 9–20:

*Table 9–20. TMS320C62x/C67x EEPROM Size Support*

| EESZ[2:0] | EEPROM Size Supported on C62x/C67x (bits) |
|---|---|

9–45 Change the Description of EESZ (bits 5:3) in Table 9–25:

*Table 9–25. EEPROM Control Register (EECTL) Bit Field Description*

| Bits | Name | Reset Source | Description |
|---|---|---|---|
| 5:3 | EESZ | $\overline{\text{RESET}}$ | Indicates the state of the EESZ[2:0] pins at power-on reset<br>EESZ = 000b: No EEPROM<br>EESZ = 001b: 1K **bits (C6205 only)**<br>EESZ = 010b: 2K **bits (C6205 only)**<br>EESZ = 011b: 4K **bits**<br>EESZ = 100b: 16K **bits (C6205 only)** |

| Page: | Change or Add: |
|---|---|
| 9–60 | Change the first paragraph in section 9.16: |

This section discusses the PCI configuration registers in detail. These registers are only accessible from the external host PCI. Table 9–28 **to Table 9–50** summarize the bit fields in the PCI configuration registers. Table 9–51 lists the power management states in the PWRDATA register. See section 9.3.1.

9–61 Add a note to Table 9–34:

*Table 9–34. Cache Line Size Register Bit Field Description*

| Bits | Access | Default | Description |
|---|---|---|---|
| 7:0 | R/W | 00h | Cache Line Size |

Note: This field only accepts power-of-2 cache line sizes. If a cache line size other than a power of 2 is written, 0 will be written to this field.

9–62 Change Table 9–37:

*Table 9–37 Base 0 Address Register Bit Field Description*

| Bits | Access | Default | Host Read† | Description |
|---|---|---|---|---|
| 31:0 | R/W | 0000<br>0008 | FFC0<br>0008 | Mask for 4 Mbytes, prefetchable memory |

† The host reads this value after writing FFFF FFFFh, used as mask bits to determine the size of the base address region during register initialization.

9–62 Change Table 9–38:

*Table 9–38 Base 2 Address Register Bit Field Description*

| Bits | Access | Default | Host Read† | Description |
|---|---|---|---|---|
| 31:0 | R/W | 0000<br>0000 | FF80<br>0000 | Mask for 8 Mbytes, nonprefetchable memory |

† The host reads this value after writing FFFF FFFFh, used as mask bits to determine the size of the base address region during register initialization.

9–62 Change Table 9–39:

*Table 9–39 Base 1 Address Register Bit Field Description*

| Bits | Access | Default | Host Read† | Description |
|---|---|---|---|---|
| 31:0 | R/W | 0000<br>0001 | FFFF<br>FFF1 | Mask for 16 Bytes, I/O space |

† The host reads this value after writing FFFF FFFFh, used as mask bits to determine the size of the base address region during register initialization.

**Page:** **Change or Add:**

9–68 Add a new section 9.17:

## 9.17 C620x/C670x Special Circumstance of PCI-to-Memory Accesses

When the PCI executes a read from the DSPs memory space, it does so by performing burst prefetches of 3 words. This results in the DMA auxiliary channel reading 3 higher-word addresses that you may not have explicitly requested. This occurs only under the following situations:

- The PCI master performs prefetchable reads from an external PCI master.
- The C6205 PCI acts as the PCI master performing a prefetchable read from the DSPs address space to the PCI address.

These prefetchable reads can generate the following undesired operations:

1) Accesses to undesired CE spaces:
   - When reading the top 3 words of EMIF CE0, the resulting prefetches can cause an inadvertent access to CE1 that may cause an undesired read to a device or a stall if the inadvertent access is to an asynchronous memory space with ARDY left floating or pulled inactive (not-ready).
   - The above example also applies to CE2 with the resulting prefetches possibly causing an inadvertent access to CE3 (see section 11.3, *Memory Map*).

   Associated design tip: If not using ARDY, always pull to the ready state to avoid stalls. If you always want to detect bad software setups, always pull to the not-ready state to detect system stalls.

2) Unintended port crossings or illegal accesses to a reserved location:
   - When reading the top 3 words of EMIF CE1, the access can cross into either program memory (PMEM) block 0 when in Map 0 or to the internal peripheral bus (PBus) region storing EMIF control registers when in Map 1. This is an illegal port crossing.
   - When reading the top 3 words of EMIF CE3, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of PMEM block 0 the access can cross into PMEM block 1. This is an illegal port crossing.
   - When reading the top 3 words of PMEM block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading the top 3 words of data memory (DMEM) block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading anywhere in the PBus space, you may prefetch ahead to three undesired control registers. This can cause an illegal access when accessing a reserved register address. If the register access has side effects (like reading the McBSP DRR, clearing RRDY), then you may inadvertently cause these side effects.

**Page:** **Change or Add:**

Note: A restriction does NOT exist when crossing between DMEM block 0 and block 1 because they both use the same DMA port.

Associated design tips:

- When reading internal peripheral registers:
  - For reads from an external master, use fixed-mode addressing. As a broader statement, it is good practice to use fixed mode also when writing to peripheral registers as sometimes there are gaps between them.
  - On PCI, always use non-prefetchable reads of peripheral registers.
- When reading the top 3 locations of an EMIF CEx, internal program block or DMEM block 1, use fixed-mode addressing. Note this procedure does not have to be followed when accessing the top 3 words of DMEM block 0, this is because DMEM block 0 and block 1 are in the same DMA port.

| Page: | Change or Add: |
|---|---|
| 10–all | Chapter 10: change all occurrences of CExCTL, CE0CTL, CE1CTL, CE2CTL, CE3CTL, CExSEC, CE0SEC, CE1SEC, CE2SEC, and CE3SEC: |

| From | To |
|---|---|
| CExCTL | CECTLx |
| CE0CTL | CECTL0 |
| CE1CTL | CECTL1 |
| CE2CTL | CECTL2 |
| CE3CTL | CECTL3 |
| CExSEC | CESECx |
| CE0SEC | CESEC0 |
| CE1SEC | CESEC1 |
| CE2SEC | CESEC2 |
| CE3SEC | CESEC3 |

10–10 Change the footnote in Figure 10–7:

† See **Table 10–2** for ED, EA, $\overline{\text{CE}}$, and $\overline{\text{BE}}$ pins on EMIFA and EMIFB.

10–14 Change the paragraph in section 10.2:

Control of the EMIF and the memory interfaces it supports is maintained through memory-mapped registers within the EMIF. **The EMIF clock is needed to access these registers.** The memory-mapped registers are listed in Table 10–3.

| Page: | Change or Add: |
|---|---|
| 10–15 | Change the C621x/C671x bits 6 and 5 read/write fields and the footnotes in Figure 10–8: |

*Figure 10–8. EMIF Global Control Register (GBLCTL)*

**C621x/C671x:**

| 31–16 |
|---|
| Rsv |
| R, +0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsv | Rsv† | Rsv† | Rsv† | BUSREQ | ARDY | HOLD | HOLDA | NO HOLD | Rsv | **EKEN‡** | CLK1EN | CLK2EN | Rsv | Rsv | Rsv |
| R,+0 | RW,+0 | RW,+1 | RW,+1 | R, +0 | R, +x | R, +x | R, +x | RW, +0 | **RW,** +1 | **RW,** +1 | RW, +1 | RW, +1 | R, +0 | R, +0 | R, +0 |

† The reserved bit fields should always be written with their default values when modifying the GBLCTL. **Writing other values to these fields may cause improper operation.**
‡ **For all C621x/C671x devices, except C6713, C6711C, and C6712C, this bit field is reserved, RW, +1**.

10–16 Change the Device Group of CLK4EN and CLK6EN in Table 10–4:

*Table 10–4. EMIF Global Control Register (GBLCTL) Field Descriptions*

| Field | Description | Apply to Device Group† |
|---|---|---|
| CLK4EN | CLKOUT4 enable<br>CLK4EN = 0, CLKOUT4 held high<br>CLK4EN = 1, CLKOUT4 enabled to clock<br><br>For C64x, CLKOUT4 pin is muxed with GP1 pin. Upon exiting reset, CLKOUT4 is enabled and clocking. After reset, CLKOUT4 maybe be configured as GP1 via the GPIO enable register GPEN. | **4** |
| CLK6EN | CLKOUT6 enable<br>CLK6EN = 0, CLKOUT6 held high<br>CLK6EN = 1, CLKOUT6 enabled to clock<br><br>For C64x, CLKOUT6 pin is muxed with GP2 pin. Upon exiting reset, CLKOUT6 is enabled and clocking. After reset, CLKOUT6 maybe be configured as GP2 via the GPIO enable register GPEN. | **4** |

**Page:** **Change or Add:**

10–17 Add EKEN, a Group 6 to the footnote, and two new footnotes. Change the Group 3 footnote and the Description of EK1EN, EK2EN, EK1HZ, EK2HZ, EK2RATE, NOHOLD, and ARDY in Table 10–4:

*Table 10–4. EMIF Global Control Register (GBLCTL) Field Descriptions*

| Field | Description | Apply to Device Group† |
|---|---|---|
| EKEN | ECLKOUT enable<br>EKEN = 0, ECLKOUT held low<br>EKEN = 1, ECLKOUT enabled to clock (default) | 6 |
| EK1EN‡ | ECLKOUT1 enable<br>**EK1EN** = 0, ECLKOUT1 held low<br>**EK1EN** = 1, ECLKOUT1 enabled to clock | 4,5 |
| EK2EN‡ | ECLKOUT2 enable<br>**EK2EN** = 0, ECLKOUT2 held low<br>**EK2EN** = 1, ECLKOUT2 enabled to clock | 4,5 |
| EK1HZ‡ | ECLKOUT1 High-Z control<br>**EK1HZ** = 0, ECLKOUT1 continues clocking during Hold **(if EK1EN = 1)**<br>**EK1HZ** = 1, ECLKOUT1 High-Z during Hold | 4,5 |
| EK2HZ‡ | ECLKOUT2 High-Z control<br>**EK2HZ** = 0, ECLKOUT2 continues clocking during Hold **(if EK2EN = 1)**<br>**EK2HZ** = 1, ECLKOUT2 High-Z during Hold | 4,5 |
| EK2RATE§ | ECLKOUT2 Rate. ECLKOUT2 runs at:<br>**EK2RATE** = 00, 1x EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate<br>**EK2RATE** = 01, 1/2x EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate<br>**EK2RATE** = 10, 1/4x EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate | 4,5 |
| NOHOLD | External **NOHOLD enable**<br>NOHOLD = 0: **no hold disabled. Hold requests via the HOLD input are acknowledged via the HOLDA output at the earliest possible time.**<br>NOHOLD = 1: **no hold enabled. Hold requests via the HOLD input are ignored.** | 1,2,3,4,5 |
| ARDY | ARDY = 0: ARDY input is low. External device not ready.<br>ARDY = 1: ARDY input is high. External device ready.<br>**For C64x only, valid ARDY bit reflects the true value of the ARDY input pin only during an asynchronous memory access, indicated by asynchronous CEx active.** | 1,2,3,4,5 |

† Group1 devices include: C6201/C6701.
Group 2 devices include: all C620x/C670x *except* C6201/C6701.
Group 3 devices include: C621x/C671x **(including C6713, C6711C, and C6712C)**.
Group 4 devices include: C64x EMIFA.
Group 5 devices include: C64x EMIFB.
Group 6 devices include: C6713, C6711C, and C6712C.
‡ **ECLKOUTx does not turn off/on glitch free via EKxEN or via EKxHZ. See section 10.14**.
§ **ECLKOUT2 rate should only be changed once during EMIF initialization from the default (1/4x) to either 1/2x or 1x**.

| **Page:** | **Change or Add:** |
|---|---|
| 10–19 | Change the footnote and the TA (bits15–14) read/write field in Figure 10–9: |

*Figure 10–9. TMS320C62x/C67x/C64x EMIF CE Space Control Register (CExCTL)*

**C621x/C671x/C64x:**

| 31–28 | 27–22 | 21–20 | 19–16 |
|---|---|---|---|
| Write setup | Write strobe | Write hold | Read setup |
| RW, +1111 | RW, +111111 | RW, +11 | RW, +1111 |

| 15–14 | 13–8 | 7–4 | 3 | 2–0 |
|---|---|---|---|---|
| TA | Read strobe | MTYPE | Write hold MSB‡ | Read hold |
| **RW, +11** | RW, +111111 | RW, +0010† | RW, +0 | RW, +011 |

† **For C64x,** MTYPE default value is RW,+0000.
‡ For C621x/C671x, this field is reserved. R,+0.

10–20 Change Table 10–5:

*Table 10–5. EMIF CE Space Control Registers (CExCTL) Field Descriptions*

| Field | Description |
|---|---|
| Read setup<br>Write setup | Setup width. Number of clock† cycles of setup time for address (EA), chip enable ($\overline{CE}$), and **byte enables ($\overline{BE}$)** before read strobe or write strobe falls. For asynchronous read accesses, this is also the setup time of $\overline{AOE}$ before $\overline{ARE}$ falls. |
| Read hold<br>**Write hold MSB**<br>**Write hold** | Hold width. Number of clock† cycles that address (EA) and **byte strobes ($\overline{BE}$)** are held after read strobe or write strobe rises. For asynchronous read accesses, this is also the hold time of $\overline{AOE}$ after $\overline{ARE}$ rising. **Write hold MSB is the most-significant bit of write hold (C64x only).** |
| TA | **Minimum Turn-Around time** (C621x/C671x/C64x only). Turn-around time controls the **minimum number** of ECLKOUT cycles between **a read followed by a write (same or different CE spaces), or between reads from different CE spaces. Applies only to asynchronous memory types.** |

| **Page:** | **Change or Add:** |
|---|---|
| 10–23 | Change the SDCSZ (bits 27–26) and INIT (bit 24) read/write fields in Figure 10–11: |

*Figure 10–11. EMIF SDRAM Control Register (SDCTL)*

**C621x/C671x:**

| 31 | 30 | 29 28 | 27 26 | 25 | 24 | 23 20 | 19 16 |
|---|---|---|---|---|---|---|---|
| Rsv | SDBSZ | SDRSZ | SDCSZ | RFEN | INIT | TRCD | TRP |
| R,+0 | RW, +0 | RW, +00 | RW, +**00** | RW, +1 | **W, +0** | RW, +0100 | RW, +1000 |

| 15 12 | 11 0 |
|---|---|
| TRC | Reserved |
| RW, +1111 | R, +0000 0000 0000 |

**C64x:**

| 31 | 30 | 29 28 | 27 26 | 25 | 24 | 23 20 | 19 16 |
|---|---|---|---|---|---|---|---|
| Rsv | SDBSZ | SDRSZ | SDCSZ | RFEN | INIT | TRCD | TRP |
| R,+0 | RW, +0 | RW, +00 | RW, +**00** | RW, +1 | **W, +0** | RW, +0100 | RW, +1000 |

| 15 12 | 11 1 | 0 |
|---|---|---|
| TRC | Reserved | SLFRFR† |
| RW, +1111 | R, +000 0000 0000 | RW, +0 |

| | |
|---|---|
| 10–24 | Change the Description of INIT and RFEN in Table 10–7: |

*Table 10–7. EMIF-to-SDRAM Control Register (SDCTL) Field Descriptions*

| Field | Description |
|---|---|
| INIT | Forces initialization of all SDRAM present<br>INIT = 0: no effect<br>INIT = 1: initialize SDRAM in each CE space configured for SDRAM. **The CPU should initialize all of the CE space control registers and SDRAM extension register before setting INIT = 1.**<br>**Reading this bit returns an undefined value.** |
| RFEN | Refresh enable<br>RFEN = 0: SDRAM refresh disabled<br>RFEN = 1: SDRAM refresh enabled<br>**If SDRAM is not used, be sure RFEN = 0; otherwise, BUSREQ may become asserted when SDRAM timer counts down to 0.** |

| Page: | Change or Add: |
|---|---|
| 10–25 | Delete the first paragraph below Table 10–7 that begins: The EMIF automatically clears the INIT field to zero... |
| 10–26 | Change the PERIOD (bits 11–0) read/write field in Figure 10–12: |

*Figure 10–12. EMIF SDRAM Timing Register (SDTIM)*

| 31–26 | 25–24 | 23–12 | 11–0 |
|---|---|---|---|
| Reserved | XRFR‡ | COUNTER | PERIOD |
| R, +0000 00 | R, +00†<br>RW,+00‡ | R, +0000 1000 0000†<br>R, +0101 1101 1100‡ | **RW, +0000 1000 0000†**<br>RW, +0101 1101 1100‡ |

† Applies to C620x/C670x
‡ Applies to C621x/C671x/C64x

10–27 Change the paragraph in section 10.2.5:

The SDRAM extension register of the C621x/C671x/C64x allows programming of many parameters of SDRAM. The programmability offers two distinct advantages. First, it allows an interface to a wide variety of SDRAMs and is not limited to a few configurations or speed characteristics. Second, the EMIF can maintain seamless data transfer from external SDRAM due to features like hidden precharge and multiple open banks. **It should be noted that the SDCTL register must be set after configuring the SDEXT register.** Figure 10–13 shows the SDRAM extension register and Table 10–9 discusses these parameters.

10–27 Change the WR2DEAC (bits 19–18) read/write field in Figure 10–13:

*Figure 10–13. TMS320C621x/C671x/C64x SDRAM Extension Register (SDEXT)*

| 31–21 | 20 | 19–18 | 17 | 16–15 | 14–12 | 11–10 | 9 | 8–7 | 6–5 | 4 | 3–1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsvd | WR2RD | WR2DEAC | WR2WR | R2WDQM | RD2WR | RD2DEAC | RD2RD | THZP | TWR | TRRD | TRAS | TCL |
| R, +0 | RW,+1 | **RW,+10†**<br>**RW,+01‡** | RW,+1 | RW,+11†<br>RW,+10‡ | RW,+101 | RW,+11 | RW,+1 | RW,+10 | RW,+01 | RW,+1 | RW,+111 | RW,+1 |

† Applies to C621x/C671x.
‡ Applies to C64x.

10–32 Change the last paragraph in section 10.3.3:

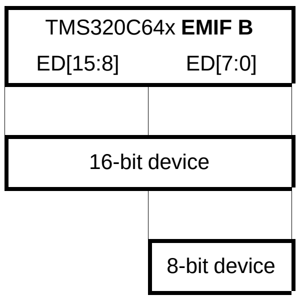
Figure 10–15 shows the byte lane used on C64x. The external memory is always right aligned to the ED[7:0] side of the bus. The endianness mode determines whether byte lane 0 (ED[7:0]) is accessed as byte address 0 (little endian) or as byte address N (big endian), where $2^N$ is memory width in bytes. **Similarly, byte lane N is addressed as either byte address 0 (big endian) or as byte address N (little endian).**

| Page: | Change or Add: |
|---|---|

10–33 Change the label in Figure 10–15:

*Figure 10–15. TMS320C64x Byte Alignment by Endianness*

**EMIFB (16-bit bus):**




10–34 Change the third paragraph in section 10.5:

The C621x/C671x/C64x EMIF allows programming of the addressing characteristics of the SDRAM, including the number of column address bits (page size), the row address bits (pages per bank), and banks (maximum number of pages that can be opened). **The maximum number of open pages is limited by the number of EMIF address registers.** Using this information, the C621x/C671x/C64x is able to open up to four pages of SDRAM simultaneously. The pages can all be in different banks of a single CE space, or distributed across multiple CE spaces. Only one page can be open per bank at a time. The C621x/C671x/C64x can interface to any SDRAM that has 8 to 10 column address pins, 11 to 13 row address pins, and two or four banks.

| **Page:** | **Change or Add:** |
|---|---|
| 10–38 | Change Table 10–15: |

*Table 10–15. TMS320C620x/C670x Compatible SDRAM*

| **SDRAM Size** | **B** | **W** | **D** | **Max Devices/ CE** | **Address-able Space (MBytes)** | | **Column Address** | **Row Address** | **Bank Select** | **Pre-charge** |
|---|---|---|---|---|---|---|---|---|---|---|
| 16M bit | 2 | ×8 | 1M | 4 | 8M | SDRAM | A8–A0 | A10–A0 | **BA0** | A10 |
| | | | | | | EMIF | EA10–EA2 | SDA10, EA11–EA2 | EA13 | SDA10 |
| | 2 | ×16 | 512K | 2 | 4M | SDRAM | A7–A0 | A10–A0 | **BA0** | A10 |
| | | | | | | EMIF | EA9–EA2 | SDA10, EA11–EA2 | EA13 | SDA10 |
| 64M bit | 4 | ×16 | 1M | 2 | 16M | SDRAM | A7–A0 | A11–A0 | **BA1–BA0** | A10 |
| | | | | | | EMIF | EA9–EA2 | **EA13,** SDA10, **EA11–EA2** | EA15–EA14 | SDA10 |
| | 4 | ×32 | 512K | 1 | 8M | SDRAM | A7–A0 | A10–A0 | **BA1–BA0** | A10 |
| | | | | | | EMIF | EA9–EA2 | SDA10, EA11–EA2 | EA14–EA13 | SDA10 |
| 128M bit (1) | 4 | ×32 | 1M | 1 | 16M | SDRAM | A7–A0 | A11–A0 | **BA1–BA0** | A10 |
| | | | | | | EMIF | EA9–EA2 | **EA13, SDA10, EA11–EA2** | EA15–EA14 | **SDA10** |
| | 4 | ×16 | 2M | 2 | 8M | SDRAM | A8–A0 | A11–A0 | BA1–BA0 | A10 |
| | | | | | | EMIF | EA10–EA2 | SDA10, EA11–EA2 | EA14–EA13 | SDA10 |
| 256M bit (1) | 4 | ×16 | 4M | 2 | 16M | SDRAM | A8–A0 | A12–A0 (3) | BA1–BA0 | A10 |
| | | | | | | EMIF | EA10–EA2 | SDA10, EA11–EA2 | EA14–EA13 | SDA10 |
| 512M bit (1) | 4 | ×16 | 8M | 2 | 16M | SDRAM | A9–A0 (2) | A12–A0 (3) | BA1–BA0 | A10 |
| | | | | | | EMIF | EA10–EA2 | SDA10, EA11–EA2 | EA14–EA13 | SDA10 |

**Legend:** B = Banks; W = Width; D = Depth

**Notes:**
1) Not all of the memory space will be used in larger memories. This is due to column and row address size limitations of the 620x/670x EMIF architecture.
2) SDRAM address A9 reflects EA11 during a $\overline{\text{RAS}}$ command. See Table 10–20 for more details.
3) SDRAM address A12 should follow the SDRAM datasheet guidelines (Typically this signal should be tied low).

| Page: | Change or Add: |
|---|---|

10–43 Change the paragraph in section 10.5.1:

### 10.5.1 SDRAM Initialization

After reset, none of the CE spaces are configured as SDRAM. The CPU should initialize all of the CE space control registers and the SDRAM extension register before performing SDRAM initialization by setting the INIT bit to 1. You should not write a 1 to the INIT bit, if SDRAM does not exist in the system.

The EMIF performs the following steps when INIT is set to 1:

1) Sends a DCAB command to all CE spaces configured as SDRAM.
2) Sends eight refresh commands.
3) Sends an MRS command to all CE spaces configured as SDRAM.

10–43 Delete the last paragraph in section 10.5.1: The DCAB cycle is performed immediately after reset, provided the $\overline{\text{HOLD}}$ input is not active (a host request). If $\overline{\text{HOLD}}$ is active, the DCAB command is not performed until the hold condition is removed. In this case the external requester should not attempt to access any SDRAM banks, unless it performs SDRAM initialization and control itself.

10–43 Add a new section 10.5.2. The subsequent sections are renumbered accordingly:

### 10.5.2 C620x/C670x Bootmode

If BOOTMODE[4:0] bits are set such that CE0 is configured for SDRAM, SDRAM initialization proceeds according to the steps listed in section 10.5.1 under the control of hardware, prior to the boot process. However, if $\overline{\text{HOLD}}$ is active, the DCAB command is not performed until the hold condition is removed. In this case, the external requester should not attempt to access any SDRAM banks, unless it performs SDRAM initialization and control itself. If other CE spaces besides CE0 are configured for SDRAM, and since CE0 is initialized with slower default timings following reset, SDRAM initialization should be performed by software.

10–59 Change the paragraph in section 10.5.6:

The C621x/C671x/C64x uses a mode register value of either 0032h or 0022h. The register value and description are shown in Figure 10–26 and summarized in Table 10–24. Both values program a default burst length of four words for both reads and writes. **The value programmed depends on the CAS latency parameter defined by the TCL field in the SDRAM extension register (SDEXT). If the CAS latency is three (TCL = 1), 0032h is written during the MRS cycle. If the CAS latency is two (TCL = 0), 0022h is written during the MRS cycle. Figure 10–27 shows the timing diagram during execution of the MRS command.**

10–64 Change the title of Figure 10–29:

*Figure 10–29. TMS320C621x/C671x**/C64x** SDRAM DEAC — Deactivate Single Bank*

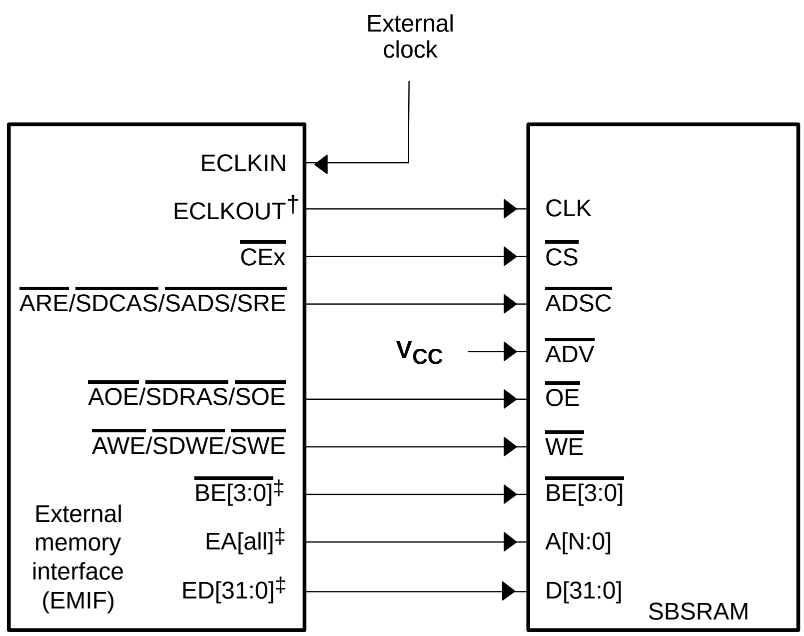
**Page:** **Change or Add:**

10–72 Change the first paragraph in section 10.6:

The C6000 EMIF interfaces directly to industry-standard synchronous burst SRAMs (SBSRAMs). This memory interface allows a high-speed memory interface without some of the limitations of SDRAM. Most notably, since SBSRAMs are SRAM devices, random accesses in the same direction can occur in a single cycle. Besides supporting the SBSRAM interface, the programmable synchronous interface on the C64x supports additional synchronous device interfaces. See **section 10.7** for details on the C64x interface with the other synchronous devices. This section discusses the SBSRAM interface on all the C6000 devices.

10–75 Change the input on $\overline{\text{ADV}}$ and the footnote in Figure 10–39:

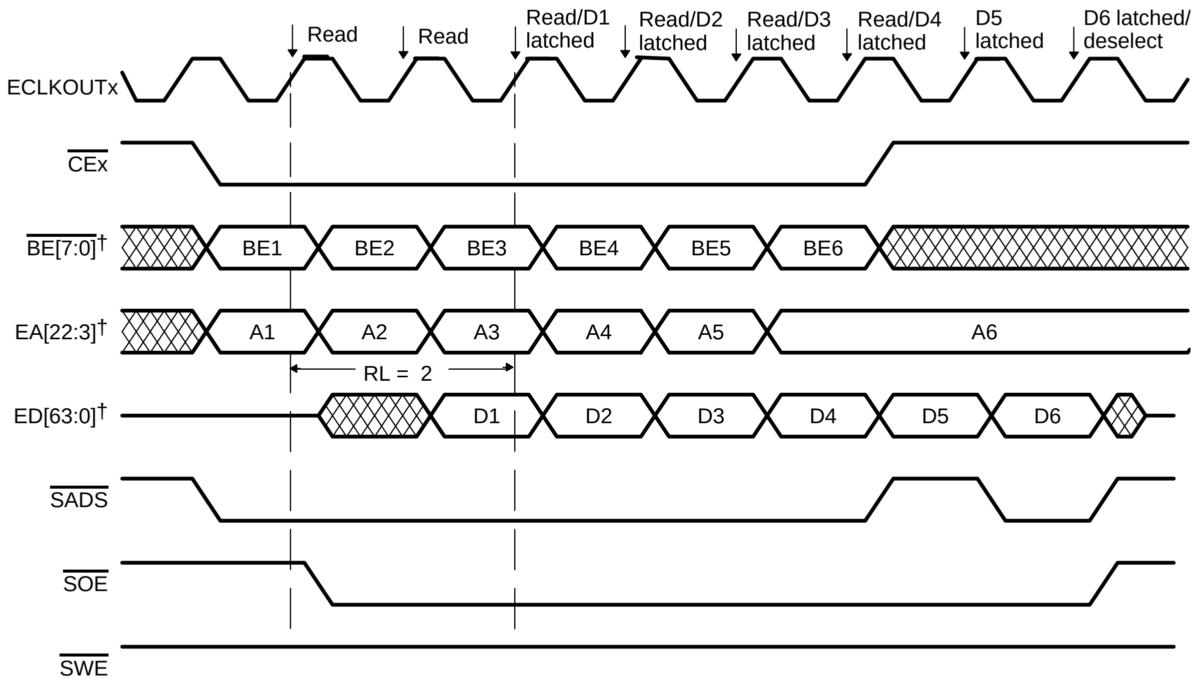
*Figure 10–39 TMS320C64x SBSRAM Interface*




† ECLKOUTx used is selected by the SNCCLK bit in the CExSEC register.
‡ For interface to a 64-bit data bus, BE[7:0], **EA[all],** and ED[63:0] are used.
For interface to a 16-bit data bus, BE[1:0], **EA[all],** and ED[15:0] are used.

| Page: | Change or Add: |
|---|---|
| 10–79 | Delete the PDT signal and change the footnotes in Figure 10–42: |

*Figure 10–42. TMS320C64x SBSRAM Six-Element Read*

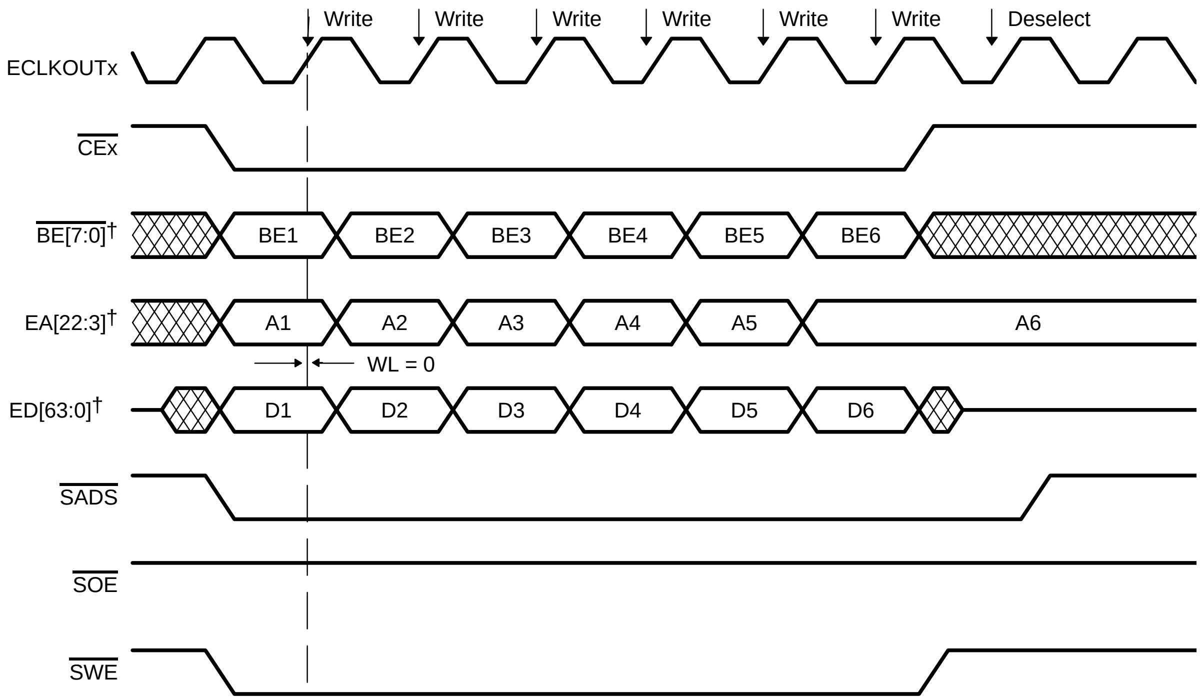



† For EMIFB, $\overline{BE[1:0]}$, EA[20:1], and ED[15:0], respectively, are used instead.

**Page:** **Change or Add:**

10–82 Delete the PDT signal and change the footnotes in Figure 10–45:

*Figure 10–45. TMS320C64x SBSRAM Six-Element Write*

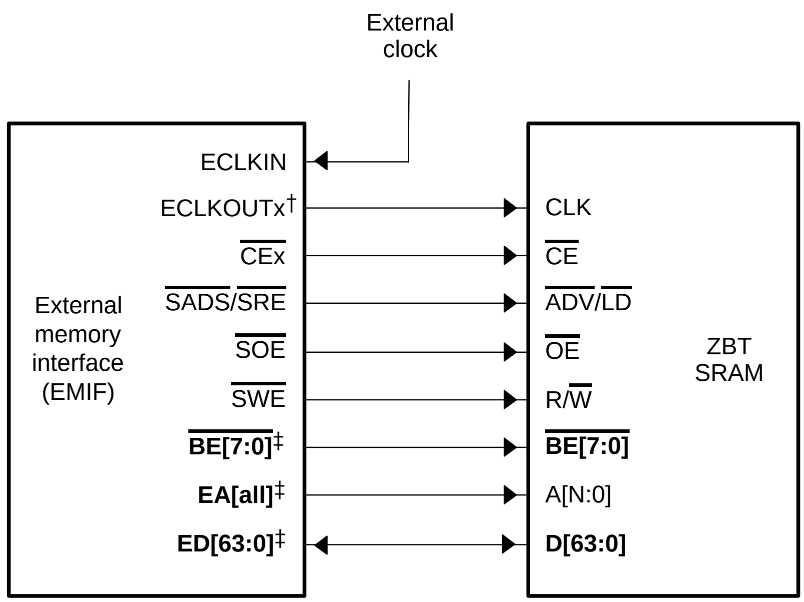



† For EMIFB, $\overline{BE[1:0]}$, EA[20:1], and ED[15:0], respectively, are used instead.

**Page:** **Change or Add:**

10–84 Change the signal names and footnote in Figure 10–46:

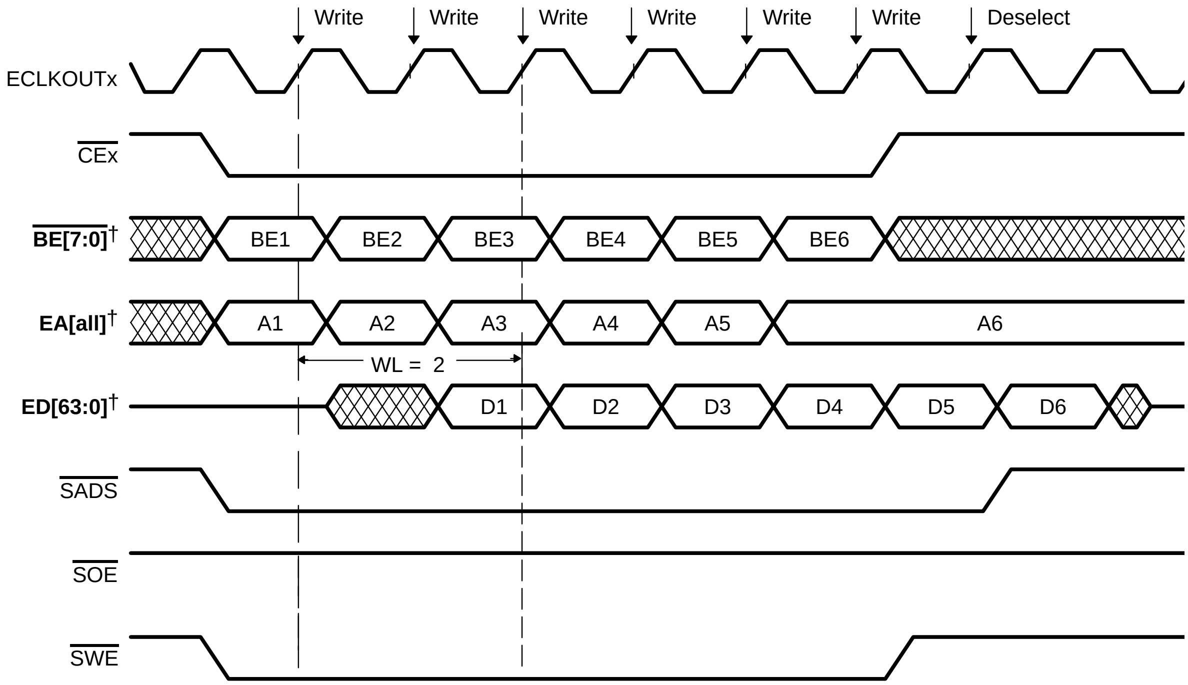
*Figure 10–46. TMS320C64x ZBT SRAM Interface*




† ECLKOUTx used is selected by the SNCCLK bit in the CExSEC register.
‡ **The MTYPE field selects the interface to be 8-, 16-, 32-, or 64-bits wide.**
**For 32-bit interface, BE[3:0], EA[all], and ED[31:0] are used**
**For 16-bit interface, BE[1:0], EA[all], and ED[15:0] are used**

| **Page:** | **Change or Add:** |
|---|---|
| 10–85 | Delete the PDT signal and change the signal names and footnotes in Figure 10–47: |

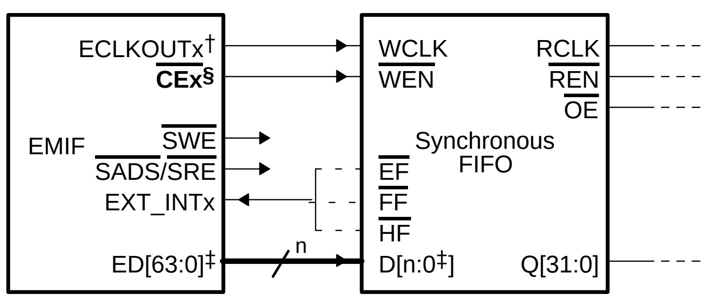
*Figure 10–47. TMS320C64x ZBT SRAM Six-Element Write*




† **Figure shows 64-bit interface. The MTYPE field selects the interface type to be 8-, 16-, 32-, or 64-bits wide.**
**For 32-bit interface, BE[3:0], EA[all], and ED[31:0] are used**
**For 16-bit interface, BE[1:0], EA[all], and ED[15:0] are used**

| | |
|---|---|
| 10–87 | Change the figure title, signal name, and footnote in Figure 10–50: |

*Figure 10–50. TMS320C64x Glueless Synchronous FIFO Write Interface*




† ECLKOUTx used is selected by the SNCCLK bit in the CExSEC register.
‡ The MTYPE field selects the interface to be 8-, 16-, 32-, or 64-bits wide. For EMIFB, only 8- and 16-bit interfaces are available. Therefore only ED[15:0] is used.
§ **Reads to CEx must not be performed in this interface, since reads cause CEx to go active, causing FIFO data contention.**

**Page:** **Change or Add:**

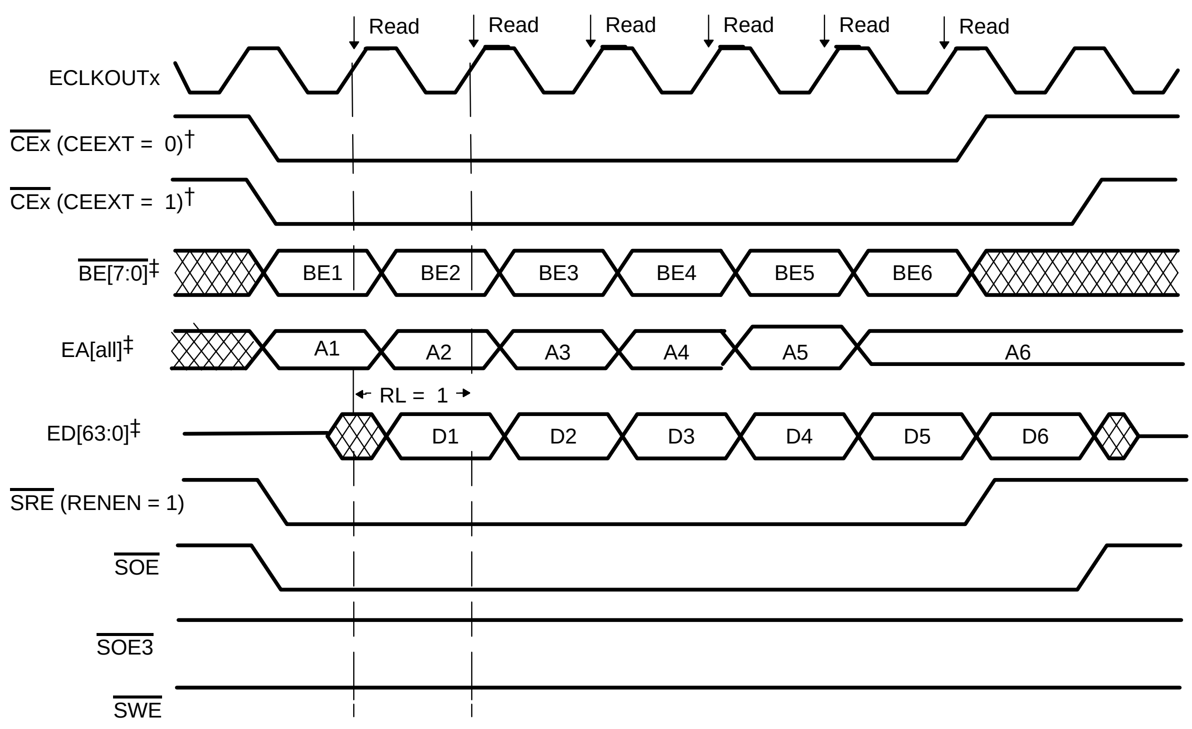
10–87 Add text below Figure 10–50 at the end of section 10.7.2:

Care must be taken when implementing glueless synchronous FIFO interface:

- ❑ For glueless synchronous FIFO read interface in CE3 space (Figure 10–49), writes to CE3 must not be performed. Internally, SOE3 = CE3 OR SOE. Performing a write causes CE3 and SOE3 to go active, hence REN and OE will be active. Data contention will occur on the ED bus since both DSP and FIFO will be driving data at the same time.
- ❑ For glueless synchronous FIFO write interface in any CE space (Figure 10–50), reads must not be performed. Reads cause CEx signal to go active; therefore, FIFO data corruption will occur since FIFO expects data from DSP.

10–88 Change Figure 10–51 into two figures. The subsequent figures are renumbered accordingly:

*Figure 10–51. TMS320C64x Standard Synchronous FIFO Read for CE0, CE1, or CE2*

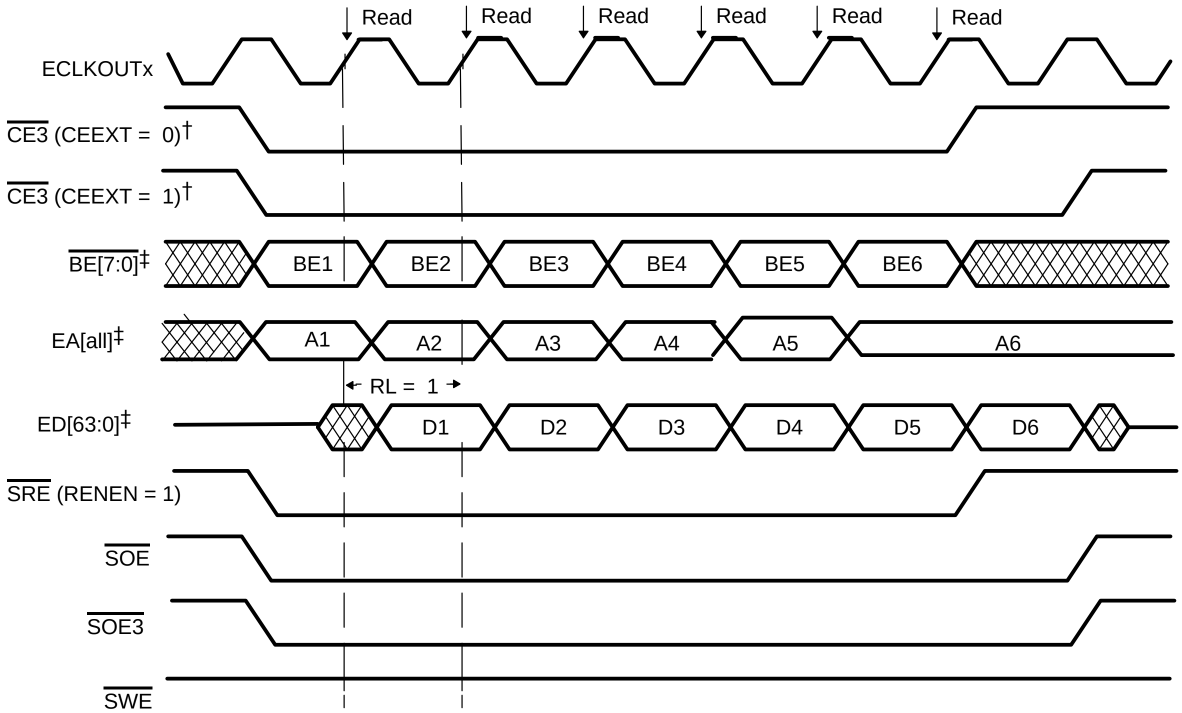



† CEEXT = 0 for glueless synchronous FIFO interface. CEEXT = 1 for interface with glue.

‡ Figure shows 64-bit interface. The MTYPE field selects the interface type to be 8-, 16-, 32-, or 64-bits wide.
For 32-bit interface, BE[3:0], EA[all], and ED[31:0] are used
For 16-bit interface, BE[1:0], EA[all], and ED[15:0] are used

**Page:** **Change or Add:**

*Figure 10–52. TMS320C64x Standard Synchronous FIFO Read for CE3 only*

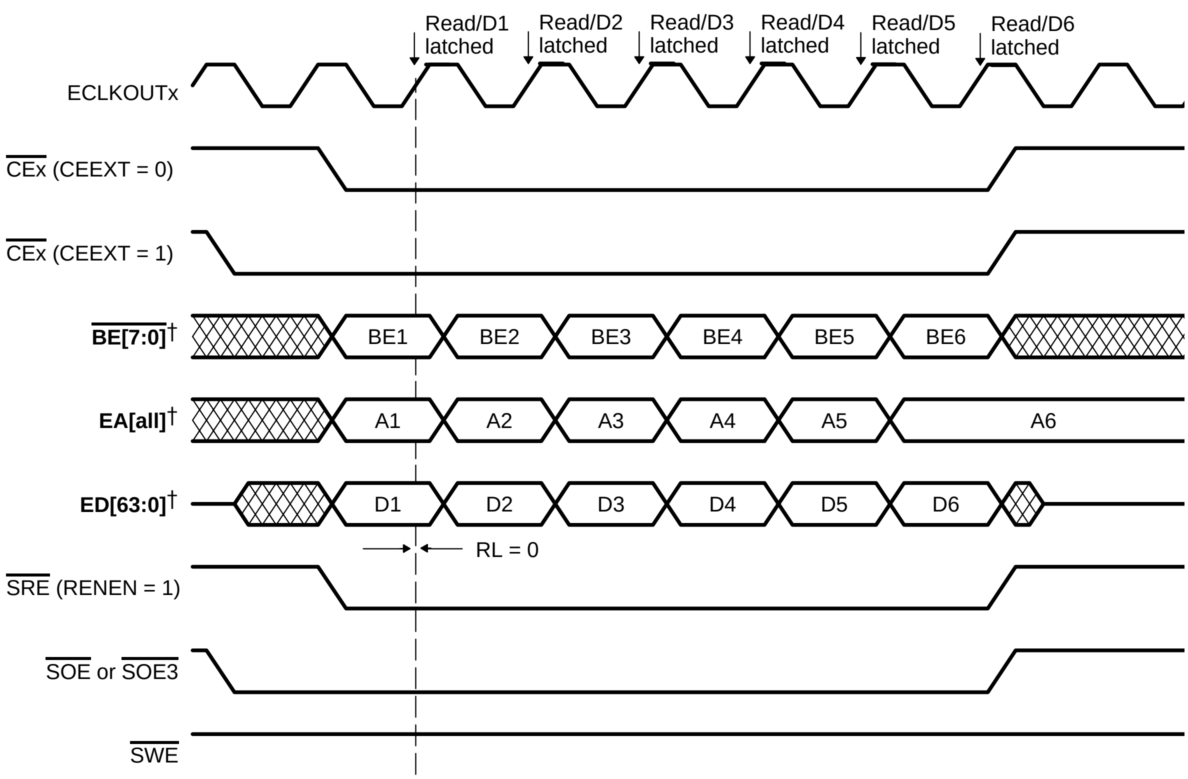



† CEEXT = 0 for glueless synchronous FIFO interface. CEEXT = 1 for interface with glue.
‡ Figure shows 64-bit interface. The MTYPE field selects the interface type to be 8-, 16-, 32-, or 64-bits wide.
For 32-bit interface, BE[3:0], EA[all], and ED[31:0] are used
For 16-bit interface, BE[1:0], EA[all], and ED[15:0] are used

**Page:** **Change or Add:**

10–90 Change the signal names and footnote in Figure 10–53:

*Figure 10–53. TMS320C64x FWFT Synchronous FIFO Read*




† Figure shows 64-bit interface. The MTYPE field selects the interface type to be 8-, 16-, 32-, or 64-bits wide.
For 32-bit interface, BE[3:0], EA[all], and ED[31:0] are used
For 16-bit interface, BE[1:0], EA[all], and ED[15:0] are used

10–91 Change the third paragraph in section 10.8:

... It has also been enhanced to allow for longer read hold time, and the 8- and 16-bit interface modes have been extended to include writable asynchronous memories, instead of ROM devices. **To avoid bus contention,** a programmable turnaround time (TA) also allows the user to control the **minimum number** of cycles between between **a read followed by a write (same or different CE spaces), or between reads from different CE spaces.**

10–97 Add a sub-bullet to the 4th bullet in section 10.8.3:

- At the end of the hold period: AOE becomes inactive as long as another read access to the same CE space is not scheduled for the next cycle.
  - CE becomes inactive only if another read or write access to the same CE space is not pending.

**Page:** **Change or Add:**

10–99 Change the second sub-bullet of the 4th bullet in section 10.8.4:

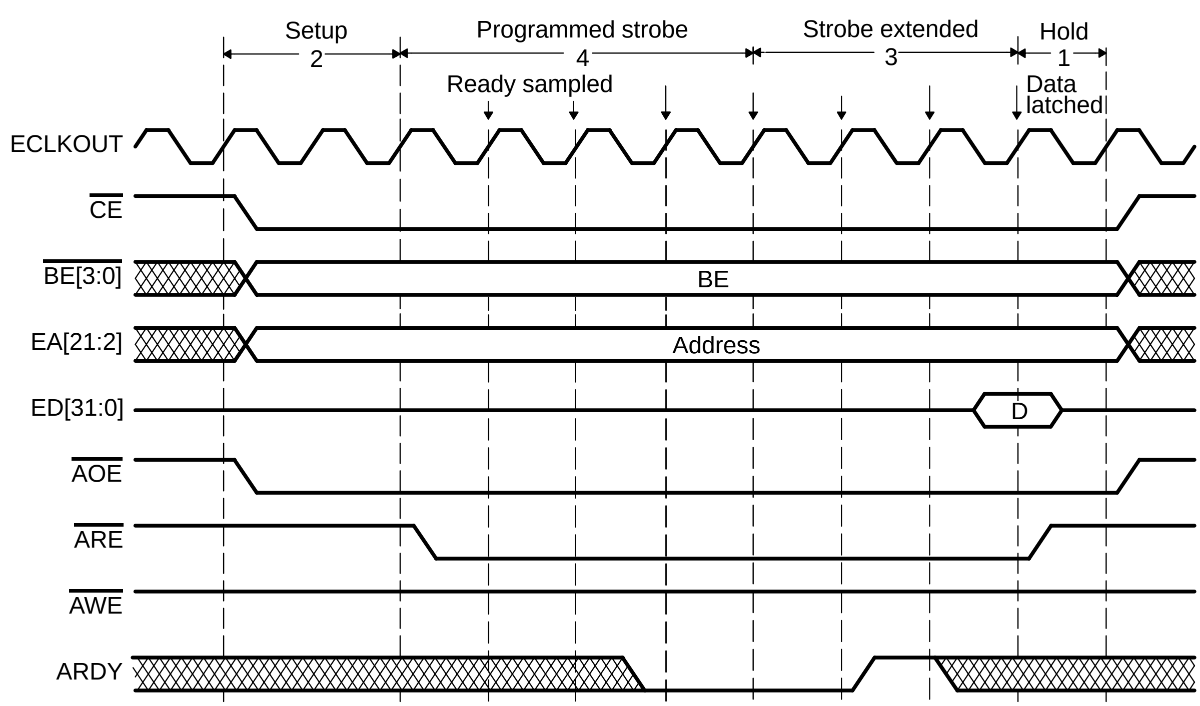
- At the end of the hold period:
  - ED goes into the high-impedance state only if another write access to the same CE space is not scheduled for the next cycle.
  - $\overline{\text{CE}}$ becomes inactive only if another **read or write access** to the same CE space **is not pending.**

10–102 Change the paragraph in section 10.8.5:

- **TMS320C621x/C671x/C64x Operation: For C621x/C671x**, ARDY is sampled for the first time on the ECLKOUT cycle at the end of the programmed strobe period **(Figure 10–62)**. **For C64x, ARDY is sampled two clock cycles before the last cycle of the programmed strobe period (Figure 10–63).** If sampled low, the strobe period is extended and ARDY is sampled again on the next ECLKOUT cycle. Read data is latched by the C621x on the cycle that ARDY is sampled high. The $\overline{\text{ARE}}$ signal goes high on the the following cycle. Therefore, the strobe period is visibly extended by three cycles in Figure 10–62 **and Figure 10–63**, although data is latched after the second cycle.

10–102 Add a new Figure 10–63. The subsequent figures are renumbered accordingly:

*Figure 10–63. TMS320C64x Ready Operation*



| Page: | Change or Add: |
|---|---|
| 10–102 | Add a new section 10.8.6: |

### 10.8.6 C620x/C670x Illegal Access to Asynchronous Memory

An access to a section of memory that does not return a ready indication is not allowed. This includes accesses to EMIF asynchronous spaces with ARDY pulled inactive externally or left floating on the device. Possible requestors are: CPU program fetches, CPU loads and stores, programmed DMA channels or HPI/PCI/XBUS host mastering of the DMA through the auxiliary DMA. This type of access can create a stall indefinitely.

| Page: | Change or Add: |
|---|---|
| 10–103 | Change section 10.9. The subsequent figures and tables are renumbered accordingly: |

## 10.9 Peripheral Device Transfers (PDT) (TMS320C64x)

To perform a peripheral device transfer (PDT), the PDTS or PDTD bits in the EDMA options parameter must be appropriately set. Refer to Chapter 6, *EDMA Controller*, for details. A PDT allows you to directly transfer data from an external peripheral (such as a FIFO) to another external memory (such as SDRAM), and vice versa. Normally, this type of transfer would require an EMIF read of a peripheral followed by an EMIF write to memory, or an EMIF read of a memory followed by an EMIF write to a peripheral.

In a typical system, however, both the peripheral and memory are connected to the same physical data pins, and thus an optimization can be made. In a PDT write transfer, data is driven by the peripheral directly, and written to the memory in the same bus transaction. In a PDT read transfer, data is driven by the memory directly, and written to the peripheral in the same bus transaction. Typically, the memory device will be mapped to an addressable location via a CEx signal. Normally, the peripheral device is not memory mapped (it does not use a CEx signal). It is activated with the PDT signal and optionally a combination of other control signals (via external logic).

PDT transfers are classified in terms of the memory on the EMIF. A PDT write is a transfer from a peripheral to memory (memory is physically written). A PDT read is a transfer from memory to a peripheral (memory is physically read). For a PDT read, the EMIF ignores the read data on the external bus. For a PDT write, the EMIF data bus is placed in a high-impedance state during the transaction to allow the external peripheral or memory to drive the data bus. A PDT transfer is only supported when the external memory is SDRAM (specified by the MTYPE field in the CE space control register). PDT transfers should not be performed to non-SDRAM CE spaces.

In a PDT transaction, the EMIF:

1) Generates normal SDRAM read bus cycles for a PDT read, or generates normal SDRAM write bus cycles for a PDT write. For example, for a PDT read from CE0 configured as SDRAM, the EMIF asserts CE0 and generates the SDRAM read control signals. The EMIF does not explicitly generate the control signals to the destination peripheral in a PDT read. For a PDT write to CE0 with an SDRAM, the EMIF asserts CE0 and generates the SDRAM write control signals. The EMIF does not explicitly generate the control signals to the source in a PDT write.

2) Generates PDT control signal (PDT) and the PDT address pins. PDT is asserted low 0, 1, 2, or 3 cycles prior to the data phase of the transaction. The PDTWL and PDTRL fields in the PDT control register (PDTCTL) control the latency of the PDT signal for write and read transfers, respectively (see section 10.9.1).

**Page:** **Change or Add:**

3) In addition to the direct control provided by the $\overline{\text{PDT}}$ signal, the EMIF uses two upper-address pins ($\overline{\text{PDTA}}$ and PDTDIR) during a PDT transfer. Table 10–35 describes each of these signals, listing their appropriate EMIF and SDRAM pin and their function.

4) Drives EMIF data outputs (ED pins) to a high-impedance state.

*Table 10–35. PDT Signal Description*

| Pin Name | 64-bit EMIF | 32-bit EMIF | 16-bit EMIF | SDRAM | Function |
|---|---|---|---|---|---|
| $\overline{\text{PDTA}}$ | EA19 | EA18 | EA17 | A16 | PDT access |
| PDTDIR | EA20 | EA19 | EA18 | A17 | PDT read, not write |
| $\overline{\text{PDT}}$ | $\overline{\text{PDT}}$ | $\overline{\text{PDT}}$ | $\overline{\text{PDT}}$ | | PDT data |

During a PDT transfer, the EMIF drives $\overline{\text{PDTA}}$ active and PDTDIR to its appropriate state. Activation of $\overline{\text{PDTA}}$ signals that a PDT transfer is controlling the bus while the state of the PDTDIR denotes the type of transfer, either a read (high) or write (low) to memory.

For a non-PDT transfer to SDRAM:

- $\overline{\text{PDT}}$ is inactive
- $\overline{\text{PDTA}}$ is high
- PDTDIR is not used

For a non-PDT transfer to asynchronous or programmable synchronous memory:

- $\overline{\text{PDT}}$ is inactive
- $\overline{\text{PDTA}}$ functions as an address bit
- PDTDIR functions as an address bit

**Page:** **Change or Add:**

### 10.9.1 PDT Control Register (PDTCTL)

The PDT control register, shown in Figure 10–63 and defined in Table 10–36, configures the latency of the $\overline{\text{PDT}}$ signal with respect to the data phase of the transaction.

*Figure 10–63. EMIF PDT Control Register (PDTCTL)*

| 31 – 4 | 3 – 2 | 1 – 0 |
|---|---|---|
| Reserved | PDTWL | PDTRL |
| R, +0 | RW, +00 | RW, +00 |

*Table 10–36. EMIF PDT Control Register (PDTCTL) Field Descriptions*

| Field | Description |
|---|---|
| PDTRL | PDT Read Latency<br>PDTRL=00: $\overline{\text{PDT}}$ asserted 0 cycles prior to the data phase of a read transaction<br>PDTRL=01: $\overline{\text{PDT}}$ asserted 1 cycle prior to the data phase of a read transaction<br>PDTRL=10: $\overline{\text{PDT}}$ asserted 2 cycles prior to the data phase of a read transaction<br>PDTRL=11: $\overline{\text{PDT}}$ asserted 3 cycles prior to the data phase of a read transaction |
| PDTWL | PDT Write Latency<br>PDTRL=00: $\overline{\text{PDT}}$ asserted 0 cycles prior to the data phase of a write transaction<br>PDTRL=01: $\overline{\text{PDT}}$ asserted 1 cycle prior to the data phase of a write transaction<br>PDTRL=10: $\overline{\text{PDT}}$ asserted 2 cycles prior to the data phase of a write transaction<br>PDTRL=11: $\overline{\text{PDT}}$ asserted 3 cycles prior to the data phase of a write transaction |

**Page:** **Change or Add:**

### 10.9.2 PDT Write

A PDT write transfer refers to a transfer from a peripheral to memory, in which the memory is physically written. To enable a PDT write transfer, the PDTD bit in the EDMA options field must be set to 1. The assertion/deassertion of the PDT address pins ($\overline{\text{PDTA}}$ and PDTDIR) and the PDT pin are timed according to the destination memory clock. Since the destination memory is SDRAM, ECLKOUT1 is used.

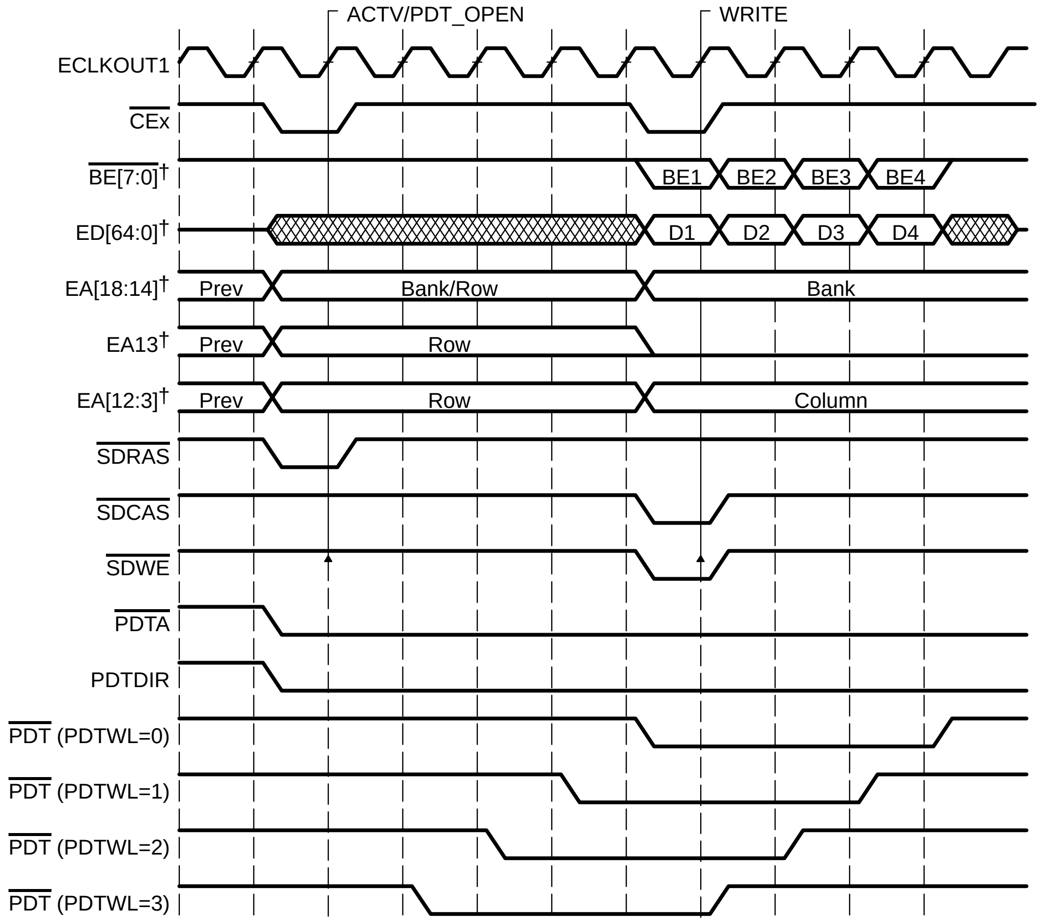
A PDT write transfer procedure is as follows:

1) The destination address is to a CE space set as SDRAM:
   - The PDT access bit ($\overline{\text{PDTA}}$) and the PDT direction (PDTDIR) are used to give the system advance warning that a PDT transaction is pending. This may be useful to activate bus switches or other external logic that will control the actual PDT transfer.
   - If the access is to a closed page, then during the ACTV cycle, $\overline{\text{PDTA}}$ is low, and PDTDIR is low to indicate a write access.
   - If the access is to an open page previously accessed without a PDT operation, then the page will be closed and reopened, with the PDT address pins asserted low during the ACTV cycle.
   - If the access is to an open page previously accessed with a PDT operation, then the access goes directly to the data phase.
2) Normal write control signals are generated to the appropriate CE space.
3) The write transaction proceeds as normal except:
   - EMIF data outputs remain in a high-impedance state. Therefore, the memory latches data from the peripheral device, instead of data from the EMIF.
   - $\overline{\text{PDT}}$ is asserted low PDTWL cycles prior to the data being latched by the destination device. This implies that the peripheral must drive valid data PDTWL cycles after $\overline{\text{PDT}}$ is active.

Figure 10–64 displays the timing diagram for a PDT write transaction.

**Page:** **Change or Add:**

*Figure 10–64. PDT Write Transaction Timing Diagram*




† For EMIFB, BE[1:0], EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

#### *10.9.2.1 PDT Write Examples*

PDT write transactions are supported from both a standard synchronous (STD) FIFO interface and a first word fall through (FWFT) FIFO interface. Table 10–37 gives an overview of the supported systems. Figure 10–65 through Figure 10–70 describe the various systems where PDT write transfers are supported. These examples can be extended to other external peripherals.

*Table 10–37. Supported Set Ups for PDT Write Transfers*

| Case | System Description |
|---|---|
| A | Glueless PDT write transfer to either FWFT FIFO or standard FIFO. PDTWL should be programmed accordingly with respect to the FIFO interface selected. Limited to SDRAM only in the system. |
| B | PDT write transfer from a FWFT FIFO with glue. PDTWL programmed to 1 cycle latency. |
| C | PDT write transfer from a standard FIFO with glue. PDTWL programmed to 1 cycle latency. |

**Page:** **Change or Add:**

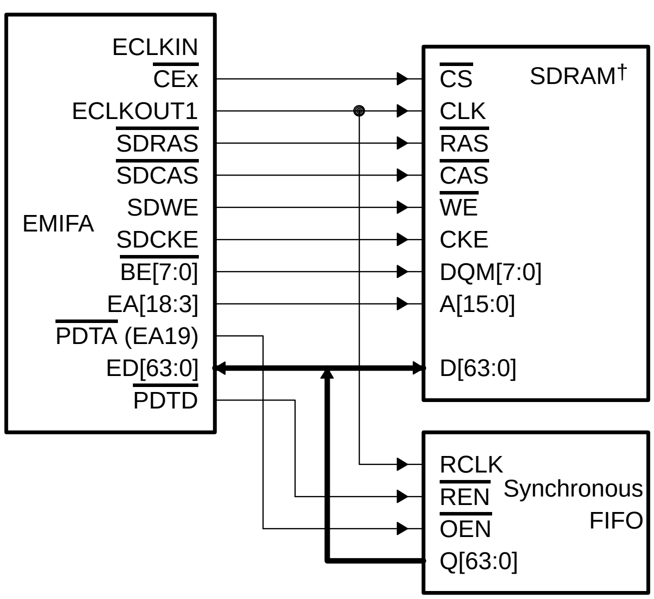
Figure 10–65 shows the glueless synchronous FIFO interface for a PDT write transaction. When the glueless interface is implemented, SDRAM must be the only memory type present in the system. This is because the glueless interface uses PDTA to generate the output enable (OE) to the FIFO. If a memory type other than SDRAM is included in the system, the upper EMIF address bit used to generate PDTA (EA17, EA18, or EA19 depending on the EMIF data bus interface), will be used (see Table 10–12). In this setup, PDT generates the read enable (REN) to the FIFO. PDT latency should be programmed as follows:

- FWFT FIFO: PDTWL = 0
- Standard FIFO: PDTWL = 1

Figure 10–66 shows the timing diagram for a glueless PDT write transaction to a synchronous FIFO. Note, the PDT and REN waveforms differ between the standard FIFO interface and the FWFT FIFO interface.

Figure 10–67 and Figure 10–68 show a PDT write interface with glue to a FWFT FIFO. Figure 10–69 and Figure 10–70 show a PDT write interface with glue to a standard FIFO. In each of these systems, external logic is used to shape the PDT signal to generate the appropriate control inputs to the FIFOs. For both systems, PDTWL should be programmed to 1. These systems are not restricted to SDRAM only, any combination of memory types is allowed.

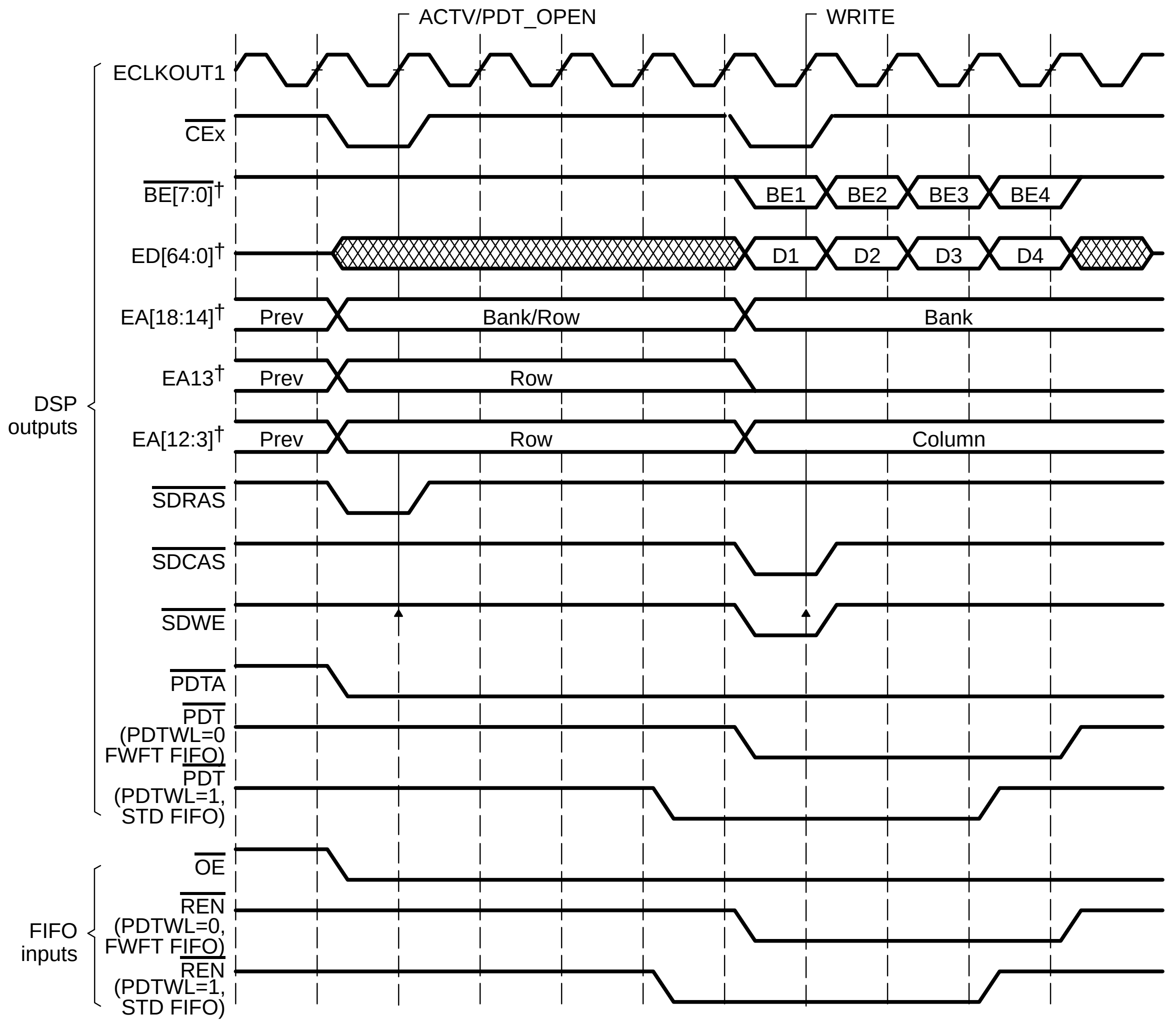
*Figure 10–65. Case A: Glueless PDT Write Interface From Synchronous FIFO*




† SDRAM must be the only memory type present in the system.

**Page:** **Change or Add:**

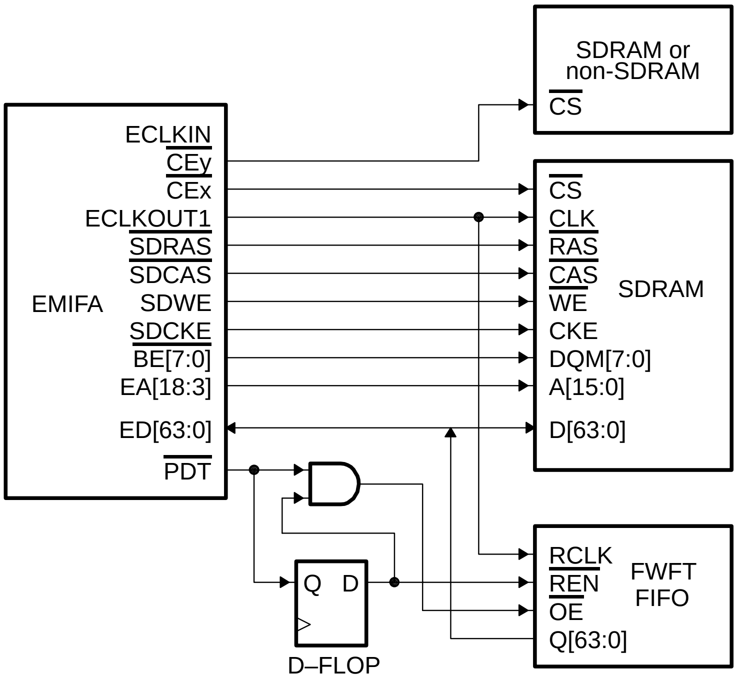
*Figure 10–66. Case A: Glueless PDT Write Transfer From Synchronous FIFO Timing Diagram*




† For EMIFB, BE[1:0], EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

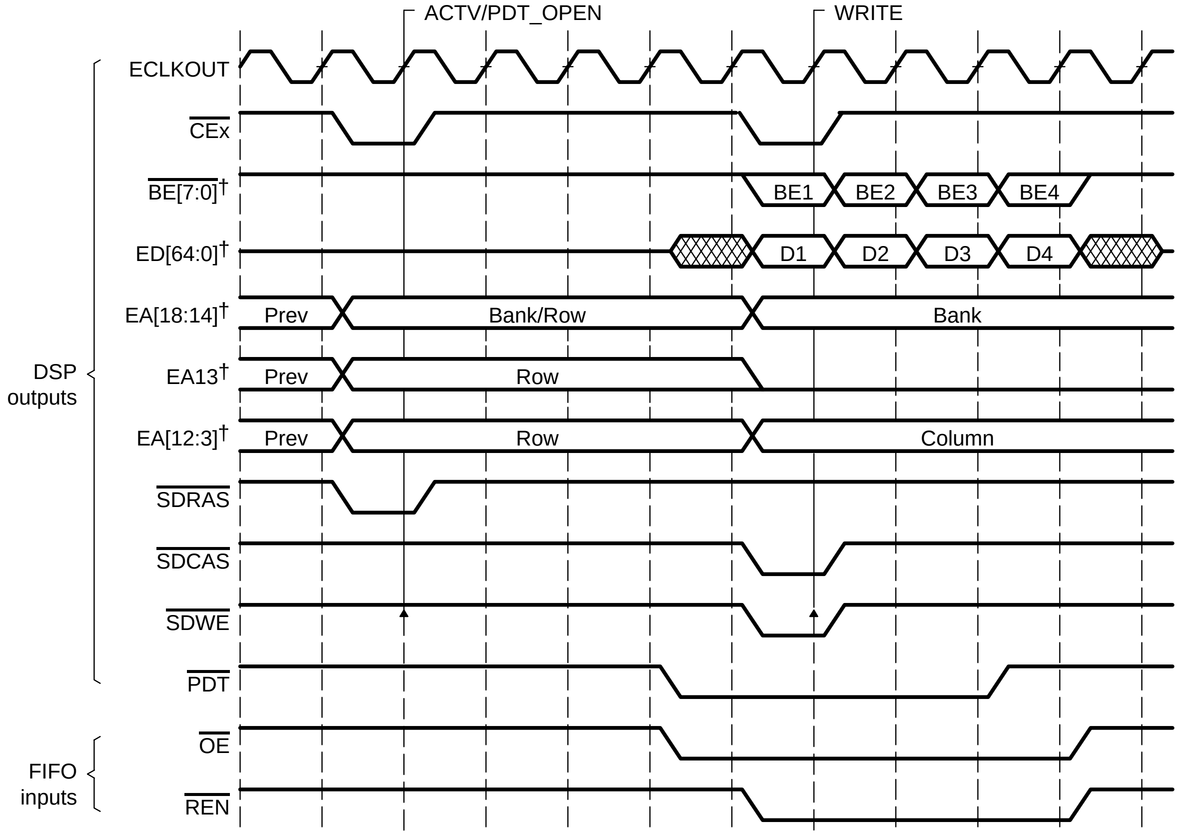
**Page:** **Change or Add:**

*Figure 10–67. Case B: PDT Write Interface From FWFT FIFO With Glue*



**Page:** **Change or Add:**

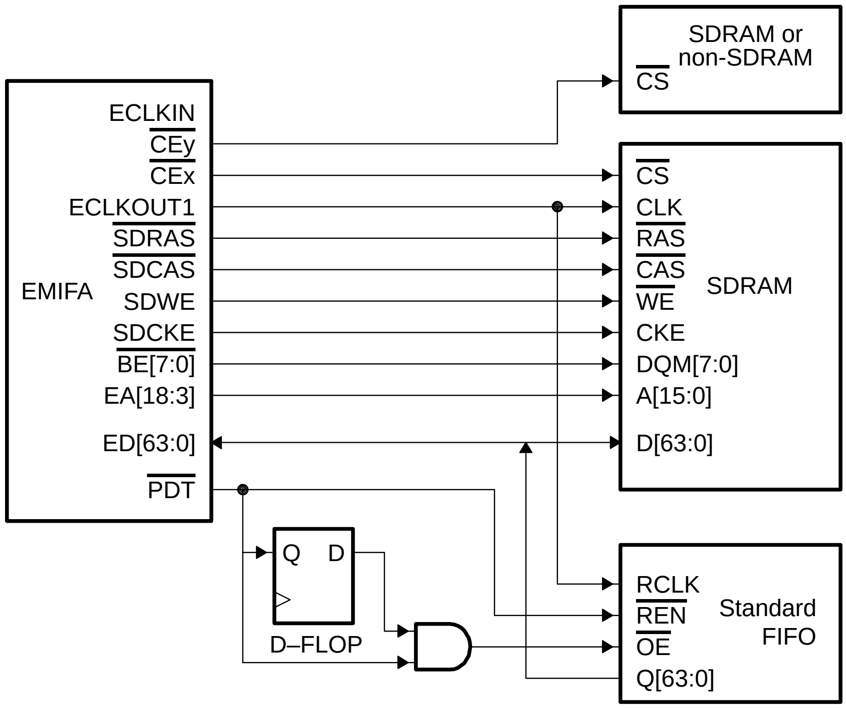
*Figure 10–68. Case B: PDT Write Transfer From FWFT FIFO With Glue Timing Diagram*




† For EMIFB, BE[1:0], EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

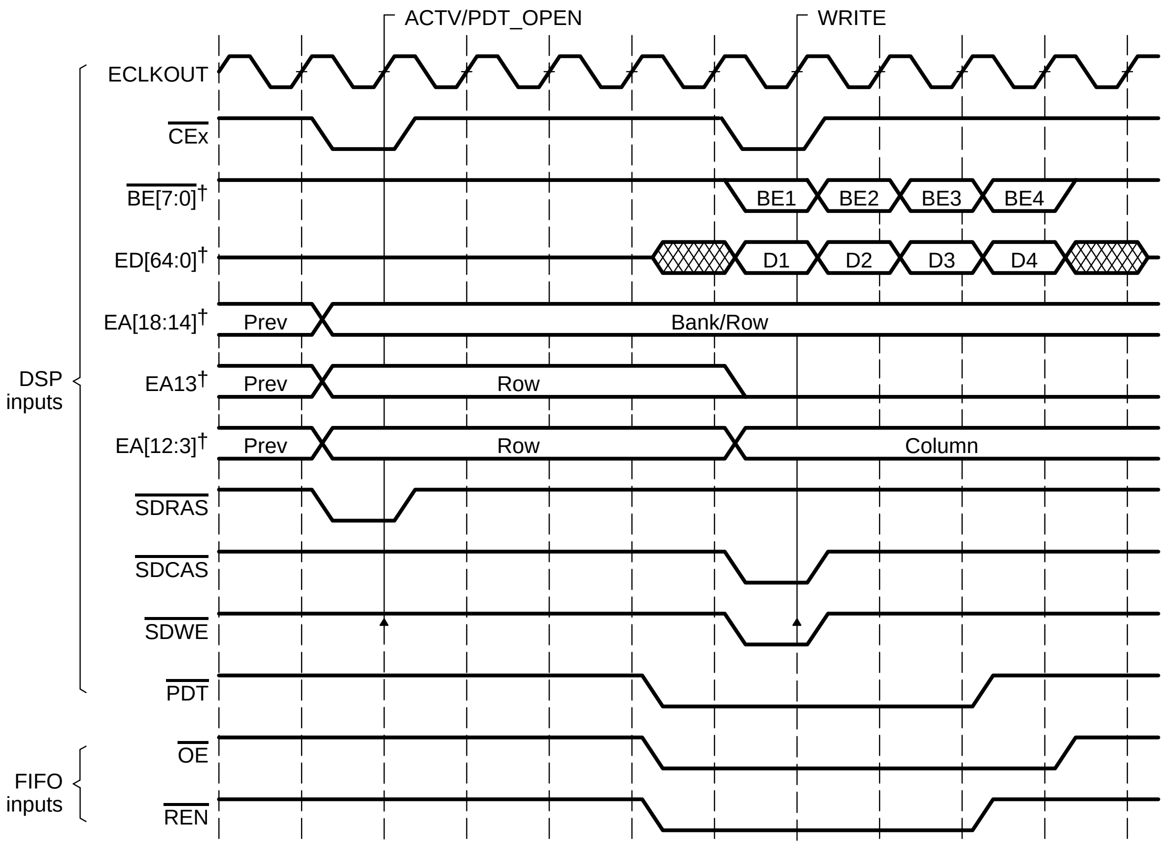
**Page:** **Change or Add:**

*Figure 10–69. Case C: PDT Write Interface From Standard FIFO With Glue*



**Page:** **Change or Add:**

*Figure 10–70. Case C: PDT Write Transfer From Standard FIFO With Glue Timing Diagram*




† For EMIFB, $\overline{BE[1:0]}$, EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

**Page:** **Change or Add:**

### 10.9.3 PDT Read

A PDT read transfer refers to a transfer from memory to a peripheral, in which the memory is physically read. To enable a PDT read transfer, the PDTS bit in the EDMA options field must be set to 1. The assertion/deassertion of the PDT address pins ($\overline{\text{PDTA}}$ and PDTDIR) and the PDT pin are timed according to the source memory clock. Since the source memory is SDRAM, ECLKOUT1 is used.

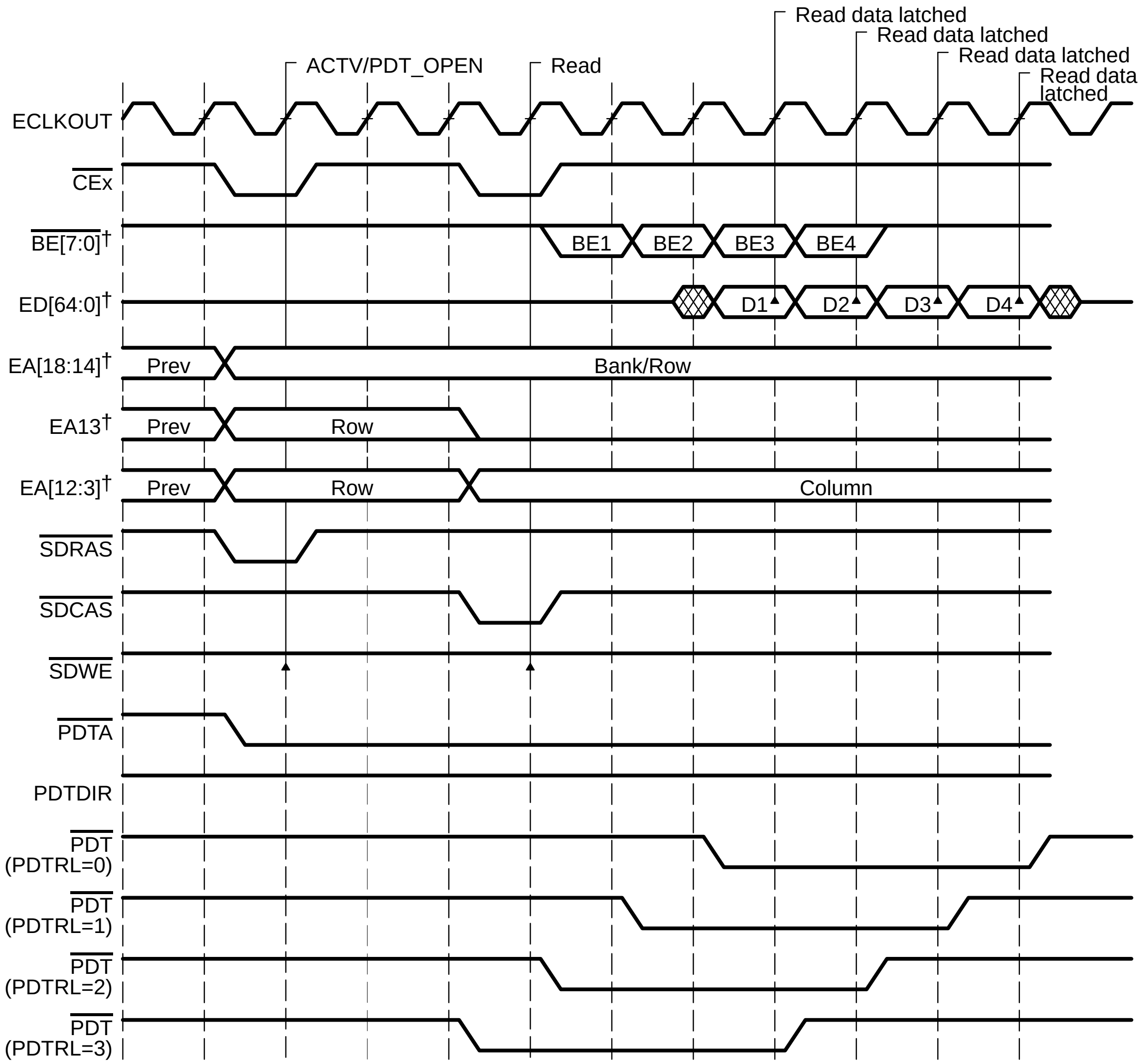
A PDT read transfer procedure is as follows:

1) The source address is to a CE space set as SDRAM:
   - The PDT access bit ($\overline{\text{PDTA}}$) is used to give the system advance warning that a PDT transaction is pending. This may be useful to activate bus switches or other external logic that will control the actual PDT transfer.
   - If the access is to a closed page, then during the ACTV cycle, $\overline{\text{PDTA}}$ is asserted low.
   - If the access is to an open page previously accessed without a PDT operation, then the page will be closed and reopened, with $\overline{\text{PDTA}}$ asserted low during the ACTV cycle.
   - If the access is to an open page previously accessed with a PDT operation, then the access goes directly to the data phase.
   - PDTDIR remains inactive (high) throughout the course of the transaction.
2) Normal read control signals are generated to the appropriate CE space.
3) The read transaction proceeds as normal except:
   - EMIF ignores data at the ED pins
   - $\overline{\text{PDT}}$ is asserted low PDTRL cycles before the data is to be returned by the source SDRAM.

Figure 10–71 displays the timing diagram for a PDT read transaction.

**Page:** **Change or Add:**

*Figure 10–71. PDT Read Transaction (CAS Latency is 3) Timing Diagram*




† For EMIFB, $\overline{\text{BE[1:0]}}$, EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

### *10.9.3.1 PDT Read Examples*

PDT read transactions are supported from both a standard synchronous FIFO interface and a first word fall through (FWFT) FIFO interface. Figure 10–72 shows an example of a PDT read transaction from SDRAM to a synchronous FIFO. The $\overline{\text{PDT}}$ signal is used to generate the write enable ($\overline{\text{WEN}}$) input to the FIFO. $\overline{\text{PDT}}$ latency should be programmed to 0. This system is not restricted to SDRAM only, any combination of memory types is allowed. This example can be extended to other external peripherals. Note in this example that no glue is required. However, if both read and write PDT transactions are required on the same bus, glue is required to properly create the $\overline{\text{OE}}$, $\overline{\text{REN}}$, and $\overline{\text{WEN}}$ signals for the FIFO (see section 10.9.4.1). Figure 10–73 shows the timing diagram for a PDT read transfer to a synchronous FIFO.

**Page:** **Change or Add:**

*Figure 10–72. Case D: Glueless PDT Read Interface to Synchronous FIFO*

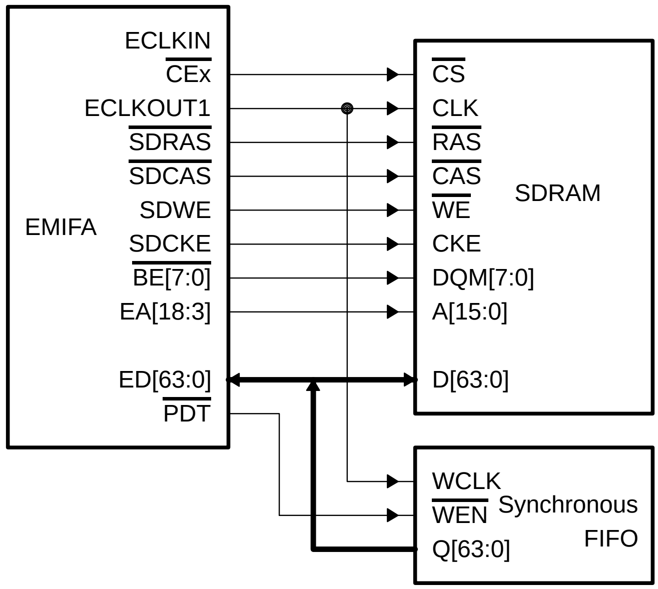
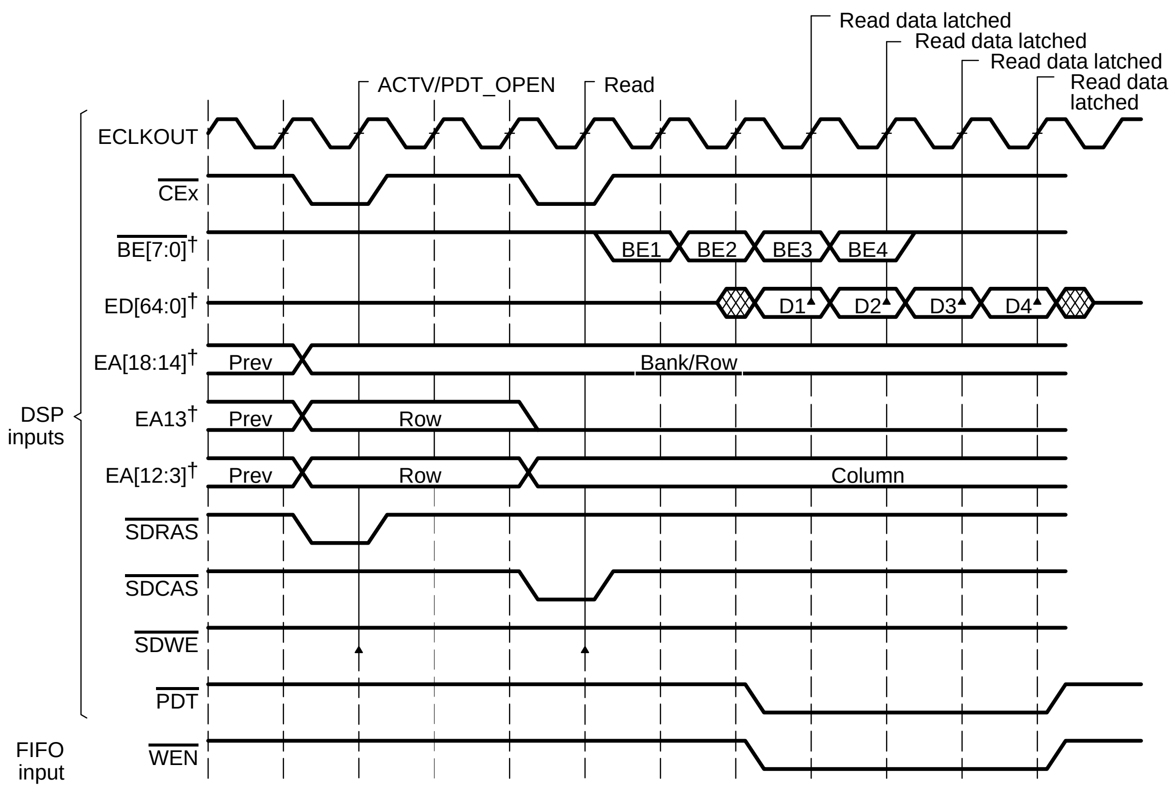



*Figure 10–73. Case D: Glueless PDT Read Transfer to Synchronous FIFO Timing Diagram*




† For EMIFB, BE[1:0], EA[16:12], EA[10:1], EA11, and ED[15:0], respectively, are used instead.

**Page: Change or Add:**

## 10.9.4 PDT Transfers with Multiple FIFOs on the Same Bus

The following sections describe PDT transfers with multiple FIFOs connected to a single CE space via the same data bus.

### *10.9.4.1 PDT Read and Write Transactions on the Same Bus*

If both PDT read and write transactions are required on the same bus, glue is required to properly create the OE, REN, and WEN signals for the FIFO. Figure 10–74 shows a system that can perform both a PDT read and write transaction to a CE space configured as SDRAM.

The discrete logic needed to generate the appropriate input signals to the FIFOs has two main stages: Direction Detect and Demux and Signal Generation. The direction detect stage latches the state of the PDTDIR (EA20, E19, or E18, depending on the EMIF data bus interface) signal, if CEx is active and outputs the signal DIR. The demux stage receives the DIR signal and the PDT signal as inputs. Based on these inputs, the demux stage generates the appropriate input signals to drive the selected read or write FIFO. Figure 10–75 and Figure 10–76 show the timing diagrams when write and read transactions are performed in a system with FWFT FIFOs, respectively.

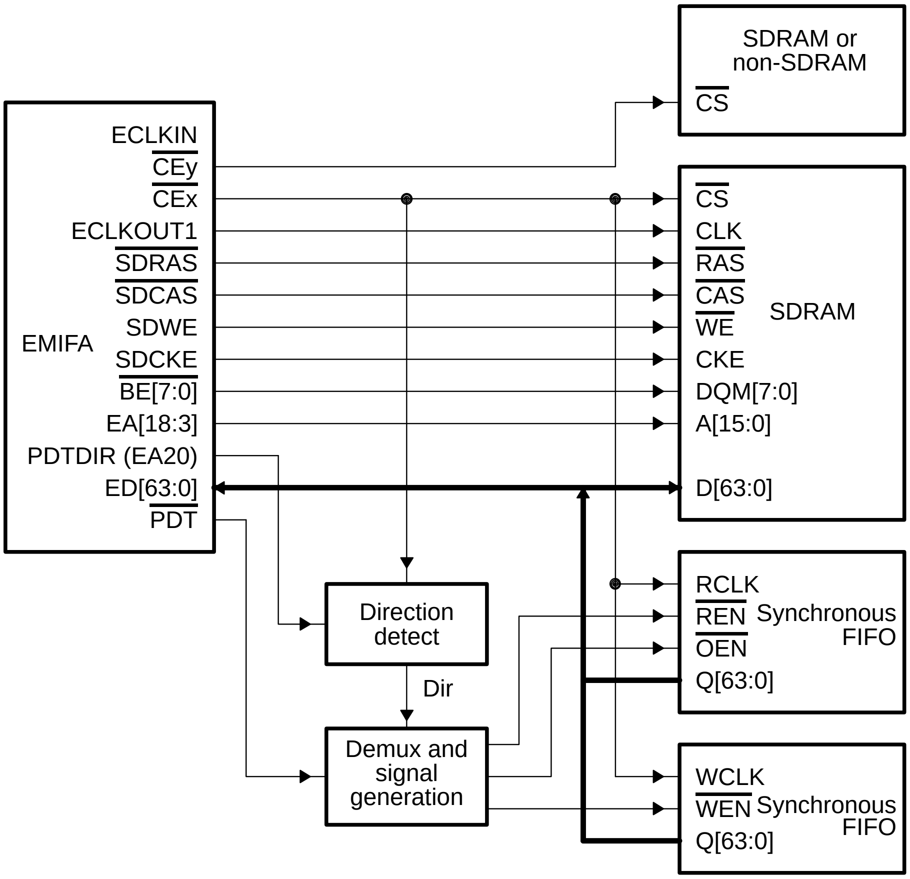
During a PDT write transaction, PDTDIR is active (low) denoting data is read from the synchronous FIFO and written to SDRAM. Therefore, during a PDT write transaction the appropriate OE and REN signals should be generated by the demux stage (see Figure 10–75).

During a PDT read transaction, PDTDIR is inactive (high) denoting data is read from SDRAM and written to the synchronous FIFO. Therefore, during a PDT read transaction the appropriate WEN signal should be generated by the demux stage (see Figure 10–76).

The OE and REN signals generated during the PDT write transaction, shown in Figure 10–75, are identical to those described for case B. The WEN signal generated during the PDT read transaction, shown in Figure 10–76, is identical to that described for case D. Therefore, for PDT write and PDT read timing diagrams when standard FIFOs are in the system, see Figure 10–70 and Figure 10–73, respectively.

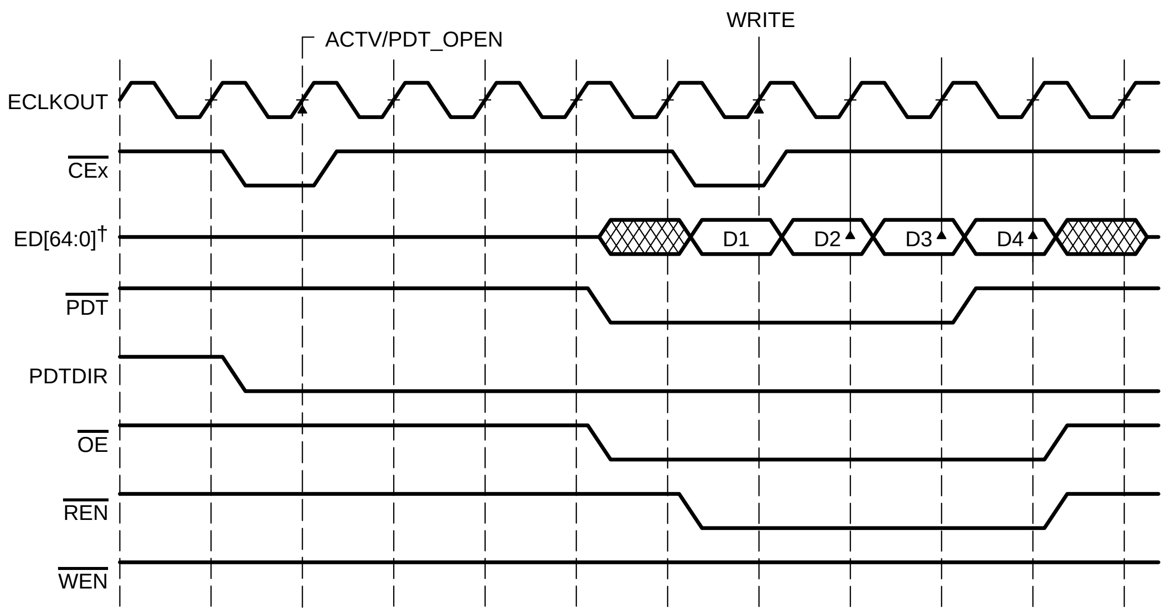
**Page:** **Change or Add:**

*Figure 10–74. Case E: PDT Read and Write Interface With Multiple FIFOs*



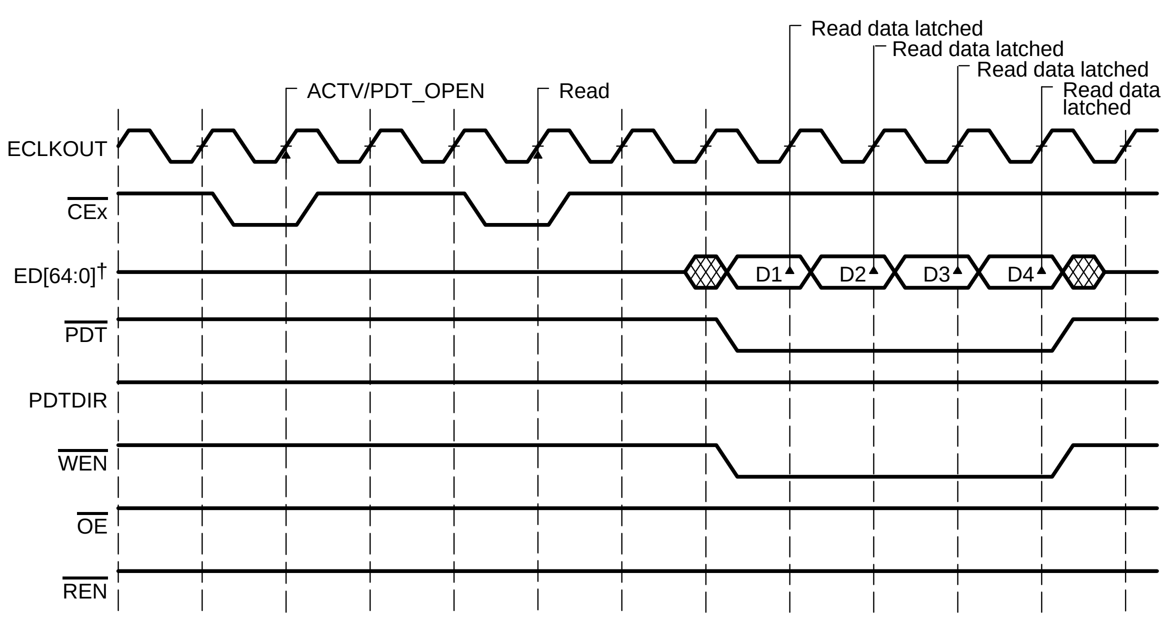
**Page:** **Change or Add:**

*Figure 10–75. Case E: PDT Write Transfer with Read and Write FIFOs in the System (FWFT FIFO)*




† For EMIFB, ED[15:0] is used instead.

*Figure 10–76. Case E: PDT Read Transfer with Read and Write FIFOs in the System (FWFT FIFO)*




† For EMIFB, ED[15:0] is used instead.

**Page:** **Change or Add:**

### 10.9.4.2 Multiple PDT Read and Write Transactions on the Same Bus

Each of the previous systems can be extended to include additional read and or write FIFOs. In a system where more than two synchronous FIFOs are interfaced to a single CE space, additional unused upper row address bits of the SDRAM can be used to select the appropriate FIFO for the current transaction. In this case, the required glue receives PDTDIR, $\overline{\text{PDT}}$, EAxx (upper address bits), and $\overline{\text{CEx}}$ as inputs. Based on the state of these inputs, the control signals are generated for the selected FIFO.

The addition of multiple read and or write FIFOs is limited by the size of the SDRAM in a given CE space. This is because the size of the SDRAM dictates the number of column and row address bits required per transaction. Therefore, in some systems, it is possible to have zero unused valid address bits (see Figure 10–22 and Figure 10–23). For example, a 64-bit SDRAM that requires 11 row address bits, 8 column address bits, and 1 bank bit leaves 4 unused valid address bits for EMIFA (see the first row in Figure 10–22). This means an additional 16 FIFOs per direction can be added to the data bus. However, a 64-bit SDRAM that requires 13 row address bits, 10 column address bits, and 2 bank bits leaves zero unused valid address bits for EMIFA (see the last row in Figure 10–22). This means no additional FIFOs can be added to the data bus. Table 10–38 shows the relationship between unused valid address bits and the number of additional peripherals that can be added to the same data bus.

*Table 10–38. Limitations on the Number of Additional Peripherals for a PDT Transfer*

| Number of Unused Valid Address Bits | Number of Additional Peripherals that can be Added |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |

**Page:** **Change or Add:**

### 10.9.5 PDT Transfers: Bus Width and DMA Considerations

When performing a PDT transfer, the bus width and DMA configuration must be considered. The following describes the proper system configurations for a PDT transfer:

- The FIFO (or external peripheral) bus width must equal the SDRAM bus width.
- DMA must be configured such that:
  - The SRC and DST addresses are set to the same address, which matches the SDRAM address.
  - The SRC and DST addresses must be aligned to the memory (MTYPE) bus width.
  - Element size (ESIZE) is set to a 32-bit word. This is a preferred setting, see section 6.18, *EDMA Performance*.
  - Element count (ELECNT) is set to a multiple of the bus width size (in elements).

Table 10–39 summarizes the DMA configurations for supported SDRAM bus widths (MTYPE). SUM/DUM bits are set to increment.

*Table 10–39. DMA Configuration for a PDT Transfer*

| MTYPE (SDRAM width in bits) | Element size (ESIZE) | Element count (ELECNT) |
|---|---|---|
| 8 | 32, 16, 8 | No restrictions |
| 16 | 32, 16 | No restrictions |
| 32 | 32 | No restrictions |
| 64 | 32 | Even number |

| Page: | Change or Add: |
| --- | --- |
| 10–107 | Add a new section 10.11. The subsequent sections are renumbered accordingly: |

## 10.11 EMIF and CLKOUTx Usage Condition

There is a usage condition associated with the EMIF (EMIFA on C64x) that can affect the functionality of CLKOUTx. The EMIF global control register (GBLCTL) controls the logic that outputs the internal CPU/x clocks to the CLKOUTx pins. The bits in GBLCTL that enable CLKOUTx are clocked with the boot-time selected EMIF clock. Without a valid EMIF clock selected, ECLKIN (AECLKIN on C64x) or an internal clock, it is possible to have unknown values in GLBCTL and, therefore, a non-functional CLKOUTx. This happens only when ECLKIN is selected at boot time, but no external clock is provided. Furthermore, without a valid EMIF clock, the EMIF registers are not accessible or assured to have their default values. To avoid a non-functional CLKOUTx, a valid clock must be provided to the EMIF during the entire RESET active pulse.

10–108 Change the bullets in section 10.11:

- HOLD: hold request input. HOLD is synchronized internally to the CPU clock. This synchronization allows an asynchronous input while avoiding metastability. The external device drives this pin low to request bus access. HOLD is the highest priority request that the EMIF can receive during active operation. When the hold is requested, the EMIF stops driving the bus at the earliest possible moment, which may entail completion of the current accesses, device deactivation, and SDRAM bank deactivation. The external device must continue to drive HOLD low for as long as it wants to drive the bus. **The external device may deassert HOLD after HOLDA is asserted and the bus is no longer needed.** If any memory spaces are configured for SDRAM, these memory spaces are deactivated and refreshed after HOLD is released by the external master.

- HOLDA: Hold acknowledge output. The EMIF asserts this signal active after it has placed its signal outputs in the high-impedance state. The external device can then drive the bus as required. The EMIF places all outputs in the high-impedance state with the exception of BUSREQ, HOLDA, and the clock outputs (CLKOUT1, CLKOUT2, ECLKOUT, SDCLK, and/or SSCLK, depending on the device). For the C64x, the EKxHZ bits in the GBLCTL register determine the state of the ECLKOUTx signals while HOLDA is asserted. **There may be glitches on the ECLKOUTx signals when they transition from being driven to being placed in the high-impedance state, and vice-versa. If any memory spaces are configured for SDRAM, these memory spaces are deactivated before HOLDA is asserted to the external master.**

**Page:** **Change or Add:**

- BUSREQ. Bus request output (C621x/C671x/C64x only). The EMIF asserts this signal active when any request is either pending to the EMIF or is in progress. The BUSREQ signal is driven without regard to the state of the $\overline{\text{HOLD}}$/$\overline{\text{HOLDA}}$ signals or the type of access pending. This signal can be used by an external master to release control of the bus if desired and may be ignored in some systems. **The BUSREQ signal may also go active when the SDRAM timer count reaches zero if SDRAM refresh is enabled (RFEN = 1).** For C64x, the BRMODE bit in the GBLCTL register indicates the bus request mode (section 10.2.1).

10–112 Change the title of section 10.13:

## 10.13 Boundary Conditions When Accessing EMIF Registers

10–112 Add a bullet to the end of section 10.13:

- Reading or writing registers if the EMIF is unclocked (if EMIF is configured to be clocked externally and no clock is provided)
  - Attempting to do so will lock up the chip
  - Some tools will attempt to access these registers automatically or as part of a script to provide default memory configurations. This must be disabled to prevent locking up the chip

10–113 Change the second paragraph in section 10.14:

For the C64x, the operation of the EMIF and device clocks is highly flexible. The CLKOUTx and ECLKOUTx can be disabled by setting the appropriate bits (CLK4EN, CLK6EN, EK1EN, EK2EN) in the EMIF global control register (GBLCTL). The ECLKOUT2 can be configured to run at 1×, 1/2×, or 1/4× the ECLKIN rate for the generic synchronous interface. **ECLKOUT2 rate should only be changed once during EMIF initialization from the default (1/4x) to either 1/2x or 1x**. In addition, the EK1HZ and EK2HZ bits in the GBLCTL configure the output EMIF clock behavior during hold. **The reset controller controls the output buffer of ECLKOUT1, ensuring that ECLKOUT1 is in a high-impedance state during device reset, see Figure 10–65. Table 10–37** summarizes the function of the EKxEN and EKxHZ bits. See also section 10.2.1.

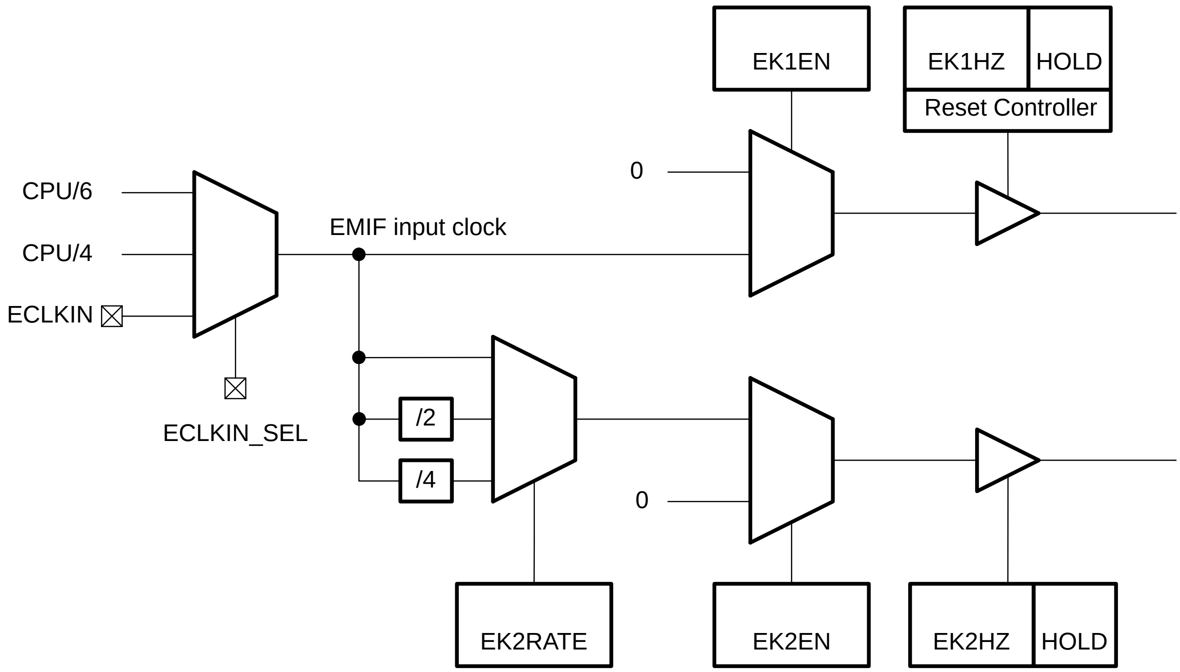
10–113 Add a paragraph between the second and third paragraphs in section 10.14:

ECLKOUTx does not turn off/on glitch free when it is disabled by setting the EKxEN or EKxHZ bits in the global control register. Therefore, ECLKOUT:

- Should not be disabled/enabled when connected to devices that have a strict requirement for glitch free disable/enable or external logic should be implemented to provide glitch free disable/enable.
- Should only be disabled if unused.

| Page: | Change or Add: |
|---|---|
| 10–113 | Add new Figure 10–65: |

*Figure 10–65. EMIF Clock Block Diagram—TMS320C64x EMIF*




11–5 Change two rows in Table 11–3:

*Table 11–3. TMS320C6203(B) Memory Map Summary*

| Address Range (Hex) | Size (Bytes) | Description of Two Memory Blocks: Map 0 | Description of Two Memory Blocks: Map 1 |
|---|---|---|---|
| 019C 0200–**019F FFFF** | 256K–512 | Internal peripheral bus power-down registers | |
| 8008 0000–FFFF FFFF | **2G–512K** | Reserved | |

| Page: | Change or Add: |
|---|---|
| 11–6 | Change four rows and add a footnote in Table 11–4: |

*Table 11–4. TMS320C621x/C671x Memory Map Summary*

| Address Range (Hex) | Size (Bytes) | Description of Memory Block |
|---|---|---|
| 8000 0000 – 8FFF FFFF | 256M | External memory interface **CE0**† |
| 9000 0000 – 9FFF FFFF | 256M | External memory interface **CE1**† |
| A000 0000 – AFFF FFFF | 256M | External memory interface **CE2**† |
| B000 0000 – BFFF FFFF | 256M | External memory interface **CE3**† |

† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space. To get 256MB of addressable memory, additional general-purpose output pins or external logic is required.

11–7 Change two rows in Table 11–5:

*Table 11–5. TMS320C64x Memory Map Summary*

| Address Range (Hex) | Size (Bytes) | Description of Memory Block |
|---|---|---|
| 0x01B40000 – 0x01B7FFFF | 256K | Internal configuration bus UTOPIA registers† **(C6415/C6416 only)** |
| 0x01C00000 – 0x01C3FFFF | 256K | Internal configuration bus PCI registers† **(C6415/C6416 only)** |

11–8 Change two rows, add address range 0x50000000 – 0x5FFFFFFF, and add a footnote in Table 11–5:

*Table 11–5. TMS320C64x Memory Map Summary (Continued)*

| Address Range (Hex) | Size (Bytes) | Description of Memory Block |
|---|---|---|
| 0x3C000000 – 0x3FFFFFFF | 64M | UTOPIA queues† **(C6415/C6416 only)** |
| 0x40000000 – **0x4FFFFFFF** | **256M** | Reserved |
| **0x50000000 – 0x5FFFFFFF** | **256M** | **TCP/VCP‡ (C6416 only)** |

† Address range is reserved for C6414.
‡ **Address range is reserved for C6414 and C6415**

| Page: | Change or Add: |
|---|---|

11–10 Change two rows and add a footnote in Table 11–6:

*Table 11–6. TMS320C620x/C670x Boot Configuration Summary*

| BOOTMODE [4:0] | Memory Map | Memory at Address 0 | Boot Process |
|---|---|---|---|
| 00000 | MAP 0 | SDRAM: **SDWID = 0 (512 elements per row)†** | None |
| 00001 | MAP 0 | SDRAM: **SDWID = 1 (256 elements per row)†** | None |

**† See section 10.2.3, *EMIF SDRAM Control Register*.**

11–12 Change the table title and the Description of HMOD (bit 11) in Table 11–7:

*Table 11–7. **TMS320C6202(B)**/C6203(B)/C6204 Boot and Device Configuration Description*

| XD Bit | Field | Description |
|---|---|---|
| 11 | HMOD | **Host mode.**<br>HMOD = 0: external host interface operates in asynchronous slave mode.<br>HMOD = 1: external host interface is in synchronous master/slave mode. |

11–13 Change the table title in Table 11–7:

*Table 11–7. **TMS320C6202(B)**/C6203(B)/C6204 Boot and Device Configuration Description (Continued)*

11–14 Change the table title in Table 11–8:

*Table 11–8. **TMS320C6205** Boot and Device Configuration Description*

11–15 Change the title of section 11.4.4:

### 11.4.4 TMS320C621x/C671x Boot and Device Configuration

11–16 Change the title of section 11.4.5:

### 11.4.5 TMS320C64x Boot and Device Configuration

| Page: | Change or Add: |
|---|---|
| 11–16 | Change the footnote in Table 11–10: |

† For C6414, HPI is used for host boot.
**For C6415/C6416,** HPI is used for host boot if PCI_EN = 0, and PCI is used for host boot if PCI_EN = 1.

11–17 to 20 Change all occurrences of C6415 to C6415/C6416 in paragraphs, tables, section title, table titles, and figure title in section 11.4.5.2.

11–22 Change the bullet in section 11.5:

- **Host boot process**: The CPU is held in reset while the remainder of the device is released. During this period, an external host can initialize the CPU's memory space as necessary through the host interface, including internal configuration registers, such as those that control the EMIF or other peripherals. Once the host is finished with all necessary initialization, it must set the DSPINT to complete the boot process. This transition causes the boot configuration logic to remove the CPU from its reset state. The CPU then begins execution 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 the CPU from internal reset only if the host boot process is selected. All memory may be written to and read by the host. This allows for the host to verify what it sends to the processor, if required. **After waking up from reset, the CPU needs to clear the DSPINT bit; otherwise, no more DSPINTs can be received.**

11–22 Change the Note bullet in section 11.5:

**Note:**

- **Expansion Bus:** For devices with XBUS, the XBUS can be used for the host boot. The type of host interface is determined by a set of latched signals **during reset.**

12–5 Change the last paragraph in section 12.2:

For devices with the EDMA peripheral (TMS320C621x/C671x/C64x), the data receive and transmit registers (DRR and DXR) are also mapped to memory locations 3xxxxxxxh shown in Table 12–5. The DRR and DXR locations 018Cxxxxh/0190xxxxh/01A4xxxxh are accessible via the peripheral bus, while the 3xxxxxxxh locations are accessible via the EDMA bus. Both the CPU and the EDMA in these devices can access the DRR and DXR in all the memory-mapped locations shown in Table 12–5. An access to *any* EDMA bus location in Table 12–5 is equivalent to an access to the DRR/DXR of the corresponding McBSP. For example, a read from any **word-aligned address** in 30000000h–33FFFFFFh is equivalent to a read from the DRR of McBSP0 at 018C0000h. A write to any **word-aligned address** in 30000000h–33FFFFFFh is equivalent to a write to the DXR of McBSP0 at 018C0004h. The user has a choice of reading from/writing to the DRR and DXR in either the 3xxxxxxxh or the ...

| **Page:** | **Change or Add:** |
|---|---|

12–6 Change the last paragraph in section 12.2:

... 018Cxxxxh/0190xxxxh/01A4xxxxh location. It is recommended that the user set up the EDMA to use the 3xxxxxxxh addresses for serial port servicing in order to free up the peripheral bus for other functions. The McBSP control registers are accessible only via the peripheral bus, and thus are mapped only to the 018Cxxxxh/0190xxxxh/01A4xxxxh locations. **The McBSP should be halted before making changes to the serial port control register (SPCR), receive control register (RCR), transmit control register (XCR), and pin control register (PCR). Changes made to these registers without halting the McBSP could result in an undefined state.**

12–6 Change the footnote in Table 12–3:

|| **For C64x, RCER and XCER** are replaced by RCERE0 and XCERE0, respectively.

12–11 Change the CLKS_STAT (bit 6) read/write field in Figure 12–3:

*Figure 12–3. Pin Control Register (PCR)*

| 15 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsvd | XIOEN | RIOEN | FSXM | FSRM | CLKXM | CLKRM | Rsvd | CLKS_STAT | DX_STAT | DR_STAT | FSXP | FSRP | CLKXP | CLKRP |
| R,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW, +0 | R,+0 | **R,+0** | RW,+0 | R,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 |

12–19 Add a paragraph after the bullet at the end of section 12.3.1:

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled ($\overline{XRST}$ = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1) Wait for 2 CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2) Disable the transmitter ($\overline{XRST}$ = 0). This clears any XSYNCERR.
3) Reenable the transmitter ($\overline{XRST}$ = 1).

See also section 12.5.1.2 for details on initialization procedure.

| Page: | Change or Add: |
|---|---|
| 12–41 | Add a paragraph after the section header in section 12.3.7.5: |

### *12.3.7.5 Unexpected Transmit Frame Synchronization: XSYNCERR*

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled ($\overline{\text{XRST}}$ = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1) Wait for 2 CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2) Disable the transmitter ($\overline{\text{XRST}}$ = 0). This clears any XSYNCERR.
3) Reenable the transmitter ($\overline{\text{XRST}}$ = 1).

See also section 12.5.1.2 for details on initialization procedure.

12–50 Change the procedure in section 12.5.1.2:

### *12.5.1.2. McBSP and Sample Rate Generator Reset Procedure*

The McBSP and sample rate generator reset and initialization procedure is as follows:

1) Ensure that no portion of the McBSP is using the internal sample-rate generator signals CLKG and FSG (if necessary, clear $\overline{\text{RRST}}$ and/or $\overline{\text{XRST}}$ to 0). Clear $\overline{\text{FRST}}$ = $\overline{\text{GRST}}$ =0 in the SPCR. If the device has been reset ($\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{GRST}}$ = 0), this step is not required. CLKG and FSG are inactive low when $\overline{\text{GRST}}$ = 0.
2) Program the SRGR as required. Other control registers can be programmed if the respective portion (receiver/transmitter) is in reset.
3) Wait two CLKSRG clock source cycles for proper internal synchronization.
4) To use the sample rate generator, set $\overline{\text{GRST}}$ = 1 and wait 2 CLKG bit clocks for synchronization. Skip this step if the internal sample-rate generator is not used.
5) **On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator input clock (see section 12.5.2.1).**
6) **Set $\overline{\text{XRST}}$ to 1 to enable transmitter. When the transmitter is enabled for the first time after device reset, a transmit sync error (XSYNCERR) may occur. To avoid this:**
   - **Wait for 2 CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.**
   - **Disable the transmitter ($\overline{\text{XRST}}$ = 0). This clears any XSYNCERR.**
7) **If the DMA/EDMA is used to service the McBSP, setup data acquisition as desired and start the DMA/EDMA.**
8) **Set $\overline{\text{XRST}}$ and/or $\overline{\text{RRST}}$ to 1 to enable the corresponding section of the serial port. The McBSP is now ready to transmit and/or receive. If the CPU is used to service the McBSP, it can do so now using the polling or interrupt method described in section 12.3.2. If the DMA/EDMA is used instead, it services the McBSP automatically upon receiving the XEVT and/or REVT.**

| Page: | Change or Add: |
|---|---|

9) **If the internal sample rate generator is used to generate the frame sync signal, set $\overline{FRST}$ = 1 in the SPCR. FSG is generated on an active edge after 7–8 CLKG clocks have elapsed.**

12–57 Change the table header in Table 12–19:

*Table 12–19. Receive Frame Synchronization Selection*

| DLB in SPCR | **FSRM in PCR** | GSYNC in SRGR | Source of Receive Frame Synchronization | FSR Pin Function |
|---|---|---|---|---|

12–62 Change the Note in section 12.6:

> **Note:**
>
> **For C64x, RCER and XCER are replaced by RCERE0 and XCERE0, respectively.** Additional registers XCERE1, XCERE2, XCERE3, RCERE1, RCERE2, and RCERE3 are also used in this mode.

12–71 Change Table 12–22:

*Table 12–22. Receive Channel Enable Register Field Description*

| Name | Function |
|---|---|
| RCEA*n* 0 ≤ n ≤ 15 | **Receive channel enable A**<br>RCEA*n* = 0: Disables reception of the *n*th element in an even-numbered subframe in partition A<br>RCEA*n* = 1: Enables reception of the *n*th element in an even-numbered subframe in partition A |
| RCEB*n* 0 ≤ n ≤ 15 | **Receive channel enable B**<br>**RCEB*n*** = 0: Disables reception of the *n*th element in an odd-numbered subframe in partition B<br>**RCEB*n*** = 1: Enables reception of the *n*th element in an odd-numbered subframe in partition B |

12–80 Change the Clock Scheme description in Table 12–26:

*Table 12–26. SPI-Mode Clock Stop Scheme*

| CLKSTP | CLKXP | Clock Scheme |
|---|---|---|
| 0X | X | Clock stop mode disabled. Clock enabled for non-SPI mode. |
| 10 | 0 | Low inactive state without delay. The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of **CLKX**. |
| 11 | 0 | Low inactive state with delay. The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of **CLKX**. |
| 10 | 1 | High inactive state without delay. The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of **CLKX**. |
| 11 | 1 | High inactive state with delay. The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of **CLKX**. |

| **Page:** | **Change or Add:** |
|---|---|

12–81 Change the third paragraph in section 12.7:

Figure 12–55 is the timing diagram when CLKSTP = 10b. In this SPI transfer format, the transition of the first clock edge (CLKX) marks the beginning of data transfer, provided the slave enable (FSX/$\overline{SS}$) is already asserted. Data transfer is synchronized to the first clock edge. Figure 12–56 is the timing diagram when CLKSTP = 11b. Data transfer begins before the transition of the serial clock. Therefore, the transition of the slave enable signal FSX/$\overline{SS}$ from high to low, instead of the transition of the serial clock, marks the beginning of transfer in this SPI transfer format. **In SPI master mode, as well as SPI slave mode, the McBSP requires a FSX/SS edge for each transfer. This means the FSX/SS signal must toggle for each word.** The McBSP clock stop mode requires single-phase frames ((R/X)PHASE = 0) and one element per frame ((R/X)FRLEN = 0).

13–4 Add a footnote symbol to Timer 2 column and a footnote in Table 13–2:

*Table 13–2. Timer Registers*

| **Hex Byte Address** | | | **Name and Abbreviation** | **Description** | **Section** |
|---|---|---|---|---|---|
| **Timer 0** | **Timer 1** | **Timer 2**† | | | |

† Available only on C64x.

13–4 Change the figure title in Figure 13–2:

*Figure 13–2. TMS320C62x/C67x Timer Control Register (CTL)*

13–4 Add a new Figure 13–3. The subsequent figures are renumbered accordingly:

*Figure 13–3. TMS320C64x Timer Control Register (CTL)*

| 31–16 | 15 | 14–12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|
| Rsvd | SPND | Rsvd | TSTAT | INVINP | CLKSRC | C/$\overline{P}$ |
| R, +0 | RW, +0 | R, +0 | R, +0 | RW, +0 | RW, +0 | RW, +0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| $\overline{HLD}$ | GO | Rsvd | PWID | DATIN | DATOUT | INVOUT | FUNC |
| RW, +0 | RW, +0 | R, +0 | RW, +0 | R, +X | RW, +0 | RW, +0 | RW, +0 |

| **Page:** | **Change or Add:** |
|---|---|

13–5 Add a row (SPND bit) and a footnote in Table 13–3:

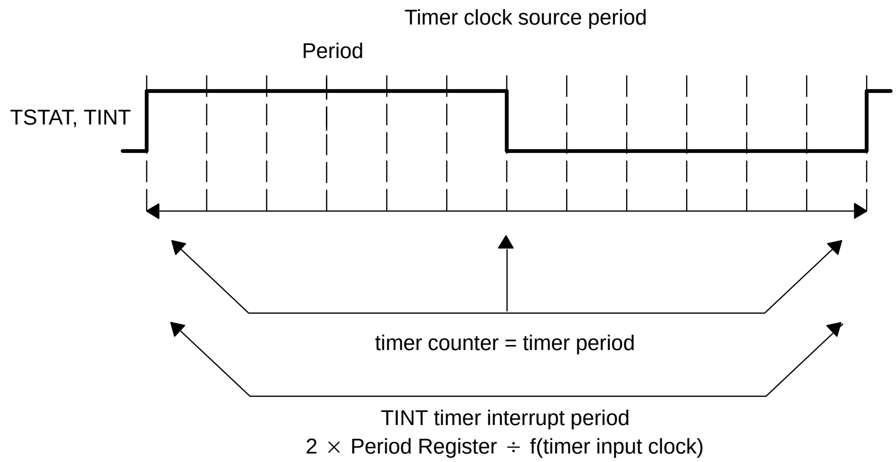
*Table 13–3. Timer Control Register (CTL) Field Descriptions*

| No. | Bitfield | Description | Section |
|---|---|---|---|
| **31–16** | Rsvd | Reserved. | |
| 15 | SPND† | Suspend Mode. Stops timer from counting during an emulation halt. Only affects operation, if the clock source is internal, CLKSRC =1. Reads always return a 0.<br>SPND=0: Timer continues to count during an emulation halt.<br>SPND=1: Timer stops counting during an emulation halt. | |
| **14–12** | Rsvd | Reserved. | |
| 11 | TSTAT | Timer status. Value of timer output. | 13.6 |

† For C64x only; for C621x/C671x, this bit is reserved. R,+0.

13–9 Change Figure 13–6:

*Figure 13–6. Timer Operation in Clock Mode (C/$\overline{P}$ = 1)*



| Page: | Change or Add: |
|---|---|

14–5 Add 8 rows to Table 14–4:

*Table 14–4. TMS320C64x Available Interrupts*

| Interrupt Selection Number | Interrupt Acronym | Interrupt Description |
|---|---|---|
| 10110b | Reserved | Not used |
| 10111b | UINT | UTOPIA **interrupt** |
| **11000b** | **–** | **Reserved** |
| **11001b** | **–** | **Reserved** |
| **11010b** | **–** | **Reserved** |
| **11011b** | **–** | **Reserved** |
| **11100b** | **–** | **Reserved** |
| **11101b** | **–** | **Reserved** |
| **11110b** | **VCPINT** | **VCP interrupt** |
| **11111b** | **TCPINT** | **TCP interrupt** |

14–10 Change the Note in section 14.5:

**Note:**

**After the registers** have been set, the interrupt flag register should be cleared **by the user to remove** any spurious transitions caused by the configuration.

17–5 Add a paragraph in section 17.2.2 before Figure 17–4:

When GPIO pins are configured as output pins, these pins do not have high-impedance capability. At reset, GPIO output pins default to the value in the GPIO value register (GPVAL), see section 17.2.3. If it is necessary to drive the GPIO output to the high-impedance state, the GPIO pins can be configured as an input pin and then changed to an output pin.

17–11 Change Figure 17–11. Add a footnote.

*Figure 17–11. GPIO Interrupt Polarity Register (GPPOL)*†

| 31–16 |
|---|
| Reserved |
| R, +0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPINT 15 POL | GPINT 14 POL | GPINT 13 POL | GPINT 12 POL | GPINT 11 POL | GPINT 10 POL | GPINT 9 POL | GPINT 8 POL | GPINT 7 POL | GPINT 6 POL | GPINT 5 POL | GPINT 4 POL | GPINT 3 POL | GPINT 2 POL | GPINT 1 POL | GPINT 0 POL |
| RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 | RW,+0 |

† For GPINT mapping with EDMA events and external interrupts, refer to device-specific datasheet.

| Page: | Change or Add: |
|---|---|

17–11 Change the bit numbers (No.) of GPINTxPOL in Table 17–10:

*Table 17–10. GPIO Interrupt Polarity Register (GPPOL) Bit Field Description*

| No. | Field | Description |
|---|---|---|
| **15:0** | GPINTxPOL | GPINTx Polarity. Applies to Pass Through Mode only.<br>GPINTxPOL = 0; GPINTx is asserted (high) based on a rising edge of GPx (effectively based on the value of the corresponding GPxVAL)<br>GPINTxPOL = 1; GPINTx is asserted (high) based on a falling edge of GPx (effectively based on the inverted value of the corresponding GPxVAL) |

18–2 Change the fourth paragraph in section 18.1:

All references to the term “slave devices are analogous to multi-PHYs (MPHYs) as referenced in the ATM Forum specification. **For multi-PHY systems, reference ATM Forum standard specification af-phy-0039.000. For single-PHY (single device) systems, reference ATM Forum standard specification af-phy-0017.000.**

18–5 Change the SLID bit range in Figure 18–3:

*Figure 18–3. UTOPIA Control Register (UCR)*

| 31 | 30 29 | 28 **24** | **23–22** | 21 18 | 17 | 16 |
|---|---|---|---|---|---|---|
| BEND | Reserved | SLID | **Rsvd** | XUDC | Rsvd | UXEN |
| RW, +0 | R, +0 | RW,+0 | **R,+0** | RW, +0 | R,+0 | RW, +0 |

| 15 | 14 | 13 6 | 5 2 | 1 | 0 |
|---|---|---|---|---|---|
| Rsvd | MPHY | Reserved | RUDC | Rsvd | UREN |
| R, +0 | RW, +1 | R, +0 | RW,+0 | R, +0 | RW, +0 |

18–6 Change the SLID bit range in Table 18–3:

*Table 18–3. UTOPIA Control Register (UCR) Bit Field Description*

| Bit | Field | Description | Section |
|---|---|---|---|
| **28:24** | SLID | Slave ID: Applicable in MPHY mode. SLID is a programmable 5-bit PHY address used to identify the UTOPIA in a MPHY set up. Does not apply to single-PHY slave operation. | 18.4.7 |

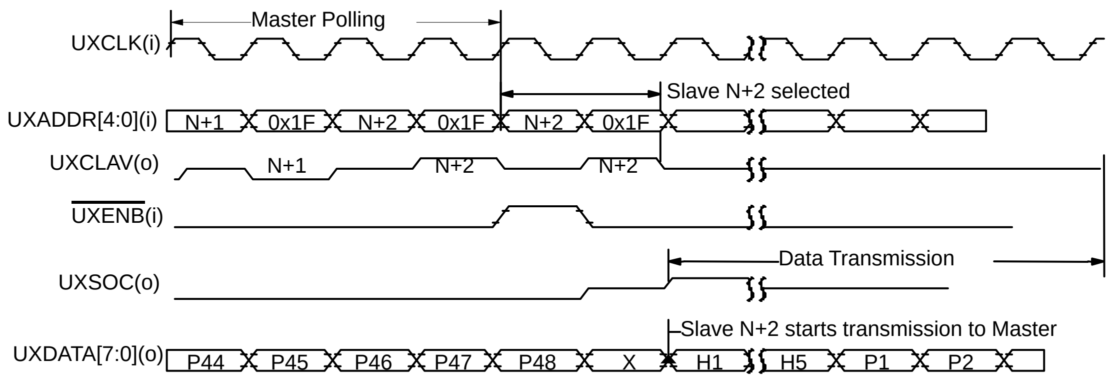
| Page: | Change or Add: |
|---|---|

18–7 Change the paragraph in section 18.3:

The ATM Forum specification for UTOPIA Level 2 specifies the order in which header and payload information is sent across the ATM-PHY interface. The header information is sent first followed by the 48-byte payload. A standard ATM cell is 53 bytes (5-byte header + 48-byte payload). The UTOPIA peripheral also supports a non-standard ATM cell (R/XUDC = 1 to 11) of size 54 to 64 bytes. **The UTOPIA transmit queue and the receive queue each accommodates two cells. The number of cells each queue accommodates is not dependent upon cell size.** Figure 18–4 and Figure 18–5 show the standard and non-standard cell transfer format, respectively, with reference to time.

18–11 Change the UXCLK signal in Figure 18–7:

*Figure 18–7. ATM Controller Slave Transmit Timing*

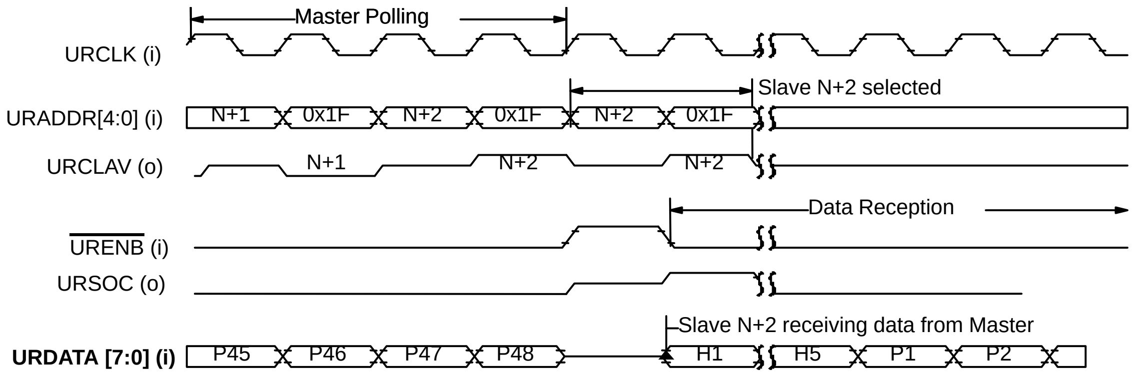



18–13 Change the paragraph in section 18.4.3:

The UTOPIA slave will agree for transmission to the master by asserting its UXCLAV signal when there is at least one cell available in the slave-transmit queue. If the slave cannot provide the next cell in a contiguous fashion, it deasserts its UXCLAV in the cycle following the completion of the current cell transmission. The UXCLAV remains asserted if the slave has another cell available to transfer to the master. **The UXCLAV signal is not available to the CPU, so the CPU cannot determine if the transmit queue is empty.** The master may disable RxEnb* on its side (connected to the $\overline{\text{UXENB}}$ pin for this ATMC slave), which causes the UTOPIA slave to hold off the next cell transfer until the master indicates as such.

| Page: | Change or Add: |
|---|---|
| 18–14 | Change the URDATA signal name in Figure 18–8: |

*Figure 18–8. ATM Controller Slave Receive Timing*



| Page: | Change or Add: |
|---|---|
| 18–16 | Change section 18.5.1: |

### 18.5.1 EDMA Setup for UTOPIA Transmitter

As mentioned in section 18.4.6, the UTOPIA transmitter generates a UXEVT synchronization event to the EDMA when at least one cell-packet space is available in the slave-transmit queue. EDMA channel 40 is dedicated to the UXEVT event. Per UXEVT synchronization event, one frame of cell-packet data is transferred to the slave-transmit queue. A standard cell-packet comprises of 14 words (56 bytes), while a nonstandard cell-packet comprises of 14, 15, or 16 words (56, 60, or 64 bytes). The EDMA access to UTOPIA is always 32 bits. The EDMA source address should point to the UTOPIA source buffer in the DSP memory (internal or external). **The EDMA destination address should point to the slave-transmit queue data port UXQ.**

**An example of an EDMA setup to service a UTOPIA transmitter is shown below:**

```
//**************************************************\
* EDMA XMIT Channel Configuration
\**************************************************/
void
setup_EdmaXmit() {
  /* Setup EDMA registers */
  cfgEdmaOut0.opt = EDMA_OPT_RMK(
    EDMA_OPT_PRI_HIGH,
    EDMA_OPT_ESIZE_32BIT,
    EDMA_OPT_2DS_YES,
    EDMA_OPT_SUM_INC,
    EDMA_OPT_2DD_NO,
    EDMA_OPT_DUM_NONE,
    EDMA_OPT_TCINT_YES,
    EDMA_OPT_TCC_OF(TCCXMIT0NUM),
    EDMA_OPT_TCCM_OF(TCCXMIT0NUM>>4),
    EDMA_OPT_ATCINT_NO,
    EDMA_OPT_ATCC_OF(0),
    EDMA_OPT_PDTS_DISABLE,
    EDMA_OPT_PDTD_DISABLE,
    EDMA_OPT_LINK_YES,
    EDMA_OPT_FS_NO
  );
  cfgEdmaOut0.src = EDMA_SRC_RMK(DSP_out0);
  cfgEdmaOut0.dst = EDMA_DST_RMK(UTOP_XMTQ_ADDR);
  cfgEdmaOut0.cnt = EDMA_CNT_RMK((NUM_XMIT_CELL–1), xcell_sz);
  cfgEdmaOut0.idx = EDMA_IDX_RMK((xcell_sz*4), 0); /* xcell_sz
= # of 32–b words in each transmit cell */
  cfgEdmaOut0.rld = EDMA_RLD_RMK(0, hEdmaNullTbl);

  /* Copy above setup to the EDMA Handle */
  EDMA_config(hEdmaOut0, &cfgEdmaOut0);
}
```

**Page:** **Change or Add:**

18–17 Change section 18.5.2:

### 18.5.2 EDMA Setup for UTOPIA Receiver

As mentioned in section 18.4.6, the UTOPIA receiver generates an UREVT synchronization event to the EDMA when the slave-receive queue has space for at least one cell-packet. EDMA channel 32 is dedicated to the UREVT event. Per UREVT synchronization event, one frame of cell-packet data is read from the slave-receive queue via the data port URQ. A standard cell-packet comprises of 14 words (56 bytes), while a nonstandard cell-packet comprises of 14, 15, or 16 words (56, 60, or 64 bytes). **The EDMA destination address should point to the destination buffer in the DSP memory (internal or external).**

**An example of an EDMA setup to service a UTOPIA receiver is shown below:**

```
/**************************************************\
* EDMA RECV Channel Configuration
\**************************************************/
void
setup_EdmaRecv() {
  /* Setup EDMA registers */
  cfgEdmaIn0.opt = EDMA_OPT_RMK(
    EDMA_OPT_PRI_MEDIUM,
    EDMA_OPT_ESIZE_32BIT,
    EDMA_OPT_2DS_NO,
    EDMA_OPT_SUM_NONE,
    EDMA_OPT_2DD_YES,
    EDMA_OPT_DUM_INC,
    EDMA_OPT_TCINT_YES,
    EDMA_OPT_TCC_OF(TCCRECV0NUM),
    EDMA_OPT_TCCM_OF(TCCRECV0NUM>>4),
    EDMA_OPT_ATCINT_NO,
    EDMA_OPT_ATCC_OF(0),
    EDMA_OPT_PDTS_DISABLE,
    EDMA_OPT_PDTD_DISABLE,
    EDMA_OPT_LINK_YES,
    EDMA_OPT_FS_NO
  );
  cfgEdmaIn0.src = EDMA_SRC_RMK(UTOP_RCVQ_ADDR);
  cfgEdmaIn0.dst = EDMA_DST_RMK(DSP_in0);
  cfgEdmaIn0.cnt = EDMA_CNT_RMK((NUM_RECV_CELL–1), rcell_sz);
  cfgEdmaIn0.idx = EDMA_IDX_RMK((rcell_sz*4), 0); /* rcell_sz
= # of 32–b words in each receive cell */
  cfgEdmaIn0.rld = EDMA_RLD_RMK(0, hEdmaNullTbl);

  /* Copy above setup to the EDMA Handle */
  EDMA_config(hEdmaIn0, &cfgEdmaIn0);
}
```

| Page: | Change or Add: |
|---|---|

18–26 Change all occurrences of XCPP, XCFP, XQSP, RCPP, RCFP, and RQSP in Table 18–11:

*Table 18–11. Error Interrupt Enable Register (EIER) Bit Field Description*

| No. | Field | Description |
|---|---|---|
| 18 | **XCPE** | Transmit Clock Present Interrupt Enable.<br>**XCPE** = 0: Transmit Clock Present interrupt disabled.<br>**XCPE** = 1: Transmit Clock Present interrupt enabled. |
| 17 | **XCFE** | Transmit Clock Failed Interrupt Enable.<br>**XCFE** = 0: Transmit Clock Failed interrupt disabled.<br>**XCFE** = 1: Transmit Clock Failed interrupt enabled. |
| 16 | **XQSE** | Transmit Queue Stall Interrupt Enable.<br>**XQSE** = 0: Transmit Queue Stall interrupt disabled.<br>**XQSE** = 1: Transmit Queue Stall interrupt enabled. |
| 2 | **RCPE** | Receive Clock Present Interrupt Enable.<br>**RCPE** = 0: Receive Clock Present interrupt disabled.<br>**RCPE** = 1: Receive Clock Present interrupt enabled. |
| 1 | **RCFE** | Receive Clock Failed Interrupt Enable.<br>**RCFE** = 0: Receive Clock Failed interrupt disabled.<br>**RCFE** = 1: Receive Clock Failed interrupt enabled. |
| 0 | **RQSE** | Receive Queue Stall Interrupt Enable.<br>**RQSE** = 0: Receive Queue Stall interrupt disabled.<br>**RQSE** = 1: Receive Queue Stall interrupt enabled. |

18–28 Change the third paragraph in section 18.10:

The cell-packet format in Figure 18–14 through Figure 18–19 indicates the data stored in the DSP memory (internal or external) and in the transmit/receive queues. Only the ATM data, including the header and payload information but not **the Dummy bytes** (as shown in **Figure 18–14 and Figure 18–15**), is actually sent or received across the UTOPIA pins.

18–31 Change the second paragraph in section 18.11:

Table 18–12 shows **the recommended reset values** of the UTOPIA pins. The UTOPIA pins have no internal pull-up or pull-down resistors. At reset, all outputs are driven to a high-impedance state to facilitate MPHY operation. **All input pins should be pulled externally to bring inputs to a known state when not driven as shown in Table 18–12.**

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265