TMS320C80 (MVP) Transfer Controller User's Guide

Literature Number: SPRU105B
March 1998

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Preface

Read This First

About This Manual

The TMS320C80 MVP (multimedia video processor) is Texas Instruments first single-chip multiprocessor DSP (digital signal processor) device. The MVP contains five powerful, fully programmable processors: a master processor (MP) and four parallel processors (PPs). The MP is a 32-bit RISC (reduced instruction set computer) with an integral, high-performance IEEE-754 floating-point unit. Each PP is an advanced 32-bit DSP. In addition to having similar processing capabilities as conventional DSPs, each PP has advanced features to accelerate operation on a variety of data types.

The MVP supports a variety of parallel-processing configurations, which facilitates a wide range of multimedia and other applications that require high processing speeds. Applications include image processing, two- and three-dimensional and virtual reality graphics, audio/visual digital compression, and telecommunications.

This manual describes the MVP transfer controller (TC). The TC is a combined DMA (direct memory access) machine and memory interface that queues, prioritizes and services the data requests and cache misses of the five processors in the MVP. The TC provides an interface between the MP, the PPs. the video controller (VC), and external (off-chip) memory. This manual provides information about the TC features, functional blocks, and operation; it also includes examples of block write operations for big- and little-endian modes.

Related Documentation from Texas Instruments

The following books describe the TMS320C80 MVP and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

TMS320C8x System-Level Synopsis (literature number SPRU113) describes the'C8x features, development environment, architecture, memory organization, and communication network (the crossbar).

TMS320C8x Master Processor User's Guide (literature number SPRU109) provides information about the master processor (MP) features, architecture, operation, and assembly language instruction set; it also includes sample applications that illustrate various MP operations.

TMS320C8x Parallel Processor User's Guide (literature number SPRU110) provides information about the parallel processor (PP) features, architecture, operation, and assembly language instruction set; it also includes software applications and optimizations.

TMS320C80 Video Controller User's Guide (literature number SPRU111) provides information about the video controller (VC) features, architecture, and operation; it also includes procedures and examples for programming the serial register transfer (SRT) controller and the frame timer registers.

TMS320C8x Multitasking Executive User's Guide (literature number SPRU112) provides information about the multitasking executive software features, operation, and interprocessor communications; it also includes a list of task error codes.

TMS320C80 (MVP) Code Generation Tools User's Guide (literature number SPRU108) describes the 'C8x code generation tools. This manual provides information about the features and operation of the linker and the master processor (MP) and parallel processor (PP) C compilers and assemblers. It also includes a description of the common object file format (COFF) and shows you how to link MP and PP code.

TMS320C80 (MVP) C Source Debugger User's Guide (literature number SPRU107) describes the 'C8x master processor and parallel processor C source debuggers. This manual provides information about the features and operation of the debuggers and the parallel debug manager; it also includes basic information about C expressions and a description of progress and error messages.

TMS320C8x Software Development Board Installation Guide (literature number SPRU150B), included with the TMS320C8x SDB, provides information about how to install and use the SDB.

TMS320C8x Software Development Board Programmer's Guide (literature number SPRU178), included with the TMS320C8x SDB, provides descriptions of hardware functions, complete API references, theory of operation, and example code for the SDB.

TMS320C80 Digital Signal Processor Data Sheet (literature number SPRS023) describes the features of the TMS320C80 and provides pinouts, electrical specifications, and timings for the device.

TMS320C80 to TMS320C82 Software Compatibility User's Guide (literature number SPRU154) describes how to port software developed for one of these devices to the other. It also presents a set of software compatibility guidelines for developing software that will run on either device.

Implementation of the Vector Maximum Search Benchmark on the TMS320C8x Parallel Processor Application Report (literature number SPRA087) uses the *Vector Maximum Search* benchmark to demonstrate the efficient performance of the TMS320C8x parallel processors. This manual describes a software implementation that uses the parallel processor's advanced assembly language features to implement this benchmark.

Modified Goertzel Algorithm in DTMF Detection Using the TMS320C80 Application Report (literature number SPRA066) describes the C-callable Goertzel dual-tone multi-frequency (DTMF) detection algorithm implementation on one of the TMS320C80's parallel processors.

Acoustic Echo Cancellation — Algorithms and Implementation on the TMS320C8x Application Report (literature number SPRA063) describes the implementation of an integral N-tap digital acoustic echo canceller on the TMS320C8x parallel processor. The report presents a brief discussion of generic echo cancellation algorithms. The implementation considerations for a 512-tap (64-ms span) echo canceller on the TMS320C8x are described in detail, as well as the software logic and flow for each program module.

Interfacing DRAM to the TMS320C80 Application Report (literature number SPRA056) describes an interface between the 'C80 and the TMS417400 DRAM(s). The report also describes the interface's connection to 4MB and 8MB SIMMs, a timing analysis for the proposed design, and a complete set of schematics.

TMS320C80 (MVP) Code Generation Tools User's Guide (literature number SPRU108) describes the 'C8x code generation tools. This manual provides information about the features and operation of the linker and the master processor (MP) and parallel processor (PP) C compilers and assemblers. It also includes a description of the common object file format (COFF) and shows you how to link MP and PP code.

TTMS320C8x Software Development Board Installation Guide (literature number SPRU150B), included with the TMS320C8x SDB, provides information about how to install and use the SDB.

TMS320C8x Software Development Board Programmer's Guide (literature number SPRU178), included with the TMS320C8x SDB, provides descriptions of hardware functions, complete API references, theory of operation, and example code for the SDB.

Acoustic Echo Cancellation — Algorithms and Implementation on the TMS320C8x Application Report (literature number SPRA063) describes the implementation of an integral N-tap digital acoustic echo canceller on the TMS320C8x parallel processor. The report presents a brief discussion of generic echo cancellation algorithms. The implementation considerations for a 512-tap (64-ms span) echo canceller on the TMS320C8x are described in detail, as well as the software logic and flow for each program module.

Interfacing DRAM to the TMS320C80 Application Report (literature number SPRA056) describes an interface between the 'C80 and the TMS417400 DRAM(s). The report also describes the interface's connection to 4MB and 8MB SIMMs, a timing analysis for the proposed design, and a complete set of schematics.

TMS320C8x (DSP) — Fundamental Graphic Algorithms Application Book (literature number SPRA069) contains application notes that describe specific 'C8x capabilities. These notes are as follows:

TMS320C80 H.320 Software Library White Paper (literature number SPRY002) describes the TMS320C80's single-chip implementation of the H.320 videoconferencing standard.

TMS320C80 Digital Signal Processor Data Sheet (literature number SPRS023) describes the features of the TMS320C80 and provides pinouts, electrical specifications, and timings for the device.

TMS320C8x (DSP) — Fundamental Graphic Algorithms Application Book (literature number SPRA069) contains application notes that describe specific 'C8x capabilities. These notes are as follows:

Interfacing SDRAM to the TMS320C80 Application Report (literature number SPRA055) describes an interface between the 'C80 and the TMS626802-10 SDRAM(s). It illustrates the operation of the SDRAM interface and provides schematics for a baseline SDRAM interface to the TMS320C80.

If You Need Assistance...

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(713) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
Microcontroller Hotline	(713) 274-2370	Fax: (713) 274-4203	Email:*H370@msg.ti.com
Microcontroller Modem BBS	(713) 274-3700 8-N-1		

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:			
Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32	Email: epic@ti.com
Deutsch	+49 8161 80 33 11 or +33 1 30 70 11 68		
English	+33 1 30 70 11 65		
Francais	+33 1 30 70 11 64		
Italiano	+33 1 30 70 11 67		
EPIC Modem BBS	+33 1 30 70 11 99		
European Factory Repair	+33 1 93 22 25 40		
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10	

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200	

❑ Japan

Product Information Center (in Japan)	+0120-81-0026 (in Japan)	Fax: +0120-81-0036
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail:
Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: comments@books.sc.ti.com

Note:When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Trademarks

PC, MS-DOS, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

HP-UX is a registered trademark of Hewlett-Packard Company.

SunOS is a registered trademark of SunSoft, Inc., a subsidiary of Sun Microsystems Incorporated.

SPARCstation is licensed exclusively to Sun Microsystems, Inc.

XDS and XDS510 are registered trademarks of Texas Instruments.

Xilinx is a trademark of Xilinx, Inc.

Contents

Figures

Tables

Chapter 1

Overview of the Transfer Controller

The *transfer controller* provides a common interface for the TMS320C80 processors—the *master processor* (MP), the *parallel processors* (PPs), the *video controller* (VC)—and external (off-chip) memory. In addition, the TC performs block data transfers (packet transfers) between on-chip and/or off-chip locations.

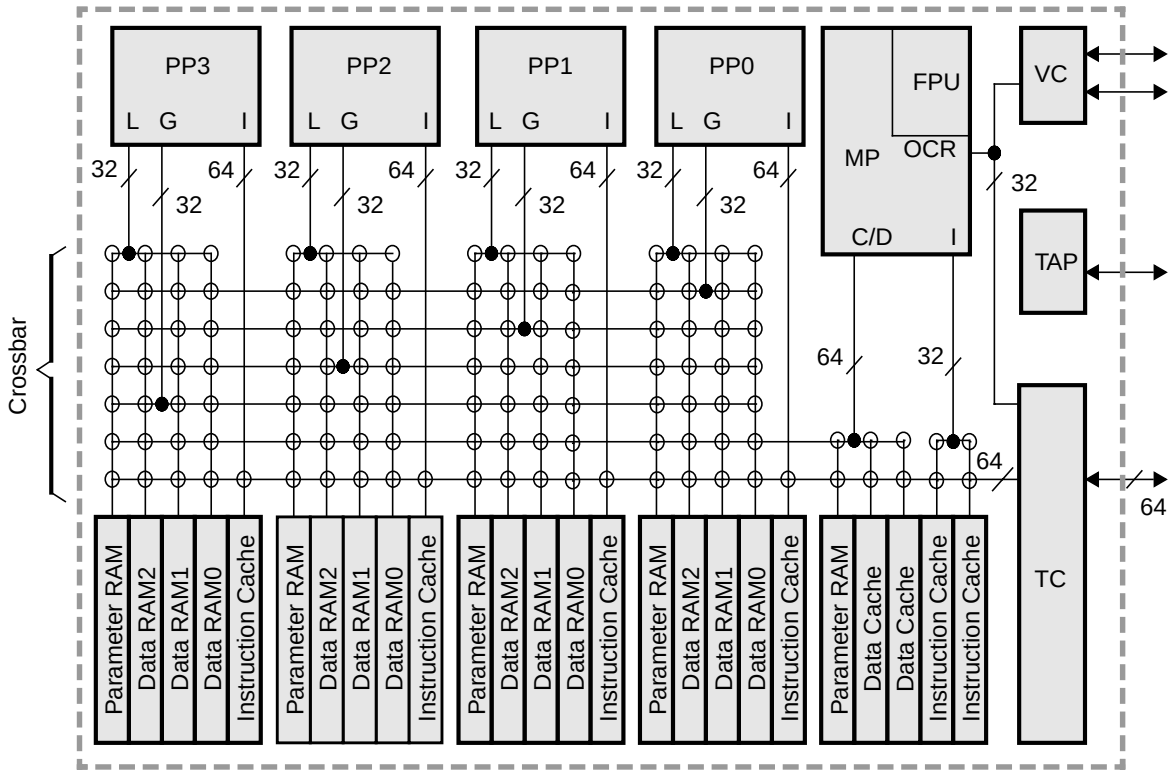
Topics in this chapter include:

Topic **Page**

1.1 Transfer Controller Description

Figure 1–1 shows that the TC has a 64-bit interface to the TMS320C80's crossbar which provides access to all on-chip data RAMs, parameter RAMs, and instruction and data caches. A second 64-bit interface provides off-chip access.

Figure 1–1. TMS320C80 Block Diagram Showing Data Paths



Note: L = Local port; G = Global port; I = Instruction port; FPU = Floating-point unit; OCR = On-chip register port; C/D = Cache/data port; TAP = Test access port; TC = Transfer controller; VC = Video controller; MP = Master processor; PP0-3 = Parallel processors 0-3

The TC services data requests and cache misses by the MP and PPs. Cache logic within each processor automatically requests TC operations. The requested data is moved from its memory location to the appropriate subblock in the cache array of the requesting processor. The TC also saves

the dirty subblocks of the MP's data cache when required by the MP. A separate cache controller within the TC can preempt program-controlled packet transfers to service cache misses. A dedicated cache buffer within the TC allows cache requests to be serviced without having to empty packet data from a separate packet transfer buffer.

TC operations involving the data RAMs are usually performed in response to *packet transfer* requests (PTs) from the PPs, MP, video controller, or external devices. Packet transfers provide an extremely flexible method of transferring data between on-chip and/or off-chip memory.

The TC can not only transfer multi-dimensional blocks of data between on-chip processors, it can assist in processing data moved to and from external memory. For example, in graphics and imaging, it is common for the TC to fetch data from an image region as a two-dimensional X/Y array and bring the data on-chip for processing as a linear array. After processing, the results can be stored off-chip as an X/Y array. By automatically transforming data this way, the TC adds to the efficiency of on-chip processors.

The TC also provides data directly to the processors using *direct external access* (DEA) cycles. DEA cycles allow the PPs to access off-chip memory. DEA cycles also allow the MP to bypass its data cache when accessing external memory.

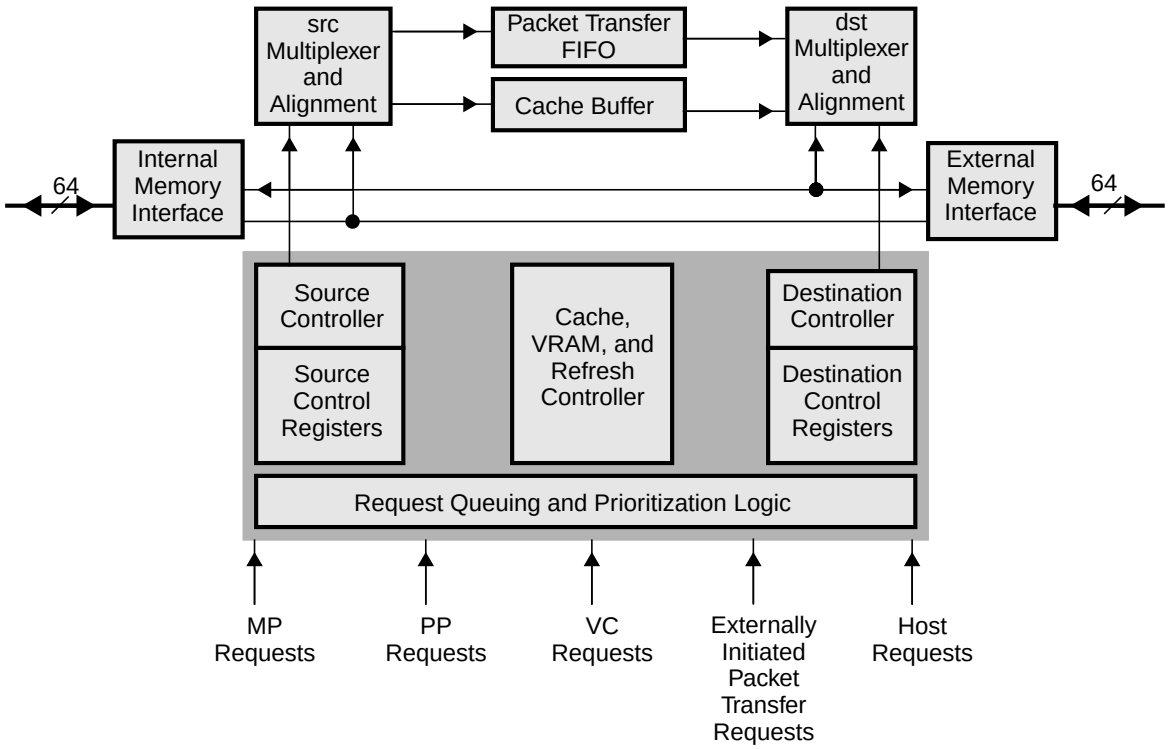
In addition to servicing cache misses and packet transfers, the TC also performs external SDRAM/DRAM refreshes, and performs the serial register transfer cycles required by the VC to update VRAM-based display/capture buffers. The TC prioritizes the various requests and, when necessary, time shares the external memory interface between packet transfers.

The TC also allows an external host to gain access to the external memory through the use of its host request and acknowledge mechanism.

1.2 Functional Blocks

Figure 1–2 shows a high-level block diagram of the TC. Following the diagram is a brief description of each of the major blocks.

Figure 1–2. Transfer Controller Block Diagram



1.2.1 Request Queuing and Prioritization Logic

The TC evaluates all requests from the MP, PPs, VC, externally initiated packet transfer (XPT) pins, and the external host and then services these requests based on a fixed prioritization scheme. When multiple requests of the same priority are pending, the TC services them on a round robin basis (see section 2.1.6, *Arbitrating Requests of the Same Priority*).

1.2.2 Cache, VRAM, and Refresh Controller

The TC contains a programmable refresh controller that automatically generates DRAM and SDRAM refresh cycles as required by the external memory system. The cache control logic generates the addresses necessary to perform the cache fills (and write-backs) as requested by the MP and PPs. VC shift register transfer requests are handled by the VRAM control logic.

1.2.3 Source/Destination Controllers

The TC has two independent controllers that handle packet transfers:

- The *source (src) controller* generates the addresses necessary to fetch the data from the source memory. When a packet transfer request is submitted to the TC, the packet transfer request contains a number of parameters specifying how the source data is to be accessed. These parameters are loaded into the source control registers and are used to generate the source addresses.
- The *destination (dst) controller* generates the addresses needed to write the packet data into the destination memory area. A set of parameters similar to those of the source controller are loaded into the destination control registers to generate the addresses.

The source and destination controllers can address both on-chip and off-chip memory.

1.2.4 Internal Memory Interface

The TC's internal memory interface provides access to on-chip memory via the TMS320C80's crossbar. The 64-bit data bus can transfer zero to eight bytes per cycle.

1.2.5 External Memory Interface

The external memory interface provides access to all off-chip memory and peripherals. A state sequencer generates the cycles and control signals necessary to interface to a variety of RAM/peripheral device types. The 64-bit data bus can transfer zero to eight bytes per cycle and provide dynamic bus size support for 8-, 16-, 32-, and 64-bit wide devices.

1.2.6 Source/Destination Multiplexer and Alignment and Packet Transfer FIFO

During packet transfers, data alignment is limited to byte-alignment. Because the source and destination controllers are independent, they can each fetch or store zero to eight bytes every cycle. This means that the alignment of the source and destination addresses with respect to each other can change constantly. To support the fluctuating alignment, the TC contains a packet transfer FIFO (first-in, first-out buffer) and alignment logic. The packet transfer FIFO is a 16-byte FIFO that can be simultaneously loaded and emptied with zero to eight bytes from the source and zero to eight bytes to the destination. The source multiplexer/aligner extracts the appropriate bytes from the source and stores them in the FIFO, contiguous to the previous source bytes. The destination multiplexer/aligner then extracts the *oldest* bytes from the FIFO and aligns them to the correct position in currently addressed destination (eight-byte) doubleword. Transfer alignment and FIFO operation is automatic and transparent to the programmer.

1.2.7 Cache Buffer

The cache buffer is an eight-byte buffer similar to the packet transfer FIFO in operation, and is used during cache and DEA operations. Transfers in and out of the caches are always eight bytes wide. The cache buffer helps align the data in cases where the external memory bus is less than 64 bits wide. The cache buffer allows higher priority cache and DEA requests to be serviced in the middle of a packet transfer without having to first empty the packet data currently in the packet transfer FIFO.

Chapter 2

Transfer Controller Requests

The TC handles many types of requests from the other processors and external controllers. To ensure optimum system performance, these requests are prioritized by their urgency and importance. Because the TC operates at these priorities, its own priority on the internal crossbar can vary from cycle to cycle. This chapter discusses how the TC prioritizes these requests. This chapter also describes two types of requests: cache-service and DEA requests.

Topics in this chapter include:

Topic **Page**

2.1 Request Types and Priorities

Figure 2–1 shows the various requests that the TC can receive prioritized from top to bottom. An explanation of each priority follows. When multiple requests of the same priority are received, the TC will round robin between them. The following sections describe the request types and their priorities.

2.1.1 Bus Request

An external device, such as a system host, must be able to access the bus whenever necessary. This is the highest-priority request that the TC can receive.

2.1.2 Video Controller SRT request

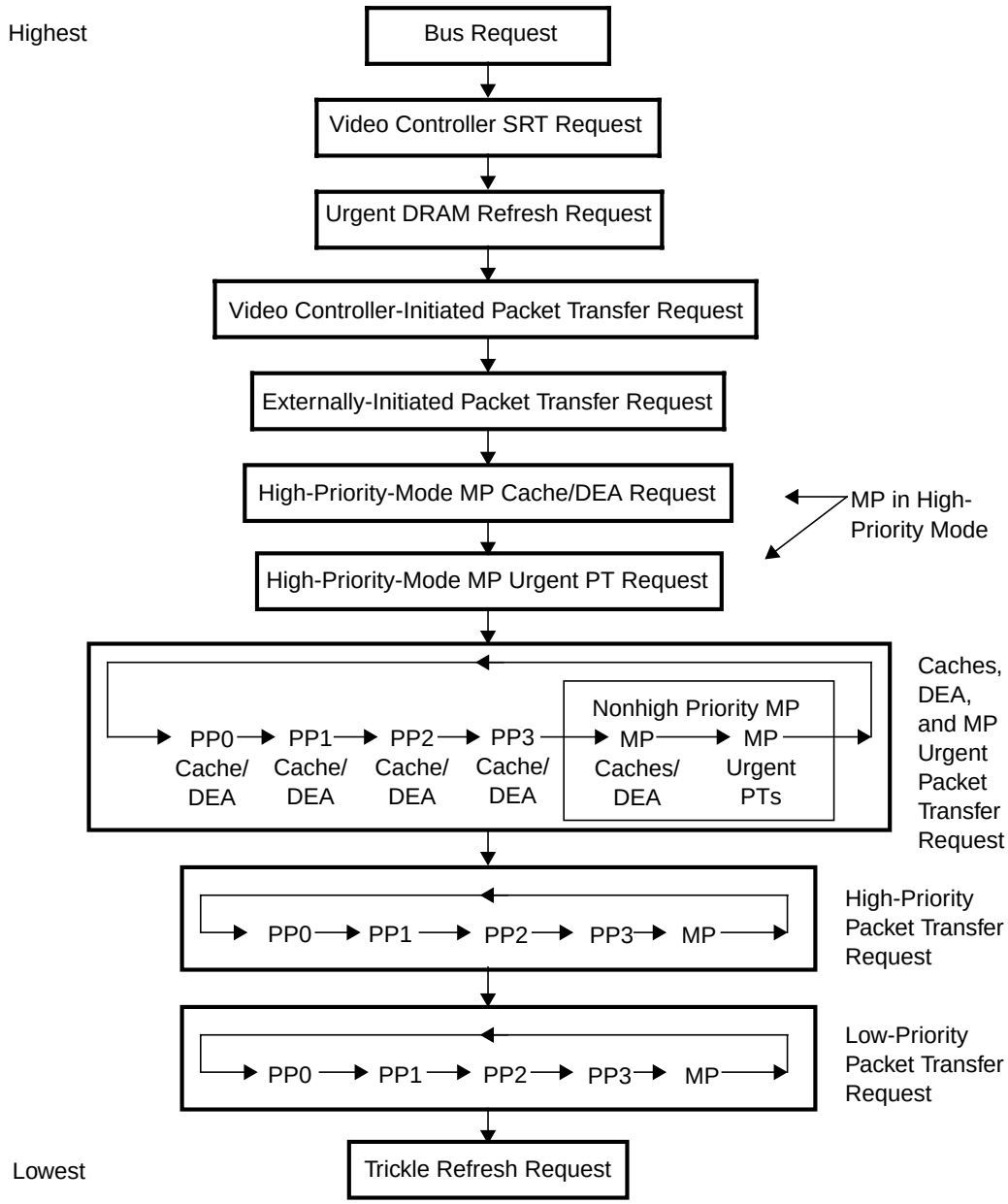
The VC SRT requests receive the second-highest priority so that time-critical VRAM transfer cycles occur without disruption to video display or capture.

2.1.3 Refresh Requests

You program the TC's refresh controller logic to generate refresh requests at regular intervals. As seen in Figure 2–1, a refresh request can occur at one of two priority levels:

- *Urgent DRAM refresh request.* Because VC and host-request cycles occur only intermittently, DRAM refreshes are prioritized below them.
- *Trickle refresh request.* Trickle refresh cycles have the lowest priority of all possible requests types. These are performed only when the external bus is idle and the refresh backlog is not 0. This helps lower the backlog and reduce the likelihood of requesting a high-priority urgent refresh at a later time.

Figure 2–1. Transfer Controller's Request Prioritization



2.1.4 Packet Transfer Requests

A *packet transfer* (PT) is a transfer of data blocks between two areas of TMS320C80 memory. Data is transferred by the transfer controller from a source (*src*) memory area to a destination (*dst*) memory area. The source and destination areas can be either on-chip or off-chip memory.

Figure 2–1 shows that packet transfers may be initiated by the master processor (MP), parallel processor (PPs), video controller (VC), or external devices as requests to the TC. Packet transfers initiated by the VC and external devices take precedence over packet transfers initiated by a TMS320C80 processor. Note that each processor can have only one active priority for its packet transfer. Parallel processors are restricted to high- or low-priority transfers, whereas the master processor can submit a packet-transfer request at any priority level.

The possible priority levels of packet transfers are as follows:

- *Video controller-initiated packet-transfer (VCPT) request.* Packet transfers requested by the VC may be performing video display or capture functions, and are prioritized above externally initiated packet-transfer requests.
- *Externally initiated packet-transfer (XPT) request.* Externally initiated packet transfers can have an effect on system performance and are given higher priority than cache and DEA requests and any packet-transfer requests initiated by a TMS320C80 processor.
- *MP urgent packet-transfer request.* If the MP is not operating in high-priority mode, packet transfers at this priority level share the same priority as MP and PP cache-service and DEA requests.
- *High-priority mode MP urgent packet-transfer request.* These requests can also be essential to MP high-priority tasks. They are prioritized below (high-priority) MP cache and DEA requests so as not to stall the MP's execution. However requests at this level have a higher priority than cache-service and DEA requests initiated by the PP.
- *High-priority packet transfer request.* High-priority packet transfers imply that the requesting processor is waiting for data to finish transferring or that the TC needs to take priority over the PPs for crossbar access to optimize external bus bandwidth. These requests have lower priority than cache-service and DEA requests.

- *Low-priority packet transfer request.* Low-priority packet transfers imply that the processor is not waiting for data. This is the second-lowest priority level for requests, above trickle-refresh requests.

Higher priority transfers interrupt lower priority transfers. When this occurs, the lower priority transfer is suspended by the TC, and the current state of the transfer is saved. When the higher priority transfer is completed, the suspended transfer resumes automatically at the point at which it was interrupted.

For details about the mechanics of a packet transfer, see Chapter 3, *Transfer Controller Requests*.

2.1.5 Cache-Service and DEA Requests

Cache-service requests occur as a result of a miss to a PP or MP instruction or data cache. Direct external access (DEA) requests allow the PPs to access data in external memory directly, and allow the MP to bypass its data cache.

Figure 2–1 shows that the number of priority levels allocated to cache-service and DEA requests depends on the operating mode of the MP. If the MP is operating in high-priority mode MP and PP requests operate at different priority levels. If the MP is not in high-priority mode, MP and PP requests operate at the same priority level. The priority levels of cache-service and DEA requests with respect to other possible requests are as follows:

- *High-priority mode MP cache/DEA request.* If the MP is in high-priority mode, MP cache-service and DEA requests have a lower priority than externally initiated packet-transfer requests, but have a higher priority level than PP cache-service and DEA requests and any TMS320C80 packet-transfer requests (including urgent packet-transfer requests initiated by the MP). This allows maximized system performance through the quick performance of interrupt service routines and other high-priority MP tasks.

- *PP and non-high-priority MP cache/DEA request.* If the MP is not in high-priority mode, all cache-service and DEA requests have the same priority level, which is less than externally initiated packet transfer requests, has the same priority level as MP urgent packet-transfer requests, but has a higher priority than any other type of packet transfer initiated by a TMS320C80 processor. It is important that these be serviced quickly, because the processor is idle until the request is serviced. For further details about cache-service and DEA requests, refer to section 2.4, *Cache-Service Requests* and section 2.5, *DEA Requests*.

2.1.6 Arbitrating Requests of the Same Priority

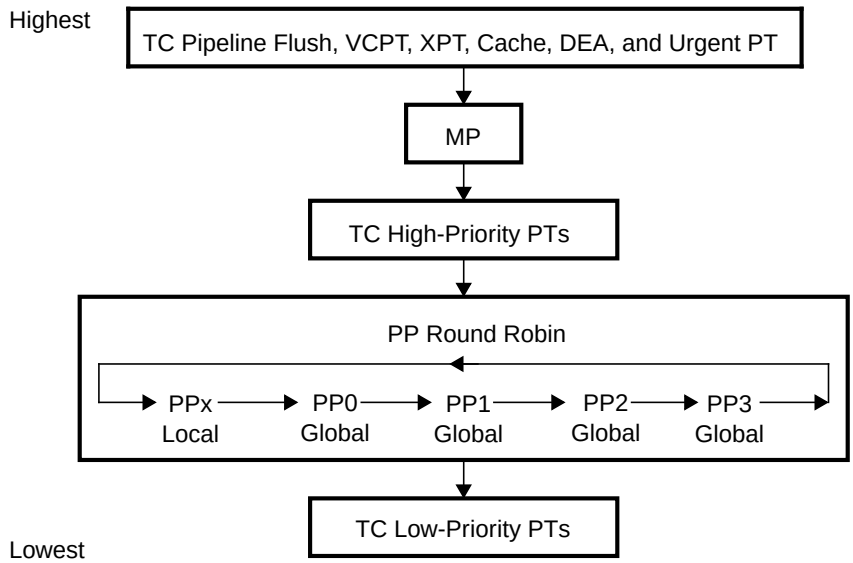
Whenever the TC receives multiple requests of the same priority from different processors, it *round robins* between them. The round robin is a fixed cyclical-priority scheme. This means that no processor can be removed from the round robin, and the order of processors within the cycle cannot be changed. When a processor's request is completed, the round-robin token is passed to the next processor in the chain with a pending request. This prevents any one processor from monopolizing the TC when requests of equal priority from other processors need to be serviced.

The round robin can be disabled by setting the T bit in the MP's CONFIG register to a 1. When this is done, the MP is always serviced first if there are multiple requests of the same priority, followed by PP0, PP1, PP2, and PP3.

2.2 Effect of Transfer Controller Requests on Crossbar Priority

The TC's priority on the crossbar changes dynamically according to the level of request that the TC is servicing. This is shown in Figure 2–2.

Figure 2–2. Crossbar Priority



The TC's priority is assigned as follows:

- The TC operates above the MP priority when servicing an urgent-priority packet transfer request, a cache service request, a DEA request, a VCPT, an XPT, or when flushing its pipeline. TC pipeline flushing occurs when the TC receives an urgent refresh, video controller, or host interface request or a soft reset, none of which may begin with external cycles waiting to be completed. Pipeline flushing occurs only occasionally, and locks out the MP for only a short period of time.
- The TC has priority above the PPs, but below the MP, for high-priority packet transfers. This gives the TC maximum possible priority without locking out the MP, which could have undesirable system implications.

- The TC has priority below the PPs if it is performing a low-priority packet transfer. This prevents the TC from stealing crossbar bandwidth from the PPs.

Whenever a higher priority request is received by the TC, it completes or suspends its current operations at the crossbar priority of the new request. This ensures that no blockages occur in the system. For example, a low-priority packet transfer is suspended at a high-priority level when a high-priority transfer request is received.

2.3 Request Access Capabilities

The TC's access to on-chip memory areas is restricted by the type of service it is performing. Table 2–1 shows the types of accesses available to the TC for each type of request that it may receive. For example, the TC can access the MP instruction cache only to write the instructions fetched during an MP instruction cache fill. Note that in Figure 2–1 there are two types of accesses that the TC is not requested to do. One is direct access from the processor, and the other is data cache write-back to an on-chip location.

Table 2–1. TC Access Capabilities

		MP			PP			On-Chip Registers	Off-Chip Memory
		Parameter RAM	Instruction Cache	Data Cache	Parameter RAM	Instruction Cache	Data RAM		
MP	Instruction Cache Fill	Fault	Write	Fault	Read	Fault	Read	Fault	Read
	Data Cache Fill	N/A †	Fault	Write	N/A †	Fault	N/A †	N/A †	Read
	Data Cache Write-Back	N/A ‡	N/A ‡	Read	N/A ‡	N/A ‡	N/A ‡	N/A ‡	Write
	Packet Transfer	Read/Write	Fault	Fault	Read/Write	Fault	Read/Write	Fault	Read/ Write
	Direct External Access	N/A †	Fault	N/A †	N/A †	Fault	N/A †	N/A †	Read/ Write
PP	Instruction Cache Fill	Fault	Fault	Fault	Read	Write	Read	Fault	Read
	Packet Transfer	Fault	Fault	Fault	Read/Write	Fault	Read/Write	Fault	Read/ Write
	Direct External Access	Fault	Fault	Fault	N/A †	Fault	N/A †	Fault	Read/ Write
VC Serial Register Transfer		Fault	Fault	Fault	Fault	Fault	Fault	Fault	Legal
XPT VCPT		Read/Write	Fault	Fault	Read/Write	Fault	Read/Write	Fault	Read/ Write

† The processor accesses these locations directly.

‡ Only data from off-chip locations is loaded in the MP data cache. Write-back to on-chip locations is never requested.

2.4 Cache-Service Requests

The transfer controller automatically services cache misses for PP instructions and MP instructions, and data. When receiving multiple cache-service requests, the TC prioritizes them on a round-robin basis, as shown in Figure 2–1, and signals the requesting processor when its cache miss has been serviced. The MP can have both its instruction and data caches serviced within its turn on the round robin.

2.4.1 PP Cache-Miss Servicing

The PP instruction caches are one-way set-associative (fully associative) caches consisting of four blocks. Each block contains four 128-byte (16-instruction) subblocks. When a PP experiences a cache miss, its program-flow-control unit signals the TC, requesting a cache-miss service. The PP determines which cache block address the instructions should be placed in and passes this information to the TC. The TC fetches a complete subblock (128 bytes) from external memory and places it in the appropriate cache subblock of the requesting PP. The TC then informs the PP that the request has been serviced so that the PP can continue executing.

2.4.2 MP Cache-Miss Servicing

MP caches are four-way set-associative caches, with each set consisting of four blocks. Each block contains four 64-byte (16-instruction) subblocks. The MP can request service on its instruction cache or its data cache (or both). The MP instruction cache-service request is handled identically to the PP cache-service request, except that the size of the MP subblock fetched is only 64 bytes.

The MP's data cache is different from the instruction cache in that the TC may be requested to write the data contents back to external memory. For an MP data-cache miss, the TC fetches a 64-byte subblock, just as with the MP instruction cache. However, if the MP experiences a block miss (no matching tag address found) and all blocks are used, it will request that the TC write back any dirty subblocks in the least recently used block before the block is replaced. Dirty subblock writebacks can all take place in the same turn in the round-robin priority. The TC can also be requested to write back a dirty subblock in response to the MP's *dcachec* or *dcachef* command.

Note:

Access to on-chip RAM or memory-mapped registers is direct. On-chip locations are not cached in the MP data cache.

2.5 DEA Requests

The transfer controller is responsible for handling all direct external access (DEA) requests from the MP and PPs. DEA cycles allows the PPs to access data in external memory directly and allow the MP to bypass its data cache. DEA accesses can be a byte, halfword or a doubleword in length. They are given a higher priority but are limited to a single access. This prevents a single processor from monopolizing the external bus with multiple DEA cycles, which prevents other processors' DEA request and cache misses from being serviced. DEA cycles are intended to be used when fast accesses to a single off-chip memory location (such as a program variable or off-chip register) is needed.

Table 2–2 shows the type of accesses that are available to the MP and PPs for the various TMS320C80 address ranges. The subsections that follow discuss how the PPs and the MP handle the DEA requests.

Table 2–2. PP and MP Memory Access Capabilities

Memory Area	PPs	MP User Mode	MP Supervisor Mode
External memory (beyond 0x01FF FFFF)	DEA	DEA/cache	DEA/cache
On-chip registers	Fault	Direct	Direct
MP instruction cache	Fault	Fault	Fault
MP data cache	Fault	Crossbar (read only)	Crossbar (read/write)
MP parameter RAM	Fault	Crossbar (read only)	Crossbar (read/write)
PP instruction cache	Fault	Fault	Fault
PP parameter RAM	Crossbar	Crossbar	Crossbar
Data RAMs	Crossbar	Crossbar	Crossbar
Other	Fault	Fault	Fault

Note: DEA—Available through DEA request
Cache—Available through MP data cache
Crossbar—Direct access through crossbar
Fault—Illegal access; fault occurs
Direct—Direct access via processor's internal bus

2.5.1 PP DEA Requests

As Table 2–2 shows, the PPs can access their parameter RAMs and data RAMs normally. Accesses to addresses 0x02000000 and above automatically cause a DEA request to be sent to the TC. The request is serviced when the requesting PP's turn in the cache/DEA round robin is reached. If a PP tries to access an on-chip memory area not accessible via

the crossbar (such as the MP parameter RAM), the access converts to a DEA request to the TC. That DEA, however, results in a fault. PP DEA cycles that cause a fault to occur are handled in the same way that PP-cache-cycle faults are handled (see section 7.7.4.2, *PP Cache/DEA Faults*).

2.5.2 MP DEA Requests

The MP uses DEA cycles in a slightly different manner than the manner in which the PPs use them. The MP normally accesses external memory through its data cache. It uses DEA cycles to bypass its data cache and access memory directly. DEA cycles are explicitly specified using special versions of the *ld* and *st* instructions, *dld* and *dst*.

If the MP attempts an *ld* or *dld* instruction (or an *st* or *dst* instruction) to a non-accessible on-chip address (such as a PP instruction cache), the operation converts into a DEA request and is then faulted by the TC (see section 7.7.4.3, *MP Cache/DEA Faults*). A *dld* or *dst* operation to an accessible on-chip memory area (such as the on-chip registers of a PP data RAM) converts to a normal *ld* or *st* operation; no DEA request occurs.

Note:

There is no hardware mechanism to ensure coherency between DEA accesses and the MP data cache. Care should be taken when performing DEAs to memory locations that may be cached.

2.5.3 DEA Cycle Latency

Table 2–3 shows the latency of DEA cycles generated by an MP load (dld) or store (dst) instruction, or by a PP access to external memory:

Table 2–3. DEA Cycle Latency

		Number of Cycles	
Direction	**Memory Type**	**Page Hit**	**Page Miss**
Store	Any type of memory	8	12
Load	1 cycle per column, nonpipelined	11	15
	1 cycle per column, pipelined	12	16
	2 cycles per column	11	16
	3 cycles per column	12	18

The latency data given in Table 2–3 assumes 64-bit external memory. To access memory on a 32-bit, 16-bit, or 8-bit wide bus, the number of additional column cycles required to complete the transfer must be added to the given values. The table does not include the time that it takes for the cache-service round robin to reach the requesting processor. The requesting processor may have to wait for all other processors to have their cache-service and DEA requests performed and for other higher priority requests to be completed.

DEA requests do not necessarily stall the MP's pipeline, because the MP registers are scoreboarded. The MP will stall because of a DEA store only if another attempt to access data (from on-chip RAMs, memory-mapped registers, or from off-chip) is made. DEA loads will cause the MP to stall under the same conditions. The MP will also stall if there is an attempt to use the register(s) being loaded by the DEA. The PPs always stall until the DEA they requested has completed.

Back-to-back DEA requests to the same memory page do not always occur in page mode because DEAs are individual transfers, and other requests may occur between each transfer. (A page-mode cycle is performed whenever possible, however.) For PPs, the cache/DEA round-robin token always passes on to the next requesting processor after the completion of a DEA request. If more than two data transfers are involved, a packet transfer is usually a more efficient transfer method. The MP can issue back-to-back DEA requests without the round-robin token advancing. Using packet transfers, the MP is not stalled by a considerable amount of data, whereas multiple DEAs are.

Chapter 3

Packet Transfers

The primary method for transferring data on and off chip is the packet transfer. Packet transfers can be requested by the master processor (MP), parallel processor (PPs), video controller (VC), or external devices. This chapter discusses packet transfer formats and operation.

Topics in this chapter include:

Topic **Page**

3.1 Introduction to Packet Transfers

A *packet transfer* (PT) is a transfer of data blocks between two areas of TMS320C80 memory. Data is transferred by the transfer controller from a source (*src*) memory area to a destination (*dst*) memory area. The source and destination areas can be either on-chip or off-chip memory.

Packet transfers are initiated by the MP, PPs, VC, or external devices as requests to the TC. The TC services the requests using the fixed and round-robin prioritizations. discussed in section 2.1.6, *Arbitrating Requests of the Same Priority* and section 3.2, *Initializing a Packet Transfer*. Once a processor has submitted a request for a transfer, it can continue program execution. The packet transfer is completed by the TC without the need for additional processor cycles.

Packet transfers can be submitted on different priority levels, with higher priority transfers interrupting lower priority transfers. When this occurs, the lower priority transfer is suspended by the TC, and the current state of the transfer is saved. When the higher priority transfer is completed, the suspended transfer resumes automatically at the point at which it was interrupted.

3.1.1 Packet Transfer Terminology

To help you understand packet transfers, here is a list of packet transfer terms and brief definitions.

- *Line:* A number of contiguous bytes in memory.
- *Patch:* A group of lines of equal length whose starting addresses are an equal distance apart.
- *Packet:* A collection of patches.
- *Pitch:* The difference in addresses between the start of two lines or two patches.
- *Parameter table:* A group of parameters that describe a data packet and how it is to be moved from src to dst.
- *Linked list:* A collection of parameter tables, each pointing to the next table in the list.
- *Guide table:* A table of parameters describing individual patches within a packet transfer.
- *Src transfer:* The transfer of data from src memory.
- *Dst transfer:* The transfer of data to dst memory.

3.1.2 Packet Transfer Parameter RAM Usage

The MP and the PPs each have a number of parameter RAM locations set aside for the TC to use when servicing that processor's packet transfer requests. These areas are shown in Figure 3–1 and Figure 3–2.

Figure 3–1. MP Parameter RAM Contents

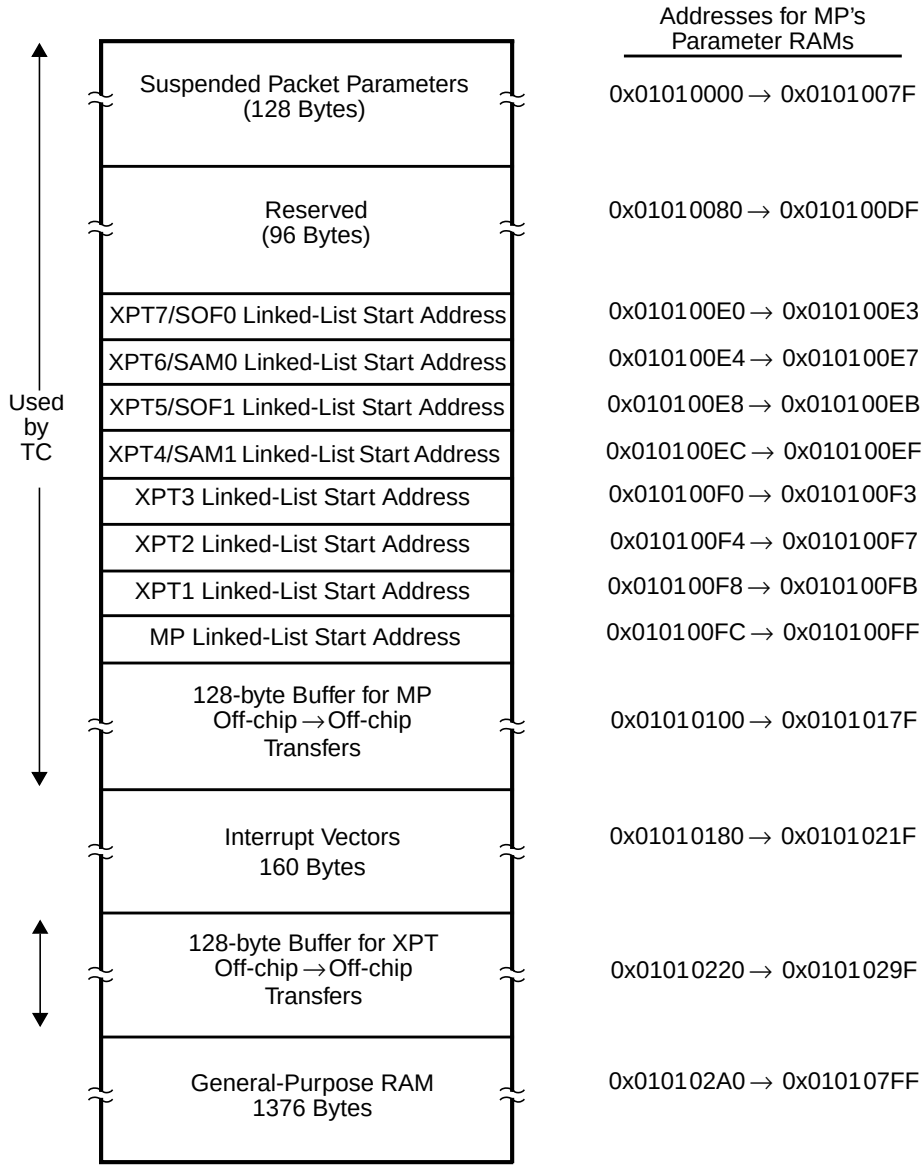
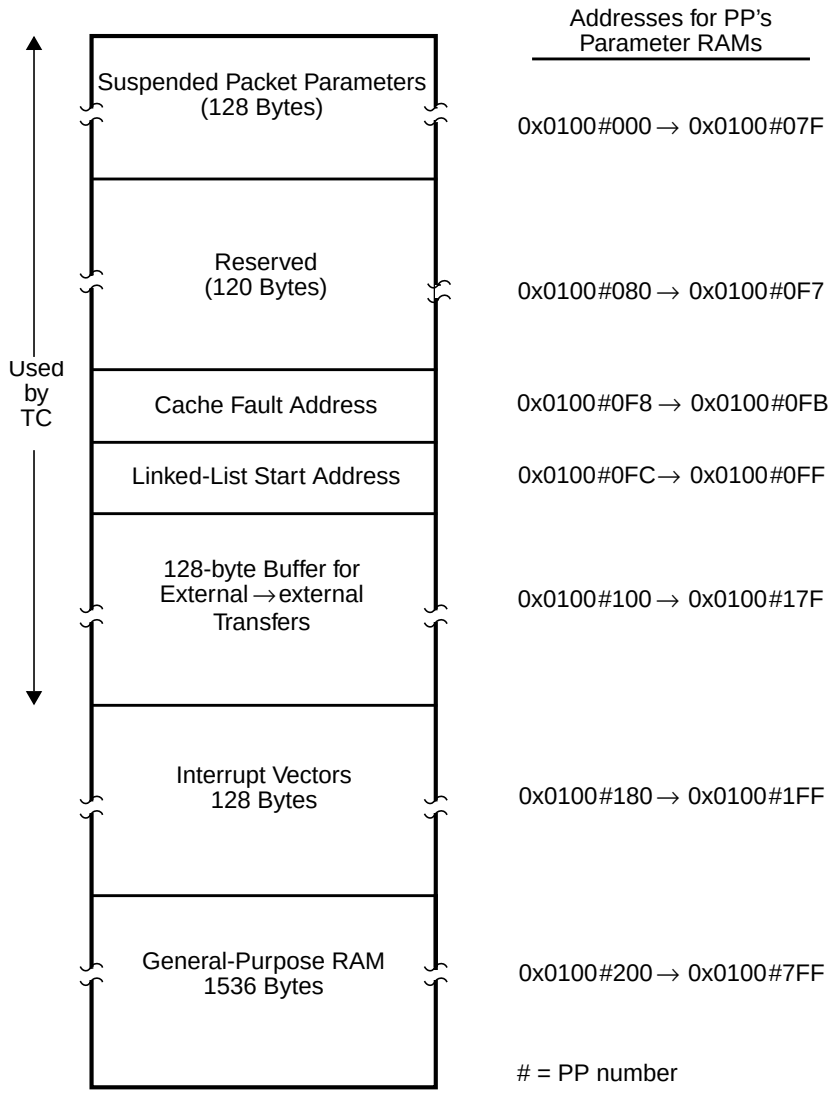


Figure 3–2. PP Parameter RAM Contents



Note:

Parameter RAM areas used by the TC are not restricted to TC use only. Data placed in these locations can be overwritten by the TC during a packet transfer operation. Do not write to these locations when a transfer request is active or has been suspended, because this may corrupt the transfer and/or its data.

3.2 Initializing a Packet Transfer

The general routine for a processor to initialize a packet transfer is as follows:

1) Create the packet transfer parameter table(s) in on-chip memory. For long-form packet transfers, the start of each table must be 64-byte aligned. Short-form packet transfers are also possible; in this case, the start of each table must be 16-byte aligned.

2) If necessary, generate the packet transfer's guide table(s) in on-chip memory. See section 3.6.1, *Guide Tables.*

3) Set the linked list start address in parameter RAM to point to the beginning of the (first) parameter table.

4) Verify that the Q bit is not set; thus verifying if another packet transfer is active.

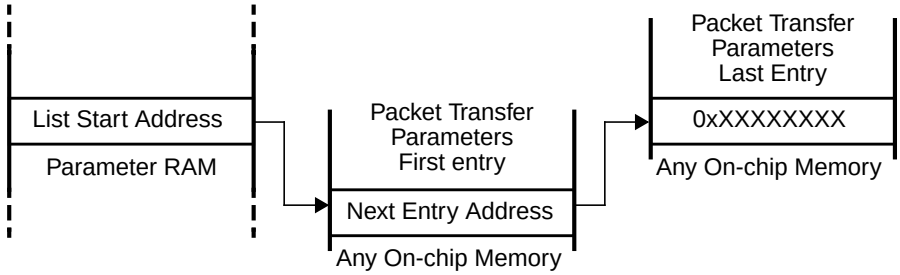
5) Set the appropriate packet transfer priority bits and the P bit to submit the request to the TC. (These bits are located in the PKTREQ control register for the MP, and in the comm register of the PPs.) Detailed descriptions of these processes are included in Chapter 7, *Understanding the Packet Transfer Request Protocol,* in the *Master Processor User's Guide* and Chapter 12, *Packet Transfers,* in the *Parallel Processor User's Guide*.

3.2.1 The Linked-List Structure

Packet-transfer requests are submitted as linked-list structures, one per processor. A *linked list* is a collection of packet transfer parameter tables with each packet transfer entry pointing to the next entry on the list. Although packet transfers can operate in on-chip and/or off-chip memory, the linked lists of parameter tables themselves are restricted to on-chip memory only. Each processor can have any number of linked lists stored in memory, but only one of them can be active at a time. The start of the (active) linked list is given in the dedicated linked-list start-address location in the requesting processor's parameter RAM. Each entry (parameter table) in the linked list contains a pointer to the next entry location on the list. The final entry in the list can point anywhere, because the end of the list is marked by the S (stop) bit in the packet transfer parameter table's PT options field. However, the linked-list start address is always updated with the pointer from the final entry in the linked list. (See section 3.10, *Linked-List Management*.)

A simple linked-list structure is shown in Figure 3–3. This list contains two packet transfers, but a linked list can be any length, as long as it can fit in on-chip memory.

Figure 3–3. Linked List Structure



3.2.2 Packet Transfer Service

Once a processor submits a packet transfer by setting its P bit, its Q bit is then set, indicating that the processor's linked list has been queued within the TC. When the round-robin token appropriate to the packet transfer priority level reaches the submitting processor, the TC begins to actively service the request as follows:

1) The TC reads the starting location of the linked list from the linked-list start-address location in the processor's parameter RAM.

2) The TC reads the first packet-transfer parameter table into registers within its source and destination controllers.

3) The TC uses its source and destination controllers to transfer the data as indicated within the parameter table.

4) When the packet transfer is complete, the TC updates the linked list start address in parameter RAM with the next address in the linked list (the packet transfer's Next Entry Address).

5) The TC repeats the procedure until it completes the last entry in the linked list.

Note:

There is no hardware mechanism to ensure coherency between the MP's data-cache and packet transfers. Use care when performing packet transfers on memory areas that might be cached.

3.3 Normal Packet Transfers

Packet transfers provide a number of different formats and options to allow flexibility in the movement of data. When considering these formats, it is important to remember that the packet transfer's source transfer and destination transfer are independent of one another. This allows packet data to be read using one format, and written using an entirely different format, enabling a number of spreading or merging functions to be achieved.

The two basic types of transfer formats available to normal packet transfers are:

- ❏ Dimensioned transfers, described in section 3.5, *Dimensioned Transfers*.
- ❏ Guided transfers, described in section 3.6, *Guided Transfers*.

These formats determine how data is read or written, depending on whether they describe a src transfer or a dst transfer. It is acceptable to specify a different format for each transfer.

Although the flexibility provided by dimensioned and guided transfers is extremely powerful, many applications require the use of only simple, single-dimensional transfers. In order to minimize the amount of on-chip memory required for the parameter tables of these simple transfers, you may make use of the short-form packet transfer.

Short-form packet transfers are a restricted form of the dimensioned transfer. They allow the transfer of a contiguous number of bytes (up to 65 535) from source memory to destination memory. Because they are only one dimensional, short-form PTs may be specified using a fewer number of parameters. This saves on-chip memory by reducing the size of the parameter table for the packet transfer from 16 words to 4 words. Programming overhead may also be reduced because there are no table entries that require clearing to disable unused transfer features.

Other than the different parameter-table format and the simplified transfer modes, the short-form packet transfer operation is identical to that of normal packet transfers: short-form transfer length is controlled by PTMIN and PTMAX, short-form transfers follow the same linked-list management rules and use the same suspend mechanism as normal packet transfers. A normal packet transfer may point to a short-form packet transfer in the same linked list. A short-form packet transfer may also point to a normal packet transfer in the same linked list provided the start address for the normal packet transfer is 64-byte aligned. Short form PT parameter tables need only be 16-byte aligned.

3.4 Normal Packet-Transfer Parameters

The variety of transfer formats results in a large number of possible src and dst transfer combinations. Each combination of src and dst transfer requires a specific parameter format, which resides in the packet transfer parameter table. This section describes normal (long-form) parameter tables. For a discussion of short-form parameter tables, see section 16.3, *Short-Form Packet-Transfer Parameters* and section 3.7.1, *Short-Form PT Parameter Table*. A complete listing of the src and dst operating mode combinations and their associated parameter-table formats is included in Appendix A.

Packet transfer parameter tables must be located in on-chip memory (MP or PP parameter RAMs or PP data RAMs) and must be 64-byte aligned (six LSBs of the address are 0). The table can reside in any processor's RAM; thus, the MP can use a parameter table located in PP's parameter RAM, or vice versa. The requesting processor simply places the appropriate starting address in the linked-list start-address location in its own parameter RAM before submitting its transfer request.

For further information, see section 16.2, *Normal Packet-Transfer Parameters*.

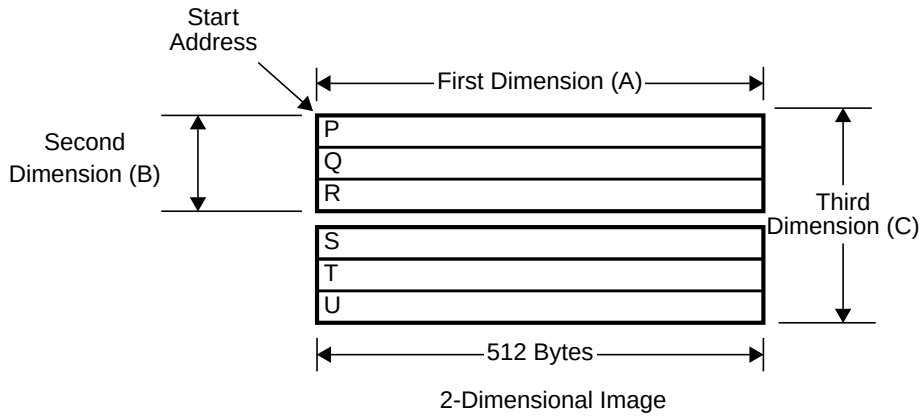
3.5 Dimensioned Transfers

Dimensioned transfers are the simplest type of transfer, but are also the most rigid. Dimensioned transfers describe sources or destinations that can be a simple contiguous linear sequence of data bytes or that can consist of a number of such regions. The addressing mechanism allows an array of up to three dimensions to be specified. This allows several two-dimensional patches to be transferred by a single packet transfer.

Data along the first (*A*) dimension is always one byte apart. The spacing along the second (*B*) and third (*C*) dimensions is arbitrary, but fixed for the entire packet.

Figure 3–4 demonstrates how a dimensioned transfer might access the src or the dst memory. It shows a packet consisting of two patches of three lines, each consisting of 512 adjacent 8-bit pixels. This type of transfer might be needed, for example, if two PPs were going to perform a 3 x 3 convolution, with each working on one of the patches of lines. The first patch (PQR) might represent data to be transferred into PP0's data RAM, and the second patch (STU) might represent data to be transferred into PP1's data RAM. This example represents a src transfer because it specifies how the TC reads data from the source memory area.

Figure 3–4. Dimensioned Packet Transfer



The data packet is specified in terms of the following parameters:

- ❏ A count: The number of contiguous bytes in a line (first dimension).
- ❏ B count: The number of steps required to form a patch. (second dimension = number of lines minus one).
- ❏ C count: The number of patch steps required to form a packet. (third dimension = number of patches minus one).
- ❏ Start address: The linear address of the start of the packet
- ❏ B pitch: The linear pitch of the second dimension.
- ❏ C pitch: The linear pitch of the third dimension.

The example shown in Figure 3–4 would therefore have the following parameters:

A count	—	512
B count	—	2
C count	—	1
Start Address	—	Address of byte *P*
B pitch	—	Difference of addresses of bytes *P* and *Q*
C pitch	—	Difference of addresses of bytes *P* and *S*

Both src and dst transfers can be defined in the above manner, but the parameters are independent so that the shape of the src and dst can be quite different.

Not all dimensions of a dimensioned transfer are required to be active. By setting the B count and/or the C count to 0, you can limit transfers to individual bytes, pixels (multiple bytes), lines, or patches.

Note:

An A count value of 0 results in no data being transferred and can cause an error to occur. For more details, see section 3.11, *Packet-Transfer Errors*.

3.6 Guided Transfers

Guided transfers are those in which the sequence of dimension addresses is guided from an on-chip memory table, rather than calculated solely from values within the packet transfer parameters. This operation is more complicated than that of dimensioned transfers, but it is also much more flexible. There are two classes of guided transfers:

- ❏ Fixed patch
- ❏ Variable patch

Fixed-patch guided transfers have the first and second dimension described within the packet transfer parameters as with dimensioned transfers, but the third dimension is guided from entries in an on-chip table. In *variable-patch guided transfers,* the guide table also determines the sizes of the A and B dimensions for each patch.

In both classes of guided transfers, the first two dimensions are active, similar to that of dimensioned transfers. Each guided table entry can move an individual byte, an individual pixel (multiple bytes), a line, or a two-dimensional patch according to the sizes of the first two dimensions. This allows a number of irregular operations, such as those encountered in line draws or processing data by using lookup tables.

Unless otherwise noted, the information in the following descriptions of guided transfers applies equally to both src and dst transfers.

3.6.1 Guide Tables

A guide table is simply a block of either 32-bit or 64-bit entries, depending on the transfer type. The table must be located in on-chip memory. For fixed-patch transfers, the table must be aligned to a word (32-bit) address. For

variable-patch transfers, the guide table must be doubleword (64-bit) aligned. The starting address of the guide table and the number of entries it contains are located within the packet transfer parameter table. Each guide table entry corresponds to a two-dimensional patch within the transfer. As the TC services the packet transfer, it fetches the guide table entries one at a time, as needed to process the next patch in the transfer. Additional information about guide table entries is found in each of the guided transfer descriptions found in the following sections.

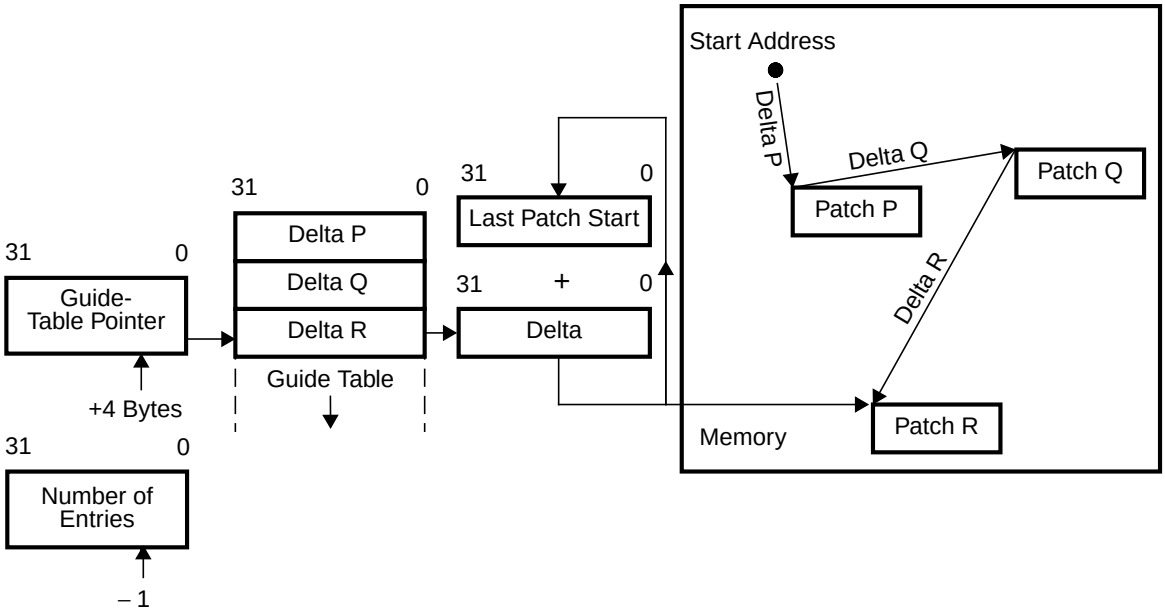
3.6.2 Fixed-Patch Guided Transfers

Fixed-patch guided transfers use an on-chip guide table containing 32-bit entries. The table must be word-aligned (two least significant bytes (LSBs) of the table's entry address must be 0s). Each entry contains information used to calculate the addresses for the third dimension of the transfer. Fixed-patch transfers come in three types:

- ❏ Delta-guided
- ❏ Offset-guided
- ❏ Offset-guided look-up table (LUT)

Fixed-patch delta-guided transfers. For fixed-patch delta-guided transfers, the guide table contains 32-bit delta values to be added to the starting address of the previous two-dimensional patch to form the starting address of the current patch. The patch size is fixed and defined by the A count and B count packet transfer parameters.

Figure 3–5. Fixed-Patch Delta-Guided Transfer Address Mechanism



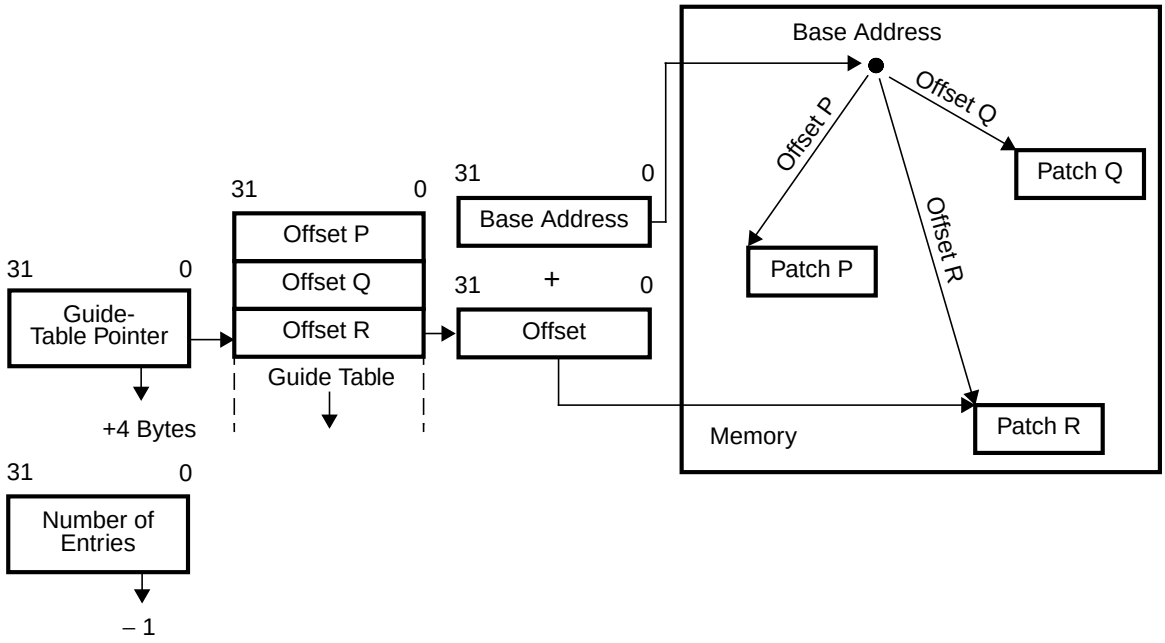
An example of a fixed-patch delta-guided transfer is shown in Figure 3–5. Here the value delta P is added to the starting address (given in the packet transfer parameters) to form the start address of the first patch, P. Delta Q is then added to the start address of patch P to form the start address of patch Q, and so on.

As Figure 3–5 shows, an internal table pointer is incremented by four bytes after each patch and points to the next guide table entry. An entry counter is loaded with the number of entries value taken from the packet transfer parameter table. This counter is then decremented by 1 after each patch, and the transfer is terminated when it reaches 0.

Fixed-patch offset-guided transfers. Fixed-patch offset-guided transfers use a guide table that contains 32-bit values to be added to a base address given in the packet-transfer parameters to form the starting address of each patch. The patch size is fixed and defined by the A count and B count packet transfer parameters.

Figure 3–6 shows the addressing mechanism for fixed-patch offset-guided transfers. The value from the first entry in the guide table, offset P, is added to the base address (specified in the packet transfer parameter table) to form the start address of patch P. Offset Q is then added to the base address to form patch Q's start address, and so on. If the base address specified in the packet transfer parameter table is 0, the guide table specifies absolute addresses.

Figure 3–6. Fixed-Patch Offset-Guided Transfer Address Mechanism



As in the delta-guided transfer, a table pointer points to the current guide table entry, and a counter tracks the number of patches performed.

Fixed-patch offset-guided LUT transfers. For fixed-patch offset-guided *look-up table (LUT)* transfers, the guide table contains 32-bit offset values that are to be left-shifted (zero-filled) by zero, one, two, or three bits and then added to a base address given in the packet transfer parameters. This allows the transfer to be used for LUT operations, independent of the data size.

Note:

You can use the fixed-patch offset-guided LUT format only for src transfers.

The shift amount is indicated by the position of the leftmost 1 in bits 0-3 of the packet transfer parameter's A count field. A value of 1 in bit 3 indicates a left shift three places; in bit 2, two places; in bit 1, one place; and in bit 0, zero places. The left shift allows 8-, 16-, 32-, and 64-bit data-sized LUTs to be supported.

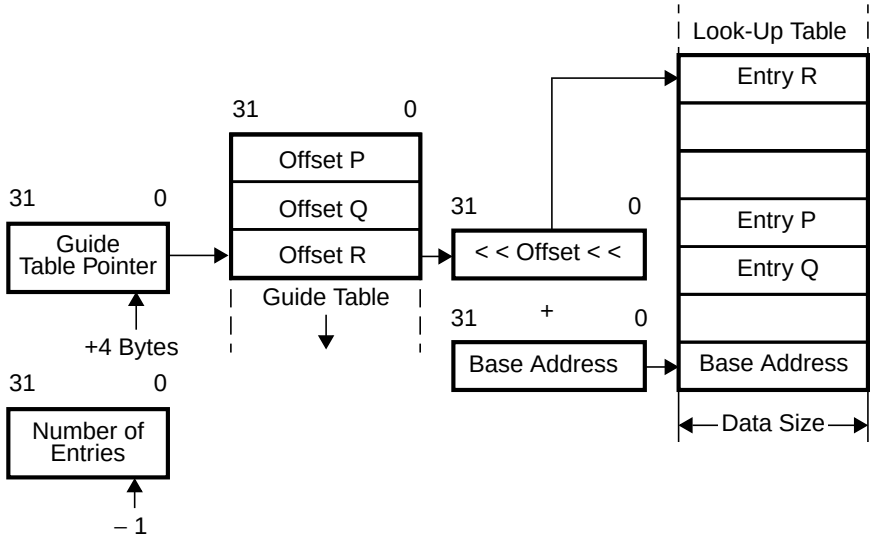
Table 3–1. Shift Calculation for LUT Transfers

A Count Bits					Left Shift
15–4	**3**	**2**	**1**	**0**	
0x000	0	0	0	1	0
0x000	0	0	1	0	1
0x000	0	1	0	0	2
0x000	1	0	0	0	3

The patch size for LUT transfers is normally one dimensional, and set to one, two, four, or eight bytes. It must be programmed to have an A count of one, two, four, or eight bytes. No other A count values are supported. A B count value of 0 is normally used but is not required.

An example of a fixed-patch offset-guided LUT transfer is shown in Figure 3–7. Offset P is shifted left by zero, one, two, or three bits according to the LUT data size (indicated by the value in the A count field). This value is then added to the base address to form the starting address of patch P. The offset Q value is then shifted and added to the base address to form the patch Q starting address, and so on. The shifting occurs as the offset value is being loaded from the guide table.

Figure 3–7. Fixed-Patch Offset-Guided LUT Transfer Address Mechanism



As before, the table pointer and entry registers keep track of the current position in the guide table and the number of patches.

3.6.3 Variable-Patch Guided Transfers

Variable-patch guided transfers specify all patch-size information within the guide table rather than in the packet-transfer parameters. This allows each patch within the transfer to have different dimensions. Variable-patch guided transfers can be either of the following:

- ❏ Delta-guided
- ❏ Offset-guided

For variable-patch guided transfers, the guide table consists of 64-bit doubleword entries. The little-endian format for the guide table is shown in Figure 3–8. The lower half of the doubleword contains the A and B count values for the first two dimensions. The upper 32 bits contain the value used to calculate the address of the third dimension. Figure 3–9 shows the big-endian guide table format.

Figure 3–8. Variable-Patch Guide Table Format (Little Endian)

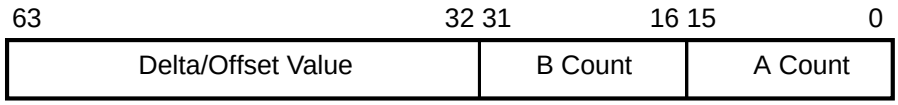
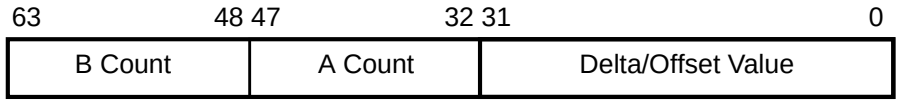
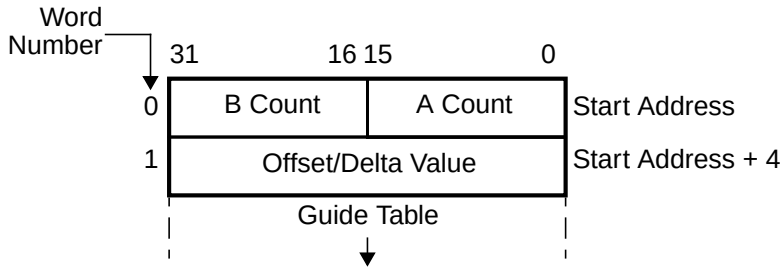


Figure 3–9. Variable-Patch Guide Table Format (Big Endian)



Software can be written to create guide tables independent of the endian format by using 32-bit writes. As Figure 3–10 shows, the A and B counts always appear at word 0 addresses and the offset/delta values appear at word 1 addresses (where word 1 is at the address four bytes greater than word 0). The TC always accesses both 32-bit words at the same time and adjusts the word order as necessary for correct internal operation.

Figure 3–10. Variable-Patch Guide Table Format (Endian-Independent)

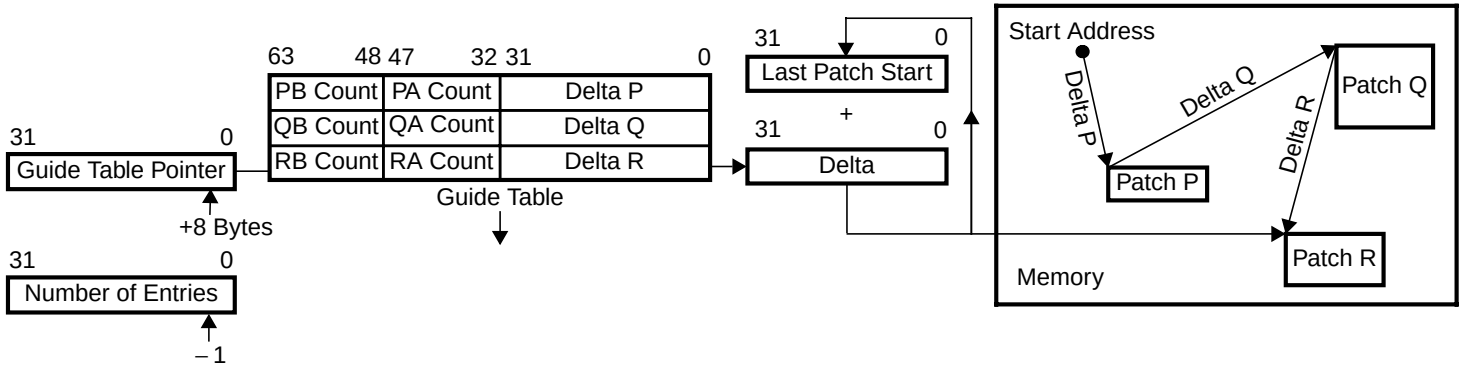


Guide-table entries for variable-patch transfers must be doubleword aligned (the three LSBs of the address must be 0).

Variable-patch delta-guided transfers. Word 1 of a guide-table entry contains the 32-bit delta amount to be added to the start address of the previous patch. The starting address is given in the packet-transfer parameters, and the patch size is variable and specified in word 0 of each guide table entry.

Figure 3–11 shows an example of a variable-patch delta-guided transfer. Here, delta P is added to the starting address (specified in the packet transfer parameters) to form the starting address of patch P. The PA count determines the length (number of bytes) of the first dimension, and the PB count is the size of the second dimension (number of lines -1). Delta Q is then added to the patch P starting address to generate the patch Q starting address. As with fixed-patch transfers, a table pointer and entry counter keep track of the position in the guide table.

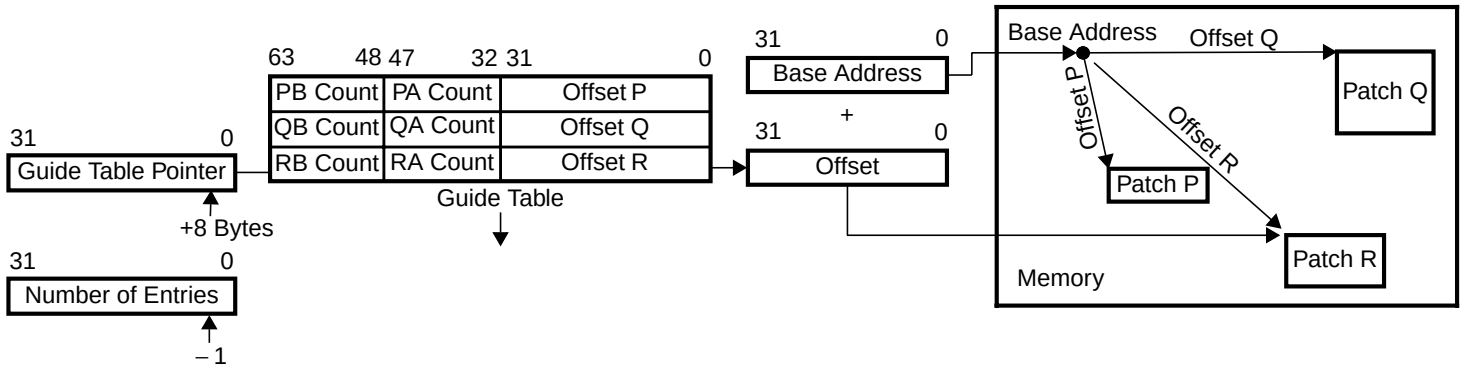
Figure 3–11. Variable-Patch Delta-Guided Transfer Address Mechanism (Big Endian)



Variable-patch offset-guided transfers. Word 1 of guide table entries for variable-patch offset-guided transfers contains a 32-bit offset value that is added to the base address to calculate the start address of each patch. The base address is specified in the packet-transfer parameters. The patch size is variable and specified in the least significant half (word 0) of each guide table entry.

Figure 3–12 is an illustration of a variable-patch offset-guided transfer. The offset P value is added to the base address to generate the starting address of patch P. The PA count and PB count values determine the size of the patch. Offset Q is then added to the original base address to get the starting address for patch Q, and so on.

Figure 3–12. Variable-Patch Offset-Guided Transfer Address Mechanism (Big Endian)



3.6.4 Fill-With-Value Transfers

The *fill-with-value transfer*, like the LUT transfer, can only be specified for src transfers. Fill-with-value transfers do not actually transfer data from src memory, but instead specify the source value within the packet transfer parameters. Two 32-bit fields (most significant fill-value word and least significant fill value word) specify the 64-bit value that is used to fill the dst memory. If the fill pattern is less than 64-bits, then the pattern must be replicated throughout the least significant and most significant fill value words. No alignment operations are performed on the fill value. The bytes written to the destination doubleword are the corresponding bytes from the fill value doubleword.

Figure 3–13 shows an illustration of a fill-with-value transfer with a dimensioned destination. In this example, the fill pattern is only 32 bits long and replicated within the fill-value doubleword. The destination start address is 0x02000002, which corresponds to the third byte within the doubleword. A destination A count of 11 (bytes) and B count of 1 results in the destination being filled as shown. Figure 3–14 shows a fill transfer using the same parameters in big-endian mode.

Figure 3–13. Fill Transfer (Little Endian)

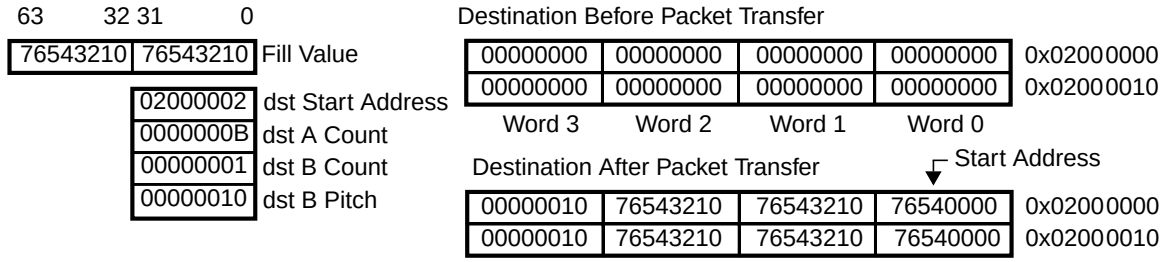
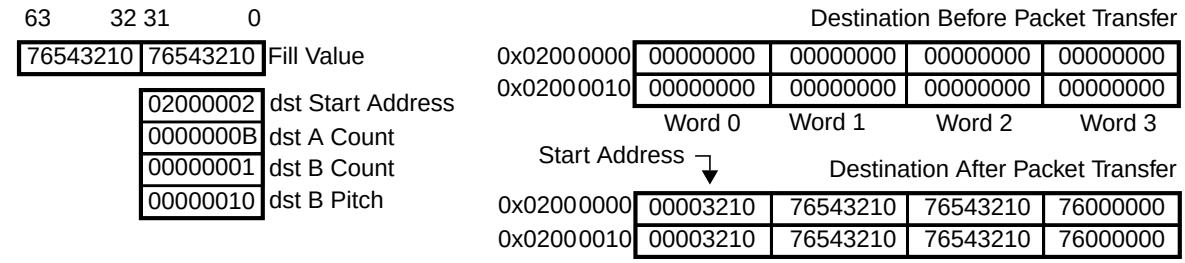


Figure 3–14. Fill Transfer (Big Endian)



No src address or src dimension counts are specified for fill transfers. The size of the packet transfer is determined by the dst transfer parameters.

3.7 Short-Form Packet Transfers

Short-form packet transfers are a restricted form of the dimensioned transfer. They allow the transfer of a contiguous number of bytes (up to 65 535) from source memory to destination memory. Because they are only one-dimensional, short form PTs may be specified using a fewer number of parameters. This saves on-chip memory by reducing the size of the packet transfer parameter table from 16 words to 4 words. Programming overhead may also be reduced because there are no unused table entries that need to be cleared in order to disable unused transfer features.

3.7.1 Short-Form PT Parameter Table

Short-form PT parameter tables must reside in on-chip memory and must be 16-byte aligned (the four LSBs of the address are 0). This table can reside in any processor's RAM; thus, the MP can use a parameter table located in PP's parameter RAM, or vice versa. Figure 3–15 shows the short-form parameter-table format for big- and little-endian modes. The value PT represents the 16-byte aligned starting address of the parameter table. The parameter table may be programmed independently of the endian mode by using 32 bit writes to the proper word address as shown in Figure 3–16.

For information about short-form parameter fields, see Chapter 16, *Programming Details*.

Figure 3–15. Short-Form Parameter Table

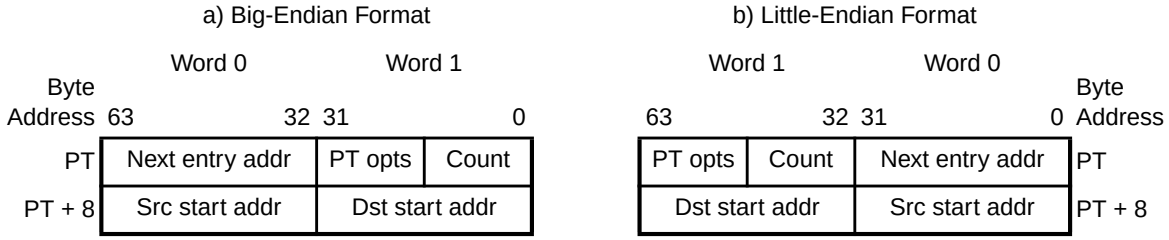
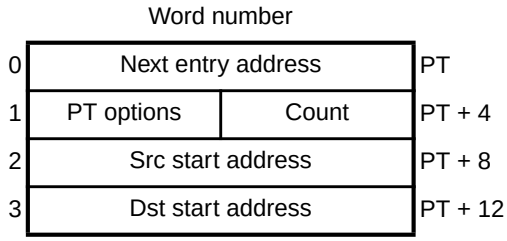


Figure 3–16. Short-Form Parameter Table (Endian Independent)



3.8 Packet Data Transfer

For details about programming the packet-transfer parameters into a parameter table, see section 16.2, *Normal Packet-Transfer Parameters*.

When the TC has read the packet-transfer parameters from the parameter table, it is ready to begin transferring data. To do this, the TC generates either crossbar or external memory accesses (or both). There are four basic src-to-dst data flow possibilities:

- ❏ On-chip to on-chip
- ❏ On-chip to off-chip
- ❏ Off-chip to on-chip
- ❏ Off-chip to off-chip

The first three transfers are considered normal, while the latter is a special case. During the course of a packet transfer, the src and dst addresses can change from on-chip to off-chip and vice-versa, without restrictions. This means that the data flow can change during a packet transfer.

3.8.1 Normal Transfer Process

During the normal packet data-transfer flow, the TC's src controller generates src addresses based on the transfer parameters and uses these to fetch data from the appropriate on-chip or off-chip memory. Once the crossbar or external memory bus receives the data, the required bytes are extracted, aligned, and placed in the TC's packet transfer FIFO. At the same time, the dst controller is also generating addresses for the dst memory. When the packet-transfer FIFO contains the required number of bytes for the next dst memory access, the dst controller generates the required crossbar or external-memory cycle.

The TC's packet transfer FIFO controls the data flow and keeps the src and dst controller in sync. If at any time the FIFO does not contain enough src bytes for the next dst access, the dst controller waits until the data becomes available. In the same manner, if the packet transfer FIFO is full, the src controller stalls until the dst controller processes enough bytes for the next src access to complete. This prevents the src transfer from overrunning the dst transfer. This data flow can be modified for special packet-transfer access modes. See Chapter 11, *Special Packet-Transfer Access Modes*.

Because the TC's crossbar and external memory interfaces are independent, src transfers can occur on the external bus, and dst transfers can occur on

the crossbar (or vice versa) in parallel. For on-chip to on-chip transfers, the src and dst must share the crossbar interface and interleave cycles.

Cache service requests, DEA requests, VC SRT requests, urgent refreshes, and host requests do not cause packet transfer suspension, but may stall one or both of the src or dst controllers, but only if a request has a higher priority than the transfer. SRT and urgent refresh cycles use only the external memory interface. Packet transfer crossbar accesses can continue to occur. If one of the controllers is using the external memory interface, the packet transfer FIFO eventually becomes either full or empty, and the controller using the crossbar has to stall until the external memory interface becomes available again. If the transfer is on-chip to on-chip then both src and dst controllers can continue unimpeded. Cache and DEA servicing use both crossbar and external interfaces and they always stall any packet transfer for the number of cycles that they require.

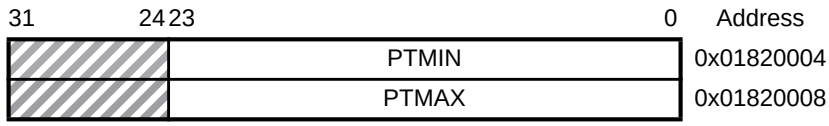
3.8.2 Off-Chip to Off-Chip Packet Transfers

Off-chip src to off-chip dst packet transfers are handled differently than the other three types of transfers. To take advantage of page mode on DRAMs and VRAMs, the TC performs a page-mode burst of column accesses from the off-chip source to on-chip and then another page-mode burst from on-chip to the off-chip destination. This requires using an on-chip buffer. Each processor has a 128-byte area in its parameter RAM reserved for this purpose (see Figure 3–2). The transfer of data in and out of the buffer is handled by the TC hardware and is transparent to the user.

3.9 Controlling the Length of Packet Transfers

Packet transfers that are very large can create problems because they take a long time to complete. Another problem is higher priority packet-transfer requests continually interrupting a packet transfer, and preventing it from completing. Controlling the size of transfers prevents either of these situations from occurring. To do this, the TC has two 24-bit registers, PTMIN and PTMAX, that specify the minimum and maximum length of a packet transfer. These registers are shown in Figure 3–17. An internal register, PTCOUNT, tracks the number of clock cycles that a transfer has been executing.

Figure 3–17. Packet-Transfer Length Registers



3.9.1 Setting the Minimum Packet Transfer Length

The PTMIN register indicates the minimum number of clock cycles a packet transfer must execute before it can be interrupted by a *higher priority* transfer request. After the parameters have been loaded and a packet transfer begins executing, this value is loaded into the PTCOUNT counter. PTCOUNT is decremented once for each clock cycle during which the packet transfer is actively serviced by the TC. PTCOUNT does not decrement during cache service and DEA service. It does decrement during retries and wait states that occur during active transfer service.

The packet transfer cannot be interrupted by a higher priority transfer request until PTCOUNT reaches 0. It can be suspended only by the requesting processor, or by an error or fault condition.

One important use of PTMIN is being able to create an unstoppable packet transfer that moves the parameters of a suspended packet transfer to another memory area. Because another suspension will overwrite the parameter RAM from which data is being transferred, PTMIN ensures that the transfer is completed. It is important that the transfer not fault because a fault would also cause the parameter RAM to be overwritten.

Note:

PTMIN is loaded with 0x10000 (65 536 cycles) at reset.

3.9.2 Setting the Maximum Packet-Transfer Length

The PTMAX register prevents any single packet transfer from monopolizing the transfer of data. Once the PTMIN time has elapsed (that is, PTCOUNT is decremented to 0), PTMAX is loaded into the PTCOUNT counter to define the remaining period of time for which the transfer can proceed before it is timed out. The maximum duration for an uninterrupted transfer is PTMIN + PTMAX clock cycles.

The PTCOUNT register is decremented for every cycle that the packet transfer continues to be active (excluding nontransfer cycles, such as cache or DEA). If the counter reaches 0 before the packet transfer completes, the transfer is considered to have timed out. The packet transfer is suspended, and the TC moves to the next request of the same priority in the round robin. If no other packet transfer request of the same priority is pending, the suspended transfer is reloaded, and PTCOUNT is again loaded with PTMIN in the normal manner. If a higher priority packet transfer request is received, the active transfer is suspended (assuming PTMIN has been satisfied), regardless of whether PTMAX is reached.

Notes:

PTMAX is loaded with 0x10000 (65 536 cycles) at reset.

3.9.3 PTMIN and PTMAX Application

The PTMIN and PTMAX values are applied whenever a packet transfer is loaded. They are also applied to each packet transfer within a linked list. If a packet transfer is suspended, the entire linked list is suspended. In other words, the round robin token advances to the next processor with a pending request, not to the next packet transfer within the linked list. When a packet transfer times out, the state of the suspended transfer is saved to the requesting processor's parameter RAM. It is automatically resubmitted for continuation when the round-robin priority is returned to the processor. Whenever a suspended packet transfer resumes, PTCOUNT is reloaded with PTMIN.

3.10 Linked-List Management

Packet-transfer linked lists are managed by the TC using the following guidelines:

- If a refresh, SRT, host interface, cache, or DEA service request is received during a packet transfer, the current state of the packet transfer parameters is retained in the TC's internal registers, and if required, the src and/or dst transfer is stalled. When the other requests have been serviced, the transfer is resumed. There are two exceptions however, and they are:

- An XPT or VCPT is allowed to complete before a cache/DEA request is serviced.
- A low-priority MP urgent packet transfer is allowed to complete before a low-priority MP or PP cache/DEA request is serviced.

❏ If a packet transfer is interrupted by a higher priority packet-transfer request, a timeout, a suspend request from the requesting processor, a fault, or an error, then the packet transfer is suspended and the linked list start address in the parameter RAM is changed to point to the saved packet-transfer parameters.

❏ If a packet transfer is interrupted by a higher priority packet transfer request, then the round-robin priority remains with the interrupted transfer so that its service resumes when the lower priority level is resumed. If the higher priority request occurs while the packet-transfer parameters are being loaded, the load is abandoned. No suspension occurs. The packet transfer is loaded from the original parameter list when the current higher priority packet transfer has completed.

❏ If a packet transfer is suspended because of a timeout, fault, error, or a suspension request, then the round-robin token is advanced, placing the suspended packet transfer at the end of the priority chain.

❏ If the interrupt bit in the PT options field is set, the requesting processor is sent an end-of-packet interrupt (PC bit in the MP or the PTEND bit in the PP) when the packet transfer has been completed. XPTs and VCPTs cannot interrupt a processor upon their completion.

❏ When the last packet transfer (the one with the stop bit set) in a linked list is completed, the requesting processor is sent an end-of-packet interrupt (PC bit in the MP or the PTEND bit in the PP.) For more information, see the *TMS320C80 Master Processor User's Guide* and the *TMS320C80 Parallel Processor User's Guide.*

❏ When a packet transfer has been completed, the *next address* field from the packet-transfer parameters is written to the linked-list start-address location in the requesting processor's parameter RAM (even if the packet transfer's stop bit is set).

❏ When the packet transfer parameters are loaded, if the PTS bits in the PT options field indicates the packet transfer was suspended, then the additional state information of the suspended transfer is loaded.

3.11 Packet-Transfer Errors

If a packet transfer experiences an error condition the TC immediately stops transferring data and performs a suspension (XPTs and VCPTs are excluded). The requesting processor's PT error flag (the BP bit in the MP, or the PTERR bit in the PP) is set and its linked list is terminated (the Q bit is cleared). The requesting processor may then be able to determine the cause of the error from the saved parameters. Any of the following conditions causes an error:

- ❏ Attempting a VRAM access mode transfer (block-write dst transfer, SRT src or dst transfer) to on-chip memory (see section 16.2.2, *PT Options*, and *PT Access Mode (PAM) - Bits 18–16*.
- ❏ Attempting a VRAM access mode (block write or SRT) by an XPT or a VCPT
- ❏ Attempting an on-chip dst access using transparency (see section 11.6, *Transparency-Packet Transfers*)
- ❏ The length (total number of bytes) of a packet's dst transfer exceeding the length of its src transfer. The exceptions are a fill-with-value, shift register transfer, or peripheral device transfer.
- ❏ Attempting an LUT or fill dst operation. For example, as a result of a src and dst parameters exchange operation.
- ❏ Attempting a PDPT read in which the src is not off-chip, or src start/base address (guided) is not off-chip.
- ❏ Attempting a PDPT write in which the dst address or dst start/base address (guided) is not off-chip.
- ❏ The linked-list start address or the next entry of the linked list pointing off-chip
- ❏ The linked-list start address in the packet transfer parameter table not aligning to a 64-byte boundary
- ❏ The guide table being off-chip or not properly aligned

The packet-transfer parameters are not saved to the suspend area of the parameter RAM when the last three conditions listed above occur, because the transfer itself never actually begins. When attempting to find the cause of an error, check the validity of the linked-list address to ensure that the

suspended parameters are valid. No status bits indicate the actual error condition; this must be deduced from the status of the suspend parameters.

3.11.1 Short-Form Packet Transfer Errors

The following error conditions exist for short-form PTs:

- ❏ A PT Access Mode (PAM) other than 000 or 001 was specified in the PT Options field of a short form PT.
- ❏ A short-form PT parameter table was not aligned to a 16-byte boundary.

The second error does not cause any data to be written to the suspend area because the transfer never actually begins.

3.12 Packet-Transfer Suspension

Any of the following cause a packet transfer to be suspended:

- ❏ If the TC receives a higher priority packet transfer request and PTMIN for the current packet transfer has expired. The Q bit for the current packet transfer remains set.
- ❏ If the packet transfer exceeds PTMIN plus PTMAX, and times out. The Q bit for the current packet transfer remains set.
- ❏ If the processor that submitted a packet transfer requests that it be suspended.
- ❏ If a memory fault occurs during src addressing.
- ❏ If a memory fault occurs during dst addressing.
- ❏ If an error condition is detected. (See section 3.11, *Packet-Transfer Errors*.)
- ❏ If a retry occurs during an external memory access, a packet-transfer request of equal priority from another processor is waiting, and PTMIN for the current transfer has expired. The Q bit for the current packet transfer remains set.

The suspension mechanism in each of these cases is identical, and involves saving the current packet-transfer parameters and the internal state of the TC. This information allows the packet transfer to be continued in the future.

The parameters are saved to the suspension area of the requesting processor's parameter RAM, shown in Figure 3–2. The format for these parameters is shown in Figure 3–18 and Figure 3–19. doublewords marked by per cent are not adjusted for the current endian mode.

Figure 3–18. Suspended Packet Transfer Parameters (Little Endian)

Word 1		Word 0		MP Addresses	PP Addresses
63	32	31	0		
PT options		Original PT entry address		0x01010000	0x0100#000
Current dst address		Current src address		0x01010008	0x0100#008
dst B count	dst A count	src B count	src A count	0x01010010	0x0100#010
dst C count/guide count		src C count/guide ptr		0x01010018	0x0100#018
dst B Pitch		src B pitch		0x01010020	0x0100#020
dst C Pitch/Guide Ptr		src C pitch/guide ptr		0x01010028	0x0100#028
% src transparency/color register value(s)				0x01010030	0x0100#030
% Value does not matter				0x01010038	0x0100#038
Reserved (TC internal state)				0x01010040	0x0100#040
Reserved (TC internal state)				0x01010048	0x0100#048
Reserved (TC internal state)				0x01010050	0x0100#050
Reserved (TC internal state)				0x01010058	0x0100#058
Reserved (TC internal state)				0x01010060	0x0100#060
Reserved (TC internal state)				0x01010068	0x0100#068
Reserved (TC internal state)				0x01010070	0x0100#070
Reserved (TC internal state)				0x01010078	0x0100#078

= PP Number

Figure 3–19. Suspended Packet Transfer Parameters (Big Endian)

MP Addresses	PP Addresses	Word 0 (63–32)		Word 1 (31–0)	
0x01010000	0x0100#000	Original PT entry address		PT options	
0x01010008	0x0100#008	Current src address		Current dst address	
0x01010010	0x0100#010	src B count	src A count	dst B count	dst A count
0x01010018	0x0100#018	src C count/guide count		dst C count/guide count	
0x01010020	0x0100#020	src B pitch		dst B pitch	
0x01010028	0x0100#028	src C pitch/guide ptr		dst C pitch/guide ptr	
0x01010030	0x0100#030	% src transparency/color register value(s)			
0x01010038	0x0100#038	Value does not matter			
0x01010040	0x0100#040	Reserved (TC Internal state)			
0x01010048	0x0100#048	Reserved (TC internal state)			
0x01010050	0x0100#050	Reserved (TC internal state)			
0x01010058	0x0100#058	Reserved (TC internal state)			
0x01010060	0x0100#060	Reserved (TC internal state)			
0x01010068	0x0100#068	Reserved (TC internal state)			
0x01010070	0x0100#070	Reserved (TC internal state)			
0x01010078	0x0100#078	Reserved (TC internal state)			

= PP Number

Suspension begins immediately after a suspend condition arises and any pending external memory column accesses have completed. The external-to-external buffer is not emptied, but the current status of the buffer is saved. This allows packet transfers to be suspended in a consistent manner, regardless of the cause. It also guarantees rapid suspension, because the buffer does not have to be emptied. An urgent priority packet transfer request can be serviced rapidly without waiting for a potentially sluggish packet transfer to empty the buffer.

Suspension into the parameter RAM is performed at the TC crossbar priority level of the pending higher priority request, if that was the cause of the suspension. The PPs or the MP may experience temporary contention until the suspension has been completed. When suspension is due to a timeout, a fault, an error, or a suspend request from the requesting processor, it is performed at the packet transfer's original TC crossbar priority.

The suspended packet transfer parameters can be copied elsewhere and resubmitted from their new location, provided that the linked-list start-address pointer points to the new address. Because the PTS bits indicate that the transfer was suspended, the entire set of suspended parameters are loaded upon resubmission, no matter where they are located. The new starting

address of the suspended parameters must be 128-byte aligned (the seven LSBs must be 0) for the suspended packet transfer to be resubmitted.

3.12.1 Normal Versus Internal-Packet Transfer Parameters

The first eight 64-bit doublewords saved to the packet transfer suspend area are considered *normal* packet transfer parameters. These are the same parameters programmed by the user with a few exceptions. The first 32-bit word contains the original entry address of the transfer that was suspended, rather than the address of the next entry on the linked list. The C count fields contain the current C counts or guide counts, and the guide ptr fields (if guided transfers are used) contain the current value of the guide table pointer field. In addition, the PTS bits within the PT options field are modified to reflect the fact that the parameters represent a suspended transfer. (See section 16.2.2, *PT Options*, and *PT Status (PTS) - Bits 30–29*. The suspend area also contains eight reserved *internal state* doublewords. These fields are automatically loaded when a suspended packet transfer is restarted. They contain information about the intradimensional state of the transfer when suspended. These values are saved so that the suspended transfer can resume exactly where it left off when it is resubmitted.

Chapter 4

Externally Initiated and Video Controller Initiated Packet Transfers

This chapter discusses two other types of packet transfers: externally initiated packet transfers (XPTs) and packet transfers initiated by video controllers (VCPTs). XPTs occur when a peripheral device submits a transfer request directly to the TC instead of going through an MP external interrupt. VCPTs are similar to XPTs, but are initiated by the VC rather than a peripheral device.

Topics in this chapter include:

Topic **Page**

4.1 Externally Initiated Packet Transfers (XPTs)

XPTs provide a mechanism for external devices to submit a packet-transfer request directly to the TC. Although XPTs are designed to accommodate peripheral-device packet transfers, they can specify other types of transfers as well.

4.1.1 XPT Parameter RAM Usage

Dedicated locations within the MP parameter RAM have been reserved to accommodate XPTs. These locations serve as the linked-list start addresses for each XPT and video-controller packet-transfer (VCPT) request. They also serve as the off-chip to off-chip transfer buffer for any XPTs and VCPTs. The MP parameter RAM memory map is shown in Figure 4–1.

XPT4-XPT7 share linked-list start addresses with VCPTs. All XPTs and VCPTs share a single off-chip to off-chip buffer.

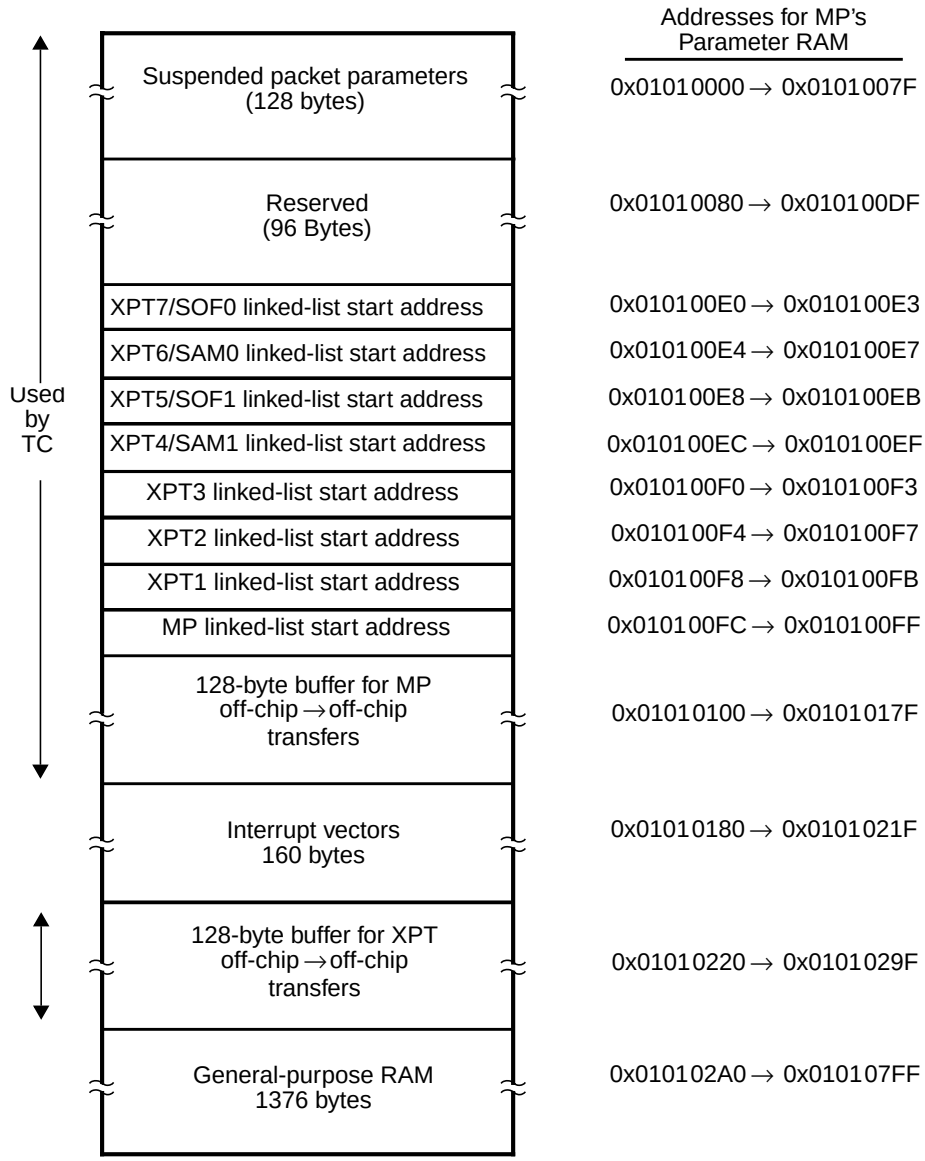
4.1.2 XPT Operation

XPTs operate similarly to normal packet transfers. They can be linked together in a linked list to perform a number of different transfers. The most important difference between XPTs and normal packet transfers is that XPTs cannot be suspended. Once begun, the linked list of an XPT runs to completion.

4.1.3 XPT Errors

If an error or fault occurs during an XPT, the TC sets the XPT bits of the FLTSTS register to indicate which XPT the error or fault occurred in. This also sets the MF bit of the MP's INTPEN register. The TC then terminates the XPT and ignores all further XPT requests until the XPT bits in FLTSTS are cleared by the MP (by writing 1s to the appropriate bits).

Figure 4–1. MP Parameter RAM Contents



4.1.4 XPT Priority

XPT requests are intended for urgent activities such as emptying a full data buffer or moving display refresh data. They have priority above all processor-initiated packet transfers and cache requests. Only host requests, urgent refresh requests, and SRT requests can interrupt an XPT once it has begun. An XPT request suspends any current packet transfer as soon as possible (after PTMIN is reached.)

An XPT request causes REQ[1:0] = 11 to be output to indicate to devices using the host interface that an urgent request is pending. This places XPTs at the same internal request level as VC and urgent refresh requests.

For more information about XPT requests, see Chapter 6, *XPT Initialization*.

4.2 Video Controller-Initiated Packet Transfers (VCPTs)

Video controller initiated packet transfers (VCPTs) are similar to XPTs, but are initiated by a VC event rather than a peripheral device.

4.2.1 VCPT Operation

VCPTs share their linked-list start-address pointers with the XPTs. They also use the same off-chip to off-chip buffer and generate the same status codes as XPTs.

The frame timers can be in either internal or external sync mode, and the SRT controller can be in either display, capture, or merge-capture mode when generating VCPTs. The direction of the transfer is controlled by the packet transfer parameters, not by the sync or SRT mode.

4.2.2 VCPT Priority

VCPTs are prioritized directly above XPTs. They cannot be suspended; their linked list must run to completion. They cannot interrupt XPTs, because XPTs must also run to completion. VCPTs can be interrupted by host requests, urgent refresh requests, and SRT requests (if the other frame memory is using SRT control).

Note:

Unlike SRT requests, VCPT requests cannot interrupt packet transfers. They must wait for the current packet transfer to suspend (after PTMIN is reached) before they can begin execution.

See Chapter 5, *Serial Register Transfer Controller*, in the *MVP Video Controller User's Guide* for more information about VCPTs.

Chapter 5

External-Memory Interface

This chapter describes how the memory-interface inputs indicate the configuration of the external memory to the transfer controller, and how the transfer controller uses the memory-interface output signals to convey information about the type of memory cycle, the request type, and the processor originating the request.

Topic **Page**

5.1 Memory-Interface Signals

Following is a list of the TMS320C80 signals used for external data access.

Signal Name	Direction	Description
A[31:0]	**O**	**Address bus.** Provides a 32-bit byte address in external memory. The address can be multiplexed for DRAM accesses.
AS[2:0]	**I**	**Address shift selection.** Determines shift amount of the column address. Eight shift amounts are supported, including none. Figure 5–1 shows the relation between address shifts and memory size.
BS[1:0]	**I**	**Bus size selection.** Allows dynamic bus sizing for data buses less than 64 bits wide. The decoding of BS[1:0] for bus sizing is shown in Table 5–5. During block-write and load-color-register cycles, the size of the data bus is fixed at 64 bits, and BS[1:0] are used for indicating the type of block write that the memory supports. The decoding of BS[1:0]for block-write types is shown in Figure 5–6.
CLKOUT	**O**	**Output clock.** Allows for synchronous control of external logic.
CT[2:0]	**I**	**Column Timing selection**. CT[2:0] and $\overline{\text{UTIME}}$ together determine what the timing of the current memory cycle will be. External decode logic drives these pins to the appropriate levels, after decoding the address to determine the type of memory currently being accessed. Four of the timings are most suited to interfacing with synchronous DRAM (SDRAM) devices; the other four are most suited to interfacing with static RAM (SRAM) and standard DRAM. The various memory cycle timings supported are shown in Table 5–3.
$\overline{\text{CAS}}$/ DQM[7:0]	**O**	**Column address strobes.** Drives the $\overline{\text{CAS}}$ inputs of DRAMs/VRAMs, and DQM inputs of SDRAM/SGRAM. Eight strobes are provided to allow individual byte access. $\overline{\text{CAS}}$/DQM0 corresponds to the transfer of data on D[7:0], $\overline{\text{CAS}}$/DQM1 to a transfer on D[15:8], etc., regardless of the endian mode.
D[63:0]	**I/O**	**Data bus.** Allows access of up to 64 bits per memory cycle.
$\overline{\text{DBEN}}$	**O**	**Data buffer enable.** Turns on data transceivers.
$\overline{\text{DDIN}}$	**O**	**Data direction indicator.** Provides direction of data through transceivers. When $\overline{\text{DDIN}}$ is low, the data transfer is from external memory to the ‘C80.
DSF	**O**	**Special function pin.** Selects special VRAM/SGRAM functions.
$\overline{\text{FAULT}}$	**I**	**Fault.** Informs the TMS320C80 that a memory fault has occurred.
PS[3:0]	**I**	**Page size select.** Indicates the page size of the memory currently being accessed.

Signal Name	Direction	Description
$\overline{\text{RAS}}$	O	**Row address strobe.** Drives the $\overline{\text{RAS}}$ inputs of DRAMs, VRAMs, and SDRAMs.
READY	I	**Ready.** Indicates that the external device is ready for the memory cycle to be completed. Used to insert wait states into memory cycles.
$\overline{\text{RETRY}}$	I	**Retry.** Indicates that the memory is busy and the TC should begin the cycle again.
$\overline{\text{RL}}$	O	**Row latch.** Indicates that a valid 32-bit address is present on the address bus.
STATUS[5:0]	O	**Status code.** Provides detailed information about the type and origin of the memory cycle currently being performed. The information output during row time is the type of memory cycle. Figure 5–1 shows how STATUS[5:0] are encoded during row time. The information output during column time (while $\overline{\text{CAS}}$/DQM are active) is the type and the origin of the request. Figure 5–2 shows how STATUS[5:0] are encoded during column time.
$\overline{\text{TRG}}$/$\overline{\text{CAS}}$	O	**Transfer/output enable.** Enables DRAM output drivers and VRAM transfer cycles. Used as $\overline{\text{CAS}}$ input for SDRAMs.
$\overline{\text{UTIME}}$	I	**User-timing selection.** Causes the timings of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM[7:0] to be modified so that users can produce their own memory timings. Also used at reset to determine the endian mode in which the TMS320C80 will operate. When used in conjunction with CT[2:0], it indicates the type of SDRAM timing.
$\overline{\text{W}}$	O	**Write enable.** Indicates that a write (or write transfer) cycle is occurring.
$\overline{\text{XPT}}$[2:0]	I	**XPT request.** These encoded inputs are used to submit externally initiated packet transfer requests. Up to seven XPT requests are supported. See Table 6–1 on page 6-2 for these inputs.

5.2 Memory-Cycle Status Codes

During both the address and data phases of a memory cycle, the TMS320C80 outputs status codes on the STATUS[5:0] pins. These status codes provide information about the external cycle being performed. The two types of status codes are

- ❏ Row-time status codes
- ❏ Column-time status codes

For more details about the address and data phases of a memory cycle, see section 7.1, *General Form of an External-Memory Cycle*.

5.2.1 Row-Time Status Codes

During row time (the address phase of a memory cycle), the STATUS[5:0] pins indicate the type of cycle being performed. These codes are shown in Table 5–1. The row-time status code can be latched by the $\overline{RL}$ or $\overline{RAS}$ signal and used by external logic to perform memory-bank decoding or to enable special hardware features.

Table 5–1. Row-Time Status Codes

STATUS[5:0]	Cycle Type	STATUS[5:0]	Cycle Type
0 0 0 0 0 0	Normal read	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	Normal write	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	Refresh	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	SDRAM Deactivate (DCAB)	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	Peripheral device PT read	1 0 0 1 0 0	XPT1 read
0 0 0 1 0 1	Peripheral device PT write	1 0 0 1 0 1	XPT1 write
0 0 0 1 1 0	Reserved	1 0 0 1 1 0	XPT1 PDPT read
0 0 0 1 1 1	Reserved	1 0 0 1 1 1	XPT1 PDPT write
0 0 1 0 0 0	Reserved	1 0 1 0 0 0	XPT2 read
0 0 1 0 0 1	Block write PT	1 0 1 0 0 1	XPT2 write
0 0 1 0 1 0	Reserved	1 0 1 0 1 0	XPT2 PDPT read
0 0 1 0 1 1	Reserved	1 0 1 0 1 1	XPT2 PDPT write
0 0 1 1 0 0	SDRAM Mode Register Set (MRS)	1 0 1 1 0 0	XPT3 read
0 0 1 1 0 1	Load color register	1 0 1 1 0 1	XPT3 write
0 0 1 1 1 0	Reserved	1 0 1 1 1 0	XPT3 PDPT read
0 0 1 1 1 1	Reserved	1 0 1 1 1 1	XPT3 PDPT write
0 1 0 0 0 0	Frame 0 read transfer	1 1 0 0 0 0	XPT4 / SAM1 read
0 1 0 0 0 1	Frame 0 write transfer	1 1 0 0 0 1	XPT4 / SAM1 write
0 1 0 0 1 0	Frame 0 split read transfer	1 1 0 0 1 0	XPT4 / SAM1 PDPT read
0 1 0 0 1 1	Frame 0 split write transfer	1 1 0 0 1 1	XPT4 / SAM1 PDPT write
0 1 0 1 0 0	Frame 1 read transfer	1 1 0 1 0 0	XPT5 / SOF1 read
0 1 0 1 0 1	Frame 1 write transfer	1 1 0 1 0 1	XPT5 / SOF1 write
0 1 0 1 1 0	Frame 1 split read transfer	1 1 0 1 1 0	XPT5 / SOF1 PDPT read
0 1 0 1 1 1	Frame 1 split write transfer	1 1 0 1 1 1	XPT5 / SOF1 PDPT write
0 1 1 0 0 0	Reserved	1 1 1 0 0 0	XPT6 / SAM0 read
0 1 1 0 0 1	Reserved	1 1 1 0 0 1	XPT6 / SAM0 write
0 1 1 0 1 0	Reserved	1 1 1 0 1 0	XPT6 / SAM0 PDPT read
0 1 1 0 1 1	Reserved	1 1 1 0 1 1	XPT6 / SAM0 PDPT write
0 1 1 1 0 0	PT read transfer	1 1 1 1 0 0	XPT7 / SOF0 read
0 1 1 1 0 1	PT write transfer	1 1 1 1 0 1	XPT7 / SOF0 write
0 1 1 1 1 0	Reserved	1 1 1 1 1 0	XPT7 / SOF0 PDPT read
0 1 1 1 1 1	Idle	1 1 1 1 1 1	XPT7 / SOF0 PDPT write

The row-time status codes are defined as follows:

Normal read. Output for any normal read cycle; generated by a packet transfer, cache miss, or DEA request.

Normal write. Output for any normal write cycle; generated by a packet transfer, data-cache write back or DEA request.

Refresh. Output during trickle-refresh cycles, and during burst-refresh cycles generated by an urgent refresh request, from the refresh controller.

SDRAM Deactivate (DCAB). Output during SDRAM deactivate cycles. This type of cycle occurs only after reset as part of the SDRAM power-up sequence. If SDRAMs are present in the system, the TC performs this cycle to ensure that all SDRAM banks are deactivated before performing the power-up refresh and mode-initialization sequences. This code should not be confused with the column-time deactivate code issued at the end of an SDRAM access sequence; a separate column-time status code exists for that purpose (see the final entry in Table 5–2).

Peripheral device PT (packet transfer) read. Output for a memory-read cycle occurring as a result of a processor requesting a peripheral-device-packet transfer. The status code indicates that the data read from memory on the subsequent column accesses should be latched by the peripheral device.

Peripheral device PT write. Output for a memory-write cycle occurring as a result of a processor requesting a peripheral-device packet transfer. The status code indicates that the TMS320C80 will be placing its data bus in high-impedance during the subsequent column access so that the peripheral device may drive the bus.

Block write. Output during a block-write cycle to VRAMs/SGRAMs; generated by a packet transfer with a block-write access mode.

SDRAM Mode Register Set (MRS). The TC performs this cycle only after reset. After all power-up refresh cycles have completed, an MRS cycle is performed (if SDRAM devices are present in the system) to initialize SDRAM operation for the TMS320C80.

Load color register. Output during the color-register-load portion of a packet transfer that has specified block write as its access mode. This cycle is used to place data in the color registers of system VRAMs/SGRAMs.

Frame 0/1 read transfer. Output during full read-transfer cycles requested by the video controller. These cycles perform full read-transfer cycles on system

VRAMs/SGRAMs. STATUS[2] is 0 for Frame Memory 0 cycles and 1 for Frame Memory 1 cycles.

Frame 0/1 write transfer. Output during full-write-transfer cycles requested by the video controller. These cycles perform full write-transfer cycles on system VRAMs/SGRAMs. The STATUS[2] signal is 0 for Frame Memory 0 cycles and 1 for Frame Memory 1 cycles.

Frame 0/1 split-read transfer. Output during split-read-transfer cycles requested by the video controller. These cycles transfer from a row of VRAM/SGRAM memory into half of the serial register. STATUS[2] is 0 for Frame Memory 0 cycles and 1 for Frame Memory 1 cycles.

Frame 0/1 split-write transfer. Output during split write-transfer cycles requested by the video controller. These cycles transfer half of a VRAM/SGRAM serial register into the memory array. STATUS[2] is 0 for requests from Frame Memory 0 and 1 for requests from Frame Memory 1.

PT-read transfer. Output during source cycles of a packet transfer that specifies serial-register transfer as its access mode. This cycle performs a read transfer (memory-to-register transfer) on system VRAMs/SGRAMs.

PT-write transfer. Output during destination cycles of transfers that specify serial-register transfer as the access mode. This cycle performs a write transfer (register-to-memory transfer) on system VRAMs/SGRAMs.

Idle. Output when the external-memory interface is in the *r1* state, but is not actually starting a memory access. By monitoring the status code, memory-interface logic can detect when the TMS320C80 actually begins a memory cycle and determine when a valid row address is present on the address bus. For more information about memory states, see section 7.3, *Standard DRAM Interface* and section 7.6.1, *Row-Time Wait States*.

XPTn read. Output for a normal read cycle generated by XPT*n*.

XPTn write. Output for normal write cycle generated by XPT*n*.

XPTn PDPT read. Output for a memory-read cycle that occurs as a result of a peripheral-device-packet transfer requested by XPT*n*. This output indicates that the data read from memory on subsequent column accesses should be latched by the peripheral device that generated the XPT*n* request.

XPTn PDPT write. Output for a memory-write cycle that occurs as a result of a peripheral-device-packet transfer requested by XPT*n*. This output indicates

that the TMS320C80 will be placing its data bus in high-impedance during subsequent column accesses, so that the peripheral device can drive the bus.

SAM0/1. Identical to the XPT6 and XPT4 codes respectively, but resulting from a SAM overflow event from Frame Memory 0 or Frame Memory 1. See Chapter 5, *Serial Register Transfer Controller*, in the *TMS320C80 Video Controller User's Guide* for more information.

SOF0/1. Identical to the XPT7 and XPT5 codes respectively, but resulting from a Start-of-Field event by Frame Timer 0 or Frame Timer 1. See Chapter 5, *Serial Register Transfer Controller*, in the *TMS320C80 Video Controller User's Guide* for more information.

Column-Time Status Codes. During column time (data subcycle), the information on the STATUS[5:0] pins changes to provide detail about the request type and the processor originating the request. These processor-activity codes are shown in Table 5–2. They provide no information about cycle status, and are intended mainly to facilitate system debug.

5.2.2 Column-Time Status Codes

During column time (data subcycle), the information on the STATUS[5:0] pins changes to provide details about the cycle and its requesting processor. These processor activity codes are shown in Table 5–2.

Table 5–2. Column-Time Processor Activity Codes

STATUS[5:0]	Processor Activity	STATUS[5:0]	Processor Activity
0 0 0 0 0 0	PP0 low-priority packet transfer	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	PP0 high-priority packet transfer	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	PP0 instruction cache	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	PP0 DEA	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	PP1 low-priority packet transfer	1 0 0 1 0 0	Reserved
0 0 0 1 0 1	PP1 high-priority packet transfer	1 0 0 1 0 1	Reserved
0 0 0 1 1 0	PP1 instruction cache	1 0 0 1 1 0	Reserved
0 0 0 1 1 1	PP1 DEA	1 0 0 1 1 1	Reserved
0 0 1 0 0 0	PP2 low-priority packet transfer	1 0 1 0 0 0	Reserved
0 0 1 0 0 1	PP2 high-priority packet transfer	1 0 1 0 0 1	Reserved
0 0 1 0 1 0	PP2 instruction cache	1 0 1 0 1 0	Reserved
0 0 1 0 1 1	PP2 DEA	1 0 1 0 1 1	Reserved
0 0 1 1 0 0	PP3 low-priority packet transfer	1 0 1 1 0 0	Reserved
0 0 1 1 0 1	PP3 high-priority packet transfer	1 0 1 1 0 1	Reserved
0 0 1 1 1 0	PP3 instruction cache	1 0 1 1 1 0	Reserved
0 0 1 1 1 1	PP3 DEA	1 0 1 1 1 1	Reserved
0 1 0 0 0 0	MP low-priority packet transfer	1 1 0 0 0 0	Reserved
0 1 0 0 0 1	MP high-priority packet transfer	1 1 0 0 0 1	Reserved
0 1 0 0 1 0	MP urgent packet transfer (low)	1 1 0 0 1 0	Reserved
0 1 0 0 1 1	MP urgent packet transfer (high)	1 1 0 0 1 1	Reserved
0 1 0 1 0 0	XPT/VCPT in progress	1 1 0 1 0 0	Reserved
0 1 0 1 0 1	XPT/VCPT complete	1 1 0 1 0 1	Reserved
0 1 0 1 1 0	MP instruction cache (low)	1 1 0 1 1 0	Reserved
0 1 0 1 1 1	MP instruction cache (high)	1 1 0 1 1 1	Reserved
0 1 1 0 0 0	MP DEA (low)	1 1 1 0 0 0	Reserved
0 1 1 0 0 1	MP DEA (high)	1 1 1 0 0 1	Reserved
0 1 1 0 1 0	MP data cache (low)	1 1 1 0 1 0	Reserved
0 1 1 0 1 1	MP data cache (high)	1 1 1 0 1 1	Reserved
0 1 1 1 0 0	Frame controller 0	1 1 1 1 0 0	Reserved
0 1 1 1 0 1	Frame controller 1	1 1 1 1 0 1	Reserved
0 1 1 1 1 0	Urgent refresh	1 1 1 1 1 0	Reserved
0 1 1 1 1 1	Idle	1 1 1 1 1 1	Write Drain/ SDRAM DCAB

Note: (low) - low-priority MP request
(high) - high-priority MP request

5.3 Memory Configuration

For the TC to properly communicate with external memory, it must identify the type of memory it is accessing. This identification is accomplished at row time using the TMS320C80's memory-interface inputs AS[2:0], BS[1:0], CT[2:0], PS[3:0], and $\overline{\text{UTIME}}$. This allows the TMS320C80 to support different memory types, sizes, and speeds, and allows the TC to dynamically change bus sizing and page sizing according to the memory address.

At the start of an external-memory cycle, the TMS320C80 outputs the external-memory address and row-time status code and then samples the memory-interface inputs to determine the memory type. This gives external logic time to decode the address and status, and drive the memory identification signals to their proper levels. The selected memory type remains in effect until the next address subcycle.

5.3.1 Memory Size and Address Multiplexing

External logic decodes the address output by the TMS320C80 at row time and supplies a 3-bit shift code on the AS[2:0] inputs. The TC samples and latches this value, and uses it to determine if the address should be multiplexed as required for dynamic memories or remain unmultiplexed for use with static memories and peripherals.

Figure 5–1 shows how the sampled AS[2:0] value affects the address output at column time. If the value is 000, an unshifted 32-bit address is output for each subsequent column access. If the value is nonzero, then subsequent column addresses are shifted. Shift values range from 8 to 14 bits, meaning that dynamic memories with 8 to 14 address pins (64K x n to 256M x n array sizes) are supported. A[2:0] always outputs byte address bits 0 to 2 regardless of the shift amount. This supports byte-access control for peripheral devices during peripheral device packet transfers.

Figure 5–1. Row/Column Address Multiplexing on A[31:0]

A Pins

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Row Time

AS[2:0] at Row Time																															RAM Type
000	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Static
001	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	2	1	0	64K x n (128Kxn)
010	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	2	1	0	256K x n
011	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	2	1	0	1M x n
100	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	2	1	0	4M x n
101	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	2	1	0	16M x n
110	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	x	2	1	0	64M x n
111	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	x	x	2	1	0	256M x n

Column Time

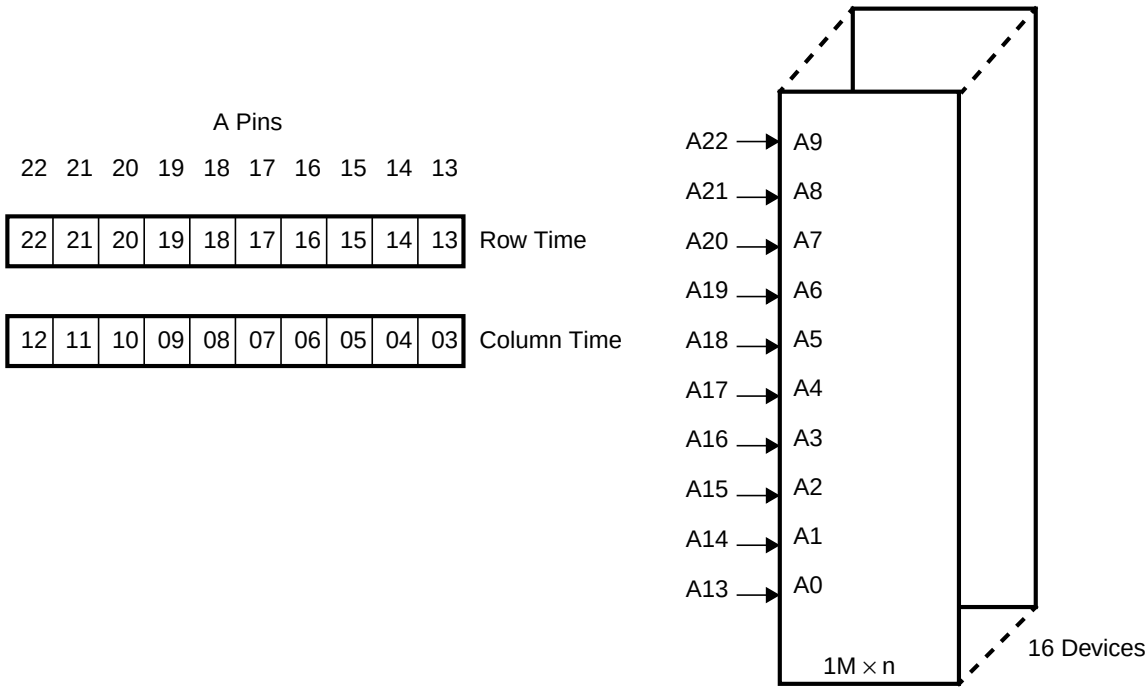
For example, if 1M x 4 DRAMs are connected to the data bus in a 64-bit wide configuration, the memories could be connected to the address bus as shown in Figure 5–2. The memories require ten bits each of row and column address. A(2:0) of the TMS320C80's address bits represent the byte address and can be ignored because the memory bank is 64 bits wide and the individual bytes are controlled by the $\overline{\text{CAS}}$/DQM[7:0] strobes. This means that the memories need 20 contiguous address bits starting with bit A(3). Figure 5–1 shows that an AS[2:0] value of 011 provides 10 bits of multiplexed address.

Note:

In notation used in this chapter, [ ] (brackets) indicate pin locations, and ( ) (parentheses) indicate bit locations. Thus, A[*n*] refers to the address pin *n*, and A(*n*) refers to the address bit *n*. Because of address shifting, they do not necessarily represent the same thing.

Because the starting address bit A(3) corresponds to address pin A[13] at column time, the DRAM address pins are connected to the ten TMS320C80 address pins starting with A[13]. This results in the addressing shown in Figure 5–2.

Figure 5–2. Address Shift for 1M X 4 RAM (64-Bit Bus)



5.3.2 Memory-Speed and Column-Timing Selection

The TC supports ten basic sets of memory timings. Four are best suited for access to SRAM and standard DRAM, and six are best suited for access to synchronous DRAM (SDRAM).

The timing used is determined by the CT[2:0] and $\overline{\text{UTIME}}$ inputs. External decode logic drives these pins to the appropriate levels after decoding the address, to determine the type of memory currently being accessed. Figure 5–3 shows how these signals are decoded.

Table 5–3. CT[2:0] Codes

CT[2:0]	$\overline{\text{UTIME}}$	Memory Timing
0 0 0	1	Pipelined (Burst Length 1) SDRAM with CAS Latency of 2
0 0 0	0	Reserved
0 0 1	1	Pipelined (Burst Length 1) SDRAM with CAS Latency of 3
0 0 1	0	Pipelined (Burst Length 1) SDRAM with CAS Latency of 4
0 1 0	1	Interleaved (Burst Length 2) SDRAM with CAS Latency of 2
0 1 0	0	Reserved
0 1 1	1	Interleaved (Burst Length 2) SDRAM with CAS Latency of 3
0 1 1	0	Interleaved (Burst Length 2) SDRAM with CAS Latency of 4
1 0 0	X	Pipelined one cycle per column
1 0 1	X	Nonpipelined one cycle per column
1 1 0	X	Nonpipelined two cycles per column
1 1 1	X	Nonpipelined three cycles per column

External logic must provide the CT code to the TMS320C80 at the beginning of each memory access, to select the cycle timing. A CT[2] value of 0 selects the SDRAM cycle timings. A CT[2] value of 1 selects the SRAM/standard-DRAM cycle timings.

SDRAM Timings (CT[2] = 0). These timings support CAS latencies of two to four cycles and burst lengths of 1 or 2. CAS latency applies to read cycles only and is the clock delay between the column address and the input data. The 3- and 4-cycle latencies allow the use of slower SDRAMs. If the SDRAM is a pipelined architecture that supports a new column address on every cycle, the TC can be configured to use a burst length of 1. Otherwise, the TC must be configured for a burst length of 2.

SRAM/Standard DRAM Timings (CT[2] = 1). These timings allow the choice of having one, two, or three clock cycles-per-column access, without having

to use wait states. Although all three types are DRAM-like in nature, each is best suited to a particular type of DRAM or SRAM.

- *Pipelined one cycle per column.* This timing is used with DRAMs/VRAMs that support pipelined page-mode cycles, but it could also be used with synchronous SRAM or other devices that support pipelining. It can begin a new column access on every cycle but the access is spread over two cycles so that the address of the current access coincides with the data of the previous cycle.

- *Nonpipelined one cycle per column.* This timing is used with devices that have very fast access times. It provides one clock cycle of access time per column.

- *Nonpipelined two cycles per column.* These cycles provide two clock cycles of column access time for SRAMs and fast DRAM devices. They also add one cycle to the row address time.

- *Nonpipelined three cycles per column.* This cycle is used with DRAM and other slower devices. In addition to providing three clock cycles of column-access time, it adds two clock cycles to the row-access time.

The selected column timing remains in effect for the entire page until the next row access occurs. For standard DRAM timings (CT[2] = 1), the $\overline{\text{UTIME}}$ input can be used to modify the way the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM[7:0] outputs operate, but does not change the length of the row or column-access times. For SDRAM timings (CT[2] = 0), $\overline{\text{UTIME}}$ affects the memory timing as shown in Table 5–3.

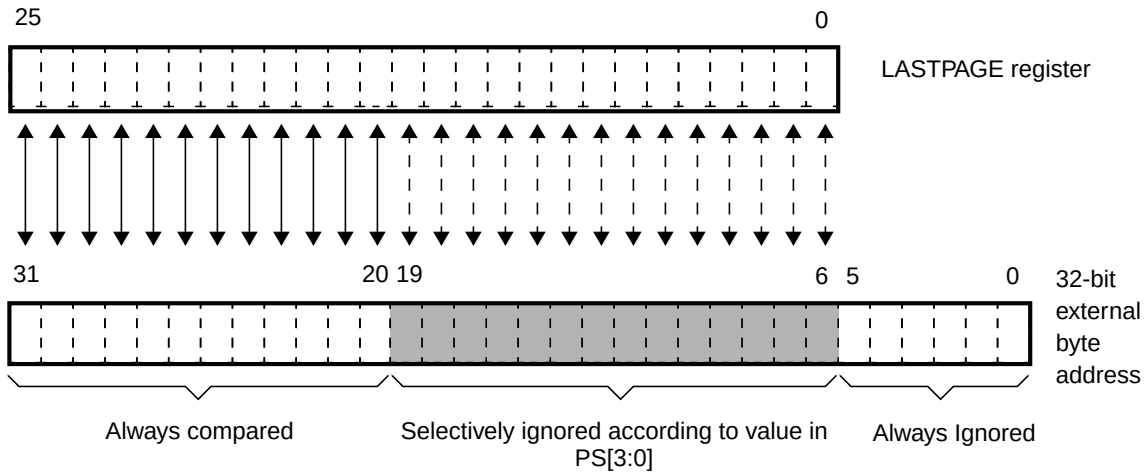
5.3.3 Dynamic Page Sizing

The TC performs page-mode cycles whenever possible. It must be able to detect crossing a page (row) boundary for the memory it is currently accessing so that it can perform a row access on the new page. The page size for the current access is indicated by the 4-bit value placed by external logic on the PS[3:0] pins. The TC samples these pins at row time and determines which changing address bits indicate a page change. The TC retains the sampled value until the next row access.

Whenever an external-memory access occurs, the TC records the 26 MSBs of the address in its internal LASTPAGE register. The address of each subsequent column access is then compared to this value, as shown in Figure 5–3. The value that was entered on PS[3:0] is used to selectively ignore up to 14 of the LSBs of LASTPAGE during the comparison. The 12

MSBs of the next address are always compared, and the 6 LSBs are always ignored. If there is a difference between the enabled bits of LASTPAGE and the next memory address, then the page has changed and the next memory access begins with a row-address cycle. If PS[3:0] = 1000, then page mode is disabled and any subsequent cycles begin with another row access.

Figure 5–3. Page-Size Control



For example, assume the memory being accessed consists of 16 1M x 4 DRAMs connected as a 64-bit data bus with no interleaving with other banks. Each memory device has a row size of 2^{10} bits. Any location within a single row can be accessed during a DRAM's page-mode cycle; the page size for this configuration would be:

2^{10} locations/page x 8 bytes/location = 8K bytes/page

As Table 5–4 shows, this corresponds to a PS[3:0] value of 1011. When the TC samples PS[3:0] = 1011 at row time, it checks only bits 31-13 of subsequent accesses to determine if a page boundary has been crossed. Notice that this corresponds to the DRAM's row address bits and bank-decode bits for this particular configuration. The compared address bits and page sizes for the 16 possible PS[3:0] values are shown in Table 5–4.

Note:

The page size indicated on PS[3:0] may not correspond to the shift amount indicated on AS[2:0] because multiple banks may be interleaved.

Table 5–4. Page-Size Values

PS[3:0]	Address Bits Compared	Page Size (Bytes)
0 0 0 0	A(31:6)	64
0 0 0 1	A(31:7)	128
0 0 1 0	A(31:8)	256
0 0 1 1	A(31:9)	512
0 1 0 0	A(31:10)	1K
0 1 0 1	A(31:18)	256K
0 1 1 0	A(31:19)	512K
0 1 1 1	A(31:20)	1M
1 0 0 0	A(31:0)	1-8†
1 0 0 1	A(31:11)	2K
1 0 1 0	A(31:12)	4K
1 0 1 1	A(31:13)	8K
1 1 0 0	A(31:14)	16K
1 1 0 1	A(31:15)	32K
1 1 1 0	A(31:16)	64K
1 1 1 1	A(31:17)	128K

† PS[3:0] =1000 disables page mode cycles so that the page size is effectively the data bus size (1, 2, 4, or 8 bytes).

The LASTPAGE register is treated as invalid after reset, host accesses, faults, retries, changes in direction of the access (read vs. write), or changes in row status. This forces a row access to always occur before subsequent accesses. In addition, LASTPAGE is considered invalid both before and after refresh cycles, SRT cycles, and packet transfer-generated serial register transfer cycles so that these always occur as single nonpage-mode cycles. In the special case of peripheral device packet transfers or XPT/VCPT cycles, the transfers always begin with a row access to ensure that the corresponding peripheral device transfer or XPT status code is output. LASTPAGE is considered invalid at the completion of the transfer so that a new status code can be output. During the transfer, however, LASTPAGE behaves normally to allow the peripheral device or XPT transfer to use page-mode cycles wherever possible. Generally, LASTPAGE is considered invalid anytime the row status must be changed.

5.3.4 Dynamic Bus Sizing

The BS[1:0] pins are sampled at row time to determine the bus size for the current access. The TC supports bus sizes of 8, 16, 32, and 64 bits, as shown in Table 5–5.

Table 5–5. BS[1:0] Bus Size Codes

BS[1:0]	Bus Size
0 0	8 bits
0 1	16 bits
1 0	32 bits
1 1	64 bits

Setting the bus size determines the maximum number of bytes that the TC can transfer during each column access. If the number of bytes requested exceeds the bus size, the TC automatically performs multiple accesses to complete the transfer. The selected bus size also determines which portion of the data bus is used for the transfer, as follows:

- ❏ For 64-bit buses, the entire bus is available for transfers.
- ❏ For 32-bit buses, D[63:32] is used for big-endian mode, and D[31:0] is used for little-endian mode.
- ❏ For 16-bit buses, D[63:48] is used for big-endian mode, and D[15:0] is used for little-endian mode.
- ❏ For 8-bit buses, D[63:56] is used for big-endian mode, and D[7:0] is used for little-endian mode.

This is explained in more detail in section 8.1, *Selecting Big- or Little-Endian Format*. No matter what bus size is used, the TC always aligns data to the proper portion of the bus and activates the appropriate $\overline{\text{CAS}}$/DQM strobes to ensure that only valid bytes are transferred.

During block-write and load-color register cycles, the BS[1:0] pins are used for a different purpose. Because block writes are only supported for 64-bit buses, the bus size information is not needed. Instead, BS[1:0] are used to indicate the type of block write that the addressed memory supports. The values for BS[1:0] during these cycles are shown in Table 5–6. Block writes are discussed in detail in section 11.3, *The Block-Write Modes*.

Table 5–6. BS[1:0] Block-Write Codes

BS[1:0]	Block-Write Mode
0 0	Simulated
0 1	Reserved
1 0	4x
1 1	8x

Chapter 6

XPT Initialization

This chapter describes the XPT request mechanism, how XPTs are enabled and disabled, and the memory status codes associated with XPTs.

Topics in this chapter include:

Topic **Page**

6.1 XPT Request Inputs

XPTs are requested via the TMS320C80's $\overline{\text{XPT}}$[2:0] inputs. These inputs are encoded to support up to seven different XPT requests. The inputs are active low and are encoded as shown in Table 6–1.

Table 6–1. XPT Request Codes

$\overline{\text{XPT2}}$	$\overline{\text{XPT1}}$	$\overline{\text{XPT0}}$	Active Request
1	1	1	No request
1	1	0	$\overline{\text{XPT 1}}$
1	0	1	$\overline{\text{XPT 2}}$
1	0	0	$\overline{\text{XPT 3}}$
0	1	1	$\overline{\text{XPT 4}}$
0	1	0	$\overline{\text{XPT 5}}$
0	0	1	$\overline{\text{XPT 6}}$
0	0	0	$\overline{\text{XPT 7}}$

XPT inputs are internally synchronized and passed to the TC. An XPT request should remain valid until acknowledged by the cycle-type code output on STATUS[5:0] to assure recognition. An input code can be changed to indicate another request, but the TC services the first code that is successfully synchronized. The cycle-type status code should always be checked to determine which XPT request the TC is servicing.

6.2 Row-Time Status Codes

When ready to begin servicing an XPT request, the TC outputs the appropriate XPT cycle-type code on the STATUS[5:0] pins. The condition STATUS[5]=1, indicates that an XPT or VCPT request is being serviced. The other status bits determine which transfer is being serviced and whether a read, write, peripheral-device read, or peripheral-device write cycle is occurring.

Table 6–2. Row-Time Status Codes

STATUS[5:0]	Cycle Type	STATUS[5:0]	Cycle Type
0 0 0 0 0 0	Normal read	1 0 0 0 0 0	Reserved
0 0 0 0 0 1	Normal write	1 0 0 0 0 1	Reserved
0 0 0 0 1 0	Refresh	1 0 0 0 1 0	Reserved
0 0 0 0 1 1	SDRAM DCAB	1 0 0 0 1 1	Reserved
0 0 0 1 0 0	Peripheral-device PT read	1 0 0 1 0 0	XPT1 read
0 0 0 1 0 1	Peripheral-device PT write	1 0 0 1 0 1	XPT1 write
0 0 0 1 1 0	Reserved	1 0 0 1 1 0	XPT1 PDPT read
0 0 0 1 1 1	Reserved	1 0 0 1 1 1	XPT1 PDPT write
0 0 1 0 0 0	Reserved	1 0 1 0 0 0	XPT2 read
0 0 1 0 0 1	Block-write PT	1 0 1 0 0 1	XPT2 write
0 0 1 0 1 0	Reserved	1 0 1 0 1 0	XPT2 PDPT read
0 0 1 0 1 1	Reserved	1 0 1 0 1 1	XPT2 PDPT write
0 0 1 1 0 0	SDRAM MRS	1 0 1 1 0 0	XPT3 read
0 0 1 1 0 1	Load-color register	1 0 1 1 0 1	XPT3 write
0 0 1 1 1 0	Reserved	1 0 1 1 1 0	XPT3 PDPT read
0 0 1 1 1 1	Reserved	1 0 1 1 1 1	XPT3 PDPT write
0 1 0 0 0 0	Frame 0 read transfer	1 1 0 0 0 0	XPT4 / SAM1 read
0 1 0 0 0 1	Frame 0 write transfer	1 1 0 0 0 1	XPT4 / SAM1 write
0 1 0 0 1 0	Frame 0 split read transfer	1 1 0 0 1 0	XPT4 / SAM1 PDPT read
0 1 0 0 1 1	Frame 0 split write transfer	1 1 0 0 1 1	XPT4 / SAM1 PDPT write
0 1 0 1 0 0	Frame 1 read transfer	1 1 0 1 0 0	XPT5 / SOF1 read
0 1 0 1 0 1	Frame 1 write transfer	1 1 0 1 0 1	XPT5 / SOF1 write
0 1 0 1 1 0	Frame 1 split read transfer	1 1 0 1 1 0	XPT5 / SOF1 PDPT read
0 1 0 1 1 1	Frame 1 split write transfer	1 1 0 1 1 1	XPT5 / SOF1 PDPT write
0 1 1 0 0 0	Reserved	1 1 1 0 0 0	XPT6 / SAM0 read
0 1 1 0 0 1	Reserved	1 1 1 0 0 1	XPT6 / SAM0 write
0 1 1 0 1 0	Reserved	1 1 1 0 1 0	XPT6 / SAM0 PDPT read
0 1 1 0 1 1	Reserved	1 1 1 0 1 1	XPT6 / SAM0 PDPT write
0 1 1 1 0 0	PT read transfer	1 1 1 1 0 0	XPT7 / SOF0 read
0 1 1 1 0 1	PT write transfer	1 1 1 1 0 1	XPT7 / SOF0 write
0 1 1 1 1 0	Reserved	1 1 1 1 1 0	XPT7 / SOF0 PDPT read
0 1 1 1 1 1	Idle	1 1 1 1 1 1	XPT7 / SOF0 PDPT write

After a peripheral has generated an XPT request, it should monitor the STATUS[5:0] pins to determine when its request is being serviced. The XPT status code indicates when the peripheral can remove its request from the XPT pins. It also indicates if the peripheral needs to begin reading or writing data (in the case of a peripheral-device mode transfer). An XPT request must be removed from the XPT inputs within two clock cycles of the falling edge of $\overline{RL}$ to ensure that a second XPT request does not occur. Once the TC begins servicing an XPT request, it ignores all other XPT inputs until it completes the current XPT.

Note:

There is no provision for signalling the requesting peripheral if the XPT is an on-chip to on-chip transfer (since no external row access is required). If using an XPT to generate an on-chip to on-chip transfer, link to another packet transfer that causes an external bus access to occur so that the requesting peripheral can determine if its request has been serviced.

6.3 Column-Time Status Codes

At column time, the STATUS[5:0] pins output the processor activity code to indicate which processor has requested the cycle. There are two pertinent XPT column-time codes that are shown in Table 6–1.

The *XPT-in-progress* code indicates that the column cycle is part of the XPT currently being serviced. The *XPT complete* code indicates the completion of the transfer. It is generated on the last column access of the transfer when either the S (stop) bit or I (interrupt when finished) bit is set in the PT options field of the parameter table. An XPT does not generate an interrupt to any processor upon completion; only the XPT complete code on STATUS[5:0].

6.4 Enabling XPTs

XPTs cannot occur unless they have been enabled. To do this, set the X bit (bit 27) of the MP's CONFIG register to 1. Clearing this bit to 0, causes the TC to ignore all XPT requests. Reset also clears the X bit to 0 to prevent erroneous requests from occurring.

Table 6–3. Column-Time Processor Activity Codes

STATUS[5:0]	Processor Activity
0 0 0 0 0 0	PP0 low-priority packet transfer
0 0 0 0 0 1	PP0 high-priority packet transfer
0 0 0 0 1 0	PP0 instruction cache
0 0 0 0 1 1	PP0 DEA
0 0 0 1 0 0	PP1 low-priority packet transfer
0 0 0 1 0 1	PP1 high-priority packet transfer
0 0 0 1 1 0	PP1 instruction cache
0 0 0 1 1 1	PP1 DEA
0 0 1 0 0 0	PP2 low-priority packet transfer
0 0 1 0 0 1	PP2 high-priority packet transfer
0 0 1 0 1 0	PP2 instruction cache
0 0 1 0 1 1	PP2 DEA
0 0 1 1 0 0	PP3 low-priority packet transfer
0 0 1 1 0 1	PP3 high-priority packet transfer
0 0 1 1 1 0	PP3 instruction cache
0 0 1 1 1 1	PP3 DEA
0 1 0 0 0 0	MP low-priority packet transfer
0 1 0 0 0 1	MP high-priority packet transfer
0 1 0 0 1 0	MP urgent packet transfer (low)
0 1 0 0 1 1	MP urgent packet transfer (high)
0 1 0 1 0 0	XPT/VCPT in progress
0 1 0 1 0 1	XPT/VCPT complete
0 1 0 1 1 0	MP instruction cache (low)
0 1 0 1 1 1	MP instruction cache (high)
0 1 1 0 0 0	MP DEA (low)
0 1 1 0 0 1	MP DEA (high)
0 1 1 0 1 0	MP data cache (low)
0 1 1 0 1 1	MP data cache (high)
0 1 1 1 0 0	Frame controller 0
0 1 1 1 0 1	Frame controller 1
0 1 1 1 1 0	Urgent refresh
0 1 1 1 1 1	Idle
1 0 0 0 0 0 through 1 1 1 1 1 0	Reserved
1 1 1 1 1 1	Write drain / SDRAM DCAB

Note: (low) - low-priority MP request
(high) - high-priority MP request

Chapter 7

Overview of External-Memory Cycles

The transfer controller has an external-memory controller that generates external-memory cycles. This controller contains a state machine that generates a sequence of states to control the transition of the memory-interface signals. This chapter discusses how memory cycles are generated and controlled.

This chapter covers the following topics:

Topic **Page**

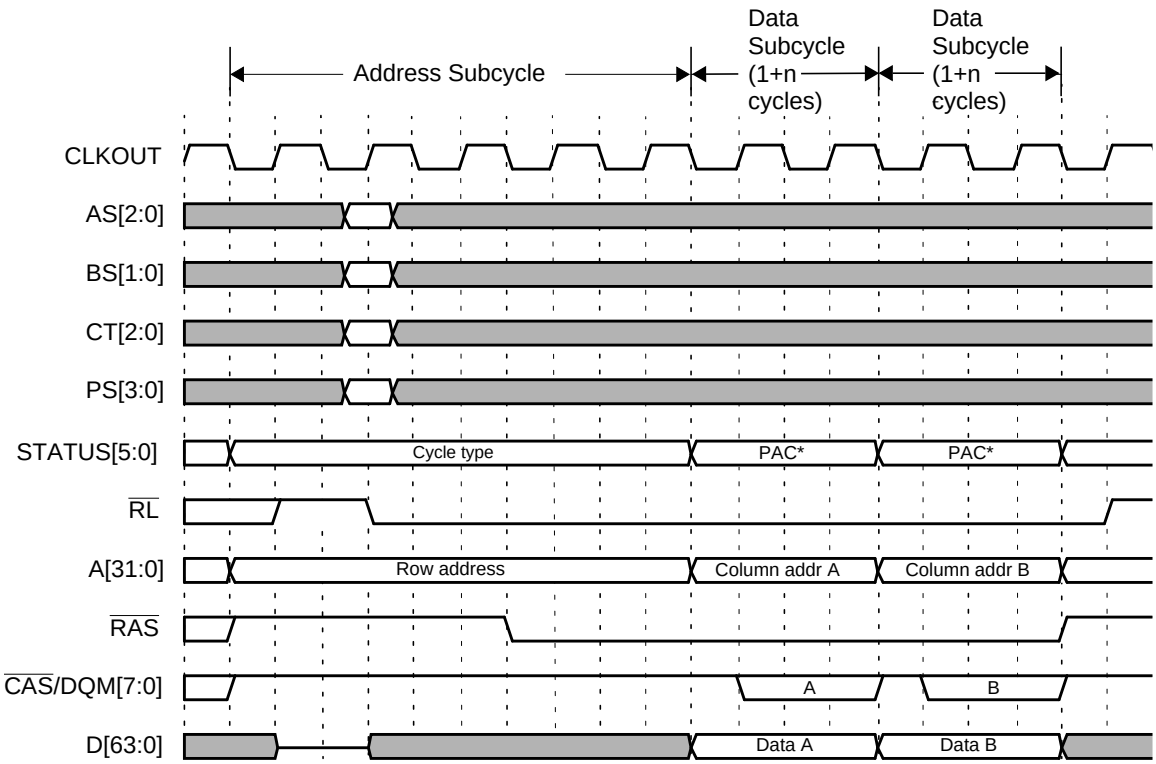
7.1 General Form of an External-Memory Cycle

Each external-memory cycle generated by the TMS320C80 is at least five machine states in duration (except for page-mode cycles). A *machine state* is one clock period long and begins on the falling edge of CLKOUT. As Figure 7–1 shows, a memory cycle has two parts:

- ❏ Address subcycle
- ❏ Data subcycle

Page-mode cycles are a variation of this form in which an access has one address subcycle and multiple data subcycles.

Figure 7–1. General Form of a Local Memory Cycle



* PAC - Processor Activity Code

7.1.1 The Address Subcycle

The address subcycle begins with the first machine state of the external memory cycle and is at least four machine states long. The address and status code for the access are output at this time. This portion of the cycle is also called the row address time because the row address for DRAMs and VRAMs is latched at this time. During SDRAM accesses, the row activation is performed at this time.

The A[31:0] bus outputs a 32-bit address that points to the beginning byte of the 64-bit word currently being accessed. (The access can be anywhere from one to eight bytes, depending on the starting byte and the amount of data to be transferred.) This address is used to decode external memory space. The external decode logic sends a number of signals back to the ‘C80 to indicate the addressing (AS[2:0]). speed (CT[2:0]), page size (PS[3:0]), and data bus width (BS[1:0]) of the device(s) being accessed. This information is used to determine the length of the address subcycle as well as the length and addressing of the data subcycle(s). The address and status can be latched with $\overline{\text{RL}}$ or $\overline{\text{RAS}}$.

The address subcycle is automatically extended beyond four machine states as required by the access type being performed. It can also be extended by the insertion of wait states.

Note:
In notation used in this chapter, [ ] (brackets) indicate pin locations, and ( ) (parentheses) indicate bit locations. Thus, A[*n*] refers to the address pin *n*, and A(*n*) refers to the address bit *n*. Because of address shifting, they do not necessarily represent the same thing.

7.1.2 The Data Subcycle

The data subcycle is at least one machine state long and immediately follows the address subcycle. The column address for DRAMs and VRAMs is output at this time, and data is transferred between the ‘C80 and memory. This portion of the memory cycle is often called the **column address time**. During SDRAM accesses, read and write commands are performed at this time.

The D[63:0] bus transfers the data between the ‘C80 and memory. Data is either driven out for a write cycle or latched in for a read cycle. The position of valid data on the bus is determined by the ‘C80’s endian mode, the amount

of data being transferred, and the width of the memory (see Chapter 8, *Big- and Little-Endian Formats*).

The column address output during this time is a shifted version of the 32- bit byte address. The alignment of addresses on the A[31:0] bus is determined by the address shift (AS[2:0]) value entered into the ‘C80 during the address subcycle. This is discussed in more detail in subsection 5.3.1, *Memory Size and Address Multiplexing*.

The length of the data subcycle is normally one to three machine states long, as determined by the column timing value (CT[2:0]) entered into the ‘C80 during the address subcycle (see subsection 5.3.2, *Memory-Speed and Column-Timing Selection*). Devices requiring longer access times can insert wait states in either the address or the data subcycle.

Whenever the current memory access is in the same direction and is within the same memory page as the previous access, based on the PS[3:0] inputs at row time, page-mode cycles will be used. Page-mode cycles consist of one address subcycle followed by multiple data subcycles. Data need not be contiguous, only within the same memory page.

7.2 Overview of Memory States

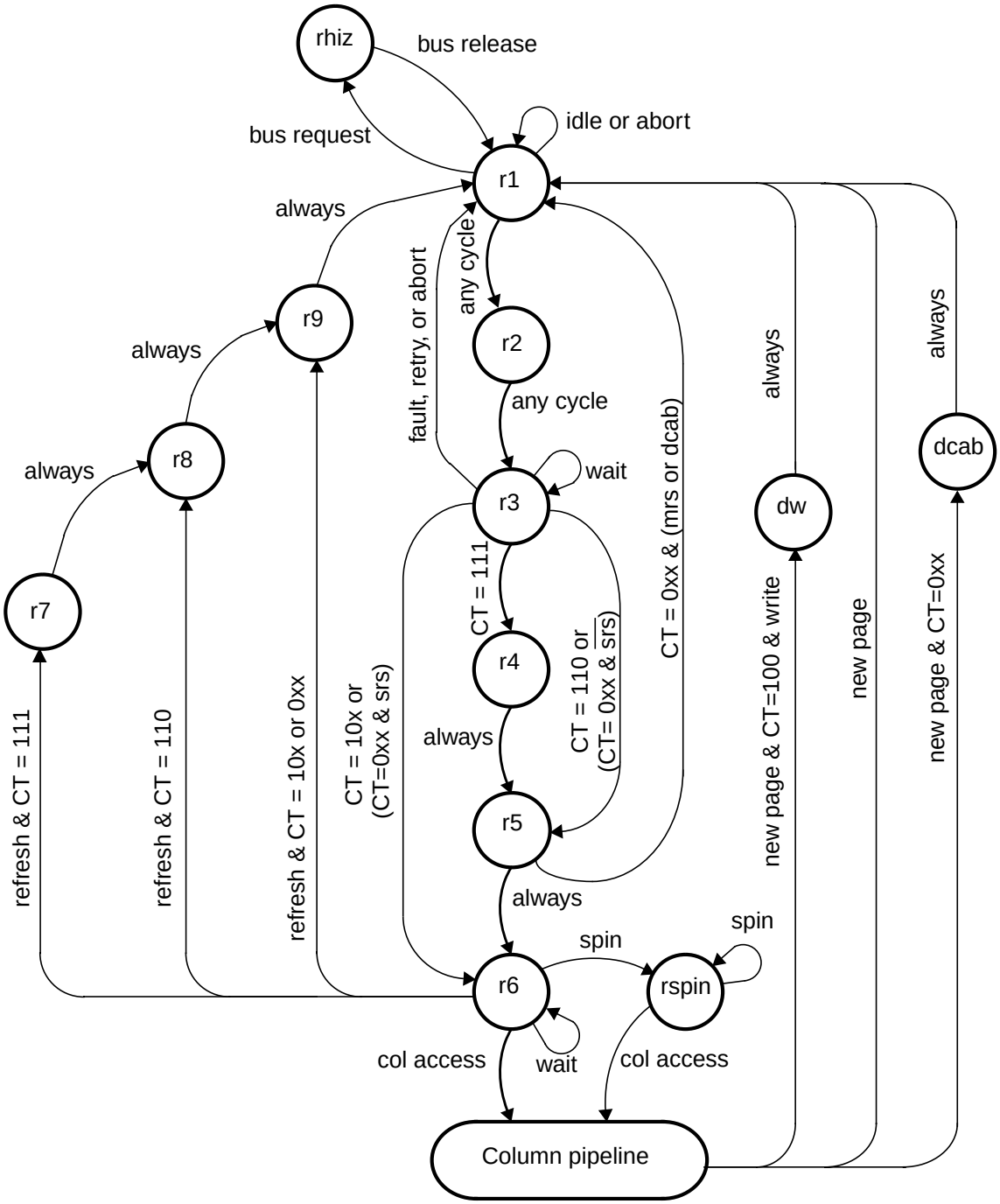
The TMS320C80 external-memory cycles are generated by the TC's external-memory controller. This controller contains a state machine that generates a sequence of states to control the transition of the memory-interface signals. The generated states and their sequences vary based on the type of cycle being performed, the column timing of the memory being accessed, the next access to be performed, and internal or external events (such as faults, etc.).

Figure 7–2 shows the state diagram for the external-memory controller. Although a large number of states and state transitions exist, their sequence is basically dependent on the column timing selected for the memory access being performed. The memory states can be broken into the following two groups:

- ❏ Row-time states
- ❏ Column-time pipeline

For a description of the conditions or events that cause transitions from one state to another, see subsection 7.2.1, *State-Transition Indicators*. The definitions of the row-time states and column-pipeline states depend on the type of memory being accessed. For standard DRAM, see section 7.3, *Standard DRAM Interface*. For synchronous DRAM (SDRAM), see section 7.4, *SDRAM Interface*.

Figure 7–2. External-Memory Interface State Diagram



7.2.1 State-Transition Indicators

The state-transition indicators determine the conditions or events that cause transitions to another state. In some cases, multiple conditions must occur in order to effect transitions to certain states.

State	Description
any cycle	Continuation of current cycle.
CT=xxx	State change occurs for indicated CT[2:0] value (as latched in r3 state).
abort	Current cycle aborted so that a higher priority cycle may occur.
fault	$\overline{\text{FAULT}}$ input sampled low (in r3 state), memory access faulted.
retry	$\overline{\text{RETRY}}$ input sampled low (in r3 state), row-time retry.
wait	READY input sampled low (in r3, r6, or final column states), repeat current state.
spin	Internally generated request adds additional state to allow the TC pipeline to load. Always occurs once during each two-cycles-per-column write, and twice during each one-cycle-per-column-write. Also occurs once during one-cycle-per-column accesses when the bus size has changed from the previous page access.
new page	The next access requires a page change (new row access).
srs	Special-register-set cycle, used for writing a value into SGRAM color registers. This special cycle is identified by STATUS[5:0] = 001101 during row time.
mrs	Mode-register-set cycle, performed following power-up refresh after the TMS320C80 is reset. This special cycle is identified by STATUS[5:0] = 001100 during row time.
dcab	Power-up deactivation, performed when an SDRAM CT code (CT=0xx) is input during one of the power-up refresh cycles that occurs after a hardware reset. This special cycle is identified by STATUS[5:0] = 000011 during row time.

7.3 Standard DRAM Interface

The DRAM interface comprises the row-time states and the column-time pipeline. The *row-time states* make up the address subcycle (or *row time*) of each memory access. The *column-time pipeline* makes up the data subcycle (or *column time*) for each memory access.

7.3.1 DRAM Row-Time States

Row-time states occur whenever a new page access begins. During these states, the TC determines the type of memory being addressed. A minimum of four row states occurs for every row access. Unless otherwise stated, input-sampling and output-signal transitions occur on the falling edge of the clock signal CLKOUT.

For a description of row-time states during SDRAM and SGRAM cycles, see subsection 7.4.7, *SDRAM Row-Time States*

For SRAM and standard DRAM, the states are defined as follows:

State	Description
r1	**Beginning state for all memory accesses.** Outputs row address (A[31:0]) and cycle type (STATUS[5:0]), and drives the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM[7:0] signals to their inactive levels. On the rising edge of CLKOUT, the TMS320C80 drives all other control signals to their inactive state. This results in $\overline{\text{DBEN}}$, DSF, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, and $\overline{\text{W}}$ being driven to their row-time levels. This state is repeated if idle at row time.
r2	**Common to all memory accesses.** Drives $\overline{\text{DDIN}}$ according to the data-transfer direction. Drives $\overline{\text{RL}}$ low on the rising edge of CLKOUT, (to indicate that a valid 32-bit address is on the address bus), and samples AS[2:0], BS[1:0], CT[2:0], PS[3:0], and $\overline{\text{UTIME}}$ inputs.
r3	**Common to all memory accesses.** Samples $\overline{\text{FAULT}}$, READY, and $\overline{\text{RETRY}}$ inputs. During DRAM refresh cycles, all $\overline{\text{CAS}}$/DQM[7:0] strobes are activated.
r4	**Inserted during three-cycles-per-column accesses only.** No signal transitions occur. The $\overline{\text{RETRY}}$ input is sampled.
r5	**Common to two- and three-cycles-per-column accesses.** $\overline{\text{RAS}}$ output is driven low and $\overline{\text{RETRY}}$ input is sampled.
r6	**Common to all memory accesses.** Drives $\overline{\text{RAS}}$ low (if not already) and samples the READY and $\overline{\text{RETRY}}$ inputs. Drives, on the rising edge of CLKOUT, $\overline{\text{DBEN}}$, DSF, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, and $\overline{\text{W}}$ to their appropriate (column-time) levels.
rspin	**Additional state that allows the TC pipeline to load.** No signal transitions occur. $\overline{\text{RETRY}}$ input is sampled. This state may be repeated multiple times.
r7	**Common to two-and three-cycles-per-column refreshes.** Processor activity code is output on STATUS[5:0].
r8	**Occurs for three-cycles-per-column refreshes only.** Processor activity code is output on STATUS[5:0].
r9	**Final state for all refreshes.** Processor activity code is output on STATUS[5:0].
rhiz	**High- impedance state.** Occurs during external bus (host) requests. Repeats until return of memory bus.
dw	**Occurs for pipelined one-cycle-per-column writes only.** This write drain state occurs prior to the r1 state after a pipelined one-cycle-per-column write. It activates all $\overline{\text{CAS}}$/DQM[7:0] strobes.

7.3.2 DRAM Column-Time Pipeline

The transfer of data, if any, occurs during column time. There are a number of different pipeline-flow sequences, depending on the selected column timing and whether the access is a read or a write. During page-mode operation, multiple column accesses occur, so the pipeline sequence may be repeated many times during the course of a single page-mode access. A general description of pipeline operation is discussed in section 7.5 and detailed descriptions for each type of memory cycle are provided in subsequent chapters.

Figure 7–3 shows the basic read-cycle column-access pipelines for all the standard DRAM column-timing selections. The pipelines are shown without wait states. Each example shows the pipeline sequence when three column addresses are read (Col A, Col B, and Col C) from a single page (row) of memory. The pipeline stages are shown in boxes (c1, c2, c3, and ci). Each box represents one clock cycle. Vertically aligned stages execute in parallel, in the same clock cycle. The events occurring in each stage depends on the column-timing selection as follows:

- *Nonpipelined one-cycle-per-column timing (a).* This is the fastest of the four selections. The memory device can accept a new address in one cycle. In the next clock cycle, the memory can output read-cycle data and simultaneously accept a new address.

- *Pipelined one-cycle-per-column timing (b).* As with the nonpipelined timing above, the memory device can accept a new address every clock cycle. However, the data bus is pipelined so that two clock cycles are required to clock data into the TMS320C80. Data is clocked onto the output of the memory device in two clock cycles (state c2 and c3) and input into the TMS320C80 in the latter clock cycle (state c3).

- *Nonpipelined two-cycles-per-column timing (c).* This selection is the most suitable when the memory device can accept a new address in one clock cycle (state c1) and output read-cycle data in the next (c2), but cannot accept a new address while there is a memory cycle in progress. With this selection, one clock cycle is also added to the row-address time.

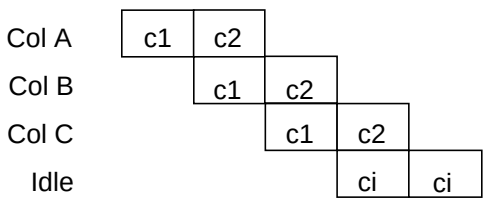
- *Nonpipelined three-cycles-per-column timing (d).* This selection is suitable for slower DRAM devices. The TMS320C80 uses two clock cycles to strobe the column address into the memory device: in the first clock cycle (c1), the memory device receives the column address; in the second clock cycle (c2), the address is strobed into the memory using the $\overline{\text{CAS}}$/DQM[7:0] signals. In the third clock cycle (c3), the memory drives

the read data into the TMS320C80. The memory device cannot accept a new address while there is a memory cycle in progress. With this selection, two clock cycles are added to the row-address time.

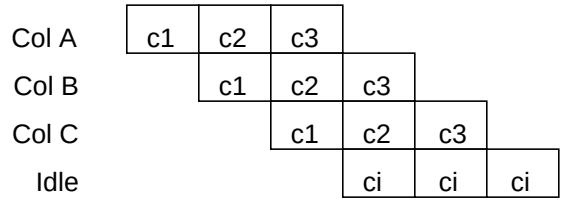
In all the above cases, the system bus must be configured so that data appearing on the memory output is driven into the TMS320C80 in the same clock cycle.

The idle stage (ci) occurs when no access is loaded into the pipeline.

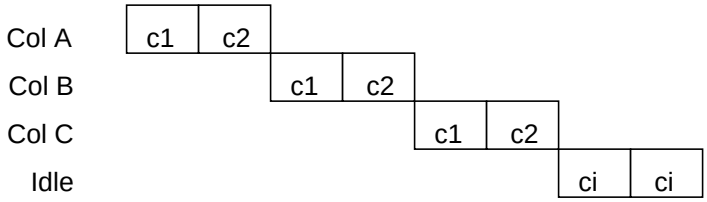
Figure 7–3. Basic Column-Time Pipelines for a Single-Page 3-Column Read Sequence



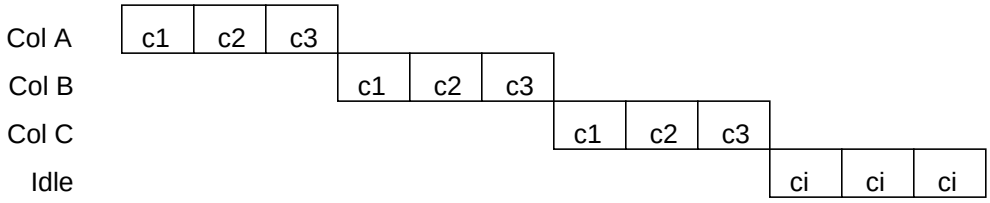
(a) Nonpipelined one-cycle-per-column read



(b) Pipelined one-cycle-per-column-read



(c) Nonpipelined two cycles-per-column read



(d) Nonpipelined three cycles-per-column-read

7.4 SDRAM Interface

The TMS320C80 provides direct interface support for synchronous DRAMs (SDRAMs) and synchronous graphics RAMs (SGRAMs). This provision allows the TMS320C80 more options for interfacing with memory that can be accessed in a single cycle.

7.4.1 Supported SDRAM Control Cycles

SDRAMs use various commands to control their operation and enable various features. The TMS320C80 implements the following SDRAM commands:

DCAB	Deactivate (also known as precharge) all banks
ACTV	Activate the selected bank and select the row
READ	Input the starting column address and begin the read operation
WRT	Input the starting column address and begin the write operation
MRS	Mode register set, configures the SDRAM mode register
REFR	Auto refresh cycle with internal address
SRS	Special register set (color register) - SGRAM only
BLW	Block write - SGRAM only

SDRAM commands not supported by the TMS320C80 are:

READ-P	Read with auto deactivate
WRT-P	Write with auto deactivate
SLFR	Self refresh
PDE	Power-down entry
STOP	Burst stop

7.4.2 Pin Mapping Between the TMS320C80 and SDRAM Devices

The SDRAM control pins map to the TMS320C80 pins as shown in Table 7–1.

Table 7–1. Mapping of SDRAM Control Pins to TMS320C80 Memory Control Pins

Pin Number	TMS320C80 Signal	SDRAM Signal
A21	$\overline{RAS}$	$\overline{RAS}$
B22	$\overline{TRG}/\overline{CAS}$	$\overline{CAS}$
C23	$\overline{W}$	$\overline{W}$
A13	$\overline{CAS}$/DQM[7]	DQM[7]
B14	$\overline{CAS}$/DQM[6]	DQM[6]
A15	$\overline{CAS}$/DQM[5]	DQM[5]
E17	$\overline{CAS}$/DQM[4]	DQM[4]
B16	$\overline{CAS}$/DQM[3]	DQM[3]
C19	$\overline{CAS}$/DQM[2]	DQM[2]
B20	$\overline{CAS}$/DQM[1]	DQM[1]
D20	$\overline{CAS}$/DQM[0]	DQM[0]
E25	CLKOUT	CLK
A23	DSF	DSF

Since the DQM pins serve essentially as byte strobes, their function is implemented on the TMS320C80's $\overline{CAS}$/DQM[7:0] outputs. This requires the remapping of the $\overline{CAS}$ function onto the TMS320C80's $\overline{TRG}/\overline{CAS}$ output. The SDRAM $\overline{CS}$ input must be implemented in external hardware. This is easily done using address and status decode. The CKE input is used primarily to place the SDRAM into self-refresh and power-down modes. The TMS320C80 does not support these modes, so no CKE signal is provided.

7.4.3 CAS Latency Options

CAS latency only applies to read cycles. The TMS320C80 supports SDRAMs with CAS latencies of two-to-four clock cycles. The input signals CT[2:0] select the latency at the beginning of the row-activate cycle. The three-cycle and four-cycle latencies allow use of lower-speed SDRAMs.

7.4.4 Burst Length Options

The TMS320C80 supports read and write burst lengths of one or two, depending on the type of SDRAM device being used. Because the TC has the ability to change column addresses on every cycle, a burst length of one is the most attractive. This allows the TC to access nonsequential columns within a row as may be required for packet transfers. However, certain SDRAM architectures (interleaved), require the column address to change only every other cycle. There is no advantage to a burst of one over a burst of two for these devices.

If the SDRAM uses a pipelined architecture that supports a new column address on every cycle, then the TC can be configured to use a burst length of one. Otherwise, the TC must be configured to use a burst length of two. The TC always uses the *serial* burst sequence. This allows either forward or reverse addressing depending on the starting column address. If a burst length of two is used and the second column access in the burst is not required, it is disabled via the DQM pin(s).

The burst length to be used is selected by the value input on the CT[2:0] pins at the beginning of the row-activate cycle.

7.4.5 SDRAM Bank Operation

Two rules apply to accessing a location in an SDRAM bank:

- ❏ A bank must be activated with a row address before it may be accessed.
- ❏ A bank must be deactivated (precharged) before it may be activated with a different row address.

The SDRAM access sequence is:

1) Activate bank.
2) Perform burst read/writes on row.
3) Repeat step 2 as needed.
4) Deactivate bank.
5) Go to step 1.

Because the TC's memory interface is DRAM-based, it follows a page-mode-access method. This page-mode operation has been adapted to the SDRAM operation by equating SDRAM commands with their page-mode

counterparts. Row activation corresponds to a TMS320C80 row access at the start of page. The burst reads and writes are similar to column-time accesses within the page. The bank deactivation is similar to the *drain* cycle that is performed at the end of pipelined writes prior to a page change.

7.4.6 External Logic Requirements Regarding the Address Bus

The transfer controller normally outputs the full 32-bit address at row time, and then outputs a shifted address at column time to provide the correct multiplexed address for the DRAM devices. The shift amount can be from 8 to 14 bits, depending on the value specified by the AS inputs. For SDRAMs, the shift mechanism behaves in an identical manner. Because SDRAMs use the address inputs for control as well as for addressing, this has the following system implications:

- ❏ The address line that corresponds to the SDRAM's Bank Select bit (A11 on 16-MBit SDRAM) must be latched externally by the falling edge of the $\overline{RL}$ signal. This ensures that the bank select remains correct during read and write commands.
- ❏ The system must force the address bit below the Bank Select bit (A10 on 16-MBit SDRAM) to be low unless $\overline{RAS}$ is active (low). This disables auto-precharge from occurring at the end of a read or write command.
- ❏ During DCAB cycles, the TMS320C80 outputs all 1s on A[31:0], to ensure that the A10 bit is 1 and both banks are deactivated.

7.4.7 SDRAM Row-Time States

The row-time states for the SDRAM are as follows:

State	Description
r1	**Beginning state for all memory accesses.** Outputs row address (A[31:0]) and cycle type (STATUS[5:0]), and drives the $\overline{RAS}$ and $\overline{CAS}$/DQM[7:0] signals to their inactive levels. For refresh cycles, the refresh pseudo-address appears on A[31:16] and the address lines A[15:0] are all driven low. For Mode Register Set cycles(MRS), the address on A[31:0] is invalid. On the rising edge of CLKOUT, the TMS320C80 drives all other control signals except $\overline{DDIN}$ to their inactive state. This results in $\overline{DBEN}$, DSF, $\overline{TRG}$/$\overline{CAS}$, and $\overline{W}$ being driven to their row-time levels.

State	Description
r2	**Common to all memory accesses.** Drives the transceiver-control signal $\overline{\text{DDIN}}$ according to the data transfer direction. On the rising edge of CLKOUT, the TMS320C80 drives $\overline{\text{RL}}$ low (to indicate that a valid 32-bit address is on the address bus), and samples AS[2:0], BS[1:0], CT[2:0], PS[3:0], and $\overline{\text{UTIME}}$ inputs.
r3	**Common to all memory accesses.** Samples $\overline{\text{FAULT}}$, READY, and $\overline{\text{RETRY}}$ inputs.
r5	**Common to all memory accesses except SGRAM SRS Cycles.** For normal read/write cycles, this state corresponds to the bank-activation cycle where the $\overline{\text{RAS}}$ output is driven low. The $\overline{\text{RETRY}}$ input is sampled. For refresh cycles, the refresh command is generated in this cycle by setting both $\overline{\text{RAS}}$ and $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ low. For an MRS cycle, this is the final state in the cycle; $\overline{\text{RAS}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, and $\overline{\text{W}}$ are set low, and the MRS value is output on A[11:0].
r6	**Common to all memory accesses except Mode-Register-Set cycles.** Drives $\overline{\text{RAS}}$ inactive (high) and samples the READY and $\overline{\text{RETRY}}$ inputs. For refresh cycles, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ also goes inactive. On the rising edge of CLKOUT, the TMS320C80 drives $\overline{\text{DBEN}}$ and $\overline{\text{W}}$ to their appropriate (column-time) levels.
rspin	**Additional state that allows the TC pipeline to load.** No signal transitions occur. $\overline{\text{RETRY}}$ input is sampled. This state may be repeated multiple times.
r9	**Final state for refreshes.** Processor activity code is output on STATUS[5:0]. $\overline{\text{RETRY}}$ input is sampled.
rhiz	**High-impedance state.** Occurs during external bus (host) requests. Repeats until return of memory bus.
dcab	**SDRAM bank deactivate cycle.** Occurs when the next access is on a new page from the access already loaded in the column pipe.

7.4.8 SDRAM Column-Time Pipeline

The column-time pipeline makes up the data subcycle (or *column time***)** for each memory access. The transfer of data, if any, occurs during these states. There are a number of different pipeline-flow sequences, depending on the selected column timing and whether the access is a read or a write. During page-mode operation, multiple column accesses occur, so the pipeline sequence may be repeated many times during the course of a single page-mode access. section 7.5 provides a general description of the pipeline operation, and detailed descriptions for each type of memory cycle are provided in subsequent chapters.

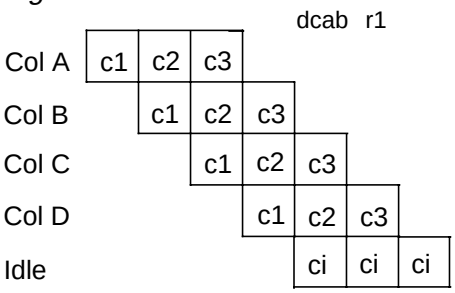
Figure 7–4 shows the basic column-access pipelines for all the SDRAM burst sequences.

The examples show the pipeline sequence when four column addresses (Col A, Col B, Col C, and Col D) are read from or written to a single page (row) of memory. An even number of column accesses was chosen for these examples because for sequences with a burst length of two, there is always an even number of accesses on a page.

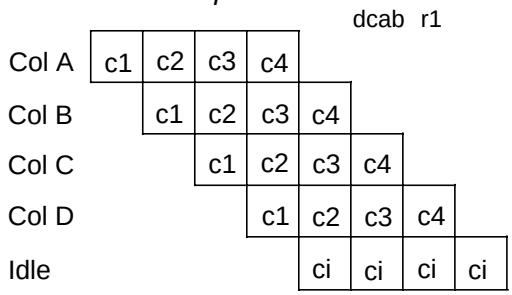
For accesses of burst-length two, the address of column B is the address of column A plus or minus the column width in bytes. Similarly, the address of column D is the column-C address plus or minus the column width. If the TC does not require the data in column B or column D, it sets the DQM lines high for those accesses.

The pipeline stages are shown in Figure 7–4 as boxes labelled c1, c2, c3, c4, and ci. Each box represents one clock cycle. Vertically aligned stages execute in parallel, in the same clock cycle. State *dcab* represents a bank deactivate (DCAB) cycle, and is performed whenever the next data access is on a new page. This cycle occurs after loading the column pipeline for the current access. The TMS320C80 does not wait for the column pipeline to empty to perform the DCAB cycle. For burst-read sequences, one or more idle states (ci) may be inserted into the column pipeline to ensure that the next r1 state, immediately following DCAB, is concurrent with the last stage (c3 for CAS latency 2; c4 for CAS latency 3) of the last column transfer (column D in Figure 7–4).

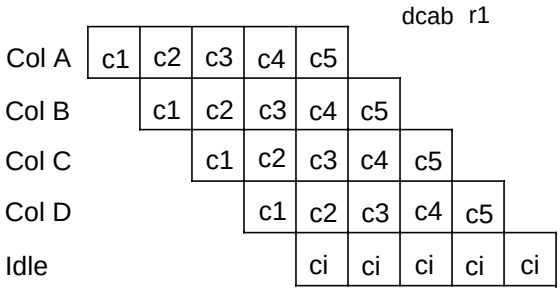
Figure 7–4. Basic Column-Time Pipelines for SDRAM Burst Sequences



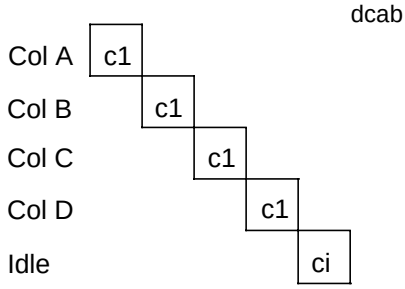
(a) Burst read, CAS latency 2



(b) Burst read, CAS latency 3



(c) Burst read, CAS latency 4



(d) Burst write

7.5 Transfer Controller Pipelines

The TC contains pipelines in both its internal- and external-memory interfaces. These pipelines can queue up the memory accesses required by the TC, if the current access has not yet completed. For example, if a packet transfer is transferring data from on-chip to off-chip memory, the dst may require two cycles per access. Because data can be extracted from the FIFO at one cycle per access (assuming the required data is in the FIFO), another dst cycle can be placed in the pipeline before the first cycle is completed.

The pipeline is completely transparent to the user. Its effect on operation can only be seen when the pipeline is drained. In order for a cycle to be loaded into the external-memory pipeline, it must be located within the same memory page as any other cycles already contained in the pipeline. Once a cycle has been placed in the pipeline, it cannot be removed; the cycle must occur. When an access to a new memory page is requested, the cycles in the pipeline must first be completed. The last clock cycle of an SDRAM read cycle may be concurrent with the first clock cycle of the access on the new memory page.

If, for example, the TC is performing a packet transfer to external memory and the VC issues a memory request, any (column) cycles currently in the pipeline must be completed before the cycle requested by the video controller can occur. (This is true even though the VC cycle has higher priority). Pipeline draining can occur prior to cache/DMA accesses, host accesses, and urgent refreshes, during the suspension of packet transfers, and after column-time retries. As mentioned in section 2.2, *Effect of Transfer Controller Requests on Crossbar Priority*, pipeline draining takes highest priority when an urgent request is pending.

7.6 Wait States

The TMS320C80 allows the insertion of wait states to extend memory-cycle times by using the READY input. The READY input is sampled at appropriate times on the falling edge of CLKOUT. If READY is sampled high, the cycle continues. If READY is sampled low, the current machine state is repeated and READY is sampled again on the next CLKOUT falling edge. The memory cycle continues to stall on (repeat) the current state until the TC samples READY high.

Note:
The TC has no time-out or abort mechanism to terminate a memory access that is being stalled by a large number of wait states. Memory accesses that cannot be completed in a reasonable time should be faulted or retried to prevent locking out high-priority accesses to external memory that may be waiting to be serviced.

There are two kinds of wait states that can be inserted for a memory access:

- ❏ Row-time wait states
- ❏ Column-time wait states

7.6.1 Row-Time Wait States

Row-time wait states can be inserted at two locations during row time; before and after the fall of $\overline{\text{RAS}}$. READY is first sampled by the TC at the start of the r3 state. This allows time to decode the row address and/or cycle type and determine if the addressed device needs additional access time prior to the fall of $\overline{\text{RAS}}$. The r3 state is repeated until the TC samples READY high.

The READY input is sampled again during the r6 state. This occurs after the fall of $\overline{\text{RAS}}$ and allows the creation of additional RAS access time for devices which may need it. If READY is sampled low, the r6 state is repeated until READY is sampled high again.

Note:
Additional *rspin* states are automatically inserted into the cycle by the TC during one-and two-cycle-per-column writes, and one-cycle-per-column accesses that change the bus size. Any extra *r6* states resulting from READY being sampled low, are inserted prior to the rspin states. READY is not sampled during rspin.

7.6.2 Column-Time Wait States

Column-time wait states are not supported for single-cycle-per-column or SDRAM accesses, so READY is not sampled beyond the r6 state for these cycles. For two- and three-cycle-per-column accesses, the READY pin is sampled during the c2 or c3 state. This occurs after the column address and fall of $\overline{\text{CAS}}$, and allows extended CAS access time for devices that require it. If READY is sampled high, the stage completes the column access. Otherwise, the stage is repeated, the column pipeline stalls, and the TC samples READY on each subsequent CLKOUT falling edge until it is sampled high.

7.7 Ending an External-Memory Cycle

An external-memory cycle can be terminated by normal termination, retry, a new page request, or faulting.

7.7.1 Normal Termination

When the TC has completed all the pending column accesses in its pipeline, it is ready to terminate the memory access. Termination does not occur, however, until a new row access is required. The external-memory signals remain active (in the ci state) until the next memory access. This allows DRAM devices to remain in a page-mode state and SDRAM devices to maintain their active row. If the next address falls within the same memory page as the previous memory access (and has the same row-time status), then no row-access cycle is needed. If the next memory access requires a row access, then the current page mode access is terminated and the new row access begins.

Even if there is little other external bus activity, most memory cycles are terminated soon after completion of their last column access by the occurrence of trickle-refresh cycles, which require a new row access to occur.

7.7.2 Retry

A retry is a mechanism by which external logic notifies the TMS320C80 that the current access cannot be completed and should be retried. Retries are generated at row time by driving the $\overline{\text{RETRY}}$ input low when it is first sampled at the beginning of the r3 state. When this happens, the TC terminates the current access at the end of the r3 state. The retried access then restarts immediately with a new r1 state unless a higher priority request is pending. In

the latter case, the retried cycle won't be restarted until *after* the higher priority request has been serviced.

If a retry occurs during a packet-transfer cycle and a packet-transfer request of equal or higher priority is pending, then the TC suspends the packet transfer in which the retry occurred. This assumes that PTMIN has been satisfied. The retry does not occur until the packet transfer is resumed, when it reaches its turn in the priority/round-robin chain again.

7.7.3 New Page Request

Because column accesses are already loaded in the pipeline, they cannot be retried. However, the TC does support a page-request mechanism at column time.

If a page request occurs, the TC completes all pending column accesses in its pipeline and begins the next access with a row access. A number of column accesses can occur after the new page has been requested. These column access *are not* repeated after the row access has restarted.

Page requests occur any time that the $\overline{\text{RETRY}}$ input is sampled low after the r3 state. The TC samples $\overline{\text{RETRY}}$ at the end of each state following r3 (on each CLKOUT falling edge) and must be at a valid high or low level during each sample period. If the TC samples $\overline{\text{RETRY}}$ low, then the current page mode terminates as soon as all column accesses currently in the TC pipeline are completed. Once the new page has been requested, the value on the $\overline{\text{RETRY}}$ input has no further effect. However, $\overline{\text{RETRY}}$ continues to be sampled by the TC during each remaining column access and must be maintained at a valid high or low level during each sample period. Normal operation is for the system to drive and maintain $\overline{\text{RETRY}}$ at a low level until the end of the current row access.

Because of the way in which the TC's external interface pipeline is loaded, asserting $\overline{\text{RETRY}}$ in states r4 or r5 during read cycles has no effect. The $\overline{\text{RETRY}}$ input is still sampled and must be at a valid high or low level. For this reason, the practice of asserting $\overline{\text{RETRY}}$ low until the end of the row access is recommended if a new page request is desired.

SRT cycles and refresh cycles are not affected by new page requests because a new row access always follows their single-column access.

7.7.4 Faulting

If a system error prevents a memory access from being completed, the system can notify the TMS320C80 by faulting the memory cycle. This allows the MP to correct the error before the memory access is retried. Memory faults can be generated at row time only and are initiated by driving the $\overline{\text{FAULT}}$ input low at the start of the r3 state. $\overline{\text{FAULT}}$ is not sampled by the TC during any other part of the memory cycle. The faulting mechanism varies somewhat depending on the type of access when the fault occurred. Fault support for different types of cycle requests are defined below.

- *SRT cycles*—Faulting not supported. $\overline{\text{FAULT}}$ pin is ignored.
- *Refresh cycles*—Faulting not supported. $\overline{\text{FAULT}}$ pin is ignored.
- *PP cache/DEA request*—Faulting supported. Request is not completed until fault is cleared. Requests from other PPs can continue to be serviced.
- *MP instruction/data cache request*—Faulting supported. The faulted cache request is immediately canceled and the MP interrupted. The other cache can still have its pending request serviced.
- *MP DEA request*—Faulting supported. The DEA request is immediately canceled and the MP is interrupted.
- *MP/PP packet transfer*—Faulting supported. The packet transfer is suspended and its state is saved to the requesting processor's parameter RAM. Packet transfer requests from other processors can still be serviced.
- *XPT/VCPT*—Faulting supported. The packet transfer terminates immediately but no suspension occurs. Further XPTs are blocked until the fault is cleared.

7.7.4.1 Packet-Transfer Faults

When a fault occurs during an MP or PP packet transfer, the transfer is suspended and its state is included in the parameters. If the transfer is from off-chip to off-chip memory, the state of the off-chip-to-off-chip packet-transfer buffer is also saved. The buffer itself is not modified. The transfer status bits in the saved transfer-options field are set to show whether the fault occurred on the src or dst transfer (see section 3.12, *Packet-Transfer Suspension*).

Once the parameters have been saved, the TC sets the appropriate bit in the FLTS register, indicating which processor's transfer was faulted. Setting this bit also sets the mf bit of the MP's INTPEN register and generates a fault interrupt to the MP. The MP can read the FLTSTS register to find out which processor was faulted. When the processor is identified, the MP can examine the suspended transfer's parameters to determine the memory access that caused the fault.

Note:
A PP does not know when a fault occurs during its packet transfer. It only knows that its transfer has not yet completed. The MP must correct the fault or signal the PP to cancel its packet transfer request.

If the MP can correct the fault, it clears the bit in the FLTSTS register and the packet-transfer request is automatically resubmitted. Once the faulted transfer gets its turn in the round-robin/priority scheme, its internal state is restored from the saved parameter RAM and the TC continues the packet transfer at the faulted access.

When a fault occurs during an XPT or VCPT, the transfer is immediately terminated. The transfer is *not* suspended, so its state is not saved. The TC sets the appropriate XPT bits in FLTSTS, and the MP is interrupted in the normal manner. Further XPTs and VCPTs are blocked until the fault is cleared by the MP. This prevents the system lockup caused by the faulted XPT request being maintained on the XPT[2:0] inputs.

7.7.4.2 PP Cache/DEA Faults

If a fault occurs during a PP-requested cache service or DEA request, the address where the fault occurred is saved in the cache-fault-address location in the requesting processor's parameter RAM. (The layout of the parameter RAM is shown in Figure 3–2). The appropriate bit in the FLTSTS register is set, causing an memory-fault interrupt to the MP. The MP then examines the parameter RAM to determine the faulted address.

If the MP can correct the fault, it clears the bit in the FLTSTS register and the request is rescheduled. If the fault cannot be corrected, the MP can send the PP a reset request so that the cache-service request will be aborted.

7.7.4.3 MP Cache/DEA Faults

If an MP requested cache-fill or DEA cycle faults, then the request is immediately canceled and the TC sends the MP a memory-fault interrupt. The appropriate bit is set in the FLTOP register (FLTOP [d] for data-cache fault and FLTOP [i] for instruction-cache fault) to indicate the type of faulted access. If a data-cache fault occurs, the address is written to the FLTADR register, the data is written to the FLTDTL and FLTDTH registers, and additional cycle information is placed in the FLTOP and FLTTAG registers. See section 3.8, *Memory Fault Registers*, in the *TMS320C80 Master Processor User's Guide* for more information.

7.7.4.4 On-Chip Faults

Certain accesses to on-chip addresses can cause faults that are independent of the $\overline{\text{FAULT}}$ input. These occur when an illegal on-chip access is attempted. The normal fault mechanism for the cycle being attempted applies. The on-chip faults include:

- A PP packet transfer to/from any address under 0 x 02000000 that is not a data RAM or PP parameter RAM.
- A PP or MP instruction cache service to/from any address under 0 x 02000000 that is not a data RAM or PP parameter RAM.
- A PP DEA to/from any address under 0x02000000 that is not a data RAM or PP parameter RAM.
- An MP DEA to/from any address under 0x02000000 that is not a data RAM, parameter RAM, on-chip register, or MP data cache.
- An MP packet transfer to/from any address under 0 x 02000000 that is not a data RAM or parameter RAM.

7.8 User-Modified Timing

You can use the $\overline{\text{UTIME}}$ input to generate memory timings that are different from those provided by the TMS320C80. If $\overline{\text{UTIME}}$ is sampled low at row time, then the timings of the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM[7:0] outputs are modified for the current page. The $\overline{\text{RAS}}$ signal is modified to indicate when column accesses are begun. User-timed $\overline{\text{CAS}}$/DQM signals can be triggered by $\overline{\text{RAS}}$ falling. $\overline{\text{RAS}}$ is only asserted (active low for one machine state) when an actual column access is begun so that any bubbles in the TC pipeline can be detected.

$\overline{\text{CAS}}$/DQM[7:0] are output at the same time as the column address. This enables them to provide an earlier indication of which bytes are being accessed, making external $\overline{\text{CAS}}$ generation easier.

Because the timing of $\overline{\text{RAS}}$ is modified, external logic must generate its own $\overline{\text{RAS}}$ timings (if required). The externally generated $\overline{\text{RAS}}$ can be triggered by the falling edge of $\overline{\text{RL}}$. The status code output at the beginning of the cycle provides all the information necessary to generate the memory timings (for $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, $\overline{\text{W}}$, etc.) for the current cycle. The timing of these outputs is not modified during user-timed accesses.

Chapter 8

Big- and Little-Endian Formats

The TC accesses data in either big-endian or little-endian format. The endian mode selects the way in which bytes are addressed. In little-endian, byte 0 is the rightmost byte in a word and successive bytes are numbered leftward. In big-endian format, byte 0 is the leftmost byte in a word and successive bytes are numbered rightward.

Topic **Page**

8.1 Selecting Big- or Little-Endian Format

The endian format for the TMS320C80 is selected at reset using the $\overline{\text{UTIME}}$ input. The TMS320C80 samples and latches the value of $\overline{\text{UTIME}}$ on the clock cycle before the rising edge of the $\overline{\text{RESET}}$ input. If $\overline{\text{UTIME}}$ was sampled low (0) at the end of reset, the TMS320C80 operates in big-endian mode until the next hardware reset occurs. If $\overline{\text{UTIME}}$ was sampled high, the TMS320C80 operates in little-endian format.

8.2 Big- or Little-Endian Byte Ordering and Bus Size

The three LSBs of the address and the number of bytes to be transferred determine the positions of valid data bytes. Table 8–3 and Table 8–4 show byte positions for 64-bit bus transfers. The ● symbol represents the valid byte positions, and the ❍ symbol represents the invalid bytes. Dashes indicate that the operation cannot be performed.

When the external bus is limited to 32 bits, only bytes 0-3 of the bus are used for transferring data. This means that D[31:0] are used for little-endian transfers and D[63:32] are used for big-endian transfers. The byte positions, based on the two LSBs of the address for little and big endian, are shown in Table 8–3 and Table 8–4, respectively. The **x** indicates bytes of the 64-bit bus that are ignored. Dashes indicate transfers that cannot be performed.

When the external bus is limited to 16 bits, only bytes 0 and 1 (D[15:0] for little endian or D[63:48] for big endian) are used to transfer data. Table 8–5 and Table 8–6 show the byte positions based on the LSB of the address. The **x** indicate bytes of the 64-bit bus that are ignored and dashes indicate transfers that cannot be performed.

When the external bus is configured for 8 bits, only byte 0 (D[7:0] for little endian or D[63:56] for big endian) is used to transfer data. Table 8–7 and Table 8–8 show the byte positions for 8-bit bus size.

Table 8–1. Little-Endian 64-Bit Bus Byte Positions

Three LSBs of Address	Number of Bytes Being Transferred							
	1 Byte	**2 Bytes**	**3 Bytes**	**4 Bytes**	**5 Bytes**	**6 Bytes**	**7 Bytes**	**8 Bytes**
000	○○○○○○○●	○○○○○○●●	○○○○○●●●	○○○○●●●●	○○○●●●●●	○○●●●●●●	○●●●●●●●	●●●●●●●●
001	○○○○○○●○	○○○○○●●○	○○○○●●●○	○○○●●●●○	○○●●●●●○	○●●●●●●○	●●●●●●●○	-
010	○○○○○●○○	○○○○●●○○	○○○●●●○○	○○●●●●○○	○●●●●●○○	●●●●●●○○	-	-
011	○○○○●○○○	○○○●●○○○	○○●●●○○○	○●●●●○○○	●●●●●○○○	-	-	-
100	○○○●○○○○	○○●●○○○○	○●●●○○○○	●●●●○○○○	-	-	-	-
101	○○●○○○○○	○●●○○○○○	●●●○○○○○	-	-	-	-	-
110	○●○○○○○○	●●○○○○○○	-	-	-	-	-	-
111	●○○○○○○○	-	-	-	-	-	-	-

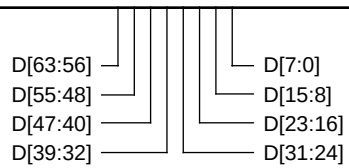
D[63:56] D[55:48] D[47:40] D[39:32] — D[7:0] D[15:8] D[23:16] D[31:24]

● Valid byte positions

○ Invalid byte positions

Table 8–2. Big-Endian 64-Bit Bus Byte Positions

Three LSBs of Address	Number of Bytes Being Transferred							
	1 Byte	2 Bytes	3 Bytes	4 Bytes	5 Bytes	6 Bytes	7 Bytes	8 Bytes
000	●○○○○○○○	●●○○○○○○	●●●○○○○○	●●●●○○○○	●●●●●○○○	●●●●●●○○	●●●●●●●○	●●●●●●●●
001	○●○○○○○○	○●●○○○○○	○●●●○○○○	○●●●●○○○	○●●●●●○○	○●●●●●●○	○●●●●●●●	-
010	○○●○○○○○	○○●●○○○○	○○●●●○○○	○○●●●●○○	○○●●●●●○	○○●●●●●●	-	-
011	○○○●○○○○	○○○●●○○○	○○○●●●○○	○○○●●●●○	○○○●●●●●	-	-	-
100	○○○○●○○○	○○○○●●○○	○○○○●●●○	○○○○●●●●	-	-	-	-
101	○○○○○●○○	○○○○○●●○	○○○○○●●●	-	-	-	-	-
110	○○○○○○●○	○○○○○○●●	-	-	-	-	-	-
111	○○○○○○○●	-	-	-	-	-	-	-



● Valid byte positions

○ Invalid byte positions

Table 8–3. Little-Endian 32-Bit Bus Byte Positions

Two LSBs of Address	Number of Bytes Being Transferred			
	1 Byte	2 Bytes	3 Bytes	4 Bytes
00	xxxx○○○●	xxxx○○●●	xxxx○●●●	xxxx●●●●
01	xxxx○○●○	xxxx○●●○	xxxx●●●○	-
10	xxxx○●○○	xxxx●●○○	-	-
11	xxxx●○○○	-	-	-

● Valid byte positions

○ Invalid byte positions

x Ignored byte positions

Table 8–4. Big-Endian 32-Bit Bus Byte Positions

Two LSBs of Address	Number of Bytes Being Transferred			
	1 Byte	2 Bytes	3 Bytes	4 Bytes
00	●○○○xxxx	●●○○xxxx	●●●○xxxx	●●●●xxxx
01	○●○○xxxx	○●●○xxxx	○●●●xxxx	-
10	○○●○xxxx	○○●●xxxx	-	-
11	○○○●xxxx	-	-	-

Table 8–5. Little-Endian 16-Bit Bus Byte Positions

LSB of Address	Number of Bytes Being Transferred	
	1 Byte	2 Bytes
0	xxxxxx○●	xxxxxx●●
1	xxxxxx●○	-

Table 8–6. Big-Endian 16-Bit Bus Byte Positions

LSB of Address	Number of Bytes Being Transferred	
	1 Byte	2 Bytes
0	●○xxxxxx	●●xxxxxx
1	○●xxxxxx	-

Table 8–7. Little-Endian 8-Bit Bus Byte Positions

LSB of Address	Number of Bytes Being Transferred	
	1 Byte	2 Bytes
x	xxxxxx●	-

Table 8–8. Big-Endian 8-Bit Bus Byte Positions

LSB of Address	Number of Bytes Being Transferred	
	1 Byte	2 Bytes
x	●xxxxxx	-

8.3 Dynamic Bus Sizing

The portion of the bus that transfers valid data during a read or write cycle depends on the bus size and the endian mode in operation. The TC can support 8-, 16-, 32-, and 64-bit data buses. Bus size also determines the number of column accesses required to transfer a given number of bytes.

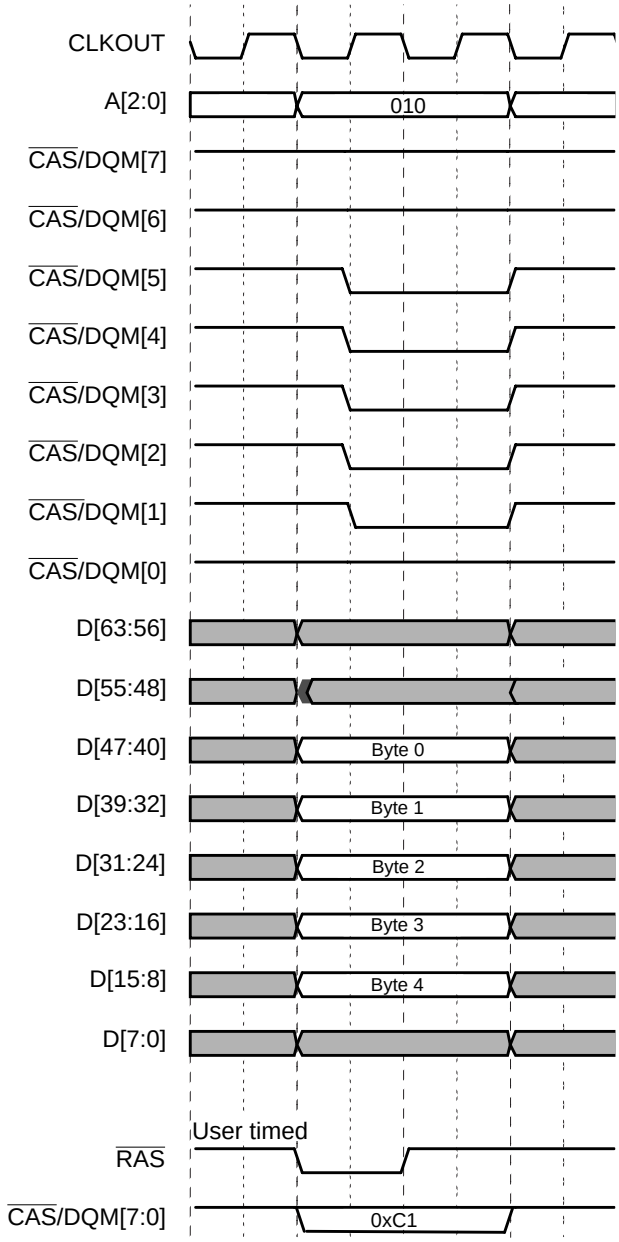
The following sections demonstrate possible column accesses when writing data to external memory. A common example shows a write of five bytes of data using a 64-bit bus, then a 32-bit bus, a 16-bit bus, and an 8-bit bus. The example is shown first with big-endian ordering and then with the corresponding little-endian ordering. The starting address of these five bytes is 0x02xxxxx2 (that is, starting with byte 2 of a 64-bit doubleword in external memory). This type of transfer might occur as part of a packet-transfer operation.

Each example shows the byte address as output on A[2:0], the active $\overline{\text{CAS}}$/DQM strobes, and the valid portions of the data bus. The two-cycles-per-column timing is the only timing shown. However, the order and addresses of the column accesses are identical for other timings as well.

8.4 Byte Ordering Example for 64-Bit Big-Endian Transfer

For a 64-bit data bus, only one column access is required because the data size is less than 64 bits. The memory cycle accesses the 64-bit doubleword at address 0xxx (xxx represents the byte address within the doubleword). Figure 8-1 shows a write of five bytes starting at address 0x02xxxxx2. Because the write begins at address 0b0010, the five data bytes are placed on the data bus beginning with D[47:40]. $\overline{\text{CAS}}$/DQM[5:1] are active so that only the valid bytes are written. A read access behaves identically to the way a write access behaves, except that all the $\overline{\text{CAS}}$/DQM strobes are active and the invalid bytes are discarded by the TC.

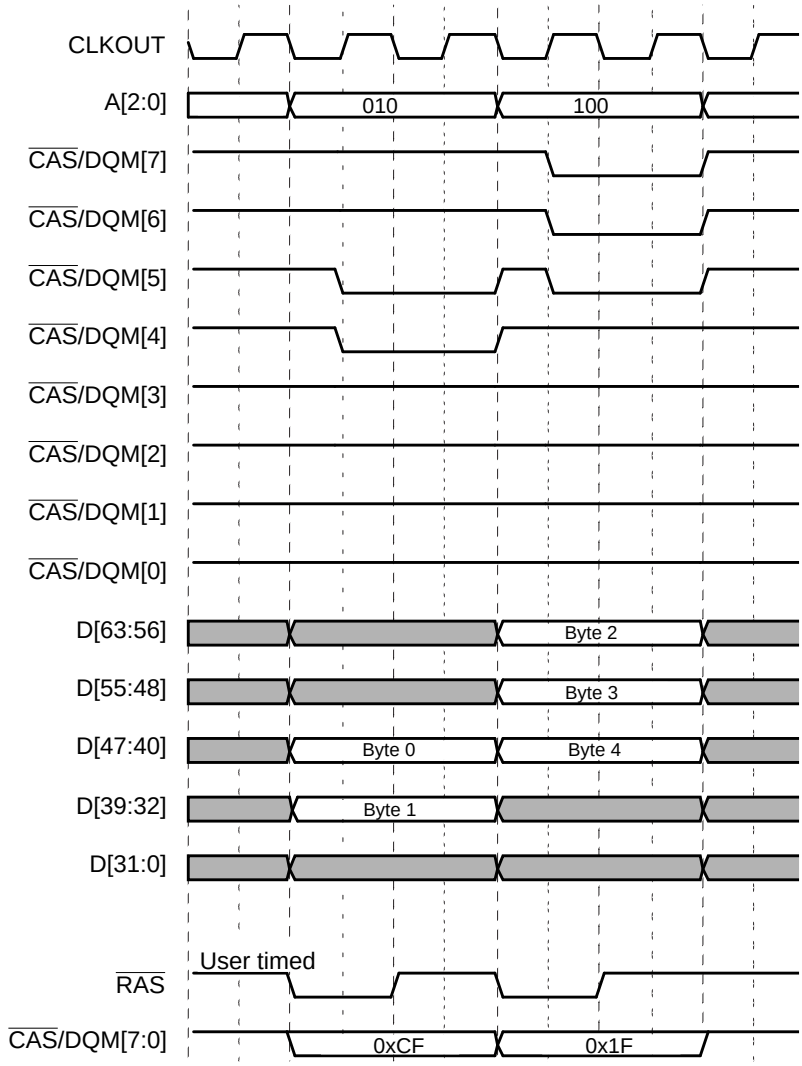
Figure 8–1. 64-Bit Big-Endian Transfer



8.5 Byte Ordering Example for 32-Bit Big-Endian Transfer

Figure 8–2 shows a big-endian transfer of five bytes beginning at address 0x02xx xxx2 on a 32-bit bus. Because the write crosses a 32-bit word boundary, two column accesses are required. The first access is to word 00xx. Only the two MSbytes are written, so valid data is placed on D[47:32] and $\overline{\text{CAS}}$/DQM[5:4] are activated. The second access to word 01xx writes the three final bytes. These are written to the LSbytes of the word so that D[63:40] contain valid data and $\overline{\text{CAS}}$/DQM[7:5] are active. For a read access of the same bytes, the same column cycles occur but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes are discarded by the TC.

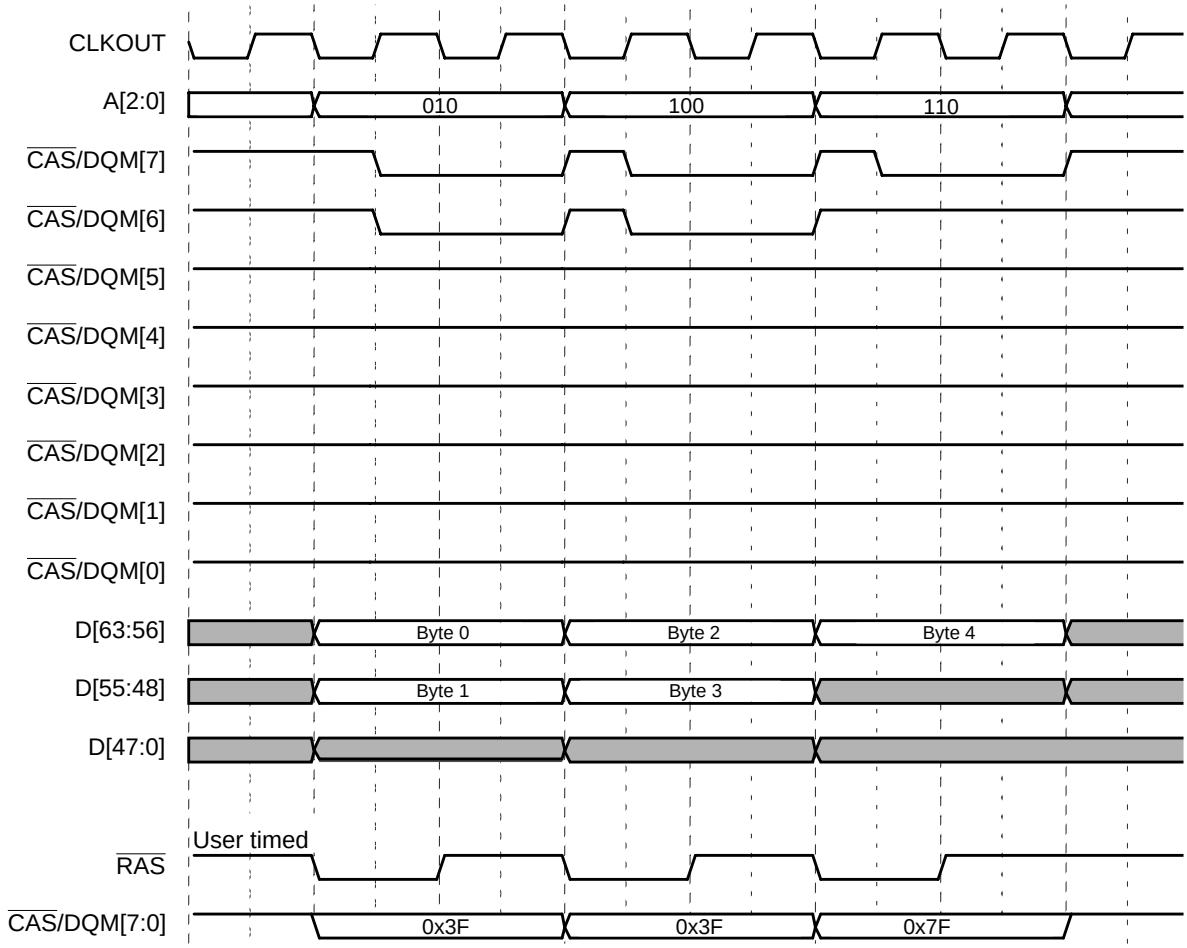
Figure 8–2. 32-Bit Big-Endian Transfer



8.6 Byte Ordering Example for 16-Bit Big-Endian Transfer

Figure 8–3 shows a big-endian transfer of five bytes beginning at address 0x02xx xxx2 on a 16-bit bus. The transfer requires three column accesses. The first access is to halfword 001x. (An access to halfword 000x is not required because none of its bytes are being written.) Both bytes in this word are written so that D[63:48] are valid, and both $\overline{\text{CAS}}$/DQM strobes ($\overline{\text{CAS}}$/DQM[7] and $\overline{\text{CAS}}$/DQM[6]) are active. The second column accesses halfword 010x and writes to both of its bytes. The final access is to halfword 011x. One byte is transferred on D[63:56] so that only $\overline{\text{CAS}}$/DQM[7] is active. For a read access of the same bytes, the identical column cycles occur, but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes are discarded by the TC.

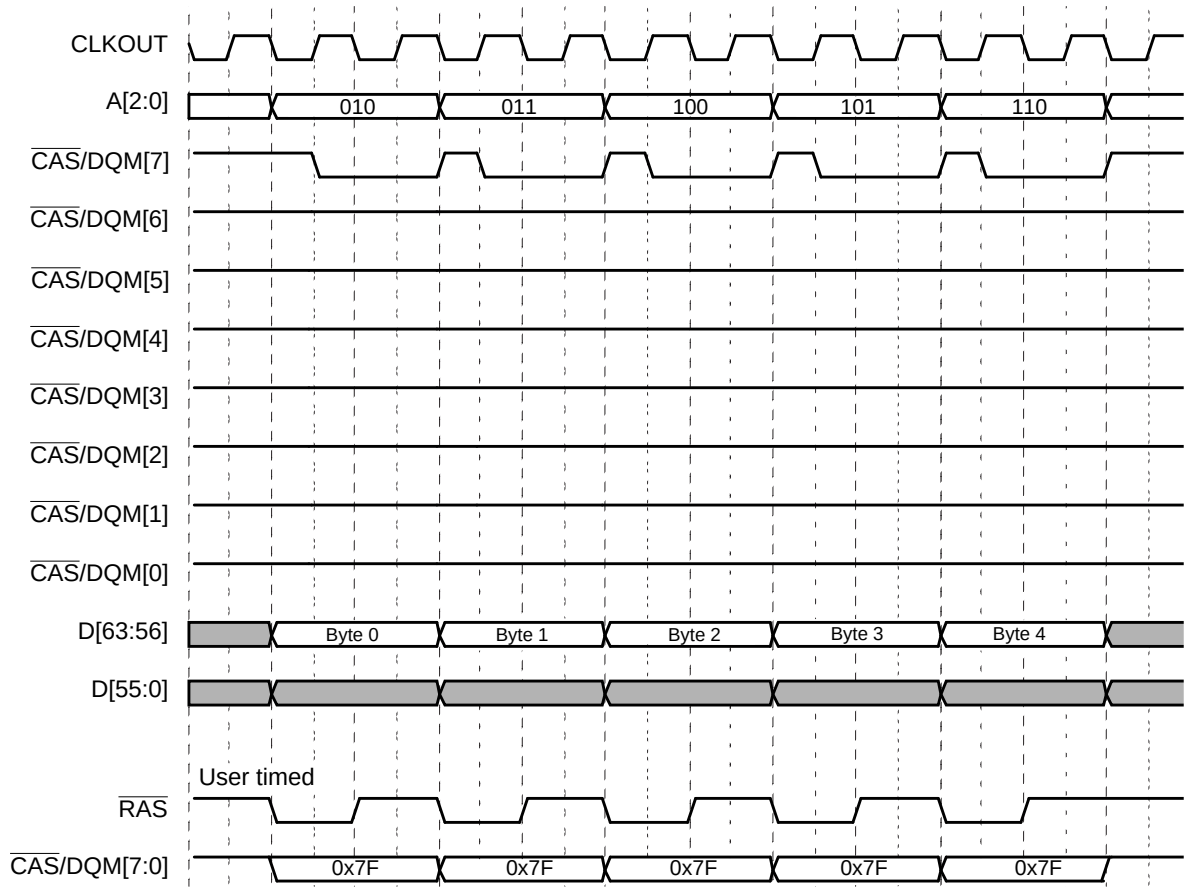
Figure 8–3. 16-Bit Big-Endian Transfer



8.7 Byte Ordering Example for 8-Bit Big-Endian Transfers

Figure 8–4 shows a big-endian transfer of five bytes beginning at address 0x02xx xxx2 on an 8-bit bus. For this bus size, each column access transfers a single byte on D[63:56]. Thus, a total of five column accesses are required. The bytes accessed are 0010, 0011, 0100, 0101, and 0110. $\overline{\text{CAS}}$/DQM[7] is active for each column access. For a read access of the same five bytes, the identical column accesses occur, but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes on D[55:0] are discarded by the TC.

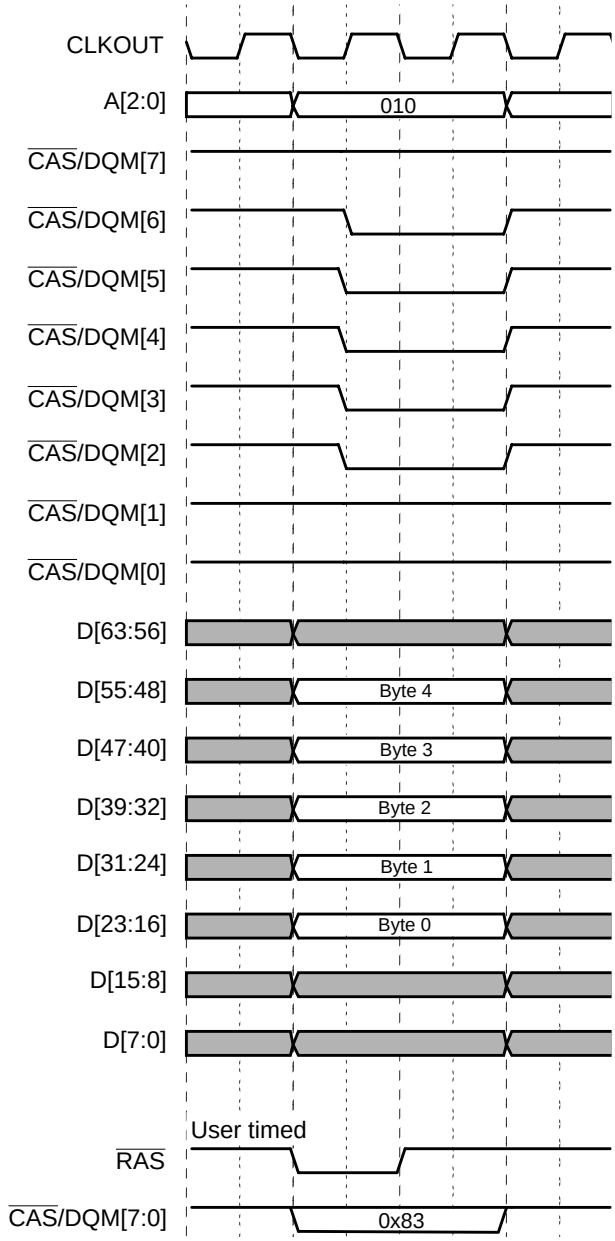
Figure 8–4. 8-Bit Big-Endian Transfer



8.8 Byte Ordering Example for 64-Bit Little-Endian Transfer

Figure 8–5 shows a little-endian transfer of five bytes beginning at address 0x02xx xxx2 on a 64-bit bus. Only one column access is required because the data size is less than 64 bits. The memory cycle accesses the 64-bit doubleword at address 0xxx (xxx represents the byte address within the doubleword). The write begins at address 0010, with the five data bytes positioned on the bus beginning with D[23:16]. $\overline{\text{CAS}}$/DQM[6:2] are active so that only the valid bytes are written. A read access behaves identically to the way a write behaves, except that all the $\overline{\text{CAS}}$/DQM strobes are active, and the invalid bytes are discarded by the TC.

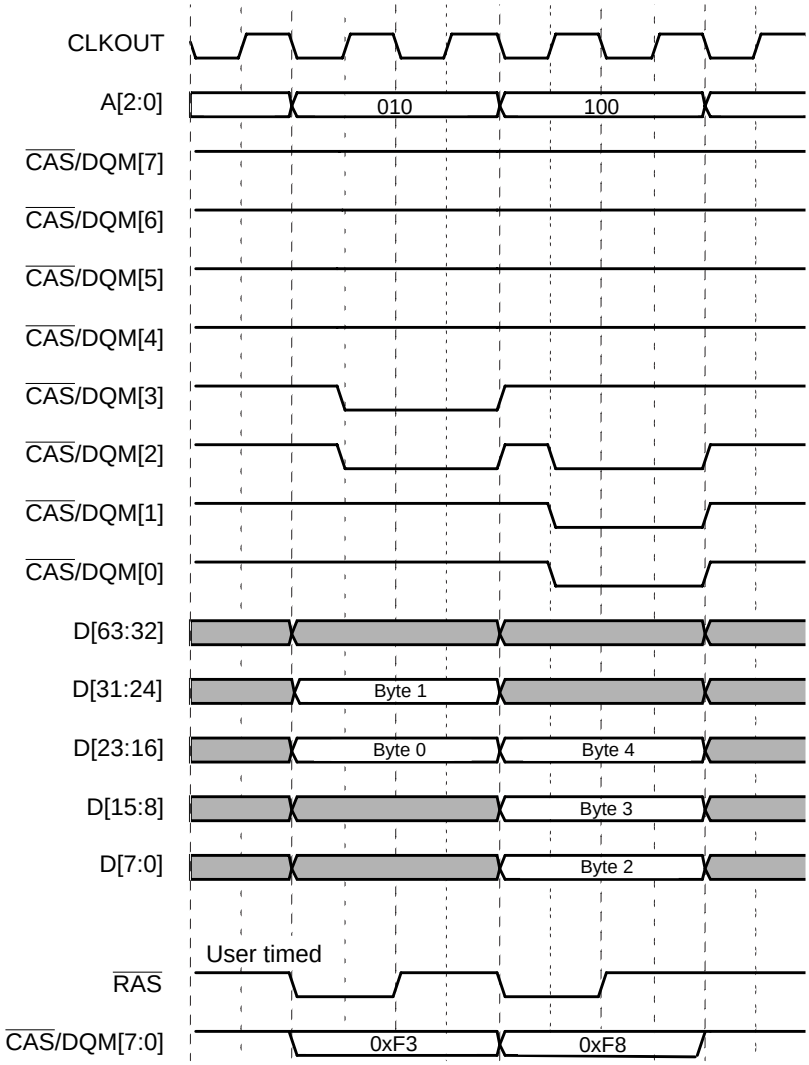
Figure 8–5. 64-Bit Little-Endian Transfer



8.9 Byte Ordering Example for 32-Bit Little-Endian Transfer

Figure 8–6 shows a little-endian transfer of five bytes beginning at address 0x02xx xxx2 on a 32-bit bus. Because the write crosses a 32-bit word boundary, two column accesses are required. The first access is to word 00xx. Only the two MSbytes of 00xx are written, so valid data is placed on D[31:16], and $\overline{\text{CAS}}$/DQM[3:2] are activated. The second access to word 01xx writes the three final bytes. These are written to the LSbytes of the word, so D[23:0] contain valid data and $\overline{\text{CAS}}$/DQM[2:0] are active. For a read access of the same bytes, the same column cycles occur, but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes are discarded by the TC.

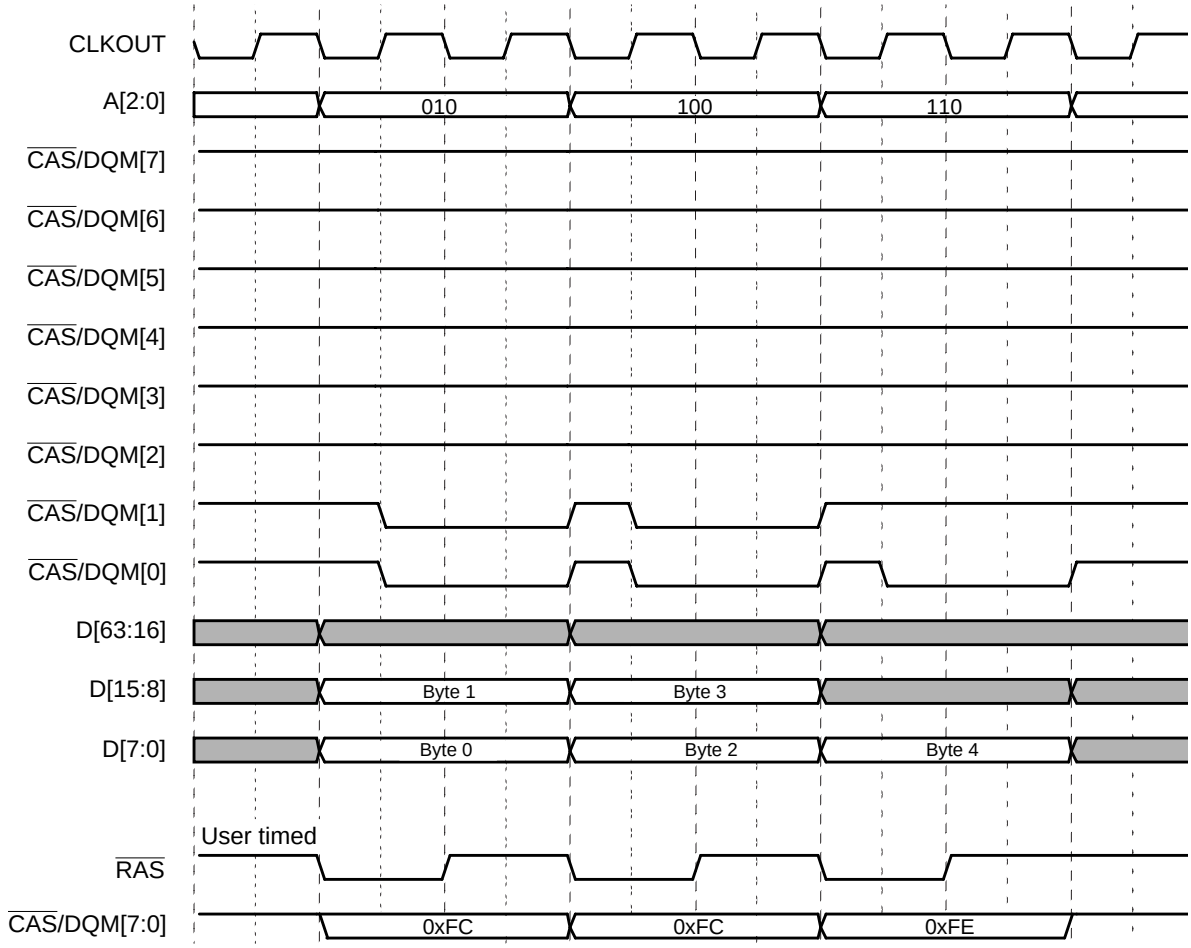
Figure 8–6. 32-Bit Little-Endian Transfer



8.10 Byte Ordering Example for 16-Bit Little-Endian Transfer

Figure 8–7 shows a little-endian transfer of five bytes beginning at address 0x02xx xxx2 on a 16-bit bus. The transfer requires three column accesses. The first access is to halfword 001x. (No access to halfword 000x is required, because none of its bytes are written.) Both bytes in this word are written, so D[15:0] are valid, and both $\overline{\text{CAS}}$/DQM strobes ($\overline{\text{CAS}}$/DQM[1] and $\overline{\text{CAS}}$/DQM[0]) are active. The second column accesses halfword 010x and writes to both of its bytes. The final access is to halfword 011x. One byte is transferred on D[7:0], so only $\overline{\text{CAS}}$/DQM[0] is active. For a read access of the same bytes, the identical column cycles occur, but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes are discarded by the TC.

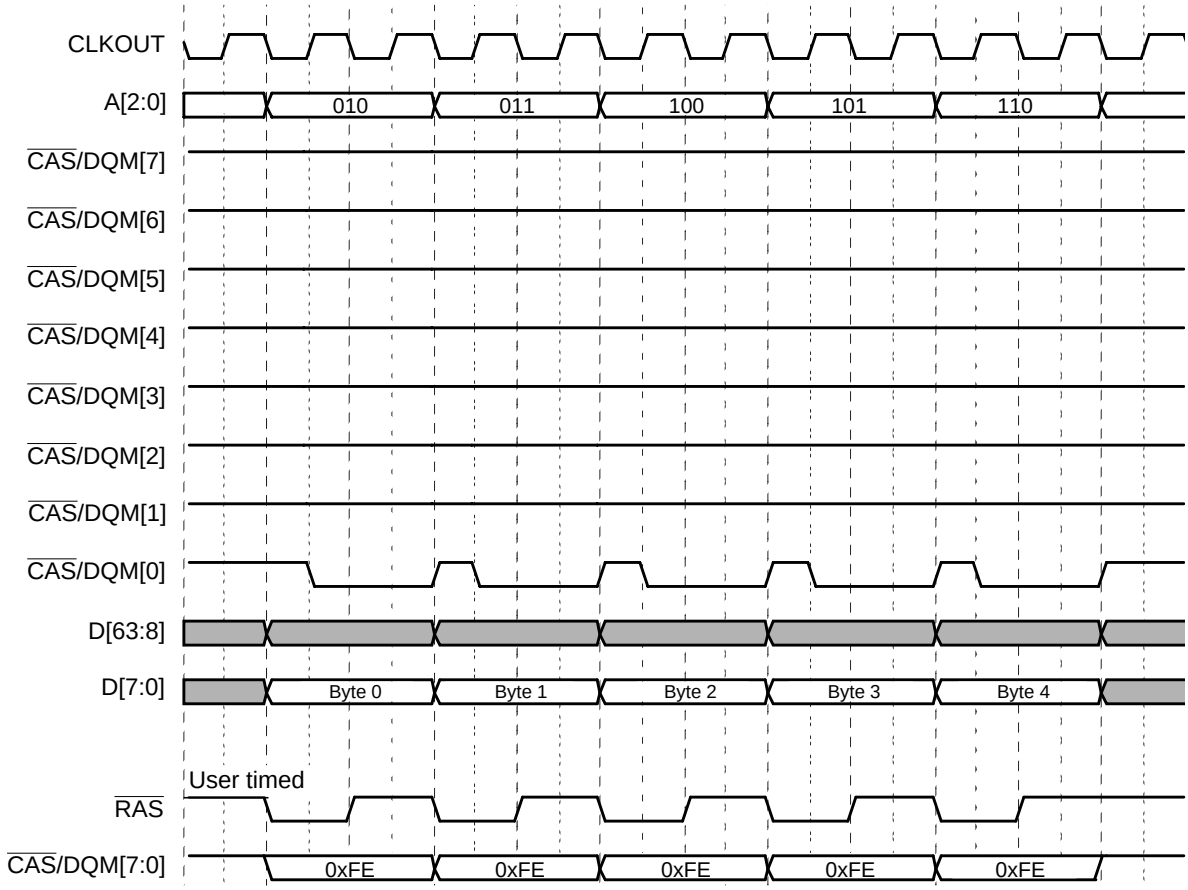
Figure 8–7. 16-Bit Little-Endian Transfer



8.11 Byte Ordering Example for 8-Bit Little-Endian Transfers

Figure 8–8 shows a little-endian transfer of five bytes beginning at address 0x02xx xxx2 on an 8-bit bus. For this bus size, each column access transfers a single byte on D[7:0]. Thus, a total of five column accesses are required. $\overline{\text{CAS}}$/DQM[0] is active for each column access. The five bytes accessed are 0010, 0011, 0100, 0101, and 0110. For a read access of the same bytes, the identical column accesses occur, but all $\overline{\text{CAS}}$/DQM[7:0] strobes are active, and the invalid bytes on D[63:8] are discarded by the TC.

Figure 8–8. 8-Bit Little-Endian Transfer



8.12 How Endian Format Affects Packet-Transfer Parameters

As discussed in Chapter 3, *Packet Transfers*, the packet-transfer parameters are endian-independent at the word (32-bit) level only. The TC always fetches and stores packet-transfer parameters as doubleword (64-bit) transfers. The TC swaps 32-bit words according to the selected endian format. 16-bit quantities within words (such as A count and B count) are not swapped according to the endian format because the field containing these values is considered to be a single 32-bit quantity. Likewise, the 64-bit transparency/color-register-value field is always treated as a single 64-bit quantity and its bytes are not swapped according to the endian format.

Chapter 9

DRAM Cycles

Local memory cycles transfer data between memory and the TMS320C80. These cycles occur as a result of either a packet transfer, a cache request, or a DEA request to the transfer controller.

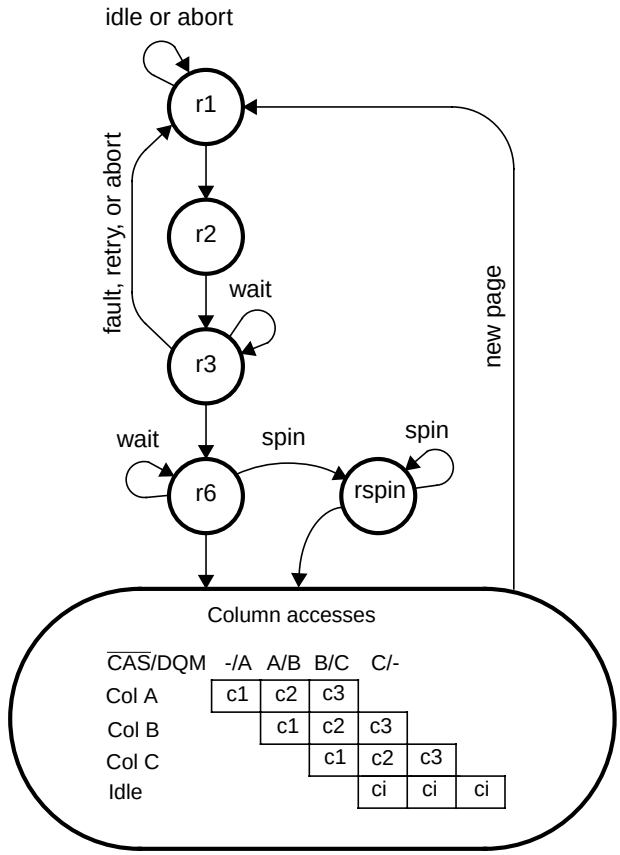
Topics in this chapter include:

Topic **Page**

9.1 Pipelined One-Cycle-per-Column Read

The sequence for a pipelined one-cycle-per-column read is shown in Figure 9–1. This timing is designed for use with pipelined memory devices in which the data is driven out on the cycle after it is addressed. The $\overline{CAS}$/DQM strobes act as clocks to clock the memory pipeline as well as strobe in the column address.

Figure 9–1. Pipelined One-Cycle-per-Column Read States



The row access consists of the r1, r2, r3, r6, and possibly rspin states. The r3 and r6 state can be repeated by adding wait states. The rspin state will always occur once if the bus size has changed from the previous access.

The column accesses consist of a three-stage pipeline. A new column access can begin every cycle, but each access takes three cycles to complete. The pipeline stages are as follows:

c1 The address (A[31:0]) and processor activity code (STATUS[5:0]) for the current access are output, and all $\overline{\text{CAS}}$/DQM strobes are activated. This latches the address into the memories.

c2 The $\overline{\text{CAS}}$/DQM strobes are all activated again to clock out the data addressed in c1.

c3 The data clocked out in c2 is latched by the TMS320C80.

ci Column idle stage. $\overline{\text{CAS}}$/DQM strobes are inactive and no data is latched. This stage occurs when no access is loaded into the pipeline.

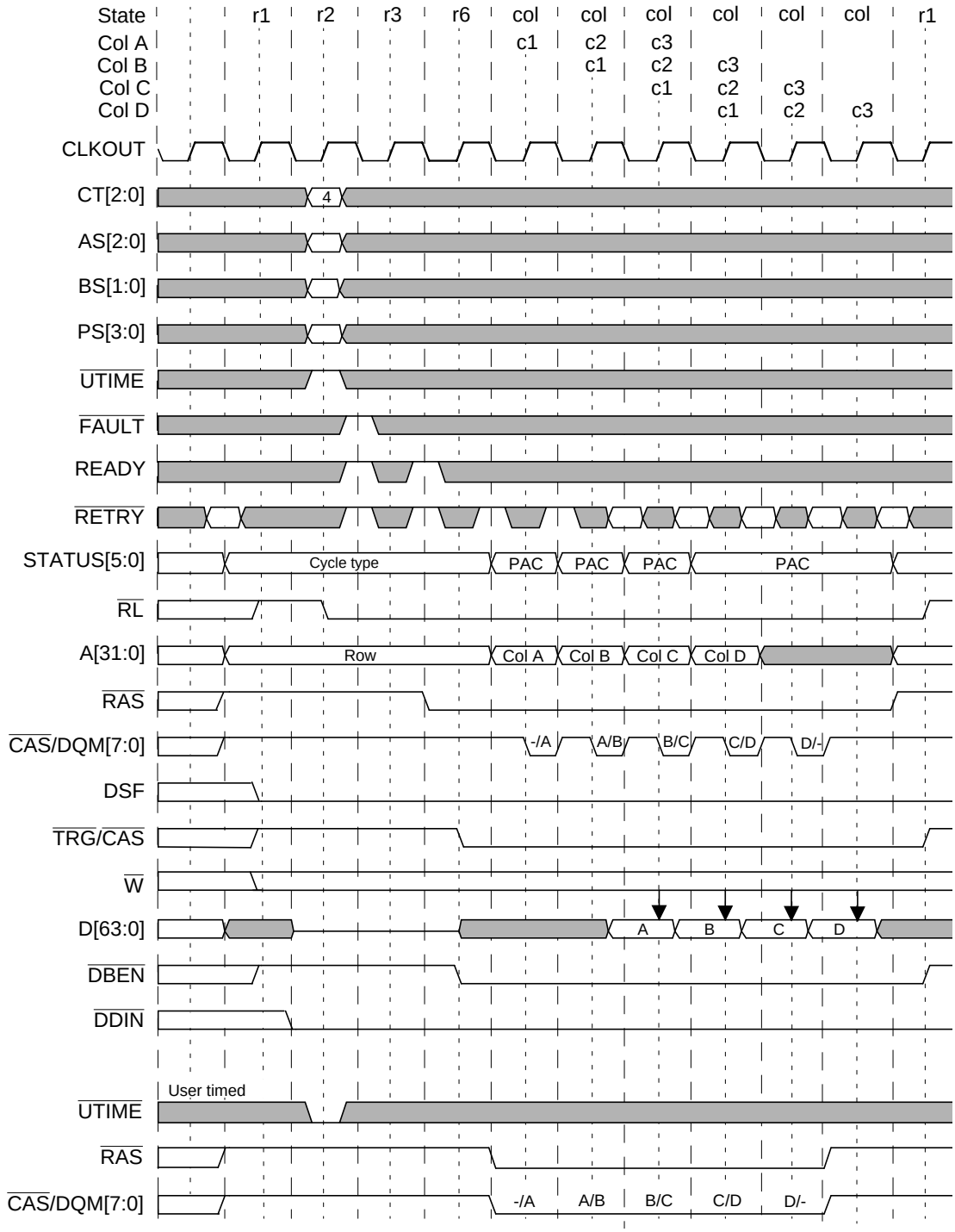
Because the pipeline stages overlap, the $\overline{\text{CAS}}$/DQM strobes may represent either the address strobes for the current access (c1 stage), or the data strobe for the previous access (c2 stage). Once a column access begins, all pipeline stages are completed.

After completing the last access, the memory interface returns either to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column pipeline sequence will begin again.

9.1.1 Pipelined One-Cycle-per-Column Read Example

The example in Figure 9–2 shows a page mode read of four column addresses. A new column access begins on each cycle. The $\overline{\text{CAS}}$/DQM labels indicate the access for which the $\overline{\text{CAS}}$/DQM strobe represent the data and address strobe, respectively. For example, A/B indicates that the $\overline{\text{CAS}}$/DQM is strobing data A out and address B in. After access D is completed, the memory interface returns to the r1 state to begin a new row access.

Figure 9–2. Pipelined One-Cycle-per-Column Read



9.1.2 User-Timed Pipelined One-Cycle-per-Column Read Example

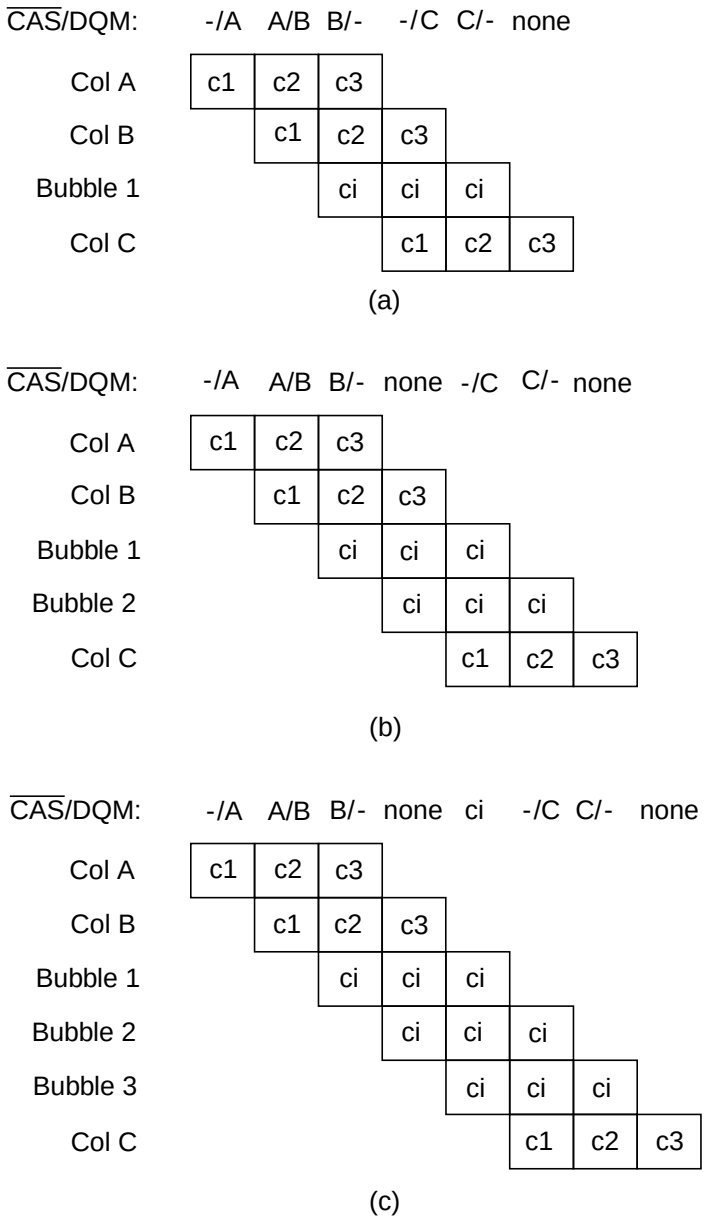
The bottom of Figure 9–2 shows how the cycle is modified when user timing is selected. Note that the pipeline sequence does not change; only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition. $\overline{RAS}$ goes low for one cycle during an address or data strobe stage. For pipelined one-cycle-per-column reads, these are the c1 and c2 stages. Because a new access can begin on each cycle, the $\overline{RAS}$ strobe remains low throughout the entire column access time unless pipeline bubbles occur.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address and data strobe time. This allows the $\overline{CAS}$/DQM strobes to provide early indication of which bytes are being accessed. During read cycles, all $\overline{CAS}$/DQM strobes are active, as 64 bits are always read. For pipelined one-cycle-per-column reads, the $\overline{CAS}$/DQM strobes are active during c1 and c2 pipeline stages.

9.1.3 Pipelined One-Cycle-per-Column Read With Pipeline Bubbles

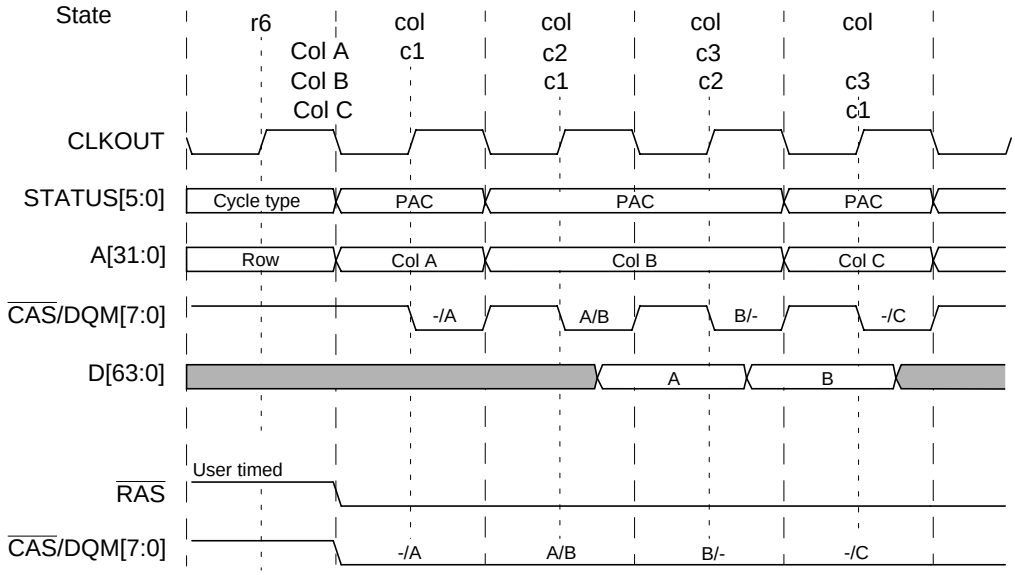
When TC pipeline bubbles occur during page mode reads, the loading of new column accesses into the pipeline is delayed but previously loaded cycles continue through their c1, c2, and c3 stages. This is shown in Figure 9–3.

Figure 9–3. The Effect of Bubbles on Pipeline Operation



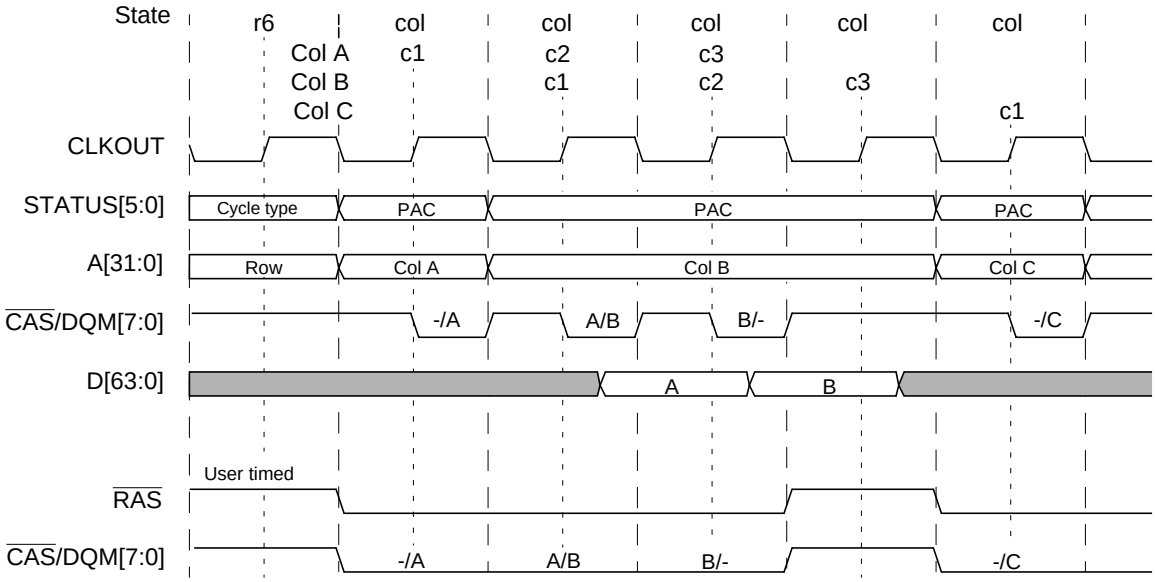
❑ **Pipelined one-cycle-per-column read with single-cycle bubbles.** Figure 9−3(a) shows the pipeline sequence for a single cycle bubble. After beginning columns A and B, a bubble occurs preventing the column C access from beginning on the next cycle. Stages c3 of column A, and c2 of column B occur as normal. In the next cycle, the column C access is ready to begin, and the pipeline resumes normal operation. The waveform for single-cycle bubbles is shown in Figure 9−4.

Figure 9−4. Pipelined One-Cycle-per-Column Read (Single Cycle Bubbles)



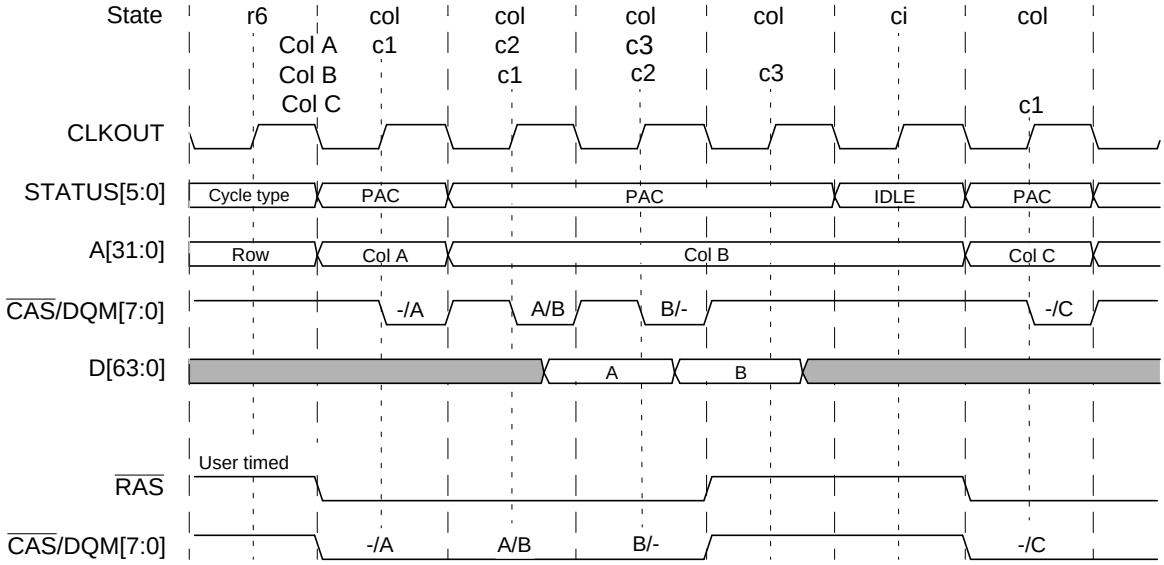
- **Pipelined one-cycle-per-column read with two cycle bubbles.** When a TC pipeline bubble lasts for two cycles, the pipeline appears as shown in Figure 9–3(b). The waveform for two-cycle bubbles is shown in Figure 9–5.

Figure 9–5. Pipelined One-Cycle-per-Column Read (Two-Cycle Bubbles)



- **Pipelined one-cycle-per-column read with multi-cycle bubbles.** If the TC's pipeline bubble lasts longer than two cycles, the pipeline drains and the memory interface enters the idle (ci) state until the next access is ready to begin. This is shown in Figure 9–3(c). The waveform for multi-cycle bubbles is shown in Figure 9–6.

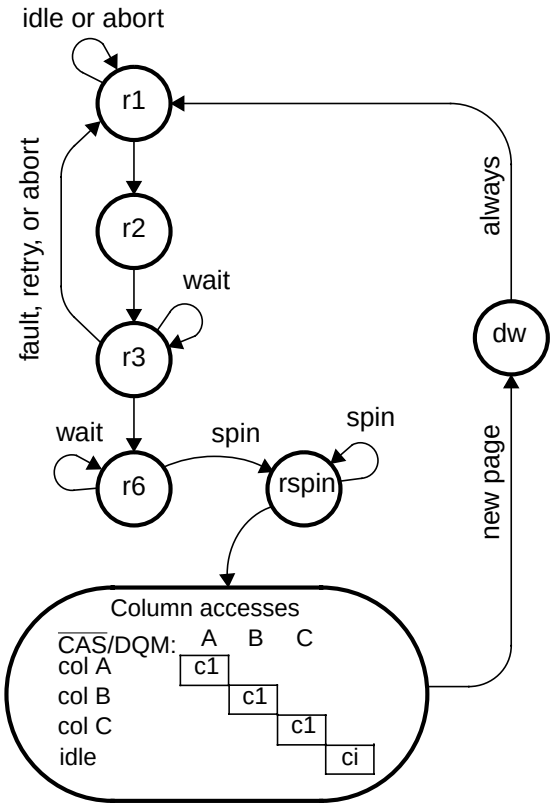
Figure 9–6. Pipelined One-Cycle-per-Column Read (Multi-Cycle Bubbles)



9.2 Pipelined One-Cycle-per-Column Write

Figure 9–7 shows the state sequence for a pipelined one-cycle-per-column write. This timing is designed for use with pipelined page-mode memory devices. Although these devices latch data at the fall of $\overline{CAS}$/DQM, it is not written into the memory array until the following $\overline{CAS}$/DQM strobe. The $\overline{CAS}$/DQM strobes act as clocks to clock the memory pipeline as well as strobing in the column address.

Figure 9–7. Pipelined One-Cycle-per-Column Write States



The row access consists of the r1, r2, r3, r6, and rspin states. The r3 and r6 states can be repeated by adding wait states. The rspin state *always* occurs at least twice for pipelined one-cycle-per-column writes.

Column accesses consist of a single-stage pipeline. A new column access may begin every cycle. Each access takes two cycles to complete. However, the second cycle for each accessed byte corresponds to the next cycle in which the same $\overline{\text{CAS}}$/DQM strobe is active, and thus may not occur immediately. The pipeline stages are as follows:

c1 The TC drives the address (A[31:0]) and processor activity code (STATUS[5:0]) for the current access. Data is also driven on the appropriate bytes of the data bus (D[63:0]) and the $\overline{\text{CAS}}$/DQM strobes corresponding to the valid bytes are activated. This latches the address and data into the memory device. The $\overline{\text{CAS}}$/DQM strobes also causes the data from any previous write with the corresponding $\overline{\text{CAS}}$/DQM to be written into the memory array.

ci This stage (column idle) occurs when no access is loaded into the pipeline. $\overline{\text{CAS}}$/DQM strobes are inactive and no new data is output.

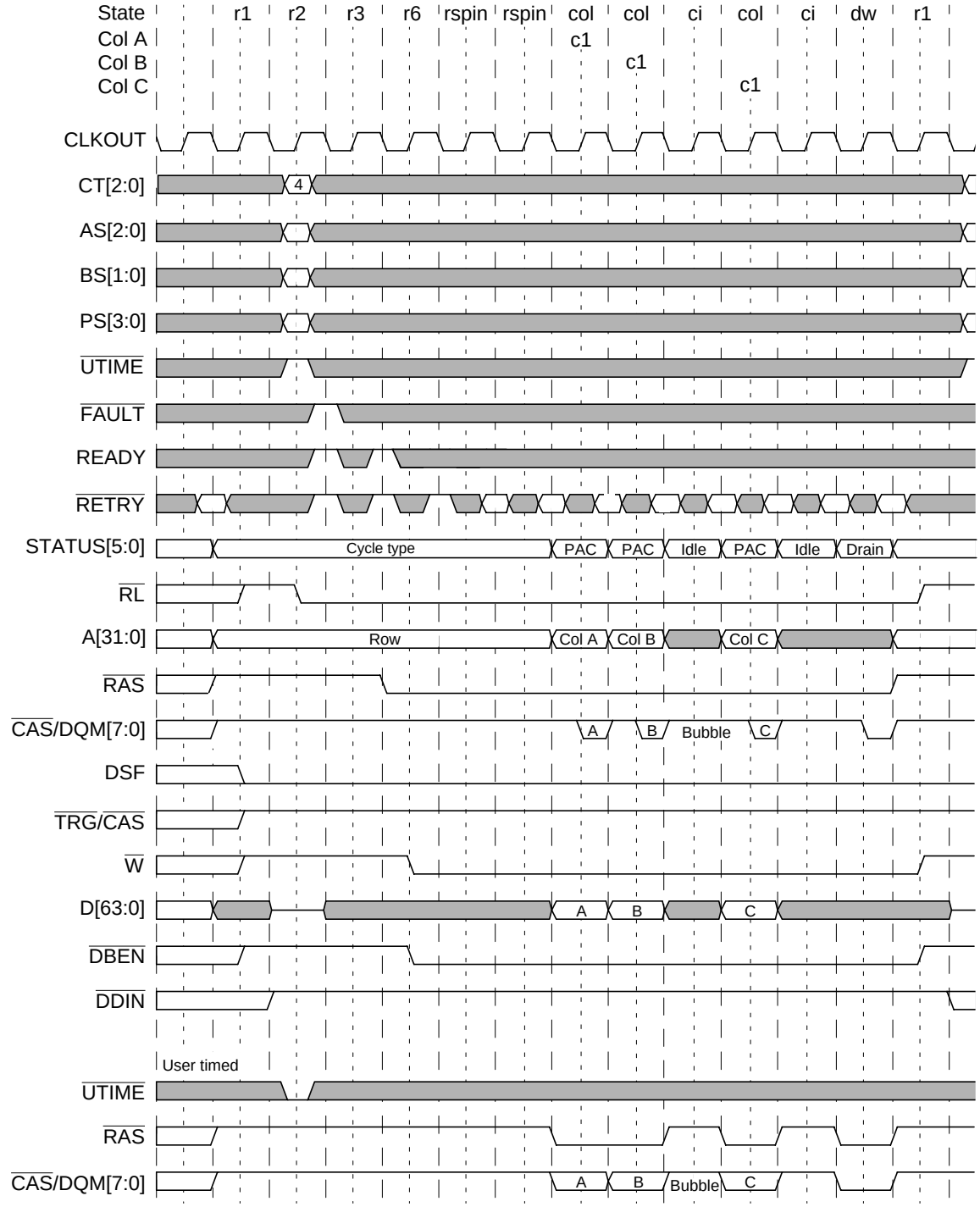
dw This special state always occurs immediately before returning to the r1 state (to begin a new page). It activates all $\overline{\text{CAS}}$/DQM[7:0] strobes to update the memory array with the last data written.

After completing the last access, the memory interface either returns to the r1 state (via dw) if a new row access is required, or enter the ci state to wait for the next access. If the next access does not require a new row address, then the column-pipeline sequence begins again.

9.2.1 Pipelined One-Cycle-per-Column Write Example

The example in Figure 9–8 shows a page-mode write of three column addresses. The $\overline{\text{CAS}}$/DQM labels indicate the access to which the strobes correspond. In this example, the memory interface is idle (ci) for one cycle after completing the last write (column C) before receiving a request requiring a new page access. The TC enters the dw state to ensure that the last data written to each byte is transferred into the memory array. Then, the interface returns to the r1 state to begin a new row access.

Figure 9–8. Pipelined One-Cycle-per-Column Write



9.2.2 User-Timed Pipelined One-Cycle-per-Column Write

The bottom of Figure 9–8 shows how the cycle is modified when user timing is selected. The pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition changes. $\overline{RAS}$ goes low for one cycle during each column access. Because a new access may begin on each cycle, the $\overline{RAS}$ strobe remains low throughout the entire column time unless pipeline bubbles occur.

The $\overline{CAS}$/DQM output timing is modified so that $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide an earlier indication of which bytes are being accessed. During write cycles, only those $\overline{CAS}$/DQM strobes corresponding to valid data bytes are active. For pipelined one-cycle-per-column writes, the $\overline{CAS}$/DQM strobes are active during the c1 pipeline stage and the dw state.

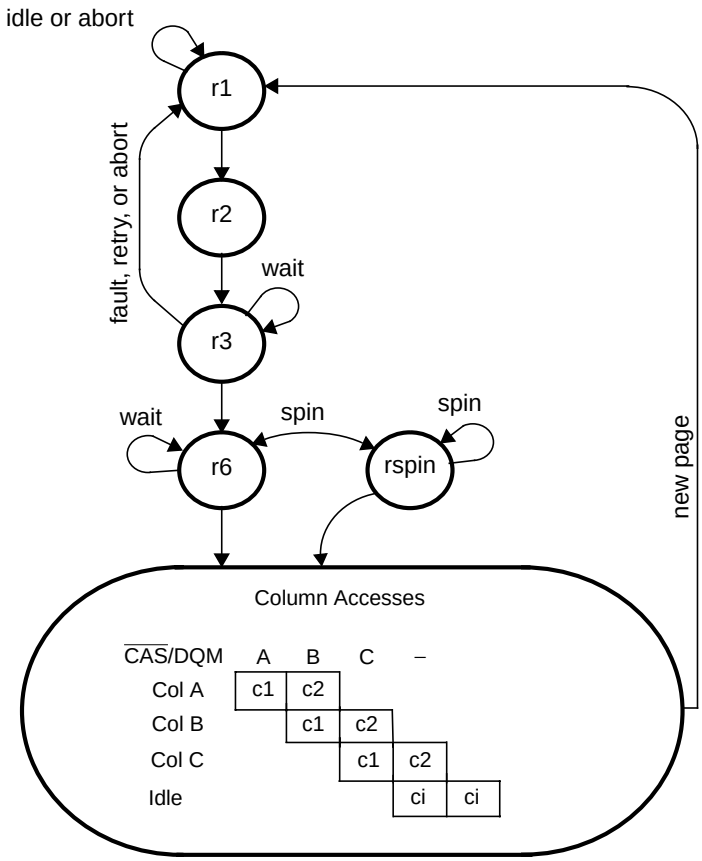
9.2.3 Pipelined One-Cycle-per-Column Write Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page-mode writes simply delay the loading of the next access into the pipeline. These delays appear as ci states externally. Figure 9–8 shows the occurrence of a single bubble between column B and column C accesses.

9.3 Nonpipelined One-Cycle-per-Column Read

The sequence for a nonpipelined one-cycle-per-column read cycle is shown in Figure 9–9.

Figure 9–9. Nonpipelined One-Cycle-per-Column Read States



The row access consists of the r1, r2, r3, r6, and possibly rspin states. The r3 and r6 states can be repeated by adding wait states. The rspin state will always occur once if the bus size has changed from the previous access.

The column accesses consist of a two-stage pipeline. A new column access can begin every cycle, but each access takes two cycles to complete. The pipeline stages are as follows:

c1 The address (A[31:0]) and processor activity code (STATUS[5:0]) for the current access are output and all $\overline{\text{CAS}}$/DQM strobes are activated. This latches the address into the memories.

c2 The data addressed in c1 is latched by the TMS320C80.

ci Column idle stage. $\overline{\text{CAS}}$/DQM strobes are inactive and no data is latched. This stage occurs when no column access is loaded in the pipeline.

Once a column access begins, all pipeline stages are completed.

After completing the last access, the memory interface either returns to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column-pipeline sequence begins again.

9.3.1 Nonpipelined One-Cycle-per-Column Read Example

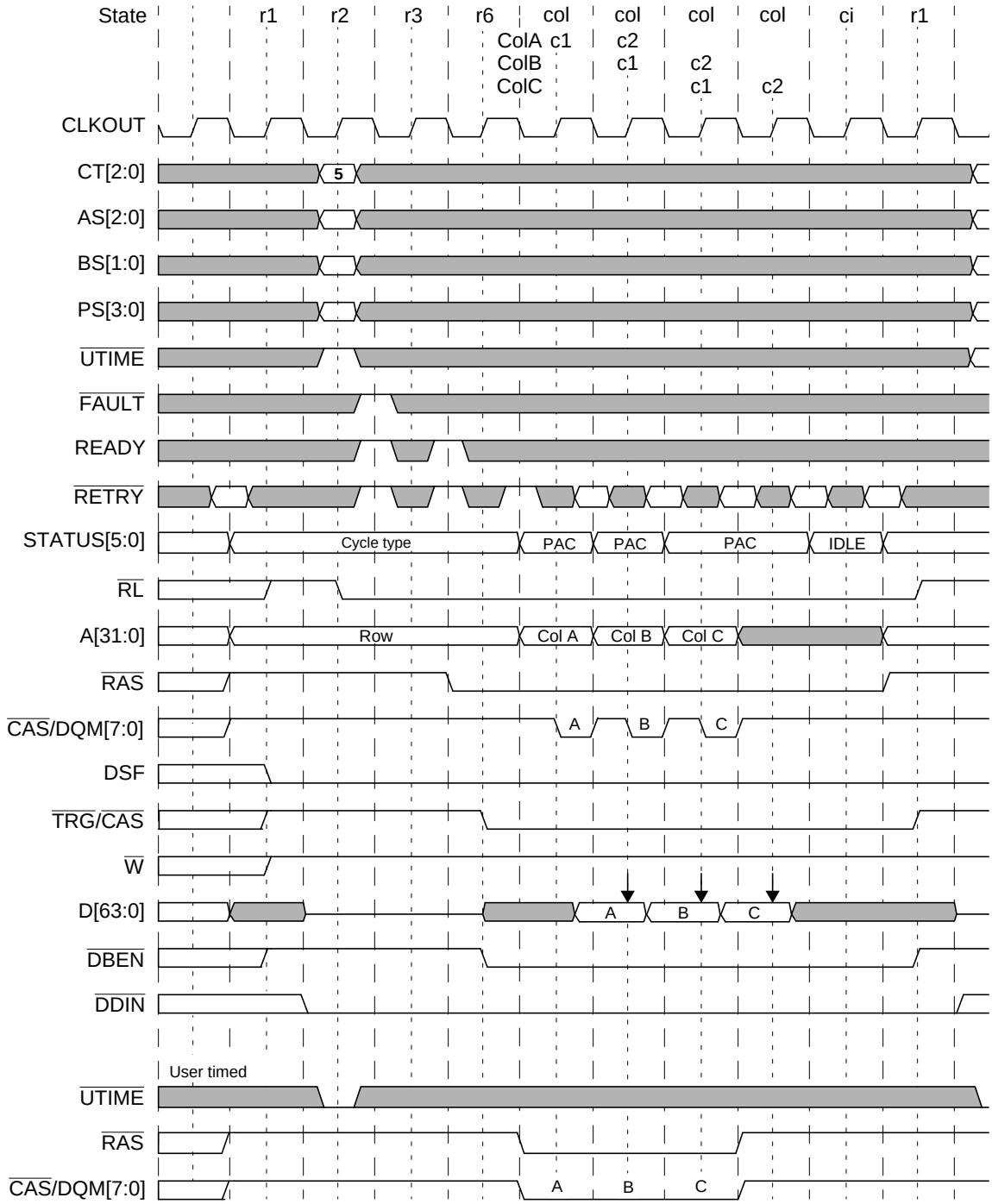
The example in Figure 9–10 shows a page-mode read of three column addresses. A new column access begins on each cycle. The $\overline{\text{CAS}}$/DQM labels indicate the column access that corresponds to each strobe. In this example, the memory interface is idle for one cycle after completing access C. The next access requires a new row access, so the interface then returns to the r1 state.

9.3.2 User-Timed Nonpipelined One-Cycle-per-Column Read Example

The bottom of Figure 9–10 shows how the cycle is modified when user timing is selected. Note that the pipeline sequence does not change, only the way in which the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM signals transition. $\overline{\text{RAS}}$ goes low for one cycle during each address stage. For nonpipelined one-cycle-per-column reads, this is the c1 stage. Because a new c1 stage can begin each cycle, the $\overline{\text{RAS}}$ strobe remains low throughout the entire column access time unless pipeline bubbles occur.

The $\overline{\text{CAS}}$/DQM output timing is modified so that the $\overline{\text{CAS}}$/DQM strobes are active for the entire address time. This allows the $\overline{\text{CAS}}$/DQM strobes to provide early indication of which bytes are being accessed. During read cycles, all $\overline{\text{CAS}}$/DQM strobes are active, as 64 bits are always read. For nonpipelined one-cycle-per-column reads, the $\overline{\text{CAS}}$/DQM strobes are active during the c1 pipeline stage.

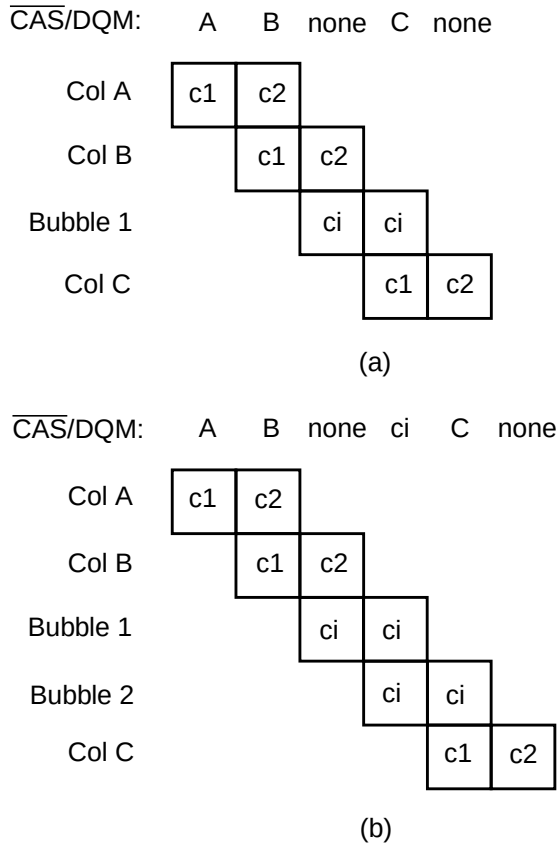
Figure 9–10. Nonpipelined One-Cycle-per-Column Read



9.3.3 Nonpipelined One-Cycle-per-Column Read With Pipeline Bubbles

When TC pipeline bubbles occur during page mode reads, the loading of new column accesses into the pipeline is delayed but previously loaded cycles continue through their c1 and c2 stages. This is shown in Figure 9–11.

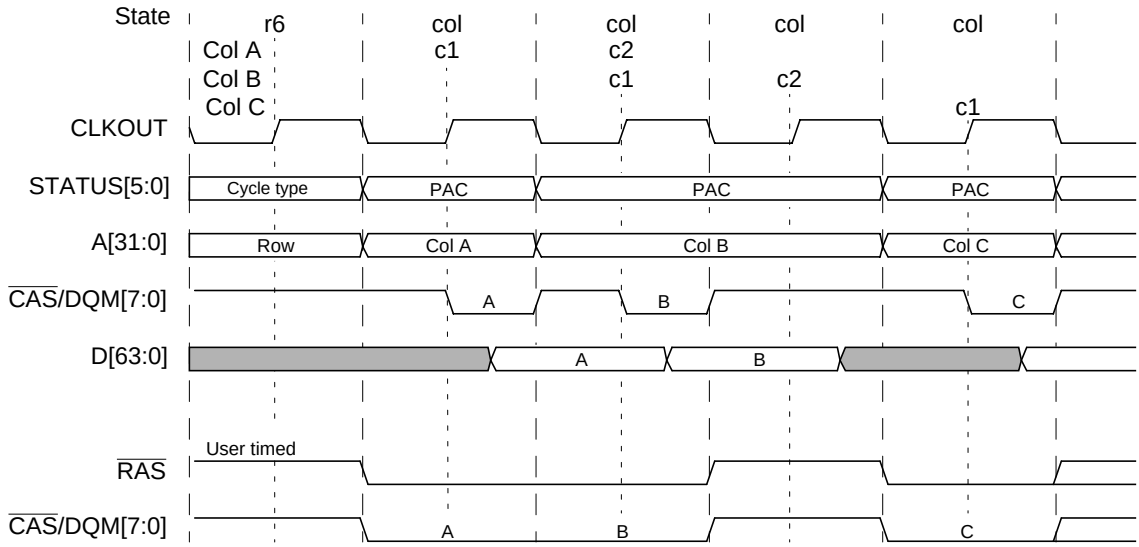
Figure 9–11. The Effect of Bubbles on the Pipeline Operation



- **Nonpipelined one-cycle-per-column read with single-cycle bubbles.** Figure 9–11(a) shows the pipeline sequence for a single cycle bubble. After beginning columns A and B, a bubble occurs preventing the column C access from beginning on the next cycle. Stage c2 of column B occurs as normal. In the next cycle, the column C access is ready to begin and

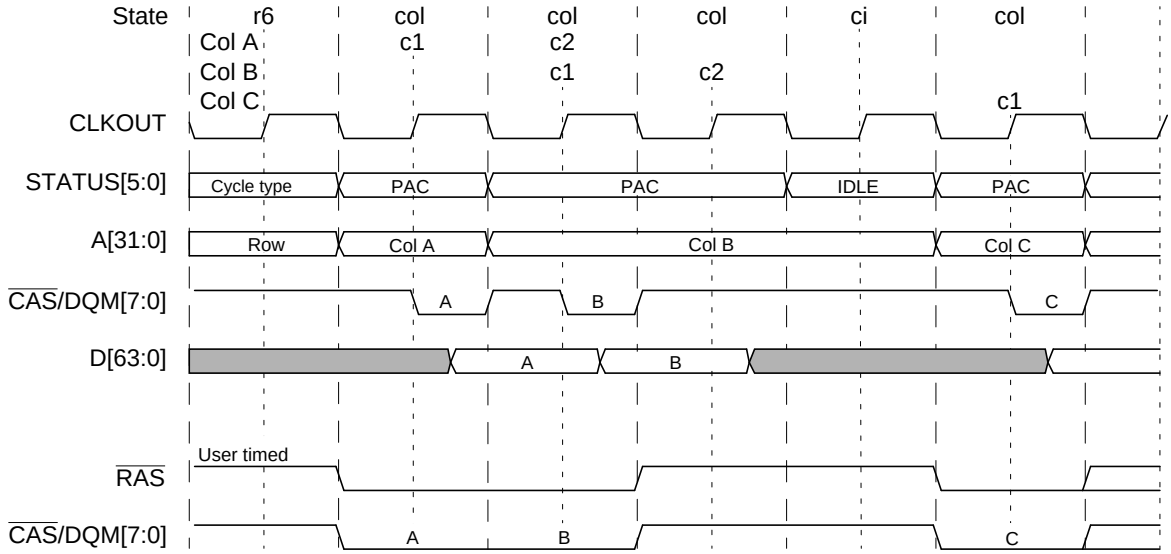
the pipeline resumes normal operation. The waveform for single cycle bubbles is shown in Figure 9–12.

Figure 9–12. Nonpipelined One-Cycle-per-Column Read (Single Cycle Bubbles)



- **Nonpipelined one-cycle-per-column read with multi-cycle bubbles.** If the TC's pipeline bubble lasts longer than one cycle, the pipeline drains and the memory interface enters the idle (ci) state until the next access is ready to begin. This is shown in Figure 9−11(b). The waveform for multi-cycle bubbles is shown in Figure 9−13.

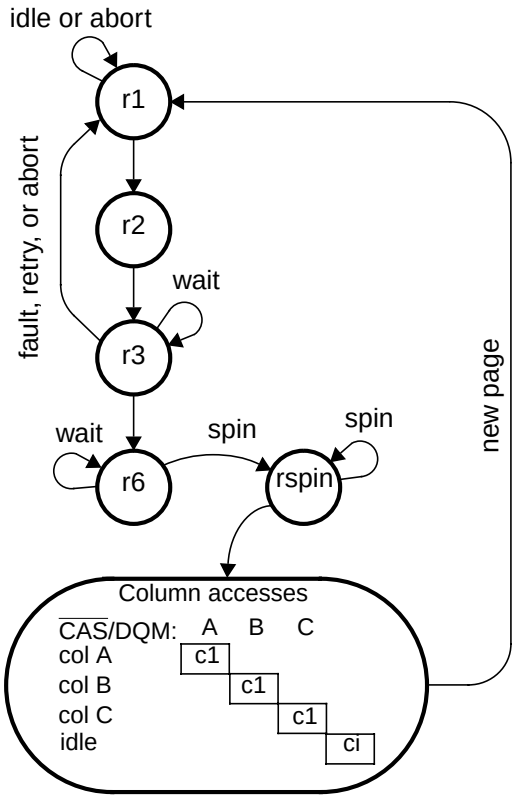
Figure 9−13. Nonpipelined One-Cycle-per-Column Read (Multi-Cycle Bubbles)



9.4 Nonpipelined One-Cycle-per-Column Write

The state sequence for a nonpipelined one-cycle-per-column write cycle is shown in Figure 9–14.

Figure 9–14. Nonpipelined One-Cycle-per-Column Write States



The row access consists of the r1, r2, r3, r6, and rspin states. The r3 and r6 states can be repeated by adding wait states. The rspin state *always* occurs at least twice for nonpipelined one-cycle-per-column writes.

The column accesses consist of a single-stage pipeline. A new column access may begin every cycle. Each access takes one cycle to complete. The pipeline stages are as follows:

c1 The TC drives the address (A[31:0]) lines and processor activity code (STATUS[5:0]) for the current access. The TC also drives data on the appropriate bytes of the data bus (D[63:0]), and the $\overline{\text{CAS}}$/DQM strobes corresponding to the valid bytes are activated.

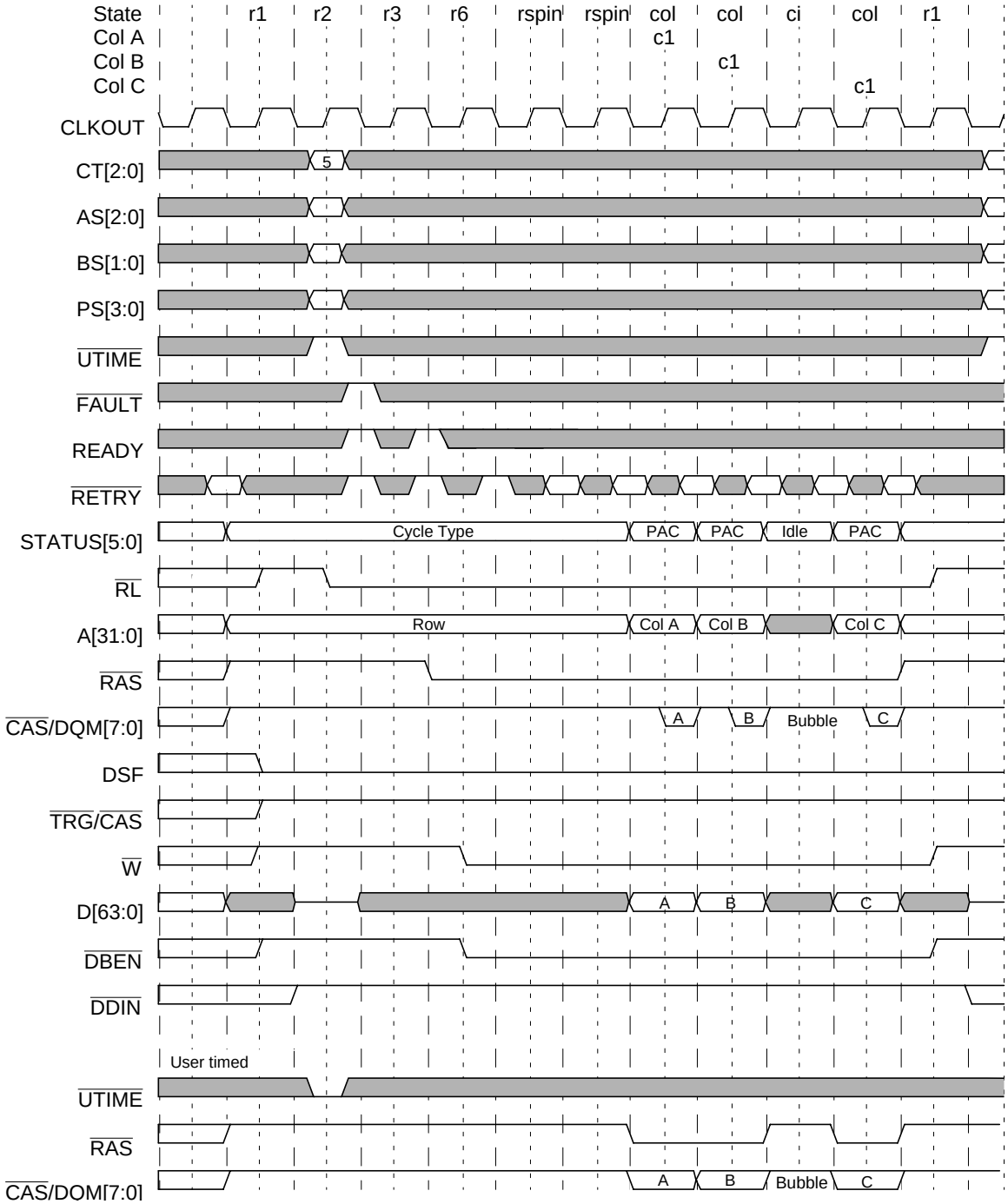
ci This stage (column idle) occurs when no access is loaded into the pipeline. $\overline{\text{CAS}}$/DQM strobes are inactive and no new data is output.

After completing the last access, the memory interface will either return to the r1 state if a new row access is required, or enter the ci state to wait for the next access. If the next access does not require a new row address, then the column-pipeline sequence will begin again.

9.4.1 Nonpipelined One-Cycle-per-Column Write Example

The example in Figure 9–15 shows a page-mode write of three column addresses. The $\overline{\text{CAS}}$/DQM labels indicate the access to which the strobes correspond. In this example, after completing the last write (column C), a new page access is required and the memory interface immediately returns to the r1 state.

Figure 9–15. Nonpipelined One-Cycle-per-Column Write



9.4.2 User-Timed Nonpipelined One-Cycle-per-Column Write Example

The bottom of Figure 9–15 shows how the cycle is modified when user timing is selected. The pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition changes. $\overline{RAS}$ goes low for one cycle during each column access. Because a new access may begin on each cycle, the $\overline{RAS}$ strobe remains low throughout the entire column-access time unless pipeline bubbles occur.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide an earlier indication of which bytes are being accessed. During write cycles, only those $\overline{CAS}$/DQM strobes corresponding to valid data bytes are active.

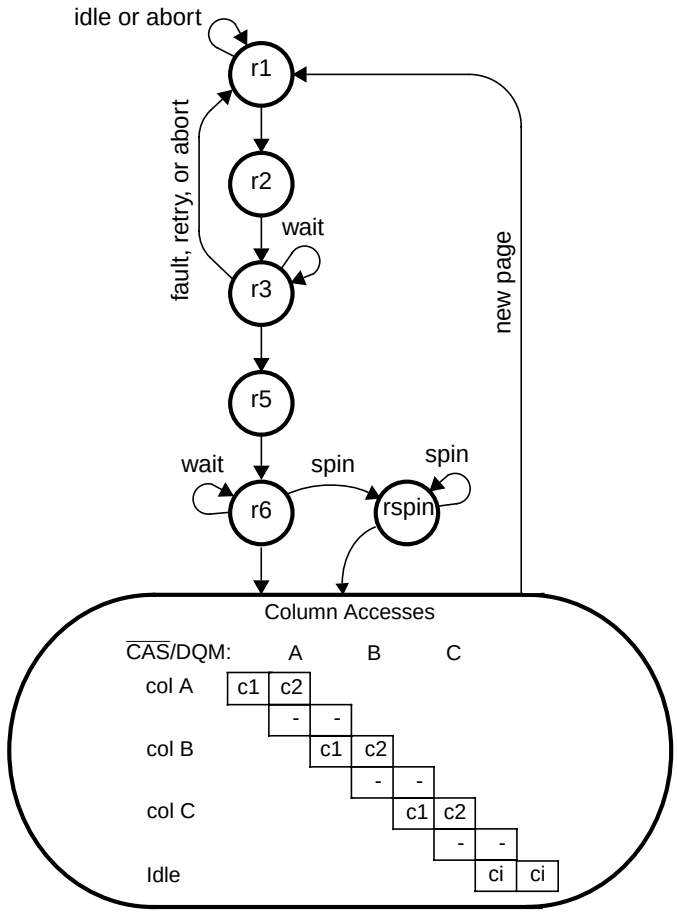
9.4.3 Nonpipelined One-Cycle-per-Column Write Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page-mode writes simply delay the loading of the next access into the pipeline. These delays appear as ci states externally. Figure 9–15 shows the occurrence of a single bubble between column B and column C accesses.

9.5 Two-Cycles-per-Column Read

The state sequence for a two cycles-per-column read is shown in Figure 9–16.

Figure 9–16. Two Cycle-per-Column Read State Sequence



The row access consists of the r1, r2, r3, r5, r6, and possibly rspin states. The r3 and r6 states can be repeated by adding wait states. The r5 state begins with the fall of $\overline{\text{RAS}}$ to extend the available $\overline{\text{RAS}}$ access time.

The column accesses consist of a two-stage pipeline. A new column access can begin every other cycle. Each access takes two cycles to complete so accesses never overlap. The pipeline stages are as follows:

c1 The address (A[31:0]) and processor activity code (STATUS[5:0]) for the current access are output and all $\overline{\text{CAS}}$/DQM strobes are activated. This latches the address into the memories.

c2 The READY input is sampled and if low, the c2 stage is repeated and READY sampled until it is sampled high. The data is latched by the TMS320C80 in the final c2 stage.

ci Column idle stage. $\overline{\text{CAS}}$/DQM strobes are inactive and no data is latched. This stage occurs when no access is loaded in the pipeline.

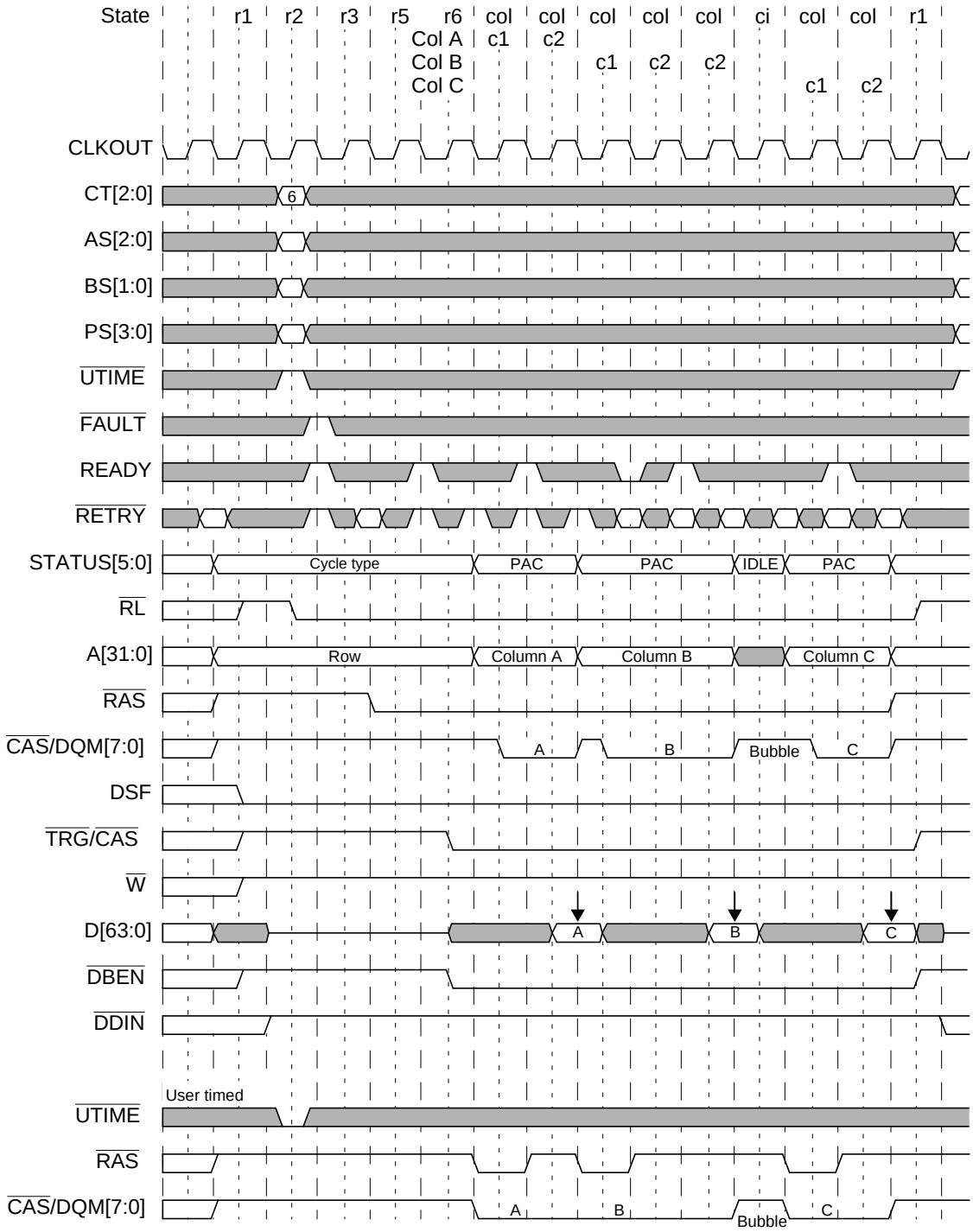
Once a column access begins, all pipeline stages are completed.

After completing the last access, the memory interface either returns to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column pipeline sequence begins again.

9.5.1 Two-Cycles-per-Column Read Example

The example in Figure 9–17 shows a page-mode read of three column addresses. The $\overline{\text{CAS}}$/DQM labels indicate which column access corresponds to each strobe. In this example, the column B access shows an added column time wait state. The READY input is driven low by external logic at the start of the c2 stage. This causes the c2 stage to repeat at which time READY is sampled high, input data B is latched, and the pipeline advances to the next access.

Figure 9–17. Two-Cycles-per-Column Read



9.5.2 User-Timed Two-Cycles-Per-Column Read Example

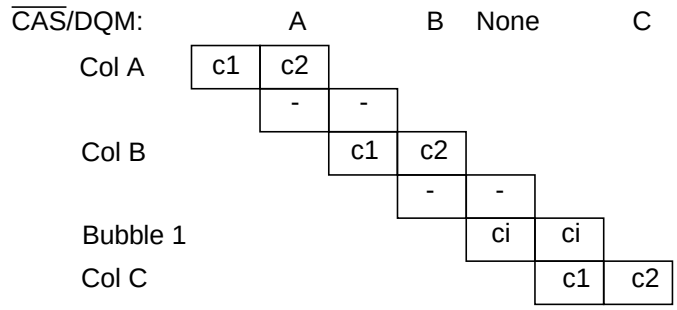
The bottom of Figure 9–17 shows how the cycle is modified when user timing is selected. The pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition changes. $\overline{RAS}$ goes low for one cycle during each c1 stage to indicate the beginning of each column access.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide early indication of which bytes are being accessed. During read cycles, all $\overline{CAS}$/DQM strobes are active as 64 bits are always read. For two cycle-per-column reads, the $\overline{CAS}$/DQM strobes are active during the c1 and c2 stages, including any added wait states, but are inactive during idle (ci) states.

9.5.3 Two-Cycles-Per-Column Read With Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page mode reads simply delay the loading of the next access into the pipeline as shown in Figure 9–18. These delays appear as ci states externally. Figure 9–17 shows the occurrence of a single bubble between the column B and column C accesses.

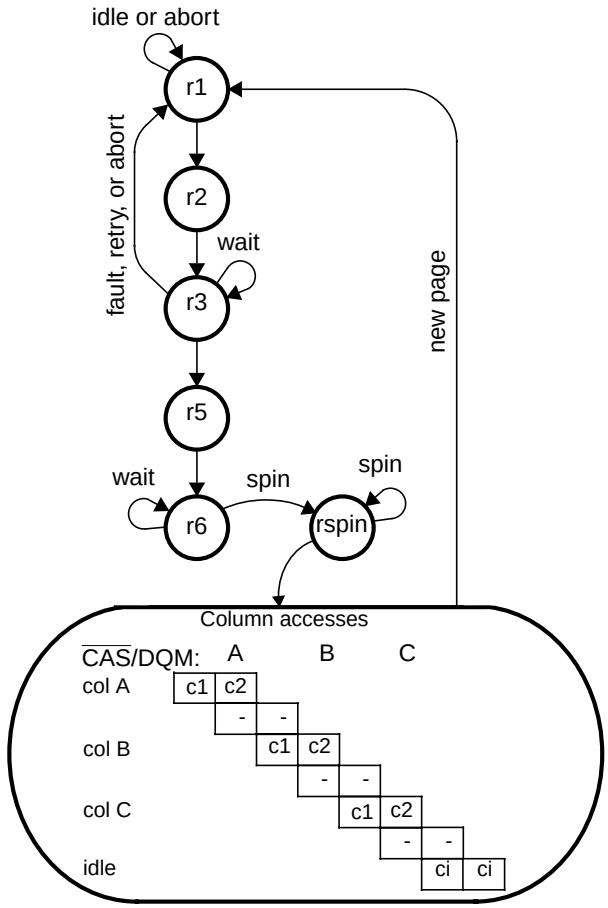
Figure 9–18. The Effect of Bubbles on Pipeline Operation



9.6 Two-Cycles-per-Column Write

The state sequence for a two-cycles-per-column write cycle is shown in Figure 9–19.

Figure 9–19. Two-Cycles-per-Column Write States



The row access consists of the r1, r2, r3, r5, r6, and rspin states. The r3 and r6 states may be repeated by adding wait states. The rspin state will *always* occur at least once for two-cycle-per-column writes.

The column accesses consist of a two-stage pipeline. A new column access may begin every other cycle. Each access requires two cycles to complete. Pipeline stages are as follows:

c1 The TC drives the address (A[31:0]) lines and processor activity code (STATUS[5:0]) for the current access. The TC also drives data on the appropriate bytes of the data bus (D[63:0]), and the $\overline{\text{CAS}}$/DQM strobes corresponding to the valid bytes are activated.

c2 Valid data (D[63:0]) and the $\overline{\text{CAS}}$/DQM strobes continue to be driven. The READY input is sampled, and if it is low, the pipeline stalls and the c2 state is repeated. Once READY is sampled high, the pipeline operation is resumed.

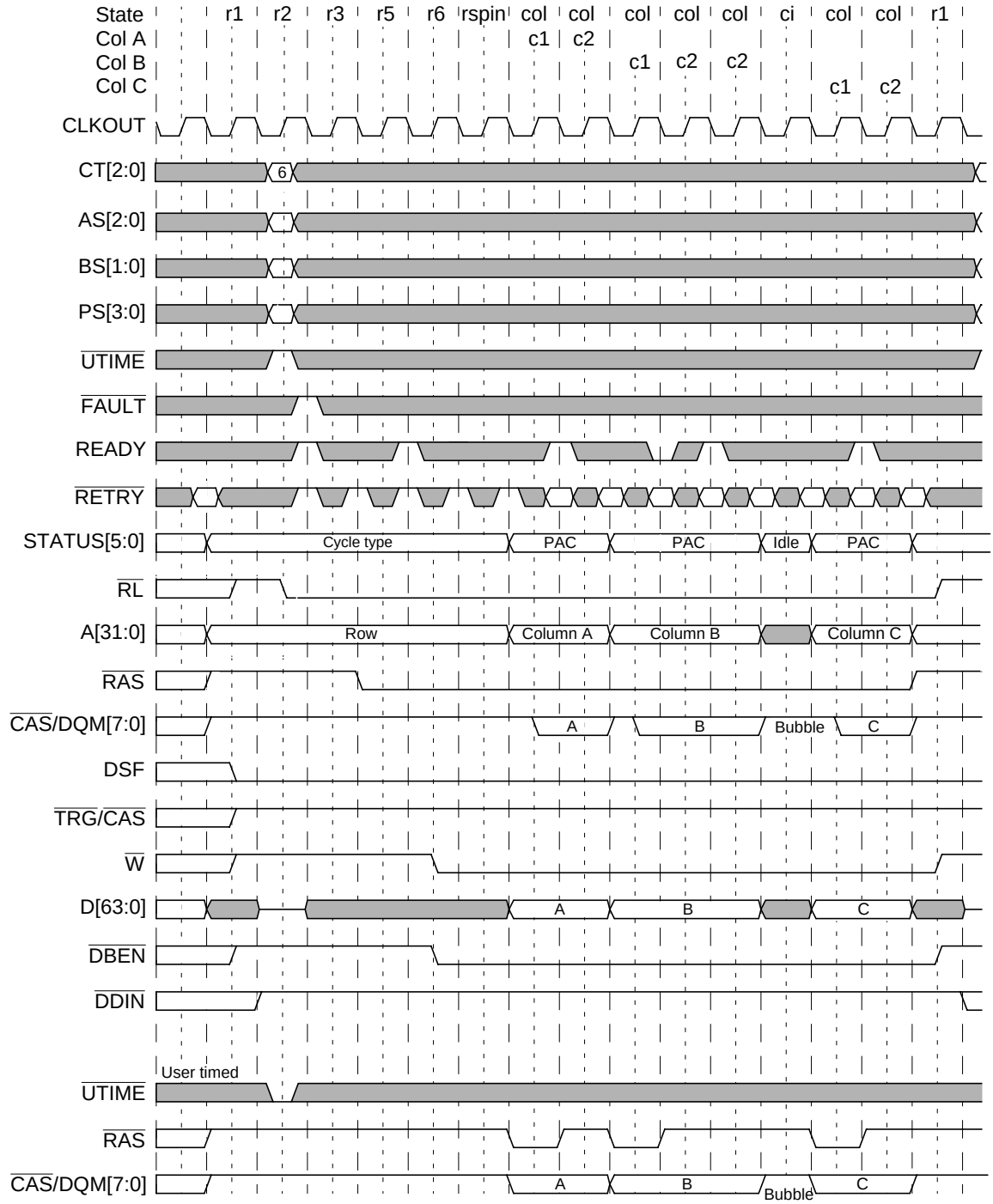
ci This stage (column idle) occurs when no new access is loaded into the pipeline. $\overline{\text{CAS}}$/DQM strobes are inactive and no new data is output.

After completing the last access, the memory interface either returns to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column-pipeline sequence will begin again.

9.6.1 Two-Cycles-per-Column-Write Example

The example in Figure 9–20. shows a page-mode write of three column addresses. The $\overline{\text{CAS}}$/DQM labels indicate the access to which the strobes correspond. In this example, the column-B access is extended one cycle by the insertion of a column-time wait state. External logic drives READY low at the start of the c2 stage. The TMS320C80 samples READY low and causes the c2 stage to be repeated. At the start of the second c2 stage READY has been driven high so the access completes. After completing the last write (column C), a new page access is required and the memory interface immediately returns to the r1 state.

Figure 9–20. Two-Cycles-per-Column Write



9.6.2 User-Timed Two-Cycles-per-Column Write Example

The bottom of Figure 9–20 shows how the cycle is modified when user timing is selected. The pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition changes. $\overline{RAS}$ goes low for one cycle during each column access.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide an earlier indication of which bytes are being accessed. During write cycles, only those $\overline{CAS}$/DQM strobes corresponding to valid data bytes are active.

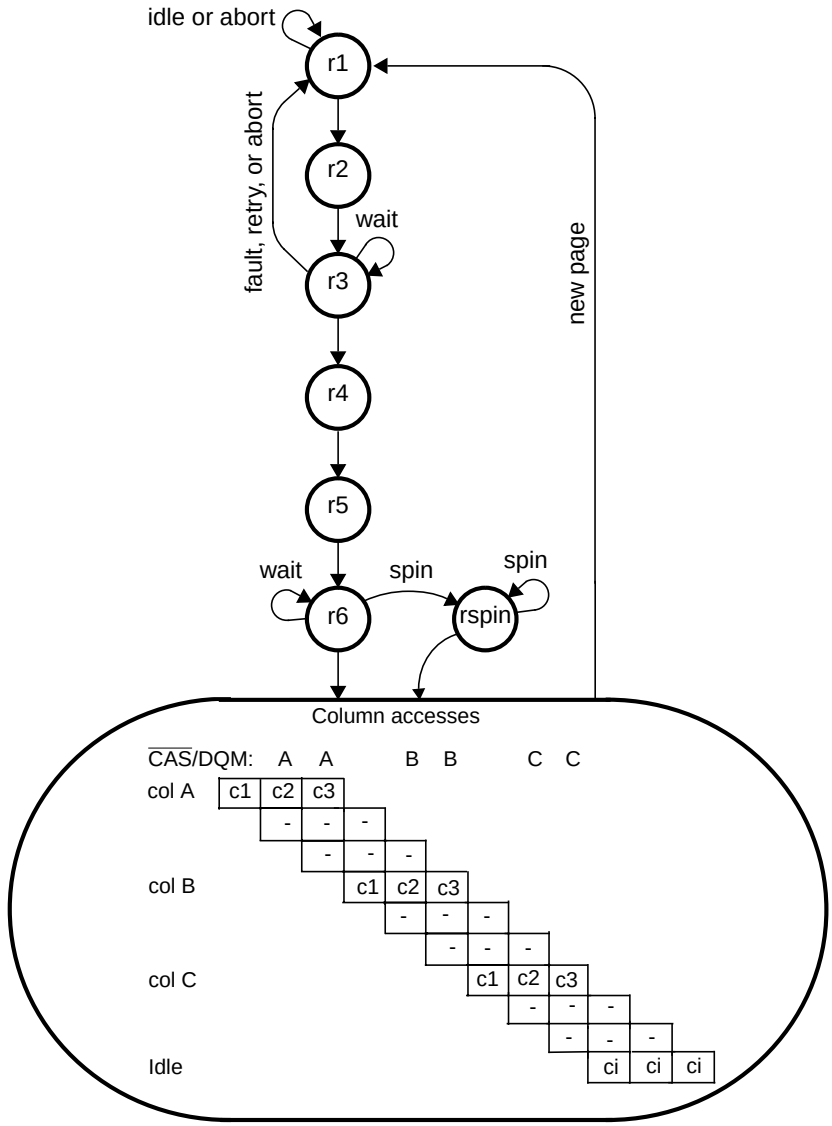
9.6.3 Two-Cycles-per-Column Write Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page-mode writes simply delay the loading of the next access into the pipeline. These delays appear as ci states externally. Figure 9–20. shows the occurrence of a single bubble between column B and column C accesses.

9.7 Three-Cycles-per-Column Read

The state sequence for a three-cycles-per-column read cycle is shown in Figure 9–21.

Figure 9–21. Three-Cycles-per-Column Read States



The row access consists of the r1, r2, r3, r4, r5, r6, and possibly rspin states. The r3 and r6 states can be repeated by adding wait states. The r4 state provides additional time before the fall of $\overline{RAS}$ for address decode. The r5 state begins with the fall of $\overline{RAS}$ and extends the available $\overline{RAS}$ access time.

The column accesses consist of a three-stage pipeline. A new column access can begin every third cycle, and each access takes three cycles to complete so accesses never overlap. The pipeline stages are as follows:

c1 The address (A[31:0]) and processor activity code (STATUS[5:0]) for the current access are output.

c2 All $\overline{CAS}$/DQM strobes are activated. This latches the address into the memories.

c3 The READY input is sampled and if low, the c3 stage is repeated and READY sampled until it is sampled high. The data is latched by the TMS320C80.

ci Column idle stage. $\overline{CAS}$/DQM strobes are inactive and no data is latched. This stage occurs when no access is loaded in the pipeline.

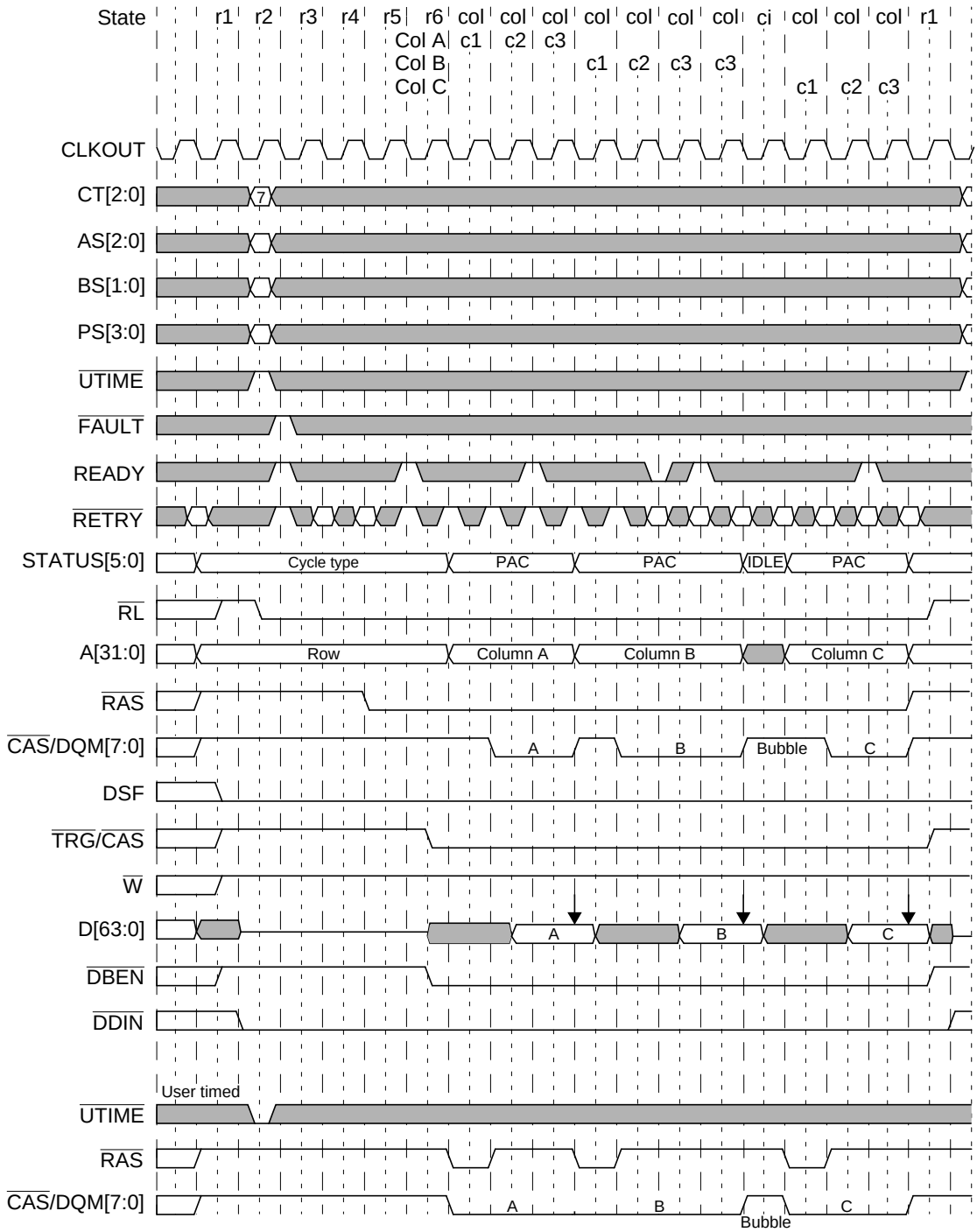
Once a column access begins, all pipeline stages will be completed.

After completing the last access, the memory interface will returns to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column pipeline sequence begins again.

9.7.1 Three-Cycles-per-Column Read Example

The example in Figure 9–22 shows a page-mode read of three column addresses. The $\overline{CAS}$/DQM labels indicate which column access corresponds to each strobe. In this example, the column B access shows an added column time wait state. The READY input is driven low by external logic at the start of the c3 stage. This causes the c3 stage to repeat at which time READY is sampled high, input data B is latched, and the pipeline advances to the next access.

Figure 9–22. Three-Cycles-per-Column Read



9.7.2 User-Timed Three-Cycles-Per-Column Read Example

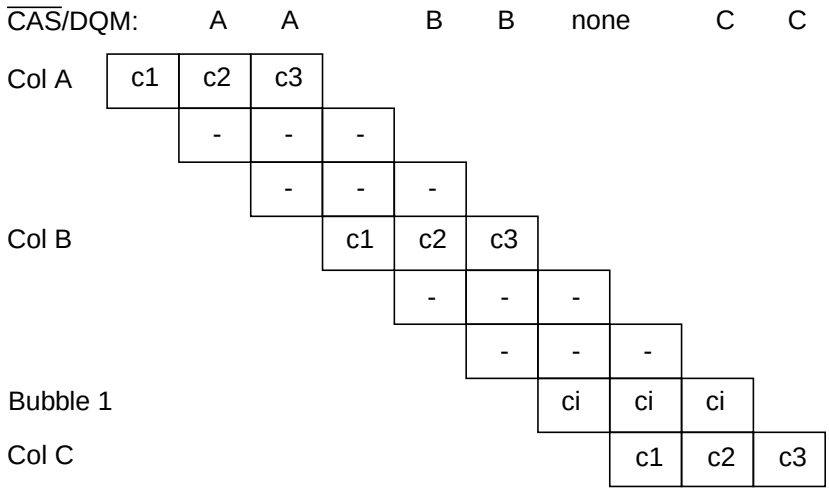
The bottom of Figure 9–22 shows how the cycle is modified when user timing is selected. Note that the pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition. $\overline{RAS}$ goes low for one cycle during each c1 stage to indicate the beginning of each column access.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide early indication of which bytes are being accessed. During read cycles, all $\overline{CAS}$/DQM strobes are active as 64 bits are always read. For three-cycles-per-column reads, the $\overline{CAS}$/DQM strobes are active during the c1, c2, and c3 stages including any added wait states, but are inactive during idle (ci) states.

9.7.3 Three-Cycles-per-Column Read With Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page-mode reads simply delay the loading of the next access into the pipeline as shown in Figure 9–23. These delays appear as ci states externally. Figure 9–22 shows the occurrence of a single bubble between column B and column C accesses.

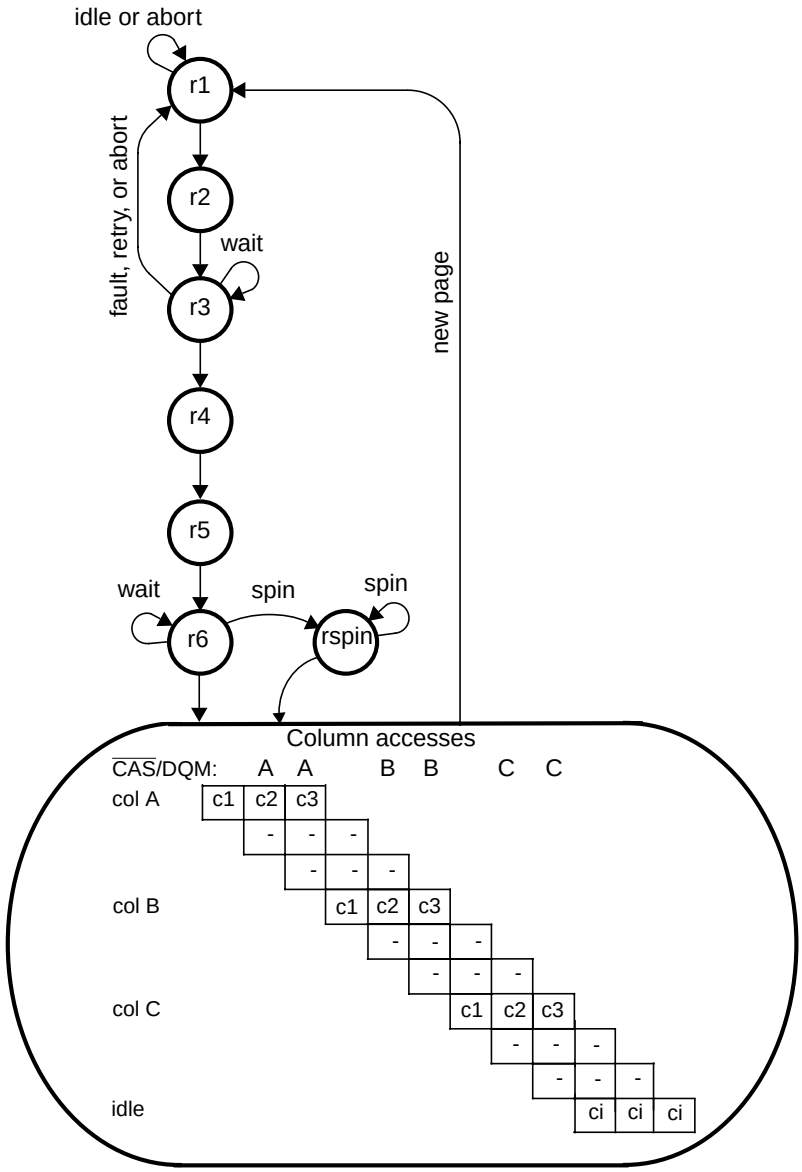
Figure 9–23. The Effect of Bubbles on Column Pipeline



9.8 Three-Cycles-per-Column Write

The state sequence for a three-cycles-per-column write cycle is shown in Figure 9–24.

Figure 9–24. Three-Cycles-per-Column Write States



The row access consists of the r1, r2, r3, r4, r5, r6, and possibly rspin states. The r3 and r6 states may be repeated by adding wait states.

The column accesses consist of a three-stage pipeline. A new column access may begin every third cycle. Each access requires three cycles to complete. The pipeline stages are as follows:

c1 The TC drives the address (A[31:0]) lines and processor activity code (STATUS[5:0]) for the current access. The TC also drives data on the appropriate bytes of the data bus (D[63:0]).

c2 Valid data (D[63:0]) continues to be driven and the $\overline{\text{CAS}}$/DQM strobes corresponding to the valid bytes are activated.

c3 Valid data (D[63:0]) and the $\overline{\text{CAS}}$/DQM strobes continue to be driven. The READY input is sampled and, if low, the pipeline stalls and the c3 state is repeated. Once READY is sampled high, the pipeline operation is resumed.

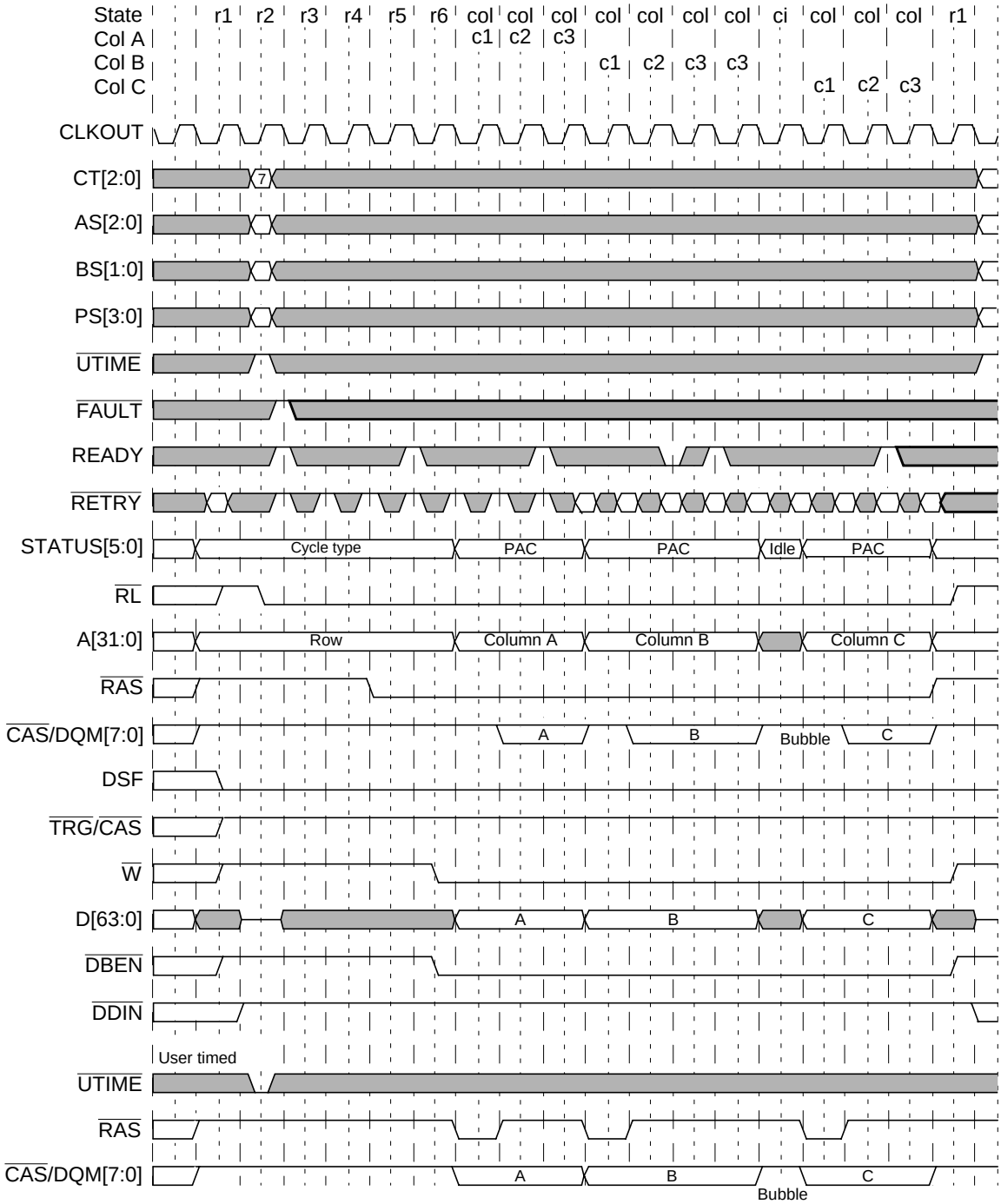
ci This stage (column idle) occurs when no new access is loaded into the pipeline. $\overline{\text{CAS}}$/DQM strobes are inactive and no new data is output.

After completing the last access, the memory interface either returns to the r1 state if a new row access is required, or enters the ci state to wait for the next access. If the next access does not require a new row address, then the column-pipeline sequence begins again.

9.8.1 Three-Cycles-per-Column Write Example

The example in Figure 9–25 shows a page-mode write to three column addresses. The $\overline{\text{CAS}}$/DQM labels indicate the access to which the strobes correspond. In this example, the column-B access is extended one cycle by the insertion of a column-time wait state. External logic drives READY low at the start of the c3 stage. The TMS320C80 samples READY low and causes the c3 stage to be repeated. At the start of the second c3 stage. READY has been driven high so the access completes. After completing the last write (column C) a new page access is required and the memory interface immediately returns to the r1 state.

Figure 9–25. Three-Cycles-per-Column Write



9.8.2 User-Timed Three-Cycles-per-Column Write Example

The bottom of Figure 9–25 shows how the cycle is modified when user timing is selected. The pipeline sequence does not change, only the way in which the $\overline{RAS}$ and $\overline{CAS}$/DQM signals transition changes. $\overline{RAS}$ goes low for one cycle during each column access.

The $\overline{CAS}$/DQM output timing is modified so that the $\overline{CAS}$/DQM strobes are active for the entire address time. This allows the $\overline{CAS}$/DQM strobes to provide an earlier indication of which bytes are being accessed. During write cycles, only those $\overline{CAS}$/DQM strobes corresponding to valid data bytes are active.

9.8.3 Three-Cycle-per-Column Write Pipeline Bubbles

Because the column accesses do not overlap, TC pipeline bubbles during page-mode writes simply delay the loading of the next access into the pipeline. These delays appear as ci states externally. Figure 9–25 shows the occurrence of a single bubble between column B and column C accesses.

Chapter 10

SDRAM Cycles

This chapter provides a description of bus cycles generated by the transfer controller when interfacing with SDRAM. This chapter is divided into four general sections. The first section describes how to select SDRAM operation and configure TMS320C80 SDRAM cycles to work with the selected device. The second section describes the sequence of operations required to transfer data with an SDRAM bank. The next section describes bus cycles that occur when reading and writing data. The final section describes special control cycles for performing power-up deactivation and setting up of the mode registers.

This chapter covers the following topics:

Topic **Page**

10.1 SDRAM-Specific Configuration

This section describes the implementation of the SDRAM interface on the TMS320C80. It discusses how to select SDRAM operation and configure TMS320C80 SDRAM cycles to work with the selected device.

10.1.1 Enabling SDRAM Support at Power Up

For the TMS320C80 to support synchronous DRAM, the external system must signal the presence of SDRAM. This is done by presenting one of the SDRAM CT codes to the TMS320C80 during the initial power-up DRAM initialization (refresh) sequence. The TC abandons the first refresh that receives an SDRAM CT code and sets an internal latch to indicate the presence of SDRAM in the system. The TC then performs a DCAB command to deactivate all SDRAM banks and continue with the refresh sequence.

The power-up refresh sequence comprises 32 refresh cycles. You should use one of the 5 LSBs of the refresh pseudo address (output on A[31:16]) to select between SDRAM and standard DRAM banks (if both are present) to ensure that all RAMs are properly initialized.

Once the initial refresh cycles have been completed, the TC performs an MRS cycle to initialize the SDRAM mode register.

10.1.2 Cycle Timing Codes for SDRAM Cycles

Six SDRAM timing configurations are available. The first six entries in Table 5–3 are for SDRAM cycles. The CT and $\overline{\text{UTIME}}$ code must be input at the beginning of each memory access. For SDRAMs this is just prior to the row-activate command. The TC then operates at the requested CAS latency and burst length for all subsequent commands until the current row is deactivated.

10.1.3 Setting the SDRAM Mode Register

If SDRAM is present, the TC sets the mode register after a hardware reset. External logic can detect the type of memory cycle in progress by decoding the STATUS[5:0] lines at the beginning of the cycle; a value of 001100b corresponds to a Mode Register Set (MRS) cycle. During an MRS cycle, the TC programs the mode register using the SDRAM address pins as shown in Table 10–1.

Table 10–1. MRS Value

SDRAM Address pin	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Meaning	Reserved			0	0	Read Latency			S/I	Burst Length		
320C80 Output Value	0	0	0	0	0	UTIME	$\overline{\text{UTIME}}$	$\overline{\text{UTIME}}$ &CT0†	0	0	0	CT1

† & indicates a logical AND.

As Table 10–1 shows, the CT code input at the beginning of the MRS cycle determines the value programmed into the SDRAM mode register. The CT code in conjunction with $\overline{\text{UTIME}}$ specifies burst length and CAS latency as follows:

CT0 and **$\overline{\text{UTIME}}$** together determine whether the TC should latch data two, three, or four cycles after the READ command. See Table 5–3 for various CT and $\overline{\text{UTIME}}$ combinations that support these CAS latencies.

CT1 determines burst length. A value of 0 sets a burst length of 1, and a value of 1 sets a burst length of 2. A burst length of 1 should only be specified for pipeline-architecture SDRAMs that support changing of the column address on every cycle.

CT2 must be set to 0 when responding to an MRS row-time status code.

The S/I value is always fixed at zero, so that the TC always uses the “serial” burst sequence. This allows either forward or reverse addressing, depending on the starting column address.

Because the MRS register is programmed through the SDRAM address inputs, the location of the MRS bits on the TMS320C80’s *logical* address bus depends on the selected bus size. Table 10–2 shows the alignment of the MRS value. The location of the MRS on the *physical* address depends on the selected address shift (AS[2:0]).

Table 10–2. MRS Value Alignment

Bus Size		Logical Address Bits															
BS1	**BS0**	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
0	0	X	X	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	X	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X
1	0	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X	X
1	1	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X	X	X

The TC assumes that all SDRAM banks have their mode registers set at the same time. The TC cannot perform multiple MRS cycles to different banks because it does not know how to decode the bank addresses. However, if multiple banks of SDRAM with different CAS latencies and burst lengths exist, the system can supply one CT code to the TC followed by a retry operation (using the $\overline{\text{RETRY}}$ pin) to force a second MRS cycle. During the second MRS, a different CT code can be specified for a different bank. When using this scheme, external logic is responsible for providing the bank-decode information to enable the correct banks for each MRS cycle.

Note:

An MRS is not performed after a return from a host request. If another device changes the mode register then, to ensure correct operation, it must reset the register to its previous value before returning the bus.

10.1.4 Loading the SGRAM Color Register

The special-register-set (SRS) cycle on the TMS320C80 is a special form of its load-color-register (LCR) cycle. The TC performs an SRS cycle whenever the external system inputs one of the SDRAM CT codes at the start of an LCR cycle. The cycle-type output on STATUS[5:0] during SRS cycles is identical to that of the LCR cycle (STATUS[5:0] = 001101).

The TC performs SRS cycles to load SGRAM color register prior to performing block writes. During an SRS cycle, the TC selects the register to be loaded using the SGRAM address pins as shown below. The SDRAM bank-select pin (BS) is always set to zero.

Table 10–3. Special Register Set Value

SDRAM Address pin	A8	A7	A6	A5	A4	A3	A2	A1	A0
Meaning	0	0	LC	LM	LS	Stop Register			
320C80 Output Value	0	0	1	0	0	0	0	0	0

In principle, SRS cycles may load either the color register (LC = 1), or the mask register (LM = 1). However, the TMS320C80 only supports loading of the color register for block-write usage.

The TMS320C80 supports block write for 64-bit busses only. This allows the BS[1:0] inputs to be redefined during block writes, so that they determine the type of block write rather than bus size. Thus the SRS value cannot be shifted for bus size but always appears starting with *logical* address bit A3 as shown in Table 10–4. As with the MRS value, the location on the *physical* address bus is dependent on the address shift (AS[2:0]) selected for the cycle.

Table 10–4. SRS Value Alignment

Logical Address Bits															
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
0	0	0	0	A8	A7	A6	A5	A4	A3	A2	A1	A0	X	X	X

An SRS cycle is not performed for simulated block writes; an SRS cycle is only performed if the value of BS [1] is 1 at the start of the cycle. For simulated block writes, the TC converts an SRS cycle into a standard write cycle.

10.1.5 Address Output

The TC normally outputs the full 32-bit address at row time, and then outputs a shifted address at column time to provide the correct multiplexed address for DRAMs or SDRAMs. The shift amount is from 8 to 14 bits depending on the value specified by the AS inputs. Because SDRAMs use the address inputs for control as well as address, this has the following system implications:

- The address line that corresponds to the SDRAM's Bank Select bit (A11 on 16Mbit SDRAM) must be latched externally by RL falling. This ensures that the bank select remains correct during READ and WRT commands.

- ❏ The system must force the address bit below bank select (A10 on 16Mbit) to be low unless RAS is active low. This disables the auto precharge from occurring following a READ or WRT command.
- ❏ During the DCAB cycles, the TMS320C80 outputs all 1s on A[3:10] to ensure that the A10 bit is 1 and both banks are deactivated.

10.2 SDRAM Bank Operation

The following access rules apply specifically to SDRAM:

- ❏ A bank must be activated with a row address before it may be accessed.
- ❏ A bank must be deactivated (precharged) before it may be activated with a different row address.

The SDRAM access sequence is:

1) Activate bank.
2) Perform burst read/writes on row.
3) Repeat (2) as needed.
4) Deactivate bank.
5) Go to step 1.

Because the transfer controller's memory interface is DRAM based, it follows a page-mode-access method. This page-mode operation has been adapted to the SDRAM operation by equating SDRAM commands with their page-mode counterparts. Row activation corresponds to a TMS320C80 row access at the start of a page. The burst reads and writes are similar to column-time accesses within the page. Bank deactivation is similar to the drain cycle that is performed at the end of pipelined writes prior to a page change.

10.2.1 CAS Latency

The TMS320C80 supports SDRAMs with CAS latencies of two, three, or four cycles. The latency is selected by $\overline{\text{UTIME}}$ and the CT code received by the TC at the beginning of the row-activate cycle. The TC will then adjust the way it samples input data to account for either the two-, three-, or four-clock delay.

10.2.2 Burst Length

The TMS320C80 supports read and write burst lengths of one and two depending on the type of SDRAM device being used. Because the TC has the ability to change column addresses on every cycle, a burst length of one is the most attractive. This allows the TC to access nonsequential columns within a row as may be required for packet transfers. However, because of the *2n rule* for certain SDRAM architectures (interleaved), the column address may be restricted by the SDRAM to only change once every other cycle. There is no advantage to a burst of one over a burst of two for these devices.

If the SDRAM uses a pipelined architecture that supports a new column address on every cycle, then the TC can be configured to use a burst length of 1. Otherwise the TC must be configured to use a burst length of two. The TC always uses the *serial* burst sequence. This allows either forward or reverse addressing depending on the starting column address. If a burst length of two is used and second-column access in the burst is not required, it is disabled via the $\overline{\text{CAS}}$/DQM pin(s).

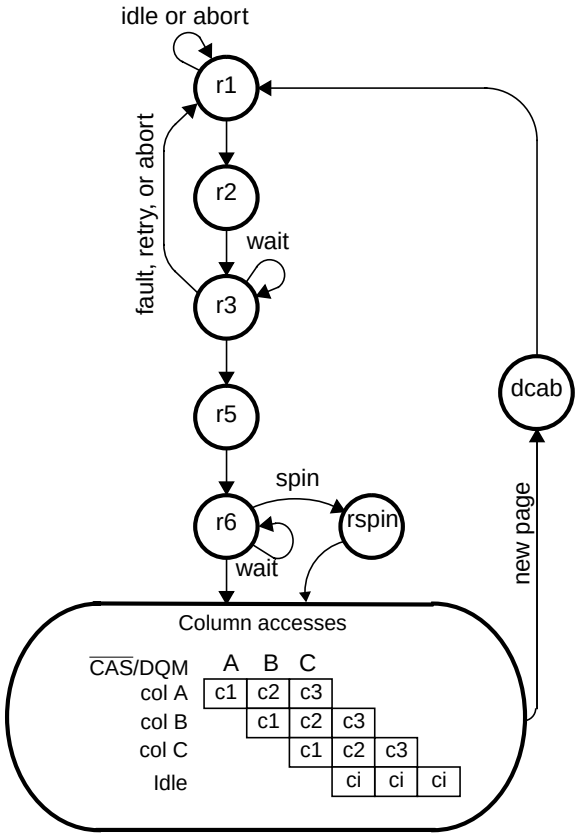
The burst length is selected by the value input on the CT[2:0] pins at the beginning of the row-activate cycle.

10.3 SDRAM Read and Write Cycles

10.3.1 SDRAM Read with CAS Latency 2 and Burst Length 1

Figure 10–1 shows the state sequence for an SDRAM read using a CAS latency of 2 and a burst length of 1. This cycle should only be selected for SDRAMs that support a new column address on every cycle.

Figure 10–1. State Sequence for SDRAM Read with CAS Latency 2, and Burst Length 1



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states.

The column accesses consist of a three-stage pipeline. A new column access may begin every cycle. Each access requires three cycles to complete. Pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0], and enables all the bytes by activating the $\overline{\text{CAS}}$/DQM[7:0] lines. The $\overline{\text{TRG}}$/$\overline{\text{CAS}}$ strobe is activated and this latches the address into the SDRAM.

c2 This stage does not affect bus activity; that depends on other stages concurrently being performed, which in turn depends on the next access in the pipeline. If the next access is on the same page as the current one, the c1 stage of the next access is concurrently performed with the c2 stage for this access. If the next access is on a new page, a DCAB command is concurrently performed (see state dcab, below). If no access is loaded into the pipeline an idle state is concurrently performed (see state ci, below).

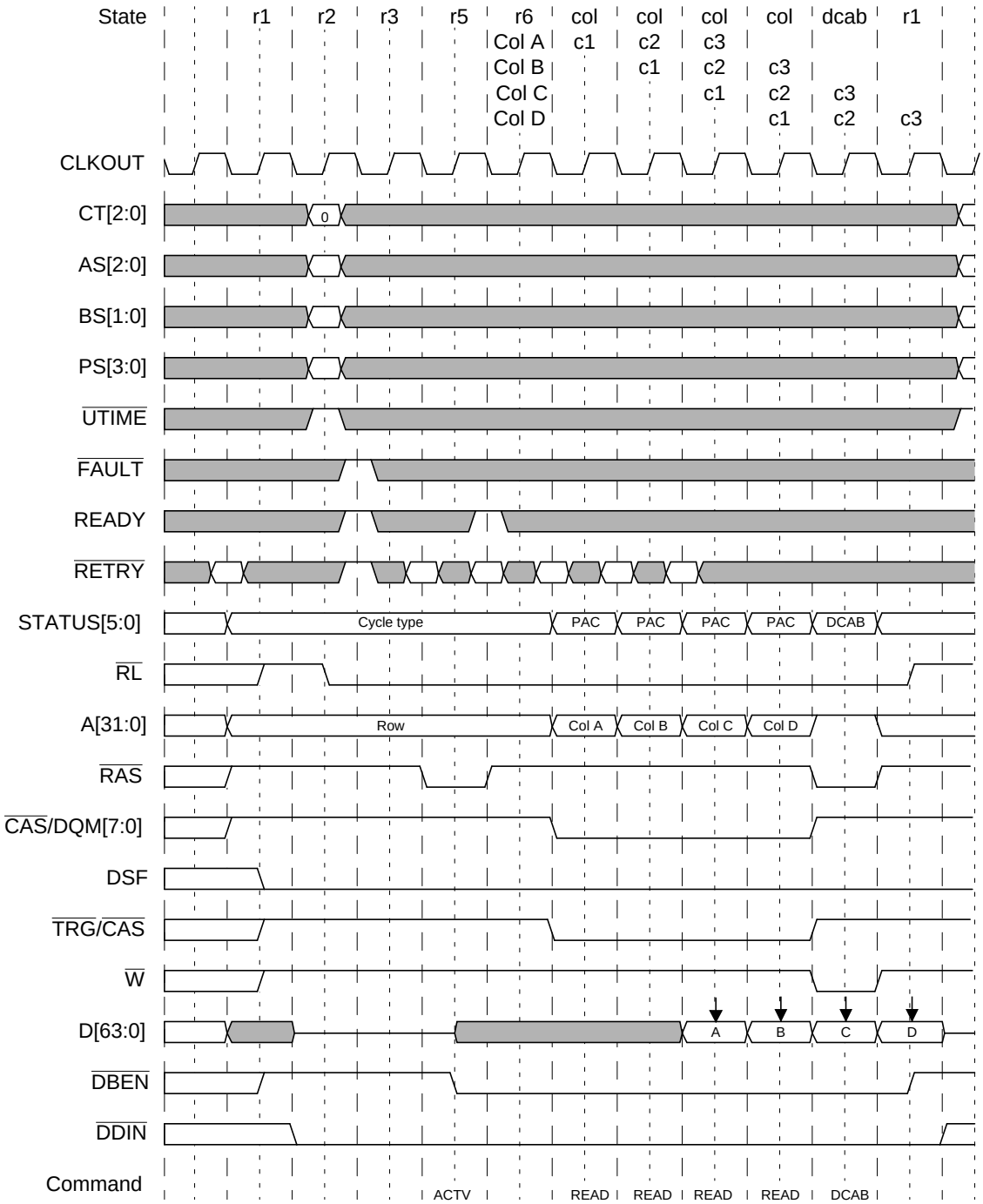
c3 The TC reads the data from D[63:0].

ci The column-idle stage, ci, occurs when no access is loaded into the pipeline. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates $\overline{\text{TRG}}$/$\overline{\text{CAS}}$. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed to deactivate the bank. The TC sets all the A[31:0] and $\overline{\text{CAS}}$/DQM lines high, and deactivates $\overline{\text{TRG}}$/$\overline{\text{CAS}}$. Also, $\overline{\text{W}}$ and $\overline{\text{RAS}}$ are activated and the STATUS[5:0] lines set to 111111b to indicate a DCAB cycle.

The timing diagram in Figure 10−2 shows an example of an SDRAM read sequence using a CAS latency of 2 and a burst length of 1. The CT inputs, in conjunction with $\overline{\text{UTIME}}$, configure the latency and burst length in state r2. The TC activates the selected bank with the row address in state r5. In this example, four read commands are performed to four different column addresses. The two-cycle latency causes data to appear two cycles after the corresponding column address. Following the last column access, the TC performs a DCAB cycle to deactivate the bank. The transfer of data completes during the DCAB and the following r1 state.

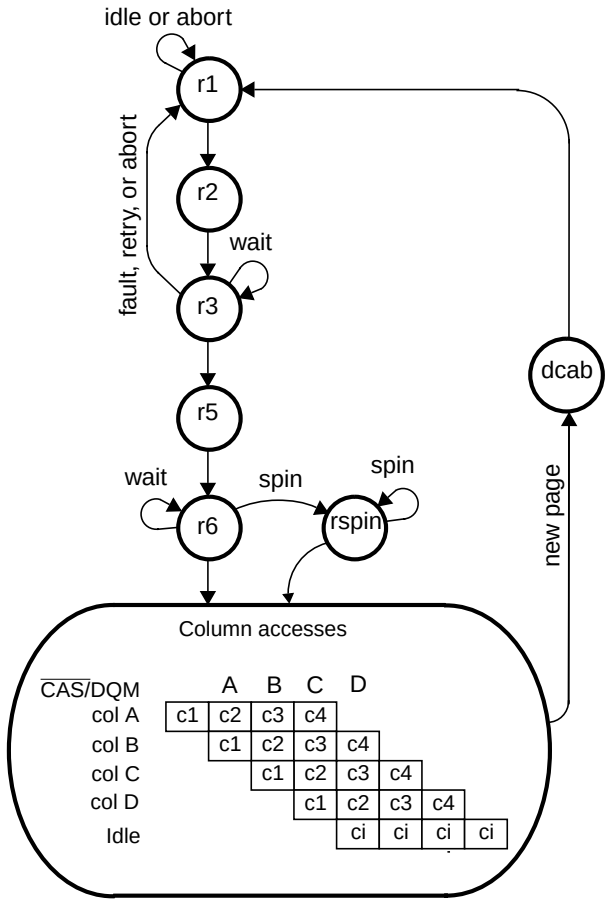
Figure 10–2. SDRAM Read with CAS Latency 2 and Burst Length 1



10.3.2 SDRAM Read with CAS Latency 3 and Burst Length 1

Figure 10–3 shows the state sequence for an SDRAM read using a CAS latency of 3 and a burst length of 1. This cycle should only be selected for SDRAMs that support a new column address on every cycle.

Figure 10–3. State Sequence for SDRAM Read with CAS Latency 3 and Burst Length 1



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states.

The column accesses consist of a four-stage pipeline. A new column access may begin every cycle. Each access requires four cycles to complete. Pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0], and activates the $\overline{TRG}/\overline{CAS}$ strobe. The $\overline{TRG}/\overline{CAS}$ strobe latches the address into the SDRAM.

c2 The TC enables all the bytes by activating $\overline{CAS}$/DQM[7:0].

c3 This stage does not affect bus activity; that depends on other stages concurrently being performed.

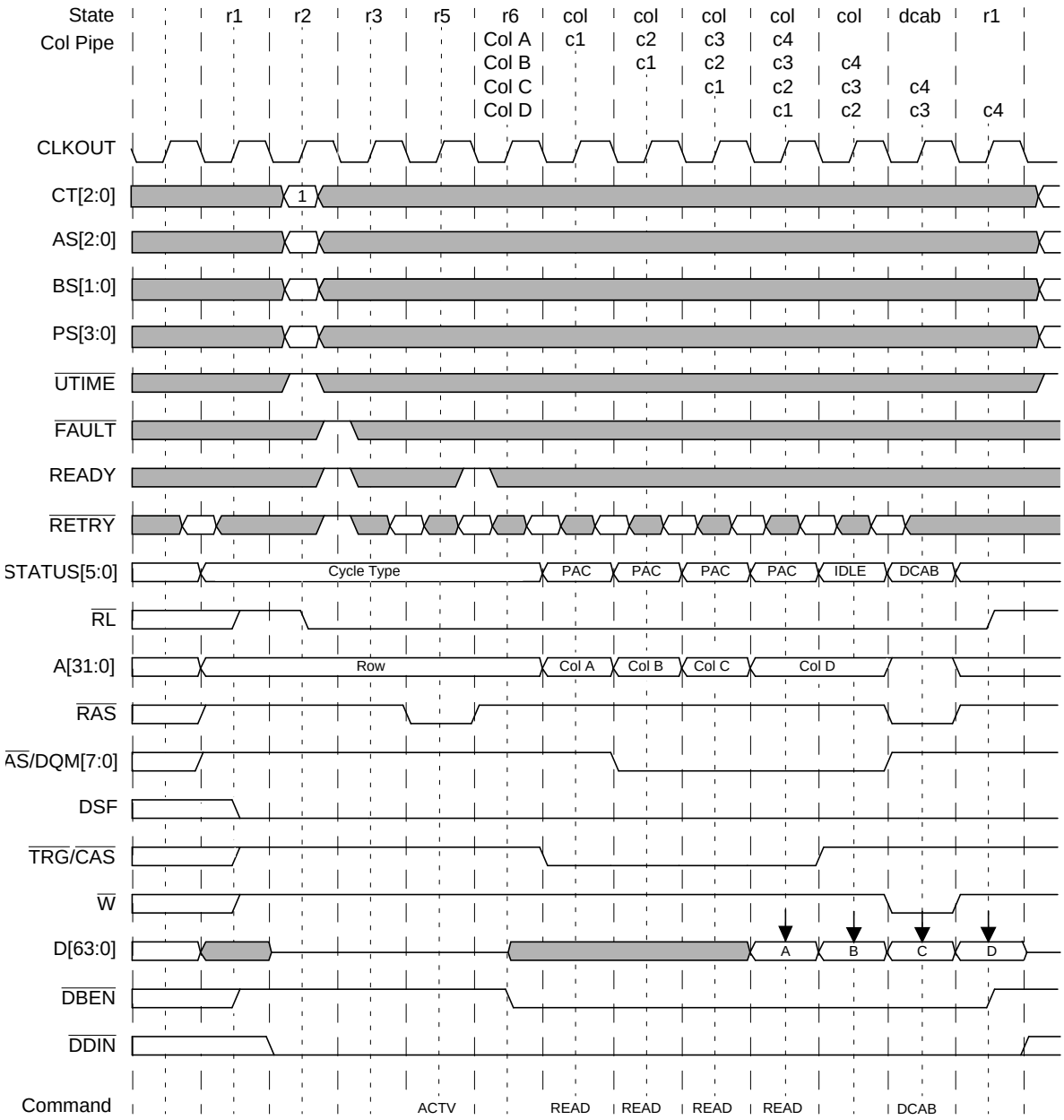
c4 The TC reads the data from D[63:0].

ci The column-idle stage, ci, occurs after the last access has been loaded into the pipeline. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{TRG}/\overline{CAS}$ strobe. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed after the column-idle cycle that followed the final column access. The TC sets all the A[31:0] and $\overline{CAS}$/DQM lines high, and deactivates the $\overline{TRG}/\overline{CAS}$. Also, $\overline{W}$ and $\overline{RAS}$ are activated and the STATUS[5:0] lines set to 111111b to indicate a DCAB cycle.

The timing diagram in Figure 10–4 shows an example of an SDRAM read sequence using a CAS latency of 3 and a burst length of 1. The CT and $\overline{UTIME}$ inputs in state r2 configure the latency and burst length. The TC activates the selected bank with the row address in state r5. In this example, four read commands are performed to four different column addresses. The three-cycle latency causes data to appear three cycles after the corresponding column address. Following the last column access, an idle cycle must be inserted to meet SDRAM requirements. A DCAB cycle is then performed to deactivate the bank. The transfer of data completes during the idle, DCAB, and following r1 state.

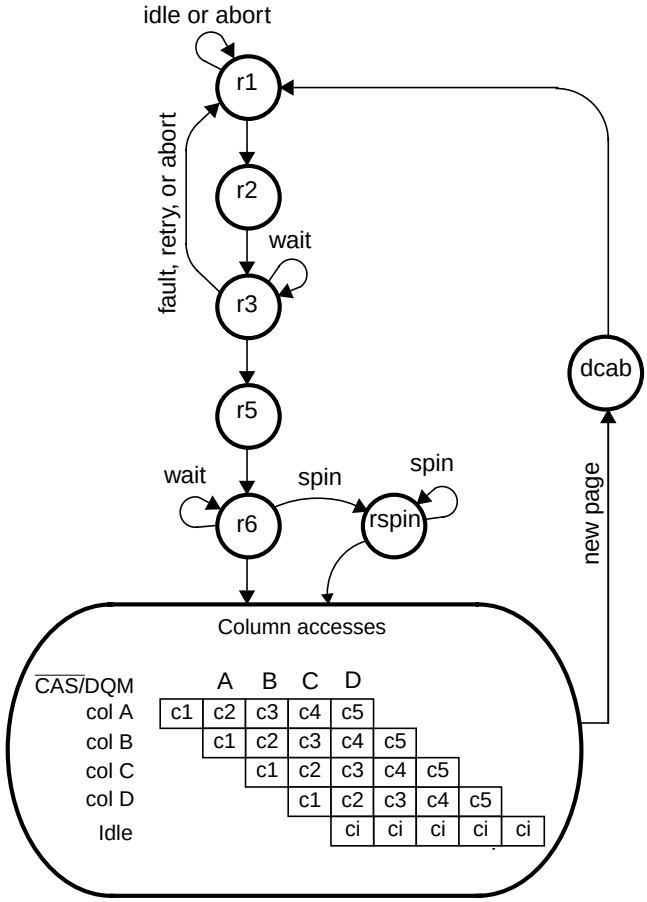
Figure 10–4. SDRAM Read with CAS Latency 3, Burst Length 1



10.3.3 SDRAM Read with CAS Latency 4 and Burst Length 1

Figure 10–5 shows the state sequence for an SDRAM read using a CAS latency of 4 and a burst length of 1. This cycle should only be selected for SDRAMs that support a new column address on every cycle.

Figure 10–5. State Sequence for SDRAM Read with CAS Latency 4 and Burst Length 1



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states.

The column accesses consist of a five-stage pipeline. A new column access may begin every cycle. Each access requires five cycles to complete. Pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0], and activates the $\overline{\text{TRG}}/\overline{\text{CAS}}$ strobe. The $\overline{\text{TRG}}/\overline{\text{CAS}}$ strobe latches the address into the SDRAM.

c2 The TC enables all the bytes by activating $\overline{\text{CAS}}$/DQM[7:0].

c3, c4 These stages do not affect bus activity. Such activity depends on other stages concurrently occurring in the column pipeline.

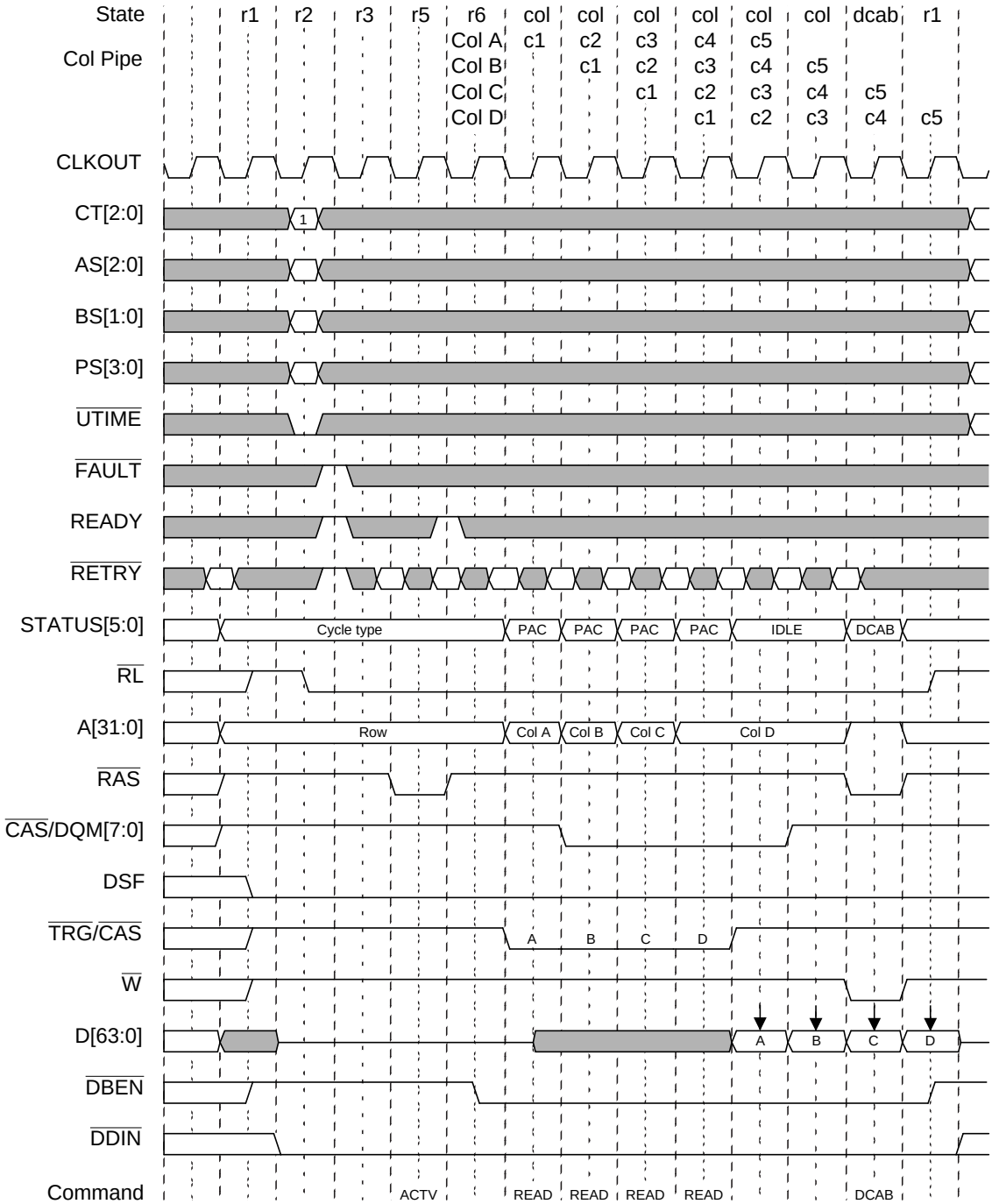
c5 The TC reads the data from D[63:0].

ci The column-idle stage, ci, occurs when no access is loaded in the pipeline. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{\text{TRG}}/\overline{\text{CAS}}$ strobe. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed after the column-idle cycle that followed the final column access. The TC sets all the A[31:0] and $\overline{\text{CAS}}$/DQM lines high, and deactivates the $\overline{\text{TRG}}/\overline{\text{CAS}}$ lines. Also, $\overline{\text{W}}$ and $\overline{\text{RAS}}$ are activated and the STATUS[5:0] lines set to 111111b to indicate a DCAB cycle.

The timing diagram in Figure 10–6 shows an example of an SDRAM read sequence using a CAS latency of 4 and a burst length of 1. The $\overline{\text{UTIME}}$ and CT inputs in state r2 configure the latency and burst length. The TC activates the selected bank with the row address in state r5. In this example, four read commands are performed to four different column addresses. The four-cycle latency causes data to appear four cycles after the corresponding column address. Following the last column access, an idle cycle must be inserted to meet SDRAM requirements. A DCAB cycle is then performed to deactivate the bank. Note that the last data transfer (column D in this example) is completed during the ‘r1’ state of the next access.

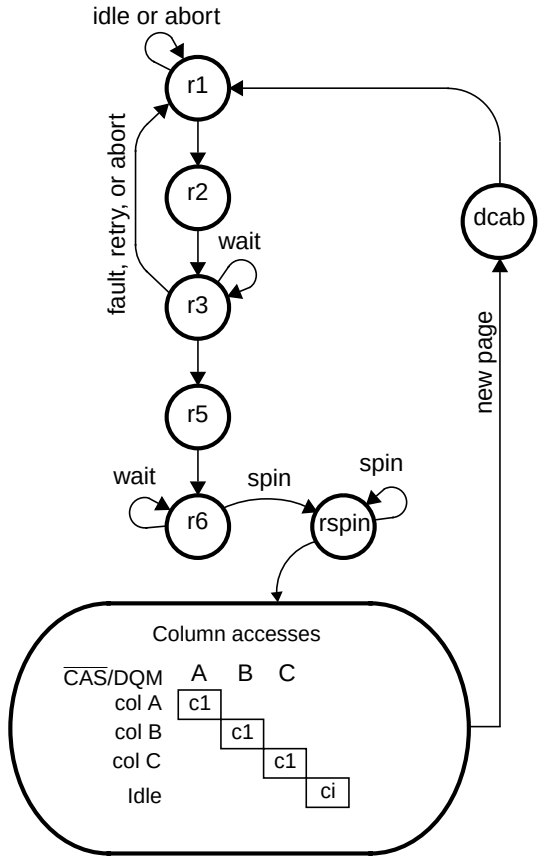
Figure 10–6. SDRAM Read with CAS Latency 4, Burst Length 1



10.3.4 SDRAM Write with Burst Length 1

Figure 10−7 shows the state sequence for an SDRAM write using a burst length of 1.

Figure 10−7. State Sequence for SDRAM Write with Burst Length 1



The row access consists of the r1, r2, r3, r5, r6, and rspin states. The r3 and r6 states may be repeated by adding wait states. The TC activates the selected bank with the row address in state r5. The rspin state always occurs at least once for SDRAM burst writes.

The column accesses consist of a single-stage pipeline. A new column access may begin every cycle. There is no latency on writes, so data is output on the same cycle as the column address. The pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0] and activates $\overline{TRG}$/$\overline{CAS}$. The TC also drives the valid data onto the data bus (D[63:0]) and disables writes to invalid bytes by deactivating the appropriate $\overline{CAS}$/DQM outputs. The $\overline{W}$ signal is set low to perform a write command.

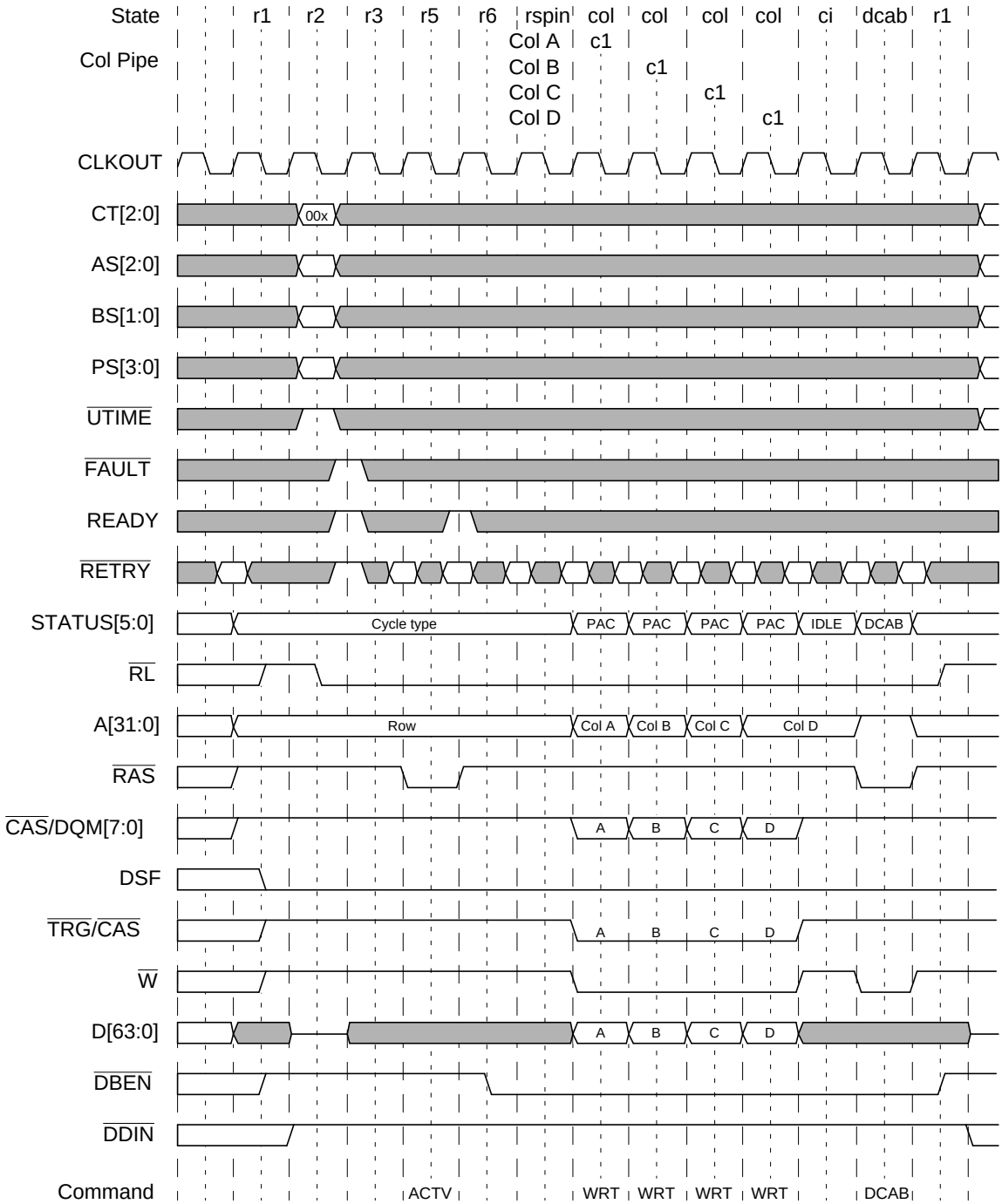
ci The column-idle stage, ci, occurs when no access is loaded into the pipeline. Also, at least one ci stage is inserted after the final column-write access on the current page. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{TRG}$/$\overline{CAS}$ strobe. No data is output.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed at least one column-idle cycle after the final column access. The TC sets all the A[31:0] lines and $\overline{CAS}$ /DQM lines high, and deactivates $\overline{TRG}$/$\overline{CAS}$. Also, $\overline{W}$ and $\overline{RAS}$ are activated and the STATUS[5:0] lines are set to 111111b to indicate a DCAB operation.

A burst length of 1 SDRAM write is selected by a CT code of 00x (CT0 determines CAS latency and is ignored for writes) at the start of the access. This cycle should only be selected for SDRAMs that support a new column address on every cycle.

Figure 10–8 shows a write to four columns. Following the final write command, an idle cycle is inserted (to meet SDRAM timing requirements). The bank is then deactivated with a DCAB command and the memory interface can begin a new page access.

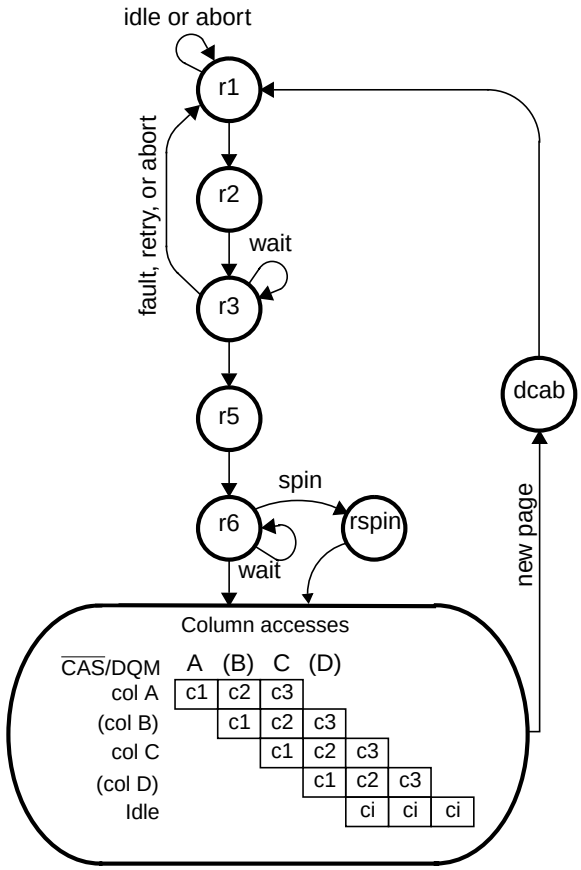
Figure 10–8. SDRAM Burst Length 1 Write



10.3.5 SDRAM Read with CAS Latency 2 and Burst Length 2

Figure 10–9 shows the state sequence for an SDRAM read using a CAS latency of 2 and a burst length of 2.

Figure 10–9. State Sequence for SDRAM Read with CAS Latency of 2 and Burst Length 2



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states. The TC activates the selected bank with the row address in state r5.

The column accesses consist of a three-stage pipeline. A new column-read command may begin every other cycle. Each access requires three cycles to complete.

Column-pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0], enables all the bytes by activating the $\overline{CAS}$/DQM[7:0] lines. The $\overline{TRG}$/$\overline{CAS}$ strobe is activated and this latches the address into the SDRAM.

c2 If the TC does not require the data from the second column in the burst, it sets the $\overline{CAS}$/DQM strobes high. The TC sets the $\overline{TRG}$/$\overline{CAS}$ signal high during this stage, to ensure that no column address is latched into the SDRAM until the start of the next burst transfer.

c3 The TC reads the data from D[63:0].

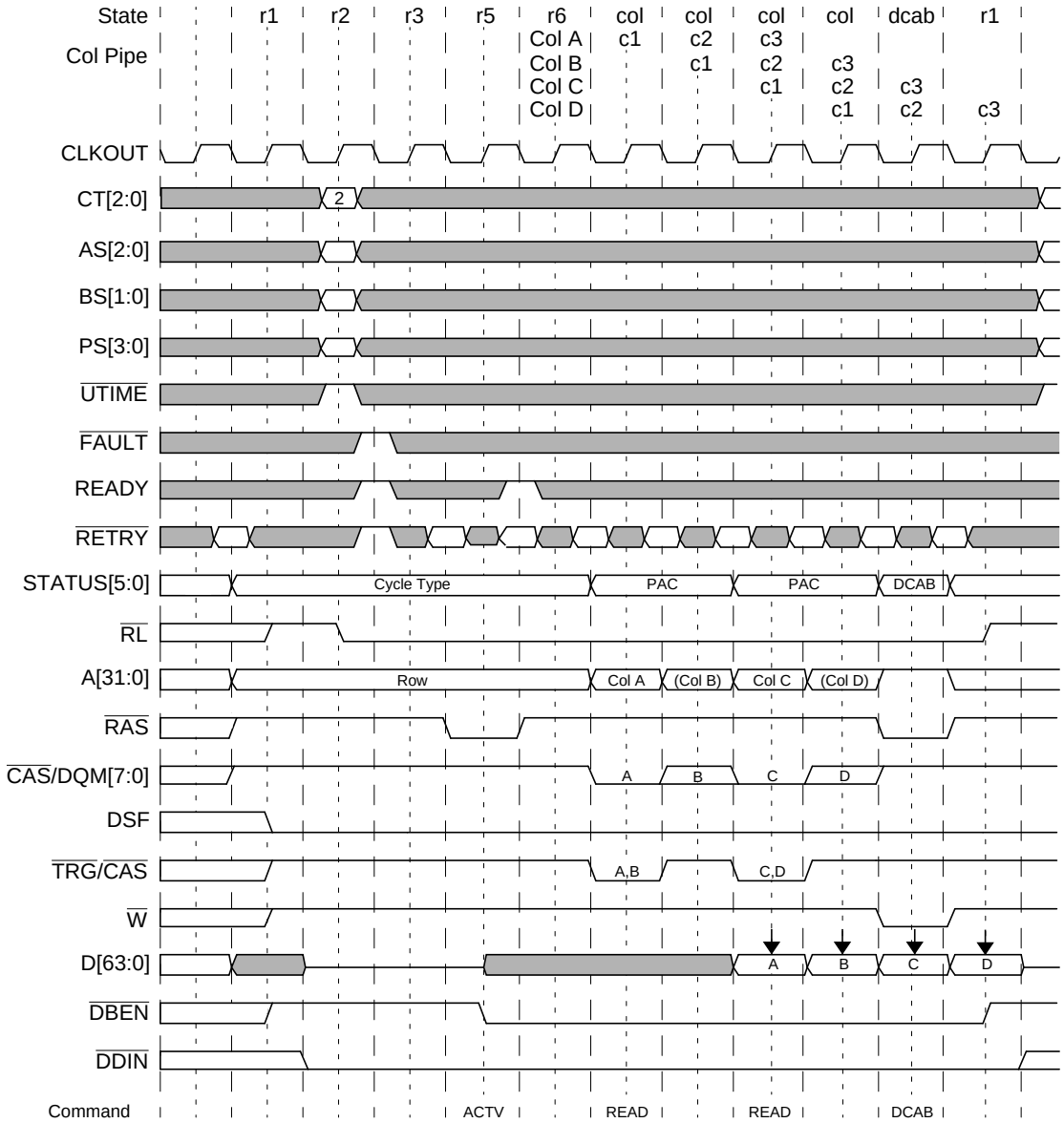
ci The column-idle stage, ci, occurs when no access is loaded into the pipeline. The ci stage is inserted into the pipeline in the same place that the c1 stage for the start of a burst transfer would have been inserted had it been present. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{TRG}$/$\overline{CAS}$ strobe. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed after the c2 stage of the final burst transfer or any column-idle cycle that may have followed the final column access. The TC sets all the A[31:0] and $\overline{CAS}$/DQM lines high, and deactivates the $\overline{TRG}$/$\overline{CAS}$. Also, $\overline{W}$ and $\overline{RAS}$ are activated and the STATUS[5:0] lines are set to 111111b to indicate a DCAB operation.

A CT[2:0] input of 010 and $\overline{UTIME}$ high selects an SDRAM read sequence using a CAS latency and burst length of two, as shown in Figure 10–10. In this example, two read commands are performed with each command causing a burst of two sequential column locations to be output by the SDRAM. The two-cycle latency causes data to appear two cycles after the corresponding column address. The transfer controller may output the addresses for the second column in the burst (column B and D), but these addresses are ignored by the SDRAMs since no new command is being issued during the state, and the $\overline{RAS}$ and $\overline{TRG}$/$\overline{CAS}$ strobes are inactive. If the TC does not require the data from the second part of the burst, the SDRAM output is disabled by driving the $\overline{CAS}$/DQM strobes high on the cycle following the read command (marked as B and D in Figure 10–10).

After completing the read commands, the TC deactivates the SDRAM banks using the DCAB command. Data transfer completes during the DCAB and following r1 state.

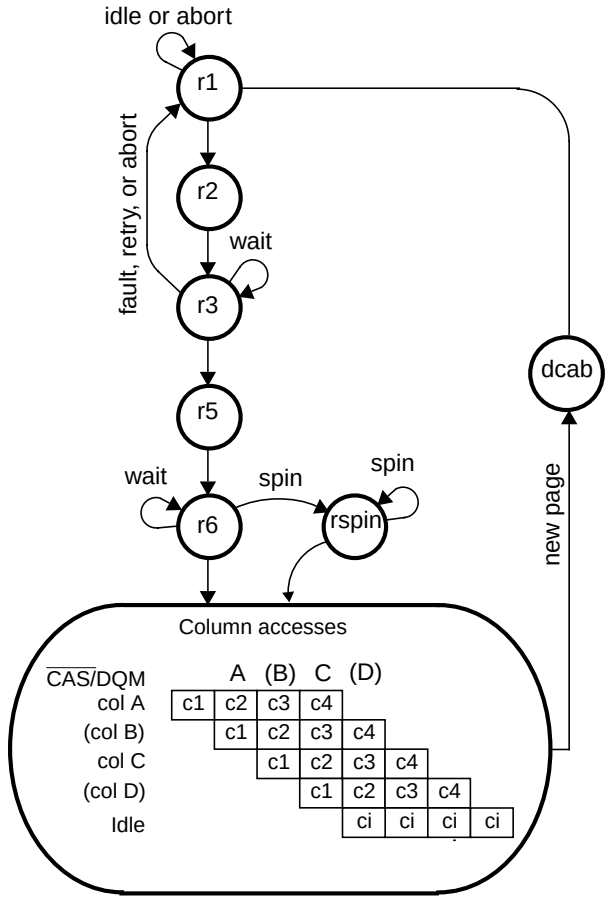
Figure 10–10. SDRAM Read with CAS Latency 2 and Burst Length 2



10.3.6 SDRAM Read with CAS Latency 3 and Burst Length 2

Figure 10–11 shows the state sequence for an SDRAM read using a CAS latency of 2 and a burst length of 2.

Figure 10–11. State Sequence for SDRAM Read with CAS Latency 3 and Burst Length 2



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states. The TC activates the selected bank with the row address in state r5.

The column accesses consist of a four-stage pipeline. A new column-read command may begin every other cycle. Each access requires four cycles to complete. The pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0]. The $\overline{TRG}/\overline{CAS}$ strobe is activated and latches the address into the SDRAM.

c2 The TC enables all the bytes for the first column in the burst, by setting the $\overline{CAS}$/DQM[7:0] lines low. The TC sets the $\overline{TRG}/\overline{CAS}$ signal high during this stage, to ensure that no column address is latched into the SDRAM until the start of the next burst transfer.

c3 If the TC does not require the data from the second column in the burst, it sets the $\overline{CAS}$/DQM strobes high.

c4 The TC reads the data from D[63:0].

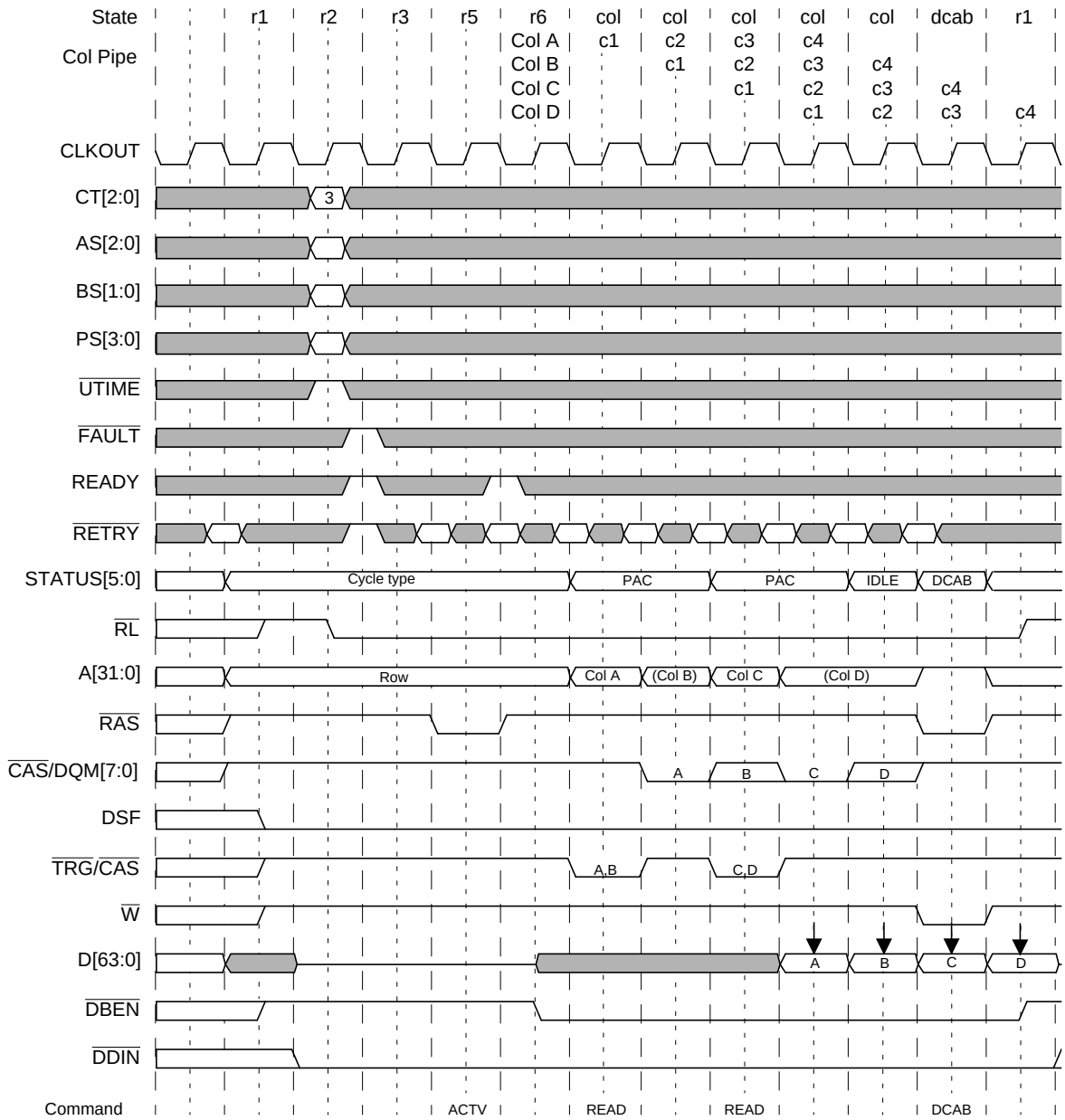
ci The column-idle stage, ci, occurs when no access is loaded into the pipeline. The ci stage is inserted into the pipeline in the same place that the c1 stage for the start of a burst transfer would have been inserted had it been present. Also, at least one ci stage will be inserted after the c2 stage of the final column-read command on the current page. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{TRG}/\overline{CAS}$ strobe. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed at least one column-idle cycle after the c2 stage of the final burst transfer. The TC sets all the A[31:0] and $\overline{CAS}$/DQM high, and deactivates the $\overline{TRG}/\overline{CAS}$. Also, $\overline{W}$ and $\overline{RAS}$ are activated, and the STATUS[5:0] lines are set to 111111b indicate a DCAB operation.

A CT[2:0] input of 011 and $\overline{UTIME}$ high selects an SDRAM read cycle with CAS latency 3 and burst length 2, as shown in Figure 10–12. Each read command causes a burst of two sequential column locations to be output by the SDRAM. The three-cycle latency causes data to appear 3 cycles after the corresponding column address. The transfer controller may output the addresses for the second column in the burst (column B and D) but these addresses are ignored by the SDRAMs since no new command is being issued during the state, and the $\overline{RAS}$ and $\overline{TRG}/\overline{CAS}$ signals are inactive. If the TC does not require the data from the second half of the burst, the SDRAM output is disabled by driving the $\overline{CAS}$/DQM strobes high two cycles after the read command.

After completing the read commands, an idle cycle is inserted to meet SDRAM timing requirements. The SDRAM banks are then deactivated by the DCAB command. Data transfer completes during the idle, DCAB, and following r1 state.

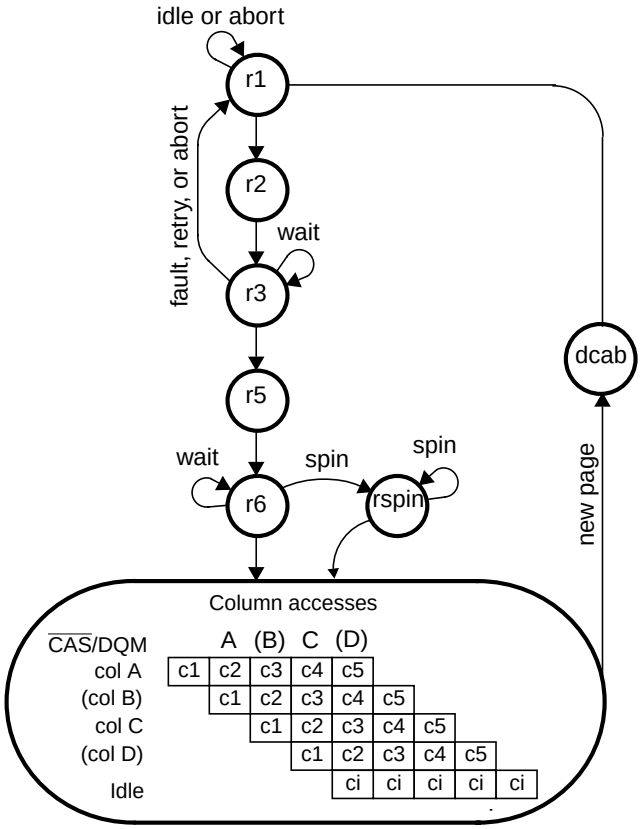
Figure 10–12. SDRAM Read with CAS Latency 3 and Burst Length 2



10.3.7 SDRAM Read with CAS Latency 4 and Burst Length 2

Figure 10–13 shows the state sequence for an SDRAM read using a CAS latency of 4 and a burst length of 2.

Figure 10–13. State Sequence for SDRAM Read with CAS Latency 4 and Burst Length 2



The row access consists of the r1, r2, r3, r5, and r6 states. The r3 and r6 states may be repeated by adding wait states. The TC activates the selected bank with the row address in state r5.

The column accesses consist of a five-stage pipeline. A new column-read command may begin every other cycle. Each access requires five cycles to complete. The pipeline stages are as follows:

c1 The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0], and activates the $\overline{\text{TRG}}/\overline{\text{CAS}}$ strobe. The $\overline{\text{TRG}}/\overline{\text{CAS}}$ strobe latches the address into the SDRAM.

c2 The TC enables all the bytes for the first column in the burst, by setting the $\overline{CAS}$/DQM[7:0] lines low. The TC sets the $\overline{TRG}$/$\overline{CAS}$ signal high during this stage, to ensure that no column address is latched into the SDRAM until the start of the next burst transfer.

c3 If the TC does not require the data from the second column in the burst, it sets the $\overline{CAS}$/DQM strobes high.

c4 This stage does not affect bus activity. Such activity depends on other stages concurrently occurring in the column pipeline.

c5 The TC reads the data from D[63:0].

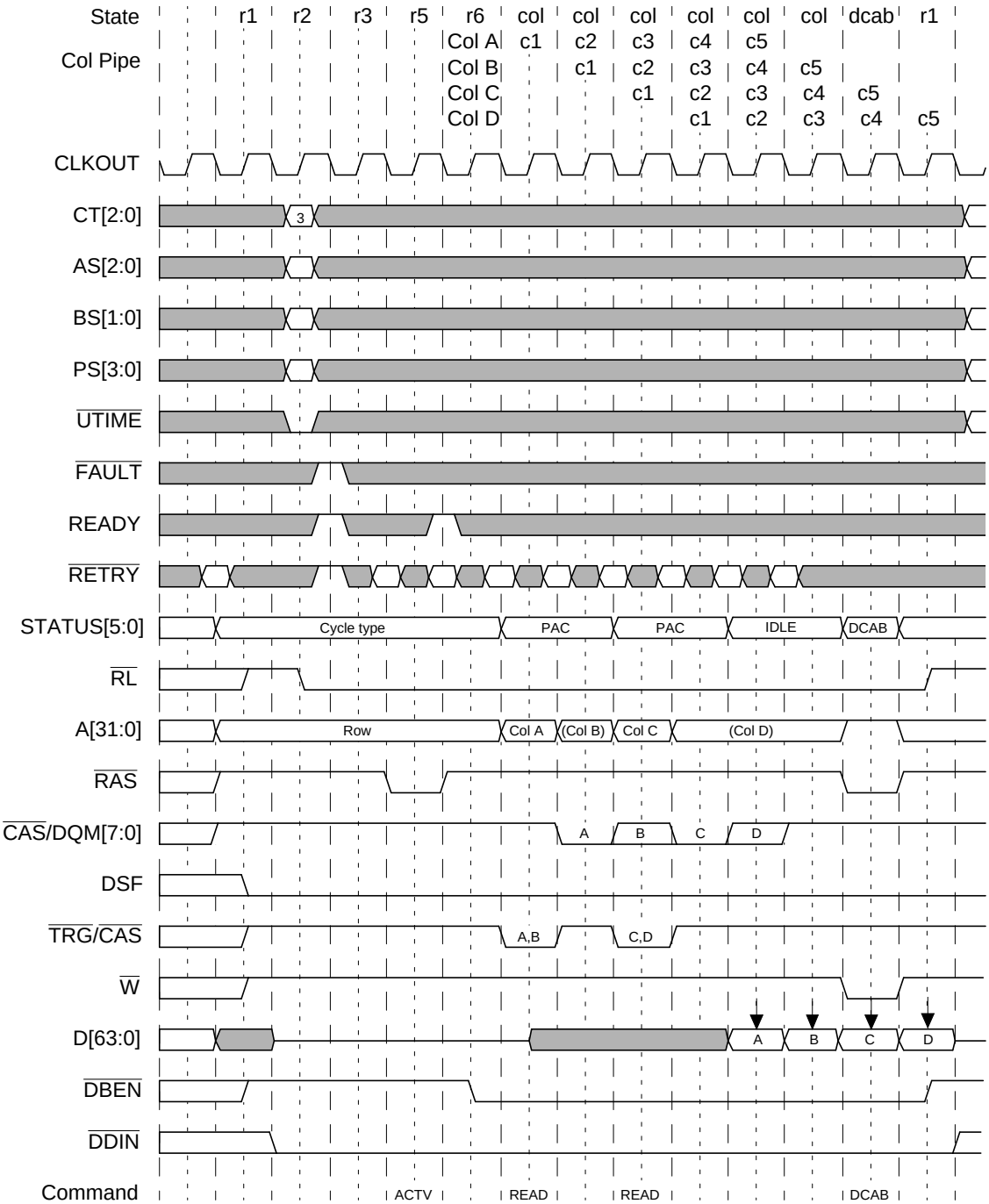
ci The column-idle stage, ci, occurs when no access is loaded into the pipeline. The ci stage is inserted into the pipeline in the same place that the c1 stage for the start of a burst transfer would have been inserted had it been present. Also, at least one ci stage is inserted after the c2 stage of the final column-read command on the current page. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b and deactivating the $\overline{TRG}$/$\overline{CAS}$ strobe. No data is latched.

dcab DCAB stage. If the next access is on a new page, a DCAB cycle is performed at least one column-idle cycle after the c2 stage of the final burst transfer. The TC sets all the A[31:0] and $\overline{CAS}$/DQM lines high, and deactivates the $\overline{TRG}$/$\overline{CAS}$. Also, $\overline{W}$ and $\overline{RAS}$ are activated and the STATUS[5:0] lines are set to 111111b to indicate a DCAB operation.

A CT[2:0] input of 011 and $\overline{UTIME}$ low selects an SDRAM read cycle with CAS latency 4 and burst length 2, as shown in Figure 10–14. Each read command causes a burst of two sequential column locations to be output by the SDRAM. The four-cycle latency causes data to appear 4 cycles after the corresponding column address. The TC may output the addresses for the second column in the burst (column B and D) but these addresses are ignored by the SDRAMs since no new command is being issued during the state, and the $\overline{RAS}$ and $\overline{TRG}$/$\overline{CAS}$ signals are inactive. If the TC does not require the data from the second half of the burst, the SDRAM output is disabled by driving the $\overline{CAS}$/DQM strobes high two cycles after the read command.

After completing the read commands, an idle cycle is inserted to meet SDRAM timing requirements. The SDRAM banks are then deactivated by the DCAB command. The last data transfer is completed during the r1 stage of the next access.

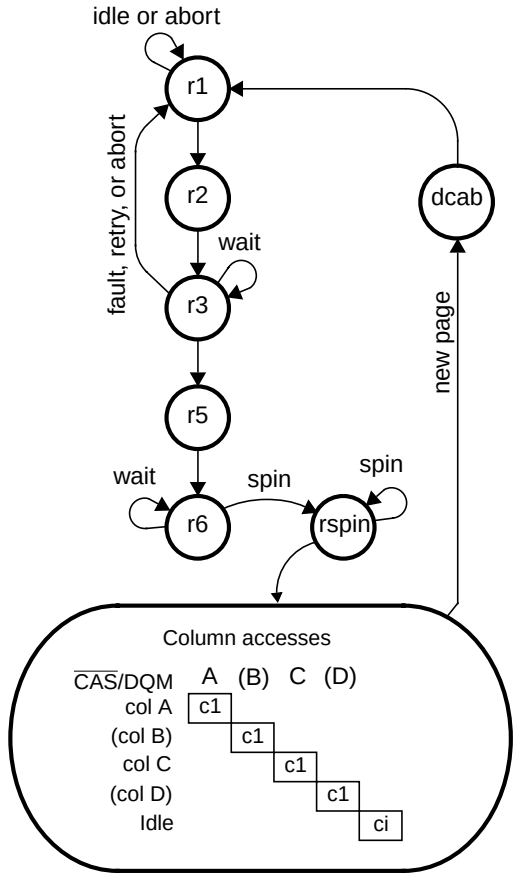
Figure 10–14. SDRAM Read With CAS Latency 4 and Burst Length 2



10.3.8 SDRAM Write with Burst Length 2

Figure 10–15 shows the state sequence for an SDRAM write using a burst length of 2.

Figure 10–15. State Sequence for SDRAM Write with Burst Length 2



The row access consists of the r1, r2, r3, r5, r6, and rspin states. The r3 and r6 states may be repeated by adding wait states. The TC activates the selected bank with the row address in state r5. The rspin state always occurs at least once for SDRAM burst writes.

The write command is broken into two column accesses, each consisting of a single-stage pipeline. Each column access takes one clock cycle to complete. There is no latency on writes, so data is output on the same cycle as the column address. Since a burst length 2 write command results in two column accesses, the command takes two clock cycles to complete. The pipeline stages are as follows:

c1 **First-column access.** The TC drives the column address onto A[31:0], the processor activity code onto STATUS[5:0] and activates $\overline{TRG}/\overline{CAS}$. The TC also drives the valid data onto the data bus (D[63:0]) and disables writes to invalid bytes by deactivating the appropriate $\overline{CAS}$/DQM outputs. The $\overline{W}$ signal is set low to perform a write command.

c1 **Second-column access.** As with the first column access in the burst transfer, the TC drives the valid data onto the data bus (D[63:0]); using the appropriate $\overline{CAS}$/DQM outputs to disable invalid bytes.

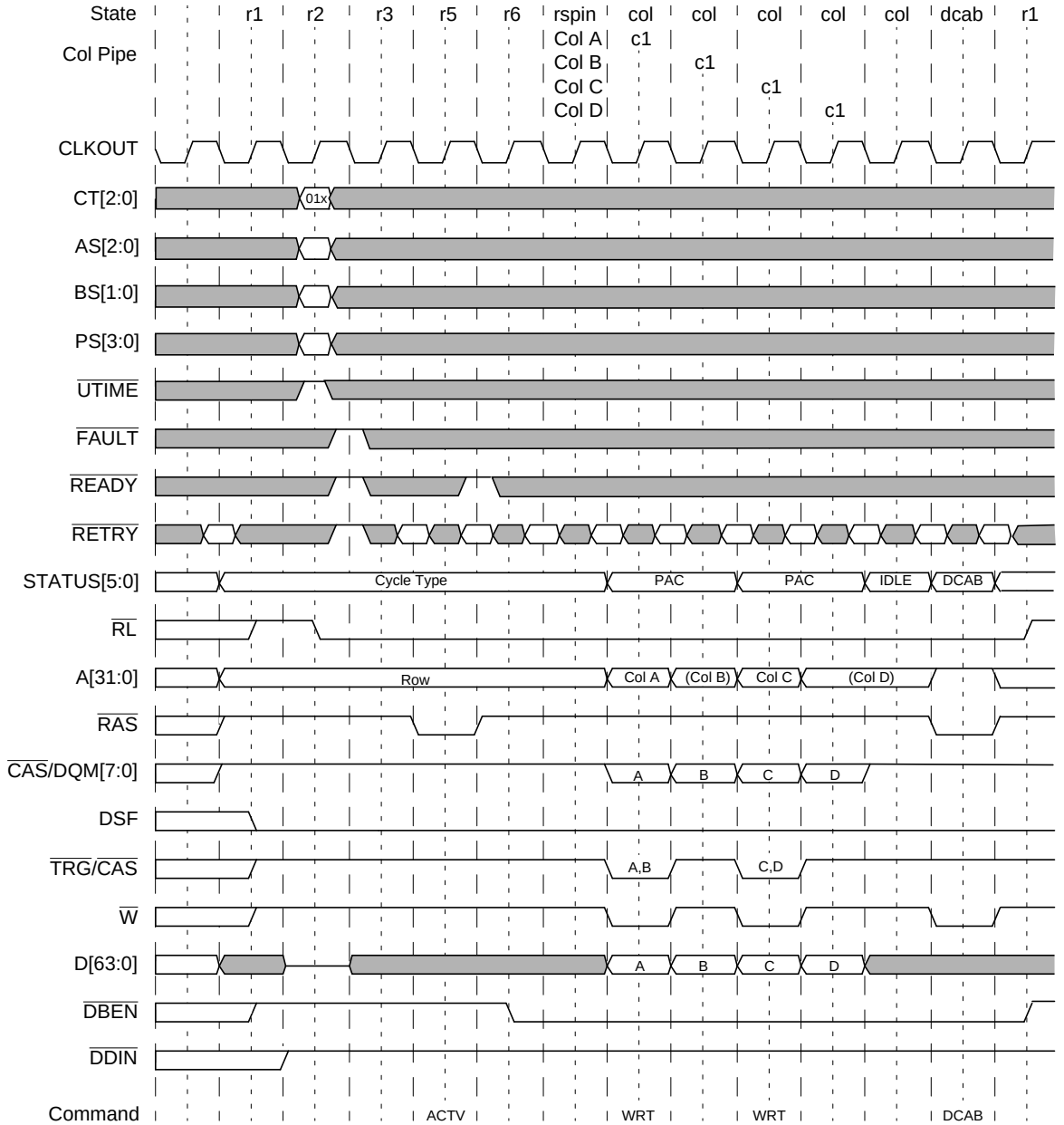
ci **Column-idle stage**. ci occurs when no access is loaded into the pipeline. Also, at least one ci stage will be inserted after the final column-write access on the current page. The TC indicates the column-idle state by setting STATUS[5:0] to 011111b, and deactivates the $\overline{CAS}$/DQM strobe. No data is output.

dcab **DCAB stage.** If the next access is on a new page, a DCAB cycle is performed at least one column-idle cycle after the final column access. The TC sets all the A[31:0] and $\overline{CAS}$/DQM lines high, and deactivates $\overline{TRG}/\overline{CAS}$ lines. Also, $\overline{W}$ and $\overline{RAS}$ are activated, and the STATUS[5:0] lines are set to 111111b to indicate a DCAB cycle.

As Figure 10–16 shows, an SDRAM write with a burst length of 2 is selected by a CT code of 01x (CT0 determines CAS latency and is ignored for writes) at the start of the access. The example shows two write commands. Each write command performs a write to the specified column address and its adjacent column. The TC may output the next column address in the burst (Col B and Col D) but it will be ignored by the SDRAMs since no command is being issued.

There is no latency on writes so the burst data is output with the column address and on the cycle immediately following. Writes to invalid bytes are disabled via the appropriate $\overline{CAS}$/DQM outputs. If no write to the second column in the burst is required then all $\overline{CAS}$/DQM outputs are high for the second cycle (B or D in the example). Following the final write command an idle cycle is inserted (to meet SDRAM timing requirements). The bank is then deactivated with a DCAB command and the memory interface can begin a new page access.

Figure 10–16. SDRAM Burst Length 2 Write



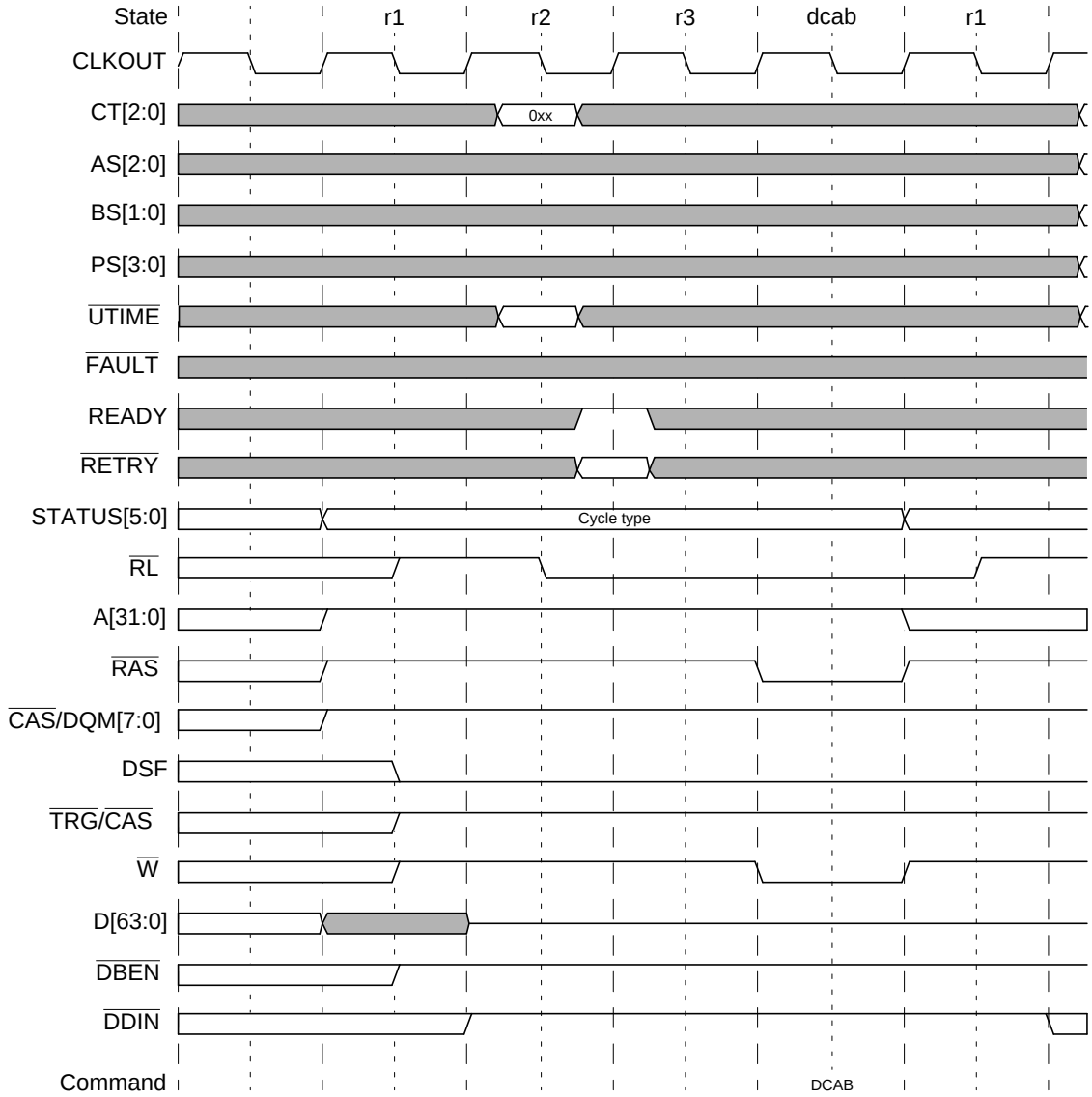
10.4 Special SDRAM Cycles

10.4.1 SDRAM Power-up Deactivation

The SDRAM power-up deactivation is performed when an SDRAM CT code is input during one of the power-up refresh cycles that occurs after a hardware reset. This cycle is required by the SDRAMs prior to refresh as part of their initialization sequence.

The SDRAM DCAB cycle type (STATUS[5:0] = 000011) indicates that the deactivate cycle is occurring. The TC drives all address outputs high to ensure that all SDRAM banks are deactivated, so external decode logic must enable SDRAM banks based upon the STATUS code only. The CT2 input must be low to indicate that SDRAM is being accessed, the other CT inputs and $\overline{\text{UTIME}}$ must be valid, but do not affect the cycle.

Figure 10–17. SDRAM Power-up Deactivation

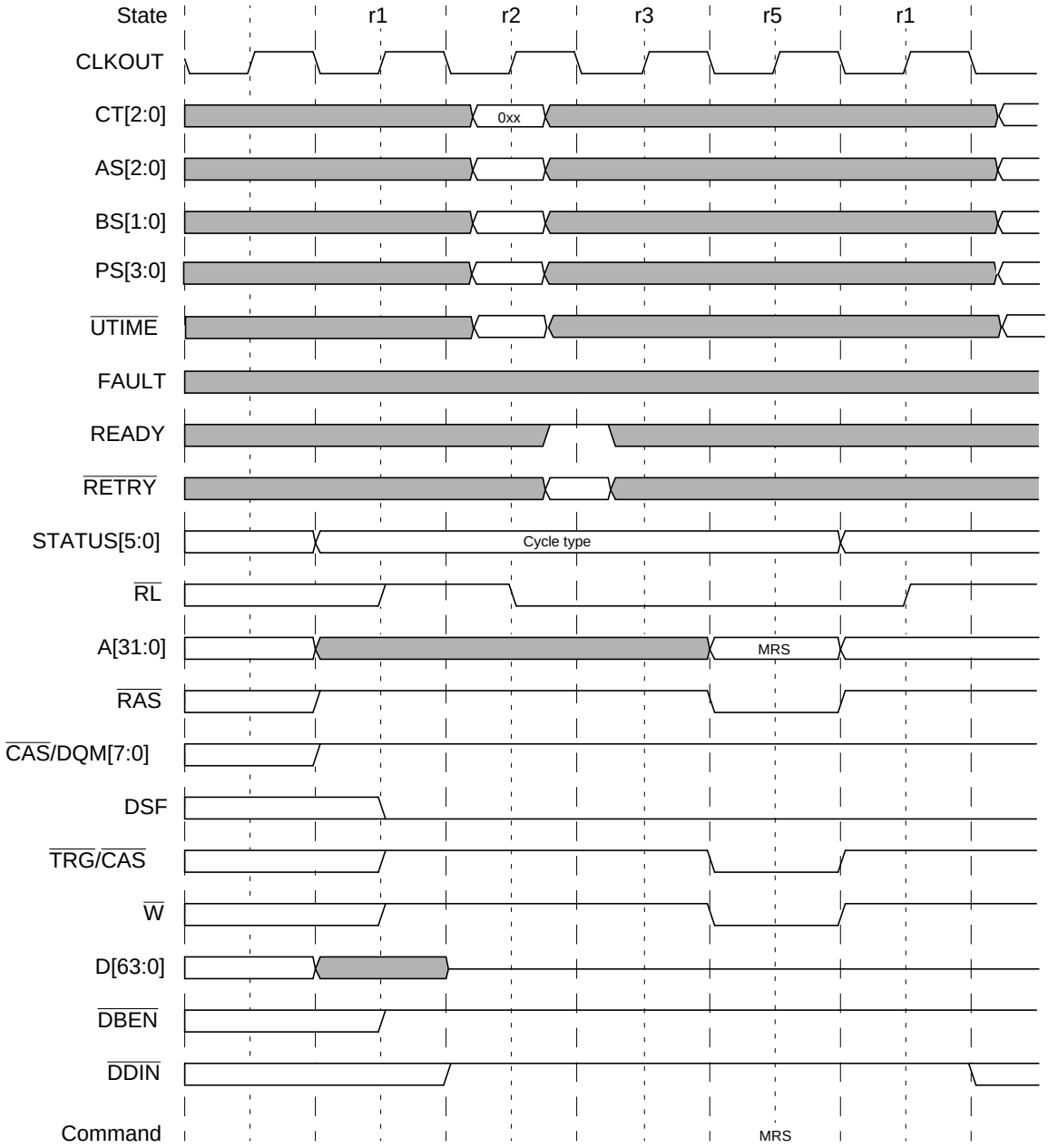


10.4.2 SDRAM Mode Register Set

Mode register set (MRS) cycles are performed (if SDRAM is present) following power-up refresh after the TMS320C80 is reset. The cycle initializes SDRAM mode registers to the proper value for operation with the transfer controller. Generally, only one MRS cycle is performed. Additional MRS cycles (for SDRAM banks configured differently) can be generated by retrying the current MRS cycle using the $\overline{\text{RETRY}}$ input.

The address output at the beginning of the cycle contains no valid information, so SDRAMs must be enabled using the STATUS code (001100). The value input on the CT[2:0] pins and $\overline{\text{UTIME}}$ selects which CAS latency and burst length values will be programmed into the mode registers. The location of the MRS value on the address bus is determined by the AS and BS inputs at the start of the cycle.

Figure 10–18. SDRAM Mode Register Set



Chapter 11

Special Packet-Transfer Access Modes

This chapter continues the discussion of packet transfers and covers some nonstandard packet transfer access modes.

This chapter covers the following topics:

Topic **Page**

11.1 Peripheral-Device Packet Transfers

The peripheral device transfer mode (PAM = 001) allows a peripheral device to use the TC's memory controller to read from or write to external TMS320C80 memory. When a peripheral device transfer occurs, the TC drives memory address and control lines normally, but does not read in or drive out data. This allows a peripheral device such as a system bus to read or write data. The direction of the peripheral transfer is determined by the way in which the packet transfer parameters are programmed. Programming peripheral packet transfers can be broken into the following tasks:

- ❏ Programming peripheral reads
- ❏ Programming peripheral writes
- ❏ Initiating peripheral device transfers
- ❏ Transfer synchronization

11.1.1 Programming Peripheral Reads

Peripheral read transfers (reading from memory into a peripheral device) are generated by programming the src parameters of the packet transfer to access the memory data needed by the peripheral. The src access mode can be either dimensioned or guided (excluding fill-with-value.) The dst transfer is disabled by setting the dst transfer mode (DTM field) to 000, the dst start address to 0x00000000, and the dst A, B, C counts to zero. The src address must be off chip. For guided transfers, the src start/base address must be off chip.

11.1.2 Programming Peripheral Writes

Peripheral write transfers (writing from a peripheral into memory) are generated by programming the dst parameters to access the memory area to which the peripheral needs to write. The dst access mode can be either dimensioned or guided. The src transfer must be disabled by setting the src transfer mode (STM field) to 000, the src start address to 0x00000000, and the src A, B, and C counts to zero. Dst addresses (including the start/base address for guided transfers) must be off chip.

11.1.3 Initiating Peripheral Device Transfers

A peripheral-device packet transfer request can be submitted by any processor at any priority and serviced using the normal prioritization scheme. In normal operation, however, a peripheral initiates the transfer (via the $\overline{\text{XPT}}$ inputs) when it desires to read or write data.

11.1.4 Transfer Synchronization

Once a peripheral device has requested a packet transfer, it waits until the TC begins the packet transfer before it can read or write data. The beginning of the peripheral device transfer is signaled by a unique cycle-code output on STATUS[5:0] at row time. The beginning and end of a peripheral device transfer always invalidate the LASTPAGE register and force a row access (see section 5.3.3, *Dynamic Page Sizing*.) The peripheral device must monitor the STATUS[5:0] pins to determine when to begin the data transfer. The type of memory cycles (bus size, column timing, etc.) generated by the TC corresponds to what the memory identification inputs (BS[1:0], CT[2:0], etc.) select at the beginning of the cycle. The peripheral device can then synchronize its data transfer to the subsequent column accesses using $\overline{\text{CAS}}$/DQM[7:0], CLKOUT, etc.

Because memory addressing is generated by the transfer parameters, the transferring peripheral must be ready to send or receive data in the order it is accessed by the TC. The peripheral must also be able to handle the transfer rate of the TC or be able to insert wait states to slow the transfers.

Because the peripheral uses the TMS320C80's data bus for a transfer, it must not drive the data bus until the peripheral transfer begins. This is achieved by placing transceivers between the peripheral data bus and the TMS320C80 data bus, and enabling them only during peripheral transfers. The TMS320C80 puts its data bus in high impedance during peripheral transfers, and also drives $\overline{\text{DBEN}}$ high to disable its external transceivers, if present.

11.2 Peripheral-Device Transfer Cycles

When external-memory read and write cycles are performed as part of a peripheral device packet transfer, the cycle operation is slightly modified. The state sequence is identical to what would occur in a normal read or write. However, the following signals are changed during peripheral device transfers:

- **STATUS[5:0]**—The status code output during row time indicates either a peripheral device read or peripheral device write instead of a normal read or write.
- **$\overline{\text{DBEN}}$**—$\overline{\text{DBEN}}$ remains high (inactive) during the entire peripheral device transfer cycle. This prevents any transceivers present on the TMS320C80 data bus from driving the memory data bus while it is being driven by the peripheral device.
- **D[63:0]**—The TMS320C80 data bus remains in high impedance during the entire peripheral device transfer so that it can be driven by the peripheral device.

Peripheral-device packet transfer cycles cannot be page moded with other read or write cycles. The TC's internal LASTPAGE register is treated as invalid both before and after a peripheral-device transfer cycle. These cycles always begin with a row access and return to the r1 state upon completion.

Figure 11–1 and Figure 11–1 show a peripheral device packet transfer read and write cycle, respectively.

Figure 11–1. Peripheral Device Packet Transfer Read (Two-Cycles-per-Column) Peripheral

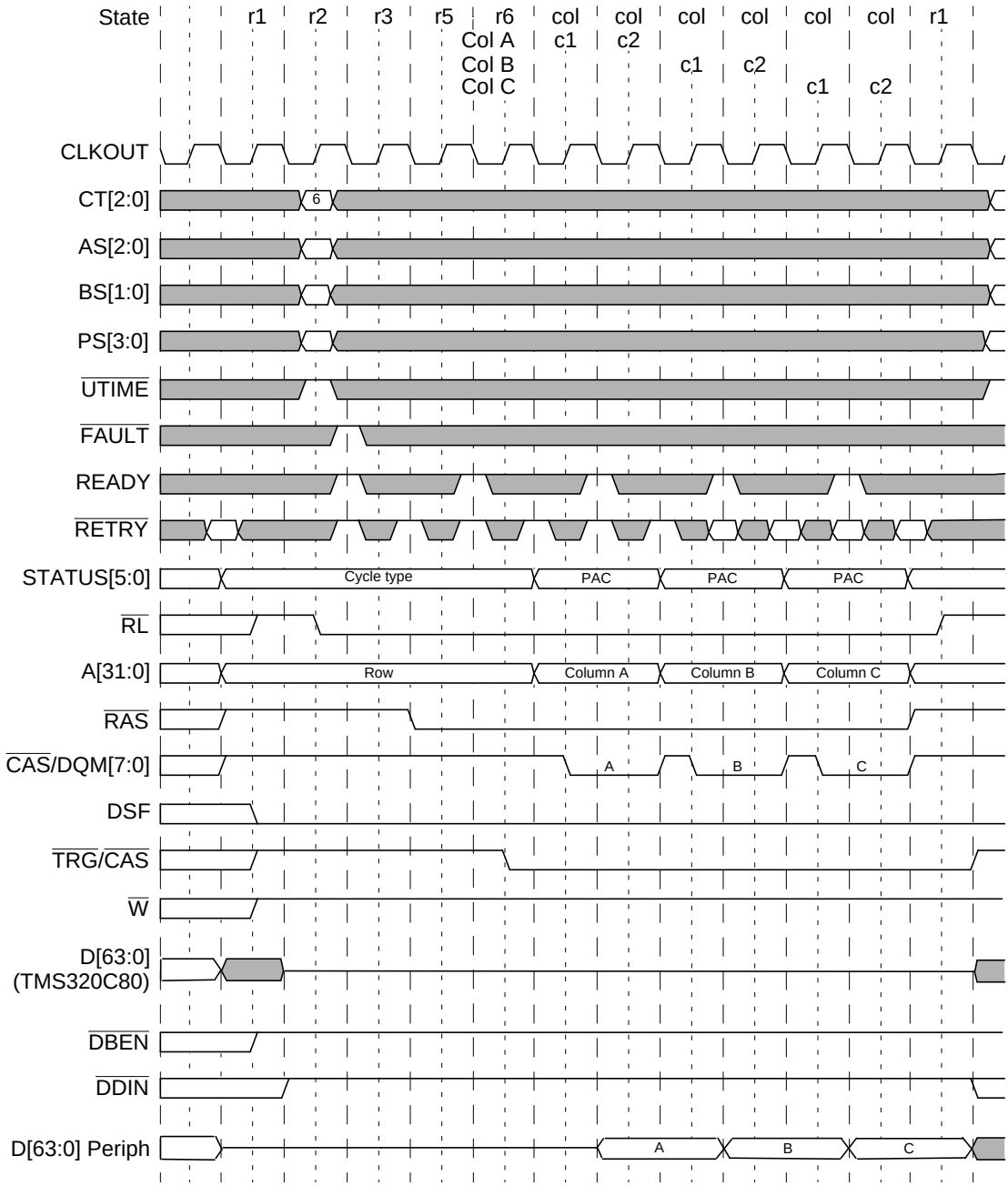
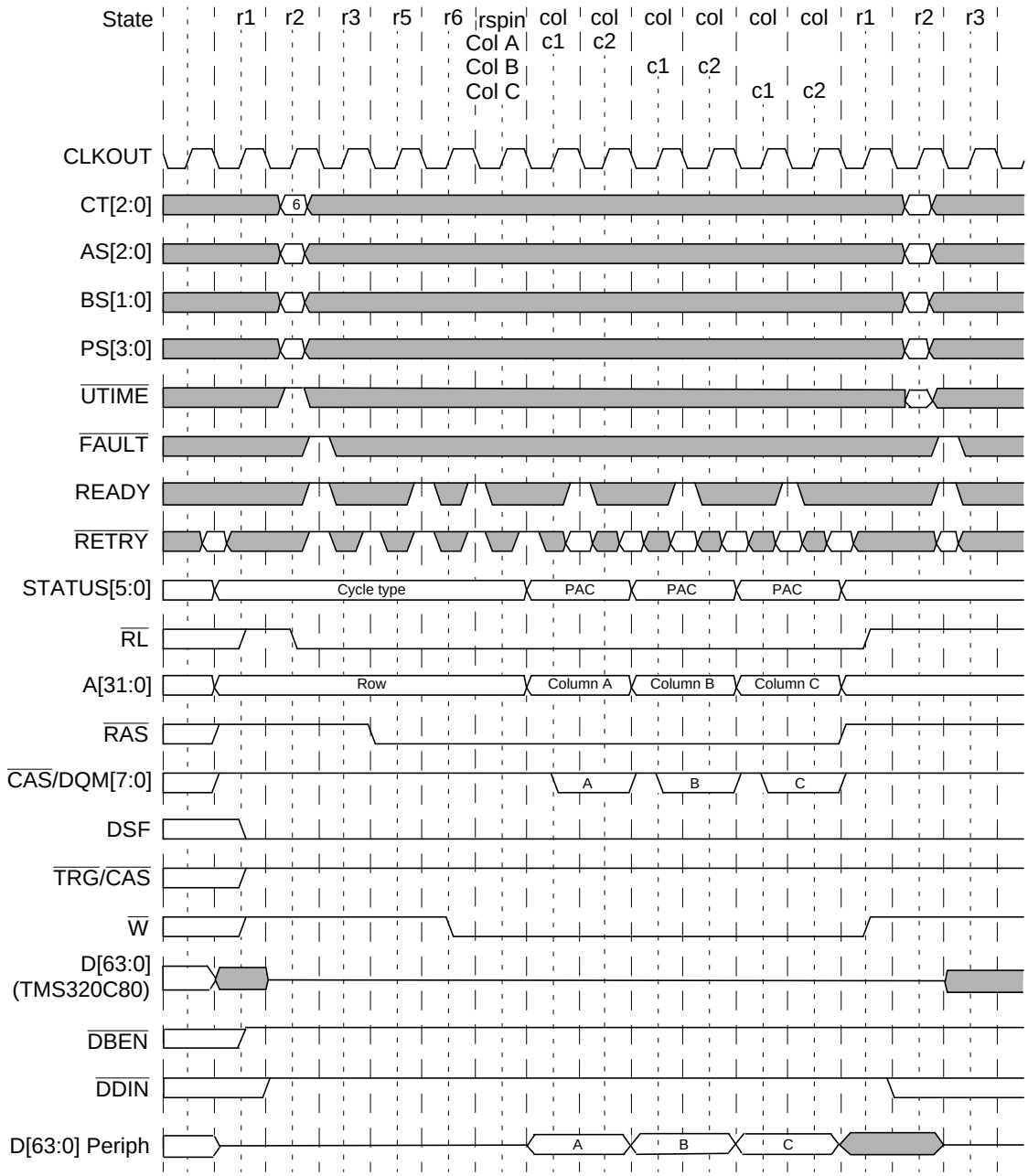


Figure 11–2. Device Packet Transfer Write (Two-Cycles-per-Column)



11.3 The Block-Write Modes

Enabling the block-write option field of a packet transfer's parameters (PAM=010) allows the TC to perform block-write operations using any one of the three following modes:

- **8x**
- **4x**
- **Simulated**

Mode selection is determined by system hardware when the block write begins. This allows software to use block write without regard to the type of block write or to whether the addressed memory supports it. Each block-write mode is only supported for a 64-bit bus size and 8-bit values (a 1-to-8 expand). Dst start addresses for block writes must be off chip and 64-bit aligned.

11.3.1 Selecting Block-Write Modes

The TC block-write mode is selected by the value input to the BS[1:0] pins by external circuitry. Because block write is supported for a 64-bit data bus only, these bus size inputs are used as block-write selects during block write and load-color-register cycles. Table 11–1 shows the block-write modes selected by BS[1:0].

Table 11–1. BS[1:0] Block-Write Codes

BS[1:0]	Block-Write Mode
0 0	Simulated
0 1	Reserved
1 0	4x
1 1	8x

11.3.2 Expressing Block-Write Notation

A number of block-write methods exist due to the variety in VRAM sizes and architectures. The following notation can be used to express the various block-write methods:

column locations per color register × *color register length* × *number of color registers*

For example, most 1-M VRAMs (256k, 4 array) support a 4 x 4 x 1 block write. They have one 4-bit color register, and each block-write cycle controls the

writing of the color register to four adjacent column locations (on each of the four memory-array planes).

11.3.3 8x Block-Write Mode

The 8x block-write mode should be used with VRAMs that can write eight column locations per access (for example, 8 x 8 x 1 or 8 x 8 x 2) at 8 bits per location. Each bit of source data is output to one bit on the data bus (D[63:0]) and enables or disables writing of the 8-bit VRAM color register to one of the 64 columns accessed by the cycle. Each column represents a column location on each of the eight memory array planes within each VRAM. Up to 64 bytes of color register data can be written in a single access.

VRAMs that support 8x block writes ignore the three least significant column-address bits during block-write cycles. Block-write cycles always begin on 64-byte boundaries, because the data bus is 64 bits (eight bytes) wide, and also because the three LSBs are ignored. For dst addresses that are not 64-byte aligned, the TC aligns the src bits and provides the missing 0s for the locations within the 64-byte access that are not written to. The TC then remaps the bits so that they address the proper column locations within each of the VRAMs on the data bus.

Figure 11–3 shows data remapping for an 8x block write in little-endian mode. Figure 11–3(a) shows how the aligned source bits are mapped to the external data pins. The first eight src bits control the first eight bytes of the dst. These bytes are actually the least significant column location in each of the eight 8-plane VRAM arrays being accessed. Because the LSbyte is controlled by the VRAM's D0 inputs (D0 and D8 inputs for 16-bit VRAMs) during block writes, src bits 0-7 get mapped to bits 0, 8, 16, 24, 32, 40, 48, and 56, respectively, of the data bus. These bits represent the D0 inputs (D0 and D8 inputs for 16-bit devices) of the accessed VRAMs. The remaining src bits are mapped in a similar manner. Figure 11–3(b) shows how the src bits appear on the data bus.

Figure 11–4 shows how the src data is mapped for 8x block writes in big-endian mode. Note that in big-endian mode, the LSB of the src image is the leftmost bit and the MSB is the rightmost bit. The mapping mechanism is the same as for little endian. This means src bits 0-7 are mapped to bits 63, 55, 47, 39, 31, 23, 15, and 7, as shown in Figure 11–3 (a). These bits represent the least significant column locations being accessed in the VRAM arrays. These bytes are controlled by the D0 inputs (D0 and D8 inputs for 16-bit devices) of the VRAMs. Therefore, the data bus must be connected to the

VRAMs in reverse order for block writes to work correctly. This is shown in Figure 11–5.

Complete examples of the 4x and 8x block write for both big and little endian can be found in section 11.4, *Block-Write Examples*.

Figure 11–3. 8x Block-Write Output (Little-Endian)

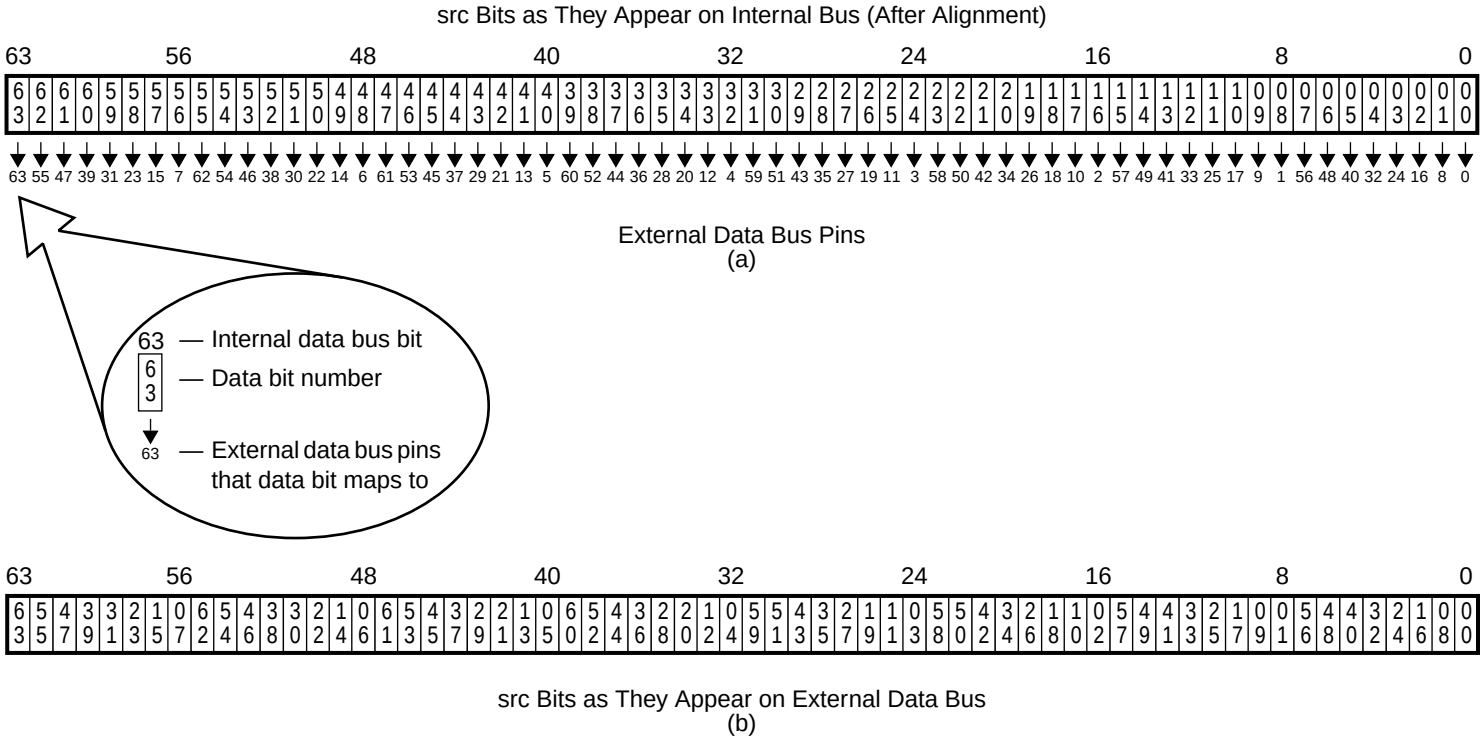


Figure 11–4. 8x Block-Write Output (Big-Endian)

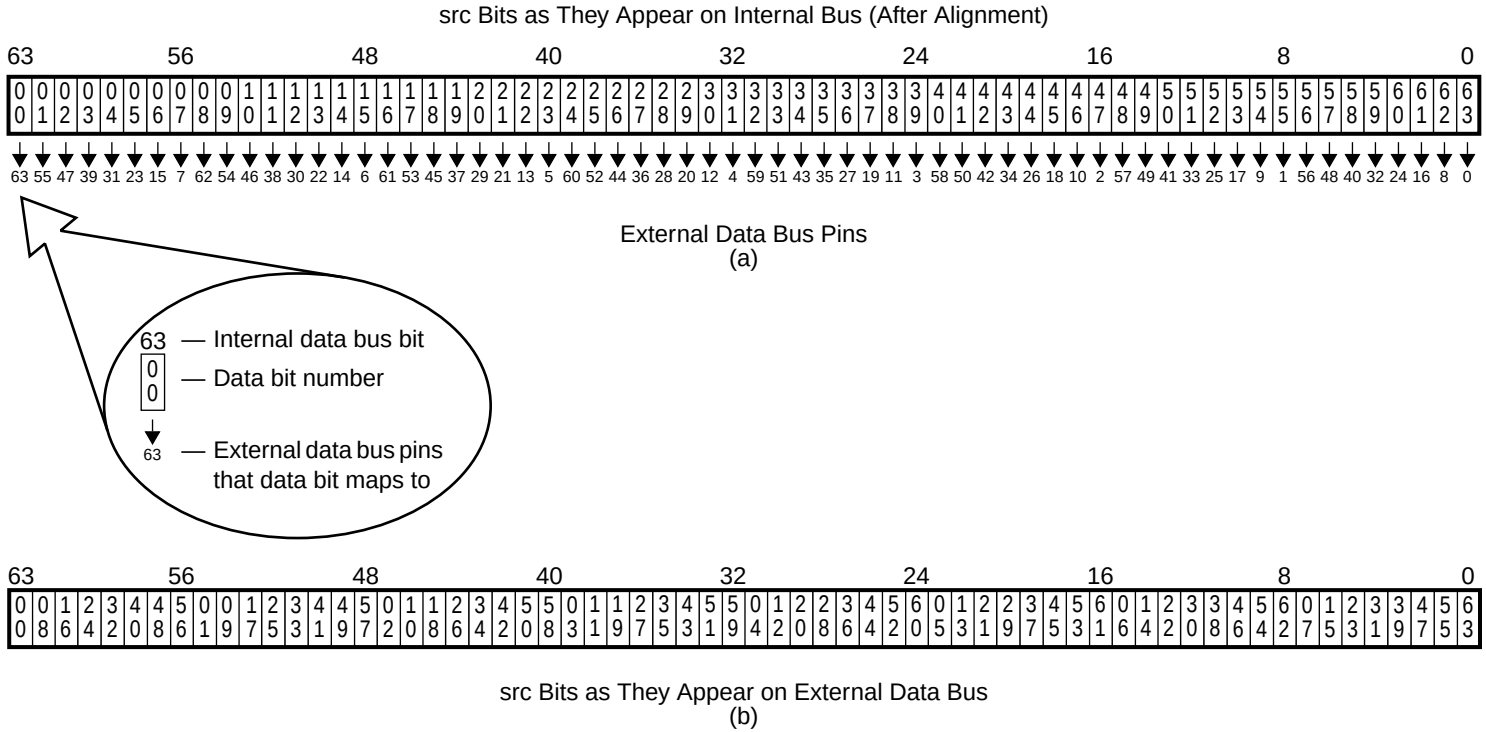
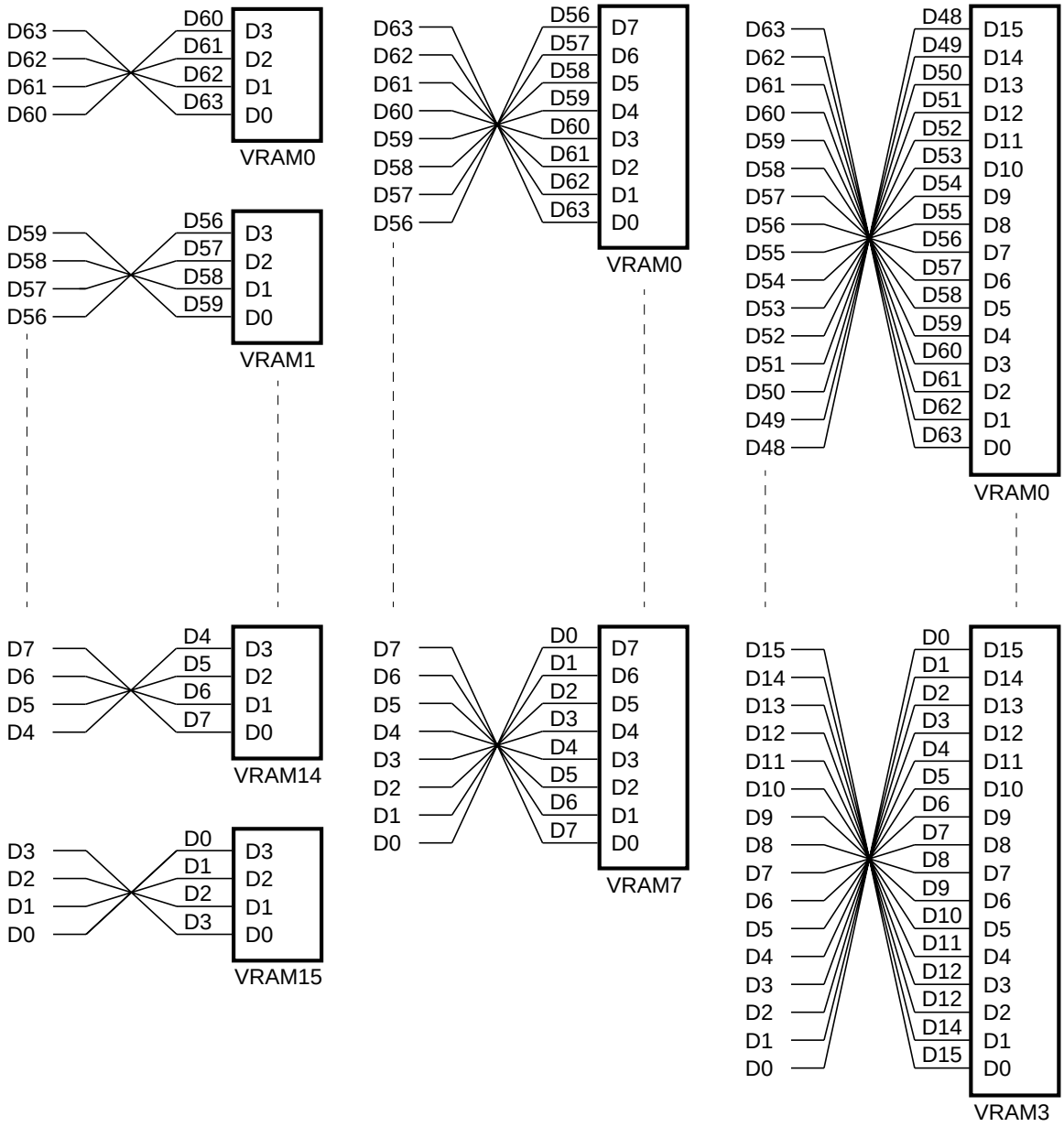


Figure 11–5. Connection to Data Bus for 4-Bit, 8-Bit, and 16-Bit VRAMS in Big-Endian Mode



Note:

Connecting the data bus in reverse order does not affect normal reads and writes, because the data is both written and read in reverse order. However, be aware that the bits that are shifted out of (or into) the VRAM's serial port are also in reverse order. This may require reversing the serial data bus order when connecting an output (or input) device to assure proper operation.

11.3.4 4x Block-Write Mode

The 4x block-write mode should be used with VRAMs that can write four column locations per access (for example, 4 x 4 x 1, 4 x 4 x 4, 4 x 8 x 1, or 4 x 8 x 2) at either four or eight bits per location. In the case of 4x4 block writes, each bit of source data is output as two bits on the data bus (D[63:0]) and enables or disables the writing of two of the 4-bit VRAM color registers to two of the 64 columns. Each column represents a column location on four planes of the array accessed by the cycle. Because each color register is only four bits wide, it takes two registers to represent an 8-bit pixel. Each source bit must control the two color registers that write to adjacent nibbles in order for an 8-bit write to occur. This allows up to 32 bytes (of color register data) to be written in a single access.

VRAMs that support 4x block writes ignore the two least significant column address bits during block-write cycles. Block writes always begin on 32-byte boundaries. As with 8x block writes, the TC aligns the src data to the specified (by the dst start address) doubleword within the 32-byte block, and fills in the missing 0s for double words that are not being written.

Figure 11–6 shows the data remapping for a 4x block write in little-endian mode. Figure 11–6 (a) shows how the aligned source bits are mapped to the external data pins. The first eight src bits control the first eight bytes of the dst. However, because each data input in 4 x 4 block writes causes only four bits to be written, each src bit must be mapped to two data bus pins. This causes a full byte to be written. Thus, src bits 0-7 are mapped to bits 0 and 4, 8 and12, 16 and 20, 24 and 28, 32 and 36, 40 and 44, 48 and 52, and 56 and 60, respectively. Figure 11–6 (b) shows how the src bits appear on the data bus.

In the case of 4 x 8x1 or 4 x 8 x 2 block writes, the VRAM color registers are eight bits wide. Only four of the VRAM's eight data inputs are used to select the bytes to be written. Thus, half of the data bus is not used and only 32 bytes can be written in a single access. Figure 11–6 (c) shows the data bus

bits used during 4x8 block writes. Note that the data output on D[63:0] is identical to the output in Figure 11–6 (b); the VRAMs simply ignore every other nibble.

Figure 11–7 shows the src data mapping for 4x block writes in big- endian mode. In big-endian mode, the src bits 0-7 are mapped to bits 63 and 59, 55 and 51, 47 and 43, 39 and 35, 31 and 27, 23 and 19, 15 and 11, and 7 and 3 of the external data bus respectively. As with 8x big-endian block write, the data bus must be connected to the VRAMs in reverse order to assure correct operation, as shown in Figure 11–5.

Figure 11–6. 4x Block-Write Output (Little-Endian)

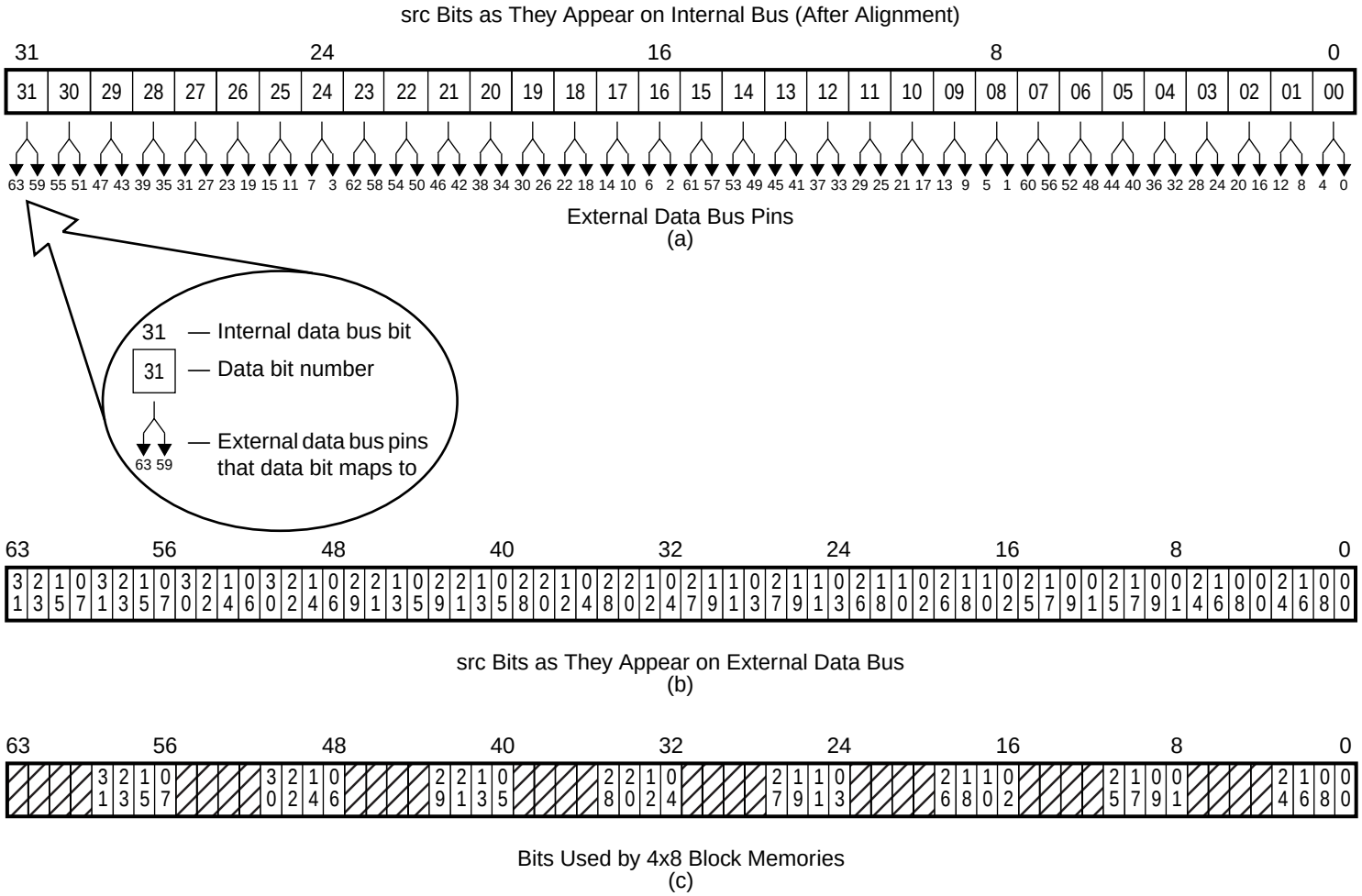
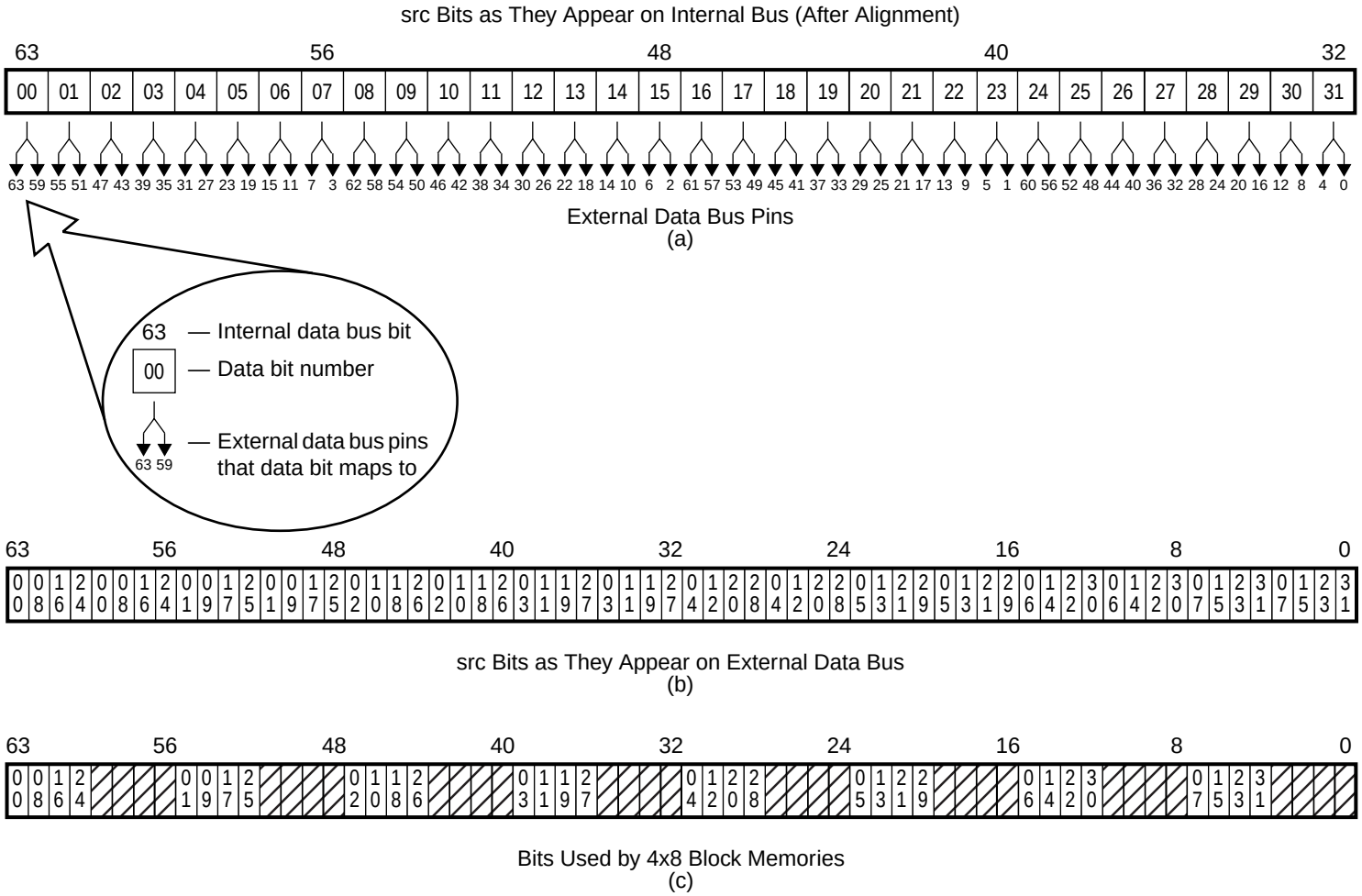


Figure 11–7. 4x Block-Write Output (Big-Endian)



11.3.5 Simulated Block Writes

The TC provides a simulated block-write mode for memory devices that do not support block write. In this third mode, the 64-bit color register value contained in the packet transfer parameters is output on the data bus. Each source data bit functions as a byte select by controlling the $\overline{\text{CAS}}$/DQM pins. This enables or disables one of the eight bytes addressed during the cycle. Block writes are basically converted to fill-with-value type transfers in which the color register value becomes the fill value. Accesses to the dst become normal page-mode 64-bit write cycles, in which writes to some bytes are disabled, as specified by the src data.

11.3.6 Color Register Loading

Before 4x and 8x block-write cycles can be performed, the VRAM/SGRAM color registers must be loaded with the correct values. The TC does this by performing a load color register (LCR) or special register set (SRS) cycle using the color register value contained in the packet transfer parameters. A block-write transfer can be interrupted by a higher priority request that can change the VRAM/SGRAM color registers, such as a host access or another block-write transfer. Because of this, an LCR/SRS cycle is also performed whenever a block-write transfer resumes. An LCR/SRS cycle occurs whenever:

- ❏ A 4x or 8x block-write transfer begins.
- ❏ A block-write packet transfer resumes after suspension.
- ❏ A block-write packet transfer resumes after the host has used and returned the local bus.

The LCR/SRS cycle is not performed if the memory being accessed requires simulated block writes.

Once the color latches are loaded, no more LCR/SRS cycles are performed unless one of the above conditions occurs. If, for example, the block write begins in 8x mode, and then alternates between simulated and 8x modes, the LCR/SRS does not repeat each time the 8x mode is entered.

11.3.7 Block-Write Mechanism

Following is the sequence of events for block-write packet transfers:

1) The TC outputs the LCR/SRS status code and the address of the first block write to be performed.

2) The TC reads the input value on the BS[1:0] pins.

3) If BS[1:0] = 10 or 11

   a) The LCR/SRS cycle is completed using the 64-bit color register value contained in the packet transfer parameters.

   b) 4x or 8x block-write cycles are generated to complete the packet transfer.

4) If BS[1:0] = 00

   a) The LCR/SRS cycle becomes a normal page-mode write using the 64-bit color register value contained in the packet-transfer parameters as data, and using the src data bits as byte selects.

   b) If a new row access begins because of a page change or interruption from a higher priority cycle, then step (1) is repeated for the next destination address.

5) The TC outputs the block-write status code and the address of the next block write to be performed.

6) If BS[1:0] = 10 or 11

   a) The block-write page-mode cycle is completed using the src data bits.

   b) If a new row access begins, step 5 is repeated.

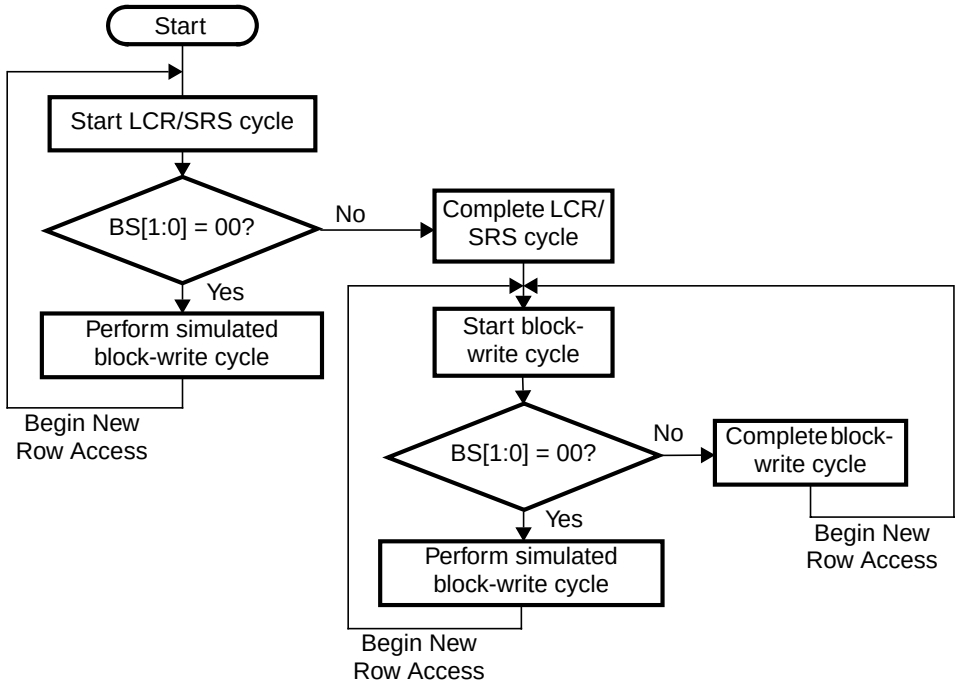
7) If BS[1:0] = 00

   a) The block-write cycle becomes a normal page-mode write using the 64-bit color register value contained in the transfer as data, and the src data bits as byte selects.

   b) If a new row access is begun, step 5 is repeated for the next destination address.

These steps are shown graphically in Figure 11–8.

Figure 11–8. Block-Write Sequence



Note:

The TC always attempts to perform real (4x or 8x) block writes. The normal write cycles that occur during simulated block-write modes always have either the LCR or block-write status codes.

11.4 Block-Write Examples

The following sections provide detailed examples of 4x and 8x block-write operations for little- and big-endian modes.

Each example is a dimensioned src to dimensioned dst transfer that writes an identical 7-byte (56-bit) src pattern into the dst memory. Block writes occur at eight bits per pixel, which results in 56 bytes being written into the memory array from the color latch. Technically, only the src bits with a value of 1 cause a byte to be written to. Although 56 bytes could be written to for a source of all 1s, the actual number in the examples found in this section is less. The resulting dst memory image would be identical for each example on a display device.

Each transfer (src and dst) consists of one patch with one line of seven bytes. The src image starting address is 0x10000002. This requires that the TC fetch six bytes from the double word at 0x10000000 and one byte from the double word at 0x10000008. The dst start address is 0x02000008. It must be 64-bit aligned. The color register value is 0x55555555BBBBBBBB in little-endian mode. When loaded into the VRAM color register, the eight LSbytes are written as 0xBB, and the eight MSbytes are written as 0x55. In big-endian mode, the LSbytes are at the left of the doubleword, so the color register value must be 0xBBBBBBBB55555555 to achieve the same result.

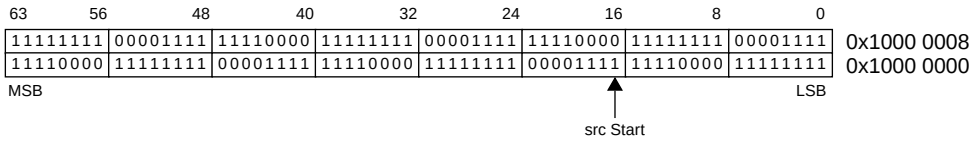
11.4.1 8x Little-Endian Block Write

Figure 11–9 shows an 8x block-write operation in little-endian mode. The seven bytes of src data are fetched starting at address 0x10000002. Note that the dst starting address (0x02000008) is not 64-byte aligned. It points to the second doubleword in the 64-byte area, beginning at 0x02000000. This means the src bits must also be aligned to the dst address. Because the least significant doubleword of the 64-byte area is not written, the TC inserts 0s into bits 0-7 of the aligned src data. The TC then maps the data onto the external data bus to perform the block write. When the block write has completed, dst bytes corresponding to 0 src data bits remain unchanged while those corresponding to 1 src data bits are replaced by the color-register value corresponding to that byte.

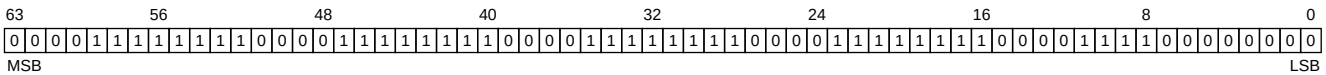
Figure 11–9. 8x Block Write (Little-Endian)

src Start Address:	0x1000 0002	dst Start Address:	0x0200 0008
src A Count:	7	dst A Count:	7
src B Count:	0	dst B Count:	0
src C Count:	0	dst C Count:	0
Color Register Value:	0x5555 5555 BBBB BBBB		

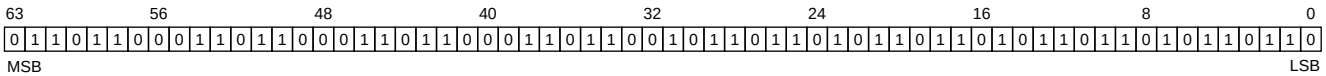
src Memory Image



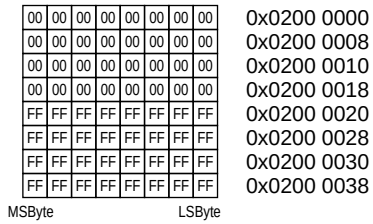
src Bits After Alignment



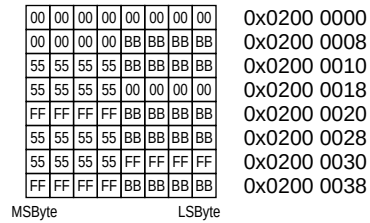
src Bits as They Appear on External Data Bus



dst Memory Image Before Block Write



dst Memory Image After Block Write



11.4.2 8x Big-Endian Block Write

Figure 11–10 shows an 8x block-write operation in big-endian mode. This example performs a block write identical to the little-endian example with the following exceptions:

- First, the color register values are reversed, because in big-endian the LSbyte is on the left and the MSbyte is on the right.
- Second, the src memory image is reversed. It is identical to that of the little-endian example except that it is a mirror image.
- Third, the connection of the data bus to the VRAMs in big-endian mode must be reversed.

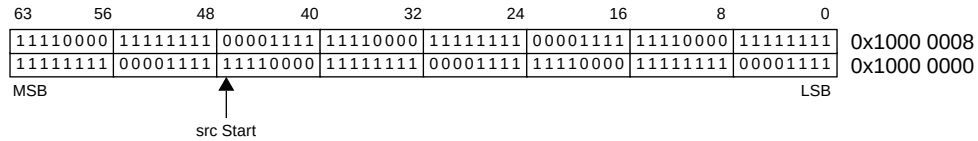
The numbers at the top of the src image diagram represent data bus pins, not the src bit number. In this case, the least significant doubleword is represented by bits 63-56 of the aligned src data and 0s are written in these locations. The aligned src bits are mapped to the external data bus as shown, and the block-write cycle results in the final dst memory image.

Figure 11–10. 8x Block Write (Big-Endian)

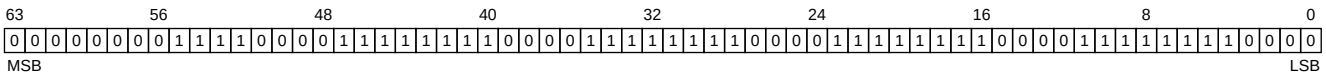
src Start Address:	0x1000 0002	dst Start Address:	0x0200 0008
src A Count:	7	dst A Count:	7
src B Count:	0	dst B Count:	0
src C Count:	0	dst C Count:	0

Color Register Value: 0x5555 5555 BBBB BBBB

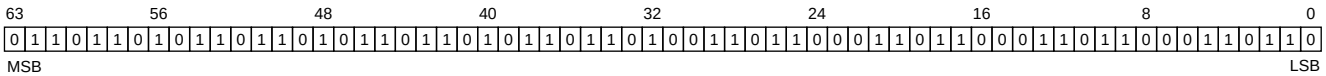
src Memory Image



src Bits After Alignment



src Bits as They Appear on External Data Bus



dst Memory Image Before Block Write

0x0200 0000	00	00	00	00	00	00	00	00
0x0200 0008	00	00	00	00	00	00	00	00
0x0200 0010	00	00	00	00	00	00	00	00
0x0200 0018	00	00	00	00	00	00	00	00
0x0200 0020	FF	FF	FF	FF	FF	FF	FF	FF
0x0200 0028	FF	FF	FF	FF	FF	FF	FF	FF
0x0200 0030	FF	FF	FF	FF	FF	FF	FF	FF
0x0200 0038	FF	FF	FF	FF	FF	FF	FF	FF
	LSByte							MSByte

dst Memory Image After Block Write

0x0200 0000	00	00	00	00	00	00	00	00
0x0200 0008	BB	BB	BB	BB	00	00	00	00
0x0200 0010	BB	BB	BB	BB	55	55	55	55
0x0200 0018	00	00	00	00	55	55	55	55
0x0200 0020	BB	BB	BB	BB	FF	FF	FF	FF
0x0200 0028	BB	BB	BB	BB	55	55	55	55
0x0200 0030	FF	FF	FF	FF	55	55	55	55
0x0200 0038	BB	BB	BB	BB	FF	FF	FF	FF
	LSByte							MSByte

11.4.3 4x Little-Endian Block Write

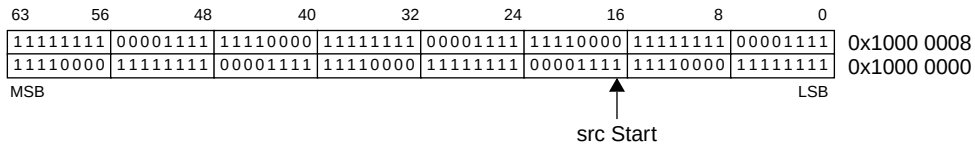
Figure 11–11 shows a 4x block-write operation in little-endian mode. The src bits are fetched and aligned as with the 8x example. However, each src bit must be mapped to two bits on the external data bus, so only 32 bytes can be written from the VRAM color register into the memory array in a single cycle. This means that two cycles are required to complete the block write. Figure 11–11 shows how the 32 LSBs (0-31) are mapped onto the data bus for the first cycle, and the 32 MSBs (32-63) are mapped for the second cycle.

Figure 11–11. 4x Block Write (Little-Endian)

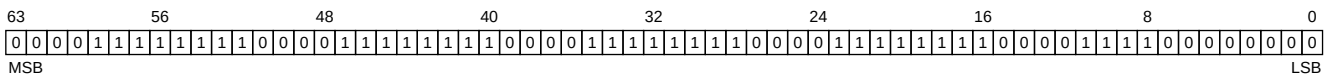
src Start Address:	0x1000 0002	dst Start Address:	0x0200 0008
src A Count:	7	dst A Count:	7
src B Count:	0	dst B Count:	0
src C Count:	0	dst C Count:	0

Color Register Value: 0x5555 5555 BBBB BBBB

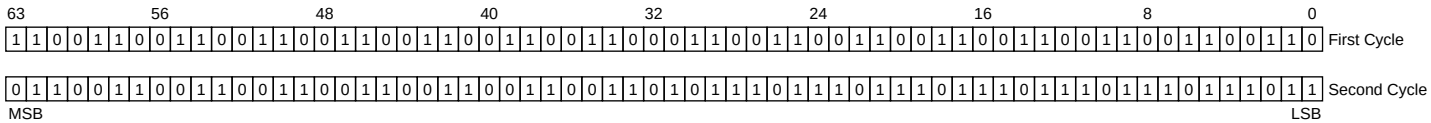
src Memory Image



src Bits After Alignment



src Bits as They Appear on External Data Bus



dst Memory Image
Before Block Write

00 00 00 00 00 00 00 00	0x0200 0000
00 00 00 00 00 00 00 00	0x0200 0008
00 00 00 00 00 00 00 00	0x0200 0010
00 00 00 00 00 00 00 00	0x0200 0018
FF FF FF FF FF FF FF FF	0x0200 0020
FF FF FF FF FF FF FF FF	0x0200 0028
FF FF FF FF FF FF FF FF	0x0200 0030
FF FF FF FF FF FF FF FF	0x0200 0038

MSByte LSByte

dst Memory Image
After First Block Write

00 00 00 00 00 00 00 00	0x0200 0000
00 00 00 00 BB BB BB BB	0x0200 0008
55 55 55 55 BB BB BB BB	0x0200 0010
55 55 55 55 00 00 00 00	0x0200 0018
FF FF FF FF FF FF FF FF	0x0200 0020
FF FF FF FF FF FF FF FF	0x0200 0028
FF FF FF FF FF FF FF FF	0x0200 0030
FF FF FF FF FF FF FF FF	0x0200 0038

MSByte LSByte

dst Memory Image
After Second Block Write

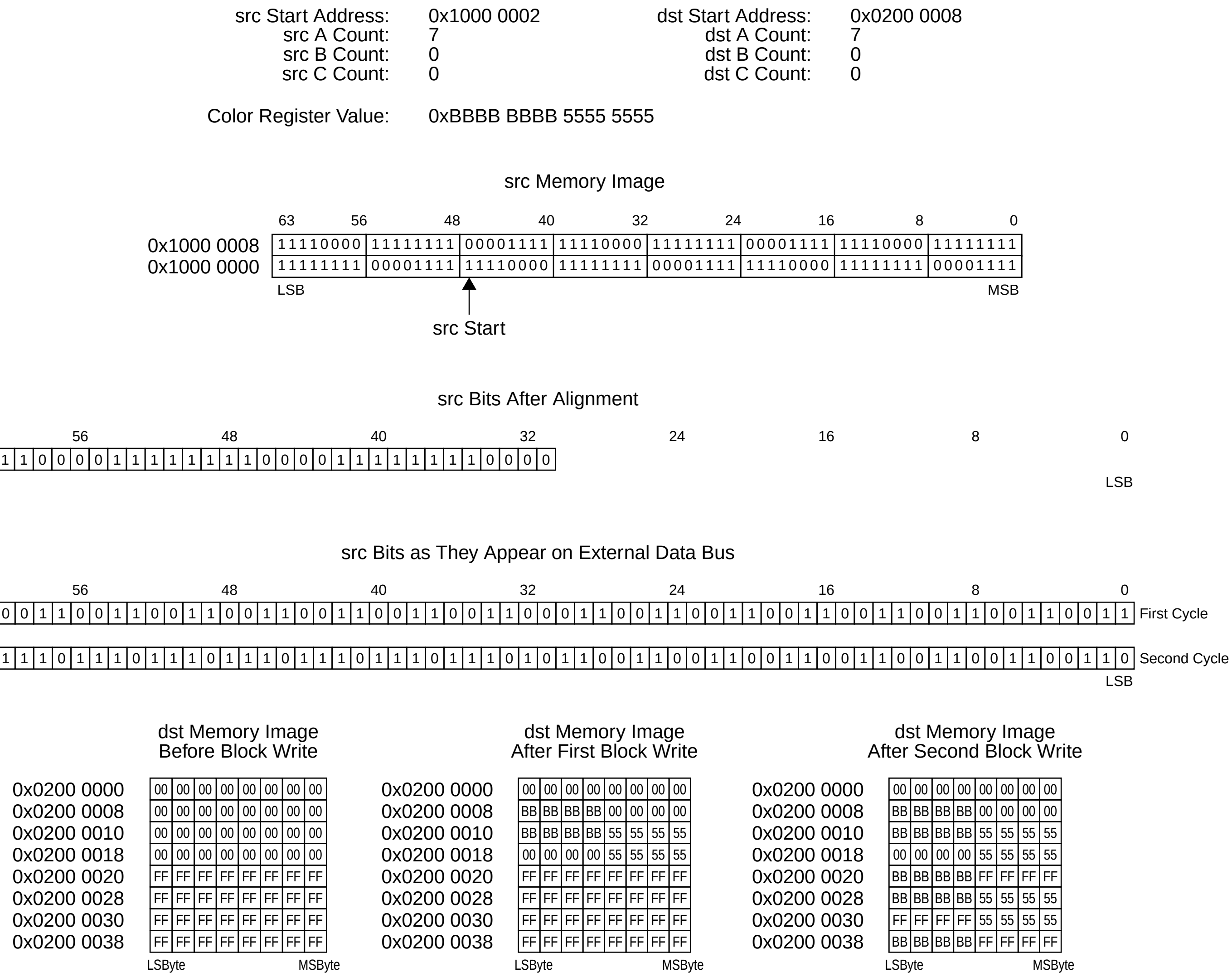
00 00 00 00 00 00 00 00	0x0200 0000
00 00 00 00 BB BB BB BB	0x0200 0008
55 55 55 55 BB BB BB BB	0x0200 0010
55 55 55 55 00 00 00 00	0x0200 0018
FF FF FF FF BB BB BB BB	0x0200 0020
55 55 55 55 BB BB BB BB	0x0200 0028
55 55 55 55 FF FF FF FF	0x0200 0030
FF FF FF FF BB BB BB BB	0x0200 0038

MSByte LSByte

11.4.4 4x Big-Endian Block Write

Figure 11–12 shows a 4x block-write operation in big-endian mode. The src bits are fetched and aligned and then mapped onto the external data bus. The 32 LSBs (63-32) are mapped for the second block-write cycle. The dst memory image is shown as it appears after each of the block-write cycles.

Figure 11–12. 4x Block Write (Big-Endian)



11.5 Serial Register Packet Transfers

Selecting the serial-register transfer mode (PAM = 011) in the PT options field of a packet transfer's parameters disables the TC's data transfer mode. The TC only outputs addresses and $\overline{CAS}$/DQMs. It will also drive the $\overline{TRG}$/$\overline{CAS}$, $\overline{W}$ and DSF pins at row time to select either *read transfer* or *write transfer* VRAM modes. When performing read transfers, the D[63:0] pins are placed in high impedance. When performing write transfers, the D[63:0] pins output mask value of 0xFFFF FFFF FFFF FFFF at row time to disable VRAM transfer masks.

The src parameters generate addresses to perform read transfers (memory-to-register transfers), and the dst parameters perform write transfers (register-to-memory transfers). Each access performed by both src and dst is a single-row access. Because no data is transferred (through the TC), all src accesses are performed before dst accesses. Each src access causes a VRAM row to be transferred into the VRAM shift register. In practice, the src parameters are usually set up (an aligned address, an A count of 1, a B count of 0, and a C count of 0) so that only one transfer is performed. Because all $\overline{CAS}$/DQM lines are active for read transfers, setting A count equal to or less than the src bus size results in a single transfer. A src A count of 1 is ideal for most situations.

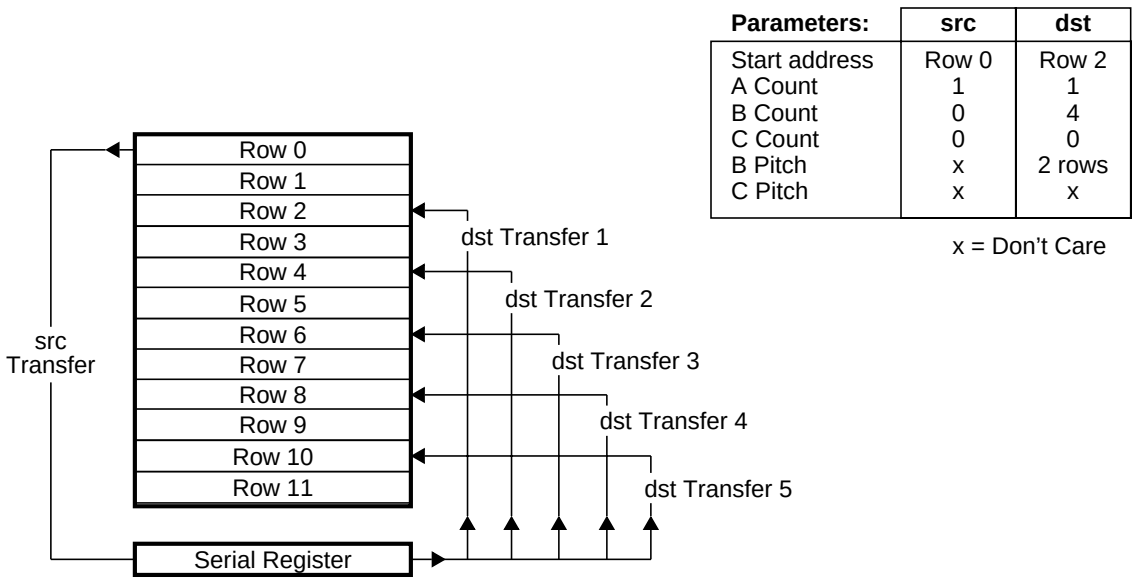
Each dst access causes the VRAM shift register to be transferred into a VRAM memory array row. Normally, each dst access is an aligned transfer to the next sequential row address. This is achieved by setting up the dst parameters with an aligned start address, an A count of 1, a B count of (number of rows) -1, and a B pitch equal to the VRAM row address pitch. Again, all $\overline{CAS}$/DQM lines are activated during the write transfers so setting the A count to less than or equal to the dst bus size results in a single transfer (per line). A dst A count of 1 is ideal for most situations.

A *serial register transfer (SRT)* can be interrupted by a higher priority request, which might change the VRAM shift register contents. It is necessary to perform the read transfer cycle whenever the VRAM shift register may have been corrupted. An SRT transfer's src operation (read transfer) is performed whenever:

- ❏ An SRT packet transfer begins.
- ❏ An SRT packet transfer resumes after being suspended.
- ❏ An SRT packet transfer resumes after the host has used and returned the local bus.

A simple illustration of an SRT transfer is shown in Figure 11–13. Row 0 is assumed to have been set to the desired pattern. This is copied to rows 2, 4, 6, 8, and 10 to produce a striped effect.

Figure 11–13. Serial Register Transfer Mode



Note:

SRT packet transfers should be performed on inactive or blanked frame memories only. Performing SRT packet transfers on active frame memories can disrupt video display or capture.

11.6 Transparency-Packet Transfers

The transparency modes are enabled by setting the packet transfer access mode (PAM) bits in the PT options field to 1XX. Specifying one of the transparency options enables a transparency on source operation. The source data is compared to the 64-bit transparency value specified in the transfer parameters. Transparency can be specified as an 8-, 16-, 32-, or 64-bit data size. This allows one 64-bit, two 32-bit, four 16-bit, or eight 8-bit comparisons to be made. If any of the comparisons are true, the TC disables the corresponding byte strobe(s) to prevent the dst byte(s) from being written.

Note:

Transparency is supported for off-chip destinations only. Specifying transparency for an on-chip dst causes the transfer to suspend with an error condition (see section 3.11, *Packet-Transfer Errors*).

Transparency detection is applied after the source data has been aligned to the destination and external bus size. All eight bytes of data are compared with the corresponding eight bytes of the transparency value even if the bus size is less than 64 bits. The eight comparisons are grouped according to the transparency data size. If all the compared bytes within a group match, then the byte strobes ($\overline{CAS}$/DQM signals) associated with that group are disabled, preventing writes to any of the bytes within that group.

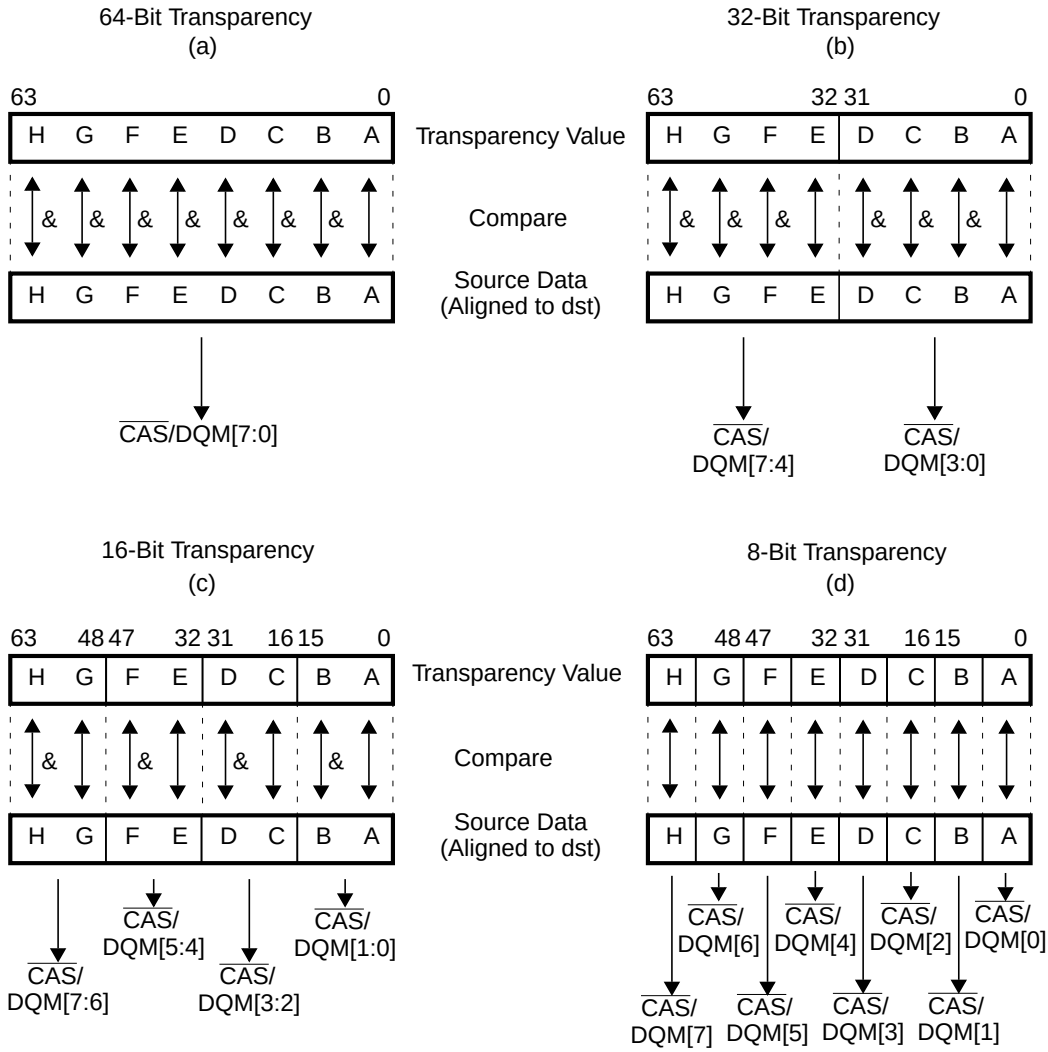
Figure 11–14 (a-d) shows how comparisons are made for 64-bit, 32-bit, 16-bit, and 8-bit transparency data sizes. The ampersand (&) symbol shows which byte comparisons are ANDed together to form a group. As Figure 11–14 (a) shows a 64-bit transparency size causes one 64-bit comparison to be made. If the (src) data and transparency value are equal, all the $\overline{CAS}$/DQM strobes are disabled. Otherwise, all eight bytes are written.

In Figure 11–14 (b), two 32-bit comparisons are made, controlling $\overline{CAS}$/DQM[7:4] and $\overline{CAS}$/DQM[3:0]. Parts (c) and (d) of Figure 11–14 show comparisons of 16- and 8-bit transparency sizes for which the comparison groups control two and one $\overline{CAS}$/DQM lines each, respectively. Note that the $\overline{CAS}$/DQM[7:0] strobes are always identified with the same bits on the data bus regardless of the endian operation. Figure 11–14 applies to both big- and little-endian formats.

Note:

The transparency comparisons take place *after* alignment to the external bus occurs. If the external-bus size is 32 bits, data is always compared to bits 31-0 (bits 63-32 in big endian) of the transparency value, even if 64-bit transparency mode is selected. The transparency mechanism works with a current bus size that can be divided into an integral number of comparison groups. The bus size should always be equal to or larger than the transparency size.

Figure 11–14. Transparency Operation



Chapter 12

VRAM and SGRAM Memory Cycles

In addition to standard read and write cycles, the TC also supports a variety of special VRAM/SGRAM cycles. These include:

- ❏ Load color-register cycles
- ❏ Block-write cycles
- ❏ Read transfer cycles
- ❏ Write transfer cycles
- ❏ Split-read transfer cycles
- ❏ Split-write transfer cycles

Load color-register and block-write cycles are generated by block-write packet transfers, and split-read transfer and split-write transfer cycles are generated by the video controller (VC). The read-transfer and write-transfer cycles can be requested by either packet transfers or VC events. With the exception of block writes, each cycle consists of a row access followed by a single-column access. Block writes can generate multiple page-mode column accesses.

This chapter covers the following topics:

Topic **Page**

12.1 Load Color-Register and Special Register Set Cycles

Load color-register (LCR) cycles write a value into the VRAM's color registers for later use during a block-write cycle. Special register set (SRS) cycles write a value into SGRAM color registers for later use during a block write. LCR/SRS cycles are only supported on 64-bit data buses. A status code of 001101 indicates an LCR/SRS cycle. Because an LCR cycle writes data into the VRAM, its timing closely resembles a standard write cycle. The difference is that unlike normal writes, DSF is active high at both the fall of $\overline{RAS}$ and the fall of $\overline{CAS}$/DQM. Also, because the VRAM color register is a single location, page-mode accesses do not occur.

The row address output by the TC is important for bank-decode information only. In most cases, you will want to select all VRAM banks during LCR cycles because another LCR *is not performed* when a memory page change occurs (see section 11.3.7, *Block-Write Mechanism*). The row addressed at the VRAM is refreshed, but otherwise is a don't care. Similarly, the column address provided to the VRAM for LCR cycles is irrelevant. All $\overline{CAS}$/DQM[7:0] outputs are active during LCR cycles.

Although the $\overline{RETRY}$ input is sampled and must be valid high or low during column states, asserting a column time retry has no effect because only one column access is performed.

The BS[1:0] inputs during LCR/SRS cycles determine the block-write type supported by the addressed memory. If simulated block-write mode is selected, the cycle becomes a standard write rather than an LCR cycle. The following load color-register cycles are detailed, including timing diagrams:

- ❏ Pipelined one-cycle-per-column LCR cycle (CT[2:0] = 100)
- ❏ Nonpipelined one-cycle-per-column LCR cycle (CT[2:0] = 101)
- ❏ Two-cycles-per-column LCR cycle (CT[2:0] = 110)
- ❏ Three-cycles-per-column LCR cycle (CT[2:0] = 111)
- ❏ Special-register-set cycle (CT[2:0] = 0xx)

The first four cycles support standard VRAM; the fifth supports synchronized graphics RAM (SGRAM). The cycle type selected depends on the values sampled at the CT[2:0] inputs. The decoding of these signals is the same as for DRAM and SDRAM cycles, as shown by the values given in parentheses in the above list.

12.1.1 Pipelined One-Cycle-per-Column Load Color-Register Cycle

Figure 12–1 shows an LCR cycle to a pipelined, one-cycle-per-column memory. Although data is latched by the memories at the fall of $\overline{CAS}$/DQM, no data is written into the color register until the following $\overline{CAS}$/DQM strobe. The $\overline{CAS}$/DQM strobes clock the memory pipeline, as well as strobe in column addresses.

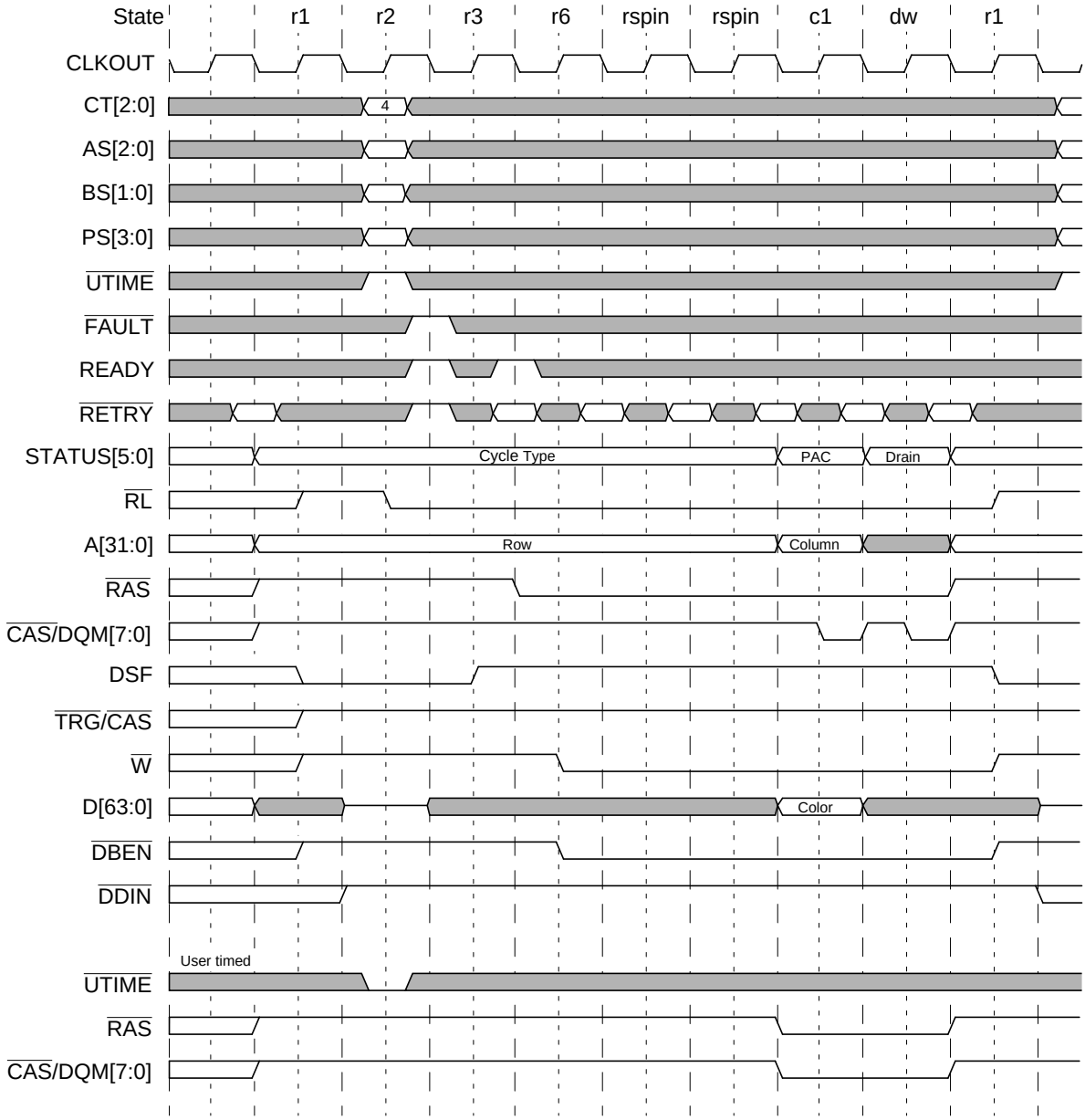
Note:

VRAMs that support pipelined page mode may support only nonpipelined LCR cycles. For these devices, select a nonpipelined memory timing during LCR cycles.

The example in Figure 12–1 shows a single load of the VRAM color registers. The state sequence is identical to a one-cycle-per-column pipeline write. (See section 9.2, *Pipelined One-Cycle-per-Column Write*.) The row access consists of the r1, r2, r3, r6 and rspin states. The rspin state always occurs at least two times. Additional r3 and/or r6 states can be inserted by adding wait states. The column access consists of a single c1 state during which the 64-bit color register value is output on the data bus. This is immediately followed by a dw state to strobe the data into the color register. The memory interface then returns to state r1, where it remains until the next row access begins.

The bottom of Figure 12–1 shows how the cycle is modified when user timing is selected. The $\overline{RAS}$ and $\overline{CAS}$/DQM operations are identical to a user-timed write cycle.

Figure 12–1. Pipelined One-Cycle-per-Column Load-Color-Register



Note: BS[1:0] must have a value of 10 or 11 in order for an LCR cycle to occur.

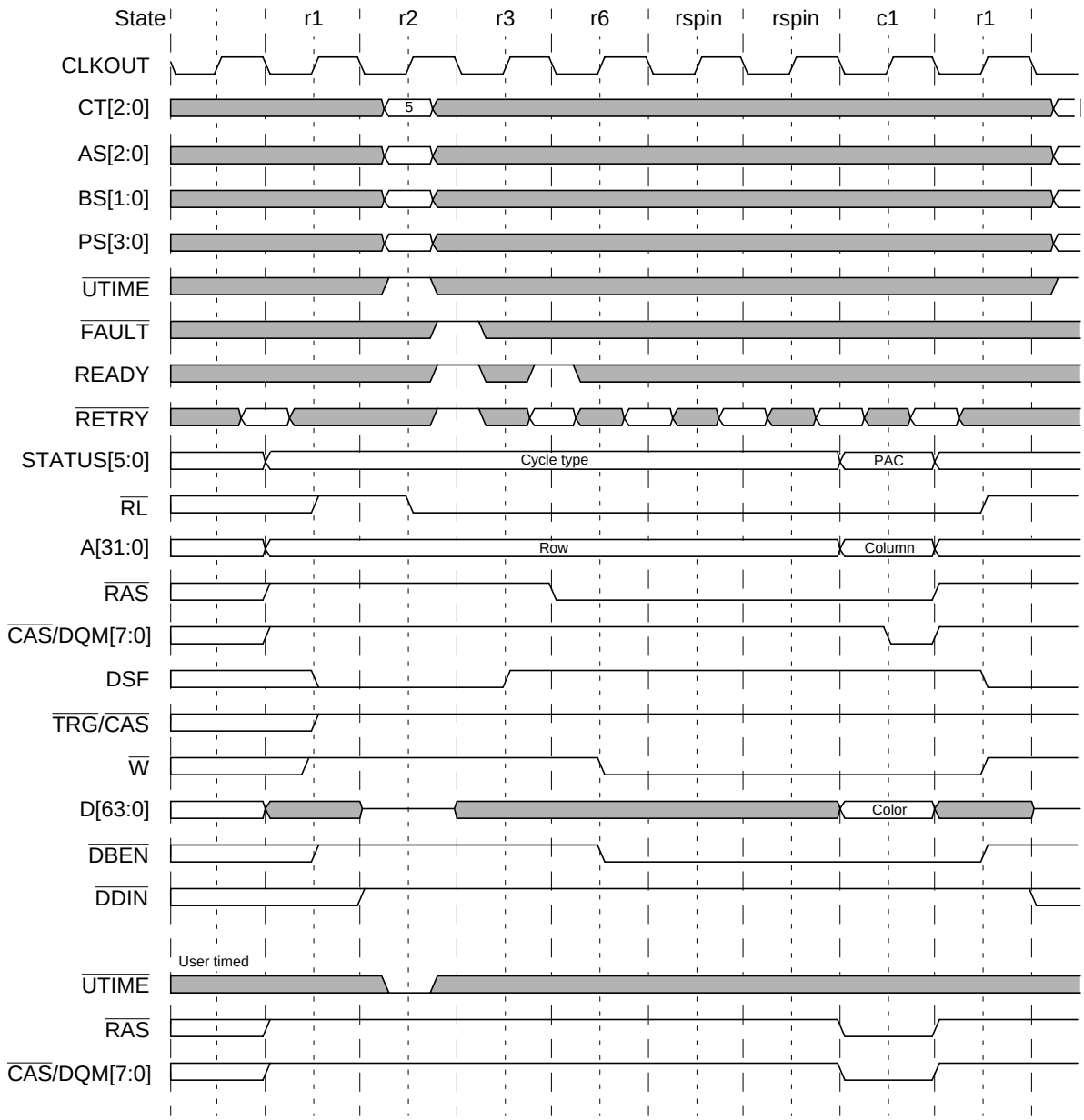
12.1.2 Nonpipelined One-Cycle-per-Column Load-Color-Register Cycle

Figure 12–2 shows an LCR cycle to nonpipelined, one-cycle-per-column memory.

The example in Figure 12–2 shows a single load of the VRAM color registers. The state sequence is identical to that of a nonpipelined one-cycle-per-column write. (See section 9.4, *Nonpipelined One-Cycle-per-Column Write*.) The row access consists of the r1, r2, r3, r6 and rspin states. The rspin state always occurs at least twice. Additional r3 and/or r6 states can be inserted by adding wait states. The column access consists of a single c1 state during which the 64-bit color register value is output on the data bus. The memory interface then returns to state r1, where it remains until the next row access begins.

The bottom of Figure 12–2 shows how the cycle is modified when user timing is selected.

Figure 12–2. Nonpipelined One-Cycle-per-Column Load-Color-Register



Note: BS[1:0] must have a value of 10 or 11 in order for an LCR cycle to occur.

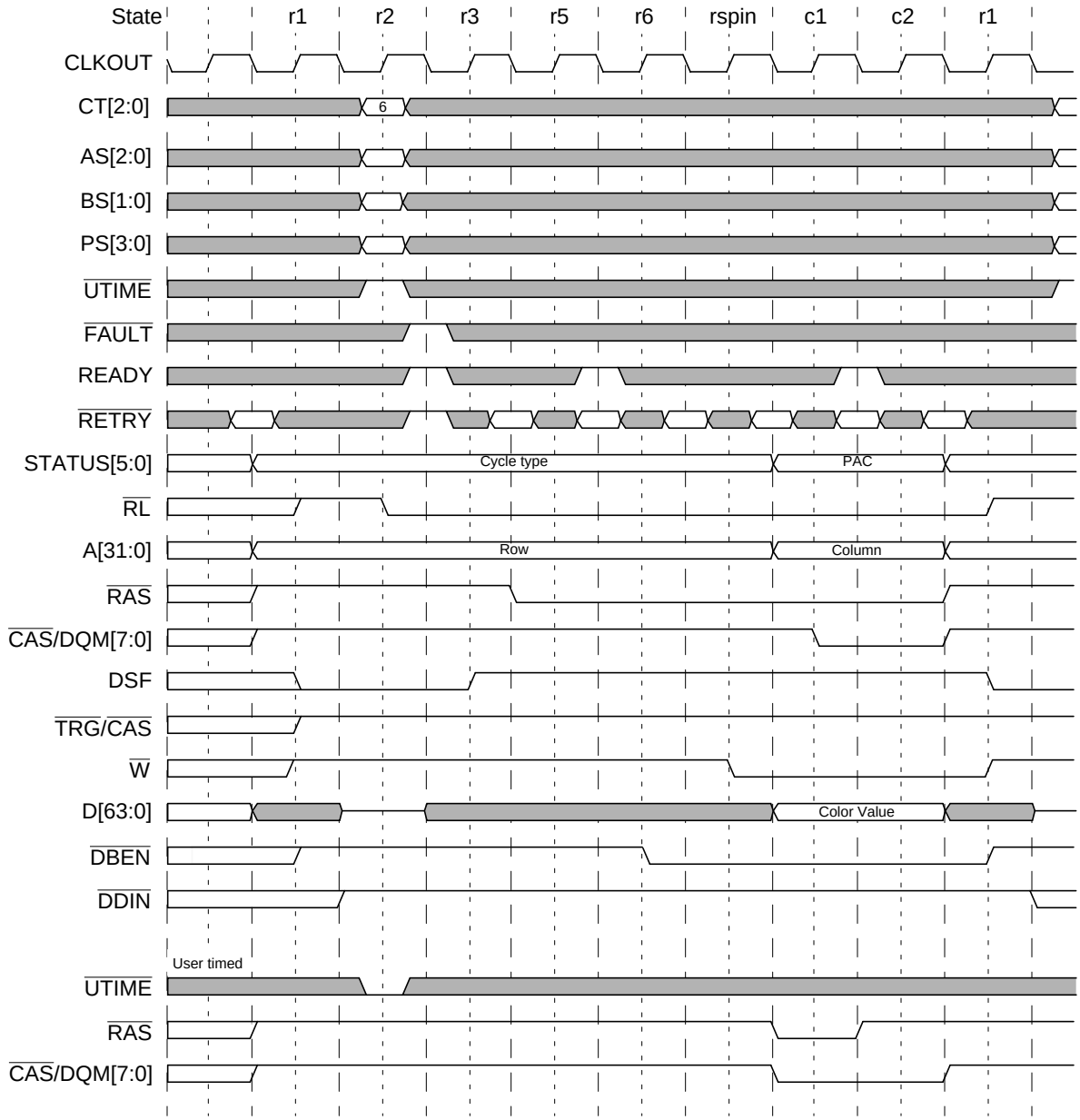
12.1.3 Two-Cycles-per-Column Load-Color-Register Cycle

Figure 12–3 shows an LCR cycle to two-cycles-per-column memory.

The example in Figure 12–3 shows a single load of the VRAM color registers. The state sequence is identical to that of the two-cycle-per-column write. (See section 9.6, *Two-Cycles-per-Column Write.*) The row access consists of the r1, r2, r3, r5, r6, and rspin states. The rspin state always occurs at least one time. Additional r3 and/or r6 states may be inserted by adding wait states. The column access consists of a c1 state, during which the 64-bit color register value is output on the data bus, and a c2 state. The memory interface then returns to state r1, where it remains until the next row access begins.

The bottom of Figure 12–3 shows how the cycle is modified when user timing is selected.

Figure 12–3. Two-Cycles-per-Column Load Color Register



Note: BS[1:0] must have a value of 10 or 11 in order for an LCR cycle to occur.

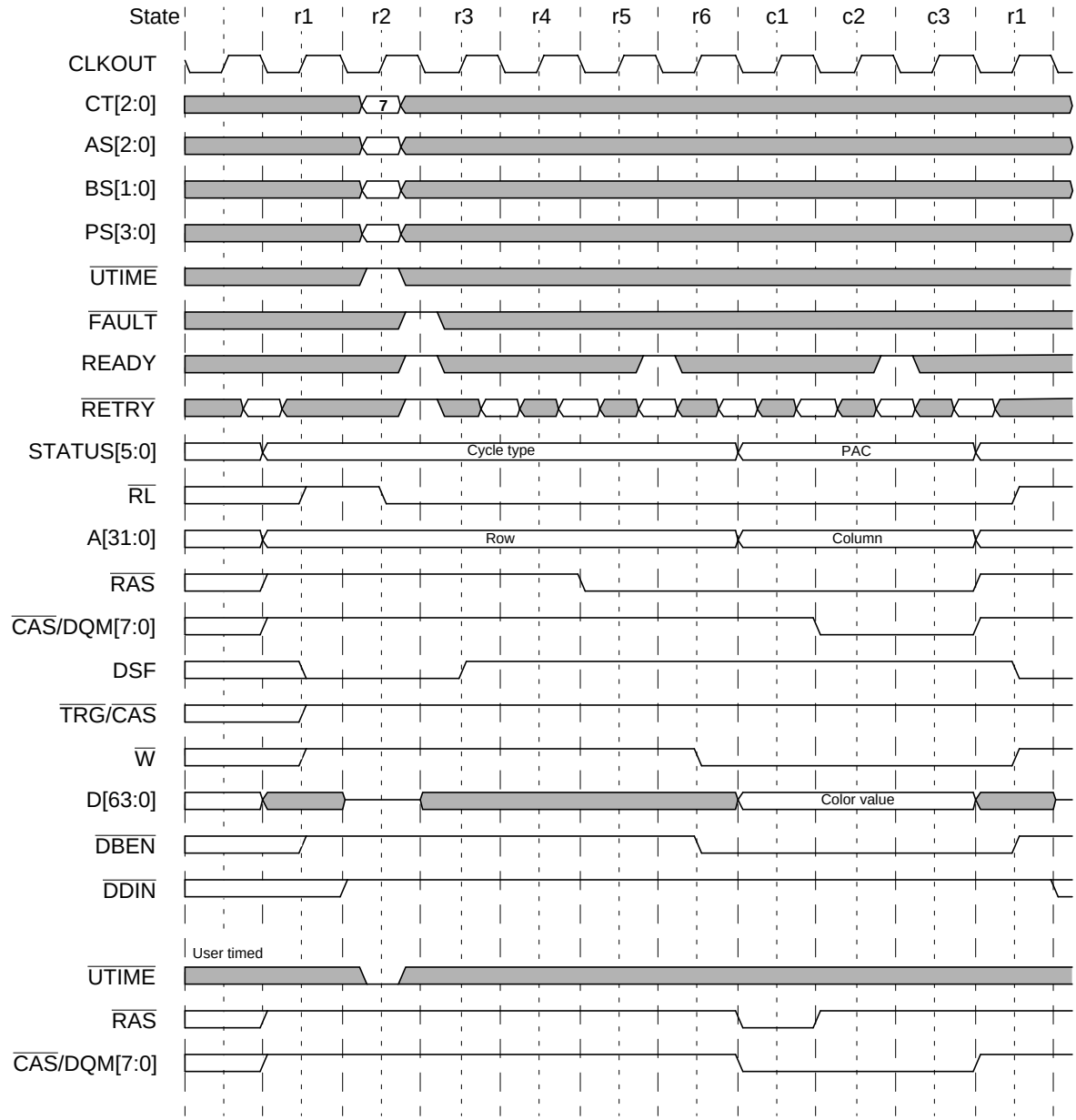
12.1.4 Three-Cycles-per-Column Load-Color-Register Cycle

Figure 12–4 shows an LCR cycle for a three-cycles-per-column memory.

The example in Figure 12–4 shows a single load of the VRAM color registers. The state sequence is identical to that of a three-cycles-per-column write. (See section 9.8, *Three-Cycles-per-Column Write*.) The row access consists of the r1, r2, r3, r4, r5, r6 and possibly rspin states. Additional r3 and/or r6 states can be inserted by adding wait states. The column access consists of a c1, c2, and c3 state. The 64-bit color register value is output on the data bus during c1 and strobed into the VRAMs with $\overline{\text{CAS}}$/DQM during c2. After the c3 state, the memory interface returns to state r1, where it remains until the next row access begins.

The bottom of Figure 12–4 shows how the cycle is modified when user timing is selected.

Figure 12–4. Three-Cycles-per-Column Load-Color-Register



Note: BS[1:0] must have a value of 10 or 11 in order for an LCR cycle to occur.

12.1.5 SGRAM Special-Register-Set (Load Color Register) Cycle

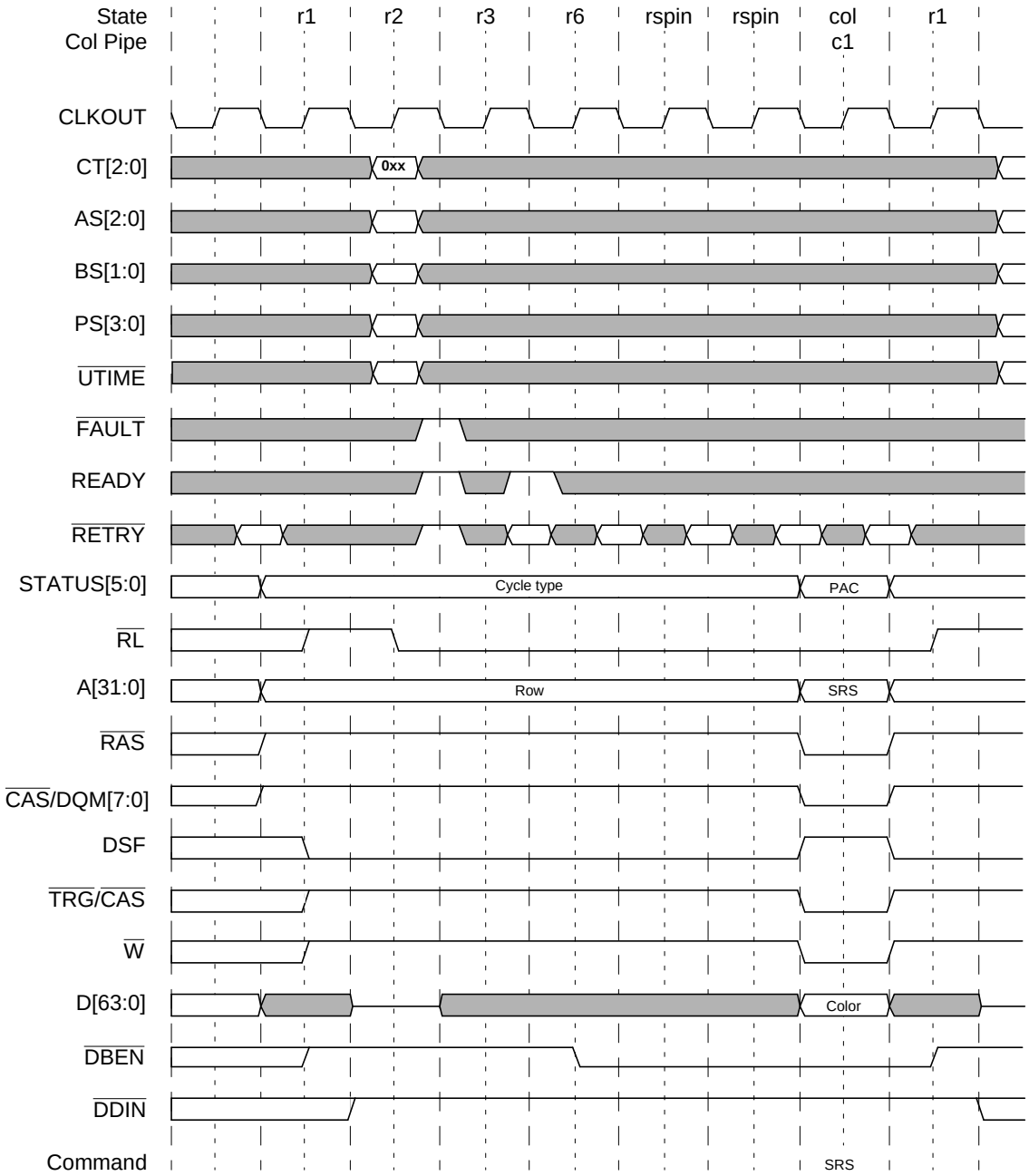
Figure 12–5 shows an SGRAM special-register-set (SRS) cycle.

If the transfer controller samples a CT value of 0xx during a load color-register cycle, the access is converted into an SGRAM SRS cycle. This cycle should only be performed on SGRAMs. Allowing this cycle to be performed on SDRAM could corrupt the mode register.

The SRS cycle consists of a single column access. The special-register-select information is output on the address bus as described in section 10.1.4, *Loading the SGRAM Color Register*. When performing an SRS cycle, the transfer controller always selects the load-color-register operation. The color written on the data bus is the value in the 64-bit color register contained in the packet-transfer parameters for the associated block write.

Since block-write operations are only supported with 64-bit buses, all $\overline{\text{CAS}}$/DQM strobes are active during the SRS command.

Figure 12–5. SGRAM Special Register Set (Load Color Register)



12.2 Block-Write Cycles

Block-write cycles cause the data stored in the VRAM/SGRAM color registers to be written to the memory locations enabled by the appropriate data bits output on the D[63:0] bus. This allows as many as 64 bytes (depending on the type of block write being used) to be written in a single column access. A status code of 001001 indicates a block-write cycle. This cycle is identical to a standard write cycle with the following exceptions:

- ❏ DSF is high (active) at the fall of $\overline{\text{CAS}}$/DQM, enabling the block-write function within the VRAM/SGRAMs.
- ❏ Only 64-bit bus sizes are supported during block write. Therefore, BS[1:0] inputs are used to indicate the type of block write that is supported by the addressed VRAM/SGRAM, rather than to indicate the bus size.
- ❏ The two or three LSBs (depending on the type of block write) of the column address are ignored by the VRAM/SGRAMs because these column locations are specified by the data inputs.
- ❏ The values output by the TC on D[63:0] represent the column locations to be written to using the color-register value. Depending on the type of block write supported by the VRAM/SGRAM, all of the data bits may not be used by the VRAM/SGRAMs.
- ❏ Block writes always begin with a row access. Upon completion of a block write, the memory interface returns to state r1 to await the next access.

Three block-write modes are supported by the TC: 4X, 8X, and simulated. For more information see section 11.3, *The Block-Write Modes*.

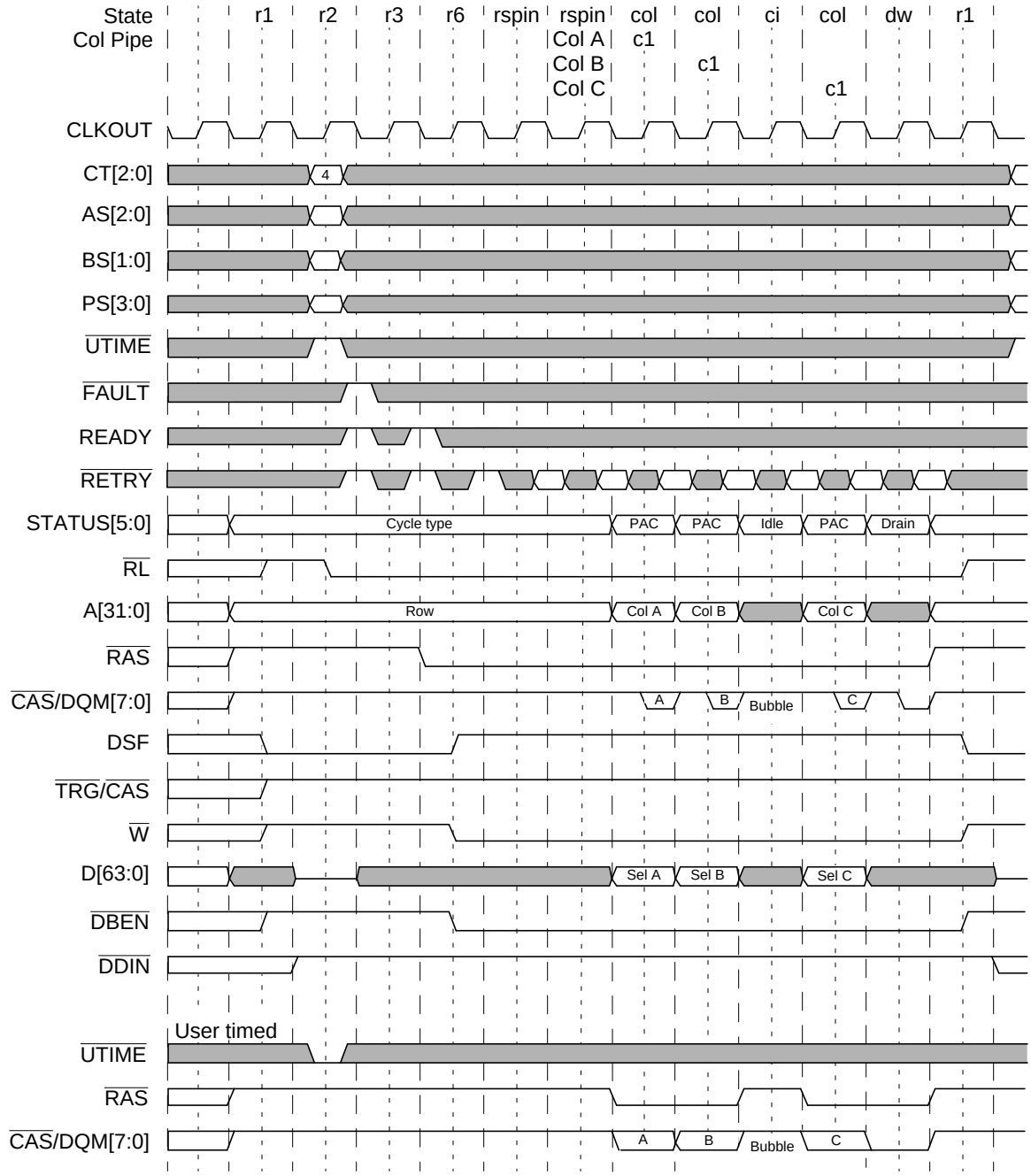
12.2.1 Block Writes to Standard VRAM

Figure 12–6, Figure 12–7, Figure 12–8, and Figure 12–9 show examples of block-write cycles to standard VRAM for pipelined one-cycle-per-column, nonpipelined one-cycle-per-column, two-cycles-per-column, and three-cycle-per-column memories, respectively. The state sequence for each of these cycles is identical to its corresponding standard write cycle.

Note:

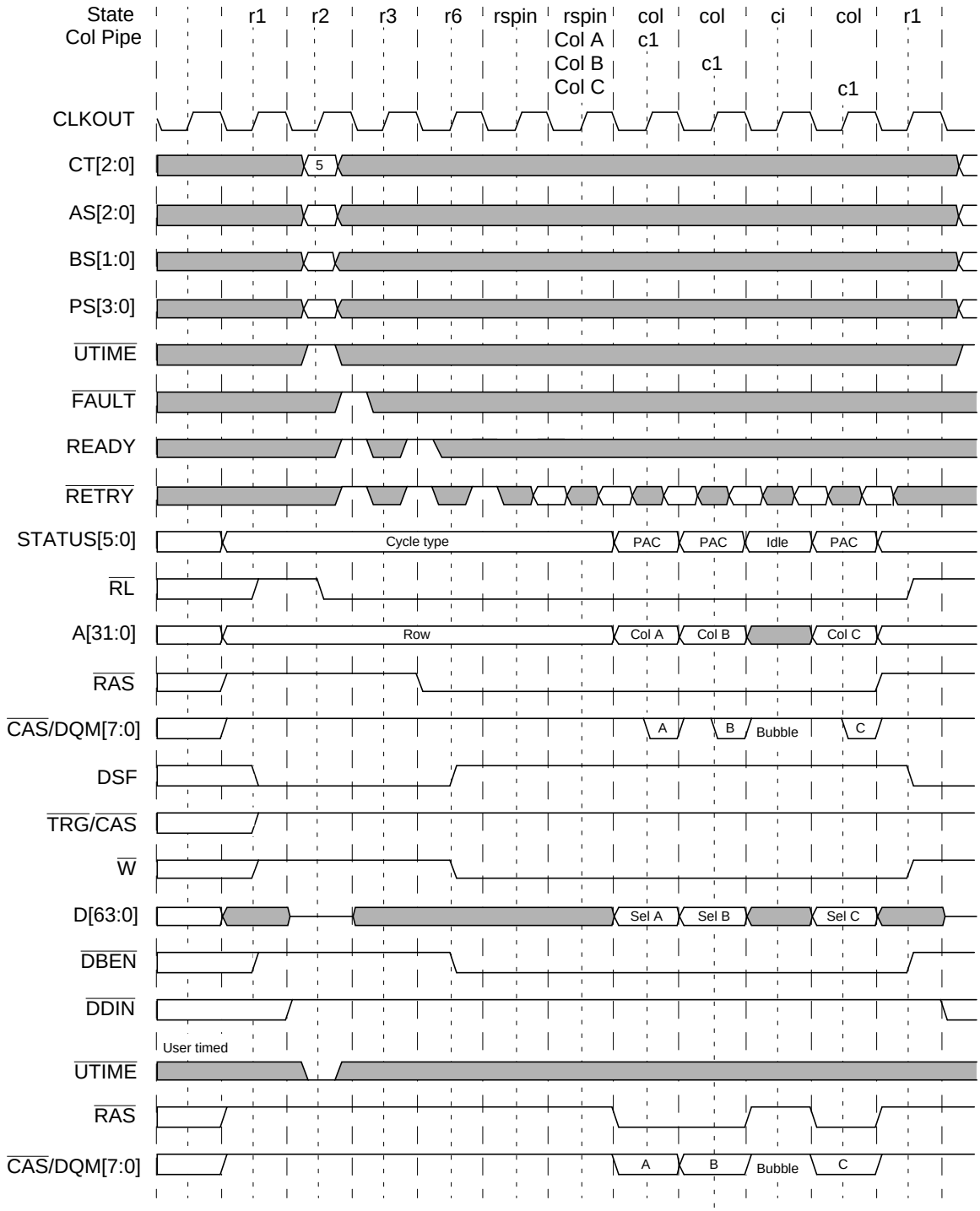
VRAMs that support pipelined page mode may support only nonpipelined block-write cycles. For these devices, select a nonpipelined memory timing during block-write cycles.

Figure 12–6. Pipelined One-Cycle-per-Column Block Write



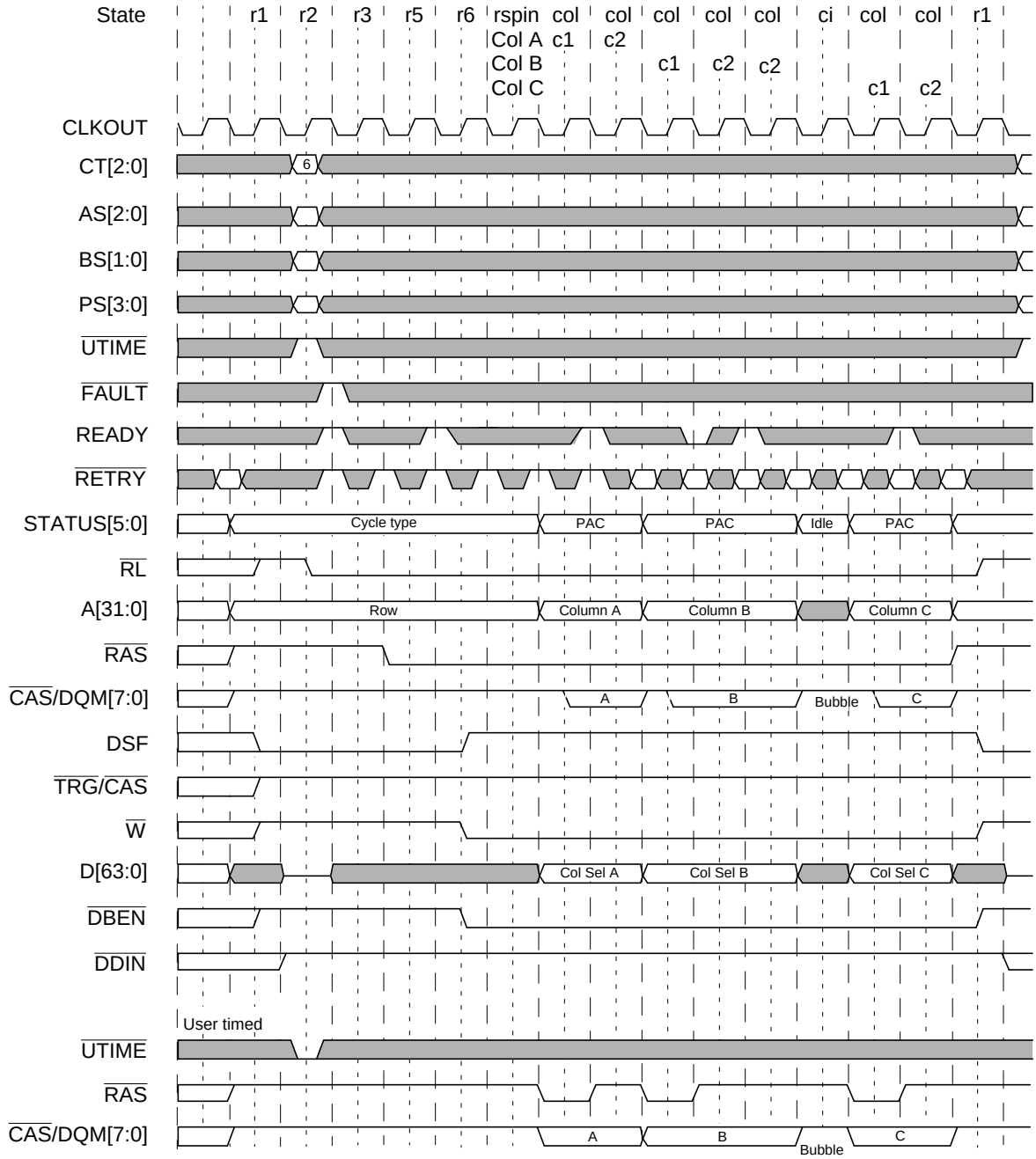
Note: BS[1:0] must have a value of 10 or 11 in order for a block write to occur.

Figure 12–7. Nonpipelined One-Cycle-per-Column Block Write



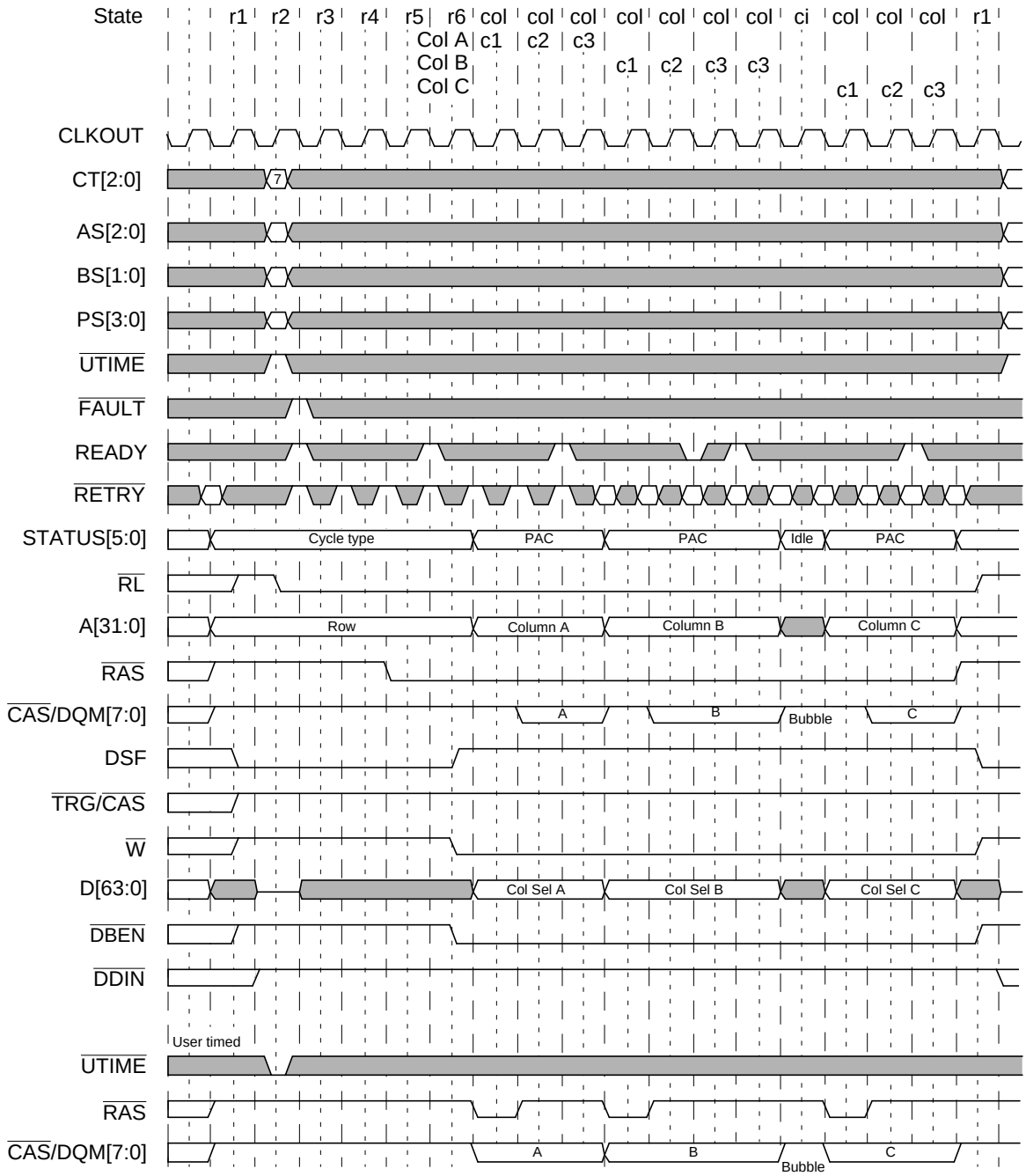
Note: BS[1:0] must have a value of 10 or 11 in order for a block write to occur.

Figure 12–8. Two-Cycles-per-Column Block Write



Note: BS[1:0] must have a value of 10 or 11 in order for a block write to occur.

Figure 12–9. Three-Cycles-per-Column Block Write



Note: BS[1:0] must have a value of 10 or 11 in order for a block write to occur.

12.2.2 Block Writes to SGRAM

A block-write cycle is similar to a normal SDRAM write cycle, except that the DSF signal is high. The value output on the data bus selects the columns to be written to by the SGRAM color register. Because block-write cycles are supported on 64-bit buses only, all $\overline{\text{CAS}}$/DQM outputs are low during the command. The TC always inserts an idle state after the last block write, as required by SGRAM timing. The TC then deactivates the banks with a deactivate (DCAB) command before beginning a new page.

If the sampled values of the CT[2:0] inputs are 00x during a block-write cycle, the TC generates a one-cycle-per-column SGRAM block-write cycle. Cycle timing is similar to SDRAM writes with a burst length one. This cycle should only be performed on SGRAMs that support a new column address every clock cycle. Figure 12–10 shows a sequence of four block-write commands.

If the sampled values of the CT[2:0] inputs are 01x during a block-write cycle, the TC generates a SGRAM block-write cycle with each write taking two cycles to complete. The cycle timing is similar to SDRAM block writes with a burst length of two. However, bursts are not supported for block-write accesses, and the coding of CT[2:0], in this case, indicates that the SGRAM can accept a new column address only on *every other* clock cycle. Figure 12–11 shows a sequence of two block-write commands; each block-write command takes two clock cycles to complete.

Figure 12–10.SGRAM Burst Length 1 Column Block Write

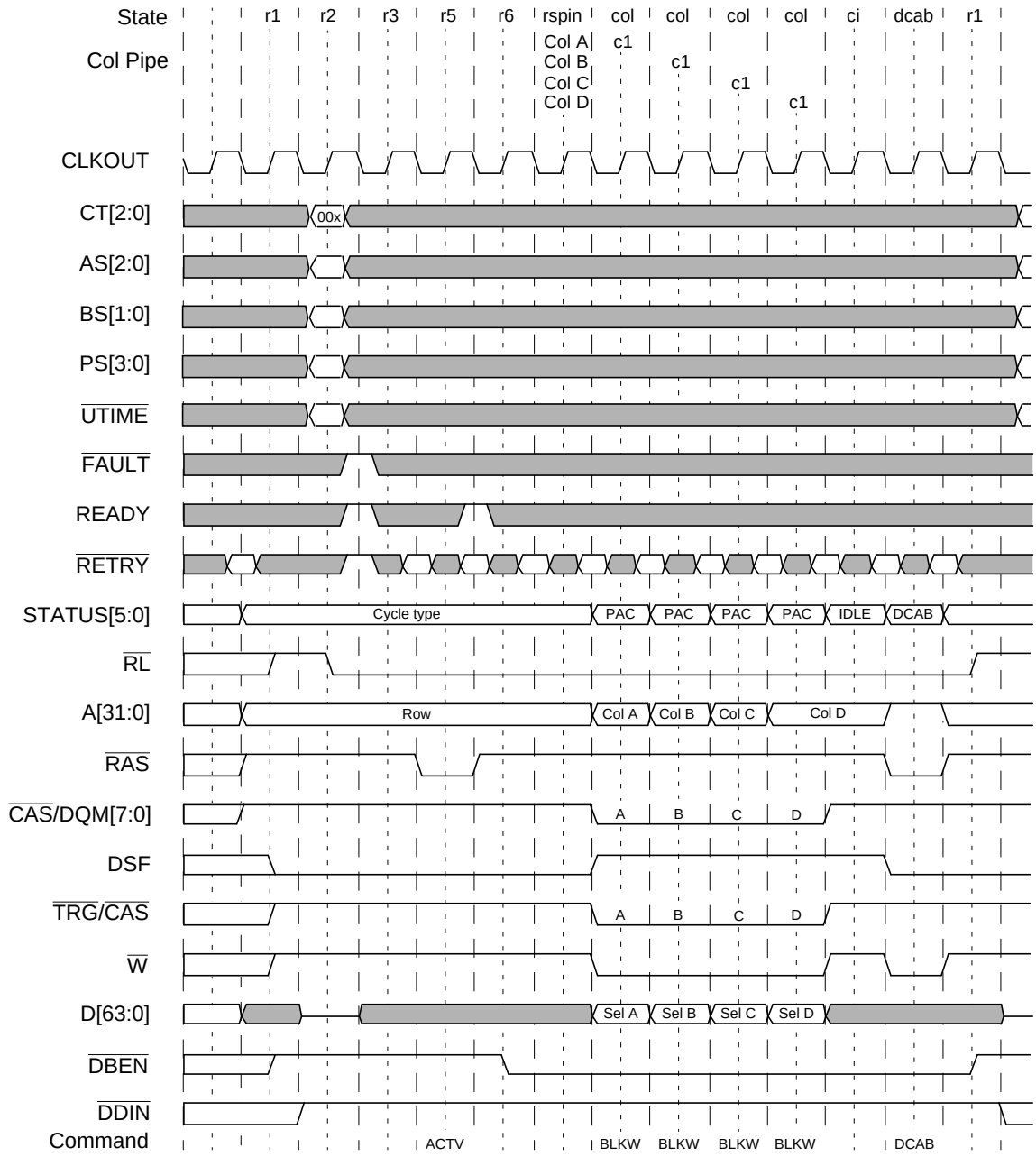
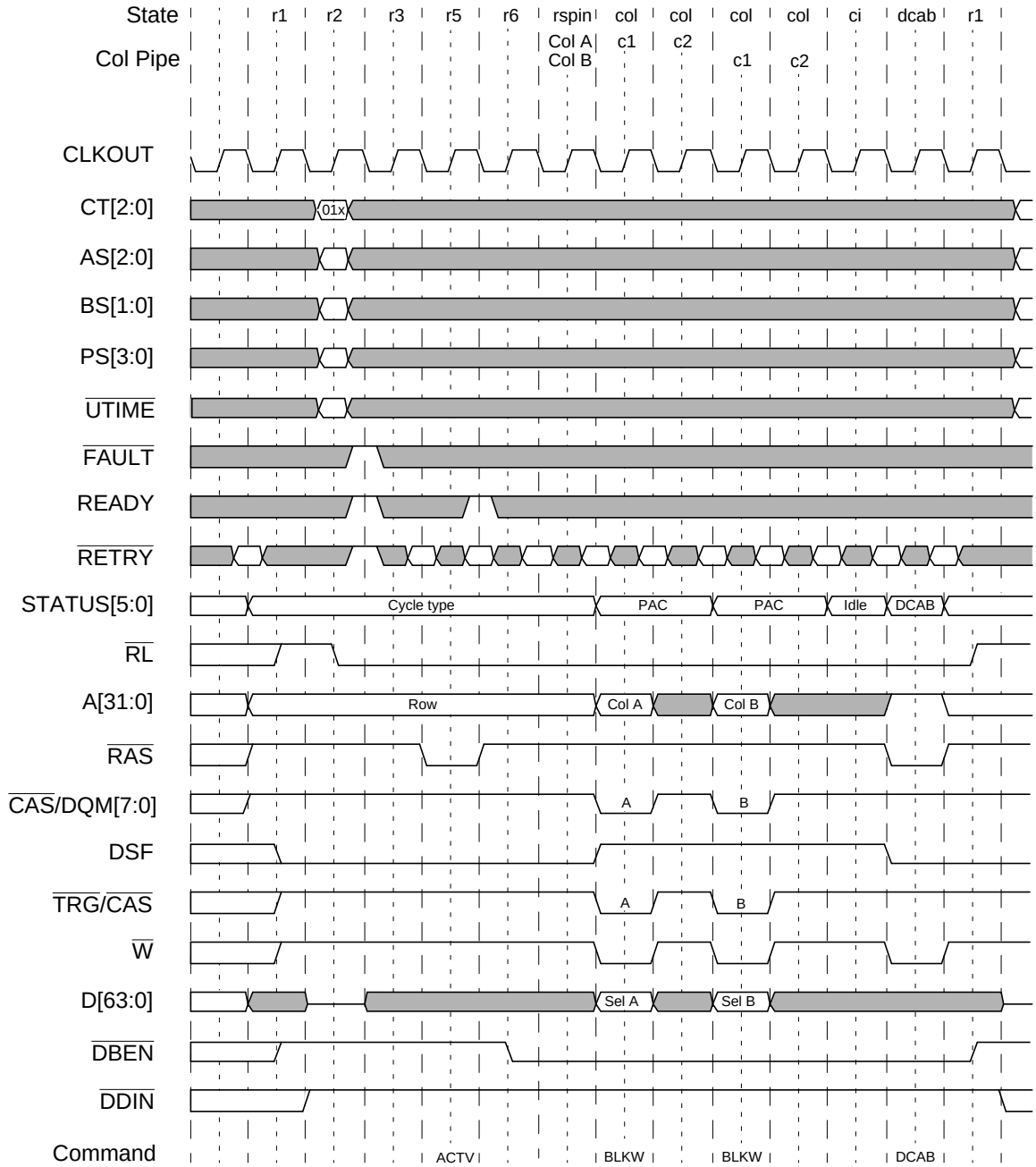


Figure 12–11.SGRAM Burst Length 2 Column Block Write



12.3 VRAM Read Transfer and Split Read Transfer Cycles

Read transfer (memory-to-register) cycles transfer a row from the VRAM memory array into the VRAM serial register (SAM). This causes the entire SAM (both split SAMs) to be loaded with the array data. During read-transfer cycles, the STATUS[5:0] bus has one of the following values, depending on the origin of the cycle request:

010000 The transfer was requested by the video controller for frame memory 0.

010100 The transfer was requested by the video controller for frame memory 1.

011100 The transfer was requested by an SRT controller packet transfer (see section 11.5, *Serial Register Packet Transfers*).

Split read-transfer (memory-to-split-register) cycles also transfer data from a row in the memory array to the SAM. However, this transfer causes only half of the SAM to be written. It is generated when a SAM overflow event occurs in the video controller, or when the two split SAMs need to be loaded with data from two different memory rows. Split read transfers allow the inactive half of the SAM to be loaded with new data while the other active half continues to shift data in or out.

Split read transfers have either of the two following status codes on STATUS[5:0]:

010010 The transfer was requested by the video controller for frame memory 0.

010110 The transfer was requested by the video controller for frame memory 1.

The state sequence for read transfer cycles is identical to that for single-access read cycles. The transfer cycles are differentiated only by the following signal behavior:

- For standard VRAM, $\overline{TRG}/\overline{CAS}$ timing is different: $\overline{TRG}/\overline{CAS}$ is low at the fall of $\overline{RAS}$ and goes high prior to $\overline{RAS}$ rising.
- $\overline{DBEN}$ remains high (inactive) during the cycle because no data is transferred over the data bus.
- The data bus (D[63:0]) is placed in high impedance.
- The value output on A[31:0] at column time represents the SAM tap point.
- Because only a single column access is made, page-mode accesses do not occur. The transfer always begins with a row access and returns to state r1 immediately upon completion.

- Only one column operation is performed; column time retries have no effect. However, $\overline{\text{RETRY}}$ must be at a valid level when sampled during the cycle.
- The $\overline{\text{FAULT}}$ input is not sampled during state r3. No faulting is supported. (This is true only if the transfer is VC initiated.)

The timing for the split read transfer is identical to that for the read transfer cycle with the following exception: DSF is high (active) at the fall of $\overline{\text{RAS}}$ and remains high throughout the cycle to enable split read transfers on the VRAMs.

Figure 12–12, Figure 12–13, Figure 12–14, and Figure 12–15 show examples of pipelined one-cycle-per-column, nonpipelined one-cycle-per-column, two-cycles-per-column, and three-cycles-per-column (split) read transfers, respectively.

Note:

VRAMs that support pipelined page mode may support only nonpipelined read transfer and split read transfer cycles. For these devices, select a nonpipelined memory timing during read transfer and split read transfer cycles.

Figure 12–12.Pipelined One-Cycle-per-Column Full/Split Read Transfer

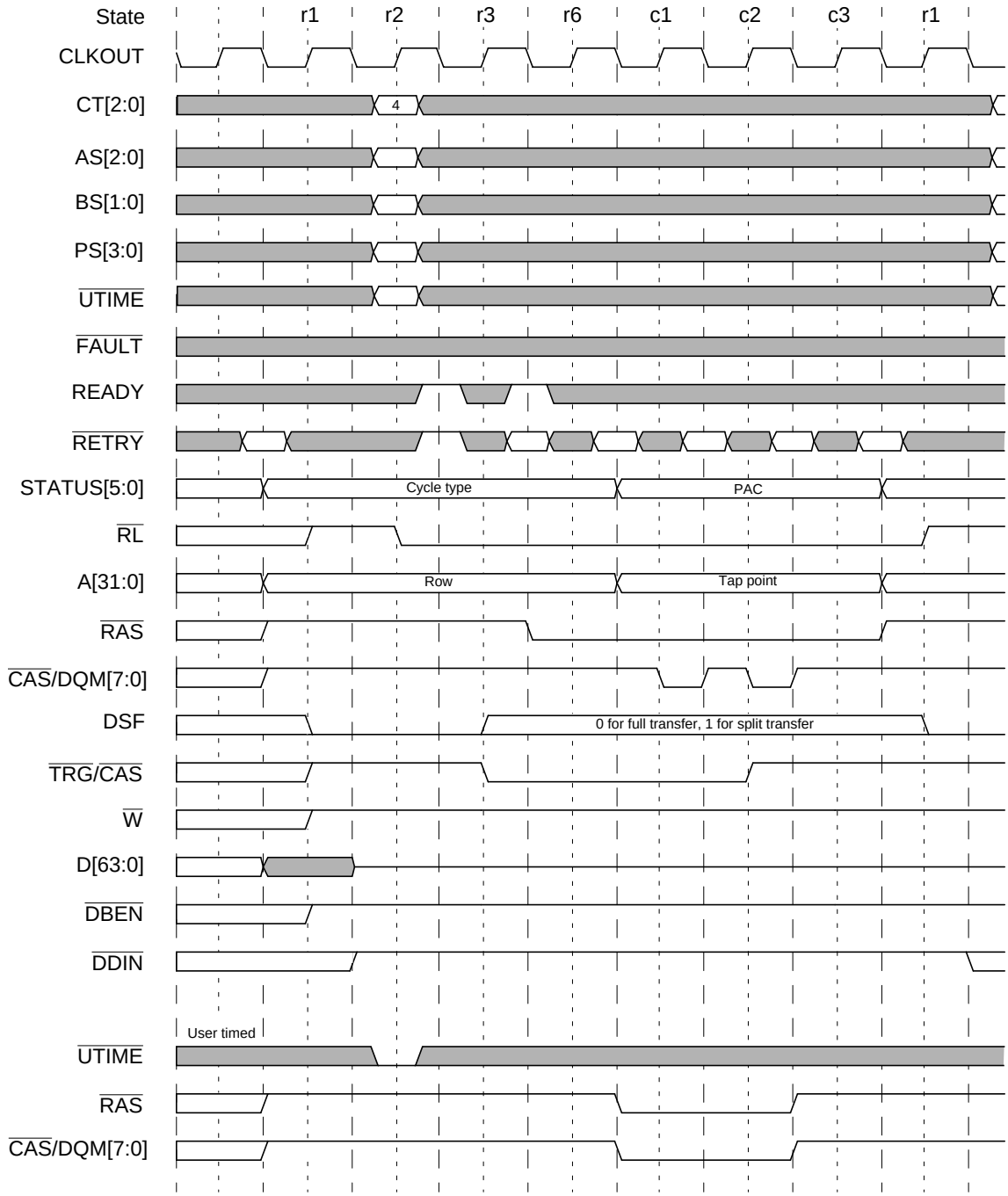


Figure 12–13.Nonpipelined One-Cycle-per-Column Full/Split Read Transfer

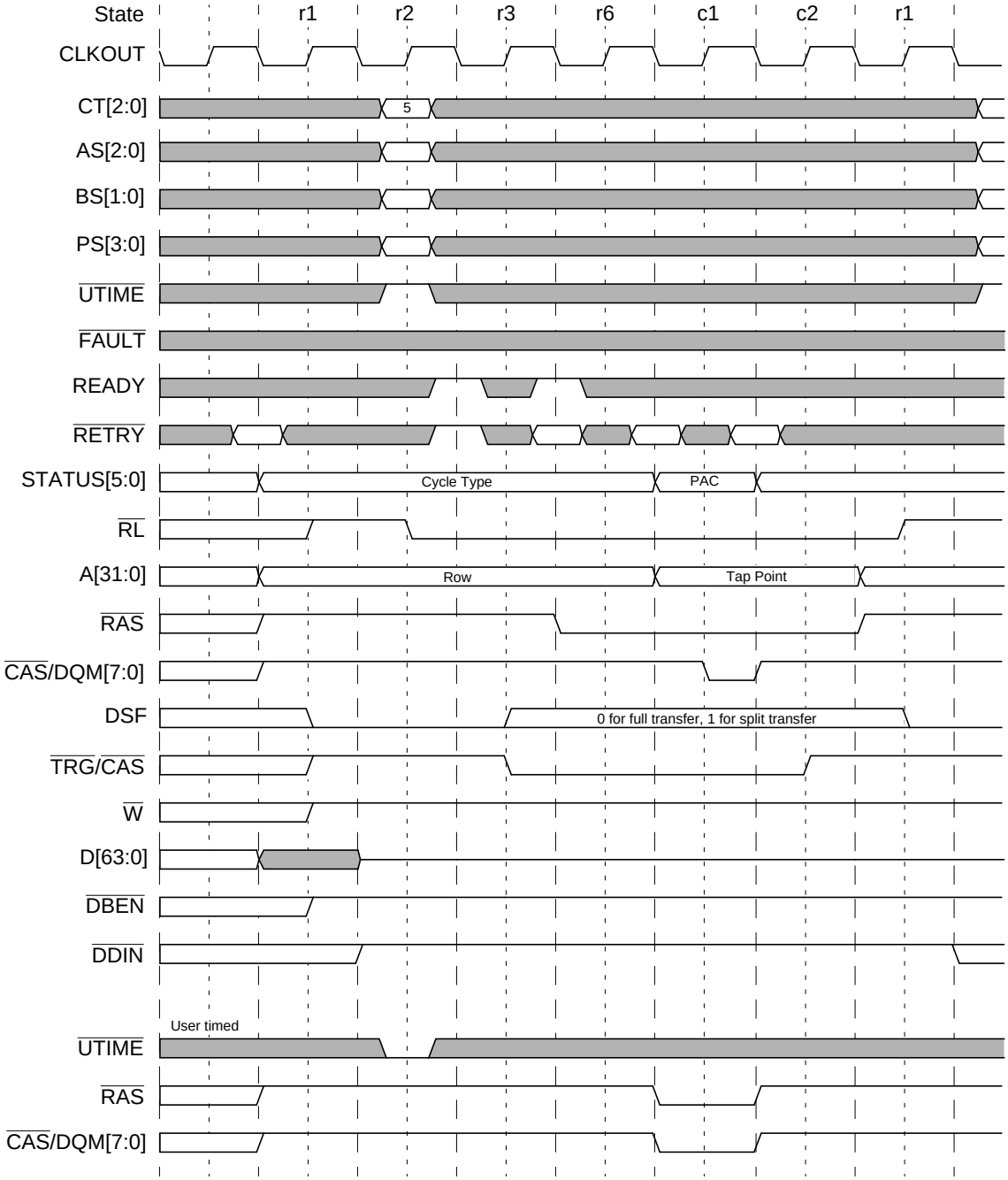


Figure 12–14. Two-Cycles-per-Column Full/Split Read Transfer

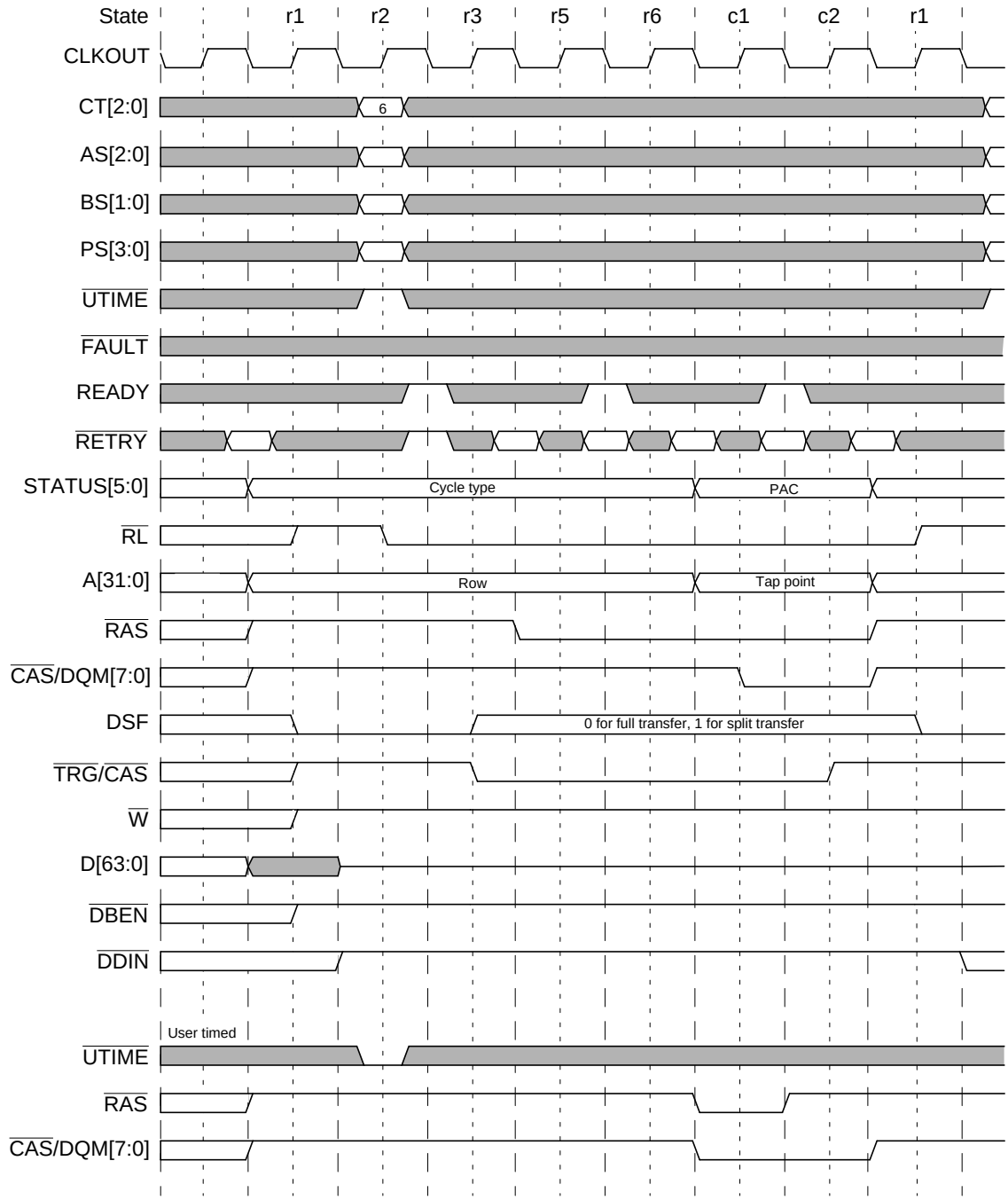
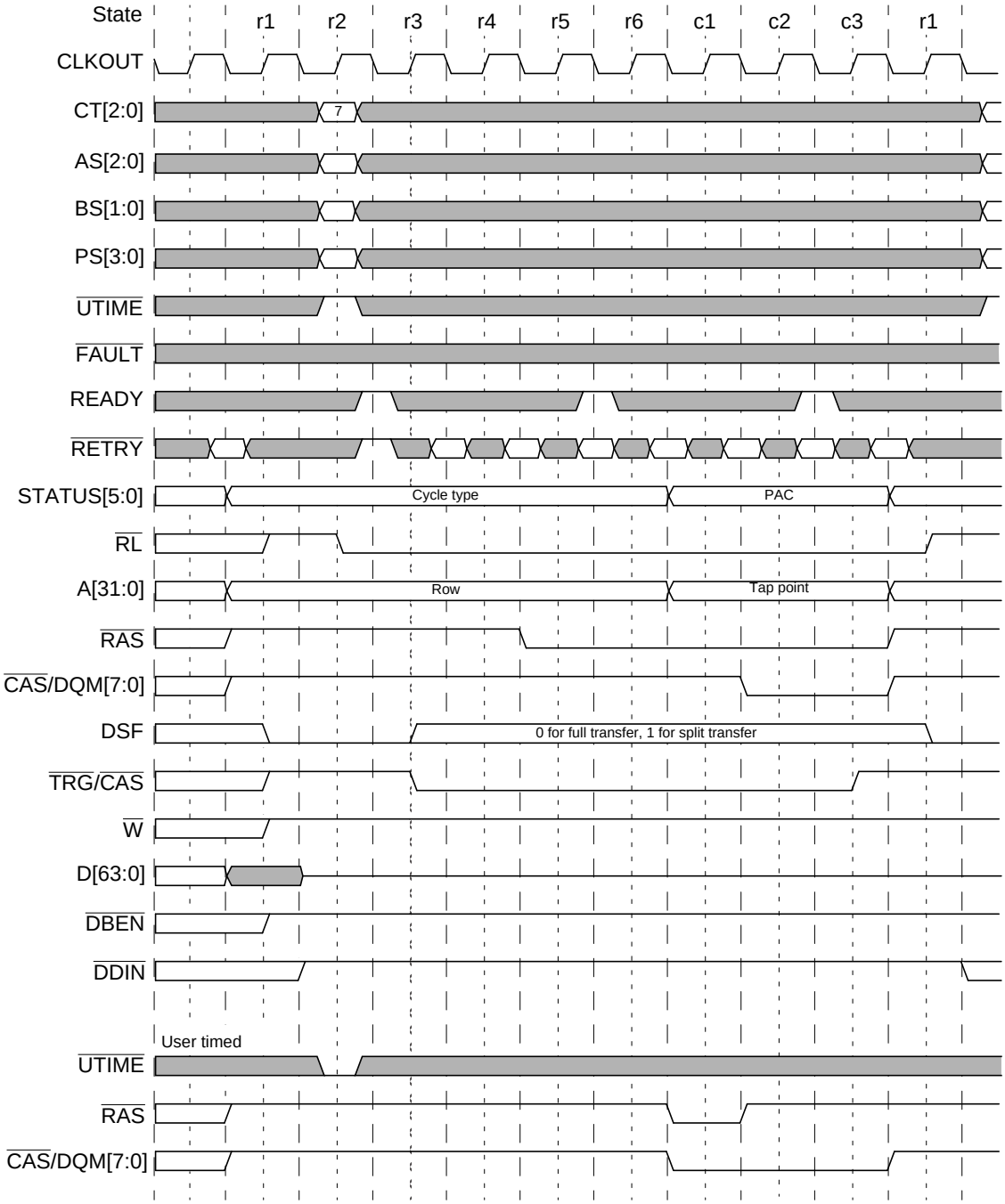


Figure 12–15.Three-Cycles-per-Column Full/Split Read Transfer



12.4 VRAM Write Transfer and Split Write Transfer Cycles

Write transfer (register-to-memory) cycles transfer data from the SAM into a row of the VRAM memory array. This transfer causes the entire SAM (both split SAMs) to be written into the array. During write-transfer cycles, the STATUS[5:0] bus has one of the following values, depending on the origin of the cycle request:

010001 The transfer was requested by the video controller for frame memory 0.

010101 The transfer was requested by the video controller for frame memory 1.

011101 The transfer was requested by a packet transfer (see section 11.5, *Serial Register Packet Transfers*).

Split write-transfer (split-register-to-memory) cycles also transfer data from the SAM to a row in the memory array. However, this transfer writes only half of the SAM into the array. Split write transfers allow the inactive half of the SAM to be transferred into memory while the active half continues to shift serial data in or out.

Split write transfers have either of the two following status codes on STATUS[5:0]:

010011 The transfer was requested by the video controller for frame memory 0.

010111 The transfer was requested by the video controller for frame memory 1.

The state sequence for write transfer cycles is identical to that for single-access write cycles. The transfer cycles are differentiated only by the following signal behavior:

- For standard VRAM, $\overline{TRG}/\overline{CAS}$ and $\overline{W}$ are low at the fall of $\overline{RAS}$, enabling the write transfer function on the addressed VRAMs.
- The data bus (D[63:0]) is driven to all 1s before the fall of $\overline{RAS}$ to disable the write-transfer masks within the VRAMs. (The TMS320C80 does not support masked write transfers.)
- The value output on A[31:0] at column time represents the SAM tap point. This determines which bit within the SAM is aligned to column 0 when the data is written into the selected memory row.
- Because only a single column access is made, page-mode accesses do not occur. The transfer begins with a row access and returns to state r1 immediately upon completion.

- Only one column operation is performed; column-time retries have no effect. However, $\overline{\text{RETRY}}$ must be at a valid level when sampled during the cycle.
- $\overline{\text{FAULT}}$ input is not sampled during state r3. No faulting is supported. (This is true only if the transfer is VC initiated.)

The timing for the split write transfer is identical to that of the write transfer cycle with the following exception: DSF is high (active) at the fall of $\overline{\text{RAS}}$ and remains high throughout the cycle to enable split write transfers on the VRAMs.

Figure 12–16, Figure 12–17, Figure 12–18, and Figure 12–19 show examples of pipelined one-cycle-per-column, nonpipelined one-cycle-per-column, two-cycles-per-column, and three-cycles-per-column (split) write transfers, respectively.

Note:

VRAMs that support pipelined page mode may support only nonpipelined write transfer and split write transfers. For these devices, select a nonpipelined memory timing during write transfer and split write transfer cycles.

Figure 12–16.Pipelined One-Cycle-per-Column Full/Split Write Transfer

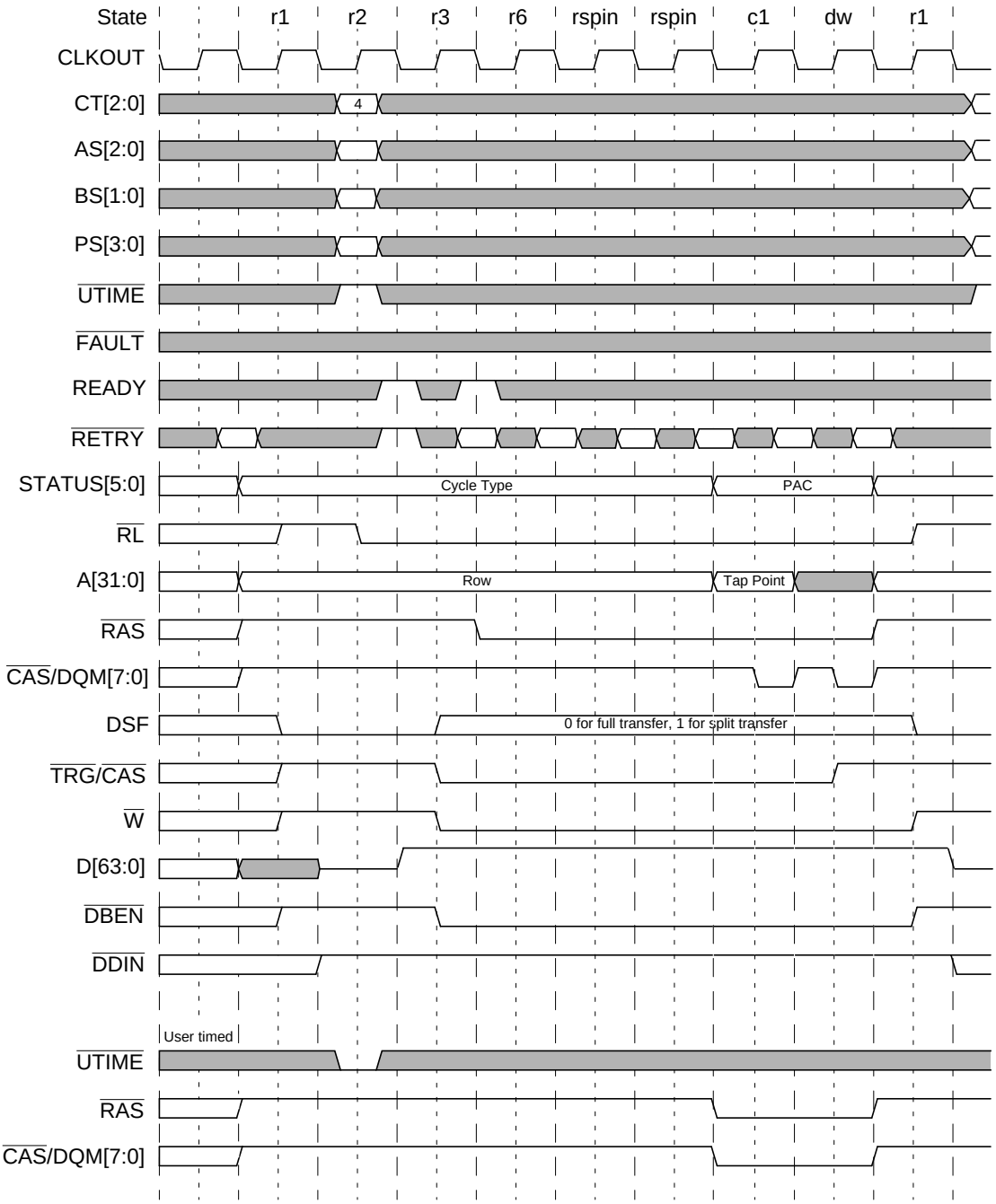


Figure 12–17.Nonpipelined One-Cycle-per-Column Full/Split Write Transfer

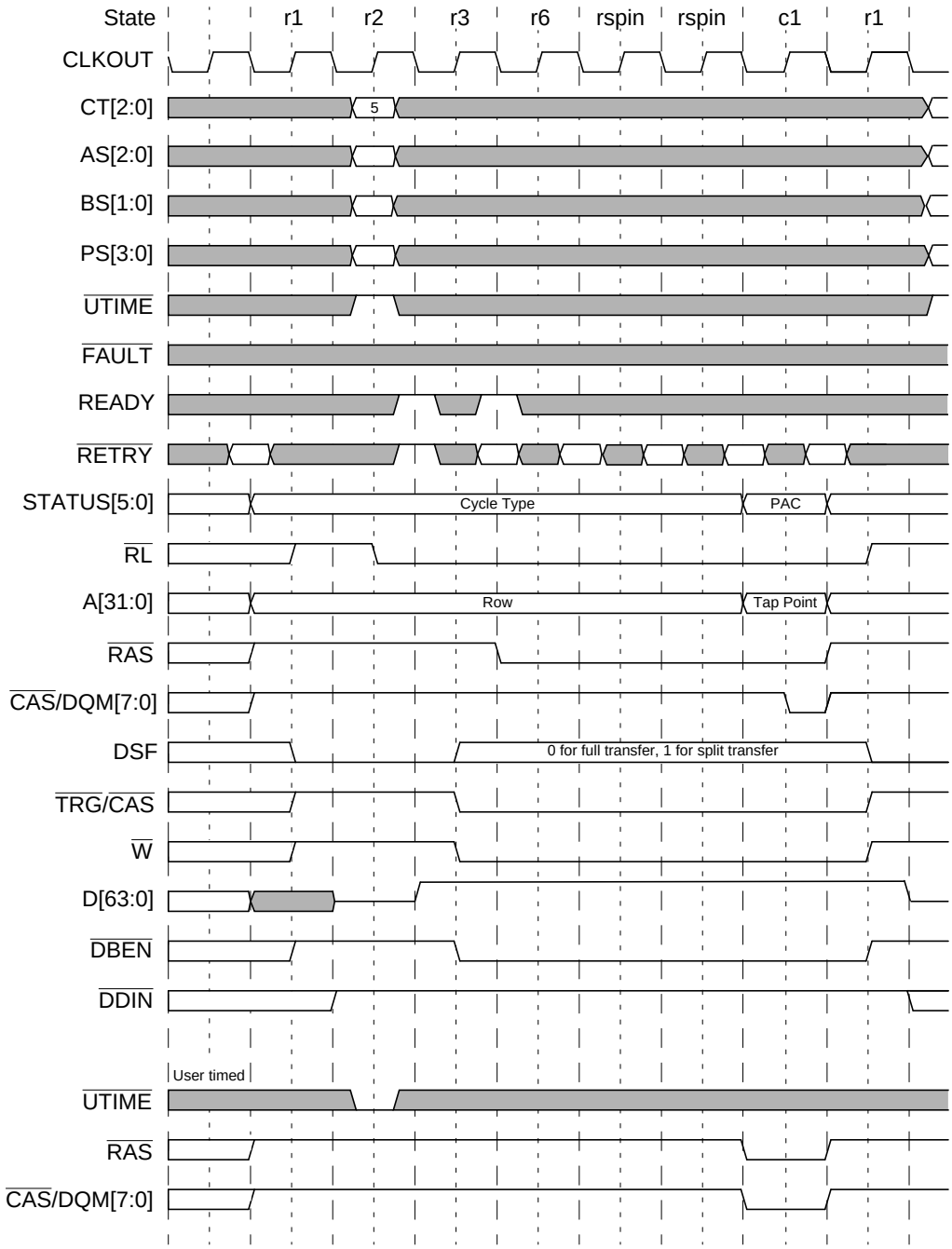


Figure 12–18. Two-Cycles-per-Column Full/Split Write Transfer

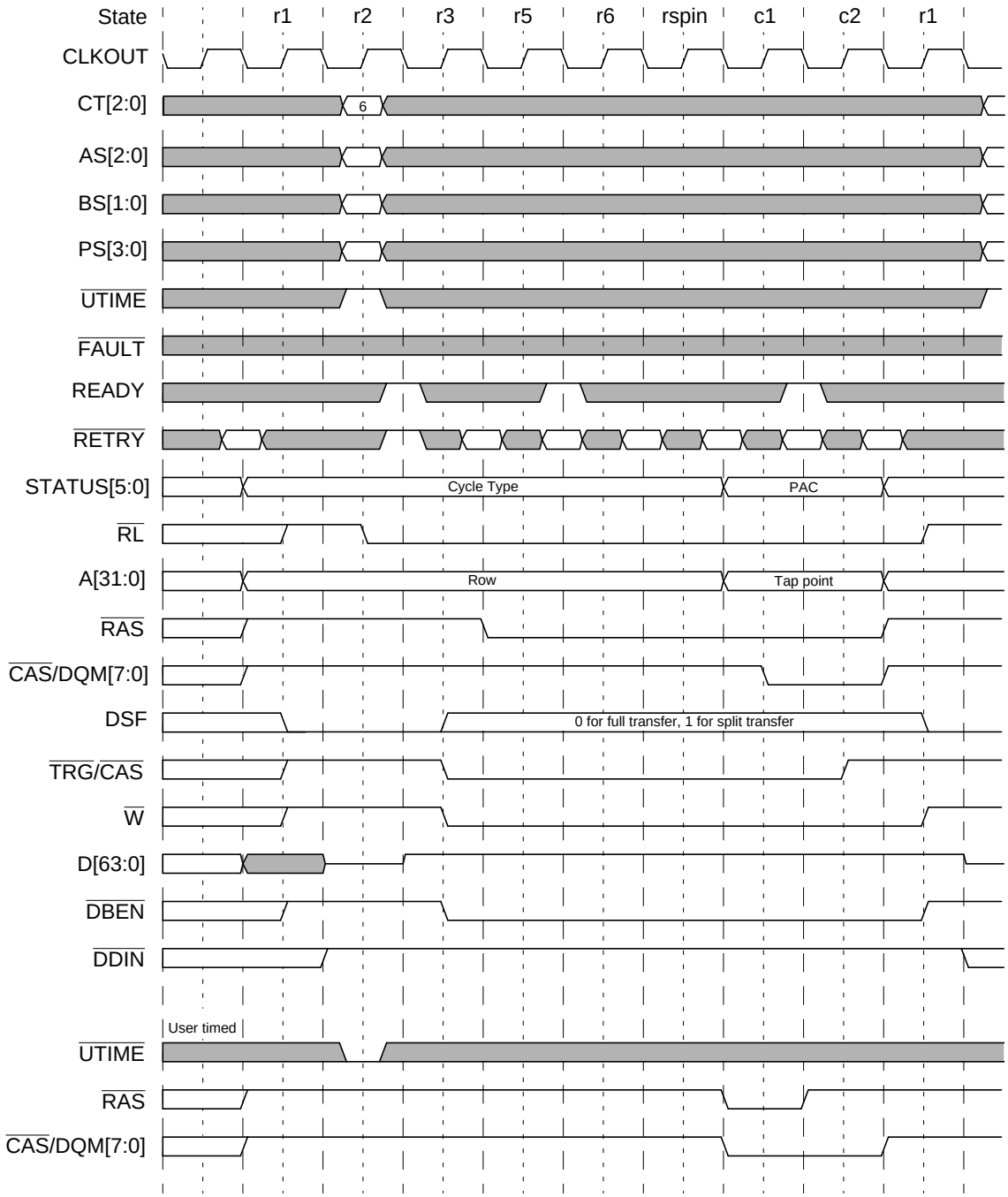
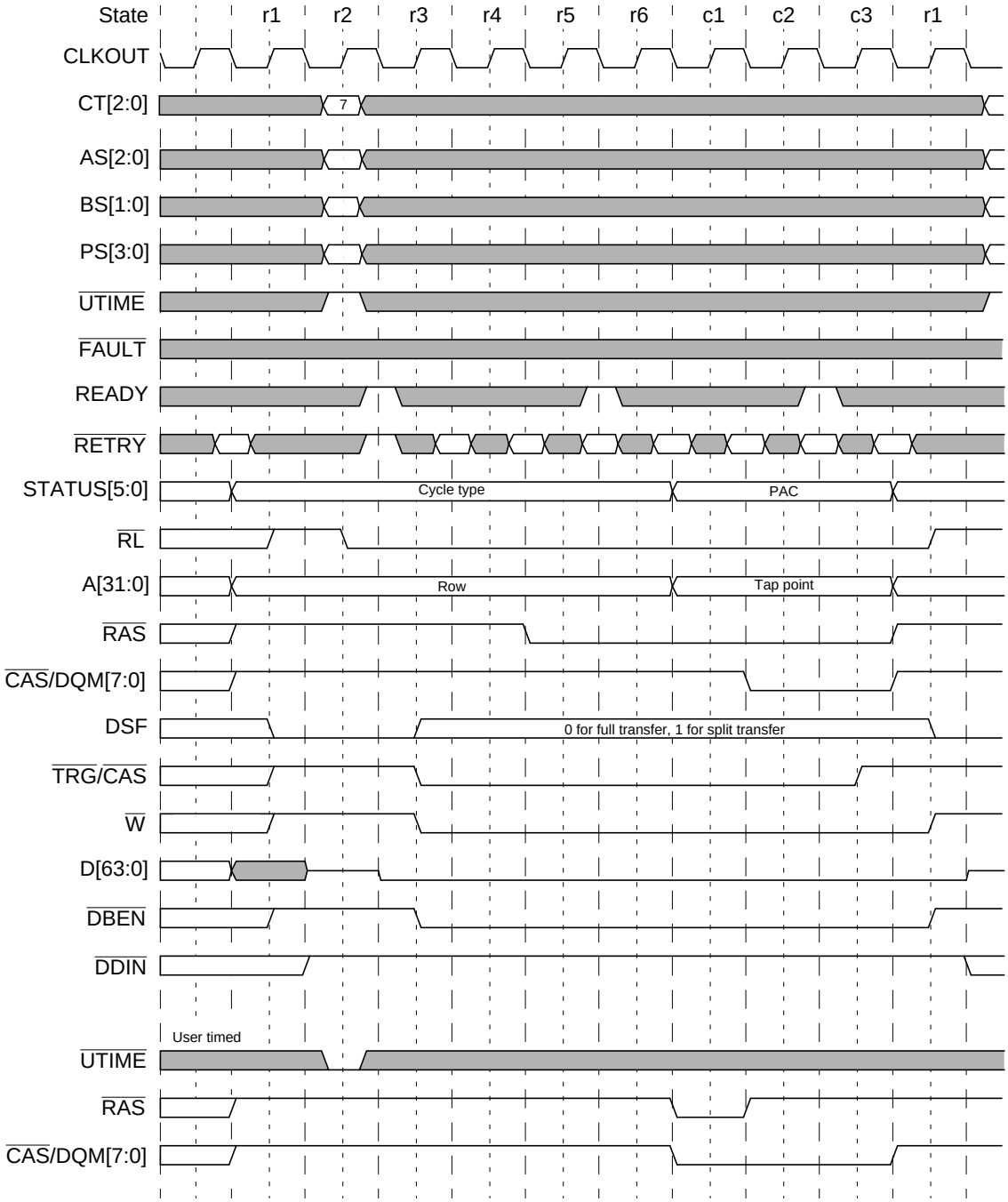


Figure 12–19.Three-Cycles-per-Column Full/Split Write Transfer



Chapter 13

Refresh Cycles

The TC's refresh controller logic generates refresh requests at regular intervals that you program. These refresh cycles occur in the form of $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycles The STATUS[5:0] pins output the refresh cycle status code at the beginning of the cycles, enabling external bank-select logic, which in turn generates appropriate $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals to the RAMs. Refresh cycles can be low priority *trickle* refresh cycles or high priority *urgent* refresh cycles.

Topics in this chapter include:

Topic **Page**

13.1 Refresh Rate

The refresh rate is defined in terms of ‘C80 clock cycles according to the value in the 16-bit REFRATE field of the REFCNTL register. This register is loaded with a value of 32 (0 x 20) at reset. You can modify the refresh rate by writing a new value into REFRATE. REFRATE values of less than 32 (0x20) disable refresh.

13.2 Refresh Priority

A requested refresh may have to wait because higher priority cycles are active on the bus. For this reason the TC stores a backlog of pending refresh requests. Each time a refresh is requested, the BACKLOG value is incremented. When the BACKLOG value reaches 16, an urgent refresh request is generated and gives the refresh controller higher bus priority.

When the bus is granted, the refresh controller performs a burst of at least four refresh cycles, decrementing the BACKLOG to 12. If the burst refresh is interrupted by a higher priority bus request (such as host cycle or SRT cycle), the higher priority request may take over and complete. The REQ pins indicate when the urgent refreshes are being performed, and if necessary, this can be used by external logic to prevent the host from requesting the bus.

The burst refresh then regains bus ownership and continues to decrement the BACKLOG to 12.

The refresh controller only requests the bus when an urgent refresh is required. When the external bus is idle, the bus prioritization logic allows the refresh controller to perform single refresh cycles to reduce any growing backlog. These trickle refresh cycles are only performed if the BACKLOG in nonzero.

At hardware reset, the BACKLOG is loaded with a value of 32. The prioritization logic prevents other bus requests from being granted until the backlog reaches 0. This provides the refresh cycles necessary for proper DRAM power-up initialization. No other requests (including host) are serviced until these have completed.

13.3 Programming the Refresh Pseudo-address

During refresh cycles, the refresh controller outputs a 16-bit refresh pseudo-address on the A[31:16] of the external bus. This address can be used to reduce system power consumption by refreshing banks of memory separately. It can also be used as the row address in systems that generate $\overline{\text{RAS}}$-only refresh cycles.

The upper half of the REFCNTL register contains the 16-bit RPARLD (refresh pseudo-address reload) field. You can program this value to select the maximum 16-bit value to be output on the external address bus refresh. This value is loaded into an internal counter (RPADEC), which is decremented by 1 each time a refresh cycle is performed. When RPADEC reaches 0, it is reloaded from the RPARLD field. RPARLD and RPADEC are both loaded with 0xFFFF following hardware reset.

13.4 Refresh Cycle Characteristics

The TC signals a refresh cycle by asserting a value of 000010 on the STATUS[5:0] bus at row time. Refresh cycles by the TC are characterized by the following signal activity:

- $\overline{CAS}$/DQM falls prior to $\overline{RAS}$ (for DRAM refresh).
- All $\overline{CAS}$/DQM[7:0] strobes are active (for DRAM refresh).
- $\overline{TRG}$/$\overline{CAS}$, $\overline{W}$, and $\overline{DBEN}$ all remain high (inactive), because no data transfer occurs (for DRAM refresh).
- For SDRAM refreshes, a single REFR command is performed.
- The data bus is driven to high impedance.
- The upper half of the address bus (A[31:16]) contains the refresh pseudo-address, and the lower half (A[15:0]) is driven to all 0s.
- If retry is asserted low during the r3 state, the cycle timing is not modified. Instead, the pseudo-address and backlog counters are simply not decremented.
- Selecting user timing has no effect on refresh cycles.
- Upon completion, the memory interface returns to state r1 to await the next access.

The following sections detail the three DRAM refresh cycles and the SDRAM refresh cycle. Timing diagrams are presented for each cycle type.

13.5 One-Cycle-per-Column Refresh (Pipelined and Nonpipelined)

The state sequence for a one-cycle-per-column refresh is shown in Figure 13–1. Note that pipelined cycles are identical to nonpipelined cycles.

Figure 13–1. One-Cycle-per-Column Refresh States

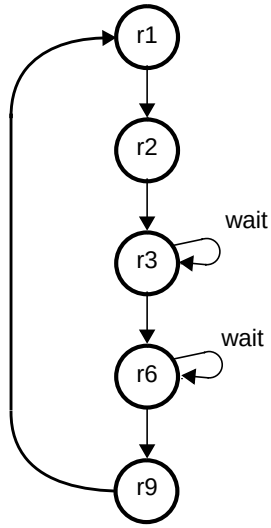
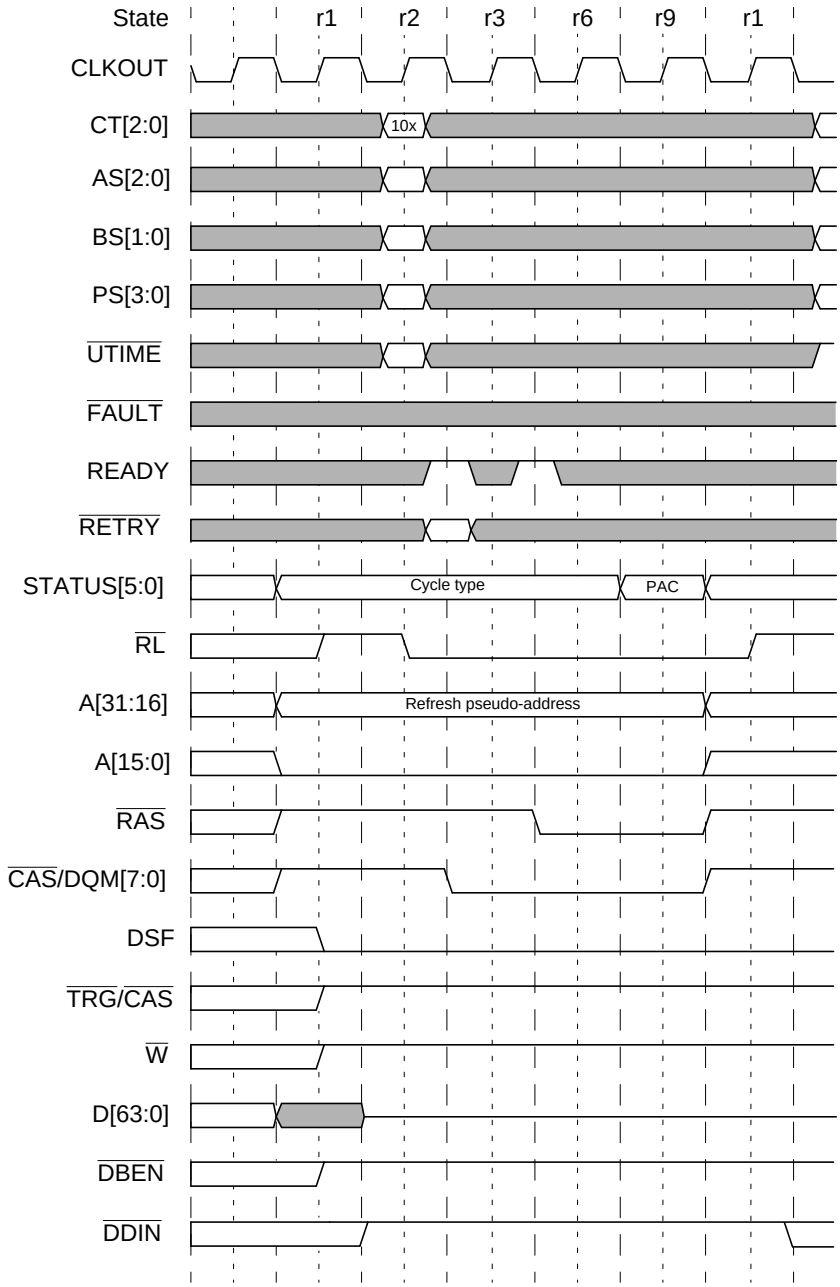


Figure 13–2 shows the general timing for a single refresh cycle to a one-cycle-per-column memory. The cycle consists of the r1, r2, r3, r6, and r9 states. Additional r3 and r6 states can be inserted by adding wait states. The refresh pseudo-address is output in state r1. All $\overline{\text{CAS}}$/DQM strobes are activated at the start of r3, and $\overline{\text{RAS}}$ is driven active at r6. The cycle completes after the next state (r9) when $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM are driven high.

Figure 13–2. One-Cycle-per-Column Refresh Timings



13.6 Two-Cycles-per-Column Refresh

The state sequence for a two-cycles-per-column refresh is shown in Figure 13–3.

Figure 13–3. Two-Cycles-per-Column Refresh States

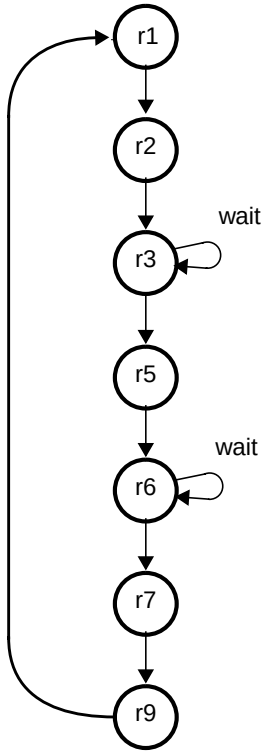
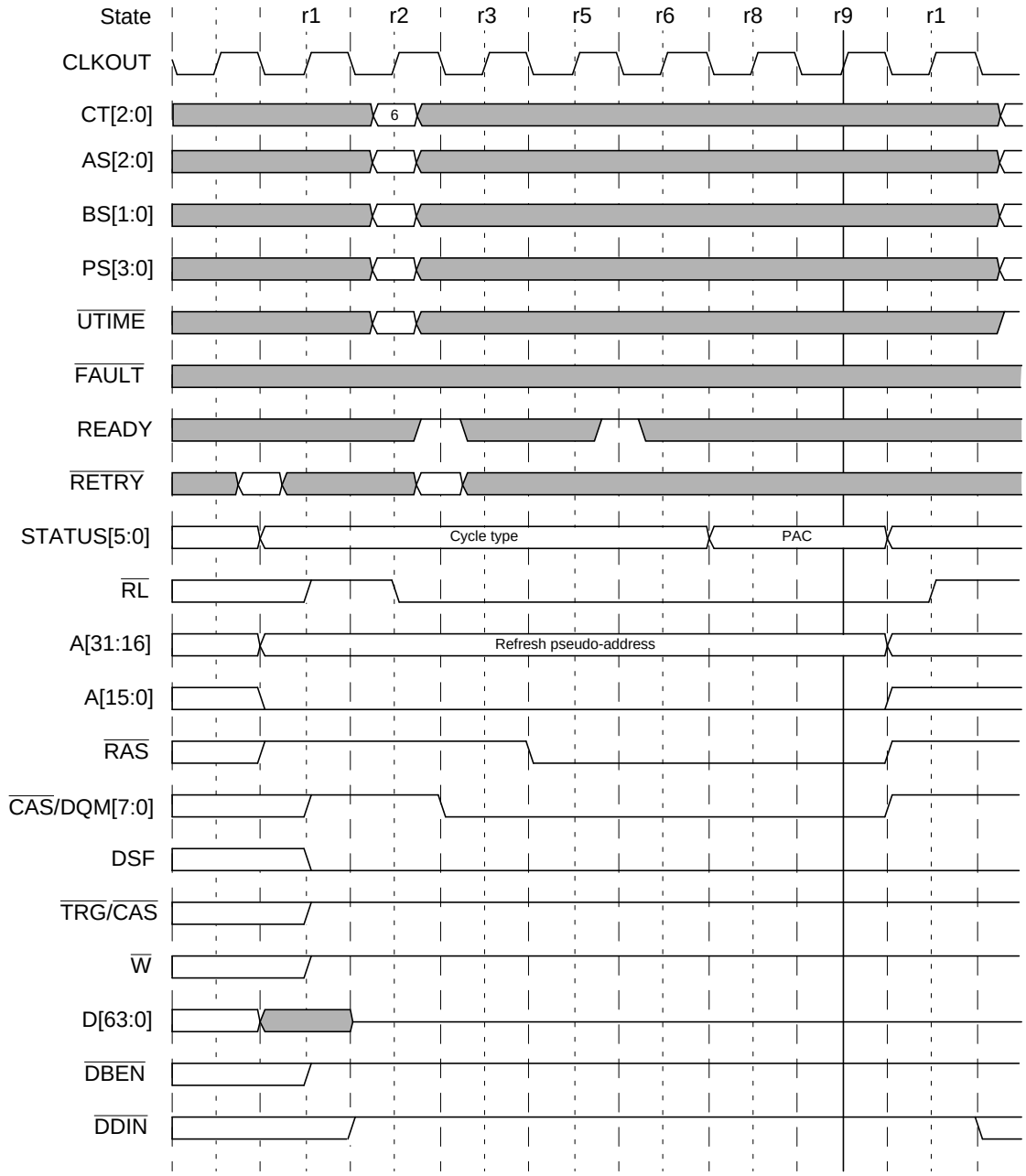


Figure 13–4 shows the general timing for a single refresh cycle to a two-cycles-per-column memory. The cycle consists of the r1, r2, r3, r5, r6, r7, and r9 states. Additional r3 and r6 states can be inserted by adding wait states. The refresh pseudo-address is output in state r1. All $\overline{\text{CAS}}$/DQM strobes are activated at the start of r3, and $\overline{\text{RAS}}$ is driven active at r5. The cycle completes after state r9 when $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM are driven high.

Figure 13–4. Two-Cycles-per-Column Refresh Timings



13.7 Three-Cycles-per-Column Refresh

The state sequence for a three-cycles-per-column refresh is shown in Figure 13–5.

Figure 13–5. Three-Cycles-per-Column Refresh States

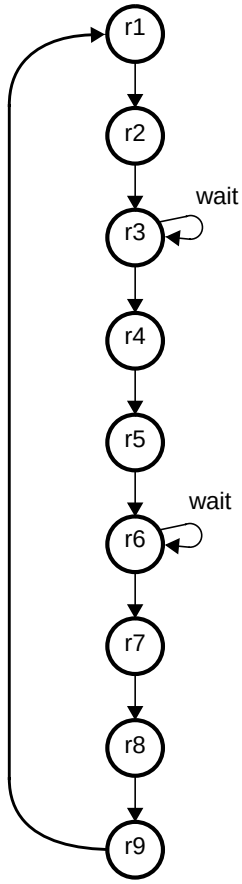
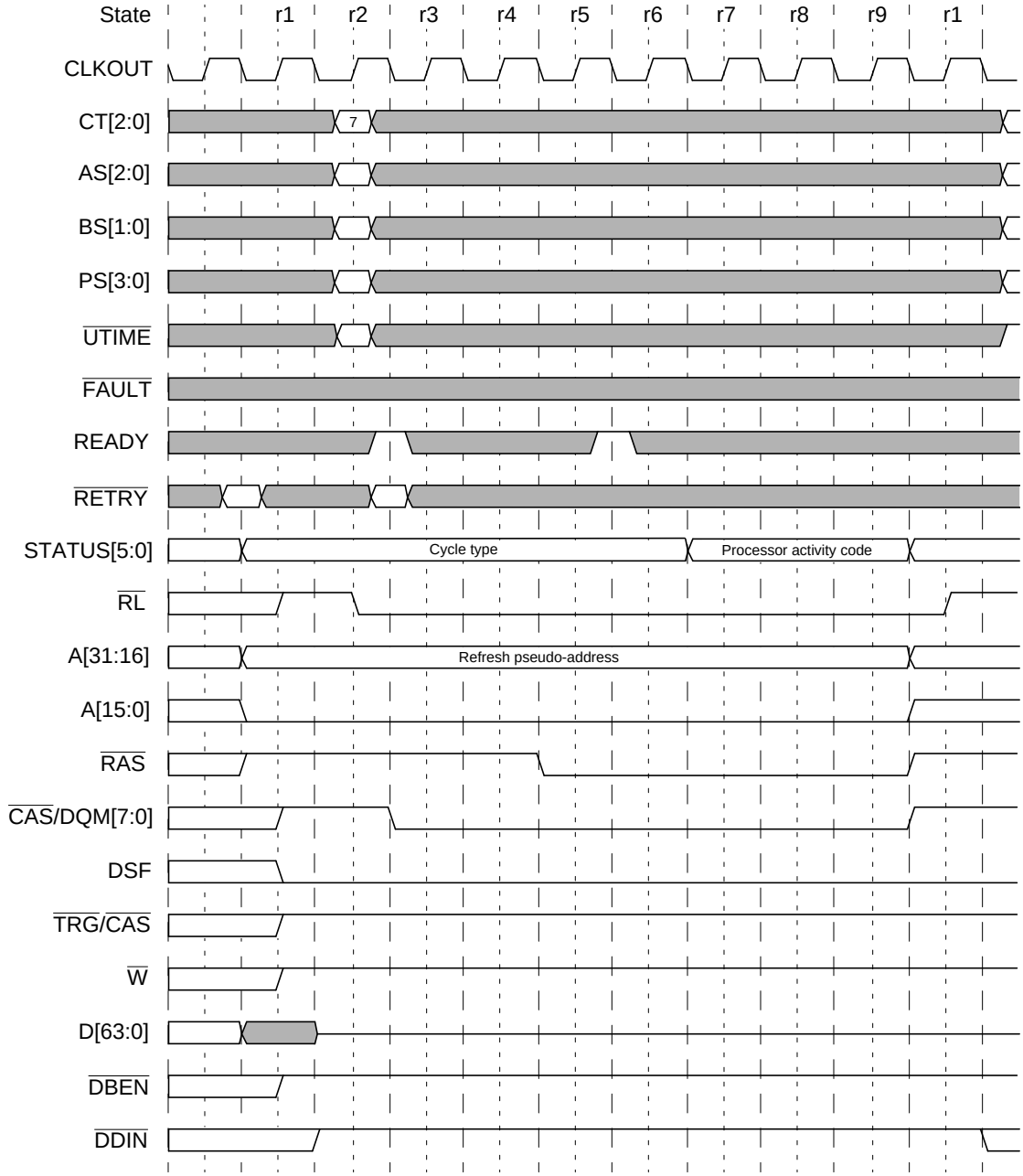


Figure 13–6 shows the general timing for a single refresh cycle to a three-cycles-per-column memory. The cycle consists of the r1, r2, r3, r4, r5, r6, r7, r8, and r9 states. Additional r3 and r6 states can be inserted by adding wait states. The refresh pseudo-address is output in state r1. All $\overline{\text{CAS}}$/DQM strobes are activated at the start of r3, and $\overline{\text{RAS}}$ is driven active at r5. State r4 provides additional $\overline{\text{CAS}}$/DQM before $\overline{\text{RAS}}$ set up time. The cycle is completed after state r9 when $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$/DQM are driven high.

Figure 13–6. Three-Cycles-per-Column Refresh Timings



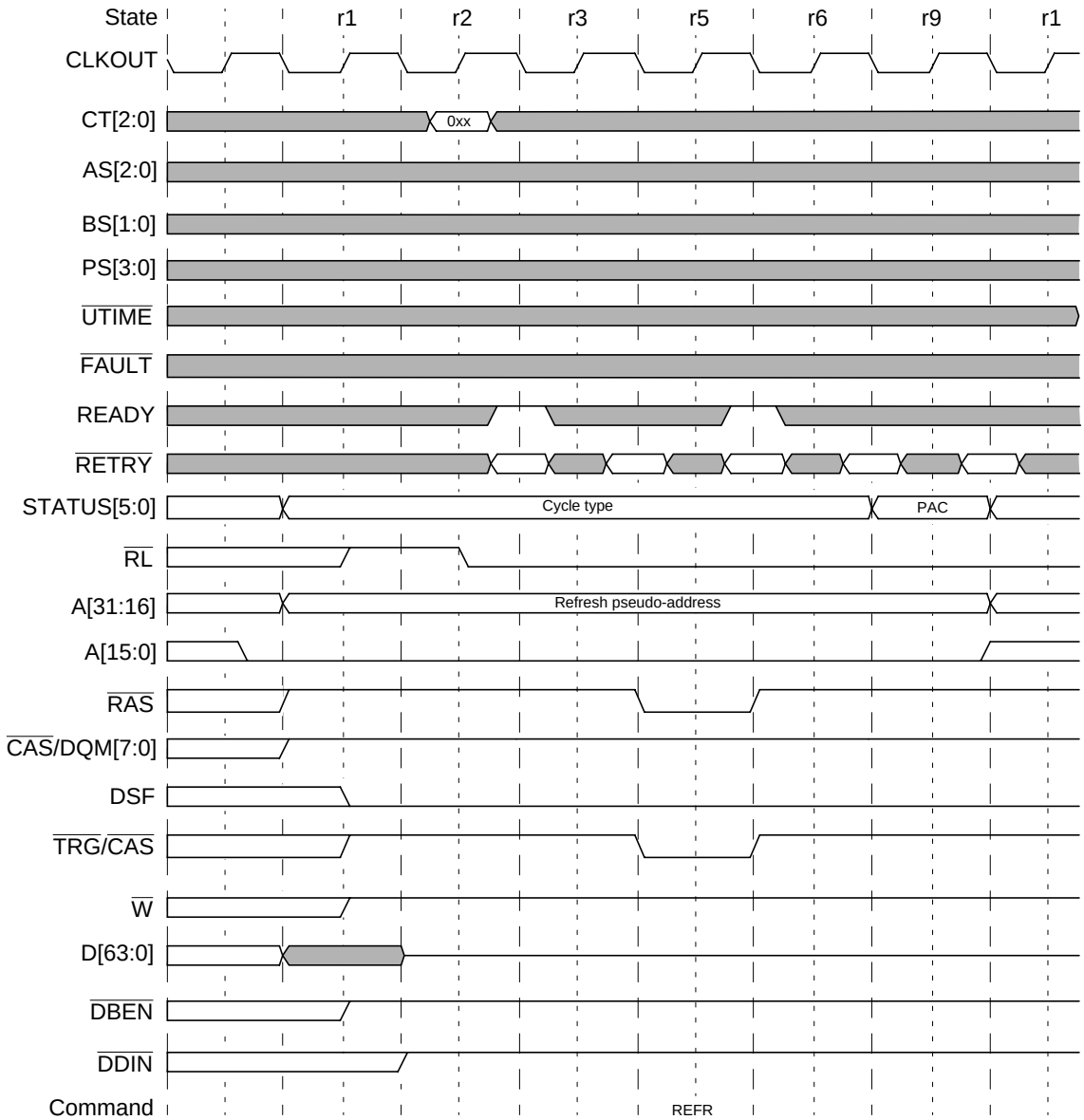
13.8 SDRAM Refresh

The TC converts $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycles to SDRAM refresh cycles by signaling a value of 0xx on the CT[2:0] bus at the beginning of the cycle. The low-order bits of the refresh pseudo-address on A[31:16] can be used to distinguish SDRAM from DRAM refresh cycles for system decode purposes.

As Figure 13–7 shows, the SDRAM refresh command is generated in state r5. Then the TC goes through the normal refresh states before starting a new access.

Since the TC performs a DCAB cycle to deactivate the SDRAM banks at the end of each page access, there is no need for a deactivate cycle prior to the SDRAM refresh.

Figure 13–7. SDRAM Refresh



Chapter 14

Host Interface

This chapter provides a signal summary of the transfer controller's host interface.

This chapter covers the following topics:

Topic **Page**

14.1 Host-Interface Signals

The TMS320C80's host interface is a simple handshake mechanism that allows the TC and an external device to both share the bus. The handshake mechanism uses the following signals:

Signal Name	Direction	Description
$\overline{\text{HREQ}}$	I	**Host-request input.** This signal is driven low when an external device wants to take control of the bus. This is the highest priority request that the TC can receive; it stops driving the bus at the earliest possible moment. TC bus ownership terminates when the current operation completes and the TC pipeline is empty. The external device must continue to drive $\overline{\text{HREQ}}$ active (low) for as long as it desires to control the bus. Whenever $\overline{\text{HREQ}}$ is inactive (high), the TC owns and drives the bus. The $\overline{\text{HREQ}}$ input is internally synchronized to the TMS320C80's internal clock. The $\overline{\text{HREQ}}$ pin also determines whether the MP is active or halted at reset (see section 15.1.1).
$\overline{\text{HACK}}$	O	**Host-acknowledge output.** This signal is driven low by the TC after an active $\overline{\text{HREQ}}$ to indicate that the TC has placed its signals in high impedance and is relinquishing the bus. An external device can then drive the bus as required. $\overline{\text{HACK}}$ is driven inactive (high) asynchronously after $\overline{\text{HREQ}}$ is detected as being inactive, and the TC resumes driving the bus.
REQ[1:0]	O	**Internal- request outputs.** These signals are a two-bit encoding of the highest priority internal request being received by the TC. The request codes and their associated cycles are shown below. External logic can monitor these signals to determine when to relinquish the bus back to the TC.

REQ[1:0]	Associated Internal Request
1 1	SRT, urgent refresh, XPT, and VCPT
1 0	Cache/DEA requests and urgent packet transfers
0 1	High-priority packet transfers
0 0	Low-priority packet transfers, trickle refresh, idle

Because host requests are given highest priority by the TC, you can determine the host operating level by deciding which REQ[1:0] values cause the host to relinquish bus ownership.

When $\overline{\text{HACK}}$ is active (low), all TMS320C80 bus-interface outputs, except CLKOUT and STATUS[5:0], are placed in high impedance. The REQ[1:0], $\overline{\text{HACK}}$, and video controller outputs continue to be driven.

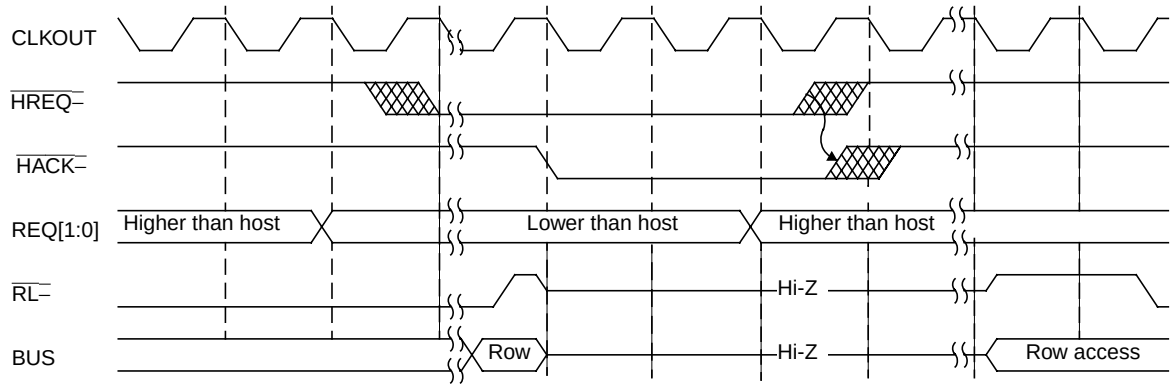
14.2 Host-Interface Timing

Figure 14–1 shows the general timing for the host-interface handshake signals. Note that external logic determines whether REQ[1:0] is a higher or lower priority than the host. It also determines the amount of time between when the higher priority REQ[1:0] value appears and when $\overline{\text{HREQ}}$ is driven inactive.

Note:

There are no internal pullup resistors on any of the pins that the TC drives to high impedance. Either the host interface drives these or external pullup resistors should be used.

Figure 14–1. Host Interface Timing



Chapter 15

Resetting the Transfer Controller

This chapter describes hardware and software mechanisms for resetting the transfer controller.

This chapter covers the following topics:

Topic **Page**

15.1 Hardware Reset

A hardware reset of the TMS320C80 occurs when the $\overline{\text{RESET}}$ input is driven active (low). The TC immediately (and asynchronously) drives its local bus output and control signals to high impedance. These signals are A[31:0], $\overline{\text{CAS}}$/DQM[7:0], D[63:0], $\overline{\text{DBEN}}$, $\overline{\text{DDIN}}$, DSF, $\overline{\text{RAS}}$, $\overline{\text{RL}}$, $\overline{\text{TRG}}$/$\overline{\text{CAS}}$, and $\overline{\text{W}}$. The pins that are *not* driven to high impedance are CLKOUT, $\overline{\text{HACK}}$, REQ[1:0], and STATUS[5:0]. These pins continue to be driven with valid data.

Note:

The TC has no internal pullup resistors on any of the pins driven to high impedance. External pullup resistors must pull these signals high if required by the system.

A hardware reset causes the TC to abort all cache services, DEA services, and packet transfers. The packet transfer timer, refresh logic, round robins, and all other internal logic are reset to their default state.

Once hardware reset is complete and $\overline{\text{RESET}}$ is driven high, the TC begins driving all tristated signals with their inactive values. It then executes at least 32 refresh cycles to provide proper DRAM and SDRAM power-up initialization.

After completing the initial refresh cycles, the TC will service the MP instruction cache fill to fetch the cache block beginning at 0xFFFF FFC0. This block contains the address 0xFFFF FFF8 which is where the MP begins instruction execution.

15.1.1 Reset Halt

During hardware reset, the MP can be configured to exit reset in either a halted or running state. The reset condition of the MP is selected by setting the $\overline{\text{HREQ}}$ pin on the rising edge of $\overline{\text{RESET}}$. If $\overline{\text{HREQ}}$ is low at the $\overline{\text{RESET}}$ rising edge, the MP comes up running and immediately requests a cache fill to fetch its starting instruction. (The TC actually performs the cache fill after the DRAM and SDRAM initialization cycles are completed.) If $\overline{\text{HREQ}}$ is high at the $\overline{\text{RESET}}$ rising edge, the MP comes up halted. It does not request the initial cache fill until it has been unhalted with the first interrupt occurrence on $\overline{\text{EINT3}}$; however, DRAM and SDRAM initialization and refreshes still occur.

The hold time of $\overline{\text{HREQ}}$ after the $\overline{\text{RESET}}$ rising edge is 0. This allows the MP to be reset in running mode by tying $\overline{\text{HREQ}}$ to $\overline{\text{RESET}}$. Systems with a host that wish to reset with the MP halted, must ensure that $\overline{\text{HREQ}}$ is inactive (high) at the end of reset.

15.1.2 Reset Endian Initialization

In addition to determining the state of the MP, the rising edge of $\overline{\text{RESET}}$ also determines in which endian mode the TMS320C80 operates. The $\overline{\text{UTIME}}$ input is sampled on $\overline{\text{RESET}}$'s rising edge, and if low, the TMS320C80 will operate in big-endian mode. If $\overline{\text{UTIME}}$ is sampled high, little-endian mode is selected. After reset, the endian mode of operation can only be changed by another hardware reset.

As with $\overline{\text{HREQ}}$, the sampling of $\overline{\text{UTIME}}$ is such that tying $\overline{\text{UTIME}}$ to $\overline{\text{RESET}}$ causes it to be sampled low. Thus, the TMS320C80 will operate in big-endian mode.

15.2 Resetting the Transfer Controller Under Software Control

The master processor can perform a reset of the transfer controller with a reset command word.

During a software reset, the TC drains its pipeline and then terminates all cache services, DEA services, and packet transfers. The software reset causes the packet transfer timer logic, round robins, and other internal logic to reset to their default state. Refresh logic is not reset, and refresh cycles continue.

In addition to resetting itself, the transfer controller sends reset signals to the MPs, PPs, and VC and causes those subsystems to reset.

Chapter 16

Programming Details

This chapter includes the following programming information:

- How to program the TC registers mapped into on-chip memory space.
- How to program packet transfers, using parameter tables in on-chip memory.

Topics in this chapter include:

Topic **Page**

16.1 Transfer-Controller Registers

The TC has four registers that are mapped into on-chip memory. These registers can be accessed using the MP's *load* (ld) and *store* (st) *instructions*. Because the registers are accessed via the MP's 32-bit on-chip register bus, doubleword loads and stores are not permitted. The registers cannot be accessed using the PPs.

The TC registers are:

- ❏ REFCNTL — Controls system DRAM refresh cycles
- ❏ PTMIN — Determines minimum packet transfer cycle time
- ❏ PTMAX — Determines maximum packet transfer cycle time
- ❏ FLTSTS — Indicates if a fault occurs during a TC operation

The TC register map is summarized in Table 16–1, and each register is described in the sections that follow.

Table 16–1. Transfer Controller Register Map

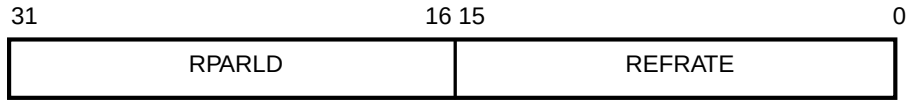
Address	TC Register
0x01820000	REFCNTL
0x01820004	PTMIN
0x01820008	PTMAX
0x0182000C	FLTSTS

Note:
Reserved bits in TC registers should be written as zeros to maintain compatibility with future devices.

16.1.1 The REFCNTL Register

Address: 0x01820000

Format:



Description: The refresh control register (REFCNTL) contains two 16-bit values that control system DRAM refresh cycles:

REFRATE **Refresh interval, bits 15–0**

The 16-bit REFRATE field determines the interval at which DRAM refresh cycle requests are generated. The value in REFRATE represents the number of TMS320C80 clock cycles that occur between each refresh request. Values of less than 32 (0x0020) in REFRATE cause DRAM refreshes to be disabled. REFRATE is set to 32 (0x0020) at reset.

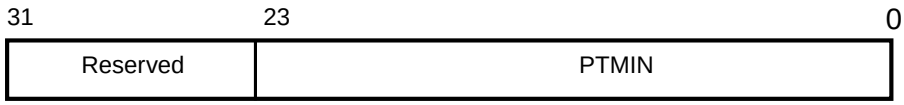
RPARLD **Refresh pseudo-address reload, bits 31–16**

During DRAM refreshes, a 16-bit pseudo-address is output on A[31:16] of the external address bus to be used with refresh bank decoding or $\overline{\text{RAS}}$-only refresh. RPARLD represents the maximum value that is output during refresh cycles. A refresh address counter keeps track of the current refresh address. Each time that a refresh cycle occurs, the counter is decremented. When the counter reaches 0, it is reloaded with the value in RPARLD. RPARLD is set to 0xFFFF at reset.

16.1.2 The PTMIN Register

Address: 0x01820004

Format:

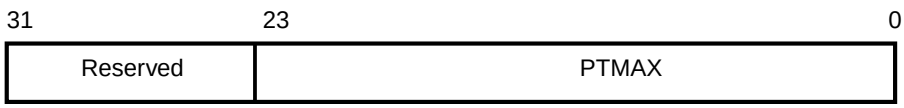


Description: The 24-bit packet transfer minimum length register (PTMIN) indicates the minimum number of clock cycles a transfer must be serviced by the TC before it can be interrupted by a higher priority transfer. PTMIN is loaded with the value 0x00010000 (64K cycles) at reset. For more information about how to program PTMIN, see section 3.9.1, *Setting the Minimum Packet Transfer Length*.

16.1.3 The PTMAX Register

Address: 0x01820008

Format:

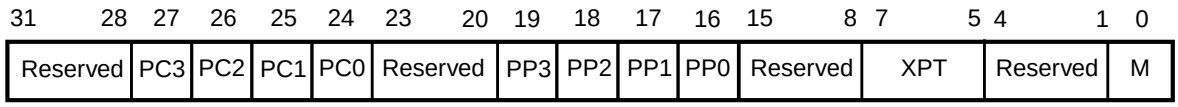


Description: The 24-bit packet transfer maximum length register (PTMAX) determines the maximum amount of time that a transfer can continue after executing for the time specified by PTMIN before it is timed out. Once a PT is serviced by the TC for PTMIN + PTMAX cycles, the transfer can be suspended for execution of another transfer of the same priority. (PTMAX does not affect when a transfer is suspended for a higher priority transfer.) At reset, the value 0x00010000 (64K cycles) is loaded into PTMAX. For more information on how to program PTMAX, see section 3.9.2, *Setting the Maximum Packet-Transfer Length*.

16.1.4 The FLTSTS Register

Address: 0x0182000C

Format:



Fields:

Bits	Name	Function
0	M	MP packet-transfer fault
7–5	XPT	XPT fault or error
16	PP0	PP0 packet-transfer fault
17	PP1	PP1 packet-transfer fault
18	PP2	PP2 packet-transfer fault
19	PP3	PP3 packet-transfer fault
24	PC0	PP0 cache/DEA fault
25	PC1	PP1 cache/DEA fault
26	PC2	PP2 cache/DEA fault
27	PC3	PP3 cache/DEA fault

Description: The fault status (FLTSTS) register contains status bits that indicate that a fault has occurred during a packet transfer, PP instruction cache fill, or DEA cycle.

M **MP packet-transfer fault, bit 0**

The M bit is set to 1 when a fault occurs during a packet transfer requested by the MP.

XPT **Externally initiated packet-transfer fault, bits 7–5**

The XPT bits are set to correspond to an externally- or VC-initiated packet transfer that encounters a fault or error. A faulted XPT is immediately terminated and the XPT number is copied into the XPT field. The TC ignores all further inputs to the $\overline{\text{XPT[2:0]}}$ pins and VCPT requests until the XPT bits in FLTSTS are cleared.

7	6	5	Faulted Request	
1	1	1	XPT 7/SOF0	faulted
1	1	0	XPT 6/SAM0	faulted
1	0	1	XPT 5/SOF1	faulted
1	0	0	XPT 4/SAM1	faulted
0	1	1	XPT 3	faulted
0	1	0	XPT 2	faulted
0	0	1	XPT 1	faulted
0	0	0	No XPT fault	

PP **PP packet-transfer fault, bits 19–16**

A PP bit is set to 1 when a fault occurs during a packet transfer requested by a PP.

- ❑ PP0 (bit 16) indicates a PP0 packet transfer fault.
- ❑ PP1 (bit 17) indicates a PP1 packet transfer fault.
- ❑ PP2 (bit 18) indicates a PP2 packet transfer fault.
- ❑ PP3 (bit 19) indicates a PP3 packet transfer fault.

PC **PP instruction-cache fill/DEA fault, bits 27–24**

A PC bit is set to 1 when a fault occurs during a PP cache fill or DEA operation.

- ❑ PC0 (bit 24) indicates a PP0 cache-fill/DEA fault.
- ❑ PC1 (bit 25) indicates a PP1 cache-fill/DEA fault.
- ❑ PC2 (bit 26) indicates a PP2 cache-fill DEA fault.
- ❑ PC3 (bit 27) indicates a PP3 cache-fill/DEA fault.

The setting of any one of the M, XPT, PP0–3, or PC0–3 bits causes the mf bit to be set in the MP's INTPEN register. Clear FLTSTS bits before clearing mf in INTPEN to ensure correct operation. Clearing a FLTSTS bit that is set causes the associated packet transfer or cache fill to be rescheduled. In the event of an XPT fault, clearing the FLTSTS bit(s) restarts the TC's sampling of the $\overline{\text{XPT}[2:0]}$ pins. You can clear FLTSTS bits by writing a 1 to the appropriate bit. Writing 0 to a bit has no effect.

16.2 Normal Packet-Transfer Parameters

Because of the variety of transfer formats, there are a large number of src and dst combinations. This section describes normal (long-form) parameter tables only. For more information about short-form parameter tables, see section 16.3, *Short-Form Packet-Transfer Parameters* and Appendix A, *Examples of Packet-Transfer Parameter Tables*.

Packet-transfer parameters tables must adhere to the following rules:

- ❏ They must be located in MP or PP parameter RAMs or in PP data RAMs.
- ❏ They must be 64-byte aligned.

Before submitting the request, the requesting processor places the starting address of the parameter table in the linked list start address location in its own parameter RAM.

Figure 16–1 through Figure 16–6 show the packet-transfer parameter fields for various types of packet transfers. The address of each field relative to the start of the packet transfer parameters (represented by PT) is shown, as well as the valid transfer types for each field. Some fields are not used and can be left unprogrammed.

Figure 16–1. Dimensioned src With Transparency to Dimensioned dst Packet Transfer Parameters

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst C Count		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src C Pitch		PT + 40
src Transparency Value(s)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst C Pitch	
PT + 48	src Transparency Value(s)			
PT + 56	Don't Care			

Figure 16–2. Block Write Dimensioned src to Dimensioned dst Packet Transfer Parameters

a) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst C Count		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src C Pitch		PT + 40
Color Register Value(s)				PT + 48
Don’t Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst C Pitch	
PT + 48	Color Register Value(s)			
PT + 56	Don’t Care			

Figure 16–3. Fill-With-Value src to Dimensioned dst Packet Transfer

a) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
dst Start Address		Don’t Care		PT + 8
dst B Count	dst A Count	Don’t Care		PT + 16
dst C Count		Don’t Care		PT + 24
dst B Pitch		LS Fill Value Word		PT + 32
dst C Pitch		MS Fill Value Word		PT + 40
Don’t Care				PT + 48
Don’t Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT + 8	Don’t Care		dst Start Address	
PT + 16	Don’t Care		dst B Count	dst A Count
PT + 24	Don’t Care		dst C Count	
PT + 32	LS Fill Value Word		dst B Pitch	
PT + 40	MS Fill Value Word		dst C Pitch	
PT + 48	Don’t Care			
PT + 56	Don’t Care			

Figure 16–4. Dimensioned src to Fixed-Patch Guided dst Packet Transfer Parameters

a) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
dst Start/Base Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst Number of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start/Base Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst Number of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Figure 16–5. Dimensioned src to Variable-Patch Guided dst Packet Transfer Parameters

a) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
dst Start/Base Address		src Start Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst Number of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start/Base Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src C Count		dst Number of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Packet transfer parameter tables must be located in on-chip memory (MP or PP parameter RAMs or PP data RAMs) and must be 64-byte aligned (six LSBs of the address are 0). The table can reside in any processor's RAM; thus, the MP can use a parameter table located in PP's parameter RAM, or vice versa. The requesting processor simply places the appropriate starting address in the linked list start address location in its own parameter RAM before submitting its transfer request.

Figure 16−1 through Figure 16−6 show that the order of the packet transfer parameters within the 64-bit doublewords is different for the two endian modes. This allows software to set up the packet transfer parameters independent of the endian mode by using 32-bit writes to the proper word address, as illustrated in Figure 16−6. When the TC fetches a 64-bit word from the parameter table it adjusts the word order to match its internal operation.

Figure 16−6. General Packet Transfer Parameter Format (Endian-Independent)

Word Number	31 … 0	Byte Address
0	Next entry address	PT
1	PT options	PT + 4
0	src start address	PT + 8
1	dst start address	PT + 12
0	src B count | src A count	PT + 16
1	dst B count | dst A count	PT + 20
0	src C count	PT + 24
1	dst C count	PT + 28
0	src B pitch	PT + 32
1	dst B pitch	PT + 36
0	src C pitch	PT + 40
1	dst C pitch	PT + 44
	Transparency/color word 0	†
	Transparency/color word 1	†
0	Don't care	PT + 56
1	Don't care	PT + 60

† The TC always uses the 64-bit value in these two locations *as is*. In other words, it will use:

63 … 32	31 … 0	
PT + 52	PT + 48	Little Endian
PT + 48	PT + 52	Big Endian

Note:
The 64-bit transparency or color-register values are not reordered by the TC. The 64-bit value addressed at location PT+48 represents the actual 64-bit value used by the TC, regardless of the endian mode.

The following sections describe the packet-transfer parameter fields. In many cases there are two identical fields, one for src transfers and one for dst transfers. In these cases, the field description is given as a single context. Depending on the transfer type, some fields are not used and can be left unprogrammed.

16.2.1 Next Entry Address

Address: PT
Transfers: All

The 32-bit next entry address field points to the start of the next entry on the packet transfer request's linked list. This must point to a 64-byte aligned on-chip address (the six LSBs must be 0). The last entry of a linked list needs no special next address entry, because the stop bit in the options field terminates the linked list.

Whenever a packet transfer completes successfully, the value in the next entry address is written to the linked-list start-address location in the requesting processor's parameter RAM, to advance the pointer to the next transfer. The pointer is also updated after the final packet in the linked list has been completed. If the stop bit is being used to pause linked list execution, the pointer is pointing to the next transfer on the linked list when the list is re-enabled.

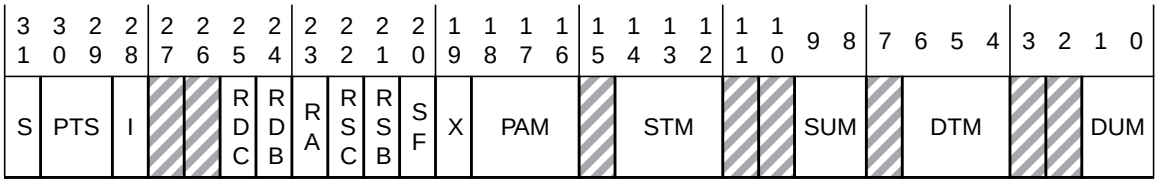
16.2.2 PT Options

Address: PT + 4
Transfers: All

The PT options field selects the form of transfer to be used for the src and dst transfers and determines if the packet will end the linked list. It also enables the selection of additional features, such as special transfer modes, performing address calculations at the completion of a packet transfer, changing the direction of dimensional address calculations, and reversing the src and dst transfers. The format of the options field is shown in Figure 16–7, and each field is discussed in detail in the following sections. Loading the

options field with all 0's results in a default packet transfer that uses dimensioned transfers on src and dst with no special addressing modes.

Figure 16–7. PT Options Field Contents



- ❏ DUM—dst update mode
- ❏ DTM—dst transfer mode
- ❏ SUM—src update mode
- ❏ STM—src transfer mode
- ❏ PAM—PT access mode
- ❏ X—exchange src/dst
- ❏ SF—short-form packet transfer
- ❏ RSB—reverse source B addressing
- ❏ RSC—reverse source C addressing
- ❏ RA—reverse A addressing
- ❏ RDB—reverse destination B addressing
- ❏ RDC—reverse destination C addressing
- ❏ I—interrupt when finished
- ❏ PTS—packet transfer status
- ❏ S—stop

dst Update Mode (DUM) — Bits 0–1

These two bits indicate how the dst start address in the original packet transfer parameter table is updated once the transfer has completed. If these bits are nonzero, an extra address calculation is performed. This value is then written over the original address specified in the packet-transfer parameter table. This allows the transfer to be resubmitted, possibly continuing from where it left off. This is especially useful when ping-ponging between two memory areas.

The dst update modes are shown in Table 16–2. Note that if the appropriate dst reverse addressing bit (RDC or RDB) is set, a subtraction is performed rather than an addition.

Table 16–2. dst Update Mode (DUM)

Bit 1	Bit 0	dst Update Operation
0	0	No update
0	1	Add (subtract) the B pitch to (from) the start address of the last line in the dst transfer, and write the result to the dst start address in the packet transfer parameter table.
1	0	Add (subtract) the C pitch to (from) the start address of the last patch in the dst transfer, and write the result to the dst start address in the packet transfer parameter table.
1	1	Add (subtract) the C pitch to (from) the start address of the last patch in the dst transfer, and write the result to the dst start address in the packet-transfer parameter table; then toggle the reverse dst C addressing (RDC) bit in the PT options field.

In normal practice, you would perform one extra step on the largest dimension in use; however, other useful operations can also be performed. Mode 10, for example, can be used to add a C pitch, even though the transfer may have been only two-dimensional. This allows the next two-dimensional packet to be positioned relative to the last.

Mode 11 is especially useful for resubmitting packet transfers that ping-pong between two one- or two-dimensional patches, such as on-chip RAM spaces. Because the direction of addressing the third dimension is reversed each time that the packet-transfer parameters are updated upon completion, it alternates between the memory areas.

Note:
This functionality is intended primarily for use with dimensioned transfers. It can be specified with guided transfers, but use caution. The natural operation is always performed, so remember that for guided transfers, the dst C pitch is replaced by the dst guide table pointer.

dst Transfer Mode (DTM) — Bits 4–6

These three bits indicate the form of transfer used for dst addressing. The bit codes are shown in Table 16–3. src-specific modes (look-up table and fill-with-value) are not defined (reserved) for dst addressing. Specifying one of the reserved values results in a packet-transfer error.

Table 16–3. dst Transfer Mode (DTM)

Bit			
6	5	4	dst Transfer Mode
0	0	0	Dimensioned
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Variable-patch delta-guided
1	0	1	Variable-patch offset-guided
1	1	0	Fixed-patch delta-guided
1	1	1	Fixed-patch offset-guided

src Update Mode (SUM) — Bits 8–9

These two bits indicate the value that the src start address in the original packet transfer parameters should be updated with when the transfer completes. If these bits are nonzero, an extra src address calculation is performed after the transfer completes. This value is then written over the original src start address specified in the packet transfer parameters. This allows the transfer to be resubmitted, possibly continuing from where it left off. This is especially useful when ping-ponging between two memory areas.

The src update modes are shown in Table 16–4. If the appropriate src reverse addressing bit (RSC or RSB) is set, a subtraction, rather than an addition, is performed.

Table 16–4. src Update Mode (SUM)

Bit		
9	8	Source Update Operation
0	0	No update
0	1	Add (subtract) the B pitch to (from) the start address of the last line in the src transfer and write the result to the src start address in the packet-transfer parameter table.
1	0	Add (subtract) the C pitch to (from) the start address of the last patch in the src transfer and write the result to the src start address in the packet transfer parameter table.
1	1	Add (subtract) the C pitch to (from) the start address of the last patch in the src transfer and write the result to the src start address in the packet-transfer parameter table; then toggle the reverse src C addressing (RSC) bit in the PT options field.

In normal practice, you would perform one extra step of the largest dimension in use; however, other practical operations can also be performed. Mode 10, for example, can be used to add a C pitch, even though the transfer may have been only two-dimensional. This allows the next two-dimensional packet to be positioned relative to the last.

Mode 11 is especially useful for resubmitting packet transfers that ping-pong between two one- or two-dimensional patches, such as on-chip RAM spaces. Because the direction of addressing the third dimension is reversed each time that the packet-transfer parameters are updated upon completion, it alternates between the memory areas.

Note:
This functionality is intended primarily for use with dimensioned transfers. It can be specified with guided transfers, but caution should be used. The natural operation is always performed, so remember that the src C pitch is replaced by the src guide table pointer for guided transfers.

src Transfer Mode (STM) — Bits 12–14

These three bits indicate the form of transfer used during src addressing. The bit codes are shown in Table 16–5. Specifying the reserved value results in a packet-transfer fault.

Table 16–5. src Transfer Mode (STM)

Bit			
14	**13**	**12**	**src Transfer Mode**
0	0	0	Dimensioned
0	0	1	Fill with value
0	1	0	Reserved
0	1	1	Fixed-patch offset-guided LUT
1	0	0	Variable-patch delta-guided
1	0	1	Variable-patch offset-guided
1	1	0	Fixed-patch delta-guided
1	1	1	Fixed-patch offset-guided

PT Access Mode (PAM) — Bits 16–18

These three bits encode special access modes as shown in Table 16–6. These modes modify the way that the src data is written to the dst. For on-chip memory destinations, only normal transfers (mode 000) are allowed. Each special mode is described in detail in Appendix A, *Special Packet-Transfer Access Modes*.

Table 16–6. PT Access Modes (PAM)

Bit			
18	**17**	**16**	**Access Mode**
0	0	0	Normal
0	0	1	Peripheral device transfer
0	1	0	Block write
0	1	1	Serial-register transfer
1	0	0	8-bit src transparency
1	0	1	16-bit src transparency
1	1	0	32-bit src transparency
1	1	1	64-bit src transparency

- **Mode 000** is encoded to specify normal access. No special addressing modes are used on the src or dst. Data is transferred from src to dst without alteration.
- **Mode 001** is the peripheral-device transfer mode. This mode allows another device to read or write to TMS320C80 external memory using the TC as a memory controller. Device reads of memory are accomplished by programming the src. Device writes to memory are accomplished by programming the dst. In either case, the TC drives memory address and control lines normally, (with address generated according to the transfer parameters) but places the data bus in high impedance so that the peripheral device can read or drive data. The peripheral-device transfer mode can be used with any form of src or dst transfer (except fill-with-value). For more details see section 11.1, *Peripheral-Device Packet Transfers*.
- **Mode 010** allows the packet transfer to make use of VRAM/SGRAM block writes. It causes the TC to load the VRAM/SGRAM color register and then perform dst writes to external memory using the VRAM/SGRAM block-write mode. In this mode, the src data represents block-write address mask bits that specify which locations in the VRAM/SGRAM will be written to with the VRAM/SGRAM color register data. These bits are fetched from the src using normal addressing and subsequently written to the dst VRAM/SGRAM using block-write mode. The value loaded in the VRAM/SGRAM color registers is specified as one of the packet transfer parameters (color register value). Block write is described in detail in section 11.4, *Block-Write Examples*.

Note:

The block-write operation is supported for off-chip destinations only. Attempting a block write to an on-chip destination address causes the transfer to suspend with an error condition (see section 3.11, *Packet-Transfer Errors*).

- **Mode 011** is a serial-register transfer mode used to do bulk initialization of VRAMs. The src address is used to copy a VRAM row into the VRAM serial shift register. dst addresses are then used to copy the shift register into a number of VRAM memory rows. The normal operation entails loading the src with an A count of 1 and the dst with an A count of 1 and a B Count of n-1, with *n* being the number of rows to be written. Note that no transfer of data over the data bus (or internal crossbar) takes place. All src and dst accesses occur in nonpage mode. A detailed explanation of this mode is available in section 11.5, *Serial Register Packet Transfers*.

Note:

Serial register transfer operations are supported for off-chip src and dst only. Attempting a serial register transfer with an on-chip src or dst address causes the transfer to suspend with an error condition (see section 3.11, *Packet-Transfer Errors*).

- **Modes 1XX** enable transparency. src and dst dimensioned or guided transfers are performed normally. Before the dst data is written however, it is compared to the transparency value(s) given in the packet transfer parameters. The two LSBs of PAM indicate the size of the transparency data. Thus, one 64-bit, two 32-bit, four 16-bit, or eight 8-bit comparisons are made. If any of the comparisons are true, the TC disables the corresponding byte strobe(s) so that the dst byte(s) are not written. For a complete description of transparency, see section 11.6, *Transparency-Packet Transfers*.

Note:

Transparency operation is supported for off-chip destinations only. Attempting transparency with an on-chip destination causes the transfer to suspend with an error condition (see section 3.11, *Packet-Transfer Errors*).

Exchange src and dst Parameters (X) — Bit 19

Setting this bit reverses the direction of a packet-transfer without manually swapping the src and dst parameters. This is useful when returning data to the location that it originally came from.

When the X bit is set, the TC swaps all the src and dst values (start addresses, pitches, counts, and guide table pointers) when it loads the packet-transfer parameters. Table 16–7 shows all of the 32-bit word swaps, assuming that PT represents the address of the packet-transfer parameter table's next entry address.

Table 16–7. Exchange src and dst Parameter Word Swap

Source	Byte Address	Byte Address	Destination
src start address	PT + 08	PT + 12	dst start address
src A/B counts	PT + 16	PT + 20	dst A/B counts
src C count	PT + 24	PT + 28	dst C count
src B pitch	PT + 32	PT + 36	dst B pitch
src C pitch	PT + 40	PT + 44	dst C pitch

PT (next entry address field) and PT + 04 (PT options field) remain in their usual locations since these values are not src/dst related. Also note that PT + 48 and PT + 52 are not swapped. This allows the 64-bit transparency/color-register parameter to retain its value.

In addition to swapping the src- and dst-related parameter words, the TC also swaps the src- and dst-related bits within the PT options word. This is shown in Table 16–8.

Note:
If a swap results in an unsupported function, then the packet transfer suspends with an error condition.

The exchange of src and dst parameters is performed whenever the packet-transfer parameters are loaded. If the transfer is suspended, then the current parameters are swapped back to their original positions before being saved to the processor's parameter RAM. If the suspended transfer is then restored, the parameters are again swapped as they are loaded by the TC.

Table 16–8. Exchange src and dst PT Option Bit Swap

Destination	Bit Number		Bit Number	Source
dst update mode	0	´	8	src update mode
	1	´	9	
dst transfer mode	4	´	12	src transfer mode
	5	´	13	
	6	´	14	
dst reverse B addressing	24	´	21	src reverse B addressing
dst reverse C addressing	25	´	22	src reverse C addressing

Figure 16–8. Exchanging src and dst Parameters (Little Endian)

a) Packet Transfer In Memory

63 32	31 0	Byte Address
802C0260	01010340	PT
02800000	02000F00	PT + 8
0002 0014	0010 000A	PT + 16
00000004	00000001	PT + 24
00000200	00000100	PT + 32
01010400	00000000	PT + 40
8383838383838383		PT + 48
0000000000000000		PT + 56

b) PT Parameters as Used by TC (After Exchange)

63 32	31 0	Byte Address
810C6002	01010340	PT
02000F00	02800000	PT + 8
0010 000A	0002 0014	PT + 16
00000001	00000004	PT + 24
00000100	00000200	PT + 32
00000000	01010400	PT + 40
8383838383838383		PT + 48
0000000000000000		PT + 56

Figure 16–8 shows the effects of using the exchange bit. Part (a) of the example shows a packet transfer that performs a transparent dimensioned transfer with reversed B addressing from the source to a fixed-patch, delta-guided destination. Because the X bit is set, the src and dst parameters are exchanged as they are loaded into the TC. Part (b) of the example shows how the packet transfer parameters actually appear to the TC after it has reversed the src and dst.

If one of the update modes is specified (see dst Update Mode and src Update Mode), then the operation occurs as normal when the packet transfer is completed. If, for example, a src update operation is selected, the src start address in the original packet transfer parameters is updated, even though it was actually used as the dst start address during the transfer. Likewise, specifying the toggle reverse src C addressing bit as the update mode causes bit 22 of the original packet transfer options to be toggled. This causes the dst C addressing to be reversed if the packet transfer is resubmitted.

- The X bit should be used only in conjunction with transfer modes supported by both the src and dst. Setting the X bit will cause the packet transfer to terminate with an error, if used with any of the following:
  - Fill-with-value transfer
  - Fixed-patch, offset-guided LUT transfer
  - Block write with an on-chip src
  - Transparency with an on-chip src
  - Packet transfer with an unequal number of src and dst bytes

Short Form (SF) - Bit 20

The SF bit selects between normal or short-form operation. When set to 1, short-form mode is enabled. The parameter table is interpreted as four 32-bit words in length and containing short-form PT fields. When set to a 0, normal operation is enabled. The parameter table is interpreted as sixteen 32-bit words in length and containing normal PT fields. For details about programming short-form parameter tables, see section 16.3, *Short-Form Packet-Transfer Parameters*.

Reverse src B Addressing (RSB) — Bit 21

Setting this bit to 1 causes the second dimension of the src to be addressed backwards. The B pitch is subtracted from, rather than added to, the previous line start address.

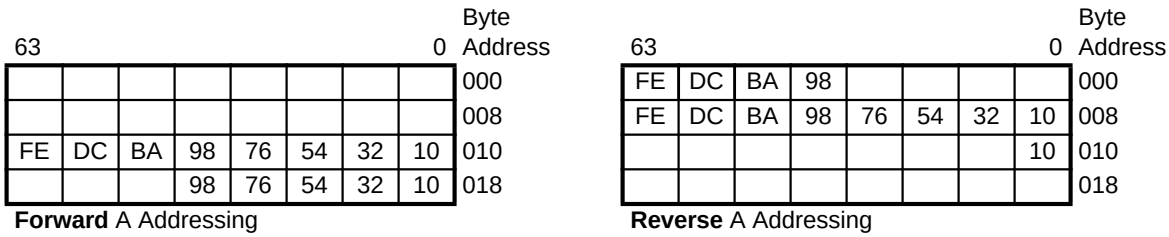
Reverse src C Addressing (RSC) — Bit 22

Setting this bit to 1 causes the third dimension of the src to be addressed backwards. The C pitch is subtracted from, rather than added to, the previous patch start address. For guided transfers, setting the RSC bit causes the guide table pointer to be decremented instead of incremented.

Reverse A Addressing (RA) — Bit 23

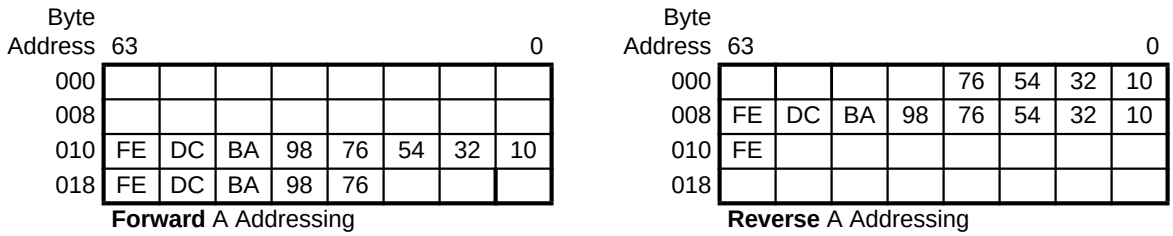
Setting this bit to 1 causes the first dimensions of the src and dst to be addressed backwards. Byte addressing is reversed. This means that the value to which the B pitch is added (or subtracted) is the highest address in the first dimension. Examples of reverse addressing in little-endian and big-endian modes are shown in Figure 16−9 and Figure 16−10, respectively.

Figure 16–9. Reverse A Addressing (Little Endian)



Fill-With-Value src to Dimensioned dst Transfer
Fill Value = 0xFEDC BA98 7654 3210
dst Start Address = 0xXXXX X010
dst A Count = 0x000D

Figure 16–10.Reverse A Addressing (Big Endian)



Fill-With-Value src to Dimensioned dst Transfer
Fill Value = 0xFEDC BA98 7654 3210
dst Start Address = 0xXXXX X010
dst A Count = 0x000D

Reverse dst B Addressing (RDB) — Bit 24

Setting this bit to 1 causes the second dimension of the dst to be addressed backwards. The B pitch is subtracted from, rather than added to, the previous line start address.

Reverse dst C Addressing (RDC) — Bit 25

Setting this bit to 1 causes the third dimension of the dst to be addressed backwards. The C pitch is subtracted from, rather than added to, the previous patch start address. For guided transfers, setting the RDC bit causes the guide table pointer to be decremented instead of incremented.

Interrupt When Finished (I) — Bit 28

Setting this bit to 1 causes an end-of-packet interrupt (the pc bit in the MP or the PTEND bit in the PP) to be sent to the processor that initiated the packet transfer as soon as this entry on the linked list has finished. The linked list can contain further entries. This flags the requesting processor when a particular point in the linked list has been reached.

Setting this bit to a 0 prevents an interrupt to the processor until either an entry with this bit set is encountered and completed, or the last packet transfer on the linked list has completed. If an error occurs at any time, however, the TC immediately sends an *error* interrupt (the bp interrupt bit in the MP's INTPEN register, or the PTERR interrupt bit in the PP's INTFLG register) to the requesting processor.

PT Status (PTS) — Bits 29–30

These two bits reflect the state of a packet transfer request. They should always be set to 0 when a processor submits a request. If a packet transfer within the linked list is suspended, the TC sets the appropriate PTS bits in the PT options field that it saves to the suspended packet parameters area. This is because a suspended transfer contains more parameters than a new transfer, and is saved and restored differently. If either of these bits is 1, when the TC loads packet transfer parameters it recognizes that the transfer is a suspended transfer and restores all the extra parameters. Table 16–9 shows the encoding of these bits.

Table 16–9. PT Status (PTS) Bits Coding

Bit		
30	**29**	**Packet Request Status**
0	0	Not suspended
0	1	Suspended, but not faulted
1	0	Suspended, and faulted on source
1	1	Suspended, and faulted on destination

The TC writes a value of 01 to these bits if the packet transfer is suspended because any of the following conditions occurs:

- ❏ The TC receives a higher priority transfer request.
- ❏ The transfer has timed out.
- ❏ The processor that initiated the packet transfer sends a request that it be suspended.
- ❏ An error condition occurs.

Bit 30 indicates that the packet transfer was suspended because a fault occurred. Bit 29 indicates whether the fault was on the src or the dst. The MP should use this information to resolve the fault condition. It is irrelevant to the TC when it reloads a faulted transfer's parameters because the suspension and restoration process is the same for all types of suspended transfers.

Note:
If the X bit is set in the suspended packet transfer options field, then the value of bit 29 for a faulted transfer is reversed. (In other words, 10 indicates a fault on dst, and 11 indicates a fault on src.) MP software should therefore examine both bits 29 and 19 when determining the faulted address location.

If a fault does occur during a packet transfer, bits 29 and 30 and the appropriate bit in the FLTSTS register are set (see section 16.1.4, *The FLTSTS Register*), and a fault interrupt is sent to the MP (mf bit in the INTPEN register). If the request was from a PP, the processor will be unaware that a fault has occurred. The MP should clear the fault condition. When the corresponding fault flag in FLTSTS has cleared, the transfer is automatically resubmitted.

For more information on the packet-transfer suspend mechanism, see section 3.12, *Packet-Transfer Suspension*.

Stop (S) — Bit 31

This bit marks the end of a linked list. When a packet transfer is encountered with this bit set, the transfer is completed and the linked list is terminated. Before termination, however, the next entry address field is copied into the linked list start address location in the requesting processor's parameter RAM. If the linked list is re-enabled, execution begins at the next entry in the linked list. This allows the creation of circular linked lists, which are useful for repeated operations, such as ping-ponging between two memory areas. Stop bits can also be used to break linked lists at desired locations.

16.2.3 src/dst Start Address

Address: PT + 8 (src) / PT + 12 (dst)
Transfers: dimensioned, delta-guided

The 32-bit src/dst start address field gives the starting byte address for src or dst dimensioned transfers. For delta-guided transfers, it is the starting address to which the first delta offset is added for the src or dst transfer.

16.2.4 src/dst Base Address

Address: PT + 8 (src) / PT + 12 (dst)
Transfers: offset-guided

The 32-bit src/dst base address value takes the place of the start address when offset-guided transfer modes are used.

16.2.5 src/dst A Count

Address: PT + 16 bits 0-15 (src) / PT + 20 bits 0-15 (dst)
Transfers: dimensioned, fixed-patch

The 16-bit src/dst A count field specifies the number of bytes to be transferred in the first dimension of the src or dst for dimensioned or fixed-patch transfers. The A count field is unused for variable-patch guided transfers. In addition, src A count is unused for fill-with-value transfers.

16.2.6 src/dst B Count

Address: PT + 16 bits 16-31 (src) / PT + 20 bits 16-31 (dst)
Transfers: dimensioned, fixed-patch

The 16-bit src/dst B count field specifies the number of steps to occur in the second dimension of the src or dst for dimensioned or fixed-patch transfers. This is equal to the number of lines minus 1. A value of 0, therefore, disables the second dimension and results in the transfer of only one line per patch. The B count field is unused during variable-patch transfers. In addition, src B count is unused during fill-with-value transfers.

16.2.7 src/dst C Count

Address: PT + 24 (src) / PT + 28 (dst)
Transfers: dimensioned

The 32-bit src/dst C count field specifies the number of patch steps to occur in the third dimension of the src or dst for dimensioned transfers. Its value is equal to the number of patches minus 1. Values of 0 disable the third dimension and result in only one patch being transferred. The src C count field is unused for fill-with-value transfers.

16.2.8 src/dst Number of Entries

Address: PT + 24 (src) / PT + 28 (dst)
Transfers: guided

The 32-bit src/dst number-of-entries field specifies the number of entries in the guide table, and indicates the number of patches of information to be transferred. This value cannot be 0.

16.2.9 src/dst B Pitch

Address: PT + 32 (src) / PT + 36 (dst)
Transfers: dimensioned, fixed-patch

The 32-bit src/dst B pitch field specifies the pitch of the second dimension of the src and dst. This pitch value is added to the start address of the previous src or dst line. If the value in the corresponding B count field is 0, you can leave this field unprogrammed.

16.2.10 src/dst C Pitch

Address: PT + 40 (src) / PT + 44 (dst)
Transfers: dimensioned

This 32-bit src/dst C pitch field specifies the pitch of the third dimension of the src or dst. This pitch value is added to the start address of the previous src or dst patch. If the value in the corresponding C count field is 0, then this field can be left unprogrammed.

16.2.11 src/dst Guide Table Pointer

Address: PT + 40 (src) / PT + 44 (dst)
Transfers: guided

The *src/dst guide table pointer* field is loaded with an aligned address that points to the first entry in the guide table. This value is incremented by four (for fixed-patch transfers) or eight (for variable-patch transfers) each time an entry is taken from the table. It must be aligned to a 32-bit word address (fixed-patch) or 64-bit doubleword address (variable-patch), and must also point to on-chip memory. An interrupt error occurs if either condition is not true.

16.2.12 LS and MS Fill Value

Address: PT + 32 / PT + 40
Transfers: fill-with-value (src only)

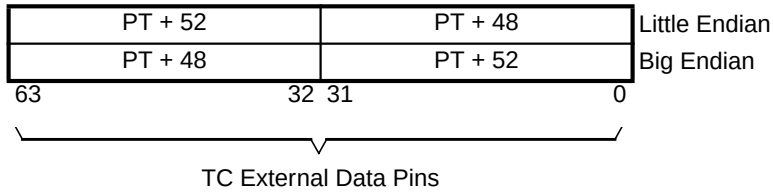
These two 32-bit fields are used when the fill-with-value mode is specified in a src transfer. Together they define a 64-bit fill pattern. The byte(s) written to the destination doubleword are taken from corresponding bytes within the fill value doubleword. Data values should be repeated if the fill pattern is less than 64 bits.

16.2.13 src Transparency Value

Address: PT + 48
Transfers: all src (with transparency enabled)

The src transparency value field indicates the value(s) that is /are to be compared with when transparency is selected. The number of values contained in this field is indicated by the transparency size, defined by the packet transfer access mode (PAM) coding in the PT options field. The byte(s) about to be written to the destination doubleword is/are compared against the corresponding bytes within the transparency value, and the byte strobes are driven inactive if a match is found. Data values should be replicated if the pixel size is less than 64 bits. For more details, see section 11.6, *Transparency-Packet Transfers*. This field is unused for all nontransparency transfers (except block write).

The src transparency value is used by the TC exactly as written to memory, regardless of the endian mode (no word-swap occurs). It appears as follows:



You can avoid confusion by writing the value as a doubleword write to address PT + 48.

16.2.14 Color Register Value

Address: PT + 48
Transfers: all src (with block write enabled)

The 64-bit color register field contains the value used to load the color registers of VRAMs/SGRAMs in preparation for block-write cycles. It is also used when performing simulated block writes. The TC uses the color register value exactly as written, regardless of the endian mode (just as it does the transparency value). Use of this field is described in detail in section 11.3, *The Block-Write Modes*.

16.2.15 Unused Field

Address: PT + 56
Transfers: all

The last doubleword of the packet transfer parameters is currently unused for all transfer modes and can be left unprogrammed.

16.3 Short-Form Packet-Transfer Parameters

Short-form PT parameter tables must reside in on-chip memory and must be 16-byte aligned (the four LSBs of the address are 0). The table may reside in any memory accessible to the requesting processor (as with long-form packet transfers). Figure 16–11 shows the format for short-form parameter tables in both big-endian and little-endian modes. The value PT represents the 16-byte-aligned starting address of the parameter table. The parameter table may be programmed independently of the endian mode by using 32-bit writes to the proper word address as shown in Figure 16–12.

Figure 16–11.Short-Form Parameter Table

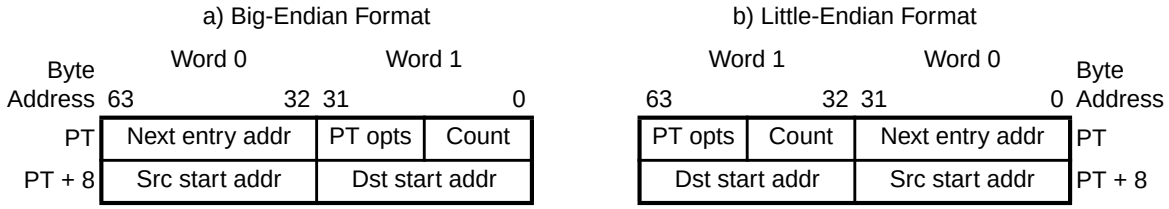
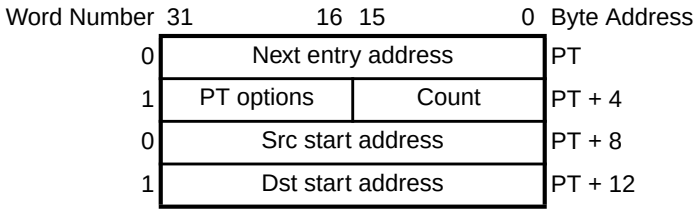


Figure 16–12.Short-Form Parameter Table (Endian Independent)



The following sections describe the five parameter fields used in short-form packet transfers. The address of each field is given relative to PT, which represents the memory address of the start of the parameter table.

16.3.1 Next Entry Address

Address: PT

The 32-bit next entry address points to the start of the next parameter table in the linked list for the packet-transfer request. To be a short-form transfer, this field must contain a 16-byte aligned, on-chip address (the four LSBs must be 0). For a standard packet transfer, the field must contain a 64-byte aligned, on-chip address (the six LSBs must be 0). If the current table is the last entry in the linked list then next entry address need not contain a valid address because the stop bit in the PT options terminates the linked list.

Whenever a packet transfer completes successfully, the value in the next entry address is written to the appropriate linked list start-address location in the processor's parameter RAM, to advance the pointer to the next packet transfer. The link list start-address pointer is also updated after the final packet in the link list is completed. If the stop bit is used to pause execution of the linked list, the start-address pointer points to the next transfer on the linked list when the list is restarted.

16.3.2 Count

Address: PT+4

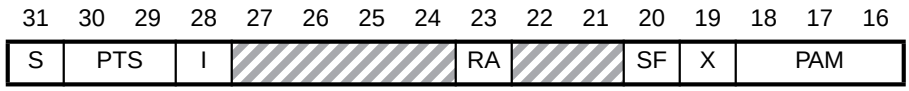
The Count field contains the number of contiguous bytes to be transferred from the src to the dst memory area. The 16-bit field allows from 0 to 65 535 to be transferred in a single short-form packet transfer. A value of 0 in count results in no data being transferred, however the overhead required to load the packet parameters, update the link-list start address and signal completion of the packet is still incurred.

16.3.3 Short-Form PT Options

Address: PT+6

The PT Options field specifies how to transfer the packet data. For short-form packet transfers, only a few of the bits are valid. The format of the 16-bit short form PT Options field is shown in Figure 16–13. Shaded bits are reserved and should be written as 0.

Figure 16–13.PT Options Field



Stop (S) - Bit 31

The S bit marks the end of a linked list. When a parameter table with this bit set is read, the transfer is completed and the linked list is terminated. Before termination, however, the next entry address is copied into the linked-list start-address location in parameter RAM. If the packet transfer is resubmitted, execution continues with the parameter table at the next entry address. This allows the creation of circular linked lists that are useful for repeated memory operations such as ping-ponging data between two memory areas. The stop bit can also break a linked list at a desired location so that processor activity may be synchronized to the progress of the packet transfers.

PT Status (PTS) - Bits 30–29

The PTS bits reflect the current state of the packet-transfer request. These bits must be set to 0 before submitting a packet-transfer request. If a packet transfer is suspended, the TC modifies the PTS bits in the copy of the PT Options field that it saves to the suspended packet-parameter area. This is because a suspended transfer contains more parameters than a new transfer and a different parameter-table load mechanism must be used. If either of the PTS bits is a 1 when the parameter table is being loaded, the TC recognizes that transfer is a suspended transfer and restores the extra parameters. Figure 16–9 shows the encoding of the PTS bits.

Table 16–10. PT Status (PTS Bits)

Bit		
30	**29**	**Packet Transfer Request Status**
0	0	Not suspended
0	1	Suspended but not faulted
1	0	Suspended due to fault on Src
1	1	Suspended due to fault on Dst

The TC writes a value of 01 to PTS if the packet transfer is suspended for any reason other than a fault. This could be because:

- ❏ The TC received a higher priority packet transfer request (and PTMIN for the current transfer has been exceeded).
- ❏ The packet transfer has timed out. (The request was serviced for PTMAX cycles after PTMIN was exceeded).
- ❏ The initiating processor suspended the request (by setting its packet transfer suspend bit).
- ❏ An error condition occurs (invalid next entry address, invalid options selected, and so forth).

The TC sets bit 30 of PT options if the packet transfer is suspended due to a fault. Bit 29 then indicates if the fault occurred on the src or dst. The MP can check these bits to determine where the fault occurred.

Note:
If the X bit is set in the suspended packet transfer options field, then the meaning of bit 29 for a faulted transfer is reversed. For example, a 10 indicates a fault on dst and a 11 indicates a fault on src. Interrupt service routines should examine both bits 29 and 19 when attempting to determine the cause of a faulted PT.

If a fault does occur, the appropriate bit in FLTSTS is set and a fault interrupt is sent to the MP (mf bit in INTPEN register). If the packet transfer was PP initiated, the processor is unaware that the fault occurred. The MP should correct the fault condition and clear the FLTSTS flag. Once the corresponding fault flag in FLTSTS is cleared, the packet transfer is automatically resubmitted.

Interrupt When Finished - Bit 28

Setting this bit to 1 causes an end-of-packet interrupt (the pc bit in the MP or the PTEND bit in the PP) to be sent to the processor that initiated the packet transfer as soon as this entry on the linked list has finished. The linked list can contain further entries. This flags the requesting processor when a particular point in the linked list has been reached.

Setting this bit to a 0 prevents an interrupt to the processor until either an entry with this bit set is encountered and completed, or the last packet transfer on the linked list has completed. If an error occurs at any time, however, the TC immediately sends an *error* interrupt (the bp interrupt bit in the MP's INTPEN register, or the PTERR interrupt bit in the PP's INTFLG register) to the requesting processor.

Reverse Addressing (RA) - Bit 23

Setting the RA bit to 1 causes the src and dst to be addressed backwards. Byte addressing is reversed. This means that the src start address and dst start address are the highest addresses accessed by the src and dst transfers respectively.

Short Form (SF) - Bit 20

The SF bit is used to select normal or short-form operation. When set to a 1, short-form mode is enabled. The parameter table is interpreted as four 32-bit words in length and containing short-form PT fields. When set to a 0, normal operation is enabled. The parameter table is interpreted as sixteen 32-bit words in length and containing normal PT fields.

Exchange Src and Dst Parameters (X) - Bit 19

Setting the X bit allows the direction of a packet transfer to be reversed without manually swapping the src and dst parameters. This is useful for returning transferred data back to its original location after processing.

When the X bit is set, the TC swaps the src and dst start addresses as it loads the packet-transfer parameter table. This exchange is performed whenever the parameters are loaded. If the transfer is suspended, the current parameters are swapped back to their original positions before being saved to the parameter RAM suspend area. If the suspended transfer is then restored, the parameters are again swapped as they are loaded by the TC.

PT Access Mode (PAM) - Bits 18–16

These three bits encode the special-access modes as shown in Table 16–6. The mode modifies the way in which the src data is written to the dst. For short-form packet transfers, only normal and peripheral-device transfers are allowed.

Appendix A

Examples of Packet-Transfer Parameter Tables

This appendix provides examples of normal packet transfer parameter tables. Table A–1 lists the 46 possible combinations of packet transfer *src* and *dst* operating modes. Figure A–1 through Figure A–46 show the parameter-table contents for each of the combinations in big- and little-endian formats. Each diagram also shows the location of the color-register value (used only when the block-write access mode is specified) and the transparency value (used only when one of the transparency access modes is specified).

Topic **Page**

A.1 Packet Transfer src and dst Operating Modes

Table A–1 lists the combinations of src and dst operating modes.

Table A–1. Available src and dst Transfer Combinations

No.	Source Operation	Destination Operation
1	Dimensioned	Dimensioned
2	Dimensioned	Fixed-patch, delta-guided
3	Dimensioned	Fixed-patch, offset-guided
4	Dimensioned	Variable-patch, delta-guided
5	Dimensioned	Variable-patch, offset-guided
6	Dimensioned	Peripheral device read
7	Fixed-patch, delta-guided	Dimensioned
8	Fixed-patch, delta-guided	Fixed-patch, delta-guided
9	Fixed-patch, delta-guided	Fixed-patch, offset-guided
10	Fixed-patch, delta-guided	Variable-patch, delta-guided
11	Fixed-patch, delta-guided	Variable-patch, offset-guided
12	Fixed-patch, delta-guided	Peripheral device read
13	Fixed-patch, offset-guided	Dimensioned
14	Fixed-patch, offset-guided	Fixed-patch, delta-guided
15	Fixed-patch, offset-guided	Fixed-patch, offset-guided
16	Fixed-patch, offset-guided	Variable-patch, delta-guided
17	Fixed-patch, offset-guided	Variable-patch, offset-guided
18	Fixed-patch, offset-guided	Peripheral device read
19	Fixed-patch, offset-guided, look-up table	Dimensioned
20	Fixed-patch, offset-guided, look-up table	Fixed-patch, delta-guided
21	Fixed-patch, offset-guided, look-up table	Fixed-patch, offset-guided
22	Fixed-patch, offset-guided, look-up table	Variable-patch, delta-guided
23	Fixed-patch, offset-guided, look-up table	Variable-patch, offset-guided
24	Fixed-patch, offset-guided, look-up table	Peripheral device read
25	Variable-patch, delta-guided	Dimensioned
26	Variable-patch, delta-guided	Fixed-patch, delta-guided
27	Variable-patch, delta-guided	Fixed-patch, offset-guided
28	Variable-patch, delta-guided	Variable-patch, delta-guided
29	Variable-patch, delta-guided	Variable-patch, offset-guided
30	Variable-patch, delta-guided	Peripheral device read
31	Variable-patch, offset-guided	Dimensioned
32	Variable-patch, offset-guided	Fixed-patch, delta-guided
33	Variable-patch, offset-guided	Fixed-patch, offset-guided
34	Variable-patch, offset-guided	Variable-patch, delta-guided
35	Variable-patch, offset-guided	Variable-patch, offset-guided
36	Variable-patch, offset-guided	Peripheral device read
37	Fill-with-value	Dimensioned
38	Fill-with-value	Fixed-patch, delta-guided
39	Fill-with-value	Fixed-patch, offset-guided
40	Fill-with-value	Variable-patch, delta-guided
41	Fill-with-value	Variable-patch, offset-guided
42	Peripheral device write	Dimensioned
43	Peripheral device write	Fixed-patch, delta-guided
44	Peripheral device write	Fixed-patch, offset-guided
45	Peripheral device write	Variable-patch, delta-guided
46	Peripheral device write	Variable-patch, offset-guided

A.2 Packet Transfer Parameter Table Listings

Figure A–1. Dimensioned src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst C Count		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src C Pitch		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Dimensioned

Figure A–2. Dimensioned src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–3. Dimensioned src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src C Count		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–4. Dimensioned src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src C Count		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Variable-Patch, Delta-Guided

Figure A–5. Dimensioned src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Start Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src C Count		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src C Pitch		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src C Count		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src C Pitch		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Variable-Patch, Offset-Guided

Figure A–6. Dimensioned src to Peripheral Device Read dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
0x00000000		src Start Address		PT + 8
0x0000	0x0000	src B Count	src A Count	PT + 16
0x00000000		src C Count		PT + 24
Don't Care		src B Pitch		PT + 32
Don't Care		src C Pitch		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		0x00000000	
PT + 16	src B Count	src A Count	0x0000	0x0000
PT + 24	src C Count		0x00000000	
PT + 32	src B Pitch		Don't Care	
PT + 40	src C Pitch		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Dimensioned
Destination Operation: Peripheral Device Read

Figure A–7. Fixed-Patch, Delta-Guided src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst C Count		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Dimensioned

Figure A–8. Fixed-Patch, Delta-Guided src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–9. Fixed-Patch, Delta-Guided src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Start Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–10. Fixed-Patch, Delta-Guided src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Variable-Patch, Delta-Guided

Figure A–11. Fixed-Patch, Delta-Guided src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Start Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Variable-Patch, Offset-Guided

Figure A–12. Fixed-Patch, Delta-Guided src to Peripheral Device Read dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
0x00000000		src Start Address		PT + 8
0x0000	0x0000	src B Count	src A Count	PT + 16
0x00000000		src No. of Entries		PT + 24
Don't Care		src B Pitch		PT + 32
Don't Care		src Guide Table Pointer		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		0x00000000	
PT + 16	src B Count	src A Count	0x0000	0x0000
PT + 24	src No. of Entries		0x00000000	
PT + 32	src B Pitch		Don't Care	
PT + 40	src Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Delta-Guided
Destination Operation: Peripheral Device Read

Figure A–13. Fixed-Patch, Offset-Guided src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst C Count		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Dimensioned

Figure A–14. Fixed-Patch, Offset-Guided src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–15. Fixed-Patch, Offset-Guided src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Base Address	
PT + 16	src B Count	src A Count	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–16. Fixed-Patch, Offset-Guided src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Variable-Patch, Delta-Guided

Figure A–17. Fixed-Patch, Offset-Guided src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Base Address		PT + 8
Don't Care	Don't Care	src B Count	src A Count	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Base Address	
PT + 16	src B Count	src A Count	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Variable-Patch, Offset-Guided

Figure A–18. Fixed-Patch, Offset-Guided src to Peripheral Device Read dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
0x00000000		src Base Address		PT + 8
0x0000	0x0000	src B Count	src A Count	PT + 16
0x00000000		src No. of Entries		PT + 24
Don't Care		src B Pitch		PT + 32
Don't Care		src Guide Table Pointer		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		0x00000000	
PT + 16	src B Count	src A Count	0x0000	0x0000
PT + 24	src No. of Entries		0x00000000	
PT + 32	src B Pitch		Don't Care	
PT + 40	src Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided
Destination Operation: Peripheral Device Read

Figure A–19. Fixed-Patch, Offset-Guided, Look-Up Table src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	Left Shift	PT + 16
dst C Count		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	Left Shift	dst B Count	dst A Count
PT + 24	src No. of Entries		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Dimensioned

Figure A–20. Fixed-Patch, Offset-Guided, Look-Up Table src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	Left Shift	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	Left Shift	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–21. Fixed-Patch, Offset-Guided, Look-Up Table src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Base Address		PT + 8
dst B Count	dst A Count	src B Count	Left Shift	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Base Address	
PT + 16	src B Count	Left Shift	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–22. Fixed-Patch, Offset-Guided, Look-Up Table src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
Don't Care	Don't Care	src B Count	Left Shift	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	src B Count	Left Shift	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Variable-Patch, Delta-Guided

Figure A–23. Fixed-Patch, Offset-Guided, Look-Up Table src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Base Address		PT + 8
Don't Care	Don't Care	src B Count	Left Shift	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Base Address	
PT + 16	src B Count	Left Shift	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Variable-Patch, Offset-Guided

Figure A–24. Fixed-Patch, Offset-Guided Look-Up Table src to Peripheral Device Read dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
0x00000000		src Base Address		PT + 8
0x0000	0x0000	src B Count	Left Shift	PT + 16
0x00000000		src No. of Entries		PT + 24
Don't Care		src B Pitch		PT + 32
Don't Care		src Guide Table Pointer		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		0x00000000	
PT + 16	src B Count	Left Shift	0x0000	0x0000
PT + 24	src No. of Entries		0x00000000	
PT + 32	src B Pitch		Don't Care	
PT + 40	src Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Fixed-Patch, Offset-Guided, Look-Up Table
Destination Operation: Peripheral Device Read

Figure A–25. Variable-Patch, Delta-Guided src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst C Count		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Dimensioned

Figure A–26. Variable-Patch, Delta-Guided src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Start Address		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–27. Variable-Patch, Delta-Guided src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Byte Address	Word 1 63–32		Word 0 31–0	
PT	PT Options		Next Entry Address	
PT + 8	dst Base Address		src Start Address	
PT + 16	dst B Count	dst A Count	Don't Care	Don't Care
PT + 24	dst No. of Entries		src No. of Entries	
PT + 32	dst B Pitch		src B Pitch	
PT + 40	dst Guide Table Pointer		src Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–28. Variable-Patch, Delta-Guided src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Byte Address	Word 1 63–32		Word 0 31–0	
PT	PT Options		Next Entry Address	
PT + 8	dst Start Address		src Start Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	dst No. of Entries		src No. of Entries	
PT + 32	dst B Pitch		src B Pitch	
PT + 40	dst Guide Table Pointer		src Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Start Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Variable-Patch, Delta-Guided

Figure A–29. Variable-Patch, Delta-Guided src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Byte Address	Word 1 (63–32)		Word 0 (31–0)	
PT	PT Options		Next Entry Address	
PT + 8	dst Base Address		src Start Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	dst No. of Entries		src No. of Entries	
PT + 32	dst B Pitch		src B Pitch	
PT + 40	dst Guide Table Pointer		src Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 (63–32)		Word 1 (31–0)	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		dst Base Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Variable-Patch, Delta-Guided

Figure A–30. Variable-Patch, Delta-Guided src to Peripheral Device Read dst

a) Little-Endian Format

Byte Address	Word 1 (63–32)		Word 0 (31–0)	
PT	PT Options		Next Entry Address	
PT + 8	0x00000000		src Start Address	
PT + 16	0x0000	0x0000	Don't Care	Don't Care
PT + 24	0x00000000		src No. of Entries	
PT + 32	Don't Care		src B Pitch	
PT + 40	Don't Care		src Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 (63–32)		Word 1 (31–0)	
PT	Next Entry Address		PT Options	
PT + 8	src Start Address		0x00000000	
PT + 16	Don't Care	Don't Care	0x0000	0x0000
PT + 24	src No. of Entries		0x00000000	
PT + 32	src B Pitch		Don't Care	
PT + 40	src Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Delta-Guided
Destination Operation: Peripheral Device Read

Figure A–31. Variable-Patch, Offset-Guided src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst C Count		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst C Pitch		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst C Count	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Dimensioned

Figure A–32. Variable-Patch, Offset-Guided src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–33. Variable-Patch, Offset-Guided src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		src Base Address		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Base Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–34. Variable-Patch, Offset-Guided src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		src Base Address		PT + 8
Don't Care	Don't Care	Don't Care	Don't Care	PT + 16
dst No. of Entries		src No. of Entries		PT + 24
dst B Pitch		src B Pitch		PT + 32
dst Guide Table Pointer		src Guide Table Pointer		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	src Base Address		dst Start Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	src No. of Entries		dst No. of Entries	
PT + 32	src B Pitch		dst B Pitch	
PT + 40	src Guide Table Pointer		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Variable-Patch, Delta-Guided

Figure A–35. Variable-Patch, Offset-Guided src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Byte Address	Word 1 63–32	Word 0 31–0
PT	PT Options	Next Entry Address
PT + 8	dst Base Address	src Base Address
PT + 16	Don't Care / Don't Care	Don't Care / Don't Care
PT + 24	dst No. of Entries	src No. of Entries
PT + 32	dst B Pitch	src B Pitch
PT + 40	dst Guide Table Pointer	src Guide Table Pointer
PT + 48	(Color Register/Transparency Value)	
PT + 56	Don't Care	

b) Big-Endian Format

Byte Address	Word 0 63–32	Word 1 31–0
PT	Next Entry Address	PT Options
PT + 8	src Base Address	dst Base Address
PT + 16	Don't Care / Don't Care	Don't Care / Don't Care
PT + 24	src No. of Entries	dst No. of Entries
PT + 32	src B Pitch	dst B Pitch
PT + 40	src Guide Table Pointer	dst Guide Table Pointer
PT + 48	(Color Register/Transparency Value)	
PT + 56	Don't Care	

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Variable-Patch, Offset-Guided

Figure A–36. Variable-Patch, Offset-Guided src to Peripheral Device Read dst

a) Little-Endian Format

Byte Address	Word 1 63–32	Word 0 31–0
PT	PT Options	Next Entry Address
PT + 8	0x00000000	src Base Address
PT + 16	0x0000 / 0x0000	Don't Care / Don't Care
PT + 24	0x00000000	src No. of Entries
PT + 32	Don't Care	src B Pitch
PT + 40	Don't Care	src Guide Table Pointer
PT + 48	Don't Care	
PT + 56	Don't Care	

b) Big-Endian Format

Byte Address	Word 0 63–32	Word 1 31–0
PT	Next Entry Address	PT Options
PT + 8	src Base Address	0x00000000
PT + 16	Don't Care / Don't Care	0x0000 / 0x0000
PT + 24	src No. of Entries	0x00000000
PT + 32	src B Pitch	Don't Care
PT + 40	src Guide Table Pointer	Don't Care
PT + 48	Don't Care	
PT + 56	Don't Care	

Source Operation: Variable-Patch, Offset-Guided
Destination Operation: Peripheral Device Read

Figure A–37. Fill-With-Value src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		Don't Care		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst C Count		Don't Care		PT + 24
dst B Pitch		Fill Value LS Word		PT + 32
dst C Pitch		Fill Value MS Word		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	Don't Care		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	Don't Care		dst C Count	
PT + 32	Fill Value LS Word		dst B Pitch	
PT + 40	Fill Value MS Word		dst C Pitch	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fill-With-Value
Destination Operation: Dimensioned

Figure A–38. Fill-With-Value src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		Don't Care		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst No. of Entries		Don't Care		PT + 24
dst B Pitch		Fill Value LS Word		PT + 32
dst Guide Table Pointer		Fill Value MS Word		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	Don't Care		dst Start Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	Don't Care		dst No. of Entries	
PT + 32	Fill Value LS Word		dst B Pitch	
PT + 40	Fill Value MS Word		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fill-With-Value
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–39. Fill-With-Value src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		Don't Care		PT + 8
dst B Count	dst A Count	Don't Care	Don't Care	PT + 16
dst No. of Entries		Don't Care		PT + 24
dst B Pitch		Fill Value LS Word		PT + 32
dst Guide Table Pointer		Fill Value MS Word		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	Don't Care		dst Base Address	
PT + 16	Don't Care	Don't Care	dst B Count	dst A Count
PT + 24	Don't Care		dst No. of Entries	
PT + 32	Fill Value LS Word		dst B Pitch	
PT + 40	Fill Value MS Word		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fill-With-Value
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–40. Fill-With-Value src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		Don't Care		PT + 8
Don't Care	Don't Care	Don't Care	Don't Care	PT + 16
dst No. of Entries		Don't Care		PT + 24
dst B Pitch		Fill Value LS Word		PT + 32
dst Guide Table Pointer		Fill Value MS Word		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	Don't Care		dst Start Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	Don't Care		dst No. of Entries	
PT + 32	Fill Value LS Word		dst B Pitch	
PT + 40	Fill Value MS Word		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fill-With-Value
Destination Operation: Variable-Patch, Delta-Guided

Figure A–41. Fill-With-Value src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		Don't Care		PT + 8
Don't Care	Don't Care	Don't Care	Don't Care	PT + 16
dst No. of Entries		Don't Care		PT + 24
dst B Pitch		Fill Value LS Word		PT + 32
dst Guide Table Pointer		Fill Value MS Word		PT + 40
(Color Register/Transparency Value)				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	Don't Care		dst Base Address	
PT + 16	Don't Care	Don't Care	Don't Care	Don't Care
PT + 24	Don't Care		dst No. of Entries	
PT + 32	Fill Value LS Word		dst B Pitch	
PT + 40	Fill Value MS Word		dst Guide Table Pointer	
PT + 48	(Color Register/Transparency Value)			
PT + 56	Don't Care			

Source Operation: Fill-With-Value
Destination Operation: Variable-Patch, Offset-Guided

Figure A–42. Peripheral Device Write src to Dimensioned dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		0x00000000		PT + 8
dst B Count	dst A Count	0x0000	0x0000	PT + 16
dst C Count		0x00000000		PT + 24
dst B Pitch		Don't Care		PT + 32
dst C Pitch		Don't Care		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	0x00000000		dst Start Address	
PT + 16	0x0000	0x0000	dst B Count	dst A Count
PT + 24	0x00000000		dst C Count	
PT + 32	Don't Care		dst B Pitch	
PT + 40	Don't Care		dst C Pitch	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Peripheral Device Write
Destination Operation: Dimensioned

Figure A–43. Peripheral Device Write src to Fixed-Patch, Delta-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Start Address		0x00000000		PT + 8
dst B Count	dst A Count	0x0000	0x0000	PT + 16
dst No. of Entries		0x00000000		PT + 24
dst B Pitch		Don't Care		PT + 32
dst Guide Table Pointer		Don't Care		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	0x00000000		dst Start Address	
PT + 16	0x0000	0x0000	dst B Count	dst A Count
PT + 24	0x00000000		dst No. of Entries	
PT + 32	Don't Care		dst B Pitch	
PT + 40	Don't Care		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Peripheral Device Write
Destination Operation: Fixed-Patch, Delta-Guided

Figure A–44. Peripheral Device Write src to Fixed-Patch, Offset-Guided dst

a) Little-Endian Format

Word 1 63–32		Word 0 31–0		Byte Address
PT Options		Next Entry Address		PT
dst Base Address		0x00000000		PT + 8
dst B Count	dst A Count	0x0000	0x0000	PT + 16
dst No. of Entries		0x00000000		PT + 24
dst B Pitch		Don't Care		PT + 32
dst Guide Table Pointer		Don't Care		PT + 40
Don't Care				PT + 48
Don't Care				PT + 56

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	0x00000000		dst Base Address	
PT + 16	0x0000	0x0000	dst B Count	dst A Count
PT + 24	0x00000000		dst No. of Entries	
PT + 32	Don't Care		dst B Pitch	
PT + 40	Don't Care		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Peripheral Device Write
Destination Operation: Fixed-Patch, Offset-Guided

Figure A–45. Peripheral Device Write src to Variable-Patch, Delta-Guided dst

a) Little-Endian Format

Byte Address	Word 1 63–32		Word 0 31–0	
PT	PT Options		Next Entry Address	
PT + 8	dst Start Address		0x00000000	
PT + 16	Don't Care	Don't Care	0x0000	0x0000
PT + 24	dst No. of Entries		0x00000000	
PT + 32	dst B Pitch		Don't Care	
PT + 40	dst Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	0x00000000		dst Start Address	
PT + 16	0x0000	0x0000	Don't Care	Don't Care
PT + 24	0x00000000		dst No. of Entries	
PT + 32	Don't Care		dst B Pitch	
PT + 40	Don't Care		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Peripheral Device Write
Destination Operation: Variable-Patch, Delta-Guided

Figure A–46. Peripheral Device Write src to Variable-Patch, Offset-Guided dst

a) Little-Endian Format

Byte Address	Word 1 63–32		Word 0 31–0	
PT	PT Options		Next Entry Address	
PT + 8	dst Base Address		0x00000000	
PT + 16	Don't Care	Don't Care	0x0000	0x0000
PT + 24	dst No. of Entries		0x00000000	
PT + 32	dst B Pitch		Don't Care	
PT + 40	dst Guide Table Pointer		Don't Care	
PT + 48	Don't Care			
PT + 56	Don't Care			

b) Big-Endian Format

Byte Address	Word 0 63–32		Word 1 31–0	
PT	Next Entry Address		PT Options	
PT + 8	0x00000000		dst Base Address	
PT + 16	0x0000	0x0000	Don't Care	Don't Care
PT + 24	0x00000000		dst No. of Entries	
PT + 32	Don't Care		dst B Pitch	
PT + 40	Don't Care		dst Guide Table Pointer	
PT + 48	Don't Care			
PT + 56	Don't Care			

Source Operation: Peripheral Device Write
Destination Operation: Variable-Patch, Offset-Guided

Appendix B

Glossary

B

block miss: A cache miss in which the addressed block is not resident in the cache. The least recently used (LRU) algorithm determines which existing cache block is discarded. If the cache contains any modified data (MP data cache only), then any modified subblocks are written back to external memory before the requested subblock is brought into cache.

block write: A nonstandard packet transfer that allows the TC to perform multicolumn write operations.

bubble: A pipeline cycle during which the TC performs no operation. Bubbles occur because of contention, an insufficient amount of data for the next access, or simply an absence of requests for activity.

C

cache: A fast memory into which frequently used data or instructions from slower memory are copied for fast access. Fast access is facilitated by the cache's high speed and its on-chip proximity to the CPU.

cache block: A section of cache memory. Each block has an associated tag register and is divided into four subblocks. Cache memory is allocated in block-size portions, but cache servicing is performed at the subblock level, with subblocks brought in as needed.

cache miss: The state or condition in which the cache does not contain the requested instruction or data word.

cache subblock: One of four partitions of a cache block. Cache subblocks are the unit of memory brought into a cache on a subblock miss. Each subblock has a present bit (and a dirty bit for MP data cache only) in the tag register for that block.

cache tag register: A register containing the address of the block whose subblock(s) have been copied into cache. It also contains a present bit for each subblock indicating whether or not the subblock is present in the cache. For MP data cache, there is also a dirty bit for each subblock.

$\overline{\text{CAS}}$: *Column address strobe*. A memory interface signal that drives the column address strobe inputs of DRAMs/VRAMs.

contention: A situation where two or more simultaneous access attempts for the same 2K-byte RAM are made. Contention is resolved automatically in hardware by arbitration, though a delay can occur.

crossbar: A generally configurable, high-speed bus switching network for a multiprocessor system, permitting any of several processors to connect to any of several memory modules.

D

data cache: The MP's two SRAM banks that hold cached data needed by the MP. Data RAMs for the PPs are not cached.

DEA: *Direct external access.* A method of accessing off-chip (external) memory without having to issue a packet transfer request to the TC.

delta-guided transfer: A type of guided packet transfer in which the guide table consists of 32-bit delta values to be added to the starting address of the previous two-dimensional patch to form the starting address of the new patch. See also *guided transfer.*

destination controller: One of two independent controllers within the TC that handle packet transfers. The destination controller generates the addresses needed to write packet data into the destination memory area.

dimensioned transfer: A transfer consisting of sources and/or destinations that can be a simple contiguous linear sequence of data bytes or can consist of a number of such regions. See also *guided transfer.*

direct external access: See *DEA.*

doubleword: A 64-bit value.

E

external address: See *off-chip address.*

externally initiated packet transfer: See *XPT.*

F

fill-with-value transfer: A type of source transfer that does not transfer data but instead specifies a source value within the packet transfer parameters. This fill value is used to fill the destination memory as defined by the destination transfer parameters.

fixed-patch guided transfer: Guided transfer that uses an on-chip guide table consisting of 32-bit word-aligned entries. See also *guided transfer.*

G

guide table: A table of parameters describing individual patches within a packet transfer. See also *patch.*

guided transfer: A transfer in which the sequence of dimension addresses is guided from an on-chip memory table, rather than calculated solely from values within the packet transfer parameters. See also *dimensioned transfer.*

H

halfword: A 16-bit value.

I

instruction cache: An on-chip SRAM that contains current instructions being executed by one of the TMS320C80 processors. Cache misses are handled by the transfer controller.

internal address: See *on-chip address.*

interrupt: An exceptional condition caused either by an event external to the processor or by a previously executed instruction that forces the current program to be interrupted. After the processor has serviced the interrupt, it typically resumes execution of the interrupted program at the instruction whose execution was interrupted.

J

JPEG standard: *Joint Photographic Experts Group standard*. A standard used for compressed still-picture data.

L

LCR cycle: *Load color register cycle*. A TC-generated memory cycle that writes a specified value into the VRAM's color registers for use during a block-write cycle. LCR cycles are supported only on 64-bit data buses and are indicated by STATUS [5:0] = 001101.

load color register cycle: See *LCR cycle.*

LSB: *Least significant bit.* The bit having the smallest effect on the value of a binary numeral, usually the rightmost bit. The TMS320C80 numbers the bits in a word from 0 to 31, where bit 0 is the LSB.

M

master processor: See *MP.*

memory fault: An exception caused by an attempt to access an illegal or invalid address in memory.

MP: *Master processor*. A general-purpose RISC processor that coordinates the activity of the other processors on the TMS320C80. The MP includes an IEEE-754 floating-point hardware unit.

MSB: *Most significant bit*. The bit having the greatest effect on the value of a binary numeral. It is the leftmost bit. The TMS320C80 numbers the bits in a word from 0 to 31, where bit 31 is the MSB.

O

off-chip address: An address external to the TMS320C80 chip. Addresses from 0x0200 0000 to 0xFFFF FFFF are off-chip addresses. See also *on-chip address*.

offset-guided look-up table transfer: Type of guided transfer in which the guide table consists of values to be left-shifted (zero-filled) by zero, one, two or three bits and added to a base address given in the packet transfer parameters to form the starting address of each patch. See also *guided transfer.*

offset-guided transfer: Type of source-guided transfer in which the guide table consists of values to be added to a base address given in the packet transfer parameters to form the starting address of each patch. See also *guided transfer*.

on-chip address: An address internal to the TMS320C80 chip. Addresses from 0x0000 0000 to 0x1FFF FFFF are on-chip addresses. See also *off-chip address.*

P

packet: A collection of patches of data. See also *patch.*

packet transfer: See *PT.*

packet transfer access mode: See *PAM.*

packet transfer request: An I/O request submitted to the TC that is issued when a block of data is to be moved via packet transfer. Packet transfer requests can be submitted by the MP, the PPs, the VC, or an external device.

PAM: *Packet transfer access mode.* A mode that modifies the method for writing source data to the destination, including normal, peripheral

device transfer, block write, SRT, and 8-bit, 16-bit, 32-bit, and 64-bit src transparency.

parallel processor: See *PP*.

parameter RAM: A general-purpose 2K-byte RAM that is associated with a specific processor, part of which is dedicated to packet transfer information and the processor interrupt vectors.

parameter table: A group of parameters, eight doublewords long, that describe a data packet and how it is to be moved from source to destination.

patch: A group of lines of equal length whose starting addresses are an equal distance apart.

pipeline stall: Temporary halt to the normal fetching of operations. Events which cause a pipeline stall include: a cache-miss, an illegal operation detection, diversion of local port access to global port, a DEA, and crossbar contention.

pipelining: A design technique for reducing the effective propagation delay per operation by partitioning the operation into a series of stages, each of which performs a portion of the operation. A series of data is typically clocked through the pipeline in sequential fashion, advancing one stage per clock period.

PP: *Parallel processor*. The TMS320C80's advanced digital signal processor that is used for video compression/decompression (P 64 or MPEG), still-image compression/decompression (JPEG), 2-D and 3-D graphic functions such as line draw, trapezoid fill, antialiasing, and a variety of high-speed integer operations on image data. An TMS320C80 single-chip multiprocessor device may contain from one to eight PPs, depending on the device version.

PT: *Packet transfer*. A transfer of data blocks between two areas of memory. The TMS320C80 supports packet transfers of one, two, or three dimensions. See also *dimensioned transfer*, *guided transfer.*

PT options field: Packet-transfer parameter field which selects the form of transfer for source and destination. It determines if the packet will end the linked list and enables the selection of additional features such as special transfer modes.

R

$\overline{\text{RAS}}$: *Row address strobe.* A memory interface signal that drives the row address strobe inputs of DRAMs/VRAMs.

refresh: A method of restoring the charge capacitance to a memory device (such as a DRAM or VRAM) or of restoring memory contents.

S

subblock: See *cache subblock.*

subblock miss: A cache miss where the desired block is present but the desired subblock is not. Results in a pipeline stall until the required subblock is brought into cache.

T

TC: *Transfer controller.* The TMS320C80's on-chip DMA controller for servicing the cache and for transferring one-, two-, and three-dimensional data blocks between each processor on the TMS320C80 and its external memory.

transfer controller: See *TC.*

transparency on source operation: A transfer by the TC in which the source data is compared to a transparency value on a byte-by-byte basis and these comparisons are grouped according to the transparency data size. If the compared bytes within a group match, the TC disables the corresponding byte strobes to prevent writes to any of the bytes within that group.

trickle refresh cycles: Low-priority refresh cycles. These refresh cycles occur only when the bus is idle.

U

urgent refresh cycles: High-priority refresh cycles that occur when the backlog of refresh requests exceeds 16. A burst of four refresh cycles is performed, with remaining refresh requests waiting for the completion of higher priority cycles.

V

variable-patch guided transfer: A type of guided transfer in which all patch size information is specified within the guide table rather than in the packet transfer parameters, allowing each patch within the transfer to have different dimensions. See also *delta-guided transfer*, *offset-guided transfer*, *guided transfer.*

VC: *Video controller*. The portion of the TMS320C80 responsible for the video interface.

video controller: See *VC.*

W

word: A sequence of 32 adjacent bits that constitutes a register or memory value. The PP supports 32-bit words. The MP also supports doublewords of 64 bits for loads and stores.

X

XPT: *Externally initiated packet transfer*. A packet transfer initiated by an external device through the TMS320C80's $\overline{\text{XPT}}$ [2:0] inputs.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X