TMS320C82
Transfer Controller
User’s Guide

Literature Number: SPRU261
June 1998

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserves the right to make changes to their products or to discontinue any product or service without notice, and advises customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current and complete. All products are sold subject to the terms and conditions of sale at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products 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 ("CRITIACL APPLICATIONS). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUTABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Contents

Figures

Tables

Chapter 1

Transfer Controller Overview

The TMS320C82’s *Transfer Controller* (TC) is a programmable controller responsible for moving data on and off chip. The TC contains the ’C82’s memory interface allowing it to interface directly to various memory devices and peripherals. The TC also acts as a DMA controller for the ‘C82’s processors (the MP and PPs) and peripherals by performing autonomous block data transfers (called packet transfers) between on-chip and/or off-chip memory.

Topics in this chapter include:

Topic **Page**

1.1 Transfer Controller Description

The TMS320C82 Transfer Controller (TC) is a flexible, high performance data transfer machine that provides the mechanism for moving data between on-chip and off-chip memory. The TC serves as a DMA controller by transferring data blocks as requested by the Master Processor (MP), Parallel Processors (PPs) or external peripheral devices. These autonomous transfers increase processor performance by freeing the MP and PPs to continue instruction execution while the TC transfers the data.

The TC contains a memory controller that provides address and control signals to interface directly to DRAMs, SDRAMs, SRAMs, ROMs and peripherals. The memory controller includes a unique cycle configuration cache that maintains configuration information for each of the memory and peripheral devices contained in the system. This allows the timing of each memory access to be tailored to the requirements of the device being accessed, thus minimizing access times.

1.1.1 Key Features

Key features of the T C include:

- ❑ 64-bit data transfers
  - ■ Dynamic bus sizing for 8-, 16-, 32-, or 64-bit devices
- ❑ 32-bit addressing
  - ■ 4-GByte address range
  - ■ Multiplexed row/column address bus for direct DRAM and SDRAM interface
- ❑ Memory configuration cache
  - ■ Stores memory configuration and timing information for up to six different memories and peripherals
  - ■ Auto-loads memory configuration on first access to each bank.
  - ■ Multi-level Least Recently Used (LRU) replacement algorithm allows prioritization of system memory banks.
- ❑ Up to 480 MByte/sec data transfer rate
- ❑ Versatile addressing capabilities
- ❑ Intelligent queueing and prioritization of data transfer requests
- ❑ Big or little endian operation (selected at reset)

1.1.2 Transfer Controller Functions

Since the TC provides the only access to off-chip memory, it is responsible for all data movement and memory interface functions. These functions include:

❑ MP and PP Instruction Cache Fills
❑ MP Data Cache Fills and Dirty Subblock Write-back
❑ MP and PP requested packet transfers (PTs)
❑ Externally initiated packet transfers (XPTs)
❑ MP and PP Direct External Accesses (DEAs)
❑ DRAM and SDRAM Refresh
❑ External Bus Requests

Cache operations are requested automatically by the cache logic contained within each processor. The TC moves the requested instructions or data from memory to the appropriate subblock in the cache array of the requesting processor. The TC will also write back the dirty subblocks of the MP's data cache when required by the MP.

Packet transfers (PTs) move blocks of data from one location to another. They are most frequently used to transfer data into or out of the PP data RAMs. The TC performs packet transfers when requested by the processors. Packet transfers may also be initiated by external devices. These XPTs can be used to assist in system I/O by transferring data between on-chip or system memory and I/O buffers such as video or audio data FIFOs.

During direct external accesses (DEAs) the TC transfers data directly between off-chip memory and processor registers. These cycles are generated by MP or PP load or store instructions. They allow the PPs to directly access off-chip memory and let the MP by-pass it's data cache to access off-chip memory directly.

External bus requests cause the TC to place memory interface signals into high impedance so that they may be driven directly by a host processor or another device.

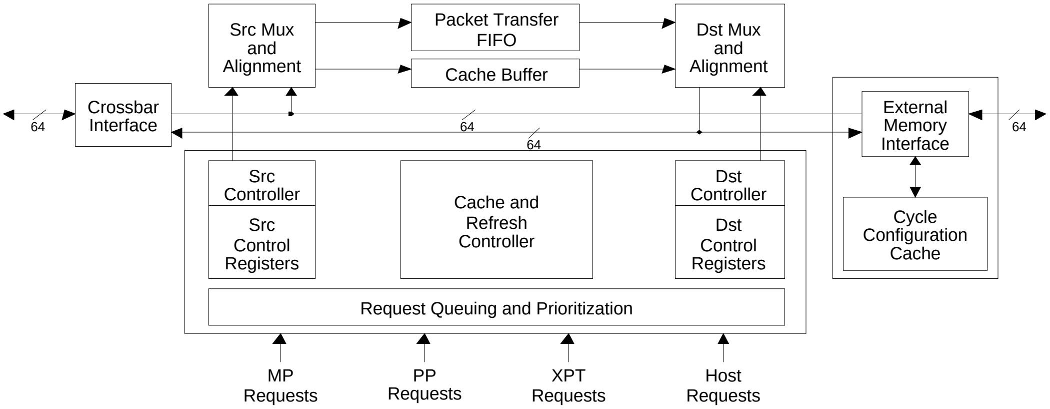
DRAM refresh cycles keep system memory devices refreshed. These are generated automatically by the TC's programmable refresh controller. Refresh may be disabled in systems which do not use dynamic memory.

The various requests to the TC are prioritized by importance. When necessary, the TC will time-multiplex the memory interface between requests to ensure that all requests get serviced.

1.2 Functional Blocks

Figure 1–1. shows a high-level block diagram of the Transfer Controller. A brief description of each of the major blocks follows the diagram.

Figure 1–1. Transfer Controller Block Diagram



1.2.1 Request Queuing and Prioritization Logic

The TC evaluates all requests from the MP, PPs, XPT pins, and host interface pins and services them based on a fixed prioritization scheme. When multiple requests of the same priority are pending, the TC services them on a round-robin basis.

1.2.2 Cache and Refresh Controller

The TC contains a programmable refresh controller that automatically generates DRAM refresh cycles as required by the external memory system. The TC's cache control logic generates the addresses necessary to perform cache fills (and write-backs) as requested by the MP and PPs.

1.2.3 Source / Destination Controllers

The TC contains two independent controllers that handle packet transfers:

- ❏ The **source controller** generates the addresses necessary to fetch the packet data from the source memory. When a packet transfer request is submitted to the TC, the transfer has 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 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 destination addresses.

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

1.2.4 Internal Memory (Crossbar) Interface

The TC's internal memory interface provides access to on-chip memory via the 'C82's crossbar. The 64-bit crossbar bus can transfer zero to eight bytes per cycle. Each crossbar access requires two clock cycles to complete (assuming no contention) but a new access can begin on every 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 control signals and cycle timing necessary to interface to a variety of memory and peripheral devices. The 64-bit external data bus can transfer up to eight bytes per cycle.

1.2.6 Source/Destination Mux and Alignment and PT FIFO

During packet transfers, data alignment is limited only to byte-alignment (transfers may begin and end on any byte boundary). Since the source and destination controllers are independent, they may transfer different numbers of bytes (from zero to eight) on any given cycle. This means that the alignment of the source and destination addresses with respect to each other can be constantly changing. To support the fluctuating alignment, the TC contains a packet transfer FIFO (first-in, first-out) register and alignment logic. The packet transfer FIFO is a 16-byte FIFO that can be simultaneously loaded with up to eight bytes from the source and emptied off up 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 extracts the oldest bytes from the FIFO and aligns them to the correct position in the currently addressed destination doubleword. Transfer alignment and FIFO operation is automatic and transparent to the programmer.

1.2.7 Cache Buffer

The cache buffer is a 16-byte buffer similar in operation to the packet transfer FIFO. It is used during cache, DEA, and short-form XPT operations. TC transfers in and out of the cache RAMs 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, DEA, and short-form XPT 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.

1.2.8 Cycle Configuration Cache

The cycle configuration cache stores configuration information for up to six different memory banks at a time. This configuration information controls the operation of the external memory interface's state sequencer allowing each memory cycle to be optimized for the type of device being accessed. When a memory bank whose configuration is not cached is accessed, a configuration cycle is automatically performed to load the new bank configuration information into the cache.

1.3 Transfer Controller Registers

The TC contains four programmable 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, double-word load and stores are not permitted. The TC's registers cannot be accessed by the PPs.

The four TC registers are:

- **REFCNTL** — Controls system DRAM refresh cycles
- **PTMIN** — Determines minimum packet transfer time
- **PTMAX** — Determines maximum packet transfer time
- **FLTSTS** — Indicates source of memory faults that occur during TC operations

The TC register map is summarized in Table 1–1. Each register is described in detail in the following subsections.

Table 1–1. Transfer Controller Register Map

Address	TC Register
0x01820000	REFCNTL
0x01820004	PTMIN
0x01820008	PTMAX
0x0182000C	FLTSTS

Reserved bits in the TC registers should be written as zeros to ensure compatibility with future devices.

1.3.1 The REFCNTL Register

Address: 0x01820000

Format:

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">RPARLD</td><td colspan="16">REFRATE</td></tr>
</table>

Fields:

Bits	Name	Function
15-0	REFRATE	Refresh Interval
31-16	RPARLD	Pseudo-Address Reload Value

Description: The refresh control register (**REFCNTL**) contains two 16-bit values that control the operation of the TC's refresh controller.

REFRATE *Refresh interval, bits 15-0*

The 16-bit **REFRATE** field determines the interval at which DRAM refresh cycle request will be generated within the TC. The value in **REFRATE** represents the number of TMS320C82 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 refresh cycles, a 16-bit pseudo-address is output on AD[31:16] and RCA[16:1] of the ‘C82’s memory interface to be used for refresh bank decoding. **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.

1.3.2 The PTMIN Register

Address: 0x01820004

Format:

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
Reserved								PTMIN																							

Description: The 24-bit packet transfer minimum length register (**PTMIN**) indicates the minimum number of clock cycles that a packet transfer must be serviced by the TC before it can be interrupted by a higher priority packet transfer. **PTMIN** is loaded with the value 0x00010000 (64K cycles) at reset. For more information on programming **PTMIN** see Section 9.2, *Controlling PT Execution Time.*

1.3.3 The PTMAX Register

Address: 0x01820008

Format:

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
Reserved								PTMAX																							

Description: The 24-bit packet transfer maximum length register (**PTMAX**) indicates the maximum number of clock cycles that a packet transfer can continue after being serviced for the time specified by PTMIN before it will time-out. Once **PTMIN** + **PTMAX** cycles have been used by the packet transfer, it will be suspended so that another packet transfer of the same priority may be serviced. (**PTMAX** does not affect when suspension will occur for a higher priority packet transfer - this is dependent solely on the **PTMIN** value.) **PTMAX** is loaded with the value 0x00010000 (64K cycles) at reset. For more information on programming **PTMAX** see Section 9.2, *Controlling PT Execution Time.*

1.3.4 The FLTSTS Register

Address: 0x0182000C

Format:

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
Reserved						PC1	PC0	Reserved						PP1	PP0	Reserved							XPT				Reserved				M

Fields:

Bits	Name	Function
0	M	MP Packet Transfer Fault
8-5	XPT	XPT Fault or Error
17-16	PP*n*	PP*n* Packet Transfer Fault
25-24	PC*n*	PP*n* Cache or DEA Fault

Description: The fault status (**FLTSTS**) register contains status bits that indicate faults which have occurred during packet transfers, PP instruction cache fills, or DEA cycles.

M *MP Packet Transfer Fault, bit 0*
The **M** bit is set to 1 when a fault occurs during a packet transfer that was requested by the MP.

XPT *Externally Initiated Packet transfer Fault, bits 8 - 5*
The **XPT** bits are set to indicate that an externally initiated packet transfer has faulted. A faulted XPT is immediately terminated and the XPT number copied into the **XPT** field. The TC will ignore all further XPT requests on the XPT[3:0] inputs until the **XPT** bits in **FLTSTS** have been cleared.

Bit 8 7 6 5	Faulted Request	Bit 8 7 6 5	Faulted Request
0 0 0 0	No XPT fault	1 0 0 0	XPT 8 fault
0 0 0 1	XPT 1 fault	1 0 0 1	XPT 9 fault
0 0 1 0	XPT 2 fault	1 0 1 0	XPT 10 fault
0 0 1 1	XPT 3 fault	1 0 1 1	XPT 11 fault
0 1 0 0	XPT 4 fault	1 1 0 0	XPT 12 fault
0 1 0 1	XPT 5 fault	1 1 0 1	XPT 13 fault
0 1 1 0	XPT 6 fault	1 1 1 0	XPT 14 fault
0 1 1 1	XPT 7 fault	1 1 1 1	XPT 15 fault

PP *PP Packet Transfer Fault, bits 17-16*

A **PP** bit is set 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.

PC *PP Instruction Cache-fill / DEA Fault, bits 25-24*

A PC bit is set when a fault occurs during a PP instruction cache-fill or DEA operation.

- ❏ **PC0** (bit 24) indicates a PP0 cache-fill/DEA fault.
- ❏ **PC1** (bit 17) indicates a PP1 cache-fill/DEA fault.

The setting of any one of the **M**, **XPT**, **PP0-1**, or **PC0-1** bits causes the **mf** bit in the MP’s **INTPEN** register to be set. Clearing a **FLTSTS** bit that has been set causes the associated packet transfer or cache-fill request to be rescheduled. Clearing the **FLTSTS** bit(s) in the event of an XPT fault will result in the TC’s sampling of the XPT[3:0] pins. The **FLTSTS** bit(s) can be cleared by writing a ‘1’ to the appropriate bit(s). Writing ‘0’ to the bit(s) has no effect. **FLTSTS** bit(s) must be cleared before clearing the **mf** bit in the **INTPEN** register to ensure correct operation.

Chapter 2

Transfer Controller Priorities

The TC must service many types of requests from the on-chip processors as well as from off-chip devices. In order to ensure optimum system performance, these requests are prioritized by their urgency and importance. Because the TC services these requests of different priorities, its own priority on the internal crossbar (with regards to the MP and PPs) will vary from access to access. This chapter discusses how the TC prioritizes its requests.

Topics in this chapter include:

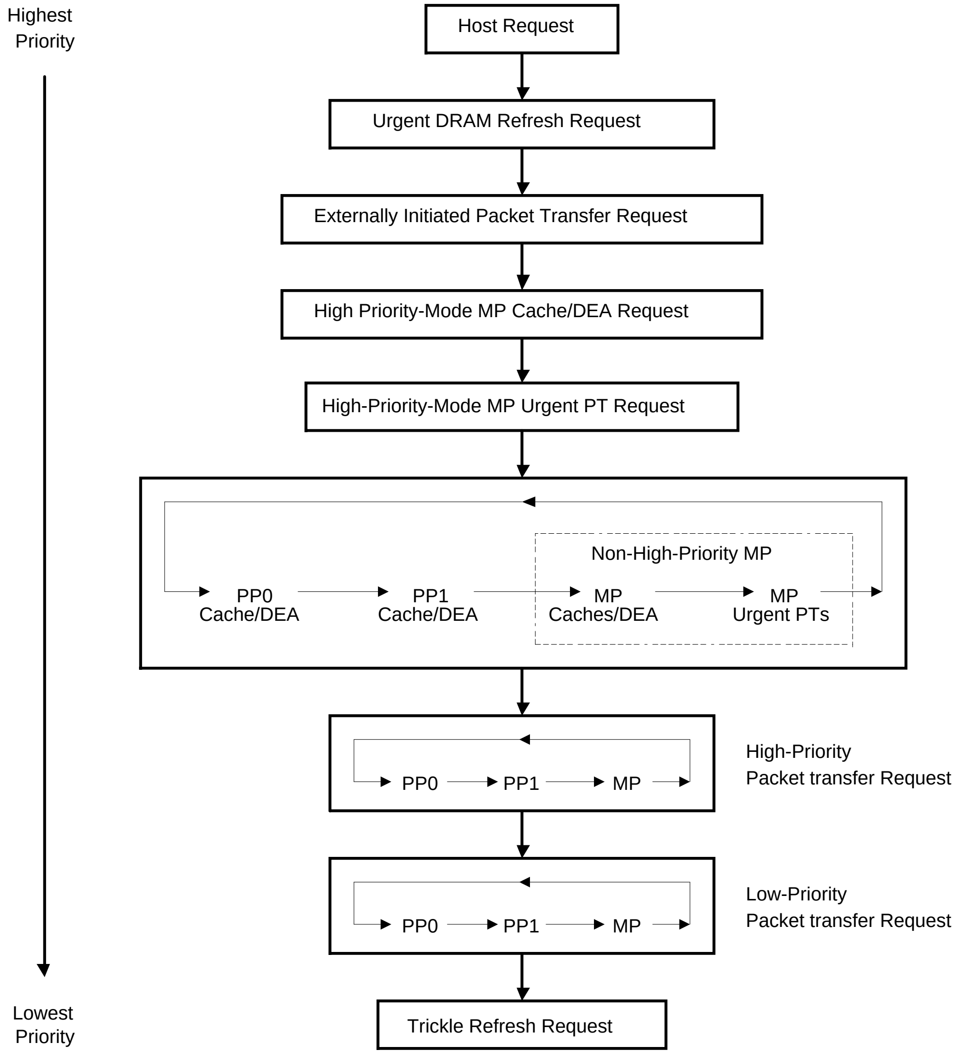
Topic **Page**

2.1 Request Prioritization

The various requests that the TC can receive are shown in Figure 2–1, prioritized from top to bottom. An explanation of each priority follows the diagram.

The TC will service the highest priority request which it receives. If the TC is currently servicing a request and it receives a higher priority request, it will switch to the higher priority request as quickly as possible.

Figure 2–1. Transfer Controller Request Prioritization


Highest Priority
Host Request
Urgent DRAM Refresh Request
Externally Initiated Packet Transfer Request
High Priority-Mode MP Cache/DEA Request
High-Priority-Mode MP Urgent PT Request
Non-High-Priority MP
PP0 Cache/DEA
PP1 Cache/DEA
MP Caches/DEA
MP Urgent PTs
PP0
PP1
MP
High-Priority Packet transfer Request
PP0
PP1
MP
Low-Priority Packet transfer Request
Trickle Refresh Request
Lowest Priority

The prioritization of each request is explained below:

- **Host Request.** An external device, such as a system host is always given highest priority so that it may access ‘C82 memory whenever necessary. The device can return control to the TC for high priority requests (such as urgent refreshes) by monitoring the status of the ‘C82’s REQ (high priority request) output. (For more information on host requests see Chapter 20, *Host Interface*)
- **Urgent DRAM Refresh Request.** When the TC has been unable to perform scheduled trickle refreshes, the system is in danger of loosing memory data. Refresh priority temporarily raised to ensure that needed refreshes are performed. (See Section 19.2, *Refresh Priority)*
- **Externally Initiated Packet Transfer (XPT) Request.** XPTs are typically used to perform system critical real-time data transfers. To ensure timely completion, these requests are given the highest priority of any data transfer.
- **High-Priority-Mode MP Cache/DEA Request.** The next priority level is for MP cache service and DEA requests that occur when the MP is operating in high priority mode. This allows interrupt service routines and other high priority MP tasks to be performed quickly thereby reducing interrupt latency and maximizing system performance.
- **High-Priority-Mode MP Urgent Packet Transfer Request.** These requests may be critical transfers within an MP interrupt service routine or other high priority MP task. They are prioritized below the high-priority-mode MP cache and DEA requests so as not to stall MP instruction execution.
- **Cache, DEA and MP Urgent Packet Transfer Requests.** PP cache service and DEA requests are next in priority. It is important that these be serviced quickly because the processor is stalled until the request is serviced. MP cache service, DEA, and urgent packet transfers requests when the MP is not operating in high-priority mode are also serviced on this level. Multiple requests on this level are serviced on a round-robin basis.
- **High-Priority Packet Transfer Requests.** Packet transfer requests submitted at high priority may imply that the requesting processor is waiting for the data to be transferred or that the TC needs priority over the PPs during crossbar accesses to optimize external bus bandwidth. Multiple requests on this level are serviced on a round robin basis.
- **Low-Priority Packet Transfer Requests.** Packet transfer requests submitted at low priority typically specify background data transfers when the processor is not waiting for the data. They are given the lowest data transfer priority. Multiple requests on this level are serviced on a round robin basis.

- **Trickle Refresh Request.** Trickle refreshes are given lowest priority. They are performed only when the external memory bus is idle and the refresh backlog is not 0. These cycles help to lower the refresh backlog and reduce the likelihood of a high-priority urgent refresh being requested.

2.2 Round-Robin Operation

Whenever the TC receives multiple requests of the same priority from different processors, it will service the requests on a **round-robin** basis. The round robin is a fixed cyclical priority scheme. No processor can be removed from the round robin cycle and the order of the processors within the cycle cannot be changed, When a processor's request has been serviced, 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 one exception to this operation occurs during MP cache and DEA requests. Because of the way MP requests are pipelined, it is possible for the MP to request another cache/DEA service before the round-robin token is passed to the next request in the chain. For this reason, back-to-back MP cache and/or DEA requests may temporarily lock-out PP requests of the same priority.

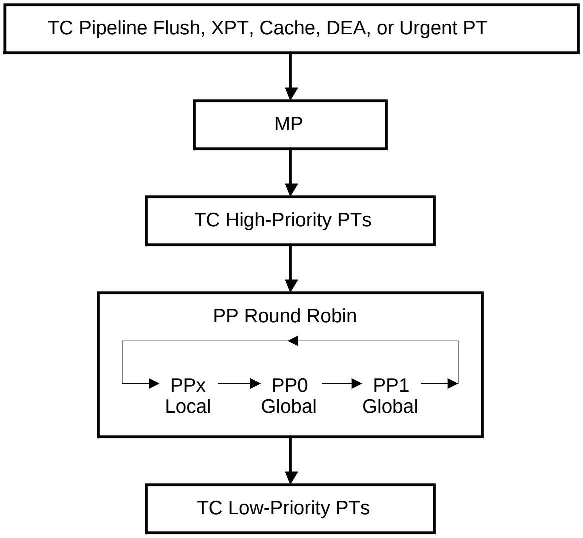
The round robin operation can be disabled by setting the **T** bit in the MP's **CONFIG** register to a 1. When the round-robin is disabled, the MP will always be serviced first followed by PP0 and PP1 when multiple requests of the same priority are pending.

Note that the round-robin operation only affects how multiple requests of the same priority are serviced. Higher priority requests will always be serviced first regardless of where the request originates.

2.3 Crossbar Priority

The ‘C82’s internal crossbar allows only one access to each on-chip RAM on a given clock cycle. Because the TC may attempt to access the same RAM as a processor, a priority scheduling mechanism is used to grant the RAM to the processor with the highest priority or to the TC. The priority level of the TC on the crossbar changes dynamically according to the level of the request which the TC is servicing as seen in Figure 2–2.

Figure 2–2. Crossbar Access Priority



The TC’s crossbar 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, an XPT request or when flushing its pipeline. Pipeline flushing occurs when the TC receives an urgent refresh or host request and must flush external memory accesses currently in its pipeline in order to service the request. These can only lock out the MP from a RAM access for a short period of time.

❏ The TC operates above the PP priority but below that of the MP for high-priority packet transfers. This gives the TC the greatest bandwidth into crossbar RAMs without locking out the MP which may be performing critical system functions such as interrupt service.

- The TC operates below the MP priority 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 will be suspended at the high-priority level when a high priority request is received.

Note that the crossbar priority scheme is only implemented when crossbar contention occurs (when more than one processor (or the TC) attempts to access the same internal RAM in the same clock cycle). Best results will be obtained by programming crossbar accesses to avoid contention. (For example structure data transfers such that the PP is processing data contained in one of its data RAMs while the TC transfers previously processed data (using a PT) from another data RAM. Since the PP and TC are accessing different RAMs, no contention will occur and neither will be locked out.)

Chapter 3

Cache Service and DEA Requests

This chapter discusses how the TC services MP and PP instruction cache, MP data cache, and direct external access (DEA) requests.

Topics in this chapter include:

3.1 Cache Service Requests

The MP and PPs each have their own instruction cache and the MP also has a data cache. Whenever one of the processors experiences a cache miss, it sends a cache service request to the transfer controller. The TC automatically fetches the instructions (or data) required by the processor and places them in the processor’s on-chip cache allowing the processor to continue program execution.

3.1.1 PP Cache Miss Servicing

The PP instruction caches are one-way set associative (fully associative) caches consisting of four blocks. Each block contains eight 128-byte (16-instruction) subblocks. When a PP experiences a cache miss (the next instruction is not present in the cache), its program flow control (PFC) unit signals the TC, requesting a cache miss service. The PP determines the off-chip memory address of the needed instructions and the cache block address in which the instructions should be placed and passes this information to the TC. The TC fetches a complete subblock (128 bytes) from off-chip memory and places it in the appropriate cache subblock of the requesting PP. The TC then signals the PP that the request has been serviced so that the PP can continue execution.

3.1.2 PP Cache Service Latency

Table 3–1 shows the minimum latencies of PP instruction cache fills. The numbers shown represent the number of clock cycles for which the requesting PP will be stalled while waiting for the cache miss to be serviced by the TC. Values are given both for when the TC is already accessing the requested memory page (page hit) and for when the TC needs to initiate an access to a new memory page (page miss). All values assume that the requested memory bank is already loaded in the TC’s configuration cache.

Table 3–1. PP Instruction Cache Service, Minimum Latencies

		Minimum Latency	
CT(3:0)	**Cycle Timing**	**Page Hit**	**Page Miss**
0 0 0 0	DRAM: Pipelined 1 Cyc/Col	28	32
0 0 0 1	DRAM: 1 Cyc/Col	27	31
0 0 1 0	DRAM: 2 Cyc/Col	43	47
0 0 1 1	DRAM: 3 Cyc/Col	60	64
0 1 0 0	SRAM: Synchronous 1 Cyc/Col	28	31
0 1 0 1	SRAM: 1 Cyc/Col	26	29

0 1 1 0	SRAM: 2 Cyc/Col	42	45
0 1 1 1	SRAM: 3 Cyc/Col	58	61
1 0 0 0	SDRAM: Burst 1, Read Latency 2	28	32
1 0 0 1	SDRAM: Burst 1, Read Latency 3	29	33
1 0 1 0	SDRAM: Burst 1, Read Latency 4	30	34
1 0 1 1	Reserved	-	-
1 1 0 0	SDRAM: Burst 2, Read Latency 2	28	32
1 1 0 1	SDRAM: Burst 2, Read Latency 3	29	33
1 1 1 0	SDRAM: Burst 2, Read Latency 4	30	34
1 1 1 1	Reserved	-	-

The latency numbers shown in Table 3–1 assume 64-bit wide external memory. For accesses to 8-, 16-, or 32-bit wide memory, the number of additional column access cycles necessary to complete the transfer must be added to the given values. (See Section 12.2, *Dynamic Bus Sizing*.)

The table shows only minimum latency values. It does not include the time it takes for the cache service round robin to reach the requesting PP. The request may have to wait for all other processors to have their cache service and DEA requests performed and for other higher priority requests to complete before being serviced. Since cache fills require multiple accesses to memory, they can also be interrupted by higher priority requests before completion. Latencies will also be increased if wait states or other exceptions caused by system hardware occur during the memory access.

3.1.3 MP Cache Miss Servicing

MP instruction caches are four-way set associative caches, with each set consisting of four blocks. Each block contains eight 64-byte (16-instruction) subblocks. The MP instruction cache service request is handled identically to the PP instruction cache service request, except that the size of the MP subblock fetched from off-chip memory is only 64 bytes.

The MP data cache is also a four-way set associative cache with eight 64-byte subblocks in each of four blocks. The MP will request a data cache service when the off-chip memory location accessed by a load (`ld`) or store (`st`) instruction is not present in the cache. When the MP sends a data cache request to the TC, the TC fetches a 64-byte subblock, and places it in the appropriate data cache subblock.

In addition to keeping track of which subblocks are present in its data cache, the MP also monitors which subblocks have been modified (by `st` instructions) using a dirty bit associated with each subblock. 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 the dirty subblocks in the least recently used block before fetching the new data. The MP will also request a dirty subblock write back when it executes a `dcachec` or `dcachef` cache flush instruction.

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

3.1.4 MP Cache Service Latency

Table 3–2 shows the minimum latencies of MP cache fills. The numbers shown represent the number of clock cycles for which the MP will be stalled while waiting for the cache miss to be serviced by the TC. Values are given both for when the TC is already accessing the requested memory page (page hit) and for when the TC needs to initiate an access to a new memory page (page miss). All values assume that the requested memory bank is already loaded in the TC's configuration cache.

Table 3–2. MP Cache Service, Minimum Latencies

		Minimum Latency	
CT(3:0)	**Cycle Timing**	**Page Hit**	**Page Miss**
0 0 0 0	DRAM: Pipelined 1 Cyc/Col	21	25
0 0 0 1	DRAM: 1 Cyc/Col	20	24
0 0 1 0	DRAM: 2 Cyc/Col	28	32
0 0 1 1	DRAM: 3 Cyc/Col	37	41
0 1 0 0	SRAM: Synchronous 1 Cyc/Col	21	24

0 1 0 1	SRAM: 1 Cyc/Col	19	22
0 1 1 0	SRAM: 2 Cyc/Col	27	30
0 1 1 1	SRAM: 3 Cyc/Col	35	38
1 0 0 0	SDRAM: Burst 1, Read Latency 2	21	25
1 0 0 1	SDRAM: Burst 1, Read Latency 3	22	26
1 0 1 0	SDRAM: Burst 1, Read Latency 4	23	27
1 0 1 1	Reserved	-	-
1 1 0 0	SDRAM: Burst 2, Read Latency 2	21	25
1 1 0 1	SDRAM: Burst 2, Read Latency 3	22	26
1 1 1 0	SDRAM: Burst 2, Read Latency 4	23	27
1 1 1 1	Reserved	-	-

The latency numbers shown in Table 3–2 assume 64-bit wide external memory. For accesses to 8-, 16-, or 32-bit wide memory, the number of additional column access cycles necessary to complete the transfer must be added to the given values. (See Section 12.2, *Dynamic Bus Sizing*.)

The table shows only minimum latency values. It does not include the time it takes for the cache service round robin to reach the MP. The request may have to wait for all other processors to have their cache service and DEA requests performed and for other higher priority requests to complete before being serviced. Since cache fills require multiple accesses to memory, they can also be interrupted by higher priority requests before completion. Latencies will also be increased if wait states or other exceptions caused by system hardware occur during the memory access.

The minimum cache latencies do not always apply to MP data cache requests. Because the MP's registers are scoreboarded, data cache misses do not necessarily stall the MP instruction pipeline. A store (`st`) instruction which causes a data cache miss will stall the MP if another attempt to access data (from on-chip RAMs, memory-mapped registers, or off-chip memory) is made. (This is because the MP's data port is busy waiting for the cache fill to complete.) A load (`ld`) instruction will stall the MP under the same conditions or if there is an attempt to use the register(s) being loaded. The data cache latency may increase significantly if a block miss occurs. In this case, all dirty subblocks must be written back to memory before the cache fill can begin.

3.2 DEA Requests

The transfer controller is responsible for handling all direct external access (DEA) requests from the MP and PPs. DEA cycles transfer data directly between off-chip memory and a processor's registers. DEA accesses can be a byte, halfword, word or doubleword (MP only) in length. DEAs are given a high priority level (the same as cache operations) but are limited to a single access. This limits the possibility of a single processor monopolizing the external memory bus with multiple DEA cycles which would prevent other processor's DEA and cache requests from being serviced. DEA cycles are intended to be used when fast access a single off-chip memory location (such as a program variable or off-chip register) is needed.

3.2.1 PP DEA Requests

Rather than process data contained in a data cache, the PPs are designed to operate on data placed within their data RAMs by packet transfers. The PPs have direct access to PP parameter and data RAMs through the crossbar. Data accesses to all other memory locations automatically cause a DEA request to be sent to the TC. When the requesting PP's turn in the cache/DEA priority round-robin is reached, the TC will service the DEA request. If the request is to off-chip memory (address `0x02000000` or above), the TC will transfer the requested datum between the PP and memory. If the request is to an on-chip address not accessible to the PP via the crossbar (such as the MP data cache) the TC will generate a DEA fault (see Section 11.5.2, *PP Cache/DEA Faults*.)

3.2.2 MP DEA Requests

The MP normally accesses off-chip data through its data cache. However, when trying to read a single program variable or off-chip hardware register it may not be desirable to fill an entire cache subblock just to access a single location. It is also sometimes desirable to have writes occur directly to the off-chip location rather than being posted to the cached location which prevents them being written off-chip until the data cache is flushed. In these cases it is possible for the MP program to bypass its data cache and access the off-chip location directly using a DEA. MP DEA requests are generated explicitly by the program by using special versions of the `ld` and `st` instructions, `dld` and `dst`.

If the MP attempts to perform a `ld` or `st` (or a `dld` or `dst`) operation on a non-accessible on-chip address (such as a PP instruction cache), the operation is converted to a DEA request which will be faulted by the TC (see Section 11.5.3, *MP Cache/DEA Faults*). A `dld` or `dst` instruction to an on-chip address directly accessible to the MP via the crossbar (such as PP data

RAM) is automatically converted into a normal `ld` or `st` operation and no DEA request occurs.

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 as this could result in the cache and off-chip locations containing different data.

3.2.3 DEA Service Prioritization

DEA requests are serviced by the TC using the cache/DEA prioritization level. A PP may have either a DEA or instruction cache service request active but not both at once. The cache/DEA round-robin token always passes on to the next requesting processor after completion of a PP DEA. The MP may have an instruction cache *and* either a DEA or data cache service active at one time. The TC will service both the instruction cache and DEA during the MP's round robin turn if both are pending. Because of the handshake mechanism used between the MP and TC, it is also possible for the MP to request another DEA at the same time at which a previous DEA is completing. This next DEA will be serviced before the advance of the round-robin token. Thus back-to-back `dld` or `dst` instructions on the MP will result in back-to-back service by the TC (assuming no higher priority requests).

3.2.4 DEA Service Latency

Table 3–3 shows the minimum latencies of DEA requests generated by an MP load (`dld`) or store (`dst`) instruction, or by a PP access of off-chip memory. The numbers shown represent the number of clock cycles for which the requesting processor may be stalled while waiting for the DEA to be serviced by the TC. Values are given both for when the TC is already accessing the requested memory page (page hit) and for when the TC needs to initiate an access to a new memory page (page miss). All values assume that the requested memory bank is already loaded in the TC's configuration cache.

Table 3–3. DEA Service, Minimum Latencies

		DEA Load Latency		DEA Store Latency	
CT(3:0)	**Cycle Timing**	**Page Hit**	**Page Miss**	**Page Hit**	**Page Miss**
0 0 0 0	DRAM: Pipelined 1 Cyc/Col	12	16	7	10
0 0 0 1	DRAM: 1 Cyc/Col	11	15	7	10
0 0 1 0	DRAM: 2 Cyc/Col	12	16	7	10
0 0 1 1	DRAM: 3 Cyc/Col	14	18	7	10
0 1 0 0	SRAM: Synchronous 1 Cyc/Col	12	15	7	10
0 1 0 1	SRAM: 1 Cyc/Col	10	13	7	10
0 1 1 0	SRAM: 2 Cyc/Col	11	14	7	10
0 1 1 1	SRAM: 3 Cyc/Col	12	15	7	10
1 0 0 0	SDRAM: Burst 1, Read Latency 2	12	16	7	10
1 0 0 1	SDRAM: Burst 1, Read Latency 3	13	17	7	10
1 0 1 0	SDRAM: Burst 1, Read Latency 4	14	18	7	10
1 0 1 1	Reserved	-	-	7	10
1 1 0 0	SDRAM: Burst 2, Read Latency 2	12	16	7	10
1 1 0 1	SDRAM: Burst 2, Read Latency 3	13	17	7	10
1 1 1 0	SDRAM: Burst 2, Read Latency 4	14	18	7	10
1 1 1 1	Reserved	-	-	7	10

The latency numbers shown in Table 3–3 assume 64-bit wide external memory. For accesses to 8-, 16-, or 32-bit wide memory, the number of additional column access cycles necessary to complete the transfer must be added to the given values. This number will depend on the size of the DEA transfer (byte, halfword, word, or doubleword) as well as the memory bus width. (See Section 12.2, *Dynamic Bus Sizing*.)

The table shows only minimum latency values. It does not include the time it takes for the cache service round robin to reach the requesting processor. The request may have to wait for all other processors to have their cache service and DEA requests performed and for other higher priority requests to complete before being serviced. Latencies will also be increased if wait states or other exceptions caused by system hardware occur during the memory access.

The DEA latencies do not always apply to MP DEA requests. Because the MP's registers are scoreboarded, MP DEAs do not necessarily stall the MP instruction pipeline. The MP will stall due to a DEA store (`dst`) only if another attempt to access data (from on-chip RAMs, memory-mapped registers, or off-chip memory) is made. A DEA load (`dld`) will stall under the same condition or if there is an attempt to use the register(s) being loaded by the

DEA. The PPs will always stall until the DEA they have requested has completed.

Back-to-back DEA requests to the same memory page do not always occur in page mode because the DEAs are individual transfers and other requests may be serviced between the DEAs. A DEA request will always be serviced using page mode if a page hit occurs when the TC begins the service. For more information on page mode operation see Section 10.5.6, *Page Size*.

Because of the latencies involved in performing DEA cycles they should typically be used only for single accesses. If more than two data transfers are involved, packet transfers are usually a more efficient transfer method.

Chapter 4

Packet Transfer Overview

The packet transfer is the primary method for transferring data on and off chip. A packet transfer can be requested by the MP, PPs, or external devices. This chapter introduces the basic concepts of packet transfers, the terminology used to discuss them, and packet transfer management by the TC.

Topics in this chapter include:

Topic **Page**

4.1 Introduction to Packet Transfers

A packet transfer (PT) is a transfer of a block (or blocks) of data between two areas of TMS320C82 memory. A packet transfer is performed by the Transfer Controller and causes data to be transferred from a source (src) memory area to a destination (dst) memory area. The src and dst memory areas can either be off-chip or on-chip memory.

Packet transfers are initiated by the MP, PP(s), or external devices as requests to the TC. The TC processes these requests using the fixed and round-robin prioritizations discussed in ***insert reference to round robin***. Once a processor has submitted a request for a packet transfer, it can resume its program execution. The packet transfer is completed by the TC without the need for additional processor cycles.

Packet transfers can be submitted with different priority levels wherein higher priority PTs interrupt lower priority PTs. When this occurs, the lower priority packet transfer is *suspended* by the TC, and its current state saved. When the higher priority PT is completed, the suspended PT resumes automatically at the point at which it was interrupted.

4.1.1 Packet Transfer Terminology

The following is a list of terms and their definitions used in the discussion of packet transfers.

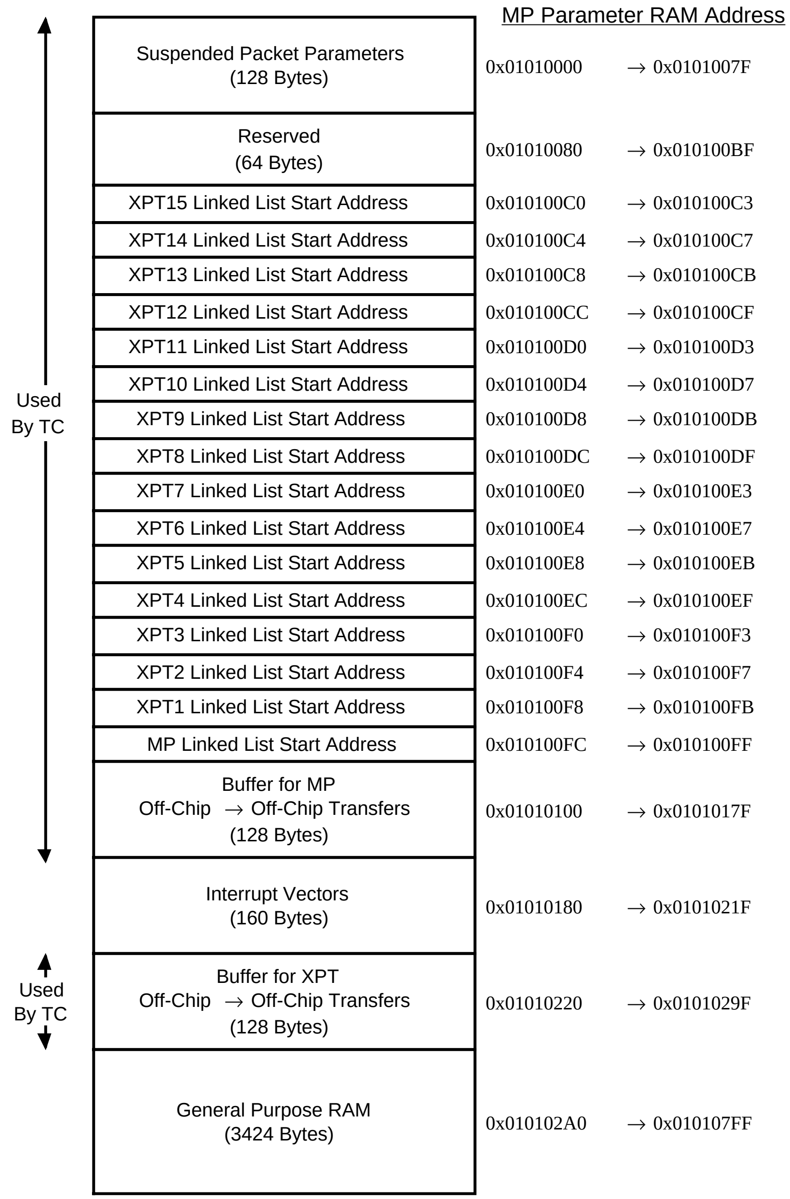
- ❏ **Line:** A number of contiguous bytes in memory.
- ❏ **Patch:** A group of lines of equal length whose starting addresses are an equal distance apart.
- ❏ **Packet:** A collection of patches.
- ❏ **Pitch:** The difference in addresses between the start of two lines or two patches.
- ❏ **Parameter 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.

- **Long-form packet transfer:** An extremely flexible type of packet transfer which can perform any type of data transfer (1-, 2-, or 3-Dimensional data) using a wide variety of addressing modes.
- **Short-form packet transfer:** A packet transfer with a compact parameter table, limited to a single block of contiguous data and with restricted addressing modes.

4.2 Packet Transfer Parameter RAM Usage

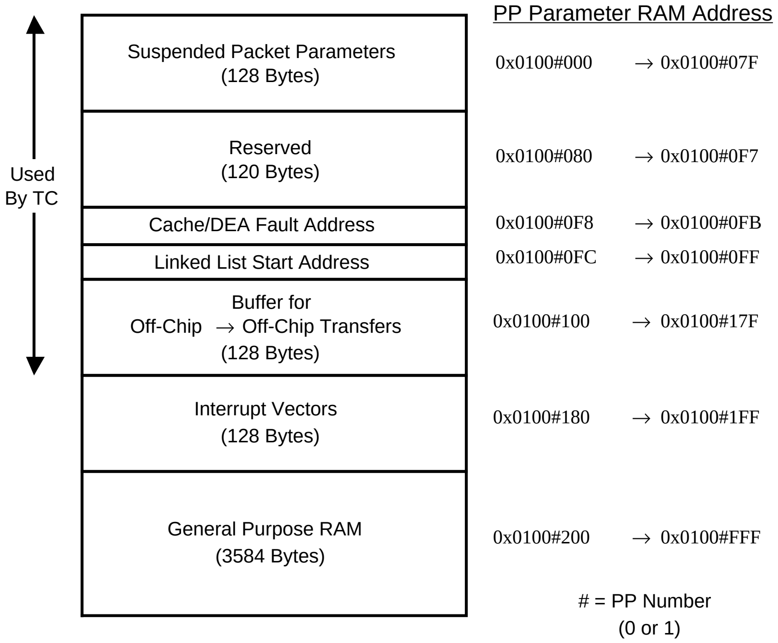
The MP and the PPs each have a number of locations set aside in the parameter RAM for the TC to use when servicing that processor's packet transfer requests. These areas are shown in Figure 4–1 and Figure 4–2.

Figure 4–1. MP Parameter RAM Contents



	MP Parameter RAM Address
Suspended Packet Parameters (128 Bytes)	0x01010000 → 0x0101007F
Reserved (64 Bytes)	0x01010080 → 0x010100BF
XPT15 Linked List Start Address	0x010100C0 → 0x010100C3
XPT14 Linked List Start Address	0x010100C4 → 0x010100C7
XPT13 Linked List Start Address	0x010100C8 → 0x010100CB
XPT12 Linked List Start Address	0x010100CC → 0x010100CF
XPT11 Linked List Start Address	0x010100D0 → 0x010100D3
XPT10 Linked List Start Address	0x010100D4 → 0x010100D7
XPT9 Linked List Start Address	0x010100D8 → 0x010100DB
XPT8 Linked List Start Address	0x010100DC → 0x010100DF
XPT7 Linked List Start Address	0x010100E0 → 0x010100E3
XPT6 Linked List Start Address	0x010100E4 → 0x010100E7
XPT5 Linked List Start Address	0x010100E8 → 0x010100EB
XPT4 Linked List Start Address	0x010100EC → 0x010100EF
XPT3 Linked List Start Address	0x010100F0 → 0x010100F3
XPT2 Linked List Start Address	0x010100F4 → 0x010100F7
XPT1 Linked List Start Address	0x010100F8 → 0x010100FB
MP Linked List Start Address	0x010100FC → 0x010100FF
Buffer for MP Off-Chip → Off-Chip Transfers (128 Bytes)	0x01010100 → 0x0101017F
Interrupt Vectors (160 Bytes)	0x01010180 → 0x0101021F
Buffer for XPT Off-Chip → Off-Chip Transfers (128 Bytes)	0x01010220 → 0x0101029F
General Purpose RAM (3424 Bytes)	0x010102A0 → 0x010107FF

Figure 4–2. PP Parameter RAM Contents



Parameter RAM areas used by the TC have no restriction on use by the local processor for other means. However data placed in these locations can be overwritten by the TC during a packet transfer operation. Never write to these locations when a packet transfer request is active or has been suspended because this will corrupt the data being transferred or the transfer mechanism itself.

4.3 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. Long-form PT parameter tables must be 64-byte aligned, short-form PT parameter tables must be 16-byte aligned.

2) If necessary, create the packet transfer's guide table(s) in on-chip memory. See Section 6.3.1, *Guide Tables*

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

4) 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 of the MP, and the **comm** register of the PPs. Detailed descriptions of how to set these bits are included in Chapter 7, *Understanding the Packet Transfer Request Protocol*, in the *TMS320C8x Master Processor User's Guide* and Chapter 12, *Packet Transfers*, in the *TMS320C8x Parallel Processor User's Guide*.

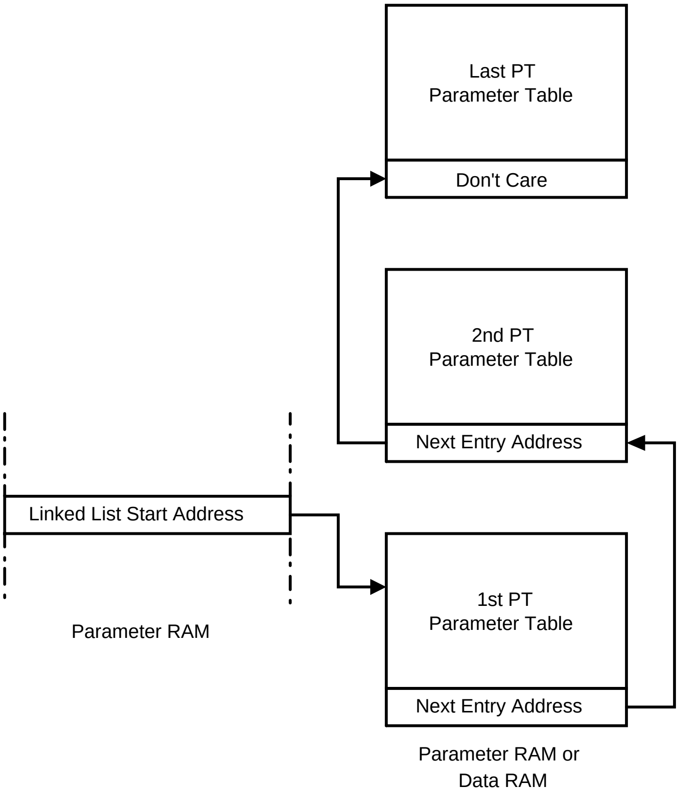
4.3.1 The Linked List Structure

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 parameter table pointing to the next one in 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 parameter RAMs or data RAMs.

Each processor can have any number of linked lists stored in memory, but only one list can be active at a time. The starting address of the active linked list is written to the dedicated Linked List Start Address location in the requesting processor's parameter RAM (see Figure 4–1 and Figure 4–2). Each parameter table in the linked list contains a pointer (called the Next Entry Address) to the next parameter table in the list. The end of the linked list is marked by a **S** (stop) bit in the packet transfer parameter table's PT Options field.

A simple linked list structure is shown in Figure 4–3. This list contains three packet transfers, but a linked list can be of any length, as long as it fits in available on-chip memory.

Figure 4–3. Linked List Structure



4.3.2 Packet Transfer Service

A processor submits a packet transfer by setting it's **P** bit (in the **PKTREQ** register of the MP or **comm** register of a PP). On the next clock cycle, the **Q** bit (also in the **PKTREQ** or **comm** register) is automatically set and the **P** bit is cleared indicating that the processor's packet transfer request has been queued within the TC. When the round-robin token appropriate to the packet transfer priority level reaches the requesting 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. This tells the TC where to find the first parameter table.

2) The TC reads the packet transfer parameter table beginning at the address it read from the Linked List Start Address. The parameter table contents are placed into registers within the TC's source and destination controllers.

3) The TC uses its source and destination controllers to transfer the data as indicated by the transfer parameters.

4) When the packet transfer is complete, the TC updates the Linked List Start Address in the processor's parameter RAM to point to the next address in the linked list (i.e. the parameter table's Next Entry Address).

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

6) When the packet transfer is completed, the requesting processor's **Q** bit is cleared and the processor is sent a packet-transfer-complete interrupt (the **pc** bit in the MP's **INTPEN** register or the **PTEND** bit in a PP's **intflg** register).

More details on linked list operation are available in Section 9.3, *Linked List Management*.

There is no hardware mechanism to ensure coherency between the MP's data cache and packet transfers. Packet transfers which modify cached locations will change the off-chip memory only and will not alter the cache contents.

Chapter 5

Short-Form Packet Transfers

Short-form packet transfers provide a simple method for moving small blocks of contiguous data within TMS320C82 memory space. This chapter describes how to setup short-form PTs and their operation.

Topics in this chapter include:

Topic **Page**

5.1 Short-Form Packet Transfer Overview

A Short-form packet transfer moves a single contiguous block of data. The data block can be up to 64K bytes in length and is moved from the src memory area to the dst memory area. Short-form PTs can be programmed more quickly and take up less on-chip memory space than long-form PTs because they have a small parameter table. Short-form PTs also incur less TC overhead resulting in faster execution of linked-lists.

Although limited in their addressing capability when compared to long-form PTs, short-form PTs offer a number of options to enhance their operation. A special operating mode also exists to allow short-form transfers to perform the Read Transfer (Memory-to-Register) Cycles required to control VRAM-based display buffers.

5.2 Short-Form Packet Transfer Parameter Tables

Short-form packet transfers are specified using a 16-byte parameter table. The format of the short-form parameter table is shown in Figure 5–1. The TC loads the parameter table by reading 64-bit double words. The ordering of the parameters within the doublewords is different for the big and little endian formats as seen in Figure 5–1. This allows software to create the parameter table independent of endian mode by using 32-bit writes to the proper word address as shown by Figure 5–2. In both diagrams, the address of each of the parameter fields are shown relative to the start of the packet transfer parameter table (represented by PT).

The short-form packet transfer parameter tables must be located in MP or PP parameter RAMs or PP data RAMs. The start of each parameter table must be 16-byte aligned (the four LSBs of the address must be zero). The tables can reside in any processor's RAM in the sense that the MP can use a PP's parameter RAM to store its packet transfer parameter table and vice-versa. The processor requesting a short-form packet transfer simply places the appropriate starting address in the Linked List Start Address location in its *own* parameter RAM before submitting its transfer request.

Figure 5–1. Short-Form Parameter Table Format

a) Big-Endian Format

Byte Address	Word 0	Word 1	
	63 32	31	0
PT	Next Entry Addr	PT Opt	Count
PT+8	Src Start Addr	Dst Start Addr	

b) Little-Endian Format

Word 1		Word 0	Byte Address
63	32	31 0	
PT Opts	Count	Next Entry Addr	PT
Dst Start Addr		Src Start Addr	PT+8

Figure 5–2. Short-Form Parameter Table (Endian Independent)

Word Number	31 16	15 0	Byte Address
0	Next Entry Address		PT
1	PT Options	Count	PT+4
0	Src Start Address		PT+8
1	Dst Start Address		PT+12

5.3 Short-Form Packet Transfer Parameter Fields

The following sections describe each of the fields in the short-form packet transfer parameter table. The address of each field relative to the start of the parameter table (designated by address “PT”) is shown.

5.3.1 Next Entry Address

Address: PT

This is a 32-bit address field which points to the start of the next entry (parameter table) in the packet transfer request’s linked list. If the next entry is a short-form packet transfer, this must point to a 16-byte aligned on-chip address (four LSBs must be zero). If the next entry in the linked list is a long-form packet transfer, this must point to a 64-byte aligned on-chip address. The last parameter table does not require a valid Next Entry Address because the linked-list is terminated by the stop bit (**S**) in the PT Options field.

When a packet transfer completes successfully, the value in Next Entry Address is written to the Linked List Start Address location in the requesting processor’s parameter RAM. This advances the pointer to the next packet transfer in the list. The pointer is also updated after the final packet transfer in the linked list is completed. Thus, if the stop bit is being used to pause linked list execution, the pointer will be pointing to the next transfer on the linked list when the list is re-enabled.

The Next Entry Address in the last parameter table is often programmed to point back to the beginning of the linked list. This allows the processor to request the same linked list of transfers be performed again by simply resubmitting a PT request. (The processor is saved the overhead of reprogramming the Linked List Start Address.)

5.3.2 Count / Update Value

Address: PT + 4 (lower 16 bits)

The value in the Count/Update field indicates the number of bytes to be transferred by the short-form packet transfer. The 16-bit field allows from 0 to 65,535 bytes to be specified. This field also indicates the amount by which the src and/or dst Start Address will be updated when an update mode is enabled in the PT Options field.

5.3.3 PT Options

Address: PT + 4 (upper 16 bits)

The 16-bit PT Options field specifies various options concerning how data will be transferred for the current short-form packet transfer. This field is discussed in detail in Section 5.4, *Short-Form Packet Transfer Options Field*.

5.3.4 src Start Address

Address: PT + 8

The 32-bit src Start Address field provides the starting byte address of the source data to be read. This must be an off-chip address or legal on-chip address (i.e. data RAM or parameter RAM).

5.3.5 dst Start Address

Address: PT + 12

The 32-bit dst Start Address field provides the starting byte address of the destination where data is to be written. This must be an off-chip address or legal on-chip address (i.e. data RAM or parameter RAM).

5.4 Short-Form Packet Transfer Options Field

The PT Options field contains bits to enable various special features during the packet transfer. These features include special access modes, reversing transfer direction, etc. The register also contains information about the status of the packet transfer and determines if the current PT will end the linked list.

The format of the short-form PT Options field is shown in Figure 5–3. Each bit field is described in the following sections

Figure 5–3. Short Form PT Options Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S	PTS		I	▨		DU	▨	RA	SU	▨	SF	X	PAM		

- ❏ PAM - Short-Form PT Access Mode
- ❏ X - Exchange src/dst
- ❏ SF - Short-Form Packet Transfer
- ❏ SU - src Update
- ❏ RA - Reverse Addressing
- ❏ DU - dst Update
- ❏ I - Interrupt When Finished
- ❏ PTS - Packet Transfer Status
- ❏ S - Stop

5.4.1 Short-Form PT Access Mode (PAM) - Bits(18:16)

These three bits select the access mode to be used by the TC when performing the packet transfer. The available modes and their encodings are shown in Table 5–1. Specifying a reserved value will result in a packet transfer fault. Each of the special modes is described in Chapter 8, *Special Packet Transfer Access Modes*

Table 5–1. Short-From PT Access Modes

Bit			
18	**17**	**16**	**Access Mode**
0	0	0	Normal
0	0	1	Peripheral Device Transfer
0	1	0	Split Read Transfer
0	1	1	Read Transfer
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved

1	1	1	Reserved

- **Mode 000** specifies normal access. No special addressing modes are used on the src or dst. Data is transferred from src to dst without alteration.

- **Mode 001** selects peripheral device transfer mode. This mode allows another device to read from or write to the TMS320C82 external memory using the TC as the memory controller. A device read of memory is accomplished by programming the src Start Address to the off-chip location to be read and programming dst Start Address to `0x00000000`. Device writes to memory are accomplished by programming the dst Start Address to the off-chip location to be written and src Start Address to `0x00000000`. In either case, the TC drives the memory address and control signals normally (with address generated according to the transfer parameters), but places the data bus in a high impedance state so that the peripheral device can read or write data. For more details, see Section 8.3, *Peripheral Device Transfer Mode*

- **Mode 010** causes the PT to generate split read (memory-to-split register) transfer cycles. Split read transfers are used to update the inactive half of a VRAM serial register during an active line of a VRAM display buffer. If src update is enabled (**SU** bit of PT Options) then a single split read transfer is performed. If src update is not enabled then the TC will perform the number of split read transfers specified by the Count parameter.

- **Mode 011** causes the PT to generate full read (memory-to-register) transfer cycles. Read transfers are used to update both halves of a VRAM serial register during the inactive (blanked) portion of a line of a VRAM display buffer. If src update is enabled (**SU** bit of PT Options) then a single read transfer is performed. If src update is not enabled then the TC will perform the number of read transfers specified by the Count parameter.

Split read and read transfer modes (**PAM** = ‘`01X`’) are supported for off-chip locations only. Attempting a split read transfer or read transfer with an on-chip src start address will cause the packet transfer to suspend with an error (see Section 9.4, *Packet Transfer Errors)*.

5.4.2 Exchange src and dst Parameters (X) - Bit(19)

Setting this bit reverses the direction of a packet transfer without the need to manually swap the src and dst parameters. This is useful for returning processed data to its original location using the same packet transfer that read in the data.

When the **X** bit is set, the transfer controller swaps all the src- and dst-related values when it loads the packet transfer parameters. Table 5–2 shows the word swap that occurs, assuming that PT represents the address of the parameter table's start address. The Next Entry Address field at PT and the PT Options field at PT + 4 remain in the same location as they are not src/dst related. In addition to swapping the src- and dst-starting addresses, the TC also swaps the src- and dst-update bits within the PT Options field. This is shown in Table 5–3.

Table 5–2. Exchange src/dst Parameter Word Swap

Source	Byte Address		Byte Address	Destination
src Start Address	PT + 8	´	PT + 12	dst Start Address

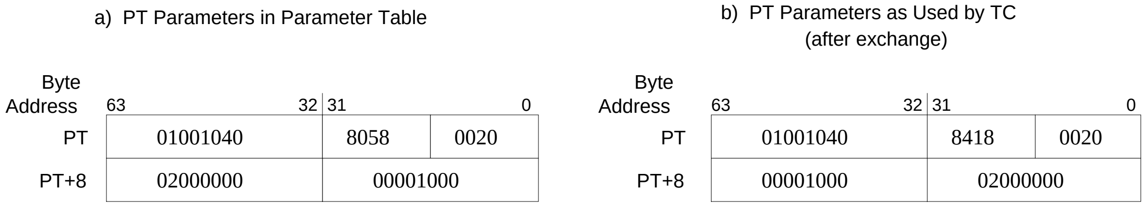
Table 5–3. Exchange src/dst PT Options Bit Swap

Source	Bit Number		Bit Number	Destination
src Update	22	´	25	dst Update

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. When the suspended transfer is restored, the parameters are again swapped as they are loaded by the TC.

Figure 5–4 shows the effects of using the exchange bit on a short-form packet transfer. Part (a) of the example shows a short-form packet transfer that moves 32 bytes from off-chip memory (`0x02000000`) into PP1 data RAM 0 (`0x00001000`) and then updates the src Start Address to `0x02000020`. Because the **X** bit is set in PT Options, the src and dst parameters are exchanged as they are loaded by the TC. Part (b) of the example shows how the packet parameters actually appear to the TC after reversing the src and dst.

Figure 5–4. Exchanging Short-Form src/dst Parameters (Big Endian)



When one of the update modes is specified, then the operation is performed as usual when the packet transfer is completed. If, for example, src update is enabled, the src Start Address in the original packet transfer table is updated even though it was actually used as the dst start address during the packet transfer.

5.4.3 Short-Form Packet Transfer (SF) - Bit(20)

The **SF** bit must be set to 1 for short-form packet transfers. The TC checks this bit when it reads in PT Options to determine if the packet transfer is short or long form. This allows the TC to determine the size of the parameter table and whether to service the packet transfer as a short- or long-form PT.

5.4.4 src Update (SU) - Bit(22)

Setting this bit to 1 causes the src Start Address of the packet transfer to be updated. When a short-form packet transfer successfully completes, the TC will add (subtract) Count to (from) src Start Address and write the result back to the src Start Address location in the parameter table. This allows the transfer to be resubmitted and have the src continue from where it left off. Addition or subtraction of the Count value is based on the state of the **RA** bit. (Subtraction is performed if **RA** is ‘1’).

When the short-form packet transfer is performing a read transfer or split read transfer, the **SU** bit also determines the number of transfers that will occur. In most applications **SU** will be set to ‘1’ for (split) read transfers. This causes the TC to perform a single read transfer and then update the src Start Address with the value in Count. Count is typically programmed in this case to be the length (or half the length) of the VRAM split serial register. Thus when the packet transfer is resubmitted, the next line (or partial line) of the display will be transferred to the serial register. If **SU** is not set, the TC will perform Count number of (split) read transfers (all to the same address) and the src Start Address will not be changed.

5.4.5 Reverse A Addressing (RA) - Bit(23)

The **RA** bit determines the direction in which data is addressed. When **RA** is ‘0’, the src and dst are addressed forwards from their starting addresses. When **RA** is ‘1’, the src and dst are addressed backwards from their starting addresses. Examples of forward and reverse addressing are shown in Figure 5–5 and Figure 5–6 for big and little endian mode respectively.

The **RA** bit also determines how the address calculation for the src or dst update is performed. When **RA** is ‘0’, Count is added to the src and/or dst starting address. When **RA** is ‘1’, Count is subtracted for the src and/or dst starting address.

Figure 5–5. Short-Form Reverse Addressing (Big Endian)

Byte Address	63							0
1000	10	11	12	13	14	15	16	17
1008	20	21	22	23	24	25	26	27
1010	30	31	32	33	34	35	36	37
1018	40	41	42	43	44	45	46	47

Src Memory Area

Byte Address	63							0
3000	00	00	00	00	00	00	00	00
3008	00	00	00	00	00	00	00	00
3010	00	00	00	00	00	00	00	00
3018	00	00	00	00	00	00	00	00

Dst Memory Before Transfer

Byte Address	63							0
3000	00	00	00	00	00	00	00	00
3008	00	00	00	00	00	00	00	00
3010	30	31	32	33	34	35	36	37
3018	40	41	42	43	44	00	00	00

Dst After Forward Transfer

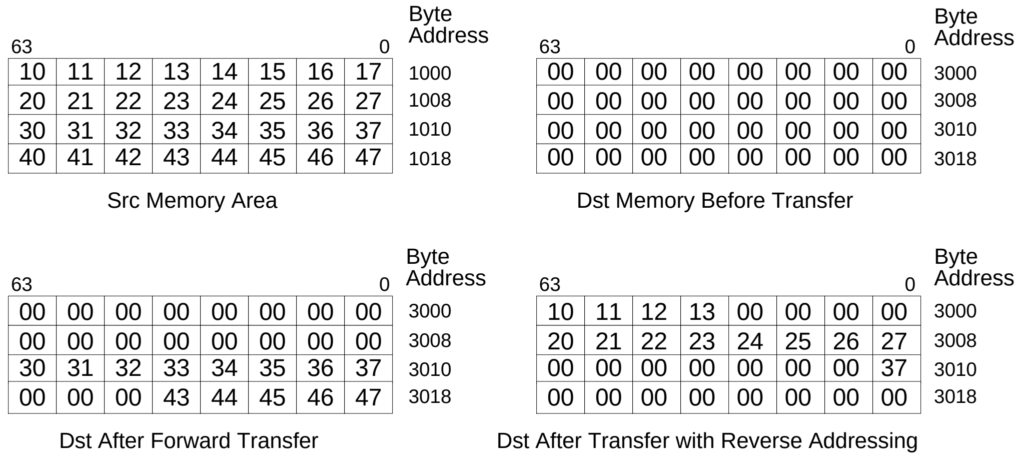
Byte Address	63							0
3000	00	00	00	00	14	15	16	17
3008	20	21	22	23	24	25	26	27
3010	30	00	00	00	00	00	00	00
3018	00	00	00	00	00	00	00	00

Dst After Transfer with Reverse Addressing

Short-form Packet Transfer

Src Start Address:	0xXXXX1010
Dst Start Address:	0xXXXX3010
Count:	0x000D

Figure 5–6. Short-Form Reverse Addressing (Little Endian)



Src Memory Area

63							0	Byte Address
10	11	12	13	14	15	16	17	1000
20	21	22	23	24	25	26	27	1008
30	31	32	33	34	35	36	37	1010
40	41	42	43	44	45	46	47	1018

Dst Memory Before Transfer

63							0	Byte Address
00	00	00	00	00	00	00	00	3000
00	00	00	00	00	00	00	00	3008
00	00	00	00	00	00	00	00	3010
00	00	00	00	00	00	00	00	3018

Dst After Forward Transfer

63							0	Byte Address
00	00	00	00	00	00	00	00	3000
00	00	00	00	00	00	00	00	3008
30	31	32	33	34	35	36	37	3010
00	00	00	43	44	45	46	47	3018

Dst After Transfer with Reverse Addressing

63							0	Byte Address
10	11	12	13	00	00	00	00	3000
20	21	22	23	24	25	26	27	3008
00	00	00	00	00	00	00	37	3010
00	00	00	00	00	00	00	00	3018

Short-form Packet Transfer

Src Start Address:	0xXXXX1010
Dst Start Address:	0xXXXX3010
Count:	0x000D

5.4.6 dst Update (DU) - Bit(25)

Setting this bit to ‘1’ causes the dst Start Address of the packet transfer to be updated. When a short-form packet transfer successfully completes, the TC will add (subtract) Count to (from) dst Start Address and write the result back to the dst Start Address location in the parameter table. This allows the transfer to be resubmitted and have the dst continue from where it left off. Addition or subtraction of the Count value is based on the state of the **RA** bit. (Subtraction is performed if **RA** is ‘1’).

5.4.7 Interrupt when Finished (I) - Bit(28)

Setting this bit to ‘1’ causes an end-of-packet interrupt (**pc** in the MP’s **INTPEN** or **PTEND** in the PP’s **intflg**) to be sent to the processor that initiated the packet transfer as soon as this entry in the linked list has finished. Note that the linked list can contain further entries. Hence, this bit flags the processor when a particular point in the linked list has been reached.

Setting the **I** bit in more than one packet transfer in the same linked list is not recommended. Unless the programmer can ensure that the first interrupt will be serviced before the next one occurs there is no way to determine if only one or if more than one of the interrupts has occurred (there is only one interrupt per linked-list per processor). Note that the last packet transfer in a linked list (the one with its stop bit set) will always generate an end-of-packet interrupt to the requesting processor regardless of the state of the **I** bit. Once again, this interrupt is indistinguishable from an interrupt caused by a set **I** bit in a packet transfer earlier in the linked-list unless that interrupt has already been serviced.

If an error occurs during linked-list processing, the requesting processor will be sent a packet transfer error interrupt (**bp** in the MP's **INTPEN** or **PTERR** in the PP's **intflg**). This occurs independent of the state of the **I** or **S** bits.

5.4.8 Packet Transfer Status(PTS) - Bit(30:29)

These two bits reflect the state of a packet transfer request. They must always be set to '0' when a processor submits a request. If a packet transfer within a linked list is suspended, the TC sets the appropriate **PTS** bits in the PT Options field that it saves to the suspended packet transfer parameters area in the processor's parameter RAM. This is done to distinguish the suspended packet transfer (which has more parameters than a new PT) from a new packet transfer. When either of the status bits is '1', the TC recognizes that it is loading a suspended packet transfer's parameters. Table 5–4 shows the encoding of the **PTS** bits.

Table 5–4. Packet Transfer Status (PTS) Coding

Bit		
30	**29**	**Packet Transfer Status**
0	0	Not suspended (new)
0	1	Suspended but not faulted
1	0	Suspended, fault on src
1	1	Suspended, fault on dst

The TC writes a value '01' to these bits if the packet transfer is suspended due to any of the following conditions:

- ❏ The TC receives a higher priority packet transfer request.
- ❏ The packet transfer times out (**PTMAX** exceeded)
- ❏ The processor that initiated the transfer sends a request that the transfer be suspended.
- ❏ An error condition occurs.

Bit 30 indicates that the PT was suspended because a fault occurred. Bit 29 indicates if the fault occurred on the src or dst. This information is required by the MP in order to resolve the fault condition. However, this is irrelevant to the TC itself when it loads the faulted PT parameters since the suspension and restoration process is the same for all types of suspended transfers.

If the **X** bit is set in the suspended packet transfer's PT Options field, then the value of bit 29 for a faulted transfer is reversed. (In other words, '10' in PTS indicates a fault on dstsrc, and '11' indicates a fault on src.) Therefore MP software should examine both bits 29 and 19 of the PT Options field when determining the faulted address location.

If a fault does occur during a PT, bits 29 and 30 and the appropriate bit in the **FLTSTS** register are set (see Section 1.3.4, *The FLTSTS Register*), and a fault interrupt is sent to the MP (the **mf** bit in the **INTPEN** register).

If the request that faulted was from a PP, the PP will be unaware that the fault occurred. The MP must resolve the condition that caused the fault or reset the PP. When the corresponding fault flag in the **FLTSTS** register is cleared, the PT is automatically resubmitted.

More information on the packet transfer suspend mechanism, see Section 9.5, *Packet Transfer Suspension.*

5.4.9 Stop (S) - Bit(31)

This bit, when set, indicates that the current packet transfer is the last in the linked list. When the TC encounters a packet transfer with the stop bit set, it completes the transfer and then terminates the linked list. Before termination, however, the Next Entry Address field is copied into the Linked List Start Address in the requesting processor's parameter RAM and any specified parameter updates are performed. If the linked list is re-enabled, execution begins at the next entry in the linked list. Thus the **S** bit can also be used to break linked lists at desired locations.

Chapter 6

Long-Form Packet Transfers

Long-form packet transfers allow the transfer of multiple, complex blocks of data within TMS320C82 memory space. This chapter describes the various long-form packet transfer formats and their operation. It also explains how to set up long-form PTs and select the desired options.

Topics in this chapter include:

Topic **Page**

6.1 Long-Form Packet Transfer Overview

Long-form packet transfers can move multiple, non-contiguous blocks of data from a src memory area to a dst memory area. Long-form PTs require a larger parameter table than short-form PTs because more parameters are needed to describe the complex data patterns. Thus long-form PTs can take longer to program and take up more on-chip memory space than short-form PTs. They also require more TC overhead to read in and save their parameter tables. The advantages of long-form PTs are that they can transfer complex patterns of data in a single packet transfer and that they can reduce processor overhead by transforming the order of data as it is moved on or off chip.

Long-form 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 long form packet transfer's source transfer and destination transfer are independent of one another. This allows the packet data to be read in one format, and written in an entirely different format. This flexibility enables a number of spreading or merging functions to be performed on the data as it is transferred.

Long-form packet transfers make use of two basic transfer formats:

❑ Dimensioned Transfers

❑ Guided Transfers

These transfer formats determine either how data is read or how data is written, depending on whether the particular format is being used as a src transfer or a dst transfer. Different formats can be used for the src and dst transfer.

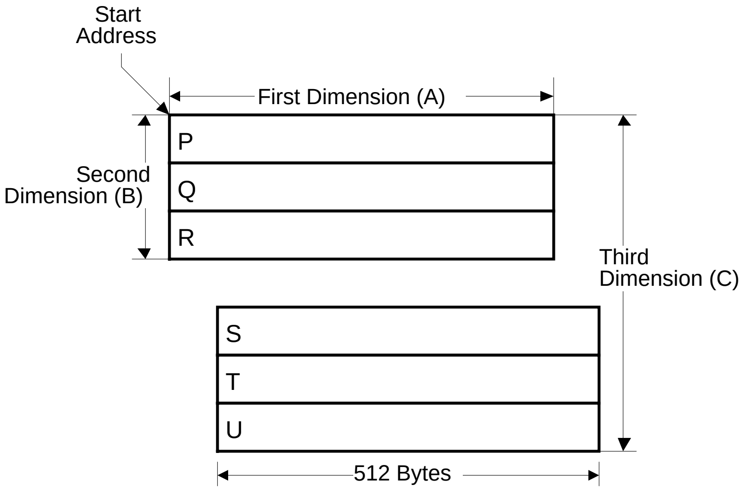
6.2 Dimensioned Transfers

Dimensioned transfers are the simplest type of transfer and are also the most constrained. Dimensioned transfers describe src or dst data consisting of a number of lines of contiguous bytes in multiple regions. The transfer addressing mechanism allows up to three dimensions to be specified. This enables the transfer of multiple two-dimensional patches by a single packet transfer.

The first dimension (A) consists of contiguous bytes that form a line. The second dimension (B) is a collection of lines that form a patch. The third dimension (C) is a collection of patches which form the transfer.

Figure 6–1 shows an example of how a dimensioned transfer might access src or dst memory. It shows a dimensioned transfer consisting of two patches of three lines. The lines are made up of 512 contiguous bytes.

Figure 6–1. Dimension Transfer Example



The transfer shown in might be used if two PPs were going to perform a 3x3 convolution with each PP working on a different patch. The first patch (PQR) represents data to be placed in PP0’s data RAM and the second patch (STU) represents data to be placed in PP1’s data RAM. In this case, the transfer represents the src transfer because it describes how data is to be read from source memory. The associated dst transfer would describe how to place the data into the two PP data RAMs.

The data packet for a dimensioned transfer is specified in terms of the following parameters:

❑ A Count: The number of contiguous bytes that form a line (First Dimension)

❑ B Count: The number of line steps that form a patch. (Second Dimension = number of lines minus one)

❑ C Count: The number of patch steps that 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.

❑ B Count: The linear pitch of the third dimension.

The example shown in Figure 6–1 would have the following transfer parameters:

A Count:	512
B Count:	2
C Count:	1
Start Address:	Address of byte *P*
B Pitch:	Difference in addresses of bytes *P* and *Q*
C Pitch:	Difference in addresses of bytes *P* and *S*

Both src and dst transfers can be defined in the above manner, but the parameters for the two transfers are independent. This means that the shape of the src and dst packets can be quite different.

Not all the dimensions of a dimensioned transfer need to be used. By setting B Count and C Count to ‘0’, a single line of contiguous bytes is transferred. (If both src and dst are configured this way, then this is much like a short-form packet transfer.) If C Count is ‘0’, then a single two-dimensional patch is transferred.

An A Count of ‘0’ results in no data being transferred. However, the packet is still loaded and necessary address calculations are still performed. This will incur TC overhead. Specifying a ‘0’ A Count may also result in a PT error if it causes the total dst byte count to exceed the total src byte count. (See Section 9.4, *Packet Transfer Errors*)

6.3 Guided Transfers

Guided transfers are those in which the data addressing is guided from an on-chip memory table rather than calculated solely from the PT parameter table values. This operation is more complicated than that of the dimensioned transfer but provides more flexibility in addressing data. There are two classes of guided transfers:

- ❑ Fixed-Patch Guided Transfers
- ❑ Variable-Patch Guided Transfers

Fixed-patch guided transfers have the size of their first (A) and second (B) dimensions *fixed* within the parameter table, but the third dimension (C) is specified by an on-chip guide table. In variable-patch guided transfers, the guide table determines the size of A and B dimensions as well so the patch size is *variable* from patch to patch within the packet.

In both classes of guided transfers, the two-dimensional patch is addressed similar to that of the dimensioned transfer. Each guide table entry can thus specify movement of an single byte, a single line, or a two-dimensional patch according to the programming of the A and B dimension parameters. This allows programming of a number of irregular operations such as those encountered in line drawing or processing of look-up table data.

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

6.3.1 Guide Tables

A guide table is a table of either 32-bit or 64-bit entries used to describe individual patches within a transfer. The table must be located in on-chip memory (data or parameter RAM). 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 which it contains are specified within the packet transfer parameter table.

Each guide table entry contains information corresponding to a single two-dimensional patch. Thus the number of guide table entries determines how many patches will be transferred. As the TC services the packet transfer, it fetches each guide table entry, as needed, to process the next patch in the transfer. Additional information about guide table entries is found in each of the guide table description sections.

6.3.2 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 LSBs of the starting address must be 0). Each guide table entry contains information used to calculate the third dimension of the transfer. Fixed-patch transfers have three addressing modes:

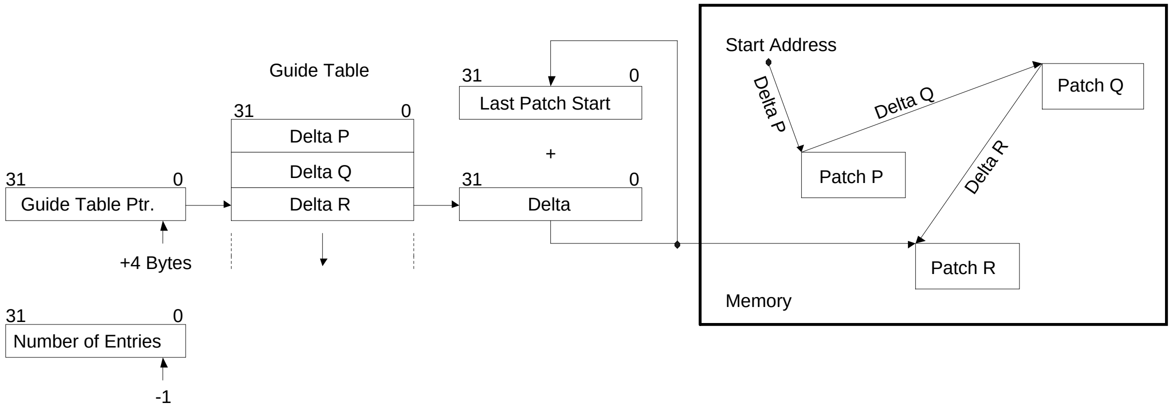
- ❏ Delta-Guided
- ❏ Offset-Guided
- ❏ Offset-Guided Look-Up Table (LUT)

6.3.2.1 Fixed-Patch, Delta Guided Transfers

For fixed-patch delta-guided transfers, the guide table contains 32-bit “delta” values. These values are added to the starting address of the previous two-dimensional patch to form the starting address of the current patch. The patch size is the same for each patch and is defined by the A Count and B Count fields in the parameter table.

An example of a fixed-patch delta-guided transfer is shown in Figure 6–2. The guide table value Delta P is added to the src or dst Start Address (specified in the parameter table) to form the starting address of the first patch, “P”. Delta Q is then added to the starting address of “Patch P” to form the start address of “Patch Q”, and so on.

Figure 6–2. Fixed-Patch Delta-Guided Transfer Mechanism



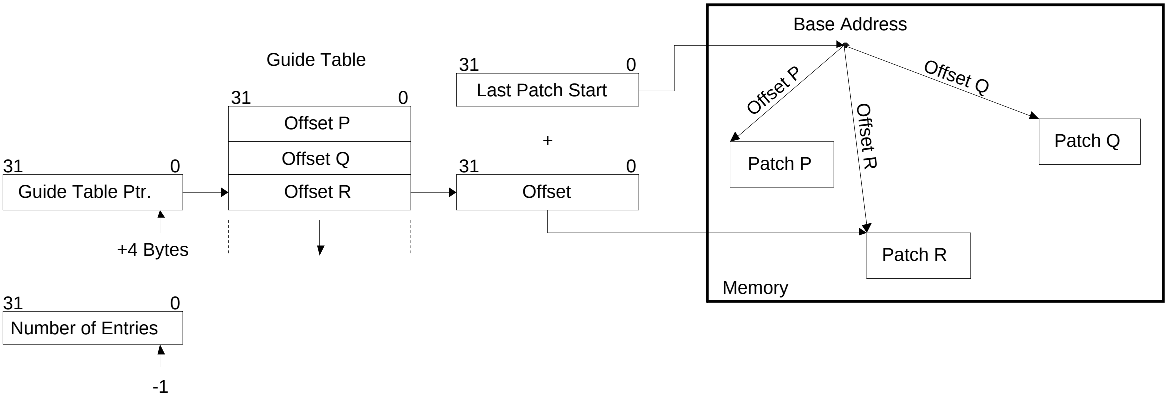
As Figure 6–2 shows, an internal guide table pointer is used to keep track of the address of the current entry. This pointer is loaded with the Guide Table Pointer value from the parameter table to point to the first guide table entry. It is then incremented by 4 bytes after each patch to point to the next guide table entry. An entry counter is loaded with the Number of Entries field specified in the parameter table. This counter is then decremented by 1 after each patch. The transfer is complete when the counter reaches 0.

6.3.2.2 Fixed-Patch Offset-Guided Transfers

Fixed-patch offset guided transfers use a guide table that contains 32-bit "offset" values. The offset values are added to a base address to form the starting address of the current patch. The patch size is the same for each patch and is defined by the A Count and B Count fields in the parameter table.

An example of a fixed-patch offset-guided transfer is shown in Figure 6–3. The first guide table entry, Offset P, is added to the src or dst Base Address (specified in the parameter table) to form the starting address of the first patch, "P". Offset Q is then added to the base address to form the starting address of "Patch Q", and so on. Note that if the Base Address is `0x00000000`, then the entries in the guide table specify absolute addresses rather than relative addresses.

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



As Figure 6–3 shows, an internal guide table pointer is used to keep track of the address of the current entry. This pointer is loaded with the Guide Table Pointer value from the parameter table to point to the first guide table entry. It is then incremented by 4 bytes after each patch to point to the next guide table entry. An entry counter is loaded with the Number of Entries field specified in the parameter table. This counter is then decremented by 1 after each patch. The transfer is complete when the counter reaches 0.

6.3.2.3 Fixed-Patch Offset-Guided LUT Transfers

The guide table for fixed-patch offset-guided look-up table (LUT) transfers consists of 32-bit “offset” values. These values are left-shifted (zero-filled) by zero, one, two, or three bits and then added to a base address to form the starting address of the current patch. This allows the transfer to perform table look-ups independent of the data size.

The fixed-patch offset-guided LUT format can only be specified for src transfers.

The offset shift amount is indicated by the position of the leftmost 1 in bits 0-3 of the packet transfer parameter table’s (src) A Count field. A “1” in bit 3 indicates a left shift of three places; in bit 2, two places, in bit 1, one place, and in bit 0, no left shift. The left shift thus supports look up tables of 8-, 16-, 32-, and 64-bit data.

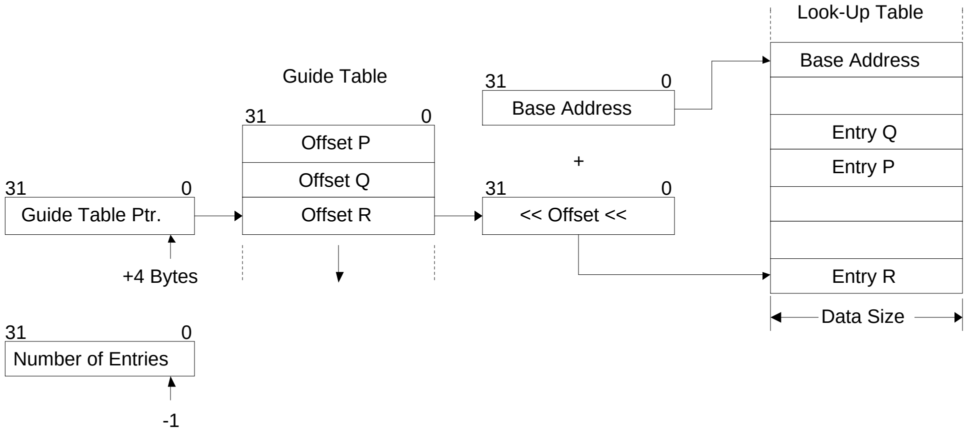
Table 6–1. LUT Transfer A Count Format

A Count Bits						**LUT**
15:4	**3**	**2**	**1**	**0**	**Left Shift**	**Data Size (Bits)**
0x000	0	0	0	1	0	8
0x000	0	0	1	0	1	16
0x000	0	1	0	0	2	32
0x000	1	0	0	0	3	64

As Table 6–1 shows, an LUT transfer must have its A Count field programmed to one, two, four, or eight bytes. Any other value will result in an error. A B Count of 0 is normally used so that each guide table entry looks up a single table entry, however, non-zero B Counts are supported.

An example of a fixed-patch offset-guided LUT transfer is shown in Figure 6–4. The first guide table entry, Offset P, is left shifted (by the A Count value) and added to the src or dst Base Address (specified in the parameter table) to form the address of the first entry, “P”. Offset Q is then shifted and added to the base address to form the address of “Entry Q”, and so on. The shifting of the offset value occurs as it is being loaded from the guide table.

Figure 6–4. Fixed-Patch Offset-Guided LUT Transfer Mechanism



As with the fixed-patch offset-guided transfer, an internal guide table pointer tracks the current position in the guide table and a entry counter tracks the number of guide table entries left to perform.

6.3.3 Variable-Patch Guided Transfers

Variable-patch guided transfers specify the patch size information within the guide table rather than in the packet transfer parameter table. This allows each patch in the transfer to have different dimensions. Variable-patch transfers have two different addressing modes:

❏ Variable-Patch Delta-Guided

❏ Variable-Patch Offset-Guided

The guide table for variable-patch transfers consists of 64-bit doubleword entries. The big-endian format for the guide table is shown in Figure 6–5. The upper half of the doubleword contains the A Count and B Count that specify the patch size. The lower 32-bits contain the values used to calculate the starting address of the patch. Figure 6–6 show the little-endian variable-patch guide table format.

Figure 6–5. Variable-Patch Guide Table Format (Big-Endian)

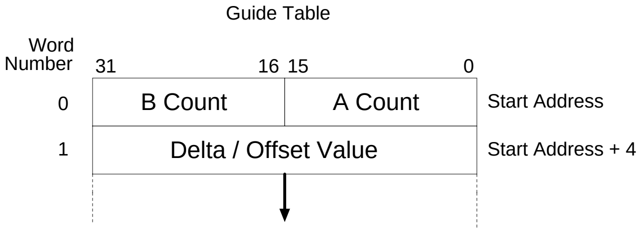
63 48	47 32	31 0
B Count	A Count	Delta / Offset Value

Figure 6–6. Variable-Patch Guide Table Format (Little Endian)

63 – 32	31 – 16	15 – 0
Delta / Offset Value	B Count	A Count

Software can be written to create guide tables independent of the endian format by using 32-bit writes. As Figure 6–7 shows, the A and B Count fields always appear at word 0 address and the offset/delta values appear at word 1 addresses (where word 1 is the address 4 bytes greater than word 0). The TC always reads both words at the same time and adjusts the word order as necessary for the current endian operation.

Figure 6–7. Variable-Patch Guide Table (Endian Independent)



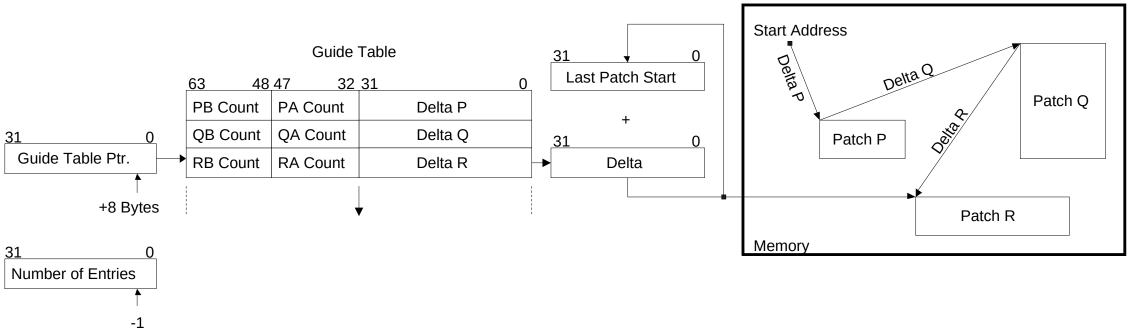
Guide tables for variable-patch transfers must be 64-bit doubleword aligned. (The three LSBs of the address must be 0).

6.3.3.1 Variable-Patch Delta-Guided Transfers

For variable-patch delta-guided transfers, word 1 of the guide table contains 32-bit “delta” values. These values are added to the starting address of the previous two-dimensional patch to form the starting address of the current patch. The patch size is the variable and is defined by the A Count and B Count fields in word 0 of each guide table entry.

An example of a variable-patch delta-guided transfer is shown in Figure 6–8. The guide table value Delta P is added to the src or dst Start Address (specified in the parameter table) to form the starting address of the first patch, “P”. The size of the first dimension (bytes in a line) of “Patch P” is determined by PA Count. PB Count determines the size of the second dimension (number of lines -1). Delta Q is then added to the starting address of “Patch P” to form the starting address of “Patch Q”, and so on.

Figure 6–8. Variable-Patch Delta-Guided Transfer Mechanism (Big -Endian)



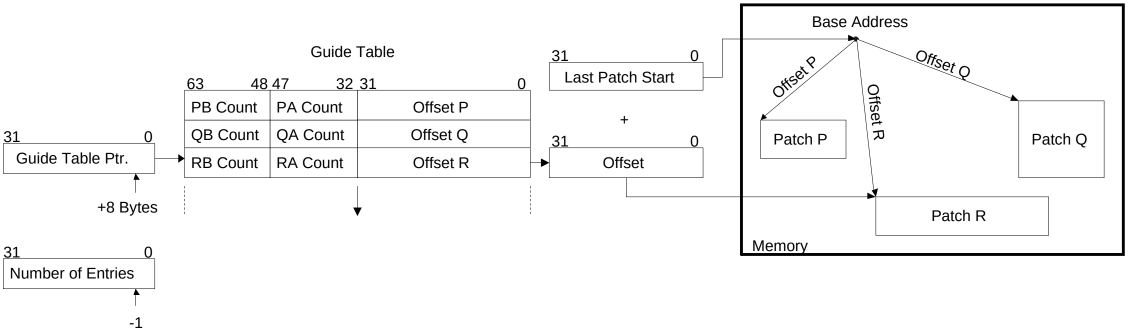
As Figure 6–8 shows, an internal guide table pointer is used to keep track of the address of the current entry. This pointer is loaded with the Guide Table Pointer value from the parameter table to point to the first guide table entry. It is then incremented by 8 bytes after each patch to point to the next guide table entry. An entry counter is loaded with the Number of Entries field specified in the parameter table. This counter is then decremented by 1 after each patch. The transfer is complete when the counter reaches 0.

6.3.3.2 Variable-Patch Offset-Guided Transfers

Variable-patch offset guided transfers use a guide table that contains 32-bit “offset” values in word 1. The offset values are added to a base address to form the starting address of the current patch. The patch size is variable and is defined by the A Count and B Count fields in word 0 of each guide table entry.

An example of a variable-patch offset-guided transfer is shown in Figure 6–9. The first guide table entry, Offset P, is added to the src or dst Base Address (specified in the parameter table) to form the starting address of the first patch, “P”. The PA Count and PB Count fields in the guide table entry determine the size of “Patch P”. Offset Q is then added to the base address to form the starting address of “Patch Q”, and so on. Note that if the Base Address is `0x000000`, then the entries in the guide table specify absolute addresses rather than relative addresses.

Figure 6–9. Variable-Patch Offset-Guided Transfer Mechanism (Big-Endian)



As Figure 6–9 shows, an internal guide table pointer is used to keep track of the address of the current entry. This pointer is loaded with the Guide Table Pointer value from the parameter table to point to the first guide table entry. It is then incremented by 8 bytes after each patch to point to the next guide table entry. An entry counter is loaded with the Number of Entries field specified in the parameter table. This counter is then decremented by 1 after each patch. The transfer is complete when the counter reaches 0.

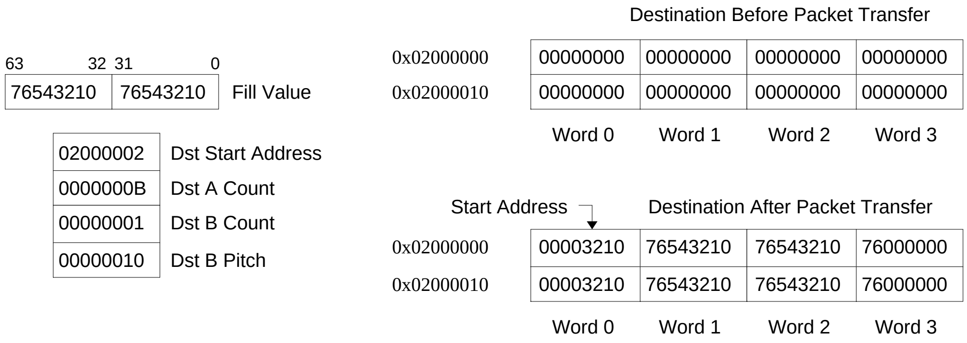
6.3.4 Fill-With Value Transfers

The fill-with-value transfer is a special transfer mode available as a src transfer only. This transfer mode does not actually transfer (read) data from src memory. Instead, it specifies the value of the src data within the packet transfer parameter table. This data value is written to dst memory according to the dst transfer parameters.

Two 32-bit fields (MS Fill Value and LS Fill Value) specify a 64-bit fill value that is used to fill the dst memory. If the desired fill pattern is less than 64-bits then it must be replicated through the MS and LS fill value words. No alignment of the fill pattern is performed, thus the bytes in the dst doubleword will correspond to the bytes of the fill value doubleword.

Figure 6–10 illustrates the use of a fill-with-value transfer with a dimensioned dst transfer. In this big-endian example, the fill pattern is only 32 bits long and is therefore replicated within the fill value doubleword. The dst Start Address is `0x02000002`, which corresponds to the third byte within the doubleword. The dst A Count of 11 (bytes) and dst B Count of 1 results in the destination being filled as shown. Figure 6–11 shows the results of the transfer using the same parameters in little-endian mode.

Figure 6–10. Fill Transfer Example (Big-Endian)



63 ... 32	31 ... 0	
76543210	76543210	Fill Value

02000002	Dst Start Address
0000000B	Dst A Count
00000001	Dst B Count
00000010	Dst B Pitch

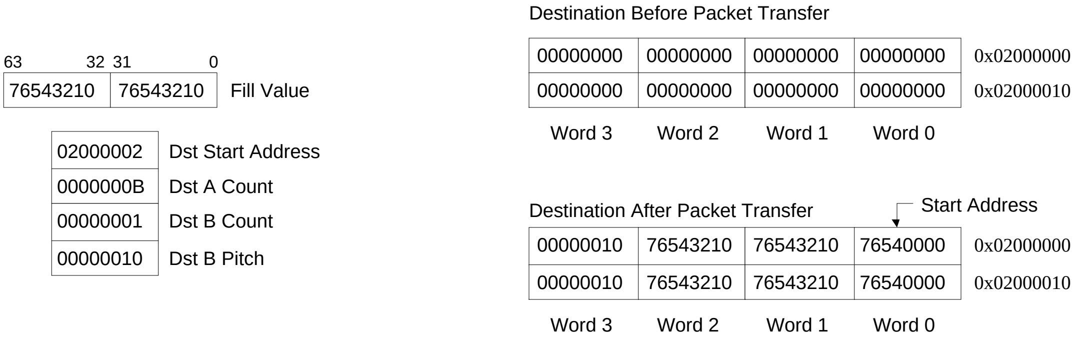
Destination Before Packet Transfer

	Word 0	Word 1	Word 2	Word 3
0x02000000	00000000	00000000	00000000	00000000
0x02000010	00000000	00000000	00000000	00000000

Destination After Packet Transfer

	Word 0	Word 1	Word 2	Word 3
0x02000000	00003210	76543210	76543210	76000000
0x02000010	00003210	76543210	76543210	76000000

Figure 6–11. Fill Transfer Example (Little-Endian)



63 ... 32	31 ... 0	
76543210	76543210	Fill Value

02000002	Dst Start Address
0000000B	Dst A Count
00000001	Dst B Count
00000010	Dst B Pitch

Destination Before Packet Transfer

Word 3	Word 2	Word 1	Word 0	
00000000	00000000	00000000	00000000	0x02000000
00000000	00000000	00000000	00000000	0x02000010

Destination After Packet Transfer

Word 3	Word 2	Word 1	Word 0	
00000010	76543210	76543210	76540000	0x02000000
00000010	76543210	76543210	76540000	0x02000010

6.4 Long-Form Packet Transfer Parameter Tables

Long-form packet transfers are specified using a 64-byte parameter table. The variety of transfer formats available in long-form packet transfers results an a large number of possible src and dst transfer combinations. Each combination of src and dst transfer requires a different parameter table format. Several examples of possible parameter tables are shown in Figure 6–12 to Figure 6–15. A complete listing of all src and dst operating mode combinations and their associated parameter table formats is included in , *Packet Transfer Parameter Tables.*

The parameter table examples show the address of each doubleword relative to the beginning of the table (designated by “PT”). For some transfers modes some fields are unused and may be left unprogrammed. These are designated by “Don’t Care”.

Figure 6–12 through Figure 6–15 include parameters for special access modes. These modes are discussed in Chapter 8, *Special Packet Transfer Access Modes*. They are included in the examples in order to show the full range and location of parameters.

Figure 6–12. Dimensioned src (with Transparency) to Dimensioned dst Parameter Table

a) 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)			
PT+56	Don't Care			

b) 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)				PT+48
Don't Care				PT+56

Figure 6–13. Dimensioned src to Fixed-Patch Guided dst Parameter Table

a) 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 Ptr	
PT+48	Don't Care			
PT+56	Don't Care			

b) 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 Ptr		Src C Pitch		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure 6–14. Dimensioned src to Variable-Patch Guided dst Parameter Table

a) 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 Ptr	
PT+48	Don't Care			
PT+56	Don't Care			

b) 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 Ptr		Src C Pitch		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure 6–15. Fill-With-Value src to Dimensioned dst Parameter Table

a) 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	Don't Care	Dst A Count	Dst B Count
PT+24	Don't Care		Dst C Count	
PT+32	LS Fill Value		Dst B Pitch	
PT+40	MS Fill Value		Dst C Count	
PT+48	Don't Care			
PT+56	Don't Care			

b) 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 A Count	Dst B Count	Don't Care	Don't Care	PT+16
Dst C Count		Don't Care		PT+24
Dst B Pitch		LS Fill Value		PT+32
Dst C Count		MS Fill Value		PT+40
Don't Care				PT+48
Don't Care				PT+56

The TC loads the parameter table by reading 64-bit double words. As the parameter table examples show, the ordering of the parameters is different for big and little endian operation. This allows software to create the parameter table independent of endian mode by using 32-bit writes to the correct word address. When the TC reads a 64-bit doubleword from the parameter table, it adjusts the word order to match its internal operation. Figure 6–16 shows the general long-form packet transfer parameter table format for endian independent programming.

Figure 6–16. General Long-Form Parameter Table Format (Endian-Independent)

<table>
<tr><th>Byte Address</th><th colspan="2">31   0</th><th>Word Number</th></tr>
<tr><td>PT</td><td colspan="2">Next Entry Address</td><td>0</td></tr>
<tr><td>PT+4</td><td colspan="2">PT Options</td><td>1</td></tr>
<tr><td>PT+8</td><td colspan="2">Src Start Address</td><td>0</td></tr>
<tr><td>PT+12</td><td colspan="2">Dst Start Address</td><td>1</td></tr>
<tr><td>PT+16</td><td>Src B Count</td><td>Src A Count</td><td>0</td></tr>
<tr><td>PT+20</td><td>Dst B Count</td><td>Dst A Count</td><td>1</td></tr>
<tr><td>PT+24</td><td colspan="2">Src C Count</td><td>0</td></tr>
<tr><td>PT+28</td><td colspan="2">Dst C Count</td><td>1</td></tr>
<tr><td>PT+32</td><td colspan="2">Src B Pitch</td><td>0</td></tr>
<tr><td>PT+36</td><td colspan="2">Dst B Pitch</td><td>1</td></tr>
<tr><td>PT+40</td><td colspan="2">Src C Pitch</td><td>0</td></tr>
<tr><td>PT+44</td><td colspan="2">Dst C Pitch</td><td>1</td></tr>
<tr><td>*</td><td colspan="2">Transparency / Color Word 0</td><td></td></tr>
<tr><td>*</td><td colspan="2">Transparency / Color Word 1</td><td></td></tr>
<tr><td>PT+56</td><td colspan="2">Don't Care</td><td>0</td></tr>
<tr><td>PT+60</td><td colspan="2">Don't Care</td><td>1</td></tr>
</table>

* 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 + 48	PT + 52	Big Endian
PT + 52	PT + 48	Little Endian

Long-form packet transfer parameter tables must be located in MP or PP parameter RAM or in PP data RAM. The start of each parameter table must be 64-byte aligned (the 6 LSBs of the address must be zero). The tables can reside in any processor's RAM in the sense that the MP can use a PP's parameter RAM to store its packet transfer parameter tables and vice-versa. The processor requesting a long-form packet transfer simply places the appropriate starting address in the Linked List Start Address location in its *own* parameter RAM before submitting its transfer request.

6.5 Long-Form Packet Transfer Parameter Fields

The following sections describe each of the fields in the long-form packet transfer parameter table. The address of each field relative to the start of the parameter table (designated by address “PT”) is shown. The “Transfers” entry lists the transfer operating modes (dimensioned, guided, etc.) for which the field is applicable.

Many of the fields apply to both src and dst transfers. In these cases the field description is given in a singular context with both src and dst table addresses provided. Depending on the transfer type, some fields are unused and may be left unprogrammed.

6.5.1 Next Entry Address

Address: PT
Transfers: All

This is a 32-bit address field which points to the start of the next entry (parameter table) in the packet transfer request’s linked list. If the next entry is a short-form packet transfer, this must point to a 16-byte aligned on-chip address (four LSBs must be zero). If the next entry in the linked list is a long-form packet transfer, this must point to a 64-byte aligned on-chip address. The last parameter table does not require a valid Next Entry Address because the linked-list is terminated by the stop bit (**S**) in the PT Options field.

When a packet transfer completes successfully, the value in Next Entry Address is written to the Linked List Start Address location in the requesting processor’s parameter RAM. This advances the pointer to the next packet transfer in the list. The pointer is also updated after the final packet transfer in the linked list is completed. Thus, if the stop bit is being used to pause linked list execution, the pointer will be pointing to the next transfer on the linked list when the list is re-enabled.

The Next Entry Address in the last parameter table is often programmed to point back to the beginning of the linked list. This allows the processor to request the same linked list of transfers be performed again by simple resubmitting a PT request. (The processor is saved the overhead of reprogramming the Linked List Start Address.)

6.5.2 PT Options

Address: PT + 4
Transfers: All

The 32-bit PT Options field specifies various options concerning how data will be transferred for the current long-form packet transfer. This field is discussed in detail in Section 6.6, *Long-Form Packet Transfer Options Field*.

6.5.3 src / dst Start Address

Address: PT + 8 (src) / PT + 12 (dst)
Transfers: Dimensioned, Delta-Guided

The 32-bit src / dst Start Address field provides the starting byte address of the first src or dst packet in a dimensioned transfer. For dimensioned transfers, this must be an off-chip address or legal on-chip address (i.e. data RAM or parameter RAM). For delta-guided transfers, it is the address to which the delta value from the first guide table entry is added to form the starting address of the first src or dst patch. For delta-guided transfers, any value is allowable as long as the addition of the guide table entry forms an off-chip or legal on-chip address.

6.5.4 src / dst Base Address

Address: PT + 8 (src) / PT + 12 (dst)
Transfers: Offset-Guided

The 32-bit src / dst Base Address is used in place of the start address for offset-guided operating modes. This is the address to which each guide table entry's Offset value is added to form the start address of the current src or dst patch. The addition of the Offset value to the base address must result in an off-chip or legal on-chip address.

6.5.5 src / dst A Count

Address: PT + 16 *bits 15:0* (src) / PT + 20 *bits 15:0* (dst)
Transfers: Dimensioned, Fixed-Patch

The 16-bit src / dst A Count field specifies the number of (contiguous) bytes to be transferred in the first dimension of a dimensioned or fixed-patch transfer. The A Count field can be programmed to zero, however this must not result in a packet transfer in which the total number of bytes specified in the dst transfer exceeds the total number of bytes specified by the src transfer or an error will result. Thus src A Count should only be 0 if dst A Count (or each A Count value in the guide table if the dst is variable-patch) is also 0.

The A Count field is not used for variable-patch transfers. For fill-with-value transfers, the src A Count field is unused.

6.5.6 src / dst B Count

Address: PT + 16 *bits 31:16* (src) / PT + 20 *bits 31:16* (dst)
Transfers: Dimensioned, Fixed-Patch

The 16-bit src / dst B Count specifies the number of steps to occur in the second dimension of a dimensioned or fixed-patch transfer. This is equivalent to the number of lines minus 1. A B Count of 0, therefore, disables the second dimension and results in the transfer of one line per patch.

The B Count field is not used for variable-patch transfers. For fill-with-value transfers, the src B Count field is unused.

6.5.7 src / dst C Count

Address: PT + 24 (src) / PT + 28 (dst)
Transfers: Dimensioned

The 32-bit src / dst C Count specifies the number of patch steps to occur in the third dimension of a dimensioned transfer. This is equivalent to the number of patches minus 1. A C Count of 0, therefore, disables the third dimension and results in the transfer of one patch per packet.

For fill-with-value transfers, the src C Count field is unused.

6.5.8 src / dst Number of Entries

Address: PT + 24 (src) / PT + 28 (dst)
Transfers: Guided

The 32-bit src / dst Number of Entries field replaces the C Count field for all guided transfers. This field specifies the number of entries contained in the packet transfer’s associated guide table. This value cannot be 0 or a packet transfer error will occur.

6.5.9 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 a dimensioned or fixed-patch transfer. The B Pitch is the value that is added to the starting address of the previous line to form the starting address of the current line. Thus it should be programmed with the difference (in bytes) between the starting addresses of any two lines in a patch. B Pitch may be left unprogrammed if its associated B Count field is 0.

6.5.10 src / dst C Pitch

Address: PT + 40 (src) / PT + 44 (dst)
Transfers: Dimensioned

The 32-bit src / dst C Pitch field specifies the pitch of the third dimension of a dimensioned transfer. The C Pitch is the value that is added to the starting address of the previous patch to form the starting address of the current patch. Thus it should be programmed with the difference (in bytes) between the starting addresses of any two patches in a transfer. C Pitch may be left unprogrammed if its associated C Count field is 0.

6.5.11 src / dst Guide Table Pointer

Address: PT + 40 (src) / PT + 44 (dst)
Transfers:Guided

The 32-bit src / dst Guide Table Pointer field replaces the C Pitch for guided transfers. This field is programmed with the starting address of the packet transfer's associated guide table. The address must be aligned to a 32-bit word for fixed-patch guided transfers and aligned to a 64-bit doubleword for variable-patch guided transfers and must point to legal on-chip memory (data or parameter RAM). A packet transfer error will occur if either condition is not met.

6.5.12 LS and MS Fill Value

Address: PT + 32 / PT + 40
Transfers: Fill-with-Value (src only)

These two 32-bit fields are used for fill-with-value src transfers. Together they define a 64-bit fill pattern. They are programmed with the value with which the dst memory area is to be filled. If the desired fill pattern is less than 64-bits then it must be replicated throughout the entire LS and MS Fill Value words.

6.5.13 src Transparency Value

Address: PT + 48
Transfers: Any src Transfer with Transparency Enabled

The 64-bit src Transparency Value is programmed to the value(s) that is (are) to be compared with when transparency access mode is enabled. The field may contain one 64-bit, two 32-bit, four 16-bit or eight 8-bit transparency values depending on the transparency size selected by the **PAM** bits of PT Options. The data bytes about to be written to the off-chip dst are compared to the corresponding bytes in src Transparency Value and the associated byte strobes remain inactive if a match is found. For more details on transparency operation see Section 8.2, *Transparency Modes*

The src Transparency Value doubleword is always used by the TC exactly as it is fetched from the parameter table. The words within the 64-bit field are *not* swapped to adjust for endian-mode. Thus the field appears as follows:

63 32	31 0	
PT + 48	PT + 52	Big Endian
PT + 52	PT + 48	Little Endian

6.5.14 Color Register Value

Address: PT + 48
Transfers: Any src Transfer with Block Write Enabled

The 64-bit Color Register Value is programmed with the value to be loaded into VRAM color registers in preparation for block-write cycles. This is also the value written to the dst if a simulated block write is performed. This field is only used if the block write access mode is selected by the **PAM** bits of PT Options.

The Color Register Value is always used by the TC exactly as it fetches it from the parameter table. The words within the 64-bit field are *not* swapped to adjust for endian-mode. (See above.)

6.5.15 Unused Field

Address: PT + 56
Transfers: All

The last 64-bit doubleword of the long-form packet transfer parameter table is not used and can be left unprogrammed for all transfers.

6.6 Long-Form Packet Transfer Options Field

The PT Options field contains bits to enable various special features during the packet transfer. These features include special access modes, reversing transfer direction, etc. The register also contains information about the status of the packet transfer and determines if the current PT will end the linked list.

The format of the long-form PT Options field is shown in Figure 6–17. Each bit field is described in the following sections. Note that programming PT Options with all ‘0’s results in a dimensioned src to dimension dst packet transfer with no special access or update modes.

Figure 6–17. Long Form PT Options Field

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
S	PTS		I	▨		RDC	RDB	RA	RSC	RSB	SF	X	PAM			▨	STM			▨		SUM		▨	DTM			▨		DUM	

- ❏ DUM - dst Update Mode
- ❏ DTM - dst Transfer Mode
- ❏ SUM - src Update Mode
- ❏ STM - src Transfer Mode
- ❏ PAM - Long-Form PT Access Mode
- ❏ X - Exchange src/dst
- ❏ SF - Short-Form Packet Transfer
- ❏ RSB - Reverse src B Addressing
- ❏ RSC - Reverse src C Addressing
- ❏ RA - Reverse A Addressing
- ❏ RDB - Reverse dst B Addressing
- ❏ RDC - Reverse dst C Addressing
- ❏ I - Interrupt When Finished
- ❏ PTS - Packet Transfer Status
- ❏ S - Stop

6.6.1 dst Update Mode (DUM) - Bits (1:0)

These two bits indicate how the dst Start Address in the packet transfer parameter table is to be updated once the packet transfer has completed. If **DUM** is nonzero, an extra dst address calculation is performed at the end of the packet transfer. The TC then writes the resulting value over the original dst Start Address value in the parameter table. This allows the packet transfer to be resubmitted and have the dst transfer continue from where it left off.

The dst update modes are shown in Table 6–2. Addition or subtraction operation is selected by the state of the **RDB** or **RDC** bits in PT Options.

Table 6–2. dst Update Mode (DUM)

Bit		
1	**0**	**dst Update Operation**
0	0	No update
0	1	Add (subtract) dst 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 PT parameter table.
1	0	Add (subtract) dst 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 PT parameter table.
1	1	Add (subtract) dst 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 PT parameter table; then toggle the reverse dst C addressing (**RDC**) bit in the PT Options field of the PT parameter table.

In normal practice, you would set **DUM** such that one extra step is performed on the largest dimension in use (add B Pitch for 2-dimensional transfers, add C Pitch for 3-dimensional transfers). However, other useful operations can also be performed. **DUM** = ‘10’, for example, can be used with a two dimensional transfer (C Count = 0) to add a specified C Pitch (even though the C Pitch is not used during the actual transfer). When the transfer is resubmitted, the next two dimensional packet is positioned relative to the last.

The **DUM** = ‘11’ setting is especially useful for ping-ponging between two dst memory areas using a two-dimensional patch. The C Pitch represents the difference in address between the two memory areas. Because the **RDC** bit toggles after each packet transfer, the update switches between adding and subtracting the C Pitch. Thus the other memory area is selected each time the packet transfer is submitted.

The **DUM** functionality is intended primarily for use with dimensioned transfers. There are no restrictions on specifying updates with guided transfers but care must be used. The same operation is performed regardless of transfer mode so for guided transfers the dst Guide Table Pointer (which resides in the dst C Pitch location for guided transfers) will be the value used for updates involving C Pitch.

6.6.2 dst Transfer Mode (DTM) - Bits (6:4)

The three **DTM** bits determine which transfer mode will be used for the dst transfer. The five available modes are shown in Table 6–3. Specifying one of the reserved transfer modes will result in a packet transfer error.

Table 6–3. 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

6.6.3 src Update Mode (SUM) - Bits (9:8)

These two bits indicate how the src Start Address in the packet transfer parameter table is to be updated once the packet transfer has completed. If **SUM** is nonzero, an extra src address calculation is performed at the end of the packet transfer. The TC then writes the resulting value over the original src Start Address value in the parameter table. This allows the packet transfer to be resubmitted and have the src transfer continue from where it left off.

The src update modes are shown in Table 6–4. Addition or subtraction operation is selected by the state of the **RSB** or **RSC** bits in PT Options.

Table 6–4. src Update Mode (SUM)

Bit		
9	**8**	**src Update Operation**
0	0	No update
0	1	Add (subtract) src 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 PT parameter table.
1	0	Add (subtract) src 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 PT parameter table.
1	1	Add (subtract) src 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 PT parameter table; then toggle the reverse src C addressing (**RSC**) bit in the PT Options field of the PT parameter table.

In normal practice, you would set **SUM** such that one extra step is performed on the largest dimension in use (add B Pitch for 2-dimensional transfers, add

C Pitch for 3-dimensional transfers). However, other useful operations can also be performed. **SUM** = ‘10’, for example, can be used with a two dimensional transfer (C Count = 0) to add a specified C Pitch (even though the C Pitch is not used during the actual transfer). When the transfer is resubmitted, the next two-dimensional packet is positioned relative to the last.

The **SUM** = ‘11’ setting is especially useful for ping-ponging between two src memory areas using a two-dimensional patch. The C Pitch represents the difference in address between the two memory areas. Because the **RSC** bit toggles after each packet transfer, the update switches between adding and subtracting the C Pitch. Thus the other memory area is selected each time the packet transfer is submitted.

The **SUM** functionality is intended primarily for use with dimensioned transfers. There are no restrictions on specifying updates with guided transfers but care must be used. The same operation is performed regardless of transfer mode so for guided transfers the src Guide Table Pointer (which resides in the src C Pitch location for guided transfers) will be the value used for updates involving C Pitch.

6.6.4 src Transfer Mode (STM) - Bits (14:12)

The three **STM** bits determine which transfer mode will be used for the src transfer. The seven available modes are shown in Table 6–5. Specifying the reserved transfer mode will result in a packet transfer error.

Table 6–5. 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

6.6.5 Long-Form PT Access Mode (PAM) - Bits (18:16)

These three bits select the access mode to be used by the TC when performing the packet transfer. The available modes and their encodings are shown in Table 6–6. Specifying a reserved value will result in a packet transfer fault. Each of these special modes is described in Chapter 8, *Special Packet Transfer Access Modes*.

Table 6–6. Long-From PT Access Modes

Bit			
18	**17**	**16**	**Access Mode**
0	0	0	Normal
0	0	1	Peripheral Device Transfer
0	1	0	Block-Write
0	1	1	Reserved
1	0	0	8-bit Source Transparency
1	0	1	16-bit Source Transparency
1	1	0	32-bit Source Transparency
1	1	1	64-bit Source Transparency

- ❏ **PAM = 000** specifies normal access. No special addressing modes are used on the src or dst. Data is transferred from src to dst without alteration.

- ❏ **Pam = 001** selects peripheral device transfer mode. This mode allows another device to read from or write to the TMS320C82 external memory using the TC as the memory controller. A device read of memory is accomplished by programming the src Start Address to the off-chip location to be read and programming dst Start Address to `0x00000000`. Device writes to memory are accomplished by programming the dst Start Address to the off-chip location to be written and src Start Address to `0x00000000`. In either case, the TC drives the memory address and control signals normally (with address generated according to the transfer parameters), but places the data bus in a high impedance state so that the peripheral device can read or write data. For more details, see Section 8.3, *Peripheral Device Transfer Mode*.

- ❏ **PAM = 010** allows the packet transfer to perform block-write operations for fast memory fills. The TC will load VRAM/SGRAM color registers with the color value and then perform VRAM/SGRAM block-write cycles on the off-chip dst memory. In this mode, the src data actually represents block-write address mask bits that specify which locations in the VRAM/SGRAM will be written with the VRAM/SGRAM color register data. The src data is read using normal addressing and written to the dst using block-write cycles. The value loaded into the VRAM/SGRAM color registers is specified in the **Color Register** field of the parameter table. Block-write operation is described in detail in Section 8.1, *Block-Write Modes*.

The block-write address mode is supported for off-chip dst memory only. Attempting a block write to an on-chip dst address causes the packet transfer to suspend with an error. (See Section 9.4, *Packet Transfer Errors*.)

- ❑ **PAM = 1xx** enables “transparency on source” operation. The src and dst transfers (dimensioned or guided) are performed normally. However, before data is written to the dst memory, it is compared to the src Transparency Value in the parameter table. Comparison is performed as eight 8-bit, four 16-bit, two 32-bit, or one 64-bit compare for 8-bit, 16-bit, 32-bit, or 64-bit transparency respectively. If any of the data values match the corresponding src Transparency Value, the TC will disable the corresponding byte strobe(s) so that the values are not written to dst memory. For a complete description of transparency operation, see Section 8.2, *Transparency Modes*.

Transparency addressing modes (**PAM** = ‘1xx’) are supported for off-chip dst memory only. Attempting transparency with an on-chip dst address will cause the packet transfer to suspend with an error (see Section 9.4, *Packet Transfer Errors)*.

6.6.6 Exchange src and dst Parameters (X) - Bit (19)

Setting this bit reverses the direction of a packet transfer without the need to manually swap the src and dst parameters. This is useful for returning processed data to its original location using the same packet transfer that read in the data.

When the **X** bit is set, the transfer controller swaps all the src- and dst-related values when it loads the packet transfer parameters. Table 6–7 shows the word swap that occurs, assuming that PT represents the parameter table’s start address.

Table 6–7. Exchange src/dst Parameter Word Swap (long-Form PT)

Source	Byte Address		Byte Address	Destination
src Start Address	PT + 8	´	PT + 12	dst Start Address
src A/B Count	PT + 16	´	PT + 20	dst A/B Count
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

Note that the Next Entry Address field at “PT” and the PT Options field at “PT + 4” remain in the same location as they are not src/dst related. Also note that “PT + 48” and “PT + 52” words are not swapped. This allows the 64-bit src Transparency / Color Register parameter to retain its value.

In addition to swapping the src and dst parameter fields, the TC also swaps the src and dst related bits within the PT Options field. This is shown in Table 6–8.

Table 6–8. Exchange src/dst PT Options Bit Swap (Long-Form PT)

	Bit		Bit	
Source	**Number**		**Number**	**Destination**
src Update Mode (SUM)	8	´	0	dst Update Mode (DUM)
	9	´	1	
src Transfer Mode (STM)	4	´	12	dst Transfer Mode (DTM)
	5	´	13	
	6	´	14	
src Reverse B Addr. (SRB)	21	´	24	dst Reverse B Addr. (DRB)
src Reverse C Addr. (src)	22	´	25	dst Reverse C Addr. (DRC)

If a bit swap results in an unsupported function or reserved coding, then the packet transfer suspends with an error.

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. When the suspended transfer is restored, the parameters are again swapped as they are loaded by the TC.

Figure 6–18 shows the effects of using the exchange bit on a long-form packet transfer. Part (a) of the example shows a long-form packet transfer that performs a dimensioned src transfer with 8-bit transparency, reversed B addressing, and C Pitch update and a fixed-patch delta-guided dst transfer. Because the **X** bit is set in PT Options, the src and dst parameters are exchanged as they are loaded by the TC. Part (b) of the example shows how the packet parameters actually appear to the TC after reversing the src and dst. This results in a fixed-patch delta-guided src transfer with 8-bit transparency and a dimensioned dst transfer with reversed B addressing and C Pitch update.

Figure 6–18. Exchanging Long-Form src/dst Parameters (Big Endian)

a) PT Parameters in Parameter Table

Byte Address	63	32	31	0
PT	01010340		802C0260	
PT+8	02000F00		02800000	
PT+16	0010	000A	0002	0014
PT+24	00000001		00000004	
PT+32	00000100		00000200	
PT+40	00000000		01010400	
PT+48	8383838383838383			
PT+56	0000000000000000			

b) PT Parameters as Used by TC (after exchange)

Byte Address	63	32	31	0
PT	01010340		810C6002	
PT+8	02800000		02000F00	
PT+16	0002	0014	0010	000A
PT+24	00000004		00000001	
PT+32	00000200		00000100	
PT+40	01010400		00000000	
PT+48	8383838383838383			
PT+56	0000000000000000			

When one of the update modes is specified, then the operation is performed as usual when the packet transfer is completed. If, for example, src update is enabled, the src Start Address in the original packet transfer table is updated even though it was actually used as the dst start address during the packet transfer. Likewise, specifying the “toggle reverse src C addressing” update mode (**SUM** = ‘`11`’) will cause bit 22 (**RSC**) in the original PT Options to be toggled. This will cause dst C addressing to be reversed if the packet transfer is resubmitted.

The **X** bit should only be used in conjuction with transfer modes supported by both the src and dst. The packet transfer will terminate with an error if the **X** bit is set in conjuction with any of the following:

- Fill-with-Value transfer
- Fixed-Patch Offset-Guided LUT transfer
- Block-Write with on-chip src
- Transparency with on-chip src
- Any PT that specifies an unequal number of total bytes in the src and dst

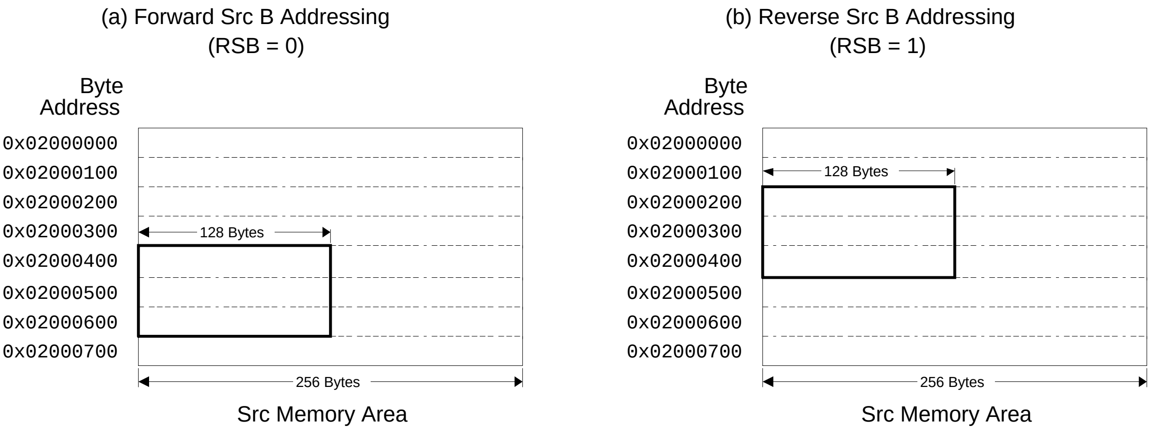
6.6.7 Short-Form Packet Transfer (SF) - Bit (20)

The **SF** bit must be set to ‘`0`’ for long-form packet transfers. The TC checks this bit when it reads in PT Options to determine if the packet transfer is short or long form. This allows the TC to determine the size of the parameter table and whether to service the packet transfer as a short- or long-form PT.

6.6.8 Reverse src B Addressing (RSB) - Bit (21)

Setting **RSB** to ‘1’ causes the second dimension of the src to be addressed backwards. The src B Pitch is subtracted from, rather than added to, the starting address of the previous line to form the starting address of the current line. This is illustrated in Figure 6–19. In this example three lines of 128 bytes each are read from the src starting at address 0x02000400. In (a) the src memory is forward addressed so the line starting address is incremented by src B Pitch. In (b) the src memory is reverse B addressed so the line starting address is decremented by src B Pitch.

Figure 6–19. Reverse src B Addressing Example



Long-Form Packet Transfer

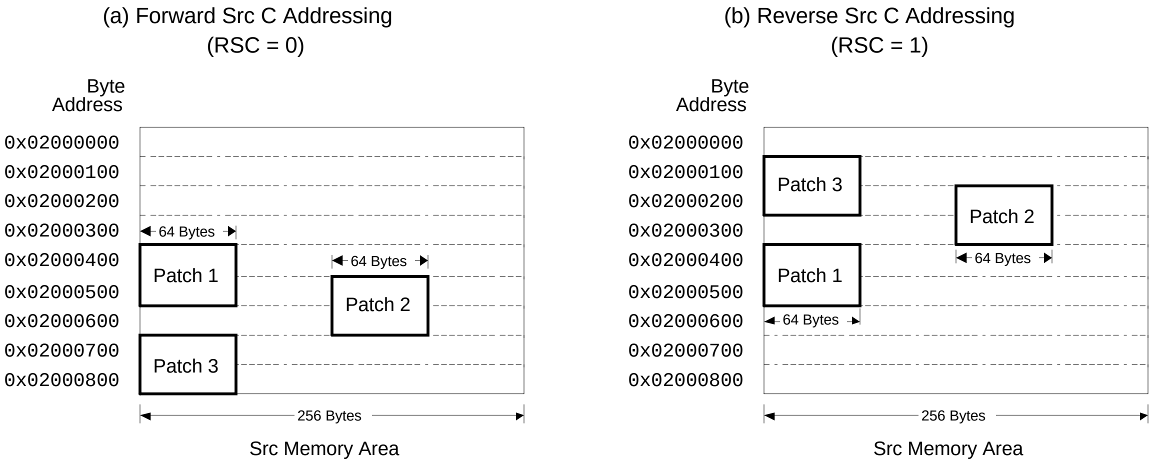
Src Start Address:	0x02000400
Src A Count:	0x0080
Src B Count:	0x0002
Src B Pitch:	0x00000100

RSB also controls source update operation when B Pitch update is selected (**SUM** = ‘01’). If **RSB** is ‘1’, the src B Pitch is subtracted from (rather than added to) the start address of the last line when the update is performed.

6.6.9 Reverse src C Addressing (RSC) - Bit (22)

Setting **RSC** to ‘1’ causes the third dimension of the src to be addressed backwards for dimensioned transfers. The src C Pitch is subtracted from, rather than added to, the starting address of the previous patch to form the starting address of the current patch. This is illustrated in Figure 6–20. In this example, three patches of two lines of 64 bytes each are read from the src, starting at address 0x02000400. In (a) the src memory is forward addressed so the patch starting address is incremented by src C Pitch. In (b) the src memory is reverse C addressed so the patch starting address is decremented by src C Pitch.

Figure 6–20. Reverse src C Addressing



(a) Forward Src C Addressing (RSC = 0)

(b) Reverse Src C Addressing (RSC = 1)

Long-Form Packet Transfer

Src Start Address:	0x02000400	Src B Pitch:	0x00000100
Src A Count:	0x0040	Src C Pitch:	0x00000180
Src B Count:	0x0001		
Src C Count:	0x00000002		

For guided transfers, **RSC** controls how the guide table pointer is modified during the transfer. If **RSC** is ‘1’, the guide table pointer is decremented by 4 or 8 bytes (for fixed or variable patch respectively) to advance to the next entry rather than incremented.

RSC also controls source update operation when C Pitch update is selected (**SUM** = ‘1x’). If **RSC** is ‘1’, the src C Pitch is subtracted from (rather than added to) the start address of the last patch when the update is performed.

6.6.10 Reverse A Addressing (RA) - Bit (23)

The **RA** bit determines the direction in which the first dimension is addressed. When **RA** is ‘0’, the src and dst are addressed forwards from their starting addresses. When **RA** is ‘1’, the src and dst are addressed backwards from their starting addresses. Examples of forward and reverse addressing are shown in Figure 6–21 and Figure 6–22 for big and little endian mode respectively.

Figure 6–21. Long-Form Reverse A Addressing (Big Endian)

Byte Address	63							0
1000	10	11	12	13	14	15	16	17
1008	20	21	22	23	24	25	26	27
1010	30	31	32	33	34	35	36	37
1018	40	41	42	43	44	45	46	47

Src Memory Area

Byte Address	63							0
3000	00	00	00	00	00	00	00	00
3008	00	00	00	00	00	00	00	00
3010	00	00	00	00	00	00	00	00
3018	00	00	00	00	00	00	00	00

Dst Memory Before Transfer

Byte Address	63							0
3000	00	00	00	00	00	00	00	00
3008	00	00	00	00	00	00	00	00
3010	30	31	32	33	34	35	36	37
3018	40	41	42	43	44	00	00	00

Dst After Forward Transfer

Byte Address	63							0
3000	00	00	00	00	14	15	16	17
3008	20	21	22	23	24	25	26	27
3010	30	00	00	00	00	00	00	00
3018	00	00	00	00	00	00	00	00

Dst After Transfer with Reverse Addressing

Long-Form Packet Transfer

PT Options:	0x80800000	Dst Start Address:	0x02003010
Src Start Address:	0x02001010	Dst B/A Count:	0x0000000D
Src B/A Count:	0x0000000D	Dst C Count:	0x00000000
Src C Count:	0x00000000		

Figure 6–22. Long-Form Reverse A Addressing (Little Endian)

63							0	Byte Address
10	11	12	13	14	15	16	17	1000
20	21	22	23	24	25	26	27	1008
30	31	32	33	34	35	36	37	1010
40	41	42	43	44	45	46	47	1018

Src Memory Area

63							0	Byte Address
00	00	00	00	00	00	00	00	3000
00	00	00	00	00	00	00	00	3008
00	00	00	00	00	00	00	00	3010
00	00	00	00	00	00	00	00	3018

Dst Memory Before Transfer

63							0	Byte Address
00	00	00	00	00	00	00	00	3000
00	00	00	00	00	00	00	00	3008
30	31	32	33	34	35	36	37	3010
00	00	00	43	44	45	46	47	3018

Dst After Forward Transfer

63							0	Byte Address
10	11	12	13	00	00	00	00	3000
20	21	22	23	24	25	26	27	3008
00	00	00	00	00	00	00	37	3010
00	00	00	00	00	00	00	00	3018

Dst After Transfer with Reverse Addressing

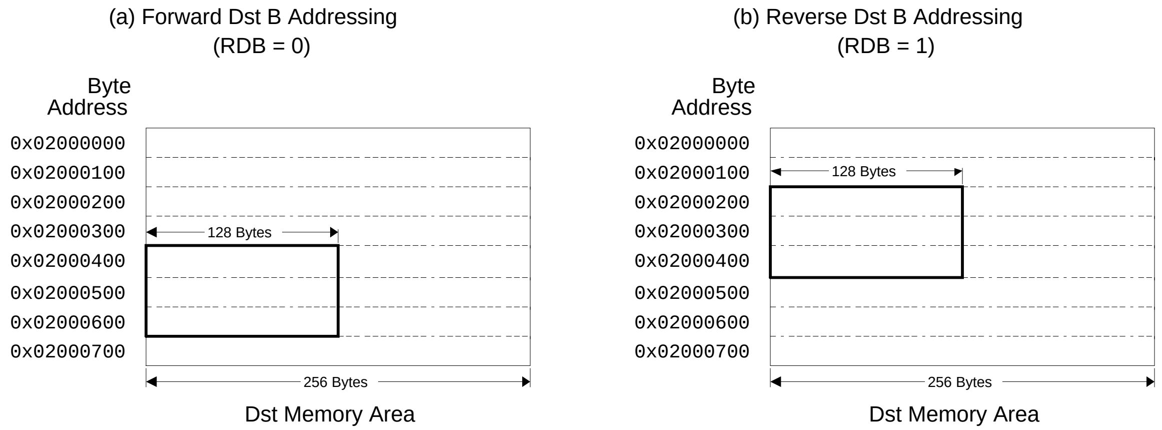
Long-Form Packet Transfer

PT Options:	0x80800000	Dst Start Address:	0x02003010
Src Start Address:	0x02001010	Dst B/A Count:	0x0000000D
Src B/A Count:	0x0000000D	Dst C Count:	0x00000000
Src C Count:	0x00000000		

6.6.11 Reverse dst B Addressing (RDB) - Bit (24)

Setting **RDB** to ‘1’ causes the second dimension of the dst to be addressed backwards. The dst B Pitch is subtracted from, rather than added to, the starting address of the previous line to form the starting address of the current line. This is illustrated in Figure 6–23. In this example three lines of 128 bytes each are written to the dst starting at address 0x02000400. In (a) the dst memory is forward addressed so the line starting address is incremented by dst B Pitch. In (b) the dst memory is reverse B addressed so the line starting address is decremented by dst B Pitch.

Figure 6–23. Reverse dst B Addressing Example



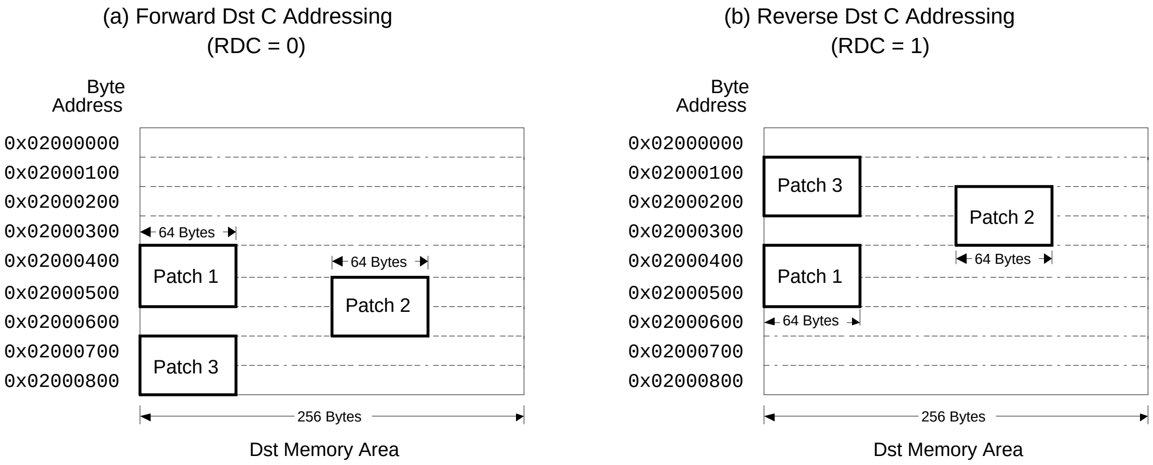
Long-Form Packet Transfer	
Dst Start Address:	0x02000400
Dst A Count:	0x0080
Dst B Count:	0x0002
Dst B Pitch:	0x00000100

RDB also controls destination update operation when B Pitch update is selected (**DUM** = ‘01’). If **RDB** is ‘1’, the dst B Pitch is subtracted from (rather than added to) the start address of the last line when the update is performed.

6.6.12 Reverse dst C Addressing (RDC) - Bit (25)

Setting **RDC** to ‘1’ causes the third dimension of the dst to be addressed backwards for dimensioned transfers. The dst C Pitch is subtracted from, rather than added to, the starting address of the previous patch to form the starting address of the current patch. This is illustrated in Figure 6–24. In this example, three patches of two lines of 64 bytes each are read from the dst starting at address 0x02000400. In (a) the dst memory is forward addressed so the patch starting address is incremented by dst C Pitch. In (b) the dst memory is reverse C addressed so the patch starting address is decremented by dst C Pitch.

Figure 6–24. Reverse dst C Addressing



Long-Form Packet Transfer

Dst Start Address:	0x02000400	Dst B Pitch:	0x00000100
Dst A Count:	0x0040	Dst C Pitch:	0x00000180
Dst B Count:	0x0001		
Dst C Count:	0x00000002		

For guided transfers, **RDC** controls how the guide table pointer is modified during the transfer. If **RDC** is ‘1’, the guide table pointer is decremented by 4 or 8 bytes (for fixed or variable patch respectively) to advance to the next entry rather than incremented.

RDC also controls destination update operation when C Pitch update is selected (**DUM** = ‘1x’). If **RDC** is ‘1’, the dst C Pitch is subtracted from (rather than added to) the start address of the last patch when the update is performed.

6.6.13 Interrupt when Finished (I) - Bit (28)

Setting this bit to ‘1’ causes an end-of-packet interrupt (**pc** in the MP’s **INTPEN** register or **PTEND** in the PP’s **intflg** register) to be sent to the processor that initiated the packet transfer as soon as this entry in the linked list has finished. Note that the linked list can contain further entries. Hence, this bit flags the processor when a particular point in the linked list has been reached.

Setting the **I** bit in more than one packet transfer in the same linked list is not recommended. Unless the programmer can ensure that the first interrupt will be serviced before the next one occurs, there is no way to determine if only one or if more than one of the interrupts has occurred (there is only one interrupt per linked-list per processor). Note that the last packet transfer in a linked list (the one with its stop bit set) will always generate an end-of-packet interrupt to the requesting processor regardless of the state of the **I** bit. Once again, this interrupt is indistinguishable from an interrupt caused by a set **I** bit in a packet transfer earlier in the linked-list unless that interrupt has already been serviced.

If an error occurs during linked-list processing, the requesting processor will be sent a packet transfer error interrupt (**bp** in the MP's **INTPEN** or **PTERR** in the PP's **intflg**). This occurs independent of the state of the **I** or **S** bits.

6.6.14 Packet Transfer Status(PTS) - Bit(30:29)

These two bits reflect the state of a packet transfer request. They must always be set to '0' when a processor submits a request. If a packet transfer within a linked list is suspended, the TC sets the appropriate **PTS** bits in the PT Options field that it saves to the suspended packet transfer parameters area in the processor's parameter RAM. This is done to distinguish the suspended packet transfer (which has more parameters than a new PT) from a new packet transfer. When either of the status bits is '1', the TC recognizes that it is loading a suspended packet transfer's parameters. Table 6–9 shows the encoding of the **PTS** bits.

Table 6–9. Packet Transfer Status (PTS) Coding

Bit		
30	**29**	**Packet Transfer Status**
0	0	Not suspended (new)
0	1	Suspended but not faulted
1	0	Suspended, fault on src
1	1	Suspended, fault on dst

The TC writes a value '01' to these bits if the packet transfer is suspended due to any of the following conditions:

- ❏ The TC receives a higher priority packet transfer request.
- ❏ The packet transfer times out (**PTMAX** exceeded)
- ❏ The processor that initiated the transfer sends a request that the transfer be suspended.
- ❏ An error condition occurs.

Bit 30 indicates that the PT was suspended because a fault occurred. Bit 29 indicates if the fault occurred on the src or dst. This information is required by the MP in order to resolve the fault condition. However, this is irrelevant to the TC itself when it loads the faulted PT parameters since the suspension and restoration process is the same for all types of suspended transfers.

If the **X** bit is set in the suspended packet transfer's PT Options field, then the value of bit 29 for a faulted transfer is reversed. (In other words, '10' in **PTS** indicates a fault on dst, and '11' indicates a fault on src.) Therefore MP software should examine both bits 29 and 19 of the PT Options field while determining the faulted address location.

If a fault does occur during a PT, bits 29 and 30 and the appropriate bit in the **FLTSTS** register are set (see Section 1.3.4, *The FLTSTS Register*), and a fault interrupt is sent to the MP (the **mf** bit in the **INTPEN** register).

If the request that faulted was from a PP, the PP will be unaware that the fault occurred. The MP must resolve the condition that caused the fault or reset the PP. When the corresponding fault flag in the **FLTSTS** register is cleared, the PT is automatically resubmitted.

More information on the packet transfer suspend mechanism, see Section 9.5, *Packet Transfer Suspension.*

6.6.15 Stop (S) - Bit(31)

This bit, when set, indicates that the current packet transfer is the last in the linked list. When the TC encounters a packet transfer with the stop bit set, it completes the transfer and then terminates the linked list. Before termination, however, the Next Entry Address field is copied into the Linked List Start Address in the requesting processor's parameter RAM. If the linked list is re-enabled, execution begins at the next entry in the linked list. Thus the **S** bit can also be used to break linked lists at desired locations.

Chapter 7

Externally Initiated Packet Transfers

This chapter discusses the generation and operation of externally initiated packet transfers (XPTs). XPTs have special features which make them especially suited to performing low-level data movement between the TMS320C82 and peripheral devices.

Topics in this chapter include:

Topic **Page**

7.1 XPT Overview

The XPT mechanism is designed to facilitate the low-level transfer of data between the TMS320C82 and peripheral devices. Through the use of XPTs, a peripheral device can submit a packet transfer request directly to the transfer controller. This eliminates the overhead required to send an external interrupt to the MP and then wait for the MP to perform the packet transfer as part of its interrupt service routine. XPTs are given a higher priority than MP and PP requests. This makes them especially suited to the transfer of high speed data such as filling or emptying video or audio buffers.

XPTs are often performed using the peripheral device transfer mode to move data between the peripheral and ‘C82 system memory. However, XPTs may be performed using any legal packet transfer modes or options available to a processor-requested packet transfer.

In addition to their smaller parameter tables, short-form XPTs have a special service mechanism which allows them to be serviced even more quickly than processor requested short-form packet transfers.

7.2 Requesting XPTs

XPTs cannot be requested unless they are enabled. XPTs are enabled by setting the **X** bit (bit 27) of the MP’s **CONFIG** register to ‘1’. When this bit is ‘0’, the TC will ignore XPT requests. The **X** bit is ‘0’ at reset to prevent the occurrence of erroneous requests.

XPTs are requested via the TMS320C82’s XPT[3:0] input pins. These inputs are encoded to provide 15 different XPT requests. The inputs are active low and encoded as shown in Table 7–1.

Table 7–1. XPT Request Codes

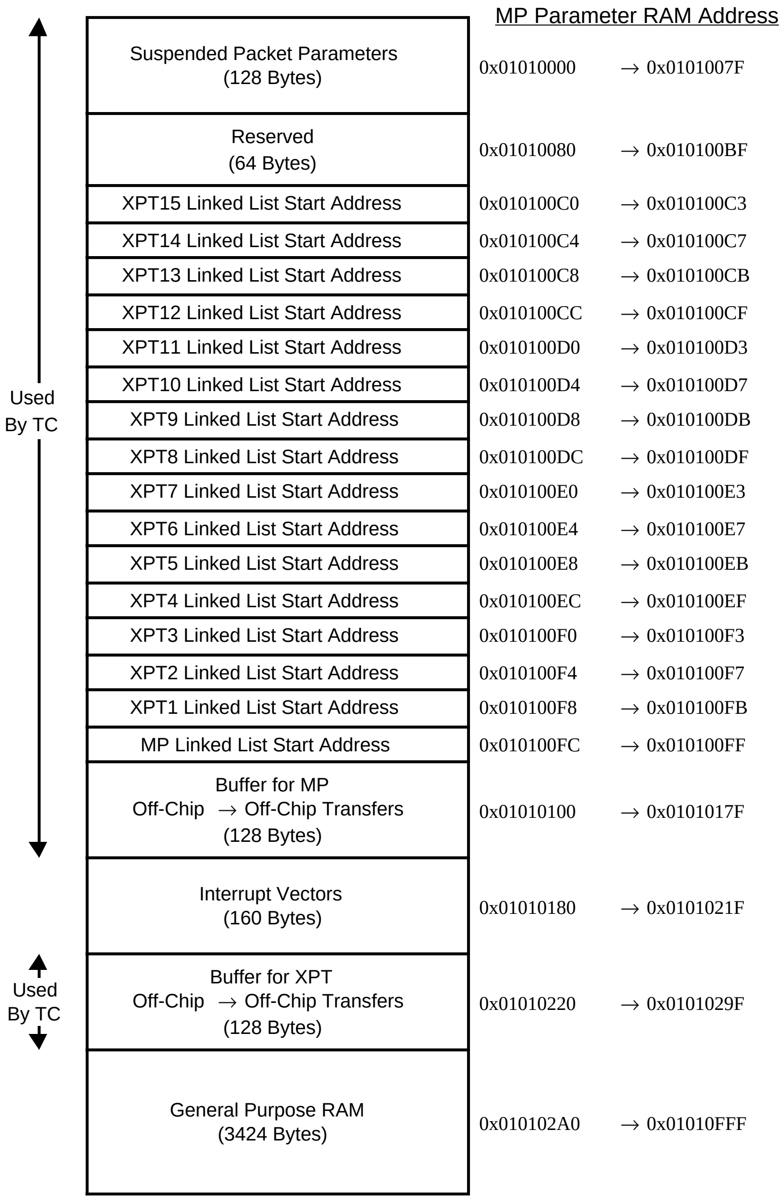
$\overline{\text{XPT3}}$	$\overline{\text{XPT2}}$	$\overline{\text{XPT1}}$	$\overline{\text{XPT0}}$	**Active Request**
1	1	1	1	None
1	1	1	0	XPT1 Request
1	1	0	1	XPT2 Request
1	1	0	0	XPT3 Request
1	0	1	1	XPT4 Request
1	0	1	0	XPT5 Request
1	0	0	1	XPT6 Request
1	0	0	0	XPT7 Request
0	1	1	1	XPT8 Request
0	1	1	0	XPT9 Request
0	1	0	1	XPT10 Request
0	1	0	0	XPT11 Request
0	0	1	1	XPT12 Request
0	0	1	0	XPT13 Request
0	0	0	1	XPT14 Request
0	0	0	0	XPT15 Request

The XPT inputs are internally synchronized and passed to the TC. An XPT request should remain valid until acknowledged by the access type code output at row-time on AD[39:32]. An input code can be changed to indicate another request, but the TC services the first code that is successfully synchronized. Therefore, the access type code on AD[39:32] should always be verified to determine which XPT request the TC is servicing.

7.3 XPT Parameter RAM Usage

The MP contains reserved locations within its parameter RAM which serve as the Linked List Start Address locations for XPTs. Each XPT has its own dedicated Linked List Start Address, allowing all 15 XPTs to be set up and ready for request at any time. A 128-byte area in the MP parameter RAM also is reserved for use as an off-chip to off-chip buffer for XPTs with off-chip src and dst. Only one buffer is required because only a single XPT can be active at any one time. Figure 7–1 shows XPT usage of the MP parameter RAM.

Figure 7–1. XPT MP Parameter RAM Usage



	MP Parameter RAM Address	
Suspended Packet Parameters (128 Bytes)	0x01010000	→ 0x0101007F
Reserved (64 Bytes)	0x01010080	→ 0x010100BF
XPT15 Linked List Start Address	0x010100C0	→ 0x010100C3
XPT14 Linked List Start Address	0x010100C4	→ 0x010100C7
XPT13 Linked List Start Address	0x010100C8	→ 0x010100CB
XPT12 Linked List Start Address	0x010100CC	→ 0x010100CF
XPT11 Linked List Start Address	0x010100D0	→ 0x010100D3
XPT10 Linked List Start Address	0x010100D4	→ 0x010100D7
XPT9 Linked List Start Address	0x010100D8	→ 0x010100DB
XPT8 Linked List Start Address	0x010100DC	→ 0x010100DF
XPT7 Linked List Start Address	0x010100E0	→ 0x010100E3
XPT6 Linked List Start Address	0x010100E4	→ 0x010100E7
XPT5 Linked List Start Address	0x010100E8	→ 0x010100EB
XPT4 Linked List Start Address	0x010100EC	→ 0x010100EF
XPT3 Linked List Start Address	0x010100F0	→ 0x010100F3
XPT2 Linked List Start Address	0x010100F4	→ 0x010100F7
XPT1 Linked List Start Address	0x010100F8	→ 0x010100FB
MP Linked List Start Address	0x010100FC	→ 0x010100FF
Buffer for MP Off-Chip → Off-Chip Transfers (128 Bytes)	0x01010100	→ 0x0101017F
Interrupt Vectors (160 Bytes)	0x01010180	→ 0x0101021F
Buffer for XPT Off-Chip → Off-Chip Transfers (128 Bytes)	0x01010220	→ 0x0101029F
General Purpose RAM (3424 Bytes)	0x010102A0	→ 0x01010FFF

7.4 XPT Servicing

Once an XPT request has been synchronized, the TC will begin servicing the request as follows:

❑ If a processor requested PT is being serviced, the TC will wait until the **PTMIN** requirement for the transfer has been satisfied.

❑ The TC fetches the starting address of the XPT linked list from the appropriate Linked List Start Address location in MP parameter RAM.

❑ The TC reads the PT Options field from the first parameter table to determine if the XPT is short form or long form.

The mechanism used for transferring the packet data differs between long- and short-form XPTs.

7.4.1 Long-Form XPT Service

If the XPT is a long-form packet transfer, the TC uses the normal packet transfer mechanism to move data. If a processor requested packet transfer was being serviced at the time the XPT occurred, it is suspended and its parameters saved to the requesting processor's packet transfer suspend area. The TC then loads the first XPT parameter table and performs the packet transfers until the linked list is complete.

7.4.2 Short-Form XPT Service

If the XPT is a short-form packet transfer, the TC uses a special transfer mechanism to move the data. Rather than use the normal PT service hardware, the TC uses the cache/DEA service hardware to perform the transfer. By bypassing the PT hardware, the TC avoids the need to suspend any current processor requested PT. Instead, it simply halts the current PT and leaves its parameters loaded. It then loads the short-form XPT parameters into the cache/DEA service registers and transfers data until the linked list is complete.

7.4.3 XPT Status Codes

When the TC begins transferring XPT data, it outputs the appropriate row time access code on the AD[39:32] signals. This code tells which XPT number is being serviced and what type of cycle is being performed. After a peripheral device has generated an XPT request, it should monitor the row time access code to determine when its request is being serviced. The XPT access code indicates to the peripheral that it can remove its request from the XPT pins. (See Section 10.4.2, *Row-time Access Type Codes* for more information on row-time access codes.)

There is no provision for signaling the start of an XPT service if the XPT performs an on-chip to on-chip transfer because no external memory cycles are generated. If using an XPT to generate an on-chip to on-chip transfer, the PT should be linked to the end of another (dummy) packet transfer that generates an external bus access. This will allow the peripheral to determine that the request is being serviced and remove its request.

Once the XPT service begins, the TC will once again begin sampling the XPT[3:0] inputs. The current XPT request must be removed within three clock cycles of the falling edge of the RL output to ensure that a second XPT request does not occur.

7.5 XPT Completion

XPTs, unlike processor requested PTs cannot be suspended. Once the TC has begun servicing an XPT, the XPT linked list will run to completion. Because of their high priority, XPT transfers can only be interrupted by host requests, or urgent refresh requests. Once these requests have been serviced, the TC resumes the XPT transfer.

Because XPTs do not suspend, they are not subject to time out due to **PTMIN** and **PTMAX**. There is no provision for preventing an XPT from tying up the TC indefinitely with large transfers or a circular linked list.

7.5.1 XPT Completion Codes

The XPT Complete status code output on STATUS[1:0] at column time indicates the completion of the XPT transfer. It is generated on the last src and dst column access of the XPT packet in which either the **S** (stop) bit or **I** (interrupt when finished) bit of PT Options is set. This completion code lets the requesting peripheral know that the transfer is finished. An XPT does not generate any processor interrupt on completion or as a result of the **I** bit; only the completion code. (For more details on the STATUS[1:0] codes, see Section 10.4.1, *Memory Cycle Status Codes*).

7.5.2 XPT Errors

If an error or fault occurs during an XPT, the transfer controller sets the **XPT** bits (bits 5-8) in the MP's **FLTSTS** register to indicate the XPT in which the fault or error occurred. This also causes the **mf** bit in the MP's **INTPEN** register to be set, generating an interrupt to the MP (if the interrupt is enabled). The TC then terminates the XPT and ignores all further XPT requests until the MP clears the **XPT** bits in **FLTSTS** (by writing 1s to the set bits).

Chapter 8

Special Packet Transfer Access Modes

This chapter provides a details on the operation and use of the special access modes available in packet transfers. The special modes are selected by the programming of the PT Access Mode (**PAM**) bits in the PT Options parameter field.

Topics in this chapter include:

Topic **Page**

8.1 Block-Write Modes

The block-write transfer mode can greatly enhance operations such as rectangle fills, text drawing, and screen clear operations by increasing the rate at which memory is filled with a set pattern. In block-write transfers, the src data is not written to the dst. Instead, the src consists of 1s and 0s which enable or disable the writing of a previously specified *color value* to the dst locations. Block-write mode is available only for long-form packet transfers. It is selected by setting **PAM** = ‘`010`’. The transfer controller supports three types of block-write operation:

- o 8x
- o 4x
- o Simulated

The type of block-write performed is controlled by the transfer controller. This allows the programmer to use block-write operations without regard to what type of block-write the addressed memory supports. Block-write is supported only for 8-bit color values (a 1-to-8 expand) and 64-bit wide off-chip memory. dst Start Addresses for block-write packet transfers must be off-chip and 64-byte aligned or a packet transfer error will result.

8.1.1 Block-Write Type Selection

The type of block-write which the TC will perform is specified for each bank of memory by the **BW(1:0)** value in the cycle configuration cache entry for the bank. If a block-write packet transfer is generated to the bank, the TC performs the cycles needed by the memory device for the block-write to occur. Table 8–1 shows the block-write selection. Memory devices with block-write hardware (such as VRAMs and SGRAMs) should specify the 4x or 8x mode as appropriate for the device. Memory banks without block-write hardware should specify simulated mode.

Table 8–1. Block-Write Selection

BW(1:0)	Block Write Mode
0 0	Simulated
0 1	Reserved
1 0	4x
1 1	8x

8.1.2 Block-Write Notation

A number of different block-write methods exist due to the variety of VRAM/SGRAM sizes and architectures. The following notation is used to express the block-write that a device supports:

$$\text{column locations per color register} \times \text{color register length} \times \text{number of color registers}$$

For example most 1 Mbyte VRAMs (256K array x 4 bits) support a 4x4x1 block write. Each block write cycle writes the color register to four column locations (the first “4”), the color register is 4 bits in length (the “x4”), and there is one color register in the device (the “x1”).

8.1.3 8x Block-Write

The 8x block-write operation should be specified for devices which can write eight column locations per access with an 8-bit color register (8x8x1 or 8x8x2 for example). During the transfer, each bit of src data is output on one of the data bus (AD[63:0]) signals. Each bit enables or disables the writing of the associated 8-bit color register to one of the 64 column locations accessed by the block-write cycle. Each column location corresponds to 8 bits (a bit in each of 8 memory array planes). Thus, up to 64 bytes of color register data in written in a single 8x block-write cycle.

Memories that support 8x block-write ignore the three least significant column address bits during block-write cycles. Since the block-write data bus must be 64 bits wide, the column address bits represent double-word addresses. Thus block-write cycles will always occur on 64-byte boundaries.

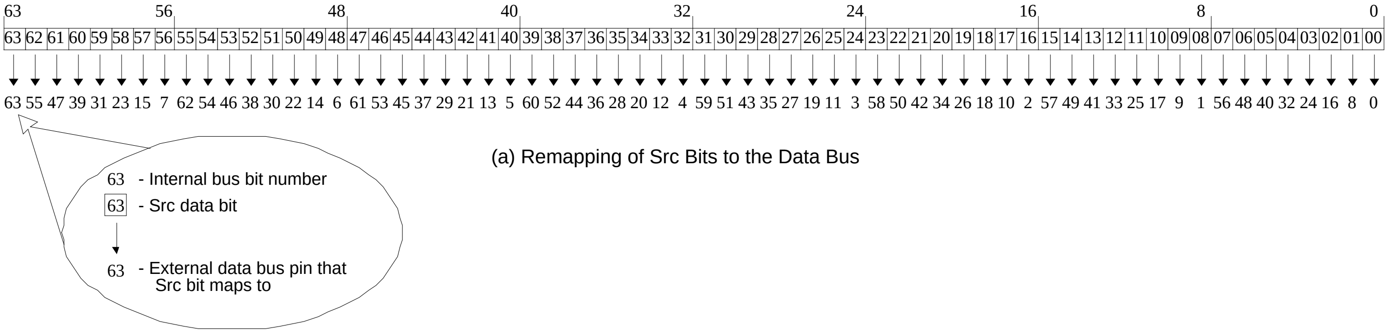
$$2^3 \text{column address} \times \frac{8 \text{ bytes}}{\text{column address}} = 64 \text{ bytes}$$

If the transfer dst address is not 64-byte aligned, the TC will adjust the src data bits and provide missing 0’s for the unwritten dst locations within the 64-byte memory access. The TC then remaps the bits so that they will address the correct column locations within each memory device on the data bus.

Figure 8–1 shows the mapping of src data bits for 8x block-write in little endian mode. (Block-write operation on memory devices is inherently little endian so this mode is most natural.) (a) shows how the aligned src bits are mapped to the external data bus. The first eight src bits control the first eight bytes of the dst. These bytes are actually the least significant column locations in each of the 8-plane memory arrays being accessed. The LS byte is controlled by the D0 input of 8-bit wide devices, the D0 and D8 bits of 16-bit wide devices and the D0, D8, D16 and D24 bits of 32-bit wide devices. Therefore, during block-writes src bits 0-7 are mapped to pins 0, 8, 16, 24, 32, 40, 48, and 56, respectively. These pins are connected to the D0 (and D8, D16, and D24) inputs of the memory devices. The remaining src bits are mapped in a similar manner. (b) shows how the src bits are output on the external data bus (AD[63:0]) after mapping.

Figure 8–2 shows how the src data is mapped for 8x block write in big endian mode. Note that in big endian mode, the LSB of the src data is the leftmost bit and the MSB is the rightmost bit. The bit mapping is the same as for little endian. Thus src bits 0-7 are mapped to pins 63, 55, 47, 39, 31, 23, 15, and 7, respectively as shown in (a). These bits represent the least significant bytes to be written in the memories. Since these bytes are controlled by the D0 (and D8, D16, and D24 if present) inputs of the memory devices, the data bus must be connected to the memories in reverse order. This is shown in Figure 8–3

Figure 8–1. 8x Block Write Output (Little Endian)

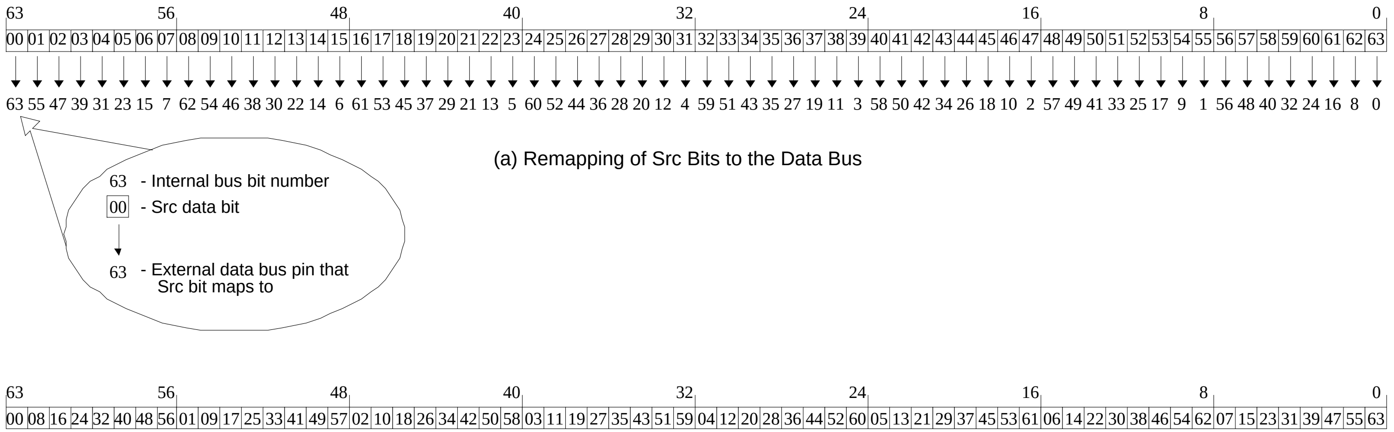


(a) Remapping of Src Bits to the Data Bus

63							56								48								40								32								24								16								8								0
63	55	47	39	31	23	15	07	62	54	46	38	30	22	14	06	61	53	45	37	29	21	13	05	60	52	44	36	28	20	12	04	59	52	43	35	27	19	11	03	58	50	42	34	26	18	10	02	57	49	41	33	25	17	09	01	56	48	40	32	24	16	08	00

(b) Src Bits as They Appear on AD[63:0]

Figure 8–2. 8x Block-Write Output (Big Endian)


63 - Internal bus bit number
00 - Src data bit
63 - External data bus pin that Src bit maps to

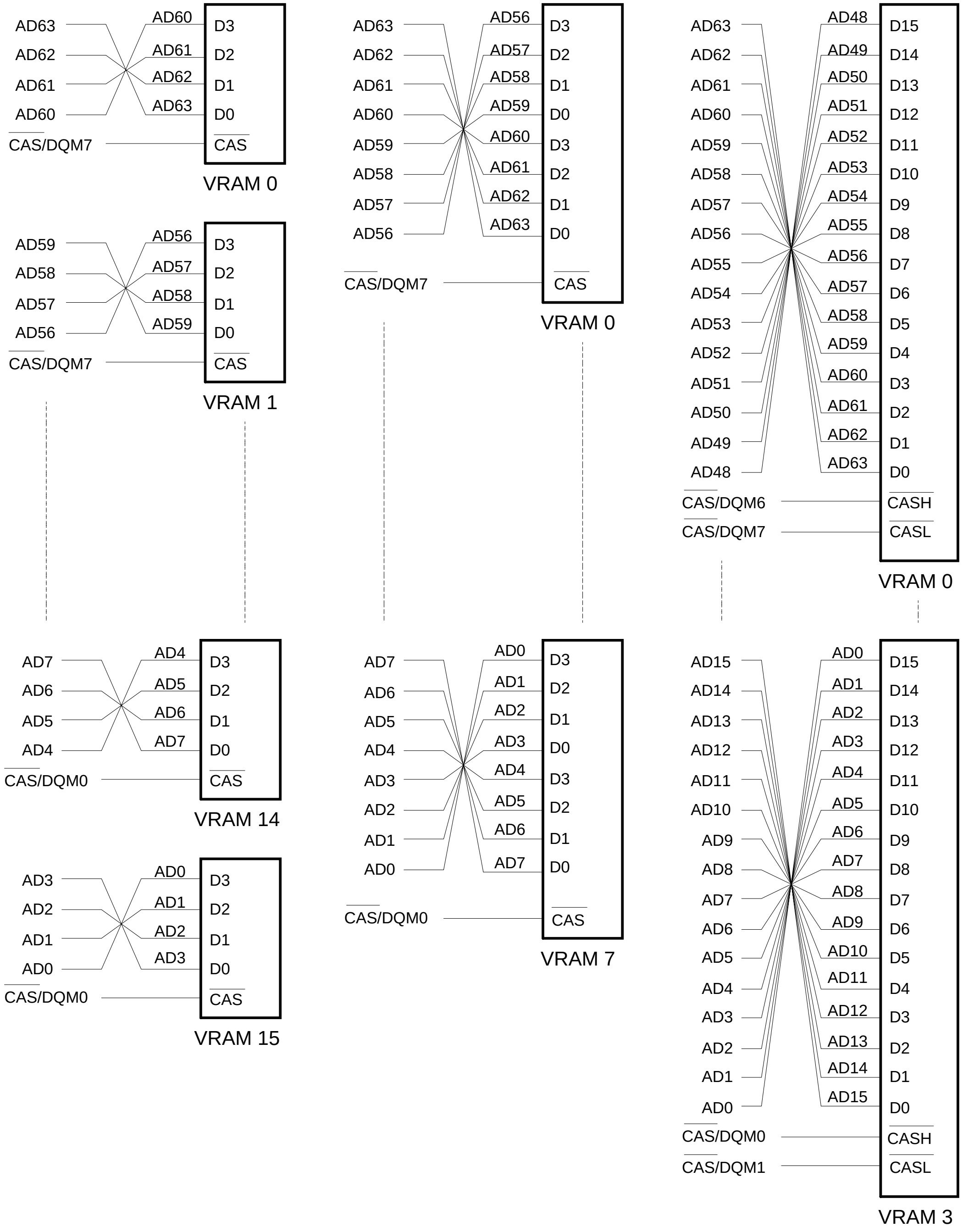
63							56								48								40								32								24								16								8								0
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
63	55	47	39	31	23	15	7	62	54	46	38	30	22	14	6	61	53	45	37	29	21	13	5	60	52	44	36	28	20	12	4	59	51	43	35	27	19	11	3	58	50	42	34	26	18	10	2	57	49	41	33	25	17	9	1	56	48	40	32	24	16	8	0

(a) Remapping of Src Bits to the Data Bus

63							56								48								40								32								24								16								8								0
00	08	16	24	32	40	48	56	01	09	17	25	33	41	49	57	02	10	18	26	34	42	50	58	03	11	19	27	35	43	51	59	04	12	20	28	36	44	52	60	05	13	21	29	37	45	53	61	06	14	22	30	38	46	54	62	07	15	23	31	39	47	55	63

(b) Src Bits as They Appear on AD[63:0]

Figure 8–3. Data Bus Connections for 4-bit, 8-bit, and 16-bit VRAMs in Big Endian Mode



Connecting the data bus in reverse order does not affect normal read and write operations because the data is both written and read in reverse order. However, note that the bits shifted out of the VRAM's serial port will also be in reverse order. This may require changing the order in which the serial data bus is connected to an output device to ensure proper operation.

8.1.4 4x Block-Write

The 4x block-write operation should be specified for devices which can write four column locations per access with a 4- or 8-bit color register (4x4x1, 4x4x2, or 4x8x2 for example). During the transfer, each bit of src data is output on two of the data bus (AD[63:0]) signals. Each bit enables or disables the writing of the associated 4-bit color register to two of the 64 column locations accessed by the block-write cycle. Each column location corresponds to 4 bits (a bit in each of 4 memory array planes). Since only 8-bit color values are supported, each src bit must control the write of two (4-bit) color registers to adjacent nibbles. This allows up to 32 bytes of color register data in written in a single 4x block-write cycle.

Memories that support 4x block-write ignore the two least significant column address bits during block-write cycles. Since the block-write data bus must be 64 bits wide, the column address bits represent double-word addresses. Thus block-write cycles will always occur on 32-byte boundaries.

$$2^2 \text{column address} \times \frac{8 \text{ bytes}}{\text{column address}} = 32 \text{ bytes}$$

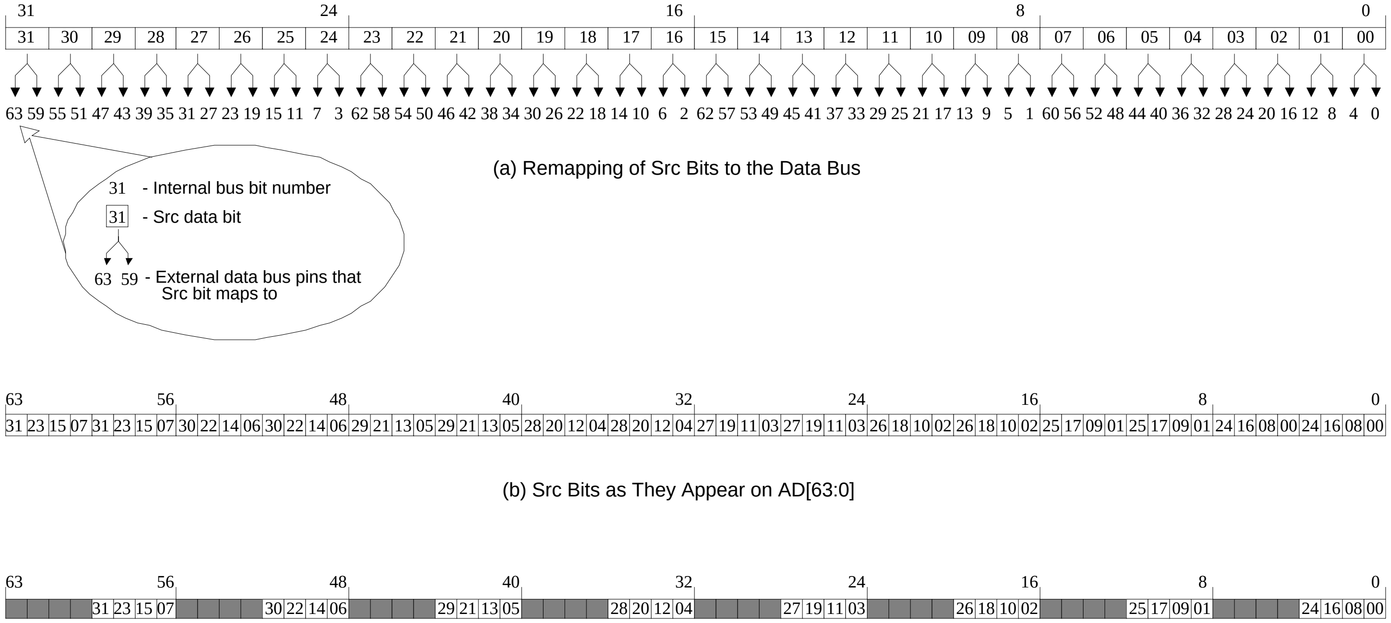
If the transfer dst address is not 32-byte aligned, the TC will adjust the src data bits and provide missing 0's for the unwritten dst locations within the 32-byte memory access. The TC then remaps the bits so that they will address the correct column locations within each memory device on the data bus.

Figure 8–4 shows the mapping of src data bits for 4x block-write in little endian mode. (Block-write operation on memory devices is inherently little endian so this mode is most natural.) (a) shows how the aligned src bits are mapped to the external data bus. The first eight src bits control the first eight bytes of the dst. These bytes are actually the least significant column locations in each 4-plane memory array being accessed. Since each data input on a 4x4 block write causes only 4 bits to be written, each src bit must be mapped to two data pins. The LS byte is controlled by the D0 & D4 inputs of 8-bit wide devices, the D0 & D4, and D8 & D12 bits of 16-bit wide devices and the D0 & D4, D8 & D12, D16 & D20, and D24 & D28 bits of 32-bit wide devices. Therefore, during block-writes src bits 0-7 are mapped to pins 0&4, 8&12, 16&20, 24&28, 32&36, 40&44, 48&52, and 56&60, respectively. These pins are connected to the D0 & D4 (and D8 & D12, D16 & D20, and D24 & D28) inputs of the memory devices. The remaining src bits are mapped in a similar manner. (b) shows how the src bits are output on the external data bus (AD[63:0]) after mapping.

In the case of 4x8xn block write, the memories’ color registers are eight bits wide. Only four of each eight data inputs are used to select which bytes are written. Thus only half of the data bus is used and only 32 bytes can be written by a single block-write cycle.   (c) shows the data bus bits as used for 4x8 block writes. The data output is the same as 4x4 block-write, the memories simply ignore every other nibble.

Figure 8–5 shows how the src data is mapped for 4x block write in big endian mode. Note that in big endian mode, the LSB of the src data is the leftmost bit and the MSB is the rightmost bit. The bit mapping is the same as for little endian. Thus src bits 0-7 are mapped to pins 63&59, 55&51, 47&43, 39&35, 31&27, 23&19, 15&11, and 7&3, respectively as shown in (a). These bits represent the least significant bytes to be written in the memories. Since these bytes are controlled by the D0 & D4 (and D8 & D12, D16 & D20, and D24 & D28 if present) inputs of the memory devices, the data bus must be connected to the memories in reverse order. This is shown in Figure 8–3.

Figure 8–4. 4x Block-Write Output (Little Endian)

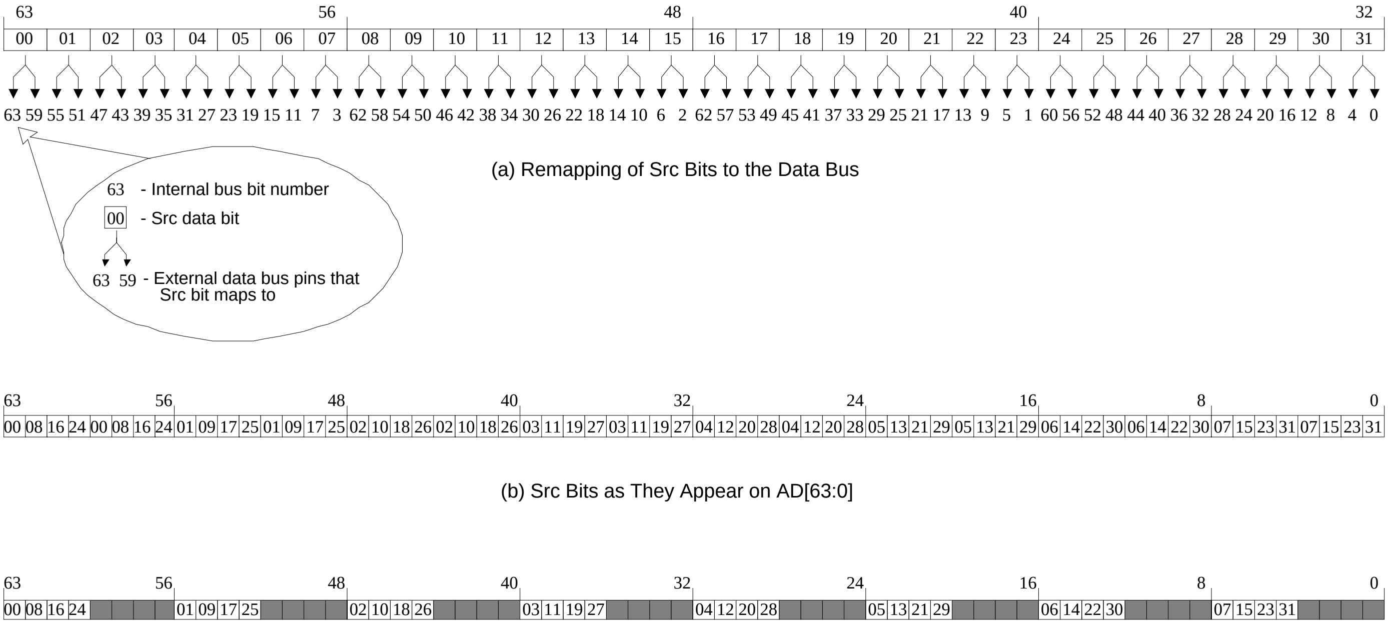


(a) Remapping of Src Bits to the Data Bus

(b) Src Bits as They Appear on AD[63:0]

(c) Bits as Used by 4x8 Block-Write Memories

Figure 8–5. 4x Block-Write Output (Big Endian)



(a) Remapping of Src Bits to the Data Bus

(b) Src Bits as They Appear on AD[63:0]

(c) Bits as Used by 4x8 Block-Write Memories

8.1.5 Simulated Block-Write

The TC provides a simulated block-write capability which allows block-write packet transfers to be performed on memory devices which do not have block write hardware built in. During simulated block-writes, the 64-bit Color Value contained in the packet transfer parameters is output on the external data bus. Each src data bit is used as a byte select to control the CAS/DQM[7:0] byte strobes where src “1”s enable a CAS/DQM and src “0”s disable a CAS/DQM. Thus block-writes are basically converted to fill-with-value type transfers where the Color Value becomes the fill value. The dst transfers are performed as normal 64-bit write cycles in which writes to some bytes are disabled as specified by the src bits.

8.1.6 Color Register Loading

Before 4x or 8x block-write cycles can be performed, the VRAM or SGRAM color registers must be loaded with the correct color value. The TC does this by performing either a load color register (LCR) cycle (for VRAMs) or a special register set (SRS) cycle (for SGRAMs). The data written during the cycle is the Color Value contained in the packet transfer parameter table. Because a block-write packet transfer could be interrupted by a higher priority request which could, itself, change the VRAM/SGRAM color registers, an LCR/SRS cycle will also be performed whenever an interrupted block-write PT resumes. An LCR/SRS therefore, is performed whenever:

- ❑ A 4x or 8x block-write mode transfer begins.
- ❑ A block-write mode packet transfer resumes after being suspended.
- ❑ A block-write mode packet transfer resumes after return from a host request.

An LCR/SRS cycle is *not* performed if the memory bank is configured for simulated block writes.

Once the color registers have been loaded for a block-write mode packet transfer, no more LCR/SRS cycles are performed unless one of the above conditions occurs. Therefore the entire range of dst addresses should lie within the same configuration cache entry.

8.2 Transparency Modes

Transparency mode is available only for long-form packet transfers. Transparency mode packet transfers are selected by setting the packet transfer access mode (**PAM**) bits in the PT Options field to ‘1xx’ as shown in Table 8–2. This enables transparency-on-source operation. When performing the packet transfer, the src data is compared to the 64-bit Transparency value specified in the parameter table. Transparency may be specified for a 64-, 32-, 16-, or 8-bit data size. This causes one 64-bit, two 32-bit, four 16-bit, or eight 8-bit comparisons between the src data and transparency value 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.

Table 8–2. Transparency Mode Selection

Bit			
18	**17**	**16**	**Access Mode**
0	0	0	Normal
0	0	1	Peripheral Device Transfer
0	1	0	Block-Write
0	1	1	Reserved
1	0	0	8-bit Source Transparency
1	0	1	16-bit Source Transparency
1	1	0	32-bit Source Transparency
1	1	1	64-bit Source Transparency

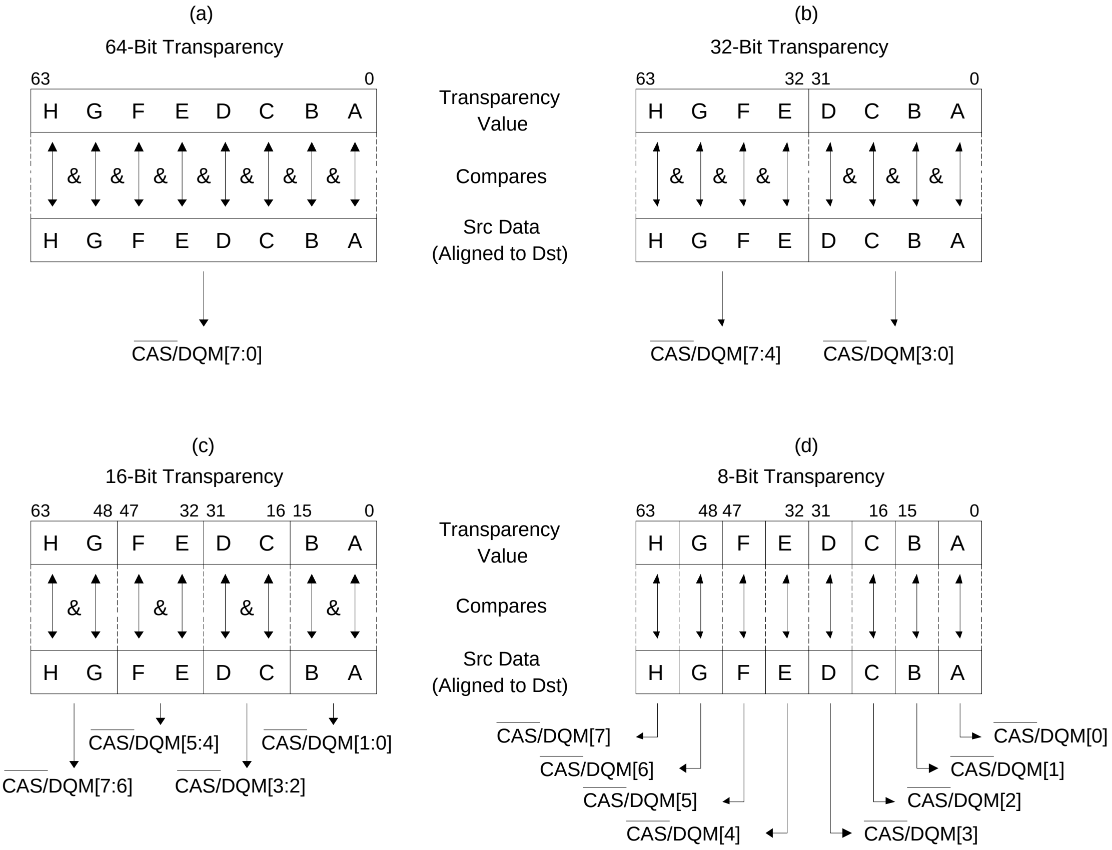
The TC uses the CAS/DQM[7:0] byte strobes to perform transparency. Therefore transparency is supported for off-chip dst addresses only. Attempting to perform a transparency mode packet transfer to an on-chip dst address will result in an error and cause the transfer to be suspended.

Transparency detection is applied after the src data has been aligned to the dst address and memory bus size. All eight bytes of src 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 (CAS/DQM outputs) associated with that group are disabled, preventing a write to any of the bytes within that group.

Figure 8–6 shows how comparisons are made for the different transparency sizes. The ampersand (&) symbols indicate which comparisons are ANDed together to form a group. As Figure 8–6 (a) shows, 64-bit transparency causes a single 64-bit comparison to be made. If the src data and transparency value are equal, all the CAS/DQM strobes are disabled. Otherwise all eight bytes of src data are written to the dst. In Figure 8–6 (b), two 32-bit comparisons are made which control the CAS/DQM[7:4] and CAS/DQM[3:0] strobes. Figure 8–6 (c) and (d) show the comparisons for 16- and 8-bit transparency for which each comparison controls two and one CAS/DQM strobes respectively. Note that the CAS/DQM[7:0] strobes are always identified with the same bits on the data bus regardless of the endian mode. Thus Figure 8–6 applies to both big- and little-endian formats.

The transparency comparisons take place *after* alignment of the src data to the external bus. If the external bus size is 32-bits, src data will always be compared to bits 31-0 (bits 63-32 in big endian) of the transparency value even if 64-bit transparency is selected. The transparency mechanism works correctly only when the bus size can be divided into an integral number of comparison groups. The bus size should be equal to or greater than the transparency size.

Figure 8–6. Transparency Operation



8.3 Peripheral Device Transfer Mode

Peripheral device mode is available for both short-form and long-form packet transfers and is selected by setting **PAM** = ‘001’ in the PT Options field of the parameter table. This mode allows a peripheral device to use the TC’s memory controller to access ‘C82 off-chip memory. When a peripheral device PT is performed, the ‘C82 will drive its external memory address and control lines normally but place its data bus in hi-impedance. This allows a peripheral device, such as a system bus, to read or write to the addressed memory. The direction of a peripheral device packet transfer is determined by the values of the packet transfer parameters.

8.3.1 Programming Peripheral Device Read Packet Transfers

Peripheral device read packet transfers allow a peripheral device to read data from ‘C82 off-chip memory. To perform this type of transfer, the PT src parameters are programmed to access the memory addresses of the data needed by the peripheral. The dst portion of the transfer must be disabled by programming the dst Start Address to `0x00000000`. For long form packet transfers, the dst Transfer Mode (**DTM** in PT Options) must be set to ‘000’, the src transfer mode (**STM**) can be either dimensioned or guided (not fill-with-value). The src Start/Base Address must point off-chip.

8.3.2 Programming Peripheral Device Write Packet Transfers

Peripheral device write packet transfers allow a peripheral device to write data to ‘C82 off-chip memory. To perform this type of transfer, the PT dst parameters are programmed to access the memory addresses which the peripheral needs to write. The src portion of the transfer must be disabled by programming the src Start Address to `0x00000000`. For long form packet transfers, the src Transfer Mode (**STM** in PT Options) must be set to ‘000’, the dst Transfer Mode (**DTM**) can be either dimensioned or guided. The dst Start/Base Address must point off-chip.

8.3.3 Initiating Peripheral Device Packet Transfers

A peripheral device packet transfer request can be submitted by any processor at any priority level and serviced using the normal prioritization scheme. Typical operation, however, is to have the peripheral device initiate the transfer as an XPT whenever it needs to read or write data.

8.3.4 Transfer Synchronization

Once a peripheral device packet transfer request has been submitted, the peripheral device must wait for the TC to begin the external memory cycles before it attempts to read to write data. The beginning of the peripheral device packet transfer is signaled by a special access-type code output on AD[35:32] at row time. (AD[35:32] = ‘0010’ for peripheral device reads and ‘0011’ for writes.) The peripheral device must monitor the access-type code to determine when it should begin transferring data. The type (cycle timing, bus size, etc.) of memory cycles generated by the TC is determined by the values in the corresponding cycle configuration cache entry. Once the peripheral detects the appropriate access-type code, it can synchronize its data transfer to the subsequent column accesses using CAS/DQM[7:0], CLKOUT, etc.

Because memory addressing is generated by the packet 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 transfer.

Since the peripheral device uses the ‘C82’s data bus for the transfer, it must not drive the bus until the data portion of the peripheral transfer begins. This is achieved by placing transceivers between the peripheral data bus and the ‘C82’s data bus and enabling them only during peripheral transfers. The ‘C82’s data bus will be put into the high-impedance state during the transfer and the DBEN output will be driven high to disable its own external transceivers (if present).

8.4 Read Transfer Modes

The read transfer packet transfer modes allow the control of VRAM based display buffers. This mode is available for short-form packet transfers only and is selected by setting the **PAM** = ‘01x’ in the PT Options field as seen in Table 8–3. Specifying one of these modes causes the TC to perform memory-to-register transfer cycles which load VRAM serial registers with display data from their memory arrays.

Table 8–3. Read Transfer Mode Selection

Bit			
18	**17**	**16**	**Access Mode**
0	0	0	Normal
0	0	1	Peripheral Device Transfer
0	1	0	Split Read Transfer
0	1	1	Read Transfer
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Since read transfer cycles are used to control VRAM display buffers during active display. They will typically need to occur at least once during each display line. These packet transfers will typically occur as short form XPTs which are requested by the display timing logic.

The short form packet transfer parameter table for read transfers is shown in Figure 8–7. The src Start Address is used to specify the row address of the data to be transferred into the VRAM serial register and the tap point of the register (to indicate which bit is to be shifted out first). The Count/Update field specifies the amount by which the src Start Address is to be incremented in preparation for the next transfer.

Figure 8–7. Read Transfer Packet Transfer Parameters

Word Number	31 … 16	15 … 0	Byte Address
0	Next Entry Address		PT
1	PT Options	Count / Update	PT+4
0	Src Start Address		PT+8
1	Don't Care		PT+12

8.4.1 Full Read Transfers

Full read transfers are selected by setting **PAM** = ‘`010`’. These transfers generate full memory-to-register transfer cycles which move data from a VRAM memory row into both halves of the VRAM split serial register. These transfers are typically performed at the end of each active line to load the data for the next line into the serial register.

If src update mode is selected (**SUM** = ‘`1`’) in PT Options then a single read transfer cycle is performed to the src Start Address. The value in Count is then added to the start address and the result written back to the src Start Address in the parameter table. Typical operation is to set Count equal to the address pitch between two lines of the display. The display timing logic can submit the short form XPT at the end of each horizontal display line. After the read transfer cycle is performed, the src Start Address will be pointing to the start of the next line and ready for the next XPT. At the end of the active frame, the src Start Address needs to be reset to again point to the first line in the display.

If src update mode is not selected, then Count number of read transfer cycles will be performed to the src Start Address and the address will not be updated following their completion.

8.4.2 Split Read Transfers

Split read transfers are selected by setting **PAM** = ‘`011`’. These transfers generate split memory-to-register transfer cycles which move data from a VRAM memory row into half of the VRAM split serial register. These transfers are typically used when the display lines exceed the length of the VRAM serial register and are packed contiguously into the VRAM array. The split read transfer causes (the inactive) half of the serial register to be updated with new data while old data is being shifted out of the other (active) half. This allows the VRAM to shift out a continuous stream of data which is longer than its serial register.

If src update mode is selected (**SUM** = ‘`1`’) in PT Options then a single split read transfer cycle is performed to the src Start Address. The value in Count is then added to the start address and the result written back to the src Start Address in the parameter table. Typical operation is to set Count equal to the size of the VRAM half serial register. The display timing logic can submit the short form XPT whenever the VRAM changes halves of its serial register to update the inactive half. After the split read transfer cycle is performed the src Start Address will be pointing to the data to be loaded into the next inactive half of the serial register and ready for the next XPT. A full read transfer is normally performed at the start of each frame to initially load the entire VRAM serial register. Split read transfers are then performed whenever the active serial register half changes. At the end of the active frame, the src Start Address needs to be reset to again point to the first split read transfer address.

If src update mode is not selected, then Count number of read transfer cycles will be performed to the src Start Address and the address will not be updated following their completion.

Chapter 9

Packet Transfer Operation

This chapter describes the processes which the TC uses to service packet transfer requests. It provides details on how the TC uses its hardware to move packet data as well as the mechanisms involved when errors occur

Topics in this chapter include

9.1 Packet Data Transfer

Once 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

During the course of a packet transfer, the src and dst addresses can change from on-chip to off-chip and vice-versa. This means that the data flow can change during the packet transfer.

9.1.1 Normal Transfer Process

During the normal packet transfer data 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 data has been received from the crossbar or external memory bus, 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 controllers in sync. If at any time, the FIFO does not contain enough src bytes for the next dst access, the src 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 has processed enough bytes for the next src access to complete. This prevents the src transfer from overrunning the dst transfer. For special packet transfer access modes, this data flow may be somewhat modified. These modified data flows are explained in detail in Chapter 8, *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, urgent refreshes, and host requests do not cause packet transfer suspension but may stall one or both of the src or dst controllers if the request has a higher priority than the PT. Urgent refresh cycles use only the external memory interface. Thus packet transfer crossbar accesses will continue to occur. If one of the controllers is using the external memory interface, the PT FIFO eventually becomes either full or empty, and the controller using the crossbar will stall until the external memory interface becomes available again. If the transfer is on-chip to on-chip, the both src and dst controllers can continue unimpeded. Cache and DEA servicing use both crossbar and external memory interfaces and they always stall any packet transfer until their service is complete.

9.1.2 Off-Chip to Off-Chip Transfer Process

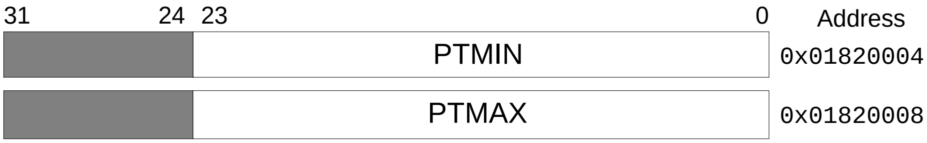
Off-chip src to off-chip dst packet transfers are handled differently from other data flows. To take advantage of the page mode architecture of DRAMs and SDRAMs, the TC performs a page-mode burst of column accesses from the off-chip src to on-chip and then another page-mode burst from on-chip to the off-chip dst. To perform the two transfers, the TC makes use of an on-chip PT buffer.

Each processor has a 128-byte area in its parameter RAM reserved for off-chip to off-chip transfers. In addition, a second 128-byte area in the MP parameter RAM is reserved for off-chip to off-chip XPTs. The transfer of data in and out of this buffer is handled by the TC hardware and is transparent to the user. The src transfer will read up to 128 bytes at a time from off-chip memory and write them into the buffer area. The buffer is considered full whenever it has less than 8 empty bytes remaining. Thus the src alignment and A Count may cause a src transfer burst to end anywhere between 121 and 128 bytes. The dst transfer will completely empty the on-chip buffer and write the data to the off-chip dst. Once the dst burst is complete, the cycle will repeat until all the data has been transferred.

9.2 Controlling PT Execution Time

Packet transfers that are very large can starve other tasks if they take a long time to complete. Transfers with low priority may never complete if allowed to be continually interrupted by higher priority packet transfers. Controlling a PT's execution time prevents either one of these problems from occurring. To do this, the TC has two 24-bit registers **PTMIN** and **PTMAX**, that are used to specify the minimum and maximum execution time of a packet transfer. These registers are shown in Figure 9–1. The TC also has an internal register PTCOUNT which tracks the execution time of a packet transfer by cycles.

Figure 9–1. Packet Transfer Length Registers



9.2.1 Setting the Minimum Packet Transfer Length

The **PTMIN** register is used to indicate the minimum number of clock cycles for which a packet transfer must execute before it can be interrupted by a *higher priority* packet transfer request. This value is loaded into PTCOUNT *after* a packet's parameters have been loaded and data transfer begins. PTCOUNT is decremented by one on each clock cycle during which the TC is actively transferring packet data. PTCOUNT does not decrement during the servicing of cache, DEA, or short form XPT requests which may interrupt the packet transfer. PTCOUNT does decrement during wait states and pipeline bubbles that occur during packet data transfer.

The current packet transfer cannot be suspended by a higher priority packet transfer request until PTCOUNT reaches 0. The PT can be suspended only by the requesting processor or by a packet transfer error or fault.

The value programmed into **PTMIN** is very dependent on the size, frequency, and types of packet transfers which will occur in the system. If a low-priority PT is not being completed because of frequent high-priority PTs then **PTMIN** can be increased so that the low priority PT never suspends. On the other hand, if time-critical high-priority PTs or long form XPTs are not being serviced in time, then **PTMIN** should be decreased so low-priority PTs will suspend sooner.

PTMIN is loaded with the value `0x010000` (65,536 cycles) at reset.

9.2.2 Setting the Maximum Packet Transfer Length

The **PTMAX** register is used in conjunction with **PTMIN** to determine the maximum number of clock cycles for which a packet transfer may execute before it suspends. Once the **PTMIN** time has elapsed (PTCOUNT has decremented to 0), PTMAX is loaded into PTCOUNT. PTCOUNT then decrements by one on each clock cycle during which the TC is actively transferring packet data. If PTCOUNT reaches 0 before the packet transfer completes then the transfer is considered to have timed-out.

Once a transfer times out, it is suspended and the TC will service the next request in that packet transfer priority round robin. If no other packet transfer request of the same priority is pending, the suspended transfer is immediately reloaded and PTCOUNT is reloaded with **PTMIN** to restart the timing process. If a *higher* priority PT request is received, the current transfer is suspended (assuming the **PTMIN** time has been satisfied) regardless of whether the count down from **PTMAX** has completed.

PTMAX is loaded with `0x010000` (65,536 cycles) at reset.

9.2.3 PTMIN and PTMAX Application

The **PTMIN** and **PTMAX** values are applied whenever a packet transfer is loaded. They are applied to each packet transfer within a linked list. If a transfer times-out and gets 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 state of the suspended transfer is saved in the requesting processor’s parameter RAM. When the round robin token is returned to the processor, its suspended transfer is automatically reloaded. PTCOUNT is then reloaded with **PTMIN** and the timing process is restarted.

If **PTMIN** and **PTMAX** are programmed to their maximum values, a packet transfer will time out after approximately .67 seconds of active data transfer (assuming 50 MHz ‘C82 operation).

9.3 Linked List Management

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

❑ If a refresh, host request, cache request, DEA request, or short form XPT request is received during a packet transfer, the current state of the packet transfer is retained in the TC's internal registers. The src and/or dst transfer is stalled if the interrupting request requires the same (crossbar or external) interface. When the interrupting request has been serviced, the packet transfer resumes. Two exceptions to this rule are:

n An XPT will always complete before a cache or DEA request is serviced. (Recall that XPTs have higher priority.)

n A high-priority-mode MP urgent packet transfer will complete before a PP cache/DEA request or low-priority-mode MP cache/DEA request is serviced. (Because the MP is in high-priority mode, the urgent PT has higher priority.)

❑ An internally requested packet transfer linked list will be suspended by a higher priority PT, a time out, a suspend request, a fault, or an error. (See Section Section 9.5, *Packet Transfer Suspension* for more information on suspend conditions.)

❑ If an internally requested packet transfer is suspended, the state of the transfer is saved to parameter RAM (suspend area) and the Linked List Start Address is changed to point to the suspend area.

❑ If a packet transfer is suspended by a higher priority PT request, the round robin token for the lower priority remains with the suspended transfer. When the higher priority request is complete the TC returns to servicing the lower priority requests and the suspended transfer is resumed because it still has the token for that priority level.

❑ If the higher priority PT request occurs while a lower priority PT's parameter table is being loaded then the low-priority PT load is abandoned and no suspension occurs. When the higher priority PT is complete, the low priority PT will begin loading again from its original parameter table.

❑ If a packet transfer is suspended because of a time out, fault, error, or suspension request, then the round robin token is advanced, placing the suspended link list at the end of the priority chain.

❑ If the **I** (interrupt) bit in the PT Options field is set, the requesting processor is sent an end-of-packet interrupt (**pc** bit in the MP's **INTPEN** register or **PTEND** in a PP's **intflag** register) when the packet transfer is finished. The processing of the linked list then continues with the load of the next packet transfer. (For XPTs, no interrupt is generated. Instead, the XPT complete status code is output on the last external column access.)

❑ When the last packet transfer (the one with its stop (**S**) bit set in PT Options) in a linked list is completed, the requesting processor is sent an end-of-packet interrupt (**pc** bit in the MP's **INTPEN** register or **PTEND** in a PP's **intflag** register). (For XPTs, no interrupt is generated. Instead, the XPT complete status code is output on the last external column access.)

❑ When a packet transfer has been completed, the Next Address field from the packet transfer parameter table is written to the Linked List Start Address location in the appropriate parameter RAM. This occurs even if the PT's stop bit was set.

❑ When a packet transfer's parameter table is loaded, if the **PTS** bits in PT Options indicate that the packet transfer was suspended, then the additional state information of the suspended transfer is loaded.

9.4 Packet Transfer Errors

If an error occurs during packet transfer service, the TC immediately stops transferring data and suspends the transfer. The requesting processor's PT error flag (**bp** bit in MP's **INTPEN** or **PTERR** bit in a PP's **intflag**) is set and its linked list is terminated (the **Q** bit is cleared). An interrupt routine on the requesting processor can then examine the suspend parameters to determine the cause or the error.

Any of the following conditions will cause a packet transfer error:

❏ Attempting a block-write transfer with an on-chip dst address.

❏ Attempting a long form XPT with block-write mode selected.

❏ Specifying transparency in a packet transfer with an on-chip dst address

❏ The length (total number of bytes) of a packet's dst transfer exceeding the length of the src transfer. (Fill-with-value and PDPT writes are exceptions and will not cause an error under this condition.)

❏ Specifying one of the reserved src or dst access modes.

❏ Attempting a PDPT read with an on-chip src address. This includes the start/base address of guided PDPT reads.

❏ Attempting a PDPT write with an on-chip dst address. This includes the start/base address of guided PDPT writes.

❏ The Linked List Start Address in parameter RAM or the Next Entry Address in the PT parameter table points off-chip or is not properly aligned. (Alignment is checked only to a 16-byte boundary which is the short form PT requirement.)

❏ The start or base address of a guided transfer does not point to valid on-chip memory or is not properly aligned.

The last three conditions listed will not cause any information to be written to the PT suspend area because the transfer never actually begins. When attempting to find the cause of a PT error, check the Linked List Start Address to see that it points to the suspend area to ensure that the suspend parameters are valid. No information is provided to indicate which condition caused the error. This must be deduced from the values of the suspend parameters.

9.5 Packet Transfer Suspension

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

- ❑ The TC receives a higher priority packet transfer request and **PTMIN** cycles for the current transfer have occurred.
- ❑ The packet times out (exceeds **PTMIN** + **PTMAX** cycles)
- ❑ The requesting processor suspends the packet transfer (by setting its **S** bit)
- ❑ A memory fault occurs during src addressing
- ❑ A memory fault occurs during dst addressing
- ❑ A packet transfer error occurs

The suspension mechanism for each case is identical and involves saving the current packet transfer parameters and the internal state of the TC packet transfer hardware. This information allows a suspended packet transfer to restart from the exact point at which it was suspended.

The parameters and state information are saved to the suspension area of the requesting processor's parameter RAM. (XPTs cannot be suspended and thus have no suspend area.) Suspension begins immediately after a suspend condition arises and any pending external memory column accesses have completed. If the suspension is due to a higher priority PT request then the transfer of the information into the parameter RAM is performed at the TC crossbar priority level of the higher priority PT. The MP or PPs may therefore experience contention for the parameter RAM while the suspension completes. If the suspension is due to a time out, a fault, an error, or a suspend request then it is performed at the original transfer's TC crossbar priority.

If the suspending transfer is an off-chip to off-chip type, the 128 byte off-chip to off-chip buffer is not emptied but the current state of the buffer is saved. This ensures that an interrupting transfer can be serviced without having to wait for the buffer to empty.

Once a packet transfer has been suspended, the suspend parameters can be copied to another on-chip location. The transfer can then be resubmitted from the new location, provided the Linked List Start Address points to the new address. Because the **PTS** bits in PT Options indicate that the transfer was suspended, the entire set of suspend parameters will be loaded no matter where they are located. The new starting address of the suspended parameter table must be 128-byte aligned in order for it to be resubmitted.

The suspension mechanism is identical for both long- and short-form packet transfers. Both save the same number of parameters, however, not all saved parameters are valid for a short-form suspension because it uses only a single transfer dimension.

9.5.1 Suspended Long-Form Packet Transfer Parameters

The format of the suspended parameters for long-form packet transfers is shown in Figure 9–2 and Figure 9–3 for big and little endian operation, respectively.

Figure 9–2. Suspended Long-Form Packet Parameters (Big Endian)

MP Address	PP Address	Word 0		Word 1	
		63	32	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 Pointer		Dst C Pitch / Guide Pointer	
0x01010030	0x0100#030	Src Transparency Value / Color Register Value Ü			
0x01010038	0x0100#038	Don't Care Ü			
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

Ü This double word not adjusted for endianess

Figure 9–3. Suspended Long-Form Packet Transfer Parameters (Little Endian)

Word 1		Word 0		MP Address	PP Address
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 Count		0x01010018	0x0100#018
Dst B Pitch		Src B Pitch		0x01010020	0x0100#020
Dst C Pitch / Guide Pointer		Src C Pitch / Guide Pointer		0x01010028	0x0100#028
Src Transparency Value / Color Register Value Ü				0x01010030	0x0100#030
Don't Care Ü				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

Ü This double word not adjusted for endianess

# - PP Number

The first eight 64-bit doublewords saved in the packet transfer suspend area are formatted like the original parameter table. The saved values are the same as those programmed into the parameter table with the following exceptions:

- The Next Entry Address field is replaced with the starting address of the original parameter table (Original PT Entry Address).
- The src and dst Start Addresses are replaced with the current addresses (Current src Address and Current dst Address). The current address is the next address at which the src or dst machine will read or write data.
- The C Count fields represent the current real-time value of the C Counts or guide table Number of Entries rather than their original values. (The C Count decrements as each packet is completed.)
- The Guide Table Pointer fields contain the current value of their pointers rather than the original value. (The Guide Table Pointer is updated after each packet to point to the next entry.)
- The **PTS** bits in the PT Options field reflect the suspended status.

If the packet suspension was caused by an error, the programmer can look at the modified parameters in the suspend area to determine where in the transfer the error occurred.

The suspend area also contains eight reserved doubleword which contain TC internal state information. These fields contain information about the intradimensional state of the transfer when it suspended. They are automatically loaded when a suspended transfer is restarted and allow the transfer to resume exactly when it left off.

9.5.2 Suspended Short-Form Packet Transfer Parameters

The format of the suspended parameters for short-form packet transfers is shown in Figure 9–4 and Figure 9–5.

Figure 9–4. Suspended Short-Form Packet Transfer Parameters (Big Endian)

MP Address	PP Address	Word 0		Word 1	
		63	32	31	0
0x01010000	0x0100#000	Original PT Entry Address		PT Options	Count
0x01010008	0x0100#008	Current Src Address		Current Dst Address	
0x01010010	0x0100#010	Don't Care	Don't Care	Don't Care	Don't Care
0x01010018	0x0100#018	Don't Care		Don't Care	
0x01010020	0x0100#020	Don't Care		Don't Care	
0x01010028	0x0100#028	Don't Care		Don't Care	
0x01010030	0x0100#030	Don't Care			Ü
0x01010038	0x0100#038	Don't Care			Ü
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

Ü This double word not adjusted for endianess

Figure 9–5. Suspended Short-Form Packet Transfer Parameters (Little Endian)

Word 1		Word 0			
63	32	31	0	MP Address	PP Address
Count	PT Options	Original PT Entry Address		0x01010000	0x0100#000
Current Dst Address		Current Src Address		0x01010008	0x0100#008
Don't Care	Don't Care	Don't Care	Don't Care	0x01010010	0x0100#010
Don't Care		Don't Care		0x01010018	0x0100#018
Don't Care		Don't Care		0x01010020	0x0100#020
Don't Care		Don't Care		0x01010028	0x0100#028
Don't Care Ü				0x01010030	0x0100#030
Don't Care Ü				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

Ü This double word not adjusted for endianess

# - PP Number

The first two 64-bit doublewords saved in the packet transfer suspend area are formatted like the original parameter table. The saved values are as follows:

- The Original PT Entry Address field contains the starting address of the original parameter table.
- The src and dst Start Current Address contain the next address at which the src and dst machine will read and write data.
- The **PTS** bits in the PT Options field reflect the suspended status.

If the packet suspension was caused by an error, the programmer can look at the first two doublewords in the suspend area to determine where in the transfer the error occurred.

The next six doublewords in the suspend area contain invalid data. The data written is for the additional parameters used by a long-form PT. For short-form packet transfers it has no meaning and should be ignored.

The suspend area also contains eight reserved doubleword which contain TC internal state information. These fields contain information about the intradimensional state of the transfer when it suspended. They are automatically loaded when a suspended transfer is restarted and allow the transfer to resume exactly when it left off.

Chapter 10

External Memory Interface

The TMS320C82’s external memory interface is a highly flexible, programmable interface that allows the transfer controller to access a wide range of memory devices and peripherals. Direct support is provided for SRAM, EDO DRAM, SDRAM and VRAM devices and peripherals such as video and audio capture devices.

The memory interface contains a special memory configuration cache which stores memory device information for system devices. This information allows the timing of each memory access to be optimized for the device being accessed.

Topics in this chapter include:

10.1 Memory Interface Signals

The following is a list of signals that are used by the TMS320C82 to interface to external memory and peripherals.

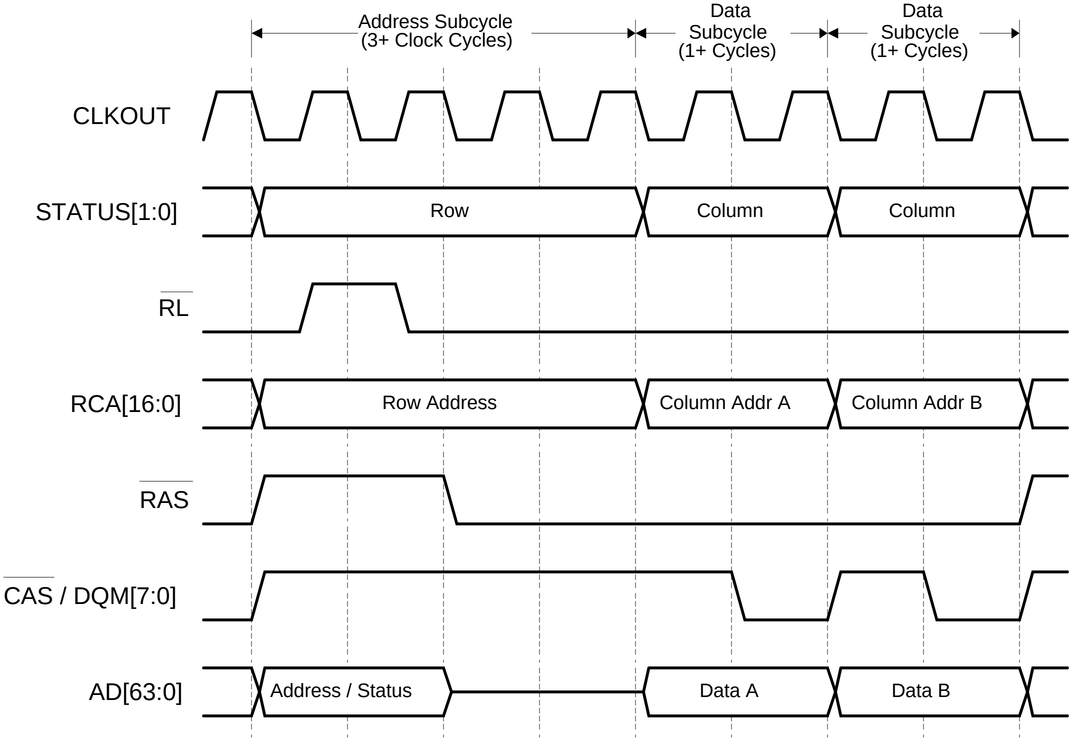
Signal Name	Direction	Description
AD[63:0]	I/O	**Address/Data.** This multiplexed bus serves as the data bus during the data subcycle (column time), allowing the transfer of up to 64-bits per access. During the address subcycle (row time) the 32-bit byte address is output on AD[31:0], and pins AD[39:32] act as a status bus to indicate the type and origin of the memory access being performed.
$\overline{\text{CAS}}$ / DQM[7:0]	O	**Column Address Strobe / Data I/O Mask.** Drive the $\overline{\text{CAS}}$ inputs of EDO DRAMs/VRAMs and DQM inputs of SDRAMs. Eight strobes are provided to control individual byte access. CAS/DQM[0] corresponds to data transfer on AD[7:0], CAS/DQM[7] to data on AD[63:56], etc., irrespective of the endian mode.
CLKOUT	O	**Output Clock.** Allows for synchronous control of external logic. CLKOUT runs at the device clock rate.
DBEN	O	**Data Buffer Output Enable.** Turns on optional system data transceivers.
DDIN	O	**Data Direction Indicator.** Provides direction indication to optional system data transceivers. Active low for read cycles.
DSF	O	**Special Function.** Selects special functions for VRAM or SGRAM cycles.
$\overline{\text{EXCEPT}}$[1:0]	I	**Exception inputs.** These pins are encoded to indicate exceptions to the current memory access (fault, retry, etc.). During refresh cycles they indicate whether a DRAM or SDRAM refresh is to be performed.
RAS	O	**Row Address Strobe.** This pin drives the $\overline{\text{RAS}}$ inputs of DRAMs, VRAMs, and SDRAMs.
RCA[16:0]	I/O	**Row Column Address.** This 17-bit bus provides a row/column multiplexed address for DRAMs and SDRAMs
READY	I	**Ready.** Indicates that the external device is ready for the completion of the memory cycle. An inactive low on this pin causes wait states to be inserted into the memory cycle. It is also used to indicate endian mode at reset.
RL	O	**Row Latch.** This pin indicates that a valid 32-bit row address is present on the address/data bus.
STATUS[1:0]	O	**Status.** This bus provides real-time status of the current memory access.
$\overline{\text{TRG}}$ / $\overline{\text{CAS}}$	O	**Transfer/Output Enable / Column Address Strobe.** This pin serves as an output enable during read cycles. During SDRAM cycles, it serves as the SDRAM $\overline{\text{CAS}}$ input. During VRAM read transfer cycles, $\overline{\text{TRG}}$ serves as a transfer enable.
W	O	**Write Enable.** Indicates that a write cycle is occurring.
$\overline{\text{XPT}}$[3:0]	I	**XPT Request.** These encoded inputs allow one of 15 externally initiated packet transfer requests to be submitted.

10.2 General Form of an External Memory Cycle

Each external memory access generated by the transfer controller is comprised of an **address subcycle** and at least one **data subcycle**. The number of machine states within each subcycle depends on the type of memory device being accessed. A machine state is one clock period in length and begins on the falling edge of CLKOUT.

Whenever possible, the TC generates page-mode cycles consisting of one address subcycle and multiple data subcycles. Page-mode operation is configured within the memory configuration cache. Figure 10–1 shows the general form of a memory access.

Figure 10–1. General Form of an External Memory Access



10.2.1 The Address Subcycle

The address subcycle begins with the first machine state of the external memory cycle and is a minimum of 3 machine states long. The address and status code for the access are output at this time. This portion of the memory cycle is also called the *row address time* because DRAM-type devices latch the row address at this time.

AD[31:0] output the 32-bit address of the first byte in the access. A status code indicating the type of access being performed and the source of the request is output on AD[39:32]. External logic can latch this information with RL falling and use it to decode the current access and select the appropriate memory bank or peripheral. The row portion of the multiplexed row/column address is output on RCA[16:0] at this time. This address is latched by DRAM-type devices using the falling edge of RAS.

The length of the address subcycle is automatically extended by an integral number of machine states as needed for the memory type being addressed. (This is based on the contents of the memory configuration cache entry for the current access.) The address subcycle may also be extended by the insertion of wait states.

10.2.2 The Data Subcycle

The data subcycle is at least one machine state long and immediately follows the address subcycle. The column address for DRAM-type devices is output at this time, and data is transferred between the ‘C82 and external memory. This period of the external memory cycle is also referred to as the *column address time*.

The AD[63:0] bus transfers the data between the ‘C82 and memory. During a write cycle, data is driven out from the ‘C82 and during a read cycle, data is latched in. The position of valid data on the bus is determined by the endian mode, the amount of data being transferred, and the width of the external memory (see Chapter 12, *Endianess and Bus Sizing*). The column portion of the multiplexed row/column address is output on RCA[16:0] at this time and latched by DRAM-type devices using the falling edges of CAS/DQM[7:0].

The length of the data subcycle is nominally one, two, or three machine states as determined by the current memory configuration cache entry. For devices requiring additional access time, wait states can be inserted into the data subcycle (for two and three machine state subcycles only).

Whenever the current memory access is in the same direction and is within the same memory page as the previous access, page-mode cycles will be used. This allows another data subcycle to be generated without the overhead of the address subcycle. For page-mode cycles, data need not be contiguous, only within the same memory page.

10.3 Memory Cycle Configuration

The ‘C82 supports a large number of memory and peripheral devices. In order to minimize the amount of external logic required to interface to these devices, each memory access can be tailored to match the particular device being addressed. Configuration options include cycle timing, address multiplexing, and bus width, among others. This configuration information is kept on-chip within the TC’s cycle configuration cache.

10.3.1 Cycle Configuration Cache Overview

The cycle configuration cache stores the configuration information for up to six different external memory banks. The information controls the operation of the external memory interface state sequencer and the transition of external memory control signals. This allows each memory cycle to be optimized for the type of device being accessed.

Figure 10–2 shows the cycle configuration cache structure. Each cache entry is a 32-bit word. A 32-bit address register is associated with each cache entry and a valid bit marks whether or not the entry has been loaded with valid configuration data.

Figure 10–2. Cycle Configuration Cache Structure

Valid	Tag Address	Cache Entry
	31 0	31 0
V0	Bank 0 Address	Bank 0 Configuration
V1	Bank 1 Address	Bank 1 Configuration
V2	Bank 2 Address	Bank 2 Configuration
V3	Bank 3 Address	Bank 3 Configuration
V4	Bank 4 Address	Bank 4 Configuration
V5	Bank 5 Address	Bank 5 Configuration

Whenever an external memory access is required, the address is compared to the bank addresses of the valid cache entries. If a match occurs, then the memory cycle proceeds using the cache entry information. If no match occurs, then a cache entry is created for the new bank. The entry is created using data read during a special bank configuration cycle. Once the configuration cycle is complete, the access to the new memory bank proceeds using the corresponding configuration information.

When performing the address comparison, the full 32-bit address is not typically used. Each cache entry contains a memory bank size (the **MS** bits) which specifies the size of the bank for which the cache entry corresponds. This determines which address bits in each of the tag addresses will be compared to the address of the current access. Thus each tag covers a range of addresses corresponding to the size of its associated memory bank.

10.3.2 Configuration Cache Entry Replacement

If an access to a memory location without a matching tag occurs and all cache entries are full, then one of the current cache entries will be replaced with a new entry corresponding to the current location. Selection of the entry for replacement is made based on a multilevel LRU (least recently used) algorithm. Each cache entry contains a priority level field (the **PL** bits) which allow it to be assigned a low, medium, or high priority. When a cache entry must be flushed, the least recently used entry of the lowest priority level is selected. Thus any low priority entries will be flushed prior to selection of a medium priority entry and any medium priority entries will be flushed prior to any high priority entries.

The multilevel LRU replacement scheme allows the system designer to minimize (or eliminate) the possibility that an important memory or peripheral device will be flushed from the cache. A high priority entry (such as a FIFO which holds real-time data) will always remain in the cache unless five more high-priority banks are also loaded. Medium priority can be used for frequently accessed memory (such as instructions or data). These entries are unlikely to be flushed unless the system contains a number of different memory types (requiring many cache entries) or unless there are already many high-priority cache entries. Low priority entries should be used for seldom accessed devices where access time is not critical. This typically includes peripheral devices which are not accessed except for initialization or mode changes. These entries will likely be flushed from the cache unless the entire system requires six or less cache entries.

A cache entry may also be flushed and reconfigured by system request. During any memory access, external logic can generate a flush request (using the EXCEPT[1:0] inputs) for the current memory. When this occurs, the associated cache entry is flushed, and a cache configuration cycle is performed to reload the entry. This dynamic configuration of memory enables (for example) features such as mapping RAM over ROM after boot, or adding wait states to compensate for slow bus resources.

10.3.3 Cycle Configuration Cache Format

Figure 10–3 shows the format of a configuration cache entry. Each bit-field in the 32-bit cache entry specifies a different parameter of the associated memory bank. These parameters in turn determine what memory cycles will look like when accessing the bank.

Figure 10–3. Cycle Configuration Cache Entry

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
▨	PL		MS					R	▨	BW		WS		BS		PS				B	AS			E	TO		▨	CT			

The fields and their meanings are listed below. A detailed explanation of each field is contained in Section.

- ❑ **PL(1:0)** - Priority Level. Determines the priority of the cache entry
- ❑ **MS(4:0)** - Memory Bank Size. Specifies the size of the memory bank to which the cache entry applies.
- ❑ **R** - Row-Time Wait State. Enables addition of a wait state at row time.
- ❑ **BW(1:0)** - Block Write Mode. Determines which type of block-write will be used for the memory bank.
- ❑ **WS(1:0)** - Wait States. Enables addition of column-time wait states for certain types of memory cycles.
- ❑ **BS(1:0)** - Bus Size. Specifies the width of the data bus for the memory bank.
- ❑ **PS(3:0)** - Page Size. Used along with **BS(1:0)** to determine the size of a page with in the memory bank.
- ❑ **B** - Bit Replacement. Causes byte address bits to be output on the RCA bus during peripheral device transfers.
- ❑ **AS(2:0)** - Address Shift. Along with **BS(1:0)**, specifies how the 32-bit byte address is to be multiplexed onto the RCA[16:0] pins.
- ❑ **E** - Exceptions. Enables the use of exceptions (fault, retry, etc.) within the memory bank.
- ❑ **TO(1:0)** - Turn-Off. Allows insertion of additional states after a memory access to compensate for devices with slow turn-off times.
- ❑ **CT(3:0)** - Cycle Timing. Specifies the bank memory type (SRAM, DRAM, SDRAM, etc.) and determines the basic cycle timing to be used.

10.4 Memory Cycle Information

Comprehensive information about the current memory cycle is output during each access. This information is useful for bank decode, external logic control, and system debug. The information comes in the form of a real-time memory cycle status and as a per-access “access type” code.

10.4.1 Memory Cycle Status Codes

Throughout the memory cycle, a dedicated status bus, STATUS[1:0], provides information about the current machine state of the external memory cycle being performed. Table 10–1 shows the four status codes available on the bus.

Table 10–1. Memory Cycle Status Codes

STATUS[1:0]	Activity
0 0	Idle/DCAB/Drain
0 1	Row Access
1 0	End of XPT
1 1	Column Access

The memory cycle status codes are defined as follows:

Idle/DCAB/Drain: Output during machine states in which no memory cycle is active or during data subcycle in which no data transfer is occurring. Idle states occur at row time when no memory access is in progress. The idle code is also output during wait states, column pipeline bubbles, and read turn-off cycles. DCAB states occur during SDRAM deactivate bank cycles at the end of each SDRAM page access. Drain states occur at the end of pipelined 1 cyc/col write accesses.

Row Access: Output during address subcycle (row time).

End of XPT: Special code output during the last column access of an XPT src or dst transfer. Used to signal external devices that the XPT is complete.

Column Access: Output during the data subcycle (column-time) when active data transfer is occurring.

10.4.2 Row-time Access Type Codes

During row time, the AD[39:32] pins act as a status bus to indicate the type and origin of the memory cycle being performed. AD[35:32] indicate the type of activity (or access) being performed, whereas AD[39:36] indicate the source of the request that initiated the activity. The TMS320C82 source code indicates that the request originated from on-chip (the MP, a PP, or the TC itself). The other source codes indicate that the TC is servicing a request caused by one of the XPTs. Any activity can occur with any source except refresh, SDRAM DCAB and MRS, and idle cycles which can originate only from within the ‘C82. Table 10–2 shows the row-time access type codes. The row-time codes can be latched by fall of RL and used by external logic to perform memory bank decoding or to enable special hardware features.

Table 10–2. Row-Time Access Type Codes

AD					AD				
39	**38**	**37**	**36**	**Source**	**35**	**34**	**33**	**32**	**Activity**
0	0	0	0	TMS320C82	0	0	0	0	Read
0	0	0	1	XPT1	0	0	0	1	Write
0	0	1	0	XPT2	0	0	1	0	PDPT Read
0	0	1	1	XPT3	0	0	1	1	PDPT Write
0	1	0	0	XPT4	0	1	0	0	Read Transfer
0	1	0	1	XPT5	0	1	0	1	Reserved
0	1	1	0	XPT6	0	1	1	0	Split Read Transfer
0	1	1	1	XPT7	0	1	1	1	Reserved
1	0	0	0	XPT8	1	0	0	0	SDRAM MRS
1	0	0	1	XPT9	1	0	0	1	Block Write
1	0	1	0	XPT10	1	0	1	0	Reserved
1	0	1	1	XPT11	1	0	1	1	Load Color Register
1	1	0	0	XPT12	1	1	0	0	Refresh
1	1	0	1	XPT13	1	1	0	1	SDRAM DCAB
1	1	1	0	XPT14	1	1	1	0	Bank Configuration
1	1	1	1	XPT15	1	1	1	1	Idle

The row-time status codes are defined as follows:

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

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

PDPT Read: Output for a memory read cycle occurring as a result of a processor-requested or XPT initiated peripheral device packet transfer. Indicates that the data read from memory on the subsequent column accesses should be latched by the peripheral device that initiated the transfer.

PDPT Write: Output for a memory write cycle occurring as a result of a processor-requested or XPT initiated peripheral device packet transfer. Indicates that the ‘C82 will be placing its data bus in high-impedance state during the subsequent column accesses so that the peripheral device may drive the bus with data to be placed in memory.

Read Transfer: Output during short-form packet transfers that perform VRAM read (memory to register) transfers.

SDRAM MRS: Output during SDRAM mode register set (MRS) cycle. This cycle is performed only if SDRAM is present in the system.

Block Write: Output during a block write cycle generated by a packet transfer with block-write access mode enabled.

Load Color Register: Output during the load color register (LCR) cycle of a packet transfer with block-write access mode.

Refresh: Output during trickle refresh cycles and during burst refresh cycles generated by an urgent refresh request from the TC’s refresh controller.

SDRAM DCAB: Output during SDRAM deactivate cycle during power-up initialization. This cycle is performed only if SDRAMs are present in the system.

Bank Configuration: Output during a memory bank cache configuration cycle.

Idle: Output when no access is being performed (row-time idle).

10.5 Memory Bank Configuration

Whenever the transfer controller performs an external memory cycle, it configures the cycle to the addressed memory device using the parameters contained in the memory bank configuration cache entry. The following sections discuss the various configuration options available and their effect on memory cycles.

10.5.1 Column Timing Selection

The TC supports fourteen sets of memory timings allowing it to interface to a wide variety of memory devices and peripherals. Primary support is for EDO DRAM/VRAM, SRAM, and SDRAM devices but cycle timings also support (or can be modified to support) other devices such as FIFOs and peripheral chips. The cycle timing configuration field, **CT(3:0)**, in the cycle configuration cache selects the timing to be used for all accesses corresponding to that cache entry. Table 10–3 shows cycle timing as selected by **CT(3:0)**. A complete description of each cycle timing can be found in the appropriate cycle description chapter.

Table 10–3. Cycle Timing Selection

CT(3:0)	Cycle Timing
0 0 0 0	Pipelined 1 cycle/column EDO DRAM
0 0 0 1	Non-pipelined 1 cycle/column EDO DRAM
0 0 1 0	2 cycle/column EDO DRAM
0 0 1 1	3 cycle/column EDO DRAM
0 1 0 0	1 cycle/column synchronous SRAM
0 1 0 1	1 cycle/column asynchronous SRAM
0 1 1 0	2 cycle/column asynchronous SRAM
0 1 1 1	3 cycle/column asynchronous SRAM
1 0 0 0	Burst Length 1; read latency 2 SDRAM
1 0 0 1	Burst Length 1; read latency 3 SDRAM
1 0 1 0	Burst Length 1; read latency 4 SDRAM
1 0 1 1	Reserved
1 1 0 0	Burst Length 2; read latency 2 SDRAM
1 1 0 1	Burst Length 2; read latency 3 SDRAM
1 1 1 0	Burst Length 2; read latency 4 SDRAM
1 1 1 1	Reserved

10.5.2 Turn-Off Cycle Selection

The ‘C82’s AD[63:0] pins form a multiplexed address and data bus. Address and access-type information are driven on the bus at row time and data is written or read over the bus at column time. Because of this the potential for drive conflict exists during row-time following a read access to a slow device with long turn-off time (such as EPROM). To prevent this conflict, additional machine states may be selectively added to the end of accesses to slow devices to allow extra turn-off time before row address is driven.

The **TO(1:0)** bits of the configuration cache enable the insertion of 1-3 turn-off cycles as shown in Table 10–4. The selected number of turn-off cycles will be added to all read accesses and PDPT writes to the corresponding memory bank.

Table 10–4. Turn-Off Cycle Selection

TO(1:0)	Added Turn-Off Cycles
0 0	None
0 1	1
1 0	2
1 1	3

For some memory accesses, the TC automatically inserts a turn-off state (rto) at the end of the access as required by the selected cycle timing. Turn-off cycles specified by **TO(1:0)** will occur *in addition to* any turn-off states automatically generated for the access.

10.5.3 Exception Support Enabling

During row access time, cycle exceptions such as fault and retry may be input on the ‘C82’s EXCEPT[1:0] inputs. However, supporting these exceptions can increase the cycle time for fast memory accesses (particularly writes), because the TC must sample and decode these inputs before loading the memory access into its internal pipeline. In order to prevent this delay, exception support can be selectively enabled for each configuration cache entry through the **E** bit. If **E** = ‘1’, row-time exceptions are supported and addition row-time states may be added to allow sampling of the EXCEPT[1:0] inputs. If **E** = ‘0’, then row-time exceptions are not supported, exception codes input on EXCEPT[1:0] at row time are ignored, and the minimum number of row-time states are generated.

10.5.4 Address Multiplexing

The TMS320C82 provides muxed row and column addresses to support a variety of DRAM-type devices. During row-time, the TC outputs the complete 32-bit byte address on AD[31:0]. The address is shifted and muxed onto the RCA[16:0] bus so that high order bits are output at row time and low order bits are output at column time. The alignment of the logical address bits on the RCA bus is determined by the value of the address shift specified in the **AS(2:0)** bits of the configuration cache and by the bus size as specified by **BS(1:0)**. Memories and peripherals are normally connected to RCA[n-1:0] where n is the number of address pins on the device.

Table 10–6 shows how **AS(2:0)** and **BS(1:0)** select the address multiplexing. Note that an **AS(2:0)** value of 000 results in the same unshifted address being output at row and column time. This allows connection to SRAMs and other nonmultiplexed address devices.

Table 10–5. RCA Addressing Multiplexing

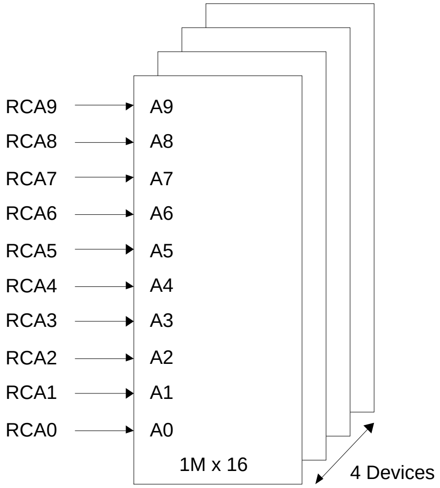
		Logical Address Bits Output on RCA[16:0]																	
AS(2:0)	**BS(1:0)**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0 0 0	0 0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
0 0 0	0 1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
0 0 0	1 0	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
0 0 0	1 1	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
0 0 1	0 0	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
0 0 1	0 1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
0 0 1	1 0	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
0 0 1	1 1	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
0 1 0	0 0	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
0 1 0	0 1	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
0 1 0	1 0	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
0 1 0	1 1	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col

		Logical Address Bits Output on RCA[16:0]																	
AS(2:0)	**BS(1:0)**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0 1 1	0 0	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
0 1 1	0 1	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
0 1 1	1 0	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
0 1 1	1 1	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
1 0 0	0 0	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
1 0 0	0 1	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
1 0 0	1 0	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
1 0 0	1 1	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
1 0 1	0 0	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
1 0 1	0 1	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
1 0 1	1 0	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
1 0 1	1 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
1 1 0	0 0	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
1 1 0	0 1	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
1 1 0	1 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
1 1 0	1 1	xx	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col
1 1 1	0 0	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	Row
		16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Col
1 1 1	0 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	Row
		17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Col
1 1 1	1 0	xx	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Row
		18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	Col
1 1 1	1 1	xx	xx	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	Row
		19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	Col

As an example, consider 1M x 16 DRAMs connected to the RCA bus in a 64-bit wide configuration. The address bus of these memories will be connected as shown in Figure 10–4. The memories require 10 bits each of row and column address. The ‘C82’s A(2:0) bits represent the byte address and can be ignored because the data bus is 64 bits wide and the individual bytes of data are controlled by the CAS/DQM[7:0] strobes. This means that the DRAMs need 20 contiguous address bits starting with bit A(3). Figure 10–4 shows that an **AS(2:0)** value of ‘011’ and **BS(1:0)** value of ‘11’ provide 10 bits of multiplexed address. If the DRAMs are connected as shown in Figure 10–4, then RCA[9:0] will provide address bits A(22:13) at row time and A(12:3) at column time.

Figure 10–4. Address Shift for 1M x 16 DRAM (64-bit Bus)

RCA Pins	9	8	7	6	5	4	3	2	1	0
Row TIme	22	21	20	19	18	17	16	15	14	13
Column TIme	12	11	10	9	8	7	6	5	4	3



10.5.5 PDT Address Bit Replacement

During memory accesses, the low order address bits (0,1, and 2) are selectively output based on the configured bus size. During read accesses, the TC activates all CAS/DQM[7:0] strobes and ignores any invalid bytes it reads in. Without the lower order address bits however, there is no information about on which byte the access is beginning. This poses a problem during peripheral device packet transfers (PDPTs) because the external device which is actually reading the data may not know which bytes are valid and which should be discarded. To compensate, the **B** bit may be set to cause the missing address bits to be output on the upper bits of the RCA bus during column time. External logic may then decode these bits to determine the starting byte of the access. Table 10–6 shows the bit replacement that occurs when **B** = ‘1’. Note that the address output on RCA[16:0] at row time is unaffected.

Table 10–6. PDT Address Bit Replacement

		Logical Address Bits Output on RCA[16:0] at Column Time																
BS(1:0)	**Bus Size**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0 0	8-bit	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 1	16-bit	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1 0	32-bit	1	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
1 1	64-bit	2	1	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3

10.5.6 Page Size

The TC always attempts to perform as many column accesses in page-mode as possible. It therefore must be able to detect when the next column access crosses a page (row) boundary so that it can first perform a row access on the new page. The **PS(3:0)** bits are used in conjunction with the **BS(1:0)** bits to determine the page size for the configuration cache entry.

Whenever an external access begins (row time), the TC records the 29 MSBs of the address in it’s internal LASTPAGE register. The address of each subsequent column access is then compared to this value. As Figure 10–5 shows, the 11 MSBs of the address are always compared. The 18 LSBs of LASTPAGE are selectively compared depending on the values of **PS(3:0)** and **BS(1:0)**. The three LSBs of the column address are always ignored. If there is a difference between the compared bits of LASTPAGE and the external memory address, then a page change has occurred and the next access will begin with an address subcycle (row access). If **PS(3:0)** = ‘`0000`’, then page-mode operation is disabled and every memory access corresponding to the cache entry will occur as a row access and single column access.

Figure 10–5. Page Size Control

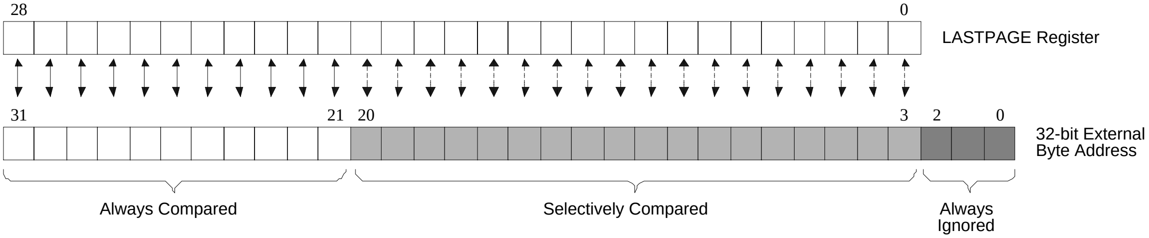


Table 10–7 shows page size selection based on the values of **PS(3:0)** and **BS(1:0)**. The compared address bits and the page size, in bytes, are shown.

The page size should not be confused with the memory bank size. The page size represents the number of different bytes that can be accessed by a single page-mode cycle. For DRAM-type devices this is usually the number of columns in a single row of the RAM array. The memory bank size represents the total number of bytes within the memory device(s) corresponding to the configuration cache entry.

Table 10–7. Page Size Selection

PS(3:0)	BS(1:0)	Address Bits Compared	Page Size (Bytes)	PS(3:0)	BS(1:0)	Address Bits Compared	Page Size (Bytes)
0 0 0 0	0 0	A(31:0)	1[†]	1 0 0 0	0 0	A(31:10)	1K
	0 1	A(31:0)	1-2[†]		0 1	A(31:11)	2K
	1 0	A(31:0)	1-4[†]		1 0	A(31:12)	4K
	1 1	A(31:0)	1-8[†]		1 1	A(31:13)	8K
0 0 0 1	0 0	A(31:3)	8	1 0 0 1	0 0	A(31:11)	2K
	0 1	A(31:4)	16		0 1	A(31:12)	4K
	1 0	A(31:5)	32		1 0	A(31:13)	8K
	1 1	A(31:6)	64		1 1	A(31:14)	16K
0 0 1 0	0 0	A(31:4)	16	1 0 1 0	0 0	A(31:12)	4K
	0 1	A(31:5)	32		0 1	A(31:13)	8K
	1 0	A(31:6)	64		1 0	A(31:14)	16K
	1 1	A(31:7)	128		1 1	A(31:15)	32K
0 0 1 1	0 0	A(31:5)	32	1 0 1 1	0 0	A(31:13)	8K
	0 1	A(31:6)	64		0 1	A(31:14)	16K
	1 0	A(31:7)	128		1 0	A(31:15)	32K
	1 1	A(31:8)	256		1 1	A(31:16)	64K
0 1 0 0	0 0	A(31:6)	64	1 1 0 0	0 0	A(31:14)	16K
	0 1	A(31:7)	128		0 1	A(31:15)	32K
	1 0	A(31:8)	256		1 0	A(31:16)	64K
	1 1	A(31:9)	512		1 1	A(31:17)	128K
0 1 0 1	0 0	A(31:7)	128	1 1 0 1	0 0	A(31:15)	32K
	0 1	A(31:8)	256		0 1	A(31:16)	64K
	1 0	A(31:9)	512		1 0	A(31:17)	128K
	1 1	A(31:10)	1K		1 1	A(31:18)	256K
0 1 1 0	0 0	A(31:8)	256	1 1 1 0	0 0	A(31:16)	64K
	0 1	A(31:9)	512		0 1	A(31:17)	128K
	1 0	A(31:10)	1K		1 0	A(31:18)	256K
	1 1	A(31:11)	2K		1 1	A(31:19)	512K
0 1 1 1	0 0	A(31:9)	512	1 1 1 1	0 0	A(31:17)	128K
	0 1	A(31:10)	1K		0 1	A(31:18)	256K
	1 0	A(31:11)	2K		1 0	A(31:19)	512K
	1 1	A(31:12)	4K		1 1	A(31:20)	1M

[†] PS(3:0)=0000 disables page mode operation so the page mode is equal to the transfer size. The minimum transfer size is 1 byte and the maximum is the bus size.

As an example, assume that the memory being accessed consists of four 1M x 16 DRAMs connected as a 64-bit data bus. Each memory device has a row size of 2^{10} bits. Any location within a row may be accessed during a DRAM page-mode cycle. Thus the page size for this memory bank would be:

$$2^{10}\text{locations/page} \times 8\text{ bytes/location} = 8\text{K bytes/page}$$

As Table 10–7 shows, this corresponds to a **PS(3:0)** value of ‘`1000`’ for a bus size of 64 (**BS(1:0)** = ‘`11`’). When the TC gets a cache hit on this entry, it checks only address bits 31-13 to determine if a page boundary has been crossed. Notice that this corresponds to the DRAM’s row address bits and the bank decode bits for this configuration.

The LASTPAGE register is treated as invalid when the direction of the next access changes (read to write or vice-versa) and after the occurrence of reset, a host access, fault, retry, or flush request. This forces the next memory cycle to begin with a row access. LASTPAGE is also treated as invalid both before and after refresh cycles, read-transfer and split read-transfer cycles, power-up DCAB cycles, and MRS cycles because these must occur as single nonpage-mode accesses.

In the case of peripheral device packet transfers and XPTs, the transfer will always begin with a row access to ensure that the corresponding peripheral device transfer or XPT row-time access code is output on AD[39:32]. LASTPAGE is also treated as invalid at the completion of the transfer so that a new access 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. In general, LASTPAGE is treated as invalid anytime the row-time access code has to change.

10.5.7 Bus Size

The **BS(1:0)** bits determine the bus size of the memory or peripheral corresponding to the configuration cache entry. The TC supports bus sizes of 8, 16, 32, and 64 bits, as shown in Table 10–8.

Table 10–8. Bus Size Selection

BS(1:0)	Bus SIze
0 0	8 bits
0 1	16 bits
1 0	32 bits
1 1	64 bits

The bus size determines the maximum number of bytes which the TC can transfer during each column access. If the number of bytes needed to be transferred exceeds the bus size, the TC automatically performs multiple accesses to complete the transfer. The bus size also determines which part of the address/data bus will be used for the transfer as shown in Table 10–9. Bus sizing effects are explained in detail in Section 12.1, *Big- and Little-Endian Operation*.

Table 10–9. AD[63:0] Bus Usage

	AD Pins Used for Transfer	
BS(1:0)	**Big Endian**	**Little Endian**
0 0	AD[63:56]	AD[7:0]
0 1	AD[63:48]	AD[15:0]
1 0	AD[63:32]	AD[31:0]
1 1	AD[63:0]	AD[63:0]

No matter what bus size is selected, the TC always aligns data to the proper portion of the bus and activates the appropriate CAS/DQM strobes to ensure that only valid data bytes get transferred.

The **BS(1:0)** bits are also used in conjunction with the **AS(2:0)** bits to determine row/column address shifting and in conjunction with the **PS(3:0)** bits to determine the page size.

10.5.8 Column-Time Wait State Insertion

When accessing slow peripheral devices, it may be necessary to extend the data subcycle to provide additional data access or write time. The **WS(1:0)** bits may be used to insert a predefined number of wait states into each column access for the addressed memory bank. Column-time wait states are only supported for 2 and 3 cycle per column accesses.

Table 10–10 shows the **WS(1:0)** encoding and the resulting wait states. The number and location of wait states is dependent on the selected cycle timing (as specified by the cache entry's **CT(3:0)** bits). For two and three cycle/column DRAM accesses, one or two wait states may be added when CAS/DQM is low to increase access time. One wait state can also be added when CAS/DQM is high to increase CAS precharge time. For two and three cycle per column SRAM accesses, up to three wait states may be inserted when CAS/DQM is low to provide more access time.

The **WS(1:0)** bits will be ignored for any other cycle timing. Setting **WS(1:0)** bits in conjunction with **CT(3:0)** values other than those shown in Table 10–10 will have no affect.

Table 10–10. Column-Time Wait State Selection

CT(3:0)	WS(1:0)	Additional Cycle Location
0 0 1 X	0 0	None
(DRAM)	0 1	1 CAS low
	1 0	1 CAS low, 1 CAS high
	1 1	2 CAS low, 1 CAS high
0 1 1 X	0 0	None
(SRAM)	0 1	1 CAS low
	1 0	2 CAS low
	1 1	3 CAS low

Column-time wait states may also be inserted by using the READY input. The insertion of automatic wait states using the **WS(1:0)** bits does not affect READY operation. READY may be used to insert wait states in addition to those generated by the WS field.

10.5.9 Block-Write Support

The **BW(1:0)** bits specify the block-write mode to be used with the current memory access. Packet transfers can specify block-write for fast pattern fills to take advantage of VRAMs' block-write feature. Because the TC uses the **BW(1:0)** bits to determine how to perform the block-write, software may specify block-write mode without having to know which type is supported by the addressed memory. Table 10–11 shows the available block-write modes.

Table 10–11. Block-Write Mode Selection

BW(1:0)	Block Write Mode
0 0	Simulated
0 1	Reserved
1 0	4x
1 1	8x

10.5.10Row-Time Wait State Insertion

The TMS320C82's memory cycles are designed to match target memory device timing requirements as closely as possible. However, certain combinations of memory device speeds and operating frequencies may require the addition of wait states in order to meet timing requirements. Setting the **R** bit in the configuration cache causes the automatic addition of a single wait state to row time of the external memory access. The placement of the extra state is dependent upon the cycle timing selected for the access as shown in Table 10–12. For DRAM and SRAM access cycles, an additional state is inserted with RAS high (between ad2 and rl1 states) to provide additional RAS decode time. For SDRAM access cycles, an additional state is inserted between the ac2 state and the column pipeline to increase the time between ACTV and READ/WRT commands.

Table 10–12. Row Time Wait State Selection

R	CT(3:0)	Additional Cycle Location
0	X X X X	None
1	0 0 X X	Between ad2 and rl1 states
1	0 1 X X	Between ad2 and rl1 states
1	1 X X X	Between ac2 and 1st column access

10.5.11Memory Bank Size

In most cases, a memory bank will correspond to a range of addresses. The locations within this range may address a single device or multiple devices of the same type.

The memory bank size field (**MS(4:0)**) determines the number of contiguous bytes to which the cache entry pertains. Any memory cycle to an address that falls within the address range defined by **MS(4:0)** will use the cycle configuration defined by the cache entry. Each memory bank must be aligned to its bank size. In other words, a 64-byte memory bank must begin and end on 64-byte address boundaries.

The value in **MS(4:0)** is used by the TC to generate an internal 32-bit address comparison mask. Whenever a new memory access begins, its address is compared to the tag addresses of the valid cache entries. The address comparison mask is applied to this comparison to determine which address bits should be compared. When all the unmasked bits (mask 1's) of the address stored in a valid cache entry match the corresponding bits of the address of the external memory access, a cache hit has occurred. The memory cycle will proceed using the configuration from the cache entry that generated the hit.

No internal mechanism exists for ensuring that memory banks do not overlap. The TC will use the configuration from the first entry which causes a cache hit. The MS field for each cache entry should be programmed to ensure that each valid off-chip memory address falls within the address range of only one memory bank.

Table 10–13 shows the masks generated for each **MS(4:0)** value, the addressed bits that will be compared, and the resulting memory bank size. If

MS(4:0) is set to all zeros then every external access will generate a cache hit on that entry resulting in a memory system of only one bank.

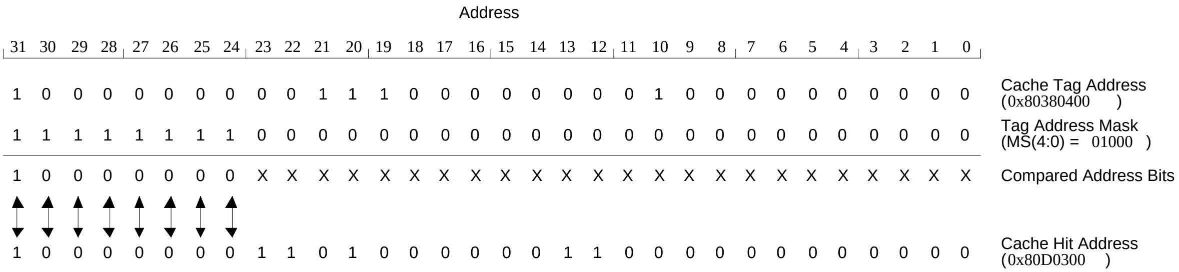
Table 10–13. Memory Bank Size

MS(4:0)	Mask	Address bits compared	Bank size (bytes)
0 0 0 0 0	0000 0000 0000 0000 0000 0000 0000 0000	None	4G
0 0 0 0 1	1000 0000 0000 0000 0000 0000 0000 0000	A[31]	2G
0 0 0 1 0	1100 0000 0000 0000 0000 0000 0000 0000	A[31:30]	1G
0 0 0 1 1	1110 0000 0000 0000 0000 0000 0000 0000	A[31:29]	512M
0 0 1 0 0	1111 0000 0000 0000 0000 0000 0000 0000	A[31:28]	256M
0 0 1 0 1	1111 1000 0000 0000 0000 0000 0000 0000	A[31:27]	128M
0 0 1 1 0	1111 1100 0000 0000 0000 0000 0000 0000	A[31:26]	64M
0 0 1 1 1	1111 1110 0000 0000 0000 0000 0000 0000	A[31:25]	32M
0 1 0 0 0	1111 1111 0000 0000 0000 0000 0000 0000	A[31:24]	16M
0 1 0 0 1	1111 1111 1000 0000 0000 0000 0000 0000	A[31:23]	8M
0 1 0 1 0	1111 1111 1100 0000 0000 0000 0000 0000	A[31:22]	4M
0 1 0 1 1	1111 1111 1110 0000 0000 0000 0000 0000	A[31:21]	2M
0 1 1 0 0	1111 1111 1111 0000 0000 0000 0000 0000	A[31:20]	1M
0 1 1 0 1	1111 1111 1111 1000 0000 0000 0000 0000	A[31:19]	512K
0 1 1 1 0	1111 1111 1111 1100 0000 0000 0000 0000	A[31:18]	256K
0 1 1 1 1	1111 1111 1111 1110 0000 0000 0000 0000	A[31:17]	128K
1 0 0 0 0	1111 1111 1111 1111 0000 0000 0000 0000	A[31:16]	64K
1 0 0 0 1	1111 1111 1111 1111 1000 0000 0000 0000	A[31:15]	32K
1 0 0 1 0	1111 1111 1111 1111 1100 0000 0000 0000	A[31:14]	16K
1 0 0 1 1	1111 1111 1111 1111 1110 0000 0000 0000	A[31:13]	8K
1 0 1 0 0	1111 1111 1111 1111 1111 0000 0000 0000	A[31:12]	4K
1 0 1 0 1	1111 1111 1111 1111 1111 1000 0000 0000	A[31:11]	2K
1 0 1 1 0	1111 1111 1111 1111 1111 1100 0000 0000	A[31:10]	1K
1 0 1 1 1	1111 1111 1111 1111 1111 1110 0000 0000	A[31:9]	512
1 1 0 0 0	1111 1111 1111 1111 1111 1111 0000 0000	A[31:8]	256
1 1 0 0 1	1111 1111 1111 1111 1111 1111 1000 0000	A[31:7]	128
1 1 0 1 0	1111 1111 1111 1111 1111 1111 1100 0000	A[31:6]	64
1 1 0 1 1	1111 1111 1111 1111 1111 1111 1110 0000	A[31:5]	32
1 1 1 0 0	1111 1111 1111 1111 1111 1111 1111 0000	A[31:4]	16
1 1 1 0 1	1111 1111 1111 1111 1111 1111 1111 1000	A[31:3]	8
1 1 1 1 0	1111 1111 1111 1111 1111 1111 1111 1100	A[31:2]	4
1 1 1 1 1	1111 1111 1111 1111 1111 1111 1111 1110	A[31:1]	2

As an example, consider a system which has two 8 Mbyte DRAM banks mapped contiguously. If the first bank begins at address `0x80000000`, the second bank would start at address `0x80800000` and end at address `0x80FFFFFF`. For configuration purposes, the DRAM can be considered to be a single 16 Mbyte bank. Thus **MS(4:0)** should be set to ‘`01000`’.

If the first access within the DRAM bank occurs at address `0x80380400`, then this address will be loaded into the cycle configuration cache tag address for the cache entry. On a later access to, for example, `0x80D03000`, the address will be compared to the masked cache tag addresses. In this case **MS(4:0)** mask off cache tag address bits 23:0 for this entry. Because the unmasked address bits (A(31:24)) of the tag address match the corresponding bits of the access, the comparison will yield a cache hit and this cache entry will be used. The address masking and comparison for this example are shown in Figure 10–6.

Figure 10–6. Memory Bank Size Address Masking



Address

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	
1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	Cache Tag Address (0x80380400)
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tag Address Mask (MS(4:0) = 01000)
1	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Compared Address Bits
1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	Cache Hit Address (0x80D0300)

10.5.12 Priority Level

The priority level of each cycle configuration cache entry is determined by the value of the cache entry’s **PL(1:0)** bits. The priority level may bet set to low, medium, or high, as shown in Table 10–14. When a cycle configuration cache miss requires that an entry be replaced, the least recently used entry of the lowest priority level is selected for replacement.

Table 10–14. Priority Level Codes

PL(1:0)	Priority Level
0 0	Low
0 1	Medium
1 0	Reserved
1 1	High

Chapter 11

Generating Memory Cycles

The TC contains an external memory controller that generates external memory cycles. This controller contains a state machine that generates a sequence of states to control the behavior of the memory interface signals. This chapter describes how memory cycles are generated and controlled.

Topics in this chapter include:

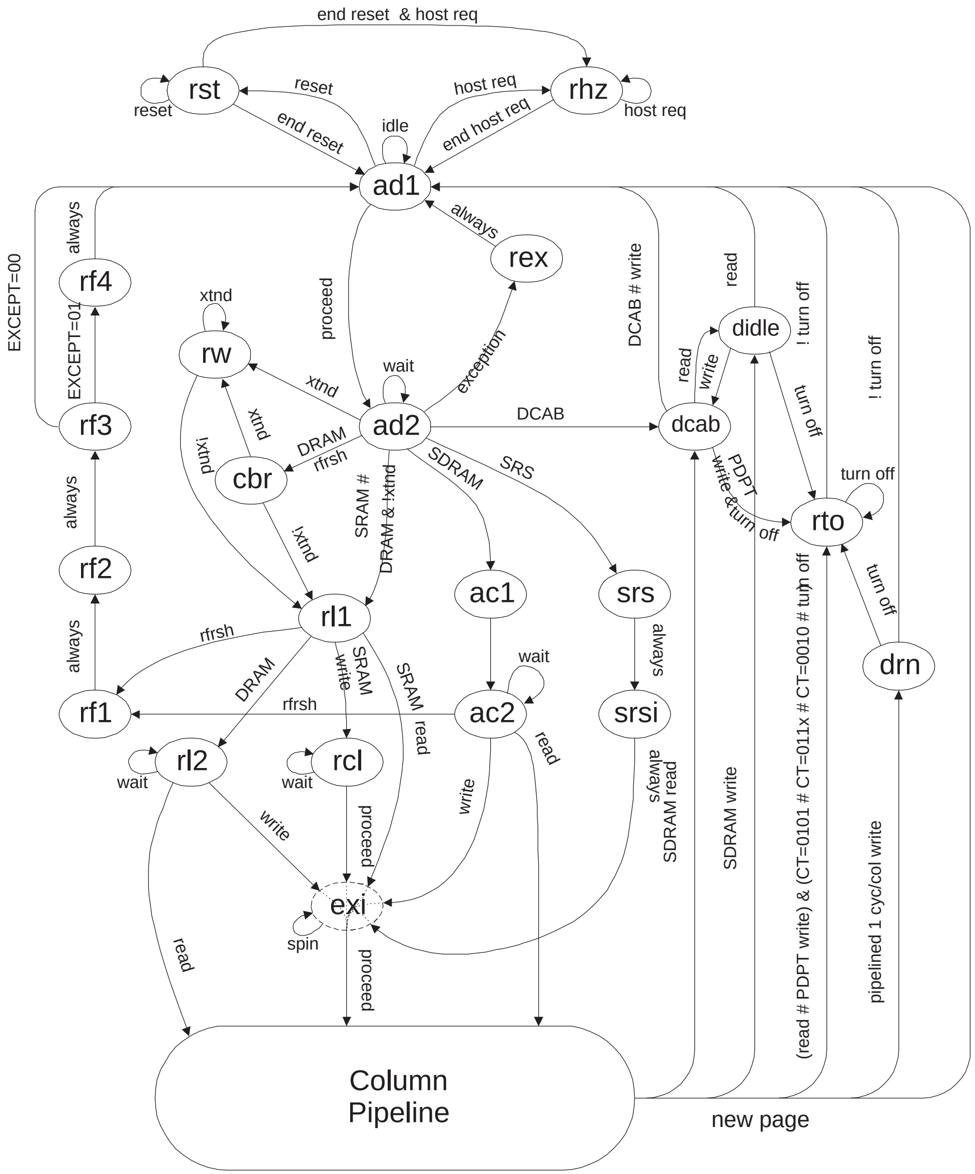
Topic **Page**

11.1 Overview of Memory States

All ‘C82 external memory cycles are generated by the TC’s external memory controller. This controller contains a state machine which generates a sequence of states to determine the transitions of the memory interface signals. The sequence of generated states varies based on the type of cycle being performed, the configuration of the addressed memory, the next access to be performed, and internal or external events (retries, faults, etc.).

Figure 11–1 shows the complete state diagram for the external memory controller. Although there are many different states and state transitions, the state sequence is basically determined by the type of access being performed and the cycle timing configuration for the addressed memory bank. The operation of the external memory controller is most easily understood by referencing the state sequences contained in the individual cycle description sections. The memory interface consists row-time states and the column pipeline.

Figure 11–1. External Memory Interface State Diagram



11.1.1 Row-Time States

The row-time states make up the address subcycle (or **row time**) of each memory access. They occur whenever a new page access begins. Each type of memory access has a predefined row time sequence. However, this sequence can be modified to meet the timing requirements of the accessed device. The modification is achieved either automatically by the TC, by the values programmed into the cache entry, or by external hardware. The row-time states are defined as follows:

State	Description
ad1	**Address 1.** The beginning state of every memory access. The row address is output on AD[31:0] and the access type code on AD[39:32]. All control signals are driven to their inactive levels. This state is repeated if the memory interface is idle (there is no access to begin).
ad2	**Address 2.** The second state of every memory access. Asserts RL low and drives DDIN according to the data transfer direction. The READY and EXCEPT[1:0] (if exceptions are enabled for the bank) inputs are sampled. This state is repeated if READY is sampled low.
rl1	**RAS Low 1.** Common to DRAM and SRAM accesses. RAS is activated, and TRG/CAS, W, DBEN, and DSF are driven to their appropriate row-time levels.
rl2	**RAS Low 2.** Common to all DRAM non-refresh cycles. No signal transitions occur. This state provides necessary RAS low time prior to CAS activation. READY is sampled and if low, the state is repeated.
rcl	**Column Load.** This state occurs during SRAM write cycles. No signal transitions occur. The state is inserted to allow the TC time to load the column pipeline. READY is sampled and if low, the state is repeated.
ac1	**Activate 1.** Common to SDRAM cycles. The SDRAM activate (ACTV) command is performed during this state. DBEN is activated except for peripheral device packet transfers.
ac2	**Activate 2.** Common to all SDRAM cycles. This state is generated to ensure sufficient time between the ACTV command and subsequent commands. No signal transitions occur. READY is sampled and if low, the state is repeated.
exi	**Exception Idle.** This state is inserted for certain access types when exceptions are enabled. No signal transitions occur. The state is inserted to allow the TC time to load the column pipeline after the sampled exception inputs have been evaluated. This state is not present when exceptions are not enabled or on cycles that don’t require the extra exception evaluation time.
rex	**Row-Time Exception.** This state occurs when a row-time exception is sampled on the EXCEPT[1:0] inputs. The RAS output is kept high preventing the external memory cycle from beginning. This serves as a transition state back to ad1.

State	Description
rw	**RAS High Wait.** A special state that occurs during DRAM accesses. This state is inserted as necessary to ensure that sufficient RAS high time is provided prior to a DRAM access. No signal transitions occur.
cbr	**CAS Before RAS.** During DRAM refresh cycles, this state is inserted to activate the CAS/DQM[7:0] outputs prior to the fall of RAS.
srs	**Special Register Set.** This state only occurs during SDRAM SRS cycles. The DBEN output is activated but no other signal transitions occur.
srsi	**SRS Idle.** A special idle cycle which always follows the srs state to allow the TC to load the column pipeline. No signal transitions occur.
rf1	**Refresh 1.** Common to all DRAM refresh cycles. No signal transitions occur.
rf2	**Refresh 2.** Common to all DRAM refresh cycles. No signal transitions occur.
rf3	**Refresh 3.** Common to all DRAM refresh cycles. No signal transitions occur.
rf4	**Refresh 4.** Occurs for slow DRAM refresh cycles (EXCEPT[1:0] = ‘01’). No signal transitions occur. Provides an extra cycle of RAS low time.
dcab	**Deactivate All Banks.** This state occurs at the end of an SDRAM access. RAS and W are activated to perform the DCAB command.
didle	**DCAB Idle.** This special idle state is inserted either before or after the dcab state to properly align the DCAB timing relative to other SDRAM commands.
drn	**Drain.** Used to drain the memory device pipeline at the end of pipelined 1 cyc/col DRAM writes. All CAS/DQM outputs are activated to complete the last write of data into the memory array.
rto	**Read Turn Off.** Special state to allow memory devices to turn off before a new address is output on AD[31:0]. It is automatically added whenever there is less than one clock cycle between when data is sampled and a new page access (ad1 state) begins. Additional rto states are added as specified by the value of **TO(1:0)** in the configuration cache entry. All control signals except RL are driven to their inactive values.
rhz	**High Impedance.** This state is entered when the ‘C82 releases control of the local memory bus in response to a host request. This state is repeated until the host request is removed. All ‘C82 memory interface signals except CLKOUT, REQ and HACK are placed in high impedance.
rst	**Reset.** The reset state of the ‘C82 external memory interface. This state is entered when RESET is active. All memory interface signals are placed in high impedance (except CLKOUT, HACK, and REQ).

11.1.2 Column-Time Pipeline

The column-time pipeline makes up the data subcycle (or **column time**) of each memory access. During these states the transfer of data occurs (if required). There are a number of different pipeline flow sequences depending on the configured column timing and whether the access is a read or write. During page-mode operation, multiple column accesses are performed so the pipeline sequence may be repeated many times over the course of a single page-mode access. The column-time pipeline is unique for each type of memory access and is explained in detail in the subsequent cycle description chapters.

11.1.3 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 a certain state.

Transition	Description
always	Transition is always made after completion of the current state.
idle	Memory interface is idle. No access to perform.
proceed	Continuation of current cycle.
xtnd	Extend RAS high time to meet minimum requirement
rfrsh	Refresh cycle.
wait	READY was sampled low. Repeat the state until READY is sampled high.
exception	An exception (e.g. fault or retry) was sampled on the EXCEPT[1:0] inputs in state ad2. Abort the current access.
CT=xxxx	State change occurs for configuration cache entry **CT(3:0)** value indicated
EXCEPT=xx	State change occurs for indicated value sampled during state ad2 on the EXCEPT[1:0] inputs.
spin	Internally generated condition. Repeat current state until the column pipeline is loaded.
turn off	Insert extra cycle to allow memory device outputs to turn off. This could occur automatically or as a result of turn off cycles programmed into **TO(1:0)** of the configuration cache entry.
new page	The column pipeline has ended because a new page must be started.
PDPT	Cycle is part of a peripheral device packet transfer.
hostreq	A host request has occurred (HREQ low) and the ‘C82 is ready to release the memory bus.
reset	RESET pin is sampled as active low.

11.2 Transfer Controller Pipelines

The TC contains pipelines in its internal and external memory interfaces. Memory accesses are queued in the pipelines when the current access is not yet completed. For example, if a packet transfer is transferring data from on-chip to off-chip memory, the off-chip dst may require two cycles per access. Since data can be extracted from the PT FIFO on every cycle (assuming the FIFO contains enough data), another dst access will be placed in the external interface pipeline before the first access has completed.

In order for an access to be loaded into the external memory pipeline, it must belong to the same memory page as the other accesses that are already in the pipeline. Once an access has been placed in the pipeline, it cannot be removed; the access must occur. When an access to a new memory page is requested, the cycles queued in the pipeline will first be completed before beginning the new page access. This is referred to as pipeline draining.

If, for example, the TC is performing a packet transfer to external memory and the TC's refresh controller issues an urgent refresh request, any (column) accesses currently in the pipeline must be completed before the urgent refresh cycle can occur.

The external memory interface pipeline is transparent to the user. Its effect on operation can only be seen when the pipeline is drained. Pipeline draining can occur prior to cache/DMA accesses, host accesses, and urgent refreshes, during packet transfer suspension, and after hardware-initiated new page requests. Therefore, pipeline draining has the highest priority when an urgent request is pending. This is described in Section 2.3, *Crossbar Priority*.

11.2.1 Pipeline Bubbles

During external memory accesses, there may be cycles of inactivity during which no active column accesses occur. This is due to the **bubbles** within the TC's internal pipeline. Bubbles occur when the TC performs no operation during a memory cycle. This can occur as a result of crossbar contention, because the TC's packet transfer FIFO contains an insufficient amount of data for the TC to perform the next access (write cycles), because the TC's FIFO is full and no more data can be loaded (read cycles), or simply because no activity is requested. If none of these conditions occur, no bubbles will occur.

An example of a bubble is when the TC is servicing a packet transfer with an internal source (src) and an external destination (dst). If the packet transfer is set up such that the TC can only access src data one byte at a time (src A Count = `0x01`) but can write dst data eight bytes at a time (dst A Count > `0x08`), then the TC is able to perform only one dst access for every eight internal cycles. This results in the TC inserting bubbles into the external memory cycles while it waits for enough src data (eight bytes) to perform the next column access. The number of bubble cycles inserted depends on the timing of the external memory cycles.

11.3 Ending an External Memory Cycle

In order to support a wide range of memory types and peripherals, the TMS320C82 provides a variety of methods for extending, retrying, and terminating memory cycles. They are described in the following sections.

11.3.1 Normal Termination

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

Note that 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.

11.3.2 Wait States

The TMS320C82 allows the insertion of wait states in order to extend the duration of a memory cycle. This is achieved through the use of the READY input pin. The READY input is sampled at appropriate times (as determined by the memory timing) during row and column accesses. If READY is sampled high, the memory access continues. If READY is sampled low, the memory access stalls and the current machine state is repeated until READY is again sampled high.

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 access that cannot be completed in a reasonable time should be faulted or retried to prevent locking out high priority external memory accesses that may be waiting to be serviced.

There are two types of wait states that can be inserted into a memory access:

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

Row-time Wait States: Row-time wait states are used to extend the memory access during the row-time states. The READY input is always sampled at the beginning of the ad2 state. This allows the system to decode the row address and status and determine if the addressed device needs additional access or decode time prior to the fall of RAS. If READY is low, the ad2 state is repeated until READY is sampled high.

For DRAM cycles, the READY input is sampled again at the start of the rl2 state after RAS has fallen. Wait states may be inserted here to provide additional RAS access time. For SDRAM cycles, the READY input is sampled after the ACTV (activate) command at the start of the ac2 state. Wait states can be inserted here to allow for additional time between SDRAM activate and read or write commands.

A single row time wait state can also be inserted automatically by setting the **R** bit in the configuration cache entry. (See Section 10.5.10, *Row-Time Wait State Insertion.*) Wait states generated using the READY input will occur in addition to the programmed wait state.

Column-time Wait States: Column-time wait-states are used to stall the TC's column pipeline. These wait states are only supported for 2 and 3 cycle/column DRAM and SRAM accesses. The READY input is sampled at different stages of the column access depending on the selected cycle timing. For more information, see the wait state sections of the 2 and 3 cycle/column DRAM and SRAM timings.

Column time wait states can also be inserted automatically by setting the **WS(1:0)** bits in the configuration cache entry. These programmable wait states allow each column access to be extended by up to three clock cycles. (See Section 10.5.8, *Column-Time Wait State Insertion)* This allows the 'C82 to interface to devices with long access times without the need for external hardware to generate wait states. Any column-time wait states generated using the READY input will occur in addition to the programmed wait states.

11.4 Memory Exceptions

Support for exceptions slows down the memory interface for the faster memory types, because the TC cannot load accesses into the column pipeline until after it has had time to sample and evaluate the exception inputs (EXCEPT[1:0]). To ensure that this delay occurs only when necessary, the enable exceptions bit (**E**) in the memory configuration cache entry is used to indicate whether the bank requires exception support. Thus, exception support may be enabled or disabled on a bank by bank basis. When the **E** bit is set for a bank, additional row-time states will be added to its memory accesses as needed to enable sampling of the exception inputs.

Table 11–1 shows the available row-time exception codes. The EXCEPT[1:0] inputs are sampled during the ad2 state of each row access (if the cache entry's **E** bit is set).

Table 11–1. Row-Time Exception Encoding

$\overline{\text{EXCEPT}}$[1:0]	**Exception**
0 0	Flush Cache Entry
0 1	Fault
1 0	Retry
1 1	None

The EXCEPT[1:0] inputs have higher priority than the READY input. This means that exceptions take precedence over wait states. If an exception is generated at the same time as a wait state, the exception will occur. If no exception is detected but a wait state occurs (READY low), then the EXCEPT[1:0] inputs (and READY) will be resampled on the next cycle.

The **E** bit has no effect on cache configuration accesses. Exceptions are always supported for these cycles. The flush cache entry code (‘00’) is not supported for cache configuration cycles since the cache entry has already been flushed.

The EXCEPT[1:0] inputs are sampled on the falling edge of CLKOUT on every cycle following the rl1 state to allow external logic to request a new page access. Only values of ‘11’ and ‘10’ are valid during this column-time sampling as shown in Table 11–2. The value of the **E** bit has no effect on column-time exception sampling.

Table 11–2. Column-Time Exception Codes

$\overline{\text{EXCEPT}}$[1:0]	Exception
0 0	Invalid
0 1	Invalid
1 0	New Page Request
1 1	None

11.4.1 Flush Cache Entry

The flush cache entry exception is generated by driving the EXCEPT[1:0] inputs to ‘00’ during the ad2 state. This exception forces the cycle configuration cache entry corresponding to the current access to be flushed. The current access terminates and a cache configuration access will occur to reload the bank’s cache entry. After completion of the cache configuration the access that was originally terminated by the exception will be restarted. The flush cache entry feature is especially useful for shadow mapping two different types of memory to the same address.

11.4.2 Retry

The retry mechanism allows external logic to signal that the current access cannot be completed and needs to be tried again at a later time. A retry is generated by driving the EXCEPT[1:0] inputs with a ‘10’ value during the ad2 state.

A retry causes the TC to terminate the current row access. The TC then reevaluates all pending requests. If no higher priority request is pending, then the retried cycle will restart. If the TC has higher priority requests pending then those requests will be serviced first and then the retried cycle will be restarted.

If a retry occurs during a packet transfer service and another packet transfer of equal or higher priority is pending, then the PT which generated the retry will be suspended (assuming PTMIN has been satisfied). When the suspended transfer reaches its turn in the appropriate round-robin priority chain it will be resubmitted and the transfer will resume by restarting the retried access.

Retries during DRAM and SDRAM refreshes will not cause the memory cycle to be restarted. However, the refresh pseudo-address is not decremented so that the next refresh cycle will use the same pseudo-address. Retries during SDRAM Mode Register Set (MRS) cycles also have no effect on the cycle timing. Instead, they cause another MRS cycle to be generated immediately following the retried MRS.

11.4.3 New Page Request

Once a memory access is loaded in the TC's column pipeline it cannot be retried. However, the TC does support a column-time new page request.

When a new page request occurs, the TC completes all pending column accesses in its pipeline and terminates the current page-mode access. Thus the next access will begin with an address subcycle (row access). Since the TC's column pipeline must be drained, a number of column accesses will occur after the new page request is made. These column accesses will **not** be repeated after the new row access occurs.

A new page request will occur anytime that the EXCEPT[1:0] pins are sampled as '10' after the rl1 state. The current page-mode access will then terminate upon completion of all pending column accesses. Typically the EXCEPT[1:0] inputs will be held at the '10' level until the new row access begins but this is not required.

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

11.5 Faults

If a system error prevents a memory access from being completed, the system can signal the ‘C82 by faulting the memory cycle. The fault will generate an interrupt to the MP allowing it to correct the error before the cycle is retried. External memory faults are generated at row time by driving the EXCEPT[1:0] pins to the value ‘01’ when they are sampled during the ad2 state. The fault mechanism varies depending on the type of access that was being performed when the fault occurred.

11.5.1 Fault Support

The fault support for the different types of memory accesses are defined below.

- ❏ **SRT Cycles -** Faulting not supported. Fault code on EXCEPT[1:0] inputs is ignored.
- ❏ **Refresh Cycles** - Faulting not supported. Fault code on EXCEPT[1:0] inputs is ignored.
- ❏ **PP Data Cache/DEA Request -** Faulting supported. The PP will remain stalled until the fault is cleared. Requests from the other PP will continue to be serviced.
- ❏ **MP Instruction/Data Cache Request -** Faulting supported. The faulted cache request is immediately canceled and the MP receives a memory fault interrupt. The other cache request, if pending, will still be serviced.
- ❏ **MP DEA Request -** Faulting supported. The DEA request is immediately canceled and the MP is sent a memory fault interrupt.
- ❏ **MP/PP Packet Transfer** - Faulting supported. The packet transfer is suspended and its state is saved in the requesting processor’s parameter RAM. Packet transfer requests from other processors will continue to be serviced by the TC.
- ❏ **XPT -** Faulting supported. The packet transfer will terminate immediately but will not be suspended since there is no suspend area assigned to XPTs. Subsequent XPT requests are blocked until the fault is cleared.

11.5.2 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 (see Figure 4–2, *PP Parameter RAM Contents*). The appropriate **PCx** bit in the **FLTSTS** register is set, causing an **mf** interrupt to the MP. The MP can then examine the parameter RAM to determine the faulted address.

The address which caused the cache/DEA fault can not be changed. These faults can only be corrected if they are due to an external system condition or event (such as a system resource that is temporarily unavailable). If the MP can correct the fault, it clears the bit in the **FLTSTS** register and the request is rescheduled. If the fault can not be corrected, the MP can send the PP a reset request so that the cache service request is aborted.

The PP does not know when a fault occurs during its cache/DEA request. It only knows that its request has not yet been completed (and remains stalled). The MP must correct the fault or reset the PP.

11.5.3 MP Cache/DEA Faults

If an MP requested cache fill or DEA request faults, the request is immediately canceled and the MP is sent a memory fault interrupt (**mf**). The appropriate bit is set in the MP's **FLTOP** register (**d** for data cache fault or **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 of the *TMS320C8x Master Processor User's Guide* for more information.

11.5.4 Packet Transfer Faults

When a fault occurs during an MP or PP packet transfer, the transfer is suspended and its state is saved in the suspend area of the requesting processor's parameter RAM. If the transfer was 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 packet transfer status (**PTS**) bits in the saved PT Options field are modified to indicate that the fault has occurred and whether it was on the src or dst.

Once the parameters have been saved, the TC sets the appropriate bit its **FLTSTS** register (see Section 1.3.4, *The FLTSTS Register*) which indicates which processor's packet transfer was faulted. The TC also sets the **mf** bit in the MP's **INTPEN** register which generates a memory fault interrupt to the MP. The MP can then read the **FLTSTS** register to find out which processor' PT 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.

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

Packet transfer faults are typically caused by a bad parameter table value (such as an invalid address) or by an external system condition (such as an unavailable system bus). Bad parameters can be corrected by changing their value within the suspended PT parameter area. If the MP can correct the fault, it then clears the bit in the **FLTSTS** register and the PT request is automatically resubmitted. Once the faulted transfer gets its turn in the round robin priority scheme, its internal state is restored from the suspend area of the parameter RAM and the TC continues the packet transfer starting with the faulted access.

When a fault occurs during an XPT, the transfer is immediately terminated. The transfer is not suspended, so its state is not saved. The TC will set the appropriate **XPT** bits in **FLTSTS**, and the MP will be interrupted with a memory fault. Subsequent XPT requests are blocked until the fault is cleared by the MP. This prevents system lock-up caused by the faulted XPT request being maintained on the $\overline{\text{XPT}}$[3:0] input pins.

11.5.5 On-chip Faults

Certain accesses to on-chip addresses can cause faults independent of the EXCEPT[1:0] inputs. These faults occur when an illegal on-chip access is attempted. The normal fault mechanism applies depending on the cycle being attempted. The on-chip faults include:

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

Chapter 12

Endianess and Bus Sizing

This chapter discusses how big-endian and little-endian operation affect the ordering of data on the external data bus. It also demonstrates the effects of different bus sizes on memory accesses and provides a comparison of transfers for all bus sizes using both endian modes.

Topics in this chapter include:

Topic **Page**

12.1 Big- and Little-Endian Operation

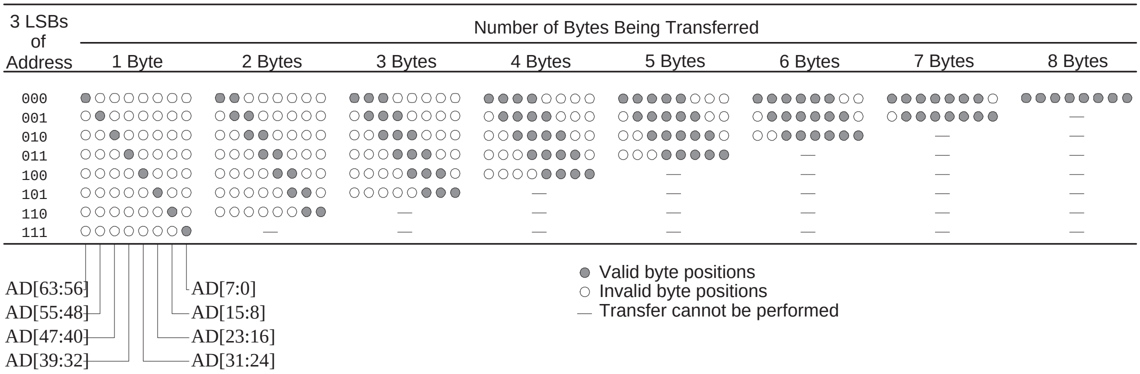
The TC accesses data in either big-endian or little-endian formats. The endian mode determines the way in which bytes are addressed. In big-endian mode, byte 0 is the leftmost byte in a word and successive bytes are numbered rightward. In little-endian mode, byte 0 is the right most byte in a word and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

12.1.1 External Transfers in Big- and Little-Endian Formats

The three LSBs of the memory address and the number of bytes to be transferred determine the positions of valid data bytes on the external data bus (AD[63:0]). These positions are shown in the following tables. The ● symbols represent valid byte positions and the ❍ symbols indicate invalid bytes. The – symbols indicate transfer operations that cannot be performed because the transfer size would be larger than the bus.

Figure 12–1 and Figure 12–2 show the byte positioning for various transfer sizes and addresses for 64-bit bus transfers. Up to 8 bytes of data may be transferred at a time.

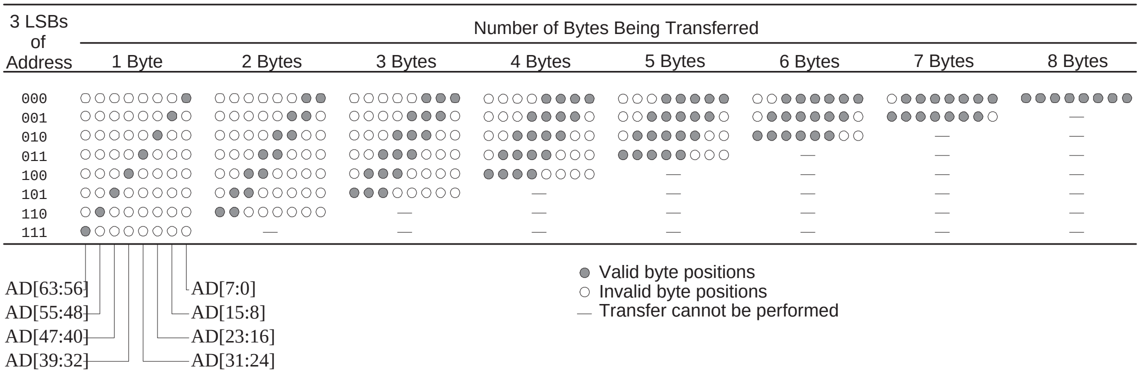
Figure 12–1. Big-Endian 64-bit Bus Byte Positions



3 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	○○○○○○○●	—	—	—	—	—	—	—

AD[63:56]
AD[55:48]
AD[47:40]
AD[39:32]
AD[7:0]
AD[15:8]
AD[23:16]
AD[31:24]
● Valid byte positions
○ Invalid byte positions
— Transfer cannot be performed

Figure 12–2. Little-Endian 64-bit Bus Byte Positions

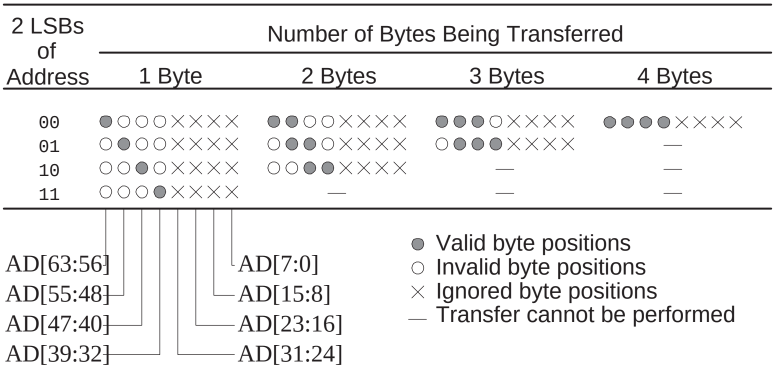

AD[63:56]
AD[55:48]
AD[47:40]
AD[39:32]
AD[7:0]
AD[15:8]
AD[23:16]
AD[31:24]

3 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
— Transfer cannot be performed

When the external bus is 32-bits wide, only bytes 0-3 of the bus are used for data transfer. For big-endian transfers this corresponds to AD[63:32] and for little-endian transfers it corresponds to AD[31:0]. The byte positions, based on the two LSBs of the address are shown in Figure 12–3 and Figure 12–4 for big and little endian, respectively. The × symbols indicate byte positions which are ignored.

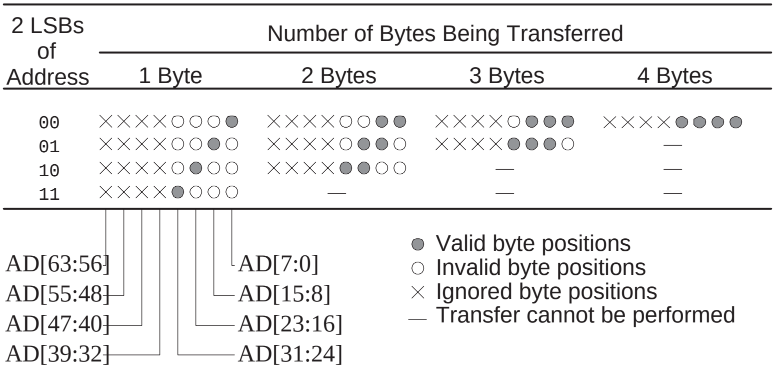
Figure 12–3. Big-Endian 32-Bit Bus Byte Positions


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

2 LSBs of Address	Number of Bytes Being Transferred			
	1 Byte	2 Bytes	3 Bytes	4 Bytes
00	●○○○××××	●●○○××××	●●●○××××	●●●●××××
01	○●○○××××	○●●○××××	○●●●××××	—
10	○○●○××××	○○●●××××	—	—
11	○○○●××××	—	—	—

● Valid byte positions
○ Invalid byte positions
× Ignored byte positions
— Transfer cannot be performed

Figure 12–4. Little-Endian 32-bit Bus Byte Positions

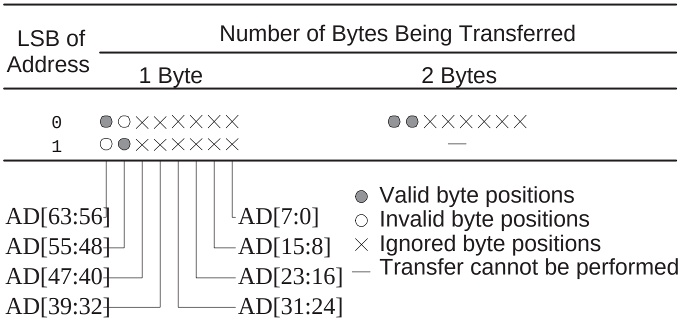


2 LSBs of Address	Number of Bytes Being Transferred			
	1 Byte	2 Bytes	3 Bytes	4 Bytes
00	××××○○○●	××××○○●●	××××○●●●	××××●●●●
01	××××○○●○	××××○●●○	××××●●●○	—
10	××××○●○○	××××●●○○	—	—
11	××××●○○○	—	—	—

● Valid byte positions
○ Invalid byte positions
× Ignored byte positions
— Transfer cannot be performed

When the external bus is 16 bits wide, only bytes 0-1of the bus are used for data transfer. For big-endian transfers this corresponds to AD[63:48] and for little-endian transfers it corresponds to AD[15:0]. The byte positions, based on the LSB of the address are shown in Figure 12–5 and Figure 12–6 for big and little endian, respectively.

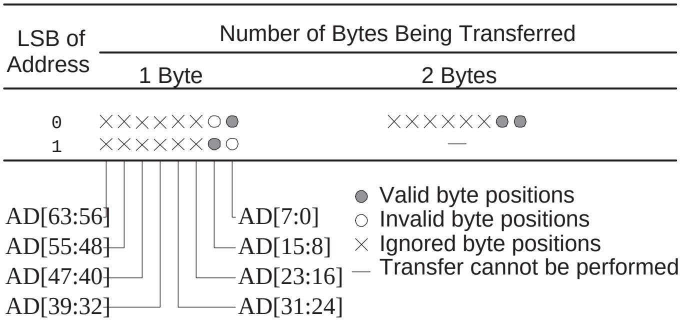
Figure 12–5. Big-Endian 16-Bit Bus Byte Positions



LSB of Address	Number of Bytes Being Transferred	
	1 Byte	2 Bytes
0	●○××××××	●●××××××
1	○●××××××	—

● Valid byte positions
○ Invalid byte positions
× Ignored byte positions
— Transfer cannot be performed

Figure 12–6. Little-Endian 16-Bit Bus Byte Positions



When the external bus is configured as 8-bits wide, only byte 0 is used for data transfer. This corresponds to AD[63:56] and AD[7:0] for big-endian and little-endian transfers respectively. The byte positions for 8-bit busses are shown in Figure 12–7 and Figure 12–8.

Figure 12–7. Big-Endian 8-Bit Bus Byte Positions

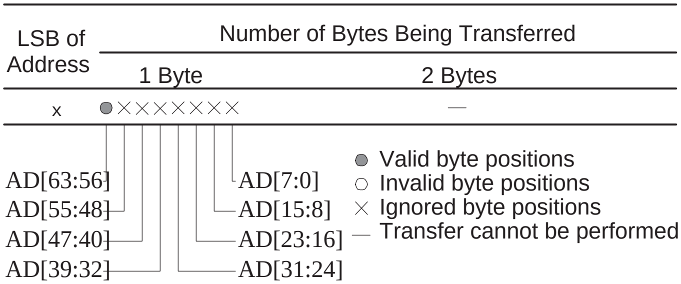
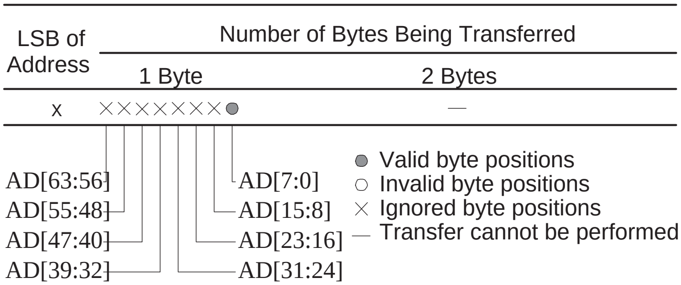


Figure 12–8. Little-Endian 8-Bit Bus Byte Positions



12.1.2 Selecting Endian Format

The endian format for the TMS320C82 is selected at reset using the READY input. The ‘C82 samples and latches the value of READY on the clock cycle just prior to the rising edge of the RESET input. If READY is sampled low then the ‘C82 will operate in big-endian mode until the next hardware reset occurs. If READY is sampled high, then the ‘C82 will operate in little-endian mode.

12.1.3 Packet Transfer Endian Considerations

As discussed in Chapter 5, *Short-Form Packet Transfers* and Chapter 6, *Long-Form Packet Transfers*, the packet transfer parameters are independent at the word (32-bit) level only. The TC always loads and stores packet transfer parameters using double-word (64-bit) transfers. The TC swaps 32-bit parameter words according to the selected endian format. The 16-bit fields within the words (such as **A Count** and **B Count**) are not swapped according to endian format because the word containing these values is considered to be a single 32-bit entity. Likewise, the 64-bit **Transparency/Color Register** value is always treated as a single 64-bit entity and its bytes are not swapped according to endian format.

12.2 Dynamic Bus Sizing

As discussed in Section 12.1.1 , the portion of the bus which transfers valid data during an off-chip memory access depends on the bus size and the current endian-mode. The bus size also determines the number of column accesses that are required to transfer a given number of bytes.

The following sections demonstrate the required column accesses for writing data to external memory. A common example shows a write of five bytes of data using a 64-bit, 32-bit, 16-bit, and 8-bit bus for both big-endian and little-endian formats. The five bytes are written starting at address `0x02000002` (that is, starting with byte 2 of a 64-bit double word in external memory). This type of transfer might occur as part of a packet transfer. Each example shows the internal byte address, the column address output on the RCA bus, the active CAS/DQM strobes, and the valid portions of the data bus. The 2 Cycle/Column DRAM timing is shown but the column accesses would occur in the same order for all other timings.

12.2.1 Big-Endian Bus Sizing

In big-endian mode, the bytes are addressed from left to right on the data bus, beginning with AD[63:56]. Busses less than 64-bits wide are connected to the upper portion of the data bus.

For a 64-bit data bus, the 5-byte write requires only one column access because the data size is less than the bus width. The write cycle accesses the 64-bit double word at address `0x02000000`. Because the write begins in byte 2 of the double word, the five data bytes are placed on the data bus beginning with AD[47:40]. Only CAS/DQM[5:1] are activated so that only the valid bytes are written. This is shown in Figure 12–9. A read access behaves identically except that all CAS/DQM strobes are active and the invalid bytes are discarded by the TC.

Figure 12–9. 64-Bit Big-Endian Transfer Example

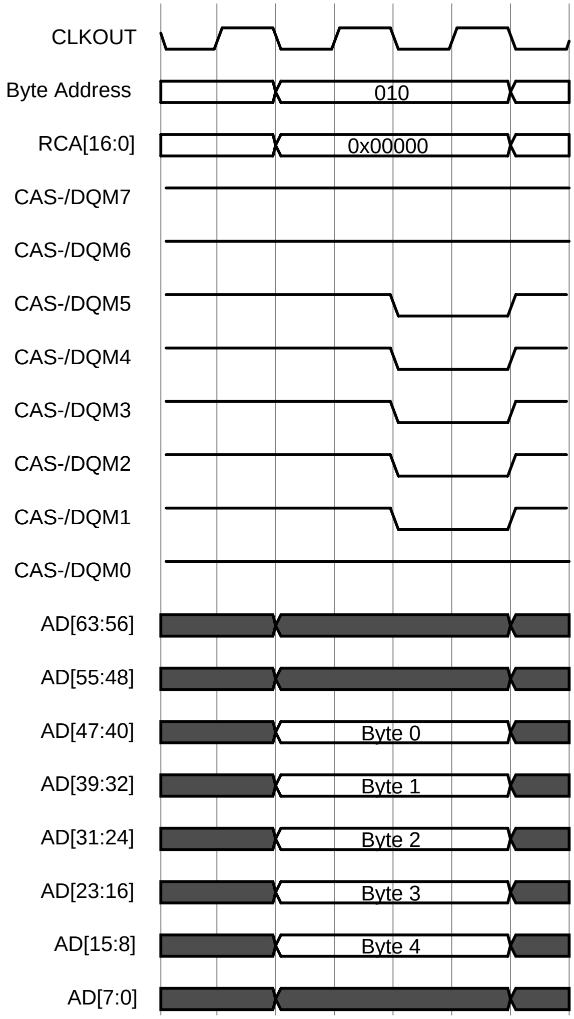
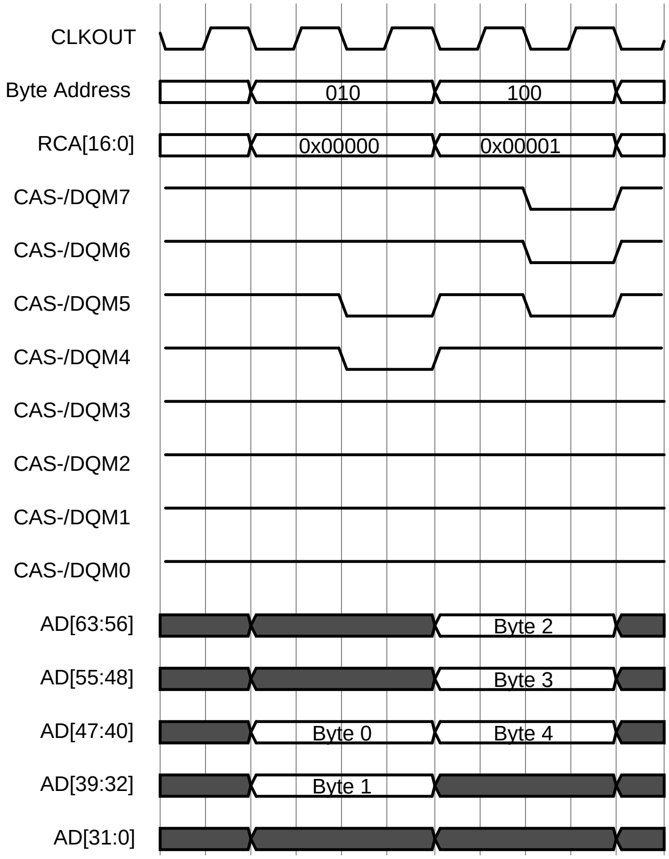


Figure 12–10 shows the big-endian write of the five bytes over a 32-bit data bus. Because the write crosses a 32-bit word boundary, two column accesses are required. The first access is to word `0x02000000`. Only the two MSbytes of the word are written so the data is placed on AD[47:32] and the CAS/DQM[5:4] strobes are activated. The second access to word `0x02000004` writes the last three bytes. These are written to the LSbytes of the word so AD[63:40] have valid data and CAS/DQM[7:5] are active. For a read access of the same bytes, the same column accesses occur, but all CAS/DQM strobes are active and the invalid bytes are discarded after being read by the TC.

Figure 12–10.32-Bit Big-Endian Transfer Example



The 5-byte write on a 16-bit bus requires three column accesses as shown in Figure 12–11. The first access is to halfword `0x02000002`. (An access to halfword `0x02000000` is not required as none of its bytes are being written.) Both bytes in the first halfword are written so that AD[63:48] are valid and CAS/DQM[7:6] are active. The second column access writes both bytes of halfword `0x02000004`. The final access is to halfword `0x02000006`. One byte is transferred on AD[63:56] so only CAS/DQM7 is active. For a read of the same bytes, the identical column accesses occur but all CAS/DQM strobes are active and the invalid bytes are discarded after being read by the TC.

Figure 12–11.16-Bit Big-Endian Transfer Example

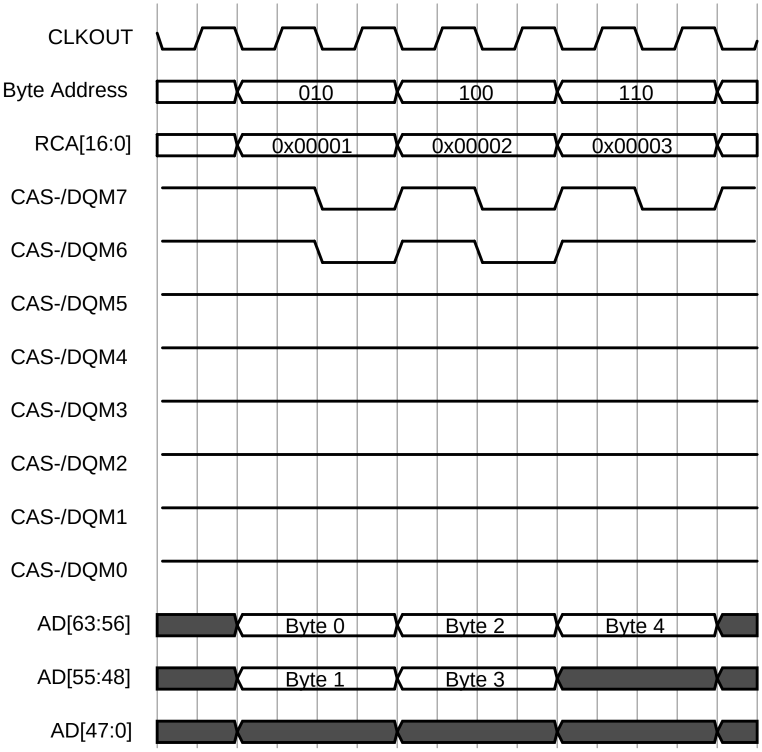
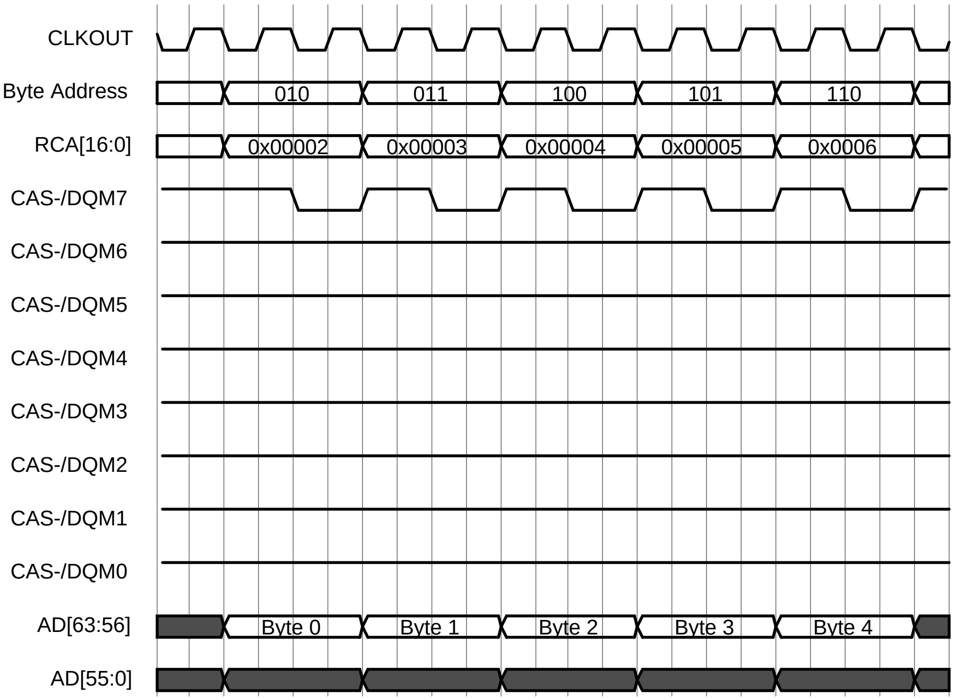


Figure 12–12 shows the big-endian write of the five bytes on an 8-bit bus. For this bus size, each column access transfers a single byte on AD[63:56]. Thus a total of five column accesses are required. The bytes accessed are `0x02000002`, `0x02000003`, `0x02000004`, `0x02000005`, and `0x02000006`. The CAS/DQM7 strobe is active for each column access. For a read access of the same five bytes, the identical column accesses occur, but all CAS/DQM[7:0] strobes are active and the invalid bytes on AD[55:0] are discarded after being read by the TC.

Figure 12–12.8-Bit Big-Endian Transfer Example



12.2.2 Little-Endian Bus Sizing

In little-endian mode, the bytes are addresses from right to left on the data bus, beginning with AD[7:0]. Busses less than 64-bits wide are connected to the lower portion of the data bus.

For a 64-bit data bus, the 5-byte write requires only one column access because the data size is less than the bus width. The write cycle accesses the 64-bit double word at address `0x02000000`. Because the write begins in byte 2 of the double word, the five data bytes are placed on the data bus beginning with AD[23:16]. Only CAS/DQM[6:2] are activated so that only the valid bytes are written. This is shown in Figure 12–13. A read access behaves identically except that all CAS/DQM strobes are active and the invalid bytes are discarded by the TC.

Figure 12–13.64-Bit Little-Endian Transfer Example

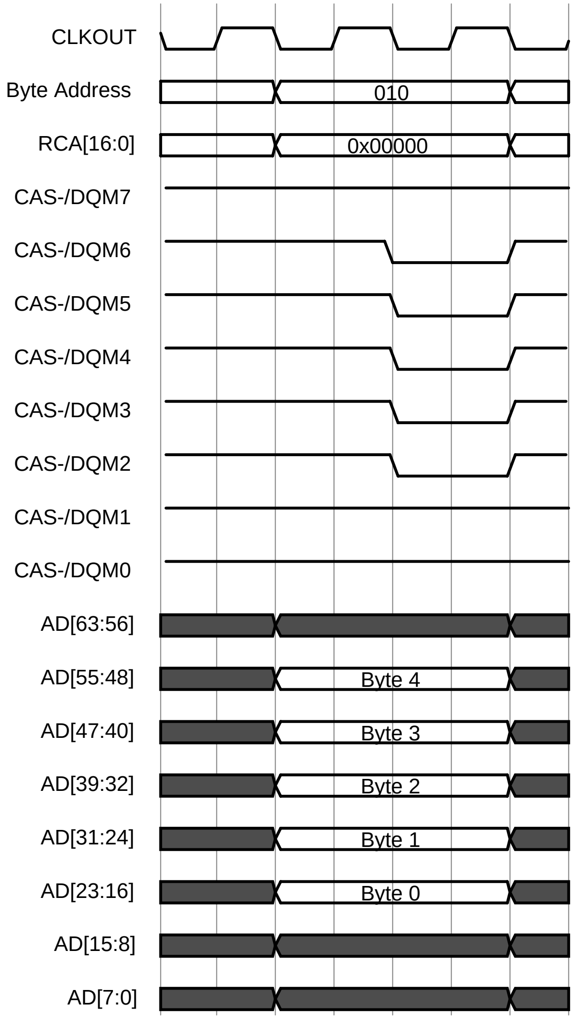
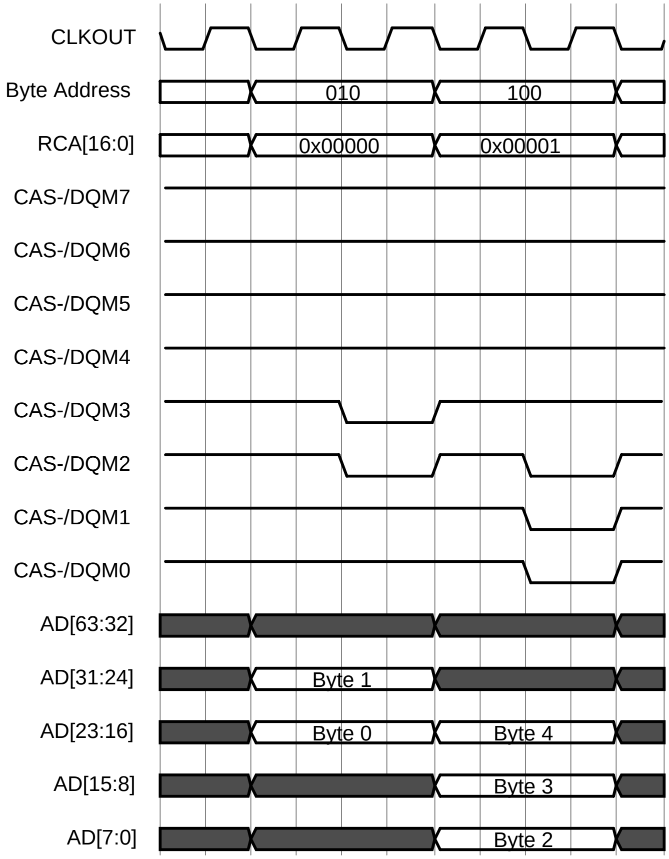


Figure 12–14 shows the little-endian write of the five bytes over a 32-bit data bus. Because the write crosses a 32-bit word boundary, two column accesses are required. The first access is to word `0x02000000`. Only the two MSbytes of the word are written so the data is placed on AD[31:16] and the CAS/DQM[3:2] strobes are activated. The second access to word `0x02000004` writes the last three bytes. These are written to the LSbytes of the word so AD[23:0] have valid data and CAS/DQM[2:0] are active. For a read access of the same bytes, the same column accesses occur, but all CAS/DQM strobes are active and the invalid bytes are discarded after being read by the TC.

Figure 12–14.32-Bit Little-Endian Transfer Example



The 5-byte write on a 16-bit bus requires three column accesses as shown in Figure 12–15. The first access is to halfword `0x02000002`. (An access to halfword `0x02000000` is not required as none of its bytes are being written.) Both bytes in the first halfword are written so that AD[15:0] are valid and CAS/DQM[1:0] are active. The second column access writes both bytes of halfword `0x02000004`. The final access is to halfword `0x02000006`. One byte is transferred on AD[7:0] so only CAS/DQM0 is active. For a read of the same bytes, the identical column accesses occur but all CAS/DQM strobes are active and the invalid bytes are discarded after being read by the TC.

Figure 12–15.16-Bit Little-Endian Transfer Example

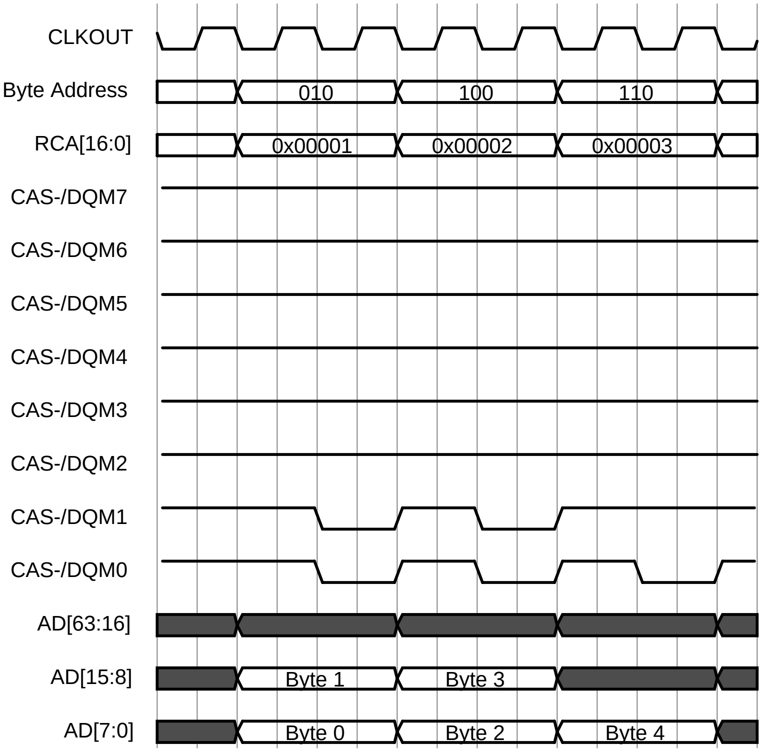
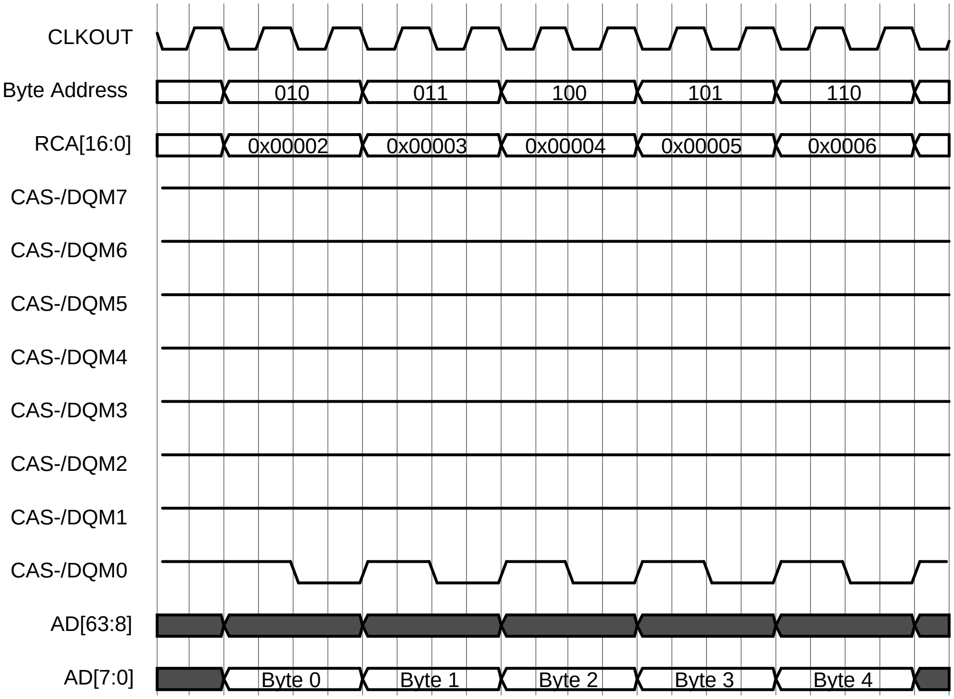


Figure 12–16 shows the little-endian write of the five bytes on an 8-bit bus. For this bus size, each column access transfers a single byte on AD[7:0]. Thus a total of five column accesses are required. The bytes accessed are `0x02000002`, `0x02000003`, `0x02000004`, `0x02000005`, and `0x02000006`. The CAS/DQM0 strobe is active for each column access. For a read access of the same five bytes, the identical column accesses occur, but all CAS/DQM strobes are active and the invalid bytes on AD[63:8] are discarded after being read by the TC.

Figure 12–16.8-Bit Little-Endian Transfer Example



Chapter 13

Bank Configuration Cycle

The bank configuration cycle is used to load data into the TC's cycle configuration cache.

Topics in this chapter include:

Topic **Page**

13.1 Bank Configuration Cycle Overview

The bank configuration cycle is used to load information into a TC cycle configuration cache entry. Bank configuration cycles are performed whenever a configuration cache miss occurs. The data read by the bank configuration cycle is used to determine what type of memory cycle will be subsequently performed on the bank being configured.

13.1.1 Bank Configuration Data

A bank configuration cycle always performs four column accesses. Each access reads one byte of data. This acquires the 32-bits needed for a configuration cache entry. By reading only one byte at a time, the TC supports the use of a flash, ROM, or PLD device to store configuration cache information.

Figure 13–1 shows the format for the four bytes read by a bank configuration cycle. The data must appear on the least significant byte of the AD bus - AD[63:56] in big-endian mode or AD[7:0] in little-endian mode.

Figure 13–1. Bank Configuration Data Format

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th><th></th></tr>
<tr><td>E</td><td colspan="2">TO</td><td>▨</td><td colspan="4">CT</td><td>Byte 0</td></tr>
<tr><td colspan="4">PS</td><td>B</td><td colspan="3">AS</td><td>Byte 1</td></tr>
<tr><td>R</td><td>▨</td><td colspan="2">BW</td><td colspan="2">WS</td><td colspan="2">BS</td><td>Byte 2</td></tr>
<tr><td>▨</td><td colspan="2">PL</td><td colspan="5">MS</td><td>Byte 3</td></tr>
</table>

13.1.2 Bank Configuration Address

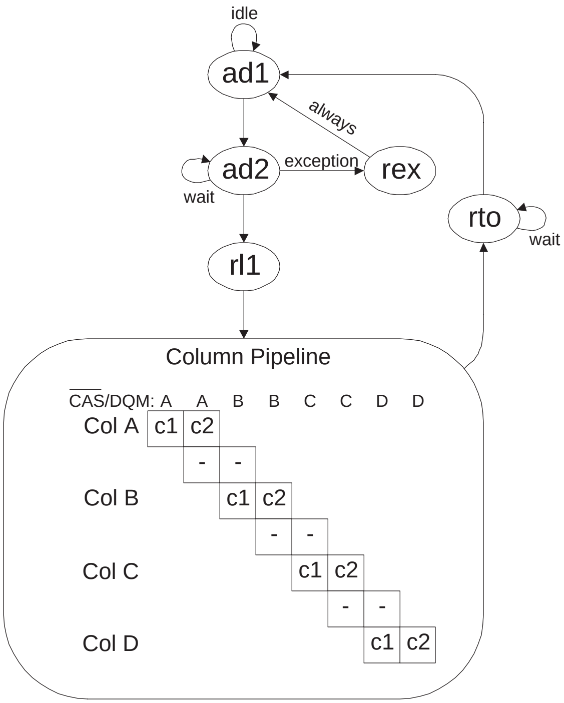
The address output during a bank configuration cycle corresponds to the address on which the configuration cache miss occurred. At row time, the 32-bit address is output on AD[31:0] and logical address bits 31:15 are output on RCA[16:0]. At column time, the RCA[16:0] will be driven with `0x000`, `0x001`, `0x002`, and `0x003` to address the four consecutive configuration bytes.

In most systems, the cache configuration information will reside in a device not within the addressed memory bank itself. In order to access the configuration data, rather than the addressed bank, the system can decode the row-time access-type code (ATC) output on AD[39:32] at the beginning of the cycle. An ATC value of `0x0E` indicates that a bank configuration cycle is being performed. This value can be used to enable the bank configuraton device (and disable the addressed memory bank). The address bits normally used for bank decode (typically the MS bits of the row address) can then be used to select the proper bank configuration bytes within the bank configuration device for the memory being configured.

13.2 Bank Configuration States

The state sequence for the bank configuration cycle is shown in Figure 13–2. The row access consists of the ad1, ad2, and rl1 states. Wait states may be inserted by deasserting the READY input in the ad2 state. Exceptions are always enabled during bank configuration cycles. A fault or retry exception (EXCEPT[1:0]=’01’ or ‘10’) will cause the access to abort and return to ad1 through the rex state. A flush cache entry exception (EXCEPT[1:0]=’00’) is illegal during bank configuration and should be avoided.

Figure 13–2. Bank Configuration States



The column accesses consist of a two-stage pipeline. A new column access will start on every two cycles and each access takes two clock cycles to complete. The pipeline stages are as follows:

c1 The column address is output on RCA[16:0]. The status code (STATUS[1:0]) indicates that a column access is being performed. All CAS/DQM strobes are activated midway through the stage.

c2 The READY input is sampled at the beginning of the stage and, if low, the pipeline stalls and the stage is repeated until

READY is sampled high. The configuration data is latched at the end of the final c2 stage.

After completing the last column access, the memory interface enters the rto state to allow the memory configuration device time to turn off its output drivers. Additional rto states may be added by deasserting the READY input at the beginning of the state. After completion of the (final) rto state, the memory interface returns to the ad1 state to begin the access which initially caused the cache miss.

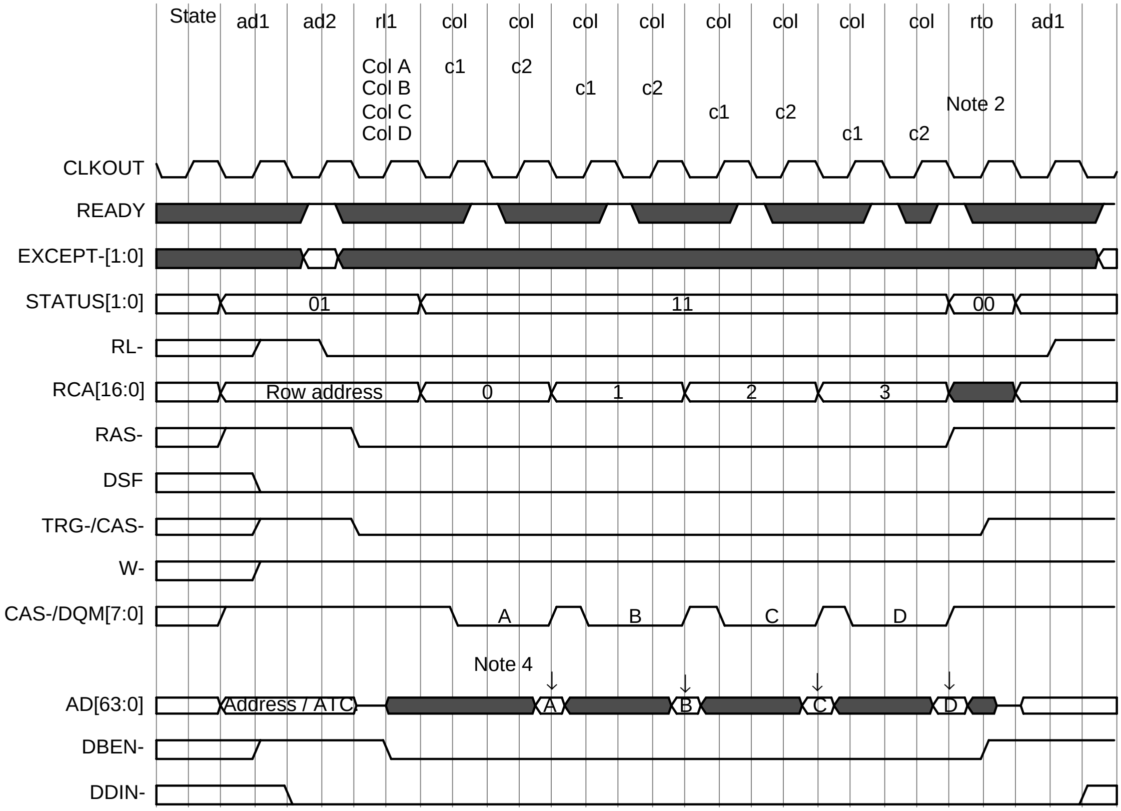
The bank configuration cycle is an atomic operation. The four column accesses will always occur as part of the same page access (unless the ‘C82 is reset). Higher priority requests (such as a host request) will not be serviced until after the bank configuration is complete.

During a bank configuration cycle, the data is transferred directly from the data bus to the appropriate configuration cache register. Since neither the TC’s packet transfer FIFO nor the Cache/DEA FIFO is used in the transfer of data, no contention with the crossbar can occur. For this reason, bank configuration cycles are guaranteed to be free of pipeline bubbles.

13.3 Bank Configuration Example

Figure 13–3 shows an example of a bank configuration. Four column accesses are performed to read the four bytes of the configuration cache entry. (Note that only the LSbyte of the AD bus will be read.) A single rto state then occurs and the memory interface then begins the originally requested access.

Figure 13–3. Bank Configuration



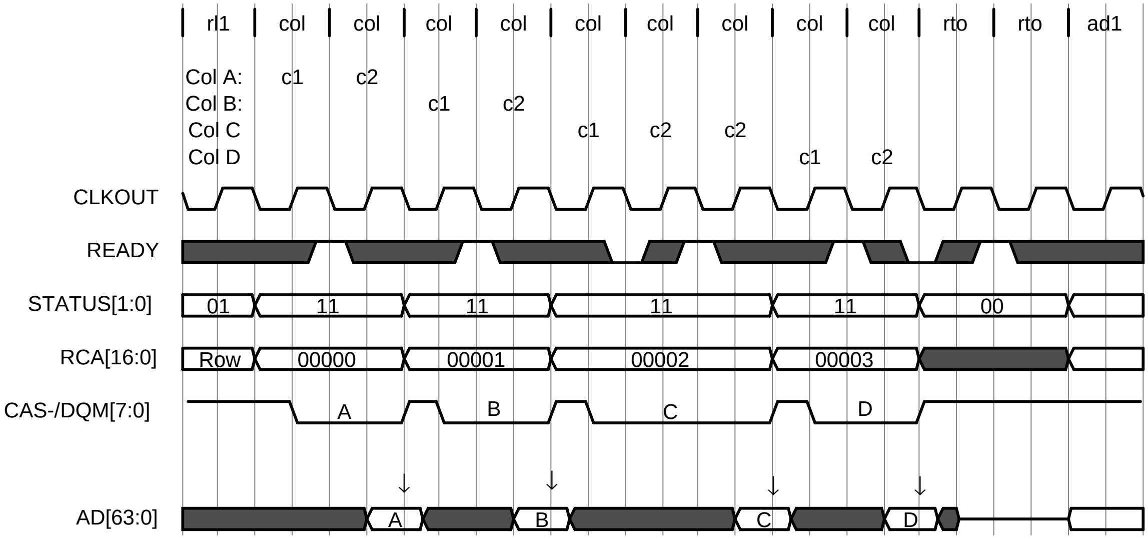
Note 2: Additional turn-off cycles can be inserted by adding waitstates during the rto (turnoff) cycle. This capability is unique to this cycle.

Note 4: Only the least significant byte of the AD[63:0] bus needs to have valid data

Figure 13–4 demonstrates the addition of a wait state to a bank configuration cycle using the READY input. The Col C access is extended by driving READY low at the start of the c2 pipeline stage. The c2 stage is therefore repeated. The CAS/DQM outputs remain low and the READY input is resampled. This time READY is sampled high so the Col C access completes by sampling the data.

Figure 13–4 also shows how extra turn-off time can be added to the end of the bank configuration cycle. Because READY is sampled low at the beginning of the rto state, the rto state is repeated. The READY input is sampled high at the start of the second rto state so the bank configuration completes and the memory interface returns to the ad1 state to start the original access.

Figure 13–4. Bank Configuration with Wait State and Extra Turn-Off



Chapter 14

DRAM Cycles

The ‘C82’s external memory interface provides direct support for EDO DRAMs/VRAMs and devices with similar timing.

Topics in this chapter include:

Topic **Page**

14.1 DRAM Cycle Overview

DRAM memory cycles facilitate the interface to EDO DRAMs and devices with similar timing characteristics. One of four basic DRAM timings can be selected by programming the **CT(3:0)** field in the configuration cache entry to one of the CT = ‘00xx’ values shown in Table 14–1. The cycles vary mainly in the number of clock cycles allotted to each column access. By selecting the proper number of column-time clock cycles, the designer can match the memory cycle to the access time of the selected memory device for the operating frequency of the ‘C82. The column access time for 2 and 3 cycle/column accesses can be further extended through the use of programmed or hardware generated wait states.

Table 14–1. DRAM Cycle Timing Selection

CT(3:0)	Cycle Timing
0 0 0 0	Pipelined 1 Cycle/Column EDO DRAM
0 0 0 1	Non-pipelined 1 Cycle/Column EDO DRAM
0 0 1 0	2 Cycle/Column EDO DRAM
0 0 1 1	3 Cycle/Column EDO DRAM

The ‘C82 accesses EDO DRAMs using page-mode read and write cycles. A multiplexed row/column address is provided on the RCA[16:0] address bus. Page-mode cycles are performed on any column location within the same DRAM row. The row size for each bank is programmed using the **PS(3:0)** field in the cycle configuration cache.

14.1.1 DRAM Row Accesses

Each DRAM access provides a minimum of four clock cycles of row address time. In addition, a RAS high time of at least 3 cycles between DRAM page mode accesses is guaranteed to provide the required RAS precharge time. Wait states may be inserted prior to the fall of RAS to increase RAS decode time. They may also be inserted after the fall of RAS to increase the available RAS access time.

The row-time minimum applies only to memory banks in which exceptions (faults, retries, and cache flushes) are disabled. If exceptions are enabled (by setting the **E** bit in the configuration cache entry), then the TC requires additional time to sample the EXCEPT[1:0] inputs (during the ad2 state) and ensure that an exception has not occurred before it begins loading its column pipeline. If exceptions are enabled, then two extra (exi) states may be inserted during writes prior to the first column access.

14.1.2 DRAM Read Cycles

Read cycles occur as a result of a packet transfer, a cache request, or a DEA request to the TC. DRAM read cycles transfer data or instructions from EDO DRAM-type devices to the TMS320C82. During a read cycle, W is held high, TRG/CAS is driven low after RAS to enable memory output drivers, and DBEN and DDIN are low so that data transceivers may drive into ‘C82. During column time, the TC places the data bus (AD[63:0]) in high impedance state so that the external memory can drive data on the bus. The input data is latched by the TC in the appropriate column stage.

The TC always reads 64 bits (all CAS/DQM[7:0] signals active) and then extracts and aligns the required bytes. Invalid bytes for bus sizes of less than 64 bits are discarded.

Some DRAM read timings have an automatic turn-off cycle added after the last column access in a page. This cycle prevents bus conflict by allowing memory device output buffers to turn off before the ‘C82 begins driving a new address on the AD[63:0] bus. For devices with slow turn-off times, additional cycles can be inserted by programming the **TO(1:0)** field in the configuration cache entry.

14.1.3 Peripheral Device DRAM Read Cycles

These cycles occur as a result of a packet transfer which uses the peripheral device transfer mode. The cycle timing is the same as normal DRAM read cycles except that DBEN remains high and data is latched by a peripheral device rather than the ‘C82.

14.1.4 DRAM Write Cycles

DRAM write cycles transfer data from the TMS320C82 to external memory. Write cycles occur as a result of a packet transfer, a MP data cache write-back, or a DEA request to the TC. During a write cycle, W is driven low prior to CAS/DQM falling, TRG/CAS remains high to disable memory output drivers, and DDIN is driven high to enable data transceivers to drive into memory devices. During column-time, the TC drives the AD[63:0] with the data to be written to external memory. The TC indicates valid data bytes by activating the appropriate CAS/DQM[7:0] strobes.

14.1.5 Peripheral Device DRAM Write Cycles

These cycles occur as a result of a packet transfer which uses the peripheral device transfer mode. Because data is written by the peripheral device, the AD[63:0] is placed in high impedance by the ‘C82 at column time so that it may be driven by the peripheral. The DBEN signal remains high during the cycle so that external transceivers (if present) are disabled.

For some DRAM peripheral device writes, the cycle timing is modified from the normal DRAM write cycle. An extra rto state is added to these cycles after the last column access. This state provides time for the peripheral device to stop driving the data bus after the last column access before the ‘C82 begins driving with the address for a new page access. Additional turn-off cycles can be added by programming the **TO(1:0)** field of the configuration cache entry.

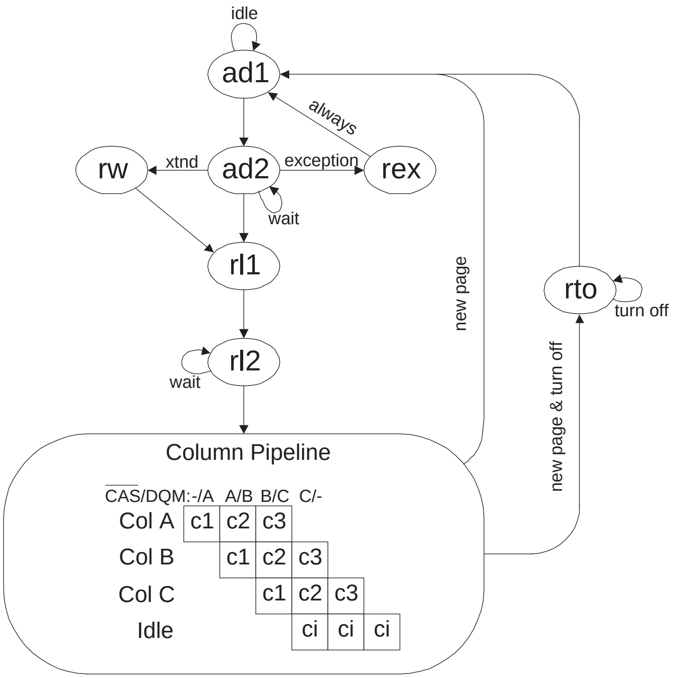
14.2 Pipelined 1 Cycle/Column DRAM Cycles

Pipelined 1 cycle/column DRAM timings are designed to interface to DRAMs/VRAMs which support pipelined page-mode. These devices use the CAS input to clock data in and out of the device and to latch addresses. Reads occur in a pipelined fashion so that an address is clocked in on one falling CAS edge and data is clocked out by the next falling CAS edge. Data inputs are registered allowing data to be latched on the same clock edge as address for write cycles.

14.2.1 Pipelined 1 Cycle/Column DRAM Read

Figure 14–1 shows the state sequence for a pipelined 1 cycle/column DRAM read. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 14–1. Pipelined 1 Cycle/Column DRAM Read States



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

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. All CAS/DQM strobes are activated to latch the address into the memories.

c2 The CAS/DQM strobes are activated again to clock out the data that was addressed in the previous stage, c1.

c3 The data clocked out in the c2 stage is latched by the ‘C82.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. CAS/DQM strobes may still go active due to a previous access and data from previous accesses will still be latched.

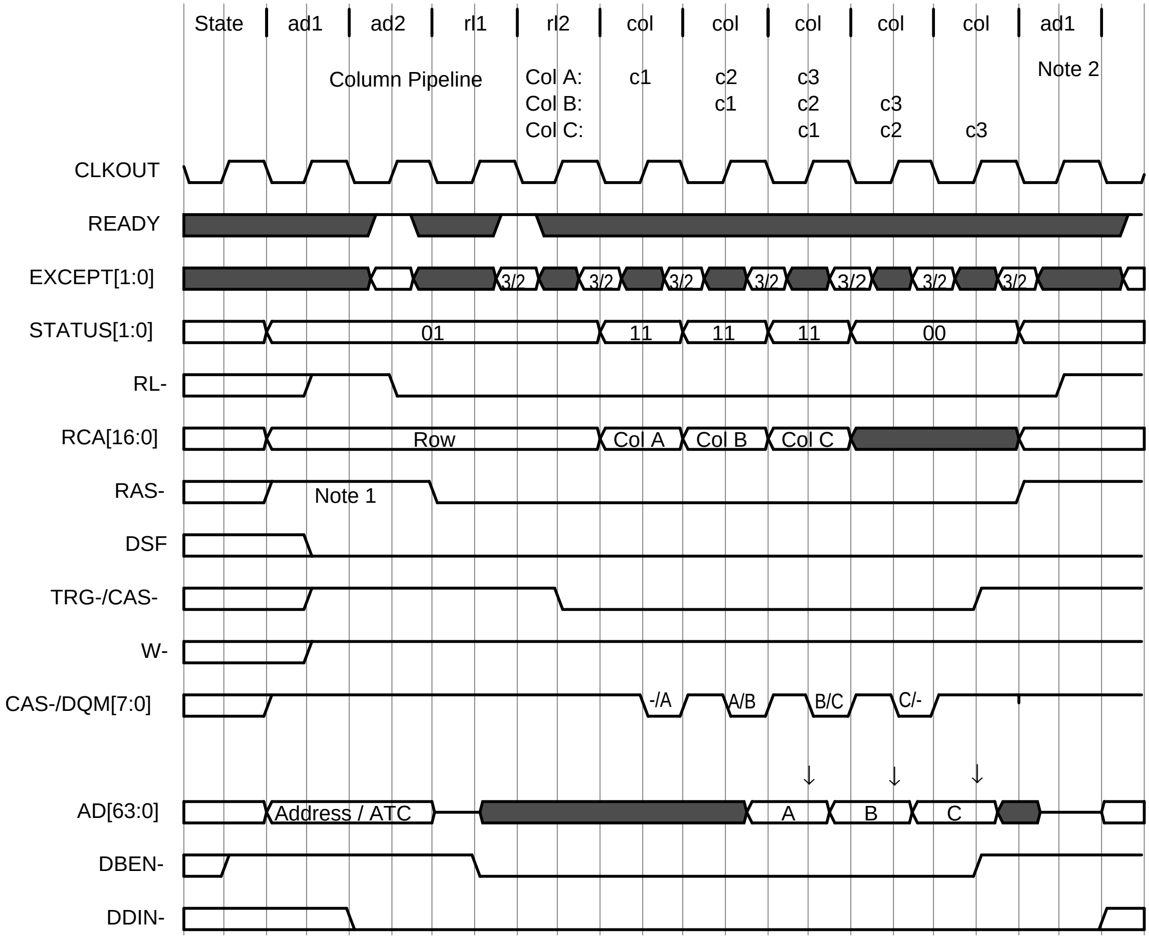
Because the pipeline stages overlap, the CAS/DQM strobes represent either the address strobes for the current column (c1 stage), or the data strobes for the previous column (c2 stage) in the access. Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One or more rto states will be inserted prior to returning to ad1 if specified in the configuration cache entry.

14.2.1.1 Pipelined 1 Cycle/Column Read Example

An example of a pipelined 1 cycle/column DRAM read access is shown in Figure 14–2. The example shows a page-mode read of three column addresses. The CAS/DQM labels indicate the access for which the CAS/DQM strobe represents the data and address strobe respectively. For example, A/B indicates that CAS/DQM is strobing data A out and address B in. The read data for each column access is latched by the ‘C82 at the point indicated by the ↓ symbols. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–2. Pipelined 1 Cycle/Column DRAM Read



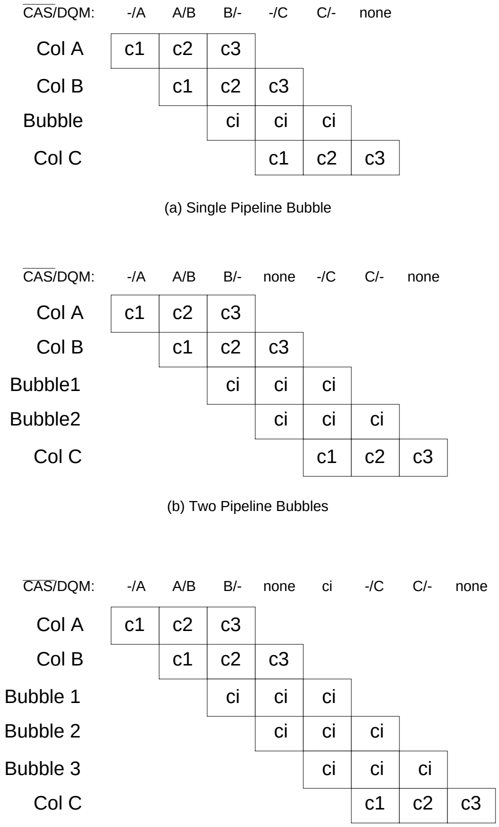
Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS high of at least 3 cycles.

Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

14.2.1.2 Pipelined 1 Cycle/Column Read With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed but the previously loaded cycles continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 14–3.

Figure 14–3. Effect of Bubbles on Pipelined 1 Cycle/Column DRAM Read Column Pipeline



$\overline{\text{CAS}}$/DQM:	-/A	A/B	B/-	-/C	C/-	none
Col A	c1	c2	c3			
Col B		c1	c2	c3		
Bubble			ci	ci	ci	
Col C				c1	c2	c3

(a) Single Pipeline Bubble

$\overline{\text{CAS}}$/DQM:	-/A	A/B	B/-	none	-/C	C/-	none
Col A	c1	c2	c3				
Col B		c1	c2	c3			
Bubble1			ci	ci	ci		
Bubble2				ci	ci	ci	
Col C					c1	c2	c3

(b) Two Pipeline Bubbles

$\overline{\text{CAS}}$/DQM:	-/A	A/B	B/-	none	ci	-/C	C/-	none
Col A	c1	c2	c3					
Col B		c1	c2	c3				
Bubble 1			ci	ci	ci			
Bubble 2				ci	ci	ci		
Bubble 3					ci	ci	ci	
Col C						c1	c2	c3

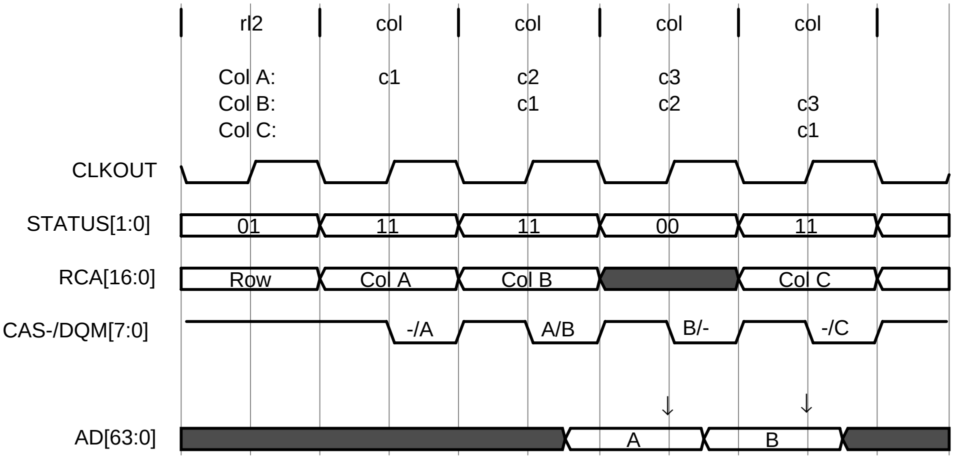
(c) Multiple Pipeline Bubbles

14.2.1.3 Pipelined 1 Cycle/Column Read With Single-Cycle Bubble

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–3(a). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for single cycle bubbles is shown in Figure 14–4. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The CAS/DQM strobes are still activated however, in order to strobe out the data from Col B and the Col A data is latched.

Figure 14–4. Pipelined 1 Cycle/Column DRAM Read (Single Cycle Bubble)

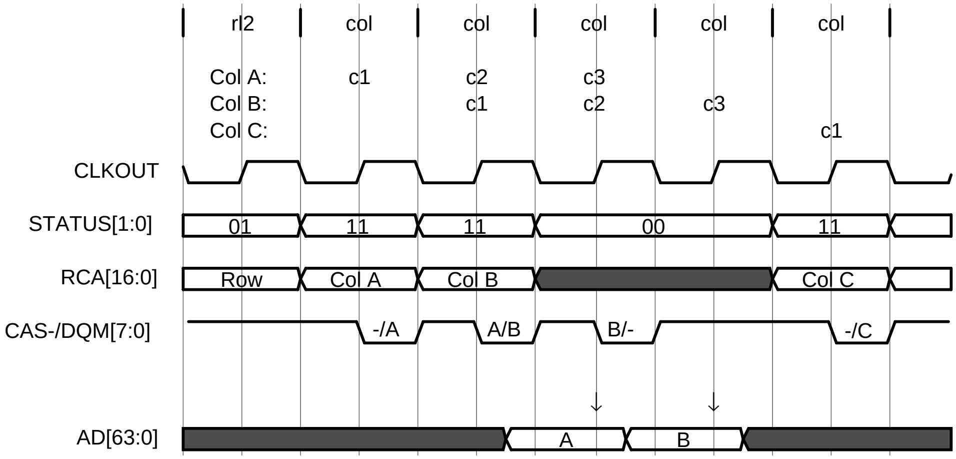


14.2.1.4 Pipelined 1 Cycle/Column Read With Two-Cycle Bubble

When a column pipeline bubble lasts for two cycles, the pipeline appears as shown in Figure 14–3(b). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second bubble delays the access for one more cycle. Stages c3 of Col A and c2 & c3 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for two-cycle bubbles is shown in Figure 14–5. The bubble causes an idle status code to be output for two cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are still activated in order to strobe out the data from Col B and the Col A data is latched. During the second idle cycle, the CAS/DQM strobes remain inactive but the data from Col B is latched.

Figure 14–5. Pipelined 1 Cycle/Column DRAM Read (Two-Cycle Bubble)

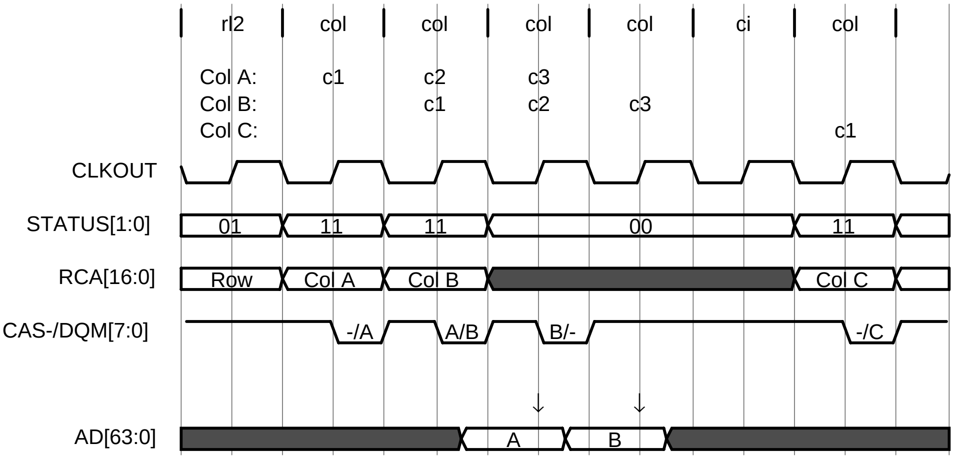


14.2.1.5 Pipelined 1 Cycle/Column Read With Multi-Cycle Bubble

When a column pipeline bubble lasts longer than two cycles, the pipeline drains and the memory enters the idle (ci) state until the next access is ready to begin. This pipeline sequence is shown in Figure 14–3(c). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second and third bubble delay the access for two more cycles. Stages c3 of Col A and c2 & c3 of Col B complete as normal and the pipeline then enters the idle state. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for multi-cycle bubbles is shown in Figure 14–6. The bubble causes an idle status code to be output for three cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are still activated in order to strobe out the data from Col B and the Col A data is latched. During the second idle cycle, the CAS/DQM strobes remain inactive but the data from Col B is latched. When the third idle cycle is reached, the pipeline is completely empty. The CAS/DQM strobes are inactive and no data is latched.

Figure 14–6. Pipelined 1 Cycle/Column DRAM Read (Multi-Cycle Bubble)

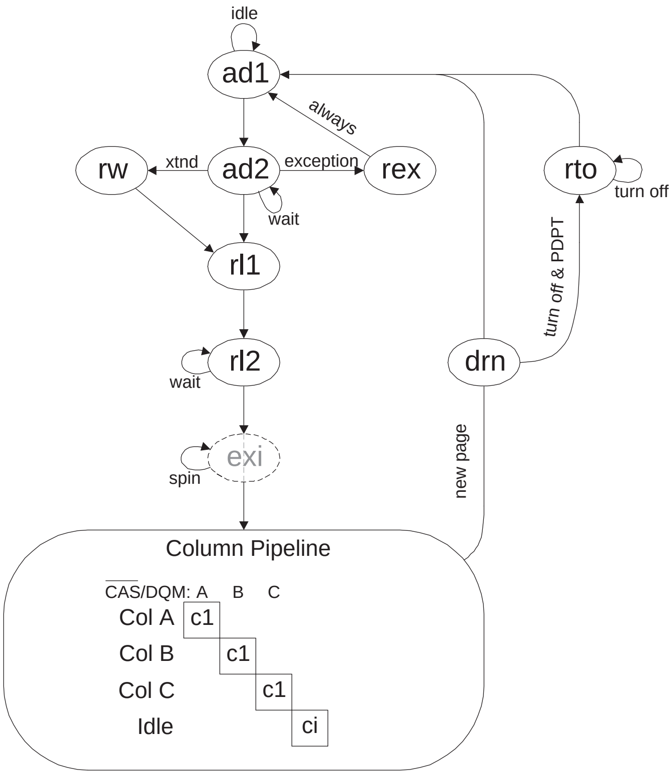


14.2.2 Pipelined 1 Cycle/Column DRAM Write

Figure 14–7 shows the state sequence for a pipelined 1 cycle/column DRAM write. This cycle timing is designed for use with pipelined memory devices. These devices latch data at the fall of CAS but don't update the memory array until the following CAS strobe. Thus the CAS strobes act as clocks to clock the memory pipeline as well as strobe the column address.

The row access consists of the ad1, ad2, rl1, rl2, and possibly rw and exi states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 14–7. Pipelined 1 Cycle/Column DRAM Write States



Column accesses consist of a single pipeline. A new column access can begin every cycle. Each column access requires a second CAS strobe to complete but the strobe corresponding to the written byte may not occur immediately. The pipeline stages are as follows:

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. Data is driven on the appropriate bytes of the AD[63:0] bus and the CAS/DQM strobe corresponding to the valid bytes are activated. This latches the address and data into the memory devices. The CAS/DQM strobes will also cause the data from the previous write to the same byte to be written to the memory array.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. CAS/DQM strobes are inactive and no new data is output.

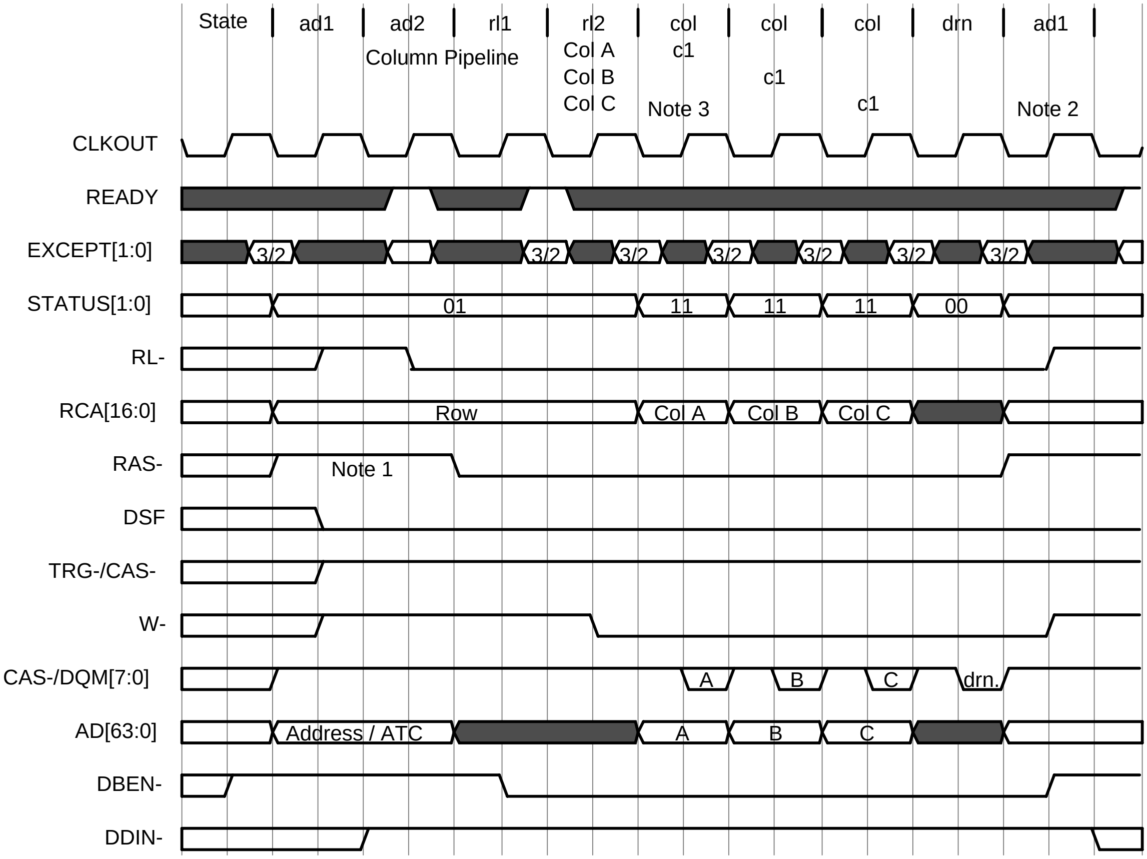
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. Before returning to the ad1 state, the drn state will always be executed. This state activates all CAS/DQM strobes in order to update the memory array with the data last written to each byte. If the write access is the result of a peripheral device packet transfer, one or more rto states will be inserted after the drn state and prior to returning to ad1 if specified in the configuration cache entry.

14.2.2.1 Pipelined 1 Cycle/Column Write Example

An example of a pipelined 1 cycle/column DRAM write access is shown in Figure 14–8. The example shows a page-mode write of three column addresses. After the access to Col C is completed, the memory interface receives a request requiring a new page access. The drn state is entered to ensure that the last data written to each byte of the data bus is transferred to the memory array. The interface then returns to the ad1 state to begin a new row access.

Figure 14–8. Pipelined 1 Cycle/Column DRAM Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS high of at least 3 cycles.

Note 2: During peripheral data transfers, turn-off cycles will be inserted prior to ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

14.2.2.2 Pipelined 1 Cycle/Column Write With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode writes, the loading of a new column access into the pipeline is simply delayed for one cycle. This effect is shown in Figure 14–9.

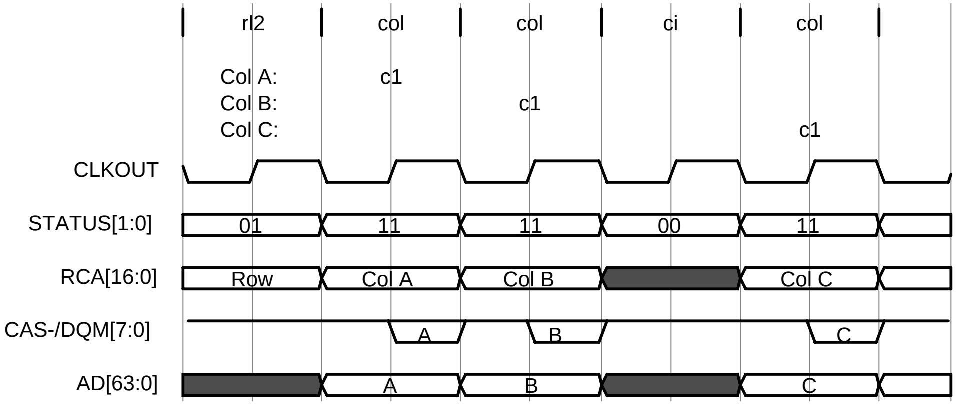
Figure 14–9. Effect of Bubbles on Pipelined 1 Cycle/Column DRAM Write Column Pipeline

$\overline{\text{CAS}}$/DQM:	A	B	ci	C
Col A	c1			
Col B		c1		
Bubble			ci	
Col C				c1

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–9. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for bubbles is shown in Figure 14–10. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output.

Figure 14–10.Pipelined 1 Cycle/Column DRAM Write (With Bubble)



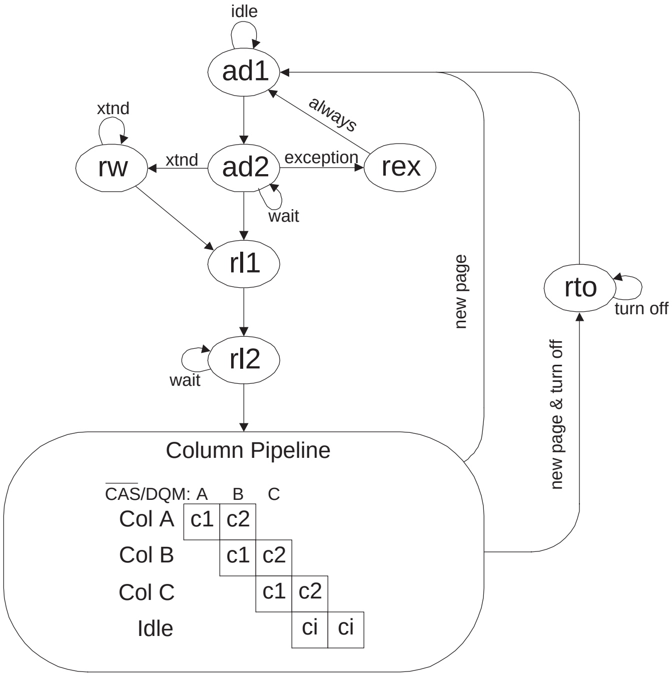
14.3 Nonpipelined 1 Cycle/Column DRAM Cycles

The nonpipelined 1 cycle/column DRAM timings are designed to interface to DRAMs/VRAMs with very fast access times. These devices must be able to provide read data within 1 clock cycle of the fall of CAS.

14.3.1 Nonpipelined 1 Cycle/Column DRAM Read

Figure 14–11 shows the state sequence for a non-pipelined 1 cycle/column DRAM read. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 14–11.Nonpipelined 1 Cycle/Column DRAM Read States



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 column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. All CAS/DQM strobes are activated to latch the address into the memories.

c2 The data addressed in stage c1 is latched by the ‘C82.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. Data from previous accesses will still be latched.

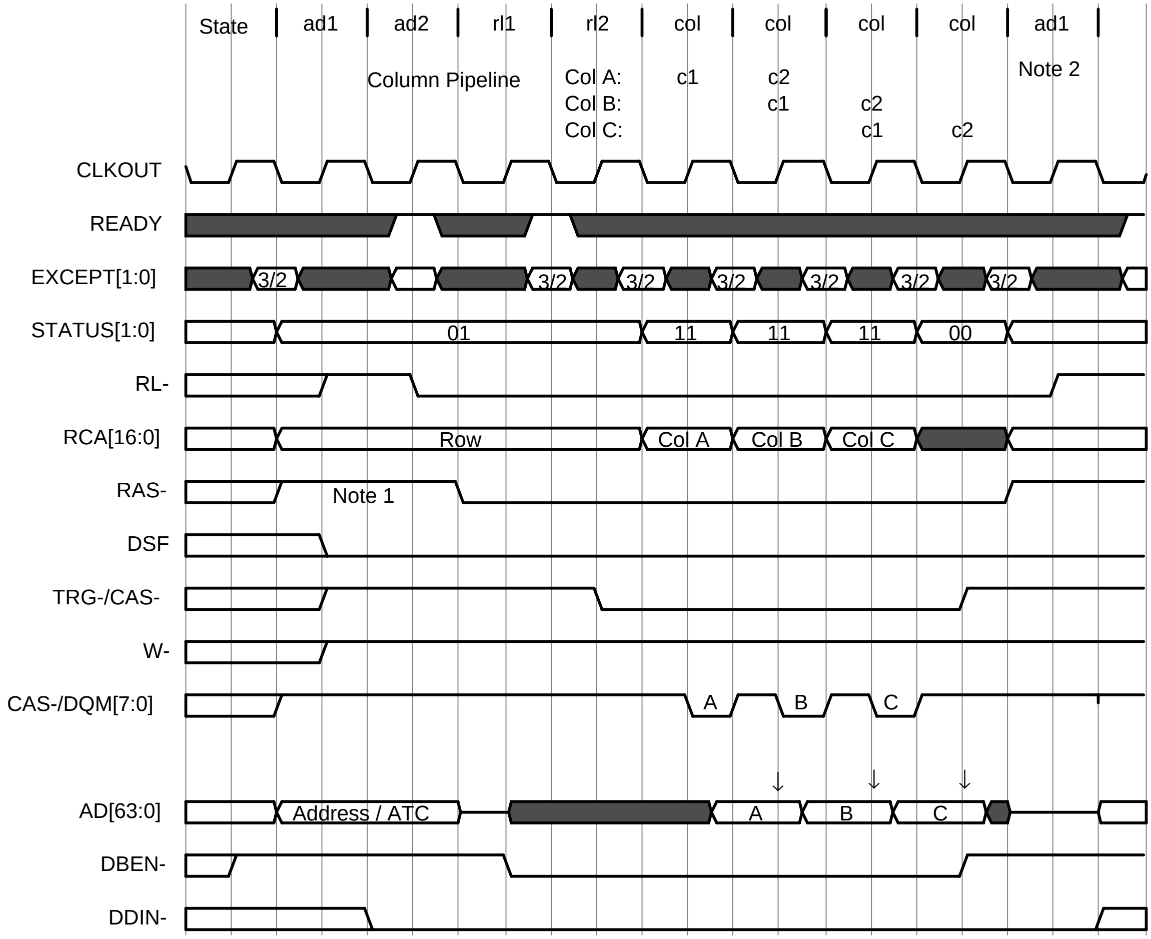
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One or more rto states will be inserted prior to returning to ad1 if specified in the configuration cache entry.

14.3.1.1 Nonpipelined 1 Cycle/Column Read Example

An example of a nonpipelined 1 cycle/column DRAM read access is shown in Figure 14–12. The example shows a page-mode read of three column addresses. A new column access begins on each cycle. The read data for each column access is latched by the ‘C82 at the point indicated by the ↓ symbols. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–12.Nonpipelined 1 Cycle/Column DRAM Read



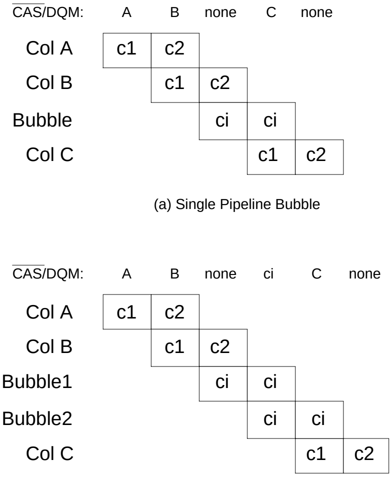
Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS high of at least 3 cycles.

Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

14.3.1.2 Nonpipelined 1 Cycle/Column Read With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed but the previously loaded cycles continue to completion through their c1 and c2 stages. This effect is shown in Figure 14–13.

Figure 14–13.Effect of Bubbles on Nonpipelined 1 Cyc/Col DRAM Read Column Pipeline



(a) Single Pipeline Bubble

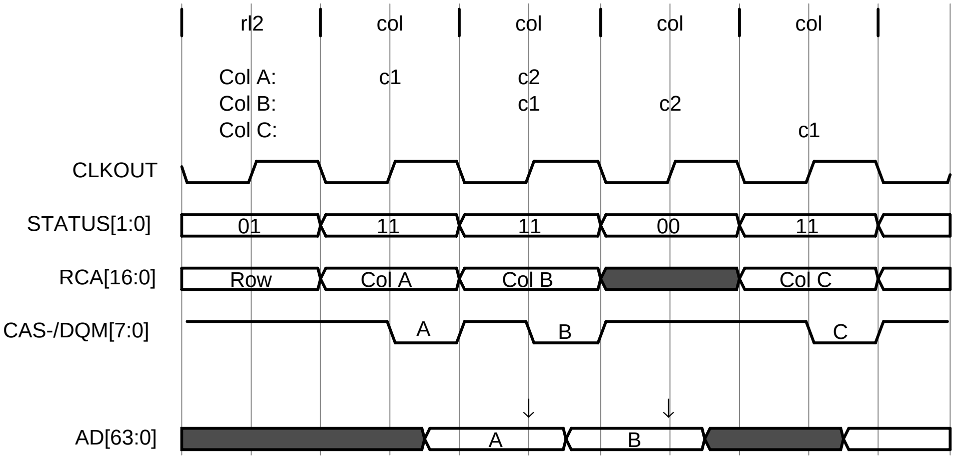
(b) Multiple Pipeline Bubbles

14.3.1.3 Nonpipelined 1 Cycle/Column Read With Single-Cycle Bubble

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–13(a). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stage c2 of Col B completes as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for single cycle bubbles is shown in Figure 14–14. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The Col B data is latched.

Figure 14–14.Nonpipelined 1 Cycle/Column DRAM Read (Single-Cycle Bubble)

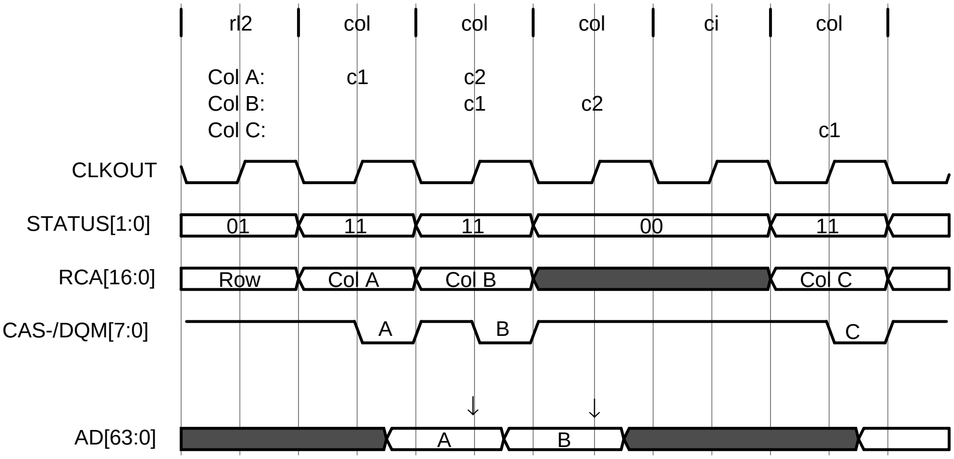


14.3.1.4 Nonpipelined 1 Cycle/Column Read With Multi-Cycle Bubble

When a column pipeline bubble lasts longer than one cycle, the pipeline drains and the memory enters the idle (ci) state until the next access is ready to begin. This pipeline sequence is shown in Figure 14–13(b). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second bubble delays the access for another cycle. Stage c2 of Col B completes as normal and the pipeline then enters the idle state. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for multi-cycle bubbles is shown in Figure 14–15. The bubble causes an idle status code to be output for two cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are inactive but the Col B data is latched. When the second idle cycle is reached, the pipeline is completely empty. The CAS/DQM strobes are inactive and no data is latched.

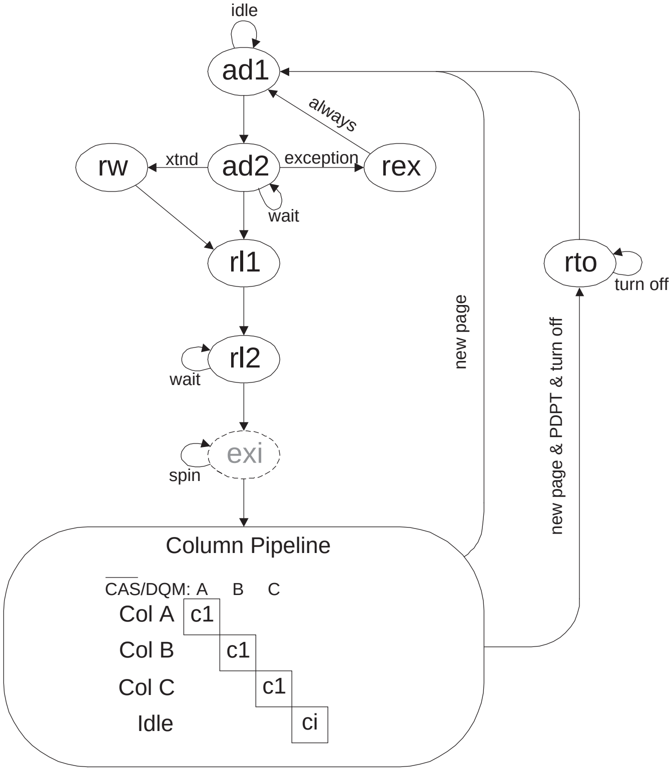
Figure 14–15.Nonpipelined 1 Cycle/Col DRAM Read (Multi-Cycle Bubbles)



14.3.2 Nonpipelined 1 Cycle/Column DRAM Write

Figure 14–16 shows the state sequence for a non-pipelined 1 cycle/column DRAM write. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw and exi states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 14–16.Nonpipelined 1 Cycle/Column DRAM Write States



Column accesses consist of a single stage pipeline. A new column access can begin every cycle. The pipeline stages are as follows:

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. Data is driven on the appropriate bytes of the AD[63:0] bus and the CAS/DQM strobes corresponding to the valid bytes are activated.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. CAS/DQM strobes are inactive and no new data is output.

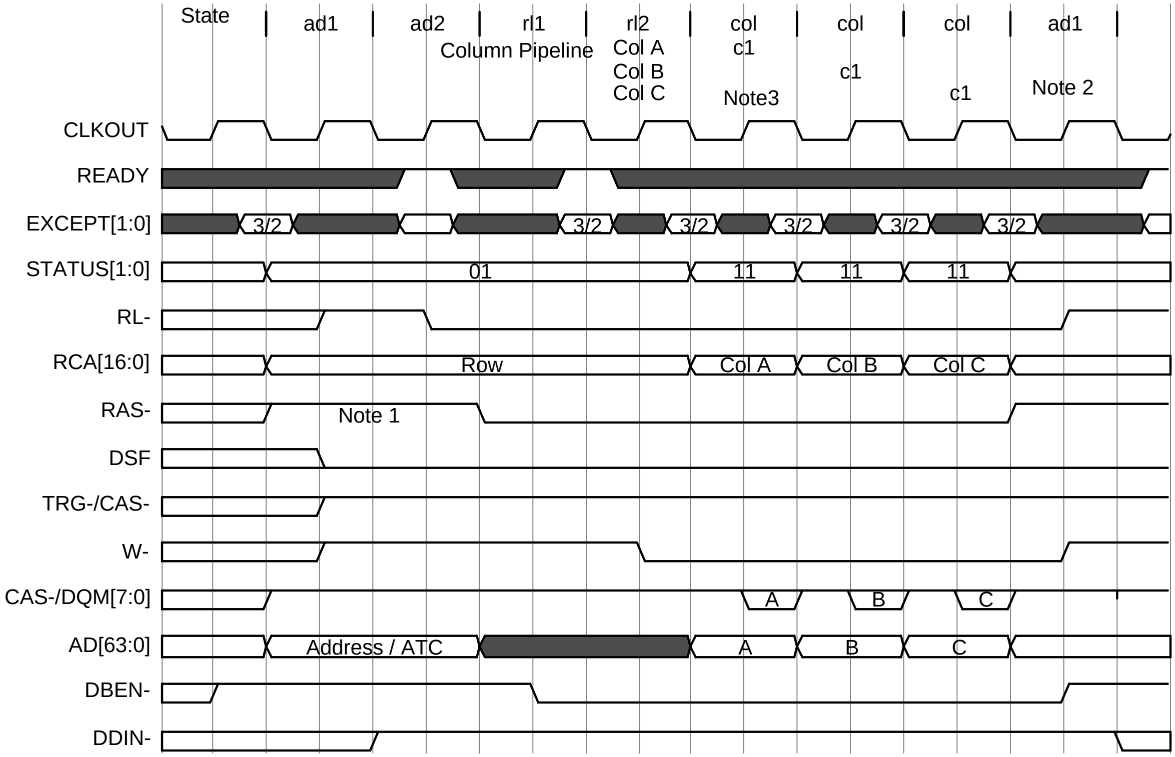
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, one or more rto states will be inserted prior to returning to ad1 if specified in the configuration cache entry.

14.3.2.1 Nonpipelined 1 Cycle/Column Write Example

An example of a nonpipelined 1 cycle/column DRAM write access is shown in Figure 14–17. The example shows a page-mode write of three column addresses. A new column access begins on each cycle. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–17.Nonpipelined 1 Cycle/Column DRAM Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
Note 2: During peripheral device PTs, turn-off cycles will be inserted prior to ad1 as specified by the bank configuration
Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

14.3.2.2 Nonpipelined 1 Cycle/Column Write With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode writes, the loading of a new column access into the pipeline is delayed by one cycle. This effect is shown in Figure 14–18.

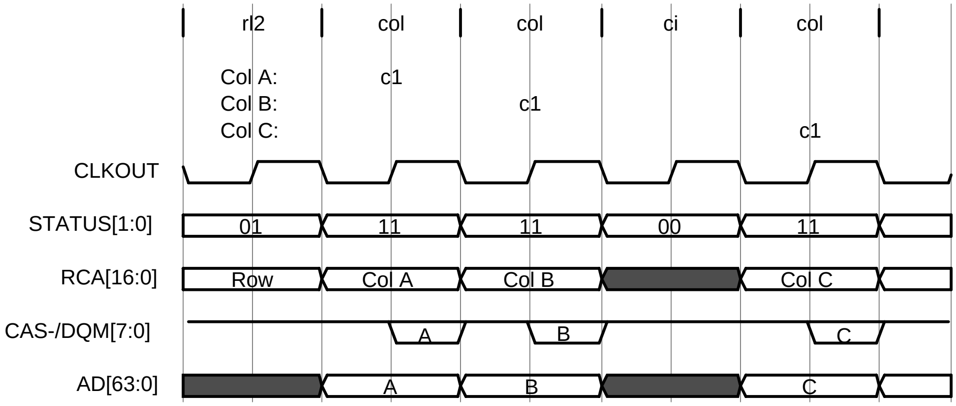
Figure 14–18.Effect of Bubbles on Nonpipelined 1 Cyc/Col DRAM Write Column Pipeline

CAS/DQM:	A	B	ci	C
Col A	c1			
Col B		c1		
Bubble			ci	
Col C				c1

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–18. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for bubbles is shown in Figure 14–19. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output.

Figure 14–19.Nonpipelined 1 Cycle/Column DRAM Write (with Bubble)



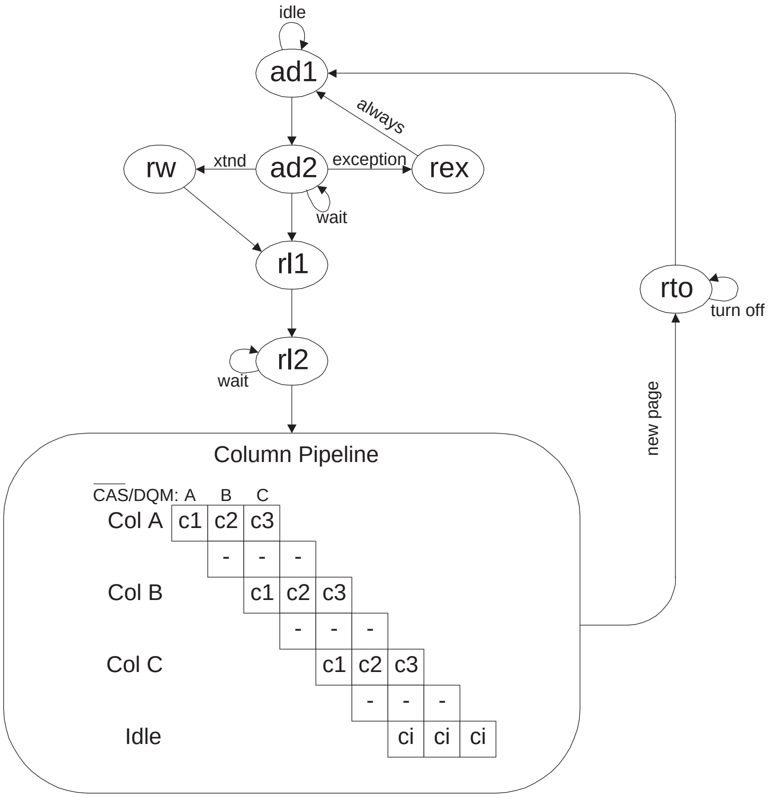
14.4 2 Cycle/Column DRAM Cycles

The 2 cycle/column DRAM timings are designed to interface to EDO DRAMs/VRAMs with standard access times. These cycles provide one cycle of column address setup time, one cycle of CAS low time and two cycles of EDO access time from CAS. Column access time may be further increased by the addition of programmable or hardware generated wait states.

14.4.1 2 Cycle/Column DRAM Read

Figure 14–20 shows the state sequence for a 2 cycle/column DRAM read. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 14–20.2 Cycle/Column DRAM Read States



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

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c1 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = `1x`.

c2 All CAS/DQM strobes are activated to latch the column address into the memories. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c2 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = `01`. The c2 state is repeated twice if **WS(1:0)** = `11`.

c3 The CAS/DQM strobes are deactivated at the beginning of the cycle. Data addressed in stage c1 is latched at the end of the cycle.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. Data from previous accesses will still be latched.

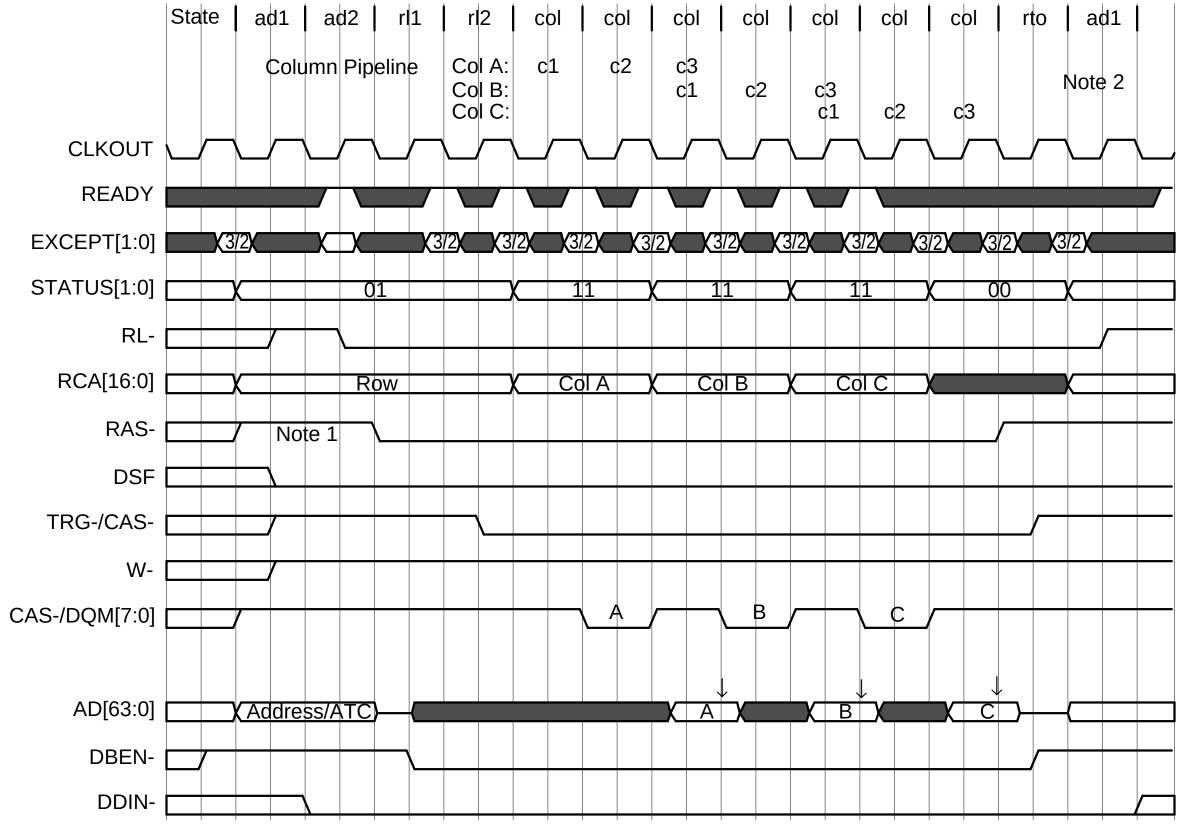
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One rto state will always be inserted prior to returning to ad1 to allow memory output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

14.4.1.1 2 Cycle/Column DRAM Read Example

An example of a 2 cycle/column DRAM read access is shown in Figure 14–21. The example shows a page-mode read of three column addresses. A new column access begins on evry two clock cycles. The read data for each column access is latched by the ‘C82 at the point indicated by the ↓ symbols. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–21.2 Cycle/Column DRAM Read



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

14.4.1.2 2 Cycle/Column DRAM Read With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed but the previously loaded cycles continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 14–22.

Figure 14–22.Effect of Bubbles on 2 Cycle/Column DRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	none	B	none	none	C	none
Col A	c1	c2	c3					
		-	-	-				
Col B			c1	c2	c3			
				-	-	-		
Bubble					ci	ci	ci	
Col C						c1	c2	c3

(a) Single Pipeline Bubble

$\overline{\text{CAS}}$/DQM:	none	A	none	B	none	ci	none	C	none
Col A	c1	c2	c3						
		-	-	-					
Col B			c1	c2	c3				
				-	-	-			
Bubble1					ci	ci	ci		
Bubble2						ci	ci	ci	
Col C							c1	c2	c3

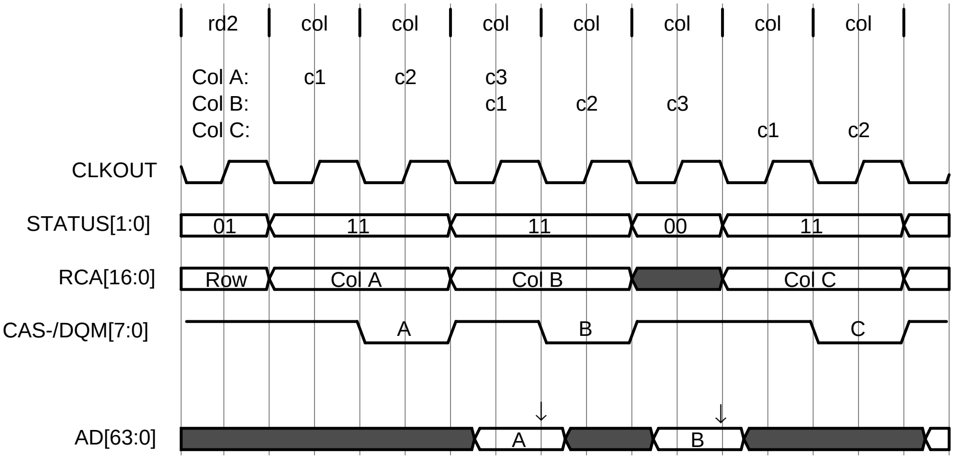
(b) Multiple Pipeline Bubbles

14.4.1.3 2 Cycle/Column DRAM Read With Single-Cycle Bubble

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–22(a). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stage c3 of Col B completes as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for single cycle bubbles is shown in Figure 14–23. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The Col B data is latched.

Figure 14–23.2 Cycle/Column DRAM Read (Single-Cycle Bubble)

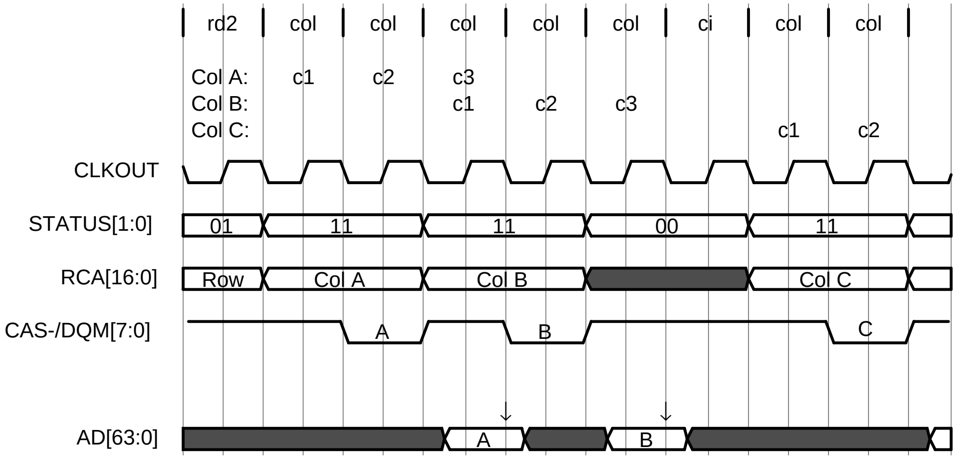


14.4.1.4 2 Cycle/Column Read With Multi-Cycle Bubble

When a column pipeline bubble lasts longer than one cycle, the pipeline drains and the memory enters the idle (ci) state until the next access is ready to begin. This pipeline sequence is shown in Figure 14–22(b). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second bubble delays the access for another cycle. Stage c3 of Col B completes as normal and the pipeline then enters the idle state. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for multi-cycle bubbles is shown in Figure 14–24. The bubble causes an idle status code to be output for two cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are inactive but the Col B data is latched. When the second idle cycle is reached, the pipeline is completely empty. The CAS/DQM strobes are inactive and no data is latched.

Figure 14–24.2 Cycle/Col DRAM Read (Multi-Cycle Bubbles)



14.4.1.5 2 Cycle/Column DRAM Read Column Wait States

Column accesses can be automatically extended by an integral number of states using the wait state field (**WS(1:0)**) of the configuration cache entry. The automatic wait states cause the column pipeline to stall for one cycle while the current state is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 14–25.

Figure 14–25.Effect of Wait States on 2 Cycle/Column DRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	A	none	B	B	none
Col A	c1	c2	c2	c3			
		-	-	-	-		
c2 wait			-	-	-	-	
Col B				c1	c2	c2	c3

(a) WS(1:0) = 01

$\overline{\text{CAS}}$/DQM:	none	none	A	A	none	none	B	B	none
Col A	c1	c1	c2	c2	c3				
c1 wait		-	-	-	-	-			
			-	-	-	-	-		
c2 wait				-	-	-	-	-	
Col B					c1	c1	c2	c2	c3

(b) WS(1:0) = 10

$\overline{\text{CAS}}$/DQM:	none	none	A	A	A	none	none	B	B	B	none
Col A	c1	c1	c2	c2	c2	c3					
c1 wait		-	-	-	-	-	-				
			-	-	-	-	-	-			
c2 wait				-	-	-	-	-	-		
c2 wait					-	-	-	-	-	-	
Col B						c1	c1	c2	c2	c2	c3

(c) WS(1:0) = 11

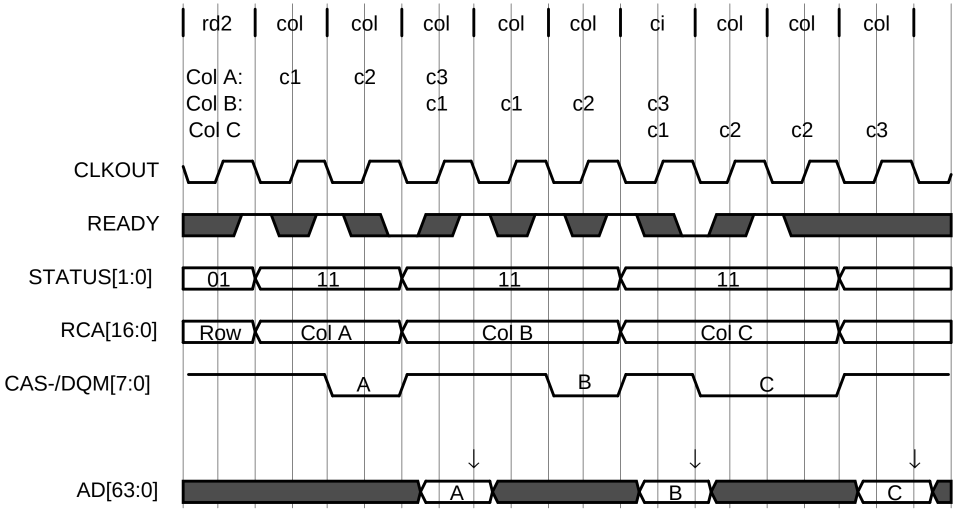
A **WS(1:0)** value of ‘01’ adds a CAS low wait state to each access. This causes the pipeline to repeat the c2 stage as shown in Figure 14–25(a). A value of **WS(1:0)** = ‘10’ adds a CAS high and CAS low wait state to each access. As Figure 14–25(b) shows, this causes the pipeline to repeat the c1 and c2 stages. Programming **WS(1:0)** to ‘11’ adds a CAS high and two CAS low wait states to each column access. The c1 state is repeated and the c2 state is repeated twice as shown in Figure 14–25(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c2 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 14–26 demonstrates the addition of wait states using the READY input. A single CAS high wait state is added to the Col B access by driving

READY low at the start of its c1 stage. READY is high when resampled in the next c1 stage and the Col B access proceeds. A single CAS low wait state is added to the Col C access by driving READY low at the start of its c2 stage. READY is sampled high in the next c2 stage and the Col C access completes.

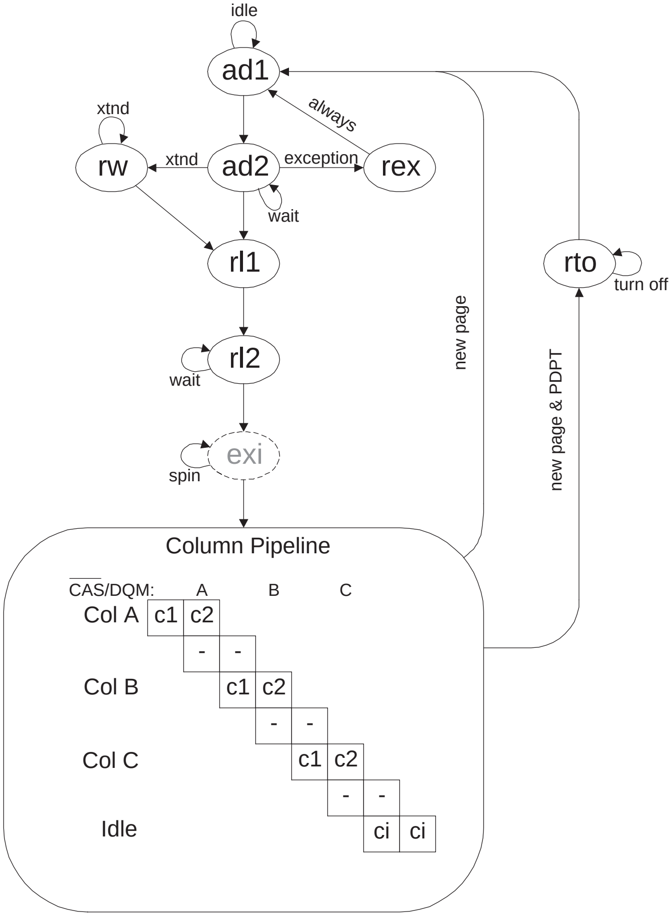
Figure 14–26.2 Cycle/Column DRAM Read with Wait States



14.4.2 2 Cycle/Column DRAM Write

Figure 14–27 shows the state sequence for a 2 cycle/column DRAM write. It provides one cycle of column address setup time and one cycle of CAS low time. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw and exi states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a single rw state will automatically be inserted after ad2 to ensure a RAS high time of at least three clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 14–27. 2 Cycle/Column DRAM Write States



Column accesses consist of a two-stage pipeline. A new column access can begin every two cycles, and each access takes two cycles to complete so no overlap of pipeline stages occurs. The pipeline stages are as follows:

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. Data is driven on the appropriate bytes of the data bus (AD[63:0]). The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c1 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = 1x.

c2 The CAS/DQM strobes corresponding to the valid bytes on the data bus are activated. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c2 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = `01`. The c2 state is repeated twice if **WS(1:0)** = `11`.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on **STATUS[1:0]** to indicate that no new column access is beginning. The CAS/DQM strobes are inactive.

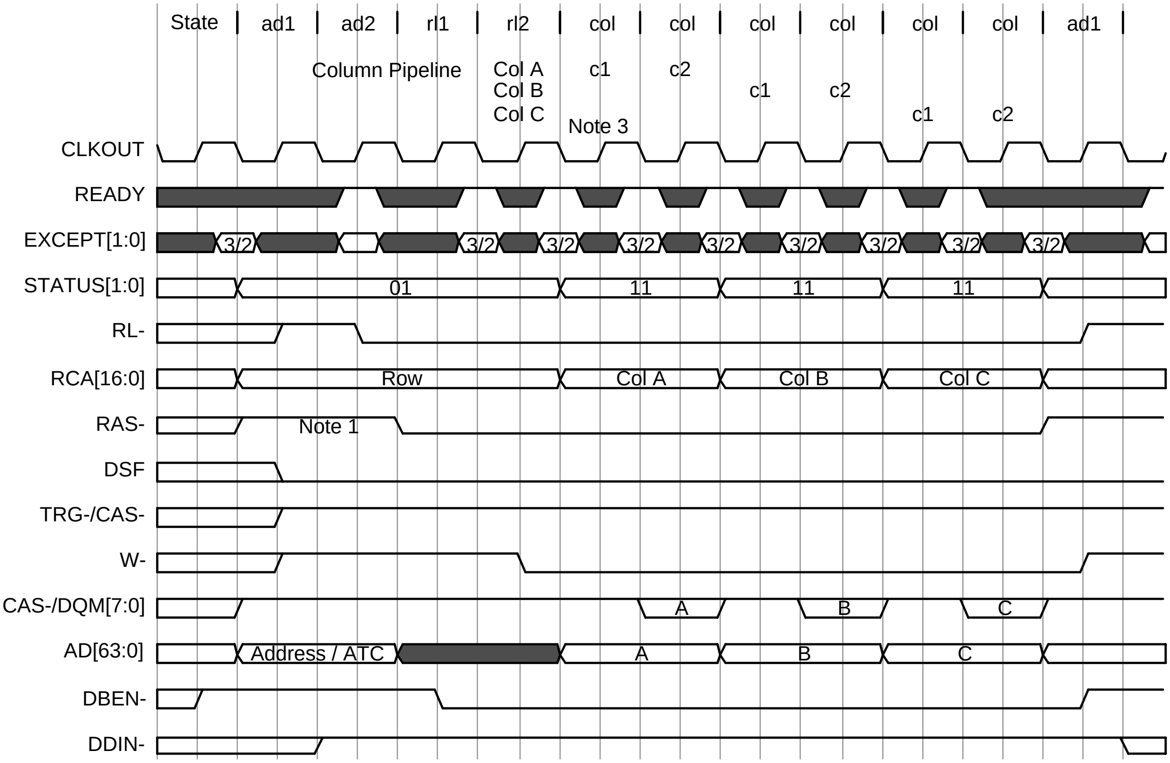
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, then one rto state will always be inserted prior to returning to ad1 to allow peripheral device output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

14.4.2.1 2 Cycle/Column DRAM Write Example

An example of a 2 cycle/column DRAM write access is shown in Figure 14–28. The example shows a page-mode write of three column addresses. A new column access begins on every other cycle. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–28.2 Cycle/Column DRAM Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Note 2: During peripheral data transfer, one turn-off cycle is automatically inserted prior to ad1. Additional turn-off cycles will be inserted as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

14.4.2.2 Cycle/Column DRAM Write With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed. This effect is shown in Figure 14–29.

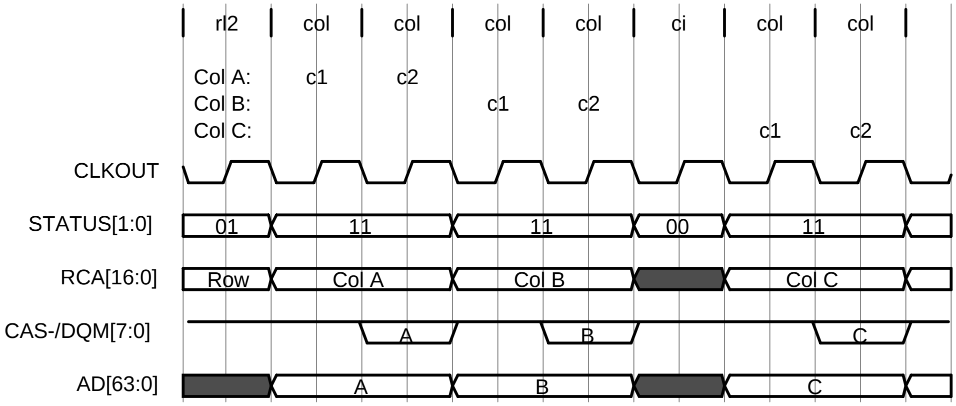
Figure 14–29.Effect of Bubbles on 2 Cycle/Column DRAM Write Column Pipeline

CAS/DQM:	none	A	none	B	ci	none	C
Col A	c1	c2					
		-	-				
Col B			c1	c2			
				-	-		
Bubble					ci	ci	
Col C						c1	c2

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–29. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning at its normal time. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for bubbles is shown in Figure 14–30. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output.

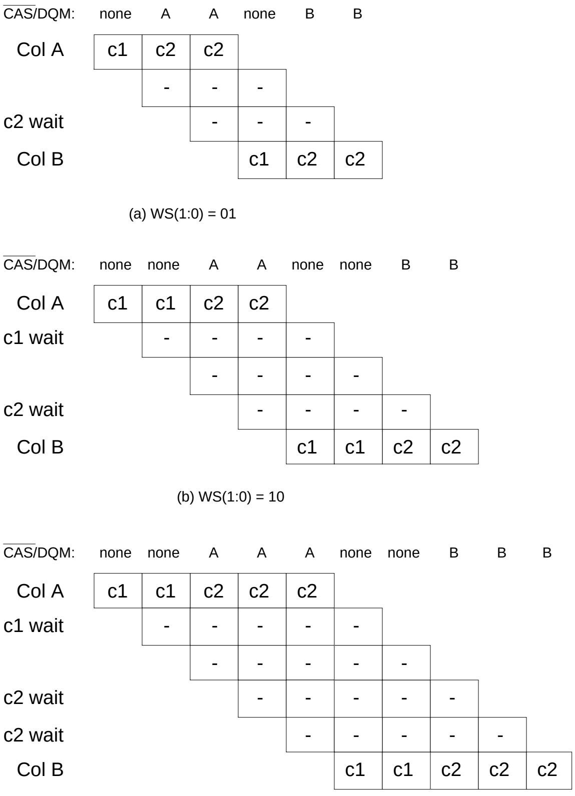
Figure 14–30.2 Cycle/Column DRAM Write (Single-Cycle Bubble)



14.4.2.3 Cycle/Column DRAM Write Column Wait States

Column accesses can be automatically extended by an integral number of states using the wait state field (**WS(1:0)**) of the configuration cache entry. The automatic wait states cause the column pipeline to stall for one cycle while the current state is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 14–31.

Figure 14–31.Effect of Wait States on 2 Cycle/Column DRAM Write Column Pipeline



(a) WS(1:0) = 01

(b) WS(1:0) = 10

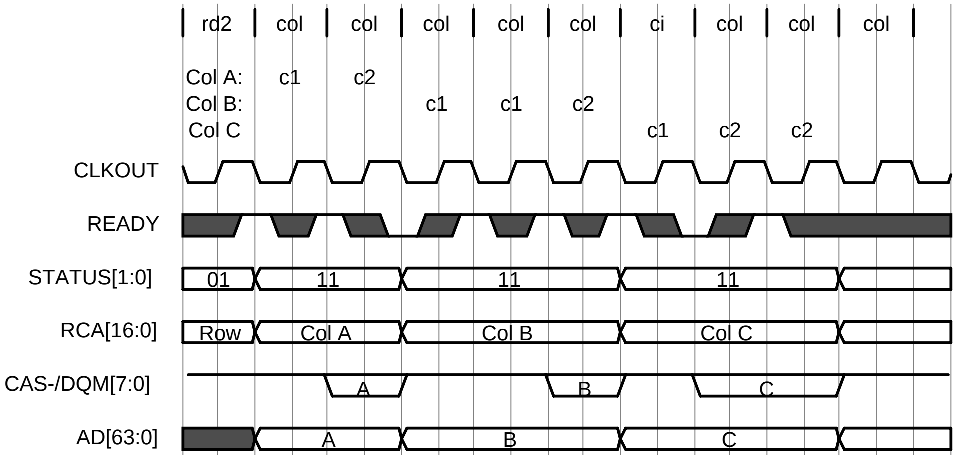
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘01’ adds a CAS low wait state to each access. This causes the pipeline to repeat the c2 stage as shown in Figure 14–31(a). A value of **WS(1:0)** = ‘10’ adds a CAS high and CAS low wait state to each access. As Figure 14–31(b) shows, this causes the pipeline to repeat the c1 and c2 stages. Programming **WS(1:0)** to ‘11’ adds a CAS high and two CAS low wait states to each column access. The c1 state is repeated and the c2 state is repeated twice as shown in Figure 14–31(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c2 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 14–32 demonstrates the addition of wait states using the READY input. A single CAS high wait state is added to the Col B access by driving READY low at the start of its c1 stage. READY is high when resampled in the next c1 stage and the Col B access proceeds. A single CAS low wait state is added to the Col C access by driving READY low at the start of its c2 stage. READY is sampled high in the next c2 stage and the Col C access completes.

Figure 14–32.2 Cycle/Column DRAM Write with Wait States



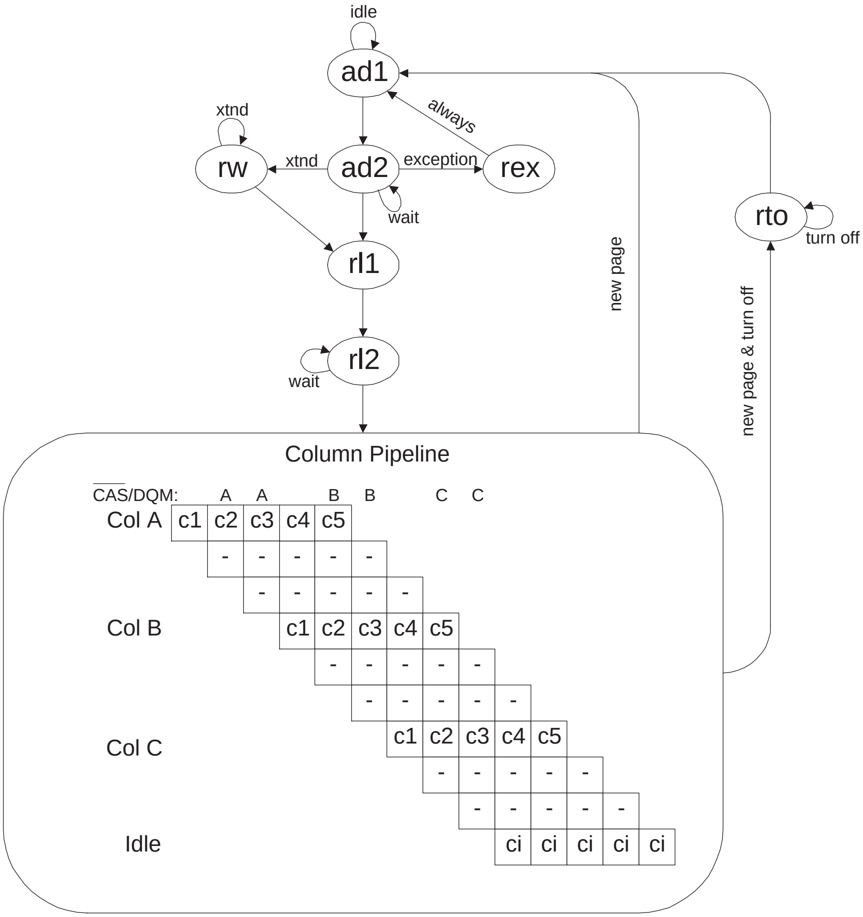
14.5 3 Cycle/Column DRAM Cycles

The 3 cycle/column DRAM timings are designed to interface to EDO DRAMs/VRAMs with slower access times. These cycles provide 1.5 cycles of column address setup time, 1.5 cycles of CAS low time and three cycles of EDO access time from CAS. Access time and column precharge time may be further increased by the use of programmable or hardware inserted wait states.

14.5.1 3 Cycle/Column DRAM Read

Figure 14–33 shows the state sequence for a 3 cycle/column DRAM read. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, a up to two rw states will automatically be inserted after ad2 to ensure a RAS high time of at least four clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 14–33.3 Cycle/Column DRAM Read States



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

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c1 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = 1x.

c2 All CAS/DQM strobes are activated to latch the column address into the memories.

c3 All CAS/DQM strobes remain active. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c3 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = 01. The c3 state is repeated twice if **WS(1:0)** = 11.

c4 The CAS/DQM strobes are deactivated at the beginning of the cycle.

c5 The data addressed in state c1 is latched by the ‘C82.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. Data from previous accesses will still be latched.

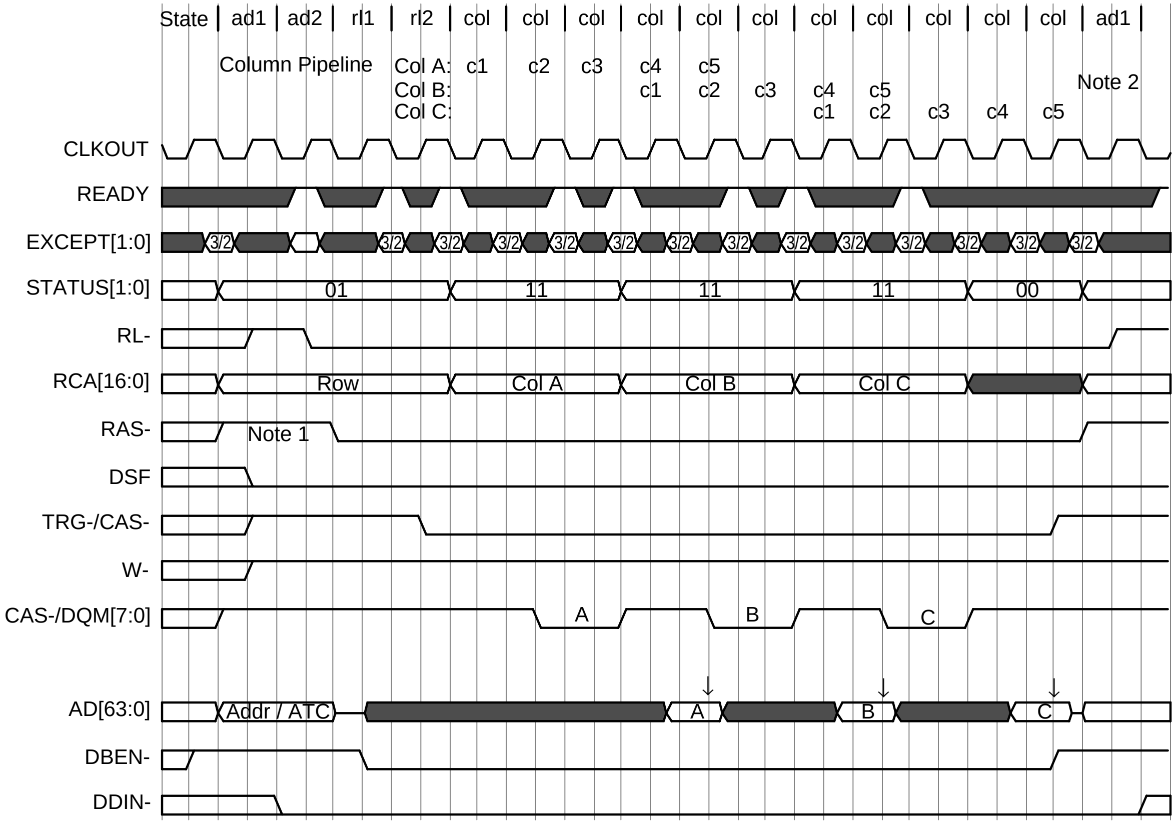
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One or more rto states will be inserted before returning to state ad1 if specified in the configuration cache entry.

14.5.1.1 3 Cycle/Column DRAM Read Example

An example of a 3 cycle/column DRAM read access is shown in Figure 14–34. The example shows a page-mode read of three column addresses. A new column access begins every three clock cycles. The read data for each column access is latched by the ‘C82 at the point indicated by the ↓ symbols. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access.

Figure 14–34.3 Cycle/Column DRAM Read



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

Note 2: Additional turn-off cycles will be inserted between the final column access and ad1 as specified by the bank configuration.

14.5.1.2 3 Cycle/Column DRAM Read With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed but the previously loaded cycles continue to completion through their c1, c2, c3, c4, and c5 stages. This effect is shown in Figure 14–35.

Figure 14–35.Effect of Bubbles on 3 Cycle/Column DRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	A	none	B	B	none	none	C	C	none	none
Col A	c1	c2	c3	c4	c5							
		-	-	-	-	-						
			-	-	-	-	-					
Col B				c1	c2	c3	c4	c5				
					-	-	-	-	-			
						-	-	-	-	-		
Bubble							ci	ci	ci	ci	ci	
Col C								c1	c2	c3	c4	c5

(a) Single Pipeline Bubble

$\overline{\text{CAS}}$/DQM:	none	A	A	none	B	B	none	none	none	C	C	none	none
Col A	c1	c2	c3	c4	c5								
		-	-	-	-	-							
			-	-	-	-	-						
Col B				c1	c2	c3	c4	c5					
					-	-	-	-	-				
						-	-	-	-	-			
Bubble1							ci	ci	ci	ci	ci		
Bubble2								ci	ci	ci	ci	ci	
Col C									c1	c2	c3	c4	c5

(b) Two Pipeline Bubbles

$\overline{\text{CAS}}$/DQM:	none	A	A	none	B	B	none	none	ci	none	C	C	none	none
Col A	c1	c2	c3	c4	c5									
		-	-	-	-	-								
			-	-	-	-	-							
Col B				c1	c2	c3	c4	c5						
					-	-	-	-	-					
						-	-	-	-	-				
Bubble1							ci	ci	ci	ci	ci			
Bubble2								ci	ci	ci	ci	ci		
Bubble3									ci	ci	ci	ci	ci	
Col C										c1	c2	c3	c4	c5

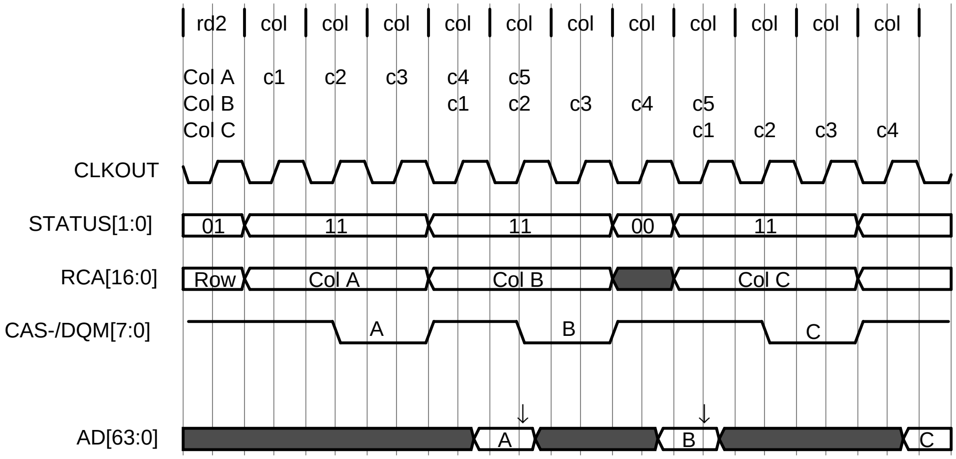
(c) Multiple Pipeline Bubbles

14.5.1.3 3 Cycle/Column DRAM Read With Single-Cycle Bubble

The column pipeline sequence for a single-cycle bubble is shown in Figure 14–35(a). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stage c4 of Col B completes as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for single cycle bubbles is shown in Figure 14–36. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning.

Figure 14–36.3 Cycle/Column DRAM Read (Single-Cycle Bubble)

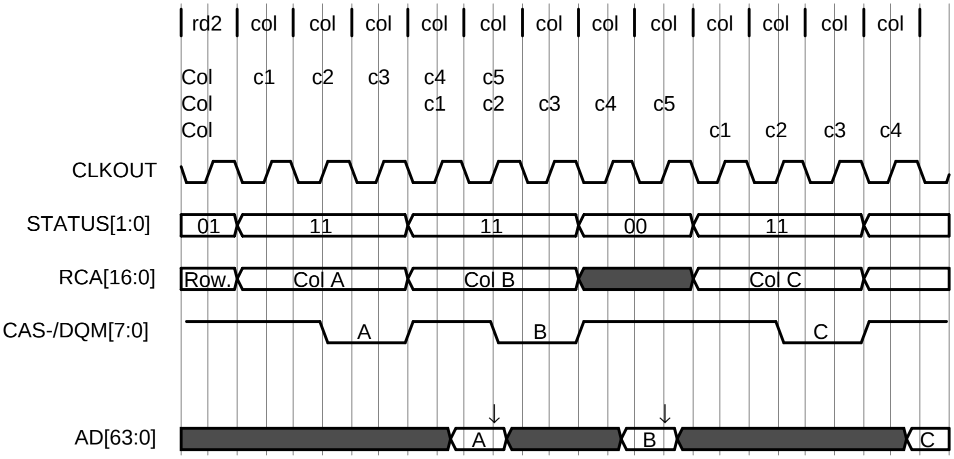


14.5.1.4 Cycle/Column Read with Two-Cycle Bubble

When a column pipeline bubble lasts for two clock cycles, the pipeline behaves as shown in Figure 14–35(b). After loading Col A and Col B into the pipeline, a bubble occurs preventing the Col C access from being loaded at its normal time. A second bubble delays the access for another cycle. Stages c4 and c5 of Col B complete as normal. In the next cycle the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for two cycle bubbles is shown in Figure 14–37. The bubble causes an idle status code to be output for two cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are inactive. During the second idle cycle, the CAS/DQM strobes remain inactive but the Col B data is latched.

Figure 14–37.3 Cycle/Column DRAM Read (Two-Cycle Bubble)

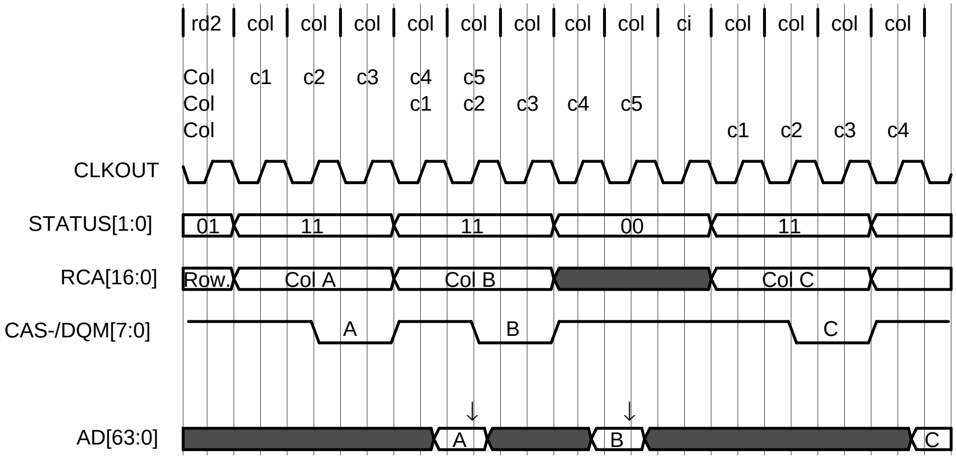


14.5.1.5 3 Cycle/Column Read with Multi-Cycle Bubble

When a column pipeline bubble lasts longer than two cycles, the pipeline drains and the memory enters the idle (ci) state until the next access is ready to begin. This pipeline sequence is shown in Figure 14–35(c). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning at its normal time. A second and third bubble delay the access for two more cycles. Stages c4 and c5 of Col B complete as normal and the pipeline then enters the idle state. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for multi-cycle bubbles is shown in Figure 14–38. The bubble causes an idle status code to be output for three cycles indicating that no new column access is beginning. During the first idle cycle, the CAS/DQM strobes are inactive. During the second idle cycle, the CAS/DQM strobes remain inactive, but the Col B data is latched. When the third idle cycle is reached, the pipeline is completely empty. The CAS/DQM strobes are inactive and no data is latched.

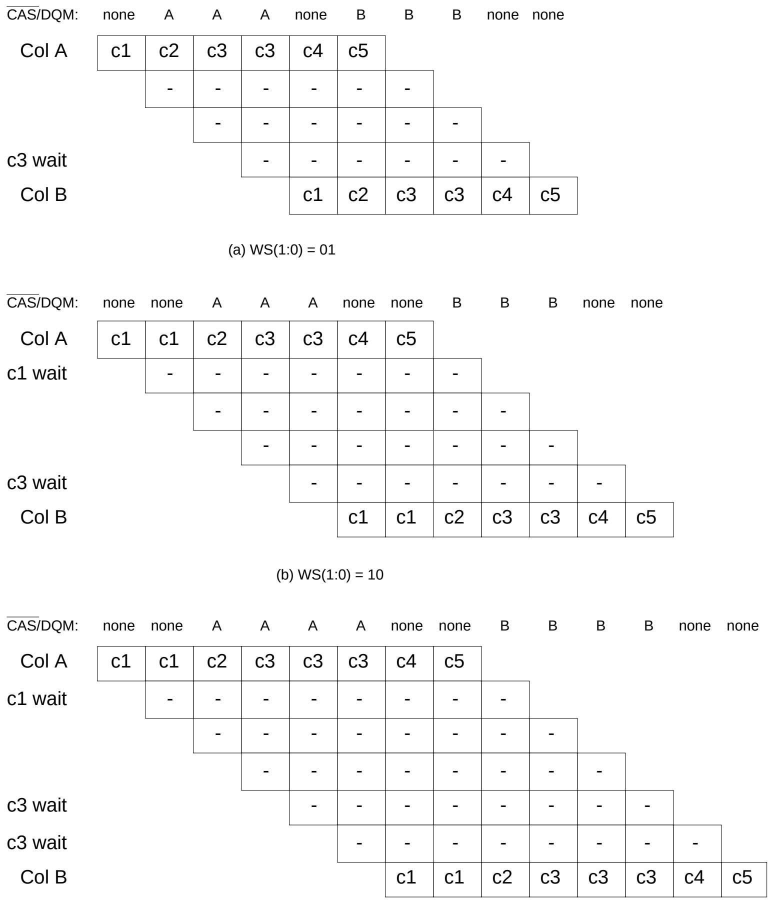
Figure 14–38.3 Cycle/Col DRAM Read (Multi-Cycle Bubbles)



14.5.1.6 3 Cycle/Column DRAM Read Column Wait States

Column accesses can be automatically extended by an integral number of states using the wait state field (**WS(1:0)**) of the configuration cache entry. The automatic wait states cause the column pipeline to stall for one cycle while the current state is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 14–39.

Figure 14–39.Effect of Wait States on 3 Cycle/Column DRAM Read Column Pipeline


CAS/DQM: none A A A none B B B none none
Col A c1 c2 c3 c3 c4 c5
- - - - - -
- - - - - -
c3 wait - - - - - -
Col B c1 c2 c3 c3 c4 c5
CAS/DQM: none none A A A none none B B B none none
Col A c1 c1 c2 c3 c3 c4 c5
c1 wait - - - - - - -
- - - - - - -
- - - - - - -
c3 wait - - - - - - -
Col B c1 c1 c2 c3 c3 c4 c5
CAS/DQM: none none A A A A none none B B B B none none
Col A c1 c1 c2 c3 c3 c3 c4 c5
c1 wait - - - - - - - -
- - - - - - - -
- - - - - - - -
c3 wait - - - - - - - -
c3 wait - - - - - - - -
Col B c1 c1 c2 c3 c3 c3 c4 c5

(a) WS(1:0) = 01

(b) WS(1:0) = 10

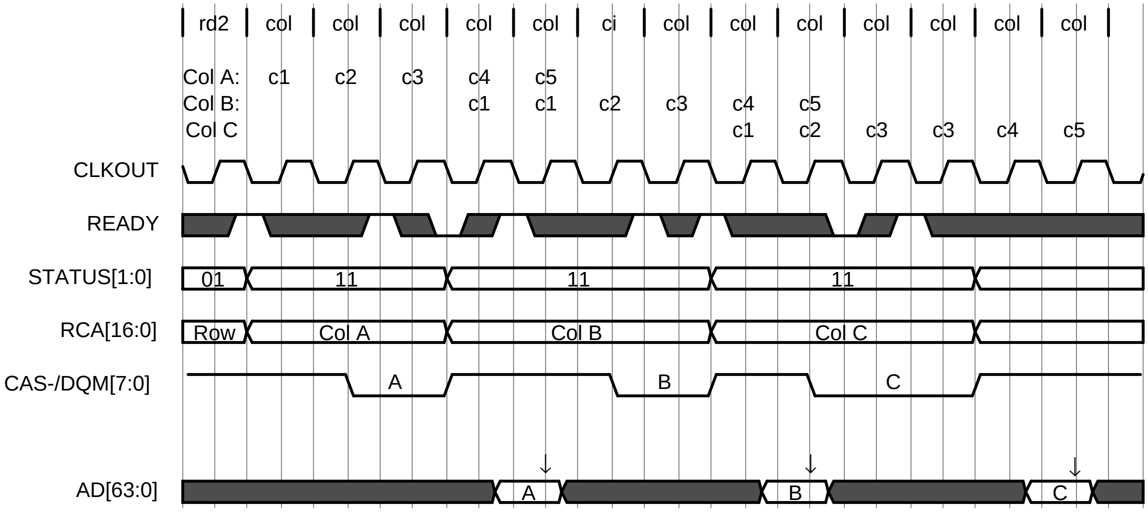
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘01’ adds a CAS low wait state to each access. This causes the pipeline to repeat the c3 stage as shown in Figure 14–39(a). A value of **WS(1:0)** = ‘10’ adds a CAS high and CAS low wait state to each access. As Figure 14–39(b) shows, this causes the pipeline to repeat the c1 and c3 stages. Programming **WS(1:0)** to ‘11’ adds a CAS high and two CAS low wait states to each column access. The c1 state is repeated and the c3 state is repeated twice as shown in Figure 14–39(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c3 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 14–40 demonstrates the addition of wait states using the READY input. A single CAS high wait state is added to the Col B access by driving READY low at the start of its c1 stage. READY is high when resampled in the next c1 stage and the Col B access proceeds. A single CAS low wait state is added to the Col C access by driving READY low at the start of its c3 stage. READY is sampled high in the next c3 stage and the Col C access completes.

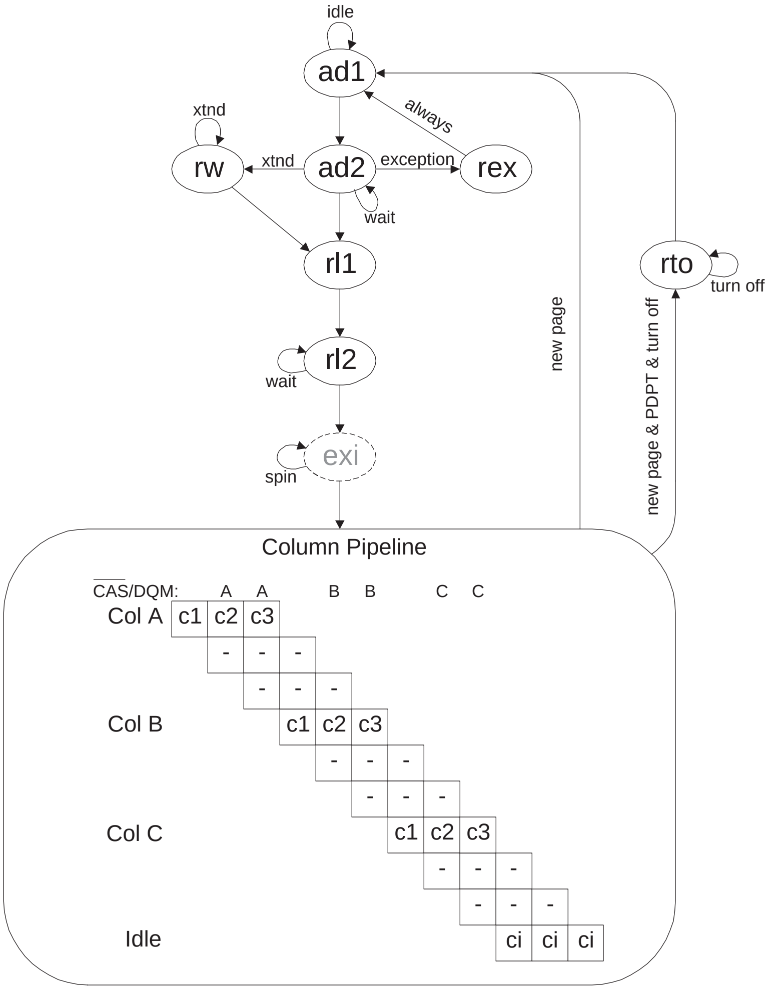
Figure 14–40.3 Cycle/Column DRAM Read with Wait States



14.5.2 3 Cycle/Column DRAM Write

Figure 14–41 shows the state sequence for a 3 cycle/column DRAM write. It provides 1.5 cycles of column address setup time, and 1.5 cycles of CAS low time. The row access consists of the ad1, ad2, rl1, rl2, and possibly rw and exi states. Wait states may be inserted by deasserting the READY input in the ad2 or rl2 states. If needed, up to two rw states will automatically be inserted after ad2 to ensure a RAS high time of at least four clock cycles. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 14–41.3 Cycle/Column DRAM Write States



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 no overlap of pipeline stages occurs. The pipeline stages are as follows:

c1 The column address, (on RCA[16:0]), and the status (STATUS[1:0]) information for the current access are output. Data is driven on the appropriate bytes of the data bus (AD[63:0]). The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c1 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = `1x`.

c2 The CAS/DQM strobes corresponding to the valid bytes on the data bus are activated to latch the column address and data into the memories.

c3 All CAS/DQM strobes remain active. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c3 state will automatically be repeated once if the wait state field in the configuration cache entry is programmed to **WS(1:0)** = `01`. The c3 state is repeated twice if **WS(1:0)** = `11`.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. The CAS/DQM strobes are inactive.

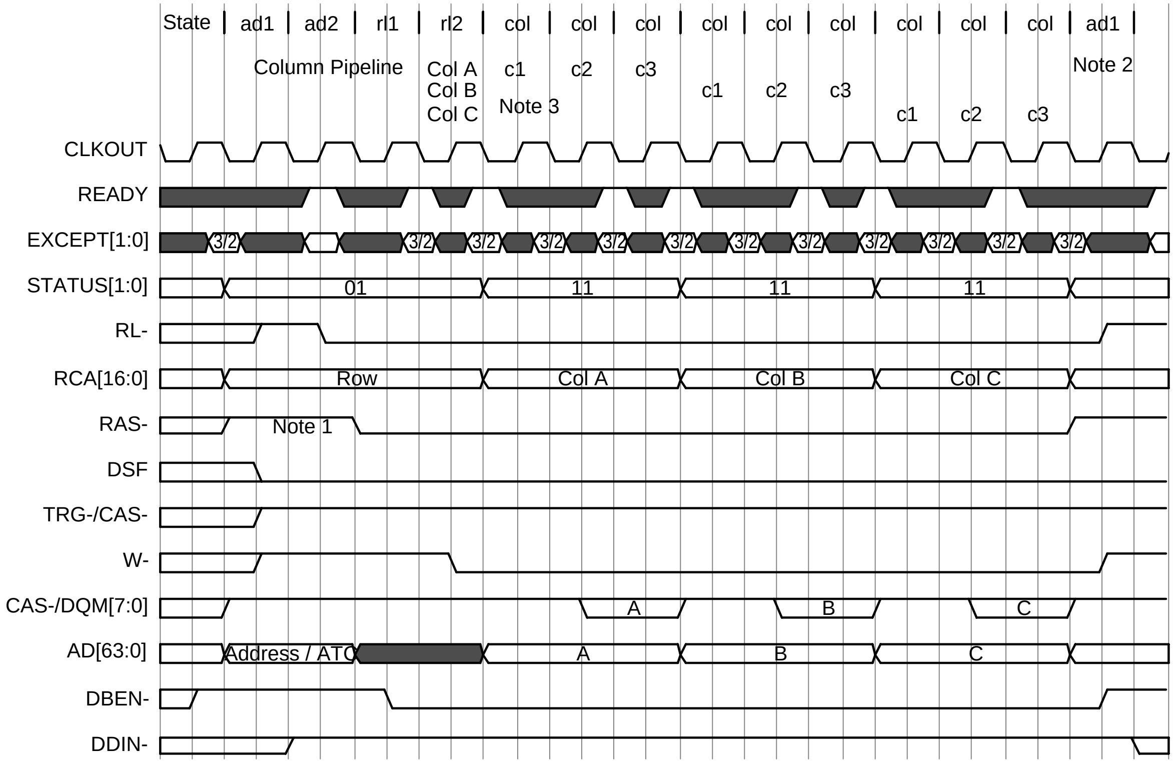
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One or more rto states will be inserted before returning to state ad1 if specified in the configuration cache entry.

14.5.2.1 3 Cycle/Column DRAM Write Example

An example of a 3 cycle/column DRAM write access is shown in Figure 14–42. The example shows a page-mode write of three column addresses. A new column access begins every three cycles. After the access to Col C is completed, the memory interface returns to the ad1 state to begin a new row access. If the write access is the result of a peripheral device packet transfer, then one or more rto state will be inserted prior to returning to ad1 as specified in the configuration cache entry.

Figure 14–42.3 Cycle/Column DRAM Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

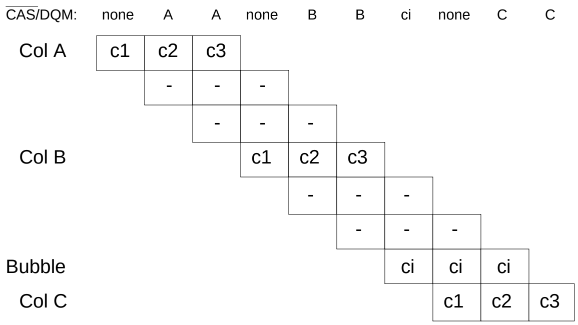
Note 2: During peripheral data transfers, turn-off cycles will be inserted prior to ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

14.5.2.2 3 Cycle/Column DRAM Write With Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed by one cycle. This effect is shown in Figure 14–43.

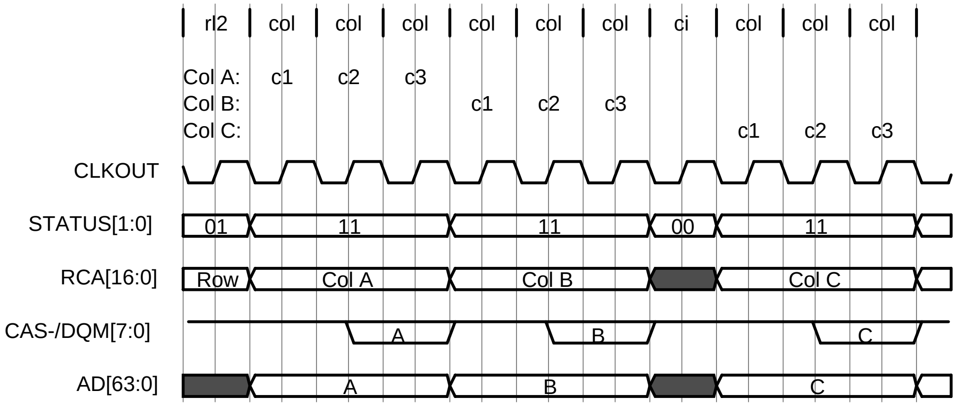
Figure 14–43.Effect of Bubbles on 3 Cycle/Column DRAM Write Column Pipeline



The column pipeline sequence for a single-cycle bubble is shown in Figure 14–43. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning at its normal time. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for bubbles is shown in Figure 14–44. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output.

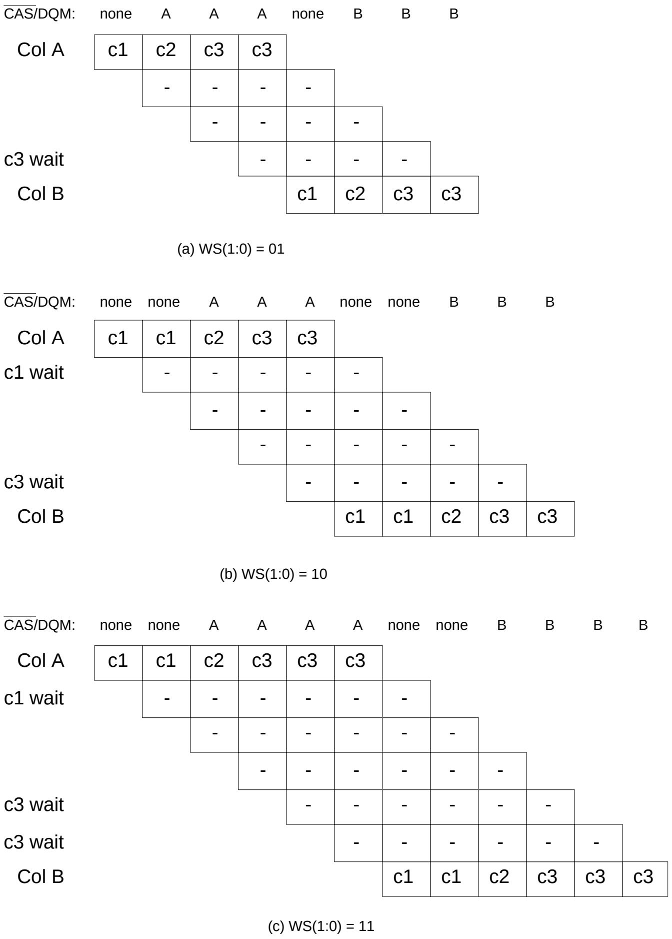
Figure 14–44.3 Cycle/Column DRAM Write (with Bubble)



14.5.2.3 3 Cycle/Column DRAM Write Column Wait States

Column accesses can be automatically extended by an integral number of states using the wait state field (**WS(1:0)**) of the configuration cache entry. The automatic wait states cause the column pipeline to stall for one cycle while the current state is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 14–45.

Figure 14–45.Effect of Wait States on 3 Cycle/Column DRAM Write Column Pipeline



(a) WS(1:0) = 01

(b) WS(1:0) = 10

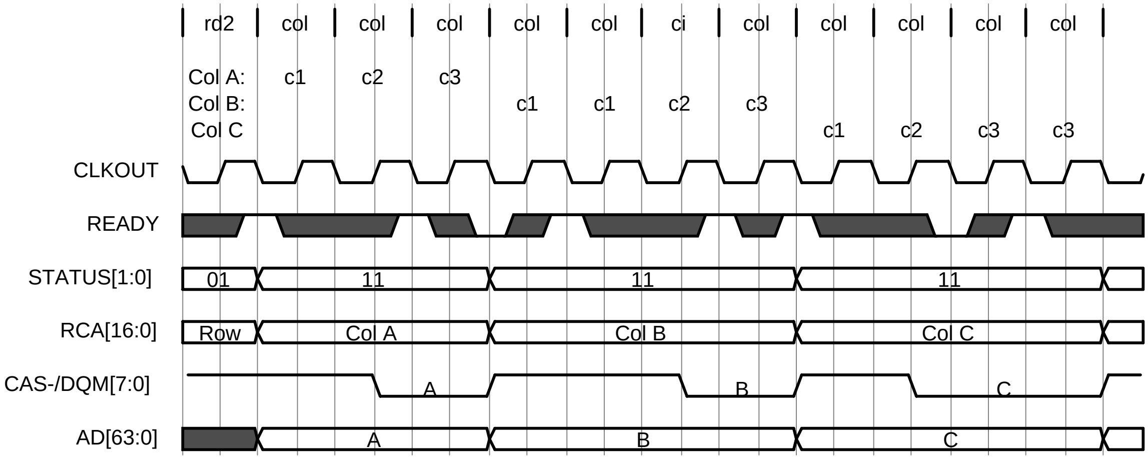
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘`01`’ adds a CAS low wait state to each access. This causes the pipeline to repeat the c3 stage as shown in Figure 14–45(a). A value of **WS(1:0)** = `10` adds a CAS high and CAS low wait state to each access. As Figure 14–45(b) shows, this causes the pipeline to repeat the c1 and c3 stages. Programming **WS(1:0)** to ‘`11`’ adds a CAS high and two CAS low wait states to each column access. The c1 state is repeated and the c3 state is repeated twice as shown in Figure 14–45(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c3 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 14–46 demonstrates the addition of wait states using the READY input. A single CAS high wait state is added to the Col B access by driving READY low at the start of its c1 stage. READY is high when resampled in the next c1 stage and the Col B access proceeds. A single CAS low wait state is added to the Col C access by driving READY low at the start of its c3 stage. READY is sampled high in the next c3 stage and the Col C access completes.

Figure 14–46.3 Cycle/Column DRAM Write with Wait States



Chapter 15

SRAM Cycles

The TMS320C82 provides four sets of SRAM timings. These timings support a wide range of devices with SRAM-type interfaces such as high-speed SRAMs, synchronous SRAMs, ROMs, FIFOs, and peripheral devices.

Topics in this chapter include:

Topic **Page**

15.1 SRAM Cycle Overview.

SRAM memory cycles facilitate the interface to SRAM and devices with similar timing characteristics such as ROM, EPROM, and peripheral devices. Four basic SRAM timings are available. An SRAM timing is selected by programming the **CT(3:0)** field in the configuration cache entry to one of the CT = `01xx` values shown in Table 15–1. The different timings allow the system designer to match the memory cycle to the access time of the selected memory device. The 2 and 3 cycle/column timings can be further modified by the addition of programmed or hardware generated wait states.

Table 15–1. SRAM Cycle Timing Selection

CT(3:0)	Cycle Timing
0 1 0 0	1 Cycle/Column Synchronous SRAM
0 1 0 1	1 Cycle/Column SRAM
0 1 1 0	2 Cycle/Column SRAM
0 1 1 1	3 Cycle/Column SRAM

Because SRAM-type devices are fully randomly addressable, there is no need for multiplexed row and column address. The ‘C82 still generates page-mode type accesses but reduces the row-time portion of the cycles to be as small as possible since all addressing generally takes place at column time.

Even though no physical page limits exist on SRAM-type devices, a logical page size limit of 128K locations (at 1-8 bytes/location depending on bus size) is imposed by the 17-bit RCA[16:0] address bus. Larger devices are supported by setting the **PS(3:0)** field in the bank configuration to ‘`1111`’ and latching the high order address bits at row time. The 17 least significant bits of the address will appear on RCA[16:0] and can be used at column time. If at any time, the higher order address bits change, the TC will automatically begin a new page access allowing the new high order bits to be latched.

15.1.1 SRAM Row Accesses

Since SRAM-type devices do not require a multiplexed addressed, the ‘C82 attempts to minimize the number of clock cycles that occur at the beginning of a page access during what would normally be row access time. All SRAM read cycles have a minimum of 3 row-time cycles which allow time for memory bank decoding and for external logic to generate row time wait states if needed. All SRAM write cycles have a minimum of 4 row-time cycles, an additional cycle being required to load write data into the pipeline. These minimums apply only to memory banks in which exceptions (faults, retries, and cache flushes) are disabled. If exceptions are enabled (by setting the **E** bit in the configuration cache entry), then the TC requires additional time to sample the EXCEPT[1:0] inputs (during the ad2 state) and ensure that an exception has not occurred before it begins loading its column pipeline. If exceptions are enabled, then one extra state (exi) is inserted during reads and two extra (exi) states are inserted during writes.

15.1.2 SRAM Read Cycles

Read cycles occur as a result of a packet transfer, a cache request, or a DEA request to the TC. SRAM read cycles transfer data or instructions from SRAM-type devices to the TMS320C82. During a read cycle, W is held high, TRG/CAS is driven low after RAS to enable memory output drivers, and DBEN and DDIN are low so that data transceivers may drive into ‘C82. During column time, the TC places the data bus (AD[63:0]) in high impedance state so that the external memory can drive data on the bus. The input data is latched by the TC in the appropriate column stage.

The TC always reads 64 bits (all CAS/DQM[7:0] signals active) and then extracts and aligns the required bytes. Invalid bytes for bus sizes of less than 64 bits are discarded.

Some SRAM read timings have an automatic turn-off cycle added after the last column access in a page. This cycle prevents bus conflict by allowing memory device output buffers to turn off before the ‘C82 begins driving a new address on the AD[63:0] bus. For devices with slow turn-off times, additional cycles can be inserted by programming the **TO(1:0)** field in the configuration cache entry.

15.1.3 Peripheral Device SRAM Read Cycles

These cycles occur as a result of a packet transfer which uses the peripheral device transfer mode. The cycle timing is the same as normal SRAM read cycles except that DBEN remains high and data is latched by a peripheral device rather than the ‘C82.

15.1.4 SRAM Write Cycles

SRAM write cycles transfer data from the TMS320C82 to external memory. Write cycles occur as a result of a packet transfer, a MP data cache write-back, or a DEA request to the TC. During a write cycle, W is driven low prior to CAS/DQM falling, TRG/CAS remains high to disable memory output drivers, and DDIN is driven high to enable data transceivers to drive into memory devices. During column-time, the TC drives the AD[63:0] with the data to be written to external memory. The TC indicates valid data bytes by activating the appropriate CAS/DQM[7:0] strobes.

15.1.5 Peripheral Device SRAM Write Cycles

These cycles occur as a result of a packet transfer which uses the peripheral device transfer mode. Because data is written by the peripheral device, the AD[63:0] is placed in high impedance by the ‘C82 at column time so that it may be driven by the peripheral. The DBEN signal remains high during the cycle so that external transceivers (if present) are disabled.

For some SRAM peripheral device writes, the cycle timing is modified from the normal SRAM write cycle. An extra rto state is added to these cycles after the last column access. This state provides time for the peripheral device to stop driving the data bus after the last column access before the ‘C82 begins driving with the address for a new page access. Additional turn-off cycles can be added by programming the **TO(1:0)** field of the configuration cache entry.

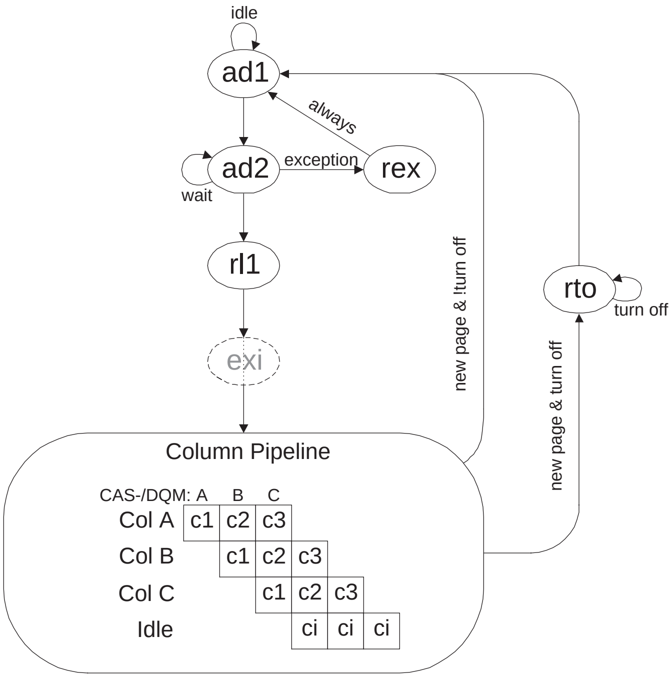
15.2 1 Cycle/Column Synchronous SRAM Cycles

Synchronous SRAM timings are designed to interface to registered synchronous SRAM devices. These devices use a clock input to clock data in and out of the device and to latch addresses. Reads occur in a pipelined fashion so that an address is clocked in on one rising clock edge and data is clocked out by the next rising clock edge. Data inputs are registered allowing data to be latched on the same clock edge as address for write cycles.

15.2.1 1 Cycle/Column Synchronous SRAM Read

The state sequence for a synchronous SRAM read is shown in Figure 15–1. The row access consists of the ad1, ad2, rl1, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in the ad2 state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of one exi state, to ensure that the minimum number of cycles between the ad2 and the column pipeline increases from two to three.

Figure 15–1. 1 Cycle/Column Synchronous SRAM Read State Diagram



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

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is in progress. All CAS/DQM strobes are activated. The rising CLKOUT edge may be used to latch the address into the memories.

c2 The CLKOUT rising edge clocks the data addressed in c1 out of the memories.

c3 The data clocked out in stage c2 is latched by the ‘C82.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

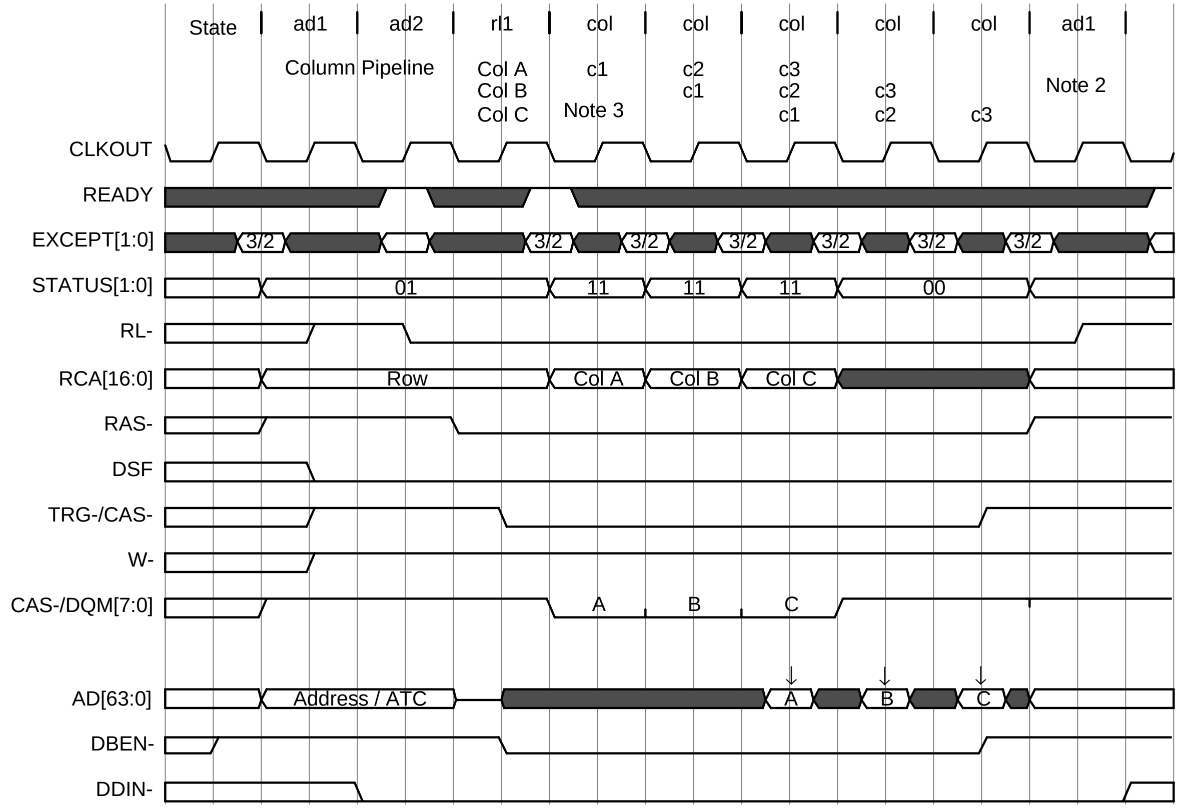
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One or more rto states will be inserted prior to returning to ad1 if specified in the configuration cache entry.

15.2.1.1 1 Cycle/Column Synchronous SRAM Read Example

An example of a 1 cycle/column synchronous SRAM read access is shown in Figure 15–2. The example shows a read of three column addresses. Read data is latched two clock cycles after being addressed (as indicated by the ↓ symbols). After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

Figure 15–2. 1 Cycle/Column Synchronous SRAM Read Timing



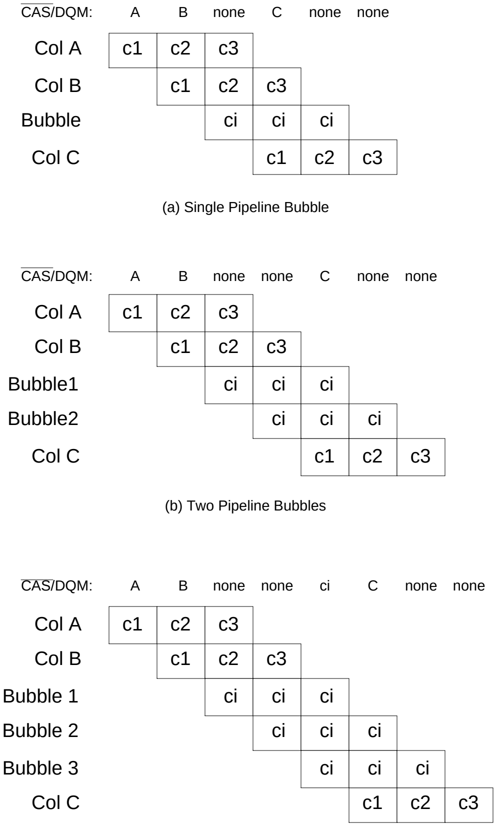
Note 2: Turn-off cycles will be inserted between the final column access and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, an additional cycle will be inserted after rl1 as required to ensure a minimum of 3 cycles between ad2 and the first c1 stage.

15.2.1.2 1 Cycle/Column Synchronous SRAM Read with Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed but previously loaded accesses continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 15–3.

Figure 15–3. Effect of Bubbles on 1 Cycle/Column Synchronous SRAM Read Pipeline



(a) Single Pipeline Bubble

(b) Two Pipeline Bubbles

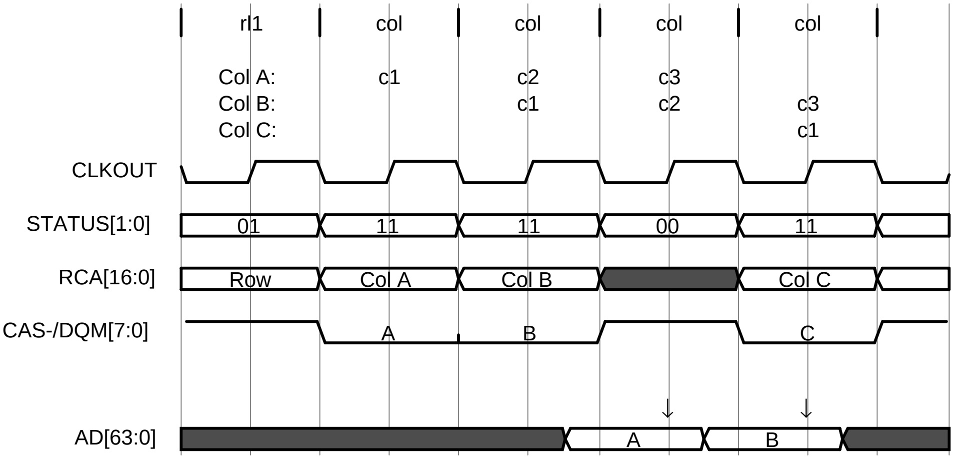
(c) Multiple Pipeline Bubbles

15.2.1.3 1 Cycle/Column Synchronous SRAM Read with Single-Cycle Bubble

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–3(a). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next clock cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for single-cycle bubbles is shown in Figure 15–4. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. Since the c3 stage of the Col A access overlaps the bubble, the Col A data is latched.

Figure 15–4. 1 Cycle/Column Synchronous SRAM Read (Single Cycle Bubble)

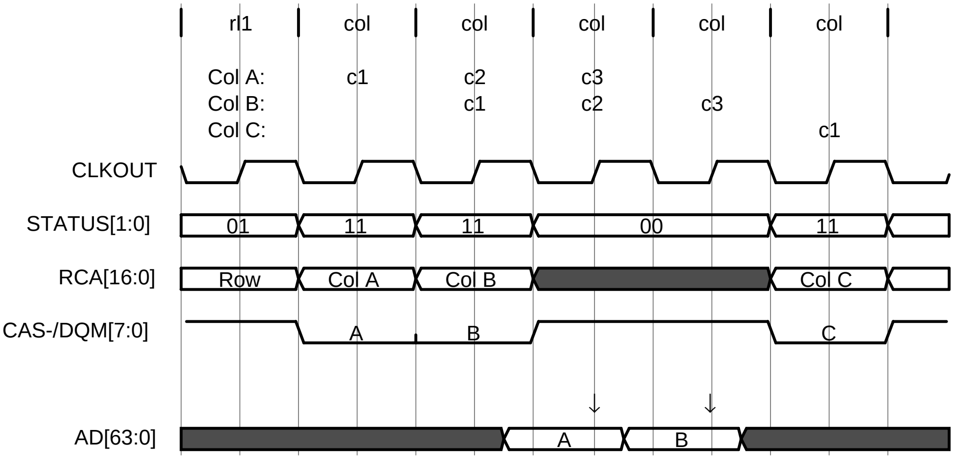


15.2.1.4 1 Cycle/Column Synchronous SRAM Read with Two-Cycle Bubble

When a column pipeline bubble lasts for two cycles, the column pipeline sequence appears as shown in Figure 15–3(b). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second bubble delays the access for one more cycle. Stages c3 of Col A and c2 & c3 of Col B complete as normal. In the next clock cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for two-cycle bubbles is shown in Figure 15–5. The bubble causes an idle status code to be output for two clock cycles indicating that no new column access is beginning. Since the c3 stages of the Col A and Col B access overlap the bubble cycles, their data is latched.

Figure 15–5. 1 Cycle/Column Synchronous SRAM Read (Two-Cycle Bubble)

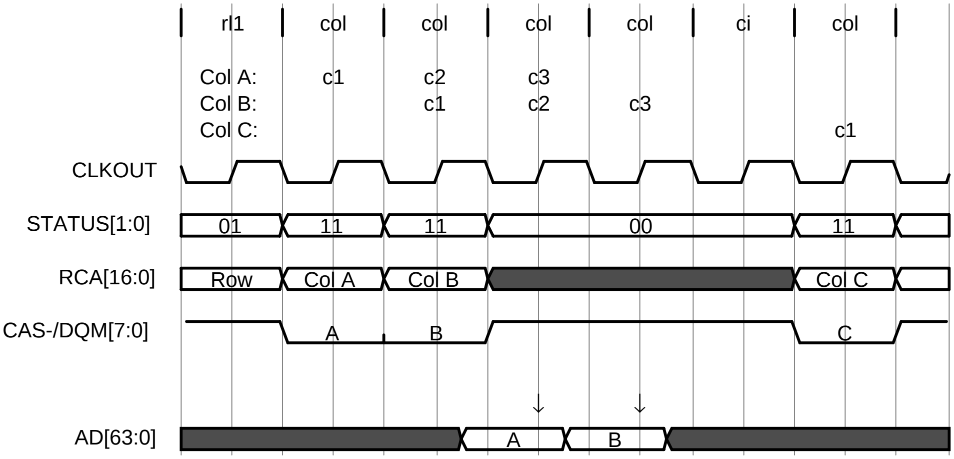


15.2.1.5 1 Cycle/Column Synchronous SRAM Read with Multi-Cycle Bubble

When a column pipeline bubble lasts longer than two cycles, the pipeline drains and the memory interface enters the idle (ci) stage until the next access is ready to begin. This pipeline sequence is shown in Figure 15–3(c). After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. A second bubble and third bubble delay the access for two more cycles. Stages c3 of Col A and c2 & c3 of Col B complete as normal and the pipeline then enters the idle stage. On the next clock cycle, the Col C access is ready to begin and the pipeline resumes normal operation.

The column-time waveform for multi-cycle bubbles is shown in Figure 15–6. The bubble causes an idle status code to be output for three clock cycles indicating that no new column access is beginning. Since the c3 stages of the Col A and Col B access overlap the bubble cycles, their data is latched during the first and second bubble cycles, respectively. When the third bubble cycle is reached, the column pipeline is completely empty and no new data is latched. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

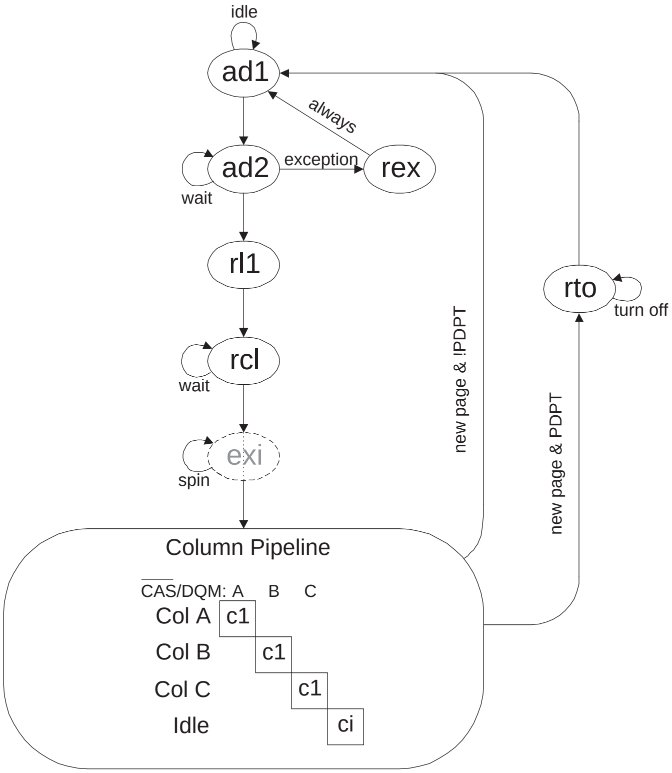
Figure 15–6. 1 Cycle/Column Synchronous SRAM Read (Multi-Cycle Bubble)



15.2.2 1 Cycle/Column Synchronous SRAM Write

The state sequence for a synchronous SRAM write is shown in Figure 15–7. The row access consists of the ad1, ad2, rl1, rcl, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in either the ad2 or rcl state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of two exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 15–7. 1 Cycle/Column Synchronous SRAM Write States.



Column accesses consist of a single-stage pipeline. A new column access can begin every cycle, and each access takes one clock cycle to complete. The column pipeline stages are as follows:

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is in progress. Data is driven on the appropriate bytes of the AD[63:0] bus and the CAS/DQM strobes corresponding to the valid bytes are. The rising CLKOUT edge may be used to latch the address and data into the memories.

ci The column idle stage occurs when no access is loaded in the TC's column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

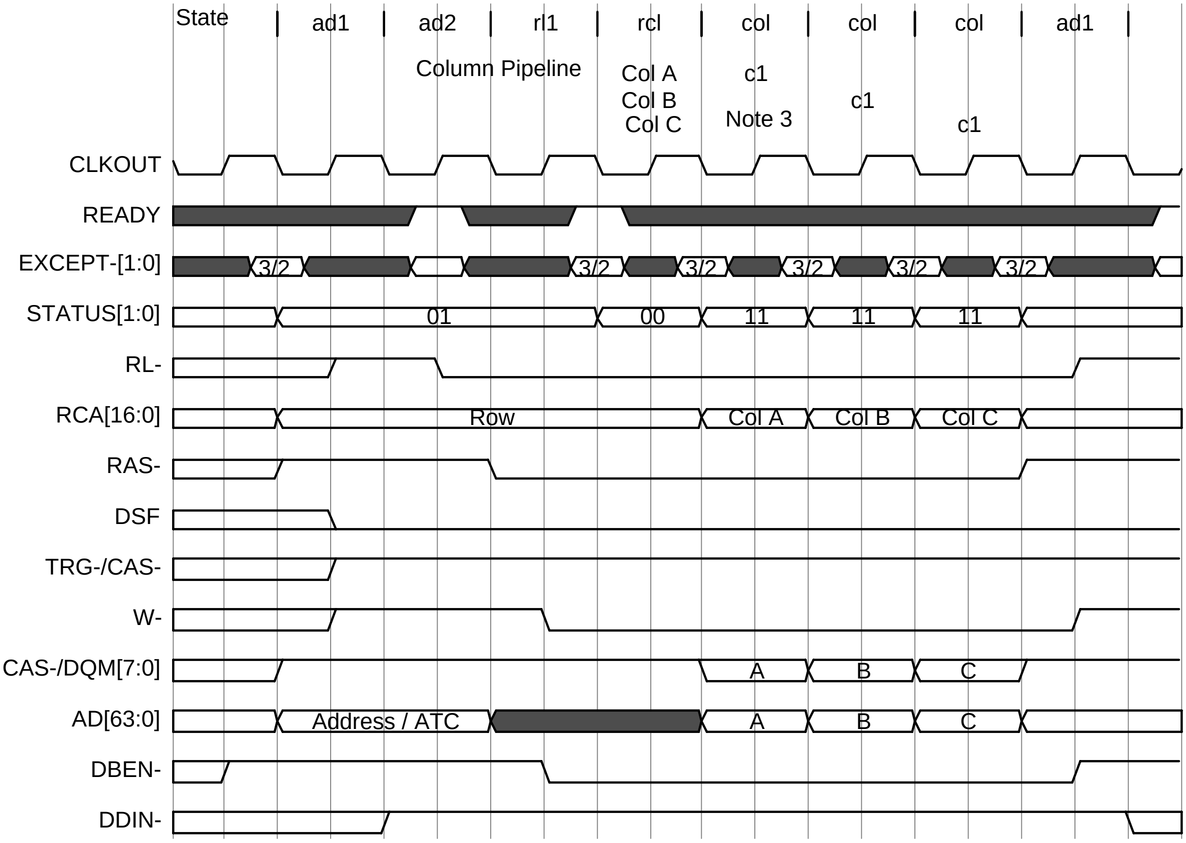
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, then one rto state will be automatically inserted prior to returning to ad1 to allow peripheral device output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.2.2.1 1 Cycle/Column Synchronous SRAM Write Example

An example of a 1 cycle/column synchronous SRAM write access is shown in Figure 15–8. The example shows a write of three column addresses. A new column access begins on each clock cycle. After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new row access.

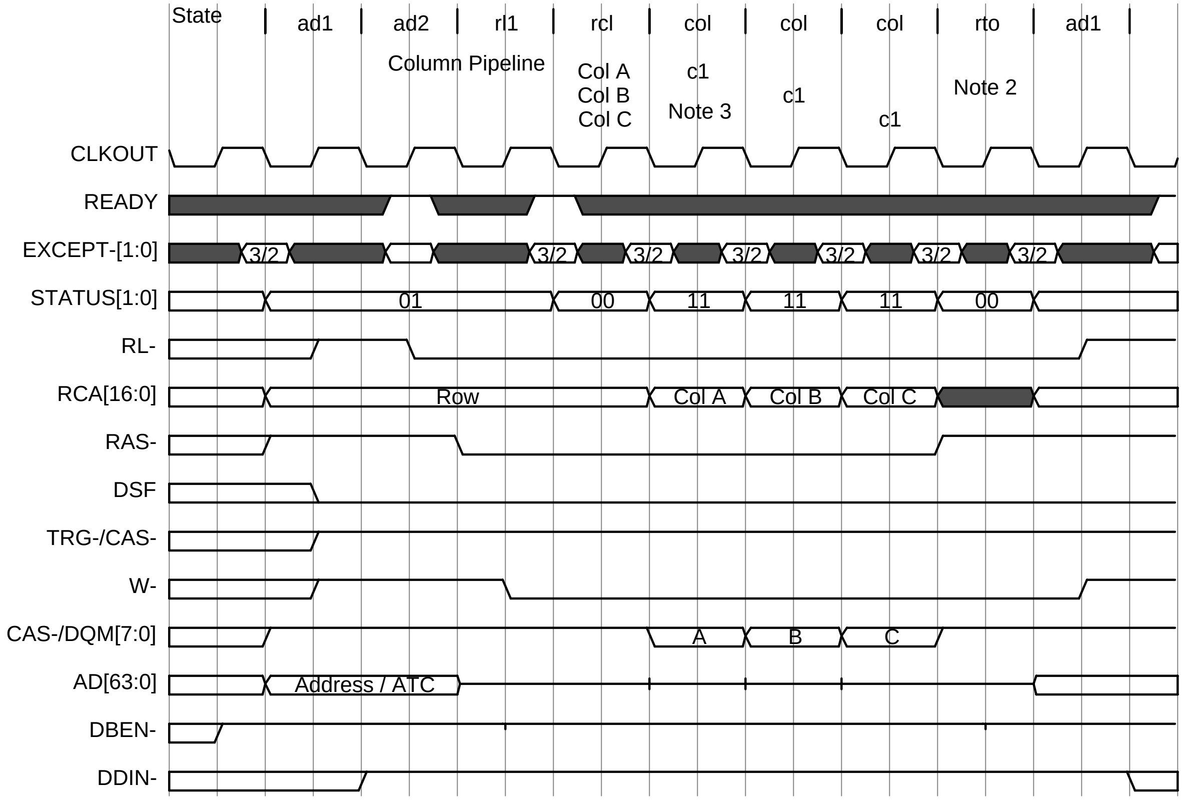
Figure 15–8. 1 Cycle/Column Synchronous SRAM Write Cycle Timing



Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The timing for peripheral device packet transfer, 1 cycle/column SRAM writes is modified by the addition of an automatic rto state at the end of the access to allow peripheral device output drivers to turn off. In addition, the AD[63:0] bus is placed in high-impedance to allow the peripheral to drive the bus and the DBEN output remains high to deactivate ‘C82 external bus transceivers, if present. An example is shown in Figure 15–9.

Figure 15–9. 1 Cycle/Column Synchronous SRAM Peripheral Device Write

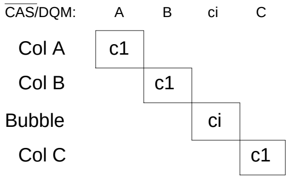


Note 2: Additional turn-off cycles will be added between rto and ad1 as specified by the bank configuration.
Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

15.2.2.2 1 Cycle/Column Synchronous SRAM Write with Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode writes, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–10.

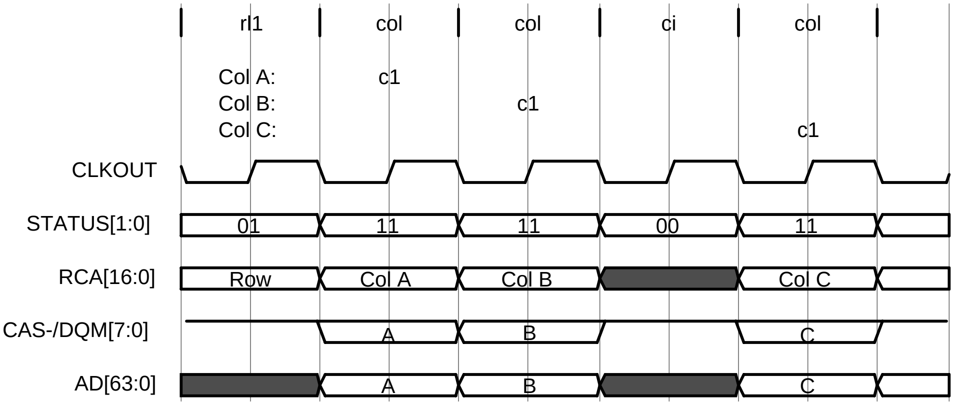
Figure 15–10.Effect of Bubbles on 1 Cycle/Column Synchronous SRAM Write Pipeline



The column pipeline sequence for a single-cycle bubble is shown in Figure 1-10. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single cycle bubbles is shown in Figure 1-11. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

Figure 15–11.1 Cycle/Column Synchronous SRAM Write (With Bubble)



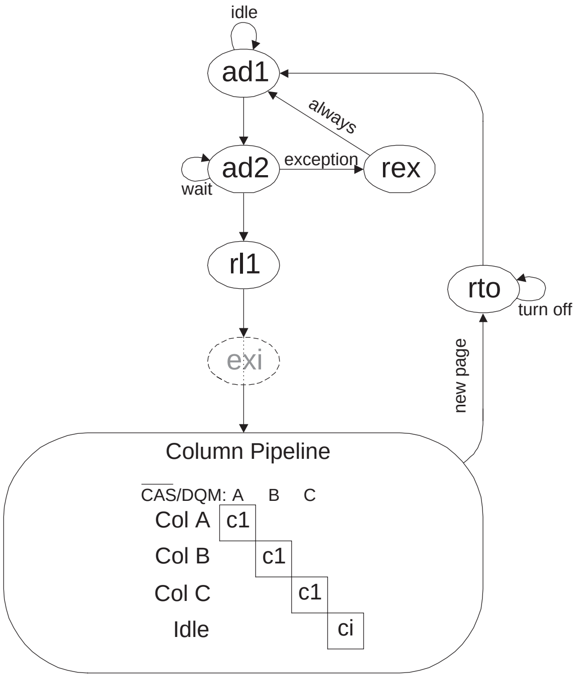
15.3 1 Cycle/Column SRAM Cycles

The 1 cycle/column SRAM timings are designed to interface to high-speed, asynchronous SRAM devices. These devices must be able to output data within one clock cycle of address. The CAS/DQM strobes are typically used as the SRAM chip enables.

15.3.1 1 Cycle/Column SRAM Read

The state sequence for a 1 cycle/column SRAM read is shown in Figure 15–12. The row access consists of the ad1, ad2, rl1, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in the ad2 state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of one exi state if necessary to ensure that the minimum number of cycles between the ad2 and the column pipeline increases from two to three.

Figure 15–12.1 Cycle/Column SRAM Read State Diagram



Column accesses consist of a single-stage pipeline. A new column access can begin on every clock cycle, and each access takes one cycle to complete. The column pipeline stages are as follows:

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is being performed. All CAS/DQM strobes are activated midway through the stage and then driven high at the end of the stage at which point input data is latched by the ‘C82.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

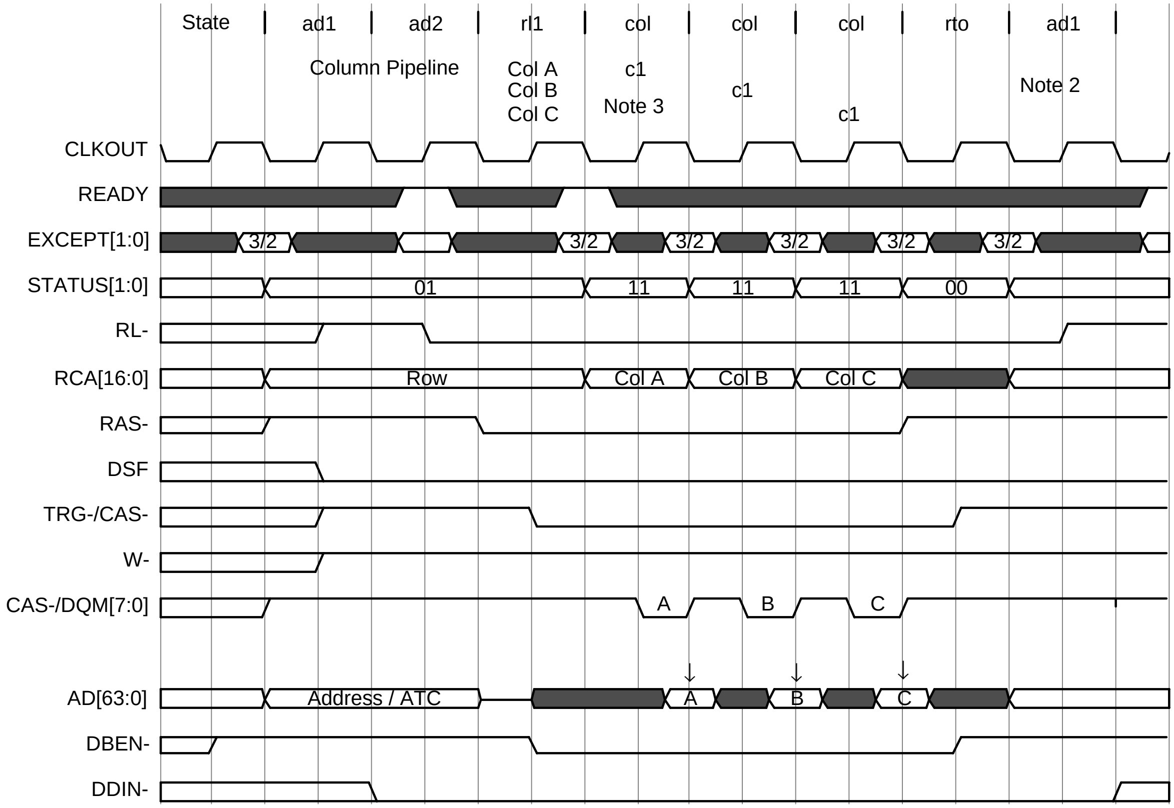
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One rto state will be automatically inserted prior to returning to ad1 to allow the memory output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.3.1.1 1 Cycle/Column SRAM Read Example

An example of a 1 cycle/column SRAM read access is shown in Figure 15–13. The example shows a read of three column addresses. Read data is latched at the end of each clock cycle in which it was addressed (as indicated by the ↓ symbols). After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

Figure 15–13.1 Cycle/Column SRAM Read Timing



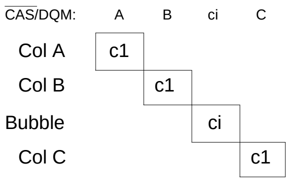
Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, an additional cycle will be inserted after rl1 as required to ensure a minimum of 3 cycles between ad2 and the first c1 stage.

15.3.1.2 1 Cycle/Column SRAM Read with Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–14.

Figure 15–14.Effect of Bubbles on 1 Cycle/Column SRAM Read Pipeline

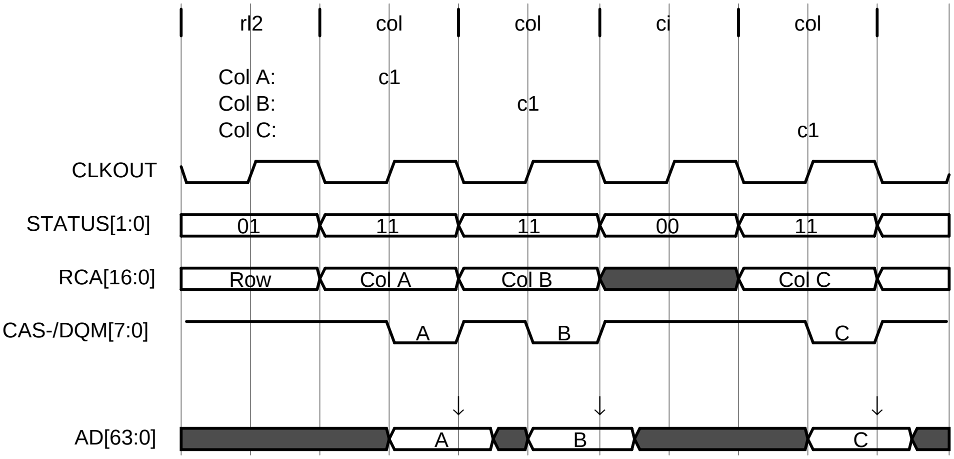


$\overline{\text{CAS}}$/DQM:	A	B	ci	C
Col A	c1			
Col B		c1		
Bubble			ci	
Col C				c1

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–14. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single-cycle bubbles is shown in Figure 15–15. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is latched. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

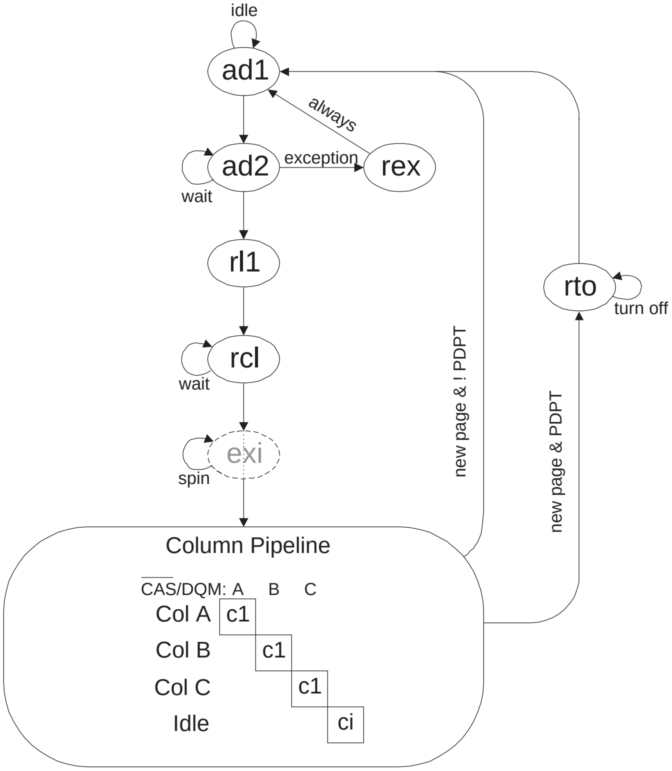
Figure 15–15.1 Cycle/Column SRAM Read (With Bubble)



15.3.2 1 Cycle/Column SRAM Write

The state sequence for a 1 cycle/column SRAM write is shown in Figure 15–16. The row access consists of the ad1, ad2, rl1, rcl, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in either the ad2 or rcl state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 15–16.1 Cycle/Column SRAM Write States



Column accesses consist of a single-stage pipeline. A new column access can begin every cycle, and each access takes one clock cycle to complete. The column pipeline stages are as follows:

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is in progress. Data is driven on the appropriate bytes of the AD[63:0] bus. The CAS/DQM strobes corresponding to the valid bytes are asserted and the beginning of the stage and deasserted midway through the stage to latch the data into the memories.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

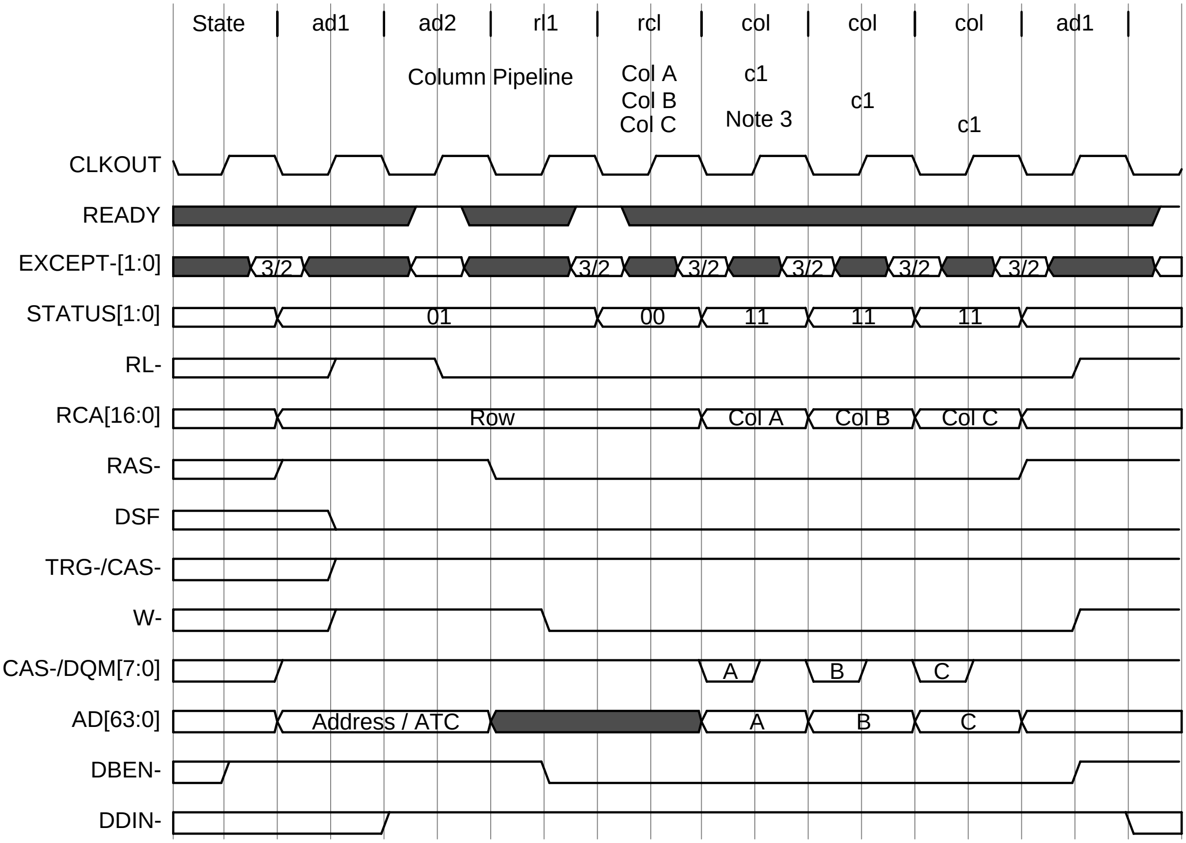
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, then one rto state will be automatically inserted prior to returning to ad1 to allow peripheral device output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.3.2.1 1 Cycle/Column SRAM Write Example

An example of a 1 cycle/column SRAM write access is shown in Figure 15–17. The example shows a write of three column addresses. A new column access begins on each clock cycle. After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new row access.

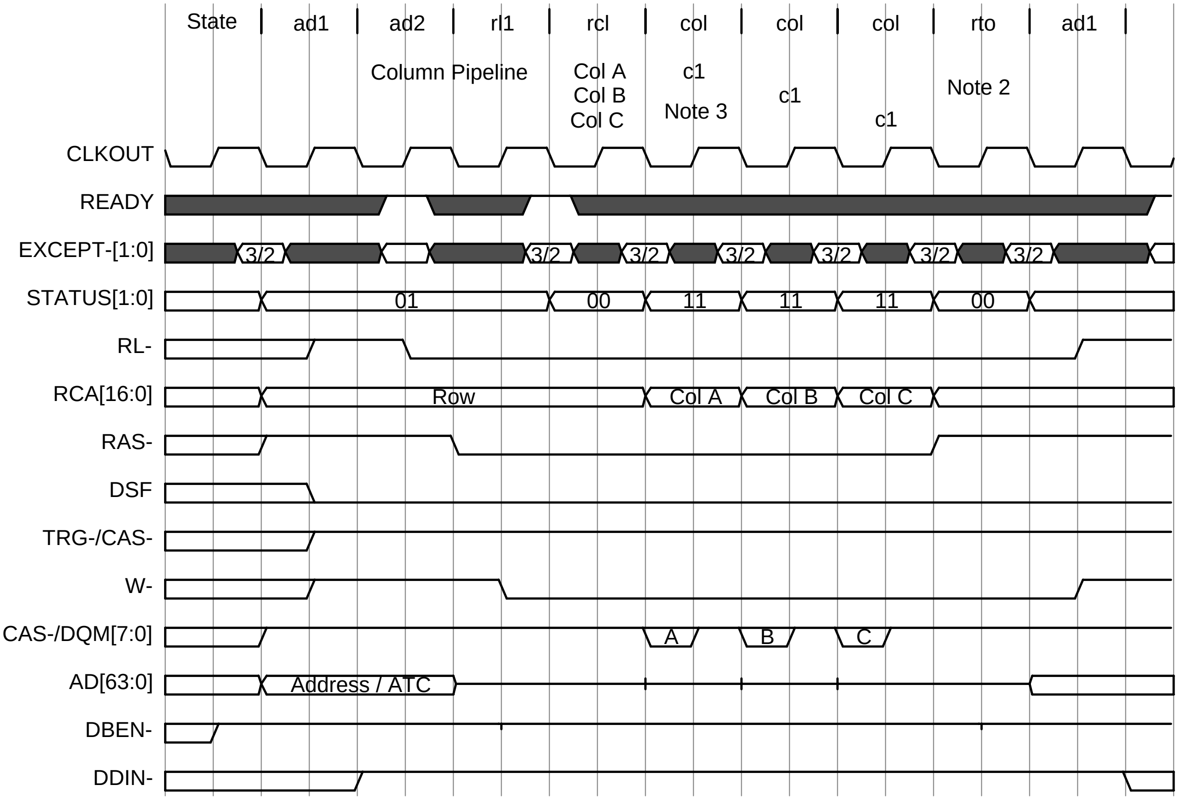
Figure 15–17.1 Cycle/Column SRAM Write Cycle Timing



Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The timing for peripheral device packet transfer 1 cycle/column SRAM writes is modified by the addition of an automatic rto state at the end of the access to allow peripheral device output drivers to turn off. In addition, the AD[63:0] bus is placed in high-impedance to allow the peripheral to drive the bus and the DBEN output remains high to deactivate ‘C82 external bus transceivers, if present. An example is shown in Figure 15–18.

Figure 15–18.1 Cycle/Column SRAM Peripheral Device Write

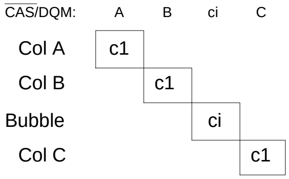


Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.
Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

15.3.2.2 1 Cycle/Column SRAM Write with Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode writes, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–19.

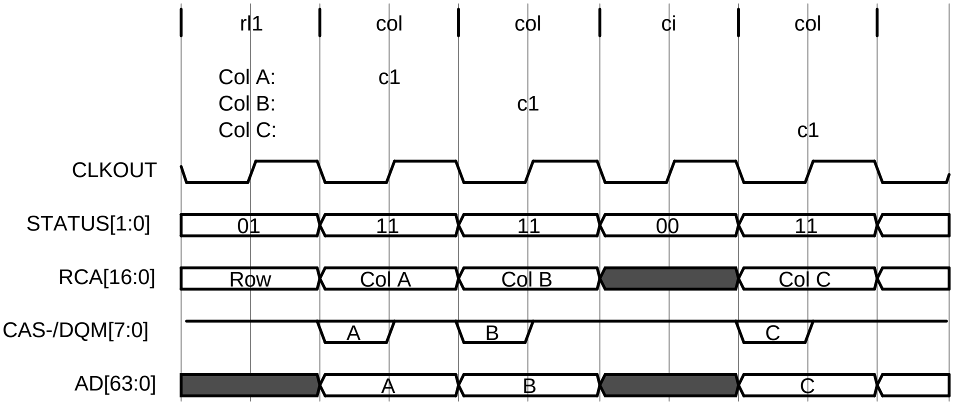
Figure 15–19.Effect of Bubbles on 1 Cycle/Column SRAM Write Pipeline



The column pipeline sequence for a single-cycle bubble is shown in Figure 15–19. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single-cycle bubbles is shown in Figure 15–20. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

Figure 15–20.1 Cycle/Column SRAM Write (With Bubble)



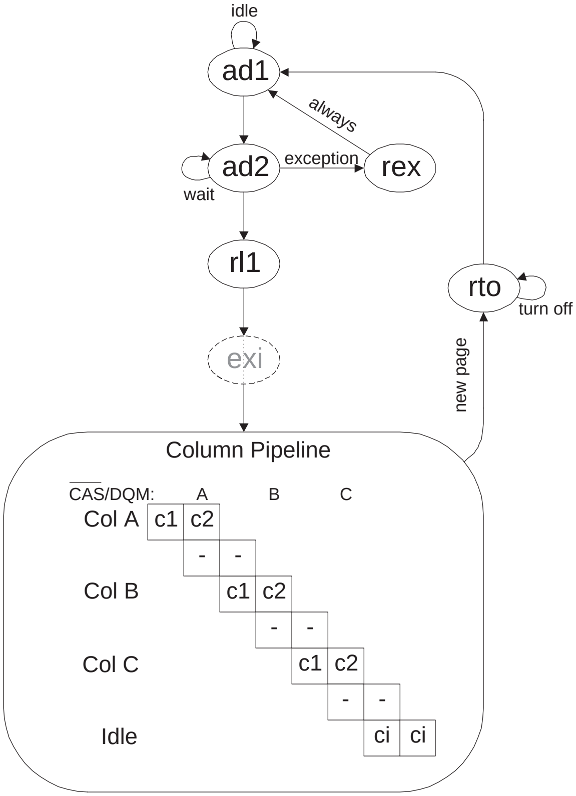
15.4 2 Cycle/Column SRAM Cycles

The 2 cycle/column SRAM timings are designed to interface to asynchronous SRAM devices with reasonably fast access times. These devices must be able to output data within two clock cycles of address. Access time may be extended by programming the wait state (**WS(1:0)**) field of the configuration cache entry or by inserting column-time wait states using the READY input. The CAS/DQM strobes are typically used as the SRAM chip enables.

15.4.1 2 Cycle/Column SRAM Read

The state sequence for a 2 cycle/column SRAM read is shown in Figure 15–21. The row access consists of the ad1, ad2, rl1, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of one exi state to ensure that the minimum number of cycles between the ad2 and the column pipeline increases from two to three.

Figure 15–21.2 Cycle/Column SRAM Read State Diagram



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

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is being performed. All CAS/DQM strobes are activated midway through the stage.

c2 The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c2 stage will automatically be repeated 1, 2, or 3 times for cache configuration entries of **WS(1:0)** = ‘01’, ‘10’, and ‘11’, respectively. Input data is latched at the end of the final c2 stage.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on

STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

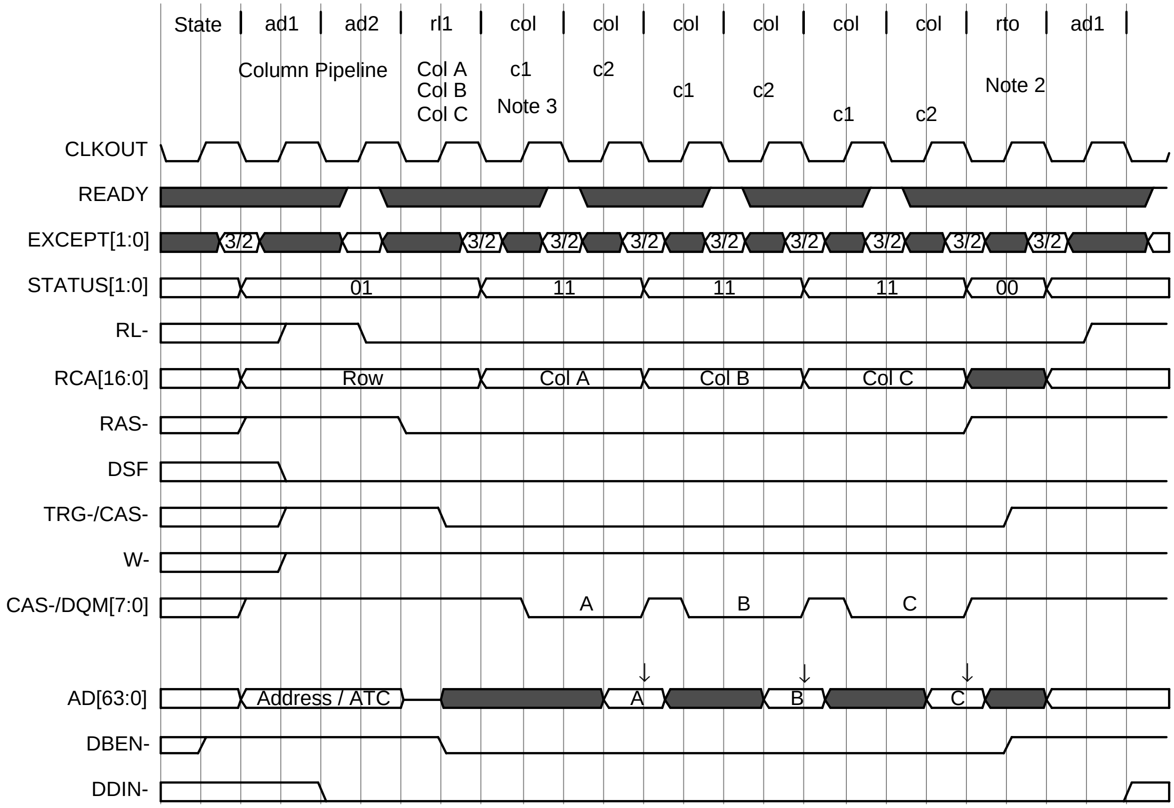
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One rto state will be automatically inserted prior to returning to ad1 to allow the memory output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.4.1.1 2 Cycle/Column SRAM Read Example

An example of a 2 cycle/column SRAM read access is shown in Figure 15–22 The example shows a read of three column addresses. Read data is latched at the end of each c2 stage (as indicated by the ↓ symbols). After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

Figure 15–22.2 Cycle/Column SRAM Read Timing



Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, an additional cycle will be inserted after rl1 as required to ensure a minimum of 3 cycles between ad2 and the first c1 stage.

15.4.1.2 2 Cycle/Column SRAM Read with Pipeline Bubbles

When a TC pipeline bubble occurs during page-mode reads, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–23.

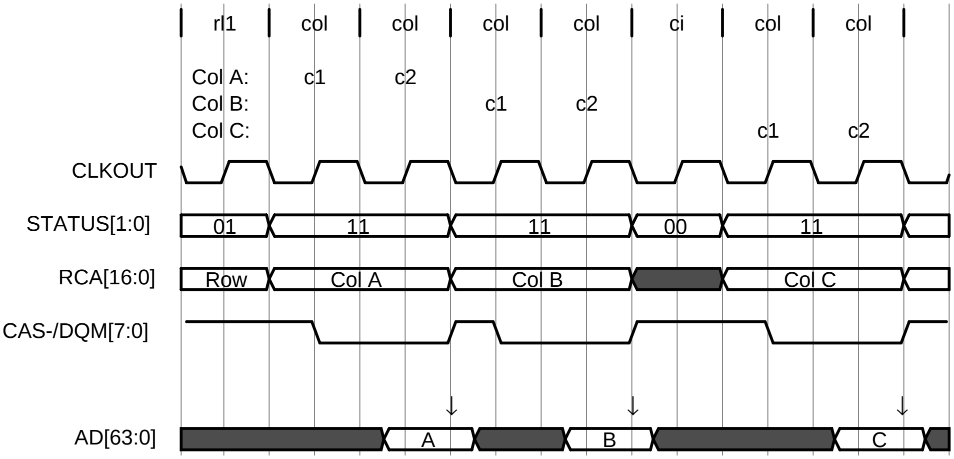
Figure 15–23.Effect of Bubbles on 2 Cycle/Column SRAM Read Pipeline

CAS/DQM:		A		B		ci	C
Col A	c1	c2					
		-	-				
Col B			c1	c2			
				-	-		
Bubble					ci	ci	
Col C						c1	c2

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–23. Two cycles after loading the Col B access into the pipeline, a bubble occurs preventing the Col C access from loading at its normal time. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single-cycle bubbles is shown in Figure 15–24. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

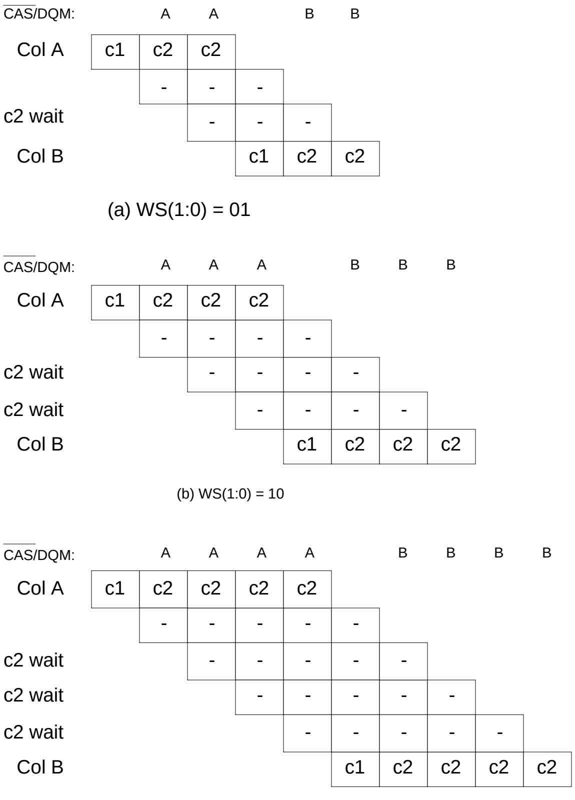
Figure 15–24.2 Cycle/Column SRAM Read (With Bubble)



15.4.1.3 2 Cycle/Column SRAM Read Wait States

Column accesses can be automatically extended by an integral number of cycles using the wait state field (**WS(1:0)**) of the configuration cache entry. Each automatic wait states causes the column pipeline to stall for one clock cycle while the current pipeline stage is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 15–25.

Figure 15–25.Effect of Wait States on 2 Cycle/Column SRAM Read Pipeline



(a) WS(1:0) = 01

(b) WS(1:0) = 10

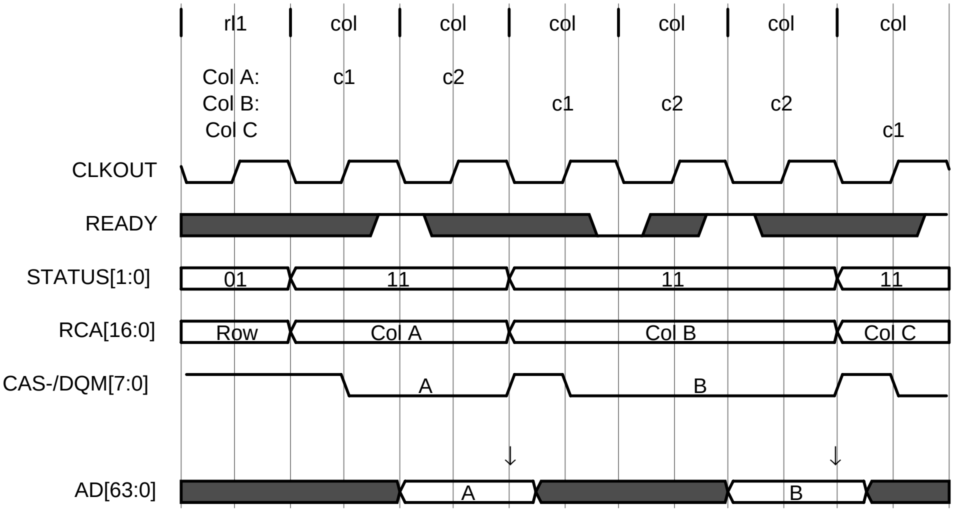
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘`01`’ adds a single wait state to each access. This causes the pipeline to repeat the c2 stage as shown in Figure 15–25(a). A value of **WS(1:0)** = `10` adds two wait states to each access. As Figure 15–25(b) shows, this causes the pipeline to repeat the c2 stage twice. The c2 stage is repeated three times for **WS(1:0)** = `11`, as seen in Figure 15–25(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c2 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 15–26 demonstrates the addition of a wait state using the READY input. A single wait state is added to the Col B access by driving READY low at the start of the c2 stage. READY is high when resampled in the repeated c2 stage and the Col B access completes.

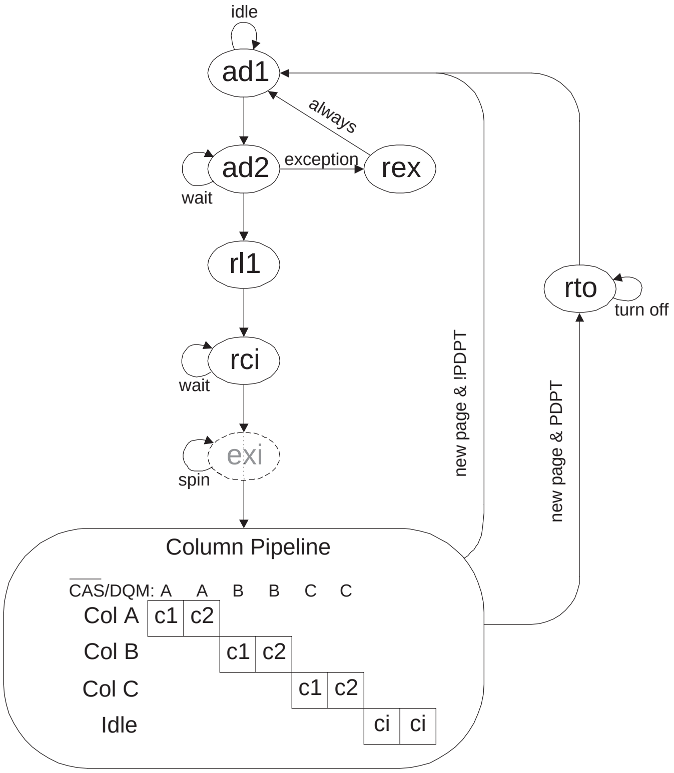
Figure 15–26.2 Cycle/Column SRAM Read with Wait State



15.4.2 2 Cycle/Column SRAM Write

The state sequence for a 2 cycle/column SRAM write is shown in Figure 15–27. The row access consists of the ad1, ad2, rl1, rcl, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in either the ad2 or rcl state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 15–27.2 Cycle/Column SRAM Write States



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

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is in progress. Data is driven on the appropriate bytes of the AD[63:0] bus. The CAS/DQM strobes corresponding to the valid bytes are asserted at the beginning of the stage.

c2 The address and data continue to be driven. The active CAS/DQM strobes are deasserted midway through the stage to latch data into the memories. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c2 stage will automatically be repeated 1, 2, or 3 times for cache configuration entries of **WS(1:0)** = ‘`01`’, ‘`10`’, and ‘`11`’, respectively.

ci The column idle stage occurs when no access is loaded in the TC's column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

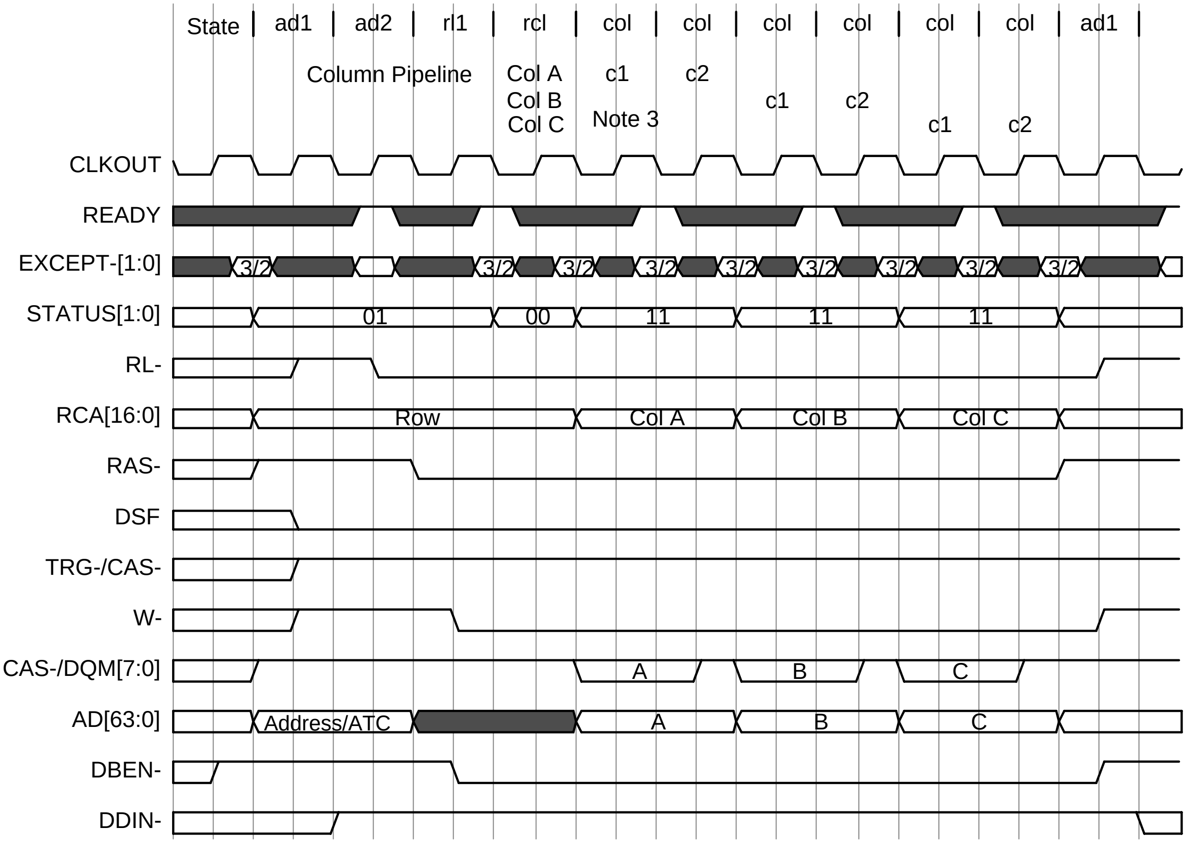
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, then one rto state will be automatically inserted prior to returning to ad1 to allow peripheral device output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.4.2.1 2 Cycle/Column SRAM Write Example

An example of a 2 cycle/column SRAM write access is shown in Figure 15–28. The example shows a write of three column addresses. A new column access begins every two clock cycles. After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

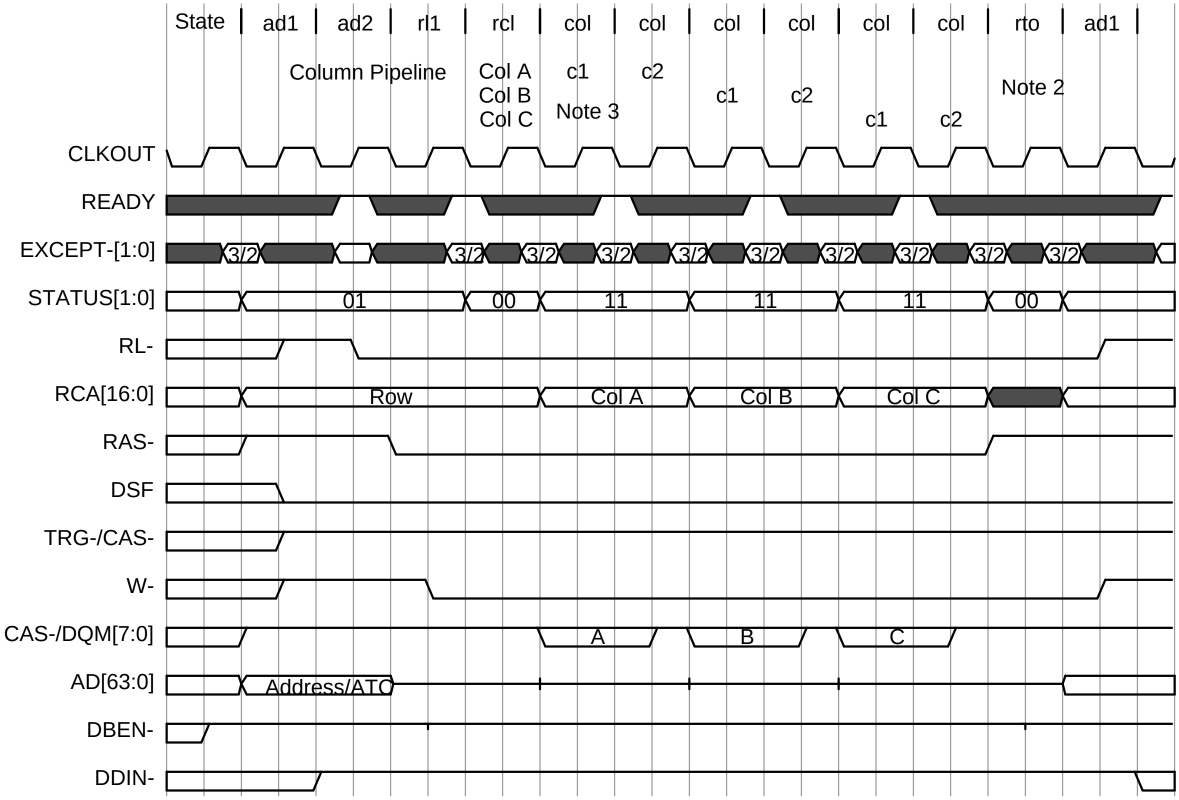
Figure 15–28.2 Cycle/Column SRAM Write Cycle Timing



Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The timing for peripheral device packet transfer 2 cycle/column SRAM writes is modified by the addition of an automatic rto state at the end of the access to allow peripheral device output drivers to turn off. In addition, the AD[63:0] bus is placed in high-impedance to allow the peripheral to drive the bus and the DBEN output remains high to deactivate ‘C82 external bus transceivers, if present. An example is shown in Figure 15–29.

Figure 15–29.2 Cycle/Column SRAM Peripheral Device Write



Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

15.4.2.2 2 Cycle/Column SRAM Write with Pipeline Bubbles

When a TC pipeline bubble occurs during writes, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–30.

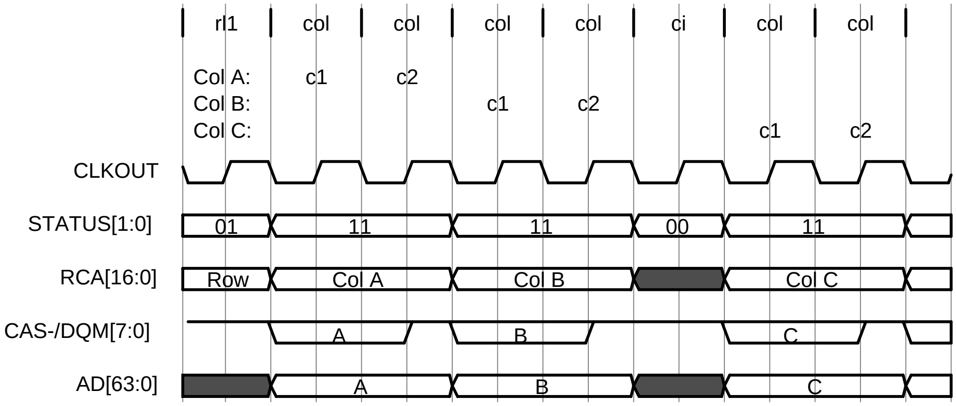
Figure 15–30.Effect of Bubbles on 2 Cycle/Column SRAM Write Pipeline

CAS/DQM:	A	A	B	B	ci	C	C
Col A	c1	c2					
		-	-				
Col B			c1	c2			
				-	-		
Bubble					ci	ci	
Col C						c1	c2

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–30. A bubble occurs after loading the Col B access into the pipeline preventing the Col C access from being loaded at its normal time. After the bubble, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single cycle bubbles is shown in Figure 15–31. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

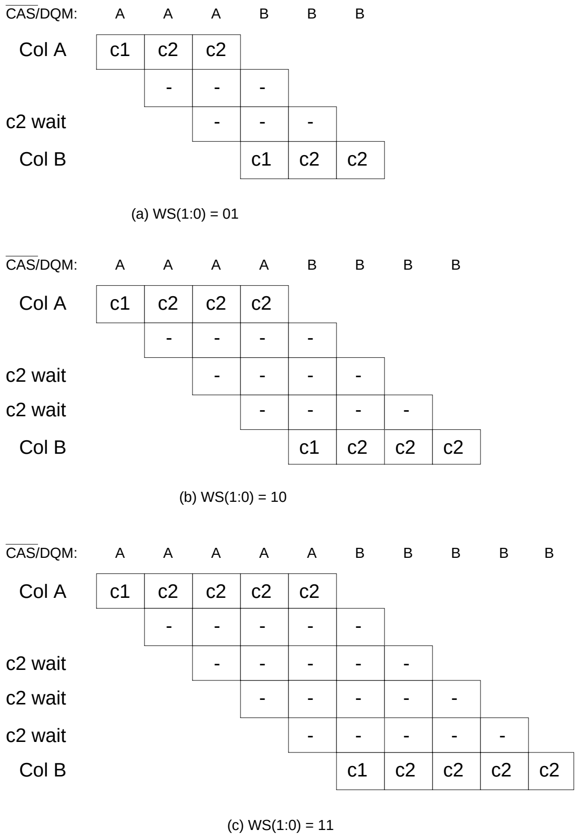
Figure 15–31.2 Cycle/Column SRAM Write (With Bubble)



15.4.2.3 2 Cycle/Column SRAM Write Wait States

Column accesses can be automatically extended by an integral number of cycles using the wait state field (**WS(1:0)**) of the configuration cache entry. Each automatic wait states causes the column pipeline to stall for one clock cycle while the current pipeline stage is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 15–32.

Figure 15–32.Effect of Wait States on 2 Cycle/Column Write Pipeline



(a) WS(1:0) = 01

(b) WS(1:0) = 10

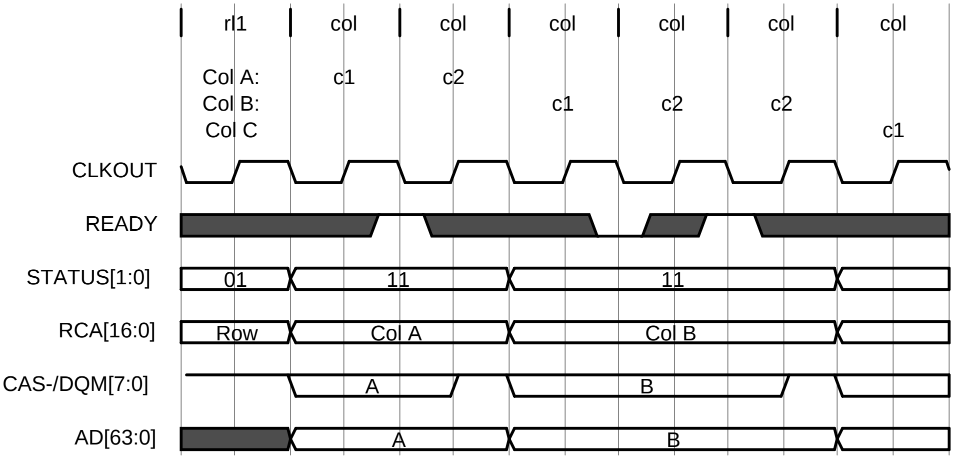
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘`01`’ adds a single wait state to each access. This causes the pipeline to repeat the c2 stage as shown in Figure 15–32(a). A value of **WS(1:0)** = `10` adds two wait states to each access. As Figure 15–32(b) shows, this causes the pipeline to repeat the c2 stage twice. The c2 stage is repeated three times for **WS(1:0)** = `11`, as seen in Figure 15–32(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c2 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 15–33 demonstrates the addition of a wait state using the READY input. A single wait state is added to the Col B access by driving READY low at the start of the c2 stage. READY is high when resampled in the repeated c2 stage and the Col B access completes.

Figure 15–33.2 Cycle/Column Write with Wait State



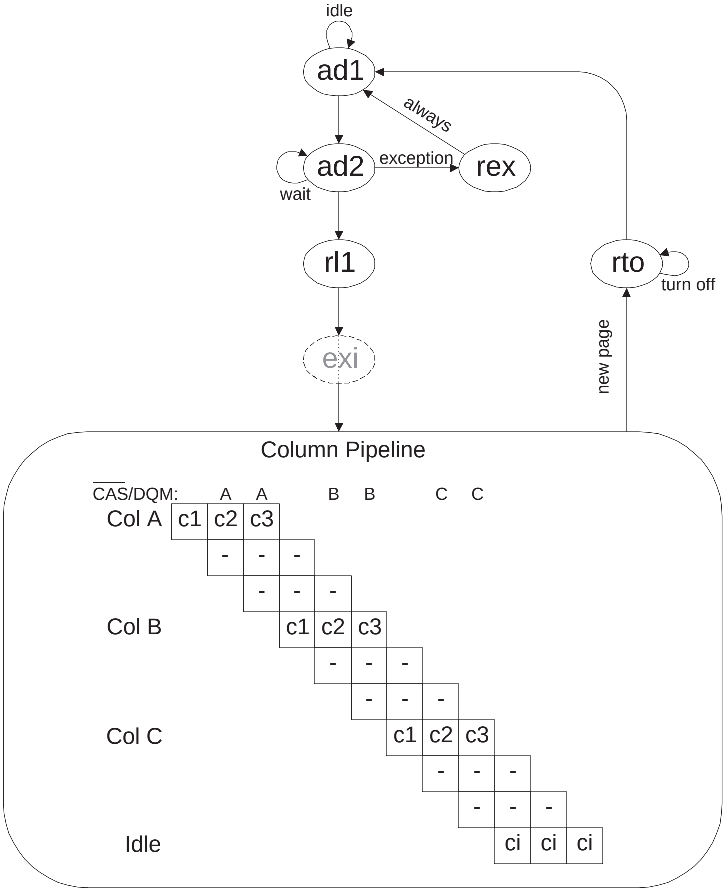
15.5 3 Cycle/Column SRAM Cycles

The 3 cycle/column SRAM timings are designed to interface to asynchronous SRAM devices with slow access times. These devices must be able to output data within three clock cycles of address. The timing is also intended to be used for very slow devices with SRAM-like interfaces such as EPROMs and peripheral devices. For these devices, the access time may be extended by programming the wait state (**WS(1:0)**) field of the configuration cache entry or by inserting column-time wait states using the READY input. The CAS/DQM strobes are typically used as the SRAM chip enables.

15.5.1 3 Cycle/Column SRAM Read

The state sequence for a 3 cycle/column SRAM read is shown in Figure 15–34. The row access consists of the ad1, ad2, rl1, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of one exi state if necessary to ensure that the minimum number of cycles between the ad2 and the column pipeline increases from two to three.

Figure 15–34.3 Cycle/Column SRAM Read State Diagram



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

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is being performed. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high.

c2 All CAS/DQM strobes are asserted.

c3 The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls and the stage is repeated until READY is sampled high. The c3 stage will automatically be repeated 1, 2, or 3 times for cache configuration entries of **WS(1:0)** = ‘01’, ‘10’, and ‘11’, respectively. Input data is latched at the end of the final c3 stage.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

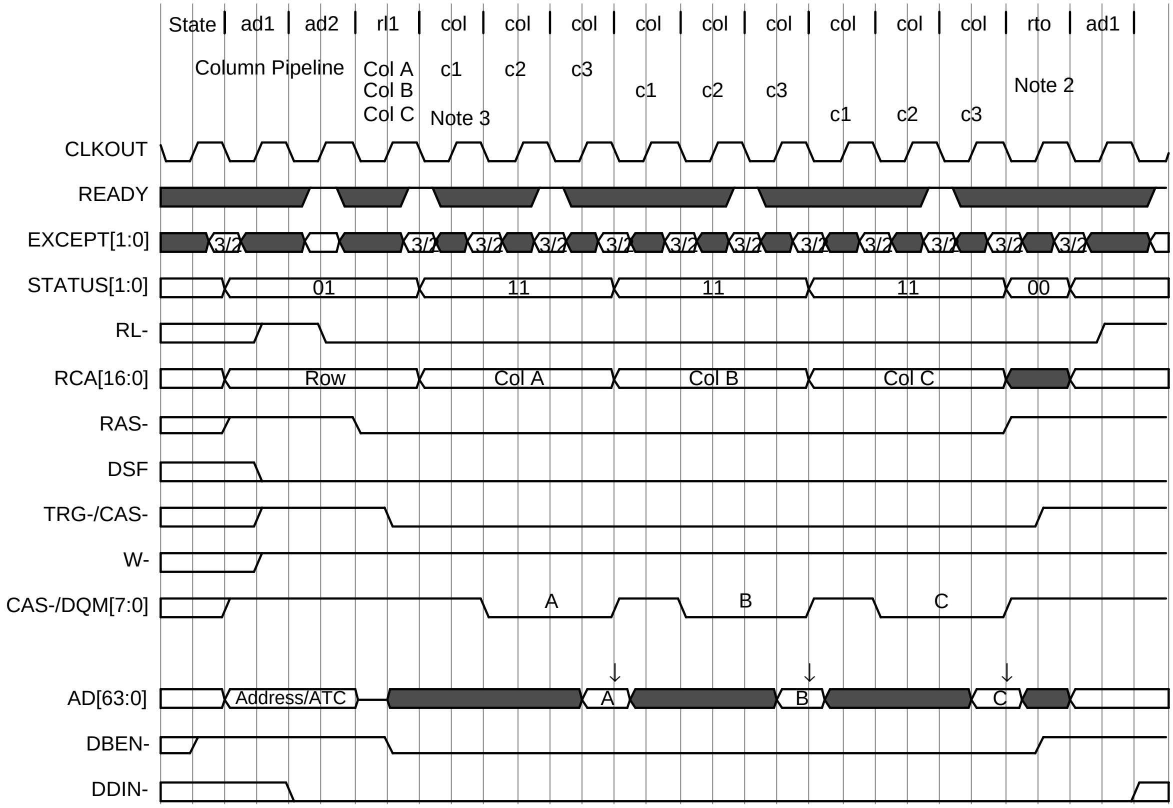
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new row access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new row access, then the column pipeline sequence will begin again. One rto state will be automatically inserted prior to returning to ad1 to allow the memory output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.5.1.1 3 Cycle/Column SRAM Read Example

An example of a 3 cycle/column SRAM read access is shown in Figure 15–35. The example shows a page-mode read of three column addresses. Read data is latched at the end of each c3 stage (as indicated by the ↓ symbols). After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

Figure 15–35.3 Cycle/Column SRAM Read Timing



Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, an additional cycle will be inserted after rl1 as required to ensure a minimum of 3 cycles between ad2 and the first c1 stage.

15.5.1.2 3 Cycle/Column SRAM Read with Pipeline Bubbles

When a TC pipeline bubble occurs during reads, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–36.

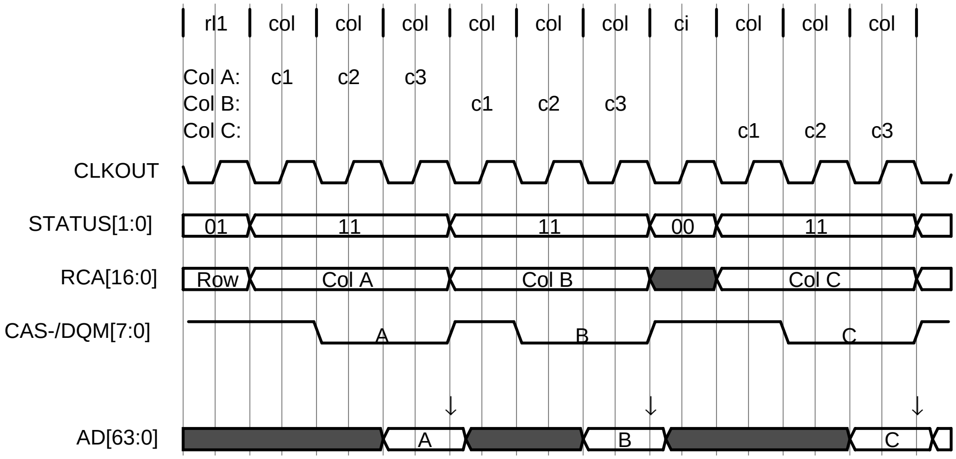
Figure 15–36.Effect of Bubbles on 3 Cycle/Column SRAM Read Pipeline

CAS/DQM:	A	A	A	B	B	B	ci	C	C	C
Col A	c1	c2	c3							
		-	-	-						
			-	-	-					
Col B				c1	c2	c3				
					-	-	-			
						-	-	-		
Bubble							ci	ci	ci	
Col C								c1	c2	c3

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–36. Three cycles after loading Col B into the pipeline, a bubble occurs preventing the Col C access from loading at its normal time. After the bubble, the Col C access is ready to be loaded and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single-cycle bubbles is shown in Figure 15–37. The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

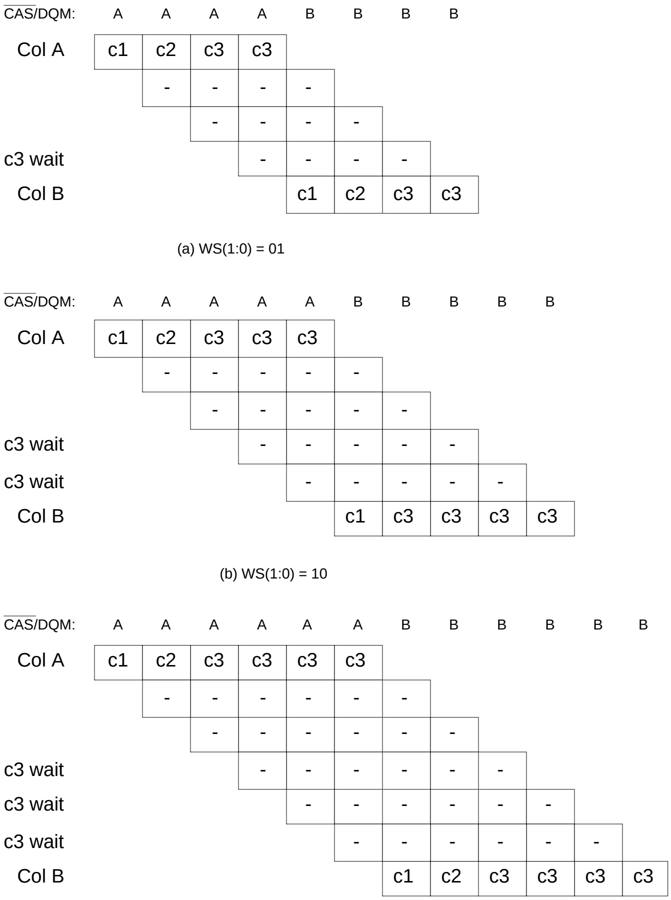
Figure 15–37.3 Cycle/Column SRAM Read (With Bubble)



15.5.1.3 3 Cycle/Column SRAM Read Wait States

Column accesses can be automatically extended by an integral number of cycles using the wait state field (**WS(1:0)**) of the configuration cache entry. Each automatic wait states causes the column pipeline to stall for one clock cycle while the current pipeline stage is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 15–38.

Figure 15–38.Effect of Wait States on 3 Cycle/Column SRAM Read Pipeline



(a) WS(1:0) = 01

(b) WS(1:0) = 10

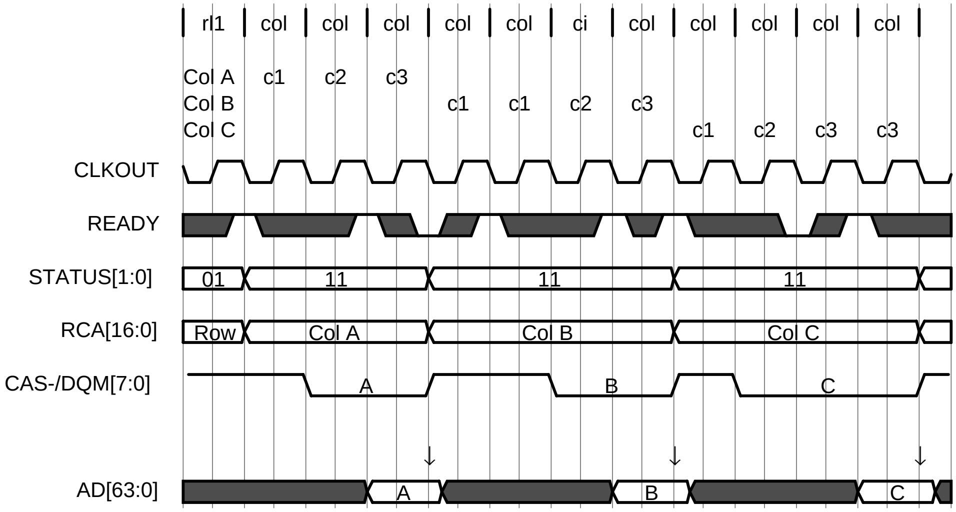
(c) WS(1:0) = 11

A **WS(1:0)** value of ‘`01`’ adds a single wait state to each access. This causes the pipeline to repeat the c3 stage as shown in Figure 15–38(a). A value of **WS(1:0)** = `10` adds two wait states to each access. As Figure 15–38(b) shows, this causes the pipeline to repeat the c3 stage twice. The c3 stage is repeated three times for **WS(1:0)** = `11`, as seen in Figure 15–38(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c3 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 15–39 demonstrates the addition wait states using the READY input. A single wait state is added to the Col B access by driving READY low at the start of the c1 stage. This causes the stage to repeat, generating a CAS-high wait state. READY is high when resampled in the repeated c1 stage and the Col B access continues. During the Col C access, READY is driven low at the start of the c3 stage. The c3 stage is repeated to create a CAS-low wait state. The READY input is high at the start of the repeated c3 stage so the Col C access completes.

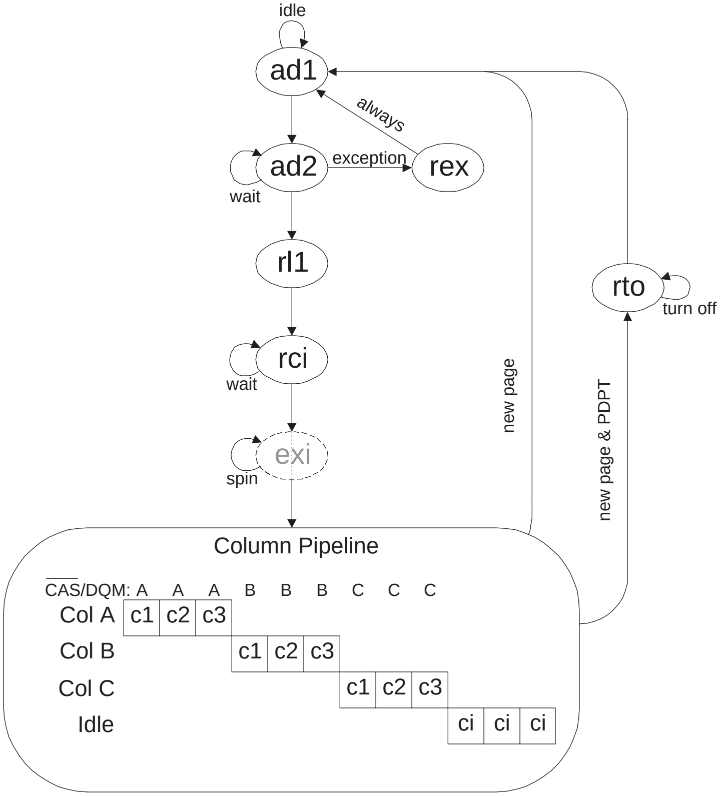
Figure 15–39.3 Cycle/Column SRAM Read with Wait State



15.5.2 3 Cycle/Column SRAM Write

The state sequence for a 3 cycle/column SRAM write is shown in Figure 15–40. The row access consists of the ad1, ad2, rl1, rcl, and possibly exi states. Row-time wait states may be inserted by deasserting the READY input in either the ad2 or rcl state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 15–40.3 Cycle/Column SRAM Write States



CAS/DQM:	A	A	A	B	B	B	C	C	C			
Col A	c1	c2	c3									
Col B				c1	c2	c3						
Col C							c1	c2	c3			
Idle										ci	ci	ci

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

c1 The column address (on RCA[16:0]) and status information (STATUS[1:0]) for the current access are output. The status code indicates that a column access is in progress. Data is driven on the appropriate bytes of the AD[63:0] bus. The CAS/DQM strobes corresponding to the valid bytes are asserted midway through the stage. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls, the CAS/DQM strobes are *not* asserted, and the stage is repeated until READY is sampled high.

c2 The address and data continue to be driven.

c3 The address and data continue to be driven. The active CAS/DQM strobes are deasserted midway through the stage to latch data into the memories. The READY input is sampled at the beginning of the cycle and if low, the pipeline stalls, the CAS/DQM strobes are not deasserted, and the stage is repeated until READY is sampled high. The c3 stage will automatically be repeated 1, 2, or 3 times for cache configuration entries of **WS(1:0)** = ‘01’, ‘10’, and ‘11’, respectively.

ci The column idle stage occurs when no access is loaded in the TC’s column pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All CAS/DQM strobes remain inactive.

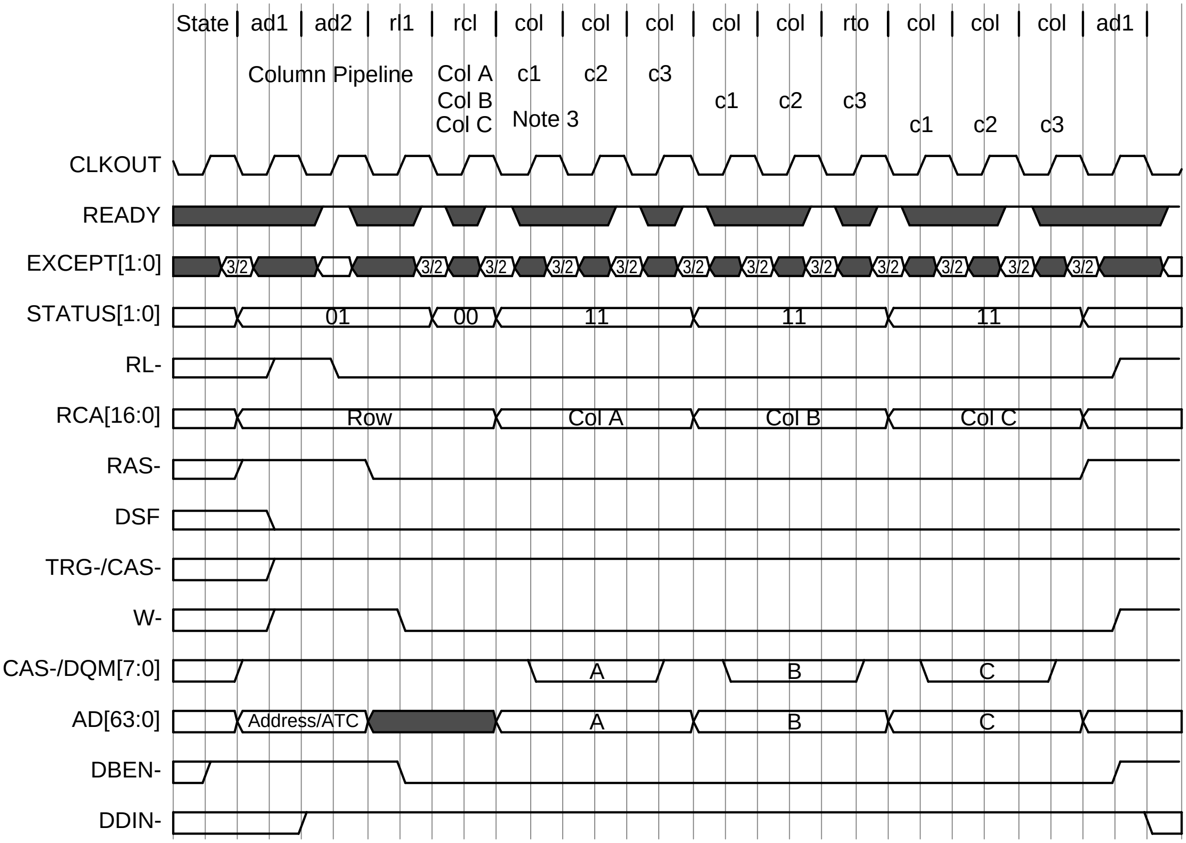
Once a column access is begun, all its pipeline stages will be completed.

After completing the last requested access, the memory interface either returns to the ad1 state if a new page access is required, or stays in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again. If the write access is the result of a peripheral device packet transfer, then one rto state will be automatically inserted prior to returning to ad1 to allow peripheral device output drivers to turn off. Additional rto states will be added if specified in the configuration cache entry.

15.5.2.1 3 Cycle/Column SRAM Write Example

An example of a 3 cycle/column SRAM write access is shown in Figure 15–41. The example shows a write of three column addresses. A new column access begins every three clock cycles. After the access to Col C is complete, the memory interface returns to the ad1 state to begin a new page access.

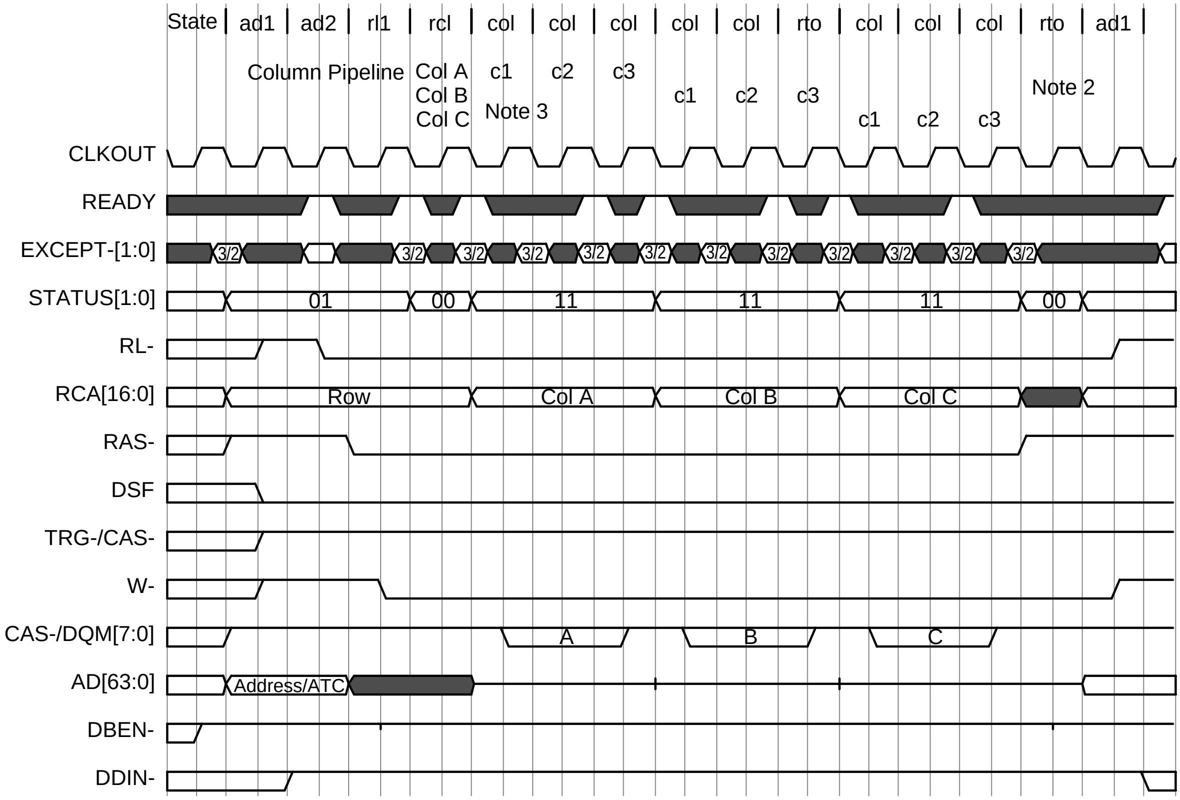
Figure 15–41.3 Cycle/Column SRAM Write Cycle Timing



Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The timing for peripheral device packet transfer, 3 cycle/column SRAM writes is modified by the addition of an automatic rto state at the end of the access to allow peripheral device output drivers to turn off. In addition, the AD[63:0] bus is placed in high-impedance to allow the peripheral to drive the bus and the DBEN output remains high to deactivate ‘C82 external bus transceivers, if present. An example is shown in Figure 15–42.

Figure 15–42.3 Cycle/Column SRAM Peripheral Device Write



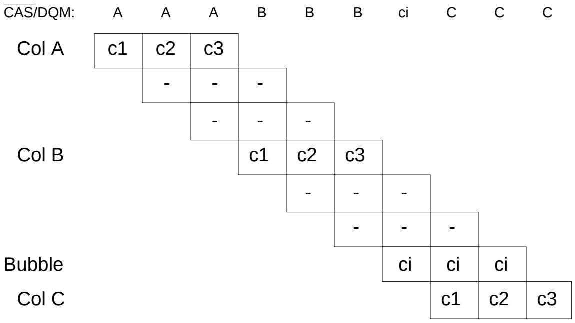
Note 2: Additional turn-off cycles will be inserted between rto and ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after rcl as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

15.5.2.2 3 Cycle/Column SRAM Write with Pipeline Bubbles

When a TC pipeline bubble occurs during writes, the loading of a new column access into the pipeline is delayed by a cycle. This effect is shown in Figure 15–43.

Figure 15–43.Effect of Bubbles on 3 Cycle/Column SRAM Write Pipeline

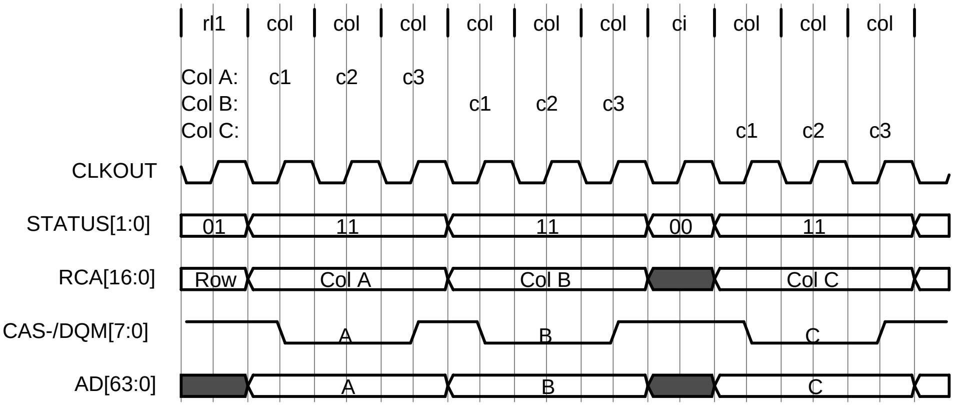


CAS/DQM:	A	A	A	B	B	B	ci	C	C	C
Col A	c1	c2	c3							
		-	-	-						
			-	-	-					
Col B				c1	c2	c3				
					-	-	-			
						-	-	-		
Bubble							ci	ci	ci	
Col C								c1	c2	c3

The column pipeline sequence for a single-cycle bubble is shown in Figure 15–43. A bubble occurs three cycles after loading the column Col B access into the pipeline preventing Col C from loading at the expected time. After the bubble, the Col C access is ready to load and the pipeline resumes normal operation. Additional bubbles would extend the ci delay by one cycle per bubble.

The column-time waveform for single cycle bubbles is shown in Figure 15–44 The bubble causes an idle status code to be output for one clock cycle indicating that no new column access is beginning. The CAS/DQM strobes remain inactive and no new data is output. If additional bubbles were to occur, the memory interface would remain in the ci stage until the next column access is ready to begin.

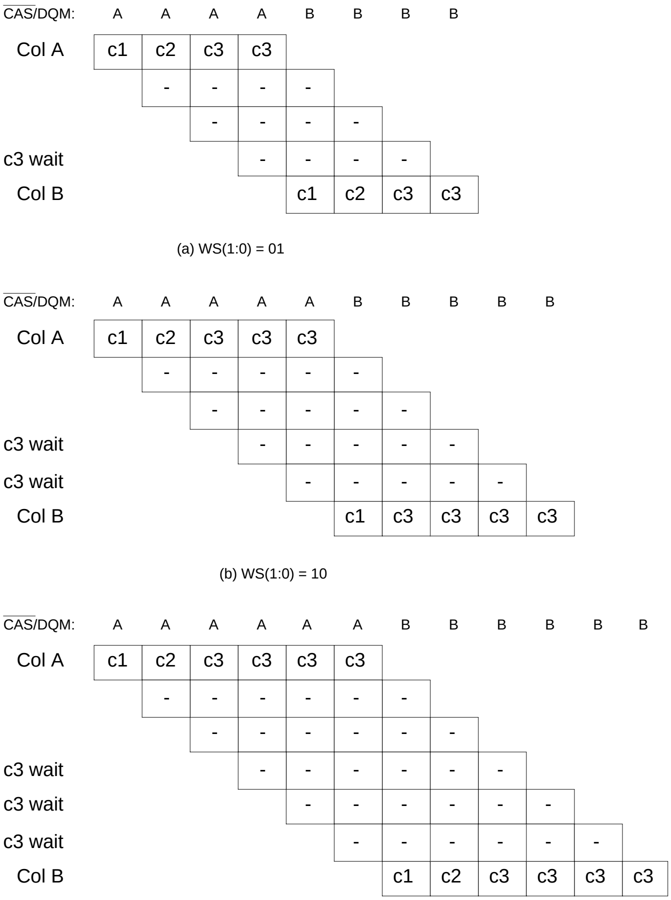
Figure 15–44.3 Cycle/Column SRAM Write (With Bubble)



15.5.2.3 3 Cycle/Column SRAM Write Wait States

Column accesses can be automatically extended by an integral number of cycles using the wait state field (**WS(1:0)**) of the configuration cache entry. Each automatic wait states causes the column pipeline to stall for one clock cycle while the current pipeline stage is repeated. The effect of **WS(1:0)** programming on the pipeline is shown in Figure 15–45.

Figure 15–45.Effect of Wait States on 3 Cycle/Column Write Pipeline

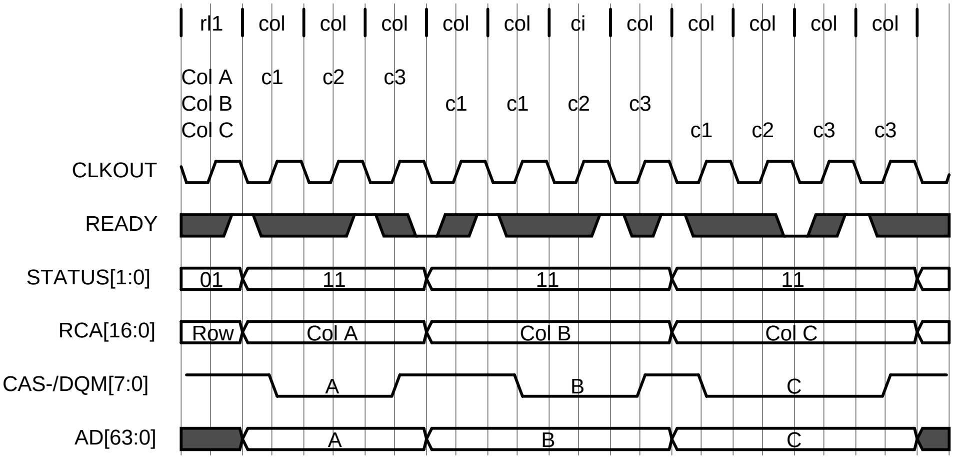


A **WS(1:0)** value of ‘01’ adds a single wait state to each access. This causes the pipeline to repeat the c3 stage as shown in Figure 15–45(a). A value of **WS(1:0)** = 10 adds two wait states to each access. As Figure 15–45(b) shows, this causes the pipeline to repeat the c3 stage twice. The c3 stage is repeated three times for **WS(1:0)** = 11, as seen in Figure 15–45(c).

Wait states may also be added by system logic through the use of the READY input. READY is sampled at the beginning of each c1 and c3 pipeline stage (including those repeated due to automatic wait states). If READY is sampled low, the stage is repeated and READY is sampled again. Wait states will continue to be inserted until READY is again sampled high at which point the pipeline continues operation.

Figure 15–46 demonstrates the addition of a wait states using the READY input. A single wait state is added to the Col B access by driving READY low at the start of the c1 stage. This adds a CAS-high wait state by causing the c1 stage to be repeated. READY is high when resampled in the repeated c1 stage and the Col B access continues. During the Col C access, the READY input is driven low at the start of the c3 stage. The cycle is repeated, adding a CAS-low wait state to the access. READY is high at the start of the repeated c3 stage so the Col C access completes.

Figure 15–46.3 Cycle/Column Write with Wait State



Chapter 16

SDRAM Cycles

The TMS320C82's memory interface includes direct support for synchronous DRAMs (SDRAMs) and synchronous graphics RAMs (SGRAMs). The high bandwidth of these devices allow the creation of large memory banks with single cycle access for enhanced system performance.

Topics in this chapter include:

Topic **Page**

16.1 SDRAM Cycle Overview

The SDRAM memory cycles allow for direct connection of SDRAM and SGRAM devices to the ‘C82’s memory interface. SDRAM timing is selected by programming the **CT(3:0)** in the configuration cache entry to one of the CT = 1xxx values shown in Table 16–1. The transfer controller supports SDRAM burst lengths of one or two columns in combination with read latencies of two, three, or four clock cycles. These combinations allow the system designer to select the timing required by the selected SDRAM device and operating frequency of the ‘C82.

Table 16–1. SRAM Cycle Timing Selection

CT(3:0)	Cycle Timing
1 0 0 0	Burst length 1; read latency 2 SDRAM
1 0 0 1	Burst length 1; read latency 3 SDRAM
1 0 1 0	Burst length 1; read latency 4 SDRAM
1 0 1 1	Reserved
1 1 0 0	Burst length 2; read latency 2 SDRAM
1 1 0 1	Burst length 2; read latency 3 SDRAM
1 1 1 0	Burst length 2; read latency 4 SDRAM
1 1 1 1	Reserved

16.1.1 SDRAM Command Support

SDRAMs use commands to control their operation and enable various features. The ‘C82 memory interface implements a subset of available SDRAM commands which are sufficient for most SDRAM and SGRAM implementations. The following SDRAM commands are supported:

- ACTV- Activate selected bank and select the row
- BLW- Block Write
- DCAB- Deactivate all banks (precharge)
- MRS- Mode register set
- READ - Initiate burst read
- REFR- Auto refresh with internal address
- SRS- Special register set
- WRT- Initiate burst write

Other SDRAM commands may be implemented using external logic but they are not directly supported or comprehended by the ‘C82.

16.1.2 SDRAM Bank Operation

SDRAMs consist of two independent array banks. Before a bank may be accessed, it must be activated with a row address. A bank must be deactivated (precharged) before it may be activated with a different row address. A bank select address input (typically A11) determines the bank on which each command will be performed.

The ‘C82 performs page-mode style accesses to SDRAM memory banks where a page corresponds to a row within an SDRAM bank. Each access begins with an activate (ACTV) command to select the desired SDRAM row. The transfer controller then performs either burst read or burst write commands on the current row until a page change occurs. Before completing the access, a deactivate (DCAB) command is performed to precharge the bank in preparation for the next access.

The ‘C82 does not support multiple active bank operation. The active bank is always deactivated when a SDRAM page change occurs.

16.1.3 SDRAM Read Latency

The read latency determines the number of clock cycles between a read command and the requested data becoming available. Most SDRAMs allow read latencies of one, two, or three clock cycles depending on the device speed and operating frequency. The ‘C82 supports latencies of two, three, or four clock cycles. The four cycle latency is intended to be used in systems with large amounts of SDRAMs or in systems where the SDRAM is not “local” to the ‘C82. This allows the insertion of a clocked data buffer between the SDRAM and ‘C82. By operating the SDRAM at read latency three, an extra clock cycle is available to clock the data through the buffer before it is sampled by the ‘C82.

16.1.4 SDRAM Burst Length

SDRAMs support various burst lengths which determine the number of column locations that will be accessed by read or write commands. The ‘C82 memory interface supports burst lengths of 1 or 2. Because the transfer controller can generate a new column address (and read or write command) on every clock cycle, a burst length of 1 is optimal. This allows the TC to access nonsequential columns within a row as may be required by packet transfers.

Some early SDRAMs used an “interleaved” architecture which allowed a new read or write command to be issued only on every other clock cycle. For SDRAMs that implement this “2n” rule, a burst length of 2 must be selected.

The TC always uses the SDRAM’s “serial” burst sequence. If a burst length of 2 is used and the second column in the burst is not required, it will be disabled by inactivating the appropriate CAS/DQM signals.

16.1.5 SDRAM Pin Connections

The ‘C82 memory interface pins have names corresponding to both DRAM and SDRAM signals. Table 16–2 illustrates the connections between the TMS320C82 and SDRAMs for most systems.

Table 16–2. SDRAM Pin Connections

SDRAM Pin	TMS320C82 Signal
RAS	RAS
CAS	$\overline{\text{TRG}}$/$\overline{\text{CAS}}$
DQM	$\overline{\text{CAS}}$/DQMx
W	W
CLK	CLKOUT
DQxx	ADxx
A[9:0]	RCA[9:0]
A10	-
A11	-
CS	-
CKE	-

As the table shows, the ‘C82 memory interface provides direct connections for all the SDRAM control signals. The DQM inputs are connected to the ‘C82 CAS/DQM[7:0] output appropriate for the data byte. The SDRAM data pins (DQxx) are connected to the appropriate bytes on the ‘C82 AD[63:0] bus. The SDRAM chip select ($\overline{\text{CS}}$) should be driven by external bank decode logic (typically based on latched row address bits) or can be tied low if one bank of SDRAM is the only external memory present. The clock enable signal (CKE) is used to enable special power down and refresh modes not supported by the ‘C82. This signal may be pulled high in most systems not requiring these modes.

The SDRAM A10 and A11 inputs require special logic because they behave more like control signals than address signals. The A11 input represents the SDRAM bank select signal. SInce the ‘C82 does not support multi-bank operation, this address bit must maintain its value for activate and read/write commands. This is easily achieved by latching the appropriate address bit on the AD[63:0] bus using the fall of RL. The A10 input acts as a row address bit during ACTV commands but must be low during READ and WRT commands. This is typically implemented by driving A10 low when RAS is high and allowing it to be driven by the appropriate address signal (typically the address bit below the one used for A11) when RAS is low.

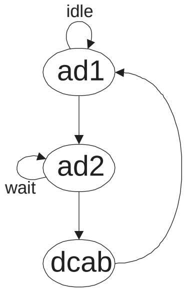
16.2 SDRAM Initialization

In order for the ‘C82 to properly support SDRAM, its presence in the system must be signalled using external logic. This is done by generating the SDRAM refresh code (on EXCEPT[1:0]) during the power-up initialization (refresh) sequence. The first refresh that receives the SDRAM refresh code will be abandoned and a power-up DCAB command will be performed to deactivate all SDRAM banks.

16.2.1 SDRAM DCAB State Sequence

Figure 16–1 shows the state sequence for the power-up DCAB command. The cycle consists of the ad1, ad2, and dcab states. The ad2 may be repeated by deasserting the READY input. After completing the dcab state the memory interface will immediately return to the ad1 state.

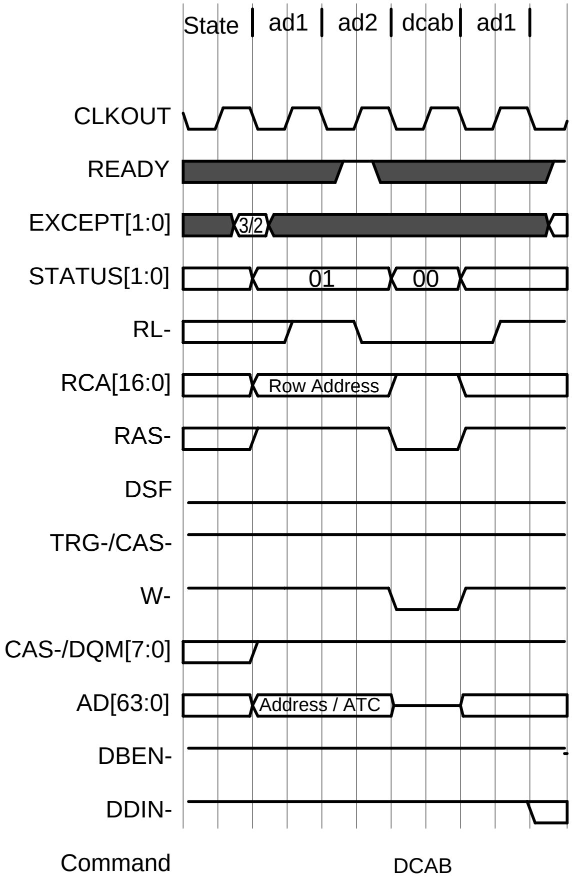
Figure 16–1. SDRAM Power-Up DCAB State Sequence



16.2.2 SDRAM DCAB Example

An example of an SDRAM DCAB cycle is shown in Figure 16–2. The DCAB command is generated by driving both the RAS and W control signals low. The RCA[16:0] bus is driven high to ensure that all banks within the SDRAMs are precharged. The actual row address output on RCA[16:0] and AD[31:0] is unspecified. System designers should decode the access type code (ATC) on AD[39:32]. When the power-up DCAB code (AD[39:32] = `0x0D`) is detected, the chip selects on all SDRAM devices should be enabled to ensure that the DCAB is applied to all SDRAMs.

Figure 16–2. SDRAM DCAB



16.3 SDRAM Mode Register Programming

Once power-up initialization has been performed, each SDRAM must be configured for the desired operation by programming its mode register. The mode register determines operating parameters such as burst length and read latency. The ‘C82 configures system SDRAMs using the mode register set (MRS) cycle.

The MRS cycle will be the first access performed to an SDRAM memory bank after completion of the bank configuration cycle. This ensures that each SDRAM bank is properly configured to work with the cycle timing selected by the configuration cache entry. This is the only time that MRS cycles will be performed. If the SDRAM mode register value is changed by the host or another memory controller, that device must return the mode register to its expected value before the next ‘C82 access. Alternatively, a cache flush exception can be generated on the next ‘C82 access to force a bank configuration cycle and a new MRS command.

16.3.1 Mode Register Value

The mode register is programmed via the SDRAM address inputs during an MRS cycle. The address inputs are defined as shown in Figure 16–3. The programmable fields are write burst length (WB), read latency, serial/interleave access (S/I) and burst length.

Figure 16–3. MRS Value Generation

SDRAM Address	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Definition	Reserved		WB	0	0	Read Latency			S/I	Burst Length		
'C82 Output Value	0	0	0	0	0	0	1		0	0	0	CT2

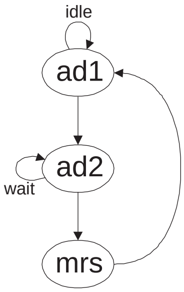
CT1 # CT0

The ‘C82 determines the mode register value based upon the cycle timing field (CT(3:0)) of the cache configuration entry. The burst length is programmed to either ‘`000`’ (burst length 1) or ‘`001`’ (burst length 2) based on the value of CT2. The S/I bit is always programmed to ‘`0`’ for serial access. Read latency is programmed to ‘`010`’ for latency 2 configuration or ‘`011`’ for read latency 3 or 4. The WB bit is always set to ‘`0`’ to enable write bursts of the burst length. The mode register value is always output on the RCA[11:0] address outputs of the ‘C82.

16.3.2 SDRAM MRS State Sequence

Figure 16–4 shows the state sequence for the MRS command. The cycle consists of the ad1, ad2, and mrs states. The ad2 may be repeated by deasserting the READY input. After completing the cmd state the memory interface will immediately return to the ad1 state.

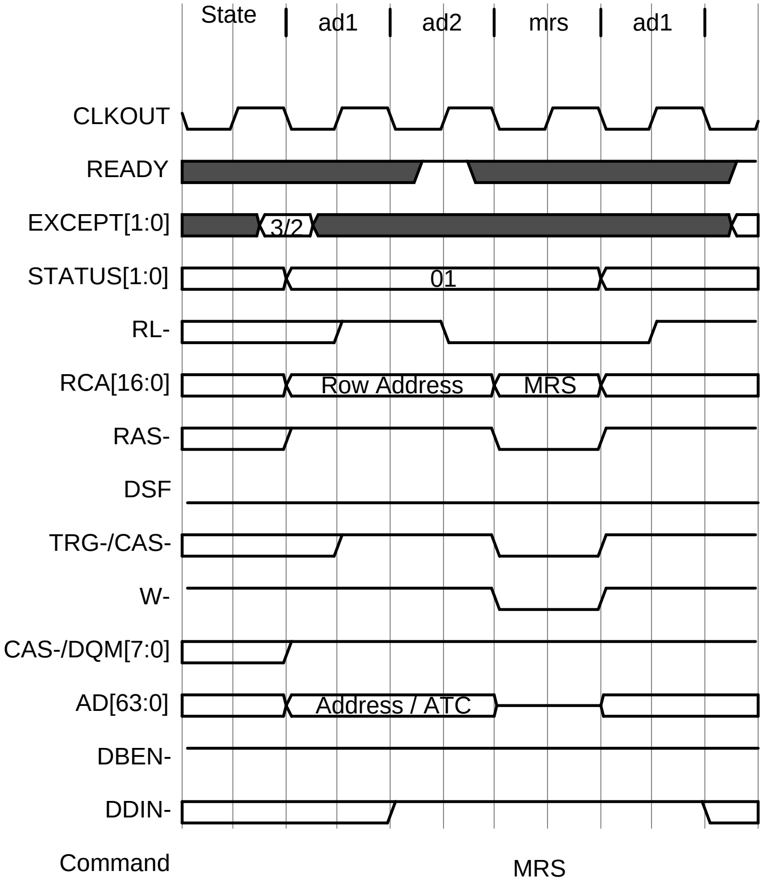
Figure 16–4. SDRAM MRS State Sequence



16.3.3 SDRAM MRS Example

An example of an SDRAM MRS cycle is shown in Figure 16–5. The MRS command is generated by driving the RAS, TRG/CAS, and W control signals low. The RCA[16:0] bus is driven with the value to be placed in the mode register.

Figure 16–5. SDRAM MRS



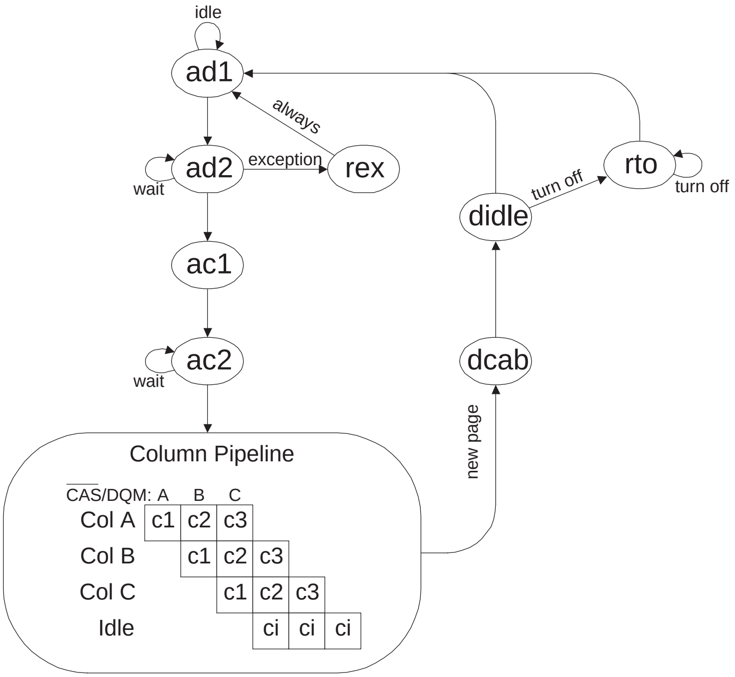
16.4 SDRAM Burst Length 1 Cycles

The burst length 1 SDRAM cycles are designed for use with SDRAMs that support a new read or write command on every clock cycle. This provides maximum bandwidth to the transfer controller when servicing packet transfers which may (depending on PT parameters) access individual columns rather than blocks of sequential locations.

16.4.1 SDRAM Burst Length 1, Latency 2 Read

Read latency 2 can be used for SDRAMs that can return data within two ‘C82 clock cycles of the read command. The state sequence for this access is shown in Figure 16–6. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–6. SDRAM Burst Length 1, Latency 2 Read States



$\overline{\text{CAS}}$/DQM:	A	B	C			
Col A	c1	c2	c3			
Col B		c1	c2	c3		
Col C			c1	c2	c3	
Idle				ci	ci	ci

Column accesses consist of a three stage pipeline. A read command can begin on every cycle but each command takes three clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0] and a read command is issued by driving TRG/CAS low. The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The column status code is driven on STATUS[1:0].

c2 The ‘C82 waits for data. The command signals driven in c1 are deactivated.

c3 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

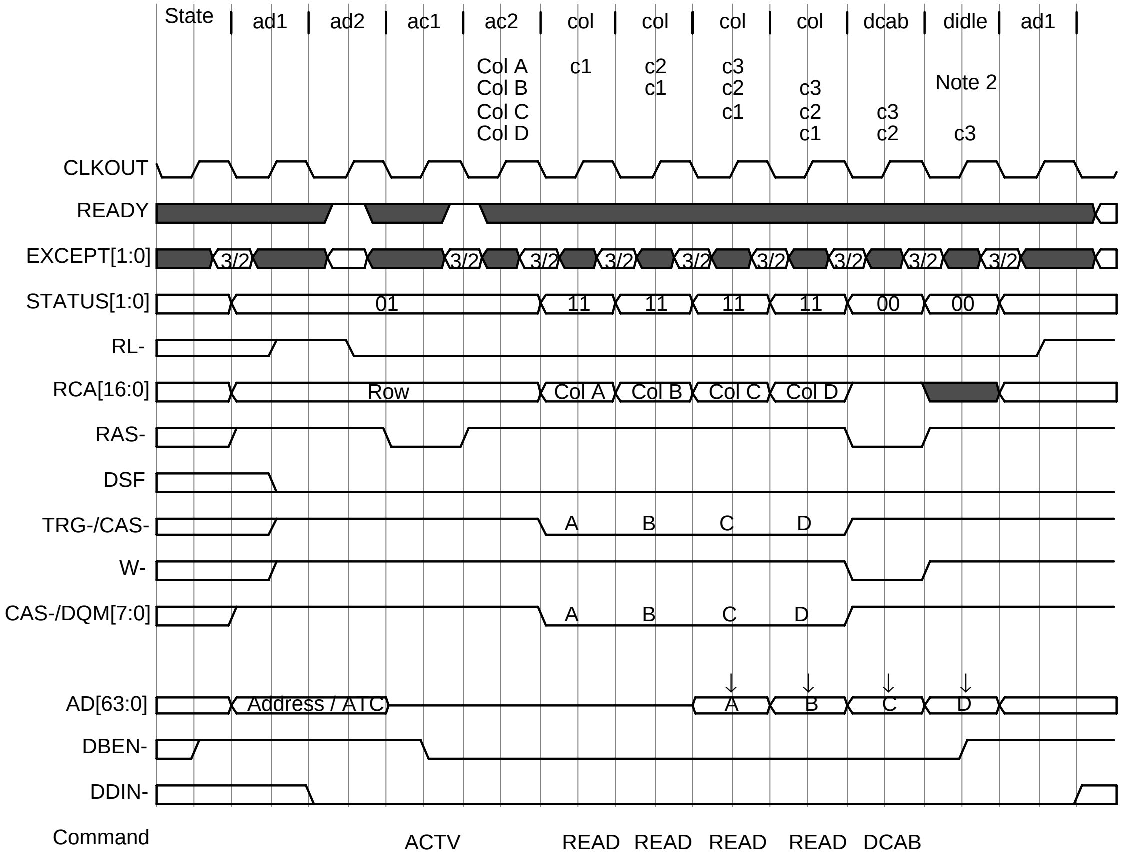
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c2 stage of the final column access. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c3 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.4.1.1 SDRAM, Burst Length 1, Latency 2 Read Example

An example of a burst length 1, latency 2 SDRAM read access is shown in Figure 16–7. The example shows a read of four column addresses. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–7. Burst Length 1, Latency 2 SDRAM Read



Note 2: Additional turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.4.1.2 Burst Length 1, Latency 2 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 1 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 16–8.

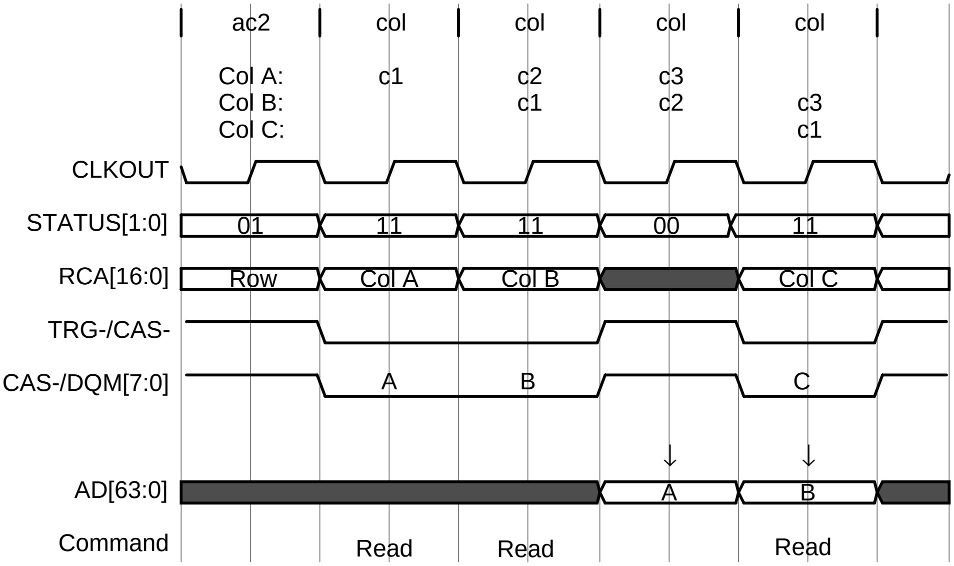
Figure 16–8. Effect of Bubbles on Burst Length 1, Latency 2 SDRAM Read Column Pipeline

CAS/DQM:	A	B	ci	C	none	none
Col A	c1	c2	c3			
Col B		c1	c2	c3		
Bubble			ci	ci	ci	
Col C				c1	c2	c3

After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble is shown in Figure 16–9. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble, however the data from the Col A access is still latched normally.

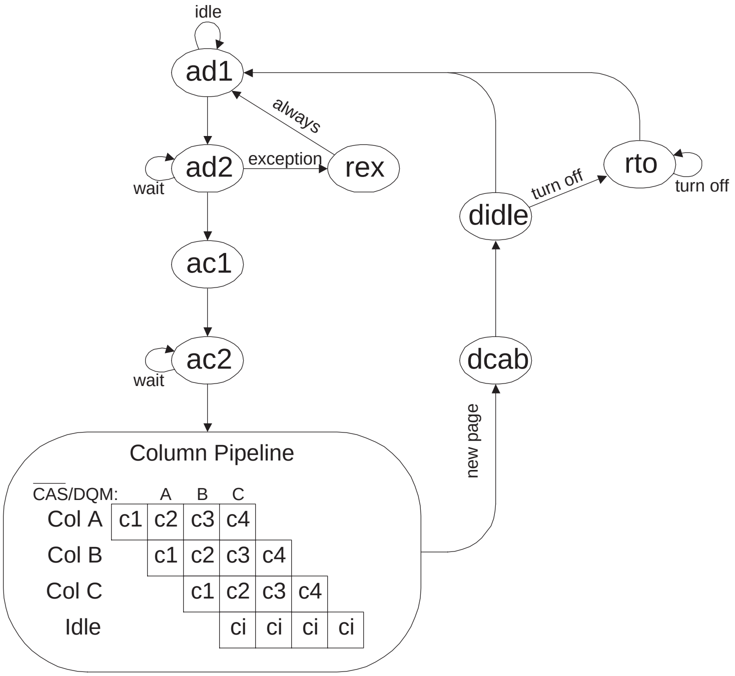
Figure 16–9. Burst Length 1, Latency 2 SDRAM Read (Single Cycle Bubble)



16.4.2 SDRAM Burst Length 1, Latency 3 Read

Read latency 3 can be used for SDRAMs that can return data within three ‘C82 clock cycles of the read command. The state sequence for this access is shown in Figure 16–10. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–10.SDRAM Burst Length 1, Latency 3 Read States



CAS/DQM:		A	B	C			
Col A	c1	c2	c3	c4			
Col B		c1	c2	c3	c4		
Col C			c1	c2	c3	c4	
Idle				ci	ci	ci	ci

Column accesses consist of a four stage pipeline. A read command can begin on every cycle but each command takes three clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0] and a read command is issued by driving TRG/CAS low. The column status code is driven on STATUS[1:0].

c2 The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The command signals driven in c1 are deactivated.

c3 The CAS/DQM outputs are deactivated. The ‘C82 waits for data.

c4 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

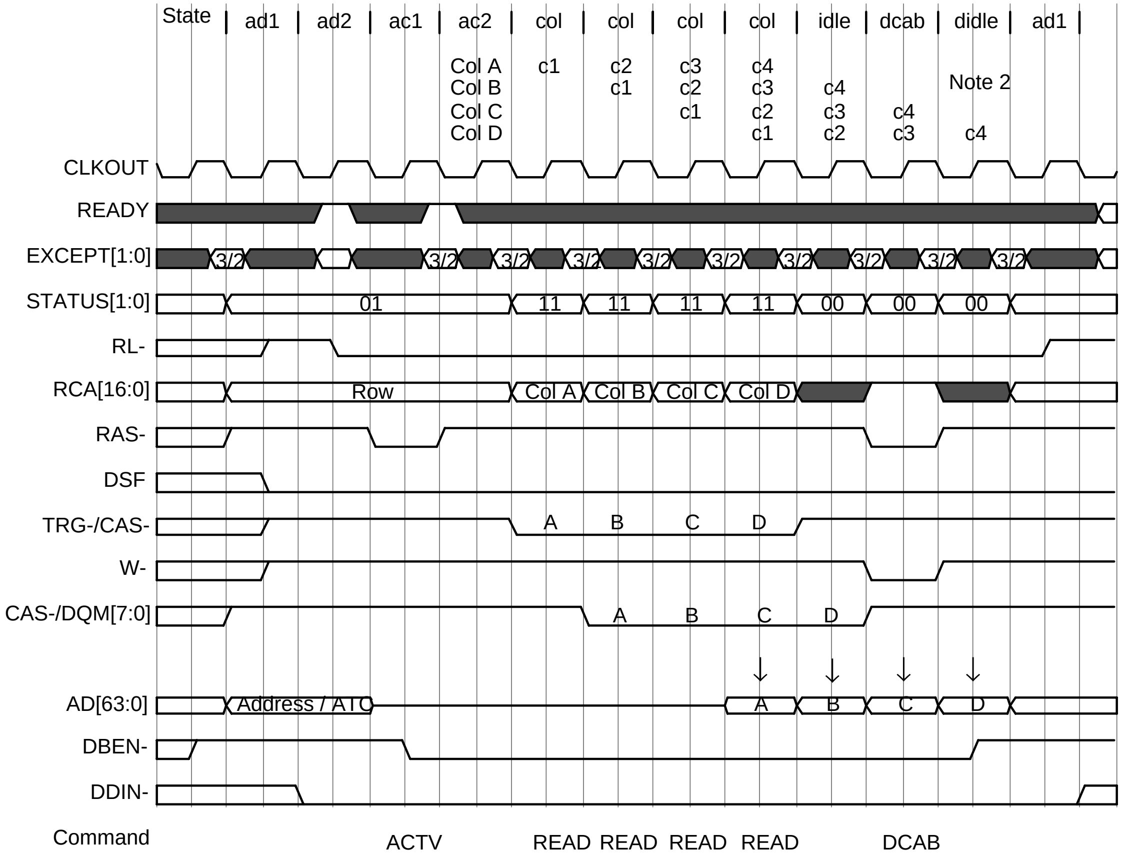
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c3 stage of the final column access. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c4 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.4.2.1 SDRAM, Burst Length 1, Latency 3 Read Example

An example of a burst length 1, latency 3 SDRAM read access is shown in Figure 16–11. The example shows a read of four column addresses. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–11.Burst Length 1, Latency 3 SDRAM Read



Note 2: Additional turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.4.2.2 Burst Length 1, Latency 3 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 1 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 16–12.

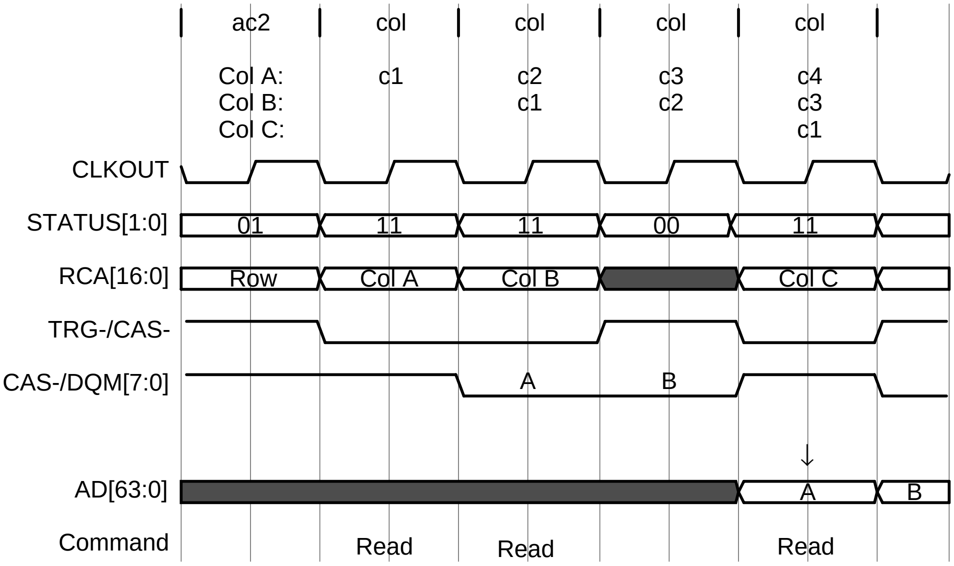
Figure 16–12.Effect of Bubbles on Burst Length 1, Latency 3 SDRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	B	ci	C	none	none
Col A	c1	c2	c3	c4			
Col B		c1	c2	c3	c4		
Bubble			ci	ci	ci	ci	
Col C				c1	c2	c3	c4

After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble is shown in Figure 16–13. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

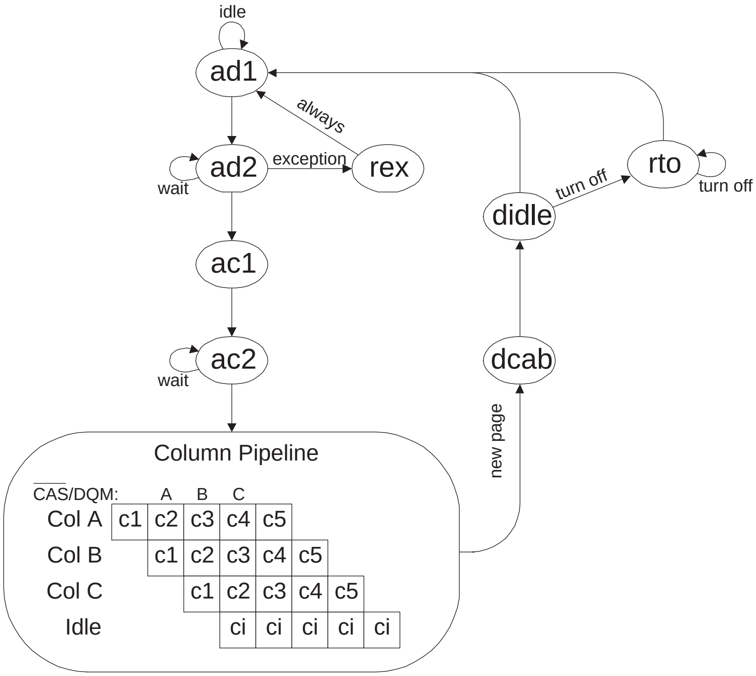
Figure 16–13.Burst Length 1, Latency 3 SDRAM Read (Single Cycle Bubble)



16.4.3 SDRAM Burst Length 1, Latency 4 Read

Read latency 4 can be used in systems requiring an extra cycle of access time. The SDRAMs are programmed for latency 3 reads but the ‘C82 delays the latching of data by one cycle. This allows the use of clocked data buffers between the ‘C82 and SDRAM. The state sequence for this access is shown in Figure 16–14. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–14.SDRAM Burst Length 1, Latency 4 Read States



CAS/DQM:		A	B	C				
Col A	c1	c2	c3	c4	c5			
Col B		c1	c2	c3	c4	c5		
Col C			c1	c2	c3	c4	c5	
Idle				ci	ci	ci	ci	ci

Column accesses consist of a five stage pipeline. A read command can begin on every cycle but each command takes three clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0] and a read command is issued by driving TRG/CAS low. The column status code is driven on STATUS[1:0].

c2 The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The command signals driven in c1 are deactivated.

c3 The CAS/DQM outputs are deactivated.

c4 The ‘C82 waits for data. The SDRAMs will typically output valid data at this time where it can be latched into an external buffer.

c5 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

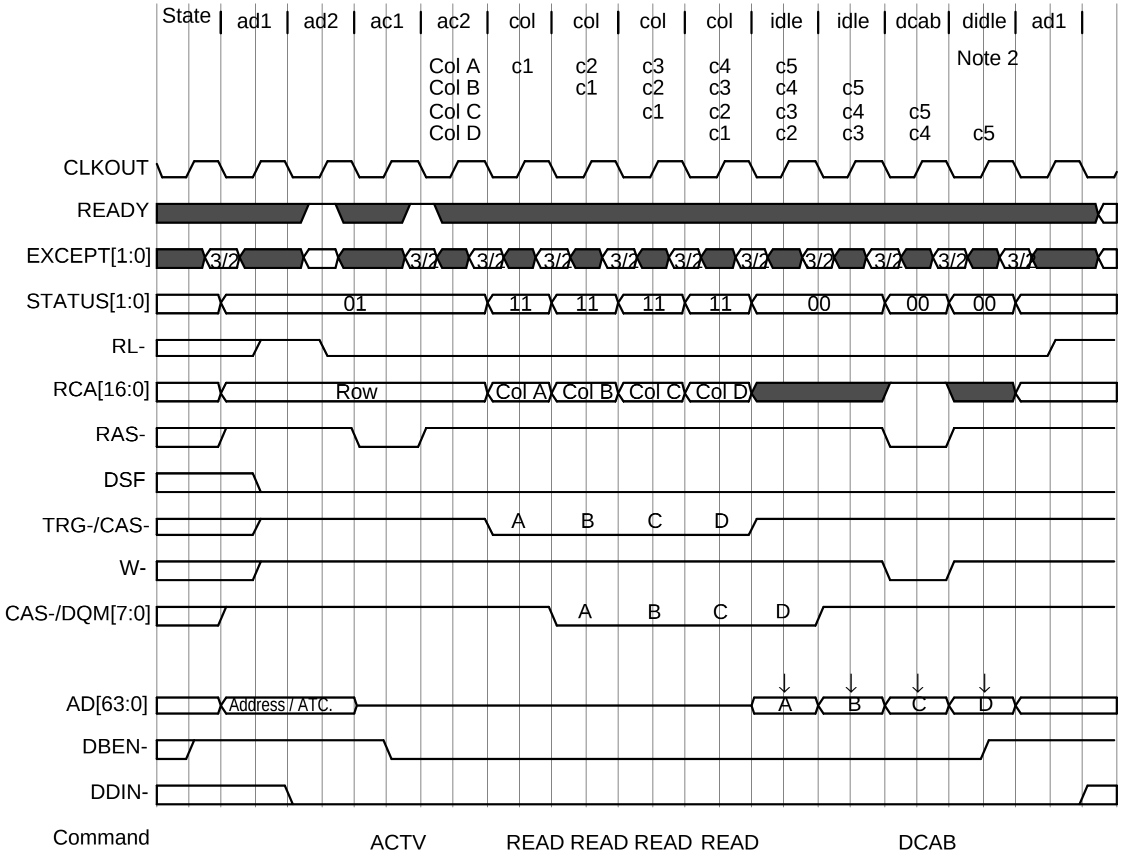
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c4 stage of the final column access. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c5 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.4.3.1 SDRAM, Burst Length 1, Latency 4 Read Example

An example of a burst length 1, latency 4 SDRAM read access is shown in Figure 16–15. The example shows a read of four column addresses. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–15.Burst Length 1, Latency 4 SDRAM Read



Note 2: Additional turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.4.3.2 Burst Length 1, Latency 4 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 1 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, c3, c4, and c5 stages. This effect is shown in Figure 16–16.

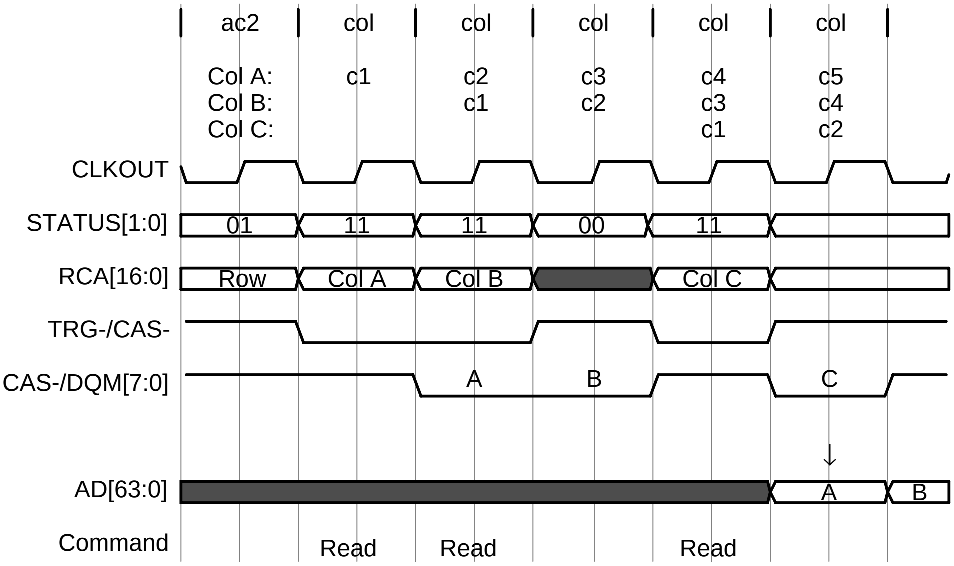
Figure 16–16.Effect of Bubbles on Burst Length 1, Latency 4 SDRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	B	ci	C	none	none	none
Col A	c1	c2	c3	c4	c5			
Col B		c1	c2	c3	c4	c5		
Bubble			ci	ci	ci	ci	ci	
Col C				c1	c2	c3	c4	c5

After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble is shown in Figure 16–17. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

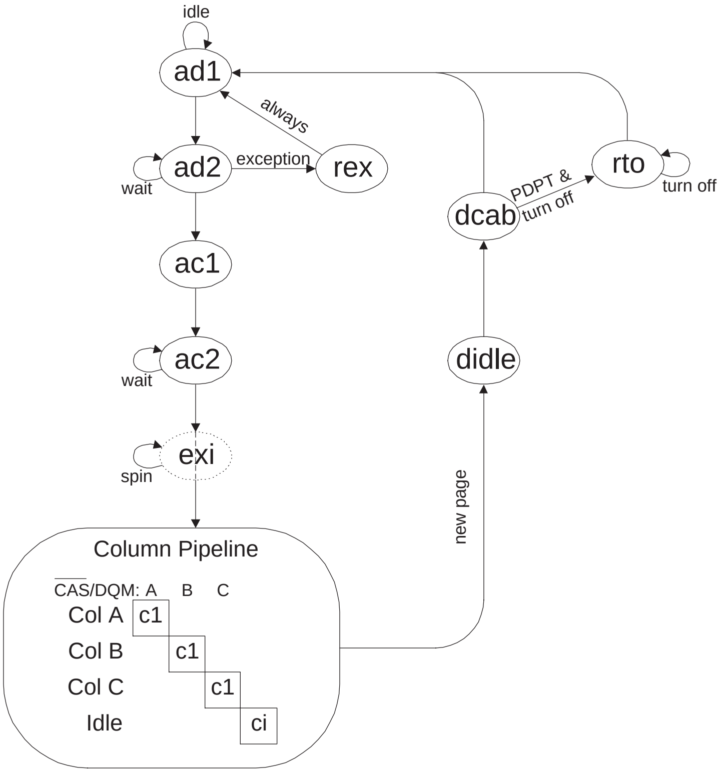
Figure 16–17.Burst Length 1, Latency 4 SDRAM Read (Single Cycle Bubble)



16.4.4 SDRAM Burst Length 1 Write

Burst length 1 write accesses are performed on all SDRAMs configured for burst length 1. The state sequence for this access is shown in Figure 16–18. The row access consists of the ad1, ad2, ac1, ac2, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi cycles such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 16–18.SDRAM Burst Length 1, Write States



Column accesses consist of a single stage pipeline. A write command can begin on every cycle and each command takes one clock cycle to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0] and a write command is issued by driving TRG/CAS and W low. The column status code is driven on STATUS[1:0]. The write data is driven on AD[63:0] and the appropriate CAS/DQM[7:0] signals are driven low to indicate the valid bytes.

ci The column idle stage occurs when no access is loaded in the pipeline. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state.

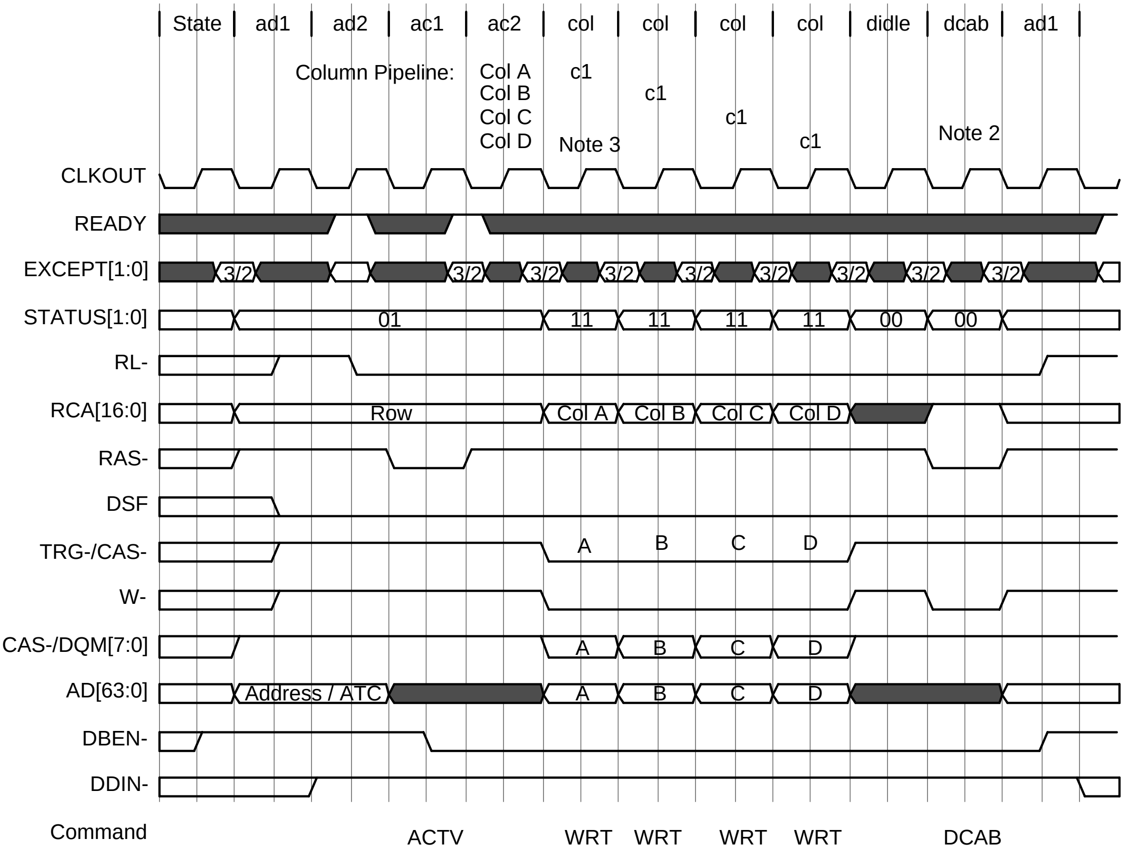
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may occur no sooner than 2 clock cycles after the last write so a DIDLE state occurs prior to the DCAB to ensure that SDRAM timing is met. If the write access is a result of a peripheral device packet transfer, then one or more rto states will be inserted as specified in the configuration cache entry prior to returning to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.4.4.1 SDRAM, Burst Length 1, Write Example

An example of a burst length 1, SDRAM write access is shown in Figure 16–19. The example shows a write of four column addresses. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–19.Burst Length 1 SDRAM Write



Note 2: During peripheral data transfers, turn-off cycles will be inserted prior to ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after ac2 a required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

16.4.4.2 Burst Length 1, Writes with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 1 writes, the loading of a new column access into the pipeline is delayed. This effect is shown in Figure 16–20.

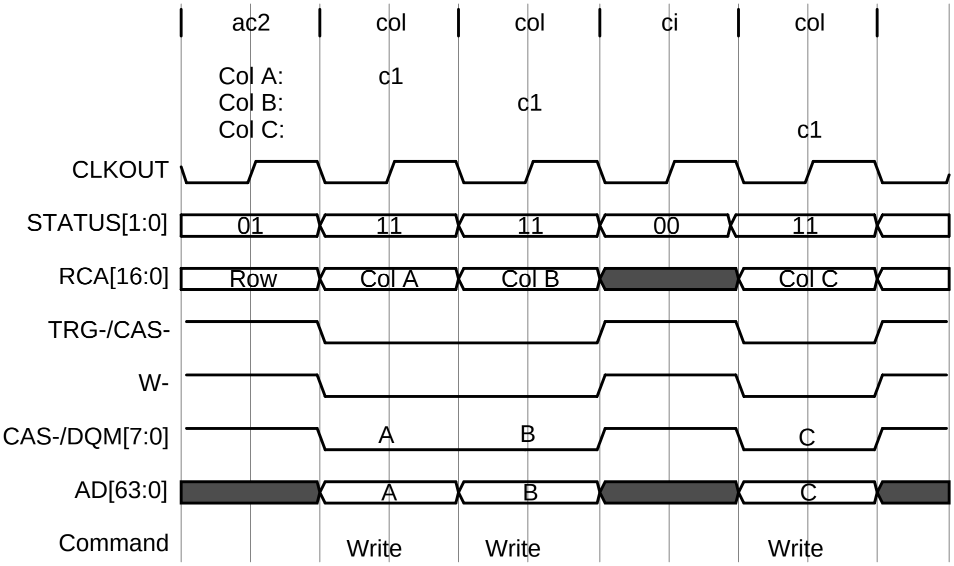
Figure 16–20.Effect of Bubbles on Burst Length 1, SDRAM Write Column Pipeline

$\overline{\text{CAS}}$/DQM:	A	B	ci	C
Col A	c1			
Col B		c1		
Bubble			ci	
Col C				c1

After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble is shown in Figure 16–21. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

Figure 16–21.Burst Length 1, SDRAM Write (Single Cycle Bubble)



16.5 SDRAM Burst Length 2 Cycles

The burst length 2 SDRAM cycles are designed for use with SDRAMs that follow the “2n” rule allowing read or write commands to be issued only 2 clock cycles after the previous command. By using a burst length of two, data can still be read or written on every clock cycle provided two adjacent column locations are being accessed.

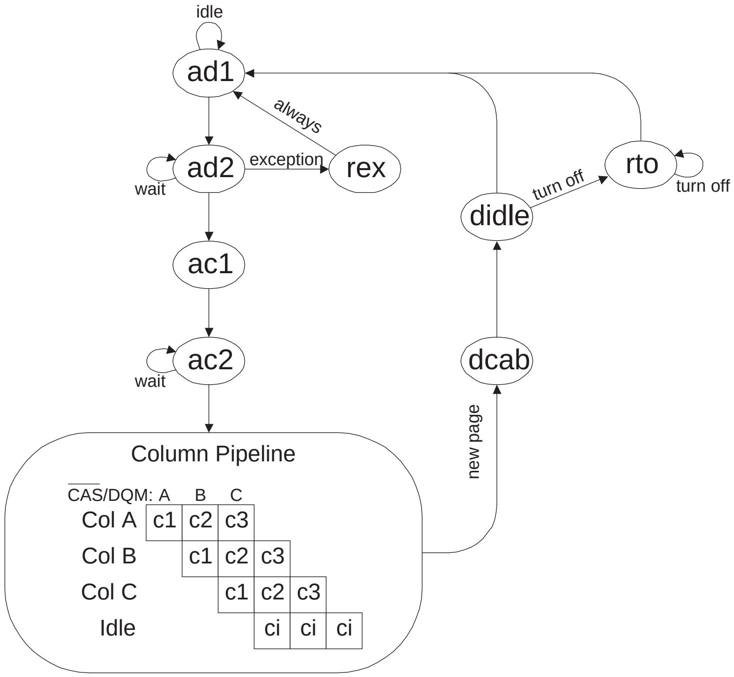
A burst length of two causes the SDRAM to access two adjacent column locations - 0 and 1. The access order is based on the least significant column address bit at the time the command is issued. Thus a read of “column 00” would cause column 00 data to be output (after the appropriate latency) on the first output cycle followed by column 01 data on the next clock cycle. A read of “column 01” would cause column 01 data to be output first followed by column 00 data on the next clock cycle. Note that the second value is the column 00 data *not* the column 10 data.

The transfer controller makes use of the second access in the burst whenever possible. If a command is issued to one column and the next access is to the adjacent column in the burst, then both accesses will be performed. If the next required access is not the second column in the burst sequence then the TC will disable the second access in the burst (using the CAS/DQM outputs) and issue a new command on the next clock cycle.

16.5.1 SDRAM Burst Length 2, Latency 2 Read

Read latency 2 can be used for SDRAMs that can return data within two ‘C82 clock cycles of the read command. The state sequence for this access is shown in Figure 16–22. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–22.SDRAM Burst Length 2, Latency 2 Read States



Column accesses consist of a three stage pipeline. A read access can be loaded into the pipeline on every cycle but each command takes three clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0]. If the access is the first in a burst, a read command is issued by driving TRG/CAS low. The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The column status code is driven on STATUS[1:0].

c2 The ‘C82 waits for data. The command signals driven in c1 are deactivated.

c3 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. This could be due to a pipeline bubble or because the second column location in the burst is not needed. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

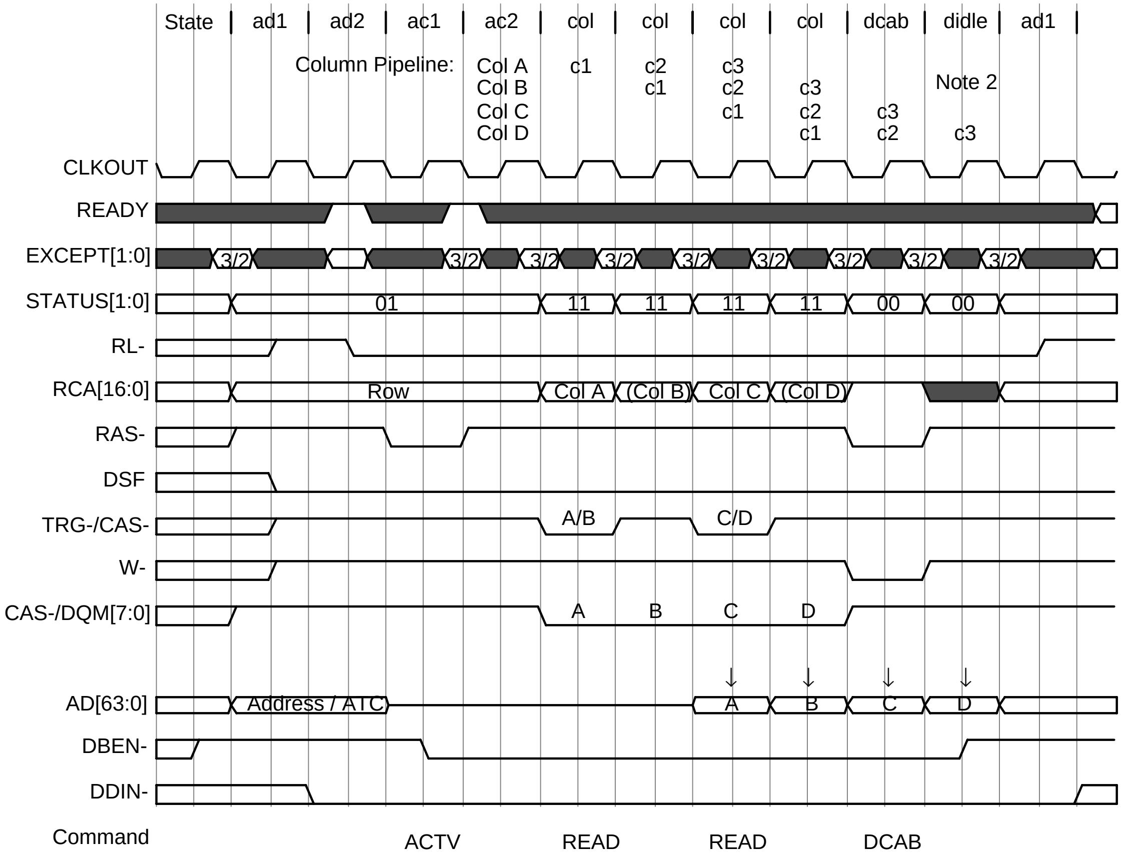
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c2 stage of the second column access in the final burst. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c3 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.5.1.1 SDRAM, Burst Length 2, Latency 2 Read Example

An example of a burst length 2, latency 2 SDRAM read access is shown in Figure 16–23. The example shows a read of four column addresses. In the example, Col A and Col B are contiguous locations as are Col C and Col D. The first read command causes the Col A/Col B burst to be output and the second read command causes the Col C/Col D burst to be output. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–23.Burst Length 2, Latency 2 SDRAM Read



Note 2: Turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.5.1.2 Burst Length 2, Latency 2 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 2 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, and c3 stages. This effect is shown in Figure 16–24.

Figure 16–24.Effect of Bubbles on Burst Length 2, Latency 2 SDRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	A	B	ci	C	D	none	none
Col A	c1	c2	c3				
(Col B)		c1	c2	c3			
Bubble			ci	ci	ci		
Col C				c1	c2	c3	
(Col D)					c1	c2	c3

(a) Bubble Occurring on Burst Boundary

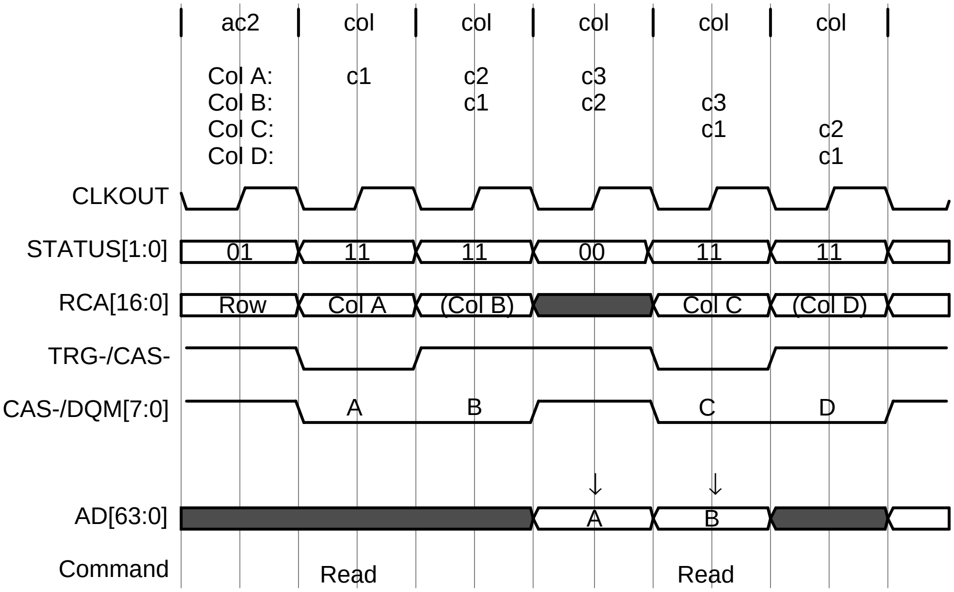
$\overline{\text{CAS}}$/DQM:	A	ci	B	ci	C	none	none
Col A	c1	c2	c3				
Bubble		ci	ci	ci			
Col B			c1	c2	c3		
Disc				ci	ci	ci	
Col C					c1	c2	c3

(b) Bubble Occurring During Burst

Figure 16–24(a) shows the column pipeline sequence of a bubble occurring between burst accesses. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble occurring between bursts is shown in Figure 16–25. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble, however the data from the Col A access is still latched normally.

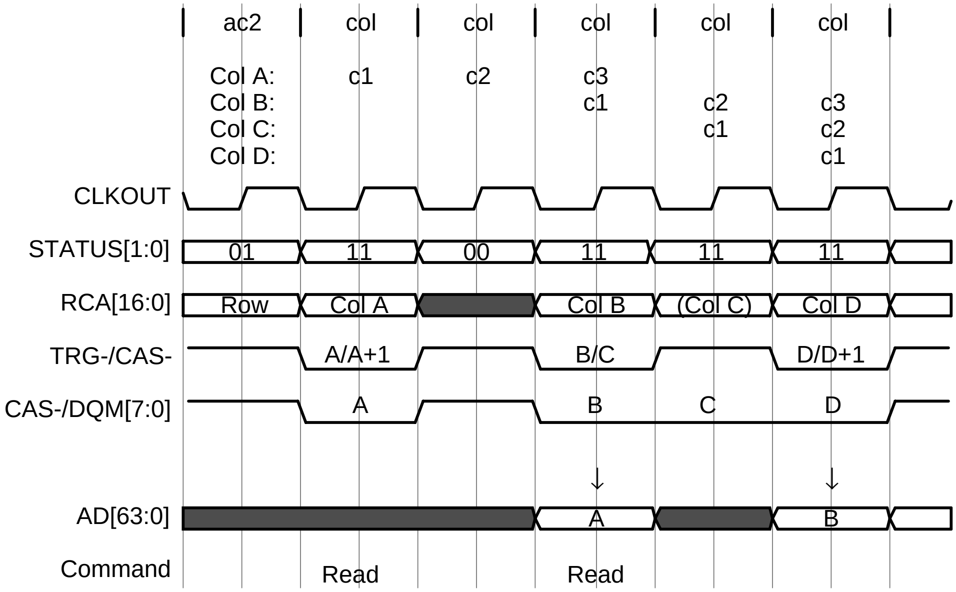
Figure 16–25.Burst Length 2, Latency 2 SDRAM Read (Single Bubble Between Bursts)



When the TC does not need the second access in a burst, a burst discontinuity occurs. Since commands cannot be issued on consecutive clock cycles, the TC must wait one clock before it can load the column pipeline for the next (noncontigous) column access. On the external memory interface, this operation appears much like a pipeline bubble.

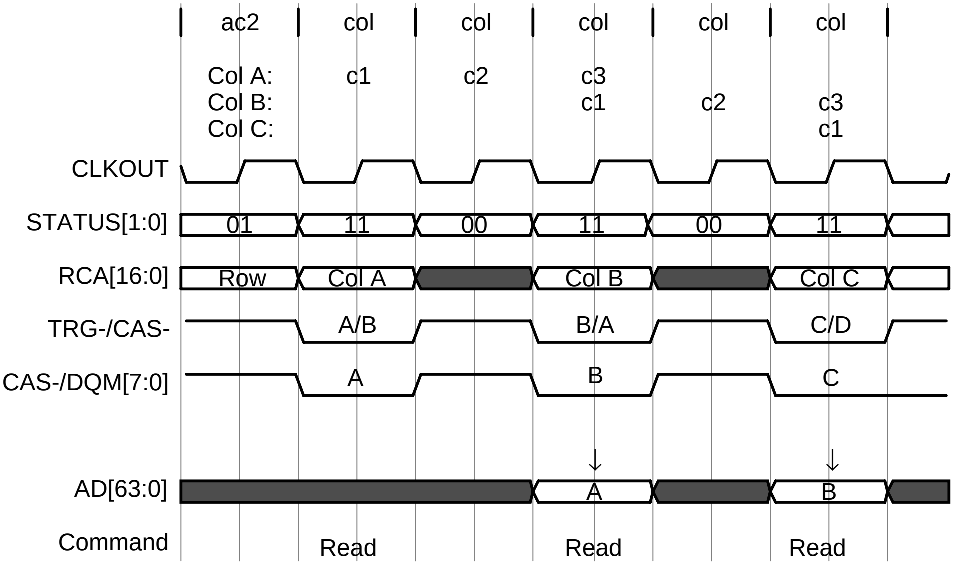
Figure 16–26 shows an example of a burst discontinuity. In the example, Col A and Col B are not within the same 2 column burst. The first read command accesses Col A and the next column in the burst (A+1). Since the TC needs to read Col B next rather than Col A+1, it must wait one cycle before loading its pipeline with the Col B access. The TC disables the CAS/DQM outputs during the c2 state of Col A to indicate that it won't be using the A+1 data. After waiting a clock cycle (thus satisfying the "2n" rule) the TC then loads the pipeline with the Col B access. A new read command is generated for the two burst sequence of Col B and Col C and the pipeline continues normal operation.

Figure 16–26.Burst Length 2 Latency 2 SDRAM Read with Burst Discontinuity



When a pipeline bubble occurs within a burst, both a bubble and discontinuity will occur as shown in Figure 16–24(b). In this example, Col A and Col B are a “col 0/ col 1” burst pair. The TC loads Col A into its pipeline, generating a read command of the ColA/ColB burst. However, a pipeline bubble prevents the TC from loading Col B on the next clock. This prevents the TC from using the Col B data output during the second cycle of the burst. On the next clock cycle the Col B access is loaded. This generates a read command of the Col B/Col A burst. Since the Col A data has already been read, the second access of the burst is not needed so a burst discontinuity occurs. Figure 16–27 shows the bubble and discontinuity as they occur externally.

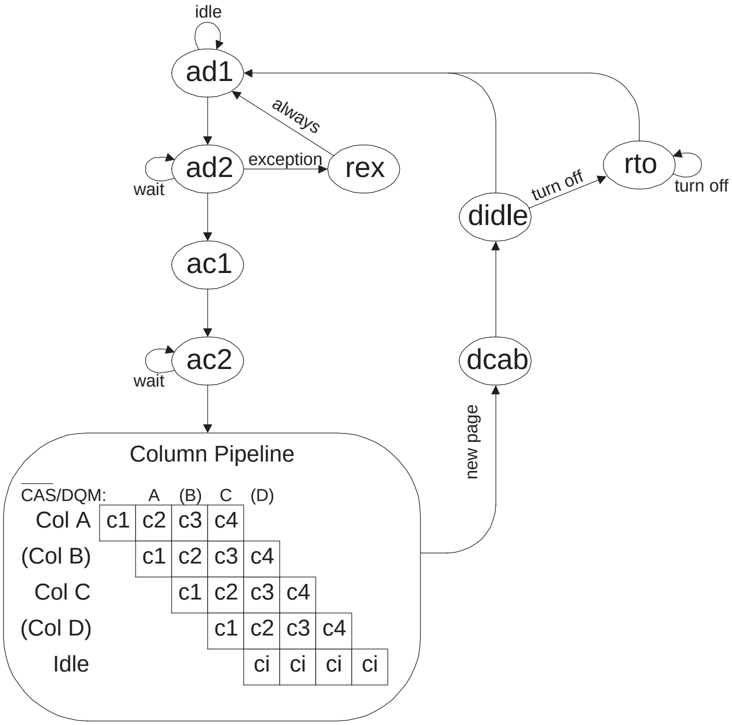
Figure 16–27.Burst Length 2, Latency 2 SDRAM Read (Single Bubble During Burst)



16.5.2 SDRAM Burst Length 2, Latency 3 Read

Read latency 3 can be used for SDRAMs that can return data within three ‘C82 clock cycles of the read command. The state sequence for this access is shown in Figure 16–28. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–28.SDRAM Burst Length 2, Latency 3 Read States



CAS/DQM:		A	(B)	C	(D)			
Col A	c1	c2	c3	c4				
(Col B)		c1	c2	c3	c4			
Col C			c1	c2	c3	c4		
(Col D)				c1	c2	c3	c4	
Idle					ci	ci	ci	ci

Column accesses consist of a four stage pipeline. A new read access can be loaded into the pipeline on every cycle but each command takes four clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0]. If the access is the first in a burst, a read command is issued by driving TRG/CAS low. The column status code is driven on STATUS[1:0].

c2 The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The command signals driven in c1 are deactivated.

c3 The CAS/DQM outputs are deactivated. The ‘C82 waits for data.

c4 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. This could be due to a pipeline bubble or because the second column location in the burst is not needed. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

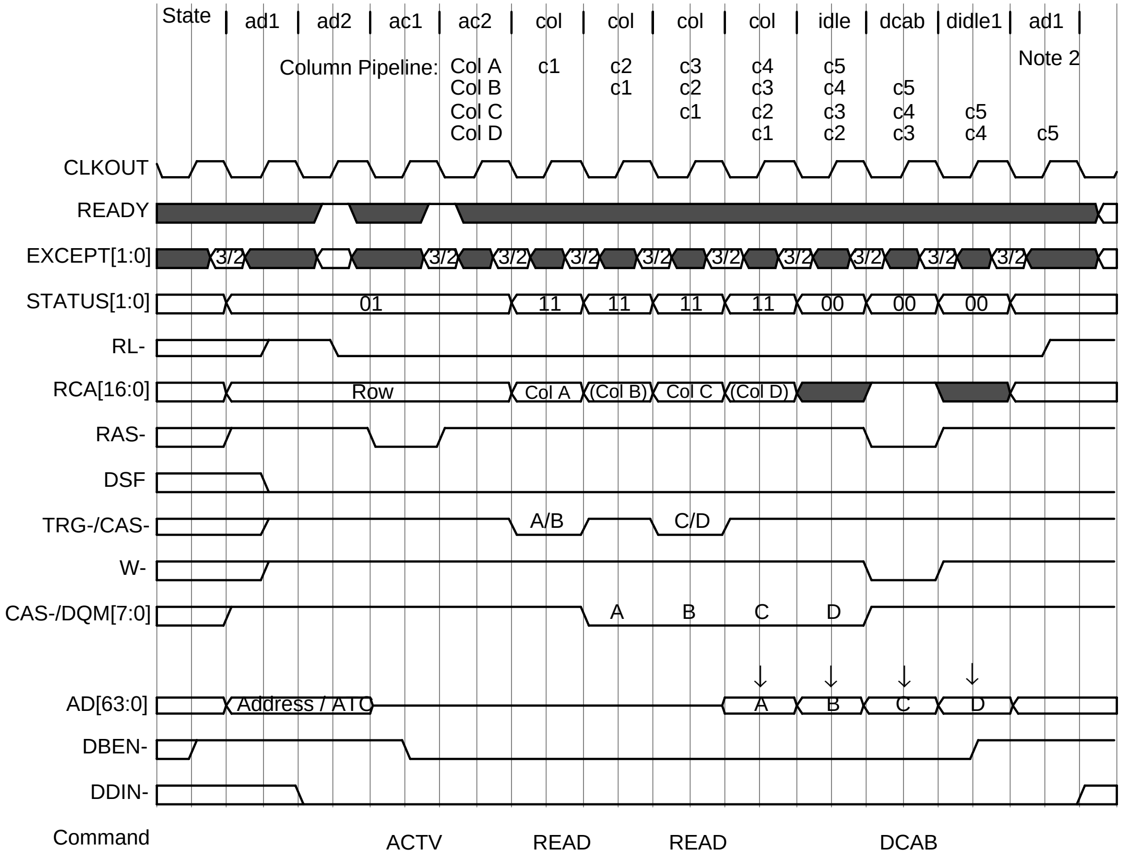
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c3 stage of the second column access in the final burst. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c4 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.5.2.1 SDRAM, Burst Length 2, Latency 3 Read Example

An example of a burst length 2, latency 3 SDRAM read access is shown in Figure 16–29. The example shows a read of four column addresses. In the example, Col A and Col B are within the same burst pair as are Col C and Col D. The first read command causes the Col A/Col B burst to be output and the second read command causes the Col C/Col D burst to be output. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–29.Burst Length 2, Latency 3 SDRAM Read

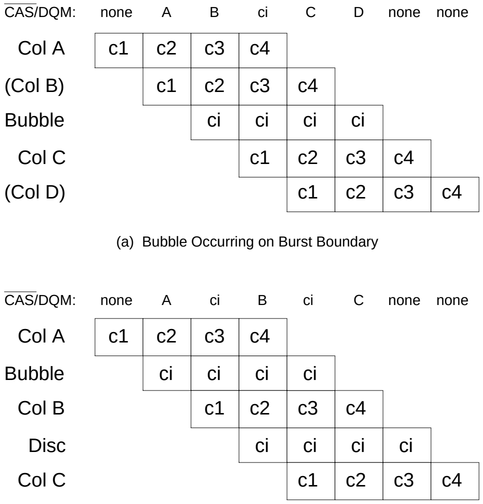


Note 2: Turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.5.2.2 Burst Length 2, Latency 3 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 2 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, c3, and c4 stages. This effect is shown in Figure 16–30.

Figure 16–30.Effect of Bubbles on Burst Length 2, Latency 3 SDRAM Read Column Pipeline



$\overline{\text{CAS}}$/DQM:	none	A	B	ci	C	D	none	none
Col A	c1	c2	c3	c4				
(Col B)		c1	c2	c3	c4			
Bubble			ci	ci	ci	ci		
Col C				c1	c2	c3	c4	
(Col D)					c1	c2	c3	c4

(a) Bubble Occurring on Burst Boundary

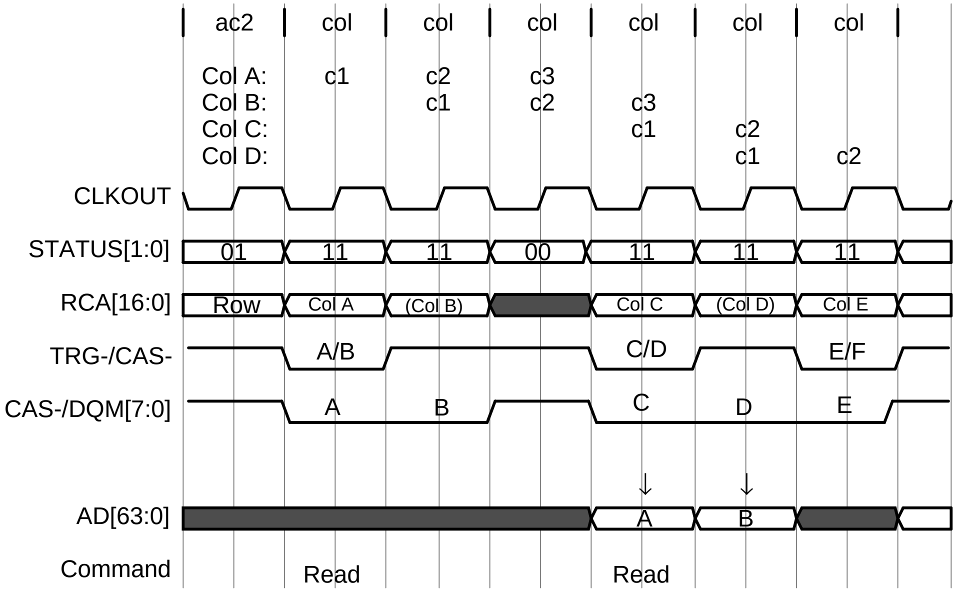
$\overline{\text{CAS}}$/DQM:	none	A	ci	B	ci	C	none	none
Col A	c1	c2	c3	c4				
Bubble		ci	ci	ci	ci			
Col B			c1	c2	c3	c4		
Disc				ci	ci	ci	ci	
Col C					c1	c2	c3	c4

(b) Bubble Occurring During Burst

Figure 16–30(a) shows the column pipeline sequence of a bubble occurring between bursts. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble occurring between bursts is shown in Figure 16–31. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

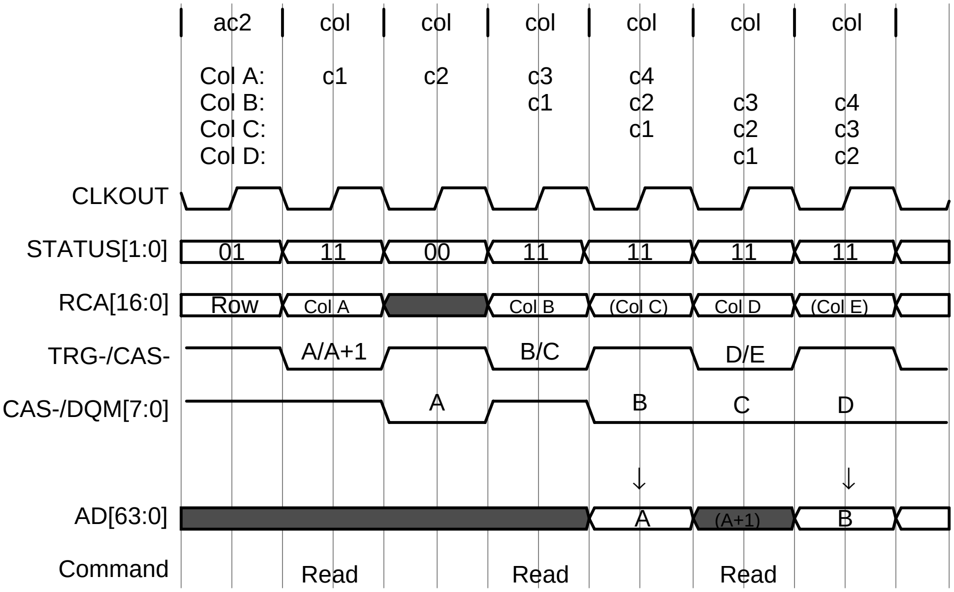
Figure 16–31.Burst Length 2, Latency 3 SDRAM Read (Single Bubble Between Bursts)



When the TC does not need the second access in a burst, a burst discontinuity occurs. Since commands cannot be issued on consecutive clock cycles, the TC must wait one clock before it can load the column pipeline for the next (noncontigous) column access. On the external memory interface, this operation appears much like a pipeline bubble.

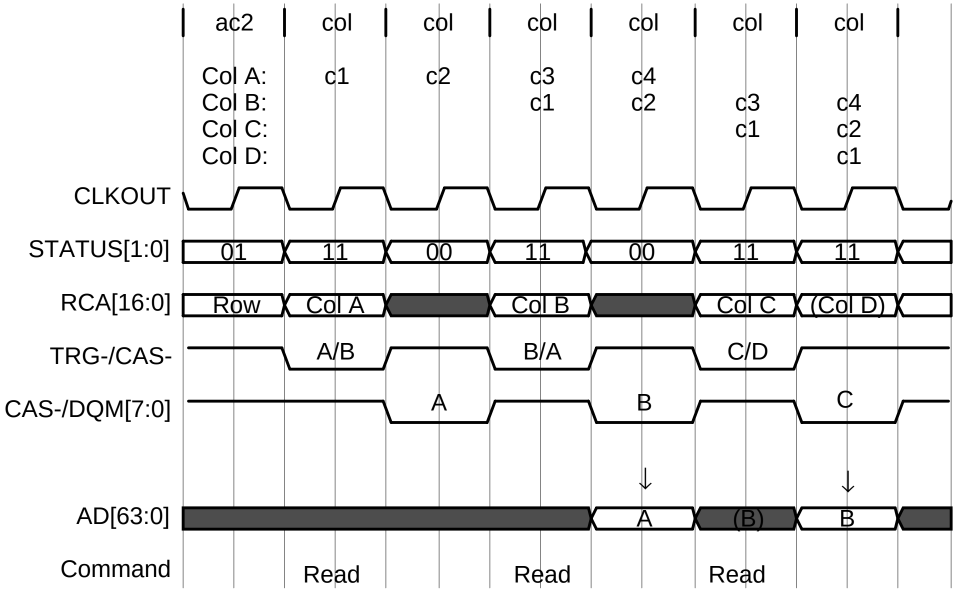
Figure 16–32 shows an example of a burst discontinuity. In the example, Col A and Col B are not within the same 2 column burst. The first read command accesses Col A and the next column in the burst (A+1). Since the TC needs to read Col B next rather than Col A+1, it must wait one cycle before loading its pipeline with the Col B access. The TC disables the CAS/DQM outputs during the c3 state of Col A to indicate that it won’t be using the A+1 data. After waiting a clock cycle (thus satisfying the “2n” rule) the TC then loads the pipeline with the Col B access. A new read command is generated for the two burst sequence of Col B and Col C and the pipeline continues normal operation.

Figure 16–32.Burst Length 2 Latency 3 SDRAM Read with Burst Discontinuity



When a pipeline bubble occurs within a burst, both a bubble and discontinuity will occur as shown in Figure 16–30(b). In this example, Col A and Col B are a “col 0/ col 1” burst pair. The TC loads Col A into its pipeline, generating a read command of the ColA/ColB burst. However, a pipeline bubble prevents the TC from loading Col B on the next clock. This prevent the TC from using the Col B data output during the second cycle of the burst. On the next clock cycle the Col B access is loaded. This generates a read command of the Col B/Col A burst. Since the Col A data has already been read, the second access of the burst is not needed so a burst discontinuity occurs. Figure 16–33 shows the bubble and discontinuity as they appear externally.

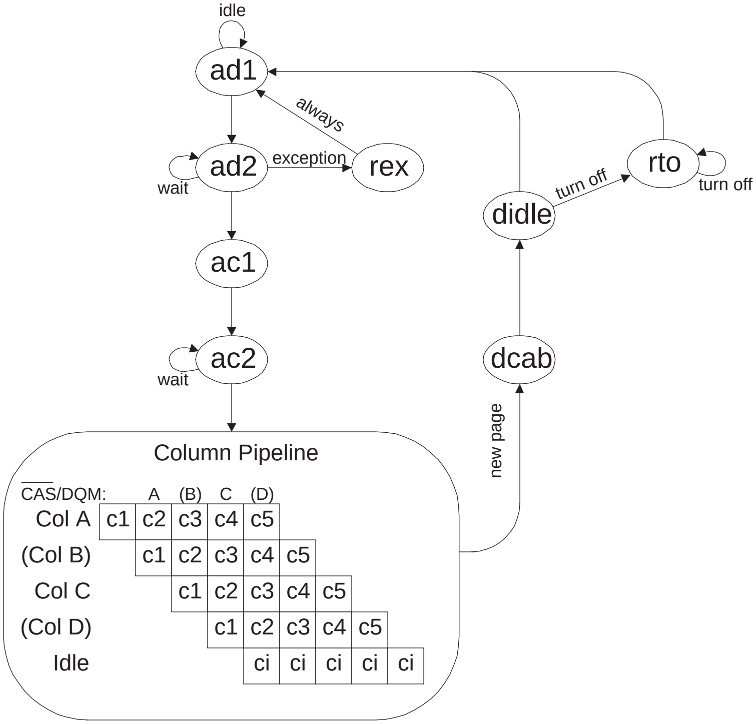
Figure 16–33.Burst Length 2, Latency 3 SDRAM Read (Single Bubble During Burst)



16.5.3 SDRAM Burst Length 2, Latency 4 Read

Read latency 4 can be used in systems requiring an extra cycle of access time. The SDRAMs are programmed for latency 3 reads but the ‘C82 delays the latching of data by one cycle. This allows the use of clocked data buffers between the ‘C82 and SDRAM. The state sequence for this access is shown in Figure 16–34. The row access consists of the ad1, ad2, ac1 and ac2 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state.

Figure 16–34.SDRAM Burst Length 2, Latency 4 Read States



CAS/DQM:		A	(B)	C	(D)				
Col A	c1	c2	c3	c4	c5				
(Col B)		c1	c2	c3	c4	c5			
Col C			c1	c2	c3	c4	c5		
(Col D)				c1	c2	c3	c4	c5	
Idle					ci	ci	ci	ci	ci

Column accesses consist of a five stage pipeline. A read command can begin on every cycle but each command takes five clock cycles to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0]. If the access is the first in a burst, a read command is issued by driving TRG/CAS low. The column status code is driven on STATUS[1:0].

c2 The SDRAM outputs are enabled for two clock cycles later by driving all the CAS/DQM outputs low. The command signals driven in c1 are deactivated.

c3 The CAS/DQM outputs are deactivated.

c4 The ‘C82 waits for data. The SDRAMs will typically output valid data at this time where it can be latched into an external buffer.

c5 The ‘C82 latches the read data.

ci The column idle stage occurs when no access is loaded in the pipeline. This could be due to a pipeline bubble or because the second column location in the burst is not needed. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state. Input data from previous accesses will still be latched.

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

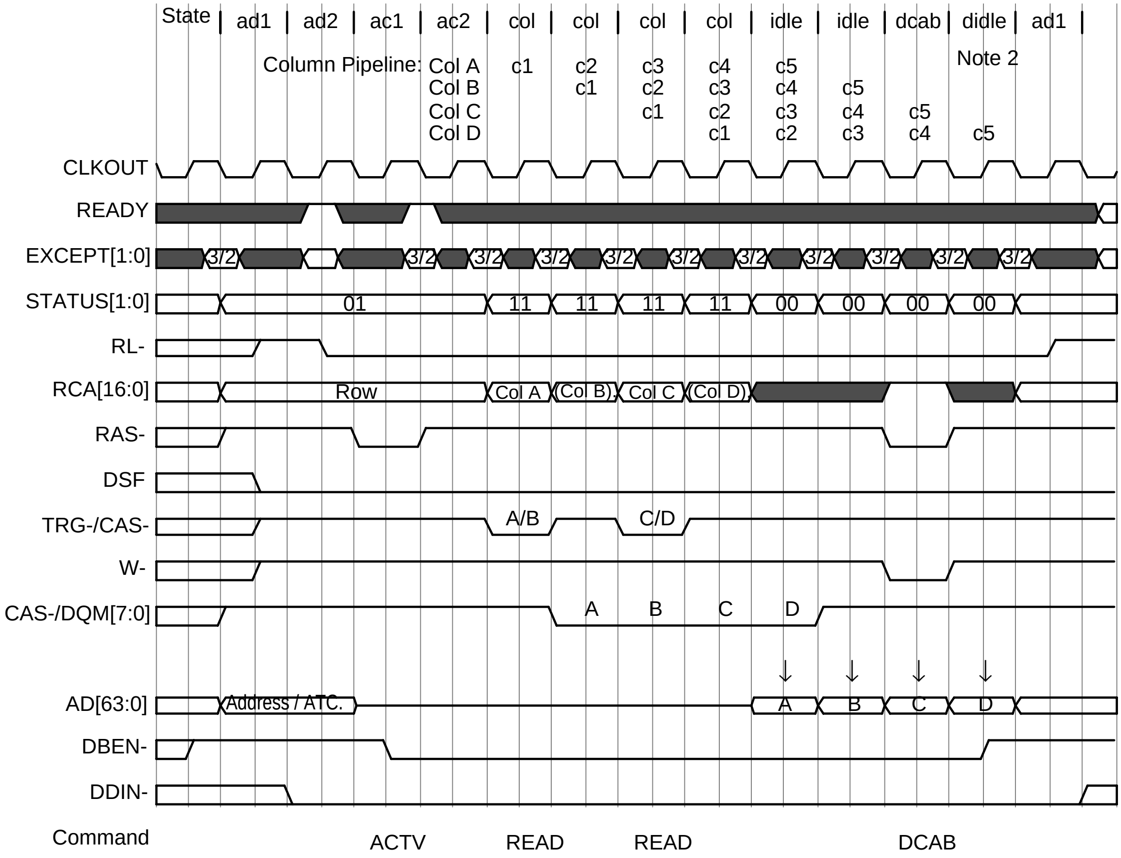
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may actually coincide with the c4 stage of the second column access in the final burst. After completing the DCAB, a dcab idle (DIDLE) state occurs to allow the final read data to be latched (stage c5 of the last column access) if needed. One or more rto states are inserted if specified in the configuration cache entry and the interface then returns to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.5.3.1 SDRAM, Burst Length 2, Latency 4 Read Example

An example of a burst length 2, latency 4 SDRAM read access is shown in Figure 16–35. The example shows a read of four column addresses. In the example, Col A and Col B are within the same burst pair as are Col C and Col D. The first read command causes the Col A/Col B burst to be output and the second read command causes the Col C/Col D burst to be output. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–35.Burst Length 2, Latency 4 SDRAM Read



Note 2: Additional turn-off cycles will be inserted between didle and ad1 as specified by the bank configuration.

16.5.3.2 Burst Length 2, Latency 4 Reads with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 1 reads, the loading of a new column access into the pipeline is delayed. Previously loaded accesses continue to completion through their c1, c2, c3, c4, and c5 stages. This effect is shown in Figure 16–36.

Figure 16–36.Effect of Bubbles on Burst Length 2, Latency 4 SDRAM Read Column Pipeline

$\overline{\text{CAS}}$/DQM:	none	A	B	ci	C	D	none	none	none
Col A	c1	c2	c3	c4	c5				
(Col B)		c1	c2	c3	c4	c5			
Bubble			ci	ci	ci	ci	ci		
Col C				c1	c2	c3	c4	c5	
(Col D)					c1	c2	c3	c4	c5

(a) Bubble Occurring on Burst Boundary

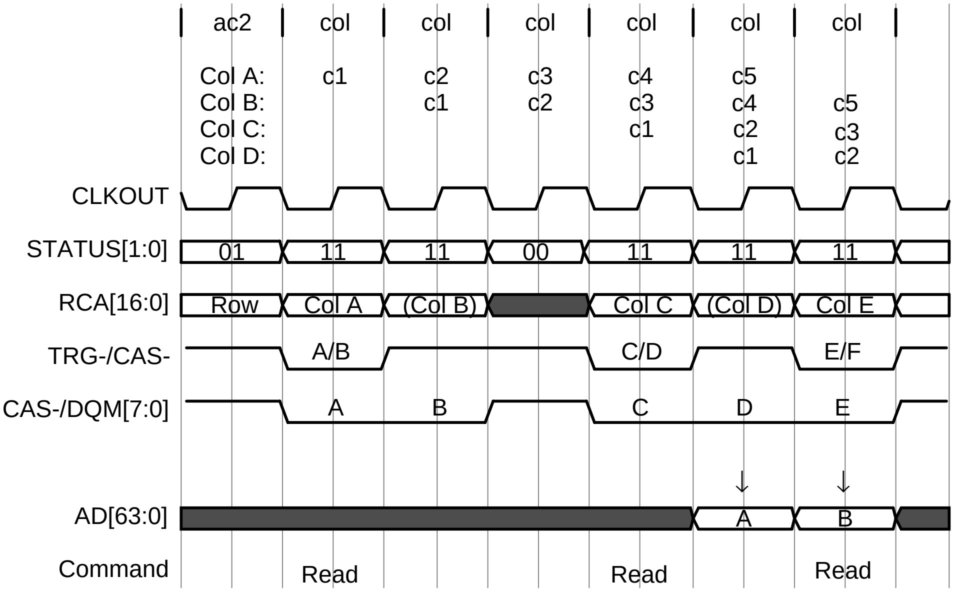
$\overline{\text{CAS}}$/DQM:	none	A	ci	B	ci	C	none	none	none
Col A	c1	c2	c3	c4	c5				
Bubble		ci	ci	ci	ci	ci			
(Col B)			c1	c2	c3	c4	c5		
Disc				ci	ci	ci	ci	ci	
Col C					c1	c2	c3	c4	c5

(b) Bubble Occurring During Burst

Figure 16–36(a) shows the column pipeline sequence of a bubble occurring between burst accesses. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. Stages c3 of Col A and c2 of Col B complete as normal. In the next cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble between bursts is shown in Figure 16–37. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

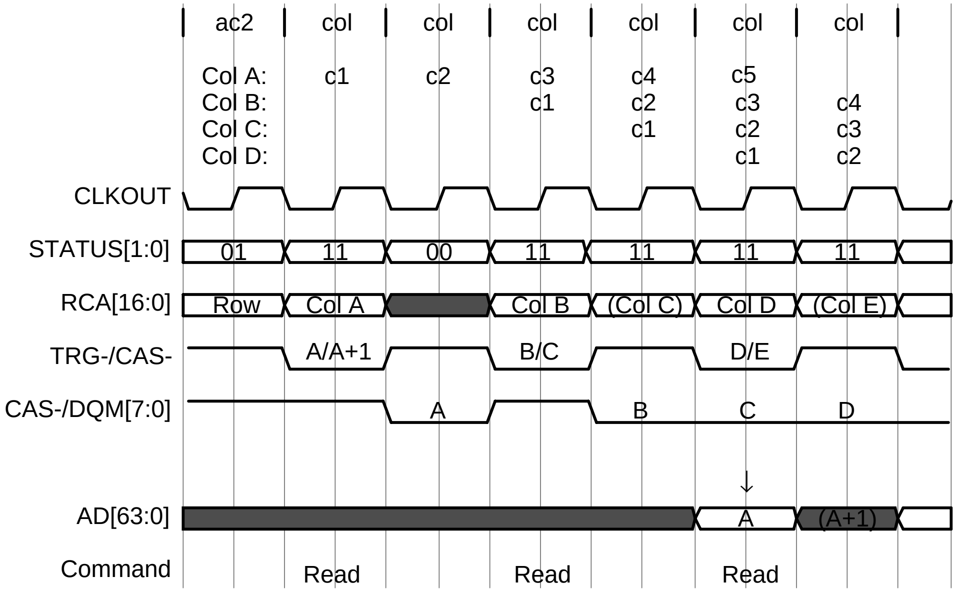
Figure 16–37.Burst Length 2, Latency 4 SDRAM Read (Single Bubble Between Bursts)



When the TC does not need the second access in a burst, a burst discontinuity occurs. Since commands cannot be issued on consecutive clock cycles, the TC must wait one clock before it can load the column pipeline for the next (noncontigous) column access. On the external memory interface, this operation appears much like a pipeline bubble.

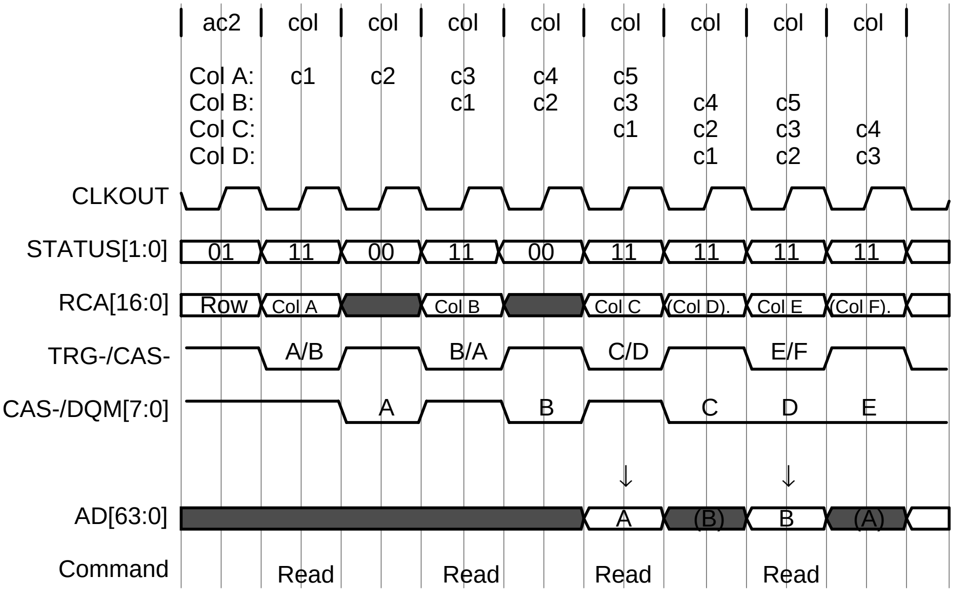
Figure 16–38 shows an example of a burst discontinuity. In the example, Col A and Col B are not within the same 2 column burst. The first read command accesses Col A and the next column in the burst (A+1). Since the TC needs to read Col B next rather than Col A+1, it must wait one cycle before loading its pipeline with the Col B access. The TC disables the CAS/DQM outputs during the c3 state of Col A to indicate that it won’t be using the A+1 data. After waiting a clock cycle (thus satisfying the “2n” rule) the TC then loads the pipeline with the Col B access. A new read command is generated for the two burst sequence of Col B and Col C and the pipeline continues normal operation.

Figure 16–38.Burst Length 2 Latency 4 SDRAM Read with Burst Discontinuity



When a pipeline bubble occurs within a burst, both a bubble and discontinuity will occur as shown in Figure 16–36(b). In this example, Col A and Col B are a “col 0/ col 1” burst pair. The TC loads Col A into its pipeline, generating a read command of the ColA/ColB burst. However, a pipeline bubble prevents the TC from loading Col B on the next clock. This prevent the TC from using the Col B data output during the second cycle of the burst. On the next clock cycle the Col B access is loaded. This generates a read command of the Col B/Col A burst. Since the Col A data has already been read, the second access of the burst is not needed so a burst discontinuity occurs. Figure 16–39 shows the bubble and discontinuity as they appear externally.

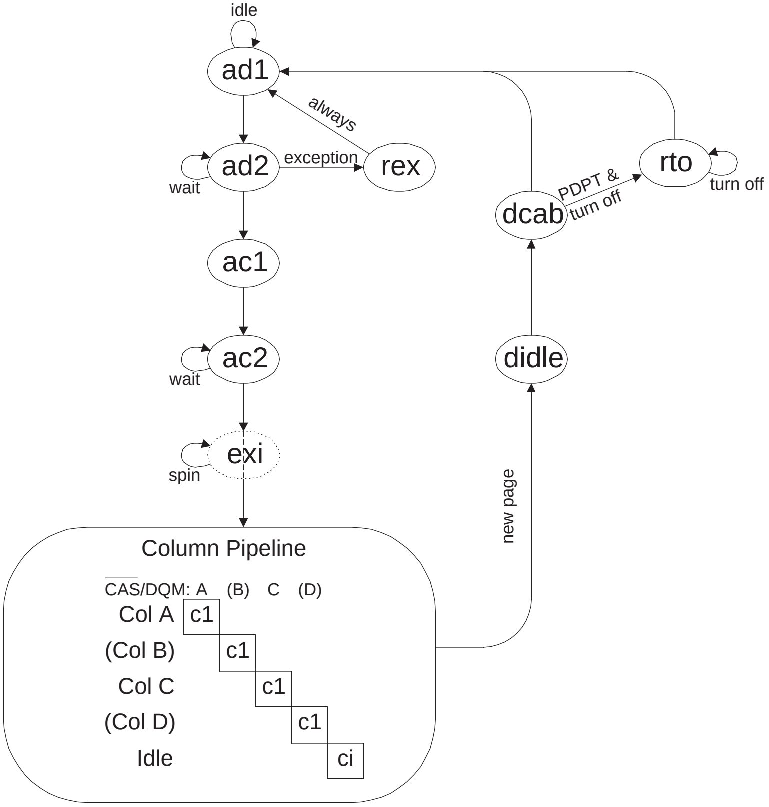
Figure 16–39.Burst Length 2, Latency 4 SDRAM Read (Single Bubble During Burst)



16.5.4 SDRAM Burst Length 2 Write

Burst length 2 write accesses are performed on all SDRAMs configured for burst length 2. The state sequence for this access is shown in Figure 16–40. The row access consists of the ad1, ad2, ac1, ac2, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi cycles such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

Figure 16–40.SDRAM Burst Length 2, Write States



Column accesses consist of a single stage pipeline. A write command can begin on every cycle and each command takes one clock cycle to complete. The pipeline stages are a follows:

c1 The column address is output on RCA[16:0]. If the access is the first in a burst, a write command is issued by driving TRG/CAS and W low. The column status code is driven on STATUS[1:0]. The write data is driven on AD[63:0] and the appropriate CAS/DQM[7:0] signals are driven low to indicate the valid bytes.

ci The column idle stage occurs when no access is loaded in the pipeline. This could be due to a pipeline bubble or because the second column location in the burst is not needed. The idle code is output on STATUS[1:0] to indicate that no new column access is beginning. All command signals are driven to their inactive state.

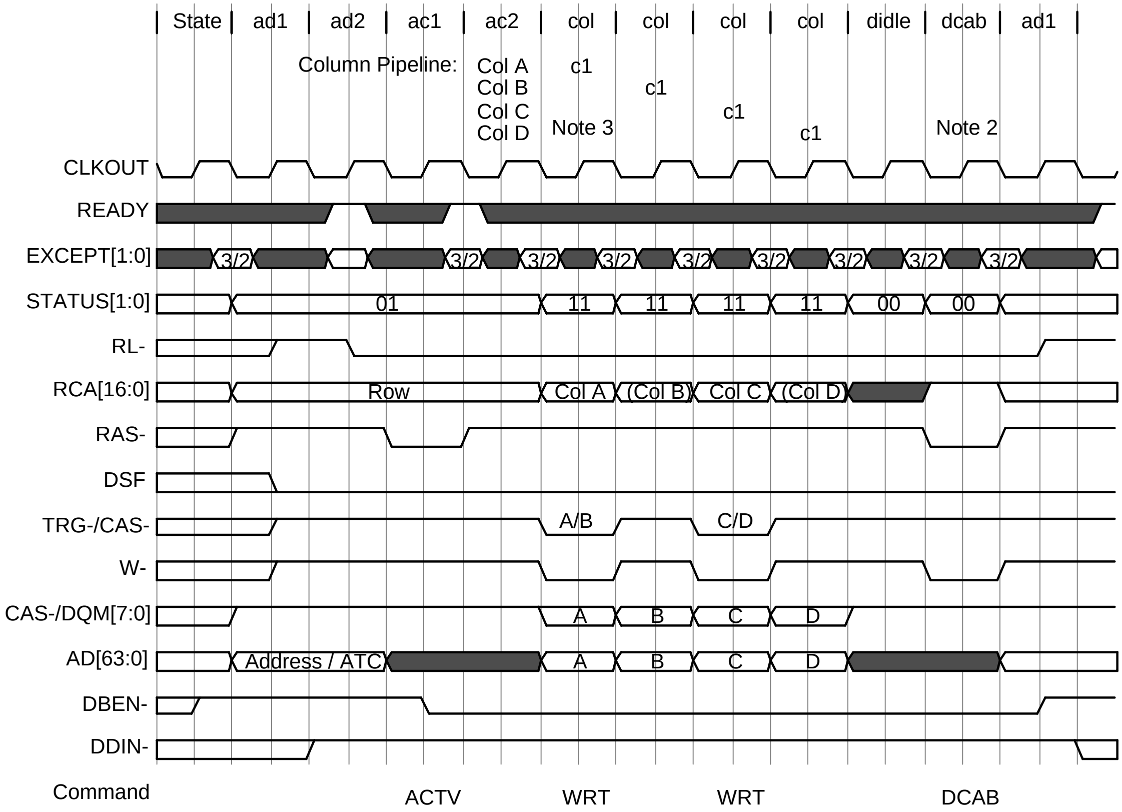
After the last requested column access, if a new page access is required, the memory interface deactivates the current page with a DCAB command. The DCAB command may occur no sooner than 2 clock cycles after the last write in a burst so a DIDLE state occurs prior to the DCAB to ensure that SDRAM timing is met. If the write access is a result of a peripheral device packet transfer, then one or more rto states will be inserted as specified in the configuration cache entry prior to returning to the ad1 state.

If no access is pending after the last column access, the memory interface will remain in the ci (column idle) pipeline stage waiting for the next access. If the next access does not require a new page access, then the column pipeline sequence will begin again.

16.5.4.1 SDRAM, Burst Length 2, Write Example

An example of a burst length 2, SDRAM write access is shown in Figure 16–41. The example shows a write of four column addresses. In the example, Col A and Col B are within the same burst pair as are Col C and Col D. The first write command causes the Col A/Col B burst to be written and the second write command causes the Col C/Col D burst to be written. After the access to Col D, the memory interface deactivates the SDRAM bank and returns to the ad1 state to begin a new page access.

Figure 16–41.Burst Length 2 SDRAM Write



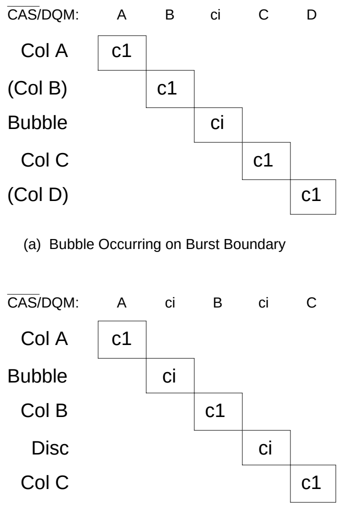
Note 2: During peripheral data transfers, turn-off cycles will be inserted prior to ad1 as specified by the bank configuration.

Note 3: When exceptions are enabled, additional cycles will be inserted after ac2 a required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

16.5.4.2 Burst Length 2, Writes with Pipeline Bubbles

When a TC pipeline bubble occurs during burst length 2 writes, the loading of a new column access into the pipeline is delayed. This effect is shown in Figure 16–42.

Figure 16–42.Effect of Bubbles on Burst Length 2, SDRAM Write Column Pipeline



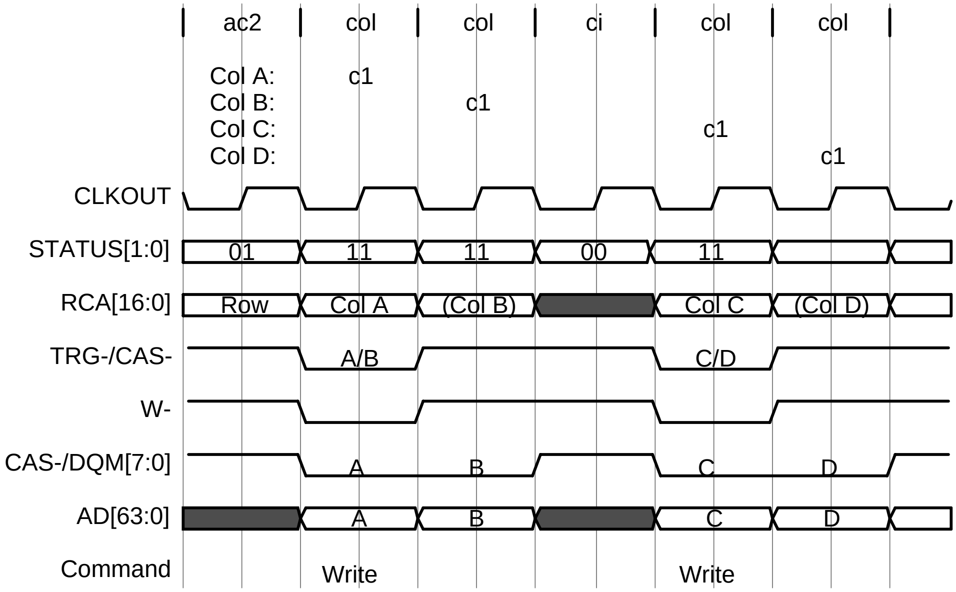
(a) Bubble Occurring on Burst Boundary

(b) Bubble Occurring During Burst

Figure 16–42(a) shows the column pipeline sequence of a bubble occurring between burst accesses. After loading column accesses A and B into the pipeline, a bubble occurs preventing the Col C access from beginning on the next cycle. After the bubble cycle, the Col C access is ready to begin and the pipeline resumes normal operation. Additional bubbles would delay the loading of the pipeline by one cycle per bubble.

The column-time waveform for a single-cycle bubble is shown in Figure 16–43. The bubble causes an idle status code to be output for one cycle indicating that no new column access is beginning. No command is issued during the bubble.

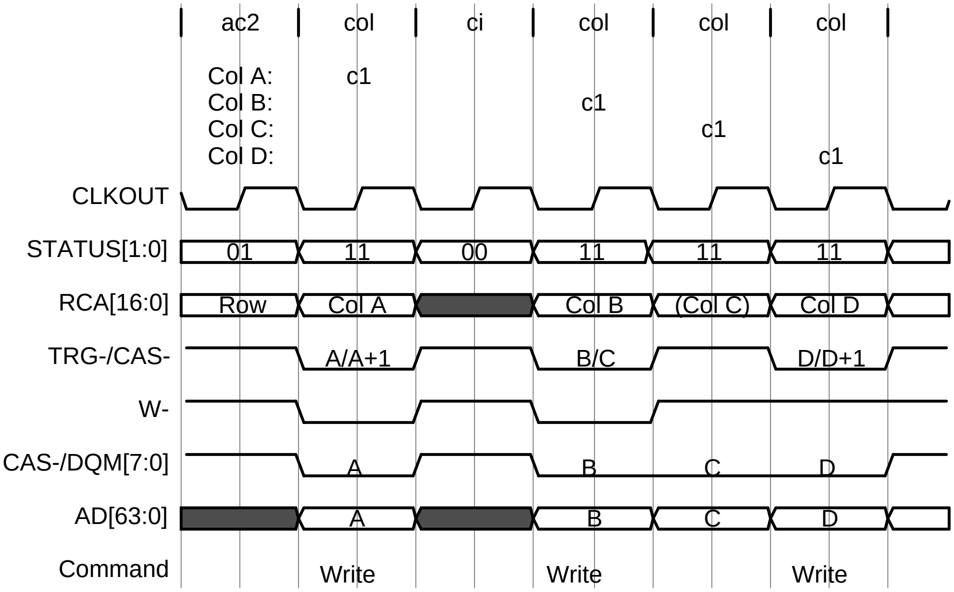
Figure 16–43.Burst Length 2 SDRAM Write (Single Cycle Bubble)



When the TC does not need the second access in a burst, a burst discontinuity occurs. Since commands cannot be issued on consecutive clock cycles, the TC must wait one clock before it can load the column pipeline with the next (noncontiguous) column access. On the external memory interface, this operation appears much like a pipeline bubble.

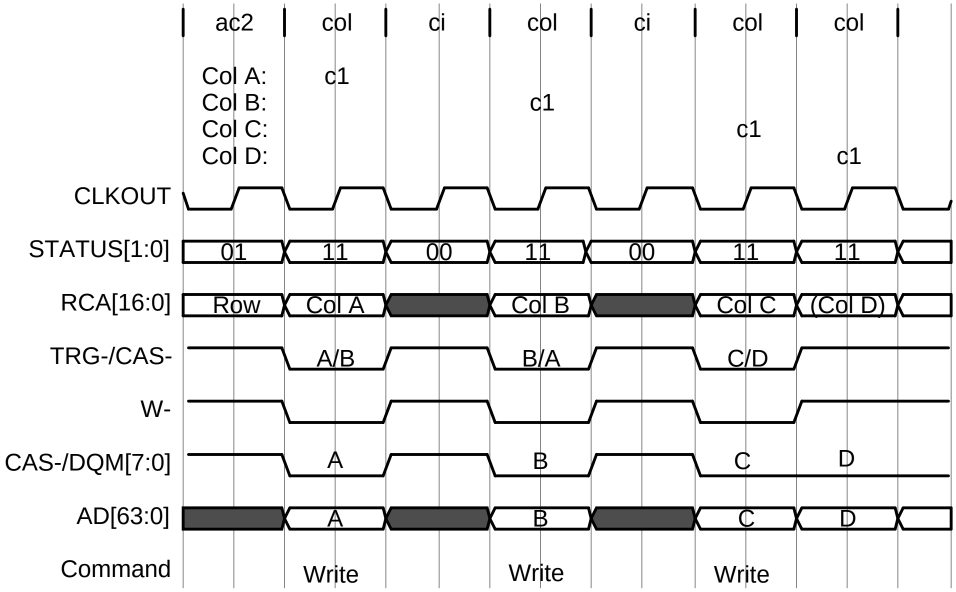
Figure 16–44 shows an example of a burst discontinuity. In the example, Col A and Col B are not within the same two column burst. The first write command accesses Col A and the next column in the burst (A+1). Since the TC needs to read Col B next rather than Col A+1, it must wait one clock cycle before loading the pipeline with the Col B access. The TC disables the CAS/DQM outputs during the second access of the burst to indicate that it won’t be using the Col A+1 data. After waiting one clock cycle (thus satisfying the “2n” rule) the TC then loads the pipeline with the Col B access. A new write command is generated for the two burst sequence of Col B and Col C and the pipeline continues normal operation.

Figure 16–44.Burst Length 2 SDRAM Write with Burst Discontinuity



When a pipeline bubble occurs within a burst, both a bubble and a discontinuity will occur as shown in Figure 16–42(b). In this example, Col A and Col B are a “col 0/col 1” burst pair. The TC loads Col A into its pipeline generating a write command of the Col A/Col B burst. However, a pipeline bubble prevents the TC from loading Col B on the next clock cycle. This prevents the TC from writing the Col B data during the second cycle of the burst. On the next clock cycle, the Col B access is loaded. This generates a write command of the Col B/Col A burst. Since the Col A data has already been written, the second access of this burst is not needed and a burst discontinuity occurs. Figure 16–45 shows the bubble and discontinuity as they appear externally.

Figure 16–45.Burst Length 2 SDRAM Write (Single Bubble During Burst)



Chapter 17

VRAM Cycles

In addition to standard EDO DRAM read and write cycles, the external memory interface supports many of the special cycles available on EDO Video RAMs (VRAMs). This facilitates the use of VRAMs as display buffer devices in high performance or high resolution systems.

Topics in this chapter include:

Topic **Page**

17.1 Load-Color-Register Access

Load-color-register (LCR) accesses write a value into VRAM color registers for later use during block-write operations. LCR cycles can occur for memory banks with DRAM timing (**CT(3:0)** = ‘00xx’) and block-write support (**BW(1:0)** = ‘1x’). An LCR will be performed whenever the TC begins servicing a long-form packet transfer which uses the block write access mode. (See Section 8.1, *Block-Write Modes*)

The timing for an LCR access is identical to that of a normal write of a single column location. During an LCR access, however, the DSF output is high at the fall of RAS and CAS/DQM. LCR accesses are only supported for 64-bit data buses so all CAS/DQM[7:0] outputs will be activated.

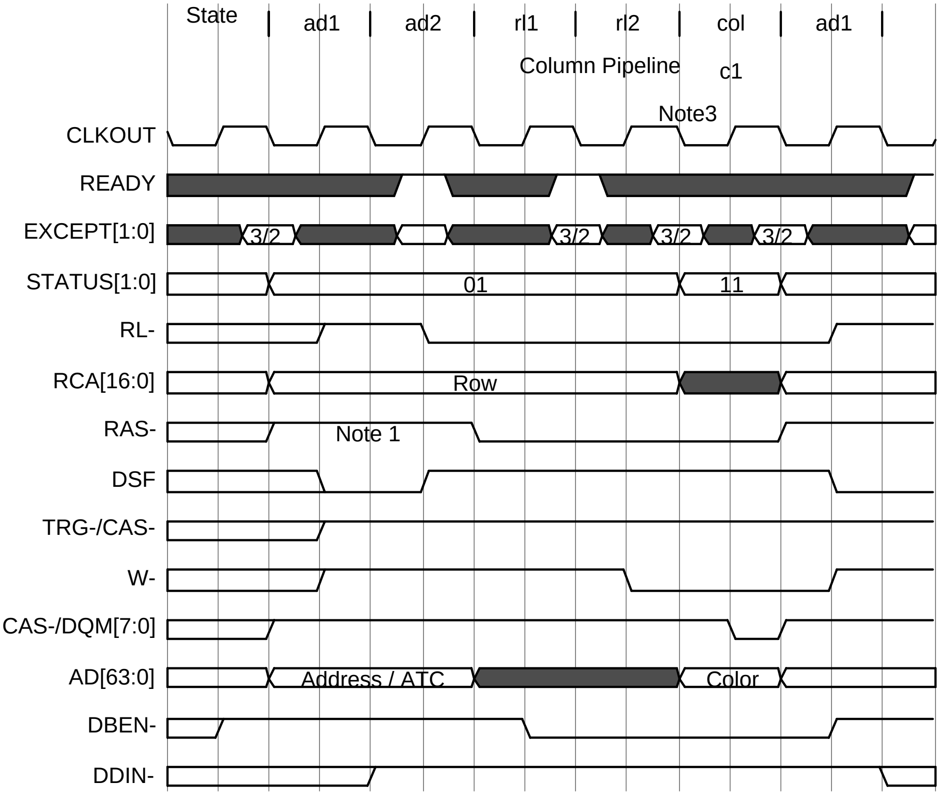
17.1.1 1 Cycle/Column LCR Access

The 1 cycle/column LCR access is performed for memories with a configuration cache entry selecting either pipelined or non-pipelined 1 cycle/column timing (**CT(3:0)** = ‘000x’).

An example of a 1 cycle/column LCR access is shown in Figure 17–1. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column access consists of a single c1 pipeline stage during which the color register value is output on the AD[63:0] bus.

Figure 17–1. 1 Cycle/Column Load Color Register



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the c1 stage.

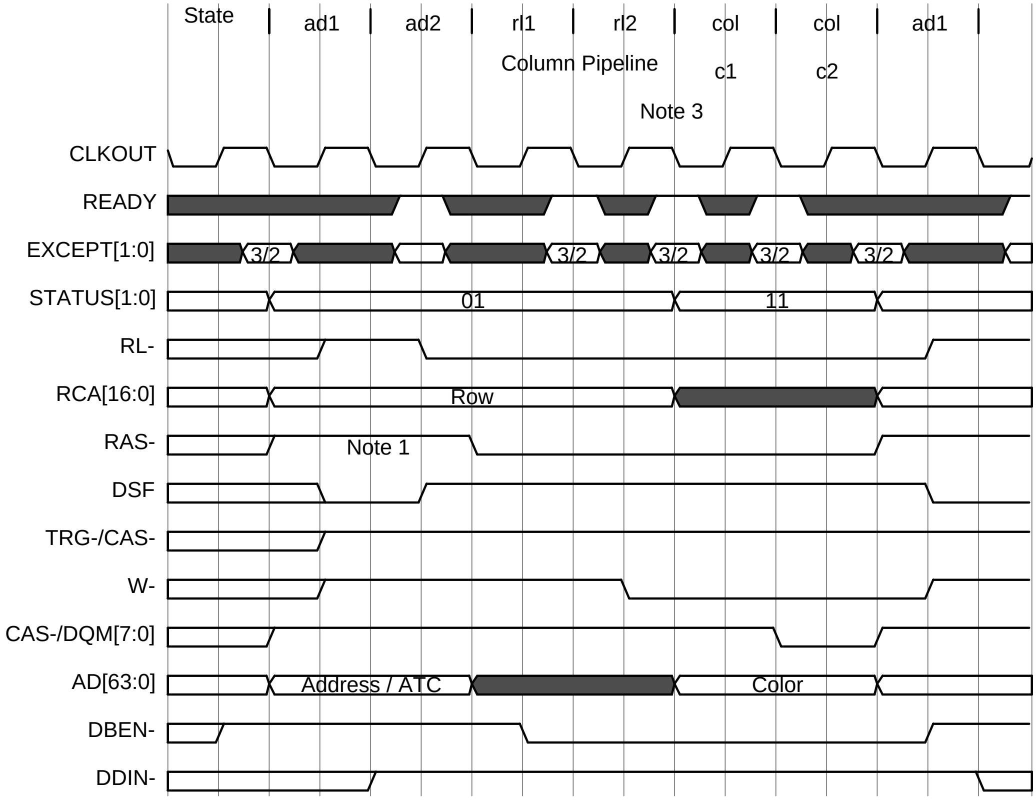
17.1.2 2 Cycle/Column LCR Access

The 2 cycle/column LCR access is performed for memories with a configuration cache entry selecting 2 cycle/column timing (**CT(3:0)** = ‘0010’).

An example of a 2 cycle/column LCR access is shown in Figure 17–2. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column access consists of c1 and c2 pipeline stages during which the color register value is output on the AD[63:0] bus.

Figure 17–2. 2 Cycle/Column Load Color Register



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS high of at least 3 cycles.
Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the c1 stage.

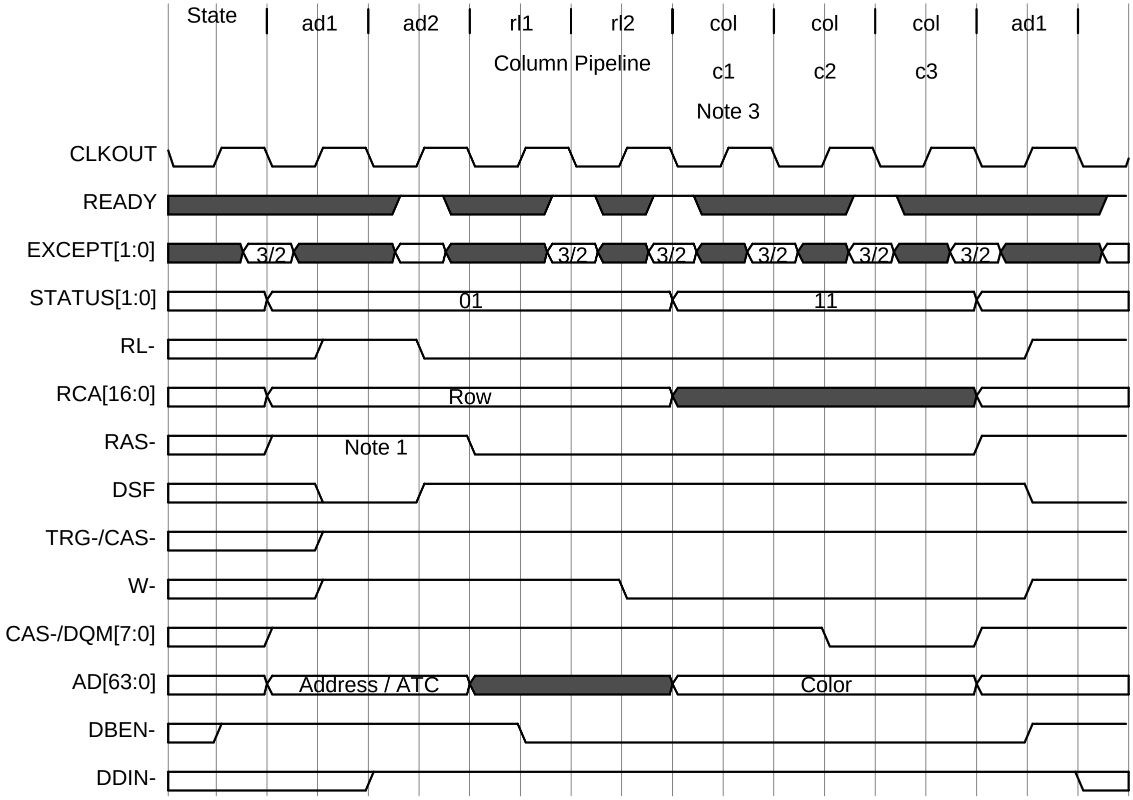
17.1.3 3 Cycle/Column LCR Access

The 3 cycle/column LCR access is performed for memories with a configuration cache entry selecting 3 cycle/column timing (**CT(3:0)** = ‘0011’).

An example of a 3 cycle/column LCR access is shown in Figure 17–3. The row access consists of the ad1, ad2, rl1, and rl2 states. Up to two rw states will be inserted after ad2 to ensure a RAS high time of at least 4 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column access consists of c1, c2, and c3 pipeline stages during which the color register value is output on the AD[63:0] bus.

Figure 17–3. 3 Cycle/Column Load Color Register



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS-high of at least 4 cycles.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the c1 stage.

17.2 VRAM Block-Write Access

Block write cycles cause the data stored in the VRAM color registers to be written to the memory locations enabled by the appropriate data bits the AD[63:0] bus. This allows as many as 64 bytes (depending on the type of block-write being performed) to be written in a single column access.

Block-write accesses can occur for memory banks with DRAM timing (**CT(3:0)** = ‘00xx’) and block-write support (**BW(1:0)** = ‘1x’). block write will be performed when the TC is servicing a long-form packet transfer which uses the block write access mode. (See Section 8.1, *Block-Write Modes*)

The timing for a block-write access is identical to that of a normal write access. However, a block-write differs from a standard write in the following signal characteristics:

- ❏ DSF is high at the fall of CAS/DQM, enabling the VRAM block-write function.
- ❏ Block-write is supported only for 64-bit buses. All CAS/DQM[7:0] outputs will be active.
- ❏ The two or three LSBs (depending on the type of block-write) of the column address are ignored by the VRAMs because these column locations are specified by the bits on the AD[63:0] bus.
- ❏ The values output by the ‘C82 on AD[63:0] (at column time) represent the column locations to be written using the color register value. Depending on the type of block-write supported by the VRAM, all of the data bits may not be used by the VRAMs.
- ❏ Block-write accesses always begin with a row access. Upon completion of the block-write, the memory interface will return to the ad1 state and will not enter column-time idle.

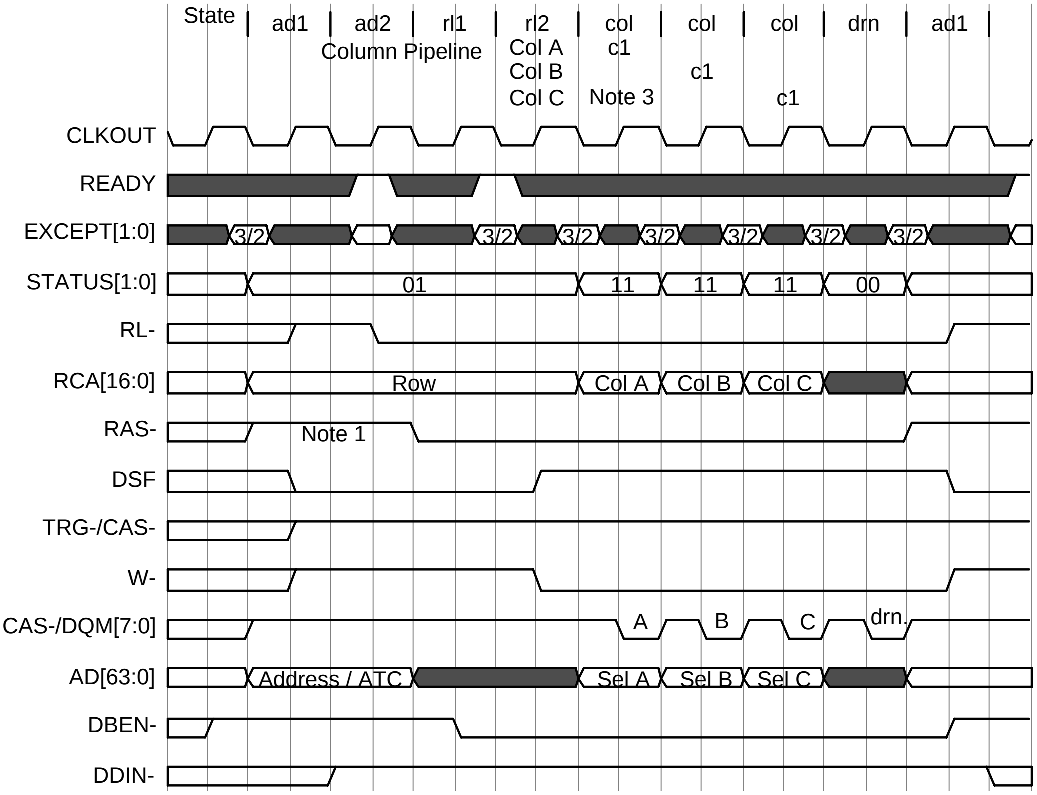
17.2.1 Pipelined 1 Cycle/Column Block-Write Access

The pipelined 1 cycle/column block-write access is performed for memories with a configuration cache entry selecting pipelined 1 cycle/column timing (**CT(3:0)** = ‘0000’).

An example of a pipelined 1 cycle/column block-write access is shown in Figure 17–4. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column accesses consist of a c1 pipeline stage during which the AD[63:0] bus indicates the column locations to which the color register value is written. Following the final block-write column access, the drn state will always be executed. This state activates all CAS/DQM strobes in order to update the memory array with the data last written to each byte.

Figure 17–4. Pipelined 1 Cycle/Column Block-Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure a RAS high time of at least 3 cycles.
Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

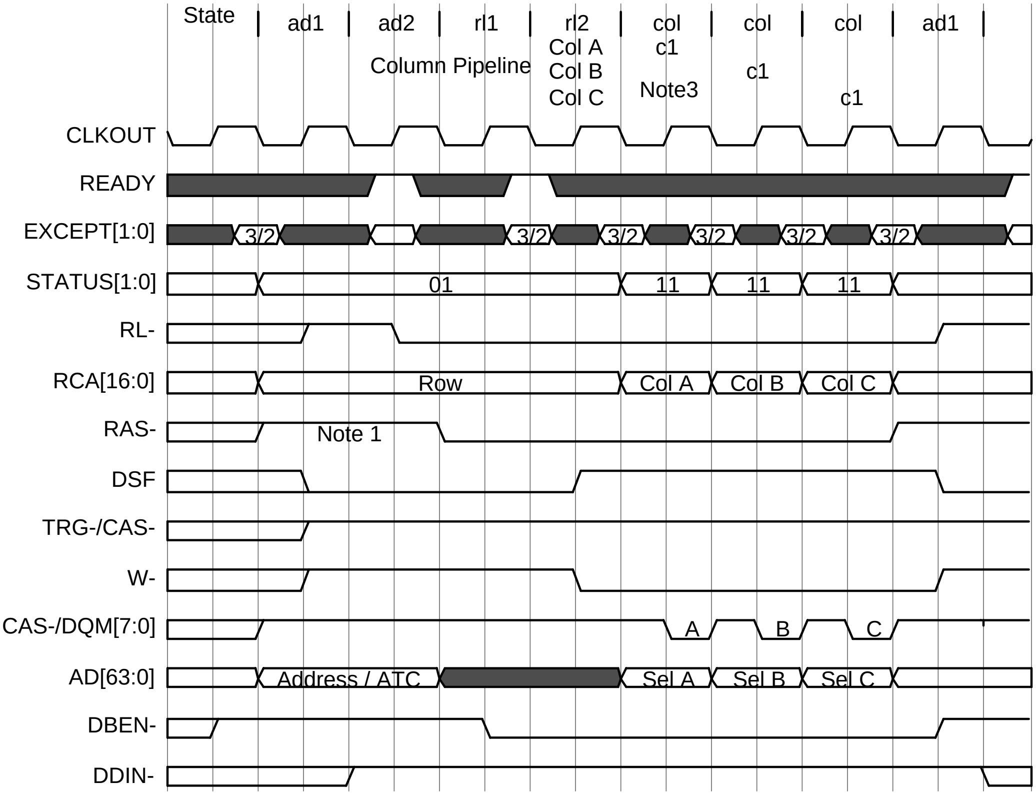
17.2.2 Nonpipelined 1 Cycle/Column Block-Write Access

The nonpipelined 1 cycle/column block-write access is performed for memories with a configuration cache entry selecting nonpipelined 1 cycle/column timing (**CT(3:0)** = ‘`0001`’).

An example of a nonpipelined 1 cycle/column block-write access is shown in Figure 17–5. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column accesses consist of a c1 pipeline stage during which the AD[63:0] bus indicates the column locations to which the color register value is written.

Figure 17–5. Nonpipelined 1 Cycle/Column Block-Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure a RAS high of at least 3 cycles.
Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

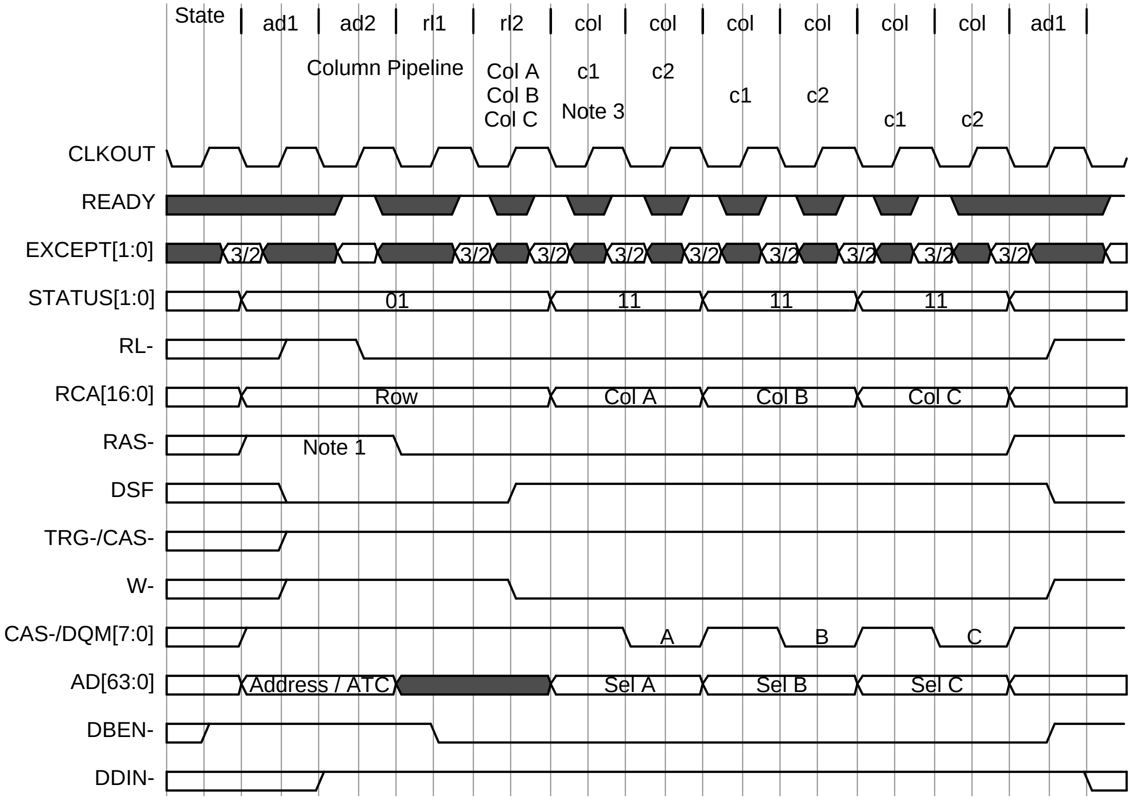
17.2.3 2 Cycle/Column Block-Write Access

The 2 cycle/column block-write access is performed for memories with a configuration cache entry selecting 2 cycle/column DRAM timing (**CT(3:0)** = ‘`0010`’).

An example of a 2 cycle/column block-write access is shown in Figure 17–6. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column accesses consist of c1 and c2 pipeline stages during which the AD[63:0] bus indicates the column locations to which the color register value is written.

Figure 17–6. 2 Cycle/Column Block-Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

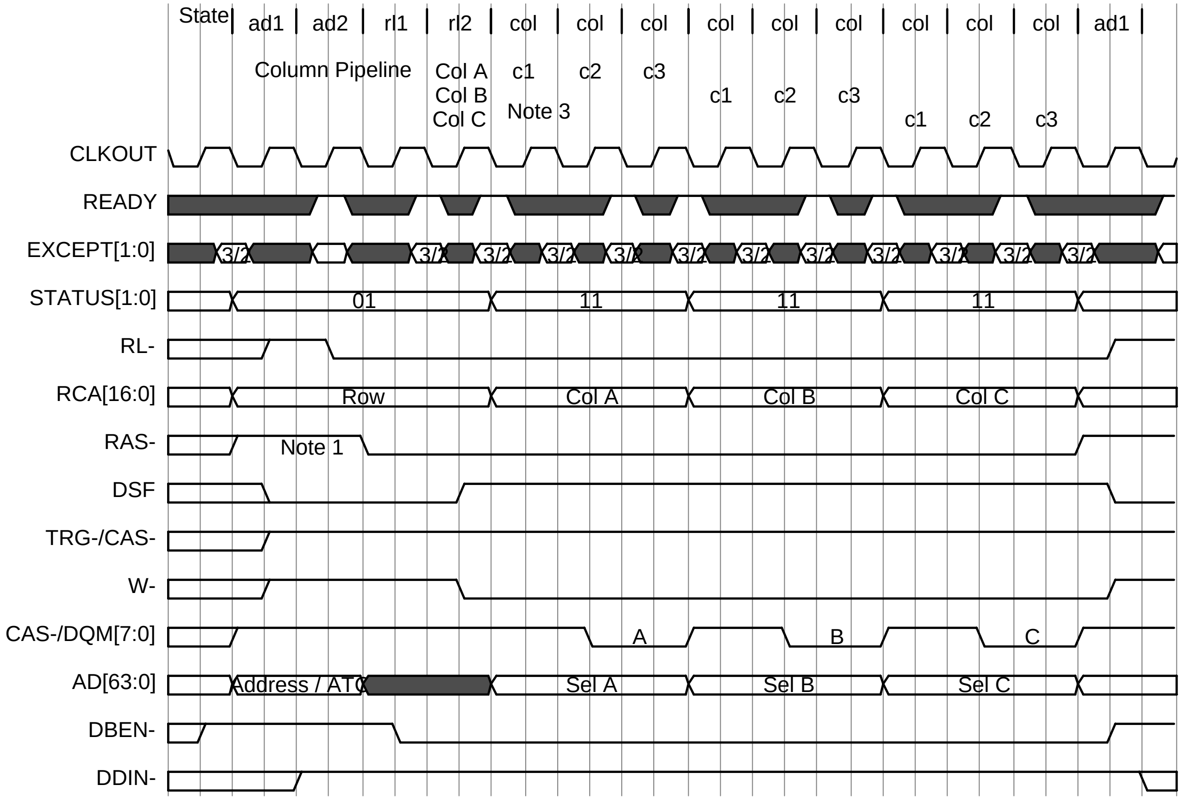
17.2.4 3 Cycle/Column Block-Write Access

The 3 cycle/column block-write access is performed for memories with a configuration cache entry selecting 3 cycle/column DRAM timing (**CT(3:0)** = ‘0011’).

An example of a 3 cycle/column block-write access is shown in Figure 17–7. The row access consists of the ad1, ad2, rl1, and rl2 states. Two rw states may be inserted after ad2 to ensure a RAS high time of at least 4 clock cycles. If exceptions are enabled in the configuration cache entry, exi states will be added after rl2 increasing the minimum number of cycles between ad2 and the column access from three to five.

The column accesses consist of c1 and c2 pipeline stages during which the AD[63:0] bus indicates the column locations to which the color register value is written.

Figure 17–7. 3 Cycle/Column Block-Write



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

Note 3: When exceptions are enabled, additional cycles will be inserted after rl2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

17.3 Read Transfer and Split Read Transfer Accesses

Read transfer (memory-to-register transfer) accesses transfer a row from the VRAM memory array into the VRAM shift register (SAM). This causes the entire SAM (both split SAMs) to be loaded with the array data.

Split read transfer (memory-to-split-register transfer) accesses also transfer data from a row in the memory array to the SAM. However, the split read transfer causes only half of the SAM (one split SAM) to be written. This allows the inactive half of the SAM to be loaded with new data while the other active half continues to shift data out.

Read and split read transfers are generated by short-form packet transfers that specify read or split read PT access modes (**PAM** = ‘011’ and **PAM** = ‘010’), respectively. For more information on generating read and split-read transfers see Section 8.4, *Read Transfer Modes*.

The memory interface state sequence for read and split read transfers is identical to that of a single access write cycle. However, a read or split-read transfer differs from a standard write in the following signal characteristics:

❑ W remains high throughout the cycle.
❑ TRG/CAS is low at the fall of RAS to enable the read transfer function on the addressed VRAMs.
❑ DBEN remains high throughout the access because no data is transferred between the memory and ‘C82.
❑ The data bus (AD[63:0]) is placed in high impedance.
❑ The values output by the ‘C82 on AD[63:0] (at column time) represent the SAM tap point. This determines which column location will be the first to be shifted out of the SAM.
❑ Read and split-read transfers always begin with a row access. Upon completion of the transfer, the memory interface will return to the ad1 state and will not enter column-time idle.
❑ For split-read transfers, DSF is high at the fall of RAS and CAS/DQM to indicate that the transfer is a split-read rather than a normal read transfer.

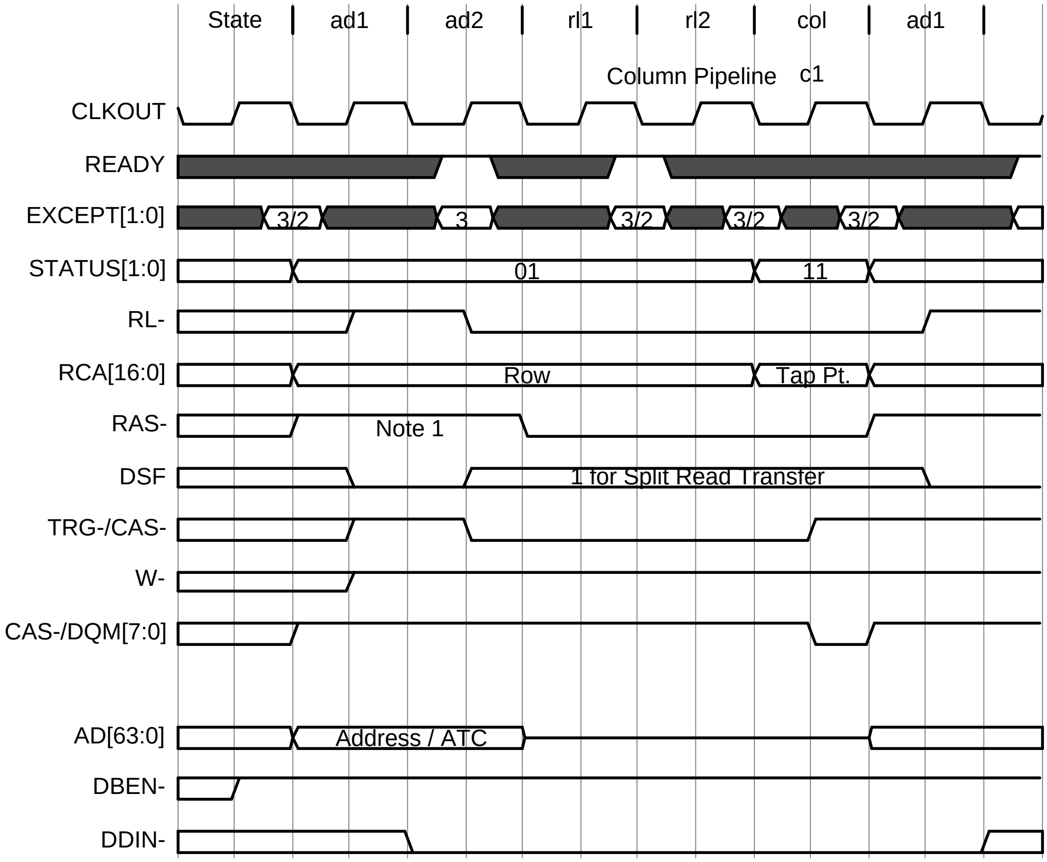
17.3.1 1 Cycle/Column Read and Split-Read Transfer

The 1 cycle/column read and split-read transfers are performed for memories with a configuration cache entry selecting 1 cycle/column timing (**CT(3:0)** = ‘000x’).

An example of a 1 cycle/column read or split-read transfer is shown in Figure 17–8. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. Memory exceptions are not supported for this type of transfer.

The column accesses consist of a c1 pipeline stage during which the tap point is selected and the data transferred within the VRAM.

Figure 17–8. 1 Cycle/Column Read or Split-Read Transfer



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

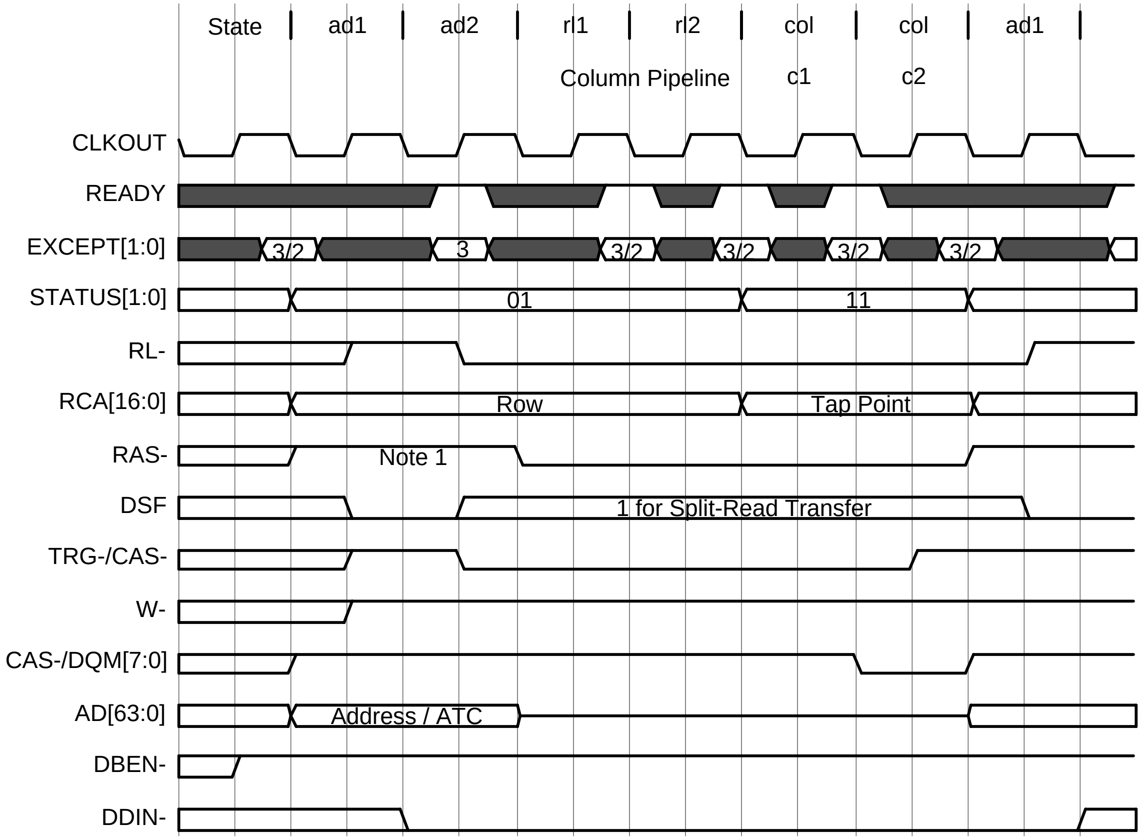
17.3.2 2 Cycle/Column Read and Split-Read Transfer

The 2 cycle/column read and split-read transfers are performed for memories with a configuration cache entry selecting 2 cycle/column timing (**CT(3:0)** = ‘0010’).

An example of a 2 cycle/column read or split-read transfer is shown in Figure 17–9. The row access consists of the ad1, ad2, rl1, and rl2 states. A single rw state may be inserted after ad2 to ensure a RAS high time of at least 3 clock cycles. Memory exceptions are not supported for this type of transfer.

The column accesses consist of c1 and c2 pipeline stages during which the tap point is selected and the data transferred within the VRAM.

Figure 17–9. 2 Cycle/Column Read or Split-Read Transfer



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

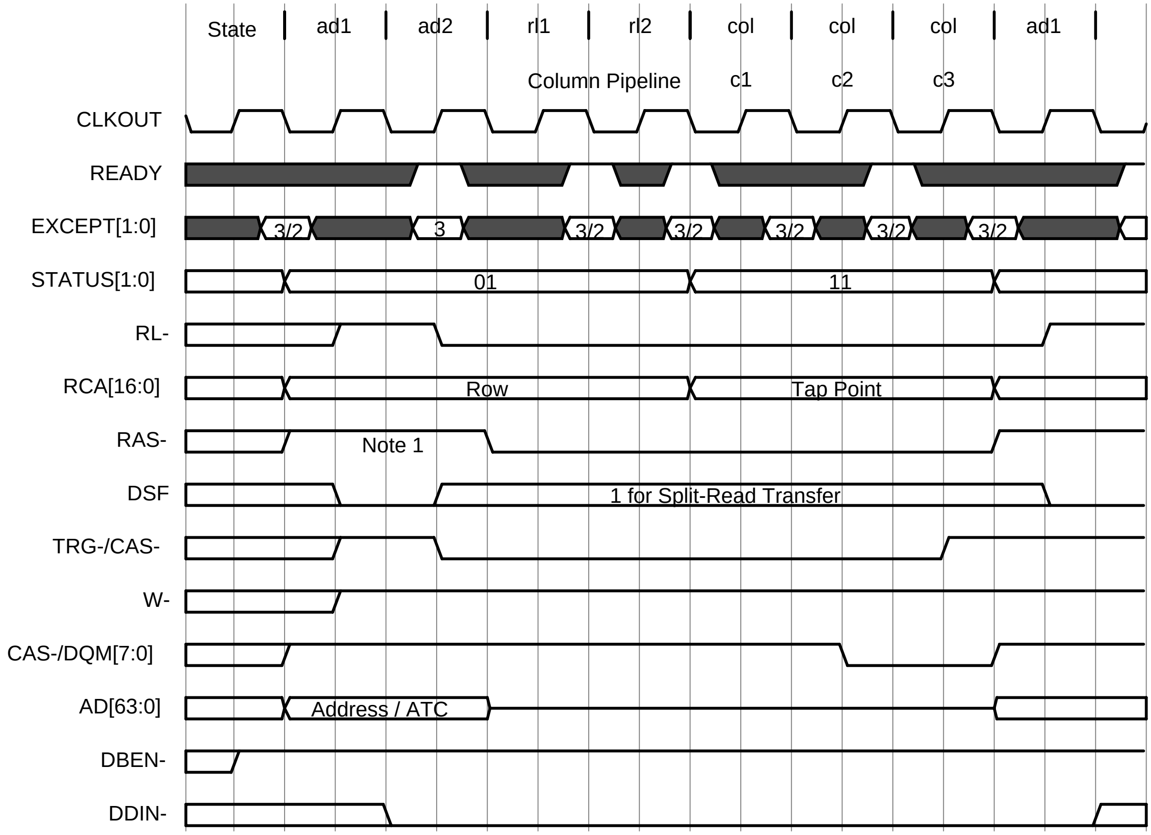
17.3.3 3 Cycle/Column Read and Split-Read Transfer

The 3 cycle/column read and split-read transfers are performed for memories with a configuration cache entry selecting 3 cycle/column timing (**CT(3:0)** = ‘`0011`’).

An example of a 3 cycle/column read or split-read transfer is shown in Figure 17–10. The row access consists of the ad1, ad2, rl1, and rl2 states. Two rw states may be inserted after ad2 to ensure a RAS high time of at least 4 clock cycles. Memory exceptions are not supported for this type of transfer.

The column accesses consist of c1, c2, and c3 pipeline stages during which the tap point is selected and the data transferred within the VRAM.

Figure 17–10.3 Cycle/Column Read or Split-Read Transfer



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS high of at least 4 cycles.

Chapter 18

SGRAM Cycles

In addition to standard SDRAM read and write cycles, the external memory interface supports the block-write feature available on synchronous graphics RAMs (SGRAMs).

Topics in this chapter include:

18.1 Special Register Set

Special Register Set (SRS) accesses are used to program the mode registers within the SGRAM to enable various features. The TMS320C82 supports only writes into the SGRAM color register. The ‘C82 SRS cycles write a value into SGRAM color registers for later use during block-write operations. SRS cycles can occur for memory banks with SDRAM timing (**CT(3:0)** = ‘1xxx’) and block-write support (**BW(1:0)** = ‘1x’). An SRS will be performed whenever the TC begins servicing a long-form packet transfer which uses the block write access mode. (See Section 8.1.6, *Color Register Loading*)

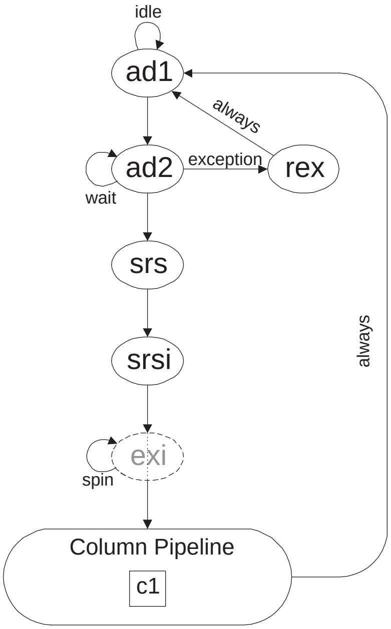
During the SRS access, the value on the SGRAM address inputs determines which special register is being written. Since only color register SRS cycles are supported by the ‘C82, the TC will always output the value to select the color register as shown in Figure 18–1. The ‘C82 supports only 64-bit buses for block-write operations so the SRS address will be output on RCA[12:3].

Figure 18–1. Special Register Set Address Output

SGRAM Address Bit	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Meaning	0	0	0	LC	LM	0	0	0	0	0
Value	0	0	0	1	0	0	0	0	0	0

Figure 18–2 shows the state sequence for an SRS access. The row access consists of the ad1, ad2, srs, srsi, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 state. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi states such that the minimum number of clock cycles between ad2 and the column pipeline increases from three to five.

Figure 18–2. Special Register Set States

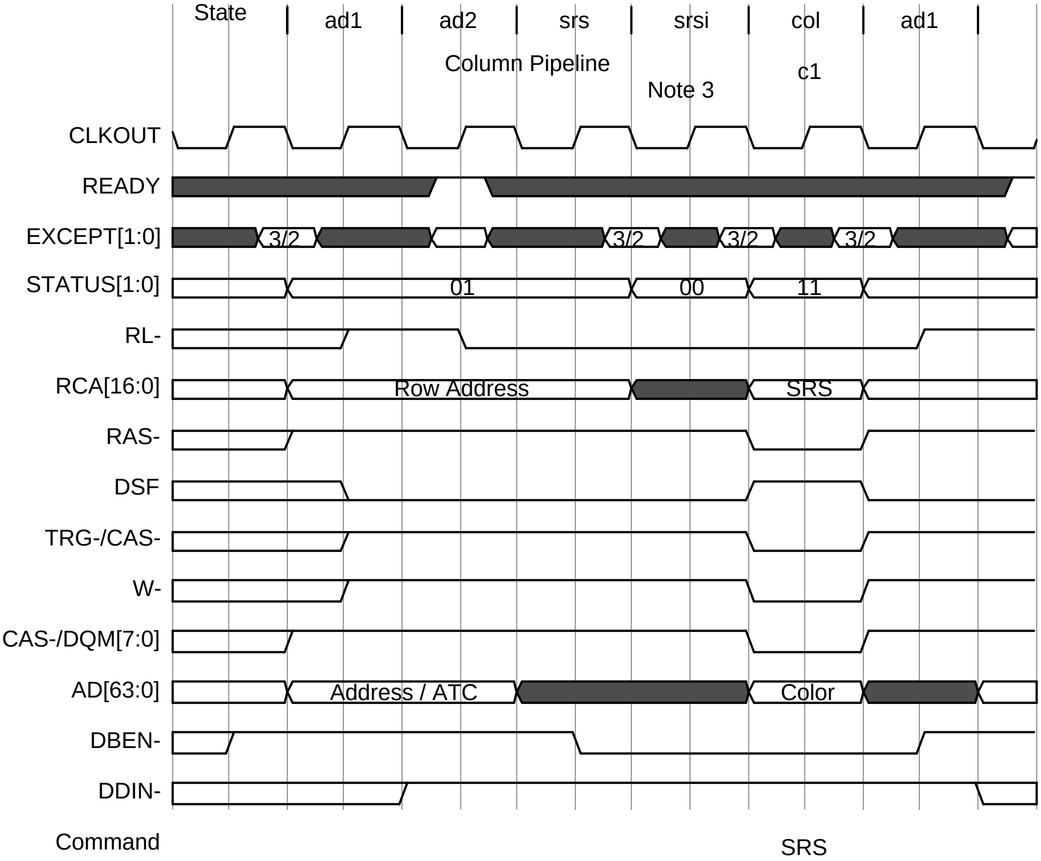


The column access consists of a single pipeline stage.

c1 The special register set address is output on RCA[12:3] and the color register value is output on AD[63:0]. All CAS/ DQM[7:0] outputs are driven low and an SRS command is issued by driving RAS, TRG/CAS, and W low and DSF high.

Once the column access is completed, the memory interface will immediately return to the ad1 state to begin the next access. An example of a color register SRS access is shown in Figure 18–3.

Figure 18–3. Special Register Set



Note 3: When exceptions are enabled, additional cycles will be inserted after srsi as required to ensure to ensure a minimum of 5 cycles between ad2 and the c1 stage.

18.2 SGRAM Block-Write Access

Block write cycles cause the data stored in the SGRAM color registers to be written to the memory locations enabled by the appropriate data bits on the AD[63:0] bus. This allows as many as 64 bytes (depending on the type of block-write being performed) to be written in a single column access.

Block-write accesses can occur for memory banks with SDRAM timing (**CT(3:0)** = ‘1xxx’) and block-write support (**BW(1:0)** = ‘1x’). Block- write accesses will be performed when the TC is servicing a long-form packet transfer which uses the block write access mode. (See Section 8.1, *Block-Write Modes*)

The timing for a block-write access is identical to that of a normal write access. However, a block-write differs from a standard write in the following signal characteristics:

- DSF is high at column time, causing block-write commands to be sent to the SGRAM.
- Block-write is supported only for 64-bit buses. All CAS/DQM[7:0] outputs will be low.
- The two or three LSBs (depending on the type of block-write) of the column address are ignored by the SGRAMs because these column locations are specified by the bits on the AD[63:0] bus.
- The values output by the ‘C82 on AD[63:0] (at column time) represent the column locations to be written using the color register value. Depending on the type of block-write supported by the SGRAM, all of the data bits may not be used by the SGRAMs.
- Block-write accesses always begin with a row access. Upon completion of the block-write, the memory interface will return to the ad1 state and will not enter column-time idle.

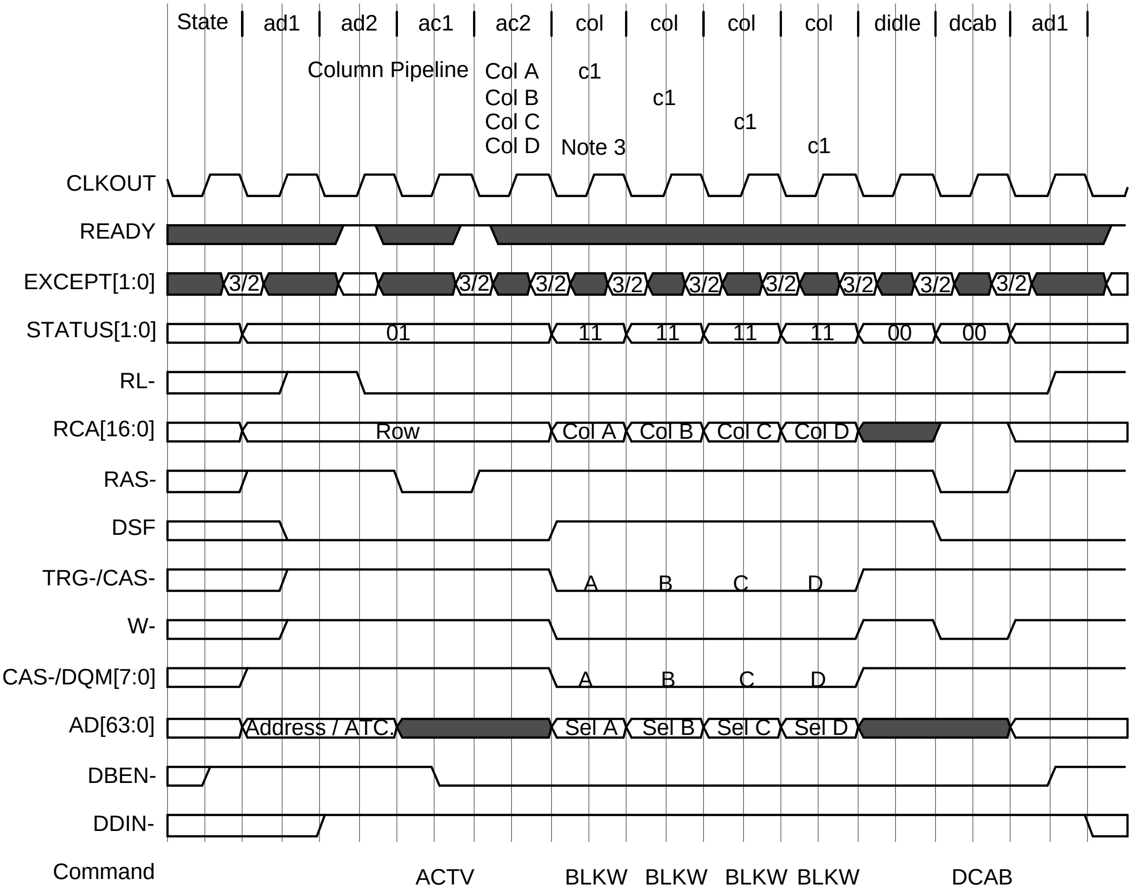
18.2.1 Burst Length 1 Block-Write Access

The burst length 1 block-write access is performed for memories with a configuration cache entry selecting burst length 1 timing (**CT(3:0)** = ‘10xx’).

An example of a burst length 1 block-write access is shown in Figure 18–4. The row access consists of the ad1, ad2, ac1, ac2, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi cycles such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

The column accesses consist of a c1 pipeline stage during which the AD[63:0] bus indicates the column locations to which the color register value is written. Following the final block-write column access, a DCAB command will always be executed to close the SGRAM row and the memory interface will return to the ad1 state.

Figure 18–4. Burst Length 1 Block-Write



Note 3: When exceptions are enabled, additional cycles will be inserted after ac2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The example shows four block writes. After block write D is completed, a DCAB command is performed to close the row. The didle state is inserted prior to the dcab to meet SGRAM timing requirements.

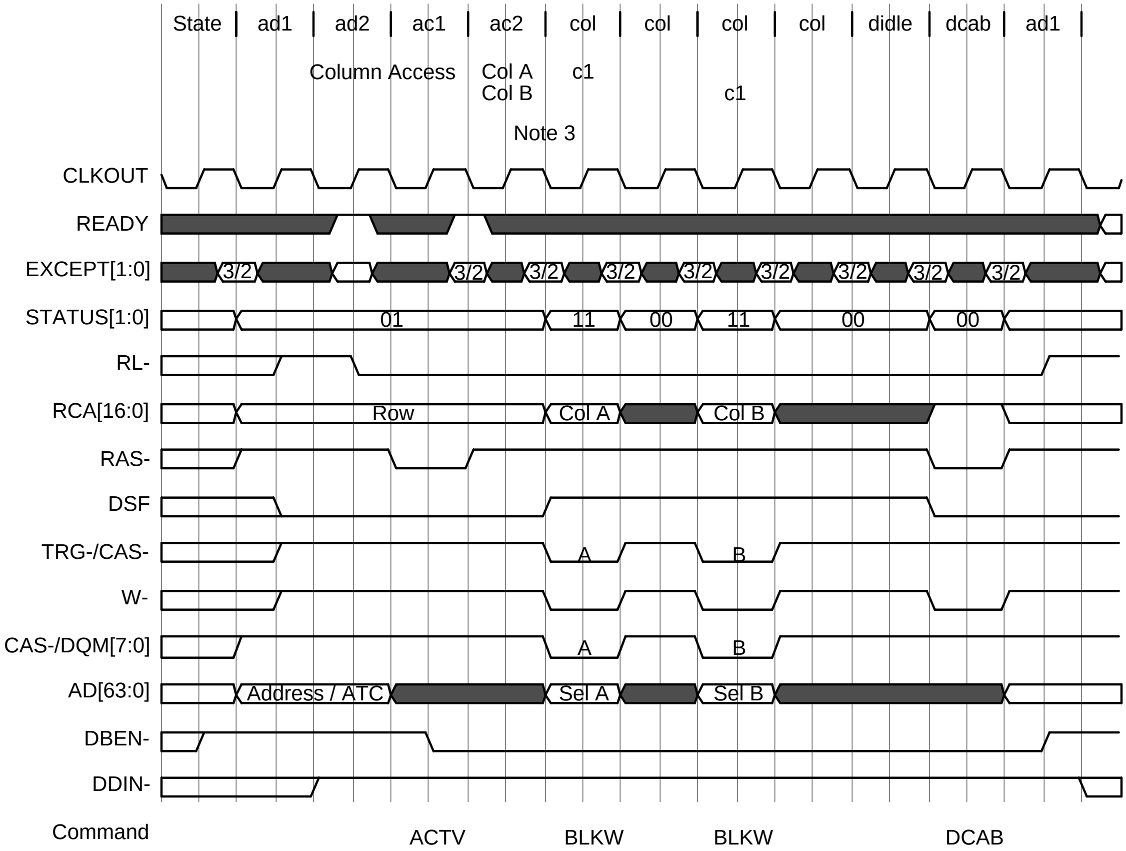
18.2.2 Burst Length 2 Block-Write Access

The burst length 2 block-write access is performed for memories with a configuration cache entry selecting burst length 2 timing (**CT(3:0)** = ‘11xx’).

An example of a burst length 2 block-write access is shown in Figure 18–5. The row access consists of the ad1, ad2, ac1, ac2, and possibly exi states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If exceptions are enabled in the configuration cache entry, the access may be aborted by an exception in the ad2 state and will return to ad1 through the rex state. Enabling exceptions will cause the addition of exi cycles such that the minimum number of cycles between ad2 and the column pipeline increases from three to five.

The column accesses consist of a c1 pipeline stage during which the AD[63:0] bus indicates the column locations to which the color register value is written. Because bursting is not supported for block write accesses for most SGRAMs, only one block write is performed for each BLKW command. However, the “2n” rule must still be satisfied so a BLKW command will be issued only on every other clock cycles during the access. Following the final block-write column access, a DCAB command will always be executed to close the SGRAM row and the memory interface will return to the ad1 state.

Figure 18–5. Burst Length 2 Block-Write



Note 3: When exceptions are enabled, additional cycles will be inserted after ac2 as required to ensure a minimum of 5 cycles between ad2 and the first c1 stage.

The example shows two block writes. After block write B is completed, a DCAB command is performed to close the row. The didle state is inserted prior to the dcab to meet SGRAM timing requirements.

Chapter 19

Refresh Controller

The TC’s refresh controller logic generates refresh requests at a programmable interval. The requests are serviced as DRAM or SDRAM refresh cycles on the external memory interface.

Topics in this chapter include:

Topic **Page**

19.1 Refresh Rate

The refresh rate is defined in terms of ‘C82 clock cycles by the value in the 16-bit **REFRATE** field of the **REFCNTL** register. The value in **REFRATE** determines the number of clock cycles that occur between the generation of refresh requests. **REFRATE** should be programmed to the largest value necessary to meet the required refresh rate of all dynamic RAM banks in the system. Programming too large a value may result in data loss. Programming to small a value will reduce external bus bandwidth. Values of less than 32 (0x20) in **REFRATE** will cause refresh requests to be disabled. **REFRATE** is loaded with a value of 32 (0x20) at reset.

An example of calculating the **REFRATE** value is shown in the equation below. The example assumes a system with one bank of DRAM and one bank of SDRAM, each requiring 4096 refresh cycles to be performed every 64 ms. A ‘C82 operating frequency of 50 MHz is used.

$$\frac{(64ms)\ \S\ (2 \times 4096)}{20ns} = 391$$

The **REFRATE** field should be programmed to a value slightly smaller than this to account for account for the fact that trickle refresh requests may not be serviced immediately due to their low priority. A value of 385 (0x181) would ensure that the refresh rate is met for both memory banks.

19.2 Refresh Priority

Each time the refresh counter reaches **REFRATE** clock cycles, a trickle refresh request is generated. Because trickle refreshes have the lowest request priority, they frequently will not be serviced immediately due to higher priority requests. For this reason, the TC keeps track of refresh requests in an internal BACKLOG register. Each time a refresh request is generated, the BACKLOG value is incremented. The BACKLOG counter is decremented every time a refresh cycle is performed.

If the BACKLOG counter reaches a value of 16 (0x10), an urgent refresh request is generated to the bus prioritization logic. When this request is serviced the TC performs a burst of refresh cycles until the BACKLOG counter reaches 12 (0x0C). If a burst refresh is interrupted by a higher-priority host request, the refresh burst will resume when the bus is returned and continue until BACKLOG decrements to 12.

A refresh cycle will only interrupt another request when an urgent refresh is required. If the external memory interface is idle, the bus prioritization logic will allow single refresh cycles to occur in order to reduce any growing backlog. These trickle refreshes will only occur if BACKLOG is nonzero.

The backlog counter is loaded with a value of 44 (0xBC) at hardware reset. The prioritization logic prevents other requests (except host requests) from being serviced until BACKLOG reaches 12. This allows the TC to perform refresh cycles for power-up initialization of DRAMs and SDRAMs. If a host request is active at reset, the power-up refresh cycles will be delayed until the host returns the bus to the TC.

19.3 Programming the Refresh Pseudo-Address

During refresh cycles, the TC outputs a 16-bit refresh pseudo-address at the start of the cycle on AD[31:16] and throughout the cycle on RCA[16:1]. This address can be decoded to separate refresh cycles to different memory banks. This is a requirement in systems needing both DRAM and SDRAM refresh cycles. In can also be done to reduce system power requirements by refreshing banks of memory separately.

The upper half of the **REFCNTL** register contains the 16-bit **RPARLD** (refresh pseudo-address reload) field. RPARLD can be programmed to select the maximum 16-bit value that will be output on AD[31:16] and RCA[16:1] during refresh cycles. This value is loaded into an internal counter (RPADEC) which is decremented by 1 each time a refresh cycle is performed. When RPADEC reaches `0x0001`, it is reloaded with the value in **RPARLD**. **RPARLD** and RPADEC are both loaded with the value `0xFFFF` following hardware reset.

19.4 Refresh Cycles

Refresh cycles differ from other types of memory cycles in that they do not make use of the cycle configuration cache. Refresh cycles are not directed to specific memory banks but are “broadcast” to the entire system. In systems with multiple refresh banks or more than one type of dynamic memory, separate decode logic is required to configure the refresh cycles and direct them to the proper bank.

19.4.1 Refresh Timing Selection

Three basic refresh timings are available; two for DRAM and one for SDRAM. The refresh timing is selected by the value on the EXCEPT[1:0] inputs at the beginning of the refresh cycle. Table 19–1 shows the refresh timing selection.

Table 19–1. Refresh Timing Selection

EXCEPT[1:0]	
0 0	DRAM Refresh (3 Cycles RAS high)
0 1	DRAM Refresh (4 Cycles RAS high)
1 0	Retry
1 1	SDRAM Refresh

The ‘00’ and ‘01’ values select DRAM CAS-before RAS refresh cycles for use with DRAM and VRAM memory banks. A choice of 3 or 4 cycles of RAS high time allows the system designer to match the ‘C82’s operating speed to the selected DRAM requirements. A ‘11’ value on the EXCEPT inputs causes the TC to generate an SDRAM refresh cycle for use with SDRAM and SGRAM memory banks. A value of ‘10’ causes the refresh cycle to be retried. (Retries for refreshes, unlike exceptions for reads or writes, cannot be disabled since the bank configuration cache is not used.)

The EXCEPT inputs may be easily generated by decoding a refresh access type code (AD[35:32] = ‘1100’) and using the lower bits of the refresh pseudo-address (AD[31:16]) to determine which memory bank to refresh and which refresh code to generate.

19.4.2 DRAM Refresh (3 Cycle RAS High)

The state sequence for a 3 cycle RAS-high refresh is shown in Figure 19–1. The access consists of the ad1, ad2, cbr, rl1, rf1, rf2, and rf3 states. Wait states may be inserted by deasserting the READY input in the ad2 or rl1 states. If a retry is requested, the access will be aborted in the ad2 state and return to ad1 through the rex state. The column pipeline is not used for refresh cycles since no data is transferred. After completing the rf3 state, the memory interface will return immediately to the ad1 state to await the next access.

Figure 19–1. DRAM Refresh States(3 Cycle RAS High)

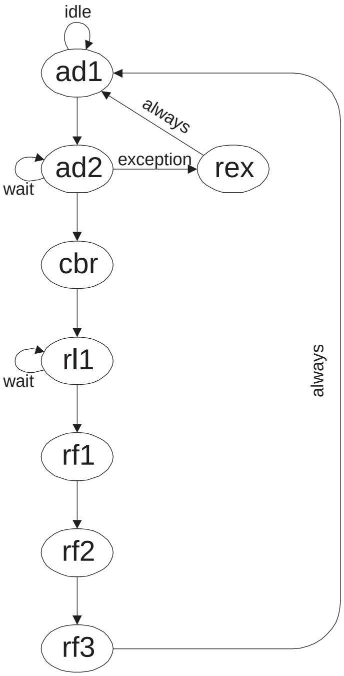
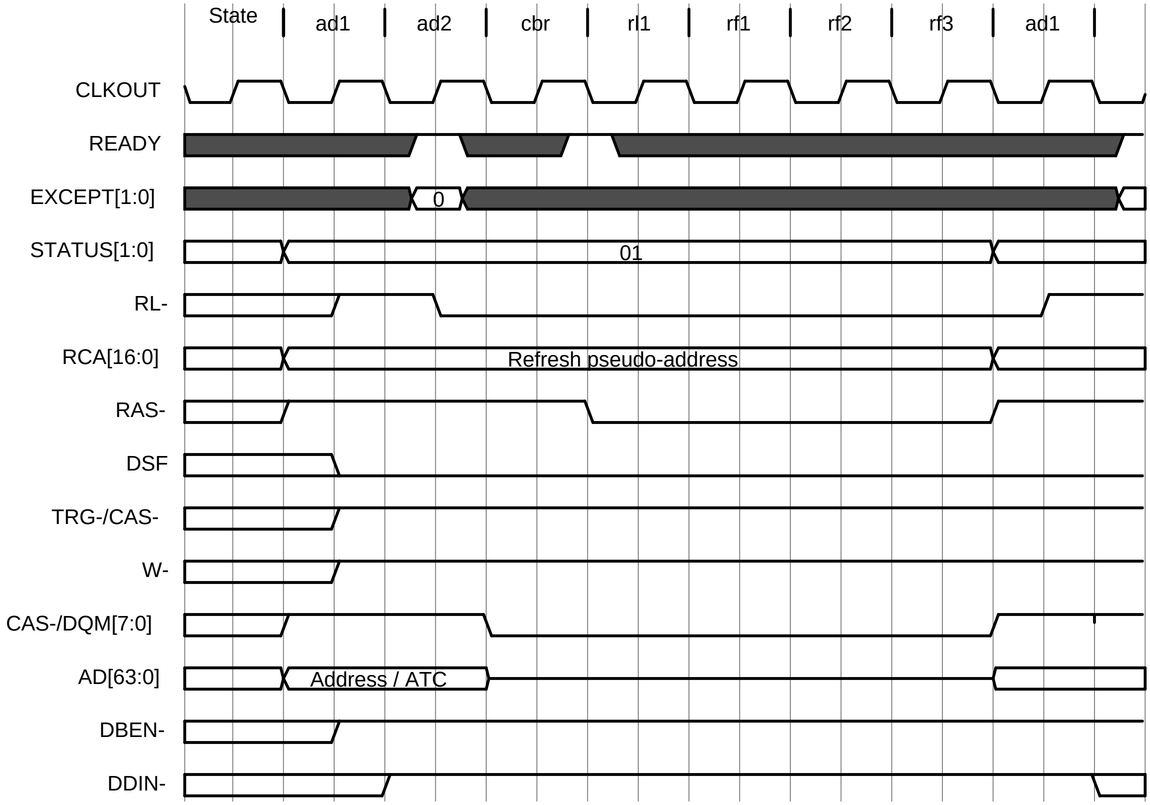


Figure 19–2 shows the timing for a 3 cycle RAS high DRAM refresh. A CAS-before-RAS refresh cycle is generated with 3 cycles of RAS high time and 4 cycles of RAS low time. The AD[31:16] signals output the refresh pseudo-address during the ad1 and ad2 states. The remaining address bits (AD[15:0]) are zeros.

Figure 19–2. DRAM Refresh (3 Cycle RAS High)



19.4.3 DRAM Refresh (4 Cycle RAS High)

The state sequence for a 4 cycle RAS-high refresh is shown in Figure 19–3. The access consists of the ad1, ad2, cbr, rl1, rf1, rf2, rf3, rf4 and possibly rw states. Wait states may be inserted by deasserting the READY input in the ad2 or rl1 states. If needed, a single rw state will be automatically inserted after cbr to ensure a RAS high time of at least 4 four clock cycles. If a retry is requested, the access will be aborted in the ad2 state and return to ad1 through the rex state. The column pipeline is not used for refresh cycles since no data is transferred. After completing the rf4 state, the memory interface will return immediately to the ad1 state to await the next access.

Figure 19–3. DRAM Refresh States(4 Cycle RAS High)

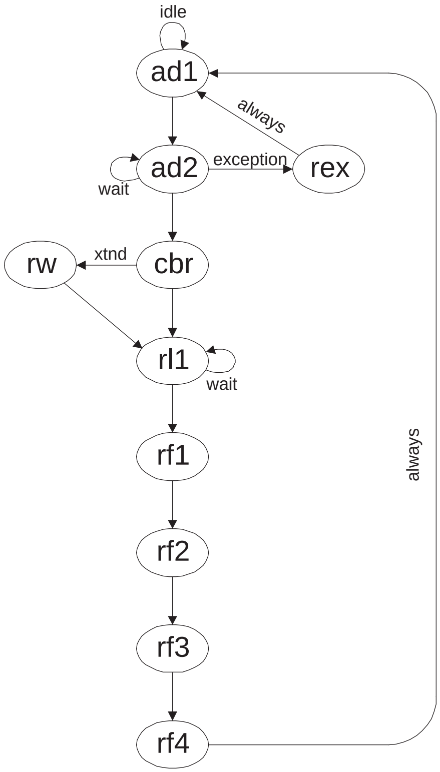
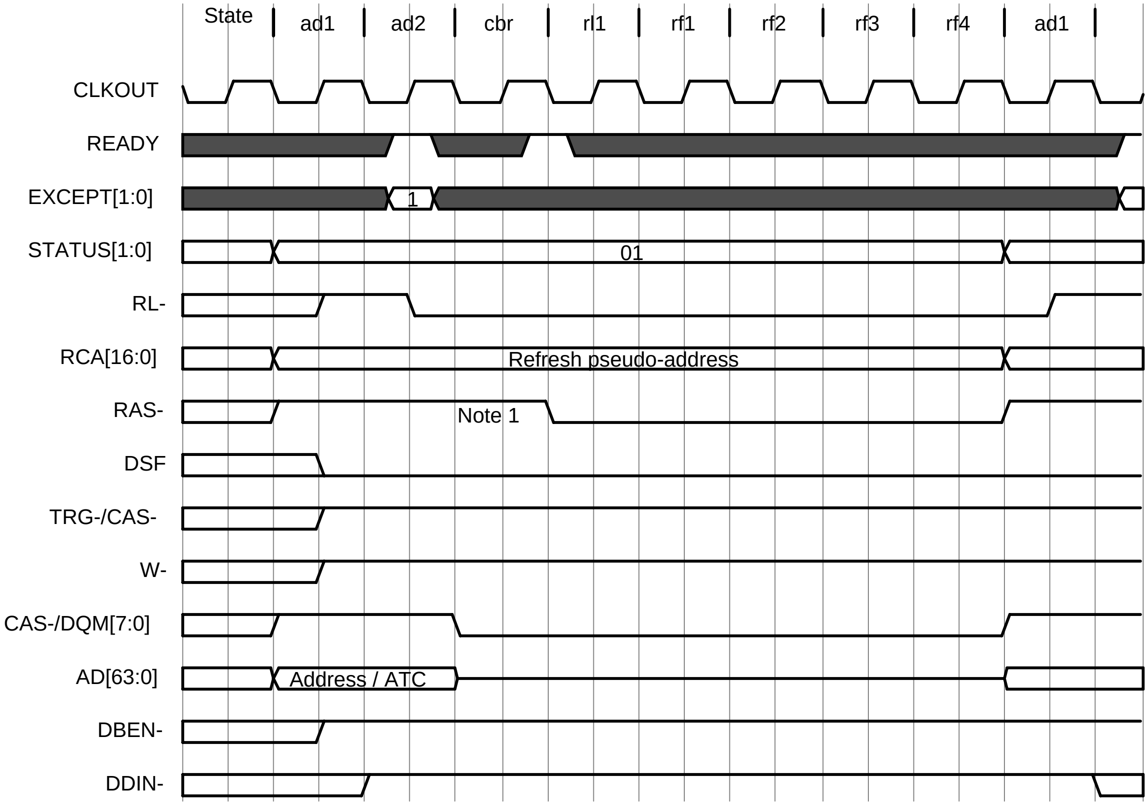


Figure 19–4 shows the timing for a 4 cycle RAS high DRAM refresh. A CAS-before-RAS refresh cycle is generated with 4 cycles of RAS high time and 5 cycles of RAS low time. The AD[31:16] signals output the refresh pseudo-address during the ad1 and ad2 states. The remaining address bits (AD[15:0]) are zeros.

Figure 19–4. DRAM Refresh (4 Cycle RAS High)



Note 1: An additional cycle will be inserted between cbr and rl1 as required to ensure a RAS- high time of at least 4 cycles

19.4.4 SDRAM Refresh

The state sequence for an SDRAM refresh is shown in Figure 19–5. The access consists of the ad1, ad2, ac1, ac2, rf1, rf2, and rf3 states. Wait states may be inserted by deasserting the READY input in the ad2 or ac2 states. If a retry is requested, the access will be aborted in the ad2 state and return to ad1 through the rex state. The column pipeline is not used for refresh cycles since no data is transferred. After completing the rf3 state, the memory interface will return immediately to the ad1 state to await the next access.

Figure 19–5. SDRAM Refresh States

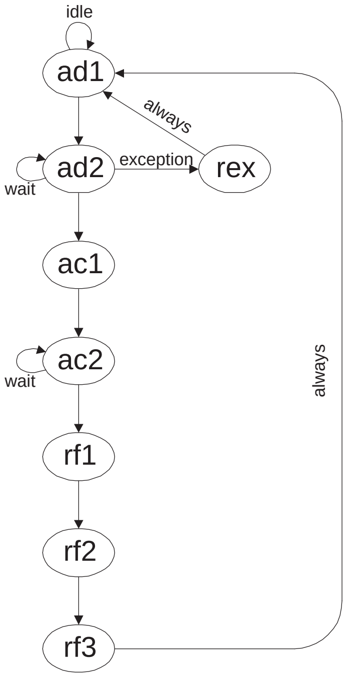
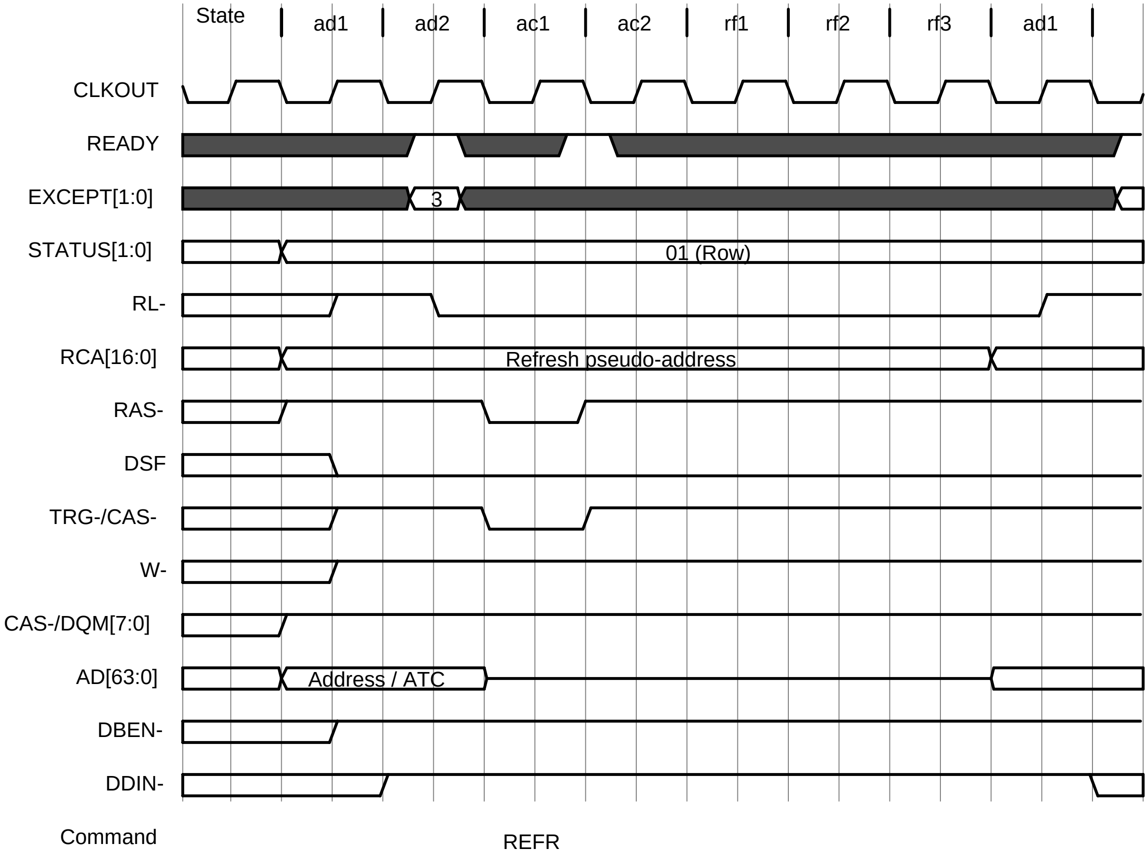


Figure 19–6 shows the timing for an SDRAM refresh. A REFR command is issued by activating both RAS and TRG/CAS in the ac1 state. A minimum of 3 clock cycles will occur between a previous DCAB command and the REFR command and a minimum of 7 clock cycles will occur between the REFR command and a subsequent REFR or ACTV command. The AD[31:16] signals output the refresh pseudo-address during the ad1 and ad2 states. The remaining address bits (AD[15:0]) are zeros.

Figure 19–6. SDRAM Refresh



Chapter 20

Host Interface

The TMS320C82's host interface is a simple handshake mechanism that allows the TC to share the external memory interface with another device.

Topics in this chapter include:

Topic **Page**

20.1 Host Interface Signals

The host interface is a simple handshake mechanism that allows another device to take control of the ‘C82’s external memory interface. The handshake mechanism uses the following signals:

HREQ **Host request input**. When another device wants control of the ‘C82 external memory interface, it drives this input low. The TC will stop driving the external memory bus at the earliest possible moment. Whenever **HREQ** is high, the TC owns and drives the bus. **HREQ** is internally synchronized the C82’s clock.

The **HREQ** input also determines whether the master processor is running or halted following a hardware reset. (See Section 21.1, *Hardware Reset*)

HACK **Host acknowledge output**. This signal is driven low by the TC to indicate that it has relinquished the bus. **HACK** is driven high (asynchronously) after **HREQ** becomes inactive and the TC resumes driving the bus.

REQ **Priority request output**. The **REQ** output is driven high to indicate that the TC has received a high priority request that requires the external memory interface. **REQ** will be driven high whenever an urgent refresh or XPT request is pending.

20.2 Host Handshake Mechanism

In order to provide maximum design flexibility, the host request is given the highest priority by the TC. This allows the system designer to determine when the host (or other device) will take or relinquish bus ownership.

When another device activates the HREQ input, the signal is synchronized to the ‘C82’s internal clock and a host request is generated to the TC. When the TC receives a host request, it immediately stops loading new accesses into its column pipeline. The TC then completes any accesses in the pipeline and places the memory interface signals in high-impedance.

The **HACK** output is driven low to indicate that the requesting device now owns the bus. When **HACK** is low, all ‘C82 memory interface outputs except **CLKOUT**, **HACK**, and **REQ** are in high-impedance. An external device must continue to drive **HREQ** low as long as it desires to control the bus. Once **HREQ** is high, **HACK** will be driven high and the TC will resume driving the memory interface outputs.

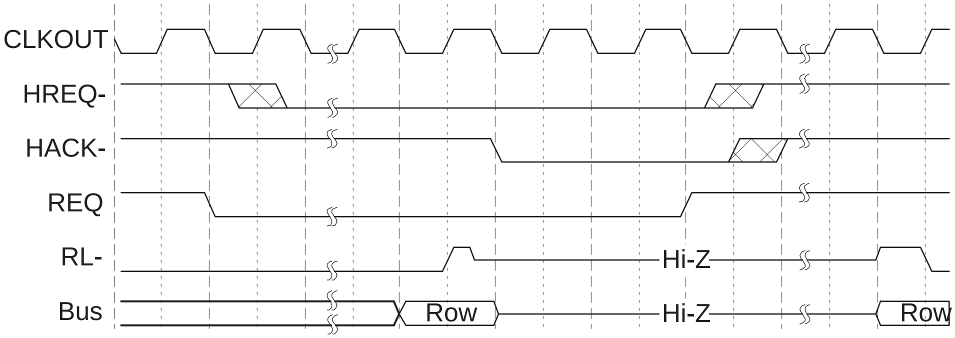
The **REQ** output can be monitored by external logic to determine when to return the bus to the TC. When **REQ** is high, it indicates that an urgent refresh or XPT request is being received by the TC. Since failure to service these requests may result in a system failure, the **REQ** output is normally used to terminate a host request (or prevent one from beginning). Note that the TC will never remove the host acknowledge of its own accord. The **REQ** output is the only indication the TC provides that an high priority transfer needs to be serviced. The **REQ** output remains low for all other requests (such as cache or DEA requests).

20.3 Host Interface Timing

Figure 20–1 shows the general timing for the host interface handshake signals. In this example a host request is initiated when the current high priority request (as indicated by **REQ** high) completes. After some amount of time (as required to complete any accesses currently in its pipeline), the TC responds to the host request be placing the memory interface signals (designated "Bus") in high impedance and driving **HACK** low.

The TC signals the presence of an urgent refresh or XPT request by driving **REQ** high and the host responds by driving **HREQ** high to release the bus. The TC immediately removes the host acknowledge signal and will then begin servicing the highest priority request.

Figure 20–1. Host Interface Timing



There are no internal pull-up resistors on any of the memory interface pins. When driven to high impedance, these outputs should be driven by the host device. Pins not driven by the host device should have external pull-up resistors installed.

Chapter 21

Resetting the Transfer Controller

The transfer controller may be reset by either hardware or software. Any reset of the TC also causes a reset of the MP and the PPs.

Topics in this chapter include:

Topic **Page**

21.1 Hardware Reset

A hardware reset of the ‘C82 occurs when the RESET input is driven low. When the reset occurs, the TC immediately (and asynchronously) places the memory interface in the high impedance state. Signals driven to high impedance are AD[63:0], RCA[16:0], CAS/DQM[7:0], STATUS[1:0], RAS, TRG/CAS, W, DSF, DBEN, and DDIN. Signals that continue to be driven are CLKOUT, HACK, and REQ.

There are no internal pullups on any of the memory interface outputs.

A hardware reset causes the TC to abort all cache, DEA and packet transfer requests. The cycle configuration cache present bits are cleared and the packet transfer timer, refresh controller, round robin token and other internal logic are reset to their default state. All TC registers are set to their default values.

21.1.1 Normal Reset Sequence

Upon completion of reset (RESET driven high), the TC will normally begin driving all high impedance signals to their inactive levels. The TC then performs at least 32 urgent refresh cycles to provide proper initialization for DRAM and SDRAM devices. If the presence of SDRAM within the system is indicated during the refresh cycles, a power-up DCAB cycle will also be performed.

After the refresh cycles are complete, will service the first MP instruction cache request. The MP begins instruction execution at its reset vector address of 0xFFFFFFF8. This memory location is contained in the cache subblock beginning at address 0xFFFFFFC0 so the TC will begin reading memory from this address. Since no present bits are set in the cycle configuration cache, this read attempt will cause a configuration cache miss and a bank configuration cycle will be performed. Upon completion of the bank configuration, the MP instruction cache fill will occur, and the MP can begin execution.

21.1.2 Reset Halt

During hardware reset, the master processor can be configured to exit reset in either a halted or running state. The reset state of the MP is selected by the value of the HREQ input on the rising edge of RESET. If HREQ is low on the RESET rising edge, the MP comes up running and immediately requests a cache fill to fetch its reset vector. (The cache fill is serviced after the TC completes the RAM initialization refresh cycles.)

If HREQ is high at the rising edge of RESET, the MP comes up in a halted state. It will not issue its first cache service request to the TC until it has been unhalted. The MP is unhalted by the first occurrence of an external interrupt on the EINT3 input. This first EINT3 serves only to unhalt the MP and no interrupt request will occur. (Other external interrupts that occur while the MP is halted will have their corresponding interrupt pending bits set as usual). Although the MP is halted, the TC is not, so it will proceed with normal RAM initialization refresh cycles and trickle refresh cycles thereafter until the MP is unhalted.

Power-up halt is useful in systems without boot ROM. It allows a host to download application code into local memory before the MP begins execution.

The hold time of HREQ after the RESET rising edge is 0. This allows the MP to be brought up running by tying HREQ to RESET.

21.1.3 Reset High-Impedance

While the HREQ input determines the state of the MP after reset, it also continues to function as the host request input. If HREQ remains low after the rising edge of RESET (or is driven low on the rising edge of RESET), a host request is initiated and the TC will respond with HACK low. Because the TC does not own the bus, it will not perform the RAM initialization refresh sequence. The TC will resume the normal reset sequence once HREQ is high.

Reset into high-impedance is useful in multi-’C82 systems where more than one device share the same memory interface. In such systems, only one device controls the bus while the others are held off by host requests. By powering up all but one device in the high-impedance state the bus contention that would occur when all devices attempted their RAM initialization sequences is avoided.

21.1.4 Endian Initialization

In addition to the operating state of the MP, the ‘C82 endian mode is also determined at the rising edge of RESET. The READY input is sampled on RESET’s rising edge. If READY is sampled low, the ‘C82 will operate in big-endian mode. If READY is sampled high, little-endian mode is selected. Once selected, the endian mode may only be changed by another hardware reset.

The sampling of READY is such that tying READY to RESET will cause READY to be sampled low, enabling big-endian operation.

21.2 Software Reset

The TC may also be reset by software. This may be done by the MP by issuing a command word to the TC with the reset command enabled. Upon receiving a software reset command, the TC will complete the accesses in its column pipeline and then terminate all cache, DEA, and packet transfer requests. The packet transfer timer logic, round robins tokens are reset to their default state, The cycle configuration cache, and refresh control logic are *not* reset. This allows the TC to maintain the current memory interface information.

After resetting itself, the TC will send reset signals to the MP and PPs causing them to be reset as well.

Appendix A

Packet Transfer Parameter Tables

This appendix shows the detailed formats for long-form 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 long-form packet transfer 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.)

Topics in this chapter include:

Topic **Page**

A.1 Long Form Packet Transfer src and dst Operating Modes

Table A–1. Long form src and dst Transfer Combinations

No	src Operation	dst 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 Long-Form Packet Transfer Parameter Table Listings

Figure A–1 through Figure A–46 show the parameter table formats for all the src and dst transfer mode combinations. The color register and transparency value location is shown for transfers which specify block-write or transparency access modes, respectively.

Figure A–1. Dimensioned src to Dimensioned dst

a) 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 / Transparency Value)			
PT+56	Don't Care			

b) 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 / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–2. Dimensioned src to Fixed-Patch, Delta-Guided dst

a) 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 No. of Entries	
PT+32	Src B Pitch		Dst B Pitch	
PT+40	Src C Pitch		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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 No. of Entries		Src C Count		PT+24
Dst B Pitch		Src B Pitch		PT+32
Dst Guide Table Ptr.		Src C Pitch		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–3. Dimensioned src to Fixed-Patch, Offset-Guided dst

a) 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 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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src C Pitch		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–4. Dimensioned src to Variable-Patch, Delta-Guided dst

a) 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	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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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
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 Ptr.		Src C Pitch		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–5. Dimensioned src to Variable-Patch, Offset-Guided dst

a) 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 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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src C Pitch		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–6. Dimensioned src with Peripheral Device Read

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Start Address		0x00000000	
PT+16	Src B Count	Src A Count	Don't Care	Don't Care
PT+24	Src C Count		Don't Care	
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			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
0x00000000		Src Start Address		PT+8
Don't Care	Don't Care	Src B Count	Src A Count	PT+16
Don't Care		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

Figure A–7. Fixed-Patch, Delta-Guided src to Dimensioned dst

a) 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 No. of Entries		Dst C Count	
PT+32	Src B Pitch		Dst B Pitch	
PT+40	Src Guide Table Ptr.		Dst C Pitch	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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 No. of Entries		PT+24
Dst B Pitch		Src B Pitch		PT+32
Dst C Pitch		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–8. Fixed-Patch, Delta-Guided src to Fixed-Patch, Delta-Guided dst

a) 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 No. of Entries		Dst No. of Entries	
PT+32	Src B Pitch		Dst B Pitch	
PT+40	Src Guide Table Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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 No. of Entries		Src No. of Entries		PT+24
Dst B Pitch		Src B Pitch		PT+32
Dst Guide Table Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–9. Fixed-Patch, Delta-Guided src to Fixed-Patch, Offset-Guided dst

a) 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 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 Tble Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–10. Fixed-Patch, Delta-Guided src to Variable-Patch, Delta-Guided dst

a) 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	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–11. Fixed-Patch, Delta-Guided src to Variable-Patch, Offset-Guided dst

a) 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 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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–12. Fixed-Patch, Delta-Guided src with Peripheral Device Read

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Start Address		0x00000000	
PT+16	Src B Count	Src A Count	Don't Care	Don't Care
PT+24	Src No. of Entries		Don't Care	
PT+32	Src B Pitch		Don't Care	
PT+40	Src Guide Table Ptr.		Don't Care	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
0x00000000		Src Start Address		PT+8
Don't Care	Don't Care	Src B Count	Src A Count	PT+16
Don't Care		Src No. of Entries		PT+24
Don't Care		Src B Pitch		PT+32
Don't Care		Src Guide Table Ptr.		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–13. Fixed-Patch, Offset-Guided src to Dimensioned dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst C Pitch	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–14. Fixed-Patch, Offset-Guided src to Fixed-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–15. Fixed-Patch, Offset-Guided src to Fixed-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Tble Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–16. Fixed-Patch, Offset-Guided src to Variable-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–17. Fixed-Patch, Offset-Guided src to Variable-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–18. Fixed-Patch, Offset-Guided src with Peripheral Device Read

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Base Address		0x00000000	
PT+16	Src B Count	Src A Count	Don't Care	Don't Care
PT+24	Src No. of Entries		Don't Care	
PT+32	Src B Pitch		Don't Care	
PT+40	Src Guide Table Ptr.		Don't Care	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
0x00000000		Src Base Address		PT+8
Don't Care	Don't Care	Src B Count	Src A Count	PT+16
Don't Care		Src No. of Entries		PT+24
Don't Care		Src B Pitch		PT+32
Don't Care		Src Guide Table Ptr.		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–19. Fixed-Patch, Offset-Guided, LUT src to Dimensioned dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst C Pitch	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–20. Fixed-Patch, Offset-Guided, LUT src to Fixed-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–21. Fixed-Patch, Offset-Guided, LUT src to Fixed-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Tble Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–22. Fixed-Patch, Offset-Guided, LUT src to Variable-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–23. Fixed-Patch, Offset-Guided, LUT src to Variable-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–24. Fixed-Patch, Offset-Guided, LUT src with Peripheral Device Read

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Base Address		0x00000000	
PT+16	Src B Count	Left Shift	Don't Care	Don't Care
PT+24	Src No. of Entries		Don't Care	
PT+32	Src B Pitch		Don't Care	
PT+40	Src Guide Table Ptr.		Don't Care	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
0x00000000		Src Base Address		PT+8
Don't Care	Don't Care	Src B Count	Left Shift	PT+16
Don't Care		Src No. of Entries		PT+24
Don't Care		Src B Pitch		PT+32
Don't Care		Src Guide Table Ptr.		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–25. Variable-Patch, Delta-Guided src to Dimensioned dst

a) 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	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 Ptr.		Dst C Pitch	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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	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 Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–26. Variable-Patch, Delta-Guided src to Fixed-Patch, Delta-Guided dst

a) 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	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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	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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–27. Variable-Patch, Delta-Guided src to Fixed-Patch, Offset-Guided dst

a) 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 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 Tble Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Base 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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–28. Variable-Patch, Delta-Guided src to Variable-Patch, Delta-Guided dst

a) 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	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–29. Variable-Patch, Delta-Guided src to Variable-Patch, Offset-Guided dst

a) 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 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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Base Address		Src Start 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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–30. Variable-Patch, Delta-Guided src with Peripheral Device Read

a) Big-Endian Format

	Word 0		Word 1	
Byte Address	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Start Address		0x00000000	
PT+16	Don't Care	Don't Care	Don't Care	Don't Care
PT+24	Src No. of Entries		Don't Care	
PT+32	Src B Pitch		Don't Care	
PT+40	Src Guide Table Ptr.		Don't Care	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		
63	32	31	0	Byte Address
PT Options		Next Entry Address		PT
0x00000000		Src Start Address		PT+8
Don't Care	Don't Care	Don't Care	Don't Care	PT+16
Don't Care		Src No. of Entries		PT+24
Don't Care		Src B Pitch		PT+32
Don't Care		Src Guide Table Ptr.		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–31. Variable-Patch, Offset-Guided src to Dimensioned dst

a) Big-Endian Format

	Word 0		Word 1	
Byte Address	63	32	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 Ptr.		Dst C Pitch	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		
63	32	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 Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–32. Variable-Patch, Offset-Guided src to Fixed-Patch, Delta-Guided dst

a) Big-Endian Format

	Word 0		Word 1	
Byte Address	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		
63	32	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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–33. Variable-Patch, Offset-Guided src to Fixed-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Tble Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–34. Variable-Patch, Offset-Guided src to Variable-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–35. Variable-Patch, Offset-Guided src to Variable-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	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 Ptr.		Dst Guide Table Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Base 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 Ptr.		Src Guide Table Ptr.		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–36. Variable-Patch, Offset-Guided src with Peripheral Device Read

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	Src Base Address		0x00000000	
PT+16	Don't Care	Don't Care	Don't Care	Don't Care
PT+24	Src No. of Entries		Don't Care	
PT+32	Src B Pitch		Don't Care	
PT+40	Src Guide Table Ptr.		Don't Care	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
0x00000000		Src Base Address		PT+8
Don't Care	Don't Care	Don't Care	Don't Care	PT+16
Don't Care		Src No. of Entries		PT+24
Don't Care		Src B Pitch		PT+32
Don't Care		Src Guide Table Ptr.		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–37. Fill with Value src to Dimensioned dst

a) 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	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			

b) 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	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

Figure A–38. Fill with Value src to Fixed-Patch, Delta-Guided dst

a) 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	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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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	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 Ptr.		Fill Value MS Word		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–39. Fill with Value src to Fixed-Patch, Offset-Guided dst

a) 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 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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Fill Value MS Word		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–40. Fill with Value src to Variable-Patch, Delta-Guided dst

a) 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	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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) 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
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 Ptr.		Fill Value MS Word		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–41. Fill with Value src to Variable-Patch, Offset-Guided dst

a) 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 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 Ptr.	
PT+48	(Color Register / Transparency Value)			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
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 Ptr.		Fill Value MS Word		PT+40
(Color Register / Transparency Value)				PT+48
Don't Care				PT+56

Figure A–42. Peripheral Device Write to Dimensioned dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	0x00000000		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	Don't Care		Dst B Pitch	
PT+40	Don't Care		Dst C Pitch	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Start Address		0x00000000		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		Don't Care		PT+32
Dst C Pitch		Don't Care		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–43. Peripheral Device Write to Fixed-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	0x00000000		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	Don't Care		Dst B Pitch	
PT+40	Don't Care		Dst Guide Table Ptr.	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Start Address		0x00000000		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		Don't Care		PT+32
Dst Guide Table Ptr.		Don't Care		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–44. Peripheral Device Write to Fixed-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	0x00000000		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	Don't Care		Dst B Pitch	
PT+40	Don't Care		Dst Guide Table Ptr.	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Base Address		0x00000000		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		Don't Care		PT+32
Dst Guide Table Ptr.		Don't Care		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–45. Peripheral Device Write to Variable-Patch, Delta-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	0x00000000		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	Don't Care		Dst B Pitch	
PT+40	Don't Care		Dst Guide Table Ptr.	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Start Address		0x00000000		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		Don't Care		PT+32
Dst Guide Table Ptr.		Don't Care		PT+40
Don't Care				PT+48
Don't Care				PT+56

Figure A–46. Peripheral Device Write to Variable-Patch, Offset-Guided dst

a) Big-Endian Format

Byte Address	Word 0		Word 1	
	63	32	31	0
PT	Next Entry Address		PT Options	
PT+8	0x00000000		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	Don't Care		Dst B Pitch	
PT+40	Don't Care		Dst Guide Table Ptr.	
PT+48	Don't Care			
PT+56	Don't Care			

b) Little-Endian Format

Word 1		Word 0		Byte Address
63	32	31	0	
PT Options		Next Entry Address		PT
Dst Base Address		0x00000000		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		Don't Care		PT+32
Dst Guide Table Ptr.		Don't Care		PT+40
Don't Care				PT+48
Don't Care				PT+56