EK-MBOX-UD-004

MBOX
STORAGE CONTROLLER
UNIT DESCRIPTION

digital equipment corporation • marlborough, massachusetts

1st Edition, June 1975
2nd Edition (Rev), January 1976
3rd Edition (Rev), September 1976
4th Edition (Rev), May 1977

The material in this manual is for informational purposes and is subject to change without notice. Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

ILLUSTRATIONS (Cont)

TABLES

TABLES (Cont)

PREFACE

The MBox Technical Description contains three levels (sections) of descriptions as do all other unit descriptions. The three levels are:

1. Overview
2. Functional Description
3. Logic Descriptions

The Overview section identifies and introduces the major elements of the MBox and provides a brief description of their individual functions and how they operate collectively to execute the primary MBox functions which are to service EBox and CBus requests.

The Functional Description section describes the primary MBox functions. To describe these functions, an orderly functional presentation with appropriate introductory and support material, is provided. The level of detail in this section is limited to a functional perspective; it does not provide specific details.

The Logic Description section contains a detailed logic description of the basic elements introduced in the Overview. These functional elements are further described in the primary functional context in the Functional Description section. The Logic Description section is the most comprehensive part of the MBox Technical Description because not only are the basic elements of the MBox described in detail, they are described in the context of how they execute the primary MBox functions. In addition, this section provides a direct index into the logic print set and wire lists through the use of print prefixes.

SECTION 1
OVERVIEW

1.1 INTRODUCTION

This section contains an overview of the MBox. The MBox is the storage controller of the KL10 processor (Figure 1-1). Each functional element in the MBox is introduced in this section. The functional elements are:

a. Pager
b. Physical Memory Address selector (PMA)
c. Data Cache and Use Logic
d. Memory Buffers (MBs)
e. Channel I/O Processor (channel controller)
f. Several Autonomous Controls (Cache/Core/MB/CCA Control)

Besides the functional elements, this section also introduces some of the operational concepts unique to these elements.

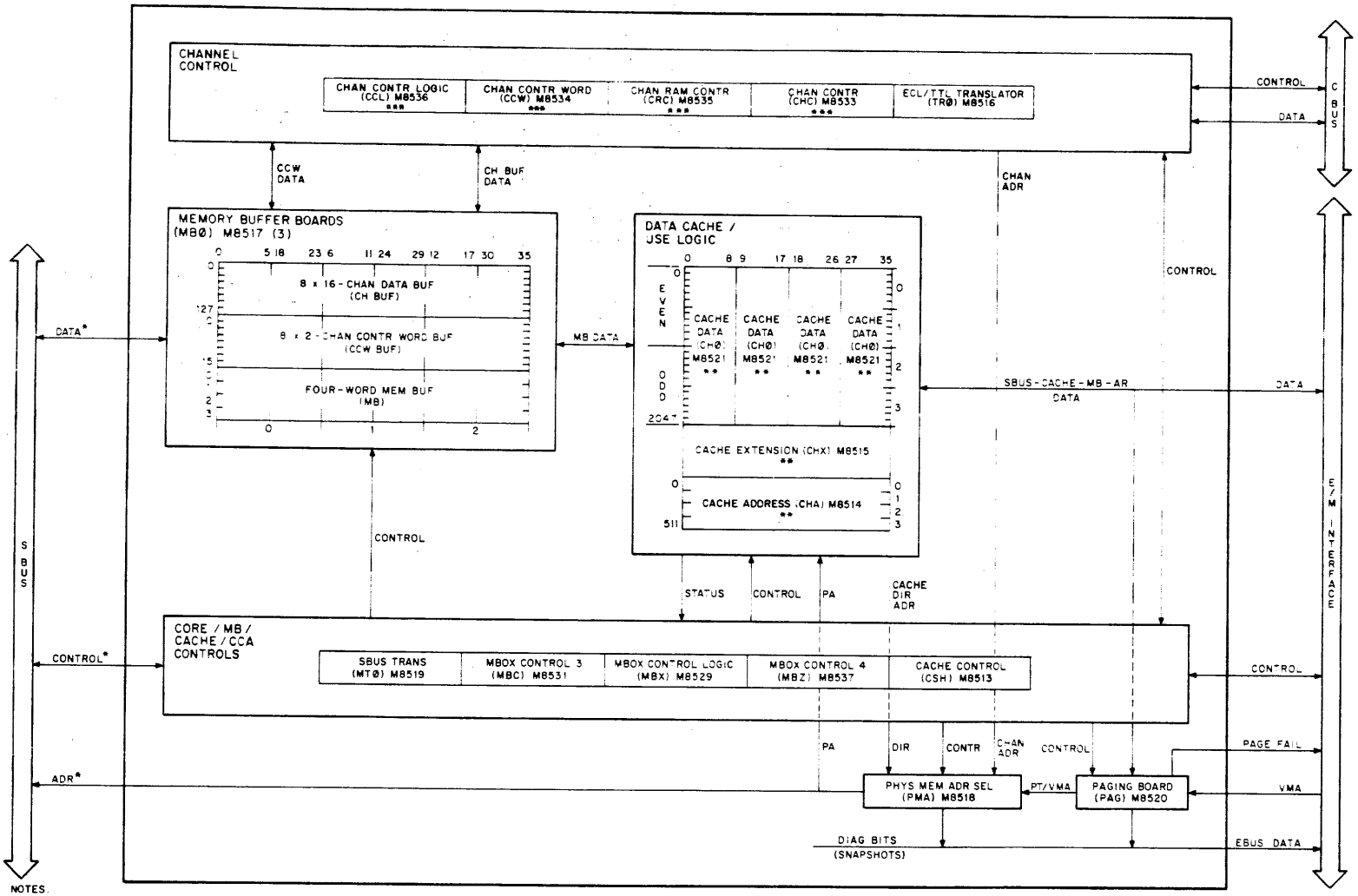
The pager, the PMA, the optional four-segment data cache, and the four MB registers provide the EBox instruction execution unit access to physical memory. The physical memory address is formed by the Pager and the PMA, while the data path between main memory and the EBox is created by the MBs and the cache.

The MBox can also be equipped with an integral data channel I/O processor (a multiplexed channel controller). This I/O processor interfaces with the Cache and the MBs to form a data path from the physical memory Storage Bus (SBus) to the Channel Bus (CBus). The CBus is multiplexed by the channel I/O processor to orderly select up to eight Massbus controllers (channels). The channel I/O processor interacts with the Cache to maintain the integrity of the data flow between physical memory and mass storage.

There are several versions of the MBox: for example, one version is implemented in DECsystem-1080; another is implemented in DECsystem-2040. The MBox implemented in DECsystem-1080 contains a cache but does not contain an integral channel I/O processor; the MBox implemented in DECsystem-2040 contains the integral channel I/O processor but does not contain a cache. In both cases, the interface signals for the functional element that is not implemented are terminated in substitute terminator boards. Table 1-1 summarizes four variations. The module designator, name, mnemonic, quantity, and used on code are specified.

Besides the four MBox variations, two models of the CPU (EBox and MBox) have been released. They are designated KL10-PA and KL10-PV CPU. The module complements that compose the MBox for both the KL10-PA and KL10-PV CPUs are also identified in Table 1-1. Except for some minor changes to facilitate a higher operating clock (MBox clock), the MBox is identical for both CPU models. The MBox clock for the KL10-PA CPU is 25 MHz while the clock for the KL10-PV CPU is 30 MHz.

When reading this text to gain an understanding of an MBox without an integral channel I/O processor (channel control), as implemented in the DECsystem-1080, simply ignore any reference to the channel control, CBus requests to the channel control, and channel requests to the cache/core control. Although the CH BUF and CCW BUF remain on the MB boards, the four channel control boards are not included; instead, the signals that would interface with these boards are terminated on substitution boards.



NOTES:

* These signals are routed thru SBus Translator Board M8519

** These boards are replaced by Cache Substitution Boards if the Cache is not implemented.

*** These boards are replaced by Channel Substitution Boards if the channels are not implemented

10-2482

Figure 1-1 MBox Simplified Block Diagram

Table 1-1 MBox Module Complement

Designation		Name	Mnemonic	Quantity	Used On			
KL10-PA	KL10-PV				1080	1090	2040	2050
M8513	M8513-YA	Cache Control	CSH	1	X	X	X	X
M8514	M8514	Cache Address	CHA	1	X	X		X
M8515	M8515	Cache Extension	CHX	1	X	X		X
M8516	M8516	ECL/TTL Translator (EBus and CBus)	TR0	3	X	X	X	X
M8517	M8517	Memory Buffer	MB0	3	X	X	X	X
M8518	M8518-YA	Physical Memory Address Buffer	PMA	1	X	X	X	X
M8519	M8519	Internal Mem Bus Translator (SBUS)	MT0	2	X	X	X	X
M8520	M8520-YA	Paging Board	PAG	1	X	X	X	X
M8521	M8521	Cache Data	CH0	4	X	X		X
M8529	M8529-YA	MBox Control Logic	MBX	1	X	X	X	X
M8531	M8531-YA	MBox Control 3	MBC	1	X	X	X	X
M8533	M8533	Channel Control	CHC	1		X	X	X
M8534	M8534	Channel Control Word	CCW	1		X	X	X
M8535	M8535	Channel RAM Control	CRC	1		X	X	X
M8536	M8536	Channel Control Logic	CCL	1		X	X	X
M8537	M8537	MBox Control 4	MBZ	1	X	X	X	X
M8549-YA	M8549-YA	Channel Control Substitute	CHCS	1	X			
M8549-YB	M8549-YA	Channel Control Word Substitute	CCWS	1	X			
M8549-YC	M8549-YA	Channel RAM Control Substitute	CRCS	1	X			

Table 1-1 MBox Module Complement (Cont)

Designation		Name	Mnemonic	Quantity	Used On			
KL10-PA	KL10-PV				1080	1090	2040	2050
M8549-YD	M8549-YD	Channel Control Logic Substitute	CCLS	1	X			
M8549-YE	M8549-YE	Cache Address Substitute	CHAS	1			X	
M8549-YF	M8549-YF	Cache Extension Substitute	CHXS	1			X	
M8549-YH	M8549-YH	Cache Data Substitute	CDOS	4			X	

When reading this text to gain an understanding of an MBox without a cache, as implemented in the DECsystem-2040, simply disregard any reference to checking the cache in the cache control decision path and any reference to MB and Cache Clearer (CCA) requests. Even though the cache is not implemented (six boards which include four data boards, one cache address board, and one cache extension board), the cache control logic, which is contained on three boards, remains and memory read/write requests are executed as if the EBox issued a request to bypass the cache; that is, one-word read/write operations will be executed. The cache control signals that would interface with the six cache boards are terminated in substitution boards.

The pager is a high-speed, 512-word, set-associative automatic buffer memory where physical page addresses and page descriptor keys are stored. It serves as a high-speed extension of the page table portions of the user and executive process tables (UPTs and EPT) (KI paging) or the page table pointed to by entries in the UPTs and EPT (KL paging). When the EBox issues a request for paged memory, the MBox automatically checks the contents of the pager to see if it contains a valid physical page address. If there is a valid address, it simply concatenates the entry with the low-order nine bits of the virtual address (Q-WORD and WORD No.). This address is then used to look in the cache and, if necessary, issue a core request. If the pager does not contain a valid physical page address for KI paging, the MBox automatically issues a core read cycle to refill the hardware page table from the UPT or EPT. Since four words are typically fetched at a time and since the process table contains two physical page address entries per word, eight page table entries will be fetched and moved to the Pager at a time. Consequently, a page refill cycle will be required only when the program addresses pass through the boundary of every eighth page. For KL paging, the EBox executes the page refill operation.

The cache is a high-speed, 2048-word, multiple set-associative automatic data buffer memory where instructions and data are stored and maintained as the EBox issues requests for memory. It serves a high-speed extension of core memory. When the EBox issues a memory request, the MBox fetches a 4-word block (quadword) from core, transfers the requested word to the EBox, and stores the words in the cache (refills the cache). Once instructions and data have been moved from core to the cache, the EBox can fetch instructions and operands much faster via the cache on subsequent references, since a time-consuming core cycle will not have to be executed. By fetching 4-word blocks instead of single words from memory, and due to the principle that the program may need the next sequential word or words in the program, results in what is referred to as having the ability to "look ahead." Another characteristic of programs is to execute the same instructions many times as in iterative loops. In this situation, the cache is particularly effective because once the instructions and operands are resident in the cache, further references to core will not be required in executing the code comprising the loops.

For write operations, the MBox writes the word directly into the cache instead of core. Write operations to core are initiated only when core needs to be updated. This feature has the effect of conserving core cycles while a user program is running.

The channel I/O processor is a multiplexed channel controller that can handle up to eight simultaneous high-speed block transfers without program intervention. After being started by a Massbus controller, the channel I/O processor executes the block transfer under the control of a channel command list that is stored in physical memory. The channel I/O processor employs a set of random access memories (RAMs) for storing control and status bits, maintaining the channel command list pointer (CLP) and the channel command word (CCW), and buffering the data.

Besides the functional elements introduced above, the MBox contains several autonomous control elements to execute operations and maintain order. The controls are:

a. Cache Control
b. Channel Control
c. MB Control
d. Core Control
e. Cache Clearer Control

These controls operate autonomously in that each can run independent of the other until the requested operation is completed. Requests are issued by the EBox, the CBus, or by the controls themselves. This control structure has the effect of compressing time in that several operations can be going on at the same time.

On a priority basis, the MBox grants and executes all memory requests made by the EBox and up to eight high-speed multiplexed data channels. The MBox will execute a request whenever the request is made, unless it is busy executing a previous request. Once a request is granted, the MBox can remain busy for a number of clock ticks. To ensure the channels adequate service, the EBox is prevented from getting the next core memory cycle if a channel has requested service in the meantime. If channel requests are backed up, the channels will continue to get the available core cycles. Although it is not considered to be its main function, the cache also affords the channels more available core cycles than would otherwise be possible.

The cache is included in the MBox to provide the EBox with a high-speed buffer memory for instructions and operands (Figures 1-2 and 1-3). The access speed of this memory is a function of the machine clock (160 ns at 25 MHz and 133 ns at 30 MHz). As the EBox makes requests for instructions and operands, memory cycles are granted by the MBox and the cache is filled up four words at a time. Data is transferred from core to the cache via the four MBs. Considering that it is very likely that the EBox will request the next consecutive word in a string, the word will already be in the cache and, therefore, will be available to the EBox sooner, since it will come from the cache rather than from core. When the EBox makes a request for a word that is not already in the cache, the MBox will grant another core cycle to place four more words in the cache. To identify each quadword group, the cache contains a directory that stores the physical page number of the quadword (ADR). The directory also contains locations for the purpose of identifying which words are valid (VAL bit) and which words were written by the EBox (WR bits). As the cache is filled with instructions and operands, the associated locations of the directory are updated to specify the physical page address (ADR) of the quadword and to specify which words were fetched from core (VAL bits). Words that have been written into the cache by the EBox are identified by updating directory address, VAL bits, and WR bits, accordingly, so that they can be moved back to core before they are supplanted.

NOTE

If the cache is not implemented in the MBox, EBox requests are serviced by transferring a single word to/from core memory.

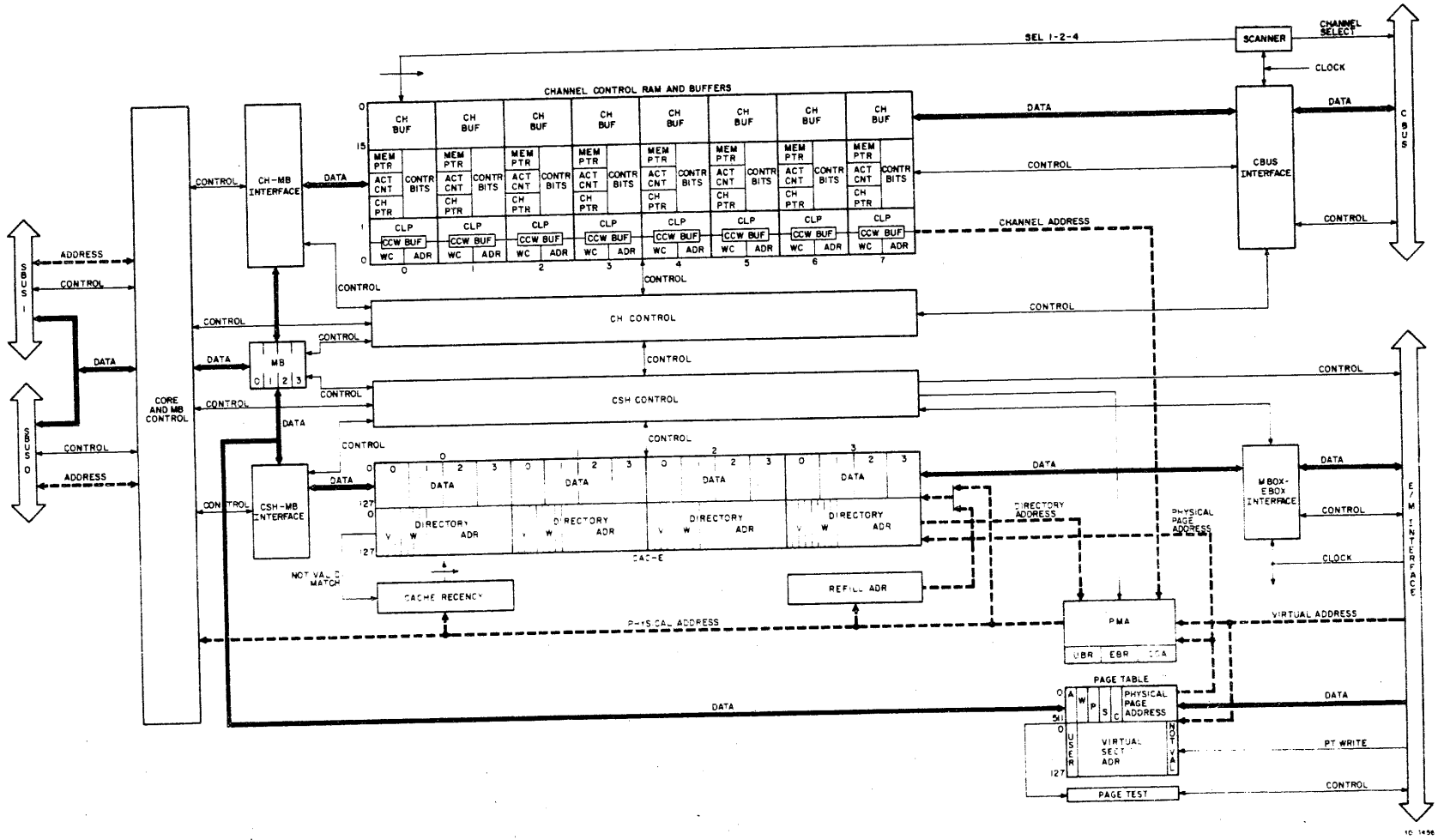


Figure 1-2 MBox RAM Structures, Interfaces and Controls, Block Diagram

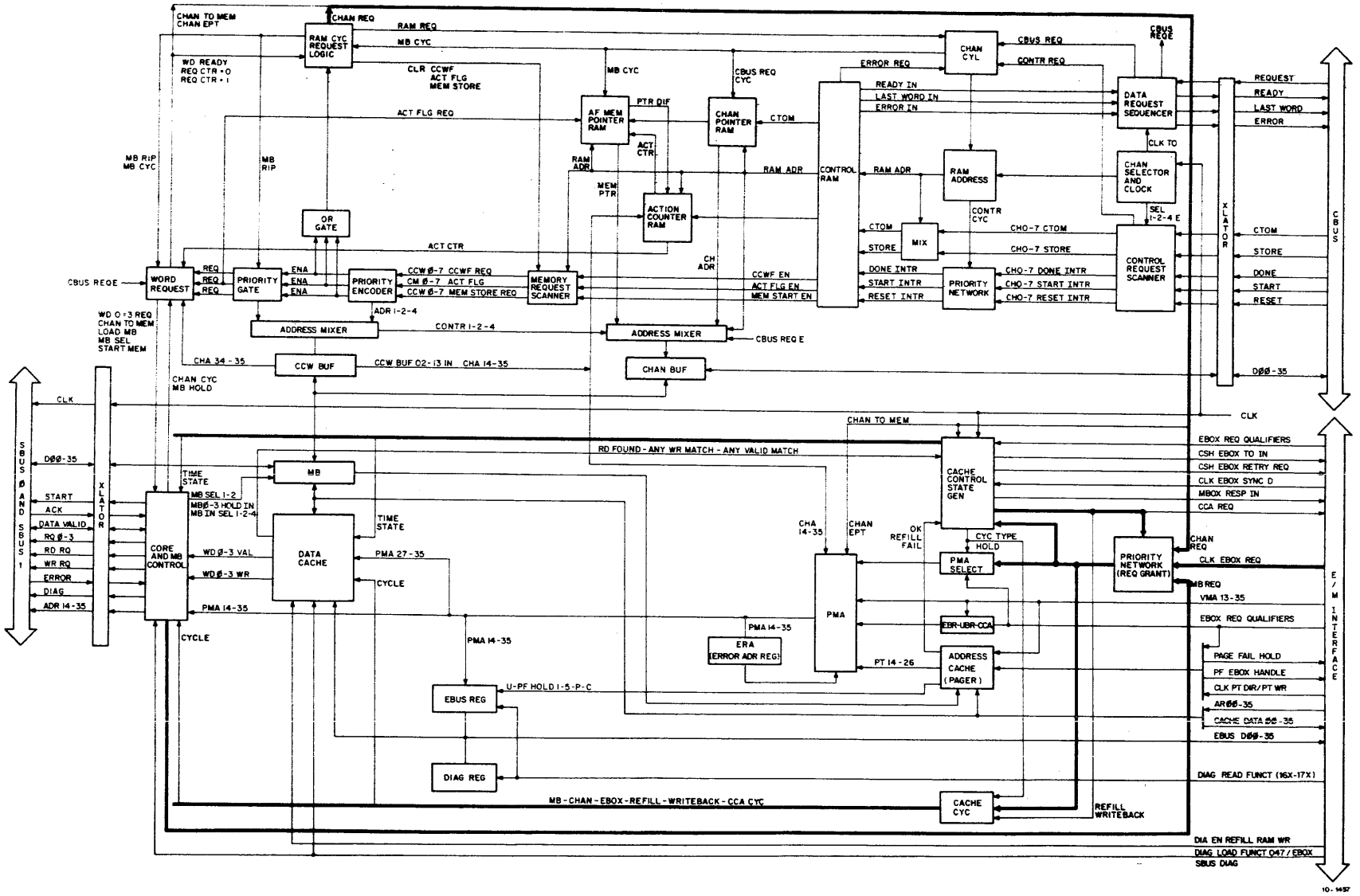


Figure 1-3 MBox Functional Block Diagram

Channels are granted core cycles if core is not busy or after a core cycle that is started by the EBox is done. The EBox can get core cycles only if the channels do not have a request pending. This feature is incorporated into the MBox to minimize potential data overruns on channel transfers. Channel data is accumulated by 16-word CH buffers. Each channel has the use of such a buffer to smooth the transfer of data between the CBus and the MBs. Only 15 locations in each CH buffer are used. This is because a 4-bit code is used to keep track of the buffer contents. On channel writes (controller to memory), four words may have to be accumulated before a core write cycle can be requested; on channel reads (memory to controller), four empty locations may be needed before a core read cycle can be requested. As words are moved in and out of the CH buffers, the number of words remaining for channel writes and the number of empty locations remaining for channel reads can be computed by comparing the channel pointer (CH PTR) with the memory pointer (MEM PTR). The CH PTR is advanced every time a word is moved via the CBus into or out of the CH buffer. The MEM PTR is advanced every time a group of words (maximum of four) is moved into or out of the CH buffer as a result of a request for a core cycle. The WC and ADR in the CCW BUF are also updated every time a core cycle is completed. The Action Count (ACT CNT) specifies how many words are to be moved to or from core when a core cycle is started. This count is a function of the Word Count (WC) and Address (ADR) stored in the CCW buffer. Besides keeping track of all the words to be transferred, the channels must keep track of how many words are to be moved to or from core for a given core cycle, because core control is designed to transfer four words (quadword defined by all but bits 34 and 35 of the address) at a time and because the starting address and WC may be such that the first or last word to be transferred may not fall on the quadword boundary of the quadword group. Therefore, it is possible that the first and last core cycle will have to transfer less than four words. Less than four words must also be transferred when fetching CCWs and storing status. In addition to holding the WC and ADR, the CCW BUF also holds the channel CLP. As data is moved to or from core, the WC is decremented by the value in the ACT CNT to keep track of the number of words. When the WC goes to zero, the CLP is used to fetch the next CCW.

Besides granting memory cycles to the EBox and to the channels, the MBox assembles the desired physical address to accommodate the type of request. All addresses that may be needed are made available to the PMA at all times. Then, depending on the type of request the MBox granted, the PMA is controlled to select the desired address mixture. The PMA gets the entire virtual address from the EBox virtual memory address register (VMA), the physical page address from the page table, the physical page address from the cache directory, and the physical channel address from the CCW BUF. In addition, the PMA has access to the User Base Register (UBR), Executive Base Register (EBR), and the Cache Clearer Address register (CCA), which are loaded at some point with an appropriate address from the VMA. The page table is filled as the EBox makes paged requests for words for which the page table has no valid physical page address. A page refill mechanism is employed to automatically fetch page table entries from one of the core process tables and write them into the page table (KI mode) or to alternately inform the EBox that it must perform a page refill operation and write the physical page address into the page table (KL mode).

1.2 PAGER

The pager is a high-speed, set-associative, automatic buffer memory that holds the mapping information from the process tables (page tables) in main memory.

User programs reference instructions and data via virtual (or logical) addresses. These addresses are not absolute (physical core addresses) since any given page can reside anywhere in core when the program is running. The monitor determines where the entire program will reside and also, if a contiguous segment is not available, it will assign core on a page-by-page basis. Therefore, since user programs are allocated core dynamically, the transformation from virtual address to physical address must also be performed dynamically. As the monitor assigns core to a user program, the user process table and associated page tables are created to specify where in physical core the user program resides.

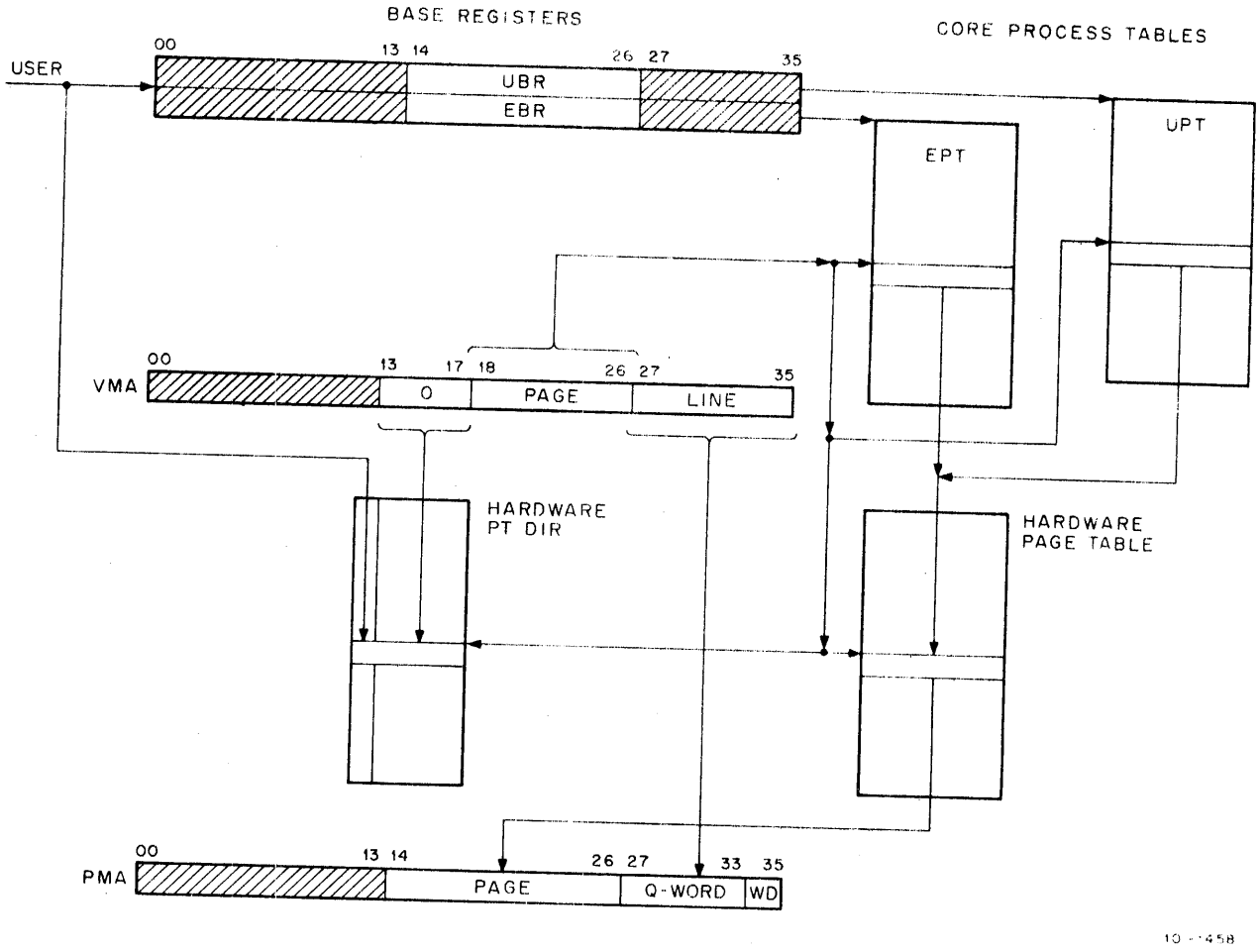


Figure 1-4 KI Paging Scheme (User and Exec Mode)

This information is specified on a page-by-page basis. Then, when a given user program is given time to run by the scheduler, paging data is transferred from the user process table and associated page tables to the hardware tables in the pager (Figures 1-4 and 1-5). The hardware tables include a page table and a directory. The page table contains 512 locations to accommodate translation requirements for all 512 pages of a section. The page table is logically divided into sets of four, which are identified by virtual section address entries in the 128-location directory. Both the page table and the directory are addressed by the virtual page address to store and retrieve translation entries. Consequently, this structure of the pager facilitates maintaining translation information for all 512 pages from any section. The pager may contain a mixture of pages from several sections of both the user and executive address space.

As the running program references core memory, the pager is filled. Eventually, the pager will have enough entries to eliminate the need for further references to memory for paging information. At this point, all virtual addresses can be transformed by the entries in the hardware tables when KI paging is used (Subsection 3.2).

When a given user program runs out of time, all entries in the hardware tables are invalidated by setting the NOT VALID bits in the directory table and the procedure is then repeated for the next scheduled program.

The pager transforms the virtual page address into a physical page address and checks the page access keys every time the EBox makes a paged read or write request.

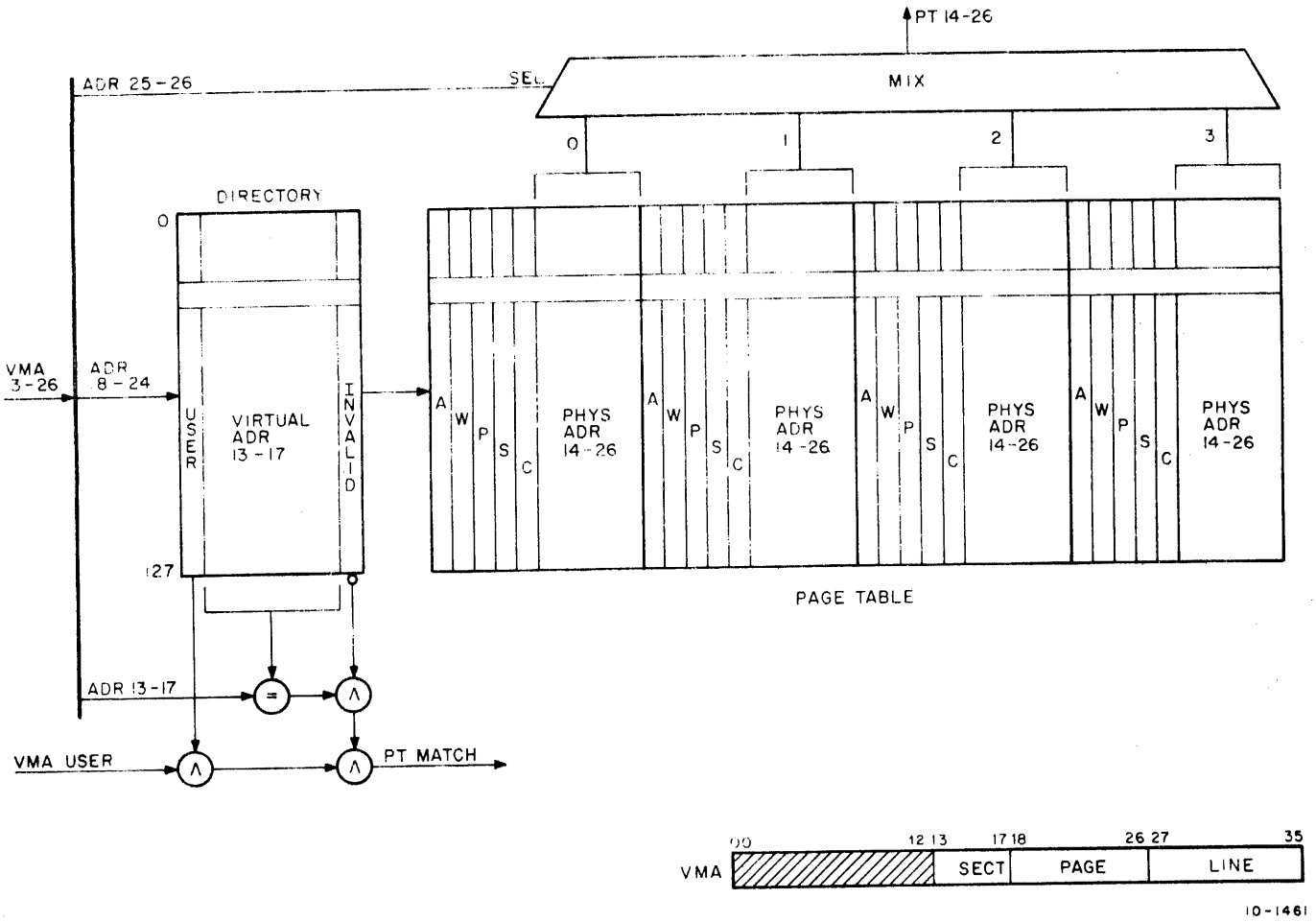


Figure 1-5 Pager Structure

The transformation, essentially, is the replacement of the virtual section and page address with the physical page address. Both tables are automatically filled as virtual addresses are referenced by the user program. These entries are then used to determine if the entries are valid, and if so, to use the desired entry (addressed entry) as a replacement for the virtual section and page number.

If the pager does not contain a valid entry, one of two courses of action can take place. For KI paging, the MBox starts a page refill cycle to fetch four words (8 entries) from the process table and then retries the request. If, after refilling the page table, the request cannot be honored because of the state of the access keys, the EBox is informed that a page fail condition occurred. The EBox must then take an alternate course of action and retry the request. For KL paging, the MBox clears the addressed page table location and informs the EBox that a page fail condition occurred. The EBox must then calculate the physical page address, write the address into the page table, and retry the request (Subsection 3.2).

1.3 CACHE

The cache is a high-speed, multiple set-associative, automatic buffer memory. This buffer serves as a high-speed extension of main memory to hold some selection of words from the main (core) memory system to reduce access time and to cut the percentage of available memory cycles needed by the EBox.

Besides reducing the memory access time, this benefits the channels in that they can get a greater percentage of available memory cycles, thereby minimizing possible data overruns. The basic addressable element of core memory is a 36-bit unit called a word. The memory address mechanism generates a 22-bit physical memory address allowing for up to 2^{22} words (4 million) of main memory (Figure 1-6). Consequently, main memory can be considered to be a string of words as shown in Figure 1-7.

Core memory can also be viewed as a 2-dimensional array as shown in Figure 1-8.

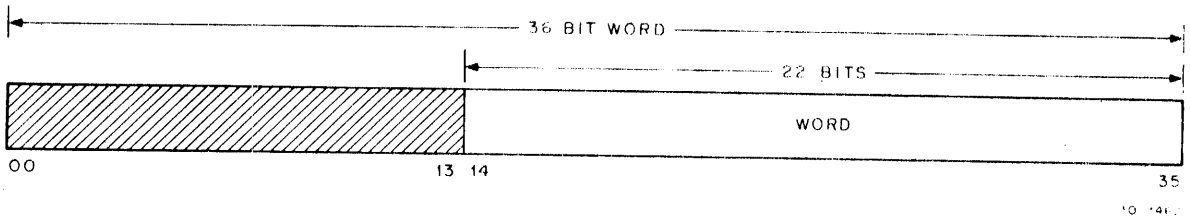


Figure 1-6 Address Format for Linear Address Space

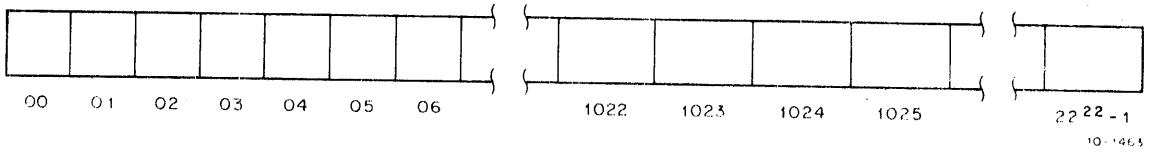


Figure 1-7 Linear Address Space Representation

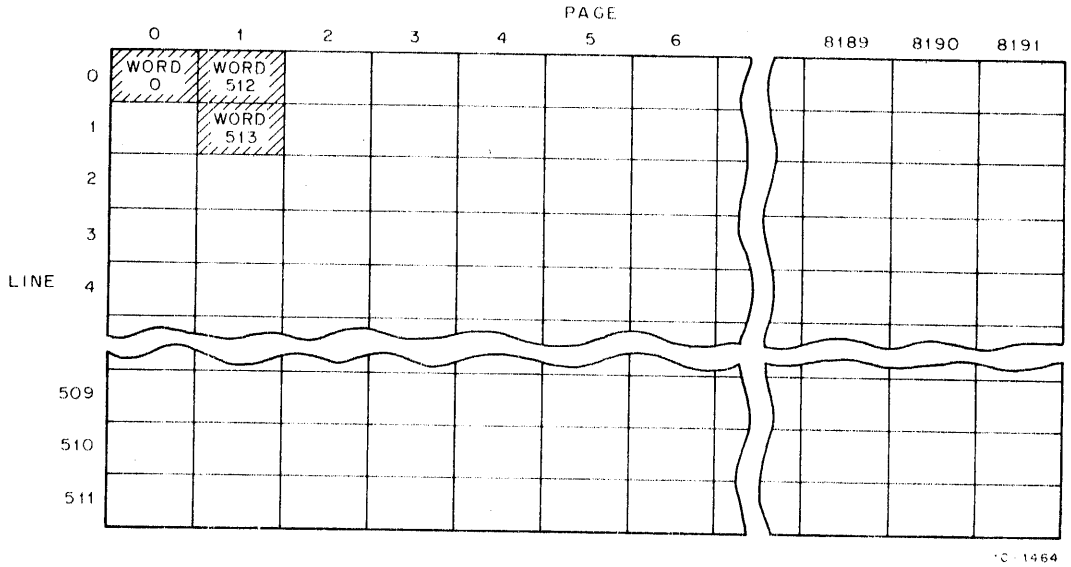


Figure 1-8 Two-Dimensional Address Representation

To complement the two-dimensional address space as shown in Figure 1-8, the 22-bit address is interpreted as a 13-bit Page number and a 9-bit Line number, as shown in Figure 1-9. For example, word 513 is a word in Page 1, Line 1. This is the convention that is used in the KI10.

Another way of looking at core memory is somewhat 3-dimensional as shown in Figure 1-10.

In this perspective, memory is logically divided into pages of 512 words that are divided into 128 sets of four words. A line then contains four words from each page. The 22-bit address is interpreted as a 13-bit page number, a 7-bit quadword (Q-Word) number, and a 2-bit word number, as shown in Figure 1-11.

It is this perspective of main memory that should be kept in mind when reading about the cache.

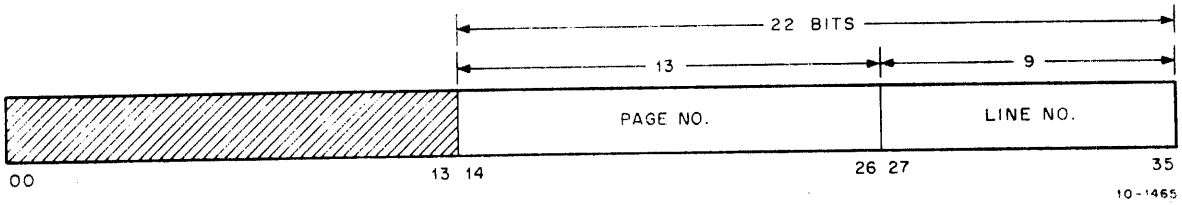


Figure 1-9 Address Format for Two-Dimensional Address Space

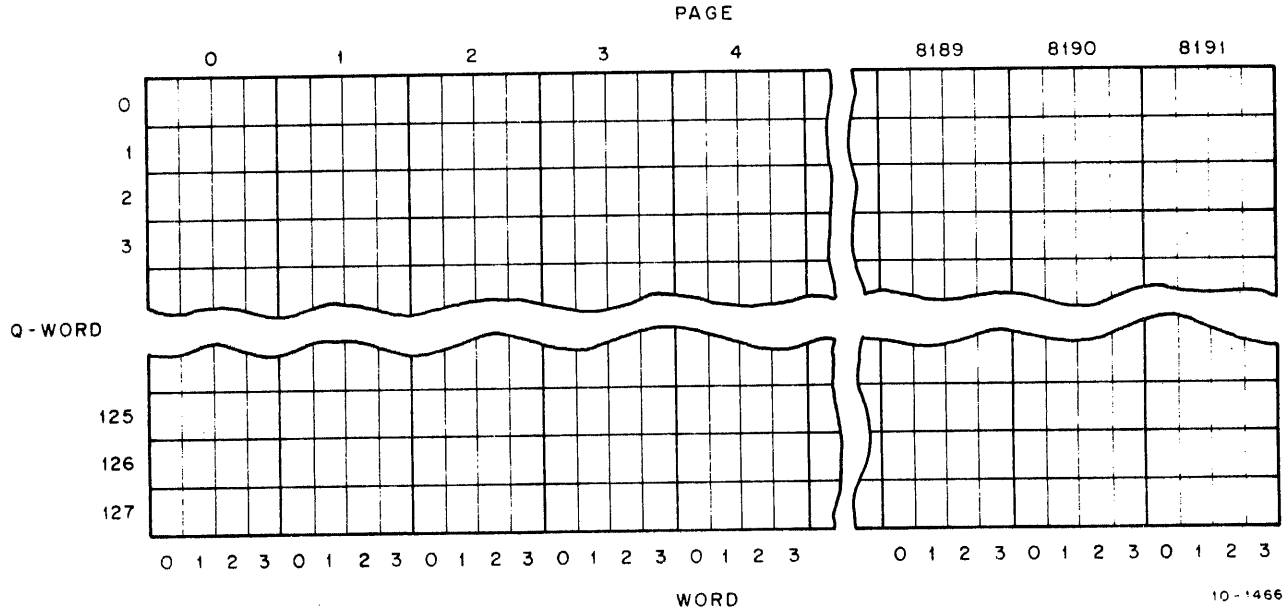


Figure 1-10 Pseudo Three-Dimensional Address Space Representation

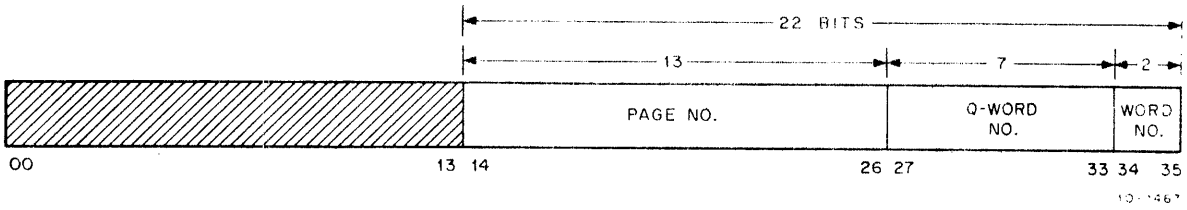


Figure 1-11 Address Format for Pseudo Three-Dimensional Address Space

The cache consists of a data buffer for storing instructions and operands, and a directory buffer for storing the physical memory address and status (VALID and WRITTEN bits) information (Figures 1-12 and 1-13). The contents of the Directory buffer identify the contents of the data buffer. The cache data buffer contains 2048 locations, each of which is associated with a valid and a written bit location in the directory. These 2048 word data and status bit locations are divided into 512 sets of four, which are directly associated with corresponding address locations in the directory. In addition, the 512 sets of data and directory locations are divided further into sets of 128, resulting in four cache quarters (pages). This results in a cache structure similar to the pseudo 3-dimensional structure described previously, where the least significant nine bits of the memory address, which are not subject to paging, can be used to address four blocks (a cache line) of the cache simultaneously. If a copy of a block is made from main memory, it is always and only stored in one of the four corresponding (addressed) blocks of the data buffer. The actual block to be used is specified by the contents of a use table. This table maintains a record of the order in which the four addressed cache blocks are used and maintains one entry for each of the 128 lines in the cache. The contents of the use table are employed to select the block that contains the Least Recently Used (LRU) data for storing the new data; thereby, the LRU data is always supplanted. Besides writing a block of four words into the cache data buffer, the associated directory locations are also updated to specify the valid words and the physical address of the data block. The written bits in the cache directory are not set when data is moved from memory to the cache but are set only when the EBox writes into the cache. When words are written into the cache by the EBox, the address and the valid bit in the directory are also updated.

NOTE

Write through to memory is not implemented to conserve core cycles while the user program is running.

The convention that a block from main memory is always stored in the LRU block of the corresponding data buffer line ensures that a given line in the data buffer will never contain more than one quadword from a given page. Therefore, a conflict (more than one address in one line matching) will never occur when comparing the address with the contents of the directory to determine if the desired word is in the data buffer. This feature of refilling the cache also tends to keep instructions and operands that are used more frequently stored in the cache for a longer period of time.

At any given time the cache may contain up to 512 quadwords (2048 words). The distribution may range from four complete pages, from anywhere in core, to four words from every page of any section of core. A section of core contains 512 pages. Every time the EBox makes a paged request for which the page test was OK (or an unpaged request) to read or write a word, the cache directory is checked to see if a record exists for the quadword in which the requested word is located. If an address matches and at least one valid bit in that block is set, then the cache has a record of the quadword.

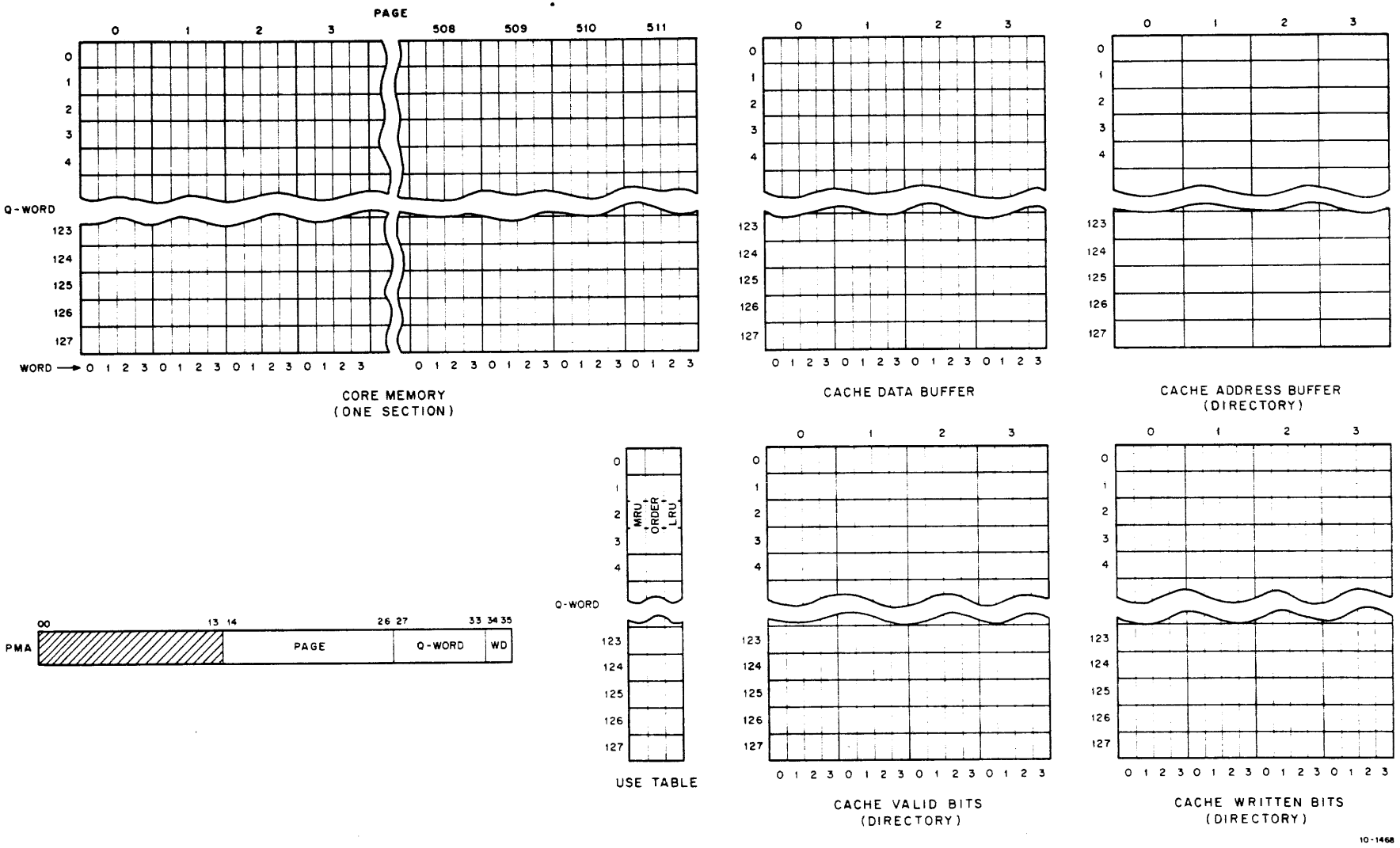


Figure 1-12 Logical Structure of Core and Cache Memory

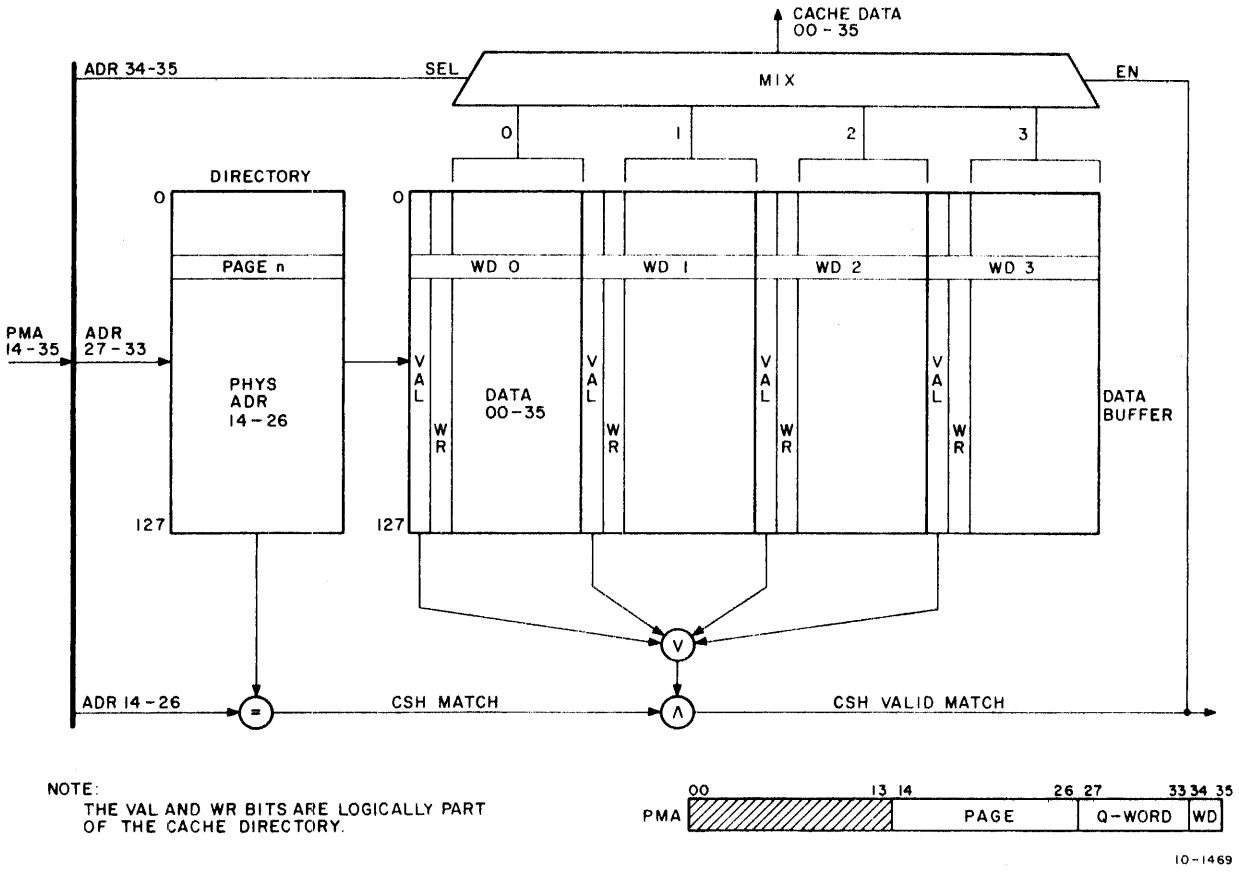


Figure 1-13 Cache Structure (Detail A)

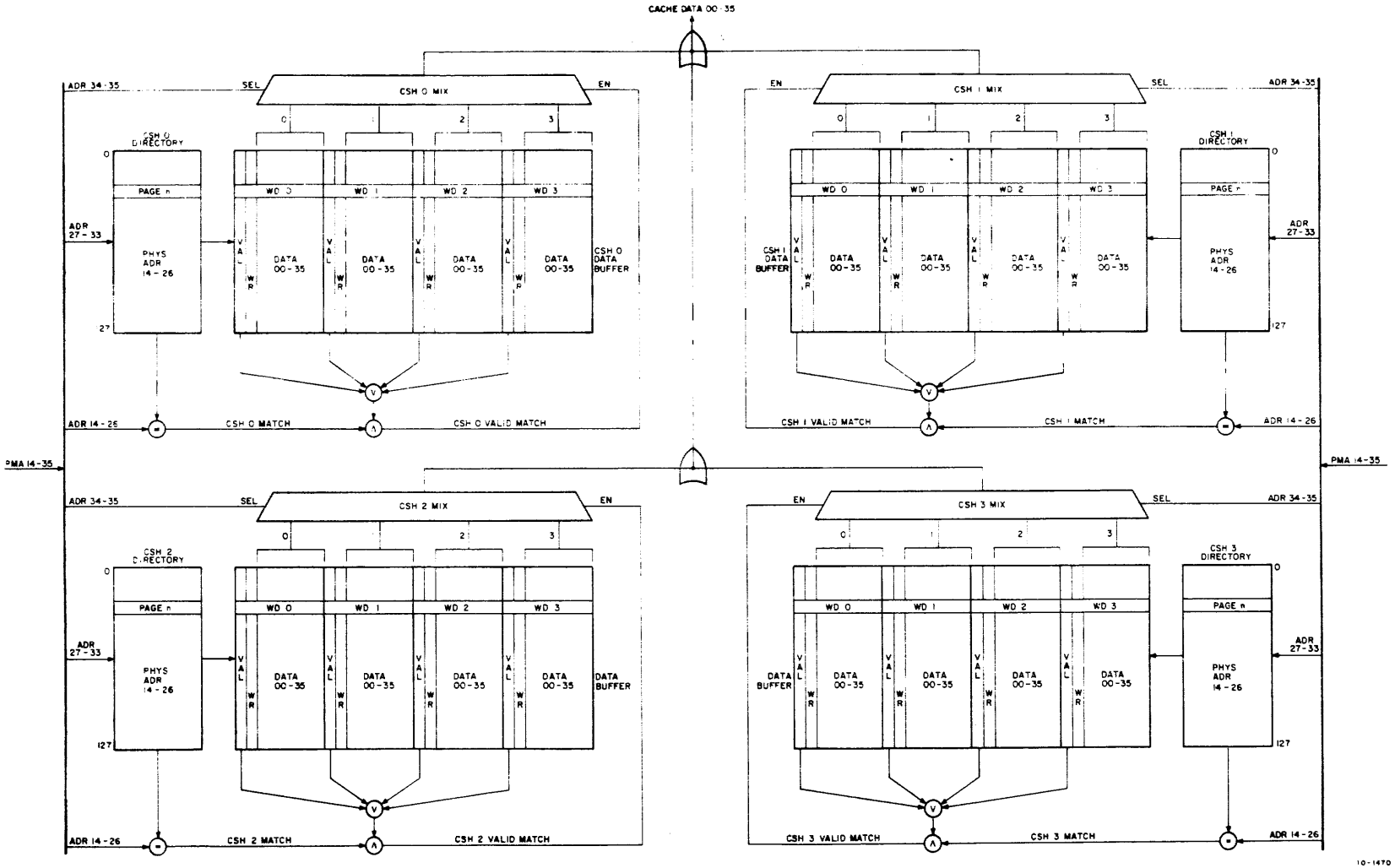


Figure 1-13 Cache Structure (Detail B)

When set, the valid bits identify those words that were placed in the cache due to a cache refill operation or in response to an EBox write request. A cache refill operation is initiated by the MBox cache control in response to an EBox read request if the requested word is not found in the cache. The written bits, when set, identify those words that were placed in the cache in response to an EBox write request. The words that are written into the cache by the EBox are identified so that the core copy can be updated when necessary.

If the valid bit for the requested word (EBox read request) is set and the directory address matches the given address, the word is in the cache and the addressed location is simply read.

If one or more words of a quadword group are valid, but the requested word is not valid, a cache refill operation is initiated to fetch all non-valid words of the quadword group. The word requested by the EBox will come in first to be available for the EBox and the remaining words will come in from core in ascending modulo four order. Besides making the first word available to the EBox, the words are moved into the MBs and then into the cache. As each word is placed in the cache, the valid bit for that word is set to update the cache directory record.

If the cache does not have a record of the quadword (address does not match and/or no valid bits are set) the LRU cache block is checked to see if any written bits are set. If none of the written bits are set, then the block is available for use. In the case of an EBox write request, the addressed word in the selected block is simply written, and the corresponding directory address, valid bit, and written bit locations are updated. In the case of an EBox read request, the page address is recorded in the cache directory and a core read cycle is initiated to fetch the desired word first and the three other words of the quadword group, in ascending modulo four order, as described before. As the words are moved from the MB to the cache, the corresponding valid bits are set to update the directory.

If one or more written bits are set, the core copy must be updated before the LRU cache block can be used. Core is updated by initiating a writeback cycle. This cycle causes all written words in the LRU block to be moved to the MBs and then to core. As each word is moved to the respective MB, the written bit for the word is cleared. After all words are on their way to core, the EBox request is retried. This time, no written bits will be set, permitting this block to be used for the current request, as described before (Subsection 3.3).

1.4 CACHE CONTROL

The cache control executes requests initiated by the EBox and the channel control. Both the EBox and the channel control can issue data read and data write requests to the cache control. The EBox can also request to load or read internal MBox registers, check if a given page is writable, map the virtual address, and sweep the cache.

Data read and write requests from the EBox and from the channel control cause the cache control to enter a specific cache cycle and step through a set of time states. (The relevant time-state-set varies with the cycle.) The cache control can execute four major and two minor cache cycles (Table 1-2).

Table 1-2 Cache Cycle Types

Cycle	Major	Minor
CSH EBOX	X	
CSH PAGE REFILL		X
CSH WRITEBACK		X
CSH MB	X	
CSH CCA	X	
CSH CHAN	X	

All EBox requests are serviced by the MBox by starting a cache EBox cycle. As the cache control advances through the relevant states in response to an EBox request the page table (if paged reference) and cache directory are checked for valid entries. Page table entries are valid when the USER bit and section address matches the EBOX USER signal and the virtual section address presented by the EBox and the INVAL bit in the table is cleared. Cache entries are valid if the address of the requested word is found and the valid bit is set in the cache directory. If a valid entry is found for an EBox request, the data is simply transferred between the cache and the Arithmetic Register (AR). If a valid entry is not found and the EBox requested to read a word, the cache control initiates a core read cycle to fetch the desired word along with adjacent words of the quadword group. For EBox write requests, the cache control writes the word into the cache block that has a record of one quadword or into the least recently used cache block; no core cycle is started. Words coming in from core are placed into the MBs by the core and MB controls and then are individually moved into the cache by the cache and MB controls. The first word, which will be the word the EBox requested, is placed on cache data lines so that the EBox can take it. Words are written back into core only when the EBox makes a request to read or write a word (except for cache sweep) and a valid entry is not found but the written bit is set. Having the written bit set means that the corresponding data is more up to date than the core copy and, therefore, core must be validated before that cache location can be used for the pending request. To write words back to core, the cache and MB controls move the words into the MBs and start a core write cycle after the first word is placed into an MB.

The channel control does not write into the cache, but moves the words to be written from the channel buffer to the MBs and causes the cache control to invalidate any valid entries in the cache. On channel writes, the valid entries in the cache (if any) are invalidated because it is defined that data coming in from mass storage is more up to date (or is another process) than any data that may still be in core or in the cache. Therefore, on channel write requests, the cache control always initiates a core write cycle. On channel reads, any valid entries in the cache will be moved into the MBs and a core read cycle will be initiated for the remaining words requested, if any. The channel control then moves the words from the MBs to the channel buffer (Subsection 3.3).

1.5 CHANNELS

The channel I/O processor (channel control), which is an integral part of the storage controller (MBox), is time-divison multiplexed to provide service for up to eight separate synchronous channel paths simultaneously. A typical disk channel consists of Main Memory (MA20), the channel control in the MBox, one RH20 Massbus Controller, and one of eight mass storage drives. Each mass storage drive, implemented on a given channel, is connected to the same RH20 Massbus Controller. The controller is connected to the EBox via the asynchronous EBus, which allows the EBox to issue control and data transfer commands to the controller and the associated drives. The controller is also connected to the MBox via the CBus. This path is the synchronous data path, which allows the controller to access memory via the MBox channel control without having to utilize the EBus and the EBox. This configuration frees the EBox to perform computation and execute direct I/O operations to other controllers and devices while the channels are executing a data block transfer. Memory fetch and store operations can also be performed by the EBox while the channels are busy executing a block transfer, provided the cache is implemented. Otherwise, the EBox must compete with the channels for core cycles.

Each block transfer between main memory and a mass storage drive must be initiated by the EBox. This is done by the EBox (under program control) by setting up the channel command list in main memory, and by execcuting DATAO instructions to transfer one or more command words and other control information to a specific controller. The channel command list serves as a control program for executing the block transfer to/from a series of contiguous segments (buffers) of main memory. The control information and commands specify one particular drive of those connected to the controller, a physical starting block address, a block count, a command function (read or write) code, and other control bits such as reset CLP and/or store status, if required.

As soon as the block address and command are transferred to the drive, which is done automatically as soon as the drive is not busy, the controller informs both the channel control and the drive to start the block transfer. To get ready, the channel control fetches the first word in the channel command list. If the block transfer is a channel read operation (NOT CTOM), which is specified by the RH20, the channel control also fetches at least two words of data from the locations specified by the address field of the CCW. This is done because the controller has a two-word data buffer for which words will be requested as soon as the channel control is ready. The drive, on the other hand, will remain dormant until it reaches the specified block address. When the block is reached, the drive, the controller, and the channel control will operate together under the control of the channel command list and the block counter to transfer the block(s) of data. Both the controller and the channel control contain data buffers to normalize the transfer speeds of the different components in the channel path. As the buffers are filled/emptied, additional requests will be made via the buses and interfaces in the path to keep the data moving until the entire block transfer is done. The transfer is done when the channel control fetches a HALT CCW, or when it is executing a LAST DATA XFER CCW and the WC field of that CCW has reached zero and when the block counter in the controller overflows.

The RH20 controller maintains and updates the block count as the block transfer is executed. Up to 1024 blocks can be specified when the read/write command is issued by the EBox. When the block count overflows, the RH20 interrupts the EBox to inform it that the transfer is done. The RH20 also informs the channel control that the transfer is done.

The channel control logic maintains a status and CLP word and a CCW. These two words are kept in the CCW BUF. To keep track of these words for all the channels, the CCW BUF contains two locations for each of the eight possible channels. The format of the two channel command words is given in Figure 1-14. The status/CLP word (relative location 1 in the CCW BUF) contains the status of the channel and the so-called address (program counter or CLP) of the next CCW to be executed.

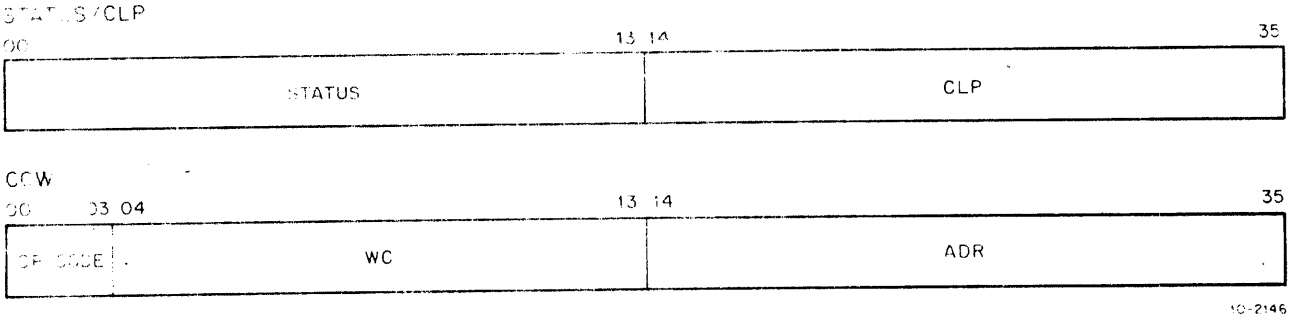


Figure 1-14 Channel Command Word Formats

The initial CCW is kept in the EPT. The status bits of word 1 are updated by the channel control when the channel logs out, which occurs on an error condition, or when the block transfer is completed (done) if a store operation was specified when the transfer was initiated by the EBox. The channel control logs out by writing the appropriate status/CLP and CCW words into the preassigned EPT locations.

The CCW word (relative location 0 in the CCW BUF) contains the current channel command word. This word specifies the operation (instruction) the channel control is to perform. The word contains a 3-bit op code field that specifies one of the following six operations.

a. Op code 0 specifies a Halt operation.
b. Op code 2_8 specifies a Jump operation.
c. OP code 4_8 specifies a Forward Data Transfer operation.
d. Op code 5_8 specifies a Reverse Data Transfer operation.
e. Op code 6_8 specifies a Forward Last Data Transfer operation.
f. Op code 7_8 specifies a Reverse Last Data Transfer operation.

After being started, the channel control will continue to fetch CCW until it gets a HALT CCW or a DATA TRANSFER CCW. In response to a HALT CCW, the channel control will simply halt and it may cause the channel control to log out, if so specified, when the transfer was initiated. In response to a JUMP, the channel control will simply fetch another CCW. The location of the next CCW is specified by the contents of the ADR field of the JUMP CCW. In response to a DATA TRANSFER CCW, the channel control will transfer the number of words specified by the WC field from/to the starting address specified by the ADR field.

1.6 CHANNEL CONTROL

The channel control continuously scans the RH20 Massbus Controllers to see if a data transfer is to be started, executed, or terminated. A controller is allowed to transmit or receive control information (to start or terminate a transfer) and data only after it is selected. A RAM is used by the channel control to buffer the data, and to keep track of the channel status/CLP and CCW words of each channel. When a controller starts a transfer, the RAM is initialized to remember the type of transfer the controller requested. As data is transferred between the channel and the controller, the RAM is continually updated to keep track of various parameters describing the status of the transfer. At the beginning and at specific times throughout the transfer, the channel control will request to transfer data to or from memory by initiating an MB request. These requests are made to:

a. Fetch a CCW
b. Transfer data to or from memory
c. Store status

These requests are initiated by monitoring the contents of the RAM as a function of the scanner, thereby monitoring the status of the selected channel and, when needed, issuing a request for that channel. When an MB request is initiated, the channel control requests a cache cycle to check the cache for any valid words, to move the data between the CH BUF and the MB, and to start a core cycle (Subsection 3.8).

1.7 CACHE CLEARER CONTROL

The cache clearer control executes the cache sweep operation after the EBox executes the "Sweep" instruction. The Sweep instruction is used in a program to validate core and/or invalidate the cache. Core must be validated in the event of a power failure to prevent the loss of written data, before initiating a channel read operation (1080/1090 external channels only), or when rescheduling a job to another processor in a multiprocessor system. The cache will need to be invalidated when the system is powered-up and after a channel write operation is executed (1080/1090 external channels only). When powering the system up, this operation must be done after the cache refill RAM is loaded to initialize the cache memory (Subsection 3.5).

1.8 MB CONTROL

The MB control moves data in and out of the four MBs in response to gating functions from the cache control, core control, or the channel control. It can move data out of the MBs while data is still being moved into the MBs. The input and output operations are independent of each other to minimize the transfer time (Subsection 3.6).

1.9 CORE CONTROL

The core control executes core read and write cycles in response to requests from the cache control and the channel control. Up to four words, in any combination, can be requested by either control. The number of words to be read or written depends on a number of conditions.

a. Read Request from Cache Control: Requests are made to read a single word or read those words that are not in the cache. Bits 34 and 35 (LSB) of the SBus address specify which word is to be fetched first. The remaining words will come back in ascending modulo four order. As each word comes in, it is placed in the MB by the core and MB controls. Words that were not in the cache are then written into the cache by the cache control.

b. Write Request from Cache Control: Requests are made to write a single word or write those words that have been written in the cache by the EBox to make room in the cache or to validate core. The written words in the cache are moved to the MBs by the cache and MB controls and then written back to core by the core control.

c. Read and Write Requests from Channel Control: Requests are made to read or write one, two, three, or four words depending on the current CCW address (ADR) and WC. A given request is confined to those words that occupy the same quadword. That is, the quadword boundary cannot be crossed during a request (Subsection 3.7).

SECTION 2
FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION

This section contains a functional description of the MBox. Appropriate introductory and supportive material is included at the beginning of this section and in each functional description subsection. The following MBox functions are described in this section:

a. EBox Requests
b. Channel Requests
c. CCA Requests
d. Core Requests
e. CBus Requests

In addition, this section describes the error checking and reporting functions and the diagnostic registers implemented in the MBox.

Figure 2-1 illustrates the major functional elements of the MBox. The purpose of this drawing is to support the functional descriptions contained in this section.

The major data and address paths and the individual controls introduced in the previous section are shown in Figure 2-1 with some additional detail. Major interfaces are also shown in some detail.

The EBox is shown gutted in Figure 2-1 to provide a better functional perspective of the MBox in the system.

The interfaces between the EBox and the MBox and between the channels and the cache are not buses, but are functionally shown and described as such because their operation is similar to that of the system buses.

As described before, the MBox serves as the storage controller for the EBox and for up to eight optional integral data channels. Since there is logically only one SBus connected to core memory, the EBox and the integral channels must share the bus in referencing memory. Therefore, one of the main functions of the MBox is to allocate core cycles to the EBox and to the channels. This is done by executing cache cycles on a priority basis. Cache cycles are executed by the cache control in response to requests issued by the EBox and the channels. If the requested word(s) is not found in the cache, a core read cycle is started. Core write cycles are always started in response to channel write requests. Core write cycles are also started when the cache cycle control decides to write written words back to core. Channels are assigned a higher priority than the EBox to minimize channel data overruns. When neither the EBox nor the channels have a request pending, the cache clearer control can get a core cycle if it has a request pending. After a core cycle is started, core will remain busy until all the requested words have been transferred. This means that another core cycle cannot be initiated until the current request is satisfied. In satisfying an EBox request, all but the first word of a quadword group coming in from core will cause MB requests to be issued to request cache cycles for moving the words into the cache.

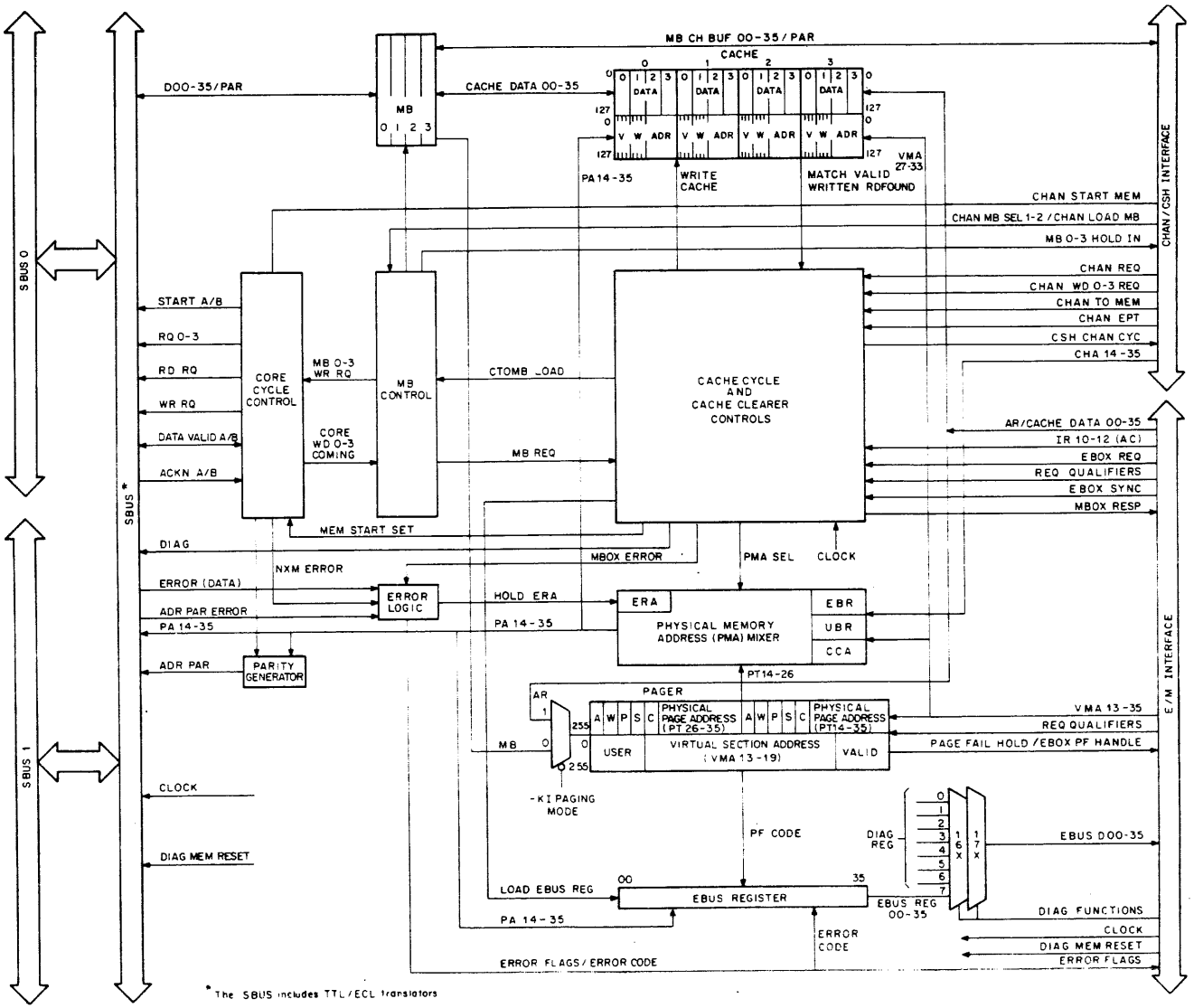


Figure 2-1 MBox Functional Block Diagram (Sheet 1 of 2)

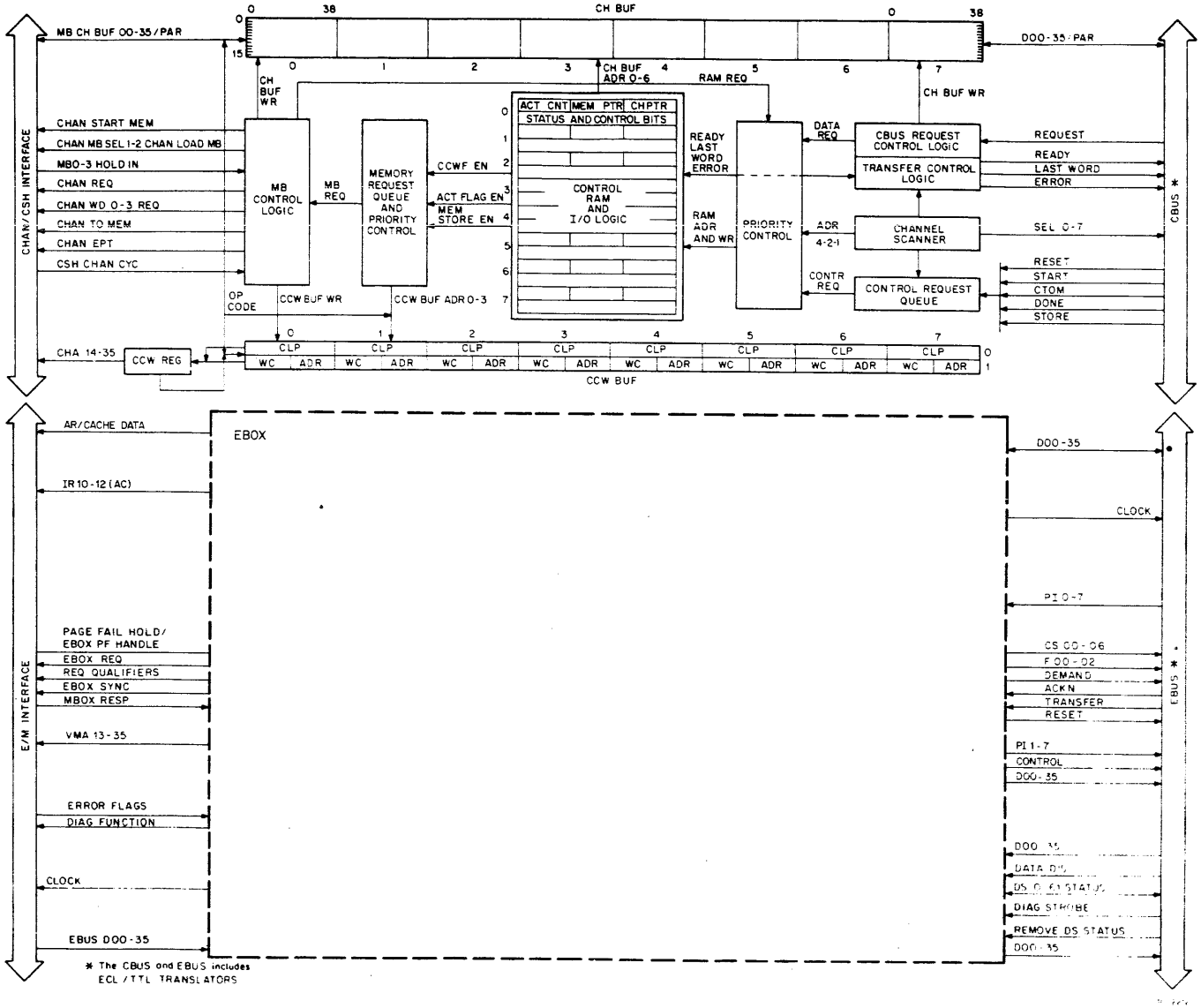


Figure 2-1 MBox Functional Block Diagram (Sheet 2 of 2)

As core cycles are allocated to the EBox and the channels, the MBox also forms the correct physical memory address. To this end, the MBox contains a number of address registers. The address registers that are used in forming the address to service an EBox request can be loaded and read by the EBox.

2.2 CHANNEL RAM CYCLES

RAM cycles are executed by the channel control to keep the contents of the control RAM up to date and to move data in and out of the CH BUF and the CCW BUF. RAM cycles are granted and executed on a priority basis in response to CBus control and data requests, and in response to internally generated MB requests. Accordingly, there are three major types of RAM cycles that can be granted and executed. The types of RAM cycles and their order of priority are given in Table 2-1.

Table 2-1 Major Channel Control RAM Cycle Priorities

Request	RAM Cycle	Priority
CBUS REQUEST	CBUS REQ CYC	1
CBUS START/ RESET/DONE	CBUS CONTR CYC	2
MEMORY REQUEST	MB CYC	3

2.2.1 CBus Request Cycle

CBus request cycles (Figure 2-2) are executed by the channel control in response to CBus requests from the RH20 Massbus Controllers. These RAM cycles are executed to move 36-bit data words between the CH BUF of the MBox channel control logic and the data buffers in the RH20 via the 36-bit CBus data lines.

The controller, when asserting CBUS START, also asserts or negates CBUS CTOM to specify the direction of the transfer. This information is stored in the channel control and is used by the channel control to execute the block transfer correctly.

2.2.2 CBus Control Cycle

CBus control cycles are executed by the channel control in response to CBUS START, RESET, or DONE from the RH20 Massbus Controllers. These RAM cycles are executed to initiate and terminate data block transfers. Data block transfers are initiated by fetching the initial or next CCW. This operation is started by an internally generated memory request. Data block transfers are terminated by emptying the CH BUF and by clearing CBUS READY. A store operation to store the channel status words (current CCW and status/CLP words) will also be executed if the RH20 controller asserted CBUS STORE along with CBUS DONE. The store operation is also initiated by an internally generated memory request.

2.2.3 Channel MB Cycle

MB cycles are executed by the channel control in response to internally generated memory requests (MB REQ). These RAM cycles are executed to request access to main memory (cache/core) and to update the control RAM after a memory operation is done. The channel control will request access to main memory when it needs to fetch a CCW, to fetch or store data, and to store status. Figure 2-2 depicts three types of MB cycles. One type of MB cycle (ADR=0) is shown for the case where the channel is performing a zero fill/skip operation. Another type of MB cycle (INIT) is shown for setting up the channel request for main memory. A third MB cycle (REQ CTR=0) is shown for updating the control RAM after a group (maximum of four) of words is transferred to/from the MBs.

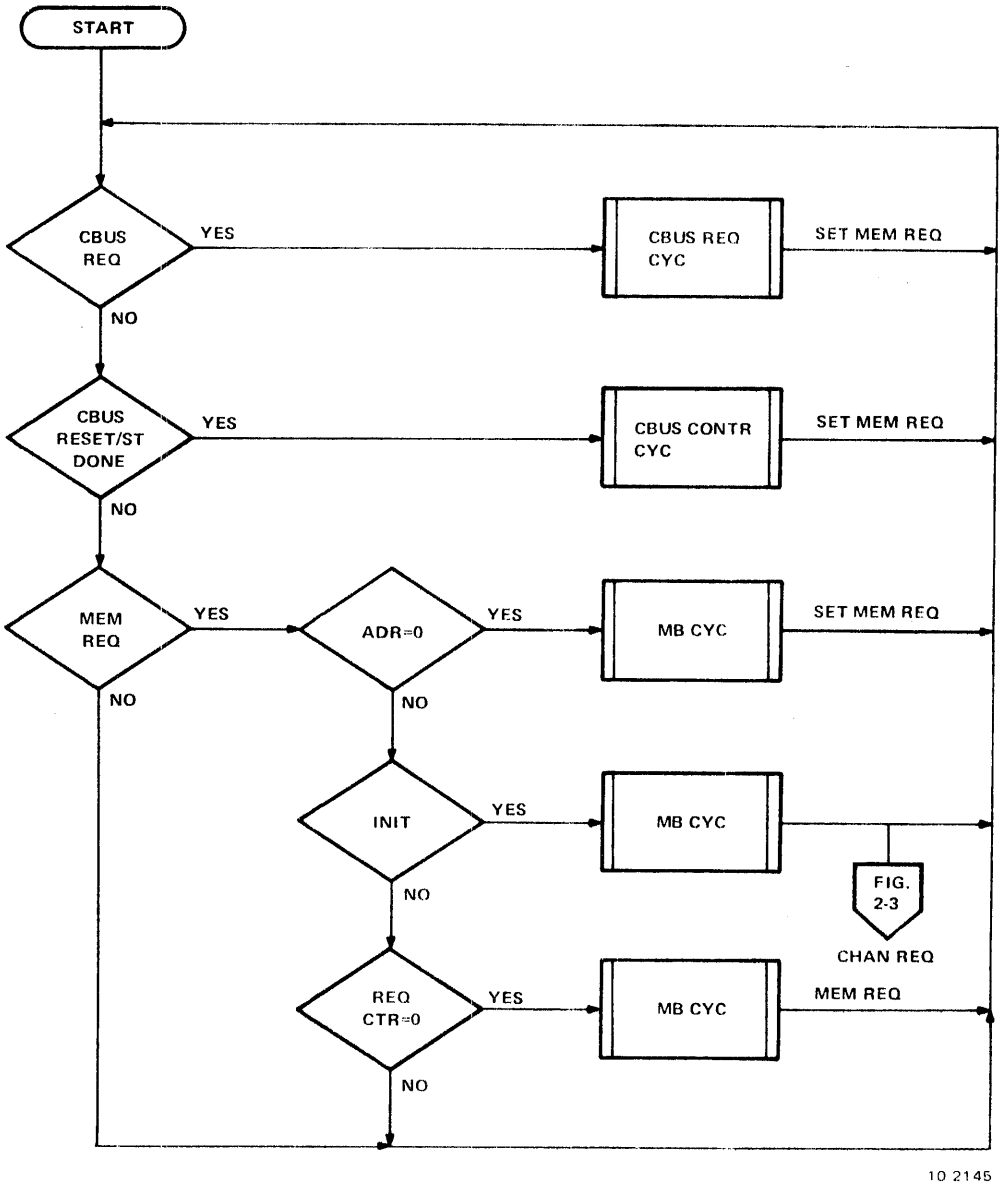


Figure 2-2 Channel RAM Cycle Control, Simplified Flow Diagram

2.3 CACHE CYCLES

Cache cycles are executed to move data in and out of internal registers, the cache or the MBs, to invalidate individual, pages or all pages in the cache, to update core, and to start core cycles. Depending on the type of request that is granted, a particular type of cache cycle is executed. Requests are granted on a priority basis (Table 2-2). There are four major cache cycles that can be executed by the cache cycle control, one to accommodate each type of request.

Table 2-2 Major Cache Cycle Priorities

Request	Cache Cycle	Priority
MB REQ	CSH MB CYC	1
CHAN REQ	CSH CHAN CYC	2
EBOX REQ	CSH EBOX CYC	3
CCA REQ	CSH CCA CYC	4

In addition, there are two secondary cache cycles that can be executed by the cache cycle control. These cache cycles are:

a. Cache Page Refill cycle
b. Cache Writeback cycle

A cache page refill cycle can only be started by a cache EBox cycle to refill the page table when KI paging mode is specified by the EBox. The cache writeback cycle can be started by either the cache EBox cycle or the cache CCA cycle to move written words back to core (Figure 2-3).

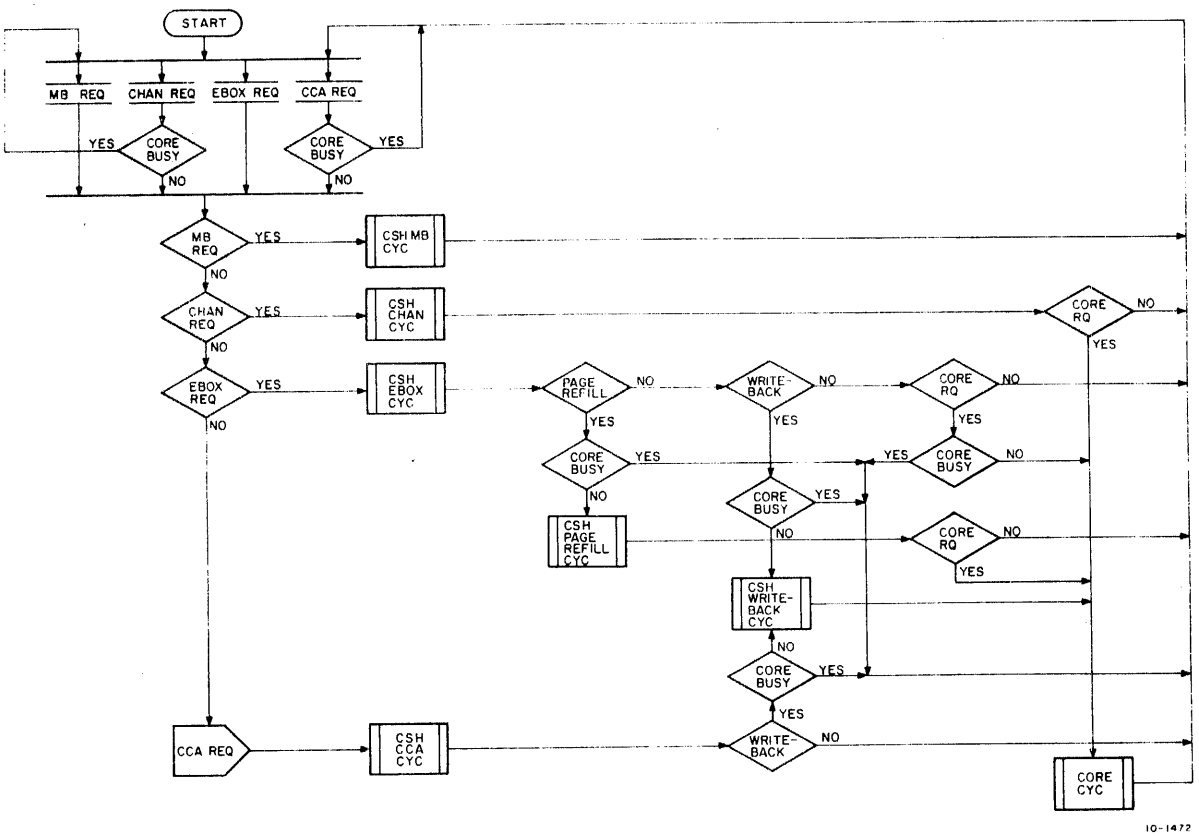


Figure 2-3 Cache Cycle Control, Simplified Flow Diagram

2.3.1 Cache MB Cycle

Cache MB cycles are executed by the cache cycle control in response to MB requests from the core cycle control to move words, which have come in from core and have been placed in the MBs, out of the MBs into the cache. MB requests are issued only if a core read cycle was entered from a cache EBox cycle that is initiated in response to an EBox read request.

NOTE

MB requests are issued only for those words following the first word. This permits the cache cycle control to be freed while core is still busy. The first word is moved to the EBox and the cache before another request can be serviced.

2.3.2 Cache Channel Cycle

Cache channel cycles are executed by the cache cycle control in response to channel requests from the channel control to pick up or invalidate any valid words in the cache and, if necessary, to start a core cycle. To satisfy a channel read request, any valid words are moved into the MBs so that the channel control can pick them up. If all the requested words are not in cache, a core read cycle is initiated to read them for core. To satisfy a channel write request, any valid words in the cache are invalidated and a core write cycle is started after the channel control moves the first word into an MB. The valid words in the cache are invalidated during a channel write operation because the strategy is that the words coming from a mass storage drive are the correct copy. The only case for which a core cycle is not started is if all requested words are found in the cache for a channel read request.

2.3.3 Cache EBox Cycle

Cache EBox cycles are executed by the cache cycle control in response to EBox requests for the EBox to read and write registers, RAMs, and main memory. The EBox can also issue a request to execute a memory diagnostic cycle. To satisfy a memory reference request, the cache cycle control can also enter a cache refill cycle, cache writeback cycle, or a core cycle from the cache EBox cycle. A considerable amount of decision logic is contained in the cache cycle control to determine which path is to be taken to satisfy the request.

2.3.4 Cache CCA Cycle

Cache CCA cycles are executed by the cache cycle control in response to CCA requests from the cache clearer control to invalidate the cache and/or validate core. These operations can be executed for a single page or the entire physical address space. The cache clearer control is activated when the CCA register is loaded by the EBox, which is done when the EBox executes a Sweep instruction.

2.4 CORE CYCLES

Core cycles are executed by the core cycle control to move data in and out of core memory. Core read cycles are executed to read up to four words and core write cycles are executed to write up to four words. If more than one word is to be transferred, they will be transferred in ascending modulo four order, starting with the word specified by SBus address bits 34 and 35.

2.5 ADDRESS PATH SUMMARY

All the address paths implemented in the MBox are shown in Figure 2-4. These paths are implemented to facilitate the formation of the appropriate SBus address and to address the various RAMs in the MBox. The addressable RAMs include the page table and its directory, the cache and its directory, the use table and its refill table, the CCW buffer, the CH buffer, the control RAM, and the RAMs for the pointers and the action count.

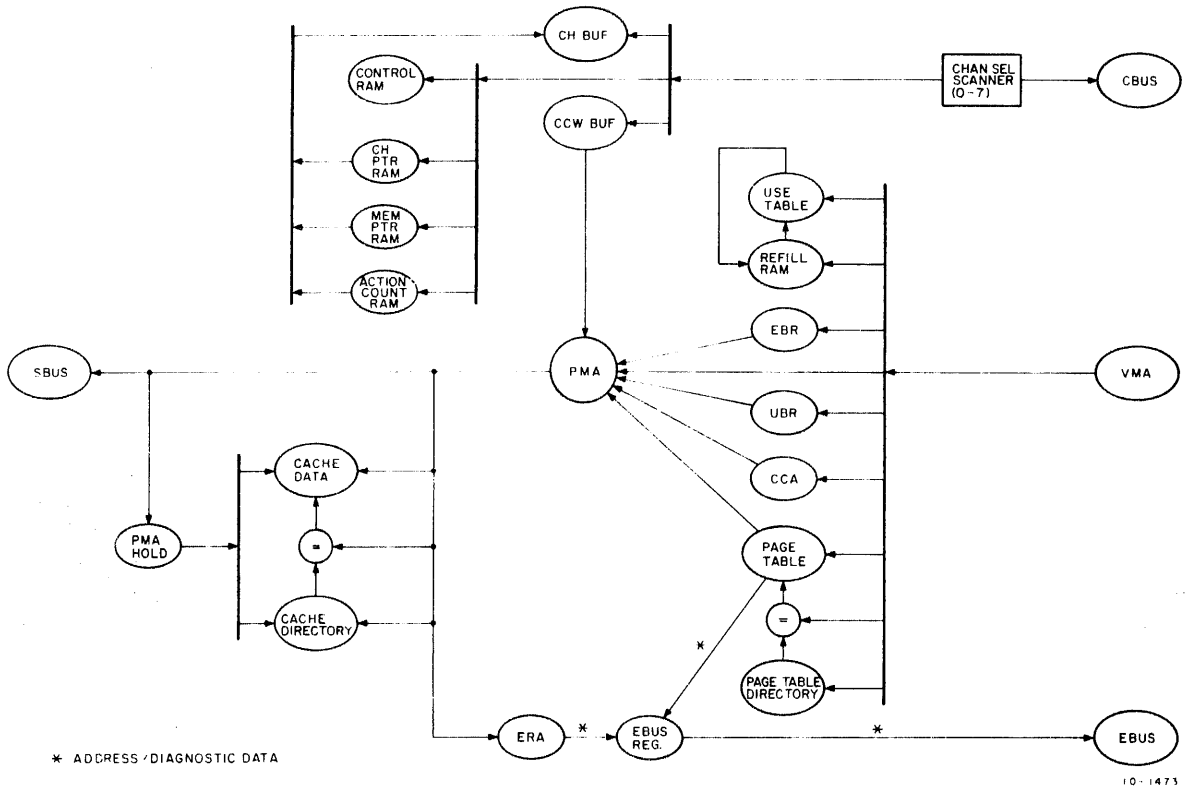


Figure 2-4 MBox Address Paths, Simplified Path Diagram

Any memory request, whether from the channel or from the EBox, must be accompanied by an address. The address accompanying EBox requests is supplied by the VMA in the EBox. The CCW BUF provides the address when the channel makes a request. For EBox requests other than references to memory, the VMA also serves as an address and/or data source. For example, the VMA serves as a data source when loading the UBR, EBR, or CCA, and as an address and data source when loading the cache refill RAM.

The PMA HOLD register supplies the address for cache cycles executing MB requests and the CCA register supplies the address for cache cycles executing CCA requests.

2.6 DATA PATH SUMMARY

All data paths implemented in the MBox are shown in Figure 2-5. These paths are implemented to move data from source to destination buses, registers, and RAMs. The desired path is selected by the cache cycle control when a request is granted, by the core control during a core cycle, and by the channel control. The MBs serve as a buffer in executing most data transfers.

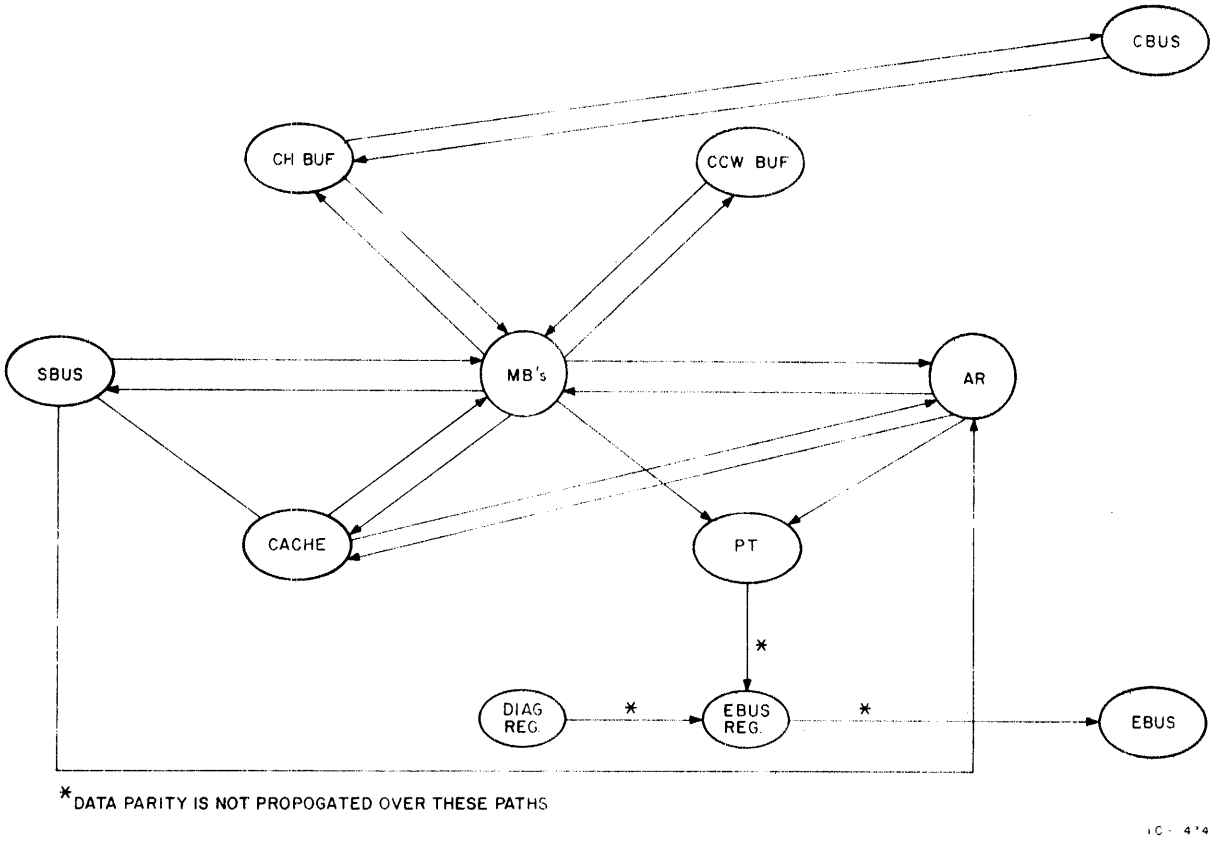


Figure 2-5 MBox Data Paths, Simplified Path Diagram

2.7 EBOX REQUESTS

Requests are issued by the EBox to:

a. Read and write memory
b. Read and load MBox registers
c. Read and write MBox RAMs
d. Initiate a diagnostic cycle

To qualify the request, the EBox asserts a set of interface signals along with CLK EBOX REQ to specify exactly what type of service is desired. From what has been described so far in this section, an obvious request qualification is to differentiate between reads and writes and between memory and register references. Besides these basic qualifications, each request is qualified further by asserting other signals to declare the register of interest in the case of a register reference or declare the type of addressing to be used and whether the cache is to be used in the case of memory references. After the MBox executes a cache cycle to process the EBox request, the MBox will always assert MBOX RESP IN to notify the EBox that the operation is completed.

2.7.1 E/M Interface Summary

A summary of the E/M interface is presented in Table 2-3. The interface signals are grouped into sets according to their function. The notations in parentheses are field maintenance print set prefixes that specify the source of the signals.

Table 2-3 F/M Interface Summary

Signal	Description
A. Control Commands	
CLK EBOX REQ (CLK4)	Issued by the EBox to request service.
CSH EBOX RETRY REQ (CSH2)	Asserted by the MBox to set CLK EBOX REQ so that request will be retried.
MBOX GATE VMA 27–33 (CSH3)	Asserted by the MBox when a Cache EBox cycle is granted to service the EBox Request to enable gated VMA bits 27–33 for addressing the Cache directory.
CSH EBOX T0 IN (CSH4)	Asserted for one clock period when the cache cycle control starts processing an EBox Request. This signal is used to clear CLK EBOX REQ.
VMA AC REF A (VMA1)	Asserted by the EBox when it finds that the reference is to one of the AC blocks (fast memory) to abort the MBox cache cycle if it was started. This is done to allow the MBox to start servicing a request earlier than would otherwise be possible.
PT PUBLIC (PAG1)	Transferred to the EBox to allow the EBox to decide whether it should assert MCL PAGE ILLEGAL ENTRY for the next reference or change its mode of operation from public to private.
PAGE FAIL HOLD (CSH6)	Asserted by the MBox if the page test for any paged memory reference request failed.
PF EBOX HANDLE (PAG4)	Asserted by the MBox if the KL mode page test for a paged memory request failed.
MBOX RESP IN (CSH2)	Asserted by the MBox after the request is processed.
CLK EBOX SYNC D (CLK3)	Asserted by the EBox to inform the MBox that the data will be taken.
CCA REQ (MBX1)	Cleared by the MBox to inform the EBox that the cache clear operation is done.
B. Request Qualifiers	
1. Memory Reference	
MCL VMA READ (MCL2)	Read a word from memory. Read check the page for paged references and assert PAGE FAIL HOLD if page test failed.
MCL VMA WRITE (MCL2)	Write a word into memory. Write check the page for paged references and assert PAGE FAIL HOLD if page test failed.
MCL VMA READ and WRITE (MCL2)	Read a word from memory, read and write check the page for paged references and assert PAGE FAIL HOLD if page test failed.

Table 2-3 E/M Interface Summary (Cont)

Signal	Description
B. Request Qualifiers (Cont)	
MCL EBOX CACHE (MCL6)	Asserted by the EBox for references to those instructions and operands that may reside in the cache. Instructions and operands that must be shared by two processors cannot reside in the cache.
CON CACHE LOOK EN (CON3)	Asserted by the EBox to take the word from the cache if it is found even if MCL EBOX CACHE is negated or for paged references if PT CACHE is cleared.
CON WR EVEN PAR DIR (CON3)	Asserted by the EBox to write even parity into cache directory during a write request.
APR WR BAD ADR PAR (APR2)	Asserted by the EBox to generate even address parity on the SBUS.
APR EBOX SBUS DIAG (APR6)	Asserted by the EBox to initiate and execute an SBUS Diagnostic cycle. All other request qualifiers must be negated for this request.
2. Register References	
APR EBOX LOAD REG (APR6)	Asserted by the EBox to load a register (UBR, EBR, CCA) in the MBox. The EBox also specifies which register is to be loaded by asserting the appropriate register signal.
APR EBOX READ REG (APR6)	Asserted by the EBox to get ready to read a register (UBR, EBR, CCA, ERA) in the MBox. The EBox also specifies which register is to be read by asserting the appropriate register signal. After the Read Register Request is executed by the MBox, the EBox can read the value of the register by simply asserting the Read EBus Register diagnostic function.
APR EBOX UBR (APR6)	Asserted by the EBox when the UBR is to be loaded or read.
APR EBOX EBR (APR6)	Asserted by the EBox when the EBR is to be loaded or read.
APR EBOX CCA (APR6)	Asserted by the EBox when the CCA Register is to be loaded or read.
	NOTE **Instruction bits 10–12 (AC field) must be correctly set or cleared and the VMA bits 27–33 must contain the page address when clearing one page in the cache.**
IR AC10 (IRD1)	Instruction bit 10 (AC10) is set when only one page is to be cleared from the cache and is not set when the entire cache is to be cleared.

Table 2-3 E/M Interface Summary (Cont)

Signal	Description
B. Request Qualifiers (Cont)	
MCL VMA READ, PAUSE and WRITE (MCL2)	Execute the read portion of the read-pause-write cycle. Read and write check the page for paged references and assert PAGE FAIL HOLD if page test failed. The write portion of the cycle is started by asserting CLK EBOX REQ a second time.
MCL VMA PAUSE and WRITE (MCL2)	Write check the page for paged references and assert PAGE FAIL HOLD if page test failed. **NOTE** **When issuing memory reference requests, the EBox must also set up the VMA and the Paging and Cache Qualifiers appropriately.**
MCL EBOX MAY BE PAGED (MCL6)	Asserted by the EBox to indicate that the reference is to be paged. The EBox decides whether the reference is paged or unpaged.
CON KI PAGING MODE (CON3)	Indicates KI Paging mode when asserted and KL Paging mode when negated.
MCL VMA USER (MCL2)	Asserted by the EBox when the memory reference is to the user address space.
MCL PAGE UEBR REF (MCL3)	Asserted by the EBox when the UPT or the EPT is referenced to bypass the page check.
MCL VMA UPT (MCL3)	Asserted by the EBox when the reference is to the UPT to inform the MBox that the contents of the UBR must be used in forming the physical memory address.
MCL VMA EPT (MCL3)	Asserted by the EBox when the reference is to the EPT to inform the MBox that the contents of the EBR must be used in forming the physical memory address.
MCL PAGE ILL ENTRY (MCL3)	Asserted by the EBox to force a page fail condition in the MBox to abort the current request. The EBox asserts PAGE ILL ENTRY if the previous instruction was fetched from a proprietary area and the instruction is not a Portal instruction (JRST1).
MCL PAGE TEST PRIVATE (MCL2)	Asserted by the EBox for a non-instruction reference in the PUBLIC mode to check whether the page is private. PAGE FAIL HOLD is asserted if the page is not public.
MCL PAGE ADDRESS COND (MCL3)	Asserted when the EBox detects an address break condition. The EBox also asserts PAGE ILL ENTRY at this time to force a page fail condition in the MBox and cause PAGE FAIL HOLD to be asserted.

Table 2-3 E/M Interface Summary (Cont)

Signal	Description
B. Request Qualifiers (Cont)	
	NOTE **The term "Clear The Cache" means to write back to core all words that are written in the cache (words that have their written bits set) and/or invalidate the words in the cache.**
IR AC11 (IRD1)	Instruction bit 11 (AC11) is set when the written words in the Cache are to be written back into core to validate core.
IRD AC12 (IRD1)	Instruction bit 12 (AC12) is set when the cache entries are to be invalidated.
	NOTE **The contents of AC bit 10–12 of the instruction is transferred to the CCA control register in the MBox when the EBox issues a request to load the CCA register.**
APR EBOX ERA (APR6)	Asserted by the EBox when the Error Address (ERA) register is to be read. This register can only be read. It is frozen when the MBox senses a parity or a non-existent memory (NXM) error; otherwise it tracks.
MCL EBOX MAP (MCL6)	Asserted by the EBox along with APR EBOX READ REG to transform the virtual address into the physical address. If the page table contains a valid entry, this entry will be transferred to the AR of the EBox. If the page table does not contain a valid entry, a page refill operation will be initiated.
APR EN REFILL RAM WR (APR6)	Asserted by the EBox along with APR EBOX READ REG to load the Cache refill RAM when the Cache is initialized. Before this operation can be executed, VMA bits 27–33 must be set up with the desired address and VMA bits 18–20 must be loaded with the data to be loaded in the refill RAM.
C. Error Reporting Commands MBOX NXM ERR (MBZ3)	This error flag is set when the MBox memory control logic times out ("hangs") or when non-existent memory is addressed.
APR NXM ERR (APR1)	This line serves as the recirculation path for the MBOX NXM ERR flag.
MBOX SBUS ERR (MBZ4)	This error flag is set when the memory system senses a data parity error or times out ("hangs").
APR SBUS ERR (APR1)	This line serves as the recirculation path for the MBOX SBUS ERR flag.

Table 2-3 E/M Interface Summary (Cont)

	Signal	Description
	MBOX MB PAR ERR (MBZ4)	This error flag is set when the MBox senses an MB parity error.
	APR MB PAR ERR (APR1)	This line serves as the recirculation path for the MBOX MB PAR ERR flag.
	MBOX ADR PAR ERR (MBZ4)	This error flag is set when the memory system senses an address parity error.
	APR S ADR P ERR (APR2)	This line serves as the recirculation path for the MBOX ADR PAR ERR flag.
	CSH ADR PAR ERR FLG (MBX5)	This error flag is set when the MBox senses a cache directory parity error.
	APR C DIR P ERR (APR2)	This line serves as the recirculation path for the CSH ADR PAR ERR FLG.
	APR ANY EBOX ERR FLG (APR2)	This line is true when any APR EBox error flag is set to prevent the ERA (Error Address Register) in the MBox from being changed facilitating error recovery procedures.
D.	Direct Commands APR WR PT SEL 0-1 (APR5)	When writing the Page Table, the EBox places the appropriate write select code on these lines.
	CLK PT DIR WR (CLK2)	Asserted by the EBox during KL paging mode to write or clear a page table directory entry.
	CLK PT WR (CLK2)	Asserted by the EBox during KL paging mode to write or clear a page table entry.
	DIAG READ FUNCT 16X and 17X (CTL3)	One or the other line is asserted by the EBox to read a Diagnostic register. The Diagnostic register to be read is specified by the code presented on DIAG 04–06 (0–7). Also asserted by the EBox to read the EBus register (167_8). Octal code seven must be presented on the DIAG 04–06 lines to read the EBus register. This register will contain the contents of the register specified with the EBox Read Register request or it will contain the Page Fail Word in the event the MBox Pager sensed a page fail condition. The EBox is informed that a page fail condition was sensed by the MBox (PAGE FAIL HOLD is asserted by the MBox).
	DIAG LOAD FUNCT 071 (CTL3)	Asserted by the EBox to set up the MEM TO C mixer to read the contents of the memory data register (SBus), the MBs, the CBus (CH REG), or the cache. The contents of the AR can also be looped back. The code presented on the EBus Data bits 30–35 determines which data specified above will be read back on the cache data lines.

Table 2-3 E/M Interface Summary (Cont)

Signal	Description
DIAG 04–06B (CTL3)	These lines present a control code to the MBox for selecting diagnostic and EBus registers.
MR RESET 05/06 (CLK2)	Asserted to initialize system logic.
E. Address	
VMA 13–35A (VMA2)	Register load data or virtual address from EBox.
VMA 27–33G (VMA1)	Gated address from EBox. This address is gated by the MBox to address the cache directory when a cache EBox cycle is started.
	NOTE **Address parity is not propagated.**
F. Data	
AR 00–35A (DP01)	Data from EBox AR.
SH AR PAR ODD A (SHM1)	Data parity from EBox AR parity generator.
CACHE DATA 00–35B (CD01)	Cache/core data to EBox IR, AR and ARX.
CACHE DATA 00–35C (CD01)	Cache/core data to EBox IR.
CSH PAR BIT A/B (MBZ6)	Cache/core data parity to EBox.
EBUS D00–35 (CRC5, MBZ2, CCW5, CHC5, CCL1, CHX4, CSH7, MBC5, MBX6)	EBus data lines.
	NOTE **Data parity is not propagated from the MBox to the EBus.**
G. Clocks	The following clocks are generated on the CLK Module in the EBox and are distributed to the MBox Boards.
CLK CCL (CLK1)	Clock for Channel Control Logic Module M8536
CLK CCW (CLK1)	Clock for Channel Control Word Module M8534
CLK CH (CLK1)	Clock for Channel Control Module M8533
CLK CRC (CLK1)	Clock for Channel RAM Control Module M8535
CLK CSH (CLK1)	Clock for Cache Control Module M8513
CLK CHX (CLK1)	Clock for Cache Extension Module M8515

Table 2-3 E/M Interface Summary (Cont)

Signal	Description
CLK MBC (CLK1)	Clock for MBox Control Module No. 3 M8531
CLK MBX (CLK1)	Clock for MBox Control Module M8529
CLK MBZ (CLK1)	Clock for MBox Control Module No. 4 M8537
CLK PMA (CLK1)	Clock for PMA Module M8518
CLK MB00 (CLK1)	Clock for MB Module No. 1 M8517
CLK MB06 (CLK1)	Clock for MB Module No. 2 M8517
CLK MB12 (CLK1)	Clock for MB Module No. 3 MB8517
CLK SBUS CLK (CLK1)	Clock of SBus
DIAG CHANNEL CLK (CTL3)	Controllable clock for diagnosing channel logic

2.7.2 Request Dialogue

The EBox issues requests to the MBox by asserting CLK EBOX REQ (Figure 2-6). At the same time or one clock tick after CLK EBOX REQ is asserted, the VMA and all request qualifiers become valid. These signals remain valid until the request has either been processed to completion or aborted. CLK EBOX REQ is cleared by CSH EBOX T0 IN when the MBox starts processing the request.

For the first clock period after CLK EBOX REQ is asserted, the request can be aborted by the EBox by asserting VMA AC REF. If the EBox aborts the request, CLK EBOX REQ is also cleared by the EBox if the MBox has not yet started to process the request.

When the MBox starts to process the EBox request, the MBox asserts CSH EBOX T0 IN. This signal causes CLK EBOX REQ to be cleared. This occurs on the clock tick after which the request is made, if the MBox has no higher priority request pending. If the MBox is busy when the request is made a number of clock ticks may transpire before the MBox asserts CSH EBOX T0 IN. Consequently, CLK EBOX REQ will remain asserted until the MBox starts processing the request.

After CSH EBOX T0 IN is asserted, a number of clock ticks may transpire before the MBox completes processing the request. The MBox notifies the EBox that is has completed processing the request by asserting MBOX RESP IN. This signal remains asserted until the EBox asserts CLK EBOX SYNC D. While MBOX RESP IN is asserted, the instruction or operand requested by the EBox will be valid on the cache data lines. The MBox holds the data on the cache data lines until CLK EBOX SYNC D is asserted because the EBox will take the data only when CLK EBOX SYNC D is asserted. One clock tick after CLK EBOX SYNC D is asserted MBOX RESP IN is cleared.

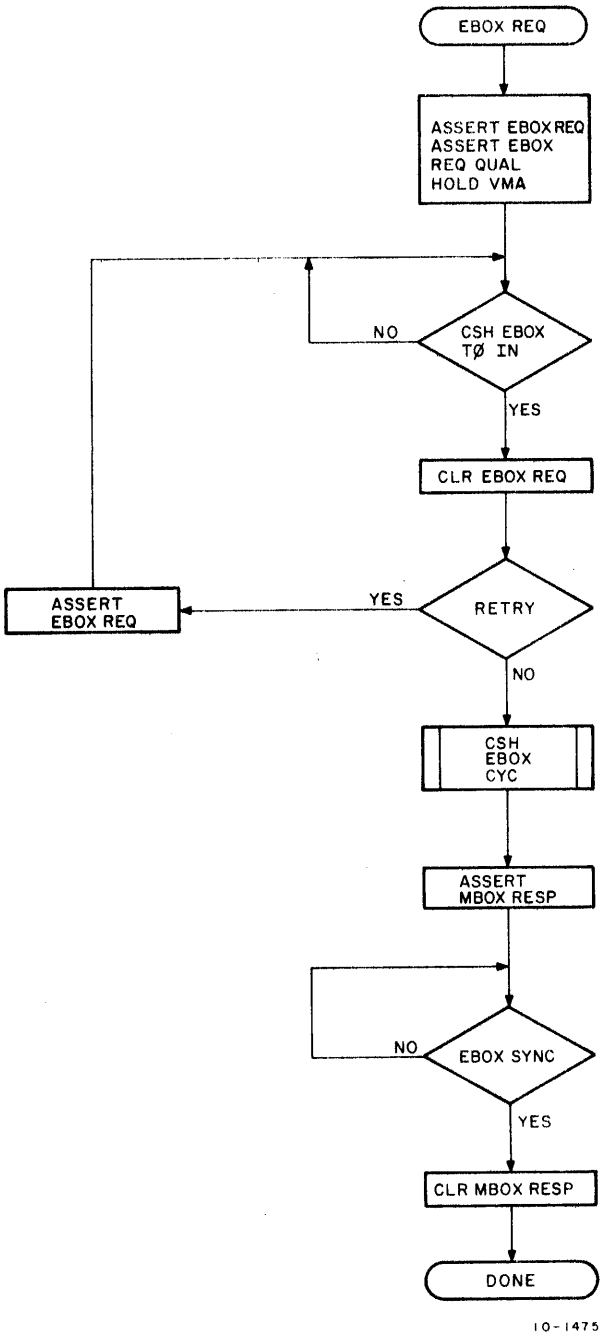


Figure 2-6 EBox Request Dialogue, Simplified Flow Diagram

2.7.3 Register References

The MBox contains a number of registers that can be loaded and read by the EBox. These registers are address registers for storing the address in the event of an error and for modifying the physical memory address in response to certain request qualifiers. The registers are:

a. User Base Register (UBR)
b. Executive Base Register (EBR)
c. Cache Clearer Address Register (CCA)
d. Error Address Register (ERA)

NOTE

The ERA register can only be read by the EBox.

In addition, the EBox can also read the contents of the page table to transform (map) the virtual address to the physical address and load the cache refill RAM with the cache refill algorithm.

To read and load any of the registers and RAMs previously mentioned, the MBox must execute a cache cycle in response to the EBox request to prevent potential conflicts with other pending requests.

NOTE

Some registers and RAMs can also be loaded and read by the EBox directly, without having to execute a cache cycle. The registers and RAMs that fall into this class are those for which a conflict with another type of request (CHAN REQ, for example) cannot occur. The MEM TO C diagnostic register and the page table can be loaded and 16 diagnostic registers (including the EBus register) can be read directly from the EBox.

To read or write the registers and RAMs in the MBox, the EBox must assert a specific set of qualifier signals along with CLK EBOX REQ for each type of reference. When loading registers, the EBox must also move the data to be loaded into the VMA no later than one clock tick after issuing the request. All register operations the EBox is capable of requesting, and the required request qualifiers, are given in Table 2-4. Flows for each type of register operation are shown in Figure 2-7.

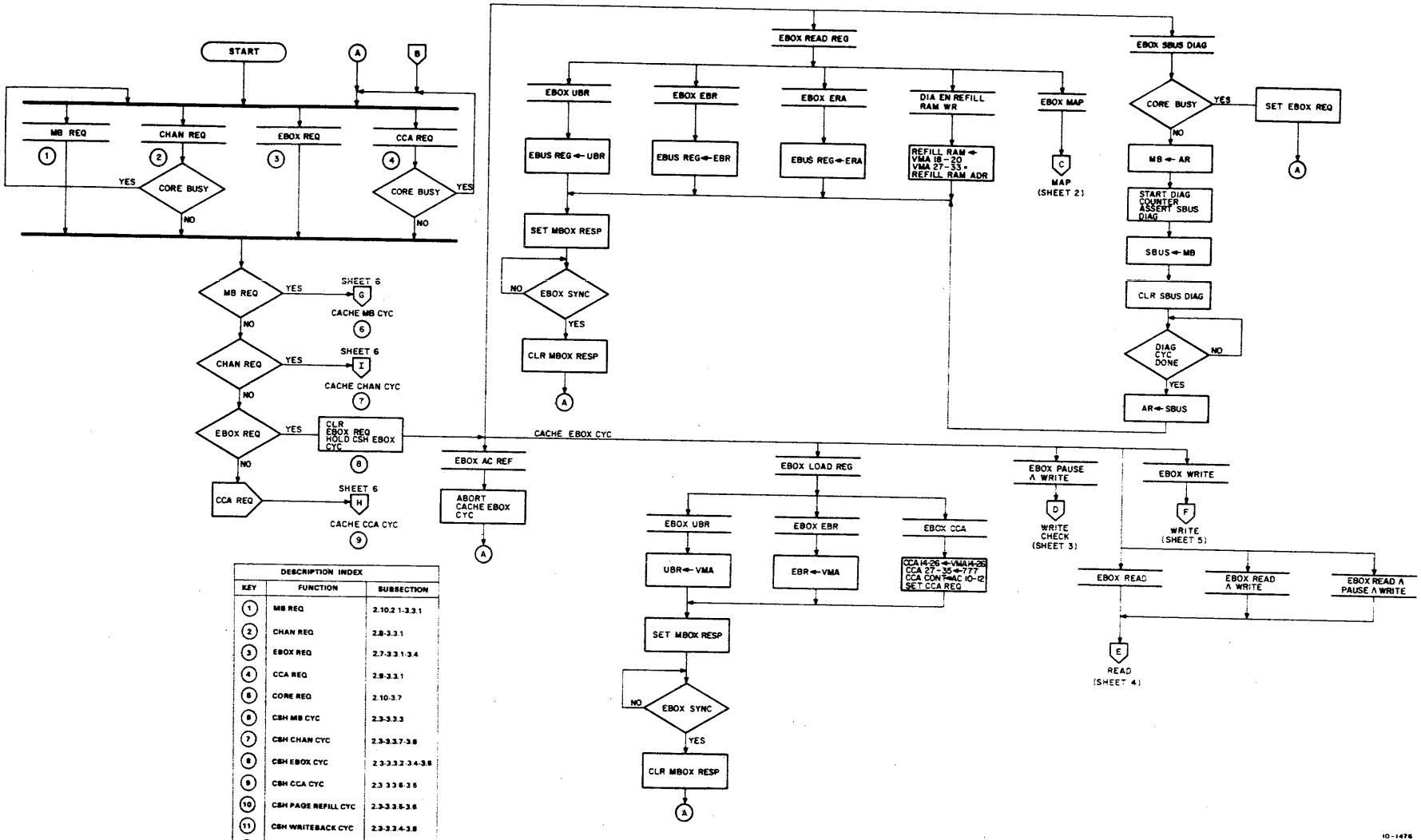
Table 2-4 Register Reference Requests

Register Operation	EBOX REQUEST QUALIFIERS																Remarks
	CLK EBOX REQ	APR EBOX LOAD REG	APR EBOX READ REG	APR EBOX UBR	APR EBOX EBR	APR EBOX CCA	APR EBOX ERA	IR AC 10-12	MCL EBOX MAP	APR EN REFILL RAM WR	CLK PT DIR WR	CLK PT WR	APR WR PT SEL 0-1	DIAG READ FUNCT	DIAG LOAD FUNCT	DIAG 04-06	
Load UBR	X	X		X													VMA contains address data
Load EBR	X	X			X												VMA contains address data
Load CCA	X	X				X		X									VMA contains address data
Load REFILL RAM	X		X							X							VMA contains address and data
Read UBR	X		X	X													Contents of UBR is transferred to EBUS REG
Read EBR	X		X		X												Contents of EBR is transferred to EBUS REG
Read CCA	X		X			X											Contents of CCA is transferred to EBUS REG
Read ERA	X		X				X										Contents of ERA is transferred to EBUS REG
Read PT	X		X						X								Contents of PT is transferred to EBUS REG
Load MEM TO C Diag Reg															X	X	EBUS D30-35 carries the data to be loaded
Write PT Directory											X		X				VMA 13-17 contains the section No. to be written
Write PT												X	X				VMA 18-26 contains the page No. to be written
Read Diag Registers														X		X	DIAG 04-06 carries the register No. to be read
Read EBus Register														X		7_8	Contents of EBUS REG is transferred to AR via EBUS data lines

Table 2-5 Memory Reference Requests

Memory Operation	EBOX REQUEST QUALIFIERS																Remarks
	CLK EBOX REQ	MCL VMA READ	MCL VMA PAUSE	MCL VMA WRITE	MCL VMA USER	MCL EBOX MAY BE PAGED	CON KI PAGING MODE	MCL PAGE ILLEGAL ENTRY	MCL PAGE TEST PRIVATE	MCL PAGE ADDRESS COND	MCL PAGE UEBR REF	MCL VMA EPT	MCL VMA UPT	MCL EBOX CACHE	CON CACHE LOOK EN	APR EBOX SBUS DIAG	
Read EPT	X	X									X	X		*	*		VMA contains address
Read UPT	X	X									X		X	*	*		VMA contains address
Read Instructions and Data	X	X			*	*	*	*	*	*				*	*		VMA contains address
Write EPT	X			X							X	X		*	*		VMA contains address
Write UPT	X			X							X		X	*	*		VMA contains address
Write Instructions and Data	X			X	*	*	*	*	*	*				*	*		VMA contains address
Write Check	X		X	X	*	X											VMA contains address
Read Modify Write (1st)	X	X	X	X	*	*	*	*	*	*				*	*		VMA contains address
Read Modify Write (2nd)	X			X	*	*	*	*	*	*				*	*		VMA contains address
SBUS Diag Cyc	X															X	AR contains control word

*These qualifiers may be true or not true depending on the specific type of request the EBox decides to make.



DESCRIPTION INDEX		
KEY	FUNCTION	SUBSECTION
1	MB REQ	2.10.2.1-3.3.1
2	CHAN REQ	2.8-3.3.1
3	EBOX REQ	2.7-3.3.1-3.4
4	CCA REQ	2.9-3.3.1
5	CORE REQ	2.10-3.7
6	CSH MB CYC	2.3-3.3.3
7	CSH CHAN CYC	2.3-3.3.7-3.8
8	CSH EBOX CYC	2.3-3.3.2-3.4-3.8
9	CSH CCA CYC	2.3-3.3.6-3.5
10	CSH PAGE REFILL CYC	2.3-3.3.5-3.6
11	CSH WRITEBACK CYC	2.3-3.3.4-3.8
12	CORE CYC	2.4-2.10.2-3.7

NOTE: Also refer to Figures 2-13 thru 2-17

Figure 2-7 Cache Cycle Control, Functional Flow Diagram (Sheet 1 of 6)

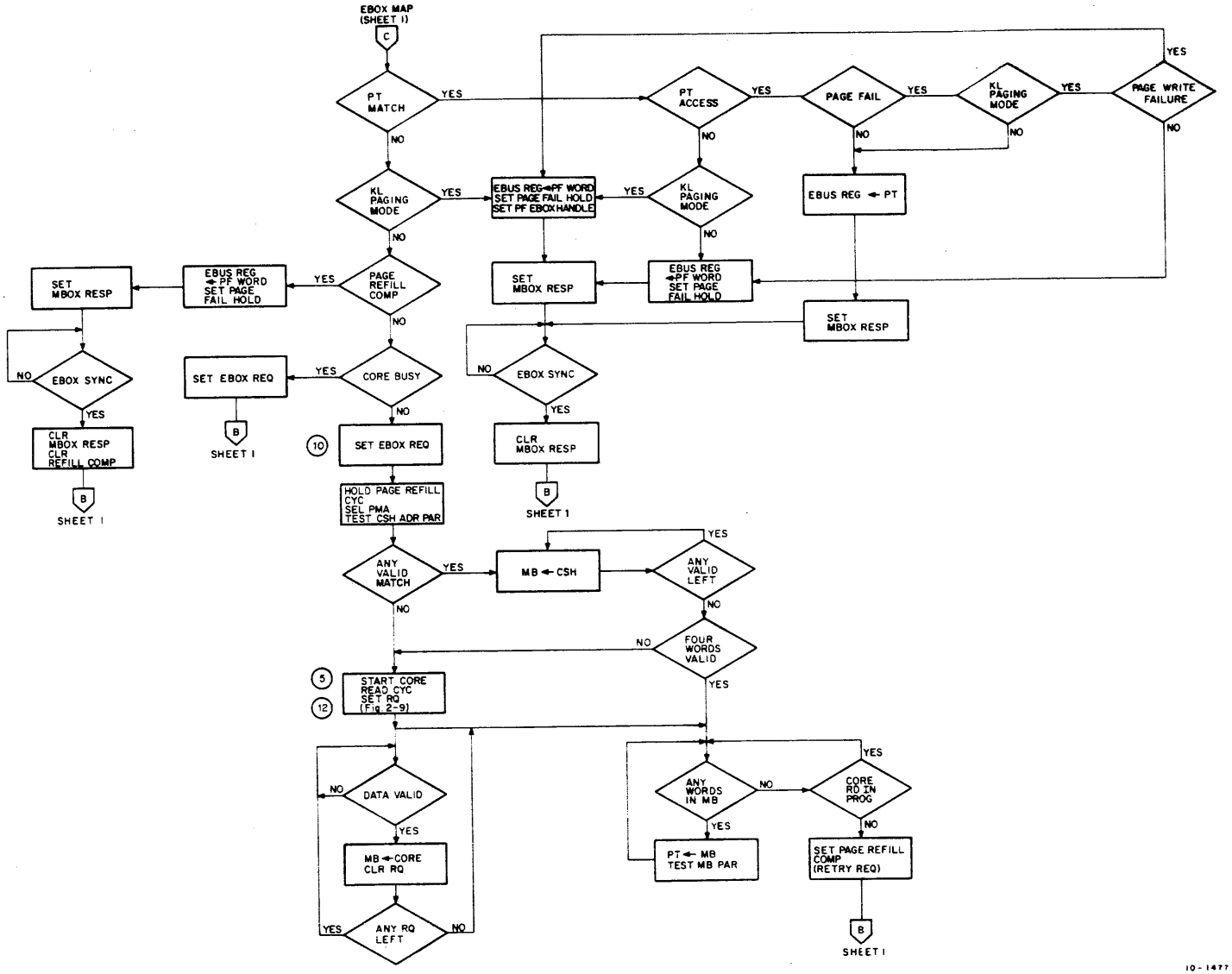


Figure 2-7 Cache Cycle Control, Functional Flow Diagram (Sheet 2 of 6)

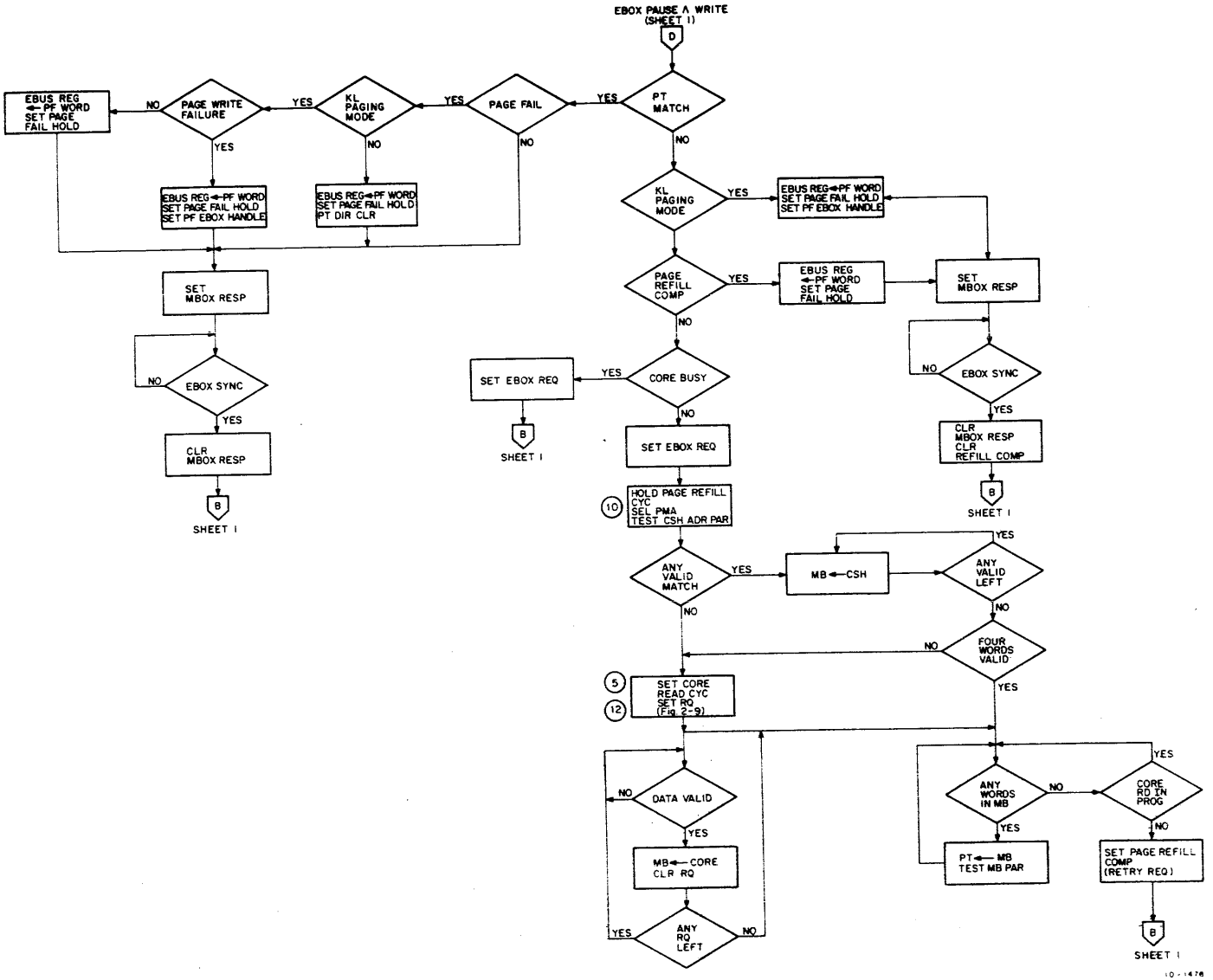


Figure 2-7 Cache Cycle Control, Functional Flow Diagram (Sheet 3 of 6)

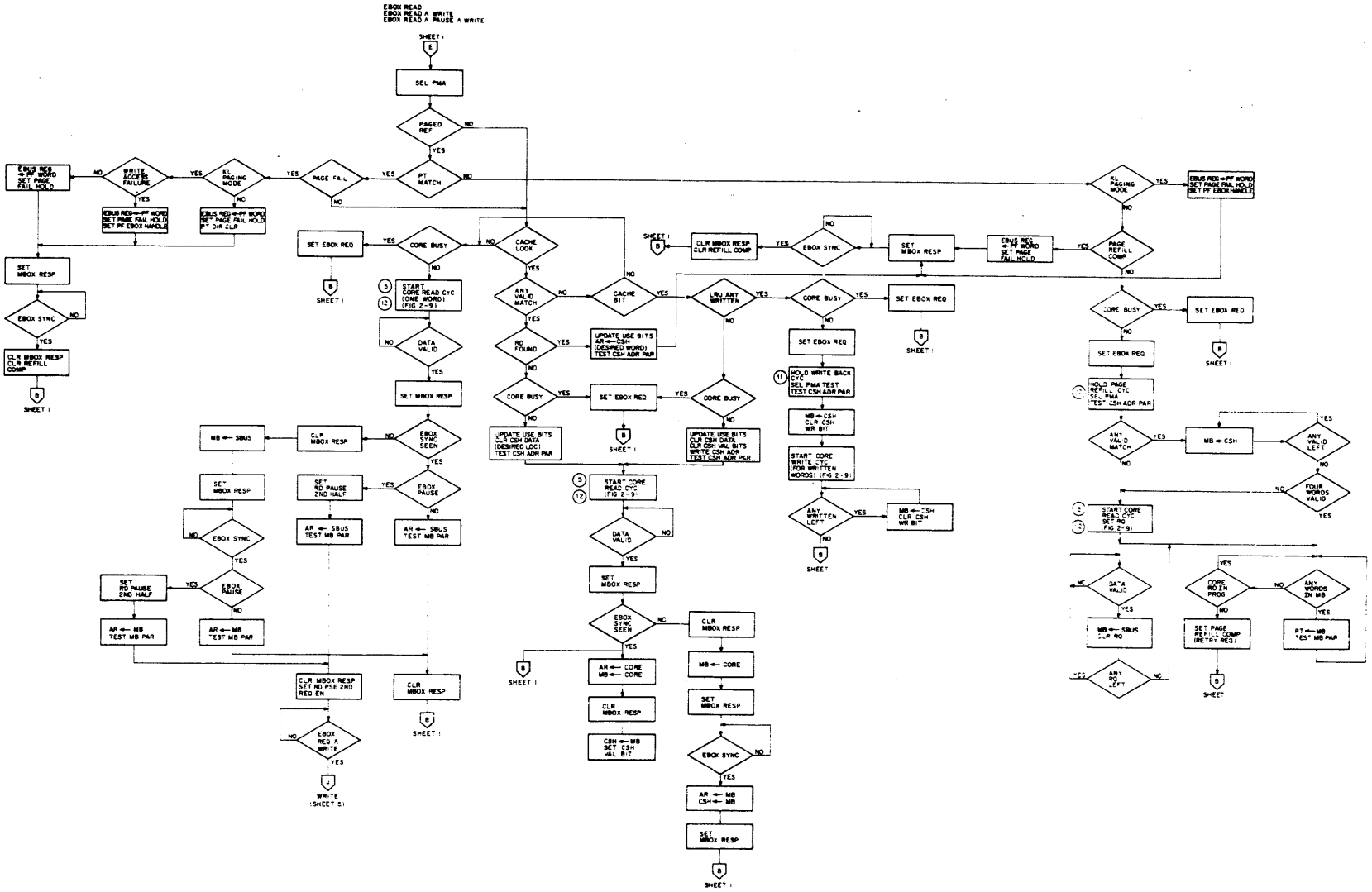


Figure 2-7 Cache Cycle Control, Functional Flow Diagram
(Sheet 4 of 6)

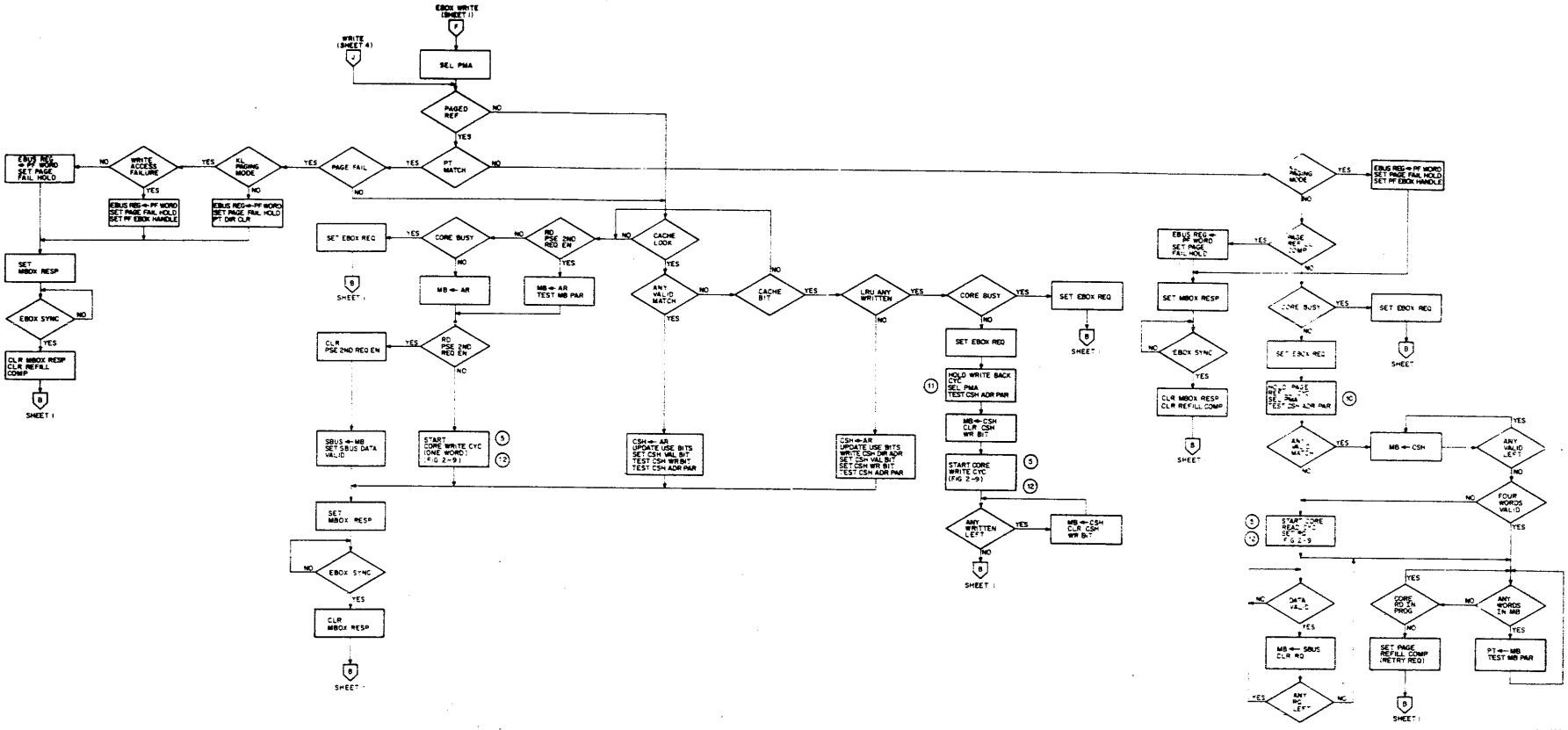


Figure 2-7 Cache Cycle Control, Functional Flow Diagram (Sheet 5 of 6)

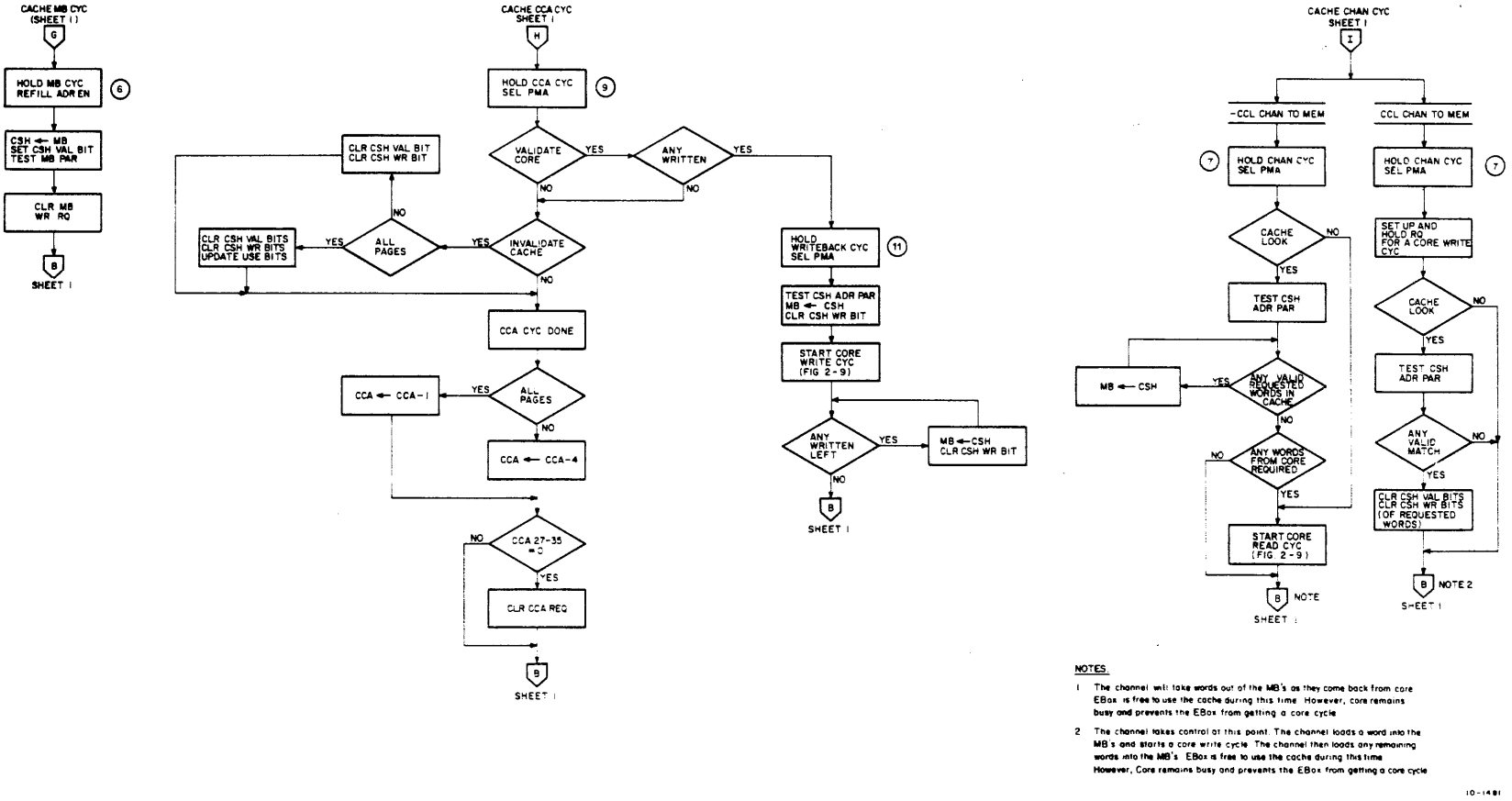


Figure 2-7 Cache Cycle Control,
Functional Flow Diagram
(Sheet 6 of 6)

The following is a summary of why and/or when these registers can be loaded or read by the EBox.

a. The cache refill RAM is loaded with the refill algorithm during system initialization. The refill algorithm specifies the extent to which the cache is used by using all or bypassing some cache quarters.

b. The EBR is loaded with the base address of the EPT during system initialization. The register can be read for diagnostic purposes.

c. The UBR is loaded with the base address of the UBR every time another user process is started. The register can be read for diagnostic purposes.

d. The CCA register (including the request qualifier bits) is loaded to invalidate the cache and/or validate core. One page or the entire address space in the cache can be specified for this operation. The entire address space in the cache is invalidated during system initialization. Core may be validated for various reasons. One case where core must be validated is prior to initiating a channel read operation, when external channels (DF10 or DAS33) are used. When external channels are implemented, one or more pages may also have to be invalidated in the cache before a channel write operation is executed. The CCA register can be read for diagnostic purposes.

e. The ERA register is loaded with the current address and error source code by the MBox automatically whenever the MBox senses a parity or a non-existent memory (NXM) error. The register is read by the EBox to determine the cause of the error.

f. The content of the addressed page table location is read by the EBox when a MAP instruction is executed. This instruction is executed to obtain the physical address of the I/O buffer when building the channel command list. This address is placed into the address field (ADR) of the Data Transfer CCW.

g. The diagnostic register is loaded with a code from the EBus data lines to adjust the MBox data path during system initialization and for diagnostic purposes. The data path can be adjusted to read data from MB0, SBUS, addressed cache data location, the CBus via the CH REG or the AR.

h. The section address (VMA 13-17) is written into the addressed page table location when the EBox writes an entry into the page table (KL paging mode only). The EBox will also write the directory to clear all entries when switching users.

i. The physical page address (AR00-17) is written into the addressed page table location after it is fetched from the core page table to update the hardware table.

j. The 15 diagnostic registers in the MBox are read by the front-end processor for diagnostic purposes.

k. The EBus register is a holding register for the read register function (APR EBOX READ REG asserted).

2.7.4 Memory References

The EBox can issue requests to read and to write memory. The EBox can request to read or write the executive and user process tables and user or executive paged and unpaged memory. The EBox will also specify whether the cache is to be used in servicing the memory request. When the MBox starts processing a memory request, it automatically forms the correct physical memory adress in response to the request qualifiers presented with the request. If the EBox requested a reference to paged memory, it also automatically reads and/or write-checks the referenced page. If the page check fails, the MBox informs the EBox of this condition by asserting PAGE FAIL HOLD.

To read or write memory, the EBox must set up the address in the VMA and assert a specific set of qualifier signals along with CLK EBOX REQ for each type of reference. When writing memory, the EBox must also move the word to be written into the AR when issuing the request. All memory operations the EBox is capable of requesting and the required request qualifiers are given in Table 2-5. Flows for each type of memory reference operation are presented in Figure 2-7.

2.7.4.1 Read Memory – To read memory, the EBox asserts CLK EBOX REQ and the appropriate EBox request qualifiers. When the request is granted, a cache EBox cycle is executed by the cache cycle control to service the request. In executing a cache EBox cycle for a memory read request, the following operations are performed by the cache cycle control:

a. The required physical memory address, as specified by the EBox request qualifiers, is selected (PMA 14–35 SEL). The virtual section and page addresses will be replaced with the contents of the EBR, UBR, or the page table, as needed. The resultant address is used to address the cache and, if necessary, to address core memory if a core read cycle is required.

b. If the EBox issued a request to read paged memory, the contents of the pager are checked to see if the reference is permitted. For KI paging mode references, the cache cycle control will also execute a page refill cycle automatically to update the pager, if required.

c. If so specified by the EBox, the cache is checked to see if the desired word is in the cache.

d. A core read cycle is initiated if:

   1. The requested word is not found but some of the words of the respective quadword group are in the cache.

   2. None of the words of the associated quadword group are in the cache and the Least Recently Used (LRU) cache block does not contain any written words. If written words from another page were found in the LRU cache, the cache cycle control will initiate a core write cycle to write back the written words to core before starting the core read cycle.

   3. The EBox did not specify the cache to be used or the cache does not exist (is not implemented).

2.7.4.2 Write Memory – To write memory, the EBox asserts CLK EBOX REQ and the appropriate EBox request qualifiers. When the request is granted, a cache EBox cycle is executed by the cache cycle control to service the request. In executing a cache EBox cycle for a memory write request, the following operations are performed by the cache cycle control:

a. The required physical memory address, as specified by the EBox request qualifiers, is selected (PMA 14–35 SEL). The virtual section and page addresses will be replaced with the contents of the EBR, UBR, or the page table, as needed. The resultant address is used to address the Cache and, if necessary, to address core memory if a core write cycle is required.

b. If the EBox issued a request to write paged memory, the contents of the pager are checked to see if the reference is permitted. For KI paging mode references, the cache cycle control will also execute a page refill cycle automatically to update the hardware page table, if required.

c. If so specified by the EBox, the word is written into the cache; otherwise, a core write cycle is initiated to move the word to core.

d. If the addressed cache line does not contain any words from the associated quadword group and the LRU cache contains written words from another page, the cache cycle control will initiate a core write cycle to write back the written words before writing the cache.

2.7.4.3 Read and Write-Check Paged Memory – To both read and write-check a paged memory location, the EBox asserts CLK EBOX REQ and the appropriate request qualifiers. When the request is granted, a cache EBox cycle is executed by the cache cycle control to write-check (Subsection 2.7.4.4) and read (Subsection 2.7.4.1) the addressed memory location.

2.7.4.4 Write-Check Paged Memory – To write-check a paged memory location, the EBox asserts CLK EBOX REQ and the appropriate request qualifiers. When the request is granted, a cache EBox cycle is executed by the cache cycle control to service the request. In executing a cache EBox cycle for a write-check request, the following operations are performed by the cache cycle control:

a. The pager is checked to see if it contains a valid entry and if the page is writable.

b. If the pager contains a valid entry and the page is writable, the MBox simply responds in the normal manner.

c. If the page is not writable, the Page Fail word is loaded into the EBus register and the EBox is notified that a page fail condition was sensed.

d. If the pager does not contain a valid entry and the EBox specifies the KI paging mode, the cache cycle control will automatically execute a page refill cycle to update the pager.

e. If the KL paging mode was specified, the MBox will notify the EBox to initiate the refill cycle.

2.7.4.5 Read-Modify-Write Memory – To read, modify and write memory, the EBox asserts CLK EBOX REQ and the appropriate request qualifiers for each of the read and the write portions of the operation. This operation is the same as requesting a separate read and a separate write operation if the cache is specified for use. If the cache is not specified by the EBox, then the cache cycle control, the core cycle control, and core memory wait for the EBox to request the second half of the operation.

2.7.4.6 SBus Diagnostic Cycle – An SBus diagnostic cycle is issued by the EBox to initialize core memory and to read core memory status information. To issue an SBus diagnostic cycle, the EBox moves a diagnostic control word into the AR and asserts CLK EBOX REQ and APR EBOX SBUS DIAG. When the request is granted by the MBox, a cache EBox cycle is executed by the cache cycle control to service the request. In executing the cache EBox cycle for an SBus diagnostic cycle request to read core memory status, the following operations are performed by the cache cycle control:

a. The control word is moved from the AR to the MB.
b. SBUS DIAG is asserted.
c. The contents of the register specified by the control word are transferred to the AR.

The control word transferred to the MB is moved to the core memory system to select a controller and the function to be performed. The core memory system, in response, will transfer the status of the selected function to the AR.

2.8 CHANNEL REQUESTS

Requests are issued by the channel control to read and to write memory after a channel is started (Subsection 2.11). Request qualifiers are used in issuing the request to specify precisely what type of service is desired by the requesting channel. To write, CCL CHAN TO MEM is asserted; to read CCL CHAN TO MEM is negated. If a channel needs to fetch a CCW from the EPT, or needs to store the status in the EPT, then the channel will assert CCL CHAN EPT to qualify the request. After issuing a request, the requesting channel waits for a cache channel cycle to be initiated by the cache cycle control to check the cache and/or start a core cycle. When the cache channel cycle is started, the channel assumes direct control of the MBs to move data in or out. In the case of channel write operations, the channel will load the MBs and start a core write cycle after the first word is loaded. For a channel write operation, the cache channel cycle is executed only to invalidate any valid entries in the cache. In the case of channel read operations, the channel specifies which words are needed and waits for a cache channel cycle to transfer any valid words in the cache to the appropriate MBs and to initiate a core read cycle for those words that are not in the cache. After a core read cycle is started (a core read cycle is started only if all the requested words are not in the cache), the channel continues to wait for the words to come in from core. As each word is placed into the appropriate MB by the cache cycle control and/or the core control, the channel moves the word into the channel data buffer (CH BUF) by selecting the appropriate MB. Words are moved into the CH BUF only in ascending order, starting with word zero.

2.8.1 Channel/Cache Interface Summary

A summary of the CHAN/CSH interface is presented in Table 2-6. The interface signals are grouped into sets according to their function. The notations in parentheses are field maintenance print set prefixes that specify the source of the signals.

Table 2-6 CHAN/CSH Interface Summary

Signal	Description
A. Control Commands	
CCL CHAN REQ (CCL3)	Issued by the channels to request service.
CCL HOLD MEM (CCL2)	Asserted by the channels if the channels have requests backed up. By asserting this signal, the channel is assured the next core cycle by preventing an EBox Request from initiating a core cycle.
CSH CHAN CYC A (MBX4)	Asserted when the Cache cycle control starts processing the Channel request. This signal informs the channel that it can start writing the MBs in the case of channel write operation or start looking for words ready to be taken from the MBs in the case of channel read operations.
CCL START MEM (CCL4)	Asserted by the channel during channel write operations after the first word is loaded into the MBs. Subsequent words are moved into the MBs at four clock-tick intervals assuring the core control has a word to move to core when it gets ready. The core control moves words to core at six clock-tick intervals. During channel read operations, the Cache cycle control starts the core cycle when it is ready.
CCL CH MB SEL 1-2 (CCL4)	The channel places a two bit code on these lines to select the correct MB to be loaded during channel write operations or read during channel read operations.

Table 2-6 CHAN/CSH Interface Summary (Cont)

Signal	Description
CCL CH LOAD MB (CCL4)	Asserted by the channel to load the selected MB during channel write operations.
MB0–3 HOLD IN (MBX6)	Asserted by the Cache cycle control and/or the core cycle control during a channel read operation to load the MBs and to inform the channel that the corresponding word is ready to be taken.
CCL CH TEST MB PAR (CCL4)	Asserted by the channel to check the parity of the selected word before it is taken from the MB during channel read operations.
B. Request Qualifiers CCL CHAN TO MEM (CCL4)	Asserted by the channel to specify a channel write operation is to be executed. When negated, a channel read operation is executed.
CCW WD0–3 RQ (CCW4)	These four signals are asserted by the channel to specify the words to be read or written.
CCL CHAN EPT (CCL3)	Asserted by the channel to read or write the Executive Process Table. The EPT is read to fetch the initial CCW and is written to store the channel status at the end of a transfer. The Cache cycle control will automatically select the correct address for referencing the EPT.
C. Error Reporting Commands CHAN PAR ERR (MBZ4)	Asserted for one clock period when the MB parity check fails during a channel read or channel write operation. During channel write operations, parity is checked when the channel asserts CCL CH TEST MB PAR and during channel read operations, parity is checked when the Cache cycle control or the core cycle control loads the word into the MB. This signal informs the channel that a data parity error occurred during the transfer.
CHAN ADR PAR ERR (MBZ4)	Asserted for one clock period when the SBus address parity check fails during channel read or channel write operations.
CHAN NXM ERR (MBZ3)	Asserted for one clock period when the NXM counter in the MBox times out. This counter times out if one of the ACKN pulses for the requested words is not received from the memory. (Subsection 2.14.3).
D. Address CCW CHA 14–35 (CCW2)	Physical memory address from channel.
E. Data	Data Buffer and path is an integral part of the MB modules.
F. Clocks	Clocks are distributed to the channels from the EBox (Table 2-3, E/M Interface Summary.

2.8.2 Request Dialogue

The channels issue requests to the cache cycle control for core cycles by asserting CCL CHAN REQ and CCL HOLD MEM (Figure 2-8) during an initial MB RAM cycle. Along with asserting CCL CHAN REQ and CCL HOLD MEM, the channels also set up the channel address (CHA) and the request qualifiers. The request qualifiers are:

a. CCL CHAN TO MEM
b. CCL CHAN WD0–3RQ
c. CCL CHAN EPT

These signals remain valid until the request has been processed to completion. If another request is ready to be processed, CCL CHAN REQ and CCL HOLD MEM remain asserted while the address and request qualifiers are adjusted to specify the next request.

When the cache cycle control starts to process a request, the cache cycle control asserts CSH CHAN CYC. This signal informs the channel that it can start moving words from the CHAN BUF to the MBs. in the case of channel data write operations, or can start looking for words that are ready to be moved out of the MBs into the CH BUF, in the case of channel data read operations.

2.8.2.1 Channel Read Operations – Two types of read requests can be issued by the channels:

a. Read a single word from the EPT. The EPT contains eight locations for storing the initial CCW. One location is assigned to each channel.

b. Read one, two, three, or four words (instructions and data) from physical core memory.

To read the initial CCW from the EPT, the channel issues and qualifies the request as follows:

a. Assert CCL CHAN REQ.
b. Clear CCL CHAN TO MEM.
c. Assert CCL CHAN EPT.
d. Assert CCW WD0 RQ.

NOTE
Word 0 is requested because the initial CCW is stored in location 0 of a quadword group.

e. Set up CCL CH MB SEL 1–2 lines to point to MB0.
f. Assert CCL HOLD MEM.
g. Hold CCW CHA 14–35.

The channel then waits for a cache cycle. When the cache cycle is started, the correct address is formed by replacing CCW CHA 14–26 with the contents of the EBR. This address is then used to look in the cache; if the word is not in the cache, the word is read from core (refer to cache channel cycle description). In either case, the word is moved into MB0. The channel recognizes that MB0 was loaded when MB0 HOLD IN was negated for one clock tick. The channel will then move the word from MB0 to the CCW BUF and cause MB parity to be checked.

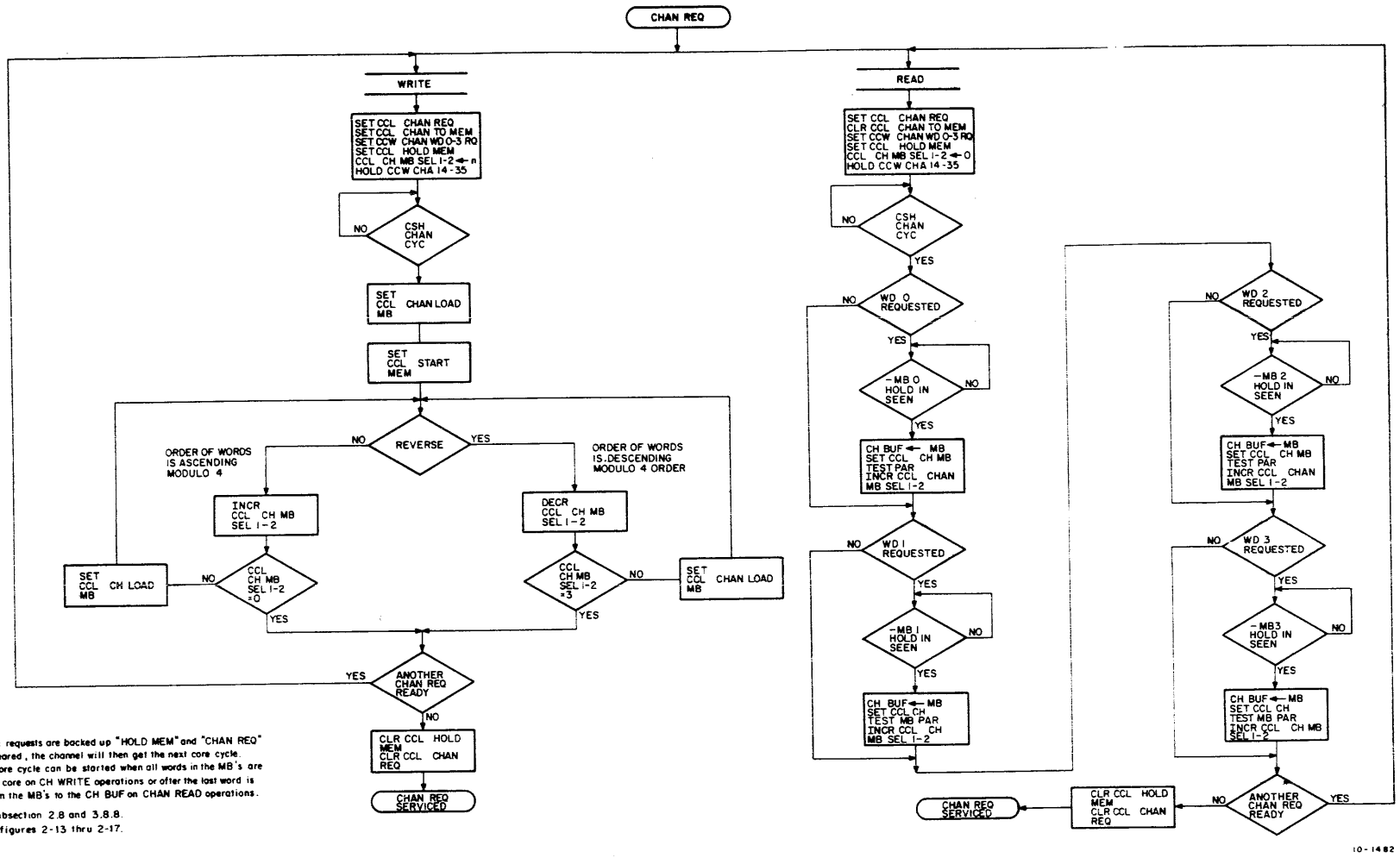


Figure 2-8 Channel Request Dialogue, Simplified Flow Diagram (Data Read and Write)

To read data and instructions from physical core memory, the channel issues and qualifies the request as follows:

a. Assert CCL CHAN REQ.

b. Clear CCL CHAN TO MEM.

c. Clear CCL CHAN EPT.

d. Set up CCW WD0–WD3 RQ lines to indicate which words are to be read.

e. Set up CCL CH MB SEL 1–2 lines to point to the MB that corresponds to the lowest order word requested.

f. Assert CCL HOLD MEM.

g. Hold CCW CHA 14–35.

The channel then waits for a cache cycle. When the cache cycle is started, the channel address (CCW CHA 14–35) is used to look in the cache, and if all the requested words are not in the cache, to read those words from core. In either case, the requested words are moved into the MBs. The channel recognizes that an MB is loaded when MB0, 1, 2, or 3 HOLD IN is negated for one clock tick. The channel will start moving the words to the CH BUF as soon as the lowest order requested word is placed into the corresponding MB. Subsequent words are moved from the MBs to the CH BUF in ascending order. As each word is transferred, its parity is also checked in the MB.

2.8.2.2 Channel Write Operations – Two types of write requests can be issued by the channels:

a. Write two words into the EPT. The EPT contains 16 locations for storing channel status information. Two locations are assigned to each channel.

b. Write one, two, three, or four words (data and instruction) into physical core memory.

NOTE

These words may have been read from a magnetic tape drive that is capable of reading forward and reverse.

To write the two status words into the EPT, the channel issues and qualifies the request as follows:

a. Assert CCL CHAN REQ.
b. Assert CCL CHAN TO MEM.
c. Assert CCL CHAN EPT.
d. Assert CCW WD1 and WD2 REQ.

NOTE

Word 1 and Word 2 are specified because the status words are stored in locations 1 and 2 of a quadword group.

e. Set up CCL CH MB SEL 1–2 lines to point to MB1.
f. Assert CCL HOLD MEM.
g. Hold CCW CHA 14–35.

The channel then waits for a cache cycle. When the cache cycle is started, the correct address is formed by replacing CHA 14–26 with the contents of the EBR. This address is then used to write the words to core after they are moved to the MBs. The cache is also checked to see if there is a copy of the referenced EPT locations in the cache, if CON CACHE LOOK EN is set, and if the cache is implemented. If there is, this copy is invalidated because it is assumed to be an old copy. After the first word is moved into the MB, the channel initiates a core write cycle to move the word to core. The second word is moved into its MB four clock ticks after the first word, in time for the core control.

To write data and instructions into physical core memory, the channel issues and qualifies the request as follows:

a. Assert CCL CHAN REQ.

b. Assert CCL CHAN TO MEM.

c. Clear CCL CHAN EPT.

d. Set up CCW WD0–WD3 RQ lines to indicate which words are to be written.

e. Set up CCL CH MB SEL 1–2 lines to point to the MB that corresponds to the first word to be transferred by the channel.

NOTE

If the words were read from a magnetic tape drive operating in the forward mode, the words will be transferred in ascending modulo four order. However, if the drive was operating in the reverse mode, the words will be transferred in descending modulo four order.

f. Assert CCL HOLD MEM.

g. Hold CCW CHA 14–35.

The channel then waits for a cache channel cycle. When the cache cycle is started, the channel address (CCW CHA 14–35) is used to write the words into core after they are moved into the MBs. The cache is also checked to see if there is a copy of the referenced memory locations in the cache. If there is, this copy is invalidated since it is assumed that this must be an old copy. After the first word (lowest numbered word) is moved into the MB, the channel initiates a core write cycle to move the word to core. Subsequent words are moved into the MBs at four clock-tick intervals so that the words will be available for transfer to core. The core cycle control moves a word to core every six clock ticks once a core cycle is started.

2.9 CCA REQUESTS

Requests are issued by the cache clearer control to invalidate the cache and/or validate core. The cache clearer control is activated by the EBox when it executes a Sweep instruction. While executing a Sweep instruction, the EBox issues a request to load the CCA register. This request loads the CCA register and activates the CCA control by latching CCA REQ and loading a 3-bit request qualifier register. After this operation is done, the CCA control will issue requests, accompanied with the preset request qualifiers, until the Sweep operation is completed, at which time, the CCA REQ latch is cleared. The preset request qualifiers include:

a. CSH CCA ONE PAGE
b. CSH CCA VAL CORE
c. CSH CCA INVAL CSH

Whenever an MB, CHAN, or EBox request is not pending, a cache cycle is executed for the cache clearer control to:

a. Sweep one page, or

b. Sweep entire cache

In either case, depending on the request qualifiers, the cache may be cleared and/or core may be validated.

When sweeping one page, each line of the cache is checked to see if there is a valid entry. An entry is valid if the address in the cache directory matches the high-order 14 address bits in the CCA register and one or more valid bits are set. If there is a valid entry in the addressed cache line then the entry is invalidated and/or a core cycle is started to move any written words to core. After this operation is done, the low-order nine cache clearer address bits in the CCA register are decremented by four to address the next cache line in preparation for the next cache CCA cycle.

When sweeping the entire cache, each of the four cache blocks in each cache line is checked to see if there is a valid entry. An entry is valid if one or more valid bits in the addressed cache block are set. The high-order 14 cache clearer address bits are not required when sweeping the entire cache because every entry in the cache, regardless of its address, is subject to the sweep operation. If there is a valid entry in the addressed cache block, the entry is invalidated and/or a core cycle is started to move any written words to core. After this operation is done, the low-order nine cache clearer address bits are decremented by one to address the next cache block in preparation for the next cache CCA cycle.

After the cache clearer control has stepped through the entire cache, CCA REQ is cleared to inform the EBox that the Sweep operation is done.

2.10 CORE REQUESTS

Core requests to read or write main memory are issued by the core cycle in response to a start signal and appropriate request qualifiers from the cache cycle control or from the channel control. All control signals, the address, and the data are transferred between the MBox and the main memory system via the SBus. Both the channel and the cache cycle control can initiate a core cycle to read or write up to four words at a time. Once the core cycle control is set up by the channel or cache cycle control, the core cycle control will execute the requested operation to completion, independently.

NOTE

SBus diagnostic cycles are executed by the cache cycle control not the core cycle control (Subsection 2.7.4.6).

2.10.1 SBus Summary

A summary of the SBus is presented in Table 2-7. The SBus signals are grouped into sets, according to their function. The notations in parentheses are field maintenance print set prefixes that specify the source of the signals.

2.10.2 Request Dialogue

The core cycle control starts a core cycle by asserting SBUS START A or B, asserting the appropriate SBus request qualifiers and holding the physical memory address (Figure 2-9). The request qualifiers are:

a. SBUS RQ 0–3
b. SBUS RD RQ
c. SBUS WR RQ

Table 2-7 SBus Summary

Signal	Description
A. Control Commands START A/B (MT01)	START A or START B is asserted by the core control to start a core cycle.
ACKN PULSE A/B (SBUS 0/1)	ACKN PULSE A or ACKN PULSE B is asserted by the core memory system to acknowledge the requests.
DATA VALID A/B (SBUS 0/1-MT01)	DATA VALID A or DATA VALID B is asserted by the core memory system when a word is placed on the data lines of the SBus. Also asserted by the MBox during the write portion of a Read-Pause-Write cycle.
	NOTE **The above control signals are phase-locked with the leading or trailing edges (Phase A or B) of the SBus clock to minimize bus latency.**
DIAG (MT01)	Asserted by the Cache cycle control to start a diagnostic cycle.
B. Request Qualifiers RQ0–3 (MT01)	These four signals are asserted by the core control to specify the words to be read or written.
RD RQ (MT01)	Asserted by the core cycle control to specify a core read cycle is to be executed.
WR RQ (MT01)	Asserted by the core cycle control to specify a core write cycle is to be executed.
	NOTE **During a Read-Pause-Write Cycle both RD RQ and WR RQ are asserted.**
C. Error Reporting Commands ADR PAR ERR (SBUS 0/1)	Asserted by the core memory system when an address parity error is sensed.
ERROR (SBUS 0/1)	Asserted by the core memory system when a data parity error is sensed. Data parity is checked on both read and write cycles.
D. Address ADR 14–35 (MT04)	Physical address for memory system.
ADR PAR (MT01)	Address and Request Qualifier parity for memory system.
E. Data D00–35 (SBUS 0/1-MT02-3)	Bidirectional data path between MBox and core memory system.

Table 2-7 SBus Summary (Cont)

Signal	Description
E. Data (Cont) DATA PAR (SBUS 0/1-MT05)	Bidirectional data parity line between MBox and core memory system.
F. Clocks INT CLK (MT01)	Clock for internal memory system (MA/MB20);
EXT CLK (MT01)	Clock for external memory system (DMA20).

These signals remain valid until all requested words have been acknowledged. All further core requests from the channels or cache cycle control will be deferred until core is freed at the completion of the core cycle in progress. Three types of core cycles can be initiated by the core cycle control:

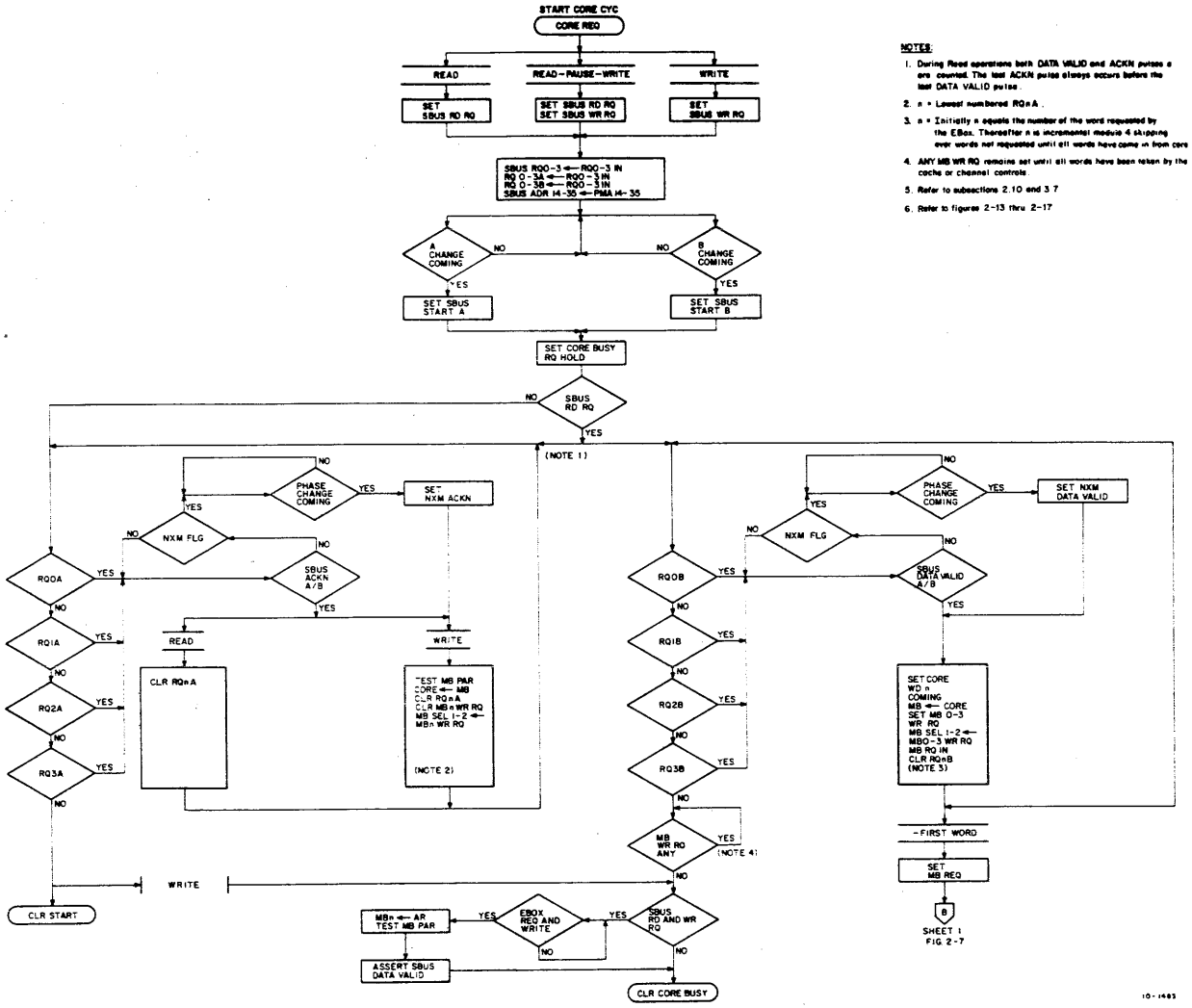
a. Read cycle
b. Write cycle
c. Read-Pause-Write cycle

The read-pause-write cycle will only be initiated in response to an EBox request for which the cache is not to be used.

2.10.2.1 Core Read Cycle – To read from core, the core cycle control, in response to a command from the cache cycle control or the channel control, issues and qualifies the request as follows:

a. Assert SBUS START A or B.
b. Assert SBUS RD RQ.
c. Assert SBUS RQ 0–3.
d. Hold SBUS ADR 14–35.

At the same time the core cycle control issues the request, CORE BUSY is set and the acknowledge and data valid pulse counters are initialized (Subsection 3.7). The counters keep track of the requested words coming back from core by counting the SBUS ACKN and DATA VALID pulses. After setting up the request, the core cycle control waits for the words to come in from core. As each request is acknowledged and each word comes in from core, the associated requests held in the acknowledge and data valid pulse counters are cleared and the word is moved into the appropriate MB. The first word may also be moved to the AR in the EBox. When all requested words have been acknowledged, SBUS START A or B is cleared. CORE BUSY remains set until the channel or the cache cycle control, depending on which control requested the core cycle, moves the words out of the MBs. If a core read cycle for more than one word was started by the cache cycle control in response to an EBox request, the core cycle control will issue MB requests to the cache cycle control for all but the first word. As MB requests are granted by the cache cycle control, the words that were moved into the MBs by the core cycle control are moved in the cache. If the core read cycle was initiated to satisfy a channel request, the channel will take the word in ascending modulo four order after they have come in from core.



NOTES:

1. During Read operations both DATA VALID and ACKN pulses are counted. The last ACKN pulse always occurs before the last DATA VALID pulse.
2. n = Lowest numbered RQnA.
3. n = Initially n equals the number of the word requested by the EBox. Thereafter n is incremented modulo 4 skipping over words not requested until all words have come in from core.
4. ANY MB WR RQ remains set until all words have been taken by the cache or channel controls.
5. Refer to subsections 2.10 and 3.7.
6. Refer to figures 2-13 thru 2-17.

Figure 2-9 Core Control Cycle, Functional Flow Diagram

2.10.2.2 Core Write Cycle – To write into core, the core cycle control, in response to a command from the cache control or the channel control, issues and qualifies the request as follows:

a. Assert SBUS START A or B.
b. Assert SBUS WR RQ.
c. Assert SBUS RQ 0–3.
d. Hold SBUS ADR 14–35.

Before the core request is issued, the cache cycle control or the channel control will have moved the first word to be written into the appropriate MB.

At the same time, the core cycle control issues the request, it also sets CORE BUSY and initializes the acknowledge pulse counter. This counter keeps track of which words have been moved to core by counting the SBUS ACKN pulses. After setting up the request, the core cycle control waits for each word that is to be written to core to be acknowledged. As each word is acknowledged, the associated request held in the acknowledge pulse counter is cleared and the next word is placed on the SBus data lines by selecting the appropriate MB. When all words have been acknowledged, SBUS START A or B and CORE BUSY are cleared.

2.10.2.3 Core Read-Pause-Write Cycle – To read, modify, and write a core location without releasing core (PAUSE) between the read and write operation, the core cycle control, in response to a command from the cache cycle control, issues and qualifies the request as follows:

a. Assert SBUS START A or B.
b. Assert SBUS RD RQ.
c. Assert SBUS WR RQ.
d. Assert SBUS RQ 0–3.
e. Hold SBUS ADR 14–35.

At the same time the core cycle control issues the request, CORE BUSY is set and the acknowledge and data valid pulse counters are initialized. During this type of core cycle, only one word will be requested from core. Consequently, the acknowledge and data valid pulse counters will be cleared after the request is acknowledged and the first word comes in from core. When the word comes in from core, it is placed in the appropriate MB and is made available to the EBox AR. SBUS START A or B is also cleared at this time. However, CORE BUSY is not cleared until the EBox issues the write request and the word is on its way to core memory. When the EBox makes the write request, the word to be written is moved from the AR to the MB and SBUS DATA VALID is asserted to inform core memory that it is to write the word.

2.11 CBUS REQUESTS

The CBus is a synchronous bus system that connects the integral channel control logic of the MBox to a maximum of eight RH20 Massbus controllers. These controllers are selected (scanned) in such a way that the first four controllers (0–3) can handle a data transfer rate of approximately one 36-bit word per microsecond, while the second four controllers (4–7) handle a data transfer rate of half that speed.

The MBox is a logical unit that provides the path to the main memory subsystem for both the integral data channels and the EBox. Each Massbus controller can control up to eight mass-storage disk drives (fixed-head disks or moving-head disks) or up to eight TM02 or TM03 magtape controllers with each controller having up to four TU16 or TU45 drives. The purpose of the CBus is to provide a high-speed path between the MBox channel control logic and up to eight controllers for control and data information.

2.11.1 CBus Summary

A summary description of the CBus is given in Table 2-8. The notations in parentheses are field maintenance print set prefixes that specify the source of the signals.

Table 2-8 CBus Summary

Signal	Description
SEL 0–7 (TR05)	These eight radial lines are controlled by the channel control to select one Massbus controller at a time every four MBox clock ticks. The SELECT line of a controller defines the beginning of its four data transfer cycles (SELECT cycle, REQUEST cycle, WAIT cycle, and DATA cycle).
RESET (CBUS)	This signal may be asserted by a Massbus controller during its DATA cycle. The channel control logic, upon detecting this signal, will clear the control RAM location associated with the controller (channel) and will store the fact that reset has occurred.
START (TR05)	A Massbus controller will always begin a block transfer by asserting this line once during its DATA cycle. The line will be asserted only when CBUS READY is negated. The channel control logic will assert READY when it is prepared for data transfer.
CTOM (CBUS)	A Massbus controller begins a block transfer by asserting START for exactly one DATA cycle. The controller will inform the channel control logic during the same cycle of the direction of the block transfer by: a. Asserting CTOM for an input block transfer (Channel to Memory). b. Negating CTOM for an output block transfer (Memory to Channel).
READY (TR05)	The channel control logic will assert this line (during the DATA cycle only) after it detects a START signal sent by a Massbus controller and after the channel control logic is ready for data transfer. For an output block transfer, the channel control will have at least two words of data from memory (if WC > 2) before asserting READY. The READY signal, once asserted, will normally be negated only after sensing the DONE signal and after the channel control is prepared to start another block transfer operation. Errors will also cause READY to be negated.
REQUEST (CBUS)	A Massbus controller will assert REQUEST during its REQUEST cycle when: a. One of its data buffers is full (for an input block transfer operation). b. One of its data buffers is empty (for an output block transfer operation).

Table 2-8 CBus Summary (Cont)

Signal	Description
REQUEST (CBUS) (Cont)	A Massbus controller will not assert REQUEST if: a. READY line is not asserted by the channel control. b. ERROR line has been asserted by the channel control during the current block transfer. c. LAST WORD has been asserted by the channel control during the current block transfer. d. DONE has been asserted by the Massbus controller during the current block transfer. For an input data transfer, the Massbus controller will place data (throughout its DATA cycle) on the DATA lines and the channel control will strobe the DATA lines at the trailing edge of the same data cycle. For an output transfer, the above operation is reversed.
DONE (CBUS)	The Massbus controller will terminate a block transfer by asserting this signal once during its DATA cycle. No more data requests will be made after DONE is asserted. The channel control, after detecting DONE, will get ready for a new block transfer (empty the input data buffers, etc.). The error line can still be used to inform the Massbus controller that an error has been detected in the current block transfer as long as the READY line is not negated.
STORE (CBUS)	The Massbus controller will send STORE to the channel control once (at the same time the controller sends DONE) when: a. The current block transfer is terminated due to errors detected in the Massbus controller and/or b. The current block transfer command in the Massbus controller specifies that STORE be sent to the channel control at the conclusion of the block transfer. The channel control, upon detecting STORE, will write all status information associated with the controller into memory. The channel control will keep READY asserted until it is prepared to initiate another block transfer.
LAST WORD (TR05)	The channel control (for an output block transfer only) will assert this line (during the DATA cycle) one cycle after the last data word is sent to a controller. No more data requests will be made by the Massbus controller after detecting LAST WORD.

Figure 2-8 CBus Summary (Cont)

Signal	Description
ERROR (TR05)	The channel control will assert this line (during DATA cycle only) to inform the controller that the current input/output block transfer must not continue due to error conditions detected in the channel control. The Massbus controller, upon sensing the ERROR signal, will terminate the block transfer by not making any more data requests and will assert DONE exactly once during a subsequent DATA cycle. The ERROR line will be negated before the channel control negates the READY line. If the ERROR line is detected after the READY line is negated, it may be interpreted by a Massbus controller to be an error associated with the next block transfer.
DATA 00–35 (TR01/2-CBUS)	These 36 bidirectional lines carry the high speed data and are valid only during DATA cycle. The channel control will apply zeros on the DATA lines for a Massbus controller (during its DATA cycle) whenever there is no data transfer request from the Massbus controller.
PAR LEFT/PAR RIGHT (TR01-CBUS)	These two bidirectional lines carry the computed parity for the left and right half word of the DATA lines.

2.11.2 CBus Timing

A clock-time-division multiplexing technique is used to control the CBus operations. A free-running clock exists in the EBox and is sent to the MBox by internal connections. One delay line per Massbus controller is used to synchronize (deskew) the signals between each Massbus controller and the channel control logic of the MBox.

The channel control continuously selects one of the eight controllers by generating eight selection lines in the following sequence: 0,1,2, 3,4,5,0,1,2,3,6,7;0,1,2,3,4,5. . . . (Figure 2-10). The sequence is stepped with the leading edge of the clock signal.

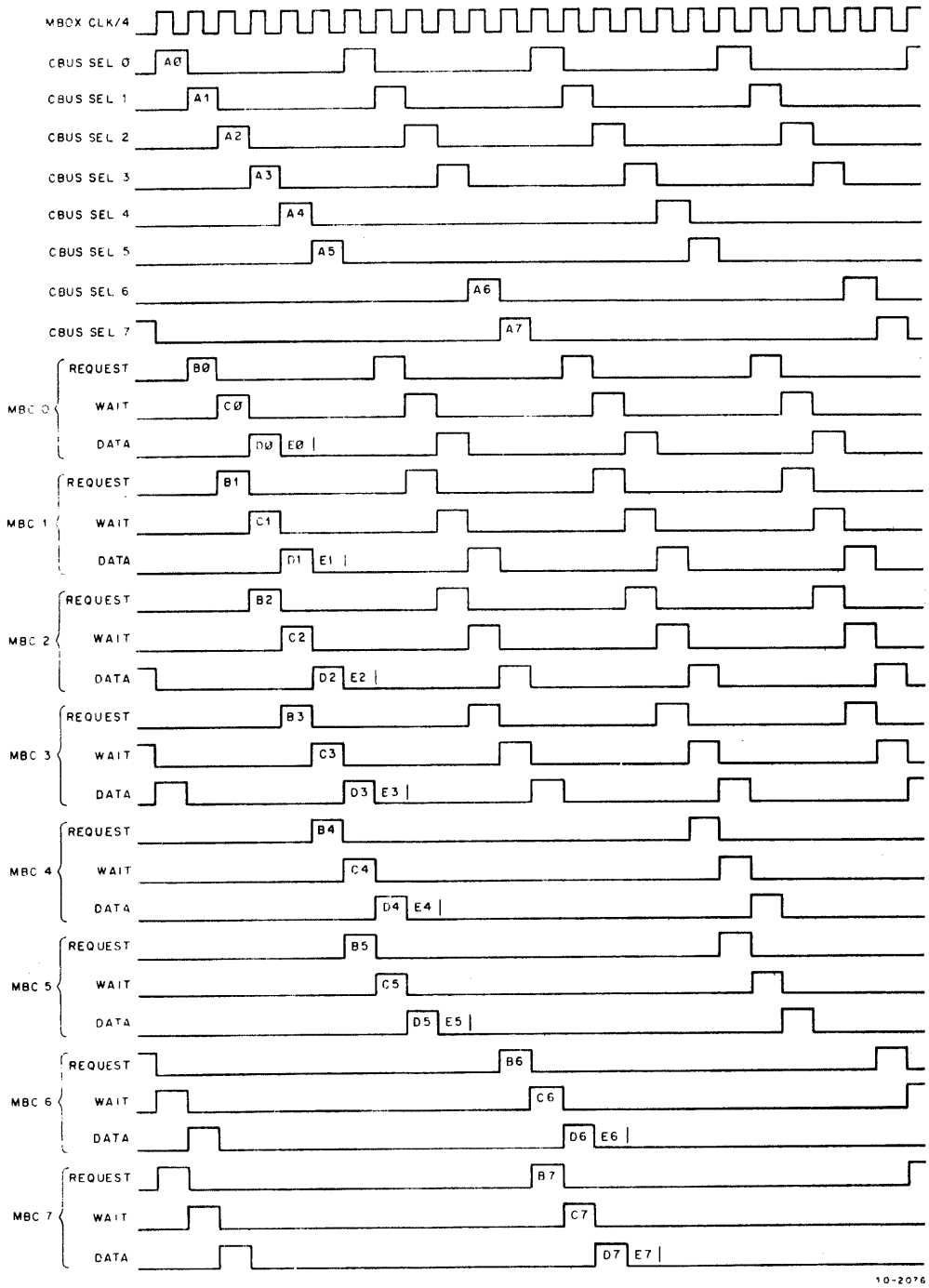
A Massbus controller is allowed to transmit or receive data and control information only after it has been selected by the channel control. Figure 2-10 shows the four cycles used by the channel control and a Massbus controller during a data transfer operation. Each cycle is asserted by the leading edge of a clock pulse and is negated by the leading edge of the next clock pulse.

a. SELECT cycle – The SELECT line of a particular Massbus controller is asserted throughout this cycle.

b. REQUEST cycle – The selected Massbus controller will assert the REQUEST line (if data request is needed) throughout this cycle.

c. WAIT cycle – This cycle is used by the channel control to prepare data and status for transmission. Neither data nor status is asserted during this cycle.

d. DATA cycle – Data is placed on the DATA lines either by the MBox (output data transfer) or by the Massbus controller (input data transfer) during this cycle. The recipient of the data will strobe the data lines at the trailing edge of the data cycle. All CBus control lines (ERROR, READY, LAST WORD, CTOM, START, RESET, DONE, and STORE), except the REQUEST line, are allowed to be asserted during this cycle only.



NOTES 1. CBUS control signals are asserted only during the DATA time slot
The control signals are: START, RESET, CTOM, READY, LAST WORD, DONE, STORE.

2. CBUS REQUEST is asserted only during the REQUEST time slot.

Figure 2-10 Channel Scanner Timing Diagram

Controllers 0, 1, 2, and 3 are selected twice as often by the channel control's selection sequence as controllers 4, 5, 6, and 7.

a. The maximum transfer rate of Massbus controllers 0, 1, 2, 3 is 1 μsec/word.

b. The maximum transfer rate of Massbus controllers 4, 5, 6, and 7 is 2 μsec/word.

2.11.3 Functional Description of Channel Read (NOT CTOM) and Channel Write (CTOM)
The following are descriptions of channel read and channel write operations presented in a chronological context. Refer to Figure 2-11 and 2-12.

2.11.3.1 Channel Write Operation (CTOM) – A channel write operation transfers data from the drive (reads from the drive medium) to main memory.

a. A user program will trap to the monitor when I/O is required because the timesharing user programs I/O using monitor calls.

b. The monitor decides which and how many physical blocks to read. For directory devices (disks), a file search may need to be done to obtain this information.

c. The monitor sets up the CCW list in main memory. The starting address is EPT + (4 × Phys No.) + 0.

d. The monitor sets up the Massbus drive and RH20 Massbus controller to execute a READ operation. This may involve a seek and/or a search operation after which the monitor executes a DATAO instruction to transfer the drive read command to the RH20 and the drive.

e. If the channel control is not busy (CBUS READY is negated), the RH20 asserts CBUS START/RESET and CTOM during the DATA cycle (time D slot) of the scanner when the command is transferred from the RH20 to the drive.

f. The channel control responds to CBUS START/RESET and CTOM by fetching the first CCW and then asserting CBUS READY. The control RAM, CCWF queue, and MB request logic are all involved in this operation. The address for the memory request is obtained from the CCW BUF which contains a CLP.

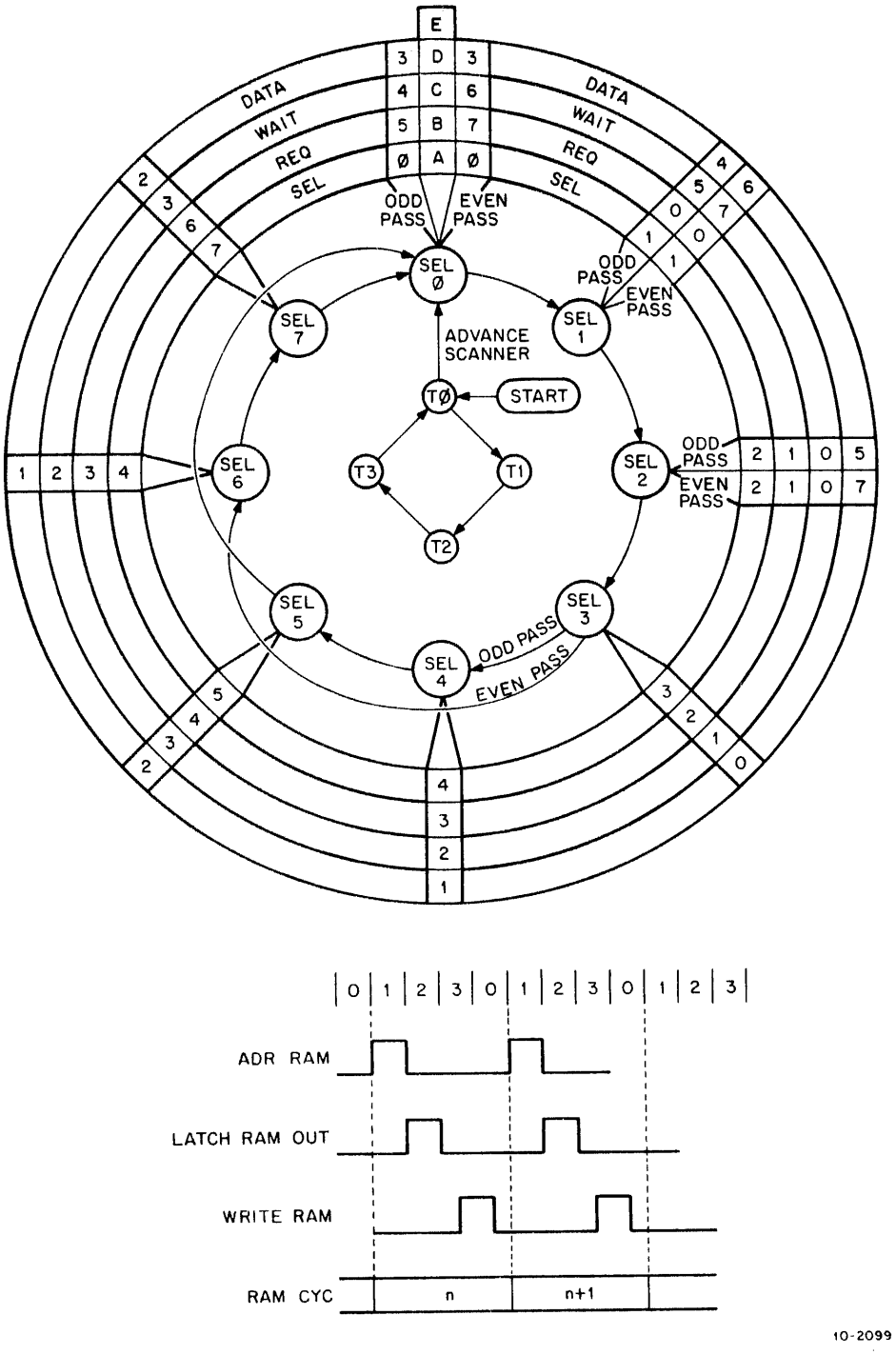
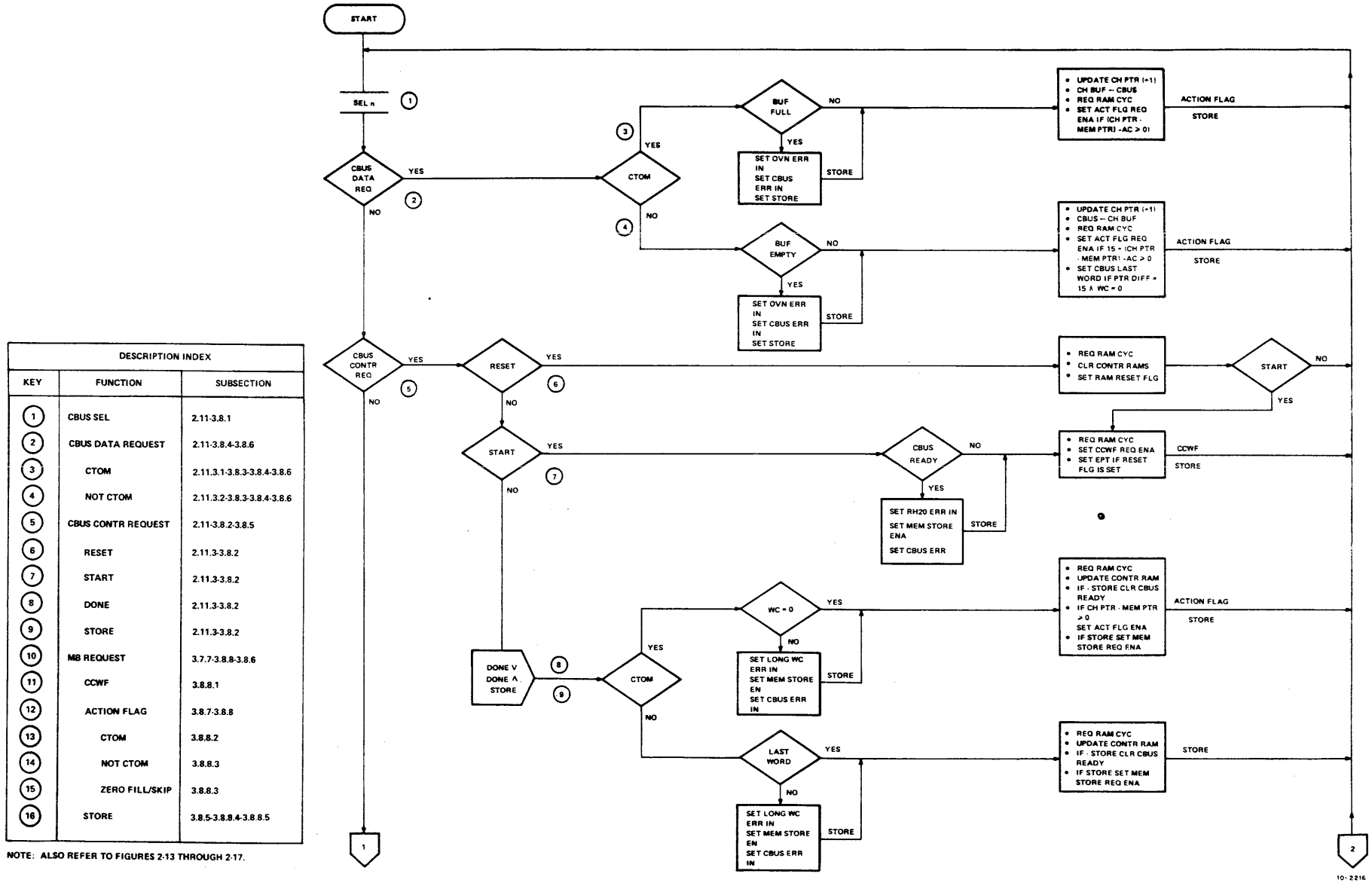


Figure 2-11 Channel Scanner State Diagram



DESCRIPTION INDEX		
KEY	FUNCTION	SUBSECTION
1	CBUS SEL	2.11-3.8.1
2	CBUS DATA REQUEST	2.11-3.8.4-3.8.6
3	CTOM	2.11.3.1-3.8.3-3.8.4-3.8.6
4	NOT CTOM	2.11.3.2-3.8.3-3.8.4-3.8.6
5	CBUS CONTR REQUEST	2.11-3.8.2-3.8.5
6	RESET	2.11.3-3.8.2
7	START	2.11.3-3.8.2
8	DONE	2.11.3-3.8.2
9	STORE	2.11.3-3.8.2
10	MB REQUEST	3.7.7-3.8.8-3.8.6
11	CCWF	3.8.8.1
12	ACTION FLAG	3.8.7-3.8.8
13	CTOM	3.8.8.2
14	NOT CTOM	3.8.8.3
15	ZERO FILL/SKIP	3.8.8.3
16	STORE	3.8.5-3.8.8.4-3.8.8.5

NOTE: ALSO REFER TO FIGURES 2-13 THROUGH 2-17.

Figure 2-12 Channel RAM Cycle Control Functional Flow Diagram (Sheet 1 of 3)

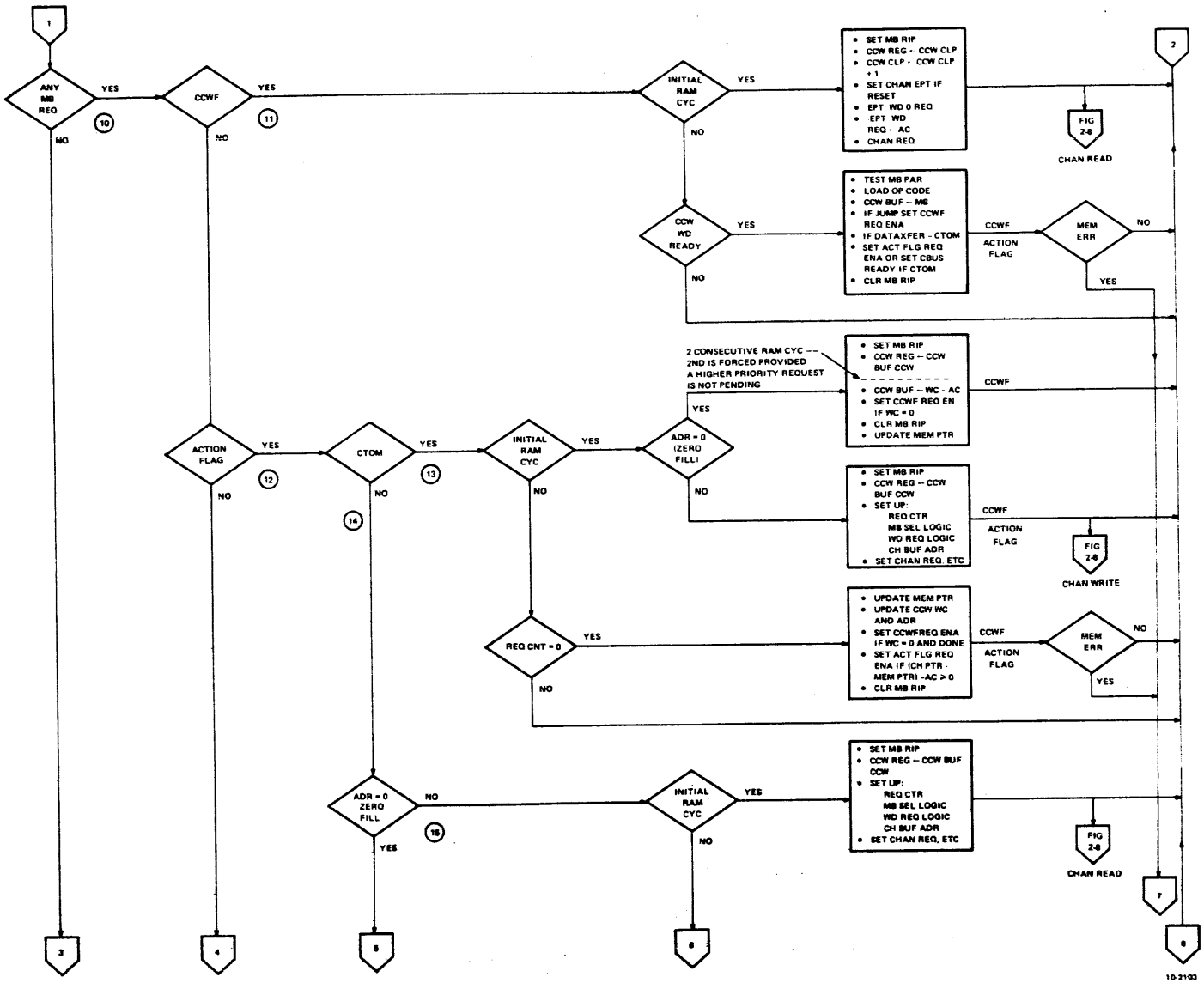


Figure 2-12 Channel RAM Cycle Control Functional Flow Diagram (Sheet 2 of 3)

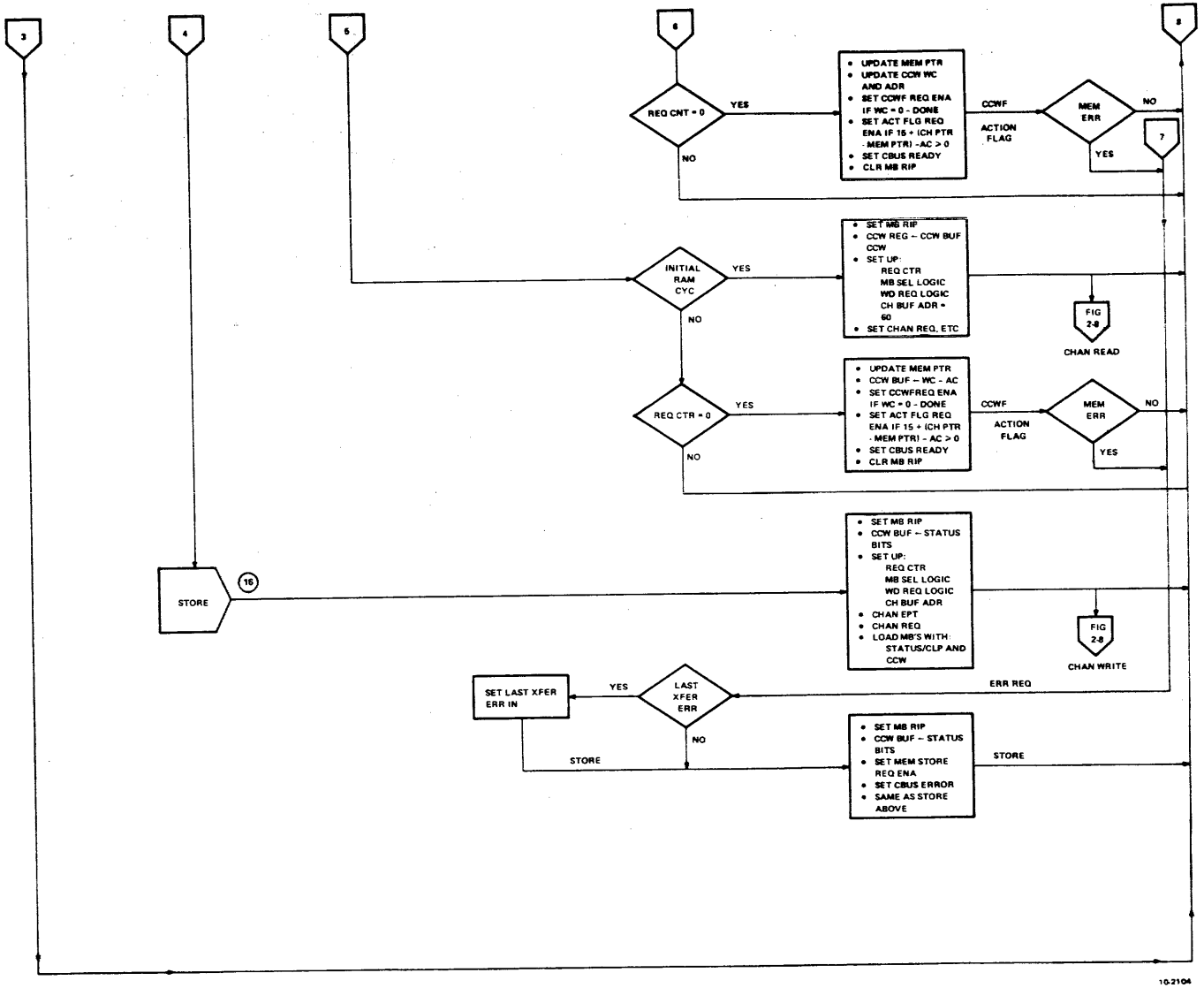


Figure 2-12 Channel RAM Cycle Control Functional Flow Diagram (Sheet 3 of 3)

g. Memory requests (CCL HOLD MEM) are made by the MB request logic to fetch additional channel command words as long as DATA XFER CCW is not received. Memory requests are made as follows:

1. Assertion of CBUS START/RESET and CTOM causes the channel control to initiate a CBUS CONTR CYC to update the control RAM.

2. Thereafter, a memory request is issued to fetch the CCW and load it into the CCW BUF.

   If CBUS RESET was asserted by the RH20, then the first CCW is fetched from the EPT (EPT + [4 × Phys No.] + 0).

   If CBUS RESET was not asserted, then the first CCW is fetched from the location pointed to by the CLP + 1 in the CCW BUF.

NOTE
Additional CCWs are fetched until a DATA XFER CCW is received.

3. CBUS READY enables the RH20 to allow it to transfer words from its buffer to the channel buffer (CH BUF) via the CBUS. CBUS REQUEST is set every time a word is placed on the CBus.

h. At this point, the channel control is executing two operations:

1. Fetching CCWs until a DATA XFER CCW is received.

2. Placing a word into the CH BUF every time CBUS REQUEST is asserted by the RH20.

NOTE
Enough time is assured to receive a DATA XFER CCW before a memory request is forced to transfer these words to memory.

i. CBUS REQUEST causes the channel control logic to execute a REQ CYC to move the word from the CBus into a CH BUF location. The RH20 asserts CBUS REQUEST only when it has a word to transfer. Words are assembled in the RH20 in a two-word data buffer, one half-word at a time. These half-words are received by the RH20 from the Massbus drive via the Massbus at a rate dependent on the drive characteristics.

j. Every time a word is placed in the CH BUF, the CH PTR is updated. Also, an arithmetic algorithm is applied to the CH POINTER, MEM POINTER, and ACT COUNT to see if enough words are in the CH BUF to warrant a memory request to store the words.

1. ACTION COUNT is a function of CCW CHA 34–35 and the CCW WC. This count specifies how many words must be in the CH BUF before a memory request can be started to store up to four more words.

NOTE
Memory requests cannot be made for words that cross the quadword boundary.

2. MEM PTR is advanced by the ACTION CNT when all the words have been moved to the MBs.

3. The CH PTR is advanced by one when a CBUS REQ CYC is executed.

4. The difference between the MEM and CH PTRs specifies the number of words in the CH BUF for a given channel.

5. An action flag is set to initiate a memory request if (CHAN PTR -MEM PTR) -ACTION COUNT $\geqslant$ 0.

When the above condition is satisfied, the action flag is set and a memory request for the number of words specified by the action count can be initiated. The memory request will be executed as follows:

1. Set CCL ACT FLAG REQ.
2. Set CCL HOLD MEM.
3. Request INIT RAM cycle.

NOTE

CBus data requests, CBus control requests, and memory requests all require at least one RAM cycle to obtain the necessary information for executing the request and for reading and writing the RAMs (Control RAM, CCW BUF, CH BUF, and Pointer RAMs).

Since there are only a limited number of RAM cycles available (one every four ticks), an order of priority has been established for granting RAM cycles. This order is:

1. **CBUS DATA REQ for channels 0-7.**
2. **CBUS CONTROL REQ for channesl 0-7.**
3. **MEMORY REQUESTS for channels 0-7.**

Memory requests are made for fetching CCWs, transfering data, and storing status. To ensure efficient channel operation, an order of priority also exists for allocating RAM cycles to memory requests. The order is:

1. **Fetch CCW (CCWF REQ).**
2. **Data (ACT FLAG REQ).**
3. **Store Status (MEM STORE REQ).**

Since heavy channel activity (CBus requests for data) can consume many of the available RAM cycles, control requests and memory requests are queued so that they are remembered and can be executed in the proper order of priority when RAM cycles become available.

All channels are assured at least one RAM cycle for each scanner pass. When there is more than one request pending, only the higher priority request is excuted for a given channel. Therefore, a given channel may have to wait before a pending memory request or control request gets a RAM cycle. When initiated, the memory request and type (CCWF, ACT FLAG, or MEM STORE REQ) are latched. Then, if a higher priority request comes in, it will not be granted until the current request is done.

4. A RAM cycle is needed for an action flag memory request to transfer the appropriate CCW word (CCW CHA 14–35) from the CCW BUF to the CCW register and to read the ACT CNT and the MEM PTR. This address is needed to address core/cache. The ACT CNT, in conjunction with the least two significant bits of the address (bits 34 and 35), is used to set up the word request logic; the MEM PTR, in conjunction with the channel code of the ACT FLAG REQ, is used to form the CH BUF ADR.

NOTE

After the memory transfer is completed, the CCW address (ADR) will be incremented and the WC will be decremented by the value contained in the action counter and written back into the CCW BUF.

5. After the action flag memory request gets a RAM cycle, CCL CHAN REQ, along with the appropriate request qualifiers, is set to request a cache cycle.

6. If the cache control is not busy (IDLE), if core is not busy, and an MB request is not pending, the cache control grants a cache channel cycle.

7. The channel control recognizes that a cache channel cycle is granted by sensing that CSH CHAN CYC is asserted.

8. The channel control then asserts CCL START MEM and CCL CH LOAD MB.

NOTE

CCW WD 0–3 REQ and CCL CHAN TO MEM where latched, along with CCL CHAN REQ, to request a Cache cycle. The CCW WD 0–3 REQ signals are a function of ACTION COUNT, and the MB SEL 1–2 signals are a function of CHA ADR 34–35.

9. A word is transferred to an MB every four clock ticks. As each word is transferred, the MB SEL 1–2 counter is incremented, the REQ counter is decremented, and the CH BUF ADR is advanced until all words are transferred.

10. When the contents of the request counter are 0, a request for a RAM cycle is again made to update the MEM PTR (MEM PTR ← MEM PTR – ACT CNT). The CCW ADR and WC are also updated (ADR ← ADR + ACT CNT; WC ← WC – ACT CNT).

11. While the channel control is transferring the words to the MB, the cache control checks to see if cache has any valid words. If any valid words are found, the VALID and WRITTEN bits are cleared for these words. Thereafter the cache control returns to IDLE.

12. CCL START MEM is asserted approximately the same time the first word is transferred to an MB, assuring that the MBs have at least one word when the core control is started. Subsequent words are transferred to the MBs faster than the core control can move them into core.

13. While the core control is transferring the words to core, the EBox can access the cache, but a core reference cannot be started until the current reference is done.

14. The core control acknowledge pulse counter keeps track of the number of words transferred and clears core busy after all words are transferred.

l. As long as there are enough words in the CH BUF (CHAN PTR - MEM PTR) - ACTION COUNT≥0 and the CCW WC is not zero, additional action flag memory requests are initiated and executed as described in j and k above.

m. Each time CBUS REQUEST is asserted by the RH20, another word is moved from the CBUS to the CH BUF and the pointers are updated as described in j above.

n. When the WC of the CCW reaches zero, a request to fetch the next CCW, which is pointed to by the CLP in the CCW BUF, is initiated.

o. The operations described above are repeated until either a LAST DATA XFER or a HALT CCW is fetched. If a HALT CCW is fetched, the channel simply halts. If a LAST DATA XFER CCW is fetched, the channel continues to execute the transfer until the WC reaches zero. In either case, the RH20 interrupts the processor when the Block Count (BC) reaches zero to inform it that the channel operation is done.

NOTE

Various error conditions can be sensed throughout the entire write operation (Paragraph 3.8.5). In addition, when the channel halts, both the CCW WC, which is maintained by the channel control logic, and the BC, which is maintained by the RH20, must be zero.

2.11.3.2 Channel Read Operation (NOT CTOM) - A channel read operation transfers data from main memory to the drive (writes on the drive medium).

a. A user program will trap to the monitor when I/O is required because the timesharing user programs I/O using monitor calls.

b. The monitor decides which and how many physical blocks to write. For directory devices (disks), a file search may need to be done to obtain this information.

c. The monitor sets up the CCW list in main memory. The starting address is EPT + (4 × Phys No.) + 0.

d. The Monitor sets up the Massbus drive and the RH20 Massbus controller to execute a write operation. This may involve a seek and/or a search operation after which the monitor executes a DATAO instruction to transfer the device write command to the RH20 and the drive.

e. If the channel control is not busy (CBUS READY is negated), the RH20 asserts CBUS START/RESET (but not CTOM) during the DATA cycle (time slot D) of the scanner when the command is transferred from the RH20 to the drive.

f. The channel control responds to CBUS START/RESET and NOT CTOM by fetching the first CCW. The control RAM, CCWF REQ queue, and MB request logic are involved in this operation. The address for the memory request is obtained from the CCW buffer, which contains a CLP.

g. Memory requests (CCL HOLD MEM) are made by the MB request logic to fetch additional CCW as long as a DATA XFER CCW is not received. Memory requests are made as follows:

1. Assertion of CBUS START/RESET and NOT CTOM causes the channel control to initiate a CBUS CONTR CYC to update the control RAM.

2. Thereafter, a memory request is issued to fetch the CCW and load it into CCW BUF.

   If CBUS RESET was asserted by the RH20, then the first CCW is fetched from the EPT (EPT + 4 × Phys No. + 0).

   If CBUS RESET was not asserted, then the first CCW is fetched from the location pointed to by CLP + 1 in the CCW BUF.

NOTE
Additional CCW are fetched until a DATA XFER CCW is received.

h. When a DATA XFER CCW is received, memory requests are made to fetch the data words from memory specified by the WC and ADR in the CCW. As the words are received, they are moved into the CH BUF.

i. When two words are in the CH BUF providing the WC≥2, CBUS READY is asserted.

j. The RH20 responds to CBUS READY by asserting CBUS REQUEST since its two data buffers are empty.

k. CBUS REQUEST causes the channel control logic to execute a REQ CYCLE to move a word from the CH BUF to the CBus. Two requests will be made by the RH20 back-to-back since the RH20 has a two word buffer. Additional requests will be made every time a buffer location is empty. The RH20 buffer is unpacked one half word at a time and placed on the Massbus to be written on the drive medium.

l. The CH PTR is updated every time a word is taken from the CH BUF. Also, an arithmetic algorithm is applied to the CH POINTER, MEM POINTER, and ACTION COUNT to see if there are enough empty locations in the CH BUF to warrant another memory request to fetch up to four more words from core.

1. ACTION COUNT is a function of CCW CHA 34–35 and the CCW WC. This count specifies how many empty locations (number of words to be fetched next) must be in the CH buffer before a memory request for additional words can be made.

NOTE

Memory requests cannot be made for words that cross the quadword boundary.

2. The MEM PTR is advanced by the ACTION COUNT when all requested words have been received.

3. The CH PTR is advanced by one when a CBUS REQ CYC is executed.

4. The difference between the MEM and CH PTRS + 15 specifies the number of empty locations in the CH BUF for a given channel.

5. An action flag is set to initiate a memory request if:

$$15 + (\text{CHAN PTR} - \text{MEM PTR}) - \text{ACTION COUNT} \geq 0.$$

When the above condition is satisfied, the action flag is set and a memory request for the number of words specified by the ACTION COUNT can be initiated. The memory request will be executed as follows:

1. Set CCL ACT FLAG REQ.
2. Set CCL HOLD MEM.
3. Request INIT RAM cycle.

NOTE

CBus data requests, CBus control requests, and memory requests all require at least one RAM cycle to obtain the necessary information for executing the request and for reading and writing the RAMs (control RAM, CCW BUF, CH BUF, and pointer RAMs). Since there are only a limited number of RAM cycles available (one every four clock ticks), an order of priority has been established for granting RAM cycles. This order is:

1. **CBUS DATA REQ for channels 0–7.**
2. **CBUS CONTROL REQ for channels 0–7.**
3. **MEMORY REQUESTS for channels 0–7.**

Memory requests are made for fetching CCW's, transferring data, and for storing status. To ensure efficient channel operation, an order of priority also exists for allocating RAM cycles to memory requests. The order is:

1. **Fetch CCW (CCWF REQ).**
2. **Data (ACT FLAG REQ).**
3. **Store Status (MEM STORE REQ).**

Since heavy channel activity (CBus requests for data) can consume many of the available RAM cycles, control requests and memory requests are queued so that they are remembered and can be executed in the proper order of priority when RAM cycles become available.

All channels are assured a RAM cycle for each scanner pass. But, when there are more than one request pending, only the higher priority request is executed. Therefore, a given channel may have to wait before a pending memory request or a control request gets a RAM cycle. The memory request and type (CCWF, ACT FLAG, MEM STORE REQ) are latched when made. Then, if a higher request comes in, it will not be granted until the current request is done.

4. A RAM cycle is needed for an action flag memory request to transfer the appropriate CCW word (CCW CHA 14–35) from the CCW BUF to the CCW register and to read the ACT CNT and MEM PTR. This address is needed to address core/Cache. The ACT CNT, in conjunction with the least two significant bits of the address (bits 34 and 35), is used to set up the word request logic; the MEM PTR, in conjunction with the channel code of the ACT FLAG REQ, is used to form the CH BUF ADR.

NOTE

After the memory transfer is completed, the CCW ADR will be incremented and the WC will be decremented by the value contained in the action counter and written back into the CCW BUF.

5. After the action flag memory request gets a RAM cycle, CCL CHAN REQ, along with the appropriate request qualifiers, is set to request a Cache cycle.

6. If the cache control is not busy (IDLE), if core is not busy, and an MB request is not pending, the cache control grants a cache channel cycle.

7. The channel control recognizes that a cache channel cycle is granted by sensing that CSH CHAN CYC is asserted.

NOTE

The channel control then waits for the cache control and core control to execute the request.

8. The cache control checks to see if any valid words are in the cache.

   If there are valid words in the cache, the cache control moves the valid words into corresponding MBs and starts a core cycle for those words that are not valid.

   If there are no valid words in the cache, the cache control starts a core read cycle for all requested words.

9. As the words are placed in the MB by cache control or core control, MB 0-3 HOLD IN is negated for one clock tick to load the corresponding MB. The channel control senses this and sets AF WD READY to move the word into the CH BUF and to advance the REQ CTR, MB SEL CTR, and the CH BUF ADR.

NOTE

The channel control will take the words only in the order 0, 1, 2, and 3. Therefore, even if some high-order words are in the cache, the low-order words have to come in from core first and be transferred to the CH BUF before the high-order words are transferred.

10. As each word is taken by the channel, the REW CTR, MB SEL CTR, and the CH BUF ADR are advanced.

11. When REQ CTR reaches zero, a second request for a RAM cycle is made to update the MEM PTR (MEM PTR ← MEM PTR + ACT CNT). The CCW WC and ADR are also updated (ADR ← ADR + ACT CNT; WC ← WC - ACT CNT).

12. Core busy is cleared by core data valid counter of the core control when all requested words have come in.

n. As long as there are enough empty locations in the CH BUF (15 + [CHAN PTR - MEM PTR] - ACT CNT ⩾ 0), additional action flag memory requests are initiated and executed as described in l and m above.

o. Each time CBUS REQUEST is asserted by RH20, another word is moved from the CH BUF to the CBus and the pointers are updated as described in l above.

p. When the WC reaches zero, a request to fetch the next CCW, which is pointed to by the CLP in the CCW BUF, is initiated.

q. The operation described above is repeated until either a LAST DATA XFER or a HALT CCW is fetched. If a HALT CCW is fetched, the channel simply halts. If a LAST DATA XFER CCW is fetched, the channel continues to execute the transfer until the WC reaches zero. In either case, the RH20 interrupts the EBox when the BC reaches zero to inform it that the channel is done.

NOTE

Various error conditions can be sensed throughout the entire read operation (Subsection 3.8.5). In addition, when the channel halts both the CCW WC, which is maintained by the channel control logic, and the BC, which is maintained by the RH20, must be zero.

2.12 ADDRESS AND DATA PATHS

The specific address and data paths in the MBox are shown on Figure 2-13.

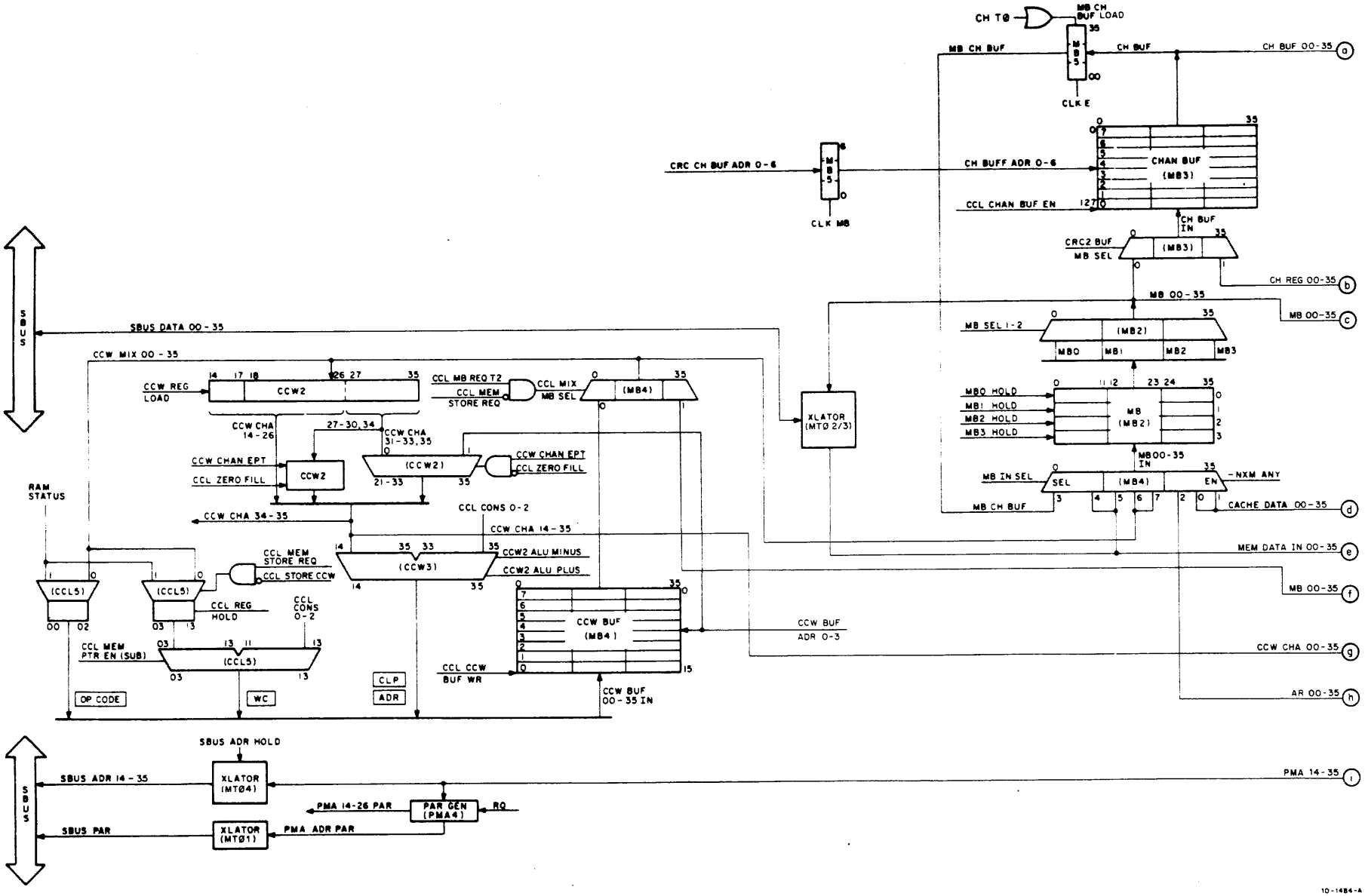


Figure 2-13 MBox Address and Data Path, Logic Diagram (Sheet 1 of 3)

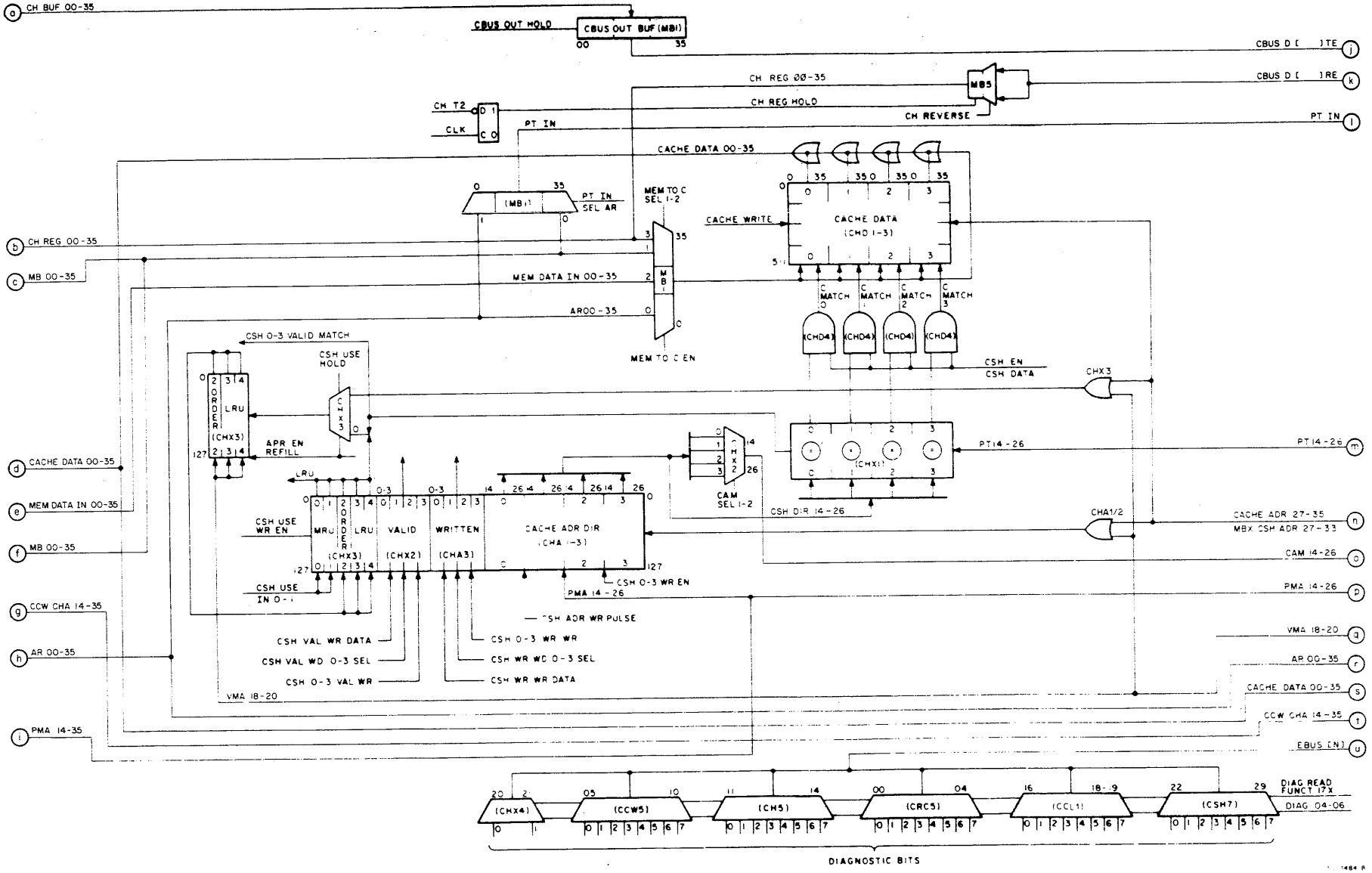


Figure 2-13 MBox Address and Data Path, Logic Diagram (Sheet 2 of 3)

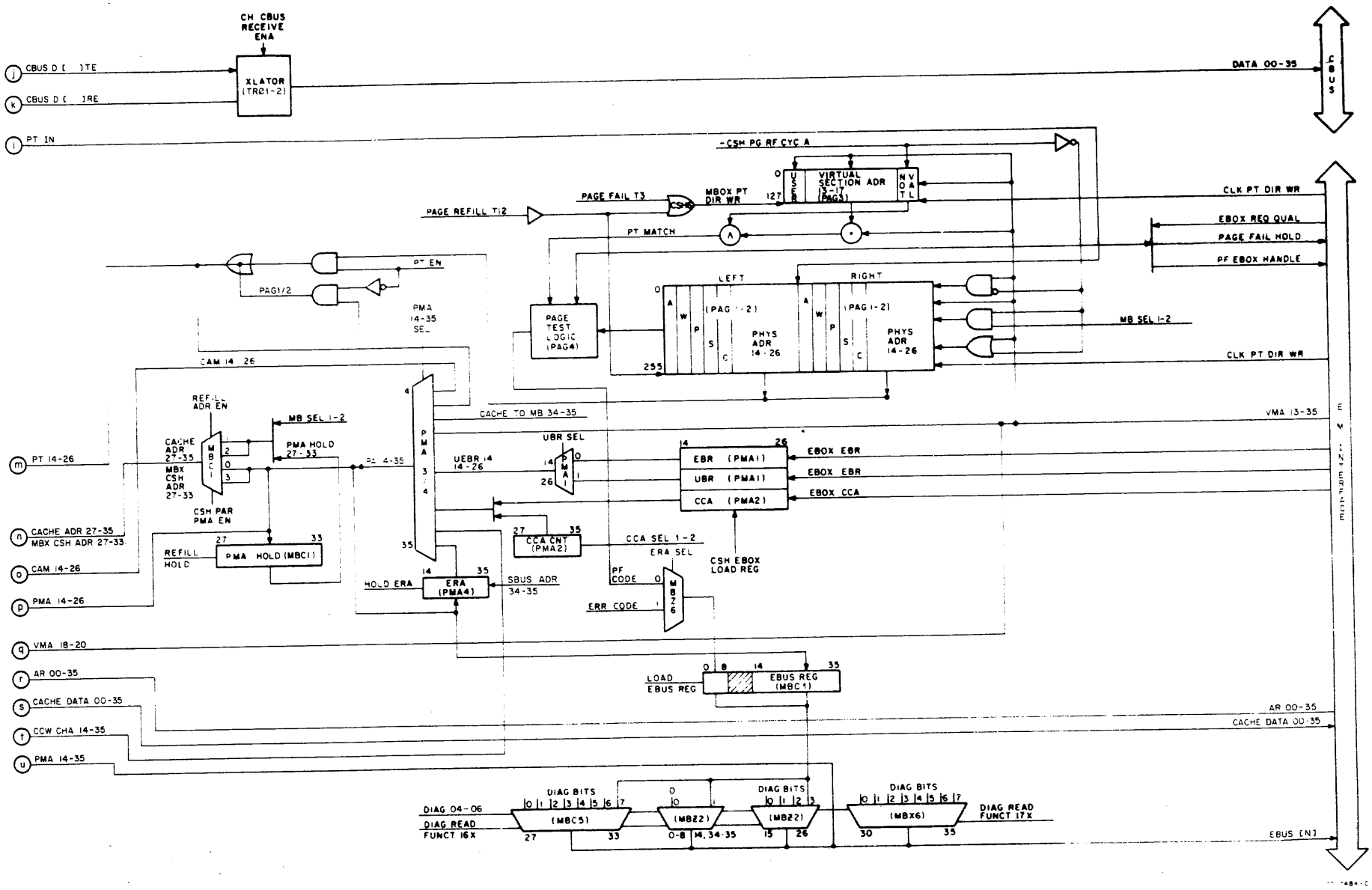


Figure 2-13 MBox Address and Data Path, Logic Diagram (Sheet 3 of 3)

The functional elements in the address path between the EBox VMA, the CBus, and the SBus involved in forming the physical memory address are:

a. Physical Memory Address Mixer (PMA)
b. Page Table and Page Table Directory
c. User and Executive Base Registers (UBR and EBR)
d. Cache Clearer Address Register (CCA)
e. PMA HOLD Register
f. Cache Directory
g. Cache Address Mixer (CAM)
h. Channel Command Word (CCW) Register and CCW Buffer

The correct physical memory address is formed by the PMA under explicit control of the cache cycle control. The desired address mixture is selected and held when a particular cache cycle is started. This address is then used to address the cache and core memory if a core cycle is started.

The PMA is a 22-bit eight-input mixer that receives various types of addresses for forming the desired physical memory address for a given cache cycle.

The page table contains 512 entries that are associated with (indexed by) entries in the page table directory. Each page table directory entry identifies four adjacent entries in the page table; consequently, the directory contains 128 entries. Both the page table and the page table directory are addressed by the virtual address every time a cache EBox cycle is started.

The UBR, EBR, and CCA registers are loaded from the VMA. The contents of these registers are made available to the PMA so that the correct physical memory address can be formed by the PMA.

The PMA HOLD register is loaded when a core read cycle is started. This address is then used to move the words coming in from core into the cache. This address needs to be held since the EBox can issue another request and can get into the cache after the first word comes in from core.

The cache directory contains one physical memory page address location for each corresponding quadword location in the cache data buffer. This address is made available to the PMA so that the correct physical memory address can be formed by the PMA for a write-back operation.

The cache directory is addressed by the VMA, PMA, or the refill address from the PMA HOLD register, depending on the particular cache cycle being executed as outlined in Table 2-9.

Table 2-9 Cache Directory Address Sources

Cache Cycle	Address Source
CSH MB CYC	PMA HOLD 27–33
CSH CHAN CYC	PMA 27–33
CSH EBOX CYC	VMA 27–33
CSH CCA CYC	PMA 27–33
CSH PAGE REFILL CYC	PMA 27–33
CSH WRITEBACK CYC	PMA 27–33

The CAM, a 13-bit four-input mixer, provides the means for distributing the address from the appropriate cache directory quarter to the PMA during a write-back operation. The mixer is controlled by the CAM SEL 1-2 code, which is a function of the cache quarter in which the written words are located.

The CCW buffer contains two words for each channel. These words supply the channel WC, ADR, CLP, and status bits. The CLP (or the address) is transferred to the CCW register and held when the channel issues a request so that the address can be selected by the PMA for distribution to the SBus.

The functional elements in the data path between the EBox AR, the CBus, and the SBus involved in transferring and storing data are:

a. MEM TO C mixer
b. Cache
c. MB IN mixer
d. MBs
e. MB SEL mixer
f. PT IN mixer
g. CH BUF IN mixer
h. CH BUF
i. MB CH BUF
j. CBUS OUT BUF
k. CH REG mixer-latch
l. CCW mixer
m. CCW BUF

Some of these functional elements are controlled by the cache cycle control and core cycle control when a cache cycle is executed and some are controlled by the channel control when channel moves data between the MBs and the CH BUF or the CCW BUF.

The MEM TO C mixer, a 36-bit four-input mixer, provides a means for adjusting the data path within the MBox. The MEM TO C mixer is controlled by the MEM TO C SEL 1-2 code produced by the cache cycle control when a cache cycle is started. Table 2-10 lists the paths that may be established by the mixer.

NOTE

MEM TO C SEL 1-2 code 1 is used for transferring the first word coming in from core if EBOX SYNC is not seen, and for transferring the words following the first word (if any). MEM TO C SEL 1-2 code 2 is used for transferring the first word coming in from core if EBOX SYNC is seen, and for transferring the word coming from core when the SBus diagnostic cycle is executed.

Table 2-10 MEM TO C Mixer Select Codes

MEM TO C SEL 1–2 CODE	Data Path	Function
0	CSH ← AR	EBOX WRITE
1	CSH ← MB AR ← MB	EBOX READ
2	CSH ← SBUS AR ← SBUS	EBOX READ OR EBOX SBUS DIAG
3	CSH ← CH REG	DIAG Function

The cache data buffer contains 512 quadword locations that are associated with (indexed by) corresponding entries in the cache directory. The cache data buffer is addressed by the PMA or the refill address from the PMA HOLD register concatenated with the MB SEL 1–2 code. PMA 27–35 are used for all but the cache MB cycle. When a cache MB cycle is executed, the cache is addressed by the refill address concatenated with the MB SEL 1–2 code to move a word from the MB into the appropriate cache location.

The MB IN mixer a 36-bit eight-input mixer, provides a means for adjusting the data path within the MBox. The MB IN mixer is controlled by the MB IN SEL 1–2–4 code. By adjusting the select code, the MBs can be loaded with data from the following sources:

a. Cache
b. AR
c. CH Buffer
d. SBus
e. CCW Buffer

The four MBs are 36-bit memory buffer registers for temporarily holding the data as it is moved from the source to the destination registers or RAMs. In effect, the MBs serve as a buffer to normalize (compensate for the differences in speed) the transfer of data between the source and destination. The sources for data are selected by the MB IN mixer and the desired destination is selected by one of the following mixers:

a. CH BUF IN
b. PT IN
c. MEM TO C
d. CCW

The MB SEL mixer, a 36-bit, four-input mixer, selects the contents of one of the four MBs when transferring the data to the destination.

The cache cycle, core cycle, and the channel controls all can affect control of the MBs and their input and output mixers.

The CH BUF IN mixer, a 36-bit, two-input mixer, is controlled by the channel control to move data into the CHAN BUF from the selected MB during a channel read operation, or from the CBus data lines during a channel write operation.

The PT IN mixer, a 36-bit, two-input mixer, is controlled by the cache cycle control to load page table entries into the page table from the MBs or from the AR.

The CH BUF contains 16 locations of buffer storage for each channel; consequently, there are 128 locations in the CH BUF to accommodate all eight channels. The CH BUF is addressed by CH BUF ADR 00–06, which is a function of the selected channel and the buffer location to be read or written. This address is formed by the channel control.

The MB CH BUF is a 36-bit register that holds the word to be moved from the CH BUF to the MB via the MB IN mixer during a channel write operation.

The CBUS OUT BUF is a 36-bit register that holds the word to be moved from the CH BUF to the CBus.

The CH REG mixer latch is a 36-bit, two-input mixer combined with a 36-bit register (latch). This mixer latch is controlled by the channel control to adjust the two half words coming in from the CBus (each half word is one word from the drive) in the correct order to accommodate both forward and reverse read operation of a magtape drive before moving the word into the CH BUF.

The CCW mixer, a 36-bit, two-input mixer, is controlled by the channel control in executing the following operations:

a. Transfer a newly fetched CCW that was placed into an MB by the core cycle control from the MB to the CCW BUF.

b. Transfer the ADR or the CLP from the CCW BUF to the CCW register when the channel issues a request to read or write memory.

c. Transfer the status from the CCW BUF to the MBs when the channel issues a request to store the status words.

The channel CCW buffer (CCW BUF) contains two locations of storage for each channel; consequently, there are 16 locations in the CCW BUF to accommodate all eight channels. This buffer contains the WC, the ADR, the CLP, and status information for each channel. The CCW BUF is addressed by CCW BUF ADR 00–03, which is a function of the selected channel and the buffer location to be read or written. This address is formed by the channel control.

2.13 CONTROL LOGIC

The MBox control logic is introduced here in two parts:

a. That logic that is involved in controlling the execution of cache cycles and core cycles. This logic is shown in block form on Figure 2-14.

b. That logic that is involved in servicing CBus requests and issuing channel requests for core cycles. This logic is shown in block form on Figure 2-15.

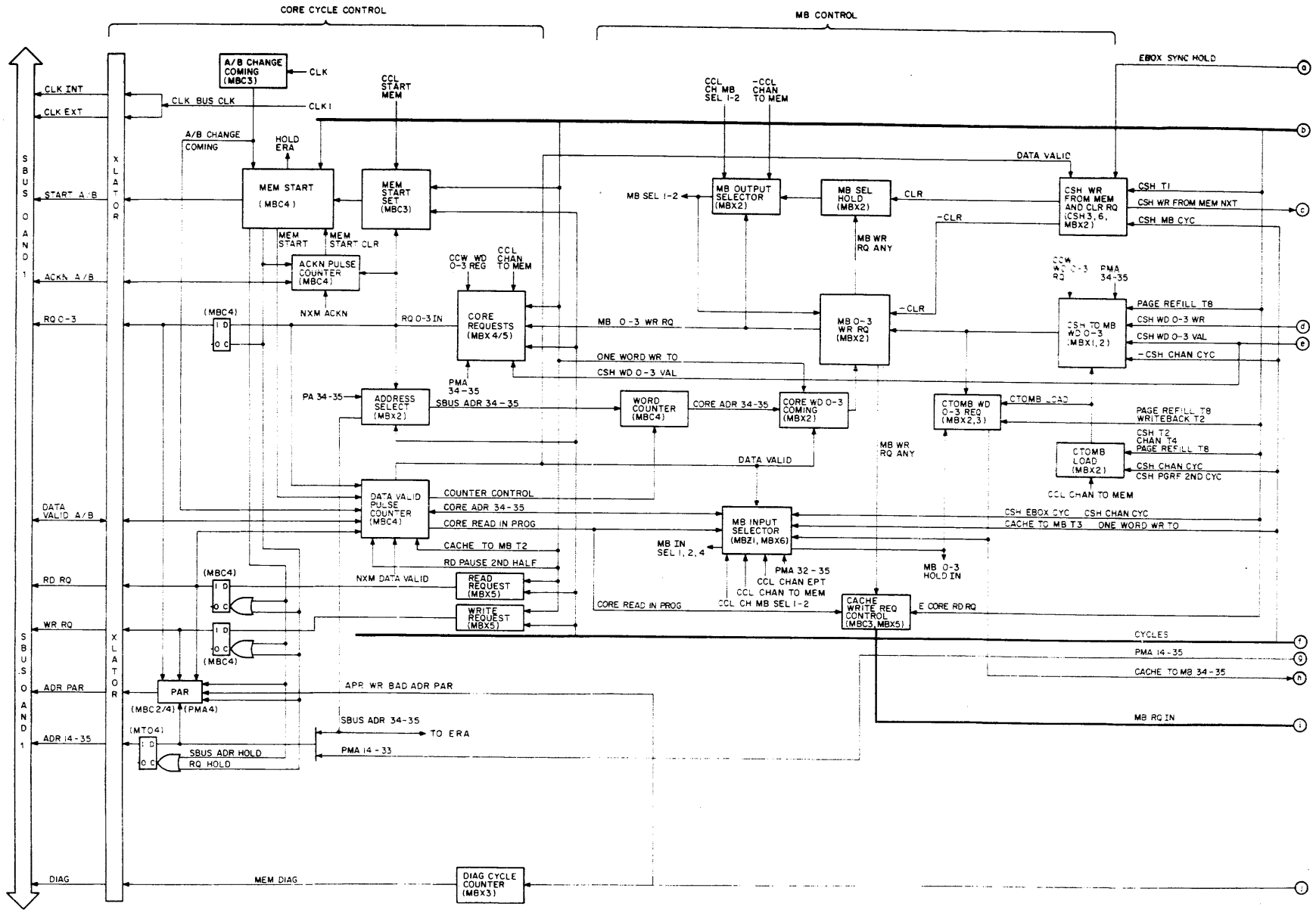


Figure 2-14 Cache/Core Control Logic Block Diagram (Sheet 1 of 3)

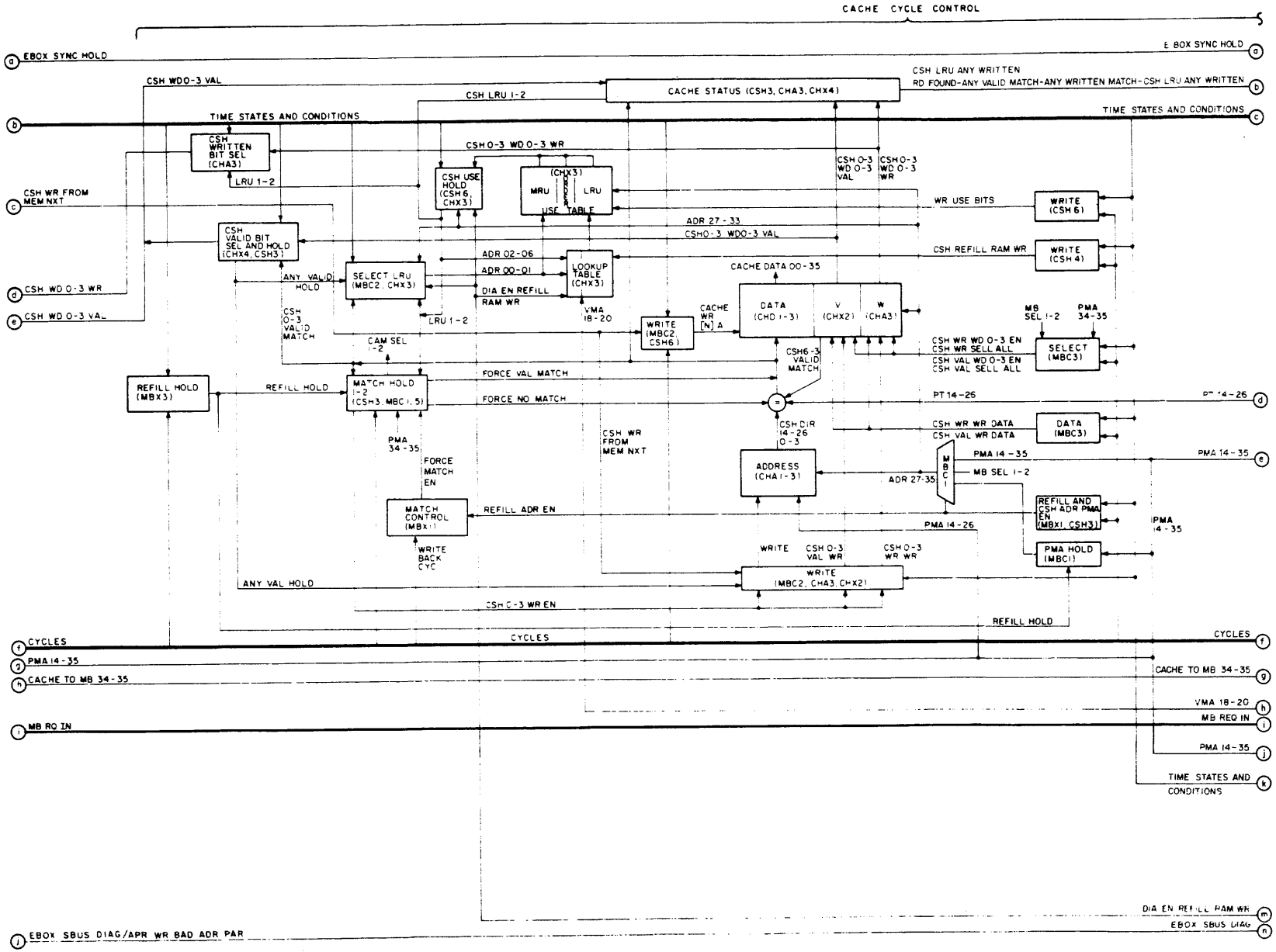


Figure 2-14 Cache Core Control Logic
Block Diagram (Sheet 2 of 3)

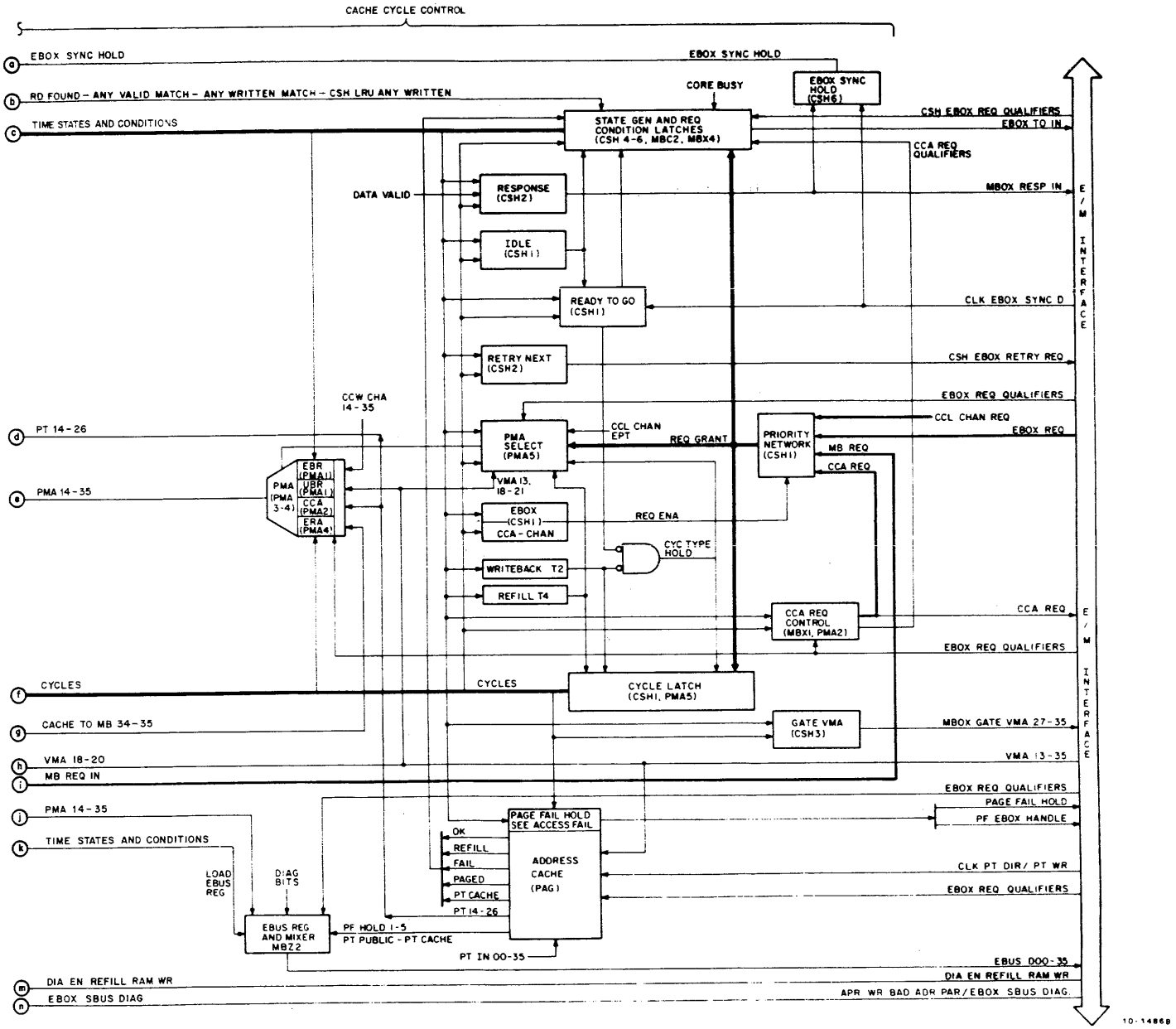


Figure 2-14 Cache/Core Control Logic Block Diagram (Sheet 3 of 3)

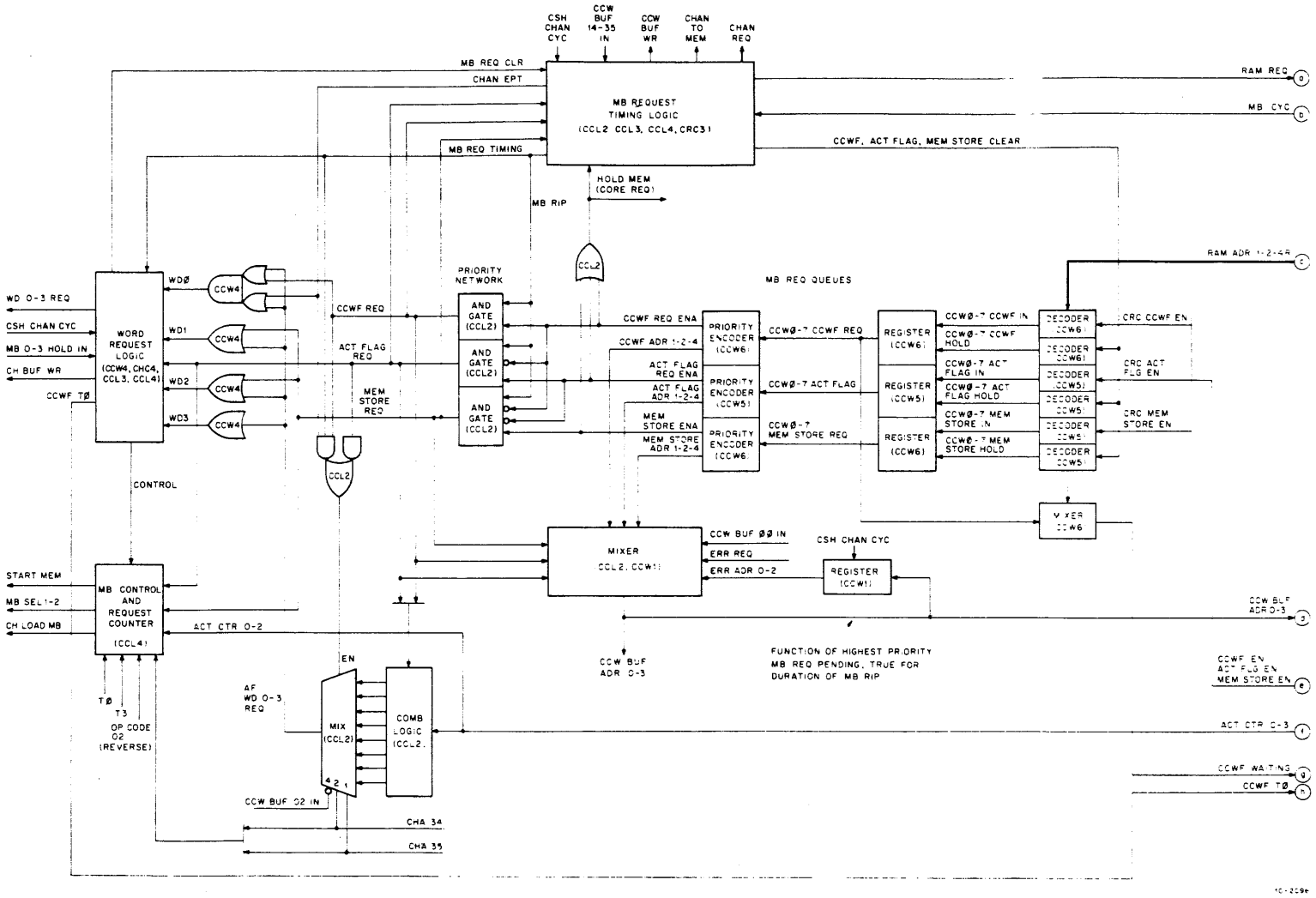


Figure 2-15 Channel Control Logic, Block Diagram (Sheet 1 of 3)

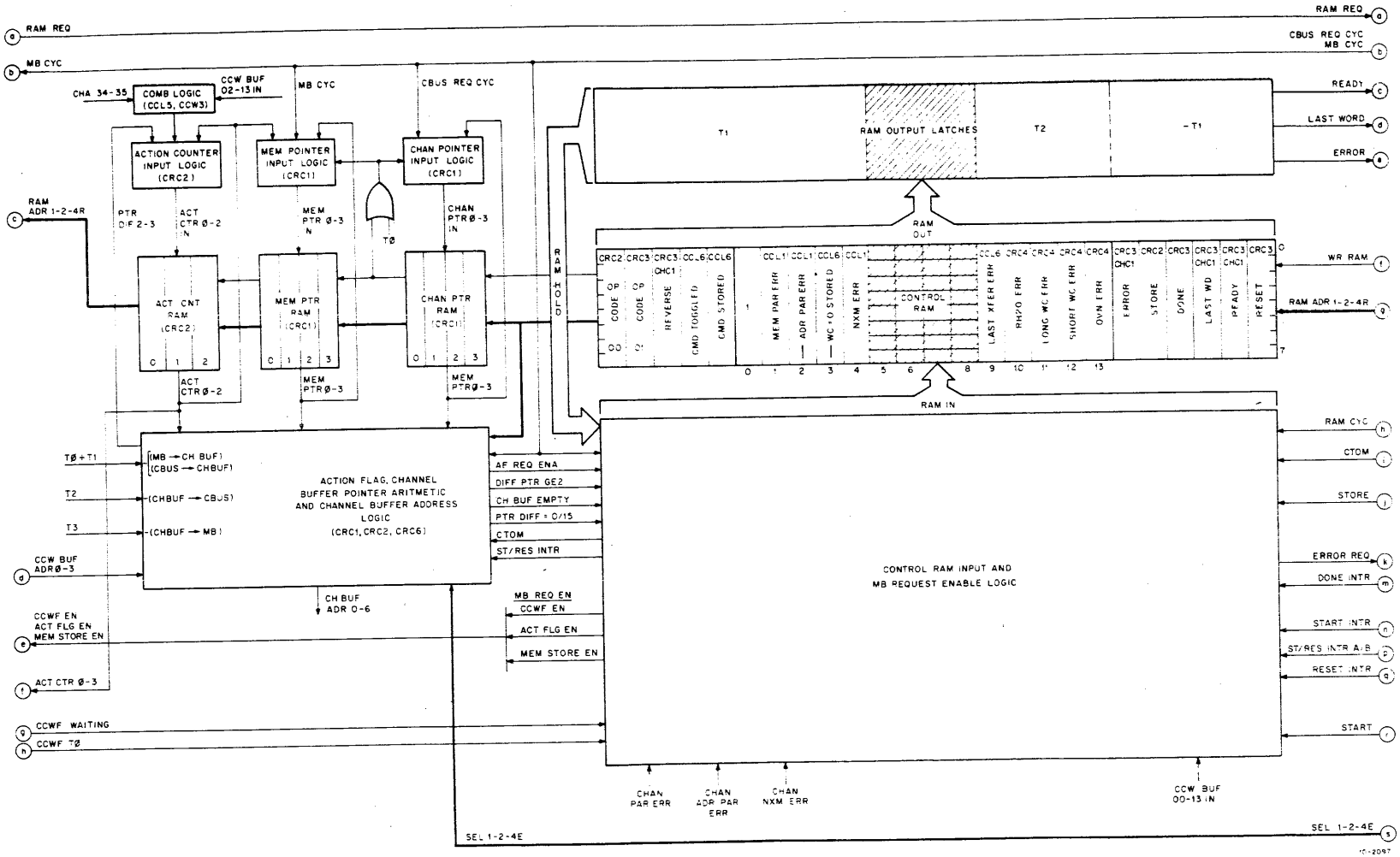


Figure 2-15 Channel Control Logic, Block Diagram (Sheet 2 of 3)

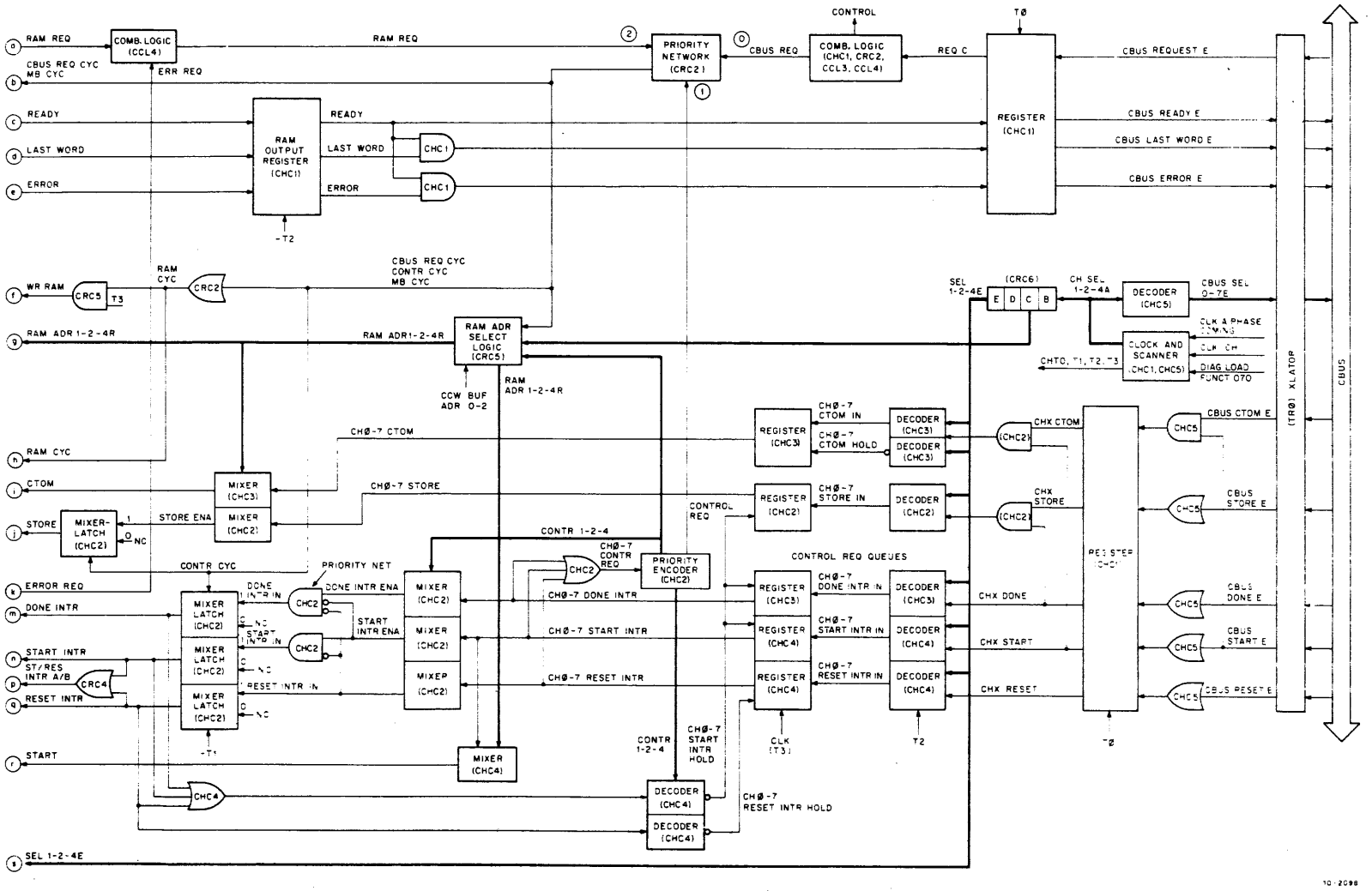


Figure 2-15 Channel Control Logic, Block Diagram
(Sheet 3 of 3)

The purpose of this subsection is to provide some insight into the nature of the MBox control logic. It is not intended to be a detailed description; rather, this subsection attempts to show how the various control functional elements hang together. Appropriate prefixes are included on the block diagram to permit a student or reader to jump directly to the logic print that shows the actual logic.

NOTE
Refer to Section 3 for the logic description.

2.13.1 Cache and Core Cycle Control

The priority network (Figure 2-14, sheet 3) grants a Cache cycle to the pending request having the highest priority. The assigned priorities are:

1. MB REQ
2. CHAN REQ
3. EBOX REQ
4. CCA REQ

When a request is granted, the appropriate cache cycle latch is held, the required physical memory address is selected, and the time state generator is started. The time state generator will then step through a specific set of time states depending on the request qualifiers associated with the granted request and on what, if anything, is found in the cache if it is implemented.

The cache and core cycle control block diagram shows, extending from the time state generator and from the cycle latches, a time-state bus and a cycle bus, respectively. These buses have been defined for the sake of this presentation; they are not so defined in the actual logic. As can be seen by reviewing the block diagram, elements from both the time state and the cycle bus extend to many of the control elements. For the most part, a control element for the cache and core cycle is simply an AND function of a particular time state and a particular cycle.

2.13.2 Channel Control

The channel control consists of essentially two autonomous controls with data, status, and control buffers (RAMs) in between. One control services CBus data and control requests; the other executes memory requests.

The priority network (Figure 2-15, sheet 3) grants a RAM cycle to the pending request having the highest priority. The assigned priorities for a given channel are:

1. CBus Request (for data)
2. Control Request (CBUS RESET, START, or DONE)
3. MB RAM Request (for memory access)

When the request is granted, the appropriate RAM address is selected and the RAM is updated.

Each time the RAM is updated, its contents are also read to generate internal operations for executing the granted request.

When a CBus request is granted, one data word is transferred between the CH BUF and the CBus (to or from the RH20 as specified) and the status bits and pointers in the RAM are updated.

When a control request is granted, appropriate control bits in the control RAM are set, cleared or updated, and appropriate internal requests are initiated to execute the control operation.

MB RAM requests are issued to initiate a memory operation and to update the control RAM after the memory operation is completed.

2.14 ERROR CHECKING AND REPORTING LOGIC

The following error checking and reporting logic (Figures 2-16 and 2-17) is implemented in the MBox:

a. Address Parity
b. Data Parity
c. Timeout Error
d. Error Flags
e. Status Words

2.14.1 Address Parity Logic

In the MBox, an address parity bit is generated for the cache directory and the SBus. The parity bit for the cache directory is generated for physical address bits 14–26 (PA14–26) whenever the cache control updates the cache directory. The cache directory is updated for EBox read requests in preparation of a core read cycle. The parity bit for the SBus is generated for the entire physical memory address (bits 14–35) and the SBus request qualifiers whenever a core request is made by the cache control.

In addition, a parity bit is also written into the page table whenever a page refill operation is executed. This parity bit is picked up from core memory for KI-style page refills and from the EBox for KL-style refills.

Address parity is checked in the MBox for paged memory references, references to cache memory, and references to core memory.

Page table parity is checked for all EBox memory requests to paged memory. If the page check fails, the MBox asserts the PAGE FAIL HOLD flag, transfers the page fail word to the EBus register, and terminates the cycle. The EBox then traps to the microstore page fail routine to read the EBus register and evaluate the failure.

Cache directory parity is checked whenever the cache is referenced. The cache is referenced for both channel and EBox-initiated memory reference requests if CON CACHE LOOK EN is set.

For EBox memory requests the cache is referenced to:

a. Write a word and its page address into the cache.
b. Write the page address into the cache in preparation for a core read cycle.
c. Read a word from the cache.
d. Pick up any valid words during a KI-style page refill operation.
e. Pick up all written words during a write-back operation.

For channel memory requests the cache is referenced to:

a. Invalidate any valid entries during channel write operations to memory.
b. Pick up any valid entries during channel read operations from memory.

If the cache address parity check fails for any of the above references, the MBox sets the CSH ADR PAR ERR FLG which, in turn, disables the cache after the current request is executed to completion.

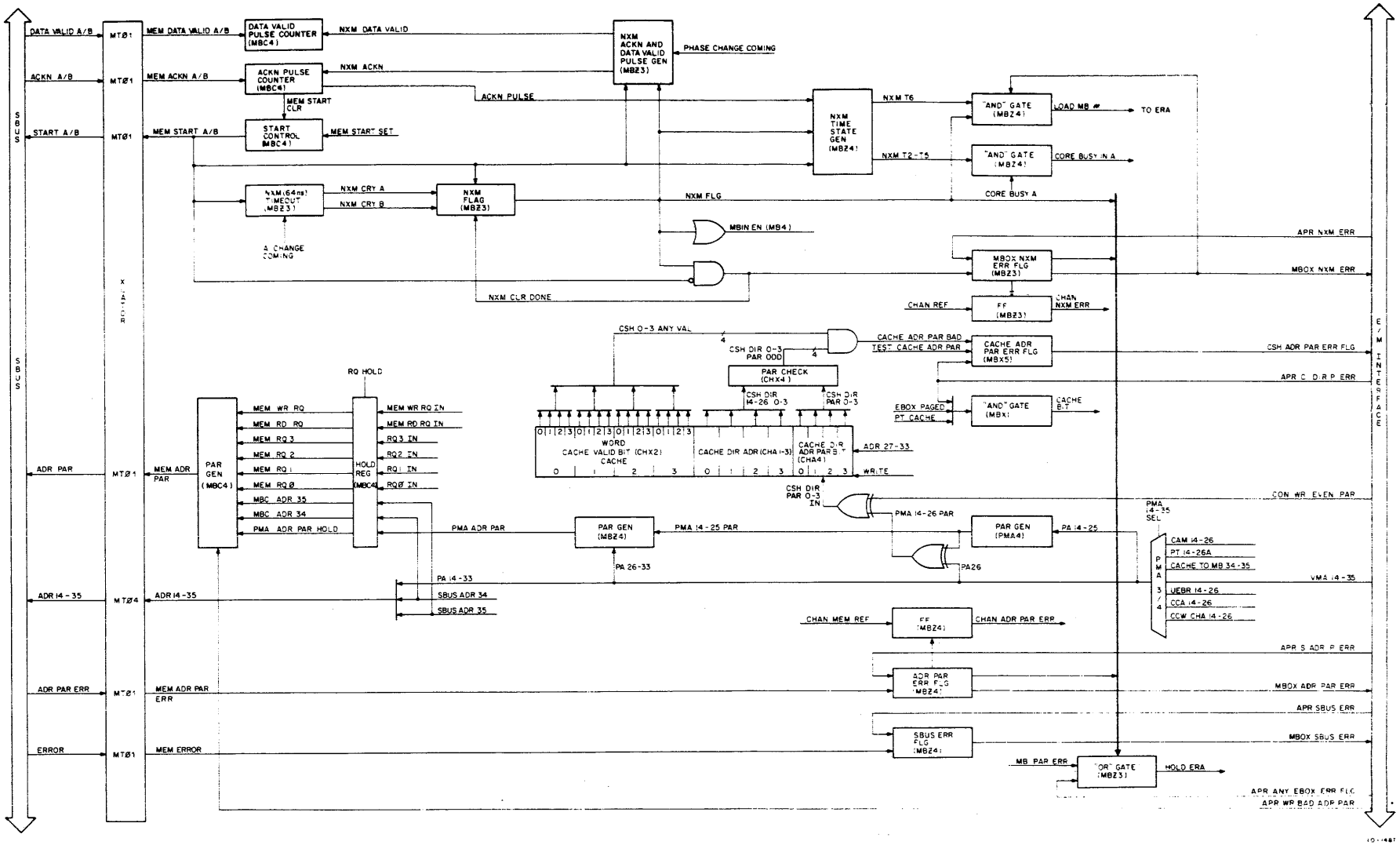


Figure 2-16 MBox Address Parity, NXM, and SBus Error Logic Paths, Logic Diagram

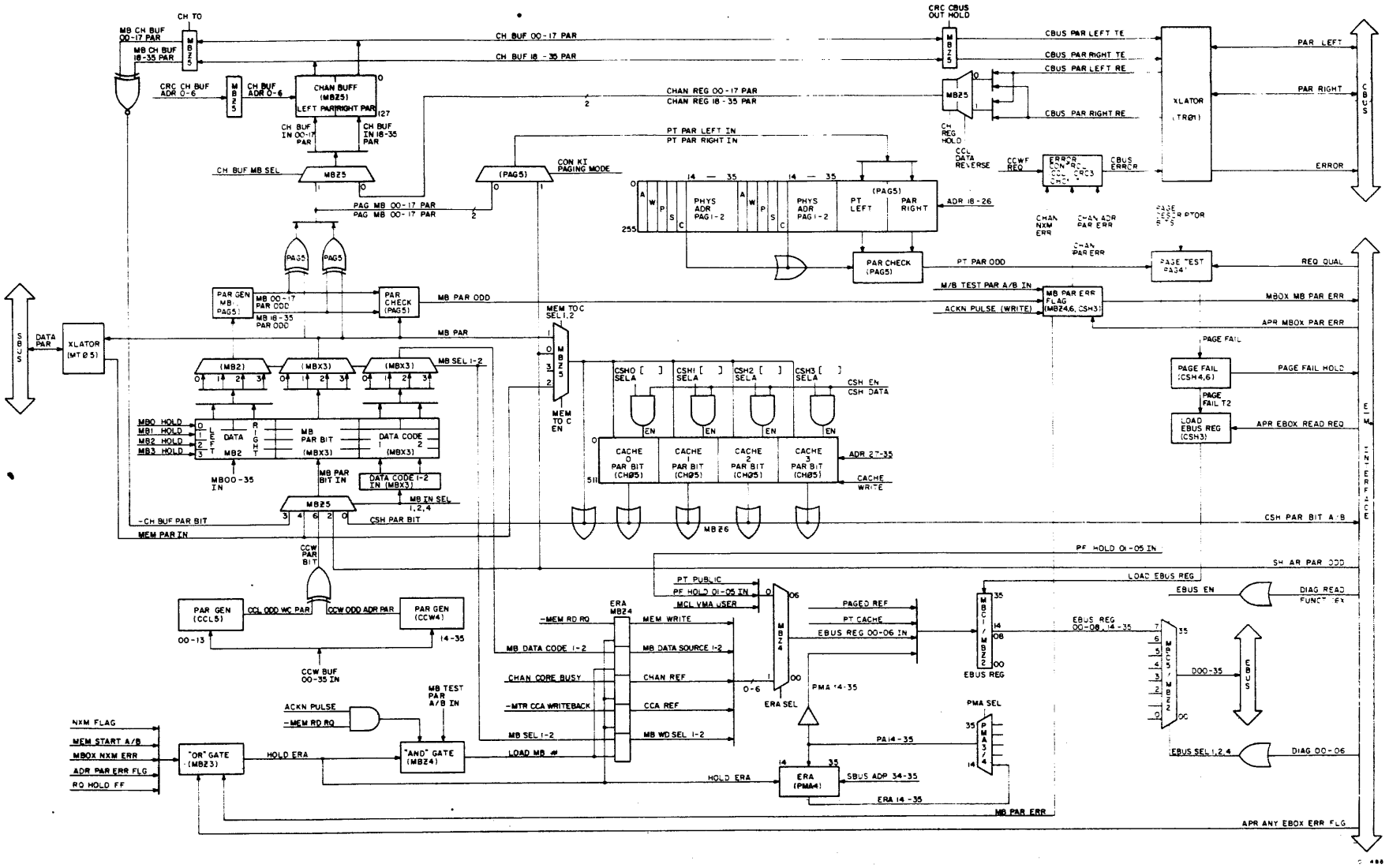


Figure 2-17 MBox Data and Page Table Parity, Path Logic Diagram

When the MBox issues a core read or write cycle, the MBox generates the SBus address parity bit and transfers this bit with the address to the core memory system via the SBus. Parity of the SBus address is checked by the core memory system. If the parity check fails, the core memory system asserts SBUS ADR PAR ERR which, in turn, sets the MBOX ADR PAR ERR flag and holds the ERA register in the MBox. For core read operations, four words of zeros with bad parity are returned by the MA/MB20, which causes the MBOX MB PAR ERR flag in the MBox to be asserted. For core write operations, the data sent to the MA/MB is thrown away, thereby preserving the data in the addressed locations. The DMA20 will not respond other than asserting SBUS ADR PAR ERR if it senses an address parity error. This will cause a NXM error to be detected in the MBox if the request was for the DMA20.

2.14.2 Data Parity Logic

In general, data parity is propagated through the system with the data from source to destination and is checked along the way at various strategic points. In the MBox, data parity is propagated with the data for both EBox and channel-initiated transfers (Refer to simplified data path drawing, Figure 2-5). This figure shows all data sources and destinations for the MBox. Data parity is propagated with the data for all paths except those noted. Data parity is checked in the MBox only at the output of the MBs and then only as data is moved out of the MBs by the cache, core, or channel controls. These controls move data from the MBs to the EBox, cache, page table, CCW buffer, CH buffer, and core memory via the SBus. A parity splitting network is employed between the MBs and the CH buffer and page table (Figure 2-17) to convert full word parity to half word parity; a parity folding network is employed between the CH buffer and the MBs. Each word transferred on the CBus and stored in the page table is associated with two parity bits, one for each half word, while the data word on the SBus is associated with only one parity bit. If a word in an MB has bad parity, not only will the MBOX MB PAR ERR flag set, but the word (or half words) leaving the MB will also contain bad parity when the word is moved out to the SBus, CCW buffer, CH buffer, page table, cache, or EBox AR.

For EBox write requests, a data parity bit is generated by the EBox for the contents of the AR, which are transferred with the AR data to the MBox. If the cache is to be used, the parity bit is stored in the cache along with the data. Parity is not checked in the MBox in this case. However, if a core cycle is required, then the parity bit is transferred to core memory via the MBs along with the data. When core acknowledges the write request for the addressed word, the MBox checks the parity of the word at the output of the MB. If MB parity is not odd, the MBOX MB PAR ERR flag is set and the ERA register is loaded and held. Core memory will then check data parity (DMA20 only), assert SBUS ERROR if parity is not odd, and write the data and parity bit into core. Asserting SBUS ERROR causes the MBOX SBUS ERR flag to be set.

NOTE

Data is written into core whether data parity is good or bad.

If the cache cycle control decides it must execute a write-back cycle, the parity bits associated with the written words in the cache are picked up and are written along with the data into core memory, as described previously for the core write cycle.

For EBox read requests, the parity bits associated with the addressed words in core memory are picked up and transferred to the MBox where they are stored in the cache along with the data. The first word and its parity bit is also transferred to the AR in the EBox. As each word leaves core memory, its parity is checked (DMA20 only). If parity is not odd, SBUS ERROR is asserted by core memory which sets the MBOX SBUS ERR flag in the MBox. As each word and its parity bit is received by the MBox, they are stored in the MBs. When the EBox takes the first word, parity is checked at the output of the MB and in the AR of the EBox. Parity for subsequent words is checked at the output of the MBs as the

cache cycle control moves the words from the MBs to the cache. If the parity check fails for any of the remaining words, the MBOX MB PAR ERR flag is set and the ERA register is loaded and held. When the EBox initiates a read request and the word is found in the cache, the word and its parity bit are simply transferred to the AR where parity is then checked. If the cache cycle control decides it must execute a write-back cycle before satisfying the EBox read request, the parity bits associated with the written words in the cache are picked up and are written along with the data into core memory, as described previously for the EBox write request.

For a channel read request to fetch a CCW, the parity bit associated with the addressed word in core memory is picked up and transferred to the MBox, where it is placed into the MB along with the data. As the word leaves core memory, its parity is checked (DMA20 only). If parity is not odd, SBUS ERROR is asserted by core memory, which then sets the MBOX SBUS ERR flag in the MBox. The channel recognizes that the word was placed into an MB; in response, the channel moves the word into the CCW buffer and causes the MB parity to be checked.

NOTE

Only the CCW is stored in the CCW buffer. The parity bit is not stored in the CCW buffer with the data but is dropped after MB parity is checked.

If the MB parity check failed, the MBOX MB PAR ERR flag is set. The ERA register is loaded and held and CBUS ERROR is asserted.

For a channel read request to move data from memory to the CH buffer, the parity bits associated with the addressed words in core memory (or from the cache, if the words are in the cache) are picked up and transferred to the MBs along with the data. For those words that come from core, parity is checked as they leave core memory (DMA20 only). If the parity check fails, SBUS ERROR is asserted by core memory which, in turn, sets the MBOX SBUS ERR flag in the MBox.

Parity is not checked for those words that are valid in the cache when they are moved from the cache to the MBs. The channel recognizes that the requested words and the associated parity bits were placed in the MBs; in response, the channel moves the words and the parity bits into the CH buffer and causes MB parity to be checked. If the MB parity check fails on any word as it is moved from the MB to the CH buffer the MBOX MB PAR ERR flag is set and the ERA register is loaded and held.

NOTE

CBUS ERROR is not asserted for this case.

In the data parity path from the MBs to the CH buffer, the single data parity bit that was received from core (or the cache) is split into two parity bits, one for each half word. These parity bits are then stored in the CH buffer and are placed on the CBus with the data when the mass storage system requests a word. The mass storage system asserts CBUS REQUEST whenever a word is needed.

For a channel write request to move data from the CH buffer to core memory, the parity bits associated with the addressed words in the CH buffer are picked up and transferred to the MBs along with the data. The CH buffer contains one parity bit for each half word. The two parity bits and the data word are moved into the CH buffer from the CBus when the mass storage system sends a word (asserts CBUS REQUEST). In the data parity path, from the CH buffer to the MBs, the two parity bits are folded into one bit to accommodate the SBus. From the MBs, each word and the associated parity bit is moved to core memory. As each word is transferred, parity is checked at the output of the MB and in core memory. If the parity check fails at the output of the MB, the MBOX MB PAR ERR flag is set and the ERA register is loaded and held. If the parity fails in core memory, MBOX SBUS ERR is asserted by core memory, which in turn causes the MBOX SBUS ERR flag in the MBox to be set.

For a channel write request to store the two status words, parity for each word is generated by the channel. The two status words are held by the CCW buffer after a channel transfer terminates. After the two words and the associated parity bits are transferred to the MBs, they are moved to core memory. As each word is moved to core, parity is checked at the output of the MBs and in core memory, as described for the channel data write request.

The page table can be refilled from core or from the AR. During the KL paging mode, the page refill operation is executed by the EBox microcode. Essentially, the EBox will perform a table lookup to find a valid page address. When a valid address is found, it is written into the page table from the AR. During the KI paging mode, the page refill operation is executed by the MBox automatically. In this case, eight entries are written into the page table from the process table in core memory via the MBs. In either case, the parity bits associated with the entries are transferred along with the data and are written into the page table. During the KL paging mode, parity on the contents of the AR is generated by the EBox and is transferred with the page table entry. The MBox does not check the parity of this transfer before it is written into the page table. During the KI paging mode, the parity bits associated with the addressed words in core memory (or the cache for any valid words) are transferred with the data and parity is checked along the way (in core memory and at the output of the MBs), as described previously for core read operations. In the parity path from the MBs to the page table, a parity-splitting network is used to convert full-word parity to half-word parity. This is done to provide a parity bit for each page table entry. Page table parity is checked whenever the EBox makes a paged memory reference.

2.14.3 Time-out Error

The MBox and the core memory system employ time-out counters to sense incompleted memory cycles and NXM. The time-out duration and the location of the time-out networks are itemized in Table 2-11.

Table 2-11 Memory Timeouts

Duration μs		
25 MHz	30 MHz	Location
10.240	8.448	MA/MB (Internal)
36.000	29.700	DMA (External)
81.900	67.567	MBox

If a core cycle is started by either internal or external core memory, and the cycle is not completed within the specified time-out duration, the core memory system asserts SBUS ERROR, which in turn sets the MBOX SBUS ERR flag in the MBox. The time-out is activated whenever the MBox initiates a core cycle by asserting SBUS START. When SBUS START is cleared at the end of the core cycle, the time-out is reset. Consequently, if all the requested words are not acknowledged by the core memory system, the time-out is allowed to expire, which in turn causes the MBOX NXM ERR flag to be set. Besides reporting errors due to hardware failures in the core memory system, the MBOX NXM ERR flag can be used to find out how much memory is connected to the system.

2.14.4 Error Flags

The following flags are implemented in the MBox for error reporting purposes:

a. PAGE FAIL HOLD
b. CSH ADR PAR ERR
c. MBOX ADR PAR ERR
d. MBOX MB PAR ERR
e. MBOX SBUS ERR
f. MBOX NXM ERR
g. CBUS ERR

2.14.4.1 PAGE FAIL HOLD Flag – The PAGE FAIL HOLD flag is set when the MBox senses a page table parity error or when the page test fails. Accessability of a given page and page table parity is checked only for EBox memory read and write requests to paged memory. When the flag is set, the Page Fail Word is also loaded into the EBus register so that it can be read by the EBox. Setting the PAGE FAIL HOLD flag causes the EBox to trap to the microcode page fail handler. The flag is cleared automatically when the current cache EBox cycle is completed.

2.14.4.2 CSH ADR PAR ERR Flag – The CACHE ADR PAR ERR flag is set when the MBox senses a cache directory parity error. Parity is checked on the address in the directory whenever the cache is referenced. If the CACHE ADR PAR ERR flag is set, the APR C DIR P ERR flag in the EBox is set on the next EBox clock tick to interrupt the Priority Interrupt (PI) system if the APR flag is enabled. The APR flag is cleared by executing a CONO APR instruction. The MBox error flag is cleared by virtue of setting the EBox APR C DIR P ERR flag.

2.14.4.3 MBOX ADR PAR ERR Flag – The MBOX ADR PAR ERR flag is set when the core memory system senses an address parity error. Parity is checked on the SBus address and the request qualifiers whenever the MBox initiates a core cycle. If the MBOX ADR PAR ERR flag is set, the contents of the ERA is held and the APR S ADR P ERR flag in the EBox is set on the next EBox clock tick to interrupt the PI system if the APR flag is enabled. The APR flag is cleared by executing a CONO APR instruction. The MBox error flag is cleared by virtue of setting the EBox APR S ADR P ERR flag.

2.14.4.4 MBOX MB PAR ERR Flag – The MBOX MB PAR ERR flag is set when the MBox senses an MB parity error. Parity is checked on the data in the MB whenever data is moved out of the MB to the AR, cache, page table, CH buffer, or SBus. If the MBOX MB PAR ERR flag is set, the contents of the ERA are held and the APR MB PAR ERR flag in the EBox is set on the next EBox clock tick to interrupt the PI system, if the APR flag is enabled. The APR flag is cleared by executing a CONO APR instruction. The MBox error flag is cleared by virtue of setting the EBox APR MB PAR ERR flag.

2.14.4.5 MBOX SBUS ERR Flag – The MBOX SBUS ERR flag is set when the core memory system senses a data parity error or times out. Parity is checked on the data during both core read and core write cycles (DMA20 only). The core memory system times out if all requested words are not acknowledged, which would occur in the event of a hardware failure. If the MBOX SBUS ERR flag is set, the APR SBUS ERR flag in the EBox is set on the next EBox clock tick to interrupt the PI system, if the APR flag is enabled. The flag is cleared by executing a CONO APR instruction. The MBox error flag is cleared by virtue of setting the EBox APR SBUS ERR flag.

2.14.4.6 MBOX NXM ERR Flag – The MBOX NXM ERR flag is set when the MBox times out. The NXM timer is started when a core cycle is initiated (SBUS START asserted) and is reset when all requested words are accounted for (SBUS START clears). If all requested words are not acknowledged by the core memory system, the NXM time-out expires and sets the MBOX NXM ERR flag. If the MBOX NXM ERR flag is set, the ERA is loaded and held; and APR NXM ERR flag in the EBox is set on the next EBox clock tick to interrupt the PI system, if the flag is enabled. The flag is cleared by executing a CONO APR instruction. The MBox error flag is cleared by virtue of setting the EBox APR NXM ERR flag.

2.14.4.7 CBUS ERR Flag – The CBUS ERR flag is asserted if an error is sensed by the MBox or by the core memory system when a channel request to fetch a CCW is executed. The errors that are sensed include:

a. MEM ADR PAR ERR
b. MB PAR ERR
c. NXM ERR

Asserting CBUS ERR causes a status bit in the controller of the selected channel to be set.

NOTE

Address and data parity are not checked for regular data transfer operations or for memory store operations. Only NXM will be sensed and reported on the CBUS ERROR line for these operations.

2.14.5 Status Words

One of two status words are formed and stored by the MBox in the event an error is sensed:

a. Page Fail Word
b. Error Address (ERA)

One or the other is stored in a register so that the EBox can read the word and evaluate the failure. In the case of a page test failure, which includes the page table parity check, the PAGE FAIL HOLD flag is set and the Page Fail Word is loaded into the EBus register. The format of the Page Fail Word is shown in Figure 2-18. This register is read by the EBox by asserting the diagnostic register read function for register 167_8.

In the case of a parity, time-out, or NXM error, the corresponding error flags are set and the error address and associated status bits are loaded into the ERA register. The format of this word is shown in Figure 2-19. This register is read by the EBox when an RDERA (BLKI, PI) instruction is executed.

2.15 DIAGNOSTIC REGISTERS

There are 16 diagnostic registers in the MBox (Figure 2-20 and Tables 2-12 through 2-27). They are essentially test points for collecting MBox snapshots on a per-clock-tick basis, or to monitor an individual signal to determine or validate its individual characteristics versus function. The Diagnostic registers can be read by the privileged PDP-11 front end processor.

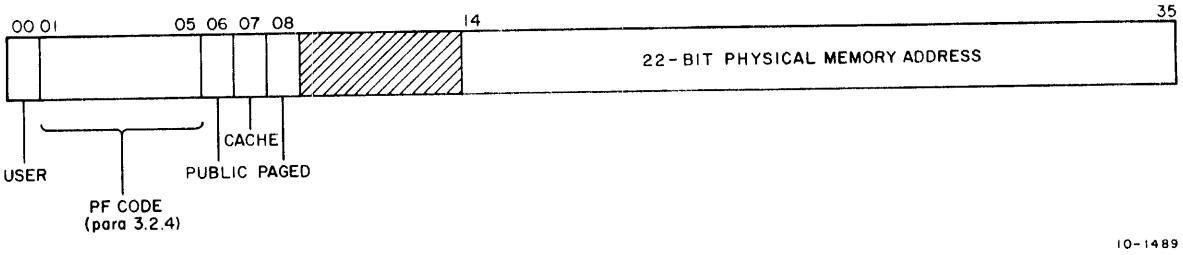


Figure 2-18 Page Fail Word Format

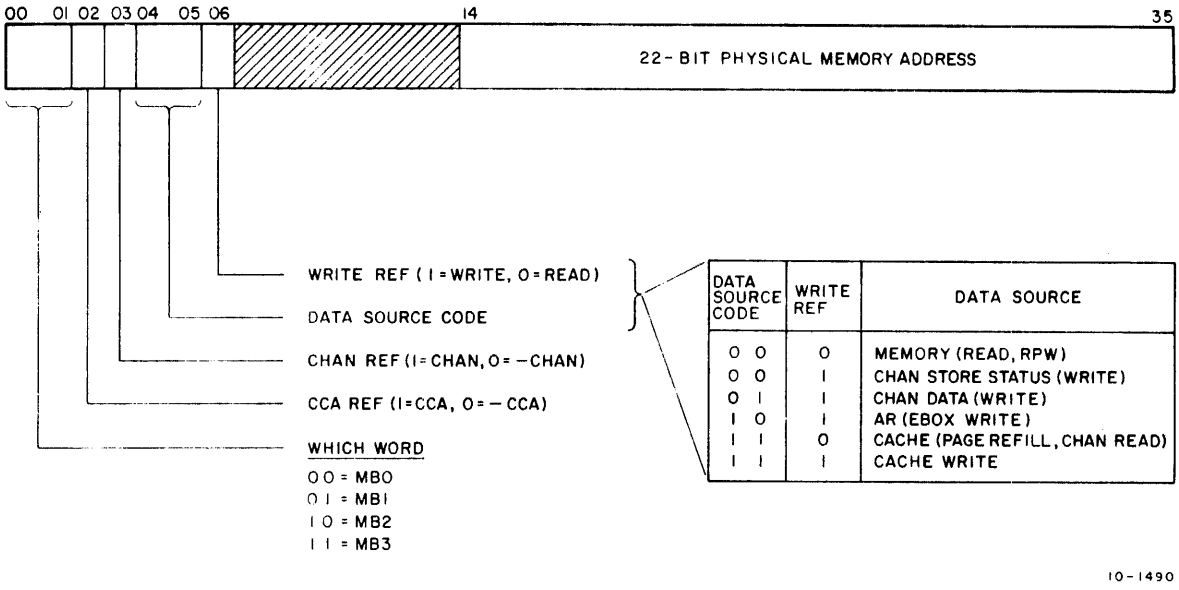


Figure 2-19 ERA Word Format

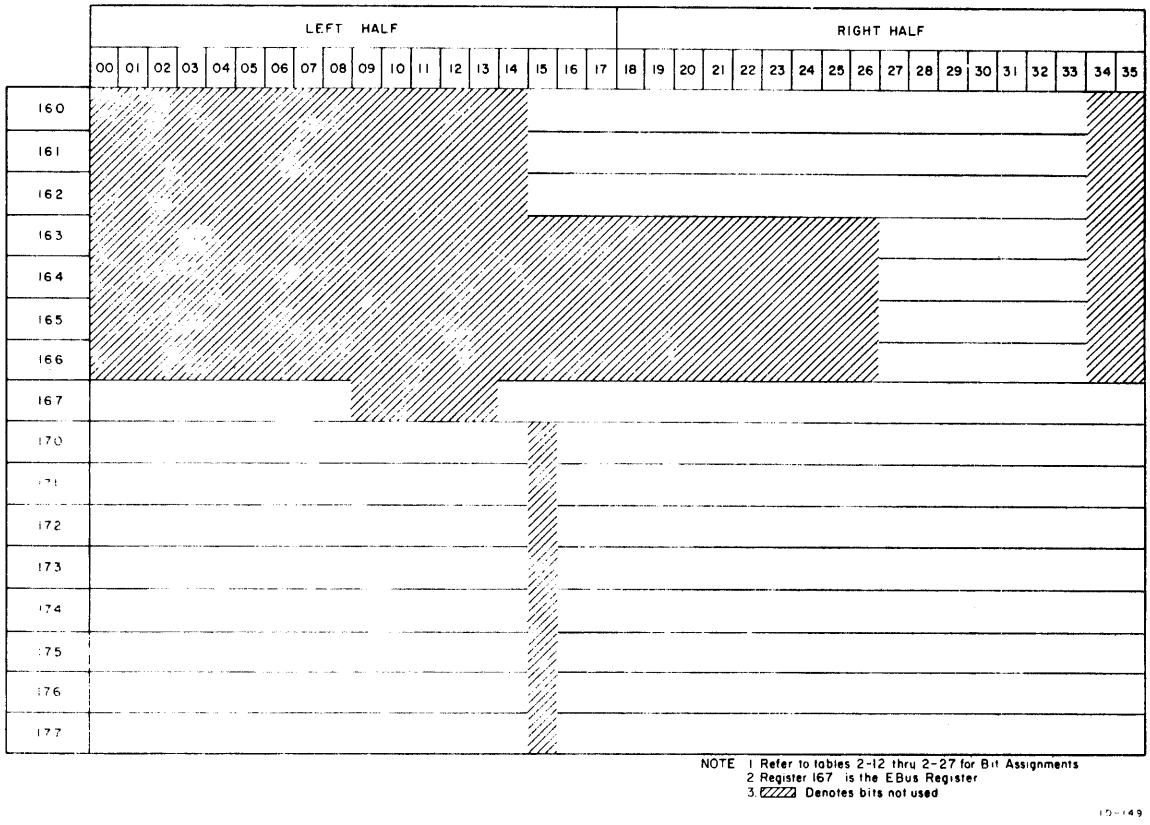


Figure 2-20 MBox Diagnostic Register Bit Maps

Table 2-12 Diagnostic Register 160_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
15	MBZ1	CORE BUSY H	18	MBZ5	MB PAR BIT IN H
16	MBZ4	CHAN PAR ERR L	19	MBZ1	CSH EN CSH DATA L
17	SHD1	SH AR PAR ODD A H	20	MBZ1	MB IN SEL 1 H
			21	MBZ3	NXM ACKN H
			22	MBZ1	CHAN CORE BUSY H
			23	MBZ3	NXM ANY L
			24	MBZ4	NXM T6-7 L
			25	MBZ3	CHAN NXM ERR L
			26	PAG5	PAG MB 18–35 PAR H
			27	MBC5	FORCE VALID MATCH 0 H
			28	MBC5	FORCE VALID MATCH 1 H
			29	MBC5	FORCE VALID MATCH 2 H
			30	MBC5	FORCE VALID MATCH 3 H
			31	MBC1	WRITE OK H
			32	MBC2	CSH ADR WR PULSE H
			33	MBC2	CSH DATA CLR DONE IN L

Table 2-13 Diagnostic Register 161_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
15	MBZ4	MBOX ADR PAR ERR L	18	MBZ6	CSH PAR BIT H
16	MBZ5	CBUS PAR LEFT TE H	19	MBZ1	MEM TO C DIAG EN L
17	MT05	MEM PAR IN H	20	MBZ1	MB IN SEL 2 H
			21	MBZ1	MBZ1 RD-PSE-WR REF L
			22	MBZ3	MBOX NXM ERR L
			23	MBZ3	CHAN MEM REF L
			24	MBZ4	MBOX SBUS ERR L
			25	MBZ3	NXM DATA VAL L
			26	MBZ6	CSH PAR BIT A H
			27	MBC2	CSH DATA CLR T1 L
			28	MBC2	CSH DATA CLR T2 L
			29	MBC2	CSH DATA CLR T3 L
			30	MBC2	CSH SEL LRU H
			31	MBC2	CSH VAL WR PULSE H
			32	MBC2	CSH WR WR PULSE H
			33	MBC2	RQ HOLD FF H

Table 2-14 Diagnostic Register 162_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
15	MBZ4	CHAN ADR PAR ERR L	18		(Not Used)
16	MBZ5	CBUS PAR RIGHT TE H	19	MBZ1	CHAN READ L
17	MBZ5	CSH PAR BIT IN H	20	MBZ1	MB IN SEL 4 H
			21	MBZ1	MEM BUSY H
			22	MBZ3	HOLD ERA L
			23	MBZ4	NXM T2 H
			24	MBZ4	MBOX MB PAR ERR L
			25	PAG5	PAG MB 00–17 PAR H
			26	MBZ6	CSH PAR BIT B H
			27	MBC2	CACHE WR 00 A H
			28	MBC2	CACHE WR 09 A H
			29	MBC2	CACHE WR 18 A H
			30	MBC2	CACHE WR 27 A H
			31	MBC2	SBUS ADR HOLD H
			32	MBC3	A CHANGE COMING A L
			33	MBC3	ANY SBUS RQ IN L

Table 2-15 Diagnostic Register 163_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
			27	MBC3	B CHANGE COMING L
			28	MBC3	CORE BUSY A H
			29	MBC3	CSH VAL SEL ALL H
			30	MBC3	CSH VAL WR DATA H
			31	MBC3	CSH WR SEL ALL H
			32	MBC3	CSH WR WR DATA H
			33	MBC3	DATA VALID A OUT H

Table 2-16 Diagnostic Register 164_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
			27	MBC3	DATA VALID B OUT H
			28	MBC3	MBC INH 1ST MB REQ H
			29	MBC3	MEM TO C EN L
			30	MBC3	PHASE CHANGE COMING L
			31	MBC4	ACKN PULSE L
			32	MBC4	CORE ADR 34 H
			33	MBC4	CORE ADR 35 H

Table 2-17 Diagnostic Register 165_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
			27	MBC1	CAM SEL 1 H
			28	MBC1	CAM SEL 2 H
			29	MBC4	CORE DATA VALID -1 L
			30	MBC4	CORE DATA VALID -2 L
			31	MBC4	CORE DATA VALID L
			32	MBC4	CORE RD IN PROG H
			33	MBC4	MEM ADR PAR H

Table 2-18 Diagnostic Register 166_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
			27	MBC4	MEM RD RQ B H
			28	MBC4	MEM RQ 0 H
			29	MBC4	MEM RQ 1 H
			30	MBC4	MEM RQ 2 H
			31	MBC4	MEM RQ 3 H
			32	MBC4	MEM START L
			33	MBC4	MEM WR RQ L

Table 2-19 Diagnostic Register 167_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00 08	MBZ1	EBUS REG 00 08 H	18 26	MBZ1	EBUS REG 18 26 H
14 17	MBZ1	EBUS REG 14 17 H	27 33	MBC1	EBUS REG 27 33 H
			34,35	MBZ1	EBUS REG 34, 35 H

Table 2-20 Diagnostic Register 170_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 0 H	18	CCL5	CCL WC GE4 H
01	CRC4	CRC RESET IN L	19	CCL5	CCL WC = 0 L
02	CRC4	CRC MEM STORE ENA L	20	CHX2	CSH 0 ANY VAL L
03	CRC4	CRC DONE IN H	21	CHX3	CSH USE IN 0 H
04	CRC4	CRC STORE IN H	22	CSH5	PAGE REFILL COMP L
05	CCW4	CCW WD READY H	23	CSH6	CACHE WR IN H
06	CCW6	CCW CCWF REQ ENA H	24	CSH6	MBOX PT DIR WR L
07	CCW6	CCW MEM STORE ENA H	25	CSH2	CSH WR TEST L
08	CCW5	CCW ACT FLAG REQ ENA H	26	CSH3	ANY VAL HOLD H
09	CCW3	CCW ALU C8 OUT H	27	CSH4	CSH DATA CLR DONE L
10	CCW3	CCW ALU C2 OUT H	28	CSH4	CSH REFILL RAM WR L
11	CH1	CH T0 H	29	CSH4	CSH EBOX T3 L
12	CHC5	CBUS SEL 0 E H	30	MBX1	CACHE BIT H
13	CHC1	CHX RESET H	31	MBX1	CCA REQ L
14	CHC2	CH RESET INTR H	32	MBX4	CSH WR WD 2 EN H
16	CCL5	CCL ODD WC PAR H	33	MBX5	MB REQ IN H
			34	MBX5	MBX MEM TO C EN L
			35	MBX5	RQ 1 IN H

Table 2-21 Diagnostic Register 171_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 1 H	18	CCL3	CCL ALU MINUS L
01	CRC4	CRC RH20 ERR IN H	19	CCL4	CCL CH TEST MB PAR L
02	CRC4	CRC OVN ERR IN H	20	CHX2	CSH 1 ANY VAL L
03	CRC4	CRC SHORT WC ERR H	21	CHX3	CSH USE IN 1 H
04	CRC4	CRC LONG WC ERR H	22	CSH5	CHAN RD T5 L
05	CCW4	CCW WD0 REQ H	23	CSH6	CSH WR DATA RDY L
06	CCW4	CCW WD1 REQ H	24	CSH4	PAGE FAIL T2 L
07	CCW4	CCW WD2 REQ H	25	CSH6	CSH EBOX LOAD REG H
08	CCW4	CCW WD3 REQ H	26	CSH7	CSH FILL CACHE RD L
09	CCW1	CCW MEM ADR = 0 H	27	CSH5	CHAN WR T5 L
10	CCW6	CCW CCWF WAITING H	28	CSH3	MB WR RQ CLR NXT L
11	CHC1	CH T1 H	29	CSH4	CSH EBOX T1 L
12	CHC5	CBUS SEL 1 E H	30	MBX2	CACHE TO MB 34 H
13	CHC1	CHX START H	31	MBX1	CCA SEL 1 H
14	CHC2	CH START INTR H	32	MBX4	CSH WR WD 3 EN H
16	CCL3	CCL MB RIP A H	33	MBX2	MB SEL 1 H
			34	MBX3	MEM DIAG L
			35	MBX5	RQ 2 IN H

Table 2-22 Diagnostic Register 172_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 2 H	18	CCL3	CCL MB REQ T2 H
01	CRC3	CRC READY IN H	19	CCL4	CCL REVERSE H
02	CRC3	CRC LAST WORD IN H	20	CHX2	CSH 2 ANY VAL L
03	CRC3	CRC ERR IN H	21	CHX3	CSH USE IN 2 H
04	CRC3	CRC REVERSE IN H	22	CSH6	CHAN WR CACHE L
05	CCW3	CCW ACT CTR 0 EN H	23	CSH6	CCA CYC DONE L
06	CCW3	CCW ACT CTR 1 EN H	24	CSH5	CHAN T4 L
07	CCW3	CCW ACT CTR 2 EN H	25	CHX3	CSH LRU 2 H
08	CCW1	CCW BUF ADR 0 L	26	CSH1	READY TO GO A H
09	CCW1	CCW BUF ADR 1 L	27	CSH6	CSH USE HOLD H
10	CCW1	CCW BUF ADR 2 L	28	CSH1	CSH CCA CYC L
11	CHC1	CH T2 H	29	CSH1	CSH EBOX REQ EN L
12	CHC5	CBUS SEL 2 E H	30	MBX2	CACHE TO MB 35 H
13	CHC1	CHX DONE H	31	MBX1	CCA SEL 2 H
14	CHC2	CH DONE INT R H	32	MBX1	FORCE NO MATCH H
16	CCL3	CCL CCWF T2 H	33	MBX2	MB SEL 2 H
			34	MBX5	MEM RD RQ IN H
			35	MBX5	RQ 3 IN H

Table 2-23 Diagnostic Register 173_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 3 H	18	CCL4	CCL CH MB SEL 1 H
01	CRC2	CRC ACT CTR 0R H	19	CCL3	CCL AF T2 L
02	CRC2	CRC ACT CTR 1R H	20	CHX2	CSH 3 ANY VAL L
03	CRC2	CRC ACT CTR 2R H	21	CHX3	CSH USE IN 3 H
04	CRC2	CRC RAM CYC H	22	CSH2	ONE WORD RD L
05–10	CCW2	CCW CHA 30–35 H	23	CSH2	MBOX RESP L
11	CHC1	CH T3 H	24	CSH2	RD PSE 2ND REQ EN L
12	CHC5	CBUS SEL 3 E H	25	CHX3	CSH LRU 1 H
13	CHC1	CHX STORE H	26	CSH5	CSH T1 L
14	CHC2	CH STORE H	27	CSH4	WRITEBACK T1 A H
16	CCL4	CCL CH MB SEL 2 H	28	CSH7	CSH CCA WRITEBACK L
			29	CSH4	CSH EBOX T2 L
			30	MBX4	CACHE TO MB DONE L
			31	MBX2	CHAN WR CYC L
			32	MBX3	MEM DATA TO MEM H
			33	MBX2	MB SEL HOLD H
			34	MBX3	MEM TO C SEL 1 H
			35	MBX2	SBUS ADR 34 H

Table 2-24 Diagnostic Register 174_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 4 H	18	CCL3	CCL CHAN EPT H
01	CRC1	CRC ACT FLAG ENA H	19	CCL4	CCL CHAN TO MEM H
02	CRC5	CRC WR RAM L	20	CHX4	CSH DIR 0 PAR ODD H
03	CRC3	CRC OP CODE 00 H	21	CHX3	CSH USE IN 4 H
04	CRC3	CRC OP CODE 01 H	22	CSH2	E CORE RD RQ L
05–10	CCW2	CCW CHA 24–29 H	23	CSH6	PAGE FAIL HOLD L
11	CHC1	CBUS READY E H	24	CSH5	PAGE REFILL T9, 12 L
12	CHC5	CBUS SEL 4 E H	25	CHA3	CSH 3 ANY WR L
13	CHC1	CHX CTOM H	26	CSH5	CSH T0 L
14	CHC3	CH CTOM H	27	CSH3	CSH ADR PMA EN H
16	CCL3	CCL CHAN REQ H	28	CSH1	CSH EBOX CYC B L
			29	CSH1	CACHE IDLE L
			30	MBX4	CACHE TO MB T2 L
			31	MBX1	CSH CCA INVAL CSH H
			32	MBX3	MB DATA CODE 1 H
			33	MBX6	MB0 HOLD IN H
			34	MBX3	MEM TO C SEL 2 H
			35	MBX2	SBUS ADR 35 H

Table 2-25 Diagnostic Register 175_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 5 H	18	CCL2	CCL ACT FLAG REQ H
01	CRC6	CRC SEL 1D L	19	CCL2	CCL MEM STORE REQ H
02	CRC6	CRC SEL 2D L	20	CHX4	CSH DIR 1 PAR ODD H
03	CRC6	CRC SEL 4D L	21	CHX3	CSH USE ADR 2 H
04	CRC1	CRC AF REQ ENA L	22	CSH2	CSH EBOX RETRY REQ L
05–10	CCW2	CCW CHA 18–23 H	23	CSH6	CSH USE WR EN H
11	CHC1	CBUS LAST WORD E H	24	CSH3	MB TEST PAR A IN L
12	CHC5	CBUS SEL 5 E H	25	CHA3	CSH 1 ANY WR L
13	CHC5	CH SEL 8A H	26	CSH5	CSH T3 L
14	CHC2	CH CONTR REQ H	27	CSH3	MBOX GATE VMA 27–33 H
16	CCL2	CCL CCWF REQ H	28	CSH1	CSH MB CYC L
			29	CSH4	ONE WORD WR T0 L
			30	MBX4	CACHE TO MB T3 L
			31	MBX1	CSH CCA VAL CORE H
			32	MBX3	MB DATA CODE 2 H
			33	MBX6	MB1 HOLD IN H
			34	MBX5	MEM WR RQ IN H
			35	MBX3	SBUS DIAG 3 L

Table 2-26 Diagnostic Register 176, Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC6	CRC CH BUF ADR 6 H	18	CCL2	CCL BUF ADR 3 H
01	CRC1	CRC MEM PTR0 H	19	CCL4	CCL START MEM L
02	CRC1	CRC MEM PTR1 H	20	CHX4	CSH DIR 2 PAR ODD H
03	CRC1	CRC MEM PTR2 H	21	CHX3	CSH USE ADR 3 H
04	CRC1	CRC MEM PTR3 H	22	CSH6	CCA INVAL T4 L
05	CCW3	CCL WC = 3 H	23	CSH5	PAGE REFILL T8 L
06	CCW4	CCL CCW REG LOAD H	24	CSH4	CSH EBOX T0 L
07–10	CCW2	CCW CHA 14–17 H	25	CHA3	CSH 2 ANY WR L
11	CHC1	CBUS ERROR E H	26	CSH5	CSH T2 L
12	CHC5	CBUS SEL 6 E H	27	CSH2	E CACHE WR CYC H
13	CHC1	CH MB REQ INH H	28	CSH7	CSH E WRITEBACK L
14	CHC1	CH REVERSE H	29	CSH5	PAGE REFILL T4 L
16	CCL4	CCL STORE CCW H	30	MBX4	CACHE TO MB T4 A L
			31	MBX4	CSH WR WD 0 EN H
			32	MBX3	MB PAR H
			33	MBX6	MB2 HOLD IN H
			34	MBX3	REFILL HOLD H
			35	MBX3	SBUS DIAG CYC L

Table 2-27 Diagnostic Register 177_8 Bit Assignments

LEFT HALF			RIGHT HALF		
Bit No.	Source	Signal Name	Bit No.	Source	Signal Name
00	CRC1	CRC PTR DIF = 0 H	18	CCL6	CCL CSH CHAN CYC L
01	CRC6	CRC CH ADR 0C L	19	CCL3	CCL MEM PTR EN H
02	CRC6	CRC CH ADR 1C L	20	CHX4	CSH DIR 3 PAR ODD H
03	CRC6	CRC CH ADR 2C L	21	CHX3	CSH USE ADR 4 H
04	CRC6	CRC CH ADR 3C L	22	CSH6	PAGE REFILL ERROR L
05	CCW6	CCW RAM ADR 1 H	23	CSH6	DATA DLY 1 L
06	CCW6	CCW RAM ADR 2 H	24	CSH4	PAGE FAIL DLY H
07	CCW6	CCW RAM ADR 4 H	25	CHA3	CSH 0 ANY WR L
08	CCW3	CCL WC = 1 H	26	CSH5	PAGE REFILL T10 L
09	CCW3	CCL WC = 2 H	28	CSH2	RD PAUSE 2ND HALF L
10	CCW4	CCW ODD ADR PAR H	29	CSH4	CSH EBOX WR T4 L
11	CHC1	CH CBUS REQ H	30	MBX1	CCA ALL PAGES CYC H
12	CHC5	CBUS SEL 7 E H	31	MBX 4	CSH WR WD 1 EN H
13	CHC2	CH CONTR CYC H	32	MBX2	MB REQ HOLD H
14	CHC2	CH START H	33	MBX6	MB3 HOLD IN H
16	CCL1	CCL ERR REQ H	34	MBX5	RQ 0 IN H
			35	MBX4	WRITEBACK T2 L

SECTION 3
LOGIC DESCRIPTIONS

3.1 INTRODUCTION

This section contains a logic description of each functional element of the MBox. These functional elements are introduced in Section 1 and include the following:

a. Pager
b. Cache, Cache Control, and Use Logic
c. Cache Clearer Control
d. MB Control
e. Core Control
f. Channel Control

The logic description covers not only the logic itself, but also how the logic operates in the functional context detailed in Section 2.

3.2 PAGER

The pager consists of two hardware tables, associated address, enable and write drivers and combinational logic for detecting illegal page references (Figure 3-1). One table serves as a Directory and the other as the page table. The directory contains 128 locations for storing virtual section numbers and the page table contains 512 locations for storing physical page numbers. Each directory entry implicitly identifies four page table entries. These tables also contain status bits to identify valid entries and access privileges. If the virtual section address matches the contents in the directory and the NOT VALID bit is cleared, then the corresponding four entries in the page table are current for the running process. The entries themselves may show the page to be accessible and legal for transforming the virtual address to the physical address.

When the EBox makes a paged memory reference, the page table and its directory are addressed by a function of the virtual user/executive section and page address, resulting in a modified page address. EBOX USER and bit 17 of the virtual section address are Exclusive-ORed with bits 19 and 20 of the virtual page address to modify bits 19 and 20 of the PT address as a function of the section number and the EXEC/USER address space. This modified page address is used to distribute the entries in the table for different sections (refer to HASH chart, Figure 3-2). This deters identical page entries from different sections from occupying the same table locations and therefore minimizes conflict and additional memory references (thrashing) when switching sections during KL paging mode. In the KI mode, references outside section 0 will not occur. The directory table is addressed by the seven high-order bits of PT address and the page table is addressed by all nine PT address bits. Therefore, for a given virtual address, one directory entry and one page tables entry are selected. When the pager is addressed by the EBox, a comparison is simultaneously made to determine if the directory entry (virtual section address) is valid and the same as the virtual address presented by the EBox. If a match occurs, the corresponding four entries in the page table are valid.

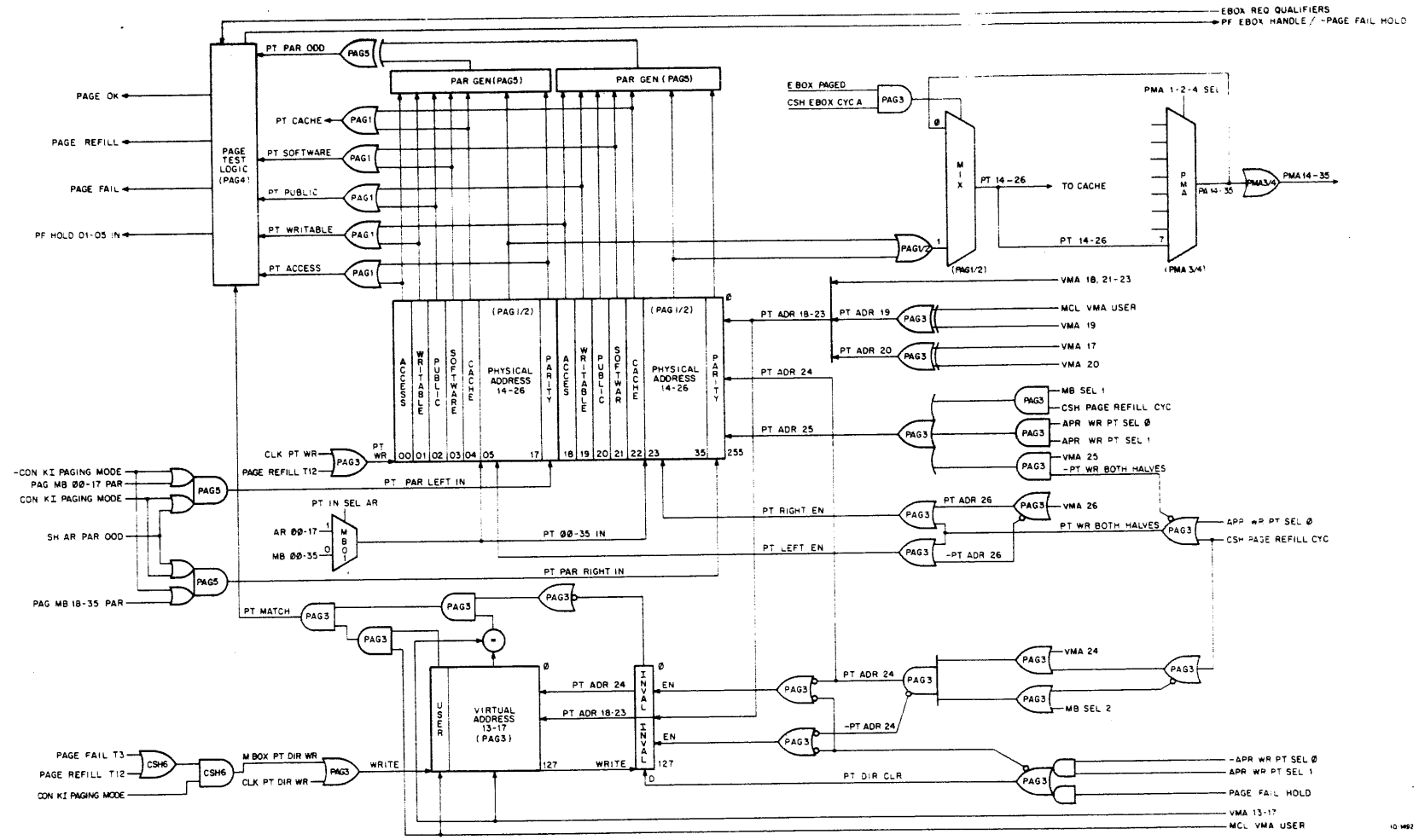


Figure 3-1 Pager, Simplified Logic Diagram

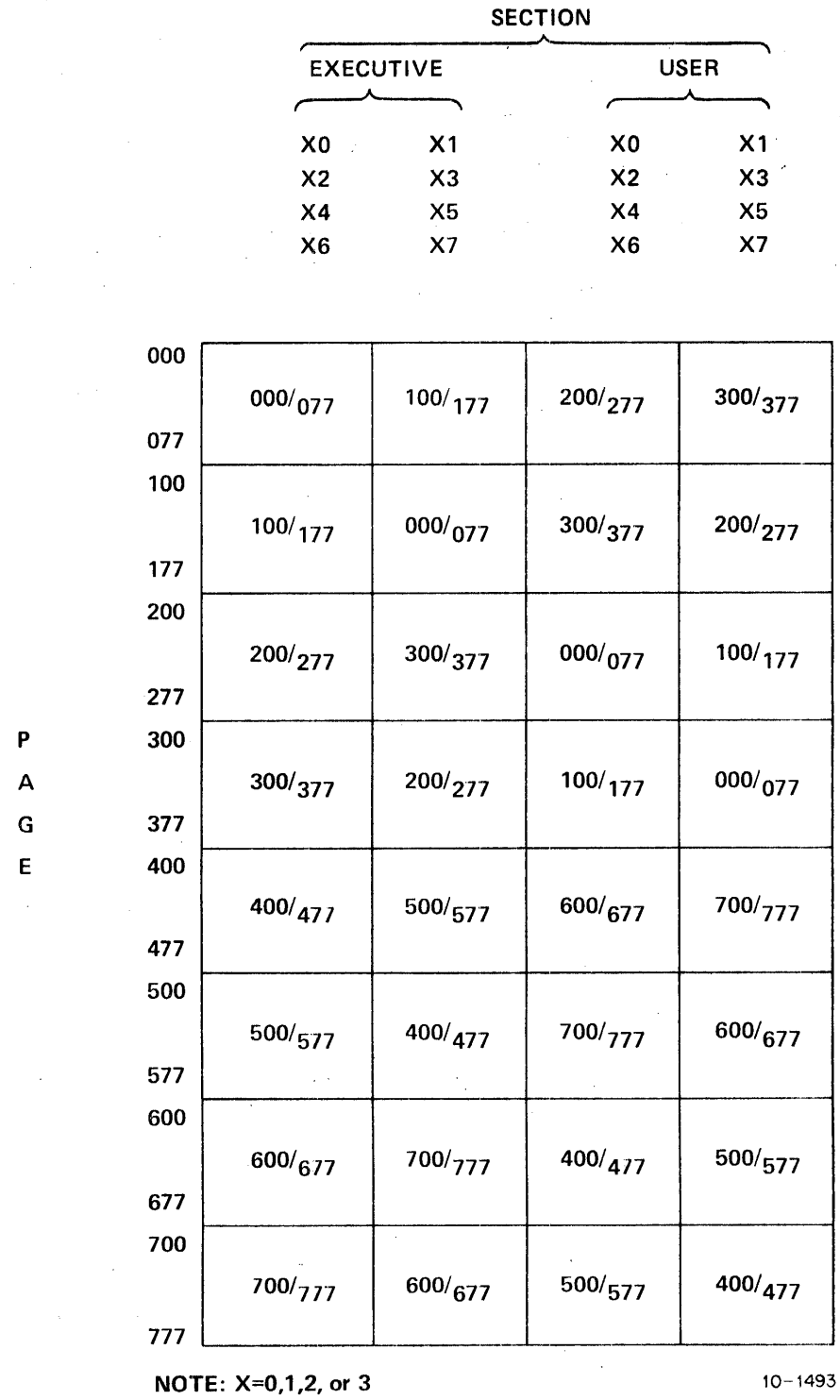


Figure 3-2 Page Table Address Hash Function

The directory table, in addition to containing the virtual section address of the corresponding four physical addresses in the page table, also contains a USER bit and a NOT VALID bit. The USER bit indicates whether the corresponding four entries in the page table are for the user mode or for the executive mode. The NOT VALID bit, when cleared, indicates that the entry is current; that is, it is valid for the current user program (has been written and validated for current user program). The NOT VALID bit of all directory entries is set when another user program starts to run and is cleared as transfers are made to the hardware tables for a given user program.

The first five bits (ACCESS, PUBLIC, WRITABLE, SOFTWARE, and CACHE) of each page table entry are page descriptor bits that specify what type of entry (what kind of page) it is. These bits, along with the physical address, are transferred from the core table to the hardware table when the user program references a page that does not have a valid entry in the directory, or if the ACCESS bit of the page table entry is cleared.

ACCESS bit	KI Mode: Entry has been brought in from core. This bit is set in the core table by the monitor when that page is brought into core. If this bit is not set in the core table, the corresponding user page is still on mass storage. KL Mode: Entry was placed into the page table by the EBox.
PUBLIC bit	Specifies a public page if bit is set; if bit is cleared the page is concealed and access from public mode is not permitted unless access is made via a portal instruction. Referencing the correct portal instruction causes the processor to switch from public to concealed mode. In the concealed mode, the public bit is not set. An instruction that clears the Public flag of the PC word causes the processor to switch back to the public mode. A reference to a concealed page that is not a legal entry point (portal) while the processor is in the public mode, causes a page failure.
WRITABLE bit	This bit must be set to be able to write into the page. If the bit is cleared and a write operation is attempted into this page, a page fault occurs. This bit is typically used to protect shared programs.
SOFTWARE bit	If a page just brought into core is writable, but has not been written into, there is no need to write it back out on mass storage. The SOFTWARE bit is used to identify those pages that are writable but have not been written into and consequently those that do not have to be swapped out to mass storage.
CACHE bit	Specifies that this page should or should not be placed in the cache. When cleared, the page is not to be placed into the cache but must be maintained in core. This permits two processors to use the data in the page. When the bit is set, the page is maintained in the cache. The monitor decides whether the cache bit of a given page is to be set or cleared.

These bits, along with the request qualifiers presented by the EBox, are used to determine whether a given reference by the user program or executive program is legal.

3.2.1 Page Refill

The Page Refill condition is sensed during the KI paging mode when the following conditions are all true:

a. A paged reference is made.
b. No PT match occurred or the ACCESS bit of the entry in the page table is cleared.
c. No Page Refill error occurred.

A Page Refill condition exists when a paged reference is made by the EBox and an entry is not found in the page table before a refill cycle is started. After a Page Refill cycle is executed and a valid entry is still not found, a hardware failure is implied and a page fail trap occurs.

A paged reference can occur in either user or executive mode.

A user paged reference is sensed when the following conditions are all true:

a. VMA User bit is set
b. Not a UEBR (User/Exec Base Register) reference
c. Not an AC (Accumulator) reference
d. Not an illegal entry

An executive paged reference is sensed when the following conditions are all true:

a. VMA User bit is not set
b. Not a UEBR reference
c. Not an AC reference
d. Not an illegal entry
e. Not an executive unpaged reference (Kernel mode)

A PT match does not occur when the 128-location directory does not have a valid entry (NOT VALID bit set) that matches the virtual section address and user bit presented by the EBox with the request.

A refill error occurs when a Page Refill cycle, in response to a Page Refill condition, was already executed and a Page Refill condition is sensed a second time. This implies a hardware failure.

3.2.2 Page OK

A Page OK condition, which is required for all paged references, is sensed when any of the following conditions are true:

a. The reference is not an illegal entry and it is a UEBR reference.

b. The page was found in the table, the previous reference was not an illegal entry, and the page is public and writable or it is not being written into, or it is a Kernel mode reference.

c. The executive page is found, and it is not going to be written into. This condition applies to references to concealed (not public) pages. The supervisor can read but not write into concealed pages.

NOTE

For cases described in (b) and (c) above, the page table parity check must also pass to obtain a Page OK condition.

3.2.3 Page Fail

A Page Fail condition is sensed when any of the following conditions are true.

a. The directory does not contain a valid entry during KL paging mode.

b. A non-accessible paged reference is made (ACCESS bit is cleared). During the KI paging mode, this indicates that the page is not in core and a reference to mass storage is required. During the KL paging mode, this indicates that the page is not in the hardware page table and a reference to core is required.

c. The previous reference was an illegal entry.

d. The reference is a private unpaged executive reference.

e. A refill error occurred during a paged reference.

f. The referenced page is not writable and a write operation is attempted.

g. The reference page is concealed (not public), and a write operation is attempted in a paged executive page, or a portal instruction was not used to enter a concealed paged user page.

3.2.4 Page Fault (PF) Codes

Whenever a page fault is sensed, the physical address and five Page Fail (PF HOLD 01-05 IN) bits are transferred to the EBox, which then stores these bits and the address in the process (user or executive) table. PF EBOX HANDLE qualifies the meaning of these bits for the KI and KL paging modes. The logic levels for the PF bits are provided by the PF HOLD combinational logic. Five bits permit the encoding of 32 different fault conditions. However, only a few codes are meaningful. Refer to the PF truth table (Table 3-1) for the definition of legal fault codes. After the physical address and PF code is stored in the process table, the monitor will jump to the appropriate page fault handler (identified by the PF code) to resolve the fault.

Table 3-1 Page Fault (PF) Code Truth Table

PF EBox Handle	PF Code 01 F	02 A	03 W	04 S	05 T	OCT	Error Type	Mode KI	Mode KL
0	0	0	X	X	X	0X	No Access	X	
0	0	1	0	X	1	11/13	Write Failure	X	
0	1	0	0	0	1	21	Proprietary Violation	X	X
0	1	0	0	1	0	22	Page Refill Error	X	
0	1	0	0	1	1	23	Address Break	X	X
0	1	0	1	0	1	25	PT Parity Error	X	X
1	0	0	X	X	X	0X	No PT Entry		X
1	0	1	0	X	1	11/13	Write Failure		X
1	1	0	X	X	X	2X	No PT DIR Entry		X

Notes: 1. Only meaningful codes are given above
2. X denotes arbitrary (don't care) conditions

3.2.5 Operating Modes

The pager is designed to operate in two different modes: KI and KL paging modes. The EBox specifies which mode the pager is to operate in, by asserting or negating CON KI PAGING MODE. When asserted, the pager will operate in the KI mode; when negated, the pager will operate in the KL mode. In the KI paging mode, page refills are executed by the MBox using the KI10 format page pointers in the process tables. Extended addressing will not be in effect for this style of paging. In the KL paging mode, page refills are executed by the EBox in response to a signal from the MBox, using the KL10 format page pointers. Extended addressing may or may not be in effect for this style of paging (refer to EBox Unit Technical Description).

3.2.5.1 KI Paging Mode – When the EBox issues a request to read or write paged memory, it also asserts CON KI PAGING MODE. This allows the MBox to automatically refill the page table when required.

The page table must be refilled when a valid entry is not found in the directory. When the page table needs to be refilled, the pager asserts PAGE REFILL and the cache control will then execute a cache cycle (refer to Cache Page Refill cycle description) to fetch eight page table entries (4 words) from the process table (executive or user, depending on the EBox request qualifiers). If one or more of the needed words are in the cache, these words will be taken from the cache instead of core. In either case, the words are moved into the MBs. From the MBs the words are moved into the page table one at a time. During the Cache Page Refill cycle, PT ADR bits 24–26 are modified to move the words from the MBs into the correct page table locations. VMA 26 is blocked to select both halves of the page table (PT RIGHT and LEFT EN are asserted). VMA 24 and 25 are blocked and are replaced with two bit codes that correspond to the MB selected (MB SEL 1–2). PT ADR 18–23 remains unchanged for the duration of the Cache Page Refill cycle. The resulting PT address then changes only when another MB is selected to move another word into the page table. The page table is written at PAGE REFILL T12. At the same time the page table is written, the directory is also updated. The directory is updated by storing the virtual section address and the state of EBOX USER and validating the entry. One entry is placed in the directory for every two words that are written into the page table. When all the words have been written into the page table, the cache control retries the request.

If the directory contains a valid entry and the EBox requested a legal operation, the pager asserts PAGE OK. This signal informs the cache control to simply transform the virtual section and page address into the physical address by selecting the address from the page table. The directory contains a valid entry if the NOT VALID bit is cleared and the USER bit and the virtual section address in the table matches the section address and MCL VMA USER signal presented with the request. The reference is legal if page descriptor bits allow the request (refer to Subsection 3.2.2).

If the directory contains a valid entry and the EBox requested an illegal operation, the pager asserts PAGE FAIL. The page test logic of the pager senses that the EBox requested an illegal operation by checking the page descriptor bits of the referenced page. When the pager asserts PAGE FAIL, the cache control time state generator will assert PAGE FAIL HOLD and will advance through the PAGE FAIL time states. PAGE FAIL HOLD is asserted to inform the EBox that the page test failed. The PAGE FAIL time states are entered to transfer the page fail status word into the EBus register (LOAD EBUS REG) so that the EBox can read the word and evaluate the failure and take remedial action. The format of the page fail status word is shown in Figure 3-3.

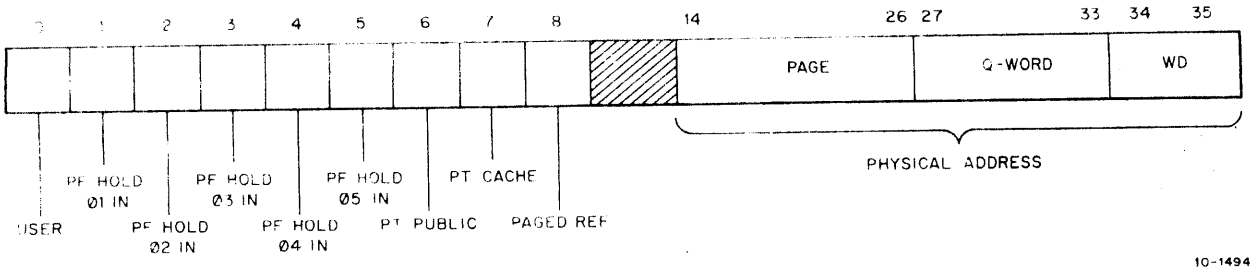


Figure 3-3 Page Fail Word Format

3.2.5.2 KL Paging Mode - When the EBox issues a request to read or write paged memory it issues the request with CON KI PAGING MODE negated. This prevents the MBox from executing the refill operation and forces the MBox to assert PF EBOX HANDLE and PAGE FAIL HOLD in the event a valid entry is not found in the page table. PF EBOX HANDLE will be asserted by the MBox only when the EBox specifies the KL paging mode is to be used. Therefore, the EBox knows that it must execute the refill operation when PF EBOX HANDLE is asserted by the MBox. If the MBox asserts PAGE FAIL HOLD but not PF EBOX HANDLE, this means that an illegal reference in accordance with the page descriptor bits of the page table entry was made by the EBox.

NOTE

The pager will never assert PAGE REFILL when the EBox specifies that the KL paging mode is to be used.

The page table must be refilled when a valid entry is not found in the directory or when an entry in the page table is not accessible (ACCESS bit is cleared). When the page table needs to be refilled, the pager asserts PAGE FAIL and PF EBOX HANDLE and the cache control will then assert PAGE FAIL HOLD and advance to the PAGE FAIL time states to transfer the Page Fail status word into the EBus register. The EBox recognizes that the page test failed because a valid entry was not found in the page table because of the fact that both PF EBOX HANDLE and PAGE FAIL HOLD was asserted by the MBox. The EBox will then issue a request to read the EBus register. The Page Fail status word will then be evaluated by the EBox to determine what kind of refill operation is required.

If a valid entry is not found in the Directory (PF code $2X_8$) the EBox will clear four entry locations in the page table. Each word in the page table contains two entry locations, therefore, the EBox must clear two page table words. Bits 25 and 26 of the PT address are set up and CLK PT WR is asserted by the EBox to select and clear the correct words. The EBox sets up the correct address by presenting a two-bit code to the MBox via the APR PT WR SEL0 and 1 control lines. The codes and their functions are defined in Table 3-2.

Table 3-2 Page Fault (PF) Code Truth Table

APR PT WR SEL 0 1	Functions
0 0	Select VMA address
1 0	Clear even PT word
0 1	Clear two directory entries
1 1	Clear odd PT word

After the EBox has cleared the even and odd words in the page table, the EBox will issue process table read requests and, if necessary, additional read requests to fetch a valid page table entry. When a valid page table entry is found, it is written (CLK PT WR) into the page table by the EBox. At the same time the page table entry is written, the virtual section address is also written into the directory by asserting CLK PT DIR WRITE. During this operation, the tables are addressed by virtual address bits 18–26, because the EBox will present a code of "00" on the APR PT WR SEL control lines. At this point, one of the four locations in the page table that corresponds to the validated entry in the directory will have an accessible entry so that the original request can be retried by the EBox.

If a valid entry is found in the directory, but an accessible entry is not found in the page table (PF code $0X_8$), the EBox will fetch the page table entry and write it into the page table, as previously described, without initially clearing two page table words.

Before the EBox writes a page table entry, it checks the W bit of the entry. If the EBox intends to read from this page and the W bit is set, it clears the W and sets the S bit before writing the entry into the page table. Consequently, a page fail condition will be sensed by the pager if the EBox issues a write request for that page. When this occurs, the EBox checks the PF code to see if the S bit is set. If the S bit is set, the EBox clears the S bit, sets the W bit, and writes the entry back into the page table. To indicate that the page will be written, the EBox also updates the Core Status Table (CST). After these operations are done, the EBox retries the original request. This scheme speeds up swapping programs out to mass storage, since only those pages that were written can be identified and swapped out.

If the MBox presents a page fail code other than those indicating that a refill operation is required, or the write test failed, the EBox will evaluate the failure and take appropriate remedial action.

The EBox can also clear the entire directory. This is done whenever another user program is started. To clear an entry in the directory, the EBox sets up VMA address bits 18–23, places code "01" on the APR PT WR SEL0 and 1 control lines, and asserts CLK PT DIR WR. The EBox must execute this operation 64 times to clear the entire directory.

3.3 CACHE AND CACHE CONTROL

Basically, the cache control (Figure 3-4) allocates core cycles to the EBox and the integral data channels by arbitrating EBox and channel requests and executing appropriate cache cycles. Cache cycles are executed to this end to see if the cache (Figure 3-5) contains any valid words. The channels need core cycles to fetch CCWs, read and write data, and store status. The EBox needs core cycles to read or write instructions and data, and to read or write locations in the process tables. If while executing a cache cycle it is found that a particular request is needed to satisfy a core cycle, a core cycle is then started. Core read cycles are needed to satisfy channel and EBox requests if the Cache does not contain the requested word(s). Core read cycles are also initiated to refill the page table when the EBox makes a paged reference and the page table does not contain a valid entry (KI mode). Core write cycles are always needed to satisfy a channel write request because the channels do not write into the cache. One of the reasons for this is that the data file coming from mass storage is not necessarily related to what the EBox is operating on at the time. Core write cycles are also required to satisfy EBox read or write requests if the LRU cache contains written words from another part of core. Cache cycles are also granted to the EBox to load and read internal MBox register, write-check a page of memory, map the virtual address, execute an SBus diagnostic cycle, or load the cache refill RAM. The page table can be written by the EBox directly (KL mode), therefore, a cache cycle is not needed.

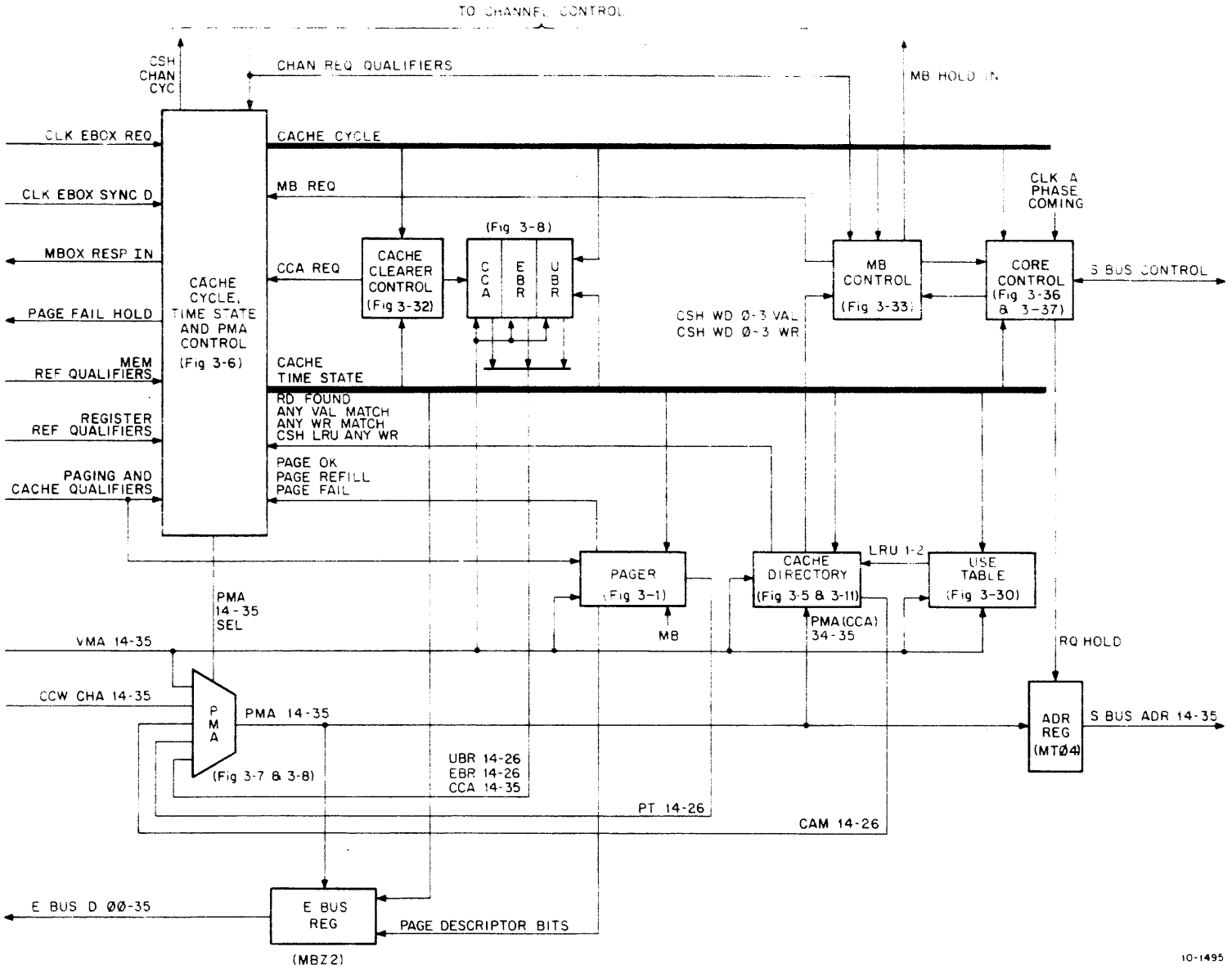


Figure 3-4 Cache Control Block Diagram

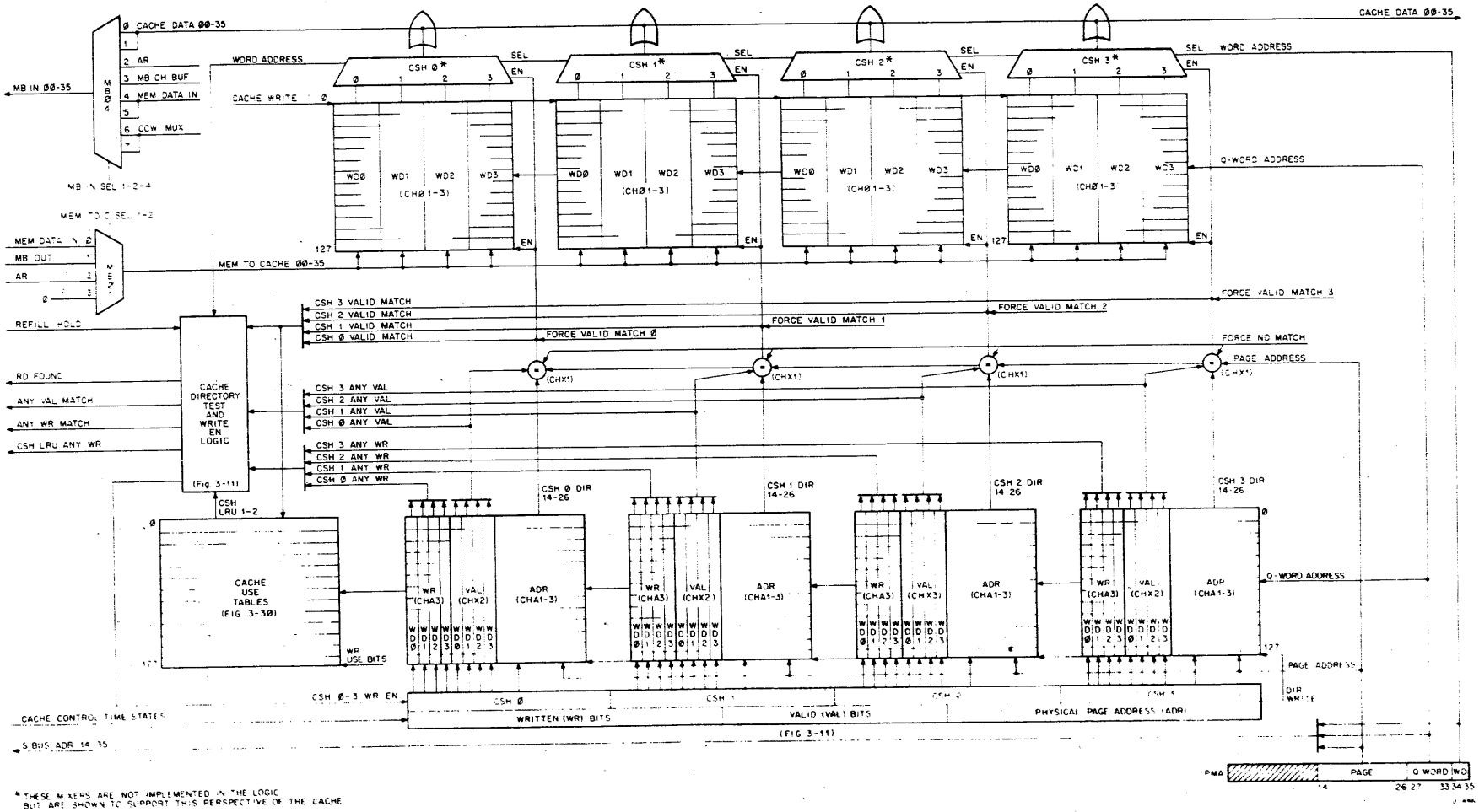


Figure 3-5 Cache Block Diagram

3.3.1 Cache Control Logic

The cache control consists of a priority request grant network to arbitrate and grant pending requests, a set of major and minor cycle latches, the address selector (PMA SEL), and a time state generator (Figure 3-6). To execute a particular cache cycle, the time state generator is started and then steered by the following variables:

a. Granted Request
b. Cycle Latch
c. Request Qualifiers from the Channel or EBox
d. Contents of the page table (EBox requests only)
e. Contents of the Cache Directory

3.3.1.1 Request Arbitration Logic – The priority request grant network arbitrates requests from the EBox, channel, MB control, and the CCA control. Requests from these components must be arbitrated and granted, one at a time, since each can issue a request independently of the other. However, the MB control and the CCA control will request cache cycles only after they have been started by a cache EBox cycle. Cache EBox cycles are granted only in response to EBox requests. The order in which the requests are serviced is as follows:

1. MB Request
2. CHAN Request
3. EBox Request
4. CCA Request

MB requests are issued by the MB control after a core read cycle to refill the cache is started by a cache EBox cycle. MB requests are issued by the MB control to request a cache cycle to move a word from the MB to the cache. Any words in the MBs must be moved out of the MBs before another core cycle can be started. Consequently, MB requests demand the highest priority for cache cycles.

Channel requests are issued by the channel control to request a cache cycle to move data in or out of core and invalidate the cache. During channel read operations, channel requests are issued by the channel control when the Channel Data buffer contains enough empty locations. During the cache cycle, any valid words in the cache are taken from the cache, instead of core, and a core read cycle is started to get the words that are not valid from core. However, during channel write operations, the channel control issues Channel request only when enough words have been accumulated in the Channel Data buffer. Any valid words in the cache are invalidated and a core write cycle is started when the cache cycle is executed during a channel write operation. Channel requests demand a higher priority for cache cycles than EBox requests so that the EBox can be prevented from getting a core cycle as long as Channel requests are pending. This feature minimizes data overruns.

EBox requests are issued by the EBox to reference memory, load or read internal registers (CCA control is started by loading the CCA register), write-check a paged location, map the virtual address, execute an SBus diagnostic cycle, or load the refill RAM. EBox requests demand a higher priority than CCA requests so that the EBox will not be locked out from the MBox while the cache control is validating core and/or invalidating the cache. This permits the EBox to get into the MBox and core memory to set up the next operation while the cache clearer is running.

CCA requests are issued by the cache clearer control to validate core and/or invalidate the cache. CCA Requests are granted last in the cache cycle priority scheme. It takes a maximum of 1024 cache cycles and up to 512 core cycles to clear the cache. Since it is more critical to permit the EBox to reference memory and internal registers while the cache clearer is running, EBox requests are granted cache cycles in preference to CCA requests.

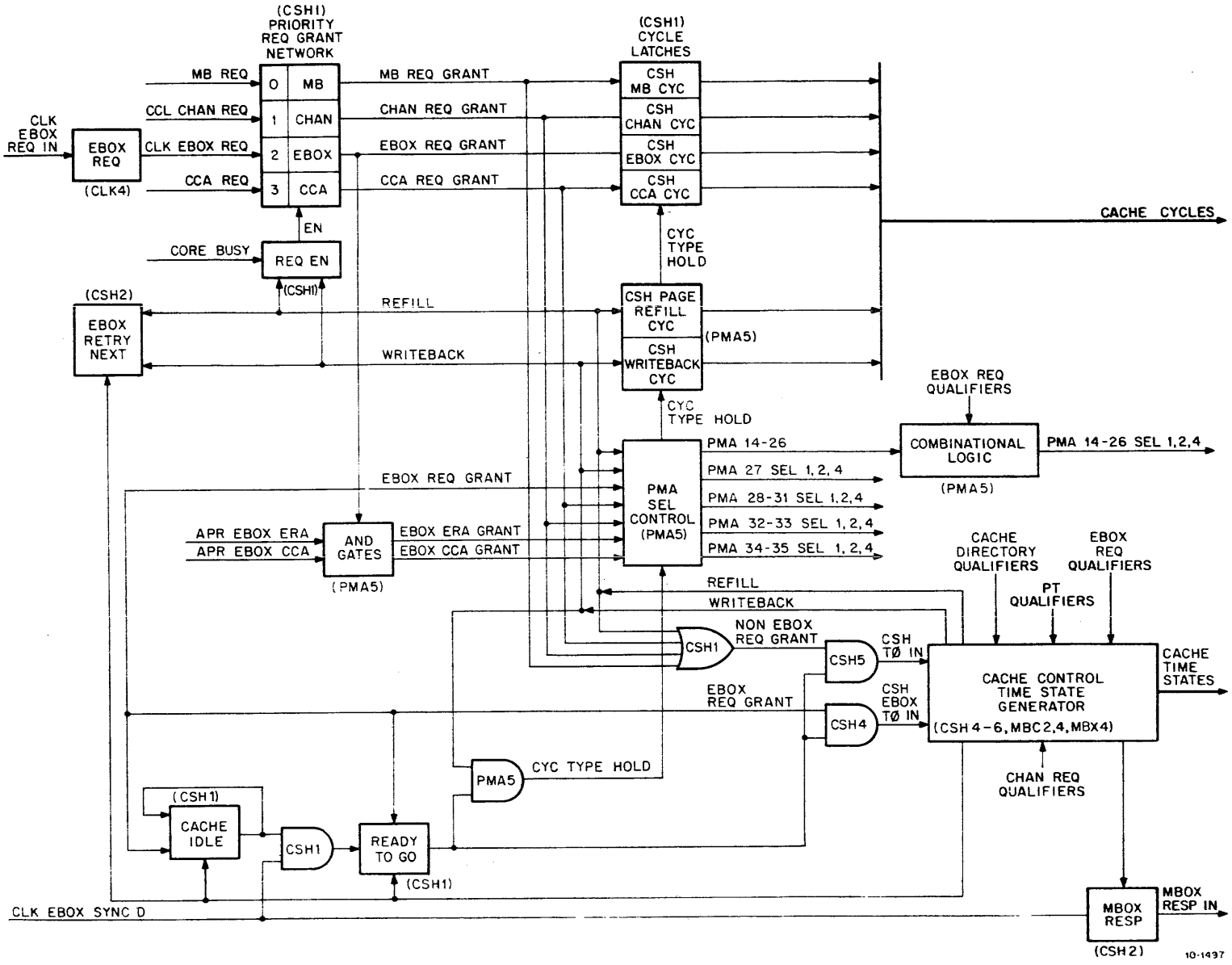


Figure 3-6 Cache Control Time State and PMA Control Block Diagram

A cache cycle can be started only after the cache control enters its idle state (CACHE IDLE). Initially, the cache control is forced to IDLE by MR RESET; thereafter the cache control returns to IDLE every time a cache cycle is done. From CACHE IDLE, the cache control advances to READY TO GO. If a request is pending, the next cache cycle is started at READY TO GO if the priority request grant network is not pre-empted. If neither a request is pending nor the priority request grant network is pre-empted, the cache control remains in its READY TO GO state until a request is received. If a request is received and granted, the cycle and PMA SEL latches are loaded and the time state generator is started. The cycle and PMA SEL latches will then be held for the duration of the cache cycle.

NOTE
These latches are loaded during READY TO GO or WRITEBACK T2.

In some cases, the priority request grant network is pre-empted to initiate another cache cycle (page refill or writeback). These cases are:

a. If during a cache EBox cycle it is found that the page table does not contain a valid entry for a paged memory reference (read or write) the priority request grant network is pre-empted (KI paging mode only) to prevent another pending request from being granted. Instead, a cache page refill cycle is initiated to refill the page table. After the cache page refill cycle is done, the cache control returns to IDLE and the EBox request is retried.

b. If during a cache EBox cycle, it is found that the cache does not have a record of the word and the LRU cache contains one or more written words from another page, the priority request grant network is pre-empted to prevent another pending request from being granted. Instead, a cache writeback cycle is initiated to write the written word back to core to free a cache block. After the cache writeback cycle is done, the cache control returns to IDLE and the EBox request is retried.

c. If during a cache clearer cycle to validate core, it is found that the cache block being looked at contains some written words, the priority request grant network is pre-empted to prevent another pending request from being granted. Instead, a cache writeback cycle is initiated to write the written words back to core to update (validate) the core copy. After the cache writeback cycle is done, the cache control returns to IDLE and READY TO GO, at which time the highest priority request pending is granted.

3.3.1.2 Request Execution Logic – The cache control time state generator is steered by a number of variables depending on the particular cache cycle that is being executed. A summary of the variables that control the operation of the state generator during specific cache cycles is presented in Table 3-3.

The state generator is steered to effect specific operations while a cache cycle is being executed. Some of these operations are: update cache directory, update use table, move valid or written words from the cache into the MBs, start core cycle, move words from the MBs into the page table and start the cache clearer control.

Besides setting a cycle latch, the PMA SEL latches are set up every time a new cache cycle is started, as described previously. The PMA SEL latches control the address mix out of the PMA. The PMA supplies the address for the cache and if a core cycle is required for the SBus, the cycle latches steer the state generator to execute a specific cache cycle. Each type of cache cycle performs a different function, based on the variables that steer the time state generator. A summary of the assigned functions of each type of cache cycle is presented in Table 3-4.

Table 3-3 Time State Generator Control Variables

CACHE CYCLE	VARIABLES			
	EBOX REQ QUALIFIERS	CHAN REQ QUALIFIERS	PAGE TABLE CONTENTS	CACHE DIRECTORY CONTENTS
MB	CLK EBOX SYNC D	-	-	-
CHAN	-	X	-	X
EBOX	X		X	X
CCA	X[1]			X
REFILL (KI ONLY)	X			X
WRITEBACK				X

[1] These qualifiers are stored in the 3-bit CCA Control Register of the MBox

Table 3-4 Cache Cycle Functions

Cycle	Function
MB	Move words from MBs to cache. The words are moved into the MBs by the core control during a core read cycle.
CHAN	READ: Move the valid words from the cache to the MBs and start a core read cycle to fetch the words that are not valid. If all the requested words are in the cache, a core read cycle is not initiated. WRITE: Start a core write cycle; if any valid words are in the cache, clear their valid and written bits.
EBOX	READ: Page check the read request (paged only) by comparing the EBox request qualifiers with the contents of the page table and checks the cache to see if the word is there. If the reference to the page is OK and the word is in the cache it is simply presented to the EBox. If the page check is not OK, either a page refill cycle is initiated or the EBox is informed that a page fail condition exists. If the desired word is not in the cache but some of the words in the quadword group are in the cache, a core read cycle is initiated by the cache cycle to fetch the words that are not valid in the quadword group. The word the EBox requested will come in first and will be presented to the Box and be written into the cache. If none of the words in the quadword group are in the cache, the the LRU cache is used for the refill operation if there are no written words from another page in that cache. If written words are found in the LRU cache, they are written back to core before the core read cycle is started.

Table 3-4 Cache Cycle Functions (Cont)

Cycle	Function
EBox (Cont)	WRITE: The page table and the cache directory are checked as described for the Read request. Words are always written into the cache unless the cache is to be bypassed. If the cache has a record of the quadword (ANY VALID MATCH) the word is simply written into the cache quarter that has the record. If the cache has no record of the quadword, the word is written into the LRU cache if there are no written words from another page in that cache. If written words are found in the LRU cache, they are written back to core before the LRU cache is written. NOTE **Cache cycles are also used to load and read internal registers, to check a page, and to map the virtual address.**
CCA	Invalidate the cache and/or validate core for a single page or the entire cache. To validate core, writeback cycles are initiated for all words that are written.
REFILL	Move any valid words from the cache to the MBs and start a core read cycle to fetch the words that are not valid. Then move the words into the page table. If all the words are in the cache, a core read cycle is not initiated.
WRITEBACK	Move all written words into the MBs, clear their written bits, and start a core write cycle.

3.3.1.3 Page Table and Cache Address Logic – The cache cycles that check the cache directory and/or the page table must allow for logic transit time and RAM address to output access time. Approximately 120 ns (three MBox clock ticks at 25 MHz) are required from the time the address is presented to the page table and the cache before their contents can be checked to decide the next step in the cache cycle. The page table is addressed by the VMA when the EBox issues the request. However, the cache address varies with the cycle to be executed and is presented to the cache when the cycle is started (Figures 3-7 and 3-8). A summary of the sources that contribute to forming the cache address is presented in Table 3-5.

The cache is addressed by the nine least significant bits of the 22-bit physical address or the 23-bit virtual address. These bits point to a word within a page and are not subject to modification by the paging mechanism in the system; only entire pages can be relocated through the paging mechanism. All nine address bits are used to address the data portion of the cache, while only the seven high-order bits address the directory portion of the cache. This has the effect of addressing one data word in each cache and a directory entry for each cache. Consequently, if one of the directory entries matches the page address that was presented with the request and the valid bit for the word in the associated cache is set, then the desired word is in the cache. Note that only one word is addressed in each cache while the cache directory, in conjunction with the valid bit of the word, specifies which cache has the requested word. To further edify the addressing scheme, consider that all four words associated with a given directory entry are in the cache. This means that for all four combinations of the two least significant address bits (bits 34–35), which address only the data portion of the cache, a word would be found, since each word would be associated with its own valid bit and the same cache directory.

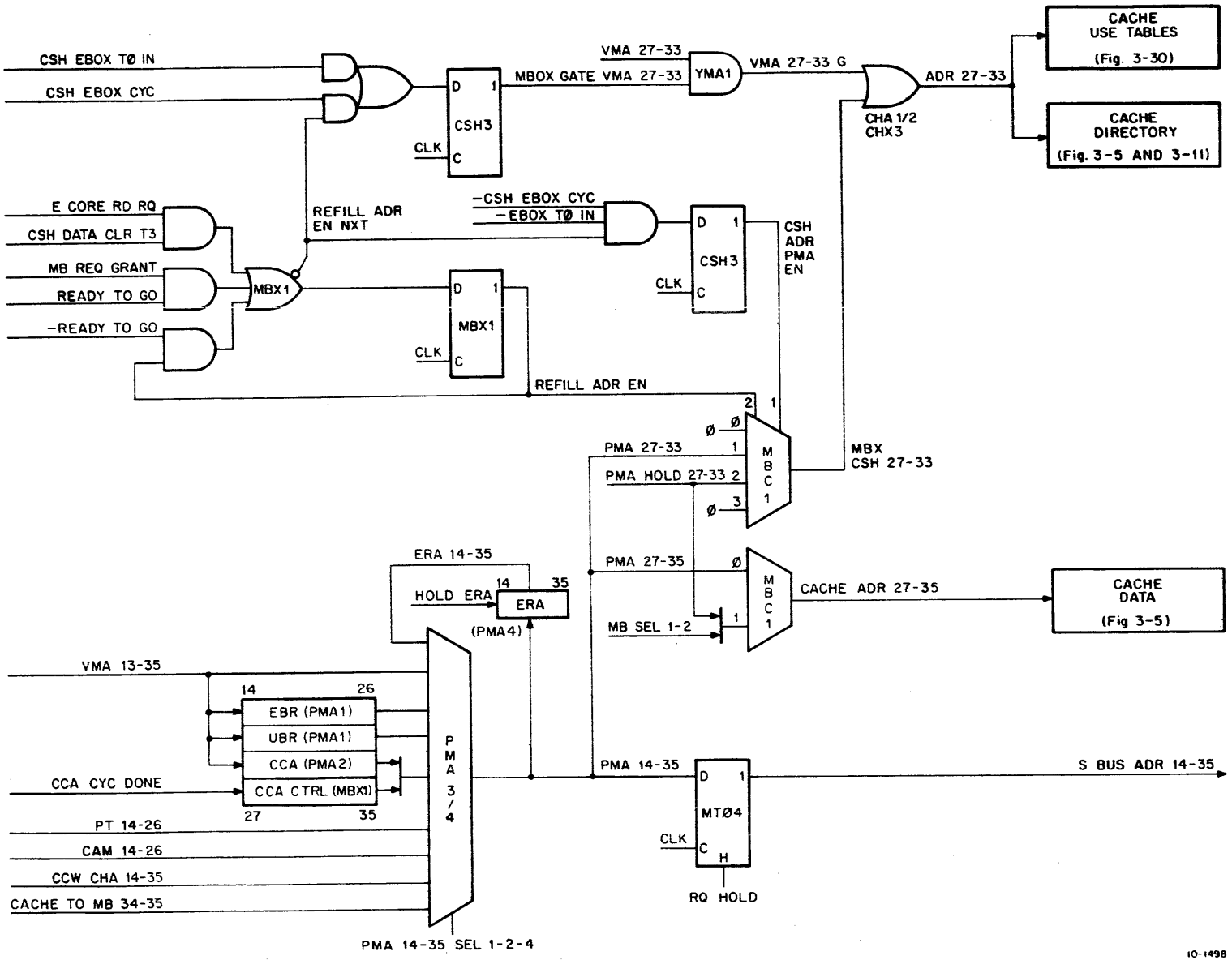
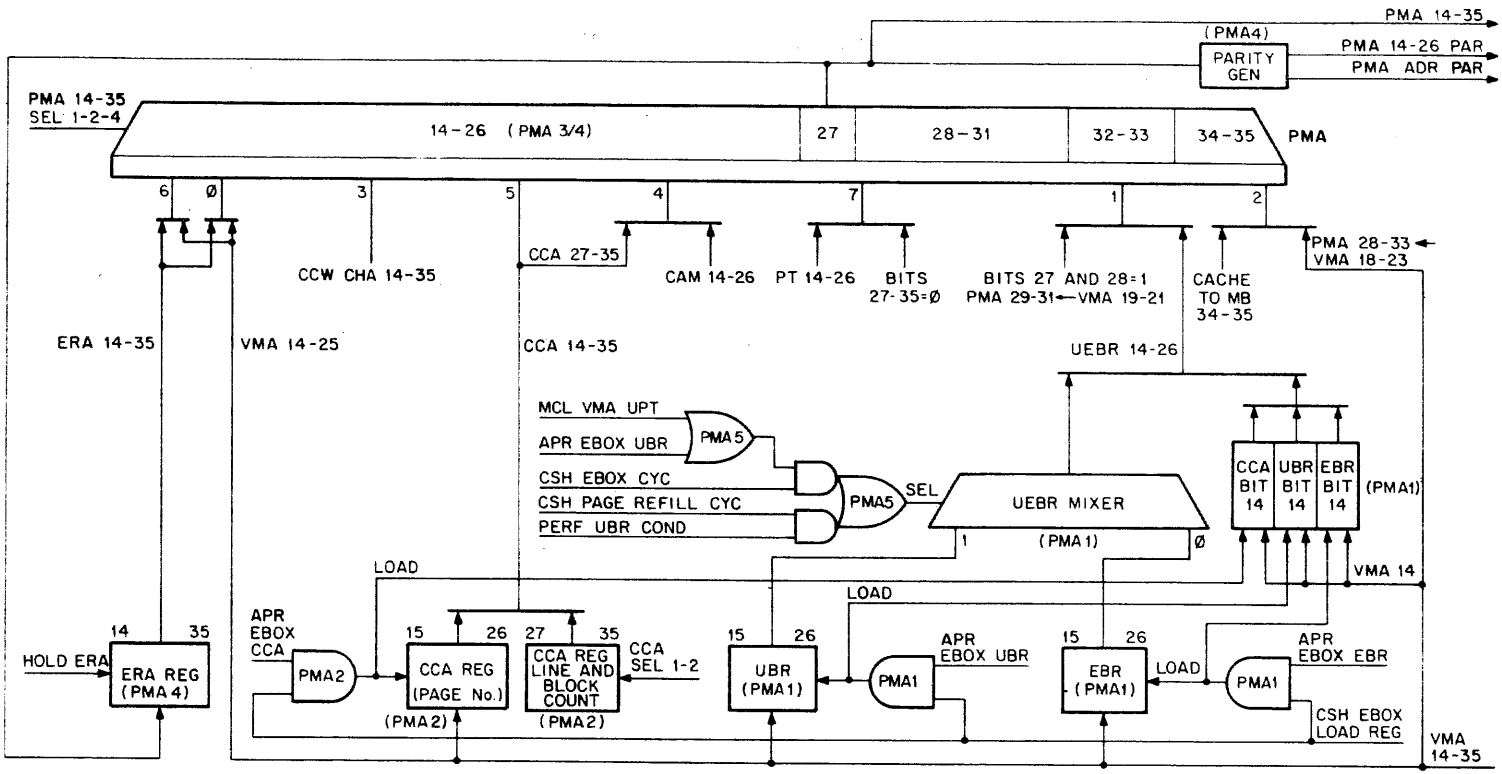


Figure 3-7 Cache Address Simplified Logic Diagram



OCTAL CODE FOR MIXER SEL	PMA SEL 14-26	PMA SEL 27	PMA SEL 28-31	PMA SEL 32-33	PMA SEL 34-35
Ø	VMA 14—26	ERA 27	ERA 28—32	ERA 32—33	ERA 34—35
1	UEBR 14—26	"1"	VA "1" 19—21		
2			VA 18—21	VA 22—23	PMA X, Y
3	CHA 14—26	CHA 27	CHA 28—31	CHA 32—33	CHA 34—35
4	CAM 14—26	CCA 27	CCA 28—31	CCA 32—33	CCA 34—35
5	CCA 14—26	CCA 27	CCA 28—31	CCA 32—33	CCA 34—35
6	ERA 14—26	VMA 27	VMA 28—31	VMA 32—33	VMA 34—35
7	PT 14—26	"Ø"	"ØØØØ"	"ØØ"	"ØØ"

Figure 3-8 PMA Mixer Simplified Logic Diagram

Table 3-5 Cache Address Combinations

Cycle	Address Source	
	Cache Directory (27-33)	Cache Data (27-35)
MB	PMA HOLD	PMA HOLD + MB SEL 1-2
CHAN	PMA ← CHA	PMA ← CHA + CTOMB
EBOX	VMA	PMA ← VMA
CCA	PMA ← CCA	PMA ← CCA
REFILL	PMA ← QUADWORD POINTER	PMA ← QUADWORD WD POINTER + CTOMB
WRITEBACK		
CCA REQ	PMA ← CCA	PMA ← CCA
EBOX REQ	PMA ← VMA	PMA ← VMA + CTOMB

During the cache MB cycle, the cache address is provided by the PMA HOLD register and the MB control. The seven high-order bits of the nine-bit cache address are supplied by the PMA HOLD register; the two low-order bits are a function of which MB is selected (MB SEL 1-2) at the time. The PMA HOLD register is loaded when a core read cycle is started and is held for the duration of the cycle (CORE RD IN PROG). The MB control provides the two low-order bits of the cache address (MB SEL 1-2) to move the word into the correct location of the data portion of the cache. The contents of the PMA HOLD register and the MB SEL 1-2 control lines are selected (REFILL ADR EN) to address the cache every time a cache MB cycle is executed.

During the cache channel cycle, the cache address is provided by the channel which is selected by the PMA control when the cycle is started. The two least significant bits of the cache address (Cache To MB 34-35) are needed only during the channel read operation if some of the requested words are valid in the cache. These address bits are a function of which words the channel requested and are used to move the valid word into the MBs (refer to MB Control Description). During a channel write operation, the two least significant cache address bits are not needed because data is not moved in or out of the cache.

During the cache EBox cycle, the cache address is provided by the VMA. The seven high-order bits of the cache address are not passed through the PMA to minimize the transit time, thereby permitting the cache control to check the contents of the cache directory somewhat earlier than would otherwise be possible. Consequently, this speeds up the cache EBox cycle.

During the cache CCA cycle, the cache address is provided by the cache clearer control, which is selected by the PMA control when the cycle is started.

During the cache page refill cycle, the cache address is supplied by the PMA. The seven high-order bits constitute a quadword pointer into the process table (EPT or UPT) and the two low-order bits are a function of which words in the cache are not valid. If some of the words in the cache are valid, the low-order two address bits are used to move the valid words into the MBs (refer to MB Control Description) otherwise; these address bits are not needed.

During a cache writeback cycle, the cache address is provided by either the CCA or the VMA, depending on which request was granted. If an EBox request was granted, the cache writeback cycle is entered from a cache EBox cycle; the PMA control, therefore, selects the VMA to address the cache. If a CCA Request was granted, the cache writeback cycle is entered from a cache clearer cycle; the PMA control, therefore, selects the CCA register to address the cache. As described before, the two low-order bits of the cache address are used to move the words of interest (which are the written words in this case) from the cache to the MBs.

3.3.1.4 Cycle Decision Logic Three MBox clock ticks after a cache EBox cycle is started, the contents of the page table and the contents of the cache directory are checked (Figure 3-9). For cache CHAN and CCA cycles, the contents of the cache directory are checked at CSH T3 (Figure 3-10). One more clock tick is needed for these cycles to compensate for the additional transit that is contributed by the PMA. The page table and the cache directory supply the following variables which, in conjunction with the request qualifiers, are used to steer the state generator and control the subsequent operation of the cache control.

a. Page Table

   1. PAGE OK
   2. PAGE REFILL
   3. PAGE OK

b. Cache Directory

   1. ANY VALID MATCH
   2. RD FOUND
   3. ANY WRITTEN MATCH
   4. LRU ANY WRITTEN
   5. WD 0–3 VAL
   6. WD 0–3 WR

Besides the variables specified, the Cache Directory Cache Address Mixer (CAM) also presents the physical page address (CAM 14–26) to the PMA. This address is needed to write the written words in the cache back to core.

Figure 3-11 illustrates the control logic for testing and writing the cache directory. Only one page of the cache directory is shown to simplify this presentation.

NOTE

The cache contains four pages of storage; a given line of the cache will never contain more than four words from the same page and all four words will always be in the same cache quarter. Consequently, only one page (quarter) of the cache directory will contain a valid entry.

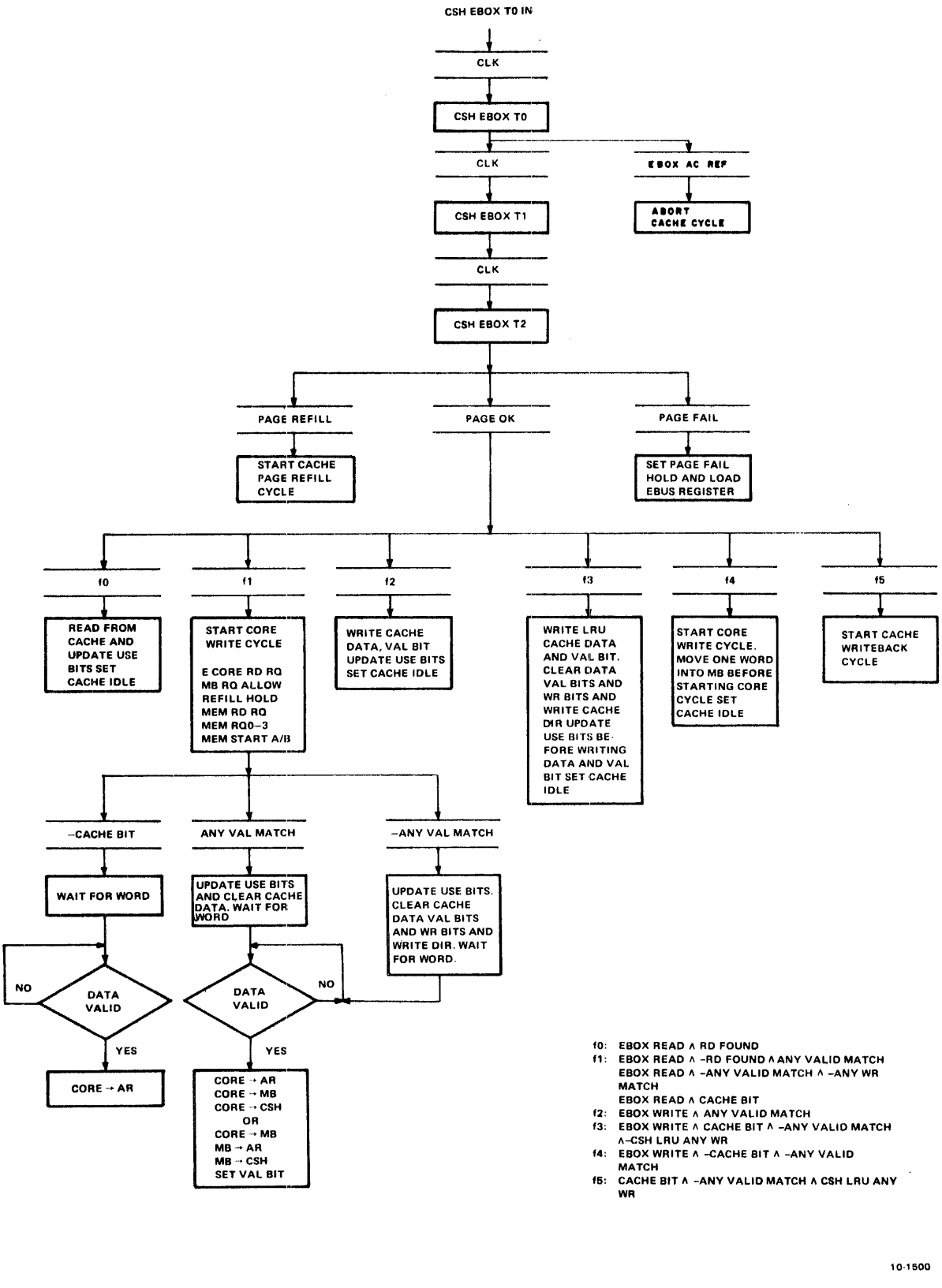


Figure 3-9 Cache EBox Cycle Decisions Flow Diagram For Read and Write Requests

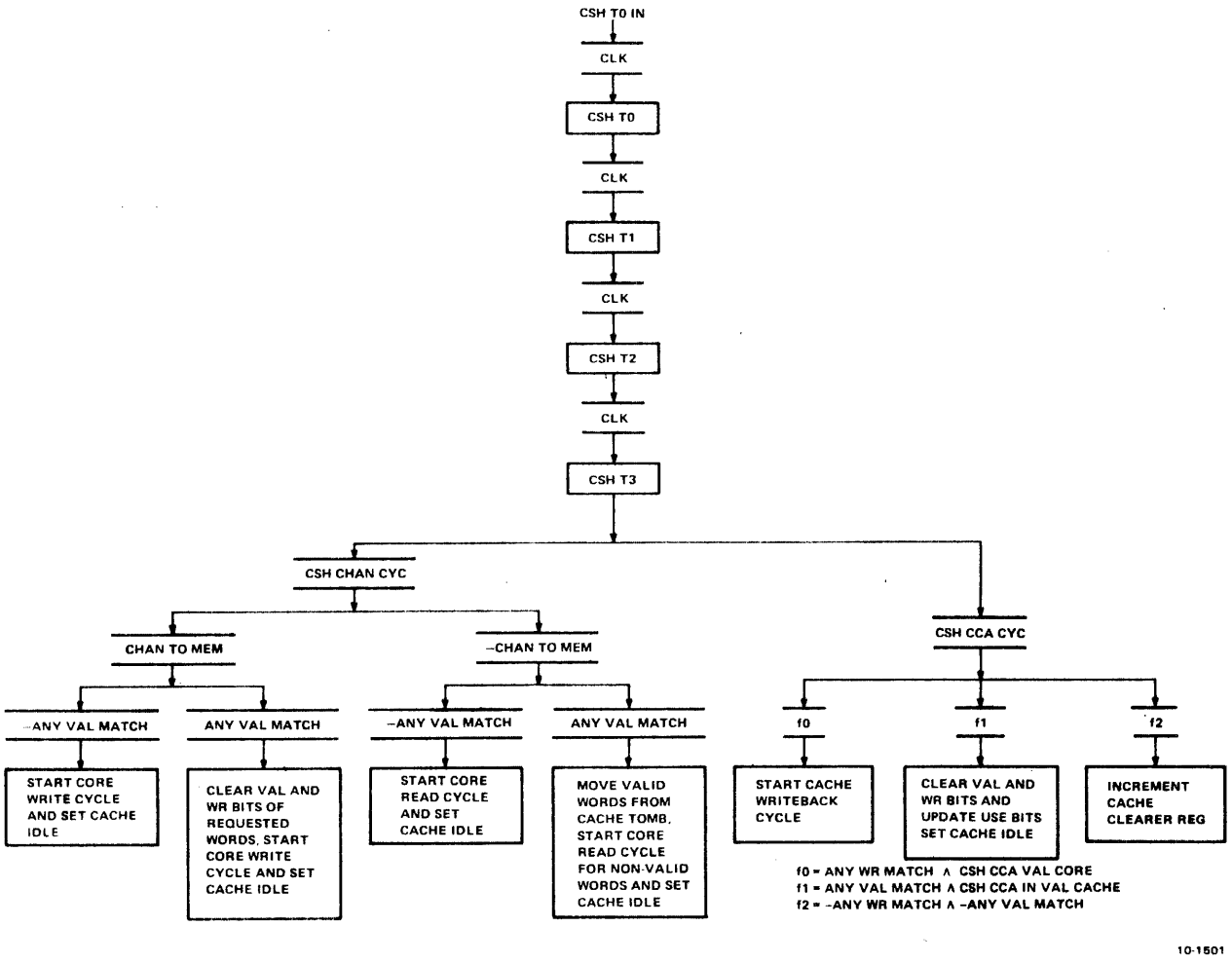


Figure 3-10 Cache Channel and CCA Cycle Decisions Flow Diagram

The control logic for the cache directory includes steering logic for selecting the variables that will be tested during the cache cycle and for enabling the write logic for updating the directory. When the cache is addressed, four data words and four directory entries, comprising one line in the cache are selected. Each directory entry consists of a page address and the VALID and WRITTEN bits for a quadword. If the cache contains the requested word, or some of the words in the addressed quadword group, then the page address presented with the request will match the contents of the directory and one or more of the valid bits in the quadword group will be set resulting in a CSH n VALID MATCH condition. The number n corresponds to the page number (or CSH No.) that contained the address that matched. An Exclusive-OR equality (=) comparator is used to compare the addresses. The result of this test is ANDed with the OR function of all the VALID bits of the addressed quadword group for which the address matched to produce the CSH n VALID MATCH. The condition ANY VALID MATCH is simply an OR function of the outputs from all four equality comparators. One comparator is used for each page of the cache. RD FOUND is true when a VALID bit of a word in the addressed cache line (one word in each cache quarter) is set and the address in the corresponding directory matches.

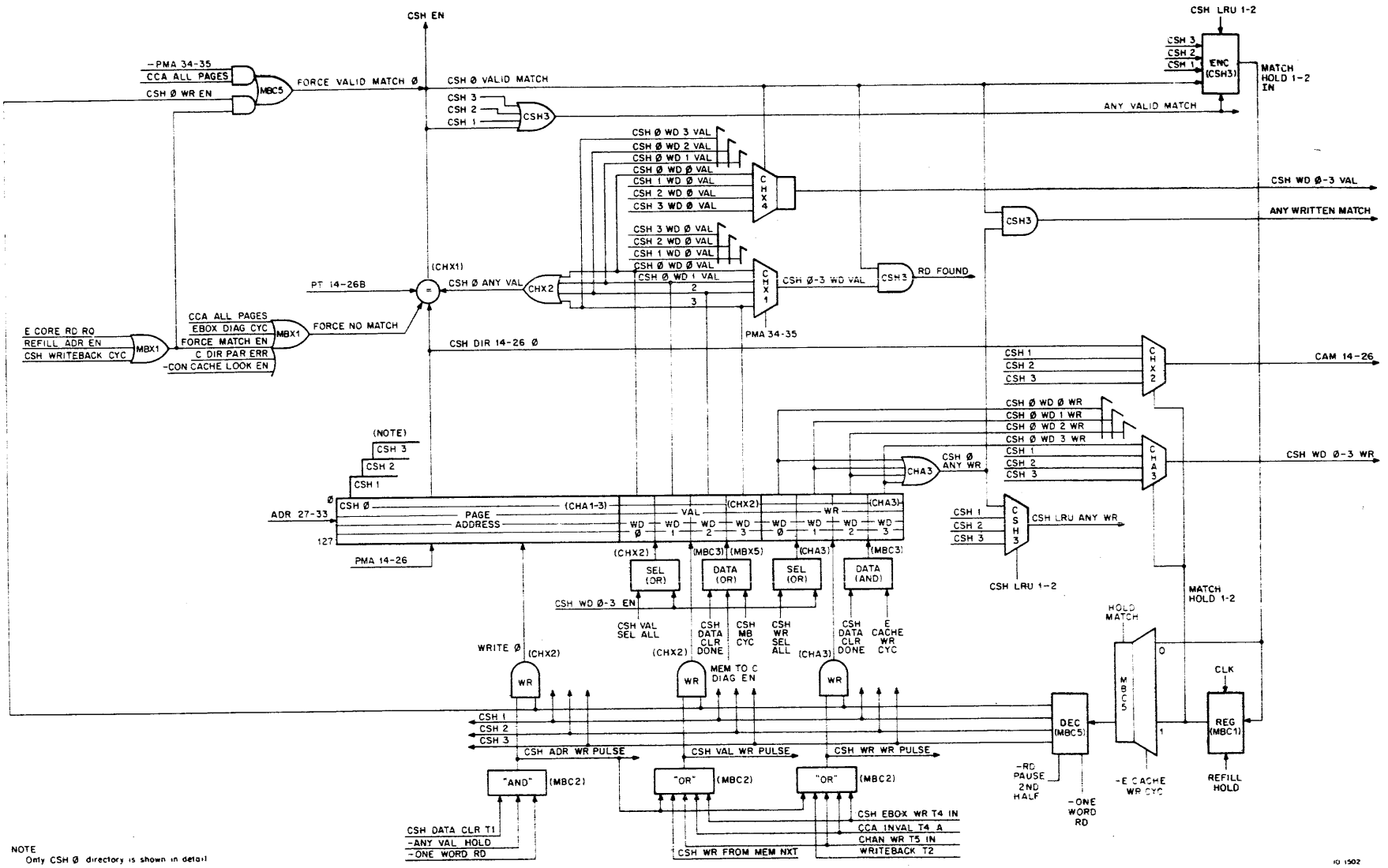


Figure 3-11 Cache Directory Test and Control, Simplified Logic Diagram

The condition ANY WRITTEN MATCH is the OR function of all the WRITTEN bits of the addressed quadword in the page of the cache that yielded a CSH n VALID MATCH. As is the case for the VALID bits, all four WRITTEN bits corresponding to the quadwords that are being addressed in the cache are selected at the same time. Consequently, a set WRITTEN bit for any word within a quadword of any cache block can cause the condition ANY WRITTEN MATCH. This condition is tested to validate core (refer to cache clearer control description). The condition CSH LRU ANY WR indicates that one or more words in the LRU block of the cache are written and may need to be written back to core. If the cache directory does not contain an address that matches the address presented with the request, then the words belong to another page and must be written back to core before the LRU cache block can be used in refilling the cache. These words will also be written back to core when a cache clearer cycle to validate core is executed.

The signals CSH WD 0–3 VAL and CSH WD 0–3 WR specify which words of the addressed quadword are valid and which words are written. These signals are used to set up the MB 0–3 WR RQ queue and the CTOMB WD 0–3 RQ queue of the MB control. These queues are set up when valid or written words are to be moved from the cache into the MBs. Valid words are moved from the cache to the MBs:

a. During a cache page refill cycle so that the valid words can be moved from the MBs into the page table.

b. During a cache channel cycle that is executing a channel read request so that the valid words can be taken by the channel control.

Written words are moved from the cache to the MBs:

a. During a cache writeback cycle that was initiated by a cache EBox cycle to make room in the cache to permit a cache refill operation to be done.

b. During a cache writeback cycle that was initiated by a cache CCA cycle to validate core.

The complement of CSH WD 0–3 VAL is used to set up the MEM RQ 0–3 lines and in some cases SBus address bits 34–35 to initiate a core read cycle (refer to Core Control Description).

To update the cache directory the correct cache directory must be selected.

NOTE

The cache address selects one data word in each cache (a line) and the four corresponding directory entries, as described before.

The correct cache directory is selected by the CSH n WR EN signal, which is a function of either the cache directory that produced the matched entry (CSH n VALID MATCH) or the USE bits (LRU 1–2 bits) of the use table if no match occurred. This selection is automatically made by the encoder that produces MATCH HOLD 1-2 IN. These signals are presented to a decoder via a holding register to produce a WR EN signal that corresponds to the applied code (0, 1, 2, or 3) which specifies the cache to be used. The holding register allows the code to be held for the duration of a core read cycle, the writeback cycle, and for the case where the valid bits are to be cleared. Besides enabling the cache directory write logic, the CSH n WR EN signals are also used to force a valid match (FORCE VALID MATCH n) to select the correct cache for writing written words in the LRU cache back to core (writeback cycle) and for refilling the cache (core read and MB cycles).

When a valid match is forced, as described above, the equality comparators (=) are disabled to avoid potential conflicts. A valid match is also forced when clearing the cache of all written words from all pages (PMA 34–35 ∧ CCA ALL PAGES). This function permits the cache control to look into each cache directory to see if any written words are in the cache.

3.3.1.5 Cache Control Time States – Bar charts are presented in the following subsections to illustrate how the state generator continues from CSH EBOX T2 and CSH T3 to execute the request. All subsequent branch conditions are shown. A summary of the significant time states and their functions are given in Table 3-6.

Table 3-6 Cache Control Time State Summary

Time States	Assigned Function
CSH EBOX T0–T3 (CSH4)	Besides serving as a delay to compensate for transit time associated with testing the contents of the Page Table and the Cable Directory for the EBox Read and Write Requests, these time states are also used to execute the following EBox Requests. a. Abort the Cache cycle in the event the EBox references the ACs. b. Load CCA, EBR or UBR Registers. c. Read the contents of the Page Table (MAP). d. Read CCA, EBR, UBR, EBUS or ERA register. e. Write check a page (PAUSE WRITE). f. Load Refill RAM. In addition, CSH EBOX T2 initiates a core read cycle if a Cache Refill is required and updates the Use Table if the Cache contains some valid words.
PAGE FAIL T2–T3 (CSH4)	Load Page Fail code and Physical Address into EBus register and send PAGE FAIL HOLD to EBox if a PAGE FAIL condition from the Page Table is sensed in response to an EBox Read or Write Request.
CSH EBOX WR T3–T4 (CSH4)	Write Data into Cache and set VALID and WRITTEN bits in Cache Directory in response to an EBox Write Request.
CSH DATA CLR T1–T3 (MBC2)	These time states initiate operations to satisfy both Read and Write request from the EBox. EBox Write: Before the data is written into the LRU Cache the CSH DATA CLR time states cause the Cache Directory to be updated as follows: The new page address is written and all the VALID and WRITTEN bits are cleared in the Cache Directory. EBox Read: When a core read cycle to refill the Cache is initiated (by CSH EBOX T2) the CSH DATA CLR time states are also entered to update the Cache Directory by writing the new page address and clearing all the VALID and WRITTEN bits.

Table 3-6 Cache Control Time State Summary (Cont)

Time States	Assigned Function
CLR WR T0 (CSH4)	Checks that the EBox is not trying to write into a Cache block for which a core read cycle has been started and has not been finished. This test is also made during CSH EBOX WR T3. If this test were not made, then one or more words in the Cache block could be over written when the word(s) comes in from core. The test is made by checking that the contents of the PMA HOLD register (bits 27–33) which holds the Cache refill address is not the same as the corresponding address bits presented by the EBox with the request.
ONE WORD WR T0 (CSH4)	Enables the MB control to move a word from the AR into the MB pointed to be PMA 34 and 35. This time state is entered only when the Cache bit is cleared when the EBox requests to write or requests an SBUS DIAG Cycle.
CACHE TO MB T1–T4 (MBX4)	These time states control the MB control and move valid or written words from the Cache to the MB during a Cache Page Refill Cycle, a Cache Writeback Cycle or during a Cache Channel Cycle that is executing a Read operation. CACHE TO MB T2 is held until A or B PHASE IS COMING is asserted to synchronize the state generator with the SBus clock so that a core cycle can be started at a later time state without delay and in synchronism with the SBus clock.
CORE DATA VALID (-1) (MBC4)	These time states serve as a two-MBox-clock-tick-delay to allow the data placed on the SBus by core memory during a core read cycle to stabilize before moving it into the MB.
CSH WR DATA RDY (CSH6)	During a core read cycle, writes the first word that comes in from core into the Cache and sets the VALID bit in the directory. Subsequent words coming in from core cause MB Requests to be issued.
DATA DLY 1–2 (CSH6)	Checks parity (in the MB) of the first word that comes in from core during a core read cycle.
CSH T0–CSH T3 (CSH5)	Besides serving as a delay to compensate for transit time associated with testing the contents of the Cache directory during Cache Page Refill, Cache CCA and Cache CHAN cycles, CSH T0–T2 are also used to execute Cache MB Cycles to move words from the MBs to the Cache and set the valid bit in the directory after they come in from core during a core read cycle.

Table 3-6 Cache Control Time State Summary (Cont)

Time States	Assigned Function
PAGE REFILL T4, T8–T13 (CSH5)	PAGE REFILL T4 initiates the Cache page refill cycle by pre-empting other pending requests. This is done by setting the Cache page refill cycle latch and setting up a new address without going through the priority request logic. PAGE REFILL T8 sets up the MB control to move any valid words from the Cache to the MBs so that they can be transferred to the Page Table. PAGE REFILL T8 sets up the core read cycle for those words the Cache has no valid entries. If all four words in the Cache are valid a core read cycle is not started. PAGE REFILL T10–T13 moves the words from the MBs to the Page Table after all valid words have been moved from the Cache to the MBs and a core read cycle for the remaining words has been started. After all the valid words from the Cache are moved to the Page Table PAGE REFILL T10 is held until another word is received from core at which time that word is also moved to the Page Table. This is repeated until all four words have been moved to the Page Table.
WRITEBACK T1 (CSH4) and T2 (MBX4)	These time states initiate the Cache writeback cycle by pre-empting other pending requests. This is accomplished by setting the Cache writeback cycle latch and setting up a new address without going through the priority request grant logic. The WRITEBACK time states also clear the written bits in the directory and set up the MB control to move the written words from Cache to the MBs so that they can be transferred to core.
CCA INVAL T4 (CSH6)	Clears the valid and written bits in the Cache Directory and updates the Use Table during a Cache CCA cycle.
CCA CYC DONE (CSH6)	Decrements Cache Clearer address counter (bits 27–35) and clears CCA REQ when the counter overflows.
CHAN RD T5 (CSH5)	Starts the Core Read Cycle for those words the Cache has no valid entries. If all words requested by the channel control are valid a core read cycle is not started. CHAN RD T5 is held for one additional clock tick if A or B PHASE COMING is not asserted when the state generator reaches CHAN RD T5 to synchronize the start of the core cycle with the SBus clock.

Table 3-6 Cache Control Time State Summary (Cont)

Time States	Assigned Function
CHAN T4 (CSH5)	This time state initiates operations to satisfy both Read and Write Requests from the channels. Chan Read: Sets up the MB Control to move any valid words from the Cache to the MBs so that they can be transferred to the channel data buffer. Chan Write: Sets up logic to clear the valid and written bits in the Cache Directory of those words specified by the channel control.
CHAN WR T5 (CSH5)	Clears the valid and written bits in the Cache Directory of those words the channel control requested to write to core. **NOTE** **The core write cycle is not started by the Cache control but by the channel control when a Cache channel cycle is started.**

3.3.2 Cache EBox Cycle

Besides granting and executing channel requests, the cache control grants and executes EBox requests. The EBox request is granted if a higher priority request (MB or CHAN REQ) is not pending. To execute the request, the cache control enters the cache EBox cycle (Figures 2-6 and 3-12). In many cases, the request is executed without the cache control having to leave the cache EBox cycle. However, if a page table entry has to be fetched (KI paging), or written words have to be written back to core, the cache control enters the appropriate subcycle to execute these operations and then retries the original request by executing the cache EBox cycle again. Page refills, writebacks, and core reads require a core cycle. Therefore, if core is busy, these subcycles cannot be executed. In this case, the cache control simply retries the request until core is released. The following requests can be issued by the EBox:

a. Load Register
b. Read Register
c. Map (Read PT)
d. Read Memory
e. Write Memory
f. Read Pause Write
g. Write Check
h. Write Refill RAM
i. SBus Diag

NOTE

If the cache is not implemented, EBox requests to read/write memory are serviced by transferring a single word (one word read/write) from/to memory (Subsections 3.3.2.4 and 3.3.2.5).

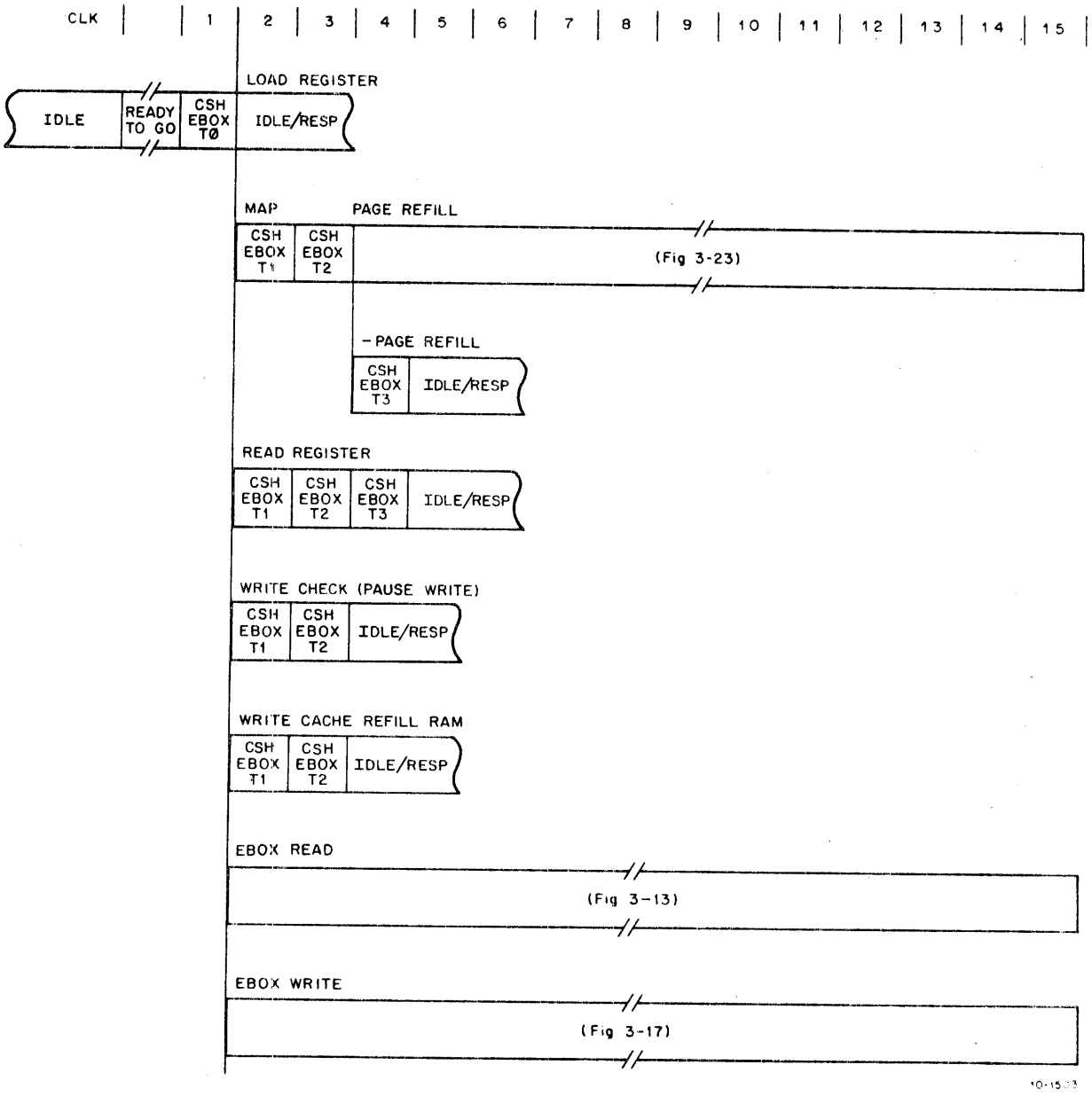


Figure 3-12 Cache EBox Cycle, Time State Bar Chart

3.3.2.1 EBox Load Register – The MBox contains three operational registers that can be loaded by the EBox:

EBR – Executive Base Register
UBR – User Base Register
CCA – Cache Clearer Address Register

These registers are loaded using CONO PAG, DATAO PAG, and the Sweep instructions, respectively.

To load these registers, the EBox, in response to the instruction, raises CLK EBOX REQ, APR EBOX LOAD REG and APR EBOX EBR, APR EBOX UBR or APR EBOX CCA, depending on which register is to be loaded. The CCA register needs to be loaded with the physical address only when one page is to be swept from the cache. If the CCA register is to be loaded, the EBox will specify what kind of cache sweep is to be performed by setting up the following control signals correctly: CSH CCA INVAL CSH, CSH CCA VAL CORE, and CSH CCA ONE PAGE. These control signals are set up by IR AC10–12 from the EBox. In addition, the data to be loaded into the register must be in the VMA (bits 14–26).

This sets up the conditions required for the MBox to service the EBox request to load a register. If the cache control is IDLE, or when the cache control enters its IDLE state and no higher priority requests are pending, the cache control will grant the EBox request and start a cache EBox cycle to load the register. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. The cache control time state generator then clears the CLK EBOX REQ latch and advances directly to the CSH EBOX time state branch. At CSH EBOX T1, the desired register is loaded and the cache control asserts MBOX RESP IN and returns to IDLE. If the CCA register was loaded, the cache control also asserts the CCA request to inform the EBox that the cache clearer cycle was started and that the EBox should not make another request until the cache is cleared. CCA REQ is cleared when the operation is done. Asserting CCA REQ also causes the cache control to grant a cache clearer cycle when no higher priority requests are pending. CCA REQ remains set until the entire sweep operation is done.

3.3.2.2 EBox Read Register – The MBox contains three operational registers that can be read by the EBox:

EBR
UBR
ERA

The EBR, UBR, and ERA are read using the CONI PAG, DATAI PAG, and BLKI PI instructions, respectively. To read these registers, the EBox raises CLK EBOX REQ, APR EBOX READ REG, and APR EBOX EBR, APR EBOX UBR or APR EBOX ERA, depending on which register is to be read. The EBox also sets up the appropriate diagnostic function and code (DIAG READ FUNCT 167_8) in response to the instruction to connect the output of the EBus register to the AR. This sets up the conditions required for the MBox to service the EBox request to read a register.

If the cache control is IDLE, or when the cache control enters its IDLE state and the priority request grant logic is not pre-empted by a subcycle request (page refill or writeback) that may be required in satisfying the previous EBox request, and if no higher priority requests are pending (CHAN or MB request), the cache control will grant the EBox request and start a cache EBox cycle to read the register. This decision is made as the cache control advances from IDLE to READY TO GO. The cache control time state generator then clears the CLK EBOX REQ latch and advances directly to the CSH EBOX time state branch. At CSH EBOX T3, the content of the desired register is read into the AR via the EBus register and the cache control asserts MBOX RESP IN and returns to IDLE.

3.3.2.3 EBox Map – The MAP instruction causes the EBox to generate an EBox request for transferring the contents of the addressed page table location to the AR via the EBus register in a manner similar to that described in Subsection 3.3.2.2. The purpose of this instruction is to transform the virtual page address into the physical page address and transfer this address, with its assigned page descriptor bits, from the page table to the AR via the EBus register.

The physical page address, with its assigned page descriptor keys, is stored in the page table. These entries are placed in the page table when needed, as described in Subsection 3.3.5. If a valid entry is not in the page table when the EBox requests to map the address, the MBox will automatically fetch the entry from core and present it to the EBox (KI paging mode only).

3.3.2.4 EBox Read - The EBox initiates an EBox request to read memory whenever an instruction that needs to read memory is executed. (Refer to the hardware reference manual for information relating to classes of instructions.) Note that many instructions do not reference memory.

To read memory, the EBox sets up the request as follows:

a. Loads VMA bits 13-35 with the effective memory address (E) from which the data or instruction is to be read. The EBox also asserts or negates MCL VMA USER to specify whether the reference is to the user or executive address space.

b. Sets up the following signals to specify the type of read request.

1. MCL EBOX CACHE
2. CON CACHE LOOK EN
3. MCL EBOX MAY BE PAGED
4. CON KI PAGING MODE
5. MCL VMA EPT
6. MCL VMA UPT
7. MCL PAGE TEST PRIVATE
8. MCL PAGE ILLEGAL ENTRY
9. MCL PAGE ADDRESS COND

c. Asserts MCL VMA READ and CLK EBOX REQ. MCL VMA WRITE may also be asserted to write-check the page for paged memory references.

NOTE

The EBox can also issue an advance request where CLK EBOX REQ is raised one MBox clock tick before the VMA and the request qualifiers become valid.

This sets up the conditions required for the MBox to service the EBox request to read memory. If the cache control is IDLE, or when the cache control enters its IDLE state and if a higher priority request (MB or CHAN REQ) is not pending, the cache control will grant the EBox request and start a cache EBox cycle to execute the read request (Figure 3-13). This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set, the CLK EBOX REQ latch is cleared, and the PMA is set up to supply the correct physical memory address mixture. The PMA provides the desired memory address mix in response to the request qualifiers from the EBox. The request qualifiers involved in setting up the correct address mix include: MCL EBOX MAY BE PAGED, MCL VMA UPT and MCL VMA EPT, because the EBox may make any of the following types of memory read requests:

a. Read unpaged memory
b. Read paged memory
c. Read an entry in the user process table
d. Read an entry in the executive process table

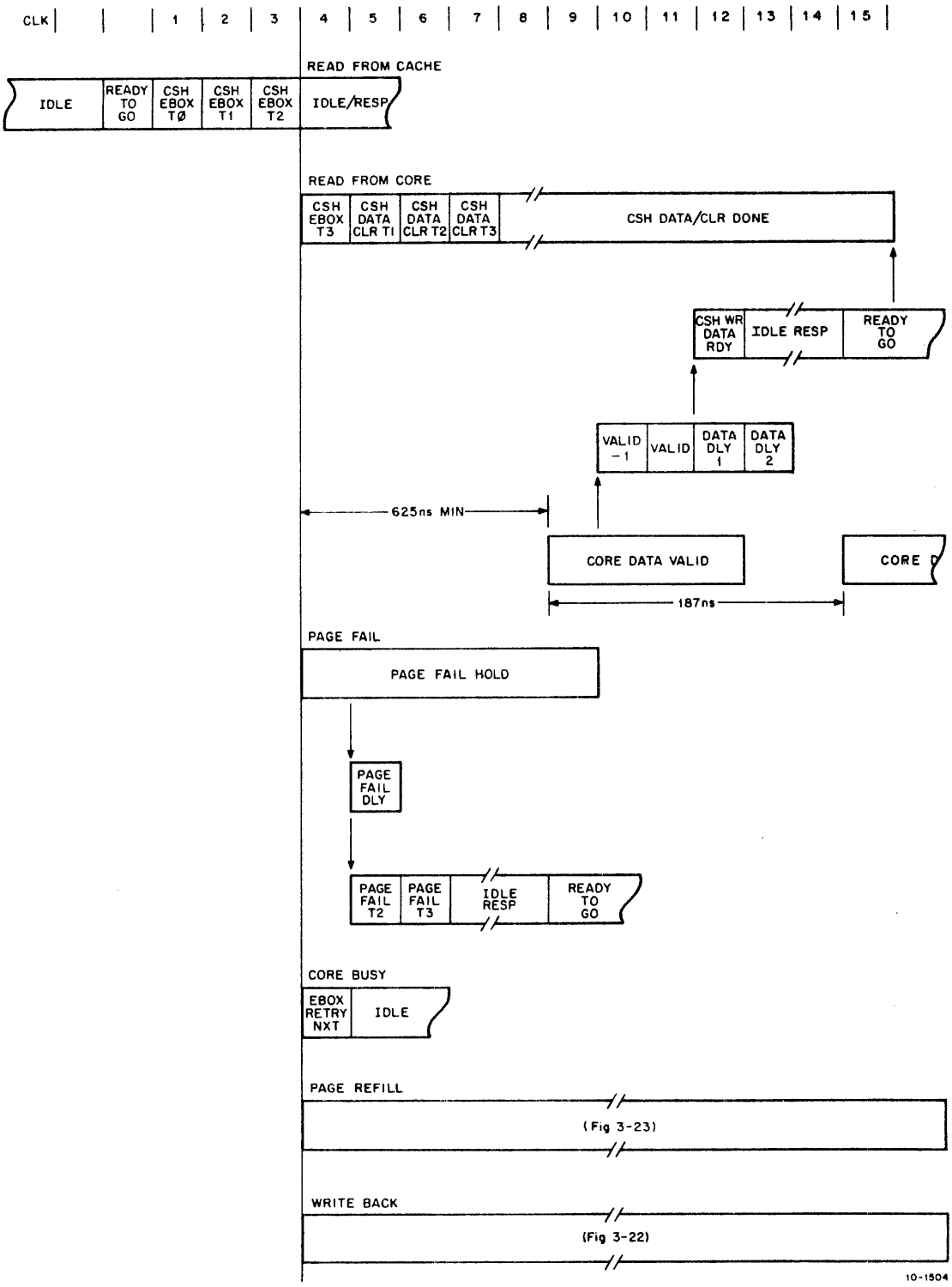


Figure 3-13 EBox Read, Time State Bar Chart

For an unpaged memory reference, the PMA simply supplies the VMA address unchanged, as shown in Figure 3-14.

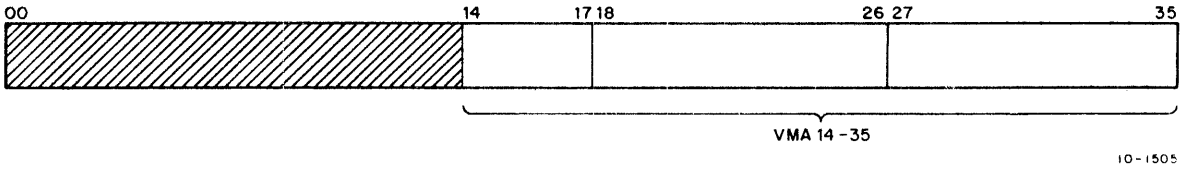


Figure 3-14 PMA Format for Unpaged Memory Read Request

In the case of a paged reference, the valid content of the page table (the physical page address) is combined with (linked with or concatenated) the virtual word address of the page, as shown in Figure 3-15.

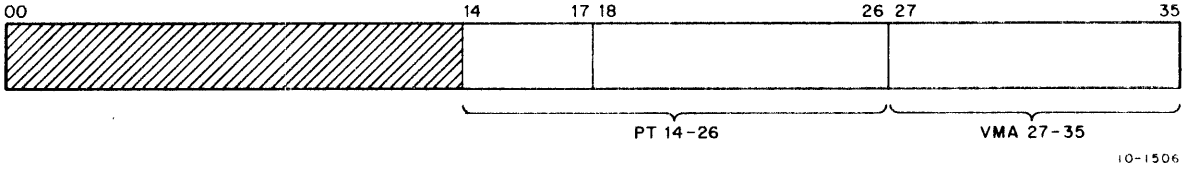


Figure 3-15 PMA Format for Paged Memory Read Request

For references to the process tables, the content of the UBR or EBR (depending on whether MCL VMA UPT or EPT is asserted) is linked with the virtual word address, as shown in Figure 3-16.

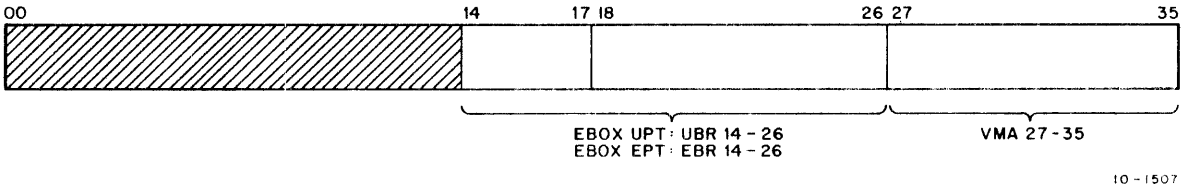


Figure 3-16 PMA Format for EPT or UPT Read Request

From READY TO GO, the time state generator advances to the CSH EBOX time state branch to execute the cache EBox cycle.

MCL EBOX CACHE (LOAD) and CON CACHE LOOK EN are set up by the EBox to relate to the MBox if and how the cache is to be used in satisfying the memory request. Table 3-7 identifies the cache strategies that can be specified by the EBox as related to EBox read requests. For paged memory references, the CACHE bit in the page table also affects the use strategy of the cache. If the CACHE bit is cleared, the page may or may not be cached (depending on the state of CON CACHE LOOK EN) and the MBox will service the request in the same manner as it would if MCL EBOX CACHE (LOAD) was cleared.

Table 3-7 Cache Strategies for Memory Read Requests

CON CACHE LOOK EN	MCL EBOX CACHE (LOAD)	Strategy
0	0	Bypass the Cache and read the requested word from core memory.
0	1	Not used.
1	0	If the requested word is found to be in the Cache (RD FOUND) read the word from the Cache. If the requested word is not found but some of the words of the associated quadword group are in the Cache (ANY VALID MATCH) refill the Cache from core with the non-valid words (core read cycle) and transfer the requested word to the EBox. If the Cache does not contain any of the words of the quadword group read the requested word from core.
1	1	Read the word from the Cache if it is found, otherwise, refill the Cache from core and transfer the requested word to the EBox.

Any of the following cache conditions could prevail when the read request is made.

a. The cache directory has a record of the referenced page in the addressed line and the addressed word in the cache block for which there is a record is valid. This means that the requested word is in the cache.

b. The cache directory has a record of the referenced page in the addressed line and the addressed word in the cache block for which there is a record is not valid but some of the words are valid. This means that the requested word is not in the cache but some of the words of the quadword group are in the cache.

c. The cache has no record of the referenced page in the addressed line and the LRU cache block does not have any written words. This means that none of the words of the quadword group are in the cache and the LRU cache block is not written.

d. Same as (c) except that the LRU cache block is written.

Besides the cache variables described above, the content of the page table also contributes to how a read request is executed when a paged request is made by the EBox. The execution algorithm for an EBox request to read a word from a memory area that is not paged is not affected by the content of the page table.

After clearing the CLK EBOX REQ latch and setting up the cycle latch and the PMA, the cache control time state generator advances from READY TO GO to the CSH EBOX time stage branch to execute the read request. The state generator advances to the CSH EBOX time state branch because an EBox request is granted. CSH EBOX T0 and CSH EBOX T1 serve as a delay to allow for the logic transit time associated with addressing the page table and the cache directory and testing their contents.

NOTE

The cache directory and page table are addressed with the VMA, not the PMA, thereby avoiding the PMA transit time.

At CSH EBox T2, a complex decision is made based on the request qualifiers, the content of the cache directory and, if it is a paged reference, on the content of the page table.

If the EBox requests a word from memory that is paged and the page table contains a valid entry (PT MATCH), the virtual page address is transformed into a physical page address, the page descriptor keys are checked to see if the reference is legal and whether to modify the cache strategy. An entry in the page table is valid if MCL VMA USER and the virtual section address match the content of the page table directory and the NOT VALID bit is cleared. Five page descriptor bits are associated with each page table entry.

1. A - ACCESS
2. W - WRITABLE
3. P - PUBLIC
4. S - SOFTWARE
5. C - CACHE

The ACCESS, WRITABLE, and PUBLIC bits serve as the page access keys. The state of these keys is checked against the EBox request qualifiers to determine if the reference is legal. If the reference is not legal, the Page Fail word is transferred to the EBus register and PAGE FAIL HOLD is asserted to inform the EBox that it made an illegal memory reference. The EBox can then read the EBus register and determine its next course of action. Refer to Subsection 3.3.5 for the case where a valid entry is not found (-PT MATCH) in the page table.

If the CACHE bit is not set, the cache is bypassed and one word is read from core when the request is executed, unless the CON CACHE LOOK EN request qualifier is asserted and some of the words of the quadword group are already in the cache.

a. The following case descriptions apply to those read requests for which CON CACHE LOOK EN and MCL EBOX CACHE (LOAD) are asserted and the CACHE bit of the valid page table entry is set for a paged reference:

   1. For the case where the requested word is in the cache (RD FOUND) the cache control updates the use table at CSH EBOX T2, returns to IDLE, and asserts MBOX RESP IN. The EBox can then strobe the word off the cache data lines. The cache control will not start another cycle to service another request until the EBox takes the data. To inform the MBox that the EBox took the data, the EBox asserts CLK EBOX SYNC D, causing the cache control to advance to READY TO GO to start another cycle if a request is pending.

      NOTE

      The cache control time state generator also advances to CSH EBOX T3 because this state is unconditional. However, this time state will not evoke another time state for this and some other cases.

2. The case where the requested word is not in the cache but some of the words in the quadword group are (ANY VALID MATCH), the time stage generator advances to CSH EBOX T3 to initiate a core read request and hold the address if core is not busy. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At CSH EBOX T3 a core read request is initiated to read from core those words that are not valid in the cache, starting with the word requested by the EBox.

   At the same time the core cycle is started, the cache control time state generator continues with the CSH DATA CLR time state to clear the data in the cache and update the use table.

   NOTE
   The use table was also updated at CSH EBOX T2 for the VALID MATCH case.

   In addition, the cache block number that contained the valid word and the PMA (address bits 27–33) are held as a result of REFILL HOLD to facilitate refilling the cache when the words come in. When the first word comes in from core, it is presented to the EBox and is written into the cache that provided the earlier match using the refill address. The remaining words are moved into the same cache block by initiating an MB cycle as each word comes in.

   The MBox recognizes that a word has come in from core when it receives SBUS DATA VALID. This causes the cache control time state generator to advance sequentially to CORE DATA VALID-2, CORE DATA VALID-1, and CORE DATA VALID. Besides controlling the MB write request and MB load (MB HOLD IN) logic, these time states normalize the transit time difference between the SBUS DATA VALID control path and the SBUS DATA PATH.

   The MB is loaded (-MB HOLD IN) and MBOX RESP IN is asserted at CORE DATA VALID-1 when the first word comes in to inform the EBox that it can take the word.

   NOTE
   The word the EBox requested will come in first. At the CORE DATA VALID time state, a decision is made to determine if the EBox took the word.

   If CLK EBOX SYNC D is asserted at CORE DATA VALID, the EBox took the data directly from core and the cache control, therefore, can terminate the cache EBox cycle simply by testing MB parity, moving the word into the cache, validating the directory, and clearing the appropriate MB WR RQ. The MB WR RQ is cleared at CORE DATA VALID, the cache is updated at CSH WR DATA RDY, and MB parity is tested at DATA DLY1. From CSH WR DATA RDY, the cache control returns to IDLE and then to READY TO GO since CLK EBOX SYNC D is asserted, thereby allowing another request to be serviced.

   If CLK EBOX SYNC D is not asserted at CORE DATA VALID, the EBox did not take the data. In this case, the data is still moved into the cache, the directory is updated, and MB parity is checked at DATA DLY2 instead of 1 but this is not done, and the cache EBox cycle is not terminated until the EBox takes the data. The cache control will then wait in the CSH WR DATA RDY time state until the EBox takes the data from the MB. At that time, the cache control will return to its READY TO GO state via IDLE to service another request.

3. For the case where the cache does not have a valid directory entry (-VALID MATCH) and the LRU cache block does not contain any written words (-CSH LRU ANY WRITTEN), the time state generator advances to CSH EBOX T3 to initiate a core cycle as in the previous case, but this time a request is made for all four words and these words will be moved to the LRU cache block.

   Another difference in the way the request is executed in this case is that the new address is written into the cache directory, the valid bits and the data bits are cleared, and the use table is updated during the CSH DATA CLR time states.

4. For the case where the cache does not have a valid directory entry (-ANY VALID MATCH) and the LRU cache block contains written words (CSH LRU ANY WRITTEN) the time state generator advances from CSH EBOX T2 to WRITEBACK T1 to initiate a writeback cycle. After the writeback cycle is done and core becomes not busy, the EBox request is retried.

   When the EBox issues a paged memory read request and the page table does not contain a valid entry (-PT MATCH) to transform the virtual page address to the physical page address, the cache control will either start a page refill cycle or will inform the EBox that a page fail condition exists. If the EBox specified KI style paging (KI paging mode), the time state generator advances from CSH EBOX T2 to CSH EBOX T3 and then to PAGE REFILL T4 to start a page refill cycle. After the page refill cycle is done the EBox request is retried (refer to Page Refill Cycle description). If the page table still does not contain a valid entry after the request is retried, the time state generator steps through the page fail time states to load the PF HOLD word into the EBus register and to inform the EBox that a page fail condition exists by asserting PAGE FAIL HOLD. For the case when the EBox specifies KL style paging (-KI paging mode), the cache control does not initiate an automatic page refill cycle but informs the EBox that a page fail condition exists by asserting PAGE FAIL HOLD at PAGE FAIL T1. The PF HOLD word is loaded into the EBus register at PAGE FAIL T3. PF EBOX HANDLE is also asserted by the MBox for this case.

b. The following case description applies to those read requests for which CON CACHE LOOK EN is not asserted; it also applies if the cache is not implemented:

If CON CACHE LOOK EN is not asserted (or if the cache does not exist) for the EBox read request, the cache is automatically bypassed and a core read cycle is started to read one word from core. To initiate the core read cycle, the state generator advances from CSH EBOX T2 to CSH EBOX T3 if core is not busy, as described before for reading non-valid words. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At CSH EBOX T3, a core read request is initiated to read the word (ONE WORD RD) the EBox requested from core.

NOTE

At the same time the core read cycle is started, the cache control state generator steps through the CSH DATA CLR time states, as described before, but the use table and the cache directory are not updated at this time because ONE WORD RD is asserted and inhibits this operation.

The MBox recognizes that the word has come in from core when SBUS DATA VALID is asserted. This causes the cache control time state generator to step sequentially through the CORE DATA VALID time states. Besides controlling the MB write request and MB load (MB HOLD IN) logic, these time states normalize the transit time difference between the SBUS DATA VALID control path and the SBUS data path. The MB WR RQ queue is set at CORE DATA VALID-2 to remember which MB is loaded. At CORE DATA VALID-1, the MB is loaded (-MB HOLD IN) and MBOX RESP IN is asserted to inform the EBox that it can take the word. At the CORE DATA VALID time state, a decision is made to determine if the EBox took the word.

If CLK EBOX SYNC D is asserted at CORE DATA VALID, the EBox took the data directly from core and the cache control therefore, can, derminate the CSH EBOX cycle simply by clearing the appropriate MB WR RQ and testing MB parity. MB WR RQ is cleared at CORE DATA VALID; MB parity is checked at DATA DLY 1. At the same time the cache control state generator advances from CORE DATA VALID to DATA DLY 1, the state generator also advances to READY TO GO, allowing another request to be serviced.

If CLK EBOX SYNC D is not asserted at CORE DATA VALID, the EBox did not take the data. In this case, the MEM TO C mixer is switched to select the MB instead of core and the state generator advances to the DATA DLY time state to test MB parity and to wait for the EBox to take the data from the MB. When the EBox takes the data, the EBox asserts CLK EBOX SYNC D which will cause the state generator to advance to READY TO GO, thereby terminating the cache EBox cycle and allowing another request to be serviced.

c. The following case descriptions apply to those read requests for which EBOX CACHE LOOK EN is asserted and MCL EBOX CACHE (LOAD) is not asserted, or MCL EBOX CACHE (LOAD) is asserted but the CACHE bit of the valid page table entry is not set for a paged reference.

1. For the case where the requested word is in the cache (RD FOUND), the cache control updates the use table at CSH EBOX T2, returns to IDLE, and asserts MBOX RESP IN. The EBox can then strobe the word off the cache data lines. The cache control will not start another cycle to service another request until the EBox takes the data. To inform the MBox that the EBox took the data, the EBox asserts CLK EBOX SYNC D causing the cache control to advance to READY TO GO to start another cycle if a request is pending.

2. In the case where the requested word is not in the cache but some of the words in the quadword group are (ANY VALID MATCH), the time state generator advances to CSH EBOX T3 to initiate a core read request and hold the address if core is not busy. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At CSH EBOX T3, a core read request is initiated to read from core those words that are not valid in the cache, starting with the word requested by the EBox.

   At the same time the core cycle is started, the cache control time state generator continues with the CSH DATA CLR time state to clear the data in the cache and update the use table.

NOTE

The use table was also updated at CSH EBOX T2 for the VALID MATCH case.

In addition, the cache block number that contained the valid word and the PMA (address bits 27–33) are held as a result of REFILL HOLD to facilitate refilling the cache when the words come in. When the first word comes in from core, it is presented to the EBox and is written into the cache that provided the match earlier using the refill address. The remaining words are moved into the same cache block by initiating an MB cycle as each word comes in.

The MBox recognizes that a word has come in from core when it receives SBUS DATA VALID. This causes the cache control time state generator to advance sequentially to CORE DATA VALID-2, CORE DATA VALID-1, and CORE DATA VALID. Besides controlling the MB write request and MB load (MB HOLD IN) logic, these time states normalize the transit time difference between the SBUS DATA VALID control path and the SBUS DATA path.

The MB is loaded (-MB HOLD IN) and MBOX RESP IN is asserted at CORE DATA VALID-1 when the first word comes in to inform the EBox that it can take the word.

NOTE

The word the EBox requested will come in first. At the CORE DATA VALID time state, a decision is made to determine if the EBox took the word.

If CLK EBOX SYNC D is asserted at CORE DATA VALID, the EBox took the data directly from core and the cache control therefore, can, terminate the cache EBox cycle simply by testing MB parity, moving the word into the cache, validating the directory, and clearing the appropriate MB WR RQ. The MB WR RQ is cleared at CORE DATA VALID, the cache is updated at CSH WR DATA RDY, and MB parity is tested at DATA DLY1. From CSH WR DATA RDY, the cache control returns to IDLE and then to READY TO GO since CLK EBOX SYNC D is asserted, thereby allowing another request to be serviced.

If CLK EBOX SYNC D is not asserted at CORE DATA VALID, the EBox did not take the data. In this case, the data is still moved into the cache, the directory is updated, and MB parity is checked at DATA DLY2 instead of 1, but this is not done and the cache EBox cycle is not terminated until the EBox takes the data. The cache control will then wait in the CSH WR DATA RDY time state until the EBox takes the data from the MB. At that time the cache control will return to its READY TO GO state via IDLE to service another request.

3. For the case where the cache does not have a valid directory entry (-ANY VALID MATCH), the cache is automatically bypassed and a core read cycle is started to read one word from core. To initiate the core read cycle, the state generator advances from CSH EBOX T2 to CSH EBOX T3 if core is not busy, as described before, for reading non-valid words. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At CSH EBOX T3, a core read request is initiated to read the word (ONE WORD RD) the EBox requested from core.

NOTE

At the same time the core read cycle is started, the cache control state generator steps through the CSH DATA CLR time states, as described before, but the use table and the cache directory are not updated this time because ONE WORD RD is asserted that causes this operation to be inhibited.

The MBox recognizes that the word has come in from core when SBUS DATA VALID is asserted. This causes the cache control time state generator to step sequentially through the CORE DATA VALID time states. Besides controlling the MB write request and MB load (MB HOLD IN) logic, these time states normalize the transit time difference between the SBUS DATA VALID control path and the SBUS data path. The MB WR RQ queue is set at CORE DATA VALID-2 to remember which MB is loaded. At CORE DATA VALID-1, the MB is loaded (-MB HOLD IN) and MBOX RESP IN is asserted to inform the EBox that it can take the word. At the CORE DATA VALID time state, a decision is made to determine if the EBox took the word.

If CLK EBOX SYNC D is asserted at CORE DATA VALID, the EBox took the data directly from core and the cache control, therefore, can terminate the cache EBox cycle simply by clearing the appropriate MB WR RQ and testing MB parity. The MB WR RQ is cleared at CORE DATA VALID and MB parity is checked at DATA DLY1. At the same time the cache control state generator advances from CORE DATA VALID to DATA DLY1, the state generator also advances to READY TO GO, allowing another request to be serviced.

If CLK EBOX SYNC D is not asserted at CORE DATA VALID, the EBox did not take the data. In this case, the MEM TO C mixer is switched to select the MB instead of core and the state generator advances to the DATA DLY time state to test MB parity and to wait for the EBox to take the data from the MB. When the EBox takes the data, the EBox asserts CLK EBOX SYNC D, which will cause the state generator to advance to READY TO GO, thereby terminating the cache EBox cycle and allowing another request to be serviced.

3.3.2.5 EBox Write - The EBox initiates an EBox request to write memory whenever an instruction that needs to write memory is executed. (Refer to the hardware reference manual for information relating to classes of instructions.) Note that many instructions do not reference memory.

To write memory, the EBox sets up the request as follows:

a. Loads VMA 13-35 with the effective memory address (E) into which the data or instruction is to be written. The EBox also asserts or negates MCL VMA USER to specify whether the reference is to the user or executive address space.

b. Sets up the following signals to specify the type of write request.

1. MCL EBOX CACHE
2. CON CACHE LOOK EN
3. MCL EBOX MAY BE PAGED
4. CON KI PAGING MODE
5. MCL VMA EPT
6. MCL VMA UPT
7. MCL PAGE TEST PRIVATE
8. MCL PAGE ILLEGAL ENTRY
9. MCL PAGE ADDRESS COND

c. Asserts MCL VMA WRITE and CLK EBOX REQ.

NOTE

The EBox can also issue an advance request where CLK EBOX REQ is raised one MBox clock tick before the VMA and the request qualifiers become valid.

This sets up the conditions required for the MBox to service the EBox request to write memory. If the cache control is IDLE, or when the cache control enters its IDLE state and if a higher priority request is not pending (MB or CHAN REQ), the cache control will grant the EBox request and start a cache EBox cycle to execute the write request (Figure 3-17). This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set, the CLK EBOX REQ latch is cleared, and the PMA is set up to supply the correct physical memory address mixture. The PMA provides the desired memory address mix in response to the request qualifiers from the EBox. The request qualifiers involved in setting up the correct address mix include: MCL VMA MAY BE PAGED, MCL VMA UPT, and MCL VMA EPT; because the EBox may make any of the following types of memory write requests:

a. Write unpaged memory
b. Write paged memory
c. Write a location in the user process table
d. Write a location in the executive process table

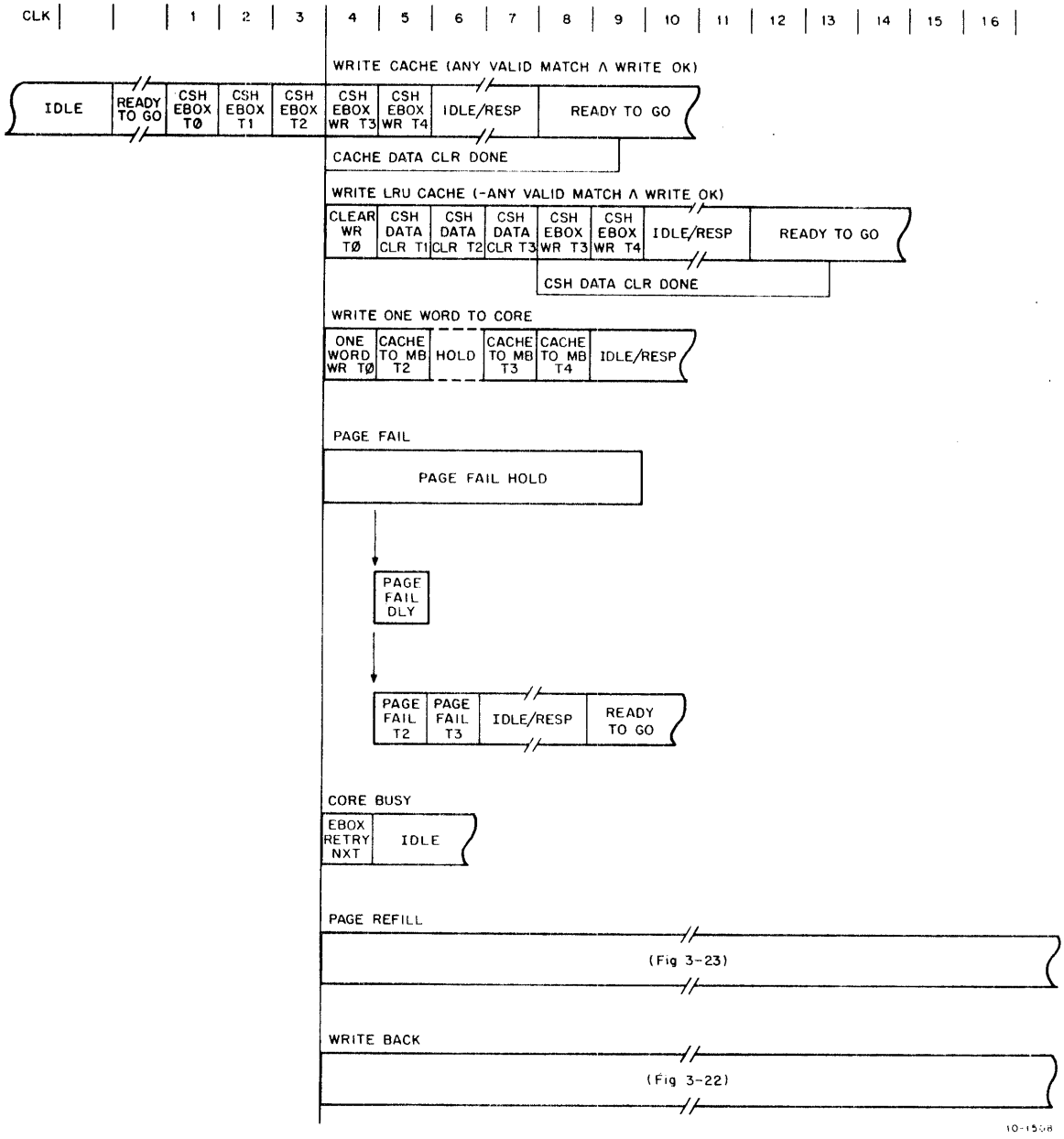


Figure 3-17 EBox Write, Time State Bar Chart

For an unpaged memory reference, the PMA simply supplies the VMA address unchanged, as shown in Figure 3-18.

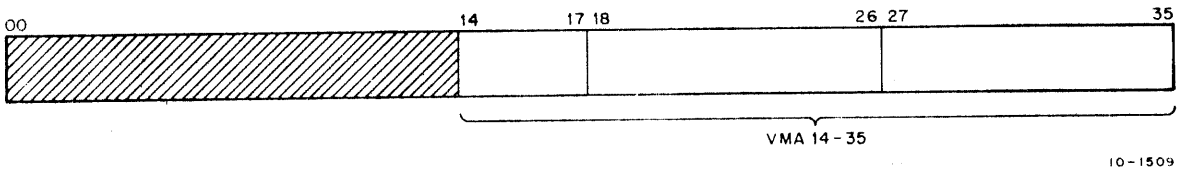


Figure 3-18 PMA Format for Unpaged Memory Write Request

In the case of a paged reference, the valid content of the page table (the physical page address) is combined with (linked with or concatenated) the virtual word address of the page as shown in Figure 3-19.

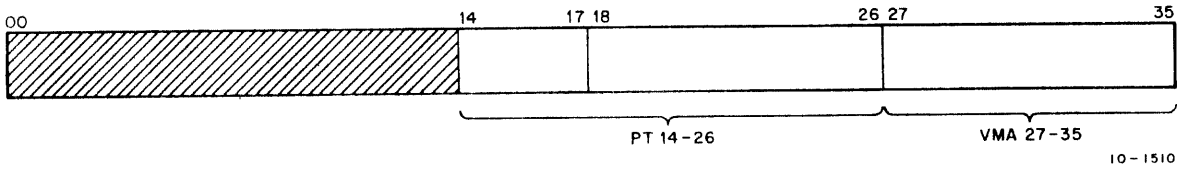


Figure 3-19 PMA Format for Paged Memory Write Request

For references to the process tables, the content of the UBR or EBR, depending on whether MCL VMA UPT or EPT is asserted by the EBox, is linked with the virtual word address of the referenced page, as shown in Figure 3-20.

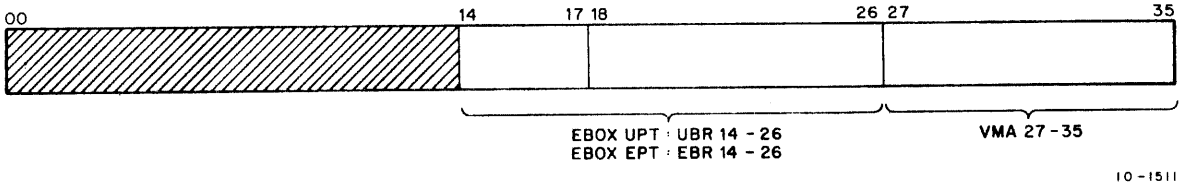


Figure 3-20 PMA Format for EPT or UPT Write Request

From READY TO GO, the time state generator advances to the CSH EBOX time state branch to execute the write request. CON CACHE LOOK EN and MCL EBOX CACHE (LOAD) are set up by the EBox to specify to the MBox if and how the cache is to be used in servicing the memory request. Table 3-8 identifies the cache strategies that can be specified by the EBox as related to the EBox write requests. For paged memory references, the CACHE bit in the page table also affects the use strategy of the cache. If the CACHE bit is cleared, the page may or may not be cached (depending on the state of the CON CACHE LOOK EN), and the MBox will execute the request in the same manner as it would if MCL EBOX CACHE (LOAD) was cleared.

Table 3-8 Cache Strategy for Memory Write Requests

CON CACHE LOOK EN	MCL EBOX CACHE (LOAD)	Strategy
0	0	Bypass the Cache and write the word into core memory.
0	1	Not used.
1	0	If one or more words of the quadword group associated with the word to be written are in the Cache (ANY VALID MATCH), the word is written into the Cache. If none of the words of the quadword group are in the Cache (-ANY VALID MATCH), the word is written into core.
1	1	Write the word into the Cache.

Any of the following cache conditions could prevail when the write request is made.

a. The cache directory has a record of the referenced page in the addressed line and at least one word in the cache block for which there is a record is valid.

b. The cache directory does not have a record of the page and the LRU cache block does not contain any written words.

c. The cache directory does not have a record of the page but the LRU cache block contains some written words.

Besides the cache variables described above, the contents of the page table also contribute to how a write request is executed when a paged write request is made by the EBox. The execution algorithm for an EBox request to write a word into a memory area that is not paged is not affected by the content of the page table.

After clearing the CLK EBOX REQ latch and setting up the cycle latch and the PMA, the cache control time state generator advances from READY TO GO to the CSH EBOX time state branch to execute the write request. The state generator advances to the CSH EBOX time state branch because an EBox request is granted. CSH EBOX T0 and CSH EBOX T1 serve as a delay to allow for the logic transit time associated with addressing the page table and the cache directory and testing their contents.

NOTE

The page table and the cache directory are addressed with the VMA not the PMA thereby minimizing the transit time.

At CSH EBOX T2, a complex decision is made by the cache control, based on the request qualifiers, the contents of the cache directory and, if it is a paged reference on the contents of the page tables.

If the EBox requests a word from a memory area that is paged and the page tables contains a valid entry (PT MATCH), the virtual page address is transformed into a physical page address, the page descriptor keys are checked to see if the reference is legal and whether to modify the cache strategy. An entry in the PT is valid if MCL VMA USER and the virtual section address match the contents of the page table directory and the NOT VALID bit is cleared. Five page descriptor keys are associated with each page table entry:

1. A - ACCESS
2. W - WRITABLE
3. P - PUBLIC
4. S - SOFTWARE
5. C - CACHE

The ACCESS, WRITABLE, and PUBLIC bits serve as page access keys. The state of these keys is compared with the request qualifiers to determine if the reference is legal. If the reference is not legal, the Page Fail word is transferred to the EBus register and PAGE FAIL HOLD is asserted to inform the EBox that it made an illegal memory reference. The EBox can then read the EBus register and determine its next course of action. Refer to Subsection 3.3.5 for the case where a valid entry is not found (-PT MATCH) in the page table.

If the CACHE bit is not set, the cache is bypassed and one word is written into core when the request is executed, unless the CON CACHE LOOK EN request qualifier is asserted and some of the words of the quadword group are already in the cache (ANY VALID MATCH).

a. The following case descriptions apply to those write requests for which CON CACHE LOOK EN and MCL EBOX CACHE (LOAD) are asserted and the CACHE bit of the valid page table entry is set for a paged reference.

   1. For the case where the cache directory has a record of the referenced page in the addressed line and at least one word in the cache block for which there is a record is valid (ANY VALID MATCH), the cache control advances from CSH EBOX T2 to CSH EBOX T3 and to CSH DATA CLR DONE at the same time and updates the use table. CSH DATA CLR DONE is set to facilitate setting the VALID and WRITTEN bits of the cache directory. At CSH EBOX WR T3, a test is made to determine if the cache can be written (WRITE OK). The cache cannot be written (-WRITE OK) if the core control is busy fetching words for the same line in the cache. Even though these words may be moved into another block (there are four blocks per line), the cache EBox cycle to write the cache is aborted to prevent conflict if these words were to be moved into the same block that is to be written. To abort the cache EBox cycle, the state generator advances from CSH EBOX WR T3 to EBOX RETRY NEXT to retry the request. When the request is retried and the core control and cache control have finished moving the words into the cache, the state generator will advance from CSH EBOX WR T3 to WR T4 to write the data in the cache and set the cache directory VALID and WRITTEN bits associated with the word being written. The correct cache block and its directory is written by virtue of having a valid entry in the cache. From CSH EBOX WR T4, the cache control returns to IDLE and asserts MBOX RESP IN.

NOTE

The cache control time state generator also advances to CSH EBOX T3 because this state is unconditional. However, this time state will not evoke another time state for this and some other cases.

2. For the case where the cache directory does not have a record of the referenced page in the addressed line (-ANY VALID MATCH) and the LRU cache block does not contain any written words (-CSH LRU ANY WRITTEN), the cache control advances from CSH EBOX T2 to CLEAR WR T0. At this time state, a test is made to determine if the cache can be written (WRITE OK), as described for the previous case. If the test passes, the state generator advances from CLEAR WR T0 to the CSH DATA CLR time states to write the address into the cache directory, clear the VALID and WRITTEN bits of the LRU cache block and update the use table. The LRU cache block is selected by virtue of not having a valid entry (-ANY VALID MATCH) in the cache. From CSH DATA CLR T3, the state generator advances to both CSH DATA CLR DONE and CSH EBOX WR T3. The state generator advances to CSH DATA CLR DONE to select the LRU cache block by forcing a valid match (FORCE MATCH EN) for that block so that cache can be written. From CSH EBOX WR T3, the state generator advances to CSH EBOX WR T4 to write the data in the cache and set the cache directory VALID and WRITTEN bits associated with the word that is being written. From CSH EBOX WR T4, the cache control returns to IDLE and asserts MBOX RESP IN.

3. For the case where the cache directory does not have a record of the referenced page in the addressed line (-ANY VALID MATCH) and the LRU cache block contains some written words (CSH LRU ANY WRITTEN), the cache control time state generator advances from CSH EBOX T2 to WRITEBACK T1 to initiate a writeback cycle. After the writeback cycle is done, the EBox request is retried.

   When the EBox issues a paged memory write request and the page table does not contain a valid entry (-PT MATCH) to transform the virtual page address to the physical page address, the cache control will either start a page refill cycle or will inform the EBox that a page fail condition exists. If the EBox specified KI style paging (KI paging mode), the time state generator advances from CSH EBOX T2 to CSH EBOX T3 and then to PAGE REFILL T4 to start the page refill cycle (Subsection 3.3.5). After the page refill cycle is done, the EBox request is retried. If the page table still does not contain a valid entry after the request is retried, the time state generator steps through the PAGE FAIL time states to load the PF HOLD word into the EBus register and to inform the EBox that a page fail condition exists by asserting PAGE FAIL HOLD. For the case when the EBox specifies KL style paging (-KI Paging Mode), the cache control does not initiate an automatic page refill cycle but instead informs the EBox that a page fail condition exists by asserting PAGE FAIL HOLD at PAGE FAIL T1. The PF HOLD word is loaded into the EBus register at PAGE FAIL T3. PF EBOX HANDLE is also asserted by the MBox for this case.

b. The following case description applies to those write requests for which CON CACHE LOOK EN is not asserted; it also applies if the cache is not implemented:

If CON CACHE LOOK EN is not asserted (or if the cache does not exist) for the EBox write request, the cache is automatically bypassed and a core write cycle is started to write the word into core after the word is moved to an MB. A one word write cycle is also started when APR EBOX SBUS DIAG is asserted. To move the word into an MB and start the core write cycle, the state generator advances from CSH EBOX T3 to ONE WORD WR T0 if core is not busy. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At ONE WORD WR T0, the MB addressed by PMA 34 and 35 is loaded by clearing MB HOLD IN for one clock tick, and the MB WR RQ queue is set to remember which MB was loaded. The state generator then advances from ONE WORD WR T0 to the CACHE TO MB time state to align with PHASE CHANGE COMING. At CACHE TO MB T4, the core write cycle is started and MBOX RESP IN is asserted. The MB WR RQ queue drives the MB select logic (MB SEL 1-2) to select the MB that contains the word to be written. At the same time the core write cycle is started, the cache control state generator advances to CACHE TO MB T1. From this time state, the cache control returns to IDLE, allowing another request to be serviced. When the memory asserts SBUS ACKN, the MB WR RQ queue is cleared and the core cycle is terminated.

c. The following case descriptions apply to those write requests for which CON CACHE LOOK EN is asserted and MCL EBOX CACHE (LOAD) is not asserted or MCL EBOX CACHE (LOAD) is asserted but the CACHE bit of the valid page table entry is not set for a paged reference.

1. For the case where the cache directory has a record of the referenced page in the addressed line and at least one word in the cache block for which there is a record is valid (ANY VALID MATCH), the cache control advances from CSH EBOX T2 to CSH EBOX WR T3 and to CSH DATA CLR DONE at the same time and updates the use table. CSH DATA CLR DONE is set to facilitate setting the VALID and WRITTEN bits of the cache directory. At CSH EBOX WR T3, a test is made to determine if the cache can be written (WRITE OK). The cache cannot be written (-WRITE OK) if the core control is busy fetching words for the same line in the cache. Even though these words may be moved into another block (there are four blocks per line), the cache EBox cycle to write the cache is aborted to prevent conflict if these words were to be moved into the same block that is to be written. To abort the cache EBox cycle, the state generator advances from CSH EBOX WR T3 to EBOX RETRY NEXT to retry the request if no higher priority requests are pending. When the request is retried and the core control and cache control have finished moving the words into the cache, the state generator will advance from CSH EBOX WR T3 to WR T4 to write the data in the cache and set the cache directory VALID and WRITTEN bits associated with the word being written. The correct cache block and its directory is written by virtue of having a valid entry in the cache. From CSH EBOX WR T4, the cache control returns to IDLE and asserts MBOX RESP IN.

2. For the case where the cache does not have a valid directory entry (-ANY VALID MATCH), the cache is automatically bypassed and a core write cycle is started to write the word into core after the word is moved to an MB. A one word write cycle is also started when APR EBOX SBUS DIAG is asserted. To move the word into an MB and start the core write cycle, the state generator advances from CSH EBOX T3 to ONE WORD WR T0 if core is not busy. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At ONE WORD WR T0, the MB addressed by PMA 34 and 35 is loaded by clearing MB HOLD IN for one clock tick, and the MB WR RQ queue is set to remember which MB was loaded. The state generator then advances from ONE WORD WR T0 to the CACHE TO MB time state to align with PHASE CHANGE COMING. At CACHE TO MB T4, the core write cycle is started and MBOX RESP IN is asserted. The MB WR RQ queue drives the MB select logic (MB SEL 1-2) to select the MB that contains the word to be written. At the same time the core write cycle is started, the cache control state generator advances to CACHE TO MB T1. From this time state, the cache control returns to IDLE, allowing another request to be serviced. When the memory asserts SBUS ACKN, the MB WR RQ queue is cleared and the core cycle is terminated.

3.3.2.6 EBox Read-Pause-Write – A read-pause-write request from the EBox is serviced by the MBox by executing a read operation followed by a write operation, into the same location. To issue this type of request, the EBox asserts MCL VMA READ, MCL VMA PAUSE, MCL VMA WRITE, CLK EBOX REQ, and the appropriate request qualifiers (refer to EBox read and EBox write descriptions). After the read operation is completed, the EBox may modify the data and will assert CLK EBOX REQ and MCL VMA WRITE to write the word back to the same memory location. If the MBox finds that the cache is to be bypassed, the MBox will read one word from core, present the word to the EBox, and wait until the EBox issues the write request. When the write request is issued, the MBox will write the word into core memory. The consequence of bypassing the cache for this type of memory request is that core remains busy for the entire operation, thereby preventing the channels from getting a core cycle.

3.3.2.7 EBox Write-Check – The EBox initiates an EBox request to write-check a page whenever an instruction that will ultimately cause a request to move a word to paged memory is executed. (Refer to the hardware reference manual for information relating to classes of instructions.)

To write-check a paged memory location, the EBox sets up the request as follows:

a. Loads VMA bits 13–35 with the effective memory address (E) of the location for which the write-check operation is to be performed. The EBox also asserts or negates MCL VMA USER to specify whether the reference is to the user or executive address space.

b. Sets up the following signals to specify the type of write request for which the write-check is to be made.

   1. CON CACHE LOOK EN
   2. MCL EBOX MAY BE PAGED
   3. CON KI PAGING MODE
   4. MCL VMA EPT
   5. MCL VMA UPT
   6. MCL PAGE TEST PRIVATE
   7. MCL PAGE ILLEGAL ENTRY
   8. MCL PAGE ADDRESS COND

c. Asserts MCL VMA PAUSE, MCL VMA WRITE, and CLK EBOX REQ.

This sets up the conditions required for the MBox to service the EBox request to write-check a memory location. If the cache control is IDLE, or when the cache control enters its IDLE state and a higher priority request is not pending (MB or CHAN REQ), the cache control will grant the EBox request and start a cache EBox cycle to execute the write-check operation. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set. From READY TO GO, the cache control time state generator advances to the CSH EBOX time state branch because an EBox request is granted. CSH EBOX T0 and CSH EBOX T1 serve as a delay to allow for the logic transit time associated with addressing the page table and testing its content.

NOTE

The page table is addressed with the VMA, not the PMA, to avoid the PMA transit time, thereby minimizing this time.

At CSH EBOX T2, a decision is made by the cache control based on the request qualifiers and the content of the page table. If the page table contains a valid entry (PT MATCH), the page descriptor keys are checked to see whether the reference is legal. An entry in the page table is valid if MCL VMA USER and the virtual section address match the contents of the page table directory and the NOT VALID bit is cleared. Associated with each page table entry are five page descriptor keys:

1. A - ACCESS
2. W - WRITABLE
3. P - PUBLIC
4. S - SOFTWARE
5. C - CACHE

The ACCESS, WRITABLE, and PUBLIC bits serve as page access keys. The state of these keys are compared with the request qualifiers to determine if the page is writable.

If the page has access privileges and is writable, the MBox simply responds by asserting MBOX RESP IN. If the page is restricted or is not writable, the Page Fail word is transferred to the EBus Register and PAGE FAIL HOLD is asserted by the MBox to inform the EBox that the page-check failed. The EBox can then read the EBus register and determine the next course of action.

For the case where a valid entry is not found in the page table (-PT MATCH), refer to Subsection 3.3.5.

3.3.2.8 **Write Refill RAM** - The Ebox initiates an EBox request to write a word into the refill RAM whenever the BLKO APR instruction is executed. Each time this instruction is executed, one 3-bit data word is written into the addressed location of the refill RAM.

To write a word into the refill RAM, the EBox sets up the request as follows:

a. Loads VMA bits 18-20 with the data to be written into the refill RAM.

b. Loads VMA bits 27-33 with the appropriate address to select the desired location in the Refill RAM.

c. Asserts APR EN REFILL RAM WR, MCL VMA READ, and CLK EBOX REQ.

This sets up the conditions required for the MBox to service the EBox request to load one word into the refill RAM. If the cache control is IDLE, or when the cache control enters its IDLE state and if a higher priority request is not pending (MB or CHAN REQ), the cache control will grant the EBox request and start a cache EBox cycle to execute the request. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set and the address is gated from the VMA to the refill RAM via the refill RAM address mixers. These mixers are set up by APR EN REFILL RAM WR to select the correct address. The APR EN REFILL RAM WR signal also sets up the data input mixer for the lookup table. From READY TO GO, the cache control time state generator advances to CSH EBOX T0, T1, and T2, in sequence. At CSH EBOX T2, the CSH USE HOLD flip-flop is set to hold the address and data, and CSH REFILL RAM WR is asserted to write the data into the addressed location of the lookup table. From CSH EBOX T2, the cache control returns to IDLE and asserts MBOX RESP IN.

3.3.2.9 SBus Diagnostic Cycle – The EBox initiates an EBox request to execute an SBus diagnostic cycle when the EBox executes the BLKO PI instruction. Whenever this instruction is executed, a 36-bit control word is transferred from the EBox AR to the core memory system via the data lines and a status word, which is specified by the control word, is returned to the EBox from the core memory system.

To execute an SBus diagnostic cycle, the EBox sets up the request as follows:

a. Loads the AR with the SBus diagnostic control word to be transferred to the core memory system.

b. Asserts APR EBOX SBUS DIAG and CLK EBOX REQ.

This sets up the conditions required for the MBox to service an EBox request for executing an SBus diagnostic cycle. If the cache control is IDLE, or when the cache control enters its IDLE state and if a higher priority request is not pending (MB or CHAN REQ), the cache control will grant the EBox request and start a cache EBox cycle to execute the SBus diagnostic cycle. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set. From READY TO GO, the time state generator advances to the CSH EBOX time state branch to move the control word from the AR to the MB, and to start the SBUS DIAG CYC counter. To move the control word into an MB, the state generator advances from CSH EBOX T3 to ONE WORD WR T0, if core is not busy. If core is busy, the state generator advances instead to EBOX RETRY NEXT to retry the request. At ONE WORD WR T0, the MB addressed by PMA 34 and 35 (which may point to any one of the four MBs) is loaded by clearing MB HOLD IN for one clock tick, and the MB WR RQ queue is set to remember which MB was loaded. The state generator then advances from ONE WORD WR T0 to the CACHE TO MB time states to align with PHASE CHANGE COMING and start the SBUS DIAG CYC counter.

NOTE

The MB WR RQ queue drives the MB select logic (MB SEL 1-2) to select the MB that contains the diagnostic control word that is to be transferred to the core memory system.

As the SBUS DIAG CYC counter steps through its states it causes:

a. SBUS DIAG to be asserted for four MBox clock ticks.

b. MB WR RQ queue to be cleared.

c. MBOX RESP IN to be asserted at the time the requested word is available on the SBus data lines.

When the EBox senses MBOX RESP IN, it simply strobes the cache data lines to transfer the data word from the SBus data lines to the AR. At the same time MBOX RESP IN is asserted, the cache control also returns to IDLE, allowing another request to be serviced.

3.3.3 Cache MB Cycle

MB requests are issued by the core control during a core read cycle to move words that have come in from core from the MB to the cache. The first word, which is the word the EBox requested, is presented to the EBox and is moved into the cache before the cache EBox cycle is terminated. Subsequent words, however, are moved into the cache by executing a cache MB cycle (Figures 2-6 and 3-21). MB requests are assigned the highest priority and are granted cache cycles before another EBox request, a channel request, or a CCA request. This is necessary because the words coming in from core must be moved into the cache before another core cycle can be started. If an MB request is not pending and core is still busy because all words have not yet come in, EBox requests will be granted only to read from or write into the cache but will be aborted if the request results in a core reference. In this case, the request will be retried every time a word comes in from core until the retried request succeeds, which will occur when core becomes not busy and a channel request is not pending.

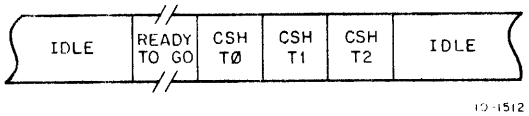


Figure 3-21 Cache MB Cycle, Time State Bar Chart

After the first word is moved into the cache and is taken by the EBox, the cache control returns to READY TO GO. While core is busy, only EBox and MB requests will be granted by the cache control because CCA requests and CHAN requests are disabled as long as core is busy. The MB request has the higher priority to move the words from the MB to the cache as fast as possible to free the MBs. The MBox recognizes that another word has come in from core when it receives SBUS DATA VALID. This triggers the core data valid time state chain and causes MB 0–3 WR RQ and MB REQ IN to be asserted.

If the cache control is not executing an EBox request at the time MB REQ IN is asserted, the MB request is granted and the state generator advances to READY TO GO to execute the cache MB cycle. At READY TO GO the CSH MB CYC latch is set. From READY TO GO the cache control advances to CSH T0 because a request other than an EBox request (ANY REQ) is granted (Figure 3-21).

Time states CSH T0, T1, and T2 enable the refill address and match control to write the data and associated valid bit in the appropriate cache block. The cache block that is written into is either that block that provided a valid match during the cache EBox cycle or the LRU block if no match occurred. From CSH T2 the cache control time state generator advances to IDLE and then to READY TO GO after clearing the MB 0–3 WR RQ since the current cycle is not an EBox cycle. The cache control is then ready to service another request. As long as core is busy, only EBox and MB requests will be granted by the cache control.

3.3.4 Cache Writeback Cycle

Words written into the cache by the EBox are written back to core to update the core copy before the contents of the LRU Cache block is supplanted with a word(s) from another page. Written words in the cache are also written back to core when the EBox issues a request to clear the cache which occurs when the EBox executes a "sweep" instruction to validate core.

During the course of executing a cache EBox cycle to service an EBox read or write request, the decision to start a writeback cycle is made at CSH EBOX T2 (Figure 3-22). CSH EBOX T0 and CSH EBOX T1 serve as a delay to allow for logic transit time associated with addressing the cache directory and testing its contents (refer to cache EBox cycle description and EBox Read/Write request descriptions).

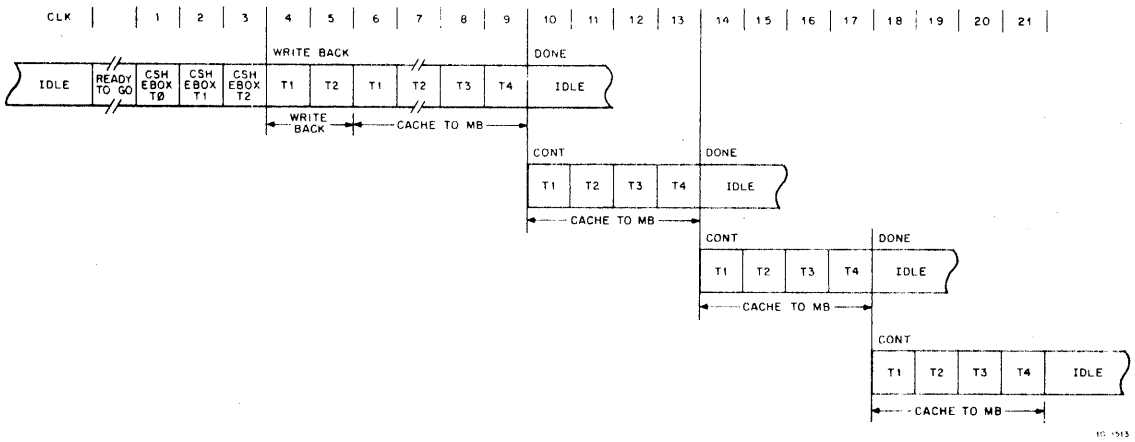


Figure 3-22 Cache Writeback Cycle, Time State Bar Chart

At CSH EBOX T2, the contents of the cache are checked to see if any written words are in the LRU cache block. The function LRU ANY WRITTEN∧ – ANY VALID MATCH indicates that none of the four addressed cache blocks contain any words from the referenced page but the LRU cache block contains one or more words from another page that have been written by the EBox. It is this condition, if core is not busy, that causes the cache control time state generator to advance from CSH EBOX T2 to WRITEBACK T1, thereby initiating the writeback cycle. If core is busy at CSH EBOX T2, the time state generator does not advance to WRITEBACK T1 but advances to EBOX RETRY NEXT to retry the request until core is freed.

From WRITEBACK T1 the state generator advances to WRITEBACK T2, sets the CLK EBOX REQ latch, loads the CSH WRITEBACK CYC latch, and selects the desired address mixture from the PMA. Note that the cache control time state generator does not transgress IDLE and READY TO GO to start the writeback cycle, but instead forces the writeback cycle by setting the CSH WRITE-BACK CYC latch and selecting the desired address mixture from the PMA, thereby bypassing the priority request grant logic. The priority request grant logic is inhibited from granting CHAN and CCA requests during WRITEBACK T1 to block these potential inputs from the cycle latch to start the writeback cycle. The CLK EBOX REQ latch is set to cause the EBox request to be retried after the writeback cycle is done. The address mix includes the contents of the cache directory (CAM 14–26), the quadword address which consists of VMA 27–33 and RQ 1–2.

As the state generator advances from WRITEBACK T2 to CACHE TO MB T1, the cache block number of interest (LRU cache block in the case of a writeback cycle or the cache block that provided the match in the case of a CCA cycle) is latched so that the written bits for that cache block can be cleared. At the same time, the MB WR RQ, core RQ, and CTOMB RQ logic are set up. The state generator then steps through the CACHE TO MB time states to move the written words from the

cache to the associated MBs. The CTOMB WD request logic supplies the word address (CACHE TO MB 34–35) for the cache block of interest and drives the MB HOLD IN logic to generate the appropriate MB load pulse at CACHE TO MB T3. At CACHE TO MB T4, the associated CTOMB WD RQ is cleared. The MB WR RQ logic is set up to remember which MBs received a word from the cache as the state generator steps through the CACHE TO MB time state so that they can be presented to the SBus data lines. After the first written word is moved from the Cache to the MB, the state generator starts the core write cycle and latches the SBus address at CACHE TO MB T4.

The SBUS ACKN pulse clears current MB WR RQ to select the next MB that has a word. The core write cycle is started after the first written word is moved into the MB. Core can be started at this time because it takes only four clock ticks to move one word from the cache to the MB, which is faster than the core control can write the words into memory. After all the written words are moved into the MBs, the cache control time state generator advances to IDLE and to READY TO GO because the CSH EBOX CYC latch is not set. When the time state generator reaches READY TO GO, core will still be busy and, therefore, a request requiring a core cycle cannot be executed. Therefore, neither a CHAN nor CCA request will be granted by the REQ GRANT logic. This allows the EBox request to be retried immediately. If a core cycle is not needed in executing the request, as in the case of a write request, the request is satisfied by writing the cache and asserting MBOX RESP IN. If, however, the request is an EBox read request, it must be retried again since a core cycle will be needed. When core is freed, the priority request grant logic is again fully enabled to grant requests on a priority basis. If both a CHAN and an EBox request are pending at that time, the CHAN request will be granted first, preventing the EBox from getting two core cycles in a row, thereby, holding up the channels.

During the course of executing a cache CCA cycle to service an EBox cache clear request (LOAD CCA REG) the decision to start a writeback cycle is made at CSH T3 (refer to cache CCA cycle description). At CSH T3, the contents of the cache are checked to see if any written words are in the selected cache block. If any written words are found, and the EBox request to clear the cache included the validate core qualifier (CSH CCA VAL CORE – IR AC11 = 1), the cache control time state generator advances from CSH T3 to WRITEBACK T1, thereby initiating the writeback cycle, as described previously.

NOTE
The CLK EBOX REQ latch is not set for this case.

3.3.5 Cache Page Refill Cycle (KI Mode Only)

The page table is refilled automatically in the KI paging mode every time the EBox makes a paged memory reference for which a valid entry is not found in the page table. For KL paging mode, the EBox executes the refill. A valid entry is in the page table if the virtual section address (user or executive) from the VMA matches the contents of the page table directory and the NOT VALID bit is cleared. During the course of executing a cache EBox cycle to service an EBox map, EBox read, write or write-check request, the decision to start a page refill cycle is made at CSH EBOX T2 (Figure 3-23). CSH EBOX T0 and CSH EBOX T1 serve as a delay to allow for logic transit time associated with addressing the page table and directory and testing their contents. If the page table does not contain a valid entry (–PT MATCH) and a page refill cycle has not yet been executed for the current EBox request, the page test logic asserts PAGE REFILL instead of PAGE OK. The presence of this condition is sensed at CSH EBOX T2 to advance the state generator to EBOX RETRY NEXT and to CSH EBOX T3, simultaneously. If core is busy, the request is retried. If core is not busy, on the next clock tick the state generator advances to PAGE REFILL T4 and to CACHE IDLE, and the priority request grant logic is forced to grant a page refill cycle by disabling any CHAN and CCA requests that may be pending.

NOTE
An MB request will not be pending because this path is taken only if core is not busy.

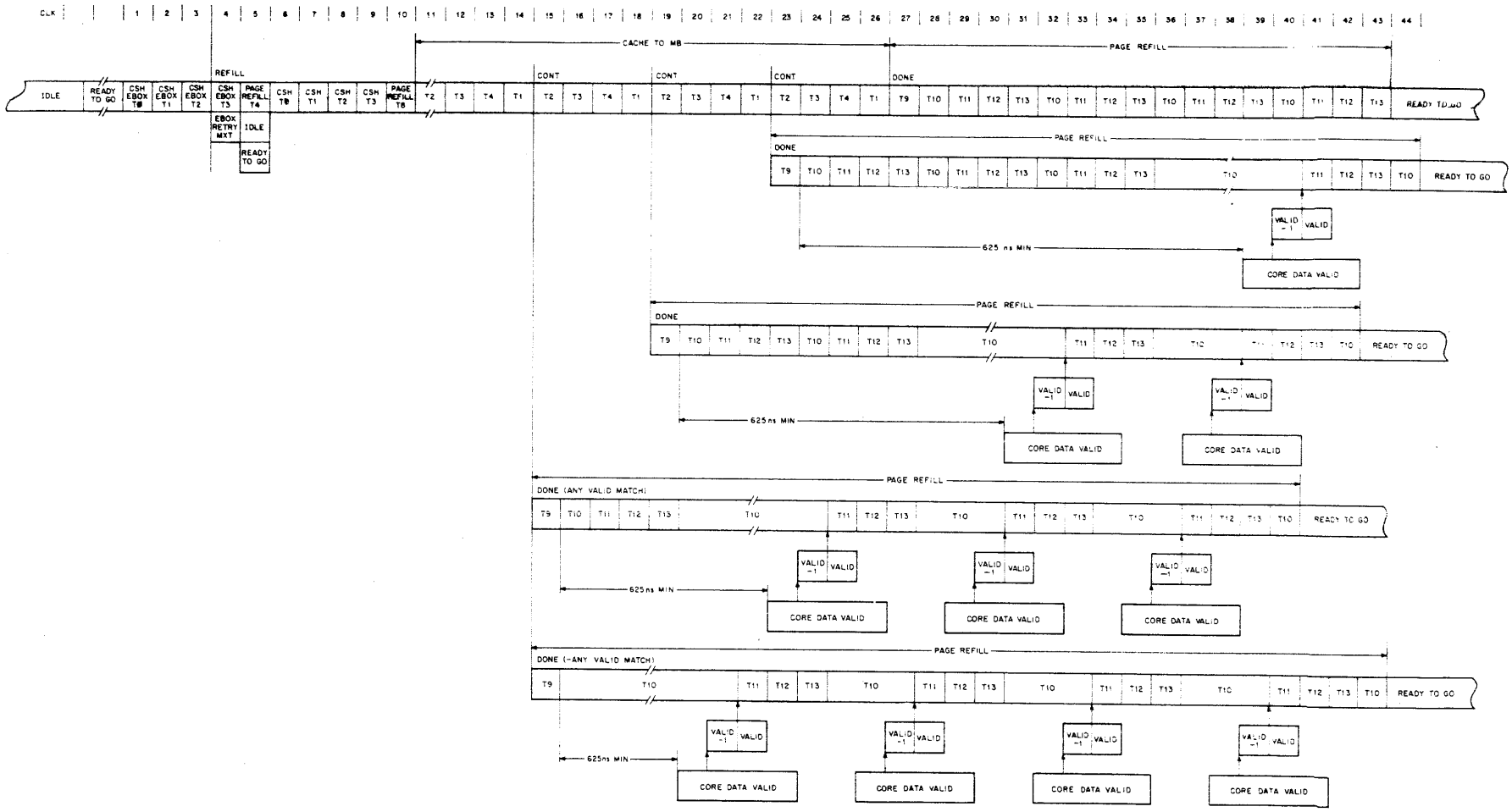


Figure 3-23 Cache Page Refill Cycle, Time State Bar Chart

Because PAGE REFILL T4 is set at the same time the state generator advances to CACHE IDLE, the state generator advances directly to READY TO GO to force the page refill cycle. At READY TO GO, the CSH PAGE REFILL CYC latch is set and the PMA is set up to supply the correct memory address to fetch the page table entry from the appropriate process table. The address mix depends on whether the memory reference is to the user or the executive address space. If the memory reference is to the executive address space, the specific address mix also depends on whether the reference is to the "per process area," to the upper executive area, or to the lower executive area. Consequently, depending on the state of MCL VMA USER (1 = User space; 0 = Executive space), and virtual page address (VMA 18–26), one of four possible addresses will be configured.

For the case where the EBox makes a memory reference to the user address space, all of which is paged, the SBus address for the page refill cycle is configured as shown in Figure 3-24.

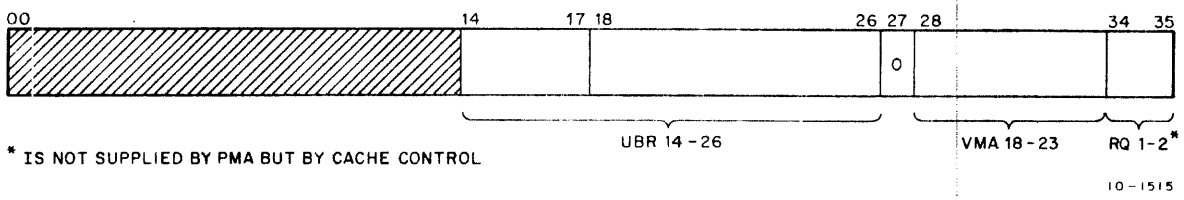


Figure 3-24 SBus Address Format for User Page Refills

UBR 14–26 points to the physical page in core that contains the user process table; VMA 18–23 points to the quadword in the process table that contains the page table entry of the referenced virtual page and RQ 1-2 (output of priority encoder E28 on MBX2) points to the first word in the quadword group that was not found in the cache. Bit 27 of the SBus address is jammed to "zero" to select the lower half (locations $0–377_8$) of the user process table, which contains the 512 page table entries (two entries per location) for the user address space.

For the case where the EBox makes a memory reference to the lower executive address space (pages $000–337_8$), the SBus address for the page refill cycle is configured as shown in Figure 3-25.

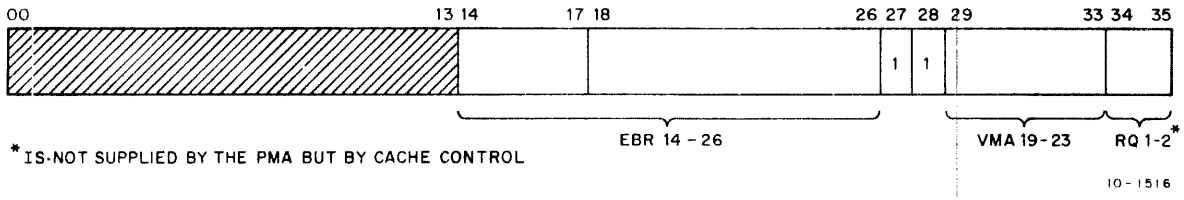


Figure 3-25 SBUS Address Format for Executive Page (Pages $000–337_8$)Refills

EBR 14–26 points to the physical page in core that contains the executive process table; VMA 19–23 points to the quadword location in the process table that contains the page table entry of the referenced virtual page; and RQ 1-2 points to the first word in the quadword group that was not found in the cache. Bits 27 and 28 are jammed to "one" (6XX) to select the upper quarter (locations 600–777) of the executive process table, of which locations 600–757 contain the 224 page table entries (two entries per location) for the lower executive address space.

For the case where the EBox makes a memory reference to the upper executive address space (pages $400–777_8$), the SBus address for the page refill cycle is configured as shown in Figure 3-26.

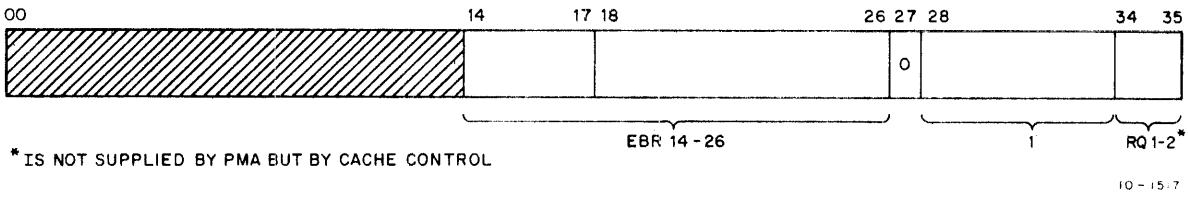


Figure 3-26 SBus Address Format for Executive Page (Pages $400–777_8$) Refills

EBR 14–26 points to the physical page in core that contains the executive process table; VMA 18–23 points to the quadword location in the process table that contains the page table entry of the referenced virtual page; and RQ 1-2 points to the first word in the quadword group that was not found in the Cache. Bit 27 of the SBus address is jammed to "zero" to select the lower half (locations $000–377_8$) of the executive process table, of which locations $200–377_8$ contain the 256 page tables entries (two entries per location) for the upper executive address space.

For the case where the EBox makes a memory reference to the paged executive address space defined to be the "per process area" (pages $340–377_8$), the SBus address for the page refill cycle is configured as shown in Figure 3-27.

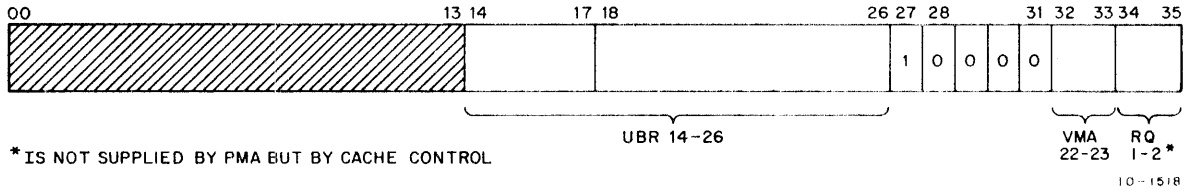


Figure 3-27 SBus Address Format for Executive Page (Pages $340–377_8$) Refills

UBR 14–26 points to the physical page in core that contains the user process table; VMA 22 and 23 point to the quadword location in the process table that contains the page table entry of the referenced virtual page; and RQ 1-2 points to the first word in the quadword group that was not found in the cache. Bits 27–31 are jammed to $4XX_8$ to select the upper half (locations 400–777) of the user process table, of which locations $400–417_8$ contain the 32 page table entries (two entries per location) for the paged executive address space defined to be the "per process area."

From READY TO GO the cache control state generator advances to CSH T0 because a request other than an EBox request (ANY REQ) is granted. The state generator then advances to PAGE REFILL T8 via CSH T1, T2, and T3; to set up the MB WR RQ, core RQ 1-2, and CTOMB WD RQ logic, and latch the SBus address. From PAGE REFILL T8, the state generator advances to the CACHE TO MB time states to move any valid words from the cache to the associated MBs. The CTOMB WD request logic supplies the word address (CACHE TO MB 34–35) for the cache block of interest and drives the MB HOLD IN logic to generate the appropriate MB load pulse at CACHE TO MB T3. At CACHE TO MB T4, the associated CTOMB WD RQ is cleared. The MB WR RQ logic is set up to remember which MBs receive a word from the cache as the state generator steps through the CACHE TO MB time states (or from core) so that they can be moved into the page table. MB SEL HOLD is asserted if any one MB received a word. After all valid words are moved from the cache to the MBs, the state generator advances to PAGE REFILL T9 to start a core read cycle for those words in the quadword group that were not in the cache (RD NON VALID WDS). From PAGE REFILL T9, the state generator advances to PAGE REFILL T10. If any valid words were written into the MBs, the state generator steps through PAGE REFILL T11, T12, T13, and back to T10 because MB SEL HOLD will be asserted. MB SEL HOLD is asserted whenever an MB WR RQ is set. As the state generator advances through these states, the word from the selected MB is written into the page table, the associated MB WR RQ is cleared, and the next highest priority MB that contains a word is selected so that the process can continue. This continues until all the words in the MBs have been written into the page table. At the same time, the core control will clear appropriate MB 0–3 HOLD IN for one clock tick to move the words coming in from core into the MB and set the associated MB WR RQ to inform the cache control that another word has arrived and can be written into the page table. After all the requested words have been received from core (see core control description), and have been written into the page table, core is freed (–CORE BUSY) allowing the state generator to advance from PAGE REFILL T10 to READY TO GO. At the same time the state generator advances to READY TO GO, the REFILL COMP latch is set to remember that a refill cycle for the current EBox request was made. The fact that a refill cycle was executed must be known when the EBox request is retried to prevent another refill cycle from being started.

At READY TO GO, a new cycle can be started. If a CHAN REQ is not pending, another cache EBox cycle is started to retry the request. If the page test does not pass (PAGE FAIL) during the second pass through the cache EBox cycle, a page fail signal is sent to the EBox. Several conditions, based on the current mode the EBox is operating in and the status of the page descriptors, must be met for the page test to pass (PAGE OK) (refer to Pager description).

3.3.6 Cache CCA Cycle

CCA requests are issued by the cache clearer control after it is initialized to validate core and/or invalidate the cache (Subsection 3.5). The cache clearer is initialized when the CCA register is loaded by the EBox (cache Sweep instruction is executed by the EBox). CCA requests are assigned the lowest priority and are granted cache cycles only if no other requests (MB, CHAN, or EBOX) are pending and core is not busy. Depending on the cache clearer qualifiers presented to the cache clearer control by the EBox when the request to load the CCA register was made, the cache control, when executing the cache CCA cycle, initiates writeback cycles for those words that are written and/or clears the valid and written bits in the cache and updates the use table for a single page or for the entire Cache. A summary of CCA cycle variations is presented in Table 3-9.

If the cache control is IDLE, or when the cache control enters its IDLE state and no higher priority requests are pending and if core is not busy, the cache control grants the CCA request and starts a cache CCA cycle (Figure 3-28). This decision is made as the cache control time state generator advances from IDLE to READY TO GO. The cache control will not advance to READY TO GO if the previous cycle was a cache EBox cycle and the EBox has not yet asserted CLK EBOX SYNC D.

Table 3-9 Cache CCA Cycle Variations

ONE PAGE	VAL CORE	INVAL CSH	Function
0	0	1	Update Use Table and clear VAL and WR bit for entire Cache one block at a time.
0	1	0	Writeback all written words in the Cache by initiating a writeback cycle for each Cache block that is written.
0	1	1	Perform both of the above. First initiate the writeback, then invalidate the Cache. CCA register is decremented by 1 to check each block in the Cache.
1	X	X	Same as above except that only those lines containing words from a specific page (specified by CCA register bits 14-26) are effected. If a Cache line does not contain any words from that page nothing is done. CCA register is decremented by 4 to check each line in the Cache.

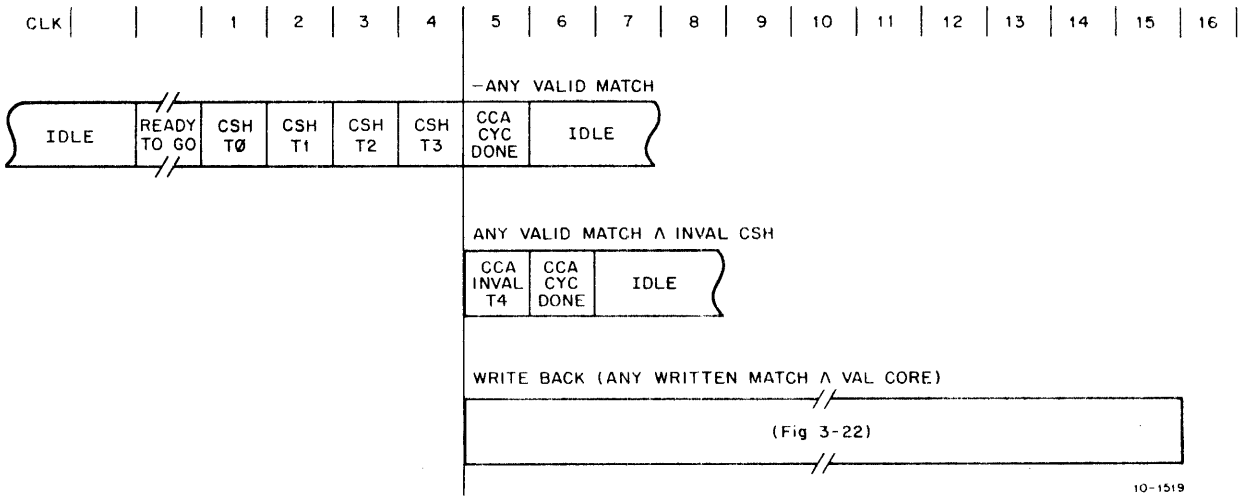


Figure 3-28 Cache CCA Cycle, Time State Bar Chart

This exception is necessary to satisfy EBox read requests because the EBox will take data only when CLK EBOX SYNC D is asserted. At READY TO GO, the CSH CCA CYC latch is set and the PMA is set up to select the address from the CCA register. From READY TO GO, the time state generator advances to the CSH time state branch to execute the cycle. CSH T0, T1, and T2 serve as a delay to allow for logic transit time associated with addressing the cache directory and testing its contents. One extra time state is needed to provide adequate delay in this time state branch because the address is supplied via the PMA instead of the VMA. At CSH T3 a decision is made based on the contents of the cache directory and the cache clearer control.

3.3.6.1 **One Page** – If the CCA request is for one page (CSH CCA ONE PAGE is asserted) then the entire cache is checked, one line at a time, to see if the line contains valid entries from the page specified by the CCA register. Any valid entries for which the cache directory address matches the contents of the CCA register (ANY VALID MATCH) are invalidated and/or are written back to core if they are also written (ANY WRITTEN MATCH). Two passes through the cache control (cache cycles) are required for each cache line to both validate core and invalidate the cache. The first pass causes a writeback cycle to be initiated at WRITEBACK T1 for the written words. During the write-back cycle, the written words are moved to the MBs. The corresponding written bits in the cache directory are cleared and a core write cycle is started. During the second pass, all the valid bits in the cache block that contained the valid entries are cleared and the use table is updated after CCA INVAL T4 if the cache is to be invalidated. The correct cache block in the line is selected by asserting REFILL HOLD. The cache control then advances to CCA CYC DONE. If the cache is not to be invalidated, the cache control bypasses CCA INVAL T4 and advances to CCA CYC DONE. At CCA CYC DONE, the CCA cache line counter is decremented by 1 (CCA register is decremented by 4) to advance the address to point to the next cache line in preparation for the next CCA cycle. If the counter overflows (carry), which means that all 128 cache lines have been taken care of, then the CCA REQ latch is cleared and no further requests for cache cycles will be initiated.

3.3.6.2 **All Pages** – If the CCA request is for all pages (-CSH CCA ONE PAGE) then the entire Cache is checked, one cache block at a time, to see if the cache contains any written entries. Any entries in the cache that have been written are written back to core and/or all valid entries in the cache are invalidated. To accomplish this, the CCA register is decremented by 1 instead of four to permit the cache control to examine the contents of each cache block by forcing a valid match for the cache that is pointed to by bits 34 and 35 of the CCA.

3.3.7 Cache Channel Cycle

Channel requests are issued by the channel control to move data, CCWs, or status information between the channel buffers in the MBox and core memory. As words are moved from the channel to core (channel write), a cache cycle is executed to invalidate any valid words in the cache if CON CACHE LOOK EN is set (Figure 3-29). When words are moved from core to the channel (channel reads), a cache cycle is executed to pick up any words that are valid in the cache provided CON CACHE LOOK EN is set. This ensures that mass storage will always get the latest copy of the data. Valid words in the cache are invalidated when the channel is writing core to clear the cache of any valid entries that would conflict with the core copy. Channel requests are assigned the highest priority and are granted cache cycles as soon as the cache control becomes IDLE and core is not busy. If channel requests are backed up, the channels will also get the next core cycle.

3.3.7.1 **Channel Read** – After a channel is started (a channel is started by initializing the drive, setting up the channel command list, and issuing a Write command), the channel control initiates channel requests to read from core memory as long as the channel data buffer has enough empty locations to store the words. Requests to read from memory are also made by the channel control to fetch the CCWs which then control the transfer of data. Read requests for data are normally made for four words at a time. To read from core memory, the channel control sets up the request as follows:

a. Transfers the CCW address from the CCW BUF to the CCW register to present the PMA with the correct address (CCW CHA 14–35). CCL CHAN EPT is asserted only if the reference is to the EPT which is made to fetch the initial CCW.

b. Sets up CCW WD 0–3 REQ to specify the words in the quadword group that are needed and sets up CCL CH MB SEL 1-2 to select the MB from which the first word will be taken. Bits 34 and 35 of the channel address point to the first word in the quadword group that is to be read. CCL CHAN TO MEM will not be asserted when the channel issues a read request.

c. Asserts CCL CHAN REQ.

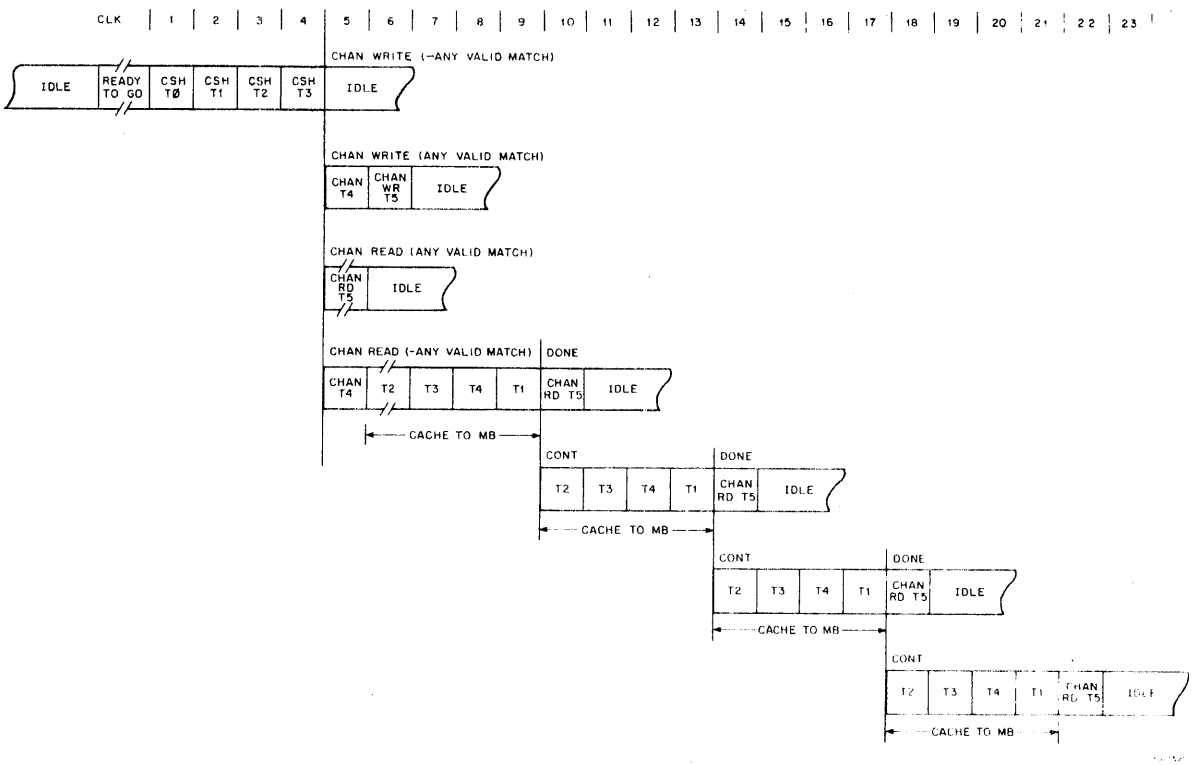


Figure 3-29 Cache Channel Cycle, Time State Bar Chart

The channel control must then wait until the cache control grants the request and the requested words come out of the cache and/or from core.

If the cache control is IDLE, or when the cache control enters its IDLE state and core is not busy, the cache control will grant the channel request and start a cache CHAN cycle to execute the read request. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. The state generator will always advance from IDLE to READY TO GO if the previous cycle was not a cache EBox cycle. The state generator will not advance to READY TO GO if the previous cycle was a cache EBox cycle and the EBox has not yet asserted CLK EBOX SYNC D. This condition is necessary to satisfy EBox read requests because the EBox will take the data only when CLK EBOX SYNC D is asserted. At READY TO GO, the CSH CHAN CYC latch is set and the PMA is set up to transfer the channel address (CCW CHA 14–35). From READY TO GO, the time state generator advances to the CSH time state branch to execute the cycle because a request other than an EBox request (ANY REQ) is granted. At CSH T3, the contents of the cache are checked to see if any valid words are in the cache (ANY VALID MATCH). If the cache does not contain any valid words, the cache control time state generator advances from CSH T3 to CHAN RD T5 to latch the SBus address and start a core read cycle for all requested words. However, if the cache contains some valid words, the state generator advances instead to CHAN T4 and the CACHE TO MB time states to set up the MB WR RQ and CTOMB WD RQ logic to move the valid words into the MBs. The CTOMB WD RQ logic supplies the word address (CACHE TO MB 34-35) for the cache block of interest and drives the MB HOLD IN logic to generate the appropriate MB load pulse at CACHE TO MB T3. At CACHE TO MB T4, the associated CTOMB WD RQ is cleared. The MB WR RQ logic is set up to remember which MBs received a word from the cache as the state generator steps through the CACHE TO MB time states. After all valid words are moved to the MBs, the state generator advances to CHAN RD T5 to latch the SBus address and start a core read cycle for those words that are not valid (RD NON VALID WDS). When the core read cycle is started, the cache control returns to IDLE.

While the channel and core controls are busy transferring the words, the cache control will only grant the EBox cache cycles to read or write the cache. If during this time the channel(s) makes another request, the channel will also get the next core cycle. The EBox can get a core cycle only when a channel request is not pending. As the words come in from core, they are moved into the appropriate MBs by the core control. As each word comes in, the core control sets the appropriate MB 0–3 WR RQ latch and clears the appropriate MB 0–3 HOLD IN signal for one clock tick to load the MB. The channel control will move the requested words from the MBs to the CH BUF, in ascending modulo 4 order. That is, if words 2 and 3 come from the cache and words 0 and 1 are coming from core, the channel control waits to take the words until word 0 is placed in the MB. Besides loading the MBs, the MB 0–3 HOLD IN signals inform the channel control that the corresponding word has been loaded into the associated MB. When the lowest numbered word of the requested group is placed in the MB, the channel control sets up CCL CH MB SEL 1-2 to select that MB and strobes the contents of that MB into the CH BUF (or the CCW BUF when the channel is fetching a CCW). This operation is repeated by the channel control until all requested words have been transferred.

As each word is taken by the channel control, the associated MB WR RQ is also cleared. Core will remain busy as long as an MB WR RQ is pending. This prevents another core cycle from being started until the channel control has taken all the words.

3.3.7.2 Channel Write – After a channel is started, the channel control initiates channel requests to write into memory as long as the channel has enough words in its data buffers. Requests to write into memory are also made by the channel control to store status information at the conclusion of a data transfer operation or in the event of an error. Write requests for data are normally made for four words at a time. To write core memory, the channel control sets up the request as follows:

a. Transfers the CCW address from the CCW BUF to the CCW register to present the PMA with the correct address (CCW CHA 14–35). CCL CHAN EPT is asserted only if the reference is to the EPT, which is made when storing the status.

b. Sets up CCW WD 0–3 REQ to specify the words in the quadword group that are to be written and sets up CCL CH MB SEL 1-2 to select the MB that will be loaded first. Bits 34 and 35 of the channel address point to the first word in the quadword group that will be written. CCL CHAN TO MEM will be asserted when the channel issues a write request.

c. Asserts CCL CHAN REQ

The channel control must then wait until the cache control grants the request.

If the cache control is IDLE, or when the cache control enters its IDLE state and core is not busy, the cache control will grant the channel request and start a cache channel cycle to execute the write request. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. The state generator will always advance from IDLE to READY TO GO if the previous cycle was not a cache EBox cycle. The state generator will not advance to READY TO GO if the previous cycle was a cache EBox cycle and the EBox has not asserted CLK EBOX SYNC D. This condition is necessary to satisfy EBox read requests because the EBox will take the data only when CLK EBOX SYNC D is asserted. At READY TO GO the CSH CHAN CYC latch is set, a core write cycle is started, and the PMA is set up to transfer the channel address (CCW CHA 14–35). The core write cycle is started by the channel control by asserting CCL MEM START and loading one word into an MB when it recognizes that the CSH CHAN CCYC latch is set. After starting the core write cycle, the channel control loads the remaining MBs at a rate of one word every four clock ticks by setting up the CCL CH MB SEL 1-2 lines to select the desired MB and asserting CCL CH LOAD MB.

At the same time the MBs are loaded by the channel control, the cache control state generator advances from READY TO GO to the CSH time state branch to execute the cache channel cycle because a request other than an EBox request (ANY REQ) is granted. The cache CHAN cycle is executed to set up the MB WR RQ queue and to clear the valid and written bits in the cache directory, if any valid entries are found in the cache. The MB WR RQ queue is set up at CSH T2 to remember which MBs the channel is loading so that the core control can place these same words on the SBus data lines during the core write cycle. As each word is written into core, the corresponding MB WR RQ is cleared by the SBus ACKN pulse. At CSH T3, the contents of the cache are checked to see if any valid words are in the cache (ANY VALID MATCH). If the cache does not contain any valid words, the cache control returns to IDLE. However, if the cache contains some valid words, the state generator advances instead to CHAN T4 and then to CHAN WR T5, to clear the valid and written bits. From CHAN WR T5, the cache control time state generator advances to IDLE. While the channel and core controls are busy transferring the words, the cache control will only grant the EBox cache cycle to read or write the cache. If during this time the channel(s) makes another request the channel will also get the next core cycle. The EBox can get a core cycle only when a channel request is not pending. As each word is written into core, the associated MB WR RQ is clearedd and the next MB pointed to by the MB WR RQ queue is selected. After the last word is written into core, another core cycle may be started.

NOTE

When reading magtape in the reversed direction, channel write operations are executed slightly differently; that is, memory is not started until all the words have been transferred to the MBs. This is done so that the words can be transferred to core in the correct order.

3.4 CACHE USE LOGIC

The cache use logic (Figure 3-30) keeps track of the order in which the four cache blocks of a given quadword line are used. Since there are 128 quadword lines, each containing four cache blocks (0, 1, 2, and 3) in the cache, 128 entries must be maintained to keep track of the order in which all cache blocks in the cache are used. Consequently, a use table containing 128 locations is employed by the use logic to maintain the use history of the cache.

The cache use logic consists of two RAMs and a set of mixers. One RAM contains the use information and is named the use table. The other RAM contains update information for the use table and is named the lookup table (Refill RAM). The use table contains 128 locations, one for each quadword line of the cache. The lookup table also contains 128 locations, but not for the same reason. The lookup table contains entries for all possible history combinations as a function of the four cache quarters, which turns out to be 128 entries. After the cache is initialized for full cache service, only 96 out of the 128 locations are required to provide the use history update information, because 32 combinations are illegal. Although 32 combinations are illegal after initialization, these combinations may be encountered during initialization and are therefore accounted for in the lookup table.

The use table is five bits wide and is structured into the following three fields:

a. MRU: Bits 0 and 1
b. ORDER: Bit 2
c. LRU: Bits 3 and 4

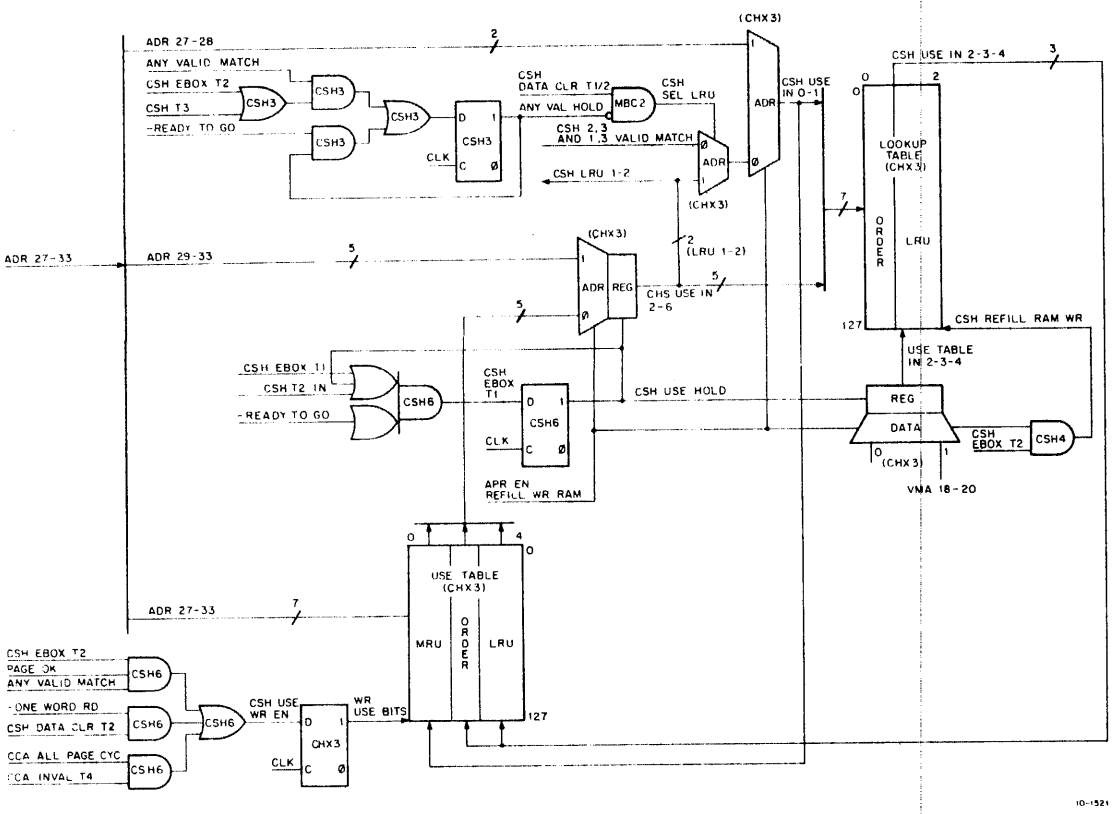


Figure 3-30 Cache Use Logic, Simplified Block Diagram

The Most Recently Used (MRU) field contains a 2-bit code to specify which cache was used most recently. The LRU field contains a 2-bit code to specify which cache was least recently used. The ORDER bit specifies the order of use of the other two cache blocks of a given line. A "zero" in this field indicates the order was ascending; a "one" indicates the order was descending. For example, if the use table contained the bit pattern "00011," it means that the order of use is 0, 1, 2, and 3.

The source of the MRU code is either a function of the cache directory that yielded a matched entry, or the contents of the LRU field of the use table if no match exists. The source of the ORDER and the LRU codes is the lookup table which is loaded at power up.

The lookup table is three bits wide and is structured into the following two fields:

a. ORDER: Bit 0
b. LRU: Bits 1 and 2

Collectively, the contents of the lookup table represent the refill algorithm of the cache. The refill algorithm can be adjusted by changing the sequence of the bit patterns to bypass one, two, or three cache quarters in any combination. Normally, the algorithm is set to use all four cache quarters equally. Table 3-10 specifies the bit patterns and the sequence of these patterns for using all cache quarters equally.

Table 3-10 Cache Refill Algorithm

Locations	Contents							
0-7	0	1	2	3	4	5	6	7
8-15	3	1	2	3	2	1	2	3
16-23	7	1	2	7	1	1	2	7
24-31	6	5	6	7	5	5	6	7
32-39	0	3	2	3	0	2	2	3
40-47	0	1	2	3	4	5	6	7
48-55	0	7	7	7	0	0	0	7
56-63	4	6	6	6	4	4	6	4
64-71	3	1	3	3	1	1	1	3
72-79	0	7	7	7	0	0	0	7
80-87	0	1	2	3	4	5	6	7
88-95	4	5	5	7	4	5	4	7
96-103	0	1	2	2	0	1	2	1
104-111	0	5	6	6	0	5	6	0
112-119	4	5	6	5	4	5	6	4
120-127	0	1	2	3	4	5	6	7

During normal operation the lookup table is addressed by the contents of the use table in conjunction with a 2-bit code that specifies the cache directory that yielded a match, or a 2-bit code that specifies the LRU cache block if no match occurred. When the lookup table is loaded (APR EN REFILL RAM WR), the table is addressed by address bits 27–33, which are the same as those used to address the cache directory. After the lookup table is loaded by the EBox at power up, a cache sweep instruction to invalidate the entire cache must be executed by the EBox to initialize the use table to purge all illegal bit patterns from the table. The illegal patterns for full cache service are those where the contents of the MRU field is the same as the contents of the LRU field of the use table. After the use table is initialized, it will contain "00 0 11" in every location, indicating that the order of use of each cache Line is 0 1 2 3.

3.4.1 Load Lookup Table (Refill RAM)

The lookup table is loaded by the EBox by executing the BLKO APR, E instruction. Each time this instruction is executed, one 3-bit word of data will be loaded into the lookup table. A total of 128 3-bit words of data must be loaded into the table.

Each time the BLKO APR instruction is executed the EBox sets up the request as follows:

a. Loads VMA bits 18–20 with the data to be written into the lookup table.
b. Loads VMA bits 27–33 with an address to select a location in the lookup table.
c. Asserts CLK EBOX REQ, APR EBOX READ REG, and APR EN REFILL RAM WR.

This sets up the conditions required for the MBox to service the EBox request to load one word into the lookup table. If the cache control is IDLE, or when the cache control enters its IDLE state and if a higher priority request is not pending (MB or CHAN request), the cache control will grant the EBox request and start a cache EBox cycle to execute the request. This decision is made as the cache control time state generator advances from IDLE to READY TO GO. At READY TO GO, the CSH EBOX CYC latch is set and the address is gated from the VMA to the lookup table via the lookup table address mixers. These mixers are set up by APR EN REFILL RAM WR to select the correct address. The APR EN REFILL RAM WR signal also sets up the data input mixer for the lookup table. From READY TO GO, the cache control time state generator advances to CSH EBOX T0, T1, and T2 in sequence. At CSH EBOX T2, the CSH USE HOLD flip-flop is set to hold the address and data, and CSH REFILL RAM WR is asserted to write the data into the addressed location of the lookup table. From CSH EBOX T2, the cache control returns to IDLE and asserts MBOX RESP IN.

3.4.2 Initialize Cache Directory and Use Table

The cache directory and the use table are initialized at the same time by the cache clearer control after the cache clearer is started by the EBox. In the cache directory, all valid and written bits are cleared; in the use table, all entries are initialized to reflect the refill algorithm for full cache service. This means that the use table is initialized to specify a use order of 0, 1, 2, 3.

NOTE

The use table can also be initialized to provide partial cache service where one, two, or three cache quarters are bypassed.

The cache clearer control is started by the EBox by executing a Sweep instruction (refer to Cache Clearer Control Description) to invalidate all pages in the cache.

If the cache clearer control is set up to invalidate the cache for all pages, then the CCA INVAL T4 time state is entered when a cache CCA cycle is executed by the cache control. One clock tick after CCA INVAL T4, the use table data is written. The cache directory VALID and WRITTEN bits are also cleared at this time. During the cache CCA cycle, the address for the use table is obtained from the CCA register via the PMA; the address for the lookup table is the contents of the addressed location of the use table concatenated with a 2-bit code that indicates which cache is selected (valid match is forced) for the current cache CCA cycle. The data for the use table is the contents of the addressed location of the lookup table concatenated with the 2-bit code that indicates which cache is selected (valid match is forced) for the current cache CCA cycle. This arrangement of address and data selection will cause the following to occur, one clock tick after CCA INVAL T4:

1. One of four quadrants, depending on which cache is selected (valid match is forced), is pointed to while one of 32 locations in that quadrant is addressed by the five bits contained in the addressed location of the use table.

2. The contents of the addressed location in the lookup table is written into the ORDER and LRU fields of the addressed location of the use table.

3. The MRU field of the use table will receive the code that indicates which cache is currently selected by forcing a valid match.

Consequently, the MRU field of the use table of a given addressed location will be set to the cache code for which a match is currently forced. Since the cache clearer address register is counted down from 777 to 000 in increments of one, each location will wind up with a code of 00 in the MRU field and the ORDER and LRU field will wind up set with the contents of the Lookup Table location that is being addressed at this time which should be 011.

3.4.3 Normal Operation

The use table is updated during a cache EBox cycle that is executing an EBox read or write request for which the cache is to be used. If the cache contains a valid entry (ANY VALID MATCH), even though the desired word may not be in the cache (-RD FOUND), the use table is updated by asserting WR USE BITS one clock tick after CSH EBOX T2. If the cache does not contain a valid entry (-ANY VALID MATCH), the use table is updated by asserting WR USE BITS one clock tick after CSH DATA CLR T2. The major difference between these two cases, besides the timing, is the way the lookup table is addressed and the data for the MRU field of the addressed use table location is derived. For the case where a valid entry is found, ANY VALID MATCH is asserted, which causes the ANY VAL HOLD latch to be set one clock after CSH EBOX T2. This inhibits the CSH SEL LRU gate to make sure that the two high-order bits of the lookup table address and the data for the MRU field of the use table is a two-bit code that identifies the cache that yielded the valid entry. This condition satisfies the case where the desired word was not in the cache and the CSH DATA CLR time state are entered to clear the data and update the use table. For the case where a valid entry is not found in the cache, ANY VALID MATCH is not asserted, which causes ANY VAL HOLD to remain cleared after CSH EBOX T2. This enables the CSH SEL LRU gate at CSH DATA CLR T1 to make sure that the two high-order bits of the lookup table address and the data for the MRU field of the use table is a two-bit code that identifies the LRU cache.

Figure 3-31 illustrates the current state and the next state of the use table as a function of the selected cache. The selected cache may be the one that yielded a valid entry or the LRU cache. By using this table, one may determine what the next state of a given use table location should be.

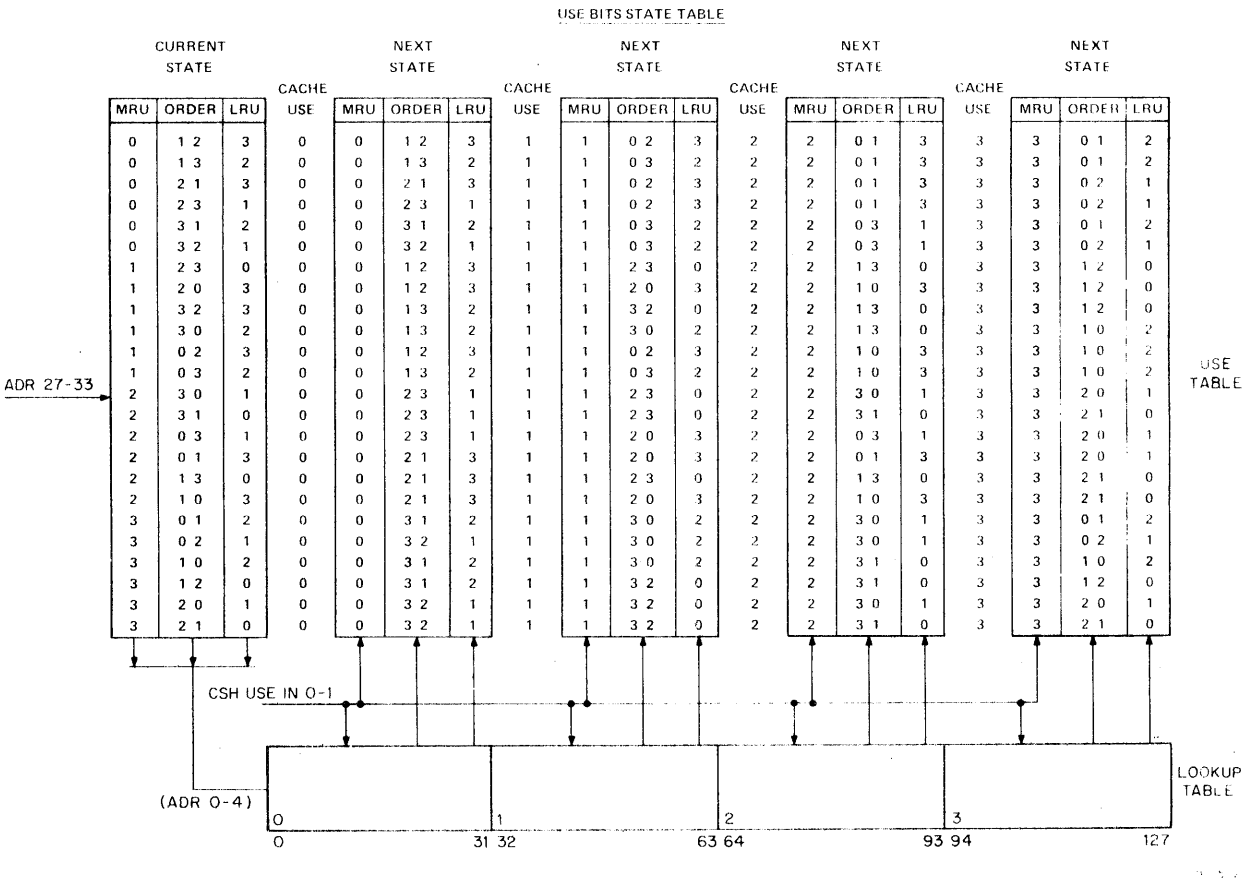


Figure 3-31 Cache Use History Update Functions

3.5 CACHE CLEARER CONTROL

The cache clearer control (Figure 3-32) requests cache cycles to invalidate the cache and/or validate core after it is set up by the EBox. When setting up the cache clearer control, the EBox specifies which of these operations are to be executed and whether the operations are to be done for only one page or the entire cache. The cache clearer control consists of two binary counters (LINE and BLOCK) for generating the cache address and a number of control flip-flops. Collectively, the LINE and BLOCK counters are referred to as the CCA register. The cache clearer is set up by the EBox when it executes a sweep instruction. This instruction causes the EBox to issue a request to load the CCA register and three control latches that specify what type of cache clear operation is to be performed (refer to EBox load register and cache CCA cycle descriptions). Before the EBox issues the request to load the CCA register, it must set up VMA bits 14–26 and IR AC bits 10–12 correctly. VMA bits 14–26 specify the page for which the cache sweep operation is to be performed. When the entire cache is to be swept, the VMA does not have to be set up. IR AC bit 10–12 specifies the type of sweep operation that is to be performed. These bits are interpreted by the cache clearer control as follows:

IR AC10: CSH CCA ONE PAGE
IR AC11: CSH CCA VAL CORE
IR AC12: CSH CCA INVAL CACHE

When the EBox Request is granted by the cache control, a cache EBox cycle is executed and the CCA register and the control latches are loaded at CSH EBOX T1. The CCA register receives VMA bits 14–26 and the control latches receive IR AC bits 10–12. The CCA register is loaded by CCA LOAD and the control latches are loaded by CCA SEL 1. Both these signals are true for only one clock period when CSH EBOX T1 is asserted. At the same time the CCA register and the control latches are loaded, the CCA REQ latch is set and the line and block counters are loaded with a count of 777 (all nine counter bits are set to "one"). The line and block counters are loaded because both CCA SEL 1 and 2 are "zero" at CSH EBOX T1. On the next clock tick both CCA SEL 1 and 2 return to "one" to hold the counter contents and the MBox asserts MBOX RESP IN to inform the EBox that the cache clearer control is set up. The EBox will then continue with executing the program. The cache clearer control will remain in this initialized state until the cache control grants a cache cycle to the cache clearer control. If no other requests are pending, the CCA request is granted and a cache CCA cycle is executed (refer to cache CCA cycle description). While the cache CCA cycle is executed, one of several different operations may be performed, depending on what type of cache sweep operation was requested and what is found in the cache directory.

At the end of a cycle, the CCA CYC DONE flip-flop is set to decrement the line or block counter and to check if the counter generated a carry (CCA CRY OUT). The counter generates a carry when the cache clearer has finished scanning the entire cache. The block counter is decremented to select each block of a given line when sweeping the entire cache. The carry from the block counter decrements the line counter. The line counter is decremented to select each line when sweeping the cache for a given page. The counters are decremented because CCA SEL 2 is forced to zero and CCA SEL 1 remains in the one state when the CCA CYC DONE flip-flop is set. This flip-flop remains set for one clock period. When the counter overflows, CCA CRY OUT is asserted causing the CCA REQ flip-flop to be cleared when CCA CYC DONE is set. This informs the EBox that the cache clearer has completed the cache sweep operation.

3.6 MB CONTROL

The MB control (Figures 3-33 through 3-35) moves data in and out of the MBs in response to gating function from the cache control, core control, or the channel control. Two request queues are employed to facilitate moving data in and out of the MBs in an orderly fashion.

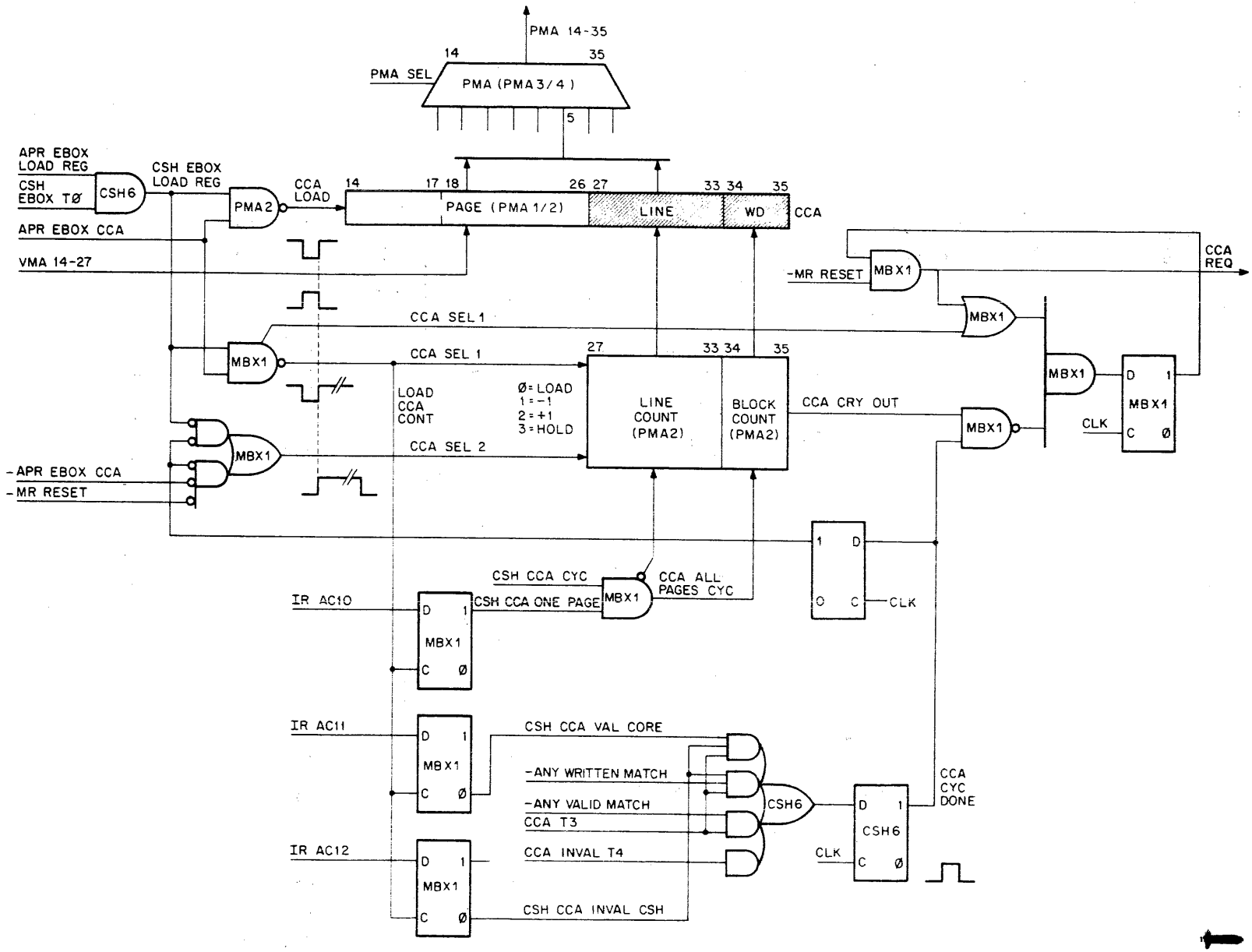


Figure 3-32 Cache Clearer Control, Simplified Logic Diagram

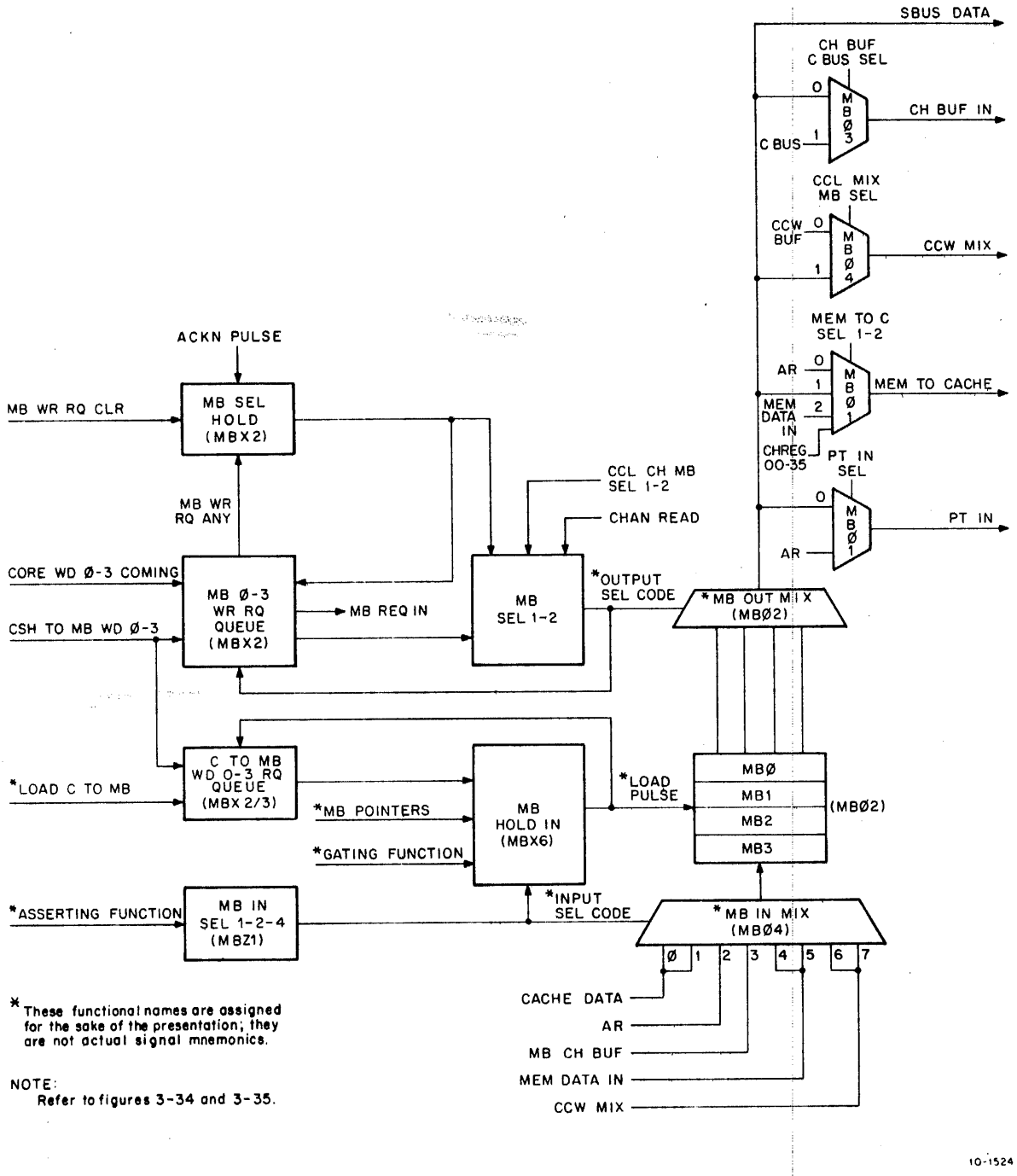


Figure 3-33 MB Control, Functional Block Diagram

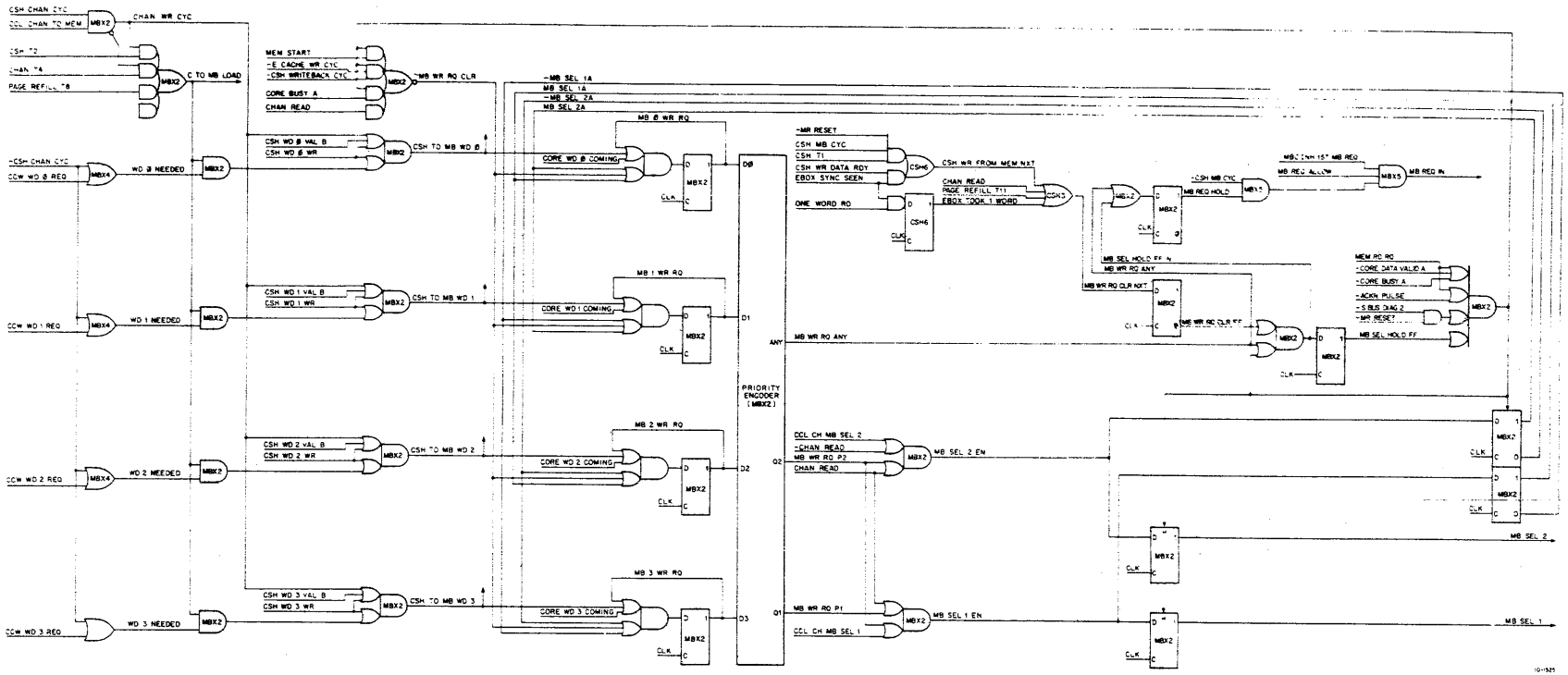


Figure 3-34 MB WR RQ Queue and MB SEL Logic, Simplified Logic Diagram

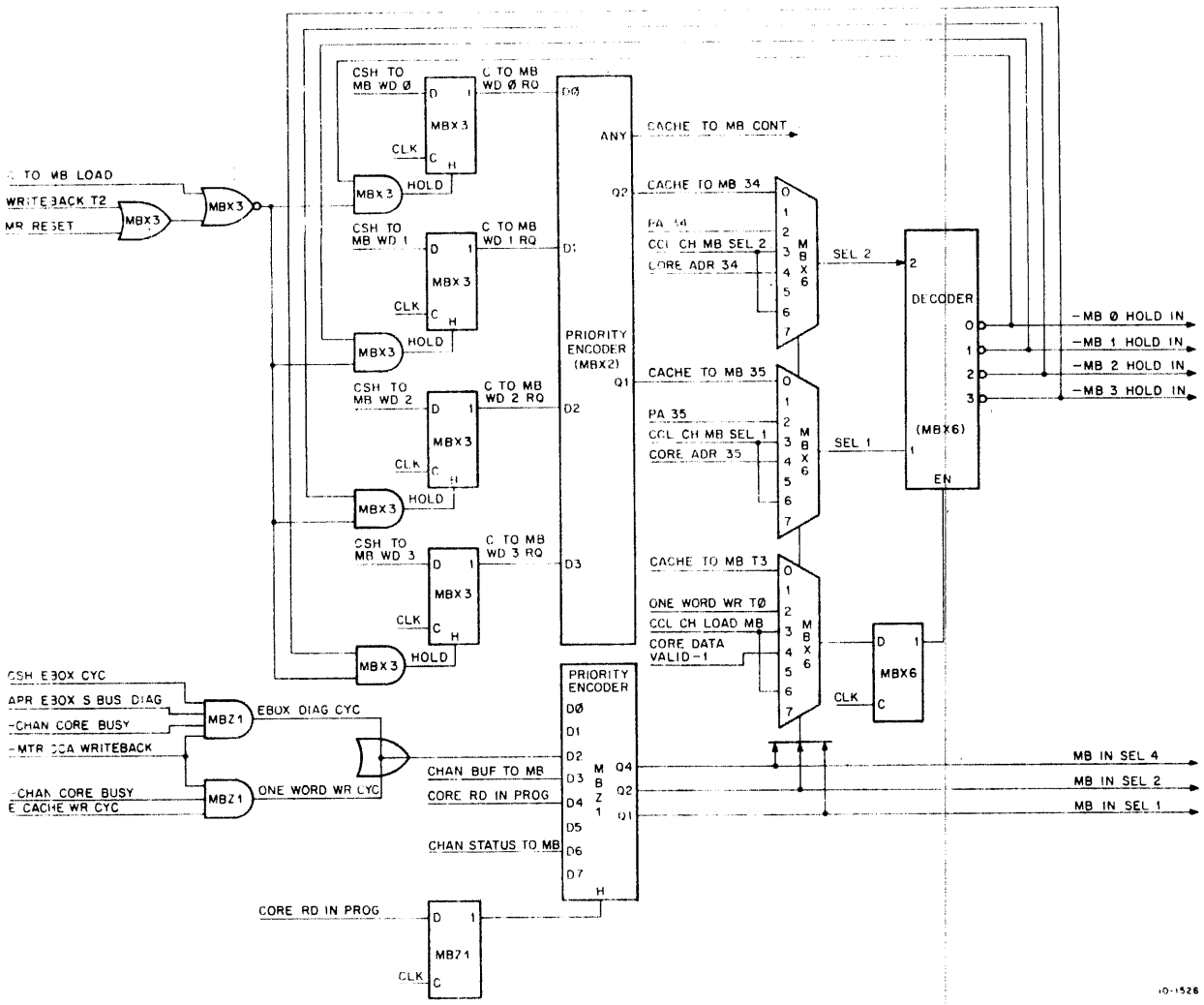


Figure 3-35 CTOMB WD RQ Queue, Load Pulse Generator, and MB IN Selector Simplified Logic Diagram

3.6.1 MB 0–3 WR RQ Queue

The MB 0–3 WR RQ queue is loaded to remember which MB received data so that the data can be moved to the desired destination in the most expedient manner. As the contents of an MB are transferred, the associated MB 0–3 WR RQ in the queue is cleared.

The MBs may receive data from the following sources:

a. AR – While a cache EBox cycle is executed to write one word.

b. CACHE – During a cache page refill cycle, a cache writeback cycle, or during a cache channel cycle that is executing a read request.

c. SBUS – During a core read cycle that was initiated by a cache page refill cycle, a cache EBox cycle, or a cache channel cycle.

d. CH BUF or CCW BUF – Words from the CH BUF or the CCW BUF are moved into the MBs by the channel control independently.

The MB input selector and load pulse generator control data selection and loading of the MBs.

Data that has been loaded into an MB can be transferred to the following destination:

a. AR – While a cache EBox cycle is executed to read one word or the words that are non-valid in the cache from core. Only the first word is transferred to the AR; the remaining words are transferred only to the cache if it is not a "ONE WORD RD".

b. Cache – While a cache EBox cycle is executed to read one or more words from core. Only the first word is moved into the cache during the cache EBox cycle. The remaining words are moved into the cache by the cache MB cycle.

c. SBUS – During a core write cycle that has been initiated by a cache writeback cycle, a cache EBox cycle, or a cache channel cycle.

d. CH BUF or CCW BUS – Words are moved from the MBs to the CH BUF or the CCW BUF by the channel control.

e. PT – During a cache page refill cycle.

The MB OUTPUT selector and a number of mixers control the output transfer of the data. The MB 0–3 WR RQ queue is loaded whenever the MBs are loaded. The queue is loaded to:

a. Remember which MB received the word from the AR for a one-word write or an SBUS DIAG operation (refer to cache EBox cycle description).

b. Remember which MBs will receive the valid or written word from the cache (CSH TO MB WD 0–3). These words will be moved into the MBs by the CSH TO MB time states. At the same time the MB 0–3 WR RQ queue is loaded, the CTOMB WD 0–3 RQ queue is also loaded to provide the correct cache address (CACHE TO MB 34–35) and MB load pulse (–MB 0–3 HOLD). Written words are moved into the MBs for writeback operations to make room in the cache, or to validate core. Valid words are moved into the MBs for page refill or channel read operations (refer to cache writeback, cache page refill, and cache channel cycle descriptions).

c. Remember which MB received a word from core via the SBus during a core read cycle (refer to cache EBox, cache MB, and cache page refill cycle descriptions).

d. Remember which MB received a word from a channel (refer to cache channel cycle description).

The appropriate request stored in the MB 0–3 WR RQ queue is cleared whenever the contents of an MB are transferred to the desired destination. The appropriate MB 0–3 WR RQ is cleared when:

a. The first word that comes in from core during a core read cycle is taken by the EBox (moved into the AR). Refer to cache EBox cycle description.

b. A word from the MB is written into the cache (refer to cache EBox and cache MB cycle descriptions).

c. Core has accepted a word (by asserting SBUS ACKN) during a core write cycle (refer to cache EBox and cache writeback cycle descriptions).

d. The channel control selects an MB to read its contents (refer to cache channel cycle description).

e. A word from the MB is written into the page table (refer to cache page refill cycle description).

Except during channel read operations, each time an MB 0–3 WR RQ is cleared, the next highest priority (ascending modulo 4) MB 0–3 WR RQ in the queue causes the corresponding MB to be selected to get ready for the next transfer. Core is freed (–CORE BUSY) only after all MB 0–3 WR RQs are cleared (–MB WR RQ ANY).

3.6.2 MB Input Selector and Load Pulse Generator

The source for loading the MBs is selected by the MB IN mixer. This mixer is controlled by the MB IN SEL 1-2-4 control code which is generated by a priority encoder. Besides selecting the desired data source for the MBs, this control code is also used to select the appropriate logic for generating the MB load pulses (–MB 0–3 HOLD IN). The association between the MB SEL 1-2-4 code, the MB IN mixer data connections, and the functions that assert a particular code and thereby the desired data source is defined in Table 3-11.

Table 3-11 MB Input Functions

ASSERTING FUNCTION	MB IN SEL 1-2-4 CODE (MIXER INPUT) 1 2 4	DATA SOURCE
WRITEBACK-PAGE REFILL-CHAN READ	X 0 0	CACHE DATA
ONE WORD WRITE (–CACHE) OR SBUS DIAG CYC	0 1 0	AR
CHAN WRITE DATA	1 1 0	MB CH BUF
CORE READ	X 0 1	MEM DATA IN
CHAN WRITE STATUS	X 1 1	CCW MIX

X = ARBITRARY

Besides controlling the MB IN mixer, the MB SEL 1-2-4 code also selects the desired logic via a set of mixers for producing the desired MB load pulse (–MB 0–3 HOLD IN) at the correct time. The association between the MB SEL 1-2-4 code, the signals that specify which MB is currently to be loaded (MB Pointer), and the gating function that generates the pulse is defined in Table 3-12.

Table 3-12 MB Load Functions

MB IN SEL 1-2-4 CODE 1 2 4	MB POINTER	GATING FUNCTION
X 0 0	CACHE TO MB 34-35	CACHE TO MB T3
0 1 0	PMA 34-35	ONE WORD WR T0
1 1 0	CCL CH MB SEL 1-2	CCL CH LOAD MB
X 0 1	CORE ADR 34-35	CORE DATA VALID -1
X 1 1	CCL CH MB SEL 1-2	CCL CH LOAD MB

X = ARBITRARY

3.6.3 CTOMB WD 0-3 RQ QUEUE

The CTOMB WD 0-3 RQ queue is loaded when data is to be transferred from the cache to the MB during a writeback, page refill, or a channel read operation. This queue is loaded to remember which words are to be transferred. The queue also serves as a source for generating the correct cache address (CACHE TO MB 34-35) and MB load pulse (-MB 0-3 HOLD). As each valid or written word is moved into the appropriate MB during the CACHE TO MB time states, the associated CTOMB WD 0-3 RQ in the queue is cleared. Clearing a CTOMB WD 0-3 RQ causes the next highest priority CTOMB WD 0-3 RQ to generate another address and MB load pulse. When the state generator advances through the CACHE TO MB time states again. This operation will continue until all CTOMB WD 0-3 RQs in the queue are cleared.

3.6.4 MB Output Selector

After an MB is loaded, the contents of that MB may be transferred to the desired destination by setting up one or more mixers.

a. The MB OUT mixer selects the desired MB in response to the appropriate MB SEL 1-2 code. The MB SEL 1-2 code is a function of the contents of the MB WR RQ queue, unless the channel is executing a read request, in which case the channel control selects the MB it needs to read. The selected data is then distributed to the SBus, the CH BUF IN mixer, the MEM TO C mixer, the CCW mixer, and the PT IN mixer.

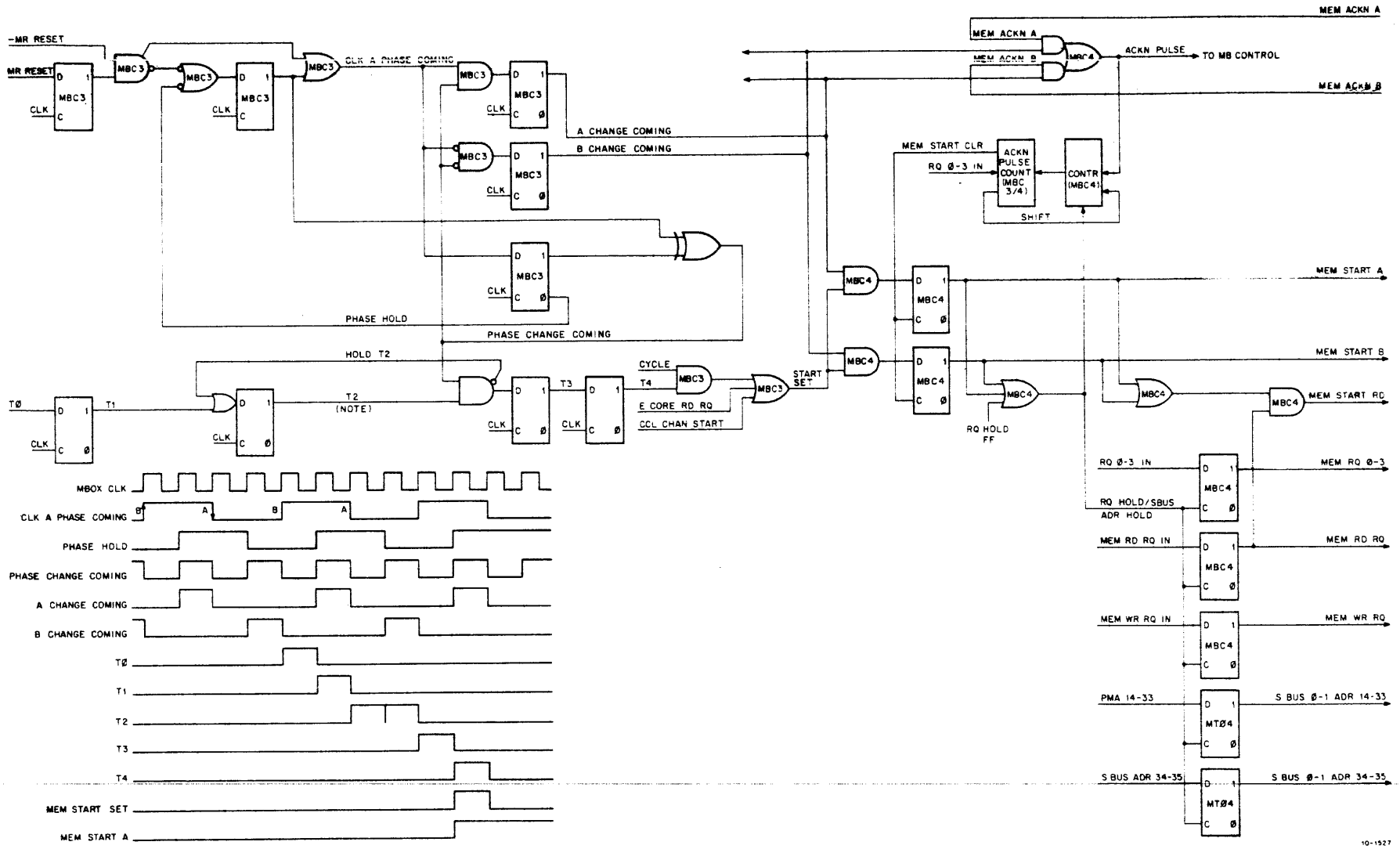
b. The CH BUF IN and CCW mixers are controlled by the channel control when data is transferred from the MB to the channel.

c. The MEM TO C mixer is controlled by the cache control to direct the MB data to the cache during a core read cycle to refill the cache.

d. The PT IN mixer is controlled by the cache control to direct the MB data to the page table during a core read cycle to refill the page table.

3.7 CORE CONTROL

The core control starts and executes core read and write cycles in respponse to requests from the cache and channel controls. Since either control may request to read or write up to four words, the core control must keep track of which word has been transferred. To this end the core control employs counters to keep track of the ACKN pulses and CORE DATA VALID pulses (Figures 3-36 and 3-37).



NOTE
Time state chain TO–T4 is idealized and does not represent actual logic.

Figure 3-36 Memory Start Control and Acknowledge Pulse Counter, Simplified Block Diagram

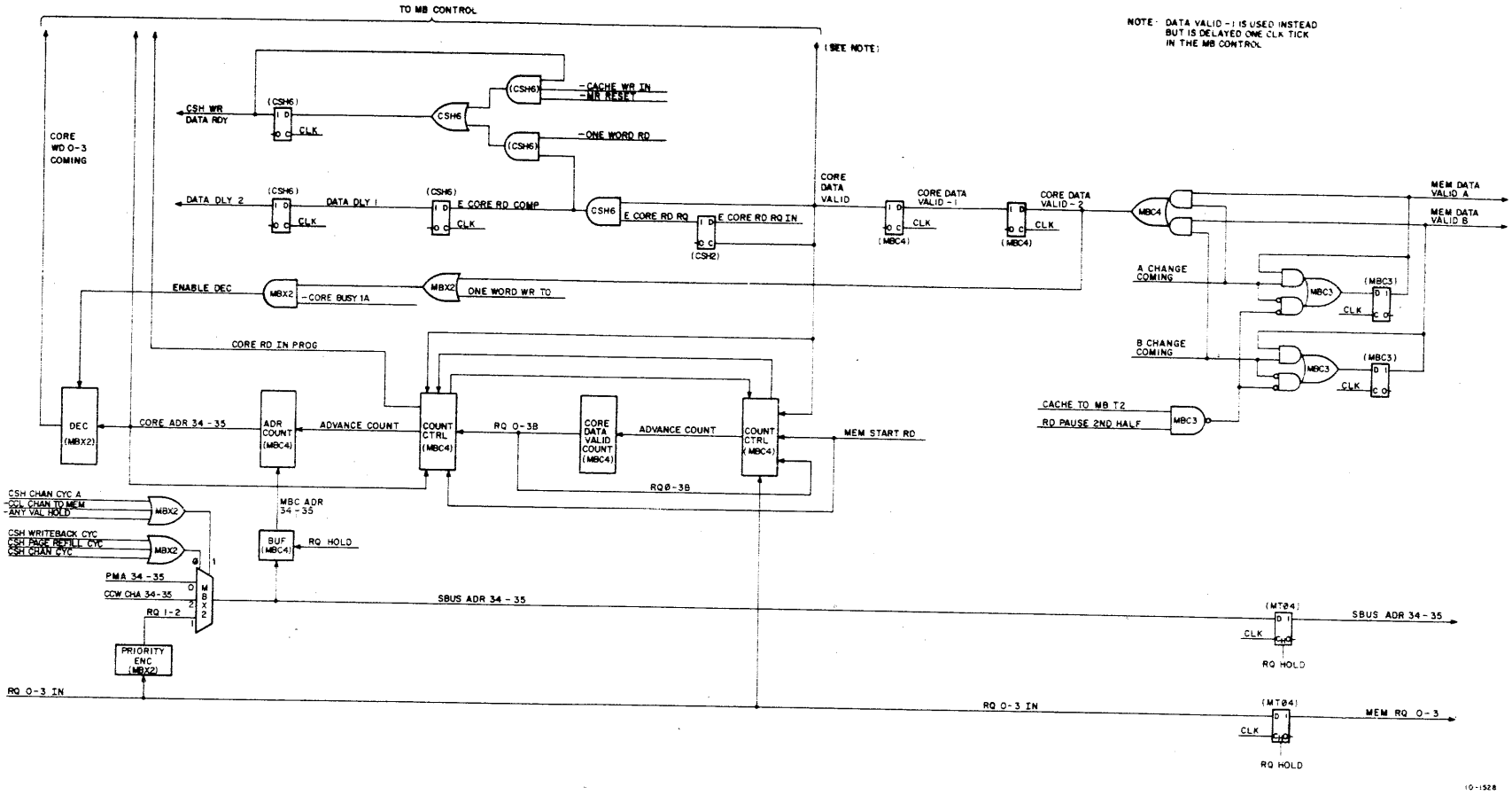


Figure 3-37 Core Data Valid Pulse Counter, Simplified Logic Diagram

The SBus control dialogue to start and execute a core cycle is synchronized with the SBus clock to minimize bus latency. Because the propagation time of the SBus control signals is less than the SBus clock period (four MBox clock periods), a control signal generated at one end of the SBus can be sensed at the other end without the need for synchronization logic. This speeds up the control bus operation. To further speed the operation of the SBus, two sets of SBus control signals are used. One set is synchronized on the A phase and the other on the B phase of the SBus clock.

A core cycle is started by asserting SBus drive signals in the following manner:

a. MEM START A or B is asserted and held. MEM START A is asserted if phase A of the SBus clock is coming when the core cycle is ready to be started. However, if phase B is coming when the core cycle is ready to be started, MEM START B is asserted instead.

b. MEM RD RQ or MEM WR RQ is asserted and held, depending on whether a read or a write cycle is to be initiated. Both MEM RD RQ and MEM WR RQ are asserted to execute a read-pause-write cycle.

c MEM RQ 0–3 are asserted and held to specify which words (and how many) are to be transferred.

d. MEM ADR 14–35 is held to address core. Bits 34 and 35 point to the word to be transferred first. Bits 14–33 point to the quadword (page and line).

The bus drive signals mentioned above are held by the MBox core control and are transferred to the SBus as long as MEM START A/B is set. Core memory responds to SBUS START A/B by asserting SBUS ACKN A/B during core read and core write cycles as each word is addressed if the address is valid and no address parity error is sensed. Core memory effectively addresses each requested word, in ascending modulo 4 sequence, starting with the first word requested. After each requested word (MEM RQ 0–3) is acknowledged with an SBUS ACKN pulse, MEM START A/B is cleared, allowing core memory to terminate its cycle after placing the last word on the SBus data lines. The acknowledge pulses are counted by the acknowledge pulse counter. If the number of acknowledge pulses do not correspond to the number of words requested, MEM START A/B is not cleared and the NXM Error flag is set. Reception of the acknowledge pulses also influences the operation of the MB control during a core write operation to transfer the contents of the appropriate MB to the SBus data lines.

During core read operations, core memory asserts SBUS DATA VALID A/B as each word is placed on the SBus data lines. The MBox core control waits two MBox clock ticks after receiving SBUS DATA VALID for the data bus drivers to stabilize before loading the data into the appropriate MB. Another core cycle cannot be started (core remains busy) until each requested word (MEM RQ 0–3) is received and moved out of the MB into the cache or the channel data buffer. DATA VALID pulses are counted by the core data valid counter which drives the MB control to load the appropriate MB and the MB 0–3 WR RQ queue. Loading the MB 0–3 WR RQ queue causes an MB request to be initiated to move the word from the MB to the cache. The contents of MB 0–3 WR RQ queue select the MB and address the cache (refer to MB control description).

3.7.1 SBus Dialogue Synchronization

The SBus uses two sets of START, ACKN, and DATA VALID lines. One set is synchronized with the A phase and the other with the B phase of the SBus clock. The period of the SBus clock is four MBox clock periods. By synchronizing one set of bus dialogue signals on phase A (trailing edge) and the other on phase B (leading edge) these signals can be placed on the bus two MBox clock periods earlier than would otherwise be possible. This, therefore, reduces the bus latency by two MBox clock periods.

Since MEM START A or B, depending on which phase is coming, can be asserted when the cache control time state generator enters one of several time states, a need exists for holding the time state for one tick to wait for A or B phase coming. Otherwise, the state generator may miss A and B phase coming.

3.7.2 Acknowledge Pulse Counter (MBC4)

When memory is started (MEM START A/B is asserted) to initiate either a core read or write cycle, the Acknowledge Pulse counter (RQ 0–3A), which is a shift register, is loaded with a bit pattern (RQ 0–3 IN) that specifies which words (and how many) are to be transferred. This is the same bit pattern that is transferred to core memory via the SBus RQ 0–3 lines. The counter then initializes itself to shift out all leading "zeros" if any (Table 3-13). When the first acknowledge pulse comes in, the leading "one" is shifted out of the counter. After each acknowledge pulse shifts out the corresponding word request (RQ 0–3A), the counter again shifts out any leading "zeros" to position the next word request in the most significant position so that it can also be shifted out when the next acknowledge pulse arrives. This is repeated until all requests (RQ 0–3A) are shifted out of the counter. At the same time the last request is shifted out MEM START A/B is cleared.

Table 3-13 is the Acknowledge Pulse counter initialization (shift RQ 0–3A until RQ 0A = 1) Truth table.

Table 3-13 Acknowledge Pulse Counter Initialization Truth Table

RQ 0-3 IN				RQ 0-3 A			
0	1	2	3	0	1	2	3
1	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0
1	1	0	0	1	1	0	0
0	0	1	0	1	0	0	0
1	0	1	0	1	0	1	0
0	1	1	0	1	1	0	0
1	1	1	0	1	1	1	0
0	0	0	1	1	0	0	0
1	0	0	1	1	0	0	1
0	1	0	1	1	0	1	0
1	1	0	1	1	1	0	1
0	0	1	1	1	1	0	0
1	0	1	1	1	0	1	1
0	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1

3.7.3 Data Valid Pulse Counter

When memory is started (MEM START A/B is asserted) to initiate a core read cycle, the Data Valid Pulse counter (RQ 0–3B) is loaded with a bit pattern (RQ 0–3 IN) that specifies which words (and how many) are to be transferred and the Core Address Counter (CORE ADR 34–35) is loaded with bits 34 and 35 of the MEM ADR, which specifies the word that will be received first. The Data Valid Pulse counter is a shift register and the Core Address counter is binary up/down counter. RQ 0–3 IN and MEM ADR 34–35 are also latched and transferred to core memory via the SBUS RQ 0–3 lines and SBus ADR 34–35 lines, respectively, to specify which words are to be read and the order in which they are to be transferred. RQ 0–3 IN is derived from RD NON-VALID WDS or RD PMA SINGLE when the core read cycle is initiated. MEM ADR 34–35 is derived from PMA 34–35 or from RQ 0–3 IN, depending on which cache cycle initiated the core cycle. When the EBox requests a word and the word is not in the cache, or the cache is not to be used, the cache EBox cycle initiates a core read cycle to read the words that are not valid in the cache or read a single word. For this case, MEM ADR 34–35 is produced from PMA 34–35, which points to the word the EBox requested. If, however, a cache page refill or a cache CHAN read cycle initiates the core read cycle, then MEM ADR 34–35 is derived from RQ 0–3 IN to generate an address that points to the first word, in ascending modulo 4 order, that is to be transferred. This is illustrated in Table 3-14.

Table 3-14 MEM ADR 34-35 Derivation Truth Table for Page Refill and Channel Read Cache Cycles

RQ 0-3 IN 0	1	2	3	MEM ADR 34-35
1	X	X	X	00
0	1	X	X	01
0	0	1	X	10
0	0	0	1	11

X = ARBITRARY

After the Data Valid and Core Address counters are loaded, they initialize themselves automatically. The counters initialize themselves as follows:

a. Core Address counter decrements until its content is 0.

b. The contents of the Data Valid counter are rotated to the left one position for each time the Core Address counter decremented, for example:

RQ 0-3 IN 0 1 2 3	CORE ADR 34 35	RQ 0-3 B 0 1 2 3	CORE ADR 34 35
1 1 0 0	01	1 0 0 1	00

Table 3-15 illustrates how RQ 0–3B is rotated and CORE ADR is decremented until it is zero for the Core Data Valid counter initialization (INT COMP) operation for page refill and channel read cycle.

Table 3-15 Core Data Valid Counter Initialization Truth Table

RQ 0-3 IN				CORE ADR		RQ 0-3 B				CORE ADR	
0	1	2	3	34	35	0	1	2	3	34	35
1	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	1	1	0	0	0	0	0
1	1	0	0	0	0	1	1	0	0	0	0
1	1	0	0	0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	1	0	1	0	1	0	0	0
0	1	1	0	0	1	1	1	0	0	0	0
0	1	1	0	1	0	1	0	0	1	0	0
1	1	1	0	0	0	1	1	1	0	0	0
1	1	1	0	0	1	1	1	0	1	0	0
1	1	1	0	1	0	1	0	1	1	0	0
0	0	0	1	1	1	1	0	0	0	0	0
1	0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	1	1	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1	0	0	0
0	1	0	1	1	1	1	0	1	0	0	0
1	1	0	1	0	0	1	1	0	1	0	0
1	1	0	1	0	1	1	0	1	1	0	0
1	1	0	1	1	1	1	1	1	0	0	0
0	0	1	1	1	0	1	1	0	0	0	0
0	0	1	1	1	1	1	0	0	1	0	0
1	0	1	1	0	0	1	0	1	1	0	0
1	0	1	1	1	0	1	1	1	0	0	0
1	0	1	1	1	1	1	1	0	1	0	0
0	1	1	1	0	1	1	1	1	0	0	0
0	1	1	1	1	0	1	1	0	1	0	0
0	1	1	1	1	1	1	0	1	1	0	0
1	1	1	1	0	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	1	1	0	0
1	1	1	1	1	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	0	0

When the Core Address counter reaches zero, the initialization phase is complete (INIT COMP) and CORE RD IN PROG is set. At this time, MEM ADR 34–35 is loaded into the Core Address counter again. The operation modes of these counters are also changed at this time to count the SBUS DATA VALID pulses. The Core Address counter is set up to increment rather than decrement the core address and the Data Valid counter is set up to shift instead of rotate left every time an SBUS DATA VALID pulse is received. Leading zeros are automatically shifted out during this operation. When the last request is in the RQ 0B position of the counter and the last core data valid pulse has come in, CORE RD IN PROG is cleared. However, core remains busy until all the words have been moved from the MBs to the Cache, page table, or channel, as the case may be.

After the SBUS DATA VALID pulse is received, the MBox core control waits two clock ticks before it triggers the MB control to transfer the data from the SBus data lines into the appropriate MB. This delay is provided by the two CORE DATA VALID time state flip-flops. The correct MB is selected by CORE ADR 34–35 from the Core Address counter and the load pulse is generated at CORE DATA VALID time. At this same time, the contents of the Core Data Valid counter are shifted left to shift out the request and the Core Address counter is incremented to select the next MB. Any leading zeros that may develop are automatically shifted out. Each time a leading zero is shifted out, the Core Address counter is also incremented to point to the next MB.

The MB 0–3 WR RQ queue in the MB control is set by CORE WD 0–3 COMING at CORE DATA VALID –2 to remember which MB was loaded. From the MB, the data may be moved to the cache (sometimes the AR) to the page table or the channel, depending on which cache cycle initiated the core read cycle.

Under certain conditions the core control operates slightly different for transferring the first word than it does for transferring subsequent words. This difference lies in the fact that the first word may have to be transferred to the AR in the EBox if the core cycle was initiated by a cache EBox cycle. To satisfy this requirement, the core control employs the following three special time state flip-flops: CSH WR DATA RDY, DATA DLY 1, and DATA DLY 2. When the first word comes in, these time states are triggered by the CORE DATA VALID state. They will not be triggered as subsequent words come in because E CORE RD RQ is cleared after the first word is transferred to the AR. Primarily, the purpose of these time states is to ensure that the EBox takes the data. These time states also enable the parity check logic for the first word. Normally, the EBox will take the data (when CLK EBOX SYNC D is asserted) directly from the SBus via the MEM TO C mixer. However, if the EBox does not take the data when it arrives, the EBox must then take the data from the MB into which the data is moved in any case. In either case, the data is moved into the cache only when the data is taken by the EBox. All subsequent words will be moved into the cache by initiating a cache MB cycle.

3.8 CHANNEL CONTROL

The channel logic in the MBox continuously scans the RH20 Massbus controllers. A given controller is selected when its select line (0–7) is asserted. After being selected, the controller can issue control/data requests. Since the MBox channel logic will remember which controller was selected at a given time, it can identify and process requests from all eight channels.

3.8.1 Timing Logic

The timing logic (Figures 3-38 and 3-39) generates the basic timing signals for the MBox channel control logic and the CBus select signals for enabling the RH20 Massbus controllers in a specific repetitive sequence. The timing logic includes a state generator, a scanner, a shift register, and a decoder.

The state generator, which is formed by a shift register with some control and feedback connections, generates the basic time states (T0–T3) for the channel control. Upon initialization (MR RESET is asserted), the state generator is synchronized with the EBox clock (CLK A PHASE COMING). After the initialization sequence, the state generator continues generating the T0–T3 time states in synchronism with the MBox clock.

The scanner, which is formed by an arithmetic logic unit (ALU), a 1-bit shift register, and some Exclusive OR and AND function feedback logic, generates the count sequence defined in scanner count sequence truth table shown in Figure 3-38.

Both the ALU and the shift register are cleared when the machine is initialized (MR RESET is asserted). After initialization (MR RESET is negated) and after the clock is started, the scanner advances from zero and continues through its prescribed sequence.

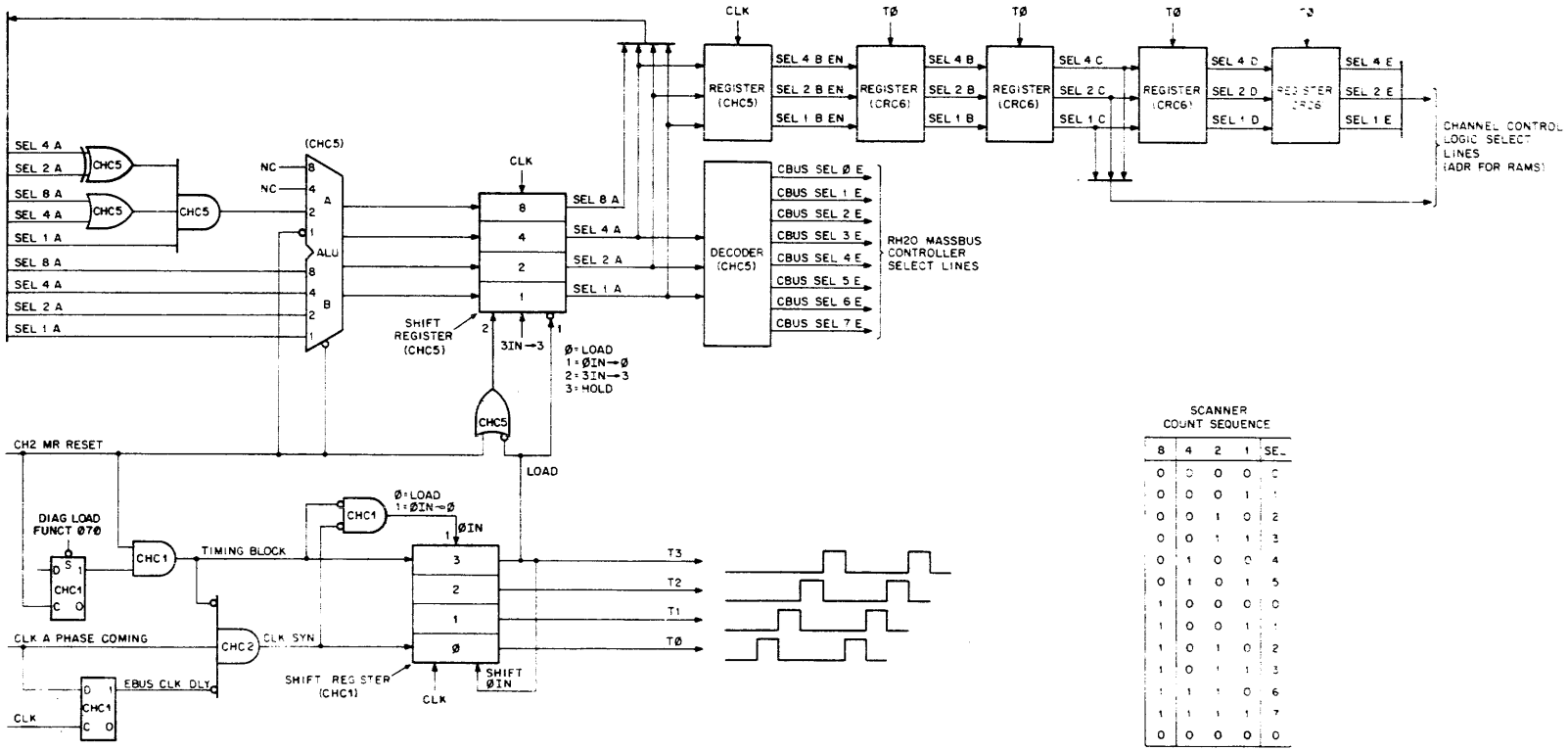


Figure 3-38 Timing Logic, Simplified Logic Diagram

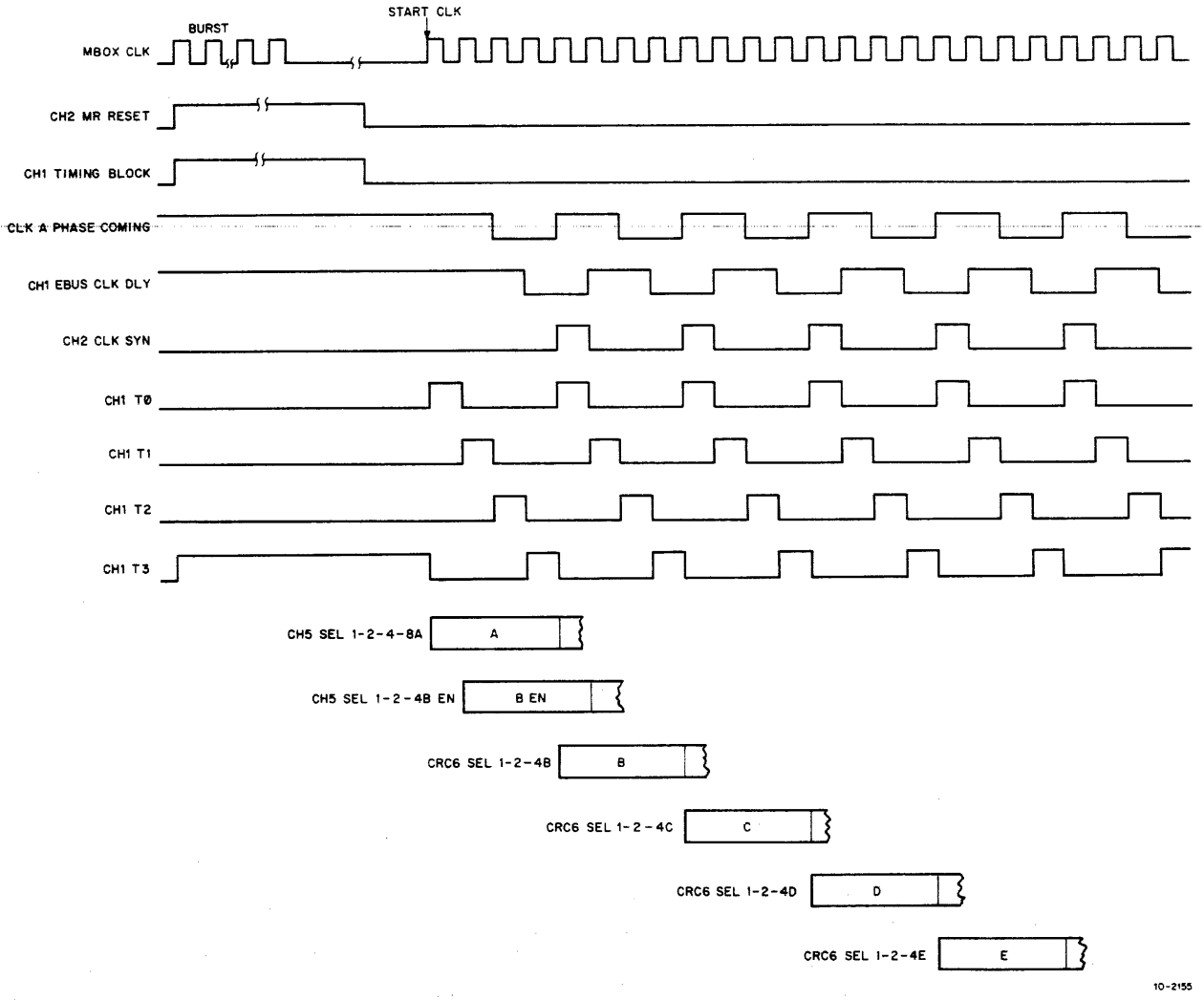


Figure 3-39 Timing Logic, Timing Diagram

The output of the scanner is applied to a decoder and a four-state 3-bit shift register. The decoder converts the three least significant bits of the scanner count to one of eight select signals (CBUS SEL 0–7E) for enabling the corresponding RH20 Massbus controllers one at a time. The contents of the four-state 3-bit shift register are advanced every four clock ticks at T0 to advance the contents (selected channel number) so that the appropriate RAMs and queues can be addressed for transferring the data.

3.8.2 Control Request Queues

The three control request queues (Figure 3-40) are implemented in the channel control logic to queue the CBus control requests (RESET, START, and DONE) so that they can be executed in accordance with a specific order of priority. Since the channel control is designed to handle up to eight separate RH20 Massbus controllers, each control request queue has eight locations, one for each channel. As the scanner (Subsection 3.8.1) selects each RH20 Massbus controller in the prescribed order, so are the individual locations in control request queues also selected. It is this operating characteristic that causes the control request queues to remember which RH20 asserted a particular CBus control line (RESET, START, or DONE).

The control request must be queued because the channel control may be busy at the time the request is made. The channel control may be busy executing a data transfer request (CBUS REQUEST) for the same channel or it may be busy executing a memory request for the same or another channel.

The queued control requests are executed in a set order of priority. This priority arrangement is:

a. RESET 0–7
b. START 0–7
c. DONE 0–7

Further, the priority arrangement is set up to execute the request for a lower numbered channel before the higher numbered channel. That is, the order of priority is 0, 1, 2, 3, 4, 5, 6, and 7.

If a given RH20 controller does not request to transfer data by asserting CBUS REQUEST after it is selected, a current or pending CBus control request can be granted and executed (Figure 3-40 and 3-41). When granted, the request initiates a RAM cycle to update the channel control RAMs (Subsection 3.8.5). If the control request is a reset request, all bits of the control RAM are simply cleared and the RESET bit is set. If the control request is a start request, the control RAM is cleared as with the reset request and the MB request queue (Subsection 3.8.7) is set up to fetch the CCW for the requesting channel.

NOTE

CBUS RESET and START may both be asserted by the RH20 at the same time. If this is the case, the channel control executes two separate RAM cycles. First, it will execute a RAM cycle to satisfy CBUS RESET; then a second RAM cycle is executed for CBUS START.

If the control request is a done request, the RAM cycle is executed to set the DONE bit in the control RAM. Thereafter, this bit is checked everytime another RAM cycle for the same channel is executed to ensure that the transfer is executed without error.

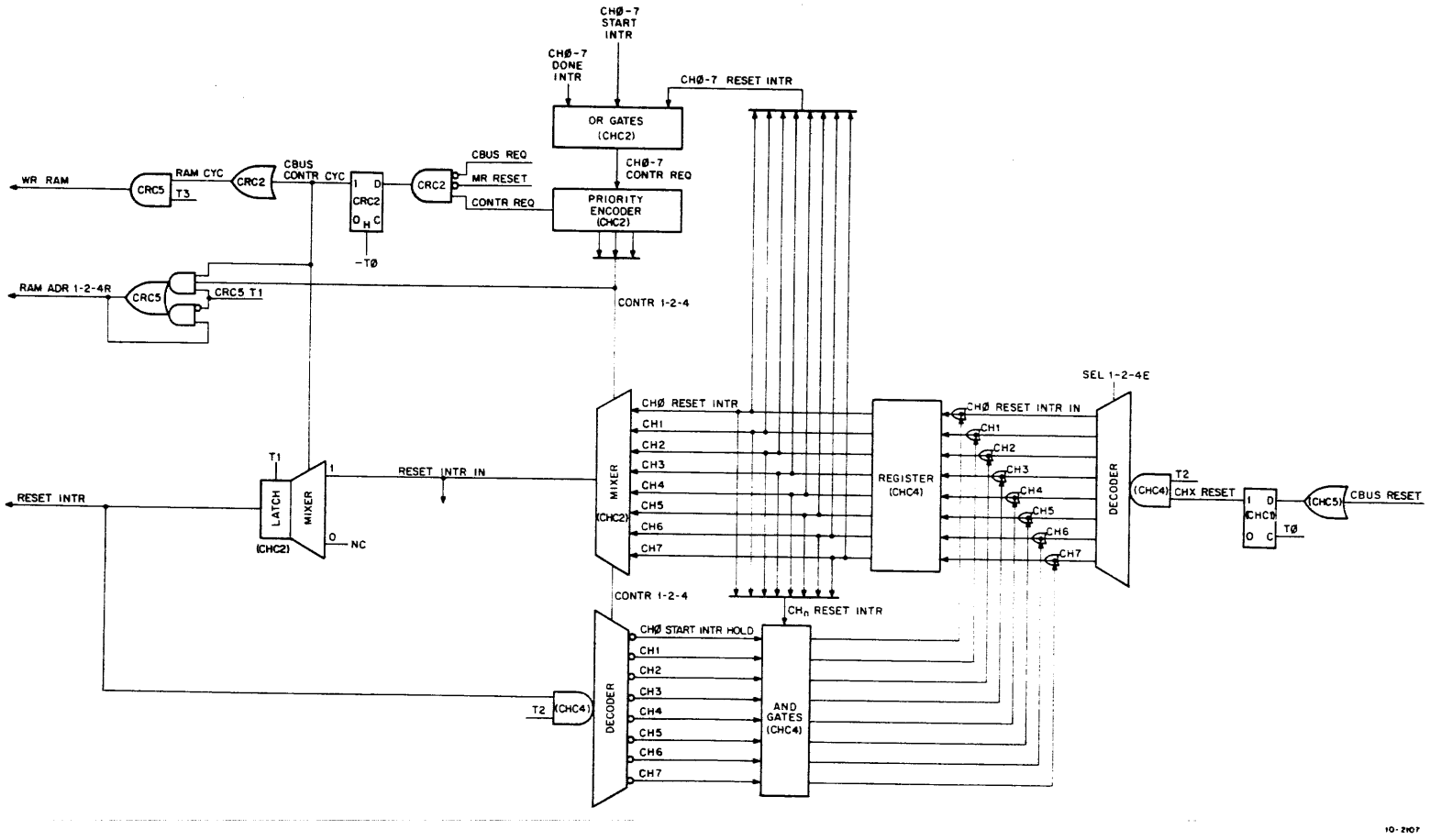


Figure 3-40 Control Request Queues, Simplified Logic Diagram (Sheet 1 of 4)

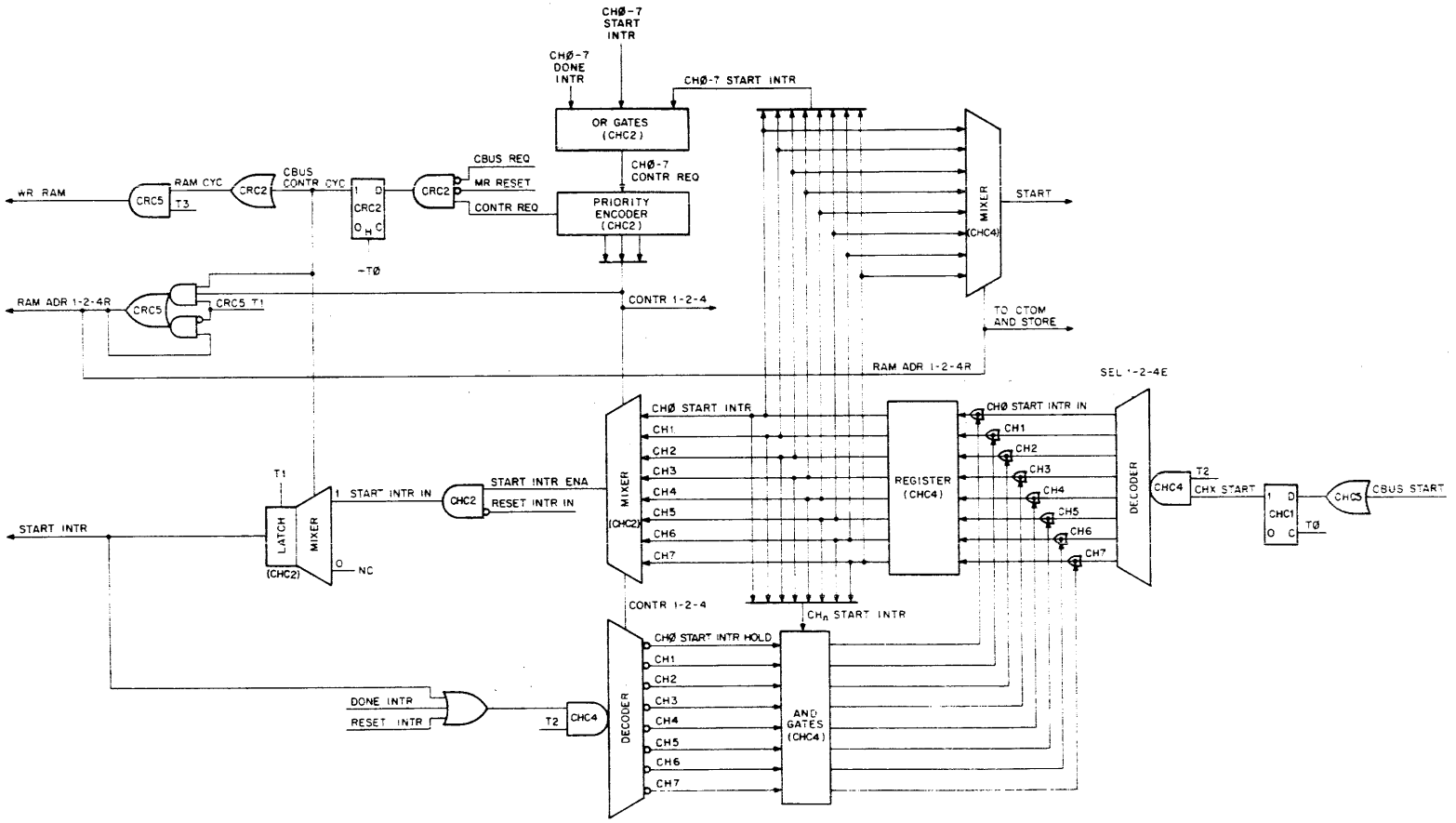


Figure 3-40 Control Request Queues, Simplified Logic Diagram (Sheet 2 of 4)

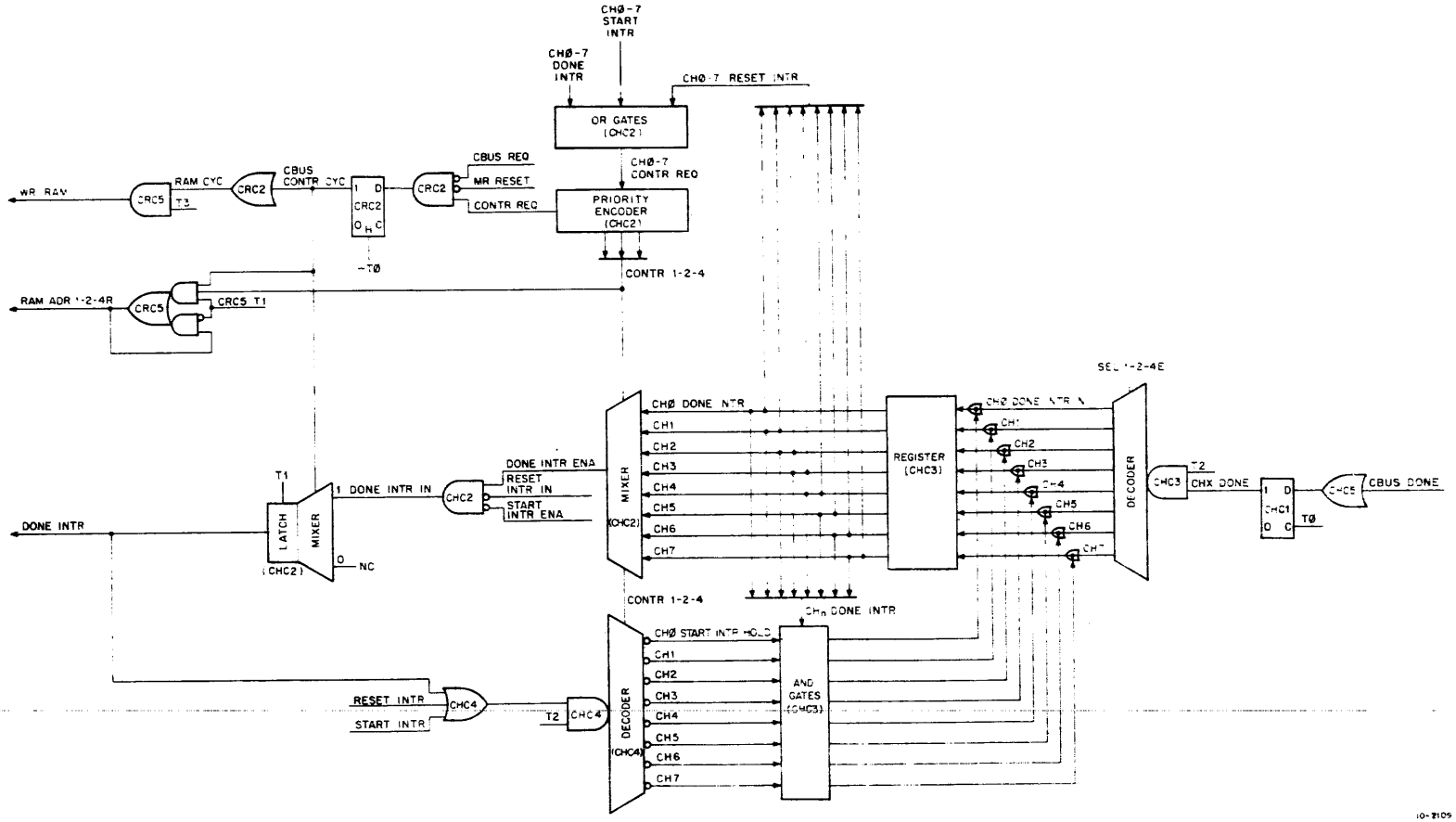


Figure 3-40 Control Request Queues, Simplified Logic Diagram (Sheet 3 of 4)

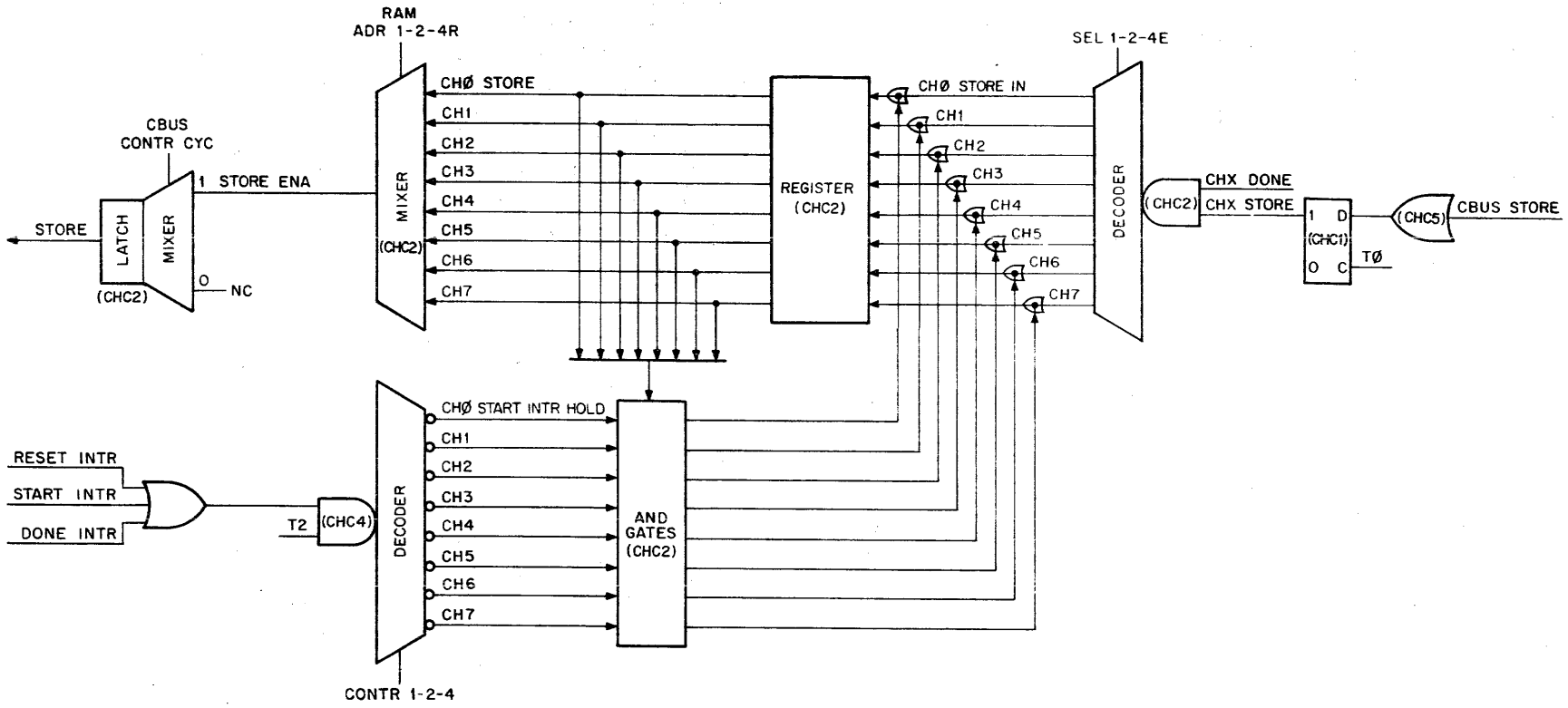


Figure 3-40 Control Request Queues,
Simplified Logic Diagram
(Sheet 4 of 4)

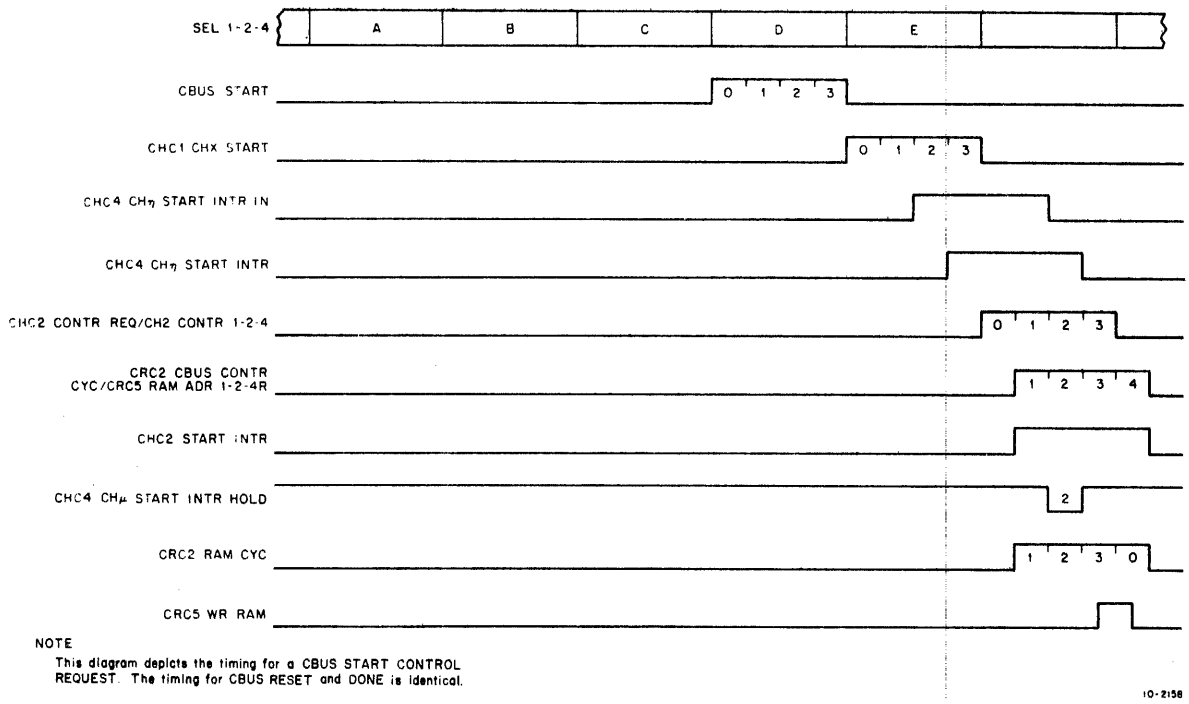


Figure 3-41 Control Request Queue, Timing Diagram

3.8.3 CTOM Register

Once a data block transfer is started, the channel logic must remember the direction of the transfer (CTOM or NOT CTOM). This is necessary so that the pointers can be updated correctly every time a CBUS REQUEST to transfer a word is executed. For this reason, the CTOM register (Figure 3-42) is implemented in the channel logic. The register contains eight bits, one for each channel. Steering logic is implemented for loading and reading the appropriate bit of the register. A specific bit of the register is set or cleared depending on the state of the CBUS CTOM line whenever the RH20 asserts CBUS START, which occurs when a block transfer is started. The actual bit that is loaded depends on the current position of the scanner (SEL 1-2-4E), which specifies the channel number that asserted CBUS START. The decoder at the input of the register, which serves as the steering network, is controlled by the SEL 1-2-4E signals to select the appropriate bit. A mixer at the output of the register serves as a steering network to select the appropriate bit when the register needs to be read. The mixer is controlled by the CRC RAM ADR 1-2-4R, which specifies the channel number for which a request is being executed.

3.8.4 CBUS Request Logic

This logic provides timing and control functions for moving 36-bit data words between the MBox CH BUF and the RH20 data buffer via the CBus (Figure 3-43). The RH20 controller asserts CBUS REQUEST one scanner time slot (slot B) after it is selected to inform the channel control that it is ready to send or receive a word. Along with CBUS REQUEST, the RH20 also asserts or negates CBUS CTOM, although this fact is stored in the CTOM register when the transfer is started (Subsection 3.8.3).

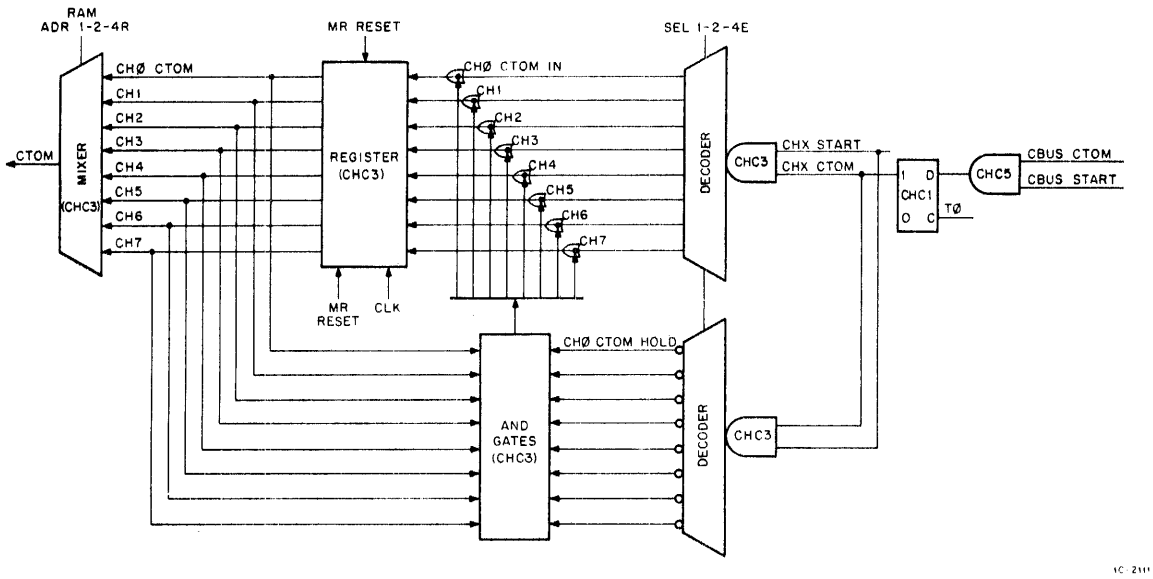


Figure 3-42 CTOM Register, Simplified Logic Diagram

The timing differs for CTOM and NOT CTOM transfers (Figures 3-44 and 3-45) because:

a. For CTOM transfers, the RH20 places the data on the CBUS data lines at the beginning of scanner time slot D and, therefore, the channel control logic must strobe the data into the CH BUF at the beginning of time slot E.

b. For NOT CTOM transfers, the channel control logic places the data on the CBus data lines at the beginning of scanner time slot D and, therefore, the RH20 must strobe the data into its buffer at the beginning of time slot E.

In either case, the RAM cycle is executed to load the current CH PTR from the RAM into a shift register and to update (increment) the CH PTR in the RAM. The CH PTR that is loaded in the shift register is then used in forming the CH BUF address.

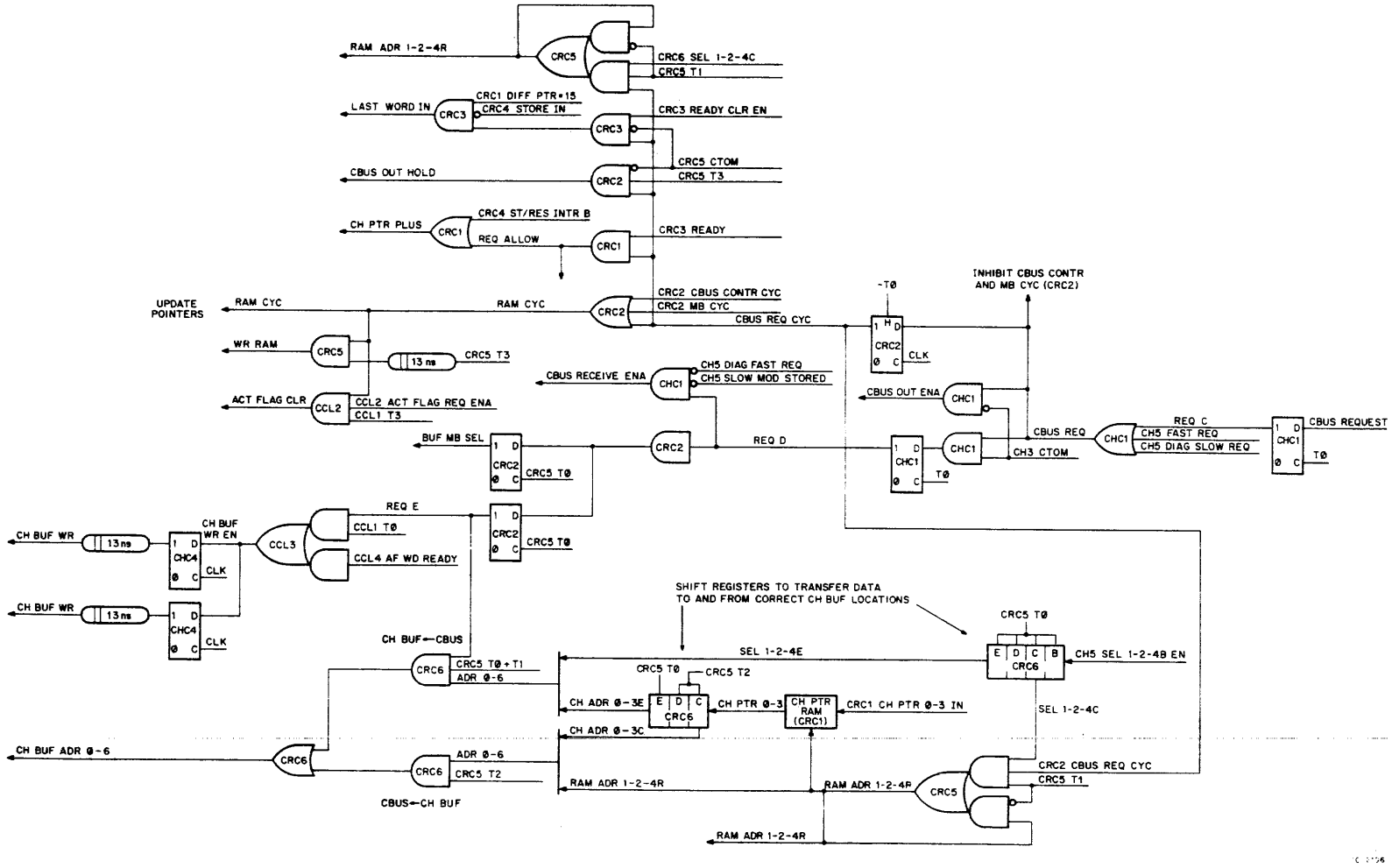


Figure 4-43 CBus Data Request Logic, Simplified Logic Diagram

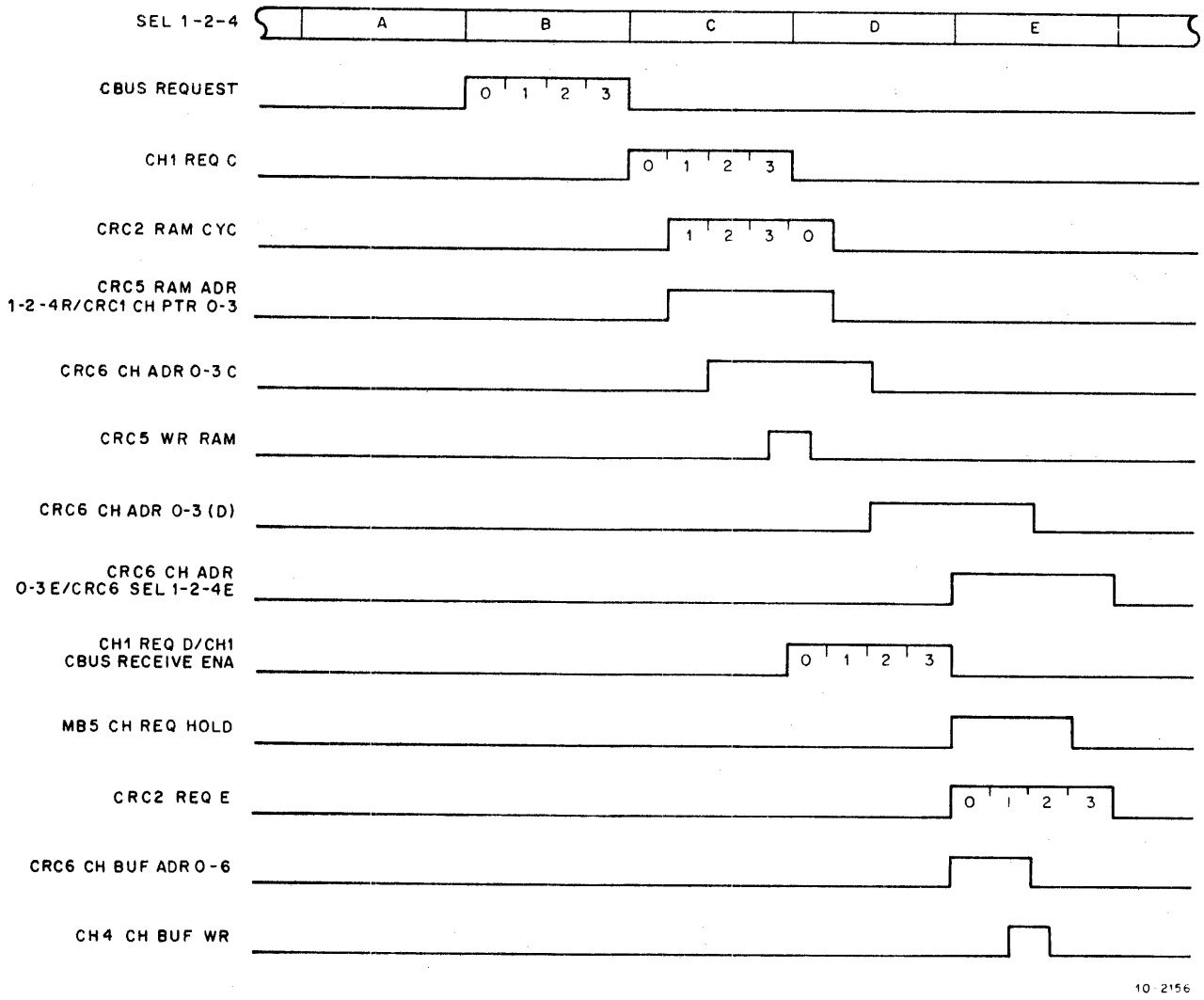


Figure 3-44 CBus Data Request (CTOM) Logic, Timing Diagram

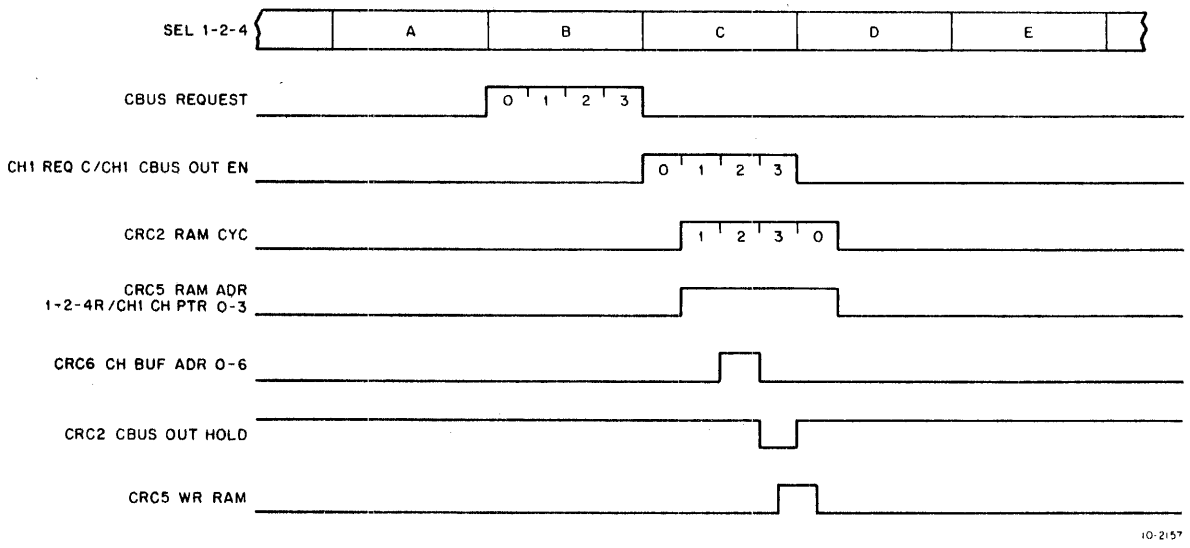


Figure 3-45 CBUS Data Request (NOT CTOM) Logic, Timing Diagram

3.8.5 Control RAMs

Status and control information for each of the eight channels is maintained in the control RAMs (Figure 3-46). Every time a RAM cycle is executed, the RAMs are addressed and updated. The RAM contains eight sets of status and control information, one set for each channel. Included in each set of status and control information (Table 3-16) is the following:

a. Control Bits
   1. RESET
   2. READY
   3. LAST WORD
   4. DONE
   5. STORE
   6. ERROR
   7. OP Code 00
   8. OP Code 01
   9. OP Code 02 (REVERSE)
   10. CMD TOGGLED
   11. CMD STORED
   12. ACT CNT 0-2
   13. MEM PTR 0-3
   14. CH PTR 0-3

b. Status Bits
   1. MEM PAR ERR
   2. - ADR PAR ERR
   3. - WC=0 STORED
   4. NXM ERR
   5. LAST XFER ERR
   6. RH20 ERR
   7. LONG WC ERR
   8. SHORT WC ERR
   9. OVN ERR

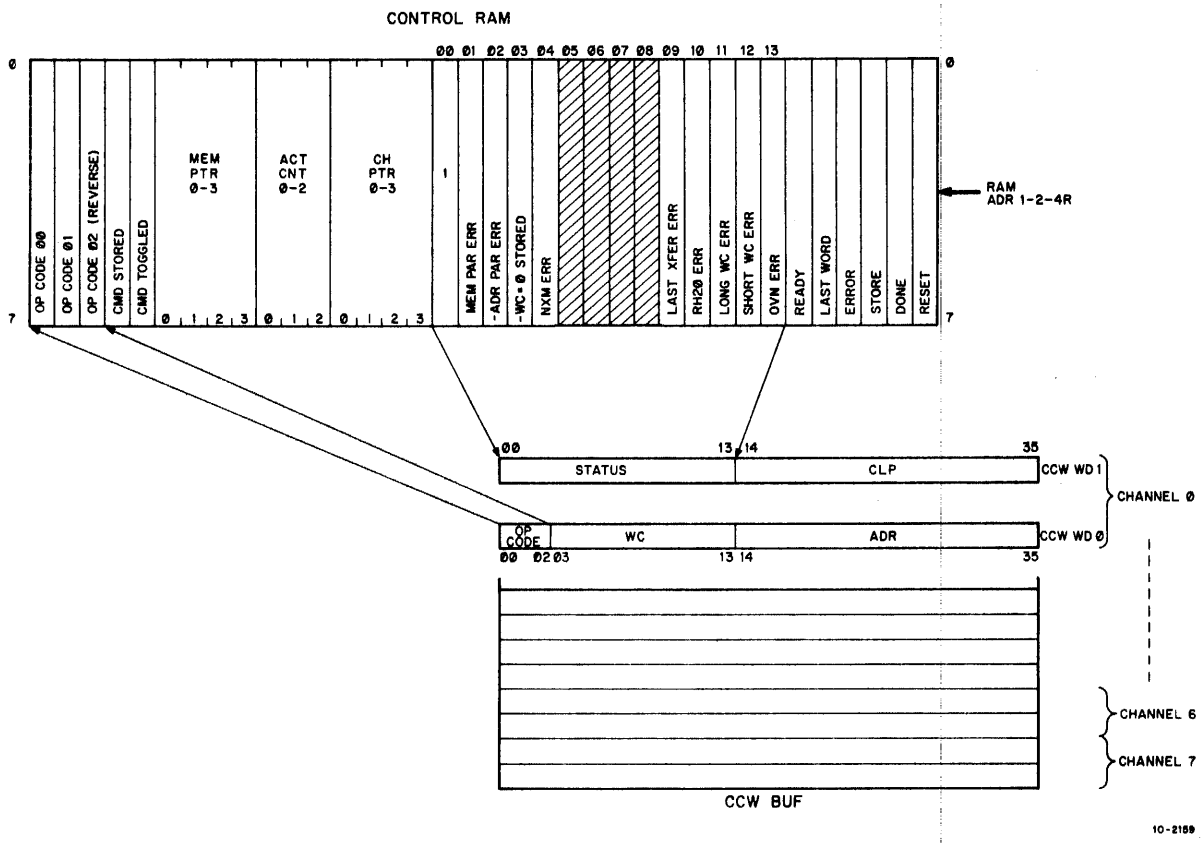


Figure 3-46 Control RAM Structure

Table 3-16 Control RAM Bit Description

NAME	DESCRIPTION
Control Bits – one for each channel	
OP CODE 00–02 (CRC3)	These bits are loaded with the OP Code of the CCW when it arrives from memory.
CMD TOGGLED/STORED (CCL6)	If a memory error occurs, these two bits identify which block transfer (current or previous) caused the error.
MEM PTR 0–3 (CRC1)	These four bits point to (address) the next location in the CH BUF for memory transfers. This location is the next empty location for NOT CTOM transfers or the next word for CTOM transfers. Everytime an action flag memory request is executed, these bits are used to address the CH BUF. After the request is executed, these bits are updated.
ACT CTR 0–2 (CRC2)	These three bits are typically loaded with a function of the two LSBs of the WC and ADR of the CCW. This function specifies how many words are to be transferred.
CHAN PTR 0–3 (CRC1)	These four bits point to (address) the next location in the CH BUF for CBUS transfers. This location is the next empty location for CTOM transfers or the next word for NOT CTOM transfers. Everytime a CBUS data request is executed, these bits are used to address the CH BUF. After the request is executed, these bits are updated.
READY (CRC3/CHC1)	Set when the channel control is ready to transfer data. The channel control is ready to transfer data after it fetches a DATA XFER CCW. For NOT CTOM transfers, the channel control must also fetch at least two data words (unless a single word is to be transferred) before it is ready. This bit is the source for the CBUS READY line.
LAST WORD (CRC3/CHC1)	Set only for NOT CTOM transfer when the last word is placed on the CBUS. This bit is the source for the CBUS LAST WORD line.
ERROR (CRC3/CHC1)	Set if any of the following error bits are set: a. Memory errors 1. NXM ERR 2. MEM PAR ERR 3. ADR PAR ERR b. SHORT WC ERR c. LONG WC ERR d. RH20 ERR e. OVN ERR f. LAST XFER ERR This bit is the source of the CBUS ERROR line.

Table 3-16 Control RAM Bit Description (Cont)

NAME	DESCRIPTION
Control Bits – one for each channel	
STORE (CRC2)	Set when a Control RAM cycle is executed in response to CBUS DONE, providing CBUS STORE was also asserted. After this bit is set, a memory store request is queued in the MB Request queue.
DONE (CRC3)	Set when a Control RAM cycle is executed in response to CBUS DONE. This bit is used by the channel control logic in terminating the transfer orderly.
RESET (CRC3)	Set when a Control RAM cycle is executed in response to CBUS RESET. At the same time this bit is set, all other bits in the control RAM are cleared. The fact that CBUS RESET was asserted is stored so that the appropriate address for fetching the initial CCW from the EPT can be formed.
Status Bits – one for each channel	
Bit 00	Always set.
MEM PAR ERR (CCL1) Bit 01	Set when a data parity error is sensed while transferring a CCW from an MB to the CCW BUF.
–ADR PAR ERR (CCL1), Bit 02	Cleared when an address parity error is sensed after a Channel Request for a CCW is issued.
–WC = 0 STORED (CCL6), Bit 03	Cleared when WC reaches zero as a result of action flag CHAN request (CTOM or NOT CTOM) or when a CCW with a zero WC field is fetched from memory.
NXM ERR (CCL1), Bit 04 (CCL1)	Set if the NXM timer (MBZ3) expires after a Channel Request is issued by the channel control and granted by the cache control.
Bit 05–08	Not Used.
LAST XFER ERR (CCL6), Bit 09	Set if NXM ERR bit 04 is set and a second block transfer was started. This means that the NXM was caused by the previous block transfer.
RH20 ERR (CRC4), Bit 10	Set if the RH20 asserts CBUS START even though the channel control is not ready. The channel control is ready only when CBUS READY is negated.
LONG WC ERR (CRC4), Bit 11	Set if the cumulative word count, specified by the channel command list, was larger than the number of words the RH20 massbus controller transferred.

Table 3-16 Control RAM Bit Description (Cont)

NAME	DESCRIPTION
Status Bits – one for each channel	
SHORT WC ERR (CRC4), Bit 12	Set if the RH20 massbus controller sends more words than are specified by the cumulative word count of the channel command list.
OVN ERR (CRC4), Bit 13	Set if the channel control is unable to keep up with the RH20 controller's demands for data.

The control bits reflect the current state of the channel control logic and the CBus for a given channel. These bits are tested/set by the channel control during the course of executing a data block transfer operation to ensure error-free operation.

3.8.6 Action Flag Arithmetic Logic

The action flag arithmetic logic (Figure 3-47) keeps track of the words in the CH BUF for all eight channels by maintaining a channel pointer and a memory pointer. These pointers are stored in RAM's that are updated every time a word is moved in or out of the buffer. The memory pointer is updated every time a memory request is executed and the channel pointer is updated everytime a CBus request is executed. The difference between these pointers (PTR DIFF) represents the number of words in the CH BUF for CTOM transfers or the number of empty locations for NOT CTOM transfers.

Besides the pointers, the action flag arithmetic logic also maintains an action count for each channel. This count, which is normally a function of the WC and ADR of the CCW, specifies how many words are to be moved to/from memory. Typically, four words are moved to/from memory at a time. However, if the address does not fall on the quadword boundary, less than four words must be transferred. This can only occur at the beginning and at the end of a block transfer specified by a CCW. The action count is maintained in a RAM, like the pointers, and is updated every time a memory request is executed or when a new CCW is loaded.

All three RAMs are addressed by a 3-bit code that identifies the channel for which the request is being executed. In addition to the RAMs and their input logic for maintaining the action count and the pointers for each channel, the action flag arithmetic logic also includes a number of Arithmetic Logic Units (ALU's) and mixers for applying an equation to the action count and the pointers to determine if a memory request (CRC AF REQ ENA) is needed. Two equations are implemented; one for CTOM transfers and another for NOT CTOM transfers. They are:

a. CTOM: (CHAN PTR-MEM PTR) - ACT CNT $\geq$0 SET CRC AF REQ ENA

b. NOT CTOM: 15+(CHAN PTR-MEM PTR) - ACT CNT $\geq$0 SET CRC AF REQ ENA

3.8.6.1 Action Count – The action count specifies the number of words (1, 2, 3, or 4) to be transferred to/from memory. The action count is used to set up the word request logic and the MB control logic (Subsection 3.8.7) when a request for memory is initiated. In addition, this count, along with the memory and channel pointers, is used in an equation to determine whether to set CRC AF REQ ENA, which indicates that more words must be transferred to/from memory.

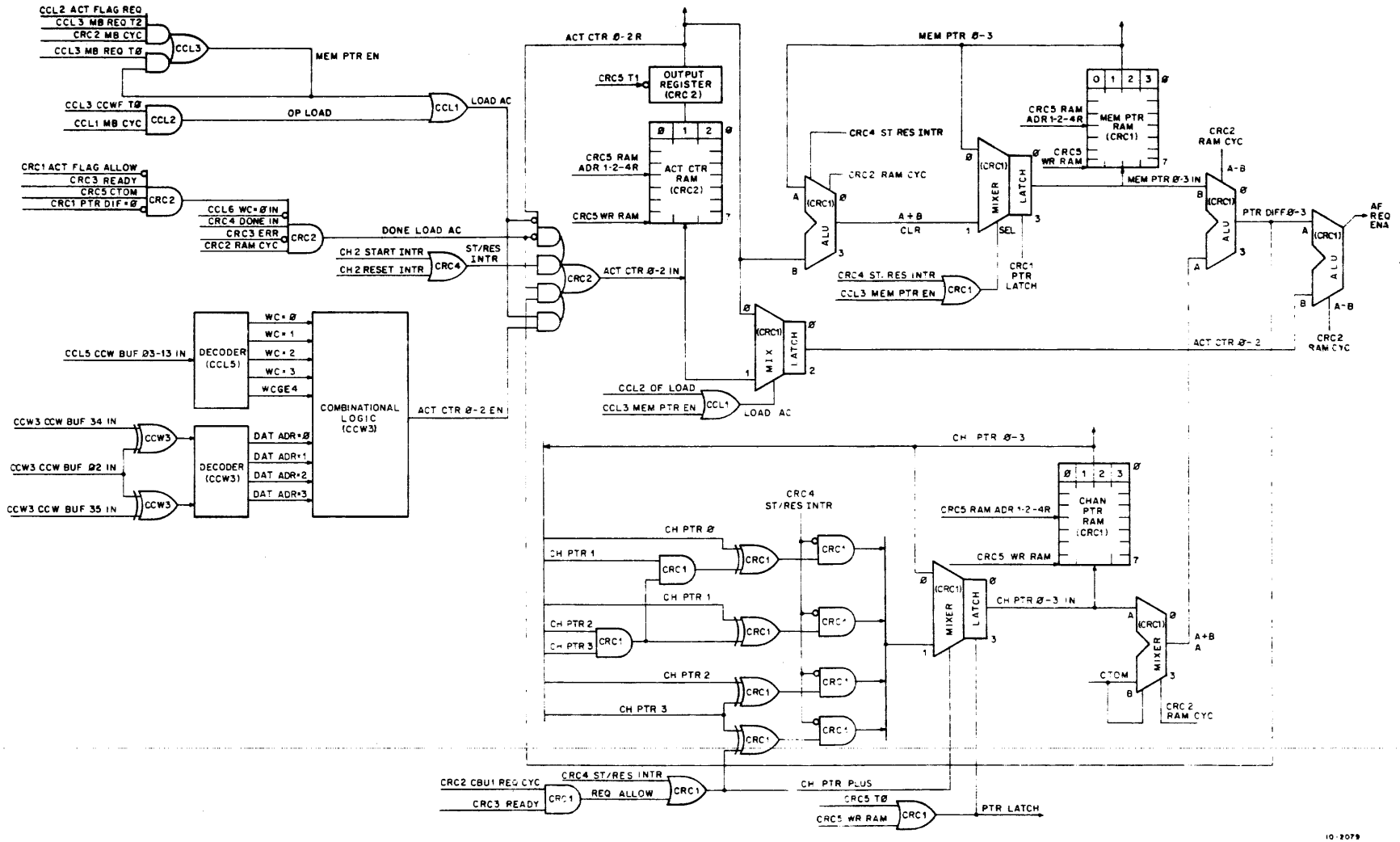


Figure 3-47 Action Flag Arithmetic Logic, Simplified Logic Diagram

On CBUS START/RESET, the action count is set to 7. This is done to initially inhibit CRC AF REQ ENA. Since a start/reset operation will set the CH PTR and MEM PTR to zero, a non-zero action count is required to prevent CRC AF REQ ENA from asserting.

When a CCW is loaded into the CCW buffer or when the second RAM cycle for an action flag request is executed, the action count is set to a function of the WC and the ADR (Table 3-17).

On done for CTOM block transfers, the Action Count is set to the value of the PTR DIF to empty the CH BUF.

Table 3-17 Action Count Truth Table

CCW ADR (CCW3)	DAT ADR (CCW3)		CCW WC (CCL5)	ACTION COUNT IN (CRC2)	
(BITS 34-35)	FORWARD	REVERSE		FORWARD	REVERSE
0	0	3	⩾4	4	1
0	0	3	3	3	1
0	0	3	2	2	1
0	0	3	1	1	1
0	0	3	0	4*	1*
1	1	2	⩾4	3	2
1	1	2	3	3	2
1	1	2	2	2	2
1	1	2	1	1	1
1	1	2	0	3*	2*
2	2	1	⩾4	2	3
2	2	1	3	2	3
2	2	1	2	2	2
2	2	1	1	1	1
2	2	1	0	2*	3*
3	3	0	⩾4	1	4
3	3	0	3	1	3
3	3	0	2	1	2
3	3	0	1	1	1
3	3	0	0	1*	4*

*These cases are not used.

3.8.6.2 Memory Pointer – The memory pointer is used in forming the address for the CH BUF during a memory transfer. It points to the CH BUF location which is to receive the next word from memory for NOT CTOM transfers or from which the next word to be moved to memory is to be taken for CTOM transfers. This pointer, along with the channel pointer and action count, is also used in an equation to decide whether to set CRC AF REQ ENA, which indicates that more words must be transferred to/from memory.

On CBUS START/RESET, the memory pointer is set to 0.

When the second RAM cycle for an action flag request is executed, the memory pointer is updated by adding the action count.

3.8.6.3 Channel Pointer – The channel pointer is used in forming the address for the CH BUF during CBus transfers. It points to the CH BUF location which is to receive the next word from the RH20 for CTOM transfers or from which the next word is to be moved to the RH20 for NOT CTOM transfers. This pointer, along with the memory pointer and the action count, is also used in an equation to determine whether to set CRC AF REQ ENA, which indicates that more words must be transferred to/from memory.

On CBUS START/RESET, the channel pointer is set to 0.

When a RAM cycle for a CBUS REQUEST is executed, the channel pointer is updated by incrementing the pointer by one.

3.8.6.4 Operation – A number of different functions control the operation of the action flag arithmetic logic. The functions and how these functions effect the operation of the action flag arithmetic logic are described in the following paragraphs.

a. **Initialization** – The action flag arithmetic logic is initialized when the channel control executes a CBus control cycle in response to a START or RESET INTR from the control queues (Subsection 3.8.2). The logic is initialized as follows:

   1. The action count is set to seven.

   2. The memory pointer is set to zero.

   3. The channel pointer is set to zero.

   4. CRC AF REQ EN is not set because the result of the applied equation is not zero (Subsection 3.8.6 a and b).

b. **Fetch CCW** – After a channel is started, the CCWs are automatically fetched from memory and loaded (CCL CCW BUF WR) into the CCW BUF throughout the block transfer. At the same time the CCW is loaded into the CCW BUF, the action counter RAM is updated (CCL OP LOAD is asserted) to reflect the WC and ADR of the new CCW. CCL OP LOAD is asserted to enable the input logic to the action counter RAM to load the new action count (CCW ACT CTR 0–2 EN). At the same time, the new action count is also transferred to the ALU, which generates CRC AF REQ EN when the equation is satisfied.

c. **Transfer a group of words to/from memory** – When the action flag equation (Subsection 3.8.6 a and b) is satisfied, CRC AF REQ ENA is set to request a memory transfer. Subject to the priority scheme, the action flag request is granted to transfer a group of words to/from memory (via the MBs). After the words are transferred, the action count and the memory pointer are updated (CCL MEM PTR EN is asserted). CCL MEM PTR EN is asserted to enable the input to the action count RAM to load the new action count (CCW ACT CNT 0–2 EN), to add the action count to the memory pointer, and to transfer the action count to the ALU that generates CRC AF REQ ENA when the equation is satisfied.

d. **Transfer a single word to/from RH20** – Every time CBUS REQUEST is asserted by the RH20, a single word is transferred to/from the CH BUF via the CBus from/to the RH20. When the word is transferred, the channel pointer is also updated (CRC CH PTR PLUS) by simply adding one to the pointer count. As soon as enough words/empty locations are in the CH BUF, as computed by the action flag equation, CRC AF REQ EN is set to request another memory transfer.

e. **DONE for CTOM** – When the RH20 asserts CBUS DONE, providing the block transfer is from controller to memory (CTOM), the channel control must continue to empty the CH BUF. To facilitate this operation, some additional logic is implemented to update the action counter (CRC DONE LOAD AC) until all the words are transferred. CRC DONE LOAD AC is asserted to transfer the pointer difference (CRC PTR DIFF 0–2) to the action count RAM. The pointer difference specifies the number of words still remaining in the CH BUF.

NOTE

This logic is not required for NOT CTOM transfers because the RH20 will not assert DONE unless it has received the last word that leaves an empty CH BUF.

3.8.7 MB Request Queues

The three MB request queues (Figure 3-48) are implemented to queue the MB (memory) requests (CCWF, ACTION FLAG, and MEM STORE) so that they can be executed in accordance with a specific order of priority as RAM cycles become available. Since the channel control is designed to handle up to eight separate channels (RH20 Massbus controllers), each MB request queue has eight locations, one for each channel.

Each of the three queues are constructed from a pair of 3 by 8 decoders and their associated enabling (INPUT) logic; an 8-bit register, an 8 by 3 priority encoder, and a set of eight AND gates. The decoders serve as steering networks for setting and clearing individual bits in the register. The 3-bit address that is formed when a RAM cycle (CBUS REQ CYC, CONTR CYC, or MB CYC) is granted, is used to control the decoders to connect the decoder input logic to the register bit that corresponds to the channel number for which the RAM cycle is executed.

As the individual bits of the MB request queues are selected, so also are the associated control bits in the control RAM selected. The input logic of the decoder, therefore, receives the control bits for the channel for which the RAM cycle is being executed.

This operating characteristic of the channel control logic facilitates the testing of the state of each channel to set/clear the appropriate bits in the MB request queues when required. The AND gates provide the paths for latching and clearing the register bits. The priority encoder forms a 3-bit address that corresponds to the lowest number request (highest priority) in the queue. When a RAM cycle for a pending MB request is executed, the address formed by the priority encoder is selected by the mixer and is used to address the MB request queues and the control RAM. These three bits are also used in forming the address for the CH BUF and the CCW BUF.

MB requests are granted and executed only when CBus data and control requests are not pending, and then only in the following order of priority.

a. CCWF 0–7
b. ACT FLAG 0–7
c. MEM STORE 0–7

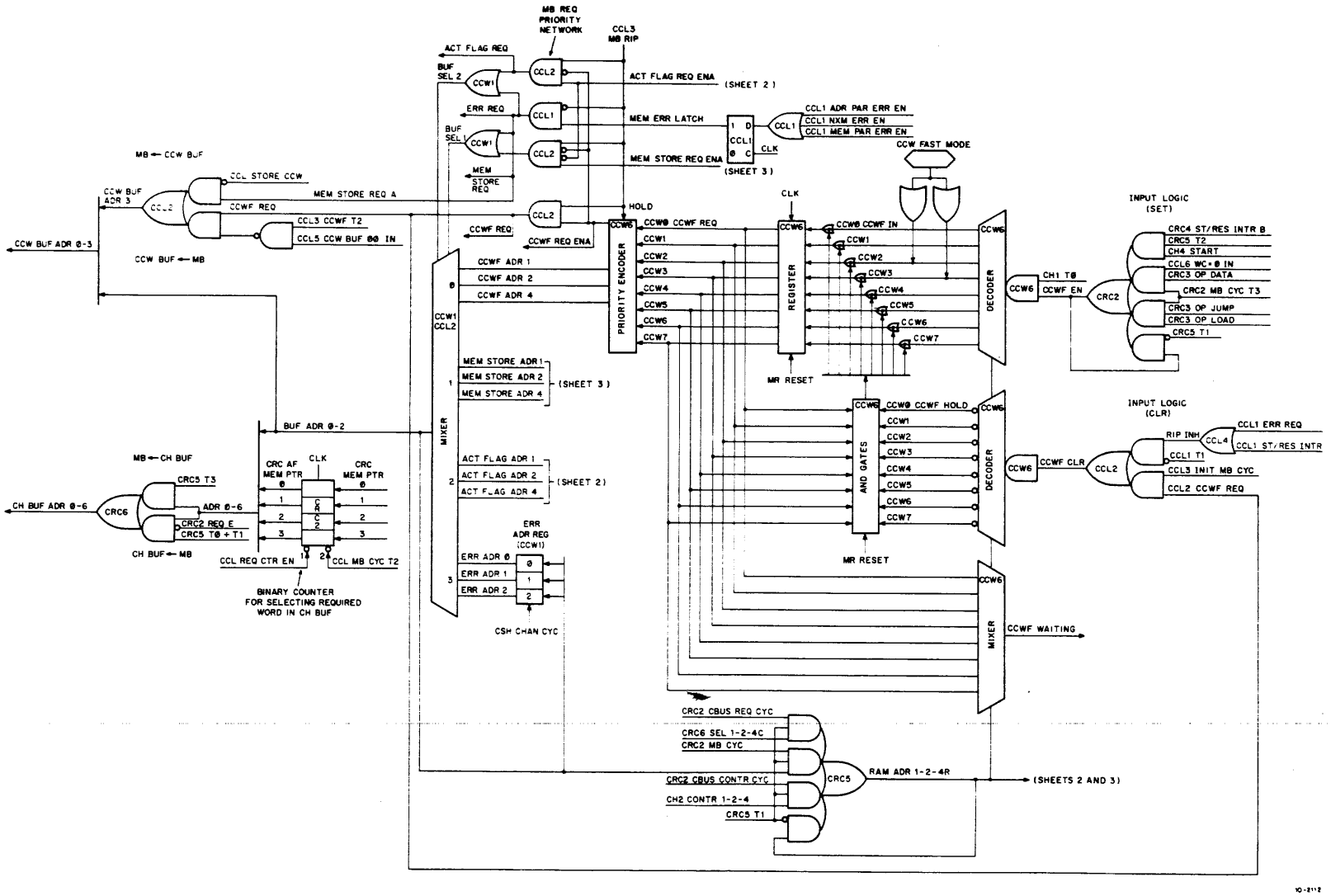


Figure 3-48 MB Request Queues, Simplified Logic Diagram (Sheet 1 of 3)

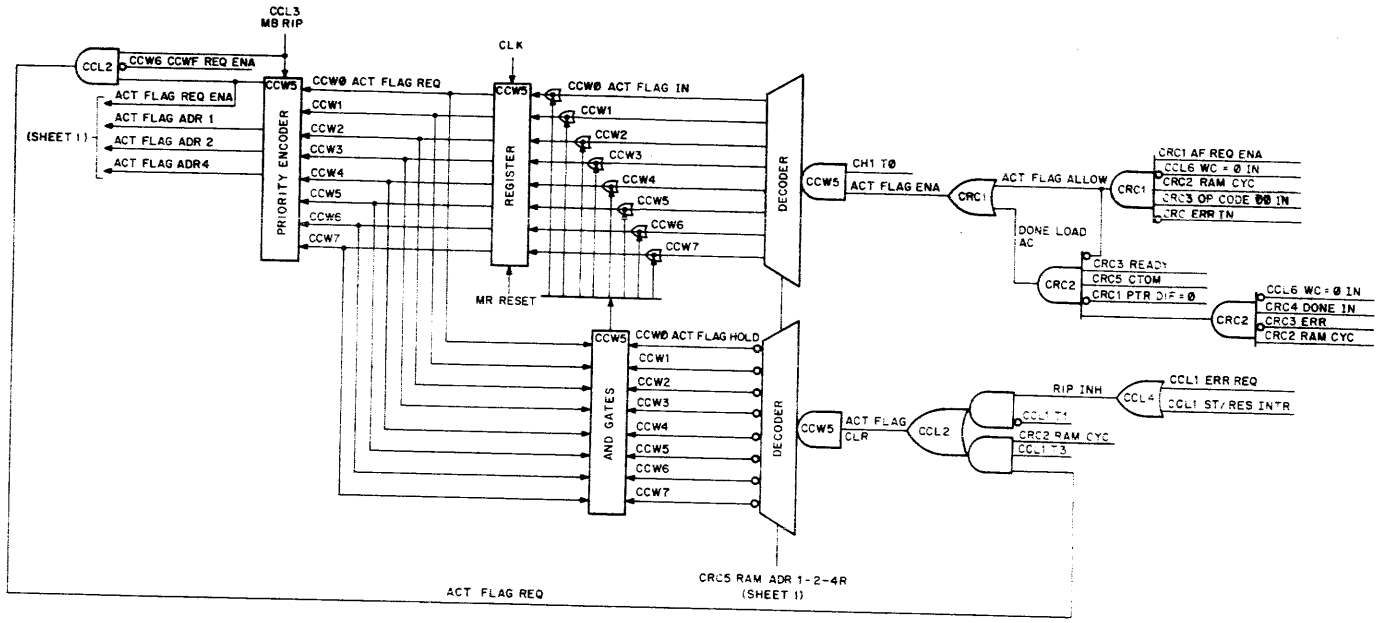


Figure 3-48 MB Request Queues. Simplified Logic Diagram (Sheet 2 of 3)

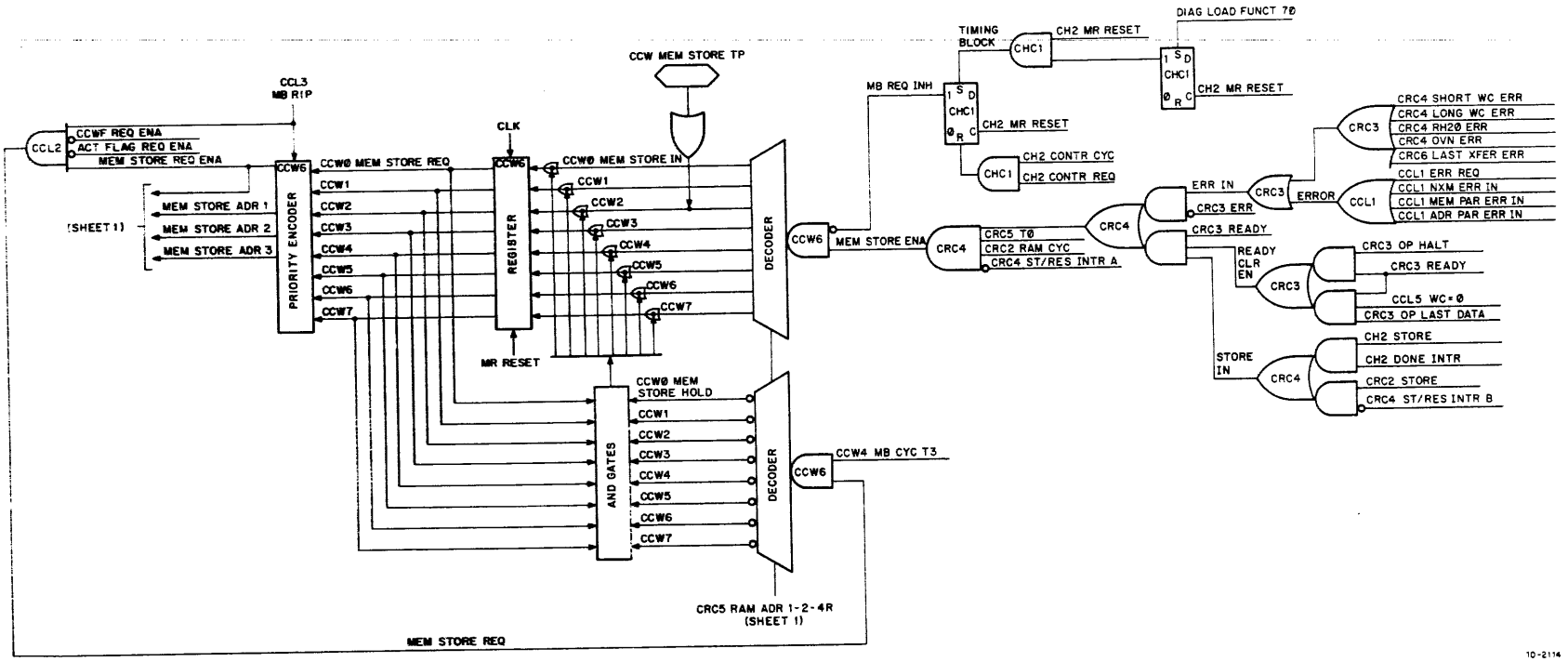


Figure 3-48 MB Request Queues, Simplified Logic Diagram (Sheet 3 of 3)

Further, the priority arrangement is set up to execute the request for a lower numbered channel before a higher numbered channed. That is, the order of priority is 0, 1, 2, 3, 4, 5, 6 and 7.

If a given RH20 controller does not request to transfer data by asserting CBUS REQUEST after it is selected and if a control request is not pending in the control request queues (Subsection 3.8.2), the highest priority MB request will be granted and executed. When granted, the request initiates a RAM cycle to set up the request (CCL CHAN REQ) for memory and clear out the request in the queue. The MB request timing and control logic (Subsection 3.8.8) then takes over to transfer the words between the channel buffers (CCW BUF or CH BUF) and the MBs. After all the words are transferred, a second RAM cycle is executed to update the CCW, the pointers, and the control bits in the control RAM. If an error is sensed during the course of the transfer, a RAM cycle will be initiated after the transfer to set the error-bit in the control RAM.

3.8.8 MB REQUEST LOGIC

MB requests are initiated to transfer words between main memory via the MB's and the channel buffers (CCW and CH). Four basic types of MB requests can be executed. They are:

a. Transfer a CCW from main memory to the CCW BUF.

b. Transfer a group of words (maximum of four) from main memory to the CH BUF.

c. Transfer a group of words (maximum of four) from the CH BUF to main memory.

d. Transfer the CCW and the CLP/Status Word from the CCW BUF to main memory.

The logic that executes the MB request includes the MB request timing logic, the MB control logic, and the word request logic (Figures 3-49 through 3-51).

The MB request timing logic, as the name implies, provides the timing signals for executing the request. Combinational and sequential logic is employed in generating the timing signals.

The MB control logic incorporates three binary counters in addition to some combinational and sequential logic. The main function of this logic is to keep track of the number of words and their location in the buffers as they are transferred. The word request logic specifies and remembers how many words are to be transferred. For NOT CTOM transfers, this logic also keeps track of the words that are moved into and out of the MBs.

The MB request logic is driven by the MB request queues. As long as requests are pending in the queue, memory is held and requests are initiated, one after the other, in the prescribed order of priority (Subsection 3.8.7).

3.8.8.1 CCWF Request – The following description details a CCWF MB request to transfer a CCW from main memory to the CCW BUF (Figure 3-52).

a. A CCWF request is queued when:

1. A channel is started (CBUS START).

2. WC reaches zero and the current CCW is a normal data transfer CCW.

3. A jump CCW was fetched and loaded into the CCW BUF. (The OP code is loaded into the control RAM.)

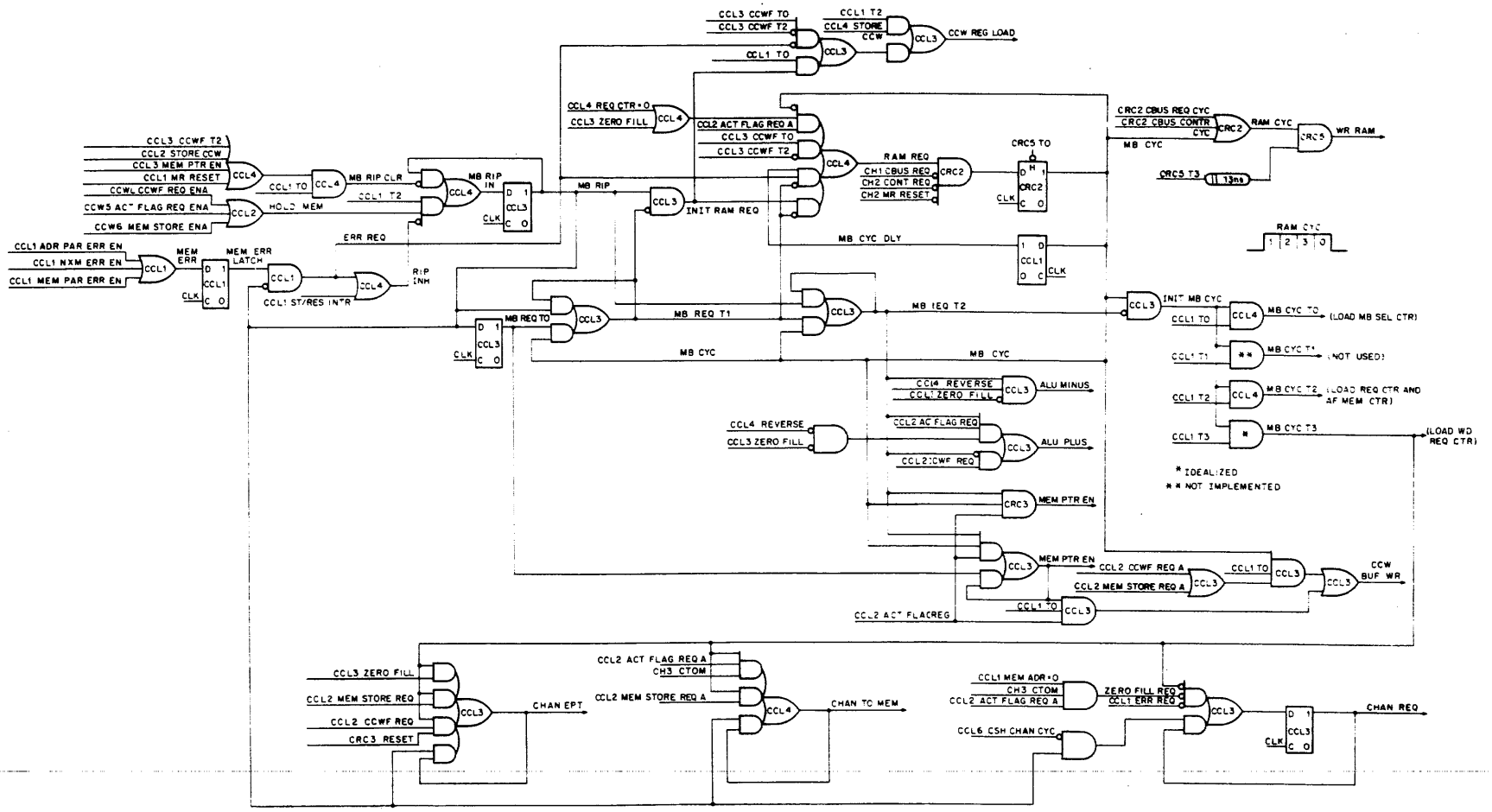


Figure 3-49 MB Request Timing Logic, Simplified Logic Diagram

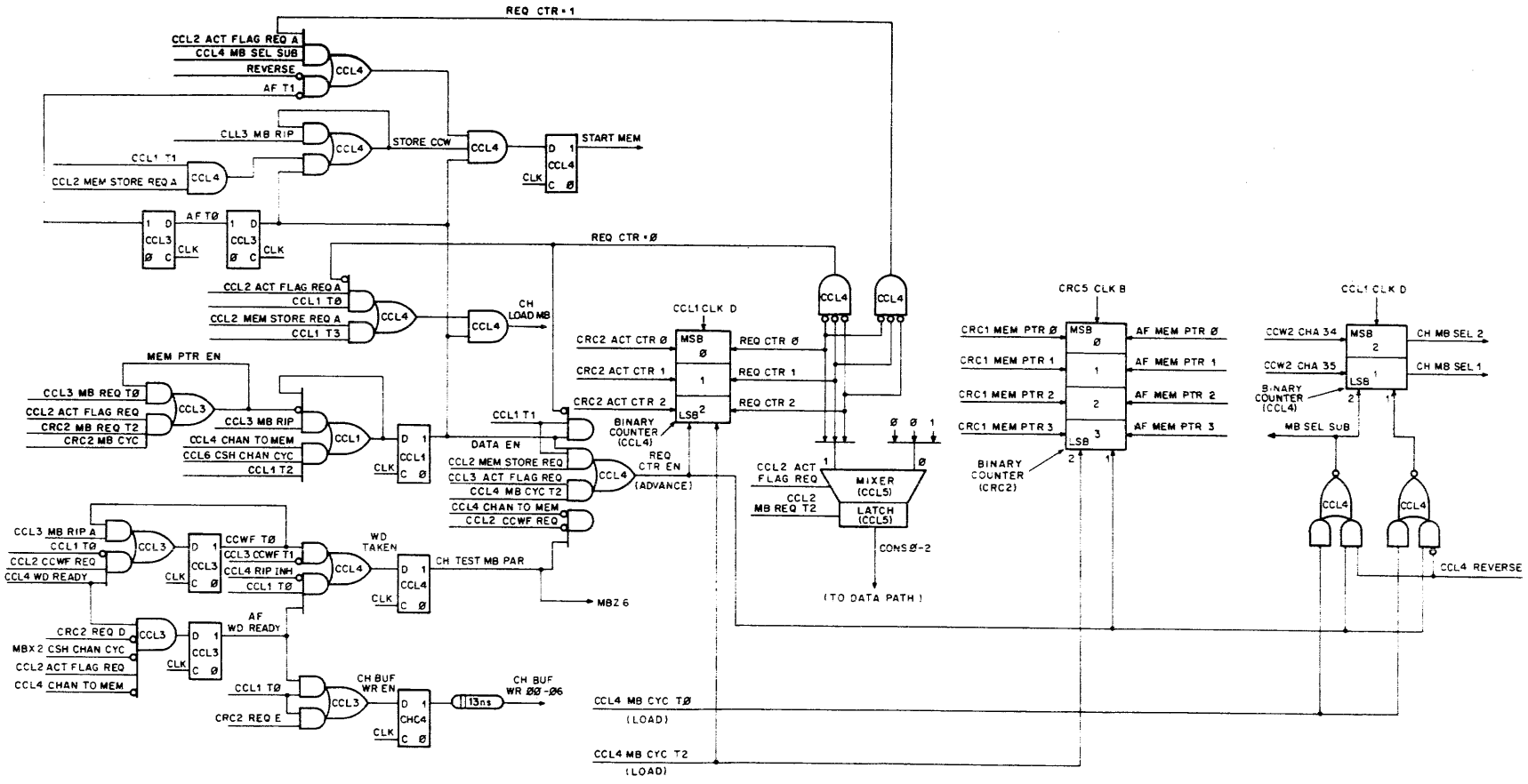


Figure 3-50 MB Request Control Logic, Simplified Logic Diagram

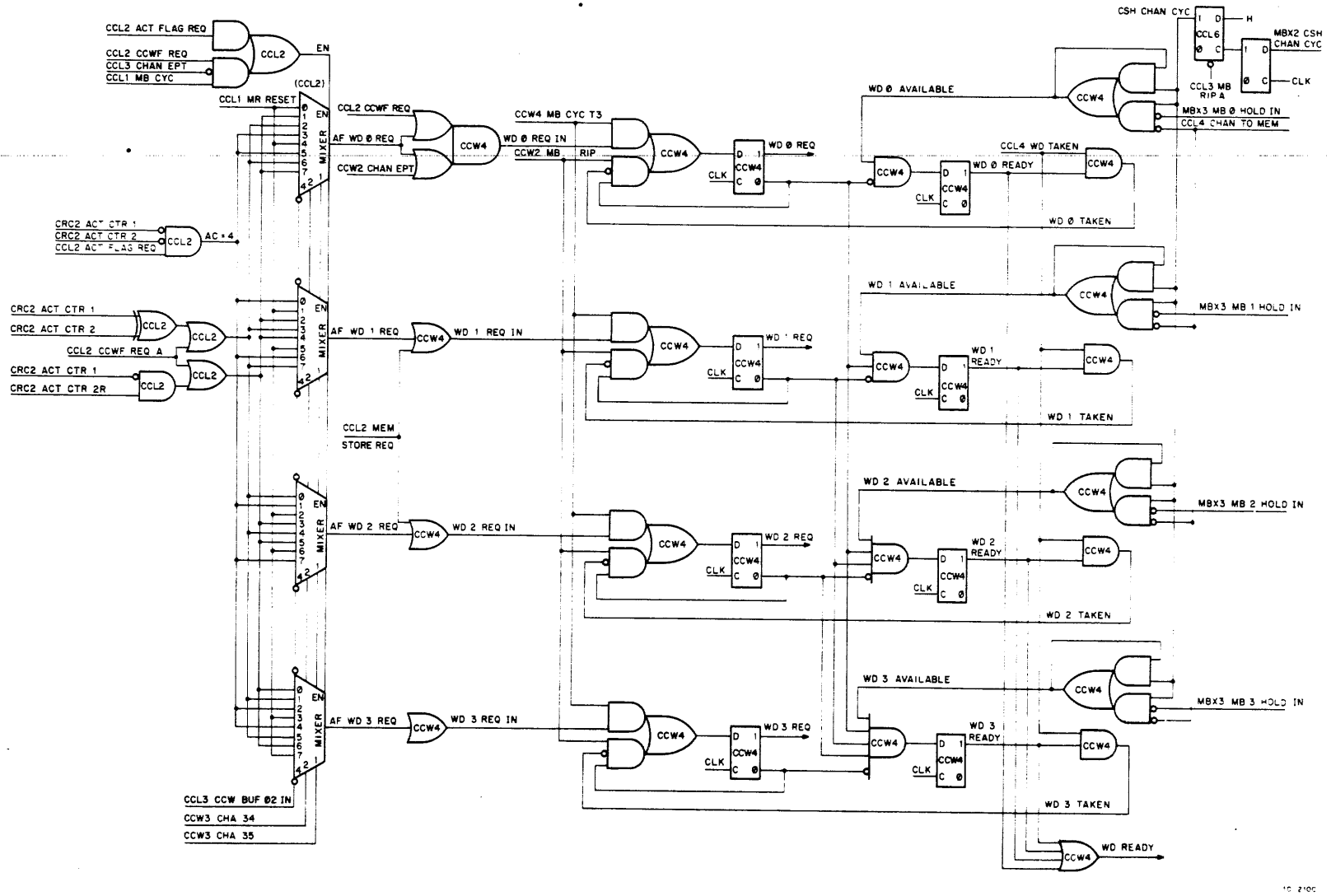


Figure 3-51 Word Request Logic, Simplified Logic Diagram

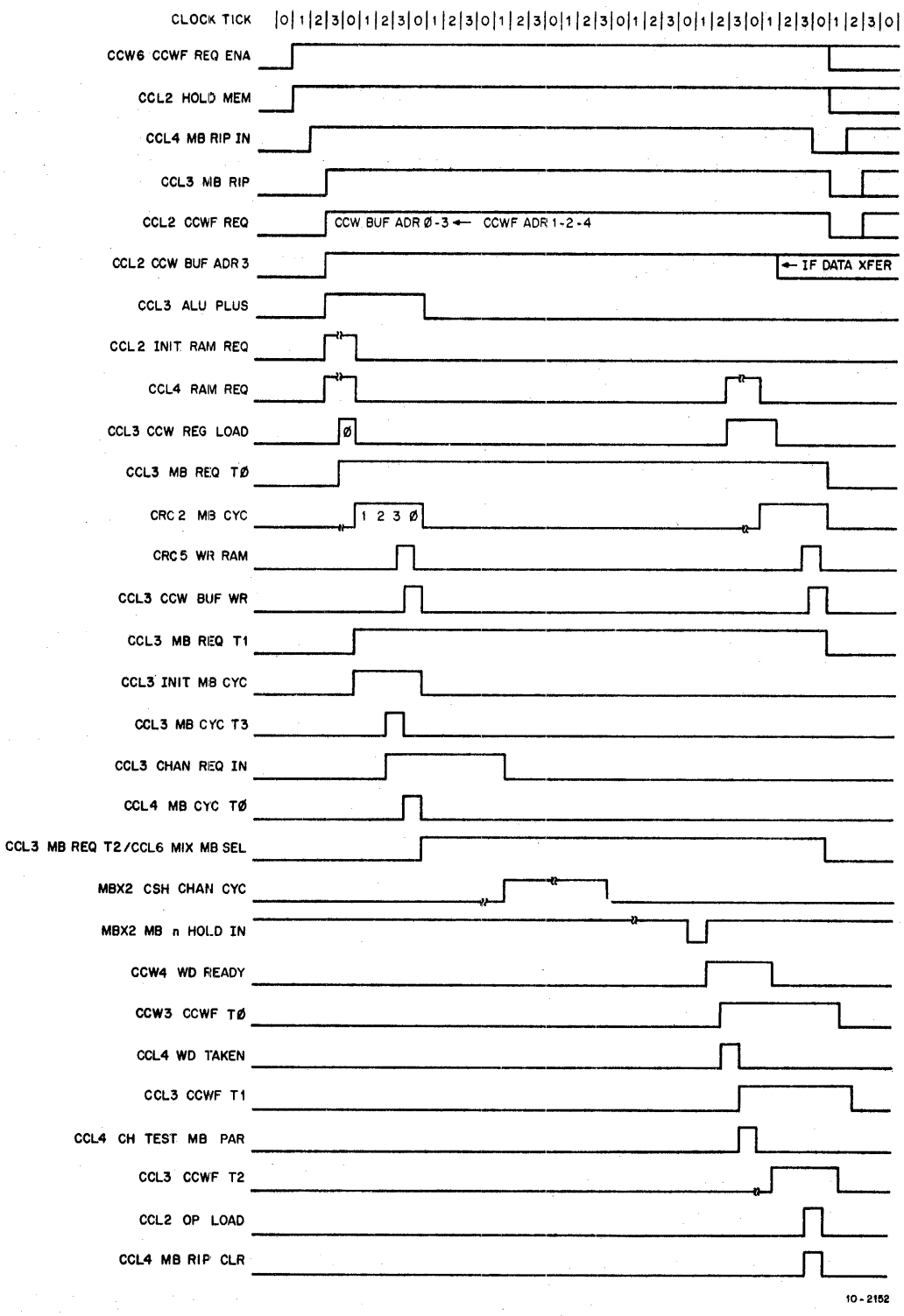


Figure 3-52 CCWF MB Request Timing Diagram

b. An MB request (CCL MB RIP is asserted) to fetch a CCW for the highest priority channel is initiated as soon as the previous MB request is completed.

c. When the MB request is initiated, the CCW request is granted (CCL CCWF REQ is asserted), the CCW BUF ADR is formed (bits 0–2 specify the channel number for which the request is being executed and bit 03 is set to a "one" to select the CLP for that channel), a RAM request is initiated, and the contents of the addressed location of the CCW BUF are loaded into the CCW register.

d. Providing a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, a RAM cycle (MB CYC) is granted and executed to set up the channel request for a cache cycle and update (increment) the CLP in the CCW BUF. (The contents of the CCW register are not changed.)

e. When the RAM cycle is executed, the following operations are performed:

1. CCL CHAN REQ and, if required, CCL CHAN EPT are asserted and the word request logic is set up to specify the word number at CCL MB CYC T3.

2. The status and CLP word in the addressed CCW BUF location is updated (CLP is incremented by one) at CCL MB CYC T0 (CCL CCW BUF WR is asserted).

3. The MB Select (MB SEL) counter is set up (loaded) at CCL MB CYC T0 to point to the MB that will receive the CCW. Bits 34 and 35 of the Channel Address (CHA) in the CCW register specify the word number and consequently the MB that will receive the word.

NOTE

The CCW register contains the physical address that is selected by the PMA when a cache cycle is executed. The request and memory pointer counters are not used for this transfer.

f. The channel control MB request logic then waits until the cache cycle control grants a cache cycle to process the channel request.

g. When a cache channel cycle (CSH CHAN CYC) is executed, a core read cycle is initiated if the word is not in the cache and the channel control word request logic is enabled to detect when the CCW is loaded into an MB. If the word is in the cache, the word is simply transferred to the selected MB.

h. When the word is loaded into an MB by the cache/core control, the word request logic asserts WD READY, which initiates the CCL CCWF T0–T2 timing logic to request a second RAM cycle for writing the CCW into the CCW BUF and for writing the OP code into the control RAM.

i. If a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, the second RAM cycle is granted and executed to write the CCW and op code into the CCW BUF and control RAM, respectively. The CCW is written into location 0 if it is a data transfer CCW, and the right half of location 1 if it is a jump or halt CCW.

j. After the second RAM cycle is executed, CCL MB RIP is cleared to allow another MB request to be initiated.

3.8.8.2 Action Flag (CTOM) Request – The following description details an action flag MB request to transfer data from the CH BUF to main memory. This description applies providing the block transfer is not a zero fill (skip) operation and, in the case of a block transfer from a magtape, the transfer is not a read-reverse operation (Figure 3-53).

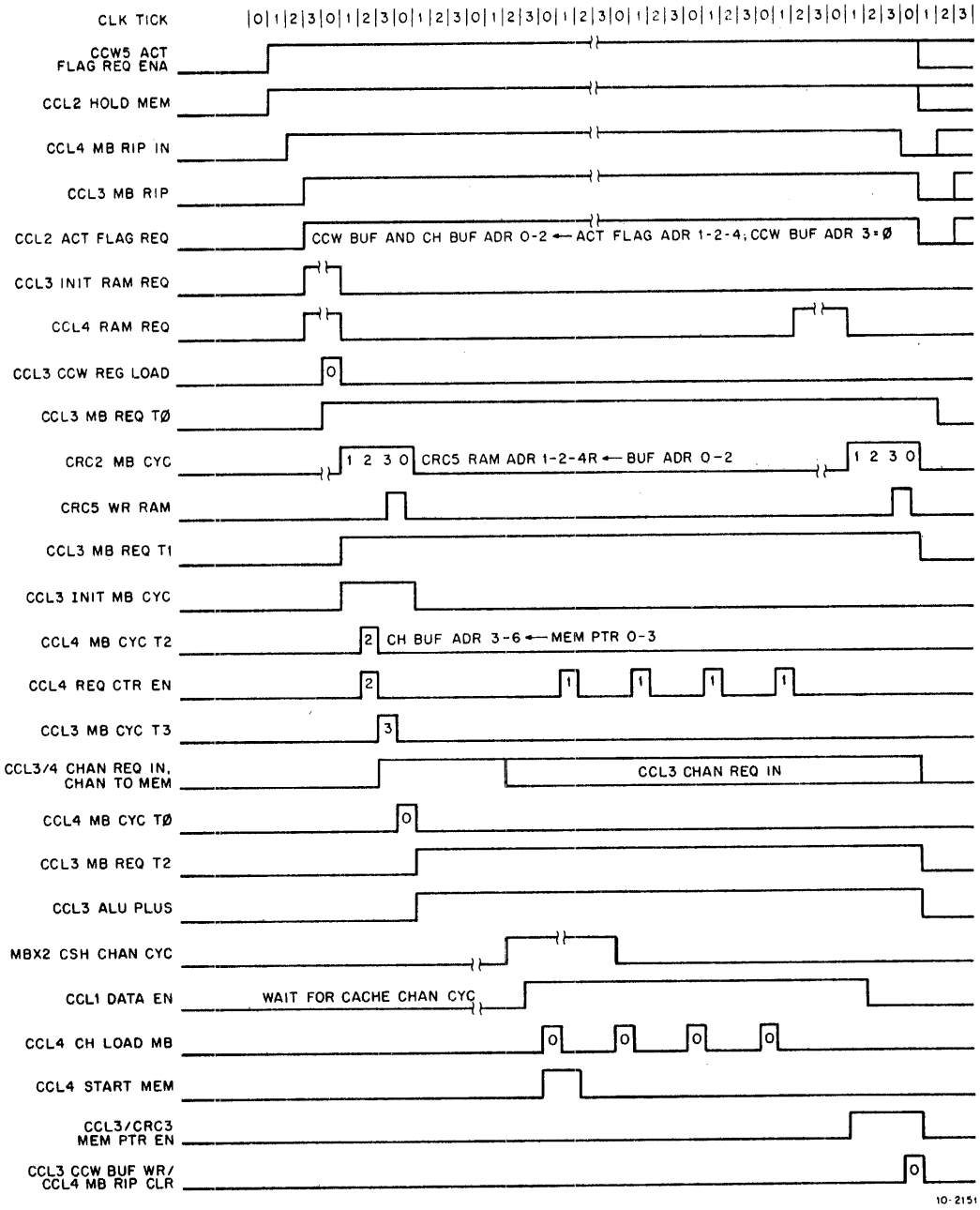


Figure 3-53 Action Flag MB Request (CTOM), Timing Diagram

a. An action flag request is queued when enough words for a given channel have been accumulated in the CH BUF (CRC AF REQ ENA). The action flag arithmetic logic keeps track of the words in the CH BUF (Subsection 3.8.6). This logic asserts CRC AF REQ ENA when enough words have been accumulated.

b. An MB request (CCL MB RIP is asserted) to transfer a group of words (maximum of four) for the highest priority channel is initiated as soon as the previous MB request is completed, providing a CCWF request is not pending.

c. When the MB request is initiated, the action flag request is granted (CCL ACT FLAG REQ is asserted), the CCW BUF address and part of the CH BUF address are formed to address the desired segment of the buffers, a RAM request is initiated, and the contents of the addressed location of the CCW BUF are loaded into the CCW register. Address bits 0–2 of both the CH BUF address and the CCW BUF address specify the channel number for which the request is being executed. Bit 3 of the CCW BUF address is assured to be zero at this time so that the CCW in the CCW BUF is addressed and transferred to the CCW register. The CCW contains the current WC and the current memory address (ADR) of the transfer.

d. Providing a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, a RAM cycle (MB CYC) is granted and executed to set up the channel request for a cache cycle and to form the rest of the CH BUF address (bits 3–6).

e. When the RAM cycle is executed, the following specific operations are performed:

   1. The Request counter (REQ CTR) and the Action Flag Memory Pointer (AF MEM PTR) counters are loaded at CCL MB CYC T2. The Request counter receives the current action count which specifies how many words are to be transferred to memory. The counter is decremented each time a word is transferred to an MB until its content reaches zero. The Memory Pointer counter receives the current memory pointer which specifies the starting location (address) in the CH BUF from which to take the words. As the words are transferred to the MBs, this counter is incremented to point to the next word.

   2. At CCL MB CYC T3, CCL CHAN REQ IN and CCL CHAN TO MEM are asserted and the word request logic is set up to specify the words to be transferred. The number of words to be transferred is a function of the action count and the memory address that was loaded into the CCW register.

   3. The MB Select (MB SEL) counter is set up (loaded with CCW CHA 34 and 35) at CCL MB CYC T0 to point to the MB that will get the first word. As each word is transferred to an MB, the MB Select counter is incremented to point to the next MB.

NOTE

The CCW register contains the physical address that is selected by the PMA when a cache cycle is executed.

f. The channel control MB request logic then waits until the cache control grants a cache cycle to process the channel request.

g. When a cache channel cycle is executed (CSH CHAN CYC is asserted), the words are transferred from the CH BUF to the MBs one at a time (CCL CH LOAD MB is asserted for each word), and CCL START MEM is asserted to initiate a memory write cycle. Each time a word is transferred, the valid words in the cache are invalidated, the request counter is decremented, and the AF MEM PTR and MB SEL counters are incremented. (CCL REQ CTR EN is asserted.)

h. When the Request counter reaches zero, a second RAM request is initiated to update the memory pointer in the control RAM and the CCW in the CCW BUF.

i. If a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, the second RAM cycle is granted and executed to update the memory pointer and the CCW as follows:

   1. The pointer is updated by adding the action count to the pointer and writing the result back into the control RAM (CRC WR RAM).

   2. The CCW is updated by subtracting the action count from the WC, adding the action count to the ADR, and writing the result back into the CCW BUF. (CCL CCW BUF WR).

j. After the second RAM cycle is executed, CCL MB RIP is cleared to allow another MB request to be initiated.

If the block transfer is a zero fill operation, a cache and core cycle will not be requested by the channel control. The words that were placed in the CH BUF are simply ignored. The request for a second RAM cycle will be made directly after the first RAM cycle has been executed.

If the block transfer is from a magtape and it is a read-reverse operation, the words are transferred to the MBs in reverse order so that they can be written into main memory in the correct order. In addition, the address of the CCW is updated by subtracting the action count from the address, instead of adding it as described in step i (2) above. Also, CCL START MEM is not asserted until all the words have been transferred to the MBs.

3.8.8.3 Action Flag (NOT CTOM) Request – The following description details an action flag MB request to transfer data from main memory to the CH BUF. This description applies providing the block transfer is not a zero fill operation (Figure 3-54).

a. An action flag request is queued when enough empty locations have been accumulated in the CH BUF (CRC AF REQ ENA). The action flag arithmetic logic keeps track of the empty locations in the CH BUF (Subsection 3.8.6).

b. An MB request (MB RIP is asserted) to transfer a group of words (maximum of four) for the highest priority channel is initiated as soon as the previous MB request is completed providing a CCWF request in not pending.

c. When the MB request is initiated, the action flag request is granted (CCL ACT FLAG REQ is asserted), the CCW BUF address and part of the CH BUF address are formed to address the desired segments of the buffers, a RAM request is initiated, and the contents of the addressed location of the CCW BUF are loaded into the CCW register. Address bits 0–2 of both the CH BUF address and the CCW BUF address specify the channel number for which the request is being executed. Bit 3 of the CCW BUF address is assured to be zero at this time so that the CCW in the CCW BUF is addressed and transferred to the CCW register.

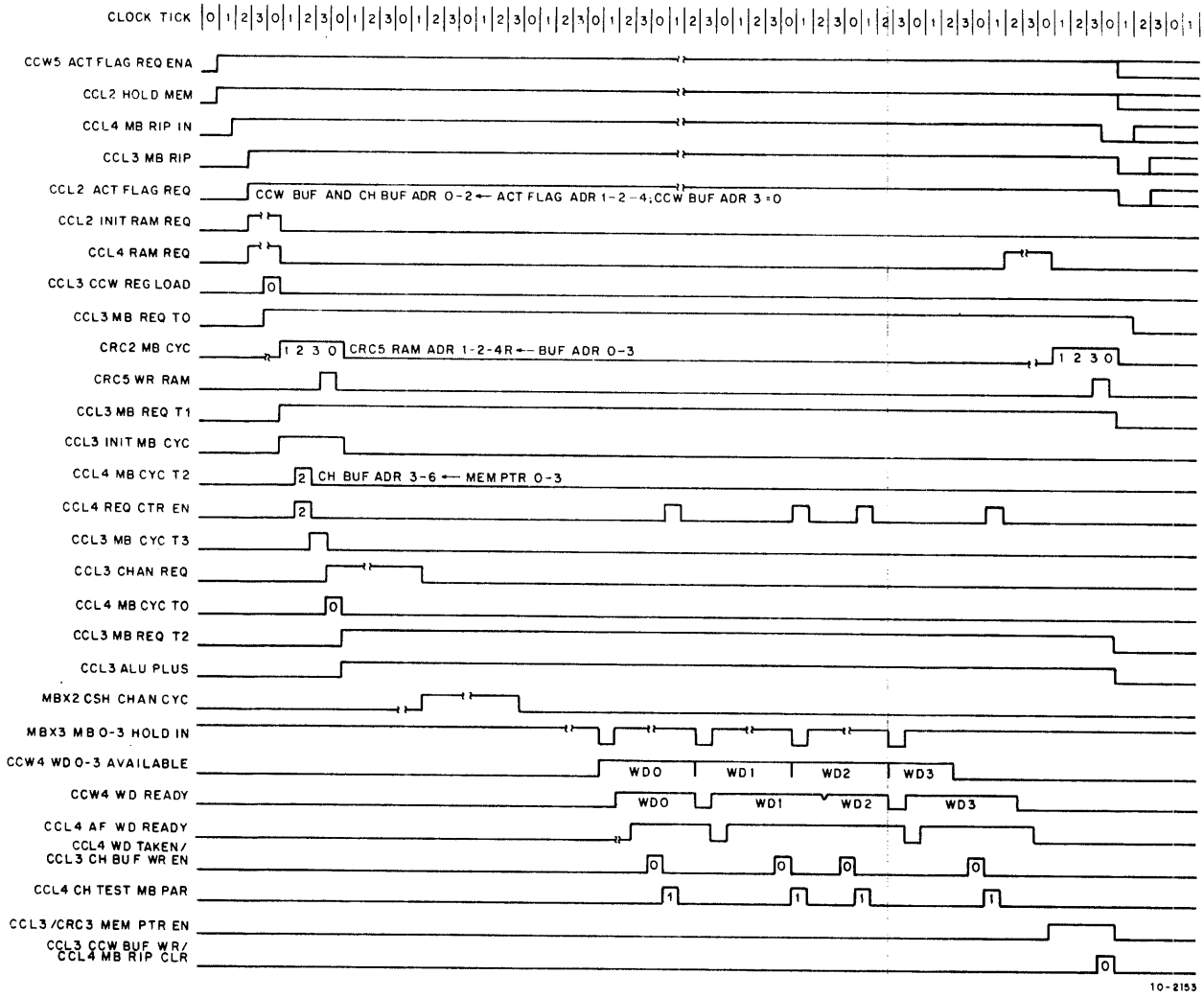


Figure 3-54 Action Flag MB Request (NOT CTOM), Timing Diagram

d. Providing a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, a RAM cycle (MB CYC) is executed to set up the channel request for a cache cycle and to form the rest of the CH BUF address (bits 3–6).

e. When the RAM cycle is executed, the following specific operations are performed:

1. The Request counter (REQ CTR) and the Action Flag Memory Pointer (AF MEM PTR) counter are loaded at CCL MB CYC T2. The Request counter receives the current action count which specifies how many words are to be transferred from memory. The counter is decremented each time a word is transferred to the CH BUF until its content reaches zero. The Memory Pointer counter receives the current memory pointer, which specifies the starting location (address) in the CH BUF in which to write the words. As the words are written into the CH BUF, the counter is incremented to point to the next empty location.

2. At CCL MB CYC T3, CCL CHAN REQ is asserted and the word request counter is set up to specify the words to be fetched. The number of words to be fetched is a function of the action count and the memory address that was loaded into the CCW register.

3. The MB SEL counter is set up (loaded with CCW CHA 34 and 35) at CCL MB CYC T1 to point to the MB that will receive the first word. As each word is transferred to the CH BUF, the MB Select counter is incremented to point to the next MB.

NOTE

The CCW register contains the physical address which is selected by the PMA when a cache cycle is executed.

f. The MB request logic then waits until the cache control grants a cache cycle to process the channel request.

g. When a cache cycle is executed (CSH CHAN CYC is asserted), a core read cycle is initiated if all the words are not in the cache and the channel control word request logic is enabled to detect when the words are loaded into the MBs. Any words in the cache are simply transferred to the corresponding MBs.

h. When the lowest number word that was requested is loaded into the corresponding MB by the cache/core control, the word request logic asserts CCW WD READY. If a CBUS REQUEST is not being executed (CH REQ D is not asserted), the word is transferred from the MB to the CH BUF, the Request counter is decremented, and the Memory Pointer and MB Select counters are incremented (CH BUF WR 00–06 and CCL REQ CTR EN are asserted). This operation is repeated for each word until all requested words are transferred, at which time the Request counter will contain zero.

i. When the Request counter reaches zero, a request for a second RAM cycle (RAM REQ) is initiated to update the memory pointer in the control RAM and the CCW in the CCW BUF.

j. If a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, the second RAM cycle is granted and executed to update the memory pointer and the CCW as follows:

   1. The pointer is updated by adding the action count to the pointer and writing the result back into the control RAM.

   2. The CCW is updated by subtracting the action count from the WC, adding the action count to the ADR, and writing the result back into the CCW BUF.

k. After the second RAM cycle is executed, CCL MB RIP is cleared to allow another MB request to be initiated.

If the block transfer is a zero fill operation, the channel request will be initiated to fetch the four zero fill words from the EPT. Four locations for all channels are reserved in the EPT (starting at location 60) for storing the zero fill words.

3.8.8.4 Memory Store Request – The following description details a memory store MB request to transfer the current CCW and the status and CLP word, which are maintained in the CCW BUF, to the EPT in main memory (Figure 3-55).

a. A memory store request is queued when:

1. A store operation is specified by the RH20 Massbus controller at the end of a block transfer. (Both CBUS STORE and DONE are asserted.)

2. A memory error or a channel error is sensed while the block transfer is being executed.

b. An MB request (CCL MB RIP is asserted) to store the two words in the CCW BUF, for the highest priority channel that has a request pending, is initiated as soon as the previous MB request is completed, providing a CCWF request and an action flag request are not pending.

c. When the MB request is initiated, the memory store request is granted (CCL MEM STORE REQ is asserted), the CCW BUF ADR is formed (bits 0–2 specify the channel number for which the request is being executed and bit 3 is set to a "one" to select the CLP for that channel), a RAM request is initiated, and the contents of the addressed location of the CCW BUF are loaded into the CCW register. These bits, along with the status bits from the control RAM, which form the first word to be transferred to an MB (word 1 in CCW BUF) for the channel the request is being executed, are required in computing the parity bit so that the parity check performed on the contents of the MB is consistent with the data word that was transferred to the MB.

NOTE

The CCW register is loaded only to compute the MB data parity; its contents are not required in forming the PMA because the address is formed exclusive of the contents of the CCW register. Besides loading the CCW register and computing a parity bit consistent with the word transferred to the MB, the CCW BUF is also updated to reflect the status bits that are maintained by the control RAM. This word, therefore, is made available so that it can be read under diagnostic control for diagnostic purposes.

d. Providing a higher priority request (CBUS REQUEST or CBUS CONTR REQ) is not pending, a RAM cycle (MB CYC) is granted and executed to set up the channel request for a Cache cycle and to select the appropriate channel status bits which comprise bits 00–13 of the word to be transferred to the MB. These bits will also be written into the appropriate CCW BUF location.

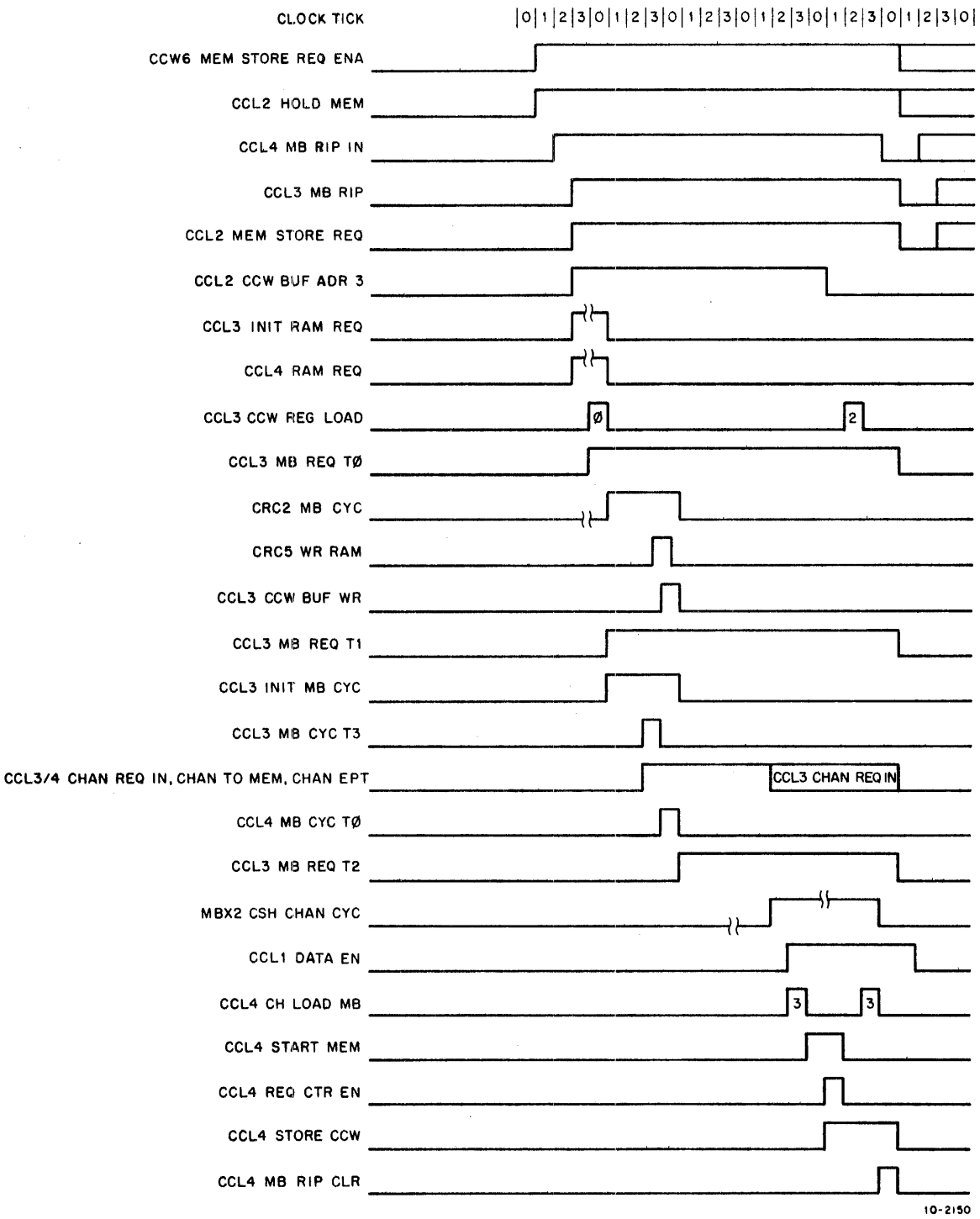


Figure 3-55 Memory Store MB Request, Timing Diagram

e. When the RAM cycle is executed, the following specific operations are performed:

1. At CCL MB CYC T3, CCL CHAN REQ IN, CCL CHAN TO MEM, and CCL CHAN EPT are asserted and the word request logic is set up to specify the words (word 1 and 2) to be transferred.

2. The status and CLP word in the addressed CCW BUF location is updated (the status bits are written) at CCL MB CYC T1 (CCL CCW BUF WR is asserted).

3. The MB SEL counter is set up (loaded) at CCL MB CYC T1 to point to the MB (MB1) that will receive the status and CLP word. Bit 35 of the channel address (CHA) is forced to a "one" for this operation to ensure that the MB SEL counter is set up correctly.

NOTE

The contents of the CCW register are not used as the memory address for the memory store operation. Instead, the CHA address is forced to point to the correct location in the EPT page. When a cache cycle is granted, the PMA supplies the base address for the EPT (contents of EBR). In addition, the Request and Memory Pointer counters are not used for the memory store operation.

f. The channel control MB request logic then waits until the cache control grants a cache cycle to process the channel request.

g. When a cache channel cycle is executed (CSH CHAN CYC is asserted), the first word is transferred from the CCW register to MB1 (CCL CH LOAD MB is asserted) and CCL START MEM is asserted to initiate a memory write cycle. The second word is then loaded into the CCW register to compute its parity and then loaded into MB2. If any valid words are found in the cache, they are invalidated. After the first word is transferred, CCW BUF ADR bit 3 is cleared (CCL STORE CCW is asserted) to point to the next word (CCW) and the MB SEL counter is incremented to point to MB2 (CCL REQ CTR EN is asserted).

h. After the second word is loaded into MB2, CCL MB RIP is cleared to allow another MB request to be initiated.

3.8.8.5 Error Request – The channel control MB request logic must be guarded against potential memory errors while an MB request is being executed. When an MB Request is in progress (MB RIP) and a cache channel cycle is granted in response to the channel request, the Error Address register is loaded to preserve the address of the channel for which the request is being executed. If a memory error is detected while the channel request (CCWF request only) is being executed, the CCL MEM ERROR LATCH is set (Figure 3-56). Then, when CCL MB RIP is cleared on completion of the request, CCL ERR REQ is asserted to initiate a RAM cycle to update the control RAM error bits and to set the appropriate bit in the store request queue using the address that was preserved by the Error Address Register. Another MB request will not be started until a RAM cycle for the error request is executed. After the RAM cycle is executed, the CCL MEM ERR LATCH is cleared.

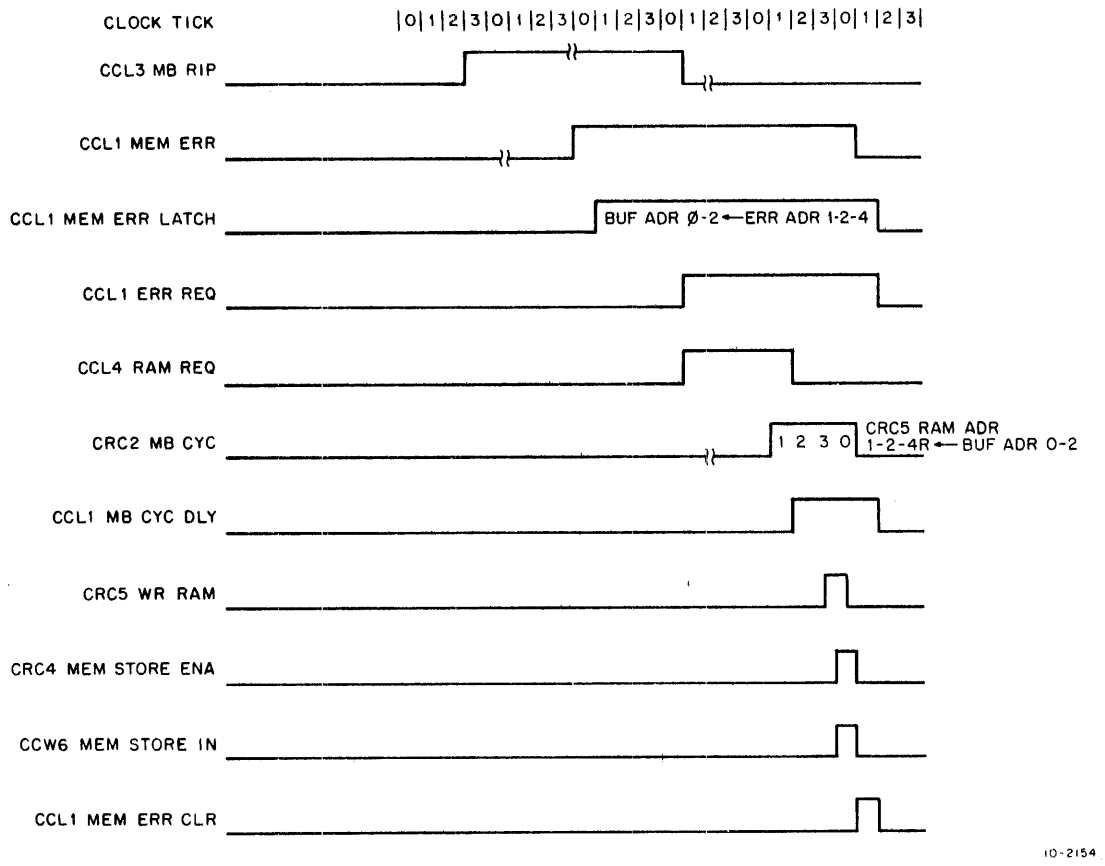


Figure 3-56 Memory Error MB Request, Timing Diagram

APPENDIX A
ABBREVIATIONS AND MNEMONICS

	A
AC	Accumulator
AC	Action Count
ACKN	Acknowledge
ACT	Action
AD	Adder
ADA	Adder A
ADB	Adder B
ADR	Address
ADX	Adder Extension
AF	Action Flag
ALT	Alternate
ALU	Arithmetic Logic Unit
APR	Arithmetic Processor Register
AR	Arithmetic Register
ARL	Arithmetic Register Left
ARM	Arithmetic Register Mixer
ARMM	Arithmetic Register Mixer Mixer
ARR	Arithmetic Register Right
ARX	Arithmetic Register Extension
ARXL	Arithmetic Register Extension Left
ARXM	Arithmetic Register Extension Mixer
ARXR	Arithmetic Register Extension Right
	B
BOOLE	Boolean
BR	Buffer Register
BRK	Break
BRX	Buffer Register Extension
BUF	Buffer
	C
CAM	Cache Address Mixer
CBUS	Channel Bus
CCA	Cache Clearer Address
CCL	Channel Control Logic
CCW	Channel Command Word
CCWF	Channel Command Word Fetch
C DIR P	Cache Directory Parity
CG	Carry Generate
CH	Channel
CHA	Channel Address
CHAN	Channel
CHK	Check
CHX	Cache Extension
CLK	Clock
CLR	Clear
COMP	Complete
CON	Control
COND	Condition
CONS	Constant
CONTR	Control
CP	Carry Propagate
CP	Central Processor
CPU	Central Processing Unit
CR	Control RAM
CRA	Control RAM Address
CRAM	Control RAM Address Mixer
CRC	Channel RAM Control
CRM	Control RAM
CRY	Carry
CS	Controller Select
CSH	Cache
CTL	Control
CTOM	Controller-to-Memory or Cache-to-Memory
CTR	Counter
CWSX	Called With Special Execute
CYC	Cycle

	D
D	Data
DAT	Data
DIAG	Diagnostic
DIF	Difference
DIR	Directory
DIS	Disable
DISP	Dispatch
DIV	Divide
DRAM	Dispatch RAM
	E
E	EBox Cyc
E to T	ECL to TTL
EBR	Executive Base Register
EBUS	Execution Bus
ECL	Emitter-Coupled Logic
EDP	EBox Data Path
EN	Enable
ENA	Enable
ERR	Error
ERA	Error Address
EPT	Executive Process Table
EX	Extension
EXP	Exponent
EXT	External
EXT TRA REC	External Transfer Receiver
	F
F	Function
FE	Floating Exponent
FE	Front End
FLG	Flag
FM	Fast memory
FOV	Floating Overflow
FPD	First Part Done
FPD	Floating Point Divide
FUNC	Function
FXU	Floating Exponent Underflow
	G, H
G	Gated
GE	Greater or Equal
GEN	Generate
H	High
	I
IN	Input
INC	Increment
INH	Inhibit
INSTR	Instruction
INT	Internal
INTR	Interrupt
INVAL	Invalid
IOT	Input/Output Transfer
IR	Instruction Register
	J, K, L
J	Jump
L	Low
LRU	Least Recently Used
	M
MB	Memory Buffer
MBC	MBox Control
MBX	MBox Control
MBZ	MBox Control
MCL	Memory Control
MEM	Memory
MHz	Megahertz
MIX	Mixer
MQ	Multiplier Quotient
MQM	Multiplier Quotient Mixer
MR	Master
MRU	Most Recently Used
MTR	Meter
	N
NICOND	Next Instruction Condition
NXM	Non-Existent Memory
NXT	Next
	O
OK	011 Korrect
OP	Operation (code)
OVN	Overrun
	P
PA	Physical Address
PAG	Pager
PAR	Parity
PC	Program Counter
PCF#	Previous Context Flags from Number
PCP	Previous Context Public
PC	Program Counter
PERF	Performance
PF	Page Fault
PGRF	Page Refill
PI	Priority Interrupt
PIA	Priority Interrupt Assignment
PIH	Priority Interrupt Hold
PMA	Physical Memory Address
PMA	Physical Memory Address Selector
PREV	Previous
PT	Page Table/Process Table
PTR	Pointer
PWR	Power

	R
RAM	Random Access Memory
RD	Read
RE	Receive ECL
REC	Receive
REF	Reference
REG	Register
REL	Release
REQ	Request
RES	Reset
RESP	Response
RET	Return
RIP	Request In Progress
RQ	Request
	S
S ADR P	Storage Address Parity
SBR	Subroutine
SBUS	Storage Bus
SC	Shift Count
SCAD	Shift Count Adder
SCADA	Shift Count Adder A
SCADB	Shift Count Adder B
SCD	Shift Count Adder
SCM	Shift Count Mixer
SEL	Select
SH	Shifter
SHRT	Shift Right
SIM	Simulate
SP	Special
SPEC	Special
SR	State Register
ST	Start
SYNC	Synchronize
	T, U
T to E	TTL to ECL
TE	Transmit ECL
T	Time
TRA	Transfer
TTL	Transistor-Transistor Logic
UBR	User Base Register
UCODE	Microcode
	V, W, X, Y, Z
VAL	Valid
VMA	Virtual Memory Address
XFER	Transfer
XR	Index Register
WARN	Warning
WC	Word Count
WD	Word
WR	Write

INDEX

R

MBOX STORAGE CONTROLLER
UNIT DESCRIPTION
EK-MBOX-UD-004

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgment is it complete, accurate, well organized, well written, etc.? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults do you find with the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

Would you please indicate any factual errors you have found. _____

_____

_____

_____

Please describe your position. _____

Name _____ Organization _____

Street _____ Department _____

City _____ State _____ Zip or Country _____

Fold Here

Do Not Tear - Fold Here and Staple

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

Digital Equipment Corporation
Technical Documentation Department
Maynard, Massachusetts 01754