V-SERIES SPECIFICATIONS
DISTRIBUTION LIST
EMSG - MISSION VIEJO

SPECIFICATION NO: 1993 5220 SPECIFICATION TITLE: MEMORY CONTROL & CACHE MOD

REVISION LETTER: D DATE OF ISSUE: 06-23-89

MISSION VIEJO, CA.

D.	ABALOS	MS 105
J.	BARRETT	MS 310
F.	BEUGER	MS 380
R.	BIGHAM	MS 195
D.	BRITTLE	MS 275
R.	CAIRNS	MS 235
C.	DOOLING	MS 165
S.	ESCOBAR	MS 115
J.	GEORGE	MS 135
B.	GOULD	MS 175
F.	HAAS	
D.	HANSON	MS 205
R.	HARRIS	MS 135
A.	HATCH	MS 150
S.	HERMAN	MS 225
J.	HORKY	MS 170
M.	JACOBS	MS 165
M.	KALLAL	MS 105
C.	KING	MS 225
W.	LESH	RSM115
C.	LEUTHEN	MS 170
D.	MACMILLAN	MS 285
J.	MARTIN	MS 195
R.	MATHEWS	MS 225
V.	MCCLELLAND	MS 165
R.	MENDOZA	MS 135
J.	MOLASH	MS 135
B.	MORRIS	MS 135
W.	MURRAY	MS 165
M.	NAKAMOTO	MS 195
M.	OGRADY	MS 380
S.	PATRICK	MS 275
B.	POTTER	MS 135
R.	QUAYLE	MS 135

CONTINUED - MISSION VIEJO, CA.

J.	RUCKER	MS 520
J.	SINGLETON	MS 205
H.	SOHL	MS 165
S.	SOLOMON	MS 135
T.	STERNAL	MS 165
D.	STONE	MS 285
W.	TERRY	MS 215
T.	VECCHIARELLI	MS275 (2 COPIES)
D.	WALSH	MS 170
R.	WARREN	MS 135
P.	WORCESTER	MS 135
J.	YETTER	MS 165

PASADENA, CA.

J.	BURTON	
J.	COLBURN	
N.	FLEMING	
R.	KATHOL	
C.	NUSE	
W.	TREINEN	CC 6130
J.	VANZWALUWENBURG	

TREDYFFRIN, PA.

A.	WORKS	

For changes in this distribution list, contact D. Patten, M.V. Release and Configuration.

Company Confidential

1993 5220

MEMORY CONTROL AND CACHE MODULE

ENGINEERING DESIGN SPECIFICATION

UNISYS

GENERAL SYSTEMS GROUP

ORANGE COUNTY

UNISYS

1993 5220

V500 MEMORY CONTROL AND CACHE MODULE

Entry/ Medium Systems Group
PASADENA DEVELOPMENT CENTER

ISYS RESTRICTED

ENGINEERING DESIGN SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED ADDED DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	ECN 50426	Initial Issue JUNE 5, 1986	D. Castle L. Simpson	D. Welbourn H. Li R. Young J. Chen D. Lau
B	ECN 52003 02-24-88	Updated to expand the description of Cache per REE 099715.	L. Simpson Y. Chang	M. Ho R. Young A. Gomez H. Li N. Fleming
C	ECN 95797	UPDATED TO INCLUDE PAGES 122A, 123A, 127A, 204A, 205A ADDED TO THE END OF THIS DOCUMENT.		

EDS25

UNISYS

1993 5220

V500 MEMORY CONTROL AND CACHE MODULE

Entry/ Medium Systems Group
PASADENA DEVELOPMENT CENTER

UNISYS RESTRICTED

ENGINEERING DESIGN SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED, ADDED, DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	ECN 50426	Initial Issue JUNE 5, 1986	D. Castle L. Simpson	D. Welbourn H. Li R. Young J. Chen D. Lau
B	ECN 52003 02-24-88	Updated to expand the description of Cache per REE 099715.	L. Simpson Y. Chang	M. Ho R. Young A. Gomez H. Li N. Fleming

EDS25

UNISYS
GENERAL SYSTEMS GROUP
ORANGE COUNTY

1993 5220

MEMORY CONTROL AND CACHE MODULE

COMPANY CONFIDENTIAL

ENGINEERING DESIGN SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED ADDED. DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
D	6/23/89	To correct errors in header, footer and format; folded in changes made in Rev. C into text of document.	S. Solomon	6/23/89 M. Jacobs

UNISYS CORPORATION
GENERAL SYSTEMS GROUP
MISSION VIEJO

TABLE OF CONTENTS

TABLE OF ILLUSTRATIONS

1 PURPOSE

This specification defines the memory control and cache module which in conjunction with the memory storage structure performs the main memory function for the V500 System.

2 APPLICABLE DOCUMENTS

1257 6005	Corporate Environmental Standards
A1, A14, A16	Corporate Technical Standards
1995 5301	V500 Circuit Rules
xxxx xxxx	V500 Element Rules
xxxx xxxx	V500 Chip Chart
1997 5390	V Series Instruction Set
1993 5279	V500 Architecture
1993 5163	V500 System
1993 5170	V500 System Index
1993 5204	V500 Execute Module Engineering Design Spec
1993 5212	V500 Fetch Module Engineering Design Spec
1993 5238	V500 Memory Data Card SDS
1993 5253	I/O Memory Concentrator Eng Design Spec
1993 5329	Data Transfer Module Engineering Design Spec
1993 5337	Fault Detection Guidelines
1993 5295	System Maintenance Controller
1993 5303	V500 Maintenance Subsystem
1998 5266	Maintenance Users Guide

3 OVERVIEW

The memory control and cache module provides the read and write mechanism for the V500 processor to 500 Mbytes of main memory in a multi-tasking, multi-processor environment. The main memory is divided into process dependent areas which are scheduled by the operating system. Each read or write request to memory specifies an area-relative address and a base indicant which selects which of the memory areas to access.

The memory control and cache module is a link in the memory hierarchy extending from processor scratch-pad registers outwards through main memory, to disks, then tape libraries.

The memory control is split into 2 sections which provide the interfaces between the processor, I/O, cache, and main memory. The first section is an interface between the processor and the cache which is optimized to provide 10 digit operands in a hardware independent format. The second section is an interface between cache and main memory which is optimized for the block-level transactions of cache.

The central portion of the memory control and cache module is the cache memory. The V500 cache memory is organized in 40 digit blocks. There are 2 blocks of 1024 sets in the cache memory. This provides 40,960 bytes of cache memory.

3 OVERVIEW (Continued)

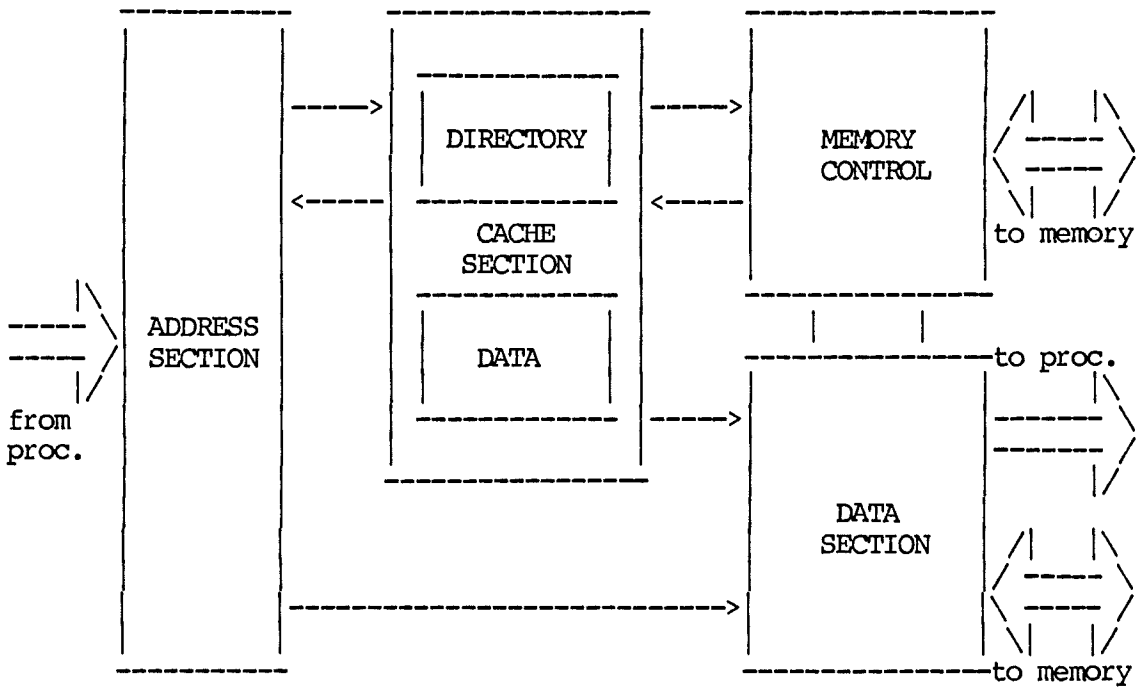


Fig. 3-1 Block Diagram of Memory Control and Cache Module

The memory control and cache module contains 4 separate sections. These 4 sections are: address, memory control, data, and cache. This specification describes the operation and interfaces of the memory control and cache module, then each of these sections in detail.

3 OVERVIEW (Continued)

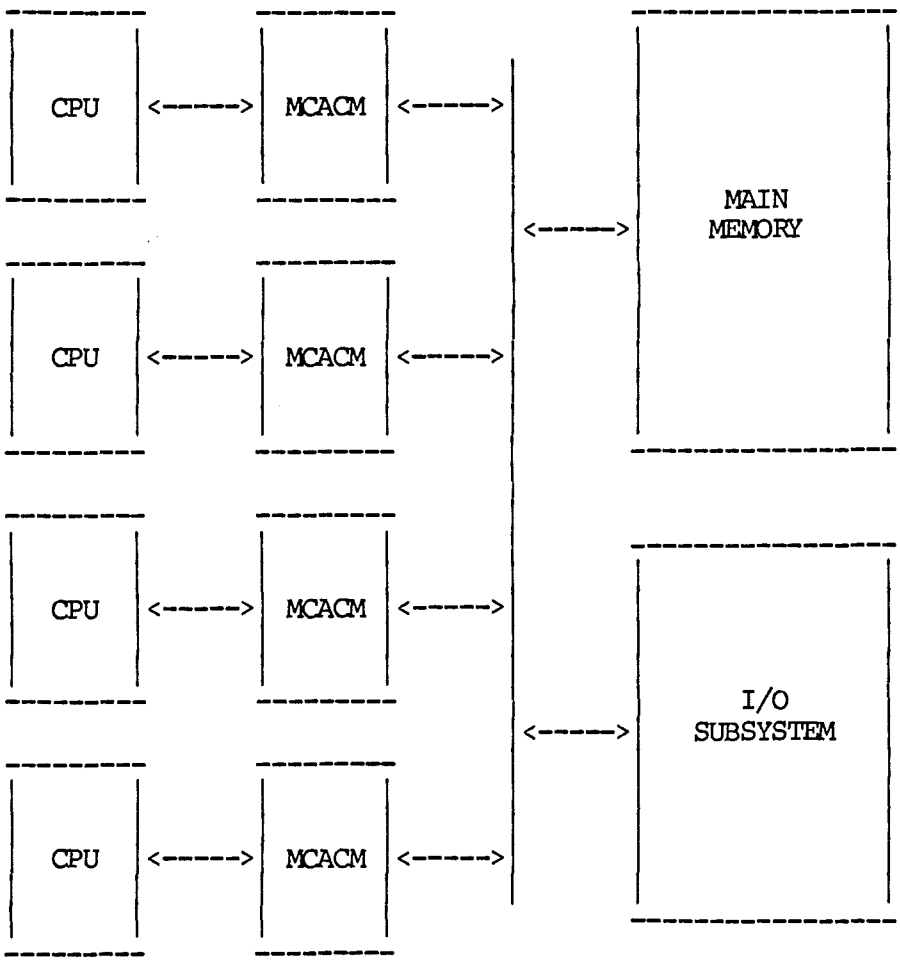


Fig. 3-2 V500 Main Memory and Cache Structure

3.1 COMMANDS

The memory control and cache module provides an interface between the processor and main memory. The MCACM does memory reads and writes for the processor. The MCACM also does the read-with-lock to arbitrate software interlocks between processors in a system.

The memory control and cache module maintains a set of base and limit tables. These tables contain the base and limit addresses of memory areas. Each memory request must specify which memory area to use. The memory request address is then added to the appropriate base address to obtain the absolute memory address. The absolute memory address is then compared to the limit address to check for limit errors.

Whenever a memory request causes or encounters an error, an error flag is returned to the requesting processor module. A subsequent read error report returns the physical memory address and information for the MCP error log.

A special command is also provided which allows the processor to communicate with the I/O subsystem. This communication is done on the main memory address and data buses through the I/O memory concentrators. The I/O memory concentrators accept a subset of the main memory commands across the memory interface. The I/O memory concentrator acts like a logical memory data card.

There are certain commands which the memory control ignores. These No Operation commands are provided to allow non-standard communications between processor modules. An example is a process switch wherein the execute module needs to redirect the fetch module into a new code stream. This implementation was chosen because additional backplane buses are expensive and this type of communication is infrequent.

There are 4 pipe flush commands. These flush commands cause the memory control and cache module to abort any outstanding read requests. These commands are used to reduce the penalty of pre-fetching operands and instructions when a branch prediction is incorrect. All 4 flush commands are treated the same by MCACM but not by

the processor modules.

3.2 CACHE

The memory control contains a 81,920 digit cache memory. The cache memory is organized as 1024 of 2 block sets of 40 digits. The cache is controlled with a distributed write-back algorithm which simplifies multiple processor interdependencies.

The cache memory utilizes a spy mechanism to monitor the main memory address bus. When a read from another processor is a hit into local cache and the cache data has been modified, then the cache signals the memory module to disable its outputs and the cache supplies the data, instead of the memory module. The techniques used to insure consistency are discussed later.

3.3 INTERFACES

The MCACM interfaces to all three major elements of the V500 processor: the processor modules, the main memory structure, and the I/O subsystem. Each of these interfaces have different requirements, accounting for most of the complexity of the memory control and cache module.

The processor to MCACM data interface contains the BCD address and write data (if any) as well as other relevant information concerning the processor's request. This bus is called the address and write bus (or AWBUS). The AWBUS access arbitration is distributed; each requestor determines if it has the highest priority. This technique is fast enough to allow arbitration and data transfer on the same clock cycle.

The MCACM to processor return interface is much simpler than the AWBUS. Only read data and an error vector are returned on this bus. The processor modules determine the destination by analyzing a tag field which is sent one clock cycle prior to the data. This allows the processor time to decide where to latch the data. A cache miss, which is an exception, can result in not having data to send for the tag that was already sent. Therefore, a data valid signal is sent to inform the processor module which bus transactions are valid.

3.3 INTERFACES (Continued)

The MCACM to main memory address and data interface is quite complex because of the long time delay for signals which must cross between both cabinets of a multi-processor system. The memory address bus and the memory data bus are both bidirectional two-clock buses. The bus protocol is relatively simple, however, data is returned a certain number of clocks after a cycle has been requested.

Communication between the V500 processor and the I/O subsystem is conducted through the MCACM. This communication occurs over the main memory address and data bus; the IOMC acts like a memory data card. The I/O Translator and MCP also communicate through data structures in main memory which do not use this path.

4 FUNCTIONS

The memory control and cache module responds to commands from the processor. The command field is part of the AWBUS, which is described in Section 5.1. Figure 4-1 is a numeric list of the memory control and cache module commands.

Code	Command	Code	Command
00 -	Reserved	10 -	Write PC
01 -	Read Data	11 -	Read PC
02 -	Write I/O	12 -	Read Error Address
03 -	Read I/O	13 -	Read Error Report
04 -	Test Error Detection	14 -	Reserved
05 -	Read Base Table	15 -	Reserved
06 -	Reserved	16 -	Read with Lock
07 -	Read Limit Table	17 -	Reserved
08 -	No Operation 0	18 -	Write-Back
09 -	No Operation 1	19 -	Write Base Table
0A -	No Operation 2	1A -	Reserved
0B -	No Operation 3	1B -	Write Limit Table
0C -	Flush 0	1C -	Reserved
0D -	Flush 1	1D -	Write Data
0E -	Flush 2	1E -	Reserved
0F -	Flush 3	1F -	Reserved

Fig. 4-1 MCACM Command Codes

4.1 READ DATA COMMAND

The read data command requests up to 10 digits of main memory data. The returned data is left justified and padded with trailing zeros. Data is always returned from cache, therefore, if it is not already in cache it is first loaded into cache from main memory and then sent to the processor.

Whenever the requested data spans two blocks of memory, the MCACM generates two main memory requests to obtain the desired data. The MCACM concatenates these requests and returns the data. This is called operand read. The V500 processor's memory data cards do not have the ability to concatenate data onto the memory backplane like the B4900. Throughout this document the term "operand read" is used

to describe this concatenation across block boundaries.

4.2 WRITE DATA COMMAND

The write data command writes data from the processor into cache memory. The write data must be aligned as it is to appear in a "mod-10" 10 digit main memory word. The length of the supplied data may be up to 10 digits long as long as it does not cause the data to extend beyond the boundaries of a 10 digit word. Figure 4-2 shows two examples of writes; it shows the desired write and the series of commands required to accomplish it.

Examples:

To write 01234 at address 93 for 5.
Write: Address = 093, Length = 5, Data = xxx01234xx

To write 6789 at address 78 for 4.
1st Write: Address = 078, Length = 2, Data = xxxxxxxx67
2nd Write: Address = 080, Length = 2, Data = 89xxxxxxxx

Fig. 4-2 Example of Write Data Formats

If the addressed block is not "writeable" by the cache, then a main memory cycle is initiated to obtain a "writeable" copy before the write is done.

The memory control and cache module assists instruction retry in the V500 processor by aborting a write operation that contains a hardware generated error. After such a fatal error has been detected in a write operation, the remaining write operations in the pipeline are ignored. Write commands are handled after the processor requests an error report, which resets the MCACM. This allows the processor to do instruction level retry without external intervention.

4.3 READ WITH LOCK COMMAND

The read-with-lock command has been provided to allow interprocessor software interlocks. The read-with-lock command is a hybrid of the read and write commands which reads the previous data while writing new data. The read-with-lock command allows the processor to execute the lock and unlock instructions defined in Section 20.4 of the V Series Instruction Set Specification. These lock and unlock instructions depend on the value of a lock status field. This status field is part of a data structure addressed by the instruction. A certain value in this status field corresponds to an available lock while other values indicate the lock is not available.

Processors trying to obtain a lock, execute the read-with-lock command with the "unavailable" value in the data field. The read-with-lock command writes this value into the addressed memory location, which is usualy the lock status field, and returns the previous value of the field.

The memory subsystem is designed so that only one processor can do this at a time. The first processor to write the "unavailable" value into the status field receives "available" as the previous value. This processor has gotten the lock. The next processor to try this also writes the "unavailable" value into the status field but gets back the previous value which is already "unavailable".

The format of the data in a read-with-lock command is best described as a write which also reads the prior contents of the addressed location. The format of the data to be written into memory is the same as data for a write data command. The format of the data that is returned from memory is the same as the data from a read data command.

NOTE: The data format of a write operation does not allow data to cross "mod-10" boundaries; therefore, the read-with-lock command does not allow it either.

4.4 WRITE BASE TABLE COMMAND

The write base table command writes the absolute address from the address field into the base table entry specified by the Base Indicant Field. The base address must be MOD 10.

4.5 WRITE LIMIT TABLE COMMAND

The write limit table command writes the absolute address from the address field into the limit table entry specified by the base indicant field. The limit address must be MOD 10.

4.6 READ BASE TABLE COMMAND

The read base table command returns the absolute base address from the base table entry specified by the base indicant field. The base address is returned in the internal 27-bit binary format.

Note: The processor must pop the Read Base results immediately after issuing a Read Base Command. Failure to do so may result in invalid data. This is because the tag specified in the command may be returned more than once by MCACM: only the first one is accompanied by valid data.

4.7 READ LIMIT TABLE COMMAND

The read limit table command returns the absolute limit address from the limit table entry specified by the base indicant field. The limit address is returned in the internal 27-bit binary format.

Note: The processor must pop the Read Limit results immediately after issuing a Read Limit Command. Failure to do so may result in invalid data. This is because the tag specified in the command may be returned more than once by MCACM: only the first one is accompanied by valid data.

4.8 READ ERROR REPORT COMMAND

The read error report command is used in conjunction with the

read error address command to get details about hardware failures which resulted in a fatal error being reported to the processor module. The read error report command and the read error address command must both be issued to reset the error recorder and to allow future write commands to be handled. The read error report command must be issued before the read error address command.

The read error report command returns the memory data card number, the memory module location, and the error syndrome of the first-worst error that occured since the last error report was requested.

The read error report command causes the error recorder to stop accumulating errors and hold the current error until after the read error address command has been serviced. This assures that the error address is for the error report.

4.9 READ ERROR ADDRESS COMMAND

The read error address command is used in conjunction with the read error report command to get details about hardware failures which resulted in a fatal error being reported to the processor module. The read error report command and the read error address command must both be issued to reset the error recorder and to allow future write commands to be handled. The read error report command must be issued before the read error address command.

The read error address command returns the absolute BCD address of the same first-worst error that was reported in response to the read error report command. The error address is in the internal 27 bit binary format. The error address is right justified with leading zeros.

The read error address command allows the error recorder to restart accumulating errors after supplying the address of the current error. This assures that the error address matches the error report. The read error address command also resets the write-abort lockout condition; caused by

write errors.

4.10 WRITE PC COMMAND

The write PC command performs the dual purpose of separating instructions that write and assisting write error reporting. The execute module sends the PC (address of the current instruction) to the memory control and cache module with this command before the first write data command of any instruction that writes data. The 2 functions of this command are:

Separation of write instructions

This prevents instructions, which modify memory, from completing out of order in the memory control and cache module. This is necessary because out of order completion could prevent instruction retry. The write PC command prevents the memory control and cache module from handling any new write requests (from the processor) until all outstanding requests have been completed.

Error reporting

The PC of the current instruction is stored in a special PC register before any write requests are accepted from the processor. When a write request causes or contains an error, an error information field is sent to the processor. This guarantees that the proper PC is reported for any error occuring during the operation of a write request. An error detected during a write operation causes future write requests to be ignored until a read PC command is handled. This limits the possible data corruption to the single instruction which failed.

The PC is supplied in the write data field (NOT the address field) for a write PC command.

The memory control and cache module stores the PC in a special PC register in the data section of the memory control and cache module; therefore, the base indicant and address fields are "don't cares" for this command.

4.11 READ PC COMMAND

The read PC command is used by the execute module to determine the instruction address which caused a write error in the memory control and cache module. The read PC command returns the PC, which is stored in a special PC register in the rotate/concatenate section, to the processor.

The PC is stored in a special PC register. Therefore, the address, base indicant, and write data fields are "don't cares" for this command. The PC does not go through any BCD to or from binary conversion. All 40 bits (10 digits) of the PC register are returned unaltered to the processor.

4.12 READ I/O COMMAND

The read I/O command is used to communicate with the I/O subsystem. The I/O Translator's hardware registers and the time of day clock are accessible with this command. The desired register is encoded in the LSD of the address field from the processor.

I/O Hardware Register	Definition
0	reserved
1	"
2	"
3	"
4	"
5	"
6	"
7	"
8	"
9	time of day

Fig. 4-3 Read I/O Hardware Register Definitions

4.13 WRITE I/O COMMAND

The write I/O command is used to communicate with the I/O subsystem. The I/O Translator's hardware registers and the time of day clock can be written with this command. The register is specified by the LSD of the address field from the processor.

I/O Hardware Register	Definition
0	reserved
1	I/O Inititate
2	reserved
3	"
4	"
5	"
6	"
7	"
8	"
9	Time of day

Fig. 4-4 Write I/O Hardware Register Definitions

4.14 WRITE-BACK COMMAND

The write-back command forces the MCACM to write modified data to main memory. This is a maintenance command (it is not used during normal system operation) which allows a processor to be shut down after moving all locally cached, modified data to main memory. There are 3 possible actions which may result from this command.

1- Neither of the blocks in this set is modified: don't do anything.

2- One of the blocks in this set is modified: write the data from this block to main memory and mark the block invalid.

3- Both of the blocks in this set are modified: write the data from one of the blocks to main memory and mark that block invalid.

The address portion of the the write-back command is only used to select 1 of the 1024 sets in cache. Therefore, to purge the cache of all modified data, the processor must issue 2048 write-back commands. Each command bust have a different mod-40 address (the block size) so that every block is handled.

4.15 FLUSH COMMANDS

There are 4 flush commands which are all treated the same. Only 1 flush command is needed by the memory control and cache module but the processor encodes information for itself in the the least significant 2 bits.

The flush commands operates in 2 modes: immediate and token. The immediate mode occurs when a flush command is detected on the backplane from the processor (before being pushed into the input queue). The token mode occurs when the flush command is popped out of the input queue. The actions of these modes are:

Immediate

The flush command immediately (hence the name of the mode) signals the input and output interfaces of the flush-state condition. The immediate actions of the interfaces are:

Disable RBUS output

Outstanding read requests are not sent to the processor. The data-valid flag described in Section 5.1.3 is forced not valid.

Ignore queued read requests

The flush-state treats read requests as no-operation commands.

Token

The flush command token is handled as a read operation which does not return any data. The RBUS output is enabled when all outstanding read requests have been flushed.

NOTE: Writes are handled during the flush-state and errors which occur during write operations are still reported to the processor.

NOTE: Flushes are nestable. When a second flush is received, before a flush is finished, then the flush state is extended until the last flush has been finished.

4.16 NO OPERATION COMMANDS

There are 4 no operation commands which are ignored by the memory control and cache module. They are only used for communication between processor modules.

4.17 TEST ERROR DETECTION COMMAND

The test error detection command allows function test micro-code to report a "fake" error from the memory control and cache module to the SMC.

5 INTERFACES

All of the interfaces to and from the memory control and cache module are active high. All bus transfers are protected by parity or ECC. The parity bits are calculated such that the overall parity of the protected field (including the parity bit) is even. This prevents the cleared state from being detected as an error.

5.1 PROCESSOR INTERFACE

The interface from the processor to the memory control and cache module is based on 4 major parameters which are each protected by even parity. These parameters are:

COMMAND
: The commands are described in Section 4.

ADDRESS
: The address consists of a 9 digit BCD area-relative address and a 4 bit base indicant.

WRITE DATA
: The write data field is described with the write data and read-with-lock commands in Sections 4.2 and 4.3 respectively.

TAG
: The tag field is the link between the specific requests from the processor and the resulting data or error messages from the memory control and cache module.

The interface from the memory control and cache module back to the processor is based on 3 parameters which are also protected by parity. These parameters are:

READ DATA
: The requested data or address.

ERROR INFORMATION
: The status of the requested operation.

TAG
: The link to the data and error information.

5.1.1 ADDRESS AND WRITE BUS SIGNALS

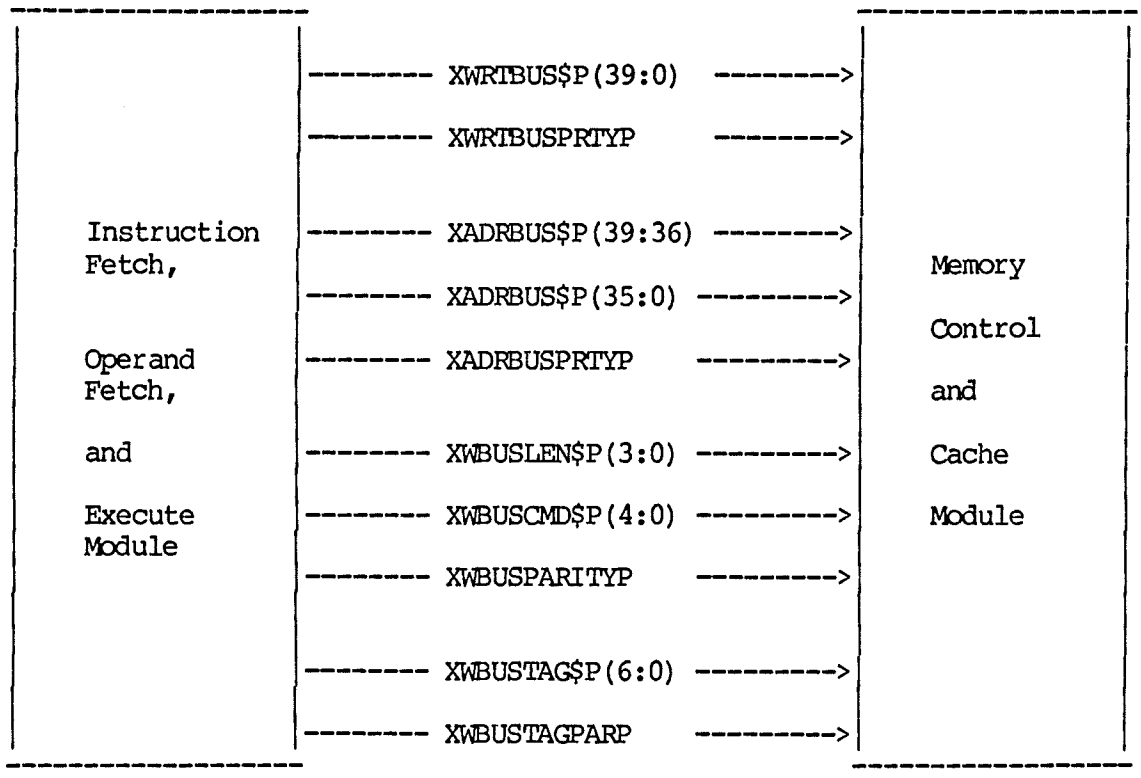


Fig. 5-1 Address and Write Bus Interface Signals

The address and write bus contains the following fields:

XWRITBUS$P(39:0)
The write data field is 10 digits wide. See the description of the write-data command for a detailed definition of the data in this field.

XWRITBUSPRTYP
The write data parity bit must be calculated such that the overall parity of the data field and the data parity bit is even.

5.1.1 ADDRESS AND WRITE BUS SIGNALS (Continued)

XADRBUS$P(39:36)
The base indicant field addresses the base/limit tables to specify which of the 16 possible bases to add to the address and the corresponding limit to check. The base indicant field also addresses the base/limit tables for the commands which read or write the base or limit table entries.

XADRBUS$P(35:0)
The Address field contains a right justified, zero packed address with no undigits. All addresses are relative to the particular base that is specified in the base indicant field. A base/limit table entry can be dedicated to the absolute base and limit of the machine to allow absolute addressing.

XADRBUSPRTYP
The address parity bit is calculated such that the overall parity of the address field, the base indicant field, and the address parity bit is even.

XWBUSLEN$P(3:0)
The length field specifies the length of the requested or supplied data. Only the values of '0' to '9' are valid. The values ('A' through 'F') are invalid. The hex value of '0' is interpreted as 10 digits by the memory subsystem.

For write commands, including read-with-lock, this length field must be consistent with the data format required. For example, a write to address 9 with a length field greater than 1 is a fatal, invalid-length error.

5.1.1 ADDRESS AND WRITE BUS SIGNALS (Continued)

XWBUSCMD$P(4:0)
The command field specifies the operation to be performed by the memory subsystem. A full description of each of the commands is contained in Section 4, but a list is included here for completeness.

01	-	Read Data
1D	-	Write Data
16	-	Read with Lock
19	-	Write Base Table
1B	-	Write Limit Table
05	-	Read Base Table
07	-	Read Limit Table
13	-	Read Error Report
12	-	Read Error Address
10	-	Write PC
11	-	Read PC
02	-	Write I/O Register
03	-	Read I/O Register
011xx	-	Flushes
010xx	-	No-Operations
18	-	Write-Back
06	-	Read ECC codes
1A	-	Write ECC codes
1E	-	Read Data Uncorrected
04	-	Test Error Detection

Fig. 5-2 Memory Control and Cache Module Commands

XWBUSPARITYP
The control parity bit is calculated such that the overall parity of the command, the length, and the control parity bit is even.

XWBUSTAG$P(6:0)
The tag field is a "don't care" to the memory subsystem, which does not modify or even consider its contents.

5.1.1 ADDRESS AND WRITE BUS SIGNALS (Continued)

Certain aspects of the tag field have been agreed to by the processor modules, and are included here for completeness. The tag field is composed of 2 subfields. The first subfield identifies the destination module of the RBUS information. The second subfield addresses the destination's queue(s) or contains the instruction number for write operations.

tag	entries	source --> destination
0000000	1	MCACM generated = no transfer
0000001 through 0001111	15	Reserved
001xxxx	16	Instruction Fetch --> Instruction Fetch
0101xx0	4	Operand Fetch --> Execute "A" Operand Queue
0101xx1	4	Operand Fetch --> Execute "B" Operand Queue
011xxxx	4	Operand Fetch --> Operand Fetch
100xxxx	16	Fetch Write
1010xxx	8	Execute Module Write
1100xxx	8	Execute Module --> RBUS Queue
1101xxx	8	Execute Module --> Operand Queue
1110000 through 1111110	15	Reserved
1111111	1	MCACM generated = no transfer

Fig. 5-3 Definition of Tag Field

XWBUSTAGPARP
The tag parity bit is calculated such that the overall parity of the tag field and the tag parity

bit is even.

5.1.2 ADDRESS AND WRITE BUS ARBITRATION

The access arbitration mechanism is similar to the B4900. Each module monitors the request signals from modules with higher priorities and the registered AWFULL signal. When a module requests the bus, it drives the entire AWBUS (address, command, data, length, tag, base indicant, source ID, and write data). If a higher priority module is also requesting the AWBUS, then all lower priority modules disable their bus drivers.

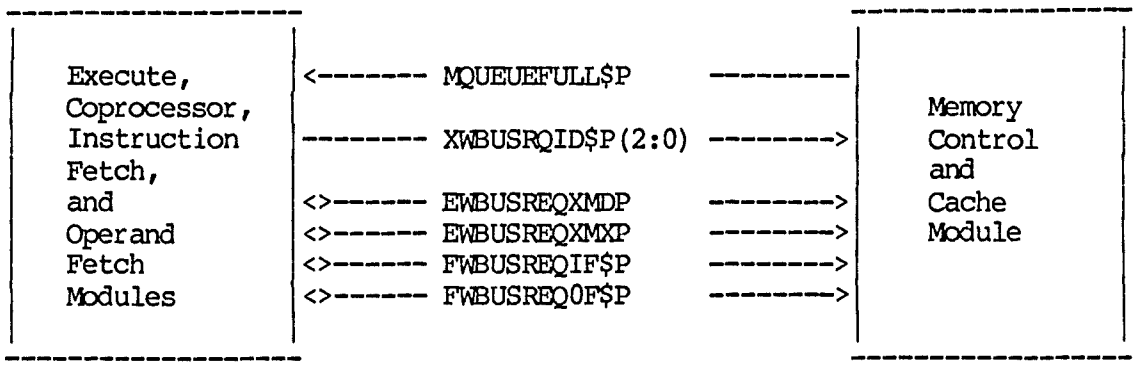


Fig. 5-4 Address and Write Bus Access Arbitration Signals

The address and write bus access arbitration signals are:

MQUEUEFULL$P

The queue full signal is sent to the processor modules whenever the AWBUS input queue is almost full. This signal is registered on each of the processor modules and used in the access arbitration on the next clock.

The queue full signal is bidirectional. The XM module will force queue full when it needs to prevent the fetch module from requesting memory.

5.1.2 ADDRESS AND WRITE BUS ARBITRATION (Continued)

XWBUSRQID$P(2:0)
The module ID of the processor module which is currently sourcing the AWBUS. This signal is compared inside the MCACM with the prioritization of the four bus request lines and AWFULL to detect errors in the access arbitration logic. The ID field is encoded as follows:

0 No Transfer (no request or AWFULL)

1 Execute Module

2 Co-processor Module

3 reserved (do not use)

4 Operand Fetch Module

5 Instruction Fetch Module

6 reserved (do not use)

7 reserved (do not use)

EWBUSREQXMDP
The AWBUS request signal from the execute module.

EWBUSREQXMXP
The AWBUS request signal from the co-processor module.

FWBUSREQIF$P
The AWBUS request signal from the instruction fetch module.

FWBUSREQOF$P
The AWBUS request signal from the operand fetch module.

5.1.3 READ BUS SIGNALS

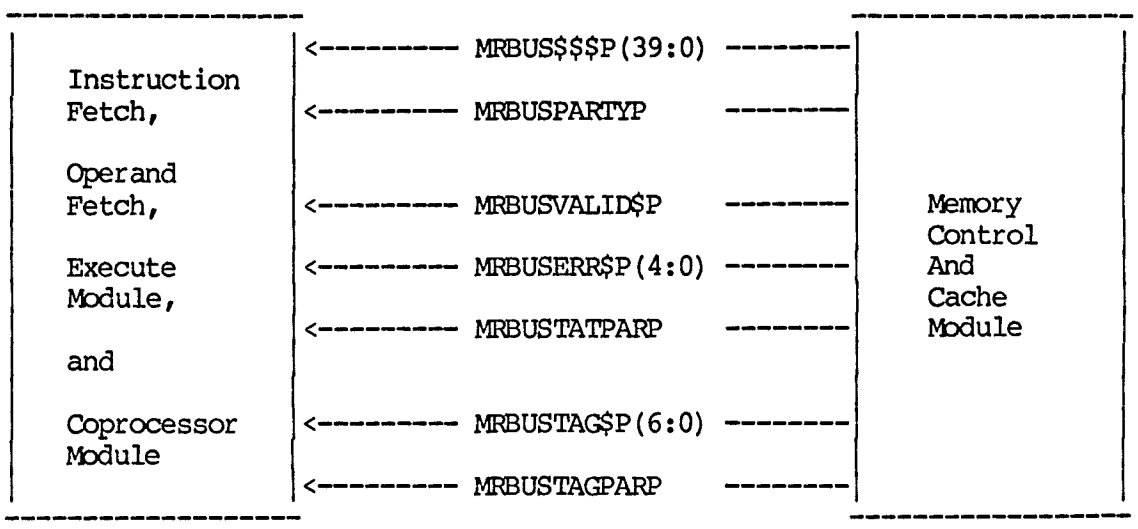


Fig. 5-5 Read Bus interface Signals

The return bus to the processor is called the RBUS. The RBUS contains 3 subfields: data, status, and tag. The data subfield consists of:

MRBUS$$$P(39:0)
The read data field is 10 digits wide. Data is returned left justified and zero padded. "Operand-reads" are spliced together by the MCACM and sent as a single RBUS cycle.

MRBUSPARITYP
The read data parity bit is calculated such that the overall parity of the data field and the data parity bit is even.

5.1.3 READ BUS SIGNALS (Continued)

The status subfield of the RBUS contains the following information.

MRBUSVALID$P

The read data valid flag. This valid flag, when active, indicates that the RBUS contains a valid transfer. When certain cache misses occur, the tag bits will be sent to the processor modules before the miss has been determined. When this occurs, the processor is signaled (via the valid flag) to ignore the data that is on the RBUS. When the miss has been satisfied from main memory, the tags are sent again and the actual data (with valid active) is actually transmitted.

MRBUSERR$P(4:0)

The error information field contains error status indications of each operation initiated by the memory subsystem. The error information field is all zero's for operations which complete without any errors. When multiple errors are encountered, only the worst error is reported. The encoding of the errors in the error information field reflects the seriousness of the error. Higher numbered errors are worse than lower numbered errors.

The "fatal" distinction means that the operation did not complete. The fatal errors "A" through "F" are not returned to the processor, but send a MCACM broken signal to the SMC. The error information field is defined in Figure 5-6.

5.1.3 READ BUS SIGNALS (Continued)

Error Vector	
Non-Fatal Errors	Fatal Errors
00) No Errors	10) Multiple-Bit Memory Board Failure
01) Corrected Memory Data	11) Limit Error
02) Corrected Memory Address	12) Un-Digits in Address Field
03) Not used	13) Invalid Length Field
04) Corrected Cache Data	14) Invalid Command Field
05) Corrected Cache Address	15) Internal & Bus Arbitration Error
06) Not used	16) Input Data Parity Error
07-0F) Not-used	17) Input Address Parity Error
	18) Input Control Parity Error
	19) Input Tag Parity Error
	1A) Multiple-Bit Memory Bus Failure
	1B) Multiple-Bit Cache Failure
	1C) Memory Address Failure
	1D) Cache Failure
	1E) Write-Back Failure
	1F) MCACM Failure

Fig. 5-6 Definition of Error Vector

MRBUSTATPARP

The status parity bit is calculated such that the overall parity of the error information field, the data valid flag, and the status parity bit is even.

5.1.3 READ BUS SIGNALS (Continued)

The tag subfield of the RBUS contains the following information.

MRBUSTAG$P(6:0)

The return tag field is an exact copy of the AWTAG field which was supplied by the processor module that initiated this operation. This tag field is returned 1 clock cycle prior to the requested data. This tag field in conjunction with the error vector are the key items which insure that a memory operation is properly executed. In the event of an error during a write operation, the tag is sent to the processor and then the error information is sent on the next clock. The processor must differentiate between read tags and write error tags.

MRBUSTAGPARP

The return tag parity bit is an exact copy of the tag parity field which was supplied by the processor module which initiated this operation. The tag parity bit was calculated such that the overall parity of the tag field and the tag parity bit is even.

5.2 MAIN MEMORY INTERFACE

The main memory structure of the V500 processor is composed of up to 16 memory data cards; which, each have 1,572,864 blocks. The main memory structure does not understand any of the peculiarities of the V-Series Instruction Set; it only handles full 40 digit (whole block) operations. The main memory structure does not rotate or concatenate data in any way at all. This simplifies all 16 memory data cards and the memory interface. This also allows a common solution to the synthesis of non-word operations by the memory control and cache module.

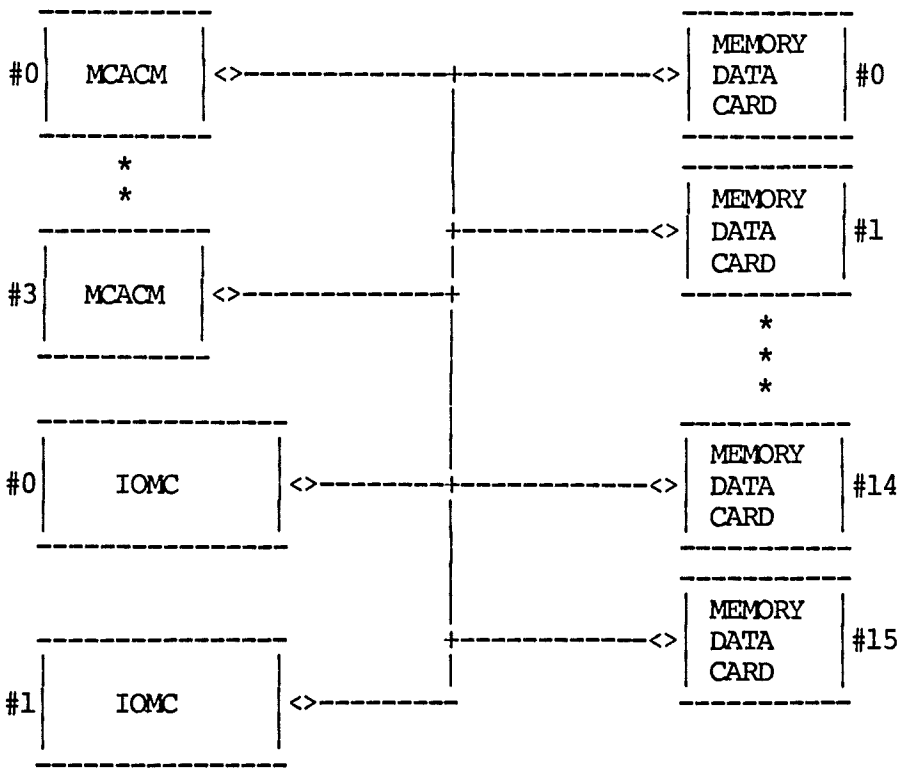


Fig. 5-7 Main Memory Storage Structure Interconnections

5.2 MAIN MEMORY INTERFACE (Continued)

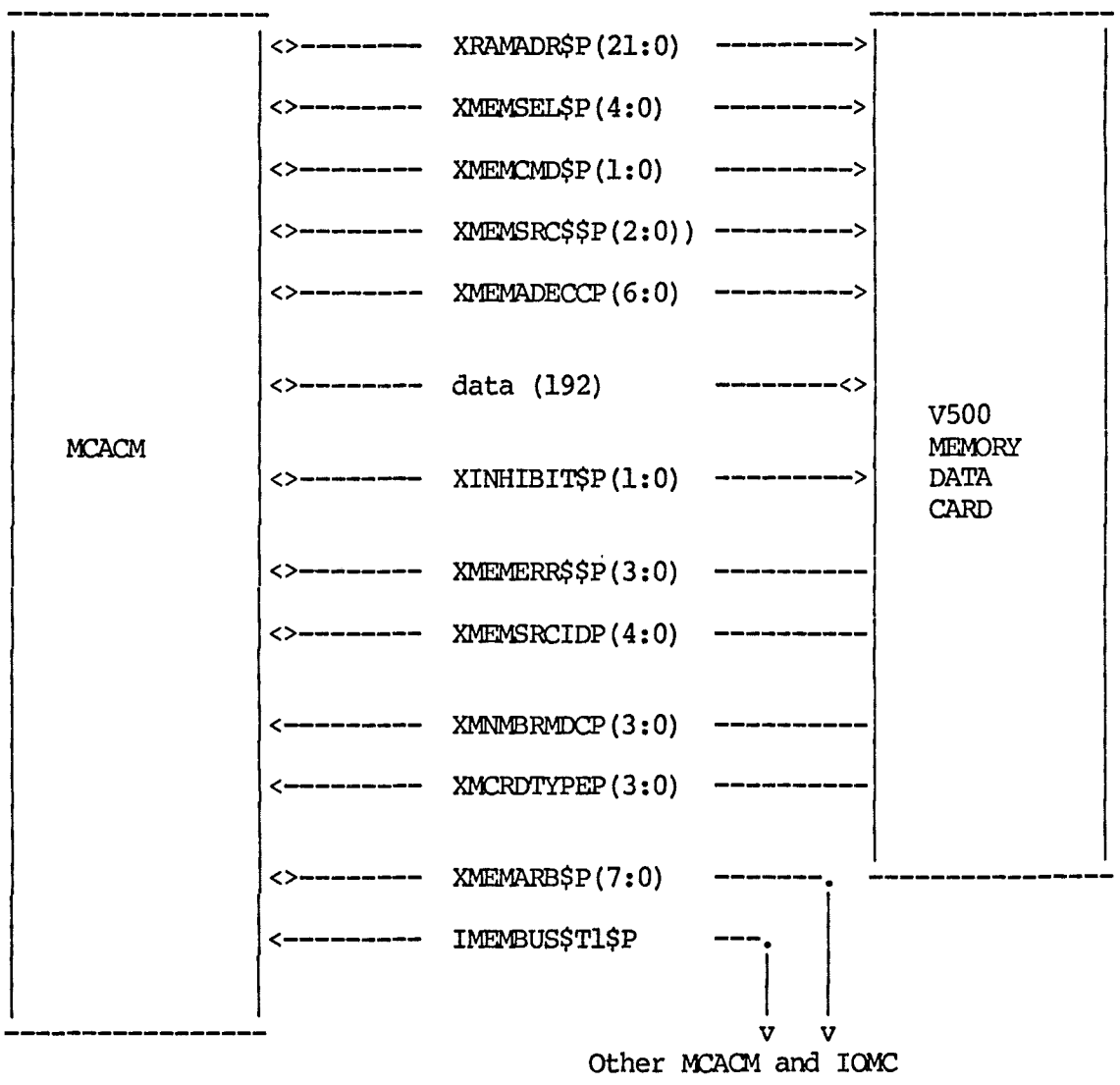


Fig. 5-8 Memory Data Card Interface

5.2.1 MEMORY ADDRESS BUS

The main memory interface bus contains the following signals:

XRAMADR$P(21:0)
The absolute binary module word address. This allows addressing all 4M blocks on a memory data card; which is, fully populated with 1M-bit DRAM chips.

XMEMSEL$P(4:0)
This field selects the memory data cards. The V500 processor can accomodate 16 memory cards, the other 16 values are used to select the I/O memory concentrators for memory mapped I/O operations.

XMEMCMD$P(1:0)
The memory data card commands are:

PUBLIC BLOCK READ = [10]
A full 40 digit block is read. The spies on all of the other processors check for locally cached data that has been modified. The response of the other spies is discussed in more detail in Section 7.

PRIVATE BLOCK READ = [11]
A full 40 digit block is read. The spies on all of the other processors check for local cached data and invalidate unmodified copies or relinquish modified copies. This is discussed in more detail in Section 7.

BLOCK WRITE = [01]
A full 40 digit block is written.

READ-MODIFY-WRITE READ = [00]
This is identical to the PRIVATE BLOCK READ, except that the selected memory data card remains busy until the next BLOCK WRITE operation. This allows the IOMC to implement a non-interuptible read-modify-write operation.

5.2.1 MEMORY ADDRESS BUS (Continued)

XMEMSRC$$P(2:0)

This field is used for distributed error detection of the memory bus arbitration logic. Figure 5-9 defines the values of this field.

The V500 System can accommodate one or two IOMC's. These IOMC's may be arranged as one in each cabinet or both in one cabinet. Therefore, the memory bus arbitration and the requestor Id field allow for four IOMC's.

0 - MCACM in Processor 1 in Cabinet A
1 - MCACM in Processor 2 in Cabinet A
2 - MCACM in Processor 1 in Cabinet B
3 - MCACM in Processor 2 in Cabinet B
4 - IOMC in Slot 1 in Cabinet A
5 - IOMC in Slot 2 in Cabinet A
6 - IOMC in Slot 1 in Cabinet B
7 - IOMC in Slot 2 in Cabinet B

Fig. 5-9 Memory Address Bus Requestor ID Field Definition

5.2.1 MEMORY ADDRESS BUS (Continued)

XMEMADECCP(6:0)

This address ECC is calculated over the address, module select, requestor ID, and command fields. This error correct code is identical to the code which protects the I/O bus between the IOMC's, the IOP's, and the DTM's.

	6	5	4	3	2	1	0
00 :=	X	X			X		
01 :=	X	X					X
02 :=	X	X				X	
03 :=	X	X	X				
04 :=	X		X	X			
05 :=		X	X		X		
06 :=	X		X				X
07 :=	X		X			X	
08 :=		X	X				X
09 :=		X	X			X	
10 :=			X	X	X	X	X
11 :=	X		X		X		
12 :=				X	X	X	
13 :=				X		X	X
14 :=			X	X			X
15 :=		X		X			X
16 :=	X			X		X	
17 :=	X			X	X		
18 :=	X	X		X			
19 :=		X	X	X			
20 :=			X		X		X
21 :=	X	X	X	X	X		
22 :=		X			X	X	
23 :=	X				X	X	
24 :=		X			X		X
25 :=	X				X		X
26 :=			X		X	X	
27 :=				X	X		X
28 :=					X	X	X
29 :=		X				X	X
30 :=	X					X	X
31 :=			X			X	X

Fig. 5-10 Address Bus Error Correction Code

5.2.1 MEMORY ADDRESS BUS (Continued)

0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
6543	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
000	*	X	X	D	X	D	D	28	X	D	D	13	D	27	12	D
001	X	D	D	31	D	20	26	D	D	14	M	D	M	D	D	10
010	X	D	D	29	D	24	22	D	D	15	M	D	M	D	D	M
011	D	08	09	D	05	D	D	M	19	D	D	M	D	M	M	D
100	X	D	D	30	D	25	23	D	D	M	16	D	17	D	D	M
101	D	06	07	D	11	D	D	M	04	D	D	M	D	M	M	D
110	D	01	02	D	00	D	D	M	18	D	D	M	D	M	M	D
111	03	D	D	M	D	M	M	D	D	M	M	D	21	D	D	M

* = NO BITS IN ERROR, X = CHECK BIT IN ERROR
D = DOUBLE BIT ERROR, M = MULTIPLE ERRORS

Fig. 5-11 Address Bus Syndrome to bit-in-error Decode Table

To further enhance the fault detection capability of the address bus error correction code, two of the bits are inverted. Bits 2 and 4 are inverted (after the generation, as specified in Figure 5-10) so that the ECC code for all-zero data is "14" hex. This makes the floating bus (all zeros, including ECC) case a double bit error. This "error" is only checked when traffic is expected on the bus.

5.2.2 MEMORY DATA BUS

The memory subsystem data communication path consists of a 160 bit wide bidirectional data bus. This bus provides communications between memory control and cache modules, I/O memory concentrators and memory data cards. This communications path contains:

DATA FIELD (192 bits)

The write data field is composed of 160 bits (40 digits) of data and 32 bits of ECC. The ECC is calculated across 4 fields of 40 bits each. Error correction across this bus is done by the memory data cards and the memory control and cache module to provide an additional level of fault tolerance. The ECC encode table for each 40 bit field of the data bus is given in Figure 5-12. The syndrome to bit in error decode table is given in Figure 5-13.

To further enhance the fault detection capability of the data bus error correction code, two of the bits are inverted. Bits 0 and 7 are inverted (after the generation specified by the code in Figure 5-12) so that the ECC code for all-zero data is "81" hex. This makes the floating bus (all zeros, including ECC) case a double bit error. This "error" is only checked when traffic is expected on the bus.

5.2.2 MEMORY DATA BUS (Continued)

	7	6	5	4	3	2	1	0
00 :=				X			X	X
01 :=			X				X	X
02 :=		X					X	X
03 :=	X						X	X
04 :=				X		X		X
05 :=			X			X		X
06 :=		X				X		X
07 :=	X					X		X
08 :=				X		X	X	
09 :=			X			X	X	
10 :=		X			X	X		
11 :=	X				X	X		
12 :=				X	X	X		
13 :=			X		X	X		
14 :=		X			X			X
15 :=	X				X			X
16 :=				X	X		X	
17 :=			X		X		X	
18 :=		X			X		X	
19 :=	X				X		X	
20 :=		X		X				X
21 :=		X		X			X	
22 :=		X		X		X		
23 :=		X		X	X			
24 :=	X			X				X
25 :=	X			X			X	
26 :=			X	X		X		
27 :=			X	X	X			
28 :=			X	X				X
29 :=			X	X			X	
30 :=		X	X			X		
31 :=		X	X		X			
32 :=	X		X					X
33 :=	X		X				X	
34 :=	X		X			X		
35 :=	X		X		X			
36 :=	X	X						X
37 :=	X	X					X	
38 :=	X	X				X		
39 :=	X	X			X			

Fig. 5-12 Data Bus Error Correction Code

5.2.2 MEMORY DATA BUS (Continued)

0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
76543	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0000	*	40	41	D	42	D	D	M	43	D	D	M	D	M	M	D
0001	44	D	D	00	D	04	08	D	D	M	16	D	12	D	D	M
0010	45	D	D	01	D	05	09	D	D	M	17	D	13	D	D	M
0011	D	28	29	D	26	D	D	M	27	D	D	M	D	M	M	D
0100	46	D	D	02	D	06	M	D	D	14	18	D	10	D	D	M
0101	D	20	21	D	22	D	D	M	23	D	D	M	D	M	M	D
0110	D	M	M	D	30	D	D	M	31	D	D	M	D	M	M	D
0111	M	D	D	M	D	M	M	D	D	M	M	D	M	D	D	M
1000	47	D	D	03	D	07	M	D	D	15	19	D	11	D	D	M
1001	D	24	25	D	M	D	D	M	M	D	D	M	D	M	M	D
1010	D	32	33	D	34	D	D	M	35	D	D	M	D	M	M	D
1011	M	D	D	M	D	M	M	D	D	M	M	D	M	D	D	M
1100	D	36	37	D	38	D	D	M	39	D	D	M	D	M	M	D
1101	M	D	D	M	D	M	M	D	D	M	M	D	M	D	D	M
1110	M	D	D	M	D	M	M	D	D	M	M	D	M	D	D	M
1111	D	M	M	D	M	D	D	M	M	D	D	M	D	M	M	D

* = NO BITS IN ERROR
nn = SINGLE BIT IN ERROR, where 'nn' is the bad bit
D = MULTIPLE ERRORS (even number)
M = MULTIPLE ERRORS (odd number)
Note that bits 40 thru 47 are the check bits.

Fig. 5-13 Data Bus Syndrome to bit-in-error Decode Table

5.2.2 MEMORY DATA BUS (Continued)

XINHIBIT$P(1:0) - INHIBIT

The INHIBIT signal is active on the bus cycle (2 system clocks) prior to data returning from the MDC. It informs the MDC that a MCACM spy has detected a hit (that is, another MCACM's requst is for locally cached, modified data); therefore, the MDC must disable its memory data bus drivers. This enables the MCACM to return the data. The INHIBIT signal is duplicated and compared by each MDC for fault detection.

XMODIFIEDP(1:0) - MODIFIED

The MODIFIED signal is active on the same bus cycle (2 system clocks) as data returning from a main memory request. MODIFIED always occurs on the bus cycle following INHIBIT. It informs the requesting MCACM that this data is already modified. The MODIFIED signal is duplicated and compared by each MCACM for fault detection.

XCACHCOPYP(1:0) - COPY

The COPY signal is active during the same bus cycle (2 system clocks) that data is on the memory data bus. It informs the requesting MCACM that another MCACM has a locally cached copy of this data; therefore, this data has the COPY status: it is not writeable. the COPY signal is duplicated and compared by each MCACM for fault detection.

5.2.3 MEMORY ERROR BUS

Two fields are sourced on the 5th clock of each memory cycle. These fields are sourced by memory data cards and by MCACM (when responding to spy hits). These fields are used by the error recorder to distinguish between memory bus errors and memory data card errors. These fields are:

ERROR (4 bits)

This field is encoded with 1 of 6 possible error conditions that are detected by the memory data cards. It is driven by the memory data cards on the 5th clock of each memory cycle. Therefore, each memory requestor must check this field on the 5th clock of the memory cycles that it initiates to detect errors. The field is encoded with odd parity for fault detection. The possible conditions are:

- 0 No memory cycle in progress.
- 1 No errors in this memory cycle.
- 2 Single bit data error.
- 4 Single bit address error (corrected).
- 7 Multiple single bit data errors.
- 8 Undefined.
- B Multi-bit data error (uncorrectable).
- D Multi-bit address error (uncorrectable).
- E Internal malfunction (fatal).

ID (5 bits)

This field contains the ID number of the module which is driving the backplane.

5.2.4 MEMORY SUBSYSTEM CONFIGURATION BUS

The memory subsystem is designed to allow great flexibility in the number and type of memory data cards which may be used in a V500 system. This allows different storage capacities and the ability to take advantage of faster and denser DRAM chips as they become available.

The Maintenance processor determines the number of on-line memory data cards in the system. This information is then shifted into a hidden state register of each MDC. The MDCs will transfer this information and the "Card Type" information to MCACM. Memory System Configuration will be done by the Maintenance Processor based on this information.

XMNMBRMDC(3:0)P (4 bits)

The number of boards field specifies how many memory data cards are currently plugged into the V500 backplane. This value only includes those cards that are "online", and not any cards that have been put into an "offline" condition by the maintenance subsystem.

# online cards	kinds of interleaving 4-way	2-way	1-way
1			1
2		2	
3		2	1
4	4		
5	4		1
6	4	2	
7	4	2	1
8	2*4		
9	2*4		1
10	2*4	2	
11	2*4	2	1
12	3*4		
13	3*4		1
14	3*4	2	
15	3*4	2	1
16	4*4		

Fig. 5-14 "Any-Number" Memory Interleaving Scheme

5.2.4 MEMORY SUBSYSTEM CONFIGURATION BUS (Continued)

XMCRDTYPE(3:0)P - Memory Data Card Type

There are 4 types of memory data cards: full-populated 1M bit, half-populated 1M bit, full-populated 256K bit, and half-populated 256K bit. Each bit of this field corresponds to a type of card which is protected with a 1-out-of-4 code. Only 1 type of card is allowed to exist in the backplane at a time, and at least 1 card must exist, so one and only one bit will always be on. The following table shows the definition of the bits in this field.

bit 3 - full-populated 1M bit MDC(s)
bit 2 - half-populated 1M bit MDC(s)
bit 1 - full-populated 256K bit MDC(s)
bit 0 - half-populated 256K bit MDC(s)

Without contradicting the only-one-type-at-time rule, it is possible to have more than 1 type of card in the system, as long as ONLY the smallest type of card is reported to the MCACM and IOMC modules.

5.2.5 MEMORY MODULE STATUS INTERFACE

The status of the individual memory data cards is maintained in the MCACM's and IOMC's. This status is derived by monitoring the MODULE SELECT field in the MEMORY ADDRESS BUS and by the busy lines from each MDC. The IOMC status is maintained in the same way because I/O operations from the processor are handled as special memory requests. There are 2 different reasons for a MDC (or IOMC) to be busy.

BUSY BY REQUEST FROM MCACM OR IOMC:

Each MDC (or IOMC) is busy for 2 bus cycles (4 system clocks) after each request. By remembering which memory modules were selected during the last 2 bus cycles, each MCACM and IOMC is able to determine which MDC's are busy.

All I/O requests are treated as a single MDC for the purpose of determing busy status. Therefore, any memory mapped I/O operation will cause the IOMC(s) to be treated as busy for the next 2 bus cycles (4 system clocks).

BUSY BY INTERNAL OPERATION:

Whenever a MDC (or IOMC) is busy by an internal operation, it drives a busy signal. These signals allow each MCACM and IOMC to determine which MDC (or IOMC) is busy. These busy signals are:

XMEMBUSYP(15:0) - MEMORY MODULE BUSY

Each MDC drives a particular BUSY line based on its logical address. The MCACM must wait until the desired MDC is not busy before requesting a memory operation from it. A MDC may drive the BUSY line for 2 reasons:

1 - The IOMC has requested a read-modify-write cycle and has not yet done the write portion.

2 - The MDC needs to do an internal refresh cycle.

5.2.5 MEMORY MODULE STATUS INTERFACE (Continued)

XIOMCBUSYN(1:0) - IOMC BUSY (ACTIVE LOW)
Each IOMC drives a particular IO BUSY signal based on its cabinet or slot number. The MCACM must wait until at least 1 IOMC is not busy before requesting a memory mapped I/O operation.

5.2.6 MEMORY BUS ACCESS ARBITRATION

Memory bus access arbitration is a 2 level process. First, the memory data card must be not-busy. Second, there must not be any higher priority devices requesting the bus. The first condition is determined by monitoring the busy lines from the memory data cards. The second condition is determined with two unidirectional buses and an arbitration bus.

XNEARSEL$P(4:0)
A unidirectional five bit bus from this cabinet to the other cabinet. This bus is driven at the same time as the address and command bus to main memory. It contains a copy of the module select field. This bus is monitored by the MCACM and IOMC modules in this cabinet to determine which memory module is being selected on this cycle.

XFARSEL$$P(4:0)
A unidirectional five bit bus from the other cabinet to this cabinet. This bus is the other end of the XNEARSEL$P bus from the other cabinet. This bus is received only to each memory requestor. It is used to determine which memory module is being selected by the other cabinet on this cycle. Note: all zero's on this bus correspond to no module being selected.

XMEMARB$P(7:0)
This is the memory arbitration bus. There is a separate wire for each requestor, determined by backplane jumpers. Each wire has only one source, and up to five receivers. The appropriate wire is only driven by a memory requestor once it determines that the module that it is requesting is not busy. The highest priority requestor is then granted the use of the bus on the next bus cycle.

5.2.6 MEMORY BUS ACCESS ARBITRATION (Continued)

It is necessary for the XFARSEL$$P bus and the XMEMARB$P buses to each cross the backplane before any memory request can be granted.

When a memory requestor is granted the memory address and control backplane bus, the requestor will drive address, control, possibly data, and a requestor ID field. This ID field is derived from the backplane jumpers which define each memory requestors priority. The requestor ID is monitored by each memory requestor to check that the highest priority requestor actually won the bus arbitration. MCACM broken will be signaled at the failure of the bus priority resolution.

5.2.7 MEMORY BUS CYCLE TIMING

The main memory bus cycle is 2 processor clocks long. It is controlled by a signal which is generated by the IOMC. This signal is:

IMEMBUS$T1$P

Main memory bus phase level. A main memory bus cycle is defined as the 2-clocks ending with the clock that IMEMBUS$T1$P is active-high on the backplane. This signal is a level, not a clock, and must be registered before being used.

5.3 I/O SUBSYSTEM INTERFACE

The memory control communicates with the I/O memory concentrator over the main memory bus with a memory mapped I/O scheme. The I/O memory concentrator is addressed as modules 24 through 31 which is beyond the addressable limit of the V500 processor and is therefore unused by the memory storage structure. The I/O memory concentrator responds like a memory data card.

The I/O memory concentrator only accepts the full-block read and write memory commands. There is no public or private distinction for IOMC commands. The I/O memory concentrator emulates a memory data card for these commands. Since the I/O memory concentrator cannot respond as frequently as the memory data cards, it supplies a signal to the MCACM's whenever it is unable to accept a command. This additional interface signal is:

I/O MEMORY CONCENTRATOR BUSY
: This signal is set by the I/O memory concentrator when it is unable to respond to a memory mapped I/O operation. During a 2 clock window after this signal is set, the IOMC still responds to memory mapped I/O operations. This provides the necessary time for the MCACM's to register the signal. After this 2 clock window, the IOMC ignores memory mapped I/O operations.

There are a number of types of information which the processor and I/O subsystem need to communicate. Among these are the time of day clock and the so-called hardware registers in the I/O Translator. Each of these types of information is associated with a different address on the I/O memory concentrator.

The processor module supplies or requests information for the I/O subsystem through the use of the I/O write and I/O read commands respectively. The address field of the command determines the type of information. The actual addresses and associated information are not specified yet.

6 MEMORY CONTROL DESCRIPTION

The memory control is composed of 3 main sections: the address section, data section, and cache. The address and data sections are split onto 2 physical boards so that it is convenient to discuss them separately. Cache is also discussed separately because of its unique nature. Fault detection and handling is also discussed as a separate subject because of the strong emphasis on fault-tolerance in the V500 System.

6.1 ADDRESS SECTION

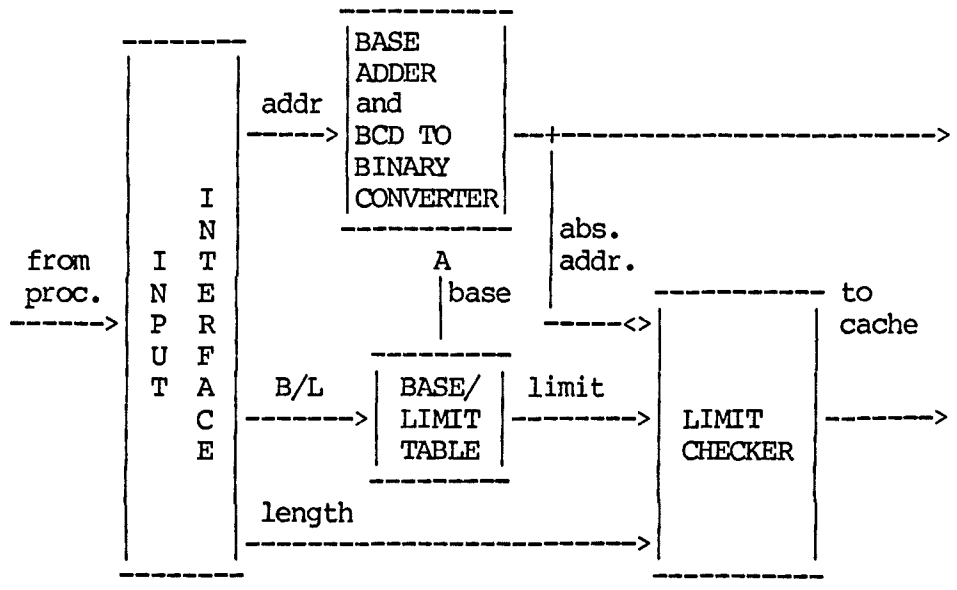


Fig. 6-1 Block Diagram of Memory Control Address Section

6.1.1 INPUT INTERFACE

The input interface is composed of a 16 deep, first-in, first-out queue structure and a queue controller state machine. The FIFO queue is quite ordinary and deserves little more than mention. The state machine however, has the responsibility of handling: operand read sequencing, flush operation, read/write PC interaction, and FIFO addressing. Therefore, some time is spent on its operation.

Instead of drawing a complicated state diagram or attempting a line-by-line description of the sequencing of the input interface state machine, the BNF notation of compiler designers will be used to describe its operation. Some definitions may be in order.

<State> ::=
Definition of a state.

<State>
Go to state.

{something}
Do something.

go/stop
Stop here if cache is asserting holdque.

not-reading
Wait until cache is finished with all reads.

not-writing
Wait until cache is finished with all writes.

6.1.1 INPUT INTERFACE (Continued)

Input Interface State Transitions in Backus Nauer Form

```
<TOP> ::= EMPTY <TOP>
<TOP> ::= 00"RESERVED"     {BROKEN}
<TOP> ::= 01"READ DATA"    go/stop <OPERAND> {POP} <TOP>
<TOP> ::= 02"WRITE I/O"    go/stop {POP} <TOP>
<TOP> ::= 03"READ I/O"     go/stop {POP} <TOP>
<TOP> ::= 04"TEST ERROR"   <PAUSE> {FAKE ERROR} {POP} <TOP>
<TOP> ::= 05"READ BASE"    {BASE RD} go/stop {POP} <TOP>
<TOP> ::= 06"READ ECC"     go/stop {POP} <TOP>
<TOP> ::= 07"READ LIMIT"   {LIMIT RD} go/stop {POP} <TOP>
<TOP> ::= 08"NOP 0"        {POP} <TOP>
<TOP> ::= 09"NOP 1"        {POP} <TOP>
<TOP> ::= 0A"NOP 2"        {POP} <TOP>
<TOP> ::= 0B"NOP 3"        {POP} <TOP>
<TOP> ::= 0C"FLUSH 0"      not-reading {POP} <TOP>
<TOP> ::= 0D"FLUSH 1"      not-reading {POP} <TOP>
<TOP> ::= 0E"FLUSH 2"      not-reading {POP} <TOP>
<TOP> ::= 0F"FLUSH 3"      not-reading {POP} <TOP>
<TOP> ::= 10"WRITE PC"     not-writing {POP} <TOP>
<TOP> ::= 11"READ PC"      go/stop {POP} <TOP>
<TOP> ::= 12"RD ERR ADR"   go/stop {POP} <TOP>
<TOP> ::= 13"RD ERR RPT"   go/stop {POP} <TOP>
<TOP> ::= 14"DIS CACHE"    {POP} <TOP>
<TOP> ::= 15"EN CACHE"     {POP} <TOP>
<TOP> ::= 16"RD W/LOCK"    go/stop {POP} <TOP>
<TOP> ::= 17"RESERVED"     {BROKEN}
<TOP> ::= 18"WRITE-BACK"   go/stop {POP} <TOP>
<TOP> ::= 19"WRITE BASE"   go/stop {POP} <TOP>
<TOP> ::= 1A"WRITE ECC"    go/stop {POP} <TOP>
<TOP> ::= 1B"WRITE LIMIT"  go/stop {POP} <TOP>
<TOP> ::= 1C"RESERVED"     {BROKEN}
<TOP> ::= 1D"WRITE DATA"   go/stop {POP} <TOP>
<TOP> ::= 1E"RD UNCORR"    go/stop <OPERAND> {POP} <TOP>
<TOP> ::= 1F"RESERVED"     {BROKEN}

<OPERAND> ::= operand/
<OPERAND> ::= operand go/stop

<PAUSE> ::= NULL, "unconditionaly wait for 1 clock"
```

Fig. 6-2 Input Interface State Transition Table

6.1.2 BASE/LIMIT TABLE

The base/limit table is composed of two 16 word dual port register files. One register file contains the base table entries and the other contains the limit table entries. Both tables are addressed by the base indicant field from the processor. Each entry of the base and limit tables contain 4 items. These items are:

Binary Base (27 bits)
: Loaded from the output of the BCD to binary converter on a write base or write limit command. This is the absolute binary base address (mod 10) of the memory area.

Binary Limit (27 bits)
: Loaded from the output of the BCD to binary converter on a write base or write limit command. This is the absolute binary limit address (mod 10) of the memory area.

Base Parity (1 bit)
: The base address parity bit is calculated such that the overall parity of the absolute binary limit address and the base parity bit is even. This increases possibility of detecting uninitialized base or limit table entries. The output of the base and limit tables are only checked for a parity error on commands which require base addition.

Limit Parity (1 bit)
: The limit address parity bit is calculated such that the overall parity of the absolute binary limit address and the limit parity bit is even. This increases possibility of detecting uninitialized base or limit table entries. The output of the base and limit tables are only checked for a parity error on commands which require base addition.

6.1.3 BASE ADDER AND BCD TO BINARY CONVERTER

The Base adder and BCD to binary converter is a single MCA II gate array chip. It is composed of a whole series of wallace tree adders which convert the input BCD address to absolute binary. The binary addition is accomplished by including the absolute binary address from the base/limit table into the inputs of the final stage of wallace tree adders.

Because of the word oriented cache and memory storage structure, only the most significant 8 digits of the address are converted to a 27 bit binary number. The least significant digit of the address is left alone and remains BCD. This is sufficient since it is only used by the cache and memory control to synthesize partial word operations. Throughout the remainder of the memory control, this artificial division between the 8 most significant digits and the least significant digit is retained. The 27 bit binary part is called the address (though it is more accurately called the "word address"), and the remaining digit is called the LSD (least significant digit).

6.1.4 LIMIT CHECKER

The limit checker compares the absolute binary block address from the base adder and BCD to binary converter with the limit address from the limit table. Limit errors are reported to the cache controller.

A binary subtractor is used to check the address with the limit. The 27 bit result is converted into a 7 bit Hamming Code with the same pattern as used for the 32 bit main memory address bus. Fault detection of the base add and limit checking is provided by duplicating the base adder and limit checker with each limit checker comparing its own 7 bit code with the one from the other checker.

The limit checker also provides the path to read the limit table. The limit table output is routed through onto the absolute address bus through bidirectional pins. This bus goes to the cache section and the error recorder. The error recorder provides the front-plane path into the data section. limit checker comparing its own 7 bit code with the one from the other checker.

6.2 MEMORY REQUEST AND ACCESS CONTROL

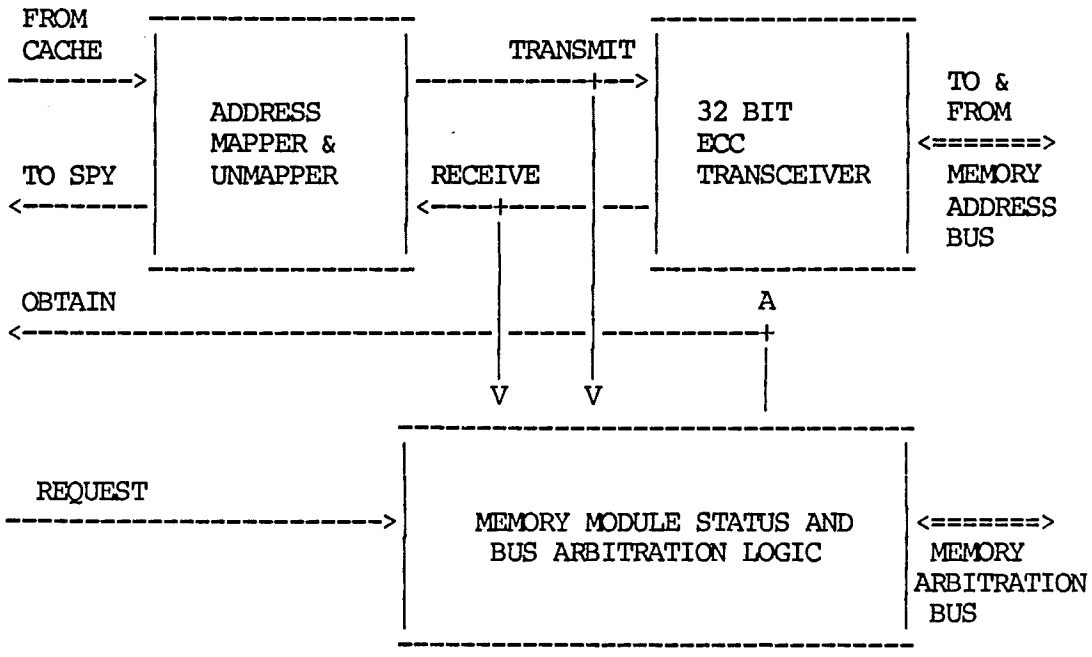


Fig. 6-3 Block Diagram of Memory Request and Access Control

The memory request and access control is the interface between cache and main memory. The cache interface is described in Section 7.1 and the main memory interface is described in Section 5.2. The memory request and access control selects memory data cards for operations as dictated by the cache control section.

6.2.1 MEMORY MODULE STATUS

There is a busy register for each of the 16 possible memory data cards. This register is loaded on each bus cycle, and it contains the refresh status of each MDC. A memory module is available for request if it is not busy refreshing, if it was not requested on the last bus cycle, and if it is not being requested now.

6.2.2 ADDRESS MAPPER

The address mapper converts the 25 bit absolute block address from the BCD to binary converter into a 22 bit memory data card address and 5 bits of module select. The mapping algorithm is an "any-number" memory interleaving scheme. The memory data cards are interleaved to maximize bandpass with "any" number of cards plugged into the V500 System backplane. Figure 5-14 shows the possbile interleaving configurations for 1 to 16 cards.

The address mapper also creates the psuedo module selects required by the IOMC's for the read and write I/O commands. The address of the requested I/O hardware register is contained in the address field of the read or write I/O command from the processor. The address mapper convets this into a pseudo module select for these commands.

6.2.3 ADDRESS BUS ECC TRANSCEIVER

The memory address bus is protected by 7 bits of error correct code as defined in Section 5.2.1. These codes are generated and checked in the ECC transceiver section. Single bit errors detected on the memory address bus are corrected for the cache spy. Double bit errors detected on the memory address bus causes a system broken stop condition.

The address bus ECC transceiver also contains the gated 25 ohm drivers for the address, module select, command, and requestor ID fields of the memory address bus. These drivers are enabled by the memory bus arbitration logic.

The Hamming Code which has been created (see Figures 5-10 and 5.11) has axial symmetry. This allows a pair of identical ECC chips to handle the required bidirectional 25 ohm backplane lines. An unsymmetric code would require 2 separate options in order to drive the necessary number of lines.

6.3 DATA SECTION

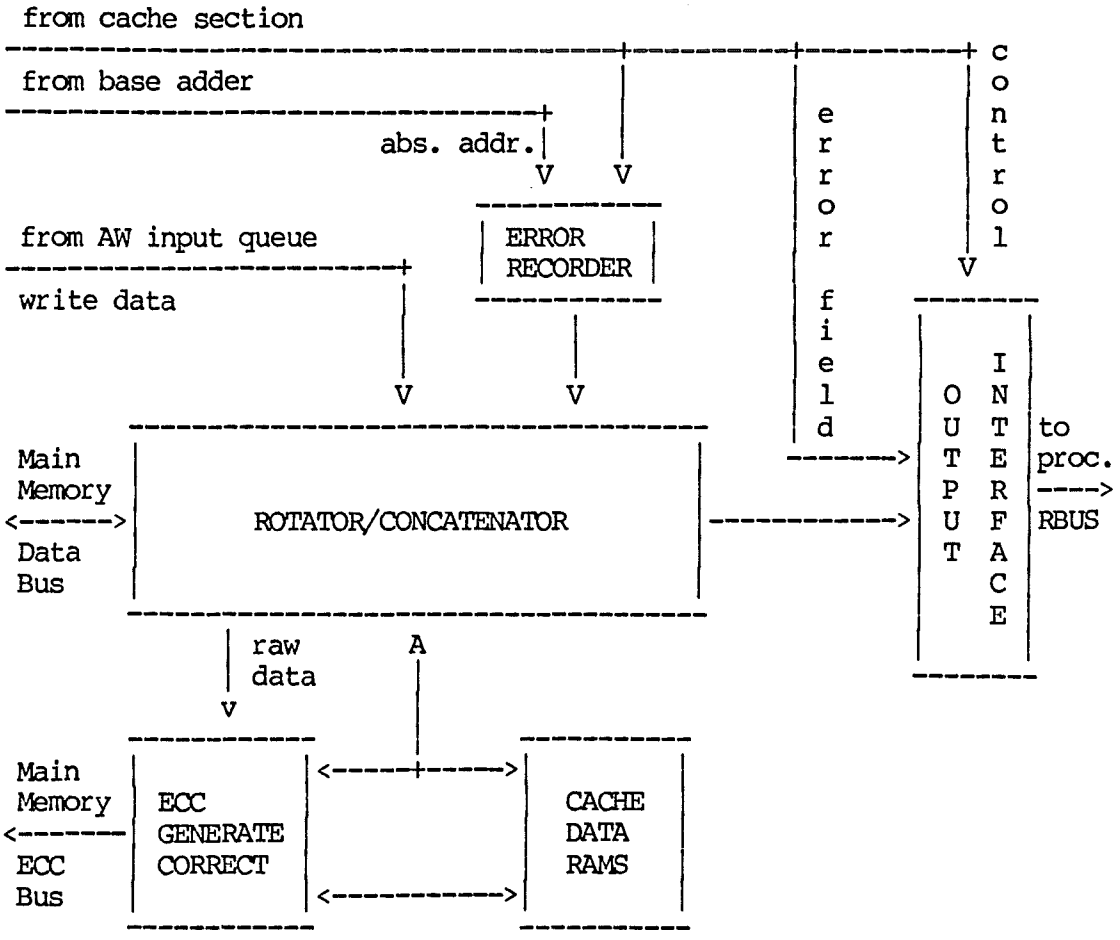


Fig. 6-4 Block Diagram of MCACM Data Section

The data section of the memory control is responsible for synthesizing the partial reads and writes that the processor has requested. The data section receives requested data from memory, corrects single bit errors, detects and reports single and multiple bit errors, rotates and concatenates data from memory to the processor modules, and provides the interface through which cache

sends data back to the processor.

6.3.1 ROTATOR/CONCATENATOR

The rotator and concatenator synthesizes the partial word operations for the memory control and cache module. The rotator and concatenator contains the data paths, rotators, maskers, and registers necessary to implement the MCACM data paths.

The rotator and concatenator handles the "operand read" function exactly like a non-operand read. The cache section drives the upper and lower words of the cache data memory with the appropriate addresses so that the output contains all of the requested data.

The rotator and concatenator also has an input from the error recorder. This path is also used to select the cache address bus which is used to handle base/limit table reads.

6.3.2 ERROR DETECTOR AND CORRECTOR

The data section error detection and correction logic contains the ECC generator and error corrector for the main memory data bus. This logic also handles ECC generation and error detection and correction for the cache data RAM.

The primary purpose of any memory subsystem is to read and write data accurately without error. Therefore, the memory subsystem attempts to correct errors that are common in large main memory structures. A modified SEC-DED-4ED (single error correct, double error detect and some 4-bit error detect) Hamming code is the primary error detection and correction tool. To further aid error detection in main memory, the ECC (error correction code) detects all triple-bit and quadruple-bit errors that occur within any single digit. This code provides maximal protection for the 512Kx4 bit dynamic RAM SIPS (single in-line packages) that are being used for the main memory structure.

The fault tolerance of data is insured by keeping ECC with data. The only time that ECC is generated is when data is written into the cache data RAMs. The process of generating ECC is protected by duplicate and compare. The data to be written is duplicated, ECC is generated on both copies, and the ECC codes are compared for equality. This ECC is maintained in the cache and main memory RAMs: it is corrected after each transfer across the main memory bus.

6.3.3 OUTPUT INTERFACE

The output interface registers the output of the rotator and concatenator, and contains the 25 ohm drivers for the RBUS. The tag associated with a request is sent 1 clock cycle before the data. This tag specifies certain processor module dependent information about the data. Certain cache miss conditions can result in the tag being sent and then no data being available to accompany it. When this happens, the data valid flag is used to signal the processor that the expected data is not available yet.

6.4 ERROR RECORDER

When an error is flagged in the error information field to the processor, the execute module requests a read error report and a read error address. The read error report returns the module in which an error occurred, the bit in error, and whatever else is appropriate to pin-point the actual error. The read error address returns the absolute binary address of the error. This information is obtained from the error recorder.

The error recorder monitors the activities of the memory control and records necessary information about each operation. When an error actually occurs, the specific information about that error and the operation being performed are captured. This information is read by the read error report and read error address commands.

The error recorder captures errors on a first-worst basis. On this basis, a double bit error is worse than a single bit error, therefore, a double bit error occuring after a single bit error (but before a read error report command) overwrites the single bit error information. The first double bit error is captured and is not overwritten by subsequent errors.

7 CACHE DESCRIPTION

The memory control and cache module contains a 81,920 digit (40 Kbyte) cache memory. The cache memory is organized in 1024 sets. Each set contains two 40 digit blocks. The V500 System has a local cache for each processor.

The cache algorithm is forced to minimize bus traffic because of the shared main memory bus. Therefore, the cache algorithm is a variant of the write-back algorithm.

7.1 INTERNAL CACHE INTERFACES

The cache section interfaces with the AW input queue, the memory control section, the data section, and the R-Bus registers. These interfaces are described below.

7.1.1 INTERFACE TO AW INPUT QUEUE

This interface is composed of 2 parts: address and control. The address portion of this interface consists of the following signals.

ADDRESS (27 bits)
: The absolute binary address of the request. This address has the base added and is checked against the limit.

ADDRESS PARITY (1 bit)
: The address parity bit is calculated so that the overall parity of the address and the address parity bit is even.

LSD (4 bits)
: The least significant digit is passed directly from the AW input queue. It is not sent through the base adder nor is it included in the limit check.

LEN (4 bits)
: The length of the request as received on the AW-BUS from the processor.

7.1.1 INTERFACE TO AW INPUT QUEUE (Continued)

LSDLEN PARITY (1 bit)
The least significant digit and length parity bit is calculated so that the overall parity, including the parity bit, is even.

The cache control interface is divided into those signals which originate at the AW queue controller and are sent to cache, and those signals which originate at cache and are sent to the queue. The control interface to the cache section consists of: a command, a valid bit, an error bit, and a command parity bit. The return interface from the cache section consists of 3 handshake signals: hold-queue, reading, and writing. The control interface to the cache section is described on the following pages.

COMMAND (5 bits)
The command is mapped by the MAWQUE array to contain the operand-read information. Only the commands which cache must execute are passed to cache. The remaining commands are handled by the MAWQUE array. The table in Figure 7-1 shows the commands which are passed to the cache section. The commands listed as "internal" are reserved in the cache for more mappings; these are commands which the cache does not need unless they contain errors. A column is included here for the read/write grouping of each command.

The "00" and "1F" codes are generated by the AW input queue controller. They are derived from the command on the AW-Bus by inverting the least significant bit. These commands are only used internally by the cache section to determine the existence of an operand read. Operand reads are defined as read operations which cross block (40 digit memory word) boundaries. The AW input queue controller detects the operand read case and changes the command field, it also splits the command into two separate commands for the cache section. The data from these two commands is spliced together before being sent over the R-Bus.

7.1.1 INTERFACE TO AW INPUT QUEUE (Continued)

CODE	COMMAND DESCRIPTION	READ/WRITE/NONE
00	Operand Read	Read
01	Non-Operand Read	Read
02	I/O Write	Write
03	I/O Read	Read
04	-internal-	None
05	Read Base	Read
06	Read ECC	Read
07	Read Limit	Read
08	-internal- {nop}	None
09	-internal- {nop}	None
0A	-internal- {nop}	None
0B	-internal- {nop}	None
0C	-internal-	None
0D	-internal-	None
0E	-internal-	None
0F	-internal-	None
10	Write PC	Write
11	Read PC	Read
12	Read Error Address	Read
13	Read Error Report	Read
14	-internal-	None
15	-internal-	None
16	Read With Lock	Write
17	-internal-	None
18	Write Back	Write
19	-internal-	None
1A	Write ECC	Write
1B	-internal-	None
1C	-internal-	None
1D	Write Data	Write
1E	Non-Operand Read-Uncorrected	Read
1F	Operand Read-Uncorrected	Read

Fig. 7-1 Internal Cache Command Decoding Table

7.1.1 INTERFACE TO AW INPUT QUEUE (Continued)

The read/write grouping, in Figure 7-1, specifies which of the CACHE READING or CACHE WRITING handshake signals back to the MAWQUE array will be activated by each command. This grouping also determines which commands will get flushed (not sent back on the R-Bus). The NONE group is only valid in this context when the command or data contained an error, and is treated as "read" during the flush operation.

The "08" through "0B" commands are used internal to the cache section to indicate non-active pipeline registers. Any command which has been finished may get translated into one of these codes if the internal pipeline is allowed to empty.

VALID (1 bit)
The output of the AW queue and therefore the address section contains a valid command that is ready for the cache section to handle. It is possible for a command to be present but not-valid because the cache is not ready. An example of this is the WRITE-PC command which is held until the cache is not CACHE WRITING.

ERROR (1 bit)
The error bit signifies that a fatal error has been detected by the AW input queue controller. These errors are not fatal to a system, but only to the request in which they were requested. The AW input queue controller generates an error vector which is sent to the error recorder and later returned on the R-BUS so that the processor can determine what caused the error.

COMMAND PARITY (1 bit)
The command parity bit is calculated such that the overall parity of the command, the valid bit, the error bit, and the command parity bit is even.

7.1.1 INTERFACE TO AW INPUT QUEUE (Continued)

The return interface to the AW queue controller consists of 3 handshake signals: holdque, reading, and writing. These signals are described below.

HOLDQUEL(1 bit)

The cache section is full or unable to handle another command. This can be caused by outstanding main memory requests or spy interactions. When HOLDQUEL is active, the AW input queue is not popped and remains valid until HOLDQUEL is removed.

CACHE READING (1 bit)

The CACHE READING handshake signal is sent to the AW queue controller when the cache is working on an operation which is defined as a read, see the table in Figure 7-1. The FLUSH operation must be held in the AW queue until all outstanding main memory reads have been finished so that the R-Bus output can be enabled without sending "flushed" requests to the processor. The CACHE READING handshake signal is used to determine this condition. The AW queue controller also uses CACHE READING in conjunction with CACHE WRITING to determine when the cache is idle.

CACHE WRITING (1 bit)

The CACHE WRITING handshake signal is sent to the AW queue controller when the cache is working on an operation which is defined as a write, see the table in Figure 7-1. The WRITE-PC operation is defined to wait until the cache is finished all outstanding writes before being executed. The AW queue controller uses the CACHE WRITING handshake to determine this condition. The AW queue controller also uses CACHE WRITING in conjunction with CACHE READING to determine when the cache is idle. When the cache is either reading or writing, it is not idle. It is probably self-evident that the cache is idle if it is neither reading nor writing. When the cache is idle and there are no requests in the AW queue, then the MCACM is idle.

7.1.2 INTERFACE TO DATA SECTION

The cache data interface is the primary interface between the processor and the memory subsystem. All processor reads or writes are handled by cache. When a cache miss occurs, the memory control requests the missing block for the cache.

WRITE DATA (40 bits)
The write data field as sent from the processor.

CACHE UPDATE DATA (192 bits)
This interface is for the return data from main memory to update or fill a block in cache. It has been corrected but not rotated or concatenated. See the block diagram in Figure 5-6.

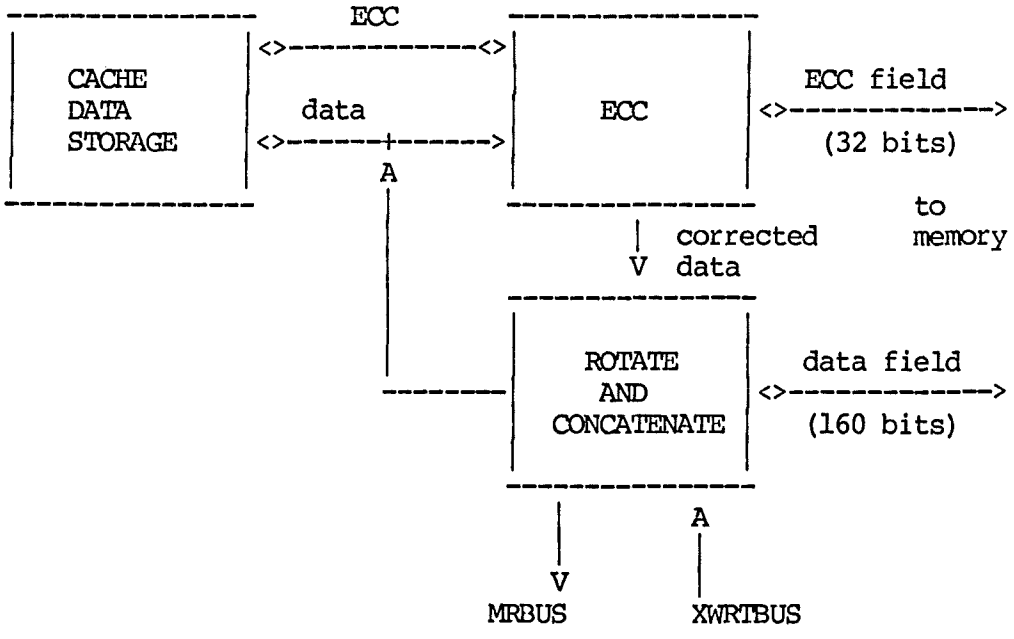


Fig. 7-2 Block Diagram of Cache Data Interface

7.1.3 MEMORY REQUEST INTERFACE

When main memory action is required, the cache section commands the memory request section to obtain the needed memory module. The cache signals the type of memory operation required with the following interface.

REQUEST
Request a memory or I/O access based on the memory/IO and command.

MEMORY/IO
This signal selects memory or I/O access.

COMMAND[1:0]

00 READ-MODIFY-WRITE: This command is not used by cache.

01 WRITE: A modified block of data is returned to main memory. This command is used before NORMAL READ or READ PRIVATE in response to a miss when the block to be updated contains modified data.

10 NORMAL READ: The returned data may be either writeable or read-only. This command is used in response to a miss on a processor read request.

11 READ PRIVATE: This operation guarantees that the returned data is writeable and that there are no other copies in the system. This command is used in response to a miss on a a processor write request.

PARITY
This parity is calculated such that the parity over the REQUEST, MEMORY/IO, COMMAND and PARITY lines is even.

7.1.3 MEMORY REQUEST INTERFACE (Continued)

The data that is returned from a read operation may have different states due to the distributed cache architecture. The data's status is sent to the cache control. This status interface is as follows.

MODIFIED
The returned data is already modified, therefore it has the writeable status. Section 7 describes the details of read-only and writeable.

COPY
The returned data already exists in some other cache, therefore it is read-only.

7.1.4 SPY INTERFACE

The spy interface to the cache contains the information required by the cache to determine if the memory bus request is a hit into locally cached data. The memory bus command is also provided so cache can give the proper response when a hit is detected. The information contained in the spy interface is:

ADDRESS (25 bits)
The absolute binary block address of the memory bus request. This binary address is derived by the memory mapper/unmapper portion of the memory bus access control logic.

ADDRESS PARITY (1 bit)
The address parity bit is calculated such that the overall parity of the address and the address parity bit is even.

SPY-COMMAND (2 bits)
The spy-command is encoded from the command field on the memory backplane. The four types of commands which are encoded into the spy command field are:

When there is no request on the main memory bus, then the command is forced to the "0" value.

0 IOMC read (public).

1 MCACM read (public).

2 IOMC write.

3 Any requestor read (private).

SPY-VALID (1 bit)
This signal is an enable bit for the spy logic, it is active whenever a memory request is detected on the memory backplane.

SPY-COMMAND PARITY (1 bit)
The spy-command parity bit is calculated such that the overall parity of the spy-command, the spy-valid, and the spy-command parity bit is even.

7.2 WRITE-BACK ALGORITHM

7.2.1 OVERVIEW

The write-back algorithm, which is implemented in the V500 processor, allows syncronization of each memory control and cacne module while not arbitrarily constricting I/O devices attached to the memory bus.

The basic philosophy of the write-back algorithm is that only a single copy of writeable data is allowed to exist at any time. Many copies of "read-only" data may exist. These 2 conditions are sufficient to prevent multiple processors and/or I/O from changing data simultaneously with unpredictable results. Each memory control and cache module maintains status information about the blocks of memory that it has cached. There are 4 possible states that a cached block can have:

1. not valid.
2. valid data with other possible copies of the same data.
3. valid data, no other copies, except in main memory.
4. valid modified data without any other copies in existence. This data does not exist anywhere else, so it must be written back to main memory.

These four states are called: not-valid, copy (also called "read-only"), private (also called "writeable"), and modified. These four states of cache are shown in Figure 7-3.

7.2.1 OVERVIEW (Continued)

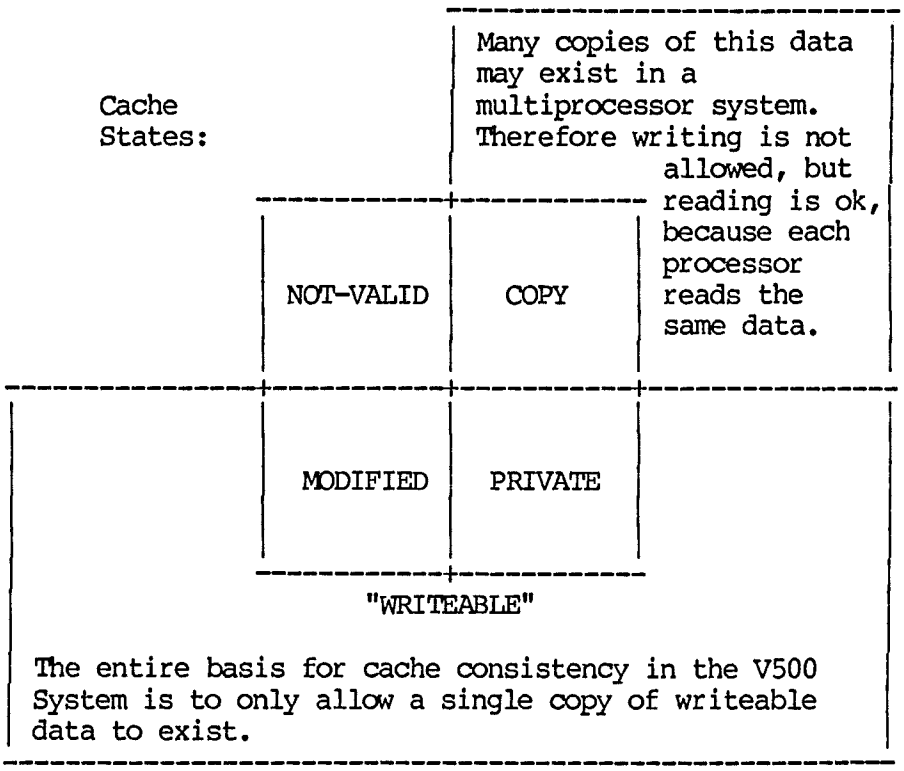


Fig. 7-3 Four States of Cached Data

When a locally cached block of data with a status of private gets modified, the status of the block is changed to modified. The modified block is not written immediately back to main memory, but remains locally cached. This is done to reduce memory accesses on the single port to memory. Since there are several separate cache modules changing and holding data in this manner, it is necessary to prevent more than one cache from changing the same data at the same time. This introduces the subject of cache consistency and the requirements of a spy device.

7.2.1 OVERVIEW (Continued)

When locally cached data is modified by MCACM, the main memory no longer contains a "real" copy. If another MCACM or an IOMC requests this block from main memory, the spy (which is discussed in Section 7.3) must detect the memory request and supply the "real" data from the local cache. When the spy supplies data, the memory data cards are inhibited, and the modified data is transferred directly from the MCACM to another MCACM or an IOMC. The requesting module does not know or care where the data came from, but only that it now has valid data. As an aside, the error recorder does maintain the information about the source module when an error occurs. The sourcing MCACM then invalidates the block in its local cache. Future writes to this block, must first acquire it in its private or modified form again.

The overall concern of any write-back cache mechanism is to only allow a single copy of writeable data in the entire system, and to force all accesses for that block to get the currently valid contents. The cache directory and spy sections contain the logic to accomplish this goal.

7.2.2 DETAILED DESCRIPTION

When requested data is contained in the cache, it is easily handled. When requested data is not contained in the cache, or is not in the proper form (i.e., read-only data with a a write command), then the actions of MCACM are more complicated. There are 5 different possibilities to consider.

READ MISS WITHOUT SOURCE CACHE -
This is the simplest miss condition. The requested data is not in the local cache or any other caches. Data is read from main memory and returned as PRIVATE.

READ MISS ON SHARED DATA -
When another cache contains an unmodified copy of the requested data, the data is read from main memory. Both caches mark this data as COPY.

READ MISS WITH SOURCE CACHE -
When another cache contains a modified copy of the requested data, the data is transfered from cache to cache. The requesting cache marks the data as MODIFIED.

WRITE MISS WITHOUT SOURCE CACHE -
This is similar to READ MISS WITHOUT SOURCE CACHE. The returned data is PRIVATE so it can be modified and marked as MODIFIED.

WRITE MISS ON SHARED DATA -
When another cache contains a copy of the requested data, the data is relinguished to the requesting cache. If the data was not MODIFIED, it is simply marked as INVALID and the data is returned from main memory. If the data was MODIFIED, then it is transferred from cache to cache. The requesting cache marks the data as MODIFIED.

7.3 SPY MECHANISM

Each cache contains a spy which monitors the activities of all memory requestors to insure that only a single copy of writeable data is allowed to exist. The spy mechanism maintains cache consistency by globally enforcing the write-back algorithm. For each type of main memory request, whether from another MCACM or an IOMC, there is a proper response which is required to maintain cache consistency.

The proper response is dependent on the type of request (read private, read public, or write), the requestor (MCACM or IOMC), and the state of data in the local cache (private, copy, or modified). These responses are not easily written out in a sensible manner, so the table in Figure 7-4 shows the proper response of a MCACM spy to a detected request from another MCACM or an IOMC. There are 5 columns which show: the source (MCACM or IOMC), the type of request, whether it is a hit or miss into local cache, the status of data (in the case of a hit), and the proper response. The first entry is interpreted as another MCACM making a public read request on data that is currently cached as copy. The response to this first case is to do nothing, which allows the memory data card to source the data.

7.3 SPY MECHANISM (Continued)

REQUEST SOURCE	TYPE OF REQUEST	HIT/ MISS	CACHE STATUS	LOCAL CACHE ACTIONS
MCACM	READ PUBLIC	HIT	COPY	NONE.
		HIT	PRIVATE	CHANGE BLOCK STATUS TO COPY.
		HIT	MODIFIED	RETURN DATA WITH MODIFIED FLAG AND THEN INVALIDATE BLOCK.
		MISS	ANY	NONE.
MCACM	READ PRIVATE	HIT	COPY	INVALIDATE BLOCK.
		HIT	PRIVATE	INVALIDATE BLOCK.
		HIT	MODIFIED	RETURN DATA WITH MODIFIED FLAG AND THEN INVALIDATE THE BLOCK.
		MISS	ANY	NONE.
MCACM	WRITE	HIT	ANY	NONE, THIS COULD ONLY BE CAUSED BY AN ERROR REQUIRING MORE THAN A SINGLE POINT OF FAILURE.
		MISS	ANY	NONE.
IOMC	READ PUBLIC	HIT	COPY	NONE.
		HIT	PRIVATE	NONE.
		HIT	MODIFIED	RETURN DATA WITH MODIFIED FLAG DO NOT INVALIDATE THE BLOCK.
		MISS	ANY	NONE.
IOMC	READ-	HIT	COPY	INVALIDATE THE BLOCK.
	MODIFY-	HIT	PRIVATE	INVALIDATE THE BLOCK.
	WRITE-	HIT	MODIFIED	RETURN DATA WITH MODIFIED FLAG AND THEN INVALIDATE THE BLOCK.
		MISS	ANY	NONE.
IOMC	WRITE	HIT	ANY	INVALIDATE THE BLOCK.

Fig. 7-4 Cache Actions for Spy Detected External Requests

7.3 SPY MECHANISM (Continued)

The spy actions for IOMC requests depend on the IOMC acting in a predictable manner.

When the IOMC is doing a memory read operation (ie. reading from memory and writing to disk), the IOMC must use the READ PUBLIC command. A request for public data is appropriate here, because we do not trust an I/O operation to complete sucessfully. If the IOMC was allowed to get private data, a MCACM whose spy had sourced this data would mark it invalid. Later, if the I/O operation failed and was retried, the "real" data would not exist.

When the IOMC is doing the read portion of a read-modify-write operation (ie. the first or last word when reading a sector from disk and writing to memory), the IOMC must use the READ-MODIFY-WRITE command. This is allowable since the IOMC already has the necessary data, with which to modify the block, and writes it back to main memory "post haste".

These 2 simple rules are necessary and sufficient to maintain cache consistency and preserve data integrity.

7.4 DIRECTORY SECTION

7.4.1 OVERVIEW

The directory section of the V500 cache accommodates requests from 2 simultaneous sources: its own processor and the other devices on the main memory bus. Additionally, the cache must handle data that is returning from the main memory structure in response to a miss.

The four states of a block are kept in a type of valid-bit RAM structure. The most significant 15 bits of the absolute binary address are also stored in the key RAM.

When a block of data has been updated by data returned from main memory, it is marked valid. Both the key and the validity bit are changed to valid. The key RAM is loaded with the most significant 15 bits of the address of the block of data that is now cached and the state of that block is updated.

The table in Figure 7-5 shows the actions of the memory control and cache module for requests from the processor attached to the AWBUS and RBUS (not through other MCACM's across the memory backplane). The first entry is interpreted as a read request to a block that is currently cached with a copy status (copy also means "read-only"). The action of this request is to return the requested data to the processor.

7.4.1 OVERVIEW (Continued)

TYPE OF REQUEST	HIT/ MISS	CACHE STATUS	CACHE ACTIONS
READ	HIT	COPY	RETURN DATA TO PROCESSOR.
READ	HIT	PRIVATE	RETURN DATA TO PROCESSOR.
READ	HIT	MODIFIED	RETURN DATA TO PROCESSOR.
READ	MISS	N/A	READ (PUBLIC) BLOCK FROM MAIN MEMORY UPDATE CACHE WITH THIS BLOCK IF BLOCK RECEIVED WITH MODIFIED FLAG THEN THE STATUS OF THIS NEW BLOCK IS MODIFIED ELSE IF OTHER COPIES EXIST IN SYSTEM THEN THE STATUS OF THIS NEW BLOCK IS COPY ELSE /* NO OTHER COPIES EXIST */ THE STATUS OF THIS NEW BLOCK IS PRIVATE RETURN DATA TO PROCESSOR.
WRITE	HIT	COPY	READ (PRIVATE) BLOCK FROM MAIN MEMORY UPDATE CACHE WITH MERGED WRITE-DATA THE STATUS OF THIS NEW BLOCK IS MODIFIED.
WRITE	HIT	PRIVATE	READ, MODIFY AND WRITE THE CACHED BLOCK CHANGE THE STATUS OF THIS BLOCK TO MODIFIED.
WRITE	HIT	MODIFIED	READ, MODIFY AND WRITE THE CACHED BLOCK.
WRITE	MISS	ANY	READ (PRIVATE) BLOCK FROM MAIN MEMORY UPDATE CACHE WITH MERGED WRITE-DATA THE STATUS OF THIS NEW BLOCK IS MODIFIED.

Fig. 7-5 Cache Actions for Processor Requests

7.4.2 DETAILED DESCRIPTION

The directory section of the MCACM cache is based around 3 pipeline registers. These 3 pipeline registers are called T1, T2, and T3. The 3 cache pipeline registers are shown in Figure 7-6 below.

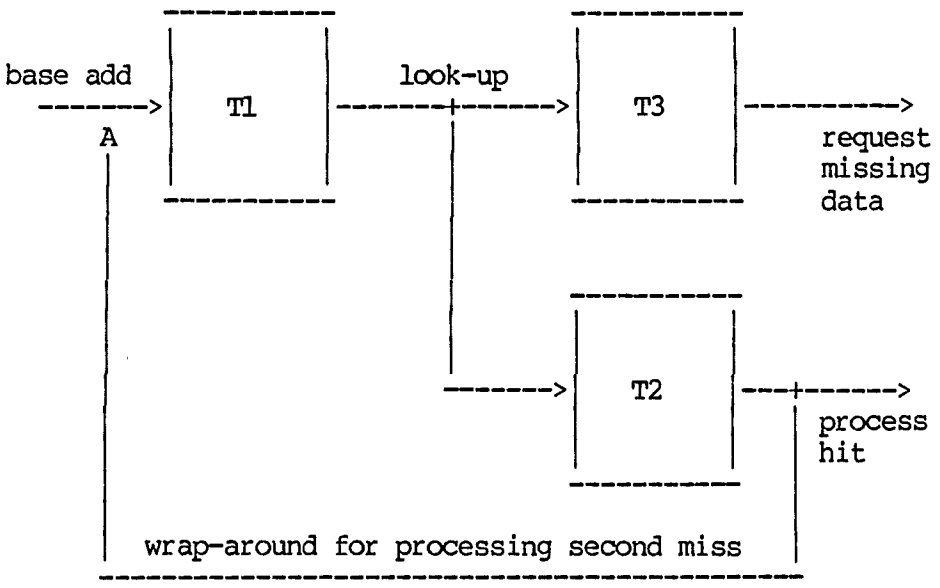


Fig. 7-6 Cache translator-pipeline registers

When a command is valid at the top of the AW Input Queue, its address is added to the appropriate base and converted to binary. This binary address is loaded into T1.

The output of T1 is used to check for hit or miss. The least-significant 10 bits of the address in T1 drives the directory RAM's. The most-significant 15 bits or the address in T1 are compared to the KEY stored in the directory RAM's.

7.4.2 DETAILED DESCRIPTION (Continued)

If the result of the comparison with T1 is a hit, then the command (and address, etc.) are loaded into the T2 register. If the result is a miss, then the command (and address, etc.) need to be loaded into T3 to process the miss. When T3 is empty, T2 and T3 can be loaded at the same time, only the appropriate one (T2 on a hit, and T3 on a miss) is used. But, when T3 already contains a miss, the command et. al. is wrapped around from T2 into T1.

When the miss in T3 is finished being processed, any commands which are circulating in T1 and T2 can then be loaded from T1 into T3. This allows 2 misses to be pending without stopping the operation of MCACM. If a 3rd miss is detected, then all 3 pipeline registers contain a miss: 1 being processed in T3, and 2 being circulated between T2 and T1, so no more processing can be done until the miss in T3 is finished.

The logic to accomplish these functions is contained in the CACNTL5, CAFLAG5, and CHADDR3 arrays which are described in detail in Section 12.

8 TIMING

This section contains timing diagrams which depict the timing relationships of the different operations and sections of the memory control and cache module.

In each of these timing diagrams, the clock is shown across the top for reference. The decision edge of the clock is the rising edge, which is standard throughout the V500 System.

8.1 ADDRESS SECTION TIMING OF READ OPERATION

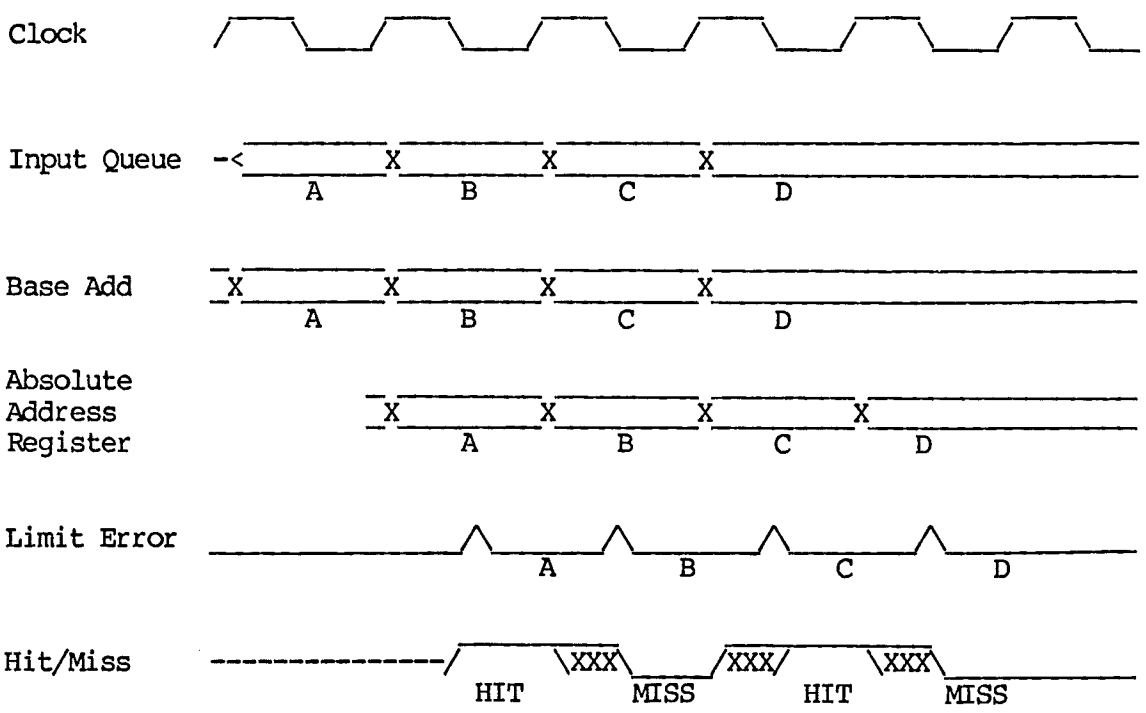


Fig. 8-1 Timing of Address Section (Reads)

8.2 ADDRESS SECTION TIMING OF WRITE OPERATION

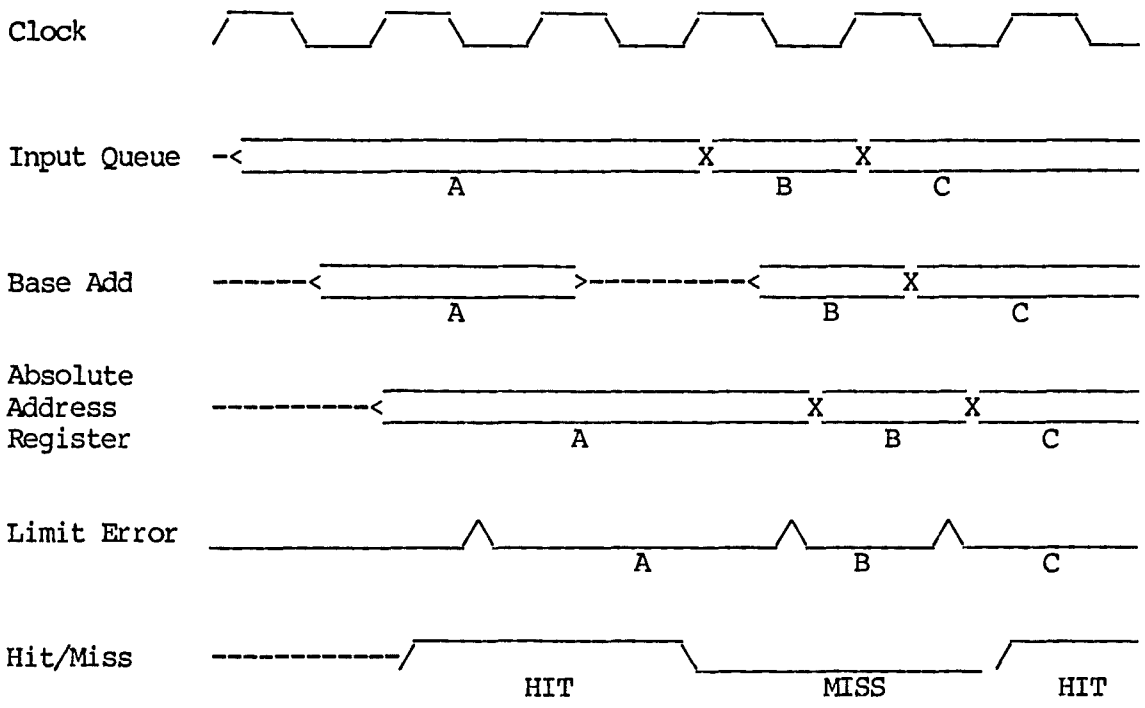


Fig. 8-2 Timing of Address Section (Writes)

8.3 CACHE READ TIMING

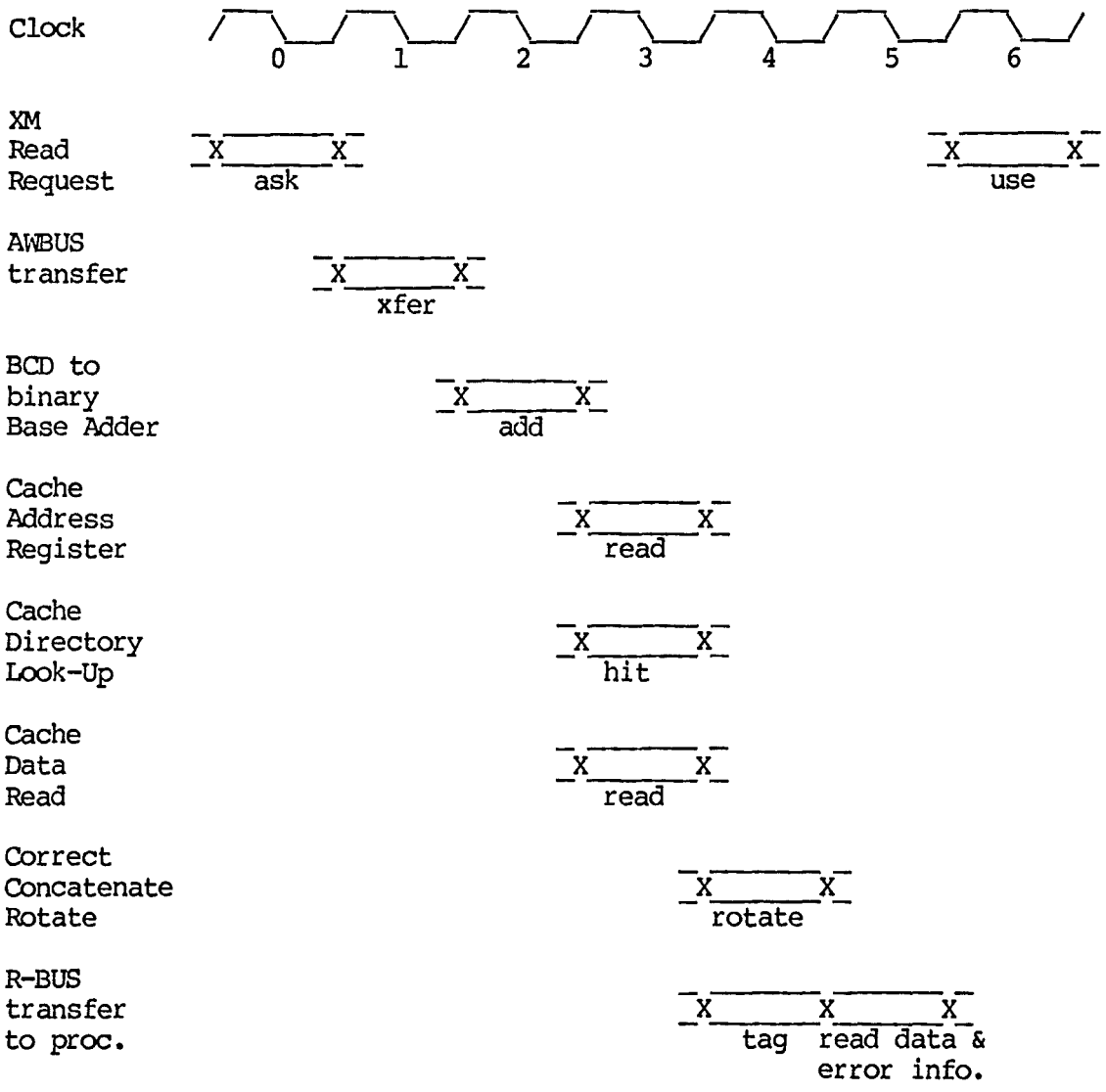


Fig. 8-3 Timing of Read Hit

8.4 CACHE WRITE TIMING

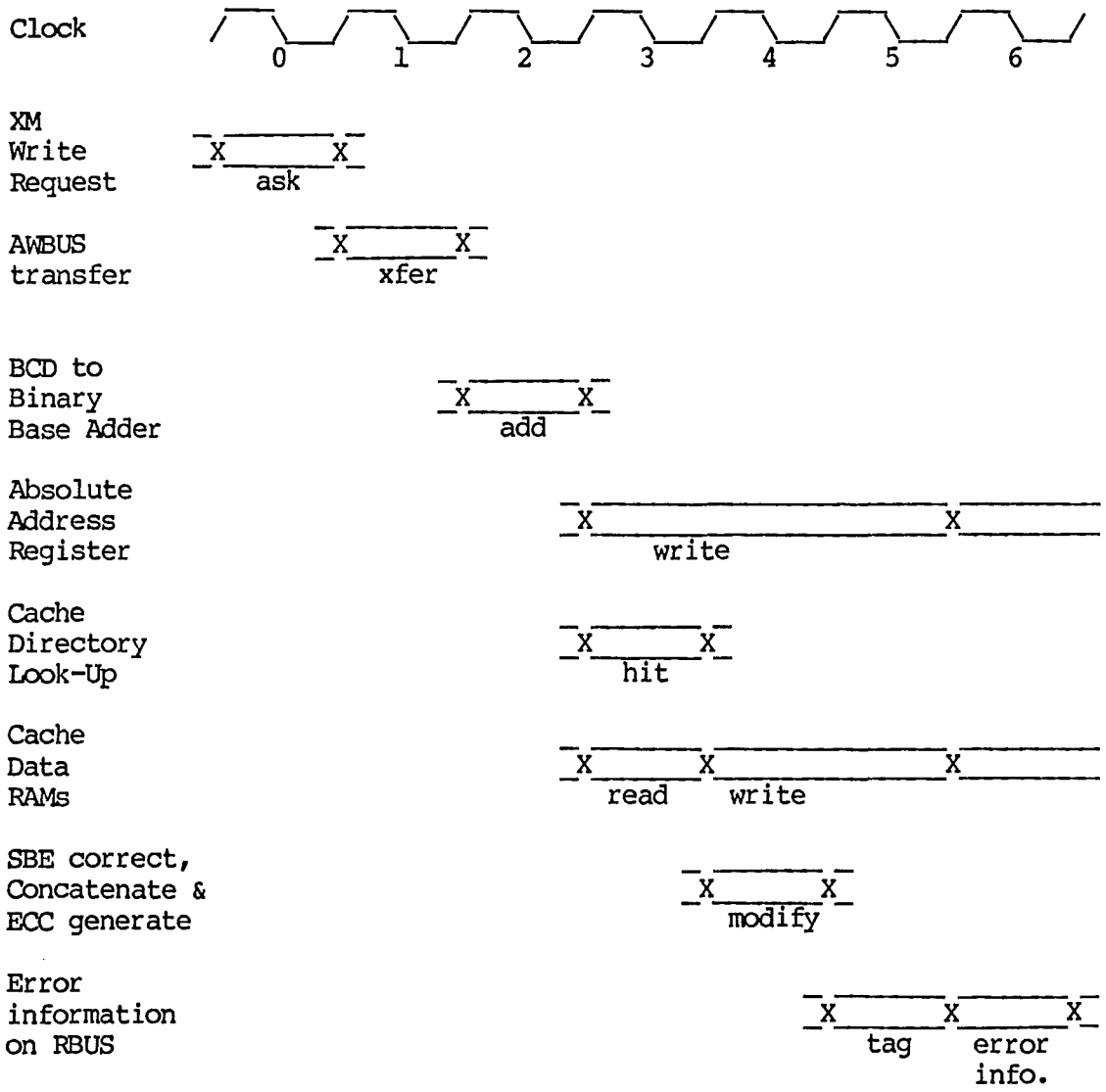


Fig. 8-4 Timing of Write Hit

8.5 CACHE READ (MISS)

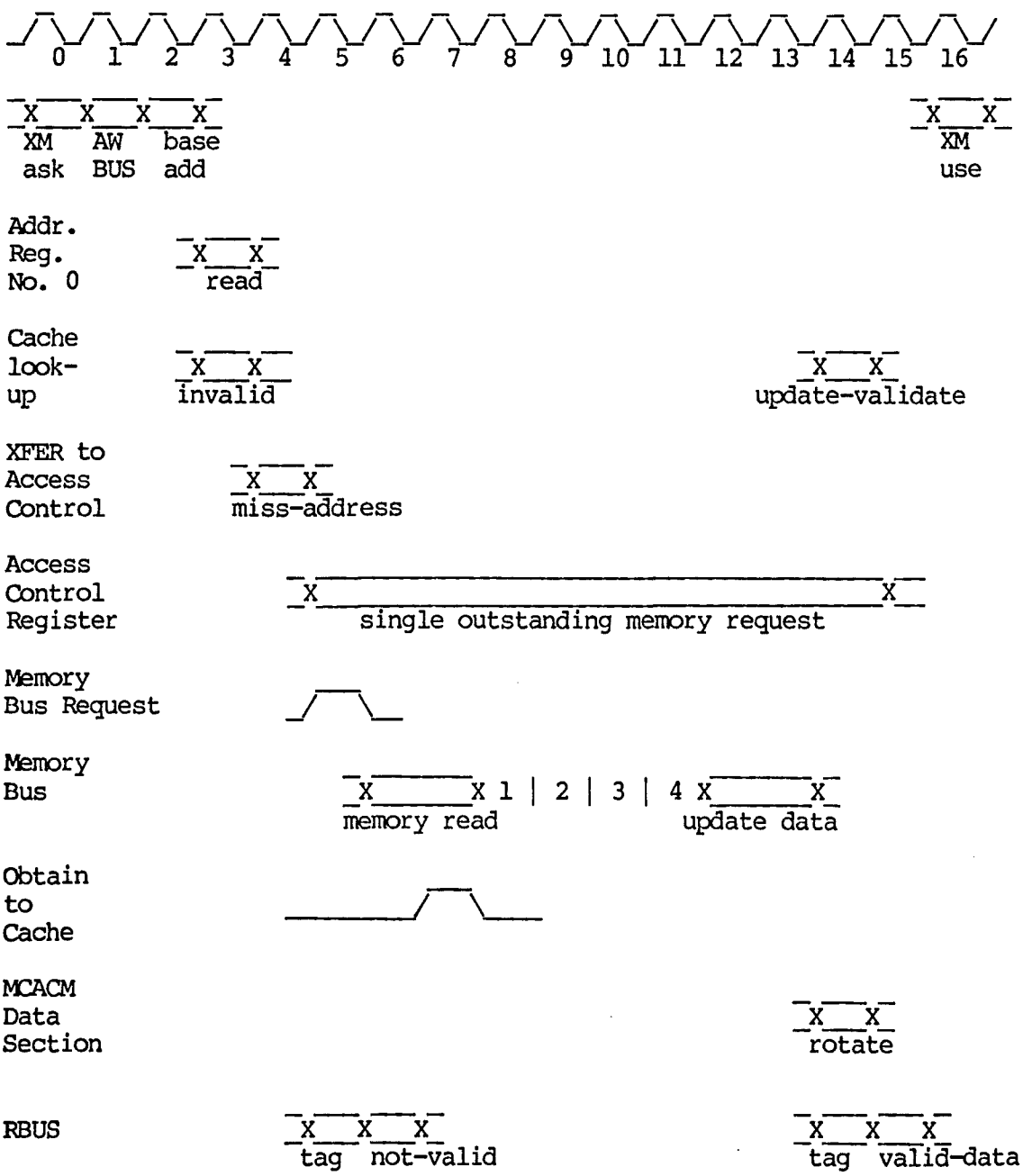


Fig. 8-5 Timing of Cache Update (read miss)

8.6 CACHE WRITEBACK TIMING

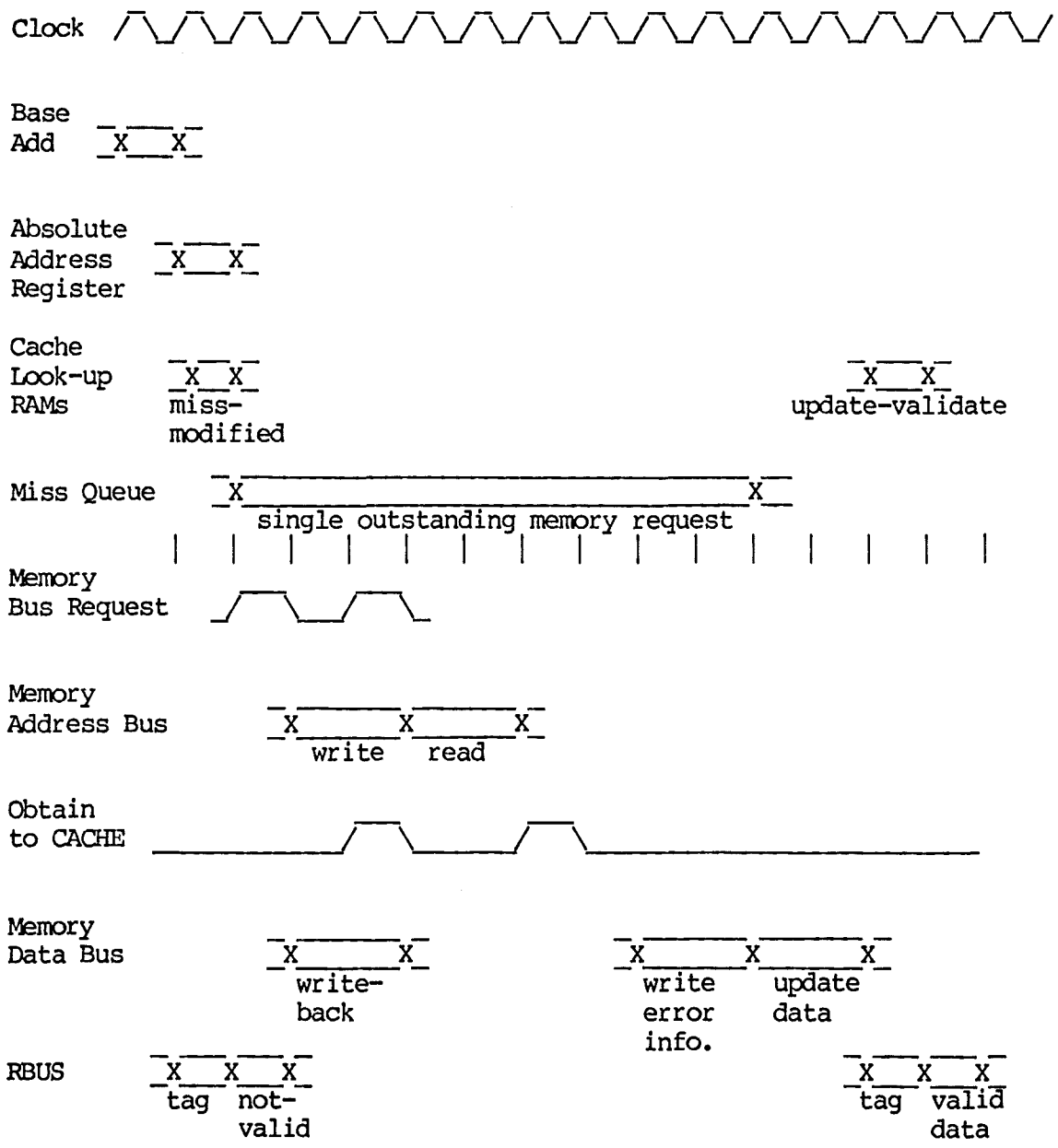


Fig. 8-6 Timing of Cache Write-Back (read miss)

9 ERROR DETECTION AND HANDLING

Error detection coverage in the memory control is very good since most of the logic is data paths and queues. The few sequencers are protected with embedded error encoding techniques. There are only a couple of chips which do not contain error detection capability. These are duplicated and the outputs are compared.

The error handling scheme for the memory control is to prevent data corruption in main memory and report all errors back to the processor through the tag field. Any retry mechanism must be implemented via some external source (either the processor or the maintenance subsystem). Most failures in the main memory or the memory control can be handled in this manner, but some require "freezing" the processor and reporting the failure directly to the maintenance processor.

The error handling mechanism is dependent on the type of operation in progress when the error occured. The MCACM handles read errors differently than write errors to assist this mechanism.

If an error is detected during a read operation, MCACM finishes the read operation and reports the error. This error is reported in the error vector field which is described in Section 5.1.3.

If an error is detected in a write operation, the MCACM enters a lock-up state which ignores write operations. This lock-up state is maintained until a read error report command is sent from a processor module. If the processor module can do instruction retry without external intervention, then it is free to do so after requesting the read error report which resets the memory control. If the processor module is unable to start instruction retry, then the maintenance subsystem needs to get involved.

There are some errors that cannot be handled in this manner. A failure in the memory request and access control sequencer, for example, could cause unpredictable data corruption in main memory. These errors are not trusted locally, and stop the system clocks by reporting a

broken condition to the SMC.

9.1 MCACM BROKEN STATE DEFINITIONS

MCACMERROR - ACCUMULATION OF MCACM ERRORS AND BROKEN STATES

BROKEN - MCACM broken.

UCONFIGERR - Invalid memory data card type.

1. Only one type of memory data cards can be used at the same time. Make sure they are the same first.
2. Check XMCRDTYPEP(3:0) with the system powered on, only one of them can be high.

BPCTLERR - Backplane intercache signals miscompare. Check XCACHCOPYP(1:0) and XMODIFIEDP(1:0), these signals are protected by duplicate and compare.

ARBERR - Memory bus arbitration error. This error is caused by the module driving Memory bus, not the module that won the arbitration. Check these signals:

XMEMARB$$P(7:0), XMEMID$$$P(2:0).

MCONFIGERR - The same as UCONFIGERR above.

MULTERROR1 - Multiple bit error detected by MCECC2. If the data came from cache, BROKEN-B would be on, otherwise the data came from memory data card.

RDATAPARER - This error does not hold or turn on BROKEN F/F above, it generates RBUS Error Vector '15'. Parity error on correct data bus between MCECC2 and ROTCAT.

1. Run pathtests MD/CRDPAR, MD/CRCTDATA0 and MD/CRCTDATA1.
2. Check these signals on MA and MD.
   CRCTDATAx(39:00) x = 0,1,2,3
   C$RDPARx x = 0,1,2,3
   CRDPARWx(3:0) x = 0,1,2,3
   CRCTDATAPARE

9.1 MCACM BROKEN STATE DEFINITIONS (Continued)

MULTERROR0 - Same as MULTERROR1.

BROKEN-B - Cache multiple bit error.

BROKEN-E - Cache Write-Back broken. This error is caused by memory bus error vector parity error or memory address bus multiple bit error or MDC internal malfunction on a memory write operation.

BROKEN-A - Same as BROKEN-E, except that this error would turn on for all memory bus operations.

QCTRLBRK - MAWQUE2 PGA at K4K6 internal broken, replace it.

10 SYSTEM MAINTENANCE INTERFACE

10.1 INTERNAL STATE ACCESS

The maintenance access into the memory control is through shift chains. All registered state is directly accessible and all RAM structures are accessible with "hidden-state" techniques. Registers which are on shift chains may not be bit-contiguous, most significant first, or monotonically ascending.

10.2 CLOCK CONTROL

The memory data cards are able to suspend operation, maintain refresh for an extended period of time, and restart operation, through the SMC. This allows synchronously stopping the clocks to the MCACM(s), IOMC(s), and MDC(s) at any time, shifting chains, analyzing hidden state, and restarting the clocks, without corrupting operations in progress.

When there are main memory accesses in progress, the entire V500 System may be stopped and/or single clocked synchronously without corrupting the operation in progress. Once the clocks have been stopped, it is possible to shift out and save the chains from the MDC(s), the MCACM(s), and the IOMC(s), do memory operation(s) via an IOMC, and finally, restore all the chains and proceed. Write collisions between the operator and any interrupted MCACM must be fixed by the maintenance processor before restarting clocks. Failure to fix the collisions could result in the operator's memory writes being ignored.

For more information about the details of the maintenance interface, see the applicable specifications which are listed in Section 2.

10.3 MCACM SYSTEM-FATAL ERROR SIGNAL

If the memory control and cache module detects an error which is sufficient to corrupt main memory or cause a loss of cache consistency, it causes the V500 System to be halted. The halt signal is transmitted to the SMC by

MCACMDEAD.

10.4 STOP LOGIC

Certain internal conditions are only visible during particular occurances of seemingly random events. The stop logic in the MCACM is designed to allow the debugger to stop system clocks, so that internal states may be examined, for the conditions specified in this section.

The system stop may be generated from the union or intersection of any of the stop conditions defined in this section. A useful example of this capability is: stopping on writing an 'F' to a certain address. This is accomplished by using both the STOP ON CACHE UPDATE and STOP ON WRITE DATA conditions.

10.4.1 STOP ON CACHE UPDATE

A cache update is detected by a write to the cache directory. There are 2 fields which may be enabled for stopping. These fields are:

WRITE ENABLE -
This is not really a separate stop condition since the directory is not getting updated when WRITE ENABLE is not active. This field must be true.

KEY -
The most-significant 15 bits of the absolute binary address. This is the value which is written into the cache directory RAM's.

SET -
The next most-significant 10 bits of the absolute binary address. This is the set address where the KEY will be written.

10.4.2 STOP ON WRITE DATA

This condition allows stopping when a certain digit in cache is written with a specified value. There are 4 fields which specify the digit and its value are:

WORD -
This is a 4-bit linear-encoded active-low word selection. It has 4 valid states which are:

E - Select Word 0
D - Select Word 1
B - Select Word 2
7 - Select Word 3

DIGIT -
This subfield selects which digit within a word to check.

VALUE -
This subfield specifies the value that will cause the stop.

MASK -
This subfield allows certain bits within the VALUE subfield to be ignored. The MASK will generally be 'F', all bits included; but certain conditions may only want to check a few of the bits.

10.4.3 STOP ON MEMORY REQUEST

A cache miss is detected by monitoring cache requests for memory. There are 4 different fields which may be enabled for stopping. These fields are:

OBTAIN -
: This is true when the request for main memory is granted. It is generally needed to qualify the other fields as corresponding to an actual miss. It is included as a separate condition in case stopping on OBTAIN is too late.

IO/MEM -
: The request is for a hardware register in the IOMC when IO/MEM is true, and for main memory when IO/MEM is false.

COMMAND -
: The command portion of the request. This is separated since it may be desireable to only stop on certain kinds of accesses to a memory address.

ADDRESS -
: The memory request address. This field is broken down into 2 subfields:

KEY -
: The most-significant 15 bits of the absolute binary address.

SET -
: The next-most-significant 10 bits of the absolute binary address.

It should be noted that the least-significant 4 bits of the BCD address are for digit selection; they are not included into the absolute binary address. Further, the least-significant 2 bits of the absolute binary address are used to select the word within a 4-word block.

10.4.4 STOP ON SPY REQUEST

The outputs of the UNMAPPER, which receives spy requests from the main memory bus, are monitored by the stop logic. There are 4 different fields of the spy request that may be enabled for stopping. These fields are:

SPYVALID -
This is not really a separate stop condition since the other fields are meaningless when the spy is not valid. Therefore, this field must be true.

COMMAND -
The command portion of the spy request. This is separated since it may be desireable to only stop on certain kinds of spy requests.

SOURCE -
The source of the spy request. This field may be separately enabled as the following 3 subfields.

PROCESSOR/IO -
The request is from another MCACM when this subfield is true. The request is from an IOMC when this subfield is false.

CABINET -
This subfield allows the cabinet from which the spy request originated to be specified.

SLOT -
This subfield allows the slot withing a cabinet from which the spy request originated to be specified.

ADDRESS -
The spy request address. This field is composed of 2 subfields:

KEY -
The most-significant 15 bits of the absolute binary address.

SET -
The next-most-significant 10 bits of the absolute binary address.

10.4.5 STOP ON RBUS ERROR

The 5-bit RBUS error vector is returned with data to the processor. This stop condition has 2 fields:

FATAL -
The most significant bit is 'fatal', it indicates whether the returned data is good, or not.

VECTOR -
This field allows stopping on some specific error. There are 16 fatal errors and 6 non-fatal errors.

10.4.6 STOP ON WRITE ABORT

A write-abort state occurs whenever a 'fatal' error is detected during a write operation. This is an unusual condition, it should not happen during normal operation of debugged code on a running system.

10.4.7 STOP ON FLUSH

The XM and FETCH modules try to pipeline instructions by pre-reading data from memory. When the predicted direction of a branch is wrong, there can be many bad requests queued in MCACM. Because this is a performance critical issue, it is included as a stop condition in MCACM.

10.4.8 MAINTENANCE MONITOR OF WRITE-ABORT

The current state of write-abort is loaded into the maintenance chain every clock. The write-abort state can be monitored therefore, without affecting the operation of the system.

11 FRONTPLANE AND BACKPLANE DEFINITION

The memory control and cache module physically resides on 2 boards. These boards are partitioned into address and data boards. These boards are connected by frontplane buses. The definition of the address board backplane, the data board backplane, and frontplane are contained in this section.

11.1 ADDRESS BOARD BACKPLANE DEFINITION

11.1.1 ADDRESS AND WRITE BUS (from processor)

SIGNAL NAME	DESCRIPTION	WIRES
XADRBUS$-##P	address field	35:0
XADRBUS$-##P	base indicant field	39:36
XADRBUSPRTYP	address parity	1
XWBUSLEN$-#P	length field	3:0
XWBUSCMD$-#P	command field	4:0
XWBUSPARITYP	control parity	1
XWBUSTAG$-#P	tag field	6:0
XWBUSTAGPARP	tag parity	1
EWBUSREQXMDP	XM request	1
EWBUSREQXMXP	COP request	1
FWBUSREQIF$P	IF request	1
FWBUSREQOF$P	OF request	1
MQUEUEFULL$P	queue full	1
XWBSRQID$-#P	bus requestor	2:0
XWRTBUS$-##P	A&W bus write data	39:0
XWRTBUSPRTYP	write data parity	1

11.1.2 RBUS (to processor)

MRBUSTAG$-#P	return bus tag field	6:0
MRBUSTAGPARP	return bus tag parity	1
MRBUSVALID$P	return bus valid	1
MRBUSERR$-#P	error vector	4:0
MRBUSTATPARP	return bus status parity	1

11.1.3 MEMORY ADDRESS BUS

XRAMADR$-##P	address field	21:0
XMEMSEL$-##P	module select field	5
XMEMSRC$-##P	requestor ID field	4
XMEMCMD$-##P	command field	2
XMEMADECC-##P	address & command ECC	8
XMEMERR$$-#P	memory data card error vector	3:0
XMEMSRC$$-#P	memory data card source field	4:0

11.1.4 MEMORY BUS ARBITRATION SIGNALS

XMEMARB$$-#P	bus request lines	7:0
MEMBUSY$-##P	memory data card busy lines	15:0
IOBSY(1:0)	IOMC busy input	2
XINHIBIT$-#P	prevent MDC's from sourcing	1:0
XCOPY$$$$-#P	copies are present of read data	1:0
XMODIFIED-#P	data is modified	1:0
IMEMBUSIlP	bus phase level	1

11.1.5 MAINTENANCE INTERFACE BUS

The backplane maintenance interface bus is:

CARD			SMC
<-------	(1)	SHIFT (4)	-------<
<-------	(1)	CLEAR (4)	-------<
<-------	(1)	DANGER (4)	-------<
<-------	(1)	CLOCK_ENABLE (32)	-------<
<-------	(1)	SHIFT_IN (4)	-------<
<-------	(1)	MODULE_ENABLE (10)	-------<
<-------	(3-5)	CARD_ENABLE (15)	-------<
<-------	(2)	CLOCK (64)	-------<
>-------	(1)	SHIFT_OUT (4)	------->
>-------	(1)	MOD_BROKEN (9)	------->
>-------	(1)	MOD_NOT_BROKEN (9)	------->
>-------	(1)	SOFT SNAP (2)	------->
>-------	(1)	STOP_ANDED (3)	------->
>-------	(1)	STOP_ORED (3)	------->

The backplane maintenance interface bus is hidden from the internal workings of the MCACM by the clock distribution array. This array buffers and decodes the backplane signals to present a clean, well ordered interface to the card.

11.2 DATA BOARD BACKPLANE DEFINITION

11.2.1 READ BUS

MRBUS$$$-##P	read data	39:0
MRBUSPARITYP	read data parity	1

11.2.2 BIDIRECTIONAL MEMORY DATA BUS

XRAMDT0$-##P	word 0 of memory data bus	39:0
XRAMEC0$-##P	ECC of word 0 of memory data bus	7:0
XRAMDT1$-##P	word 1 of memory data bus	39:0
XRAMEC1$-##P	ECC of word 1 of memory data bus	7:0
XRAMDT2$-##P	word 2 of memory data bus	39:0
XRAMEC2$-##P	ECC of word 2 of memory data bus	7:0
XRAMDT3$-##P	word 3 of memory data bus	39:0
XRAMEC3$-##P	ECC of word 3 of memory data bus	7:0

11.2.3 MAINTENANCE INTERFACE BUS

The details of the maintenance interface bus to the data board are identical to those of the address board which was defined in Section 11.1.5.

11.3 MEMORY CONTROL FRONTPLANE DEFINITION

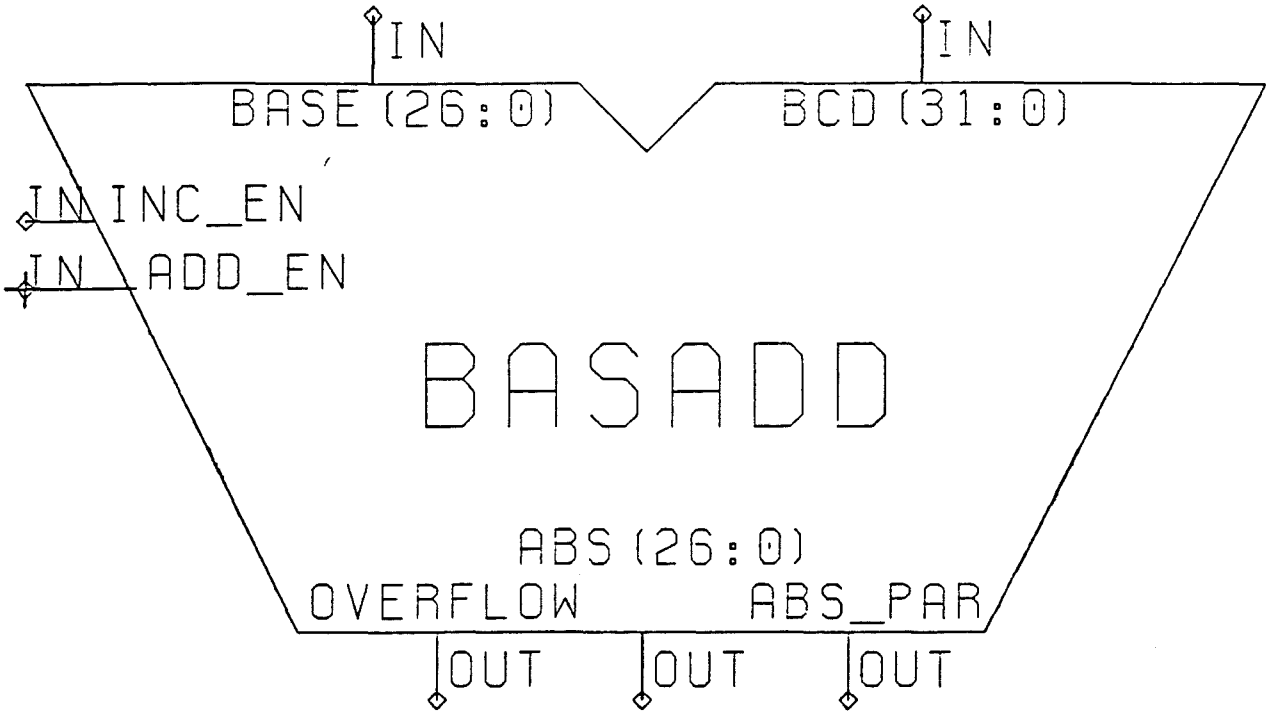
11.3.1 DATA SECTION CONTROL SIGNALS

LSB-#	decoded word select	3:0
LSD-#	least significant digit of addr.	3:0
LEN-#		3:0

11.3.2 RANDOM INTER-BOARD SIGNALS

12 ARRAY SPECIFICATIONS

12.1 BASADD



12.1.1 FUNCTION OVERVIEW

This array converts the input 8-digit BCD address into a 27-bit binary value. Then the absolute address is obtained by adding this value to the base address from base table.

12.1.2 INTERFACES

12.1.2.1 INPUTS

BCD(31:0) -
The most significant 8-digits of the processor request address. The least significant digit is only used for rotation and alignment; it is not converted to binary.

BASE(26:0) -
The output of the base table entry as indicated by the base indicant in the processor request. This is a 27-bit binary number.

ADD-ENABLE -
The base adder is enabled with this input. The base adder must be disabled when writing to the base and limit tables.

INC-ENABLE -

The carry-in to the base adder increments the absolute binary address at the output by one. This is used for operand-reads which are converted into 2 separate reads internal to MCACM. The second read is incremented via this input to point to the next block.

12.1.2.2 OUTPUTS

ABS(26:0) -
The result of the BCD-to-binary conversion of the input BCD address and addition of the binary BASE.

ABS-PAR -
The parity of the ABS(26:0) output.

OVERFLOW -
This is an error output. It indicates that the input address plus the base is greater than 1,342,177,270. This is an unreasonable value since it is greater than the amount of memory that is physically addressable by the processor.

12.1.3 DETAILED DESCRIPTION

The BASADD array converts the input BCD address into binary. This binary value is added to the binary BASE to provide the absolute binary address of the request from the processor.

12.1.3.1 BCD TO BINARY CONVERTER

The 8-digit BCD address is converted into two psuedo-binary values by a series of Wallace Tree adders. There are two sets of 27 Wallace Trees. Each Wallace Tree accumulates 1 bit of the binary output by adding the binary weight of each of the BCD inputs. These 2 outputs are combined in the base adder to yield the actual binary value of the BCD input.

The INC-ENABLE input increments the 27-bit absolute binary output. It is an input to the Wallace Tree of the least significant bit.

12.1.3.2 BASE ADDER

The base adder adds the 2 outputs of the BCD-to-binary converter and the binary BASE input. The BASE input is enabled with ADD-ENABLE. This allows a pure BCD-to-binary conversion when ADD-ENABLE is low.

The base adder consists of 2 adders. The first is a 3-input CSA (carry save adder). The CSA is very fast: it adds all 3-inputs for each bit producing both sum and carry outputs. The carry from each stage is not used by higher stages, but saved as an output.

The second adder is a fast 27-bit CLA (carry look-ahead adder). It adds the sum and carry outputs from the CSA to produce a 27-bit absolute binary address.

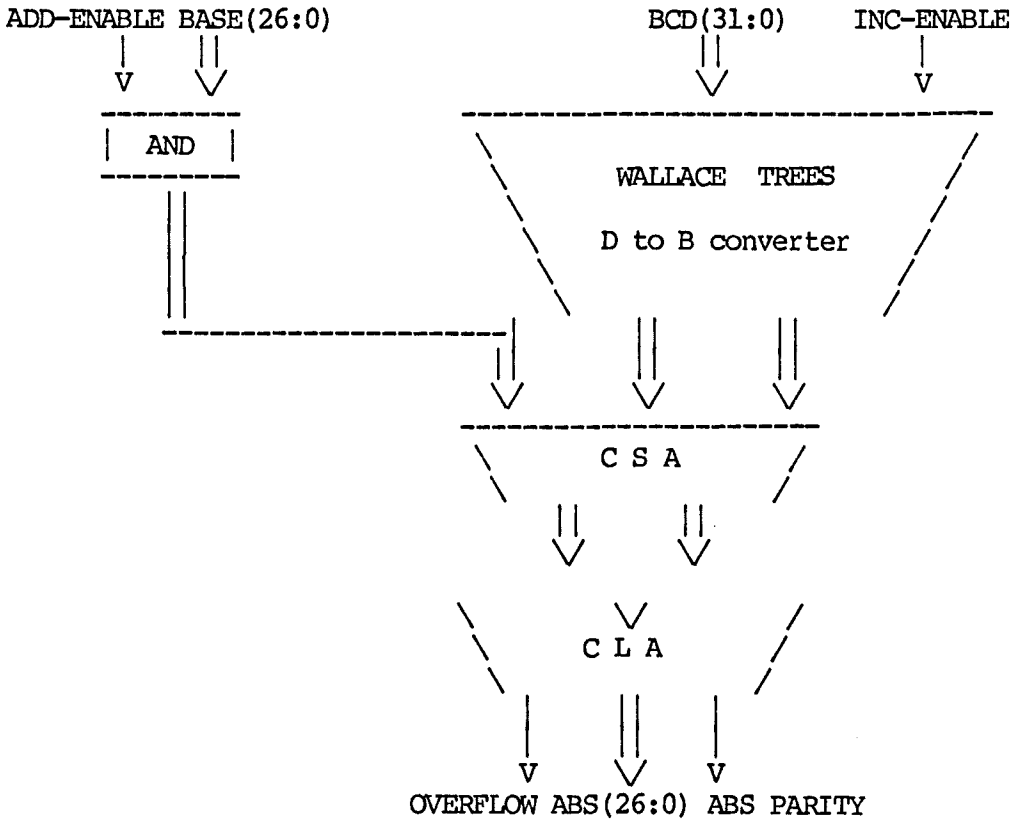


Fig. 12-1 The Structure of the Base Adder

12.1.3.3 OUTPUT PARITY GENERATION

There is a 27-bit parity tree on the output of the CLA which calculates the odd-parity that is stored with the base and limit table entries. The outputs of these tables are checked by parity checkers in the LIMCHK array.

12.1.3.4 FAULT DETECTION

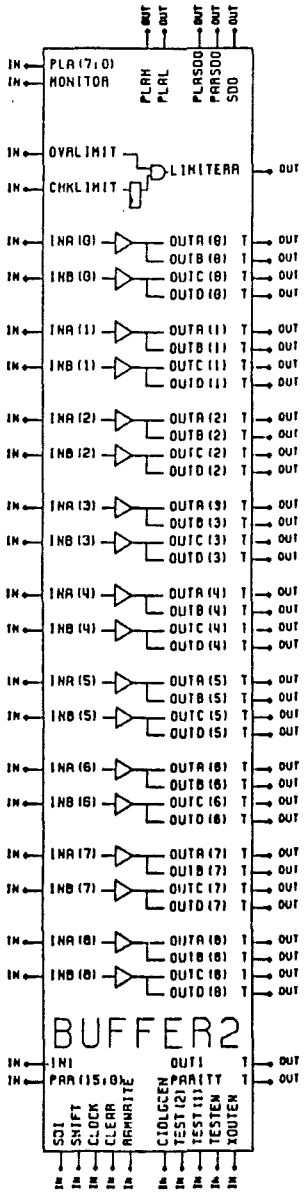
Fault detection of this array is accomplished with a duplicate-and-compare strategy. Duplicate BASADD arrays operate in parallel to generate two identical copies of the absolute binary address. These are compared inside duplicate LIMCHK (limit checker) arrays with TSC (totally self-checking) comparators.

12.1.4 SHIFT CHAIN DEFINITION

The BASADD array is purely combinatorial: there is no internal maintenance shift chain.

12.2 BUFFER2

25-OHM DRIVERS AND COMMON-I/O RAM CONTROLLER



12.2.1 FUNCTION OVERVIEW

The BUFFER2 option provides 37 twenty-five ohm drivers for buffering the interface between the MCACM address and data cards. It also provides the WE/ and CS/ logic to drive the F474 cache data RAM's. A 16 input parity tree with a twenty-five ohm driver and a gated flip-flop are also provided for random needs.

The 37 twenty-five ohm drivers are configured as: 18 1-to-2 buffers, and 1 single-input single-output twenty-five ohm buffer. This allows buffering 19 interface signals.

The WE/ and CS/ logic operate the same as the logic in the CAFLAG option which controls the reading/writing of the cache flag RAM's. The WE/ and CS/ lines are gated with clock and delayed from the the write-enable/ signal to guarantee data and address setup time to the F474's. The delays have been minimized on the trailing edges to minimize data hold times. The waveforms for this logic are included in the detailed descript

12.2.2 INTERFACES

12.2.2.1 INPUTS

MONITOR -
The MONITOR input pin controls the loading of the 8-bit PLADATA register in BUFFER2. When MONITOR is high, the PLADATA register is loaded from the PLA(7:0) inputs. When MONITOR is low, the PLADATA register is loaded from the Writeable PLA RAM.

PLA(7:0) -
These 8 inputs enter the Writeable PLA structure in the BUFFER2 array. They form the inputs to an 8-input, 4-term, 1-output Programable Logic Array. When the MONITOR input is high, these inputs are loaded into the PLADATA register.

12.2.2.1 INPUTS (Continued)

OVRLIMIT -
Input to the limit error gating logic. This input is logically anded with the registered CHKLIMIT input. The output of the AND gate is LIMITERR.

CHKLIMIT -
Input to the limit error gating logic. This input is registered before being logically anded with OVRLIMIT input. The output of the AND gate is LIMITERR.

INA(8:0) -
Inputs to the 25 ohm drivers and the common-I/O RAM controller logic. The details of this logic are discussed in Section 12.2.3.1.

INB(8:0) -
Inputs to the 25 ohm drivers and the common-I/O RAM controller logic. The details of this logic are discussed in Section 12.2.3.1.

IN1 -
Input to an uncommited 25 ohm driver. The output of this driver is OUT1.

PAR(15:0) -
Inputs to an uncommited parity tree. The output of the parity tree is PARITY.

SDI -
The input to the maintenance shift chain

CLOCK -
The system clock input to the array.

SHIFT -
The shift control input to the maintenance shift chain.

CLEAR -
The asynchronous clear input to the maintenance shift chain.

12.2.2.1 INPUTS (Continued)

RAMWRITE -
The write-enable signal to the array of latches used in the writeable PLA structure. This input should be connected to the RAM WRITE PULSE output of the CLKMNT2 array.

CIOLGCEN -
The BUFFER2 array functions in 1 of 2 primary modes: 25-ohm buffer or common-I/O RAM controller. This input selects the common-I/O RAM controller mode when high.

TESTEN -
This input is only used during wafer and chip testing. It bypasses an internal delay line, using the TEST(2:1) inputs instead.

TEST(2:1) -
These inputs are only used during wafer and chip testing. They are inserted in place of the output of an internal delay line. This allows testing the delay line by checking its outputs separately from the rest of the common-I/O RAM controller logic.

12.2.2.2 OUTPUTS

PLAH -
The active high output from the writeable PLA structure. This is the output from the internal 8-input, 4-term, 1-output PLA.

PLAL -
The active low output from the writeable PLA structure. This is the output from the internal 8-input, 4-term, 1-output PLA.

PLASDO -
The last output of the maintenance shift chain. This output must be used if the BUFFER2 array uses the internal PLA or the monitor capability. It may be ignored if neither of these features are used. This allows maximum flexibility by not requiring changes to the shift chain definition when the BUFFER2 array is installed.

PARSDO -
This is the output from a flip/flop added to the output of the internal parity tree. This will allow testing the RBUS data parity with MCACM-only pathtests. This output must be used as the output of the maintenance shift chain in order to use this feature. It may be ignored otherwise.

SDO -
This is the output of the maintenance shift chain which was contained in BUFFER. It is provided so that BUFFER2 can be plugged in place of BUFFER with no change to the length of the shift chain.

LIMITERR -
This is the output of the limit check gating logic. It is the logical AND of the CHKLIMIT (registered and delayed by 1 clock) and OVRLIMIT.

12.2.2.2 OUTPUTS (Continued)

OUTA(8:0) & OUTB(8:0) -
When CIOLGCEN is high, these outputs are the chip-select drivers for common-I/O RAMs. When CIOLGCEN is low, these outputs are 25-ohm buffers from the INA(8:0) inputs.

OUTC(8:0) & OUTD(8:0) -
When CIOLGCEN is high, these outputs are the write-enable drivers for common-I/O RAMs. When CIOLGCEN is low, these outputs are 25-ohm buffers from the INB(8:0) inputs.

OUT1 -
This is a 25-ohm buffered output of IN1.

PARITY -
Even parity over the PAR(15:0) inputs. This is a standard 50-ohm output.

12.2.3 DETAILED DESCRIPTION

12.2.3.1 WRITE ENABLE AND CHIP SELECT LOGIC

The WE/ and CS/ logic generate the RAM control signals in response to the WRITE-ENABLE/ and XOUTEN inputs. The WRITE-ENABLE/ is the active low signal to enable writing into the cache data RAM's. When WRITE-ENABLE/ is high the RAM's are in read mode. Unless, XOUTEN is high, which disables the RAM's for a RAM bypass operation.

The WE/ is brought low as early as possible during a write cycle so that the outputs of the F474 RAM are disabled and the data bus is allowed to stabilize. Then, the CS/ is pulsed low with the system clock. The CS/ pulse does the actual write.

12.2.3.1 WRITE ENABLE AND CHIP SELECT LOGIC (Continued)

The following timing diagram shows the relationships between these signals.

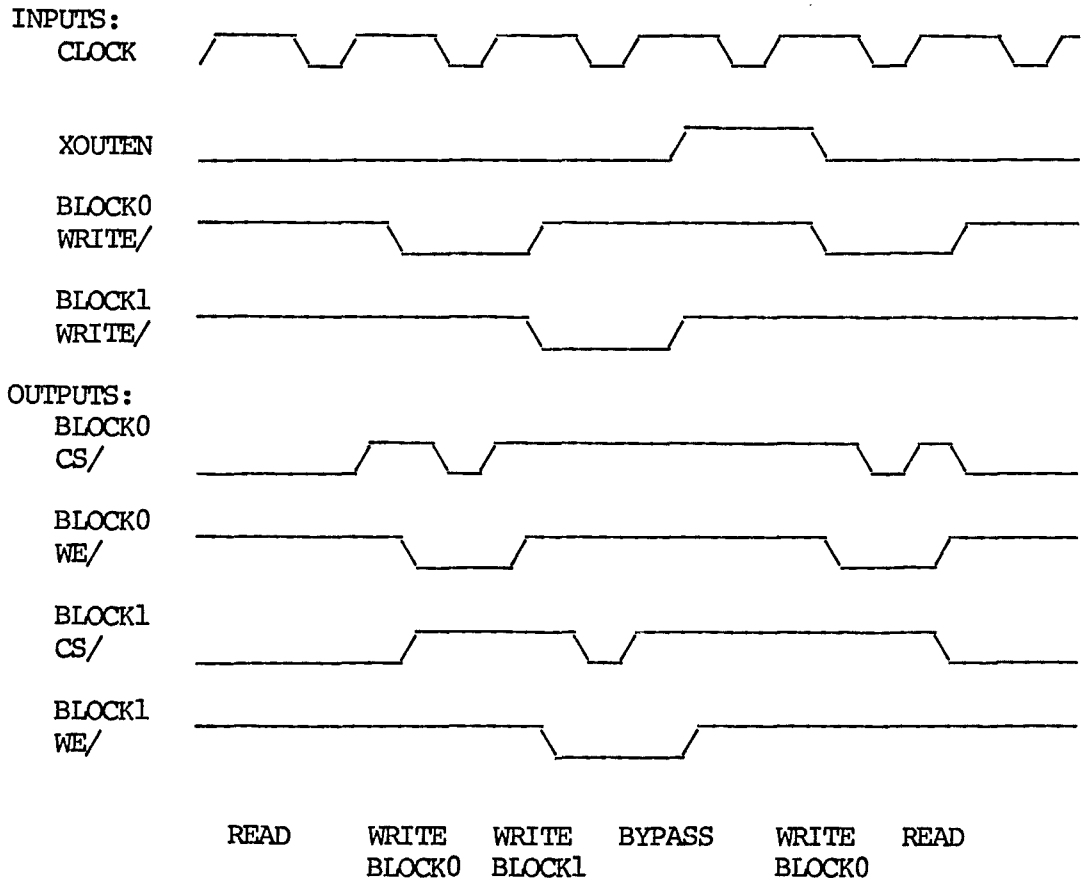


Fig. 12-2 Write-Enable and Chip-Select Timing Diagram

12.2.3.2 LIMIT ERROR DETECTION QUALIFICATION

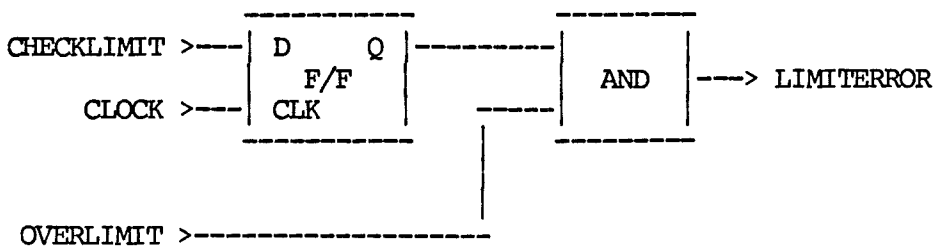


Fig. 12-3 Limit Error Detection Circuit

12.2.3.3 WRITEABLE ARRAY LOGIC

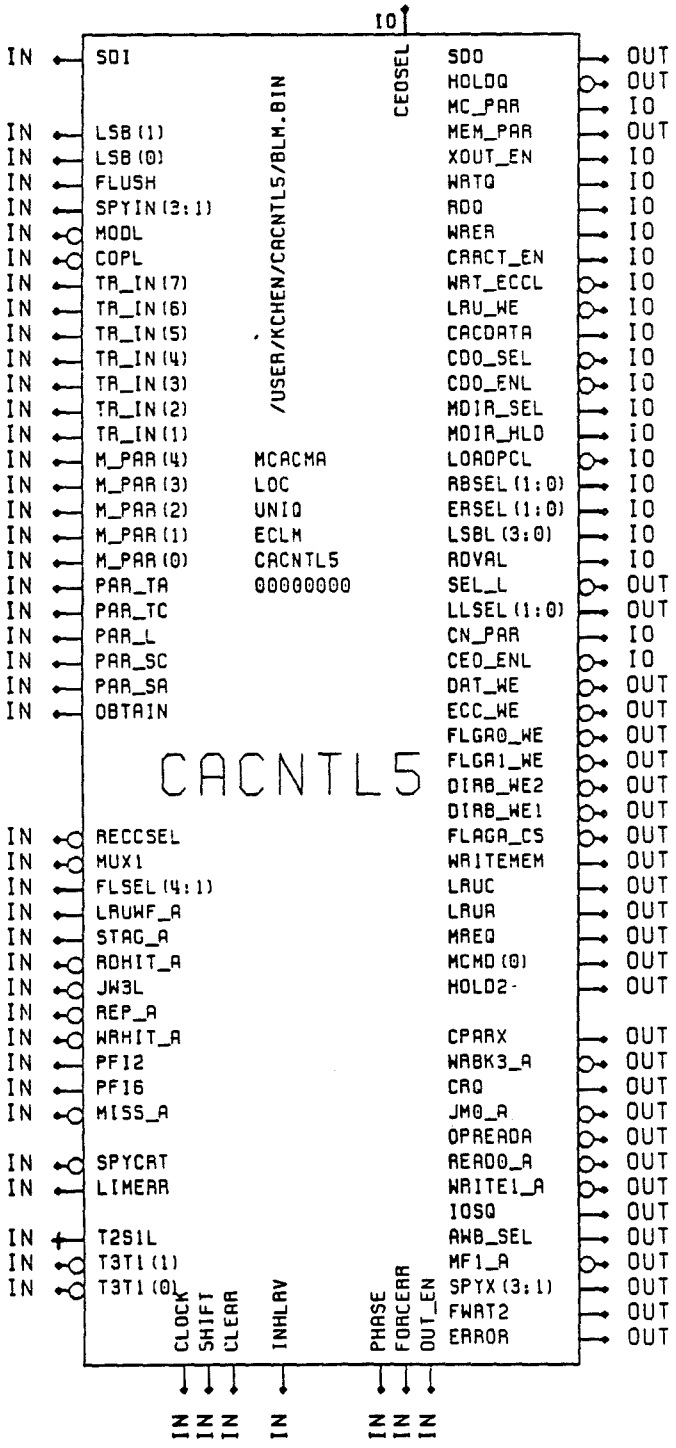
The BUFFER2 array contains an 8-input, 4-term, 1-output, hidden-state-writeable, logic array.

12.2.4 SHIFT CHAIN DEFINITION

Three separate shift-data-outputs have been provided so that retrofitting this option into current designs would have minimal impact on existing shift chains. The shift chain in the BUFFER2 array is described below.

```
                PLAADDR(2:0)      ----> PLASDO
                PLADATA(7:0)
                WRITE
                PARITY            ----> PARSDO
SDI ----->      CHECKIT           ----> SDO
```

12.3 CACNTL5


IO
IN SDI
IN LSB(1)
IN LSB(0)
IN FLUSH
IN SPYIN(3:1)
IN MODL
IN COPL
IN TR_IN(7)
IN TR_IN(6)
IN TR_IN(5)
IN TR_IN(4)
IN TR_IN(3)
IN TR_IN(2)
IN TR_IN(1)
IN M_PAR(4)
IN M_PAR(3)
IN M_PAR(2)
IN M_PAR(1)
IN M_PAR(0)
IN PAR_TA
IN PAR_TC
IN PAR_L
IN PAR_SC
IN PAR_SA
IN OBTAIN
IN RECCSEL
IN MUX1
IN FLSEL(4:1)
IN LRUWF_A
IN STAG_A
IN RDHIT_A
IN JW3L
IN REP_A
IN WRHIT_A
IN PFI2
IN PFI6
IN MISS_A
IN SPYCRT
IN LIMERR
IN T2S1L
IN T3T1(1)
IN T3T1(0)
/USER/KCHEN/CACNTL5/BLM.BIN
MCACMA
LOC
UNIQ
ECLM
CACNTL5
00000000
CEOSEL
CACNTL5
CLOCK
SHIFT
CLEAR
INHLAV
PHASE
FORCERR
OUT_EN
SDO OUT
HOLDQ OUT
MC_PAR IO
MEM_PAR OUT
XOUT_EN IO
WRTQ IO
RDQ IO
WRER IO
CRRCT_EN IO
WRT_ECCL IO
LRU_WE IO
CACDATA IO
CDO_SEL IO
CDO_ENL IO
MDIR_SEL IO
MDIR_HLD IO
LOADPCL IO
RBSEL(1:0) IO
ERSEL(1:0) IO
LSBL(3:0) IO
RDVAL IO
SEL_L OUT
LLSEL(1:0) OUT
CN_PAR IO
CEO_ENL IO
DAT_WE OUT
ECC_WE OUT
FLGR0_WE OUT
FLGR1_WE OUT
DIRB_WE2 OUT
DIRB_WE1 OUT
FLAGR_CS OUT
WRITEMEM OUT
LRUC OUT
LRUR OUT
MREQ OUT
MCMD(0) OUT
HOLD2- OUT
CPARX OUT
WRBK3_A OUT
CRQ OUT
JM0_A OUT
OPREADR OUT
READ0_A OUT
WRITE1_A OUT
IOSQ OUT
AWB_SEL OUT
MF1_A OUT
SPYX(3:1) OUT
FWRT2 OUT
ERROR OUT
IN IN IN IN IN IN IN

12.3.1 FUNCTION OVERVIEW

The CACNTL5 array contains the major logic to control CACHE miss. The CACNTL5 contains a 4-bit state machine which provides 16 different states; only 15 are used. The cache command pipeline and its associated command decodeing logic is also in this array. It also contains the logic for cache ram write enable and chip select signals, RBUS select, error bus select and major control signals to control MCACM-D card flow.

12.3.2 INTERFACES

12.3.2.1 INPUTS

SDI	- Maintenance shift chain input.
CLOCK	- System CLOCK.
SHIFT	- Maintenance control signal for the shift chain.
CLEAR	- Maintenance control signal for the shift chain.
PHASE	- Memory bus phase signal.
FORCERR	- Forces a hardware error to be reported.
OUT-EN	- Master/Slave Jumper. When high, this array is the master.
INHLRV	- Cache RAM inhibit bit source.
SPYIN(3:1)	- Spy command and valid bit.
TR-IN(7:1)	- Command pipeline input, encoded into 5-bits.
M-PAR(4:0)	- Address pipeline parity from CHADDR3
PAR-TA	- Address parity.
PAR-TC	- Command parity.
PAR-L	- Least significant digit & length parity from MAWQUE2.
PAR-SA	- Spy address parity.
PAR-SC	- Spy command parity.
RECCSEL	- Read ECC mux select signal.
MUX1	- Pipeline hold control signal.
LSB(1:0)	- Least significant 2-bits of the absolute binary address. This is used for word selection by the ROTCAT arrays in the data section.
FLUSH	- Read command flush signal.
MODL	- Return memory data is modified.
COPL	- Return memory data is a copy.

12.3.2.1 INPUTS (Continued)

OBTAIN	- The current outstanding request for main memory has been granted. If there is another request waiting, it can be requested now.
PFI2	- Cache allows to be updated.
PFI6	- Cache allowed to use the cache RAM data bus.
REP-A	- Cache contents need to be replaced into memory.
MISS-A	- Cache miss.
JW3L	- Operand read sends operation complete signal.
RDHIT-A	- Read operation hit.
WRHIT-A	- Write operation hit.
STAG-A	- When high, drive the RBUS from the T3 tag register. When low, use the T2 tag register.
FLSEL(4:1)	- Write enable control signals.
LRUWF-A	- Signal controls which block of cache to write.
IOMC	- Spy source is IOMC.
T2S1L	- Spy-1 pipeline equals processor T2 pipeline
LIMERR	- Limit error (active low)
T3T1(1:0)	- Processor T3 pipeline equals T1 pipeline.
UPGRATL	- Reserved for future use.

12.3.2.2 OUTPUTS

SDO	- The output of the maintenance shift chain.
MEM-PAR	- Memory request parity.
SEL-L	- Tag flip-flop (active low).
LLSEL(1:0)	- Length & LSD field mux select.
DAT-WE	- Data write enable.
ECC-WE	- Ecc write enable.
LRU-WE	- LRU RAM write enable.
DIRB-WE2	- KEY RAM directory write enable.
DIRB-WE1	- KEY RAM directory write enable
FLGA1-WE	- FLAG RAM write enable.
FLGA0-WE	- FLAG RAM write enable.
FLAGA-CS	- FLAG RAM chip select.
CPARX	- Parity bit.
FWRT2	- Update cache memory.
ERROR	- A fatal error has been detected.
LRUA	- Maintenance signal to LRU RAM source.
LRUC	- Maintenance signal to LRU RAM source.
MREQ	- Request memory.
MCMD(0)	- Memory command. The master CACNTL4 drives bit-1 and the slave drives bit-0.
HOLD2	- T2 pipeline hold signal.
WRBK3-A	- T3 pipeline is write-back command.
OPREADA	- pipeline is an operand read command.
WRITE1-A	- T1 pipeline is a write command.
READ0-A	- Request directory read.
SPYX(3:0)	- Spy command pipeline output.
CRQ	- Request the cache bus.
JM0-A	- M-state machine idle.
AWB-SEL	- Tag flip-flop (active high).
IOSQ	- IO operation hold processor pipeline.
MF1-A	- Memory data flag.
CRRCT-EN	- Enable correcting cache data.
HOLDQ	- Processor pipeline hold signal.
WRTQ	- Status signal to MAWQUE2: cache contains a write command.
RDQ	- Status signal to MAWQUE2: cache contains a read command.
WRER	- A write error has been detected.
WRT-ECCL	- Setup the ROTCAT array to write ECC instead of data.

12.3.2.2 OUTPUTS (Continued)

LOADPCL	- Load the PC register in the ROTCAT array from the data bus.
CACDATA	- Source data from cache RAM.
MC-PAR	- Parity of the command field for memory request.
XOUT-EN	- Cache RAM output enable.
CDO-ENL	- Cache bus data source output enable.
CDO-SEL	- Cache bus data source select.
CEO-ENL	- Cache bus ECC source output enable.
CEOSEL	- Cache bus ECC source select.
MDIR-SEL	- Memory register source select.
MDIR-HLD	- Memory register load.
RBSEL(1:0)	- RBUS source select.
ERSEL(1:0)	- Error bus source select.
LSBL(3:0)	- Word select.
RDVAL	- RBUS valid.
CN-PAR	- Parity bit to CHADDR3.

12.3.3 DETAILED DESCRIPTION

12.3.3.1 M-STATE MACHINE

The definition of the M-state machine states are as follows:

State 0 - idle state.
State 1 - replace cache contents into main memory.
State 2 - request for memory bus.
State 3
to
State 8 - waiting memory data return
State 9 - load memory data into memory data/ecc register.
State 10 - load into MCECC array ecc register and will perform ECC checking.
State 11
to
State 13 - waiting for update cache memory.
State 15 - maint. state.

12.3.3.2 PROCESSOR COMMAND PIPELINE

The definition of the command pipeline is as follows:

00 - operand read.
01 - read data.
02 - io write.
03 - io read.
04 - test error detection.
05 - read base
06 - invalid command
07 - read limit
08-0B - invalid command
0C-0F - error command
10 - write pc
11 - read pc
12 - read error address
13 - read error report
14-15 - reserved
16 - read with lock
17 - reserved
18 - write back
19 - write base
1A - invalid command
1B - write limit
1C - reserved
1D - write data
1E - invalid command
1F - operand read uncorrected

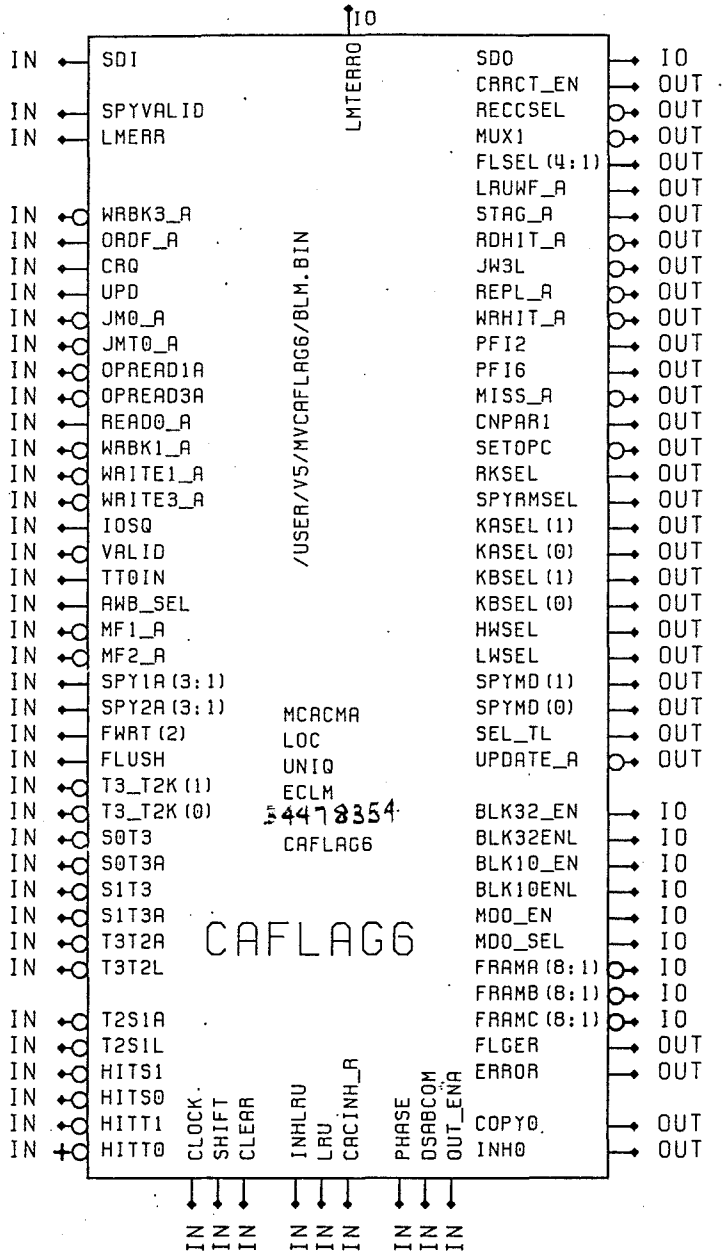
12.3.3.3 FAULT DETECTION

The fault detection in this array uses 2 different techniques. These 2 techniques are: parity prediction/checking and duplicate-and-compare checking. The duplicate and compare protection is provided by using a master/slave input jumper. One of the CACNTL5 arrays is the master: it drives the critical control signals. The other CACNTL5 array: is the slave it checks these critical control signals. Any differences are reported as a hardware broken condition.

12.3.4 SHIFT CHAIN DEFINITION

```
MRDVAL          1;      % R BUS VALID
MLSBENL         1;
MLDMREG         1;      % LOAD MDI REG
MS2COMDVAL      1;      % SPY2 COMMAND VALID BIT
MS1COMDVAL      1;      % SPY1 COMMAND VALID BIT
MS2CMD-1        1;      % SPY2 2 BITS COMMAND
MS1CMD-1        1;      % SPY1 2 BITS COMMAND
MS2CMD-0        1;
MS1CMD-0        1;
MCNTLERR        1;      % CACNTL ARRAY IN ERROR
MCDOEN          1;
MCEOEN          1;      % WRITE ECC CONTROL PATH
MCDOSL          1;
MPF6            1;
MPF2            1;
MTAGF           1;
MORDERF         1;      % PIPELINE ORDER FLIP FLOP
MSPAREF         1;
MJM             4;      % JM 4-BIT STATE MACHINE
MF1             1;
MF3             1;
MT2CMD-4        1;      % TRAN2 5 BIT COMMAND
                          (FROM PROCESSOR)
MT1CMD-4        1;
MT3CMD-4        1;
MT2CMD-3        1;
MT1CMD-3        1;
MT3CMD-3        1;
MT2CMD-2        1;
MT1CMD-2        1;
MT3CMD-2        1;
MT2CMD-1        1;
MT1CMD-1        1;
MT3CMD-1        1;
MT2CMD-0        1;
MT1CMD-0        1;
MT3CMD-0        1;
MT2COMDPAR      1;      % COMMAND PARITY BIT
MT1COMDPAR      1;
MT3COMDPAR      1;
```

12.4 CAFLAG6


IO
LMTERRO
IN SDI
IN SPYVALID
IN LMERR
IN WRBK3_A
IN ORDF_A
IN CRQ
IN UPD
IN JM0_A
IN JMT0_A
IN OPREAD1A
IN OPREAD3A
IN READ0_A
IN WRBK1_A
IN WRITE1_A
IN WRITE3_A
IN IOSQ
IN VALID
IN TT0IN
IN AWB_SEL
IN MF1_A
IN MF2_A
IN SPY1A(3:1)
IN SPY2A(3:1)
IN FWRT(2)
IN FLUSH
IN T3_T2K(1)
IN T3_T2K(0)
IN S0T3
IN S0T3A
IN S1T3
IN S1T3A
IN T3T2A
IN T3T2L
IN T2S1A
IN T2S1L
IN HITS1
IN HITS0
IN HITT1
IN HITT0
/USER/V5/MVCAFLAG6/BLM.BIN
MCACMA
LOC
UNIQ
ECLM
5447835
CAFLAG6
CAFLAG6
SDO IO
CRRCT_EN OUT
RECCSEL OUT
MUX1 OUT
FLSEL(4:1) OUT
LRUWF_A OUT
STAG_A OUT
RDHIT_A OUT
JW3L OUT
REPL_A OUT
WRHIT_A OUT
PFI2 OUT
PFI6 OUT
MISS_A OUT
CNPAR1 OUT
SETOPC OUT
RKSEL OUT
SPYRMSEL OUT
KASEL(1) OUT
KASEL(0) OUT
KBSEL(1) OUT
KBSEL(0) OUT
HWSEL OUT
LWSEL OUT
SPYMD(1) OUT
SPYMD(0) OUT
SEL_TL OUT
UPDATE_A OUT
BLK32_EN IO
BLK32ENL IO
BLK10_EN IO
BLK10ENL IO
MDO_EN IO
MDO_SEL IO
FRAMA(8:1) IO
FRAMB(8:1) IO
FRAMC(8:1) IO
FLGER OUT
ERROR OUT
COPY0 OUT
INH0 OUT
CLOCK
SHIFT
CLEAR
INHLRU
LRU
CACINH_A
PHASE
DSABCOM
OUT_ENA
IN IN IN IN IN IN IN IN IN

12.4.1 FUNCTION OVERVIEW

The CAFLAG6 array contains the major logic to control CACHE write and OPERAND READ. It maintains the prioritization between the processor and spy for the cache directory. The CAFLAG5 array determines if a miss requires replacing the current block before requesting the needed block.

12.4.2 INTERFACES

12.4.2.1 INPUTS

SDI	- serial data input.
CLOCK	- SYSTEM CLOCK.
SHIFT	- MAINTENANCE SHIFT
CLEAR	- SYSTEM CLEAR.
PHASE	- MEMORY BUS CLOCK.
FORCERR	- FORCE ERROR CONDITION
OUT-ENA	- JUMPER - HIGH IS MASTER, LOW IS SLAVE.
CACINH-R	- CACHE RAM INHIBIT BIT OUTPUT
INHLRU	- INHIBIT LRU FUNCTION
LRU	- CACHE RAM LEAST RECENTLY USED BIT OUTPUT
SPYVALID	- SPY COMMAND VALID BIT DIRECTLY FROM MCNTL. USED FOR PRIORITY DETERMINATION.
SPY1A(3:1)	- SPY COMMAND AND VALID BIT FIRST STAGE PIPELINE
SPY2A(3:1)	- SPY COMMAND AND VALID BIT SECOND STAGE PIPELINE
READ0-A	- COMMAND REQUEST FOR DIR READ. USED FOR PIORITY DETERMINATION.
CRQ	- CACHE BUS REQUEST, USED FOR PRIORITY DETERMINATION.
UPD	- CACHE UPDATE REQUEST, USED FOR PRIORITY DETERMINATION.
JM0-A	- M STATE MACHINE IDLE
JMT0-A	- M STATE MACHINE GOING TO BE IDLE
ORDF-A	- COMMAND IN T PIPELINE IS OUT OF ORDER.
OPREAD1A	- PIPELINE T1 HAS OPERAND READ COMMAND
OPREAD3A	- PIPELINE T3 HAS OPERAND READ COMMAND
WRBK1-A	- T1 PIPELINE HAS WRITE BACK COMMAND
WRBK3-A	- T3 PIPELINE HAS WRITE BACK COMMAND
WRITE1-A	- T1 PIPELINE HAS WRITE COMMAND
WRITE3-A	- T3 PIPELINE HAS WRITE COMMAND
IOSQ	- IO OPERATION HOLD PROCESSOR PIPELINE

12.4.2.1 INPUTS (Continued)

VALID - T3 HAS VALID COMMAND.
TT0IN - TAG FLIP FLOP IS GOING TO BE 0
AWB-SEL - TAG FLIP FLOP OUTPUT
MF1-A - MEMORY REGISTER FLAG
MF2-A - MEMORY DATA FLAG
FWRT(2) - UPDATE CACHE MEMORY
FLUSH - READ COMMAND FLUSH SIGNAL
LMERR - LIMIT ERROR
T3-T2K(1:0)- PROCESSOR T3 PIPELINE EQUALS REPLACED ADDR (active low).
T3T2A T3T2L- PROCESSOR T3 PIPELINE EQUALS T2 PIPELINE (active low).
S0T3 S0T3A - PROCESSOR T3 PIPELINE EQUALS S0 (active low).
S1T3 S1T3A - PROCESSOR T3 PIPELINE EQUALS S1 PIPELINE (active low).
T2S1A T2S1L- PROCESSOR S1 PIPELINE EQUALS T2 PIPELINE (active low).

12.4.2.2 OUTPUTS

SDO	- SHIFT DATA OUT
CRRCT-EN	- CORRECT ENABLE SIGNAL
RECCSEL	- SELECT CACHE BLOCK 1
MUX1	- PIPELINE LOOP CONTROL SIGNAL
FLSEL-4	- UPDATE FLAG DIRA
FLSEL-3	- UPDATE BLOCK1
FLSEL-2	- UPDATE FLAG DIRB
FLSEL-1	- UPDATE CACHE
LMTERRO	- LIMIT ERROR ACTIVE LOW SIGNAL
LRUWF-A	- MISS UPDATE CACHE BLOCK1
STAG-A	- SET TAG FLIP FLOP
RDHIT-A	- READ HIT
WRHIT-A	- WRITE HIT
REPL-A	- REPLACE
MISS-A	- MISS
JW3L	- OPREAD GOING TO BE FINISHED. SET RBUS VALID
PFI2	- READY TO UPDATE CACHE
PFI6	- READY TO USE CACHE BUS
CNPAR1	- CONTROL SIGNALS TO CHADDR ARRAY PARITY BIT
SETOPC	- LIMIT ERROR SELECT PC
RKSEL	- READ KEY SELECT
SPYRMSEL	- SPY OPERATION OBTAIN DIR A
KASEL(1:0)	- DIA A KEY ADDRESS SOURCE SELECT
KBSEL(1:0)	- DIA B KEY ADDRESS SOURCE SELECT
HWSEL	- HIGH WORD DATA ADDR. SOURCE SELECT
LWSEL	- LOW WORD DATA ADDR. SOURCE SELECT
SPYMD(1)	- SPY HIT ON BLOCK1 MODIFIED DATA
SPYMD(0)	- SPY HIT ON BLOCK0 MODIFIED DATA
SEL-TL	- BOTH DIR ARE ACCESSED BY SAME OPERATION
UPDATE-A	- MISS OPERATION REQUIRES TO UPDATE CACHE
BLK32-EN	- WORD 32 BLOCK ENABLE
BLK32ENL	- WORD 32 BLOCK ENABLE LOW
BLK10-EN	- WORD 10 BLOCK ENABLE
BLK10ENL	- WORD 10 BLOCK ENABLE LOW
MDO-EN	- RETURN SPY WITH MODIFIED DATA
MDO-SEL	- SELECT BACKPLANE SOURCE AS SPY MEM REG.
FRAMA	- IO PIN FOR FLAG RAM COPY A
FRAMB	- IO PIN FOR FLAG RAM COPY B
FRAMC	- IO PIN FOR FLAG RAM COPY C
FLGER	- CORRECTABLE FLAG ERROR
ERROR	- BROKEN
COPY0	- COPY FLAG RETURN TO SPY REQUEST
INH0	- MOD. FLAG RETURN TO SPY REQUEST

12.4.3 DETAILED DESCRIPTION

12.4.3.1 W-STATE MACHINE

The definition of the states of the W-state machine are:

State 0 : idle state.
State 1 : Write operation hit at cache. Ready to write and update flag (status) RAM DIR-B or DIR-A.
State 2 : Finish write operation.
State 3 : not used.
State 4 to State 7 : waiting to update flag ram dira.

12.4.3.2 J-STATE MACHINE

The definition of the states of the J-state machine are:

State 0 : idle state.
State 1 : operand read first operation hit at block1.
State 2 : operand read first operation hit at block0.
State 3 : operand read to be finished.
State 4 : operand read both operations hit at block0.
State 5 : operand read first operation hit at block1, second operation hit at block0.
State 6 : operand read first operation hit at block0, second operation hit at block1.
State 7 : operand read both operations hit at block1.

12.4.3.3 SPY OPERATIONS

Spy operations is syncronized in this array. A signal called busphase sent by MCNTL array does the syncronization of memory bus. Spy command is loaded only when busphase is high. There are two stages of pipeline which match spy operation return timing, doing most of the control job. When spy hits on a modified data in this cache, then cache needs to response to the request. In cases such as IO write to memory, second MCACM read public on private data or second MCACM read private on copy data, the first MCACM will change the flag status. Following are spy command decode :

00 : IOMC READ PUBLIC.
01 : MCACM READ PUBLIC.
02 : IOMC WRITE.
03 : IOMC READ MODIFY WRITE or MCACM READ PRIVATE.

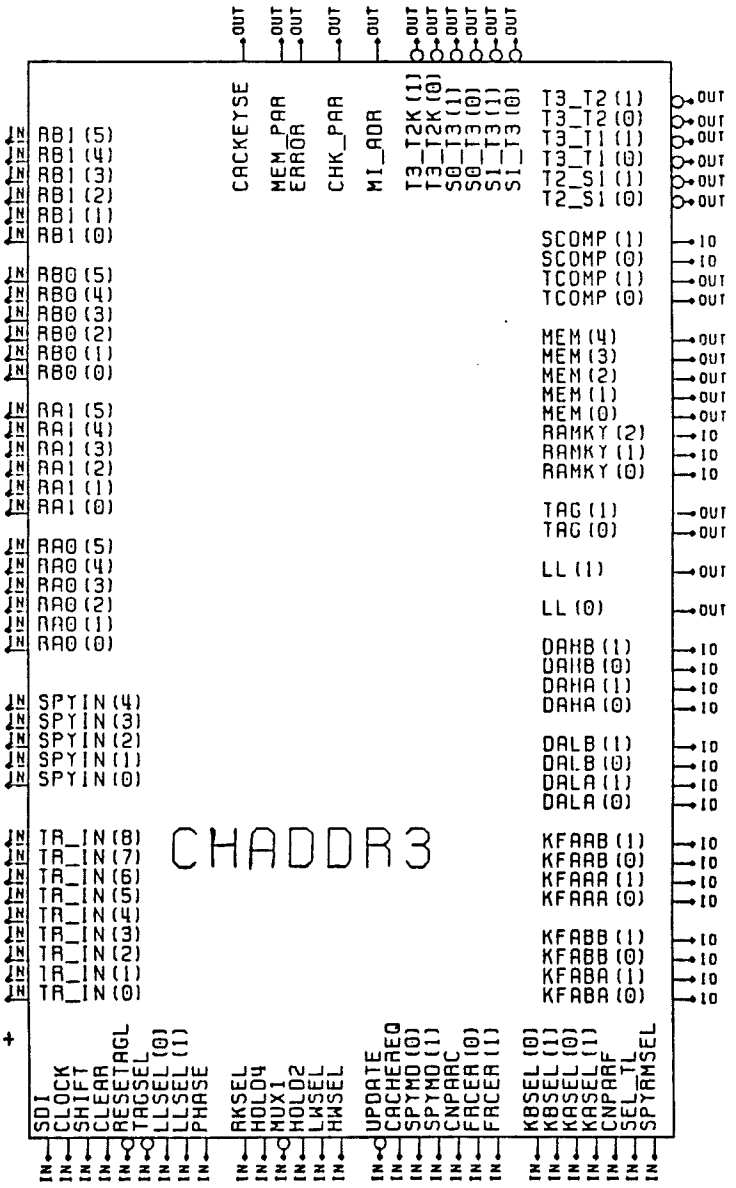
12.4.3.4 FAULT DETECTION

The CAFLAG6 array uses 3 different fault detection techniques These are: parity generation/checking, duplicate and compare checking, and triple-modular-redundancy fault tolerance.

The duplicate and compare protection is provided by duplicating the CAFLAG6 arrays. One is configured as the master; it drives control signals. The other is configured as the slave; it checks the control signals from the master. Any differences are reported as hardware errors.

Triple-modular-redundency protection is provided to detect and correct errors in the flag rams. Three separate flag rams are written with identical data from three separate sets of drivers. Voting is employed when reading these rams to read the correct data in the prescence of single-bit errors. All single-bit errors are reported to the processor for error logging.

12.5 CHADDR3


CHADDR3
CACKEYSE
MEM_PAR
ERROR
CHK_PAR
MI_ADR
T3_T2K(1)
T3_T2K(0)
S0_T3(1)
S0_T3(0)
S1_T3(1)
S1_T3(0)
T3_T2(1)
T3_T2(0)
T3_T1(1)
T3_T1(0)
T2_S1(1)
T2_S1(0)
SCOMP(1)
SCOMP(0)
TCOMP(1)
TCOMP(0)
MEM(4)
MEM(3)
MEM(2)
MEM(1)
MEM(0)
RAMKY(2)
RAMKY(1)
RAMKY(0)
TAG(1)
TAG(0)
LL(1)
LL(0)
DAHB(1)
DAHB(0)
DAHA(1)
DAHA(0)
DALB(1)
DALB(0)
DALA(1)
DALA(0)
KFAAB(1)
KFAAB(0)
KFAAA(1)
KFAAA(0)
KFABB(1)
KFABB(0)
KFABA(1)
KFABA(0)
RB1(5)
RB1(4)
RB1(3)
RB1(2)
RB1(1)
RB1(0)
RB0(5)
RB0(4)
RB0(3)
RB0(2)
RB0(1)
RB0(0)
RA1(5)
RA1(4)
RA1(3)
RA1(2)
RA1(1)
RA1(0)
RA0(5)
RA0(4)
RA0(3)
RA0(2)
RA0(1)
RA0(0)
SPYIN(4)
SPYIN(3)
SPYIN(2)
SPYIN(1)
SPYIN(0)
TA_IN(8)
TA_IN(7)
TA_IN(6)
TA_IN(5)
TA_IN(4)
TA_IN(3)
TA_IN(2)
TA_IN(1)
TA_IN(0)
SDI
CLOCK
SHIFT
CLEAR
RESETAGL
TAGSEL
LLSEL(0)
LLSEL(1)
PHASE
AKSEL
HOLD4
MUX1
HOLD2
LWSEL
HWSEL
UPDATE
CACHEREQ
SPYMD(0)
SPYMD(1)
CNPARC
FACEA(0)
FACEA(1)
KBSEL(0)
KBSEL(1)
KASEL(0)
KASEL(1)
CNPARF
SEL_TL
SPYAMSEL

12.5.1 FUNCTION OVERVIEW

The CHADDR array contains the pipelines for spy address and translate address/length/lsd/tag.

12.5.2 INTERFACES

12.5.2.1 INPUTS

SDI	- serial data input.
CLOCK	- SYSTEM CLOCK.
SHIFT	- MAINTENANCE SHIFT
CLEAR	- SYSTEM CLEAR.
TAGSEL	- RBUS TAG SELECT.
LLSEL(1:0)	- LSD/LENGTH SELECT.
PHASE	- BUS CLOCK.
RKSEL	- RAM KEY INPUT SELECT.
HOLD4	- RAM KEY REGISTER HOLD ENABLE.
MUX1	- TRANSLATE PIPELINE INPUT SELECT.
HOLD2	- PIPELINE STAGE 2 HOLD.
LWSEL	- CACHE DATA LOWER WORD ADDRESS SELECT.
HWSEL	- CACHE DATA HIGHER WORD ADDRESS SELECT.
UPDATE	- CACHE UPDATE.
CACHEREQ	- CACHE REQUEST.
SPYMD(1:0)	- SPY MODE FOR BLOCK 1/0.
CNPARC	- CONTROL SIGNAL PARITY FROM CACNTL.
FRCER(1:0)	- FORCE ERROR.
KBSEL(1:0)	- DIRECTORY B ADDRESS SELECT.
KASEL(1:0)	- DIRECTORY A ADDRESS SELECT.-
CNPARF	- CONTROL SIGNAL PARITY FROM CAFLAG.
SEL TL	- DIRECTORY COMPARE MODE ENABLE.
SPYRMSEL	- SPY DIRECTORY SELECT.
RA1(5:3)	- KEY FROM DIRECTORY A, BLOCK 1.
RA1(2:0)	- ECC OF KEY FROM DIRECTORY A, BLOCK 1.
RB1(5:3)	- KEY FROM DIRECTORY B, BLOCK 1.
RB1(2:0)	- ECC OF KEY FROM DIRECTORY B, BLOCK 1.
RA0(5:3)	- KEY FROM DIRECTORY A, BLOCK 0.
RA0(2:0)	- ECC OF KEY FROM DIRECTORY A, BLOCK 0.
RB0(5:3)	- KEY FROM DIRECTORY B, BLOCK 0.
RB0(2:0)	- ECC OF KEY FROM DIRECTORY B, BLOCK 0.
SPYIN(4:0)	- SPY PIPELINE INPUT.
TR IN(8:0)	- TRANSLATE PIPELINE INPUT.
RESETAGL	- ACTIVE LOW SIGNAL TO RESET TAG PIPELINE.

12.5.2.2 OUTPUTS

CACKEYSE - CACHE KEY SINGLE ERROR.
MEM PAR - MEMORY PARITY.
ERROR - FATAL ERROR.
CHK PAR - PARITY OF SPY PIPELINE 2 AND TRANSLATE ADDRESS PIPELINE
MI ADR - DIGIT ADDRESS OF THE REQUEST.
S0 T3(1:0)
T3 T1(1:0)
S1 T3(1:0)
T3 T2(1:0)
T3 T2K(1:0)
T2 S1(1:0) - COMPARISON RESULTS OF THE ADDRESS PIPELINES.
SCOMP(1:0) - SPY HIT OF BLOCK 1/0.
TCOMP(1:0) - TRANSLATE HIT OF BLOCK 1/0.
MEM(4:0) - REQUEST ADDRESSS.
RAMKY(2:0) - ECC PART OF THE DIRECTORY INPUT.
TAG(1:0) - RBUS TAG.
LL(1:0) - LSD/LENGTH.
DAHA(1:0)
DAHB(1:0) - CACHE DATA HIGHER WORD ADDRESS.
DALA(1:0)
DALB(1:0) - CACHE DATA LOWER WORD ADDRESS.
KFAAA(1:0)
KFAAB(1:0) - DIRECTORY/FLAG A ADDRESS.
KFABA(1:0)
KFABB(1:0) - DIRECTORY/FLAG B ADDRESS.

12.5.3 DETAILED DESCRIPTION

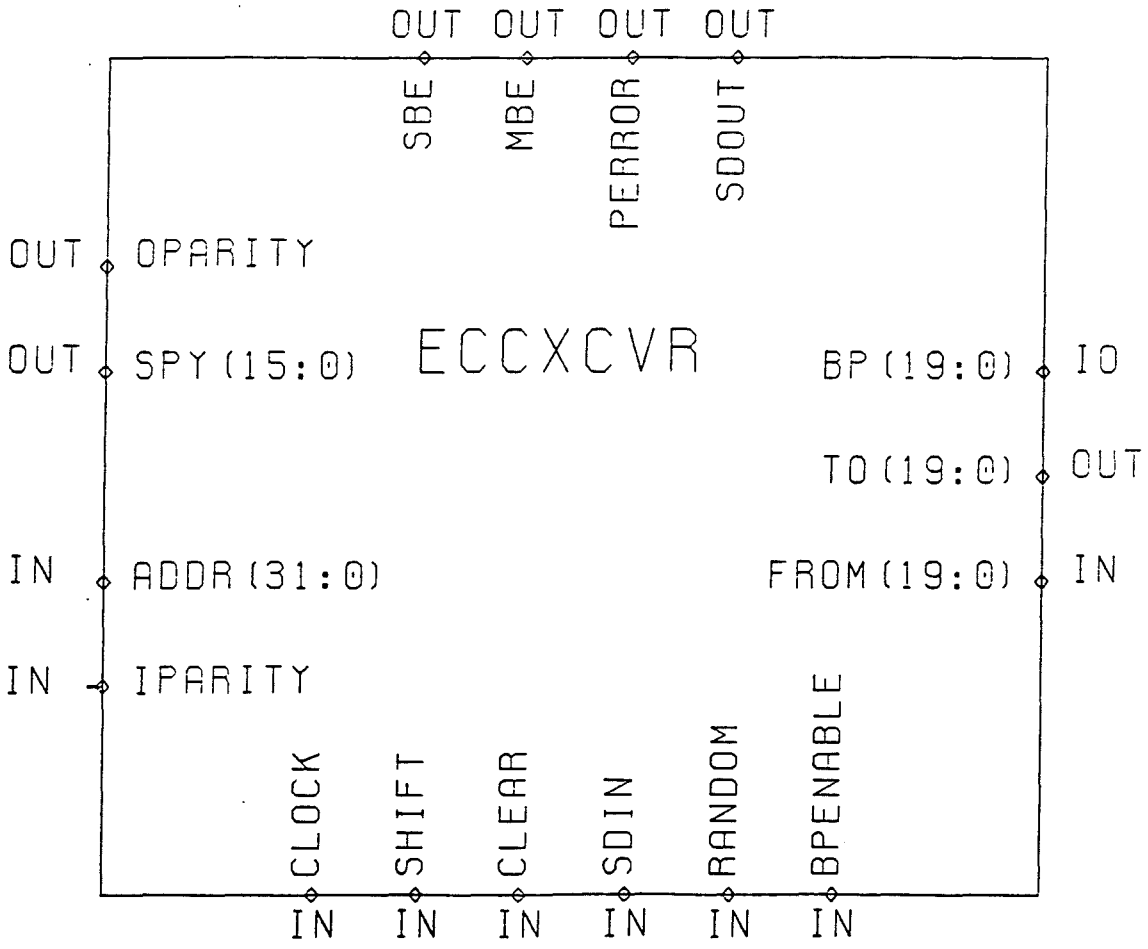
Each CHADDR3 is capable of processing 5-bit address and 3-bit key. Using five CHADDR arrays together, we can process 25-bit address and 15-bit key. The key inputs are from two directories, each directory has two blockes, therefore there are four 3-bit key inputs to the array, each key input is protected by a 3-bit SECDED ECC.

The selected key is compared with the translate/spy address to determine if there is a hit. Comparisons are also made among the translate/spy address pipelines, the result is sent to other cache arrays to determine the state of the cache module.

This array is used to address the cache data ram and the cache directories. Input to the cache directory rams are from this array also.

All the control signals to this array are protected by parity. Key input are protected by SECDED ECC. Address drivers to cache data ram and cache directories are protected by duplication such that data and ecc are addressed by physically different lines and if one of the two is broken, the addressed data and addressed ecc will from different locations which will be detected by error correcting logic.

12.6 ECCXCV


OUT OUT OUT OUT
SBE
MBE
PERROR
SDOUT
OUT OPARITY
OUT SPY(15:0)
ECCXCVR
BP(19:0) IO
TO(19:0) OUT
IN ADDR(31:0)
FROM(19:0) IN
IN IPARITY
CLOCK
SHIFT
CLEAR
SDIN
RANDOM
BPENABLE
IN IN IN IN IN IN

12.6.1 FUNCTION OVERVIEW

The ECCXCV option is an ECC generator and SEC-DED-4ED bidirectional 32 bit bus transceiver. Two ECCXCV PGA's in a mirror image configuration are required to implement the function because of the limited number of 25 ohm drivers on the MCA2 chip.

The ECCXCV option operates in a psuedo-full-duplex mode; as both a transmitter and a receiver at the same time. This mode reduces the clock-to-output and setup-to-clock delays. The chip contains separate logic for the transmit function: ECC calculation logic, output registers, and 25 ohm drivers; and the receive function: input registers, syndrome generation logic, and error correction circuitry. Both of these functions operate at the same time; although, it is obvious that the chip will receive what it transmitted, and is not true full-duplex.

12.6.2 INTERFACES

12.6.2.1 INPUTS

ADDRESS(31:0)
: This includes the absolute binary board address field, the module select field, the command field, and the source ID field.

IPARITY
: Even parity over the ADDRESS(31:0) and IPARITY fields.

BPENABLE
: The 25 ohm driver enable signal. This signal is not latched internally and must remain high throughout the 2 clock memory bus cycle.

FROM(19:0)
: The other half of the received memory address bus. This is sourced by the mirror image ECCXCV chip.

CLOCK
: Normal system clock.

SHIFT
: Serial shift chain enable signal from the clock and maintenance chip.

CLEAR
: Maintenance reset.

SDIN
: Shift chain input.

RANDOM
: A self-test signal which enables a random number generator on the shift chain. When the ECCXCV option is in random mode, it will generate a random cycle of 256 numbers which can be used to simulate activity on the memory address bus to test the cache spy.

12.6.2.2 BIDIRECTIONALS

BP(19:0)
: Half of the 39 bit (including ECC) memory address bus. One of the arrays handles 20 bits and the other handles 19 bits; it doesn't matter which is which.

12.6.2.3 OUTPUTS

SPY(15:0)
: Half of the received and corrected 32 bit memory address bus.

OPARITY
: Even parity over the entire received and corrected 32 bit memory bus and the OPARITY bit. Both mirror image arrays output parity bits; but only the array which detected and corrected a single bit error has the correct parity. When there are no errors then both parities are identical.

PERROR
: The input parity on the ADDRESS(31:0) and the IPARITY was invalid.

TO(19:0)
: The complement of FROM(19:0); each chips receives half of the memory address bus and then sends its half to the other chips so that each chips gets the whole bus without putting 2 loads on the backplane.

SDOUT
: The shift chain output.

12.6.3 DETAILED DESCRIPTION

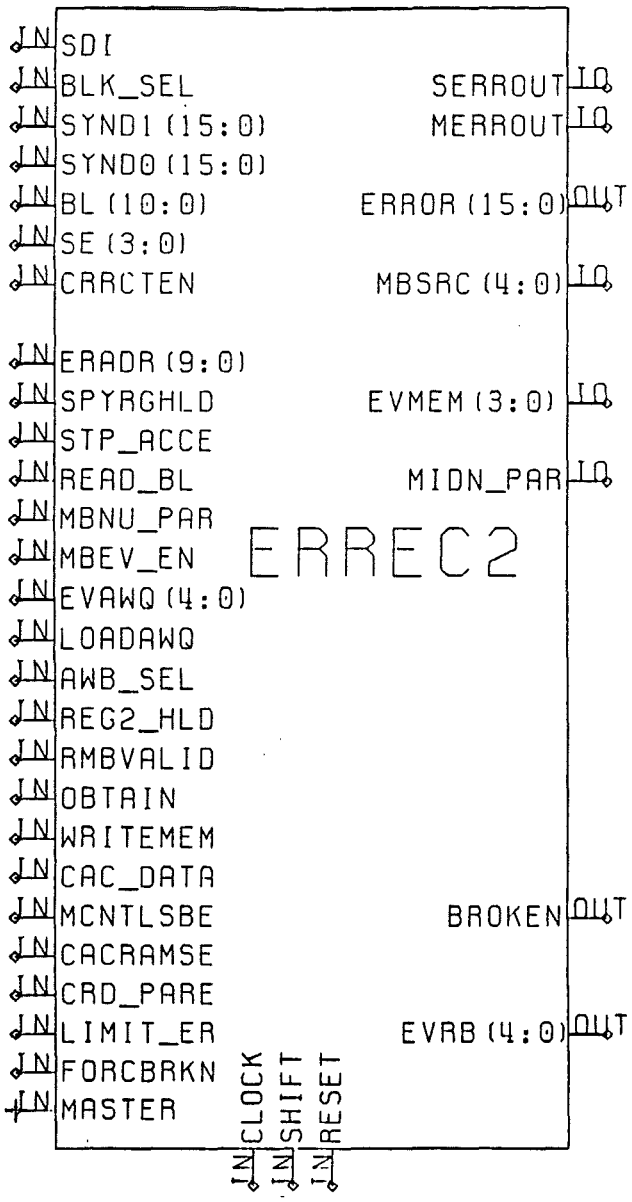
The internal bit ECCLPW causes the ECCXMT register to accumulate a longitudinal parity word. This feature will be used in conjunction with path tests which cannot drive the backplane; Errors will be disabled and the MCACM will be set-up into a known state, then a number of clocks will be issued, the result in this register will give a go/no-go indication of the cache hit/miss logic and address paths.

12.6.4 SHIFT CHAIN DEFINITION

Since this chip must be operated in tandem with another ECCXCV chip in a mirror image configuration, the shift chain is defined for the pair. It should be noted, that the upper chip has the low order bits at the top of each register. The lower chip is in the normal low, at bottom format.

```
        RCVECC(4:6)
U       DONTCARE        - this bit is tied to XMTECC(3) only
P       RCV(16:31)
P       XMTECC(3:6)     - XMTECC(3) does not drive the B.P.
E       ECCLPW          - see 12.6.3 for definition
R       XMT(16:31)
------------------------
L       RCVECC(3:0)
O       RCV(15:0)
W       XMTECC(3:0)
E       ECCLPW          - see 12.6.3 for definition
R       XMT(15:0)
```

12.7 ERREC2


IN SDI
IN BLK_SEL
IN SYND1 (15:0)
IN SYND0 (15:0)
IN BL (10:0)
IN SE (3:0)
IN CRRCTEN
IN ERADR (9:0)
IN SPYRGHLD
IN STP_ACCE
IN READ_BL
IN MBNU_PAR
IN MBEV_EN
IN EVAWQ (4:0)
IN LOADAWQ
IN AWB_SEL
IN REG2_HLD
IN RMBVALID
IN OBTAIN
IN WRITEMEM
IN CAC_DATA
IN MCNTLSBE
IN CACRAMSE
IN CRD_PARE
IN LIMIT_ER
IN FORCBRKN
IN MASTER
ERREC2
SERROUT IO
MERROUT IO
ERROR (15:0) OUT
MBSRC (4:0) IO
EVMEM (3:0) IO
MIDN_PAR IO
BROKEN OUT
EVRB (4:0) OUT
IN CLOCK
IN SHIFT
IN RESET

12.7.1 FUNCTIONAL OVERVIEW

This array receives all the error signals generated in MCACM module for each active request and encodes them into the error code to be sent back to the requester. The syndrome and address of the worst error had happened since the last read error report/address command pair are registered in this array. If any kind of 'BROKEN' errors has occured, a broken signal will be sent to stop the module.

12.7.2 INTERFACES

12.7.2.1 INPUTS

SDI	- Serial Data Input.
CLOCK	- system CLOCK.
RESET	- maintenance RESET.
SHIFT	- maintenance SHIFT.
SYND1(15:0)	- SYNDrome of cache data block 1.
SYND0(15:0)	- SYNDrome of cache data block 0.
SNGL-ER	- cache data SiNGLe bit ERror.
BLK-SEL	- cache data BLocK SELect.
BL(13:0)	- Base/Limit.
ERADR(15:5)	- error address.
STP-ACCE	- stop accumulating error.
READ-BL	- read Base/Limit.
EVAWQ(4:0)	- error vector from AWQ.
LOADAWQ	- load the pipeline with EVAWQ.
REG2-HLD	- pipeline register 2 HOLD.
AWB-SEL	- pipeline output select.
MB-LTCHL	- MBus valid LOW.
OBTAIN	- obtain mbus.
WRITEOPL	- memory WRITE OPeration Low.
CAC-DATA	- CAChe DATA is being corrected by MCECC.
MCNTLSBE	- MCNTL Single Bit Error.
CACRAMSE	- CAChe key/flag RAM Single Error.
CRD-PARE	- CoRrected Data (from MCECC) PARity Error.
LMTER(1:0)	- LiMiT ERror.
FORCBRKN	- FORCe BRoKeN.
MBNU-PAR	- PARity of NUmber of Memory Boards.
MBEV-EN	- MBus memory Error Vector output ENable.

12.7.2.2 BIDIRECTIONALS

MBSRC(4:0) - MBus SouRCe.
EVMEM(3:0) - Memory Error Vector.
MIDN-PAR - PARity of Mbus source ID and board Number

12.7.2.3 OUTPUTS

SALL0L-O - Syndrome ALL 0 Low, Output.
ERROR(15:0) - ERROR address/report or base/limit.
BROKEN - BROKEN.
EVRB(4:0) - Error Vector to RBus.
EVRB-PAR - PARity of Error Vector.

12.7.3 DETAILED DESCRIPTION

This array has the same pipeline structure as the one in the cache control arrays to maintain synchronization of the encoded error vector. The tables in Figures 12-4, 12-5, and 12-6 show this encoding of the error vector for the different kinds of errors.

error code	condition of the error signals	pipeline stage
0	no errors	
1	EVMEM = 1 or 3	2
2	EVMEM = 2 or 4	2
3	MCNTLSBE	2
4	SNGL-ER and CAC-DATA	3
	SNGL-ER and (not CAC-DATA)	2
5	CACRAMSE	1

Fig. 12-4 Encoding of non-fatal errors

12.7.3 DETAILED DESCRIPTION (Continued)

error code	condition of the error signals	pipeline stage
0	EVMEM = 5	2
1	LMTER	1
2-9	from AWQUEUE	
5	CRD-PARE and CAC-DATA	3
	CRD-PARE and (not CAC-DATA)	2

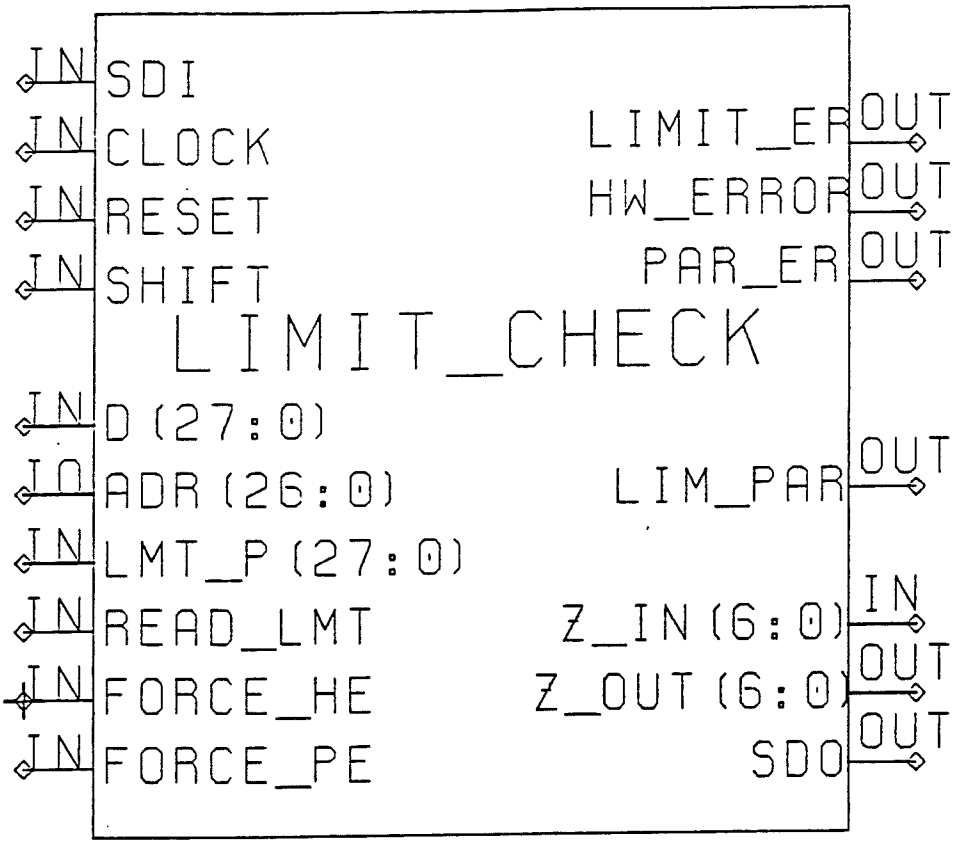
Fig. 12-5 Encoding of fatal errors

error code	condition
A	EVMEM = 6 or 7; EVMEM parity error; EVADR, MSRC-PAR, and MBRD-PAR parity error
B	MCECC multiple error
E	write-back failure, EVMEM = 6 or 7; EVMEM parity error after memory write operation.
F	LMTER(0) NEQ LMTER(1)

Fig. 12-6 Encoding of BROKEN errors

For read error report/address operations, two 16-bit registers in this array are used to store the worst error syndrome/address. Two ERREC arrays are needed in the MCACM module, each capable of storing half of the worst error syndrome/address. This is required because of the limitation of pins available on an array.

12.8 LMTCHK


IN SDI
IN CLOCK
IN RESET
IN SHIFT
LIMIT_CHECK
IN D(27:0)
IO ADR(26:0)
IN LMT_P(27:0)
IN READ_LMT
IN FORCE_HE
IN FORCE_PE
LIMIT_ER OUT
HW_ERROR OUT
PAR_ER OUT
LIM_PAR OUT
Z_IN(6:0) IN
Z_OUT(6:0) OUT
SDO OUT

12.8.1 FUNCTIONAL OVERVIEW

This array is used to check if the absolute memory address obtained by Base Adder is greater than the limit. A 28-bit parity tree that checks the parity over base/limit from base/limit table is also included.

12.8.2 INTERFACES

12.8.2.1 BIDIRECTIONALS

ADR(27:0) -
When READ-LIMIT is low, this is the absolute binary address which is to be compared to the limit. When READ-LIMIT is high, this is used to output the limit to the absolute address bus so that the limit table entry can be read by the processor.

12.8.2.2 INPUTS

READ-LIMIT - enables the output of limit register to ADR.
LIMIT(27:0) - limit from limit table.
SHIFT - maintenance shift.
RESET - maintenance reset.
CLOCK - system clock.
SDI - serial data input of the shift chain.
Z-IN(6:0) - ECC from the other LIMCHK array.
D(27:0) - input of the 28-bit parity tree.

12.8.2.3 OUTPUTS

LIMIT-ERROR - limit error output.
HW-ERROR - hardware error, output of the TSC.
SDO - serial data output of the shift chain.
Z-OUT(6:0) - ECC to the the other LIMCHK array.
PAR-ERROR - parity error of D input.

12.8.3 DETAILED DESCRIPTION

12.8.3.1 LIMIT CHECKER

The limit and the one's complement of absolute memory address are added together by a Carry Look-Ahead Adder. If the result is negative, limit error is reported. The 7-bit ECC of the 27-bit result of this operation is generated and sent to a duplicate LMTCHK array for duplicate and compare protection.

12.8.3.2 READ LIMIT TABLE

The content of the limit register in LIMCHK may be written to the input pins of absolute address if the input READ-LIMIT is set, thus the absolute address and the limit may share the same bus.

12.8.3.3 BASE/LIMIT TABLE PARITY CHECKER

A 28-bit parity tree is also incorporated in LIMCHK. Since there are two LIMCHK arrays in the system, one will check the parity of base table output and the other will check the parity of limit table output.

12.8.3.4 FAULT DETECTION

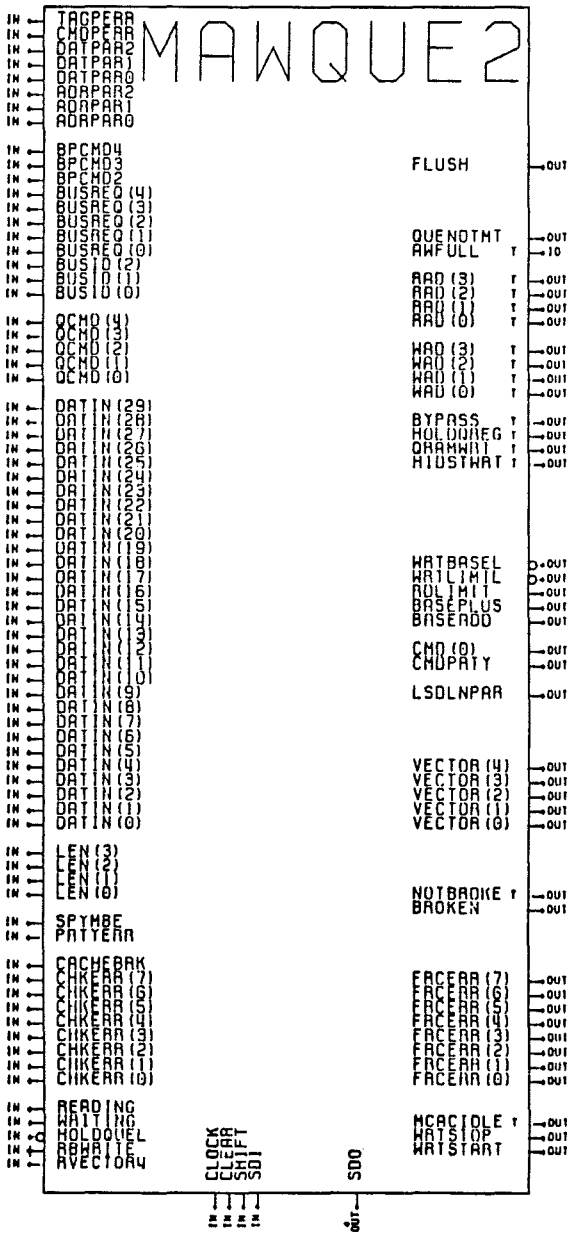
BASADD and LIMCHK are protected by duplicating these arrays and compare the result with each other. In the LIMCHK array, the ECC of the result of CLA are generated and sent to the other LIMCHK array working in parallel. Each LIMCHK compares the ECC generated and the ECC received from the other LIMCHK by a TSC (totally self checking) logic, the disagreement of the ECCs will cause the HW-ERROR signal be set.

The output of TSC logic and parity tree, HW-ERROR and PAR-ERROR, may be set by input pins FORC-HW-E and FORC-PAR-E. They can be used to make sure that the error detection logic themselves are healthy.

12.8.4 SHIFT CHAIN DEFINITION

ADDRESS(26:0) - The absolute binary address from BASADD.
LIMPAR - The result of the parity check on the input.
LIMIT(26:0) - The binary limit from the limit table.

12.9 MAWQUE2



12.9.1 FUNCTION OVERVIEW

The MAWQUE2 controls the MCACM input queue, which is the interface between the processor and cache. The MAWQUE2 drives 14 RAMARR arrays as a 16-word by 100-bit FIFO queue structure. The MAWQUE2 array detects undigit, invalid length, invalid command, and bus parity errors. The MAWQUE2 also pre-codes the command for operand read cases to the cache control.

The MAWQUE2 array handles several simple commands: write base table entry, write limit table entry, and the different varieties of no-operation commands. The MAWQUE2 also handles the commands which require global, module-level status: write-pc, flush, and force error.

The MAWQUE2 collects the hardware broken signals from the other arrays in the memory control and cache module. These broken signals and the internaly generated not-broken are then sent to the CLKMNT array.

12.9.2 INTERFACES

12.9.2.1 INPUTS

TAGPERR -
The parity of the tag and tag parity of the entry at the top of the queue is incorrect.

CMDPERR -
The parity of the command, length, and command parity of the entry at the top of the queue is incorrect.

DATPAR2 & DATPAR1 & DATPAR0 -
These 3 lines are the result of 3 14-input parity trees which check the parity over the data and data parity of the entry at the top of the queue. If the parity of these 3 lines is not even (that is, 0 or 2 inputs are active), then the entry at the top of the queue has a data parity error.

12.9.2.1 INPUTS (Continued)

ADRPAR2 & ADRPAR1 $ ADRPAR0 -
These 3 lines are the result of 3 14-input parity trees which check the parity over the address and address parity of the entry at the top of the queue. If the parity of these 3 lines is not even (that is, 0 or 2 inputs are active), then the entry at the top of the queue has an address parity error.

BPCMD4 & BPCMD3 & BPCMD2 -
The 3 most significant bits of the command field from the AW bus are used to detect a flush command.

BUSREQ(4:0) -
These 5 lines are the AW bus requests from the processor modules.

BUSID(2:0) -
This is an identification number of the winner of the AW bus arbitration. The MAWQUE2 array uses this field to check for arbitration errors on the AW bus.

QCMD(4:0) -
The 5-bit command of the entry at the top of the queue. This is the command which is currently being handled by the MAWQUE2.

DATIN(29:0) -
These 30 bits are portions of the address field of the entry at the top of the queue. Only enough bits are included here to check for operand read cases and undigits in the address field.

LEN(3:0) -
The 4-bit length field of the entry at the top of the queue. The length of the request from the processor is used to check for operand read cases. The parity of this field and the least significant digit of the address (LSD) are sent to cache for fault detection.

12.9.2.1 INPUTS (Continued)

SPYMBE -
The MCNTL5 has detected a multiple-bit error in the main memory address bus. This is a severe error which could jeopardize the consistency of data in the cache. Therefore the MAWQUE2 sets the broken signal to the SMC which stops the clocks to the system.

PRTYERR -
When cache detects an internal failure, it asserts this signal to the MAWQUE2. Cache operation is critical to consistency of data in the cache, so this sets the broken signal to the SMC.

CACHEBRK -
A failure in the duplicate-and-compare portions of the cache logic is reported with this signal. This results in the setting of the broken signal to the SMC.

CHKERR(7:0) -
Other sections of MCACM report errors with these signals. All of these are system critical and cause the broken signal to be set to the SMC. These 8 inputs also serve a second role which is to monitor the force-error logic.
This is a self-test feature which is initiated by a processor module that forces an internal error in MCACM at a predictable time. At this time the NOTBROKEN signal is also sent to the SMC. If all is well, the BROKEN and NOTBROKEN signals cancel out. If the fault detection circuitry is broken, or cannot report a fault, it is detected.

READING -
This is a cache status signal. It is active whenever cache is currently working on any kind of a read operation. This signal is used to hold flush operations until all outstanding reads have finished. It is also combined with WRITEING to generate MCACMIDLE.

12.9.2.1 INPUTS (Continued)

WRITEING -
This is a cache status signal. It is active whenever cache is currently working on any kind of a write operation. This signal is used to hold write-pc operations until all outstanding writes have finished. It is also combined with READING to generate MCACMIDLE.

HOLDQUEL (active low) -
This is a cache status signal. It is asserted by cache whenever cache is busy and cannot accept another command. Valid commands remain at the top of the queue until HOLDQUEL goes inactive, indicating that cache has accepted the command.

RBWRITE -
This is a cache status signal. It is asserted on the clock during which a tag for the completion of a write operation is being driven onto the RBUS.

RVECTOR4 -
This is the most significant bit of the error vector being driven onto the RBUS. It indicates that the data being driven this clock (which corresponds to the tag driven last clock) contains a fatal error. This is used by MAWQUE2 to determine when a write error has occured which starts the write abort mode.

CLOCK -
System clock input.

CLEAR -
System clear inputs.

SHIFT -
System mode input. This activates the maintenance chain.

SDI -
Serial data input to the maintance chain in the MAWQUE2.

12.9.2.2 BIDIRECTIONALS

AWFULL -
This signal serves 2 functions. As an input, it is the highest priority requestor onto the AW bus. When a processor drives the AWFULL signal, it effectively locks all modules (including itself) off of the AW bus.
As an output, this signal prevents a processor module from requesting the bus because the MCACM input queue is full.

12.9.2.3 OUTPUTS

FLUSH -
After a processor module has sent a FLUSH command, the MCACM does not return any data from outstanding read operations to the processor. This is simply done by disabling the RBUS valid driver.

QUENOTMT -
When the input queue is not empty, it contains something that the cache should accept and handle. This name is a little confusing: it was chosen because there might be entries in the queue which are not valid.

RAD(3:0) -
This is the read address of the queue ram in the RAMARR arrays. The read address is the top of the queue.

WAD(3:0) -
This is the write address of the queue ram in the RAMARR arrays. The write address is the bottom of the queue.

12.9.2.3 OUTPUTS (Continued)

BYPASS -
When the queue ram (in the RAMARR arrays) is empty, any incomming data is loaded into the queue ram and directly into the queue register at the output of the queue. The queue ram is bypassed.

HOLDQREG -
When the cache asserts HOLDQUEL and there is something in the output of the queue, then the queue register must hold whatever it is. Otherwise, the queue register loads from the output of the bypass mux.

QRAMWRT -
This signal is the write enable to the queue ram. During maintenance mode and shifting of the maintenance chains, this is disabled to prevent corrupting the hidden state in this RAM.

HIDSTWRT -
This signal is generated in maintenance mode to enable writing the hidden state of the queue ram. It switches the data input of the queue ram from the AW bus to the queue register. Therefore, it is possible to write into the queue ram even when there are no other modules on the AW bus.

WRTBASEL (active low) -
This signal writes the absolute address at the output of the baseadder into the base table.

WRTLIMIL (active low) -
This signal writes the absolute address at the output of the baseadder into the limit table.

RDLIMIT -
This signal gates the output of the limit table onto the absolute address bus by enableing the LMTCHK arrays to drive the bidirectional address lines.

12.9.2.3 OUTPUTS (Continued)

BASEPLUS -
This signal increments the output of the baseadder. The resultant absolute address is the address of the entry at the top of the queue plus the base table plus 1.

BASEADD -
This signal enables the baseadd. When it is low, only the BCD to binary portion of the baseadder is enable. This allows writing to the base and limit tables.

CMD(0) -
CMD(0) - This output replaces the least significant bit of the command sent to the cache. It allows MAWQUE2 to encode certain salient information into otherwise unused codes of the command field for cache.

CMDPRTY -
This output is the parity of the command sent to the cache. It includes the replacement CMD(0) bit.

LSDLNPAR -
Cache handles the least significant digit of the address and the 4-bit length field differently than the rest of the address. Therefore, MAWQUE2 provides a separate parity bit so that these lines can be protected in cache.

VECTOR(4:0) -
This is the encoded error vector. It indicates the most severe error (if any) detected by MAWQUE2 for the entry at the top of the queue.

NOTBROKE -
This output is sent to the SMC in response to a self test command from the processor. It is sent at the same time as one of the FRCERR(7:0) outputs which should result in an error being reported into the CHKERR(7:0) inputs.

BROKEN -
This is the result of the collection of the following inputs: CHKERR(7:0), CACHEBRK, PRTYERR, and SPYMBE.

12.9.2.3 OUTPUTS (Continued)

One of these outputs is driven in response to a self test command from the processor. The particular bit that is driven depends on the address of the entry at the top of the queue. This output should result in one of the CHKERR(7:0) inputs being set which causes BROKEN to be sent to the SMC.

MCACIDLE -
This output is sent to the processor and SMC whenever the input queue is empty and cache is neither reading nor writing.

WRITSTOP -
This is a MCACM status signal. It indicates that the MCACM is in a writeabort state. Write operations are being ignored because a write operation contained an error which, if allowed to continue, might corrupt main memory.

WRITSTART -
This is a MCACM status signal. It indicates that the MCACM is in a normal state. Write operations are being performed as requested.

SDO -
The output of the serial maintenance chain.

12.9.3 DETAILED DESCRIPTION

12.9.3.1 QUEUE CONTROL

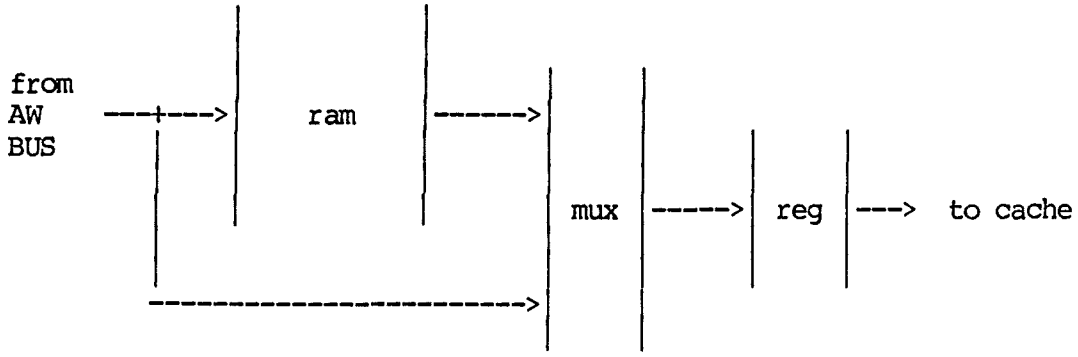


Fig. 12-7 FIFO queue structure

The operation of the queue controller is based on the type of command at the top of the queue. Certain commands are executed immediately, while others require external conditions before execution, and some require external conditions before being offered to the cache control.

The table in Figure 12-8 shows the classes of commands: the external conditions required before they are valid at the top of the queue, and the external conditions required before they are executed.

CLASS	CONDITIONS FOR VALID	CONDITIONS FOR EXECUTION
NORMAL	ALWAYS	GO/STOP
NOP	ALWAYS	IMMEDIATELY
READ	ALWAYS	GO/STOP
OPERANDABLE	ALWAYS	GO/STOP IF OPERAND THEN GO/STOP
FORCE ERROR	WAIT 1 CLOCK	ON SECOND CLOCK AT TOP OF QUEUE
FLUSH	NO READS PENDING	GO/STOP
WRITE PC	NO WRITES PENDING	GO/STOP

Fig. 12-8 Classes of commands handled by MAWQUE2

12.9.3.2 AWBUS COMMANDS PERFORMED BY MAWQUE2

WRITE BASE TABLE -

The MAWQUE2 array performs the write base table entry command separately from the cache control. The cache control will get this command and return the tag and error vector on the R-BUS, but will treat it as a NOP.

WRITE LIMIT TABLE -

The MAWQUE2 array performs the write limit table entry command exactly like the write base table entry command.

WRITE-PC -

The MAWQUE2 does not perform the write-pc command, but it does withhold it from the cache control section until all pending writes have been completed. This prevents writes from different instructions from completing out of order.

FLUSH -

The MAWQUE2 handles both the immediate and token portions of the flush operation. The MAWQUE2 withholds the flush command from the cache control section until all pending reads have completed. Flush commands are stackable; that is, up to 16 flush commands can be handled simultaneously, although this is highly unlikely.

12.9.3.3 INTERFACE TO CACHE CONTROL

The interface to cache control consists of 3 sections: command, control, and status.

The command interface consists of: the command as received from the AWBUS, and possibly modified for operand read; a command error flag; a queue valid signal; and a parity bit. The parity bit will be calculated such that the overall parity of the command interface, including the parity bit, is even.

The control interface is FLUSH and HOLDQUEL. The flush signal is sent to the cache control during a flush operation; it prevents read data from returning on the R-BUS. Note: write errors are still returned during a flush operation. The stop signal is sent by the cache control when its internal pipeline is full. The stop signal is treated like an active low pop queue signal.

The status interface consists of: READING, WRITING, and WRITE-ERROR. The reading, and writing signals report on the status of the commands inside the cache control pipeline. Every command is either a read or a write, so both reading writing will be false only when the cache control pipeline is empty. When an error is detected during a write command the write-error signal is sent to the A&W queue controller. This will suppress future write operations until a read-error-report and read-error-address command are received.

12.9.3.4 INTERFACE TO ERROR RECORDER

Catastrophic errors are reported to the A&W queue controller from the error recorder. The A&W queue controller then signals the CLKMNT array that the module is broken.

12.9.3.5 INTERFACE TO RBUS

The MAWQUE2 array monitors the RBUS error vector to determine if a write operation has gotten an error. When a write operation gets an error, the MCACM enters a write-abort state to prevent further write operations from occuring.

12.9.3.6 FAULT DETECTION

The test-the-checker feature starts from the MAWQUE2 array, which recognizes this command from the processor. The command is held for 1 clock to stabilize the decodeing and then the force-error outputs and the NOTBROKEN signal are both enabled.

12.9.3.7 CHIP TEST FACILITIES

To facilitate fault detection and chip test, there is a high level of internal fault detection. The RAD and WAD counters, and the command decoder are protected by parity prediction. The nested-flush counter is protected by detecting a negative count which should not occur.

12.9.3.8 BOARD TEST FACILITIES

All board-external inputs are registered except the command field of the A&W bus which is used to detect flush commands. The queue control logic contains the necessary enables and disables to allow hidden state reading and writing of the queue RAMs.

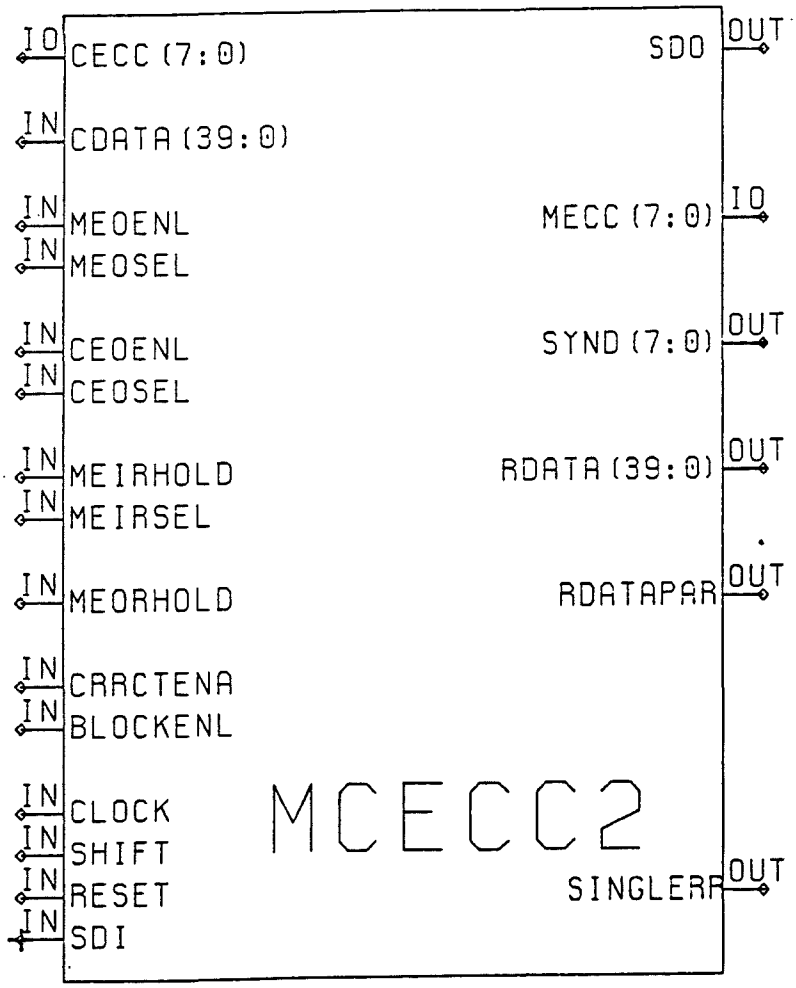
A flip-flop is provided inside the MAWQUE2 array which forces the input queue to recirculate and not go empty. This allows board testing at the functional level by preloading the RAMS with 16 test commands.

12.9.4 SHIFT CHAIN DEFINITION

```
WADPAR      1;    % PREDICTED WRITE ADDRESS PARITY
WAD         4;    % WRITE ADDRESS FOR AW QUEUE RAMS
RADPAR      1;    % PREDICTED READ ADDRESS PARITY
RAD         4;    % READ ADDRESS FOR AW QUEUE RAMS
QCTRLBRK    1;    % QUEUE CONTROL HAS DETECTED A FAULT
MSTATE      2;    % MAWQUE MAINTENANCE MODE REGISTER
                  %   0 - NORMAL OPERATION
                  %   1 - HIDDEN STATE WRITE OF QUEUE RAM
                  %   2 - QUEUE RECIRCULATION MODE
                  %   3 - HIDDEN STATE WRITE BASE/LIMIT

QUEFULL     1;    % TOP OF QUEUE CONTAINS A COMMAND
SECOND      1;    % TOP OF QUEUE IS IN SECOND OF 2 CLOCKS
WRTSTOP     1;    % WRITE ABORT (SENT TO STOP AS WRTSTOP)
RBISWRT     1;    % THE RETURN ON RBUS IS FROM A WRITE
FLUSHES     4;    % NUMBER OF FLUSHES IN AW QUEUE
NOBUS       1;    % AW INPUT QUEUE IS FULL - NO INPUT
                  % WILL BE ACCEPTED.
BUSPR       3;    % BUS PRIORITIZATION OF LAST BUSREQ'S
REGID       3;    % REGISTERED FROM BUSID(3:0)
```

12.10 MCECC2


IO CECC(7:0)
IN CDATA(39:0)
IN MEOENL
IN MEOSEL
IN CEOENL
IN CEOSEL
IN MEIRHOLD
IN MEIRSEL
IN MEORHOLD
IN CRRCTENA
IN BLOCKENL
IN CLOCK
IN SHIFT
IN RESET
IN SDI
MCECC2
SDO OUT
MECC(7:0) IO
SYND(7:0) OUT
RDATA(39:0) OUT
RDATAPAR OUT
SINGLERR OUT

12.10.1 FUNCTION OVERVIEW

The main function of MCECC2 is Error correction/detection for data read out of CACHE data ram or data received from memory data bus. It also has the logic for ECC generation, and the path for Memory bus interface for ECC. While doing error correction, the syndrome, single error signals are generated and sent to the error recorder. The corrected data are sent to four ROTCATs for rotate/mask/merge operations as specified.

12.10.2 INTERFACES

12.10.2.1 INPUTS

MEOENL	-	Memory Ecc bus Output ENable Low.
MEOSEL	-	Memory Ecc bus Output SELect.
CEOENL	-	Cache Ecc bus Output Enable Low.
CEOSEL	-	Cache Ecc bus Output SELecl.
MEIRHOLD	-	Memory Ecc Input Register HoLD.
MEIRSEL	-	Memory Ecc Input Register SELect.
MEORHOLD	-	Memory Ecc Output Register HoLD.
CRRCTENA	-	CoRReCTion ENable.
BLOCKENL	-	BLocK ENable Low (corrected data output enable low).
CDATA	-	Cache DATA bus.
SDI	-	Serial Data Input.
RESET	-	Maintenance input.
CLOCK	-	System clock input.
SHIFT	-	Maintenance input: enables the maintenance shift chain.

12.10.2.2 BIDIRECTIONALS

CECC	-	Cache ECC bus.
MECC	-	Memory ECC bus.

12.10.2.3 OUTPUTS

RDATA	-	cRrected DATA.
RDATAPAR	-	RDATA PARity.
SINGLERR	-	SiNGLe ERror.
SYND	-	SYNDrome.
SDO	-	The output of the maintenance shift chain.

12.10.3 DETAILED DESCRIPTION

The error detection/correction mechanism is based on a modified SEC-DED-4ED Hamming code. Besides single error correction and double error detection, this code is also capable of detecting all triple and quaduple errors within a single digit.

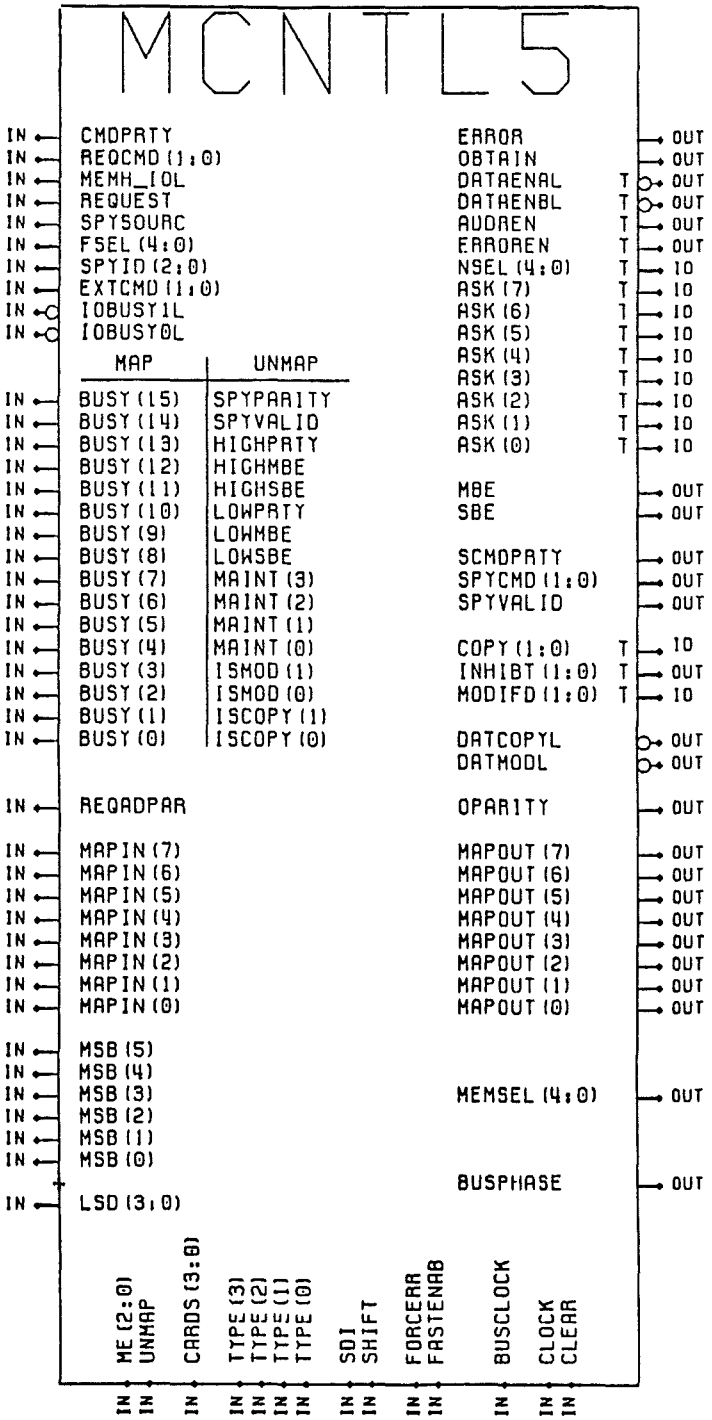
The ECC generation logic is used by both the error detect/correct logic and the write operations. After a write operation, the data is no longer the same, therfore the ECC should be generated for the new data.

The corrected data to be sent to rotater/concatenator are protected by parity. The 40 bit data are processed by 4 ROTCAT arrays, each of these array will generate a partial parity on the 10 bit data it processes. These 4 parity bits and the parity generated in MCECC2 are checked in the 5 bit parity tree in the RAMARR array.

12.10.4 SHIFT CHAIN DEFINITION

MEO	-
MEI	-
CECC(7:0)	-
ECCL(7:0)	-
DATA(39:0)	-

12.11 MCNTL5


MCNTL5
IN CMDPRTY
IN REQCMD(1:0)
IN MEMH_IOL
IN REQUEST
IN SPYSOURC
IN FSEL(4:0)
IN SPYID(2:0)
IN EXTCMD(1:0)
IN IOBUSY1L
IN IOBUSY0L
MAP | UNMAP
IN BUSY(15) | SPYPARITY
IN BUSY(14) | SPYVALID
IN BUSY(13) | HIGHPRTY
IN BUSY(12) | HIGHMBE
IN BUSY(11) | HIGHSBE
IN BUSY(10) | LOWPRTY
IN BUSY(9) | LOWMBE
IN BUSY(8) | LOWSBE
IN BUSY(7) | MAINT(3)
IN BUSY(6) | MAINT(2)
IN BUSY(5) | MAINT(1)
IN BUSY(4) | MAINT(0)
IN BUSY(3) | ISMOD(1)
IN BUSY(2) | ISMOD(0)
IN BUSY(1) | ISCOPY(1)
IN BUSY(0) | ISCOPY(0)
IN REQADPAR
IN MAPIN(7)
IN MAPIN(6)
IN MAPIN(5)
IN MAPIN(4)
IN MAPIN(3)
IN MAPIN(2)
IN MAPIN(1)
IN MAPIN(0)
IN MSB(5)
IN MSB(4)
IN MSB(3)
IN MSB(2)
IN MSB(1)
IN MSB(0)
IN LSD(3:0)
ERROR OUT
OBTAIN OUT
DATAENAL T OUT
DATAENBL T OUT
AUDAEN T OUT
ERRDAEN T OUT
NSEL(4:0) T IO
ASK(7) T IO
ASK(6) T IO
ASK(5) T IO
ASK(4) T IO
ASK(3) T IO
ASK(2) T IO
ASK(1) T IO
ASK(0) T IO
MBE OUT
SBE OUT
SCMDPRTY OUT
SPYCMD(1:0) OUT
SPYVALID OUT
COPY(1:0) T IO
INHIBT(1:0) T OUT
MODIFD(1:0) T IO
DATCOPYL OUT
DATMODL OUT
OPARITY OUT
MAPOUT(7) OUT
MAPOUT(6) OUT
MAPOUT(5) OUT
MAPOUT(4) OUT
MAPOUT(3) OUT
MAPOUT(2) OUT
MAPOUT(1) OUT
MAPOUT(0) OUT
MEMSEL(4:0) OUT
BUSPHASE OUT
IN ME(2:0)
IN UNMAP
IN CARDS(3:0)
IN TYPE(3)
IN TYPE(2)
IN TYPE(1)
IN TYPE(0)
IN SDI
IN SHIFT
IN FORCERR
IN FASTENAB
IN BUSCLOCK
IN CLOCK
IN CLEAR

12.11.1 FUNCTION OVERVIEW

The MCNTL5 array has 3 primary functions:

1) Control memory-bus arbitration and inter-cache communication.

2) Translate logical memory address to and from Memory Data Card physical address.

3) Detect and record errors that occur: on the backplane, in the ECCXCV array, and in the MCNTL5 array.

The MCNTL5 array interfaces memory bus requestors to main memory. The MCNTL5 array maintains the status of memory data cards and IOMC's. It arbitrates control of the main memory bus between all requestors. The MCNTL5 array handles the interface between other caches in the V500 mulitple-processor system.

The address translation operates in two distinct modes, map and unmap. The mode is selected by an external jumper, so a MCNTL5 array is either a MAPPER or an UNMAPPER depending on the level of the UNMAP input pin. In the MAPPER mode, the MCNTL5 translates a 25-bit absolute binary block address into a 5-bit module select field and a 21-bit physical address. The UNMAPPER mode reverses this mapping for the cache spy mechanism.

The MCNTL5 array collects the SBE and MBE signals from the ECCXCV arrays and generates the error signals to cache control.

12.11.2 INTERFACES

12.11.2.1 INPUTS

CMDPRTY
: The parity of the command from the cache control section. This parity bit is calculated such that the overall parity of the CMDPRTY, REQCMD, MEMH-IOL, and REQUEST lines is even.

REQCMD(1:0)
: The memory command of the request.

MEMH-IOL
: The request is for memory when high; I/O when low.

REQUEST
: This request is valid, ask for the main memory bus.

SPYSOURCE
: This input is driven by the cache in a MCACM, or by the I/O read logic in an IOMC. It activates the data and error enables. This allows a MCACM or an IOMC to respond like a Memory Data Card.

FSEL(4:0)
: One of the fast, unidirectional select buses. This bus is received from the far cabinet. It contains a copy of the module select field from the main memory address and command bus. see also, NSEL(4:0).

IOBUSY0L & IOBUSY1L
: Signals from the IOMC's. IOBUSY0L is low when IOMC-0 is busy; likewise with IOBUSY1L for IOMC-1. When either IOMC is available, a memory mapped I/O operation is allowed. When both IOMC's are busy, then memory mapped I/O operations must wait. If a system only has one IOMC then IOBUSY1L will be terminated low, therefore appearing busy.

12.11.2.1 INPUTS (Continued)

BUSY(15:0)
When the UNMAP input is low, these inputs are the status signals from each of the memory data cards. When a busy line is active, that memory card is unable to accept a command. This could be due to either an internal refresh operation, or a read-modify-write operation in progress. Note: the MCNTL5 will ignore busy on the write that follows a read-modify-write request.
When the UNMAP input is high, these inputs are used by other functions on the MCNTL5 array. Specifically:

BUSY(15) == SPYPARITY
Partial parity of the spy interface signals from the mapper to the unmapper array. The final spy interface parity to the cache section is calculated in the UNMAPPER.

BUSY(14) == SPYVALID
This qualifies the MBE and SBE error signals. These signals are only enabled when a valid spy command is available from the memory address bus.

BUSY(13) == HIGHPRTY
BUSY(12) == HIGHMBE
BUSY(11) == HIGHSBE
BUSY(10) == LOWPRTY
BUSY(9) == LOWMBE
BUSY(8) == LOWSBE
The error signals from the ECCXCV arrays are only valid over one half (high or low) of the memory address bus. The UNMAPPER decodes these signals and reports the errors.

BUSY(7) == MAINT(3)
BUSY(6) == MAINT(2)
BUSY(5) == MAINT(1)
BUSY(4) == MAINT(0)
These inputs are not committed by the internal logic of the UNMAPPER; they are used to latch external signals from the MCACM for maintenance purposes.

12.11.2.1 INPUTS (Continued)

BUSY(3:2) == ISMOD(1:0)
Interface from the cache section. These duplicate-and-compare signals cause the MODIFD(1:0) outputs to be driven; signaling that spy hit data being returned on the memory data bus is modified.

BUSY(1:0) == ISCOPY(1:0)
Interface from the cache section. These duplicate-and-compare signals cause the COPY(1:0) outputs to be driven; signaling that the read data from memory is not private because copies are already cached in the system.

REQADPRTY
This input is even parity over the 25-bit absolute binary address. Not all 25 bits of this address are received by the MCNTL5 array; therefore, no checking is done. The OPARITY output corresponds to this input parity and the change in parity done by MCNTL5. Note: this input is not used in the UNMAPPER mode.

MAPIN(8:0)
When the UNMAP input is low, these inputs are ADDRESS(24:19) & ADDRESS(1:0). Note: the ADDRESS here is the absolute binary block address.
When the UNMAP input is high, these inputs are SPY(17) & SPY(18) & SPY(19) & SPY(20) & SPY(21) & FALSE & FALSE & FALSE. Note: the SPY here is the corrected output of the ECCXCV arrays; this is the most significant bits of the memory data card block address.

MSB(4:0) - "most significant bits"
When UNMAP is low, these inputs are ADDRESS(24:20). The most significant bits of the address are used to determine which memory data card within the interleaved groups to address.
When UNMAP is high, these inputs are: FALSE & FALSE & SPY(25:23).

12.11.2.1 INPUTS (Continued)

LSD(3:0) - "least significant digit"
When the UNMAP input is low, these inputs are the least-significant digit of the address from the AWBUS. This digit is not base-added, converted to binary, or mapped, it is only used to provide the address of the I/O hardware register. It is mapped into the module select field of the memory address bus during I/O read/write operations.
When the UNMAP input is high, this input is SPY(25:22). This input is only used for the parity prediction logic.

ME(2:0)
This value is received from backplane straps. It specifies the logical address of the requestor, which determines the priority for bus arbitration.

UNMAP
A jumper: when low, this array maps address; when high, this array unmaps addresses.

BOARDS(3:0)
The number of boards which are installed in the system. This is received from the backplane. This value is sourced from one, or more, memory data cards. Note: the value of this field is one less than the number of Memory Data Cards in the system.

TYPE(3:0)
A 1-out-of-4 coded field which specifies the type, and therefore size, of the Memory Data Cards in the backplane. This field has four valid values, which are:

0001 - MDC's are half-populated with 256k RAM's.
0010 - MDC's are fully-populated with 256k RAM's.
0100 - MDC's are half-populated with 1M RAM's.
1000 - MDC's are fully-populated with 1M RAM's.

SDI
The serial shift chain input. Note: the output is from the OBTAIN pin.

12.11.2.1 INPUTS (Continued)

SHIFT
Serial shift chain enable signal from the clock and maintenance chip.

FORCERR
Checker-checking enable signal. This input causes the MCNTL5 array to signal a hard error with the ERROR output.

FASTENAB
The strap input causes the DATAENAL and DATAENBL outputs to be active for only 1 clock. This is used in conjunction with the MDECC array. This array has an internal flip-flop to hold its outputs valid for the 2-clock memory bus time; yielding a faster bus recovery time.

BUSCLOCK
This signals is generated by the IOMC, it specifies which phase of the 2-clock memory backplane bus is currently in effect. This is not a clock: it must be registered before being used by any module connected to the memory backplane.

CLOCK
The normal system clock.

CLEAR
Maintenance reset signal. This is an asynchronous clear.

12.11.2.2 BIDIRECTIONALS

ASK(7:0)
Bus request signals. Each requestor has a unique bus address which is determined by backplane straps which are received as ME(2:0). The highest numbered requestor, who is asking for the bus, wins the prioritization. The 8 possible requestors are:

7 - IOMC2 in cabinet 2 - highest priority
6 - IOMC2 in cabinet 1 .
5 - IOMC1 in cabinet 2 .
4 - IOMC1 in cabinet 1 .
3 - MCACM 3 in cabinet 2 .
2 - MCACM 2 in cabinet 1 .
1 - MCACM 1 in cabinet 2 .
0 - MCACM 0 in cabinet 1 - lowest priority

COPY(1:0)
This duplicated bidirectional signal is part of the interface between MCACM's. It signals the cache that the data being received is also contained in another MCACM; therefore, it is read-only data. This signal is duplicated for fault detection.

MODIFD(1:0)
This duplicated bidirectional signal is part of the interface between MCACM's. It signals the cache that the data being received is modified. This signal is duplicated for fault detection.

12.11.2.3 OUTPUTS

ERROR

A hardware error has been detected. Hardware errors in this array are serious enough to cause the system to halt. There are a number of error conditions that cause ERROR. In the MAPPER array they are:

BACKPLANE CONTROL ERROR - latched
The duplicate-and-compare inter-MCACM signals didn't compare.

MEMORY BUS ARBITRATION ERROR - latched
BACKPLANE CONTROL ERROR
The ID of the memory requestor driving the
The ID of the memory requestor driving the main memory bus is not the winner of the previous cycles arbitration.

COMMAND PARITY ERROR - not latched
There is a parity error on the command and request lines from the cache.

MEMORY CONFIGURATION ERROR - latched
There is an error in the 1-out-of-4 coded Memory Data Card type field.

In the UNMAPPER array these error conditions are:

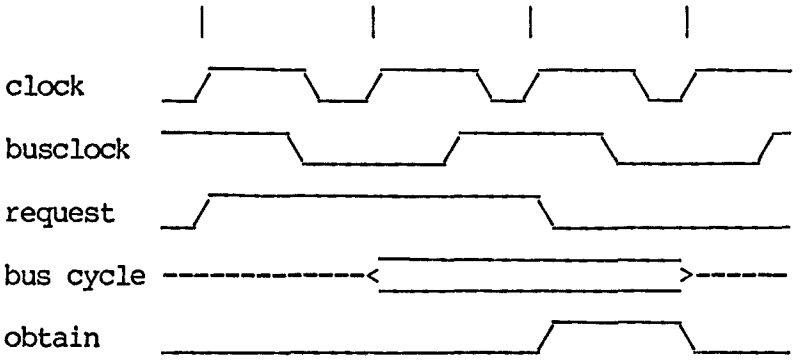
MEMORY CONFIGURATION ERROR - latched
There is an error in the 1-out-of-4 coded Memory Data Card type field.

ECC TRANSCEIVER ERROR - not latched
The MCNTL5 array has detected an error in the encoding of the MBE and SBE signals from the ECCXCV arrays.

12.11.2.3 OUTPUTS (Continued)

OBTAIN

This is a handshake signal to cache. It signals the granting of a memory request. OBTAIN goes active at the second system clock of a bus cycle. Note: the address and command information to the MCNTL5 array must remain stable until the OBTAIN signal.



DATENAL & DATENBL - active low

Backplane data driver enables. These outputs are duplicated because of loading considerations. This output is enabled by the granting of a write request or SPYSOURCE.

When FASTENAB is low, the DATAENAL and DATAENBL signals are active for the entire 2-clock memory backplane cycle. When FASTENAB is high, they are only active during the first clock of the cycle.

ADRREN - active high

Backplane address and command bus driver enable. This output is enabled by the granting of a memory or I/O request.

ERROREN - active high

Backplane error and data source bus driver enable. This output is enabled by the SPYSOURCE input.

MBE

The spy has detected a multiple bit error on the main memory address and command bus. This error is sent to the error recorder instead of being signaled as an error here.

12.11.2.3 OUTPUTS (Continued)

SBE
The spy has detected a single bit error on the main memory address and command bus.

SCMDPRTY
Spy command line parity bit. This parity is calculated such that the parity over the SCMDPRTY, SPYCMD(1:0), and SPYVALID lines is even.

SPYCMD(1:0)
The encoded spy command.

SPYVALID
This signal, when high, qualifies the SPYCMD. When low, SPYCMD should be ignored.

DATCOPY
This interface signal informs the local cache that COPY(1:0) was high for the received data.

INHIBT(1:0)
This signal is sent a clock before modified data. It disables the memory data card output drivers and switches the memory buffer card direction. This allows the sourcing cache to drive data instead of the memory data card.

DATMOD
The interface signal informs the local cache that MODFD(1:0) was high for the received data.

12.11.2.3 OUTPUTS (Continued)

OPARITY

This dual-mode output is the unmapped parity when the UNMAP input is high and the mapped parity when the UNMAP input is low.

The unmapped parity is calculated such that the parity over OPARITY and the unmapped absolute address is even. Note: some of the address bits are not changed by this array. The MCNTL5 array predicts what the change of the receive parity will be by the unmapping function. Therefore, this parity bit also protects the UNMAPPER.

The mapped parity bit is calculated such that the parity over the main memory address and command bus, sent to the ECCXCV arrays and OPARITY is even. This parity is predicted based on the inputs from the cache section. Therefore, this parity bit also protects the MAPPER.

MAPOUT(8:0)

The mapped absolute binary memory data card address. This address has the interleaving and module select information removed. It is calculated from the input binary address and the number and type of memory data cards in the system.

MEMSEL(4:0)

The module select field to the ECCXCV arrays. This value is calculated from the input binary address, MAPIN(8:0) and the number and type of memory data cards in the system. The most significant bit of this field specifies memory when high, or I/O when low.

12.11.3 DETAILED DESCRIPTION

12.11.3.1 MEMORY MAPPER

The memory mapping algorithm for the V5 System is different than previous machines. This algorithm allows plugging an arbitrary number of memory data cards into the backplane.

When a simple number of MDC's (such as: 1,2, or 4) are online, the memory is fully interleaved. The data in a fully inter- leaved memory is addressed such that the module select is the least-significant portion of the address.

When a more bizarre number of MDC's (such as:3,5,7-16) are online, the memory interleaving is slightly different. For example, with five cards, the first four cards are fully interleaved within the low addressing range. These four cards are addressed just like the simple case above. The fifth card is also interleaved, but only in the upper address range. Therefore, the addresses in a five MDC configuration go upwards through the first four cards and finally into the fifth card.

Note, the interleaving of a single card is the degenerate case: no address mapping is done.

12.11.3.2 MEMORY UNMAPPER

The memory unmapper basically undoes the mapping of the mapper. This was simplified by choosing the mapping algorithm so that the mapping and unmapping functions are nearly reciprocal. That is, applying the map function once, generates the mapper; and reapplying the map function, almost recovers the original address. The slight difference in the algorithms requires the UNMAP input which specifies which variant that this array is handling. The UNMAP input is designed to be a strap; so that an array is either a mapper or an unmapper, not both.

12.11.3.3 MEMORY DATA CARD STATUS LOGIC

Sixteen flip-flops are dedicated to maintaining the busy status of each of the memory data cards. An additional pair of flip-flops contains the busy status of the IOMC's. The state of these flip-flops is checked, when REQUEST is high. If the desired module is busy, or both IOMC's for an I/O operation, then the request is denied. The exception to this rule is: an IOMC which initiated a read-modify-write cycle to an MDC will ignore its busy state. This is handled inside the MCNTL5 array with the RMW flip-flop.

A memory module, and the IOMC's which fake memory cycles, will remain busy for two bus cycles after it is requested. Therefore, a module may be busy, even if its busy line is inactive. This is determined by latching the module select (a portion of the memory address bus) into a two bus-clock pipeline. If the desired module is being requested now, or was requested on the last cycle, then the request is denied.

If a requested MDC is not busy - and is not being requested now - and was not requested last cycle, then memory bus access arbitration is started.

12.11.3.4 MEMORY BUS ACCESS ARBITRATION

Arbitration for the main memory bus is accomplished with the eight bi-directional ASK(7:0) lines. These lines are implemented as fully bi-directional lines; however, the ME(2:0) input determines which line is driven, the others are only received.

All of the requestors who have passed the availability test, detailed in Section 1.1.3.3, drive their ASK(7:0) line. This particular line is determined by backplane jumpers which are unique to the requestor slot. The highest numbered line which is active is granted the next bus cycle.

The access obtained signal is OBTAIN. The cache control and IOMC control use this signal to determine when they have been granted a memory cycle, and can change their addresses to ask for another one.

12.11.3.5 INTER-CACHE WRITE-BACK ALGORITHM INTERFACE

Each of the caches in a distributed write-back algorithm must exchange certain information with the other caches. This information consists of: copy detection status, private status, and modified status. The cache communicates this information with the INHIBT(1:0), COPY(1:0), and MODIFD(1:0) lines.

These lines are only duplicated for fault detection.

12.11.3.6 FAULT DETECTION

The fault detection of the MCNTL5 array is broken into several distinct portions because of the variety of functions. These portions are, the memory mapper and unmapper, the bus arbitration logic, the inter-cache status interface, input parity on the command field, and checker checking.

The memory mapper and unmapper are protected by parity prediction. The input addresses have parity. The mapped, or unmapped outputs, have their parity predicted based on the change in parity caused by the mapping function. This parity prediction is semi-independent of the mapping function, so the fault detection is good.

The bus arbitration logic is protected by distributed duplicate and compare. Each of the bus requestors monitors the ASK(7:0) lines and determines which requestor should have won the arbitration. Then, each requestor checks the source ID field in the memory address and command bus to see that the proper module actualy did win.

The inter-cache status interface is protected by duplication and comparison.

The input command field, including the MEMH-IOL and the REQUEST signal, is protected by parity.

A special input is available which will force the ERROR output active. It is lamented, that this signal does not activate the checkers any deeper than this.

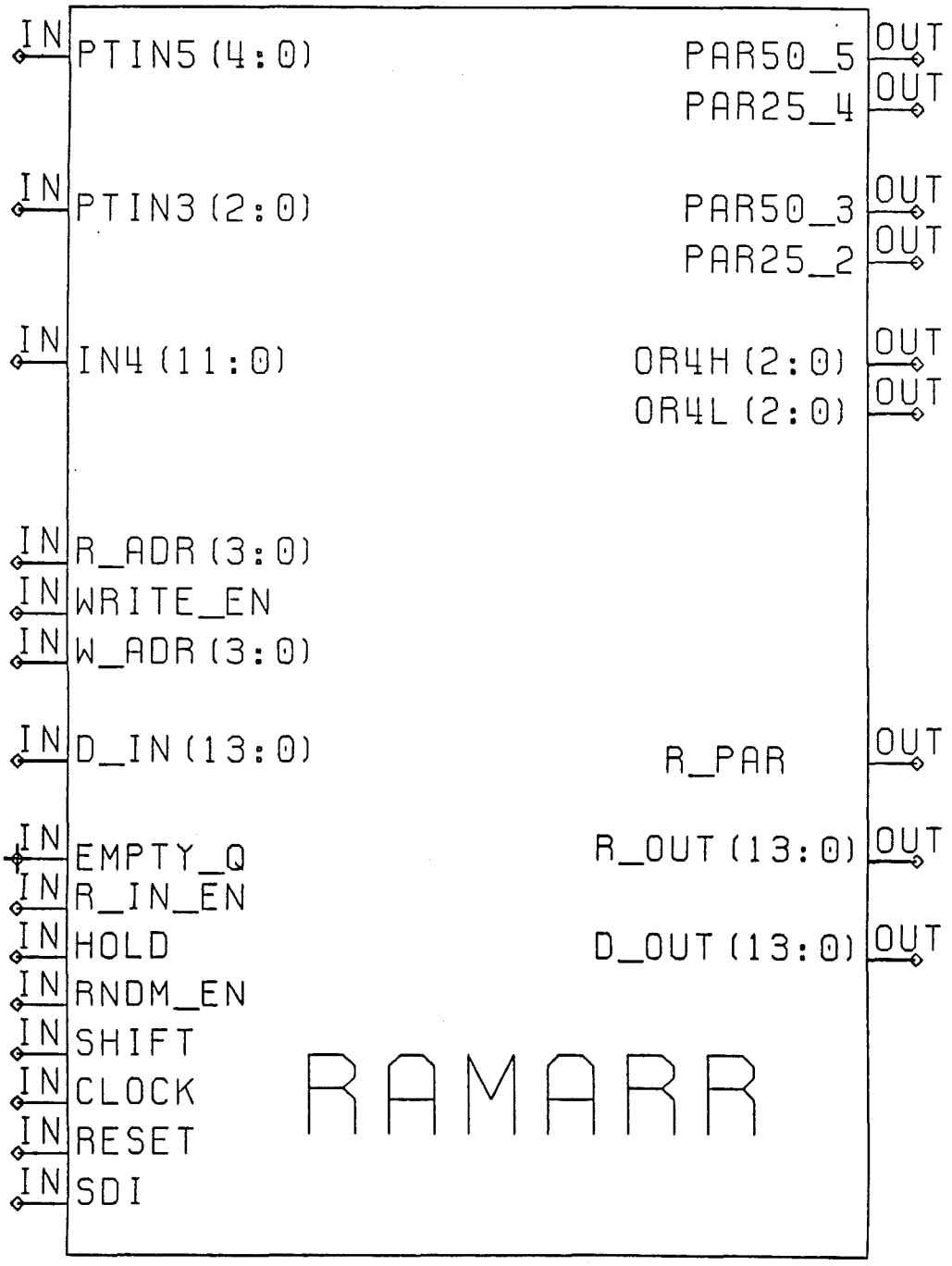
12.11.3.7 BOARD TEST FACILITIES

A TESTMODE flip/flop has been provided which disables all backplane drivers and driver-enables. These signals are also disabled during shifting. This allows one MCACM to be setup in a test mode without disturbing the other processors or memory accesses in a multi-processor system.

12.11.4 SHIFT CHAIN DEFINITION

OBTAIN	- access has been obtained and transmitted
AWINNER	- some requestor won bus arbitration
WINID(3:0)	- ID of requestor who won bus arbittration
BPCTLERR	- error detected on inter-cache interface
MODIFIED	- send modified on backplane
DATACOPY	- data on backplane was copy
COPY	- send copy on backplane
RMW	- last command was RMW, so ignore busy
BUSUSED	- data will be returning on next bus cycle
IOMCBZ(1:0)	- registered IOMC busy lines
BUSY(15:0)	- registered busy lines, also rnd latches
SEL-T(4:0)	- module requested on last bus cycle
SPYISRD	- spy command is a memory read
TESTMODE	- disables backplane output drivers
BUSCLOCK	- half-clock-frequency bus phase time receiver

12.12 RAMARR


IN PTIN5(4:0)
IN PTIN3(2:0)
IN IN4(11:0)
IN R_ADR(3:0)
IN WRITE_EN
IN W_ADR(3:0)
IN D_IN(13:0)
IN EMPTY_Q
IN R_IN_EN
IN HOLD
IN RNDM_EN
IN SHIFT
IN CLOCK
IN RESET
IN SDI
PAR50_5 OUT
PAR25_4 OUT
PAR50_3 OUT
PAR25_2 OUT
OR4H(2:0) OUT
OR4L(2:0) OUT
R_PAR OUT
R_OUT(13:0) OUT
D_OUT(13:0) OUT
RAMARR

12.12.1 FUNCTION OVERVIEW

This array works like a 16x14 RAM. The Input Queue uses eight of these arrays.

12.12.2 INTERFACES

12.12.2.1 INPUTS

R ADR(3:0)	- read address.
WRITE EN	- enables the writing of the latch array.
W ADR(3*0)	- write address.
D IN(13:0)	- normal input data to the latch array.
R IN EN	- enable the input from register to latch array.
EMPTY Q	- D IN bypass the latch array and into the register
HOLD	- the value of the register hold for one clock.
SDI	- serial data input of the shift chain.
RANDOM EN	- enables the shifting of initialization data into the internal register.
SHIFT	- maintenance shift.
CLOCK	- system clock.
RESET	- maintenance reset.
IN4(11:0)	- input to the 4-input OR/NOR gates.
PTIN5(4:0)	- input to the 5-input parity tree.
PTIN3(2:0)	- input to the 3-input parity tree.

12.12.2.2 OUTPUTS

D OUT(13:0)	- data read from the latch array.
R OUT(13:0)	- output data of the internal register.
R PAR	- parity of R OUT.
OR4H(2:0)	- positive level of the OR/NOR gates.
OR4L(2:0)	- negative level of the OR/NOR gates.
PAR50 5	- parity of PTIN5(4:0), 50 ohm driver.
PAR25 4	- parity of PTIN5(3:0), 25 ohm driver.
PAR50 3	- parity of PTIN3(2:0), 50ohm driver.
PAR25 2	- parity of PTIN3(1:0), 25 ohm driver.

12.12.3 DETAILED DESCRIPTION

The data from AWBUS may be written into the 16x14 latch array then to the internal 14-bit register or bypass the latch array and be written into the register directly. The input to the latch array may come from the input pins or the register. Parity generation for the data in the register is also included for error protection.

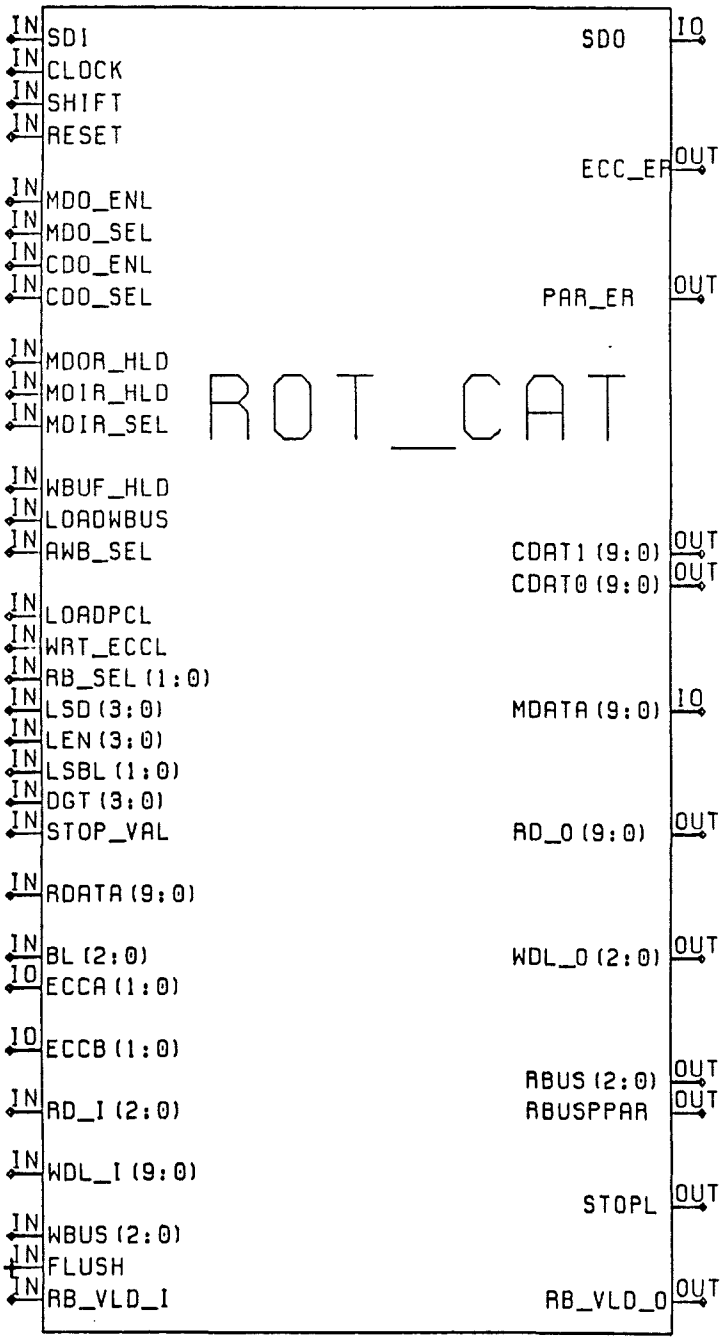
There are also four OR/NOR gates and two parity trees in this array for various applications on the MCACM address card.

12.12.4 SHIFT CHAIN DEFINITION

REGISTER(13:0) -

This register receives the output of the bypass mux. It loads from either the D-IN(13:0) inputs or the the output of the internal 16x14 RAM structure. This register is also used to write the RAM structure

12.13 ROTCAT


ROT_CAT
IN SDI
IN CLOCK
IN SHIFT
IN RESET
IN MDO_ENL
IN MDO_SEL
IN CDO_ENL
IN CDO_SEL
IN MDOR_HLD
IN MDIR_HLD
IN MDIR_SEL
IN WBUF_HLD
IN LOADWBUS
IN AWB_SEL
IN LOADPCL
IN WRT_ECCL
IN RB_SEL(1:0)
IN LSD(3:0)
IN LEN(3:0)
IN LSBL(1:0)
IN DGT(3:0)
IN STOP_VAL
IN RDATA(9:0)
IN BL(2:0)
IO ECCA(1:0)
IO ECCB(1:0)
IN RD_I(2:0)
IN WDL_I(9:0)
IN WBUS(2:0)
IN FLUSH
IN RB_VLD_I
SDO IO
ECC_ER OUT
PAR_ER OUT
CDAT1(9:0) OUT
CDAT0(9:0) OUT
MDATA(9:0) IO
RD_O(9:0) OUT
WDL_O(2:0) OUT
RBUS(2:0) OUT
RBUSPPAR OUT
STOPL OUT
RB_VLD_O OUT

12.13.1 FUNCTION OVERVIEW

ROTCAT, the primary data path of MCACM, is the interface among CACHE, Memory Bus, RBUS and Write Data Queue. It contains partial word processing logic as required by read/write operations. It also has the registers that drive/accepts the Main memory data bus and register that drives the RBUS. A pipeline for the data from AW Queue and the logic used to monitor the data written into cache are also included in this array.

12.13.2 ~ INTERFACES

12.13.2.1 INPUTS

MDO-ENL	-	Memory Data bus Output ENable Low.
MDO-SEL	-	Memory Data bus Output SELect.
CDO-ENL	-	Cache Data bus Output ENable Low.
CDO-SEL	-	Cache Data bus Output SELect.
MDIR-HLD	-	Memory Data Input Register HoLD.
MDOR-HLD	-	Memory Data Output Register HoLD.
RDATA(39:0)	-	coRrcted DATA from MCECC.
WDL-I(9:0)	-	Write Data Low, Input.
STOP-VAL	-	STOP VALue used by stop logic.
DGT(3:0)	-	DiGiT location to be monitored by stop logic.
CDAT1(9:0)	-	Cache DATa bus block 1.
CDAT0(9:0)	-	Cache DATa bus block 0.
LEN(3:0)	-	LENgth of the read/write data.
LSD(3:0)	-	Least Significant Digit of the address.
LSBL(1)	-	0: this chip is for the data at (address + 1) 1: it is not.
LSBL(0)	-	0: this chip is for the data at (address). 1: it is not.
AWB-SEL	-	Write data pipeline output SELect.
WBUS(2:0)	-	Write data from aw queue.
LOADWBUS	-	LOAD data on WBUS into the register.
WBUF-HLD	-	Write data BUFfer register HoLD.
LOADPCL	-	LOAD internal PC register from wbus, Low.
BL(2:0)	-	Base/Limit or error address/error report.
WRT-ECCL	-	WRiTe ECC, Low.
RD-I	-	Rotated Data, Input.
RB-SEL	-	RBus input SELect.
CLOCK	-	
SHIFT	-	
RESET	-	
SDI	-	

12.13.2.2 OUTPUTS

SDO	-	
PAR-ER	-	partial parity of RDATA.
RD-O(9:0)	-	Rotated Data, Output.
WDL-O(2:0)	-	Write Data, Low, Output.
RBUS(2:0)	-	Read BUS.
RBUSPPAR	-	RBUS Partial PARity.
ECC-ER	-	cache ECC bus ERror.

12.13.2.3 BIDIRECTIONALS

MDATA(9:0)	-	Memory DATA bus.
ECCA(1:0)	-	cache ECC bus block 0.
ECCB(1:0)	-	cache ECC bus block 1.

12.13.3 DETAILED DESCRIPTION

12.13.3.1 DATA PATH

For a write-back operation, the partial word processing logic is disabled by setting starting digit and length to be zero. For a read operation, The data to be read are left-justified before sent to RBUS registers. For write operation, no rotation is performed, the data in the field as specified by digit address and length are replaced by write data at the same digit location.

The registers that drive the memory data bus can hold the data for one more clock as required by the 2-clock memory cycle. There are four potential inputs to the RBUS register, PC register, Base/Limit or error report/address from error recorder, Cache ECC Bus, or the rotated (left-justified) data. The pipeline structrue for the write data from AW Queue is the same as the one in Cache control array, so that the synchronization of the write data and the tag/command can be obtained.

12.13.3.2 STOP LOGIC

The stop logic in this array is used to monitor the data written into CACHE either from memory or from AW Queue by a write operation. If the specified digit location is written wirh the specified value, a stop signal wil be sent to the STOP array.

12.13.4 SHIFT CHAIN DEFINITION

MDI (0:7) -
MDO (0:7) -
RBUS-VLD -
RBUS-REG (0:2) -
ECC-REG1 (0:1) -
ECC-REG0 (0:1) -
PC (0:2) -
AW-PIPE-REG0(0:2) -
AW-PIPE-REG1(0:2) -
AW-PIPE-REG2(0:2) -

13 TERMS AND DEFINITIONS

AWBUS	The address and write bus from the processor modules to the memory control and cache module. This bus is unidirectional.
COP	See XM/Coprocessor.
COPY	This term is used interchangeably with "read-only". Copy is used to designate a block of cache that may be read but not written. There may be many copies of this data throughout the system. See Section 7 for a detailed description of the algorithms.
DRAM	Dynamic random access memory, commonly available in 16, 64, and 256 Kbit varieties, a cheap, medium speed, volatile storage device.
DTM	The data transfer module. The DTM is the part of the I/O subsystem which interfaces with peripheral devices and buffers the data transfers to and from them.
EWBUSREQXMDP	The execute module's AWBUS request signal.
EWBUSREQXMXP	The XM/Coprocessor's AWBUS request signal.
FETCH	The fetch module, including both the instruction fetch unit and the operand fetch unit.
FWBUSREQIF$P	The instruction fetch's AWBUS request signal.
FWBUSREQOF$P	The operand fetch's AWBUS request signal.
IF	The instruction fetch unit.

13 TERMS AND DEFINITIONS (Continued)

IOMC	The I/O memory concentrator.
IOT	The I/O Translator.
MCACM	The memory control and cache module.
MDC	The memory data card.
MP	The maintenance processor.
MQUEUEFULL$P	The MCACM input queue full signal.
MRBUSERR$P(4:0)	The error information field of the RBUS.
MRBUSPARITYP	The parity of the RBUS except the tag field which is protected separately.
MRBUSTAGPARP	The parity of the tag field of the RBUS.
MRBUSTAG$P(6:0)	The tag field of the RBUS.
MRBUSVALID$P	The RBUS data valid signal.
MRBUS$$$P(39:0)	The data field of the RBUS.
OF	The operand fetch unit.
"Operand Queue"	The portion of the execute module's input queue which is dedicated to the pre-fetched data requested by the operand fetch unit. The execute module also contains an "RBUS queue" which is dedicated to data which it has requested for its own use.
"Operand Read"	A "Read Request" that requires 2 main memory accesses to acquire the requested data. Due to the organization of main memory, any request that spans an absolute mod-40 address boundary is an operand read. Note: there are no operand writes.

13 TERMS AND DEFINITIONS (Continued)

PC	The program counter; this is the instruction address.
PRIVATE	This term is used interchangeably with "writeable" and "no-copies". Private is used to designate a block of cache that may be written to by a local processor request. The nature of the cache consistency algorithm is such that only 1 "copy" of this data is valid in the entire system. See Section 7 for a detailed description of the algorithms.
RAM	An acronym for random access memory, it is generally used as a generic term for memory chips and or arrays of more than around 256 bits.
RBUS	The read data bus from the memory control and cache module.
"RBUS Queue"	The portion of the execute module's input queue which is dedicated to the data which it has requested, as opposed to the data that was pre-fetched by the operand fetch module.
"Read Request"	The term "read request" means any read operation: read data, read-with-lock, read base, read limit, read uncorrected, read ECC, or read PC.
SMC	The system maintenance controller. The SMC is responsible for controlling clocks and shifting data and maintenance chains in the V500 processor.
SRAM	Static random access memory. These RAMs are generally smaller, faster, and more expensive than their cousins the DRAMS. Static RAMS tend to have a higher failure rate than other components and require special attention for fault

detection and fault tolerance.

"Write Request"	The term "write request" means any write operation: write data, write base, write limit, write I/O, write ECC, and write PC. The term does not include the write-back operation.
XADRBUSPRTYP	The parity of the address and base indicant fields of the AWBUS.
XADRBUS$P(3:0)	The address field of the AWBUS.
XADRBUS$P(39:36)	The base indicant field of the AWBUS.
XM	The execute module.
XM/COPROCESSOR	An auxiliary arithmetic processor that connects to XM, Fetch and the memory control and cache module. It speeds up the operation of some arithmetic operations by executing them in parallel with non-arithemtic operations. AWBUS.
XWBSRQID$P(2:0)	The bus requestor ID field of the AWBUS.
XWBUSCMD$P(4:0)	The command field of the AWBUS.
XWBUSLEN$P(3:0)	The length field of the AWBUS.
XWBUSPARITYP	The parity of the command and length fields of the
XWBUSTAGPARP	The parity of the tag field of the AWBUS.

13 TERMS AND DEFINITIONS (Continued)

XWBUSTAG$P(6:0)	The tag field of the AWBUS.
XWRTBUSPRTYP	The parity of the write data field of the AWBUS.
XWRTBUS$$$P(39:0)	The write data field of the AWBUS.

14 CHAIN DEFINITIONS

14.1 MA

```
CHAIN MA;                          % MCACM-A data chain

%*************************************************************************%
% chip: BUFFER2          location: P2B0                                   %
% clear: E15,   shift: H15,   clock: C12,   sdi: J14,   sdo: H14          %
% function: BROKEN SIGNAL REGISTERED & AW BUS ARBITRATION ERROR TRAP      %
%*************************************************************************%

        PLA15DATA       8;      % READ/WRITE REGISTER FOR PLA15
        PLA15WE         1;      % WRITE ENABLE BIT FOR PLA15
        PLA15SEL        3;      % WRITE ADDRESS FOR PLA15
        TRAP15          1;      % 15h (AW BUS ARBITRATION ERROR) TRAP
        BROKEN          1;      % MCACM A BROKEN SIGNAL

%*************************************************************************%
% chip: MCNTL4           location: R6N0                                   %
% clear: P07,   shift: R05,   clock: C12,   sdi: P03,   sdo: H01          %
% function: MEMORY ADDRESS UNMAPPER AND SPY INTERFACE.                    %
%*************************************************************************%

        MC2OBT          1;      % OBTAIN F/F (NOT USED)
        MC2TRAFFIC      1;      % AWINNER F/F (NOT USED)
        MC2WINID        3;      % WINID F/F'S (NOT USED)
        MC2AERR         1;      %             (NOT USED)
        MC2SMWON        1;      %             (NOT USED)
        MC2WONID        3;      %             (NOT USED)
        MC2MREQ         1;      % REG'D REQUEST (NOT USED)

%   THESE TWO FIELDS CONTROL THE UNMAPPING ALGORITHM. THEY SHOULD BE
%   THE SAME AS THE MAPPER MCNTL FIELDS DURING NORMAL OPERATION
        UCARDS          4;      % NUMBER OF MEMORY DATA CARDS IN SYSTEM
        USIZE           2;      % TYPE OF MDC'S IN SYSTEM
        UCONFIGERR      1;      % UNMAPPER DETECTED ERROR IN XMCRDTYPE$P
        BPCTLERR        1;      % ERROR DETECTED IN INTERCACHE BP SIG'S
        DATAMOD         1;      % MODIFIED F/F
        MC2DRMOD        1;      % DRIVE MODIFIED F/F (NOT USED)
        DATACOP         1;      % DATACOPY F/F
        MC2DRCOP        1;      % DRIVE DATACOPY F/F (NOT USED)

        MC2RMW          1;      % RMW F/F (NOT USED)
        MC2NMINE        1;      % NOT MINE (NOT USED)
        MC2BUS          1;      % BUSUSED F/F (NOT USED)
        MC2IOBZ         2;      % IOMCBUSY F/F (NOT USED)
```

14.1 MA (Continued)

```
% THE FOLLOWING 16 BIT "BUSY" REGISTER IS USED TO MONITOR CERTAIN
% SIGNALS, SOME ARE PURELY COMBINATORIAL USED BY THE UNMAPPER
% CONFIGURATION, OTHERS ARE USED ONLY FOR MAINTENANCE. THIS REGISTER
% ONLY LOADS WHEN MABUSCLOCK2 = 0
        MC2BZ-15        1;      % SPYPARITY (FROM MAPPER)
        MC2BZ-14        1;      % SPYVALID (FROM MAPPER)
        MC2BZ-13        1;      % ECCXCVRPARU (FROM UPPER ECCXCV)
        MC2BZ-12        1;      % ECCXCVRMBEU   "     "     "
        MC2BZ-11        1;      % ECCXCVRSBEU   "     "     "
        MC2BZ-10        1;      % ECCXCVRPARL (FROM LOWER ECCXV)
        MC2BZ-09        1;      % ECCXCVRMBEL   "     "     "
        MC2BZ-08        1;      % ECCXCVRSBEL   "     "     "
        MC2BZ-07        1;      % MAINTENANCE READ OF LRURAMOUT
        MC2BZ-06        1;      % BUS-2V
        MC2BZ-05        1;      % MAINTENANCE READ OF INHLRURAMOUT
        MC2BZ-04        1;      % MAINTENANCE READ OF CACINHRAMOUT
        MC2BZ-03        1;      % INHIBIT(1)  FROM CACHE
        MC2BZ-02        1;      % INHIBIT(0)   "     "
        MC2BZ-01        1;      % COPY(1)      "     "
        MC2BZ-00        1;      % COPY(0)      "     "

        MC2SEL-T        5;      % SEL-T(4:0) (NOT USED)

        MC2SPYRD        1;      % SPY IS READ (NOT USED)
        MC2SRCSPY       1;      % SOURCE SPY (NOT USED)
        BUSCLOCK2       1;      % BUSCLOCK RECEIVER FOR MC2
        TESTMODE2       1;      % DISABLE OUTPUT DRIVERS ON MC2
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: ECCXCV          location:                                         %
% shift: R13,   clear: P11,   clock: C12,   sdi: R10,   sdo: N15          %
% function: UPPER HALF OF MEMORY ADDRESS BUS ECC TRANSCEIVER.             %
%*************************************************************************%

        RCVECC-3U       1;      % DUPLICATE OF RCVECC-3
        RCVECC-4        1;      % UPPER THREE BITS OF ECC RECEIVED
        RCVECC-5        1;      % FROM THE MEMORY ADDRESS BUS
        RCVECC-6        1;      % BACKPLANE.

        RCVADR-16       1;
        RCVADR-17       1;
        RCVADR-18       1;
        RCVADR-19       1;
        RCVADR-20       1;
        RCVADR-21       1;
        RCVADR-22       1;
        RCVADR-23       1;
        RCVADR-24       1;
        RCVADR-25       1;
        RCVADR-26       1;
        RCVADR-27       1;
        RCVADR-28       1;
        RCVADR-29       1;
        RCVADR-30       1;
        RCVADR-31       1;

        XMTECC-3U       1;      % NOT USED
        XMTECC-4        1;      % UPPER THREE BITS OF ECC
        XMTECC-5        1;      % TO BE TRANSMITTED ON TO
        XMTECC-6        1;      % THE MEMORY ADDRESS BACKPLANE
```

14.1 MA (Continued)

```
        ECCLPWU          1;      % GENERATE A LPW OF ECC'S FROM CACHE

        XMTADR-16        1;
        XMTADR-17        1;
        XMTADR-18        1;
        XMTADR-19        1;
        XMTADR-20        1;
        XMTADR-21        1;
        XMTADR-22        1;
        XMTADR-23        1;
        XMTADR-24        1;
        XMTADR-25        1;
        XMTADR-26        1;
        XMTADR-27        1;
        XMTADR-28        1;
        XMTADR-29        1;
        XMTADR-30        1;
        XMTADR-31        1;

%*************************************************************************%
% chip: ECCXCV          location:                                         %
% shift: R13,   clear: P11,   clock: C12,   sdi: R10,   sdo: N15          %
% function: LOWER HALF OF MEMORY ADDRESS BUS ECC TRANSCEIVER.             %
%*************************************************************************%

        RCVECC3:0        4;      % ECC RECEIVED FROM THE MEMORY ADDR. BU

        RCVADR15:0      16;      % DATA RECEIVED FROM THE MEMORY ADR. BU

        XMTECC3:0        4;      % ECC TO BE XMT'D TO MEMORY ADDR. BUS

        ECCLPWL          1;      % GENERATE LPW ON ECC'S FROM CACHE

        XMTADR15:0      16;      % ADDR. TO BE XMT'D TO MEMORY ADDR. BUS
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: MCNTL4          location: K4N0                                    %
% shift: R05,   clear: P07,   clock: C12,   sdi: P03,   sdo: H01          %
% function: MEMORY ADDRESS MAPPER AND MEMORY BUS ARBITRATION LOGIC        %
%*************************************************************************%

        OBTAIN          1;      % THIS MCACM HAS OBTAINED THE MEMORY BU
        TRAFFIC         1;      % SOMEONE WON THE MEMORY BUS ARBITRATIO
        WINID-2         1;      % THE ID OF THE MEMORY BUS WINNER BIT 2
        WINID-1         1;      %                                 BIT 1
        WINID-0         1;      %                                 BIT 0
        ARBERR          1;      %
        AWINNER         1;      %
        WONID           3;      %
        MREQ            1;      % REG'D MEMORY REQUEST SIGNAL


%   THESE TWO FIELDS CONTROL THE MAPPING ALGORITHM. THEY SHOULD BE
%   THE SAME AS THE UNMAPPER MCNTL FIELDS DURING NORMAL OPERATION
        MCARDS          4;      % NUMBER OF MEMORY DATA CARDS IN SYSTEM
        MSIZE           2;      % TYPE OF MDC'S IN SYSTEM
        MCONFIGERR      1;      %
        MC2BPERR        1;      % ERROR DETECTED IN ARBITRATION LOGIC
        MC2ISMOD        1;      % DATA ON BACKPLANE IS MODIFIED
        DRIVEMOD        1;      % DRIVE MODIFD ONTO INTER-CACHE BUS
        MC2ISCOP        1;      % DATA ON BACKPLANE IS COPY (READ ONLY)
        DRIVECOP        1;      % DRIVE DATCOPYL ONTO INTER-CACHE BUS

        RMW             1;      % READ-MODIFY-WRITE CYCLE IN PROGRESS
        NOTMINE         1;      %
        BUSREAD         1;      % MEMORY DATA BUS CONTAINS READ DATA
        IOMCBUSY        2;      % IOMC BUSY F/F'S
        MODBUSY        16;      % MEMORY BUSY F/F'S

        MODSEL-T        5;      % THE MODULE SELECT OF THE LAST CYCLE

        SPYISREAD       1;      % READ REQUEST ON BACKPLANE
        SOURCESPY       1;      % SOURCE THE DATA BUS ON A SPY HIT

        BUSCLOCK1       1;      % BUSCLOCK RECEIVER FOR MC2
        TESTMODE1       1;      % DISABLE BACKPLANE DRIVERS
```

14.1 MA (Continued)

```
%
% the following pins of CHADDR3 are driven by x-cell x291 f/f.
%  J1, K1, D15, C13, B2, M1, F15, L1, H1, G2, C1, D1, F2, P2, M2, N3
%
%*************************************************************************%
% chip: CHADDR3          location: I0I2                                   %
% shift: R11,   clear: N01,   clock: C12,   sdi: R09,   sdo: J01          %
% function: 5 OF 5 CACHE ADDRESS ARRAYS.                                  %
%*************************************************************************%

      DRAMHB-9          1;      % Data ram high address
      DRAMHB-4          1;      % Each CHADDR2 provides 2 bits
      DRAMHA-9          1;      % There are 2 copies
      DRAMHA-4          1;
      DRAMLB-9          1;      % Data ram low address
      DRAMLB-4          1;
      DRAMLA-9          1;
      DRAMLA-4          1;
      KYRMIN-14         1;      % 15 bits of key-ram input
      KYRMIN-13         1;      % Each CHADDR2 provides 3 bits
      KYRMIN-12         1;
      KYECIN-14         1;      % 15 bits of key-ram ecc input
      KYECIN-13         1;      % Each CHADDR2 provides 3 bits
      KYECIN-12         1;
      KADRAB-9          1;      % 10 bit of key-ram address input
      KADRAB-4          1;      % seperated into 2 blocks
                                % Each block has 2 copies
      KADRAA-9          1;      % NOTE : ACTIVE LOW
      KADRAA-4          1;
      KADRBB-9          1;      % Block-B of key-ram address input
      KADRBB-4          1;
      KADRBA-9          1;
      KADRBA-4          1;
      TRANLSB-1         1;      % Lsb field. In array #1-4 is
      TRANLSB-0         1;      % used as LSD and LEN field
      CHADR5ERR         1;      % Cache address array5 in error
      T2SPARE-1         1;      % not used
      T1SPARE-1         1;
      T3SPARE-1         1;
      T2TAG-4           1;      % Tag field
      T1TAG-4           1;
      T3TAG-4           1;
```

14.1 MA (Continued)

```
T2LSB-0          1;      % 2 bit lsb in processor pipeline,
T1LSB-0          1;      % In array #1-4 is used as lsd and len
T3LSB-0          1;
T2LSB-1          1;
T1LSB-1          1;
T3LSB-1          1;
T2ADR-26         1;      % 25 bit address field in processor
T1ADR-26         1;      % pipeline
T3ADR-26         1;
T2ADR-21         1;
T1ADR-21         1;
T3ADR-21         1;
T2ADR-16         1;
T1ADR-16         1;
T3ADR-16         1;
T2ADR-11         1;
T1ADR-11         1;
T3ADR-11         1;
T2ADR-6          1;
T1ADR-6          1;
T3ADR-6          1;
S1ADRPAR5        1;      % Spy1 address parity
S1KEY-14         1;      % Spy1 key.  15 bits.
S1KEY-9          1;
S1KEY-4          1;
S2ADR-9          1;      % Spy2 address. 10 bits
S1ADR-9          1;      % Spy1 address. 10 bits
S2ADR-4          1;
S1ADR-4          1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CHADDR3          location: F6I2                                   %
% shift: R11,   clear: N01,   clock: C12,   sdi: R09,   sdo: J01          %
% function: 4 OF 5 CACHE ADDRESS ARRAYS.                                  %
%*************************************************************************%

      DRAMHB-8          1;      % DATA RAM HIGH ADDRESS
      DRAMHB-3          1;
      DRAMHA-8          1;
      DRAMHA-3          1;
      DRAMLB-8          1;      % DATA RAM LOW ADDRESS
      DRAMLB-3          1;
      DRAMLA-8          1;
      DRAMLA-3          1;
      KYRMIN-11         1;      % 15 BITS OF KEY-RAM INPUT
      KYRMIN-10         1;
      KYRMIN-9          1;
      KYECIN-11         1;      % 15 BITS OF KEY-RAM ECC INPUT
      KYECIN-10         1;
      KYECIN-9          1;
      KADRAB-8          1;      % 10 BIT OF KEY-RAM ADDRESS INPUT
      KADRAB-3          1;      % SEPARATED INTO 2 BLOCKS
      KADRAA-8          1;
      KADRAA-3          1;
      KADRBB-8          1;      % BLOCK-B OF KEY-RAM ADDRESS INPUT
      KADRBB-3          1;
      KADRBA-8          1;
      KADRBA-3          1;
      TRANLSD-3         1;      % LSB FIELD. IN ARRAY #1-4 IS
      TRANLEN-3         1;      % USED AS LSD AND LEN FIELD
      CHADR4ERR         1;      % CACHE ADDRESS ARRAY5 IN ERROR
      T2SPARE-0         1;
      T1SPARE-0         1;
      T3SPARE-0         1;
      T2TAG-3           1;      % TAG FIELD
      T1TAG-3           1;
      T3TAG-3           1;
      T2LEN-3           1;      % 2 BIT LSB IN PROCESSOR PIPELINE,
      T1LEN-3           1;      % IN ARRAY #1-4 IS USED AS LSD AND LEN
      T3LEN-3           1;
      T2LSD-3           1;
      T1LSD-3           1;
      T3LSD-3           1;
```

14.1 MA (Continued)

```
T2ADR-25          1;      % 25 BIT ADDRESS FIELD IN PROCESSOR
T1ADR-25          1;      % PIPELINE
T3ADR-25          1;
T2ADR-20          1;
T1ADR-20          1;
T3ADR-20          1;
T2ADR-15          1;
T1ADR-15          1;
T3ADR-15          1;
T2ADR-10          1;
T1ADR-10          1;
T3ADR-10          1;
T2ADR-5           1;
T1ADR-5           1;
T3ADR-5           1;
S1ADRPAR4         1;      % SPY1 ADDRESS PARITY
S1KEY-13          1;      % SPY1 KEY.  15 BITS.
S1KEY-8           1;
S1KEY-3           1;
S2ADR-8           1;      % SPY2 ADDRESS. 10 BITS
S1ADR-8           1;      % SPY1 ADDRESS. 10 BITS
S2ADR-3           1;
S1ADR-3           1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CHADDR3          location: D2I2                                   %
% shift: R11,   clear: N01,   clock: C12,   sdi: R09,   sdo: J01          %
% function: 3 OF 5 CACHE ADDRESS ARRAYS.                                  %
%*************************************************************************%

     DRAMHB-7          1;      % DATA RAM HIGH ADDRESS
     DRAMHB-2          1;
     DRAMHA-7          1;
     DRAMHA-2          1;
     DRAMLB-7          1;      % DATA RAM LOW ADDRESS
     DRAMLB-2          1;
     DRAMLA-7          1;
     DRAMLA-2          1;
     KYRMIN-8          1;      % 15 BITS OF KEY-RAM INPUT
     KYRMIN-7          1;
     KYRMIN-6          1;
     KYECIN-8          1;      % 15 BITS OF KEY-RAM ECC INPUT
     KYECIN-7          1;
     KYECIN-6          1;
     KADRAB-7          1;      % 10 BIT OF KEY-RAM ADDRESS INPUT
     KADRAB-2          1;      % SEPARATED INTO 2 BLOCKS
     KADRAA-7          1;
     KADRAA-2          1;
     KADRBB-7          1;      % BLOCK-B OF KEY-RAM ADDRESS INPUT
     KADRBB-2          1;
     KADRBA-7          1;
     KADRBA-2          1;
     TRANLSD-2         1;      % LSB FIELD. IN ARRAY #1-4 IS
     TRANLEN-2         1;      % USED AS LSD AND LEN FIELD
     CHADR3ERR         1;      % CACHE ADDRESS ARRAY5 IN ERROR
     T2TAG-7           1;
     T1TAG-7           1;
     T3TAG-7           1;
     T2TAG-2           1;      % TAG FIELD
     T1TAG-2           1;
     T3TAG-2           1;
```

14.1 MA (Continued)

```
T2LEN-2          1;      % 2 BIT LSB IN PROCESSOR PIPELINE,
T1LEN-2          1;      % IN ARRAY #1-4 IS USED AS LSD AND LEN
T3LEN-2          1;
T2LSD-2          1;
T1LSD-2          1;
T3LSD-2          1;
T2ADR-24         1;      % 25 BIT ADDRESS FIELD IN PROCESSOR
T1ADR-24         1;      % PIPELINE
T3ADR-24         1;
T2ADR-19         1;
T1ADR-19         1;
T3ADR-19         1;
T2ADR-14         1;
T1ADR-14         1;
T3ADR-14         1;
T2ADR-9          1;
T1ADR-9          1;
T3ADR-9          1;
T2ADR-4          1;
T1ADR-4          1;
T3ADR-4          1;
S1ADRPAR3        1;      % SPY1 ADDRESS PARITY
S1KEY-12         1;      % SPY1 KEY.  15 BITS.
S1KEY-7          1;
S1KEY-2          1;
S2ADR-7          1;      % SPY2 ADDRESS. 10 BITS
S1ADR-7          1;      % SPY1 ADDRESS. 10 BITS
S2ADR-2          1;
S1ADR-2          1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CHADDR3          location: K4I2                                   %
% shift: R11,   clear: N01,   clock: C12,   sdi: R09,   sdo: J01          %
% function: 2 OF 5 CACHE ADDRESS ARRAYS.                                  %
%*************************************************************************%

     DRAMHB-6          1;      % DATA RAM HIGH ADDRESS
     DRAMHB-1          1;
     DRAMHA-6          1;
     DRAMHA-1          1;
     DRAMLB-6          1;      % DATA RAM LOW ADDRESS
     DRAMLB-1          1;
     DRAMLA-6          1;
     DRAMLA-1          1;
     KYRMIN-5          1;      % 15 BITS OF KEY-RAM INPUT
     KYRMIN-4          1;
     KYRMIN-3          1;
     KYECIN-5          1;      % 15 BITS OF KEY-RAM ECC INPUT
     KYECIN-4          1;
     KYECIN-3          1;
     KADRAB-6          1;      % 10 BIT OF KEY-RAM ADDRESS INPUT
     KADRAB-1          1;      % SEPARATED INTO 2 BLOCKS
     KADRAA-6          1;
     KADRAA-1          1;
     KADRBB-6          1;      % BLOCK-B OF KEY-RAM ADDRESS INPUT
     KADRBB-1          1;
     KADRBA-6          1;
     KADRBA-1          1;
     TRANLSD-1         1;      % LSB FIELD. IN ARRAY #1-4 IS
     TRANLEN-1         1;      % USED AS LSD AND LEN FIELD
     CHADR2ERR         1;      % CACHE ADDRESS ARRAY5 IN ERROR
     T2TAG-6           1;
     T1TAG-6           1;
     T3TAG-6           1;
     T2TAG-1           1;      % TAG FIELD
     T1TAG-1           1;
     T3TAG-1           1;
     T2LEN-1           1;      % 2 BIT LSB IN PROCESSOR PIPELINE,
     T1LEN-1           1;      % IN ARRAY #1-4 IS USED AS LSD AND LEN
     T3LEN-1           1;
     T2LSD-1           1;
     T1LSD-1           1;
     T3LSD-1           1;
```

14.1 MA (Continued)

```
T2ADR-23          1;      % 25 BIT ADDRESS FIELD IN PROCESSOR
T1ADR-23          1;      % PIPELINE
T3ADR-23          1;
T2ADR-18          1;
T1ADR-18          1;
T3ADR-18          1;
T2ADR-13          1;
T1ADR-13          1;
T3ADR-13          1;
T2ADR-8           1;
T1ADR-8           1;
T3ADR-8           1;
T2ADR-3           1;
T1ADR-3           1;
T3ADR-3           1;
S1ADRPAR2         1;      % SPY1 ADDRESS PARITY
S1KEY-11          1;      % SPY1 KEY.  15 BITS.
S1KEY-6           1;
S1KEY-1           1;
S2ADR-6           1;      % SPY2 ADDRESS. 10 BITS
S1ADR-6           1;      % SPY1 ADDRESS. 10 BITS
S2ADR-1           1;
S1ADR-1           1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CHADDR3          location: M8I2                                   %
% shift: R11,   clear: N01,   clock: C12,   sdi: R09,   sdo: J01          %
% function: 1 OF 5 CACHE ADDRESS ARRAYS.                                  %
%*************************************************************************%

      DRAMHB-5          1;      % DATA RAM HIGH ADDRESS
      DRAMHB-0          1;
      DRAMHA-5          1;
      DRAMHA-0          1;
      DRAMLB-5          1;      % DATA RAM LOW ADDRESS
      DRAMLB-0          1;
      DRAMLA-5          1;
      DRAMLA-0          1;
      KYRMIN-2          1;      % 15 BITS OF KEY-RAM INPUT
      KYRMIN-1          1;
      KYRMIN-0          1;
      KYECIN-2          1;      % 15 BITS OF KEY-RAM ECC INPUT
      KYECIN-1          1;
      KYECIN-0          1;
      KADRAB-5          1;      % 10 BIT OF KEY-RAM ADDRESS INPUT
      KADRAB-0          1;      % SEPARATED INTO 2 BLOCKS
      KADRAA-5          1;
      KADRAA-0          1;
      KADRBB-5          1;      % BLOCK-B OF KEY-RAM ADDRESS INPUT
      KADRBB-0          1;
      KADRBA-5          1;
      KADRBA-0          1;
      TRANLSD-0         1;      % LSB FIELD. IN ARRAY #1-4 IS
      TRANLEN-0         1;      % USED AS LSD AND LEN FIELD
      CHADR1ERR         1;      % CACHE ADDRESS ARRAY5 IN ERROR
      T2TAG-5           1;
      T1TAG-5           1;
      T3TAG-5           1;
      T2TAG-0           1;      % TAG FIELD
      T1TAG-0           1;
      T3TAG-0           1;
      T2LEN-0           1;      % 2 BIT LSB IN PROCESSOR PIPELINE,
      T1LEN-0           1;      % IN ARRAY #1-4 IS USED AS LSD AND LEN
      T3LEN-0           1;
      T2LSD-0           1;
      T1LSD-0           1;
      T3LSD-0           1;
```

14.1 MA (Continued)

```
T2ADR-22          1;      % 25 BIT ADDRESS FIELD IN PROCESSOR
T1ADR-22          1;      % PIPELINE
T3ADR-22          1;
T2ADR-17          1;
T1ADR-17          1;
T3ADR-17          1;
T2ADR-12          1;
T1ADR-12          1;
T3ADR-12          1;
T2ADR-7           1;
T1ADR-7           1;
T3ADR-7           1;
T2ADR-2           1;
T1ADR-2           1;
T3ADR-2           1;
S1ADRPAR1         1;      % SPY1 ADDRESS PARITY
S1KEY-10          1;      % SPY1 KEY.  15 BITS.
S1KEY-5           1;
S1KEY-0           1;
S2ADR-5           1;      % SPY2 ADDRESS. 10 BITS
S1ADR-5           1;      % SPY1 ADDRESS. 10 BITS
S2ADR-0           1;
S1ADR-0           1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CAFLAG6          location: R6D4                                   %
% shift: A05,   clear: P12,   clock: C12,   sdi: A06,   sdo: C02          %
% function: SLAVE CACHE FLAG ARRAY.                                       %
%*************************************************************************%

     SFLGERROR          1;      % FLAG ARRAY ERROR
     SJW                3;      % JW STATE MACHINE, 3 BITS
     SDV2F              1;      % DATA DIR AVAILABLE
     SPIF              11;      % DIRECTORY PIORITY SCHEM FLIP FLOPS.
     SSPYFG4            1;      % SPY FLAG FLIP FLOP, 4 BITS
     STRFG4             1;      % PROCESSOR FLAG FLIP FLOP, 4 BITS
     SSPYFG3            1;
     STRFG3             1;
     SSPYFG2            1;
     STRFG2             1;
     SSPYFG1            1;
     STRFG1             1;
     ST1HITFL           1;      % HIT AT BLOCK 1 (FROM PROCESSOR)
                                % active low
     ST0HITFL           1;      % HIT AT BLOCK 0 (FROM PROCESSOR)
                                % active low
     SMHITF             1;      % SPY WANTS SAME DATA AS PROCESSOR
     SMODF              1;
     SS1HITF            1;
     SS1HIT             1;      % HIT AT BLOCK 1 (FROM SPY)
     SS0HIT             1;      % HIT AT BLOCK 0
     SMFREG             2;      % Memory register flag
     SMDOSEL            1;      % MDO REGISTER SELECT
     SMDOSL1            1;      % MDO REGISTER SELECT CONDITION
     SABV2F             1;      % DIR A OR B AVAILABLE
     SLRUWF             1;      % LRU FLIP FLOP
     SINGFLG3L          1;      % IGNORE INVALID FLAG AT T3
     SINGFLG1L          1;      % IGNORE INVALID FLAG AT T1
     SINPROC3           1;      %
     SINPROC1           1;      %
     SLRUFH             1;      %
     SLIMER3            1;      % LIMIT ERROR AT T3
     SLIMER1            1;      % LIMIT ERROR AT T1
     SSTOPQF            1;      % STOPQ FLIP-FLOP OUTPUT
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CAFLAG6          location: P2D4                                   %
% shift: A05,   clear: P12,   clock: C12,   sdi: A06,   sdo: C02          %
% function: MASTER CACHE FLAG ARRAY.                                      %
%*************************************************************************%

     MFLGERROR          1;      % FLAG ARRAY ERROR
     MJW                3;      % JW STATE MACHINE, 3 BITS
     MDV2F              1;      % DATA DIR AVAILABLE
     MPIF              11;      % DIRECTORY PIORITY SCHEM FLIP FLOPS.
     MSPYFG4            1;      % SPY FLAG FLIP FLOP, 4 BITS
     MTRFG4             1;      % PROCESSOR FLAG FLIP FLOP, 4 BITS
     MSPYFG3            1;
     MTRFG3             1;
     MSPYFG2            1;
     MTRFG2             1;
     MSPYFG1            1;
     MTRFG1             1;
     MT1HITFL           1;      % HIT AT BLOCK 1 (FROM PROCESSOR)
     MT0HITFL           1;      % HIT AT BLOCK 0 (FROM PROCESSOR)
     MMHITF             1;      % SPY WANTS SAME DATA AS PROCESSOR
     MMODF              1;
     MS1HITF            1;
     MS1HIT             1;      % HIT AT BLOCK 1 (FROM SPY)
     MS0HIT             1;      % HIT AT BLOCK 0
     MMFREG             2;
     MMDOSEL            1;      % MDO REGISTER SELECT
     MMDOSL1            1;      % MDO REGISTER SELECT CONDITION
     MABV2F             1;      % DIR A OR B AVAILABLE
     MLRUWF             1;      % LRU FLIP FLOP
     MINGFLG3L          1;      % IGNORE INVALID FLAG AT T3
     MINGFLG1L          1;      % IGNORE INVALID FLAG AT T1
     MINPROC3           1;      %
     MINPROC1           1;      %
     MLRUFH             1;
     MLIMER3            1;
     MLIMER1            1;
     MSTOPQF            1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CACNTL5          location: P2I2                                   %
% shift: P13,   clear: B08,   clock: C12,   sdi: P05,   sdo: D13          %
% function: SLAVE CACHE CONTROL ARRAY.                                    %
%*************************************************************************%

      SRDVAL            1;      % R BUS VALID
      SLSBENL           1;      % Lsb enable: controls word select
      SLDMREG           1;      % LOAD MDI REG
      SS2COMDVAL        1;      % SPY2 COMMAND VALID BIT
      SS1COMDVAL        1;      % SPY1 COMMAND VALID BIT
      SS2CMD-1          1;      % SPY2 2 BITS COMMAND
      SS1CMD-1          1;      % SPY1 2 BITS COMMAND
      SS2CMD-0          1;
      SS1CMD-0          1;
      SCNTLERR          1;      % CACNTL ARRAY IN ERROR
      SCDOEN            1;      % MCACM-D CONTROL SIGNALS
      SCEOEN            1;      % WRITE ECC CONTROL PATH
      SCDOSL            1;
      SPF6              1;
      SPF2              1;
      STAGF             1;
      SORDERF           1;      % PIPELINE ORDER FLIP FLOP
      SSPAREF           1;
      SJM               4;      % JM 4-BIT STATE MACHINE
      SF1               1;      % WRITE CONTROL F/F
      SF3               1;
      ST2CMD-4          1;      % TRAN  5 BIT COMMAND (FROM PROCESSOR)
      ST1CMD-4          1;
      ST3CMD-4          1;
      ST2CMD-3          1;
      ST1CMD-3          1;
      ST3CMD-3          1;
      ST2CMD-2          1;
      ST1CMD-2          1;
      ST3CMD-2          1;
      ST2CMD-1          1;
      ST1CMD-1          1;
      ST3CMD-1          1;
      ST2CMD-0          1;
      ST1CMD-0          1;
      ST3CMD-0          1;
      ST2COMDPAR        1;      % COMMAND PARITY BIT
      ST1COMDPAR        1;
      ST3COMDPAR        1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: CACNTL5          location: P2F8                                   %
% shift: P13,   clear: B08,   clock: C12,   sdi: P05,   sdo: D13          %
% function: MASTER CACHE CONTROL ARRAY.                                   %
%*************************************************************************%

      MRDVAL            1;      % R BUS VALID
      MLSBENL           1;
      MLDMREG           1;      % LOAD MDI REG
      MS2COMDVAL        1;      % SPY2 COMMAND VALID BIT
      MS1COMDVAL        1;      % SPY1 COMMAND VALID BIT
      MS2CMD-1          1;      % SPY2 2 BITS COMMAND
      MS1CMD-1          1;      % SPY1 2 BITS COMMAND
      MS2CMD-0          1;
      MS1CMD-0          1;
      MCNTLERR          1;      % CACNTL ARRAY IN ERROR
      MCDOEN            1;
      MCEOEN            1;      % WRITE ECC CONTROL PATH
      MCDOSL            1;
      MPF6              1;
      MPF2              1;
      MTAGF             1;
      MORDERF           1;      % PIPELINE ORDER FLIP FLOP
      MSPAREF           1;
      MJM               4;      % JM 4-BIT STATE MACHINE
      MF1               1;
      MF3               1;
      MT2CMD-4          1;      % TRAN2 5 BIT COMMAND (FROM PROCESSOR)
      MT1CMD-4          1;
      MT3CMD-4          1;
      MT2CMD-3          1;
      MT1CMD-3          1;
      MT3CMD-3          1;
      MT2CMD-2          1;
      MT1CMD-2          1;
      MT3CMD-2          1;
      MT2CMD-1          1;
      MT1CMD-1          1;
      MT3CMD-1          1;
      MT2CMD-0          1;
      MT1CMD-0          1;
      MT3CMD-0          1;
      MT2COMDPAR        1;      % COMMAND PARITY BIT
      MT1COMDPAR        1;
      MT3COMDPAR        1;
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: BUFFER          location: D2B0                                    %
% clear: E15,   shift: H15,   clock: C12,   sdi: J14,   sdo: H14          %
% function: LIMIT CHECK GATE 2 OF 2.                                      %
%*************************************************************************%

        CHECKIT-1       1;      % TIME DELAYED VALUE OF BASEADD ENABLE

%*************************************************************************%
% chip: BUFFER          location: A8B0                                    %
% clear: E15,   shift: H15,   clock: C12,   sdi: J14,   sdo: H14          %
% function: LIMIT CHECK GATE 1 OF 2.                                      %
%*************************************************************************%

        CHECKIT-0       1;      % TIME DELAYED VALUE OF BASEADD ENABLE

%*************************************************************************%
% chip: ERREC2          location: I0K6                                    %
% shift: R09,   clear: R07,   clock: C12,   sdi: P03,   sdo: E03          %
% function: SLAVE ERROR RECORDER.                                         %
%*************************************************************************%

    NOTUSEDER2          6;  % NOT USED.
    MULTERROR1          1;  % THE ERROR LATCHED IS MULTIPLE ERROR.
    SNGLERROR1          1;  % THE ERROR LATCHED IS SINGLE ERROR.
    ERRORHOLD1          1;  % SET BY READ ERROR REPORT COMMAND,
                            % RESET BY READ ERROR ADDRESS COMMAND.
    NOTUSEDER1         10;  % NOT USED.
    NOTUSED3E           3;  % THE FOLLOWING ARE STUFFINGS FOR AWQUE.
    NOTUSED3D           3;
    NOTUSED3C           3;
    NOTUSED3B           3;
    NOTUSED3A           3;
    NOTUSED1A           1;

    SYND3-7:0           8;  % SYNDROME OF WORD 3.
    SYND2-7:0           8;  % SYNDROME OF WORD 2.
    ERRADR-04           1;  % ERROR ADDRESS.
```

14.1 MA (Continued)

```
BUFADR-04       1;   % BUFFER FOR CURRENT MEMORY REQUEST ADDRESS.
ERRADR-03       1;
BUFADR-03       1;
ERRADR-02       1;
BUFADR-02       1;
ERRADR-01       1;
BUFADR-01       1;
ERRADR-00       1;
BUFADR-00       1;
ERRADR-15       1;
BUFADR-15       1;
ERRADR-14       1;
BUFADR-14       1;
ERRADR-13       1;
BUFADR-13       1;
ERRADR-12       1;
BUFADR-12       1;
ERRADR-11       1;
BUFADR-11       1;
ERRADR-10       1;
BUFADR-10       1;
ERRADR-09       1;
BUFADR-09       1;
ERRADR-08       1;
BUFADR-08       1;
ERRADR-07       1;
BUFADR-07       1;
ERRADR-06       1;
BUFADR-06       1;
ERRADR-05       1;
BUFADR-05       1;
```

14.1 MA (Continued)

```
%*********************************************************************%
% chip: ERREC2          location: I0K6                                %
% shift: R09,   clear: R07,   clock: C12,   sdi: P03,   sdo: E03      %
% function: MASTER ERROR RECORDER.                                    %
%*********************************************************************%

    RBUSERVEC          5;   % RBUS ERROR VECTOR.
    RDATAPARER         1;   % CORRECTED DATA PARITY ERROR.
    MULTERROR0         1;   % THE ERROR LATCHED IS MULTIPLE ERROR.
    SNGLERROR0         1;   % THE ERROR LATCHED IS SINGLE ERROR.
    ERRORHOLD0         1;   % SET BY READ ERROR REPORT COMMAND,
                            % RESET BY READ ERROR ADDRESS COMMAND.
    BROKEN-B           1;   % CACHE MULTIPLE ERROR BROKEN HIDDEN STATE.
    BROKEN-E           1;   % CACHE WRITE BACK BROKEN HIDDEN STATE.
    BROKEN-A           1;   % MBUS MULTIBLE ERROR BROKEN HIDDEN STATE.
    WRITIMING          6;   % MBUS VALID TIMING PIPE FOR WRITE OPERATION
    MEMSRCIDO          2;   % MCACM-ID HIDDEN STATE.
    EVPIPE2            5;   % ERROR VECTOR PIPE II.
    EVPIPE3            5;   % ERROR VECTOR PIPE III.
    EVPIPE1            5;   % ERROR VECTOR PIPE I.
    SYND1-7:0          8;   % SYNDROME OF WORD 1.
    SYND0-7:0          8;   % SYNDROME OF WORD 0.
    ERRPHYID-4         1;   % ERROR PHYSICAL ID.
    BUFPHYID-4         1;   % PHYSICAL ID BUFFER FOR CURRENT MEMORY REQ.
    ERRPHYID-3         1;
    BUFPHYID-3         1;
    ERRPHYID-2         1;
    BUFPHYID-2         1;
    ERRPHYID-1         1;
    BUFPHYID-1         1;
    ERRPHYID-0         1;
    BUFPHYID-0         1;
    ERRBLKID-0         1;   % ERROR BLOCK NUMBER.
    NOTUSEDER0         1;   % NOT USED.
    ERRLGCID-3         1;   % ERROR LOGICAL ID.
    BUFLGCID-3         1;   % LOGICAL ID BUFFER FORC CURRENT MEMORY REQ.
    ERRLGCID-2         1;
    BUFLGCID-2         1;
    ERRLGCID-1         1;
    BUFLGCID-1         1;
    ERRLGCID-0         1;
    BUFLGCID-0         1;
```

14.1 MA (Continued)

```
    ERRADR-21          1;    % ERROR ADDRESS.
    BUFADR-21          1;    % BUFFER FOR CURRENT MEMORY REQUEST ADDRESS
    ERRADR-20          1;
    BUFADR-20          1;
    ERRADR-19          1;
    BUFADR-19          1;
    ERRADR-18          1;
    BUFADR-18          1;
    ERRADR-17          1;
    BUFADR-17          1;
    ERRADR-16          1;
    BUFADR-16          1;

%*************************************************************************%
% chip: LIMCHK          location:                                         %
% shift: B09,   clear: A04,   clock: C12,   sdi: R10,   sdo: E01          %
% function: LIMIT CHECKER ZERO.                                           %
%*************************************************************************%

    LC0ADDRESS        27;   % ADDRESS IN LIMIT CHECKER 0.
    LC0LIMPAR          1;   % PARITY OF ADDRESS IN LIMIT CHECKER 0.
    LC0LIMIT          27;   % ADDRESS IN LIMIT CHECKER 0.

%*************************************************************************%
% chip: LIMCHK          location:                                         %
% shift: B09,   clear: A04,   clock: C12,   sdi: R10,   sdo: E01          %
% function: LIMIT CHECKER ONE.                                            %
%*************************************************************************%

        LC1ADDRESS     27;      % ADDRESS IN LIMIT CHECKER 1.
        LC1LIMPAR       1;      % PARITY OF ADDRESS IN LIMIT CHECKER 1.
        LC1LIMIT       27;      % ADDRESS IN LIMIT CHECKER 1.
```

14.1 MA (Continued)

```
%
% the following pins of RAMARR are X-cell X291 f/f output pins.
%  J14, G14, E14, D15, A02, C04, A14, E15, C01, F02, M01, J02, N01, K02
%
%*************************************************************************%
% chip: RAMARR          location: M8K6                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 8 OF 8  -  TAG PORTION                     %
%*************************************************************************%

       TAGNU13:8       6;      % UPPER UNUSED BITS OF TAG RAMARR

       WTAGPARITY      1;      % AW INPUT QUEUE TAG PARITY BIT
       WTAG(6:0)       7;      % 7 BIT AW INPUT QUEUE TAG FIELD

%*************************************************************************%
% chip: RAMARR          location: F6K6                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 7 OF 8  -  COMMAND PORTION                 %
%*************************************************************************%

       CMDNU           4;      % UPPER UNUSED BITS OF COMMAND PORTION

       WCOMDPAR        1;      % AW INPUT QUEUE COMMAND PARITY
       WCOMMAND        5;      % AW INPUT QUEUE COMMAND FIELD
       WLENGTH         4;      % AW INPUT QUEUE LENGTH FIELD

%*************************************************************************%
% chip: RAMARR          location: A8N0                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 6 OF 8.                                    %
%*************************************************************************%

       HIADRNU         1;      % UPPER BIT OF HIGH ADDRESS PORTION

       WADRPAR         1;      % AW INPUT QUEUE ADDRESS PARITY FIELD
       WBASEIND        4;      % AW INPUT QUEUE BASE INDICANT FIELD
       WADR35:28       8;      % AW INPUT QUEUE UPPER 4 BITS OF ADDR.
```

14.1 MA (Continued)

```
%*************************************************************************%
% chip: RAMARR          location: D2N0                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 5 OF 8  -  MID ADDRESS PORTION             %
%*************************************************************************%

        WADR27:14       14;     % AW INPUT QUEUE ADDRESS FIELD

%*************************************************************************%
% chip: RAMARR          location: P2N0                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 4 OF 8  -  LOW ADDRESS PORTION             %
%*************************************************************************%

        WADR13:4        10;     % AW INPUT QUEUE ADDRESS FIELD
        WADR3:0          4;     % "    "    "  LSD  "     "

%*************************************************************************%
% chip: RAMARR          location: A8K6                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 3 OF 8  -  UPPER DATA PORTION              %
%*************************************************************************%

        WDATNU          1;      % UNUSED BITS OF UPPER DATA PORTION

        WDATPAR         1;      % AW INPUT QUEUE DATA PARITY FIELD
        WDAT39:28      12;      % AW INPUT QUEUE DATA FIELD

%*************************************************************************%
% chip: RAMARR          location: I0N0                                    %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14          %
% function: AW QUEUE RAM ARRAY 2 OF 8  -  MID DATA PORTION                %
%*************************************************************************%

        WDAT27:14       14;     % AW INPUT QUEUE DATA FIELD
```

14.1 MA (Continued)

```
%************************************************************************%
% chip: RAMARR          location: R6K6                                   %
% shift: P04,   clear: A08,   clock: C12,   sdi: P03,   sdo: J14         %
% function: AW QUEUE RAM ARRAY 1 OF 8  -  LOW DATA PORTION               %
%************************************************************************%

        WDAT13:0        14;     % AW INPUT QUEUE DATA FIELD

%************************************************************************%
% chip: MAWQUE2         location: K4K6                                   %
% shift: J01,   clear: H01,   clock: C12,   sdi: L01,   sdo: N01         %
% function: A&W INPUT QUEUE CONTROLLER ARRAY                             %
%************************************************************************%

        WADPAR          1;      % PREDICTED WRITE ADDRESS PARITY
        WAD             4;      % WRITE ADDRESS FOR AW QUEUE RAMS
        RADPAR          1;      % PREDICTED READ ADDRESS PARITY
        RAD             4;      % READ ADDRESS FOR AW QUEUE RAMS
        QCTRLBRK        1;      % QUEUE CONTROL HAS DETECTED A FAULT
        MSTATE          2;      % MAWQUE MAINTENANCE MODE REGISTER
                                %   0 - NORMAL OPERATION
                                %   1 - HIDDEN STATE WRITE OF QUEUE RAM
                                %   2 - QUEUE RECIRCULATION MODE
                                %   3 - QUEUE RAM WRITE DISABLE - USED
                                %       TO READ QUEUE RAM AND PREVENT
                                %       OTHER HIDDEN STATE OPERATIONS
                                %       FROM CORRUPTING IT.
        QUEFULL         1;      % TOP OF QUEUE CONTAINS A COMMAND -
                                %       USE "WQREGVALID" INSTEAD
        SECOND          1;      % TOP OF QUEUE IS IN SECOND OF 2 CLOCKS
        WRTSTOP         1;      % WRITE ABORT (SENT TO STOP AS WRTSTOP)
        RBISWRT         1;      % THE RETURN ON RBUS IS FROM A WRITE
        FLUSHES         4;      % NUMBER OF FLUSHES IN AW QUEUE
        NOBUS           1;      % AW INPUT QUEUE IS FULL - NO INPUT
                                % WILL BE ACCEPTED.
                                %       USE "WFULL" INSTEAD
        BUSPR           3;      % BUS PRIORITIZATION OF LAST BUSREQ'S
        WREQID          3;      % REGISTERED FROM BUSID(3:0)


CHNEND;         % END OF MCACM-ADDRESS-CARD's DATA CHAIN
```

14.1 MA (Continued)

```
REGLIST MA;     % BEGIN OF MCACM-ADDRESS-CARD'S REGISTER LIST

      MCACMERROR  12;          % accumulation of MCACM errors & broken
                  BROKEN;      % MCACM A BROKEN SIGNAL
                  UCONFIGERR;  % invalid memory data card type
                  BPCTLERR;    % error in intercache backplane signals
                  ARBERR;      % error in memory backplane arbitration
                  MCONFIGERR;  % invalid memory data card type
                  MULTERROR1;  % THE ERROR LATCHED IS MULTIPLE ERROR.
                  RDATAPARER;  % CORRECTED DATA PARITY ERROR.
                  MULTERROR0;  % THE ERROR LATCHED IS MULTIPLE ERROR.
                  BROKEN-B;    % CACHE MULTIPLE ERROR BROKEN HIDDEN STA
                  BROKEN-E;    % CACHE WRITE BACK BROKEN HIDDEN STATE.
                  BROKEN-A;    % MBUS MULTIBLE ERROR BROKEN HIDDEN STAT
                  QCTRLBRK#    % AW QUEUE CONTROL HAS DETECTED A FAULT
                               % This will not cause BROKEN.

      MAERROR1 11;             % accumulation of MCACM errors & brokens
                  BROKEN;      % MCACM A BROKEN SIGNAL
                  CHADR5ERR;   % CHADDR #5 error
                  CHADR4ERR;   % CHADDR #4 error
                  CHADR3ERR;   % CHADDR #3 error
                  CHADR2ERR;   % CHADDR #2 error
                  CHADR1ERR;   % CHADDR #1 error
                  SFLGERROR;   % slave FLAG ARRAY ERROR
                  MFLGERROR;   % master FLAG ARRAY ERROR
                  SCNTLERR;    % slave CACNTL ARRAY IN ERROR
                  MCNTLERR;    % master CACNTL ARRAY IN ERROR
                  QCTRLBRK#    % AW QUEUE CONTROL HAS DETECTED A FAULT
                               % This will not cause BROKEN.

      MAERROR2 2;              % MULTI-PROC BROKEN
                  BPCTLERR;    % error in intercache backplane signals
                  ARBERR#      % error in memory backplane arbitration

      MDERROR  6;              % MD BROKEN
                  BROKEN-B;    % CACHE MULTIPLE ERROR BROKEN HIDDEN ST
                  RDATAPARER;  % CORRECTED DATA PARITY ERROR.
                  MULTERROR1;  % THE ERROR LATCHED IS MULTIPLE ERROR.
                  MULTERROR0;  % THE ERROR LATCHED IS MULTIPLE ERROR.
                  SNGLERROR1;  % THE ERROR LATCHED IS SINGLE ERROR.
                  SNGLERROR0#  % THE ERROR LATCHED IS SINGLE ERROR.
```

14.1 MA (Continued)

```
MDCERROR1 4;              % MDC-MCACM INTERFACE BROKEN
          UCONFIGERR;     % invalid memory data card type
          MCONFIGERR;     % invalid memory data card type
          BROKEN-E;       % CACHE WRITE BACK BROKEN HIDDEN STATE.
          BROKEN-A#       % MBUS MULTIBLE ERROR BROKEN HIDDEN STA

ERRPHYID 5;
          ERRPHYID-4;
          ERRPHYID-3;
          ERRPHYID-2;
          ERRPHYID-1;
          ERRPHYID-0#

BUFPHYID 5;
          BUFPHYID-4;
          BUFPHYID-3;
          BUFPHYID-2;
          BUFPHYID-1;
          BUFPHYID-0#

ERRLGCID 4;
          ERRLGCID-3;
          ERRLGCID-2;
          ERRLGCID-1;
          ERRLGCID-0#

BUFLGCID 4;
          BUFLGCID-3;
          BUFLGCID-2;
          BUFLGCID-1;
          BUFLGCID-0#
```

14.1 MA (Continued)

```
ERRADR 22;
       ERRADR-21;
       ERRADR-20;
       ERRADR-19;
       ERRADR-18;
       ERRADR-17;
       ERRADR-16;
       ERRADR-15;
       ERRADR-14;
       ERRADR-13;
       ERRADR-12;
       ERRADR-11;
       ERRADR-10;
       ERRADR-09;
       ERRADR-08;
       ERRADR-07;
       ERRADR-06;
       ERRADR-05;
       ERRADR-04;
       ERRADR-03;
       ERRADR-02;
       ERRADR-01;
       ERRADR-00#
```

14.1 MA (Continued)

```
ERHEXADR 22;
         ERRADR-21;
         ERRADR-20;
         ERRADR-19;
         ERRADR-16;
         ERRADR-15;
         ERRADR-14;
         ERRADR-13;
         ERRADR-12;
         ERRADR-11;
         ERRADR-10;
         ERRADR-09;
         ERRADR-08;
         ERRADR-07;
         ERRADR-06;
         ERRADR-05;
         ERRADR-04;
         ERRADR-03;
         ERRADR-02;
         ERRADR-01;
         ERRADR-00;
         ERRADR-18;
         ERRADR-17#
```

14.1 MA (Continued)

```
BUFADR 22;
        BUFADR-21;
        BUFADR-20;
        BUFADR-19;
        BUFADR-18;
        BUFADR-17;
        BUFADR-16;
        BUFADR-15;
        BUFADR-14;
        BUFADR-13;
        BUFADR-12;
        BUFADR-11;
        BUFADR-10;
        BUFADR-09;
        BUFADR-08;
        BUFADR-07;
        BUFADR-06;
        BUFADR-05;
        BUFADR-04;
        BUFADR-03;
        BUFADR-02;
        BUFADR-01;
        BUFADR-00#

ERREPORT 4;
        SYND3-7:0;
        SYND2-7:0;
        SYND1-7:0;
        SYND0-7:0#

KADR1A  10;     % BLOCK1 KEY ADDRESS DIRA OF KEY RAM
        KADRAA-9;
        KADRAA-8;
        KADRAA-7;
        KADRAA-6;
        KADRAA-5;
        KADRAA-4;
        KADRAA-3;
        KADRAA-2;
        KADRAA-1;
        KADRAA-0#
```

14.1 MA (Continued)

```
KADR1B  10;     % BLOCK1 KEY ADDRESS DIRB OF KEY RAM
        KADRBA-9;
        KADRBA-8;
        KADRBA-7;
        KADRBA-6;
        KADRBA-5;
        KADRBA-4;
        KADRBA-3;
        KADRBA-2;
        KADRBA-1;
        KADRBA-0#

KADR0A  10;     % BLOCK0 KEY ADDRESS DIRA OF KEY RAM
        KADRAB-9;
        KADRAB-8;
        KADRAB-7;
        KADRAB-6;
        KADRAB-5;
        KADRAB-4;
        KADRAB-3;
        KADRAB-2;
        KADRAB-1;
        KADRAB-0#

KADR0B  10;     % BLOCK0 KEY ADDRESS DIRB OF KEY RAM
        KADRBB-9;
        KADRBB-8;
        KADRBB-7;
        KADRBB-6;
        KADRBB-5;
        KADRBB-4;
        KADRBB-3;
        KADRBB-2;
        KADRBB-1;
        KADRBB-0#
```

14.1 MA (Continued)

```
RAMKEY 30;      % KEY-RAM INPUT. 15 BITS OF ECC FOLLOWED
                % BY 15 BITS OF KEY DATA
        KYECIN-14;
        KYECIN-13;
        KYECIN-12;
        KYECIN-11;
        KYECIN-10;
        KYECIN-9;
        KYECIN-8;
        KYECIN-7;
        KYECIN-6;
        KYECIN-5;
        KYECIN-4;
        KYECIN-3;
        KYECIN-2;
        KYECIN-1;
        KYECIN-0;
        KYRMIN-14;
        KYRMIN-13;
        KYRMIN-12;
        KYRMIN-11;
        KYRMIN-10;
        KYRMIN-9;
        KYRMIN-8;
        KYRMIN-7;
        KYRMIN-6;
        KYRMIN-5;
        KYRMIN-4;
        KYRMIN-3;
        KYRMIN-2;
        KYRMIN-1;
        KYRMIN-0#
```

14.1 MA (Continued)

```
KEYECC 15;          % KEY RAM ECC INPUT.
        KYECIN-14;
        KYECIN-13;
        KYECIN-12;
        KYECIN-11;
        KYECIN-10;
        KYECIN-9;
        KYECIN-8;
        KYECIN-7;
        KYECIN-6;
        KYECIN-5;
        KYECIN-4;
        KYECIN-3;
        KYECIN-2;
        KYECIN-1;
        KYECIN-0#

KEYDAT 15;      % KEY-RAM DATA INPUT.
        KYRMIN-14;
        KYRMIN-11;
        KYRMIN-8;
        KYRMIN-5;
        KYRMIN-2;
        KYRMIN-13;
        KYRMIN-10;
        KYRMIN-7;
        KYRMIN-4;
        KYRMIN-1;
        KYRMIN-12;
        KYRMIN-9;
        KYRMIN-6;
        KYRMIN-3;
        KYRMIN-0#

RMADHA 10;      % HIGH WORD ADDRESS COPYA OF DATA RAM
        DRAMHA-9;
        DRAMHA-8;
        DRAMHA-7;
        DRAMHA-6;
        DRAMHA-5;
        DRAMHA-4;
        DRAMHA-3;
        DRAMHA-2;
        DRAMHA-1;
        DRAMHA-0#
```

14.1 MA (Continued)

```
RMADHB 10;      % HIGH WORD ADDRESS COPYB OF DATA RAM
        DRAMHB-9;
        DRAMHB-8;
        DRAMHB-7;
        DRAMHB-6;
        DRAMHB-5;
        DRAMHB-4;
        DRAMHB-3;
        DRAMHB-2;
        DRAMHB-1;
        DRAMHB-0#

RMADLA 10;      % LOW WORD ADDRESS COPYA OF DATA RAM
        DRAMLA-9;
        DRAMLA-8;
        DRAMLA-7;
        DRAMLA-6;
        DRAMLA-5;
        DRAMLA-4;
        DRAMLA-3;
        DRAMLA-2;
        DRAMLA-1;
        DRAMLA-0#

RMADLB 10;      % LOW WORD ADDRESS COPYB OF DATA RAM
        DRAMLB-9;
        DRAMLB-8;
        DRAMLB-7;
        DRAMLB-6;
        DRAMLB-5;
        DRAMLB-4;
        DRAMLB-3;
        DRAMLB-2;
        DRAMLB-1;
        DRAMLB-0#

ROTLEN 4;       % LENGTH TO ROTCAT
        TRANLEN-3;
        TRANLEN-2;
        TRANLEN-1;
        TRANLEN-0#
```

14.1 MA (Continued)

```
ROTLSB 2;       % LSB TO ROTCAT
        TRANLSB-1;
        TRANLSB-0#

ROTLSD 4;       % LSD TO ROTCAT
        TRANLSD-3;
        TRANLSD-2;
        TRANLSD-1;
        TRANLSD-0#

SPYKEY 15;      % SPY1 KEY INPUT
        S1KEY-14;
        S1KEY-13;
        S1KEY-12;
        S1KEY-11;
        S1KEY-10;
        S1KEY-9;
        S1KEY-8;
        S1KEY-7;
        S1KEY-6;
        S1KEY-5;
        S1KEY-4;
        S1KEY-3;
        S1KEY-2;
        S1KEY-1;
        S1KEY-0#

SPY1AD 10;      % SPY1 ADDRESS
        S1ADR-9;
        S1ADR-8;
        S1ADR-7;
        S1ADR-6;
        S1ADR-5;
        S1ADR-4;
        S1ADR-3;
        S1ADR-2;
        S1ADR-1;
        S1ADR-0#
```

14.1 MA (Continued)

```
SPY2AD 10;      % SPY2 ADDRESS
        S2ADR-9;
        S2ADR-8;
        S2ADR-7;
        S2ADR-6;
        S2ADR-5;
        S2ADR-4;
        S2ADR-3;
        S2ADR-2;
        S2ADR-1;
        S2ADR-0#

SSPYFG 4;       % SLAVE SPY FLAG
        SSPYFG4;
        SSPYFG3;
        SSPYFG2;
        SSPYFG1#

STRFG 4;        % SLAVE PROCESSOR FLAG
        STRFG4;
        STRFG3;
        STRFG2;
        STRFG1#

MSPYFG 4;       % MASTER SPY FLAG
        MSPYFG4;
        MSPYFG3;
        MSPYFG2;
        MSPYFG1#

MTRFG 4;        % MASTER PROCESSOR FLAG
        MTRFG4;
        MTRFG3;
        MTRFG2;
        MTRFG1#

FLAG1 2;
        MTRFG4;
        MTRFG3#
FLAG0 2;
        MTRFG2;
        MTRFG1#
```

14.1 MA (Continued)

```
%
% MASTER SPY PIPELINE
%

        MS1CMDVAL 1;    % MASTER SPY PIPELINE 1 COMMAND VALID
                MS1COMDVAL#

        MS1CMD 2;       % MASTER SPY1 COMMAND.
                MS1CMD-1;
                MS1CMD-0#

        MS2CMDVAL 1;    % MASTER SPY PIPELINE 2 COMMAND VALID
                MS2COMDVAL#

        MS2CMD 2;       % MASTER SPY2 COMMAND.
                        % SPY2 VALID BIT FOLLOWED BY 2 BITS COMMAND
                MS2CMD-1;
                MS2CMD-0#

%
% MASTER TRANSLATE PIPELINE
%

        MT1CMDPAR 1;    % MASTER TRANSLATE PIPELINE 1 COMMAND PARITY
                MT1COMDPAR#

        MT1CMD 5;       % MASTER TRAN1 COMMAND.
                        % TRAN1 5 BITS COMMAND FOLLOWED BY PARITY BIT
                MT1CMD-4;
                MT1CMD-3;
                MT1CMD-2;
                MT1CMD-1;
                MT1CMD-0#

        MT2CMDPAR 1;    % MASTER TRANSLATE PIPELINE 2 COMMAND PARITY
                MT2COMDPAR#

        MT2CMD 5;       % MASTER TRAN2 COMMAND.
                        % TRAN2 5 BITS COMMAND FOLLOWED BY PARITY BIT
                MT2CMD-4;
                MT2CMD-3;
                MT2CMD-2;
                MT2CMD-1;
```

14.1 MA (Continued)

```
        MT3CMDPAR 1;    % MASTER TRANSLATE PIPELINE 3 COMMAND PARITY
                MT3COMDPAR#

        MT3CMD 5;       % MASTER TRAN3 COMMAND.
                        % TRAN3 5 BITS COMMAND FOLLOWED BY PARITY BIT
                MT3CMD-4;
                MT3CMD-3;
                MT3CMD-2;
                MT3CMD-1;
                MT3CMD-0#

%
% SLAVE SPY PIPELINE
%

        SS1CMDVAL 1;    % SLAVE SPY PIPELINE 1 COMMAND VALID.
                SS1COMDVAL#

        SS1CMD 2;       % SLAVE SPY PIPELINE 1 COMMAND.
                SS1CMD-1;
                SS1CMD-0#

        SS2CMDVAL 1;    % SLAVE SPY PIPELINE 2 COMMAND VALID.
                SS2COMDVAL#

        SS2CMD 2;       % SLAVE SPY PIPELINE 2 COMMAND.
                SS2CMD-1;
                SS2CMD-0#

%
% SLAVE TRANSLATE PIPELINE
%

        ST1CMDPAR 1;    % SLAVE TRANSLATE PIPELINE 1 COMMAND PARITY
                ST1COMDPAR#

        ST1CMD 5;       % SLAVE TRAN1 COMMAND.
                ST1CMD-4;
                ST1CMD-3;
                ST1CMD-2;
                ST1CMD-1;
                ST1CMD-0#
```

14.1 MA (Continued)

```
        ST2CMDPAR 1;    % SLAVE TRANSLATE PIPELINE 2 COMMAND PARITY
                ST2COMDPAR#

        ST2CMD 5;       % SLAVE TRAN2 COMMAND.
                ST2CMD-4;
                ST2CMD-3;
                ST2CMD-2;
                ST2CMD-1;
                ST2CMD-0#

        ST3CMDPAR 1;    % SLAVE TRANSLATE PIPELINE 3 COMMAND PARITY
                ST3COMDPAR#

        ST3CMD 5;       % SLAVE TRAN3 COMMAND.
                ST3CMD-4;
                ST3CMD-3;
                ST3CMD-2;
                ST3CMD-1;
                ST3CMD-0#

%
% TRANSLATED ADDRESS PIPELINE
%
        TR1LSD 4;       % LSD FIELD IN PIPELINE T1
                T1LSD-3;
                T1LSD-2;
                T1LSD-1;
                T1LSD-0#

        TR1LEN 4;       % LSD FIELD IN PIPELINE T1
                T1LEN-3;
                T1LEN-2;
                T1LEN-1;
                T1LEN-0#

        TR1TAG 8;       % PIPELINE #1 TAG FIELD
                T1TAG-7;
                T1TAG-6;
                T1TAG-5;
                T1TAG-4;
                T1TAG-3;
                T1TAG-2;
                T1TAG-1;
                T1TAG-0#
```

14.1 MA (Continued)

```
TR1KEY 15;      % BINARY KEY ADDRESS FIELD IN PIPELINE T1
       T1ADR-26;
       T1ADR-25;
       T1ADR-24;
       T1ADR-23;
       T1ADR-22;
       T1ADR-21;
       T1ADR-20;
       T1ADR-19;
       T1ADR-18;
       T1ADR-17;
       T1ADR-16;
       T1ADR-15;
       T1ADR-14;
       T1ADR-13;
       T1ADR-12#

TR1SET 10;      % BINARY SET ADDRESS IN PIPELINE T1
       T1ADR-11;
       T1ADR-10;
       T1ADR-9;
       T1ADR-8;
       T1ADR-7;
       T1ADR-6;
       T1ADR-5;
       T1ADR-4;
       T1ADR-3;
       T1ADR-2#
```

14.1 MA (Continued)

```
TR1BLK 25;      % BINARY BLOCK ADDRESS OF PIPELINE 1
        T1ADR-26;
        T1ADR-25;
        T1ADR-24;
        T1ADR-23;
        T1ADR-22;
        T1ADR-21;
        T1ADR-20;
        T1ADR-19;
        T1ADR-18;
        T1ADR-17;
        T1ADR-16;
        T1ADR-15;
        T1ADR-14;
        T1ADR-13;
        T1ADR-12;
        T1ADR-11;
        T1ADR-10;
        T1ADR-9;
        T1ADR-8;
        T1ADR-7;
        T1ADR-6;
        T1ADR-5;
        T1ADR-4;
        T1ADR-3;
        T1ADR-2#

TR1LSB 2;       % LSB FIELD IN PIPELINE T1
        T1LSB-1;
        T1LSB-0#
```

14.1 MA (Continued)

```
TR1ADR 27;      % COMPLETE BINARY ADDRESS FIELD IN PIPELINE T1
        T1ADR-26;
        T1ADR-25;
        T1ADR-24;
        T1ADR-23;
        T1ADR-22;
        T1ADR-21;
        T1ADR-20;
        T1ADR-19;
        T1ADR-18;
        T1ADR-17;
        T1ADR-16;
        T1ADR-15;
        T1ADR-14;
        T1ADR-13;
        T1ADR-12;
        T1ADR-11;
        T1ADR-10;
        T1ADR-9;
        T1ADR-8;
        T1ADR-7;
        T1ADR-6;
        T1ADR-5;
        T1ADR-4;
        T1ADR-3;
        T1ADR-2;
        T1LSB-1;
        T1LSB-0#

TR2LSD 4;       % LSD FIELD IN PIPELINE T2
        T2LSD-3;
        T2LSD-2;
        T2LSD-1;
        T2LSD-0#

TR2LEN 4;       % LSD FIELD IN PIPELINE T2
        T2LEN-3;
        T2LEN-2;
        T2LEN-1;
        T2LEN-0#
```

14.1 MA (Continued)

```
TR2TAG 8;       % PIPELINE #2 TAG FIELD
        T2TAG-7;
        T2TAG-6;
        T2TAG-5;
        T2TAG-4;
        T2TAG-3;
        T2TAG-2;
        T2TAG-1;
        T2TAG-0#

TR2KEY 15;      % BINARY KEY FIELD IN PIPELINE T2
        T2ADR-26;
        T2ADR-25;
        T2ADR-24;
        T2ADR-23;
        T2ADR-22;
        T2ADR-21;
        T2ADR-20;
        T2ADR-19;
        T2ADR-18;
        T2ADR-17;
        T2ADR-16;
        T2ADR-15;
        T2ADR-14;
        T2ADR-13;
        T2ADR-12#

TR2SET 10;      % BINARY SET ADDRESS OF PIPELINE T2
        T2ADR-11;
        T2ADR-10;
        T2ADR-9;
        T2ADR-8;
        T2ADR-7;
        T2ADR-6;
        T2ADR-5;
        T2ADR-4;
        T2ADR-3;
        T2ADR-2#
```

14.1 MA (Continued)

```
TR2BLK 25;      % BINARY BLOCK ADDRESS FIELD IN PIPELINE T2
       T2ADR-26;
       T2ADR-25;
       T2ADR-24;
       T2ADR-23;
       T2ADR-22;
       T2ADR-21;
       T2ADR-20;
       T2ADR-19;
       T2ADR-18;
       T2ADR-17;
       T2ADR-16;
       T2ADR-15;
       T2ADR-14;
       T2ADR-13;
       T2ADR-12;
       T2ADR-11;
       T2ADR-10;
       T2ADR-9;
       T2ADR-8;
       T2ADR-7;
       T2ADR-6;
       T2ADR-5;
       T2ADR-4;
       T2ADR-3;
       T2ADR-2#

TR2LSB 2;       % LSB FIELD IN PIPELINE T2
       T2LSB-1;
       T2LSB-0#
```

14.1 MA (Continued)

```
TR2ADR 27;      % ADDRESS FIELD IN PIPELINE T2
        T2ADR-26;
        T2ADR-25;
        T2ADR-24;
        T2ADR-23;
        T2ADR-22;
        T2ADR-21;
        T2ADR-20;
        T2ADR-19;
        T2ADR-18;
        T2ADR-17;
        T2ADR-16;
        T2ADR-15;
        T2ADR-14;
        T2ADR-13;
        T2ADR-12;
        T2ADR-11;
        T2ADR-10;
        T2ADR-9;
        T2ADR-8;
        T2ADR-7;
        T2ADR-6;
        T2ADR-5;
        T2ADR-4;
        T2ADR-3;
        T2ADR-2;
        T2LSB-1;
        T2LSB-0#

TR3LSD 4;       % LSD FIELD IN PIPELINE T3
        T3LSD-3;
        T3LSD-2;
        T3LSD-1;
        T3LSD-0#

TR3LEN 4;       % LSD FIELD IN PIPELINE T3
        T3LEN-3;
        T3LEN-2;
        T3LEN-1;
        T3LEN-0#
```

14.1 MA (Continued)

```
TR3TAG 8;       % PIPELINE #3 TAG FIELD
        T3TAG-7;
        T3TAG-6;
        T3TAG-5;
        T3TAG-4;
        T3TAG-3;
        T3TAG-2;
        T3TAG-1;
        T3TAG-0#

TR3KEY 15;      % BINARY KEY ADDRESS FIELD IN PIPELINE T3
        T3ADR-26;
        T3ADR-25;
        T3ADR-24;
        T3ADR-23;
        T3ADR-22;
        T3ADR-21;
        T3ADR-20;
        T3ADR-19;
        T3ADR-18;
        T3ADR-17;
        T3ADR-16;
        T3ADR-15;
        T3ADR-14;
        T3ADR-13;
        T3ADR-12#

TR3SET 10;      % BINARY SET ADDRESS OF PIPELINE T2
        T3ADR-11;
        T3ADR-10;
        T3ADR-9;
        T3ADR-8;
        T3ADR-7;
        T3ADR-6;
        T3ADR-5;
        T3ADR-4;
        T3ADR-3;
        T3ADR-2#
```

14.1 MA (Continued)

```
TR3BLK 25;      % BINARY BLOCK ADDRESS FIELD IN PIPELINE T3
       T3ADR-26;
       T3ADR-25;
       T3ADR-24;
       T3ADR-23;
       T3ADR-22;
       T3ADR-21;
       T3ADR-20;
       T3ADR-19;
       T3ADR-18;
       T3ADR-17;
       T3ADR-16;
       T3ADR-15;
       T3ADR-14;
       T3ADR-13;
       T3ADR-12;
       T3ADR-11;
       T3ADR-10;
       T3ADR-9;
       T3ADR-8;
       T3ADR-7;
       T3ADR-6;
       T3ADR-5;
       T3ADR-4;
       T3ADR-3;
       T3ADR-2#

TR3LSB 2;       % LSB FIELD IN PIPELINE T3
       T3LSB-1;
       T3LSB-0#
```

14.1 MA (Continued)

```
        TR3ADR 27;      % COMPLETE BINARY ADDRESS FIELD IN PIPELINE T3
                T3ADR-26;
                T3ADR-25;
                T3ADR-24;
                T3ADR-23;
                T3ADR-22;
                T3ADR-21;
                T3ADR-20;
                T3ADR-19;
                T3ADR-18;
                T3ADR-17;
                T3ADR-16;
                T3ADR-15;
                T3ADR-14;
                T3ADR-13;
                T3ADR-12;
                T3ADR-11;
                T3ADR-10;
                T3ADR-9;
                T3ADR-8;
                T3ADR-7;
                T3ADR-6;
                T3ADR-5;
                T3ADR-4;
                T3ADR-3;
                T3ADR-2;
                T3LSB-1;
                T3LSB-0#

%
% A&W INPUT QUEUE REGISTERS
%

        WQREGVALID 1;   % QUEUE REGISTER VALID F/F
                QUEFULL#

        WFULL 1;        % AW QUEUE IS FULL
                NOBUS#
```

14.1 MA (Continued)

```
WQUEREG 19;     % ENTIRE QUEUE REGISTER
        TAGNU13:8;
        WTAGPARITY;
        WTAG(6:0);
        CMDNU;
        WCOMDPAR;
        WCOMMAND;
        WLENGTH;
        HIADRNU;
        WADRPAR;
        WBASEIND;
        WADR35:28;
        WADR27:14;
        WADR13:4;
        WADR3:0;
        WDATNU;
        WDATPAR;
        WDAT39:28;
        WDAT27:14;
        WDAT13:0#

WCMDPAR 2;      % AW INPUT QUEUE COMMAND PARITY BIT
        CMDNU;          % NOT USED FIELD: FORCE TO "0"
        WCOMDPAR#

WCMD 1;         % AW INPUT QUEUE COMMAND FIELD
        WCOMMAND#

WADDRPAR 2;     % AW INPUT QUEUE ADDRESS PARITY BIT
        HIADRNU;        % NOT USED FIELD: FORCE TO "0"
        WADRPAR#

WBI 1;          % AW INPUT QUEUE BASE INDICANT FIELD
        WBASEIND#

WADDR 4;        % AW INPUT QUEUE ADDRESS FIELD
        WADR35:28;
        WADR27:14;
        WADR13:4;
        WADR3:0#
```

14.1 MA (Continued)

```
        WLEN 1;          % AW INPUT QUEUE LENGTH FIELD
                WLENGTH#

        WDATAPAR 2;      % AW INPUT QUEUE DATA PARITY BIT
                WDATNU;          % NOT USED FIELD: FORCE TO "0"
                WDATPAR#

        WDATA 3;         % AW INPUT QUEUE DATA FIELD
                WDAT39:28;
                WDAT27:14;
                WDAT13:0#

        WTAGPAR 2;       % AW INPUT QUEUE TAG PARITY BIT
                TAGNU13:8;       % NOT USED FIELD: FORCE TO "0"
                WTAGPARITY#

        WTAG 1;          % AW INPUT QUEUE TAG FIELD
                WTAG(6:0)#
%
% MEMORY CONTROL REGISTERS
%
        WINID 3;         % THE ID OF THE MEMORY BUS ARBITRATION WINNER
                WINID-2;
                WINID-1;
                WINID-0#

        % THE FOLLOWING 4 REGISTERS ARE READ-ONLY.
        % THEY ARE CAPTURED WHEN MAARBERR = 1.

        LONG 1;          % BUSY OF THE MODULE SELECTED BY REQUEST ADDRES
                TRAFFIC#

        NOWREQ 1;        % MODULE IS CURRENTLY BEING REQUESTED ON B.P.
                WINID-2#

        WASREQ 1;        % MODULE WAS REQUESTED ON B.P. ON LAST BUS CYCL
                WINID-1#

        BUSBUSY 1;       % RETURNING READ DATA CONFLICTS WITH WRITE ON B
                WINID-0#
```

14.1 MA (Continued)

```
%
% MAIN MEMORY BUS INTERFACE REGISTERS
%
        ITSMOD 2;       % DATA ON MAIN MEMORY BACKPLANE WAS MODIFIED
                MC2BZ-03;
                MC2BZ-02#

        ITSCOPY 2;      % DATA ON MAIN MEMORY BACKPLANE WAS A COPY
                MC2BZ-01;
                MC2BZ-00#

        XCMD    2;      % COMMAND PORTION OF XMTADR
                XMTADR-31;
                XMTADR-30#

        XREQID 3;       % REQUESTOR ID PORTION OF XMTADR
                XMTADR-29;
                XMTADR-28;
                XMTADR-27#

        XMODSEL 5;      % MODULE SELECT PORTION OF XMTADR
                XMTADR-26;
                XMTADR-25;
                XMTADR-24;
                XMTADR-23;
                XMTADR-22#

        XADR    7;      % ADDRESS TO BE XMT'D ON MEM. ADDR. BUS
                XMTADR-21;
                XMTADR-20;
                XMTADR-19;
                XMTADR-18;
                XMTADR-17;
                XMTADR-16;
                XMTADR15:0#

        XECC    4;      % ECC TO BE XMT'D ON MEM. ADDR. BUS
                XMTECC-6;
                XMTECC-5;
                XMTECC-4;
                XMTECC3:0#
```

14.1 MA (Continued)

```
XMTADR  21;     % ADDRESS AND ECC TO BE XMT'D ON MEM. ADDR. BUS
        XMTECC-6;
        XMTECC-5;
        XMTECC-4;
        XMTECC3:0;
        XMTADR-31;
        XMTADR-30;
        XMTADR-29;
        XMTADR-28;
        XMTADR-27;
        XMTADR-26;
        XMTADR-25;
        XMTADR-24;
        XMTADR-23;
        XMTADR-22;
        XMTADR-21;
        XMTADR-20;
        XMTADR-19;
        XMTADR-18;
        XMTADR-17;
        XMTADR-16;
        XMTADR15:0#

RCMD    2;      % COMMAND PORTION OF RCVADR
        RCVADR-31;
        RCVADR-30#

RREQID  3;      % REQUESTOR ID PORTION OF RCVADR
        RCVADR-29;
        RCVADR-28;
        RCVADR-27#

RMODSEL 5;      % MODULE SELECT PORTION OF RCVADR
        RCVADR-26;
        RCVADR-25;
        RCVADR-24;
        RCVADR-23;
        RCVADR-22#
```

14.1 MA (Continued)

```
RADR    7;      % ADDRESS PORTION OF RCVADR
        RCVADR-21;
        RCVADR-20;
        RCVADR-19;
        RCVADR-18;
        RCVADR-17;
        RCVADR-16;
        RCVADR15:0#

RECC    4;      % ECC PORTION OF RCVADR
        RCVECC-6;
        RCVECC-5;
        RCVECC-4;
        RCVECC3:0#

RCVADR  21;     % ADDRESS AND ECC RECV'D FROM MEM. ADDR. BUS
        RCVECC-6;
        RCVECC-5;
        RCVECC-4;
        RCVECC3:0;
        RCVADR-31;
        RCVADR-30;
        RCVADR-29;
        RCVADR-28;
        RCVADR-27;
        RCVADR-26;
        RCVADR-25;
        RCVADR-24;
        RCVADR-23;
        RCVADR-22;
        RCVADR-21;
        RCVADR-20;
        RCVADR-19;
        RCVADR-18;
        RCVADR-17;
        RCVADR-16;
        RCVADR15:0#
```

14.1 MA (Continued)

```
%
% START OF REGISTER DEFINITIONS FOR LARRY'S RAMDRY USEAGE
%
        MJUNK2 3;
                MC2BZ-05;
                MC2BZ-04;
                MC2BZ-07#

        JUNK4 3;
                SSPAREF;
                MORDERF;
                MSPAREF#

        JUNK5 23;
                T2ADR-26;
                T2ADR-25;
                T2ADR-24;
                T2ADR-23;
                T2ADR-22;
                T2ADR-21;
                T2ADR-20;
                T2ADR-19;
                T2ADR-18;
                T2ADR-17;
                T2ADR-16;
                T2ADR-15;
                T2ADR-14;
                T2ADR-13;
                T2ADR-12;
                MTRFG4;
                MTRFG3;
                MTRFG2;
                MTRFG1;
                STRFG4;
                STRFG3;
                STRFG2;
                STRFG1#

        JM 2;
                MJM;
                SJM#
```

14.1 MA (Continued)

```
JW 2;
        MJW;
        SJW#

TRNHIT 2;
        MT1HITFL;
        MT0HITFL#

SPYHIT 2;
        MS1HIT;
        MS0HIT#

RBUSERR 1;
        RBUSERVEC#

ERRPIP1 1;
        EVPIPE1#

ERRPIP2 1;
        EVPIPE2#

ERRPIP3 1;
        EVPIPE3#

CDOEN 2;
        MCDOEN;
        SCDOEN#

CEOEN 2;
        MCEOEN;
        SCEOEN#

CDOSL 2;
        MCDOSL;
        SCDOSL#

PF2 2;
        MPF2;
        SPF2#

LSBENL 2;
        MLSBENL;
        SLSBENL#
```

14.1 MA (Continued)

```
        LRUWF 2;
                MLRUWF;
                SLRUWF#
%
% END OF REGISTER DEFINITIONS FOR LARRY'S RAMDRY USEAGE
%

      WQUE  21;      % AW QUEUE REGISTER DEFINED FOR RAMDRY.
                     % EASY TO READ FORMAT
                NOTUSED3E;
                WTAGPARITY;
                NOTUSED1A;
                WTAG(6:0);

                NOTUSED3D;
                WCOMDPAR;
                NOTUSED3C;
                WCOMMAND;
                WLENGTH;

                NOTUSED3B;
                WADRPAR;
                WBASEIND;
                WADR35:28;
                WADR27:14;
                WADR13:4;
                WADR3:0;

                NOTUSED3A;
                WDATPAR;
                WDAT39:28;
                WDAT27:14;
                WDAT13:0#
```

14.1 MA (Continued)

```
        ERROR  23;              % accumulation of MCACM errors & brokens
               BROKEN;          % MCACM A BROKEN SIGNAL
               UCONFIGERR;      % invalid memory data card type
               BPCTLERR;        % error in intercache backplane signals
               ARBERR;          % error in memory backplane arbitration
               MCONFIGERR;      % invalid memory data card type
               CHADR5ERR;       % CHADDR #5 error
               CHADR4ERR;       % CHADDR #4 error
               CHADR3ERR;       % CHADDR #3 error
               CHADR2ERR;       % CHADDR #2 error
               CHADR1ERR;       % CHADDR #1 error
               SFLGERROR;       % slave FLAG ARRAY ERROR
               MFLGERROR;       % master FLAG ARRAY ERROR
               SCNTLERR;        % slave CACNTL ARRAY IN ERROR
               MCNTLERR;        % master CACNTL ARRAY IN ERROR
               MULTERROR1;      % THE ERROR LATCHED IS MULTIPLE ERROR.
               SNGLERROR1;      % THE ERROR LATCHED IS SINGLE ERROR.
               RDATAPARER;      % CORRECTED DATA PARITY ERROR.
               MULTERROR0;      % THE ERROR LATCHED IS MULTIPLE ERROR.
               SNGLERROR0;      % THE ERROR LATCHED IS SINGLE ERROR.
               BROKEN-B;        % CACHE MULTIPLE ERROR BROKEN HIDDEN STA
               BROKEN-E;        % CACHE WRITE BACK BROKEN HIDDEN STATE.
               BROKEN-A;        % MBUS MULTIBLE ERROR BROKEN HIDDEN STAT
               QCTRLBRK#        % AW QUEUE CONTROL HAS DETECTED A FAULT
                                % This will not cause BROKEN.
REGEND;
```

14.2 MD

```
CHAIN     MD;                    % MCACM DATA CARD CHAIN.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT             LOCATION: A8I2
% DESCRIPTION: WORD 3  BIT 8     SDI:R11  SDO:D13  SHIFT:B12  CLEAR:H15
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW3-39  1; % MEMORY BUS DAT INPUT.
    MDATIW3-35  1;
    MDATIW3-31  1;
    MDATIW3-27  1;
    MDATIW3-23  1;
    MDATIW3-19  1;
    MDATIW3-15  1;
    MDATIW3-11  1;
    MDATIW3-07  1;
    MDATIW3-03  1;


    MDATOW3-39  1; % MEMORY BUS DAT OUTPUT.
    MDATOW3-35  1;
    MDATOW3-31  1;
    MDATOW3-27  1;
    MDATOW3-23  1;
    MDATOW3-19  1;
    MDATOW3-15  1;
    MDATOW3-11  1;
    MDATOW3-07  1;
    MDATOW3-03  1;

    NOTUSED-55  1;  % WAS RBUS VALID.

    RBUSREG-39  1;  % RBUS.
    RBUSREG-31  1;
    RBUSREG-30  1;

    WECCB1W3-7  1;  % WRITE ECC.
    WECCB1W3-6  1;
    WECCB0W3-7  1;
    WECCB0W3-6  1;
```

14.2 MD (Continued)

```
    PCREG-39    1;  % PC.
    PCREG-31    1;
    PCREG-30    1;

    WDPIPE2-39  1;  % MAW WRITE DATA PIPELINE STAGE 2.
    WDPIPE2-31  1;
    WDPIPE2-30  1;

    WDPIPE3-39  1;  % MAW WRITE DATA PIPELINE STAGE 3.
    WDPIPE3-31  1;
    WDPIPE3-30  1;

    WDPIPE1-39  1;  % MAW WRITE DATA PIPELINE STAGE 1.
    WDPIPE1-31  1;
    WDPIPE1-30  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       CHIP: ROTCAT          LOCATION: A8K6
% DESCRIPTION: WORD 3  BIT 4    SDI:P6  SDO:M15  SHIFT:B10  CLEAR:H15  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW3-38  1;
    MDATIW3-34  1;
    MDATIW3-30  1;
    MDATIW3-26  1;
    MDATIW3-22  1;
    MDATIW3-18  1;
    MDATIW3-14  1;
    MDATIW3-10  1;
    MDATIW3-06  1;
    MDATIW3-02  1;

    MDATOW3-38  1;
    MDATOW3-34  1;
    MDATOW3-30  1;
    MDATOW3-26  1;
    MDATOW3-22  1;
    MDATOW3-18  1;
    MDATOW3-14  1;
    MDATOW3-10  1;
    MDATOW3-06  1;
    MDATOW3-02  1;
```

14.2 MD (Continued)

```
NOTUSED-54  1;

RBUSREG-38  1;
RBUSREG-29  1;
RBUSREG-28  1;

WECCB1W3-5  1;
WECCB1W3-4  1;
WECCB0W3-5  1;
WECCB0W3-4  1;

PCREG-38    1;
PCREG-29    1;
PCREG-28    1;

WDPIPE2-38  1;
WDPIPE2-29  1;
WDPIPE2-28  1;

WDPIPE3-38  1;
WDPIPE3-29  1;
WDPIPE3-28  1;

WDPIPE1-38  1;
WDPIPE1-29  1;
WDPIPE1-28  1;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT            LOCATION: D2I2                                %
% DESCRIPTION: WORD 3  BIT 2    SDI:P6  SDO:M15  SHIFT:B10  CLEAR:H15  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    MDATIW3-37  1;
    MDATIW3-33  1;
    MDATIW3-29  1;
    MDATIW3-25  1;
    MDATIW3-21  1;
    MDATIW3-17  1;
    MDATIW3-13  1;
    MDATIW3-09  1;
    MDATIW3-05  1;
    MDATIW3-01  1;

    MDATOW3-37  1;
    MDATOW3-33  1;
    MDATOW3-29  1;
    MDATOW3-25  1;
    MDATOW3-21  1;
    MDATOW3-17  1;
    MDATOW3-13  1;
    MDATOW3-09  1;
    MDATOW3-05  1;
    MDATOW3-01  1;

    NOTUSED-53  1;

    RBUSREG-37  1;
    RBUSREG-27  1;
    RBUSREG-26  1;

    WECCB1W3-3  1;
    WECCB1W3-2  1;
    WECCB0W3-3  1;
    WECCB0W3-2  1;

    PCREG-37    1;
    PCREG-27    1;
    PCREG-26    1;
```

14.2 MD (Continued)

```
    WDPIPE2-37  1;
    WDPIPE2-27  1;
    WDPIPE2-26  1;

    WDPIPE3-37  1;
    WDPIPE3-27  1;
    WDPIPE3-26  1;

    WDPIPE1-37  1;
    WDPIPE1-27  1;
    WDPIPE1-26  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT           LOCATION: D2K6                                 %
% DESCRIPTION: WORD 3  BIT 1    SDI:P6  SDO:M15  SHIFT:B10  CLEAR:H15           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW3-36  1;
    MDATIW3-32  1;
    MDATIW3-28  1;
    MDATIW3-24  1;
    MDATIW3-20  1;
    MDATIW3-16  1;
    MDATIW3-12  1;
    MDATIW3-08  1;
    MDATIW3-04  1;
    MDATIW3-00  1;

    MDATOW3-36  1;
    MDATOW3-32  1;
    MDATOW3-28  1;
    MDATOW3-24  1;
    MDATOW3-20  1;
    MDATOW3-16  1;
    MDATOW3-12  1;
    MDATOW3-08  1;
    MDATOW3-04  1;
    MDATOW3-00  1;

    NOTUSED-52  1;
```

14.2 MD (Continued)

```
    RBUSREG-36  1;
    RBUSREG-25  1;
    RBUSREG-24  1;

    WECCB1W3-1  1;
    WECCB1W3-0  1;
    WECCB0W3-1  1;
    WECCB0W3-0  1;

    PCREG-36    1;
    PCREG-25    1;
    PCREG-24    1;

    WDPIPE2-36  1;
    WDPIPE2-25  1;
    WDPIPE2-24  1;

    WDPIPE3-36  1;
    WDPIPE3-25  1;
    WDPIPE3-24  1;

    WDPIPE1-36  1;
    WDPIPE1-25  1;
    WDPIPE1-24  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       CHIP: ROTCAT          LOCATION: F6I2
% DESCRIPTION: WORD 2  BIT 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW2-39  1;
    MDATIW2-35  1;
    MDATIW2-31  1;
    MDATIW2-27  1;
    MDATIW2-23  1;
    MDATIW2-19  1;
    MDATIW2-15  1;
    MDATIW2-11  1;
    MDATIW2-07  1;
    MDATIW2-03  1;
    MDATOW2-39  1;
    MDATOW2-35  1;
    MDATOW2-31  1;
```

14.2 MD (Continued)

```
MDATOW2-27  1;
MDATOW2-23  1;
MDATOW2-19  1;
MDATOW2-15  1;
MDATOW2-11  1;
MDATOW2-07  1;
MDATOW2-03  1;

NOTUSED-51  1;

RBUSREG-35  1;
RBUSREG-23  1;
RBUSREG-22  1;

WECCB1W2-7  1;
WECCB1W2-6  1;
WECCB0W2-7  1;
WECCB0W2-6  1;

PCREG-35    1;
PCREG-23    1;
PCREG-22    1;

WDPIPE2-35  1;
WDPIPE2-23  1;
WDPIPE2-22  1;

WDPIPE3-35  1;
WDPIPE3-23  1;
WDPIPE3-22  1;

WDPIPE1-35  1;
WDPIPE1-23  1;
WDPIPE1-22  1;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT          LOCATION: F6K6
% DESCRIPTION: WORD 2  BIT 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW2-38  1;
    MDATIW2-34  1;
    MDATIW2-30  1;
    MDATIW2-26  1;
    MDATIW2-22  1;
    MDATIW2-18  1;
    MDATIW2-14  1;
    MDATIW2-10  1;
    MDATIW2-06  1;
    MDATIW2-02  1;

    MDATOW2-38  1;
    MDATOW2-34  1;
    MDATOW2-30  1;
    MDATOW2-26  1;
    MDATOW2-22  1;
    MDATOW2-18  1;
    MDATOW2-14  1;
    MDATOW2-10  1;
    MDATOW2-06  1;
    MDATOW2-02  1;

    NOTUSED-50  1;

    RBUSREG-34  1;
    RBUSREG-21  1;
    RBUSREG-20  1;

    WECCB1W2-5  1;
    WECCB1W2-4  1;
    WECCB0W2-5  1;
    WECCB0W2-4  1;

    PCREG-34    1;
    PCREG-21    1;
    PCREG-20    1;
```

14.2 MD (Continued)

```
    WDPIPE2-34  1;
    WDPIPE2-21  1;
    WDPIPE2-20  1;

    WDPIPE3-34  1;
    WDPIPE3-21  1;
    WDPIPE3-20  1;

    WDPIPE1-34  1;
    WDPIPE1-21  1;
    WDPIPE1-20  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT          LOCATION: I0I2
% DESCRIPTION: WORD 2  BIT 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW2-37  1;
    MDATIW2-33  1;
    MDATIW2-29  1;
    MDATIW2-25  1;
    MDATIW2-21  1;
    MDATIW2-17  1;
    MDATIW2-13  1;
    MDATIW2-09  1;
    MDATIW2-05  1;
    MDATIW2-01  1;

    MDATOW2-37  1;
    MDATOW2-33  1;
    MDATOW2-29  1;
    MDATOW2-25  1;
    MDATOW2-21  1;
    MDATOW2-17  1;
    MDATOW2-13  1;
    MDATOW2-09  1;
    MDATOW2-05  1;
    MDATOW2-01  1;

    NOTUSED-49  1;

    RBUSREG-33  1;
    RBUSREG-19  1;
    RBUSREG-18  1;
```

14.2 MD (Continued)

```
    WECCB1W2-3  1;
    WECCB1W2-2  1;
    WECCB0W2-3  1;
    WECCB0W2-2  1;

    PCREG-33    1;
    PCREG-19    1;
    PCREG-18    1;

    WDPIPE2-33  1;
    WDPIPE2-19  1;
    WDPIPE2-18  1;

    WDPIPE3-33  1;
    WDPIPE3-19  1;
    WDPIPE3-18  1;

    WDPIPE1-33  1;
    WDPIPE1-19  1;
    WDPIPE1-18  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       CHIP: ROTCAT          LOCATION: I0K6
% DESCRIPTION: WORD 2  BIT 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW2-36  1;
    MDATIW2-32  1;
    MDATIW2-28  1;
    MDATIW2-24  1;
    MDATIW2-20  1;
    MDATIW2-16  1;
    MDATIW2-12  1;
    MDATIW2-08  1;
    MDATIW2-04  1;
    MDATIW2-00  1;
```

14.2 MD (Continued)

```
MDATOW2-36  1;
MDATOW2-32  1;
MDATOW2-28  1;
MDATOW2-24  1;
MDATOW2-20  1;
MDATOW2-16  1;
MDATOW2-12  1;
MDATOW2-08  1;
MDATOW2-04  1;
MDATOW2-00  1;

NOTUSED-48  1;

RBUSREG-32  1;
RBUSREG-17  1;
RBUSREG-16  1;

WECCB1W2-1  1;
WECCB1W2-0  1;
WECCB0W2-1  1;
WECCB0W2-0  1;

PCREG-32    1;
PCREG-17    1;
PCREG-16    1;

WDPIPE2-32  1;
WDPIPE2-17  1;
WDPIPE2-16  1;

WDPIPE3-32  1;
WDPIPE3-17  1;
WDPIPE3-16  1;

WDPIPE1-32  1;
WDPIPE1-17  1;
WDPIPE1-16  1;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT            LOCATION: K4I2
% DESCRIPTION: WORD 1  BIT 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW1-39  1;
    MDATIW1-35  1;
    MDATIW1-31  1;
    MDATIW1-27  1;
    MDATIW1-23  1;
    MDATIW1-19  1;
    MDATIW1-15  1;
    MDATIW1-11  1;
    MDATIW1-07  1;
    MDATIW1-03  1;

    MDATOW1-39  1;
    MDATOW1-35  1;
    MDATOW1-31  1;
    MDATOW1-27  1;
    MDATOW1-23  1;
    MDATOW1-19  1;
    MDATOW1-15  1;
    MDATOW1-11  1;
    MDATOW1-07  1;
    MDATOW1-03  1;

    NOTUSED-47  1;

    NOTUSED-46  1;
    RBUSREG-15  1;
    RBUSREG-14  1;

    WECCB1W1-7  1;
    WECCB1W1-6  1;
    WECCB0W1-7  1;
    WECCB0W1-6  1;

    NOTUSED-45  1;
    PCREG-15    1;
    PCREG-14    1;
    NOTUSED-44  1;
    WDPIPE2-15  1;
    WDPIPE2-14  1;
```

14.2 MD (Continued)

```
    NOTUSED-43  1;
    WDPIPE3-15  1;
    WDPIPE3-14  1;

    NOTUSED-42  1;
    WDPIPE1-15  1;
    WDPIPE1-14  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT          LOCATION: K4K6
% DESCRIPTION: WORD 1  BIT 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW1-38  1;
    MDATIW1-34  1;
    MDATIW1-30  1;
    MDATIW1-26  1;
    MDATIW1-22  1;
    MDATIW1-18  1;
    MDATIW1-14  1;
    MDATIW1-10  1;
    MDATIW1-06  1;
    MDATIW1-02  1;

    MDATOW1-38  1;
    MDATOW1-34  1;
    MDATOW1-30  1;
    MDATOW1-26  1;
    MDATOW1-22  1;
    MDATOW1-18  1;
    MDATOW1-14  1;
    MDATOW1-10  1;
    MDATOW1-06  1;
    MDATOW1-02  1;

    NOTUSED-41  1;

    NOTUSED-40  1;
    RBUSREG-13  1;
    RBUSREG-12  1;

    WECCB1W1-5  1;
    WECCB1W1-4  1;
    WECCB0W1-5  1;
    WECCB0W1-4  1;
```

14.2 MD (Continued)

```
    NOTUSED-39  1;
    PCREG-13    1;
    PCREG-12    1;

    NOTUSED-38  1;
    WDPIPE2-13  1;
    WDPIPE2-12  1;

    NOTUSED-37  1;
    WDPIPE3-13  1;
    WDPIPE3-12  1;

    NOTUSED-36  1;
    WDPIPE1-13  1;
    WDPIPE1-12  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT            LOCATION: M8I2
% DESCRIPTION: WORD 1  BIT 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW1-37  1;
    MDATIW1-33  1;
    MDATIW1-29  1;
    MDATIW1-25  1;
    MDATIW1-21  1;
    MDATIW1-17  1;
    MDATIW1-13  1;
    MDATIW1-09  1;
    MDATIW1-05  1;
    MDATIW1-01  1;

    MDATOW1-37  1;
    MDATOW1-33  1;
    MDATOW1-29  1;
    MDATOW1-25  1;
    MDATOW1-21  1;
    MDATOW1-17  1;
    MDATOW1-13  1;
    MDATOW1-09  1;
    MDATOW1-05  1;
    MDATOW1-01  1;

    NOTUSED-35  1;
```

14.2 MD (Continued)

```
    NOTUSED-34  1;
    RBUSREG-11  1;
    RBUSREG-10  1;

    WECCB1W1-3  1;
    WECCB1W1-2  1;
    WECCB0W1-3  1;
    WECCB0W1-2  1;

    NOTUSED-33  1;
    PCREG-11    1;
    PCREG-10     1;

    NOTUSED-32  1;
    WDPIPE2-11  1;
    WDPIPE2-10  1;

    NOTUSED-31  1;
    WDPIPE3-11  1;
    WDPIPE3-10  1;

    NOTUSED-30  1;
    WDPIPE1-11  1;
    WDPIPE1-10  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT            LOCATION: M8K6
% DESCRIPTION: WORD 1  BIT 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW1-36  1;
    MDATIW1-32  1;
    MDATIW1-28  1;
    MDATIW1-24  1;
    MDATIW1-20  1;
    MDATIW1-16  1;
    MDATIW1-12  1;
    MDATIW1-08  1;
    MDATIW1-04  1;
    MDATIW1-00  1;
```

14.2 MD (Continued)

```
MDATOW1-36  1;
MDATOW1-32  1;
MDATOW1-28  1;
MDATOW1-24  1;
MDATOW1-20  1;
MDATOW1-16  1;
MDATOW1-12  1;
MDATOW1-08  1;
MDATOW1-04  1;
MDATOW1-00  1;

NOTUSED-29  1;

NOTUSED-28  1;
RBUSREG-09  1;
RBUSREG-08  1;

WECCB1W1-1  1;
WECCB1W1-0  1;
WECCB0W1-1  1;
WECCB0W1-0  1;

NOTUSED-27  1;
PCREG-09    1;
PCREG-08    1;

NOTUSED-26  1;
WDPIPE2-09  1;
WDPIPE2-08  1;

NOTUSED-25  1;
WDPIPE3-09  1;
WDPIPE3-08  1;

NOTUSED-24  1;
WDPIPE1-09  1;
WDPIPE1-08  1;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT            LOCATION: P2I2
% DESCRIPTION: WORD 0  BIT 8
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW0-39  1;
    MDATIW0-35  1;
    MDATIW0-31  1;
    MDATIW0-27  1;
    MDATIW0-23  1;
    MDATIW0-19  1;
    MDATIW0-15  1;
    MDATIW0-11  1;
    MDATIW0-07  1;
    MDATIW0-03  1;

    MDATOW0-39  1;
    MDATOW0-35  1;
    MDATOW0-31  1;
    MDATOW0-27  1;
    MDATOW0-23  1;
    MDATOW0-19  1;
    MDATOW0-15  1;
    MDATOW0-11  1;
    MDATOW0-07  1;
    MDATOW0-03  1;

    NOTUSED-23  1;

    NOTUSED-22  1;
    RBUSREG-07  1;
    RBUSREG-06  1;

    WECCB1W0-7  1;
    WECCB1W0-6  1;
    WECCB0W0-7  1;
    WECCB0W0-6  1;

    NOTUSED-21  1;
    PCREG-07    1;
    PCREG-06    1;
```

14.2 MD (Continued)

```
    NOTUSED-20  1;
    WDPIPE2-07  1;
    WDPIPE2-06  1;

    NOTUSED-19  1;
    WDPIPE3-07  1;
    WDPIPE3-06  1;

    NOTUSED-18  1;
    WDPIPE1-07  1;
    WDPIPE1-06  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT           LOCATION: P2K6
% DESCRIPTION: WORD 0  BIT 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW0-38  1;
    MDATIW0-34  1;
    MDATIW0-30  1;
    MDATIW0-26  1;
    MDATIW0-22  1;
    MDATIW0-18  1;
    MDATIW0-14  1;
    MDATIW0-10  1;
    MDATIW0-06  1;
    MDATIW0-02  1;

    MDATOW0-38  1;
    MDATOW0-34  1;
    MDATOW0-30  1;
    MDATOW0-26  1;
    MDATOW0-22  1;
    MDATOW0-18  1;
    MDATOW0-14  1;
    MDATOW0-10  1;
    MDATOW0-06  1;
    MDATOW0-02  1;

    NOTUSED-17  1;
```

14.2 MD (Continued)

```
    NOTUSED-16  1;
    RBUSREG-05  1;
    RBUSREG-04  1;

    WECCB1W0-5  1;
    WECCB1W0-4  1;
    WECCB0W0-5  1;
    WECCB0W0-4  1;

    NOTUSED-15  1;
    PCREG-05    1;
    PCREG-04    1;

    NOTUSED-14  1;
    WDPIPE2-05  1;
    WDPIPE2-04  1;

    NOTUSED-13  1;
    WDPIPE3-05  1;
    WDPIPE3-04  1;

    NOTUSED-12  1;
    WDPIPE1-05  1;
    WDPIPE1-04  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT           LOCATION: R6I2
% DESCRIPTION: WORD 0  BIT 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW0-37  1;
    MDATIW0-33  1;
    MDATIW0-29  1;
    MDATIW0-25  1;
    MDATIW0-21  1;
    MDATIW0-17  1;
    MDATIW0-13  1;
    MDATIW0-09  1;
    MDATIW0-05  1;
    MDATIW0-01  1;
```

14.2 MD (Continued)

```
MDATOW0-37  1;
MDATOW0-33  1;
MDATOW0-29  1;
MDATOW0-25  1;
MDATOW0-21  1;
MDATOW0-17  1;
MDATOW0-13  1;
MDATOW0-09  1;
MDATOW0-05  1;
MDATOW0-01  1;

NOTUSED-11  1;

NOTUSED-10  1;
RBUSREG-03  1;
RBUSREG-02  1;

WECCB1W0-3  1;
WECCB1W0-2  1;
WECCB0W0-3  1;
WECCB0W0-2  1;

NOTUSED-09  1;
PCREG-03    1;
PCREG-02    1;

NOTUSED-08  1;
WDPIPE2-03  1;
WDPIPE2-02  1;

NOTUSED-07  1;
WDPIPE3-03  1;
WDPIPE3-02  1;

NOTUSED-06  1;
WDPIPE1-03  1;
WDPIPE1-02  1;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: ROTCAT           LOCATION: R6K6
% DESCRIPTION: WORD 0  BIT 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MDATIW0-36  1;
    MDATIW0-32  1;
    MDATIW0-28  1;
    MDATIW0-24  1;
    MDATIW0-20  1;
    MDATIW0-16  1;
    MDATIW0-12  1;
    MDATIW0-08  1;
    MDATIW0-04  1;
    MDATIW0-00  1;

    MDATOW0-36  1;
    MDATOW0-32  1;
    MDATOW0-28  1;
    MDATOW0-24  1;
    MDATOW0-20  1;
    MDATOW0-16  1;
    MDATOW0-12  1;
    MDATOW0-08  1;
    MDATOW0-04  1;
    MDATOW0-00  1;

    NOTUSED-05  1;

    NOTUSED-04  1;
    RBUSREG-01  1;
    RBUSREG-00  1;

    WECCB1W0-1  1;
    WECCB1W0-0  1;
    WECCB0W0-1  1;
    WECCB0W0-0  1;

    NOTUSED-03  1;
    PCREG-01    1;
    PCREG-00    1;
```

14.2 MD (Continued)

```
    NOTUSED-02  1;
    WDPIPE2-01  1;
    WDPIPE2-00  1;

    NOTUSED-01  1;
    WDPIPE3-01  1;
    WDPIPE3-00  1;

    NOTUSED-00  1;
    WDPIPE1-01  1;
    WDPIPE1-00  1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2            LOCATION: A8N0
% DESCRIPTION: BLOCK 1 WORD 3   SDI:P6  SDO:M15  SHIFT:B10  CLEAR:R11
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB1W3   8;
    MECCIB1W3   8;
    CECCB1W3    8;
    GECCLB1W3   8;
    CDATB1W3D9  4;
    CDATB1W3D8  4;
    CDATB1W3D7  4;
    CDATB1W3D6  4;
    CDATB1W3D5  4;
    CDATB1W3D4  4;
    CDATB1W3D3  4;
    CDATB1W3D2  4;
    CDATB1W3D1  4;
    CDATB1W3D0  4;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2           LOCATION: D2N0
% DESCRIPTION: BLOCK 0 WORD 3                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB0W3    8;
    MECCIB0W3    8;
    CECCB0W3     8;
    GECCLB0W3    8;
    CDATB0W3D9   4;
    CDATB0W3D8   4;
    CDATB0W3D7   4;
    CDATB0W3D6   4;
    CDATB0W3D5   4;
    CDATB0W3D4   4;
    CDATB0W3D3   4;
    CDATB0W3D2   4;
    CDATB0W3D1   4;
    CDATB0W3D0   4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2           LOCATION: F6N0
% DESCRIPTION: BLOCK 1 WORD 2                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB1W2    8;
    MECCIB1W2    8;
    CECCB1W2     8;
    GECCLB1W2    8;
    CDATB1W2D9   4;
    CDATB1W2D8   4;
    CDATB1W2D7   4;
    CDATB1W2D6   4;
    CDATB1W2D5   4;
    CDATB1W2D4   4;
    CDATB1W2D3   4;
    CDATB1W2D2   4;
    CDATB1W2D1   4;
    CDATB1W2D0   4;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2            LOCATION: I0N0
% DESCRIPTION: BLOCK 0 WORD 2                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB0W2   8;
    MECCIB0W2   8;
    CECCB0W2    8;
    GECCLB0W2   8;
    CDATB0W2D9  4;
    CDATB0W2D8  4;
    CDATB0W2D7  4;
    CDATB0W2D6  4;
    CDATB0W2D5  4;
    CDATB0W2D4  4;
    CDATB0W2D3  4;
    CDATB0W2D2  4;
    CDATB0W2D1  4;
    CDATB0W2D0  4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2            LOCATION: K4N0
% DESCRIPTION: BLOCK 1 WORD 1                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB1W1   8;
    MECCIB1W1   8;
    CECCB1W1    8;
    GECCLB1W1   8;
    CDATB1W1D9  4;
    CDATB1W1D8  4;
    CDATB1W1D7  4;
    CDATB1W1D6  4;
    CDATB1W1D5  4;
    CDATB1W1D4  4;
    CDATB1W1D3  4;
    CDATB1W1D2  4;
    CDATB1W1D1  4;
    CDATB1W1D0  4;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2           LOCATION: M8N0
% DESCRIPTION: BLOCK 0 WORD 1                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB0W1    8;
    MECCIB0W1    8;
    CECCB0W1     8;
    GECCLB0W1    8;
    CDATB0W1D9   4;
    CDATB0W1D8   4;
    CDATB0W1D7   4;
    CDATB0W1D6   4;
    CDATB0W1D5   4;
    CDATB0W1D4   4;
    CDATB0W1D3   4;
    CDATB0W1D2   4;
    CDATB0W1D1   4;
    CDATB0W1D0   4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2           LOCATION: P2N0
% DESCRIPTION: BLOCK 1 WORD 0                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB1W0    8;
    MECCIB1W0    8;
    CECCB1W0     8;
    GECCLB1W0    8;
    CDATB1W0D9   4;
    CDATB1W0D8   4;
    CDATB1W0D7   4;
    CDATB1W0D6   4;
    CDATB1W0D5   4;
    CDATB1W0D4   4;
    CDATB1W0D3   4;
    CDATB1W0D2   4;
    CDATB1W0D1   4;
    CDATB1W0D0   4;
```

14.2 MD (Continued)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        CHIP: MCECC2            LOCATION: R6N0
% DESCRIPTION: BLOCK 0 WORD 0                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    MECCOB0W0    8;
    MECCIB0W0    8;
    CECCB0W0     8;
    GECCLB0W0    8;
    CDATB0W0D9   4;
    CDATB0W0D8   4;
    CDATB0W0D7   4;
    CDATB0W0D6   4;
    CDATB0W0D5   4;
    CDATB0W0D4   4;
    CDATB0W0D3   4;
    CDATB0W0D2   4;
    CDATB0W0D1   4;
    CDATB0W0D0   4;

CHNEND;                          % END OF CHAIN.
```

14.2 MD (Continued)

```
REGLIST   MD;

  WDPIPE1     40;  % WRITE DATE PIPELINE STAGE 1.
    WDPIPE1-39;
    WDPIPE1-38;
    WDPIPE1-37;
    WDPIPE1-36;
    WDPIPE1-35;
    WDPIPE1-34;
    WDPIPE1-33;
    WDPIPE1-32;
    WDPIPE1-31;
    WDPIPE1-30;
    WDPIPE1-29;
    WDPIPE1-28;
    WDPIPE1-27;
    WDPIPE1-26;
    WDPIPE1-25;
    WDPIPE1-24;
    WDPIPE1-23;
    WDPIPE1-22;
    WDPIPE1-21;
    WDPIPE1-20;
    WDPIPE1-19;
    WDPIPE1-18;
    WDPIPE1-17;
    WDPIPE1-16;
    WDPIPE1-15;
    WDPIPE1-14;
    WDPIPE1-13;
    WDPIPE1-12;
    WDPIPE1-11;
    WDPIPE1-10;
    WDPIPE1-09;
    WDPIPE1-08;
    WDPIPE1-07;
    WDPIPE1-06;
    WDPIPE1-05;
    WDPIPE1-04;
    WDPIPE1-03;
    WDPIPE1-02;
    WDPIPE1-01;
    WDPIPE1-00#
```

14.2 MD (Continued)

```
WDPIPE2     40;  % WRITE DATE PIPELINE STAGE 2.
  WDPIPE2-39;
  WDPIPE2-38;
  WDPIPE2-37;
  WDPIPE2-36;
  WDPIPE2-35;
  WDPIPE2-34;
  WDPIPE2-33;
  WDPIPE2-32;
  WDPIPE2-31;
  WDPIPE2-30;
  WDPIPE2-29;
  WDPIPE2-28;
  WDPIPE2-27;
  WDPIPE2-26;
  WDPIPE2-25;
  WDPIPE2-24;
  WDPIPE2-23;
  WDPIPE2-22;
  WDPIPE2-21;
  WDPIPE2-20;
  WDPIPE2-19;
  WDPIPE2-18;
  WDPIPE2-17;
  WDPIPE2-16;
  WDPIPE2-15;
  WDPIPE2-14;
  WDPIPE2-13;
  WDPIPE2-12;
  WDPIPE2-11;
  WDPIPE2-10;
  WDPIPE2-09;
  WDPIPE2-08;
  WDPIPE2-07;
  WDPIPE2-06;
  WDPIPE2-05;
  WDPIPE2-04;
  WDPIPE2-03;
  WDPIPE2-02;
  WDPIPE2-01;
  WDPIPE2-00#
```

14.2 MD (Continued)

```
WDPIPE3     40;  % WRITE DATE PIPELINE STAGE 3.
  WDPIPE3-39;
  WDPIPE3-38;
  WDPIPE3-37;
  WDPIPE3-36;
  WDPIPE3-35;
  WDPIPE3-34;
  WDPIPE3-33;
  WDPIPE3-32;
  WDPIPE3-31;
  WDPIPE3-30;
  WDPIPE3-29;
  WDPIPE3-28;
  WDPIPE3-27;
  WDPIPE3-26;
  WDPIPE3-25;
  WDPIPE3-24;
  WDPIPE3-23;
  WDPIPE3-22;
  WDPIPE3-21;
  WDPIPE3-20;
  WDPIPE3-19;
  WDPIPE3-18;
  WDPIPE3-17;
  WDPIPE3-16;
  WDPIPE3-15;
  WDPIPE3-14;
  WDPIPE3-13;
  WDPIPE3-12;
  WDPIPE3-11;
  WDPIPE3-10;
  WDPIPE3-09;
  WDPIPE3-08;
  WDPIPE3-07;
  WDPIPE3-06;
  WDPIPE3-05;
  WDPIPE3-04;
  WDPIPE3-03;
  WDPIPE3-02;
  WDPIPE3-01;
  WDPIPE3-00#
```

14.2 MD (Continued)

```
RBUSREG     40;  % RBUS REGISTER.
  RBUSREG-39;
  RBUSREG-38;
  RBUSREG-37;
  RBUSREG-36;
  RBUSREG-35;
  RBUSREG-34;
  RBUSREG-33;
  RBUSREG-32;
  RBUSREG-31;
  RBUSREG-30;
  RBUSREG-29;
  RBUSREG-28;
  RBUSREG-27;
  RBUSREG-26;
  RBUSREG-25;
  RBUSREG-24;
  RBUSREG-23;
  RBUSREG-22;
  RBUSREG-21;
  RBUSREG-20;
  RBUSREG-19;
  RBUSREG-18;
  RBUSREG-17;
  RBUSREG-16;
  RBUSREG-15;
  RBUSREG-14;
  RBUSREG-13;
  RBUSREG-12;
  RBUSREG-11;
  RBUSREG-10;
  RBUSREG-09;
  RBUSREG-08;
  RBUSREG-07;
  RBUSREG-06;
  RBUSREG-05;
  RBUSREG-04;
  RBUSREG-03;
  RBUSREG-02;
  RBUSREG-01;
  RBUSREG-00#
```

14.2 MD (Continued)

```
PCREG     40;  % PC REGISTER.
  PCREG-39;
  PCREG-38;
  PCREG-37;
  PCREG-36;
  PCREG-35;
  PCREG-34;
  PCREG-33;
  PCREG-32;
  PCREG-31;
  PCREG-30;
  PCREG-29;
  PCREG-28;
  PCREG-27;
  PCREG-26;
  PCREG-25;
  PCREG-24;
  PCREG-23;
  PCREG-22;
  PCREG-21;
  PCREG-20;
  PCREG-19;
  PCREG-18;
  PCREG-17;
  PCREG-16;
  PCREG-15;
  PCREG-14;
  PCREG-13;
  PCREG-12;
  PCREG-11;
  PCREG-10;
  PCREG-09;
  PCREG-08;
  PCREG-07;
  PCREG-06;
  PCREG-05;
  PCREG-04;
  PCREG-03;
  PCREG-02;
  PCREG-01;
  PCREG-00#
```

14.2 MD (Continued)

```
MDATOW3     40;  % MBUS DATA INPUT REGISTER WORD 3.
  MDATOW3-39;
  MDATOW3-38;
  MDATOW3-37;
  MDATOW3-36;
  MDATOW3-35;
  MDATOW3-34;
  MDATOW3-33;
  MDATOW3-32;
  MDATOW3-31;
  MDATOW3-30;
  MDATOW3-29;
  MDATOW3-28;
  MDATOW3-27;
  MDATOW3-26;
  MDATOW3-25;
  MDATOW3-24;
  MDATOW3-23;
  MDATOW3-22;
  MDATOW3-21;
  MDATOW3-20;
  MDATOW3-19;
  MDATOW3-18;
  MDATOW3-17;
  MDATOW3-16;
  MDATOW3-15;
  MDATOW3-14;
  MDATOW3-13;
  MDATOW3-12;
  MDATOW3-11;
  MDATOW3-10;
  MDATOW3-09;
  MDATOW3-08;
  MDATOW3-07;
  MDATOW3-06;
  MDATOW3-05;
  MDATOW3-04;
  MDATOW3-03;
  MDATOW3-02;
  MDATOW3-01;
  MDATOW3-00#
```

14.2 MD (Continued)

```
MDATOW2     40;  % MBUS DATA INPUT REGISTER WORD 2.
  MDATOW2-39;
  MDATOW2-38;
  MDATOW2-37;
  MDATOW2-36;
  MDATOW2-35;
  MDATOW2-34;
  MDATOW2-33;
  MDATOW2-32;
  MDATOW2-31;
  MDATOW2-30;
  MDATOW2-29;
  MDATOW2-28;
  MDATOW2-27;
  MDATOW2-26;
  MDATOW2-25;
  MDATOW2-24;
  MDATOW2-23;
  MDATOW2-22;
  MDATOW2-21;
  MDATOW2-20;
  MDATOW2-19;
  MDATOW2-18;
  MDATOW2-17;
  MDATOW2-16;
  MDATOW2-15;
  MDATOW2-14;
  MDATOW2-13;
  MDATOW2-12;
  MDATOW2-11;
  MDATOW2-10;
  MDATOW2-09;
  MDATOW2-08;
  MDATOW2-07;
  MDATOW2-06;
  MDATOW2-05;
  MDATOW2-04;
  MDATOW2-03;
  MDATOW2-02;
  MDATOW2-01;
  MDATOW2-00#
```

14.2 MD (Continued)

```
MDATOW1     40;  % MBUS DATA INPUT REGISTER WORD 1.
  MDATOW1-39;
  MDATOW1-38;
  MDATOW1-37;
  MDATOW1-36;
  MDATOW1-35;
  MDATOW1-34;
  MDATOW1-33;
  MDATOW1-32;
  MDATOW1-31;
  MDATOW1-30;
  MDATOW1-29;
  MDATOW1-28;
  MDATOW1-27;
  MDATOW1-26;
  MDATOW1-25;
  MDATOW1-24;
  MDATOW1-23;
  MDATOW1-22;
  MDATOW1-21;
  MDATOW1-20;
  MDATOW1-19;
  MDATOW1-18;
  MDATOW1-17;
  MDATOW1-16;
  MDATOW1-15;
  MDATOW1-14;
  MDATOW1-13;
  MDATOW1-12;
  MDATOW1-11;
  MDATOW1-10;
  MDATOW1-09;
  MDATOW1-08;
  MDATOW1-07;
  MDATOW1-06;
  MDATOW1-05;
  MDATOW1-04;
  MDATOW1-03;
  MDATOW1-02;
  MDATOW1-01;
  MDATOW1-00#
```

14.2 MD (Continued)

```
MDATOW0     40;  % MBUS DATA INPUT REGISTER WORD 0.
  MDATOW0-39;
  MDATOW0-38;
  MDATOW0-37;
  MDATOW0-36;
  MDATOW0-35;
  MDATOW0-34;
  MDATOW0-33;
  MDATOW0-32;
  MDATOW0-31;
  MDATOW0-30;
  MDATOW0-29;
  MDATOW0-28;
  MDATOW0-27;
  MDATOW0-26;
  MDATOW0-25;
  MDATOW0-24;
  MDATOW0-23;
  MDATOW0-22;
  MDATOW0-21;
  MDATOW0-20;
  MDATOW0-19;
  MDATOW0-18;
  MDATOW0-17;
  MDATOW0-16;
  MDATOW0-15;
  MDATOW0-14;
  MDATOW0-13;
  MDATOW0-12;
  MDATOW0-11;
  MDATOW0-10;
  MDATOW0-09;
  MDATOW0-08;
  MDATOW0-07;
  MDATOW0-06;
  MDATOW0-05;
  MDATOW0-04;
  MDATOW0-03;
  MDATOW0-02;
  MDATOW0-01;
  MDATOW0-00#
```

14.2 MD (Continued)

```
MDATIW3     40;  % MBUS DATA INPUT REGISTER WORD 3.
  MDATIW3-39;
  MDATIW3-38;
  MDATIW3-37;
  MDATIW3-36;
  MDATIW3-35;
  MDATIW3-34;
  MDATIW3-33;
  MDATIW3-32;
  MDATIW3-31;
  MDATIW3-30;
  MDATIW3-29;
  MDATIW3-28;
  MDATIW3-27;
  MDATIW3-26;
  MDATIW3-25;
  MDATIW3-24;
  MDATIW3-23;
  MDATIW3-22;
  MDATIW3-21;
  MDATIW3-20;
  MDATIW3-19;
  MDATIW3-18;
  MDATIW3-17;
  MDATIW3-16;
  MDATIW3-15;
  MDATIW3-14;
  MDATIW3-13;
  MDATIW3-12;
  MDATIW3-11;
  MDATIW3-10;
  MDATIW3-09;
  MDATIW3-08;
  MDATIW3-07;
  MDATIW3-06;
  MDATIW3-05;
  MDATIW3-04;
  MDATIW3-03;
  MDATIW3-02;
  MDATIW3-01;
  MDATIW3-00#
```

14.2 MD (Continued)

```
MDATIW2     40;  % MBUS DATA INPUT REGISTER WORD 2.
  MDATIW2-39;
  MDATIW2-38;
  MDATIW2-37;
  MDATIW2-36;
  MDATIW2-35;
  MDATIW2-34;
  MDATIW2-33;
  MDATIW2-32;
  MDATIW2-31;
  MDATIW2-30;
  MDATIW2-29;
  MDATIW2-28;
  MDATIW2-27;
  MDATIW2-26;
  MDATIW2-25;
  MDATIW2-24;
  MDATIW2-23;
  MDATIW2-22;
  MDATIW2-21;
  MDATIW2-20;
  MDATIW2-19;
  MDATIW2-18;
  MDATIW2-17;
  MDATIW2-16;
  MDATIW2-15;
  MDATIW2-14;
  MDATIW2-13;
  MDATIW2-12;
  MDATIW2-11;
  MDATIW2-10;
  MDATIW2-09;
  MDATIW2-08;
  MDATIW2-07;
  MDATIW2-06;
  MDATIW2-05;
  MDATIW2-04;
  MDATIW2-03;
  MDATIW2-02;
  MDATIW2-01;
  MDATIW2-00#
```

14.2 MD (Continued)

```
MDATIW1     40;  % MBUS DATA INPUT REGISTER WORD 1.
  MDATIW1-39;
  MDATIW1-38;
  MDATIW1-37;
  MDATIW1-36;
  MDATIW1-35;
  MDATIW1-34;
  MDATIW1-33;
  MDATIW1-32;
  MDATIW1-31;
  MDATIW1-30;
  MDATIW1-29;
  MDATIW1-28;
  MDATIW1-27;
  MDATIW1-26;
  MDATIW1-25;
  MDATIW1-24;
  MDATIW1-23;
  MDATIW1-22;
  MDATIW1-21;
  MDATIW1-20;
  MDATIW1-19;
  MDATIW1-18;
  MDATIW1-17;
  MDATIW1-16;
  MDATIW1-15;
  MDATIW1-14;
  MDATIW1-13;
  MDATIW1-12;
  MDATIW1-11;
  MDATIW1-10;
  MDATIW1-09;
  MDATIW1-08;
  MDATIW1-07;
  MDATIW1-06;
  MDATIW1-05;
  MDATIW1-04;
  MDATIW1-03;
  MDATIW1-02;
  MDATIW1-01;
  MDATIW1-00#
```

14.2 MD (Continued)

```
MDATIW0     40;  % MBUS DATA INPUT REGISTER WORD 0.
  MDATIW0-39;
  MDATIW0-38;
  MDATIW0-37;
  MDATIW0-36;
  MDATIW0-35;
  MDATIW0-34;
  MDATIW0-33;
  MDATIW0-32;
  MDATIW0-31;
  MDATIW0-30;
  MDATIW0-29;
  MDATIW0-28;
  MDATIW0-27;
  MDATIW0-26;
  MDATIW0-25;
  MDATIW0-24;
  MDATIW0-23;
  MDATIW0-22;
  MDATIW0-21;
  MDATIW0-20;
  MDATIW0-19;
  MDATIW0-18;
  MDATIW0-17;
  MDATIW0-16;
  MDATIW0-15;
  MDATIW0-14;
  MDATIW0-13;
  MDATIW0-12;
  MDATIW0-11;
  MDATIW0-10;
  MDATIW0-09;
  MDATIW0-08;
  MDATIW0-07;
  MDATIW0-06;
  MDATIW0-05;
  MDATIW0-04;
  MDATIW0-03;
  MDATIW0-02;
  MDATIW0-01;
  MDATIW0-00#
```

14.2 MD (Continued)

```
WECCB1W3     8;  % WRITE ECC BLOCK 1 WORD 3.
  WECCB1W3-7;
  WECCB1W3-6;
  WECCB1W3-5;
  WECCB1W3-4;
  WECCB1W3-3;
  WECCB1W2-2;
  WECCB1W3-1;
  WECCB1W3-0#

WECCB0W3     8;  % WRITE ECC BLOCK 0 WORD 3.
  WECCB0W3-7;
  WECCB0W3-6;
  WECCB0W3-5;
  WECCB0W3-4;
  WECCB0W3-3;
  WECCB0W3-2;
  WECCB0W3-1;
  WECCB0W3-0#

WECCB1W2     8;  % WRITE ECC BLOCK 1 WORD 2.
  WECCB1W2-7;
  WECCB1W2-6;
  WECCB1W2-5;
  WECCB1W2-4;
  WECCB1W2-3;
  WECCB1W2-2;
  WECCB1W2-1;
  WECCB1W2-0#

WECCB0W2     8;  % WRITE ECC BLOCK 0 WORD 2.
  WECCB0W2-7;
  WECCB0W2-6;
  WECCB0W2-5;
  WECCB0W2-4;
  WECCB0W2-3;
  WECCB0W2-2;
  WECCB0W2-1;
  WECCB0W2-0#
```

14.2 MD (Continued)

```
WECCB1W1     8;  % WRITE ECC BLOCK 1 WORD 1.
  WECCB1W1-7;
  WECCB1W1-6;
  WECCB1W1-5;
  WECCB1W1-4;
  WECCB1W1-3;
  WECCB1W1-2;
  WECCB1W1-1;
  WECCB1W1-0#

WECCB0W1     8;  % WRITE ECC BLOCK 0 WORD 1.
  WECCB0W1-7;
  WECCB0W1-6;
  WECCB0W1-5;
  WECCB0W1-4;
  WECCB0W1-3;
  WECCB0W1-2;
  WECCB0W1-1;
  WECCB0W1-0#

WECCB1W0     8;  % WRITE ECC BLOCK 1 WORD 0.
  WECCB1W0-7;
  WECCB1W0-6;
  WECCB1W0-5;
  WECCB1W0-4;
  WECCB1W0-3;
  WECCB1W0-2;
  WECCB1W0-1;
  WECCB1W0-0#

WECCB0W0     8;  % WRITE ECC BLOCK 0 WORD 0.
  WECCB0W0-7;
  WECCB0W0-6;
  WECCB0W0-5;
  WECCB0W0-4;
  WECCB0W0-3;
  WECCB0W0-2;
  WECCB0W0-1;
  WECCB0W0-0#
```

14.2 MD (Continued)

```
MECCOB0 4;
  MECCOB0W3;
  MECCOB0W2;
  MECCOB0W1;
  MECCOB0W0#

MECCIB0 4;
  MECCIB0W3;
  MECCIB0W2;
  MECCIB0W1;
  MECCIB0W0#

CECCB0 4;
  CECCB0W3;
  CECCB0W2;
  CECCB0W1;
  CECCB0W0#

GECCLB0 4;
  GECCLB0W3;
  GECCLB0W2;
  GECCLB0W1;
  GECCLB0W0#

MECCOB1 4;
  MECCOB1W3;
  MECCOB1W2;
  MECCOB1W1;
  MECCOB1W0#

MECCIB1 4;
  MECCIB1W3;
  MECCIB1W2;
  MECCIB1W1;
  MECCIB1W0#

CECCB1 4;
  CECCB1W3;
  CECCB1W2;
  CECCB1W1;
  CECCB1W0#
```

14.2 MD (Continued)

```
GECCLB1 4;
  GECCLB1W3;
  GECCLB1W2;
  GECCLB1W1;
  GECCLB1W0#

WECCB1     32;  % WRITE ECC BLOCK 1.
  WECCB1W3-7;
  WECCB1W3-6;
  WECCB1W3-5;
  WECCB1W3-4;
  WECCB1W3-3;
  WECCB1W3-2;
  WECCB1W3-1;
  WECCB1W3-0;
  WECCB1W2-7;
  WECCB1W2-6;
  WECCB1W2-5;
  WECCB1W2-4;
  WECCB1W2-3;
  WECCB1W2-2;
  WECCB1W2-1;
  WECCB1W2-0;
  WECCB1W1-7;
  WECCB1W1-6;
  WECCB1W1-5;
  WECCB1W1-4;
  WECCB1W1-3;
  WECCB1W1-2;
  WECCB1W1-1;
  WECCB1W1-0;
  WECCB1W0-7;
  WECCB1W0-6;
  WECCB1W0-5;
  WECCB1W0-4;
  WECCB1W0-3;
  WECCB1W0-2;
  WECCB1W0-1;
  WECCB1W0-0#
```

14.2 MD (Continued)

```
WECCB0          32;  % WRITE ECC BLOCK 0.
  WECCB0W3-7;
  WECCB0W3-6;
  WECCB0W3-5;
  WECCB0W3-4;
  WECCB0W3-3;
  WECCB0W3-2;
  WECCB0W3-1;
  WECCB0W3-0;
  WECCB0W2-7;
  WECCB0W2-6;
  WECCB0W2-5;
  WECCB0W2-4;
  WECCB0W2-3;
  WECCB0W2-2;
  WECCB0W2-1;
  WECCB0W2-0;
  WECCB0W1-7;
  WECCB0W1-6;
  WECCB0W1-5;
  WECCB0W1-4;
  WECCB0W1-3;
  WECCB0W1-2;
  WECCB0W1-1;
  WECCB0W1-0;
  WECCB0W0-7;
  WECCB0W0-6;
  WECCB0W0-5;
  WECCB0W0-4;
  WECCB0W0-3;
  WECCB0W0-2;
  WECCB0W0-1;
  WECCB0W0-0#

CDB0W0 10;
  CDATB0W0D0;
  CDATB0W0D1;
  CDATB0W0D2;
  CDATB0W0D3;
  CDATB0W0D4;
  CDATB0W0D5;
  CDATB0W0D6;
  CDATB0W0D7;
  CDATB0W0D8;
  CDATB0W0D9#
```

14.2 MD (Continued)

```
CDB0W1 10;
  CDATB0W1D0;
  CDATB0W1D1;
  CDATB0W1D2;
  CDATB0W1D3;
  CDATB0W1D4;
  CDATB0W1D5;
  CDATB0W1D6;
  CDATB0W1D7;
  CDATB0W1D8;
  CDATB0W1D9#

CDB0W2 10;
  CDATB0W2D0;
  CDATB0W2D1;
  CDATB0W2D2;
  CDATB0W2D3;
  CDATB0W2D4;
  CDATB0W2D5;
  CDATB0W2D6;
  CDATB0W2D7;
  CDATB0W2D8;
  CDATB0W2D9#

CDB0W3 10;
  CDATB0W3D0;
  CDATB0W3D1;
  CDATB0W3D2;
  CDATB0W3D3;
  CDATB0W3D4;
  CDATB0W3D5;
  CDATB0W3D6;
  CDATB0W3D7;
  CDATB0W3D8;
  CDATB0W3D9#
```

14.2 MD (Continued)

```
CDB1W0 10;
  CDATB1W0D0;
  CDATB1W0D1;
  CDATB1W0D2;
  CDATB1W0D3;
  CDATB1W0D4;
  CDATB1W0D5;
  CDATB1W0D6;
  CDATB1W0D7;
  CDATB1W0D8;
  CDATB1W0D9#

CDB1W1 10;
  CDATB1W1D0;
  CDATB1W1D1;
  CDATB1W1D2;
  CDATB1W1D3;
  CDATB1W1D4;
  CDATB1W1D5;
  CDATB1W1D6;
  CDATB1W1D7;
  CDATB1W1D8;
  CDATB1W1D9#

CDB1W2 10;
  CDATB1W2D0;
  CDATB1W2D1;
  CDATB1W2D2;
  CDATB1W2D3;
  CDATB1W2D4;
  CDATB1W2D5;
  CDATB1W2D6;
  CDATB1W2D7;
  CDATB1W2D8;
  CDATB1W2D9#
```

14.2 MD (Continued)

```
CDB1W3 10;
  CDATB1W3D0;
  CDATB1W3D1;
  CDATB1W3D2;
  CDATB1W3D3;
  CDATB1W3D4;
  CDATB1W3D5;
  CDATB1W3D6;
  CDATB1W3D7;
  CDATB1W3D8;
  CDATB1W3D9#
```

14.2 MD (Continued)

```
CDATAB0 40;
  CDATB0W0D0;
  CDATB0W0D1;
  CDATB0W0D2;
  CDATB0W0D3;
  CDATB0W0D4;
  CDATB0W0D5;
  CDATB0W0D6;
  CDATB0W0D7;
  CDATB0W0D8;
  CDATB0W0D9;
  CDATB0W1D0;
  CDATB0W1D1;
  CDATB0W1D2;
  CDATB0W1D3;
  CDATB0W1D4;
  CDATB0W1D5;
  CDATB0W1D6;
  CDATB0W1D7;
  CDATB0W1D8;
  CDATB0W1D9;
  CDATB0W2D0;
  CDATB0W2D1;
  CDATB0W2D2;
  CDATB0W2D3;
  CDATB0W2D4;
  CDATB0W2D5;
  CDATB0W2D6;
  CDATB0W2D7;
  CDATB0W2D8;
  CDATB0W2D9;
  CDATB0W3D0;
  CDATB0W3D1;
  CDATB0W3D2;
  CDATB0W3D3;
  CDATB0W3D4;
  CDATB0W3D5;
  CDATB0W3D6;
  CDATB0W3D7;
  CDATB0W3D8;
  CDATB0W3D9#
```

14.2 MD (Continued)

```
CDATAB1 40;
  CDATB1W0D0;
  CDATB1W0D1;
  CDATB1W0D2;
  CDATB1W0D3;
  CDATB1W0D4;
  CDATB1W0D5;
  CDATB1W0D6;
  CDATB1W0D7;
  CDATB1W0D8;
  CDATB1W0D9;
  CDATB1W1D0;
  CDATB1W1D1;
  CDATB1W1D2;
  CDATB1W1D3;
  CDATB1W1D4;
  CDATB1W1D5;
  CDATB1W1D6;
  CDATB1W1D7;
  CDATB1W1D8;
  CDATB1W1D9;
  CDATB1W2D0;
  CDATB1W2D1;
  CDATB1W2D2;
  CDATB1W2D3;
  CDATB1W2D4;
  CDATB1W2D5;
  CDATB1W2D6;
  CDATB1W2D7;
  CDATB1W2D8;
  CDATB1W2D9;
  CDATB1W3D0;
  CDATB1W3D1;
  CDATB1W3D2;
  CDATB1W3D3;
  CDATB1W3D4;
  CDATB1W3D5;
  CDATB1W3D6;
  CDATB1W3D7;
  CDATB1W3D8;
  CDATB1W3D9#
```

14.2 MD (Continued)

```
WECC 32;
  WDPIPE2-31;
  WDPIPE2-30;
  WDPIPE2-29;
  WDPIPE2-28;
  WDPIPE2-27;
  WDPIPE2-26;
  WDPIPE2-25;
  WDPIPE2-24;
  WDPIPE2-23;
  WDPIPE2-22;
  WDPIPE2-21;
  WDPIPE2-20;
  WDPIPE2-19;
  WDPIPE2-18;
  WDPIPE2-17;
  WDPIPE2-16;
  WDPIPE2-15;
  WDPIPE2-14;
  WDPIPE2-13;
  WDPIPE2-12;
  WDPIPE2-11;
  WDPIPE2-10;
  WDPIPE2-09;
  WDPIPE2-08;
  WDPIPE2-07;
  WDPIPE2-06;
  WDPIPE2-05;
  WDPIPE2-04;
  WDPIPE2-03;
  WDPIPE2-02;
  WDPIPE2-01;
  WDPIPE2-00#
```

14.2 MD (Continued)

```
CDATAB1W3 10;
  CDATB1W3D9;
  CDATB1W3D8;
  CDATB1W3D7;
  CDATB1W3D6;
  CDATB1W3D5;
  CDATB1W3D4;
  CDATB1W3D3;
  CDATB1W3D2;
  CDATB1W3D1;
  CDATB1W3D0#

CDATAB0W3 10;
  CDATB0W3D9;
  CDATB0W3D8;
  CDATB0W3D7;
  CDATB0W3D6;
  CDATB0W3D5;
  CDATB0W3D4;
  CDATB0W3D3;
  CDATB0W3D2;
  CDATB0W3D1;
  CDATB0W3D0#

CDATAB1W2 10;
  CDATB1W2D9;
  CDATB1W2D8;
  CDATB1W2D7;
  CDATB1W2D6;
  CDATB1W2D5;
  CDATB1W2D4;
  CDATB1W2D3;
  CDATB1W2D2;
  CDATB1W2D1;
  CDATB1W2D0#
```

14.2 MD (Continued)

```
CDATAB0W2 10;
  CDATB0W2D9;
  CDATB0W2D8;
  CDATB0W2D7;
  CDATB0W2D6;
  CDATB0W2D5;
  CDATB0W2D4;
  CDATB0W2D3;
  CDATB0W2D2;
  CDATB0W2D1;
  CDATB0W2D0#

CDATAB1W1 10;
  CDATB1W1D9;
  CDATB1W1D8;
  CDATB1W1D7;
  CDATB1W1D6;
  CDATB1W1D5;
  CDATB1W1D4;
  CDATB1W1D3;
  CDATB1W1D2;
  CDATB1W1D1;
  CDATB1W1D0#

CDATAB0W1 10;
  CDATB0W1D9;
  CDATB0W1D8;
  CDATB0W1D7;
  CDATB0W1D6;
  CDATB0W1D5;
  CDATB0W1D4;
  CDATB0W1D3;
  CDATB0W1D2;
  CDATB0W1D1;
  CDATB0W1D0#
```

14.2 MD (Continued)

```
  CDATAB1W0 10;
    CDATB1W0D9;
    CDATB1W0D8;
    CDATB1W0D7;
    CDATB1W0D6;
    CDATB1W0D5;
    CDATB1W0D4;
    CDATB1W0D3;
    CDATB1W0D2;
    CDATB1W0D1;
    CDATB1W0D0#

  CDATAB0W0 10;
    CDATB0W0D9;
    CDATB0W0D8;
    CDATB0W0D7;
    CDATB0W0D6;
    CDATB0W0D5;
    CDATB0W0D4;
    CDATB0W0D3;
    CDATB0W0D2;
    CDATB0W0D1;
    CDATB0W0D0#

REGEND;
```

14.3 YMA

```
CHAIN YMA;

%*************************************************************************%
% chip: CLKMNT          location: R6F8                                    %
%                                                                         %
% function: CLOCK MAINTENANCE ARRAY                                       %
%*************************************************************************%

    ERRORIG             1;      % error ignore
    OVERRUN             1;      % over run
    COUNTERFF           1;      % counter f/f
    CLKBAD              1;      % clock bad
    SKEWLOW             1;      % skew low f/f
    SKEWHI              1;      % skew hi f/f
    MODIFYSKEW          1;      % modify skew f/f
    SKEWREG             9;      % skew register
% 16 bits

%*************************************************************************%
% chip: STOP            location: D2K6                                    %
%                                                                         %
% function: MASTER STOP LOGIC ARRAY                                       %
%*************************************************************************%

        MANDFF          1;      % bit 95
        MORFF           1;      % bit 94
        MPRETRIG        1;      % bit 93
        MOUT67          1;      % bit 92        load f/f
        MOUT66          1;      % bit 91        load f/f
        MIOC65          1;      % bit 90        set-clear f/f
        MIOC64          1;      % bit 89        set-clear f/f
        MINB63          1;      % bit 88
        MINB62          1;      % bit 87
        MINB61          1;      % bit 86
        MINB60          1;      % bit 85
        MINB59:58       2;      % bits 84:83
        MINB57:56       2;      % bits 82:81
        MINB55:48       8;      % bits 80:73
        MINB47          1;      % bit 72
        MINB46          1;      % bit 71
        MINB45:36      10;      % bits 70:61
        MIOB35          1;      % bit 60
        MIOB34          1;      % bit 59
```

14.3 YMA (Continued)

```
        MIOB33          1;      % bit 58
        MIOB32          1;      % bit 57
        MIOB31:28       4;      % bits 56:53
        MIOB27          1;      % bit 52
        MIOB26          1;      % bit 51
        MIOB25          1;      % bit 50
        MIOB24          1;      % bit 49
        MINA23          1;      % bit 48
        MINA22:8       15;      % bits 47:33
        MIOA7           1;      % bit 32
        MIOA6           1;      % bit 31
        MIOA5           1;      % bit 30
        MIOA4           1;      % bit 29
        MIOA3           1;      % bit 28
        MIOA2           1;      % bit 27
        MIOA1           1;      % bit 26
        MIOA0           1;      % bit 25
        MBLKEN19        1;      % bit 24
        MBLKEN18        1;      % bit 23
        MBLKEN17        1;      % bit 22
        MBLKEN16        1;      % bit 21
        MBLKEN15        1;      % bit 20
        MBLKEN14        1;      % bit 19
        MBLKEN13        1;      % bit 18
        MBLKEN12        1;      % bit 17
        MBLKEN11        1;      % bit 16
        MBLKEN10        1;      % bit 15
        MBLKEN9:6       1;      % bit 14
        MBLKEN5         1;      % bit 13
        MBLKEN4-3       1;      % bit 12
        MBLKEN4-2       1;      % bit 11
        MBLKEN4-1       1;      % bit 10
        MBLKEN4-0       1;      % bit  9
        MBPSTOPEN       1;      % bit 8 = ENABLE BACKPLANE STOP
        MLOCALEN        1;      % bit 7 = ENABLE LOCAL STOP
        MC8NOTSEL       1;      % bit 6
        MCOREN          1;      % bit 5 = ENABLE C-TEST OR
        MBOREN          1;      % bit 4 = ENABLE B-TEST OR
        MAOREN          1;      % bit 3 = ENABLE A-TEST OR
        MCANDEN         1;      % bit 2 = ENABLE C-TEST AND
        MBANDEN         1;      % bit 1 = ENABLE B-TEST AND
        MAANDEN         1;      % bit 0 = ENABLE A-TEST AND
%                      ---
%                     96 bits
%
```

14.3 YMA (Continued)

```
%*************************************************************************%
% chip: STOP              location: P2K6                                  %
%                                                                         %
% function: SLAVE STOP LOGIC ARRAY                                        %
%*************************************************************************%
%
        SANDFF          1;      % bit 95
        SORFF           1;      % bit 94
        SPRETRIG        1;      % bit 93
        SOUT67          1;      % bit 92        load f/f
        SOUT66          1;      % bit 91        load f/f
        SIOC65          1;      % bit 90        set-clear f/f
        SIOC64          1;      % bit 89        set-clear f/f
        SINB63          1;      % bit 88
        SINB62          1;      % bit 87
        SINB61          1;      % bit 86
        SINB60          1;      % bit 85
        SINB59          1;      % bit 84
        SINB58          1;      % bit 83
        SINB57:56       2;      % bits 82:81
        SINB55:48       8;      % bits 80:73
        SINB47          1;      % bit 72
        SINB46          1;      % bit 71
        SINB45:36      10;      % bits 70:61
        SIOB35:32       4;      % bits 60:57
        SIOB31:28       4;      % bits 56:53
        SIOB27:24       4;      % bits 52:49
        SINA23          1;      % bit 48
        SINA22:8       15;      % bits 47:33
        SIOA7           1;      % bit 32
        SIOA6           1;      % bit 31
        SIOA5           1;      % bit 30
        SIOA4           1;      % bit 29
        SIOA3           1;      % bit 28
        SIOA2           1;      % bit 27
        SIOA1           1;      % bit 26
        SIOA0           1;      % bit 25
        SBLKEN19        1;      % bit 24
        SBLKEN18        1;      % bit 23
        SBLKEN17        1;      % bit 22
        SBLKEN16        1;      % bit 21
        SBLKEN15        1;      % bit 20
        SBLKEN14        1;      % bit 19
```

14.3 YMA (Continued)

```
        SBLKEN13        1;      % bit 18
        SBLKEN12        1;      % bit 17
        SBLKEN11        1;      % bit 16
        SBLKEN10        1;      % bit 15
        SBLKEN9:6       1;      % bit 14
        SBLKEN5         1;      % bit 13
        SBLKEN4-3       1;      % bit 12
        SBLKEN4-2       1;      % bit 11
        SBLKEN4-1       1;      % bit 10
        SBLKEN4-0       1;      % bit 9
        SBPSTOPEN       1;      % bit 8 = ENABLE BACKPLANE STOP
        SLOCALEN        1;      % bit 7 = ENABLE LOCAL STOP
        SC8NOTSEL       1;      % bit 6
        SCOREN          1;      % bit 5 = ENABLE C-TEST OR
        SBOREN          1;      % bit 4 = ENABLE B-TEST OR
        SAOREN          1;      % bit 3 = ENABLE A-TEST OR
        SCANDEN         1;      % bit 2 = ENABLE C-TEST AND
        SBANDEN         1;      % bit 1 = ENABLE B-TEST AND
        SAANDEN         1;      % bit 0 = ENABLE A-TEST AND
% 96 bits              -----
%                      208 bits total

CHNEND; % YMA
```

14.3 YMA (Continued)

```
REGLIST YMA;

        SPYSRC 3;               % SPYSOURCE = REQUESTOR OF SPY OPERATION
                MIOA2;
                MIOA1;
                MIOA0#

        SPYPROC  1;             % PROCESSOR/IO PART OF SPYSOURCE
                MIOA2#

        ENSPYPROC 1;            % TEST PROCESSOR/IO PART OF SPYSOURCE
                MBLKEN4-2#

        SPYCAB   1;             % CABINET A/B PART OF SPYSOURCE
                MIOA1#

        ENSPYCAB  1;            % TEST CABINET A/B PART OF SPYSOURCE
                MBLKEN4-1#

        SPYSLOT  1;             % SLOT 1/0 PART OF SPYSOURCE
                MIOA0#

        ENSPYSLOT 1;            % TEST SLOT 1/0 PART OF SPYSOURCE
                MBLKEN4-0#

        SPYCMD 2;               % SPY COMMAND = REQUESTED OPERATION
                MIOA5;
                MIOA4#

        ENSPYCMD 1;             % TEST SPY COMMAND
                MBLKEN5#

        SPYADDR 2;              % SPY ADDRESS
                MINA22:8;
                MINB45:36#

        SPYKEY  1;              % KEY PORTION OF SPY ADDRESS
                MINA22:8#

        SPYSET  1;              % SET PORTION OF SPY ADDRESS
                MINB45:36#
```

14.3 YMA (Continued)

```
SPYVALID 3;              % SPY VALID QUALIFIER (WHEN = 7)
        MIOA6;
        MINB47;
        MINA23#

ENSPYKEY 1;              % TEST SPYKEY AND SPYVALID
        MBLKEN9:6#

ENSPYSET 3;              % TEST SPYSET (WHEN = 7)
        MBLKEN15;
        MBLKEN14;
        MBLKEN13#

RBUSERR 2;               % MRBUS ERROR VECTOR
        MIOB32;
        MIOB31:28#

RBUSFATAL 1;             % FATAL PART OF MRBUS ERROR VECTOR
        MIOB32#

RBUSVECTOR 1;            % NUMBER PART OF MRBUS ERROR VECTOR
        MIOB31:28#

ENRBUSFAT 1;             % TEST RBUS ERROR (FATAL PART)
        MBLKEN12#

ENRBUSERR 1;             % TEST RBUS ERROR (NUMBER PART)
        MBLKEN11#

WRTABORT 1;              % IS MCACM IN WRITE ABORT STATE?
        MOUT67#

WRTKEY 6;                % DIRECTORY WRITE (KEY PART)
        MINB62;
        MINB61;
        MINB60;
        MINB59:58;
        MINB57:56;
        MINB55:48#

WKEYRAMBlL 1;            % WRITE KEY RAM DIR-B BLOCK 1
        SIOA4#           % --> STOPCOMP8
```

14.3 YMA (Continued)

```
WKEYRAMB0L 1;           % WRITE KEY RAM DIR-B BLOCK 0
        MIOA3#          % --> STOPCOMP8

WRTENABLE 2;            % WRITE ENABLE TO DIRECTORY RAMS
        SINB59;         % <-- STOPCOMP8
        MINB63#         % <-- STOPCOMP8

ENKEYWE 2;              % CHECK ON KEY WRITE ENABLE (WHEN = 3)
        MBLKEN4-3;
        SBLKEN5#

WRTSET 2;               % DIRECTORY WRITE (SET PART)
        SINB57:56;
        SINB55:48#

ENWRTKEY 4;             % ENABLE WRTKEY TEST (WHEN = F)
        MBLKEN19;
        MBLKEN18;
        MBLKEN17;
        MBLKEN16#

ENWRTSET 3;             % ENABLE WRTSET TEST (WHEN = 7)
        SBLKEN18;
        SBLKEN17;
        SBLKEN16#

REQCMD 2;               % MEMORY REQUEST COMMAND
        SIOA1;
        SIOA0#

ENREQCMD 2;             % ENABLE REQCMD TEST (WHEN = 3)
        SBLKEN4-1;
        SBLKEN4-0#

REQIOM 1;               % MEMORY REQUEST IO/MEMORY
        SIOA2#

ENREQIOM 1;             % ENABLE REQIOMEM TEST
        SBLKEN4-2#

REQOBT 1;               % MEMORY REQUEST OBTAINED
        SIOA3#
```

14.3 YMA (Continued)

```
ENREQOBT 1;              % ENABLE REQOBT TEST
        SBLKEN4-3#

REQKEY 1;                % MEMORY REQUEST ADDRESS (KEY PART)
        SINA22:8#

REQUEST 2;               % MEMORY REQUEST VALID (WHEN = 3)
        SINB47;
        SINA23#

REQSET 1;                % MEMORY REQUEST ADDRESS (SET PART)
        SINB45:36#

ENREQKEY 1;              % ENABLE REQKEY & REQUEST TESTS
        SBLKEN9:6#

ENREQSET 3;              % ENABLE REQSET TEST (WHEN = 7)
        SBLKEN15;
        SBLKEN14;
        SBLKEN13#

ENDGTWRT 1;              % ENABLE STOPING ON DIGIT WRITE
        SBLKEN10#

WORDSEL 1;               % STOP ON WRITE DIGIT; WORD SELECT
        SIOB27:24#

DIGIT 1;                 % STOP ON WRITE DIGIT; DIGIT SELECT
        SIOB31:28#

VALUE 1;                 % STOP ON WRITE DIGIT; VALUE
        SIOB35:32#

ORWRTABRT 1;             % STOP ON OTHER CONDITIONS OR WRTABRT
        SINB60#

ANWRTABRT 1;             % STOP ON OTHER CONDITIONS AND WRTABRT
        SINB61#
```

14.3 YMA (Continued)

```
        ORFLUSH 1;              % STOP ON OTHER CONDITIONS OR FLUSH
                SINB62#

        ANDFLUSH 1;             % STOP ON OTHER CONDITIONS AND FLUSH
                SINB63#

        RANDOM 4;               % RANDOM-0 IS BROKEN DISABLE
                MIOB27;
                MIOB26;
                MIOB25;
                MIOB24#

        DISABROKEN 1;           % DISABLE BROKEN REPORTING FROM MCACM
                MIOB24#

REGEND; % YMA
```

14.4 YMD

```
CHAIN YMD;

%*************************************************************************%
% chip: CLKMNT          location: R6F8                                    %
%                                                                         %
% function: CLOCK MAINTENANCE ARRAY                                       %
%*************************************************************************%

    ERRORIG             1;      % error ignore
    OVERRUN             1;      % over run
    COUNTERFF           1;      % counter f/f
    CLKBAD              1;      % clock bad
    SKEWLOW             1;      % skew low f/f
    SKEWHI              1;      % skew hi f/f
    MODIFYSKEW          1;      % modify skew f/f
    SKEWREG             9;      % skew register
% 16 bits

CHNEND; % YMA


REGLIST YMA;
    BADCLK      1;              % Dummy to account for poor MP software.
        CLKBAD#
REGEND; % YMA
```