Burroughs Corporation

ENTRY/MEDIUM SYSTEMS GROUP
PASADENA DEVELOPMENT CENTER

I/O MEMORY CONCENTRATOR

ENGINEERING DESIGN SPECIFICATION

COMPANY CONFIDENTIAL

REVISIONS

REV. LTR	CONTROL DOC & NO	PAGES REVISED, ADDED, DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	ECN 51540	Initial Issue JUNE 23, 1987	L. Simpson M. Scruggs S. Munroe	R. Young 5/11/87 A. Gomez 5/20/87 D. Welbourn 6/3/87 H. Li 6/3/87

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF ILLUSTRATIONS

1 INTRODUCTION

1.1 PREFACE

This system design specification describes the Input Output Memory Concentrator (IOMC) for the V500 V-Series processor.

The IOMC is the interface between the V500 processor/memory and the V500 I/O subsytem.

1.2 RELATED DOCUMENTS

SPEC NO.	NAME
1993 5162	V500 System
1993 5220	V500 Memory Control and Cache Module
1993 5238	V500 Memory Data Card
1993 5329	V500 Data Transfer Module
1993 5303	V500 Maintenance Subsystem
1993 5279	V500 Architecture
1993 5337	Fault Detection Design Recommendations
1995 5301	V500 Circuit Rules
1998 5076	V500 Intercabinet Buffer Module

2 GENERAL DESCRIPTION

The IOMC is the link between the I/O sub-system and the processor. Its main function is to interface the I/O and memory busses. It also provides a one microsecond clock for time keeping on the XM, a Time of Day counter, an XM to I/O interrupt and an I/O to XM interrupt.

As a bus interface, the IOMC connects the 32 bit, TTL I/O bus to the 160 bit, ECL memory bus. This includes changing the electrical signal levels, aligning and concatenating the data, and buffering of data to allow for the different bus bandpasses. The IOMC also manages its memory read/write address. The IOMC can address 500 MB of memory.

Figure 2-1 shows the placement of the IOMC in the system. A V500 system may contain one or two IOMCs in a master/slave configuration. This is not shown in the figure.

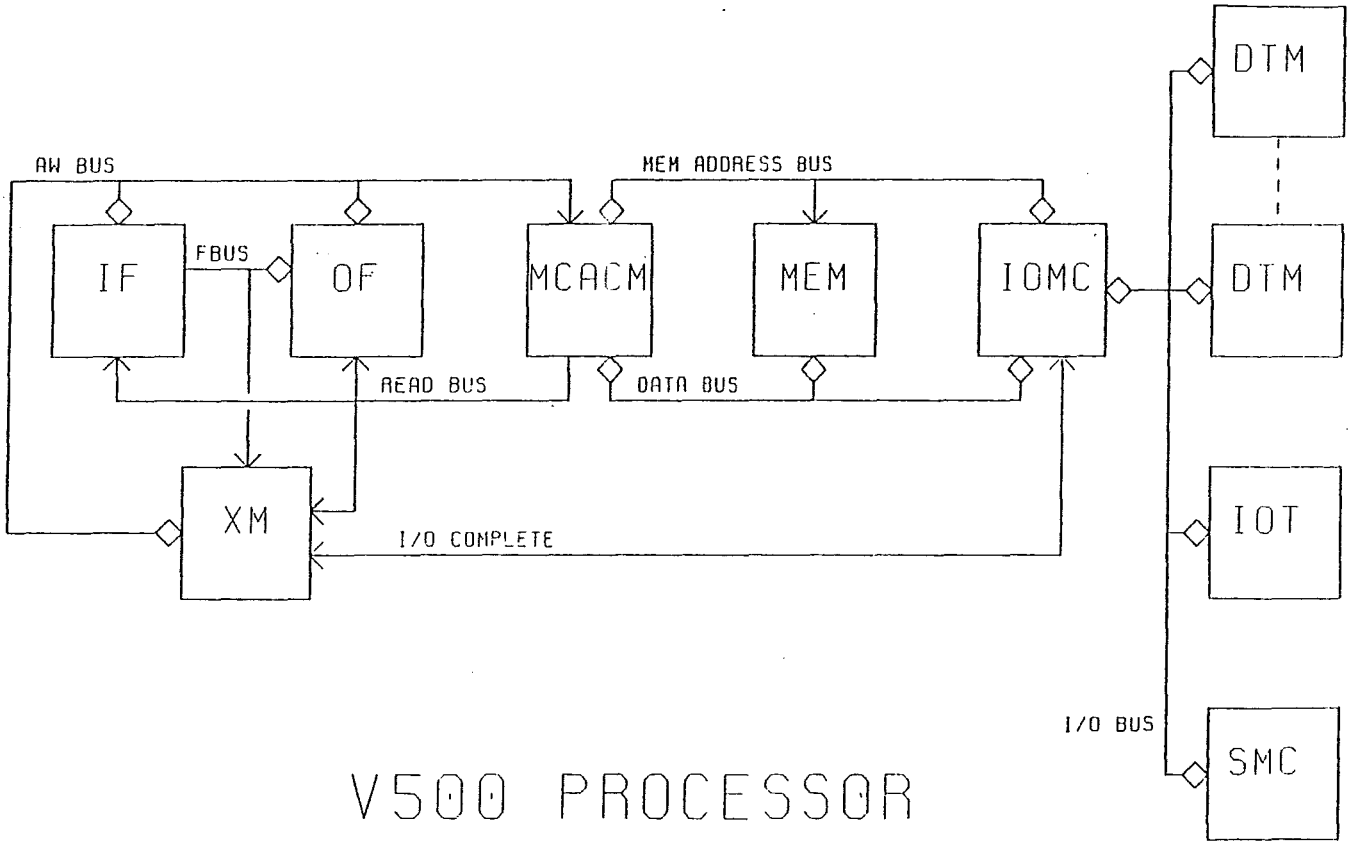


FIGURE 2-1 V500 PROCESSOR

3 DETAILED DESCRIPTION

MODULE and GATE ARRAY NAMES

IF	Instruction Fetch
OF	Operand Fetch
XM	Execute Module
MCACM	Memory Control and Cash Module
MDC	Memory Data Card (MEM)
IOMC	I/O Memory Concetrator
DTM	Data Transfer Module
IOT	I/O Translator
SMC	System Maintenance Controller
ECCIO	TTL Data and ECC transceiver
BASADD	Base Adder
MCNTL5	Memory Control
ECCXCVR	Memory Address and ECC transceiver
RAMROT	Data Ram and Rotater
MDECC	Memory Data and ECC transceiver

3.1 BLOCK DIAGRAM

The following is a general description of the blocks that make up the IOMC as shown on the block diagram in Figure 3-1.

3.1.1 I/O BUS INTERFACE

The I/O Bus Interface connects the IOMC to the SMC and the I/O subsystem's DTMs and IOTs. The I/O Bus Control handles all bus protocol and the I/O Bus register. The I/O Bus Control is a subset of the logic used on the IOTs and DTMs. The main differences are that the IOMC is never busy while the I/O Bus is idle, and the IOMC never initiates a bus transaction. See the I/O Bus specification or the interface section of this specification for a detailed description of how the bus works.

3.1.2 ADDRESS CARD DATA PATH

The data path on IOMCA is bidirectional. Using an ECCIO array it provides a 39 bit data and ECC register, ECC generation and checking for the I/O Bus. Additional logic provides byte parity generation and checking for the data card, ECL/TTL translation, ASCII/EBCDIC translation, and a path for loading the memory address.

3.1.3 IOMCA DATA PATH SEQUENCER

The IOMCA data path sequencer is a prom-based micro-sequencer that provides IOMCA data path control, control for part of the IOMCD data path, I/O Bus control interface, and memory address control. This control logic decodes the commands from the I/O bus, informs the rest of the module of the command, controls the loading of the BCD memory address register, and controls the reading or writing of the bandpass buffer on the data card for memory reads and writes. It also ensures that the required timing of the I/O bus controller is met, and checks for errors in the memory data transmission.

3.1.4 MEMORY ADDRESS AND INTERFACE

This logic contains the current memory address and the memory bus control interface. Rather than increment the beginning address, an offset counter is incremented and added to the beginning address. The memory address is converted to binary before the add. The convert and add are done by a BASADD array. The MCNTL5 array provides the memory bus control interface and address mapping. The Memory address bus interface consists of two ECCXCVR arrays. For a more detailed description of this logic, refer to the interface section of this specification.

3.1.5 XM INTERRUPTER

The XM interrupter is a small state machine that receives an I/O Complete (IOC) command from a DTM or IOT, and interrupts one of the one to four XMs in the system. The IOC command specifies which XM to try first and what the IOC type (normal, error, real) is. For a more detailed description of the XM interupter, refer to the XM interface section of this specification.

3.1.6 1 MICROSECOND TIMER

The timer generates a one microsecond reference signal that is used by the XM for its Task Timer and Timeout counters. The signal also increments the Time of Day counter on IOMCD. The signal is generated by dividing the system clock, creating a true condition every 1 microsecond. The duty cycle of this signal is 50%. The duty cycle and frequency can be changed by changing PROM patterns. This is necessary in case the frequency of the system clock changes at a later date. Since the system clock frequency is not evenly divisable into 1 microsecond, an error will result in the frequency of the timer. The greatest error that occurs in any given cycle of the timer is one TTL system clock period. The timer corrects this on the next count. The accuracy of the 1 microsecond timer also depends on the accuracy of the TTL system clock. The timer circuit is implemented in TTL MSI logic and the output is level changed to an ECL signal before it is output to the backplane. In a two IOMC configuration, the master IOMC sources the signal, while the signal from the slave is inhibited.

3.1.7 MAILBOX/TIME OF DAY SEQUENCER

The mailbox (hardware register) and time-of-day sequencer is a prom-based microsequencer which controls the memory bus access of the Mailbox register and the time-of-day counter on IOMCD. This sequencer, located on the address card, is capable of writing to or reading from the time-of-day counter, or writing to the mailbox register at the command given to the IOMC from the memory bus. This sequencer decodes the commands on the memory bus, determines if the command is for the IOMC, and reacts to the command.

3.1.8 ERROR and MAINTENANCE LOGIC

This logic includes the clock, shift, reset, event stop, and error handling logic. The IOMC stop logic monitors initiator and responder addresses on the IO bus and the data path between IOMCA and IOMCD. This provides visibility of the data on the I/O bus along with the IOMC's data path. The IOMC stop logic also monitors the binary memory address on IOMCA and the byte select logic on IOMCD.

3.1.9 IOMCD DATA PATH

Including RAMROT, MAILBOX, and PARITY LOGIC

The IOMCD data path has a bidirectional dual port 16 entry by 40-bit ram, rotate and concatenate logic to convert the 32 bit I/O bus format to the 160 bit memory bus format and vice versa, byte parity and bus ECC generation and checking, and a 40 bit mailbox register (MBR). Much of this logic is contained in eight slices of the Ram and Rotate (RAMROT) array. For a complete description of how this block converts the 32-bit I/O bus format to the 160-bit memory bus format, refer to the Data Rotation and Concatenation section of this specification.

3.1.10 DATA MDECC

The memory data register is a 160 bit, bidirectional register with ECC generation and checking. The register consists of 8 MDECC arrays. This register is a direct interface to the memory data bus and is identical to the interface used on the Memory Data Cards (MDC). A pair of MDECC arrays is used to form one of four 40-bit data bus words (0, 1, 2 and 3). Eight bits of ECC is included with each word for a total count of 48 bits per MDECC pair.

3.1.11 TOD COUNTER

The Time of Day (TOD) counter is a 10 digit binary counter driven by the 1 microsecond timer. The counter counts the number of microseconds since midnight. System software uses this function as a real time clock. The TOD counter is written and read from the memory bus, and is readable from the I/O bus. Operations on this counter can occur from the memory interface and the I/O interface simultaneously.

3.1.12 MAILBOX AND TIME OF DAY MDECC

A second 40-bit memory data register for the TOD and Mailbox functions is provided by two more MDECCs. This pair of MDECCs is wire-or'ed to word 0 of the memory data bus, and is an independent interface to that bus. If the memory bus is available, the TOD and Mailbox can be accessed from the memory bus, even when an I/O memory operation is in progress.

3.1.13 IOMCD DATA SEQUENCER

The IOMCD has one sequencer. It is a prom-based micro-sequencer that controls most of the data path and related logic on the data card. This controller receives an encoded op of either read, write, or read with lock, from the IOMCA data path sequencer. It then configures the RAMROT array group for reading from or writing to by the address card, controls the commands sent to the memory bus via the memory address bus interface on the Address Card, controls the loading of the MDECC group from the backplane or the RAMROT group, controls the loading and unloading of the RAMROT to and from the MDECC group, and keeps track of how much data has been transferred to and from memory via signals from the byte and up/down counters in the RAMROT arrays.

The mailbox, TOD counter, and reading and writing the RAMROT from the I/O side are controlled by the IOMCA data path sequencer.

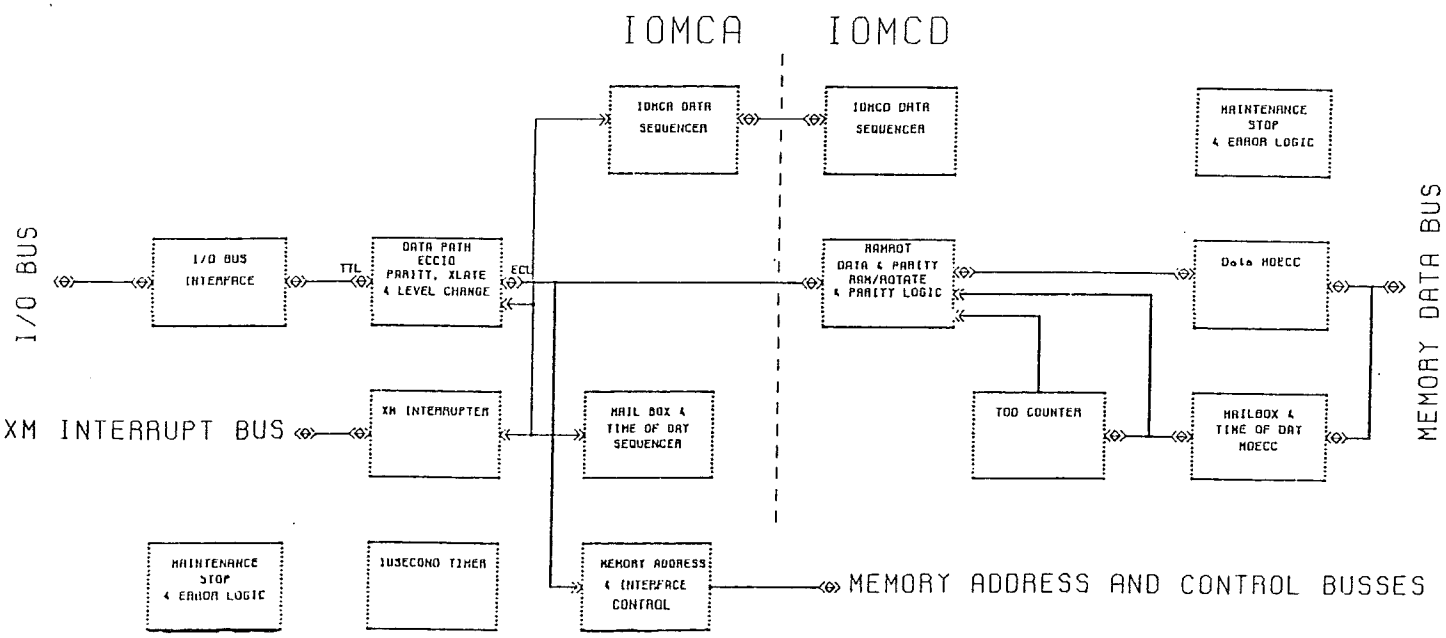


FIGURE 3-1 IOMC II BLOCK DIAGRAM/TOP

IOMCA BLOCK DIAGRAM

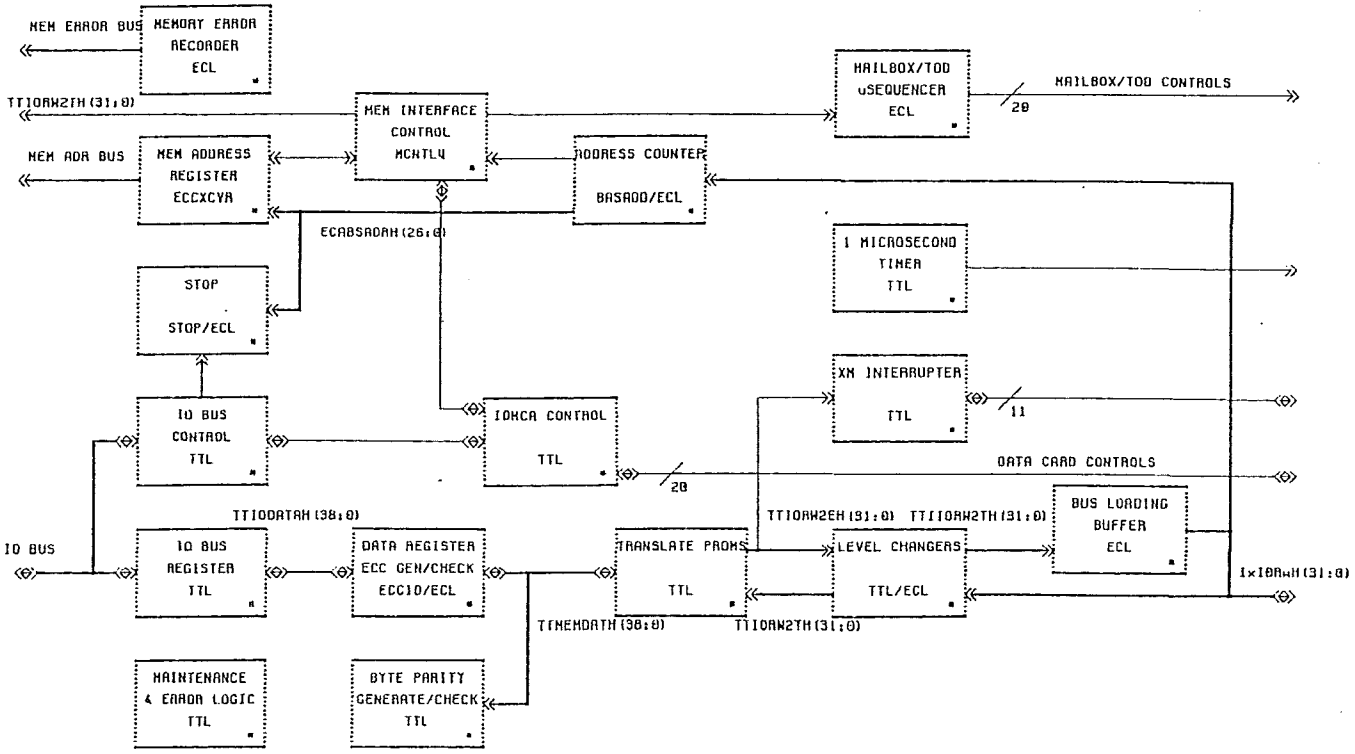


FIGURE 3-2 IOMC II BLOCK DIAGRAM/IOMCA

IOMCD BLOCK DIAGRAM

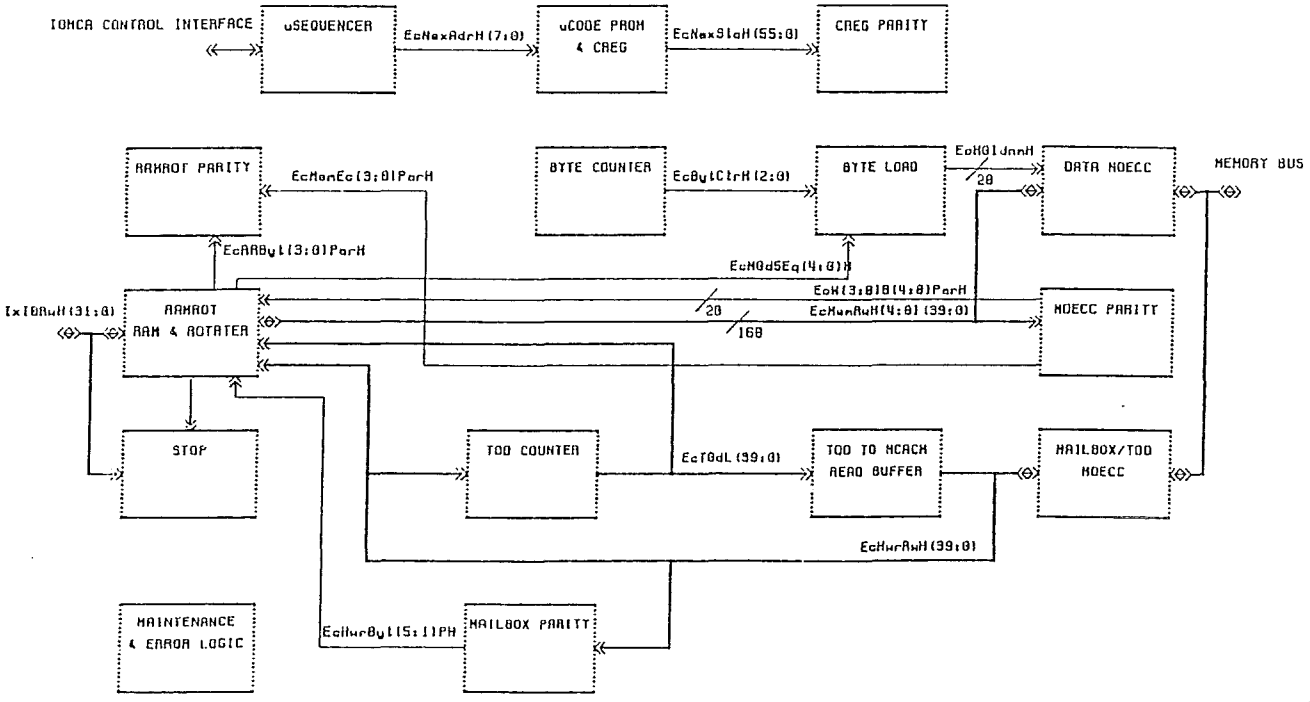
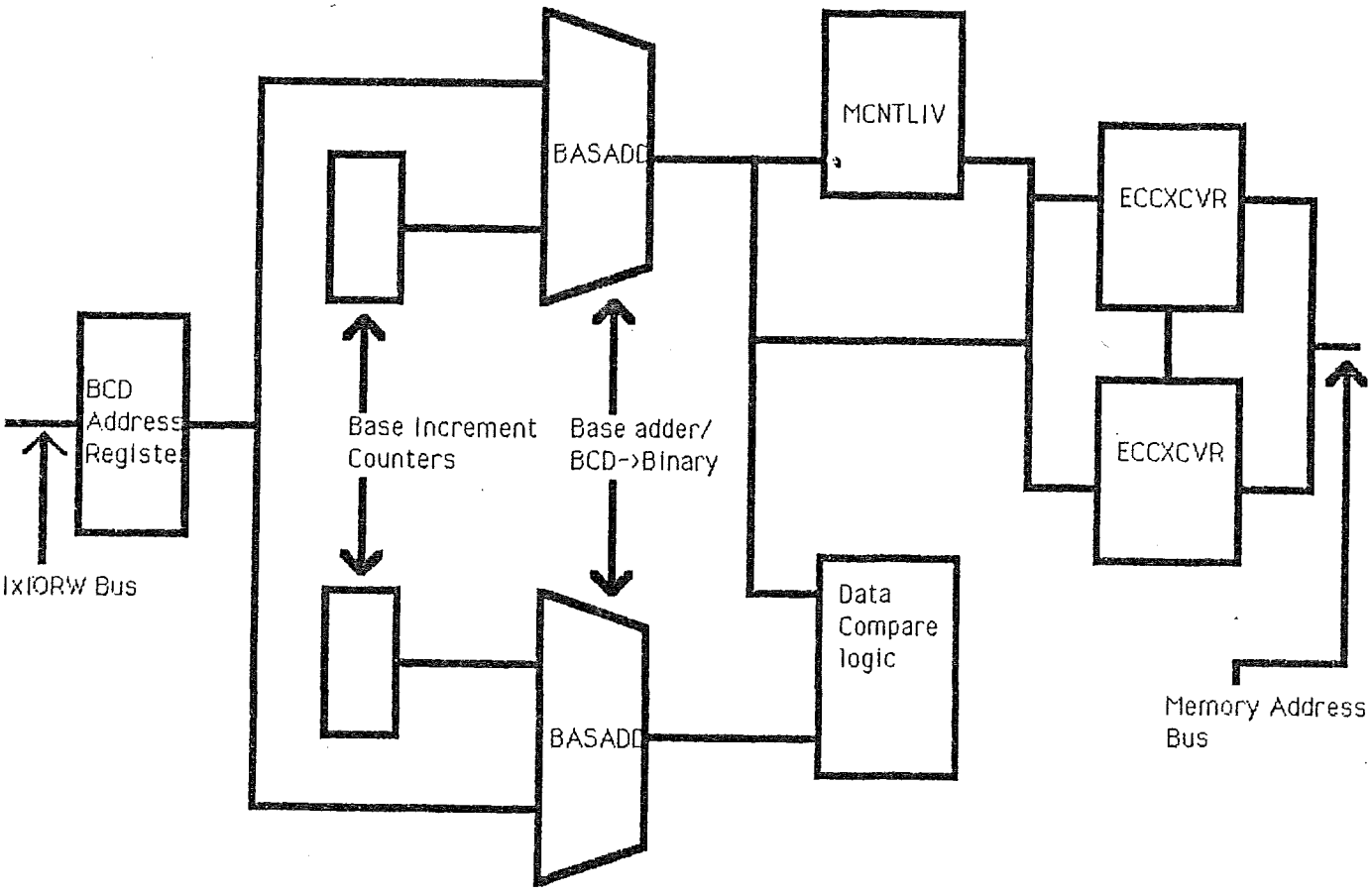


FIGURE 3-3 IOMC II BLOCK DIAGRAM/IOMCD



MEMORY ADDRESS REGISTER AND MAPPING

FIGURE 3-4 MEMORY ADDRESS REGISTER AND MAPPING

3.2 DATA ROTATION AND CONCANTENATION

The data path through the data card is responsible for modifying the 160 bits of data in the memory data registers to match the 32-bit I/O bus format and vise-versa. This is done in the 8 slices of RAMROT array. These 8 slices, henceforth called the bandpass buffer, have a 32-bit bus plus byte parity on the I/O side of the buffer, and five 32-bit busses plus byte parity to form 160 bits on the memory side. For a description of the RAMROT array, refer to appendix B of this specification.

The I/O bus consists of four bytes of data, with the most significant digit of information at bits 31 to 28 (Figure 3-5). The data alignment on the I/O bus is memory address independent. The data corresponding to the beginning memory address of an IOMC operation is always at the most significant digit of the I/O bus (bits 31 to 24), and data is read from the most significant digit to the least significant digit of the bus. For a beginning memory address of 6, the digit corresponding to memory address 6 is at bits 31 to 28. The digit corresponding to address 7 is bits 27 to 24 etc.

The memory bus is divided into four words (word 0, 1, 2 ,3), with the most significant digit at bits 0 to 3 of word 0, henceforth referred to as bits 003 to 000 where the most significant digit in the reference reflects the memory word. The memory words are aligned relating to the memory address and on modulo 40 boundaries. In other words, a block of memory in the MDECC arrays corresponds to address 0 to 8 (modulo 2) for word 0, 30 to 38 for word 3 etc., or 40 to 48 for word 0, 70 to 78 for word 3 etc. Using the previous example, the same data from address 6 on bits 31 to 28 of the I/O bus is mapped to bits 027 to 024 on the memory bus. The data of address 7 on I/O bus bits 27 to 24 would be mapped to bits 031 to 028 on the memory bus. Note that the digits are backwards from the I/O bus to the memory bus, but the bits within the digits are not.

3.2 DATA ROTATION AND CONCANTENATION (Continued)

To simplify the relationship between the I/O bus and the memory bus, the four 40-bit words of memory data can be divided into five subwords of 32 bits each. Thus, the 32 bit data words in the bandpass buffer are mapped directly to each of the five subwords. These are the five 32-bit busses on the memory side of the bandpass buffer. Subword 0 consists of bits 000 to 031, subword 1 consists of bits 032 to 123 etc. Each of the subwords is bounded on a modulo 8 address. Each location in the buffer can be mapped to any one of the five subwords using a modulo 5 counter internal to the RAMROT array, and byte mark logic external to the buffer.

A problem arises in that the I/O bus is indifferent to memory address. Regardless of the starting memory address, the data corresponding to the starting address will exist on the I/O bus at the MSD, and not on the modulo 8 boundaries required by the bandpass buffer and the 5 subwords. Mis-alignment of data by address is the reason why data rotation is necessary; I/O bus data placement is memory address independent and the memory bus data placement is address dependent.

Rotation of data is accomplished in a rotation and multiplexing circuit at the I/O side of the bandpass buffer. The rotation is by the byte, and is dependent on a 2-bit field known as the binary byte address (BBA). The BBA is decoded from the beginning address on the address card and sent over to the data card where it is used to initialize the 4 bandpass buffer read and write address counters, 1 per byte, and control the input data mux. For a memory write, the BBA tells the bandpass buffer where in the 4 byte field of the ram, the byte from the beginning address will be written into the ram. Data is rotated from this point and to the right. Thus, for the data starting at address 6, the data from address 6 would be written into the AB byte of address 0 of the ram in Figure 3-5. In order to maintain the modulo 8 boundary of the subword, and again based on the BBA, the ram write address for the next three bytes would be one more than that of the first byte, and the remaining three bytes would be written to the GH, EF, and CD bytes of address 1 of the ram.

3.2 DATA ROTATION AND CONCANTENATION (Continued)

The next word of I/O, corresponding to address 14 to 20 follows the same pattern as the first, with data from address 14 written into byte AB, ram address 1, and 16 to 20 at GH, EF, and CD, ram address 2. The result is that data for memory address 8, 10, and 12, is concatenated with data from address 14 into a single ram location. This location can then be directly written to subword 1 of the MDECC array. The beginning address byte (address 6) is in its own location of the ram along with three bytes of junk. In order to avoid writing the junk into subword 0, the byte mark logic mentioned earlier will assert the hold signal for these three bytes on the MDECC group when this subword is written. This would require a Read-Modify Write cycle on the IOMC. The pattern is repeated until all of the data is received from the I/O bus and written to memory, with the rotation and concatenation remaining constant.

For a memory read operation the sequence of events is a bit different. In this case data is written into location zero of the ram directly from MDECC subword 0, and from subword 1 to location one, etc. However when read out to the I/O bus, the ram read address counter is initialized to zero for the byte AB and one for the rest. Rotation is again based on the BBA, and the result is that byte AB of ram address 0 is concatenated with bytes GH, EF, and CD of ram address 1. Again this is continued until all the data is read to the I/O bus from memory. In this case there is no junk at the beginning of the transmission to the I/O bus as the data for the beginning address must start at bits 31 to 28 of the I/O bus. If the data at the last transmission has some bytes of junk, this junk is simply not looked at by the initiating module on the I/O bus.

Finally, if the I/O bus is for some reason faster than the memory bus due to memory bus overloading, the bandpass buffer is capable of queueing up to 14 subwords in the buffer or 15 if the data is not rotated.

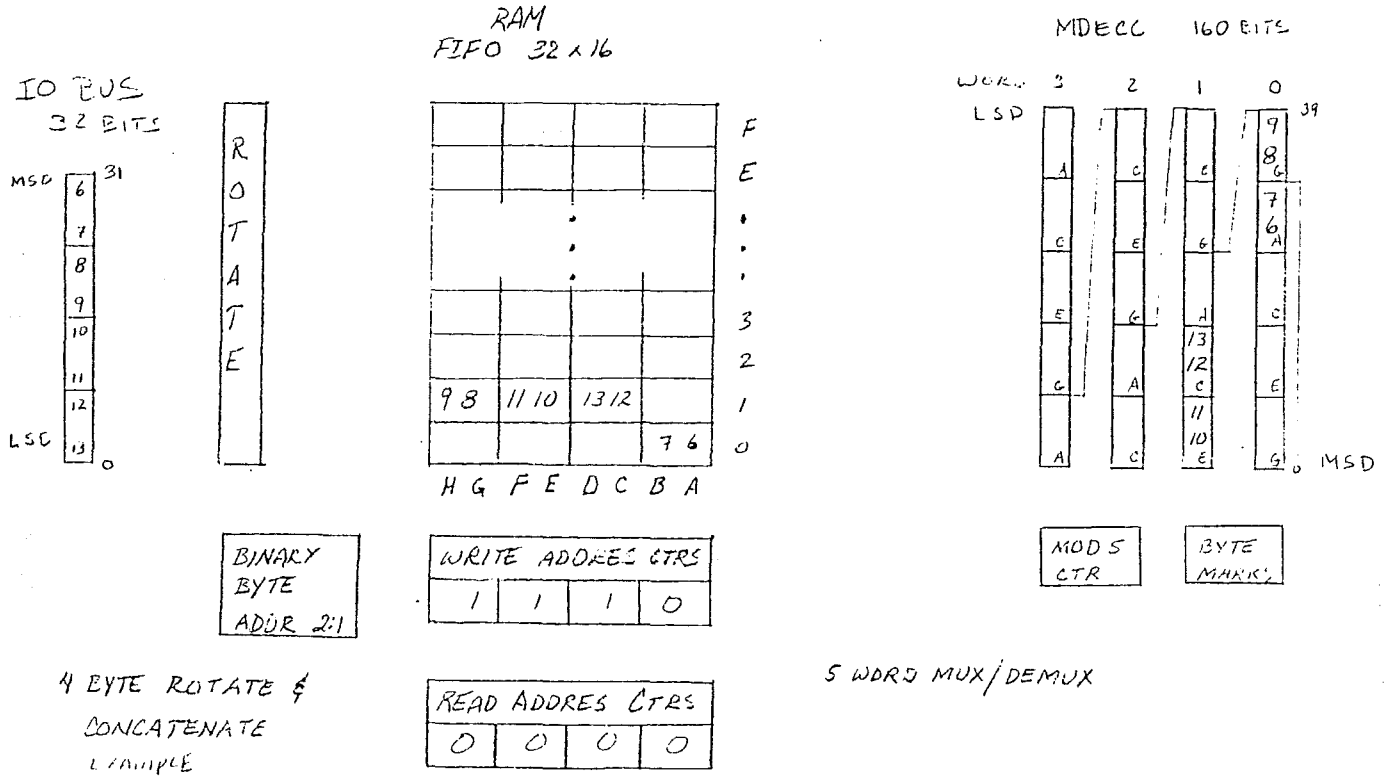


FIGURE 3-5 4 BYTE ROTATE AND CONCATENATE EXAMPLE

3.3 DATA PATH ERROR DETECTION LOGIC

3.3.1 IOMCD PARITY AND ECC LOGIC DESCRIPTION

The data card protects data with byte parity on the IOMCA interface and 8-bit ECC on the memory interface, Figure 3.7.

When reading memory, data received from the memory bus is compared with the ECC received in the MDECC arrays, block E. When the data is sent from the MDECC to the RAMROT, block D generates byte parity for the data at the MDECC output. At the same time, block G generates parity over the ECC output of the MDECC. The parity of the two should match and is checked by block C. The byte parity is stored in the RAMROT, block A, and eventually passed on to the IOMCA. If a MBE occurred during the loading of memory data and ECC from memory data bus, the Read parity error detection logic is disabled and a Soft error will be reported to DTM/IOT.

The Mailbox is protected by byte parity generated by block K. The byte parity is checked against the ECC parity by blocks H and I, and stored in the RAMROT for IOMCA.

Byte parity for the Time of Day Counter is generated within the RAMROT arrays when the Counter is read by IOMCA. The counter logic is not protected. There is no protection for the RAMROT to MDECC TOD Counter read.

3.3.2 IOMCA PARITY AND ECC LOGIC DESCRIPTION

The I/O Bus interface to IOMCA is protected by 7-bit ECC (Figure 3.6). The IOMCD interface is protected with byte parity. If the data is translated by block D, the correct byte parity is generated by the parity generators.

On a memory read, byte parity is generated, by block E, after the data from IOMCD is level changed to TTL. The new parity is checked against the parity received from IOMCD. The new parity is also kept in a register, block I, for later comparison with the ECC parity. The data is loaded into the ECCIO array, block A. As data is sent to the I/O Bus from the ECCIO, parity is generated, by block G, over the ECC generated by the ECCIO array and compared with the byte parity by block H.

The Mailbox and Time of Day reads are handled just like the memory read.

On a memory write, data received from the I/O Bus is checked against its ECC. As data is sent to IOMCD from ECCIO, byte parity is generated by block B and ECC parity is generated by block G. These two are compared by block C. The byte parity is passed on to IOMCD. If a MBE occurred on the I/O Bus, the write parity error logic will be disabled for that particular transfer.

IOMCA DATA PARITY AND ECC

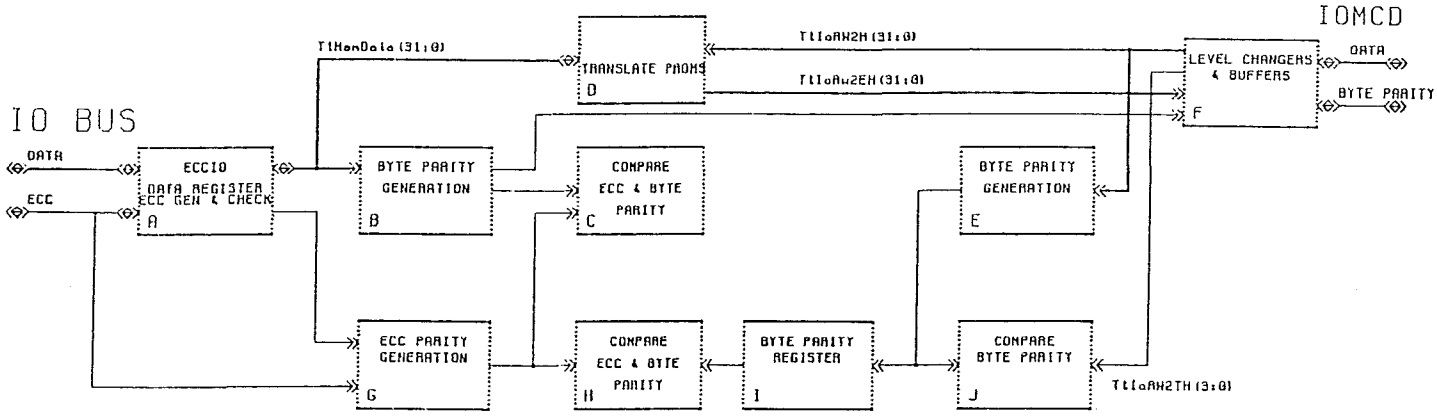


FIGURE 3-6 IOMCA DATA PARITY AND ECC

IOMCD DATA PARITY AND ECC

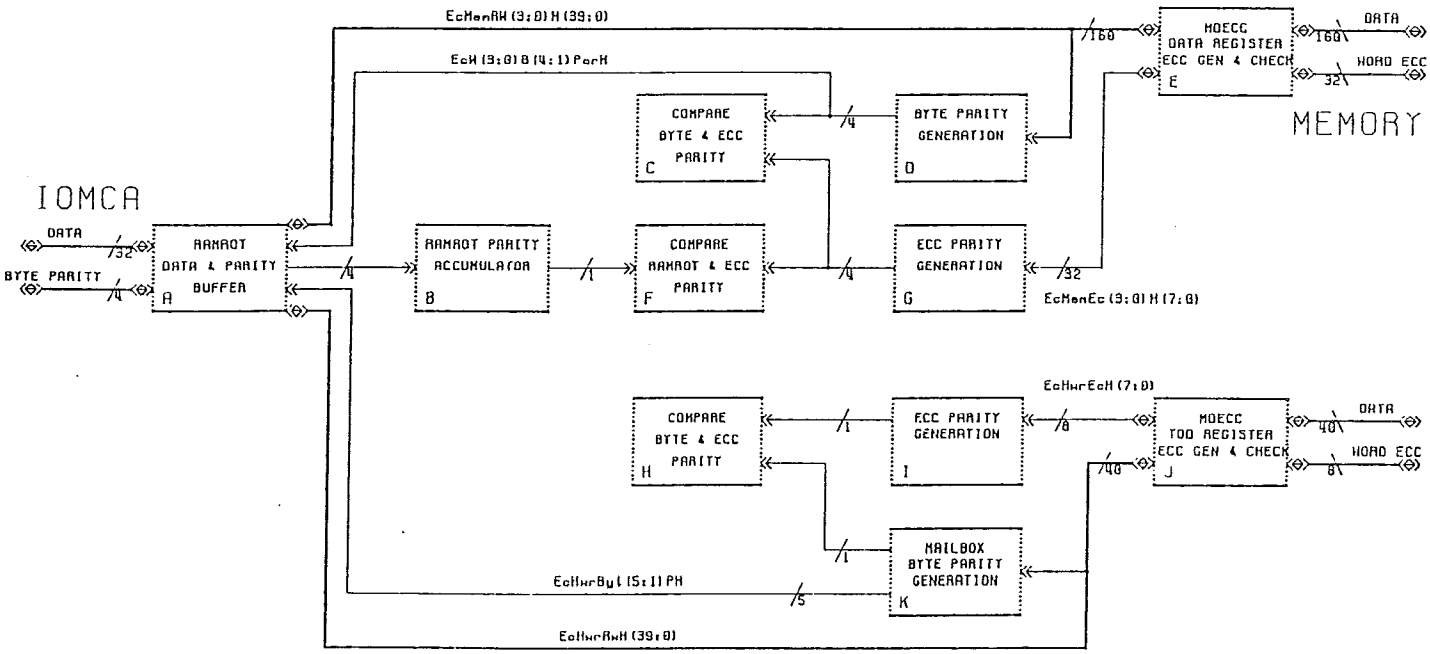


FIGURE 3-7 IOMCD DATA PARITY AND ECC

4 DETAILED COMMANDS

The IOMC will accept commands from I/O modules via the I/O bus, and commands from the Execute Module (XM) through the Memory Control and Cache Module (MCACM) via the main memory bus. Commands from I/O and the XM can be handled simultaneously by the IOMC via independent controllers within the IOMC.

4.1 XM COMMANDS

There are three commands that the IOMC can receive from the main memory bus. These are:

WRMAIL - Write Mailbox Register

WTOD - Write Time-of-Day

RTOD - Read Time-of-Day

For all IOMC commands received from the main memory bus, the IOMC emulates a Memory Data Card (MDC). From the memory bus point of view, the commands and timing to the IOMC from the XM are exactly the same as any other memory read or write, with the exception of the five bit memory selection field. For memory operations this field is 10-1F, and for I/O operations the field is 00-0F. (Refer to the section on memory bus interface for detailed timing and protocol).

4.1.1 WRMAIL - WRITE MAILBOX REGISTER

MCACM issues a Write command on the memory bus with the memory selection field set at 00-08, 0A-0F. IOMC recognizes the command and writes the most significant 40 bits (word 0) of the 160 bit memory data bus into its Time-of-Day/Mailbox MDECC array pair. The IOMC then writes this data into its Mailbox register and sets the I/O Request interrupt flag on the I/O bus. Also set is the IOMC busy bit to the memory bus. This inhibits any other WRMAIL commands from being sent to the IOMC until a response to the current I/O Request interrupt is handled by the I/O bus. The IOMC also returns a 4 bit result code to the memory bus as specified by the memory bus timing. (Refer to the Main Memory Interface section for complete glossary of the error codes for this 4 bit field.)

In a two IOMC configuration, the master IOMC will always take the IIO command unless it is busy. If the master IOMC is in fact busy, the slave IOMC will respond to the command. If both the master and slave are busy, the MCACM will wait until one of the IOMCs goes not busy.

4.1.2 WTOD - WRITE TIME-OF-DAY

MCACM issues a Write command on the memory bus with the memory selection field set at 09. IOMC recognizes the command and writes the most significant 40 bits (word 0) of the 160 bit memory data bus into its Time-of-Day/Mailbox MDECC array pair. The IOMC then writes this data into its TOD counter. The IOMC is never busy to a WTOD command. The IOMC also returns a 4 bit result code to the memory bus as specified by the memory bus timing.

In a two IOMC configuration, both the master and the slave IOMC will accept the command and write the data to their respective TOD counters, thus keeping the time-of-day constant between the master and the slave IOMC.

4.1.3 RTOD - READ TIME-OF-DAY

MCACM issues a Read command on the memory bus with the memory selection field set at 09. IOMC recognizes the command and writes the current value of its TOD counter into its TOD/Mailbox MDECC array pair. In compliance with memory bus timing, the IOMC then drives the MDECC data onto the most significant 40 bits (word 0) of the memory data bus. At the same time, the 4 bit result code is again asserted by the IOMC.

In a two IOMC configuration, only the master IOMC responds to this command. The slave IOMC will only respond to the RTOD command if the master is broken or offline.

4.2 I/O COMMANDS

There are three command types that the IOMC can receive from I/O. Each of the three types has a different format for the command on the I/O bus.

4.2.1 MEMORY INTERFACE COMMANDS

Memory Interface Commands - Those commands that require the IOMC to communicate with main memory. I/O Bus Command format:

OPAAAAAA	OP =	1 Byte of Op code as defined by the I/O Bus Specification.
	AAAAAA =	6 least significant BCD digits of the beginning address of the memory access.
XAAALLLL	AAA =	3 most significant BCD digits of the beginning address of the memory access.
	LLLL =	number of bytes (in hexadecimal) of data to be transfered to or from memory.
datadata . . . datadata extra... errors..		4 bytes of data per word. For memory write, this data is sent by the initiator of the op immediately following the above transmission. For memory read, this data is returned to the I/O Bus by the IOMC. extra unused word, and 1 word of error descriptor for memory read.

There are five possible memory interface commands. They are:

RDMEM	- Read Memory	Op = D0 Hex
RDMEMT	- Read Memory Translated	Op = D8 "
RDWLCK	- Read with Lock	Op = F0 "
WRMEM	- Write Memory	Op = A0 "
WRMEMT	- Write Memory Translated	Op = A8 "

4.2.1.1 RDMEM - Read Memory (Op = D0 Hex)

The Op is decoded, the address is loaded into the BCD address register, and the length is loaded into the byte counter. The least significant digits of address are decoded and used to set up the rotation of data. Two "Read Public" memory cycles will be started by the IOMC and the IOMC will wait until the read data is returned. After 20 bytes of data have been loaded into the IOMC bandpass buffer, the IOMC will begin passing data to the I/O Bus Control (IOBC) starting with the byte addressed by the beginning address on bits 31-24 of the I/O bus. More memory requests will be made as necessary to stay ahead of the I/O bus. Once the I/O bus data transmission begins, it must flow continuously until all the data has been sent to the I/O bus. A word of error descriptor will follow as the last word of the transmission. If an error (reported by the MDC or the IOMC) occurs, the error will be logged on the IOMC, and reported as the last transmission. If there was no error, the last transmission will reflect a no-error result.

4.2.1.2 RDMEMT - Read Memory Translated (Op = D8 Hex)

This Op is handled exactly the same as a normal read operation except that the memory data will be translated from EBCDIC to ASCII on the IOMC before being sent to the I/O bus.

4.2.1.3 RDWLCK - Read with Lock (Op = F0 Hex)

The Read with Lock command is like a write to memory of one byte of data, except that the byte of data before modification at the specified address is returned to the I/O bus. The Op is decoded, the address is loaded into the BCD address register, and a length of 0001 is loaded into the byte counter from the length field of the command. The least significant digits of address are decoded and used to set up the rotation of data. A single Read-Modify-Write memory cycle is started by the IOMC. When the read data is returned to the IOMC, the one byte specified by the beginning address is saved in the RAMROT, and then this byte is modified in the MDECC. The modified data is then written back to memory and the unmodified byte saved earlier is returned to the I/O bus along with any error information.

4.2.1.4 WRMEM - Write Memory (Op = A0 Hex)

The Op is decoded, the address is loaded into the BCD address register, and the length is loaded into the byte counter. The least significant digits of address are decoded and used to set up the rotation of data. A read-Modify-Write memory cycle at the beginning address is started by the IOMC as soon as the Op is decoded. The incoming data from the I/O bus is rotated and loaded into the buffer (RAMROT). As soon as memory data is returned from the RMW, it will be loaded into the MDECC's. The data in RAMROT will then be merged with the read data in the MDECC's. ECC will be generated for the modified block of data and the first write cycle will then be executed. From then on data will be loaded into the MDECC as available from the buffer and written as full memory blocks. If the end of the data does not reach the block boundary, a last RMW cycle will be executed. An error descriptor is returned to the I/O Bus initiator after the last write is finished.

If an error occurs in the address or length field of the command, the write will be aborted and the error reported to the I/O bus after the I/O bus is finished sending write data. When the RMW cycle is requested, the IOMC will write the data back to memory unmodified. If an error occurs during data transmission (as reported by the MDC or IOMC) the write operation will be completed and then the error descriptor returned to the I/O bus.

4.2.1.5 WRMEMT - Write Memory Translated (Op = A8 Hex)

This Op is handled exactly the same as a normal write operation except that the I/O data will be translated from ASCII to EBCDIC on the IOMC before being sent to memory.

4.2.2 XM INTERFACE COMMANDS

Those commands that require the IOMC to communicate with the Execute Module.

In order to eliminate the timing window between the XM interface commands (I/O complete) and the I/O bus modules writing the DLP result descriptor into memory, a "super message" command format is needed.

I/O Bus Command format:

```
OPXXXXX#      OP = 1 Byte of Op code as defined by the I/O Bus
                   Specification.
               # = Number of first XM to poll: 0,1,2,3
00000000           Don't Care Data must have good parity
OPAAAAAA      OP = 1 Byte of OP code. A = address digits (5:6)
XAAALLLL       A = address digits (8:3). L = # of bytes in Hex
errors             error descriptor
```

The second Op is intended to be a memory write Op of 2 bytes, to write a DLP result descriptor to memory. It is not limited to this, and in fact it can be any legal IOMC I/O bus Op. Only the I/O complete Op will be executed if the second Op is given as zero or any other invalid Op.

There are three possible XM Interface commands. They are:

```
IOCNORM   - I/O Complete Normal        Op = 82 Hex
IOCREAL   - I/O Complete Real-time     Op = 81  "
IOCEXP    - I/O Complete Exception     Op = 84  "
```

These Op's are used as the first Op of the "super message". All three of these Op's are handled the same. The Op is decoded, and if it is not an illegal Op, the IOMC XM interface logic proceedes with the handling of the I/O Complete. The IOMC then ignores the next word on the bus. The Op in the third word of the message is then decoded as a regular I/O bus Op. If an illegal Op is detected in the first Op, the error descriptor is returned to the I/O bus and no further action is taken. If an illegal Op is detected on the second Op of the message, the error descriptor is returned to the I/O bus and no action on the second Op is taken, but the First Op, the I/O complete, is performed by the IOMC.

4.2.2 XM INTERFACE COMMANDS (Continued)

A description of what happens in the XM interface logic after successful receipt of the first Op will be discussed in the XM Interrupter section of this specification. The XM interrupter runs independent of the rest of the IOMC logic.

4.2.3 IOMC INTERFACE COMMANDS

RDMAIL	Read Mailbox	Op = D2 Hex
RDTOD	Read Time of Day	Op = D1 Hex

Those commands that do not require communication with any module other than the IOMC itself. I/O Bus Command format:

OPXXXXXX	OP = 1 Byte of Op code as defined by the I/O Bus
.	Specification.
.	
datadata	Data returned by IOMC
datadata	Data format is dependent on the Op
errors	error descriptor

4.2.3.1 RDTOD - Read Time-of-Day Counter (Op = D2 Hex)

Op is decoded and if not an illegal Op, the TOD counter is read in two transmissions to the I/O bus followed by the error descriptor. The format of the data returned to the I/O bus is as follows:

.
.

76543210 9:0 = 10 digit Hex count of # of microseconds since
XXXXXX98 midnight.

If an illegal Op is detected during Op decode, the Op is ignored by the IOMC, and the error descriptor is returned on the I/O bus indicating an illegal Op error. It is possible for the I/O bus to access the TOD counter at the same time that the memory bus is accessing it. Control of the two operations is independently handled by the hardware. If the memory bus intends to write to the TOD counter, the IOMC detects when the I/O bus is attempting a read of the counter, and will wait until the I/O read is complete before writing to it. Reads from both the memory and I/O busses can occur simultaneously.

4.2.3.2 RDMAIL - Read Mailbox Register (Op = D1 Hex)

Op is decoded and if not an illegal Op, the MAILBOX register is read in two transmissions to the I/O bus followed by the error descriptor. The format of the data returned to the I/O bus is as follows:

.
.

76543210 9:0 = 10 digits of Mailbox data.
XXXX98XX

A final result of the RDMAIL command is that the I/O bus will set the I/O Acknowledge bit on the bus, resulting in the IOMC resetting the I/O Request bit as well as IOMC Busy to the memory bus. This will free the IOMC to accept another IIO command from the memory bus.

If an illigal Op is detected during Op decode, the Op is ignored by the IOMC and the error descriptor is returned on the I/O bus indicating an illegal Op error.

4.3 I/O BUS COMMANDS

Following is a list of the I/O bus commands recognized by the IOMC.

OP	COMMAND & DATA RECEIVED	DATA RETURNED
RDMEM	D0aaaaaa xaaallll	data... extraxx error
RDMEMT	D8aaaaaa xaaallll	data... extraxx error
RDWLCK	F0aaaaaa xaaa0001	ddxxxxxx extraxx error
WRMEM	A0aaaaaa xaaallll data...	error
WRMEMT	A8aaaaaa xaaallll data...	error
IOCNORM	82xxxxxn xxxxxxxx WRMEM	error
IOCREAL	81xxxxxn xxxxxxxx WRMEM	error
IOCEXP	94xxxxxn xxxxxxxx WRMEM	error
RDTOD	D2xxxxxx	dddddddd xxxxxxdd error
RDMAIL	D1xxxxxx	dddddddd xxxxddxx error

a - BCD absolute memory address
llll - length of transfer in bytes
n - XM to poll first
x - don't care
error - error descriptor
d - digit of data
data... - one or more words of data
extra - necessary but unused bus data transfer

5 INTERFACES

The IOMC is a data transfer module which interfaces the main memory bus and the Execute Module to the I/O bus and its related modules; Data Transfer, I/O Control, and System Maintenance. The IOMC includes the following interfaces.

5.1 MAIN MEMORY INTERFACE

The main memory structure of the V500 processor is composed of up to 16 memory data cards which handle 40 digit block operations. The main memory structure does not rotate or concatenate data in any way at all.

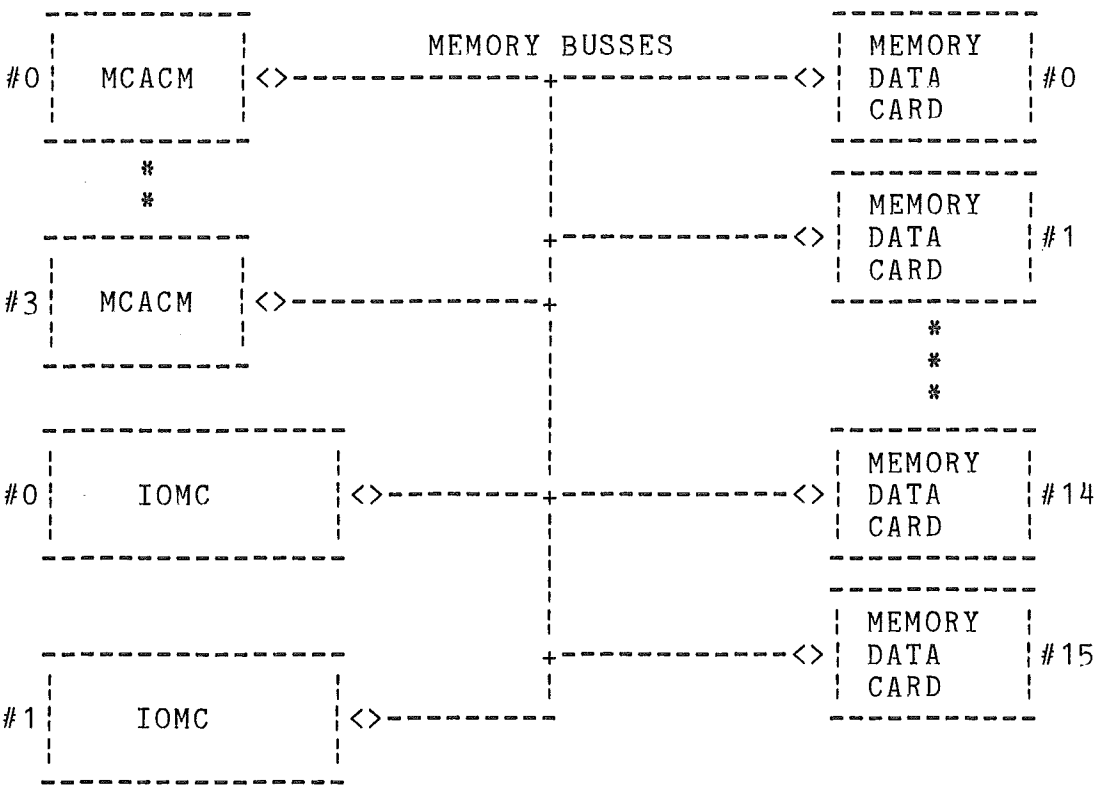


FIGURE 5-1 MAIN MEMORY STORAGE STRUCTURE INTERCONNECTIONS

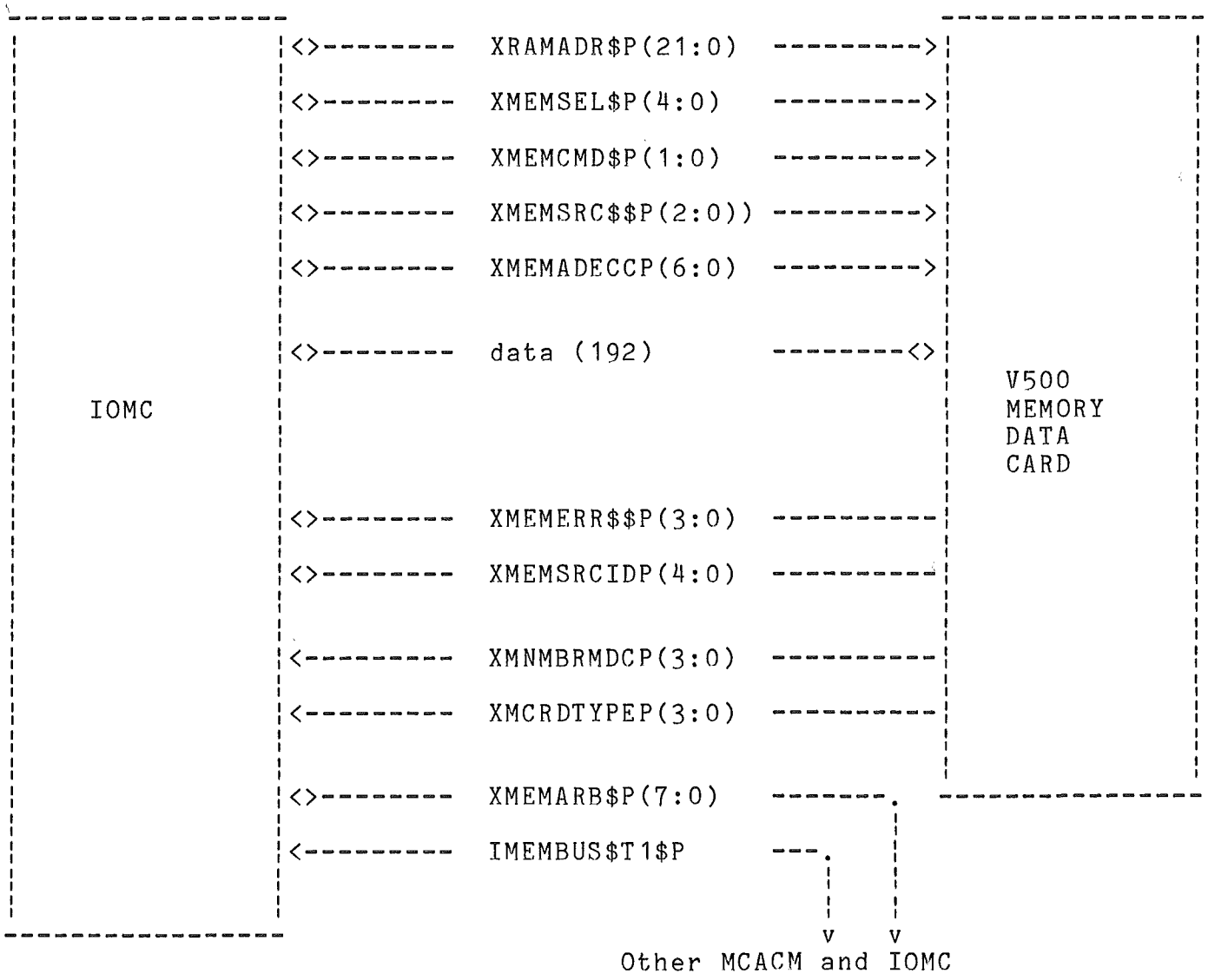


FIGURE 5-2 MEMORY DATA CARD INTERFACE

5.1 MAIN MEMORY INTERFACE (Continued)

The IOMC is capable of emulating a MCACM in order to be able to read and write to memory. This capability is made possible by a subset of the MCACMs main memory interface circuitry. This includes a BCD memory address register, a binary offset counter, the BASADD (base adder) array, the MCNTL (memory control) array, and two ECCXCVR (ECC transceiver) arrays. The BCD address is converted to binary and added to the offset. The offset starts at zero and is incremented to advance to a new memory block, on a module 40 boundary. The resulting absolute address is mapped to a specific Memory Data Card (MDC) by the MCNTL array dependent on the structure of the MDCs. The resulting mapped address is sent to the ECCXCVR array pair along with command and ID, where ECC is generated on the memory address field. The memory command can then be sent out on the memory address bus to memory after the IOMC has won bus arbitration.

The IOMC is also capable of emulating a MDC. The ECCXCV array monitors the memory backplane and its output to the IOMC includes a command and module select. The IOMC decodes the command and Mod Select to determine if the command is intended for it. If so, the IOMC responds just like a Memory Data Card. The commands that the IOMC receives from the MCACM are Initiate I/O, Read Time-of-Day, and Write time-of-Day. These commands are described in detail in Section 4 of this document.

5.1.1 MEMORY ADDRESS BUS

The main memory interface address bus contains the following signals:

XRAMADR$P(21:0)
The absolute binary module word address. This allows addressing of 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. Values of this field are 10 to 1F for addressing of memory data cards, and 00 to 0F for The IOMC. When accessing the IOMC a value of 09 is reserved for Time-of-Day reads and writes. Values of 00-08 and 0A-0F are reserved for IIO operations. The particular value of this field for IIOs will indicate the type of IIO operation. Details of the IIO type are TBS in order to facilitate the I/O Control Block (IOCB) structure of the I/O subsystem.

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

PUBLIC BLOCK READ = [10]
A full 40 digit block is read. On the IOMC this command is used to read the Time-of-Day only.

PRIVATE BLOCK READ = [11]
A full 40 digit block is read. The spies on all of the MCACMs check for local cached data and invalidate unmodified copies or relinquish modified copies. This is discussed in more detail in the MCACM Engineering Design Specification. The IOMC is incapable of this instruction. It cannot generate or comprehend this instruction and will recognize it as an error.

5.1.1 MEMORY ADDRESS BUS (Continued)

BLOCK WRITE = [01]
A full 40 digit block is written. This command is used by the IOMC to write to memory. It is used by the MCACM and the IOMC to perform an Initiate I/O instruction.

READ-MODIFY-WRITE = [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-interruptible read-modify-write operation.

XMEMID$$$P(2:0)
This field is used for distributed error detection of the memory bus arbitration logic. Figure 5-3 defines the values of this field. The V500 System can accommodate one or two IOMCs. These IOMCs 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 IOMCs.

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

FIGURE 5-3 MEMORY ADDRESS BUS REQUESTOR ID FIELD DEFINITION

5.1.1 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 IOMCs, and the other modules on the I/O bus.

	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

FIGURE 5-4 ADDRESS BUS ERROR CORRECTION CODE

5.1.1 MEMORY ADDRESS BUS (Continued)

6543 / 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

FIGURE 5-5 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 5-4) 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.

5.1.2 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, the IOMC and the MCACM 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 5-6. The syndrome to bit in error decode table is given in Figure 5-7.

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 5-6) 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.

		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			

FIGURE 5-6 DATA BUS ERROR CORRECTION CODE

76543 \ 210	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	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.

FIGURE 5-7 DATA BUS SYNDROME TO BIT-IN-ERROR DECODE TABLE

5.1.3 MEMORY ERROR BUS

Two fields are sourced on the 5th clock of each memory cycle (See Figure 5.9). These fields are sourced by memory data cards, the IOMC (when responding to RDTOD, WTOD, and WRMAIL commands; see Section 4 for details on commands) 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 7 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, and by the IOMC when responding to commands by the MCACM. 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 even parity for fault detection.

The possible conditions are:

0 - No memory cycle in progress.

1 - No errors in this memory cycle.

2 - Single bit RAM error (not corrected).
The requestor must correct and log the error.

4 - Single bit bus error (corrected).
Only applicable to write's.

7 - Multiple single bit RAM errors (not corrected).
The requestor must correct and log the error.

8 - Multiple single bit bus errors (corrected).
Only applicable to write's.

B - Multi-bit RAM error (uncorrectable).

D - Multi-bit bus error (uncorrectable).

E - Internal malfunction (fatal).

5.1.3 MEMORY ERROR BUS (Continued)

XMEMSRC$$P (4:0)
This field contains the ID number of the module which is driving the backplane. The value of the field is dependent on the card type, card slot, and cabinet ID straps.

5.1.4 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 subsystem determines the number and position of each MDC. Registers in each MDC are set up so that they can respond to requests from MCACM or IOMC. The number of cards is driven by the MDC(s) to the MCACM(s) and the IOMC(s) on the Memory Subsystem Configuration Bus. The information on this bus is described below.

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		

FIGURE 5-8 "ANY-NUMBER" MEMORY INTERLEAVING SCHEME

5.1.4 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)

5.1.5 MEMORY MODULE STATUS INTERFACE

The status of the individual memory data cards is maintained in the MCACMs and IOMCs. This status is derived from monitoring the command, module select, and memory busy fields of the memory bus.

5.1.6 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. 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 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.

5.1.7 IOMC BUSY INTERFACE

The IOMC responds to certain Memory Address Bus commands like a MDC. But the IOMC cannot respond as frequently as the memory data cards. It supplies a signal to the MCACMs 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 Write Mailbox operation. It is the same signal as the IIO request signal on the I/O bus, which will be explained later in this specification. The MCACM will wait for it to go false before sending the command.

5.1.8 MEMORY BUS DATA SOURCE INTERFACE

When the IOMC is emulating a memory data card, it must assert a signal one bus reference cycle before the actual data is to be sent on the memory bus. This signal, IOMCDATASRCP, is asserted for two ECL system clocks before Time-of-Day data, or error data is returned on the memory data bus. The reason for this is to notify the Interconnect Buffer Module (ICBM) of the origin of the data transfer so that the ICBM can set the direction of its buffers. For a complete description of the ICBM, reference the ICBM Engineering Design Spec.

5.1.9 MEMORY BUS TIMING

The main memory bus cycle is 2 processor clocks long. It is controlled on the MCACM and IOMC based on the following signal.

IMEMBUS$T1$P
Main memory bus phase level. This signal is a level, not a clock, and must be registered before being used. The following figures display the timing for the commands that the IOMC will accomodate on the memory bus:

Read Time-of-Day, Read Memory, Read-Modify-Write (first part)

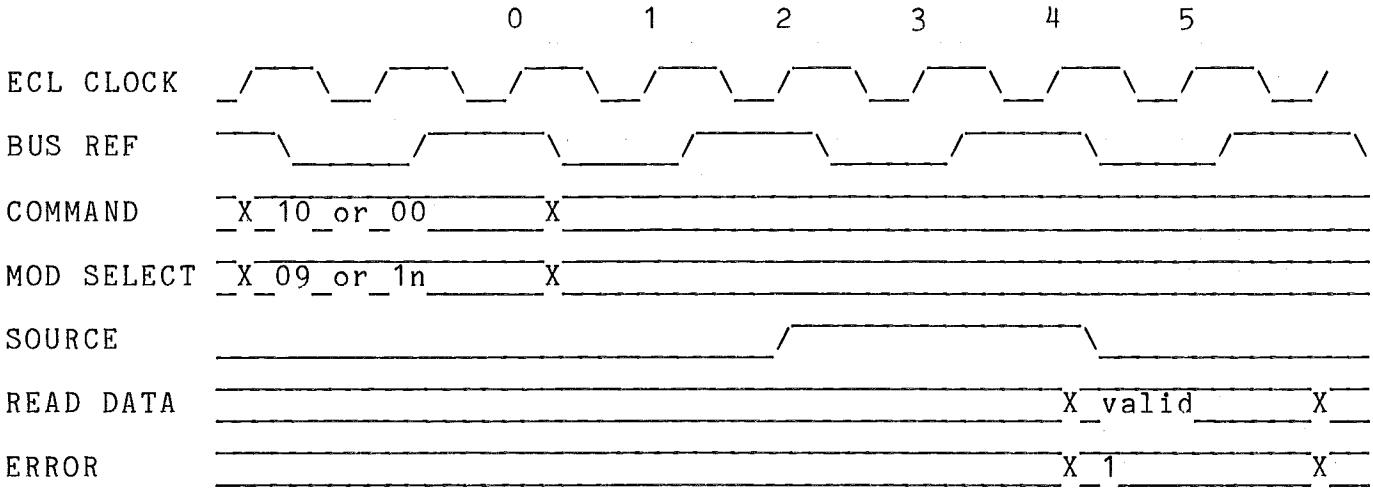


FIGURE 5-9 MEMORY BUS READ CYCLE TIMING

Write Time-of-Day, Write Hardware Register, Write Memory

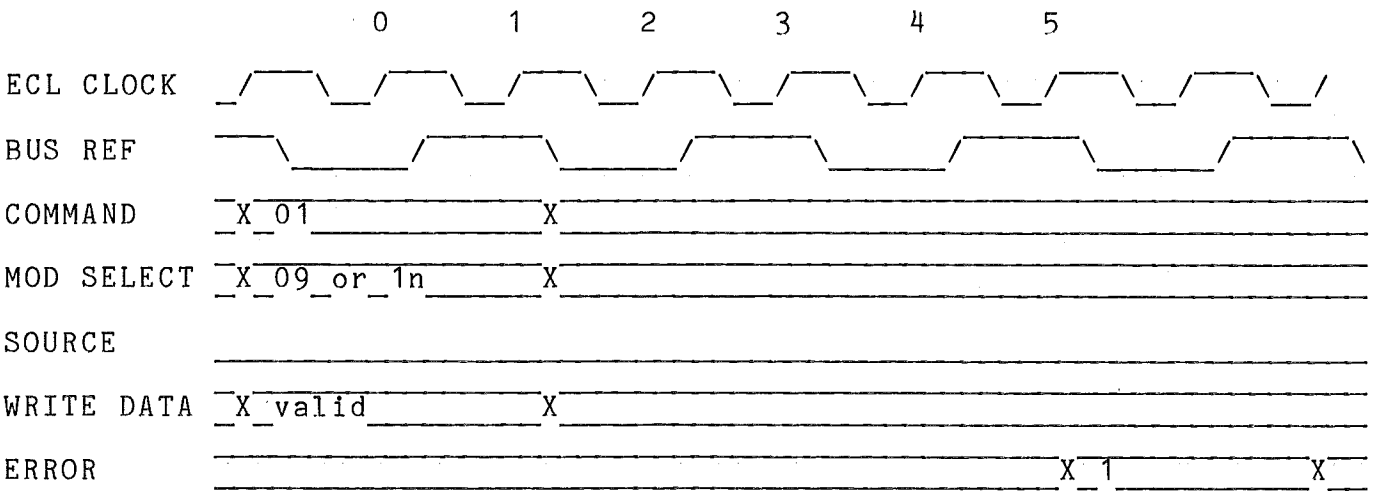


FIGURE 5-10 MEMORY BUS WRITE CYCLE TIMING

5.2 I/O BUS INTERFACE

The I/O bus is the interface between the TTL I/O Subsystem and ECL V500 processor via the IOMC. For a detailed description of the I/O bus refer to the I/O Bus and I/O bus Control Engineering Design Specification. This section will give a short summary of the function of the bus and its connectivity to the IOMC.

The V500 actually has two independent I/O busses. Each bus consists of the following signals:

TIOnDATAP(38:0) where n (0:1) is dependent on the bus
This bus is the TTL I/O bus data field. It consists of 32 bits of data and 7 bits of Error Correction Coding (ECC). To further enhance the fault detection capability of the I/O bus ECC, bits 2 and 4 of the ECC are inverted (after the generation) so that the ECC code for all-ones data is "41" hex. This makes the floating bus (all F's, including ECC) case a double bit error. This "error" is only checked when traffic is expected on the bus. Note that the ECC is identical for the I/O bus ECC and the Memory Address Bus ECC. The tables in Figures 5-3 and 5-4 of the previous section show the ECC syndrome.

TIOnBUSREF$P
I/O bus phase level. All I/O bus control is synchronized by this signal. The signal is a level qualifier that has a 50 percent duty cycle and a frequency of half that of the TTL system clock. It is generated by any card other than the IOMC that exists on the I/O bus, dependent on the maintenance configuration. This signal is a level, not a clock, and must be registered before being used.

TIOnBUSREQ$N
Active low signal used to request the bus.

TIOnBUSBUSYN
Active low signal indicating that the bus is busy. This signal is driven by the initiator of the bus transaction.

TIOnMODREQ$N
Active low signal driven by the initiator along with a module request number on the bus data lines to request a module.

5.2 I/O BUS INTERFACE (Continued)

TIOnMODPRSNN
Active low signal driven by responder indicating that the responder is present on the bus.

TIOnMODAVALN
Active low signal driven by responder indicating that the responder is available for communication.

TIOnMODBUSYN
Active low signal driven by the responder indicating that the responder is not available for communication.

TIOnTRANSFRN
Active low signal sourced by sender of data on the bus two clocks before data will exist on the bus.

TIOnERROR$$N
Active low signal driven by responder indicating that an error has occurred somewhere in the operation and indicating that an error descriptor will be sent as the last piece of data in the response.

TIOnIIOREQ$N
This active low signal is sourced by the IOMC only. It is used by the IOMC to interrupt the modules on the I/O bus and inform them that the IOMC has received an initiate I/O command from the XM. When this bit is set, the IOMC has valid data in its mailbox register to be read by an I/O bus module. When this signal is set, the IOMC is busy to MCACM and cannot receive any more IIO instructions.

TIOnIIOACK$N
Active low signal sourced by initiator on the bus to the IOMC indicating that the initiator has successfully read the IOMC mailbox register. When this signal is received, the IOMC will clear the IIOREQ signal and the busy flag to MCACM.

5.2 I/O BUS INTERFACE (Continued)

The I/O busses are used by the I/O Controllers (IOTs), Data Transfer Modules (DTMs), and the System Maintenance Controllers (SMCs) to communicate with main system memory through the IOMC. Each of the SMCs and the two possible IOMCs connect to only one of the I/O busses while the DTMs and IOTs have interfaces to both busses. For a single IOMC system, the bus not connected to an IOMC can still be used for inter-I/O module communication.

An I/O bus operation requires two communicating parties; the initiator and the responder. While the DTMs, IOTs, and SMC are capable of both initiation and response, the IOMC is only configured to act as a responder. All of the I/O operations performed by the IOMC on the I/O bus are at the request of another module on the bus.

The IOMC is currently the only module on the I/O bus not microprocessor controlled. The IOMCs understanding of messages that exist on the I/O bus is hardwired, instead of being determined by firmware.

5.2.1 I/O BUS CONTROLLER

All modules on the I/O bus contain a common TTL data path which includes a bidirectional bus driver, a bidirectional bus register, and a bidirectional ECC array known as an ECCIO array. This array checks the data coming into the IOMC from the I/O bus. It can detect and correct any single-bit errors and can detect any double-bit errors that may occur on the bus or its connecting modules. The ECC syndrome of the ECCIO is the same as that of the ECCXCVR arrays used on the memory address bus. Refer to the tables in Figures 5-4 and 5-5 for a description of this syndrome.

Also common to all I/O bus interface modules is a TTL circuit known as an I/O Bus Controller (IOBC). This circuit is used to simplify the interface between an I/O module and the I/O bus. The IOBC utilizes a relatively simple set of handshaking signals to interface the control circuitry of the module to which it is connected, to the I/O bus control signals listed in the previous section. At a modules command the IOBC will perform all of the necessary bus protocol involved in accessing the bus. In this way the module need not concern itself about the bus protocol itself. The IOBC can also respond to a request for access to a module by the I/O bus and inform the module controller of this request, again using a simple set of handshaking signals. Since the IOMC is not designed to act as an initiator, a scaled down version of the IOBC is used. This version contains only the response logic of the actual IOBC. The following is a description of the handshaking signals on the IOBC which are used by the IOMC. Again, it is a scaled down version of the complete set of IOBC handshaking signals.

HERE-IS-DATA - HID

This signal indicates that data is on the I/O bus and will be at the input of the ECCIO on the IOMC on the following clock. This signal is used by the IOMC to indicate that a bus transaction is beginning. Once this signal goes true, it will remain true until the clock before the last data of the message will exist at the input of the ECCIO. This message includes the command and any other necessary information including data for a memory write.

5.2.1 I/O BUS CONTROLLER (Continued)

CONTINUE, DONE, ERROR, DISCONNECT

Once the IOMC has received the message and decoded the command, if there is to be a response by the IOMC these signals are used to inform the IOBC that the IOMC is ready to return the response. For a memory read this response would be the memory data. The sequence of assertion of these signals tells the IOBC of the length of the response. If CONTINUE is asserted alone, the response is two or more I/O bus words. If CONTINUE and DONE are asserted at the same time the response is one word only. If DISCONNECT is asserted, there is no response, and the IOBC will disconnect the IOMC from the bus. If CONTINUE is asserted and then DONE on the following clock, the response is exactly two words. If the response is more than two words, CONTINUE is asserted, and then DONE is asserted just before the last two words of the response are to exist at the I/O side of the ECCIO. Finally, if error is asserted, the last word of the response is an error descriptor.

GIVE-ME-DATA - GMD

After the CONTINUE, DONE, and/or ERROR signals have been asserted by the IOMC, The IOBC will assert this signal two clocks before data is expected on the I/O side of the ECCIO. this signal will remain on as long as data is expected by the IOBC, as determined by the CONTINUE and DONE signals asserted by the IOMC.

The IOMC is always available to the I/O bus. When an initiator requests the IOMC, the IOMC will always respond with mod-available, unless the IOMC is offline.

5.3 EXECUTE MODULE INTERFACE

When a DTM completes an I/O request it sends an op to the IOMC which sets an interrupt in an XM which results in a call to the operating system. There are three types of IO Complete (IOC) that can be reported: normal, exception, and real-time. Each XM in the system can be masked to accept one or more of these IOC types or none at all. The IOC op sent from a DTM to the IOMC indicates IOC type and which of the XM's to poll first. The IOMC then asserts the poll line, XMATTEND, of the chosen XM and the appropriate IOC type signal. The IOMCA can have more than one IOC type queued up but will only poll for one type at a time. If the XM is able to accept the IOC it will assert the TAKEN and PRESENT signals and the IOMC will terminate the operation. If the XM is not able to accept the IOC it will assert the NOTAKEN and PRESENT signals and the IOMC will poll the next XM in the system. If the XM is OFFLINE it will not accept the IOC and will not assert any signals in response to the poll. The IOMC will continue polling one XM after another until all the IOC types it has received are taken.

The IOMC will poll each of the 4 XM's, present or not, starting with the XM specified in the command from the DTM. At each XM the IOMC will execute a poll cycle for each of the IOC types it has received that have not been accepted yet. This sequence can be upset if the IOMC receives a new IOC while in a particular state of polling an XM. On the next cycle, the IOMC may poll an XM other than the one specified by the latest command or the next one in sequence.

A list of the interface signals is as follows:

IAXMATTEND$P
IBXMATTEND$P
ICXMATTEND$P
IDXMATTEND$P

These four signals are used to poll one of the four XM's that are possible in the system. Only one of these signals will be active at a time. If none are active, no XM is being polled.

5.3 EXECUTE MODULE INTERFACE (Continued)

IONORMAL$$$P
IOREAL$$$$$P
IOERROR$$$$P

These are the three possible interrupt types that can be polled to the XM. Only one of the three will be active at a time.

IOPARITY$$P
Parity bit over itself and the interrupt type bits above. The parity should be even over the field.

EIOCTAKEN$P
Indicates that the IOC type has been accepted by the XM currently being polled.

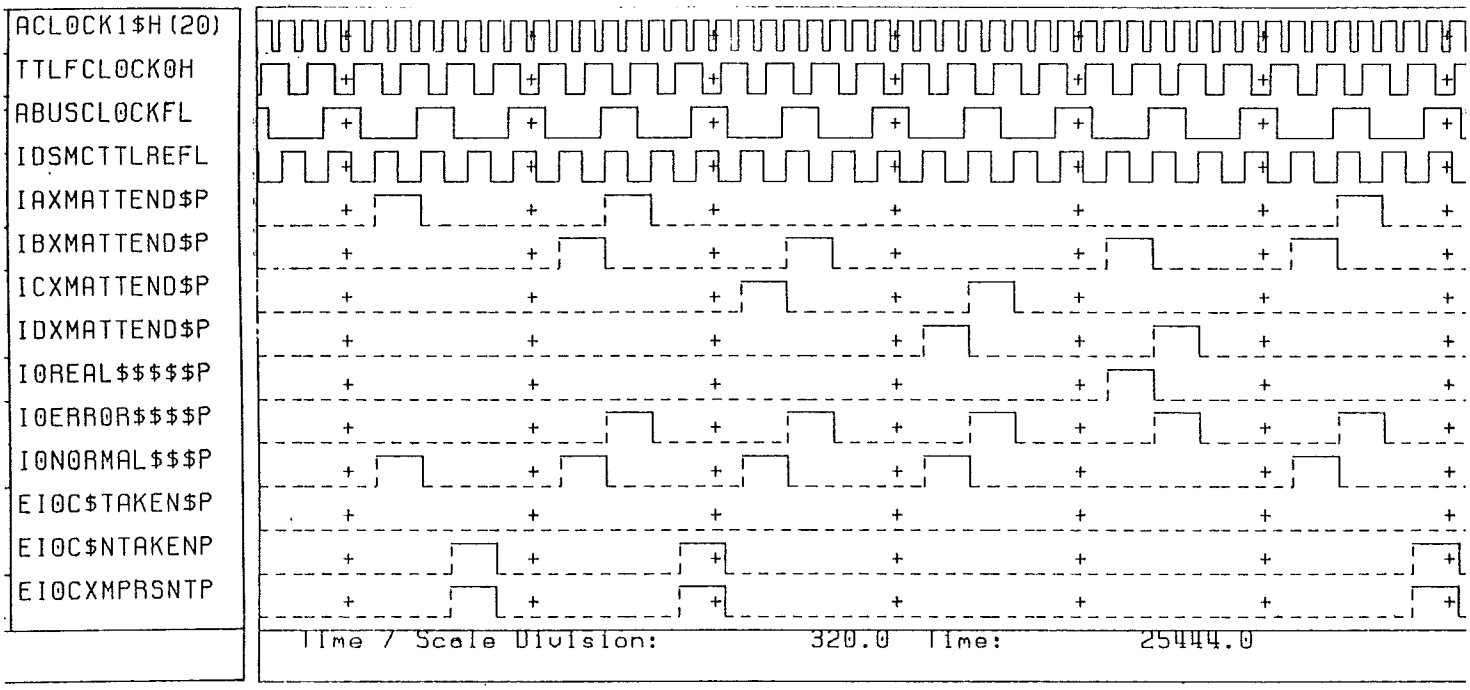
EIOCNTAKENP
Indicates that the IOC type has not been accepted by the XM currently being polled.

EXMPRESENT$P
Indicates that the XM currently being polled is present and on line.

The timing for this interface is presented in figure 5-11 below.

Clocks in Figure 5-11 are as follows:

ACLOCK1$H(20)	ECL clock.
TTLFCLOCKOH	TTL clock. 1/2 frequency of ECL clock.
ABUSCLOCKFL	Memory bus reference. 1/4 frequency of ECL clock.
IOSMCTTLREFL	TTL clock to ECL clock reference. 1/2 frequency of ECL clock.



TRANSCRIPT OF I/O COMPLETE INTERFACE TIMING EXAMP

1 SET XM-A ONLINE
SET XM-B OFFLINE
SET XM-C OFFLINE
SET XM-D OFFLINE
START A NORMAL IOC ON IOMC-A
IOMC-A POLLS XM-A WITH NORMAL IOC: NOT TAKEN

2 START AN ERROR IOC ON IOMC-B
IOMC-A POLLS XM-B WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

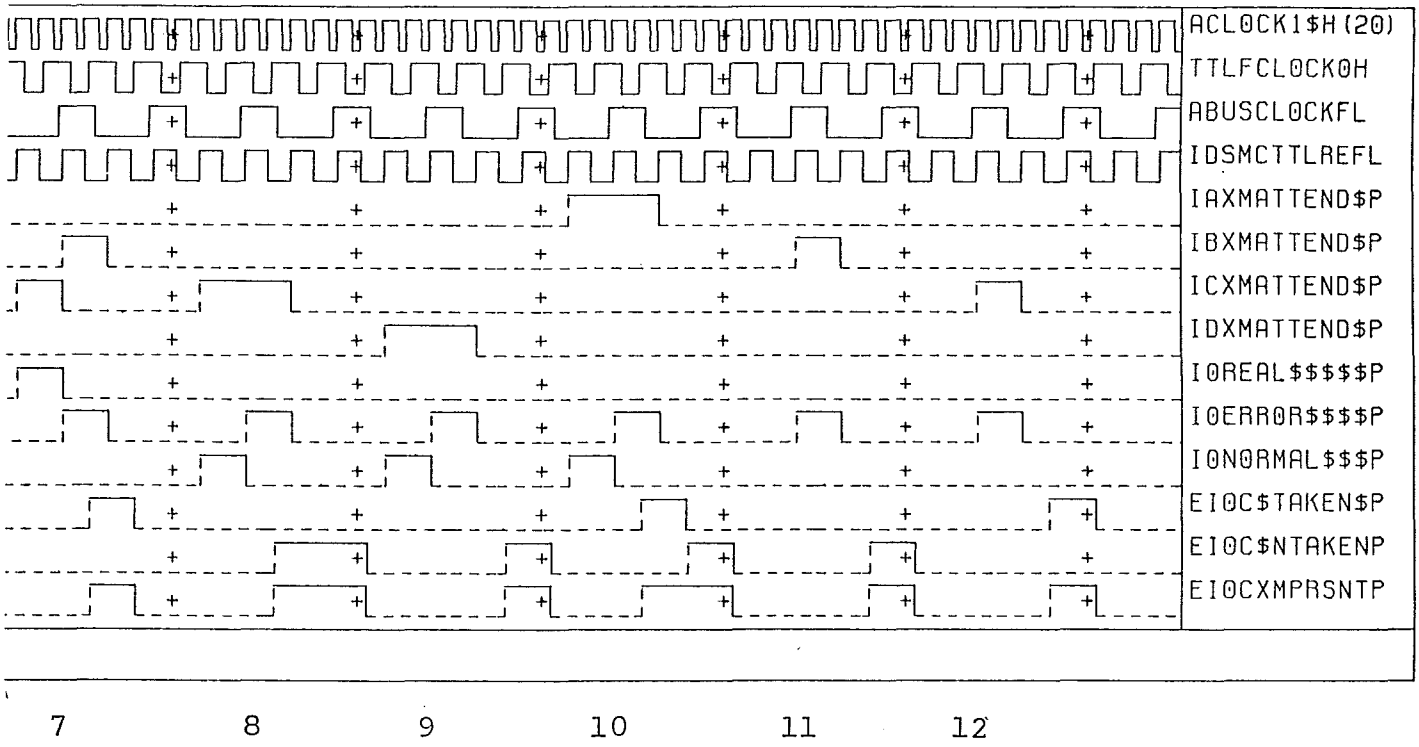
3 IOMC-A POLLS XM-C WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-B WITH ERROR IOC: OFFLINE

4 IOMC-A POLLS XM-D WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-C WITH ERROR IOC: OFFLINE

5 SET XM-C ONLINE
START A REAL IOC ON IOMC-A
IOMC-A POLLS XM-B WITH REAL IOC: OFFLINE
IOMC-B POLLS XM-D WITH ERROR IOC: OFFLINE

6 ENABLE XM-C FOR REAL IOC
IOMC-A POLLS XM-B WITH NORMAL IOC: OFFLINE
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

FIGURE 5-11 IOC TIMING EXAMPLE



7 IOMC-A POLLS XM-C WITH REAL IOC: TAKEN
IOMC-B POLLS XM-B WITH ERROR IOC: OFFLINE

8 IOMC-A POLLS XM-C WITH NORMAL IOC: NOT TAKEN
IOMC-B POLLS XM-C WITH ERROR IOC: NOT TAKEN

9 IOMC-A POLLS XM-D WITH NORMAL IOC: OFFLINE
SET XM-D ONLINE
SET XM-B ONLINE
IOMC-B POLLS XM-D WITH ERROR IOC: NOT TAKEN

10 ENABLE XM-D FOR ERROR IOC
ENABLE XM-A FOR NORMAL IOC
IOMC-A POLLS XM-A WITH NORMAL IOC: TAKEN
IOMC-B POLLS XM-A WITH ERROR IOC: NOT TAKEN

11 IOMC-B POLLS XM-B WITH ERROR IOC: NOT TAKEN

12 ENABLE XM-C FOR ERROR IOC
IOMC-B POLLS XM-C WITH ERROR IOC: TAKEN

FIGURE 5-11 IOC TIMING EXAMPLE (Continued)

5.4 INTER-CABINET BUFFER MODULE INTERFACE

The V500 can be configured as a dual cabinet system. The Inter-Cabinet Buffer Module (ICBM) is used to buffer the signals and memory between the two cabinets. The ICBM is required to know which module is driving the bus in order to enable its buffers in the correct direction. This requires a signal, IOMCDATASRCP, from each IOMC, with the driving IOMC driving its signal true and the other driving its signal false. For a complete description of the ICBM, refer to the ICBM Engineering Design Specification.

5.5 MAINTENANCE INTERFACE

The Maintenance Subsystem interfaces to the IOMC in the same way that it interfaces to the rest of the machine; via the shift chains on the IOMC. Maintenance also has the capability of interfacing to the IOMC via the I/O bus since the System Maintenance Controller (SMC) has a connection to that bus. In this way, Maintenance is capable of accessing memory through the IOMC in the same way that the I/O subsystem accesses memory. For more information on the Maintenance interface to the I/O bus, refer to the System Maintenance Controller Engineering Design Specification.

The IOMC has five separate shift chains, two on the data card and three on the address card. The data card contains only ECL logic. Its shift chains consist of an ECL data and ECL maintenance chain. The data chain includes all of the data registers including the MDECC arrays on the memory bus, the Ram/Rotator arrays (RAMROT), the time-of-day counter, the Control register, and other miscellaneous registers on the card. The maintenance chain consists of the clock maintenance array (CLKMNT) and the on card stop logic (STOP array).

The address card consists of both ECL and TTL logic. Its shift chains include TTL and ECL data chains, and an ECL maintenance chain. The TTL data chain consists of the TTL I/O bus interface control register, the I/O Bus Control (IOBC) logic, the ECCIO array, the 1 microsecond timer circuit, the XM interrupter logic, and miscellaneous data and control registers. The ECL data chain consists of the entire memory address bus interface, including the BCD address register, the address offset counter, the MCNTL5 array, the ECCXCVR array pair, and the memory interface control register, and other miscellaneous registers used in the memory interface. Also included in this chain are various registers used in the syncronization of the TTL and ECL control signals. The ECL maintenance chain consists of the CLKMNT array and the STOP array.

A complete listing of all the bits in each of the shift chains can be found in the Module Directory (MODDRY) for the IOMC.

6 IOMC EXCEPTION HANDLING

The IOMC is capable of detecting several types of errors or exceptions. These errors can be divided into two categories:

HARD ERRORS

These errors are actual hardware failures on the IOMC. If a hard error is detected, the IOMC will flag the MODBROKE bit to the maintenance array on the card and to the maintenance signal on the backplane. All clocks to the IOMC will be disabled by the IOMC maintenance logic, and the maintenance subsystem will eventually stop the entire system. Possible hard errors include parity errors in the data path, parity errors in the control path, and various duplication and comparison and parity errors throughout the module.

SOFT ERRORS

Those errors which are not fatal to the IOMC or the operation that the IOMC is currently performing. For instance, if a single-bit error occurs on the memory data bus during a memory write from the I/O bus, there is no reason to stop the operation and flag an error as the error will be corrected on the memory data card. If a double bit error occurs, there is still no reason to stop the operation. The IOMC need only report the error to the initiator of the operation and the initiator has the option to retry the operation, without the need for the machine to stop.

6.1 IOMC/DTM EXCEPTION HANDLING

If the IOMC detects a soft error of any kind on the IOMC (or the Memory Data Card via the 4-bit error code returned from memory at the end of a memory cycle), it will store this error and continue with the current operation. When the operation is complete the last information returned to the initiator will be the stored error information. If the IOMC detects an IOBUS multiple bit error in a read op or address syllable, the read is performed, the data returned, and the error reported to the requestor in the error message. A multibit error in a write op will abort the write. If the first read-modify-write cycle has started, the data read will be written back with no modification before the abort. The error message is returned to the requestor. A multibit error in an XM interface command will abort the operation before the first XM is polled. The error is reported. In case IOMC detects any single bit errors, it will report to the requesting module using the following methods and execute the command.

6.1.1 ERROR MESSAGE

The IOMC detects errors while servicing commands from I/O bus, and always returns an error descriptor.

Format of the error code :

Bit 3:0 - XMEMERR$$P(3:0) from memory bus
8:4 - XMEMSRC$$P(4:0) from memory bus
9:1 - Overflow error on IOMC
10:1 - ECCIO single bit error
11:1 - ECCIO double bit error
12:1 - MDECC 0/1 single bit error
13:1 - MDECC 2/3 single bit error
14:1 - MDECC multiple bit error

If multiple errors have occurred in the memory transfers, the most fatal error will be logged and reported.

6.1.2 UNDERFLOW/OVERFLOW IN READ MEMORY

Underflow may occur in the IOMC if transfer from buffer RAMROT has started, but memory is somehow not fast enough to provide data. DTM can detect underrun in this case by finding out that the actual number of transfers is less than expected. Overflow will not happen on read Op.

For normal reads, there will be 1 extra transfer of data and the error code will always be returned. In case of Read w/Lock command, exactly 1 data transfer and 1 error code will always be returned.

6.1.3 UNDERFLOW/OVERFLOW IN WRITE MEMORY

Buffer Overflow may occur if the Memory Bus is busy long enough. The DTM will be notified at end of a command and it is up to the DTM to retry. Underflow will not happen on a write Op except for broken hardware.

6.1.4 MULTIPLE BIT ERROR ON MEMORY WRITE ADDRESS/LENGTH

In order not to corrupt memory, the write operation will be aborted once the 1st read modify write is finished. The IOMC detects that a MBE has occurred on the address/length data coming from the I/O bus, and performs a dummy write back to memory with the same data that was received by the Read modify write command.

6.1.5 I/O BUS ABNORMAL DISCONNECT

If for any reason I/O bus disconnects from the IOMC, both the address card and data card will return to idle state, except that if the data card is in a process of executing a memory WRITE/RMW command, it will complete that before it returns to idle.

7 STOP LOGIC

7.1 OVERVIEW

Stop logic on IOMC has visibility of the following:

4 byte data passing between the two cards
the I/O bus requestor priority (initiator)
the I/O bus requested module id (responder)
loading of the memory data register, MDECC
binary absolute memory block address
memory request commands
buffer ram overflow

The first three stop conditions allow this stop logic to check for data patterns on the I/O Bus, along with which modules were requesting and requested. The IOMCA data path is set to pass I/O Bus data to IOMCD when idle so that the stop array on IOMCD is connected to the I/O bus. The stop logic has test conditions that allow it to know when the initiator or responder is valid. On the occurance of a match, the initiator or responder conditions can be stored in another register for inclusion with other stop conditions that may occur later. This allows the logic to stop on a specific initiator sending a particular (1 to 4 byte) data transfer to specific responder.

Using the byte holds (ECBYTELOADnL), the mod5 counter state (ECMOD5EQnH) and the memory block address, the logic can determine which of the 20 bytes of the memory data register is being loaded during a memory write operation. It cannot check the data being loaded at the same time.

Using the memory request commands, the logic can determine when the IOMC is requesting a memory read or write operation. The block address can be included here in order to detect what memory address the command is for.

7.2 STOP ARRAY REGISTER CONFIGURATION

7.2.1 IOMCA STOP ARRAY

Following is a description of the register signals in the STOP array on IOMCA. These signal names are taken from the IOMCA maintenance chain MODDRY Rev. B 2/2/87. In this section the initiator is known as the requestor and the responder is the destination.

ABSADRCMPH
: Data to compare with the 8 most significant digits of the binary absolute memory address.

ABSCPENH
: Block address digit compare enables. The 7 enables are arranged as (24:22)(21:18)(17:14)(13:10)(9:6)(5:2)(1:0). Bits not selected become don't cares in the stop equation.

ABSADRAENH
: Absolute address AND enable includes the compare equal of the address and command as an AND in the stop equation.

ABSADROENH
: Absolute address OR enable includes the compare equal of the address and command as an OR in the stop equation.

MEMCMDCPH
: Data to compare with the Memory Command input the the IOMCA memory control logic. 0- read with lock, 1- read, 2- write, 3- not used on IOMC.

MEMREQSTH
: Memory Request is compare with the strobe for MEMCMDH.

MEMCMDENH
: Memory command compare enable AND's the MEMCMDH and MEMREQSTH compares with the address compare, ABSADRCMPH.

DESTCMPH
: Data to compare with the I/O Bus destination data.

DESTCPENH
: Destination compare enable.

7.2.1 IOMCA STOP ARRAY (Continued)

DESTANDENH
Destination AND enable includes the compare equal of the I/O destination as an AND in the stop equation.

DESTORENH
Destination OR enable include the compare equal of the I/O destination as an OR in the stop equation.

DESTHITH
Destination Hit is loaded with the result of the destination compare when the load enable, ECMODNOLOADH, is true. The output of this register is used as input to the bit test enabled by DESTHTAENH.

DESTHTAENH.
Destination Hit AND enable includes the output of the I/O Destination Hit register as an AND in the stop equation.

DESTHTOENH
Destination hit OR enable includes the output of the I/O Destination Hit register as an OR in the stop equation.

RQSTCMPH
Data to compare with the I/O Bus requestor data.

RQSTCPENH
Requestor bit compare enables. Bits not enabled for comparison are treated as don't cares in the stop equation.

RQSTANDENH
Requestor AND enable includes the compare equal of the requestor as an AND in the stop equation.

RQSTORENH
Requestor OR enable include the compare equal of the requestor as an OR in the stop equation.

RQSTHITH
Requestor Hit is loaded with the result of the requestor compare when the load enable, ECMODNOLOADH, is true. The output of this register is used as input to the bit test enabled by RQSTHTAENH

7.2.1 IOMCA STOP ARRAY (Continued)

RQSTHTAENH
Requestor hit AND enable includes the output of the Requestor Hit register as an AND in the stop equation.

RQSTHTOENH
Requestor hit OR enable includes the output of the Requestor Hit register as an OR in the stop equation.

NORMRQSELH
Requestor Select, when high, selects the normal requestor priority for comparison with RQSTCMPH. When low, the emergency requestor priority is selected.

NRQSTCMPHH
Not Requestor Stop inverts the output of the requestor compare before it is included in the stop equation. Thus, the stop equation is true when any requestor other than the one selected by RQSTCMPH is active.

LOCBRKENH
Local Break Enable is the local stop enable. The local stop OR output causes a dead freeze.

BPSTOPENH
Backplane stop enable includes this stop logic in the system stop equation.

MASTERH
This output designates this IOMC as the master in a two IOMC system.

ONLINEL
When high, this output places this IOMC off line.

7.2.2 IOMCD STOP ARRAY

Following is a description of the register signals in the STOP array on IOMCD. These signal names are taken from the IOMCD maintenance chain MODDRY last edited on 10/08/86.

IXIORWCP
: Data to compare with the I/O Read/Write bus, IORW. This is the bi-directional data bus between the two cards.

IXIOCPENA
: I/O digit compare enables for the I/O Read/Write bus. 1 to 8 of the digits can be selected for comparison. Digits not selected will become don't cares in the stop equation.

IXIOANDEN
: I/O AND enable includes the IORW compare as an AND in the stop equation.

IXIOOREN
: I/O OR enable includes the IORW compare as an OR in the stop equation.

MD5EQCP
: Mod 5 Equals is the compare data for the 5 bit ECMOD5EQnL field. ECMOD5EQnL selects the next 32 bit block of the memory data register, MDECC, to be read or written.

MD5ECPENA
: Mod 5 compare enable.

MD5EQLAEN
: Mod 5 Equals AND enable includes the Mod 5 compare as an AND in the stop equation.

MD5EQLOEN
: Mod 5 Equals OR enable includes the Mod 5 compare as an OR in the stop equation.

BYTLOACP
: Byte Load compare data is compared with ECBYTLOADxL. ECBYTLOADxL indicates which of the 4 bytes in a 32 bit block are valid when loading the memory data register during a memory write.

7.2.2 IOMCD STOP ARRAY (Continued)

BYTLCPENA
Byte Load bit compare enables. Bits not selected become don't cares in the stop equation.

BYTLOAAEN
Byte Load AND enable includes the Byte Load compare as an AND in the stop equation.

BYTLOAOEN
Byte Load OR enable includes the Byte Load compare as an OR in the stop equation.

BYTLNOTEN
Byte Load Not Enable inverts the output of the Byte Load compare before it is included in the stop equation.

OVRFLATEN
Over Flow And Test Enable includes the RAMROT buffer over flow as an AND condition in the stop equation.

OVRFLOTEN
Over Flow Or Test Enable includes the RAMROT buffer over flow as an OR condition in the stop equation.

LOCALEN
Local Enable enables the local stop outputs. The stop OR output causes a dead freeze when high.

BPSTOEN
Backplane Stop Enable includes this stop logic in the system stop equation.

8 APPENDIX A - ARRAY FUNCTIONAL DESCRIPTIONS

8.1 BASADD

8.1.1 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.

8.1.2 INTERFACES

8.1.2.1 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 - increments the resulted value by one.
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.

8.1.2.2 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.

8.1.3 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.

8.1.3.1 BCD TO BINARY CONVERTER

The 8-digit BCD address is converted into two pseudo-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.

8.1.3.2 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.

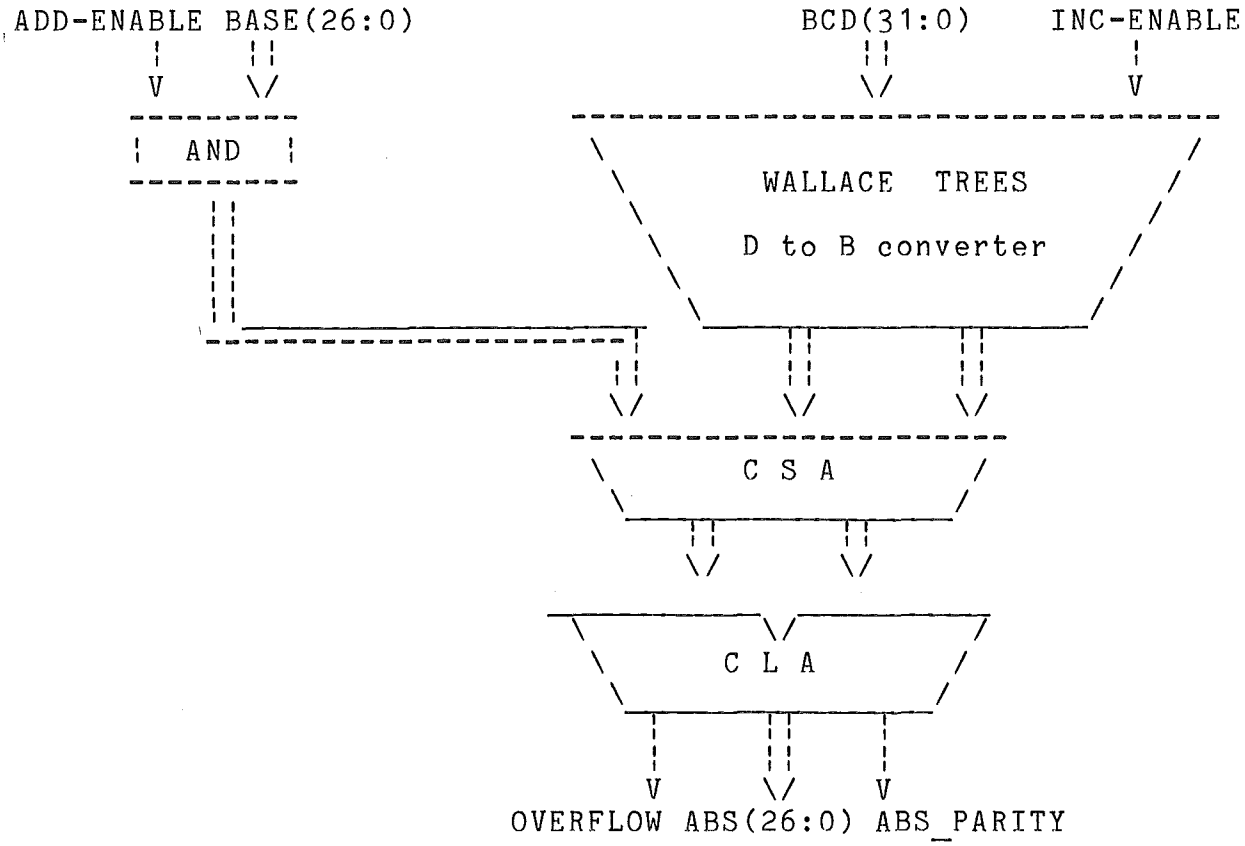


FIGURE 8-1 THE BLOCK STRUCTURE OF BASE ADDER

8.1.3.3 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.

8.1.3.4 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, or in the case of the IOMC, with discrete comparators on the card.

8.1.4 SHIFT CHAIN DEFINITION

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

8.2 ECCIO

8.2.1 FUNCTIONAL OVERVIEW

The ECCIO option is an ECC generator and SEC-DED-4ED bidirectional 32 bit bus transceiver. The primary function of the ECCIO array is to correct single bit errors and simultaneously detect double bit errors and single 4-bit byte errors in the I/O bus data path, as well as to generate new ECC on data that does not already contain it.

8.2.2 INTERFACES

8.2.2.1 INPUTS

CORRECT
This signal will cause the array to correct single bit errors if any when high.

GENECC
This signal will cause the array to generate new ECC on the data at the input to the ECC section. The ECC I/O pins must all be high for correct ECC generation.

RAMSEL
Selects between ram data and backplane data as input to ECC logic and the register. The ram data is selected when high.

HOLD
Holds all 39 bits in the F/F's.

HOLD01
Holds only bits 0-7.

HOLD02
Holds only bits 8-15.

HOLD03
Holds only bits 16-23.

HOLD04
Holds only bits 24-31

HOLDECC
Holds only ECC bits 32-38

DC_BP(3:1)
Controls direction of data in the tristate device for backplane data bits 15:0, 31:16, and 38:32 for 1,2,and 3 respectively. A 1 on these bits is used to transmit data to the backplane, and a 0 is used to turn off the tristate for receiving data from the backplane.

8.2.2.1 INPUTS (Continued)

DC_RAM(2:1)
Controls direction of data in the tristate device for ram data bits 31:0 and 38:32 for 1,and 2 respectively. A 1 on these bits will be used for transmitting data to the ram and a 0 is used to turn off the tristate for receiving data from the ram.

SHIFT
Allows dta to be shifted into and out of the array.

CLEAR
Synchronous reset of the registers in the array.

CLOCK
TTL clock input.

SHIFT_IN
Serial data input.

8.2.2.2 OUTPUTS

SBE
Single bit error. This bit will be true if a single bit is in error. The bit will be corrected if the CORRECT bit is true.

ERR
Error. This bit will be true if either a single bit error or a multiple bit error occurs.

SHIFT_OUT
Serial data output.

8.2.2.3 BIDIRECTIONALS

RAMDATA(31:0)
These 32 bits are the ram data interface. They are so called because they interface to the ram or buffer on each of the modules in which the array is used.

RAMECC(38:32)
Seven bits of ram side ECC. These bits must all be true if ECC is being generated on data bits RAMDATA(31:0).

BPDATA(31:0)
32 bits of backplane interface data.

BPECC(38:32)
7 bits of backplane side ECC. These bits must all be true if ECC is being generated on data bits BPDATA(31:0). Note: ECC should never have to be generated on the backplane side of the array for V500 applications, though it is possible.

8.2.3 DETAILED DESCRIPTION

The array is basically a bidirectional 32 bit data transceiver. It has two 32 bit tristateable interfaces; Ram side and backplane side by definition. Included with the data are 7 bits of Error Correction Code (ECC). The internal logic includes an input multiplexor, ECC generation and detection logic, a 39 bit register, and tristate I/O.

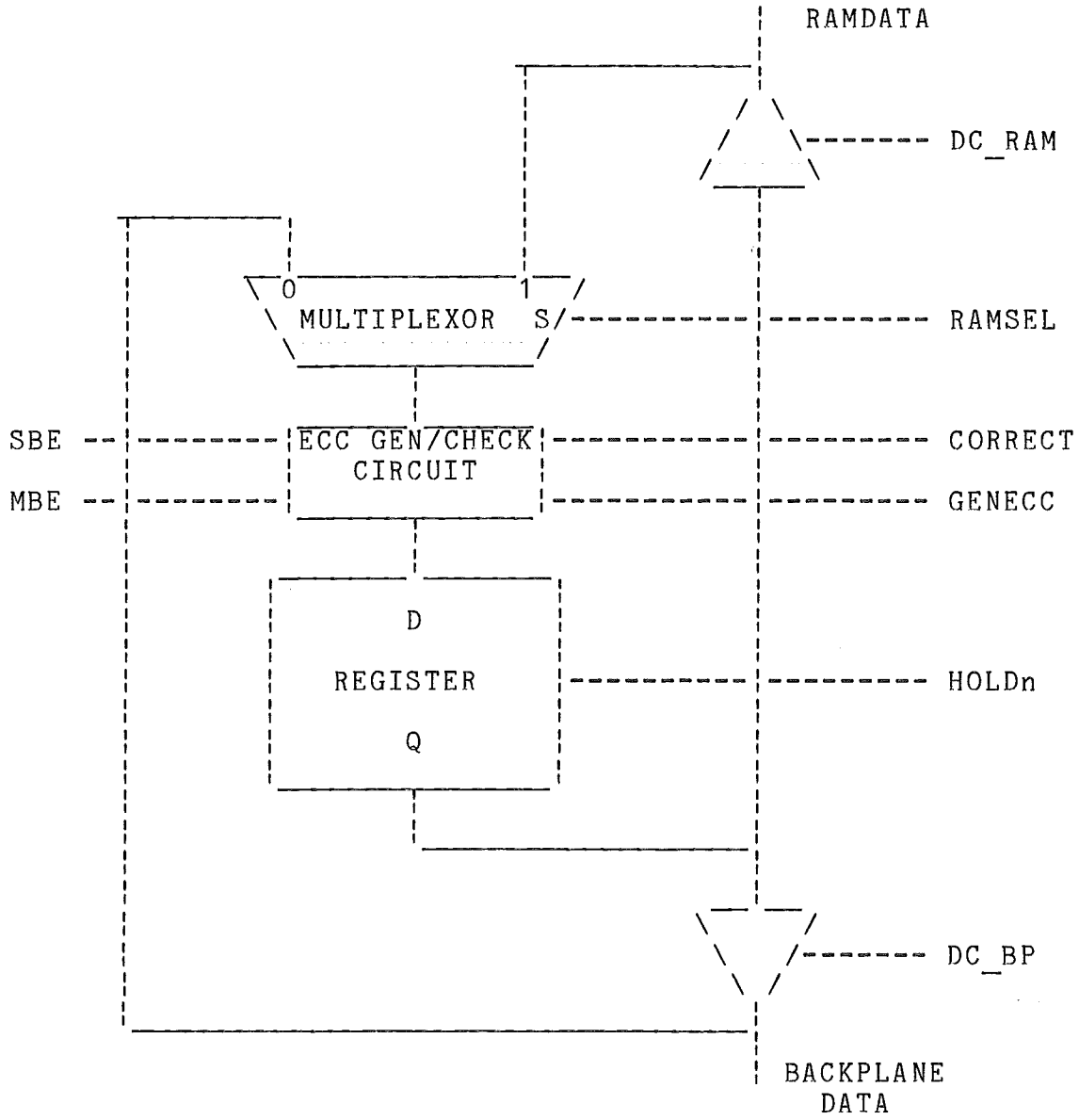


FIGURE 8-2 ECCIO BLOCK DIAGRAM

The control signals HOLD , HOLD01, HOLD02, HOLD03, HOLD04, and HOLDECC are used to hold 8 bits at a time in the F/F's. Thus, the ECCIO array can also be used to assemble an entire word, byte by byte.

The ECC is calculated over all 32 data bits. This error correct code is identical to the code which protects the Memory address bus in the ECCXCV array.

	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

FIGURE 8-3 ECCIO ERROR CORRECTION CODE

	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

FIGURE 8-4 ECCIO SYNDROME TO BIT-IN-ERROR DECODE TABLE

To further enhance the fault detection capability of the I/O bus error correction code, two of the bits are inverted. Bits 2 and 4 are inverted (after the generation, as specified in Figure 8.3.2) so that the ECC code for all-zero data is "14" hex. This makes the floating bus case a double bit error. This "error" is only checked when traffic is expected on the bus.

8.2.4 Shift Chain Definition

Data(31:0)
ECC(38:32)

8.3 ECCXCV

8.3.1 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 pseudo-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.

8.3.2 INTERFACES

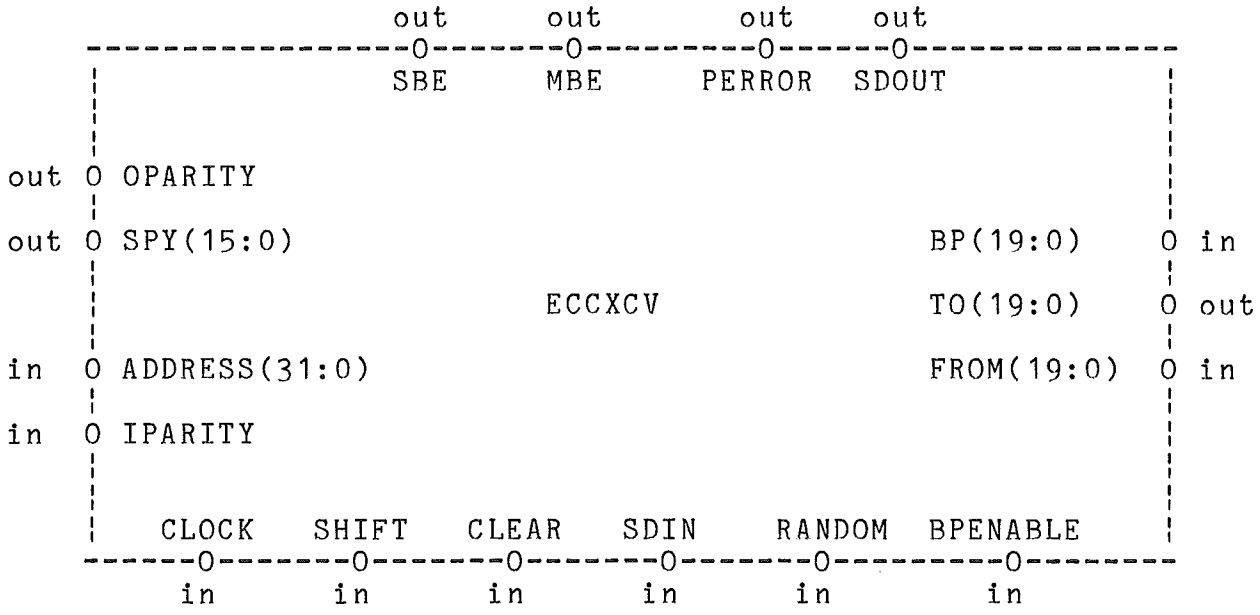


FIGURE 8-5 ECCXCV PINOUT

8.3.2.1 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.

8.3.2.2 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.

8.3.2.3 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.

8.3.3 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.

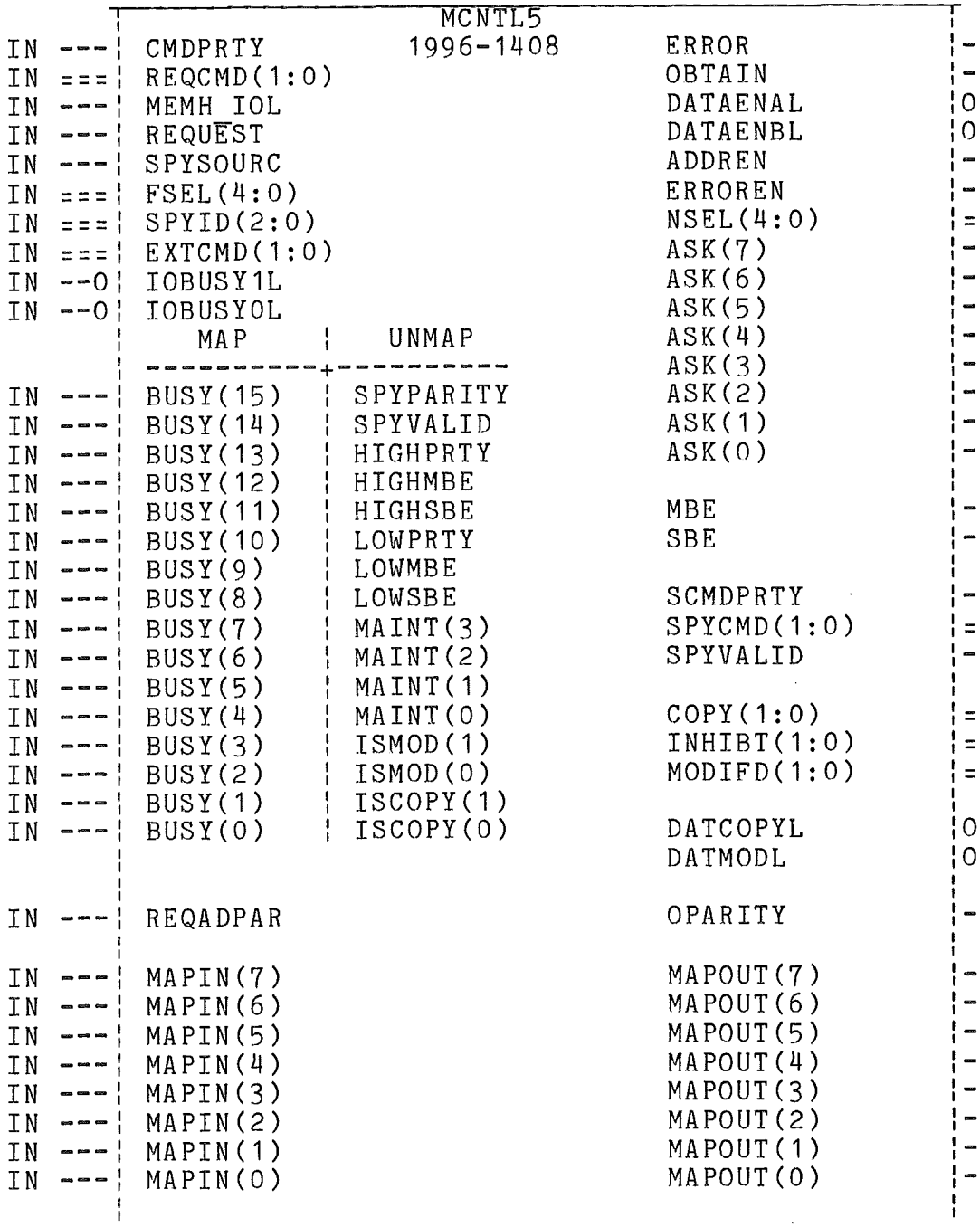
8.3.4 SHIFT CHAIN DEFINITION

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.

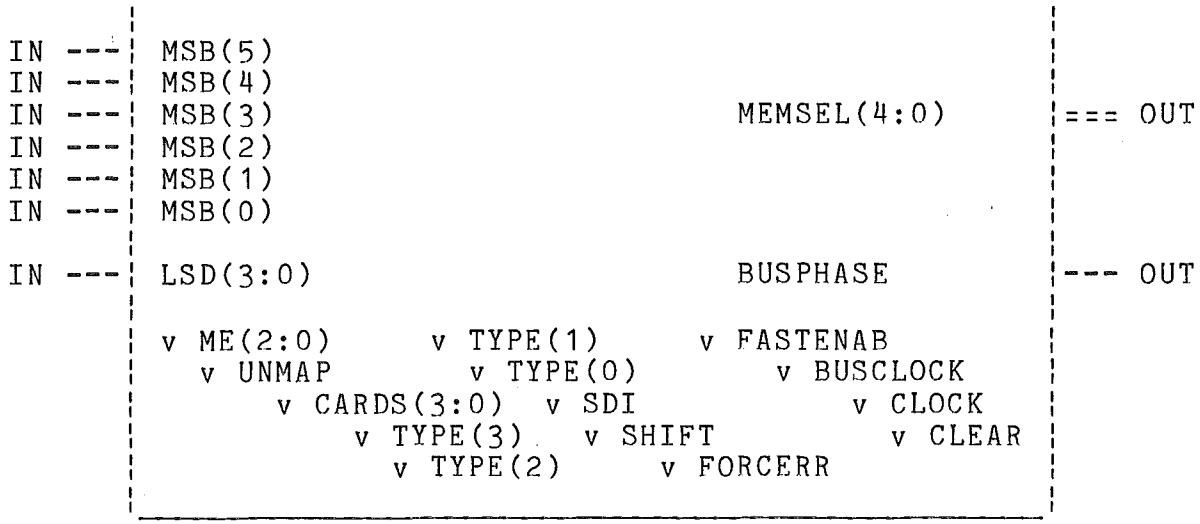
```
        RCVECC(4:6)
U       DONTCARE          - this bit is tied to XMTECC(3) only.
P       RCV(16:31)
P       XMTECC(3:6)       - XMTECC(3) does not drive the B/P.
E       ECCLPW            - see 12.6.3 for definition
R       XMT(16:31)
------------------------
L       RCVECC(3:0)
O       RCV(15:0)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

FIGURE 8-6 THE CHAIN DEFINITION FOR ECCXCV

8.4 MCNTL5



8.4 MCNTL5 (Continued)



8.4.1 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 IOMCs. It arbitrates control of the main memory bus between all requestors. The MCNTL5 array handles the interface between other caches in the V500 multiple-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. This mode is not used on the IOMC

The MCNTL5 array collects the SBE and MBE signals from the ECCXCV arrays.

8.4.2 INTERFACES

8.4.2.1 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 IOMCs. 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 IOMCs 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.

8.4.2.1 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.

8.4.2.1 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) and 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).

8.4.2.1 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 - MDCs are half-populated with 256k RAMs.
0010 - MDCs are fully-populated with 256k RAMs.
0100 - MDCs are half-populated with 1M RAMs.
1000 - MDCs are fully-populated with 1M RAMs.

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

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

8.4.2.1 INPUTS (Continued)

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.

8.4.2.2 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 MCACMs. 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 MCACMs. It signals the cache that the data being received is modified. This signal is duplicated for fault detection.

8.4.2.3 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 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.

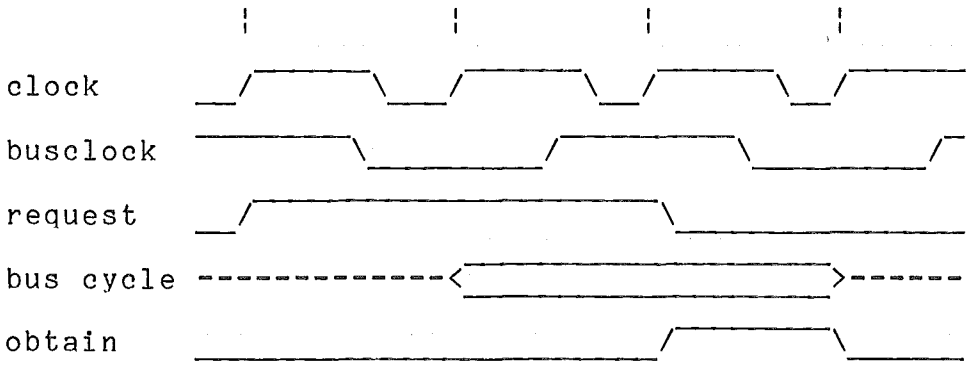
ECC TRANSCEIVER ERROR - not latched

The MCNTL5 array has detected an error in the encoding of the MBE and SBE signals from the ECCXCV arrays.

8.4.2.3 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.

8.4.2.3 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.

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.

8.4.2.3 OUTPUTS (Continued)

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 specifies memory when high, or I/O when low.

8.4.3 DETAILED DESCRIPTION

8.4.3.1 MEMORY MAPPER

The memory mapping algorithm for the V500 System is different from previous machines. This algorithm allows plugging an arbitrary number of memory data cards into the backplane.

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

When a more bizarre number of MDCs (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 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.

8.4.3.2 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. The IOMC does not use the unmap function of the array.

8.4.3.3 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 IOMCs. The state of these flip-flops is checked, when REQUEST is high. If the desired module is busy, or both IOMCs 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 IOMCs 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.

8.4.3.4 MEMORY BUS ACCESS ARBITRATION

Arbitration for the main memory bus is accomplished with the eight 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 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.

8.4.3.5 INTER-CACHE WRITE-BACK ALGORITHM INTERFACE

Each of the caches on MCACM, 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. The IOMC does not use this portion of the array as it has no cache and is indifferent to the one on MCACM.

These lines are only duplicated for fault detection.

8.4.3.6 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. 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.

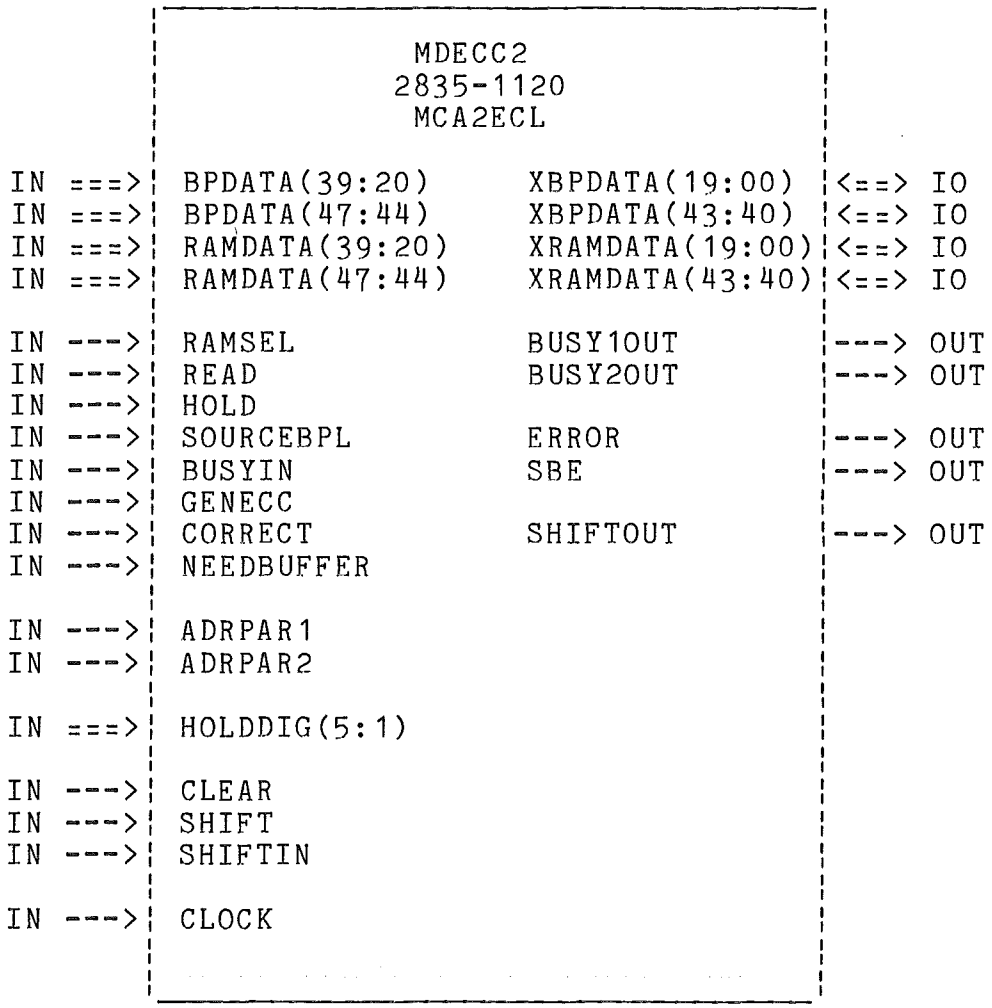
8.4.3.7 BOARD TEST FACILITIES

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

8.4.4 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

8.5 DATA ARRAY - MDECC2



8.5.1 FUNCTIONAL OVERVIEW

The MDECC2 option is an ECC generator and SEC-DED-4ED bidirectional 48 bit bus transceiver. Two MDECC2 PGA's in a mirror image configuration are required to implement the function because of the limited number of 25 ohm drivers on the MCA2ECL chip. Each PGA must see all data and check bits in order for the ECC circuitry to function properly, however, each PGA is only responsible for driving 20 data bits and 4 check bits.

8.5.2 INTERFACES

8.5.2.1 INPUTS

BPDATA(39:20)
Twenty bits of backplane data.

BPDATA(47:44)
Four check bits off the backplane.

RAMDATA(39:20)
Twenty bits of ram data.

RAMDATA(47:44)
Four check bits from ram.

RAMSEL
Selects the input to the ECC circuitry and data registers. A high chooses the RAMDATA whereas a low will select BPDATA. Remember that due to the number of output drivers half the ram data will enter on inputs RAMDATA(39:20) and RAMDATA(47:44), while the other 24 bits will invade I/O lines XRAMDATA(19:0) and XRAMDATA(43:0). Therefore, his I/O bus should be in receive mode (READ HIGH) while RAMSEL is high. See READ below.

READ
A true will allow the read data from ram memory to be received on I/O pins XRAMDATA (19:0) and XRAMDATA(43:40).

HOLD
A high on HOLD will place all data and error registers into the hold mode.

8.5.2.1 INPUTS (Continued)

SOURCEBPL - active low

A low on source backplane allows I/O pins XBPDATA(19:0) and XBPDATA(43:40) to drive data onto the backplane. This signal from the control array goes low on the fifth clock of a read cycle. It is then latched and extended another clock period in order to present data to the backplane for two clock periods.

BUSYIN

From the control array. There are two "busy" flip-flops in the MDECC2, for a total of sixteen on the board. Upon system initialization, the maintenance processor will shift a one into the flip-flop which corresponds to the MDCs logical address. BUSYIN is received by all data arrays and will gate one of sixteen "busies" onto the backplane. If NEEDBUFFER is high, BUSY1OUT and BUSY2OUT will be the noninverted buffered output of BUSYIN.

GENECC

A true on GENECC forces the data array pair to generate 8 check bits from the 40 bit data word. The data present on the syndrome input lines are ignored when GENECC is active.

CORRECT

From the control array. When CORRECT is high, single bit errors will be detected and corrected before the data is clocked into the registers. On the MDC this signal is always high when write data is captured off the backplane. In order to correct read data errors it is necessary to shift a one into the CORRECT flip-flop in the control array.

NEEDBUFFER

Tied to VTT on the MDC. The 25 ohm outputs BUSY1OUT and BUSY2OUT will, when NEEDBUFFER is high, be the noninverted buffered output of BUSYIN delayed by approximately 2.5 ns.

8.5.2.1 INPUTS (Continued)

ADRPAR1
Address parity bit 1 is the exclusive or of the MODSEL field and RAMADR bits 00, 01, 02, 03, 04, 05, 06, 07, 08, 19, and 20. When CORRECT on the address array is high, this bit is derived from the corrected address field. If address correction is not utilized ADPAR1 will be calculated with the above listed bits as they are received off the backplane. This bit is sent to all data arrays and is XOR'd with data bits 00 and 39 when both writing and reading to RAM memory. This was done to detect an address short in the RAM array. For instance, suppose data was not written to its target address because of a stuck-at-zero address line. Then when reading the stuck-at address, the ADRPAR bits (there are two) will be different (one or both) and the corresponding data bits will not be flipped to their true states. Thus, a double bit error is reported.

ADRPAR2
The XOR of RAMADR bits 09, 10, 11, 12, 13, 14, 15, 16, 17, 18 and 21. This bit is also sent to all data arrays and is XOR'd with data bits 01 and 38. See ADRPAR1 above.

HOLDDIG(5:1)
Each data array has 24 registers - 20 data and 4 check bit. HOLDDIG(1) will place a digit, data registers 00, 01, 02, 03, into the hold state. All five digits have their own individual HOLD line. The check bits, of course, have no hold control. These lines are tied to VTT on the MDC.

CLEAR
Maintenance chain reset signal originating at the clock and maintenance chip. This signal is an asynchronous clear provided the clock is high.

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

SHIFTIN
The maintenance chain serial shift input.

8.5.2.2 OUTPUTS

ERROR and SBE

ERROR and SBE work in conjunction with one another and in conjunction with the same signals of the other data array comprising the pair. The possible combinations for data array pair 1 and 2 are as follows:

ERROR1 = 0 ERROR2 = 0	SBE1 = 0 SBE2 = 0	No error.
ERROR1 = 0 ERROR2 = 1	SBE1 = 1 SBE2 = 0	Single bit error in MDECC
ERROR1 = 1 ERROR2 = 0	SBE1 = 0 SBE2 = 1	Single bit error in MDECC
ERROR1 = 1 ERROR2 = 1	SBE1 = 0 SBE2 = 0	Multi-bit error.

A single bit error in data array 1 means that the error in the 48 bit word is one of the 24 bits that gets registered in data array 1. Basically, the array pair look at the 48 bits, and if one array sees an error which is not a bit that is registered on chip, he will raise his ERROR line saying an error occurred but it is not in me. If it was a single-bit-error the other array will make SBE true. A multi-bit error will cause both error lines to go high. These error lines go to the control array for decoding and reporting.

BUSY1OUT and BUSY2OUT

There are 2 "busy" flip-flops in each data array for a total of 16 on the MDC. On system configuration, the system maintenance controller is responsible for shifting a 1 into the flip-flop, which corresponds to the cards' logical address. If a 1 was shifted into the BUSY1 flip-flop, BUSY1OUT will go true on receipt of a BUSYIN going high. Similarly with BUSY2OUT. These 25 ohm outputs may also be used as buffers. See Inputs: NEEDBUFFER. On the MDC, they are always used as "busies" as NEEDBUFFER is tied to VTT.

SHIFTOUT

The maintenance chain serial shift output.

8.5.2.3 BIDIRECTIONALS

XBPDATA(19:00)
Half of the 40 bit backplane data word. Each array handles 20 bits. Remember that each array "sees" all 48 bits but only drives half the data and half the check bits. XBPDATA(19:0) is the half of the data word that is driven onto the backplane on read commands.

XBPDATA(43:40)
Half of the 8 check bits. Driven onto the backplane on a read cycle.

XRAMDATA(39:20)
Half of the 40 bit ram data word. Driven to the ram matrix during a write cycle.

XRAMDATA(43:40)
Half of the 8 check bits. Driven to the ram matrix on write commands.

8.5.3 DETAILED DESCRIPTION

		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		
31	:=		X	X		X			
32	:=	X	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			

FIGURE 8-7 MDECC2 CHECK BIT ENCODE TABLE

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	C	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	N	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	39	D	D	M	D	M	M	D
1011	M	4D	4D	M	4D	M	M	D	4D	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
00 = SINGLE BIT IN ERROR, WHERE "nn" IS THE BAD BIT
D = MULTIPLE ERRORS (EVEN NUMBER)
M = MULTIPLE ERRORS (ODD NUMBER)

FIGURE 8-8 MDECC2 SYNDROME DECODE TO BIT-IN-ERROR TABLE

NOTE THAT BITS 40 THRU 47 ARE THE CHECK BITS.

8.5.4 SHIFT CHAIN DEFINITION

NAME	DESCRIPTION
LRAMDATA000H	LATCHED RAM DATA BUS DATA BIT 000
LRAMDATA001H	LATCHED RAM DATA BUS DATA BIT 001
LRAMDATA002H	LATCHED RAM DATA BUS DATA BIT 002
LRAMDATA003H	LATCHED RAM DATA BUS DATA BIT 003
LRAMDATA004H	LATCHED RAM DATA BUS DATA BIT 004
LRAMDATA005H	LATCHED RAM DATA BUS DATA BIT 005
LRAMDATA006H	LATCHED RAM DATA BUS DATA BIT 006
LRAMDATA007H	LATCHED RAM DATA BUS DATA BIT 007
LRAMDATA008H	LATCHED RAM DATA BUS DATA BIT 008
LRAMDATA009H	LATCHED RAM DATA BUS DATA BIT 009
LRAMDATA010H	LATCHED RAM DATA BUS DATA BIT 000
LRAMDATA011H	LATCHED RAM DATA BUS DATA BIT 011
LRAMDATA012H	LATCHED RAM DATA BUS DATA BIT 012
LRAMDATA013H	LATCHED RAM DATA BUS DATA BIT 013
LRAMDATA014H	LATCHED RAM DATA BUS DATA BIT 014
LRAMDATA015H	LATCHED RAM DATA BUS DATA BIT 015
LRAMDATA016H	LATCHED RAM DATA BUS DATA BIT 016
LRAMDATA017H	LATCHED RAM DATA BUS DATA BIT 017
LRAMDATA018H	LATCHED RAM DATA BUS DATA BIT 018
LRAMDATA019H	LATCHED RAM DATA BUS DATA BIT 019
LRAMECC000H	LATCHED RAM DATA ECC BUS DATA BIT 000
LRAMECC001H	LATCHED RAM DATA ECC BUS DATA BIT 001
LRAMECC002H	LATCHED RAM DATA ECC BUS DATA BIT 002
LRAMECC003H	LATCHED RAM DATA ECC BUS DATA BIT 003
SBE1H	SINGLE BIT ERROR DETECTED BY DATA ARRAY 1
ERROR1H	ERROR CONDITION DETECTED BY DATA ARRAY 1
SBPHOLD1H	SOURCE BACKPLANE HOLD, DATA ARRAY 1
BUSY(00)	MDC BACKPLANE BUSY
BUSY(01)	MDC BACKPLANE BUSY

8.6 RAMROT

8.6.1 FUNCTIONAL DESCRIPTION

RAMROT is a dual port 16 entry by 5 bit latch. The array contains additional inputs, outputs, and multiplexing logic that allows eight of the arrays to interface the 32 bit I/O portion of the IOMC to the 160 bit memory portion. This interface requires data rotation, concatenation, and multiplexing, all to be accomplished by the RAMROT array group. Each array also has separate read and write address counters and a subword pointer (MOD 5 counter). RAMROT also has registers for I/O Read Time-of-day and the Mailbox. For a description of the data rotation, refer to the rotation and concatenation section of the IOMC specification.

During an IOMC idle condition, the array ram and associated counters are cleared. When the I/O bus interface of the IOMC receives a command and address, the command and address are decoded and the result is sent to the data card.

The controller on the data card looks at the op and uses the decoded address information to set up the RAMROTs. Counters and registers are initialized in the RAMROTs depending on the direction of the operation (read or write memory) and the address. Once initialized, the rotation of the data into or out of the array is set. It does not change during the entire operation.

Once the array group is initialized, it is ready to be written to and/or read from. This is done via control signals from the I/O or memory interface of the array group.

8.6.2 INTERFACES

8.6.2.1 INPUT SIGNALS

MOD5INC
Increment MOD5 counter when true.

MOD5LOAD
Load MOD5 counter from MOD5ST input when true.

MOD5ST
3 bit input data for load of MOD5 counter at setup. This field is generated by address logic on the address card and is based on the starting address of an I/O-memory operation. The value of this field determines which of the 5 subwords will initially be read from or written to.

RECTRLOA
Load read address counter depending on the value of the byte address. This signal is activated when setting up for a memory read operation. For a memory read, some of the read counters will be initialized to a one depending on the binary byte address field. This controls the data concatenation. (For a memory write, the read counters from all eight slices will be initially loaded with zero's.)

WRCTRLOA
Load write address counter depending on the value of the byte address. This signal is activated when setting up for a memory write operation. For a memory write, some of the write counters will be initialized to a one to set up the correct data concatenation. (For a memory read, the write counters from all eight slices will be initially loaded with zero's.)

COUNTUP
Countup mode (normal mode) input; when countup is 1, read counter will be incremented when the ram is read. When 0, (read-back mode) the read counter will be decremented when ram is read. This is used during the last access of a memory write operation that does not end on the block boundary.

8.6.2.1 INPUT SIGNALS (Continued)

UPDNHOLD
Hold up/down counter when true (read back mode).

BBYTAD
Binary Byte Address (BBA) input. This field is generated by address logic on the address card and loaded into a BBA register in each array. It is used to determine the rotational scheme and initialize the address counters within the array. It is set at the beginning of a memory operation and does not change during the entire operation; rotation is constant during any given memory operation.

CLEARCTR
Clear scratch pad (ram) and every counter. Used to clear the ram and associated logic at the end of a memory so that it may be reinitialized at the beginning of the next operation.

MEMREASP
Memory interface read of scratch pad (ram). This signal increments the read address counter.

MEMWRISP
Memory interface write of scratch pad; write data into scratch pad and increment write address counter.

IOREADSP
I/O reads scratch pad; increment read address counter.

IOWRITSP
I/O writes scratch pad; write data into scratch pad and increment write address counter.

IORWOEN
I/O side output enable. This signal should be false whenever the I/O interface is writing the buffer, and true whenever reading the buffer, the mailbox, or the time.

SPDIRWRI
Array directional signal. I/O writes to scratch pad, and memory reads it when true, memory writes and I/O reads when false. The I/O interface is used as an input when true, so output must be disabled via IORWOEN signal.

8.6.2.1 INPUT SIGNALS (Continued)

COMPARITY
Command parity bit. Parity should be even over this bit and MOD5INC, MOD5LOAD, UPDNHOLD, COUNTUP, WRCTRLOA, REACTRLOA, MEMWRISP, MEMREASP, SPDIRWRI, CLEARCTR, and FORCERR.

MEMRW[4:0]PI
Memory bus byte parity input. The fifth bit in the scratch pad is used to store byte parity.

IORW[4:2](3:0)
I/O Bus byte 3,2 and 1 (upper or lower digit) used for I/O input rotation on a memory write operation. The fourth digit is an I/O signal.

IORWB[4:2]P
I/O Bus byte 3,2,1 parity bits. Follow IORW[4:2](3:0).

CROSI[3:1](3:0)
Ram feedback input, Bytes 3,2,1 (upper or lower digit) used for I/O output rotation on a memory read operation.

CROSI[3:1]P
Byte 3,2,1 (upper or lower digit) parity used for I/O output rotation.

HWRRW(3:0)
Hardware register input. The data in the hardware register can be read to the I/O interface.

HWRRWP
Hardware register parity input.

TOD(3:0)
Time-of-day input. This input connects to the IORW output for an I/O read of the time-of-day.

TODDP
Time of day digit parity (used to generate byte parity).

FIFTH(3:0)
Fifth register input. The "Fifth" registers are used for bits (39:32) of the Mailbox and Time-of-day counter.

8.6.2.1 INPUT SIGNALS (Continued)

FIFTHPAR
Fifth register parity.

LOADHWR
Load hardware register from HWRRW(3:0).

READHWR
Read hardware register to I/O interface.

READFIFL
Read the Fifth register to I/O interface (active low).

FORCERR
Force error signal for error logic testing. This signal will corrupt the parity of the data card command signals above.

LATCOPEN
Bypasses ram output read latch. This is used solely for chip testing.

ID(2:1)
Array digit sliced byte ID. These signals are strapped to identify which of the 4 bytes that the array corresponds to. This information is used along with the BBA to set up the data rotation.

CLOCK
System clock.

SHIFT
Shift control input.

MRESET
Maintenance clear.

SDI
Serial data input

8.6.2.2 OUTPUT SIGNALS

MEMRWOPO
Memory interface block 0 parity output.

MOD0UDE0
Modulo 5 or up/down counter = 0.

MOD1UDE1
Modulo 5 or up/down counter = 1.

MOD2UDE2
Modulo 5 or up/down counter = 2.

MOD3UDER
Modulo 5 counter = 3, up/down counter error

MOD4UDP
Modulo 5 counter = 4 or up/down counter parity.

UDGR1
Up/down counter greater than 1.

UDLE13
Up/down counter less than or equal to 13.

CROSSO (3:0)
Ram feedback output. Used in output rotation to I/O interface.

CROSSOP
Ram feedback parity output. Used in output rotation to I/O interface.

SDO
Serial data output.

ERROR
Gate array error output. Invalid parity over array command field described above.

8.6.2.3 BIDIRECTIONALS

MEMRW[4:0](3:0)
Memory interface subword 0, 1, 2, 3, and 4. This is the memory interface described above.

IORW(3:0)
I/O interface data byte 0 (upper or lower digit). This is the I/O digit which is actually used to interface to the I/O bus as a bidirectional bus. It is used for input and output rotation, depending on direction. It is the digit that the time-of-day or hardware data would be passed to the I/O bus on.

IORWP
I/O interface data parity.

8.6.3 DETAILED DESCRIPTION

8.6.3.1 RAM

A 16 location deep by 5 bit wide (4 bits data, 1 bit parity) dual port FIFO ram. The ram is written to or read from by either the I/O interface or the memory interface. It is a circular queue. Data is rotated prior to being written into the ram and concatenated inside the ram for a memory write. Data is rotated at the output for a memory read.

8.6.3.2 COUNTERS AND REGISTERS

There are four counters and a register that indicate the status of the RAMROT and dictate its operation, data rotation, and data placement on the I/O and memory interfaces.

The read and write address counters dictate which location in the ram data will be read from or written to. These counters are initialized at the beginning of an operation in order to perform correct data concatenation. The read counter is incremented each time the ram is read, unless the COUNTUP signal is false. In this case the read counter will decrement when read. The write address counter will increment each time the ram is written.

The up/down counter tells how many locations in the ram contain valid data. The counter is incremented each time the ram is written and decremented each time the ram is read. If a read and write occur simultaneously, the counter is unaffected. If the counter is incremented beyond 16 or decremented beyond 0, an over/under flow error is signaled. Several bits are available on the RAMROT output pins to monitor the status of this counter.

The modulo 5 counter is the memory subword pointer. The 160 bit memory bus is divided into 5 subwords of 32 bits each. The mod 5 counter controls which of the 5 subwords the RAMROT will read or write. This counter is initialized at the beginning of a memory operation by loading input data on the MOD5ST input pins. The counter is incremented each time the ram is read or written from the memory interface.

The Binary Byte Address register is a two bit binary address used to dictate the four possible rotational schemes in the array. It is loaded with data from the address card at the beginning of each memory operation and is not changed during an entire operation. It is cleared at the end of the operation, as are the rest of the counters.

8.6.3.3 WRITE ROTATOR

Depending upon the status of the SPDIRWRI signal, the array is set up to load from the memory or I/O side of the array. The rotation logic takes the input, whether it be from the memory or I/O side, and rotates it, providing alignment of data between the I/O and memory busses depending on a 2 bit binary byte address (BBA) field stored in the array. The rotator is formed by a large group of multiplexors at the input of the ram. They are used to select which digit from the 5 memory digits or 4 I/O digits is to be written to the ram. Selection is made based on the direction, the mod 5 counter and the binary byte address.

8.6.3.4 READ ROTATOR

When the array is configured for a memory read, only multiplexing between the five memory words is done upon input to the ram. The actual data rotation takes place on the output of the ram before the data is placed on the IORW(3:0) pins. Again, this rotator is a large mux which selects from the data in the ram, data from the three other bytes of RAMROT via feedback, the time-of-day interface, the hardware register, or the fifth register. The selection is made based on the direction of the operation, the BBA, and the COUNTUP, READHWR, and READFIFL input control pins.

8.6.3.5 MAILBOX REGISTER

A five bit register is included in each array, one digit plus parity. This register has a 5 bit input on the interface of the array and can be read to the I/O interface of the array. In the 8 slice configuration, it is used for 4 bytes of the IOMC mailbox register.

8.6.3.6 FIFTH REGISTER

Since the IOMC mailbox register is 5 bytes wide, and the eight slices of mailbox register in the RAMROT can't supply the entire width of the IOMC mailbox register, this register is used to supply the fifth byte of the mailbox. The register is one digit plus a bit for parity. Two of the eight RAMROT slices are used to supply the fifth byte of the mailbox.

The register is also used in two more of the slices as a place to store the two most significant digits of the Time-of-Day when it is read through the RAMROT interface to the I/O bus. When the time is read to the I/O bus, only 8 of the 10 digits are read to the bus at a time. The other two digits must be stored in the fifth registers. This is in case the time changes on the TOD counter between the time the first and second I/O transmissions of the time are made.

8.6.3.7 MEMORY INTERFACE

The memory interface of the array contains five output buses, each one digit wide plus parity. With eight of these arrays, the result is 40 digits of data to correspond to the 160 bit memory bus. Internal to the arrays is a set of mux's and demux's to multiplex between the 160 bit memory bus and the 32 bit ram. The array group can input from or output to any of five 32 bit subwords of the memory bus depending on the status of an internal modulo 5 counter. This mod 5 counter can be loaded and/or incremented via external control.

8.6.3.8 I/O INTERFACE

The I/O interface is 4 digits of data, corresponding to the four bytes of the I/O bus, plus parity. The array will select which of these digits will correspond to the ram input/output based on a 2 bit binary byte address (BBA) field stored in the array at array setup.

8.6.3.9 TOD INTERFACE

Each of the arrays has a digit input plus a bit of digit parity used in feedback. Parity is generated on the bytes as data is put on the I/O interface. This input is used to interface the Time-of-Day counter to the I/O bus through the array. Four bytes of data are passed directly through the array to the I/O bus. At the same time, 1 byte is loaded into two slices of RAMROT fifth register. This byte is read to the I/O bus on the following TTL system clock, as requested by control on the address card.

8.6.4 FAULT DETECTION

The data paths through the RAMROT are all protected by byte parity. The error checking logic for this byte parity is not on the array however. The RAMROT simply passes this parity for everything except parity of the TOD. The TOD Byte parity is generated on the array before it is passed to the I/O interface.

The only other physical fault detection is in a parity checking circuit which covers the control signals originating on the data card.

There is however pseudo fault detection in that all of the counters are duplicated for each slice of the RAMROT group. Should any one of them become faulty, the error will be reflected in incorrect data parity.

8.6.5 SHIFT CHAIN DEFINITION

ERROR
RCNTR(3:0)
WCNTR(3:0)
MOD5CNTR(2:0)
UDCNTR(3:0)
FIFREG(3:0)
FIFREGP
HWREG(3:0)
HWRREGP

9 APPENDIX B - BACKPLANE SIGNALS

MCACM/MDC-IOMC INTERFACE

				CARD
	<---- bus reference ------	IMEMBUS$T1$P ------>		A
	<---- address (22) -------	XRAMADR$P(21:0) --<>		A
	<>--- module sel ---------	XMEMSEL$$P(4:0) --<>		A
	<>--- command (2) --------	XMEMCMD$$P(1:0) --<>		A
MCACM	<---- requestor ID -------	XMEMID$$$P(2:0) --<>	IOMC	A
&	<>--- address ECC --------	XMEMADECCP(7:0) --<>		A
MDC	<>--- data (192) ---------	XRAMDT0$P(39:0) --<>		D
		XRAMDT1$P(39:0)		D
		XRAMDT2$P(39:0)		D
		XRAMDT3$P(39:0)		D
		XRAMEC0$P(7:0)		D
		XRAMEC1$P(7:0)		D
		XRAMEC2$P(7:0)		D
		XRAMEC3$P(7:0)		D
	<>--- error (4) ----------	XMEMERR$$P(3:0) --<>		A
	<>--- ID (5) -------------	XMEMSRC$$P(4:0) --<>		A
	<>--- ID parity ----------	XMEMSRCPRTYP -----<>		A
	----- boards (4) ---------	XNMBRMDC$P(3:0) --->		A
	<>--- full/half pop ------	XMCRDTYPEP(3:0)---<>		A
	1M/256K chips			
	<---- IOMC busy ----------	XIOMCBUSYN(1:0)-----		A
	<>--- near select --------	XNEARSEL$P(4:0) --<>		A
	<>--- far select ---------	XFARSEL$$P(4:0) --<>		A
	<>--- arbitration bus ----	XMEMARB$P(07:00) -<>		A
	>---- MDC busy -----------	XMEMBUSYP(15:00) -->		A

XM-IOMC INTERFACE

	<---- poll XM 0 ----------	IAXMATTEND$P -------		A
	<---- poll XM 1 ----------	IBXMATTEND$P -------		A
	<---- poll XM 2 ----------	ICXMATTEND$P -------		A
	<---- poll XM 3 ----------	IDXMATTEND$P -------		A
XM	<---- normal -------------	IONORMAL$$$P -------	IOMC	A
	<---- error --------------	IOERROR$$$$P -------		A
	<---- real time ----------	IOREAL$$$$$P -------		A
	<---- parity -------------	IOPARITY$$$P -------		A
	----- taken --------------	EIOC$TAKEN$P ------>		A
	----- ignore -------------	EIOC$NTAKENP ------>		A
	----- present ------------	EXMPRESENT$P ------>		A
	<---- 1 microsecond clock-	I1USECTIMERP -------		D

9 APPENDIX B - BACKPLANE SIGNALS (Continued)

MAINTENANCE-IOMC INTERFACE

----- ECL clock ----------	A$OCKECL$11N ------>			A0
----- " " ----------	A$OCKECL$11P ------>			A0
----- " " ----------	A$OCKECL$13N ---->			D0
----- " " ----------	A$OCKECL$13P ---->	IOMC		D0
----- " " ----------	A$OCKECL$12N ---->			A1
----- " " ----------	A$OCKECL$12P ---->			A1
----- " " ----------	A$OCKECL$14N ---->			D1
----- " " ----------	A$OCKECL$14P ---->			D1
----- ECL maint clear ----	ACLEAR$$F03P ---->			AD
----- ECL ram write en ---	ADANGER$F03P ---->			D
----- ECL chain enable ---	AIOENIOMC1$P ---->			DA
----- " " " ---	AIOENIOMC2$P ---->			
----- ECL module enable --	AIOMC$MODENP ---->			AD
----- ECL power up OK ----	APOWERUPOK3N ---->			AD
----- ECL shift enable ---	ASHIFTENF03P ---->			AD
----- " " " ---	ASHIFTENF04P ---->			AD
----- ECL card select ----	A3CARDEN$P(4:0) ->			AD
----- TTL Clock ----------	B$OCKTTL$12P ---->			A
----- " " ----------	B$OCKTTL$14P ---->			A
----- TTL card select ----	BCARDEN$$N(3:0)-->			A
----- TTL maint clear ----	BCLEARTTL$$N ---->			A
----- not used ----------	BDANGERTTL$N ---->			A
----- TTL chain enable ---	BIOENIOMC1$N ---->			A
----- TTL module enable --	BIOMC$MODENN ---->			A
----- TTL power up OK ----	BPOWERUPOK1N ---->			A
----- TTL shift enable ---	BSHIFTENTTLN ---->			A
----- TTL shift in -------	BSHIFTINTTLN ---->			A
<---- TTL shift out ------	TSHIFTOUTTLN -----			A
----- IOMCA cab 1 jumper -	IA1$IDJMPR$P ---->			AD
<---- IOMCA cab 1 true ---	IA1JMPRTRUEP -----			AD
----- IOMCA cab 2 jumper -	IA2$IDJMPR$P ---->			AD
<---- IOMCA cab 2 ture ---	IA2JMPRTRUEP -----			AD
----- cabinet jumper -----	RCABNETJMPRP ---->			A
<---- mod broken ---------	IOMC$MODBRKP -----			AD
<---- mod not broken -----	IOMCMODNBRKP -----			AD
----- IOMC BP en ---------	IOMCBPEN$N(3:0)-->			D
" " " ---------	" (1:0)(1:0)			A
<---- stop and -----------	JSTOP$AND$$P ----<			A
<---- stop or ------------	JSTOP$OR$$$P ----<			A
<---- shift out of IOMC --	X4SHIFTOUT$P ----<			AD

9 APPENDIX B - BACKPLANE SIGNALS (Continued)

I/O BUS-IOMC INTERFACE

	<>--- bus busy	TI01BUSBUSYN	---->		A
I/O	<>--- bus reference	TI01BUSREF$P	---->	IOMC	A
BUS	<>--- bus request	TI01BUSREQ$N	---->		A
	<>--- data (39 bits)	TI01DATAP(38:0)	-<>		A
	<>--- error	TI01ERROR$$N	----<		A
	>---- IIO acknowledge	TI01IIOACK$N	---->		A
	<---- IIO request	TI01IIOREQ$N	----<		A
	<>--- module available	TI01MODAVALN	---<>		A
	<>--- module busy	TI01MODBUSYN	---<>		A
	<>--- module present	TI01MODPRSNN	----<		A
	<>--- module request	TI01MODREQ$N	---->		A
	<>--- transfer	TI01TRANSFRN	---<>		A

ICBM-IOMC INTERFACE

	<---- master-one from each card	T.B.S	----<		A
ICBM				IOMC	
	<---- data source	IOMCDATASRCP	----<		A

IOMC0-IOMC1 INTERFACE

	----- master busy or broken or offline	IBROKEZERO$N	-----		AA

9 APPENDIX B - BACKPLANE SIGNALS

MCACM/MDC-IOMC INTERFACE

				CARD
	<---- bus reference ------	IMEMBUS$T1$P ------>		A
	<---- address (22) -------	XRAMADR$P(21:0) --<>		A
	<>--- module sel ---------	XMEMSEL$$P(4:0) --<>		A
	<>--- command (2) --------	XMEMCMD$$P(1:0) --<>		A
MCACM	<---- requestor ID -------	XMEMID$$$P(2:0) --<>	IOMC	A
&	<>--- address ECC --------	XMEMADECCP(7:0) --<>		A
MDC	<>--- data (192) ---------	XRAMDT0$P(39:0) --<>		D
		XRAMDT1$P(39:0)		D
		XRAMDT2$P(39:0)		D
		XRAMDT3$P(39:0)		D
		XRAMEC0$P(7:0)		D
		XRAMEC1$P(7:0)		D
		XRAMEC2$P(7:0)		D
		XRAMEC3$P(7:0)		D
	<>--- error (4) ----------	XMEMERR$$P(3:0) --<>		A
	<>--- ID (5) -------------	XMEMSRC$$P(4:0) --<>		A
	<>--- ID parity ----------	XMEMSRCPRTYP -----<>		A
	----- boards (4) ---------	XNMBRMDC$P(3:0) --->		A
	<>--- full/half pop ------	XMCRDTYPEP(3:0)---<>		A
	1M/256K chips			
	<---- IOMC busy ----------	XIOMCBUSYN(1:0)-----		A
	<>--- near select --------	XNEARSEL$P(4:0) --<>		A
	<>--- far select ---------	XFARSEL$$P(4:0) --<>		A
	<>--- arbitration bus ----	XMEMARB$P(07:00) -<>		A
	>---- MDC busy -----------	XMEMBUSYP(15:00) -->		A

XM-IOMC INTERFACE

	<---- poll XM 0 ----------	IAXMATTEND$P -------		A
	<---- poll XM 1 ----------	IBXMATTEND$P -------		A
	<---- poll XM 2 ----------	ICXMATTEND$P -------		A
	<---- poll XM 3 ----------	IDXMATTEND$P -------		A
XM	<---- normal -------------	IONORMAL$$$P -------	IOMC	A
	<---- error --------------	IOERROR$$$$P -------		A
	<---- real time ----------	IOREAL$$$$$P -------		A
	<---- parity -------------	IOPARITY$$$P -------		A
	----- taken --------------	EIOC$TAKEN$P ------>		A
	----- ignore -------------	EIOC$NTAKENP ------>		A
	----- present ------------	EXMPRESENT$P ------>		A
	<---- 1 microsecond clock-	I1USECTIMERP -------		D

9 APPENDIX B - BACKPLANE SIGNALS (Continued)

MAINTENANCE-IOMC INTERFACE

----- ECL clock ----------	A$OCKECL$11N ------>			A0
----- " " ----------	A$OCKECL$11P ------>			A0
----- " " ----------	A$OCKECL$13N ---->			D0
----- " " ----------	A$OCKECL$13P ---->		IOMC	D0
----- " " ----------	A$OCKECL$12N ---->			A1
----- " " ----------	A$OCKECL$12P ---->			A1
----- " " ----------	A$OCKECL$14N ---->			D1
----- " " ----------	A$OCKECL$14P ---->			D1
----- ECL maint clear ----	ACLEAR$$F03P ---->			AD
----- ECL ram write en ---	ADANGER$F03P ---->			D
----- ECL chain enable ---	AIOENIOMC1$P ---->			DA
----- " " " ---	AIOENIOMC2$P ---->			
----- ECL module enable --	AIOMC$MODENP ---->			AD
----- ECL power up OK ----	APOWERUPOK3N ---->			AD
----- ECL shift enable ---	ASHIFTENF03P ---->			AD
----- " " " ---	ASHIFTENF04P ---->			AD
----- ECL card select ----	A3CARDEN$P(4:0) ->			AD
----- TTL Clock ----------	B$OCKTTL$12P ---->			A
----- " " ----------	B$OCKTTL$14P ---->			A
----- TTL card select ----	BCARDEN$$N(3:0)-->			A
----- TTL maint clear ----	BCLEARTTL$$N ---->			A
----- not used ----------	BDANGERTTL$N ---->			A
----- TTL chain enable ---	BIOENIOMC1$N ---->			A
----- TTL module enable --	BIOMC$MODENN ---->			A
----- TTL power up OK ----	BPOWERUPOK1N ---->			A
----- TTL shift enable ---	BSHIFTENTTLN ---->			A
----- TTL shift in -------	BSHIFTINTTLN ---->			A
<---- TTL shift out ------	TSHIFTOUTTLN ----->			A
----- IOMCA cab 1 jumper -	IA1$IDJMPR$P ---->			AD
<---- IOMCA cab 1 true ---	IA1JMPRTRUEP ----->			AD
----- IOMCA cab 2 jumper -	IA2$IDJMPR$P ---->			AD
<---- IOMCA cab 2 ture ---	IA2JMPRTRUEP ----->			AD
----- cabinet jumper -----	RCABNETJMPRP ---->			A
<---- mod broken ---------	IOMC$MODBRKP ----->			AD
<---- mod not broken -----	IOMCMODNBRKP ----->			AD
----- IOMC BP en ---------	IOMCBPEN$N(3:0)-->			D
" " " ---------	" (1:0)(1:0)			A
<---- stop and -----------	JSTOP$AND$$P ----<			A
<---- stop or ------------	JSTOP$OR$$$P ----<			A
<---- shift out of IOMC --	X4SHIFTOUT$P ----<			AD

9 APPENDIX B - BACKPLANE SIGNALS (Continued)

I/O BUS-IOMC INTERFACE

	<>--- bus busy	TI01BUSBUSYN	---->		A
I/O	<>--- bus reference	TI01BUSREF$P	---->	IOMC	A
BUS	<>--- bus request	TI01BUSREQ$N	---->		A
	<>--- data (39 bits)	TI01DATAP(38:0)	-<>		A
	<>--- error	TI01ERROR$$N	----<		A
	>---- IIO acknowledge	TI01IIOACK$N	---->		A
	<---- IIO request	TI01IIOREQ$N	----<		A
	<>--- module available	TI01MODAVALN	---<>		A
	<>--- module busy	TI01MODBUSYN	---<>		A
	<>--- module present	TI01MODPRSNN	----<		A
	<>--- module request	TI01MODREQ$N	---->		A
	<>--- transfer	TI01TRANSFRN	---<>		A

ICBM-IOMC INTERFACE

	<---- master-one	T.B.S	----<		A
ICBM	from each card			IOMC	
	<---- data source	IOMCDATASRCP	----<		A

IOMC0-IOMC1 INTERFACE

	----- master busy or broken or offline	IBROKEZERO$N	-----		AA