Fifteen Dollars

Cromemco

# XMM

## Technical Reference Manual

Cromemco

# XMM

## Technical Reference Manual

August 1984

023-2010
Rev. A

CROMEMCO, Inc.
P.O. Box 7400
280 Bernardo Avenue
Mountain View, CA 94039

This manual was produced using a Cromemco System Three computer running under the Cromemco Cromix Operating System. The text was edited with the Cromemco Cromix Screen Editor. The edited text was proofread by the Cromemco SpellMaster Program and formatted by the Cromemco Word Processing System Formatter II. Camera-ready copy was printed on a Cromemco 3355B printer.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS



## LIST OF TABLES



## LIST OF APPENDICES

# Chapter 1

# INTRODUCTION

The Cromemco XMM Memory Management board provides S-100/IEEE-696 bus systems with an efficient, highly flexible means of sharing up to 16 megabytes of memory among several programs.

In conjunction with Cromemco's XPU dual processor board, the XMM translates logical addresses to physical addresses; provides extensive protection against unauthorized memory access; and supports the development of virtual memory software.

Chapter 2 of this manual describes the organization of the MC68010 processor's 16-megabyte address space into pages and segments, and how the XMM's high-speed cache memory, the Translation Lookaside Buffer, is used to translate each address. Chapter 3 discusses the MC68010's access types, the XMM's access control table, and the various approaches available for both memory protection and sharing. Memory allocation, page swapping, and the physical page table are discussed in Chapter 4, as well as the features provided to support virtual memory. Chapter 5 describes the memory management provided for the Z80 processor, and Chapter 6 explains how to examine and change the various tables and registers on the XMM board.

# Chapter 2

# MEMORY MAPPING FOR THE MC68010

The XMM was designed to operate with the Cromemco XPU dual processor (MC68010 and Z80) board. Chapters 2 through 4 focus on memory management provided for the MC68010 processor. The more limited capabilities of the Z80 are discussed in Chapter 5.

The XMM isolates concurrent processes from the hardware and from each other by translating (or mapping) logical addresses generated by a program into physical addresses chosen by the operating system. This isolation allows each process to appear to have the entire system to itself, reducing the likelihood of software errors spreading from one program to another. After each power up and reset the operating system must initialize the XMM board and enable address translation.

## MEMORY ORGANIZATION

### The MC68010 Processor

The MC68010 processor's 24 address lines allow it to address up to 16 megabytes of memory. (Although the MC68010 can address both 8-bit bytes and 16-bit words, we will refer to bytes for most of this manual.) The XMM divides the 16-megabyte memory into 4,096 pages of 4K bytes each. The upper 12 bits of each 24-bit address select one of the 4,096 pages; the lower 12 bits select one of the bytes within the page. During address translations, the XMM changes the upper 12 address bits, via a high-speed table lookup, from the logical page number selected by the process to a physical page number where the data is actually stored.

The pages are grouped into 32 segments of 128 pages each (1/2 megabyte per segment). A description of the segments accessible to each process are stored in the segment table assigned to that process. A page table associated with each segment holds the physical page numbers assigned by the operating system to each of that segment's 128 local pages. Since the same segment for each process may have a different page table, the same logical address generated by 15 different processes might point to 15 different pages in physical memory.

### The Segment Table

The XMM's on-board memory supports 16 segment tables, allowing up to 15 processes to run concurrently before reloading the tables (one table is reserved for error registers, as described in Chapter 6). Every segment table has 32 records, one for each 1/2-megabyte segment of logical address space. Each segment table record (4 bytes) has the following entries:

| | |
|---|---|
| Segment Mapped | This bit is set to 1 if the process is using the segment. Attempts to access a segment whose segment-mapped bit is 0 produce a "segment not mapped" error (refer to Chapter 6). |
| Page Table Resident | This bit is set if the page table for the segment resides in main memory (provided to support virtual memory, as discussed in Chapter 4). |
| Segment Type | This 5-bit field is assigned by the operating system for memory protection (refer to Chapter 3). |
| Page Table Pointer | This 16-bit field holds the address of the segment's page table in main memory. |
| Segment Active | This bit is set if all or part of the page table for this segment and process is already in the XMM's high-speed memory (the Translation Lookaside Buffer, or TLB). The segment active bit is associated with, rather than physically part of, every record of every segment table. |

The segment mapped, page table resident, and segment type comprise the "mode" entry of each segment table record. During address translations, a mode control line selects either the mode entry or page table pointer, depending on the operation required.

### The Page Table

All page tables (up to 32 per process) are stored in main memory, and individual records are loaded into the XMM's high-speed memory (the TLB) only as they are accessed. Every page table has 128 records, one for each of the 128 local pages in a segment. Each page table record (2 bytes) has the following entries:

| | |
|---|---|
| Physical Page Address | This 12-bit field holds the physical page number assigned by the operating system to the logical page number accessed by the current process. |
| Page Type | This 3-bit field is assigned by the operating system for memory protection (refer to Chapter 3). |
| Page Resident | This bit is set if the physical page accessed resides in main memory (provided to support virtual memory, as discussed in Chapter 4). |

### The Translation Lookaside Buffer

The Translation Lookaside Buffer (TLB) is the 8K-byte section of high-speed memory on the XMM board where address translations actually occur. With 4,096 records, one for each page in memory, the TLB can hold up to 32 page tables. A single process with access to all 32 segments could fill the entire TLB, but since the operating system switches from one process to another (changes "context"), a single process will rarely access all of its page tables in the fraction of a second allotted. Loading the TLB with individual page table records only as they are accessed allows the TLB to be shared among all active processes, and avoids the overhead of copying entire page tables into the TLB for every context change.

A "TLB valid" bit, associated with each of the TLB's 4,096 records, is set to 1 whenever its corresponding record in the TLB is loaded with a page table record from main memory.

## ADDRESS TRANSLATION

### The Active Map Number

Since the active process in a multi-processing system is constantly changing, the XMM must be told which segment table to use before translating an address. Under each Process Identification Number (PID), the operating system stores the number of the segment table (or active map number) assigned to that process. To change the active process, the operating system puts the map number (0-15) of the new process into the XMM's active user map register. The XMM then translates addresses for the process without further supervision.

There are two active map register associated with MC68010 mapping; one for general users, and one for the supervisor (operating system). If the same map number is placed in both registers, the user and supervisor share the same address space and a portion of the user's address space must be reserved for use by the operating system.

Any routine that manipulates its own active map register (while mapping is enabled) should be stored in the same logical address of the map it is switching from and the one it is switching to.

**Caution:** The MC68010 processor may prefetch one or two additional instructions before it outputs the new map number to the XMM.

### Translations with TLB Loaded

Once the active map register has been updated for the current process, the address translation may be as simple as depicted in Figure 2-1. Logical address bits A23 through A12 are sent directly to the XMM via the overhead cable from the XPU board. The segment number (0-31) on address bits A23 through A19 selects a segment from the segment table chosen by the active map number. If the segment is mapped, and the TLB is already loaded with the appropriate record of the page table (both the segment active and TLB valid bits are set), the physical page number is taken directly from the TLB record selected by the logical page number on address bits A23 through A12. The 12-bit physical page number is output from the TLB to the S-100 bus and, together with the byte number on bits A11 through A0, constitutes the complete physical address where the data is stored. The byte number is not translated, but goes directly from the XPU to the S-100 bus.

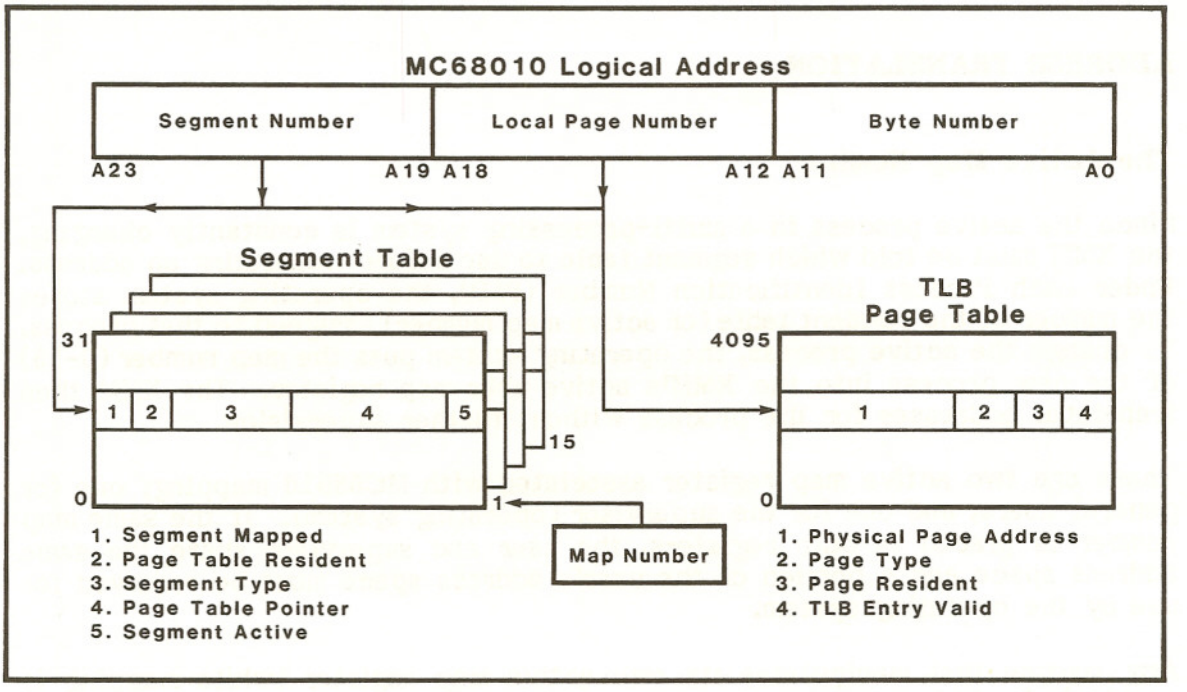


**Figure 2-1: ADDRESS TRANSLATION FOR THE MC68010**

### Translations Requiring TLB Loading

**Clearing the TLB** - If the TLB is not loaded with at least part of the correct page table for the given process and segment, then the segment active bit will be 0, and the appropriate page table record must be fetched from main memory. First, the XMM initializes the accessed segment of the TLB:

1. The TLB valid bits for all 128 pages in the accessed segment are cleared to 0.

2. The segment active bit for the accessed segment is cleared to 0 in all 16 segment tables.

3. The segment active bit for the accessed segment is set to 1 in the segment table of the current process.

Clearing the TLB valid bits invalidates any page table records that another process might have left in the TLB. Clearing the segment active bits insures that the TLB valid bits will again be flushed before another process can access this segment. (Also, whenever the operating system changes a segment's page table pointer, the segment active bit is cleared to insure that the TLB will be flushed and reloaded from the new page table.)

**Loading the TLB** - With the TLB initialized, the XMM loads the selected page table record into the TLB (see Figure 2-2):

4. The XMM takes control of the S-100 bus from the XPU.

5. The segment table mode control line selects the page table pointer from the record of the segment table determined by bits A23 through A19 of the logical address. The 16-bit page table pointer is output to S-100 bus address lines A23 through A8.

6. The local page number (0-127), selected by logical address bits A18 through A12, is output to address lines A7 through A1 (A0 is always 0). The combined page table pointer and local page number forms the physical address of the desired page table record in memory.

7. A word (16-bit) read cycle is performed, and the selected record is loaded from main memory into the record of the TLB selected by bits A23 through A12 of the logical address.

8. If an error occurs during the read cycle, or the memory addressed is not capable of 16-bit operation, then a TLB error is recorded and the XMM reports a bus error to the XPU.

9. The TLB valid bit is set for the selected record of the TLB.

10. The XPU takes back control of the S-100 bus.

With the segment active and TLB valid bits now set, the address translation can proceed as if the TLB had already been loaded. If this process now addresses a different page in the same segment, the segment active bit will be 1, but the TLB valid bit will be 0. Hence, the first three steps will be skipped, and TLB loading will commence at step 4. The segment active bit will remain set until another process accesses the same segment. The steps performed during each address translation are shown in Figure 2-3.

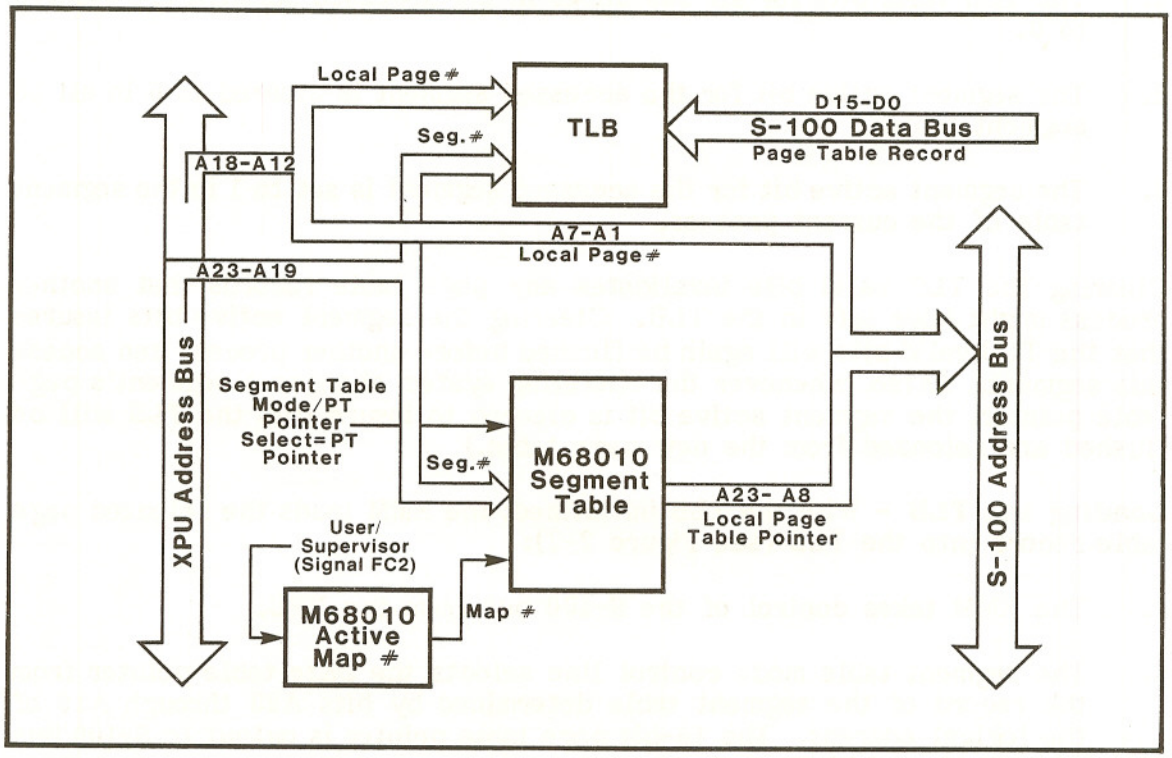


**Figure 2-2: LOADING THE TLB**

The only constraint imposed on address translations is that, in supervisor mode, logical page "0" (when address bits A23 through A12 are all 0's) must translate to a physical page containing the interrupt vector tables of the MC68010 processor. Figure 2-4 depicts the XMM data paths used to check and translate each memory and I/O access by the MC68010 processor. Access control and the physical page table will be discussed in the next two chapters.

## I/O AND MEMORY DEVICES

I/O references are made by mapping into the top 64K bytes of the physical address space (which corresponds to the unmapped I/O space of the XPU). When an address translates into this range, the XMM notifies the XPU to run an I/O cycle rather than a memory cycle. No memory devices shculd be assigned to the top 64K of memory.

Memory devices designed for 16-bit addresses should be assigned to the bottom 64K bytes of memory. Up to eight 16-bit devices can be used as long as only one responds to each memory read request. If one 16-bit device is bank-selectable, they must all be bank-selectable. They must all be disabled when an extended-address signal is generated by the XPU.

Memories and peripherals responding to the full 24-bit address should be separated so that memories are near the bottom of memory and peripherals are near the top. Peripherals responding to 24-bit addresses must respond to S-100 memory cycles, rather than S-100 I/O cycles.

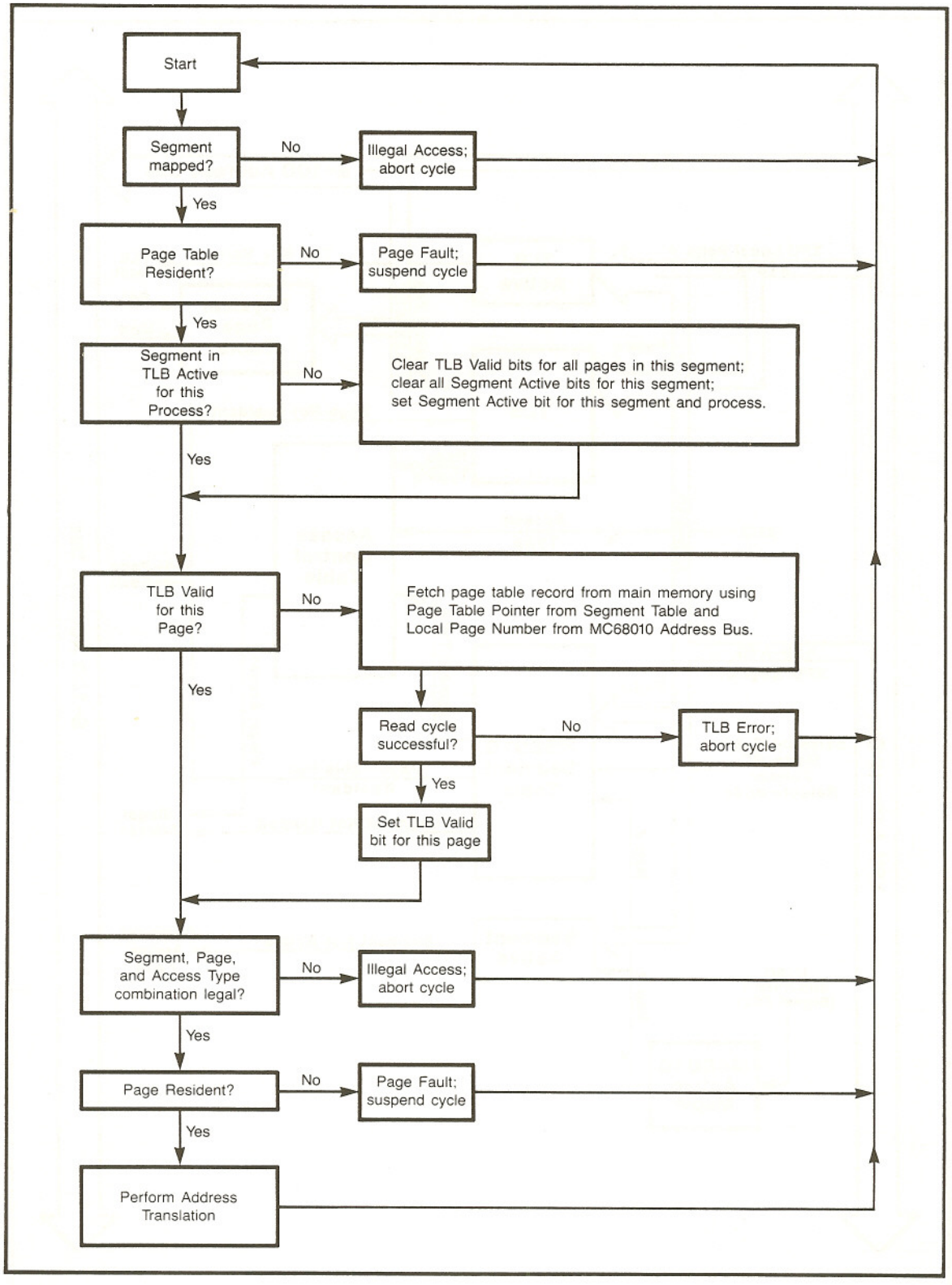


**Figure 2-3: FLOW CHART OF ADDRESS TRANSLATION**

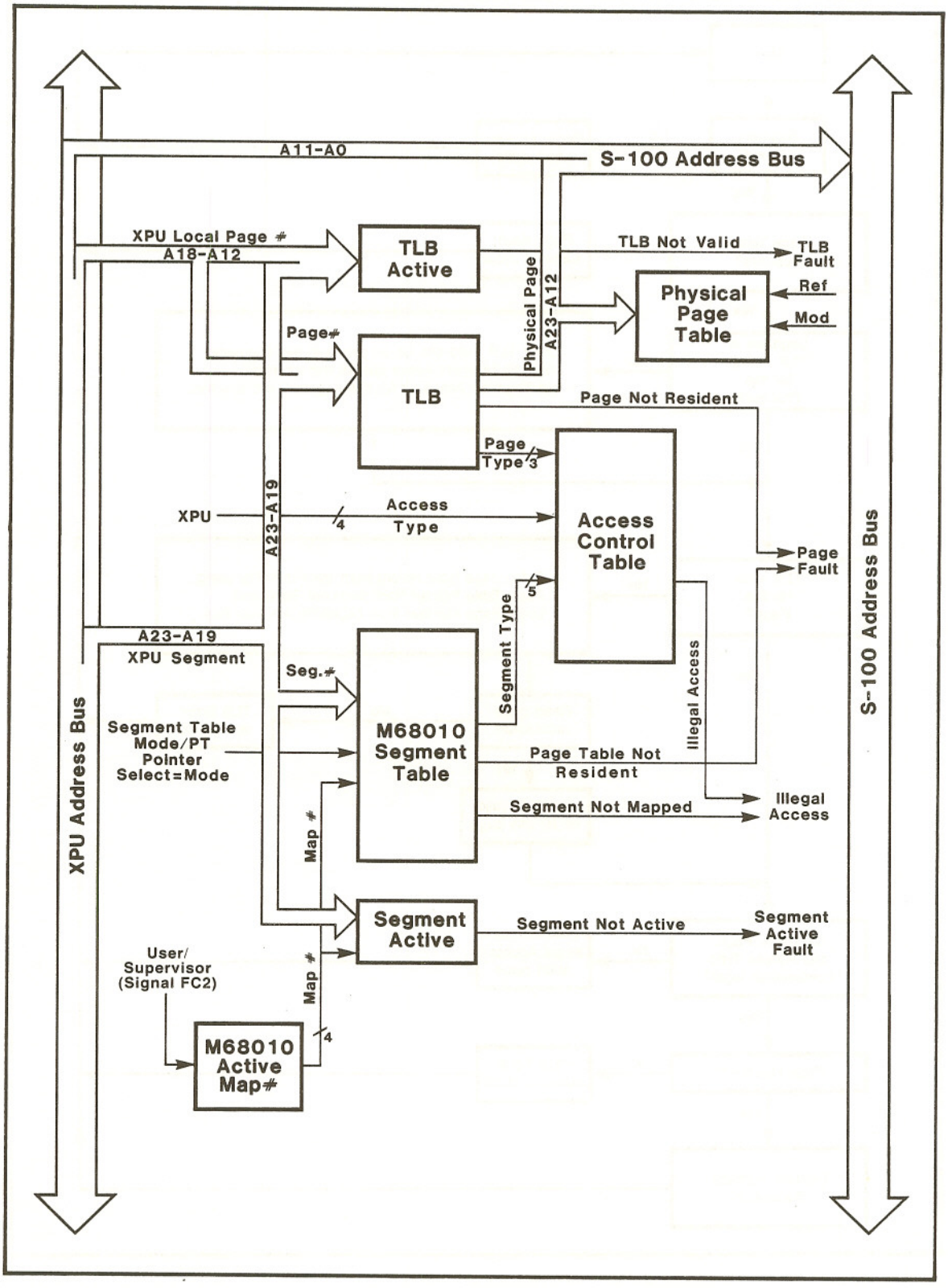


**Figure 2-4: MEMORY MANAGEMENT FOR THE MC68010**

# Chapter 3

# MEMORY PROTECTION AND SHARING

The XMM translates an address only after it verifies (based on the criteria supplied by the operating system) that the memory access is legal. This chapter describes the two methods of memory protection provided for the MC68010 processor: exclusive memory assignment and memory sharing with access control.

## EXCLUSIVE MEMORY ASSIGNMENT

The simplest way to protect all the processes from each other is to assign an exclusive set of pages to each process. If the same segment for each process has a different page table, no two logical addresses need ever translate into the same page in physical memory. The drawback to this approach is that it wastes memory. A separate copy of commonly used software, such as a compiler, would have to be kept in the memory space of each process.

## ACCESS CONTROL

Even when all the processes are protected from each other in separate areas of memory, access control is useful for protecting each process from itself. Some of the most difficult software errors to correct occur when a program executes data, reads code as data, or writes over an area of critical constants. The XMM can screen out faulty accesses by comparing the access type defined by the MC68010 processor with the page and segment types defined by the operating system.

### MC68010 Access Types

For each memory reference, four control lines from the MC68010 define the three variables of access type: user or supervisor, code or data, read or write. Table 3-1 lists the type of access for each combination of the control signals. As shown, only six of the access types are used--the rest are either not used or undefined. The XMM board is disabled during the "interrupt acknowledge read" mode (used to obtain interrupt vectors from various devices). The supervisor mode of the MC68010 has access to all system resources and is usually limited to the operating system. The operating system determines which resources to provide for the processes in user mode.

**Table 3-1: MC68010 ACCESS TYPES**

| Access Type | Mode FC2 | Mode FC1 | Bits FC0 | R/-W | Mode Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Undefined write |
| 1 | 0 | 0 | 0 | 1 | Undefined read |
| 2 | 0 | 0 | 1 | 0 | User data write |
| 3 | 0 | 0 | 1 | 1 | User data read |
| 4 | 0 | 1 | 0 | 0 | User code write (not used) |
| 5 | 0 | 1 | 0 | 1 | User code read |
| 6 | 0 | 1 | 1 | 0 | Undefined write |
| 7 | 0 | 1 | 1 | 1 | Undefined read |
| 8 | 1 | 0 | 0 | 0 | Undefined write |
| 9 | 1 | 0 | 0 | 1 | Undefined read |
| A | 1 | 0 | 1 | 0 | Supervisor data write |
| B | 1 | 0 | 1 | 1 | Supervisor data read |
| C | 1 | 1 | 0 | 0 | Supervisor code write (not used) |
| D | 1 | 1 | 0 | 1 | Supervisor code read |
| E | 1 | 1 | 1 | 0 | Interrupt acknowledge write (not used) |
| F | 1 | 1 | 1 | 1 | Interrupt acknowledge read (XMM disabled) |

## Segment and Page Types

As mentioned in Chapter 2, segment and page types are assigned by the operating system. The 3-bit page type has eight possible values. Page type "0" means that the page is not available to the process (similar to segment not mapped) and cannot be accessed. The remaining seven page types, as they might be assigned, are shown below (with their legal access types in parentheses).

1. User read only (3, 5, A, B)
2. User execute only (5, A, B)
3. User data only (2, 3, A, B)
4. User full access (2, 3, 5, A, B)
5. Supervisor read only (B, D)
6. Supervisor data only (A, B)
7. Supervisor full access (A, B, D)

The operating system assigns page types based on the kind of information stored in each page. The 5-bit segment type, with 32 possible values, defines the meaning of the page types for each segment. This permits two processes which share a segment of memory to have different levels of access to the shared pages.

### The Access Control Table

The access control table on the XMM board has 256 records, one for each combination of segment type and page type. Each bit of a 16-bit record corresponds to one of the 16 access types. A bit set to 1 indicates a legal access type for that page and segment type. For instance, looking at Table 3-1, bit 0 would be set to 0 in every record of the access control table because "undefined write" is not a legal memory access type. The bits for unused or undefined access types would all be set to 0 (bits 0, 1, 4, 6-9, 12, 14, and 15). For every segment type, the record for page type "0" would be all 0's (no access).

Once the operating system initializes the access control table, the XMM automatically checks the table before translating each address.

### Illegal Accesses

If an illegal access or segment-not-mapped error occurs, the XMM aborts the program responsible and transfers control to the supervisor's error handler. In addition, the logical page number and access type of the illegal reference are stored in one of the XMM's error registers (refer to Chapter 6).

## MEMORY SHARING

Access control is particularly important when processes are sharing memory. Memory can be shared either by the segment or by the individual page.

### Segment Sharing

A physical segment of memory (128 pages, which may or may not be contiguous) can be shared by two or more processes (or by two or more logical segments in the same process) by placing the same page table pointer in each of the segment table records for the shared logical segments. A segment could conceivably be segment 3 in one process' logical address space and segment 7 in another's. However, the fifth page in the first process' third segment would still be the fifth page in the second process' seventh segment. The page type for each page in a shared segment is the same for all processes sharing the segment. The degree of access varies with the segment type for each process.

A major advantage of segment sharing over individual page sharing is that each shared page exists in only one (shared) page table. If a shared page is swapped out to disk to make room for another process (refer to Chapter 4), the page resident bit must be updated in only one page table. Also, if all processes share the same logical segment, the segment active bit can be set to 1 in all segment tables. Once loaded into the TLB, the page table for that segment can be left there indefinitely for faster address translations.

Segment sharing minimizes system overhead.

### Page Sharing

Individual pages may be shared by two or more processes (or by two or more logical pages in the same process) by placing the same physical page address in the page table records of each of the shared logical pages. Only the physical page in memory is shared, not the entire page table. The shared page can be in a different logical address for each process. Each process may have a different segment type and page type for the shared page. Though more flexible than segment sharing, individual page sharing requires more record keeping by the operating system. For each individual shared page, a record must be kept of all the processes using the page and where the page exists in each logical address space. An individual shared page may exist in a large number of page tables, each of which must be updated if the shared page is swapped out to disk (refer to Chapter 4).

Individual page sharing is best applied to pages that are never swapped out to disk. An example is logical page 0, which holds the interrupt vector tables for the MC68010 processor.

### Logical Addresses for Shared Memory

If the shared pages or segments are located in different logical address spaces for each of the sharing processes, then an address within the shared memory cannot be accessed from within the shared memory. Since a direct-memory reference would have to be different for each logical address space, it cannot be correct for all of them. Data or code in a shared page or segment occurring in different logical address spaces must be "position independent."

# Chapter 4

# MEMORY ALLOCATION AND VIRTUAL MEMORY

The operating system allocates memory in 4K-byte pages for up to 15 concurrent processes. When all of memory is in use and more is requested, the operating system can provide more memory by temporarily "swapping" existing pages out to the hard disk. This chapter discusses memory allocation and how the XMM's physical page table supports both page swapping and virtual memory.

## MEMORY ALLOCATION

### Segment Table Allocation

If all the XMM's segment tables are in use and a new process is waiting to be run, the operating system simply reassigns one of the segment tables to the new process. Since the XMM never modifies segment table (or page table) entries, the reassigned table need not be copied out to main memory before it is overwritten.

### Page Swapping Without Virtual Memory

When a process starts, or requests additional memory, the operating system examines a pool of available physical pages and allocates as many as the process needs. The allocated pages need not be "next to" each other in physical memory, as all pages in the pool are equally valuable (assuming that all the memory boards are the same). Pages released by a process are returned to the pool for reassignment.

If more memory is requested and none is available, the operating system first copies (swaps) one or more pages out to the hard disk, then re-allocates those pages to the requesting process. The copied pages must be brought back into main memory before the process that owns them resumes execution. The XMM keeps a log of page usage in its physical page table to help the operating system to find the least-used pages to swap out to the disk.

### The Physical Page Table

The physical page table on the XMM board has 4,096 records, one for each page in physical memory. Each record has 2 bits: a page referenced bit and a page modified bit. The XMM sets the page referenced bit to 1 whenever the corresponding physical page is accessed by the MC68010 processor. The operating system should periodically test and clear all of the page referenced bits and, when it needs to swap some pages out to the disk, select the pages whose referenced bits have not been reset to 1 for the longest time.

The XMM sets the page modified bit to 1 whenever the corresponding physical page is written to by the MC68010 processor. The operating system should clear this bit when a page is first read in from the disk. (Also, the operating system should set the appropriate page modified bit to 1 whenever it instructs a DMA device to write into a page of memory.) If the page modified bit is zero when the page is about to be swapped out to the disk, the copy operation can be skipped because the page has not been modified and the copy on disk is still valid.

The XMM can set the bits in the physical page table, but only the operating system can clear them.

## VIRTUAL MEMORY

Virtual memory is a technique of memory management that allows any program, without modification, to be run without having all of its pages in memory at the same time. The pages not in use are kept on hard disk and swapped into main memory only as they are needed. The MC68010 processor and several features of the XMM board (such as the physical page table) support the development of virtual memory operating systems.

### The Working Set of a Process

Virtual memory relies on the fact that most programs actively use only a fraction of their pages at any one time. The current actively used pages of a process are known as its "working set". The working set is not constant, but changes throughout the life of the process. If the operating system's estimate of the working set is larger than necessary, memory is wasted; if the estimate is too small, the delays of constantly swapping pages to and from the disk (thrashing) become unacceptable. The goal of virtual memory operating systems is to keep the optimum working set for each process in main memory.

### The Page and Page Table Resident Bits

As mentioned in Chapter 2, a page table resident bit in each record of the segment table is set to 1 if the page table for that segment resides in main memory at the address indicated by the page table pointer. If this bit is zero when the segment is accessed, then the page table has been stored on disk and a page fault will be generated to request the operating system to fetch the page table back into main memory. The page table resident bit is checked only if the segment is mapped (see Figure 2-3).

A page resident bit in every page table record is set to 1 if the corresponding page resides in main memory at the location indicated by the physical page number entry of the page table record. If the bit is zero when the page is accessed, then the page has been stored on disk and a page fault will be generated to request the operating system to fetch the page back into main memory. The page resident bit is checked only after the legality of the access is verified (see Figure 2-3).

The page resident and page table resident bits must be updated by the routine handling the page faults; they are never changed by the XMM board.

### Page Faults

When a process references a page (or a page table) which is not resident in main memory, the XMM sends a page fault (bus error) signal to the MC68010 processor. The MC68010 saves the state of the current instruction and "traps" to its page fault handler through the bus error trap vector. The page fault handler allocates a free page to the process and has the disk controller transfer the requested page (or page table) from the disk to the free page of memory. During the disk transfer, another process is allowed to run. When the transfer is complete, the appropriate page table (or segment table) is updated to reflect the existence and location of the new page (or page table). The suspended process is then resumed from where its last instruction was interrupted.

### The Working Set and the Physical Page Table

Just as a non-virtual operating system uses the page referenced bit in the physical page table to find the least-used pages for swapping, so a virtual memory system can use the same bit to remove inactive pages from the working set of each process. All of the methods used for "aging" pages require the operating system to periodically examine and clear the referenced bits, and the more often it does so, the more accurately it knows how long each page has been idle. However, the more frequently the table is checked, the higher the system overhead (due to time spent checking).

Under virtual memory, as with swapping, the modified bit in the physical page table can be used to avoid unnecessary disk write operations when a page is removed from a process's working set.

# Chapter 5

# MEMORY MAPPING FOR THE Z80

Memory mapping for the Z80 processor is much simpler than for the MC68010. The Z80 can access only 16 pages (64K) of memory, so separate page and segment tables are unnecessary. The Z80 pages have no access control and do not support virtual memory. The XMM assumes that all Z80 pages are mapped, are resident in main memory, and are accessible in any mode. Both I/O and memory references are translated using the same table. At power up and reset, the operating system should initialize the Z80 page tables before enabling address translation.

## Z80 ADDRESS TRANSLATION

The Z80 processor's 16 address lines allow it to address up to 64K bytes of memory (16 pages of 4K bytes each). During an address translation, the logical page number on the Z80's upper 4 address lines (A15 through A12) selects a 12-bit physical page number directly from the page table. A Z80 address may be mapped into any physical page in the 16 megabyte address space. The physical page number is output to the S-100 bus and combined with the byte number from bits A11 through A0 to form the complete physical address. The byte number is not translated, but goes directly from the XPU to the S-100 bus. Note that Z80 references to the top 64K of physical memory are not converted into I/O references.

### The Z80 Page Table

The Z80 page table is actually a MC68010 segment table used as a Z80 page table. Any one (or more) of the XMM's 16 segment tables can be used to map Z80 pages. The only entry in a Z80 page table record is the physical page number assigned by the operating system to that logical page; there is no page resident bit or designated page type. As shown in Figure 5-1, Z80 addresses are translated in the segment/page table, and the TLB is not involved.

One of the segment tables used as a Z80 page table must also serve as an error register table. The XMM has separate Z80 and error map registers to keep track of segment table assignment.

The segment table's mode control line selects the segment mode entry because the Z80's physical page numbers are stored where the MC68010's mode entries normally reside.

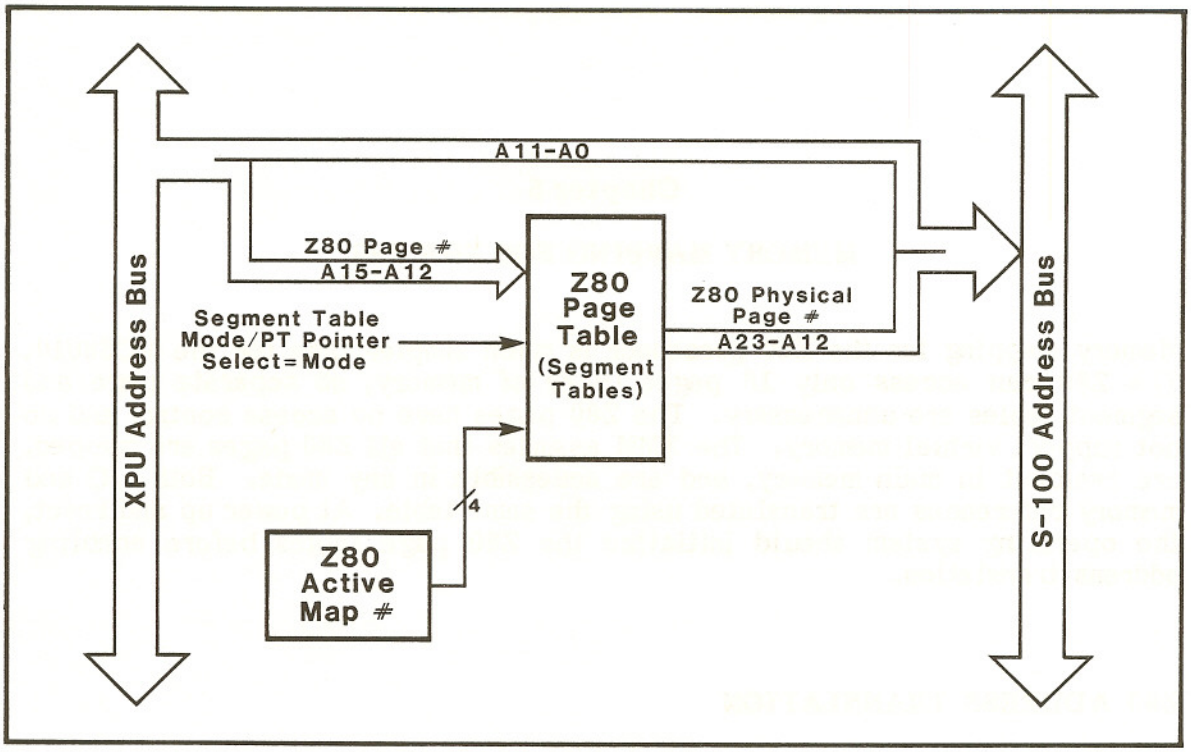


**Figure 5-1: ADDRESS TRANSLATION FOR THE Z80**

### Z80 Page Sharing

Individual pages may be shared between Z80 and MC68010 processes (or between Z80 processes) by placing the same physical page number in the page table records of the logical pages corresponding to the shared physical page.

### Input and Output

Control signals for each Z80 address indicate either a memory reference or an I/O reference. Since I/O references are mapped with the same page table used for memory mapping, the user program has no control over bits A23 through A12 of a physical I/O address. Therefore, I/O devices which communicate directly with the Z80 processor must ignore address lines A23 through A12 on the S-100 bus.

## ACCESS CONTROL FOR THE Z80

Use of the Z80 processor in multi-user systems should be strictly controlled by the operating system. There is almost no access control provided for the Z80 processor, and a faulty Z80 program could crash the entire system.

The Z80 processor can be inhibited from modifying any of the XMM internal registers or tables by setting a Z80 disable bit in the XMM control register (see Chapter 6). While this bit is set, the Z80 cannot access the XMM as an I/O device. This feature prevents the Z80 from manipulating memory not assigned to it, but it does not prevent improper Z80 I/O operations. The disable Z80 bit may be set by either processor, but it may be cleared only by the MC68010.

# Chapter 6

# XMM REGISTERS AND PORT OPERATIONS

This chapter summarizes the I/O port operations that access the XMM's tables and registers. The 12-bit address of each port is given in hexadecimal notation, followed by the port's bit assignments. The low byte of the port address (FCh) selects the XMM board, and the lower 4 bits of the high byte of the address selects the specific port. Since address bits A15 through A12 are not specified for any port, each port will appear in 16 different locations within the 64K-byte I/O address space. Both the Z80 and MC68010 processors can address the XMM's ports, but the Z80 must use the byte latch to do so (refer to the last section of this chapter). Registers that serve several purposes must be initialized correctly for the intended operation.

## THE CONTROL REGISTER

Port Address: CFCh (write only)

The control register sets the operating mode of the XMM, and holds data to be written into the physical page table. A hardware reset clears all control register bits to 0. To update the control register, write the new 16-bit value to the control register port, with the following bit assignments:

**D0 through D7** - Reserved; must be 0's when written.

**D8** - Bit 8 is set to 1 to enable mapping for the MC68010 processor; clearing this bit disables both mapping and access control. The state of this bit can be read in bit 8 of the status register.

**D9** - Bit 9 is set to 1 to enable mapping for the Z80 processor; clearing this bit disables Z80 mapping and limits Z80 addresses to the bottom 16 pages of physical memory. The state of this bit can be read in bit 9 of the status register.

**D10** - Bit 10 is set to 1 to disable the Z80's access to XMM tables and registers; only the MC68010 (or a hardware reset) can clear this bit and re-enable Z80 access. The state of this bit does not affect Z80 mapping.

**D11 and D13** - Reserved for diagnostic purposes.

**D12** - Bit 12 is set to 1 to partially disable MC68010 supervisor mode mapping for diagnostic purposes.

**D14 and D15** - These are the page-modified (D14) and page-referenced (D15) bits to be read into the physical page table during physical page table operations.

## THE STATUS REGISTER

### Read Current Status

Port Address: CFC (read only)

The validity of the data in the status register depends on the type of operation most recently performed and the port address from which the register is read. A "read current status" latches the error and mapping bits (D8 through D12) and sends them to the processor; the other bits are invalid. The latched bits may be re-examined by a "read latched status". A hardware reset clears all status register bits to 0.

**D0 through D7** - Not used; may be 0's or 1's when read.

**D8 and D9** - These bits reflect the state of the same two bits in the control register (MC68010 and Z80 mapping, respectively).

**D10 through D12** - The number (0-7) of the last error detected by the XMM (it will remain latched until a "read current status and clear errors" is performed). The error codes are as follows:

0. No error.
1. Not used.
2. Page fault (page not in main memory).
3. Illegal access (access, page, and segment type combination illegal).
4. TLB error (TLB does not load correctly from main memory).
5. TLB error after segment active bit cleared (same as above).
6. Page table fault (page table not in main memory).
7. Segment not mapped.

The error codes are also indicated by LED's D3-D5 on the XMM board (see Appendix A).

**D13** - The access bit (1 = legal, 0 = illegal) of the last record referenced by an access control table operation. This bit is valid only if a "read latched status" is performed immediately after the record is referenced. If the operation changed the record, bit 13 reflects the state prior to the change.

**D14 and D15** - The page-modified (D14) and page-referenced (D15) bits of the last record referenced by a physical page table operation. These bits are valid only if a "read latched status" is performed immediately after the record is referenced. If the operation changed the record, these bits reflect the state prior to the change.

### Read Current Status and Clear Errors

Port Address: 9FCh (read only)

This is identical to a read current status operation, except that after D8 through D15 are latched and sent to the processor, the error bits (D10 through D12) are cleared to 0. The access, page-referenced, and page-modified bits are invalid.

### Read Latched Status

Port Address: 8FCh (read only)

A "read latched status" transfers the last latched value of the status register to the processor. The meaning of the status bits depends on which of the four operations that latch the status register was performed last:

1. Read current status
2. Read current status and clear errors
3. Test or change the physical page table
4. Test or change the access control table

Each of the four latching operations destroys the data latched by a previous operation. When the access bit is valid, the modified and referenced bits are invalid, and vice versa. Reading the latched status does not affect the current state of the status register.

## PHYSICAL PAGE TABLE OPERATIONS

### Test Page Referenced and Page Modified

Port Address: 8FCh (write only)

When a physical page address is output to this port, the page-modified and page-referenced bits for the page appear in bits 14 and 15 of the status register and are latched immediately. To examine these bits, do a read latched status before another latching operation can destroy the data.

**D0 through D3** - Not used; may be 0's or 1's.

**D4 through D15** - Physical page address.

### Test and Change Page Modified

Port Address: 9FCh (write only)

Identical to test page operation, except that after the status register is latched, the page-modified bit of the selected page is set to the value of bit 14 in the control register.

### Test and Change Page Referenced

Port Address: BFCh (write only)

Identical to test page operation, except that after the status register is latched, the page-referenced bit of the selected page is set to the value of bit 15 in the control register.

## LOGICAL ADDRESS POINTER

Port Address: EFCh (read and write)

The Logical Address Pointer (LAP) has several functions with varying bit assignments, and it is up to the programmer to load this register correctly for the operation intended. The LAP is used to access the following:

1. Access control table
2. Error register table
3. Segment table mode entry
4. Segment table page table pointer entry
5. TLB entries
6. TLB valid table
7. Z80 page table

These operations are discussed in subsequent sections of this chapter.

## ACCESS CONTROL TABLE OPERATIONS

### Test Access

Port Address: 8FCh (write only)

To check the legality of a given combination of access type, page type, and segment type, the desired access type must first be output to the LAP. When the page and segment types are output to the test access register (bit assignments shown below), the selected access bit appears in bit 13 of the status register and is immediately latched. To check the access, do a read latched status before another latching operation destroys the data.

**D0, D4-7, D13-15** - Not used; may be 0's or 1's.

**D1 through D3** - Page type (0-7).

**D8 through D12** - Segment type (0-31).

### Test Access LAP

Port Address: EFCh (read and write)

When used to address the access control table, the LAP bit assignments are as follows:

**D0 through D3** - Bits 0 through 3 correspond to access type bits R/-W, FC0, FC1, and FC2 respectively (refer to Table 3-1).

**D4 through D15** - Reserved; must be 0's when written, undefined when read.

### Test and Change Access

Port Address: AFCh (write only)

This is identical to a test access operation, except that after the status register is latched, both the "no-read access" bit (D4) and the "no-write access" bit (D5) are written into the appropriate access control table locations for the specified function codes (D1 through D3) in the LAP. The no-read and no-write bits are inverted when they are read back from the status register (i.e., "read ok" and "write ok").

**D0, D6, D7, D13-15** - Not used; may be 0's or 1's.

**D1 through D3** - Page type (0-7).

**D4 and D5** - No-read access and no-write access, respectively.

**D8 through D12** - Segment type (0-31).

Bits D4 and D5 are written together because the read/write bit of the access type (D0 in the LAP) is ignored during the "change" portion of a test-and-change-access operation. If you want to change only the read or write access for a given page type, segment type and access type, you must first know the status of the access bit you don't want to change:

To change the read access only:

1. Do a test-access operation, but set D0 in the LAP to 0 (to define a write access).

2. Do a read-latched-status operation. If the access bit in the status register (D13) is 1, then the no-write bit is 0; if the access bit is 0, then the no-write bit is 1.

3. Do a test-and-change-access operation with D0 in the LAP set to 1 (to define a read access). Set the no-read access bit as desired, but set the no-write access as defined above in step 2.

4. Do another read-latched-status operation to verify the old value of the access bit for this (read) access type, page type and segment type.

To change the write access only:

1. Do a test-access operation, but set D0 in the LAP to 1 (to define a read access).

2. Do a read-latched-status operation. If the access bit in the status register (D13) is 1, then the no-read bit is 0; if the access bit is 0, then the no-read bit is 1.

3. Do a test-and-change-access operation with D0 in the LAP set to 0 (to define a write access). Set the no-write access bit as desired, but set the no-read access as defined above in step 2.

4. Do another read-latched-status operation to verify the old value of the access bit for this (write) access type, page type and segment type.

## SEGMENT TABLE OPERATIONS

### Segment Table Mode Entry

Port Address: 0FCh (read and write)

Reading this port returns the segment type, page-table-resident bit, and segment mapped bit of the map and segment number held by the LAP (the local page number in the LAP is ignored). To write new mode entries to this port, first output the desired map and logical segment numbers to the LAP. The new mode entry must have the following bit assignments:

**D0 through D7, and D13** - Not used; must be written as 0's, but may be 0's or 1's when read.

**D8 through D12** - Segment type (0-31) used to define the page types.

**D14 and D15** - Page table resident and segment mapped, respectively.

### Segment Table LAP

Port Address: EFCh (read and write)

For segment table mode or page table pointer entries, the bit assignments for the LAP are as follows:

**D0 through D3** – Bits 0 through 3 hold the segment table map number (0-15).

**D4 through D10** – The logical local page number (0-127), which is ignored during segment table operations.

**D11 through D15** – The logical segment number (0-31).

### Segment Table: Page Table Pointer

Port Address: 1FCh (read and write)

Reading this port returns the address of the page table in main memory for the map and segment numbers held by the LAP (the local page number in the LAP is ignored). To write a new page table pointer to this port, first output the desired map and logical segment numbers to the LAP. The bit assignments for the new page table address are as follows:

**D0 through D15** – The upper 16 bits of the physical address of this segment's page table in main memory. Page tables must be stored in memory on even 256-byte boundaries. The address of any page table entry is found by:

(page table pointer)*256 + 2*(local page number)

Whenever a page table pointer is changed, the XMM automatically clears to 0 the segment active bit for that segment and map.

### Z80 Page Table

Port Address: 0FCh (read and write)

Reading this port returns a physical page number for the logical Z80 page and map numbers held by the LAP. To write a new physical page number to this port, first output the map number of the Z80 page table (0-15) and the desired logical page number (0-15) to the LAP. The bit assignments for the new physical page are as follows:

**D0 through D3** – Not used; must be written as 0's, but may be 0's or 1's when read.

**D4 through D15** – The 12-bit base address of a 4K-byte page in physical memory.

### Z80 Page Table LAP

Port Address: EFCh (read and write)

When the LAP is used to address a Z80 page table, the following bit assignments apply:

**D0 through D3** - The segment table map holding the Z80 page table.

**D4 through D10** - Not used; must be 0's when written, but may be either 0's or 1's when read.

**D11** - Reserved; must be 1 when written, but may be 1 or 0 when read.

**D12 through D15** - The logical Z80 page number (0-15) to be accessed.

### Error Register Table

Port Address: 1FCh (read and write)

Reading this port returns the logical page number and access mode of the last error of this type to occur (determined by the map number and error type in the LAP). To write a new page number and access type to this port, first load the LAP with the desired map number and error type. The bit assignments for the error registers are as follows:

**D0 through D3** - The access mode during last error of the type specified: R/-W, FC0, FC1, and FC2 respectively.

**D4 through D15** - The combined logical local page number (D4-D10) and logical segment number (D11-D15) where the error occurred.

### Error Register LAP

Port Address: EFCh (read and write)

When the LAP is used to address an error register, the following bit assignments apply:

**D0 through D3** - The segment table map holding the error registers.

**D4 through D10** - Not used; must be 0's when written, but may be 0's or 1's when read.

**D11** - Not used; must be 1 when written, but may be 1 or 0 when read.

**D12 through D14** - The error type (0-7) specifies both the type of error and the number of the error register assigned to it. The error codes are as follows:

0. No error.
1. Not used.
2. Page fault (page not in main memory).
3. Illegal access (access, page, and segment type combination illegal).
4. TLB error (TLB does not load correctly from main memory).
5. TLB error after segment active bit cleared (same as above).
6. Page table fault (page table not in main memory).
7. Segment not mapped.

**D15** - Not used; must be 1 when written, but may be 0 or 1 when read.

## TLB PAGE TABLE OPERATIONS

Port Address: 2FCh (read and write)

The TLB is maintained automatically, but it can be accessed directly for diagnostic purposes, or to correct an entry that has been changed in main memory.

Reading this port returns the page type, page resident bit, and physical page number of the map and logical page number in the LAP. To write new TLB entries to this port, first output the desired map and logical page numbers to the LAP (same as segment table LAP). The bit assignments for the TLB registers are as follows:

**D0** - Page resident bit (1 = page in main memory at physical address on bits D4 through D15, 0 = page on disk and bits D4 through D15 are meaningless).

**D1 through D3** - Page type (0-7).

**D4 through D15** - The 12-bit base address of a 4K-byte page in physical memory (if D0 is 1).

## TLB VALID AND SEGMENT ACTIVE TABLES

Port Address: 3FCh (read and write)

Reading this port returns the TLB valid and segment active bits for the map and logical page numbers in the LAP. Locating just the TLB valid bit does not require the map number, and finding just the segment active bit does not require the local page part (D4-D10) of the logical address. To write new TLB valid and segment active bits to this port, first output the desired map and logical address numbers to the LAP (same as segment table LAP). The bit assignments for this port are as follows:

**D0 through D7** - Not used; must be 0's when written, but may be 0's or 1's when read.

**D8 and D9** - Segment active and TLB valid bits, respectively.

**D10 through D15** - Not used; must be 0's when written, but may be 0's or 1's when read.

## ACTIVE MAP REGISTERS

### MC68010 User Map

Port Address: 4FCh (read and write)

Reading this port returns the segment table map number (0-15) of the currently active MC68010 user process. If a new map number is output to this port, the next MC68010 user memory reference will be translated with the new map.

**D0 through D3** - MC68010 user active map number.

**D4 through D15** - Not used; must be written as 0's, but may be 1's or 0's when read.

**Caution:** The MC68010 may prefetch one or two additional instructions before it outputs the new map number to the XMM.

### MC68010 Supervisor Map

Port Address: 5FCh (read and write)

Reading this port returns the segment table map number (0-15) of the MC68010 supervisor. If a new map number is output to this port, the next supervisor memory reference will be translated with the new map.

**D0 through D3** - MC68010 supervisor map number.

**D4 through D15** - Not used; must be written as 0's, but may be 1's or 0's when read.

**Caution:** The MC68010 may prefetch one or two additional instructions before it outputs the new map number to the XMM.

### Z80 Map

Port Address: 7FCh (read and write)

Reading this port returns the segment table map number (0-15) of the active Z80 process. If a new map number is output to this port, the next Z80 memory reference will be translated with the new map. A Z80 page table occupies the segment mode bits of 16 segment table records. The same segment table can hold both the error register and a Z80 page table.

**D0 through D3** - Z80 active map number.

**D4 through D15** - Not used; must be written as 0's, but may be 1's or 0's when read.

### Error Register Map

Port Address: 6FCh (read and write)

Reading this port returns the segment table map number (0-15) holding the error registers. The error registers occupy the page table pointer bits of eight segment table records. The same segment table can be used for error registers and a Z80 page table.

**D0 through D3** - Error register map number.

**D4 through D15** - Not used; must be written as 0's, but may be 1's or 0's when read.

## THE BYTE LATCH REGISTER

Port Address: DFCh (read and write)

This register allows the 8-bit Z80 processor to access the 16-bit I/O ports on the XMM. To read an XMM port, the Z80 first reads the low byte (D0-D7) from the desired port address, then reads the high byte (D8-D15) from the byte latch. To write to a port, the Z80 first outputs the low byte to the byte latch, then outputs the high byte to the desired port address.

The Z80 must use the "in reg, (c)" and "out (c), reg" I/O instructions to access the XMM. Load the C register with the XMM's board address (FCh), and load the B register with the desired port address (0-Fh). Alternately, the BC register can be loaded with the 12-bit XMM I/O port address.

Since the same latch is used for both input and output, no XMM byte I/O operations should occur between the two steps required for either input or output. Routines that access the XMM at interrupt level should save and restore the byte latch if they use byte mode.

Table 6-1 provides a summary of the XMM's registers and their bit assignments.

## Table 6-1: SUMMARY OF XMM REGISTERS

| Port (0-Eh) | Register Bit Assignments | | | | | | | | | | | | | | | | Register Name |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0: | MPD | PTR | | ST4 | ST3 | ST2 | ST1 | ST0 | | | | | | | | | SEG MODE |
| 1: | P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 | P9 | P8 | PG TBL PTR |
| 2: | P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 | P15 | P14 | P13 | P12 | PT2 | PT1 | PT0 | RES | TLB |
| 3: | | | | | | | TBA | SGA | | | | | | | | | ACTIVE |
| 4: | | | | | | | | | | | | | MP3 | MP2 | MP1 | MP0 | 68USR MAP |
| 5: | | | | | | | | | | | | | MP3 | MP2 | MP1 | MP0 | 68SUP MAP |
| 6: | | | | | | | | | | | | | MP3 | MP2 | MP1 | MP0 | ERROR MAP |
| 7: | | | | | | | | | | | | | MP3 | MP2 | MP1 | MP0 | Z80 MAP |
| 8:o | P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 | P15 | P14 | P13 | P12 | | | | | TEST PR,PM |
| 8:o | | | | ST4 | ST3 | ST2 | ST1 | ST0 | | | | | PT2 | PT1 | PT0 | | TEST ACC |
| 8:i | REF | MOD | ACC | ER2 | ER1 | ER0 | ZEN | 6EN | | | | | | | | | LCH STATUS |
| 9:o | P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 | P15 | P14 | P13 | P12 | | | | | CH PG MOD |
| 9:i | REF | MOD | ACC | ER2 | ER1 | ER0 | ZEN | 6EN | | | | | | | | | STS,CL ERR |
| A:o | | | | ST4 | ST3 | ST2 | ST1 | ST0 | | | NRD | NWR | PT2 | PT1 | PT0 | | CH ACCESS |
| B:o | P23 | P22 | P21 | P20 | P19 | P18 | P17 | P16 | P15 | P14 | P13 | P12 | | | | | CH PG REF |
| C:o | REF | MOD | | DIA | | DZI | ENZ | E68 | | | | | | | | | CONTROL |
| C:i | REF | MOD | ACC | ER2 | ER1 | ER0 | ZEN | 6EN | | | | | | | | | STATUS |
| D:o | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | | | | | | | | | BYTE LATCH |
| D:i | | | | | | | | | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | BYTE LATCH |
| E: | A23 | A22 | A21 | A20 | A19 | A18 | A17 | A16 | A15 | A14 | A13 | A12 | MP3 | MP2 | MP1 | MP0 | LOG AD PTR |

Field Definitions

| | |
|---|---|
| 6EN | 68010 mapping enabled |
| ACC | Access legal |
| Ann | Logical address |
| DIA | Diagnostic mode |
| DZI | Disable Z80 access to XMM ports |
| Dnn | Internal data bus bits |
| E68 | Enable 68010 mapping |
| ENZ | Enable Z80 mapping |
| ERn | Error code (0-7) |
| MOD | Physical page modified |
| MPD | Segment mapped |
| MPn | Map number |
| NRD | No read access |
| NWR | No write access |
| PTR | Page table resident |
| PTn | Page type (0-7) |
| Pnn | Physical address |
| REF | Physical page referenced |
| RES | Page resident |
| SGA | Segment active |
| STn | Segment type (0-31) |
| TBA | TLB valid |
| ZEN | Z80 mapping enabled |

## Appendix A

## LED CODES

There are five LED's near the top of the XMM board. The yellow diode, D1, flashes on each Z80 address translation. The green diode, D2, flashes on each MC68010 address translation. The red diodes, D3-D5, indicate errors, as shown in the following table (a "1" means that the diode is ON). The LED's stay on after an error until reset by the software.

| D5 | D4 | D3 | Error |
|---|---|---|---|
| 0 | 0 | 0 | No errors |
| 0 | 1 | 0 | Page fault |
| 0 | 1 | 1 | Illegal access |
| 1 | 0 | 0 | TLB loading error |
| 1 | 0 | 1 | TLB loading error after segment clear |
| 1 | 1 | 0 | Page table fault |
| 1 | 1 | 1 | Segment not mapped |

# Appendix B

## XMM/XPU SIGNAL DEFINITIONS

The 34-conductor cable linking the XMM and XPU conveys both control and address signals for memory mapping. This appendix defines the control signals. The pin assignments for all of the XMM/XPU signals are shown in Table B-1. An asterisk indicates that the signal is active low; signals without the asterisk are active high.

### Signals for Z80 Mapping

**Memory Request** - The Z80 MREQ* signal indicates the start of a Z80 memory cycle. The XMM uses this signal to adjust the timing of mapped Z80 cycles to correspond to unmapped cycles.

**M1*** - The Z80 M1* signal indicates the start of either an instruction fetch cycle or an interrupt acknowledge cycle. The XMM uses this signal to adjust the timing of mapped Z80 cycles to correspond to unmapped cycles.

**Input/Output Request** - The Z80 IOREQ* signal is true whenever an I/O or interrupt acknowledge cycle is performed. The XMM uses this signal to adjust the timing of mapped cycles to correspond to unmapped cycles.

**Z80 Ready** - The ZRDY signal from the XMM notifies the XPU that the Z80 cycle may proceed. During mapping, Z80 cycles are delayed to allow time for translation of address bits A15 through A12 to S-100 address bits A23 through A12.

### Signals for MC68010 Mapping

**Clock** - The 10MHZ clock signal from the XPU synchronizes the XMM and XPU boards. This signal provides the basic timing for all XMM operations.

**CPU Transfer Enable** - The EN68010 signal notifies the XMM that the XPU is about to switch from the Z80 processor to the MC68010 processor.

**MC68010 Control Signals** AS*, R/W*, and FC0-2 are the Address Strobe, Read/Write, and Function Code control signals from the MC68010 processor. They describe the type of cycle being performed by the MC68010.

**XMM-Active*** - The XMM-Active* signal is true while the XMM board is enabled. This signal forces the XPU to disable the drivers for address lines A23 through A12 and to examine the AOK and MEM/IO* lines to determine the type and validity of the cycle being executed.

**Memory/IO** - The MEM/IO signal from the XMM notifies the XPU to run either a memory cycle or an I/O cycle. This signal can occur only if the XMM has been enabled.

**Address OK** - The AOK signal from the XMM prevents the XPU from starting an S-100 bus cycle until the address has been translated and proven valid. This signal can occur only if the XMM has been enabled.

**Table B-1: XMM/XPU SIGNALS**

| Pin | Signal | Source | Definition |
|---|---|---|---|
| 1 | GND | BOTH | Ground |
| 2 | MREQ* | XPU | Z80 Memory Request Control Line |
| 3 | A15 | XPU | MC68010/Z80 Address Line 15 |
| 4 | 10MHZ | XPU | 10MHz Master Clock |
| 5 | A14 | XPU | MC68010/Z80 Address Line 14 |
| 6 | M1* | XPU | Z80 M1 Cycle Control Line |
| 7 | A13 | XPU | MC68010/Z80 Address Line 13 |
| 8 | GND | BOTH | Ground |
| 9 | A12 | XPU | MC68010/Z80 Address Line 12 |
| 10 | MEM/IO* | XMM | MC68010 Memory or I/O Cycle |
| 11 | IOREQ* | XPU | Z80 I/O Request Control Line |
| 12 | AS* | XPU | MC68010 Address Strobe Control Line |
| 13 | GND | BOTH | Ground |
| 14 | A21 | XPU | MC68010 Address Line 21 |
| 15 | XMM-ACTIVE* | XMM | XMM turned on and mapping |
| 16 | A20 | XPU | MC68010 Address Line 20 |
| 17 | A19 | XPU | MC68010 Address Line 19 |
| 18 | A18 | XPU | MC68010 Address Line 18 |
| 19 | A17 | XPU | MC68010 Address Line 17 |
| 20 | A16 | XPU | MC68010 Address Line 16 |
| 21 | A23 | XPU | MC68010 Address Line 23 |
| 22 | GND | BOTH | Ground |
| 23 | R/W* | XPU | MC68010 Read/Write Cycle Control Line |
| 24 | A22 | XPU | MC68010 Address Line 22 |
| 25 | ZRDY | XMM | Z80 Cycle may proceed (delayed during mapping) |
| 26 | GND | BOTH | Ground |
| 27 | GND | XMM | Ground |
| 28 | | | Unused |
| 29 | FC1 | XPU | MC68010 Function Code Bit 1 Control Line |
| 30 | FC2 | XPU | MC68010 Function Code Bit 2 Control Line |
| 31 | FC0 | XPU | MC68010 Function Code Bit 0 Control Line |
| 32 | GND | BOTH | Ground |
| 33 | AOK | XMM | MC68010 address translation permitted and complete |
| 34 | EN68010 | XPU | Control will transfer from Z80 to MC68010 |

# LIMITED WARRANTY

Cromemco, Inc. ("Cromemco") warrants this equipment against defects in material and workmanship to the original purchaser for ninety (90) days from the date of purchase, subject to the following terms and conditions.

## What Is Covered By This Warranty:

During the ninety (90) day warranty period Cromemco will, at its option, repair or replace this Cromemco product or repair or replace with new or used parts any parts or components, manufactured by Cromemco, which prove to be defective, provided the product is returned to your Dealer as set forth below.

## How To Obtain Warranty Service:

You should immediately notify IN WRITING your Dealer or Cromemco of problems encountered during the warranty period in order to obtain warranty service; first obtain a return authorization number by contacting the Dealer from whom you purchased the product. Then attach to the product:

1. Your name, address, and telephone number,
2. the return authorization number
3. a description of the problem, and
4. proof of the date of retail purchase

Ship or otherwise return the product, transportation and insurance costs prepaid, to your Dealer. If you are unable to receive warranty repair from the Dealer from whom you purchased the product, you should contact Cromemco Customer Support at: Cromemco, Inc., 280 Bernardo Ave., Mountain View, Ca. 94043.

## What Is Not Covered By This Warranty:

Cromemco does not warrant any products, components, or parts not manufactured by Cromemco.

This warranty does not apply if the product has been damaged by accident, abuse, misuse, modification, or misapplication; by damage during shipment; or by improper service. This product is not warranted to operate satisfactorily with peripherals or products not manufactured by Cromemco. Transportation and insurance charges incurred in transporting the product to and from your Dealer or Cromemco are not covered by this warranty.

## Exclusion of Liability, Damages, and Other Warranties:

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. ANY IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE EXCLUDED WHERE SUCH EXCLUSION IS ALLOWED AND OTHERWISE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF PURCHASE OF THIS PRODUCT. IF THIS PRODUCT IS NOT IN GOOD WORKING ORDER AS WARRANTED ABOVE, YOUR SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. CROMEMCO SHALL NOT BE LIABLE FOR INCIDENTAL AND/OR CONSEQUENTIAL DAMAGES FOR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY, EVEN IF CROMEMCO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE AGENTS, DEALERS, AND EMPLOYEES OF CROMEMCO ARE NOT AUTHORIZED TO MAKE MODIFICATIONS TO THIS WARRANTY, OR ADDITIONAL WARRANTIES BINDING ON CROMEMCO ABOUT OR FOR PRODUCTS SOLD OR LICENSED BY CROMEMCO. ACCORDINGLY, ADDITIONAL STATEMENTS WHETHER ORAL OR WRITTEN EXCEPT SIGNED WRITTEN STATEMENTS FROM AN OFFICER OF CROMEMCO DO NOT CONSTITUTE WARRANTIES AND SHOULD NOT BE RELIED UPON. SOFTWARE, TECHNICAL INFORMATION, AND FIRMWARE ARE LICENSED ONLY BY A SEPARATE AGREEMENT ON AN "AS IS" BASIS.

## Limitation on Statute of Limitation and Transferability:

THIS WARRANTY AND THE STATUTE OF LIMITATIONS SHALL RUN CONCURRENTLY WITH ANY ACCEPTANCE PERIOD. THIS WARRANTY IS NOT TRANSFERABLE. NO SUIT, LITIGATION, OR ACTION SHALL BE BROUGHT BASED ON THE ALLEGED BREACH OF THIS WARRANTY OR IMPLIED WARRANTIES OR FOR OTHER CAUSE MORE THAN ONE YEAR AFTER THE DATE OF PURCHASE IN THOSE JURISDICTIONS ALLOWING SUCH A LIMITATION; OTHERWISE NO SUCH ACTION SHALL BE BROUGHT MORE THAN ONE YEAR AFTER THE EXPIRATION OF THIS WARRANTY.

## Other Important Provisions:

Some states do not allow the exclusion or limitation of incidental or consequential damages or limitations on how long an implied warranty lasts, so the above limitation or exclusion may not apply to you. This warranty shall not be applicable to the extent that any provision of this warranty is prohibited by any federal, state, or municipal law which cannot be preempted. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Rev. D 6/14/84

Cromemco®

280 Bernardo Ave.
P.O. Box 7400
Mountain View, CA 94039