Part Number 800-1370-02
Rev: 50 of 30 January, 1986

# Sun 2060 CPU

# Hardware Reference Manual

Sun Microsystems, Inc.,
2550 Garcia Avenue,
Mountain View,
California 94043
(415) 960-1300

# Credits and Trademarks

# Contents

# Tables

# Figures

# 1

# Introduction

# Introduction

This manual describes the Sun 2060 board, a CPU used in Sun-3 systems, and sold as a separate board product. It uses a Motorola 68020 microprocessor running at 16.67 MHz, and it may have a Motorola 68881 FPP coprocessor installed.

The 2060 CPU is based on a standard Revision B VMEbus. It can act as both a bus master or a bus slave. It features Direct Virtual Memory Access, which enables the Ethernet interface and the VME slave interface to access memory through the MMU using virtual addresses.

NOTE *The 2060 is actually* `Revision C VMEbus` *compatible with only two exceptions: 1) It doesn't support 3-byte transfers, and 2) longword accesses must be longword aligned.*

## 1.1. Using this Manual

The following sections provide information to help you use this manual.

### Audience

The audience of this manual includes system programmers, persons who need to understand the board to repair it, and persons interested in interfacing this board to another VMEbus system.

### Manual Organization

The chapters in this manual track the architecture of the Sun 2060 CPU board. Chapter 2 describes the overall function of the board, and provides block diagrams of the memory map and the local busses. Chapters 3, 4, and 5 describe the three major architectural divisions of the board; CPU space, control space, and device space. They list the devices and addresses in these spaces, and they describe how these devices are accessed and used. Chapter 6 describes the VMEbus interface that the 2060 CPU uses to interact with the rest of the system. Appendix A provides additional details about the VMEbus interface, and Appendix B lists the jumpers on the board.

There is also an index, at the end of this manual.

### Fonts in Text

In this manual, we use fonts to make things a little clearer. The most common fonts are Roman, `typewriter`, *italic*, and **bold**. We use them as follows:

Roman
: Roman font is the standard for normal text, just as it appears here.

Typewriter
: `Typewriter font` is mostly used for information in displays.

Italic
: *Italic font* is either used for notes, or it represents a variable for which you or the computer must substitute some real value. For example:

> This field contains a pointer to register *nn*

Bold
: **Bold font** indicates that something deserves more attention than the surrounding text.

## 1.2. Glossary

In order to avoid confusion, this manual uses standard definitions for some common words. These are:

CPU
: The CPU is the Motorola MC68020 and its direct supporting circuits.

Data Bus
: The path that carries data around the board.

FPA
: Floating Point Accelerator — An optional PC board that performs floating point calculations.

FPP
: Floating Point Processor — A Motorola MC68881 that performs floating point calculations as a coprocessor to the MC68020.

MMU
: Memory Management Unit — The unit which separates the physical address bus from the virtual address bus, and which translates virtual addresses to physical addresses.

Physical Address
: An absolute address, after translation by the MMU.

Physical Address Bus
: The bus that carries physical addresses.

Virtual Address
: A relative address, before translation by the MMU.

Virtual Address Bus
: The bus that carries virtual addresses.

## 1.3. References

See the following for additional information:

Intersil 7170 Data Sheet, Intersil 1985 Data Book

Motorola MC68020 User's Manual

Motorola Technical Summary of the MC68881 HCMOS Floating Point Accelerator

Motorola VMEBus Specification Manual, revisions B or C.

## Zilog Z8530 Data Sheet

# Architecture Overview

This chapter describes the basic blocks that make up the 2060 CPU board.

The heart of the 2060 board is a Motorola 68020 microprocessor that runs at 16.67 MHz and supports a Motorola MC68881 floating point processor coprocessor. It uses 8 Kbyte pages, eight 256 Mbyte contexts, and a 32-bit VMEbus.

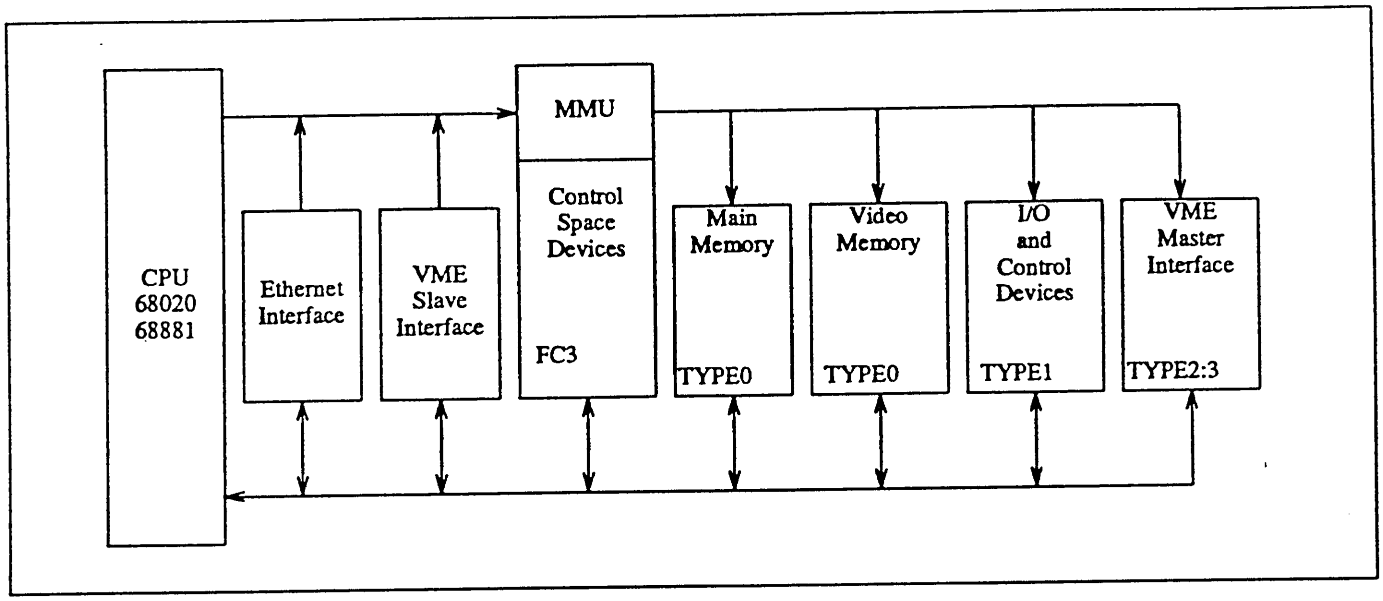
The 2060 provides Direct Virtual Memory Access (DVMA), a feature which allows VME devices to access the entire 256 Megabyte space in each of 8 contexts using virtual addresses. This saves the overhead of having to translate virtual addresses to physical addresses for VMEbus accesses.

Figure 2-1 shows the basic blocks that make up the 2060 CPU, and the busses that connect them. These blocks include the CPU, the Ethernet interface, the VME slave interface, the Memory Management Unit (MMU), the main memory, the video memory, the I/O and control devices, and the VME master interface.

This figure also shows the internal busses that connect these devices. Note that while the data bus connects all blocks in common, the address bus is divided by the MMU into two sections, the virtual address bus and the physical address bus. The MMU translates virtual addresses into physical addresses.

The VME slave interface and the Ethernet interface use DVMA to access memory using virtual addresses, which are translated by the MMU. Except for the fact that it uses virtual addresses, this works similarly to the DMA circuits in other computers.

Figure 2-1 *System Block Diagram*



The following paragraphs describe each of the blocks:

CPU
: The CPU consists of a Motorola 68020 microprocessor running at 16.67 MHz, and a Motorola 68881 Floating Point Coprocessor. It issues virtual addresses on the virtual address bus, and it controls the VME bus using the VME master interface in physical address space.

Ethernet Interface
: The Ethernet interface lives in virtual address space. It uses DVMA to access physical memory through the MMU.

VME Slave Interface
: The 2060 CPU uses its VME slave interface to accept input from other VMEbus masters. Note that the slave interface uses the virtual address bus. This is described more in the chapter *VME Interface*, and in *Appendix A*.

Main Memory
: The 2060 can have between 2 and 16 megabytes of physical memory. It is described later in this chapter.

Video Memory
: The video memory is a portion of main memory that holds the image of the current video display. It is described later in this chapter.

I/O and Control Devices
: I/O and control devices are used to directly control access to the 2060's resources. They include things like I/O ports, the keyboard and mouse ports, and various control registers.

VME Master Interface
: The CPU uses the VME master interface to control the VMEbus. This is

described in the chapter *VME Interface*, and in *Appendix A*.

## 2.1. MMU

The MMU translates virtual addresses to physical addresses. In the process, it keeps track of the context, the segment, and the page, and it provides memory protection and updating information. It appears in Figure 2-2.

The following list shows the map's capabilities:

| | |
|---|---|
| Page size | 8 Kbytes |
| Segment size | 128 Kbytes |
| Process size (context) | 256 Mbytes |
| Number of contexts | 8 |
| Segments per context | 2048 |
| Pages per segment | 16 |
| Number of pmegs | 256 |
| Total number of pages | 4096 |
| Total number of segments | 16384 |

Accesses through the map take the following steps:

The context bits are concatenated to the segment number to select an address in the segment map, which puts out an 8-bit Page Map Entry Group (PMEG).

The PMEG is concatenated to the 4-bit page field of the virtual address, and these 12-bits select an address in the page map. The page map puts out 32-bits, containing 8-bits of information about the page (type, privilege, etc), and 19 bits of high-order address information which actually points to a physical page.

The physical page number is concatenated with the 13-bit physical offset from the virtual address, and this is a complete physical address.

### Page ID bits

Bits 31 through 19 of the page map entry provide the following information about the page:

31 Valid bit. When ON, it means that the page entry is valid and it allows read and execute access to the page.

30 Write access bit. When ON, page is enabled for writing.

29 Supervisor access. When ON, only supervisor access is allowed.

28 Don't cache. Not used.

27 and 26
Type bits, which select device space access as follows:

| Bit 27 Type 1 | Bit 26 Type 0 | Type |
|---|---|---|
| 0 | 0 | Main memory |
| 0 | 1 | I/O space |
| 1 | 0 | VMEBus 16-bit data |
| 1 | 1 | VMEBus 32-bit data |

25 Accessed bit. This bit is set to indicate that the page has been accessed by the CPU or DVMA. This bit does not specify whether the access was a write or a read.

24 Modified bit. This bit is set when the page is modified.

23 through 19
Reserved bits. These bits are reserved for future use.

18 through 0
Physical page number. These bits identify an actual physical page. They are concatenated with bits 0 through 12 of the virtual address to form an actual physical address.

The MMU is loaded dynamically by the kernel, which keeps track of where it put things.

## Context Register

Bits 0 through 2 of the context register identify the current context; these bits are appended to virtual addresses before they enter the map.

## Segment Map

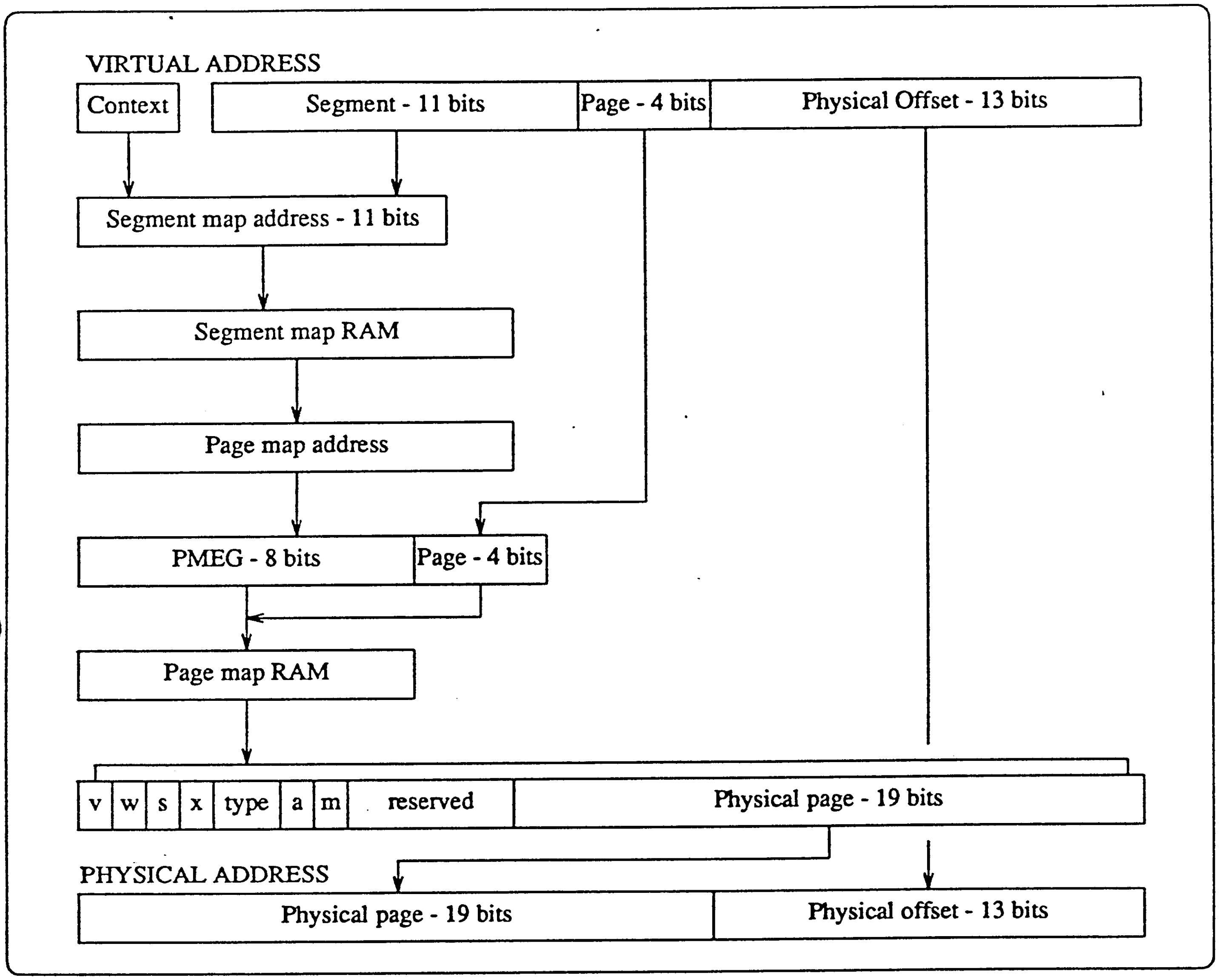
The segment map is a 16K (16384) X 8 RAM bank. Its 14 address inputs are formed by concatenating the contents of the context register (3 bits) to the 11 most significant bits of the virtual address (A17 through A27). An 8-bit PMEG index number lives at each of these addresses.

## Page Map

The page map is a 4K X 32-bit RAM bank. Its 12 address inputs are formed by concatenating the PMEG output by the segment map with the page field of the virtual address (bits 13 through 16). Effectively, the PMEG index identifies one of 256 sections, and virtual address bits 13 through 16 select one of 16 pages within that section.

Direct writes to both the segment map and the page map are described in the chapter *Control Space Registers and Utilities*.

Figure 2-2 *Memory Management Unit*



## 2.2. Memory

The 2060 can have between 2M and 16M of memory. The CPU board itself provides either 2M or 4M, and up to 4 expansion boards may supply additional memory in 2M or 4M increments.

All memory organization is done by the MMU; memory appears to the (physical side of the) MMU as a series of contiguous locations in type0 space, spanning from 0x0 to however much is installed.

## 2.3. Video Memory Frame Buffer

The video frame buffer is a 128K block of memory starting at physical location 0xFF000000. It contains the image currently on the (black and white) monitor screen; changes to this area change the monitor screen display.

The copy block is a 128K block that starts at physical address 0x00100000. If copy mode is enabled, it contains a copy of the data in the video frame buffer; changing this block automatically updates the frame buffer.

## 2.4. Reset

The 2060 has 5 possible sources of reset: 1) power-on reset, 2) VMEbus SYS-RESET, 3) watchdog reset, 4) a user reset switch, and 5) CPU reset. The following paragraphs describe each:

Power-On Reset

Power-on reset is active for 100MS after the power supply voltage reaches 4.5 volts. It resets the CPU and clears the system enable register, forcing the boot state. It resets the diagnostic register, lighting all the LEDs, and it resets other devices listed in the table below.

VMEbus Reset

A jumper selects between VME SYSRESET Master, and SYSRESET slave modes.

Watchdog Reset

A watchdog circuit on the CPU generates a watchdog reset whenever the CPU detects a double bus fault. This resets the same devices as as a power-on reset, and it is recorded in the bus error register as the WATCHDOG bit (D1).

User Reset Switch

The user reset switch resets the same devices as the power-on reset. It is recorded in the bus error register as the WATCHDOG bit.

CPU Reset

The CPU reset occurs when the CPU executes a RESET instruction.

The following table shows the devices that each source resets:

2

# Architecture Overview

Table 2-1 *Reset Sources and Their Results*

| Device/VMEbus signal | Power On | VMEbus Reset | Watch-dog | User Switch | CPU Reset |
|---|---|---|---|---|---|
| MC68020 CPU | Y | B | Y | Y | Y |
| MC68881 FPP | B | B | Y | Y | |
| VMEbus SYSRESET- | P | | P | P | P |
| VMEbus SYSFAIL- | Y | | Y | Y | Y |
| System enable reg. | Y | B | Y | Y | |
| User DVMA enable reg. | Y | B | Y | Y | |
| Diagnostic reg. | Y | B | Y | Y | |
| Floating point acc. | Y | B | Y | Y | Y |
| Memory err. cont. reg | Y | B | Y | Y | |
| Interrupt reg. | Y | B | Y | Y | Y |
| Serial port | Y | B | Y | Y | |
| Keyboard/mouse port | Y | B | Y | Y | |
| Intel Ethernet I/F | Y | B | Y | Y | Y |

NOTE *Y = always reset the indicated device. B = Bus jumper option: Reset indicated CPU devices if VMEbus SYSRESET is driven active (0) from some VMEbus board. P = CPU-to-bus jumper option: Drive VMEbus SYSRESET- active (0) during indicated reset condition for CPU board. For a complete list of the system jumpers, see Appendix B.*

## 2.5. Addressing Overview

The 2060 architecture is divided into three spaces; device space, control space, and CPU space. The 68020 function codes select these spaces as follows:

```
Func. Code      Space

    0           Reserved
    1           Device space (user data)
    2           Device space (user program)
    3           Control space
    4           Reserved
    5           Device space (supervisor data)
    6           Device space (supervisor program)
    7           CPU space
```

Device space contains all devices in physical memory. These include memory, the VMEbus master interface, and I/O. Control space contains various system control facilities, like the MMU, and some registers. CPU cycles are used for coprocessor cycles (the only coprocessor is a 68881 FPP) and interrupt acknowledges.

Device space, control space, and CPU space are each described in detail in later chapters.

3

# CPU Space

# CPU Space

CPU space accesses are used for coprocessor cycles and interrupt acknowledges. They are identified as CPU space accesses by function code 7, and are further identified by address bits A16 and A17 as follows:

| Cycle Type | A16 | A17 | Cycle Conclusion |
|---|---|---|---|
| Breakpoint cycle | 0 | 0 | BERR (bus error) |
| Ring protection | 1 | 0 | BERR |
| Coprocessor cycle | 0 | 1 | DSACK/BERR |
| Interrupt Acknowledge | 1 | 1 | AVEC/DSACK/BERR |

The Sun 2060 does not use breakpoint cycles or ring protection. The only coprocessor used with the Sun 2060 is the MC68881 Floating Point Processor. For more information on the MC68881 coprocessor cycles, see the MC68020 User's Manual, and the Technical Summary of the MC68881 HCMOS Floating Point Accelerator.

The Sun 2060 supports 7 on-board interrupts and 7 VMEbus interrupts. On-board interrupts are autovectored except for level 6, where the 8530A's provide the vector. All VMEbus interrupts must be vectored. Priority is resolved from the highest level down, with CPU board interrupts having higher priority than off-board interrupts of the same level. The following table shows the CPU board interrupt structure:

| Level | Devices |
|---|---|
| 7 | Non Maskable Interrupt - Real time clock and parity error |
| 6 | Serial controllers (8530A's) |
| 5 | Real time clock |
| 4 | Video vertical interrupt |
| 3 | Ethernet and software interrupt |
| 2 | Software interrupt |
| 1 | Software interrupt |

The Z8530s must be programmed to interrupt on certain conditions and to provide vectors when they do. The vectors can be any of the 68020 user defined vectors. For more information see the *Zilog 8530 SCC Data Sheet.*

Software interrupts are caused by toggling the appropriate bit in the interrupt register (see the chapter *Control Space Registers and Utilities*).

# 4

# Device Space

# Device Space

Device space contains all the devices which are accessed through the MMU, and the FPA, which is accessed through a direct decoding of address bits A28-A31.

Except for the FPA which is addressed when bits A28 through A31 equal 0xE, 68020 function codes select device space accesses as follows:

```
FC1 = device space (user data)
FC2 = device space (user program)
FC5 = device space (supervisor data)
FC6 = device space (supervisor program)
```

The type bits (page map bits 27 and 28) identify the type of access further:

| Bit 27 Type1 | Bit 26 Type2 | Type | Space accessed |
|---|---|---|---|
| 0 | 0 | 0 | Memory |
| 0 | 1 | 1 | I/O |
| 1 | 0 | 2 | VME 16-bit |
| 1 | 1 | 3 | VME 32-bit |

Once the above bits select a device access type, other bits select the actual address. The following sections show the devices in each type, and the bits used to address them:

The following table shows the complete device space addressing:

Table 4-1 *Device Space Physical Addresses*

| Device | Type | Physical Base | Address Offset Bits. |
|---|---|---|---|
| Main memory | 0 | 0 to max installed | N/A |
| Copy mode frame buffer | 0 | 0x00100000 | A0 - A16 |
| Video RAM frame buffer | 0 | 0xFF000000 | A0 - A16 |
| Keyboard/mouse UART | 1 | 0x00000000 | A1 and A2 |
| Serial port UART | 1 | 0x00020000 | A1 and A2 |
| EEPROM | 1 | 0x00040000 | A0 - A10 |
| Time of day clock | 1 | 0x00060000 | A0 - A4 |
| Memory error registers | 1 | 0x00080000 | A0 - A2 |
| Interrupt register | 1 | 0x000A0000 | 1 byte |
| (Intel) Ethernet interface | 1 | 0x000C0000 | 1 byte |
| EPROM (bootPROM) | 1 | 0x00100000 | A0 - A15 |
| Encryption processor | 1 | 0x001C000 | A1 |
| VMEbus 32-bit addressing (16-bit data) | 2 | 0x00000000 | A0 - A31 |
| VMEbus 24-bit addressing (16-bit data) | 2 | 0xFF000000 | A0 - A23 |
| VMEbus 16-bit addressing (16-bit data) | 2 | 0xFFFF0000 | A0 - A15 |
| VMEbus 32-bit addressing (32-bit data) | 3 | 0x00000000 | A0 - A31 |
| VMEbus 24-bit addressing (32-bit data) | 3 | 0xFF000000 | A0 - A23 |
| VMEbus 16-bit addressing (32-bit data) | 3 | 0xFFFF0000 | A0 - A15 |
| Floating point accelerator | N/A | †0xE0000000 | A0 - A12 |

## 4.1. Main Memory Addressing

Memory occupies type0 physical addresses from 0x0 up to however much memory is installed. The CPU board may have either 2 Mbytes or 4 Mbytes of memory on board, and expansion cards containing either 2 Mbytes or 4 Mbytes may be added. The system supports 16 Mbytes maximum.

### Video RAM Frame Buffer

The frame buffer is a 128 Kbyte section of RAM located starting at 0xFF000000. It contains the data for the current image in the video monitor; changes to the frame buffer change the image on the screen.

### Copy Mode Frame Buffer

The copy mode frame buffer starts at 0x00100000 and contains 128 Kbytes of memory. If it is enabled in the system enable register, it contains an exact image of the contents of the main memory frame buffer; writing to this area simultaneously updates the video RAM frame buffer.

## 4.2. I/O

I/O devices include the keyboard/mouse UART, the serial port UART, the EEPROM, the time of day clock, the memory error registers, the interrupt register, the Ethernet interface, and the encryption processor. The following sections describe each.

† This is a virtual address

| Bit | Name | Type | Function/meaning |
|---|---|---|---|
| D0 | INT | Read only | Interrupt pending |
| D1 | ERR | Read only | Error pending |
| D2 | 0 | Read only | 0 |
| D3 | 0 | Read only | 0 |
| D4 | INTEN | Read/write | Interrupt enable |
| D5 | CA | Read/write | Channel attention |
| D6 | LOOPB- | Read/write | Loopback (active LOW) |
| D7 | RESET- | Read/write | Reset (active LOW) |

INT
: This bit always records the interrupt signal from the 82586 regardless of the state of the INTEN bit.

ERR
: This indicates that a bus error occurred during an 82586 channel operation, inhibiting further channel activity. To reset the ERR condition, the RESET- bit must be set (RESET- = 0).

INTEN
: This bit enables Ethernet interrupts to the CPU. An interrupt will be generated when INTEN is active if either INT or ERR goes active.

LOOPB-
: This bit controls whether the front end encoder/decoder is configured in loopback mode (LOOPB- = 0) or connected to the transceiver cable.

RESET-
: This bit initializes the 82586 when active (RESET- = 0), and allows normal operation when inactive. It also clears the ERR condition when active.

## EPROM

The EPROM consists of one 27512 type EPROM providing 64 Kbytes of PROM storage. It contains the bootstrap, selftest, and monitor code.

It is located in type1 space at 0x00100000. It is an array of read-only addresses accessed directly by virtual address bits 0 through 15 from the CPU. Even though each 8K page of data in the EPROM must be enabled with its own entry in the page map, the translated address bits (A15 through A13) are ignored, and it is addressed with bits A0 through A15 from the CPU.

The EPROM is also accessed in boot state. At this time, all supervisor program fetches are forced to fetch from the EPROM device, regardless of the MMU settings.

## Encryption Processor

The encryption processor is an AMD 8068 data ciphering processor placed in a socket to comply with US export law. It lives in type1 space at 0x001C0000, and uses bit A1 to select a data register (A1 = 0), or the address register (A1 = 1).

The encryption processor provides high-speed NBS DES encryption. To access an internal register, the address register must be written first; once the address register is setup, the selected data register can be accessed repeatedly.

### Intel Ethernet Interface

The Intel Ethernet Interface uses the Intel 82586 chip for Ethernet DVMA transfers. Used in its maximum configuration, it can access the top 16 megabytes of the current virtual address space with a supervisor data function code.

It accesses physical memory in type0 space only; bus cycles to non-existent memory or to other type spaces will not complete, and will set the ERR bit in the Ethernet control register. Protection errors and memory errors on read cycles also set the ERR bit. Once the ERR bit is set, the hardware inhibits further Ethernet transfers until the RESET- signal in the Ethernet control register is asserted.

The 82586 is connected to the system in a permanent byte-reversed mode; within each word, bits 0 through 7 are mapped to bits 8 through 15, and vice versa. This causes Ethernet data to be stored in memory in CPU byte order, while 82586 control blocks in memory are byte-swapped.

## 4.4. VMEbus

The VMEbus master interface provides access from the CPU to the VMEbus, and the VMEbus slave interface provides access from the VMEbus to the CPU for DVMA transfers.

The VMEbus master interface supports transfers of 8-bit, 16-bit, and 32-bit data, with 16-, 24-, and 32-bit addressing. The type bits in the MMU identify the number of data bits. With 32-bit addressing, the VMEbus can address 4 gigabytes minus the top 16 megabytes; with 24-bit addressing, it can address 16 megabytes minus the top 64 Kbytes. With 16-bit addressing, it can address 64 Kbytes.

The VMEbus slave interface supports both system and user DVMA into the CPU virtual address space. It supports byte, word, and longword transfers.

System DVMA responds to the lowest megabyte of the VMEbus address range in both the 24-bit and the 32-bit VMEbus address mode. The 1 megabyte space is mapped into the highest megabyte in the virtual address space of the current context.

User DVMA responds to the most significant 2 GBytes of the 32-bit VME address (A31 must be 1). VMEbus addresses A30 to A28 select one of the 8 contexts, and A2 through A27 are concatenated with DS0 and DS1 to create a 28-bit virtual address. It ignores 16-bit addresses.

Both the VMEbus and DVMA are described in the chapter *VMEbus Interface*.

## Keyboard/Mouse UART

The keyboard/mouse UART is implemented with a Zilog Z8530 SCC, which features two high-speed, fully symmetrical and highly programmable serial channels with built-in baud-rate generators. Channel A is connected to the keyboard and channel B is connected to the mouse.

The clock input is independent of the CPU clock and runs at 4.9152 MHz. It interrupts on level 6, and is described in the *Zilog 8530 SCC Data Sheet*

These ports are located in type1 space at 0x00000000. They consist of four byte-wide read/write ports addressed by bits A0, A1 and A2 (A0 is not really present, and is assumed to be a 0). Note that software must guarantee a recovery time of 1.6ms.

The address offsets decode as follows:

| Address Offset | Register |
|---|---|
| 0 | Channel B (mouse) control |
| 2 | Channel B (mouse) data |
| 4 | Channel A (keyboard) control |
| 6 | Channel A (keyboard) data |

## Serial Port

The serial port is implemented with a Zilog Z8530 SCC which features two high-speed, fully symmetrical and highly programmable serial channels with built-in baud-rate generators. Channel A is connected to UART A, and Channel B is connected to UART B. Just like the keyboard/mouse UART, the clock input is independent of the CPU clock and runs at 4.9152 MHz. It interrupts on level 6, and is described in the *Zilog 8530 SCC Data Sheet.*

The ports are located in type1 space at 0x00020000. This interface consists of 4 byte-wide read/write channels selected by decoding A0, A1 and A2 (A0 is not really present, and is assumed to be a 0). Note that software must guarantee a recovery time of 1.6ms.

The offset bits are decoded as follows:

| Address Offset | Register |
|---|---|
| 0 | Channel B control |
| 2 | Channel B data |
| 4 | Channel A control |
| 6 | Channel A data |

## EEPROM

The EEPROM consists of one 2816 EEPROM providing 2 Kbytes of electrically erasable storage. It is located in type1 space at address 0x00040000. It is an array of single-byte read/write registers addressed by bits A0 through A10.

To modify the contents of the EEPROM, each byte must be written separately, and after writing, software must guarantee at least 10ms delay before writing or reading the EEPROM again. Bytes can be read in succession with no delays using the automatic bus sizing feature of the MC68020.

The EEPROM contains system configuration information used by the EPROM during bootstrap and startup.

## Clock

The clock is an Intersil 7170 time-of-day clock with battery backup. The timer crystal oscillates at 32.768 kHz, and the clock interrupt is driven in the 100 Hz periodic mode. The clock output signal causes an interrupt request on level 5 or 7 if the respective level is enabled.

The port for the clock is in type1 space at 0x00060000. Bits A0 through A4 provide the offset, they select one of 12 clock registers if they decode to 0x0 through 0x11. These are all byte read/write registers.

Toggling the level 5 and level 7 clock enable bits in the interrupt register (EN.INT5 and EN.INT7) can cause the hardware to lose clock interrupts under certain conditions. To avoid this, change bits in the clock's command register and in the interrupt enable register as follows:

NOTE *For information about the clock command register, see the Intersil 7170 spec. The interrupt register is described later in this chapter.*

Disable all interrupts by turning OFF the EN.INT bit.

Turn OFF the EN.INT5 and EN.INT7 bits in the interrupt register to disable the clock chip from interrupting.

Disable the clock interrupt enable by writing a command to the clock command register to turn the clock's interrupt enable bit OFF.

Set the EN.INT5 and EN.INT7 bits to their new values.

Re-enable the clock interrupt enable by writing a command to the clock command register to turn the clock's interrupt enable bit back ON.

Re-enable the clock chip by turning ON EN.INT.

Also, if the clock is interrupting at level 5, and a clock interrupt arrives just as EN.INT5 is being cleared, the hardware might miss the clock interrupt edge, disabling further clock interrupts. To work around this, read the clock interrupt status and mask register immediately after clearing the EN.INT5 bit in the interrupt register. Note that this is in addition to the read of this interrupt status and mask register required before clearing and re-enabling the EN.INT5 bit in the interrupt register.

This may allow it to lose the one clock interrupt, but at least the clock will continue interrupting, allowing UNIX or some other code to periodically read the clock register and compensate for any time lost. It is possible to directly read the clock registers and perform any necessary time adjustments, or to simply adjust the time value kept in memory if the clock is found to be interrupting the second time the interrupt enable register is read.

## Memory Error Registers

The memory error registers consist of a control and an address register. When an error occurs, the control register stores information about the error, and the address register freezes the virtual address of the bus cycle which had the error.

Errors are reported with the non-maskable level 7 interrupt. When multiple errors occur, these registers latch information about the first error. The interrupt is held pending and the information in these registers is frozen until the interrupt is cleared by a write to bits 24 to 31 of the memory error address register. This can also be done by disabling the parity check bit in the memory error control register or by doing a power-on or a watchdog reset.

The memory error registers are located in type1 space at 0x00080000. The control register is a byte read/write register, and the address register is a long word (32-bit) read-only register. They are further decoded by bits A0 through A2; if these bits = 0, it accesses the control register; addresses 0x4, 0x5, 0x6 and 0x7 are bytes in the address register.

### Memory Error Address Register

The memory error address register contains the following bits:

| Bit | Name | Meaning |
|---|---|---|
| D00 - D27 | VA00 - VA27 | Virtual address |
| D28 - D30 | CX0 - CX2 | Context number |
| D31 | DVMA bit | 1 = DVMA cycle error |

### Memory Error Control Register

The memory error control register contains some read/write bits and some read only bits. It controls the following memory error functions:

| Bit | Name | Type | Meaning |
|---|---|---|---|
| D0 | Parity error 00 | Read only | Parity error, bits D00 - D07 |
| D1 | Parity error 01 | Read only | Parity error, bits D08 - D15 |
| D2 | Parity error 02 | Read only | Parity error, bits D16 - D23 |
| D3 | Parity error 03 | Read only | Parity error, bits D24 - D31 |
| D4 | Parity check | Read/write | 1 = enable parity checking |
| D5 | Parity test | Read/write | 1 = test by inverting parity |
| D6 | Parity int. enab. | Read/write | 1 = parity interrupt enable |
| D7 | Parity interrupt | Read only | 1 = parity interrupt (level 7) |

D0 — D3
These bits identify which byte had a parity error.

D4 This bit enables parity checking when it it SET.

D5 This bit is used to test the parity circuits by inverting parity. This should cause a parity error on every byte; this can be used to ensure the parity circuit is working.

D6 This bit enables parity interrupts at level 7.

D7 This bit is SET if a parity interrupt is pending.

## Interrupt Register

The interrupt register provides for the generation of software interrupts, and controls the video and clock hardware interrupts on the board. It is located in type1 space at 0x000A0000, and it is a single byte-wide read/write register. The bits have the following functions:

| Bit | Name | Type | Meaning |
|---|---|---|---|
| D0 | EN.INT | read/write | Enable all interrupts |
| D1 | EN.INT1 | read/write | Software interrupt level 1 |
| D2 | EN.INT2 | read/write | Software interrupt level 2 |
| D3 | EN.INT3 | read/write | Software interrupt level 3 |
| D4 | EN.INT4 | read/write | Enable video interrupt level 4 |
| D5 | EN.INT5 | read/write | Enable clock interrupt level 5 |
| D6 | EN.INT6 | read/write | reserved |
| D7 | EN.INT7 | read/write | Enable clock interrupt level 7 |

EN.INT
: This bit enables all interrupts including those recorded in the memory error register. If this bit is OFF, no interrupts will occur.

EN.INT1 through EN.INT3
: These bits cause software interrupts on the corresponding level. The interrupt caused by these bits stays active until software clears the corresponding bit.

EN.INT4
: This bit enables video interrupt requests on level 4. When this bit is enabled, a level 4 interrupt is set at the rising edge of vertical retrace. The level 4 interrupt request is cleared by momentarily turning this bit OFF.

EN.INT5
: This bit enables clock interrupt requests on level 5. When this bit is ON, a level 5 interrupt request is set on the rising ege of the clock interrupt output. The level 5 interrupt request is cleared by momentarily turning this bit OFF.

EN.INT6
: This bit is reserved. It can be read and written but it has no effect.

EN.INT7
: This bit enables clock interrupt requests on level 7. When this bit is ON, a level 7 interrupt request is set on the rising ege of the clock interrupt output. The level 7 interrupt request is cleared by momentarily turning this bit OFF

## Ethernet Control Register

The Sun 2060 uses an Intel 82586 for Ethernet transfers. This chip accesses memory in type0 space, and is not really an I/O device. However, its control register is in type1 space at 0x000C0000. It is a byte-wide read/write register which contains the following bits:

# 5

# Control Space Registers and Utilities

# Control Space Registers and Utilities

Control space includes the MMU, and all Sun-3 architecture extensions to the CPU. FC3 identifies a control space access, and the specific device is identified by decoding A28 through A31. Various address bits provide an index into each device's space. For the map, the context bits also provide part of the offset.

Accesses to control space use the MOVES command, which causes the 68020 to look in the function code registers to select a function code. In order for control space accesses to work, the function code registers must contain the value 0x3.

All control space devices are byte read and write except the bus error register which is read only, and the diagnostic register which is write only. The IDPROM, page map, and segment map are implemented as arrays of bytes; the MC68020 uses its dynamic bus sizing capability to perform word and longword accesses.

The control space contains the following devices:

Table 5-1 *Control Space Devices*

| Name | A28-31 | Size | Read/Write | Offset bits |
|---|---|---|---|---|
| IDPROM | 0x0 | Byte | Read | A0 - A4 |
| Page map | 0x1 | long | R/W | Context + A13 - A27 |
| Segment map | 0x2 | byte | R/W | Context + A17-A27 |
| Context reg | 0x3 | byte | R/W | |
| System enable reg | 0x4 | byte | R/W | |
| User DVMA enable | 0x5 | byte | R/W | |
| Bus error reg | 0x6 | byte | read | |
| Diagnostic reg | 0x7 | byte | write | |
| UART bypass for serial port | 0xF | byte | R/W | A1-A2 |

The following sections describe the format of each.

## 5.1. IDPROM

The 32-byte IDPROM field contains confidential information about the system.

## Page Map

The page map is a 4K by 32-bit RAM bank. It contains the 32-bit page map entries which provide the physical page component of physical addresses and a number of information bits about that page. Control accesses to the page map provide direct access to the map RAMs, allowing you to actually read or write data in the map.

The map RAMs receive their address input from the segment map, and address bits A13 through A16. The 8 segment map outputs provide a page map entry group (PMEG) number, and the 4 address lines select one of 16 addresses within that PMEG. Each of these addresses one particular map location.

To access the map RAM directly, you must first set up the segment map. Since the segment map outputs a PMEG number, you should load the value of the PMEG you wish to access into the segment map, then use address lines A13 through A16 to identify an entry in that PMEG.

## 5.3. Segment Map

The segment map is a 16K by 8 bit RAM bank which takes as input the context bits and bits 17 through 27 of the virtual address, and outputs a PMEG number, which forms part of the page map RAM address. Control accesses to the segment map provide direct access to the segment map RAMs, allowing you to actually read or write data in the segment map.

During control space accesses to the segment map RAM, address bits A17 through A27 and the context bits select the byte address within the segment map RAM.

## 5.4. Context Register

The MMU is divided into 8 address spaces called contexts. These are selected by bits 0 through 2 of the context register. These bits get concatenated to the beginning of every virtual address to enter the MMU, thus providing the high order address bits.

The context register is byte-wide, but only bits 0 through 2 are defined.

## 5.5. System Enable Register

The system enable register enables various system facilities. It can be read and written, and is cleared on hardware reset and watchdog reset, but not CPU reset. The bits are assigned as follows:

| Bit | Name | Function |
|---|---|---|
| 0 | EN.DIAG | Read back diagnostic switch |
| 1 | EN.FPA | Enable floating point accelerator |
| 2 | EN.COPY | Enable video copy mode |
| 3 | EN.VIDEO | Enable video display |
| 4 | EN.CACHE | Not used |
| 5 | EN.SDVMA | Enable DVMA |
| 6 | EN.FPP | Enable floating point processor |
| 7 | EN.BOOT- | Enable boot state (0=boot; 1=normal) |

These bits have the following meaning:

EN.DIAG
: This bit reports the position of the external diagnostic switch; 0 means the switch is in the normal (non-diagnostic) position, and 1 means the switch is in the diagnostic position.

EN.FPA
: This bit enables the Floating Point Accelerator (FPA), if present. It this bit is deasserted, accesses to the FPA cause a bus error and set the FPAENERROR bit in the bus error register. If this bit is enabled, accesses to the FPA are directed to the FPA, but if no FPA is present, the access will timeout and a TIMEOUT bus error will be recorded in the bus error register.

EN.COPY
: This bit enables the copy block, a 128 Kbyte portion of memory that maintains a copy of the video frame buffer.

EN.VIDEO
: This bit enables the video signal to the video monitor.

EN.CACHE
: Not used.

EN.SDVMA
: This bit enables DVMA from the VMEbus.

EN.FPP
: This bit enables the floating point coprocessor (FPP) if it is present.

EN.BOOT
: When boot is enabled, all supervisor program fetches are to the EPROM device regardless of the setting of the MMU. All other types of references are unaffected.

## 5.6. User DVMA Enable Register

The user DVMA enable register is a byte wide read/write register located at 0x50000000 in control space. It controls which contexts have access to user DVMA. Each bit enables one of the contexts when it is ON. The bits are:

| Bit | Name | Meaning |
|---|---|---|
| D0 | EN.CX0 | Enable user DVMA to context 0 |
| D1 | EN.CX1 | Enable user DVMA to context 1 |
| D2 | EN.CX2 | Enable user DVMA to context 2 |
| D3 | EN.CX3 | Enable user DVMA to context 3 |
| D4 | EN.CX4 | Enable user DVMA to context 4 |
| D5 | EN.CX5 | Enable user DVMA to context 5 |
| D6 | EN.CX6 | Enable user DVMA to context 6 |
| D7 | EN.CX7 | Enable user DVMA to context 7 |

When this register is cleared after reset, all bits are set to 0 (disabled).

## Bus Error Register

The bus error register is a byte-wide read-only register located at 0x60000000 in control space. It captures the status of any bus error that occurs during a CPU bus cycle to control space or device space if the cause is identified synchronously with the bus cycle. The cycle terminates with a bus error signal, and one of the following bits is turned ON:

| Bit | Name | Meaning |
|---|---|---|
| D0 | Unused | |
| D1 | Unused | |
| D2 | FPAENERR | FPA enable error |
| D3 | FPABERR | FPA bus error |
| D4 | VMEBERR | VMEbus bus error |
| D5 | TIMEOUT | Timeout error |
| D6 | PROTERR | Protection error |
| D7 | INVALID | Invalid page access |

NOTE *When these bits are set (HIGH) the condition is TRUE. Normally, they should be LOW.*

These bits are set by the following conditions:

WATCHDOG is set when a watchdog reset is detected, or when the user reset switch is activated.

FPAENRR is set whenever the CPU attempts to access the FPA while its enable bit is reset (EN.FPA in system enable register is 0).

FPABERR is set whenever the FPA signals an error during a CPU bus access. The FPA has no interrupt capability; it reports all errors as bus errors.

VMEBERR is set when a CPU cycle to the VMEbus was acknowledged with a VMEbus bus error.

TIMEOUT is set when the CPU tries to access a non-existent device, either on-board or off-board, during a cycle which may record bus errors. Details on the various types of timeouts are provided below.

PROTERR signals a protection violation resulting from an attempted access to a device space page during a CPU bus cycle. This error may be caused by an attempt to write to a valid page whose write access bit is reset, or an attempt by an access with only user permissions to access a page that requires supervisor permission. The MMU detects these errors during address translation.

INVALID indicates that the valid bit in the page map was not set during a CPU bus cycle to a device space page.

### Timeout Errors

The following events may cause timeout errors:

Accesses to a non-existent control space device, or to a non-existent offset within a control space device.

Accesses to device space where virtual address bits A28 through A31 = 0x1 through 0xD.

Device space accesses to main memory or a frame buffer which is addressable but not present.

Accesses to I/O devices which are optional but not present (fail to respond within 1 to 2 refresh periods).

Accesses to the VMEbus master interface (type2 or type3 device space) where the addressed VMEbus device fails to respond within 737 microseconds.

## 5.8. Diagnostic Register

The diagnostic register is a byte-wide write only register located at 0x70000000 in control space. It drives the 8-bit LED display such that a 0 bit causes the corresponding LED to light and a 1 bit causes it to remain dark.

## 5.9. UART Bypass

The UART bypass is located at 0xF0000000 in control space. It enables accesses to the serial port A and B UARTs without using the MMU. The port and mode selection works the same as the normal accesses described in the chapter *Device Space.*

6

# VMEbus Interface

# VMEbus Interface

The 2060 board interfaces to other boards over a standard Revision B VMEbus. Any board designed to meet this VMEbus specification should be completely compatible.

NOTE *The 2060 is actually* `Revision C VMEbus` *compatible with only two exceptions: 1) It doesn't support 3-byte transfers, and 2) longword accesses must be longword aligned.*

The VMEbus provides both a master and a slave interface. The VMEbus master provides access from the CPU to the VMEbus, and the VMEbus slave interface provides access from the VMEbus to the P2 bus (local memory or type0 space) for DVMA transfers. The master interface uses physical addresses, and the slave interface uses virtual addresses.

It provides the following capabilities:

Table 6-1 *System VMEbus Capabilities*

| NAME | TITLE | CAPABILITY |
|---|---|---|
| | MASTER CAPABILITIES | |
| Data bus size | D32 MASTER | 32, 16, or 8 bit data. |
| Address bus size | A32 MASTER (DYN) | 32, 24, 16 bit addressing |
| Timeout option | TOUT(737) | 737 microsecond timeout period |
| Sequential access | none | |
| Interrupt handler | IH(1-7)STAT | Level 1-7 jumperable. All interrupts use vectors provided by VMEbus interruptors per VMEbus spec. |
| Requester option | ROR R(3) | Release on request, level 3 |
| Bus busy option | | Releases BBSY after AS assertion when releasing bus |
| Read/modify/write | | Will not release VMEbus during read/modify/write cycles |
| | SLAVE CAPABILITIES | |

Table 6-1 *System VMEbus Capabilities— Continued*

| NAME | TITLE | CAPABILITY |
|---|---|---|
| Data bus size | D32 SLAVE (DYN) | 32, 16, or 8 bit data |
| Address bus size | A32 SLAVE (DYN) | 32 and 24 bit addresses |
| Sequential Access | none | |
| Special access mode | | A high-speed access mode (lock mode) is engaged if the time from DTACK assertion to the next AS and DS assertion is less than 200ns. |
| Interrupter options | none | |
| 32-bit slave addressing | | Responds to the bottom 1 Mb by performing DVMA using supervisor function codes, and responds to the top 2 Gb by performing DVMA using user function codes. Response can be dynamically disabled on 256 Mb (context) boundaries. |
| 24-bit slave addressing | | Responds to the bottom 1 Mb by performing DVMA using supervisor function codes. |
| | SYSTEM CONTROLLER | |
| Clock option | SYSCLK 16 MHz | Jumperable (not used on board) |
| Arbiter option | ONE | Bus request/grant level 3 only, or Ethernet arbiter |
| Bus timeout module | | none |
| Sysreset option | | SYSRESET MASTER or SYSRESET SLAVE including manual button |
| Sysfail option | | Not monitored |
| Acfail option | | Non implemented. ACFAIL is connected to SYSRESET |

## 6.1. Master Addressing

The VMEbus master provides access from the CPU to the VMEbus.

When physical addresses come out of the MMU, the type bits identify whether they are memory, I/O, or VMEbus master accesses. If they are master accesses, these bits further identify if the access is to 16-bit data space or 32-bit data space. These bits effectively divide the physical address space into 4 gigabytes; one for memory, one for I/O devices, one for VMEbus 16-bit data and one for VMEbus 32-bit data. The actual bit assignments appear in the chapter *Device Space.*

In order for accesses to VMEbus 32-bit space to be performed as VMEbus longword accesses, the address must be longword aligned and the 68020 size bits must indicate that the cycle is longword. Otherwise the cycle will be broken down into the appropriate number of byte and word accesses, using the 68020 dynamic bus sizing feature. Note that the VME Revision C specification allows for non-aligned word, 3-byte, and longword transfers, but the 2060 CPU board does not use this feature.

With 32-bit addressing, the VMEbus can address 4 gigabytes minus the top 16 megabytes; with 24-bit addressing, it can address 16 megabytes minus the top 64 Kbytes. With 16-bit addressing, it can address 64 Kbytes. The number of VMEbus address bits is selected by the virtual address; the appropriate physical bits then become the offset within this address. The following table shows the virtual address to physical address decodings:

| Type | Physical base addr | VMEbus Addr Size | Physical Address Bits |
|---|---|---|---|
| 2 | 0x00000000 | 32-bit | A0 - A32 |
| 2 | 0xFF000000 | 24-bit | A0 - A23 |
| 2 | 0xFFFF0000 | 16-bit | A0 - A15 |

The mapping for these spaces appears in Figure 6-1.

## Long Master Cycles

If the response time for a slave is longer than 2.88 microseconds, the CPU services pending refresh or Ethernet cycles, and checks periodically to see if the slow device has finally responded. This allows the CPU to satisfy the VMEbus requirement for no limit to response time, while also servicing real-time needs elsewhere in the system.

## Address Modifiers

The VMEbus obtains information about the cycles from the VMEbus address modifier bits. Note that MC68020 function code bits translate directly to VMEbus address modifier bits AM0 through AM2.

The following table shows how it interprets these bits (note that it assumes two leading 0's):

Table 6-2 *Address Modifier Bits*

| Address Modifiers 5 4 3 2 1 0 | Hex Code | Function |
|---|---|---|
| 0 0 1 0 0 1 | 0x09 | 32 bit addressing, user data space |
| 0 0 1 0 1 0 | 0x0a | 32 bit addressing, user program space |
| 0 0 1 1 0 1 | 0x0d | 32 bit addressing, supervisor data space |
| 0 0 1 1 1 0 | 0x0e | 32 bit addressing, supervisor program space |
| 1 0 1 0 0 1 | 0x29 | 16 bit addressing, user data space |
| 1 0 1 0 1 0 | 0x2a | 16 bit addressing, user program space |
| 1 0 1 1 0 1 | 0x2d | 16 bit addressing, supervisor data space |
| 1 0 1 1 1 0 | 0x2e | 16 bit addressing, supervisor program space |
| 1 1 1 0 0 1 | 0x39 | 24 bit addressing, user data space |
| 1 1 1 0 1 0 | 0x3a | 24 bit addressing, user program space |
| 1 1 1 1 0 1 | 0x0d | 24 bit addressing, supervisor data space |
| 1 1 1 1 1 0 | 0x0e | 24 bit addressing, supervisor program space |

## Slave Addressing

The VMEbus slave interface supports user and supervisor DVMA into the CPU virtual address space from the VMEbus. This allows DMA devices to use virtual addresses instead of the usual physical addresses, saving software and hardware the overhead of translating the addresses separately for the DMA devices.

The VMEbus slave addressing appears in Figure 6-2.

This interface supports byte, word, and longword transfers. Both supervisor and user DVMA are defined entirely by the VMEbus 24- or 32-bit address, and address modifiers AM4 and AM5. AM0 through AM3 are ignored.

Accesses to non-existing memory or other (non type0) devices results in a VMEbus error. This also occurs if the DVMA cycle encounters a page fault, a protection error, or a parity error on a read cycle. If parity errors occur, they are reported to the CPU as interrupts, if enabled.

## DVMA

Direct Virtual Memory Access (DVMA) allows devices other than the CPU to access the system main memory and video memory using virtual addresses. This means the addresses provided by a DVMA device are translated by the MMU, and the software does not have to perform its own address translation. During DVMA cycles, the DVMA literally replaces the CPU, generating all the same signals, with similar timing.

## Supervisor DVMA

Supervisor DVMA allows VMEbus devices to access the entire 256 megabyte virtual address space of the CPU in 8 different contexts. This allows programs running on the CPU to share pointers to data at arbitrary locations with coprocessors located on the VMEbus; this simplifies sharing data between two processors.

Supervisor DVMA responds to the lowest megabyte of the VMEbus address range in both the 24-bit and the 32-bit VMEbus address mode. The 1 megabyte space is mapped into the highest megabyte in the virtual address space of the current context. The address mapping is:

```
VMEbus address A24/A32 to A0     Virtual Address

0x00000000  to 0x000FFFFF    0x0FF00000 to 0x0FFFFFFF
```

Supervisor DVMA is enabled with the EN.SDVMA bit in the system enable register. These cycles have supervisor access for protection checking, and a VMEbus bus error is signaled if the page being accessed is not valid or if the access has a protection violation. This error is not visible to the CPU.

The 2060 CPU makes no distinction between 24 bit data supervisor DVMA and 32 bit supervisor DVMA.

## User DVMA

The 2060 responds to VMEbus addresses in the top 2 gigabytes of 32-bit VMEbus space with user DVMA mode. To be in this range, VMEbus address bit 31 must be HIGH. User DVMA is performed with user function codes, so it is fully protected by the MMU. VMEbus address bits 28 through 30 correspond directly to the context bits, so that context 0 is accessed when these address bits are LOW.

Contexts are individually enabled by the bits in the user DVMA enable register. If a VMEbus address accesses a non-enabled context, the 2060 board will not respond. This allows up to 8 2060 boards to share the same backplane, as long as a different context is enabled on each board.

User DVMA cycles use user access for protection checking; a VMEbus bus error is signaled if the page being accessed is not valid or if the access has a protection violation. This bus error is not visible to the CPU; VMEbus masters that expect virtual I/O support from the CPU must then post an interrupt to the CPU and must make the appropriate information about the bus error available to the CPU.

The VMEbus address to virtual address mapping is as follows:

| VMEbus Address | Context | Virtual Address. |
|---|---|---|
| 0x80000000 to 0x8FFFFFFF | 0 | 0x00000000 to 0x0FFFFFFF |
| 0x90000000 to 0x9FFFFFFF | 1 | 0x00000000 to 0x0FFFFFFF |
| 0xA0000000 to 0xAFFFFFFF | 2 | 0x00000000 to 0x0FFFFFFF |
| 0xB0000000 to 0xBFFFFFFF | 3 | 0x00000000 to 0x0FFFFFFF |
| 0xC0000000 to 0xCFFFFFFF | 4 | 0x00000000 to 0x0FFFFFFF |
| 0xD0000000 to 0xDFFFFFFF | 5 | 0x00000000 to 0x0FFFFFFF |
| 0xE0000000 to 0xEFFFFFFF | 6 | 0x00000000 to 0x0FFFFFFF |
| 0xF0000000 to 0xFFFFFFFF | 7 | 0x00000000 to 0x0FFFFFFF |

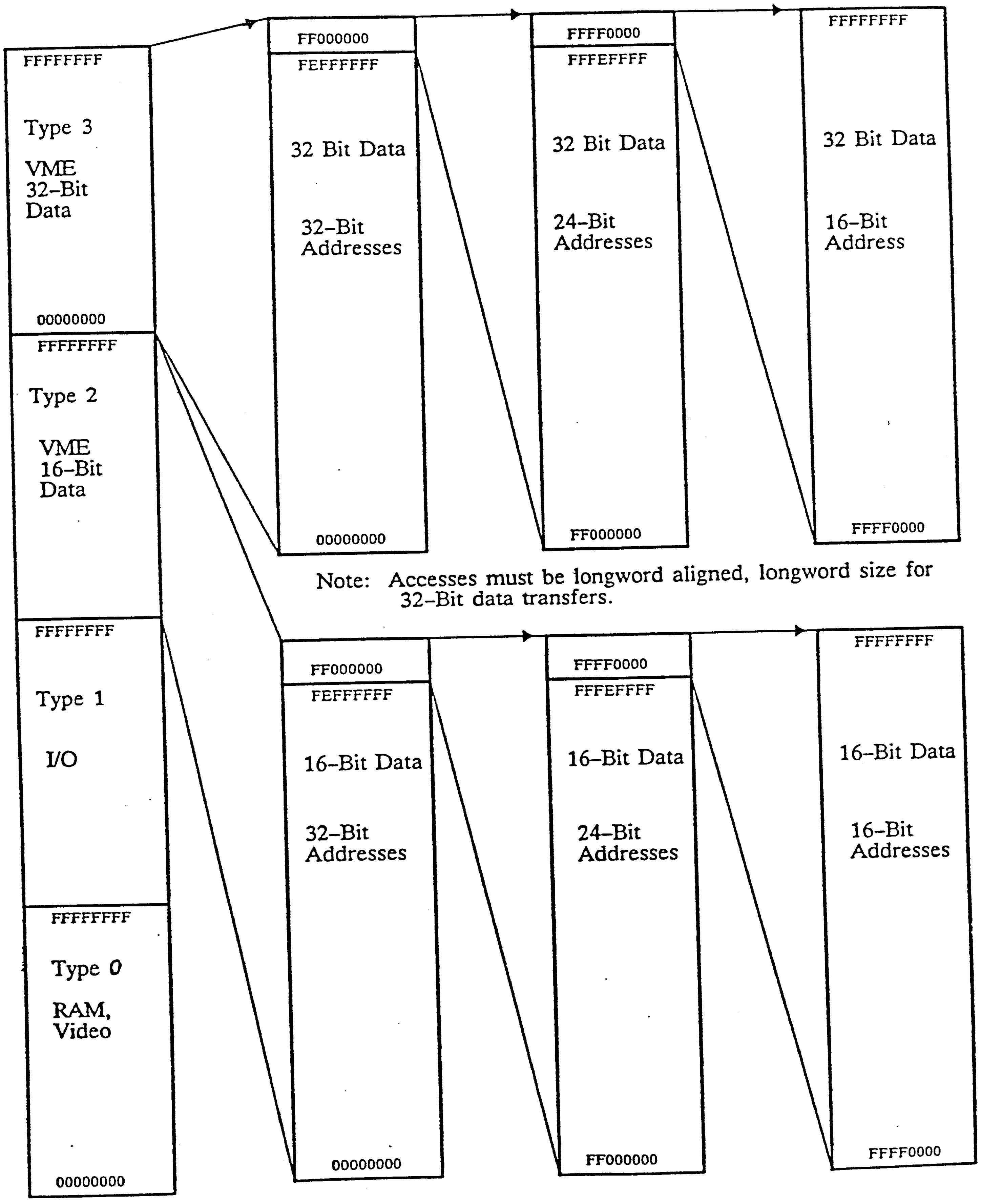
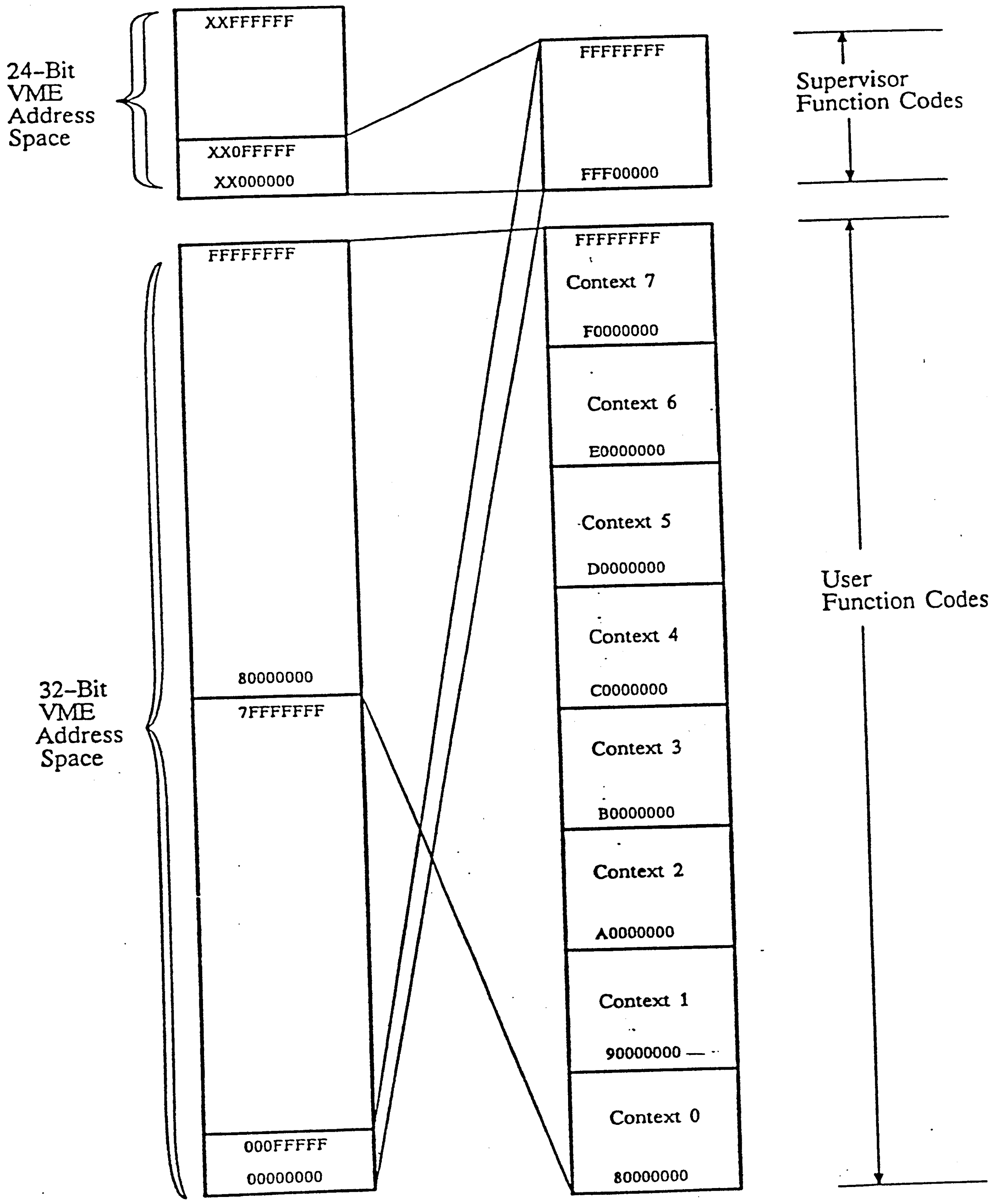


Figure 6-1 *VME Master Address Mapping*

Figure 6-2 *VME Slave Address Mapping*



A

# Additional VMEbus Information

# Additional VMEbus Information

This appendix provides additional information about the 2060 VMEbus interface. It lists additional specifications, including timing, and it describes the timing characteristics.

## A.1. Operating Requirements

Environmental Characteristics

| | |
|---|---|
| Operating Temperature: | 10-40 C |
| Humidity: | 5-90% non-condensing |

Power Characteristics

| | |
|---|---|
| +5 Volts: | 14 Amp Max. |
| -5 Volts: | 1 Amp Max. |
| +12 Volts: | 0.5 Amp Max. |
| -12 Volts: | Not used |

## A.2. Typical Performance Parameters

Table A-1 *Typical Performance Parameters*

| Parameter | Sun-2 | 2060 (Sun/3) measured | See Note: |
|---|---|---|---|
| CPU to VME Latency 1 | 468ns | 381ns | 1 |
| CPU to VME Latency 2 | 264 ns | 186 ns | 2 |
| CPU to VME Bandwidth | 3.3 MB/sec | 9.5 MB/sec | 3 |
| | 606 ns | 420 ns | |
| VME to P2 Latency | 952 ns | 743 ns | 4 |
| VME to P2 Bandwidth | 1.97 MB/sec | 7.84 MB/sec | 5 |
| | 1020 ns | 510 ns | |
| Time to Acquire VME | 162-264 ns | 141-191 ns | 6 |
| CPU-to-CPU Bandwidth | ...... | 3.44 MB/sec | 7 |
| | ...... | 1163 ns | |
| Unix Throughput | 310 KB/sec | 704 KB/sec | 8 |
| Astraea Bd. Throughput | ...... | 2.8 MB/sec | 9 |
| Xylogics Throughput | ...... | 1.41 MB/sec | 10 |
| | ...... | 1420 ns | |
| GP Cycle Time | ...... | 1200 ns | 11 |
| Lock Mode Throughput | ...... | 600 ns | 12 |
| | ...... | 6.67 MB/sec | |
| Lock Mode Latency | ...... | 440 ns | 13 |

Notes ----------

1 Measured from processor address strobe to VME dtack with an ideal VME device, with the CPU not currently bus master. Unless otherwise noted, measurements are the average of 10 samples on a logic analyzer with a 10 ns resolution. Sun-2 numbers are estimated for a Model 50.

2 Measured same as 1 above, but CPU is currently bus master.

3 Assumes an ideal 32-bit-data VME device. Sun-2 numbers are estimated for a Model 50.

4 Measured from VME address strobe to VME dtack. Sun-2 numbers are estimated for a Model 50.

5 Assumes P2 bus is locked, allows 65 ns negation period on VME address strobe. Sun-2 numbers are estimated for a Model 50. The 2060 measured number is extrapolated from actual measured data, as we currently have no VMEbus masters this fast.

6 Measured from assertion of VME bus request to assertion of VME bus grant. Sun-2 numbers are estimated for a Model 50.

7 One 2060 is not fast enough to engage lock mode on a second 2060. The estimated number is derived from analysis of the schematic.

8 Command used is cp filename /dev/null, file size is 10 MB. Both disks were Fujitsu Eagles; disk controllers were Xylogics, with the 2060 using a VME-Multibus adapter. The Sun-2 was a Sun-2/120.

9 The Astraea board is a 256 KB memory board with a response time from VME address strobe to VME dtack of 279 ns. Test was a hand assembled tight loop writing over the VMEbus to the Astraea board. The estimated number is from analysis of the schematic, combined with the information about the response time of the Astraea board.

10 Estimated from observation of oscilloscope traces. There were four distinct cycle times: 1300, 1400, 1500, and 1600 ns. Transfers were from disk into on-board memory.

11 Graphics Processor as Master, 2060 as Slave

12 Using Graphics Processor modified to automatically start a cycle 60 ns after ending previous cycle.

13 Measured from VME address strobe to dtack.

## A.3. Latency Considerations

In this case, latency is the time elapsed between when an external VME master requests a cycle and when the 2060 board responds to that cycle. This latency can be divided into four separate periods: 1) the time required to acquire control of the VMEbus from the current master, 2) the time required to acquire control of the local bus from the CPU, 3) the time required to perform any pending DMA of higher priority, and 4) the time required to actually perform the cycle.

1) The VMEbus has no specification for the amount of time that a master may keep control of the bus after another master has issued a bus request, but the 2060 releases the bus as soon as it completes the cycle that may be in progress when the bus request arrives. The worst-case situation occurs when the bus request arrives just after state 3 of a CPU access of the VMEbus, because at that point it is committed to performing a VME cycle. At state 5 it asserts VME address strobe, at state 7 it synchronizes it, and at state 9 it asserts VME bus grant, for a total worst-case elapsed time of 232 nanoseconds. Masters other than the 2060 board may keep control of the bus for an arbitrary length of time. One example is the Xylogics disk controller board, which has been observed to keep control of the bus for over 45 microseconds after bus request has been asserted.

NOTE *States are half-clock periods (30 nanoseconds) starting with state 0 at the beginning of a processor bus cycle.*

The external master must now wait until the VME address strobe is negated before it can take control of the VMEbus, which can only happen after the currently addressed slave responds with VME DTACK. The VMEbus has no specified maximum response time either, so this can be an indeterminate period. In a Sun system the worst case response time is that of the Xylogics disk controller board, which can take up to 70 microseconds.

Once the VME address strobe goes away the external master can take control of the VMEbus, enable its addresses and data onto the bus, and assert its own address and data strobes. These addresses and strobes are decoded on the 2060 board to form an on-board request called XREQ.

2) The worst-case time to acquire the local bus occurs if XREQ arrives just as the CPU starts a cycle to a slow on-board device such as an interrupt acknowledge to the vectored serial ports, which can take up to 1170 nanoseconds. This time will add to the time required to decode the VMEbus addresses and generate XREQ, which is 180 nanoseconds if the VME address strobe exactly misses a CPU synchronization clock, for a total of 1350 ns.

3) If the Ethernet interface sends a DMA request during that 1170 nanoseconds when it was waiting for the local bus, its request is serviced first. This requires approximately 500 nanoseconds to complete, during which time a refresh request might be received. This requires another 300 nanoseconds, which may give the Ethernet interface enough time to request another DMA cycle, for another 500 nanosecond delay. Now, the VME slave cycle is assured of service. Thus, the total worst-case wait to acquire the local bus is 2.65 microseconds.

4) The final component of the 2060 slave response time is the actual time needed to perform the memory cycle and generate a VME DTACK, which in the case of a main memory cycle takes 330 nanoseconds. Therefore the total time, under absolute worst-case conditions, to acquire the VMEbus and perform a memory cycle on the 2060 board is 3.21 microseconds, not counting the time required for a board that may currently be master to finish its cycles and relinquish the VMEbus.

## Throughput Considerations

The following paragraphs describe the factors that effect 2060 throughput. These factors are lock mode, throughput lost to Ethernet, throughput lost to CPU and refresh, and throughput lost to turnaround times.

### Lock Mode

Lock Mode locks out the CPU from using its local bus, and is engaged when the turnaround time of the external VME master is less than 200 nanoseconds from when the 2060 board asserts VME DTACK to when the external master asserts VME address and data strobes for the next cycle. As long as this situation occurs at the end of each slave cycle, the CPU local bus will remain locked for another cycle. During Lock Mode, refresh and Ethernet DMA cycles continue to occur, as they have a higher priority than pending VME slave cycles. In addition, the CPU will be allowed to perform one bus cycle after each refresh if it has a need to do so, so that it can continue to limp along even if the Lock circuitry becomes defective or if an external master keeps Lock Mode engaged for a long time.

If a VME master device is designed specifically to use Lock Mode, it is preferable that it limit Lock Mode transfers to 16 in a row, then back off long enough to allow the CPU to perform a cycle. This is not a requirement, however, as generic boards might be able to engage Lock Mode and we can't force them to be redesigned. The 200 nanosecond figure is also a worst-case figure: if the turnaround time is less than 200 nanoseconds, Lock Mode is guaranteed to be engaged. If the turnaround time is between 200 and 240 nanoseconds, Lock Mode might be engaged.

### Throughput Lost to Ethernet

The throughput number given above assumes no Ethernet activity. If the Ethernet interface is attempting to perform DMA cycles at the same time as the VME slave interface, the VME slave interface will slow down. The Ethernet operates at 10 megabits/second, or 1.25 megabytes/second, and takes an additional 10% for overhead such as fetching command blocks, buffer descriptors, and so on, for a total bandwidth requirement of 1.37 mb/sec.

This will subtract directly from the 7.84 mb/sec figure provided above, giving 6.46 mb/sec through the VME slave interface when ethernet is active.

### Throughput Lost to CPU and Refresh

A refresh cycle is performed every 15 microseconds, taking approximately 300 nanoseconds away from the time available for the VME slave interface. This represents a 2% overhead, corresponding to a loss of 0.16 mb/sec. In addition, as mentioned above, the CPU is allowed to perform one bus cycle after every refresh cycle. If it takes advantage of this opportunity, the cycle will take 270 nanoseconds, and four clocks will be wasted in turning mastership of the bus over to the CPU and regaining it after the cycle, for a total loss of 510 nanoseconds. This represents 3.4% overhead which adds 2% to the time taken for the actual refresh cycle, giving overhead of 5.4% or 0.42 mb/sec. To a first approximation this loss is additive to the Ethernet loss described in the previous section, so that the worst-case throughput of the VME slave interface over a long period with Ethernet active is 6.04 mb/sec.

### Throughput Lost to Turnaround Time

Note 5 in the performance parameter section above states that the master is allowed a 65 nanosecond negation period on VME address strobe. If the negation period is longer, performance will suffer in steps of 60 nanoseconds per cycle, up to the point where the turnaround time is so slow that Lock Mode is no longer engaged. At that point, the CPU will start performing a memory cycle after each VME slave cycle, so two kinds of overhead will be incurred: that due to the actual time taken to perform the CPU cycle, and that due to time wasted transferring the local bus back and forth between masters. This overhead amounts to 6 1/2 clocks, lowering the throughput from 7.84 mb/sec to 4.17 mb/sec.

## A.5. Timing

The 2060 implements the full VME Specification, Rev. B, with no exceptions. Therefore a board designed to meet the VME spec should plug right in. Nevertheless, in this section we provide minimum/maximum timings for all signals on the VMEbus in master and slave mode.

NOTE *The 2060 is actually* `Revision C VMEbus` *compatible with only two exceptions: 1) It doesn't support 3-byte transfers, and 2) longword accesses must be longword aligned.*

Table A-2 *VMEbus Master Timing*

| VME Spec # | Description | min | max | VME spec |
|---|---|---|---|---|
| 1R | Axx and AMx valid to AS* low | 388 | 238 ns | |
| 2R | DTACK low to invalid address | 137 | ?? | 0 |
| 3R | AS* high | 188 | ?? | 40 |
| 4R | DTACK low to AS* high (If DTACK arrives within 2.88 microseconds; otherwise, max=2560) | 137 | 294 | 0 |
| 5R | AS* to DS"A" skew | 7 | 30 | 0 |
| 6R | WRITE* valid to DS"A" low | 38 | 82 | 35 |
| 7R | DS"B" high to invalid WRITE* | 10 | ?? | 10 |
| 8R | DATA release to DS"A" low | 150 | ?? | 0 |
| 9R | DS"A" to DS"B" skew | 0 | 8 | 10 |
| 9W | Dxx valid to DS"A" low | 38 | 97 | 35 |
| 10R | DTACK* low to DS"A" high (see note 4R above) | 137 | 244 | 0 |
| 10W | DTACK* low to invalid data | 137 | 227 | 0 |
| 11R | DS"A" high | 168 | ?? | 40 |
| 12R | DS"B" to DS"A" low | 168 | ?? | 40 |
| 13R | DTACK*/BERR* high to DS"A" low | 15 | ?? | 0 |
| 14R | DTACK* low to DS"B" high (see note 4R above) | 137 | 244 | 0 |
| 16R | DS"B" high | 168 | ?? | 40 |

Table A-3 *VMEbus Slave Timing*

| VME Spec # | Description | min | max | VME spec |
|---|---|---|---|---|
| 15W | DS"A" low to DTACK*/BERR* low | 352 | 2980 | 30 |
| 16R | Data valid to DTACK* low | 65 | 125 | 0 |
| 18R | DS"A" high to invalid data | 17 | 67 | 0 |
| 20R | DS"B" high to DTACK*/BERR* high | 12 | 50 | 0 |

# B

# Jumpers

# Jumpers

This appendix shows the shows the 2060 jumpers:

Table B-1 *2060 Board Jumpers*

| *Jumper Block No.* | *Pins* | *Location* | *Jumper In?* | *Purpose* |
|---|---|---|---|---|
| J2502 | 1-2 | A-3 | Yes | Enable VME Clock |
| J1200 | 1-2 | A-3 | No | For 27256 Boot PROM |
| J1201 | 1-2 | A-3 | Yes | For 27512 Boot PROM |
| J2501 | 1-2 | B-6.5 | Yes | Enable Ethernet Clock |
| J2301 | 1-2 | B-21 | Yes | Enable Video Clock |
| J1001 | 1-2 | E-32 | Yes | Enable SCC Clock |
| J3102 | 1-2 | K-11 | Yes, for ⇒ | 4 Mbyte CPU board |
| J3101 | 1-2 | K-11 | Yes, for ⇒ | 2 Mbyte CPU board |
| J100 | 1-2 | K-11 | No | Cache Disable |
| J2503 | 1-2 | K-11 | †No, for ⇒ | Level 2 Ethernet |
| J400 | 1-2 | N-11 | Yes | Select 16.67MHz CPU Clock |
| J400 | 3-4 | N-11 | No | Select 12.5MHz CPU Clock |
| J400 | 5-6 | N-11 | Yes | Select Asynch. 12.5MHz FPP Clock |
| J400 | 7-8 | N-11 | No | Select Synch. 16.67MHz FPP Clock |
| J300 | 1-2 | R-5 | No | Null |
| J300 | 3-4 | R-5 | Yes | VME Interrupt Level 1 |
| J300 | 5-6 | R-5 | Yes | VME Interrupt Level 2 |
| J300 | 7-8 | R-5 | Yes | VME Interrupt Level 3 |
| J300 | 9-10 | R-5 | Yes | VME Interrupt Level 4 |
| J300 | 11-12 | R-5 | Yes | VME Interrupt Level 5 |
| J300 | 13-14 | R-5 | Yes | VME Interrupt Level 6 |
| J300 | 15-16 | R-5 | Yes | VME Interrupt Level 7 |
| J2703 | 1-2 | R-12 | Yes | CPU is VME Reset Master |
| J2702 | 1-2 | R-12 | No | CPU is VME Reset Slave |
| J2701 | 1-2 | R-12 | Yes | CPU is VME Arbiter & Requester |
| J2700 | 1-2 | R-12 | No | CPU is VME Requester Only |

† This jumper is IN for Level 1 Ethernet

# Index

# Revision History

| Revision | Date | Comments |
|---|---|---|
| 01 | 12 15, 1985 | Alpha draft |
| 50 | January 30, 1986 | Completed manual — Incorporated comments from Alpha. |
|  |  |  |