Data General

Customer Documentation

AViiON™ 300 and 400 Series Stations: Programming System Control and I/O Registers

AViiON™
PRODUCT LINE

AViiON™ 300 and 400 Series Stations: Programming System Control and I/O Registers

014-001800-05

Ordering No. 014-001800

Printed in the United States of America
Rev. 05, June 1990

Notice

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

AViiON™ 300 and 400 Series Stations:
Programming System Control and I/O Registers
014-001800-05
014-001842-05 (Japan only)

Revision History:

Original Release	- April 1989
First Revision	- June 1989
Second Revision	- July 1989
Third Revision	- September 1989
Fourth Revision	- March 1990
Fifth Revision	- June 1990

A vertical bar in the margin of a page indicates substantive technical change from the previous revision.

Preface

This manual describes the AViiON™ 300 and 400 series station architecture and their system control and input/output (I/O) registers. This manual is written for systems and applications designers familiar with assembly language and C programming and conventions. The manual does not assume that you are familiar with a particular operating system or have specific knowledge about AViiON products.

This manual contains the following chapters:

Chapter 1 System Board Architecture

This chapter describes the system board architecture, including the CPU, main memory, system buses, system control logic, graphics subsystems, and integrated I/O subsystem.

Chapter 2 Programming the System Board

This chapter describes address mapping, bus arbitration, how to address the system board resources, and how to program the time–of–boot clock, nonvolatile RAM, programmable interval timer and time–of–day clock. The chapter concludes with descriptions of the boot PROM and the System Control Monitor (SCM).

Chapter 3 Interrupts, System Errors and Bus Faults

This chapter describes interrupts, system errors and bus faults; where they originate, how the system board handles them, and what registers are involved in interrupts.

Chapter 4 Programming the Monochrome Graphics Subsystem

This chapter describes the 300 series station monochrome graphics subsystem and how to program the subsystem.

Chapter 5 Programming the Color Graphics Subsystem

This chapter describes the color graphics subsystem and how to program the subsystem. It also describes the optional Z–buffer gate array and its registers.

Chapter 6 Programming the Keyboard Interface

This chapter describes the keyboard port and how to program it.

Chapter 7 Programming the Serial Ports and Parallel Port

This chapter describes the serial ports and parallel ports, and how to program the ports. The serial ports include the mouse port.

Chapter 8 Programming the Local Area Network Interface

This chapter describes the local area network (LAN) interface and how to program it.

Chapter 9 Programming the Small Computer System Interface

This chapter describes the small computer system interface (SCSI) and how to program it.

Appendix A Workstation Address Map

This appendix lists each accessible register in the workstation with its address.

Appendix B Workstation Power–Up Flowchart

This appendix graphically presents the power–up procedure the workstation performs.

Appendix C Boot File Format

This appendix describes the format necessary for booting successfully using non–Data General magnetic media.

Appendix D I/O Connections and Specifications

This appendix lists the workstation connections and provides some related specifications.

Symbols and Conventions

The following conventions and symbols are used in this manual:

Symbol	Means
0x	In C programming examples, the combination of "0" and "x" indicates the values that follow are in hexadecimal. Note that the address and register values in this manual are given in hexadecimal unless indicated otherwise.
$\overline{\text{IRQ\_CIO}}$ IRQ_CIO*	Indicates a signal is a logical true (1) when asserted low. This manual uses the overbar in figures and the asterisk (*) in text and tables.
.PRINTER	**BOLDFACE CAPITAL** letters indicate a System Control Monitor system call.
ld	**Boldface lowercase** letters indicate an assembly language instruction.
SCM>	The default System Control Monitor prompt on single processor systems.
Jp#n/SCM>	The default System Control Monitor prompt on multiple processor systems, where n is the number of the attached job processor.
*	A signal name or mnemonic followed by an asterisk (*) indicates that the signal is true when asserted low (0).

All addresses are in hexadecimal unless otherwise noted.

All data is in binary unless otherwise noted.

Related Documents

This section lists manuals that provide more information about your AViiON computer system. For a complete list of AViiON 300 and 400 series documentation, see the "Documentation Set" following the Index.

Documents Available from Data General

MC88100 RISC Microprocessor User's Manual (014–001809)

> Describes the Motorola 88100 Central Processing Unit (CPU), including the registers, addressing modes, internal and bus timing, and assembly–language instruction set.

MC88200 Cache/Memory Management Unit User's Manual (014–001808)

> Describes the Motorola 88200 Cache/Memory Management Unit (CMMU), including the CMMU registers, the cache and cache coherency, memory management and user/supervisor space, the Processor bus (Pbus) and the Memory bus (Mbus).

*Using the AViiON*t *System Control Monitor (SCM)* (014–001802)

> Describes how technical users can use the commands and menus of the firmware monitor program to bring up software, control their system environment, and debug programs.

88open Binary Compatibility Standard (069–701043)

> Describes the binary standards for developing portable 88K code using the C programming language.

Other Organizations' Documents

The following documents are available from other organizations.

m*PD72120 Advanced Graphics Display Controller User's Manual*

> Describes the m*PD72120* graphics controller and how to program it. This document is available from NEC Electronics, Inc.

AIC–6250 High–Performance SCSI Protocol Chip

> Describes the AIC–6250 SCSI controller and how to program it. This document is available from Adaptec, Inc.

AM7990 Local Area Network Controller (LANCE) Technical Manual

> Describes the AM 7990 LAN controller and how to program it. This document is available from Advanced Micro Devices, Inc.

*Brooktree*r *Product Databook*

> Contact Brooktree Corporation to obtain this manual.

Memory Products Databook

Describes the MK48T02B 2Kx8 Zeropower/Timekeeper RAM and how to program it. This document is available from SGS–Thompson Microelectronics.

SCN2661 Enhanced Programmable Communications Interface (EPCI) Product Specification

Describes the SCC2692 universal synchronous/asynchronous data communications controller and how to program it. This document is available from Signetics, Inc.

SCC2692 Dual Asynchronous Receiver Transmitter (DUART) Product Specification

Describes the SCN2661 DUART and how to program it. This document is available from Signetics, Inc.

The VMEbus Specification

Describes Motorola's Versa Modula Europa bus (VMEbus), and how to program using the VMEbus. This document is available from Motorola Corp.

Z8536 CIO Counter/Timer and Parallel I/O Unit

Describes the Z8536 CIO and how to program it. This document is available from Zilog, Inc.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

If you have comments on this manual, please use the prepaid Comment Form that appears at the back. We want to know what you like and dislike about this manual.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, and you are within the United States or Canada, contact the Data General Service Center by calling 1–800–DG–HELPS for toll-free telephone support. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Free telephone assistance is available with your warranty and with most Data General service options. Lines are open from 8:30 a.m. to 8:30 p.m., Eastern Time, Monday through Friday.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1–800–877–4787 or 1–512–345–5316.

End of Preface

Contents

Chapter 1 – System Board Architecture

Chapter 2 – Programming the System Board

Chapter 3 – Interrupts, System Errors, and Bus Faults

Chapter 4 – Programming the Monochrome Graphics Subsystem

Chapter 5 – Programming the Color Graphics Subsystem

Chapter 6 – Programming the Keyboard Interface and Speaker

Chapter 7 – Programming the Serial Ports and Parallel Port

Chapter 8 – Programming the Local Area Network Interface

Chapter 9 – Programming the Small Computer System Interface Port

Appendix A – Address Map

Appendix B – Power–Up Flowchart

Appendix C – Boot File Format

Appendix D – System Board Connectors

Index

Documentation Set

Tables

Table

Figures

Figure

Chapter 1
System Board Architecture

This chapter describes the system board architecture, including the following topics:

- System architecture and configuration.
- System board architecture and configuration.
- The CPU set, including the CPU and CMMUs and cache coherency.
- Main memory and the memory interface.
- The Mbus and Sbus, and the interface between them.
- The graphics subsystem, including monochrome graphics, color graphics and the Z-buffer controller (400 series only).
- The I/O subsystem, including the keyboard port, serial ports, parallel port, Ethernet LAN interface, SCSI interface, and VMEbus interface (400 series only). The VMEbus section discusses VMEbus arbitration and VMEbus data transfers.
- Timers available to system programmers.
- Interrupts and the interrupt control logic.

Workstation Architecture and Configuration

Data General's AViiON™ 300 and 400 series stations use the Motorola 32-bit MC88100 RISC (Reduced Instruction Set Computing) processor and the Motorola MC88200 CMMU (Cache/Memory Management Unit). They run either Data General's DG/UX™ operating system or industry-available operating systems for MC88000-based systems.

The workstations consist of a system board, expansion memory modules, a fan, and a power supply in a desktop package. Using 4-Mbyte memory modules, 300 series stations support a maximum of 28 Mbytes of memory and 400 series stations support a maximum of 32 Mbytes of memory.

The workstation base configuration has no disk or tape storage and requires a remote file server for mass-storage. A compact, free-standing, mass-storage subsystem is available for workstations requiring local disk or tape storage.

Graphics subsystems differ between workstations. The 300 series system board comes with either a monochrome or color graphics subsystem, a frame buffer, and optionally with an additional 4 Mbytes of dynamic random access memory (DRAM) for applications that require extensive font storage such as the Kanji character set. The graphics subsystem for the 400 series station includes either an 8-bit or 24-bit color board with frame buffer, plus an optional Z-buffer board for three-dimensional applications requiring hidden line and hidden surface removal.

The monochrome graphics subsystem drives a 1280 x 1024 pixel, flicker-free display on a monochrome monitor; the color graphics subsystem drives a 1280 x 1024 pixel, flicker-free display on either a color monitor or a grayscale monitor.

Table 1-1 summarizes the configurations for each workstation.

Table 1-1 Workstation Configurations

Item	300 Series	400 Series
CPU set (1 CPU, 2 CMMUs)	1	1 or 2
CPU clock speed (MHz)	16.67, 20	16.67, 20
Physical memory range (4-Mbyte modules)	4 - 28 Mbytes	4 - 32 Mbytes
Graphics support	1-bit monochrome or 8-bit color	8-bit color or 24-bit color 24-bit Z-buffer (option)
VMEbus interface	—	1
Integrated I/O		
Parallel port (Centronics or Data Products)	1	1
Asynchronous ports (RS-232-C)	1	2
Mouse port RS-232-C	1	1
Keyboard port	1	1
SCSI port	1	1
LAN interface	1	1

System Board Architecture and Configuration

Each system board contains the following functional components: one or two CPU sets, main memory, system control logic, integrated I/O subsystem, and a monochrome or graphics subsystem. The 400 series stations also support a VMEbus and, optionally, a color graphics subsystem and a Z-buffer board. Two buses, the Memory bus (Mbus) and the System bus (Sbus), link the system board resources to each other. Figure 1-1 and Figure 1-2 illustrate the system board architecture. The shaded components apply only to the 400 series stations.

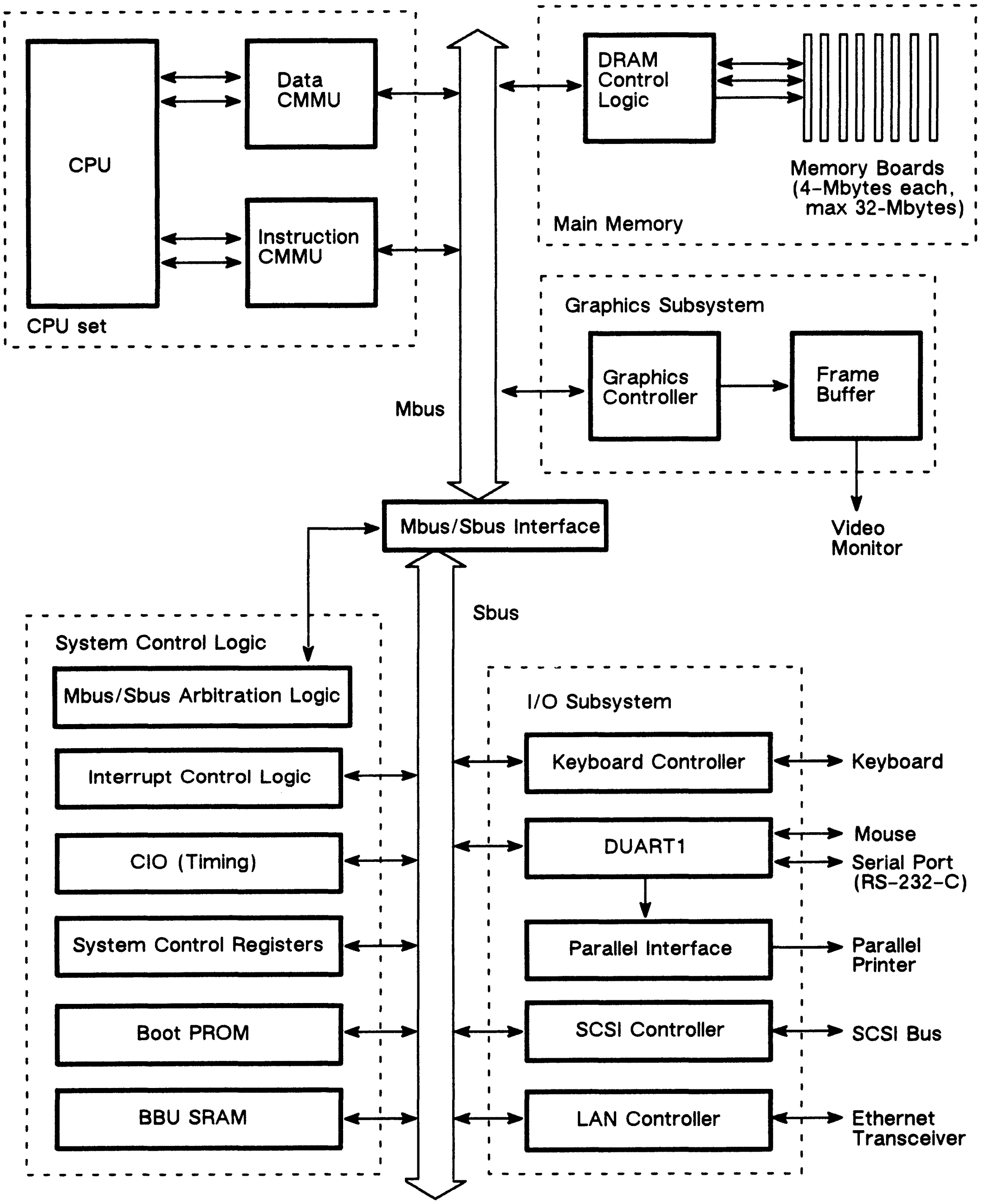


Figure 1-1 System Board Architecture – 300 Series

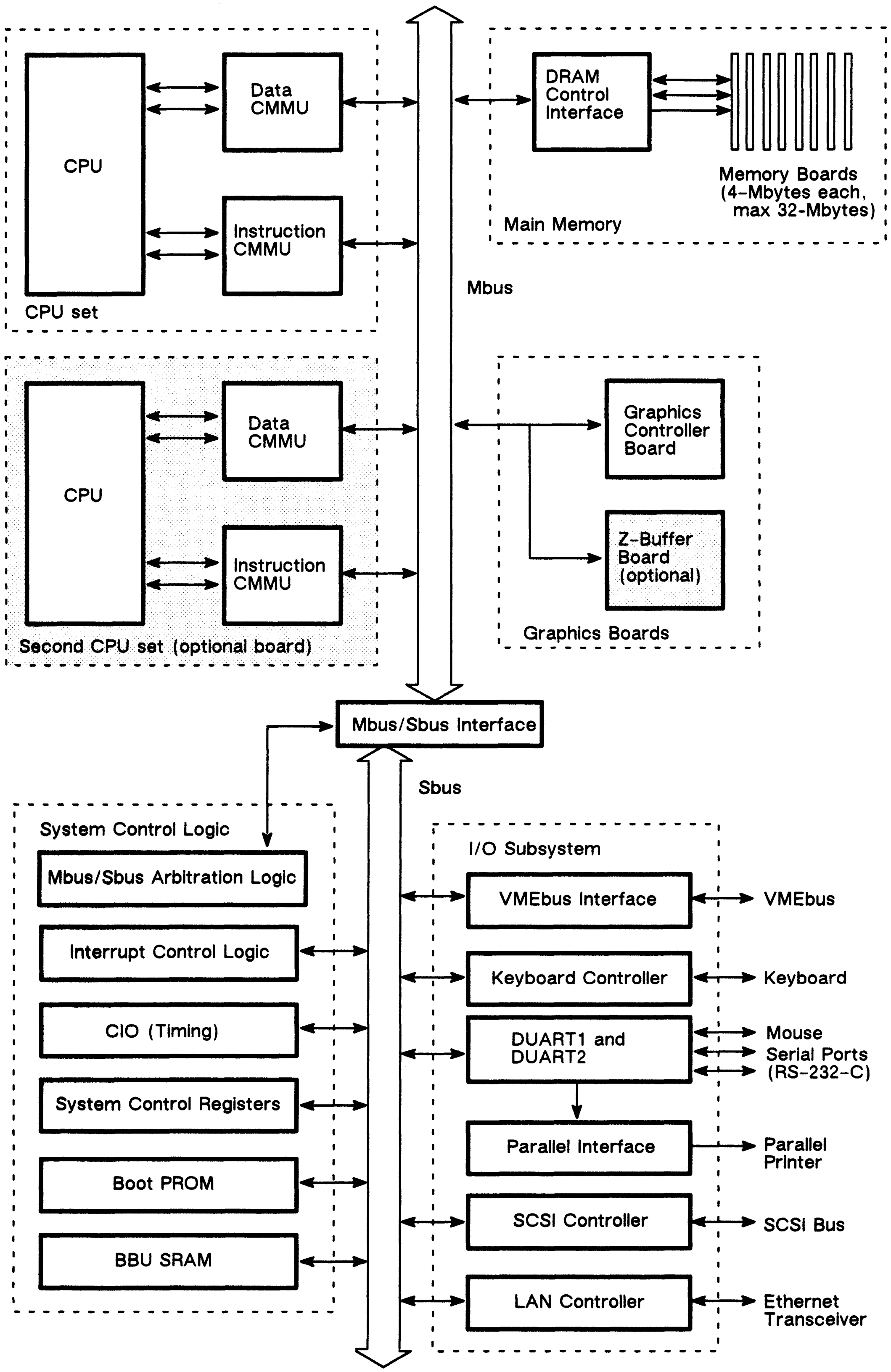


Figure 1–2 System Board Architecture – 400 Series

The CPU Set

As shown in Figure 1–3, each CPU set consists of one Motorola MC88100 CPU and two Motorola MC88200 Cache/Memory Management Units (CMMU). The CPU uses the 32-bit Reduced Instruction Set Computing (RISC) architecture with an internal, 32/64-bit floating-point processor. The CPU set supports a virtual address space of 4 Gbytes, bringing instructions and data into the processor over separate 32-bit paths. One CMMU provides caching and address translation for the instruction path while the other CMMU provides these functions for the data path. The CPU set communicates with the rest of the system using the Mbus. The 400 series station supports a second CPU set. Throughout this manual we refer to the CPU set as the CPU, except where necessary to differentiate between the components of the CPU set.

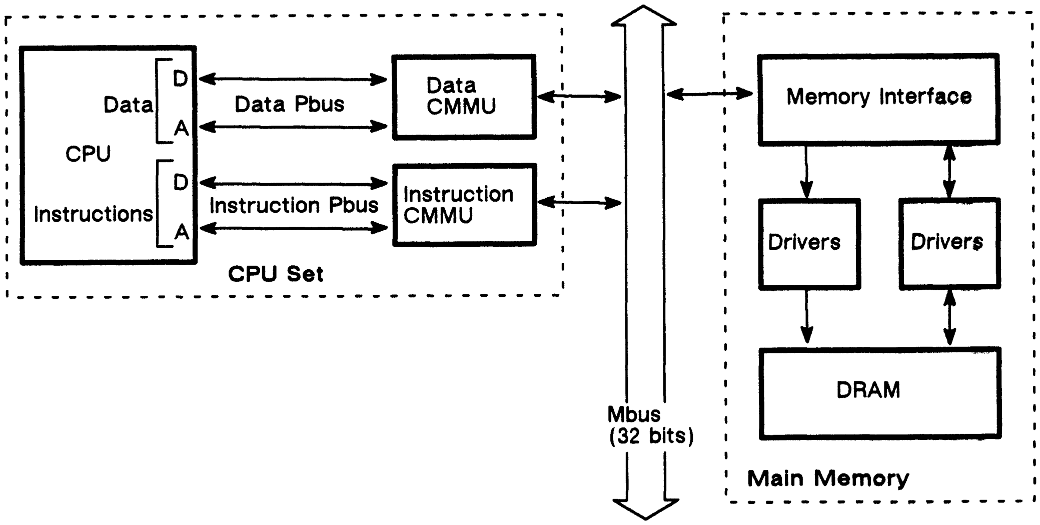


Figure 1–3 CPU Set and Main Memory

The CPU

The CPU uses the Harvard architecture with separate instruction and data ports. Each port consists of a 32-bit address bus, a 32-bit data bus, and bus control signals that together form the nonmultiplexed Processor bus (Pbus). Each Pbus connects a CPU Pbus port (instruction or data) to a respective CMMU (instruction or data).

The CPU has thirty-two 32-bit internal data registers that reduce data path traffic to approximately one load or store operation every four cycles. It has an internal 32/64-bit floating-point processor that shares the data registers with the processing unit.

For more information on the CPU, see Motorola's *MC88100 User's Manual*.

The Cache/Memory Management Unit (CMMU)

The Motorola MC88200 Cache/Memory Management Unit (CMMU) combines segmented, demand-paged virtual memory management with high-speed memory caching. Each CMMU contains 16 Kbytes of cache memory, 10 block address translation cache (BATC) entries, and 56 page address translation cache (PATC) entries. The CMMU provides 4 Gbytes of virtual memory space per process.

The cache prefetches and stores instructions and data. The cycle time is dependent on the CPU speed. Table 1-2 shows the clock frequencies and periods.

Table 1-2 CPU Clock Frequencies and Periods

Clock Frequency (MHz)	Clock Period (ns)
16.67	60
20	50

The CMMUs communicate with the CPU via the Processor bus (Pbus) and with main memory via the Memory bus (Mbus).

NOTE: Software must disable parity checking before reading some registers, such as those for the color graphics subsystem, SCSI interface, and LAN interface. For further information, see the *MC88200 User's Manual.*

The CMMU cache memory is temporary storage that speeds up both the execution of instructions and the reading and writing of data. If the cache is turned off, the CPU interacts directly with system memory. When the cache is turned on, the CPU reads from and writes to system memory through the cache. With the cache on, the CMMUs update system memory using either Writethrough or Copyback mode. When the cache is operating in

- Writethrough mode, all modifications to the cache are immediately written to system memory.
- Copyback mode, only the first modification of each cache location is written to system memory.

When a device reads system memory, the device expects to receive current accurate data. CMMU *cache coherency* makes sure that devices receive the most recent data when reading memory. Cache coherency is the ability of a CMMU to update obsolete data in system memory when a device reads a system memory location whose data is also resident in the CMMU cache. Cache coherency is valid only when the CMMU communicates with the CPU in Copyback mode. If the cache is turned off, or if the cache is on and updated in Writethrough mode, all modifications to the cache are immediately written to system memory. When a device addresses a memory location whose cache-equivalent data has been modified, the CMMU updates the data in memory before the device reads the data from memory.

Figure 1–4 illustrates cache coherency and Mbus *snooping*. *Snooping* is the comparison of Mbus global addresses with data cache tags. If the tag and the Mbus address match, and the cache data line has been modified, the cache data line is then written to memory. For a more detailed description of cache coherency and Mbus snooping, see Motorola's *MC88200 User's Manual.*

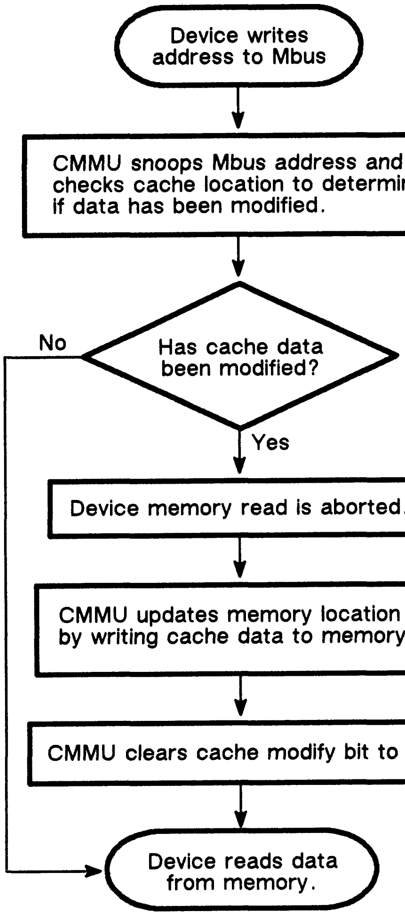

Device writes address to Mbus
CMMU snoops Mbus address and checks cache location to determine if data has been modified.
Has cache data been modified?
No
Yes
Device memory read is aborted.
CMMU updates memory location by writing cache data to memory.
CMMU clears cache modify bit to 0.
Device reads data from memory.

Figure 1–4 Mbus Snooping and Cache Coherency

The workstation's hardware supports only data cache coherency and not instruction cache coherency. Software can set up the data cache in either Writethrough or Copyback mode. Copyback mode greatly increases performance, while Writethrough mode continuously updates both the cache and memory. Since all SCSI DMA transfers are snooped, software should mark all SCSI DMA transfers as *global* (high–use, shared data). Correspondingly, since *no* LAN DMA transfers are snooped, software should mark all LAN DMA transfers as *local*, and be responsible for maintaining cache coherency of the LAN buffers.

Since the workstation's hardware does not support instruction cache coherency, the instruction cache does not snoop on the Mbus. As a result, software must maintain instruction cache coherency by invalidating the cache during I/O DMA transfers to instruction space.

The CMMUs define a block of data as four contiguous words (16 bytes). As shown in Figure 1–5, if the first word of the block is aligned on an address whose least significant nibble (4 bits) is 016 (such as FFF8 F12016), the CMMU completely reads or writes the block within one bus cycle. If this transfer crosses from one block address range to another, the transfer requires an additional bus cycle.

For more information on the CMMU, see Motorola's *MC88200 User's Manual*.

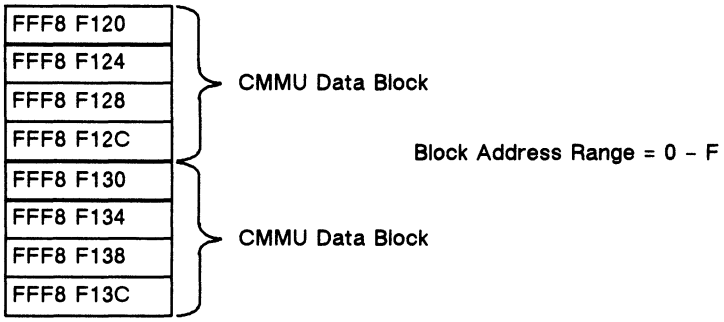


Figure 1–5 CMMU Data Block

Memory

The memory system consists of main memory, which includes onboard dynamic random-access-memory (DRAM), expansion DRAM boards and the associated memory interface logic, plus static RAM (SRAM) and programmable read-only-memory (PROM).

Main Memory

Main memory consists of dynamic random-access-memory (DRAM) and the memory interface logic needed to control accesses to the DRAM (see Figure 1–6). DRAM memory consists of 4-Mbyte memory modules that plug into the system board. Each 4-Mbyte memory module has 36 1-Mbit, fast-page 100-ns DRAMs that provide 32 data bits and 4 parity bits for each address. The 300 series stations accept as much as 28 Mbytes of RAM, while 400 series stations accept as much as 32 Mbytes of RAM.

If the workstation has 16-Mbyte memory modules, it can support either 112 Mbytes (300 series stations) or 128 Mbytes (400 series stations) of DRAM. Each 16-Mbyte memory module has 36 4-Mbit, fast-page 100-ns DRAMs that provide 32 data bits and 4 parity bits for each address.

Main memory is contiguous and begins at address 0000 0000. Each location has 36 bits: 32 bits for the data word and 4 bits for parity. The parity logic, located on the system board, generates and checks the parity bits.

The DRAM uses byte parity; one parity bit for each byte of data. The parity bits have separate data–in and data–out connections to the DRAMs, and the write parity data is driven to the DRAMs in a way that allows diagnostic software to force parity bits high during a write to memory. This logic also provides the control signals for refreshing the DRAM array.

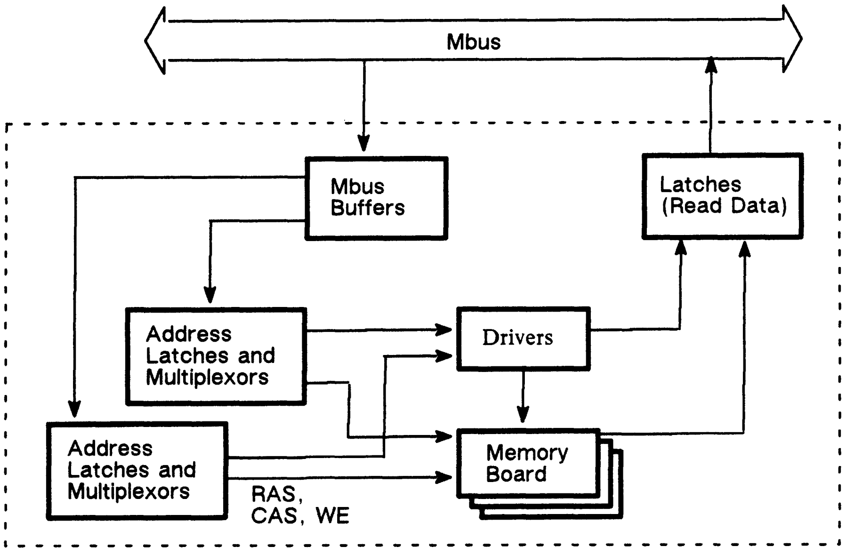


Figure 1–6 Main Memory

Main-Memory Interface

The memory interface connects the DRAM to the Mbus, controlling data transfers to and from the DRAM. When a device addresses memory, the memory interface enables the memory module that contains the addressed location; then reads from or writes to the location. The interface consists of the following:

- Address and data latches.
- Address and data drivers.
- Control logic and memory timing to regulate the memory strobes (RAS and CAS) and write enable (WE).

The memory interface responds only to addresses in the lower 128 Mbytes of system address space. Table 1–3 shows the number of clock cycles it takes to execute reads and writes.

Table 1–3 Memory Read and Write Cycles

	Number of Clock Cycles at CPU Speed:	
Type of Access	16.67 (MHz)	20 (MHz)
Single–word write	5	6
Single–word read	6	7
Block write (4 words)	11	12
Block read (4 words)	12	13

The memory interface sends status signals to the workstation's Parity Address Register (PAR) to indicate when one or more modules has 100 ns DRAMs installed and the number of modules with 4–Mbit DRAMs. For information on these status bits, see the description of the Parity Address Register (PAR) later in this chapter.

Battery Backed Up (BBU) SRAM and PROM

The BBU SRAM provides 2 Kbytes of nonvolatile storage for diagnostics, system configuration, and boot information. (Note that within this manual we also refer to the BBU SRAM as NOVRAM or nonvolatile RAM.) The 128–Kbyte PROM contains powerup diagnostic and initialization code, including the local code for booting the system over an Ethernet. The diagnostic registers provide information that allow diagnostic software to control the state of the system board, determine system board status, and obtain Mbus parity error status information.

The System Control Logic

The system control logic allocates and controls the system board resources, and includes the following: Mbus and Sbus arbitration logic, interrupt control logic, timing services, boot (PROM), battery-backed-up static RAM (BBU SRAM, also called nonvolatile RAM or NOVRAM), and system control registers. The timing services include the time-of-day (TOD) clock, time-of-boot (TOB) clock and the programmable interval timer (PIT). The NOVRAM contains the system configuration information. The system control logic resides on the Sbus. Figure 1-7 illustrates the system control logic.

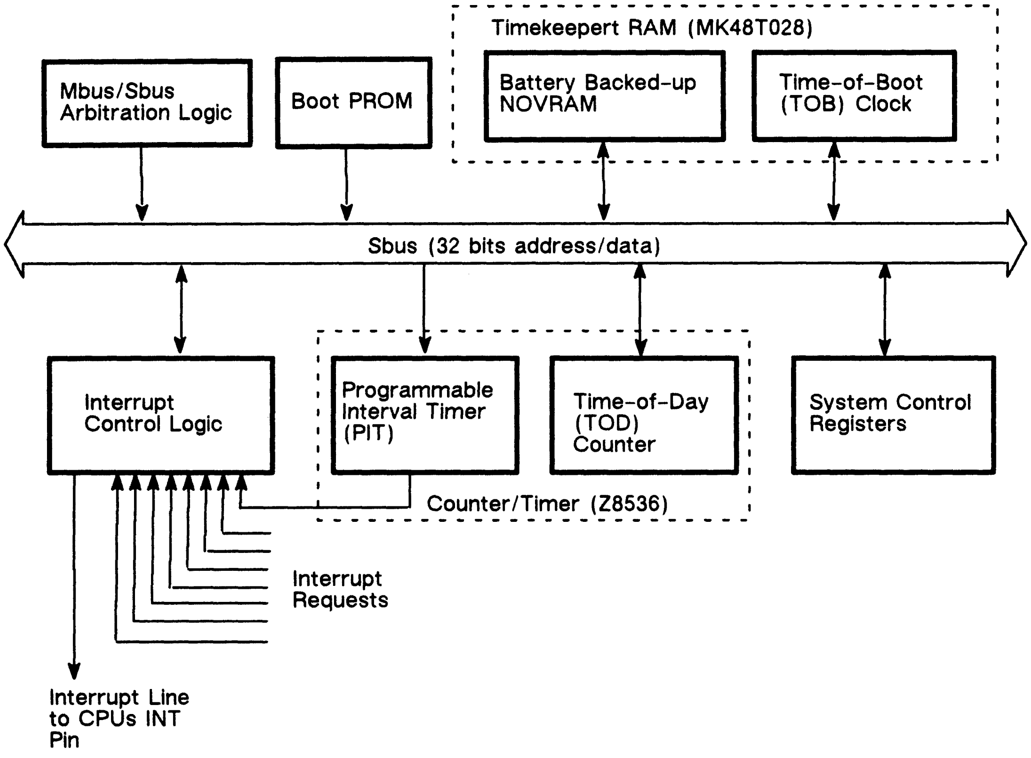

Timekeepert RAM (MK48T028)
Mbus/Sbus Arbitration Logic
Boot PROM
Battery Backed-up NOVRAM
Time-of-Boot (TOB) Clock
Sbus (32 bits address/data)
Interrupt Control Logic
Programmable Interval Timer (PIT)
Time-of-Day (TOD) Counter
System Control Registers
Counter/Timer (Z8536)
Interrupt Requests
Interrupt Line to CPUs INT Pin

Figure 1-7 System Control Logic

The Mbus and the Sbus

The Memory bus (Mbus) and the System bus (Sbus) pass address, data, and status between the CPU, memory, system control logic, and I/O controllers.

The Mbus is a 32-bit, multiplexed address/data bus generated by the CPU. It connects the CPU, main memory and graphics subsystem to each other and to the Sbus. The Mbus supports block transfers and cache coherency.

The Sbus is a 32-bit, multiplexed address/data bus that connects I/O controllers (except the video controller), system control logic and registers to each other and to the Mbus. The Sbus is a "subset" of the Mbus, providing a 32-bit address/data path, but not supporting block transfers or cache coherency protocols.

The Mbus

The Mbus connects all of the system board resources to each other. The Mbus includes a 32-bit multiplexed Data/Address bus and Arbitration/Control lines. The Mbus links the central processing components (the CPUs and CMMUs) to the memory, I/O, control logic and registers. Table 1-4 lists and describes the signals on the Mbus.

Table 1-4 Mbus Signals

Signal	Description
Data Transfer:	
AD[31-00]	Address/data
Bus Arbitration:	
BR	Bus request
BA	Bus acknowledge
BB*	Bus busy
AB*	Arbitration busy
Control and Status:	
C[6-0]	Control
ST[3-0]	Local status
SS[3-0]*	System status

When two or more devices try to access the Mbus at the same time, bus arbitration logic determines which device is granted the Mbus. The Mbus can be accessed by a system board CPU or by any VME controller.

The Sbus

Sbus masters cannot transfer data to/from Sbus slaves. If a device on the Sbus needs to communicate with another device on the Mbus, it must do so through the CPU. Table 1-5 defines the Sbus signals.

Table 1-5 Sbus Signals

Signal	Description
Data Transfer:	
SI[31-0]	Address/data
Control and Status:	
STRB[1-0]	Strobe
AS*	Address strobe
G	
READ	Read data
WAIT	Wait
PAUSE	Pause

The Mbus/Sbus Interface

The Mbus/Sbus interface links the Mbus and Sbus to each other. Arbitration logic regulates requests to access the Mbus and the Sbus.

The Mbus/Sbus interface performs the following tasks:

- Provides a bidirectional data path between the two buses.
- Defines the system memory map and decodes addresses.
- Performs bus arbitration.
- Generates and çhecks parity.
- Notifies the CPU of system error conditions.

The Ethernet and SCSI controllers support Direct-Memory Access (DMA) and have 16-bit interfaces. To avoid leaving "holes" in the memory, these DMA devices need to address memory on half-word (16-bit) boundaries. Since the Mbus supports only word-aligned accesses to memory, the Mbus/Sbus interface allows Sbus DMA masters to access memory on half-word boundaries. The Mbus/Sbus interface consists of all the logic necessary to interface the Sbus to the Mbus, so that it need not be replicated for each 16-bit peripheral. Sbus masters are half-words aligned on even addresses (i.e., 0, 2, 4, 8, A, C, E).

The Graphics Subsystem

The graphics subsystem controls the bit-mapped display device — the monitor. The subsystem resides on the Mbus and consists of a graphics display controller, a frame buffer, and a CRT interface circuit that creates and sends the video signals to the monitor. Two types of graphics subsystems are available: monochrome, or color. The 300 series system board contains either a monochrome graphics subsystem or an 8-bit color graphics subsystem. The 400 series station uses separate boards for the color graphics subsystem supporting either 24-bit or 8-bit color graphics and an optional Z-buffer for hidden line and hidden surface removal.

Both the color graphics subsystem and the monochrome graphics subsystem generate parity bits when they write data to the Mbus, but they do not check parity when they receive data.

Monochrome Graphics

A NEC uPD72120 advanced graphics display controller drives the monochrome graphics subsystem. This controller is a slave to the CPU, and it performs drawing and Bit Block Transfer (BITBLT) operations that the CPU would otherwise have to perform. The controller also refreshes the Video RAMs (VRAMs) and the screen.

The VRAMs make up a 1280 (horizontal) by 1638.4 (vertical) pixel single-plane frame buffer. This buffer provides for a 1280 by 1024 pixel screen display plus extra off-screen memory for storing fonts, icons, and pull-down windows.

The CPU can directly access the frame buffer to manipulate the video image and to use software that runs with a dumb frame buffer. However, because the CPU accesses the frame buffer through the display controller, video memory accesses take considerably longer than accesses to main memory.

For more information on the monochrome graphics subsystem, see Chapter 4, "Programming the Monochrome Graphics Subsystem."

Color Graphics

One or more Data General Complementary Metal Oxide Semiconductor (CMOS) gate arrays display and control the video display and video memory for the color graphics subsystem. The gate arrays are slaves on the Mbus that off load the CPU from performing vector drawing and pixel block transfer operations. The 8-bit color subsystem uses a single color graphics gate array; the 24-bit color subsystem contains three gate arrays.

VRAMs make up a 1536 by 1024 pixel frame buffer. The frame buffer provides a 1280 by 1024 pixel screen display and extra off-screen storage for fonts and menus. The color units differ in the number of planes available to store color information and the amount of simultaneous colors each can display from a 16.7-million color palette. The 8-bit color subsystem has 8 planes and provides 256 simultaneous colors. The 24-bit color subsystem uses 24 planes, providing 16.7-million colors. Both color units have two planes for storing overlays or window information.

For more information on the color graphics subsystem, see Chapter 5, "Programming the Color Graphics Subsystem."

Z-Buffer Controller

The 24-bit Z-buffer controller, an option in 400 stations, supports hidden line and hidden surface removal for three-dimensional applications. The Z-buffer consists of a gate array that displays and controls the video display and video memory for the frame buffer. The Z-buffer board plugs into the color graphics board.

The color graphics controller decodes addresses to registers and arrays within the Z-buffer, and also generates control signals for the Z-buffer controller. The Z-buffer controller includes the logic for performing *Hither* and *Yon* clipping, and supports color graphics screen resolutions and context switching. The Z-buffer generates parity when read.

The I/O Subsystem

The I/O subsystem contains integrated controllers that provide communication between the system board and peripheral devices. The I/O subsystem resides on the Sbus and consists of a keyboard port, a mouse port, one or more serial ports, a parallel port, a SCSI port, and a LAN interface, and in 400 series stations, a VMEbus interface.

Keyboard Port

The keyboard interface supports both AT-compatible and Japanese AX-compatible keyboards. The interface converts the serial input from the keyboard into parallel data; converts parallel data into a serial output for the keyboard; generates the start, stop, and parity bits; and controls keyboard line protocol.

Mouse Port

The mouse port uses an RS-232-C interface to communicate with a mouse. The mouse port is one channel of a dual universal asynchronous receiver/transmitter (DUART).

Serial Ports

The 300 series station has one serial port that supports either an RS-232-C interface or an RS-422 interface. The 400 series station has two RS-232-C ports that support modems. Each serial port (RS-232-C and RS-422) is one channel of a DUART.

Parallel Port

The parallel port supports peripherals with either a Centronics interface or a Data Products interface.

Small Computer Systems Interface (SCSI) Port

The SCSI port supports access to mass-storage devices on the external SCSI bus. It connects to an ANSI-standard SCSI bus. The SCSI port resides on the Sbus, and consists of a DMA controller and a SCSI controller. The DMA controller supports 16-bit data transfers between the SCSI controller and main memory.

To start a SCSI bus task, the CPU programs registers in the DMA and SCSI controllers. The SCSI controller signals the DMA controller when it is ready for a data transfer, and again when it completes the transfer or requires CPU intervention.

Local Area Network (LAN) Interface

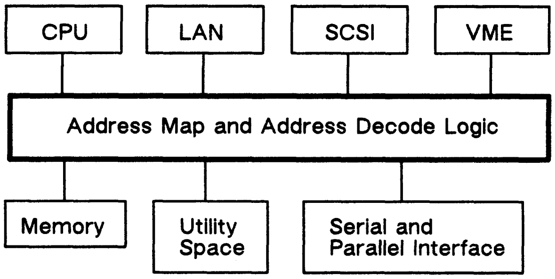
The Ethernet LAN interface resides on the Sbus, and functions as either an Sbus slave or a master. It consists of an Ethernet controller and a serial interface. An external Ethernet transceiver box may be connected directly to the serial interface using a D15 connector.

The CPU acts as the I/O controller for the Ethernet controller, using a special part of main memory as the data buffer. The Ethernet controller has direct-memory access (DMA) channels for transferring data between itself and main memory, and it supports 16-bit data transfers.

The VMEbus Interface (400 Series Only)

The Versa Modula Europa bus (VMEbus) links the VME controllers and the system board to each other. The 400 station VMEbus interface logic supports Revision C.1 of *The VMEbus Specification*. The system board's VME interface logic provides address decoding (see Figure 1-8), bus arbitration, and interrupt handling.

When any VME controller, including the system board, is master of the VMEbus, it can transfer words, halfwords and bytes to another VME controller; they *can not send blocks* of data over the VMEbus. The system board registers and memory are mapped to regulate accesses by other VME controllers. Address decode logic enables access to portions of the address space (i.e., utility space, the Ethernet LAN interface, or a specific expansion memory board).


CPU
LAN
SCSI
VME
Address Map and Address Decode Logic
Memory
Utility Space
Serial and Parallel Interface

The address map and address decode logic control accesses between the system resources.

Figure 1-8 Address Decoding

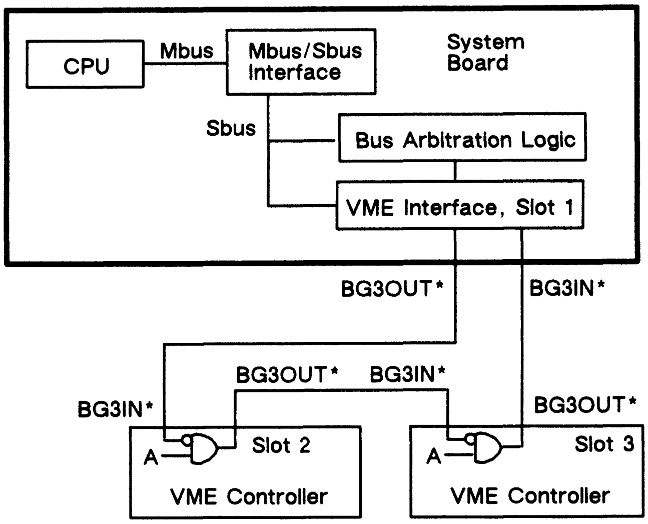
Table 1-6 lists the VMEbus signals. The system board provides the 16-MHz system clock to the SYSCLK line of the VMEbus. The system board also monitors and handles the interrupt, bus arbitration, and ACFAIL lines.

Table 1-6 VMEbus Signals

Signal	Description	Signal	Description
Data Transfer:		**Clocks:**	
A[31-01]	Address	SERCLK	Serial clock
AM[5-0]	Address modifier	SYSCLK	System clock
D[31-00]	Data	**Failures:**	
DS[1, 0]*	Data strobe	ACFAIL*	Ac failure
AS*	Address strobe	SYSFAIL*	System failure
LWORD*	Long word	SYSRESET*	System reset
SERDAT*	Serial data		
WRITE*	Write	**Interrupts:**	
DTACK*	Data transfer acknowledge	IRQ[7-1]*	Interrupt request
		IACK*	Interrupt acknowledge
Bus Arbitration:		IACKIN*	Interrupt acknowledge in
BR[3]*	Bus request	IACKOUT*	Interrupt acknowledge out
BG[3]IN*	Bus grant in		
BG[3]OUT*	Bus grant out	**Power:**	
BBSY*	Bus busy	+5V	+5 V dc
BCLR*	Bus clear	+12V	+12 V dc
BERR*	Bus error	GND	Ground

VMEbus Arbitration

VME controllers, including the system board, communicate through the VMEbus. Before communicating over the VMEbus, a VME controller requests access to the VMEbus by asserting the bus request line BR3*. VMEbus arbitration logic, located on the system board, detects this request and grants the bus, when available, via the bus grant lines BG3IN* and BG3OUT*. According to *The VMEbus Specification*, the VMEbus has four sets (or levels) of bus request and grant signals to regulate bus access. This implementation of the VMEbus uses only level 3 (BR3*, BG3*IN and BG3*OUT). The bus grant lines are daisy-chained from VME controller to VME controller; the board in slot 1 of the VMEbus has the highest access priority, and the board in slot 2 is next in priority. The system board has the lowest VMEbus priority. Therefore if all the VME controllers request access to the VMEbus at the same time, the board in slot 1 will access the VMEbus. When a controller accepts control of the VMEbus, it holds BG3OUT* High and asserts BBSY* Low. When it releases the VMEbus, it deasserts BBSY*, and the arbitration process repeats. This daisy-chain configuration is illustrated in Figure 1-9. *The VMEbus Specification* describes these signals in greater detail.



NOTE: If A is 1, the VME controller passes the bus grant to the next controller. If A is 0, the VME controller takes the bus grant and uses the VMEbus.

Figure 1-9 VMEbus Grant Daisy-Chain

Registers

The registers include

- System control registers (see Chapter 2)
- Miscellaneous registers (see Chapter 2) include the time-of-boot (TOB) clock registers and the Programmable Interval Timer (PIT) and time of day (TOD) clock registers.
- Interrupt registers (see Chapter 3) which include CPU interrupt registers and VME interrupt registers.
- Monochrome graphics registers (see Chapter 4).
- Color graphics registers (see Chapter 5).
- Keyboard port registers (see Chapter 6).
- Serial port registers (see Chapter 7).
- Parallel port registers (see Chapter 7).
- Ethernet LAN registers (see Chapter 8).
- SCSI registers (see Chapter 9).

Timers Available to System Programmers

The Programmable Interval Timer (PIT), time of boot (TOB) and time of day (TOD) clocks are available to the operating system.

The PIT provides internal timing functions for use by the operating system. This timer is a write–only countdown timer that interrupts the CPU when the count reaches zero.

The TOB and TOD clocks provide timestamps for the operating system. On powerup the TOB clock supplies the initialization software with the time and date. After powerup, the TOD clock keeps the time.

Interrupts and the Interrupt Logic

Interrupt logic:

- Receives and asserts interrupts
- Manages the interrupt registers
- Notifies the CPU of pending interrupt requests

End of Chapter

Chapter 2
Programming the System Board

This chapter describes the following topics:

- How to program the CPU.
- How to read from and write to memory.
- Address mapping.
- Mbus and Sbus arbitration.
- Bus masters and slaves.
- How to address system board resources and system memory from the CPU.
- How to address a VME controller from the CPU (400 series stations only).
- How to address system board resources from a VME controller (400 series stations only).
- How to program the time-of-boot clock, nonvolatile RAM, programmable interval timer, and time-of-day clock.
- The boot PROM.
- The System Control Monitor (SCM).

Programming the CPU

The CPU is programmed using the Reduced Instruction Set Computing (RISC) instruction set.

The registers are memory-mapped (mapped to unique locations in the workstation address space), therefore they are programmed using the Load Register from Memory (**ld**) and Store Register to Memory (**st**) instructions.

NOTE: Programming a dual-CPU or single-CPU workstation is identical; both types of workstation use the same instruction set. The WHOAMI register indicates the number of CPU chip sets the workstation contains.

For detailed information on the instruction set and the CPU programmable registers, see Motorola's *MC88100 User's Manual.* For detailed information on the CMMU programmable registers, see Motorola's *MC88200 User's Manual.*

The CPU supports the *big-endian* scheme for ordering bytes in memory. In this scheme, lower memory bits correspond to high-order bytes, as shown in Figure 2-1.

31 24	23 16	15 8	7 0
Byte 0	Byte 1	Byte 2	Byte 3

31 16	15 0
Half-Word 0	Half-Word 1

63 32	31 0
Word 0	Word 1

Figure 2-1 Big-Endian Byte Ordering

Addressing Memory

Main memory occupies the lower 128 Mbytes of address space. All memory must be contiguous, starting at address 0000 0000. If a vacant memory location, (i.e., one that has no RAM) is read, the memory interface returns data of all 1s and parity of all 1s, resulting in a parity error. If a vacant memory location is written to, the data is lost and an error is not generated.

To avoid this problem, the system should size the memory by writing to and reading from the memory and comparing the results. To do this, software needs only to perform a write/read sequence at 4–Mbyte boundaries — the smallest increment of memory expansion.

When the system is powered up or reset, the CPU reads its first instruction from location 0000 0000. This address is normally the start of main memory, but during a reset or a powerup, the system maps the boot PROM to 0000 0000. The memory interface disables the DRAM output drivers during a powerup or reset; main memory cannot be read from, but it can be written to.

Data Transfers to/from Memory

Mbus/Sbus masters can read from and write to system memory. When a device reads from system memory, the memory sends the data one word (32 bits) at a time until it receives an End of Request (EOR) signal from the requestor or until the read crosses to a new block of data. If the block crossing occurs before the EOR, the memory module asserts an end of data signal.

When a device writes data to memory, the memory receives the data (in bytes, half–words, or words) until it receives an EOR or until a block crossing occurs.

When a CPU reads or writes a block (4 words) of data from system memory, the following occurs:

1. The CPU writes the beginning address to memory.
2. The memory module receives the address.
3. If the memory module cannot respond to the address immediately (within one clock cycle), it inserts wait states until it is ready to receive or send the data.
4. Either the CPU (write to memory) or the memory module (read from memory) writes a word of data to the Mbus.
5. The memory module automatically increments the address to point to the next word; then writes or reads the new data. This continues until the end of the data block is reached.

Address Map

The Mbus/Sbus interface contains the address decode logic that defines the address map. This map consists of three main areas: main memory, video memory, and Mbus utility space.

The system memory occupies a contiguous block in the lower 128 Mbytes of physical address space, starting at address 0000 0000.

The video memory occupies a 128–Mbyte block at the beginning of the upper half of the physical address space, starting at address 8000 0000. This space provides a memory–mapped video frame buffer for the graphics subsystem.

The VMEbus address space consists of A32 and A24 space. The A32 space occupies two blocks of physical address space, a 1.57–Gbyte block starting at 2000 0000, and a 1.8–Gbyte block starting at 9000 0000. The A24 space occupies a 16–Mbyte block starting at FE00 0000.

The Mbus utility space is mapped into the upper 4 Mbytes of physical address space, starting at address FFC0 0000. This space is reserved for the memory–mapped control and status registers of integrated I/O devices, the boot PROM, and other system control functions.

When the workstation is powered up, or when a system reset occurs, the utility space, which includes the boot PROM, is mapped to address 0000 0000. During a system reset or boot, only the lower 20 address bits A[19:0] are used; the upper 12 address bits A[31–20] are ignored. This points all addresses into the remapped 4–Mbyte utility space. During reset, main memory will accept data from write operations. This situation exists until the Diagnostic Control Register (DCR) bit 5 is set to 1.
Table 2–1 identifies system memory space for 300 and 400 series stations. Appendix A, "Address Map," has complete address maps for the 300 and 400 series stations.

Table 2–1 System Memory Space

Memory Resource	Size (Bytes)	Address Range
300 Series Stations		
Boot PROM	128 K 128 K	0000 0000 – 0001 FFFF (During boot) FFC0 0000– FFC1 FFFF (After boot)
System memory	128 M	0000 0000 – 07FF FFFF (Write only during boot)
Video memory	128 M	8000 0000 – 87FF FFFF
400 Series Stations		
Boot PROM	128 K 128 K	0000 0000 – 0001 FFFF (During boot) FFC0 0000– FFC1 FFFF (After boot)
System memory	256 M	0000 0000 – 07FF FFFF (Write only during boot)
Video memory	128 M	8000 0000 – 87FF FFFF
VME A32 space	2 G 1.8 G	1000 0000 – 7FFF FFFF 9000 0000 – FDFF FFFF
VME A24 space	16 M	FE00 0000– FEFF FFFF

Mbus and Sbus

The following sections describe the Memory bus (Mbus) and System bus (Sbus).

Mbus and Sbus Arbitration

The Mbus and Sbus support several controllers that can become master of these busses. Access to the busses is regulated using the prioritized arbitration scheme. Mbus and Sbus masters share the same arbitration unit, and the current master is the master of both the Sbus and the Mbus. Sbus masters can communicate only with Mbus devices, not with other Sbus devices. Mbus masters can communicate with other Mbus devices or with Sbus devices. The Mbus/Sbus interface ensures that the data, addresses, control, and status all flow in the proper directions to effect the correct transfer.

Due to the strict latency requirements of the Ethernet LAN interface, Mbus masters must restrict their transfers to less than 5 ms. They must yield the bus for at least one cycle between word transfers, or in the case of block devices, between every 4-word block. This allows higher priority devices to access the bus in a timely fashion.

The Mbus/Sbus interface incorporates a time-out mechanism that terminates a bus cycle and returns an Mbus error when an Mbus wait does not allow the DRAMs to be refreshed. The time interval varies depending on when the Mbus wait is asserted in reference to the time from the previous DRAM refresh. For example, on 16.67-MHz workstations, the time varies from 15-31 µs; on a 20-MHz workstation, the time varies from 13-26 µs.

The Mbus/Sbus interface's Mbus arbitration logic uses a priority mechanism to grant bus mastership. There is no fairness incorporated into this arbitration mechanism.

Table 2-2 lists the potential bus masters and their priorities. The Ethernet LAN interface has the highest priority because it has the strictest bus latency requirements.

Table 2-2 Sbus Master Priorities

Device	Priority
Ethernet LAN interface	Highest
SCSI DMA controller	↕
VME interface	
Data CMMU 0	
Instruction CMMU 0	
Data CMMU 1	
Instruction CMMU 1	Lowest

Master and Slave Devices

A device is a master device when it has control of the system to read from and write to other devices. Only intelligent devices, such as the CPU/CMMU or a VME controller, can be a master. Slave devices (those accessed by a master) respond to commands from the master. Masters and slaves vary from transaction to transaction, and in many instances slaves become masters in response to commands from a previous master.

For example, if a CPU/CMMU requests data from a disk, the CPU/CMMU is the initial master device. The disk controller is the slave device that receives the read command, obtains the data from the disk, and places the data into its buffer. The disk controller then becomes the master, writes the data from its buffer to system memory. System memory is the slave device which receives the data from the disk controller. The CPU/CMMU is again the master device as it reads the data from system memory.

Data Alignment

All data transfers using the System bus (Sbus) must be aligned on word boundaries; all locations on the Sbus are word aligned. As a result, software must access Sbus slaves as 32-bit quantities regardless of how many bits of data the register actually contains. This ensures that the data is correctly aligned.

Addressing System Board Resources and System Memory

This section describes how to address the system board resources and system memory. The following numbered steps, in conjunction with Figure 2–2 and Figure 2–3, describe how the CPUs access the system board resources and the system memory.

1. The CPU puts a 32–bit address onto the Mbus.
2. The address decode logic decodes the address bits (2a) and enables access to a device (2b) such as onboard memory, expansion memory boards, utility space, and VME space.
3. The 32–bit address points to a location within the selected device.

Figure 2–2 shows how the CPU addresses memory, while Figure 2–3 shows how addresses are decoded.

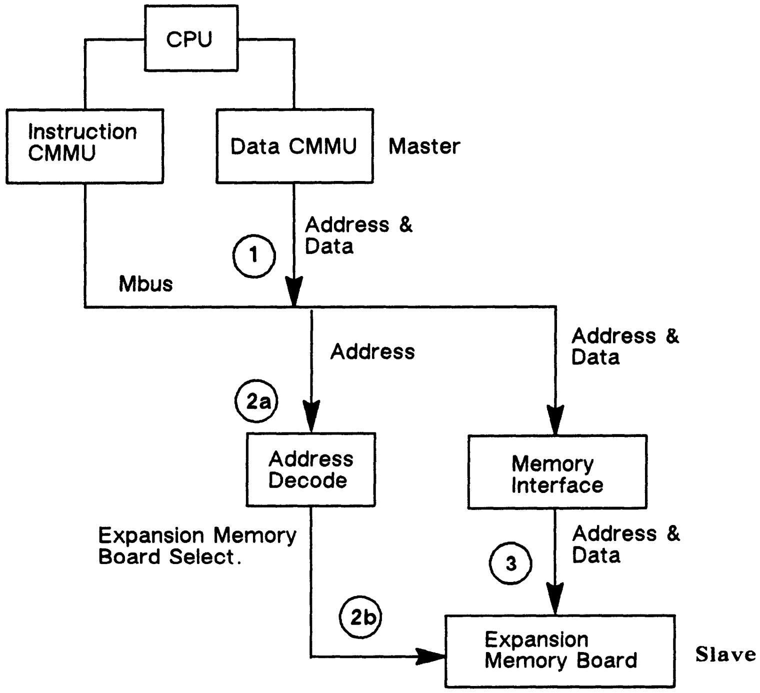


Figure 2–2 How the CPU Addresses System Memory

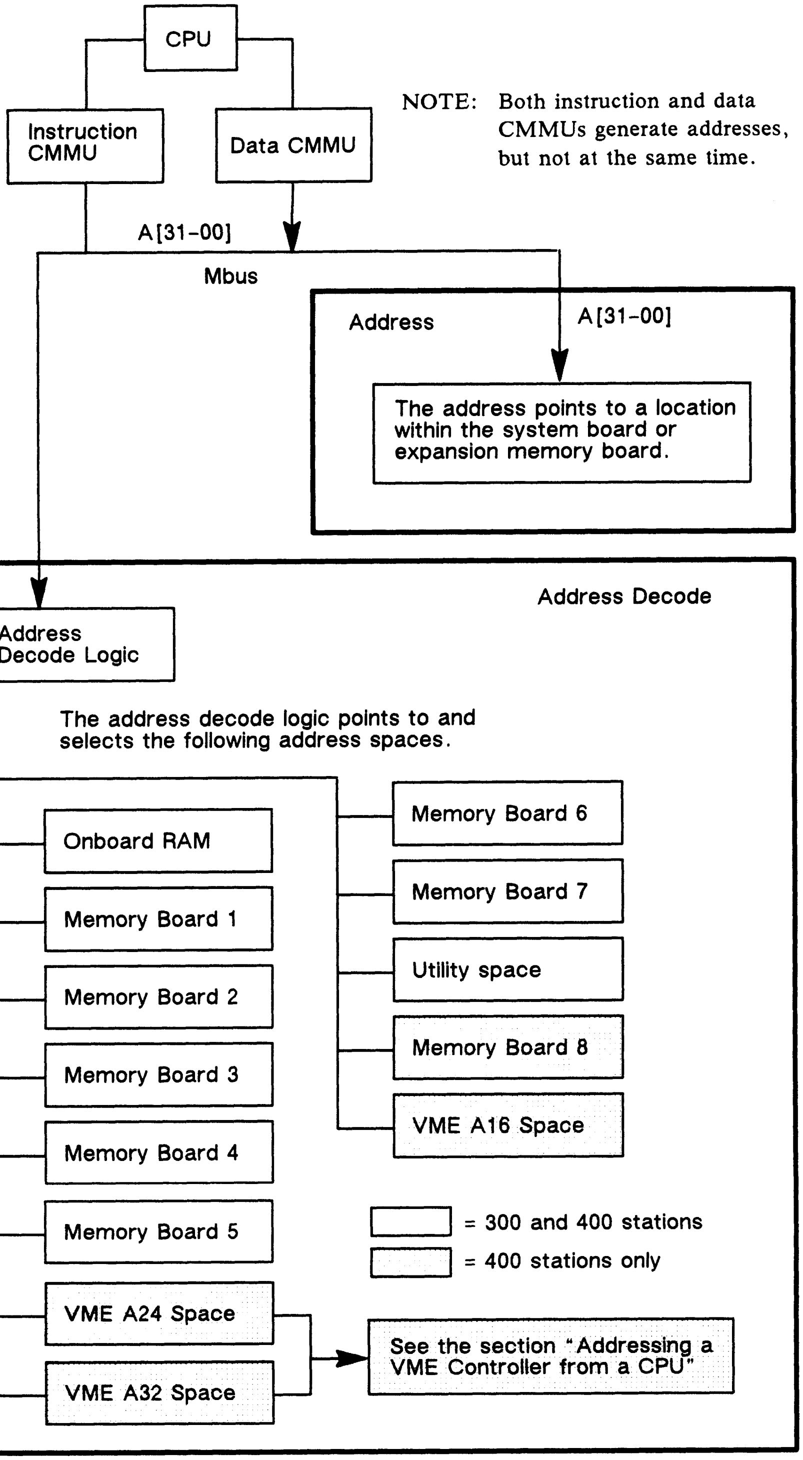


Figure 2–3 Decoding Addresses from the CPU

Addressing VME Controllers (400 Series Only)

When the CPU addresses a VME controller, the address decode logic decodes address bits A[31–16]. AM[5–4] is translated from VME address space select. The address modifier bits inform the VME controllers which VME space the address is written to: A16, A24, or A32 space. See Table 2–3 and Table 2–4 in the description of the EXTAM register for more information about the address modifier bits.

The following numbered steps in conjunction with Figure 2–4 and Figure 2–5 describe how the CPUs access the system board resources and the system memory.

When the CPU addresses a VME controller, the system board decodes the address as follows:

1. The CPU puts a 32–bit address onto the Mbus.
2. The address decode logic decodes the upper 10 address bits. If these 10 bits point to the VMEbus, the following steps are performed simultaneously.
3. The VME interface performs the following tasks simultaneously:
   - Writes the address to the VMEbus from the Sbus.
   - Writes the address modifier AM[5–0] to the VMEbus. AM[5–4] is translated from VME address space select (i.e., A32, A24, A16). AM[3–0] comes from Extended Address Modifer (EAM[3–0]) of the EXTAM register.
   - The address modifier bits AM[3–0] are put on the VMEbus from the EXTAM register.

Figure 2–4 illustrates how an address passes from the system board to a VME controller. In this example, the CPU is instructing a disk controller to send a block of data to the expansion memory.

Figure 2–5 illustrates how the VMEbus decodes addresses, and where the address modifier bits come from.

A description of the EXTAM register follows Figure 2–5.

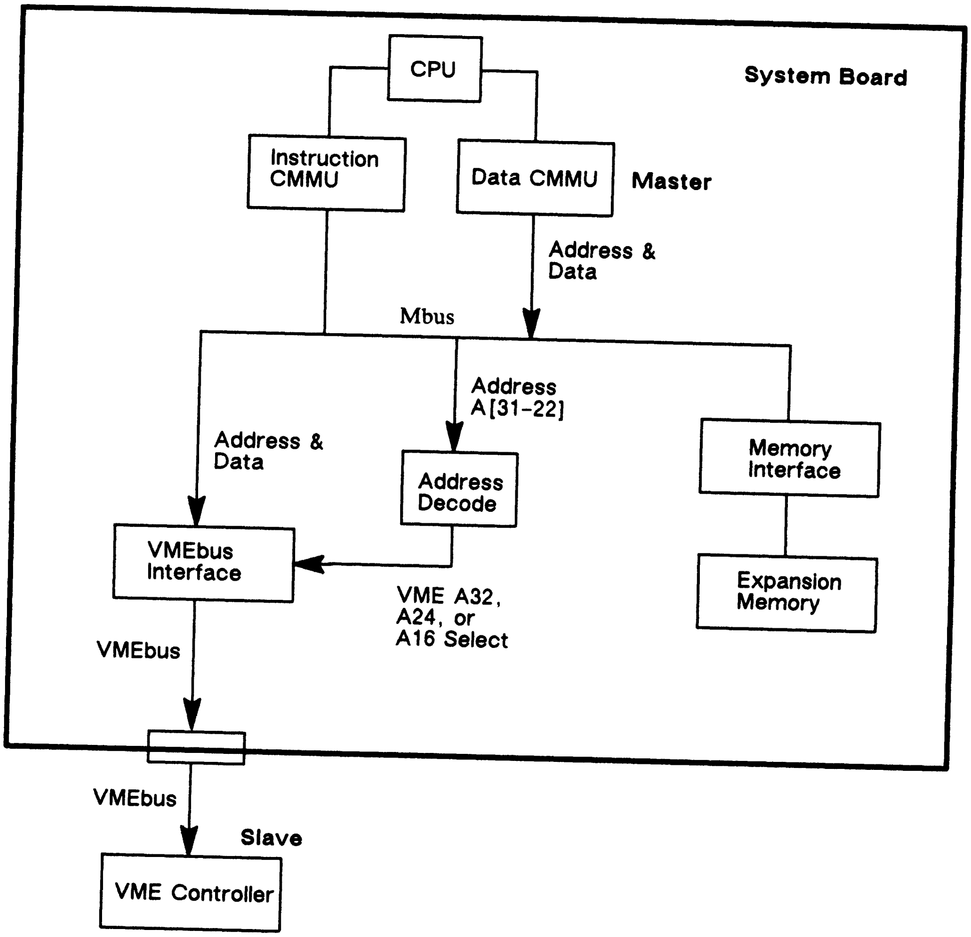


Figure 2–4 Addressing the VMEbus from the CPU

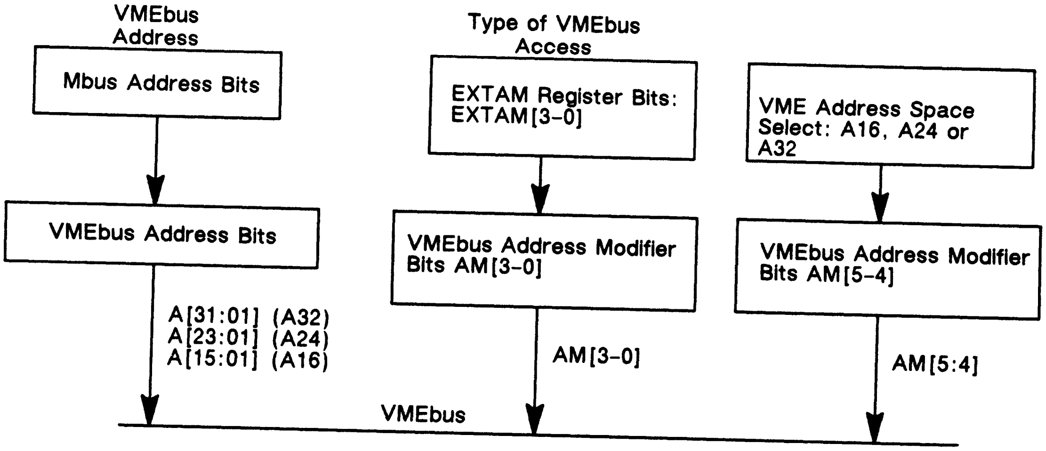

VMEbus Address
Mbus Address Bits
VMEbus Address Bits
A[31:01] (A32)
A[23:01] (A24)
A[15:01] (A16)
Type of VMEbus Access
EXTAM Register Bits: EXTAM[3-0]
VMEbus Address Modifier Bits AM[3-0]
AM[3-0]
VME Address Space Select: A16, A24 or A32
VMEbus Address Modifier Bits AM[5-4]
AM[5:4]
VMEbus

Figure 2–5 Decoding Addresses to the VMEbus

EXTAM Extended Address Modifier

Address FFF8 8014 Write only

The Extended Address Modifier (EXTAM) register supplies the address modifier bits AM[3-0] to the VMEbus when an Mbus controller accesses the VMEbus. The address modifier bits AM[5-0] supply the VME devices with information such as address size, type of access, and identification of the bus master. Address modifier bits AM[5, 4] identify the VME address space, and are decoded (by logic) from VME address space select as shown below. AM[5, 4] is put on the VMEbus. Table 2-3 defines the address modifier (AM[5, 4]) bits.

Table 2-3 Address Modifiers (VME Space)

VME Space	AM5	AM4
A32	0	0
A24	1	1
A16	1	0

Table 2-4 defines the address modifier (AM[3-0]) bits.

Table 2-4 Address Modifiers (Transfer Type)

AM3	AM2	AM1	AM0	Type of Access
1	1	1	1	Supervisor block transfer
1	1	1	0	Supervisor program access
1	1	0	1	Supervisor data access
1	0	1	1	User block transfer
1	0	1	0	User program access
1	0	0	1	User data access

NOTE: A user transfer or user access is the same as a nonpriveleged access defined in the VMEbus specification.

EXTAM is not affected by system reset or local reset.

Initialize EXTAM before addressing a VME controller from the system board.

7 4	3 0
Unused	EAM

Bit	Mnemonic	Function
7-4	Unused	
3-0	EAM[3-0]	Extended Address Modifiers. Drives the address modifiers AM[3-0] onto the VMEbus when a CPU writes an address to the VMEbus. The address modifier defines the type of transfer or access (see Table 2-3, Address Modifiers).

Addressing System Board Resources from a VME Controller (400 Series Only)

This section describes how VME controllers address system memory. This addressing process is invisible to the programmer. The following text and illustrations explain how the CPUs access the system board resources and the system memory.

VME controllers have one of three address spaces: A16, A24, or A32 space, and are respectively referred to as A16, A24, or A32 controllers. The number of address lines driven by the controller defines the address space or range available.

A16 controllers have 16 address lines in and out, and therefore their addressing range is limited to 64 Kbytes. A16 controllers can access other VME controllers, but they cannot access the system board.

A24 controllers have 24 address lines in and out, and therefore their addressing range is limited to 16 Mbytes. A24 controllers can access assigned system memory only after the CPU has instructed them to do so. When an A24 controller accesses system memory, the EXTAD register appends the upper eight address bits to the A24 address. This enables the A24 device to address the correct pages in memory. When addressing system memory, the address modifier bits AM[3–0] define the type of access: whether the transfer is to user space or to supervisor space, and whether these accesses are block, program or data accesses.

A32 controllers have 32 address lines in and out, and therefore their addressing range extends throughout the limits of the system. A32 controllers can access assigned system memory as well as other VME controllers.

Figure 2–6 illustrates how VME controllers address system memory.

A VME A24 Address to System Memory

31 24	23 16
EXTAD	24-bit address

15 0
24-bit address

A VME A32 Address to System Memory

31 22	21 16
Address of 4-Mbyte page	Address within 4-Mbyte page

15 0
Address within 4 Mbyte page

Figure 2–6 Structure of Addresses from VME Controllers to System Memory

Figure 2–7 is a flowchart that illustrates how VME controllers address system board resources and system memory.

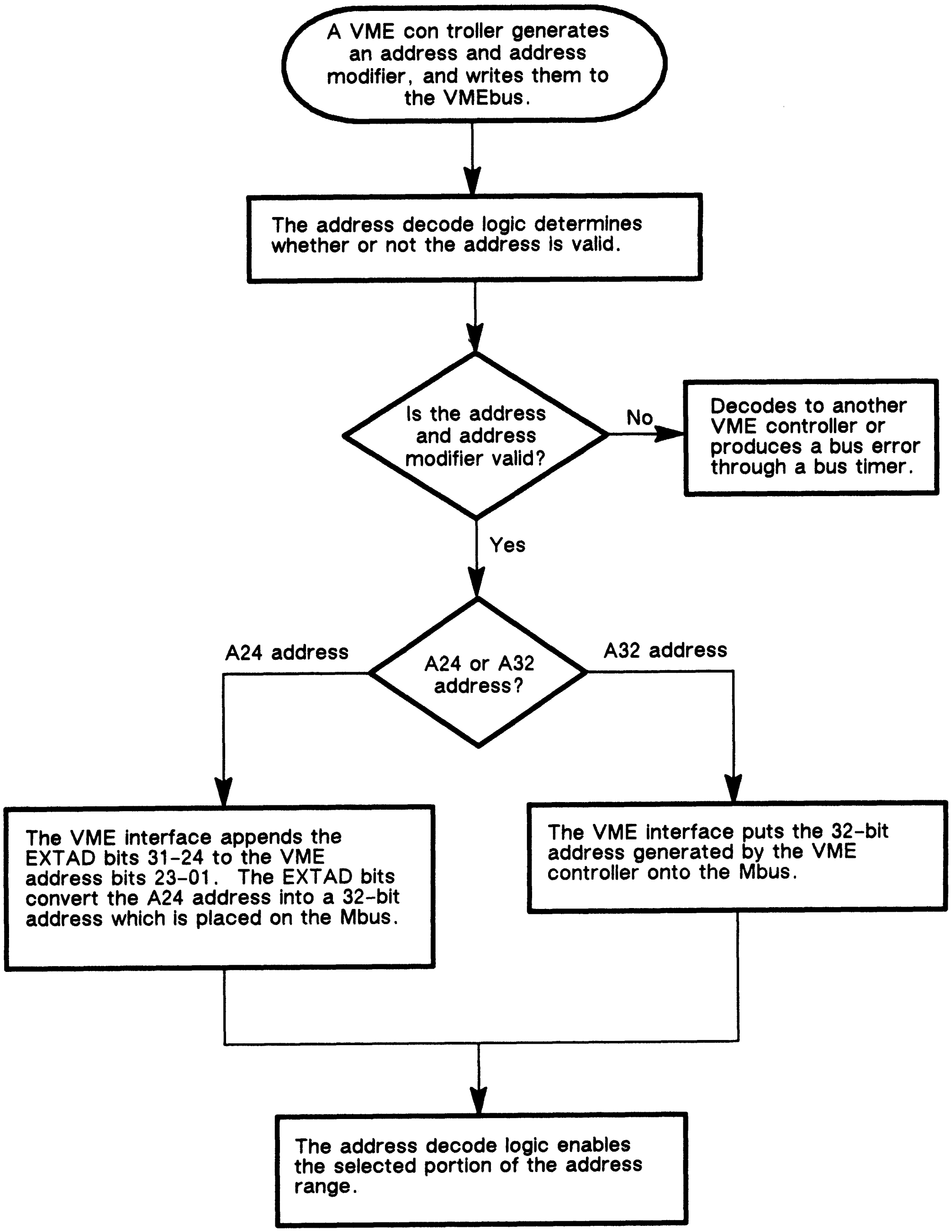

A VME con troller generates an address and address modifier, and writes them to the VMEbus.
The address decode logic determines whether or not the address is valid.
Is the address and address modifier valid?
No
Decodes to another VME controller or produces a bus error through a bus timer.
Yes
A24 address
A24 or A32 address?
A32 address
The VME interface appends the EXTAD bits 31-24 to the VME address bits 23-01. The EXTAD bits convert the A24 address into a 32-bit address which is placed on the Mbus.
The VME interface puts the 32-bit address generated by the VME controller onto the Mbus.
The address decode logic enables the selected portion of the address range.

Figure 2–7 How a VME Controller Addresses System Board Resources and System Memory (Flowchart)

Figure 2–8 illustrates how VME controllers address system board resources and system memory.

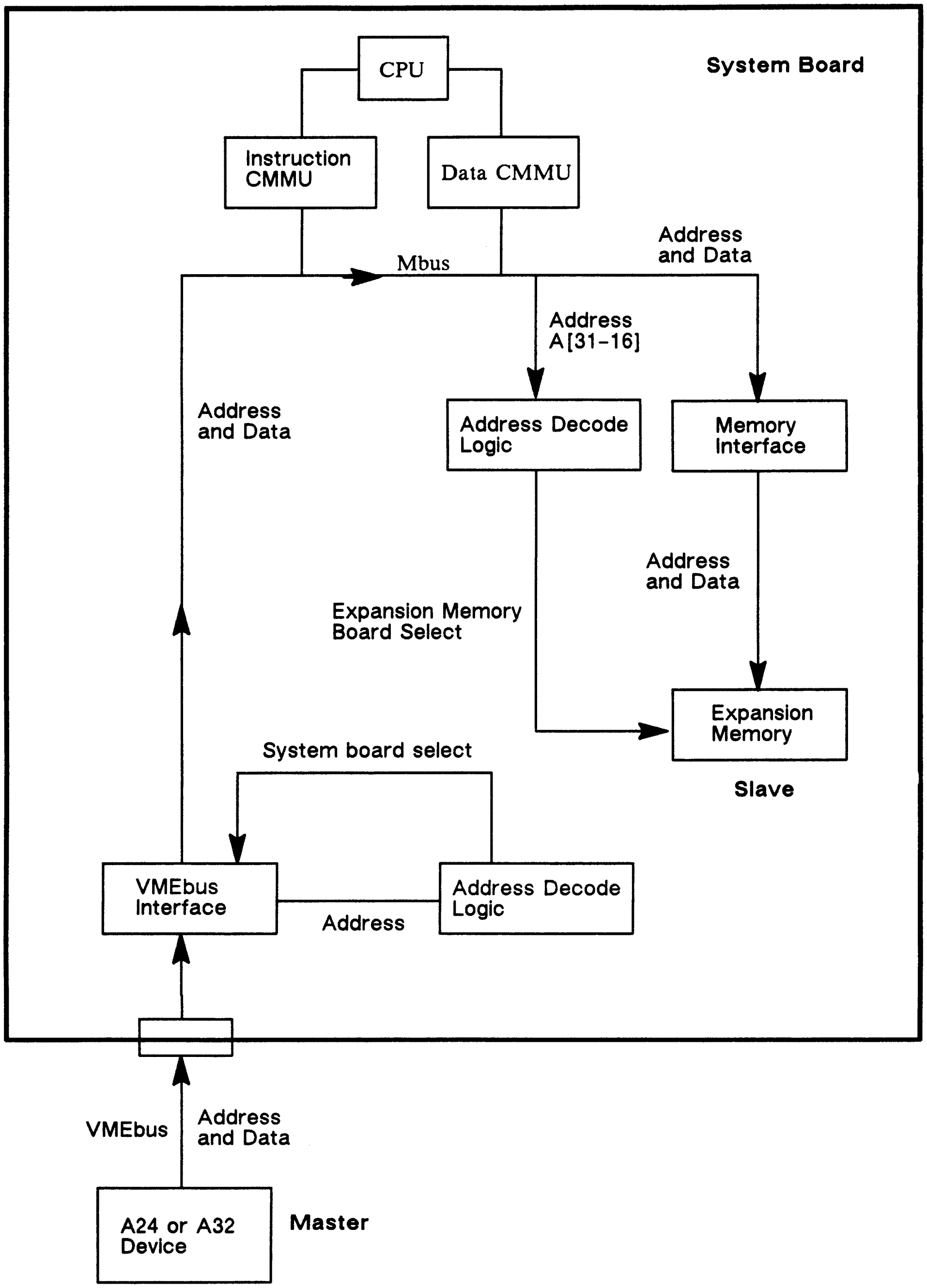


Figure 2–8 How a VME Controller Addresses System Board Resources and System Memory

EXTAD Extended Address

Address FFF8 8010 Write only

The extended address register provides the upper eight Mbus address bits when an A24 VMEbus device accesses the Mbus. EXTAD is loaded during powerup with a base address for VME access to system memory.

EXTAD is not affected by either system reset or local reset.

7	0
EXT	

Bit	Mnemonic	Function
7-0	EXT[31-24]	Extended Address. Supplies the address bits A[31-24] to the Mbus when an A24 VMEbus device accesses the Mbus.

Programming the System Control Registers

The system control registers are memory-mapped and are accessed as 32-bit registers. To access a system control register declare it as type **int** in a C program, or use the Load Register from Memory (**ld**) or Store Register to Memory (**st**) instruction in an assembly language program. Table 2-5 provides the memory map for these registers. The rest of this section describes the diagnostic registers.

Table 2-5 Memory Map of the System Control Registers

Register or Component	Address (hexadecimal)	Type
Common to both Workstations		
DCR (Diagnostic Control)	FFF8 4008 (300 series stations) FFF8 40C0 (400 series stations)	Write-only
DSR (Diagnostic Status)	FFF8 400C (300 series stations) FFF8 40C4 (400 series stations)	Read-only
PAR (Parity Address)	FFF8 4010 (300 series stations) FFF8 40C8 (400 series stations)	Read-only
400-specific		
WHOAMI (CPU Configuration)	FFF8 8018	Read-only

DCR Diagnostic Control

Address FFF8 4008 (300 stations) **Write Only**

Address FFF8 40C0 (400 stations) **Write Only**

The Diagnostic Control Register (DCR) allows software to control the state of the system board.

NOTE: All DCR bits are cleared to 0 after a power-on reset and system reset.

System components, such as the graphics subsystem, that reside on the Mbus must provide their own software reset capabilities beyond the power-on reset and software reset functions provided by the system board.

15 9	8	7	6	5	4	3	2	1	0
Unused	MSS	LED	RKB	LRD	FL1	RIO	RET	RSC	RSY

Bit	Mnemonic	Function
15–9	Unused	Unused
8	MSS	Monitor Speed Select (Available only on color graphics systems.) 1 Selects a 125 MHz video monitor. 0 Selects a 107 MHz video monitor.
7	LED	Diagnostic LED control. 1 Turns off the diagnostic LED. 0 Turns on the diagnostic LED.
6	RKB*	Reset keyboard subsystem. 0 Resets the keyboard interface. For information on the state of the keyboard interface after reset, see Chapter 6 "Programming the Keyboard Port." After a power-on reset or system reset, this bit is 0, holding the keyboard interface in the reset state.
5	LRD	Disable/enable low memory decoding of utility space. 1 Maps the utility space to the top of memory beginning at address FFC0 0000. LRD is set after the system is powered up. When mapped to this area, data can be read from and written to memory. 0 Maps the utility space to the bottom of memory, beginning at address 0000 0000. LRD is cleared during a powerup or reset so that the CPU can execute the power-up code beginning at address 0000 0000. When in this mode, the upper ten address bits (A[32:23] are ignored. Data can be written to but not read from memory.

* Signal is a logical true (1) when asserted low.

(continued)

Bit	Mnemonic	Function
4	FL1	Force a logical 1 into the parity bits. 1 Writes a logical 1 to all four byte parity bits when data is written to main memory. The data written to each byte determines whether a parity error is signaled when the data is read from main memory. Since there is no effect during a read operation, this allows you to force a parity error on any byte. For example, writing 0101 0100 produces a parity error for byte 0. In this case, byte 0 is the only byte to have odd parity when all parity bits are forced to 1s (the workstation uses even parity). Writing FFFF FFFF forces all bytes to have a parity error. Memory does not check for valid parity and treats parity bits for data the same as data bits. It is the responsibility of the Mbus master to check for parity errors.
3	RIO*	I/O subsystem reset. 0 Resets the mouse and serial ports' DUART and the parallel port. For information on the state of the DUART and the parallel port after reset and the exact state of the various subsystems after reset, see Chapter 7 "Programming the Serial Ports and Parallel Port." After a power-on reset or system reset, this bit is set to 0, holding the DUART and parallel port in the reset state.
2	RET*	Ethernet subsystem reset. 0 Resets the LAN interface. For information on the state of the LAN interface after reset, see Chapter 8, "Programming the Local Area Network Interface." After a power-on reset or system reset, this bit is set to 0, holding the LAN interface in the reset state.
1	RSC*	SCSI subsystem reset. 0 Resets the SCSI port. For information on the state of the SCSI port after reset, see Chapter 9, "Programming the Small Computer Systems Interface Port." After a power-on reset or system reset, this bit is set to 0, holding the SCSI port in the reset state.
0	RSY	System reset. 1 Initiates a system reset that puts the entire board, including the CPU, in the reset state. The power-on reset bit (STS) in the Diagnostic Status Register (DSR) is not set to 1, indicating that the reset was initiated by software and not by system powerup.

* Signal is a logical true (1) when asserted low.

(concluded)

DSR Diagnostic Status

Address FFF8 400C (300 stations) Read Only

Address FFF8 40C4 (400 stations) Read Only

The Diagnostic Status Register (DSR) allows software to determine the state of the system board.

7 3	2	1	0
Reserved	RST	TPR	STS

Bit	Mnemonic	Function
7–3	Reserved	Ignored when the register is written, and undefined when the register is read.
2	RST*	NOVRAM reset. 1 Indicates that the hardware jumper at locations TP2 and TP3 is not installed. This jumper is not normally installed. 0 Indicates that the jumper is installed so that firmware can initiate a NOVRAM initialization sequence.
1	TPR	SCSI terminator power. 1 Indicates that the onboard SCSI terminators have power. 0 Indicates that the fuse is blown and should be replaced.
0	STS	Power-on reset. 1 Indicates that the last reset was a system reset initiated by a power-on reset. When the DSR is read, this bit is set to 0, and is not reset to 1 until the next power-on reset. It allows diagnostic and power-up software to distinguish between a power-on reset and a software initiated reset.

* Signal is a logical true when asserted low.

PAR — Parity Address

Address FFF8 4010 (300 stations) — Read Only

Address FFF8 40C8 (400 stations) — Read Only

The Parity Address Register (PAR) monitors the addresses of each Mbus cycle and supplies address and strobe information. A parity error interrupt is generated only if the SCSI port or the LAN interface reads a memory location that contains bad parity during a DMA transfer or if the CPU reads the SCSI Signal register (Register 9) while it is in transition. All other parity error conditions are handled by the bus fault mechanism when one of the CMMU chips is the bus master.

If the Mbus cycle is positively acknowledged, the latched information is discarded, and the next Mbus cycle is monitored. If the cycle generates an Mbus parity interrupt, the latched address is stored in PAR until the register is read. After PAR is read, addresses can be latched again. Reading PAR clears the parity error interrupt request in the appropriate interrupt status register, and resets any pending parity error.

15	14	13	12 11	10	9	8	7	6	5	4 0
MS1	MS0	PH	Reserved	A1	PB1	PB0	100	MS2	DSP	MMAD

Bit	Mnemonic	Function
15	MS1	Memory Size 1
14	MS0	Memory Size 0 Together these bits indicate the number of memory modules with 4-Mbit DRAMs.
13	PH	Phase 1 The parity bus error terminated an address phase. 0 The parity bus error terminated a data phase. In the workstation's current implementation this bit is always 0.
12,11	Res	Reserved. Ignored when written to, returns 0s when read from.
10	A1	Half-word Address. This bit, along with the latched byte strobes, indicates which byte caused the parity error interrupt. 1 Indicates that the error occurred on either or both bytes in the bottom half-word (bits 15-0). 0 Indicates that the error occurred on either or both bytes in the top half-word (bits 31-16).

(continued)

Bit	Mnemonic	Function
9, 8	PB1, PB0	Parity Error Byte. These bits indicate which byte strobes were active during a bus cycle that was terminated by an Mbus parity error. PB0 indicates that the error occurred in the most-significant byte (either bits 7-0 or bits 23-16). PB1 indicates that the error occurred in the least-significant byte (either bits 15-8 or bits 31-24). Use PB0 and PB1 in conjunction with A1 (bit 10) to determine which byte caused the parity error.
7	100	100 ns DRAM indicator. 0 indicates that the memory modules use 100 ns DRAMs.
6	MS2	Memory Size 2 DRAM chips installed in the workstation (1-Mbit DRAM chips do not affect this count). The bits are binary encoded with MS0 being the least significant bit. Modules with 4-Mbit chips must be installed starting at physical address 0000 000016, and must occupy contiguous physical memory space. These bits count down from 0. For example, 1112 indicates 0 modules installed; 1102 indicates 1 module installed.
5	DSP	Parity Error Address Space. 1 Indicates that the Mbus parity error was caused during an access of the lower 128 Mbytes of address space. 0 Indicates that the parity error occurred during an access to any address space above the lower 128 Mbytes.
4-0	MMAD	Memory Module Address. These bits identify the memory module containingthe RAM location that caused a parity error. Depending on which size DRAM is installed, each module contains 4-Mbyte or 16-Mbyte locations, each with 32 data bits and 4 byte parity bits. Software must use this information in conjunction with bits 15, 14, 6 (MS2-0) to determine the faulting memory module.

(concluded)

WHOAMI (400-Series Only) CPU Configuration

Address FFF8 8018 Read Only

The CPU Configuration (WHOAMI) register contains the CPU configuration and defines which CPU is currently master of the Mbus. The CPU configuration defines how many CPUs and CMMUs are on the system board. The possible configurations are given by the CPU Configuration (CPC) bits. The WHOAMI register is unaffected by either system reset or a local reset.

7 4	3 0
CPC	CMM

Bit	Mnemonic	Function
7–4	CPC	CPU Configuration. These bits define the current CPU configuration as follows: **CPC** **CPU Configuration** 516 2 CPUs, 4 CMMUs A16 1 CPU, 2 CMMUs
3–0	CMM	Current Mbus Master The CMM bits indicate which CPU is currently the master of the Mbus during a data cycle. These bits are valid only during a data cycle when a CPU is initiating the read or write through its data CMMU. CMM is not valid during an instruction read or write, or while a VME controller is master of the Mbus. **CMM** **Mbus Master** 1 CPU0 is master of the Mbus. 2 CPU1 is master of the Mbus.

The Time-of-Boot (TOB) Clock and Nonvolatile RAM (NOVRAM)

The time-of-boot (TOB) clock is a battery-backed device that initializes the time-of-day clock. The TOB clock is implemented using a SGS-Thomson MK48T02B 2K X 8 Zeropowert/Timekeepert RAM chip. This chip provides total nonvolatility for over 3 years of operation without power. Real-time clock is accurate to within one minute per month at room temperature (25_C). The device provides a programmable register for calibrating the timekeeping accuracy. In addition to the timekeeping functions, this chip provides 2040 bytes of NOVRAM used for diagnostics and system configuration/boot information.

Time-of-Boot Clock Registers

The eight TOB clock registers are 8-bit registers in the timekeeper RAM. Software can adjust the TOB clock by writing the desired time and date to these registers. They are aligned on word (32-bit) boundaries and occupy the last eight locations of the 2048 words reserved for the NOVRAM. Table 2-6 shows the registers, their addresses, and illustrates their use with an example.

Table 2-6 Memory Map for the Time-of-Boot Clock Registers

Address	Register	Range of Value Hex. (Decimal)	Example Hex. (Decimal)
FFF8 1FFC	Year	00-63 (99)	59 (89)
FFF8 1FF8	Month	01-0C (12)	08 (08)
FFF8 1FF4	Date	01-1F (31)	11 (17)
FFF8 1FF0	Day	01-07 (07)	05 (05)
FFF8 1FEC	Hour	00-17 (23)	10 (16)
FFF8 1FE8	Minutes	00-3B (59)	19 (25)
FFF8 1FE4	Seconds	00-3B (59)	29 (41)
FFF8 1FE0	Control	—	40 (64)

The example returns: Thursday, August 17, 1989, 16:25:41.

The Control register (FFF8 1FE0) allows you to read, set, or calibrate the TOB clock. Before reading or writing the TOB registers, you must prevent the clock from updating the registers by setting the appropriate bit: to read data set bit 6 of the Control register to 1, to write data set bit 7 to 1. Halting the register updating does not affect the TOB clock's timekeeping. Resetting the read bit to 0 continues updating of the registers; resetting the write bit to 0 transfers the register values to the TOB clock and continues the timekeeping operations. The date/time values occupy the low-order bits of the registers with any unused bits set to 0.

To calibrate the TOB clock, write a calibration value into bits 0-5 of the Control register. Bit 5 is the sign bit: 0 indicates negative calibration, slowing the clock down; 1 indicates positive calibration, speeding the clock up. Bits 0-4 contain a binary calibration value from 0-31. Each value modifies a number of minutes in each 64-minute cycle of the clock. A value of 000012 modifies the first 2 minutes by 1 second per minute. A value of 001102 modifies the first 12 minutes by 1 second per minute.

For detailed descriptions of these registers and further programming information, see the SGS–Thomson, *Memory Products Databook.*

NOVRAM Addresses

The NOVRAM contains 2040 bytes of nonvolatile data for Data General diagnostics and system configuration/boot information. The NOVRAM occupies two 4–Kbyte pages in the Mbus utility space. Each NOVRAM byte is word aligned in Mbus address space, and is accessed as the lowest order byte (bits 8–0) of each word. Table 2–7 gives the address map for the NOVRAM bytes.

Table 2–7 NOVRAM Addresses

Address	Byte Number	Comments

FFF8 0000
. . .
FFF8 01FF
	0000
	. . .
	0127
		Reserved (not write–protected)

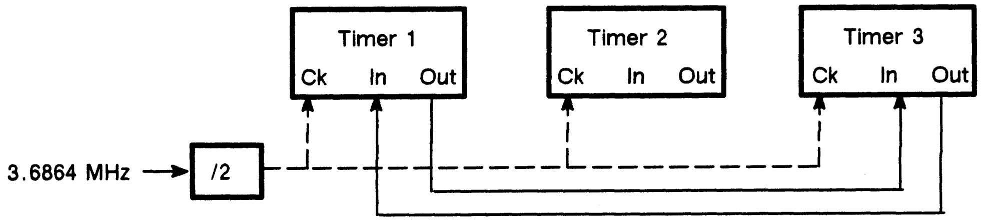
FFF8 0200
. . .
FFF8 03FF
	0128
	. . .
	0255
		Reserved (write–protected)

FFF8 0400
. . .
FFF8 1FDC
	0256
	. . .
	2039
		General use (not write–protected)

Programming the CIO

A Zilog Z8536 Counter/Timer and Parallel I/O (CIO) is used as a counter/timer only, generating an interrupt to the CPU when a countdown reaches terminal count. This process can be used to generate, through software, a time-of-day (TOD) clock.

The counter/timer lines can be enabled or disabled by programming the control registers. As shown in Figure 2-9, timers 1 and 3 are linked externally. To prevent false triggers, all of the counter/timer I/O pins are pulled High.



Ck = Clock Input
In = Counter Input
Ot = Counter Output

Figure 2-9 External Timer Connections

The 3.6864-MHz PCLK input is internally divided by 2 and supplied to the counters, providing a minimum resolution of 0.54 ms.

The counters can be programmed to generate an interrupt when they reach terminal count. The program must read the counter often enough to detect a roll-over; the counter does not generate an interrupt when it rolls over.

The 8-bit CIO registers are aligned on word boundaries and are the low-order byte. Table 2-8 defines the addresses of the CIO registers. For detailed descriptions of the CIO registers, see Zilog Corporation's manual *Z8536 Z-CIO/Z8536 CIO Counter/Timer and Parallel I/O Unit.*

Table 2-8 CIO Register Addresses

Address	Register
FFF8 3000	Port A Data register
FFF8 3004	Port B Data register
FFF8 3008	Port C Data register
FFF8 300C	Control register

The Boot PROM

The boot PROM contains 128 Kbytes arranged as 32 Kwords by 32 bits/word. The boot PROM resides in the utility space at addresses FFC0 0000 through FFC1 FFFF. During a power–on reset or a system reset, the boot PROM is mapped to 0000 0000 through 0001 FFFF. This mapping allows the CPU to fetch the boot code from address 0000 0000.

NOTE: The remapping function is enabled by clearing bit 5 (LRD) of the Diagnostic Control Register (DCR). See the Diagnostic Control Register in an earlier section, "Programming the System Control Registers."

During powerup, the CPU executes the System Control Monitor (SCM) program to initialize and test the system board and memory modules. See the next section, "The System Control Monitor (SCM)." When this power–up sequence is completed, the system operator enters the BOOT command and the CPU boots the operating system from the default boot device. The boot device can be a local disk or tape drive or a server on Ethernet. For more information on the power–up sequence, see Appendix B, "Power–Up Flowchart."

If the CPU cannot boot the operating system from the default device, it runs the SCM user interface and displays the boot menu on the system console (either the graphics monitor or a terminal connected to the serial port). The boot menu allows a user to specify the boot path. For information on the boot menu, see the manual *Using the System Control Monitor (SCM).*

If performing an automatic program load, the load will not succeed unless a valid boot path exists and the boot device contains the appropriate boot file. For information on the boot file, see Appendix C, "Boot File Format."

During powerup, the CPU also enters the SCM user interface when a system error condition occurs or when one or more diagnostics test fails. During normal system operation, the CPU accesses the SCM user interface when the operating system encounters a problem that it cannot handle while running (i.e., a virtual halt or a breakpoint not supported by a user program.) The SCM user interface consists of a command interpreter and menus. These allow the system operator to display and modify system configuration parameters.

For more information on the SCM user interface, see the manual *Using the System Control Monitor (SCM).*

Power–Up and Boot Code

When a powerup reset occurs, the CPU executes power–up diagnostic code stored in the PROM. After the system hardware passes diagnostic tests, the CPU boots from a default storage device which may be on the Ethernet LAN, SCSI bus, or VMEbus (400 series stations only.)

The System Control Monitor (SCM)

Data General's System Control Monitor (SCM) supports a standard set of system calls that use CPU registers. Programs can pass control to and from the SCM using these optional system calls. Operating system software may need to support the SCM system calls for certain value-added functions.

The SCM currently provides the following services to the operating system via system calls:

- Access to standard input/output devices.
- System configuration information.
- Panic and error reporting.

Software accesses SCM system calls through vectors in the boot PROM vector space. A program must do the following to access the SCM system calls:

1 Set CR7, the Vector Base Register (VBR), to the PROM VBR. If this changes the value of CR7, software must save the changes to copy back later. The VBR defaults to the PROM values after powerup.

2 Load register 9 (r9) and other argument-specified registers with the offset value defined in Table 2-9 (the values are in hexadecimal unless indicated otherwise).

3 Execute the following trap instruction: **tb0 0,R0,496**

Table 2-9 SCM System Calls

System Call	Argument(s)	Data Returned	Function
.BANNER	r9=113	r2=Pointer to string	Returns pointer to system banner string.
.CHAR	r9=0	r2(LSB)=ASCII character	Waits for an ASCII character from the default input port, reads it, and returns the character in the least significant byte of register 2. (A null indicates a break.)

.CHFLOW	r9=116
	r2=0 or non-0
	r3=Value
		r2=Flow control flag	Reads or writes the character flow control (XON/XOFF) flag. If r2 = 0 initially, then r2 will contain the current flag value. If r2 = non-0 initially, then stores value from r3. An r3 value of F816 indicates flow control enabled; any other value indicates disabled.

.CHKSUM	r2=Pointer to data
	r3=Byte count
	r9=68
		r2=Checksum	Performs data checksum test and returns the value in r2. (Adds all the bytes and complements the result.)

.CHSTAT	r9=5	r2(LSB)=Default input status	Polls the standard input port for character status and returns this value in the least significant byte of r2.
.COMMID	r9=114	r2=Pointer to address	Returns pointer to Ethernet address.
.CPUID	r9=102	r2=CPU ID	Returns the CPU ID.

Note: If r2 is set to 1, an error has occurred.

(continued)

Table 2-9 SCM System Calls

System Call	Argument(s)	Data Returned	Function

.GDMP	r9=105
	r2=0 or non-0
		r2=Pointer	Reads video timing parameters into SPAD buffer, and returns pointer as byte-packed data in r2. If r2 original value not 0, writes byte-packed data pointed to by r2 into BBSRAM.

.GTLINE	r2=32-bit string buffer address r9=2	r2=String length	Reads a character string of 256 characters or less from the standard input port, echoes them, and places the character string in the buffer address in r2. Terminator characters are: \n = New Line, \l = Carriage Return, and \f = Formfeed. The SCM screen edit control functions are supported.
.HALT	r9=63	None	Halts the user program and enters the SCM.
.HOSTID	r9=107	r2=Pointer to host ID	Returns to r2 a pointer to 4-byte binary host ID data.

.INVALID	r9=112
	r2=JP# or -1
		None	Invalidates the instruction cache (Icache). If r2 = a JP number, then only that JP Icache is invalidated. If r2 = -1, then all JP Icaches are invalidated.

.KBLAN	r9=106	r2=Language	Returns language code to r2. The codes and languages are 1 U.S. English 6 Spanish 2 German 7 Swiss 3 U.K. English 8 Italian 4 French 9 Japanese 5 Swedish

.MSIZE	r9=103
	r2=0 or non-0
		r2=Top of memory	Returns top of memory to r2. If r2 = 0 initially; then r2 will contain top of physical memory. If r2 = non-0 initially, then r2 will contain top of user memory.

.NBLOCAL	r9=115
	r2=0 or non-0
	r3=Value
		r2=LAN port number	Reads or writes the LAN port number. If r2 = 0 initially, then r2 will contain the LAN port number. If r2 = non-0 initially, then stores value from r3.

.OCHAR	r9=20 r2(LSB)=ASCII character	r2=0	Prints the value in the least significant byte of r2 to the standard output device.
.OCRLF	r9=26	r2=0	Prints a Carriage Return/line feed to the standard output device.
.PANIC	r9=110	r2=Error code	Halts the user program, returns an error code to r2, and enters the SCM.
.PRINTER	r9=117	r2=Printer type	Returns printer type to r2. A value of 0 = Centronics; non-0 = Data Products.
.POLLKEY	r9=5	r2=Key hit	Returns an indication of whether or not a key was pressed. If r2 = 0, no key was pressed. If r2 = non-0, a key was pressed.
.PTLINE	r9=21 r2=32-bit address of string	r2=0	Prints the character string pointed to by the address in r2 to the standard output device. Does not return until it encounters the null terminator in the string. Note that this call allows five additional arguments and uses the C **printf** characteristics.

Note: If r2 is set to 1, an error has occurred.

(continued)

Table 2-9 (continued) SCM System Calls

System Call	Argument(s)	Data Returned	Function
.REBOOT	r9=101 r2=0 or Pointer to boot path	None	Resets and reinitializes the workstation, initializes the boot time registers, and enters the boot menu. If r2 = 0, the call uses the default boot path. If r2 = non-0, the call uses the pointer in r2.
.REVNUM	r9=104	r2=Revision number	Returns PROM revision to r2 in the format: bit 31 (if 1), engineering revision; bits 30-16, major revision number; bits 15-0, minor revision number. For example, 80050002 = Rev E05.02 30000 = Rev 3.0

.RWDCR	r9=111
	r2=0 or non-0
	r3=New DCR value
		r2=DCR	Reads or writes a copy of the Diagnostic Control Register (DCR) word in memory. If r2=0, returns DCR to r2. If r2 = non-0, writes r3 value to DCR word. The description of the Diagnostic Control Register (DCR) in this chapter gives the DCR values.

.JPSTART	r2=JP# to start
	r3=Starting address
	r9=100
		r2=Status	Starts another processor (JP#) after an initial boot (used only in multiprocessor systems). The status returned to r2 is
			0 Start successful
			1 Illegal or missing JP
			2 Single JP configuration
			3 JP not halted
			4 JP does not respond

.STDIO	r9=70	r2=I/O device number	Returns the standard input and output ports. Device number values are 0 Serial input and output 1 Serial input/serial and graphics output 2 Keyboard input/graphics output

.TECW	r9=108
	r2=0 or non-0
	r3=New ECW value
		r2=ECW	Returns or sets Environment Control Word (ECW). If r2 = 0, returns ECW to r2. If r2 not = 0, writes r3 value to ECW. Table 2-8 lists the ECW bit values, functions, and default states at powerup.

Note: If r2 is set to 1, an error has occurred.

(concluded)

Table 2–10 defines the contents of the Environment Control Word.

Table 2–10 Environment Control Word (ECW) Contents

Bit	Function	State at Powerup
0	Reserved	0
1	Loop on error 0 Disables testing when program encounters an error. 1 Continues testing when program encounters an error.	0
2	Output to console 1 Directs program output to the system console.	1
3	Percent failure 0 Disables reporting of this error. 1 Enables reporting of percent of errors after looping (errors per total number of loops). Note that bit 1 (loop on error) must also be enabled.	0
4	Print pass messages 0 Disables printing of message. 1 Enables printing of messages to the system console after each test pass completes.	1
5	Output to printer 0 Disables output to printer. 1 Enables program output to the default printer port.	0
6	Disassembler 1 0 Disables display. 1 Enables displaying an additional output field that contains the mnemonics of memory address contents.	
7	Print subtest message 0 Disables printing message. 1 Enables printing subtest messages to the system console.	0
8	Report all 1 0 Print brief messages to the system console. 1 Print verbose messages to the system console.	
9	Halt on error 0 1 Enables halting the program after an error and returning the SCM prompt.	
10	Enable error logging 0 Disables error logging. 1 Enables recording all errors in system error log.	0
11, 12	Reserved	0
13	Page mode 0 0 Disables Page mode. 1 Enables displaying output on the system console one screen (page) at a time	
14–31	Reserved	0

In addition to the system calls, the System Control Monitor (SCM) supports hardwired entry points to the subroutines in Table 2–11 (accessible with a **jsr** instruction containing the appropriate entry point).

Table 2–11 SCM Subroutines

Entry Point (Hex)	Subroutine	Argument	Description
1000	putchar to stdio	r2=char	Outputs the character in r2.
1004	getchar from stdio	r2=char	Returns a character to r2.
1008	KBD_reset	r2=0	Performs a keyboard hard reset.
100C	GDM_reset	r2=0	Performs a graphics display monitor hard reset.
1010	GDM_load_fonts	—	Loads the fonts for the graphics display monitor.
1014	GDM_putchar	r2=char	Outputs the character in r2 to the graphics display monitor.

End of Chapter

Chapter 3
Interrupts, System Errors, and Bus Faults

This chapter discusses the following topics:

- Types of interrupts.
- How the interrupting devices interrupt the CPU.
- How the CPU handles the interrupts.

The workstation has two error reporting mechanisms: bus faults and interrupts. The CMMUs inform the CPU of bus faults, and the interrupt control logic informs the CPU of interrupts generated by the various subsystems.

Interrupts are a means for various system resources (memory, I/O controllers, power supply, etc.) to notify the CPU of a condition that needs attention. Each interrupt has an associated interrupt service routine that the CPU executes. Some interrupts represent a specific interrupt condition (condition–specific interrupts), while others represent one of many possible interrupt conditions (multiple–use interrupts). For specific interrupts, the operating system developer may use a table to associate an interrupt vector with the interrupt. For the multiple–use interrupts, the interrupting device must supply an interrupt vector to the system board CPU.

Interrupt control logic provides the CPU with interrupt information. When devices assert their interrupt request, the interrupt control logic first performs a logical AND of the interrupt requests with the contents of the interrupt mask register; then it asserts the interrupt line (INT) to the CPU. Workstations with two CPUs have two interrupt lines (INT0 and INT1), one for each CPU. In these systems, the interrupt control logic asserts the appropriate line or lines, depending on the masks.

The interrupt service routine reads the interrupt status register (ISR or IST) and if necessary the interrupt enable registers (IENn); then it isolates the interrupt(s).

The device faults for the I/O subsystems are discussed in the related chapters as follows:

Chapter 6 “Programming the Keyboard Interface and Speaker”

Chapter 7 “Programming the Serial Ports and Parallel Port”

Chapter 8 “Programming the Local Area Network Interface”

Chapter 9 “Programming the Small Computer System Interface”

Types of Interrupts

Interrupts fall into one of two categories: condition–specific interrupts and multiple–use interrupts.

Condition–Specific Interrupts

Condition–specific interrupts span much of the system, including all of the local system board interrupts and many VME interrupts. These interrupts represent specific conditions such as the depressing of the abort switch or the occurrence of a single–bit memory read error.

300 Series Interrupts

The condition–specific interrupts include Powerfail (PF), Parity Error (PE), CIO Interrupt (CI), Keyboard Interrupt (KB), DUART Interrupt (DU), Parallel Port Interrupt (PP), Ethernet Interrupt (ET), SCSI Interrupt (SC), SCSI DMA Terminal Count (DT), SCSI DMA Write Protect Error (DW), SCSI DMA Valid BIT (DV), Graphics Interrupt (GI), and Software Interrupt (SI). Of these, the DUART Interrupt (DI) and the CIO Interrupt (CIO) may be one of several possible, but specifically defined, interrupts from the related device.

400 Series Interrupts

The condition–specific interrupts include Abort Pushbutton (ABT), AC Failure (ACF), Bus Arbiter Timeout (ATO), Z–Buffer Interrupt (ZBF), Video Interrupt (VDI), Parity Error (PAR), Keyboard Interrupt (KBD), CIO Interrupt (CIO), System Failure (SF), Parallel Port Interrupt (PPI), DUART1 Interrupt (DT1), DUART2 Interrupt (DT2), Ethernet Controller Interrupt (ECI), DMA Terminal Count (DTC), DMA Write Protect Error (DWP), DMA Valid Bit (DVB), and SCSI Controller Interrupt (SCI). Of these, the DUART Interrupts (DI1 and DI2) and the CIO Interrupt (CIO) may be one of several possible, but specifically defined, interrupts from the related device.

Multiple–Use Interrupts (400 Series Only)

Multiple–use interrupts include Signal High Priority (SHP), Signal Low Priority (SLP), Software–Generated Interrupts (SI[7-0]), and VME Interrupts (IR[7–1]). VME interrupts are generated by VME controllers through seven interrupt request lines (IRQ[7–1]*) on the VMEbus. A VME controller can choose which interrupt line to use when it has an interrupt condition that requires servicing by the system board. These interrupt lines are not limited to specific interrupts; any serviceable interrupt can be generated through the VME interrupt request lines. To execute the correct interrupt service routine, the system board CPU must obtain the interrupt vector from the interrupting VME controller.

Besides being interrupted, the system board CPU can initiate interrupts to the VME controllers using the VME–level interrupts. The CPU, when it interrupts a VME controller, must define the interrupt level and provide the VME controller with an interrupt vector. This process is described later in this chapter in the section "Interrupting a VME Controller."

How the CPU Is Interrupted

This section describes how all interrupt requests are passed to the CPU from interrupting devices throughout the system.

Interrupts originate from system board controllers and logic (i.e., the SCSI interface, LAN interface, parity logic, address decode logic, etc.). In addition, in 400 series stations, the VME controllers generate interrupts to the system board.

When a device requires servicing by an interrupt service routine, the device asserts an interrupt request. The system board contains interrupt logic that checks the interrupt request lines and processes incoming interrupts. When an interrupt is received, the interrupt logic sets the appropriate bit in the Interrupt Status register, and asserts the Interrupt (INT) line to the CPU. In systems with more than one CPU, the INT line is multiplexed, with one interrupt line per CPU. Interrupts originate from VME controllers as well as the onboard Ethernet LAN, SCSI and Serial Interface controllers.

The interrupt control logic monitors and processes interrupt requests. The interrupt logic sets the appropriate bit(s) in the Interrupt Status (IST) register, compares the IST register with the Interrupt Enable (IENn) registers, and when a bit in IST is set and not masked by an IENn register, the logic asserts the interrupt line (INT) to the CPU associated with that IENn register.

Handling Interrupts

This section discusses some dynamics of developing software to service interrupts. Because the CPU has one interrupt line (INT) to notify it of a pending interrupt, all interrupts are stored in registers which the CPU reads and decodes. Figure 3–1 shows how interrupts are handled by a system that has one CPU on the system board.

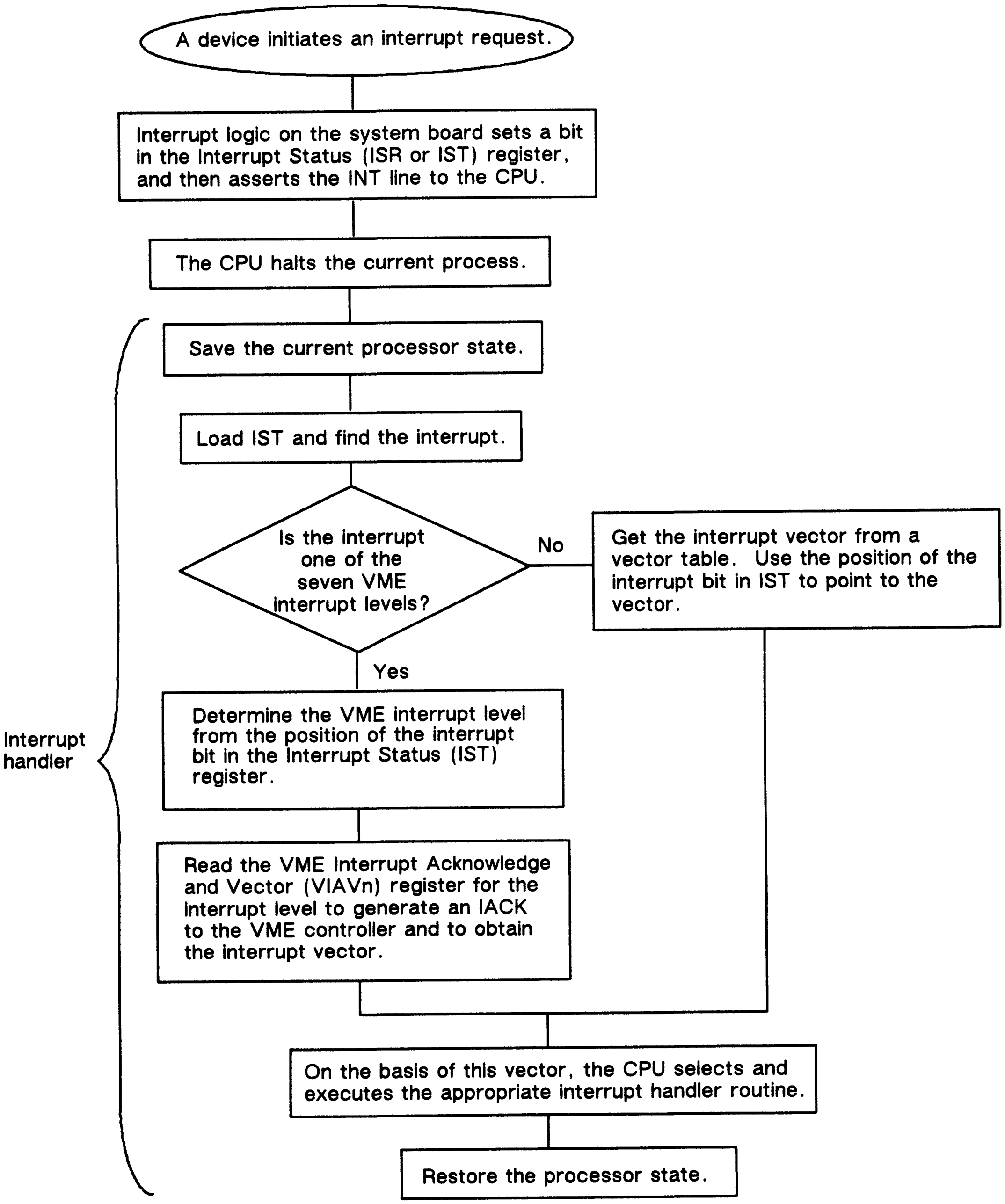

A device initiates an interrupt request.
Interrupt logic on the system board sets a bit in the Interrupt Status (ISR or IST) register, and then asserts the INT line to the CPU.
The CPU halts the current process.
Save the current processor state.
Load IST and find the interrupt.
Is the interrupt one of the seven VME interrupt levels?
No
Get the interrupt vector from a vector table. Use the position of the interrupt bit in IST to point to the vector.
Yes
Determine the VME interrupt level from the position of the interrupt bit in the Interrupt Status (IST) register.
Read the VME Interrupt Acknowledge and Vector (VIAVn) register for the interrupt level to generate an IACK to the VME controller and to obtain the interrupt vector.
On the basis of this vector, the CPU selects and executes the appropriate interrupt handler routine.
Restore the processor state.
Interrupt handler

Figure 3–1 Handling Interrupts with a Single-CPU System Board

Figure 3–2 illustrates how to service interrupts in a system that has two CPUs on the system board.

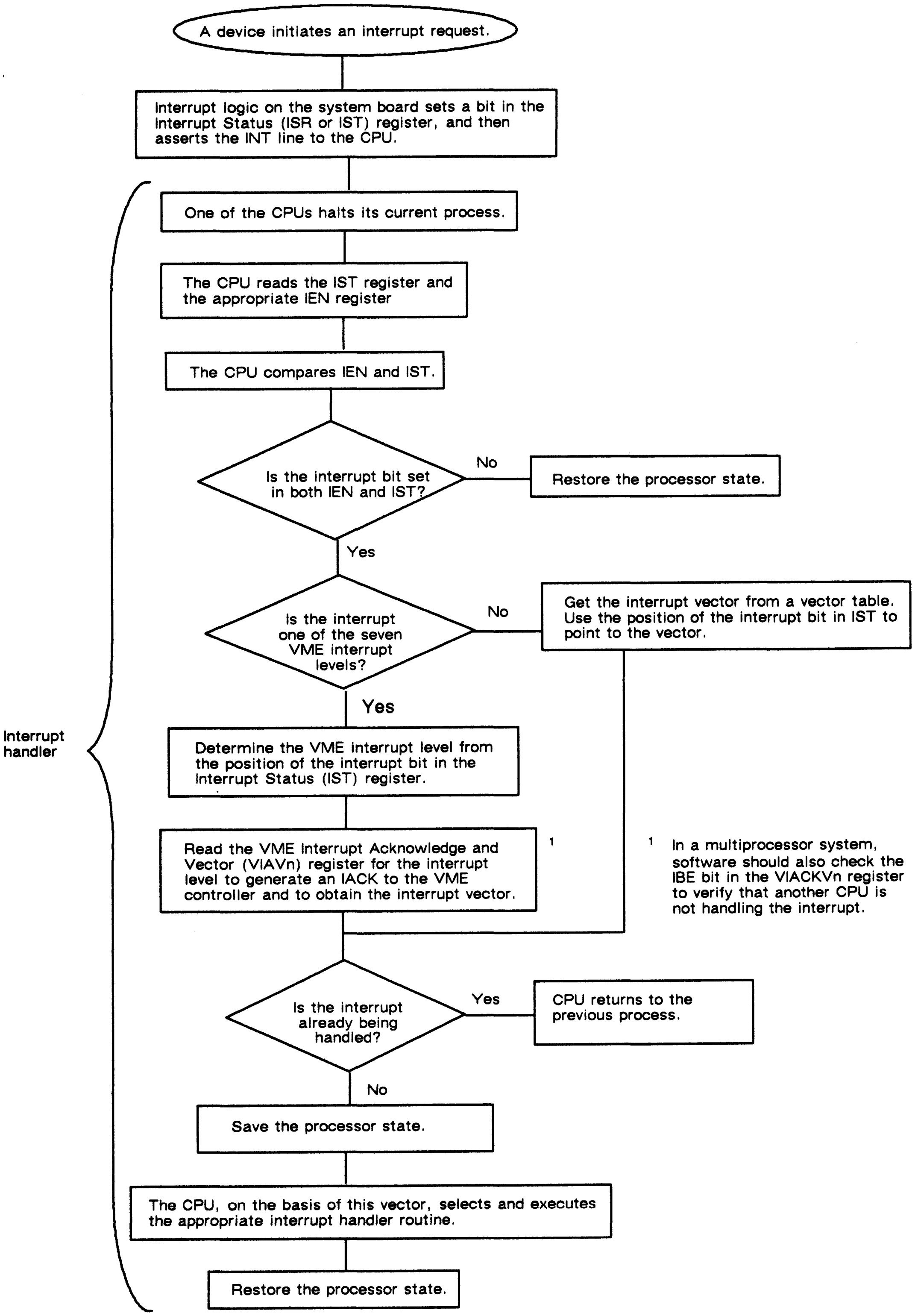

A device initiates an interrupt request.
Interrupt logic on the system board sets a bit in the Interrupt Status (ISR or IST) register, and then asserts the INT line to the CPU.
One of the CPUs halts its current process.
The CPU reads the IST register and the appropriate IEN register
The CPU compares IEN and IST.
Is the interrupt bit set in both IEN and IST?
No
Restore the processor state.
Yes
Is the interrupt one of the seven VME interrupt levels?
No
Get the interrupt vector from a vector table. Use the position of the interrupt bit in IST to point to the vector.
Yes
Interrupt handler
Determine the VME interrupt level from the position of the interrupt bit in the Interrupt Status (IST) register.
Read the VME Interrupt Acknowledge and Vector (VIAVn) register for the interrupt level to generate an IACK to the VME controller and to obtain the interrupt vector.
1
1 In a multiprocessor system, software should also check the IBE bit in the VIACKVn register to verify that another CPU is not handling the interrupt.
Is the interrupt already being handled?
Yes
CPU returns to the previous process.
No
Save the processor state.
The CPU, on the basis of this vector, selects and executes the appropriate interrupt handler routine.
Restore the processor state.

Figure 3–2 Handling Interrupts with a Dual–CPU System Board

Programming the CPU Interrupt Registers

The interrupt registers are memory mapped 32-bit registers. To access these registers from a C program, declare them as type **int**. To access them from an assembly language program, use the Load Register from Memory (**ld**) or Store Register to Memory (**st**) instruction. Table 3–1 is a memory map for the interrupt registers.

Table 3–1 Memory Map of the Interrupt Registers

Register or Component	Address	Type
300 series registers		
ISR (Interrupt Status)	FFF8 4000	Read/Write
IER (Interrupt Enable)	FFF8 4004	Write-only
SWIR (Soft Interrupt)	FFF8 4014	Write-only
400 series registers		
IEN0 (Interrupt Enable CPU0)	FFF8 4004	Write-only
IEN1 (Interrupt Enable CPU1)	FFF8 4008	Write-only
IST (Interrupt Status)	FFF8 4040	Read-only
SETSWI (Set Software Interrupt)	FFF8 4080	Write-only
CLRSWI (Clear Software Interrupt)	FFF8 4084	Write-only
ISS (Interrupt Source Status)	FFF8 4088	Read-only
CLRINT (Clear Interrupt)	FFF8 408C	Write-only

300 Series CPU Interrupt Registers

This section describes the CPU interrupt registers in 300 series stations.

ISR (300 Series Only) **Interrupt Status**

Address FFF8 4000 **Read/Write**

The Interrupt Status Register (ISR) identifies the interrupts that are currently active. Devices with the strict latency requirements are assigned to the highest order ISR bits, but the interpretation of interrupt priority is left to the system software.

ISR and IER have the same bit assignments. ISR defines the current state of all interrupt requests, and IER enables or masks the interrupts recorded in ISR.

Since the ISR bits reflect the state of interrupt requests, they are not directly affected by a system reset. Software must clear individual interrupts only by clearing the source of the interrupt in the specific I/O device.

The ISR bits are defined as follows:

31	30	29	28 22	21	20	19	18	17	16
Res	PF	PE	Reserved	CI	Reserved	KB	DU	PP	Res

15	14	13	12	11	10	9	8 0
ET	SC	DT	DW	DV	GI	SI	Reserved

Bit	Name	Function
31	Reserved	Ignore this bit.
30	PF	Powerfail 1 Indicates that a power fail has occured. 0 No error.
29	PE	Parity error 1 Indicates that a parity error has occured. 0 No error.
28–22	Reserved	Ignore these bits.
21	CI	CIO interrupt 1 Indicates that the CIO has asserted an interrupt. 0 No error.
20	Reserved	Ignore this bit.
19	KB	Keyboard interrupt 1 Indicates that the keyboard controller has asserted an interrupt. 0 No interrupt.

(continued)

Bit	Name	Function
18	DU	DUART 1 Indicates that the DUART has asserted an interrupt. 0 No interrupt.
17	PP	Parallel port interrupt 1 Indicates that the parallel port has asserted an interrupt. 0 No interrupt.
16	Reserved	Ignore this bit.
15	ET	Ethernet interrupt 1 Indicates that the Ethernet controller has asserted an interrupt. 0 No interrupt.
14	SC	SCSI controller interrupt 1 Indicates that the SCSI controller has asserted an interrupt. 0 No interrupt.
13	DT	SCSI DMA terminal count reached 1 Indicates that the SCSI terminal count has been reached. 0 No interrupt.
12	DW	SCSI DMA write-protect error 1 Indicates that a SCSI DMA write-protect error has occurred. 0 No interrupt.
11	DV	SCSI DMA valid bit 1 Indicates that the SCSI DMA valid bit is set. 0 No interrupt.
10	GI	Graphics Interrupt 1 Indicates that the graphics controller has asserted an interrupt. 0 No interrupt.
9	SI	Software interrupt 1 Indicates that software has asserted an interrupt. 0 No interrupt.
8–0	Reserved	Ignore these bits.

(concluded)

IER (300 Series Only) Interrupt Enable

Address FFF8 4004 Write Only

The Interrupt Enable Register (IER) contains interrupt enable bits for each interrupt source, except for the Nonmaskable Interrupt (NMI) source. The only NMI source is the Power Fail interrupt. Note that if software disables the single interrupt signal, the NMI is also disabled.

ISR and IER have the same bit assignments. ISR defines the current state of all interrupt requests, and IER enables or masks the interrupts recorded in ISR.

A system reset clears all IER bits to 0; therefore masking all interrupts.

The IER bits are defined as follows:

31	30	29	28 22	21	20	19	18	17	16
Res	PF	PE	Reserved	CI	Res	KB	DU	PP	Res

15	14	13	12	11	10	9	8 0
ET	SC	DT	DW	DV	GI	SI	Reserved

Bit	Name	Function
31	Reserved	Write a 0 to this bit.
30	PF	Powerfail 1 Enables the interrupt. 0 Masks the interrupt.
29	PE	Parity error 1 Enables the interrupt. 0 Masks the interrupt.
28–22	Reserved	Write a 0 to these bits.
21	CI	CIO interrupt 1 Enables the interrupt. 0 Masks the interrupt.
20	Reserved	Write a 0 to this bit.
19	KB	Keyboard port 1 Enables the interrupt. 0 Masks the interrupt.

(continued)

Bit	Name	Function
18	DU	DUART 1 Enables the interrupt. 0 Masks the interrupt.
17	PP	Parallel port 1 Enables the interrupt. 0 Masks the interrupt.
16	Reserved	Write a 0 to this bit.
15	ET	Ethernet interface 1 Enables the interrupt. 0 Masks the interrupt.
14	SC	SCSI protocol controller 1 Enables the interrupt. 0 Masks the interrupt.
13	DT	SCSI DMA terminal count reached 1 Enables the interrupt. 0 Masks the interrupt.
12	DW	SCSI DMA write-protect error 1 Enables the interrupt. 0 Masks the interrupt.
11	DV	SCSI DMA valid bit 1 Enables the interrupt. 0 Masks the interrupt.
10	GI	Graphics Interrupt 1 Enables the interrupt. 0 Masks the interrupt.
9	SI	Software interrupt 1 Enables the interrupt. 0 Masks the interrupt.
8–0	Reserved	Write a 0 to these bits.

(concluded)

SWIR (300 Series Only) **Software Interrupt**

Address FFF8 4014 Write Only

The Software Interrupt Register (SWIR) initiates software interrupts.

The SWIR bits are defined as follows:

7 1	0
Reserved	SWI

Bit	Name	Function
7–1	Reserved	Ignore these bits.
0	SWI	Software interrupt. 1 Interrupts the CPU. The status of this bit is reflected by ISR bit 9 (SI). Writing a 0 to IER bit 9 masks this interrupt. 0 Clears the interrupt request. This bit is not set to 0 after system reset or power–on reset.

Even though a reset does not enable the software interrupt request, software must ensure that bit 0 of the SWIR is set to 0 before enabling the interrupt after a reset.

400 Series CPU Interrupt Registers

This section describes the registers used to interrupt the CPU in 400 series stations.

IEN0, IEN1 (400 Series Only) **Interrupt Enable**

IEN0	**Address FFF8 4004**	**Write**
IEN1	**Address FFF8 4008**	**Write**

The Interrupt Enable registers (IEN0 and IEN1) enable and mask interrupts to the CPUs. IEN0 and IEN1 enable interrupts to CPU0 and CPU1, respectively (a single-processor system uses only IEN0). To enable an interrupt, write a 1 into the corresponding bit in IEN0 or IEN1. To mask an interrupt, write a 0 into the corresponding bit in IEN0 or IEN1. The bits in the IST register and the Interrupt Enable registers are mirror images of each other. A system reset clears all Interrupt Enable register bits to 0; a local reset does not affect these registers.

The IENn bits are defined as follows:

31	30	29	28 27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	Reserved	ZBF	VDI	PAR	IR7	KBD	CIO	SF	IR6	PPI	DI1	DI2

15	14	13	12	11	10	9	8 7	6	5	4	3 2	1	0
ECI	IR5	DTO	IR4	DWP	IR3	DVB	Reserved	IR2	SCI	IR1	Reserved	SI1	SI0

Bit	Name	Function
31	ABT	Abort. 1 Enables the abort pushbutton interrupt. 0 Masks the abort pushbutton interrupt.
30	ACF	Ac failure. 1 Enables the ac power failure interrupt. 0 Masks the ac power failure interrupt.
29	ATO	VMEbus timeout. 1 Enables the VMEbus timeout interrupt. 0 Masks the VMEbus timeout interrupt.
28, 27	Reserved	Write a 0 to these bits.
26	ZBF	Zbuffer. This bit only applies to systems containing the optional Z-buffer board. 1 Enables the Zbuffer request interrupt. 0 Masks the Zbuffer request interrupt.
25	VDI	Video. 1 Enables the video request interrupt. 0 Masks the video request interrupt.

(continued)

Bit	Name	Function
24	PAR	Parity error. 1 Enables the parity error interrupt. 0 Masks the parity error interrupt.
23	IR7	VMEbus level 7. 1 Enables the level 7 interrupt from the VMEbus. 0 Masks the level 7 interrupt from the VMEbus.
22	KBD	Keyboard. 1 Enables the keyboard request interrupt. 0 Masks the keyboard request interrupt.
21	CIO	CIO. 1 Enables the CIO interrupt. 0 Masks the CIO interrupt.
20	SF	System failure. 1 Enables the system failure interrupt. 0 Masks the system power failure interrupt.
19	IR6	VMEbus level 6. 1 Enables the level 6 interrupt from the VMEbus. 0 Masks the level 6 interrupt from the VMEbus.
18	PPI	Parallel port. 1 Enables the parallel port request interrupt. 0 Masks the parallel port request interrupt.
17	DI1	DUART1. 1 Enables the DUART1 interrupt. 0 Masks the DUART1 interrupt.
16	DI2	DUART2. 1 Enables the DUART2 interrupt. 0 Masks the DUART2 interrupt.
15	ECI	Ethernet controller. 1 Enables the Ethernet controller request interrupt. 0 Masks the Ethernet controller request interrupt.
14	IR5	VMEbus level 5. 1 Enables the level 5 interrupt from the VMEbus. 0 Masks the level 5 interrupt from the VMEbus.
13	DTC	DMA terminal count 1 Enables the DMA terminal count interrupt. 0 Masks the DMA terminal count interrupt.
12	IR4	VMEbus level 4. 1 Enables the level 4 interrupt from the VMEbus. 0 Masks the level 4 interrupt from the VMEbus.
11	DWP	DMA write protect error. 1 Enables the DMA write protect error interrupt. 0 Masks the DMA write protect error interrupt.
10	IR3	VMEbus level 3. 1 Enables the level 3 interrupt from the VMEbus. 0 Masks the level 3 interrupt from the VMEbus.

(continued)

Bit	Name	Function
9	DVB	DMA valid bit. 1 Enables the DMA valid bit interrupt. 0 Masks the DMA valid bit interrupt.
8, 7	Reserved	Write a 0 to these bits.
6	IR2	VMEbus level 2. 1 Enables the level 2 interrupt from the VMEbus. 0 Masks the level 2 interrupt from the VMEbus.
5	SCI	SCSI controller. 1 Enables the SCSI controller request interrupt. 0 Masks the SCSI controller request interrupt.
4	IR1	VMEbus level 1. 1 Enables the level 1 interrupt from the VMEbus. 0 Masks the level 1 interrupt from the VMEbus.
3, 2	Reserved	Write a 0 to these bits.
1	SI1	Software-generated interrupt 1. 1 Enables software interrupt 1. 0 Masks software interrupt 1.
0	SI0	Software-generated interrupt 0. 1 Enables software interrupt 0. 0 Masks software interrupt 0.

(concluded)

IST (400 Series Only) Interrupt Status

Address FFF8 4040 Read

The Interrupt Status (IST) register contains the current state of all interrupt requests. When a device generates an interrupt, the interrupt logic sets the corresponding bit in the IST register.

To service an interrupt, the interrupt service routine reads the IST register, and possibly one of the Interrupt Enable registers (IEN, IEN0 or IEN1) to determine whether or not the interrupt is masked. The interrupt service routine services the highest-priority interrupt. The bits in the Interrupt Enable registers and the Interrupt Status register are mirror images of each other.

Resets do not affect IST bits; IST can only be cleared by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	Reserved		ZBF	VDI	PAR	IR7	KBD	CIO	SF	IR6	PPI	DI1	DI2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECI	IR5	DTO	IR4	DWR	IR3	DVB	Reserved		IR2	SCI	IR1	Reserved		SI1	SI0

Bit	Name	Function
31	ABT	Abort pushbutton. 1 The abort pushbutton has been depressed since the last write to the CAB bit in the CLRINT register. 0 The abort pushbutton has not been depressed since the last write to the CAB bit in the CLRINT register.
30	ACF	Ac failure. 1 An ac power failure has occurred since the last write to the ACF bit in the CLRINT register. The ac failure signal originates from the power supply, which is connected to the VMEbus. 0 No ac power failure has occurred.
29	ATO	VMEbus arbitration timeout. 1 The VMEbus bus grant has timed out and generated an interrupt. 0 No VMEbus timeout has occurred.
28, 27	Reserved	Reserved and read as 0.
26	ZBF	Zbuffer This bit only applies to systems containing the optional Z-buffer board. 1 The Zbuffer is requesting an interrrupt. 0 The Zbuffer is not requesting an interrrupt.
25	VDI	Video. 1 The video controller is requesting an interrupt. 0 The video controller is not requesting an interrupt.

(continued)

Bit	Name	Function
24	PAR	Parity error. 1 A parity error has occurred. 0 No parity error has occurred.
23	IR7	VME level 7. 1 A VMEbus level 7 interrupt has occurred. 0 No VMEbus level 7 interrupt has occurred.
22	KBD	Keyboard. 0 The keyboard is not requesting an interrupt. 1 The keyboard is requesting an interrupt.
21	CIO	CIO. 1 The CIO is requesting an interrupt. 0 The CIO is not requesting an interrupt.
20	SF	System failure. 1 A system failure signal has occurred since the last write to the CSF bit in the CLRINT register. This is the SYSFAIL* signal on the VMEbus and not a workstation failure. 0 No system failure has occurred.
19	IR6	VME level 6. 1 A VME level 6 interrupt has occurred. 0 No VME level 6 interrupt has occurred.
18	PPI	Parallel port. 1 The parallel port is requesting an interrupt. 0 The parallel port is not requesting an interrupt.
17	DI1	DUART1. 1 DUART1 is requesting an interrupt. 0 DUART1 is not requesting an interrupt.
16	DI2	DUART2. 1 DUART2 is requesting an interrupt. 0 DUART2 is not requesting an interrupt.
15	ECI	Ethernet controller. 1 The Ethernet controller is requesting an interrupt. 0 The Ethernet controller is not requesting an interrupt.
14	IR5	VMEbus level 5. 1 A VMEbus level 5 interrupt has occurred. 0 A VMEbus level 5 interrupt has not occurred.
13	DTC	DMA terminal count 1 The DMA terminal count has been reached. 0 The DMA terminal count has not been reached.
12	IR4	VMEbus level 4. 1 A VMEbus level 4 interrupt has occurred. 0 No VMEbus level 4 interrupt has occurred.
11	DWP	DMA write protect error. 1 A DMA write protect error has occurred. 0 No DMA write protect error has occurred.
10	IR3	VMEbus level 3. 1 A VMEbus level 3 interrupt has occurred. 0 No VMEbus level 3 interrupt has occurred.

(continued)

Bit	Name	Function
9	DVB	DMA valid bit. 1 A DMA valid bit interrupt has occurred. 0 No DMA valid bit interrupt has occurred.
8, 7	Reserved	Reserved and read as 0.
6	IR2	VMEbus level 2. 1 A VMEbus level 2 interrupt has occurred. 0 No VMEbus level 2 interrupt has occurred.
5	SCI	SCSI controller. 1 A SCSI controller is requesting an interrupt. 0 No SCSI controller is requesting an interrupt.
4	IR1	VMEbus level 1. 1 A VMEbus level 1 interrupt has occurred. 0 No VMEbus level 1 interrupt has occurred.
3, 2	Reserved	Reserved and read as 0.
1	SI1	Software-generated interrupt 1. 1 Software interrupt 1 was generated. 0 Software interrupt 1 was not generated.
0	SI0	Software-generated interrupt 0. 1 Software interrupt 0 was generated. 0 Software interrupt 0 was not generated.

(concluded)

ISS (400 Series Only) Interrupt Source Status

Address FFF8 4088 Read Only

The Interrupt Source Status (ISS) register contains the current status of certain hardware interrupts. Clear an interrupt request by setting the appropriate bit in the Clear Interrupt (CLRINT) register. System and local resets do not affect the ISS register.

7 3	2	1	0
Unused	ABT	ACF	SF

Bit	Name	Function
7–3	Unused	Returns 0 when read.
2	ABT	Abort button status. 1 The abort button is depressed. 0 The abort button is not depressed.
1	ACF	Ac fail status. 1 An ac fail interrupt request is pending (the ACFAIL* line on the VMEbus is asserted low). 0 No ac fail interrupt is pending (the ACFAIL* line on the VMEbus is not asserted).
0	SF	System fail status. 1 The SYSFAIL* line on the VMEbus is asserted (low). 0 The SYSFAIL* line on the VMEbus is not asserted (high).

CLRINT (400 Series Only) Clear Interrupt

Address FFF8 408C Write Only

The Clear Interrupt (CLRINT) register allows software to clear certain hardware interrupts. To clear an interrupt request, write a 1 to the appropriate CLRINT bit. For example, to clear a system failure interrupt request, write 1 into the CSF bit. This clears the System Failure (SF) bit in the Interrupt Source Status (ISS) register. System and local resets do not affect the CLRINT register.

NOTE: Clearing the interrupt request bit in the CLRINT register only clears the interrupt request and not the interrupt condition. The corresponding bit in the ISS register contains the state of the interrupting condition.

7 3	2	1	0
Unused	CAB	CAC	CSF

Bit	Name	Function
7–3	Unused	Ignored when written to.
2	CAB	Clear the abort button interrupt request (ABT) Pressing the abort pushbutton generates this interrupt. 1 Clears the ABT interrupt request. 0 Leaves the ABT interrupt request unchanged.
1	CAC	Clear the ac fail interrupt request (ACF) The power supply generates this interrupt by asserting the ACFAIL* line on the VMEbus. 1 Clears the ACF interrupt request. 0 Leaves the ACF interrupt request unchanged.
0	CSF	Clear the system fail interrupt request (SF) Any controller on the VMEbus may generate this interrupt by asserting the SYSFAIL* line. 1 Clears the SF interrupt request. 0 Leaves the SF interrupt request unchanged.

SETSWI (400 Series Only) **Set Software Interrupt**

Address FFF8 4080 Write Only

The Set Software Interrupt (SETSWI) register generates software interrupts. Setting a bit in SETSWI sets the corresponding interrupt bit in the Interrupt Status (IST) register. Unlike other condition-specific interrupts, you can use the software interrupts for any software condition requiring servicing through an interrupt service routine. As with the other condition-specific interrupts, the interrupt vector is within a table of vectors, and is pointed to by the position of the interrupt in the IST register.

7	6	5	4	3	2	1	0
SI7	SI6	SI5	SI4	SI3	SI2	SI1	SI0

Bit	Name	Function
7–0	SI7–SI0	Set software interrupt. 1 Generates the corresponding software interrupt. 0 Leaves the corresponding software interrupt unchanged.

CLRSWI (400 Series Only) — Clear Software Interrupt

Address FFF8 4084 — Write Only

The Clear Software Interrupt (CLRSWI) register clears software interrupts. Setting a bit in the CLRSWI register clears the corresponding interrupt bit in the Interrupt Status (IST) register. A reset does not affect the CLRSWI register.

7	6	5	4	3	2	1	0
CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0

Bit	Name	Function
7-0	CI7-CI0	Clear software interrupt. 1 Clears the corresponding software interrupt. 0 Leaves the corresponding software interrupt unchanged.

Programming the VME Interrupt Registers

VME controllers initiate interrupts to each other via the Interrupt Request IRQ[7-1]* lines of the VMEbus. VME controllers can interrupt the system board, but the system board can not interrupt other VME controllers. The Interrupt Request (IRQ[7-1]*) lines are assigned levels of interrupt priority with IRQ7* having the highest priority and IRQ1* having the lowest priority. The power supply asserts interrupts if needed through the ACFAIL* and SYSFAIL* lines of the VMEbus.

When a VME controller requires interrupt servicing by the system board, the VME controller asserts the appropriate interrupt request (IRQn*) line on the VMEbus. The VME Interface controller passes the interrupt to the interrupt logic which monitors all incoming interrupt lines for interrupt requests, and processes all interrupts. When an interrupt is asserted, the interrupt logic sets the appropriate bit in the Interrupt Status (IST) register. If this interrupt is not masked by the Interrupt Enable (IEN) register, the interrupt logic asserts the interrupt line (INT) to the CPU associated with that IENn register. Figure 3-3 illustrates how VME controllers interrupt the system board. The CPU acknowledges one of these interrupts by reading the corresponding VME Interrupt Acknowledge and Vector (VIAVn) register. When read, VIAVn asserts the interrupt acknowledge signals IACK* and IACKOUT* on the VMEbus. These interrupt acknowledge signals trigger the VME controller to write the interrupt vector to the VMEbus. The VME interface logic writes this to the Mbus for the CPU to read.

Any VME controller can interrupt the system board and request that it service the interrupt. A VME controller initiates an interrupt to the system board via the Interrupt Status (IST) register. Table 3-2 defines the addresses associated with the VIAVn registers.

Table 3-2 Memory Map of the VME Interrupt Registers

Register	Address	Type
VIAV1	FFF8 5004	Read-only
VIAV2	FFF8 5008	Read-only
VIAV3	FFF8 500C	Read-only
VIAV4	FFF8 5010	Read-only
VIAV5	FFF8 5014	Read-only
VIAV6	FFF8 5018	Read-only
VIAV7	FFF8 501C	Read-only

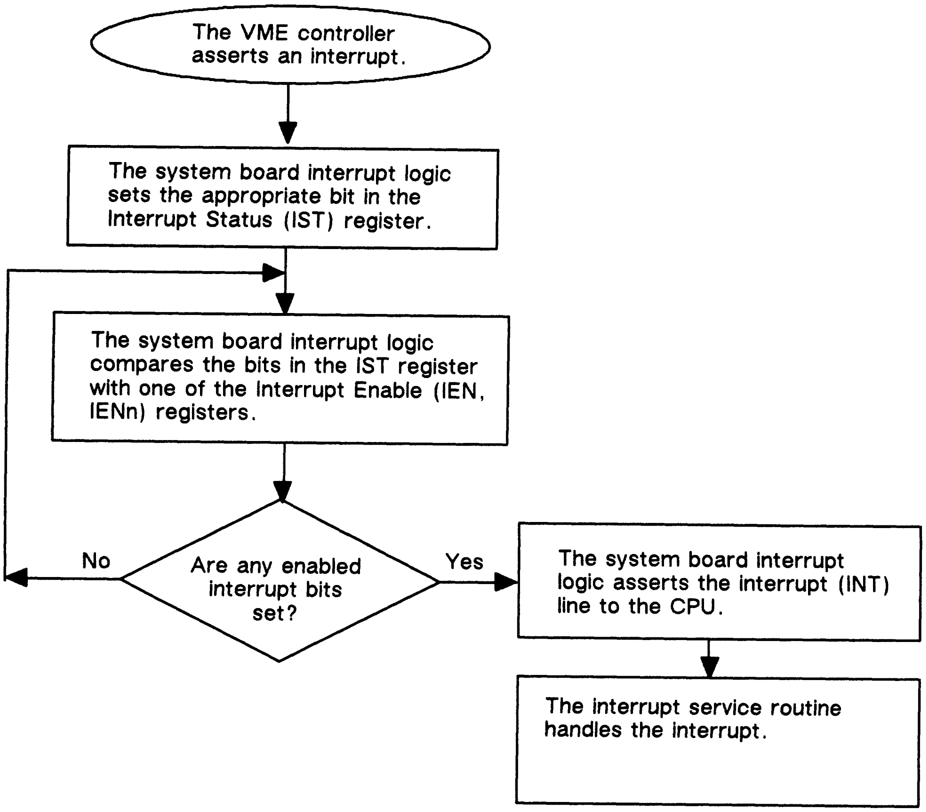

The VME controller asserts an interrupt.
The system board interrupt logic sets the appropriate bit in the Interrupt Status (IST) register.
The system board interrupt logic compares the bits in the IST register with one of the Interrupt Enable (IEN, IENn) registers.
Are any enabled interrupt bits set?
No
Yes
The system board interrupt logic asserts the interrupt (INT) line to the CPU.
The interrupt service routine handles the interrupt.

Figure 3-3 VME Interrupts to the System Board

IRQ[7-0] Level Interrupts

The following series of steps and Figure 3–4 illustrate how a VME controller initiates a level–1 interrupt to the system board.

1. The VME controller asserts the Interrupt Request 0 (IRQ1*) line on the VMEbus.
2. The system board interrupt logic sets the interrupt request level–1 (IR1) bit in the Interrupt Status (IST) register high (1).
3. The system board interrupt logic asserts the INT line to the CPU to notify the CPU that an interrupt request is pending.

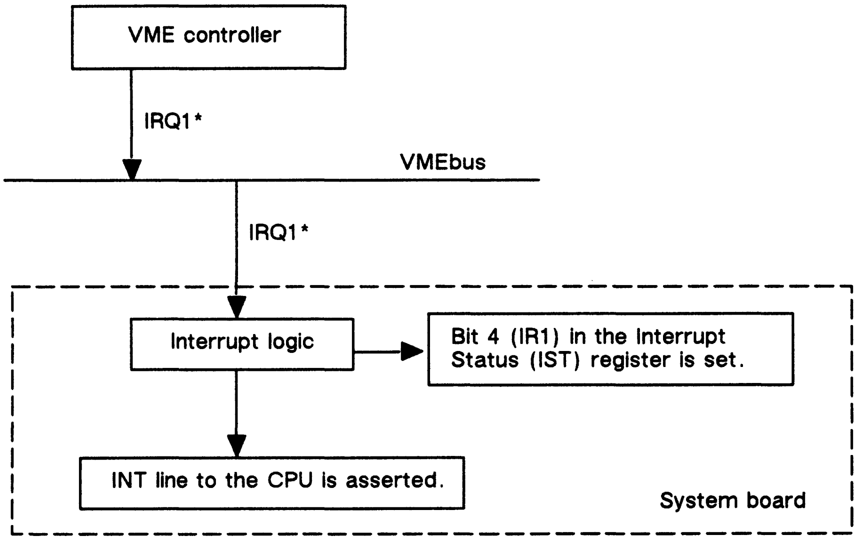
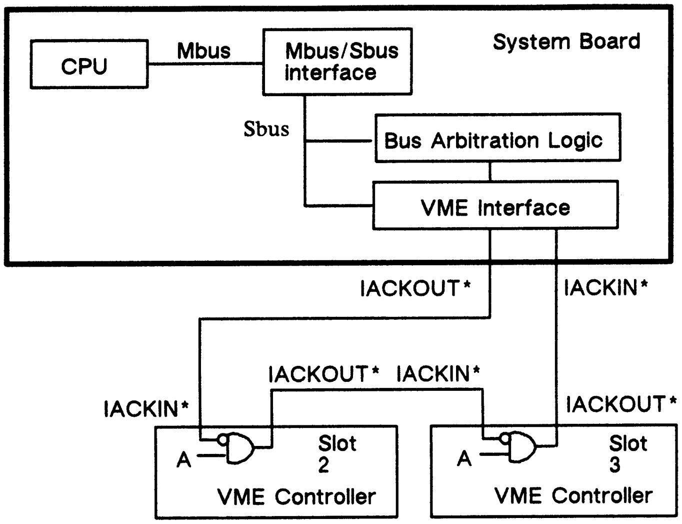

VME controller
IRQ1*
VMEbus
IRQ1*
Interrupt logic
Bit 4 (IR1) in the Interrupt Status (IST) register is set.
INT line to the CPU is asserted.
System board

Figure 3–4 VME Controller Initiating a Level–1 Interrupt to System Board

After the INT line to the CPU has been asserted, the CPU executes an interrupt service routine that finds, acknowledges and handles the interrupt as follows:

1. The CPU reads the IST register and finds the interrupt.
2. The CPU reads the corresponding VME Interrupt Acknowledge and Vector (VIAVn) register to acknowledge the interrupt and to get the interrupt vector. There are seven VIAVn registers, one for each interrupt level. When the CPU reads VIAVn it also checks the state of the IACK Bus Error (IBE) bit. In a multiprocessor system, if IBE is set, the CPU disregards the interrupt and returns to its previous process. If IBE is not set, the CPU first sets IBE to notify other CPUs that the interrupt is being handled, then the CPU executes the appropriate interrupt service routine (see the next step).
3. The CPU executes the new interrupt service routine located at the vector obtained from the VIAV[7-1] register.

When the CPU reads IBE, it asserts the Interrupt Acknowledge line (IACKOUT*), which is daisy chained with the VME controllers in slots 2 and 3 of the VMEbus (see Figure 3-5.)



NOTE: If A is 1, the VME controller passes the interrupt acknowledge to the next controller. If A is 0, the VME controller takes the interrupt acknowledge.

Figure 3-5 VMEbus Grant Daisy-Chain

VIAVn (400 Series Only) VME Interrupt Acknowledge and Vector

VIAV1	**Address FFF8 5004**	**Read Only**
VIAV2	**Address FFF8 5008**	
VIAV3	**Address FFF8 500C**	
VIAV4	**Address FFF8 5010**	
VIAV5	**Address FFF8 5014**	
VIAV6	**Address FFF8 5018**	
VIAV7	**Address FFF8 501C**	

The CPU acknowledges interrupts and obtains interrupt vectors from VME controllers via the seven VME Interrupt Acknowledge and Vector (VIAVn) registers. Each of the seven interrupt request levels has a VIAVn register associated with it. When a VME controller interrupts the system board CPU via one of the seven VME interrupt request levels, the CPU acknowledges the interrupt request and obtains the interrupt request vector via the VIAVn register. When the CPU reads VIAVn, the interrupt logic asserts the Interrupt Acknowledge (IACK* and IACKOUT*) lines to the VMEbus. The VME controller responds by writing the interrupt vector to the data lines and by asserting the Data Transfer Acknowledge (DTACK*) signal on the VMEbus. If DTACK* is not asserted, the system board interrupt logic sets the Interrupt Acknowledge Bus Error (IBE) bit in the VIAVn register. If IBE is already set, the CPU will disregard the interrupt vector. If the system board has two CPUs and if IBE is set, the CPU disregards the interrupt and returns to its previous process.

Resets do not affect the VIAVn registers.

15 9	8	7 0
Unused	IBE	VIV

Bit	Mnemonic	Function
15–9	Unused	
8	IBE	IACK Bus Error 1 Indicates that the last VME interrupt acknowledge (IACK) was terminated by a VMEbus error (BERR). 0 Indicates that the last VMEbus IACK cycle was successfully completed; it was terminated by a data transfer acknowledge. The VIV bits contain a valid interrupt vector.
7–0	VIV[7–0]	VME Interrupt Vector from D[7–0] VIV is the interrupt vector from the VME controller generating the interrupt.

NOTE: VIV contains a valid vector only when IBE is cleared.

System Errors

Table 3–3 summarizes the system error conditions and the type of fault or interrupt generated in response to the error. The error status bits differ between workstations; see the descriptions of the ISR (300 series stations) or the IST (400 series stations) registers.

Table 3–3 System Error Conditions and Responses

Error Condition	Response	
Parity error, CMMU master	Mbus parity fault [1]	
Nonexistent memory, CMMU master	Mbus parity fault [1]	
Watchdog time-out, Mbus/Sbus master	Mbus fault	
Watchdog time-out, Mbus/Sbus master	No action	
	300 ISR Bit	**400 IST Bit**
AC power fail	30 (NMI)	30 (maskable)
Parity error, DMA master	29 [1]	24 [1]
Nonexistent memory, DMA master	29 [1]	24 [1]
DUART internal error/interrupt	18	17, 16
Ethernet internal error/interrupt	15	15
SCSI internal error/interrupt	14	5
DMA write-protect error	12	11
DMA valid bit error	11	9

[1] Software must distinguish between an actual parity error and a nonexistent memory error by using the address of the faulted transaction, which is latched by the faulted CMMU, and the memory configuration of the machine.

Bus Faults

Bus faults occur when a CMMU is the bus master and a requested transaction causes an error. The CMMUs detect the bus fault condition and signal a fault to the CPU via the Pbus fault reply status. Software must read the CMMU System Status register to determine the cause of the fault.

Two bus fault conditions generate parity faults: data parity errors, and reads from nonexistent memory. Writes to nonexistent memory do not generate parity faults; the data is lost.

Parity interrupts may occur either when a controller is master of the Sbus and requests a transaction that causes an error, or when a slave device detects an internal error.

The interrupt service routine should read the interrupt status register (ISR or IST) to determine the cause of the interrupt. Some controllers have internal status registers that provide more precise interrupt information.

The Mbus/Sbus interface contains a timer that terminates bus cycles when an Mbus wait does not allow the DRAMs to be refreshed. When this happens, the timer generates an Mbus error, and the CPU receives a Pbus fault status from the faulting CMMU. The System Status register in the CMMU at fault indicates that an Mbus fault occurred. A time–out is the only event that generates an Mbus fault. (Mbus faults are different from Pbus faults.) Such a time–out occurs with the following conditions:

- A hardware fault in the slave device may exist, causing it to assert its wait signal for longer than the allowable time.
- Some slave devices are accessed without first clearing the corresponding reset bit in the Diagnostic Control Register (DCR). When this occurs, the slave device does not respond properly to the access.
- The drawing preprocessor is busy while an attempt is made to write to the parameter registers of the video controller. The write request causes a watchdog time–out fault.

End of Chapter

Chapter 4
Programming the Monochrome Graphics Subsystem

This chapter describes the monochrome graphics subsystem used in the 300 series station. To accomplish this, the chapter describes the following topics:

- The monochrome graphics subsystem.
- Hardware design details that affect the graphics controller's operation.
- How to program the monochrome graphics subsystem.
- How to program the frame buffer.

Features of the Monochrome Graphics Subsystem

The monochrome graphics subsystem controls the bit-mapped display (monitor), providing the following:

- 1280 x 1024 pixel display, 70-Hz noninterlaced refresh rate, compliant with U.S., Canadian, and European VDT standards.
- 1280 x 614.4 pixel, one plane frame buffer, providing extra off-screen storage for fonts, icons, small windows.
- 25K characters per second (16 x 16 pixel font, cached in video memory).
- 50K ten-pixel random vectors per second; 25K fifty-pixel random vectors per second.
- 50K ten-pixel random vectors transformed and clipped per second (2D or 3D).
- 65K ten-pixel polyline segments transformed and clipped per second (2D or 3D).
- 20M pixels copied or scrolled per second using Bit Block Transfer (BITBLT).
- 40M pixels filled or cleared per second using BITBLT.
- Windowing support through use of a clipping rectangle.

The subsystem resides as a slave device on the Mbus. It uses two separate memory mapped areas: a register set and the frame buffer. You program the subsystem either by setting up parameters and a command in the register set or by manipulating the frame buffer directly. Parameters include frame buffer coordinates and a command opcode. When the program writes the last parameter to the register set, the subsystem executes the requested command without further program intervention.

Components of the Monochrome Graphics Subsystem

As shown in Figure 4–1, the subsystem consists of the following:

- Mbus interface.
- NEC μPD72120 graphics controller.
- Display memory bus.
- Frame buffer with optional off–screen storage.
- Timing and control logic.
- Parallel–to–serial shift register.
- D/A converter and video output driver.

The rest of this section describes these components.

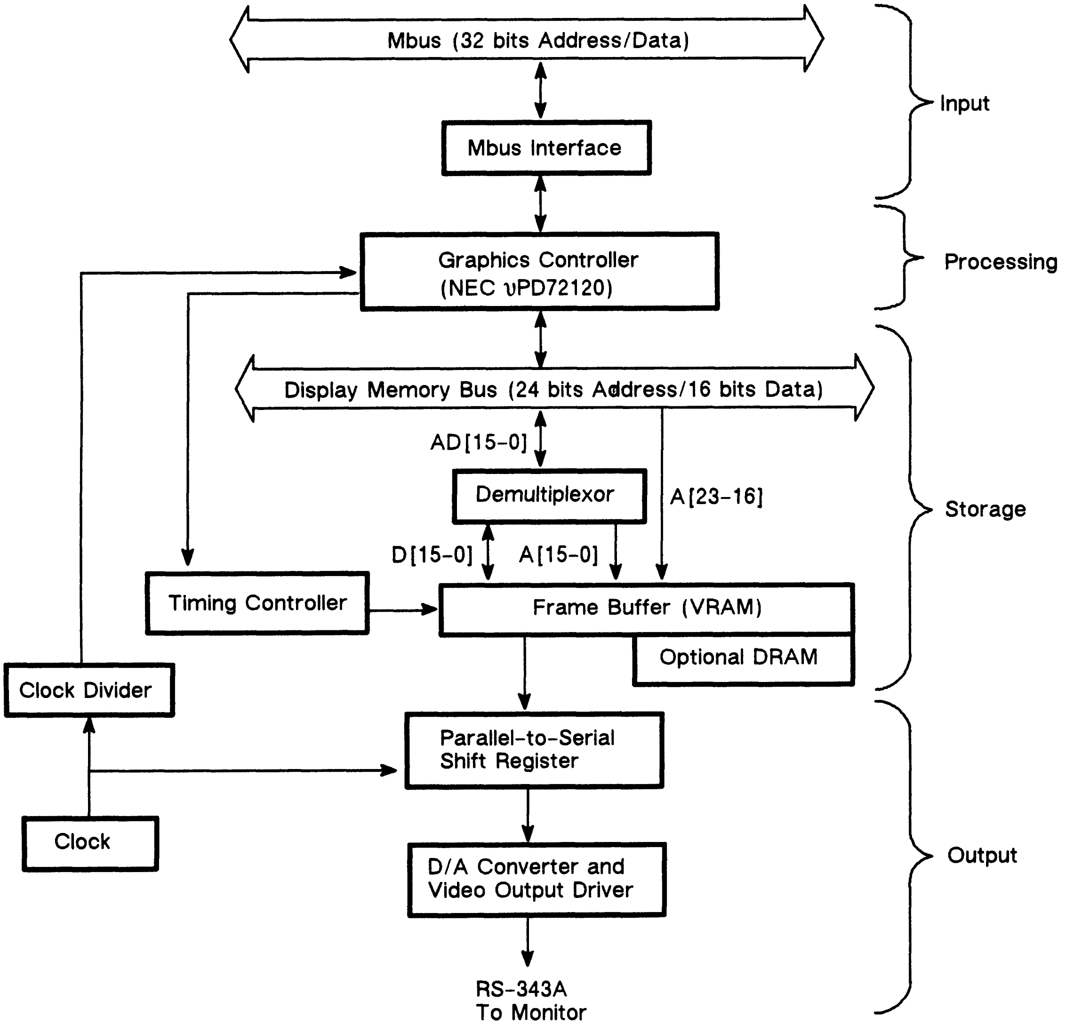


Figure 4–1 The Monochrome Graphics Subsystem

Mbus Interface

The Mbus interface links the Mbus to the monochrome graphics controller. The Mbus interface consists of an address buffer, address decode logic, data transceiver, parity generating logic (parity to the Mbus), and synchronization and control logic.

The Mbus interface monitors the Mbus and responds appropriately when the graphics subsystem is selected. The graphics subsystem is a slave device on the Mbus; it cannot be master of the Mbus. The graphics subsystem ignores parity bits when the CPU writes to it, but generates parity bits when the CPU reads from it.

Accesses by the CPU to the graphics subsystem require a wait state during both the address and data phases of the access. Since the subsystem is not fast enough to respond to the address phase of a CPU access, it latches the address, and then signals the CPU to start the data phase. The subsystem extends the data phase with wait states until the read or write operation is complete.

Monochrome Graphics Controller

The graphics subsystem uses the NEC μPD72120 graphics controller. The graphics controller contains several instructions that it performs internally, reducing demands on the CPU. Among these commands are screen refresh, video memory refresh, and block move.

The CPU programs the graphics subsystem via the graphics controller's internal registers. The graphics subsystem completes operations independent from the CPU; the graphics controller executes commands, such as scrolling a window, while the CPU performs other operations. Furthermore, the graphics subsystem is pipelined; the CPU can program the subsystem for a new operation while the subsystem is executing an existing operation.

Display Memory Bus

The graphics controller reads from or writes to the frame buffer over the display memory bus. The display memory bus has 16 multiplexed address/data bits and an additional 8 address bits to extend the address range to 24 bits. All data is read as 16-bit words.

Display Memory Bus Control

The video memory control provides the proper timing and control for the frame buffer by generating read, write, data transfer, and refresh cycles. Refresh occurs during the horizontal sync period, and is clocked by a signal that is four times the graphics controller's 8-MHz clock.

Frame Buffer

The subsystem's frame buffer (or video memory) consists of Video RAMs (VRAMs) containing the video information that is continuously displayed on the screen. A VRAM is a dual-port memory device with one port for graphics controller access and one port for sending data to the monitor.

The frame buffer is organized as a block of DRAM with a large static serial register that loads an entire DRAM row at a time for shifting out serially to the display. This organization allows the graphics controller to have over 88 percent of the frame buffer's bandwidth for drawing operations.

Each memory location in the frame buffer stores one pixel: 1 for a white pixel, or 0 for a black pixel. The frame buffer is a 1280 (horizontal) x 1638.4 (vertical) pixel block. It is further divided into a 1280 (H) x 1024 (V) visible area and a 1280 (H) x 614.4 (V) off-screen area located below the visible memory.

The frame buffer may include additional off-screen memory (4 Mbits of DRAM). This memory is for memory-intensive fonts such as the Kanji character set, and is addressed in the same way as the parallel ports of a VRAM. Unlike a VRAM, the DRAM has no serial port.

Parallel-to-Serial Shift Register

A parallel-to-serial shift register converts 16-bit parallel data from the frame buffer to serial data, and transmits the data to the D/A converter. The D/A converter converts the serial input data to analog data, and passes the data to the video output driver. The video output driver converts the analog data into RS-343A compatible data and transmits it to the video monitor.

D/A Converter and Video Output Driver

The data is transmitted to the video display monitor at 125-MHz. The RS-343A signal is an analog voltage signal that can drive a monitor capable of monochrome or gray-scale video. This data produces a 70-Hz screen refresh rate for the monitor's 1280 x 1024 pixel screen display. In addition to the data signal, the output stage produces the horizontal synchronization signal, the vertical synchronization signal, and the blanking signal.

How This Implementation Differs from NEC Specifications

The graphics controller operation differs from the µPD72120's operation specified in NEC's *µPD72120 Advanced Graphics Display Controller User's Manual.* These differences in operation affect the horizontal front and back porches, direct reads of a half-word display memory value, and register and frame buffer location addresses. The rest of this section describes these differences.

Horizontal Front and Back Porches

The horizontal synchronization signal is two clock ticks (2 SCLKs) behind the horizontal blank signal. As a result, the horizontal front porch (HFP) is one clock tick less and the horizontal back porch (HBP) is one clock tick more than the NEC user's manual states. This means that if you program the graphics controller to set HFP equal to 3 (or 8 SCLKs), the graphics subsystem sets HFP equal to 2 (or 6 SCLKs); and, if you program it to set HBP equal to 3 (or 8 SCLKs), the graphics subsystem sets HBP equal to 4 (or 10 SCLKs).

Reading Data from the Frame Buffer

When the CPU reads data from the video memory, it receives the data with the most-significant bit (MSbit) bit 15:

15	14	1	0
MSbit			LSbit

NEC's user manual indicates that the MSbit is bit 0:

15	14	1	0
LSbit			MSbit

Addressing theRegisters and Frame Buffer

The graphics controller registers assume that the system uses byte addressing. Since the CPU uses word addressing, the graphics subsystem hardware shifts each address, so that the address of each graphics controller register or frame buffer location is a word address specifying a half-word (16-bit) location. As a result, the workstation maps the graphics controller registers and frame buffer locations to different addresses than those given in the NEC user's manual. For a complete register address map and frame buffer address map, refer to the section "Accessing the Registers" later in this chapter, and the appendix "System Address Map."

Word Count

The word count is one more than stated in the NEC documentation. The workstation's system control logic initializes the word count to 4E for a 1280-pixel-wide display.

Programming the Monochrome Graphics Subsystem

The CPU writes parameters and commands to graphics controller registers, causing the controller to draw in the frame buffer. The graphics controller consists of a drawing preprocessor and a drawing processor. While the drawing processor executes the current drawing command, the preprocessor accepts parameters for the next drawing command and sets up that command.

The rest of this section lists the drawing commands implemented by the graphics controller and discusses graphics controller interrupts and how to program and initialize the registers.

Drawing Commands

Table 4–1 lists the graphics controller commands. See the NEC *μPD72120 Advanced Graphics Display Controller User's Manual* for a complete description of how each command works.

Table 4–1 Monochrome Graphics Controller Commands

Commands
Drawing Commands
Arc
Circle
Circular Bow
Dot
Ellipse
Elliptical Arc
Elliptical Bow
Elliptical Sector
Rectangle
Sector
Straight Line
Paint Commands
Arbitrary Closed Area
Circular Area
Elliptical Area
Rectangular Area
Trapezoidal Area
Triangular Area
Copy Commands
Physical address to physical address
Physical address to coordinate address
Coordinate address to coordinate address
Coordinate address to physical address
Rotation and Sizing Commands
Arbitrary angle rotation
Enlarge/shrink
90 degree rotation
Data Read Commands
Color information read
Coordinate value read
PUT/GET Commands
Coordinate address to system
System to coordinate address
Physical address to system
System to physical address

Graphics Controller Interrupts

When the graphics subsystem has a condition that requires CPU intervention, the graphics controller sets the appropriate bits in its Status register, and then it asserts its interrupt (INT) line. The INT line eventually sets the Graphics Interrupt (bit 10) of the system board's Interrupt Status Register (ISR). The operating system should contain a graphics interrupt service routine that reads the graphics controller's Status register and performs necessary operations to clear the interrupt condition.

The operating system can mask some graphics controller interrupts via the graphics controller's Control register. The following interrupts can be masked:

- The preprocessor went from a busy to a nonbusy state.
- The drawing processor went from a busy to a nonbusy state.
- Drawing is occurring in a clipped region.

The following interrupts cannot be masked:

- The preprocessor detected an error during processing.
- The drawing processor detected an error during processing.

For more information, see the NEC *μPD72120 Advanced Graphics Display Controller User's Manual.*

Programming the Monochrome Graphics Registers

The monochrome graphics registers occupy a 256-byte space in the graphics controller and are accessed as 16-bit words. Since the registers vary in length and are packed into the 256-byte space, some registers occupy different bits of the same address. Thus, a program may access more than one register when addressing a 16-bit word. Some registers also have the same address as others, so which register a program accesses may depend on whether or not the program reads or writes to the particular address. For these reasons, when writing programs to access these registers you must ensure that the program interprets all 16 bits correctly when it reads from an address, and sends all 16 bits correctly when it writes to an address. Since the 16-bit data port of the graphics controller is mapped to the low-order 16 bits (15-0) of the Mbus, any data read to or written from a graphics controller register must appear in those Mbus bits.

The registers are mapped to address space FFF8 9000 through FFF8 90FC. Table 4-2 lists each register together with the Mbus bit field that contains the register's contents. The graphics controller registers are described in the NEC *μPD72120 Advanced Graphics Display Controller User's Manual.*

Table 4-2 Address Map for the Monochrome Graphics Controller Registers

Address	Register	Type	Bits
FFF8 9078	Control (CTRL)	Write	15-8
	STATUS	Read	15-0
FFF8 9078 [1]	BANK	Write	7-0
FFF8 90E4	Address Control (AC)	Write	14-12
	DISPLAY_PITCH	Write	11-0
FFF8 90E0	DISPLAY_CTRL	Write	15-0
FFF8 90E8	Display Address (DAD) (lower and middle byte)	Write	15-0
FFF8 90EC	Display Address (DAD) (upper byte)	Write	7-0
	Word Count (WC)	Write	15-8
FFF8 90F8	Word Count (WC) (upper 4 bits)	Write	15-12
FFF8 90F0 [1]	Cursor Enable (CE)	Write	14
	Cursor Mode Select (CRS)	Write	15
	Cursor X Coordinate (GCSRX)	Write	11-0
FFF8 90F4 [1]	Cursor Y Start (GCSRYS)	Write	11-0
FFF8 90F8 [1]	Cursor Y End (GCSRYE)	Write	11-0
FFF8 90FC [2]	Horizontal Sync (HS)	Write	11-0
	Horizontal Back Porch (HBP)	Write	11-0
	Half Horizontal Time (HH)	Write	11-0
	Horizontal Display Period (HD)	Write	11-0
	Horizontal Front Porch (HFP)	Write	11-0
	Vertical Sync (VS)	Write	11-0
	Vertical Back Porch (VBP)	Write	11-0
	Lines per Field (L/F)	Write	11-0
	Vertical Front Porch (VFP)	Write	11-0
FFF8 9000	Execution Address Origin (EADORG) (lower/middle byte)	Read/Write	15-0

[1] Not used in workstation.
[2] At powerup, values are written to same address in sequence.

(continued)

Table 4–2 Address Map for the Graphics Controller Registers

Address	Register	Type	Bits
FFF8 9004	Execution Address Origin (EADORG) (upper byte)	Read/Write	7–0
	Dot Address Origin (DADORG)	Read/Write	11–8
FFF8 9060	Pitch Source (PITCHS)	Read/Write	15–0
FFF8 9064	Pitch Destination (PITCHD)	Read/Write	15–0
FFF8 9018 [1]	Plane Displacement Source (PDISPS) (lower/middle byte)	Read/Write	15–0
FFF8 9020 [1]	Plane Displacement Destination (PDISPD) (lower/middle byte)	Read/Write	15–0
FFF8 9024 [1]	Plane Displacement Destination (PDISPD) (lower/middle byte)	Read/Write	7–0
FFF8 9028	Plane Maximum (PMAX)	Read/Write	15–0
FFF8 906C	Plane Select (PLANES)	Read/Write	15–0
FFF8 902C	Drawing Mode 0 (MOD0)	Read/Write	3–0
	Drawing Mode 1 (MOD1)	Read/Write	7–4
FFF8 9030	Pattern Pointer (PTPN) (lower/middle byte)	Read/Write	15–0
FFF8 9034	Pattern Pointer (PTPN) (upper byte)	Read/Write	7–0
FFF8 90C0	Pattern Count (PTNCNT)	Read/Write	15–0
FFF8 9038	Stack Pointer (STACK) (lower/middle bytes)	Read/Write	15–0
FFF8 903C	Stack Pointer (STACK) (upper bytes)	Read/Write	7–0
FFF8 9068	Stack Maximum (STMAX)	Read/Write	15–0
FFF8 90D8	Clipping Mode (CLIP)	Read/Write	9–8
	Magnifier Horizontal (MAGH)	Read/Write	7–4
	Magnifier Vertical (MAGV)	Read/Write	3–0
FFF8 90C4	X Clipping Minimum (XCLMIN)	Read/Write	15–0
FFF8 90C8	Y Clipping Minimum (YCLMIN)	Read/Write	15–0
FFF8 90CC	X Clipping Maximum (XCLMAX)	Read/Write	15–0
FFF8 90D0	Y Clipping Maximum (YCLMAX)	Read/Write	15–0
FFF8 9008	Execution Address 1 (EAD1) (lower/middle bytes)	Read/Write	15–0
FFF8 900C	Execution Address 1 (EAD1) (upper byte)	Read/Write	7–0
	Dot Address 1 (DAD1)	Read/Write	11–8
FFF8 9010	Execution Address 2 (EAD2) (lower/middle bytes)	Read/Write	15–0
FFF8 9014	Execution Address 2 (EAD2) (upper byte)	Read/Write	7–0
	Dot Address 2 (DAD2)	Read/Write	11–8
FFF8 9080	X	Read/Write	15–0
FFF8 9084	Y	Read/Write	15–0
FFF8 9088	DX	Read/Write	15–0
FFF8 908C	DY	Read/Write	15–0
FFF8 9090	XS	Read/Write	15–0
FFF8 9094	YS	Read/Write	15–0
FFF8 9098	XE	Read/Write	15–0
FFF8 909C	YE	Read/Write	15–0
FFF8 90A0	XC	Read/Write	15–0
FFF8 90A4	YC	Read/Write	15–0
FFF8 90A8	DH	Read/Write	15–0
FFF8 90AC	DV	Read/Write	15–0
FFF8 90DC	Command	Read/Write	15–0
FFF8 907C	PGPORT (PUT/GET Port)	Read/Write	15–0

[1] Not used in workstation.

(concluded)

Before writing parameters to the registers, a program must read the graphics controller's Status register to find out if the preprocessor is busy. If the preprocessor is busy, the program must keep reading the Status register for the preprocessor's status. When the preprocessor is not busy, the program can write the parameters for the next draw command to the registers. After writing the parameters, the program should write the draw command.

Initializing the Registers

During a system reset or power on reset, the power-up code initializes the display. To do this, the power-up program writes to the Display Control Register (DCR) to set the timing registers; then it writes the time parameters to the appropriate registers. The screen remains blank until initialization is complete.

For more information on initializing the monochrome graphics controller, see the NEC *μPD72120 Advanced Graphics Display Controller User's Manual.*

Sample Program

The C program that follows initializes and programs the graphics controller registers. Since the code in the workstation's boot PROM initializes the registers in a way similar to the graph_init routine in this program on workstation powerup and reset, other software does not need to run an initialization program again.

Note that the graph_init routine initializes the origin of the video memory in the lower left corner of the on-screen space with the Y-axis growing positively upward and the X-axis growing positively to the right. Since this initializes coordinate 0,0 to be the lower-left hand corner, the 16 pixels at address 8000 0000 correspond to coordinates 0,1023 through 15,1023 (the most significant bit of the 16-bit value at 8000 0000 is 0,1023). The routine places the off-screen space directly below the on-screen space with its Y-axis growing negatively downward and its X-axis growing positively to the right. Figure 4-2 shows the video memory coordinate system.

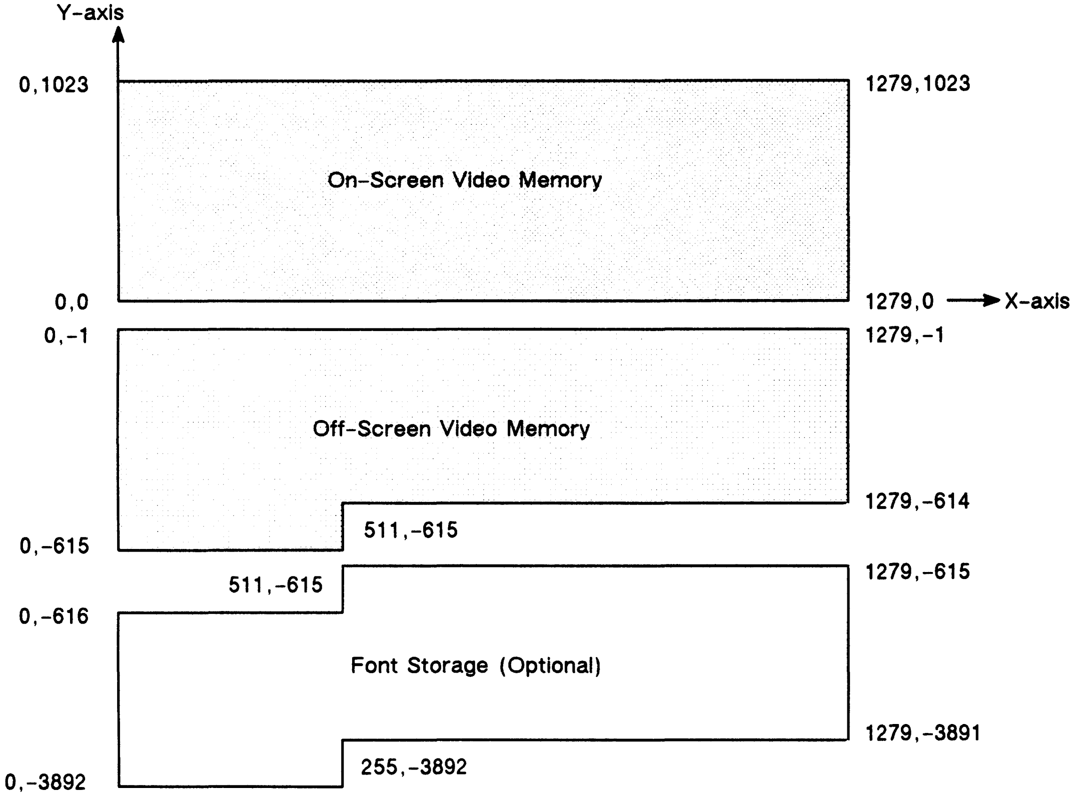


Figure 4–2 Monochrome Graphics Video Memory Coordinate System

Sample C Program

```
/********* DATA STRUCTURE FOR NEC REGISTERS *************/
struct agdc_reg
{
        int eadorg;                 /* adr 00 */
        int dadorg;                 /* adr 04 */
        int ead1;                   /* adr 08 */
        int dad1;                   /* adr 0C */
        int ead2;                   /* adr 10 */
        int dad2;                   /* adr 14 */
        int pdisps1;                /* adr 18 */
        int pdisps2;                /* adr 1C */
        int pdispd1;                /* adr 20 */
        int pdispd2;                /* adr 24 */
        int pmax;                   /* adr 28 */
        int mod0_mod1;              /* adr 2C */
        int ptpn1;                  /* adr 30 */
        int ptpn2;                  /* adr 34 */
        int stack1;                 /* adr 38 */
        int stack2;                 /* adr 3C */
/* internal working registers */

int filler[14];
        int status;                 /* adr 78 */
        int pgport;                 /* adr 7C */
        int x;                      /* adr 80 */
        int y;                      /* adr 84 */
        int dx;                     /* adr 88 */
        int dy;                     /* adr 8C */
        int xs;                     /* adr 90 */
        int ys;                     /* adr 94 */
        int xe;                     /* adr 98 */
        int ye;                     /* adr 9C */
        int xc;                     /* adr A0 */
        int yc;                     /* adr A4 */
        int dh;                     /* adr A8 */
        int dv;                     /* adr AC */
        int pitchs;                 /* adr B0 */
        int pitchd;                 /* adr B4 */
        int stmax;                  /* adr B8 */
        int planes;                 /* adr BC */
        int ptncnt;                 /* adr C0 */
        int xclmin;                 /* adr C4 */
        int yclmin;                 /* adr C8 */
        int xclmax;                 /* adr CC */
        int yclmax;                 /* adr D0 */
int rD4;
        int mag;                    /* adr D8 */
        int command;                /* adr DC */
        int display_ctrl;           /* adr E0 */
        int display_pitch;          /* adr E4 */
        int dis_ad1;                /* adr E8 */
        int dis_ad2;                /* adr EC */
        int cursor;                 /* adr F0 */
        int gcsrys;                 /* adr F4 */
        int gcsrye;                 /* adr F8 */
        int hor_vert;               /* adr FC */
};
```

```
/******* Logical Drawing Operations **********/

#define SRC_DEST                0
#define NOTSRC_DEST             1
#define ZERO_DEST               2
#define ONE_DEST                3
#define EXOR_SRC_DEST           4
#define EXOR_NOTSRC_DEST        5
#define EXOR_ZERO_DEST          6
#define EXOR_ONE_DEST           7
#define AND_SRC_DEST            8
#define AND_NOTSRC_DEST         9
#define AND_SRC_NOTDEST         10
#define AND_NOTSRC_NOTDEST      11
#define OR_SRC_DEST             12
#define OR_NOTSRC_DEST          13
#define OR_SRC_NOTDEST          14
#define OR_NOTSRC_NOTDEST       15
```

```
        /**** MACRO FOR ROUTINES THAT USE GRAPHICS *****/
```

/** This routine defines a register for the pointer to the graphics processor so it must be redeclared in every routine that writes to the graphics processor registers. The register address can be made global, but it will not be made a "register" variable by the compiler. The call to GRAPH(); should be in the variable declaration section of the routines. **/

```
#define GRAPH() register struct agdc_reg *reg = (struct agdc_reg *) 0xfff89000
```

```
        /****** Some macros that use the NEC chip directly **********/

        /****** NOTE: There are no curly brackets around macros. *****/

#define CHK_STATUS_PRE()      while(reg->status & 1)
#define CHK_STATUS_PROC()     while(reg->status & 2)
#define SETPTN(p)             CHK_STATUS_PRE();    \
                              reg->ptncnt = p
#define COL_MODE(p)           CHK_STATUS_PRE();    \
                              reg->mod0_mod1 = 0x11 * p
#define FILL()                CHK_STATUS_PRE();    \
                              reg->mod0_mod1 = 0x33; \
                              reg->ead1 = 0;       \
                              reg->dad1 = 0;       \
                              reg->dh = 1279;      \
                              reg->dv = 1023;      \
                              reg->command = 0x8e3e  /* A_REC_FILL_A */
#define BLANK()               CHK_STATUS_PRE();    \
                              reg->mod0_mod1 = 0x22; \
                              reg->ead1 = 0;       \
                              reg->dad1 = 0;       \
                              reg->dh = 1279;      \
                              reg->dv = 1023;      \
                              reg->command = 0x8e3e  /* A_REC_FILL_A */
#define LINE(x1,y1,x2,y2)     if ((y1) == (y2)) {  \
                              CHK_STATUS_PRE();    \
                              reg->x = x1;         \
                              reg->xs = x2;        \
                              reg->y = y1;         \
                              reg->ys = y2;        \
                              reg->command = 0x8c3e; }      \
                              else {               \
                              CHK_STATUS_PRE();    \
                              reg->x = x1;         \
                              reg->y = y1;         \
                              reg->xe = x2;        \
                              reg->ye = y2;        \
                              reg->command = 0x1841; } /* A LINE M1  */
/****** Above routines use fills for horizontal lines **********/

#define BITBLTUL(xd,yd,xs,ys,ws,hs)                \
                              CHK_STATUS_PRE();    \
                              reg->xs = xs;        \
                              reg->ys = ys;        \
                              reg->x = xd;         \
                              reg->y = yd;         \
                              reg->dh = ws - 1;    \
                              reg->dv = hs - 1;    \
                              reg->command = 0x840e  /* A_COPY_CC */
```

```
    /***** ROUTINE TO INITIALIZE GRAPHICS CONTROLLER *********/
graph_init()
{
GRAPH();
        /*********  Timing Setup for 125 MHz DOT Clock  ********/

reg->display_ctrl = 0xcb38;
reg->display_ctrl = 0xcb3a;       /* Setting STSP bit to 1. */
reg->hor_vert = 3;                /* Horizontal synchronizing signal. */
reg->hor_vert = 4;                /* Horizontal back porch. */
reg->hor_vert = 1;                /* Rising/falling times for even
                                  field vertical synchronizing signal
                                  during interlace display. */
reg->hor_vert = 0x27;             /* Horizontal display period. */
reg->hor_vert = 2;                /* Horizontal front porch. */
reg->hor_vert = 4;                /* Vertical synchronizing signal. */
reg->hor_vert = 0x2a;             /* Vertical back porch. */
reg->hor_vert = 0x400;            /* Line/field display period
                                  in vert direction. */
reg->hor_vert = 1;                /* Vertical front porch. */
reg->display_ctrl = 0xcb38;       /* Set the STSP bit back to 0. */
                     /***********  Drawing Setup  ************/

reg->display_pitch = 0x50;        /* Number of words covering
                                  horizontal width of memory plane. */
reg->dis_ad1 = 0;                 /* Screen display start address
                                  set to 0. */
reg->dis_ad2 = 0x4e00;            /* Number of displayed words during
                                  display period within one
                                  scanning cycle. */
reg->gcsrye = 1;                  /* Clear upper bits of word count reg.*/
reg->eadorg = 0x3fb0;             /* Absolute address within display
                                  memory corresponding to the origin of
                                  X-Y coordinates (0,0) set to LOWER LEFT
                                  hand side of screen. */
reg->dadorg = 1;                  /* Dot address origin also set to 0. */
reg->pitchs = 0x50;               /* Sets the horizontal width using the
                                  number of words for the drawing source
                                  memory plane within display memory. */
reg->pitchd = 0x50;               /* Sets the horizontal width using the
                                  number of words for the drawing target
                                  memory plane within display memory. */
reg->pdispd1 = 0;                 /* Only one memory plane so set to 0. */
reg->pdispd2 = 0;                 /* Only one memory plane so set to 0. */
reg->pmax = 1;                    /* Specifies the memory planes, using
                                  the first memory plane only. */
reg->planes = 0;                  /* Specifies the logical operation to be
                                  carried out during drawing for each
                                  plane. */
reg->mod0_mod1 = 0;               /* The logical operation is set to copy
                                  source to destination. */
reg->mag = 0x1ff;                 /* No magnification. */
reg->status = 0;                  /* Clear bank and set status. */
reg->ptncnt = 0;                  /* Set pattern register to all black. */
```

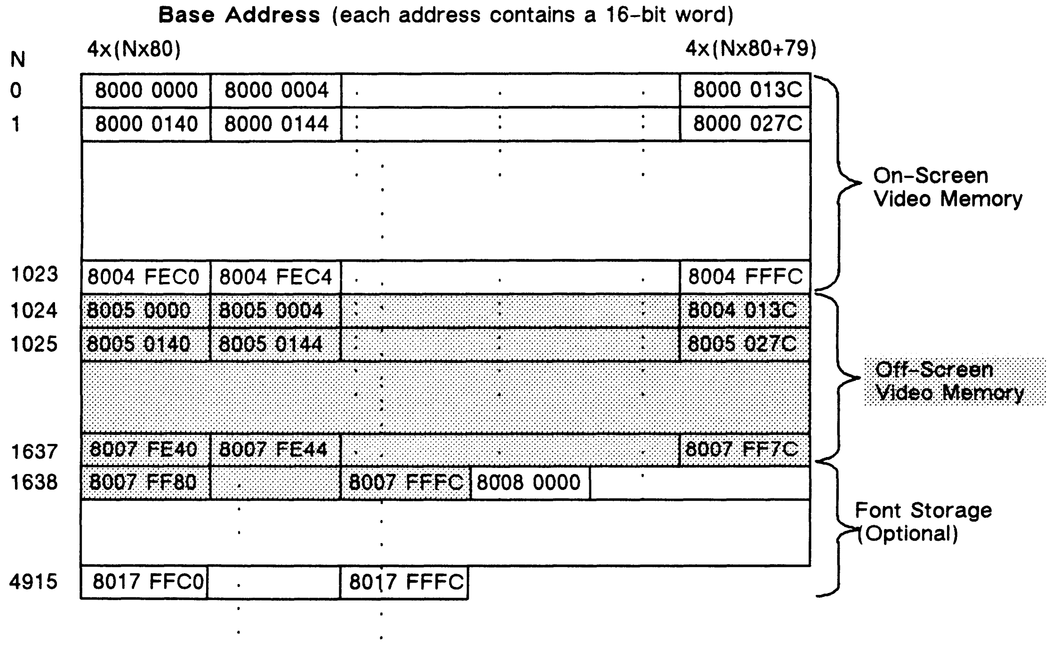
```
                    /***********   Enable Screen   ************/

FILL();
CHK_STATUS_PRE();
CHK_STATUS_PROC();                  /* Wait until screen is clear, then */
reg->display_ctrl = 0xcb30;         /* start display
                                    (activate blank signal). */
}
/* Random Line Generator — generates nn thousand random lines,
                                    half black and half white. */

void random(nn)
int nn;
{
register x1,y1,x2,y2,xmax,ymax,seed,loop,loop1,zero,p;
register val1,val2,val3,val4;
GRAPH();

zero = 0;
xmax = 1280;
ymax = 1024;
seed = 3;

val1 = 31421;
val2 = 6927;
val3 = 65535;
val4 = 16;
p = 1;

loop = nn;
while (loop-- > zero) {
        if (p > zero) {COL_MODE(ZERO_DEST); }
        else {COL_MODE(ONE_DEST); }
        loop1 = 1000;
        while (loop1-- > zero) {
            seed = (seed * val1 + val2) & val3;
            x1   = (seed * xmax) >> val4;
            seed = (seed * val1 + val2) & val3;
            x2   = (seed * xmax) >> val4;
            seed = (seed * val1 + val2) & val3;
            y1   = (seed * ymax) >> val4;
            seed = (seed * val1 + val2) & val3;
            y2   = (seed * ymax) >> val4;
            LINE(x1,y1,x2,y2);
            }
        p = -p;
        }
}
main()

{
graph_init();
random(3);
}
```

Programming the Frame Buffer

To program the frame buffer, either send parameters and drawing commands to the graphics controller, or write to the frame buffer. Direct access to the frame buffer allows you to manipulate the video image and to take advantage of existing software that assumes a dumb frame buffer. However, since direct access to the frame buffer takes place through the graphics controller, these accesses take considerably longer than main memory accesses. For example, a frame buffer read or write cycle takes about 800 ns while a main memory read or write cycle takes about 240 ns, assuming data is not in cache. Furthermore, because the graphics controller sends only the low-order 16 bits of the Mbus for data transfers, frame buffer accesses must be done in half-words (16 bits) on frame buffer word (32-bit) boundaries.

The visible screen portion of the frame buffer occupies the lower 80K by 16-bit words of the frame buffer address space. Off-screen video memory accounts for the upper 48K by 16-bits of frame buffer address space.

The starting address of the first 16-bit word in the frame buffer is 8000 0000. The last 16-bit word on the screen is at 8004 FFFC. The first 16-bit word of off-screen memory is at 8005 0000, and the last word in the frame buffer is at 8007 FFFC. Figure 4-3 shows the organization of the frame buffer.



Base Address (each address contains a 16-bit word)

N	4x(Nx80)				4x(Nx80+79)
0	8000 0000	8000 0004	.	.	8000 013C
1	8000 0140	8000 0144	:	:	8000 027C
1023	8004 FEC0	8004 FEC4	.	.	8004 FFFC
1024	8005 0000	8005 0004	:	:	8004 013C
1025	8005 0140	8005 0144	:	:	8005 027C
1637	8007 FE40	8007 FE44	.	.	8007 FF7C
1638	8007 FF80	.	8007 FFFC	8008 0000	
4915	8017 FFC0	.	8017 FFFC		

Figure 4-3 Frame Buffer Organization

End of Chapter

Chapter 5
Programming the Color Graphics Subsystem

This chapter describes the following topics:

- The color graphics subsystem.
- Handshaking between the CPU and the color graphics subsystem.
- How to program the color graphics subsystem.
- The optional Z–buffer gate array and how to program the Z–buffer registers. The 300 series stations do not support the Z–buffer.

Features of the Color Graphics Subsystem

The color graphics subsystem drives a bit–mapped color graphics monitor, and has the following features:

- Drives a 1280 x 1024 pixel display, 60 or 70 Hz noninterlaced refresh rate, compliant with U.S., Canadian, and European VDT standards.
- Can select from 256 (8–bit) or 16.7–million (24–bit) colors.
- 8–bit systems have a frame buffer of 1536 x 1024 pixels x 10 planes, and 24–bit systems have a frame buffer of 2048 x 1024 pixels x 26 planes. The planes consist of 8 or 24 color planes, plus two overlay planes. The frame buffer provides extra off–screen storage area for fonts and menus.
- Processes up to 280,000 10–pixel vectors/second (raw rate, with the CPU supplying integer coordinates as fast as possible).
- Incorporates several graphics commands such as Bit Block Transfer (BITBLT) and Line Draw (LINE) to relieve the CPU of these functions.
- Draws Gouraud–shaded polygons.
- Does windowing and pick via clipping rectangles.

Components of the Color Graphics Subsystem

This section describes the major components of the color graphics subsystem. Figure 5-1 and Figure 5-2 illustrate 8-bit and 24-bit color graphics subsystems respectively. The components are:

- Color graphics controller.
- Frame buffer (VRAM).
- RAMDAC which contains a Look Up Table (LUT) in RAM, and a Digital to Analog Converter (DAC).
- Clock generator.
- Z-buffer (optional).

The rest of this section describes each of these components.

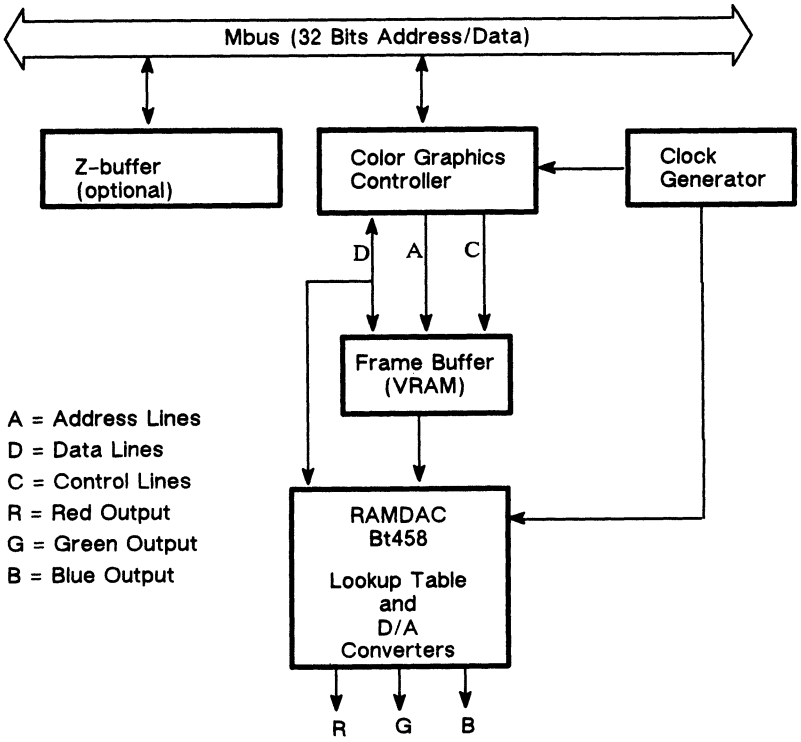


Figure 5-1 Color Graphics Subsystem (8-Bit)

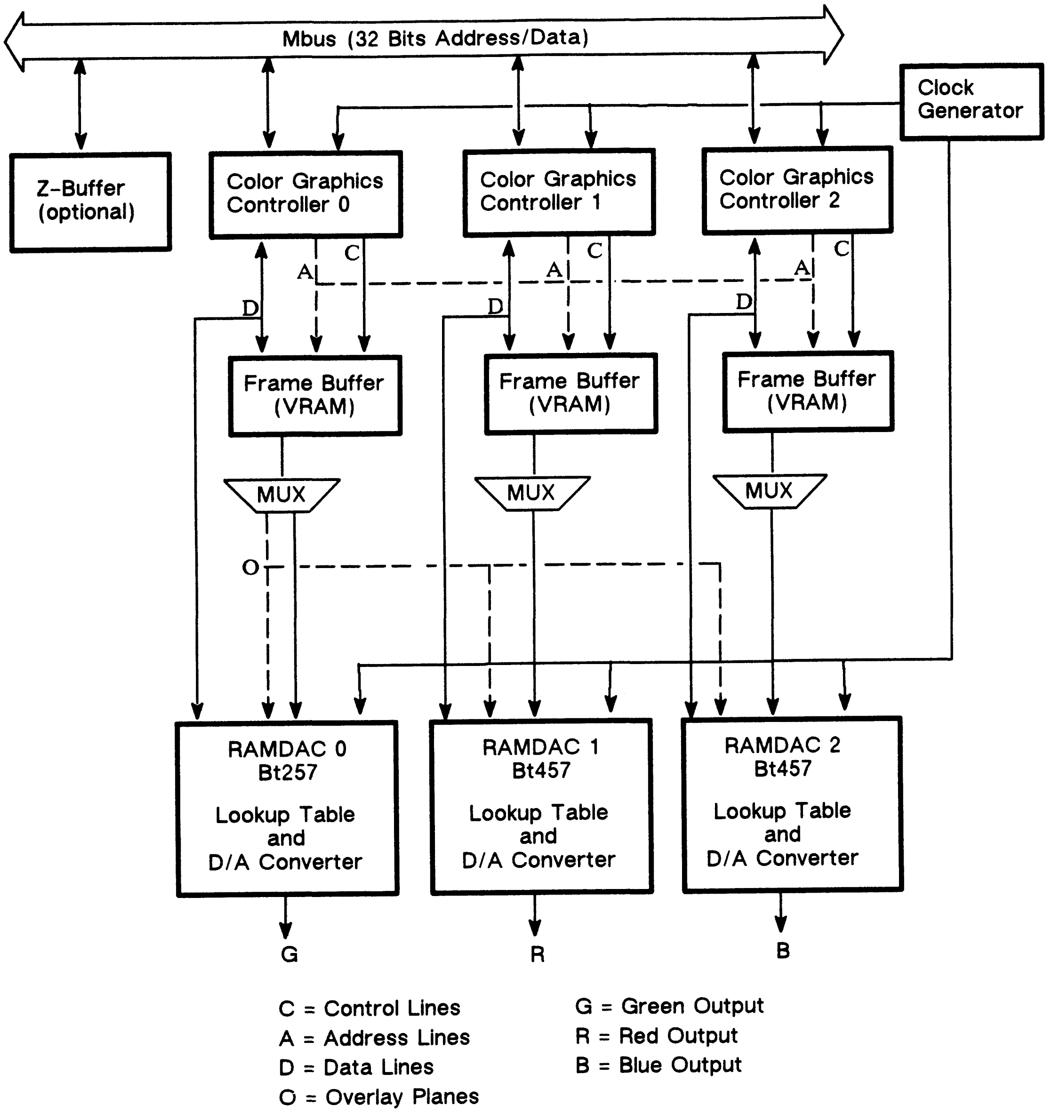

Mbus (32 Bits Address/Data)
Clock Generator
Z-Buffer (optional)
Color Graphics Controller 0
Color Graphics Controller 1
Color Graphics Controller 2
C
A
D
Frame Buffer (VRAM)
Frame Buffer (VRAM)
Frame Buffer (VRAM)
MUX
MUX
MUX
O
RAMDAC 0 Bt257 Lookup Table and D/A Converter
RAMDAC 1 Bt457 Lookup Table and D/A Converter
RAMDAC 2 Bt457 Lookup Table and D/A Converter
G
R
B
C = Control Lines
A = Address Lines
D = Data Lines
O = Overlay Planes
G = Green Output
R = Red Output
B = Blue Output

Figure 5-2 Color Graphics Subsystem (24-Bit)

The Color Graphics Controller

The color graphics controller is a gate array that manipulates addresses and data, and generates control signals for the color graphics subsystem. With minimal intervention from the CPU, this gate array can draw lines, fill in specified areas, and manipulate blocks of data. The color graphics gate array communicates with the CPU via the Mbus.

The Frame Buffer

Raster graphics systems have an array of memory called the frame buffer or bitmap in which values for pixels are stored. These pixel values are pointers into the lookup table (LUT) which is a color palette.

The frame buffer consists of Video Random-Access Memory (VRAM) which has two I/O ports; one port connects to the color graphics gate array, and the other connects to the RAMDAC.

The frame buffer is organized into planes; each plane is one bit deep. 8-bit color has 8 color planes and 2 overlay planes. 24-bit color has 24 color planes and 2 overlay planes. The color planes determine the colors displayed. 8-bit systems can generate and display 28 or 256 true colors. 24-bit color systems can generate 224 or 16.7 million true colors and display 2048 x 1024 or 2.6 million true colors.

Overlay planes temporarily store superimposed images without destroying the underlying images.

The frame buffer is located in memory within the following ranges: color data is stored at 8000 0000 - 83FF FFFF, and overlay data is stored at 8400 0000 - 87FF FFFF.

Table 5-1 identifies the sizes of frame buffers.

Table 5-1 Frame Buffer Size

Graphics	Frame Buffer
8-bit (300 and 400)	1536H x 1024V Pixels 1280H x 1024V Pixels displayed 8 Color planes + 2 Overlay planes
24-bit (400)	2048H x 1024V Pixels 1280H x 1024V Pixels displayed 8 Green planes + 2 Overlay planes 8 Red planes 8 Blue planes

RAMDAC

Both 8-bit and 24-bit color systems use Brooktree RAMDAC chips. 8-bit stations use one Brooktree Bt458 RAMDAC, and 24-bit stations use three Brooktree Bt457 RAMDACs.

The Lookup Table (LUT)

The lookup table (LUT) is a color palette; it contains values that define colors to be displayed. 8-bit systems have a single 8-bit LUT, and 24-bit systems have three 8-bit tables (one for each primary color).

The Digital to Analog Converter (DAC)

Both 8-bit and 24-bit color systems have three high-speed 8-bit Digital-to-Analog Converters (DACs). The output of each DAC drives an RS-343A interface with red, green, and blue (synchronized on green).

The Clock Generator

The clock generator consists of two oscillators. The oscillators generate 125.000 MHz (for 75-kHz color monitors) and 107.352 MHz (for 64-kHz color monitors) clocks. The power-up routine identifies which monitor is connected, and assigns an oscillator for the correct horizontal rate.

The Z-Buffer

The Z-buffer is a gate array that regulates and stores data for the Z-axis of three-dimensional images to be displayed.

Programming Conventions

This section describes programming conventions that affect more than one of the major components of the color graphics subsystem. The major topics in theis section include:

- Handshaking
- Context Switching
- Accessing color graphics resources

The color graphics subsystem is a slave device.

When the CPU accesses the color graphics subsystem, the address phase does not require extra wait states, while the data phase requires one or more wait states for accesses to the registers or frame buffer. Data phase accesses to the RAMDAC require extra wait states.

When written to, the color graphics controller ignores parity bits.

The graphics controller has built-in commands to create objects and fields, to move blocks of data within the frame buffer, and to transfer data between the frame buffer and system memory. These commands are programmed through the Command (CMD) register.

Handshaking

Handshaking between the CPU and the color graphics subsystem is regulated using bit 0 (BSY) and bit 1 (DIP) of the Control and Status Register 0 (CSR0), shown here:

Bit	Mnemonic	Function	Type
1	DIP	Drawing In Progress The color graphics controller is performing a drawing operation. Check DIP when writing parameters that cannot be pipelined. 1 Indicates the controller is executing a command. 0 Indicates the controller is not executing a command. The following conditions terminate commands: D The command completed its function and all frame buffer accesses related to it have begun. D The clipping boundary was crossed. D A context switch occurred.	Read
0	BSY	Busy Indicates whether or not the color graphics controller is busy, and whether or not the parameter registers (PARM0 - PARM15) can be loaded. 1 Indicates the controller is busy and cannot accept new parameters; i.e., do not write to the parameter registers. BSY is set when a command is executed. 0 Indicates the controller can accept new parameters; it no longer needs the existing parameters.	Read/Write

When writing to pipelined registers, make sure that BSY is cleared to 0. When writing to nonpipelined registers, make sure that DIP is cleared to 0. The ability to pipeline parameter registers (PARMn) is dependent on both the command being executed and the command to be executed. In general, do not pipeline global registers. Pipelining can be handled using two masks (one for setup and one for execution) for each command, describing the registers it needs to access. By keeping the mask for the execution part of the command currently running, and performing a logical AND of this mask and the setup mask of the next command, you can determine whether it is necessary to wait for BSY or DIP.

When using CLIP_STOP (see Command register description), you should not pipeline commands. When the clip boundary is crossed, the Clipping Boundary (BND) bit will be set and the command stopped. If you have pipelined a second command, it will immediately execute the new command and clear the BND bit.

Context Switching

Context switching is controlled via the Stop (STP) and Resume (RES) bits in the STOP register. Upon stopping execution, you must save the context by reading all the registers. The context switch operation will also function properly when pipelined commands are present. After the STP bit is set to 1, make sure that the DIP bit is cleared to 0 before reading the current context and loading a new context. If a command terminates at the same time an STP command is issued, the STATE1 register is automatically modified to appear as if the command switched out was a No Operation (NOP) command. When the new context is restarted, the NOP command will immediately terminate and subsequent commands will continue. If no new context is available, you can put the controller in an idle state by clearing the Busy (BSY) bit in CSR1, and then the STP bit in the STOP register.

NOTE: Before reading the color graphics registers, either disable parity checking or ignore the parity traps (interrupts). For information on disabling parity, see the *MC88200 User's Manual.*

In addition, bits 0 (STP) and 1 (RES) of the STOP register control context switching.

Bit	Mnemonic	Function
1	RES	Resume Resume operations following a context switch. When set to 1 by the host, directs the controller to resume execution of a previously stopped command. The controller clears the RES bit after the command continues executing. 0 No operation. 1 Restart current context; automatic clear.
0	STP	Stop Stop the current operation. 0 No operation. When cleared to 0, the CSR0 DIP bit indicates that the controller is stopped. A stop occurring simultaneously with a normal command termination is handled as a normal termination; a hardware mechanism ensures that when restarted, a "no operation" (NOP) command is executed. The STP bit is normally cleared to 0 by the controller after the controller resumes execution (RES bit). 1 When set to 1, directs the color graphics controller to stop execution at the earliest possible time. The time is command dependent: BITBLT and POLY will stop at the next end of a scan line; LINE transfers will stop at the next pixel.

Accessing Color Graphics Resources

This section describes methods of accessing resources of the color graphics subsystem. The major topic is broadcast and individual accesses.

Broadcast Accesses and Individual Accesses

The CPU accesses the color graphics registers using either a *broadcast* access or an *individual* access. A broadcast access is when the CPU writes to or reads from a register in all three controllers at the same time. An individual access is when the CPU writes to or reads from a register in only one controller. Eight-bit color systems do not distinguish between broadcast and individual accesses. In 24-bit color systems, a broadcast access writes a word of data to all three controllers, but each controller uses only the portion of the word relevant to it. During a broadcast read, each of the three controllers writes one byte to the data bus, and the CPU reads all three bytes at the same time as a word.

With a 24-bit color graphics subsystem, a broadcast address accesses a resource in the three controllers simultaneously. With an 8-bit color graphics subsystem, the broadcast address has no advantage over an individual access, but it can be used in place of the individual access (to maintain compatibility if upgraded to 24-bit color).

Table 5-2 defines the base addresses used when accessing color graphics controllers. The Position (POSN) bits in Control and Status Register 1 (CSR1) define the base address used during an individual access.

Table 5-2 Base Addresses of the Color Graphics Controllers

		Individual		
	Broadcast	**POSN 00**	**POSN 01**	**POSN 10**
Starts at	FFF8 9000	FFF8 9100	FFF8 9200	FFF8 9300
Ends at	FFF8 90FF	FFF8 91FF	FFF8 92FF	FFF8 93FF

Access Guidelines

Several factors define how to access a color graphics resource; these are

- What the resource is (a register, the frame buffer, or the LUT)
- The number of bits in the resource (8-bits or 32-bits)
- The type of color graphics subsystem (8-bit or 24-bit color)
- The type of access (broadcast or individual).

In 24-bit color systems, 8-bit registers, 32-bit registers and the frame buffer are accessed as follows:

- **8-bit registers**

  broadcast accesses:

  As shown in Figure 5-3, 8-bit registers on each color graphics controller receive from or transmit to the correct byte of the bus as determined by the controller's position. Each controller's position is defined by the Position (POSN) bits in the Control and Status Register 1 (CSR1).

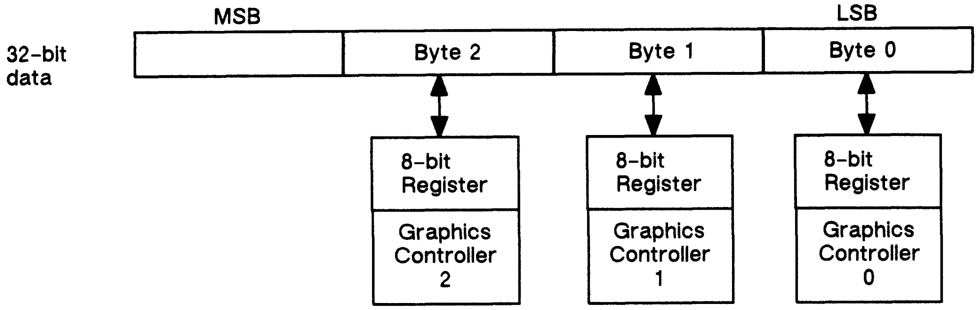


Figure 5-3 Broadcast Data Transfers of 8-bit Registers with 24-bit Color

  individual accesses:

  An 8-bit register will receive or transmit the least significant byte (bits 7-0) regardless of the POSN bits in CSR1.

- **32-bit registers**

  broadcast accesses:

  32-bit registers on each color graphics controller receive the entire 32 bits of data, but transmit the least significant byte of data in the position that corresponds to the controller's POSN value.

  individual accesses:

  32-bit registers on each color graphics controller receive or transmit the entire 32 bits of data.

- **Frame buffer**

  broadcast accesses:

  The frame buffer receives or transmits 1 byte in the byte that corresponds to the controller's POSN value.

  individual accesses:

  The frame buffer receives the byte of data that corresponds to the controller's position, but transmits a byte of data replicated four times in the data word.

Table 5–3 defines the addresses and offsets of the graphics controller registers, and also defines the addressing and context switch characteristics of the color graphics registers. The rightmost two hexadecimal digits of an address are the register's offset (e.g. 14 is the offset for the BACK register.) The "Broadcast/Individual" field indicates whether the register should be accessed using a broadcast (B) access or an individual (I) access. A Save in the Context Switch field indicates that the register should be saved during a context switch.

Table 5–3 Color Graphics Registers

Address	Broadcast/ Individual	Context Switch	Name	Function
FFF8 9000	B	Save	CSR0	Control and Status Register 0
FFF8 9004	B	Save	STOP	Stop/Resume Register
FFF8 9008	B	Save	CSR1	Control and Status Register 1
FFF8 900C	B	Save	CMD	Command Register
FFF8 9010	B	Save	MASK	Plane Mask Register
FFF8 9014	B	Save	BACK	Background Color Register
FFF8 9018	B	Save	LPAT	Line Pattern Register
FFF8 901C	B	Save	PC/WID	Pattern Control/WID Register
FFF8 9020	B	—	CRT0	CRT Control Register 0
FFF8 9024	B	—	CRT1	CRT Control Register 1
FFF8 9028	B	—	CRT2	CRT Control Register 2
FFF8 902C	—	—	—	Reserved
FFF8 9030	B	Save	STATE0	Internal State 0
FFF8 9034	B	Save	STATE1	Internal State 1
FFF8 9038– FFF8 903C	—	—	—	Reserved
FFF8 9040	B	Save	PARM0	Parameter Register 0
FFF8 9044	B	Save	PARM1	Parameter Register 1
FFF8 9048	B	Save	PARM2	Parameter Register 2
FFF8 904C	B	Save	PARM3	Parameter Register 3
FFF8 9050	B/I	Save [1]	PARM4	Parameter Register 4
FFF8 9054	B/I	Save [1]	PARM5	Parameter Register 5
FFF8 9058	B	Save	PARM6	Parameter Register 6
FFF8 905C	B	Save	PARM7	Parameter Register 7
FFF8 9060	B	Save	PARM8	Parameter Register 8
FFF8 9064	B/I	Save [1]	PARM9	Parameter Register 9
FFF8 9068	B	Save	PARM10	Parameter Register 10
FFF8 906C	B	Save	PARM11	Parameter Register 11
FFF8 9070	B/I	Save [1]	PARM12	Parameter Register 12
FFF8 9074	B/I	Save [1]	PARM13	Parameter Register 13
FFF8 9078	B/I	Save [1]	PARM14	Parameter Register 14
FFF8 907C	B	Save	PARM15	Parameter Register 15

[1] Depending on the command executing, these registers may be written to using either the broadcast address or their individual addresses. During a context switch, however, they should be read and written individually to ensure proper operation for any command.

(continued)

Table 5-3 Color Graphics Registers

Address	Broadcast/ Individual	Context Switch	Name	Function
FFF8 9080– FFF8 909C	—	—	—	Reserved
FFF8 90A0	B	—	DATA	Data Port Register
FFF8 90A4	B	—	PLT0	Palette Pointer 0
FFF8 90A8	B	—	PLT1	Palette Pointer 1
FFF8 90AC	B	—	BLNK	Blink Control Register
FFF8 90B0– FFF8 90BC	—	—	—	Reserved
FFF8 90C0	I	—	DAC0	RAMDAC Address Register
FFF8 90C4	I	—	DAC1	RAMDAC Color Palette RAM
FFF8 90C8	I	—	DAC2	RAMDAC Control Register
FFF8 90CC	I	—	DAC3	RAMDAC Overlay Palette RAM
FFF8 90D0– FFF8 90DC	—	—	—	Reserved
FFF8 90E0– FFF8 91F0	B	Save	—	Z-Buffer Control Registers (see the section, "Programming the Z-Buffer Registers.")

(concluded)

NOTE: Before reading the color graphics registers, either disable parity checking or ignore the parity interrupts. For information on disabling parity, see the *MC88200 User's Manual.*

Fixed-Point Numbers

Fixed-point numbers are required with some parameters, especially color shading parameters and many parameters associated with the POLY command. The structure of these fixed-point numbers is shown here:

Bit	Contents
31-29	Not implemented.
28-16	Most significant word (Integer Portion).
15-13	Not implemented.
12-0	Least significant word (Fractional portion).

Parameters using fixed-point numbers assign 13 bits each to integer and fractional parts. This 1/(2^13) ensures that less than one integer bit of error will accumulate when scanning as many as 8192 pixels. The range of values in this format is from -8192.0 to +8191.999.

To convert a floating-point value to fixed-point, do the following:

1. Multiply the floating-point number by 8192.
2. Convert the floating-point number to integer.
3. Shift bits 13 through 25 of the result three bits to the left.

Interrupts

The color graphics subsystem may interrupt the CPU when a drawing is completed, a drawing is being written outside the clipping area, or a vertical blank started.

When an interrupt occurs, read the Control and Status Register 1 (CSR1); it identifies the cause of interrupts. The interrupts can also be masked through bits in the CSR1 register; if an interrupt is masked, the color graphics controller will not pass the interrupt request to the interrupt control logic.

Since the graphics subsystem does not flag error conditions, the color graphics program must ensure that the CPU sends correct parameters to the subsystem.

Registers

Each graphics controller has registers that provide a variety of functions from setting up and executing graphics commands to passing and identifying graphics interrupts.

The registers are memory mapped on 32-bit word boundaries. Some of the registers use all 32-bits; others use only 8 bits. All registers must be accessed as words, therefore the programmer must be careful to read or write the correct bits.

The graphics subsystem has two sets of registers – visible and working – as illustrated in Figure 5-4. The operating system communicates with the visible registers. The working registers are copies of the visible registers and are accessed only by the graphics subsystem. The graphics subsystem writes the contents of the visible registers into the working registers as needed (i.e., the registers are *pipelined*.) Except in some cases, the visible registers can be programmed with new data while the subsystem is executing a command using the "original" data.

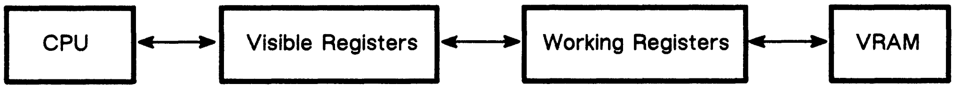


Figure 5-4 Graphics Subsystem Registers

NOTE: To ensure compatibility with future hardware revisions, write 0s to all reserved or unimplemented bits. When reading registers, ignore the reserved bits.

The color graphics controller contains the following registers:

Global Registers:

MASK	Plane Mask register
BACK	Background Color register
LPAT	Line Pattern register
PC_WID	Pattern Control/Window ID register
DATA	Dataport register

The global registers specify parameters affecting drawn pixel values, line patterning, and plane masking. They are not pipelined during a draw command, therefore do not modify their contents until the Drawing In Progress (DIP) bit in CSR0 is 0, indicating that drawing is inactive.

The POLY command uses all registers in a nonpipelined way, so that you must test the DIP bit.

Command and Status Registers:

CSRn	Control and Status Registers 0 and 1
CRTn	CRT Timing registers 0 through 2
STATEn	Internal State registers 0 and 1
STOP	Stop register
PARMn	Parameter registers 0 through 15
CMD	Command register

Global Registers

The next few pages describe the global registers in detail.

BACK Background Color

Address FFF8 9014 Read/Write

The Background Color (BACK) register contains the background color value for use during patterned line and stippled draw operations. During these operations, a line pattern or stipple bit with a value of 0 will select the BACK register as the source for the pixel value to be written into the frame buffer. The BACK register is viewed as an N-bit register where N is the number of frame buffer planes implemented. Bit 0 corresponds to pixel bits for frame buffer plane 0; bit 1 for frame buffer plane 1, and so on.

You should normally access the BACK register using its broadcast address. In multiple color graphics controller configurations, this allows you to read or write the 8-bit BACK register of each controller with a single register access.

31	16
BACK	

15	0
BACK	

Bit	Mnemonic	Function
31-0	BACK	Background Color Contains the background color value for use during patterned line and stippled draw opoerations.

DATA Dataport

Address FFF8 90A0 Read/Write

The Dataport (DATA) register is used by the color graphics controller during data transfers (RXFER and WXFER commands). The controller repeatedly reads from and writes to this register during block transfers between host memory and video memory. During a transfer, you must write or read all pixels requested before starting any other command, otherwise it will be necessary to reset or stop the color graphics controller.

This register is viewed as an N–bit register where N is the number of frame buffer planes implemented in the system. Bit 0 corresponds to pixel bits for plane 0; bit 1 for plane 1, and so on. With the STIPPLE bit set and a WXFER command active, only bit 0 of this register is used. In this case, writing a 0 or 1 to this bit results in a background (0) or foreground (1) pixel value being transferred to the frame buffer. At reset time, the contents of this register are unknown and unchanged.

31	16
DATA	

15	0
DATA	

Bit	Mnemonic	Function
31–0	DATA	Dataport Dataport is the data transport interface for block transfers between host memory and video memory.

MASK — Plane Mask

Address FFF8 9010 — Read/Write

The Plane Mask (MASK) register provides selective writing of frame buffer planes. With this function the color graphics frame buffer can be partitioned into logical plane groups, each of which can be independently modified. The MASK register is viewed as an N-bit register where N is the number of frame buffer planes implemented. Bit 0 controls the masking for plane 0; bit 1 for plane 1, and so on. A bit value of 0 disables writing to the associated plane; a bit value of 1 enables writing to that plane.

You should normally access the MASK register using its broadcast address. In multiple color graphics configurations, this allows you to read or write the 8-bit MASK register of each controller with a single register access.

31		16
	MASK	

15		0
	MASK	

Bit	Mnemonic	Function
31-0	MASK	Plane Mask Masks and enables the buffer planes

LPAT Line Pattern

Address FFF8 9018 Read/Write

The Line Pattern (LPAT) register contains the 32-bit line pattern in effect during line drawing. Bit 0 of the LPAT register represents the first pixel in a line (if the PAT_RESET bit in the Command register is set to 1). Each bit in the pattern contains a 0 for background color or a 1 for foreground color. Note that you may draw a solid line either by using an FFFF FFFF pattern in the LPAT register or by setting the SOLID bit in the Command register to 1.

You should normally access the LPAT register using its broadcast address. In multiple color graphics controller configurations, this ensures that each controller uses an identical line pattern. However, by using the individual unit number addressing, you load a different 32-bit pattern into each controller, thus providing multicolored line patterning.

31	16
LPAT	

15	0
LPAT	

Bit	Mnemonic	Function
31-0	LPAT	Line Pattern Contains the 32-bit line pattern in effect during line drawing.

PC_WID — Pattern Control/Window ID

Address FFF8 901C — Read/Write

The Pattern Control/Window ID (PC_WID) register controls line patterning operations and specifies window ID clipping parameters. The Pattern Control bits (PTCNT, CRPTCNT, and PTPNT) save and restore the context of a "stopped" Line command. The Window ID bits (WIDKEY, WIDMSK) are valid only if the WID_ENABLE bit in the Command register is set. During a drawing operation, a pixel will be modified if the corresponding WID mask matches the WIDKEY bits.

You should normally access the PC_WID register using its broadcast address. In 24-bit graphics systems, this ensures that each controller uses identical line pattern controls. The WIDMSK and WIDKEY bits of the register are significant only for controllers configured as masters (slave controllers ignore these bits).

31 27	26 22	21 19	18 16
Reserved	PTPNT	CRPTCNT	PTCNT

15 8	7 0
WIDKEY	WIDMSK

Bit	Mnemonic	Function
31-27	Reserved	Must be zeroes when written to, and undefined when read from.
26-22	PTPNT	Pattern pointer Points to the current bit within the line pattern.
21-19	CRPTCNT	Current pattern repeat count.
18-16	PTCNT	Pattern repeat count. Allows positive scaling of the current line pattern. The value corresponds to each pattern bit being repeated 1-8 times. Bits Operation 000 Normal line drawing, no stretch 001 x2 stretch 010 x3 stretch 011 x4 stretch 100 x5 stretch 101 x6 stretch 110 x7 stretch 111 x8 stretch
15-8	WIDKEY [1]	Window key Contains the value to be compared with the ID read from the ID/overlay planes during a read-modify-write (RMW) frame buffer cycle. If the unmasked bits read from the ID planes match those of the WIDKEY, the RMW cycle will replace the existing pixel (no clip), otherwise the RMW cycle will retain the existing pixel (clipping occurs).
7-0	WIDMSK	Window ID mask Specifies which ID/overlay planes to use during the clipping process.

[1] Unused bits must be 0, for instance, workstations supporting 2 bits of window ID information use only WIDKEY bits 9 and 8 and WIDMSK bits 1 and 0.

Command and Status Registers

The next few pages describe the command and status registers in detail.

CSR0 Control and Status Register 0

CSR0 Address FFF8 9000 Read/Write

The Control and Status register 0 (CSR0) returns the state of drawing instruction execution.

31	16
Reserved	

15	3	2	1	0
Reserved		BND	DIP	BSY

Bit	Mnemonic	Function	Type
31-3	Reserved	Must be zeroes when written to.	
2	BND	Clipping Boundary The previous command crossed the clipping boundary. 1 Indicates the previous command crossed clipping boundary and the CLIP_STOP bit (Command register) was set. 0 Indicates that either the previous command did not cross clipping boundary, or the previous command crossed clipping boundary and the CLIP_STOP bit (Command register) was not set.	Read/Write
1	DIP	Drawing In Progress The color graphics controller is performing a drawing operation. Check this bit when writing to registers that are not pipelined. 1 Indicates that the controller is executing a command. 0 Indicates that the controller is not executing a command.	Read
0	BSY	Busy Color graphics controller is busy. The execution of a command sets this bit. The color graphics controller clears this bit when the controller no longer needs the parameters. 1 Indicates that the controller is busy and cannot accept new parameters. 0 Indicates that the controller can accept new parameters.	Read/Write

CSR1 Control and Status Register 1

CSR1 Address FFF8 9008 Read/Write

The Control and Status Register 1 (CSR1) returns the state of the color graphics subsystem as well as information on configuration, interrupts, and timing.

31	30 23	22 20	19	18	17	16
NMN	Reserved	CBR	VBL	VSN	HSN	BtS

15	14	13	12	11	10	9	8	7	6 5	4 2	1 0
MST	CLI	CLM	DDI	DDM	VBI	VBM	IMK	WST	POSN	FBSIZE	PSIZE

Bit	Mnemonic	Function	Type
31	NMN	Control Signals Enable When written to, it enables or disables the frame buffer control signals. When read from, it indicates whether the control signals are enabled or disabled. 0 Write disables the signals. Read indicates that the signals are disabled. 1 Write enables the signals. Read indicates that the signals are enabled.	Read/Write
30–23	Reserved	Must be zeroes when written to.	
22–20	CBR	CAS Before RAS Indicates the number of CBR refresh cycles per scan line. **CBR No. of Cycles CBR No. of Cycles** 000 0 100 4 001 1 101 5 010 2 110 6 011 3 111 7	Read
19	VBL	Vertical Blank Indicates the Current status of the vertical blank (VBLANK) signal. 0 VBLANK is active. 1 VBLANK is inactive.	Read
18	VSN	Vertical Sync Indicates the current status of the vertical sync (VSYNC) signal. 0 VSYNC is active. 1 VSYNC is inactive.	Read
17	HSN	Horizontal Sync Indicates the current status of the horizontal sync (HSYNC) signal. 0 HSYNC is active. 1 HSYNC is inactive.	Read
16	BtS	Type of DAC and LUT Indicates the type of RAMDAC being used. 0 8-bit color uses a Broktree Bt458 RAMDAC and 24-bit color uses Brooktree Bt457 RAMDACs.	Read

(continued)

Bit	Mnemonic	Function	Type
15	MST	Master or Slave Indicates whether this is master or slave controller (multiple color graphics subsystems only). 0 Slave controller. 1 Master controller.	Read
14	CLI	Clipping Interrupt Indicates whether or not pixels have been clipped. 0 Pixels have not been clipped. 1 One or more pixels have been clipped.	Read/Write
13	CLM	Clipping Interrupt Mask Enables/disables the clipping interrupt. 0 Disables clipping interrupt (CLI). 1 Enables clipping interrupt (CLI).	Read/Write
12	DDI	Drawing Done Interrupt. Indicates whether or not the color graphics controller has completed a drawing operation. Also see the Drawing In Progress (DIP) bit in CSR0. 0 The controller has not completed a drawing operation. 1 The controller has completed a drawing operation.	Read/Write
11	DDM	Drawing Done Mask Enables/disables the drawing done interrupt. 0 Disables interrupt when drawing done. 1 Enables interrupt when drawing done.	Read/Write
10	VBI	Vertical Blank Interrupt. Indicates whether or not a vertical blank has occurred. 0 Vertical blank has occurred. 1 Vertical blank has not occurred.	Read/Write
9	VBM	Vertical Blank Mask Enables/disables the vertical blank interrupt. 0 Disables vertical blank interrupt (VBI). 1 Enables vertical blank interrupt (VBI).	Read/Write
8	IMK	Interrupt Mask Enables/disables all interrupt requests. 0 Disables interrupts. 1 Enables interrupts.	Read/Write
7	WST	Wait States Indicates that an additional wait state is inserted into the Mbus data phase. 0 Extra wait state. 1 No extra wait state.	Read
6-5	POSN	Position Color graphics controller base address. Defines the position and address of the graphics controller. If 8-bit color, POSN is set to 00. If 24-bit color, POSN is set to 00 for red, 01 for blue, and 10 for green. 00 FFF8 9100 01 FFF8 9200 10 FFF8 9300	Read

(continued)

Bit	Mnemonic	Function	Type
4–2	FBSIZE	Frame Buffer Size Type and size of frame buffer. 0002K x 1K, 64K x 4 VRAM 0011K x 512, 64K x 4 VRAM 0102K x 1K, 256K x 4 VRAM 0112K x 2K, 256K x 4 VRAM 1004K x 2K, 256K x 4 VRAM	Read
1–0	PSIZE	Pixel Size Number of bits/pixel. 00 8 bits per pixel 10 24 bits per pixel NOTE: PSIZE is set up during power–up reset only.	Read

(concluded)

CRT0, CRT1, CRT2 — CRT Timing

CRT0	**Address FFF8 9020**	**Read/Write**
CRT1	**Address FFF8 9024**	**Read/Write**
CRT2	**Address FFF8 9028**	**Read/Write**

The CRT Timing registers contain parameters for the composite sync signal and composite blank signal. These signals can be programmed to support a variety of screen resolutions (noninterlaced display only). Horizontal timing is programmable in units of 1/8 of the pixel clock rate (8 pixel times = 1 Hunit). Vertical timing is programmable in units of horizontal rasters. Some of the CRT Timing register values cross register boundaries.

CRT0 Register (FFF8 9020)

31 — 24	23 — 16
HTOTAL	HBSTRT

15	14 — 7	6 — 0
HBSTRT	HBEND	HSEND

Bit	Mnemonic	Function
31-24	HTOTAL	Horizontal Total Number of Hunits from the start of a horizontal sync signal to the start of the next horizontal sync signal (the line period). Program this value with 2 less than the required number of Hunits. CRT0 contains the lower 8 bits; CRT1 contains the top bit.
23-15	HBSTRT	Horizontal Blank Start Number of Hunits from the start of a horizontal sync signal to the end of the viewable display area (start of horizontal blank signal). Program this value with 3 less than the required number of Hunits.
14-7	HBEND	Horizontal Blank End Number of Hunits from the start of a horizontal sync signal to the start of the viewable display area (end of horizontal blank signal). Program this value with 3 less than the required number of Hunits.
6-0	HSEND	Horizontal Start to End Number of Hunits from the start of a horizontal sync signal to the end of the horizontal sync signal. Program this value with 2 less than the required number of Hunits.

CRT1 Register (FFF8 9024)

31 25	24 16
VBSTRT	VBEND

15 13	12 1	0
VBEND	VSEND	HTOTAL

Bit	Mnemonic	Function
31-25	VBSTRT	Viewable Area Start to End Number of rasters from the start of the viewable area to the end of the viewable area. Program this value with 1 less than the required number of rasters plus the number in VADJ. CRT1 contains the 7 lower bits; CRT2 contains the 5 top bits.
24-13	VBEND	Vertical Period Number of rasters from the start of the viewable area to the start of the next viewable area (the vertical period). Program this value with 1 less than the required number of rasters plus the number in VADJ.
12-1	VSEND	Vertical Start to End Number of rasters from the start of the viewable area (the end of VBLANK) to the end of the next vertical sync signal. Program this value with the required number of rasters plus the number in VADJ.
0	HTOTAL	Total Horizontal Sync Signal Number of Hunits from the start of horizontal sync signal to the start of the next horizontal sync signal (the line period). Program this value with 2 less than the required number of Hunits. CRT0 contains the lower 8 bits; CRT1 contains the top bit.

CRT2 Register (FFF8 9028)

31	30	29	28	17	16
FRCBLK	ENSYNC	Reserved	VADJ		See below VTOTAL

16	5	4	0
VTOTAL		VBSTRT	

Bit	Mnemonic	Function
31	FORCBLNK*	Force Blank Bit. 0 Forces the composite blank signal from the color graphics controller to below, thus blanking the screen. 1 Enables screen display. Note that VRAM refresh and drawing operations will continue even if the display is being blanked. A reset clears this bit.
30	ENSYNC	Enable Sync Bit. When set to 1, enables the sync signal and blank generation within the color graphics controller. Set this bit to 1 to enable timing only after programming the CRT registers. A reset clears this bit.
29	Res	Reserved. This bit should always be 0 (a reset clears this bit).
28-17	VADJ	Vertical Adjust Number loaded into an internal counter prior to the start of the first displayed raster. This value determines which row within the VDRAMs is transferred to the VDRAM serial port in preparation for the first line of active video. The other vertical timing parameters depend on this value (normally 0).
16-5	VTOTAL	Vertical Total Number of rasters from the start of the viewable area to the start of the next vertical sync signal. Program this value with the required number of rasters plus the number in VADJ.
4-0	VBSTRT	Viewable Start to End Number of rasters from the start of the viewable area to the end of the viewable area. Program this value with 1 less than the required number of rasters plus the number in VADJ. CRT1 contains the 7 lower bits; CRT2 contains the 5 top bits.

The following example calculates values for a 70–Hz color graphics monitor.

The timing values for this monitor are

Horizontal resolution	1280 pixels
Vertical resolution	1024 rasters
Pixel clock (125 MHz)	
(timing generation (in Hunits))	15.625 MHz = 64 ns
Horizontal line period	13.312 ms = 208 Hunits
Horizontal front porch	0.512 ms = 8 Hunits
Horizontal sync	1.024 ms = 16 Hunits
Horizontal back porch	1.536 ms = 24Hunits
Horizontal display active	1280 pixels = 160 Hunits
Vertical period	1071 rasters total
Vertical front porch	1 raster
Vertical sync	4 rasters
Vertical back porch	42 rasters
Vertical display active	1024 rasters

Using these values (with VADJ = 0) we calculate the following numbers for the timing parameters:

HSEND	horizontal sync width - 2 = 16 - 2 = 14
HBEND	hsync + horizontal back porch - 3 = 16 + 24 - 3 = 37
HBSTRT	hsync + back porch + display active - 3 = 16 + 24 + 160 - 3 = 197
HTOTAL	horiz line period - 2 = 208 - 2 = 206
VSEND	vert active + front porch + vsync = 1024 + 1 + 4 = 1029
VBEND	vert period - 1 = 1071 - 1 = 1070
VBSTRT	vert active time - 1 = 1024 - 1 = 1023
VTOTAL	vert active + back porch = 1025
VADJ	0

These registers can be programmed in any order. When programming, the ENSYNC and FORCBLNK bits in CRT2 should be low (0). After programming, you should set these bits to 1 leaving the other bits unchanged.

STATE0, STATE1 Internal State

STATE0 **Address FFF8 9030** **Read Only**

STATE1 **Address FFF8 9034** **Read Only**

The Internal State registers contain current values for drawing operations. STATE0 contains the current BITBLT mask, direction and shift values. STATE1 contains a copy of the Command register. Both registers must be saved and restored at context switch time, but otherwise are not programmer-visible. When reset, the contents of STATE0 and STATE1 are unknown and unchanged.

STATE0

31		16
	STATE0	

15		0
	STATE0	

Bit	Mnemonic	Function
31–0	STATE0	Internal State 0 Contains the current bit block transfer (BITBLT) mask, direction and shift values.

STATE1

31		16
	STATE1	

15		0
	STATE1	

Bit	Mnemonic	Function
31–0	STATE1	Internal State 1 Contains a copy of the contents of the Command (CMD) register.

STOP Stop

Address FFF8 9004 Read/Write

The Stop register controls context switching.

31	16
Reserved	

15	3	2	1	0
Reserved		RST	RES	STP

Bit	Mnemonic	Function
31-3	Reserved	Must be zeroes when written to, and undefined when read from.
2	RST	Reset Reset the color graphics controller. 0 No operation. 1 Perform a software reset and go to the idle state.
1	RES	Resume Resume operations following a context switch. 0 No operation. 1 Restart current context; automatic clear.
0	STP	Stop Stop the current operation. 0 No operation. 1 Stop the current drawing operation and go to the idle state.

PARM0–PARM15 Parameter

PARM0	Address FFF8 9040	Read/Write
PARM1	Address FFF8 9044	Read/Write
PARM2	Address FFF8 9048	Read/Write
PARM3	Address FFF8 904C	Read/Write
PARM4	Address FFF8 9050	Read/Write
PARM5	Address FFF8 9054	Read/Write
PARM6	Address FFF8 9058	Read/Write
PARM7	Address FFF8 905C	Read/Write
PARM8	Address FFF8 9060	Read/Write
PARM9	Address FFF8 9064	Read/Write
PARM10	Address FFF8 9068	Read/Write
PARM11	Address FFF8 906C	Read/Write
PARM12	Address FFF8 9070	Read/Write
PARM13	Address FFF8 9074	Read/Write
PARM14	Address FFF8 9078	Read/Write
PARM15	Address FFF8 907C	Read/Write

The Parameter (PARMn) registers supply parameters for the color graphics commands. After programming the Command register, program the parameter registers to supply parameters for the command to be executed. The function of the parameter registers varies from command to command.

The parameter registers may be grouped into two categories, the initial parameter registers and the working registers. The initial parameter registers supply the graphics controller with setup information used when executing a command; the values in these registers remain unchanged throughout the execution of the command. The working registers are changed by the graphics controller as needed when executing a command. This is not visible because the graphics controller changes the values of the background parameter registers; the foreground registers are left unchanged.

Resets do not affect the contents of the parameter registers.

CMD Command

Address FFF8 900C Read/Write

The Command register specifies a command to be executed by the color graphics controller. The commands are

LINE	Draws straight lines, solid or patterned, single–color or shaded within a clipping rectangle.
CLINE	Draws straight lines, solid or patterned, without clipping.
POLY	Draws a flat–topped triangle using Gouraud–shading or a solid color, and pattern and stipple options.
BITBLT	Moves a rectangular area of pixels within the frame buffer. The "Attributes" field specifies patterning and stippling options. BITBLT is also used to draw text; a portion of the frame buffer should contain characters that can be copied.
RXFER, WXFER	Transfer pixels between the host memory and the frame buffer. Pixels are transferred in either Z–mode or in a limited XY–mode (write transfers only) with the least significant bit selecting either background or foreground colors.

31 16
Attributes

15 12	11 8	7 4	3 0
Reserved	LU_OP	Reserved	OP

Bit	Mnemonic	Function
31–16	Attributes	Specifies options applicable to the operation (see bits 3–0, Opcode).
	Bit	**Mnemonic — Operation**
	31	Reserved
	30	SHADE Applies Gouraud shading to LINE and POLY drawing. 1 Apply shading. 0 Do no apply shading.
	29	NO_LAST When drawing a line, prevents the last pixel of the line from being drawn. A single–pixel line with this bit set will have no pixels drawn. This function is useful when drawing polylines with the XOR ALU function. 1 Do not draw last pixel. 0 Draw last pixel.

(continued)

Bit	Mnemonic		Function
31–16	Attributes		Continued
		Bit	**Mnemonic — Operation**
		28	PAT_RESET During line operations causes the pattern pointer to reset to the first bit of the pattern for every new line. 1 Reset to first bit of the pattern. 0 Do not modify pattern pointer (useful for polylines).
		27	ZBUF_ENABLE Enables the Z–buffer interface with the Z–buffer co–processor. During a BITBLT operation, if the SOLID bit (bit 23) is also set, this selects the Z–buffer "fast clear" mode. Only 400 series stations support Z–buffer operations. 1 Enables the Z–buffer interface. 0 Disables the Z–buffer interface.
		26–24	SOURCE_PLANE Selects one of eight planes for use as the source plane during stippled operations.
		23	SOLID During BITBLT operations forces skipping the read of the source (optimizes the BITBLT command for clear and fill operations). 1 Skip reading of the source. 0 Read the source.
		22	TRANSPARENT During a stippled or line operation uses the source data or the line pattern to either modify the bit (when data is 1) or rewrite the bit unmodified (when data is 0). 1 Perform TRANSPARENT operation. 0 Do not perform TRANSPARENT operation.
		21	STIPPLE During BITBLT, POLY, and transfer operations, forces source area to be treated as a one–plane bitmap rather than a pixmap. 1 Perform STIPPLE. 0 Do not perform STIPPLE.
		20	AREA_PATTERN Forces the source data during BITBLT and POLY operations to be treated as a 32 x 32 area pattern. The stored pattern must be aligned so that the upper–left corner has the 5 least significant bits as 0, both in X and Y. You can program the initial offset into the pattern, as specified in the respective commands, to make alignment relative to screen, window, or object. 1 Perform AREA_PATTERN. 0 Do not perform AREA_PATTERN.
		19	WID_ENABLE Enables clipping using the overlay planes. To use this function, the PC_WID register must first be programmed with the appropriate plane mask and key. WID is available on all operations. 1 Enable clipping with the overlay planes. 0 Disable WID_ENABLE.

(continued)

<table>
<tr><th>Bit</th><th>Mnemonic</th><th>Function</th></tr>
<tr><td>31–16</td><td>Attributes</td><td>Continued
<table>
<tr><th>Bit</th><th>Mnemonic — Operation</th></tr>
<tr><td>18</td><td>CLIP_STOP
During line operations, selects whether execution stops or continues when the clipping boundary is crossed.
<table>
<tr><td>1</td><td>Stop and set Clipping Boundary (BND in register CSR0) when clipping occurs.</td></tr>
<tr><td>0</td><td>Do not stop when clipping occurs.</td></tr>
</table></td></tr>
<tr><td>17–16</td><td>CLIP_CONTROL
Determines how clipping is used.
<table>
<tr><td>00</td><td>Draw inside clipping rectangle.</td></tr>
<tr><td>01</td><td>Do not clip.</td></tr>
<tr><td>10</td><td>Draw outside clipping rectangle (pick).</td></tr>
<tr><td>11</td><td>Do not clip.</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>15–12</td><td>Reserved</td><td>Must be zeroes when written to, and undefined when read from.</td></tr>
<tr><td>11–8</td><td>LU_OP</td><td>ALU Operation
Specifies the type of ALU operation to be performed between the source and destination data.
<table>
<tr><th>Value
(Hex)</th><th>Mnemonic</th><th>Logical Function</th></tr>
<tr><td>0</td><td>CLEAR</td><td>0 (zero)</td></tr>
<tr><td>1</td><td>AND</td><td>source AND destination</td></tr>
<tr><td>2</td><td>AND REVERSE</td><td>source AND NOT destination</td></tr>
<tr><td>3</td><td>COPY</td><td>source</td></tr>
<tr><td>4</td><td>AND INVERTED</td><td>NOT source AND destination</td></tr>
<tr><td>5</td><td>NOP</td><td>destination</td></tr>
<tr><td>6</td><td>XOR</td><td>source XOR destination</td></tr>
<tr><td>7</td><td>OR</td><td>source OR destination</td></tr>
<tr><td>8</td><td>NOR</td><td>NOT source AND NOT destination</td></tr>
<tr><td>9</td><td>EQUIV</td><td>NOT source XOR destination</td></tr>
<tr><td>A</td><td>INVERT</td><td>NOT destination</td></tr>
<tr><td>B</td><td>OR REVERSE</td><td>source OR NOT destination</td></tr>
<tr><td>C</td><td>COPY INVERTED</td><td>NOT source</td></tr>
<tr><td>D</td><td>OR INVERTED</td><td>NOT source OR destination</td></tr>
<tr><td>E</td><td>NAND</td><td>NOT source OR NOT destination</td></tr>
<tr><td>F</td><td>SET</td><td>1 (one)</td></tr>
</table></td></tr>
<tr><td>7–4</td><td>Reserved</td><td>Must be zeroes when written to, and undefined when read from.</td></tr>
<tr><td>3–0</td><td>OP</td><td>Operation
Specifies the operation to be executed.
<table>
<tr><th>Value
(Hex)</th><th>Operation</th><th></th></tr>
<tr><td>0</td><td>NOP</td><td>No operation</td></tr>
<tr><td>1</td><td>BITBLT</td><td>Bit block transfer</td></tr>
<tr><td>2</td><td>LINE</td><td>Draw line</td></tr>
<tr><td>3</td><td>CLINE</td><td>Continue line after clip</td></tr>
<tr><td>4</td><td>POLY</td><td>Polygon assist</td></tr>
<tr><td>5</td><td>Reserved</td><td></td></tr>
<tr><td>6</td><td>RXFER</td><td>Transfer from frame buffer to host</td></tr>
<tr><td>7</td><td>WXFER</td><td>Transfer from host to frame buffer</td></tr>
<tr><td>8–F</td><td>Reserved</td><td></td></tr>
</table></td></tr>
</table>

(concluded)

Table 5-4 describes several Command register bit settings. Within the table, the dash (—) indicates that the value of this bit does not affect the command.

Table 5-4 Color Graphics Command Bits

				Command Register Bits					
Command	Stipple Bit [1]	Line Ptn Bit	Mbus Bit	LU_OP	SOLID	STIPPLE	TRANS	Data Source[2]	Description
LINE	—	0	—	ACT [3]	0	—	0	BACK	Normal draw
	—	1	—	ACT	0	—	0	FORE	
	—	0	—	ACT	0	—	1	DEST	Draw transparent
	—	1	—	ACT	0	—	1	FORE	
	—	—	—	—	1	—	—	FORE	Draw solid (pattern = —)
POLY	—	—	—	ACT	0	0	—	FORE	Normal POLY
	—	—	—	—	1	—	—	FORE	Solid fill POLY
	0	—	—	ACT	0	1	0	BACK	Stipple mode
	1	—	—	ACT	0	1	0	FORE	
	0	—	—	ACT	0	1	1	DEST	Stipple mode
	1	—	—	ACT	0	1	1	FORE	(transparent)
BITBLT	—	—	—	ACT	0	0	—	SORC	Normal BITBLT
	—	—	—	—	1	—	—	FORE	Solid fill
	0	—	—	ACT	0	1	0	BACK	Stipple mode
	1	—	—	ACT	0	1	0	FORE	
	0	—	—	ACT	0	1	1	DEST	Stipple mode
	1	—	—	ACT	0	1	1	FORE	(transparent)
W/RXFER	—	—	—	ACT	0	0	—	DPORT	Z-Mode
(Z-Mode)	—	—	—	—	1	0	—	DPORT	Z-Mode
WXFER	—	—	0	ACT	0	1	0	BACK	XY-Mode
(XY-mode)	—	—	1	ACT	0	1	0	FORE	(Mbus bit 0)
	—	—	0	ACT	0	1	1	DEST	Transparent XY-Mode
	—	—	1	ACT	0	1	1	FORE	(Mbus bit 0)
	—	—	—	—	1	1	—	FORE	XY-Mode (fill solid)
WXFER	—	—	—	—	—	—	—	MDATA	Host [1] frame buffer

[1] Stipple Bit:
POLY: Single bit selected from a one-plane stipple pattern.
BITBLT: Multiple-bit word selected from a one-plane stipple pattern.

[2] Data Source: Frame buffer write "data source"; this data may be modified by the LU opcode.
BACK = Background color.
FORE = Foreground color.
DEST = Pixel data prefetched from frame buffer.
DPORT = Dataport Register.
SORC = BITBLT source data.
MDATA = Host-to-frame-buffer write data.

[3] ACT: Datapath logic unit "active"; source data is subject to modification according to current ALU operation code (LU_op).

Color Graphics Commands

This section describes the color graphics commands.

LINE Line draws a straight line within clipping boundaries.

CLINE Continue Line After Clip draws a straight line that was previously drawn by LINE, but CLINE ignores clipping boundaries.

POLY Polygon draws a filled polygon.

BITBLT Bit Block Transfer moves blocks of data within the frame buffer.

RXFER Read Transfer transfers data from the frame buffer to system memory.

WXFER Write Transfer transfers data from system memory to the frame buffer.

To execute a graphics command, the graphics program must write to the command and parameter registers. When the CPU writes the last parameter, the subsystem executes the requested command without further CPU intervention.

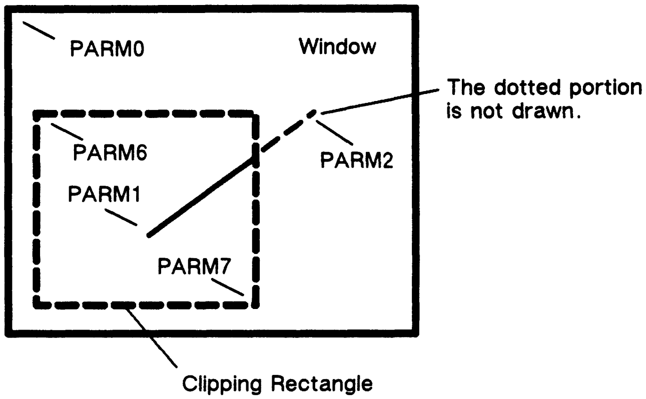
LINE

Line Draw

The LINE command draws straight lines, solid or patterned, single–color or shaded within a clipping rectangle. This command uses a modified Bresenham's Algorithm to ensure that lines drawn from point A to point B exactly match lines drawn from point B to point A. LINE directly accepts the X–Y coordinates of a source and destination point, together with the X–Y origin they are identified with.

Command capabilities include

- The use of a pattern and pattern control registers to control stippling of each pixel in the line.
- The use of 16 standard logic operations, as defined by the X Window System™.
- Optionally drawing the last pixel in the line.
- Shaded or solid color lines.
- WID clipping.



Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM0	Origin_X	Origin_Y	Upper-left coordinates of drawing window.
PARM1	Source_X	Source_Y	Line starting coordinates.
PARM2	Dest_X	Dest_Y	Line ending coordinates.
PARM3		Fore	Foreground pixel color for nonshaded lines.
PARM4	Fore_I	Fore_F	Initial foreground color for shaded lines.
PARM5	Finc_I	Finc_F	Foreground color increment for shaded lines.
PARM6	ClipTL_X	ClipTL_Y	Upper-left coordinates of clipping rectangle.
PARM7	ClipBR_X	ClipBR_Y	Lower-right coordinates of clipping rectangle.

NOTE _I = Integer and _F = Fraction.

Working Registers

Register	MSBytes	LSBytes	Description
PARM8	CP_X	CP_Y	Current pointer (coordinates of the last pixel drawn).
PARM9	Delta_X	Delta_Y	Current number of pixels in the X and Y direction.
PARM10	Error_X	Error_Y	Bresenham's Error in X and Y direction.
PARM11	Einc_X	Einc_Y	Bresenham's Error increment in X and Y direction.
PARM14	Fcurr_I	Fcurr(_F)	Current foreground color — In shaded lines, the MSB contains the integer part; in nonshaded lines, the LSB is the same as the initial foreground color value (in PARM4).

The LINE command calculates the next pixel address using a modified Bresenham's Algorithm. PARM8 (current pointer) contains the last pixel drawn, and at the end of the command will equal origin+destination.

Global Registers

The LINE command uses the following global registers:

BACK

MASK

LPAT

PC_WID

Command Register

The LINE command uses the following Command register bits.

Bit	Mnemonic
30	SHADE
29	NO_LAST
28	PAT_RESET
27	ZBUF_ENABLE
26-24	SOURCE_PLANE
23	SOLID
22	TRANSPARENT
19	WID_ENABLE
18-16	Attributes: CLIP_CONTROL (17-16), CLIP_STOP (18)
11-8	LU_OP
3-0	OP = 2

The NO_LAST attribute leaves the CP at the destination pixel, but does not draw it. If PAT_RESET is set, the pattern pointer and counter will reset to the first pattern bit before drawing any pixel; otherwise, the line will be drawn with a pattern which is a continuation of the previous command.

Executing the LINE Command

Follow this procedure to execute the LINE command:

1. If it is necessary to update the global registers, PARM7, PARM6 or PARM4, continue with steps 1a and 1b, otherwise continue with step 2.
   a. Poll the DIP bit in CSR0, wait for it to clear to 0.
   b. Write the necessary parameters to the global registers (MASK, BACK, LPAT and PC_WID), PARM7, PARM6 and PARM4.
2. Poll the BSY bit in CSR0 for 0.
3. Program the Command register.
4. Program PARM0, PARM1 and PARM3.
5. Program PARM2; the command will execute automatically.

Interrupts

When the controller completes the LINE command, it generates a drawing done interrupt (DDI).

If clipping is enabled and a pixel is drawn outside the clipping boundaries, the controller generates a clip interrupt. This depends on the clip control attributes in the CMD register; when the CLIP_STOP bit is set, line drawing stops and the BND bit in CSR0 is set.

Notes/Exceptions

If you specify the NO_LAST function, the pattern pointer will not increment for the pixel not drawn. This is consistent with the desired functionality.

Although the XY frame buffer and the WID/overlay frame buffer share a common coordinate system (with the WID/overlay below the XY and starting at X = 0, Y = 4096), lines must be drawn either completely in the XY or in the WID/overlay sections. That is, no lines can be specified that extend from the XY into WID/overlay section or vice-versa.

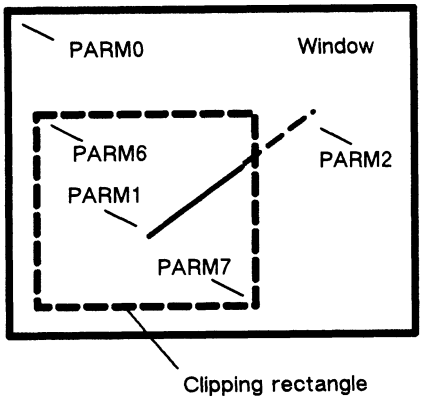
You can not pipeline the Finc (foreground color increment) parameter in PARM5 for shaded lines.

If transparency is enabled, transparent pixels are not drawn, that is, no memory cycle is executed for them. In 400 series stations, if Z-buffering is enabled, transparency takes precedence over Z-buffering, resulting in transparent pixels retaining their old Z-value, regardless of the Z-comparison.

CLINE
Continue Line After Clip

Continue Line After Clip (CLINE) redraws a line that was previously drawn using the LINE command, but was cut by clipping boundaries. CLINE needs no no new parameters, and the setup phase of the LINE command is not executed.

CLINE draws straight lines in a window that is partially obscured. Every time the line steps outside the current boundaries, the host should update PARM6 and PARM7 with the next rectangular area that the line goes into, and issue a CLINE command. Since there is no knowledge of whether the line will step outside of the window, or into which area it will go next, the CLINE command should not be pipelined.



Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM6	ClipTL_X	ClipTL_Y	Upper-left coordinates of clipping rectangle.
PARM7	ClipBR_X	ClipBR_Y	Lower-right coordinates of clipping rectangle.

Working Registers

Register	MSB	LSB	Description
PARM8	CP_X	CP_Y	Current pointer (coordinates of the last pixel drawn).
PARM9	Delta_X	Delta_Y	Current number of pixels in the X and Y direction.
PARM10	Error_X	Error_Y	Bresenham's Error in X and Y direction.
PARM11	Einc_X	Einc_Y	Bresenham's Error increment in X and Y direction.
PARM14	Fcurr_I	Fcurr(_F)	Current foreground color — In shaded lines, the MSB contains the integer part; in nonshaded lines, the LSB is the same as the initial foreground color value (in PARM4).

Global Registers

The CLINE command uses the following global registers:

MASK

BACK

LPAT

PC_WID

Command Register

The CLINE command uses the following Command register bits.

Bit	Mnemonic
30	SHADE
29	NO_LAST
27	ZBUF_ENABLE
23	SOLID
22	TRANSPARENT
19	WID_ENABLE
18-16	Attributes: CLIP_CONTROL (17-16), CLIP_STOP (18)
11-8	LU_OP
3-0	OP = 3

Modes of Operation

The CLINE command calculates the next pixel address using a modified Bresenham's Algorithm. The CP (PARM8) register contains the last pixel to be drawn, and at the end of the command will equal origin+destination. The pattern register is always referred to, so solid lines must have the pattern of FFFF FFFF. The same argument applies to shaded lines. The NO_LAST attribute leaves the CP at the destination pixel, but does not draw it.

Command Procedure

Follow this procedure to execute the CLINE command:

1. Wait for the DIP bit in CSR0 to clear and the BND bit in CSR0 to be set.
2. Update the global registers and Parameter registers (PARM6 and PARM7). PARM6 and PARM7 are the clipping boundaries.
3. Poll the BSY bit in CSR0; wait for it to clear to 0.
4. Write the CLINE opcode, logic opcode, and desired attributes into the Command register. The Command register triggers the operation.
5. To execute another LINE command, repeat the procedure from Step 1. To execute another graphics command see the appropriate command procedure.

Interrupts

When the controller completes the CLINE command, it generates a drawing done interrupt (DDI).

If clipping is enabled and a pixel is (possibly) drawn outside (or inside) the clipping boundaries, the controller generates the clip interrupt. (This is dependent on the setting of the clip control attributes.) If the CLIP_STOP bit is also set, line drawing stops and the BND bit in CSR0 is set.

Notes/Exceptions

If you specify the NO_LAST function, the pattern pointer will not increment for the pixel not drawn.

Although the XY frame buffer and the WID/overlay frame buffer share a common coordinate system (with the WID/overlay below the XY and starting at X = 0, Y = 4096), lines must be drawn either completely in the XY or in the WID/overlay sections. That is, no lines can be specified that extend from the XY into WID/overlay section or vice-versa.

You can not pipeline the Finc (foreground color increment) parameter in PARM5 for shaded lines.

If transparency is enabled, transparent pixels are not drawn; that is, no memory cycle is executed for them. If Z-buffering is also enabled, transparency takes precedence over Z-buffering, resulting in transparent pixels retaining their old Z-value, regardless of the Z-comparison. Note that only 400 series stations support Z-buffering for hidden surface removal.

POLY
Polygon Assist

The POLY command draws a filled polygon. The polygon must be a trapezoid (i.e., a four-sided figure with two parallel sides) with the parallel sides on horizontal planes. Two trapezoids are needed to draw a triangle (see Figure 5-5).

Command capabilities include

- Flat shading and Gouraud shading (linear interpolation).
- Stippling and transparency.
- WID clipping.
- Any combination of the above.

The following figure illustrates the global parameters that must be defined when drawing a polygon.

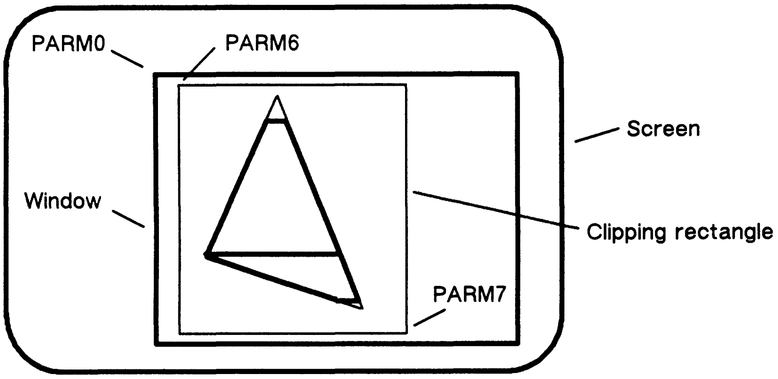
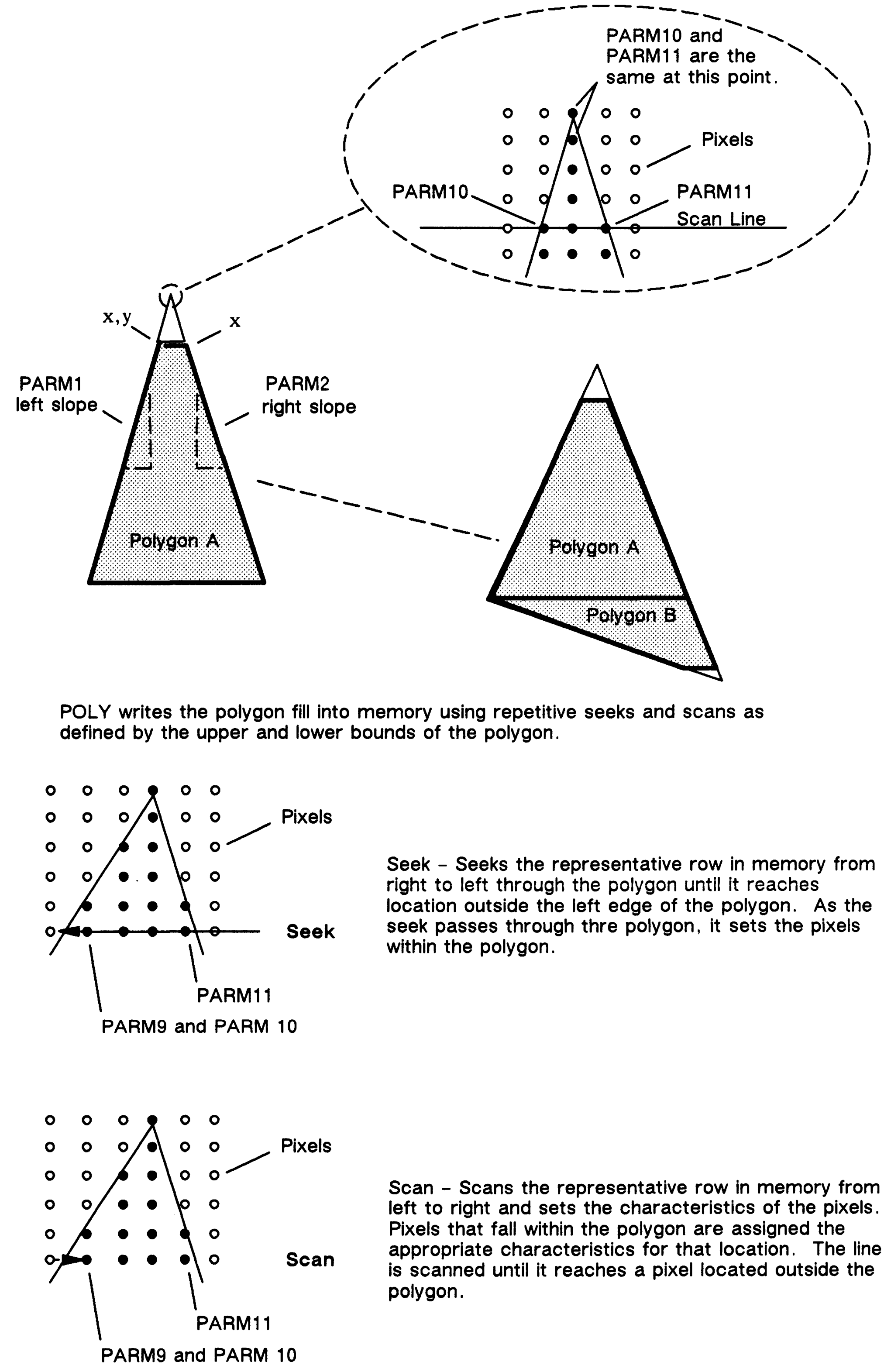


Figure 5-5 Global Elements of the POLY Command

Figure 5-6 illustrates some of the local parameters used in drawing a polygon. PARM9, the initial scan, defines where in the clipping rectangle to begin the scan. The graphics controller increments PARM9 after each scan to scan down through the clipping rectangle. If the top of the polygon is clipped, PARM9 should start with the same value as PARM6 (shown in Figure 5-5), the top left corner of the clipping rectangle.



POLY writes the polygon fill into memory using repetitive seeks and scans as defined by the upper and lower bounds of the polygon.

Seek - Seeks the representative row in memory from right to left through the polygon until it reaches location outside the left edge of the polygon. As the seek passes through thre polygon, it sets the pixels within the polygon.

Scan - Scans the representative row in memory from left to right and sets the characteristics of the pixels. Pixels that fall within the polygon are assigned the appropriate characteristics for that location. The line is scanned until it reaches a pixel located outside the polygon.

Figure 5-6 Local Elements of the POLY Command

Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM0	Origin_X	Origin_Y	Upper-left coordinates of drawing window.
PARM1	DX1_I	DX1_F	Slope of left edge (dx/dy).
PARM2	DX2_I	DX2_F	Slope of right edge (dx/dy).
PARM3	—	Flat	Foreground pixel color for flat shading (unused, integer).
PARM4	DIX_I	DIX_F	Intensity change for a positive step in the X-direction.
PARM5	DIY_I	DIY_F	Intensity change for a positive step in the Y-direction.
PARM6	ClipTL_X	ClipTL_Y	Top-left coordinates of clipping rectangle.
PARM7	ClipBR_X	ClipBR_Y	Bottom-right coordinates of clipping rectangle.
PARM9	Sp_X	Sp_Y	Initial scan leftmost pointer (X, Y).
PARM10	E1_I	E1_F	X-value of left edge in initial scan.
PARM11	E2_I	E2_F	X-value of right edge in initial scan.
PARM13	Is_I	Is_F	Intensity at scan pointer (Sp).
PARM14	nn_1	nn_2	Number of scans in left and right edges (integers).
PARM15	Patrn_X	Patrn_Y	Stipple pattern pointer (X, Y).

NOTE _I = Integer and _F = Fraction when shown under MSB and LSB.

Working Registers

Register	MSBytes	LSBytes	Description
PARM8	Cp_X	Cp_Y	Current pixel pointer (X, Y).
PARM9	Sp_X	Sp_Y	Current scan leftmost pointer (X, Y).
PARM10	E1_I	E1_F	X-value of the left edge in current scan.
PARM11	E2_I	E2_F	X-value of the right edge in current scan.
PARM12	Ic_I	Ic_F	Intensity at current pixel.
PARM13	Is_I	Is_F	Intensity at scan pointer (Sp).

NOTE _I = Integer and _F = Fraction when shown under MSB and LSB.

Global Registers

The POLY command uses the following global registers:

BACK

MASK

PC_WID (WID fields only)

Command Register

The POLY command uses the following Command register bits.

Bit	Mnemonic
30	SHADE
27	ZBUF_ENABLE
26-24	SOURCE_PLANE
23-21	Fill style bits: SOLID (23), TRANSPARENT (22), STIPPLE (21)
19	WID_ENABLE
18-16	Attributes: CLIP_CONTROL (17-16), CLIP_STOP = 0 (18)
11-8	LU_OP
3-0	OP = 4

Modes of Operation

To calculate addresses for pixels inside the triangle, the POLY command steps through every scan line and executes first a seek, and then a scanning phase. During the seek phase, the Sp (PARM9) pointer (with intensity Is) is first moved left until it points to the pixel just to the left of the left edge, and then right, until it is at the first pixel with an X-value greater than the left edge (at the same time, the intensity is adjusted). The Cp (PARM8) is assigned the value of Sp, and scanning starts, incrementing the X-value of Cp and the intensity Ic (PARM12) until all pixels to the left or at the right edge are drawn. When the end of the scan line is reached, the Y-value of Sp is incremented, and Is adjusted, and nn_1 and nn_2 (PARM14) are decremented. If either nn_1 or nn_2 become 0, execution stops. Otherwise, the edge pointers E1 (PARM10) and E2 (PARM11) are adjusted by DX1 (PARM1) and DX2 (PARM2), and the next scan line is then drawn.

If stippling is enabled, every pixel will be drawn in either the foreground color (flat or shaded) or the background color (transparent). The Pattern pointer must point to a tile aligned in a 32 ◈ 32 boundary. The 5 least significant bytes of the calculated pixel address are added (modulo 32) to the pattern pointer, and then the SOURCE_PLANE is used to select a bit in the pattern pixel, which then selects foreground or background.

Command Procedure

Follow this procedure to execute the POLY command:

1. Poll the DIP bit in CSR0; wait for it to clear to 0.
2. Write the POLY opcode, logic opcode, and desired attributes into the Command register.
3. Write the POLY operands into the Parameter registers PARM0-PARM7, PARM9-PARM11, and PARM13-PARM15. Write to PARM2 last; it triggers the command. PARM5-PARM7 are optional.

4. For the second half of the triangle, wait for the DIP bit in CSR0 to clear; then update PARM1, PARM2, PARM10, PARM11, and PARM13. Write to PARM2 last; it triggers the command.
5. To execute another POLY command, repeat the procedure from step 1. To execute another graphics command see the appropriate command procedure.

Interrupts

When the controller completes the POLY command, it generates a drawing done interrupt (DDI).

If clipping is enabled and a pixel is (possibly) drawn outside (or inside) the clipping boundaries, the controller generates the clip interrupt. (This is dependent on the setting of the clip control attributes.)

Notes/Exceptions

If transparency is enabled, transparent pixels are not drawn; that is, no memory cycle is executed for them. If Z–buffering is also enabled, transparency takes precedence over Z–buffering, resulting in transparent pixels retaining their old Z–value, regardless of the Z–comparison. Note that only 400 series workstations support Z–buffering for hidden surface removal.

The CLIP_STOP bit must be 0 for this command.

The SOLID bit will override the SHADE bit and flat shading will occur.

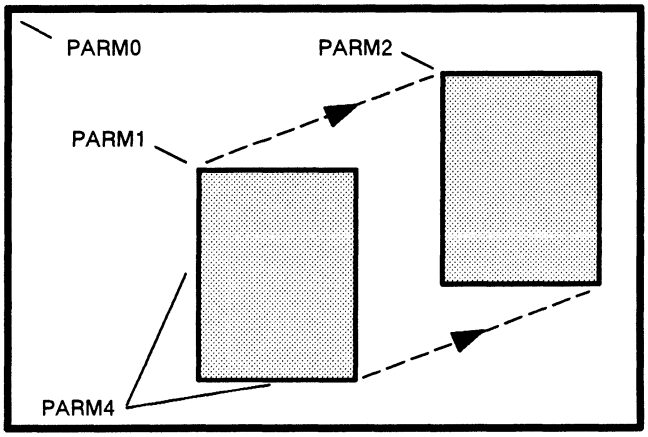
BITBLT
Bit Block Transfer

The BITBLT command moves blocks of data within the frame buffer. The transfer parameters are upper-left coordinates of the source and destination areas and the height and width of the areas. The color graphics controller computes other parameters.

BITBLT can copy from font tables, icons, and patterns stored in an off-screen portion of the frame buffer, and can set window ID and/or color overlay values in the ID/overlay frame buffer planes.

Command capabilities include

- The use of 16 standard logic operations, as defined by the X Window System.
- Fast area fills with selectable pixel values.
- 32 pixel x 32 pixel patterns.
- Color expansion (stippling) from a single selectable source plane into multiple planes, with or without patterning.
- Screen-door transparency in stipple mode.
- Clipping under control of Window ID planes.



Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM0	Origin_X	Origin_Y	Upper-left coordinates of drawing window.
PARM1	Source_X	Source_Y	Upper-left coordinates of BITBLT source rectangle.
PARM2	Dest_X	Dest_Y	Upper-left coordinates of BITBLT destination rectangle.
PARM3		Fore	Foreground pixel color (for solid fills and foreground color during stippling).
PARM4	Size_X	Size_Y	Width and height of BITBLT rectangle.

Working Registers

Register	MSBytes	LSBytes	Description
PARM8		NRows	Number of rows remaining to be transferred.
	CurWr		Number of pixels words remaining to be written to the current destination row.
PARM9		Color	Copy of foreground color register.
	CurRd		Number of pixels words remaining to be read from the current source row.
PARM10	CurSrc_X	CurSrc_Y	Current source transfer coordinates.
PARM11	CurDst_X	CurDst_Y	Current destination transfer coordinates.
PARM12	WRwds		Number of destination pixel words to be written to each row.
PARM13	RDwds		Number of source pixel words to be read from each row.
PARM14	TXsrc_X	TXsrc_Y	Translated coordinates of source rectangle after adjusting for scan direction.
PARM15	TXdst_X	TXdst_Y	Translated coordinates of destination area rectangle after adjusting for scan direction.

Global Registers

The BITBLT command uses the following global registers:

BACK

MASK

PC_WID (WID fields only)

Command Register

The BITBLT command uses the following Command register bits:

Bit	Mnemonic
27	ZBUF_ENABLE
26-24	SOURCE_PLANE
23-21	Fill style bits: SOLID (23), TRANSPARENT (22), STIPPLE (21)
20	AREA_PATTERN
19	WID_ENABLE
11-8	LU_OP
3-0	OP = 1

Modes of Operation

The Command register defines which of four modes the BITBLT command uses — Normal, Area_fill, Stipple, or Pattern.

Normal Used when none of the Command register bits is set. This mode modifies pixel values within the destination rectangle according to the current logic unit operation code (LU_OP). The pixel values written are a Boolean function of source and destination pixel values.

Area_fill Used when the SOLID bit is set. Results in the destination rectangle being filled with the pixel color value specified in the least significant byte of PARM3 (FORE). The STIPPLE, TRANSPARENT, and LU_OP bits are ignored. Area_fill operations do not need to access source data.

Stipple Used when the STIPPLE bit is set. Expands the color of a single-bit plane into multiple-bit planes. Select the source plane to be expanded with the 3-bit SOURCE_PLANE field of the Command register. The pixel values used for color expansion are supplied by the PARM3 (FORE), and BACK registers. The FORE and BACK colors correspond to 1s and 0s respectively, as read from the source plane. Similarly, with the STIPPLE and TRANSPARENT bits set, a "screen-door" transparency will result by using existing destination data in place of the BACK color for source plane 0s.

In multiple color graphics workstations, each controller must have a copy of the stipple pattern in one of its frame buffer planes. Patterning and LU_Op bits affect the operation of this mode; and the SOLID bit must be clear.

Pattern The BITBLT destination rectangle is modified based on a source pattern, normally stored in an off-screen portion of the frame buffer. The source patterns are restricted to a 32 pixel x 32 pixel rectangle and must be aligned to 32-pixel frame buffer boundaries in both X and Y, such that screen coordinate bits X4–X0 and Y4–Y0 must equal 0. Patterns are full-depth pixel values; with the STIPPLE bit set, single-plane patterns can be used as sources for stippling, using FORE and BACK as described in the previous Stippling description. The STIPPLE, TRANSPARENCY, and LU_Op bits affect the pattern mode;, and the SOLID bit must be cleared to 0. Patterns within the destination rectangle can be aligned by specifying the appropriate address of the source pattern.??????????? By specifying the upper-left coordinate of the source, alignment of patterns will be on a per object basis, whereas specifying the appropriate point within the pattern will result in alignment on a screen basis.

Command Procedure

Follow this procedure to execute the BITBLT command:

1. If necessary, update the global registers as follows, otherwise continue with step 2.
   a. Poll the DIP bit in CSR0; wait for it to clear to 0.
   b. Update the global registers; then proceed to step 2.
2. Poll the BSY bit in CSR0; wait for it to clear to 0.
3. Program the Command register with the BITBLT opcode, logic opcode, and attributes.
4. Program the Parameter registers (PARM0–PARM4). Write to PARM2 last; it executes the BITBLT command.
5. To execute another BITBLT command, repeat the procedure from step 1. To execute another graphics command see the appropriate command procedure.

Interrupts

When the controller completes a bit block transfer, it generates a drawing done interrupt (DDI).

Notes/Exceptions

The BITBLT command can move text to from tables stored in an off–screen portion of the frame buffer. To achieve optimum character transfer rates, align the characters in the frame buffer on 8–pixel column boundaries so that the screen coordinates X2–X0 = 0.

During a BITBLT, the XY Clipping mode is not available. However, Window ID clipping is supported.

Setting the Command register Z_Enable bit does not affect BITBLT operations. If the SOLID bit is also set, the combination selects the Z–buffer gate array "fast Clear" mode to quickly set the Z–buffer to a desired value.

BITBLT operations will fail if a source area starts within 40 pixels of a bank boundary in the X direction. The bank boundary occurs every 512 horizontal pixels for 64K x 4 VRAMs and every 2K horizontal pixels for 256K x 4 VRAMs.

RXFER
Read Transfer

The RXFER command controls the transfer of data from the frame buffer to the Mbus. To perform a read transfer:

1. Specify the height, width, and upper-left coordinates of the transfer area.
2. Read the resulting data from the Dataport register (DATA).
3. The graphics controller generates the frame buffer addresses and memory cycles, prefetches the requested data, and awaits the subsequent Mbus access.

Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM0	Origin_X	Origin_Y	Upper-left coordinates of drawing window.
PARM2	Pntr_X	Pntr_Y	Upper-left coordinates of transfer rectangle.
PARM4	Size_X	Size_Y	Width and height of transfer rectangle.

Working Registers

Register	MSBytes	LSBytes	Description
PARM8	Cadrs_X	Cadrs_Y	Current address of frame buffer read data.
PARM9	Csize_X	Csize_Y	Current size of remaining transfer area.
PARM10	Xent		Number of pixels per row of transfer area.
PARM11	Left_X		X address at left edge of transfer area.
PARM12	Ladrs_X	Ladrs_Y	Address of last fetched frame buffer word (used as current address when resuming command).
PARM13	Lsize_X	Lsize_Y	Size of transfer rectangle prior to last frame buffer read (used as current size when resuming command).

Global Registers

The RXFER command uses only the DATA register.

Command Register

The RXFER command uses the following Command register bits.

Bit	Mnemonic
3-0	OP = 6

Modes of Operation

The RXFER command has no special modes of operation. Once the command is initiated, the requested frame buffer data can be immediately read from the Dataport register. You do not need to poll the BSY or DIP bits in CSR0. The read data ordering will be from left-to-right, top-to-bottom within the specified transfer area. Each read access will result in one data bit per memory plane; data will be right justified within the word, such that bit 0 corresponds to plane 0, bit 1 to plane 1, and so on. Note that you can not pipeline RXFER commands. Once the command is triggered, it must complete (DIP = 0) before you issue another command.

Command Procedure

You should follow this procedure for proper operation of the RXFER command:

1. Poll the BSY bit in CSR0 for 0.
2. Write the RXFER opcode into the Command register.
3. Write the Parameter registers PARM4, PARM2, and PARM0 with RXFER operands. (PARM2 is the command "trigger" and must be written last.)
4. Read the requested frame buffer data from the Dataport register. All specified words must be read, otherwise it will be necessary to reset the color graphics controller to terminate the command.
5. Wait for the DIP bit in CSR0 to clear before executing the next command.

Interrupts

When the controller completes the RXFER command, it generates a drawing done interrupt (DDI).

Notes/Exceptions

You must read all pixels requested before starting any other command, otherwise it will be necessary to reset or stop the color graphics controller.

WXFER
Write Transfer

The Write Transfer (WXFER) command controls the transfer of data from system memory to the frame buffer. The host application first specifies the height, width, and upper-left coordinate of the target frame buffer area, then writes the required data into the Dataport register. The controller generates the actual frame buffer addresses and memory cycles and performs buffered writes of the Mbus data. The operation of the WXFER command resembles that of a BITBLT command with the host rather than the frame buffer as the data source. Accordingly, most of the BITBLT drawing modes are available.

Initial Parameter Registers

Register	MSBytes	LSBytes	Description
PARM0	Origin_X	Origin_Y	Upper-left coordinates of drawing window.
PARM2	Pntr_X	Pntr_Y	Upper-left coordinates of transfer rectangle.
PARM3		Fore	Foreground color value (used in XY-mode).
PARM4	Size_X	Size_Y	Width and height of transfer rectangle.
PARM6	ClipTL_X	ClipTL_Y	Upper-left coordinates of clipping rectangle.
PARM7	ClipBR_X	ClipBR_Y	Lower-right coordinates of clipping rectangle.

Working Registers

Register	MSBytes	LSBytes	Description
PARM8	Cadrs_X	Cadrs_Y	Current address of frame buffer write data.
PARM9	Csize_X	Csize_Y	Current size of remaining transfer area.
PARM10	Xent		Number of pixels per row of transfer area.
PARM11	Left_X		X address at left edge of transfer area.
PARM13	Lsize_X	Lsize_Y	Size of transfer rectangle after the last frame buffer write (used as current size when resuming command).

Global Registers

The WXFER command uses the following global registers:

BACK

DATA

MASK

PC_WID (WID fields only)

Command Register

The WXFER command uses the following Command register bits.

Bit	Mnemonic
23-21	Fill style bits: SOLID (23), TRANSPARENT (22), STIPPLE (21)
19	WID_ENABLE
17, 16	Attribute: CLIP_CONTROL
11-8	LU_OP
3-0	OP = 7

Modes of Operation

Depending on the setting of certain bits in the Command register, the WXFER command operates in one of four modes — Z-mode, XY-mode (without transparency, XY-mode (with transparency), and Clipping.

The following details apply to all modes of the WXFER command. Once a WXFER command is triggered, the Mbus data can be immediately written into the Dataport register. You do not need to repeatedly poll the CSR0 BSY or DIP bits. You can not pipeline WXFER commands. Once the command is triggered, it must complete (DIP = 0) before you issue another command.

Z-MODE

In this mode the frame buffer is written with data words supplied by the Mbus. The write data ordering will be from left-to-right, top-to-bottom within the specified transfer area. The BACK and PARM3 foreground colors are ignored and need not be specified in this mode. Each Mbus write will result in one data bit written per memory plane; data will be right-justified within the word, so that bit 0 corresponds to plane 0, bit 1 to plane 1, and so on.

XY-MODE (without transparency)

Activate this mode by setting the STIPPLE bit in the Command register. The value of data words written to the frame buffer will be specified by the Mbus data bit 0: where a value of 1 and 0 correspond to the FORE and BACK color registers respectfully as data sources. Mbus data bits 31-1 are ignored in this mode.

XY-MODE (with transparency)

Activate this mode by setting the STIPPLE and TRANSPARENT bits in the Command register. As in the nontransparent mode, the value of data words written to the frame buffer will be specified by the Mbus data bit 0: where a value of 1 corresponds to the FORE color register as the data source; a value of 0, however, specifies that the original data at the destination remain unmodified. Mbus data bits 31-1 are ignored in this mode.

CLIPPING

Clearing the CLIP_ENABLE bit in the Command register will turn on the clip function; the CLIP_IN/OUT bit in the Command register controls clipping relative to the defined clip rectangle. Clipped Mbus data words are flushed within the color graphics controller, and if enabled, a Clip Interrupt will be generated at this time. Note that the STOP_CONTINUE function (Command register) is ignored, and that the BND bit in CSR0 has no significance during execution of the WXFER command.

Command Procedure

You should follow this procedure for proper operation of the WXFER command:

1. Poll the BSY bit in CSR0 for 0.
2. Write the WXFER opcode and associated parameters into the Command register.
3. Write the Parameter registers PARM4, PARM2, PARM0 (defining transfer area), PARM3 (foreground color — for XY-mode only), PARM7 and PARM6 (clip rectangle — only with clipping enabled) with RXFER operands. (Note that PARM2 is the command "trigger" and must be written last.)
4. Perform successive Mbus writes to the Dataport register. All specified words must be written, otherwise it will be necessary to reset the color graphics controller to terminate the command.
5. Wait for the DIP bit in CSR0 to clear before executing the next command.

Interrupts

The WXFER command generates drawing done (DDI) and Clip interrupts. The DDI interrupt is asserted after transfer of the last word from the host. The Clip interrupt will be generated when clipping is enabled and a clip boundary is encountered during frame buffer writing.

Notes/Exceptions

You must write all pixels requested before starting any other command, otherwise it will be necessary to reset or stop the color graphics controller.

Programming the Frame Buffer (8–bit)

A graphics program may write to the frame buffer. If configured to do so, the graphics subsystem can manipulate the frame buffer directly.

The frame buffer is accessed as 32 bit words.

The color graphics controller addresses the frame buffer via eight multiplexed row and column addresses lines. Four RAS lines select banks; each bank is 512 columns of 1024 pixels. The controller writes data to the frame buffer via a 64-bit data path. In 8-bit systems, this data represents eight 8-bit pixels. In 24-bit systems, this data represents three 24-bit pixels.

The frame buffer is a contiguous block of memory, organized into two sections that representing the 4K x 4K pixels. One section is color data and the other includes overlay data window identification (WID) data. Color data occupies addresses 8000 0000 – 83FF FFFF, and overlay/window ID data occupies addresses 8400 0000 – 87FF FFFF.

The RAMDAC merges the overlay and window ID information to provide 8 bits/pixel plus 2 bits/pixel overlay and WID information.

The frame buffer is 2048 x 1024 pixels with 8 bits/pixel in the color data area and 2 bits/pixel in the overlay/WID area. Only the leftmost 1280 x 1024 pixels in this 1536 x 1024 area actually appear on the screen. The "extra" 256 x 1024 x (8+2) bits that are not displayed are available for storing the following:

- Font information (including the Kanji character set).
- Stipple patterns.
- Tile patterns.
- Palette (LUT) information.

Two bits of overlay/WID are for

- Cursor.
- Pop-up menus.
- Clipping to nonrectangular window boundaries.

The CPU can access the frame buffer only in 32-bit words. All other accesses are disallowed and may generate unusable data. Since the 8-bit and 2-bit portions of the 10-bit pixels are at different memory addresses, the CPU cannot access both at the same time. When accessing words, pixels are packed one per access with the 8 bits of the pixel in the least significant byte of the word. This is generally known as *Z-format*.

Accessing the Frame Buffer

The frame buffer memory consists of two blocks, each with as much as 4K pixels x 4K rasters. The size depends on the system configuration.

Each block is implemented as 1536 pixels x 1024 rasters, with the first 1280 pixels of each raster being displayed.

Each pixel address may be calculated as

address = base address + (((Y * 4096) + X) * 4)

where X and Y are the screen coordinates of the pixel (X and Y are 0 at the top left pixel and increment across (X) and down (Y) the screen.

Frame buffer accesses must be word (32-bit) accesses aligned on word boundaries. Each access reads or writes data for one pixel. The unused bits are ignored when read or driven onto the Mbus (write) to allow parity checking. Unimplemented bits are driven with data from the least significant Mbus byte.

Read and write frame buffer accesses on the Mbus operate directly on all planes. The graphics registers do not affect the data when the CPU reads from or writes to the frame buffer.

Frame Buffer Access Restrictions

Before you can access the frame buffer you must disable the data cache for the frame buffer memory block. For information on disabling the cache, see the *MC88200 User's Manual.*

A data transfer must be complete before you access the frame buffer.

Accessing the frame buffer slows drawing and auto LUT operations. If you continually and rapidly perform frame buffer accesses, this may cause the auto LUT option to not complete during VBLANK.

Programming the Lookup Table

If drawing a field that is all one color, the frame buffer pointers for that color point to the same LUT location. This enables the programmer to change the color of a field by changing the one color value in the LUT.

The LUT can be loaded during vertical blank intervals. This reduces CPU overhead that results from responding to vertical blank interrupts. Video memory cycles are needed when loading the LUT, therefore the drawing in progress will slow down while loading the palette.

Automatic LUT Load (ALL) Function

The automatic LUT load (ALL) function is an optimized color palette loading mechanism supporting the Bt458 RAMDAC and compatible units. The ALL function automatically loads frame-buffer resident color palettes into the RAMDAC. The Palette_0 Pointer (PLT0), Palette_1 Pointer (PLT1), and BLINK registers control the look-up table auto-load functions of the controller.

The ALL function supports two modes of operation: Palette Loading and Blinking.

- The Palette Loading operation begins at the leading edge of the vertical blank period following an ALL command and continues without further host intervention. The color graphics controller fetches the palette entries from the specified frame-buffer address and generates the appropriate RAMDAC control and palette addresses. The entire palette transfer completes within this blanking interval, thereby preventing undesirable on-screen artifacts.
- The Blinking operation functions like the Palette Loading mode, but instead of loading the palette once, the loading takes place continuously, alternating between two different color palettes, thus producing the blinking effect. The display period of each palette is an independent programmable function of the monitor frame rate, and thus allows complete control of palette blink-rate and duty-cycle.

Palette Storage Restrictions

You must store palettes on 256-pixel column boundaries and on 8-pixel row boundaries. Accordingly, frame buffer palette X-coordinate bits 7-0 and Y-coordinate bits 2-0 must equal 0. Every row of the palette table will have 256 columns loaded with successive color values. The number of rows specified is dependent on the palette size of the RAMDAC in use. For example, the Bt458 requires three rows, or 768 entries. There are no restrictions (except for memory size) on the number of palette tables you can use.

ALL Registers

Three registers govern the operation of the ALL unit: Palette_0 Pointer (PLT0), Palette_1 Pointer (PLT1), and BLINK. At reset time, the contents of these registers are unknown and unchanged.

The two color palettes pointed to by PLT0 and PLT1 are alternately loaded into the RAMDAC lookup table according to the values in the BLINK register. The timing value in either P1FA or P0FA is loaded into the BDC bits. The BDC value counts down at vertical frequency. When BDC reaches 0, the other pallette is loaded into the RAMDAC, and its timing value is loaded into BDC. This cycle continues until the BLINK_ENABLE bit in CSR2 is set to 1. Note that only the primary LUT is loaded, not the overlay RAM or the control registers.

Blinking

The subsystem has a blinking function that automatically switches between two palettes at a specified frame rate to blink colors. Blinking is controlled via the BLINK register. Turning the blink function off and putting the palette change operation under program control loads the palette at the next vertical interval following program command.

Double-Buffering

double buffering, for example, using 4 bits/pixel for each buffer, and setting up more palettes for more complex multiple buffering, such as four separate screens of 2 bits/pixel each. Since the palettes are active for all pixels on the screen, double buffering affects programs running in all open windows.

PLT0 Palette_0 Pointer

Address FFF8 90A4 Read/Write

The Palette_0 Pointer (PLT0) register contains the starting X and Y coordinates of palette 0 in the frame buffer. The coordinates force the palette table origins to the upper-left corner of the palette. The register contents are as follows:

31 28	27 24	23 16
Reserved	P0_X	Reserved

15 13	12 3	2 0
Reserved	P0_Y	Reserved

Bit	Mnemonic	Function
31–28	Reserved	Must be zeroes when written to, and undefined when read from.
27–24	P0_X	X field starting coordinate for palette 0 Corresponds to screen X–coordinate bits 11–8 (bits 7–0 are always 0).
23–13	Reserved	Must be zeroes when written to, and undefined when read from.
12–3	P0_Y	Y field starting coordinate for palette 0 Corresponds to screen Y–coordinate bits 12–3 (bits 2–0 are always 0).
2–0	Reserved	Must be zeroes when written to, and undefined when read from.

PLT1 Palette_1 Pointer

Address FFF8 90A8 Read/Write

The Palette_1 Pointer (PLT1) register contains the starting X and Y coordinates of palette 1. The coordinates force the palette table origins to the upper-left corner of the palette. The register contents are as follows:

31 28	27 24	23 16
Reserved	P1_X	Reserved

15 13	12 3	2 0
Reserved	P1_Y	Reserved

Bit	Mnemonic	Function
31-28	Reserved	Must be zeroes when written to, and undefined when read from.
27-24	P1_X	X field starting coordinate for palette 1 Corresponds to screen X-coordinate bits 11-8 (bits 7-0 are always 0).
23-13	Reserved	Must be zeroes when written to, and undefined when read from.
12-3	P1_Y	Y field starting coordinate for palette 1 Corresponds to screen Y-coordinate bits 12-3 (bits 2-0 are always 0).
2-0	Reserved	Must be zeroes when written to, and undefined when read from.

BLINK Blink

Address FFF8 90AC Read/Write

The Blink (BLINK) register contains four fields to control palette loading (PL, RPAB, FBR, and PP) and three fields to control blinking (BE, BDC, and .

The four control fields contain bits to enabe either Palette Load or Blink mode operation, select palette_0 or palette_1 for Palette Loading (PP), specify the number of transfer rows, and set palette address bits 9 and 8 when using 1024-entry RAMDACs (RPAB).

BDC, PP, The three 8-bit Blink mode fields specify the active periods for palette 0 and palette 1, and indicate the time remaining for the current palette. These fields provide the means for controlling the blink-rate and duty-cycle during palette blinking.

Bit	Mnemonic	Function
31, 30	RPAB	Palette Address Bits RAMDAC palette address bits 9 and 8 (required for DACs with more than 256 color entries).
29-27	FBR	Frame Buffer Rows Number of frame buffer rows to transfer.
26, 25	BE, PL	Blink Enable, Palette Load 01 Enables Palette Load mode, disables Blink mode. 10 Enables Blink mode, disables Palette Load mode.
24	PP	Palette pointer 0 Selects palette 0. 1 Selects palette 1.
23-16	BDC	Blink Duration Count
15-8	P1FA	Palette 1 Frames Active Number of active palette 1 frames.
7-0	P0FA	Palette 0 Frames Active Number of active palette 0 frames.

Notes

When an ALL command executes, it continues until all palette entries are transferred to the RAMDAC.

The palette is loaded during vertical blank (VBLANK) periods. Palette Load transfers begin at the leading-edge of the first VBLANK following the setting of the Palette Load (PL) bit.

The color graphics controller clears PL immediately after completing a palette load.

The ALL registers are not pipelined; their contents should not be modified while the LUT is active. See the following "Command Procedure" section. The exception to this is that during Blink mode, the palette register of the inactive palette may be loaded with a new address if the blink duration count (BDC) bits in the Blink register are equal to or greater than 2 (this gives the host a minimum of 15 milliseconds to load the new palette address before the LUT accesses it). This procedure enables blinking with more than two colors.

All Blink mode transfers start with palette 0, and begin execution at the leading edge of the first VBLANK following the setting of the Blink Enable (BE) bit.

Command Procedure

The command procedures for the Palette Load mode and the Blink mode differ only in the contents of the Blink register. For either mode, perform the following:

1. Check the Palette Load bit for 0 (LUT unit is inactive). Ensure the Blink Enable bit is 0.
2. Load the color palette entries into the frame buffer. If the desired LUT color map is not already resident, load the map into the frame buffer. Note the previous restriction on modifying the ALL registers.
3. Load the appropriate palette pointer register (Palette_0 Pointer or Palette_1 Pointer) with the address of the color map specified in Step 2.
4. Enable either the Palette Load or Blink operation by writing the following command word into the Blink register:

Accessing the RAMDAC

This section discusses how to program the RAMDAC. For details on RAMDAC operation, consult the *Brooktree® Product Databook.* The RAMDAC registers and memory are not initialized during powerup.

The RAMDAC registers and color/overlay palette RAM locations are accessed through the following addresses (these registers exist within the RAMDAC):

Address	Name	Function
FFF8 90C0	DAC0	RAMDAC Address register.
FFF8 90C4	DAC1	RAMDAC Color Palette RAM.
FFF8 90C8	DAC2	RAMDAC Control register.
FFF8 90CC	DAC3	RAMDAC Overlay Palette RAM.

The Address register (DAC0) allows you to access memory and registers within the RAMDAC. Write the address of the desired register into DAC0, then write or read the data.

To write data to the color palette RAM, write the RAM address to be modified into DAC0. Then perform three sequential write operations to DAC1, one each for red, green, and blue data for that RAM address. Within the RAMDAC, the third write operation causes all three color values to be written into the palette RAM simultaneously — thus you must perform three Mbus writes to change a palette RAM location. At the same time, an address pointer in the RAMDAC is incremented to the next palette RAM location in preparation for another write operation. To write to consecutive palette RAM locations it is only necessary to write DAC0 with the address of the first location to be modified, and then to make the appropriate number of triple writes to DAC1.

Reading data from the color palette RAM is similar to writing data. Place the address of the first location to be read into DAC0; then read DAC1 three times to provide red, green, and blue data (in that order) for that location. The RAMDAC internal address pointer increments each time reading from consecutive locations.

Reading and writing the overlay RAMs is similar. After placing the address into DAC0, read or write DAC3 for the data transfer. Again, the RAMDAC internal pointer automatically increments after each access.

The RAMDAC contains four control registers that you also access indirectly by placing their address in DAC0, and then reading or writing the data through DAC2. The RAMDAC internal address pointer does not increment when you access the control registers. The four control registers and their functions are as follows:

Address	Name	Function
04	Read Mask	Enables or disables a bit plane from addressing the color palette RAM. Bit 0 corresponds to plane 0, bit 1 to plane 1, and so on. 0 Disables bit plane addressing. 1 Enables bit plane addressing.
05	Blink Mask	Enables or disables a bit plane from blinking at the rate specified in the Command register. Bit 0 corresponds to plane 0, bit 1 to plane 1, and so on. 0 Disables blinking. 1 Enables blinking.
06	Command	Specifies a RAMDAC command. The bits and their values are as follows: **Bit Function** 7 Pixel multiplexing 0 4:1 (color graphics controller value) 1 5:1 6 Source of color data 0 Use overlay color 0 if the overlay input is 0. 1 Use color palette RAM. 5,4 Controls blink rate cycle time and duty cycle (in units of vertical syncs) 00 16 on, 48 off 01 16 on, 16 off 10 32 on, 32 off 11 64 on, 64 off 3 Enables/disables blinking of overlay plane 1. 0 Disables. 1 Enables. 2 Enables/disables blinking of overlay plane 0. 0 Disables. 1 Enables. 1 Enables/disables display of overlay plane 1. 0 Disables. 1 Enables. 0 Enables/disables display of overlay plane 0. 0 Disables. 1 Enables.
07	Test	Allows testing of the data path through the RAMDAC. See the *BrooktreeR Product Databook* for details.

RAMDAC Access Restrictions

You should not access the RAMDAC when the auto LUT load feature is in use (contention may result) or a drawing operation is in progress (test the DIP bit in CSR0 for 0).

Before you access the RAMDAC, execute the following line of C code to check for DIP = 0:

```
while (reg ! csr0 & 0x02) {}
```

Initializing the Registers

After a hardware reset, the power-up code initializes the display. The program enables the drivers for the frame buffer control signals, initializes the RAMDAC so that only planes 2–0 are valid and may be written to, initializes the look-up tables so that each one corresponds to one of the three primary colors (red, green, blue), enables the sync pulses to pass to the monitor, clears the screen, and sets the FORCBLNK bit in CRT2 high.

Sample C Program

The following C program initializes the color graphics subsystem.

```
/*******************************************************************/
/* Constants defining the screen physical characteristics */
#define XMIN 0
#define YMIN 0
#define XMAX 1279
#define YMAX 1023
/*******************************************************************/
/* Global variables */
int PC_WID_COPY;
int LUOP;
int ATTRIBUTES;
/*******************************************************************/
/* Data structure for color graphics controller internal registers */
struct color_controller_reg
{
    int csr0;                         /* adr 00 */
    int stop;                         /* adr 04 */
    int csr1;                         /* adr 08 */
    int cmd;                          /* adr 0C */
    int mask;                         /* adr 10 */
    int back;                         /* adr 14 */
    int lpat;                         /* adr 18 */
    int pc_wid;                       /* adr 1C */
    int crt0;                         /* adr 20 */
    int crt1;                         /* adr 24 */
    int crt2;                         /* adr 28 */
    int reserved1;                    /* adr 2C */
    int state0;                       /* adr 30 */
    int state1;                       /* adr 34 */
    int reserved2;                    /* adr 38 */
    int reserved3;                    /* adr 3C */
    int parm0;                        /* adr 40 */
    int parm1;                        /* adr 44 */
    int parm2;                        /* adr 48 */
    int parm3;                        /* adr 4C */
    int parm4;                        /* adr 50 */
    int parm5;                        /* adr 54 */
    int parm6;                        /* adr 58 */
    int parm7;                        /* adr 5C */
    int parm8;                        /* adr 60 */
    int parm9;                        /* adr 64 */
    int parm10;                  /* adr 68 */
    int parm11;                  /* adr 6C */
    int parm12;                  /* adr 70 */
    int parm13;                  /* adr 74 */
    int parm14;                  /* adr 78 */
    int parm15;                  /* adr 7C */
    int filler1[8];              /* adr 80-9C */
    int dataport;                /* adr A0 */
    int plt0;                    /* adr A4 */
    int plt1;                    /* adr A8 */
    int blnk;                    /* adr AC */
    int filler2[4];              /* adr B0-BC */
    int dac0;                    /* adr C0 */
    int dac1;                    /* adr C4 */
    int dac2;                    /* adr C8 */
    int dac3;                    /* adr CC */
    int filler3[4];              /* adr D0-DC */
    int zga[8];                  /* adr E0-FC */
};
/******************************************************************/
/* This macro defines the pointer to the internal register structure. */
#define GRAPH() register struct color_controller_reg *reg \
                              = (struct color_controller_reg *) 0xfff89000
/******************************************************************/
/* Logical Drawing Operations */
#define ZERO_DEST                0
#define AND_SRC_DEST             1
#define AND_SRC_NOTDEST          2
#define SRC_DEST                 3
#define AND_NOTSRC_DEST          4
#define EXOR_ONE_DEST            5
#define EXOR_SRC_DEST            6
#define OR_SRC_DEST              7
#define AND_NOTSRC_NOTDEST       8
#define EXOR_NOTSRC_DEST         9
#define EXOR_ZERO_DEST           10
#define OR_SRC_NOTDEST           11
#define NOTSRC_DEST              12
#define OR_NOTSRC_DEST           13
#define OR_NOTSRC_NOTDEST        14
#define ONE_DEST                 15
/******************************************************************/
/* Macros that use the color graphics controller directly. */
#define COL_MODE(lu_op) LUOP = lu_op
#define SET_ORG(x,y) while (reg->csr0 & 2) {} \
    reg->parm0 = x<<16 | y
#define SET_FORE(color) while (reg->csr0 & 2) {} \
    reg->parm3 = color
#define SET_BACK(color) while (reg->csr0 & 2) {} \
    reg->back = color
#define SET_MASK(pmask) while (reg->csr0 & 2) {} \
    reg->mask = pmask
```

```
#define SET_PCWID(pcwid) while (reg->csr0 & 2) {} \
    reg->pc_wid = pcwid;
#define SET_LPAT(pattern) while (reg->csr0 & 2) {} \
    reg->lpat = pattern
#define SET_PATRPT(pat_rpt) while (reg->csr0 & 2) {} \
    pcwid_copy = reg->pc_wid;\
    reg->pc_wid = (PCWID_COPY & 0xfff8ffff) | ((pat_rpt & 0x7)<<16)
#define SET_WIDKEY(key) while (reg->crs0 & 2) {} \
    pcwid_copy = reg->pc_wid;\
    reg->pc_wid = (PCWID_COPY & 0xffff00ff) | ((key & 0x3)<<8)
#define SET_WIDMASK(mask) while (reg->crs0 & 2) {} \
    pcwid_copy = reg->pc_wid;\
    reg->pc_wid = (PCWID_COPY & 0xffffff00) | (mask & 0x3)
#define SET_CLIPRECT(x,y,w,h) while (reg->csr0 & 2) {} \
    reg->parm6 = x<<16 | y;\
    reg->parm7 = (x+w)<<16 | ((y+h) & 0x1fff)
                                      /* Set clipping rectangle */
#define FILL() while(reg->csr0 & 1) {} \
    reg->cmd = 0x00010f01;\
    reg->parm4 = 0x05000400;\
    reg->parm1 = 0x00000000;\
    reg->parm2 = 0x00000000                   /* BITBLT X-set */
#define BLANK() while(reg->csr0 & 1) {} \
    reg->cmd = 0x00010001;\
    reg->parm4 = 0x05000400;\
    reg->parm1 = 0x00000000;\
    reg->parm2 = 0x00000000                   /* BITBLT X-clear */
#define LINE(x1,y1,x2,y2) while(reg->csr0 & 1) {} \
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = x1<<16 | (y1 & 0x1fff);\
    reg->parm2 = x2<<16 | (y2 & 0x1fff)       /* Line */
/* SHADED LINE (must SHADE_ENABLE before calling) */
#define SLINE(x1,y1,x2,y2,fore,finc) while(reg->csr0 & 1) {} \
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm4 = fore;\
    reg->parm5 = finc;\
    reg->parm1 = x1<<16 | (y1 & 0x1fff);\
    reg->parm2 = x2<<16 | (y2 & 0x1fff);\
    while(reg->csr0 & 2)
#define BITBLTUL(xd,yd,yc,ws,hs)\
    while(reg->csr) & 1) {}
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x01;\
    reg->parm4 = ws <<16 | hs;\
    reg->parm1 = xc <<16 | (yc & 0x1fff);\
    reg->parm2 = xd <<16 | (yd & 0x1fff);     /* BITBLT */
#define RXFER(x,y,w,h)\
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x06;\
    reg->parm4 = w <<16 | h;\
    reg->parm2 = x <<16 | (y & 0x1fff);       /* Read transfer */
#define WXFER(x,y,w,h)\
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x07;\
    reg->parm4 = w <<16 | h;\
    reg->parm2 = x <<16 | (y & 0x1fff);       /* Write transfer */
```

```
#define CHK_BLANK() while(reg->csr1 & 0x80000) {} \
    while(!(reg->csr1 & 0x80000)) {}
#define POINT(a,b) while(reg->csr0 & 1) {} \
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = a<<16 | (b & 0x1fff);\
    reg->parm2 = a<<16 | (b & 0x1fff)              /* Single-pixel line */
#define RECT(x,y,w,h) while(reg->csr0 & 1) {} \
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = x<<16 | (y & 0x1fff);\
    reg->parm2 = (x+w)<<16 | (y & 0x1fff);\
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = (x+w)<<16 | (y & 0x1fff);\
    reg->parm2 = (x+w)<<16 | ((y+h) & 0x1fff);\
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = (x+w)<<16 | ((y+h) & 0x1fff);\
    reg->parm2 = x<<16 | ((y+h) & 0x1fff);\
    reg->cmd = ATTRIBUTES | LUOP<<8 | 0x02;\
    reg->parm1 = x<<16 | ((y+h) & 0x1fff);\
    reg->parm2 = x<<16 | (y & 0x1fff)            /* 4 lines = rectangle */
/*****************************************************************/
#define CLIP_ON ATTRIBUTES = ATTRIBUTES & 0xfffeffff
#define CLIP_OFF ATTRIBUTES = ATTRIBUTES | 0x00010000
#define CLIP_INSIDE ATTRIBUTES = ATTRIBUTES & 0xfffdffff
#define CLIP_OUTSIDE ATTRIBUTES = ATTRIBUTES | 0x00020000
#define CLIP_CONTINUE ATTRIBUTES = ATTRIBUTES & 0xfffbffff
#define CLIP_STOP ATTRIBUTES = ATTRIBUTES | 0x00040000
#define WID_DISABLE ATTRIBUTES = ATTRIBUTES & 0xfff7ffff
#define WID_ENABLE ATTRIBUTES = ATTRIBUTES | 0x00080000
#define PATTERN_DISABLE ATTRIBUTES = ATTRIBUTES & 0xffefffff
#define PATTERN_ENABLE ATTRIBUTES = ATTRIBUTES | 0x00100000
#define STIPPLE_DISABLE ATTRIBUTES = ATTRIBUTES & 0xffdfffff
#define STIPPLE_ENABLE ATTRIBUTES = ATTRIBUTES | 0x00200000
#define TRANSPARENT_DISABLE ATTRIBUTES = ATTRIBUTES & 0xffbfffff
#define TRANSPARENT_ENABLE ATTRIBUTES = ATTRIBUTES | 0x00400000
#define SOLID_DISABLE ATTRIBUTES = ATTRIBUTES & 0xff7fffff
#define SOLID_ENABLE ATTRIBUTES = ATTRIBUTES | 0x00800000
#define SET_SOURCEPLANE(plane) ATTRIBUTES =
    (ATTRIBUTES & 0xf8ffffff) | (plane & 0x7)<<24
#define ZBUF_DISABLE ATTRIBUTES = ATTRIBUTES & 0xf7ffffff
#define ZBUF_ENABLE ATTRIBUTES = ATTRIBUTES | 0x08000000
#define PAT_CONTINUE ATTRIBUTES = ATTRIBUTES & 0xefffffff
#define PAT_RESET ATTRIBUTES = ATTRIBUTES | 0x10000000
#define DO_LAST ATTRIBUTES = ATTRIBUTES & 0xdfffffff
#define NO_LAST ATTRIBUTES = ATTRIBUTES | 0x20000000
#define SHADE_DISABLE ATTRIBUTES = ATTRIBUTES & 0xbfffffff
#define SHADE_ENABLE ATTRIBUTES = ATTRIBUTES | 0x40000000
```

```
/******************************************************************/
/* Initialize graphics subsystem. */
void InitGraph()
{    int i;
     GRAPH();
     /* Initialize palette with gray scale. */
     reg->dac0 = 0x00;
     for (i=0; i < 256; i++)
        {
               reg->dac1 = i;
               reg->dac1 = i;
               reg->dac1 = i;
        }
     SET_MASK(0xff);
     SET_BACK(0x00);
     SET_FORE(0xff);
     SET_LPAT(0xffffffff);
     SET_PCWID(0x0);
     SET_ORG(0,0);
     SET_CLIPRECT(0,0,XMAX,YMAX);
     /* Clear screen and overlay planes. */
     BLANK();
     SET_ORG(0,4096);
     BLANK();
     SET_ORG(0,0);
     ATTRIBUTES = 0;
     CLIP_OFF;
     COL_MODE(SRC_DEST);
     while (reg->csr0 & 2) {};
     /* Initialization done. */
}
```

```
/*********************************************************************/
/* Random line generator — generates nn thousand random lines in gray scale. */
void random(nn)
int nn;
{
register x1,y1,x2,y2,xmax,ymax,seed,loop,loop1,zero,p,pc;
register val1,val2,val3,val4;
int color;
GRAPH();
zero = 0;
xmax = XMAX;
ymax = YMAX;
seed = 3;
val1 = 31421;
val2 = 6927;
val3 = 65535;
val4 = 16;
p = 1;
color = 1;
pc = 0;
loop = nn;
while (loop-- > zero) {
        SET_FORE(color);
        COL_MODE(SRC_DEST);
        loop1 = 1000;
        while (loop1->zero) {
                seed = (seed * val1 + val2) & val3;
                x1 = (seed * xmax) >> val4;
                seed = (seed * val1 + val2) & val3;
                x2 = (seed * xmax) >> val4;
                seed = (seed * val1 + val2) & val3;
                y1 = (seed * ymax) >> val4;
                seed = (seed * val1 + val2) & val3;
                y2 = (seed * ymax) >> val4;
                LINE(x1,y1,x2,y2);
                if (++pc > 40)
                        {
                                pc = 0;
                                if (++color > 255) {color = 0;}
                                SET_FORE(color);
                        }
                }
        p = -p;
        }
}
main()
{
    InitGraph();
    random(3);
}
/*********************************************************************/
```

Programming the Z–Buffer Controller

The optional Z–buffer controller, available on 400 series stations, supports three–dimension applications, providing hidden line removal and hidden surface removal, and Hither and Yon clipping. The Z–buffer is a slave to the color graphics controller, and sits on the Mbus. The Z–buffer registers are 32–bit; some bits can be accessed by the color graphics controller. The Z–buffer controller also features

- A 25–MHz Mbus interface.
- Direct read/write access to the Z–buffer array from the host CPU.
- Selectable 24–plane Z–buffer support using 256K x 4 DRAMS.
- Fixed internal 24–bit resolution.
- Hardware configurable memory organization and resolution supporting all color graphics controller screen resolutions.
- Fast rectangular Clear/Set mode with color graphics controller assistance.
- Programmable Hither and Yon clipping planes.
- Support for color graphics controller Stop/Resume operations (context switching).

Components of the Z–Buffer

As shown in Figure 5–7, the Z–buffer gate array is partitioned into four main functional modules: an Mbus interface, Z–controller, Datapath/ALU, and a memory controller.

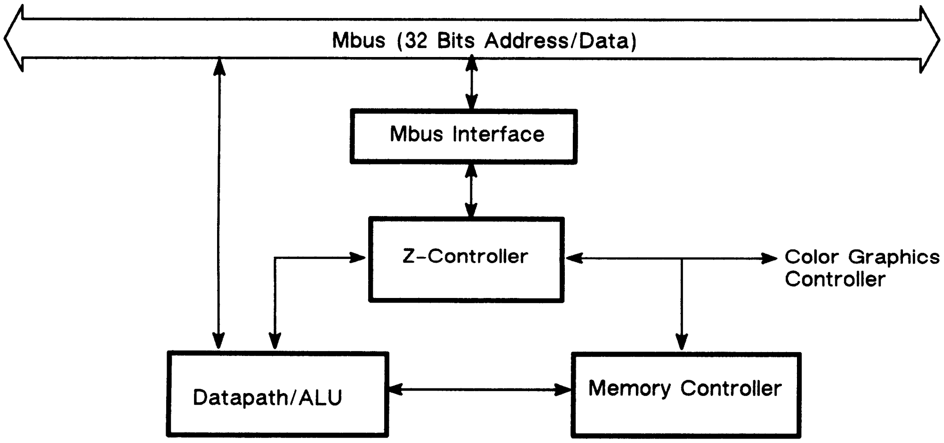


Figure 5–7 Z–Buffer Gate Array Components

Mbus Interface

The Mbus interface provides address decoding and access to the Z-buffer registers and memory. All registers and memory are accessed as 32-bit words. When read, the Z-buffer generates parity bits, but when written to, it ignores parity bits.

The Z-buffer registers are located at FFF8 90E0 - FFF8 91F0. The Z-buffer memory array consists of 32 million words located at 8800 0000 - 8FFF FFFC.

The color graphics controller also decodes the register and array addresses and generates some of the Mbus control signals for the Z-buffer controller.

Z-Controller

The Z-controller monitors and interprets the Z-protocol bits. Depending on the state and sequencing of these bits, the Z-controller directs data steering and ALU opcode selections within the Datapath unit. For example, the Z-controller may select the contents of the DZ/DX register to decrement the current Z-value.

Datapath/ALU

The Datapath/ALU consists of registers, comparators, data multiplexers, data latches, and an adder. This unit provides the various data paths needed for reading and writing the Z-buffer, updating the current Z-value, and for comparing new-Z with clipping planes and the old-Z. Control of the Datapath is shared between the Mbus for register access, the Z-controller for incrementing/decrementing Z-values, and by the memory controller for buffering Z-values between the Z-buffer and the frame buffer.

Memory Controller

The Memory Controller generates control signals for the Z-array and internal signals for steering and latching Z-data within the Datapath unit. The color graphics controller initiates all activities of the memory controller unit. At the beginning of all Z-buffer and frame buffer accesses, the color graphics controller sends a "cycle type." If the type corresponds to a Z-buffer request, the Memory Controller performs the Z-buffer access. These accesses occur in parallel to, and are sychronized with, the color graphics controller frame buffer access. Note that for all Z-buffer accesses, the color graphics controller and the Z-buffer both perform memory cycles.

Configuration Logic

The Configuration Logic controls Z-buffer functions that are dependent on the graphics configuration of the system. The host CPU loads the configuration parameters into the Configuration (CNFG) register during Z-buffer initialization.

Programming the Z-Buffer Registers

The Z-buffer register set consists of the thirteen 32-bit read/write registers as shown in Table 5-5.

NOTE: Write 0s to unused bits. Unused bits are undefined when read.

NOTE: To ensure that the Z-buffer and the color graphics controller's graphics processor use the same number of wait states, you must write to the Z-buffer Configuration (CNFG) Register as the first Z-buffer access. You may write/read the remaining registers in any order.

All Z-values are twos complement numbers that represent Z-space (Hither - Yon) from 80 0000 - 7F FFFF in 24-bit systems, and from 8000 - 7FFF in 16-bit systems. Z-values are treated differently depending on the system: in 24-bit systems, Z-values are 24-bit integers; in 16-bit systems, Z-values consist of a 16-bit integer and an 8-bit fraction.

Context Switching

Context switching occurs when the CPU sets the Stop (STP) bit in the STOP register of the color graphics controller. The color graphics controller suspends its current operation and clears the Drawing In Progress (DIP) bit in CSR0. The CPU may now save the states of the color graphics controller and the Z-buffer. To save the states of the Z-buffer, read and save the Z-buffer registers.

To restore the context of a stopped command, reload the color graphics controller and Z-buffer registers. Then issue a Resume command to the color graphics controller (set the RES bit in the STOP Register).

Table 5-5 Color Graphics Register Set Address Map

Address	Name	Function
FFF8 9008	SR	Shadow (this is the same address as the Control and Status Register 1, CSR1).
FFF8 90E0	ZCMD	Command
FFF8 90E4	ZIR	Interrupt
FFF8 90E8	Z_CNST	Z-Constant
FFF8 90EC	Z_INIT	Initial Z-depth
FFF8 90F0	Z_XINC	DZ/DX
FFF8 90F4	Z_YINC	DZ/DY
FFF8 90F8	HCLP	Hither Clip
FFF8 90FC	YCLP	Yon Clip
FFF8 91E0	ZCNFG	Configuration
FFF8 91E4	ST0	State 0
FFF8 91E8	ST1	State 1
FFF8 91EC	ST2	State 2
FFF8 91F0	ST3	State 3

ZCMD Z-Buffer Command

Address FFF8 90E0 Read/Write

The Command (CMD) register is used to modify the execution of Z-Buffer commands.

31 18	17	16
Unused	RST	See B_SLCT

16 15	14 13	12 11	10 9	8	7 6	5	4 3	2	1	0
B_SLCT	Z_UNF	Z_OVF	YCLOP	YCLEN	HCLOP	HCLEN	CMPOP	CMPEN	CLRMD	INTEN

Bit	Mnemonic	Function
31-18	Unused	Must be written as 0s and read as undefined.
17	RST	Z-Buffer Reset Causes an internal Z-buffer reset; all operations in progress are halted and the contents of all registers are undefined. 1 Causes a Z-buffer reset.
16, 15	B_SLCT	Bank Select Specifies the mapping of the Z-buffer to the frame buffer bank. These bits are only used in systems with 2Kx2K and 2Kx4K frame buffers. In these systems, the 2Kx1K Z-buffer address space will be mapped to the selected 2Kx1K frame buffer bank. Note that these bits are ignored for 2Kx1K and 1Kx512 frame buffers. 00 Z-buffer mapped to frame buffer bank 0. 01 Z-buffer mapped to frame buffer bank 1. 10 Z-buffer mapped to frame buffer bank 2. 11 Z-buffer mapped to frame buffer bank 3.
14, 13	Z_UNF	Z-Underflow Specifies the action to take if the calculation of Z-New results in an arithmetic underflow. 00 Write minimum Z-value and write new pixel value. 01 Write minimum Z-value and retain old pixel value. 10 Retain the old Z-value and write new Pixel value. 11 Retain the old Z-value and old pixel value.
12, 11	Z_OVF	Z-Overflow Specifies the action to take if the calculation of Z-New results in an arithmetic overflow. 00 Write maximum Z-value and write new pixel value. 01 Write maximum Z-value and retain old pixel value. 10 Retain the old Z-value and write new pixel value. 11 Retain the old Z-value and old pixel value.
10, 9	YCLOP	Yon Clip Operation Contains the clipping comparison that determines when a pixel is to be clipped against the Yon plane. 00 Clip if Z-New less than Yon clipping plane Z. 01 Clip if Z-New less than or equal to Yon clipping plane Z. 10 Clip if Z-New greater than Yon clipping plane Z. 11 Clip if Z-New greater than or equal to Yon clipping plane Z.

(continued)

Bit	Mnemonic	Function
8	YCLEN	Yon Clipping Enabled Enables/disables clipping against the Yon (Aft) plane specified in the Yon Clip Register. 0 Disables Yon plane clipping. 1 Enables Yon plane clipping (CMD bits 10 and 9 specify the conditions under which a Z-clip takes place).
7, 6	HCLOP	Hither Clip Operation Contains the clipping comparison that determines when a pixel is to be clipped against the Hither plane. 00 Clip if Z-New less than Hither clipping plane Z. 01 Clip if Z-New less than or equal to Hither clipping plane Z. 10 Clip if Z-New greater than Hither clipping plane Z. 11 Clip if Z-New greater than or equal to Hither clipping plane Z.
5	HCLEN	Hither Clipping Enabled Enables/disables clipping against the Hither(Fore) plane specified in the Hither Clip Register. 0 Disables Hither plane clipping (forces a "win vote" into the Z-Win logic).[1] 1 Enables Hither plane clipping (CMD bits 7 and 6 specify the conditions under which a Z-clip takes place).
4, 3	CMPOP	Compare Operation Specifies the Z-Buffer comparison to determine a Z-Win.! 00 Win if Z-New less than Z-Old. 01 Win if Z-New less than or equal to Z-Old. 10 Win if Z-New greater than Z-Old. 11 Win if Z-New greater than or equal to Z-Old.
2	CMPEN	Compare Enable Enables/disables the Z-compare operation. When Z-Compare is disabled no Z-tournament takes place and a "win vote" is forced into the Z-Win logic. When enabled, CMD bits 4 and 3 specify the conditions for a Z-win.! 0 Disables Z-Compares. 1 Enables Z-Compares.
1	CLRMD	Clear Mode Enables/disables the ability of all Z-Buffer memory write operations to use the value stored in the Z-Constant Register for the Z-Buffer write data. This mode generates a Z-Win output to the system color graphics controller, supporting simultaneous clearing of the Z-buffer and frame buffer areas. If SOLID and CLRMD are set during a BITBLT, the controller will do a "fast clear." 0 Disables Clear Mode. 1 Enables Clear Mode.
0	INTEN	Interrupt Enable Enables/disables the Mbus from interrupting (for any of the four unmasked interrupt sources). 0 Disables interrupts. 1 Enables interrupts.

[1] A new pixel and Z-value will replace the existing pixel and Z-value only when the following condition is true — Z-Compare Win & Hither Clip Win & Yon Clip Win.

(concluded)

ZIR Z–Buffer Interrupt

Address FFF8 90E4 Read/Write

The Interrupt Register (IR) contains interrupt mask and status bits.

31	16
Unused	

15	8	7	6	5	4	3	2	1	0
Unused		YCPINT	HCLINT	UNFINT	OVFINT	YONMSK	HITHMSK	UNFMSK	OVFMSK

Bit	Mnemonic	Function
31–8	Unused	Must be written as 0s and read as undefined.
7	YCPINT	Yon Clip Interrupt Indicates whether or not a clip against the yon plane has occurred. A read of the IR register resets this bit. 0 No clip. 1 Clip has occurred.
6	HCLINT	Hither Clip Interrupt Indicates whether or not a clip against the hither plane has occurred. A read of the IR register resets this bit. 0 No clip. 1 Clip has occurred.
5	UNFINT	Underflow Interrupt Indicates whether or not an underflow has occurred. A read of the IR register resets this bit. 0 No underflow. 1 Underflow has occurred.
4	OVFINT	Overflow Interrupt Indicates whether or not an overflow has occurred. A read of the IR register resets this bit. 0 No overflow. 1 Overflow has occurred.
3	YONMSK	Yon Clip Mask Controls whether or not interrupts are generated for pixels clipped against the yon clip plane. 0 Generate an interrupt (unmasked). 1 Do not generate an interrupt (masked).
2	HITHMSK	Hither Clip Mask Controls whether or not interrupts are generated for pixels clipped against the hither clip plane. 0 Generate an interrupt. 1 Do not generate an interrupt.
1	UNFMSK	Underflow Interrupt Mask Controls whether or not interrupts are generated when an arithmetic underflow occurs. 0 Generate an interrupt. 1 Do not generate an interrupt.
0	OVFMSK	Overflow Interrupt Mask Controls whether or not interrupts are generated when an arithmetic overflow occurs. 0 Generate an interrupt. 1 Do not generate an interrupt.

Z_CNST Z-Buffer Constant

Address FFF8 90E8 Read/Write

The Z_Constant (Z_CNST) register contains the twos complement value to be written to the Z-Buffer during the Clear mode. For Z-buffer clearing, load the largest 24-bit, positive Z-value (7F FFFF). Use any arbitrary value to preset areas of the Z-Buffer.

31 24	23 16
Unused	Z_CNST

15 0
Z_CNST

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	Z_CNST	Z Constant Twos complement value (bit 23 is the sign bit). For 16-plane Z-buffers, bits 23-8 contain the 16-bit Z_CNST and bits 7-0 are ignored.

Z_INIT Z-Depth Initial

Address FFF8 90EC Read/Write

The Z_Initial (Z_INIT) register contains the initial Z-depth value associated with the next line or polygon to be rendered. For both 16-bit and 24-bit , all 24 bits (23-0) must be specified.

31 24	23 16
Unused	Z_INIT

15 0
Z_INIT

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	Z_INIT	Initial Z-Value Z_INIT twos complement value (bit 23 is the sign bit).

Z_XINC DZ/DX

Address FFF8 90F0 Read/Write

The DZ/DX (Z_XINC) register contains the slope of Z with respect to X. Z_XINC is used for Line (LINE) and Polygon (POLY) operations.

31 24	23 16
Unused	DZ/DX

15 0
DZ/DX

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	DZ/DX	Slope of Z With Respect to X Twos complement Z-value (bit 23 is the sign bit). Specify all 24 bits for both 16-bit and 24-bit Z-buffers.

Z_YINC DZ/DY

Address FFF8 90F4 Read/Write

The DZ/DY (Z_YINC) register contains the slope of Z with respect to Y. Z_YINC is used for Line (LINE) and Polygon (POLY) operations.

31 24	23 16
Unused	DZ/DY

15 0
DZ/DY

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	DZ/DY	Slope of Z With Respect to Y Twos complement Z-value (bit 23 is the sign bit). Note that you must specify all 24 bits for both 16-bit and 24-bit Z-buffer implementations.

HCLP Hither Clip

Address FFF8 90F8 Read/Write

The Hither Clip (HCLP) register contains the Z-coordinate of the hither clipping plane. HCLP is used only when Hither Clipping is enabled (CMD bit 5 is 1).

31 24	23 16
Unused	HCLP

15 0
HCLP

Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	HCLP	Hither Clip Twos complement Z–value (bit 23 is the sign bit).

YCLP — Yon Clip

Address FFF8 90FC — Read/Write

The Yon Clip (YCLP) register contains the Z–coordinate of the yon clipping plane. YCLP is used only when Yon Clipping is enabled (CMD bit 8 is 1).

31 — 24	23 — 16
Unused	YCLP

15 — 0
YCLP

Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	YCLP	Yon Clip Twos complement Z–value (bit 23 is the sign bit).

ZCNFG Z-Buffer Configuration

Address FFF8 91E0 Read/Write

The Configuration (CNFG) register contains the graphics hardware configuration, including the number of Z-planes, number of Mbus wait states, etc. Write to CNFG initially to coordinate the Z-buffer and the color graphics controller.

31 16
Unused

15 10	9	8 6	5 3	2	1	0
Unused	EMO	ZCBR	FBSIZE	Unused	ZWST	ZBD

Bit	Mnemonic	Function
31–10	Unused	Must be written as 0s and read as undefined.
9	EMO	Enable Memory Output Enables/disables memory outputs. A reset enables memory outputs. 0 Disables memory outputs. 1 Enables memory outputs.
8–6	ZCBR	Z-Buffer CBR Cycles Must match the number of CBR (Cas-Before-Ras) refresh cycles programmed in the color graphics controller (CBR bits of CSR1). 000 No CBR cycles performed. 001 1 CBR cycle per scan line. 010 2 CBR cycles per scan line. 011 3 CBR cycles per scan line. 100 4 CBR cycles per scan line. 101 5 CBR cycles per scan line. 110 6 CBR cycles per scan line. 111 7 CBR cycles per scan line.
5–3	FBSIZE	Frame Buffer Size Reflects the frame buffer size of the color graphics controller. The Z-buffer uses this information for Z-Buffer address mapping. 000 2K x 1K 64K x 4 VRAMS. 001 1K x 512 64K x 4 VRAMS. 010 2K x 1K 256K x 4 VRAMS. 011 2K x 2K 256K x 4 VRAMS. 100 4K x 2K 256K x 4 VRAMS.
2	Unused	Must be written as 0s and read as undefined.
1	ZWST	Z-Buffer Wait States Specifies the minimum number of Mbus wait states inserted in the data phase. Note that a powerup sets 0 wait states for the Z-buffer. Since this value may not match the number of wait states for the color graphics controller, the first Mbus access to the Z-buffer must write the appropriate number of wait states. 0 0 wait states inserted. 1 1 wait state inserted.
0	ZBD	Z-Buffer Depth Specifies the depth of the Z-Buffer memory (read-only). Loaded when reset to indicate the number of Z-planes. 0 16 bit Z-buffer depth. 1 24 bit Z-buffer depth.

ST0 State 0

Address FFF8 91E4 Read/Write

The State0 (ST0) register contains the current Z-depth value of an interrupted draw command. The contents of this register are used during Stop and Resume operations.

31 24	23 16
Unused	IZDEPTH

15 0
IZDEPTH

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	IZDEPTH	Interrupted Zbuffer Depth. Twos complement Z-value (bit 23 is the sign bit).

ST1 State 1

Address FFF8 91E8 Read/Write

The State1 (ST1) register contains the current DZ/DX value of an interrupted draw command. The contents of this register are used during Stop and Resume operations.

31 24	23 16
Unused	IDZ/DX

15 0
IDZ/DX

Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	IDZ/DX	Interrupted Slope of Z with Respect to X Twos complement Z-value (bit 23 is the sign bit).

ST2 State 2

Address FFF8 91EC Read/Write

The State2 (ST2) register contains the current DZ/DY value of an interrupted draw command. The contents of this register are used during Stop and Resume operations.

31 24	23 16
Unused	IDZ/DY

15 0
IDZ/DY

Bit	Mnemonic	Function
31-24	Unused	Must be written as 0s and read as undefined.
23-0	IDZ/DY	Interrupted Slpoe of Z with Respect to Y Twos complement Z-value (bit 23 is the sign bit).

ST3 State 3

Address FFF8 91F0 Read/Write

The State3 (ST3) register contains a copy of the currently active draw command. The contents of this register are used during Stop and Resume operations.

31 16
Unused

15 4	3 0
Unused	IADRAW

Bit	Mnemonic	Function
31–4	Unused	Must be written as 0s and read as undefined.
3–0	IADRAW	Interrupted Draw Command Command Opcode (see the color graphics controller Command register description for the opcodes).

SHADOW Shadow

Address FFF8 9008 Read/Write

The Shadow Register (SR) contains a copy of the Opcode bits (3–0) from the Command register. SR is write–only; only the color graphics contrroller can read it.

31 16
Unused

15 4	3 0
Unused	IDRAW

Bit	Mnemonic	Function
31–4	Unused	Must be written as 0s and read as undefined.
3–0	IDRAW	Drawing Instruction Command Opcode (see the color graphics Command register description for the opcodes).

End of Chapter

Chapter 6
Programming the Keyboard Interface and Speaker

This chapter describes the following topics:

- The keyboard interface.
- How to program the keyboard interface.
- How to program the speaker.

Overview

The keyboard interface supports both AT–compatible and Japanese AX–compatible keyboards. The keyboard interface has a universal asynchronous receiver/transmitter (UART) which passes data from the keyboard interface to the CPU. The operating system must handle communication with the keyboard and interpret the keyboard scan codes.

The final sections of this chapter explain how to program the speaker.

Components of the Keyboard Interface

As shown in Figure 6–1, the keyboard interface consists of a UART, address decode and control logic, and clock and timing logic. The rest of this section describes briefly these components and the keyboard speaker function.

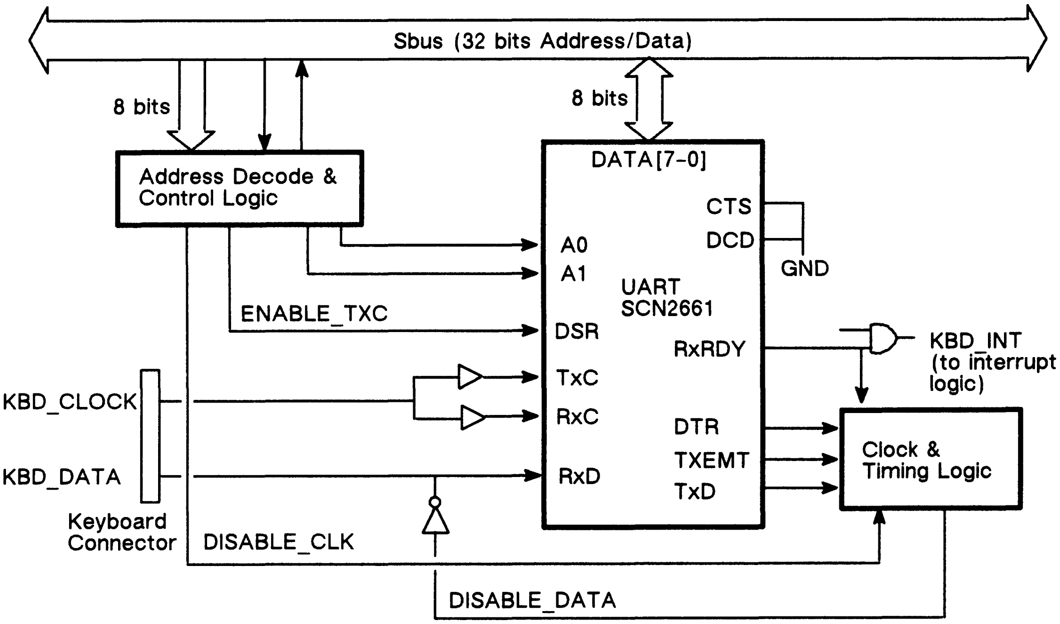


Figure 6–1 Keyboard Interface Components

UART

The UART is a Signetics SCN2661 enhanced programmable communications interface controller. It converts the incoming serial stream from the keyboard into parallel data for the CPU and optionally checks parity. It also converts parallel data to a serial stream for transmission to the keyboard, and generates the start, stop, and parity bits.

Clock and Timing Logic

The clock and timing logic contains the Disable Clock (DISABLE_CLOCK) register and the Enable Transmit Clock (ENABLE_TXC) register. The Disable Clock register allows a program to force the keyboard clock line low, which is necessary to prevent the keyboard from transmitting data. The Enable Transmit Clock register allows the program to control whether or not the keyboard clock sources the UART's TxC input.

Keyboard Speaker

The workstation provides a speaker on the system board since PC/AT compatible keyboards do not have an internal speaker. The timer output of DUART1 provides the waveform for this speaker.

Keyboard Connector

The keyboard cable connects to the keyboard port through an 8-pin DIN connector (J8). identifies the keyboard signals.

Table 6-1 Keyboard Signals

Pin	Signal	Direction
1	CLOCK	To keyboard
2	DATA	From keyboard
3	Unused	-----
4	Ground	-----
5	+5 V	To keyboard
6-8	Unused	-----

Programming the Keyboard Interface

The keyboard routine controls the interface via UART registers, as well as the Disable Clock register and the Enable Transmit Clock register. The keyboard routine must configure the UART parameters through the UART registers.

The keyboard routine must maintain a protocol to communicate correctly with the keyboard. It accomplishes this by manipulating the keyboard clock and data lines at the appropriate times. Keyboard software must convert scan codes into meaningful characters or instructions.

The speaker is controlled through DUART1.

CAUTION: *Software must not issue sequential reads and/or writes to the registers because of the setup timing from read/write inactive to read/write active that Signetics specifies for the UART. Put two nonoperative instructions between register access reads or writes.*

Clock and Data Lines

The CPU and keyboard communicate over the keyboard clock and data lines, which either the CPU (via the keyboard interface) or the keyboard can force low (inactive). When no communication is occurring, both the clock and data lines are high (active).

The CPU sends data to the keyboard by first causing the keyboard interface to force the data line low (request to send), and then releasing the clock, allowing it to go high so the keyboard can drive the clock. The keyboard drives the clock when clocking data in or out. The CPU causes the keyboard interface to force the clock line low to inhibit keyboard transmission. Table 6–2 defines the states of the clock and data lines.

Table 6–2 Keyboard Clock and Data Lines

Clock	Data	Port Control Function
H	H	Allows keyboard to transmit data.
L	H	Inhibits keyboard from transmitting data.
L	L	Indicates that the CPU is requesting to transmit a command to the keyboard.
H	L	Allows keyboard to receive data.

Data Format

Each byte of data received by the UART consists of 11 bits as defined in Table 6–3. The UART strips the start bit and stop bit, checks the parity bit, shifts the serial data bits into a shift register, and notifies the CPU that it has a byte of data. The keyboard routine must then read the byte of data and interpret the data.

Table 6–3 Keyboard Data Format

Bit	Function
1	Start bit (always 0)
2	Data bit 0 (least significant bit)
3	Data bit 1
4	Data bit 2
5	Data bit 3
6	Data bit 4
7	Data bit 5
8	Data bit 6
9	Data bit 7 (most significant bit)
10	Parity bit (odd)
11	Stop bit (always 1)

Registers

Table 6–4 shows the registers and their addresses. The Disable Clock register and the Enable Transmit Clock register are located in DUART1, all other registers are located in the UART.

Table 6–4 Keyboard Register Addresses

Address	Register	Type
FFF8 2800	Receive Holding Register (RHR) Transmit Holding Register THR)	Read Write
FFF8 2804	Status (STS)	Read
FFF8 2808	Mode Register 1 (MD1) Mode Register 2 (MD2)	Read/Write Read/Write
FFF8 280C	Command (CMD)	Read/Write
FFF8 2810	Disable Clock (DSC)	Write
FFF8 2820	Enable Transmit Clock (ENABLE_TXC)	Write

For more information on the UART registers, see the Signetics *Microprocessor Data Manual.*

RxH Receive Holding

Address FFF8 2800 Read Only

The Receive Holding (RxH) register temporarily stores a byte of data transmitted by the keyboard. The section "Keyboard Scan Codes" defines the scan code data transmitted by the keyboard. Table 6–5 lists the response codes.

The Receive Holding register is defined as follows:

7	0
Data	

Bit	Name	Function
7–0	Data	Holds data sent from the keyboard for the CPU.

Table 6–5 Keyboard Responses

Code (hex)	Response
FA	Acknowledge (ACK) Sent as the standard response to transmissions from the host. The exceptions are the Echo and Resend commands, which have specialized responses (see EE and FE codes).
EE	Response Sent in reply to the Echo command from the host.
F0	Break code prefix Sent when a key is released while the keyboard is using either scan code set 2 or scan code set 3. This is the first byte of a 2–byte sequence; the second byte is the scan code of the key released.
00	Overrun Indicates that the keyboard's 16–byte buffer has overflowed, and at least one keystroke is lost.
FE	Resend Sent when the keyboard detects a parity error or illegal command from the host. The host should respond by retransmitting its most recent transmission.
AA	Basic assurance test (BAT) completion Indicates that the keyboard has successfully completed its self–test routines. Another response code (e.g. FC16) after self–test indicates a failure has been detected.
AB, 83	Keyboard ID Sent in response to the Read Keyboard ID command.
02, 03	Key Code mode Sent in response to the Read Key Code Mode command. This value corresponds to the scan set number (0216 = set 2, 0316 = set 3).

TxH — Transmit Holding

Address FFF8 2800 — Write Only

The Transmit Holding (TxH) register temporarily stores commands to be transmitted to the keyboard. Table 6-6 lists the command codes.

The Transmit Holding register bits are defined as follows:

7   0
Data

Bit	Name	Function
7–0	Data	Holds data sent from the CPU to the keyboard.

Table 6-6 Commands

Code (hex)	Command
EE	Echo Request the keyboard to transmit an EE. This command is for diagnostic use only (the keyboard continues scanning if it was previously enabled).
EF, F1	NOP Request the keyboard to transmit an FE to the host as acknowledgement of reception. The keyboard takes no other action.
FF	Reset Halts the keyboard (cease scanning the key array), acknowledge this command by transmitting an FA, and watch the clock and data lines. If both lines remain high for 500 μs, the keyboard performs a self-test. If either line goes low during this period, the Reset operation is aborted and the keyboard continues as before this command. The self-test (600–900 ms in duration) includes a checksum test of ROM and a test of RAM. The keyboard LEDs turn on for 200 ms, and then off. Upon successful completion of the self-test, the keyboard transmits AA. If the self-test fails, the keyboard transmits the appropriate code. The keyboard then sets the typematic repeat delay and rate to the default values, clears the output buffer, and begins to scan the key array. During the self-test, the keyboard allows Clock and Data to float high.
FE	Resend Requests the keyboard to resend the most recent transmission. This command is not valid if the host has sent a transmission more recently than the keyboard, or if the host has enabled the interface after the most recent keyboard transmission. If the keyboard's most recent transmission was a Resend command, the keyboard will retransmit the byte it transmitted before the Resend command. You use this command when the host detects a parity error in a transmission from the keyboard.

(continued)

Table 6-6 Commands

<table>
<tr><th>Code (hex)</th><th>Command</th></tr>
<tr><td>F3</td><td>Set typematic repeat delay/rate
Set the typematic delay time and the rate of repeat. These parameters are coded into a byte that follows the F3 command. The "delay" indicates how long a key may stay down before the typematic repeat feature is activated. The "rate" indicates how often the keycode is repeated after the delay period elapses. When the keyboard receives the F3 command, it stops scanning the key array, transmits an FA to the host, and waits for the second byte. If the most significant bit of the second byte is 1, the F3 command is aborted, and the keyboard treats the second byte as a new command. If the most significant bit of the second byte is 0, then the low-order bits are interpreted as
<table>
<tr><th>Bit</th><th>Meaning</th></tr>
<tr><td>7</td><td>Always 0</td></tr>
<tr><td>6, 5</td><td>Delay
The keyboard adds 1 to this number and divides the sum by 4 to obtain the delay period in seconds. Thus, the delay period may vary from 250 ms (002) to 1 s (112). The default delay is 500 ms (012).</td></tr>
<tr><td>4, 3</td><td>Rate
Exponential component (B) of the rate calculation.</td></tr>
<tr><td>2-0</td><td>Rate
Offset (A) in the linear component of the rate calculation. The rate is the reciprocal of the period, which is calculated according to the formula:
Period = 0.00417 * (8 + A) * (2**B) s/character
The rate can vary from 2 Hz (111112) to 30 Hz (000002). The default is 10.9 Hz (010112).</td></tr>
</table>
After receiving a valid parameter byte, the keyboard sends an FA to the host. If the keyboard was not halted when it received the F3, the keyboard resumes scanning the key array. All typematic specifications may vary +/- 20%.</td></tr>
<tr><td>ED</td><td>Set/Reset mode indicators
Specify new states (on or off) for the three LEDs on the keyboard. The first byte of this 2-byte command causes the keyboard to cease scanning the key array and transmit an FA to the host. If the second byte is a valid command (a value of ED or higher), the ED command is aborted, and the keyboard executes the new command without changing the states of the LEDs. If the keyboard was not halted before receiving the ED command, it continues scanning the key array.
If the second byte is not a valid command, the 3 least significant bits determine the new states of the LEDs. A high (1) bit means the corresponding LED should be on; a low (0) bit indicates the LED should be off.
<table>
<tr><th>Bit</th><th>LED</th></tr>
<tr><td>7-3</td><td>Reserved</td></tr>
<tr><td>2</td><td>Caps lock</td></tr>
<tr><td>1</td><td>Num lock</td></tr>
<tr><td>0</td><td>Scroll lock</td></tr>
</table>
The keyboard sends an FA to the host to acknowledge receipt of the second byte, sets the LEDs to their new states, and resumes scanning the key array (if the keyboard was not halted before receiving the ED command). The default state of all three LEDs is off.</td></tr>
<tr><td>F2</td><td>Read keyboard ID
Request the keyboard to acknowledge this command with an FA followed by the two keyboard ID bytes, AB and 83, and then resume scanning if previously enabled.</td></tr>
<tr><td>F0</td><td>Set/Read Key Code mode
Request the keyboard to acknowledge this command with an FA; then wait for an option code from the host. The keyboard responds to the option code with an FA. If the option code is 00, the keyboard sends out a byte indicating the existing key code mode status (02 = scan set 2, 03 = scan set 3). If the option mode is 02 or 03, the keyboard sets the code mode status to scan set 2 or 3, respectively.</td></tr>
</table>

(continued)

Table 6–6 Commands

<table>
<tr><th>Code
(hex)</th><th>Command</th></tr>
<tr><td>F6</td><td>Set default
Cause the keyboard to assume its default state. The output buffer is cleared, and typematic rates are reset to the default. The keyboard acknowledges this command by transmitting an FA, and resumes scanning the key array (if it was not halted prior to receiving the command).</td></tr>
<tr><td>F5</td><td>Default disable
Cause the keyboard to perform all functions of the Set Default command (F6), except that the keyboard does not resume scanning the key array after executing the command. This is the halted state of the keyboard.</td></tr>
<tr><td>F4</td><td>Enable
Cause the keyboard to transmit an FA, clear its output buffer, and start to scan the key array.</td></tr>
<tr><td>FC, FD</td><td>Typematic key reset 1, 2
Cause the keyboard to respond with an FA, stop scanning, and wait for the option code, which specifies the key to be set. The keyboard responds to the option code with an FA. This command cancels the typematic function; then sets the key type of the specified key to either Make/Break (command FC) or Make (command FD). This command applies only to scan set 3.</td></tr>
<tr><td>FB</td><td>Typematic key set
Cause the keyboard to respond with an FA, stop scanning, and wait for the option code specifying the key to be set. The keyboard responds to the option code with an FA. If the scan set is 3, the command sets the key type of the specified key to typematic. The keyboard remains halted after the command.</td></tr>
<tr><td>F7–FA</td><td>All key typematic control
Cause the keyboard to respond with FA, enable or disable the typematic function, and allow or inhibit the transmission of a break code when a key is released. The operation is dependent on the actual code as follows:
<table>
<tr><th>Typematic</th><th>Break Code</th><th></th></tr>
<tr><th>Command</th><th>Function</th><th>Sending</th></tr>
<tr><td>F7</td><td>enable</td><td>disable</td></tr>
<tr><td>F8</td><td>disable</td><td>enable</td></tr>
<tr><td>F9</td><td>disable</td><td>disable</td></tr>
<tr><td>FA</td><td>enable</td><td>enable</td></tr>
</table>
The keyboard continues scanning if previously enabled. This command applies only to scan set 3.</td></tr>
</table>

(concluded)

Keyboard Scan Codes

The keyboard transmits a scan code when a key is depressed. If a key is held down so that it satisfies the requirements of the typematic function, the scan code is repeated until the key is released. The scan codes are linked to the position of the keys, not to the character on the key. The operating system must decode the scan codes and assign characters to the codes. Figure 6–2 illustrates the keys on a keyboard, and Table 6–7 identifies the scan codes for the keys.

The system defaults to scan set 2, but can be switched to scan set 3. See the description of the Receive Holding (RxH) register. With scan code set 3, each key sends only 1 scan code, and no keys are affected by the state of any other keys.

Table 6–7 lists the scan code values. The values in scan code set 3 remain constant regardless of the state of the shift keys. Many codes in scan set 2 and scan set 3 are identical. The state of the shift keys (Ctrl, Alt, and Shift) and the Num Lock key affect the values in scan code set 2.

Make is transmitted when the key is pressed. Each key has a unique 8–bit make scan code.

Break is transmitted when a key is released. The break code is 2 bytes long (the first byte always contains the break code prefix, F0; the second byte is the same as the make scan code for that key). Each key sends a break code when the key is released.

Typematic indicates that if the key is held down, the keyboard will repetively transmit the scan code until the key is released. The typematic scan code for a key is the same as the key's make code.

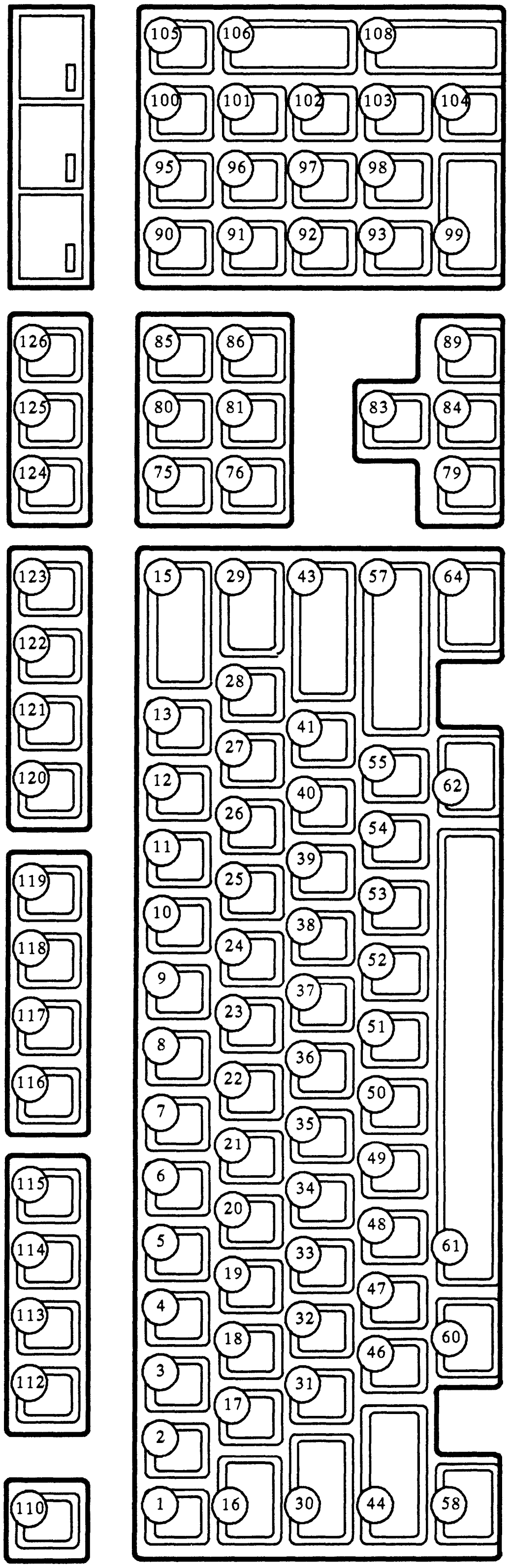


Figure 6–2 Position of Keys on Keyboard

Table 6-7 Scan Code Sets 2 and 3

	Scan Code Set 2		Scan Code Set 3		
	Codes		Codes		Default
Key #	Make	Break	Make	Break	Key State
1	0E	F0 0E	0E	F0 0E	Typematic
2	16	F0 16	16	F0 16	Typematic
3	1E	F0 1E	1E	F0 1E	Typematic
4	26	F0 26	26	F0 26	Typematic
5	25	F0 25	25	F0 25	Typematic
6	2E	F0 2E	2E	F0 2E	Typematic
7	36	F0 36	36	F0 36	Typematic
8	3D	F0 3D	3D	F0 3D	Typematic
9	3E	F0 3E	3E	F0 3E	Typematic
10	46	F0 46	46	F0 46	Typematic
11	45	F0 45	45	F0 45	Typematic
12	4E	F0 4E	4E	F0 4E	Typematic
13	55	F0 55	55	F0 55	Typematic
15	66	F0 66	66	F0 66	Typematic
16	0D	F0 0D	0D	F0 0D	Typematic
17	15	F0 15	15	F0 15	Typematic
18	1D	F0 1D	1D	F0 1D	Typematic
19	24	F0 24	24	F0 24	Typematic
20	2D	F0 2D	2D	F0 2D	Typematic
21	2C	F0 2C	2C	F0 2C	Typematic
22	35	F0 35	35	F0 35	Typematic
23	3C	F0 3C	3C	F0 3C	Typematic
24	43	F0 43	43	F0 43	Typematic
25	44	F0 44	44	F0 44	Typematic
26	4D	F0 4D	4D	F0 4D	Typematic
27	54	F0 54	54	F0 54	Typematic
28	5B	F0 5B	5B	F0 5B	Typematic
29 [1]	5D	F0 5D	5C	F0 5C	Typematic
30	58	F0 58	14	F0 14	Make/Break
31	1C	F0 1C	1C	F0 1C	Typematic
32	1B	F0 1B	1B	F0 1B	Typematic
33	23	F0 23	23	F0 23	Typematic
34	2B	F0 2B	2B	F0 2B	Typematic
35	34	F0 34	34	F0 34	Typematic
36	33	F0 33	33	F0 33	Typematic
37	3B	F0 3B	3B	F0 3B	Typematic
38	42	F0 42	42	F0 42	Typematic
39	4B	F0 4B	4B	F0 4B	Typematic
40	4C	F0 4C	4C	F0 4C	Typematic
41	52	F0 52	52	F0 52	Typematic
42 [2]	5D	F0 5D	53	F0 53	Typematic
43	5A	F0 5A	5A	F0 5A	Typematic
44	12	F0 12	12	F0 12	Make/Break
45 [2]	61	F0 61	13	F0 13	Typematic
46	1A	F0 1A	1A	F0 1A	Typematic
47	22	F0 22	22	F0 22	Typematic
48	21	F0 21	21	F0 21	Typematic
49	2A	F0 2A	2A	F0 2A	Typematic

[1] 101-key keyboard only.

[2] 102-key keyboard only.

(continued)

Table 6-7 Scan Code Sets 2 and 3

	Scan Code Set 2		Scan Code Set 3		
	Codes		Codes		Default
Key #	Make	Break	Make	Break	Key State
50	32	F0 32	32	F0 32	Typematic
51	31	F0 31	31	F0 31	Typematic
52	3A	F0 3A	3A	F0 3A	Typematic
53	41	F0 41	41	F0 41	Typematic
54	49	F0 49	49	F0 49	Typematic
55	4A	F0 4A	4A	F0 4A	Typematic
57	59	F0 59	59	F0 59	Make/Break
58	14	F0 14	11	F0 14	Make/Break
60	11	F0 11	19	F0 19	Make/Break
61	29	F0 29	29	F0 29	Typematic
62	E0 11	E0 F0 11	39	F0 39	Make
64	E0 14	E0 F0 14	58	F0 58	Make
75	E0 70	E0 F0 70	67	F0 67	Make
(Shift + Num Lock)	E0 70	E0 F0 70	—	—	—
(Num Lock on)	E0 12 E0 70	E0 F0 70 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 70	E0 F0 70 E0 12			
76	E0 71	E0 F0 71	64	F0 64	Typematic
(Shift + Num Lock)	E0 71	E0 F0 71	—	—	—
(Num Lock on)	E0 12 E0 71	E0 F0 71 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 71	E0 F0 71 E0 12			
79	E0 6B	E0 F0 6B	61	F0 61	Typematic
(Shift + Num Lock)	E0 6B	E0 F0 6B	—	—	—
(Num Lock on)	E0 12 E0 6B	E0 F0 6B E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 6B	E0 F0 6B E0 12			
80	E0 6C	E0 F0 6C	6E	F0 6E	Make
(Shift + Num Lock)	E0 6C	E0 F0 6C	—	—	—
(Num Lock on)	E0 12 E0 6C	E0 F0 6C E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 6C	E0 F0 6C E0 12			
81	E0 69	E0 F0 69	65	F0 65	Make
(Shift + Num Lock)	E0 69	E0 F0 69	—	—	—
(Num Lock on)	E0 12 E0 69	E0 F0 69 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 69	E0 F0 69 E0 12			
83	E0 75	E0 F0 75	63	F0 63	Typematic
(Shift + Num Lock)	E0 75	E0 F0 75	—	—	—
(Num Lock on)	E0 12 E0 75	E0 F0 75 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 75	E0 F0 75 E0 12			
84	E0 72	E0 F0 72	60	F0 60	Typematic
(Shift + Num Lock)	E0 72	E0 F0 72	—	—	—
(Num Lock on)	E0 12 E0 72	E0 F0 72 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 72	E0 F0 72 E0 12			
85	E0 7D	E0 F0 7D	6F	F0 6F	Make
(Shift + Num Lock)	E0 7D	E0 F0 7D	—	—	—
(Num Lock on)	E0 12 E0 7D	E0 F0 7D E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 7D	E0 F0 7D E0 12			

[3] If the left Shift key is held down, the F0 12 (make) and E0 12 (break) is sent with the other scan codes. If the right Shift key is held down, F0 59 (make) and F0 59 (break) is sent. If both Shift keys are held down, both sets of codes are sent with the other scan code.

(continued)

Table 6-7 Scan Code Sets 2 and 3

	Scan Code Set 2		Scan Code Set 3		
	Codes		Codes		Default
Key #	Make	Break	Make	Break	Key State
86	E0 7A	E0 F0 7A	6D	F0 6D	Make
(Shift + Num Lock)	E0 7A	E0 F0 7A	—	—	—
(Num Lock on)	E0 12 E0 7A	E0 F0 7A E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 7A	E0 F0 7A E0 12			
89	E0 74	E0 F0 74	6A	F0 6A	Typematic
(Shift + Num Lock)	E0 74	E0 F0 74	—	—	—
(Num Lock on)	E0 12 E0 74	E0 F0 74 E0 F0 12	—	—	—
(Shift) [3]	E0 F0 12 E0 74	E0 F0 74 E0 12			
90	77	F0 77	76	F0 76	Make
91	6C	F0 6C	6C	F0 6C	Make
92	6B	F0 6B	6B	F0 6B	Make
93	69	F0 69	69	F0 69	Make
95	E0 4A	E0 F0 4A	77	F0 77	Make
96	75	F0 75	75	F0 75	Make
95 (Shift) [3]	E0 F0 12 4A	E0 12 F0 4A	—	—	—
97	73	F0 73	73	F0 73	Make
98	72	F0 72	72	F0 72	Make
99	70	F0 70	70	F0 70	Make
100	7C	F0 7C	7E	F0 7E	Make
101	7D	F0 7D	7D	F0 7D	Make
102	74	F0 74	74	F0 74	Make
103	7A	F0 7A	7A	F0 7A	Make
104	71	F0 71	71	F0 71	Make
105	7B	F0 7B	84	F0 84	Make
106	79	F0 79	7C	F0 7C	Typematic
108	E0 5A	E0 F0 5A	79	F0 79	Make
110	76	F0 76	08	F0 08	Make
112	05	F0 05	07	F0 07	Make
113	06	F0 06	0F	F0 0F	Make
114	04	F0 04	17	F0 17	Make
115	0C	F0 0C	1F	F0 1F	Make
116	03	F0 03	27	F0 27	Make
117	08	F0 08	2F	F0 2F	Make
118	83	F0 83	37	F0 37	Make
119	0A	F0 0A	3F	F0 3F	Make
120	01	F0 01	47	F0 47	Make
121	09	F0 09	4F	F0 4F	Make
122	78	F0 78	56	F0 56	Make
123	07	F0 07	5E	F0 5E	Make
124	E0 12 E0 7C	E0 F0 7C E0 F0 12	57	F0 57	Make
(Ctrl, Shift)	E0 7C	E0 F0 7C	—	—	—
(Alt)	84	F0 84	—	—	—
125	7E	F0 7E	5F	F0 5F	Make
126 [4]	E1 14 77 E1 F0 14 F0 77	—	62	F0 62	Make
(Ctrl)	E0 7E E0 F0 7E	—	—	—	—

[3] If the left Shift key is held down, the F0 12 (make) and E0 12 (break) is sent with the other scan codes. If the right Shift key is held down, F0 59 (make) and F0 59 (break) is sent. If both Shift keys are held down, both sets of codes are sent with the other scan code.

[4] This key is not typematic. All associated scan codes occur on the make of the key.

(concluded)

STS Status

Address FFF8 2804 Read Only

The Status (STS) register contains the receiver, transmitter, and control signal status for the keyboard interface.

Its bits are defined as follows:

7	6	5	4	3	2	1	0
ETC	1	FE	OE	PE	STC	RxRDY	TxRDY

Bit	Name	Function
7	ETC	Enable Transmit Clock register (ENABLE_TXC) Status 0 Indicates that the TxC input is enabled. 1 Indicates that the TxC input is disabled.
6	1	Always 1.
5	FE	Framing Error 1 Indicates that a framing error occurred.
4	OE	Overrun Error 1 Indicates that an overrun error occurred.
3	PE	Parity Error 1 Indicates that a parity error occurred.
2	STC	Status Change 1 Indicates that the Enable Transmit Clock register changed status or the Transmit Shift register completed a transmission and a new character was loaded into the Transmit Holding register. You can determine if a transmitter-empty condition exists by reading this register twice while the Enable Transmit Clock remains unchanged.
1	RxRDY	Receive Holding register Status 0 Indicates that the Receive Holding register is empty. 1 Indicates that the Receive Holding register contains data.
0	TxRDY	Transmit Holding register Status 0 Indicates that the Transmit Holding register contains data. 1 Indicates that the Transmit Holding register is empty.

MD1, MD2 Mode

Address FFF8 2808 Read/Write

The Mode (MD1, MD2) registers program the UART mode of operation. The first time a program reads or writes the register's address, it accesses MD1. The second time, it accesses MD2. Any subsequent reads or writes recycle the UART internal sequencer between these two registers.

MD1 is defined as follows:

7 6	5	4	3 2	1 0
STB	PT	PC	CL	BRM

Bit	Name	Function
7-6	STB	Number of Stop Bits 01 1 stop bit.
5	PT	Parity 0 Odd.
4	PC	Parity Control 1 Parity enabled.
3-2	CL	Character Length 11 8 data bits.
1-0	BRM	Baud Rate Multiplier 01 Asynchronous 1x.

MD2 is defined as follows:

7 0
CKS

Bit	Name	Function
7-0	CKS	Clock Source 0000 0000 The UART derives its receive and transmit clocks from the RXC and TXC inputs. The TXC and RXC inputs are from the KBD_CLOCK signal, generated by the keyboard.

CMD Command

Address FFF8 280C Read/Write

The Command (CMD) register contains commands for the keyboard interface. Reading this register returns the current command status, and writing to it defines the command status.

Its bits are defined as follows:

7 6	5	4	3	2	1	0
OPM	RTS	RE	FB	RxEN	IKT	TxEN

Bit	Name	Function
7–6	OPM	Operation Mode 00 Normal
5	RTS	Reserved and unused (must be written to with a 0).
4	RE	Reset Error 0 Normal 1 Reset the Frame Error, Overrun Error, and Parity Error flags in the Status register.
3	FB	Force Break 0 Normal operation.
2	RxEN	Enable Receiver 0 Disable the receiver. 1 Enable the receiver.
1	IKT	Inhibit Keyboard Transmission 0 Allow the keyboard to transmit data. 1 Inhibit the keyboard from transmitting data.
0	TxEN	Enable Transmitter 0 Disable the transmitter. 1 Enable the transmitter.

DSC — Disable Clock

Address FFF8 2810 — Write Only

The Disable Clock (DSC) register allows software to force the keyboard clock line low to prevent the keyboard from sending data. While sending data to the keyboard, software must prevent the keyboard from sending data. The keyboard clock is automatically disabled when the system is Reset or when both the transmitter is empty and Command register bit 1 (Inhibit Keyboard Transmission) is set to 1.

The bit in the Disable Clock register is defined as follows:

7 — 1	0
Unused	DKC

Bit	Name	Function
0	DKC	Disable Keyboard Clock 0 Inhibit the keyboard from transmitting data by forcing the keyboard clock low. 1 Allow the keyboard to transmit data by releasing the keyboard clock line.

ETXC Enable Transmit Clock

Address FFF8 2820 Write Only

The Enable Transmit Clock (ETXC) register allows software to control whether or not the keyboard clock sources the UART Transmit Clock (TxC) input. While software is configuring the keyboard interface to receive data, it should disable the transmit clock using this register. The transmit clock is automatically disabled when the system is Reset or when both the transmitter is empty and Command register bit 1 (Inhibit Keyboard Transmission) is set to 1.

The bit in the Enable Transmit Clock register is defined as follows:

7 1	0
Unused	EKD

Bit	Name	Function
0	EKD	Enable Keyboard Data 0 Enables the keyboard to receive data by providing the keyboard clock as the source of the UART's Transmit Clock (TxC) input. 1 Inhibits the keyboard from transmitting data by preventing the keyboard clock from sourcing the UART's Transmit Clock (TxC) input.

Interrupts

When the keyboard transmits a character, the UART receives the character into the Receive Holding Register (RHR) and control logic forces the keyboard clock Low to prevent the keyboard from transmitting more characters. When the UART receives a scan code, it asserts an interrupt request to the CPU. The UART asserts an interrupt for each scan code it receives. Also, when the UART receives the scan code, keyboard interface logic forces the keyboard clock low to inhibit the keyboard from transmitting more scan codes.

Receiving Data from the Keyboard

In a typical environment, the keyboard service routine should configure the keyboard interface to receive data from the keyboard except when the program sends commands to the keyboard. When the keyboard is ready to send data, it first checks the clock and data lines for a keyboard-inhibit or request-to-send condition (see Figure 6-3). While inhibited, the keyboard stores keystroke data in its buffer. If the CPU asserts a request-to-send, the keyboard stores keystroke data in the keyboard buffer and prepares to receive data from the CPU.

If the keyboard detects an allow-keyboard-transmit condition, it clocks out the 11-bit data stream. Data is valid at the trailing edge of the clock pulse. During transmission, the keyboard checks the clock line every 60 ms. If the clock line is forced low by the CPU while the keyboard is sending data, line contention occurs and the keyboard stops sending data.

Once keyboard service routine configures the keyboard interface for proper data format and clock source, it must configure the interface to receive a character as follows:

1. Inhibit the keyboard from sending characters by writing a 0 to bit 0 of the DSC register. This forces the keyboard clock low.
2. Disable data transmission by writing a 1 to bit 0 of the the ETXC register.
3. Enable the receiver by writing 24 to the CMD register.
4. Enable the keyboard clock by writing a 1 to bit 0 of the DSC register.
5. When the RHR is loaded, inhibit the keyboard from sending more characters by writing a 0 to bit 0 of the DSC register. This forces the clock low.
6. Determine whether the scan code requires transmission to the keyboard. If it requires transmission, enter the transmit routine described in the next section, "Transmitting Data to the Keyboard." If it does not require transmission, re-enable the keyboard by writing a 1 to the DSC register.

Before reading the character from the receive buffer, the keyboard service routine must inhibit the keyboard by writing a 0 to bit 0 of the DSC register. Then, the keyboard service routine can read the character, interpret the scan code, and determine whether or not it needs to respond with a command (such as lighting an LED). Then the keyboard service routine should re-enable the keyboard by writing a 1 to bit 0 of the DSC register.

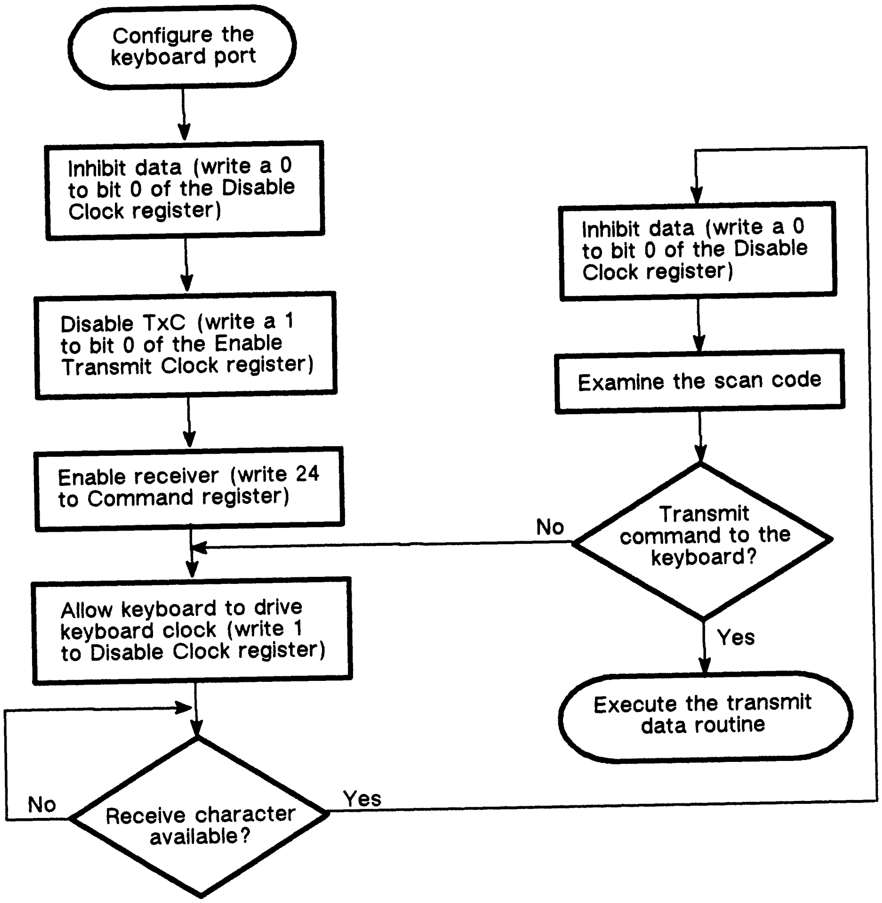

Configure the keyboard port
Inhibit data (write a 0 to bit 0 of the Disable Clock register)
Disable TxC (write a 1 to bit 0 of the Enable Transmit Clock register)
Enable receiver (write 24 to Command register)
Allow keyboard to drive keyboard clock (write 1 to Disable Clock register)
No
Receive character available?
Yes
Inhibit data (write a 0 to bit 0 of the Disable Clock register)
Examine the scan code
No
Transmit command to the keyboard?
Yes
Execute the transmit data routine

Figure 6-3 Receiving Data from the Keyboard

Transmitting Data to the Keyboard

To transmit data to the keyboard, the keyboard service routine must use the following format (see also Figure 6–4):

1. Disable data transmission (TxC) by writing a 1 to the Enable Transmit Clock register.
2. Drive keyboard clock low to prevent further keyboard transmission by writing a 0 to the Disable Clock register.
3. Determine if a receive character is available by reading the Status register and checking bit 1. If bit 1 is a 1, read the available character by reading the Receive Holding register.
4. Disable the receiver and transmitter by writing a 0 to Command register bits 2 and 0.
5. Write the data to be transmitted to the keyboard to the Transmit Holding register.
6. Enable the transmitter by writing a 1 to Command register bit 0.
7. Enable TxC by writing a 0 to the Enable Transmit Clock register and and waiting a minimum of 0.5 μs. (This is the first falling edge of TxC.)
8. Disable TxC by writing a 1 to the Enable Transmit Clock register and waiting a minimum of 0.5 μs.
9. Enable TxC by writing a 0 to the Enable Transmit Clock register and waiting a minimum of 0.5 ms. (This is the second falling edge of TxC.)
10. Disable TxC by writing a 1 to the Enable Transmit Clock register and waiting a minimum of 0.5 μs.
11. Enable TxC by writing a 0 to the Enable Transmit Clock register and waiting a minimum of 60 μs. (This is the third falling edge of TxC. At this point the UART will transmit a start bit on TxD, bringing the keyboard data line low to indicate a Request To Send.)
12. Allow the keyboard to drive the keyboard clock and clock in the data by writing a 1 to the Disable Clock register.
13. Check that the data transmission is complete by polling Status register bit 7 to determine when it to changes from a 0 to a 1.
14. When the transmission of a character is complete, reconfigure the keyboard interface to receive a character (most likely an acknowledge scan code after sending a command) as described in the previous section, "Receiving Data from the Keyboard."

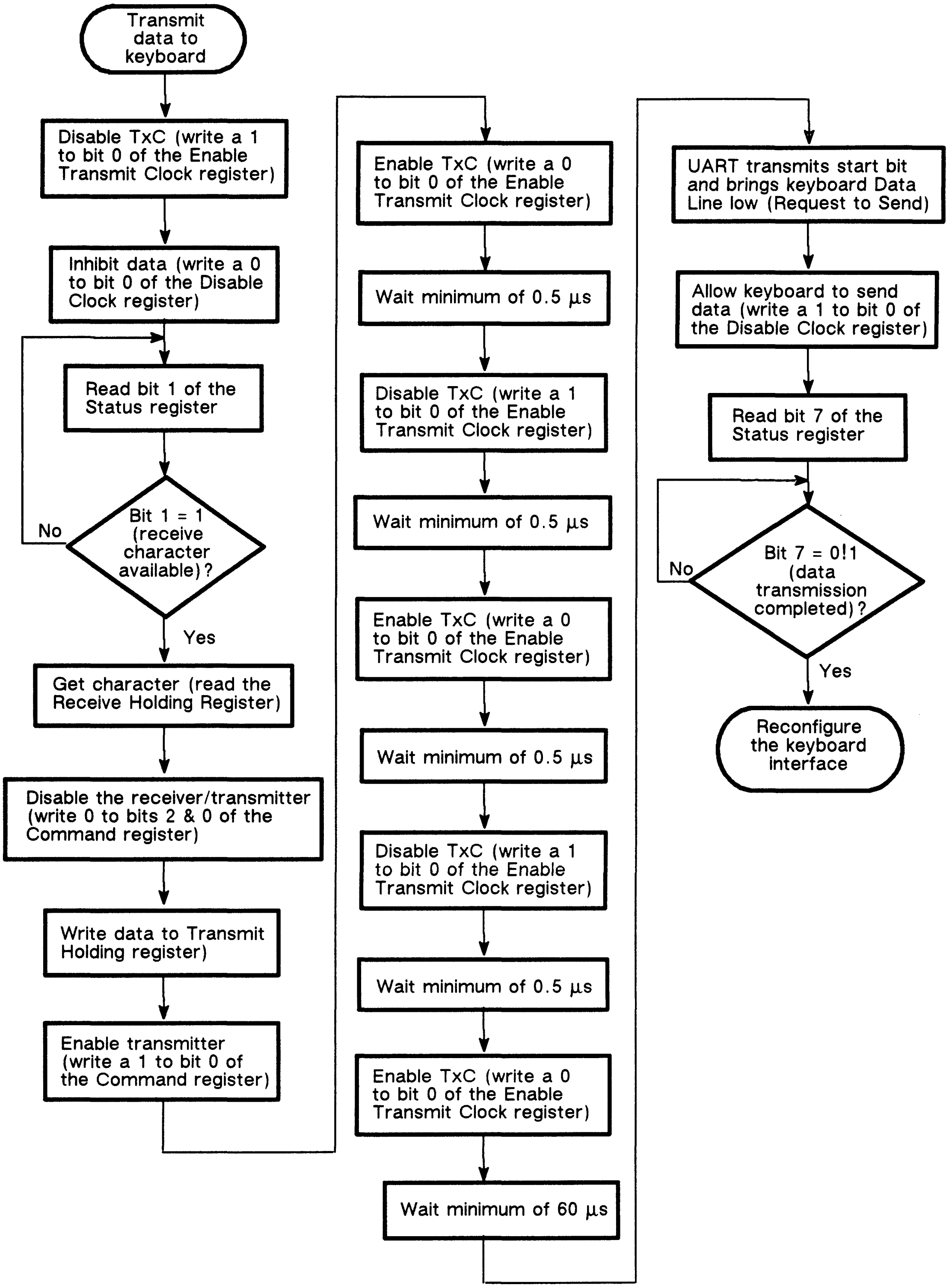

Transmit data to keyboard
Disable TxC (write a 1 to bit 0 of the Enable Transmit Clock register)
Inhibit data (write a 0 to bit 0 of the Disable Clock register)
Read bit 1 of the Status register
Bit 1 = 1 (receive character available)?
No
Yes
Get character (read the Receive Holding Register)
Disable the receiver/transmitter (write 0 to bits 2 & 0 of the Command register)
Write data to Transmit Holding register)
Enable transmitter (write a 1 to bit 0 of the Command register)
Enable TxC (write a 0 to bit 0 of the Enable Transmit Clock register)
Wait minimum of 0.5 µs
Disable TxC (write a 1 to bit 0 of the Enable Transmit Clock register)
Wait minimum of 0.5 µs
Enable TxC (write a 0 to bit 0 of the Enable Transmit Clock register)
Wait minimum of 0.5 µs
Disable TxC (write a 1 to bit 0 of the Enable Transmit Clock register)
Wait minimum of 0.5 µs
Enable TxC (write a 0 to bit 0 of the Enable Transmit Clock register)
Wait minimum of 60 µs
UART transmits start bit and brings keyboard Data Line low (Request to Send)
Allow keyboard to send data (write a 1 to bit 0 of the Disable Clock register)
Read bit 7 of the Status register
Bit 7 = 0!1 (data transmission completed)?
No
Yes
Reconfigure the keyboard interface

Figure 6–4 Transmitting Data to the Keyboard

Programming the Speaker

The speaker output, J18 of the system board, is programmed through DUART1, where OP3 is the timer and OP4 is the speaker enable.

Table 6–8 identifies the registers used to program the speaker port.

Table 6–8 Speaker Register Addresses

Address	Register	Type
FFF8 2010	Auxiliary Control (ACR)	Write
FFF8 2018	Counter/Timer Upper Register (CTUR)	Read/Write
FFF8 201C	Counter/Timer Lower Register (CTLR)	Read/Write
FFF8 2034	Output Port Configuration Register(OPCR)	Write
FFF8 2038	Set Output Port Bits Commmand (SOPBC)	Write
FFF8 203C	Reset Output Port Bits Command (ROPBC)	Write

For further information on these registers, see the Signetics *Microprocessor Data Manual.*

For proper keyboard speaker operation, program the registers as follows:

1. Write 0 to the Output Port Configuration Register (OPCR) on the DUART (base address FFF8 2000).
2. Write 06 to the OPCR to enable the timer.
3. Write 10 to the Reset Output Port Bits Command register (ROPBC) to enable the speaker.
4. Write 60 to the Auxilary Control Register (ACR) to select the clock source.
5. Load the Counter/Timer Upper (CTUR) and Lower (CTLR) registers with the desired sound frequency.
6. Start the sound by reading the Start Counter register (read SOPBC).
7. Wait as long as you want the sound to last.
8. Disable the speaker by writing 10 to SOPBC.
9. Stop the sound by reading the Stop Counter register (read ROPBC).
10. Write 0 to OPCR to clear it.

The following assembly language example shows how to program the speaker.

```
def DUART, 0xfff82000          ;Define the DUART base address.
                               ;Define the registers with offsets.
def UACR, 0x10                 ;Auxiliary Control Register at offset 10.
def CTUP, 0x18                 ;Counter/Timer Upper Register.
def CTLR, 0x1c                 ;Counter/Timer Lower Register.
def OPCR, 0x34                 ;Output Port Configuration Register
                               ;    (write only).
def ROPB, 0x3c                 ;Reset Output Port Bits Command register
                               ;    (write only).
def STRT, 0x38                 ;Start counter (read the Set Output Port
                               ;    Bits Command register).
def STOP, 0x3c                 ;Stop counter (read the Reset Output Port
                               ;    Bits Command register).
                               ;Define the constants.
def SPKR_EN, 0x10              ;Reset value to set signal high.
def K_TIMER, 0x6               ;Value to select OP3 mode (in OPCR)
                               ;    as free-running 1x clock.
def CLKSRC,  0x60              ;Value to select external clock (in ACR).
def Duration, 0x1c71           ;Constant for duration of beep.
_beep:  or     r2,r0,0x8       ;Set a pitch constant in r2.
        or     r3,r0,Duration  ;Set beep duration constant in r3.
        or.u   r6,r0,hi16(DUART)
                               ;Get the address of the DUART controller.
        or     r6,r6,lo16(DUART)
        st     r0,r6,OPCR      ;Clear Configuration register.
        bsr    delay1us        ;Kill time.
        or     r5,r0,K_TIMER   ;Enable K_TIMER as free running clock
        st     r5,r6,OPCR      ;    and enable output port n.
        bsr    delay1us
        or     r5,r0,SPKR_EN   ;Resetting individual bits will set high.
        st     r5,r6,ROPB      ;Enable speaker.
        bsr    delay1us
        or     r5,r5,CLKSRC    ;Logically OR ACR bit 7 and X1 select.
        st     r5,r6,UACR      ;Select source X1 clock.
        bsr    delay1us
        or     r5,r0,r2        ;Use constant to generate counter high.
        st     r5,r6,CTUP
        bsr    delay1us
        or     r5,r0,r2        ;Generate counter low for 16 bits.
        st     r5,r6,CTLR
        bsr    delay1us
        ld     r0,r6,STRT      ;Start counter timer.
waittime:
        subu   r3,r3,1         ;Duration of beep.
        bcnd   ne0,r3,waittime ;Done?
        ld     r0,r6,STOP      ;Stop counter timer.
        bsr    delay1us
        st     r0,r6,OPCR      ;Clear OPCR.
        jmp    r1
```

ACR Auxiliary Control

Address FFF8 2010 Write Only

The Auxiliary Control Register (ACR) contains speaker configuration parameters.

The bits are defined as follows:

7	6 4	3 0
*	CCC	*

Bit	Name	Function
7	*	Used by serial ports.
6-4	CTM	Counter/Timer Mode
		110 The counter/timer uses the CLK input.
		111 The counter/timer uses the CLK input divided by 16.
3-0	*	Used by serial ports.

CTUR, CTLR Counter/Timer

CTUR	**Address FFF8 2018**	**Read/Write**
CTLR	**Address FFF8 201C**	**Read/Write**

The Counter/Timer Upper Register (CTUR) and the Counter/Timer Lower Register (CTLR) are 8-bit registers that together form a 16-bit counter/timer.

The CTUR bits are defined as follows:

7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8

Bit	Name	Function
7-0	C15-C8	Counter/Timer Bits 15-8

The CTLR bits are defined as follows:

7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Bit	Name	Function
7-0	C7-C0	Counter/Timer Bits 7-0

OPCR Output Port Configuration

Address FFF8 2034 Write Only

The Output Port Configuration Register (OPCR) contains configuration data for the speaker function.

The bits are defined as follows:

7 5	4	3 2	1 0
*	CSE	CTW	*

Bit	Name	Function
7-5	*	Used by serial and parallel ports.
4	CSE	Configure Speaker Enable 0 Configures port output speaker enable.
3, 2	CTW	Configure Counter/Timer Waveform 01 Configures the DUART output port 3 to reflect the counter/timer waveform.
1, 0	*	Used by serial ports.

SOPBC Set Output Port Bits Command

Address FFF8 2038 Write Only

The Set Output Port Bits Command (SOPBC) register contains speaker function data.

The bit for the speaker function is defined as follows:

7 5	4	3 0
*	DSP	*

Bit	Name	Function
7-5	*	Used by serial and parallel ports.
4	DSP	Disable Speaker 1 Disables the speaker by causing output port 4 to go low. 0 Leaves ouput port 4 unchanged.
3-0	*	Used by serial and parallel ports.

ROPBC Reset Output Port Bits Command

Address FFF8 203C Write Only

The Reset Output Port Bits Command (ROPBC) register contains speaker function information.

The bit for the speaker function is defined as follows:

7 5	4	3 0
*	ESP	*

Bit	Name	Function
7–5	*	Used by serial and parallel ports.
4	ESP	Enable Speaker 1 Enables the speaker by resetting output port 4 to go high. 0 Leaves ouput port 4 unchanged.
3–0	*	Used by serial and parallel ports.

End of Chapter

Chapter 7
Programming the Serial Ports and Parallel Port

This chapter describes the following topics:

- The serial and parallel ports.
- Serial port registers and how to program the serial ports.
- Parallel port registers and how to program the parallel ports.

Overview of the Serial and Parallel Ports

AViiON 300 stations have two asynchronous serial ports: a mouse port that uses RS–232–C, and a serial port that supports a modem and uses either RS–232–C or RS–422. The I/O signals are described in Appendix D, "System Board Connectors." A specially–designed cable is needed to communicate using the RS–422 signals. See the manual *Setting Up and Starting AViiON*t *300 Series Stations* for the part number and information on this cable.

The RS–422 interface is a set of drivers and receivers in parallel with the RS–232–C drivers and receivers. They are all connected to the same DUART pins; the selection of either the RS–232–C protocol or the RS–422 protocol is invisible to the programmer.

AViiON 400 stations have three asynchronous serial ports: a mouse port and two serial ports which support modems. All three serial ports communicate using the RS–232–C protocol.

Both workstations have a parallel port which supports either a Data Products interface or a Centronics interface.

Components of the Serial and Parallel Ports

The 300 series stations have one DUART which controls two serial ports and a parallel port as shown in Figure 7-1. The 400 series stations have two DUARTs (DUART1 and DUART2) which control three serial ports and a parallel port as shown in Figure 7-2.

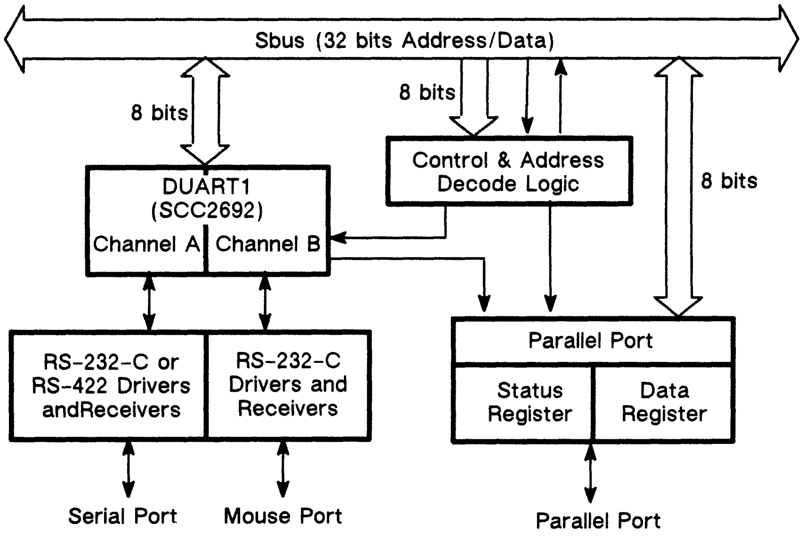


Figure 7-1 300 Series Serial and Parallel Ports

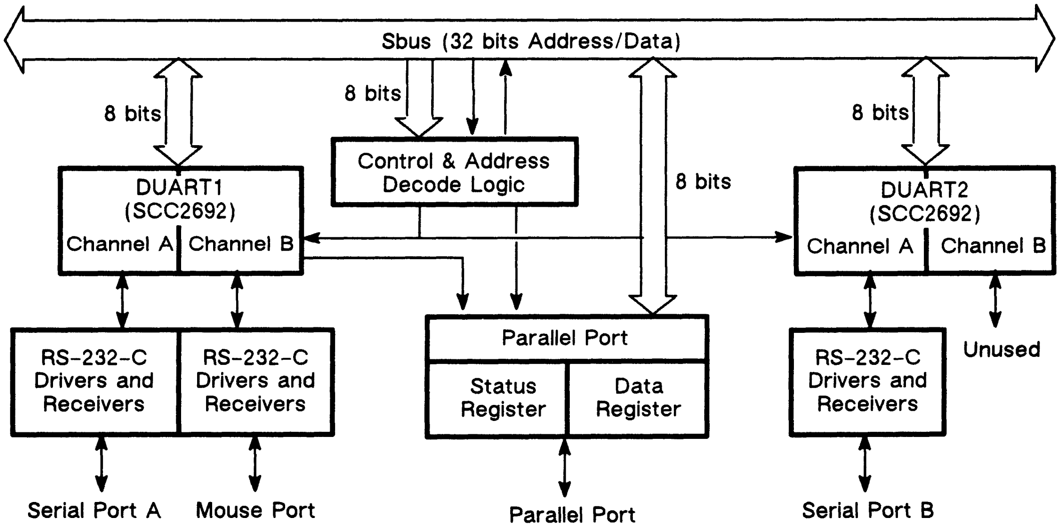


Figure 7-2 400 Series Serial and Parallel Ports

DUART

Each DUART is a Signetics SCC2692 Enhanced Programmable Communications Interface with two channels, A and B. For DUART1, the mouse port uses channel B and the serial port uses channel A. In 400 stations, channel A of a second DUART, DUART2, provides another serial port. The DUART clock input is 3.6864–MHz, and is not synchronized with the system clock.

Serial Ports

In 400 series stations, channel A of DUART1 and channel A of DUART2 support RS–232–C interfaces with modem support. In 300 series stations, channel A of DUART1 supports both RS–232–C and RS–422; only the Receive Data and Transmit Data lines are RS–422.

In both 300 and 400 series stations, channel B of DUART1 drives a mouse port using the RS–232–C interface. Only the Receive Data signal is used to communicate. The remaining signals — Transmit Data, Request to Send, and Data Terminal Ready — are used as power sources to the mouse.

Parallel Port

The parallel port consists of control logic and buffers that regulate the transmission of parallel data using either the Data Products or the Centronics protocol. The control logic includes two registers, the Parallel Port Data (PPD) register and the Parallel Port Status (PPS) register. In addition, DUART1 provides two control signals, Data Strobe and Data Select, which are programmed via the DUART's Output Port Configuration Register (OPCR), Set Output Port Bits Command (SOPBC) register, and Reset Output Port Bits Command (ROPBC) register.

Programming the Serial Ports

The serial ports are programmed via the DUART registers. The DUART registers are accessed as words and are aligned on word boundaries. Each channel has its own Mode, Command, Clock Select, and Status registers. DUART1 and DUART2 are programmed independently. Table 7–1 lists the DUART registers and their addresses.

CAUTION: *Do not change the contents of the Mode registers, the Clock Select registers, and the Output Port Configuration Register unless the DUART's receivers and transmitters are disabled. Changing the contents of these registers while the DUART is transmitting may cause errors. For example, the DUART may transmit an incorrect character if the number of bits per character is changed while the transmitter is active.*

CAUTION: *Do not sequentially read from and/or write to the registers; put two nonoperative instructions between register reads or writes. Signetics requires setup time between read/write inactive and read/write active.*

Initializing the Serial Ports

No special initialization process is necessary except to enable the keyboard speaker by programming the counter/timer. For information on the speaker, see Chapter 6, "Programming the Keyboard Interface and Speaker."

Resetting the Serial Ports

When the DUART is reset, it clears the following registers to 0:

- Status registers (SRA, SRB).
- Interrupt Mask Register (IMR).
- Interrupt Status Register (ISR).
- Output Port Configuration Register (OPCR).
- Output Port Register.

A reset also drives the RTS, DTR, and TxD outputs low.

Interrupts

The DUARTs interrupt the CPU when one of the following conditions occurs:

- Channel A or B transmitter is ready.
- Channel A or B receiver is ready or its first–in–first–out (FIFO) buffer is full.
- Channel A or B detected a beginning or end of a Break Character.
- The counter/timer has reached its terminal count or zero.
- The state of an input port changed.

IMR and ISR registers are associated with interrupts. Interrupts are masked via IMR. ISR defines the nature of active interrupts.

Table 7-1 Serial Port Register Addresses

Address	Register (A indicates channel A; B indicates channel B)	Type
DUART1		
FFF8 2000	Mode Registers A (MR1A, MR2A)	Read/Write
FFF8 2004	Status Register A (SRA) Clock Select Register A (CSRA)	Read Write
FFF8 2008	Command Register A (CRA)	Write [1]
FFF8 200C	Receive Holding Register A (RHRA) Transmit Holding Register A (THRA)	Read Write
FFF8 2010	Input Port Change Register (IPCR) Auxiliary Control Register (ACR)	Read Write
FFF8 2014	Interrupt Status Register (ISR) Interrupt Mask Register (IMR)	Read Write
FFF8 2018	Counter/Timer Upper Register (CTUR)	Read/Write
FFF8 201C	Counter/Timer Lower Register (CTLR)	Read/Write
FFF8 2020	Mode Registers B (MR1B, MR2B)	Read/Write
FFF8 2024	Status Register B (SRB) Clock Select Register B (CSRB)	Read Write
FFF8 2028	Command Register B (CRB)	Write [1]
FFF8 202C	Receive Holding Register B (RHRB) Transmit Holding Register B (THRB)	Read Write
FFF8 2030	Reserved	
FFF8 2034	Input Port Register (IPR) Output Port Configuration Register (OPCR)	Read Write
FFF8 2038	Start Counter Command Register Set Output Port Bits Commmand Register	Read Write
FFF8 203C	Stop Counter Command Register Reset Output Port Bits Command	Read Write
DUART2		
FFF8 2C00	Mode Registers A (MR1A, MR2A)	Read/Write
FFF8 2C04	Status Register A (SRA) Clock Select Register A (CSRA)	Read Write
FFF8 2C08	Command Register A (CRA)	Write [1]
FFF8 2C0C	Receive Holding Register A (RHRA) Transmit Holding Register A (THRA)	Read Write
FFF8 2C10	Input Port Change Register (IPCR) Auxiliary Control Register (ACR)	Read Write
FFF8 2C14	Interrupt Status Register (ISR) Interrupt Mask Register (IMR)	Read Write
FFF8 2C18	Counter/Timer Upper Register (CTUR)	Read/Write
FFF8 2C1C	Counter/Timer Lower Register (CTLR)	Read/Write
FFF8 2C20	Mode Registers B (MR1B, MR2B)	Unused
FFF8 2C24	Status Register B (SRB) Clock Select Register B (CSRB)	Unused Unused
FFF8 2C28	Command Register B (CRB)	Unused
FFF8 2C2C	Receive Holding Register B (RHRB) Transmit Holding Register B (THRB)	Unused Unused
FFF8 2C30	Reserved	
FFF8 2C34	Input Port Register (IPR) Output Port Configuration Register (OPCR)	Read Write
FFF8 2C38	Start Counter Command Register Set Output Port Bits Commmand Register	Unused Unused
FFF8 203C	Stop Counter Command Register Reset Output Port Bits Command	Unused Unused

[1] Do not read the Command Registers; reading them may hang the asynchronous line. (If you use this line for the system console, the workstation may hang and require a reset.)

MR1A, MR1B Mode Registers

MR1A	**Address FFF8 2000 (DUART1)**	**Read/Write**
	Address FFF8 2C00 (DUART2)	**Read/Write**
MR1B	**Address FFF8 2020 (DUART1)**	**Read/Write**

The Mode 1 Registers (MR1A and MR1B) define the operating modes of the serial ports. MR1n and MR2n share the same addresses; the state of the Reset Pointer command, in the Command register, and the state of DUART resets determines which set of mode registers is accessed. MR1n is selected when the DUART is reset or when the Reset Pointer command in the command register is asserted. After MR1n is accessed, MR2n is selected until the DUART is reset.

CAUTION: Do not change the contents of the Mode registers unless the DUART's receivers and transmitters are disabled.

7	6	5	4 3	2	1 0
RTSE	RRIS	EM	PM	PT	CF

Bit	Name	Function
7	RTSE	Request to Send Enable 0 Disables the request to send (RTS) signal. 1 Enables the RTS signal.
6	RRIS	Receiver Ready Interrupt Select 0 If a port is ready to receive data, the corresponding State of FIFO (SOFA or SOFB) bit of the interrupt status register (ISR) is set. 1 When the FIFO buffer fills, the corresponding State of FIFO (SOFA or SOFB) bit is set.
5	EM	Error Mode 0 Character 1 Block
4, 3	PM	Parity Mode 00 With parity 10 No parity 01 Force parity 11 Multidrop
2	PT	Parity Type 0 Even 1 Odd
1, 0	CF	Character Format (bits per character) 00 5 01 6 10 7 11 8

MR2A, MR2B — Mode Registers

MR2A	**Address FFF8 2000 (DUART1)**	**Read/Write**
	Address FFF8 2C00 (DUART2)	**Read/Write**
MR2B	**Address FFF8 2020 (DUART1)**	**Read/Write**

The Mode 2 Registers (MR2A and MR2B) define the operating modes of the serial ports. MR1n and MR2n share the same addresses; the state of the Reset Pointer command, in the Command register, and the state of DUART resets determines which set of mode registers is accessed. MR1n is selected when the DUART is reset or when the Reset Pointer command in the command register is asserted. After MR1n is accessed, MR2n is selected until the DUART is reset.

7 6	5	4	3 0
CHM	RTE	CTE	SBL

Bit	Name	Function
7, 6	CHM	Channel Mode 00 Normal 01 Auto-echo 10 Local loop 11 Remote loop
5	RTE	Request to Send Enable 0 Disable RTS. 1 Enable RTS.
4	CTE	Clear to Send Enable 0 Disable CTS. 1 Enable CTS.
3-0	SBL	Stop Bit Length 0 0.563 4 0.813 8 1.563 C 1.813 1 0.625 5 0.875 9 1.625 D 1.875 2 0.688 6 0.983 A 1.688 E 1.983 3 0.750 7 1.000 B 1.750 F 2.000

CSRA, CSRB — Clock Select

CSRA	**Address FFF8 2004 (DUART1)**	**Write Only**
	Address FFF8 2C04 (DUART2)	**Write Only**
CSRB	**Address FFF8 2024 (DUART1)**	**Write Only**

The Clock Select registers (CSRA, CRSB) define the receive and transmit baud rates.

CAUTION: Change the contents of the Clock Select registers only while the DUART's receivers and transmitters are disabled.

7 4	3 0
RCS	TCS

Bit	Name	Function
7–4	RCS	Receiver Clock Select Program the receiver clock for channel A (CSRA) or channel B (CSRB).
3–0	TCS	Transmitter Clock Select Program the transmitter clock for channel A (CSRA) or channel B (CSRB).

Two sets of baud rates are available via the Baud Rate Generator (BRG) bit (Auxiliary Control Register bit 7) and the Clock Select registers. These registers define the clock source and baud rate. Table 7–2 lists the baud rates for a 3.6864 MHz clock.

Table 7–2 Baud Rate Generator Characteristics

CSR Bits	Baud Rate			
(7–4 or 3–0)	BRG = 0	BRG = 1	16x Clock (KHz)	Error (%)
0000	50	75	0.8 (1.2)	0 (0)
0001	110	110	1.759	-0.069
0010	134.5	134.5	2.153	0.059
0011	200	150	3.2 (2.4)	0 (0)
0100	300	300	4.8	0
0101	600	600	9.6	0
0110	1,200	1,200	19.2	0
0111	1,050	2,000	16.756 (32.056)	-0.26 (0.175)
1000	2,400	2,400	38.4	0
1001	4,800	4,800	76.8	0
1010	7,200	1,800	115.2 (28.8)	0 (0)
1011	9,600	9,600	153.6	0
1100	38,400	19,200	614.4 (307.2)	0 (0)
1101	Timer	Timer		
1110	IPx=16x	IPx=16x		
1111	IPx=1x	IPx=1x		

SRA, SRB Status

SRA	**Address FFF8 2004 (DUART1)**	**Read Only**
	Address FFF8 2C04 (DUART2)	**Read Only**
SRB	**Address FFF8 2024 (DUART1)**	**Read Only**

The Status registers (SRA and SRB) contain DUART status information.

7	6	5	4	3	2	1	0
BRK	FE	PE	OE	TxE	TxRDY	FFL	RxRDY

Bit	Name	Function
7	BRK	Break 0 No error. 1 The channel received an all-zero character without a stop bit. Inhibits further entries into the FIFO until the RxD line returns to the idle (mark) state.
6	FE	Framing Error 0 No error. 1 The last character received did not have a stop bit.
5	PE	Parity Error 0 No error. 1 The last character received had the incorrect parity.
4	OE	Overrun Error 0 No error. 1 One or more characters in the received data stream were lost.
	NOTE:	To clear BRK, FE, PE and OE, write 0x40 into the corresponding Command Register (CRA or CRB).
3	TxE	Transmitter Empty 0 The transmit register(s) have data, or the transmitter is disabled. 1 Both the transmit holding register and the transmit shift register are empty.
2	TxRDY	Transmitter Ready 0 The transmit holding register has data, or the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.
1	FFL	FIFO Full 0 The receive holding register is not full; it has room for more data. 1 The receive holding register (FIFO) is full.
0	RxRDY	Receiver Ready 0 The receive holding register is empty. 1 The receive holding register (FIFO) has received a character for the CPU to read.

CRA, CRB Command

CRA	Address FFF8 2008 (DUART1)	Write Only
	Address FFF8 2C08 (DUART2)	Write Only
CRB	Address FFF8 2028 (DUART1)	Write Only

The Command registers (CRA, CRB) contain DUART command information. The commands executed by the CMD bits are performed on the corresponding channel.

CAUTION: Do not read the Command registers; reading them might hang the line. If connected to the system console, the system might hang and require a reset.

7 4	3	2	1	0
MSC	DTx	ETx	DRx	ERx

Bit	Name	Function	
7-4	CMD	Commands	
		Bits	**Function**
		0000	Not used.
		0001	Reset the Mode Register Pointer. Points the mode register pointer to MR1.
		0010	Reset the Receiver. Disables the receiver and flushes the FIFO.
		0011	Reset the Transmitter.
		0100	Clear BRK, PE, FE, and OE. Clear the break (BRK), parity error (PE), framing error (FE), and overrun error (OE) by clearing SRn[7-4] to 0.
		0101	Reset the Change In Break Interrupt. Clears the Change in Break (BKA or BKB) in the Interrupt Status Register to 0.
		0110	Start Break. Forces the TxD output high (spacing). If the transmitter is empty, the start of the break condition is delayed up to two bit times. If the transmitter is active, the break begins when character transmission is completed. If a character is in the channel's Transmit Holding Register (THR), the start of the break is delayed until that character, or any others loaded subsequently, are transmitted. To accept this command, the transmitter must be enabled.
		0111	Stop Break. The TxD line goes low within two bit times, and remains low for one bit time before the next character is transmitted.
		1000	Assert RTS. Assert RTS high.
		1001	Clear RTS. Clear RTS low.
		1010	Not used.
		1011	Not used.
		1100	Not used.
		1101	Not used.

(continued)

Bit	Name	Function
7–4	CMD	Commands (continued) **Bits** **Function** 1110 Stop the DUART Oscillator. Stop the DUART oscillator, suspending all functions requiring this clock. The contents of all registers are saved. Disable the transmitter and receiver before issuing this command. This command is in Command Register A (CRA) only. 1111 Reset the Power Down Mode. Reset the power down mode to start DUART oscillator running again. This command is in Command Register A (CRA) only.
3	DTn	Disable Transmitter 0 No action, transmitter state not changed. 1 Disable the transmitter.
2	ETn	Enable Transmitter 0 No action, transmitter state not changed. 1 Enable the transmitter.
1	DRn	Disable Receiver 0 No action, receiver state not changed. 1 Disable the receiver.
0	ERn	Enable Receiver 0 No action, receiver state not changed. 1 Enable the receiver.

(concluded)

ACR — Auxiliary Control

Address FFF8 2010 (DUART1) — Write Only
Address FFF8 2C10 (DUART2) — Write Only

The Auxiliary Control Register (ACR) is used to select baud rates, select the counter/timer mode and source, and enable interrupts.

CAUTION: *Some changes to the Auxiliary Control Register must be made only while the counter/timer is stopped.*

7	6 4	3	2	1	0
BRG	CTM	EIP3CI	EIP2CI	EIP1CI	EIP0CI

Bit	Name	Function
7	BRG	Baud Rate Generator set select. Select one of two sets of baud rates. CSRA or CSRB selects the actual baud rate. 0 Selects the following subset of baud rates: 50, 110, 134.5, 200, 300, 600, 1200, 1050, 2400, 4800, 7200, 9600, 38400, Timer, IPx=16x, IPx=1x. 1 Selects the following subset of baud rates: 75, 110, 134.5, 150, 300, 600, 1200, 2000, 2400, 4800, 1800, 9600, 19200, Timer, IPx=16x, IPx=1x.
6–4	CTM	Counter/Timer Mode and Source See "Speaker Registers" in Chapter 6, "Programming the Keyboard Port."
3	ERII	Enable the RI Interrupt 1 If the state of RI changes, the Input Port Change (IPC) bit of the ISR register will be set to 1. 0 A change in state of RI will not affect IPC.
2	ECDI	Enable the DCD Interrupt 1 If the state of DCD changes, the Input Port Change (IPC) bit of the ISR register will be set to 1. 0 A change in state of DCD will not affect IPC.
1	ESRI	Enable the DSR Interrupt 1 If the state of DSR changes, the Input Port Change (IPC) bit of the ISR register will be set to 1. 0 A change in state of DSR will not affect IPC.
0	ETSI	Enable the CTS Interrupt 1 If the state of CTS changes, the Input Port Change (IPC) bit of the ISR register will be set to 1. 0 A change in state of CTS will not affect IPC.

IPCR — Input Port Change

Address FFF8 2010 (DUART1) — Read Only

Address FFF8 2C10 (DUART2) — Read Only

The Input Port Change Register (IPCR) contains status information on input ports 0 through 3. These ports input the RI, DCD, DSR, and CTS signals. DUART1 receives these signals from serial port A, and DUART2 receives these signals from serial port B.

7	6	5	4	3	2	1	0
IP3SC	IP2SC	IP1SC	IP0SC	IP3S	IP2S	IP1S	IP0S

Bit	Name	Function
7	IP3SC	IP3 State Change (RI) 0 RI did not change state. 1 RI changed state.
6	IP2SC	IP2 State Change (DCD) 0 DCD did not change state. 1 DCD changed state.
5	IP1SC	IP1 State Change (DSR) 0 DSR did not change state. 1 DSR changed state.
4	IP0SC	IP0 State Change (CTS) 0 CTS did not change state. 1 CTS changed state.
3	IP3S	IP3 Current State (RI) 0 RI is asserted. 1 RI is not asserted.
2	IP2S	IP2 Current State (DCD) 0 DCD is asserted. 1 DCD is not asserted.
1	IP1S	IP1 Current State (DSR) 0 DSR is asserted. 1 DSR is not asserted.
0	IP0S	IP0 Current State (CTS) 0 CTS is asserted. 1 CTS is not asserted.

IMR Interrupt Mask

Address FFF8 2014 (DUART1) Write Only

Address FFF8 2C14 (DUART2) Write Only

The Interrupt Mask Register (IMR) controls interrupts generated through the interrupt register.

7	6	5	4	3	2	1	0
MIPC	MBKB	MSOFB	MTxRB	MCTR	MBKA	MSOFA	MTxRB

Bit	Name	Function
7	MIPC	Input Port Change Interrupt Mask 0 Mask the input port change interrupt. 1 Enable the input port change interrupt.
6	MBKB	Channel B Break Interrupt Mask 0 Mask the channel B break interrupt. 1 Enable the channel B break interrupt.
5	MSOFB	Channel B State of FIFO Interrupt Mask 0 Mask the channel B state of FIFO interrupt. 1 Enable the channel B state of FIFO interrupt.
4	MTxRB	Transmitter B Ready Interrupt Mask 0 Mask the transmitter B ready interrupt. 1 Enable the transmitter B ready interrupt.
3	MCTR	Counter Ready Interrupt Mask 0 Mask the counter ready interrupt. 1 Enable the counter ready interrupt.
2	MBKA	Channel A Break Interrupt Mask 0 Mask the channel A break interrupt. 1 Enable the channel A break interrupt.
1	MSOFA	Channel A State of FIFO Interrupt Mask 0 Mask the channel A state of FIFO interrupt. 1 Enable the channel A state of FIFO interrupt.
0	MTxRA	Transmitter A Ready Interrupt Mask 0 Mask the transmitter A ready interrupt. 1 Enable the transmitter A ready interrupt.

ISR Interrupt Status

Address FFF8 2014 (DUART1) Read Only

Address FFF8 2C14 (DUART2) Read Only

The Interrupt Status register (ISR) provides the status of all interrupt sources. When an interrupt condition occurs, the representative bit in the ISR register is set. If the corresponding bit in the Interrupt Mask Register (IMR) is also set, the DUART generates an interrupt to the CPU.

7	6	5	4	3	2	1	0
IPC	BKB	SOFB	TxRB	CTR	BKA	SOFA	TxRA

Bit	Name	Function
7	IPC	Input Port Change 0 The input ports have not changed state. 1 One of the four input ports has changed state.
	NOTE:	The contents of IPC are valid only if the ACR register bits [3–0] have been set to enable input port changes to be registered into IPC.
6	BKB	Channel B Change in Break Used in DUART1 for serial port A; not used in DUART2. 0 The channel B receiver did not detect a break. 1 The channel B receiver detected a break.
5	SOFB	Channel B State of FIFO Used in DUART1 for serial port A; not used in DUART2. SOFB is programmed to respond to one of two conditions by the receiver ready interrupt select (RRIS) bit of the mode register (MR1B). If RRIS is set to 1, SOFB reacts to the state of the receiver, whether it is full or ready to receive data. If RRIS is cleared to 0, SOFB reacts to the state of the FIFO receive buffer, whether full or not full. 0 Receiver B is not ready to receive data, or the FIFO buffer is not full. 1 Receiver B is ready to receive data, or the FIFO buffer is full.
4	TxRB	Channel B Transmitter Ready Used in DUART1 for serial port A; not used in DUART2. 0 The transmit holding register is full, the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.

(continued)

Bit	Name	Function
3	CTR	Counter/Timer Ready CTR is programmed to respond to one of two modes, the Counter mode or the Timer mode, by the Counter/Timer mode (CTM) bits of the Auxiliary Control register (ACR). 0 The counter has not reached its terminal count, or the timer has not reached the second zero state. 1 The counter has reached its terminal count, or the timer has reached its second zero state. In the timer mode, CTR is set every other time that the counter reaches zero, not every time that the timer reaches zero.
2	BKA	Channel A Change in Break Used in DUART1 for serial port A, and DUART2 for serial port B. 0 The channel A receiver did not detect a break. 1 The channel A receiver detected a break.
1	SOFA	Channel A State of FIFO Used in DUART1 for serial port A, and DUART2 for serial port B. SOFA is programmed to respond to one of two conditions by the receiver ready interrupt select (RRIS) bit of the mode register (MR1A). If RRIS is set to 1, SOFA reacts to the state of the receiver A, whether it is full or ready to receive data. If RRIS is cleared to 0, SOFA reacts to the state of the FIFO receive buffer, whether full or not full. 0 Receiver A is not ready to receive data, or the FIFO buffer is not full. 1 Receiver A is ready to receive data, or the FIFO buffer is full.
0	TxRA	Channel A Transmitter Ready Used in DUART1 for serial port A, and DUART2 for serial port B. 0 The transmit holding register is full, or the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.

(concluded)

CTUR, CTLR — Counter/Timer

CTUR	**Address FFF8 2018 (DUART1)**	**Read/Write**
	Address FFF8 2C18 (DUART2)	**Read/Write**
CTLR	**Address FFF8 201C (DUART1)**	**Read/Write**
	Address FFF8 2C1C (DUART2)	**Read/Write**

The Counter/Timer Registers (CTUR, CTLR) together form a 16–bit counter. (These registers are write–only; a read of these addresses is actually a read of the physical counter/timer.)

CTUR is defined as follows:

7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8

Bit	Name	Function
7–0	C15–C8	Counter/Timer Bits 15–8

The CTLR bits are defined as follows:

7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Bit	Name	Function
7–0	C7–C0	Counter/Timer Bits 7–0

OPCR Output Port Configuration

Address FFF8 2034 (DUART1) Write Only

Address FFF8 2C34 (DUART2) Write Only

The Output Port Configuration register (OPCR) configures the output ports OP2–OP7.

CAUTION: Change the contents of the Output Port Configuration Register only while the DUART's receivers and transmitters are disabled.

7	6	5	4	3	2	1	0
OP7	OP6	OP5	OP4	OP3		OP2	

Bit	Name	Function
7	OP7	Output Port 7 (DATASTRB) Used in DUART1 for the parallel port's DATASTRB signal; not used for the serial ports.
6	OP6	Output Port 6 (SELECTIN) Used in DUART1 for the parallel port's SELECTIN signal; not used for the serial ports.
5	OP5	Output Port 5 (DTR) Used in DUART1 for the mouse's DTR signal; not used in DUART2. 0 Invert DTR. 1 Drive the complement of bit 5 (RxRDY/FFULLB) of the ISR register.
4	OP4	Output Port 4 (SPKR_EN) Used in DUART1 for the SPKR_EN signal; not used in DUART2. 0 Invert SPKR_EN. 1 Drive the complement of bit 1 (RxRDY/FFULLA) of the ISR register.
3, 2	OP3	Output Port 3 (K_TIMER) Used in DUART1 for the K_TIMER signal; not used in DUART2. 00 Invert K_TIMER. 01 C/T OUTPUT 10 TxCB (1x) 11 RxCB (1x)
1, 0	OP2	Output Port 2 (DTR) Select the DTR output for serial port A (DUART1) or serial port B (DUART2). 00 Invert DTR. 01 TxC (16x) 10 TxC (1x) 11 RxC (1x)

Programming the Mouse Port

The mouse connects to the system board through port B of DUART1. This section defines how to interpret and program the mouse.

Initializing the Mouse Port

You should initialize the mouse port with the following settings:

Baud rate	1200
Data bits	8
Start bits	1
Stop bits	1
Parity	None

Data Protocol

When there is a change in the state of the mouse, the mouse transmits five bytes of data to the system board. The start of a data block is indicated by a sync byte whose upper five bits are 10000. The lower three bits define the state of the switches (0 indicates depressed.) The next four bytes (DeltaX and DeltaY repeated) contain two updates of the mouse movement counters. DeltaX is the horizontal distance moved relative to the grid of the pad, and DeltaY is the corresponding veritical distance. Each byte is a two's-complement 8-bit binary number.

After these five bytes have been transmitted, additional bytes may be sent before another sync byte is transmitted. The tracking software should ignore these additional bytes.

Table 7-3 Mouse Data Protocol

Byte	Bits							
SYNC	1	0	0	0	0	L	M	R
DELTAX1	X7	X6	X5	X4	X3	X2	X1	X0
DELTAY1	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
DELTAX2	X7	X6	X5	X4	X3	X2	X1	X0
DELTAY2	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

While the mouse is resting on a flat surface with the cord away from you, positive motion is to the right (X-direction) or upward (Y-direction, toward the cord.) Bit 7 of the DeltaX and DeltaY bytes defines whether the direction is positive or negative (0 is positive, 1 is negative.)

Tracking Software

A software tracking algorithm is:

1. Read a byte.
2. If the upper bits are 10000, then save the lower three bits in SWITCH_STATUS. Otherwise repeat Step 1.
3. Process the lower three switch information bits if necessary.
4. Get the next byte and add its value to the variable that is accumulating horizontal movement. Do not forget to sign–extend the value to the size of the variable used to store the accumulated movement.
5. Get the next byte and add its value to the vertical movement.
6. Get the next byte and add its value to the horizontal movement.
7. Get the next byte and add its value to the vertical movement.
8. Go to step 1.

Do not ignore DeltaX2 and DeltaY2.

Programming Hints

The RS–232–C interface on the mouse only transmits data. It does not use the handshake signals (DSR, CTS, etc.). The tracking software can either ignore these signals or program the DUART to ignore these signals.

When the mouse is powered up, it transmits a continuous Break or Start bit for approximately 150 μs. This is the duration of its reset period. You may need to reset the DUART after this Break.

The mouse does not use parity; all eight bits of each byte contain valid data. Disable operating system features that clear high–order bits, swallow nulls, respond to ^S or ^Q, and/or replicate Delete.

Because the mouse transmits five bytes of data at 1200 baud, the maximum mouse velocity must be less than 51 in/s for real–time tracking. The mouse's position is available 48 times per second.

Sensitivity

Many programmers adjust the sensitivity of the mouse by filtering the mouse input. Sensitivity is changed by multiplying or dividing the delta motion values by a constant before moving the cursor on the screen. Multiplying causes the cursor to move farther relative to the mouse motion, and dividing causes the cursor to move less, giving finer control over its motion.

Sensitivity can also be nonlinear; rapid mouse motion could move the cursor a greater distance than slower mouse motion. A common algorithm doubles the cursor motion relative to the mouse motion if the mouse is moved faster than 64 transitions per second.

The appropriate sensitivity depends on the application and the user.

Programming the Parallel Port

The parallel port is used to transmit parallel data from the system board.

Registers

The parallel port uses the following registers:

- Output Port Configuration Register (OPCR).
- Set Output Port Bits Command (SOPBC) register.
- Reset Output Port Bits Command (ROPBC) register.
- Parallel Port Data (PPD) register.
- Parallel Port Status (PPS) register.

The OPCR, SOPBC and ROPBC registers reside in DUART1. Table 7-4 illustrates the registers and addresses.

Table 7-4 Addresses of Parallel Port Registers

Address	Register	Type
FFF8 2034	Output Port Configuration Register (OPCR)	Write
FFF8 2038	Set Output Port Bits Command (SOPBC) register	Write
FFF8 203C	Reset Output Port Bits Command (ROPBC) register	Write
FFF8 2400	Parallel Port Data (PPD) register Parallel Port Status (PPS) register	Write Read

Interrupts and Transmitting Data

When the CPU receives a parallel port interrupt, the interrupt service routine must read the PPS register to determine the nature of the interrupt. If it is a Parallel Port Demand (PPD) interrupt, the interrupt service routine should write a byte of data to the PPD register; then pulse the Data Strobe signal (high for a Data Products interface or low for a Centronics interface) to transmit the data. To pulse the Data Strobe line, the interrupt service routine must write to the SOPBC and ROPBC registers. After receiving the data, the parallel device must issue a new demand (Data Products) or an acknowledgement (Centronics) to request another byte of data. This demand or acknowledgement generates a new PPD interrupt and the process repeats until all of the data is transmitted. After the program sends the last character to the parallel port, it should mask out the parallel port interrupt until ready to send more data.

NOTE: The interrupt service routine should establish a set of parallel port interrupt priorities so as not to send data when the parallel port has a problem; the interrupt servce routine should examine the parallel port's status before transmitting each character.

Programming the Data Strobe and Data Select Signals

The Set Output Port Bits Command (SOPBC) and Reset Output Port Bits Command (ROPBC) registers control the Data Strobe (PP_DSTB) and the Data Select (PP_SEL) signals. You must program these control signals for the appropriate polarity: Data Products or Centronics. The program must ensure that the Data Strobe signal meets the timing specification shown in Figure 7–3 for a Data Products interface or in Figure 7–4 for a Centronics interface.

The following pages describe the parallel port registers.

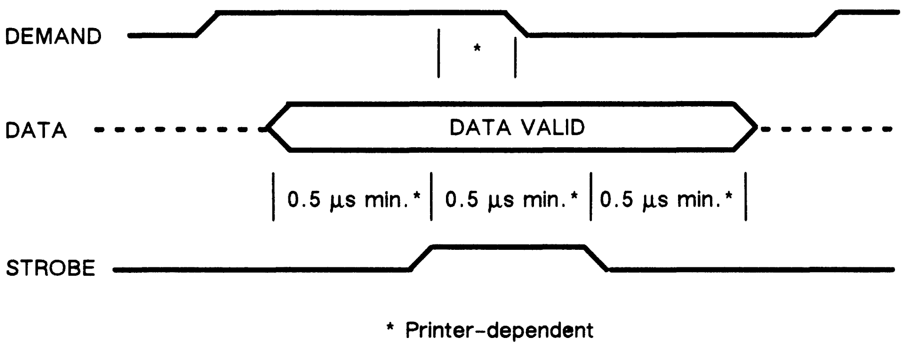


Figure 7–3 Data Strobe Timing for a Data Products Interface

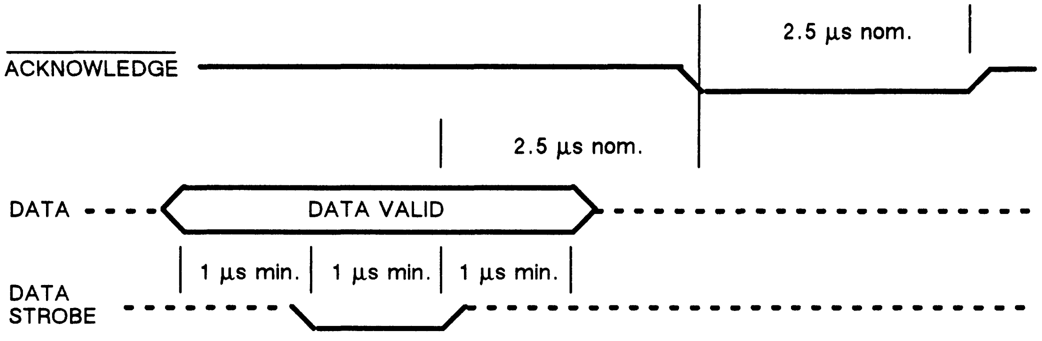


Figure 7–4 Data Strobe Timing for a Centronics Interface

OPCR Output Port Configuration

Address FFF8 2034 Write Only

The upper two bits of the Output Port Configuration Register (OPCR), located in DUART1, configure the DATASTRB and SELECTIN signals of the parallel port.

7	6	5	0
OP7	OP6	*	

Bit	Name	Function
7	OP7	Output Port 7 Configuration (DATASTRB) Used in DUART1 for the DATASTRB signal; not used in DUART2. 0 Drive the complement of the current DATASTRB signal. 1 Drive the complement of bit 4 (TxRDYB) of the ISR register.
6	OP6	Output Port 6 Configuration (SELECTIN) Used in DUART1 for the SELECTIN signal; not used in DUART2. 0 Drive the complement of the current SELECTIN signal. 1 Drive the complement of bit 0 (TxRDYA) of the ISR register.
5-0	*	Used for serial ports; not used for parallel port.

SOPBC Set Output Port Bits Command

Address FFF8 2038 Write Only

The upper two bits of the Set Output Port Bits Command (SOPBC) register, located in DUART1, set the DATASTRB and SELECTIN signals of the parallel port.

7	6	5	0
SO7	SO6	0	

Bit	Name	Function
7	SO7	Set Output Port 7 (DATASTRB) 1 Assert DATASTRB Low. 0 Leave DATASTRB unchanged.
6	SO6	Set Output Port 6 (SELECTIN) 1 Assert SELECTIN Low. 0 Leave SELECTIN unchanged.
5–0		Must be 0s for parallel port accesses.

ROPBC Reset Output Port Bits Command

Address FFF8 203C Write Only

The upper two bits of the Reset Output Port Bits Command (ROPBC) register, located in DUART1, reset the DATASTRB and SELECTIN signals of the parallel port.

7	6	5	0
RO7	RO6	0	

Bit	Name	Function
7	RO7	Reset Output Port 7 (DATASTRB) 1 Set DATASTRB High. 0 Leave DATASTRB unchanged.
6	RO6	Reset Output Port 6 (SELECTIN) 1 Set SELECTIN High. 0 Leave SELECTIN unchanged.
5-0		Must be 0s for parallel port accesses.

PPD Parallel Port Data

Address FFF8 2400 Write Only

The Parallel Port Data (PPD) register is a temporary buffer for data being transmitted through the parallel port.

7 0
PPD

Bit	Name	Function
7-0	PPD	Parallel Port Data Contains the byte of data that the parallel port will transmit.

PPS Parallel Port Status

Address FFF8 2400 Read Only

The Parallel Port Status (PPS) register contains status information on the parallel port. If a parallel port interrupt occurs, the interrupt service routine should read the PPS register to find the nature of the parallel port interrupt.

7 6	5	4	3	2	1	0
Unused	PPI	PPF	PPD	PPE	PPL	PPB

Bit	Name	Function
7, 6	Unused	Ignore these bits.
5	PPR	Parallel Port Ready 1 The parallel port is ready. 0 The parallel port is not ready.
4	PPF	Parallel Port Fault 1 A parallel port fault. has occurred. 0 There is no parallel port. fault.
3	PPD	Parallel Port Demand 1 The parallel port demand (or acknowledgement) is asserted. 0 The parallel port demand (or acknowledgement) is not asserted.
2	PPE	Parallel Port Paper Empty (Centronics only) 1 The printer is out of paper. 0 The printer is not out of paper.
1	PPL	Parallel Port On Line 1 The parallel port is on line. 0 The parallel port is off line.
0	PPB	Parallel Port Busy (Centronics) 1 The parallel port is busy. 0 The parallel port is ready. Parallel Port Busy (Data Products) 1 The parallel port is ready. 0 The parallel port is busy.

End of Chapter

Chapter 8
Programming the Local Area Network Interface

This chapter describes the following topics:

- The LAN interface.
- Ethernet frame transfers.
- How to program the LAN interface.

The local area network interface provides an Ethernet interface for IEEE 802.3 based communications with other systems. The CPU acts as the I/O controller for the LAN interface and main memory provides its data buffer.

Components of the LAN Interface

As shown in Figure 8–1, the LAN interface consists of an Sbus interface, address extension logic, an Ethernet controller, an Ethernet serial interface, and an attachment unit interface (AUI) connector.

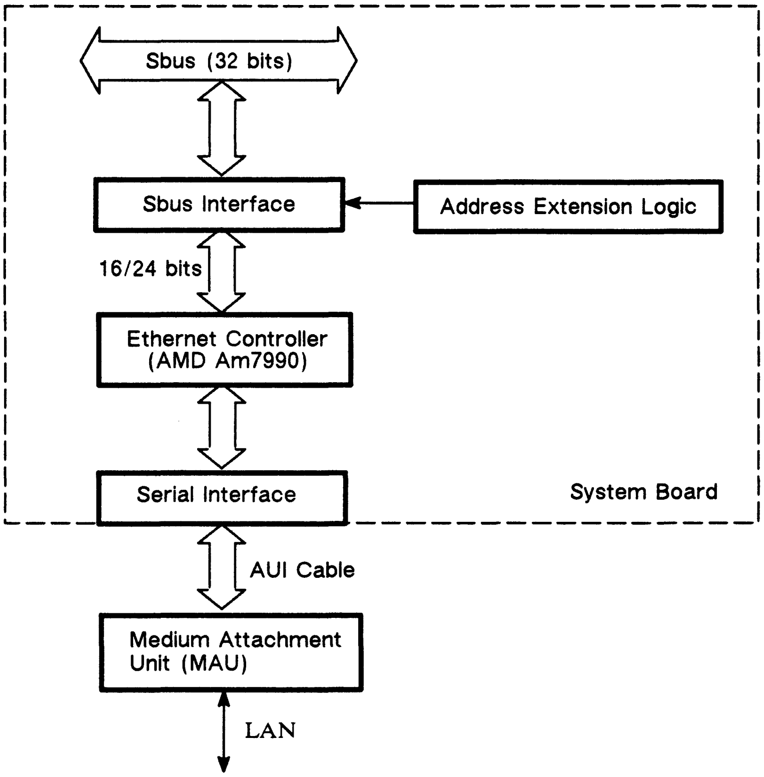


Figure 8–1 Components of the LAN Interface

Sbus Interface

The Sbus interface connects the Ethernet controller to the Sbus. When the Ethernet controller is master of the Sbus, this interface hardware adds the eight address bits from the address extension logic to the 24-bit addresses produced by the controller to provide the required 32-bit addresses for the Sbus.

The LAN interface neither asserts nor responds to any kind of bus error.

Since the Ethernet controller runs at a fixed 10–MHz rate and the Sbus is a synchronous 16.67–MHz or 20–MHz bus, the Sbus interface contains logic to synchronize the Ethernet controller and Sbus operations.

Address Extension Logic

The LAN interface provides an address extension that is not accessible by software. This extension contains the eight high-order address bits driven on the Sbus whenever the Ethernet controller is bus master. Since this extension is set by hardware to 0, the LAN interface can address only the first 16 Mbytes of main memory.

Ethernet Controller

The Ethernet controller is an Advanced Micro Devices AM7990 local area network controller for Ethernet (LANCE) chip with 24 address bits and 16 data bits. It functions as either a bus master or a bus slave. The Ethernet controller uses the CPU as the host processor and main memory as its buffer for Ethernet data.

Serial Interface

The Ethernet serial interface converts the transmit data output of the Ethernet controller into Manchester-encoded output for the Ethernet; and conversely, it performs Manchester decoding on the Ethernet's input to provide receive-data input for the Ethernet controller.

AUI Connector and Cable

The AUI cable is a 15-pin cable with D15 connectors, as described by the IEEE 802.3 specification. See Appendix D, "System Board Connectors" for the signals. The AUI cable connects the workstation to a medium attachment unit (MAU).

Medium Attachment Unit (MAU)

The medium attachment unit (MAU) contains the Ethernet transceiver that links the system board's Ethernet I/O with the Ethernet LAN. The MAU isolates the LAN from the AUI cable and controllers, and connects to one of many Ethernet media. The MAU can be any one of the following types of external 10-MHz MAUs:

- 10BASE5 (Ethernet).
- 10BASE2 (Cheapernet or Thin Ethernet).
- 10BROAD36 (Ethernet over CATV).
- 10BASET (proposed Ethernet over twisted pair).
- any other 10-MHz AUI-compatible MAU or MAU-like device.

Ethernet Frame Transfers

The LAN interface defines the following logical path for incoming and outgoing Ethernet frames.

Incoming Frame Path

An incoming Ethernet frame traverses the following path from the network media to the I/O driver buffers:

1. The Ethernet frame enters the LAN interface from the network media via the AUI connector.
2. The serial interface Manchester decodes the frame.
3. The Ethernet controller assembles the bits of the frame into a 16-bit half-word.
4. The Ethernet controller writes the frame into the area of main memory specified by the buffer descriptors.
5. The CPU copies the frame from main memory into the I/O driver buffers.

Outgoing Frame Path

An outgoing Ethernet frame traverses the following path from the I/O driver buffers to the network media:

1. The CPU copies the frame from the I/O driver buffer into the LAN interface's memory buffers in main memory.
2. The Ethernet controller reads the frame out of the buffers in main memory specified by the buffer descriptors.
3. The Ethernet controller assembles the frame into a complete Ethernet frame.
4. The serial interface Manchester encodes the frame.
5. The serial interface sends the frame to the network media via the AUI connector.

Programming the LAN Interface

The only software accessible components of the LAN interface are the Ethernet controller's internal registers. This section describes the following:

- Programming the Ethernet controller registers.
- Allocating memory to the LAN interface.
- LAN interface data structures.
- LAN interface software environment.
- Initializing the LAN interface.
- Resetting the LAN interface.
- LAN interface interrupts.

For a complete description of how to program the controller, refer to Advance Micro Device's *Am7990 Local Area Network Controller (LANCE) Technical Manual.*

Programming the Ethernet Controller Registers

The Ethernet interface has two addressable 32-bit registers, which correspond to the Ethernet controller's Register Address Pointer (RAP) register and Register Data Port (RDP) register. Together, these two registers provide access to other registers within the Ethernet controller.

Both registers are 32-bit, word-access-only registers with the following addresses: FFF8 C004 (RAP) and FFF8 C000 (RDP). This means that software should access these registers as 32-bit words using the RISC word load and store instructions. In C these registers must be defined as **int**. Since the contents of each register appear in bits 15-0 of the accessed word, software must mask out bits 31-16 of the accessed word.

The rest of this section briefly describes these two registers and the four Ethernet controller Control/Status registers (CSR0, CSR1, CSR2, CSR3). It also gives the required configurations that these registers must have for the LAN interface to operate properly.

NOTE: You must disable parity checking by the data CMMU before reading the Ethernet controller registers. For information on programming the CMMU, refer to the *MC88200 User's Manual.*

For a complete description of these registers, refer to Advanced Micro Device's *Am7990 Local Area Network Controller (LANCE) Technical Manual.*

RAP Register Address Pointer

Address FFF8 C004 Read/Write

The Register Address Pointer (RAP) register selects one of the Control and Status Registers (CSRn).

31	16
Reserved	

15	2	1	0
Reserved		CSR SELECT	

Bit	Name	Function
31-2	Reserved	Undefined.
1-0	CSR SELECT	Control and Status Register Select 00 CSR0 01 CSR1 10 CSR2 11 CSR3

RDP Register Data Port

Address FFF8 C000 Read/Write

The Register Data Port (RDP) register passes data between the CPU and the Control and Status Registers (CSRn).

31	16
Reserved	

15	0
CSR DATA	

Bit	Name	Function
31–16	Reserved	Undefined.
15–0	CSR Data	Contents of the CSR selected by the RAP register. Writing data into these bits wtites the data into the selected CSR. Reading data from these bits, reads the data from the selected CSR. CSR1, CSR2, and CSR3 are accessible only when CSR0 bit 2 (STOP) is 1. For the contents of a specific register, refer to the description of the CSR that follow.

CSR0 Control and Status Register 0

Read/Write

The Control and Status Register 0 (CSR0) passes control and status between the CPU and the Ethernet controller.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR	BABL	CERR	MISS	MERR	RINT	TINT	IDON	INTR	INEA	RXON	TXON	TDMD	STOP	STRT	INIT

Bit	Name	Function	Type
15	ERR	Error 1 Indicates that one of the following errors has occurred: Babble (BABL), Collision (CERR), Missed Packet (MISS), or Memory (MERR).	Read
14	BABL !	Babble - Transmitter timeout 1 Indicates excessive length in the transmit buffer.	Read/Write
13	CERR !	Collision error 1 Indicates a collision after transmission transceiver test feature.	Read/Write
12	MISS !	Missed Packet 1 Indicates the receiver lost a packet.	Read/Write
11	MERR !	Memory Error 1 Indicates a memory error occurred.	Read/Write
10	RINT !	Receiver Interrupt 1 Indicates the receiver generated an interrupt.	Read/Write
9	TINT !	Transmitter Interrupt 1 Indicates the transmitter generated an interrupt.	Read/Write
8	IDON !	Initialization Done 1 Indicates that the Ethernet controller has completed its initialization process.	Read/Write
7	INTR	Interrupt Flag 1 Indicates the Ethernet controller has generated an interrupt.	Read
6	INEA	Interrupt Enable 1 Enables Ethernet controller interrupts.	Read/Write
5	RXON	Receiver On 1 Indicates the receiver is enabled.	Read
4	TXON	Transmitter On 1 Indicates the transmitter is enabled.	Read

! Writing a 1 to these bits clears them; writing a 0 to them has no effect.

(continued)

Bit	Name	Function	Type
3	TDMD	Transmit Demand 1 Causes the Ethernet controller to access the transmit descriptor ring without waiting for the polling time interval to elapse.	Write
2	STOP	Stop 1 Disables the Ethernet controller from all external activity and clears its internal logic. Do not write a 1 to this bit. To reset the Ethernet controller and set this bit to 1, use the Reset Ethernet Subsystem bit in the CPU's Diagnostic Control Register (DCR).	Read/Write
1	STRT	Start 1 Enables the Ethernet controller to send and receive packets, perform DMA operations, and manage its buffers.	Read/Write
0	INIT	Initialize 1 Causes the Ethernet controller to begin its initialization procedure.	Read/Write

(concluded)

CSR1 Control and Status Register 1

Read/Write

The Control and Status Register 1 (CSR1) passes control and status between the CPU and the Ethernet controller.

15	1	0
IADRL		0

Bit	Name	Function
15–1	IADRL	Initialization Block Address Contains the low-order bits (15–1) of the initialization block. Must be 0 for the initialization block to begin on an even byte.

CSR2 Control and Status Register 2

Read/Write

The Control and Status Register 2 (CSR2) passes control and status between the CPU and the Ethernet controller.

15 8	7 0
Reserved	IADRH

Bit	Name	Function
15–8	Reserved	Reserved. Must be written to with 0s; read as 0s.
7–0	IADRH	Initialization Block Address Contains the high-order bits (23–16) of the initialization block.

CSR3 Control and Status Register 3

Read/Write

The Control/Status Register 3 (CSR3) passes control and status between the CPU and the Ethernet controller.

15	3	2	1	0
Reserved		1	0	0

Bit	Name	Function
15–3	Reserved	Reserved. Must be written to with 0s; read as 0s.
2	1	Byte Swap (BSWP) Must be 1 so that the Ethernet controller swaps the high and low-order bytes on DMA data transfers. Initialization block data and description ring entries are not swapped.
1	0	ALE Control (ACON) Must be 0 so that the ALE (Address Latch Enable) signal is asserted high.
0	0	Byte Control (BCON) Must be 0 so that the Ethernet controller's Byte Mask and Hold I/O pins are defined correctly for the LAN interface.

Required Register Configurations

Software must program the CSR0, CSR1, CSR2, and CSR3 registers so they reflect the hardware implementation of the LAN interface and operating system. Table–1 provides the configuration requirements for each register.

Table–1 Required LAN Register Configurations

Register	Configuration Requirements
CSR0	Bit 2 (STOP) must never be set to 1 after the Ethernet controller has been initialized. If software must reset or reinitialize the Ethernet controller, then it should use the Ethernet Subsystem Reset bit in the CPU's Diagnostic Control Register (DCR) to reset the Ethernet controller and set the bit 2 (STOP) to 1. Software must assert this Reset bit for a minimum of 200 ns.
CSR1	Bit 0 must be 0 because the initialization block must begin on an even byte boundary.
CSR2	Bits 15–8 must be 0s.
CSR3	Bits 2–0 must be set as follows: **Bit** **Setting** 2 BSWP must be 1 to select byte swapping since the workstation will use big-endian byte ordering. The byte swap feature only swaps Ethernet data bytes and not the initialization block data or descriptor ring entries. 1 ACON must be 0 since the LAN interface uses Address Latch Enable (ALE). 0 BCON must be 0 since the LAN interface uses byte strobes on its buses.

Allocating Memory to the LAN Interface

Since neither the Ethernet controller nor the LAN interface provides local storage for buffering, the Ethernet controller uses main memory as its buffer for Ethernet data. Since the LAN interface does not implement any address mapping and produces only physical addresses, software must ensure that the Ethernet controller does not access areas of memory not intended for it.

Due to the limited addressing capabilities of the Ethernet controller and the value of the LAN interface address extension, the Ethernet controller can only access the first 16 Mbytes of main memory. The combination address produced by the Ethernet controller and the LAN interface's address extension logic provides what can be conceptualized as a Cache/Memory Management Unit (CMMU) page descriptor. Figure 8–2 shows the Sbus addresses produced by the Ethernet controller, Figure 8–3 shows the page descriptor conceptualization, and Table–2 describes this conceptualization.

Since the LAN interface only produces bus transactions that are not snooped by the CMMUs, the main memory assigned to the LAN interace cannot be cached.

31 24	23 16
0 — 0	Ethernet Controller Address (bits 23-16)

15 0
Ethernet Controller Address (bits 15-0)

Figure 8–2 Sbus Addresses Produced by the LAN Interface

31 24	23 16
0 — 0	PFA (bits 23-16)

15 12	11 10	9	8	7	6	5	4	3	2	1	0
PFA (bits 15-12)	Res	WT	SP	G	CI	Res	M	U	WP	Res	V

Figure 8–3 Conceptual CMMU Page Descriptor Produced by the LAN Interface

Table-2 Conceptual CMMU Page Descriptor Produced by the LAN Interface

Bit	Name	Function
31-12	PFA	Page Frame Address. Bits 31-24 are wired to 0; bits 23-12 are the Ethernet controller address bits 23-12.
11-10	Res	Reserved
9	WT	Write through (not implemented).
8	SP	Supervisor protection (not implemented).
7	G	Global (Wired to 0 (equals not global)).
6	CI	Cache inhibit (not implemented).
5	Res	Reserved
4	M	Modified (not implemented).
3	U	Used (not implemented).
2	WP	Write protect. Set to 0 equals writeable.
1	Res	Reserved
0	V	Valid (Set to 1 (equals valid)).

LAN Interface Data Structures

Since the workstation uses the 68000 byte ordering scheme, the LAN interface stores data structures in main memory in the same manner as specified for the Motorola 68000 in Advanced Micro Device's *Am7990 Local Area Network Controller (LANCE) Technical Manual.*

The only restrictions on these data structures are that all data referred to by the LAN interface must be within the first 16 Mbytes of main memory. Software must assign the LAN interface a special fixed and contiguous area in the first 16 Mbytes of main memory and declare this area cache-inhibited during operating system initialization.

Since the CPU communicates with the Ethernet controller through main memory locations, it must also set up an initialization block and two buffer descriptor rings. The CPU sets these up when it executes PROM-based software during powerup, as described in the Chapter 2, "Programming the System Board."

The Ethernet controller uses the initialization block to set internal registers that are not accessible through its two register ports. The two buffer descriptor rings provide the physical address of the memory buffers that the Ethernet controller uses to move data through the LAN interface. One descriptor ring is the data structure that controls the memory buffers for data transmitted over the Ethernet. The other descriptor ring is the data structure that controls the memory buffers for data received from the Ethernet. The Ethernet controller reads and writes to the buffer descriptor rings. An ownership bit in each ring element provides mutual exclusion. For a full description of the CPU-controller protocol and the data structures, refer to Advanced Micro Device's *Am7990 Local Area Network Controller (LANCE) Technical Manual.*

Software Environment

Perform the following tasks to set up the LAN interface to operate properly. Note that none of the following spaces can be cached or swapped.

1. Set aside at least 16 Kbytes in the first 16 Mbytes of main memory for use as the Ethernet buffer space.
2. Set aside a 2–Kbyte region in the first 16 Mbytes of main memory for the Ethernet controller's ring descriptors. The Ethernet controller can deal with two buffer descriptor rings, each consisting of 128 entries. Each buffer descriptor is four half–words long, and must be doubleword aligned (address bits 2–0 are 0) and contiguous with the previous buffer descriptor.
3. Set up the Ethernet controller's initialization block somewhere in the first 16 Mbytes of main memory with the restriction that it is half–word aligned (bit 0 of address is 0). The initialization block contains the address of the two buffer descriptor rings and the Ethernet physical network address. The address of the initialization block is set in the Ethernet controller's CSR1 and CSR2. For more details, see the following section, "Initializing the LAN Interface."
4. Align the buffers pointed to by the buffer descriptor ring entries on half–word boundaries in the first 16 Mbytes of main memory.

Initializing the LAN Interface

Initialize the LAN interface as follows:

1. Clear the Ethernet Subsystem Reset bit to 0 in the CPU Diagnostic Control Register (hardware resets the LAN interface). This clears the CSR0 bit 2 (STOP) to 0.
2. Leave the Ethernet subsystem Reset bit in the CPU Diagnostic Control Register set to 0 for a minimum of 200 ns; software sets this bit to 1.
3. Initialize the following LAN interface data structures in main memory:
    - Transmit descriptor ring
    - Receive descriptor ring
    - Initialization block, containing the following:

      Mode
      Network Physical Address (PADR)
      Logical Network Address Filter (LADRF)
      Receive Descriptor Ring Pointer (RDRP)
      Transmit Descriptor Ring Pointer (TFRP)
4. Load the address of the initialization block into the Ethernet controller's CSR1 and CSR2.
5. Load 000416 into the Ethernet controller's CSR3. This value is required by the workstation.
6. Set the Ethernet controller's CSR0 bit 0 (INIT) to 1.
7. Wait for the Ethernet controller to set CSR0 bit 8 (IDON) to 1.
8. Set the Ethernet controller's CSR0 bit 1 (STRT) bit to 1.

Resetting the LAN Interface

The following events occur when software clears the Ethernet Subsystem Reset bit to 0 in the CPU Diagnostic Control Register for a minimum of 200 ns:

1. The Sbus interface is reset to an idle state, making the LAN interface registers inaccessible.
2. The Ethernet controller is in a hardware reset state so that the following conditions exist:
   - RAP register is cleared.
   - CSR0 bit 2 (STOP) is set to 1 and the other CSR0 bits are cleared to 0s.

After a hardware reset, software must reinitialize the Ethernet controller's CSR1, CSR2, and CSR3, as described in the earlier section, "Initializing the LAN Interface."

LAN Interface Interrupts

The LAN interface generates an interrupt whenever the Ethernet controller generates an interrupt by asserting its interrupt pin. The Ethernet controller can generate an interrupt on the following conditions:

- Transmitter time-out because the packet was too long (BABL).
- Missed packet because the receiver dropped a packet (MISS).
- Memory error because the memory system did not acknowledge within 25.6 ms (MERR).
- Receiver interrupt because a packet was received (RINT).
- Transmitter interrupt because a packet was sent (TINT).
- Initialization was completed (IDON).

When an interrupt occurs, the software must clear the CPU's interrupt logic by clearing the source of the interrupt in the Ethernet controller.

End of Chapter

Chapter 9
Programming the Small Computer System Interface Port

This chapter describes the following topics:

- The SCSI port.
- How to program the protocol controller registers.
- How to reset and initialize the protocol controller.
- Protocol controller interrupts.
- How to program the DMA controller.
- How to manipulate pointers and counters.
- How to implement the selection time-out function.
- DMA controller interrupts.

Overview of the SCSI Port

The Small Computer System Interface (SCSI) is an ANSI defined standard for computer and peripheral interconnection. All the workstation's mass–storage devices are external to the computer and connect to the computer via the SCSI port on the system board. The SCSI port allows the workstation to attach to mass–storage devices and other peripherals.

The workstation's SCSI port supports the following options allowed by the SCSI standard:

- Disconnect/reconnect.
- Single–ended drivers and receivers.
- Asynchronous transfer protocol.
- Shielded connector.
- SCSI bus parity.
- Initiator mode.

Components of the SCSI Port

As shown in Figure 9–1, the SCSI port consists of three major functional components: a SCSI slave interface, a SCSI protocol controller, and a direct–memory access (DMA) controller. This section describes these components briefly.

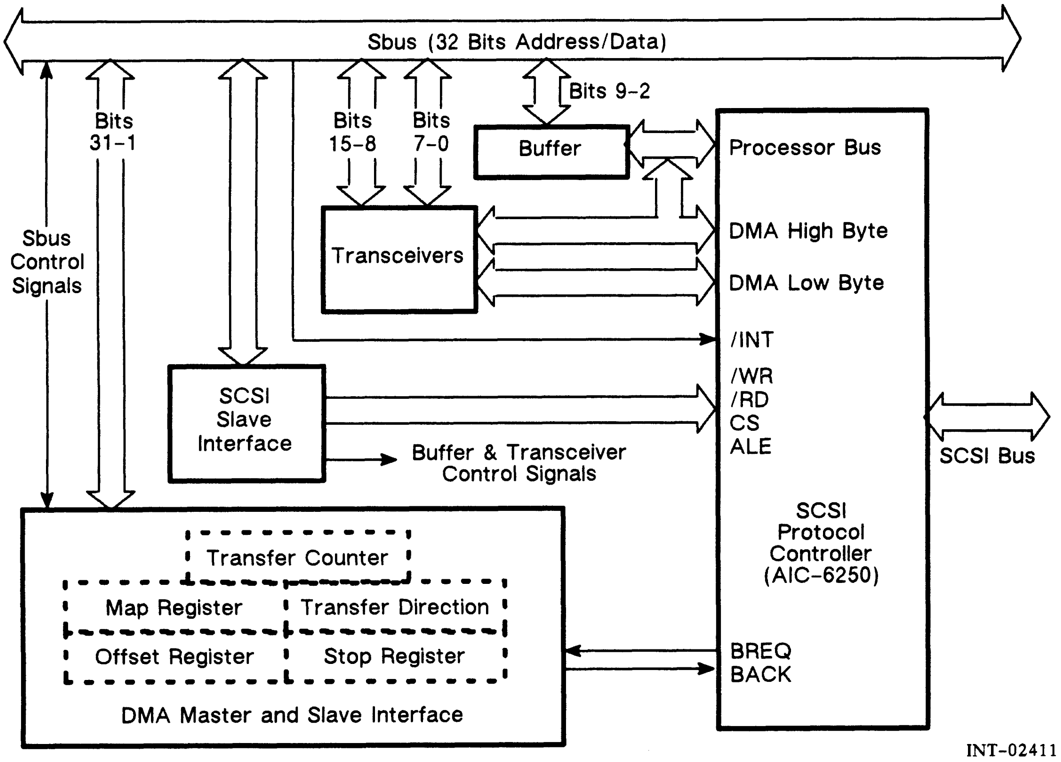


Figure 9–1 SCSI Port Components

SCSI Slave Interface

The SCSI slave interface accepts Sbus signals and generates the necessary signals for slave accesses to the protocol controller.

SCSI Protocol Controller

The SCSI protocol controller is an Adaptec AIC–6250 SCSI Protocol Controller chip that handles all the SCSI bus protocol, as well as providing the interface between the SCSI bus and the host system. The controller contains an 8–byte first–in–first–out (FIFO) buffer, SCSI bus control, a 16–bit DMA data path, an 8–bit microprocessor port, and single–ended drivers.

DMA Controller

The SCSI port provides direct–memory access between the protocol controller and main memory using a discrete DMA controller dedicated to this task. The DMA controller is a one–channel, 16–bit controller consisting of the following registers: DMA Map register, DMA Offset register, DMA Transfer Count register, DMA Transfer Direction register, DMA Stop register.

The DMA slave interface accepts Sbus control signals and produces the necessary signals for slave accesses to the DMA controller registers. The DMA master interface accepts and generates signals between the Sbus and the protocol controller for bus mastership control.

Programming the SCSI Controller

The protocol controller handles operations on the SCSI bus one phase at a time. It supports both the initiator and target roles, and SCSI bus arbitration, as well as automatic response to selection and reselection. Software can set up the protocol controller to look automatically for the SCSI bus free phase and, upon detection of this phase, to arbitrate for control of the SCSI bus. If the protocol controller wins arbitration, it proceeds automatically to the selection or reselection phase. Software can also set up the protocol controller to generate an interrupt when the selection/reselection is completed. Software can then set up the protocol controller to accomplish the desired bus phases one at a time, such as looking for an expected phase and undertaking certain data transfer tasks or else indicating a phase mismatch.

The protocol controller allows three methods of data transfer to and from the host: DMA transfer, automatic programmed I/O (PIO), and microprocessor PIO. The workstation's SCSI port supports each of these methods, although for transfers in the data phase, the DMA transfer method is the method of choice. For a detailed description of these transfer methods, refer to the Adaptec *AIC–6250 High–Performance SCSI Protocol Chip* data sheet.

Software sets up the protocol controller by programming its 16 internal registers that are mapped into the workstation's address space at addresses FFF8 A000 through FFF8 A03C. Table 9–1 gives the memory map for these registers; the rest of the section describes each register briefly. For detailed information on these registers, refer to the Adaptec *AIC–6250 High–Performance SCSI Protocol Chip* data sheet.

NOTE: Before reading the AIC–6250 registers, you must disable parity checking. For information on disabling parity, refer to the *MC88200 User's Manual.*

Table 9–1 Memory Map of the Protocol Controller Registers

Address	Register	Type	Reset State[1]
FFF8 A000	0 — DMA Byte Count (low)	Read/write	Unaffected
FFF8 A004	1 — DMA Byte Count (middle)	Read/write	Unaffected
FFF8 A008	2 — DMA Byte Count (high)	Read/write	Unaffected
FFF8 A00C	3 — Interrupt Mask Register 0	Write	0000 0000 [2]
FFF8 A010	4 — Offset Control	Write	0000 0000 [2]
FFF8 A014	5 — FIFO Status	Read	xx11 0000 #
FFF8 A018	6 — Revision Control Interrupt Mask Register 1	Read Write	0000 0000 [2] 0000 0000 [2]
FFF8 A01C	7 — Status Register 0 Control Register 0	Read Write	See text See text
FFF8 A020	8 — Status Register 1 Control Register 1	Read Write	See text See text
FFF8 A024	9 — SCSI Signal	Read/write	0000 0000 [3]
FFF8 A028	A — SCSI ID	Read/write	Unaffected
FFF8 A02C	B — Source/Destination ID	Read	Unaffected
FFF8 A030	C — Memory Data	Unused	Unaffected
FFF8 A034	D — Port A	Unused	0000 0000 [2]
FFF8 A038	E — Port B	Read/write	0000 0000 [2]
FFF8 A03C	F — SCSI Latch Data SCSI Busy Reset (TGT)	Read Write	Unaffected See text

[1] Contents of the register following a system reset unless otherwise stated.

[2] Contents of the register following a hardware or software SCSI port reset.

[3] Contents of the register following a hardware or software SCSI port reset or a SCSI bus reset.

DMABCL, DMABCM, DMABCH — DMA Byte Count

DMABCL	**Address FFF8 A000**	**Read/Write**
DMABCM	**Address FFF8 A004**	**Read/Write**
DMABCH	**Address FFF8 A008**	**Read/Write**

The DMA Byte Count Low (DMABCL or Register 0), DMA Byte Count Middle (DMABCM or Register 1), and DMA Byte Count High (DMABCH or Register 3) are 8-bit registers that contain, respectively, the low, middle, and high bytes of the DMA count. The DMA count is the number of bytes to be transferred over the SCSI bus during a DMA transfer. These registers are used only when the protocol controller is set up for DMA transfer mode. A reset does not affect these registers, and on powerup their condition is unknown.

IM0 Interrupt Mask 0

Address FFF8 A00C Write Only

The Interrupt Mask register 0 (IM0 or Register 3) enables or disables interrupts that may occur during certain operational conditions. A hardware or software SCSI port reset clears this register to all 0s. A SCSI bus reset clears bit 6 to 0.

OFC Offset Control

Address FFF8 A010 Write Only

The Offset Control register (OFC or Register 4) must contain all 0s to select asynchronous operation for the workstation's SCSI port. A hardware or software SCSI port reset clears this register to all 0s.

FIFOS FIFO Status

Address FFF8 A014 Read/Write

The FIFO Status register (FIFOS or Register 5) contains either FIFO status or DMA control information. When read, it provides the status of the FIFO buffer. When written, it controls the enabling and direction of the DMA transfer. Either a hardware or sofware SCSI port reset or a SCSI bus reset will set this register to xx11 0000.

RC Revision Control
IM1 Interrupt Mask 1

Address FFF8 A018 **Read/Write**

The Revision Control or Interrupt Mask register 1 (Register 6) contains either chip revision or masking information. When read, this register is the Revision Control (RC) register and provides the manufacturer chip revision. When written, this register is Interrupt Mask register 1 (IM1), and it allows software to enable and disable interrupts that may occur during certain operations. A hardware or software SCSI port reset clears this register to all 0s.

SR0 Status 0
CR0 Control 0

Address FFF8 A01C Read/Write

The Status Register 0 or Control Register 0 (Register 7) contains either status or control information for the protocol controller.

When read, this register is Status Register 0 (SR0), and it provides various status information about the protocol controller. The power-on reset leaves bits 1 and 0 unchanged and clears bits 7–2 to 0. A hardware or software SCSI port reset sets bit 7 to 1 and clears bits 5, 3, and 2 to 0. Bit 4 is set to 1 if Register 6 bit 2 (Enable Bus Free Detect Interrupt) is set to 1.

When written, this register is Control Register 0 (CR0), and it allows software to control various functions of the protocol controller. The power-on reset clears all bits except bit 6 to 0. A hardware or software SCSI port reset clears bits 5–3 to 0.

SR1 Status 1
CR1 Control 1

Address FFF8 A020 Read/Write

The Status Register 1 or Control Register 1 (Register 8) contains additional status or control information for the protocol controller.

When read, this register is Status Register 1 (SR1), and it provides additional status for the protocol controller. A hardware or software SCSI bus reset sets bit 6 to 1 and clears bits 5–0 to 0. Bit 7 is normally 1 as determined by the BACK*/BREQ* signal.

When written, this register is Control Register 1 (CR1). The power–on reset clears all bits except bit 5 to 0. A hardware or software SCSI port reset sets bits 7–3 to 1 and clears bit 2 to 0.

SCSIS SCSI Signal

Address FFF8 A024 Read/Write

The SCSI Signal register (SCSIS or Register 9) contains either status or control information for SCSI bus signals. When read, this register provides the status of several of the SCSI bus signals. When written, this register allows software to control these SCSI bus signals. A hardware or software SCSI port reset or a SCSI bus reset clears this register to all 0s. Note that reading this register while it is in transition produces a parity error interrupt.

SCSIID SCSI Identification

Address FFF8 A028 Read/Write

The SCSI ID register (SCSIID or Register A) acts as the SCSI ID register and the SCSI Data register. Prior to arbitration, this register contains the SCSI ID. Upon selection/reselection it is the SCSI Data register. A reset does not affect this register, and on powerup its contents are unknown.

SRDTID Source/Destination Identification

Address FFF8 A02C **Read Only**

The Source/Destination ID register (SRDTID or Register B) contains both the source and destination IDs after the selection/reselection phase. A reset does not affect this register, and on powerup its conditions are unknown.

PTB Port B

Address FFF8 A038 Read/Write

The Port B register (PTB or Register E) is an 8–bit register that contains the upper data byte of a 16–bit data transfer. A hardware or software SCSI port reset clears this register to all 0s.

SCSILD SCSIBR — SCSI Latch Data SCSI Busy Reset

Address FFF8 A03C — Read/Write

The SCSI Latch Data or SCSI Busy Reset register (Register F) contains SCSI information. When read, this register is the SCSI Latch Data (SCSILD) register, and a reset does not affect it. When written, it is the SCSI Busy Reset (SCSIBR) register and it is valid in Target mode only.

Resetting and Initializing the SCSI Controller

The SCSI port is reset when either the system power–up code, other software, or the SCSI bus causes the SCSI port reset signal to go low. A system reset sets the protocol controller's software reset bit (Control Register 1, bit 0) to 1. To complete the reset operation properly, software must clear this bit back to 0.

After the software reset bit is set to 1, software must initialize the protocol controller to set up certain parameters critical for its proper operation. To initialize the protocol controller, perform the following tasks:

1. Set the Clock Frequency Mode (bit 2) to 0.
2. Set up Port B as the upper 8 bits of the 16–bit DMA bus by writing 1 to Register 8 (CR1) bit 6.
3. Set up Port A as an output port by writing 1 to Register 7 (CR0) bit 4.
4. Select single–ended drivers and receivers by writing 0 to Register 7 (CR0) bit 3.
5. Select asynchronous mode of operation by writing 0s to Register 4 (OFC) bits 3–0.

SCSI Controller Interrupts

The protocol controller can generate an interrupt for any of the following conditions:

- Successful selection.
- Successful reselection.
- Successful arbitration.
- Command complete.
- Error (including a reset).
- Phase change.
- SCSI bus parity error.
- Bus free phase detection.
- Phase mismatch.
- DMA data parity error.
- SCSI reset.

Software can enable or disable these interrupts using the protocol controller's Interrupt Mask registers. For more information, refer to the Adaptec *AIC–6250 High–Performance SCSI Protocol Chip* data sheet.

Programming the DMA Controller

The protocol controller requests DMA service by generating a system SCSI bus request. This request occurs if either the last transfer did not cross a page boundary or the terminal transfer count has not been reached. After the DMA controller receives and acknowledges its control of the Sbus, it proceeds with a transfer if the following conditions are met:

- The Map register is valid.
- The transfer will not write to write-protected memory.
- The Mbus retry signal is not active.

If any of these conditions exist, the DMA controller relinquishes bus mastership. If either the Map register contents are invalid or a write protect error occurs, the SCSI port generates an interrupt to indicate the error condition. The DMA controller will not arbitrate for bus mastership until these error conditions are cleared. If the retry signal caused the DMA controller to relinquish bus mastership, the DMA controller will re-arbitrate for bus mastership on the next clock cycle.

Once the DMA controller receives bus mastership, the Map register generates the appropriate page address on Sbus bits 31–12 and the Offset register generates the lower 11 address bits on Sbus bits 11–1. Sbus bit 0 is not driven since half-word (16-bit) transfers are the only type supported for SCSI DMA accesses. Since the half-word transfer must be on an even boundary, Sbus bit 0 is always undefined during a read of this register.

During the data phase, data is driven onto the Sbus, the Offset register is incremented, and the Transfer Count register is decremented. The DMA controller transfers only one half-word per request.

Since the DMA controller provides 32 address bits, software can place SCSI data anywhere within the CPU's 4-Gbyte address range. The DMA Map and Offset registers segment memory into 4-Kbyte pages. The upper 20 address bits specify the page within memory and the lower 12 bits provide the offset within the page.

During DMA transfers, the Sbus interface maps Sbus bits into the protocol controller so that it will store bytes in main memory in a big-endian format.

Software programs the DMA controller by setting up its registers. These registers are mapped to address space FFF8 B000 through FFF8 B018, as shown in Table 9–2.

NOTE: Software must mask any bits not listed in the following register descriptions (their state is unknown).

Table 9–2 Memory Map of the DMA Controller Registers

Address	Register	Type
FFF8 B000	DMA Map Register	Read/write
FFF8 B004	DMA Offset Register	Read/write
FFF8 B008	DMA Transfer Count Register	Read/write
FFF8 B00C	DMA Transfer Direction Register	Write
FFF8 B010	Clear DMA Map Valid Error Interrupt	Write
FFF8 B014	Clear DMA write Protect Error Interrupt	Write
FFF8 B018	DMA Stop Register	Write

DMAMAP DMA Map

Address FFF8 B000 Read/Write

The DMA Map (DMAMAP) register contains DMA mapping information.

31 16
PFA

15 12	11 8	7	6 3	2	1	0
PFA	X	G	X	W	X	V

Bit	Name	Function
31-12	PFA	Page Frame Address Physical address of the memory page.
11-8	X	Do not care when written to, and undefined when read.
7	G	Global When set to 1, the memory mapped by this DMA map register is global memory. This bit is driven onto the Mbus signal C5; other caches sample this signal when snooping the bus for cache coherency. This bit is not driven onto the Sbus when this register is read.
6-3	X	Do not care when written to, and undefined when read.
2	W	Write Protect When set to 1, the memory mapped by this register is write protected. When cleared, a DMA device can write to main memory. A DMA device attempting to write to write-protected memory generates an interrupt and aborts the cycle before memory is corrupted. This bit is not driven onto the Sbus when this register is read.
1	X	Do not care when written to, and undefined when read.
0	V	Valid When set to 1, the contents of the DMA Map register are valid. This bit must be set when software loads the register with a valid descriptor. Invalid DMA Map register contents prevent a DMA cycle from occurring and generate an interrupt. This bit is not driven onto the Sbus when this register is read.

DMAOFF DMA Offset

Address FFF8 B004 Read/Write

The DMA Offset (DMAOFF) register contains the DMA address offset.

15 … 1	0
AO	X

Bit	Name	Function
11–1	AO	Address Offset Specifies the address offset within the 4–Kbyte page specified by the DMA Map register. When these bits are written, they receive memory address bits 11–1; when they are read, they provide address bits 11–1. Every time a half–word is transferred during a DMA operation, this register is incremented by 1 automatically. When the contents of this register roll over from 0FFF to 1000, an interrupt is generated to indicate a page boundary has been crossed. Sbus bit 12 cannot be read during a slave access of this register; thus a value of 000 (or 001 since bit 0 is undefined) is read when a page boundary has been crossed.
0	X	Undefined - Ignore Because DMA transfers occur on half–word boundaries only, this bit is meaningless, and thus ignored. Since this bit is undefined when the register is read, software must mask it off when reading the register.

DMATC DMA Transfer Count

Address FFF8 B008 Read/Write

The DMA Transfer Count (DMATC) register contains the DMA transfer count.

12 1	0
TR	X

Bit	Name	Function
12-1	TR	Transfer Count Specifies the number of bytes to be transferred (up to 1000). When written these bits reflect Sbus bits 12-1. The maximum transfer count supported by the hardware is 4 Kbytes. Since Sbus bit 0 is ignored, an even number of bytes must be specified.
0	X	Undefined - Ignore This bit is ignored, and thus an even number of bytes is specified for the transfer count. Since this bit is undefined when the register is read, software must mask it off. Sbus bit 0 is ignored, thus, the transfer count must be a number of bytes.

DMATD DMA Transfer Direction

Address FFF8 B00C Write Only

The DMA Transfer Direction (DMATD) register indicates the direction for a DMA transfer. Writing a 1 to bit 0 of this register sets up the DMA transfer to read from memory to the protocol controller. Writing a 0 to bit 0 of this register sets up the DMA transfer to write to memory from the protocol controller. Software must program this register every time it programs the DMA registers for a transaction.

DMACMV Clear DMA Map Valid Interrupt

Address FFF8 B010 Write Only

The Clear DMA Map Valid Interrupt (DMACMV) register clears DMA Valid interrupts. If this interrupt stopped a DMA transfer, the transfer restarts when this register is cleared. Because the condition of the DMA Valid bit in the DMA Map register caused the initial fault, software must rewrite the DMA Map register before clearing this interrupt so that the DMA Valid bit is not still set when the DMA transfer restarts.

DMACWP Clear DMA Write Protect Interrupt

Address FFF8 B014 Write Only

The Clear DMA Write Protect Interrupt (DMACWP) register clears DMA write protect interrupts. If this interrupt stopped a DMA transfer, the transfer restarts when this register is cleared. Because the condition of the write-protect bit in the DMA Map register caused the initial fault, software must rewrite the DMA Map register before clearing the interrupt so that the DMA Write-Protect bit is not still set when the DMA transfer restarts.

DMASTP — DMA Stop

Address FFF8 B018 — Write Only

The DMA Stop (DMASTP) register restarts or inhibits DMA transfers. Writing 0016 to this register restarts a DMA transfer that was stopped because of a reset, a page boundary crossing, or a terminal count condition. The transfer restarts only if the page boundary and terminal count conditions are cleared first. For this reason, software should program this DMA register last. Writing 0116 to this register prevents software from starting a DMA transfer.

Manipulating Pointers and Counters

Because memory is segmented into 4-Kbyte pages, software must set up the DMA registers every time a page boundary is encountered. When a page boundary is encountered, the Offset register generates an interrupt, causing the DMA Stop register to produce a stop condition that masks the bus request to the system. Software must keep track of the number of bytes actually transferred to/from memory. It can do this by storing the initial programmed transfer count, incrementing a register for each page boundary crossing, and reading the Transfer Count register for the number of bytes transferred in the current page. This number is important for resetting pointers to memory data if a SCSI bus error occurs.

Software must always maintain a way to recover from a disconnect or an error by readjusting pointers to main memory to reprogram the DMA controller registers. If a disconnect occurs while transferring data from the SCSI bus to main memory, the DMA controller's FIFO buffer could contain some valid data. The DMA controller continues to transfer this data to main memory even though a phase mismatch interrupt occurred. If the disconnect occurs while transferring data from main memory to SCSI, the data remaining in the FIFO buffer is lost, but the exact number of bytes transferred across the SCSI bus is known, so the pointer to main memory can be set back. Software can accomplish this using the protocol controllers DMA registers, the DMA controller's Transfer Count register, and any other registers that software set up.

NOTE: The protocol controller does not give the system any indication when only 1 byte is present in its FIFO buffer during DMA transfer mode. For this reason, only transfers with an even number of bytes are supported. Further, all peripheral devices will disconnect after an even number of bytes have been transferred across the SCSI bus. If a phase has an odd number of bytes of information, that phase must be handled using the protocol controller's automatic PIO mode.

Implementing a Selection Time-Out Function

Software must use the programmable interval timer (PIT) in the system control logic to provide a selection time-out function for the SCSI port. Each time the protocol controller generates a successful arbitration interrupt, software must set an event to go off in approximately 250 ms, the ANSI recommended time-out period. If a successful selection interrupt occurs before this time-out period, the time-out event must be removed. Software should turn off the time-out process when responding to a successful selection.

DMA Controller Interrupts

The DMA controller generates three types of interrupts: a transfer count or offset interrupt, a write–protect interrupt, and a Map register Valid Error interrupt. You can program the DMA controller to generate:

- A transfer count or offset interrupt when either of the following conditions occurs:

  The DMA Transfer Count register reaches the terminal count. Writing any data except a 0 or a 1 to the DMA Transfer Count register clears the interrupt from this source. On reset, the DMA Transfer Count register also generates an interrupt.

  The DMA Offset register reaches terminal count indicating a page boundary has been crossed. Writing any data to the DMA Offset register clears the interrupt condition from this source.
- A write–protect interrupt when the Write Protect bit of its DMA Map register is 1 and the DMA controller tries to write to main memory.
- A map register interrupt when the Valid bit of its DMA Map register is 0 and a DMA transfer is attempted.

End of Chapter

Appendix A
Address Map

The following are maps of the workstation address space, beginning at 0000 0000. The maps include system memory, boot PROM, video memory, and all workstation registers (excluding registers within the CPU). Table A–1 is the address map for 300 series stations; Table A–2 is the map for 400 series stations.

Table A–1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
0000 0000 - 07FF FFFF	System Memory (128 Mbytes)		Read/write
0000 0000 - 0001 FFFF	Boot PROM during boot (128 Kbytes)		Read
8000 0000 - 87FF FFFF	Video Memory (128 Mbytes)		
8000 0000 - 83FF FFFF	Color Memory		
8400 0000 - 87FF FFFF	Overlay/Window ID Memory		
FFC0 0000 - FFC1 FFFF	Boot PROM after boot (128 Kbytes)		
FFF0 0000 - FFF0 1883	CMMU Registers (For information on the CMMU registers, see the *MC88200 User's Manual.*)		
CMMU0 (Data CMMU) Registers			
System Control Registers			
FFF0 0000	CMMU ID	IDR	Read
FFF0 0004	System Command	SCR	Write
FFF0 0008	System Status	SSR	Read
FFF0 000C	System Address	SAOR	Read/Write
FFF0 0104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 0108	Pbus Status	PFSR	Read
FFF0 010C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 0200	Supervisor Area	SAPR	Read/Write
FFF0 0204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 0400	Port 0	BWP0	Write
FFF0 0404	Port 1	BWP1	Write
FFF0 0408	Port 2	BWP2	Write
FFF0 040C	Port 3	BWP3	Write
FFF0 0410	Port 4	BWP4	Write
FFF0 0414	Port 5	BWP5	Write
FFF0 0418	Port 6	BWP6	Write
FFF0 041C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 0800	Data Port 0	CDP0	Read
FFF0 0804	Data Port 1	CDP1	Read
FFF0 0808	Data Port 2	CDP2	Read
FFF0 080C	Data Port 3	CDP3	Read
FFF0 0840	Tag Port 0	CTP0	Read
FFF0 0844	Tag Port 1	CTP1	Read
FFF0 0848	Tag Port 2	CTP2	Read
FFF0 084C	Tag Port 3	CTP3	Read
FFF0 0880	Set Status	CSSP	Read

(continued)

Table A-1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF0 0000 - FFF0 1883	CMMU Registers (continued)		
CMMU1 (Instruction CMMU) Registers			
System Control Registers			
FFF0 1000	CMMU ID	IDR	Read
FFF0 1004	System Command	SCR	Write
FFF0 1008	System Status	SSR	Read
FFF0 100C	System Address	SAOR	Read/Write
FFF0 1104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 1108	Pbus Status	PFSR	Read
FFF0 110C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 1200	Supervisor Area	SAPR	Read/Write
FFF0 1204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 1400	Port 0	BWP0	Write
FFF0 1404	Port 1	BWP1	Write
FFF0 1408	Port 2	BWP2	Write
FFF0 140C	Port 3	BWP3	Write
FFF0 1410	Port 4	BWP4	Write
FFF0 1414	Port 5	BWP5	Write
FFF0 1418	Port 6	BWP6	Write
FFF0 141C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 1800	Data Port 0	CDP0	Read
FFF0 1804	Data Port 1	CDP1	Read
FFF0 1808	Data Port 2	CDP2	Read
FFF0 180C	Data Port 3	CDP3	Read
FFF0 1840	Tag Port 0	CTP0	Read
FFF0 1844	Tag Port 1	CTP1	Read
FFF0 1848	Tag Port 2	CTP2	Read
FFF0 184C	Tag Port 3	CTP3	Read
FFF0 1880	Set Status	CSSP	Read
FFF8 0000 - FFF8 1FFF	BBSRAM/RTC		
FFF8 0000 - FFF8 1FDC	NOVRAM Bytes 0000 - 2039		
FFF8 1FE0 - FFF8 1FFC	Time-of-Boot (TOB) Clock Registers		
FFF8 1FEO	Control		Read/Write
FFF8 1FE4	Seconds		Read/Write
FFF8 1FE8	Minutes		Read/Write
FFF8 1FEC	Hour		Read/Write
FFF8 1FF0	Day		Read/Write
FFF8 1FF4	Date		Read/Write
FFF8 1FF8	Month		Read/Write
FFF8 1FFC	Year		Read/Write

(continued)

Table A-1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 2000 - FFF8 2FFF	I/O Registers		
DUART1 Serial & Parallel Port Registers			
FFF8 2000	Mode Registers A	MR1A, MR2A	Read/Write
FFF8 2004	Status Register A	SRA	Read
	Clock Select Register A	CSRA	Write
FFF8 2008	RESERVED		Read !
	Command Register A	CRA	Write
FFF8 200C	Receive Holding Register A	RHRA	Read
	Transmit Holding Register A	THRA	Write
FFF8 2010	Input Port Change	IPCR	Read
	Auxiliary Control	ACR	Write
FFF8 2014	Interrupt Status	ISR	Read
	Interrupt Mask	IMR	Write
FFF8 2018	Counter/Timer Upper Register	CTUR	Read/Write
FFF8 201C	Counter/Timer Lower Register	CTLR	Read/Write
FFF8 2020	Mode Registers B	MR1B, MR2B	Read/Write
FFF8 2024	Status Register B	SRB	Read
	Clock Select Register B	CSRB	Write
FFF8 2028	RESERVED		Read !
	Command Register B	CRB	Write
FFF8 202C	Receive Holding Register B	RHRB	Read
	Transmit Holding Register B	THRB	Write
FFF8 2030	RESERVED		Read/Write
FFF8 2034	Input Port Configuration Register	IPCR	Read
	Output Port Configuration Register	OPCR	Write
FFF8 2038	Start Counter Command Register	SRCC	Read
	Set Output Port Bits Command Register	SOPBC	Write
FFF8 203C	Stop Counter Command Register	STCC	Read
	Reset Output Port Bits Register	ROPBC	Write
FFF8 2400	Parallel Port Data Register	PPD	Write
	Parallel Port Status Register	PPS	Read
Keyboard Port Registers			
FFF8 2800	Receive Holding Register	RxH	Read
	Transmit Holding Register	TxH	Write
FFF8 2804	Status Register	STS	Read
FFF8 2808	Mode Register 1	MD1	Read/Write
	Mode Register 2	MD2	Read/Write
FFF8 280C	Command Register	CMD	Read/Write
FFF8 2810	Disable Clock Register	DSC	Write
FFF8 2820	Enable Transmit Clock	ENABLE_TXC	Write
FFF8 3000 - FFF8 3FFF	CIO Registers		
FFF8 3000	Port A Data register		
FFF8 3004	Port B Data register		
FFF8 3008	Port C Data register		
FFF8 300C	Control register		
FFF8 4000 - FFF8 4FFF	System Control Logic Registers		
FFF8 4000	Interrupt Status	ISR	Read/Write
FFF8 4004	Interrupt Enable	IER	Write
FFF8 4008	Diagnostic Control	DCR	Write
FFF8 400C	Diagnostic Status	DSR	Read
FFF8 4010	Parity Address	PAR	Read
FFF8 4014	Soft Interrupt	SWIR	Write

! Do not read these registers. Reading address FFF8 202816 may hang the asynchronous line. (If you use this line for the system console, the workstation may hang and require a reset.)

(continued)

Table A-1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 9000 - FFF8 90FF	Color Graphics Subsystem Registers — Broadcast.		
FFF8 9000	Control and Status Register 0	CSR0	Read/Write
FFF8 9004	Stop/Resume Register	STOP	Read/Write
FFF8 9008	Control and Status Register 1	CSR1	Read/Write
FFF8 900C	Command Register	CMD	Read/Write
FFF8 9010	Plane Mask Register	MASK	Read/Write
FFF8 9014	Background Color Register	BACK	Read/Write
FFF8 9018	Line Pattern Register	LPAT	Read/Write
FFF8 901C	Pattern Control/WID Register	PC/WID	Read/Write
FFF8 9020	CRT Control Register 0	CRT0	Read/Write
FFF8 9024	CRT Control Register 1	CRT1	Read/Write
FFF8 9028	CRT Control Register 2	CRT2	Read/Write
FFF8 902C	Reserved		
FFF8 9030	Internal State 0	STATE0	Read
FFF8 9034	Internal State 1	STATE1	Read
FFF8 9038 - FFF8 903C	Reserved		
FFF8 9040	Parameter Register 0	PARM0	Read/Write
FFF8 9044	Parameter Register 1	PARM1	Read/Write
FFF8 9048	Parameter Register 2	PARM2	Read/Write
FFF8 904C	Parameter Register 3	PARM3	Read/Write
FFF8 9050	Parameter Register 4	PARM4	Read/Write
FFF8 9054	Parameter Register 5	PARM5	Read/Write
FFF8 9058	Parameter Register 6	PARM6	Read/Write
FFF8 905C	Parameter Register 7	PARM7	Read/Write
FFF8 9060	Parameter Register 8	PARM8	Read/Write
FFF8 9064	Parameter Register 9	PARM9	Read/Write
FFF8 9068	Parameter Register 10	PARM10	Read/Write
FFF8 906C	Parameter Register 11	PARM11	Read/Write
FFF8 9070	Parameter Register 12	PARM12	Read/Write
FFF8 9074	Parameter Register 13	PARM13	Read/Write
FFF8 9078	Parameter Register 14	PARM14	Read/Write
FFF8 907C	Parameter Register 15	PARM15	Read/Write
FFF8 9080 - FFF8 909C	Reserved		
FFF8 90A0	Data Port Register	DATA	Read/Write
FFF8 90A4	Palette Pointer 0	PLT0	Read/Write
FFF8 90A8	Palette Pointer 1	PLT1	Read/Write
FFF8 90AC	Blink Control Register	BLNK	Read/Write
FFF8 90B0 - FFF8 90BC	Reserved		
FFF8 90C0	RAMDAC Address Register	DAC0	Read/Write
FFF8 90C4	RAMDAC Color Palette RAM	DAC1	Read/Write
FFF8 90C8	RAMDAC Control Register	DAC2	Read/Write
FFF8 90CC	RAMDAC Overlay Palette RAM	DAC3	Read/Write
FFF8 90D0 - FFF8 90FF	Reserved		

(continued)

Table A–1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 9000 – FFF8 90FF	Monochrome Graphics Subsystem Registers		
FFF8 9000	Execution Address Origin (bits 15–0)	EADORG	Read/Write
FFF8 9004	Execution Address Origin (bits 7–0) Dot Address Origin (bits 11–8)	EADORG DADORG	Read/Write Read/Write
FFF8 9008	Execution Address 1 (bits 15–0)	EAD1	Read/Write
FFF8 900C	Execution Address 1 (bits 7–0) Dot Address 1 (bits 11–8)	EAD1 DAD1	Read/Write Read/Write
FFF8 9010	Execution Address 2 (bits 15–0)	EAD2	Read/Write
FFF8 9014	Execution Address 2 (bits 7–0) Dot Address 2 (bits 11–8)	EAD2 DAD2	Read/Write Read/Write
FFF8 9018	Plane Displacement Source (bits 15–0)	PDISPS	Read/Write
FFF8 901C	Plane Displacement Source (bits 7–0)	PDISPS	Read/Write
FFF8 9020	Plane Displacement Dest. (bits 15–0)	PDISPD	Read/Write
FFF8 9024	Plane Displacement Dest. (bits 7–0)	PDISPD	Read/Write
FFF8 9028	Plane Maximum	PMAX	Read/Write
FFF8 902C	Drawing Mode 0 (bits 3–0) Drawing Mode 1 (bits 7–4)	MOD0 MOD1	Read/Write Read/Write
FFF8 9030	Pattern Pointer (bits 15–0)	PTPN	Read/Write
FFF8 9034	Pattern Pointer (bits 7–0)	PTPN	Read/Write
FFF8 9038	Stack Pointer (bits 15–0)	STACK	Read/Write
FFF8 903C	Stack Pointer (bits 7–0)	STACK	Read/Write
FFF8 9040 – FFF8 9076	Not available.		
FFF8 9078	Control (bits 15–8) Bank (bits 7–0) Status (bits 15–0)	CTRL BANK STATUS	Write Write Read
FFF8 907C	PUT/GET Port	PGPORT	
CAUTION: Do NOT read from or write to the PGPORT register; the SCSI bus will hang!			
FFF8 9080	X		Read/Write
FFF8 9084	Y		Read/Write
FFF8 9088	DX		Read/Write
FFF8 908C	DY		Read/Write
FFF8 9090	XS		Read/Write
FFF8 9094	YS		Read/Write
FFF8 9098	XE		Read/Write
FFF8 909C	YE		Read/Write

(continued)

Table A-1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 9000 - FFF8 90FF	Monochrome Graphics Subsystem Registers (continued)		
FFF8 90A0	XC		Read/Write
FFF8 90A4	YC		Read/Write
FFF8 90A8	DH		Read/Write
FFF8 90AC	DV		Read/Write
FFF8 90B0	Pitch Source	PITCHS	Read/Write
FFF8 90B4	Pitch Destination	PITCHD	Read/Write
FFF8 90B8	Stack Maximum	STMAX	Read/Write
FFF8 90BC	Plane Select	PLANES	Read/Write
FFF8 90C0	Pattern Count	PTNCNT	Read/Write
FFF8 90C4	X Clipping Minimum	XCLMIN	Read/Write
FFF8 90C8	Y Clipping Minimum	YCLMIN	Read/Write
FFF8 90CC	X Clipping Maximum	XCLMAX	Read/Write
FFF8 90D0	Y Clipping Maximum	YCLMAX	Read/Write
FFF8 90D4 - FFF8 90D6	Not available.		
FFF8 90D8	Clipping Mode (bits 9-8) Magnifier Horizontal (bits 7-4) Magnifier Vertical (bits 3-0)	CLIP MAGH MAGV	Read/Write Read/Write Read/Write
FFF8 90DC	Command		Read/Write
FFF8 90E0	Display Control	DISP_CTRL	Write
FFF8 90E4	Address Control (bits 14-12) Display Pitch (bits 11-0)	AC DISP_PCH	Write Write
FFF8 90E8	Display Address (bits 15-0)	DAD	Write
FFF8 90EC	Display Address (bits 7-0) Word Count (bits 15-8)	DAD WC	Write Write
FFF8 90F0	Cursor Mode Select (bit 15) Cursor Enable (bit 14) Cursor X Coordinate (bits 11-0)	CRS CE GCSRX	Write Write Write
FFF8 90F4	Cursor Y Start	GCSRYS	Write
FFF8 90F8	Cursor Y End (bits 11-0) Word Count (bits 15-12)	GCSRYE WC	Write Write
FFF8 90FC	Horizontal Sync Horizontal Back Porch Half Horizontal Time Horizontal Display Period Horizontal Front Porch Vertical Sync Vertical Back Porch Lines per Field Vertical Front Porch	HS HBP HH HD HFP VS VBP L/F VFP	Write Write Write Write Write Write Write Write Write

CAUTION: *Only write to HS, HBP, HH, HD, HFP, VS, VBP, L/F, and VFP if you are resetting the workstation; do not write to them at any other time!*

NOTE: Since HS, HBP, HH, HD, HFP, VS, VBP, L/F, and VFP are all located at address FFF8 90FC, you must write to them in the listed order.

(continued)

Table A-1 300 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 A000 - FFF8 AFFF	SCSI Port Registers		
FFF8 A000	0 - DMA byte count (low)	DMABCL	Read/Write
FFF8 A004	1 - DMA byte count (middle)	DMABCM	Read/Write
FFF8 A008	2 - DMA byte count (high)	DMABCH	Read/Write
FFF8 A00C	3 - Interrupt Mask Register 0	IM0	Write
FFF8 A010	4 - Offset Control Register	OFC	Write
FFF8 A014	5 - FIFO Status Register	FIFOS	Read
FFF8 A018	6 - Revision Control Interrupt Mask Register 1	RC IM1	Read Write
FFF8 A01C	7 - Status Register 0 Control Register 0	SR0 CR0	Read Write
FFF8 A020	8 - Status Register 1 Control Register 1	SR1 CR1	Read Write
FFF8 A024	9 - SCSI Signal Register	SCSIS	Read/Write
FFF8 A028	A - SCSI ID Register	SCSIID	Read/Write
FFF8 A02C	B - Source/Dest ID	SRDTID	Read
FFF8 A030	C - Memory Data Register	MMD	Read/Write
FFF8 A034	D - Port A Register	PTA	Read/Write
FFF8 A038	E - Port B Register	PTB	Read/Write
FFF8 A03C	F - SCSI Latch Data SCSI BSY RST (TGT)	SCSILD SCSIBR	Read Write
FFF8 B000 - FFF8 BFFF	DMA Registers		
FFF8 B000	Map Register	DMAMAP	Read/Write
FFF8 B004	Offset Register	DMAOFF	Read/Write
FFF8 B008	Transfer Count Register	DMATC	Read/Write
FFF8 B00C	Transfer Direction Register	DMATD	Write
FFF8 B010	Clear DMA Map Valid Error Interrupt	DMACMV	Write
FFF8 B014	Clear DMA Write Protect Error Interrupt	DMACWP	Write
FFF8 B018	DMA Stop Register	DMASTP	Write
FFF8 C000 - FFF8 CFFF	LAN Interface Registers		
FFF8 C000	Register Data Pointer	RDP	Read/Write
FFF8 C004	Register Address Pointer	RAP	Read/Write

(concluded)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
0000 0000 - 07FF FFFF	System Memory (128 Mbytes)		Read/write
0000 0000 - 0001 FFFF	Boot PROM During Boot (128 Kbytes)		Read
1000 0000 - 7FFF FFFF	VME A32 space (1.75 Gbytes)		Read/write
8000 0000 - 87FF FFFF	Video Memory (256 Mbytes)		Read/write
8000 0000 - 83FF FFFF	Color Memory		
8400 0000 - 87FF FFFF	Overlay/Window ID Memory		
8800 0000 - 8FFF FFFF	Z-buffer memory		Read/write
9000 0000 - FDFF FFFF	VME A32 Space		Read/write
FE00 0000 - FEFF FFFF	VME A24 Space (16 Mbytes)		Read/write
FFC0 0000 - FFC1 FFFF	Boot PROM After Boot (128 Kbytes)		
FFE0 0000 - FFE1 FFFF	SRAM		
FFF0 0000 - FFF0 3883	CMMU Registers (For information on the CMMU registers, see the *MC88200 User's Manual.*)		
CMMU0 (CPU0, Data CMMU) Registers			
System Control Registers			
FFF0 0000	CMMU ID	IDR	Read
FFF0 0004	System Command	SCR	Write
FFF0 0008	System Status	SSR	Read
FFF0 000C	System Address	SAOR	Read/Write
FFF0 0104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 0108	Pbus Status	PFSR	Read
FFF0 010C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 0200	Supervisor Area	SAPR	Read/Write
FFF0 0204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 0400	Port 0	BWP0	Write
FFF0 0404	Port 1	BWP1	Write
FFF0 0408	Port 2	BWP2	Write
FFF0 040C	Port 3	BWP3	Write
FFF0 0410	Port 4	BWP4	Write
FFF0 0414	Port 5	BWP5	Write
FFF0 0418	Port 6	BWP6	Write
FFF0 041C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 0800	Data Port 0	CDP0	Read
FFF0 0804	Data Port 1	CDP1	Read
FFF0 0808	Data Port 2	CDP2	Read
FFF0 080C	Data Port 3	CDP3	Read
FFF0 0840	Tag Port 0	CTP0	Read
FFF0 0844	Tag Port 1	CTP1	Read
FFF0 0848	Tag Port 2	CTP2	Read
FFF0 084C	Tag Port 3	CTP3	Read
FFF0 0880	Set Status	CSSP	Read

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF0 0000 - FFF0 3883	CMMU Registers (continued)		
CMMU1 (CPU0, Instruction CMMU) Registers			
System Control Registers			
FFF0 1000	CMMU ID	IDR	Read
FFF0 1004	System Command	SCR	Write
FFF0 1008	System Status	SSR	Read
FFF0 100C	System Address	SAOR	Read/Write
FFF0 1104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 1108	Pbus Status	PFSR	Read
FFF0 110C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 1200	Supervisor Area	SAPR	Read/Write
FFF0 1204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 1400	Port 0	BWP0	Write
FFF0 1404	Port 1	BWP1	Write
FFF0 1408	Port 2	BWP2	Write
FFF0 140C	Port 3	BWP3	Write
FFF0 1410	Port 4	BWP4	Write
FFF0 1414	Port 5	BWP5	Write
FFF0 1418	Port 6	BWP6	Write
FFF0 141C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 1800	Data Port 0	CDP0	Read
FFF0 1804	Data Port 1	CDP1	Read
FFF0 1808	Data Port 2	CDP2	Read
FFF0 180C	Data Port 3	CDP3	Read
FFF0 1840	Tag Port 0	CTP0	Read
FFF0 1844	Tag Port 1	CTP1	Read
FFF0 1848	Tag Port 2	CTP2	Read
FFF0 184C	Tag Port 3	CTP3	Read
FFF0 1880	Set Status	CSSP	Read

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF0 0000 - FFF0 3883	CMMU Registers (continued)		
CMMU2 (CPU1, Data CMMU) Registers			
System Control Registers			
FFF0 2000	CMMU ID	IDR	Read
FFF0 2004	System Command	SCR	Write
FFF0 2008	System Status	SSR	Read
FFF0 200C	System Address	SAOR	Read/Write
FFF0 2104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 2108	Pbus Status	PFSR	Read
FFF0 210C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 2200	Supervisor Area	SAPR	Read/Write
FFF0 2204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 2400	Port 0	BWP0	Write
FFF0 2404	Port 1	BWP1	Write
FFF0 2408	Port 2	BWP2	Write
FFF0 240C	Port 3	BWP3	Write
FFF0 2410	Port 4	BWP4	Write
FFF0 2414	Port 5	BWP5	Write
FFF0 2418	Port 6	BWP6	Write
FFF0 241C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 2800	Data Port 0	CDP0	Read
FFF0 2804	Data Port 1	CDP1	Read
FFF0 2808	Data Port 2	CDP2	Read
FFF0 280C	Data Port 3	CDP3	Read
FFF0 2840	Tag Port 0	CTP0	Read
FFF0 2844	Tag Port 1	CTP1	Read
FFF0 2848	Tag Port 2	CTP2	Read
FFF0 284C	Tag Port 3	CTP3	Read
FFF0 2880	Set Status	CSSP	Read

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF0 0000 - FFF0 3883	CMMU Registers (continued)		
CMMU3 (CPU1, Instruction CMMU) Registers			
System Control Registers			
FFF0 3000	CMMU ID	IDR	Read
FFF0 3004	System Command	SCR	Write
FFF0 3008	System Status	SSR	Read
FFF0 300C	System Address	SAOR	Read/Write
FFF0 3104	System Control	SCTR	Read/Write
Pbus Fault Registers			
FFF0 3108	Pbus Status	PFSR	Read
FFF0 310C	Pbus Address	PFAR	Read
Area Pointers			
FFF0 3200	Supervisor Area	SAPR	Read/Write
FFF0 3204	User Area	UAPR	Read/Write
BATC Write Pointers			
FFF0 3400	Port 0	BWP0	Write
FFF0 3404	Port 1	BWP1	Write
FFF0 3408	Port 2	BWP2	Write
FFF0 340C	Port 3	BWP3	Write
FFF0 3410	Port 4	BWP4	Write
FFF0 3414	Port 5	BWP5	Write
FFF0 3418	Port 6	BWP6	Write
FFF0 341C	Port 7	BWP7	Write
Cache Diagnostic Ports			
FFF0 3800	Data Port 0	CDP0	Read
FFF0 3804	Data Port 1	CDP1	Read
FFF0 3808	Data Port 2	CDP2	Read
FFF0 380C	Data Port 3	CDP3	Read
FFF0 3840	Tag Port 0	CTP0	Read
FFF0 3844	Tag Port 1	CTP1	Read
FFF0 3848	Tag Port 2	CTP2	Read
FFF0 384C	Tag Port 3	CTP3	Read
FFF0 3880	Set Status	CSSP	Read
FFF8 0000 - FFF8 1FFF	BBSRAM/RTC		
FFF8 0000 - FFF8 1FDC	NOVRAM Bytes 0000 - 2039		
FFF8 1FE0 - FFF8 1FFC	Time-of-Boot (TOB) Clock Registers		
FFF8 1FEO	Control		Read/Write
FFF8 1FE4	Seconds		Read/Write
FFF8 1FE8	Minutes		Read/Write
FFF8 1FEC	Hour		Read/Write
FFF8 1FF0	Day		Read/Write
FFF8 1FF4	Date		Read/Write
FFF8 1FF8	Month		Read/Write
FFF8 1FFC	Year		Read/Write

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 2000 - FFF8 2FFF	I/O Registers		
Serial Port Registers — DUART1			
FFF8 2000	Mode Registers A	MR1A, MR2A	Read/Write
FFF8 2004	Status Register A	SRA	Read
	Clock Select Register A	CSRA	Write
FFF8 2008	RESERVED		Read [1]
	Command Register A	CRA	Write
FFF8 200C	Receive Holding Register A	RHRA	Read
	Transmit Holding Register A	THRA	Write
FFF8 2010	Input Port Change	IPCR	Read
	Auxiliary Control	ACR	Write
FFF8 2014	Interrupt Status	ISR	Read
	Interrupt Mask	IMR	Write
FFF8 2018	Counter/Timer Upper Register	CTUR	Read/Write
FFF8 201C	Counter/Timer Lower Register	CTLR	Read/Write
FFF8 2020	Mode Registers B	MR1B, MR2B	Read/Write
FFF8 2024	Status Register B	SRB	Read
	Clock Select Register B	CSRB	Write
FFF8 2028	RESERVED		Read [1]
	Command Register B	CRB	Write
FFF8 202C	Receive Holding Register B	RHRB	Read
	Transmit Holding Register B	THRB	Write
FFF8 2030	RESERVED		Read/Write
FFF8 2034	Input Port Configuration Register	IPCR	Read
	Output Port Configuration Register	OPCR	Write
FFF8 2038	Start Counter Command Register	SRCC	Read
	Set Output Port Bits Command Register	SOPBC	Write
FFF8 203C	Stop Counter Command Register	STCC	Read
	Reset Output Port Bits Register	ROPBC	Write
Parallel Port Registers			
FFF8 2400	Parallel Port Data Register	PPD	Write
	Parallel Port Status Register	PPS	Read
Keyboard Port Registers			
FFF8 2800	Receive Holding Register	RxH	Read
	Transmit Holding Register	TxH	Write
FFF8 2804	Status Register	STS	Read
FFF8 2808	Mode Register 1	MD1	Read/Write
	Mode Register 2	MD2	Read/Write
FFF8 280C	Command Register	CMD	Read/Write
FFF8 2810	Disable Clock Register	DSC	Write
FFF8 2820	Enable Transmit Clock	ENABLE_TXC	Write

[1] Do not read these registers. Reading them may hang the asynchronous line. (If this line is for the system console, the workstation may hang and require a reset.)

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 2000 - FFF8 2FFF	I/O Registers (continued)		
Serial Port Registers — DUART2			
FFF8 2C00	Mode Registers A	MR1A, MR2A	Read/Write
FFF8 2C04	Status Register A Clock Select Register A	SRA CSRA	Read Write
FFF8 2C08	RESERVED Command Register A	 CRA	Read [1] Write
FFF8 2C0C	Receive Holding Register A Transmit Holding Register A	RHRA THRA	Read Write
FFF8 2C10	Input Port Change Auxiliary Control	IPCR ACR	Read Write
FFF8 2C14	Interrupt Status Interrupt Mask	ISR IMR	Read Write
FFF8 2C18	Counter/Timer Upper Register	CTUR	Read/Write
FFF8 2C1C	Counter/Timer Lower Register	CTLR	Read/Write
FFF8 2C20	Mode Registers B	MR1B, MR2B	Read/Write
FFF8 2C24	Status Register B Clock Select Register B	SRB CSRB	Read Write
FFF8 2C28	RESERVED Command Register B	 CRB	Read [1] Write
FFF8 2C2C	Receive Holding Register B Transmit Holding Register B	RHRB THRB	Read Write
FFF8 2C30	RESERVED		Read/Write
FFF8 2C34	Input Port Configuration Register Output Port Configuration Register	IPCR OPCR	Read Write
FFF8 2C38	Start Counter Command Register Set Output Port Bits Command Register	SRCC SOPBC	Read Write
FFF8 2C3C	Stop Counter Command Register Reset Output Port Bits Register	STCC ROPBC	Read Write
FFF8 3000 - FFF8 3FFF	CIO Registers		
FFF8 3000	Port A Data register		
FFF8 3004	Port B Data register		
FFF8 3008	Port C Data register		
FFF8 300C	Control register		

[1] Do not read these registers. Reading them may hang the asynchronous line. (If this line is for the system console, the workstation may hang and require a reset.)

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 4000 - FFF8 4FFF	System Control Logic Registers		
FFF8 4004	Interrupt Enable CPU0	IEN0	Write
FFF8 4008	Interrupt Enable CPU1	IEN1	Write
FFF8 4040	Interrupt Status	IST	Read
FFF8 4080	Set Software Interrupt	SETSWI	Write
FFF8 4084	Clear Software Interrupt	CLRSWI	Write
FFF8 4088	Interrupt Source Status	ISS	Read
FFF8 408C	Clear Interrupt	CLRINT	Write
FFF8 40C0	Diagnostic Control Register	DCR	Write
FFF8 40C4	Diagnostic Status Register	DSR	Read
FFF8 40C8	Parity Address Register	PAR	Read
FFF8 5004 Read	VMEbus Interrupt Acknowledge and Vector 1		VIAV1
FFF8 5008 Read	VMEbus Interrupt Acknowledge and Vector 2		VIAV2
FFF8 500C Read	VMEbus Interrupt Acknowledge and Vector 3		VIAV3
FFF8 5010 Read	VMEbus Interrupt Acknowledge and Vector 4		VIAV4
FFF8 5014 Read	VMEbus Interrupt Acknowledge and Vector 5		VIAV5
FFF8 5018 Read	VMEbus Interrupt Acknowledge and Vector 6		VIAV6
FFF8 501C Read	VMEbus Interrupt Acknowledge and Vector 7		VIAV7
FFF8 8010	Extended Address	EXTAD	Read/Write
FFF8 8014	Extended Address Modifier	EXTAM	Read/Write
FFF8 8018	CPU Configuration	WHOAMI	Read
FFF8 9000 - FFF8 90FF	Color Graphics Subsystem Registers — Broadcast.		
FFF8 9000	Control and Status Register 0	CSR0	Read/Write
FFF8 9004	Stop/Resume Register	STOP	Read/Write
FFF8 9008	Control and Status Register 1	CSR1	Read/Write
FFF8 900C	Command Register	CMD	Read/Write
FFF8 9010	Plane Mask Register	MASK	Read/Write
FFF8 9014	Background Color Register	BACK	Read/Write
FFF8 9018	Line Pattern Register	LPAT	Read/Write
FFF8 901C	Pattern Control/WID Register	PC/WID	Read/Write
FFF8 9020	CRT Control Register 0	CRT0	Read/Write
FFF8 9024	CRT Control Register 1	CRT1	Read/Write
FFF8 9028	CRT Control Register 2	CRT2	Read/Write
FFF8 902C	Reserved		
FFF8 9030	Internal State 0	STATE0	Read
FFF8 9034	Internal State 1	STATE1	Read
FFF8 9038 - FFF8 903C	Reserved		

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 9000 - FFF8 90FF	Color Graphics Subsystem Registers — Broadcast.		
FFF8 9040	Parameter Register 0	PARM0	Read/Write
FFF8 9044	Parameter Register 1	PARM1	Read/Write
FFF8 9048	Parameter Register 2	PARM2	Read/Write
FFF8 904C	Parameter Register 3	PARM3	Read/Write
FFF8 9050	Parameter Register 4	PARM4	Read/Write
FFF8 9054	Parameter Register 5	PARM5	Read/Write
FFF8 9058	Parameter Register 6	PARM6	Read/Write
FFF8 905C	Parameter Register 7	PARM7	Read/Write
FFF8 9060	Parameter Register 8	PARM8	Read/Write
FFF8 9064	Parameter Register 9	PARM9	Read/Write
FFF8 9068	Parameter Register 10	PARM10	Read/Write
FFF8 906C	Parameter Register 11	PARM11	Read/Write
FFF8 9070	Parameter Register 12	PARM12	Read/Write
FFF8 9074	Parameter Register 13	PARM13	Read/Write
FFF8 9078	Parameter Register 14	PARM14	Read/Write
FFF8 907C	Parameter Register 15	PARM15	Read/Write
FFF8 9080 - FFF8 909C	Reserved		
FFF8 90A0	Data Port Register	DATA	Read/Write
FFF8 90A4	Palette Pointer 0	PLT0	Read/Write
FFF8 90A8	Palette Pointer 1	PLT1	Read/Write
FFF8 90AC	Blink Control Register	BLNK	Read/Write
FFF8 90B0 - FFF8 90BC	Reserved		
FFF8 90C0	RAMDAC Address Register	DAC0	Read/Write
FFF8 90C4	RAMDAC Color Palette RAM	DAC1	Read/Write
FFF8 90C8	RAMDAC Control Register	DAC2	Read/Write
FFF8 90CC	RAMDAC Overlay Palette RAM	DAC3	Read/Write
FFF8 90D0 - FFF8 90DC	Reserved		
FFF8 90E0 - FFF8 91F0	Z-Buffer Registers		
FFF8 90E0	Command Register	CMD	Read/Write
FFF8 90E4	Interrupt Register	IR	Read/Write
FFF8 90E8	Z-Constant Register	Z_CNST	Read/Write
FFF8 90EC	Initial Z-Depth Register	Z_INIT	Read/Write
FFF8 90F0	DZ/DX Register	Z_XINC	Read/Write
FFF8 90F4	DZ/DY Register	Z_YINC	Read/Write
FFF8 90F8	Hither Clip Register	HCLP	Read/Write
FFF8 90FC	Yon Clip Register	YCLP	Read/Write
FFF8 90E0	Configuration Register	CNFG	Read/Write
FFF8 90E4	State 0 Register	ST0	Read/Write
FFF8 90E8	State 1 Register	ST1	Read/Write
FFF8 90EC	State 2 Register	ST2	Read/Write
FFF8 90F0	State 3 Register	ST3	Read/Write
FFF8 90F4 - FFF8 90FF	Reserved		

(continued)

Table A-2 400 Series Station Address Map

Address	Resource Name	Mnemonic	Type
FFF8 9100 - FFF8 94FF	Color Graphics Subsystem Registers — POSN 00, 01, 10, 11. (Use offsets from Broadcast addresses.)		
FFF8 A000 - FFF8 AFFF	SCSI Port Registers		
FFF8 A000	0 - DMA byte count (low)	DMABCL	Read/Write
FFF8 A004	1 - DMA byte count (middle)	DMABCM	Read/Write
FFF8 A008	2 - DMA byte count (high)	DMABCH	Read/Write
FFF8 A00C	3 - Interrupt Mask Register 0	IM0	Write
FFF8 A010	4 - Offset Control Register	OFC	Write
FFF8 A014	5 - FIFO Status Register	FIFOS	Read
FFF8 A018	6 - Revision Control Interrupt Mask Register 1	RC IM1	Read Write
FFF8 A01C	7 - Status Register 0 Control Register 0	SR0 CR0	Read Write
FFF8 A020	8 - Status Register 1 Control Register 1	SR1 CR1	Read Write
FFF8 A024	9 - SCSI Signal Register	SCSIS	Read/Write
FFF8 A028	A - SCSI ID Register	SCSIID	Read/Write
FFF8 A02C	B - Source/Dest ID	SRDTID	Read
FFF8 A030	C - Memory Data Register	MMD	Read/Write
FFF8 A034	D - Port A Register	PTA	Read/Write
FFF8 A038	E - Port B Register	PTB	Read/Write
FFF8 A03C	F - SCSI Latch Data SCSI BSY RST (TGT)	SCSILD SCSIBR	Read Write
FFF8 B000 - FFF8 BFFF	DMA Registers		
FFF8 B000	Map Register	DMAMAP	Read/Write
FFF8 B004	Offset Register	DMAOFF	Read/Write
FFF8 B008	Transfer Count Register	DMATC	Read/Write
FFF8 B00C	Transfer Direction Register	DMATD	Write
FFF8 B010	Clear DMA Map Valid Error Interrupt	DMACMV	Write
FFF8 B014	Clear DMA Write Protect Error Interrupt	DMACWP	Write
FFF8 B018	DMA Stop Register	DMASTP	Write
FFF8 C000 - FFF8 CFFF	LAN Interface Registers		
FFF8 C000	Register Data Pointer	RDP	Read/Write
FFF8 C004	Register Address Pointer	RAP	Read/Write
FFFF 0000 - FFFF FFFF	VMEbus A16 space (64 Kbytes)		Read/Write

(concluded)

End of Appendix

Appendix B
Power-Up Flowchart

The following flowcharts outline the steps the workstation performs during either a cold start or a warm start. A cold start is initiated when the workstation is powered up, while a warm start results from a software system reset. The cold start flow begins with the "Initial Power-Up Flowchart" (Figure B-1), and the warm start begins with the "Reset Flowchart" (Figure B-2). This appendix discusses powerup to the point of booting an operating system (Automatic Program Load); it does not include booting an operating system.

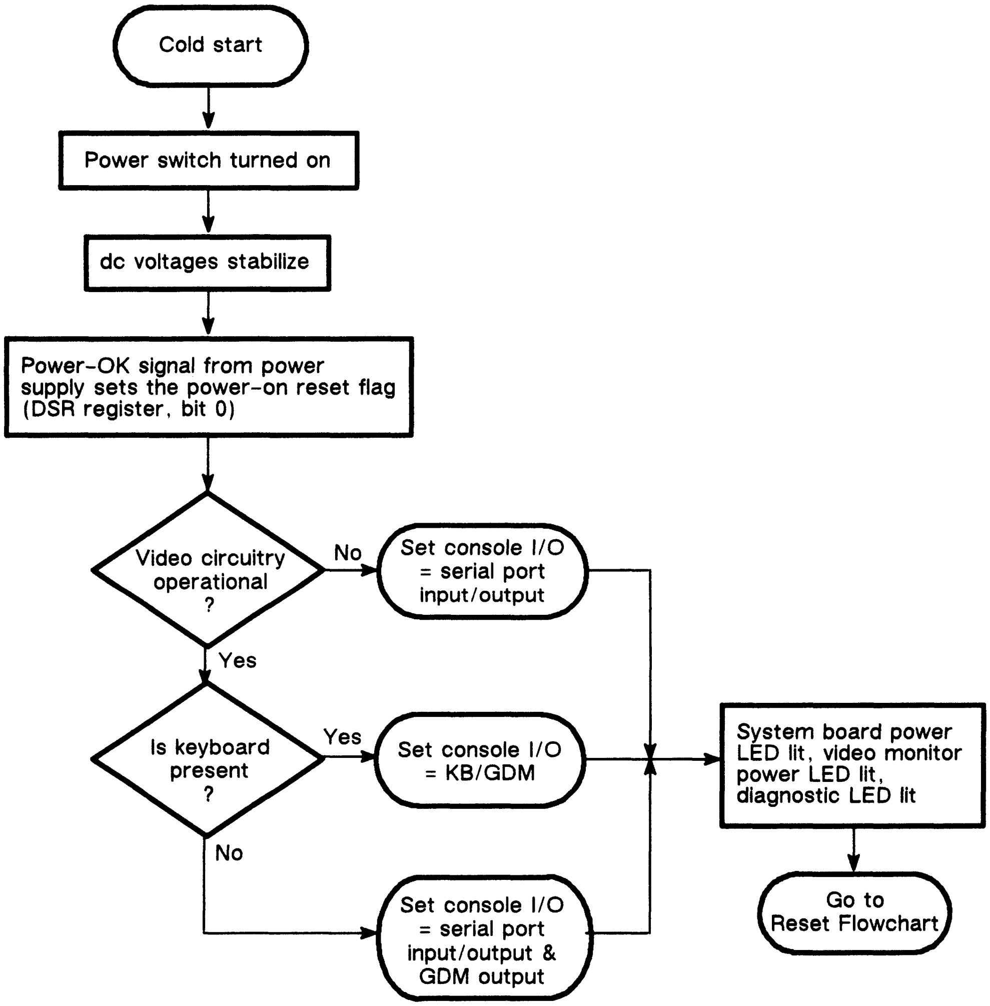
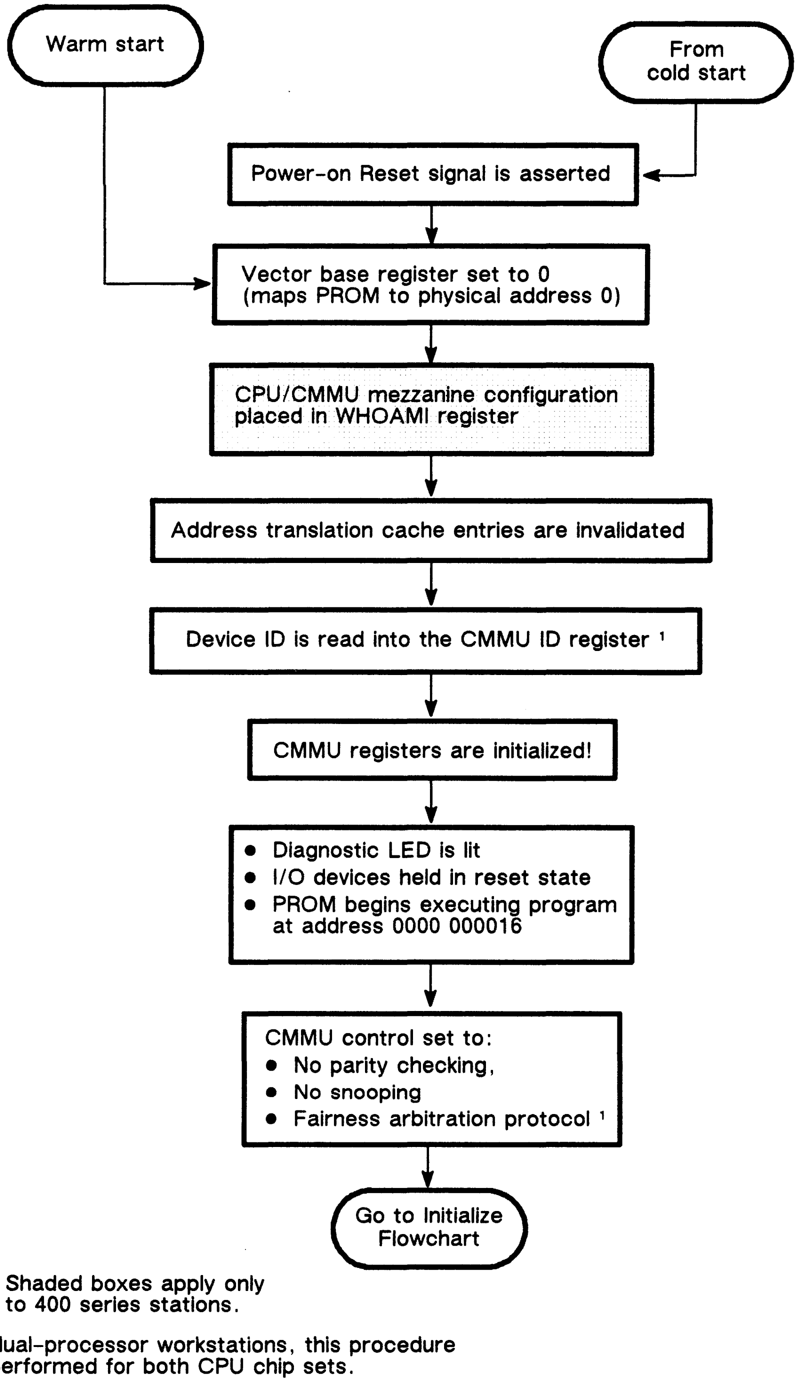
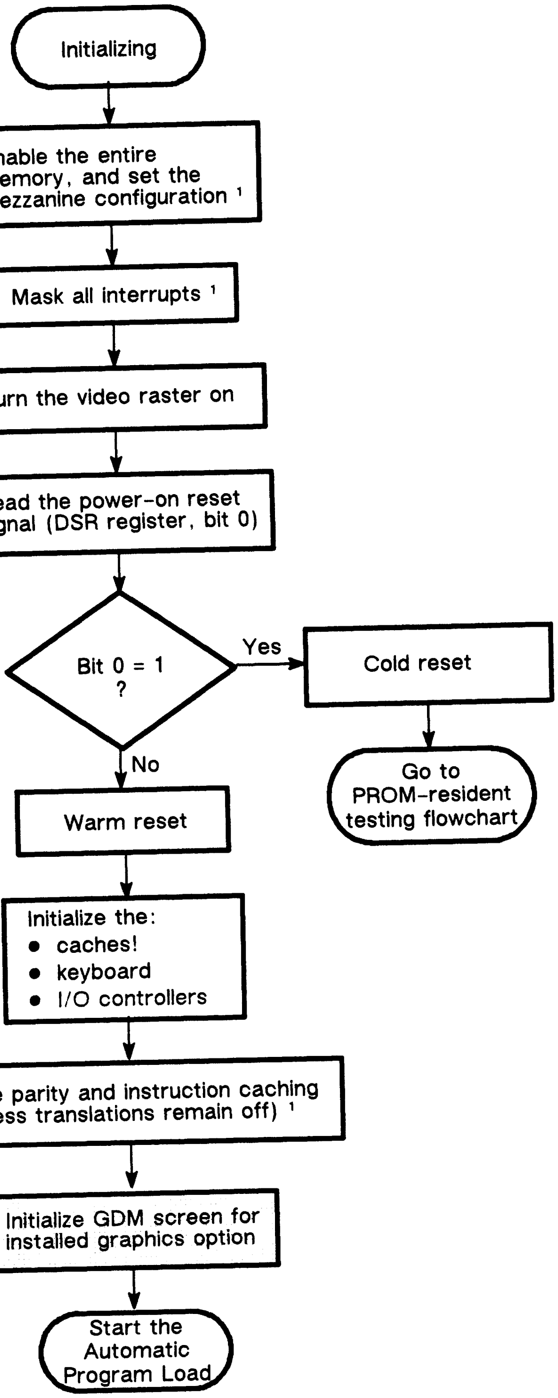

Cold start
Power switch turned on
dc voltages stabilize
Power-OK signal from power supply sets the power-on reset flag (DSR register, bit 0)
Video circuitry operational ?
No
Set console I/O = serial port input/output
Yes
Is keyboard present ?
Yes
Set console I/O = KB/GDM
No
Set console I/O = serial port input/output & GDM output
System board power LED lit, video monitor power LED lit, diagnostic LED lit
Go to Reset Flowchart

Figure B-1 Initial Power-Up Flowchart


Warm start
From cold start
Power-on Reset signal is asserted
Vector base register set to 0 (maps PROM to physical address 0)
CPU/CMMU mezzanine configuration placed in WHOAMI register
Address translation cache entries are invalidated
Device ID is read into the CMMU ID register [1]
CMMU registers are initialized!
● Diagnostic LED is lit
● I/O devices held in reset state
● PROM begins executing program at address 0000 000016
CMMU control set to:
● No parity checking,
● No snooping
● Fairness arbitration protocol [1]
Go to Initialize Flowchart
▨ Shaded boxes apply only to 400 series stations.

[1] In dual-processor workstations, this procedure is performed for both CPU chip sets.

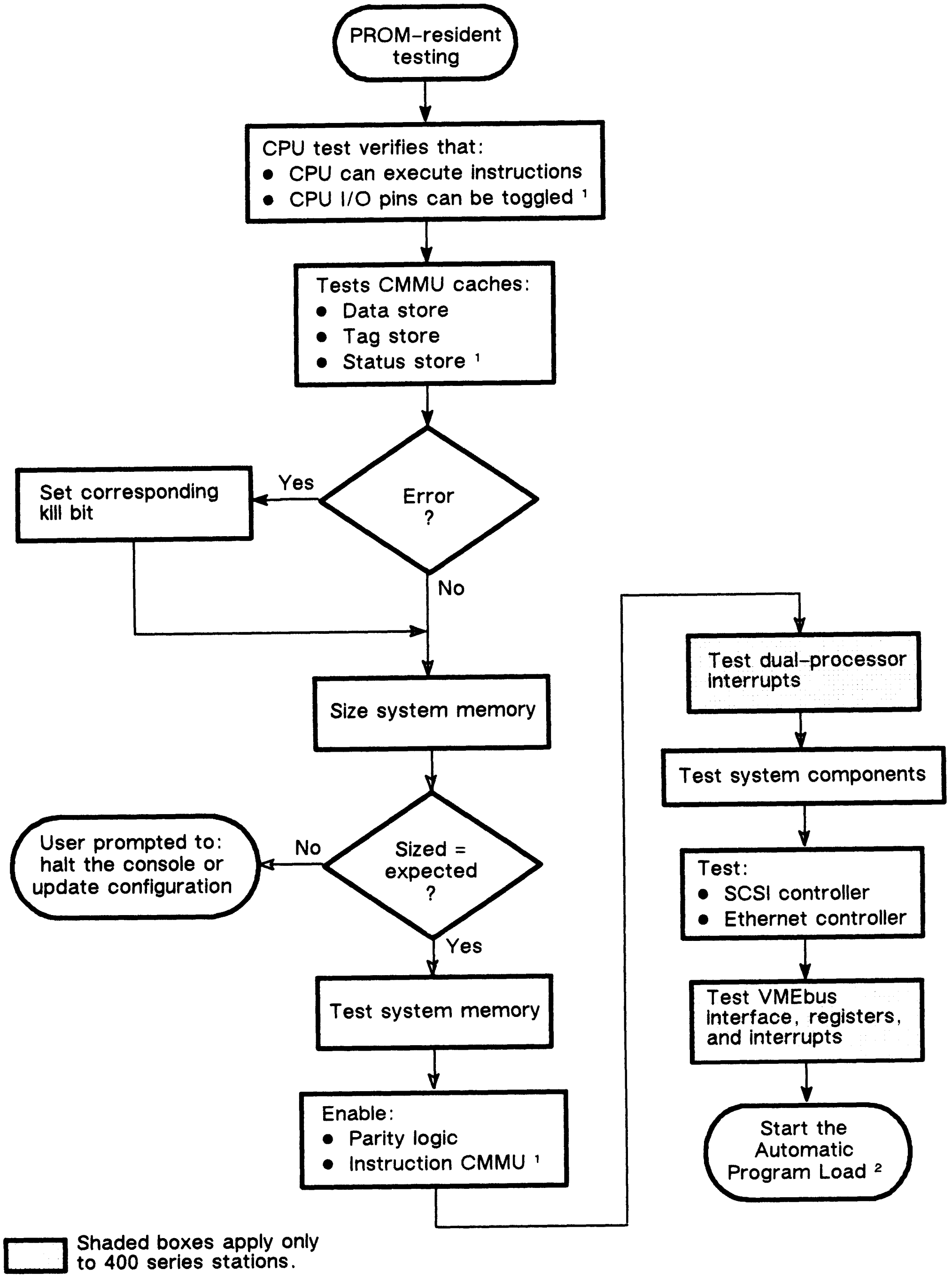
Figure B–2 Reset Flowchart


Initializing
Enable the entire memory, and set the mezzanine configuration [1]
Mask all interrupts [1]
Turn the video raster on
Read the power-on reset signal (DSR register, bit 0)
Bit 0 = 1 ?
Yes
Cold reset
Go to PROM-resident testing flowchart
No
Warm reset
Initialize the:
● caches!
● keyboard
● I/O controllers
Enable parity and instruction caching (address translations remain off) [1]
Initialize GDM screen for installed graphics option
Start the Automatic Program Load

▨ Shaded boxes apply only to 400 series stations.

[1] In dual-processor workstations, this procedure is performed for both CPU chip sets.

Figure B-3 Initialize Flowchart


PROM-resident testing
CPU test verifies that:
● CPU can execute instructions
● CPU I/O pins can be toggled ¹
Tests CMMU caches:
● Data store
● Tag store
● Status store ¹
Error ?
Yes
Set corresponding kill bit
No
Size system memory
Sized = expected ?
No
User prompted to: halt the console or update configuration
Yes
Test system memory
Enable:
● Parity logic
● Instruction CMMU ¹
Test dual-processor interrupts
Test system components
Test:
● SCSI controller
● Ethernet controller
Test VMEbus interface, registers, and interrupts
Start the Automatic Program Load ²
Shaded boxes apply only to 400 series stations.

[1] In dual-processor workstations, this procedure is performed for both CPU chip sets.

[2] At this point in a dual-processor configuration, CPU1 is idled with the SCM running on CPU0.

Figure B–4 PROM-Resident Testing Flowchart

End of Appendix

Appendix C
Boot File Format

If you are booting a non–Data General operating system, you must provide part of the bootstrap code in the form of an **a.out** file. With the exception of certain restrictions, this **a.out** file must conform to the AT&T common object file format (coff). The rest of this appendix describes the booting sequence, the arguments passed by the bootstrap program, and the **a.out** file.

Booting Sequence

When you boot the workstation from tape, the bootstrap program provided in the system PROM reads a file off the boot device starting at block 0. This bootstrap program expects to find the **a.out** file at block 0. It loads the **a.out** file into main memory based on file–header information contained within the file. It passes arguments to the **a.out** file that define the boot path. Then it starts executing the program in the **a.out** file.

For information on booting from disk, refer to your Data General sales representative.

Arguments Passed by the Bootstrap Program

When booting the system, the bootstrap program passes the arguments contained in the following 88000 registers to the **a.out** file:

- r2 = Pointer to boot string entered by user or NOVRAM.
- r3 = Device, cracked from boot string.
- r4 = Unit number, cracked from boot string.
- r5 = Part number, cracked from boot device.

NOTE: The bootstrap program does not pass the memory size. Use the SCM system call **.MSIZE** to pass the memory size. For more SCM system call information, see Chapter 2, "Programming the System Board."

a.out File

When using a non–Data General operating system, you must provide the **a.out** file. With the exception of certain restrictions, this file must conform to the AT&T common object file format (coff). The restrictions are listed in the sections that follow and the AT&T common object file format as specified in the Binary Compatibility Standard (BCS), 88open Consortium, Ltd. The next sections describe the parts of this coff file.

Contents of File Header

The file header contains the data structures described in the 4.2 File Header section of the BCS document. The bootstrap program uses the following fields and ignores all others:

- f_magic
- f_nscns
- f_opthdr
- f_flags

(Also note that the magic number, MC88MAGIC, is 0D.)

Contents of Optional Header

The optional header file contains the data structures described in the 4.3 Optional Header section of the BCS document. The bootstrap program uses the "magic" and "entry" fields and ignores all others.

Contents of Section Header File

The section header file contains the data structures described in the 4.4 Section Headers section of the BCS document. The bootstrap program uses the following fields and ignores all others:

- s_name
- s_vaddr
- s_size
- s_scnptr
- s_flags

Although the s_vendor field is not marked "ignored by exec," do not use this field.

Section Header File Restrictions

The header file must conform to the following restrictions:

- The system supports exactly one each of the **.text**, **.data**, and **.bss** sections. All other section types are ignored without side effects.
- The virtual address **s_vaddr** specifies the virtual address at which the section is loaded and aligned on an 8–byte boundary. The system treats this value as a physical address.
- The Raw Section Data offset specifier **s_scnptr** specifies the start of the Raw Data to be loaded (on an 8–byte boundary). Sections that specify this value as 0 have no section data and their address space should be filled with 0s. The system does not perform the zero fill when **s_scnptr** has a value of 0.
- The system memory map at boot time consists of 3 Mbytes of contiguous physical memory beginning at physical location 0. A bootable image should not specify **.text** or **.data** section destinations greater than the 3–Mbyte boundary. The **.bss** section is exempt from this restriction since no memory is loaded or initialized by the **.bss** section.

End of Appendix

Appendix D
System Board Connectors

This appendix identifies the connectors and signals available on your workstation. Table D-1 lists the connectors for the workstations. Figure D-1 identifies the external connectors on 300 series stations, and Figure D-2 identifies the system board connectors on 300 series stations. Figure D-3 identifies the external connectors on 400 series stations, and Figure D-4 identifies the system board connectors on 400 series stations. The remainder of the Appendix describes the signals present in the connectors.

Table D-1 Connectors on the System Board

Subsystem	Connector Number	Connector Type
Monochrome graphics	J2	BNC (Female)
Color graphics	J1, J2, J3	BNC (Female)
LAN interface	J6	D15 (Female)
Serial port (300 Series)	J10	D25 (Male)
Serial ports (400 Series)	J4, J10	D25 (Male)
Parallel port	J7	D25 (Female)
Mouse port (300 Series)	J9	Mini DIN-8 (Female)
Mouse port (400 Series)	J9	Mini DIN-8 (Female)
Keyboard port	J8	DIN-8 (Female)
SCSI port	J5	D50
VMEbus (400 Series)	J30, J31	96-pin

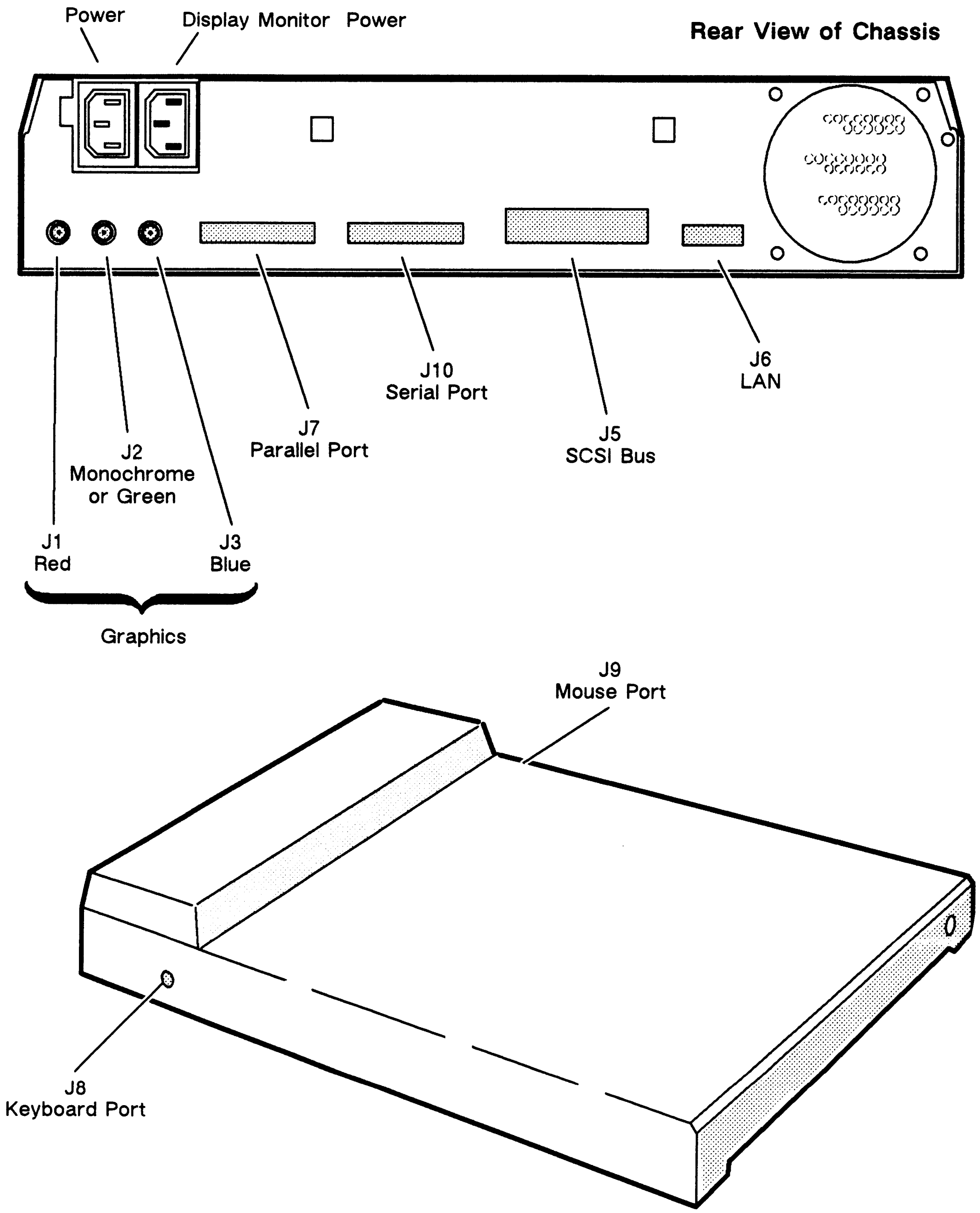
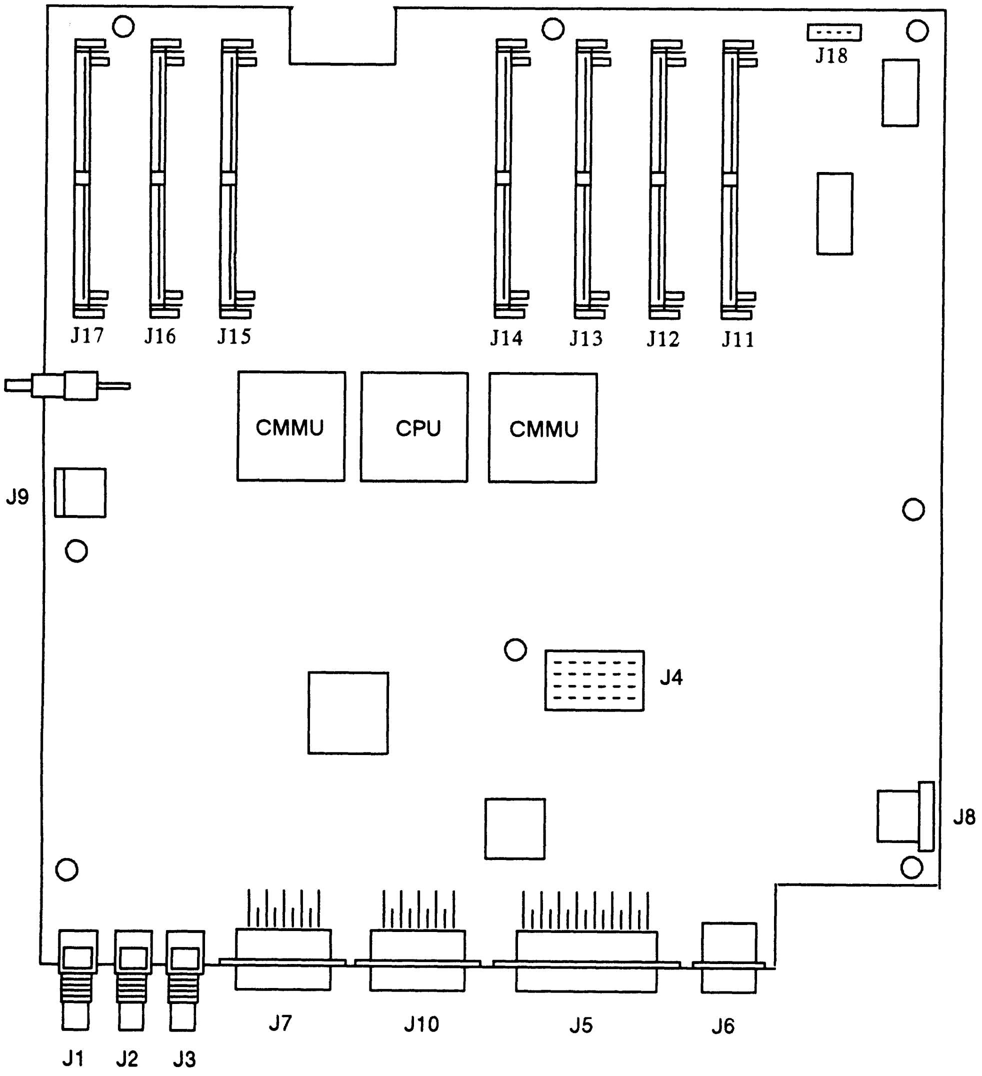


Figure D–1 300 Series External Connectors



J1	Color Graphics Monitor (Red)
J2	Color Graphics Monitor (Green), or Monochrome Graphics Monitor
J3	Color Graphics Monitor (Blue)
J4	Power Supply
J5	SCSI Bus
J6	LAN
J7	Parallel Port
J8	Keyboard
J9	Mouse
J10	Serial Port
J18	Speaker and LED

J11	Memory Board 1
J12	Memory Board 2
J13	Memory Board 3
J14	Memory Board 4
J15	Memory Board 5
J16	Memory Board 6
J17	Memory Board 7

Figure D-2 300 Series System Board Connectors

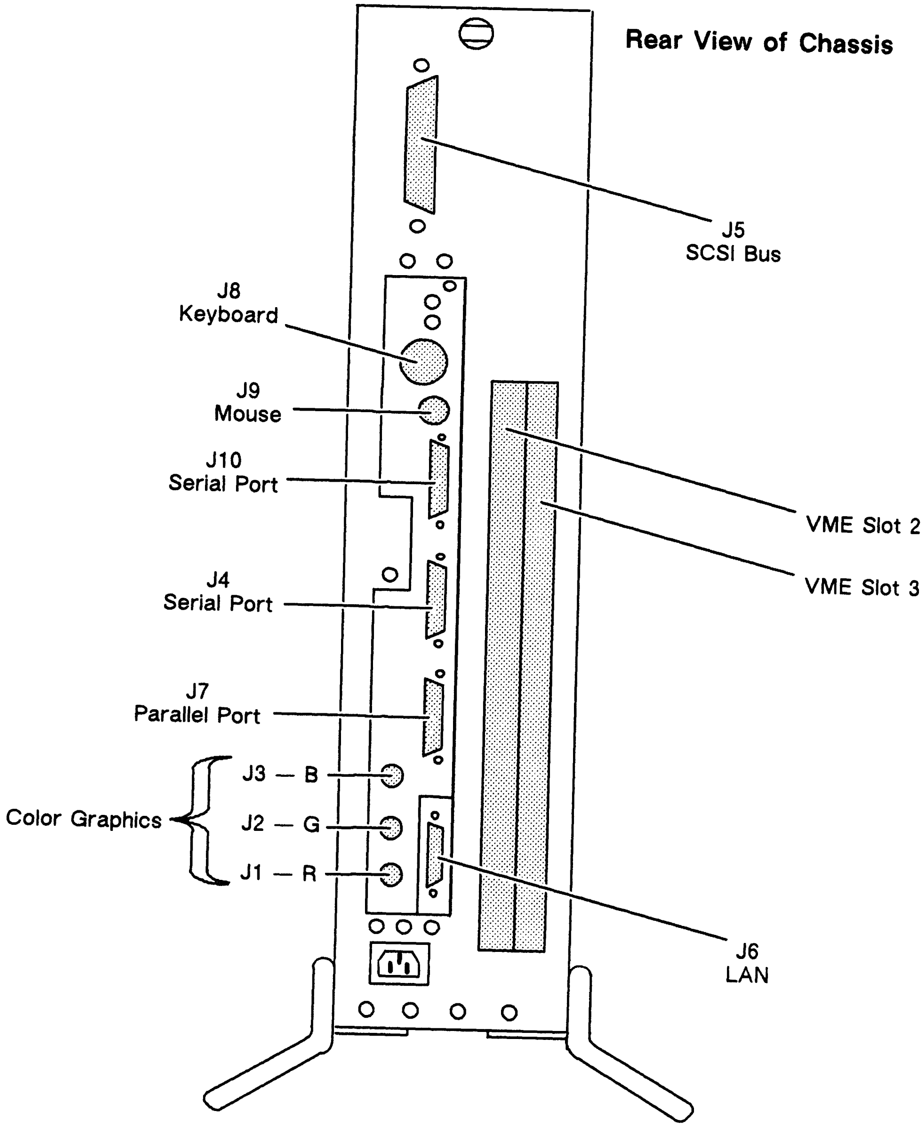
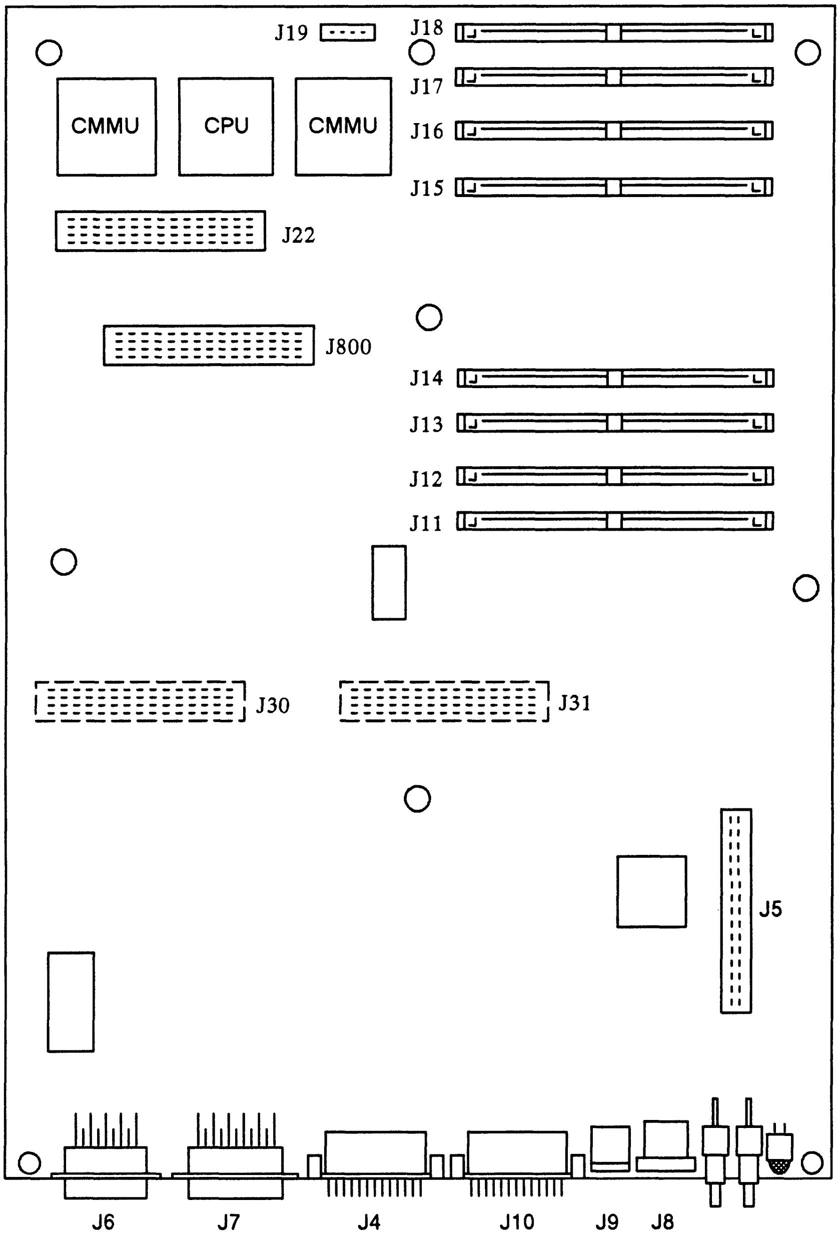


Figure D–3 400 Series External Connectors



J4	Serial Port #1	J11	Memory Board 1
J5	SCSI Bus	J12	Memory Board 2
J6	Ethernet LAN	J13	Memory Board 3
J7	Parallel Port	J14	Memory Board 4
J8	Keyboard	J15	Memory Board 5
J9	Mouse	J16	Memory Board 6
J10	Serial Port #2	J17	Memory Board 7
J19	Speaker and LED	J18	Memory Board 8
J22	Optional CPU Set (Mezzanine Board)		
J30	VMEbus (P1)		
J31	VMEbus (P2)		
J800	Graphics Controller (Mezzanine Board)		

Figure D–4 400 Series System Board Connectors

Serial Port Connector (300 Series)

Serial devices connect to the serial port through a 25-pin D-connector (J10). Table D-2 identifies the serial port signals.

Table D-2 Serial Connector Signals (300 Series)

Pin	Signal	Direction
1	CG (RS-232-C)	-----
2	TxD (RS-232-C)	From system board
3	RxD (RS-232-C)	To system board
4	RTS (RS-232-C)	From system board
5	CTS (RS-232-C)	To system board
6	DSR (RS-232-C)	To system board
7	SG (RS-232-C)	-----
8	DCD (RS-232-C)	To system board
9	RxD + (RS-422)	To system board
10	RxD - (RS-422)	To system board
11	TxD + (RS-422)	From system board
12	TxD - (RS-422)	From system board
13	RS-422 Select (Ground selects RS-422)	To system board
14-19	Unused	-----
20	DTR (RS-232-C)	From system board
21	Unused	-----
22	RI (RS-232-C)	To system board
23-25	Unused	-----

Serial Port Connectors (400 Series)

Serial devices connect to the serial port through 25-pin D-connectors (J4 and J10). J4 is channel A and J10 is channel B. Table D-3 identifies the serial port signals.

Table D-3 Serial Connector Signals (400 Series)

Pin	Signal	Direction
1	CG	-----
2	TxD	From system board
3	RxD	To system board
4	RTS	From system board
5	CTS	To system board
6	DSR	To system board
7	SG	-----
8	DCD	To system board
9-19	Unused	-----
20	DTR	From system board
21	Unused	-----
22	RI	To system board
23-25	Unused	-----

Parallel Port Connector (300 Series)

A parallel device connects to the parallel port through a 25–pin connector (J7). Table D–4 identifies the parallel port signals.

Table D–4 Parallel Connector Signals (300 Series)

Pin	Signal	Function
1	DATASTRB	This strobe pulse reads data in from the printer. Timing and polarity for this signal depend on whether software configured the parallel port as a Data Products–type interface or a Centronics–type interface. For more information, see "Programming the Data Strobe and Data Select Signals" in Chapter 7.
2 - 9	D1 - D8	D1 is the least significant bit.
10	DEMAND	Indicates that the printer is demanding another character.
11	BUSY	Tells the system that the printer is busy and cannot accept another character.
12	PE	Indicates that the printer is out of paper.
13	SELECT	The system selects the printer using this signal.
14	Unused	
15	FAULT	Indicates a printer error.
16	ONLINE	Indicates that the printer is on line.
17–25	Unused	

Parallel Port Connector (400 Series)

A parallel device connects to the parallel port through a 27–pin connector (J7). Table D–5 identifies the parallel port signals.

Table D–5 Parallel Connector Signals (400 Series)

Pin	Signal	Function
1	DATASTRB	This strobe pulse reads data in from the printer. Timing and polarity for this signal depend on whether software configured the parallel port as a Data Products–type interface or a Centronics–type interface. For more information, see "Programming the Data Strobe and Data Select Signals" in Chapter 7.
2 - 9	D1 - D8	D1 is the least significant bit.
10	DEMAND	Indicates that the printer is demanding another character.
11	BUSY	Tells the system that the printer is busy and cannot accept another character.
12	PE	Indicates that the printer is out of paper.
13	SELECT	The system selects the printer using this signal.
14	Unused	
15	FAULT	Indicates a printer error.
16	ONLINE	Indicates that the printer is on line.
17–25	Unused	

Keyboard Connector

The keyboard cable connects to the keyboard port through an 8-pin DIN connector (J8). Table D-6 identifies the keyboard signals.

Table D-6 Keyboard Signals

Pin	Signal	Direction
1	Clock	To keyboard
2	Data	From keyboard
3	Unused	-----
4	Ground	-----
5	+5 V	To keyboard
6-8	Unused	-----

Speaker Connector

The speaker and LED cable connects to the system board through a 4-pin connector (J18 in 300 Series, J19 in 400 Series). Table D-6 identifies the signals.

Table D-7 Speaker Signals

Pin	Signal
1	Speaker enable
2	+5 V
3	Unused
4	Optional CPU present (asserted low if optional CPU is present)

Mouse Connector

The mouse connects to the mouse port through an 8-pin connector (J9). The mouse is compatible with the EIA RS-232-C interface (1200 baud, asynchronous, 1 start bit, 1 stop bit, 8 data bits). It obtains its power from the RS-232-C interface (+/- 13.2 Vmax. and 15 mA max). Table D-8 identifies the mouse port signals.

Table D-8 Mouse Signals

Pin	Signal	Direction
1	RTS	From system board
2	DTR	From system board
3-5	Unused	-----
6	GND	-----
7	TxD	From system board
8	RxD (mouse data)	To system board

Power Connector (300 Series)

The power connector supplies power and power status signals to the system board of 300 series workstations. Table D-9 identifies the power signals.

Table D-9 Power Connector (300 Series)

Pin	Signal	Direction
1	IRQ_PF*	To system board
2	-12V_NET	To system board
4, 3	-6V	To system board
5	POWER_OK	To system board
6	Ground	To system board
8, 7	+12V_NET	To system board
18-9	Ground	To system board
24-19	+5V	To system board

SCSI Connector (300 Series)

In the 300 series station, the SCSI bus connector (J5), located on the system board, projects immediately out the back of the computer. Table D-10 identifies the signals at J5.

Table D-10 SCSI Connector (300 Series)

Pin	Signal	Pin	Signal
1	I/O	22	BSY
3	SEL	28	DBP
5	ACK	30	DB5
7	ATN	32	DB2
9	TERMPWR	35	C/D
13	DB6	37	RST
15	DB3	42	Unused
17	DB0	45	DB7
18	REQ	47	DB4
20	MSG	49	DB1

NOTE: Pins not listed connect to ground.

SCSI Connector (400 Series)

In the 400 series station, the SCSI bus connector (J5), located on the system board, does not appear on the outside of the chassis. A SCSI cable connects to J5, to internal drives, and to an external SCSI connector. Table D-11 identifies the signals at J5 on the system board.

Table D-11 SCSI Connector (400 Series)

Pin	Signal	Pin	Signal
1 R	DB0	26	TERMPW
4	DB1	32	ATN
6	DB2	36	BSY
8	DB3	38	ACK
10	DB4	40	RST
12	DB5	42	MSG
14	DB6	44	SEL
16	DB7	46	C/D
18	DBP	48	REQ
25	Unused	50	I/O

NOTE: Pins not listed connect to ground.

LAN Connector

The LAN interface provides a D15 connector for an AUI cable. The AUI cable connects the workstation to an external medium attachment unit (MAU). The MAU contains the Ethernet transceiver and the medium dependent interface (MDI) for connection to the physical network. The MAU provides electrical isolation between the AUI cable and the physical network. The Ethernet interface can be attached via the AUI cable to any one of the following types of external 10-MHz MAUs: 10BASE5 (Ethernet), 10BASE2 (Cheapernet or Thin Ethernet), 10BROAD36 (Ethernet over CATV), 10BASET (proposed Ethernet over twisted pair), or any other 10-MHz AUI compatible MAU or MAU-like device that does not require the Control Out signal specified in the AUI definition. Table D-12 identifies the LAN signals.

Table D-12 LAN Interface Connector Signals

Pin	Signal	Circuit Name
1	Ground	CI-S (Control In Shield)
2	Collision +	CI-A (Control In A)
3	Transmit +	DO-A (Data Out A)
4	Ground	DI-S (Data In Shield)
5	Receive +	DI-A (Data In A)
6	Ground	VC (Voltage Common)
7	No Connection	CO-A (Control Out circuit A)
8	Ground	CO-S (Control Out Shield)
9	Collision -	CI-B (Control In B)
10	Transmit -	DO-B (Data Out B)
11	Ground	DO-S (Data Out Shield)
12	Receive -	DI-B (Data In B)
13	+12 V	VP (Voltage Plus)
14	Ground	VS (Voltage Shield)
15	No Connection	CO-B (Control Out B)
Shell	Ground	PG (Protective Ground)

VMEbus Connectors (400 Series)

Figure D–5 shows the location of the VMEbus slots and connectors and identifies the pin and row positions. Table D–13 identifies the VMEbus signals on connector J1; Table D–14 identifies the VMEbus signals on connector J2. The manual *The VMEbus Specification* describes the signals and interface in detail.

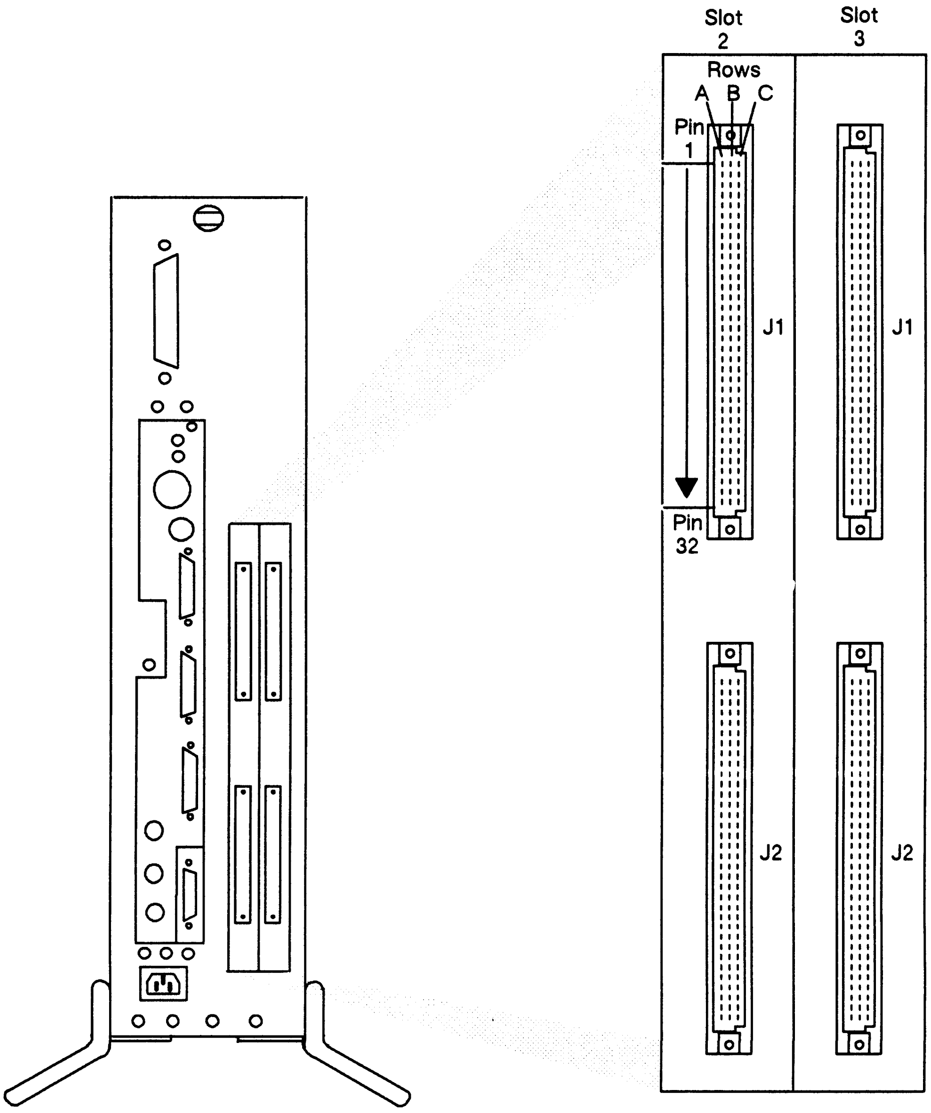


Figure D–5 VMEbus Slots, Connectors, and Pin Designations

Table D-13 VMEbus Connector J1

Pin		Signals	
	Row A	Row B	Row C
1	D00	BBSY*	D08
2	D01	BCLR* (Unused)	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN* (Unused)	D11
5	D04	BG0OUT* (Unused)	D12
6	D05	BG1IN* (Unused)	D13
7	D06	BG1OUT* (Unused)	D14
8	D07	BG2IN* (Unused)	D15
9	GND	BG2OUT* (Unused)	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0* (Unused)	SYSRESET*
13	DS0*	BR1* (Unused)	LWORD*
14	WRITE*	BR2* (Unused)	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK (Unused)	A17
22	IACKOUT*	SERDAT* (Unused)	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5V STDBY (Unused)	+12V
32	+5V	+5V	+5V

Table D-14 VMEbus Connector J2

Pin	Row A	Signals Row B	Row C
1	+5V	+5V	GND
2	+5V	GND	GND
3	+5V	Unused	GND
4	+5V	A24	GND
5	+5V	A25	GND
6	+5V	A26	GND
7	+5V	A27	GND
8	+5V	A28	GND
9	+5V	A29	GND
10	+5V	A30	GND
11	+5V	A31	GND
12	+5V	GND	GND
13	+5V	+5V	GND
14	+5V	D16	GND
15	+5V	D17	GND
16	+5V	D18	GND
17	+5V	D19	GND
18	+5V	D20	GND
19	+5V	D21	GND
20	+5V	D22	GND
21	+5V	D23	GND
22	+5V	GND	GND
23	+5V	D24	GND
24	Unused	D25	GND
25	Unused	D26	GND
26	Unused	D27	GND
27	+12V	D28	GND
28	Unused	D29	GND
29	GND	D30	GND
30	GND	D31	GND
31	GND	GND	GND
32	GND	+5V	GND

End of Appendix

Index

Within this index, the page reference is the first page for an entry (even if the subject spans several pages).

Documentation Set

This section lists additional documents currently available for AViiON 300 and 400 series stations. Those documents specifically referred to in the text of this manual are also listed in the "Related Manuals" section of the Preface.

Hardware Manuals

AViiON™ 300 and 400 Series Stations: Programming System Control and I/O Registers (014–001800)

> Describes the workstation architecture and explains how to program the system control logic, monochrome and color graphics controller subsystems, keyboard port, mouse port, serial and parallel ports, LAN interface, and SCSI port.

Ethernet/IEEE 802.3 Local Area Network Installation Guide (014–000793)

> Explains how to install both the coaxial cable plant of an Ethernet local area network (LAN) and the transceivers that connect the network to a node communication controller.

Expanding and Maintaining AViiON™ 400 Series Stations (014–001859)

> Explains how to add or replace components (mouse, keyboard, monitor, drives, memory modules, system board assembly, CPU board, Z–buffer board, graphics board, power supply, fan assembly, and PROM).

Maintaining AViiON™ 300 Series Stations (014–001803)

> Explains how system administrators can replace components (mouse, keyboard, monitor, memory modules, system board assembly, power supply, SCSI bus fuse, and fan).

Installing and Operating the Model 10565 Peripheral Housing Unit (014–001810)

> Describes how to unpack, install, and power up the subsystem. Explains how to replace the power supply, line cord, and fan, and provides general instructions for replacing a drive. Lists electrical and environmental specifications of the subsystem.

MC88100 RISC Microprocessor User's Manual (014–001809)

> Describes the Motorola 88100 Central Processing Unit (CPU), including the registers, addressing modes, internal and bus timing, and assembly–language instruction set.

MC88200 Cache/Memory Management Unit User's Manual (014–001808)

> Describes the Motorola 88200 Cache/Memory Management Unit (CMMU), including the CMMU registers, the cache and cache coherency, memory management and user/supervisor space, the Processor bus (Pbus) and the Memory bus (Mbus).

Setting Up and Installing VMEbus Options in AViiON™ Systems (014–001867)

Describes how to jumper VME controllers to operate in an AViiON environment. Explains how to install and remove the controller boards in the system's VME card cage, and how to jumper the VME printed circuit backplane when necessary. Also supplies instructions for connecting external devices to the controller boards.

Setting Up and Starting AViiON™ 300 Series Stations (014–001801)

Describes how to unpack and connect system components and optional devices. Explains how to power up the workstation and prepare for the operating system installation. Includes electrical and environmental specifications of the workstation, including the computer unit, monitor, keyboard, and mouse.

Setting Up and Starting AViiON™ 400 Series Stations (014–001858)

Describes how to unpack and connect system components and optional devices. Explains how to power up the workstation, run diagnostics, and prepare for the operating system installation. Includes electrical and environmental specifications of the workstation, including the computer unit, monitor, keyboard, and mouse.

Using AViiON™ System Diagnostics (014–001863)

Describes how to can use menu–based utilities to verify system hardware, test terminal or graphics display, test the functionality of a graphics keyboard and mouse, check for faults in LAN connections, and maintain cartridge tape and diskette media on AViiON hardware models.

Using the AViiON™ System Control Monitor (SCM) (014–001802)

Describes how to can use the commands and menus of the firmware monitor program to bring up software, control their system environment, and debug programs on AViiON hardware models.

Software Manuals

88open Binary Compatibility Standard (069–701043)

Describes the binary standards for developing portable 88K code using the C programming language.

Installing and Managing the DG/UX™ System (093–701052)

Shows how to install and manage the DG/UX operating system on AViiON hosts that will run as stand–alone, server, or client systems. Intended for system administrators who are familiar with the UNIX operating system.

Installing the DG/UX™ System on an AViiON™ Workstation with a Hard Disk (069–000520)

> Describes installing the DG/UX operating system on an AViiON workstation with a preloaded hard disk. Assumes that the workstation will operate in a stand–alone environment since the workstation is not connected to an Ethernet LAN and, therefore, does not require the installation of network software. Supports those who are unfamiliar with UNIX, and assumes no previous knowledge of the DG/UX system or UNIX. But does assume the reader has some familiarity with another operating system, such as MS–DOS®.

Other Organizations' Documents

The following documents are available from other organizations.

μPD72120 Advanced Graphics Display Controller User's Manual

> Describes the *μPD72120* graphics controller and how to program it. This document is available from NEC Electronics, Inc.

AIC–6250 High–Performance SCSI Protocol Chip

> Describes the AIC–6250 SCSI controller and how to program it. This document is available from Adaptec, Inc.

AM7990 Local Area Network Controller (LANCE) Technical Manual

> Describes the AM 7990 LAN controller and how to program it. This document is available from Advanced Micro Devices, Inc.

Brooktree® Product Databook

> This document is available from Brooktree Corporation.

Memory Products Databook

> This document is available from SGS–Thompson Microelectronics.

SCN2661 Enhanced Programmable Communications Interface (EPCI) Product Specification

> Describes the SCC2692 universal synchronous/asynchronous data communications controller and how to program it. This document is available from Signetics, Inc.

SCC2692 Dual Asynchronous Receiver Transmitter (DUART) Product Specification

> Describes the SCN2661 DUART and how to program it. This document is available from Signetics, Inc.

The VMEbus Specification (Motorola document number HB212)

> Describes Motorola's Versa Modula Europa bus (VMEbus), and how to program using the VMEbus. This document is available from Motorola Corp.

Z8536 CIO Counter/Timer and Parallel I/O Unit Technical Manual

> Describes the Z8536 CIO and how to program it. This document is available from Zilog, Inc.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name ______________________ **Your Title** ______________

Company ______________________ **Phone** ______________

Street ____

City ______________________ **State** __________ **Zip** ______

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title ______________________ **Manual No.** ______________

Who are you?

□EDP/MIS Manager
□Senior Systems Analyst
□Engineer
□Analyst/Programmer
□Operator
□End User
□Other ______________
____

How do you use this manual? *(List in order: 1 = Primary Use)*

___ Introduction to the product
___ Reference
___ Tutorial Text
___ Operating Guide
___ Other
____

		Yes	No
About the manual:	Is it easy to read?	□	□
	Is it easy to understand?	□	□
	Are the topics logically organized?	□	□
	Is the technical information accurate?	□	□
	Can you easily find what you want?	□	□
	Does it tell you everything you need to know?	□	□
	Do the illustrations help you?	□	□

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

Data General

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

Cut here and insert in binder spine pocket

DataGeneral
Data General Corporation, Westboro, Massachusetts 01580