DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Programming System Control and I/O Registers: AViiON® 4600 and 530 Series

014–002076–01

AViiON®
PRODUCT LINE

Programming System Control and I/O Registers: AViiON® 4600 and 530 Series

014–002076–01

Printed in the United States of America
Rev. 01, June 1992
Ordering No. 014−002076

Notice

Programming System Control and I/O Registers: AViiON® 4600 and 530 Series
014–002076–01

Revision History:

Original Release – September 1991
First Revision – June 1992

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

Preface

This manual is written for persons who are either developing a UNIX® operating system to run on AViiON® 4600 and 530 series, or adapting an existing UNIX operating system. This manual may also be used by hardware designers who are linking hardware to an AViiON 4600 or 530.

Organization

This manual contains the following chapters and appendixes:

Chapter 1 System Board Architecture

Describes the architecture of the system board and how this architecture relates to the rest of the system. This includes descriptions of the primary functional blocks of the system board: CPU, memory, registers, I/O, address decode, and bus arbitration.

Chapter 2 Addressing and DMA

Describes address decoding and how to create address maps; how the CPU addresses system board resources, memory, and VME controllers; and how VME controllers address memory and global registers.

Chapter 3 Interrupts

Describes all interrupts, including how they are generated and handled, and the registers involved in the interrupt processes.

Chapter 4 Global Resources

Describes the system control, status, and configuration registers, including the Global Control and Status Registers (GCSR) used by the VMEbus.

Chapter 5 Memory

Describes the memory, including addressing and error checking and correction,.

Chapter 6 Programming the Color Graphics Subsystem

Describes how to program the color graphics subsystem, including the Z−buffer.

Chapter 7 Programming the Keyboard Interface

Describes how to program the keyboard interface.

Chapter 8 Programming the Serial and Parallel Interfaces

Describes how to program the system board's serial and parallel interfaces.

Chapter 9 Programming the LAN and SCSI Interfaces

Describes the LAN and SCSI interfaces and how to program them.

Chapter 10 Programming the Real-Time Clock and the Programmable Interval Timer

Describes the RTC and the PIT and how to program them.

Chapter 11 The System Control Monitor (SCM)

Describes the System Control Monitor (SCM), including the SCM system calls, the Environment Control Word (ECW), and the boot file format.

Appendix A Address Map

Defines the addresses of all system board registers, memory, and VME controllers.

Appendix B Powerup Flowchart

Describes how the system powers up. This appendix includes flowcharts on powerup, reset, initialization, and Programmable Read−Only Memory (PROM)−resident tests.

Appendix C System Board Connectors

Illustrates the system board and describes the pin connections of the VMEbus and PExbus connectors, the serial I/O connector, and parallel printer port.

Related Data General Manuals

This section lists the documents referred to in the text of this manual. For a complete listing of AViiON documentation, see:

Guide to AViiON® and DG/UX™ System Documentation (069−701085)

Hardware Manuals

Setting Up, Starting, Expanding, and Maintaining AViiON® 530 and AViiON® 4600 Computers (014−002091)

Describes how to unpack and connect system components and optional devices. Explains how to power up the system, run diagnostics, and prepare for your operating system installation. Includes operational, physical, electrical, and environmental specifications for the computer.

Software Manuals

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. Aimed at system administrators who are familiar with the UNIX operating system.

Writing a Device Driver for the DG/UX™ System (093−701053)

Describes how to write your own device driver for a DG/UX system running on an AViiON computer. Under the AViiON architecture, drivers must be written to address either a specific device or an adapter that manages secondary bus access to specific devices. This manual address both types of driver.

Other Related Documents

This section lists other related documents, which are not available from Data General Corporation.

MC88100 RISC Microprocessor User's Manual (Motorola)

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 Unit0 User's Manual (Motorola)

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).

The VMEbus Specification (Motorola document number HB212)

Defines the mechanical and electrical specifications, protocols, and terminology of the Versa Modula Europa bus (VMEbus). This interface is used to interconnect data processing, data storage, and peripheral control devices in a closely−coupled hardware configuration.

Binary Compatibility Standard (BCS) (88open Consortium, Ltd.)

Establishes the standards needed to develop operating systems that promote portability of application code between operating systems.

SCC2692 Dual Asynchronous Receiver/Transmitter (DUART) (Signetics)

Describes the DUART used in the asynchronous serial interface.

SCN2661 Enhanced Programmable Communications Interface (EPCI) (Signetics)

Describes the EPCI used in the keyboard interface.

SCN68562 Dual Universal Serial Communications Controller (DUSCC) (Signetics)

Describes the DUSCC used in the synchronous serial interface.

AM79C900 Integrated Local Area Communications Controller™ (ILACC™) (Advanced Micro Devices)

Describes the LAN controller and its registers.

AM29C660 CMOS Cascadable 32-Bit Error Detection and Correction Circuit (Advanced Micro Devices)

Describes the ECC controller and its registers.

NCR 53C700 Data Manual

Describes the SCSI and its registers.

NCR 53C700 Programmer's Guide

Describes how to program and use the SCSI SCRIPTS machine language.

- Birrell, Andrew D., and Nelson, Bruce Jay. Implementing Remote Procedure Calls. XEROX CSL−83−7, October 1983.
- *CCITT Recommendation X.25.* You can obtain copies of this document from the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA, 22161.

Symbols and Conventions

The following symbol and conventions are used in this manual:

Symbol	Meaning
*	A signal name followed by an asterisk (*) indicates that the signal is asserted Low.

Convention	Meaning
SS[3−0]	A signal mnemonic followed by a pair of numbers within braces designates which lines or bits are discussed. In this example, SS[3−0] refers to the System Status lines 0 through 3.

All system addresses are hexadecimal unless otherwise noted with a subscript.

All data within bit diagrams is binary (i.e. the bit diagrams contain binary 1s and 0s).

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.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1−800−DG−HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

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 1−800−253−3902 or 1−508−443−3330.

End of Preface

Contents

Chapter 1 – System Board Architecture

Chapter 2 – Addressing and DMA

Chapter 3 – Interrupts

Chapter 4 – Global Resources

Chapter 5 – Memory

Chapter 6 – Programming the Color Graphics Subsystem

Chapter 7 – Programming the Keyboard Interface

Chapter 8 – Programming the Serial and Parallel Interfaces

Chapter 9 – Programming the LAN and SCSI Interfaces

Chapter 10 – Programming the Real-Time Clock and the Programmable Interval Timer

Chapter 11 – The System Control Monitor (SCM)

Appendix A – Address Map

Appendix B – Powerup Flowchart

Appendix C – System Board Connectors

Tables

Table

Table

Figures

Figure

Figure

Chapter 1
System Board Architecture

AViiON® 4600 and 530 series computers are industry−standard open systems based on the Motorola Reduced Instruction Set Computing (RISC) architecture. They host Data General's DG/UX™ operating system, as well as other Motorola 88000−based operating systems. The systems can be configured either as time−sharing systems supporting large numbers of user devices, or as network servers.

This chapter describes the system board architecture of AViiON 4600 and 530 series computers, covering the following major topics: central processing, memory including cache coherency and error checking and correction (ECC), input/output (I/O), registers, address decoding, and buses.

The system board and related boards have the following functional components:

- CPU boards
- Memory boards
- Registers
- Address decode logic
- Bus arbitration logic
- I/O controllers and I/O boards

Figure 1−1 illustrates the system board architecture, and descriptions follow.

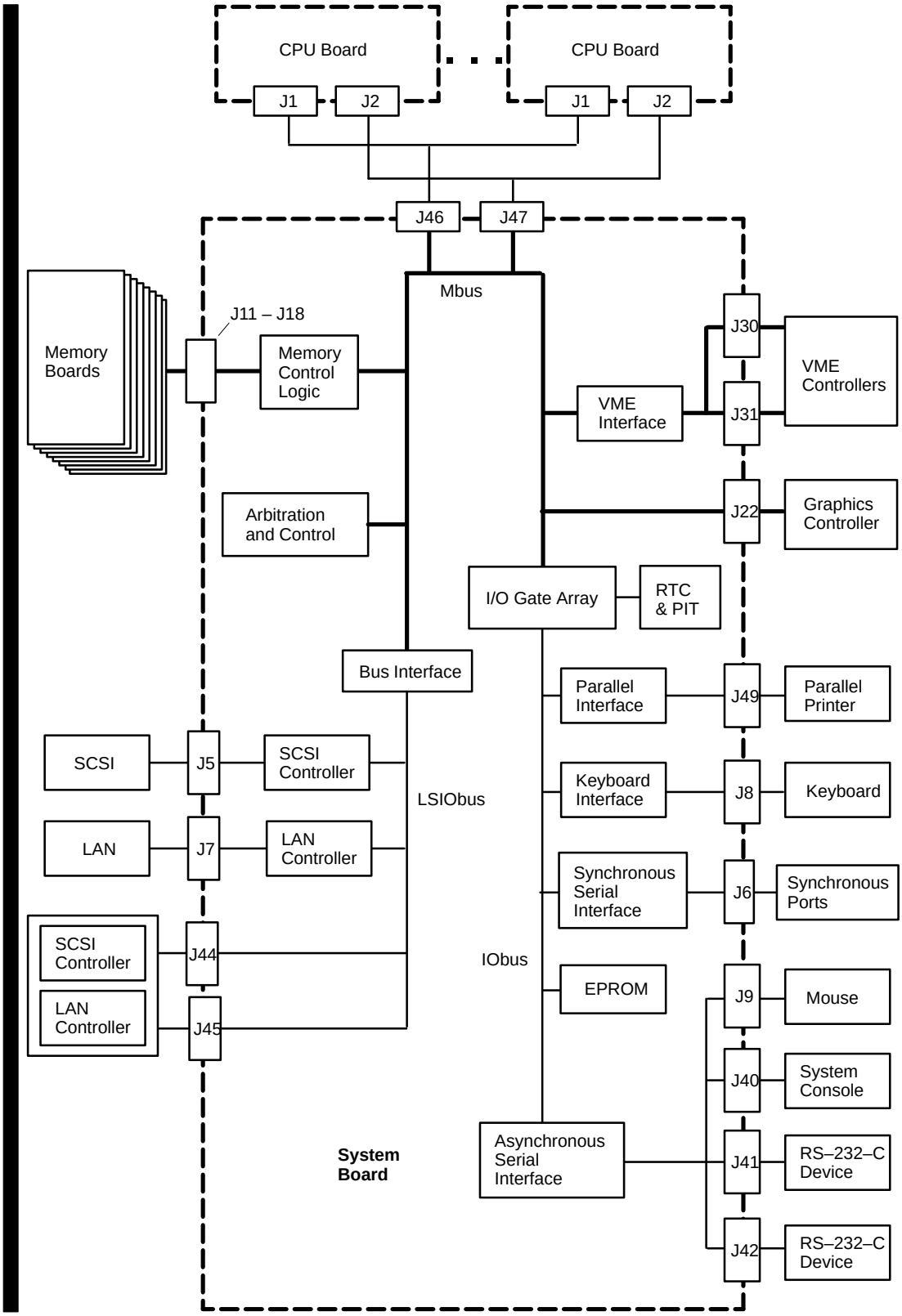


Figure 1–1 Architecture

The CPU Board

The system uses Motorola MC88100 CPUs and MC88200 CMMUs that implement the Reduced Instruction Set Computing (RISC) architecture consisting of 51 instructions. Hardware directly executes the RISC instructions, eliminating the need for microcode. The system board executes most instructions in one bus cycle. See the *MC88100 RISC Microprocessor User's Manual* for information on the CPU and the instruction set. The CPU and CMMUs run at 33 MHz.

Figure 1−2 illustrates the CPU board architecture.

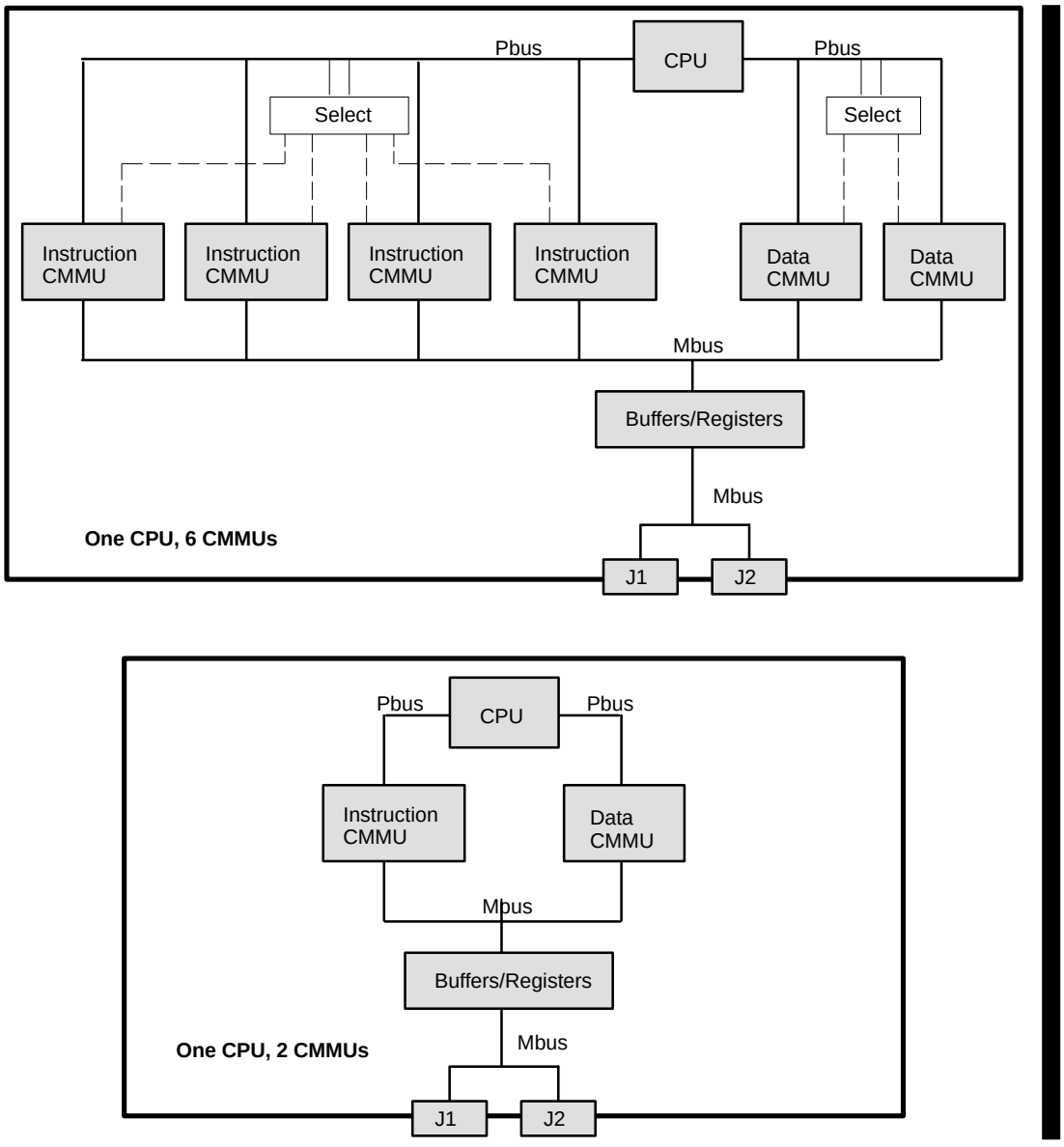


Figure 1–2 Available CPU Board Configurations

Each CMMU contains 16 Kbytes of cache memory and a memory management unit. The memory management unit translates logical addresses into physical addresses within two logical address ranges (User and Supervisor) of 4 Gbytes each. Two address translation caches, the Page Address Translation Cache (PATC) and the Block Address Translation Cache (BATC), provide logical−to−physical address translation.

Each CPU communicates with its CMMUs via two high−speed synchronous Processor buses (Pbus), one for the Instruction CMMU (Instruction Pbus) and one for the Data CMMU (Data Pbus). The Pbus is described in detail in the following manuals by Motorola:

MC88100 RISC Microprocessor User's Manual
MC88200 Cache/Memory Management Unit User's Manual

Some of the signals to/from the CPU are not used as described in Table 1−1.

Table 1–1 Unused CPU Signals

Signal	Description
Interrupt and Control Signals	
ERR	Checker Mismatch Error (held high)
PCE	Pbus Checker Enable (held low)
MCE	Mbus Checker Enable (held low)
Miscellaneous CPU Signals	
TR1	Trace 1 (held low), see note.
TM1	Tag Monitor 1 (held high), see note.
SRAMMODE	Cache Static RAM Mode (held low)
Pbus Signals	
$\overline{\text{PCS}}$	Pbus Chip Select (held low), see note.
Mbus Signals	
ADP[3−0]	Address/Data Parity (held high)
CP	Control Parity (held high)

NOTE: TR1, TM1 and $\overline{\text{PCS}}$ *are used* on CPU boards with six CMMUs per CPU.

Memory

The system supports as much as 128 MBytes of Dynamic RAM (DRAM) located on plug−in single−inline memory modules (SIMMs). The system board contains Error Checking and Correction (ECC) logic to check data passed to and from the RAM.

The system board has 512 KBytes of Programmable Read−Only Memory (PROM) which contains boot code and diagnostics.

Registers

The system board has memory−mapped registers used to analyze and control system functions. These registers are described within the following chapters:

- VME addressing registers. See 2, “Addressing and DMA”.
- DMA registers. See Chapter 2, “Addressing and DMA”.
- Interrupt status and control registers. See Chapter 3, “Interrupts”.
- Global registers. See Chapter 4,“Global Resources”.
- Memory control and status registers. See Chapter 5, “Memory”.
- Color graphics registers. These registers are located on the color graphics board which plugs into the system board. See Chapter 6, “Programming the Color Graphics Subsystem”.
- Keyboard interface registers. See Chapter 7, “Programming the Keyboard Interface and Speaker”.
- Serial Interface Registers. See Chapter 8, “Programming the Serial and Parallel Interfaces”.
- LAN registers. The LAN registers are located on a LAN board which plugs into the system board. See Chapter 9, “Programming the LAN and SCSI Interfaces”.
- SCSI registers. The SCSI registers are located on the SCSI board which plugs into the system board. See Chapter 9, “Programming the LAN and SCSI Interfaces”.
- RTC and PIT registers. The RTC and PIT registers are located on the system board. See Chapter 10, “Programming the Real-Time Clock and the Programmable Interval Timer”.
- SCM registers. The SCM registers are located in firmware on the system board. See Chapter 11, “The System Control Monitor (SCM)”.

The CPU may access all of these registers, but VME controllers can access only the Global Control and Status (GCS) registers.

Input/Output

The system board communicates with other controllers and devices via the VME interface, serial interface, parallel interface, SCSI interface and LAN interface. These interfaces are described below and later in the book as indicated.

Asynchronous Serial Interface

The asynchronous serial interface consists of two Signetics SCC2692 Dual Universal Asynchronous Receiver/Transmitters (DUART), which have memory−mapped registers. The asynchronous serial interface generates one system console port and two modem ports. See Chapter 8, “Programming the Serial and Parallel Interfaces,” for more information on the asynchronous serial interface.

Synchronous Serial Interface

The synchronous serial interface consists of a SCN68562 Dual Universal Synchronous Communications Controller (DUSCC), which has memory−mapped programmable registers. The synchronous serial interface generates two serial ports to transmit and receive data. See Chapter 8, “Programming the Serial and Parallel Interfaces,” for more information on the synchronous serial interface.

Parallel Interface

The parallel interface is a Centronics-compatible printer interface; it can only transmit data; it cannot recieve data. The parallel interface can be configured for either PIO or DMA tansfers, and has one control/status register, and one data register, and DMA registers. There are two interrupts associated with this interface.. See Chapter 8, “Programming the Serial and Parallel Interfaces,” for more information.

VME Interface

The VME interface regulates VMEbus activity and enables system board and VME controllers to communicate over the VMEbus. The interface has memory−mapped programmable registers for VMEbus status and control. The VMEbus is described later in this chapter, in Chapter 2, “Addressing and DMA”, in Chapter 3, “Interrupts”, and in Chapter 4, “Global Resources”.

Small Computer System Interface (SCSI)

The SCSI connects to an ANSI−standard SCSI bus. The SCSI controller has built−in 32-bit DMA.

Local Area Network (LAN) Interface

The Ethernet LAN interface consists of an Ethernet controller with built−in 32−bit DMA, and a serial interface.

Address Decoding

All of the system address space is mapped to regulate access to the system address space. The Mbus uses fixed memory mapping that is decoded by address decode logic. Accesses to system board resources by VME controllers are regulated by a programmable VMEbus Address Decoder (VAD). Figure 1−3 illustrates address decoding and the decode outputs. Chapter 2, "Addressing," describes address decoding in greater detail.

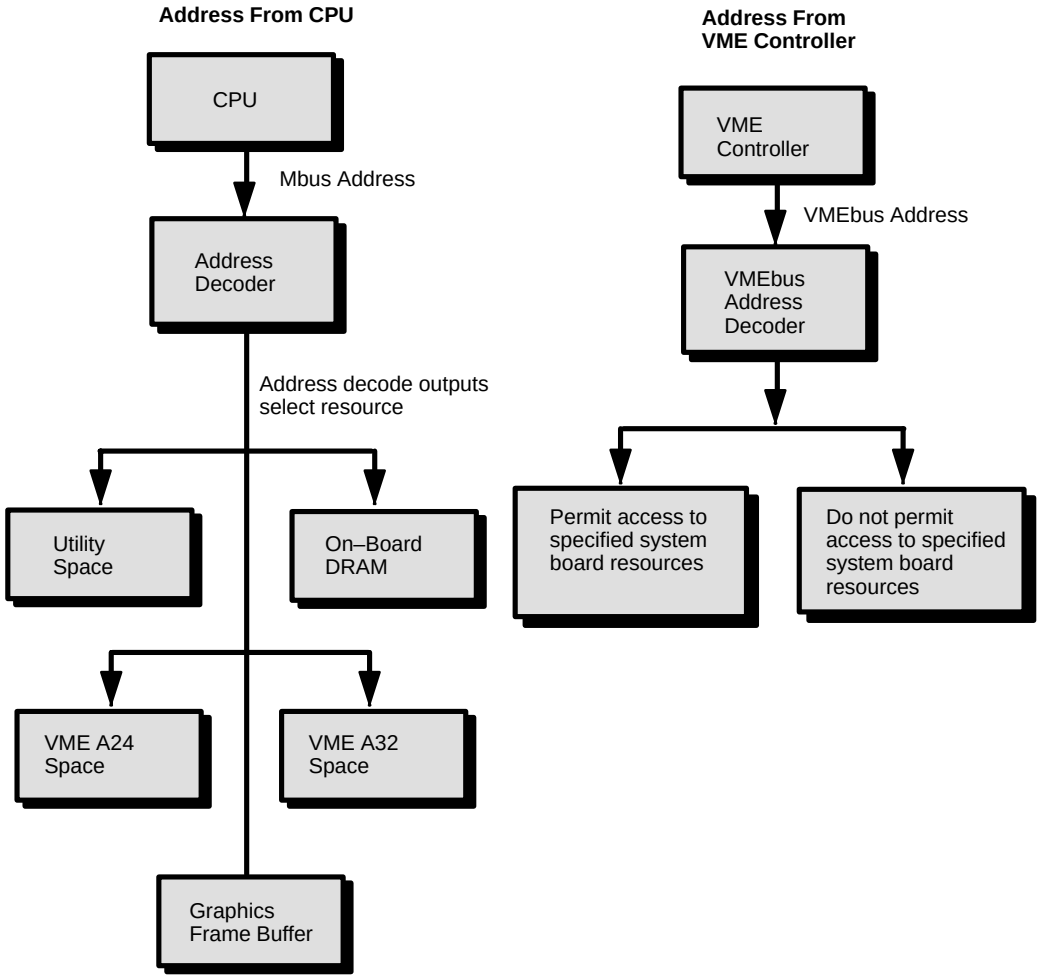


Figure 1–3 Address Decoding

Buses

The system board contains the following buses and bus interfaces:

- The Mbus supports the CMMUs, memory, control logic, VME interface, graphics controller, and the I/O gate array.
- The IObus supports the serial interface, parallel interface, , keyboard interface, and EPROM.
- The LSIObus supports the SCSI and LAN interface.
- The VMEbus supports the VME controllers.

The Mbus and IObus use *big−endian* byte ordering.

The Pbus

The processor bus (Pbus), which connects the CMMUs to the CPU, is described in the *MC88200 Cache/Memory Management Unit User's Manual*. CPU boards with one CMMU per Pbus do not use the $\overline{\text{PCS}}$ (Pbus Chip Select) input of the CMMUs. The single-CMMU configurations hold $\overline{\text{PCS}}$ Low. In configurations that have two or more CMMUs per Pbus, demultiplexed address bits select the CMMU.

The Mbus

The Mbus connects the CMMUs memory, control logic, graphics controller, LSIObus, and I/O gate array.

The Mbus and its signals are described in detail in the *MC88200 Cache/Memory Management Unit User's Manual*. Table 1−2 identifies the unused Mbus signals.

Table 1–2 Unused Mbus Signals

Signal	Description
ADP[3−0]	Address/Data Parity (held high)
CP	Control Parity (held high)

The system board has Mbus bus arbitration logic to regulate Mbus accesses and prevent bus contention.

The Mbus uses *big−endian* byte ordering as described in the *MC88100 RISC Microprocessor User's Manual*.

The IObus

The IObus supports a subset of Intel's 80386 capabilities at 33 MHz. The IObus is clocked with the 33 MHz system clock, but runs at 16.67 MHz.

An IObus master must assert an address before each and every data transfer; the IObus does not support cache−line (burst) transfers to or from system memory.

LSIObus Arbitration

LSIObus arbitration is performed by the I/O Interface in conjunction with the Mbus arbitration logic. When an I/O device asserts a bus request, the I/O interface translates this to an Mbus request. When the Mbus arbitration logic grants the request, the I/O interface passes the grant to the requesting I/O controller.

All masters but LAN0 and LAN1 perform fairness arbitration. Because of LAN latency constraints, the LAN controllers request the bus without fairness.

There is no preemption on the LSIObus. When the arbiter has granted the bus to an I/O controller, the controller owns the bus until it relinquishes the bus of it's own accord. All LSIObus masters must release the bus after 12.8 usec because of LAN latency. All bus masters which are currently expected to be used on the LSIObus will release the bus after a maximum of 16 long word data transfers. If a master cycle on the LSIObus takes more than 15.36 usec to end, a bus error will be generated and the I/O interface will terminate the master access.

The LSIObus uses *little−endian* byte ordering.

The VMEbus

The Versa Modula Europa bus (VMEbus) links the VME controllers and the system board to each other. For example, VME controllers connect to memory through the VMEbus and Mbus.

A VME interface, located on the system board, links the VMEbus and Mbus to each other. The VME interface includes bus arbitration logic and address decode logic. The system board has slot 1 VMEbus functionality, therefore the system board's VME interface monitors the ACFAIL line, handles bus arbitration and VME interrupts, and supplies the 16 MHz VME clock.

Bus arbitration logic handles traffic on the VMEbus, using the VMEbus arbitration signals described in Table 1−3 below. The interface includes logic which times−out accesses that take too long.

Table 1−3 lists the data transfer signals, bus arbitration signals, interrupt signals, failure signals and clocks contained in the VMEbus.

Table 1–3 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:**	
$\overline{\text{DS}}$[1, 0]	Data strobe	$\overline{\text{ACFAIL}}$	Ac failure
$\overline{\text{AS}}$	Address strobe	$\overline{\text{SYSFAIL}}$	System failure
$\overline{\text{LWORD}}$	Long word	$\overline{\text{SYSRESET}}$	System reset
$\overline{\text{SERDAT}}$	Serial data	**Interrupts:**	
$\overline{\text{WRITE}}$	Write	$\overline{\text{IRQ}}$[7−1]	Interrupt request
$\overline{\text{DTACK}}$	Data transfer	$\overline{\text{IACK}}$	Interrupt acknowledge
	acknowledge	$\overline{\text{IACKIN}}$	Interrupt acknowledge in
Bus Arbitration:		$\overline{\text{IACKOUT}}$	Interrupt acknowledge out
$\overline{\text{BR}}$[3−0]	Bus request	**Power:**	
$\overline{\text{BG}}$[3−0]$\overline{\text{IN}}$	Bus grant in	+5V	+5 V dc
$\overline{\text{BG}}$[3−0]$\overline{\text{OUT}}$	Bus grant out	+12V	+12 V dc
$\overline{\text{BBSY}}$	Bus busy	GND	Ground
$\overline{\text{BCLR}}$	Bus clear		
$\overline{\text{BERR}}$	Bus error		

Address decode logic, in conjunction with memory maps, determines which accesses across the VME interface are valid. Chapter 2, "Addressing," discusses the memory maps and addressing schemes in detail.

The system board accepts words, half−words, bytes, and blocks of data from VME controllers, but transmits only words, half−words or bytes to VME controllers.

VMEbus Arbitration

The VME controllers, including the system board, communicate through the VMEbus. Before communicating with another controller, a VME controller must request access to the VMEbus through arbitration logic located on the system board. This bus arbitration consists of four bus request and bus grant levels; the VMEbus signals associated with these levels are Bus Request ($\overline{\text{BR[3-0]}}$) and Bus Grant ($\overline{\text{BG[3-0]IN}}$ and $\overline{\text{BG[3-0]OUT}}$). Each bus grant level is daisy−chained from VME controller to VME controller; therefore if both VME boards request at the same level, the board in slot 1 has highest priority and is granted the VMEbus. When it releases the VMEbus, the arbitration process repeats with another arbitration cycle. This daisy−chain configuration is illustrated in Figure 1−4. The Motorola manual, *The VMEbus Specification,* describes these signals in greater detail.

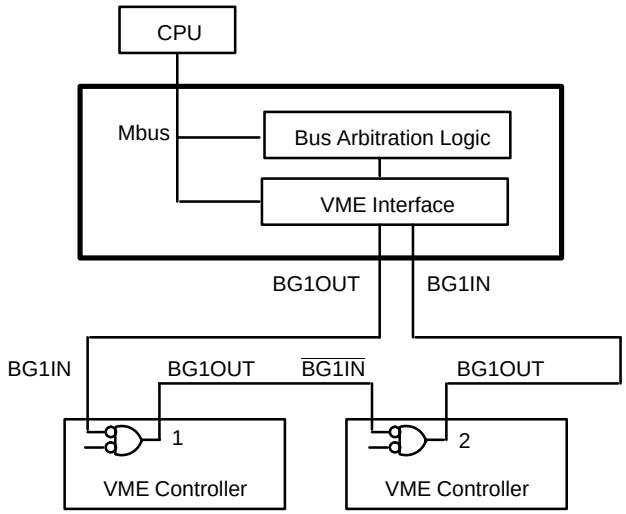


Figure 1–4 VMEbus Grant Daisy–Chain

To request access the VMEbus, write the desired request level into the VMEbus Request Level (VRL) bits of the Utility Control and Status (UCS) register. This asserts the corresponding bus request bit (BR[3−0]) to the VMEbus. If the VMEbus is free and the system board is the highest priority requestor, the system board will pull the Bus Grant ($\overline{\text{BGnOUT}}$) line High and assert the Bus Busy ($\overline{\text{BBSY}}$) line Low. If the VMEbus is busy, i.e. $\overline{\text{BBSY}}$ is asserted, and the system board request has a lower priority, the bus arbiter will wait until the bus is released. When the bus is released, the arbiter will sample the bus lines and grant the request to the highest priority requestor.

VMEbus arbitration is available in two modes: Priority and Round Robin. The mode is selected through the Round Robin (RBN) bit in the Utility Control and Status (UCS) register.

When using the Priority mode, if a controller requests access to the VMEbus while a lower−priority controller is using the bus, the VME arbiter will assert the $\overline{\text{BCLR}}$ signal to request the current VMEbus master to release the VMEbus. Access to the VMEbus is granted to the highest−priority requestor via the Bus Grant ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) lines. Level 3 is the highest priority and level 0 is the lowest priority.

When using the Round Robin mode, the Bus Grant ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) signal is passed through a loop. When a controller receives the bus grant, the controller will either pass the grant on to another controller, or it will use the grant to access the VMEbus. In Round Robin mode, all controllers in the round robin loop have the same priority.

Arbitration Modes

The system board accesses the VMEbus using one of three modes: Release Never, Release When Done (RWD) and Release On Request (ROR). These modes are selected using the Release Never (RNV) and VMEbus Release Mode (VRM) bits in the UCS register. VRM selects either Release When Done or Release on Request. These modes and bits are described in greater detail in Chapter 4, “Global Resources.”

The system board also implements a Fairness mode if the Fairness (FAIR) bit in the UCS register is set. When the FAIR bit is set, the system board will not request the VMEbus if someone else is requesting it from the same level. This passes the bus grant down the bus grant daisy chain ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) to the next board.

Arbitration Timeout

The VMEbus arbiter contains logic to limit bus arbitration delays to one second or less. This timing logic can be enabled or disabled through the Enable VMEbus Arbitration Timeout (ETO) bit of the Utility Control and Status (UCS) register. When enabled, the time−out expires if the BBSY signal is not asserted within one second of asserting the Bus Grant (BG*n*) signal. When a VMEbus arbitration timeout occurs, the interrupt logic sets the Arbitration Timeout (ATO) interrupt in the Interrupt Status (IST) register and asserts an interrupt to the CPU.

Transferring Data Over the VMEbus

The VME interface supports word, half−word and byte transfers initiated by either the VMEbus or the Mbus.

A VMEbus error will occur if either a bus time−out occurs because no device responds, or if a VME controller is accessed and receives an address modifier that it cannot handle (For example, if a controller tries to send a 32−bit data word to a 16−bit controller board, an error will result.)

The VME interface also supports block transfers of words, half−words or bytes initiated by a VME controller. The maximum block transfer initiated by a VME controller is 256 bytes (64 words) while the maximum block transfer on the Mbus is 16 bytes (four words). If a VME controller sends a block larger than four words to memory, the VME interface breaks the block into four−word blocks that the Mbus can handle.

VME−initiated block transfers of consecutive bytes or half−words reduce system performance when compared with full−word data transfers. For byte and half−word data transfers, the memory system performs read−modify−write cycles to merge the new data with the remaining bits of the word and to generate error correction bits. The VME interface provides higher throughput by packing consecutive half−words or bytes, wherever possible, into 32−bit words prior to Mbus transfer.

End of Chapter

Chapter 2
Addressing and DMA

This chapter discusses how to address the registers, I/O and memory in your system. It describes address maps and address decoding. The chapter also explains how a CPU accesses system board resources and memory, how the CPU accesses other VME controllers, and how other VME controllers access memory and the Global Control and Status (GCS) registers.

The system address space is fixed in hardware. Address decode logic decodes accesses to all of the system resources as defined in Appendix A, “Address Map.” This address decode process is transparent to the programmer.

Addressing a VME Controller

This section describes how a system board CPU addresses VME controllers. This addressing process is transparent to the programmer.

When the CPU addresses a VME controller, address decode logic generates the MDS[3−0] bits, which are translated into the address modifier bits AM[5−4]. The address modifier bits inform the VME controllers which VME space the address is written to: A16, A24 or A32 space. See Table 2−1 in the description of the EXTAM register for more information about the address modifier bits.

When the CPU addresses a VME controller, the system board decodes the address as follows: (see also Figure 2−1)

1. The CPU puts a 32−bit address onto the Mbus.
2. Address decode logic decodes the address. If it points to the VMEbus, the following steps are performed simultaneously.
3. The address bits A[31−0] from the Mbus are placed on the VMEbus.
4. The address modifier bits AM[5−0] are placed on the VMEbus.

Figure 2−1 illustrates how the VMEbus decodes addresses, and where the address modifier bits come from.

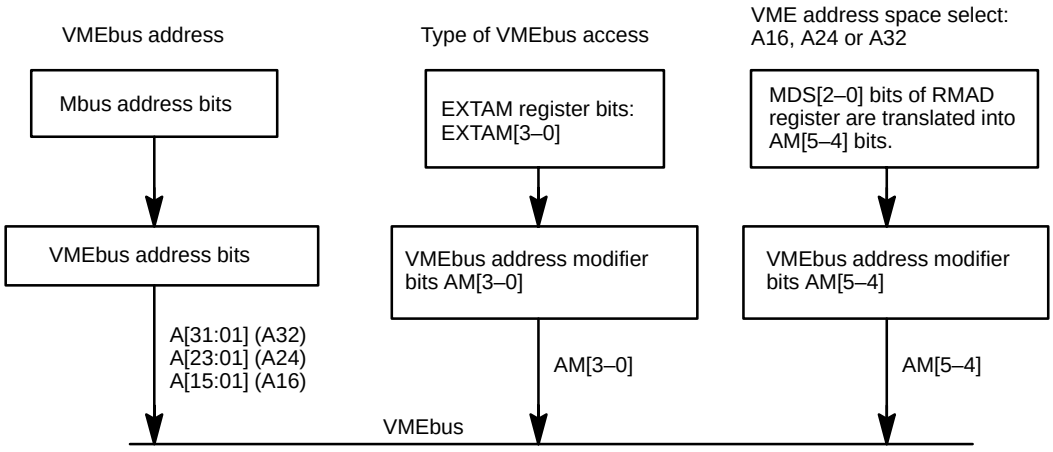


Figure 2–1 Decoding Addresses to the VMEbus

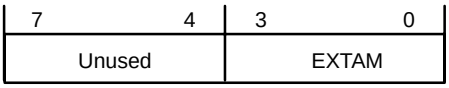
EXTAM Extended Address Modifier

FFF8 8014 Read/write

The extended address−modifier register (EXTAM) supplies the address modifier bits AM[3−0] to the VMEbus when a system board controller accesses the VMEbus. AM[5−0] defines the address size and the type of access, and is driven onto the VMEbus with each address. Table 2−1 defines AM[3−0].

Initialize EXTAM before addressing a VME controller.

EXTAM is not affected by system reset or local reset.



Bit	Mnemonic	Function
7−4	Unused	
3−0	AM[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 2−1, Address Modifiers AM[3−0]).

Table 2−1 defines EXTAM[3−0].

Table 2–1 Address Modifiers AM[3–0]

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 non−priveleged access defined in the VMEbus specification.

In addition to AM[3−0], the system board generates AM[5,4] from address decode signals. AM[5,4] identify the VME address space as defined in Table 2−2.

Table 2–2 Address Modifiers AM[5,4]

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

Addressing Memory and System Board Resources from a VME Controller

This section describes how VME controllers address memory. This addressing process is transparent to the programmer. The following text and illustrations explain how the CPUs access the system board resources and 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; the address range is limited to 64 Kbytes. A16 controllers can access only the Global Control and Status (GCS) registers and other VME controllers.

A24 controllers have 24 address lines; the address range is limited to 16 Mbytes. A24 controllers can access assigned memory. When an A24 controller accesses 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 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.

A32 controllers have 32 address lines; the address range extends throughout the system's space. A32 controllers can access assigned memory and the GCS registers as well as other VME controllers.

VME controllers can access the GCS registers located in utility space. The GRPAD[7−0] and BDAD[3−0] DIP switches define the base address of the GCS registers.

The VME Address Decoders (VAD) process addresses from VME controllers. When a VME controller addresses a system board resource or a location in memory that it has access to, the VAD enables access to the location. If the address points to space that is not allocated to the VME controller, the VAD will not allow access to the resource. This restriction prevents VME controllers from accessing system resources that are not assigned for use by VME controllers.

NOTE: Map only to valid system memory within the first 128 Mbytes from the top of memory.

When a VME controller tries to access system board resources or system memory, a programmable VMEbus Address Decoder (VAD) decodes these addresses to system board resources. The VAD determines whether or not accesses to the system board resources and memory are allowed. Each of two VADs (one for A24 controllers and one for A32 controllers) has 1024 2−bit locations (Mbus Select (MBS) and VME Snoop Enable (VSE)). These 1024 locations correspond to the 1024 4−Mbyte pages. MBS determines whether or not an access by a VME controller is allowed.

The VMEbus Address Decoder (VAD)

VME controllers are allowed access to the system board RAM and global resources through the VMEbus Address Decoder (VAD).

NOTE: Map only to valid system memory within the first 128 Mbytes from the top of memory.

Loading and Verifying the VAD

Power−up code loads and verifies the VAD while the system powers up. The VAD should not be changed after powerup. But, if you need to reload the VAD, load and verify it as follows:

1. Clear (to 0) the VMEbus Address Decoder Valid (VADV) bit of the CPU Control and Status (CCS) register.
2. Write the address and decode parameter to the Write VMEbus Address Decoder (WVAD) register.

   The data you write into WVAD includes one bit to select the VAD (A24 VAD or A32 VAD), ten bits to select one of the 1024 locations in the VAD, and two bits to store the decode value.
3. Verify the decode value in the VAD as follows:
   a. Write the page number into the VPN bits in the RVAD register.
   b. Read the RVAD register. The Mbus Select (MBS) bit defines whether or not the access is valid.
4. Set the VADV bit of the CCS register to 1.

When the VADV bit of the CCS register is cleared to zero, the VAD is turned off. In this state, the VME controllers cannot access the system board resources and memory. When the VADV bit is set to 1, the VAD is turned on to enable VMEbus address decoding.

RVAD Read VMEbus Address Decoder

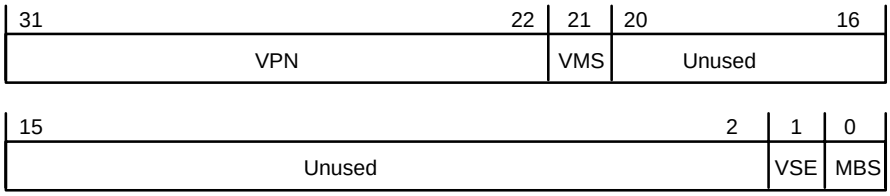
FFF8 802C Read/Write

The Read VMEbus Address Decoder (RVAD) register is used to examine the contents of the VAD. When read, the RVAD register returns the address decode value for the page defined by the VME Page Number (VPN) bits. This value is returned through the Mbus Select (MBS) and VME Snoop Enable (VSE) bits. Before reading RVAD, write the page number and the map select into the VPN and VMS bits of the RVAD register.

System reset and local reset do not affect the RVAD register.

Writing to RVAD does not modify the address decoder map. To modify the map, write to WVAD.

Clear the VMEbus Decoder Valid (VDV) bit in the CPU Control and Status (CCS) register to 0 before reading RVAD.



Bit	Mnemonic	Function
31–22	VPN[9–0]	VME Page Number (one of 1024 pages) Write only. VPN supplies the page number to which the WVAD register will write the page address.
21	VMS	VME Map Select Write only. VMS selects the map to which the page address will be written. 1 Select VME A24 map. 0 Select VME A32 map.
20–2	Unused	
1	VSE	VME Snoop Enable Read only. 1 Snoop is disabled. 0 Snoop is enabled.
0	MBS	Mbus Select Read only. MBS returns the value of MBS from the VME map defined by VMS and the page defined by VPN[9–0]. 1 Indicates that the VME controller cannot access the Mbus at that page. 0 Indicates that the VME controller can access the Mbus at that page.

WVAD Write VMEbus Address Decoder

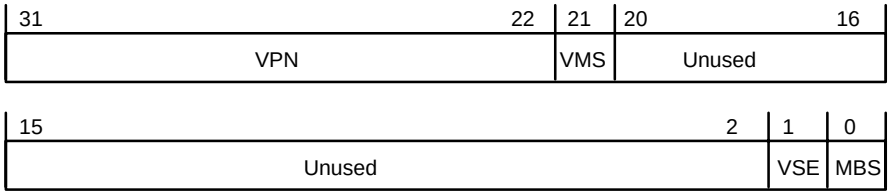
FFF8 8028 Write Only

Write VMEbus Address Decoder (WVAD) loads the VMEbus address decoders. The VMEbus has two decoders: one for the A24 address space and one for the A32 address space.

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

Clear the VMEbus Decoder Valid (VDV) bit in the CPU Control and Status (CCS) register to 0 before writing to WVAD.

CAUTION: Writing to WVAD will change the page address that RVAD accesses. NEVER write to WVAD between writing to and reading from RVAD.



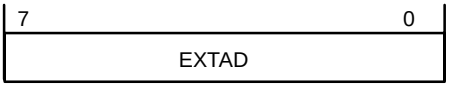
Bit	Mnemonic	Function
31–22	VPN[9–0]	VME Page Number Define the page number within the VME address decoder. The contents of VSE and MBS are written into the appropriate VAD at this page number.
21	VMS	VME Map Select Selects the VME address decoder to be written to. 1 Select the VME A24 map. 0 Select the VME A32 map.
20–2	Unused	
1	VSE	VME Snoop Enable Requests the master CMMU to snoop the VMEbus for a valid Mbus address. 1 Disable snooping. 0 Enable snooping.
0	MBS	Mbus Select Defines whether the VMEbus has access to the Mbus at the page number defined by the VPN bits. 1 Prevent VME access to the Mbus at that page number. 0 Allow VME access to the Mbus at that page number.

EXTAD Extended Address

FFF8 8010 Read/write

The extended address register provides the upper 8 Mbus address bits when an A24 VMEbus device accesses the Mbus. EXTAD is loaded during powerup with a base address for VME access to memory.

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



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

Figure 2–2 illustrates how EXTAD fits in a VME A24 address.

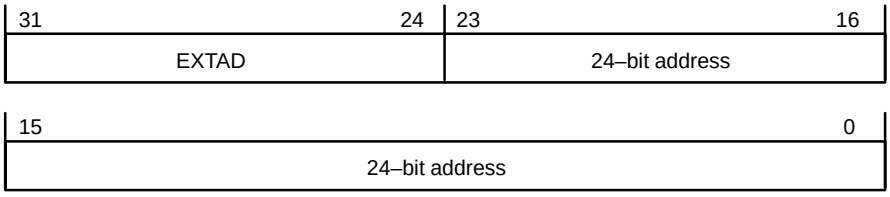


Figure 2–2 EXTAD in a VME A24 Address

Figure 2−3 illustrates how VME controllers address system board resources and memory.

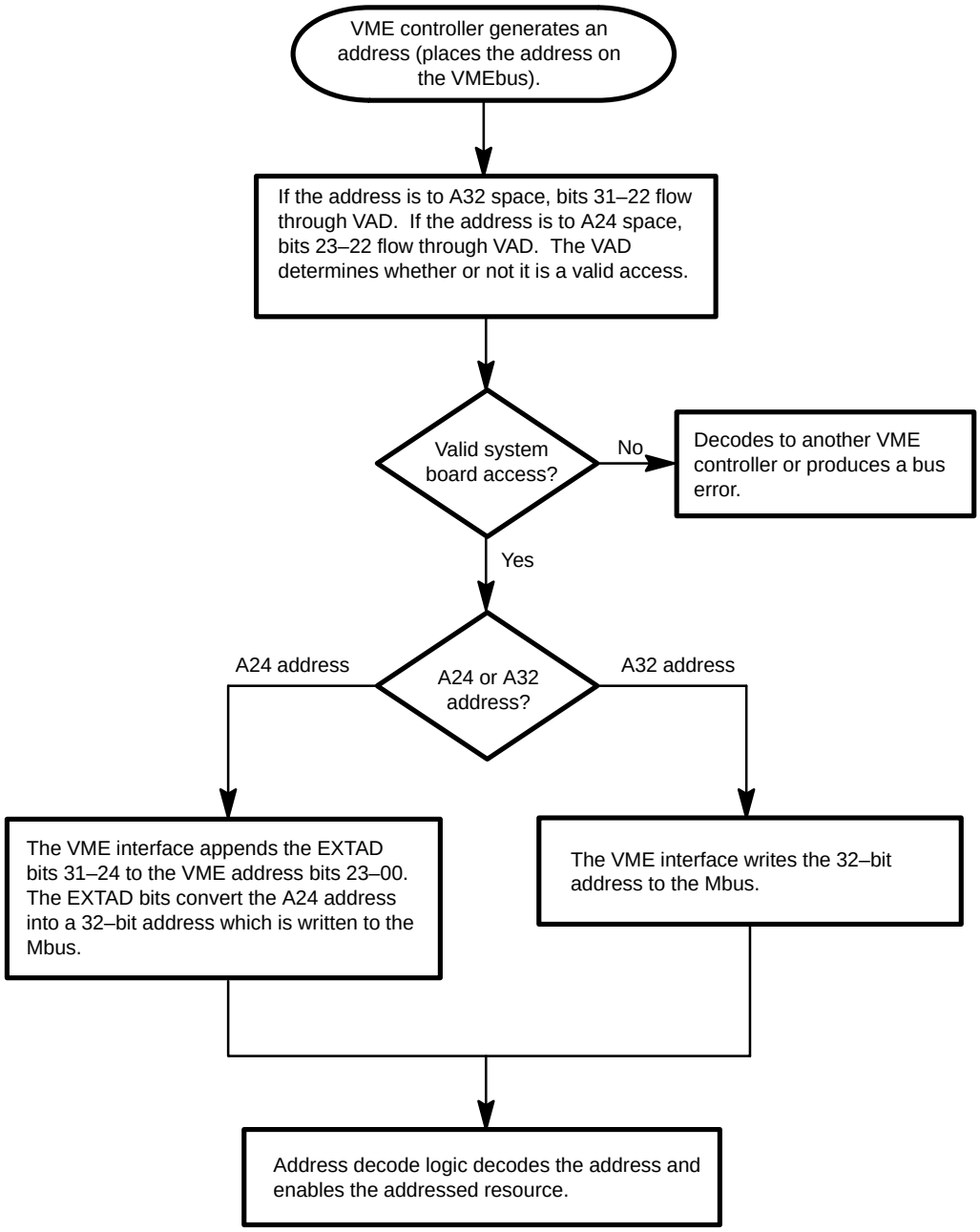


Figure 2–3 Addressing System Board Resources and memory from the VMEbus (Flowchart)

DMA

The system has six direct memory access (DMA) channels (DMA0 − DMA5). The synchronous serial interface and the parallel interface use these DMA channels. Currently, channels 0− 4 are used; channel 5 is reserved for future use. The channels are used as follows:

0 Synchronous port A receive
1 Synchronous port A transmit
2 Synchronous port B receive
3 Synchronous port B transmit
4 Printer
5 Unused

Each DMA channel has the following programmable registers:

- DMA Address (AD)
- DMA Command (CMD)
- DMA Byte Count (CNT)
- DMA Scatter/Gather Descriptor Address (SG)
- DMA Status (STAT)

The rest of the chapter describes the DMA registers.

DMA Channel 0 and 2 Writes to Memory

Scatter/gather is not available on channel 0 or 2.

When writing a block of serial data to memory via DMA channel 0 or channel 2, occasionally the last byte may not be transmitted. To check for and correct this condition, do the following:

1. Check the byte count. If the difference between the original programmed byte count and the current byte count is not a multiple of 4, continue with the following steps. If the difference is a multiple of 4, disregard the following steps; all of the data was written to memory.
2. Set the DMA Byte Count (CNT) to 1.
3. Clear (to 0) the appropriate DMA flush bit (FDMA0 or FDMA3) of the Memory Diagnostic Control (MDC) register (see Chapter 5, “Memory”). This will cause the DMA controller to write the remaining byte to memory, completing the data transfer.
4. Set the DMA flush bit to 1.

AD — DMA Address

FFF8 D808	**AD** – DMA0	**Read/Write**
FFF8 D908	**AD** – DMA1	
FFF8 DA08	**AD** – DMA2	
FFF8 DB08	**AD** – DMA3	
FFF8 DC08	**AD** – DMA4	

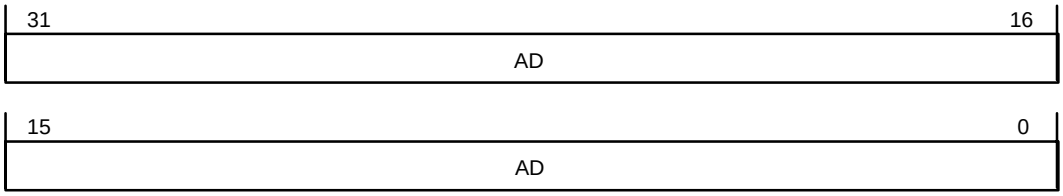
The DMA Address (AD) register contains the address of the data buffer located in system memory. AD is loaded by the I/O gate array in Scatter/Gather Mode or by software in Non−Scatter/Gather Mode:

Scatter/Gather Mode

The SG register points to a descriptor which the io gate array loads into the AD register. Software must not write to the AD register in scatter/gather mode.

Non–Scatter/Gather Mode

Software writes to the AD register directly.



Bit	Mnemonic	Function
31−0	AD	DMA Address When read, AD contains the address of the data buffer last received or transmitted. When written to, AD points to the data buffer to be transmitted.

CMD — DMA Command

FFF8 D820 **CMD** – DMA0 **Write Only**
FFF8 D920 **CMD** – DMA1
FFF8 DA20 **CMD** – DMA2
FFF8 DB20 **CMD** – DMA3
FFF8 DC20 **CMD** – DMA4

The Command (CMD) register controls the operation of the DMA channel.

15 – 9	8	7	6	5	4	3	2	1	0
Unused – all 0	MSK2	MSK1	MSK0	0	SE	SG	RST	DIR	STR

Bit	Mnemonic	Function
15–9	Unused	Should be all 0.
8	MSK2	Mask Bit 2 When MSK2 is set to 1, an external interrupt will terminate the DMA transfer.
7	MSK1	Mask Bit 1 When MSK1 is set to 1, an Mbus parity error will terminate the DMA transfer.
6	MSK0	Mask Bit 0 When MSK0 is set to 1, an Mbus error will terminate the DMA transfer.
5	FLSH	FIFO Flush When FIFO is set to 1, the contents of a 4–byte data FIFO are flushed to memory. The DMA operation is not affected. After the flush completes, clear FLSH to 0 to disable further flushing.
4	SE	Snoop Enable Writing a 0 to SE marks the DMA transfers as “snoopable” by the CMMU’s translation descriptor. For more information, see the *MC88200 User’s Manual,.*
3	SG	Scatter/Gather Defines the type of DMA. 0 Non scatter/gather. 1 Scatter/gather.
2	RST	DMA Reset Writing a 1 to RST resets the DMA and holds the reset until RST is cleared to 0. Monitor the DIP bit in the Status register; when it goes to 0, the reset is complete and RST can be cleared to 0. During powerup, RST is cleared to 0.
1	DIR	DMA Direction Defines the direction of the DMA. 0 Write to memory. 1 Read from memory
0	STR	DMA Start Writing a 1 to STR starts a DMA operation. Make sure all other registers are programmed before starting the DMA. Also, do not write a 0 to STR while a DMA is active.

CNT — DMA Byte Count

FFF8 D810	**CNT** – DMA0	**Read/Write**
FFF8 D910	**CNT** – DMA1	
FFF8 DA10	**CNT** – DMA2	
FFF8 DB10	**CNT** – DMA3	
FFF8 DC10	**CNT** – DMA4	

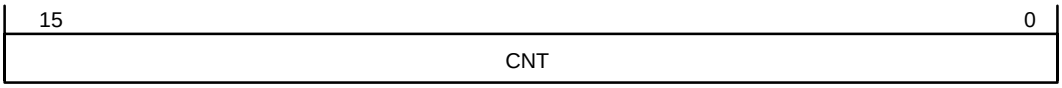
The DMA Byte Count (CNT) register stores the byte count of the DMA transfer. The count is limited to 16 bits (i.e. maximum transfer = 64 KB). Like AD, CNT is loaded in either Scatter/Gather Mode or in Non−Scatter/Gather Mode:

Scatter/Gather Mode

The SG register points to a descriptor which the io gate array loads into the CNT register. Software must not write to the CNT register in scatter/gather mode.

Non–Scatter/Gather Mode

Software writes to the CNT register directly.



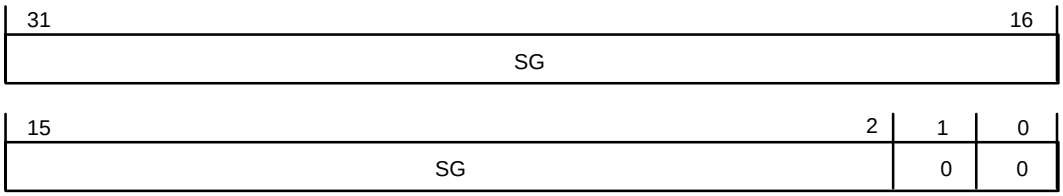
Bit	Mnemonic	Function
15−0	CNT	DMA Byte Count When read, defines the number of bytes in the most recent DMA transfer. When written to, defines the number of bytes to be transmitted in the next DMA transaction.

SG Scatter/Gather Descriptor Address

FFF8 D804	**SG** – DMA0	**Read/Write**
FFF8 D904	**SG** – DMA1	
FFF8 DA04	**SG** – DMA2	
FFF8 DB04	**SG** – DMA3	
FFF8 DC04	**SG** – DMA4	

The Scatter/Gather Descriptor Address (SG) register contains the address of the scatter/gather descriptor block. During a scatter/gather DMA operation, SG increments to point to the next entry in the scatter/gather descriptor block, which points to the next data address and byte count. Each entry to SG must be aligned on a word boundary; hence bits 0 and 1 must be 0.

SG is not available on either DMA channel 0 or 2.



Bit	Mnemonic	Function
31–2	SG	Scatter/Gather Descriptor Address Address of the next scatter/gather descriptor.
1, 0	0	Must be 0 to align the descriptor address on a word boundary.

STAT — DMA Status

FFF8 D840 **STAT** – DMA0 **Read/Write**

FFF8 D940 **STAT** – DMA1

FFF8 DA40 **STAT** – DMA2

FFF8 DB40 **STAT** – DMA3

FFF8 DC40 **STAT** – DMA4

The Status (STAT) register reflects the status of the DMA channel.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SGERR	IDI	PERR	BERR	TRM	CMP	FLSG	MSK2	MSK1	MSK0	Res	SE	SG	RST	DIR	DIP

Bit	Mnemonic	Function
15	SGERR	Scatter/Gather Error When set to 1, indicates that a bus error or parity error occurred during a scatter/gather fetch. SGERR is automatically cleared when PERR or BERR are cleared.
14	IDI	IObus Device Interrupt When set to 1, indicates that a device on the IObus asserted an interrupt.
13	PERR	Parity Error When set to 1, indicates that an Mbus parity error occurred.
12	BERR	Bus Error When set to 1, indicates that an Mbus error occurred.
11	TRM	DMA Terminated When set to 1, indicates that the current DMA operation either could not complete, or was terminated by a DMA reset.
10	CMP	DMA Complete When set to 1, indicates that the current DMA operation is complete.
9	FLSG	Fetch Last Scatter/Gather When set to 1, indicates that the last scatter/gather descriptor was fetched.
8	MSK2	Mask 2 Indicates that a DMA transfer was terminated by an external interrupt.
7	MSK1	Mask 1 Indicates that a DMA transfer was terminated by an Mbus parity error.
6	MSK0	Mask 0 Indicates that a DMA transfer was terminated by an Mbus error.
5	FLSH	Flush A 1 indicates that the contents of the 4−byte data FIFO has been flushed into memory.
4	SE	Snoop Enable A 1 indicates that the DMA transfers are marked as “snoopable” by the CMMU’s translation descriptor. For more information, see the *MC88200 User’s Manual,*.

(continued)

Bit	Mnemonic	Function
3	SG	Scatter/Gather 0 Indicates that the DMA is non−scatter/gather. 1 Indicates that the DMA is scatter/gather.
2	RST	DMA Reset A 1 indicates that a DMA reset has been asserted.
1	DIR	DMA Direction Indicates the DMA transfer direction. 0 Indicates that the data will be written to memory from the DMA controller. 1 Indicates that the data will be read from memory to the DMA controller.
0	DIP	DMA In Progress 0 Indicates that a DMA transfer or DMA reset is complete. 1 Indicates that a DMA transfer or DMA reset is active.

(concluded)

End of Chapter

Chapter 3
Interrupts

Interrupts are a means for various system resources (memory, system board I/O controllers, VME controllers, power supply...) to notify the system board CPU of a condition that needs attention.

This chapter discusses interrupts, how to interrupt the CPU, and how to process interrupts.

This chapter includes the following sections:

- CPU interrupt registers
- VME Interrupts, which includes interrupts both from and to the VME controllers
- Interrupt logic
- Handling interrupts

CPU Interrupt Registers

This section describes the registers that pass interrupts to the CPU, and how the interrupt requests are passed to the CPU.

Register	Address
IEN0	FFF8 4004
IEN1	FFF8 4008
IEN2	FFF8 4010
IEN3	FFF8 4020
IEN	FFF8 403C
IST	FFF8 4040
SETSWI	FFF8 4080
CLRSWI	FFF8 4084
ISS (ISTATE)	FFF8 4088
CLRINT	FFF8 408C
Extended interrupts	
EXIEN0	FFF8 E004
EXIEN1	FFF8 E008
EXIEN2	FFF8 E010
EXIEN3	FFF8 E020
EXIEN	FFF8 E03C
EXIST	FFF8 E040

IST Interrupt Status

FFF8 4040 Read only

The Interrupt Status (IST) register contains interrupt flags that are set to 1 by devices requesting an interrupt to the CPU. When a device requests an interrupt, the interrupt request signal goes to interrupt logic which sets the corresponding interrupt bit in the IST register. The interrupt logic also interrupts the CPU. When interrupted, the CPU must execute a primary interrupt service routine to find out what kind of interrupt it is. The interrupt service routine may read an interrupt enable register as well as the interrupt status register. If it does read an interrupt enable register, it must compare the two registers to find any enabled interrupt(s). The bits in the interrupt enable registers and the interrupt status register are mirror images of each other.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	DTI	SI7	SI6	SI5	SI4	VME7	KBD	Unused	SF	VME6	MEM	DI	SHP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused	VME5	Unused	VME4	Unused	VME3	Unused	LM	SLP	VME2	Unused	VME1	SI3	SI2	SI1	SI0

Bit	Mnemonic	Function
31	ABT	Abort Pushbutton The operating system can use this to initiate a software reset. 1 Indicates that the abort switch was pressed. 0 Indicates that the abort switch was not pressed.
30	ACF	AC Failure 1 Indicates that an ac power failure has occurred. The ac failure signal originates from the power supply, which is connected to the VMEbus. 0 Indicates that there has not been an ac power failure.
29	ATO	VMEbus Arbiter Time Out 1 Indicates that the VMEbus bus grant has timed out and generated an interrupt. 0 Indicates that the VMEbus bus grant has not timed out.
28	DTI	DUART Timer Interrupt (system console) 1 Indicates that the system console DUART is requesting an interrupt. 0 Indicates that the system console DUART is not requesting an interrupt.
27	SI7	Software Interrupt 7 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
26	SI6	Software Interrupt 6 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.

(continued)

Bit	Mnemonic	Function
25	SI5	Software Interrupt 5 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
24	SI4	Software Interrupt 4 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
23	VME7	VME Level 7 Interrupt A level 7 interrupt ($\overline{IRQ7}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ7}$, on the VMEbus, is asserted .0 Indicates that $\overline{IRQ7}$, on the VMEbus, is not asserted.
22	KBD	Keyboard Interrupt Indicates whether or not the keyboard interface has asserted an interrupt request. Read the Status (STS) register of the keyboard interface to determine the source of the interrupt. 1 Indicates that the keyboard interface has asserted an interrupt. 0 Indicates that the keyboard interface has not asserted an interrupt.
21	Unused	
20	SF	System Failure Interrupt 1 Indicates that the system failure ($\overline{SYSFAIL}$) line on the VMEbus was asserted. A system failure occurred. 0 Indicates that the system failure ($\overline{SYSFAIL}$) line on the VMEbus was not asserted. There has not been a system failure.
19	VME6	VME Level 6 Interrupt A level 6 interrupt ($\overline{IRQ6}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ6}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ6}$, on the VMEbus, is not asserted.
18	MEM	Memory Error Interrupt MEM indicates whether or not a memory error occured. 1 Indicates that a memory error occurred. 0 Indicates that a memory error did not occur.
17	DI	DUART Interrupt (system console) 1 Indicates that the system console DUART is requesting an interrupt. 0 Indicates that the system console DUART is not requesting an interrupt.
16	SHP	Signal High Priority (SHP) Interrupt SHI is a condition−specific interrupt to the system board from a VME controller. This interrupt is pre-defined within both the interrupting VME controller and the interrupt service routine of the operating system. 1 A VME controller is requesting that an SHP interrupt be serviced. The VME controller asserts SHI via the GLOBAL1 register. See Chapter 4, “Global Resources.” 0 An SHP interrupt is not being requested.
15	Unused	
14	VME5	VME Level 5 Interrupt A level 5 interrupt ($\overline{IRQ5}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ5}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ5}$, on the VMEbus, is not asserted.

(continued)

Bit	Mnemonic	Function
13	Unused	
12	VME4	VME Level 4 Interrupt A level 4 interrupt ($\overline{IRQ4}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ4}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ4}$, on the VMEbus, is not asserted.
11	Unused	
10	VME3	VME Level 3 Interrupt A level 3 interrupt ($\overline{IRQ3}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ3}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ3}$, on the VMEbus, is not asserted.
9	Unused	
8	LM	Location Monitor Interrupt Indicates whether or not a location monitor interrupt is asserted. 1 A location monitor interrupt is asserted. 0 A location monitor interrupt is not asserted.
7	SLP	Signal Low Priority (SLP) Interrupt SLI is a condition−specific interrupt to the system board from a VME controller. The result of this interrupt is pre-defined within the interrupting VME controller and within the interrupt service routine of the operating system. 1 Indicates that a pre−defined VME controller is requesting that an SLP interrupt be serviced by the operating system's interrupt service routine. The VME controller asserts SLI via the GLOBAL1 register. See Chapter 4, "Global Resources." 0 Indicates that an SLP interrupt is not being requested.
6	VME2	VME Level 2 Interrupt A level 2 interrupt ($\overline{IRQ2}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ2}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ2}$, on the VMEbus, is not asserted.
5	Unused	
4	VME1	VME Level 1 Interrupt A level 1 interrupt ($\overline{IRQ1}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ1}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ1}$, on the VMEbus, is not asserted.
3	SI3	Software Interrupt 3 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
2	SI2	Software Interrupt 2 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
1	SI1	Software Interrupt 1 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
0	SI0	Software Interrupt 0 One of eight software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.

(concluded)

IEN, IEN0, IEN1, IEN2 and IEN3 — Interrupt Enable

FFF8 403C **IEN** **Read/Write**

FFF8 4004 **IEN0**

FFF8 4008 **IEN1**

FFF8 4010 **IEN2**

FFF8 4020 **IEN3**

The Interrupt Enable registers (IEN, IEN0 − IEN3) enable and mask interrupts to the CPUs. IEN0 − IEN3 enable interrupts to the corresponding CPU (CPU0 − CPU3), while IEN enables interrupts to all CPUs. To enable an interrupt, write a 1 into the corresponding bit in the appropriate interrupt enable register. To mask an interrupt, write a 0 into the corresponding bit in the interrupt enable register. The interrupt enable registers and the interrupt status register are mirror images of each other.

The interrupt enable registers are cleared to 0 by system reset, disabling interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	DTI	SI7	SI6	SI5	SI4	VME7	KBD	0	SF	VME6	MEM	DI	SHP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	VME5	0	VME4	0	VME3	0	LM	SLP	VME2	0	VME1	SI3	SI2	SI1	SI0

Bit	Mnemonic	Function
31	ABT	Abort Pushbutton 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	DTI	DUART Timer (system console) 1 Enables the system console DUART timer interrupt. 0 Masks the system console DUART timer interrupts
27	SI7	Software Interrupt 7 1 Enables software interrupt 7. 0 Masks software interrupt 7.
26	SI6	Software Interrupt 6 1 Enables software interrupt 6. 0 Masks software interrupt 6.
25	SI5	Software Interrupt 5 1 Enables software interrupt 5. 0 Masks software interrupt 5.

(continued)

Bit	Mnemonic	Function
24	SI4	Software Interrupt 4 1 Enables software interrupt 4. 0 Masks software interrupt 4.
23	VME7	VME Level 7 Interrupt 1 Enables the $\overline{\text{IRQ7}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ7}}$ interrupt from VME controllers.
22	KBD	Keyboard Interrupt 1 Enables the keyboard interrupt. 0 Masks the keyboard interrupt.
21	Unused	
20	SF	System Failure Interrupt 1 Enables the system failure interrupt. 0 Masks the system failure interrupt.
19	VME6	VME Level 6 Interrupt 1 Enables the $\overline{\text{IRQ6}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ6}}$ interrupt from VME controllers.
18	MEM	Memory Error Interrupt 1 Enables the memory interrupt. 0 Masks the ememory interrupt.
17	DI	DUART Interrupt (system console) 1 Enables the system console DUART interrupt. 0 Masks the system console DUART interrupt.
16	SHP	Signal High Priority (SHP) interrupt 1 Enables the SHP interrupt. 0 Masks the SHP interrupt.
15	Unused	
14	VME5	VME Level 5 Interrupt 1 Enables the $\overline{\text{IRQ5}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ5}}$ interrupt from VME controllers.
13	Unused	
12	VME4	VME Level 4 Interrupt 1 Enables the $\overline{\text{IRQ4}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ4}}$ interrupt from VME controllers.
11	Unused	
10	VME3	VME Level 3 Interrupt 1 Enables the $\overline{\text{IRQ3}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ3}}$ interrupt from VME controllers.
9	Unused	
8	LM	Location Monitor Interrupt 1 Enables the location monitor interrupt. 0 Masks the location monitor interrupt.
7	SLP	Signal Low Priority (SLP) Interrupt 1 Enables the SLP interrupt. 0 Masks the SLP interrupt.
6	VME2	VME Level 2 Interrupt 1 Enables the $\overline{\text{IRQ2}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ2}}$ interrupt from VME controllers.
5	Unused	

(continued)

Bit	Mnemonic	Function
4	VME1	VME Level 1 Interrupt 1 Enables the $\overline{\text{IRQ}}$1 interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ}}$1 interrupt from VME controllers.
3	SI3	Software Interrupt 3 1 Enables software interrupt 3. 0 Masks software interrupt 3.
2	SI2	Software Interrupt 2 1 Enables software interrupt 2. 0 Masks software interrupt 2.
1	SI1	Software Interrupt 1 1 Enables software interrupt 1. 0 Masks software interrupt 1.
0	SI0	Software Interrupt 0 1 Enables software interrupt 0. 0 Masks software interrupt 0.

(concluded)

EXIST Extended Interrupt Status

FFF8 E040 Read only

The Extended Interrupt Status (EXIST) register contains interrupt flags that are set to 1 by devices requesting an interrupt to the CPU. When a device requests an interrupt, the interrupt request signal goes to interrupt logic which sets the corresponding interrupt bit in the EXIST register. The interrupt logic also interrupts the CPU. When interrupted, the CPU must execute a primary interrupt service routine to find out what kind of interrupt it is. The interrupt service routine may read an interrupt enable register as well as the interrupt status register. If it does read an interrupt enable register, it must compare the two registers to find any enabled interrupt(s). The bits in the extended interrupt enable registers and the extended interrupt status register are mirror images of each other.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTC OF	PIT3 OF	PIT2 OF	PIT1 OF	PIT0 OF	Res.	DMA 4C	DMA 3C	DMA 2C	DMA 1C	DMA 0C	Res.	LAN0	LAN1	SCSI0	SCSI1

15	14	13	12	11 9	8	7	6	5	4 0
VIDEO	ZBUF	DUART2	VDMA	Reserved	IOEXP1	Reserved	IOEXP2	PDMA	Reserved

Bit	Mnemonic	Function
31	RTCOF	Real Time Clock Overflow 1 The abort pushbutton interrupt is asserted. 0 The abort pushbutton interrupt is not asserted.
30	PIT3OF	PIT 3 Overflow 1 The PIT3 overflow interrupt is asserted. 0 The PIT3 overflow interrupt is not asserted.
29	PIT2OF	PIT 2 Overflow 1 The PIT2 overflow interrupt is asserted. 0 The PIT2 overflow interrupt is not asserted.
28	PIT1OF	PIT 1 Overflow 1 The PIT1 overflow interrupt is asserted. 0 The PIT1 overflow interrupt is not asserted.
27	PIT0OF	PIT 0 Overflow 1 The PIT0 overflow interrupt is asserted. 0 The PIT0 overflow interrupt is not asserted.
26	Reserved	
25	DMA4C	DMA Channel 4 Complete 1 The DMA channel 4 complete interrupt is asserted. 0 The DMA channel 4 complete interrupt is not asserted.
24	DMA3C	DMA Channel 3 Complete 1 The DMA channel 3 complete interrupt is asserted. 0 The DMA channel 3 complete interrupt is not asserted.
23	DMA2C	DMA Channel 2 Complete 1 The DMA channel 2 complete interrupt is asserted. 0 The DMA channel 2 complete interrupt is not asserted.
22	DMA1C	DMA Channel 1 Complete 1 The DMA channel 1 complete interrupt is asserted. 0 The DMA channel 1 complete interrupt is not asserted.

(continued)

Bit	Mnemonic	Function
21	DMA0C	DMA Channel 0 Complete 1 The DMA channel 0 complete interrupt is asserted. 0 The DMA channel 0 complete interrupt is not asserted.
20	SCC	Synchronous Controller 1 The synchronous controller interrupt is asserted. 0 The synchronous controller interrupt is not asserted.
19	LAN0	Ethernet 0 1 The Ethernet 0 interrupt is asserted. 0 The Ethernet 0 interrupt is not asserted.
18	LAN1	Ethernet 1 1 The Ethernet 1 interrupt is asserted. 0 The Ethernet 1 interrupt is not asserted.
17	SCSI0	SCSI 0 1 The SCSI 0 interrupt is asserted. 0 The SCSI 0 interrupt is not asserted.
16	SCSI1	SCSI 1 1 The SCSI 1 interrupt is asserted. 0 The SCSI 1 interrupt is not asserted.
15	VIDEO	Video 1 The video interrupt is asserted. 0 The video interrupt is not asserted.
14	ZBUF	Z−Buffer 1 The Z−Buffer interrupt is asserted. 0 The Z−Buffer interrupt is not asserted.
13	DUART2	DUART 2 1 The DUART 2 interrupt is asserted. 0 The DUART 2 interrupt is not asserted.
12	VDMA	Video DMA 1 The video DMA interrupt is asserted. 0 The video DMA interrupt is not asserted.
11−9	Reserved	
8	IOEXP1	I/O Expansion 1 1 The I/O expansion 1 interrupt is asserted. 0 The I/O expansion 1 interrupt is not asserted.
7	Reserved	
6	IOEXP2	I/O Expansion 2 1 The I/O expansion 2 interrupt is asserted. 0 The I/O expansion 2 interrupt is not asserted.
5	PDMA	Parallel Printer DMA 1 The parallel printer DMA interrupt is asserted. 0 The parallel printer DMA interrupt is not asserted.
4−0	Reserved	

(concluded)

EXIEN, EXIEN0, EXIEN1, EXIEN2 and EXIEN3 — Extended Interrupt Enable

FFF8 E004	**EXIEN**	**Read/Write**
FFF8 E008	**EXIEN0**	
FFF8 E010	**EXIEN1**	
FFF8 E020	**EXIEN2**	
FFF8 E020	**EXIEN3**	

The Extended Interrupt Enable registers (EXIEN, EXIEN0 − IEN3) enable and mask the extended interrupts to the CPUs. EXIEN0 − EXIEN3 enable interrupts to the corresponding CPU (CPU0 − CPU3), while EXIEN enables interrupts to all CPUs. To enable an interrupt, write a 1 into the corresponding bit in the appropriate interrupt enable register. To mask an interrupt, write a 0 into the corresponding bit in the interrupt enable register. The extended interrupt enable registers and the extended interrupt status (EXIST) register are mirror images of each other.

The interrupt enable registers are cleared to 0 by system reset, disabling interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RTC OF	PIT3 OF	PIT2 OF	PIT1 OF	PIT0 OF	Res.	DMA 4C	DMA 3C	DMA 2C	DMA 1C	DMA 0C	Res.	LAN0	LAN1	SCSI0	SCSI1

15	14	13	12	11 9	8	7	6	5	4 0
VIDEO	ZBUF	DUART2	VDMA	Reserved	IOEXP1	Reserved	IOEXP2	PDMA	Reserved

Bit	Mnemonic	Function
31	RTCOF	Real Time Clock Overflow 1 Enables the abort pushbutton interrupt. 0 Masks the abort pushbutton interrupt.
30	PIT3OF	PIT 3 Overflow 1 Enables the PIT3 interrupt. 0 Masks the PIT3 interrupt.
29	PIT2OF	PIT 2 Overflow 1 Enables the PIT2 interrupt. 0 Masks the PIT2 interrupt.
28	PIT1OF	PIT 1 Overflow 1 Enables the PIT1 interrupt. 0 Masks the PIT1 interrupts
27	PIT0OF	PIT 0 Overflow 1 Enables PIT0 interrupt. 0 Masks PIT0 interrupt.
26	Reserved	
25	DMA4C	DMA Channel 4 Complete 1 Enables the DMA channel 4 complete interrupt. 0 Masks the DMA channel 4 complete interrupt.

(continued)

Bit	Mnemonic	Function
24	DMA3C	DMA Channel 3 Complete 1 Enables the DMA channel 3 complete interrupt. 0 Masks the DMA channel 3 complete interrupt.
23	DMA2C	DMA Channel 2 Complete 1 Enables the DMA channel 2 complete interrupt. 0 Masks the DMA channel 2 complete interrupt.
22	DMA1C	DMA Channel 1 Complete 1 Enables the DMA channel 1 complete interrupt. 0 Masks the DMA channel 1 complete interrupt.
21	DMA0C	DMA Channel 0 Complete 1 Enables the DMA channel 0 complete interrupt. 0 Masks the DMA channel 0 complete interrupt.
20	SCC	Synchronous Controller 1 Enables the synchronous controller interrupt. 0 Masks the synchronous controller interrupt.
19	LAN0	Ethernet 0 1 Enables the Ethernet 0 interrupt. 0 Masks the Ethernet 0 interrupt.
18	LAN1	Ethernet 1 1 Enables the Ethernet 1 interrupt. 0 Masks the Ethernet 1 interrupt.
17	SCSI0	SCSI 0 1 Enables the SCSI 0 interrupt. 0 Masks the SCSI 0 interrupt.
16	SCSI1	SCSI 1 1 Enables the SCSI 1 interrupt. 0 Masks the SCSI 1 interrupt.
15	VIDEO	Video 1 Enables the video interrupt. 0 Masks the video interrupt.
14	ZBUF	Z−Buffer 1 Enables the Z−Buffer interrupt. 0 Masks the Z−Buffer interrupt.
13	DUART2	DUART 2 1 Enables the DUART 2 interrupt. 0 Masks the DUART 2 interrupt.
12	VDMA	Video DMA 1 Enables the video DMA interrupt. 0 Masks the video DMA interrupt.
11−9	Reserved	
8	IOEXP1	I/O Expansion 1 1 Enables the I/O expansion 1 interrupt. 0 Masks the I/O expansion 1 interrupt.
7	Reserved	
6	IOEXP2	I/O Expansion 2 1 Enables the I/O expansion 2 interrupt. 0 Masks the I/O expansion 2 interrupt.
5	PDMA	Parallel Printer DMA 1 Enables the parallel printer DMA interrupt. 0 Masks the parallel printer DMA interrupt.
4−0	Reserved	

(concluded)

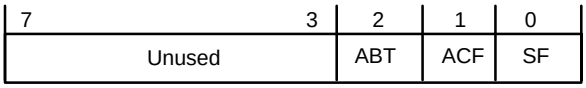
CLRINT Clear Interrupt

Address FFF8 408C Write Only

The Clear Interrupt (CLRINT) register clears the abort, ac fail, and system fail interrupts. The abort interrupt is generated when the abort pushbutton is pressed. The ac fail interrupt is generated when the power supply asserts the $\overline{\text{ACFAIL}}$ line on the VMEbus. The system fail interrupt is generated when any VME controller asserts $\overline{\text{SYSFAIL}}$.

To clear an interrupt request, write a 1 to the appropriate bit in the CLRINT register. For example, to clear a system failure interrupt request, write a 1 into the CSF bit. This clears the System Failure (SF) bit in the Interrupt Status (IST) register.

Resets do not affect CLRINT.



Bit	Mnemonic	Function
7–3	Unused	
2	CABT	Clear the ABRT Interrupt Request. 1 Clears the ABRT interrupt request. 0 Leaves the ABRT interrupt request unchanged.
1	CACF	Clear the ACFAIL Interrupt Request. 1 Clears the ACF interrupt request. 0 Leaves the ACF interrupt request unchanged.
0	CSF	Clear the SYSFAIL Interrupt Request. 1 Clears the SF interrupt request. 0 Leaves the SF interrupt request unchanged.

CLRSWI Clear Software Interrupt

FFF8 4084 Write Only

The Clear Software Interrupt (CLRSWI) register clears software interrupts. To clear a software interrupt, set a bit in CLRSWI. This clears the corresponding interrupt in the Interrupt Status (IST) register. To set a software interrupt, use the Set Software Interrupt (SETSWI) register.

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

Bit	Mnemonic	Function
7	CI7	Clear Software Interrupt 7 1 Clears software interrupt 7. 0 No action.
6	CI6	Clear Software Interrupt 6 1 Clears software interrupt 6. 0 No action.
5	CI5	Clear Software Interrupt 5 1 Clears software interrupt 5. 0 No action.
4	CI4	Clear Software Interrupt 4 1 Clears software interrupt 4. 0 No action.
3	CI3	Clear Software Interrupt 3 1 Clears software interrupt 3. 0 No action.
2	CI2	Clear Software Interrupt 2 1 Clears software interrupt 2. 0 No action.
1	CI1	Clear Software Interrupt 1 1 Clears software interrupt 1. 0 No action.
0	CI0	Clear Software Interrupt 0 1 Clears software interrupt 0. 0 No action.

SETSWI Set Software Interrupt

FFF8 4080 Write Only

The Set Software Interrupt (SETSWI) register generates software interrupts. To generate a software interrupt, set one of the eight available Set Software Interrupt (SI*n*) bits in the SETSWI register. This sets the corresponding interrupt in the Interrupt Status (IST) register. To clear a software interrupt, use the Clear Software Interrupt (CLRSWI) register.

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

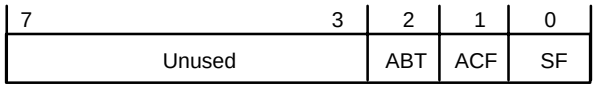
Bit	Mnemonic	Function
7	SI7	Set Software Interrupt 7. 1 Generates a software interrupt 7. 0 No action.
6	SI6	Set Software Interrupt 6. 1 Generates a software interrupt 6. 0 No action.
5	SI5	Set Software Interrupt 5. 1 Generates a software interrupt 5. 0 No action.
4	SI4	Set Software Interrupt 4. 1 Generates a software interrupt 4. 0 No action.
3	SI3	Set Software Interrupt 3. 1 Generates a software interrupt 3. 0 No action.
2	SI2	Set Software Interrupt 2. 1 Generates a software interrupt 2. 0 No action.
1	SI1	Set Software Interrupt 1. 1 Generates a software interrupt 1. 0 No action.
0	SI0	Set Software Interrupt 0. 1 Generates a software interrupt 0. 0 No action.

ISTATE Interrupt State

Address FFF8 4088 Read Only

The Interrupt State (ISTATE) register supplies the current status of the abort, ac failure and system failure interrupts.

Resets do not affect ISS.



Bit	Mnemonic	Function
7–3	Unused	
2	ABT	Abort Button Status 1 Indicates that the abort switch is toggled. 0 Indicates that the abort switch is not toggled.
1	ACF	$\overline{\text{ACFAIL}}$ Status 1 Indicates that the $\overline{\text{ACFAIL}}$ line on the VMEbus is asserted (low). 0 Indicates that. the $\overline{\text{ACFAIL}}$ line on the VMEbus is not asserted (high).
0	SF	$\overline{\text{SYSFAIL}}$ Status 1 Indicates that the $\overline{\text{SYSFAIL}}$ line on the VMEbus is asserted (low) 0 Indicates that the $\overline{\text{SYSFAIL}}$ line on the VMEbus is not asserted (high).

VME Interrupts

VME controllers can interrupt the CPU via the following:

- VME Interrupt Request $\overline{\text{IRQ}n}$ lines
- $\overline{\text{ACFAIL}}$
- $\overline{\text{SYSFAIL}}$
- Global Control and Status (GCS) register

System board interrupt logic monitors the VMEbus for interrupt requests. When an interrupt request is asserted, the interrupt logic sets the appropriate bit(s) in the Interrupt Status (IST) register, compares the interrupts with the interrupt masks, and asserts the INT line to the CPU if a valid unmasked interrupt exists.

When a VME controller interrupts the system board via one of the seven VME Interrupt Request lines, the interrupt logic on the system board sets the corresponding VME Interrupt Request (VME*n*) bit in the IST register and asserts the INT line to the CPU. The interrupt service routine must acknowledge one of these interrupts by reading the corresponding VME Interrupt Acknowledge and Vector (VIAV*n*) register. When read, the VIAV*n* register asserts the interrupt acknowledge signals $\overline{\text{IACK}}$ and $\overline{\text{IACKOUT}}$ 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.

When the system board interrupts a VME controller, the CPU loads the VME Interrupt Vector (VIV) register with the interrupt vector, and loads the VME Interrupt Request Level (VIRL) register with the interrupt level. The VME controller acknowledges the interrupt via the $\overline{\text{IACK}}$ and $\overline{\text{IACKOUT}}$ lines, then reads the interrupt vector from VIV. As a Release On Acknowledge (ROA) interrupter, when an IACK cycle completes, the system board removes the interrupt request from the VMEbus.

Table 3–1 VME Interrupt Registers

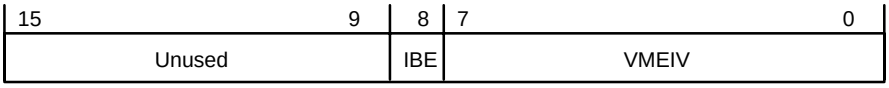
Register	Address
VIAV1	FFF8 5004
VIAV2	FFF8 5008
VIAV3	FFF8 500C
VIAV4	FFF8 5010
VIAV5	FFF8 5014
VIAV6	FFF8 5018
VIAV7	FFF8 501C
VIRL	FFF8 5000
VIV	FFF8 5020

VIAV*n* VME Interrupt Acknowledge and Vector

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

The CPU acknowledges interrupts and obtains VME interrupt vectors via the seven VME Interrupt Acknowledge and Vector (VIAV*n*) registers. Each of the seven interrupt request levels has a VIAV*n* register associated with it. When a VME controller interrupts the CPU via one of the seven VME interrupt request levels, the CPU acknowledges the interrupt request and obtains the interrupt request vector via the VIAV*n* register. When the CPU reads VIAV*n*, the interrupt logic asserts the interrupt acknowledge lines ($\overline{\text{IACK}}$ and $\overline{\text{IACKOUT}}$) to the VMEbus. The VME controller responds by writing the interrupt vector to the data lines and by asserting the Data Transfer Acknowledge ($\overline{\text{DTACK}}$) signal on the VMEbus. If the $\overline{\text{DTACK}}$ signal is not asserted, the system board interrupt logic sets the Interrupt Acknowledge Bus Error (IBE) bit in the VIAV*n* register. If IBE is 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 VIAV*n* registers.



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; the cycle was terminated by a data transfer acknowledge. The VIV bits contain a valid interrupt vector.
7–0	VMEIV[7–0]	VME Interrupt Vector VIV is the interrupt vector from the VME controller generating the interrupt. NOTE: VIV contains a valid vector only when the IBE bit is cleared.

SHP and SLP Interrupts

The following series of steps and Figure 3−1 illustrate how a VME controller initiates an SHP or SLP interrupt to the system board.

1. The VME controller configures the address modifier lines for an A16 access.
2. The VME controller writes, to the Mbus, the address of the GLOBAL1 register.
3. The VME controller writes, to the Mbus, the data to assert either SHP or SLP of the GLOBAL1 register.
4. The system board interrupt logic asserts the interrupt request bit (SHI or SLI) in the Interrupt Status (IST) register.
5. The interrupt logic notifies the CPU, via the INT line, that an interrupt request is pending.
6. The CPU executes an interrupt service routine that reads the Interrupt Status (IST) register and identifies the interrupt.
7. The CPU executes a pre−defined interrupt service subroutine for that interrupt.

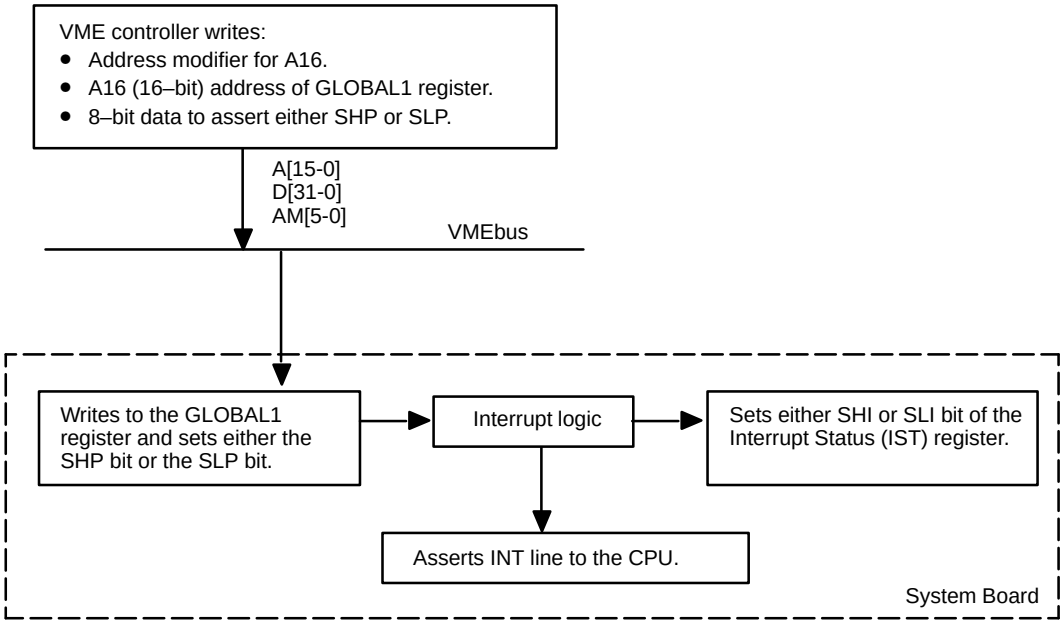


Figure 3–1 VME Controller Initiating an SHP or SLP Interrupt

IRQ*n* Interrupts

VME controllers interrupt each other via the seven VME interrupt request lines $\overline{\text{IRQ}}[7-1]$ located on the VMEbus. $\overline{\text{IRQ7}}$ has the highest interrupt priority and $\overline{\text{IRQ1}}$ has the lowest priority.

The following steps explain how a VME controller initiates a level−1 interrupt to the system board.

1. The VME controller asserts the Interrupt Request 0 ($\overline{\text{IRQ1}}$) 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 interrupt logic notifies the CPU, via the INT input, that an interrupt request is pending.

An interrupt service routine should:

1. Read the IST register and find the interrupt.
2. Read the corresponding VME Interrupt Acknowledge and Vector (VIAV*n*) register to acknowledge the interrupt and to get the interrupt vector.

   There are seven VIAV*n* registers, one for each interrupt level. When the interrupt service routine reads VIAV*n* it also examines the state of the IACK Bus Error (IBE) bit. In a multi−processor system, if the IBE bit is set, the interrupt service routine disregards the interrupt and returns to its previous process.
3. The CPU executes the routine located at the vector obtained from the VIAV*n* register.

Interrupting a VME Controller

The system board can interrupt VME controllers and request that they execute interrupt service routines. The system board initiates the interrupt via the VME Interrupt Request Level (VIRL) register, and supplies the interrupted VME controller with an interrupt vector via the VME Interrupt Vector (VIV) register.

The following steps describe how to initiate a level−1 interrupt.

1. Write the interrupt vector into the VIV register.
2. Write the interrupt level into the VIRL register.
3. The system board interrupt logic asserts the appropriate interrupt request line ($\overline{\text{IRQ}}[6-0]$) to the VME controller.

VIRL VME Interrupt Request Level

FFF8 5000 Read/Write

The VME Interrupt Request Level (VIRL) register defines the VME interrupt level when interrupting a VME controller. The interrupt levels are prioritized: level–7 interrupts have the highest priority, level–1 interrupts have the lowest priority. To interrupt a VME controller, write the interrupt vector to the VIV register, then write the binary–coded interrupt level into the VIRL register. When the VME controller acknowledges the interrupt, VIRL is reset to zero.

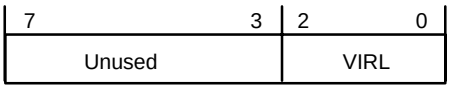
The VIRL register is cleared by both local reset and system reset.

NOTE: Do not write to the VME Interrupt Vector (VIV) register unless VIRL is cleared to 0; the interrupt vector cannot be changed while an interrupt is pending.

If a VME controller does not acknowledge an interrupt generated by the system board, clear the VIRL bits to 0 to remove the interrupt.

Interrupt VME controllers as follows:

1. Read VIRL. If VIRL is 0, indicating that there is no interrupt request pending, write the interrupt vector into the VIV register. If VIRL has a value between 1 and 7, indicating that an interrupt request is pending, wait until the existing VME interrupt completes and the VIRL register is cleared. After VIRL is cleared, write the interrupt vector into the VIV register.
2. Write the interrupt request level into the VIRL register.
3. The VME controller acknowledges the interrupt and reads the interrupt vector from the VIV register.



Bit	Mnemonic	Function
7–3	UNUSED	
2–0	VIRL[2–0]	VME Interrupt Request Level Initiates an interrupt to a VME controller. The interrupt has a specific level as defined below. Level–7 has the highest priority, and level–1 has the lowest priority.

Bit 2	Bit 1	Bit 0	VME Interrupt
0	0	0	No interrupt
0	0	1	Level–1
0	1	0	Level–2
0	1	1	Level–3
1	0	0	Level–4
1	0	1	Level–5
1	1	0	Level–6
1	1	1	Level–7

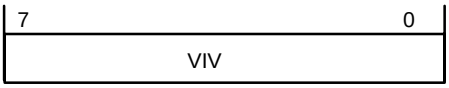
VIV VME Interrupt Vector

FFF8 5020 Read/Write

When interrupting a VME controller, the VIV register provides the interrupt vector to the VME controller. Load the interrupt vector into VIV before initiating a VME interrupt request.

System and local resets do not affect VIV.

NOTE: You cannot write to VIV unless VIRL is cleared to 0. This prevents you from changing the interrupt vector while an interrupt is pending.



Bit	Mnemonic	Function
7–0	VIV[7–0]	VME Interrupt Vector Contains the vector of the device that initiated the interrupt. Before generating an interrupt, read the contents of VIRL (must be 0) then write the interrupt vector into VIV. When the VME controller acknowledges an interrupt, if the contents of address bits A[3–0] of the VMEbus match the request level bits in VIRL, the VME controller will put the contents of VIV[7–0] on the VMEbus data bits D[7–0].

Interrupt Logic

VME controllers can interrupt the CPU via the following:

- VME Interrupt Request $\overline{\text{IRQ}n}$ lines
- $\overline{\text{ACFAIL}}$
- $\overline{\text{SYSFAIL}}$
- Global Control and Status (GCS) register

System board interrupt logic monitors the VMEbus for interrupt requests. When an interrupt request is asserted, the interrupt logic sets the appropriate bit(s) in the Interrupt Status (IST) register, compares the interrupts with the interrupt masks, and asserts the INT line to the CPU if a valid unmasked interrupt exists.

Figure 3–2 and Figure 3–3 illustrate how VME controllers interrupt the system board.

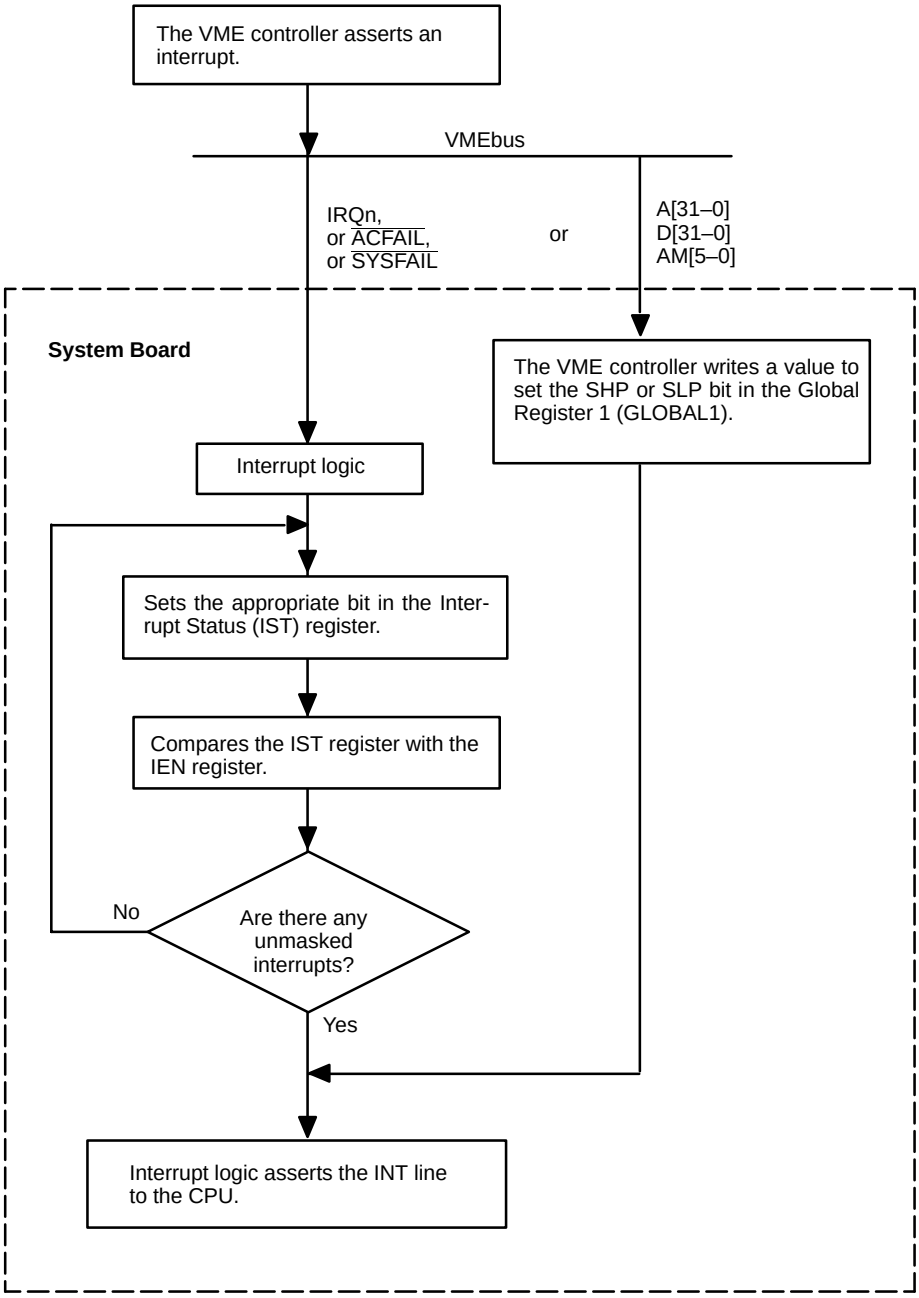


Figure 3–2 System Board Interrupt Logic

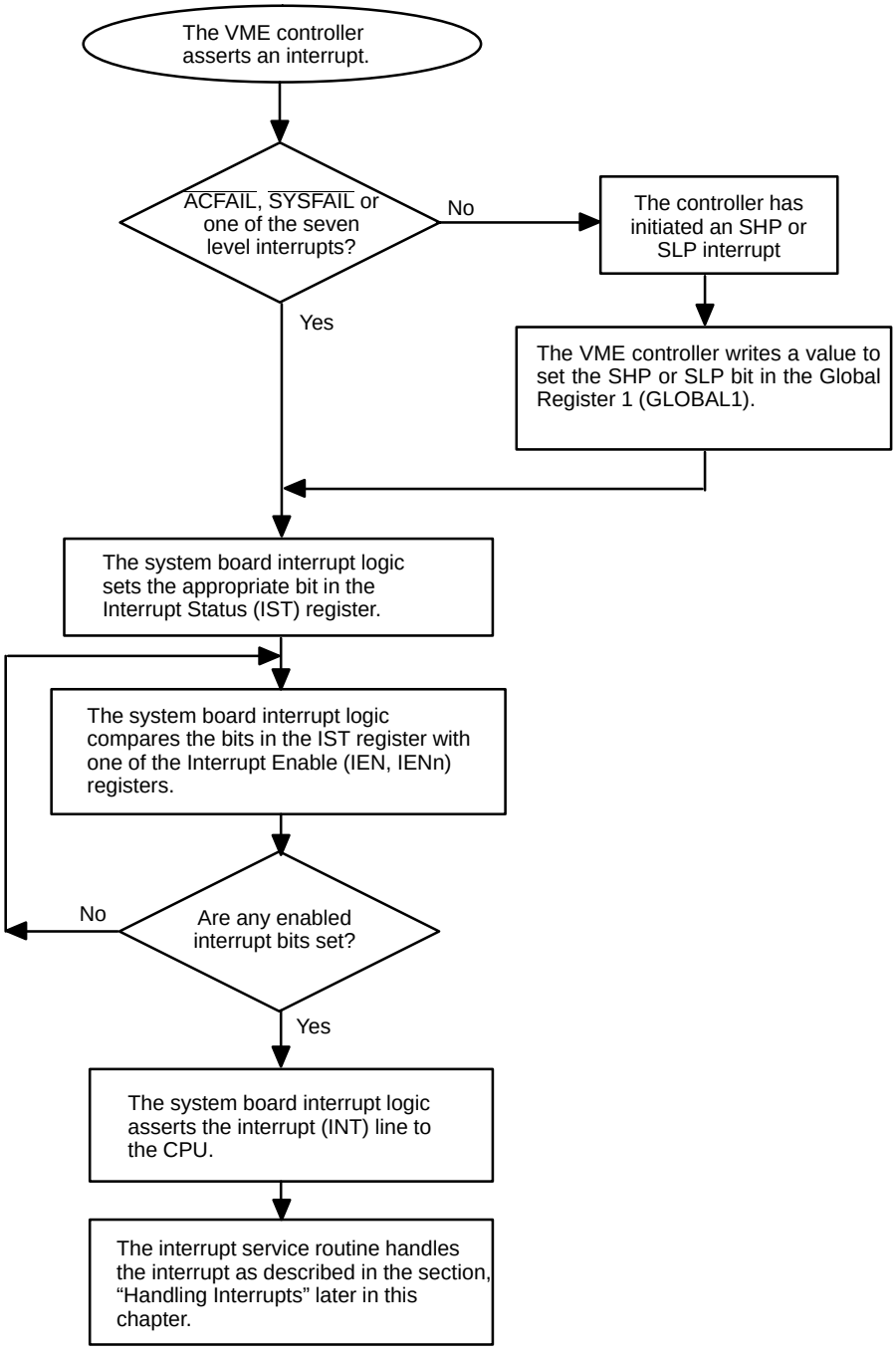


Figure 3–3 Flow Diagram of VME Interrupts to the System Board

Handling Interrupts

This section shows how interrupt service routines may be structured 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 3−4 illustrates how interrupts are handled by a system board with one CPU, and Figure 3−5 illustrates how to service interrupts in a system board with more than one CPU.

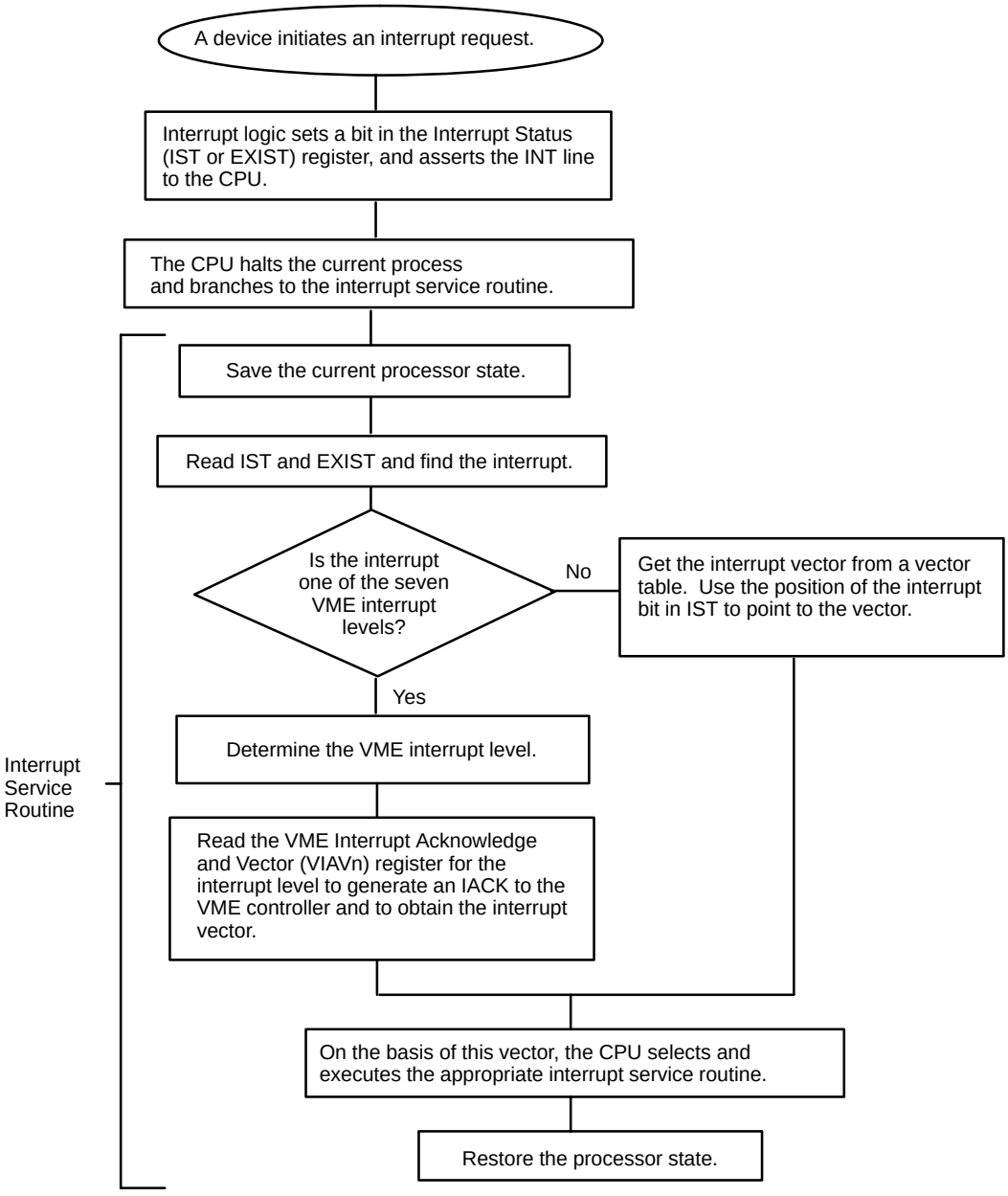


Figure 3–4 Handling Interrupts with a Single–CPU System Board

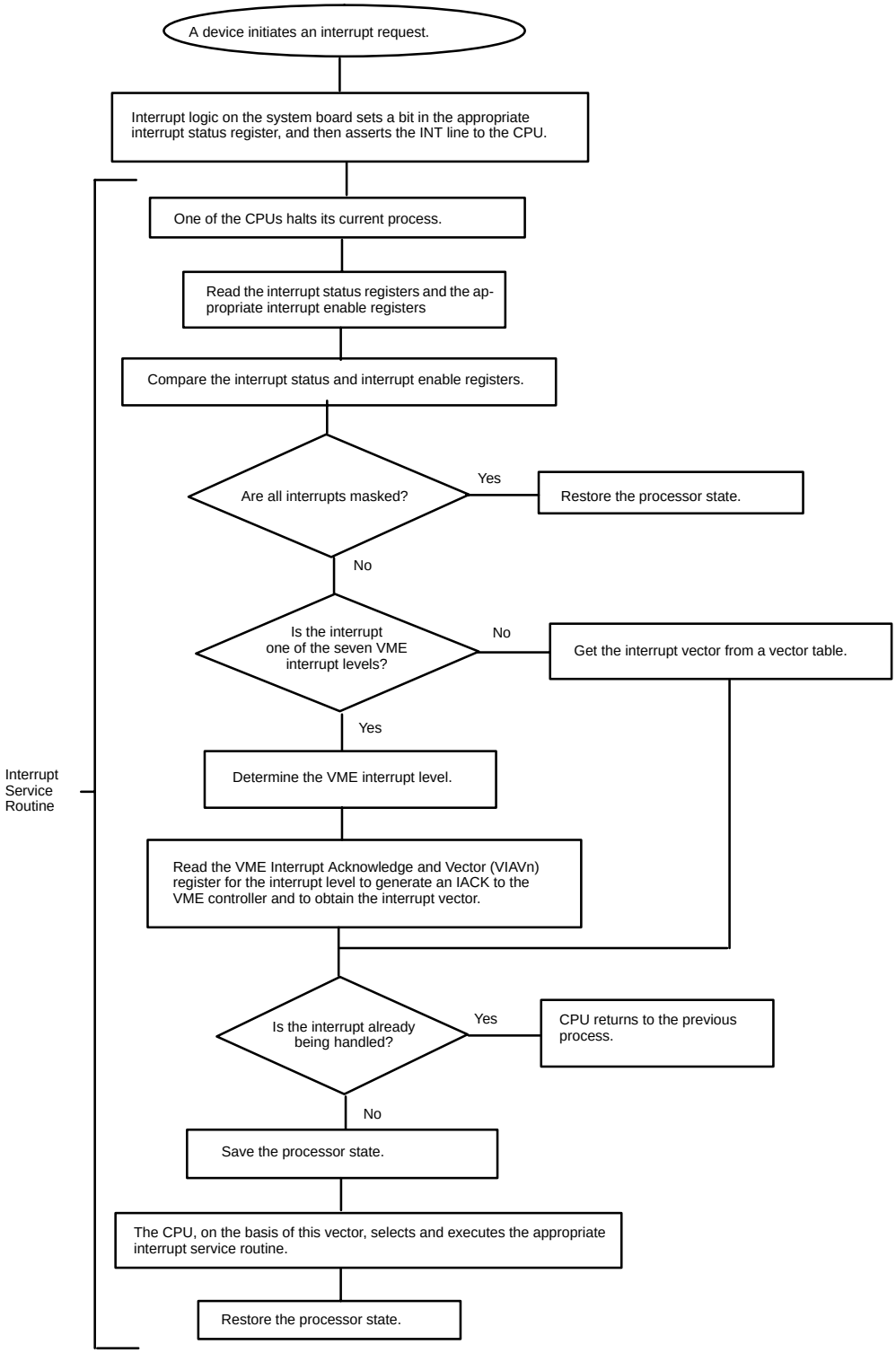


Figure 3–5 Handling Interrupts with a Multiple–CPU System Board

End of Chapter

Chapter 4
Global Resources

This chapter describes the global resources including the Global registers and the Global Control and Status (GCS) registers.

The Global registers are eight registers used by the system board for a variety of tasks such as:

- Regulating VME access to the GCS registers (done by BASAD register).
- Determining whether or not the VME address decoder has been loaded.

The Global Control and Status (GCS) registers are eight registers accessible by VME controllers as well as the system board CPUs.

Global Registers

This section describes the Global registers. Table 4−1 lists the Global registers and their addresses. The Global registers regulate such functions as the global system reset, validity of the address decoders, and VME errors.

Table 4–1 Global Registers

Register	Address
SRST	FFF8 3100
WDTO	FFF8 3108
UCS	FFF8 7000
BASAD	FFF8 7004
GLBRES	FFF8 700C
CCS	FFF8 8000
ERROR	FFF8 8004
WHOAMI	FFF8 8018
IOBRDID0	FFFC F000
IOBRDID1	FFFC F004
CONFIG	FFF8 FFFC

BASAD Base Address

FFF8 7004 Read Only

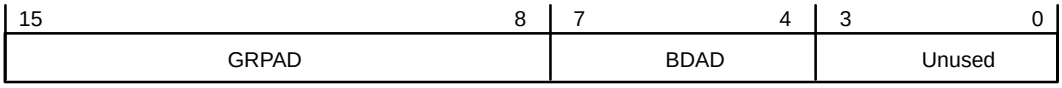
Base Address (BASAD) contains the address entered into the Group Address (GRPAD) and Board Address (BDAD) DIP switches located on the system board. These values are stored in the Group Address (GPA) and Board Address (BDA) bits in the BASAD register. Together, they define the three most−significant nibbles of the A16 address that VME controllers must use to access the Global Control and Status (GCS) registers.

When a VME controller needs to access a GCS register, it writes the A16 address of that register, and the VME interface logic compares address bits A[15−4] with the contents of the BASAD register. If the two addresses are the same, the address is put on the Mbus, and the VME controller can write to or read from the GCS register. Bits A[3−0] select one of the eight GCS registers.

When a VME controller needs to access a GCS register, it does the following within one bus cycle:

1. The VME controller writes the address of the GCS register onto the VMEbus.
2. The VME interface logic compares the value in BASAD with the address from the VME controller. If they are the same address, the VME interface logic passes the address onto the Mbus.
3. The VME controller reads from or writes to the address.

System and local resets do not affect the BASAD register.



Bit	Mnemonic	Function
15−8	GRPAD[7−0]	Group Address GPA contains a copy of the Group Address (GRPAD) switch settings.
7−4	BDAD[3−0]	Board Address BDA contains a copy of the Board Address (BDAD) switch settings.
3−0	Unused	Not used by BASAD; these bits point to one of the eight GCS registers.

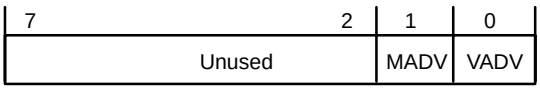
CCS CPU Control and Status

FFF8 8000 Read/Write

The CPU Control and Status (CCS) register, when read, indicates the status of the Mbus address decode logic and the VMEbus Address Decoder (VAD). During power−up, the power−up program loads the VAD and sets the VADV and MADV bits in the CCS register. Setting the MADV and VADV bits indicates that the address decode is ready.

CCS is cleared by system reset, and is not affected by local reset.

CAUTION: Use caution when clearing MADV and VADV; clearing MADV disables all memory mapping and maps the next instruction to the boot PROM. If a problem develops, reset the system to clear the CCS register.



Bit	Mnemonic	Function
7−2	Unused	
1	MADV	Mbus Address Decode Valid Defines whether or not the Mbus address decode logic decodes Mbus addresses. 1 Enables the Mbus address decode logic. System address space reflects the addresses defined in Appendix A, “Address Map”,; utility space is mapped to FFC7 0000 − FFC7 FFFF. 0 Disables the Mbus address decode logic. Only the utility space is accessible, and it is mapped to 0000 0000 − FFC7 FFFF.
0	VADV	VMEbus Address Decoder Valid Defines whether or not the VMEbus Address Decoder (VAD) RAM has been loaded. 1 The VMEbus decoder RAM has been loaded. 0 The VMEbus decoder RAM has not been loaded and is invalid.

CONFIG I/O Gate Array Configuration

FFF8 FFFC Read/Write

The I/O Gate Array Configuration (CONFIG) register defines operating parameters for the i/o gate array.

CONFIG is cleared by local reset, and is not affected by system reset.

31 29	28 16
VID	Unused

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused	EXE	0	EDF5	EDF4	EDF3	EDF2	EDF1	EDF0	1	1	MEM	0	0	1	0

Bit	Mnemonic	Function
31–29	VID	Version Identification Indicates the revision of the i/o gate array.
28–15	Unused	
14	EXE	EXbus Timeout Error When set to 1, EXE indicates that an EXbus timeout error has occurred since the last time EXE was cleared to 0.
13	Reserved	Must be set to 0.
12–7	EDF[5–0]	Enable DMA Flushing When any EDF bit is set, the corresponding DMA channel is flushed.
6	BEN	Bus Error Enable BEN must be set to 1; this enables the DMA controller to sense Mbus errors. If the DMA controller detects an Mbus error while master of the Mbus, it halts and generates an interrupt. If cleared to 0, the DMA controller ignores Mbus errors.
5	FAIR	Fairness Arbitration FAIR must be set to 1 to enable Mbus fairness arbitration.
4	MEM	Memory Disable When set to 1, MEM disables the memory configuration registers located within the address range FFF8 FF00 – FFF8 FF7F.
3	CON	Console Disable CON must be cleared to 0 to enable the DUARTs to operate within the address range FFF8 2000 – FFF8 21FF. When set to 1, CON disables the DUARTs.
2	PEN	Parity Enable PEN must be cleared to 0 to disable Mbus parity checking by the i/o gate array. When set to 1, PEN enables Mbus parity checking..
1, 0	Reserved	Must be set to bit 1 = 1, bit 0 = 0.

ERROR Error

FFF8 8004 Read Only

The Error (ERROR) register provides bus error information. ERROR is cleared automatically after it is read; therefore it contains only errors that have occurred since it was last read.

ERROR is cleared by system reset, and not affected by local reset.

15	14	13	12	11 … 0
IOTO	NELA	Unused	VBE	Unused

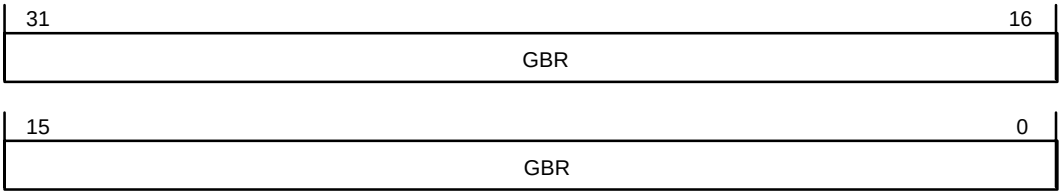
Bit	Mnemonic	Function
15	IOTO	IObus Timeout Set to 1 if an i/o transaction takes longer than 15.36 μs.
14	NELA	Non−Existent Local Address Set to 1 if an access was attempted to a non−existent system board address. This also causes a bus fault.
13	Unused	
12	VBE	VMEbus Error Set to 1 if a VMEbus error has occurred in response to a bus request from the system board.
11−0	Unused	

GLBRES Global Reset

FFF8 700C Write Only

The Global Reset (GLBRES) register generates a System Reset (SRST) when written to. GLBRES resets the entire system, including all VME controllers.

CAUTION: Setting or clearing any bit in the GLBRES register will reset the entire system.



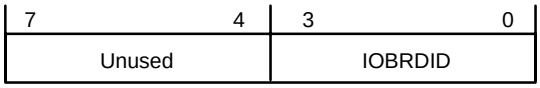
Bit	Mnemonic	Function
31–0	GBR[31–0]	Global Reset Writing to any bit in GLBRES resets the system.

IOBRDIDn I/O Board ID

FFFC F000 IOBRDID0 Write Only

FFFC F004 IOBRDID1 Write Only

The I/O Board ID (IOBRDIDn) registers provide a 4−bit identification value for each expansion I/O slot. If the IOBRDIDn register for an empty expansion I/O slot is read, the IObus times out, and a bus error (value = F_{16}) is logged into the ERROR register.

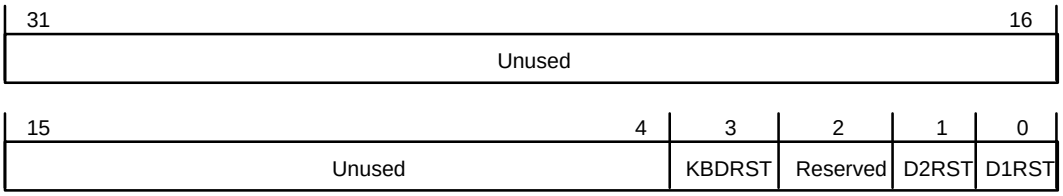


Bit	Mnemonic	Function
7−4	Unused	
3−0	IOBRDID	IO Board ID Contains the board ID for the corresponding IO board.

SRST Software Reset

FFF8 3100 Read/write

The Software Reset (SRST) register is used to reset the keyboard, synchronous communications controller, and DUARTs.



Bit	Mnemonic	Function
31–4	Unused	
3	KBDRST	Keyboard Reset Writing a 0 to KBDRST resets the keyboard.
2	Reserved	
1	D2RST	DUART2 Reset Writing a 0 to D2RST resets DUART2.
0	D1RST	DUART1 Reset Writing a 0 to D1RST resets DUART1.

UCS Utility Control and Status

FFF8 7000 Read/Write

The Utility Control and Status (UCS) register provides status information when read and control of many system board hardware functions when written to. The UCS register bits can all be read from and written to with the exception of the $\overline{\text{PUP}}$ bit. If writing to $\overline{\text{PUP}}$, you can only set $\overline{\text{PUP}}$; you cannot clear $\overline{\text{PUP}}$. Only hardware can clear $\overline{\text{PUP}}$.

15	14	13	12	11	10 9	8	7	6	5	4 3	2	1 0
Unused	PUP	ASF	BIR	VAM	VRL	RNV	FAIR	VRM	ETO	VTS	EWD	WDA

Bit	Mnemonic	Function
15	Unused	
14	$\overline{\text{PUP}}$	Power−up Initialization Indicator Indicates whether or not an ac failure occured. $\overline{\text{PUP}}$ is set when a VME controller asserts the $\overline{\text{ACFAIL}}$ line of the VMEbus. 1 Indicates that an ac failure did not occur ($\overline{\text{ACFAIL}}$ on the VMEbus was not asserted). 0 Indicates that an ac failure occurred ($\overline{\text{ACFAIL}}$ was asserted). $\overline{\text{PUP}}$ is not affected by system reset or local reset. NOTE: $\overline{\text{PUP}}$ is cleared while $\overline{\text{ACFAIL}}$ is asserted or while the system is powering up. $\overline{\text{PUP}}$ is set by the CPU after the system powers up. Only the CPU can set $\overline{\text{PUP}}$.
13	$\overline{\text{ASF}}$	Assert System Failure When set, asserts a system failure. 1 Leaves system failure unchanged. 0 Asserts a system failure. If Inhibit $\overline{\text{SYSFAIL}}$ (ISF) of the GLOBAL1 register is set to 1, the system board will not assert the $\overline{\text{SYSFAIL}}$ line on the VMEbus. If ISF is 0, the system board will assert the $\overline{\text{SYSFAIL}}$ line on the VMEbus. Cleared by either system reset or local reset. NOTE: $\overline{\text{ASF}}$ will light the FAIL LED if asserted (cleared to 0).
12	BIR	Broadcast Interrupt Request Used to assert a broadcast interrupt request over the VMEbus via $\overline{\text{IRQ1}}$. 1 Asserts $\overline{\text{IRQ1}}$. 0 Does not assert $\overline{\text{IRQ1}}$. Cleared by either system reset or local reset. NOTE: $\overline{\text{IRQ1}}$ is an open−collector line; it can be asserted by another interrupter even when BIR = 0.
11	VAM	VMEbus Arbitration Mode Selects the VMEbus Arbitration Mode, either Round Robin or Priority, used by the VMEbus arbiter. 1 Selects Round Robin bus arbitration. 0 Selects Priority bus arbitration. RBN is cleared by system reset and not affected by local reset.

(continued)

Bit	Mnemonic	Function
10,9	VRL[1, 0]	VMEbus Request/Grant Level Determines the request and grant level that the system board will use to access the VMEbus. VMEbus requests are discussed in detail in *The VMEbus Specification*. **VRL** **Level** **VMEbus Signals** 00 0 $\overline{BR0}$, $\overline{BG0IN}$, $\overline{BG0OUT}$ 01 1 $\overline{BR1}$, $\overline{BG1IN}$, $\overline{BG1OUT}$ 10 2 $\overline{BR2}$, $\overline{BG2IN}$, $\overline{BG2OUT}$ 11 3 $\overline{BR3}$, $\overline{BG3IN}$, $\overline{BG3OUT}$ $\overline{BRn}$ Bus Request lines. $\overline{BGnIN}$ Bus Grant lines (input to boards). Form daisy chain with $\overline{BGnOUT}$ lines. $\overline{BGnOUT}$ Bus Grant lines (output from boards). Form daisy chain with $\overline{BGnIN}$ lines. VRL is set to level 3 by system reset, and is not affected by local reset. CAUTION: Do not change VRL while a VMEbus request is pending.
8	RNV	Release Never Determines whether or not the CPU releases access to the VMEbus after the CPU has acquired the bus. 1 Once the system board has acquired the VMEbus, it will not release control of the bus. To release control of the VMEbus, clear the RNV bit. 0 The system board will release control of the VMEbus as defined by the FAIR and VRM bits (see the descriptions that follow). Cleared by system reset and not affected by local reset.
7	FAIR	Fairness Arbitration FAIR defines whether or not the system board uses Fairness bus arbitration when accessing the VMEbus. If operating in Fairness mode, the system board will not request access to the VMEbus if another request is asserted; the CPU waits until no other VMEbus request is pending. 1 The system board uses Fairness arbitration. 0 The system board requests access to the VMEbus as needed. Cleared by system reset and not affected by local reset.
6	VRM	VMEbus Release Mode Defines when the system board releases control of the VMEbus. 1 The system board maintains control of the VMEbus only for the time it takes to complete the transaction. This is Release−When−Done (RWD) mode. 0 The system board releases control of the VMEbus when another VME controller requests access to the VMEbus, or when the transaction is completed. This is Release−On−Request (ROR) mode. Cleared by system reset and not affected by local reset.
5	ETO	Enable VMEbus Arbitration Timeout The bus arbiter contains a timer that measures the time between the issuance of a Bus Grant by the bus arbiter and the assertion of a Bus Busy signal ($\overline{BBSY}$) by the responding controller. The ETO bit determines whether or not the VMEbus arbiter will remove the Bus Grant if the controller does not assert Bus Busy ($\overline{BBSY}$) within one second. 1 Enables the one−second VMEbus arbitration timeout. If Bus Busy ($\overline{BBSY}$) is not asserted within one second of the issuance of a Bus Grant, the VME arbiter will remove the Bus Grant. 0 Disables the one−second VMEbus arbitration timeout. Cleared either by system reset or local reset. NOTE: An Arbiter Timeout (ARBTO) interrupt is generated when a bus grant is removed as the result of a bus timeout.

(continued)

Bit	Mnemonic	Function
4, 3	VTS[1, 0]	VMEbus Data Transfer Timeout Select VTS determines the length of time that the Data Strobes ($\overline{DS0}$ and $\overline{DS1}$) must be asserted on the VMEbus before the system board asserts a Bus Error ($\overline{BERR}$). If the System Board Location (SBL) bit in the GLOBAL1 register is set to 1, the timer is disabled. **VTS — Length of Data Strobes** 00 — 32 µs 01 — 64 µs 10 — 128 µs 11 — Disabled (infinity) VTS is set by system reset and not affected by local reset.
2	EWD	Enable Watchdog Timer The watchdog timer checks for responses to CIO interrupt requests. If the CPU does not respond to a CIO interrupt request within one second, the watchdog timer is set. This results in an action as determined by the WDA[1, 0] bits of this register. (Information on the WDA bits follows.) 1 Enables reactions to watchdog timeouts. When the watchdog timer is enabled and set, an action defined by the WDA bits is performed. 0 Disables reactions to watchdog timeouts. The watchdog timer can be used for other timing functions. EWD is cleared by either system reset or local reset.
1, 0	WDA[1, 0]	Watchdog Action WDA determines what happens when the watchdog timer is set to 1. **Value in WDA — Action** 00 — The system performs a system reset. 01 — The system performs a local reset. 10 — The system performs a local reset and hold. 11 — The watchdog timer is disabled. WDA is set by system reset and is not affected by local reset.

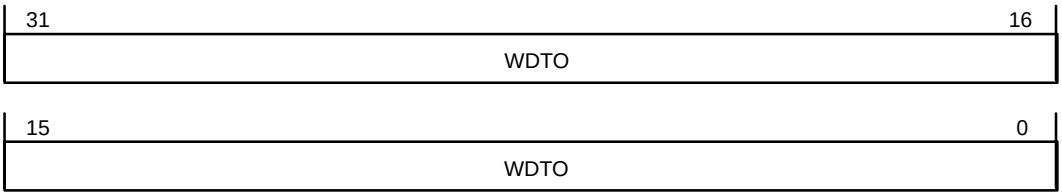
(concluded)

WDTO Watchdog Timeout

FFF8 3108 Write Only

The Watchdog Timeout (WDTO) register generates a bus timeout signal that performs an action defined by the Watchdog Action (WDA) bits of the Utility Control and Status (UCS) register.

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



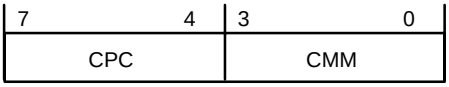
Bit	Mnemonic	Function
31–0	WDTO	Watchdog Timeout Writing any value to WDTO at least one time every second will prevent a timeout system or local reset.

WHOAMI — Who Am I

FFF8 8018 — Read Only

WHOAMI defines the CPU configuration and also defines which CPU is currently master of the Mbus. The CPU configuration defines how many CPUs and CMMUs are on the system board.

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



Bit	Mnemonic	Function
7–4	CPC[3–0]	CPU Configuration The CPU Configuration (CPC) bits define the CPU configuration as follows. **CPC** — **CPU Configuration** 3_{16} — 2 CPU, 12 CMMU 5_{16} — 2 CPU, 4 CMMU 7_{16} — 1 CPU, 6 CMMU A_{16} — 1 CPU, 2 CMMU NOTE: The CPU configuration is set via jumpers on the system board when the CPU board(s) is installed. If CPC does not reflect your configuration, check the CPU configuration jumpers. See the manual, *Setting Up, Starting, Expanding and Maintaining AViiON 530 and 4600 Series Computers* for information on the jumpers.
3–0	CMM[3–0]	Current Mbus Master The CMM bits indicate which CPU is currently the master of the Mbus. The contents of the CMM 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_{16} — CPU0 is master of the Mbus. 2_{16} — CPU1 is master of the Mbus.

Global Control and Status (GCS) Registers

This section describes the Global Control and Status (GCS) registers. The GCS registers are accessed by VME controllers as well as by the system board CPU. The system board and VME controllers pass information via the GCS registers as follows:

- The BRDID register passes the system board ID.
- The GLOBAL0 register passes the location monitors.
- The GLOBAL1 register handles system fail ($\overline{SF}$) interrupts to the VMEbus, handles local resets, passes SHP and SLP interrupts to the system board, and passes the system board location to VME controllers.
- The GPCS*n* registers pass user−defined data.

To access the GCS registers, a VME controller writes a 16−bit address of the GCS register and defines the address as an A16 address via the address modifiers AM[5−0]. The three most−significant nibbles of the address must be the same as the value defined by the BASAD register, which is loaded by the Group Address (GRPAD) and Board Address (BDAD) switches on the system board. When the VME controller defines and writes an A16 address, the VME interface logic on the system board compares the address with the value stored in the Base Address (BASAD) register. If the two addresses are the same, the address is placed on the Mbus. Bits A[3−0] of the address select one of the eight GCS registers.

Unlike the other system board registers, the GCS registers are not on word (4−byte) boundaries. If a VME controller attempts to write or read a word, a bus error is returned. If the system board CPU accesses a word, the GCS logic returns two copies of the register's contents. Table 4−2 lists the GCS registers and their addresses.

Table 4–2 GCS Registers

Register	Address
GLOBAL0	FFF8 6001
GLOBAL1	FFF8 6003
BRDID	FFF8 6005
GPCS0	FFF8 6007
GPCS1	FFF8 6009
GPCS2	FFF8 600A
GPCS3	FFF8 600C
GPCS4	FFF8 600E

CAUTION: When addressing a GCS register from the CPU, do not use the Exchange Memory with Register (XMEM) instruction except via A16 (short I/O) space. This ensures that no other VME masters access the GCS registers between the XMEM load and store operations.

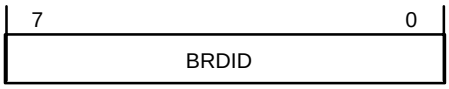
The following pages define the Global Control and Status registers.

BRDID — Board ID

FFF8 6005 — Read/Write (as noted)

Board ID (BRDID) contains the system board identification number, used by the VMEbus to identify the system board.

Cleared by system reset and not affected by local reset.

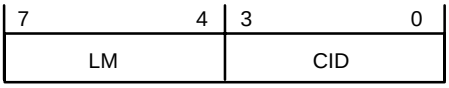


Bit	Mnemonic	Function
7–0	BRDID	Board Identification VMEbus Read only, Mbus Read/Write During powerup, the CPU writes the system board identification into BID. The VMEbus can then read BID to get the system board ID.

GLOBAL0 Global Register 0

FFF8 6001 Read/Write (as noted)

Global Register 0 (GLOBAL0) defines the state of the location monitors and the VMEbus interface chip identification number.



Bit	Mnemonic	Function
7–4	$\overline{\text{LM}}$[3–0]	VMEbus Location Monitors Read/Write 1. Defines whether or not the location monitor detected an access. 1 A location monitor access was not detected. 0 A location monitor access was detected. Set to 1 by system reset and not affected by local reset.
3–0	CID[3–0]	VMEbus/Mbus Chip Identification Number Read only. The CID bits are set to 1111, indicating that the Global Control and Status (GCS) registers use the Motorola Reference Standard. Not affected by either system reset or local reset.

GLOBAL1 Global Register 1

FFF8 6003 Read/Write (as noted)

The GLOBAL1 register sets the Mbus reset, control, and status conditions and informs the CPU of the status of these conditions.

7	6	5	4	3	2	1	0
R&H	SBL	ISF	BDF	0	0	SHP	SLP

Bit	Mnemonic	Function
7	R&H	Local Reset−and−Hold Read/Write both VMEbus and Mbus. Used to reset the CPU and CMMUs. 1 Resets the CPU and CMMUs. Cleared by system reset and not affected by local reset.
6	SBL	System Board Location Read only both VMEbus and Mbus. SBL is hardwired to switch #5 (SCON) of DIP switch S2 (BDAD). When read, SBL must be a 1, indicating that the system board is in VME slot 0 and supplying SYSCLK and BCLR to the VMEbus. If SBL is 0, check the SCON switch. 1 Indicates that the SCON switch is on; the system board is in VME slot 0 and supplying SYSCLK and BCLR to the VMEbus.. 0 Indicates that the SCON switch is off; the system board is not in slot 0 and is not supplying SYSCLK and BCLR to the VMEbus. Not affected by resets.
5	ISF	Inhibit the $\overline{\text{SYSFAIL}}$ output to the VMEbus Read/Write both VMEbus and Mbus. Determines whether or not the system board will pass a system reset on to the VMEbus ($\overline{\text{SYSRESET}}$ line). 1 The system board will not assert $\overline{\text{SYSFAIL}}$ on the VMEbus. 0 Assert $\overline{\text{SYSFAIL}}$ if $\overline{\text{ASF}}$ of the UCS register is asserted or if the watchdog timer has expired. Cleared by system reset and not affected by local reset.
4	BDF	Board Fail Status Read only both VMEbus and Mbus. Indicates whether the system board is initiating a system failure (i.e., it indicates the state of the $\overline{\text{ASF}}$ bit of the UCS register and the output state of the watchdog timer.) 1 Assert System Failure ($\overline{\text{ASF}}$) of the UCS register is asserted (0), or the watchdog timer is set (the timer has expired). 0 $\overline{\text{ASF}}$ is not asserted, and the watchdog timer output is cleared. Not affected by resets; BDF always reflects the state of $\overline{\text{ASF}}$ and the watchdog timer output.
3, 2	Reserved	Always 0

(continued)

Bit	Mnemonic	Function
1	SHP	Signal High Priority VMEbus: Read/Write 1, Mbus: Read/Write 0. SHP is a programmer−defined interrupt to the system board from a VME controller. The programmer defines the function of SHP in the operating system's interrupt service routine. If the interrupt is not already defined by the VME controller that will generate the interrupt, the programmer will have to program the VME controller. 1 Initiates an SHP interrupt request when set to 1 by a VME controller. Asserts the Signal High Priority (SHP) interrupt bit in the Interrupt Status (IST) register. 0 An SHP interrupt request not pending. Cleared by system reset, and not affected by local reset.
0	SLP	Signal Low Priority VMEbus: Read/Write 1, Mbus: Read/Write 0. SLP is a programmer−defined interrupt to the system board from a VME controller. The programmer defines the function of SLP in both the VME controller software and in the operating system's interrupt service routine. If the interrupt is not already defined by the VME controller that will generate the interrupt, the programmer will have to program the VME controller. 1 Initiates an SLP interrupt request when set to 1 by a VME controller. Asserts the Signal Low Priority (SLP) interrupt bit in the Interrupt Status (IST) register. 0 An SLP interrupt request is not pending. Cleared by system reset, and not affected by local reset.

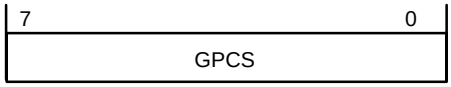
(concluded)

GPCSn General–Purpose Control and Status

FFF8 6007	**GPCS0**	**Read/Write**
FFF8 6009	**GPCS1**	**Read/Write**
FFF8 600A	**GPCS2**	**Read/Write**
FFF8 600C	**GPCS3**	**Read/Write**
FFF8 600E	**GPCS4**	**Read/Write**

The General−Purpose Control and Status (GPCS*n*) registers allow programmer−defined control and status information to be passed from one VME controller to another over the VMEbus. If a VME controller writes control and status into a GPCS*n* register, other controllers can read the GPCS*n* register and use the information. The programmer defines the functions of the bits in these registers.

NOTE: For good programming structure, the GPCS*n* registers should be used only to pass control and status information between VME controllers (including the system board).



Bit	Mnemonic	Function
7−0	GPCS[7-0]	General Purpose Control and Status The programmer defines the function of the GPCS[7-0] bits. A system reset sets GPCS0 to FF. A system reset clears GPCS1 through GPCS4. Local resets do not affect the GPCS*n* registers.

End of Chapter

Chapter 5
Memory

The system has 8 − 128 MBytes of dynamic random-access memory (DRAM) located on single in-line memory modules (SIMMs). The system board has Error Checking and Correction (ECC) logic. For information on the ECC, see this chapter and the *AMD Am29C660 CMOS Cascadable 32−Bit Error Detection and Correction Circuit* specification.

The system board has 512 KBytes of Programmable Read−Only Memory (PROM) which contains boot code. It also has non−volatile RAM (NOVRAM) that contains system configuration parameters.

Access to memory is regulated by memory maps. The CPU can access all of memory, but VME controllers can access only resources allowed by the VME address map.

Accessing Blocks of Memory

Data can be transmitted as blocks with an address asserted at the beginning of the block. A block is the same as a CMMU cache line. The only exception is on the VMEbus; VMEbus blocks can be as large as 256 bytes.

The CMMUs transfer 4−word (16 byte) blocks of data. A CMMU cache line (block) is four 32-bit words that when read from or written to memory, are aligned on an even address whose least−significant nibble begins at address 0_{16}. Figure 5−1 illustrates CMMU cache lines.

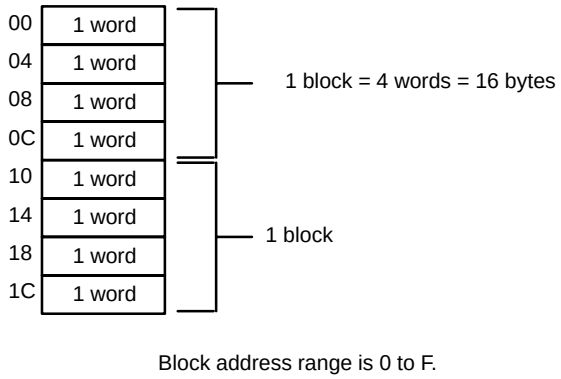


Figure 5–1 CMMU Cache Lines

When a CMMU reads or writes a block of data using an aligned read or write, the read or write will complete within 10 bus cycles. If the read or write crosses from one block to another (called a block crossing), the transfer requires an additional bus cycle.

Reading from Memory

When a bus master reads data from memory, memory sources the data at a rate of one word per bus cycle until it receives an EOR (End of Request) from the master or until the read crosses to a new block of data. If block crossing occurs first, memory asserts End of Data (EOD). Block crossings are described in the "Data Blocks and Block Crossing" section earlier in this chapter.

When data is read from memory, the ECC controller examines the data for single−bit and multiple−bit errors. When a multiple−bit error occurs, a bus error is asserted and an exception routine must be run. When a single−bit error occurs, the error correction logic asserts two wait states within which the logic corrects the data and writes the corrected data to the data bus. The error logic also notifies the interrupt logic that a single−bit error has occurred. An interrupt service routine must then update the error log and write the corrected data back to memory. ECC is performed only during reads from memory, never during writes to memory. The "Error Checking and Correction (ECC)" section later in this chapter describes the error checking process in greater detail.

When reading a block of data from memory, the following occurs:

1. The master writes the beginning address to memory.
2. If memory cannot respond within one clock cycle, memory inserts wait states. The data transfer continues when the wait state is removed.
3. The master reads a word of data from memory.
4. Memory automatically increments the address to the next word, then writes that word to the Mbus. Steps 3. and 4. are repeated until the entire block is read.

Writing to Memory

Memory receives data (in bytes, half−words or words) until it receives an End of Request (EOR) or until a block crossing occurs.

The bus master places a word on the data bus and the addressed memory receives the data. The ECC controller generates the ECC bits and writes them to memory along with the data. If the data transfer is a byte or half−word, the CPU performs a read−modify−write operation to generate the ECC bits. Read−modify−write operations extend the write cycle by adding clock cycles.

When writing a block of data to memory, the following occurs:

1. The master writes the beginning address to memory.
2. If memory cannot respond to the address within one clock cycle, it inserts wait states. The data transfer continues when the wait state is removed.
3. The master writes a word of data to memory, and the ECC controller generates and writes the check ECC bits to memory.
4. Memory automatically increments the address to the next word. Steps 3. and 4. are repeated until the entire block is read.

Cache Coherency

The CMMUs contain cache memory as temporary storage to speed up instruction execution. If the cache is turned off, the CPU interacts directly with memory. When the cache is turned on, the CPU reads from and writes to memory through the cache. When the cache is turned on, the CMMUs update memory using either Writethrough or Copyback mode. When the cache is operating in Writethrough mode, all modifications to the cache are immediately written to memory. When the cache is operating in Copyback mode, only the first modification of each cache location is written to memory.

If the cache is turned off, or if the cache is on and in Writethrough mode, memory is always current because the CPUs write changed data directly to memory. If the cache is on and updated in Copyback mode, locations in memory may not be current because the changed data is written only to the respective cache, not to the memory.

If the cache is on and in Copyback mode, the CMMUs snoop the Mbus for accesses to locations that have been modified by a CPU. If a modified location is read, the appropriate CMMU updates the location before the read is completed. Figure 5–2 illustrates cache coherency. For further information on cache coherency and Mbus snooping, see the *MC88200 Cache/Memory Management Unit User's Manual*.

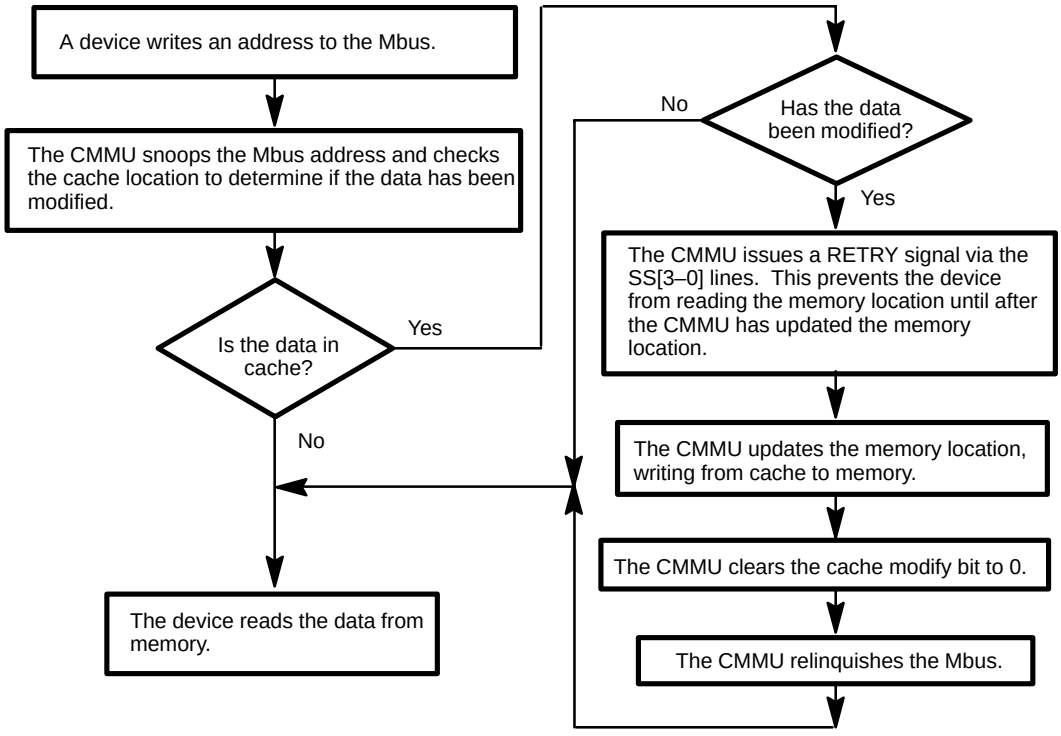


Figure 5–2 Cache Coherency

Error Checking and Correction (ECC)

The system board performs Error Checking and Correction (ECC) for all RAM. It checks for single−bit and multiple−bit errors, and corrects single−bit erors. Multiple−bit errors cannot be corrected. Figure 5−3 outlines the correction process for single−bit errors.

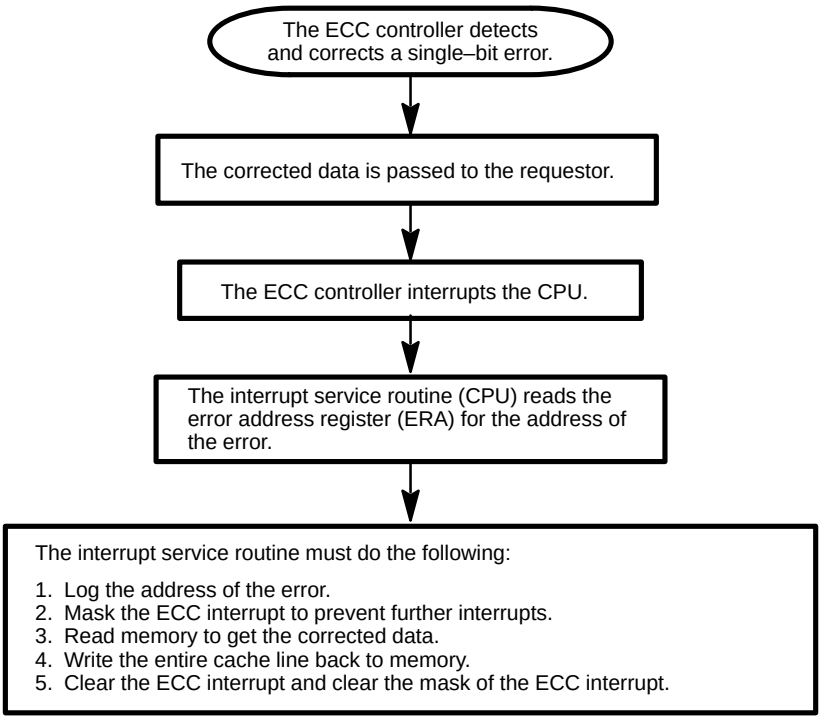


Figure 5–3 Correcting Single–Bit Memory Errors

Registers

This section describes the Memory Diagnostic Control (MDC) register, Memory Diagnostic Status (MDS) register, ECC Error Address (EEAL and EEAU) registers, ECC Check Bits (ECB) register, and the Diagnostic Latch Enable (DLE) register. The following pages describe these registers.

MDC Memory Diagnostic Control

FFF8 FF00 Write only

The Memory Diagnostic Control (MDC) register provides control for the ECC,, and also selects the speed of the video monitor.

NOTE: MDC occupies the same location as the MDS register; MDC is write only and MDS is read−only.

The MDC register is cleared to 0 by resets.

7	6	5	4	3	2	1	0
0	FDMA2	FDMA0	EMS1	EMS0	Unused	ECE	MSS

Bit	Mnemonic	Function
7	Reserved	Must be 0.
6	FDMA2	Flush DMA Channel 2 Flushes remaining bytes from DMA channel 0 to memory. See the DMA Channel 0 and 2 Write to Memory description in Chapter 8. 0 Flush the data. 1 No action.
5	FDMA0	Flush DMA Channel 0 Flushes remaining bytes from DMA channel 3 to memory. See the DMA Channel 0 and 2 Write to Memory description in Chapter 8. 0 Flush the data. 1 No action.
4, 3	EMS1, EMS0	ECC Mode Select Used in conjunction with CORR_EN to select the ECC operating modes defined here: **EMS1 EMS0 ECE Rd/Wr ECC Operating Mode** 0 0 X Write Normal ECC Function 1 0 X Write ” ” 0 0 0 Read ” ” 0 1 0 Read ” ” 0 0 1 Read ” ” 0 1 1 Read ” ” 1 1 0 Read Pass−Through 0 1 X Write Diagnostic Generate 1 0 0 Read Diagnostic Detect Only 1 0 1 Read Diagnostic Detect and Correct 1 1 1 Write Initialize, no ECC
2	Reserved	
1	ECE	ECC Correction Enable Used in conjunction with EMS1 and EMS0 to select an ECC operating Mode. 1 Enables ECC error correction. 0 Inhibits ECC error correction.
0	MSS	Monitor Speed Select 1 70 Hz monitor. 0 60 Hz monitor (default).

MDS Memory Diagnostic Status

FFF8 FF00 Read only

The Memory Diagnostic Status (MDS) register returns the diagnostic status of the memory, plus the selected monitor speed.

NOTE: MDS occupies the same location as the MDC register; MDS is read−only and MDC is write only.

15 13	12	11	10	9	8	7	6 5	4	3	2	1	0
MS	Reserved	SEB	SEA	MEB	MEA	EWE	Reserved	EMS1	EMS0	Unused	ECE	MSS

Bit	Mnemonic	Function
15−13	MS2−0	Memory SIMM Modules Indicates the number of 16-Mbyte SIMMs installed. **MS2 MS1 MS0 # of SIMM Modules** 1 1 1 0 1 1 0 2 1 0 0 4 0 1 0 6 0 0 0 8
12	Reserved	
11	SEB	Single−Bit Error Bank B 1 A single−bit error has occurred in memory bank B. 0 No error.
10	SEA	Single−Bit Error Bank A 1 A single−bit error has occurred in memory bank A. 0 No error.
9	MEB	Multiple−Bit Error Bank B 1 A multiple−bit error has occurred in memory bank B. 0 No error.
8	MEA	Multiple−Bit Error Bank A 1 A multiple−bit error has occurred in memory bank A. 0 No error.
7	EWE	EPROM Write Enable 1 Enables writes to the EPROM. 0 Inhibits writes to the EPROM.
6, 5	Reserved	
4, 3	EMS1, EMS0	ECC Mode Select Used in conjunction with CORR_EN, EMSn define the operating modes selected. See the table within the description of the MDC register.
2	Reserved	
1	ECE	ECC Correction Enable Defines whether or not ECC error correction is enabled. 1 ECC error correction is enabled. 0 ECC error correction is inhibited.
0	MSS	Monitor Speed Select 1 70 Hz monitor selected. 0 60 Hz monitor selected (default).

EEAL, EEAU ECC Error Address

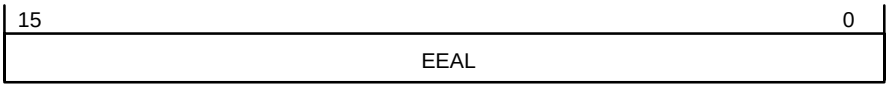
FFF8 FF08 EEAL Read only

FFF8 FF0C EEAU Read only

The ECC Error Address Lower and Upper (EEAL and EEAU) registers contain the address of the first error that occurred since the last time that the CPU read the registers.

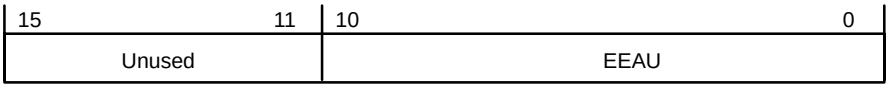
NOTE: EEAL and EEAU will not store a new error address until the existing address has been read. If more than one error occurs between reads, only the first error address is stored; subsequent error addresses are not recorded.

EEAL:



Bit	Mnemonic	Function
15–0	EEAL[15–0]	ECC Error address Lower Contains bits [15–0] of the address of a single–bit or multiple–bit error. This error is the first error that occurred since the last time that the CPU read ERA. Subsequent error addresses are not latched into ERA.

EEAU:

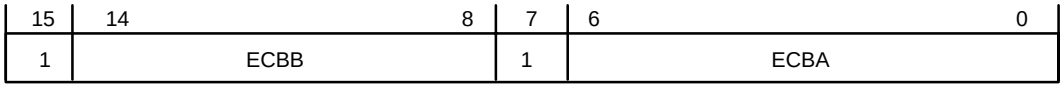


Bit	Mnemonic	Function
15–11	Unused	Always 1s
10–0	EEAU[26–16]	ECC Error address Upper Contains bits [26–16] of the address of a single–bit or multiple–bit error. This error is the first error that occurred since the last time that the CPU read ERA. Subsequent error addresses are not latched into ERA.

ECB ECC Check Bits

FFF8 FF04 Read only

The ECC Check Bits (ECB) register returns the check bits, of both banks A and B, for the first single or multiple bit error that occurred since the *ECC Error Address Upper (EEAU)* register was last read; ECB is cleared to 0 when EEAU is read. When the memory subsystem writes a value into ECB, it will not write another value until the EEAU register has been read. The Memory Diagnostic Status (MDS) register defines which memory bank has the error.

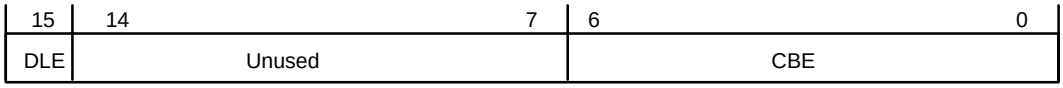


Bit	Mnemonic	Function
15	Unused	Always 1
14–8	ECBB	Bank B ECC Check Bits Contains the ECC check bits for bank B.
7	Unused	Always 1
6–0	ECBA	Bank A ECC Check Bits Contains the ECC check bits for bank A.

DLE Diagnostic Latch Enable

FFF8 FF04 Write only

The Diagnostic Latch Enable (DLE) register is a means to write data to the diagnostic latch of the ECC controller. See the *AMD Am29C660 CMOS Cascadable 32−Bit Error Detection and Correction Circuit* specification



Bit	Mnemonic	Function
15	DLE	Diagnostic Latch Enable When 0, DLE latches the CBE[6−0] bits (at the data inputs of the ECC chip) into the diagnostic latch of the ECC controller. 1 Puts the ECC diagnostic latch in flowthrough mode. 0 Latches data from the data bus into the ECC diagnostic latch.
14−7	Unused	
6−0	CBE[6−0]	Check Bits Enable CBE[6−0] enables and disables check and syndrome bits in the ECC controller. The contents of CBE [6−0] are latched into the ECC controller via the DLE bit.
		CBE **Check Bit** 0 X 1 0 2 1 3 2 4 4 5 8 6 16

End of Chapter

Chapter 6
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.

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 monitor, 60 or 70 Hz noninterlaced refresh rate, compliant with U.S., Canadian, and European VDT standards.

NOTE: The display monitor speed (60/70 Hz) must be programmed into the Memory Diagnostic Control (MDC) register as described in Chapter 5, “Memory.”

- The color palette is 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 28 planes. The planes consist of 8 or 24 color planes and two to four overlay planes. The frame buffer provides extra off−screen storage area for fonts, menus, and look-up table (LUT) entries.
- 8-bit color graphics can generate and display 2^8 or 256 true colors, while 24−bit systems can generate 2^{24} or 16.7 million true colors.
- 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 6−1 and Figure 6−2 illustrate 8−bit and 24−bit color graphics subsystems. 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).

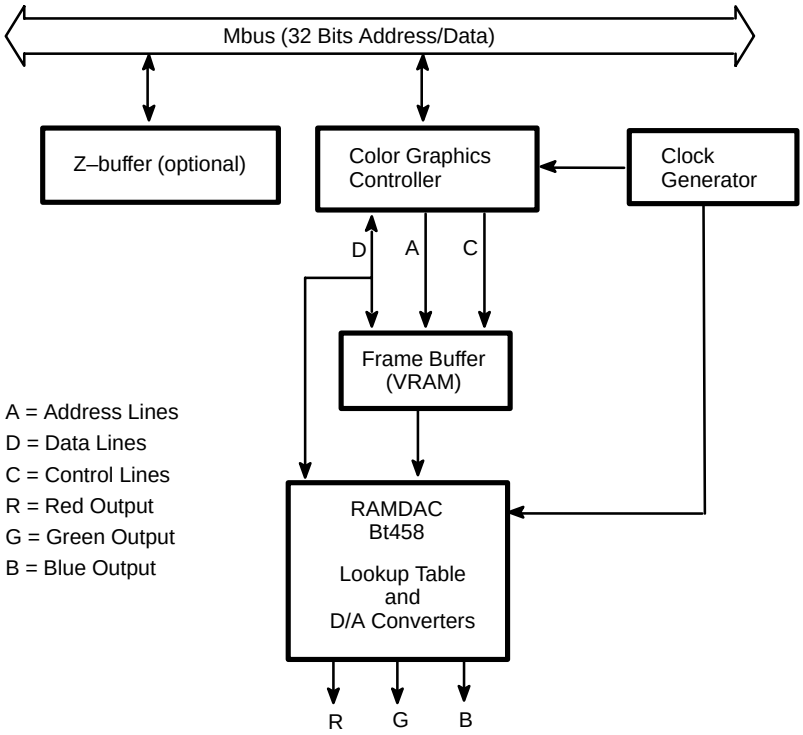
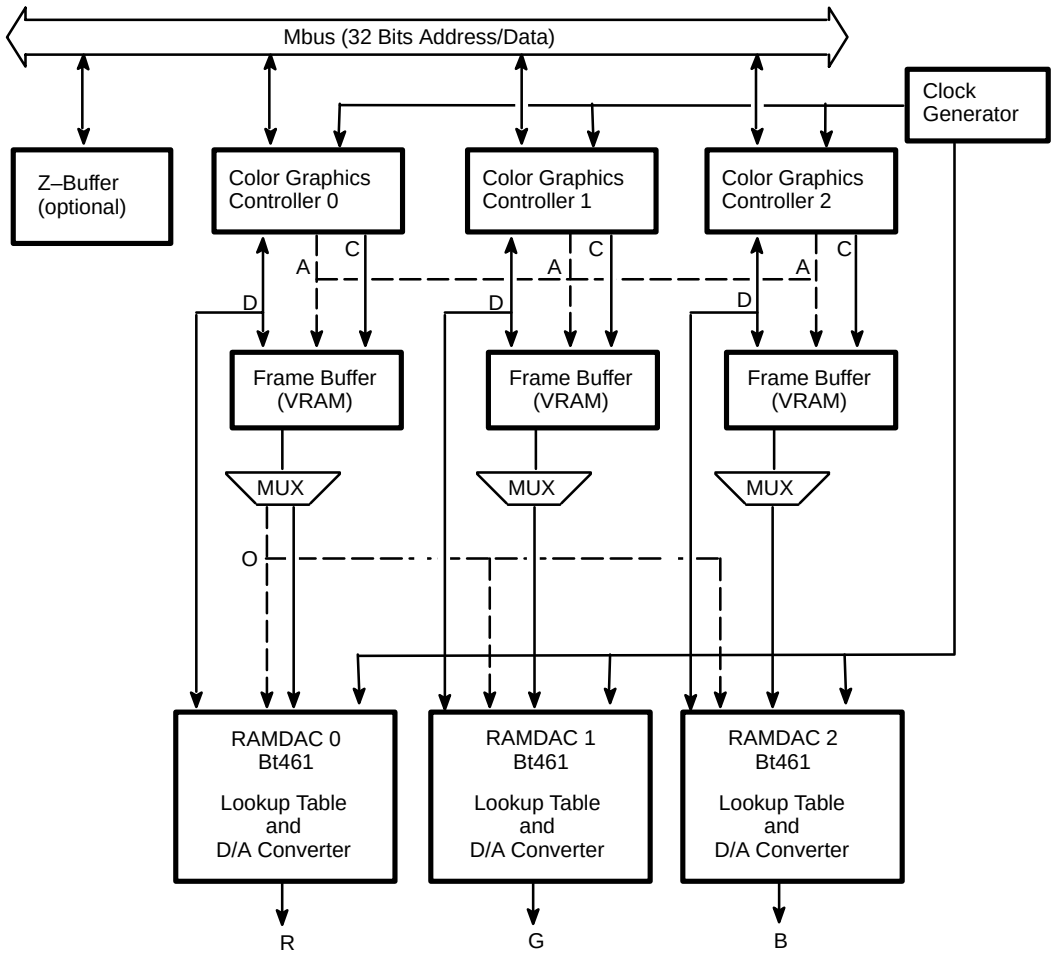


Figure 6–1 Color Graphics Subsystem (8–Bit)



C = Control Lines
A = Address Lines
D = Data Lines
O = Overlay Planes

R = Red Output
G = Green Output
B = Blue Output

Figure 6–2 Color Graphics Subsystem (24–Bit)

The Color Graphics Controller

The color graphics controller manipulates addresses and data, generates control signals, and processes graphics commands. With minimal intervention from the CPU, the graphics controller draws lines, fills in specified areas, and manipulates blocks of data. The color graphics controller communicates with the CPU via the Mbus.

The Frame Buffer

The frame buffer or bitmap stores pixel values that point into the lookup table (LUT). The LUT, or color palette, contains values that define colors to be displayed.

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

The frame buffer has color planes and overlay planes. Color planes define the colors to be displayed, and overlay planes temporarily store superimposed images without destroying underlying images. 8−bit color has 8 color planes and 2 overlay planes. 24−bit color has 24 color planes and 4 overlay planes. Each plane is one bit deep.

The RAMDAC

Both 8-bit and 24-bit color systems use Brooktree RAMDAC (Random Access Memory and Digital to Analog Controller) chips. 8-bit stations use one Brooktree Bt458 RAMDAC, and 24-bit stations use three Brooktree Bt461 RAMDACs. Each RAMDAC consists of a lookup table (LUT) and a digital to analog converter (DAC).

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 one 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 for red, green, or blue.

The Clock Generator

The color graphics subsystem supports 64-kHz and 75 kHz monitors via separate oscillators in the graphics clock generator. The oscillators are 107.352 MHz for 64-kHz monitors and 125.000 MHz for 75-kHz monitors.

NOTE: The display monitor speed (60/70 Hz) must be programmed into the Memory Diagnostic Control (MDC) register as described in Chapter 5, “Memory.”

In 8-bit systems, when you first power-up the system, select the clock by entering <ctrl> v 1 for 64-kHz or <ctrl> v 2 for 75-kHz. This updates the graphics controller and boot ROM so that future boots default to the new clock.

In 24-bit systems, only one oscillator is hardwired to the clock circuitry, therefore to change the frequency, swap the oscillators on the color graphics board before powering-up the system. After you power-up the system, enter <ctrl> v 1 for 64-kHz or <ctrl> v 2 for 75-kHz. This updates the graphics controller and boot ROM so that future boots default to the new clock.

The Z–Buffer

The Z−buffer interpolates and stores 24-bit Z−depth values used for hidden line and surface removal in rendering three-dimensional images. The Z-buffer controller selectively updates pixels in the frame buffer based on the relationship of Z-buffer values from plane to plane.

Programming Conventions

This section describes programming conventions to use with the color graphics subsystem. The major topics in this section include:

- Handshaking
- Context Switching
- Accessing color graphics resources
- Fixed−point numbers
- Interrupts

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.

The graphics controller has built−in commands to create objects and fields, move blocks of data within the frame buffer, and transfer data between the frame buffer and system memory.

The color graphics controller ignores parity bits when written to.

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
1	DIP	Drawing In Progress (Read) 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: • The command completed its function and all frame buffer accesses related to it have begun. • The clipping boundary was crossed. • A context switch occurred.
0	BSY	Busy (Read/Write) 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.

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. Whether or not to pipeline the parameter registers (PARMn) depends on 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

A context switch occurs when an operation (context) is temporarily stopped (interrupted) and replaced with another operation (context). The state of the stopped context must be saved before executing the new context.

If a command terminates at the same time that a Stop command is issued, the STATE1 register is automatically modified to appear as a No Operation (NOP) command. When the new context is restarted, the NOP command will immediately terminate and subsequent commands will continue.

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*.

NOTE: Context switching can be used with pipelined commands.

Context switching is controlled via the Stop (STP) and Resume (RES) bits of the STOP register. To change a context:

1. STOP Register: Set the STP bit 0 to 1.
2. CSR0 Register: Wait until DIP bit 1 is cleared to 0.
3. Save the current context.
4. Load the new context.

When a new context is not available, put the controller in its idle state by:

1. CSR0 Register: Clear the BSY bit 0 to 0.
2. STOP Register: Clear the STP bit 0 to 0.

STOP register:

Bit	Mnemonic	Function
1	RES	Resume Resume a stopped command. 0 No operation. 1 Resume the execution of a stopped command after a context switch is completed. The controller automatically clears the RES bit after the command resumes.
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. 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. 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.

With an 8−bit color graphics subsystem, broadcast accesses can be used to maintain compatibility if upgraded to 24−bit color.

Table 6−1 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 6–1 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 (register, frame buffer, or 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, the registers and the frame buffer are accessed as follows:

8–bit registers:

broadcast accesses:
As shown in Figure 6−3, 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).

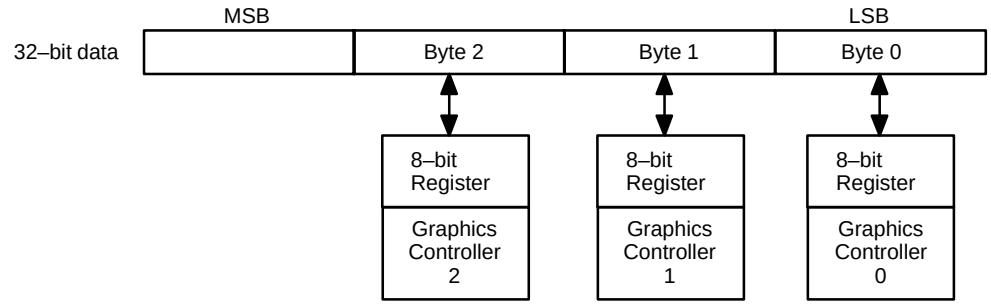


Figure 6–3 Broadcast Data Transfers of 8–bit Registers with 24–bit Color

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 6−2 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.

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*.

Table 6–2 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 [1]	Save	PARM4	Parameter Register 4
FFF8 9054	B/I [1]	Save	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 [1]	Save	PARM9	Parameter Register 9
FFF8 9068	B	Save	PARM10	Parameter Register 10
FFF8 906C	B	Save	PARM11	Parameter Register 11
FFF8 9070	B/I [1]	Save	PARM12	Parameter Register 12
FFF8 9074	B/I [1]	Save	PARM13	Parameter Register 13
FFF8 9078	B/I [1]	Save	PARM14	Parameter Register 14
FFF8 907C	B	Save	PARM15	Parameter Register 15
FFF8 9080– FFF8 909C	—	—	—	Reserved
FFF8 90A0	B	—	DATA	Data Port Register

[1] Depending on the command being executed, these registers may be written to using either a broadcast access or individual accesses. During a context switch, they should be read and written individually.

(continued)

Table 6–2 Color Graphics Registers

Address	Broadcast/ Individual	Context Switch	Name	Function
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
Bt458 RAMDAC Registers: (8-Bit Color)				
FFF8 90C0	I		DAC0	Address Register
00				Overlay Color 0
01				Overlay Color 1
02				Overlay Color 2
03				Overlay Color 3
04				Read Mask Register
05				Blink Mask Register
06				Command Register
07				Control/Test Register
FFF8 90C4	I		DAC1	Color Palette RAM
FFF8 90C8	I		DAC2	Control Register
FFF8 90CC	I		DAC3	Overlay Palette RAM
Bt461 RAMDAC Registers: (24-Bit Color)				
FFF8 90C0				Address Register Low
FFF8 90C4				Address Register High
0000 – 00FF				Alternate Color Palette
0100				Overlay Color 0
011F				Overlay Color 31
0200				ID Register
0201				Command Register 0
0202				Command Register 1
0203				Command Register 2
0204				Pixel Read Mask Register Low
0205				Pixel Read Mask Register High
0206				Pixel Blink Mask Register Low
0207				Pixel Blink Mask Register High
0208				Overlay Read Mask Register
0209				Overlay Blink Mask Register
020C				Test Register
FFF8 90D0– FFF8 90DC	—	—	—	Reserved
FFF8 90E0– FFF8 91F0	B	Save	—	Z–Buffer Registers (see the section, “Programming the Z–Buffer Registers.”)

(concluded)

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.

The following procedure converts a floating−point value to fixed−point:

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 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

Graphics register 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 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.

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

The graphics subsystem pipelines registers; i.e. it has two duplicate sets of registers, one set can be programmed while the other is used to execute an operation. These sets are the visible registers and working registers as illustrated in Figure 6−4. 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.

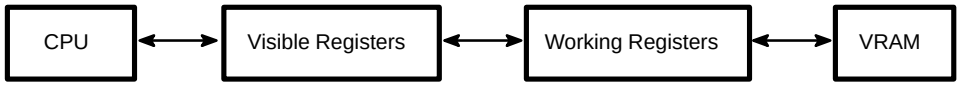


Figure 6–4 Pipelined Registers

The color graphics controller contains the following registers:

Global Registers:

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

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

Descriptions of these registers follow.

Global Registers

The global registers define parameters that affect pixel values, line patterning, and plane masking.

CAUTION: The global registers are not pipelined, therefore do not modify their contents until the Drawing In Progress (DIP) bit in CSR0 is 0, indicating that drawing is inactive.

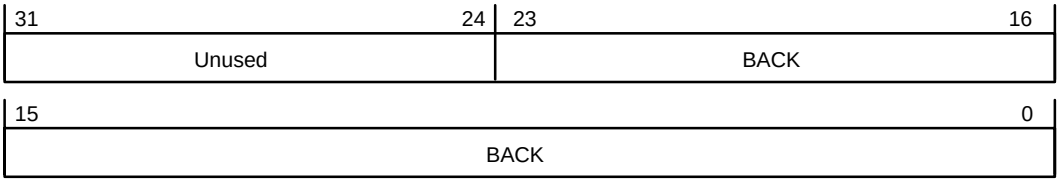
Descriptions of the global registers follow.

BACK — Background Color

FFF8 9014 — Read/Write

The Background Color (BACK) register contains the background color value for patterned line and stippled draw operations.

Use a broadcast address to access the BACK register. In 24−bit color, this allows you to read or write the 8−bit BACK register of each controller with a single access.



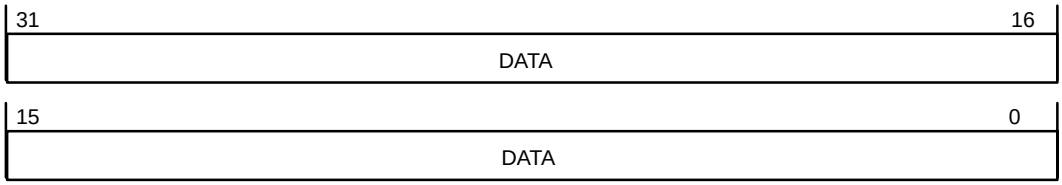
Bit	Mnemonic	Function
31−24	Unused	
23−0	BACK	Background Color Contains the background color value for use during patterned line and stippled draw operations.

DATA — Dataport

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.



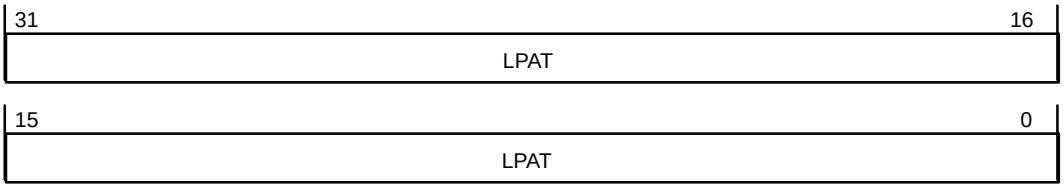
Bit	Mnemonic	Function
31−0	DATA	Dataport Dataport is the data transport interface for block transfers between host memory and video memory.

LPAT — Line Pattern

FFF8 9018 — Read/Write

The Line Pattern (LPAT) register defines the pattern used when drawing a line. 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.

In 24−bit color systems, use a broadcast access to ensure that the three controllers generate identical line patterns. To obtain different line patterns, write to the LPAT register in each of the three controllers independently.



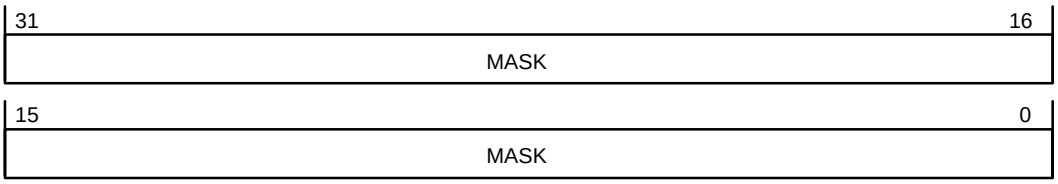
Bit	Mnemonic	Function
31−0	LPAT	Line Pattern Contains the 32-bit line pattern used when drawing a line.

MASK — Plane Mask

FFF8 9010 — Read/Write

The Plane Mask (MASK) register allows you to write to specified planes within the frame buffer. The frame buffer can be partitioned into logical plane groups, each of which can be independently modified. The MASK register is 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.

Use a broadcast address to access the MASK register. In 24−bit color, this allows you to read or write the 8−bit MASK register of each controller with a single access.



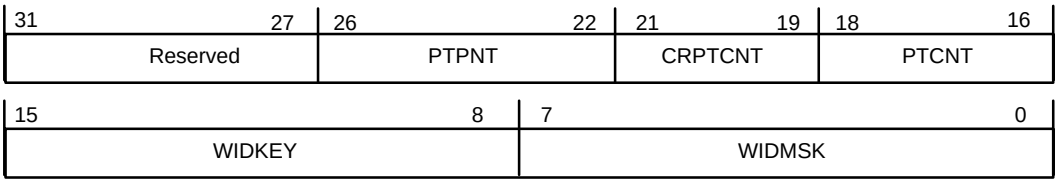
Bit	Mnemonic	Function
31−0	MASK	Plane Mask Masks and enables the buffer planes. 0 Masks the buffer plane. 1 Enables the buffer plane.

PC_WID — Pattern Control/Window ID

FFF8 901C — Read/Write

The Pattern Control/Window ID (PC_WID) register controls line pattern 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.

In 24−bit graphics systems, use a broadcast access to ensure that each controller uses identical line patterns. WIDMSK and WIDKEY are significant only for controllers configured as masters (slave controllers ignore these bits).



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.

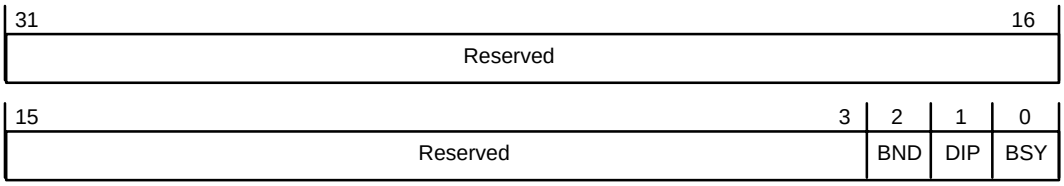
Command and Status Registers

The next few pages describe the command and status registers.

CSR0 Control and Status Register 0

FFF8 9000 Read/Write

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



Bit	Mnemonic	Function
31–3	Reserved	Must be zeroes when written to.
2	BND	Clipping Boundary (Read/Write) 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.
1	DIP	Drawing In Progress (Read) 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.
0	BSY	Busy (Read/Write) 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.

CSR1 Control and Status Register 1

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
31	NMN	Control Signals Enable (Read/Write) 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.
30–23	Reserved	Must be zeroes when written to.
22–20	CBR	CAS Before RAS (Read) Indicates the number of CBR refresh cycles per scan line. **CBR Cycles CBR Cycles** 000 0 100 4 001 1 101 5 010 2 110 6 011 3 111 7
19	VBL	Vertical Blank (Read) Indicates the Current status of the vertical blank (VBLANK) signal. 0 VBLANK is active. 1 VBLANK is inactive.
18	VSN	Vertical Sync (Read) Indicates the current status of the vertical sync (VSYNC) signal. 0 VSYNC is active. 1 VSYNC is inactive.
17	HSN	Horizontal Sync (Read) Indicates the current status of the horizontal sync (HSYNC) signal. 0 HSYNC is active. 1 HSYNC is inactive.
16	BtS	Type of DAC and LUT (Read) Indicates the type of RAMDAC being used. 0 8–bit color uses a Broktree Bt458 RAMDAC and 24–bit color uses Brooktree Bt461 RAMDACs.
15	MST	Master or Slave (Read) Indicates whether this is master or slave controller (24–bit color graphics only). 0 Slave controller. 1 Master controller.

(continued)

Bit	Mnemonic	Function
14	CLI	Clipping Interrupt (Read/Write) Indicates whether or not pixels have been clipped. 0 Pixels have not been clipped. 1 One or more pixels have been clipped.
13	CLM	Clipping Interrupt Mask (Read/Write) Enables/disables the clipping interrupt. 0 Disables clipping interrupt (CLI). 1 Enables clipping interrupt (CLI).
12	DDI	Drawing Done Interrupt. (Read/Write) 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.
11	DDM	Drawing Done Mask (Read/Write) Enables/disables the drawing done interrupt. 0 Disables interrupt when drawing done. 1 Enables interrupt when drawing done.
10	VBI	Vertical Blank Interrupt. (Read/Write) Indicates whether or not a vertical blank has occurred. 0 Vertical blank has occurred. 1 Vertical blank has not occurred.
9	VBM	Vertical Blank Mask (Read/Write) Enables/disables the vertical blank interrupt. 0 Disables vertical blank interrupt (VBI). 1 Enables vertical blank interrupt (VBI).
8	IMK	Interrupt Mask (Read/Write) Enables/disables all interrupt requests. 0 Disables interrupts. 1 Enables interrupts.
7	WST	Wait States (Read) Indicates that an additional wait state is inserted into the Mbus data phase. 0 Extra wait state. 1 No extra wait state.
6–5	POSN	Position (Read) 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
4–2	FBSIZE	Frame Buffer Size (Read) Type and size of frame buffer. 000 2K x 1K, 64K x 4 VRAM 001 1K x 512, 64K x 4 VRAM 010 2K x 1K, 256K x 4 VRAM 011 2K x 2K, 256K x 4 VRAM 100 4K x 2K, 256K x 4 VRAM
1–0	PSIZE	Pixel Size (Read) Number of bits/pixel. 00 8 bits per pixel 10 24 bits per pixel NOTE: PSIZE is set up during power–up reset.

(concluded)

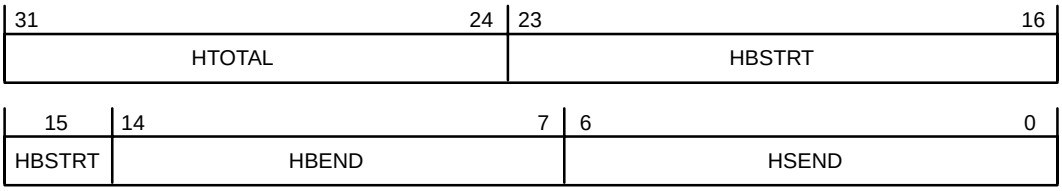
CRT0, CRT1, CRT2 — CRT Timing

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

The CRT Timing (CRTn) registers contain parameters for the composite synchronization signal and composite blank signal. When a noninterlaced display is used, these signals support a variety of screen resolutions. A timing example follows the descriptions of the timing registers.

CRT0 Register

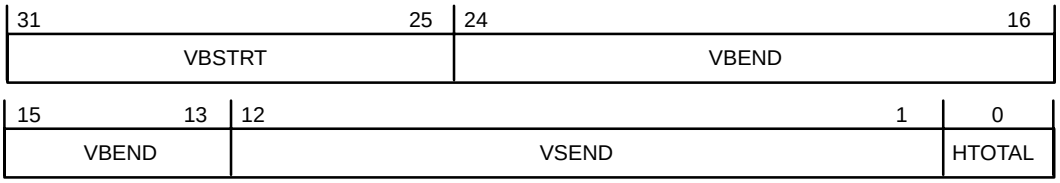
CRT0 defines the timing of the horizontal sync signal. The horizontal timing units are 1/8 of the pixel clock rate (8 pixel times = 1 Hunit).



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.

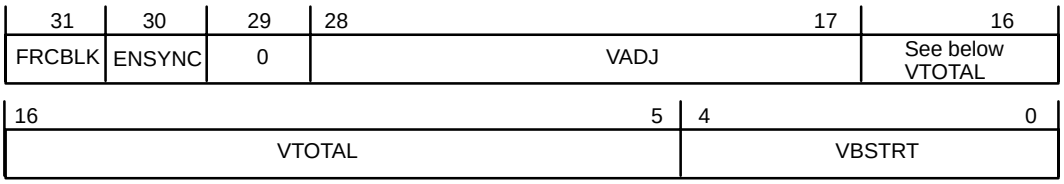
CRT1 Register

CRT1 defines the timing of the vertical sync signal. Vertical timing is programmed in units of horizontal rasters.



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



Bit	Mnemonic	Function
31	FORCBLNK	Force Blank Bit. 0 Forces the composite blank signal low, thus blanking the screen. 1 Enables screen display. Note that VRAM refresh and drawing operations will continue even if the display is being blanked. FORCBLNK is clesred by resets.
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. ENSYNC is clesred by resets.
29	Reserved	This bit should always be 0.
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 video RAMs is transferred to the VRAM 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 monitor.

The timing values for this monitor are

Horizontal resolution	1280 pixels
Vertical resolution	1024 rasters
Pixel clock (125 MHz)	15.625 MHz = 64 ns (1 Hunit)
Horizontal line period	13.312 μs = 208 Hunits
Horizontal front porch	0.512 μs = 8 Hunits
Horizontal sync	1.024 μs = 16 Hunits
Horizontal back porch	1.536 μs = 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

Program the timing registers as follows:

1. Clear the ENSYNC and FORCBLNK bits in CRT2 to 0.
2. Program the timing parameters in any order.
3. Set the ENSYNC and FORCBLNK bits in CRT2 to 1.

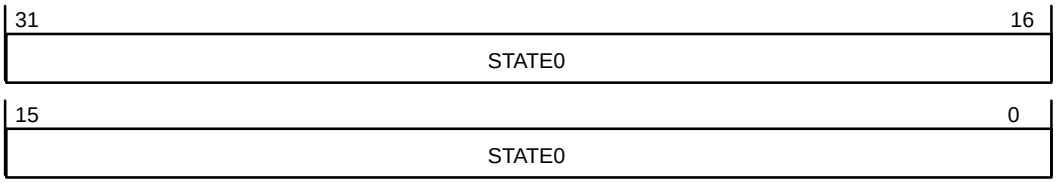
STATE0, STATE1 — Internal State

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

FFF8 9034 **STATE1**

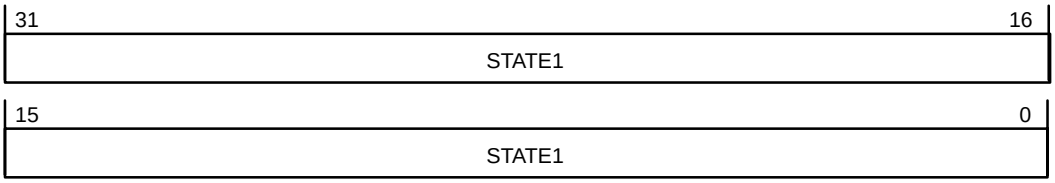
The Internal State (STATEn) registers contain current values for drawing operations. STATE0 contains the current BITBLT mask, direction and shift values. STATE1 contains a copy of the Command register. When switching context, save the STATEn registers and restore them after completing the context switch. When reset, the contents of STATE0 and STATE1 are unknown and unchanged.

STATE0



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

STATE1

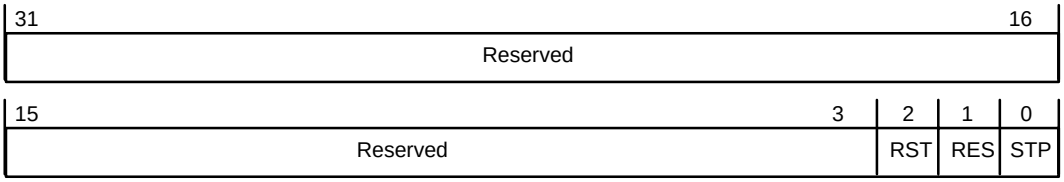


Bit	Mnemonic	Function
31–0	STATE1	Internal State 1 Contains a copy of the contents of the Command (CMD) register.

STOP Stop

FFF8 9004 Read/Write

The Stop register controls context switching.



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

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.

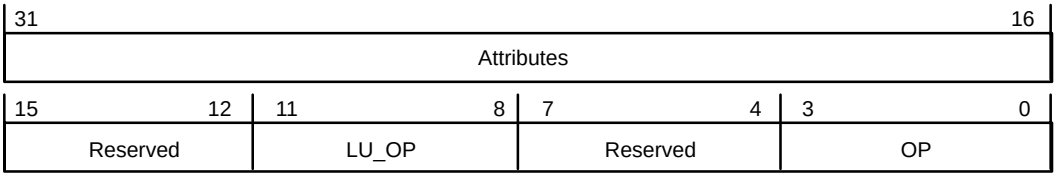
Resets do not affect the contents of the parameter registers.

CMD Command

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 within a clipping rectangle. These lines may be solid or patterned, single−color or shaded.
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.



Bit	Mnemonic	Function
31−16	Attributes	The following attributes are used in conjunction with the commands defined by the operand (OP) bits.
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.
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).

(continued)

Bit	Mnemonic	Function
31–16	Attributes (continued)	The following attributes are used in conjunction with the commands defined by the operand (OP) bits.
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	When set, the graphics controller does not repeatedly read the source plane during a BITBLT; this optimizes the BITBLT for clear and fill operations. 1 Do not read the source plane. 0 Read the source plane.
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.
18	CLIP_STOP	During line operations, selects whether execution stops or continues when the clipping boundary is crossed. 1 Stop and set Clipping Boundary (BND in register CSR0) when clipping occurs. 0 Do not stop when clipping occurs.
17–16	CLIP_CONTROL	Determines how clipping is used. 00 Draw inside clipping rectangle. 01 Do not clip. 10 Draw outside clipping rectangle (pick). 11 Do not clip.
15–12	Reserved	Must be zeroes when written to, and undefined when read from.

(continued)

<table>
<tr><th>Bit</th><th>Mnemonic</th><th>Function</th></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>Function</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 6–3 describes the Command bits. Within the table, the dash (—) indicates that the value of this bit does not affect the command.

Table 6–3 Color Graphics Command Bits

Command	Line Stipple Bit [1]	Line Ptn Bit	Mbus Bit	Command Register Bits 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.

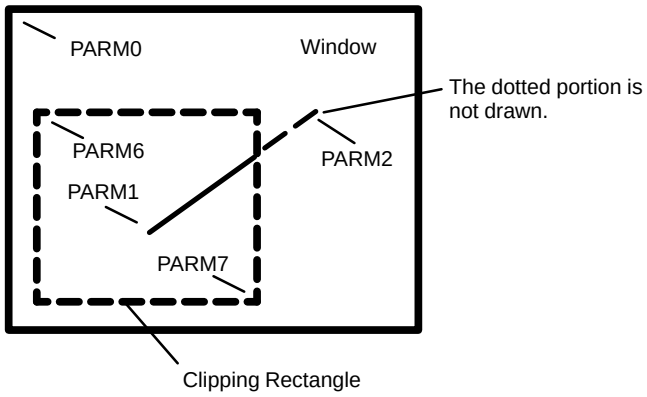
LINE

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.

The LINE command:

- Supports stippling.
- Supports 16 standard logic operations defined by the X Window System™.
- Can draw the last pixel in the line.
- Draws shaded or solid lines.
- Supports 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 bits in CMD.

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 LINE command will automatically execute when PARM2 is written to.

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.

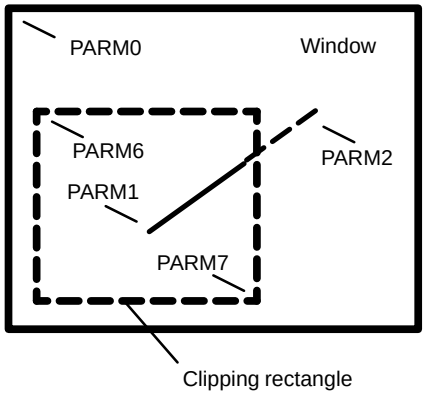
If transparency is enabled, transparent pixels are not drawn, that is, no memory cycle is executed for them. In 400 series stations, if Z−buffering is enabled, transparency takes precedence over Z−buffering, resulting in transparent pixels retaining their old Z−value, regardless of the Z−comparison.

CLINE
Continue Line After Clip

Continue Line After Clip (CLINE) redraws a previously drawn and clipped line using new clipping boundaries.

If a line is clipped, update PARM6 and PARM7 and issue the CLINE command.

Do not pipeline the CLINE command; it is unknown if the line will step outside of the window, or into which area it will go next.



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 current pointer register (PARM8) contains the last pixel to be drawn, and at the end of the command will equal origin+destination.

The line pattern (LPAT) register is 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.

Executing the CLINE Command

Follow this procedure to execute the CLINE command:

1. Poll the DIP bit and the BND bit in CSR0.
   a. Poll the DIP bit in CSR0, wait for it to clear to 0.
   a. Poll the BND bit in CSR0, wait for it to set to 1.
2. Program the parameter registers (PARM6 and PARM7) and the global registers. PARM6 and PARM7 are the clipping boundaries.
3. Poll the BSY bit in CSR0 for 0.
4. Program the Command register; the command will execute automatically.

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 6−5).

The POLY command:

- Supports flat shading and Gouraud shading (linear interpolation).
- Supports stippling and transparency.
- Supports WID clipping.

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

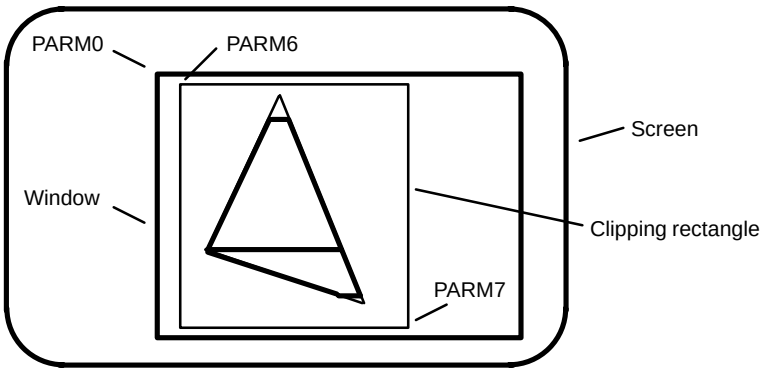
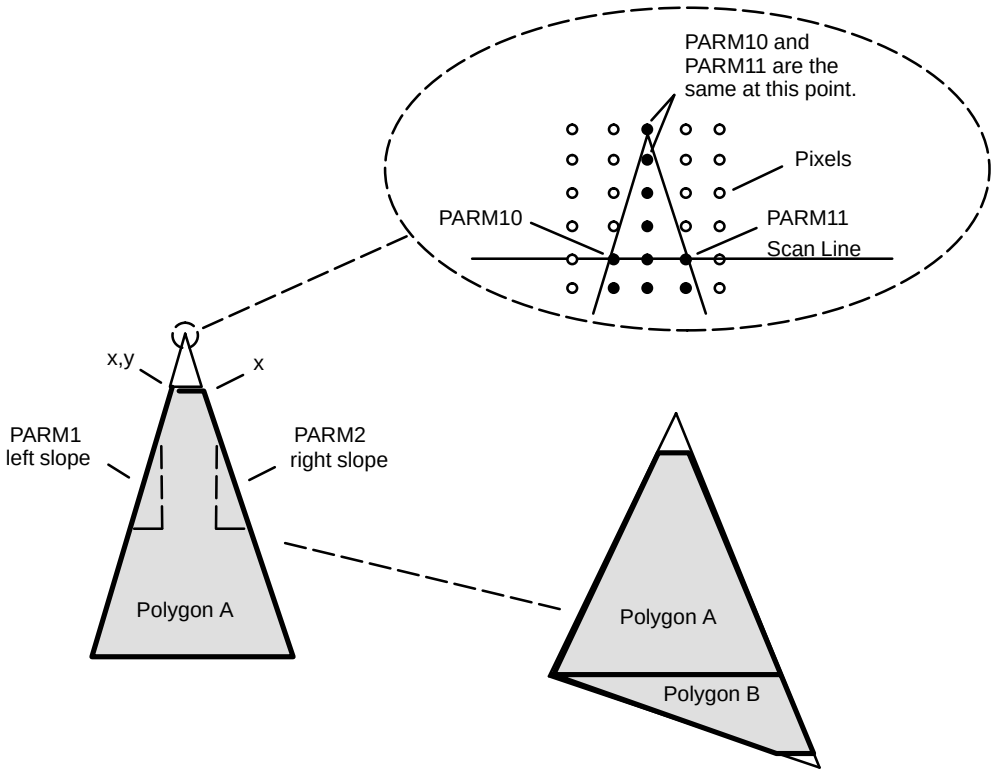
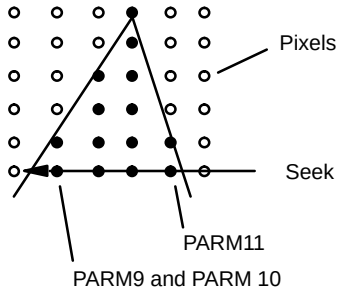


Figure 6–5 Global Elements of the POLY Command

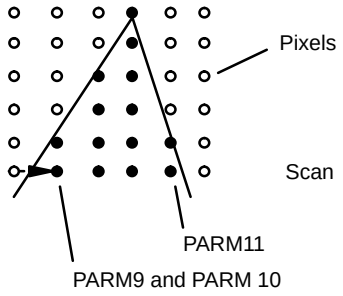
Figure 6−6 illustrates some of the local parameters used in drawing a polygon. PARM9, the initial scan, defines where in the clipping rectangle to begin the scan. The graphics controller increments PARM9 after each scan to scan down through the clipping rectangle. If the top of the polygon is clipped,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

The POLY command does not pipeline registers, therefore test the DIP bit before programming the registers.

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 x 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.

Executing the POLY Command

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. Program the parameter registers PARM0, PARM1, PARM3–PARM7, PARM9–PARM11, and PARM13–PARM15. PARM5–PARM7 are optional.
4. Program PARM2. The POLY command will automatically execute when PARM2 is written to.

5. For the second half of the triangle:
   a. Poll the DIP bit in CSR0 for 0.
   b. Update PARM1, PARM10, PARM11, and PARM13.
6. Program PARM2. The BITBLT command will automatically execute when PARM2 is written to.

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.

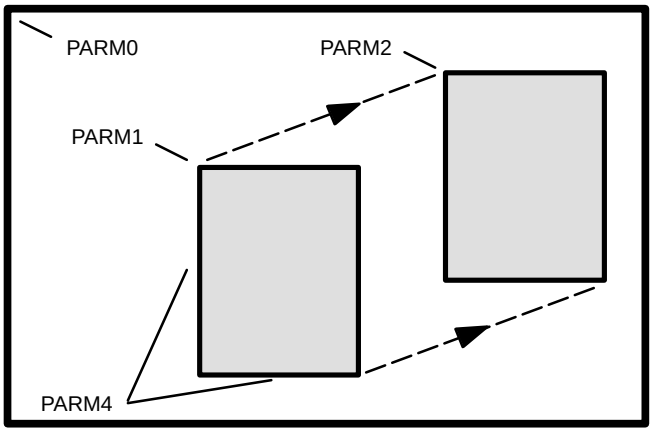
BITBLT
Bit Block Transfer

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.

The BITBLT command:

- Supports 16 standard logic operations defined by the X Window System.
- Supports fast area fills with selectable pixel values.
- Supports 32 pixel x 32 pixel patterns.
- Supports color expansion (stippling) from a single selectable source plane into multiple planes, with or without patterning.
- Supports screen−door transparency in stipple mode.
- Supports 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	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.

Executing the BITBLT Command

Follow this procedure to execute the BITBLT command:

1. If necessary, update the global registers as described in 1a and 1b, 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 PARM0, PARM1, PARM3 and PARM4.
5. Program PARM2. The BITBLT command will automatically execute when PARM2 is written to.

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.

Executing the RXFER Command

Follow this procedure to execute the RXFER command:

1. Poll the BSY bit in CSR0 for 0.
2. Write the RXFER opcode into the Command register.
3. Program the parameter registers PARM0 and PARM4.
4. Program PARM2. The RXFER command will automatically execute when PARM2 is written to.
5. Read the requested frame buffer data from the Dataport register. All specified words must be read, otherwise you must reset the color graphics controller to terminate the command.
6. Wait for the DIP bit in CSR0 to clear before executing another 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.

Executing the WXFER Command

Follow this procedure to execute the WXFER command:

1. Poll the BSY bit in CSR0 for 0.
2. Program the Command register.
3. Program PARM0 (defining transfer area), PARM3 (foreground color — for XY−mode only), PARM4, PARM6 (clip rectangle — only with clipping enabled) and PARM7.
4. Program PARM2. The WXFER command will automatically execute when PARM2 is written to.
5. Write successively to the Dataport register. All data specified must be written, otherwise you must reset the color graphics controller to terminate the command.
6. Wait for the DIP bit in CSR0 to clear before executing another 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

The graphics controller may program the frame buffer as the result of an operation, or the graphics program may program the frame buffer via the graphics controller.

The frame buffer is a contiguous block of memory organized into color planes and overlay planes. Color planes contain color data and overlay planes temporarily store superimposed images without destroying the underlying images. Color data occupies addresses 8000 0000 − 83FF FFFF, and overlay data occupies addresses 8400 0000 − 87FF FFFF.

Overlay planes are for the cursor, pop-up menus, and clipping to nonrectangular window boundaries.

Frame Buffer Size

With 8-bit color, the frame buffer is 1536 x 1024 pixels x 10 bits/pixel (8 bits color data and 2 bits overlay data). Only the leftmost 1280 x 1024 pixels appear on the screen; the remaining 256 x 1024 pixels are available for storing fonts, stipple patterns, tile patterns, and LUT data.

With 24-bit color, the frame buffer is 1024 rows (rasters) x 2048 columns x 28 bits/pixel (24 bits color data and 4 bits overlay data). Only the leftmost 1280 x 1024 pixels appear on the screen; the remaining 768 x 1024 pixels are available for storing fonts, stipple patterns, tile patterns, and LUT data.

When writing to or reading from the frame buffer, use 32 bit words aligned on word boundaries; other accesses are not allowed and may generate unusable data. Each access reads or writes data for one pixel. Ignore unused bits when reading from the frame buffer.

To find out the size of the frame buffer, read the FBSIZE bits of the CSR1 register (see the description of CSR1).

Frame Buffer Addresses

The address of each pixel can be calculated as follows:

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

where:

- the base addresses are:

  8000 0000 for the frame buffer
  8400 0000 for the overlay planes
  8800 0000 for the Z-buffer

- 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 Access Restrictions

All frame buffer activity must be completed before accessing the frame buffer.

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

Accessing the frame buffer slows drawing and auto LUT load operations. If you continually and rapidly perform frame buffer accesses, an auto LUT load may not complete during a 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. Video memory cycles are needed when loading the LUT, therefore the drawing in progress will slow down while loading the palette.

Using 8-bit color, the Bt458 has a 256 column x 24 row color palette. Programming the entire color palette requires 768 (256 columns x 3 8-bit rows) entries.

Using 24-bit color, each of three Bt461s has a 1024 column x 8 row color palette. Programming the entire color palette of the three Bt461s requires 3072 (1024 columns x 3 Bt461s) 8-bit entries.

Automatic LUT Load (ALL)

Automatic LUT load (ALL) automatically loads frame−buffer resident color palettes into the LUT. The Palette_0 Pointer (PLT0), Palette_1 Pointer (PLT1), and BLINK registers control the ALL.

ALL supports Palette Loading and Blinking.

- Palette Loading begins at the leading edge of the vertical blank period following an ALL command and continues without further host intervention. The color graphics controller automatically reads the palette entries from the frame−buffer and writes them to the LUT. The entire palette transfer completes within a vertical blank, thereby preventing undesirable on−screen artifacts.
- Blinking functions like Palette Loading, but instead of loading the palette once, loading is continuous, alternating between two 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.

ALL Registers

Three registers govern the operation of the ALL: 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 LUT 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 LUT, 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 color data is loaded, not the overlay data or the control registers.

Blinking

Blinking 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.

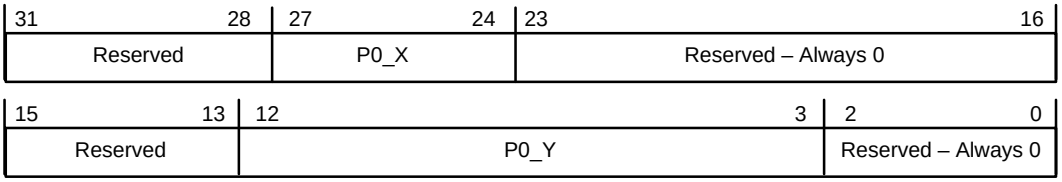
Double–Buffering

Double-buffering segments the frame buffer into two buffers, each of which can be enabled to display its contents without reloading the frame buffer. In 8-bit color graphics, each buffer has 4 bits/pixel. In 24-bit color graphics, each color (R, G, B) has 4 bits/pixel. Use the PLT0, PLT1, and BLINK registers to program and use double buffering.

PLT0 Palette_0 Pointer

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.

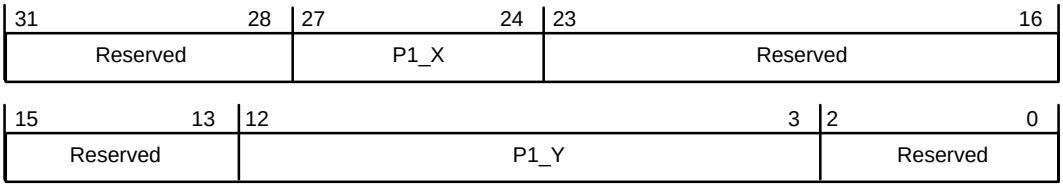


Bit	Mnemonic	Function
31−28	Reserved	Must be zeroes when written to, and undefined when read from.
27−24	P0_X	X field starting coordinate for palette 0. Corresponds to screen X−coordinate bits 11−8 (bits 7−0 are always 0).
23−13	Reserved	Must be zeroes when written to, and undefined when read from.
12−3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

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.



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.

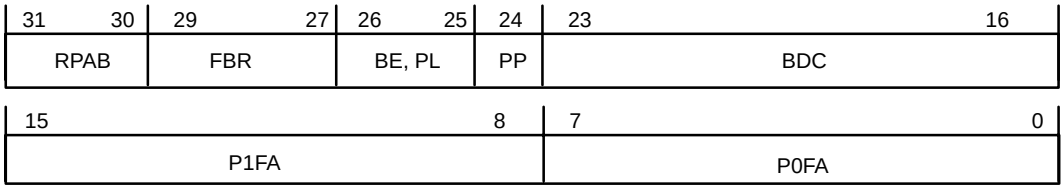
BLINK Blink

FFF8 90AC Read/Write

The Blink (BLINK) register controls palette loading and blinking. RPAB, FBR, PL, and PP control palette loading, and BE, BDC, P1FA, and P0FA control blinking.

When loading the palette, PL enables palette loading, PP selects palette 0 or palette 1, FBR specifies the number of transfer rows, and RPAB selects the upper palette addresses in 24-bit color.

When blinking, BE enables the Blink mode, P0FA and P1FA specify the active periods for palette 0 and palette 1, and BDC indicates the time remaining for the current palette. These fields control the blink-rate and duty-cycle.



Bit	Mnemonic	Function
31, 30	RPAB	Palette Address Bits RAMDAC palette address bits 9 and 8 (Set to 11_2 with Bt461 (24-bit color), clear to 00_2 with Bt458 (8-bit color)).
29–27	FBR	Frame Buffer Rows Number of frame buffer rows to transfer.
26, 25	BE, PL	Blink Enable, Palette Load Selects the palette load mode or the blink mode. 00 Disable both Blink mode and Palette Load mode. 01 Enables Palette Load mode. 10 Enables Blink 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 LUT. The ALL registers are not pipelined; do not modify their contents while the LUT is active. See the following Palette Load and Blink Procedure. The exception 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) is 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 enables blinking with more than two colors.

The palette is loaded during vertical blank (VBLANK) periods. Palette Load transfers begin at the leading−edge of the first VBLANK after the Palette Load (PL) bit is set. The color graphics controller clears PL immediately after completing a palette load.

All Blink transfers start with palette 0, and execute at the leading edge of the first VBLANK after the Blink Enable (BE) bit is set.

Palette Load and Blink Procedure

The 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 BE andPL bits of the BLINK register. Make sure that they are both cleared to 0.
2. Load the color palette into the frame buffer. If the desired LUT is not already resident, load the LUT into the frame buffer. Note the abovementioned 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 LUT specified in Step 2..
4. Enable either the blink mode or the palette load mode via the BE and PL bits of the BLINK register.

Programming the Bt458 RAMDAC (8-Bit Color)

This section discusses how to program the Brooktree Bt458 RAMDAC found in 8-bit color. For details on the Bt458 RAMDAC, see the *Brooktree® Product Databook*.

Using 8-bit color, the Bt458 has a 256 column x 24 row color palette. Programming the entire color palette requires 768 (256 columns x 3 8-bit rows) entries.

The Bt458 contains the following registers:

Table 6–4 Bt458 Registers

Address	Internal RAMDAC Address	Register	Type
FFF8 90C0		Address Register	Read/Write
FFF8 90C4		Color Palette	Read/Write
FFF8 90C8	04	Read Mask Register	Read/Write
	05	Blink Mask Register	Read/Write
	06	Command Register	Read/Write
	07	Control/Test Register	Read/Write
FFF8 90CC	00	Overlay Color 0	Read/Write
	01	Overlay Color 1	Read/Write
	02	Overlay Color 2	Read/Write
	03	Overlay Color 3	Read/Write

NOTE: Write the Internal RAMDAC Address into the Address register before writing to the color palette, RAMDAC registers or overlay planes.

NOTE: See the *Brooktree® Product Databook* for detailed descriptions of the RAMDAC registers.

The color palette and overlay palette are programmed using the RGB Mode described below and in the *Brooktree® Product Databook*.

To write data to the color palette, write the Internal RAMDAC Address to be modified to FFF8 90C0; then write the red, green, and blue data (in that order) to FFF8 90C4. When the blue data is written, the Bt458 writes all three color values to the palette simultaneously – therefore you must write all three colors to change a palette value. When the Bt458 updates the palette, it also increments the address pointer to the next palette location in preparation for another write or read. To write to or read from consecutive palette locations, it is only necessary to write the address of the first location.

Reading data from the color palette is similar to writing data. Write the Internal RAMDAC Address of the first location to be read to FFF8 90C0; then read FFF8 90C4 three times for the red, green, and blue data (in that order). The address pointer automatically increments after each access to permit sequential accesses.

Reading and writing the overlay data is similar to reading and writing the color data. Write the Internal RAMDAC Address to FFF8 90C0; then read or write FFF8 90CC. The address pointer automatically increments after each access to permit sequential accesses.

The Bt458 contains four control registers (Read Mask, Blink Mask, Command, and Test) as described in Table 6−5. To access a control register, write its Internal RAMDAC Address to FFF8 90C0; then read or write FFF8 90C8. The address pointer does not increment when you access the control registers.

Table 6–5 Bt458 Control Registers

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	Tests the data path through the RAMDAC. See the *Brooktree*R *Product Databook* for details.

NOTE: See the *Brooktree® Product Databook* for detailed descriptions of these registers.

RAMDAC Access Restrictions

Do not access the RAMDAC while performing an ALL (contention may result) or while drawing (test the DIP bit in CSR0 for 0). To test whether or not a drawing operation is in progress, execute the following C code:

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

Programming the Bt461 RAMDAC (24-Bit Color)

This section discusses how to program the Brooktree Bt461 RAMDACs found in 24-bit color. For details on the Bt461 RAMDAC, see the *Brooktree® Product Databook*.

Using 24-bit color, each of three Bt461s has a 1024 column x 8 row color palette. Programming the entire color palette of the three Bt461s requires 3072 (1024 columns x 3 Bt461s) 8-bit entries.

The Bt461 contains the following registers:

Table 6–6 Bt461 Registers

Address	Internal RAMDAC Address	Register	Type
FFF8 90C0		Address Register Low	Read/Write
FFF8 90C4		Address Register High	Read/Write
FFF8 90C8	0000 – 00FF	Alternate Color Palette	Read/Write
	0100	Overlay Color 0	Read/Write
	.	.	.
	010F	Overlay Color 15	Read/Write
	0110 – 011F	Unused	
	0200	ID Register	Read/Write
	0201	Command Register 0	Read/Write
	0202	Command Register 1	Read/Write
	0203	Command Register 2	Read/Write
	0204	Pixel Read Mask Register Low	Read/Write
	0205	Pixel Read Mask Register High	Read/Write
	0206	Pixel Blink Mask Register Low	Read/Write
	0207	Pixel Blink Mask Register High	Read/Write
	0208	Overlay Read Mask Register	Read/Write
	0209	Overlay Blink Mask Register	Read/Write
	020C	Test Register	Read/Write
FFF8 90CC		Primary Color Palette	Read/Write

NOTE: Write the Internal RAMDAC Address into the Address Registers (Low and High) before writing to the color palette, RAMDAC registers or overlay planes.

NOTE: See the *Brooktree® Product Databook* for detailed descriptions of these registers.

The color palette and overlay palette are programmed using the Normal Mode described below and in the *Brooktree® Product Databook*.

The Normal mode is a broadcast access of the RAMDAC registers, writing to all three registers within one write cycle. Write to or read RAMDAC registers as follows:

1. Write the Internal RAMDAC Address to the Address Registers (Low and High). Write the least-significant byte to FFF8 90C0 and the most-significant byte to FFF8 90C4. Within each system address, repeat the internal RAMDAC address in bytes 0, 1, and 2 so that the address is applied to all three RAMDACs.
2. Write to or read the appropriate system address (FFF8 90C8 for the alternate color palette, overlay, or RAMDAC registers, or FFF8 90CC for the primary color

palette). Assuming that byte 0 is the least-significant byte, write or read the data with the red data in byte 0, the green data in byte 1, and the blue data in byte 2.

3. The contents of the address registers automatically increment to point to the next location, therefore if you are writing or reading a block of data, repeat Step 2. as needed.

For example, if you want to write 40_{16} to Command Register 0 (Internal RAMDAC Address = 0201), write 00010101_{16} to FFF8 90C0; then write 00020202_{16} to FFF8 90C4; then write $00404040_{16\ to}$ FFF8 90C8.

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.

The RAMDAC registers and memory are not initialized during powerup.

Programming the Z–Buffer

The optional Z−buffer supports three−dimensional applications with hidden line and surface removal, and Hither and Yon clipping.

CAUTION: Rendering errors may result with a Z-value near zero. Either use the Z-buffer as a 23-bit buffer with bit 0 set to 1, or develop code to prevent a Z-value of zero.

The Z−buffer:

- Has a 25−MHz Mbus interface.
- Has selectable 16 or 24−plane Z−buffer using 256K x 4 DRAMS.
- Has fixed internal 24−bit resolution.
- Has fast rectangular Clear/Set mode with color graphics controller assistance.
- Has programmable Hither and Yon clipping planes.
- Suppors Stop/Resume operations (context switching).
- Memory organization and resolution can be configured to support all screen resolutions.
- The CPU has direct read/write access to the Z−buffer array.

Components of the Z–Buffer

As shown in Figure 6−7, the Z−buffer gate array is partitioned into four modules: an Mbus interface, Z−controller, Datapath/ALU, and a memory controller.

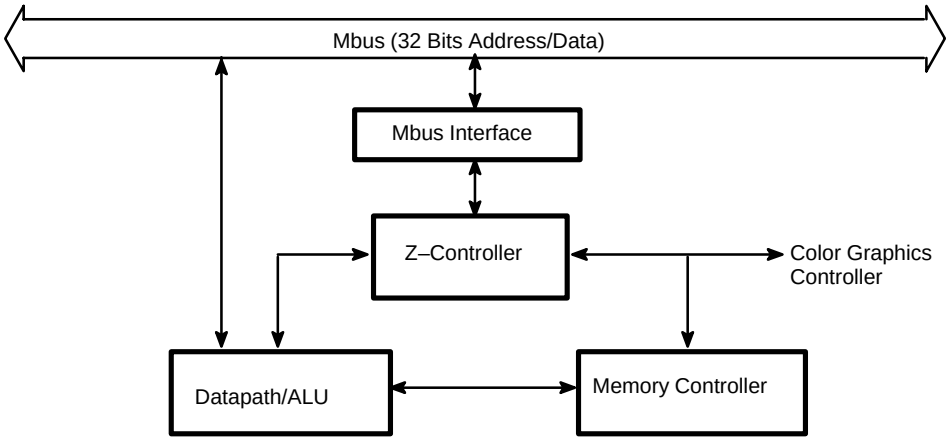


Figure 6–7 The Z–Buffer

Mbus Interface

The Mbus interface enables 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.

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

Z–Controller

The Z−controller monitors and interprets the Z−protocol bits. Depending on the state and sequence of these bits, the Z−controller steers data and ALU opcodes within the datapath. 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. It provides data paths to read and write the Z−buffer, update Z−values, and compare new Z-values with clipping planes and old Z-values. The Mbus, Z-controller, and memory controller share the datapath. The Mbus uses it to access registers, the Z−controller uses it to increment/decrement Z−values, and the memory controller uses it to buffer 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 to steer and latch Z−data within the Datapath unit. The color graphics controller initiates all activities of the memory controller. 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 accesses the Z−buffer. These accesses occur in parallel to, and are sychronized with color graphics controller frame buffer accesses.

NOTE: For all Z−buffer accesses, the color graphics controller and the Z−buffer both perform memory cycles.

The Z−buffer memory array consists of 4K x 4K pixels located at 8800 0000 − 8FFF FFFC.

Configuration Logic

The configuration logic controls Z−buffer functions that depend on the graphics configuration. The host CPU loads the configuration parameters into the Configuration (CNFG) register when the Z−buffer is initialized.

Programming the Z–Buffer Registers

The Z−buffer has thirteen 32−bit read/write registers as shown in Table 6−7.

NOTES: When writing to the Z−buffer registers, use broadcast addresses.

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

Program the Z−buffer Configuration (ZCNFG) register before reading or writing any of the other Z−buffer registers. You may write/read the remaining registers in any order.

All Z−values are twos complement numbers that represent Z−space (Hither − Yon). The Z−buffer depth can be either 16−bits or 24−bits; the 16−bit depth range is 8000 − 7FFF and the 24−bit depth range is 80 0000 − 7F FFFF. 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.

The Z−buffer registers are located at FFF8 90E0 − FFF8 91F0.

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:

1. Reload the color graphics controller and Z−buffer registers.
2. Issue a Resume command to the color graphics controller (set the RES bit in the STOP Register).

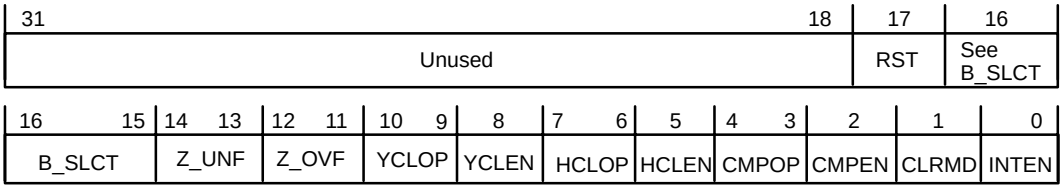
Table 6–7 Z–buffer Registers

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

FFF8 90E0 Read/Write

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



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. [1] 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.[1] 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.

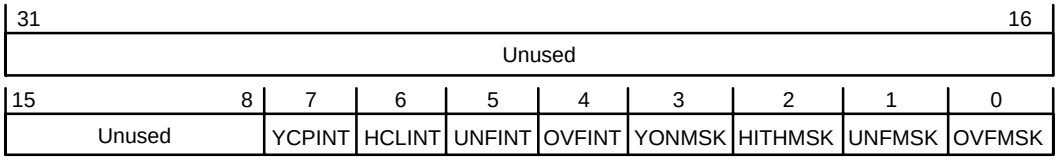
[1] A new pixel and Z−value will replace the existing pixel and Z−value only when the following condition is true — Z−Compare Win & Hither Clip Win & Yon Clip Win.

(concluded)

ZIR Z–Buffer Interrupt

FFF8 90E4 Read/Write

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

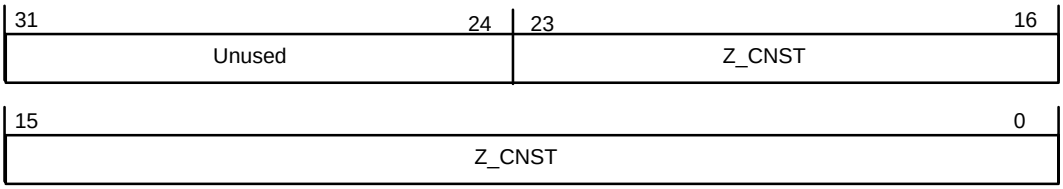


Bit	Mnemonic	Function
31–8	Unused	Must be written as 0s and read as undefined.
7	YCPINT	Yon Clip Interrupt Indicates whether or not a clip against the yon plane has occurred. A read of the IR register resets this bit. 0 No clip. 1 Clip has occurred.
6	HCLINT	Hither Clip Interrupt Indicates whether or 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

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.

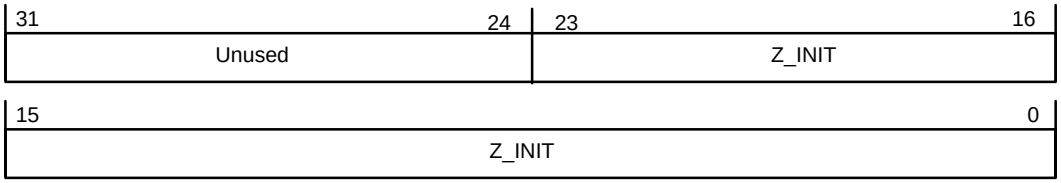


Bit	Mnemonic	Function
31−24	Unused	Must be written as 0s and read as undefined.
23−0	Z_CNST	Z Constant Twos complement value (bit 23 is the sign bit). For 16−plane Z−buffers, bits 23−8 contain the 16−bit Z_CNST and bits 7−0 are ignored.

Z_INIT Z–Depth Initial

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.

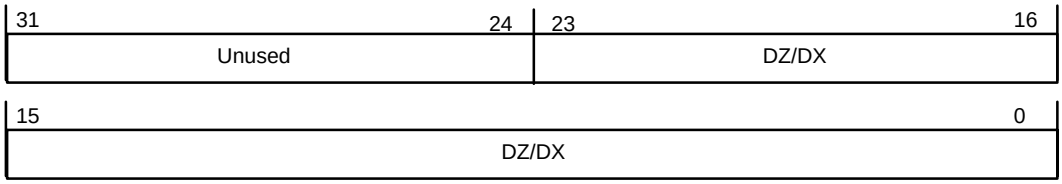


Bit	Mnemonic	Function
31−24	Unused	Must be written as 0s and read as undefined.
23−0	Z_INIT	Initial Z−Value Z_INIT twos complement value (bit 23 is the sign bit).

Z_XINC — DZ/DX

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.

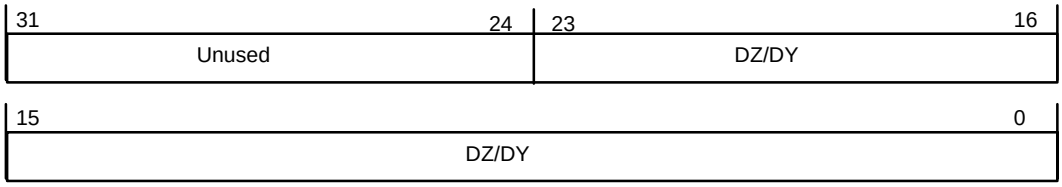


Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	DZ/DX	Slope of Z With Respect to X Twos complement Z–value (bit 23 is the sign bit). Specify all 24 bits for both 16–bit and 24–bit Z–buffers.

Z_YINC — DZ/DY

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.

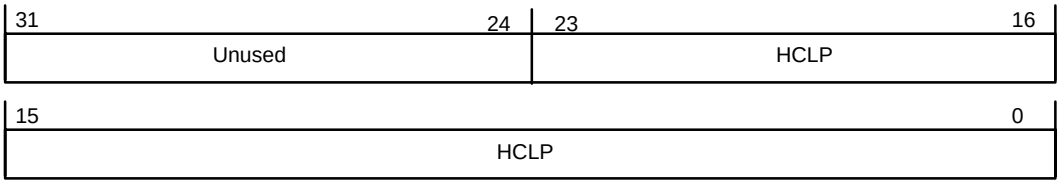


Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	DZ/DY	Slope of Z With Respect to Y Twos complement Z–value (bit 23 is the sign bit). Specify all 24 bits for both 16–bit and 24–bit Z–buffers.

HCLP — Hither Clip

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).

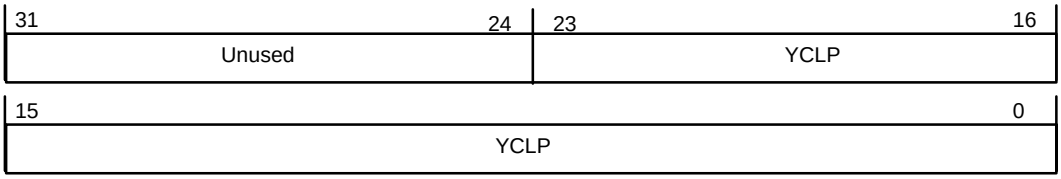


Bit	Mnemonic	Function
31−24	Unused	Must be written as 0s and read as undefined.
23−0	HCLP	Hither Clip Twos complement Z−value (bit 23 is the sign bit).

YCLP — Yon Clip

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).

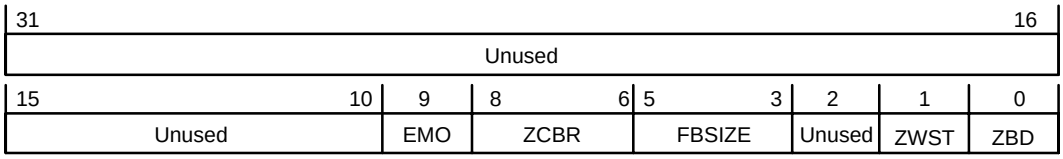


Bit	Mnemonic	Function
31−24	Unused	Must be written as 0s and read as undefined.
23−0	YCLP	Yon Clip Twos complement Z−value (bit 23 is the sign bit).

ZCNFG Z–Buffer Configuration

FFF8 91E0 Read/Write

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

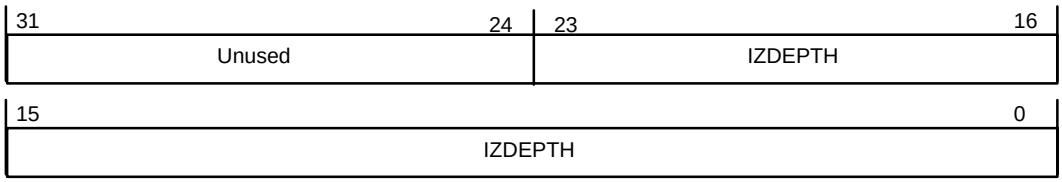


Bit	Mnemonic	Function
31−10	Unused	Must be written as 0s and read as undefined.
9	EMO	Enable Memory Output Enables/disables memory outputs. A reset enables memory outputs. 0 Disables memory outputs. 1 Enables memory outputs.
8−6	ZCBR	Z−Buffer CBR Cycles Must match the number of CBR (Cas−Before−Ras) refresh 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	
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

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.

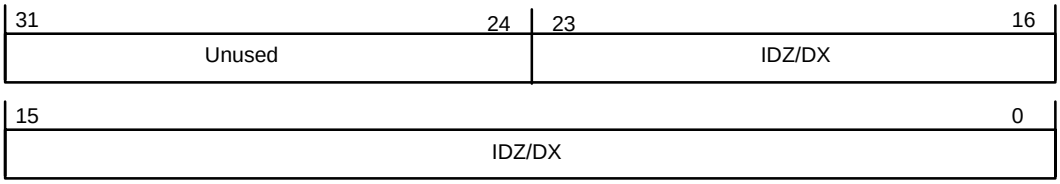


Bit	Mnemonic	Function
31−24	Unused	Must be written as 0s and read as undefined.
23−0	IZDEPTH	Interrupted Zbuffer Depth. Twos complement Z−value (bit 23 is the sign bit).

ST1 State 1

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.

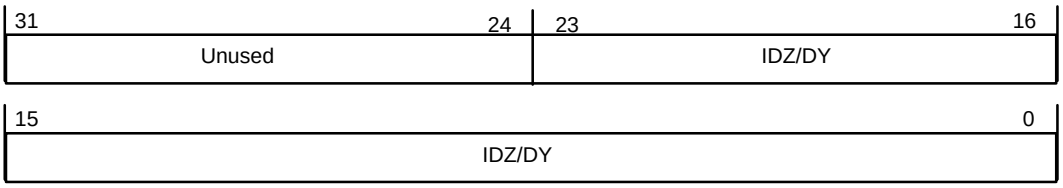


Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	IDZ/DX	Interrupted Slope of Z with Respect to X Twos complement Z–value (bit 23 is the sign bit).

ST2 State 2

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.

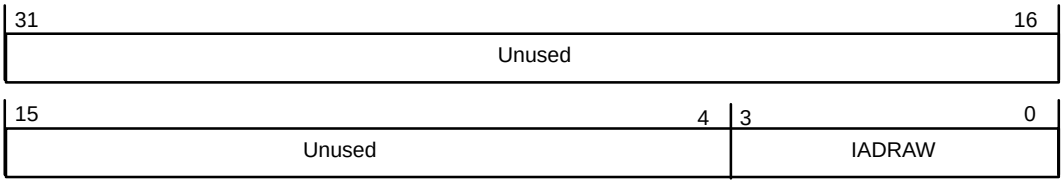


Bit	Mnemonic	Function
31–24	Unused	Must be written as 0s and read as undefined.
23–0	IDZ/DY	Interrupted Slpoe of Z with Respect to Y Twos complement Z–value (bit 23 is the sign bit).

ST3 State 3

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.

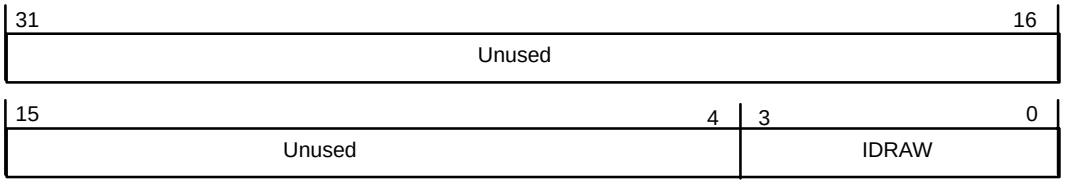


Bit	Mnemonic	Function
31–4	Unused	Must be written as 0s and read as undefined.
3–0	IADRAW	Interrupted Draw Command Command Opcode (see the color graphics Command register for the opcodes).

SHADOW — Shadow

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.



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 for the opcodes).

End of Chapter

Chapter 7
Programming the Keyboard Interface

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

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.

Keyboard Interface

As shown in Figure 7–1, the keyboard interface consists of a UART, address decode and control logic, and clock and timing logic.

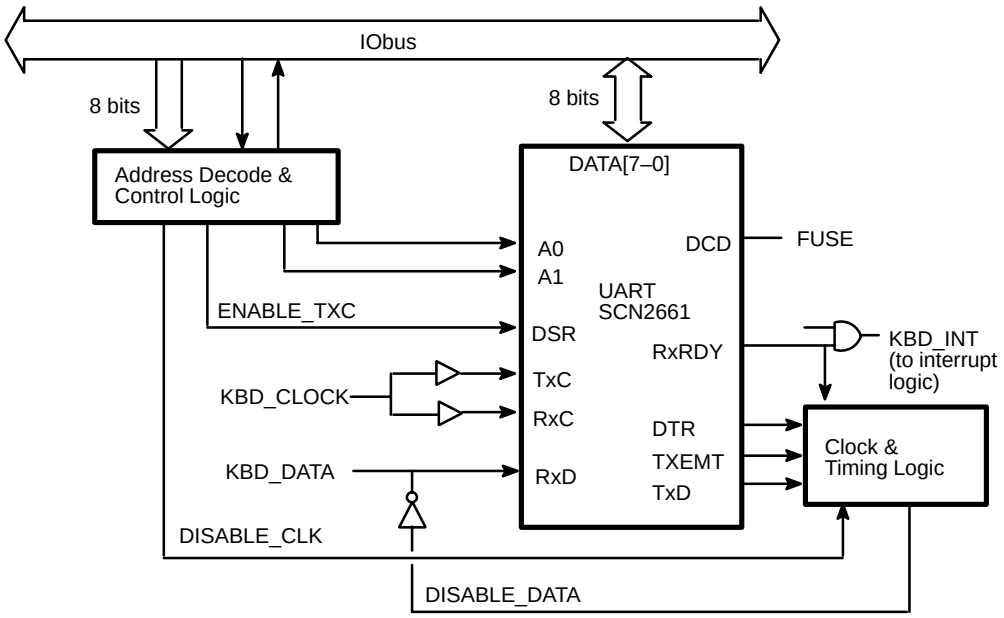


Figure 7–1 Keyboard Interface

UART

The UART is a Signetics SCN2661 enhanced programmable communications interface controller. It receives serial data from the keyboard and converts it to parallel data. It also converts parallel data to serial and transmits it to the keyboard.

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.

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.

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 7−1 defines the states of the clock and data lines.

Table 7–1 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 7–2. 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 7–2 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 7–3 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 7–3 Keyboard Registers

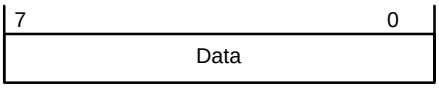
Address	Register	Type
FFF8 2800	Receive Holding Register (RxHR)	Read
	Transmit Holding Register (TxHR)	Write
FFF8 2804	Status Register (SR)	Read
FFF8 2808	Mode Register 1 (MR1)	Read/Write
	Mode Register 2 (MR2)	Read/Write
FFF8 280C	Command Register (CR)	Read/Write
FFF8 2810	Disable Clock (DSC)	Write
FFF8 2820	Enable Transmit Clock (ETxC)	Write

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

RxHR Receive Holding Register

FFF8 2800 Read Only

The Receive Holding Register (RxHR) 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 7−4 identifies the response codes.



Bit	Mnemonic	Function
7−0	Data	Holds data sent from the keyboard for the CPU.

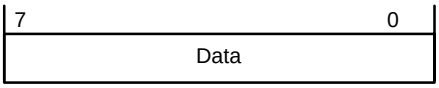
Table 7–4 Keyboard Responses

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 (02_{16} = set 2, 03_{16} = set 3).

TxHR Transmit Holding Register

FFF8 2800 Write Only

The Transmit Holding Register (TxHR) temporarily stores commands to be transmitted to the keyboard. Table 7–5 lists the command codes.



Bit	Mnemonic	Function
7–0	Data	Holds data sent from the CPU to the keyboard.

Table 7–5 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 ms, 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 7–5 Commands

Code (hex)	Command
F3	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 **Bit** **Meaning** 7 Always 0 6, 5 Delay The keyboard adds 1 to this number and divides the sum by 4 to obtain the delay period. Thus, the delay period may vary from 250 ms (00_2) to 1 s (11_2). The default delay is 500 ms (01_2). 4, 3 Rate Exponential component (B) of the rate calculation. 2−0 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 (11111_2) to 30 Hz (00000_2). The default is 10.9 Hz (01011_2). 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%.
ED	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. **Bit** **LED** 7−3 Reserved 2 Caps lock 1 Num lock 0 Scroll lock 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.
F2	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.
F0	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.

(continued)

Table 7–5 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:

Typematic Command | Break Code Function | Sending
F7 | enable | disable
F8 | disable | enable
F9 | disable | disable
FA | enable | enable

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 7−2 illustrates the keys on a keyboard, and Table 7−6 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.

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.

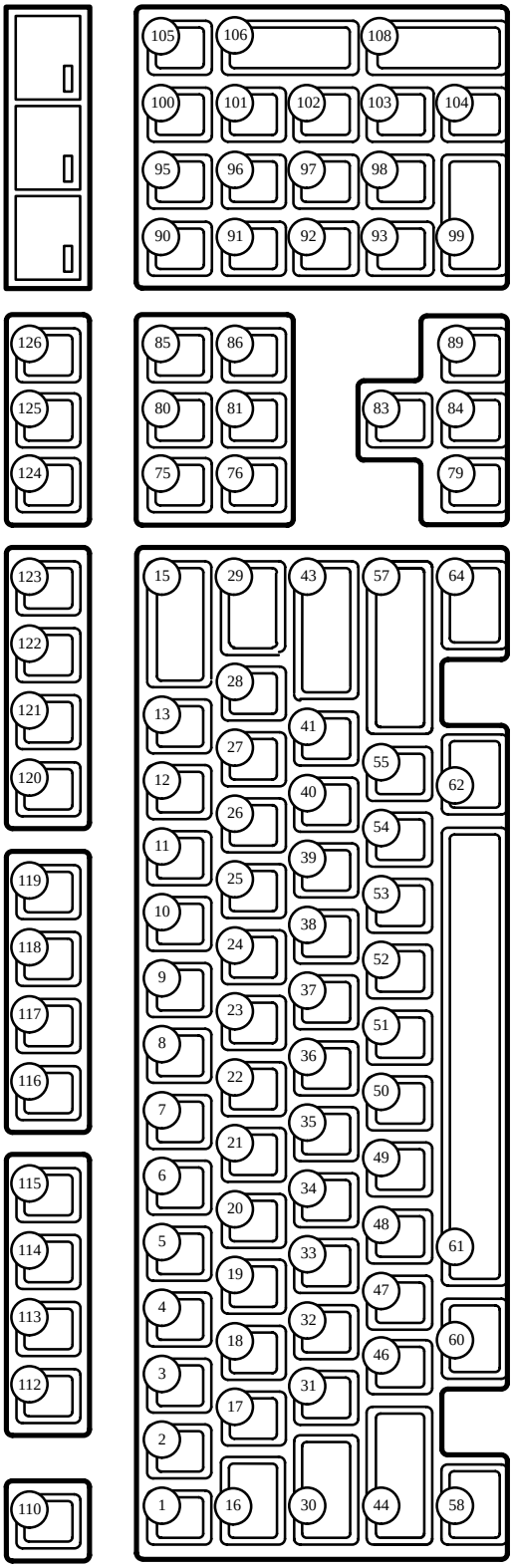


Figure 7–2 Position of Keys on Keyboard

Table 7–6 Scan Code Sets 2 and 3

Key #	Scan Code Set 2 Codes		Scan Code Set 3 Codes		Default Key State
	Make	Break	Make	Break	
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 7–6 Scan Code Sets 2 and 3

Key [3]	Scan Code Set 2 Codes Make	Scan Code Set 2 Codes Break	Scan Code Set 3 Codes Make	Scan Code Set 3 Codes Break	Default 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 7–6 Scan Code Sets 2 and 3

Key [3]	Scan Code Set 2 Codes Make	Scan Code Set 2 Codes Break	Scan Code Set 3 Codes Make	Scan Code Set 3 Codes Break	Default 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)

SR Status Register

FFF8 2804 Read Only

The Status Register (SR) contains the receiver, transmitter, and control signal status for the keyboard interface.

7	6	5	4	3	2	1	0
ETC	1	FE	OE	PE	STC	RxRDY	TxRDY

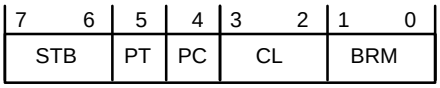
Bit	Mnemonic	Function
7	ETC	Enable Transmit Clock register (ENABLE_TXC) Status 0 Indicates that the TxC input is enabled. 1 Indicates that the TxC input is disabled.
6	FUSE	Fuse 0 The keyboard fuse is defective. 1 The keyboard fuse is good.
5	FE	Framing Error 1 Indicates that a framing error occurred.
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.

MR1, MR2 — Mode Register

FFF8 2808 — Read/Write

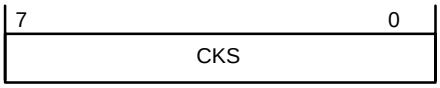
The Mode Registers (MR1 and MR2) program the UART mode of operation. The first time a program reads or writes the register's address, it accesses MR1. The second time, it accesses MR2. Any subsequent reads or writes recycle the UART internal sequencer between these two registers.

MR1:



Bit	Mnemonic	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.

MR2:



Bit	Mnemonic	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.

CR Command Register

FFF8 280C Read/Write

The Command Register (CR) contains commands for the keyboard interface. Reading CR returns the current command status, and writing to it defines the command status.

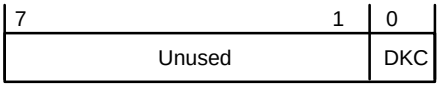
7 6	5	4	3	2	1	0
OPM	RTS	RE	FB	RxEN	IKT	TxEN

Bit	Mnemonic	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

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.



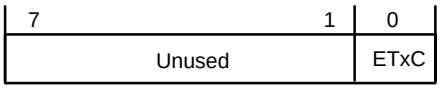
Bit	Mnemonic	Function
7–1	Unused	
0	DKC	Disable Keyboard Clock 0 Inhibit the keyboard from transmitting data by forcing the keyboard clock low. 1 Allow the keyboard to transmit data by releasing the keyboard clock line.

ETxC — Enable Transmit Clock

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.

NOTE: The ETxC bit is set to 1 by keyboard reset.



Bit	Mnemonic	Function
7–1	Unused	
0	ETxC	Enable Transmit Clock 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.

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 7−3). 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 the 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 keyboard 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.

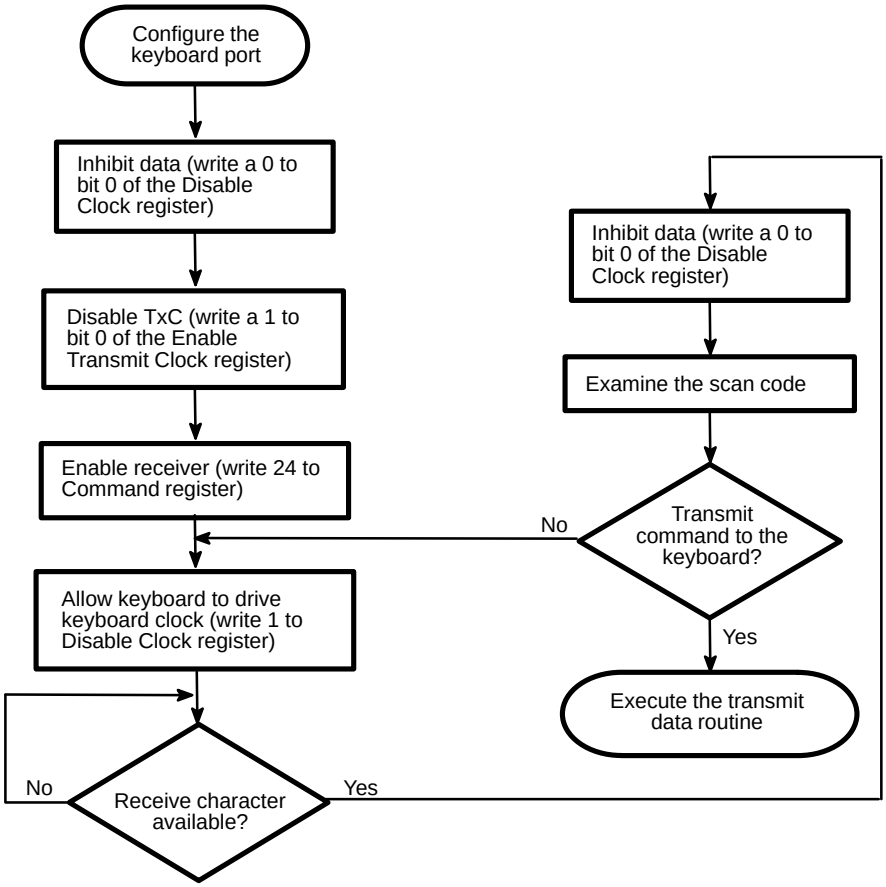


Figure 7–3 Receiving Data from the Keyboard

Transmitting Data to the Keyboard

To transmit data to the keyboard, the keyboard service routine must use the following format (see also Figure 7−4):

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 ms. (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 ms.
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 ms.
11. Enable TxC by writing a 0 to the Enable Transmit Clock register and waiting a minimum of 60 ms. (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.”

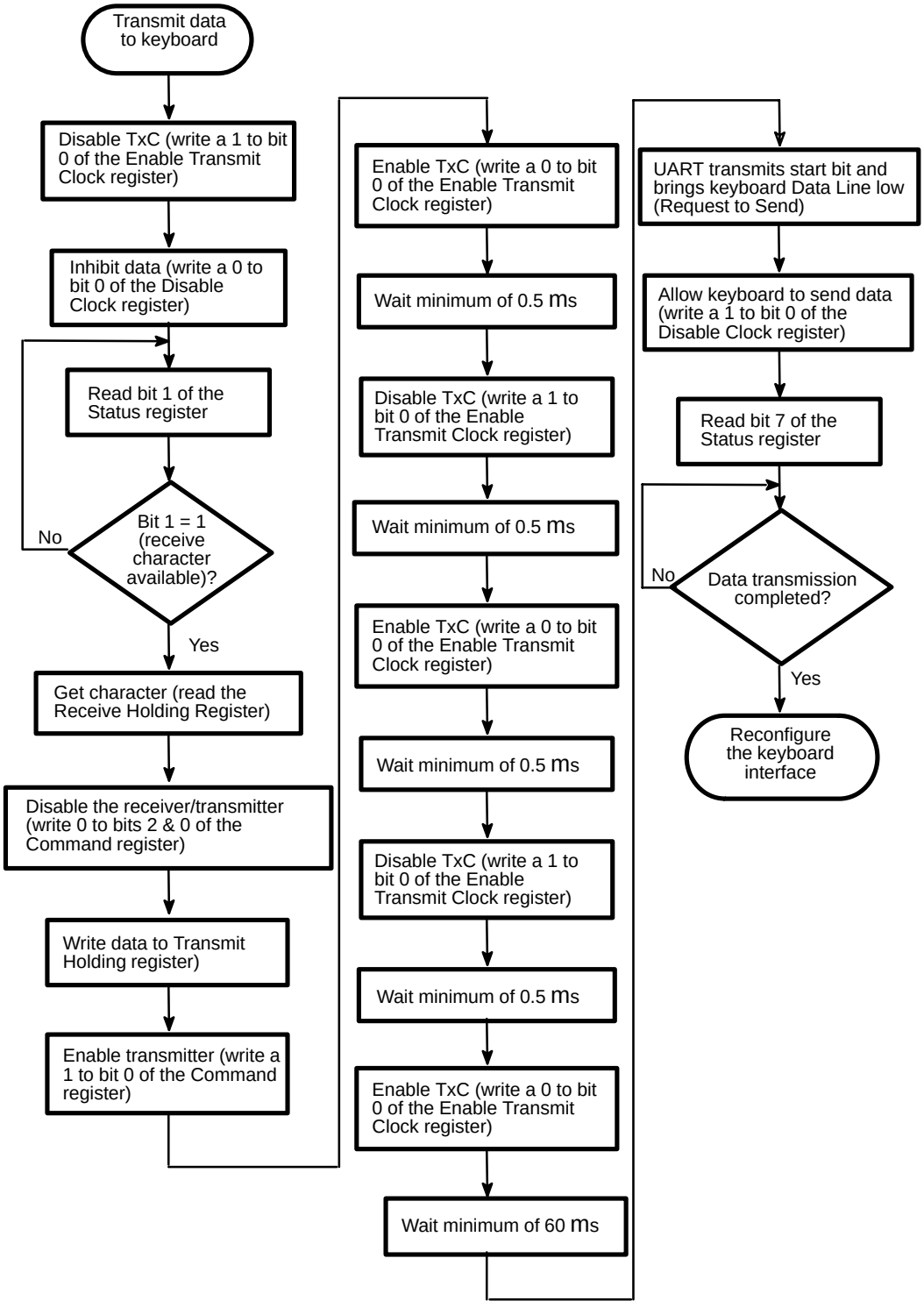


Figure 7–5 Transmitting Data to the Keyboard

End of Chapter

Chapter 8
Programming the Serial and Parallel Interfaces

This chapter describes:

- the serial and parallel interfaces,
- how to program the asynchronous serial interface,
- how to program the synchronous serial interface,
- how to program the mouse interface,
- how to program the speaker, and
- how to program the parallel interface.

Asynchronous Serial Interface

The asynchronous serial interface has two Signetics SCC2692 Dual Asynchronous Receiver/Transmitters (DUARTs). Channel A of DUART1 is for a system console and channel B is for asynchronous serial port A. Channel A of DUART2 is for a mouse and channel B is for asynchronous serial port B. The clock (CLK) input to each DUART is 3.6864 Mhz.

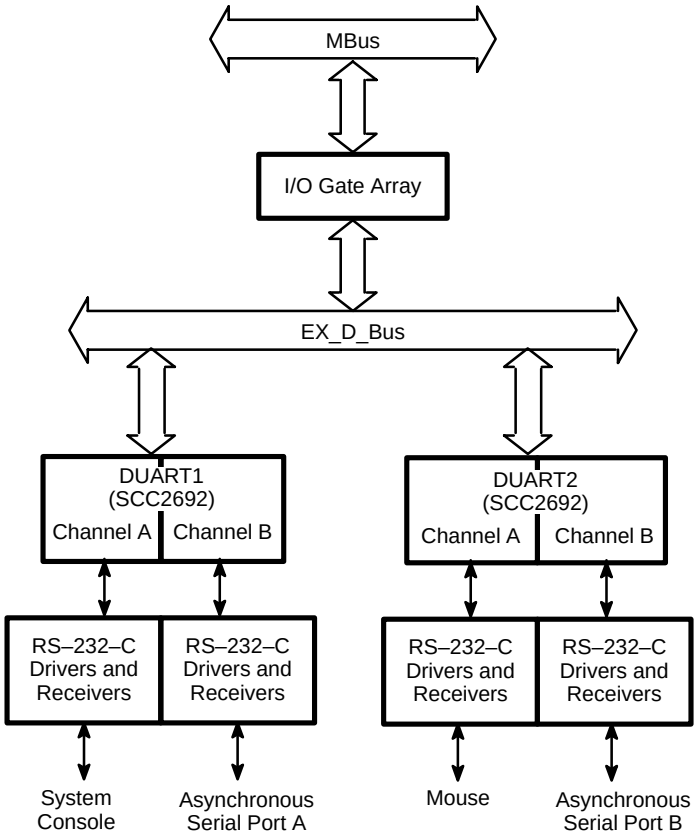


Figure 8–1 Asynchronous Serial Interface

Synchronous Serial Interface

The synchronous serial interface consists of a Signetics SCN68562 Dual Universal Serial Communications Controller (DUSCC). The DUSCC has two channels, channel A for synchronous serial port A and channel B for synchronous serial port B. The Peripheral Clock (PCLK) input is 14.7456 Mhz.

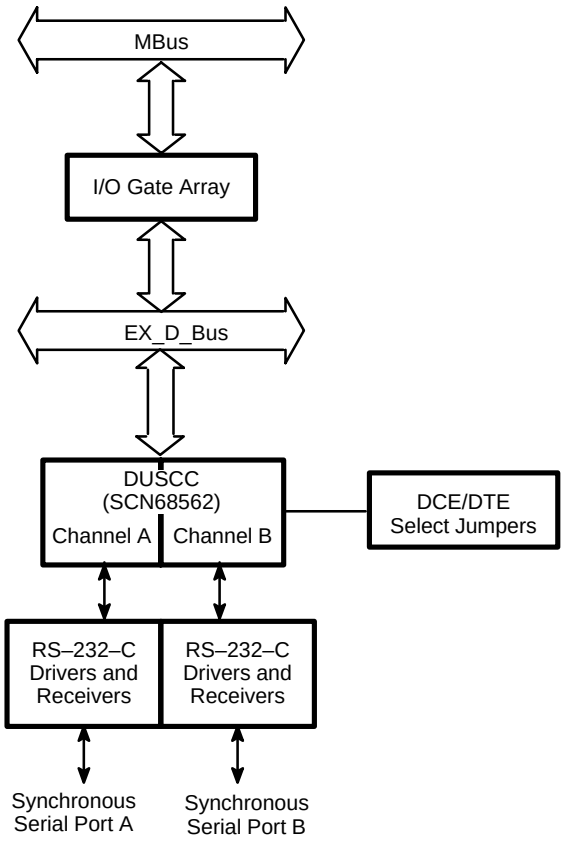


Figure 8–2 Synchronous Serial Interface

Parallel Interface

The parallel interface transmits data to a parallel device using the Centronics protocol.

See Appendix C for information on the parallel connector.

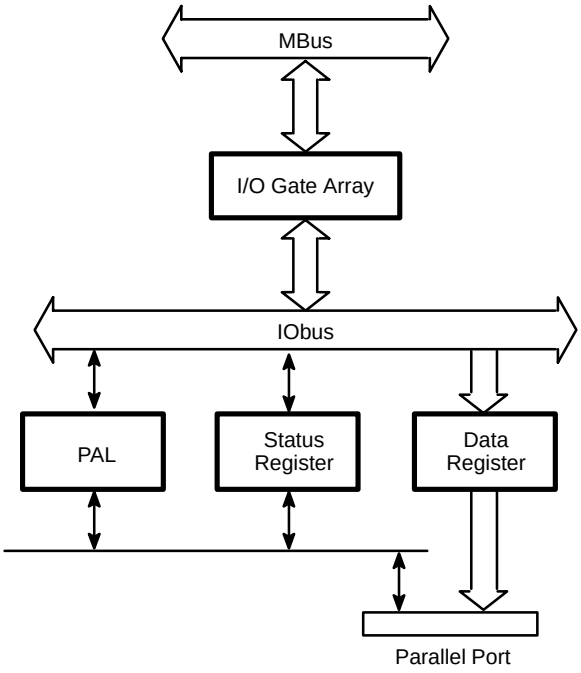


Figure 8–3 Parallel Interface

Programming the Asynchronous Serial Interface

The asynchronous serial interface uses a Signetics SCC2692 Dual Asynchronous Receiver/Transmitter (DUART).

The registers are accessed as words aligned on word boundaries. Table 8−1 lists the DUART registers and addresses.

The rest of this section describes how to initialize, program and reset the asynchronous serial interface. Use this section in conjunction with the Signetics *SCC2692 Dual Asynchronous Receiver/Transmitter (DUART)* specification to program the asynchronous serial interface.

Initializing the Asynchronous Serial Interface

No special initialization process is necessary except to enable the speaker by programming the counter/timer. The speaker is described later in this chapter.

Resetting the Asynchronous Serial Interface

A reset 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.

Asynchronous Serial Interrupts

The DUART generates several types of interrupts, but it has only one interrupt line which sets the DUART Interrupt (DI) bit in the Interrupt Status (IST) register. The interrupt service routine must read the DUART's Interrupt Status Register (ISR) to find the interrupt source. Any of the following conditions can generate an interrupt.

- Channel A or B transmitter is ready.
- Channel A or B receiver is ready, or its FIFO buffer is full.
- Channel A or B detected a Break Character.
- The counter/timer has reached its terminal count or zero.
- The input port changed state.

Interrupts may be masked via the DUART's Interrupt Mask Register (IMR). When an interrupt is masked, the DUART will not set the bit in the ISR, and consequently will not assert an interrupt request for that interrupt.

Table 8–1 Asynchronous Serial Interface Registers

Register	Address
DUART1 – Channel A – System Console	
Read Registers / Write Registers	
MR1A, MR2A / MR1A, MR2A	FFF8 2000
SRA / CSRA	FFF8 2004
Reserved / CRA	FFF8 2008
RHRA / THRA	FFF8 200C
IPCR / ACR	FFF8 2010
ISR / IMR	FFF8 2014
CTU / CTUR	FFF8 2018
CTL / CTLR	FFF8 201C
DUART1 – Channel B – Asynchronous Serial Port A	
MR1B, MR2B / MR1B, MR2B	FFF8 2020
SRB / CSRB	FFF8 2024
Reserved / CRB	FFF8 2028
RHRB / THRB	FFF8 202C
Reserved / Reserved	FFF8 2030
Input port / OPCR	FFF8 2034
Start Counter / Set Output Bits	FFF8 2038
Stop Counter / Reset Output Bits	FFF8 203C
DUART2 – Channel A – Mouse	
Read Registers / Write Registers	
MR1A, MR2A / MR1A, MR2A	FFF8 2040
SRA / CSRA	FFF8 2044
Reserved / CRA	FFF8 2048
RHRA / THRA	FFF8 204C
IPCR / ACR	FFF8 2050
ISR / IMR	FFF8 2054
CTU / CTUR	FFF8 2058
CTL / CTLR	FFF8 205C
DUART2 – Channel B – Asynchronous Serial Port B	
MR1B, MR2B / MR1B, MR2B	FFF8 2060
SRB / CSRB	FFF8 2064
Reserved / CRB	FFF8 2068
RHRB / THRB	FFF8 206C
Reserved / Reserved	FFF8 2070
Input port / OPCR	FFF8 2074
Start Counter / Set Output Bits	FFF8 2078
Stop Counter / Reset Output Bits	FFF8 207C

Programming the Synchronous Serial Interface

The synchronous serial interface has a Signetics SCN68562 Dual Universal Synchronous Communications Controller (DUSCC). Use this section in conjunction with Signetics' *SCN68562 Dual Universal Serial Communications Controller (DUSCC)* specification to program the synchronous serial interface.

The DUSCC supports polled or interrupt driven PIO transfers and half/full−duplex DMA transfers. Half−duplex DMA transfers are on DMA channel 0 for port A, and channel 2 for port B. Full−duplex DMA transfers are on DMA channels 0/1 for port A receive/transmit respectively, and on DMA channels 2/3 for port B receive/transmit respectively.

NOTE: The DUSCC does not support 5−bit synchronous data transfers.

Table 8–2 Synchronous Serial Interface Registers

Register	Address
Port A	
CMR1A	FFF8 D000
CMR2A	FFF8 D004
S1RA	FFF8 D008
S2RA	FFF8 D00C
TPRA	FFF8 D010
TTRA	FFF8 D014
RPRA	FFF8 D018
RTRA	FFF8 D01C
CTPRHA	FFF8 D020
CTPRLA	FFF8 D024
CTCRA	FFF8 D028
OMRA	FFF8 D02C
CTHA	FFF8 D030
CTLA	FFF8 D034
PCRA	FFF8 D038
CCRA	FFF8 D03C
TxFIFOA	FFF8 D040
RxFIFOA	FFF8 D050
RSRA	FFF8 D060
TRSRA	FFF8 D064
ICTSRA	FFF8 D068
IERA	FFF8 D070

(continued)

Table 8–2 Synchronous Serial Interface Registers (continued)

Register	Address
General	
GSR	FFF8 D06C
IVR	FFF8 D078
IVRM	FFF8 D0F8
ICR	FFF8 D07C
DTRTA	FFF8 D074
Port B	
CMR1B	FFF8 D080
CMR2B	FFF8 D084
S1RB	FFF8 D088
S2RB	FFF8 D08C
TPRB	FFF8 D090
TTRB	FFF8 D094
RPRB	FFF8 D098
RTRB	FFF8 D09C
CTPRHB	FFF8 D0A0
CTPRLB	FFF8 D0A4
CTCRB	FFF8 D0A8
OMRB	FFF8 D0AC
CTHB	FFF8 D0B0
CTLB	FFF8 D0B4
PCRB	FFF8 D0B8
CCRB	FFF8 D0BC
TxFIFOB	FFF8 D0C0
RxFIFOB	FFF8 D0D0
RSRB	FFF8 D0E0
TRSRB	FFF8 D0E4
ICTSRB	FFF8 D0E8
IERB	FFF8 D0F0
DTRTB	FFF8 D0F4

(continued)

Table 8–2 Synchronous Serial Interface Registers (continued)

Register	Address
DMA0 – Synchronous Serial Channel A Receive	
SG	FFF8 D804
AD	FFF8 D808
CNT	FFF8 D810
CMD	FFF8 D820
STAT	FFF8 D840
DMA1 – Synchronous Serial Channel A Transmit	
SG	FFF8 D904
AD	FFF8 D908
CNT	FFF8 D910
CMD	FFF8 D920
STAT	FFF8 D940
DMA2 – Synchronous Serial Channel B Receive	
SG	FFF8 DA04
AD	FFF8 DA08
CNT	FFF8 DA10
CMD	FFF8 DA20
STAT	FFF8 DA40
DMA3 – Synchronous Serial Channel B Transmit	
SG	FFF8 DB04
AD	FFF8 DB08
CNT	FFF8 DB10
CMD	FFF8 DB20
STAT	FFF8 DB40

(concluded)

Programming Exceptions

The following programming exceptions should be used in conjunction with the Signetics manual.

CMR2 – Channel Mode Register 2

CMR2A and CMR2B configure the data transfer interface to the synchronous communication channels. Bits 5−3 specify the type of data transfer between the Rx and Tx FIFOs and the CPU.

Set bits [5−3] to one of the following:

001 = Half−duplex dual address DMA
011 = Full−duplex dual address DMA
111 = Polled or interrupt

The wait on transmit and/or receive should not be used; they would lock the system bus while waiting for a serial character.

PCR – Pin Configuration Register

PCRA and PCRB configure the multi−purpose pins of the synchronous communication channels.

Set the PCR bits as follows:

X2/IDC (bit) = 0	The X2/IDC pin is used as a crystal.
GPO2/RTS = 0	RTS is a general purpose output.
SYNO/RTS = 0	SYNOUT/RTS is asserted one bit time after a SYN pattern.
RTxC = 00	The RTxC pin is an input.
TRxC = 110 (6)	The TRxC pin is an output for the transmitter shift register.

ICTSR – Input and Counter/Timer Status Register

ICTSRA and ICTSRB provide the current state of DSR and DTR. Two general−purpose inputs provide status on the modem handshakes: GPI1 monitors DSR, and GPI2 monitors DTR. Bit 0 provides the current state of DTR, and Bit 1 provides the current state of DTR. DTR cannot be written directly; use the DTR Toggle register to change the state of DTR.

DMA Channel 0 and 2 Writes to Memory

Scatter/gather is not available on channel 0 or 2.

When writing a block of serial data to memory via DMA channel 0 or channel 2, occasionally the last byte may not be transmitted. To check for and correct this condition, do the following:

1. Check the byte count. If the difference between the original programmed byte count and the current byte count is not a multiple of 4, continue with the following steps. If the difference is a multiple of 4, disregard the following steps; all of the data was written to memory.
2. Set the DMA Byte Count (CNT) to 1.
3. Clear (to 0) the appropriate DMA flush bit (FDMA0 or FDMA3) of the Memory Diagnostic Control (MDC) register (see Chapter 5, “Memory”). This will cause the DMA controller to write the remaining byte to memory, completing the data transfer.
4. Set the DMA flush bit to 1.

Toggling DTR

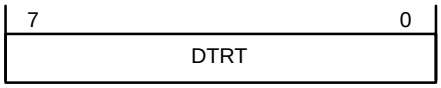
The following registers, DTRTA and DTRTB are used to toggle the DTR line of port A and port B respectively. DTRTA and DTRTB are located in the io gate array; they are not part of the Signetics serial controller.

DTRTn — DTR Toggle

FFF8 D074 **DTRTA** **Read/Write**

FFF8 D0F4 **DTRTB**

The DTR Toggle (DTRTA and DTRTB) registers toggle the current state of DTR.



Bit	Name	Function
7–0	DTRT	DTR Toggle Writing any value toggles the current state of DTR. When read, it returns all 1s.

Programming the Mouse Interface

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 Interface

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 8–3 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 ms. 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 Speaker

The system has a speaker that is driven via the Speaker Enable (SPKEN) register; the speaker pitch is controlled via OP3 (port B) of DUART2.

Table 8–4 identifies the registers used to program the speaker.

Table 8–4 Speaker Registers

Register	Address
ACR	FFF8 2050
CTUR	FFF8 2058
CTLR	FFF8 205C
OPCR	FFF8 2074
SPKEN	FFF8 3104

For further information on these registers, see the following pages as well as the Signetics manual, *SCC2692 Dual Asynchronous Receiver/Transmitter (DUART)*.

This section continues with descriptions of the registers used to program the speaker.

SPKEN — Speaker Enable

FFF8 3104 — Write Only

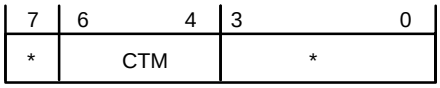
The Speaker Enable (SPKEN) register is used to generate an audible signal from the speaker. Any value written to SPKEN will generate a 200 ms speaker tone. To extend the duration of the tone, write to SPKEN before the previous tone has ended.

Bit	Mnemonic	Function
31–0	SPKEN	Speaker Enable Writing any value to SPKEN will generate a 200 ms signal to the speaker.

ACR Auxiliary Control

Address FFF8 2010 Write Only

The Auxiliary Control Register (ACR) contains speaker configuration parameters.



Bit	Mnemonic	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.

CTUR:

7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8

Bit	Mnemonic	Function
7−0	C15−C8	Counter/Timer Bits 15−8

CTLR:

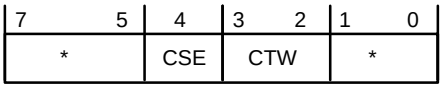
7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Bit	Mnemonic	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.



Bit	Mnemonic	Function
7–5	*	Used by serial and parallel ports.
4	Unused	
4–2	OP3	Output Port 3 Used to configure the counter/timer waveform of OP3. 01 Selects the C/Toutput. OP3 outputs the waveform generated by the counter/timer. Use the timer mode to generate the waveform.
1, 0	*	Used by serial ports.

Programming the Parallel Printer Interface

This section explains how to transmit data to a parallel printer.

Transmitting Data

The printer interface has two modes to transfer data: Parallel I/O (PIO) and Direct Memory Access (DMA). In PIO mode, the driver routine sets up the Printer Status and Control (PSC) register; then repeatedly writes data to the Printer Data (PD) register. In DMA mode, the driver routine sets up the PSC register; then enables the data transfer through the Parallel DMA (PDMA) channel.

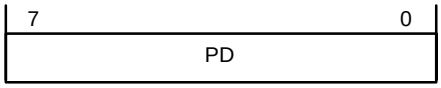
Registers

The Printer Status and Control (PSC) and Printer Data (PD) registers control the printer port. These regsters are defined as follows.

PD — Printer Data

FFF8 DF00 — **Write Only**

The Printer Data (PD) register transmits data to the printer. The parallel interface automatically processes and transmits data when PD is written to.



Bit	Mnemonic	Function
7–0	PD	Printer Data Contains the byte of data that the parallel interface will transmit to the printer.

PSC Printer Status and Control

FFF8 DF04 Read/Write

The Printer Status and Control (PSC) register provides parallel interface status when read, andconfigures the parallel interface when written. When the parallel port asserts an interrupt request, the interrupt service routine should read PSC.

15 10	9	8	7	6	5	4	3	2	1	0
Unused	PRST	PIEN	DMAEN	1	PINT	PFLT	PACK	PPE	PSEL	PBSY

Bit	Mnemonic	Function
15–10	Unused	
9	PRST	Printer Reset (Read/Write) Writing sets or resets the printer, reading provides the status of PRST. 1 Writing a 1 resets the printer. 0 Writing a 0 enables the printer.
8	PIEN	Printer Interrupt Enable (Read/Write) Writing enables or disables the printer interrupt source, reading provides the status of PIEN. 1 Enable the printer interrupt source. 0 Disable the printer interrupt source.
7	DMAEN	DMA Enable (Read/Write) Writing enables or disables DMA access to the printer, reading provides the status of DMAEN. 1 Enables DMA access to the printer. 0 Disables DMA access to the printer.
6	1	Must be set to 1
5	PINT	Printer Interrupt (Read only) 1 A printer interrupt request is asserted. 0 A printer interrupt request is not asserted.
4	PFLT	Printer Fault (Read only) 1 There is no printer. fault. 0 A printer fault. has occurred.
3	PACK	Printer (Read only) 1 The printer demand (or acknowledgement) is asserted. 0 The printer demand (or acknowledgement) is not asserted.
2	PPE	Printer Paper Empty (Read only) 1 The printer is out of paper. 0 The printer is not out of paper.
1	PSEL	Printer Select (Read only) 1 The printer is on line. 0 The printer is off line.
0	PBSY	Printer Busy (Read only) 1 The printer is busy. 0 The printer is ready.

End of Chapter

Chapter 9
Programming the LAN and SCSI Interfaces

This chapter, when used with the appropriate data books, describes the local area network (LAN) interface and the Small Computer System Interface (SCSI) and how to program them.

I/O Fuse

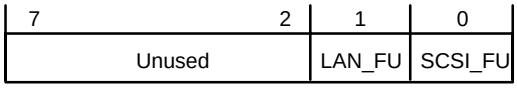
Both the SCSI and LAN are protected by an I/O fuse. The condition of the fuse can be checked by reading the IOFUSE register as follows.

IOFUSE **I/O Fuses**

FFFB 0040 **IOFUSE0** **Read Only**

FFFB 00C0 **IOFUSE1**

The I/O Fuse (IOFUSE) registers indicate whether or not the LAN and SCSI fuses are intact. IOFUSE0 represents the on−board SCSI and LAN, IOFUSE1 represents the expansion SCSI and LAN.



Bit	Mnemonic	Function
7−2	Unused	
1	LAN_FU	LAN Fuse 0 The fuse is blown. 1 The fuse is OK.
0	SCSI_FU	SCSI Fuse 0 The fuse is blown. 1 The fuse is OK.

LAN

When programming the LAN interface, use this chapter in conjunction with Advanced Micro Devices' *AM79C900 Integrated Local Area Communications Controller (ILACC™)* technical manual.

The LAN Interface contains the Ethernet LAN Controller and all asociated interface logic. The Ethernet controller is the AMD Integrated Local Area Communications Controller™ (ILACC™), AM 79C900. This device contains a built−in Serial Interface Adaptor that interfaces directly to the Ethernet channel. A LANID register, which is programmed by Data General, contains a 48 bit base Ethernet address for each Ethernet Controller in the system.

A 20 MHz crystal is wired between the XTAL1 and XTAL2 pins to provide a 10MHz network data rate.

All tri−state and open−drain outputs of the ILACC have been tied to their inactive states through pullups and pulldowns.

The ILACC resides on the LSIObus, which is multiplexed and uses the 680x0 mode. The Bus Access Control mode is configured by the BACON bits in CSR04 of the ILACC. In 680x0 mode, the ILACC uses big endian byte ordering; the LAN Interface swaps the data to little endian byte ordering for the LSIObus. The I/O interface swaps the data back to big endian byte ordering for the Mbus.

The LAN Interface

As shown in Figure 9−1, the LAN interface consists of data latches and an Ethernet controller. The LAN interface drives the Ethernet LAN using IEEE 802.3 based communications.

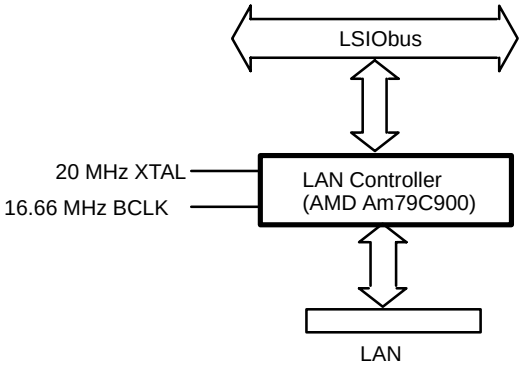


Figure 9–1 The LAN Interface

The LAN controller is an Advanced Micro Devices AM79C900 integrated local area communications controller (ILACC). It communicates with the CPU and memory via 32 data and address bits, and is either a bus master or slave. The LAN controller uses the CPU as the host processor and system memory as its data buffer.

Programming the LAN

This section defines how these registers must be configured for the LAN interface to operate properly.

LAN Registers

The LAN controller has 59 Control and Status Registers (CSRn) to regulate data transfers over the LAN.

Table 9–1 LAN Registers

Resource Name	Address
1st LAN board	
LAN FUSE SENSE	FFFB 00C0
DATA	FFFB 0100
RAP	FFFB 0104
LANID−BYTE0	FFFB 0110
LANID−BYTE1	FFFB 0114
LANID−BYTE2	FFFB 0118
LANID−BYTE3	FFFB 011C
LANID−BYTE4	FFFB 0120
LANID−BYTE5	FFFB 0124
LANID−BYTE6	FFFB 0128
LANID−BYTE7	FFFB 012C
2nd LAN board	
DATA	FFFB 0140
RAP	FFFB 0144
LANID−BYTE0	FFFB 0150
LANID−BYTE1	FFFB 0154
LANID−BYTE2	FFFB 0158
LANID−BYTE3	FFFB 015C
LANID−BYTE4	FFFB 0160
LANID−BYTE5	FFFB 0164
LANID−BYTE6	FFFB 0168
LANID−BYTE7	FFFB 016C

For a complete description of the DATA and RAP registers, see the *AM79C900 Integrated Local Area Communications Controller (ILACC)* technical manual.

For a complete description of the LANID registers, see the section “Ethernet LAN Address” that follows.

The Register Address Port and Data Port

The CSRn registers are programmed via the Register Address Port and the Data Port as described in the *AM79C900 Integrated Local Area Communications Controller (ILACC)* technical manual. The CSRn registers are programmed via a two−step process as follows:

1. Write the 6−bit address of the CSRn to the Register Address Port.
2. Read or write the data via the Data Port.

NOTE: The addresses of the Register Address Port and Data Port are defined in appendix A, “Address Map.”

Programming Exceptions

The following bits must be set as follows. Use this section in conjunction with the *AM79C900 Integrated Local Area Communications Controller (ILACC)* technical manual.

CSR3 register

Set:
BSWP (bit 2) = 1
ACON (bit 1) = 0

CSR4 register

Set:
DMAPLUS (bit 14) = 1
BACON (bits 7, 6) = 01

CSR15 register

Set:
PORTSEL (bit 8) = 0

Ethernet LAN Address

The following pre−programmed LANIDn registers define the address of the LAN controller on the Ethernet LAN.

LANIDn	LAN ID	
FFFB 0110	LANID0, byte 0	Read only
FFFB 0114	LANID0, byte 1	
FFFB 0118	LANID0, byte 2	
FFFB 011C	LANID0, byte 3	
FFFB 0120	LANID0, byte 4	
FFFB 0124	LANID0, byte 5	
FFFB 0128	LANID0, byte 6	
FFFB 012C	LANID0, byte 7	
FFFB 0150	LANID1, byte 0	
FFFB 0154	LANID1, byte 1	
FFFB 0158	LANID1, byte 2	
FFFB 015C	LANID1, byte 3	
FFFB 0160	LANID1, byte 4	
FFFB 0164	LANID1, byte 5	
FFFB 0168	LANID1, byte 6	
FFFB 016C	LANID1, byte 7	

The LANIDn registers provide the 48−bit Ethernet address plus two checksum bytes. The LANID registers are pre−programmed by Data General. Bytes 0 (08_{16}), 1 (00_{16}) and 2 ($1B_{16}$) are the company code; byte 3 (33_{16}) is the product code; bytes 4 and 5 are serialized for the specific controller board; and bytes 6 and 7 are the checksum. Byte 6 is the sum of bytes 0, 2 and 4; byte 7 is the sum of bytes 1, 3 and 5.

Bit	Mnemonic	Function
7−0	LANIDn	LAN ID The seven LANID registers together form the Ethernet LAN address plus two bytes of checksum data.

SCSI

The SCSI, which communicates with mass−storage devices and other peripherals, has an NCR 53C700 SCSI I/O Processor (SIOP). This interface adheres to the ANSI standard. The SCSI is illustrated in Figure 9−2.

The SIOP has a DMA controller which supports 16 and 32−bit data transfers as well as misaligned DMA transfers.

The SIOP has a SCSI SCRIPTS processor that allows SCSI and DMA instructions to be read from memory. In this way, the SIOP can perform complex SCSI bus sequences without CPU intervention. See The *NCR 53C700 Programmer's Guide* for information on the SCSI SCRIPTS processor and how to use it.

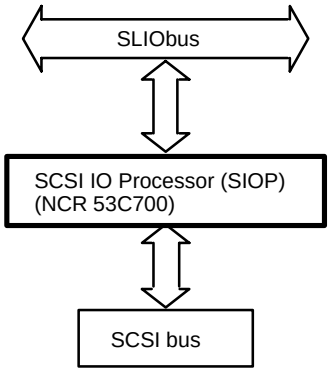


Figure 9–2 The SCSI

Programming the SCSI

To program the SCSI, use this chapter in conjunction with the following manuals from NCR:

- *NCR 53C700 Data Manual*

  This manual describes the SCSI and its registers.

- The *NCR 53C700 Programmer's Guide*

  This manual describes how to program and use the SCSI SCRIPTS machine language to control SCSI and DMA sequences.

SCSI Registers

Program the SIOP via its registers. For descriptions of the registers, see the *NCR 53C700 Data Manual*.

SCSI Byte Swapping

To correctly access SCSI registers, swap the registers as shown in Table 9−2 and Table 9−3. Table 9−2 illustrates how you must access the SCSI registers, and Table 9−3 illustrates the locations of the registers as defined by the NCR 53C700.

Table 9–2 DG SCSI Address Map

Offset	D[31–24]	D[23–16]	D15–8]	D[7–0]
00	SCNTL0	SCNTL1	SDID	SIEN
04	SCID	SXFER	SODL	SOCL
08	SFBR	SIDL	SBDL	SBCL
0C	DSTAT	SSTAT0	SSTAT1	SSTAT2
10	Reserved			
14	CTEST0	CTEST1	CTEST2	CTEST3
18	CTEST4	CTEST5	CTEST6	CTEST7
1C	TEMP_LO	TEMP_MID_LO	TEMP_MID_HI	TEMP_HI
20	DFIFO	ISTAT	Reserved	
24	DBC_LO	DBC_MID	DBC_HI	DCMD
28	DNAD_LO	DNAD_MID_LO	DNAD_MID_HI	DNAD_HI
2C	DSP_LO	DSP_MID_LO	DSP_MID_HI	DSP_HI
30	DSPS_LO	DSPS_MID_LO	DSPS_MID_HI	DSPS_HI
34	DMODE	Reserved		
38	Reserved	DIEN	DWT	DCNTL
3C	Reserved			

Table 9–3 NCR 53C700 Address Map

Offset	D[31–24]	D[23–16]	D15–8]	D[7–0]
00	SIEN	SDID	SCNTL1	SCNTL0
04	SOCL	SODL	SXFER	SCID
08	SBCL	SBDL	SIDL	SFBR
0C	SSTAT2	SSTAT1	SSTAT0	DSTAT
10	Reserved			
14	CTEST3	CTEST2	CTEST1	CTEST0
18	CTEST7	CTEST6	CTEST5	CTEST4
1C	TEMP_HI	TEMP_MID_HI	TEMP_MID_LO	TEMP_LO
20	Reserved		ISTAT	DFIFO
24	DCMD	DBC_HI	DBC_MID	DBC_LO
28	DNAD_HI	DNAD_MID_HI	DNAD_MID_LO	DNAD_LO
2C	DSP_HI	DSP_MID_HI	DSP_MID_LO	DSP_LO
30	DSPS_HI	DSPS_MID_HI	DSPS_MID_LO	DSPS_LO
34	Reserved			DMODE
38	DCNTL	DWT	DIEN	Reserved
3C	Reserved			

Initializing and Programming the SCSI

The following conventions must be met when programming the SIOP.

Always read orwrite the DFIFO register as an 8−bit register because it occupies the same word as the ISTAT register which prohibits access to other registers when accessed.

Clear all of the reserved bits to 0, and ignore them when read.

In addition, set the bits in the following registers as defined below.

SCNTL0 — SCSI Control Register 0

Set:
EPC (bit 3) = 1
EPG (bit 2) = 1

EPC (Enable Parity Checking), when set to 1, enables the SCSI to check the parity of data on the SCSI bus.

EPG (Enable Parity Generation), when set to 1, enables the SCSI to generate parity bits for data passing to the SCSI bus from the LSIObus.

CTEST7 — Chip Test Register 7

Set:
Reserved (bits 7, 6, 5) = 0
DC (bit 1) = 1
DIFF (bit 0) = 0

DC (DC/output signal), when set to 1, causes the SIOP interface to drive the DC signal low when fetching instructions from memory. DC is used as a test point.

DIFF (differential mode) selects either single−ended or differential SCSI transceivers. DIFF must be set to 0 for single−ended transceivers.

DMODE — DMA Mode Register

Set:
BL1 (bit 7) = 1
BL0 (bit 6) = 0
BW16 (bit 5) = 0
286 (bit 4) = 0
IO/M (bit 3) = 0
FAM (bit 2) = 0

BL0 and BL1 should be programmed with 1 for a burst length of 4 words.

BW16 (host bus width) selects the bus width for DMA transfers. BW16 must be set to 0 for 32−bit transfers.

286 (286 mode) determines if the 53C700 operates in 286 mode. 286 must be set to 0 to select 386 mode.

IO/M (IO mapped or memory mapped) determines if the 53C700 transfers data to an IO mapped address or to a memory mapped address. IO/M should be set to 0 for memory mapped transfers.

FAM (fixed address mode) determines if the address pointer increments after each data transfer. FAM must be set to 0 to increment after each transfer.

DWT DMA Watchdog Timer Register

Set:
The DWT value must fall within the following range: 15_{16} <= DWT <= $4F_{16}$.

The system has a 15.36 µs watchdog timer, built into the system board hardware, that should be used. Software can use DWT if the system timer is too large, but DWT must fall within the abovementioned range.

DCNTL DMA Control Register

Set:
CF1 (bit 7) = 0
CF0 (bit 6) = 1
S16 (bit 5) = 0
Reserved (bit 1) = 0

CF0 and CF1 (Clock Frequency bits) determine the clock period used by the SCSI portion of the SIOP. With this setting, the internal divide by is 1.5, and the SCSI host clock is 33.33 MHz.

S16 (SCRIPTS fetch data size) determines whether the SCRIPTS are fetched in 16 bits or 32 bits. This setting generates 32−bit fetches.

End of Chapter

Chapter 10
Programming the Real-Time Clock and the Programmable Interval Timer

This chapter describes the programmable interval timer (PIT) and the real−time clock (RTC), and how to program them.

The PIT and RTC

The programmable interval timer (PIT) and the real−time clock (RTC) are both part of the I/O gate array. Both are programmable as described in the following sections. The RTC is a 32−bit free−running counter which is the system's periodic time base. The RTC has two registers, the command/status register and the count register. Figure 10−1 illustrates the PIT and RTC, and the relationship between them.

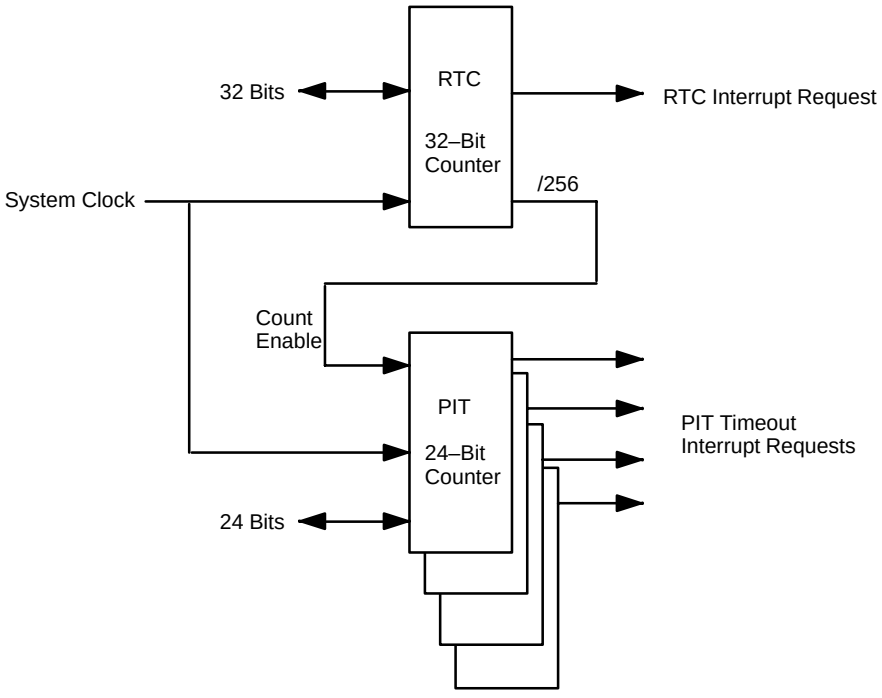


Figure 10–1 The PIT and the RTC

Programming the RTC

This section describes the real-time clock (RTC) and how to program it.

Table 10–1 RTC Registers

Register	Address
RTC_CNT	FFF8 F084
RTC_C/S	FFF8 F088

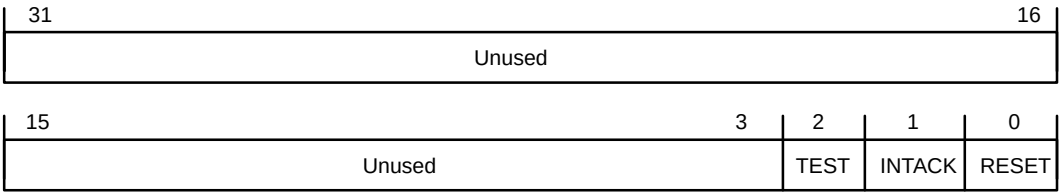
RTC Test Mode

The RTC test mode is used to test the counter. When the test mode is active, the counter increments when the RTC Count (RTC_CNT) register is read, not with the system clock. The 32-bit counter is divided into 4-bit sections while in test mode. Therefore to test the counter, read the RTC_CNT register 17 times, and each time check the eight counter nibbles. Each nibble should sequence through 0, 1, 2, ... D, E, F, then back to 0.

RTC_C/S RTC Command/Status

FFF8 F088 Read/Write

The Real-Time Clock Command/Status (RTC_C/S) register, in conjunction with the RTC_CNT register, controls the real-time clock.



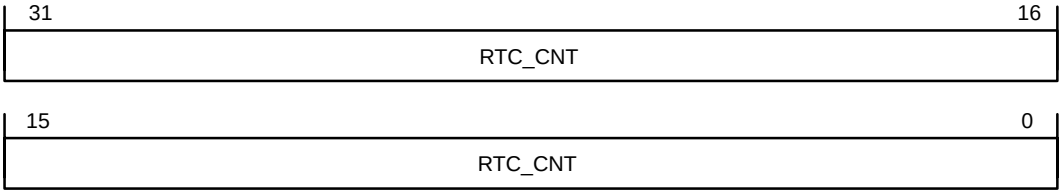
Bit	Mnemonic	Function
31–3	Unused	
2	TEST	Test TEST controls the test mode of the RTC counter. 1 Test mode. 0 Counter mode. TEST is cleared by resets.
1	INTACK	Interrupt Acknowledge Writing a 1 to INTACK clears the RTC counter rollover interrupt request. Writing a 0 has no action. INTACK is not affected by resets.
0	RESET	Reset Writing a 1 to RESET resets the RTC counter to 0. Writing a 0 has no action. NOTE: It takes 4 clock cycles for the counter to reset after the bus write cycle.

RTC_CNT — RTC Count

FFF8 F084 — Read/Write

The Real-Time Clock Count (RTC_CNT) register reflects or resets the RTC counter value. When used in conjunction with the RTC_C/S register, RTC_CNT controls the real-time clock. The RTC counter is a free-running clock that cannot be stopped; the value of RTC_CNT will change between accesses. Access RTC_CNT only as a word (32-bits).

Cleared by system reset.



Bit	Mnemonic	Function
31–0	RTC_CNT	RTC Count When read, RTC_CNT reflects the current value of the RTC. When written to, RTC_CNT sets the RTC to a new value.

Programming the PIT

The programmable interval timer (PIT) is a 24-bit counter used to measure time periods. There are four independent PITs (PIT0 − PIT3). Each is controlled through reads/writes to a dedicated command/status register (PIT_C/S0 − PIT_C/S3). The PIT values can be examined or modified via the PIT count (PIT0_CNT − PIT3_CNT) registers.

Each PIT is an up-counter that clocks at 1/256 of the system clock. The rollover point is set via the PIT_CNT register; the value can be anything from 0000 00xx to FFFF FFxx. The PIT generates a PIT overflow (PITnOF) interrupt when it reaches the counter value.

This section describes the PIT registers.

Table 10–2 PIT Registers

Register	Address
PIT0_CNT	FFF8 F004
PIT1_CNT	FFF8 F008
PIT2_CNT	FFF8 F010
PIT3_CNT	FFF8 F020
PIT0_C/S	FFF8 F044
PIT1_C/S	FFF8 F048
PIT2_C/S	FFF8 F050
PIT3_C/S	FFF8 F060
PIT_CMD_ALL	FFF8 F07C

PIT Test Mode

The PIT test mode is used to test the counters. When the test mode is active, the counter increments when the PIT Count (PIT_CNT) register is read, not with the system clock. The 24-bit counter is divided into 4-bit sections while in test mode. Therefore to test a counter, read the PIT_CNT register 17 times, and each time check the six counter nibbles. Each nibble should sequence through 0, 1, 2, ... D, E, F, then back to 0.

PITn_C/S PIT Command/Status

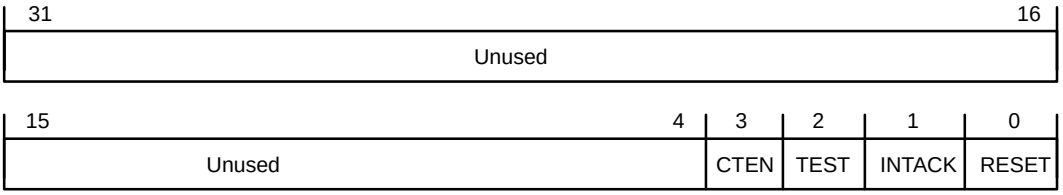
FFF8 F044 **PIT0_C/S** **Read/Write**

FFF8 F048 **PIT1_C/S**

FFF8 F050 **PIT2_C/S**

FFF8 F060 **PIT3_C/S**

The Programmable Interval Timer Command/Status (PIT0_C/S – PIT3_C/S) registers, in conjunction with the PITn_CNT registers, controls the PITs.

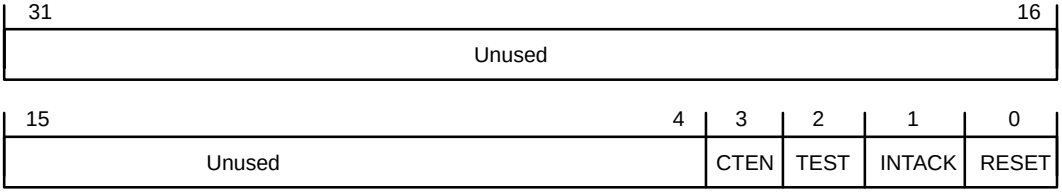


Bit	Mnemonic	Function
31–4	Unused	
3	CTEN	Count Enable 1 Enables the counter. 0 Disables the counter. CTEN is cleared by system resets.
2	TEST	Test TEST controls the test mode of the PIT counter. 1 Test mode. 0 Counter mode. TEST is cleared by system resets.
1	INTACK	Interrupt Acknowledge Writing a 1 to INTACK clears the PIT counter rollover interrupt request. Writing a 0 has no action. INTACK is not affected by resets.
0	RESET	Reset Writing a 1 to RESET resets the PIT counter to 0. Writing a 0 has no action. NOTE: It takes 4 clock cycles for the counter to reset after writing to PITn_C/S.

PIT_CMD_ALL — PIT Command All

FFF8 F07C — Write only

The Programmable Interval Timer Command All (PIT_CMD_ALL) register has the same bits and performs the same functions as the PITn_C/S registers, but when written, PIT_CMD_ALL updates all four PITs. Unlike the PITn_C/S registers, PIT_CMD_ALL is write only; to obtain the status of a PIT, read the corresponding PITn_C/S register.



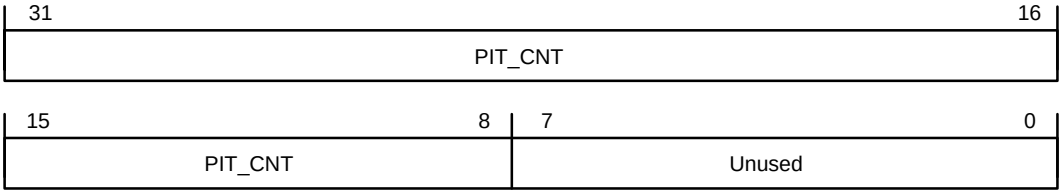
Bit	Mnemonic	Function
31–4	Unused	
3	CTEN	Count Enable 1 Enables the counter. 0 Disables the counter. CTEN is cleared by system resets.
2	TEST	Test TEST controls the test mode of the PIT counter. 1 Test mode. 0 Counter mode. TEST is cleared by system resets.
1	INTACK	Interrupt Acknowledge Writing a 1 to INTACK clears the PIT counter rollover interrupt request. Writing a 0 has no action. INTACK is not affected by resets.
0	RESET	Reset Writing a 1 to RESET resets the PIT counter to 0. Writing a 0 has no action. NOTE: It takes 4 clock cycles for the counter to reset after writing to PITn_C/S.

PITn_CNT — PIT Count

FFF8 F004	**PIT0_CNT**	**Read/Write**
FFF8 F008	**PIT1_CNT**	
FFF8 F010	**PIT2_CNT**	
FFF8 F020	**PIT3_CNT**	

The Programmable Interval Timer Count (PIT0_CNT – PIT3_CNT) registers, used in conjunction with the PITn_C/S registers, controls the PITs.

Cleared by system reset and not affected by local reset.



Bit	Mnemonic	Function
31–8	PIT_CNT	PIT Count Contains the PIT count value at which the counter will overflow and generate an interrupt. The PIT count is the 2's complement of the desired period; ex. a period of 3 x 256 clock cycles requires a PIT_CNT value of FFFF FDxx
7–0	Unused	

End of Chapter

Chapter 11
The System Control Monitor (SCM)

Overview

The System Control Monitor (SCM) is the interface to AViiON® RISC−based computer hardware. The SCM is a firmware monitor program that tests and manages the system at powerup and maintains control until the operating system takes over. The SCM resumes control whenever the operating system is halted.

Several SCM commands, described later in this chapter, were designed for system programming, diagnostic development and program debugging.

The SCM Prompt

The following command interpreter prompts indicate that. The SCM prompt appears when all processors (CPUs) are halted.

In single−processor systems, the prompt is:

```
SCM>
```

In multiprocessor systems, the prompt displays the number (n) of the attached job processor:

```
Jp#n/SCM>
```

The SCM prompt may be changed via the **Prompt** command. On multiprocessor systems, the attached processor can be changed via the **Attach** command. These commands are described later in this chapter.

To access the SCM during powerup, press <**Ctrl–C**> before the operating system boots. If the operating system is already running, bring it down before running the SCM.

The SCM prompt appears when:

- The boot fails or is interrupted
- The user halts the operating system
- A command break sequence halts active processors
- The hardware reset or power switch is pressed
- The operating system encounters an unsupported breakpoint or interrupt

Exceptions are described in Motorola's book *MC88100 RISC Microprocessor User's Manual*.

Halting the Operating System

If an operating system is running, first shut it down; then halt the system. The SCM prompt will display.

If running the DG/UX™ operating system, shut it down as follows:

```
# cd /
# shutdown
```

NOTE: The **shutdown** command can be modified to specify whether to bring down software immediately, or to provide a period of time for users to log out. See the DG/UX documentation.

Then, halt the system and display the SCM prompt, as follows:

```
# halt –q
SCM>
```

CAUTION: Halting the system while the operating system or other software is running may result in lost or corrupted data. In a multiprocessor environment, all active processors (those started with a ***.START*** *system call) must receive the* ***.HALT*** *system call.*

If you don't have a DG/UX™ operating system, see the documentation for your operating system.

Resetting the System

To reset the system, either press the reset switch or turn the power off and on.

CAUTION: Always try to shut down the operating system before resetting the computer. Resetting the hardware while the operating system or other software is running may result in lost or corrupted data.

System Configuration Menu

To display the View or Change System Configuration menu, enter the following command line at the SCM prompt:

```
SCM> F )
```

```
View or Change System Configuration

1    Change default system boot path
2    Change console parameters
3    Change modem port parameters
4    Change mouse baud rate
5    View system configuration
6    Change VME A24 configuration
7    Return to previous screen

Enter choice(s) ->
```

To select a sub−menu, type the menu number followed by a carriage return (enter).

To exit a menu, select the last item on the menu or press New Line at the prompt without selecting an item.

Changes made to the SCM become the new *default* parameter; however, the changes are not in effect until the system is powered up or reset. To reset the computer, either press the reset switch or issue the **Reset** command.

NOTE: To restore the original system configuration defaults, type Ctrl−I. Ctrl−I intializes the system console port to the following factory defaults: 9600 baud, 8 data bits, no parity, ANSI character set, enabled flow control. Also, Ctrl−I initializes the following system defaults: U.S. English keyboard language, Block transfer mode for SCSI tape drives, and 1200 baud for the mouse or modem port.

Environment Control Word

The Environment Control Word (ECW) controls program debugging and diagnostic program development. The ECW is part of the Change Testing Parameters menu, which displays the state of the ECW bits and allows you to toggle the states. These changes do not affect the operating system or application programs; the ECW defines the test environment *in the SCM only* unless read or changed using the **.TECW** (Toggle Environment Control Word) system call.

Table 11–1 Environment Control Word

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 0 Disables display. 1 Enables displaying an additional output field that contains the mnemonics of memory address contents.	1
7	Print subtest message 0 Disables printing message. 1 Enables printing subtest messages to the system console.	0
8	Report all 0 Print brief messages to the system console. 1 Print verbose messages to the system console.	1
9	Halt on error 1 Enables halting the program after an error and returning the SCM prompt.	0
10	Enable error logging 0 Disables error logging. 1 Enables recording all errors in system error log.	0
11, 12	Reserved	0

(continued)

Table 11–1 Environment Control Word (continued)

Bit	Function	State at Powerup
13	Page mode 0 Disables Page mode. 1 Enables displaying output on the system console one screen (page) at a time	0
15, 14	Reserved	0
16	Run with D_Cache 0 Disables running test. 1 Enables running test with data cache.	0
17	Run with I_Cache 0 Disables running test. 1 Enables running test with instruction cache.	0
18	Run with D_MMU 0 Disables running test. 1 Enables running test with data cache translations.	0
19	Run with I_MMU 0 Disables running test. 1 Enables running test with instruction cache translations.	0
20–31	Reserved	0

(concluded)

Modify the ECW as follows:

1. While in the View or Change System Configuration menu, type **6** and press New Line to select item 6, “Change testing parameters.”

   The system displays the following screen.

```
ECW Bit        Function                       State
-------------------------------------------------

0              Reserved                    – Disabled
1              Loop on error               + Enabled
2              Output to console           + Enabled
3              Percent failure             – Disabled
4              Print pass messages         + Enabled
5              Output to printer           – Disabled
6              Disassembler                + Enabled
7              Print subtest message       + Enabled
8              Report all                  – Disabled
9              Halt on error               – Disabled
10             Enable error logging        – Disabled
11             Continue on exception       – Disabled
12             Reserved                    – Disabled
13             Page mode                   – Disabled
14             Reserved                    – Disabled
15             Reserved                    – Disabled
16             Run with D_cache            – Disabled
17             Run with I_cache            – Disabled
18             Run with D_MMU              – Disabled
19             Run with I_MMU              – Disabled
20–31          Reserved                    – Disabled


Select bit(s) to toggle –>
```

2. To toggle the parameter states, enter the bit number(s) and then press New Line. Separate bits by either a space or a comma.

NOTE: Bit 10 can only be toggled via the **.TECW** system call. You cannot toggle it via the Change Testing Parameters menu.

To Exit, press New Line *without* entering a number at the prompt.

The next time you reset or restart the computer, the system will use the new ECW state.

System Calls

System calls manipulate CPU registers; the operating system can pass control to and from the SCM using these system calls.

System calls provide:

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

Execute system calls as follows:

1. Load register 9 (r9) and other argument registers with the offset value defined in Table 11−2 (the values are in hexadecimal unless indicated otherwise).

2. Set CR7, the Vector Base Register (VBR), to the VBR specified by the PROM during powerup (PROM VBR). If the value in CR7 is different from the PROM VBR, save the value in CR7 before writing the PROM VBR.

NOTE: During powerup, the PROM VBR is written to CR7.

3. Execute the Trap on bit clear instruction: **tb0 0,R0,496**

Specify bit 0 of R0. 496 is an offset to the VBR the PROM system call. Because bit 0 of register R0 is always 0, this instruction always executes the trap routine.

Table 11–2 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 11–2 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.
.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
.KBLAN	r9=106	r2=Language	Returns language code to r2. The codes and languages are 1 U.S. English 2 German 3 U.K. English 4 French 5 Swedish 6 Spanish 7 Swiss 8 Italian 9 Japanese
.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.

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

(continued)

Table 11–2 System Calls

System Call	Argument(s)	Data Returned	Function
.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.
.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.
.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 11−1 lists the ECW bit values, functions, and default states at powerup.

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

(concluded)

SCM Subroutines

In addition to system calls, the System Control Monitor (SCM) supports hardwired entry points to subroutines in Table 11−3 (accessible with a **jsr** instruction containing the appropriate entry point).

Table 11–3 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.

SCM Commands

The System Control Monitor (SCM) has commands to control and debug programs, boot media, and change system configuration parameters.

A command line consists of one valid SCM command and possibly one or more arguments.

If a command is issued incorrectly, the SCM displays a general message (ex. *Invalid command, Requires argument(s), Invalid argument(s))* and then returns the prompt.

Type a maximum of 80 characters per command line. You do not have to type the entire command name; the SCM accepts the first letter of a command.

SCM commands and arguments are *not* case−sensitive, with the exception of device specification arguments to the **BOOT** command, which must be lowercase.

The SCM supports several keyboard control characters. Table 11−4 describes control sequences used to edit command lines, interrupt and exit from SCM commands, and restore configuration parameters.

Address and Data Conventions

The address arguments used in SCM commands consist of two 16−bit, hexadecimal words. You may omit leading zeros. SCM commands support physical and logical address arguments according to the current mode of the memory management unit (MMU). When address translation is on, logical addresses map to physical addresses.

The assembler assumes data is decimal unless there is a dollar sign ($) preceding the data to indicate hexadecimal. The disassembler returns hexadecimal data. The SCM displays data output mnemonics and accepts data input mnemonics by default, since both the assembler and disassembler are enabled by default.

With address translation on, the SCM displays the contents of a memory location using the following format:

```
Memory logicaladdress – physicaladdress  /  data – mnemonic
```

For example, if the contents of memory location 10 is $5555FFFF_{16}$ the SCM displays the following:

```
Memory 00000010 – 00000010 / 5555FFFF – xor.u   r10 r21 $FFFF
```

The data mnemonic includes the opcode (xor.u in the example above), the registers pointed to by the first 16−bit word of the 32−bit address (r10 and r21), and the hexadecimal data in the second word of the address ($FFFF).

Table 11–4 SCM Line Editing and Keyboard Control Commands

Keyboard Entry	Function
↵[1]	Completes the current input line, begins execution of command input, and returns the SCM prompt.
<**Ctrl–A**>	Recalls and displays the last command string you entered at the SCM prompt.
<**Ctrl–C**> [2]	Interrupts execution of an SCM command and returns the SCM prompt. This is a polled interrupt; some procedures complete before they break. If you do not have an auto−repeat keyboard, execute the Ctrl−C sequence repeatedly until you see the SCM prompt.
<**Ctrl–I**> [3]	Restores default configuration parameters to the following factory settings. System console port: 9600 baud, 8 data bits, no parity, ANSI character set, enabled flow control, U.S. English keyboard language. Parallel printer: Centronics interface. SCSI tape drives: block transfer mode. Mouse or modem port: 1200 baud. Also restores video timing parameters (see Ctrl−V below). Enter <**Ctrl–I**>, wait until you hear one beep; then, enter **1** if you have a 70 Hz. monitor, or enter **2** if you have a 60 Hz. monitor.
<**Ctrl–P**> [3]	Displays the current state of the Environment Control Word (ECW).
<**Ctrl–Q**> [4]	Resumes SCM output display that was suspended with the Ctrl−S sequence.
<**Ctrl–S**> [4]	Suspends SCM output display until you resume it with the Ctrl−Q sequence.
<**Ctrl–U**>	Erases the current line of text, from the left of the cursor to the SCM prompt.
<**Ctrl–V**>[3]	Allows you restore the default video parameters for the 70 Hz. graphics monitor, or to change the defaults for a 60 Hz. monitor. Use this control sequence only if the window that appears at powerup on the graphics monitor is distorted or does not appear. After you type Ctrl−V, wait until you hear a beep; then, type **1** for 70 Hz. parameters, or type **2** to configure for a 60 Hz. monitor. The monitor is configured to the new frequency when you hear a second beep. If the graphics monitor is not clear after trying both frequencies, see the “Solving Power Up Problems” section in the startup manual.

1 The New Line and Enter keys have special functions when used with the **EXAMINE** command..

2 Only functions as an interrupt to SCM functions.

3 Execute this sequence only while in the SCM.

4 Only works when Flow Control protocol is enabled within the SCM.

Table 11–5 SCM Commands

Command	Description	Function
. (period)	Displays processor status	Debugging
Attach	Specifies attached processor	Program control, system operation
Boot	Starts system from bootstrap device	See installation book.
Continue	Restarts attached processor	Program control, debugging
Display	Shows register file contents	Debugging
Examine	Open or edit contents of registers and memory locations	Debugging
Format	Displays View or Change Configuration menu	See installation book.
Help	Lists valid SCM commands	System operation, debugging program control
Initialize	Writes data to a range of memory	Debugging
Locate	Searches memory for a data pattern	Debugging
Move	Duplicates a memory block in new location	Debugging
Onestep	Executes the next program instruction	Debugging, program control
Prompt	Changes text of SCM prompt	See installation book.
Reset	Initializes system to power−up state	System operation
Start	Begins processor at specified address	Program control, system operation
Trap	Views or inserts breakpoints	Debugging
Untrap	Removes breakpoints	Debugging
View	Displays a range of memory	Debugging
Write	Inserts data in one memory location	Debugging
Zloader	Starts s−record loader utility	See installation book.

The command descriptions follow this format:

COMMAND–NAME
Command description.

COMMAND *argument [argument]*

NOTE: The minimal mnemonic for each command is always the *first letter* of the command name.

Description

Describes the command.

Arguments

Defines the arguments.

None	The command accepts no arguments.
argument	The command requires the argument defined here.
[argument]	The argument is optional; do not include brackets.

Related Commands

Lists closely related SCM commands.

Related Messages

Lists messages that may appear after invoking the command.

Examples

Examples of the command.

. (period)

Display job processor status.

. (period)

Description

The **.** (period) command displays the status of the attached job processor and its program registers.

The status includes the following:

- PSR (Processor Status Register)
  The processor state of the program that was last running on the attached processor. The PSR is the value in Control Register 1 (cr1).
- XPC (Execute Program Counter)
  The contents of the program counter (PC) of the program that caused entry into the SCM. XPC is from Control Register 4 (cr4).
- DCSH (Data Cache Enable/Disable)
  The state of the user program data cache before entering the SCM. Y indicates enabled, and N indicates disabled.
- DMMU (Data Memory Management Unit Enable/Disable)
  The state of the user program data MMU before entering the SCM. Y indicates enabled, and N indicates disabled.
- ICSH (Instruction Cache Enable/Disable)
  The state of the user program instruction cache before entering the SCM. Y indicates enabled, and N indicates disabled.
- IMMU (Instruction Memory Management Unit Enable/Disable)
  The state of the user program instruction MMU before entering the SCM. Y indicates enabled, and N indicates disabled.

Arguments

None

Related Commands

Attach	Specify the attached processor in multiprocessor systems.
Continue	Resume program execution at the program counter value of the currently attached processor.
Onestep	Execute the next single instruction of a program and then display trace information.

Related Messages

None

Examples

- Display processor status on single processor system.

```
SCM> .↵

PSR      XPC      DCSH      DMMU     ICSH  IMMU
A00003F2 FFC039DE N         N          N     N
```

- Display processor status information on multiprocessor system.

```
Jp#0/SCM> .↵

Jp#0/PSR XPC      DCSH      DMMU     ICSH  IMMU
A00003F2 FFC039DE N         N          N     N
```

ATTACH
Specify active job processor.

ATTACH *[jp#]*

Description

Attach is only valid for multiprocessor systems. It allows you to attach the SCM to a specified job processor for subsequent operations (in other words, it makes a particlar processor active). Unattached processors remain in an idle state. The SCM prompt indicates which job processor is currently attached (Jp#n/SCM> where n is the number of the attached job processor). By default, Jp#0 is the attached processor after powerup.

Attach returns the currently attached processor if used without an argument.

Arguments

[jp#]	The number of the job processor to attach.

Related Commands

. (period)	Display the status of the attached processor.
Continue	Resume program execution at the program counter value of the currently attached processor.
Prompt	Change the SCM prompt text.
Start	Execute a program from a specified address.

Related Messages

Jp#n attached

Examples

- Attach SCM to job processor #1.

```
Jp#0/SCM> A 1 ↵
Jp#1 attached
```

- Display the currently attached job processor.

```
Jp#1/SCM> A ↵
Jp#1 attached
```

BOOT
Starts the system from a specified device.

BOOT *[dev([cntrl],[unit],[file#])][filepath]]*

Description

BOOT resets the system hardware; then it loads a bootstrap program from a device specified in the *dev* argument. The argument format conforms to the common object file format (coff) defined in section 4 of the Binary Compatibility Standard (BCS).

When you use **BOOT** *without* an argument, the SCM boots from a default boot path. If the default boot path is not initialized or not valid, the SCM tries to find a valid bootstrap file according to a hardcoded probe sequence.

Arguments

[dev([cntrl],[unit],[file#])]
: *dev* defines the boot device (always lowercase). The device driver is named by *dev* followed by open and close parentheses. The parentheses denote optional parameters.

[filepath]
: Specifies an executable filename or an Internet host address, used and defined by the bootstrap program in a second−stage boot sequence.

Related Commands

FORMAT
: Displays the View or Change System Configuration menu. You can display the Change Boot Parameters menu from this menu.

Related Messages

Booting from ...

Unable to load boot file ...

Examples

1. Boot the default system boot path.

```
SCM> b ↵
```

2. Boot from file #0 on the first QIC tape drive installed in a deskside CSS2 chassis.

```
SCM> b st(ncsc(),4,0) ↵
```

3. On a multiprocessor computer, boot from the third file on the tape in the second tape drive (SCSI ID#5).

```
Jp#0/SCM> b st(ncsc(),5,2) ↵
```

4. Boot your DG/UX operating system kernel to run level 3.

```
SCM> b sd(ncsc())root:/dgux –3 ↵
```

5. *Using a Danish keyboard*, boot the DG/UX system from the system disk.

```
SCM> b sd(ncsc)=,0=rootAE–dgux.file +1 ↵
```

6. *Using a Norwegian keyboard*, boot the DG/UX system from the system disk.

```
SCM> b sd(ncsc)=,0=rootO–dgux.file +1 ↵
```

7. Boot AViiON System Diagnostics (the program file **diags** located in the directory called **stand** on the logical disk **usr**) from the default system disk.

```
SCM> b sd(ncsc(),0)usr:/stand/diags↵
```

8. Boot any executable file called **bootfile** in the root directory on the second SCSI disk (SCSI ID#1).

```
SCM> b sd(ncsc(),1)root:/bootfile
```

9. Boot from the first host that responds on the Ethernet LAN (your computer is connected to only one network).

```
SCM> b dgen()
```

10. Boot from the host at Internet address 128.111.5.6 on the alternate Ethernet LAN (you are connected to two different LANs, one managed by an integrated and a one VME Ethernet controller).

```
SCM> b hken()128.111.5.6:
```

11. Boot from the first host than responds on the Token Ring LAN.

```
SCM> b vitr()
```

CONTINUE
Restart attached processor.

CONTINUE *[trace−count]*

Description

Continue resumes program execution at the address stored in the program counter (NPC) of the attached processor, for the number of instructions specified in the optional trace count argument. The NPC is CR5 (Control Register 5). When you use the command without the trace count argument, system control passes completely to the continued program.

Arguments

[trace−count]	The number of instructions to execute (in hexadecimal). The system displays the address, data, and mnemonic (in that order) after each instruction, then halts, displays the processor status, and returns the SCM prompt.

Related Commands

. (period)	Display the status of the attached processor, including the value of the NPC.
Attach	Specify the attached processor in multiprocessor systems.
Display	Show the contents of all register files.
Start	Execute a program from a specified address.

Related Messages

None

Examples

- Resume program execution at the NPC value; leave the SCM.

```
SCM> C ↵
```

- Resume program execution at the current PC value, display trace information after the next three instructions; then halt the processor and return to the SCM.

```
SCM> C 3 )
Trace       00000010       5555FFFF xor.u    r10 r21 $ffff
Trace       00000014       00000000 xmen.bu  r0 r0 $0
Trace       00000018       00000000 xmen.bu  r0 r0 $0

 PSR      XPC      NPC       FPC      DCSH  DMMU   ICSH  IMMU
 A0000000 0000001A 0000001E  00000022  Y     Y      N     N

SCM>
```

DISPLAY
Show contents of register files.

DISPLAY

Description

Display shows the contents of general register files (r), control register files (cr), and floating−point control registers (fcr) in the attached processor. Without an argument, only general register files are displayed.

Arguments

None

Related Commands

Attach Specify the attached processor in multiprocessor systems.

Examine Display and/or change specified registers or memory.

Related Messages

None

Examples

Display the current content of all register files.

```
SCM> D ↵

  r00 = 00000000  r01 = 00066B58  r02 = FFC00000   r03 = 00000000
  r04 = 0000000C  r05 = FFF00000  r06 = 0016E570   r07 = 0016B340
  r08 = 00003230  r09 = 00000063  r10 = 00000998   r11 = FFFFFFA1
  r12 = 00000000  r13 = 00000000  r14 = 00000000   r15 = 00000000
  r16 = 00000000  r17 = 00000000  r18 = 00000000   r19 = 00000000
  r20 = 00000000  r21 = 00000000  r22 = 00000000   r23 = 00000000
  r24 = 0016E570  r25 = 0000002C  r26 = 000E0000   r27 = 000F0000
  r28 = 00010000  r29 = 00000000  r30 = 017DEFD0   r31 = 017DEFC8
 cr00 = FFF8113C cr01 = 800003FB cr02 = 800003f0  cr03 = 00000000
 cr04 = 0006009A cr05 = 0006009E cr06 = 000600A2  cr07 = FFC00000
 cr08 = 0000403F cr09 = 0000403F cr10 = 017DEF40  cr11 = 00000000
 cr12 = 017DEF58 cr13 = 017DEF4C cr14 = 00000000  cr15 = 00000000
 cr16 = 00000000 cr17 = 00066B58 cr18 = FFC00000  cr19 = 00000000
 cr20 = 00000000fcr00 = 00000000fcr01 = 00000000 fcr02 = 00000000
fcr03 = 00000000fcr04 = 00000000fcr05 = 000048A2 fcr06 = 00000000
fcr07 = 00000000fcr08 = F8004808fcr62 = 00000001 fcr63 = 00000000
```

EXAMINE

Open and optionally change the contents of selected register or memory locations.

EXAMINE [*[M] address* | *H address* | *Q address* | *Rnumber* | *CRnumber* | *FCRnumber*]

Description

Examine opens and displays the contents of a specified memory address or register file. After opening a memory location, you can use special key functions described in Table 11–6 to enter new data, open the next or previous memory location, or return to the SCM prompt without making changes.

CAUTION: There are no restrictions to the areas of memory you can modify; modifying system control registers or NOVRAM locations could halt the system or destroy necessary system information. Ctrl–C will not recover data already modifed.

Use one argument per command line to specify whether you want to open a memory location or register file, and whether to view memory locations in word (32 bit), half–word (16 bit), or quarter–word (8–bit, or single byte) increments.

Without an argument, the **Examine** command opens and displays the last memory addressed examined. After the system halts or the first time after powerup, the SCM opens memory location 0 when you use the command without an argument.

Table 11–6 Special Key Functions for EXAMINE Command

Standard PC Keyboard	DASHER Keyboard	Function With EXAMINE Command
Escape	New Line or Break ESC	Writes data if entered; then closes the memory location or register and returns the SCM prompt.
Enter	Carriage Return	When a register file is open, closes the register and returns the SCM prompt. When a memory location is open, writes data if entered, closes the current memory location, and opens the next location.
Shift–6	Shift–6	When a register file is open, closes the register and returns the SCM prompt. When a memory location is open, writes data if entered, closes the current memory location, and opens the previous location.

Argument

[[M] address] The 32−bit (single word) memory location you want to examine or modify. Typing **M** is optional. If you type a number without a **M**, **H**, **Q**, **R**, **CR**, or **FCR** before it, the SCM interprets the number as a 32−bit memory address. Any address is masked to be a true 32−bit word (bits 0 and 1 are cleared).

[H address] The 16 bits (half−word) of memory you want to examine or modify. You must type **H**, followed by a space and the memory location. Any address is masked to be a true half−word (bit 0 is cleared).

[Q address] The 8 bits (one byte) of memory you want to examine or modify. You must type **Q**, followed by a space and the memory location. No address masking.

[Rnumber] The CPU register file you want to examine or modify. You must type **R**. Valid entries are R0 through R31.

[CRnumber] The control register you want to examine or modify. You must type **CR**. Valid entries are CR0 through CR20.

[FCRnumber] The floating point register you want to examine or modify. You must type **FCR**. Valid entries are FCR0 through FCR8, FCR62, or FCR63.

Related Commands

Display Show the contents of all register files.

Related Messages

None

Examples

NOTE: The examples in this section include keyboard characters for a standard IBM PC AT−compatible keyboard.

- Display the contents of memory address 1000 without modifying. Disassembler and MMU are disabled in this example.

```
SCM> E M 1000 ↵     or    SCM> E 1000 ↵

Memory 00001000 / 58670004 <Esc>

SCM>
```

- Display the contents of memory address 1000 without modifying. Disassembler and MMU are enabled in this example.

```
SCM> E 1000 ↵
Memory 00001000 – 00001000/12345678 – ld.d r17 r20 $5678
Esc
SCM>
```

- Display without modifying the contents of last memory address examined.

```
SCM> E ↵
Memory 00001000 – 00001000/12345678 – ld.d r17 r20 $5678
Esc
SCM>
```

- Display without modifying the contents of register file r03.

```
SCM> E R3 ↵   or   SCM> E R03 ↵
R03 = 00000000
SCM>
```

- Display without modifying the contents of control register file cr01.

```
SCM> E CR01 ↵    or   SCM> E CR1 ↵
r03 = 80000F3B  ↵
SCM>
```

- Enter data 12345678 in memory address 1000; then exit from the command and return to the SCM.

```
SCM> E M1000 ↵   or    SCM>E 1000 ↵
Memory 00001000 / 58670004  12345678 <Esc>
SCM>
```

- Display the contents of memory address 1000; deposit data 12345678 in memory address 1004; then return to the SCM.

```
Jp#0/SCM> E 1000 ↵
Memory 00001000 / 12345678 <CR>
Memory 00001004 / 58670004   12345678 <Esc>
SCM>
```

- Display the contents of floating−point control register file fcr62; then deposit data 5555FFFF into fcr62.

```
SCM> E FCR62 ↵
fcr62 = 00000000   5555FFFF ↵
SCM>
```

- Display the first 16 bits at location 1000.

```
SCM> E H 1000 ↵
Memory 00001000 / 1234 <Esc>
SCM>
```

- Display the first 8 bits at location 1000.

```
SCM> E Q 1000 ↵
Memory 00001000 / 12 <Esc>
SCM>
```

FORMAT
Displays Configuration Menus.

FORMAT

Description

Format displays the “View or Change System Configuration” menu.

To return to the SCM prompt, press New Line or select the last item in the “View or Change System Configuration” menu.

Arguments

None

Related Commands

Boot	Boot a device or display the Change Boot Parameters menu.

Related Messages

None

Examples

None

HELP
Display available SCM commands.

HELP

Description

Displays an alphabetical list of the minimal mnemonic for valid SCM commands, the arguments each command accepts, and a brief command description.

Arguments

None

Related Messages

None

Examples

None

INITIALIZE
Write specified data to a range of memory.

INITIALIZE *data beg−addr end−addr*

Description

The **Initialize** command writes specified data to a range of memory addresses that starts at the specified 32−bit beginning address and ends at the 32−bit ending address.

CAUTION: There are no restrictions to areas of memory you can initialize. Initializing system control registers or NOVRAM locations could halt the system or destroy necessary data. Use the Ctrl−C sequence to exit from the command during processing.

Arguments

data	Content written to each location of the memory range you are initializing.
beg−addr	First location in the range of memory.
end−addr	Last location in the range of memory.

Related Commands

Move	Duplicate the contents of a specified source range and write it to a specified destination range.
View	Display the contents of a range of memory.
Write	Write data to a single memory location.

Examples

Initialize memory by writing 5555FFFF to a range of memory beginning at address 0 and ending at 10. Then view the memory range to see the results.

```
Jp#1/SCM> I 5555FFFF 0 10 ↵

Jp#1/SCM> VIEW 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000004  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000008  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 0000000C  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000010  /  5555FFFF – xor.u         r10  r21  $FFFF
```

LOCATE
Find a specified data pattern in memory.

LOCATE *data [beg−addr] [end−addr]*

Description

Locate searches memory for a specified data pattern; then displays the address and contents of each location in which it finds the data. You can omit the address arguments and search all of physical memory, specify a starting address only, or specify a range of memory with starting and ending address arguments.

NOTE: A search through all of memory could take several hours! To stop a search before it completes, use Ctrl−C.

Arguments

data	Pattern in memory for which the system searches.
[beg−addr]	Location in memory where system begins searching. If you specify only one address argument, the system interprets it as a starting address.
[end−addr]	Location in memory where system stops searching.

Related Commands

Move	Duplicate the contents of a specified source range and write it to a specified destination range.
View	Display the contents of a range of memory.

Related Messages

None

Examples

- Locate each occurrence of data pattern 01234567 in a range of memory beginning at 0 and ending at 1000; then display the address of each occurrence.

```
Jp#1/SCM> L 01234567 0 1000 ↵
Memory 00000800  /  01234567 – ld.d r17  r20  $4567
```

- Locate each occurrence of data pattern 01234567 in all of main memory.

```
Jp#1/SCM> L 01234567 ↵

Memory 00000800  /  01234567 – ld.d r17  r20  $4567
Memory 00001200  /  01234567 – ld.d r17  r20  $4567
```

MOVE
Duplicate a block of memory.

MOVE *count source−addr dest−addr*

Description

The **Move** command copies the block of data that begins at the source address and ends after the specified count of 32−bit words; then moves the copy into a block of the same size starting at the destination address.

CAUTION: There are no restrictions to areas of memory into which you can move data. Overwriting data in system control registers or NOVRAM locations could halt the system or destroy necessary data. Use the Ctrl−C sequence to exit from the command during processing.

Arguments

count	Hexadecimal number of 32−bit words from the beginning address to the end of the block to be moved. There is no maximum or minimum block size.
source−addr	Memory location at the beginning of the range to be moved.
dest−addr	Memory location at the beginning of the range the block is moving to.

Related Commands

Initialize	Write data to a range of memory.
Locate	Search memory for a data pattern.
View	Display the contents of a range of memory.

Related Messages

None

Examples

- Move the data block beginning at memory address 0 and spanning 4F words to destination address 8000.

```
Jp#1/SCM> M 4F 0 8000 ↵
```

- Move the block that begins at 0 and spans 5 words to destination address 400. First, view the range of memory from locations 0 through 10.

```
Jp#1/SCM> VIEW 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000004  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000008  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 0000000C  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000010  /  5555FFFF – xor.u       r10 r21 $FFFF
```

  Next, move the block.

```
Jp#1/SCM> M 5 0 400  ↵
```

  Finally, view the contents of the block of memory from locations 400 through 410.

```
Jp#1/SCM> V 400 410  ↵

Memory 00000400  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000404  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000408  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 0000040C  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000410  /  5555FFFF – xor.u       r10 r21 $FFFF
```

ONESTEP
Execute next step of a program.

ONESTEP *[trace−count]*

Description

The **Onestep** command begins program execution at the address stored in the program counter (CR5) in increments of one instruction. It displays trace information after each instruction executes. With no argument, the system executes the single program instruction stored in CR5; then halts and displays status information.

Arguments

[trace−count]	Hexadecimal number of instructions to be executed one by one (in single steps) before the processor halts.

Related Commands

. (period)	Display the status of the attached processor.
Continue	Resume program execution at the program counter value of the currently attached processor.
Start	Execute a program from a specified address.

Related Messages

None

Examples

- Execute the instruction pointed to by the program counter.

```
SCM> 0 ↵

PSR      XPC      DCSH      DMMU     ICSH   IMMU
A0000000 FF001602 Y         N          N      N
```

- Execute three instructions one by one, beginning with the instruction pointed to by the PC.

```
SCM> 0 3 ↵

Trace       00000010       5555FFFF xor.u    r10 r21 $FFFF
Trace       00000014       00000000 xmen.bu  r0 r0 $0
Trace       00000018       00000000 xmen.bu  r0 r0 $0

PSR      XPC      DCSH      DMMU     ICSH  IMMU
A0000000 0000001A N         N          N     N
```

PROMPT
Changes SCM prompt text prefix.

PROMPT *[new−prompt]*

Description

Prompt changes the default SCM prompt to a specified ASCII string. This can be useful to uniquely identify multiple systems. The right bracket symbol (>) appears after the text prefix; if you change the prompt text to a null text string, your prompt is the right bracket symbol.

NOTE: Multiprocessor systems add the text string Jp#n/ (n = the number of the attached job processor) to the default prompt text.

Arguments

[new−prompt] Text string of ASCII characters to replace the prompt. The ASCII string can have as many as 1510 characters. There are no character or symbol restrictions.

Related Commands

Attach Specify the attached job processor in a multiprocessor system.

Related Messages

Argument(s) out of range

Examples

1. Display the current SCM prompt; then change it to **AV5000**.

```
Jp#0/SCM> P ↵

Jp#0/SCM>

Jp#0/SCM> P AV5000 ↵

Jp#0/AV5000>
```

RESET
Restore system to power–up state.

RESET

Description

Initializes system elements (excluding memory) to their original power−up state. Unlike a *cold reset* (power applied to the system), a *warm reset* (initiated by software, the **Reset** command, or a Reset switch) does not initialize memory or run power−up diagnostics.

CAUTION: Be careful not to enter **R** *at the SCM prompt accidentally. You cannot use Ctrl−C or an SCM command to recover.*

Arguments

None

Related Commands

Boot Boot a device.

Related Messages

System Reset

Examples

- Reset the system (processors, keyboard interface, graphics interface, etc.).

```
SCM> R ↵

PSR      XPC      DCSH      DMMU     ICSH   IMMU
A00003F2 FFC039DE N         N          N      N
```

START
Start job processor at specified address.

START *address [trace−count]*

Description

The **Start** command begins executing a program at the main memory address specified. The operating system or user program resumes system control unless you use the *trace−count* argument.

Arguments

address	Memory location at which the processor starts executing.
[trace−count]	The system displays the address, data, and mnemonic (in that order) after executing the hexadecimal number of instructions you specify with this argument. Then the system halts and the monitor displays status information.

Related Commands

Boot	Boot a device.
Continue	Resume program execution at the program counter value of the currently attached processor.
Onestep	Execute the next single instruction of a program and then display trace information.

Related Messages

None

Examples

- Start processor executing at address 398F0.

```
SCM> S 398F0 ↵
```

- Start processor executing at address 398F0 with a trace count of 3.

```
SCM> S 398F0 3 ↵

Trace       000398F0       5555FFFF xor.u    r10 r21 $FFFF
Trace       000398F4       00000000 xmen.bu  r0 r0 $0
Trace       000398F8       00000000 xmen.bu  r0 r0 $0

PSR      XPC      DCSH       DMMU    ICSH  IMMU
A0000000 000398FA Y         Y         N     N
```

TRAP
View or insert breakpoints.

TRAP *[address] ...*

Description

Insert a breakpoint at the specified address. You can insert up to 20_{10} breakpoints. The command does not allow duplicate breakpoints.

With no argument, **Trap** displays a list of all current breakpoints.

NOTE: The SCM implements breakpoints with the trap exception; hence, the command names **Trap** and **Untrap**.

Arguments

[address]	Memory location at which you want to insert a breakpoint.
...	The memory location of each subsequent breakpoint (when you insert more than one breakpoint per command line).

Related Commands

Untrap	Delete breakpoint(s)

Related Messages

None

Examples

- Insert a breakpoint at addresses 5000 and 77000.

```
SCM> T 5000 77000 ↵

Breakpoint # 1 – 00005000  Set
Breakpoint # 2 – 00077000  Set
```

- Display the current list of breakpoints and their addresses.

```
SCM> T ↵

Breakpoint # 1 – 00005000
Breakpoint # 2 – 00077000
```

No response indicates there are no current breakpoints.

UNTRAP
Delete breakpoints.

UNTRAP *[breakpoint] ...*

Description

The **Untrap** command removes the breakpoint number(s) specified, or removes all current breakpoints with no argument. To view current breakpoint numbers, use the **Trap** command.

Arguments

breakpoint	Number of the breakpoint you want to remove. (A decimal number 1 through 20.)
...	Numbers of each additional breakpoint number you want to remove.

Related Commands

Trap	Insert a breakpoint at specified address, or display the current breakpoints.

Related Messages

Invalid breakpoint

Examples

- Find the three current breakpoints; then remove two of them.

```
SCM> T ↵

Breakpoint # 1 – 00005000
Breakpoint # 2 – 00077000
Breakpoint # 3 – 000C6001

SCM> U 1 3 ↵

Breakpoint # 1 deleted
Breakpoint # 3 deleted
```

- Remove all current breakpoints.

```
SCM> U ↵
```

VIEW
Display a range of memory.

VIEW *beg−addr [end−addr]*

Description

The **View** command displays the contents of a specified block of contiguous memory. If you use the command with only a beginning address, the SCM displays all of memory from the specified address to the top of memory address.

NOTE: There are no restrictions to areas of memory you can specify. Viewing large blocks of memory could take hours, or cause memory exceptions. Use the Ctrl−C sequence to exit before completing the operation.

From left to right, the system displays the logical memory address, the physical memory address (only when MMU is enabled), the contents of the address, and the instruction mnemonic.

Arguments

beg−addr	First memory address in the range you want to view.
[end−addr]	Last memory address in the range you want to view.

Related commands

Initialize	Write data to a range of memory.
Locate	Search memory for a data pattern.
Move	Duplicate the contents of a specified source range and write it to a specified destination range.
Write	Write data to a single memory location.

Examples

- View the contents of a block of contiguous memory beginning at location 0 and ending at location 00000010.

```
SCM> V 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u     r10 r21 $FFFF
Memory 00000004  /  5555FFFF – xor.u     r10 r21 $FFFF
Memory 00000008  /  5555fFFF – xor.u     r10 r21 $FFFF
Memory 0000000C  /  5555FFFF – Xor.u     r10 r21 $FFFF
Memory 00000010  /  5555FFFF – xor.u     r10 r21 $FFFF
```

WRITE
Insert data in one memory location.

WRITE *[H][Q] address data*

Description

Write writes data to a specified address.

CAUTION: Write executes immediately; you cannot exit using Ctrl−C. There are no write restrictions; although writing data to system control registers or NOVRAM could halt the system or destroy necessary data.

Arguments

[H]	Address and data are 16−bits.
[Q]	Address and data are 8−bits.
address	Address to write data.
data	Hexadecimal data or assembler command.

Related Commands

Examine	Display and/or change specified registers or memory.

Related Messages

None

Examples

- Write 32 bits (one word) of data (01234567) to address 800.

```
SCM> W 800 01234567 ↵
```

- Write 16 bits (one half−word) of data (1234) to address 800.

```
SCM> W H 800 1234 ↵
```

- Write 8 bits (one byte) of data (12) to address 800.

```
SCM> W Q 800 12 ↵          or          SCM> W B 800 12 ↵
```

ZLOADER
Start the s–record load utility.

ZLOADER

Description

Zloader executes a program that downloads files in Motorola s−record format (S3/S7 type) to memory from a device connected to the system console port. The SCM recognizes s−records only after you use the **Zloader** command.

An s−record is a file that contains an unlimited number of records; each record has a maximum of 256 bytes. Servers use s−records to download files serially. The loader utility copies s−records from the serial port to system memory. The SCM reads information appended to individual s−records, stores each record at locations specified in the s−record header, and then verifies checksums. Information appended to the last s−record notifies the SCM when the entire file has been sent.

For this command to function, you must configure the system to use an s−record utility and a server must send s−record files. If you are not familiar with the s−record loader utility, if the system is not configured to receive s−records, or if the server is not sending, press the Ctrl−C sequence to exit from the **Zloader** command.

NOTE: The system will pause until it receives the last s−record in a file, or until you execute a Ctrl−C command sequence to exit to the SCM prompt.

Arguments

None

Related Commands

None

Related Messages

`DLL Started` DLL (downline loader) is an s−record load utility

`DLL Done`

Examples

None

End of Chapter

Appendix A
Address Map

The following table is a map of addressable locations within AViiON 4600 series systems and AViiON 530 series stations.

NOTE: The error logic returns a Bus Error if an invalid location is accessed. An invalid location is one that contains no resource.

Table A–1 Address Map

Resource Name	Address
User Space (3.996 Gbytes)	0000 0000 – FFBF FFFF
System Memory (RAM):	
Maximum 128 Mbytes	0000 0000 – 07FF FFFF
VME A32 (1024 Mbytes)	4000 0000 – 7FFF FFFF
Video Memory (frame buffer (256 Mbytes))	8000 0000 – 8FFF FFFF
VME A32 (1760 Mbytes)	9000 0000 – FDFF FFFF
VME A24 (16 Mbytes)	FE00 0000 – FEFF FFFF
Utility Space (4 Mbytes)	FFC0 0000 – FFFF FFFF
EPROM (256 Kbytes)	FFC0 0000 – FFC7 FFFF
CMMU Registers	FFF0 0000 – FFF7 FFFF

(continued)

Table A–1 Address Map

Resource Name	Address
CMMU	
System Control Registers	
CMMU ID (IDR)	FFF0 X000
System Command (SCR)	FFF0 X004
System Status (SSR)	FFF0 X008
System Address (SAOR)	FFF0 X00C
System Control (SCTR)	FFF0 X104
Local Registers	
Local Status (PFSR)	FFF0 X108
Local Address (PFAR)	FFF0 X10C
Area Pointers	
Supervisor Area (SAPR)	FFF0 X200
User Area (VAPR)	FFF0 X204
BATC Write Pointers	
Port 0 (BWP0)	FFF0 X400
Port 1 (BWP1)	FFF0 X404
Port 2 (BWP2)	FFF0 X408
Port 3 (BWP3)	FFF0 X40C
Port 4 (BWP4)	FFF0 X410
Port 5 (BWP5)	FFF0 X414
Port 6 (BWP6)	FFF0 X418
Port 7 (BWP7)	FFF0 X41C
Cache Diagnostic Port	
Data Port 0 (CDP0)	FFF0 X800
Data Port 1 (CDP1)	FFF0 X804
Data Port 2 (CDP2)	FFF0 X808
Data Port 3 (CDP3)	FFF0 X80C
Tag Port 0 (CTP0)	FFF0 X840
Tag Port 1 (CTP1)	FFF0 X844
Tag Port 2 (CTP2)	FFF0 X848
Tag Port 3 (CTP3)	FFF0 X84C
Set Status (CSSP)	FFF0 X880

Where X=

With 2 CMMUs per CPU:	With 6 CMMUs per CPU:
0 CPU0, Data CMMU	0 CPU0, Data CMMU0
1 CPU0, Instruction CMMU	1 CPU0, Data CMMU1
2 CPU1, Data CMMU	4 CPU0, Instruction CMMU0
3 CPU1, Instruction CMMU	5 CPU0, Instruction CMMU1
	6 CPU0, Instruction CMMU2
	7 CPU0, Instruction CMMU3
	8 CPU1, Data CMMU0
	9 CPU1, Data CMMU1
	C CPU1, Instruction CMMU0
	D CPU1, Instruction CMMU1
	E CPU1, Instruction CMMU2
	F CPU1, Instruction CMMU3

NOTE: During powerup, add 0007 0000 to the CMMU register addresses.

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

(continued)

Table A–1 Address Map

Resource Name	Address
Host Adapter (A32 space)	FFF0 C000
BBSRAM/Real−Time Clock Registers	FFF8 0000 − FFF8 1FFF
Control	FFF8 1FE0
Seconds	FFF8 1FE4
Minutes	FFF8 1FE8
Hour	FFF8 1FEC
Day	FFF8 1FF0
Date	FFF8 1FF4
Month	FFF8 1FF8
Year	FFF8 1FFC
DUART1	FFF8 2000 − FFF8 203F
Read Registers / Write Registers	
MR1A, MR2A / MR1A, MR2A	FFF8 2000
SRA / CSRA	FFF8 2004
Reserved / CRA	FFF8 2008
RHRA / THRA	FFF8 200C
IPCR / ACR	FFF8 2010
ISR / IMR	FFF8 2014
CTU / CTUR	FFF8 2018
CTL / CTLR	FFF8 201C
MR1B, MR2B / MR1B, MR2B	FFF8 2020
SRB / CSRB	FFF8 2024
Reserved / CRB	FFF8 2028
RHRB / THRB	FFF8 202C
Reserved / Reserved	FFF8 2030
Input port / OPCR	FFF8 2034
Start Counter / Set Output Bits	FFF8 2038
Stop Counter / Reset Output Bits	FFF8 203C
DUART2	FFF8 2040 − FFF8 207F
Read Registers / Write Registers	
MR1A, MR2A / MR1A, MR2A	FFF8 2040
SRA / CSRA	FFF8 2044
Reserved / CRA	FFF8 2048
RHRA / THRA	FFF8 204C
IPCR / ACR	FFF8 2050
ISR / IMR	FFF8 2054
CTU / CTUR	FFF8 2058
CTL / CTLR	FFF8 205C
MR1B, MR2B / MR1B, MR2B	FFF8 2060
SRB / CSRB	FFF8 2064
Reserved / CRB	FFF8 2068
RHRB / THRB	FFF8 206C
Reserved / Reserved	FFF8 2070
Input port / OPCR	FFF8 2074
Start Counter / Set Output Bits	FFF8 2078
Stop Counter / Reset Output Bits	FFF8 207C

(continued)

Table A–1 Address Map

Resource Name	Address
Keyboard	FFF8 2800 – FFF8 2820
RxHR and TxHR	FFF8 2800
SR	FFF8 2804
MR1 and MR2	FFF8 2808
CR	FFF8 280C
DSC	FFF8 2810
ETxC	FFF8 2820
SRST	FFF8 3100
SPKEN	FFF8 3104
WDTO	FFF8 3108
Reserved	FFF8 310C – FFF8 3FFF
Interrupt Registers	
IEN0	FFF8 4004
IEN1	FFF8 4008
IEN2	FFF8 4010
IEN3	FFF8 4020
IEN	FFF8 403C
IST	FFF8 4040
SETSWI	FFF8 4080
CLRSWI	FFF8 4084
ISS (ISTATE)	FFF8 4088
CLRINT	FFF8 408C
VIRL	FFF8 5000
VIAV1	FFF8 5004
VIAV2	FFF8 5008
VIAV3	FFF8 500C
VIAV4	FFF8 5010
VIAV5	FFF8 5014
VIAV6	FFF8 5018
VIAV7	FFF8 501C
VIV	FFF8 5020
GCS Registers	FFF8 6000 – FFF8 600F
GLOBAL0	FFF8 6001
GLOBAL1	FFF8 6003
BRDID	FFF8 6005
GPCS0	FFF8 6007
GPCS1	FFF8 6009
GPCS2	FFF8 600A
GPCS3	FFF8 600C
GPCS4	FFF8 600E
UCS	FFF8 7000
BASAD	FFF8 7004
GLBRES	FFF8 700C
CCS	FFF8 8000
ERROR	FFF8 8004
Unused	FFF8 8008
Unused	FFF8 800C
EXTAD	FFF8 8010
EXTAM	FFF8 8014
WHOAMI	FFF8 8018
WVAD	FFF8 8028
RVAD	FFF8 802C

(continued)

Table A–1 Address Map

Resource Name	Address
Color Graphics Subsystem Registers — Broadcast	.FFF8 9000 – FFF8 90FF
	FFF8 9000
STOP	FFF8 9004
CSR1	FFF8 9008
CMD	FFF8 900C
MASK	FFF8 9010
BACK	FFF8 9014
LPAT	FFF8 9018
PC/WID	FFF8 901C
CRT0	FFF8 9020
CRT1	FFF8 9024
CRT2	FFF8 9028
Reserved	FFF8 902C
STATE0	FFF8 9030
STATE1	FFF8 9034
Reserved	FFF8 9038 – FFF8 903C
PARM0	FFF8 9040
PARM1	FFF8 9044
PARM2	FFF8 9048
PARM3	FFF8 904C
PARM4	FFF8 9050
PARM5	FFF8 9054
PARM6	FFF8 9058
PARM7	FFF8 905C
PARM8	FFF8 9060
PARM9	FFF8 9064
PARM10	FFF8 9068
PARM11	FFF8 906C
PARM12	FFF8 9070
PARM13	FFF8 9074
PARM14	FFF8 9078
PARM15	FFF8 907C
Reserved	FFF8 9080 – FFF8 909C
DATA	FFF8 90A0
PLT0	FFF8 90A4
PLT1	FFF8 90A8
BLNK	FFF8 90AC
Reserved	FFF8 90B0 – FFF8 90BC

(continued)

Table A–1 Address Map

Resource Name	Address
Bt458 RAMDAC Registers: (8-Bit Color)	
Address Register (DAC0)	FFF8 90C0
Overlay Color 0	00
Overlay Color 1	01
Overlay Color 2	02
Overlay Color 3	03
Read Mask Register	04
Blink Mask Register	05
Command Register	06
Control/Test Register	07
Color Palette RAM (DAC1)	FFF8 90C4
Control Register (DAC2)	FFF8 90C8
Overlay Palette RAM (DAC3)	FFF8 90CC
Bt461 RAMDAC Registers: (24-Bit Color)	
Address Register Low	FFF8 90C0
Address Register High	FFF8 90C4
Alternate Color Palette	0000 – 00FF
Overlay Color 0	0100
Overlay Color 31	011F
ID Register0	200
Command Register 0	0201
Command Register 1	0202
Command Register 2	0203
Pixel Read Mask Register Low	0204
Pixel Read Mask Register High	0205
Pixel Blink Mask Register Low	0206
Pixel Blink Mask Register High	0207
Overlay Read Mask Register	0208
Overlay Blink Mask Register	0209
Test Register	020C
Reserved	FFF8 90D0 – FFF8 90DC
Z–Buffer Registers	FFF8 90E0 – FFF8 91F0
CMD	FFF8 90E0
IR	FFF8 90E4
Z_CNST	FFF8 90E8
Z_INIT	FFF8 90EC
Z_XINC	FFF8 90F0
Z_YINC	FFF8 90F4
HCLP	FFF8 90F8
YCLP	FFF8 90FC
CNFG	FFF8 90E0
ST0	FFF8 90E4
ST1	FFF8 90E8
ST2	FFF8 90EC
ST3	FFF8 90F0
Reserved	FFF8 90F4 – FFF8 90FF
Color Graphics Registers — POSN 00, 01, 10, 11. (Use offsets from Broadcast addresses.)	FFF8 9100 – FFF8 94FF

(continued)

Table A–1 Address Map

Resource Name	Address
Synchronous Serial Interface	FFF8 D000 – FFF8 D0FF
Port A	
CMR1A	FFF8 D000
CMR2A	FFF8 D004
S1RA	FFF8 D008
S2RA	FFF8 D00C
TPRA	FFF8 D010
TTRA	FFF8 D014
RPRA	FFF8 D018
RTRA	FFF8 D01C
CTPRHA	FFF8 D020
CTPRLA	FFF8 D024
CTCRA	FFF8 D028
OMRA	FFF8 D02C
CTHA	FFF8 D030
CTLA	FFF8 D034
PCRA	FFF8 D038
CCRA	FFF8 D03C
TxFIFOA	FFF8 D040
RxFIFOA	FFF8 D050
RSRA	FFF8 D060
TRSRA	FFF8 D064
ICTSRA	FFF8 D068
IERA	FFF8 D070
General	
GSR	FFF8 D06C
IVR	FFF8 D078
IVRM	FFF8 D0F8
ICR	FFF8 D07C
DTRTA	FFF8 D074
Port B	
CMR1B	FFF8 D080
CMR2B	FFF8 D084
S1RB	FFF8 D088
S2RB	FFF8 D08C
TPRB	FFF8 D090
TTRB	FFF8 D094
RPRB	FFF8 D098
RTRB	FFF8 D09C
CTPRHB	FFF8 D0A0
CTPRLB	FFF8 D0A4
CTCRB	FFF8 D0A8
OMRB	FFF8 D0AC
CTHB	FFF8 D0B0
CTLB	FFF8 D0B4
PCRB	FFF8 D0B8
CCRB	FFF8 D0BC
TxFIFOB	FFF8 D0C0
RxFIFOB	FFF8 D0D0

continued

(continued)

Table A–1 Address Map

Resource Name	Address
Synchronous Serial Interface (continued)	
Port B (continued)	
RSRB	FFF8 D0E0
TRSRB	FFF8 D0E4
ICTSRB	FFF8 D0E8
IERB	FFF8 D0F0
DTRTB	FFF8 D0F4
DMA Interface	
DMA0 – Synchronous Serial Channel A Receive	
SG	FFF8 D804
AD	FFF8 D808
CNT	FFF8 D810
CMD	FFF8 D820
STAT	FFF8 D840
DMA1 – Synchronous Serial Channel A Transmit	
SG	FFF8 D904
AD	FFF8 D908
CNT	FFF8 D910
CMD	FFF8 D920
STAT	FFF8 D940
DMA2 – Synchronous Serial Channel B Receive	
SG	FFF8 DA04
AD	FFF8 DA08
CNT	FFF8 DA10
CMD	FFF8 DA20
STAT	FFF8 DA40
DMA3 – Synchronous Serial Channel B Transmit	
SG	FFF8 DB04
AD	FFF8 DB08
CNT	FFF8 DB10
CMD	FFF8 DB20
STAT	FFF8 DB40
DMA4 – Parallel Printer	
SG	FFF8 DC04
AD	FFF8 DC08
CNT	FFF8 DC10
CMD	FFF8 DC20
STAT	FFF8 DC40
Parallel Interface	FFF8 DF00 – FFF8 DF1F
PD	FFF8 DF00
PSC	FFF8 DF04

(continued)

Table A–1 Address Map

Resource Name	Address
Extended interrupts	
EXIEN0	FFF8 E004
EXIEN1	FFF8 E008
EXIEN2	FFF8 E010
EXIEN3	FFF8 E020
EXIEN	FFF8 E03C
EXIST	FFF8 E040
Counter/timer	
PIT0_CNT	FFF8 F004
PIT1_CNT	FFF8 F008
PIT2_CNT	FFF8 F010
PIT3_CNT	FFF8 F020
PIT0_C/S	FFF8 F044
PIT1_C/S	FFF8 F048
PIT2_C/S	FFF8 F050
PIT3_C/S	FFF8 F060
PIT_CMD_ALL	FFF8 F07C
RTC_CNT	FFF8 F084
RTC_C/S	FFF8 F088
Memory	
MDC, MDS	FFF8 FF00
ECB, DLE	FFF8 FF04
EEAL	FFF8 FF08
EEAH	FFF8 FF0C
RESUME/PERSONALITY	FFF8 FF10
Reserved	FFF8 FF14 – FFF8 FF7F
I/O Configuration	
CONFIG	FFF8 FFFC
Reserved	FFF9 0000 – FFFA FFFF

(continued)

Table A–1 Address Map

Resource Name	Address
SCSI Interface	
SIEN	FFFB 0000
SDID	FFFB 0001
SCNTL1	FFFB 0002
SCNTL0	FFFB 0003
SOCL	FFFB 0004
SODL	FFFB 0005
SXFER	FFFB 0006
SCID	FFFB 0007
SBCL	FFFB 0008
SBDL	FFFB 0009
SIDL	FFFB 000A
SFBR	FFFB 000B
SSTAT2	FFFB 000C
SSTAT1	FFFB 000D
SSTAT0	FFFB 000E
DSTAT	FFFB 000F
Reserved	FFFB 0010 – FFFB 0013
CTEST3	FFFB 0014
CTEST2	FFFB 0015
CTEST1	FFFB 0016
CTEST0	FFFB 0017
CTEST7	FFFB 0018
CTEST6	FFFB 0019
CTEST5	FFFB 001A
CTEST4	FFFB 001B
TEMP	FFFB 001C – FFFB 001F
Reserved	FFFB 0020 – FFFB 0021
ISTAT	FFFB 0022
DFIFO	FFFB 0023
DCMD	FFFB 0024
DBC	FFFB 0025 – FFFB 0027
DNAD	FFFB 0028 – FFFB 002B
DSP	FFFB 002C – FFFB 002F
DSPS	FFFB 0030 – FFFB 0033
Reserved	FFFB 0034 – FFFB 0036
DMODE	FFFB 0037
DCNTL	FFFB 0038
DWT	FFFB 0039
DIEN	FFFB 003A
Reserved	FFFB 003B
Reserved	FFFB 003C – FFFB 003F
SCSI FUSE SENSE	FFFB 0040
Expansion SCSI Interface	FFFB 0080 – FFFB 00BF

CAUTION: Use caution when accessing TEMP, DBC, DNAD, DSP, DSPS. The bytes must be swapped as defined in Chapter 9, "Programming the LAN and SCSI Interfaces."

(continued)

Table A–1 Address Map

Resource Name	Address
LAN Interface (1st LAN board)	
LAN FUSE SENSE	FFFB 00C0
Data Port	FFFB 0100
Register Address Port	FFFB 0104
LANID−BYTE0	FFFB 0110
LANID−BYTE1	FFFB 0114
LANID−BYTE2	FFFB 0118
LANID−BYTE3	FFFB 011C
LANID−BYTE4	FFFB 0120
LANID−BYTE5	FFFB 0124
LANID−BYTE6	FFFB 0128
LANID−BYTE7	FFFB 012C
LAN Interface (2nd LAN board)	
Data Port	FFFB 0140
Register Address Port	FFFB 0144
LANID−BYTE0	FFFB 0150
LANID−BYTE1	FFFB 0154
LANID−BYTE2	FFFB 0158
LANID−BYTE3	FFFB 015C
LANID−BYTE4	FFFB 0160
LANID−BYTE5	FFFB 0164
LANID−BYTE6	FFFB 0168
LANID−BYTE7	FFFB 016C
Reserved	FFFB 0160 − FFFC EFFF
I/O BOARD ID 0	FFFC F000
I/O BOARD ID 1	FFFC F004
Reserved	FFFC F020 − FFFC FFFF
Unused	FFFD 0000 − FFFE FFFF
VME bus A16 space (64 Kbytes)	FFFF 0000 − FFFF FFFF

(concluded)

End of Appendix

Appendix B
Powerup Flowchart

This appendix illustrates how the system hardware powers up, including power−up tests, indicators, initialization and reset.

A cold start is when the system is starting up from a complete powerdown. A cold start includes a powering−up phase during which the system voltages are generated and stabilize. A warm start is when a system is starting up from a system reset; the power supply was not shut down.

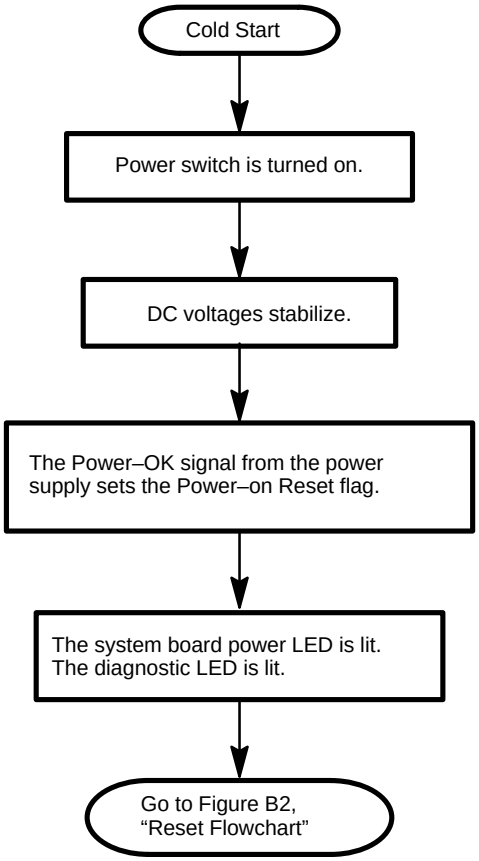


Figure B–1 Initial Powerup Flowchart

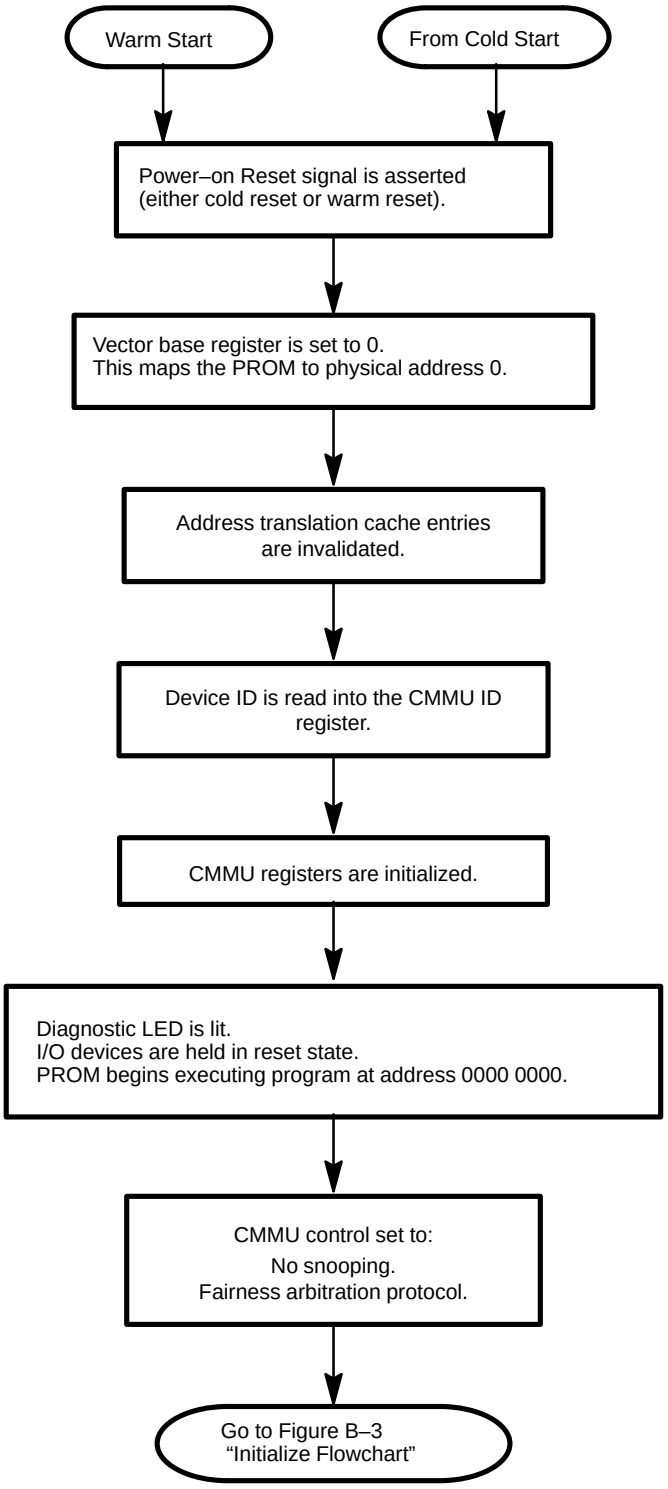


Figure B–2 Reset Flowchart

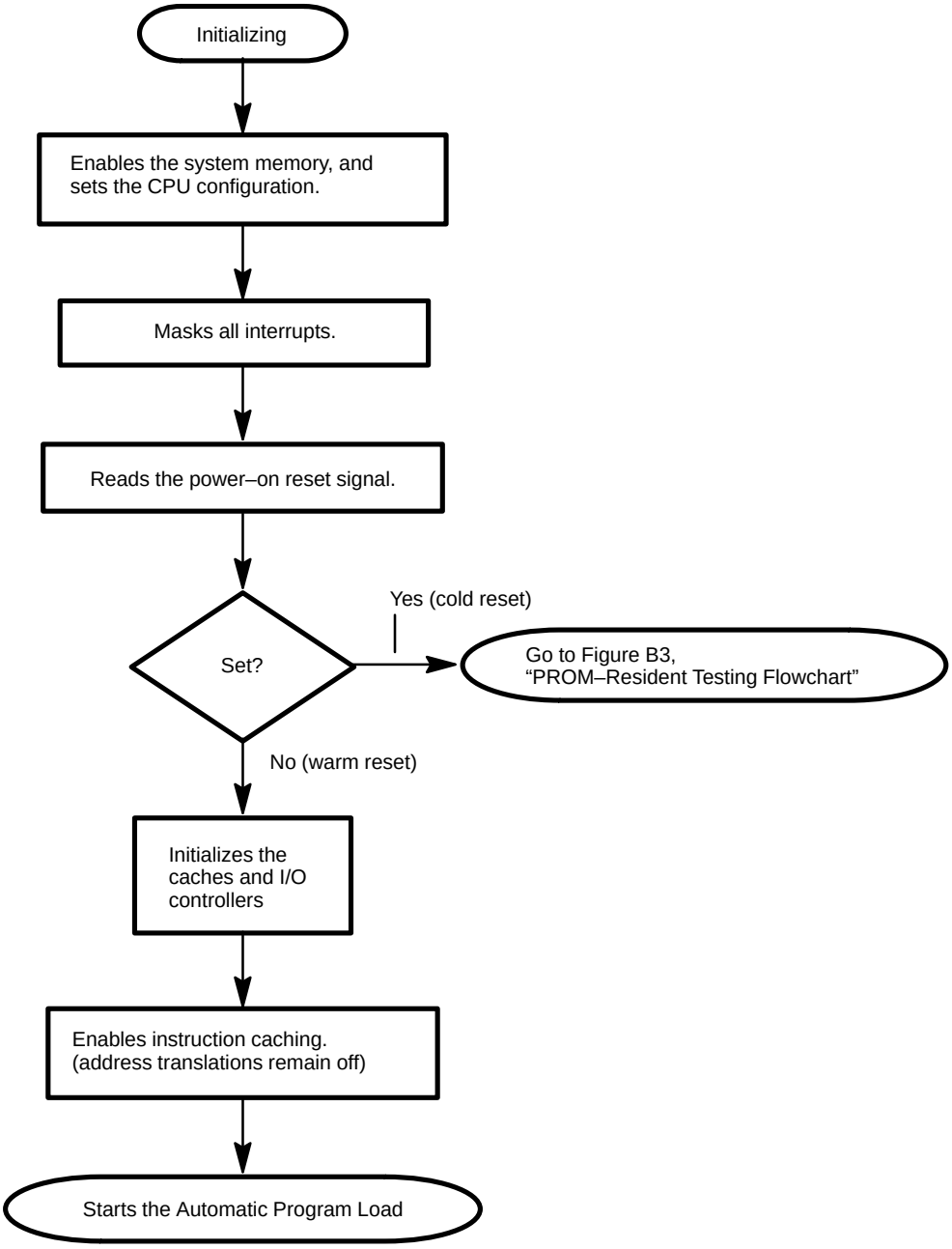


Figure B–3 Initialize Flowchart

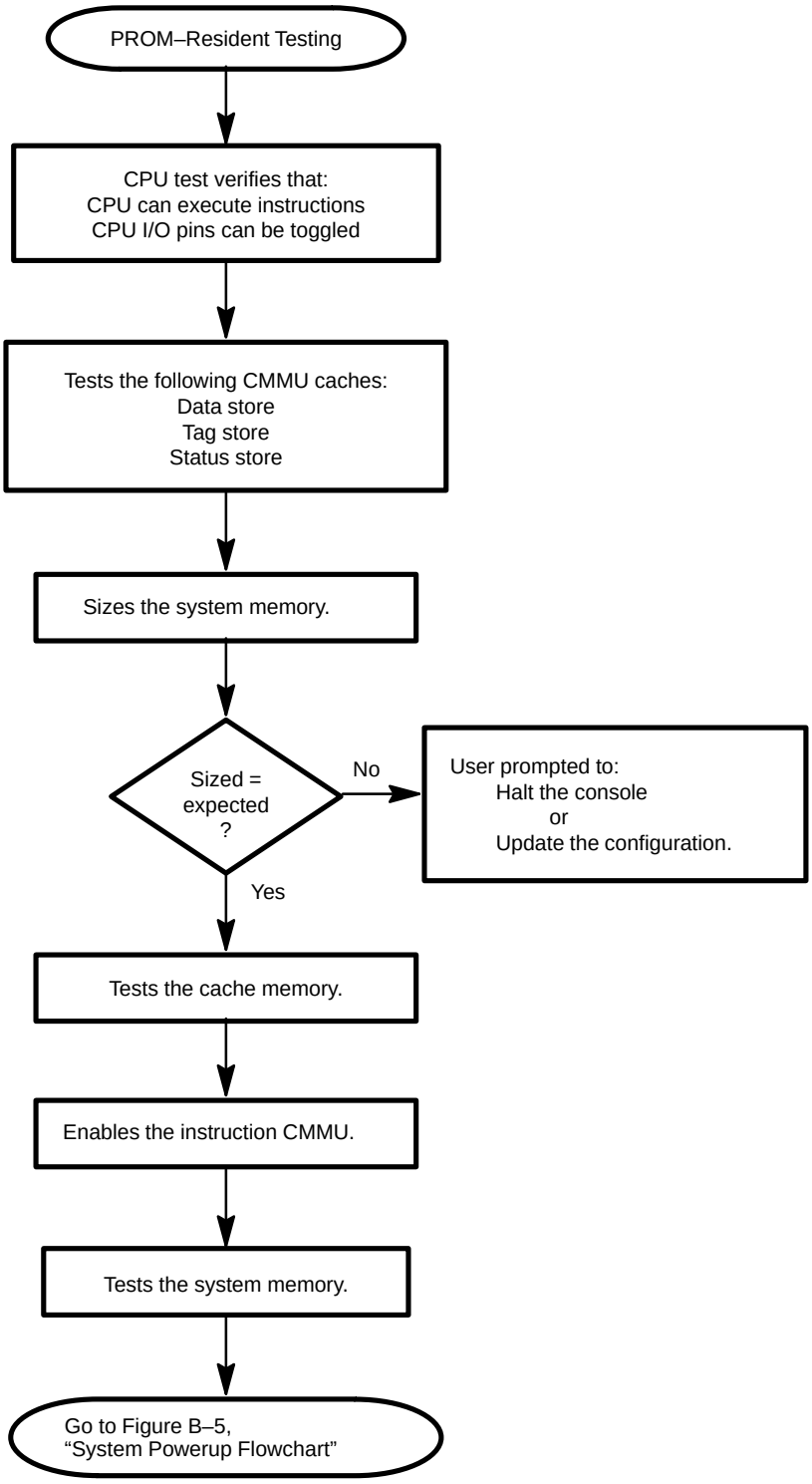


Figure B–4 PROM–Resident Testing Flowchart

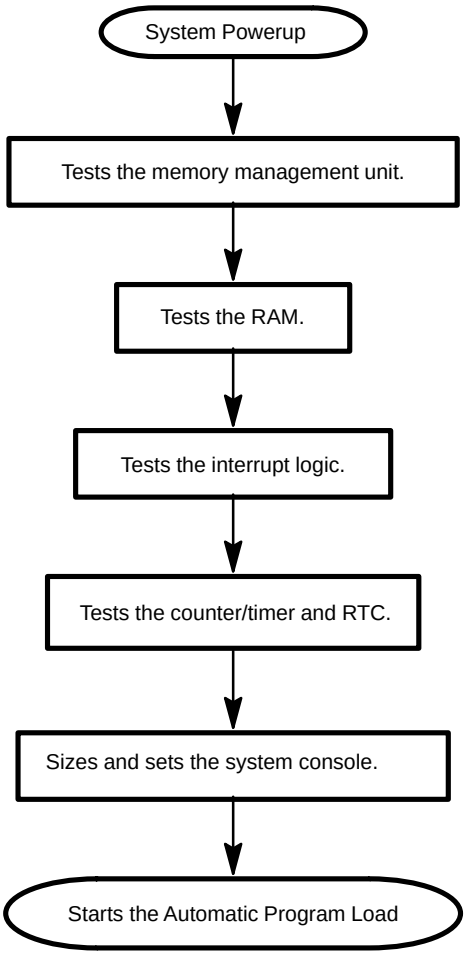
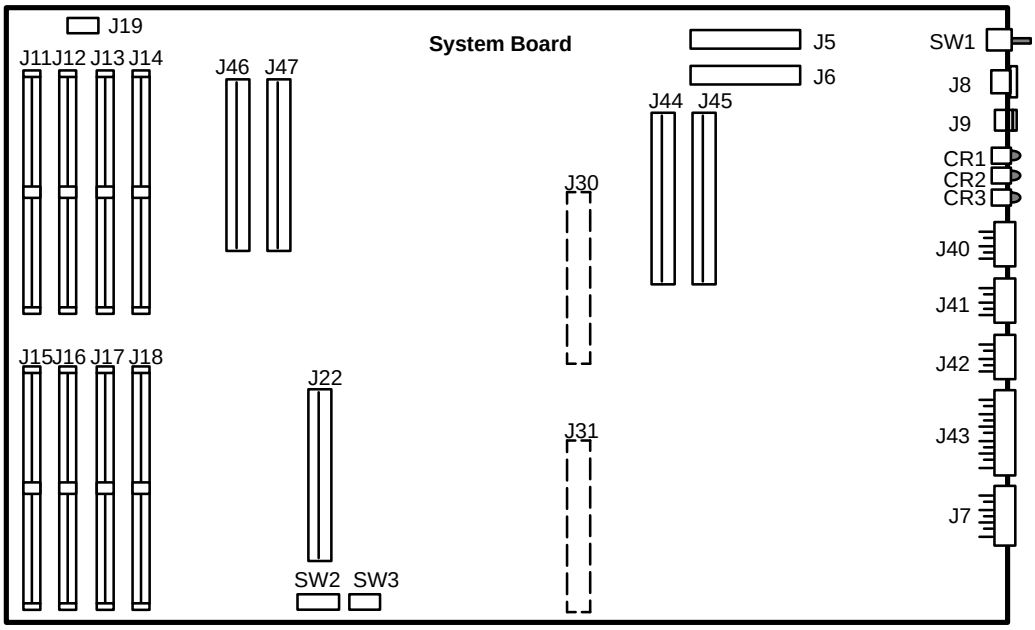


Figure B–5 System Powerup Flowchart

End of Appendix

Appendix C
System Board Connectors

The tables contained in this appendix identify the signals on the system board connectors.



J5 = SCSI
J6 = Synchronous Serial
J7 = LAN
J8 = Keyboard
J9 = Mouse
J19 = Speaker and LED
J22 = Graphics Controller Board
CR1 = Fail (yellow)
CR2 = Halt (yellow)
CR3 = Run (green)

J11 = Memory Module 0 Bank A
J12 = Memory Module 1 Bank A
J13 = Memory Module 2 Bank A
J14 = Memory Module 3 Bank A
J15 = Memory Module 0 Bank B
J16 = Memory Module 1 Bank B
J17 = Memory Module 2 Bank B
J18 = Memory Module 3 Bank B

J30 = VME P1
J31 = VME P2
J40 = Modem
J41 = Modem
J42 = System Console
J43 = Parallel Port
J44 = I/O Board J1
J45 = I/O Board J2
J46 = CPU Board J1
J47 = CPU Board J2

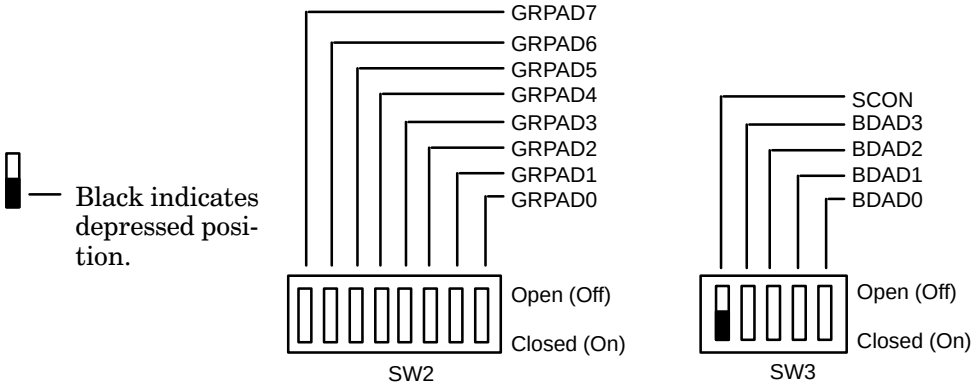


Figure C–1 System Board Connectors

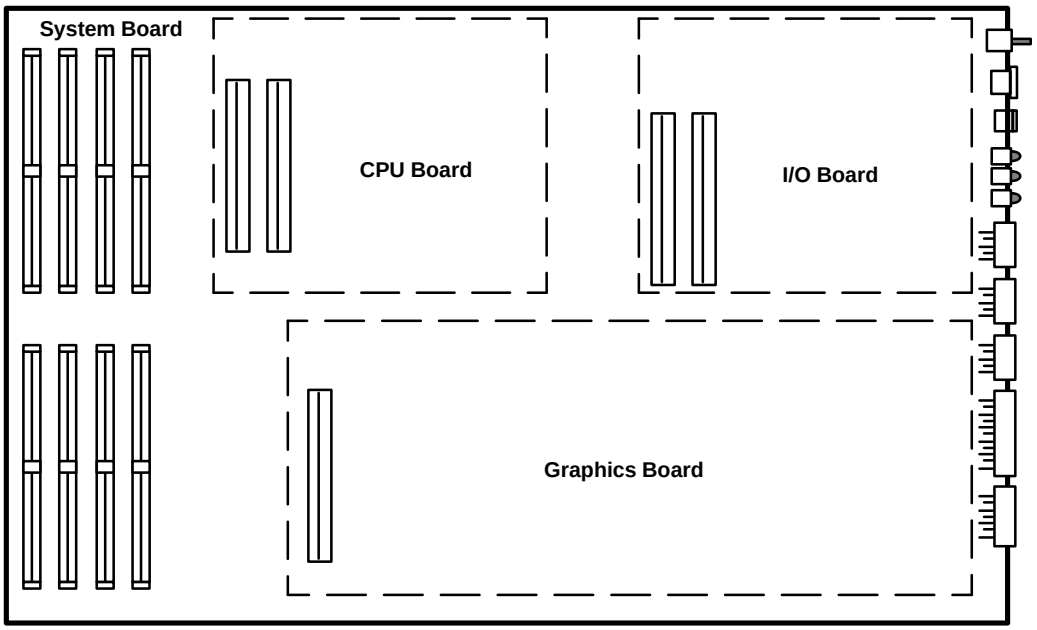


Figure C–2 Plug-In CPU and Controller Boards

J5: SCSI

Table C−1 describes connector J5.

Table C–1 Connector J5: SCSI Port

Odd Pins		Even Pins	
1	GND	2	SD0
3	GND	4	SD1
5	GND	6	SD2
7	GND	8	SD3
9	GND	10	SD4
11	GND	12	SD5
13	GND	14	SD6
15	GND	16	SD7
17	GND	18	SDP
19	GND	20	GND
21	GND	22	GND
23	GND	24	GND
25	−−−−−	26	TERM_PWR
27	GND	28	GND
29	GND	30	GND
31	GND	32	ATN
33	GND	34	GND
35	GND	36	BSY
37	GND	38	ACK
39	GND	40	RST
41	GND	42	MSG
43	GND	44	SEL
45	GND	46	C/D
47	GND	48	REQ
49	GND	50	I/O

J6: Synchronous Serial

Table C−2 describes connector J6.

Table C–2 Connector J6: Synchronous Serial Port

Odd Pins		Even Pins	
1	Unused	2	Unused
3	Unused	4	TXCO_2
5	Unused	6	Unused
7	Unused	8	Unused
9	Unused	10	Unused
11	DCD_2	12	DTR_2
13	GND	14	Unused
15	DSR_2	16	Unused
17	CTS_2	18	RXC_2
19	RTS_2	20	Unused
21	RXD_2	22	TXCI_2
23	TXD_2	24	Unused
25	Unused	26	Unused
27	Unused	28	Unused
29	TXCO_1	30	Unused
31	Unused	32	Unused
33	Unused	34	Unused
35	Unused	36	DCD_1
37	DTR_1	38	GND
39	Unused	40	DSR_1
41	Unused	42	CTS_1
43	RXC_1	44	RTS_1
45	Unused	46	RXD_1
47	TXCI_1	48	TXD_1
49	Unused	50	Unused

J7: LAN

Table C−3 describes connector J7.

Table C–3 Connector J7: LAN

Pins			
1	GND	10	XMT_NEG
2	COLL_PLUS	11	GND
3	XMT_PLUS	12	RCV_NEG
4	GND	13	+12V
5	RCV_PLUS	14	GND
6	GND	15	−−−−
7	−−−−	16	GND
8	GND	17	GND
9	COL_NEG		

J8: Keyboard

Table C−4 describes connector J8.

Table C–4 Connector J8: Keyboard

Pins	
1	----
2	GND
3	+5V
4	----
5	----
6	----
7	GND
8	GND
9	KBD_DATA
10	KBD_CLOCK

J9: Mouse

Table C−5 describes connector J9.

Table C–5 Connector J9: Mouse

Pins	
1	RTS
2	DTR
3	----
4	----
5	----
6	GND
7	TXD
8	RXD
9	GND
10	GND
11	GND

J11 – J18: Memory

Table C–6 describes connectors J11 – J18.

Table C–6 Connectors J11 – J18: Memory

Odd Pins		Even Pins	
1	GND	2	D0
3	D1	4	D2
5	D3	6	+5V
7	D4	8	D5
9	D6	10	D7
11	GND	12	D8
13	D9	14	D10
15	D11	16	+5V
17	D12	18	D13
19	D14	20	D15
21	GND	22	– – – –
23	ECC0	24	ECC1
25	ECC2	26	+5V
27	ECC3	28	ECC4
29	ECC5	30	ECC6
31	GND	32	D16
33	D17	34	D18
35	D19	36	GND
37	D20	38	D21
39	D22	40	D23
41	D24	42	D25
43	D26	44	D27
45	GND	46	D28
47	D29	48	D30
49	D31	50	GND
51	$\overline{\text{4MEG}}$	52	– – – –
53	$\overline{\text{100\_NS}}$	54	– – – –
55	+5V	56	WE0
57	WE1	58	WE2
59	WE3	60	GND
61	– – – –	62	$\overline{\text{ENRAS0}}$
63	$\overline{\text{RAS0}}$	64	CAS0
65	+5V	66	– – – –
67	A10	68	A9
69	A8	70	GND
71	A7	72	A6
73	A5	74	A4
75	+5V	76	A3
77	A2	78	A1
79	A0	80	GND

J19: Speaker and LED

Table C−7 describes connector J19.

Table C–7 Connector J19: Speaker and LED

Pins	
1	SPEAKER_ON
2	+5V
3	----
4	LED_ON

J22: Graphics

Table C−16 describes connector J22.

Table C–8 Connector J22: Graphics Connector

Row A		Row B		Row C	
1	GCAD0	1	GKAD1	1	GKAD2
2	GCAD3	2	GKAD4	2	GKAD5
3	+5V	3	GKAD6	3	+5V
4	GKAD7	4	GND	4	GRAD8
5	GRSD9	5	GRAD10	5	GRAD11
6	GND	6	GRAD12	6	GND
7	GRAD13	7	GRAD14	7	GRAD15
8	GRAD16	8	+5V	8	GRAD17
9	GND	9	GRAD18	9	GND
10	GRAD19	10	GRAD20	10	GRAD21
11	GRAD22	11	GND	11	GRAD23
12	+5V	12	GRAD24	12	+5V
13	GRAD25	13	GRAD26	13	GRAD27
14	GRAD28	14	GND	14	GRAD29
15	GND	15	GRAD30	15	GND
16	GRAD31	16	GADP	16	GADP1
17	GADP	17	+5V	17	GADP3
18	GND	18	G_C0	18	GND
19	ST0_VIDEO	19	MBUS16_C2	19	CLK16_OLGRAPH
20	$\overline{\text{ST1\_VIDEO}}$	20	GND	20	BB_VIDEO
21	+5V	21	$\overline{\text{G\_SS1}}$	21	+5V
22	$\overline{\text{IRQ\_VIDEO}}$	22	$\overline{\text{OGRPHCS/CS\_QUAL}}$	22	----
23	$\overline{\text{IRQ\_VIDEOZ}}$	23	GND	23	----
24	GND	24	$\overline{\text{BRESET}}$	24	GND
25	----	25	----	25	----
26	----	26	+5V	26	----
27	GND	27	----	27	GND
28	----	28	----	28	----
29	----	29	GND	29	----
30	+5V	30	----	30	+5V
31	----	31	----	31	OGRPHCS/GXVER_DIR
32	OGRPHCS/GXTL_125	32	MONITOR_SPEED	32	$\overline{\text{OLD\_GRAPHICS}}$

J30/P1: VMEbus

Table C−9 describes connector J30.

Table C–9 Connector J30: VMEbus

Row A		Row B		Row C	
1	D00	1	$\overline{\text{BBSY}}$	1	D08
2	D01	2	$\overline{\text{BCLR}}$	2	D09
3	D02	3	$\overline{\text{ACFAIL}}$	3	D10
4	D03	4	$\overline{\text{BG0IN}}$	4	D11
5	D04	5	$\overline{\text{BG0OUT}}$	5	D12
6	D05	6	$\overline{\text{BG1IN}}$	6	D13
7	D06	7	$\overline{\text{BG1OUT}}$	7	D14
8	D07	8	$\overline{\text{BG2IN}}$	8	D15
9	GND	9	$\overline{\text{BG2OUT}}$	9	GND
10	SYSCLK	10	$\overline{\text{BG3IN}}$	10	$\overline{\text{SYSFAIL}}$
11	GND	11	$\overline{\text{BG3OUT}}$	11	$\overline{\text{BERR}}$
12	$\overline{\text{DS1}}$	12	$\overline{\text{BR0}}$	12	$\overline{\text{SYSRESET}}$
13	$\overline{\text{DS0}}$	13	$\overline{\text{BR1}}$	13	$\overline{\text{LWORD}}$
14	$\overline{\text{WRITE}}$	14	$\overline{\text{BR2}}$	14	AM5
15	GND	15	$\overline{\text{BR3}}$	15	A23
16	$\overline{\text{DTACK}}$	16	AM0	16	A22
17	GND	17	AM1	17	A21
18	$\overline{\text{AS}}$	18	AM2	18	A20
19	GND	19	AM3	19	A19
20	$\overline{\text{IACK}}$	20	GND	20	A18
21	$\overline{\text{IACKIN}}$	21	Unused	21	A17
22	$\overline{\text{IACKOUT}}$	22	Unused	22	A16
23	AM4	23	GND	23	A15
24	A07	24	$\overline{\text{IRQ7}}$	24	A14
25	A06	25	$\overline{\text{IRQ6}}$	25	A13
26	A05	26	$\overline{\text{IRQ5}}$	26	A12
27	A04	27	$\overline{\text{IRQ4}}$	27	A11
28	A03	28	$\overline{\text{IRQ3}}$	28	A10
29	A02	29	$\overline{\text{IRQ2}}$	29	A09
30	A01	30	$\overline{\text{IRQ1}}$	30	A08
31	−12V	31	Unused	31	+12V
32	+5V	32	+5V	32	+5V

J31/P2: VMEbus

Table C−10 describes connector J31.

Table C–10 Connector J31: VMEbus

Row A		Row B		Row C	
1	+5V	1	+5V	1	GND
2	+5V	2	GND	2	GND
3	+5V	3	Reserved	3	GND
4	+5V	4	A24	4	GND
5	+5V	5	A25	5	GND
6	+5V	6	A26	6	GND
7	+5V	7	A27	7	GND
8	+5V	8	A28	8	GND
9	+5V	9	A29	9	GND
10	+5V	10	A30	10	GND
11	+5V	11	A31	11	GND
12	+5V	12	GND	12	GND
13	+5V	13	+5V	13	GND
14	+5V	14	D16	14	GND
15	+5V	15	D17	15	GND
16	+5V	16	D18	16	GND
17	+5V	17	D19	17	GND
18	+5V	18	D20	18	GND
19	+5V	19	D21	19	GND
20	+5V	20	D22	20	GND
21	+5V	21	D23	21	GND
22	+5V	22	GND	22	GND
23	+5V	23	D24	23	GND
24	Unused	24	D25	24	GND
25	Unused	25	D26	25	GND
26	Unused	26	D27	26	GND
27	+12V	27	D28	27	GND
28	Unused	28	D29	28	GND
29	GND	29	D30	29	GND
30	GND	30	D31	30	GND
31	GND	31	GND	31	GND
32	GND	32	+5V	32	GND

J40 – J42: Asynchronous RS–232–C Ports

Table C–11 describes connector J40.

Table C–11 Connector J40: RS-232-C Port C (Full Modem)

Pins	
1	DCDC
2	RXDC
3	TXDC
4	DTRC
5	GND
6	DSRC
7	RTSC
8	CTSC
9	RIC

Table C–12 describes connector J41.

Table C–12 Connector J41: RS-232-C Port B (Full Modem)

Pins	
1	DCDB
2	RXDB
3	TXDB
4	DTRB
5	GND
6	DSRB
7	RTSB
8	CTSB
9	RIB

Table C–13 describes connector J42.

Table C–13 Connector J42: RS-232-C Port A (Console)

Pins	
1	DCDA
2	RXDA
3	TXDA
4	DTRA
5	GND
6	----
7	RTSA
8	CTSA
9	----

J43: Parallel Port

Table C−14 describes connector J43.

Table C–14 Connector J43: Parallel Port

Pin	Signal
1	STROBE
2	DATA0
3	DATA1
4	DATA2
5	DATA3
6	DATA4
7	DATA5
8	DATA6
9	DATA7
10	$\overline{\text{ACK}}$
11	BUSY
12	PE
13	SELECT
14	−−−−
15	FAULT
16	RESET
17	−−−−
18	GND
19	GND
20	GND
21	GND
22	GND
23	GND
24	GND
25	GND

J44: I/O Connector

Table C−15 describes connector J44. J44 connects to J1 of the SCSI board or LAN board.

Table C–15 Connector J44: I/O Connector

Row A		Row B		Row C	
1	$\overline{\text{INT1}}$	1	GND	1	$\overline{\text{GRAPHICS\_CS}}$
2	GND	2	$\overline{\text{IO\_SEL}}$	2	GND
3	AB	3	+5V	3	$\overline{\text{LAN1\_INTR}}$
4	GND	4	SCSI1_HOLD	4	----
5	----	5	−12V	5	GRAPHICS_HOLD
6	GND	6	IO_RESET	6	GND
7	$\overline{\text{SLAN1\_HOLD}}$	7	GND	7	$\overline{\text{BRDY}}$
8	+5V	8	Reserved	8	+5V
9	$\overline{\text{BE2}}$	9	GND	9	$\overline{\text{BE1}}$
10	GND	10	SCSI0_HOLD	10	$\overline{\text{IO\_ST3}}$
11	$\overline{\text{ADS}}$	11	GND	11	$\overline{\text{RDY}}$
12	GND	12	D30	12	GND
13	A30	13	GND	13	A31
14	GND	14	A27	14	GND
15	D27	15	+5V	15	A28
16	GND	16	D24	16	GND
17	A24	17	GND	17	A25
18	GND	18	D21	18	GND
19	A21	19	GND	19	A22
20	+5V	20	D18	20	+5V
21	A18	21	GND	21	A19
22	GND	22	D15	22	GND
23	A15	23	GND	23	A16
24	GND	24	D12	24	GND
25	A12	25	GND	25	A13
26	GND	26	D9	26	GND
27	A9	27	+5V	27	A10
28	GND	28	D6	28	GND
29	A6	29	+5V	29	A7
30	GND	30	D3	30	GND
31	A3	31	GND	31	A4
32	+5V	32	D0	32	+5V

J45: I/O Connector

Table C–16 describes connector J45.

Table C–16 Connector J45: I/O Connector

Row A		Row B		Row C	
1	NEW_GRAPHICS	1	EXP_INT0	1	LAN1_DEC
2	SCSI1_DEC	2	GND	2	BRESET
3	+5V	3	LANID1_DEC	3	+5V
4	SCSI1_INTR	4	----	4	GRAPHICS_INTR
5	MASTER	5	LAN1_HLDA	5	BLAST
6	GRAPHICS_HLDA	6	GND	6	HLDA
7	GND	7	CLK16_IOCONN1	7	GND
8	BRDID_DEC	8	GND	8	SLAN0_HOLD
9	GND	9	BE0	9	GND
10	+5V	10	+5V	10	OLD_GRAPH_SPEAKER
11	GND	11	READ	11	GND
12	D31	12	+12V	12	BE3
13	GND	13	A29	13	GND
14	D28	14	GND	14	D29
15	+5V	15	A26	15	+5V
16	D25	16	GND	16	D26
17	----	17	A23	17	GND
18	D22	18	GND	18	D23
19	GND	19	A20	19	GND
20	D19	20	+5V	20	D20
21	GND	21	A17	21	GND
22	D16	22	GND	22	D17
23	GND	23	A14	23	GND
24	D13	24	+5V	24	D14
25	GND	25	A11	25	GND
26	D10	26	GND	26	D11
27	+5V	27	A8	27	+5V
28	D7	28	GND	28	D8
29	GND	29	A5	29	GND
30	D4	30	GND	30	D5
31	GND	31	A2	31	GND
32	D1	32	+5V	32	D2

J46: CPU Board

Table C−17 describes connector J46.

Table C–17 Connector J46: CPU Board

Row A		Row B		Row C	
101	WHOAMI2	201	GND	301	SS1
102	GND	202	SS3	302	GND
103	SS0	203	+5V	303	CPU_INT0
104	GND	204	BR3	304	STB1
105	BR_A	205	Reserved	305	BGA
106	GND	206	STA0	306	GND
107	STB0	207	GND	307	WHOAMI1
108	+5V	208	BBA	308	+5V
109	C4	209	GND	309	C5
110	GND	210	CLK	310	Reserved
111	C0	211	GND	311	C1
112	GND	212	DATA30	312	GND
113	ADDR30	213	GND	313	ADDR31
114	GND	214	DATA27	314	GND
115	ADDR27	215	+5V	315	ADDR28
116	GND	216	DATA24	316	GND
117	ADDR24	217	Reserved	317	ADDR25
118	GND	218	DATA21	318	GND
119	ADDR21	219	GND	319	ADDR22
120	+5V	220	DATA18	320	+5V
121	ADDR18	221	GND	321	ADDR19
122	GND	222	DATA15	322	GND
123	ADDR15	223	GND	323	ADDR16
124	GND	224	DATA12	324	GND
125	ADDR12	225	GND	325	ADDR13
126	GND	226	DATA9	326	GND
127	ADDR9	227	+5V	327	ADDR10
128	GND	228	DATA6	328	GND
129	ADDR6	229	Reserved	329	ADDR7
130	GND	230	DATA3	330	GND
131	ADDR3	231	GND	331	ADDR4
132	+5V	232	DATA0	332	+5V

J47: CPU Board

Table C−18 describes connector J47.

Table C–18 Connector J47: CPU Board

Row A		Row B		Row C	
101	$\overline{\text{PEND\_CLR}}$	201	CPU_INT1	301	PLLEN
102	PEND	202	GND	302	$\overline{\text{RST}}$
103	+5v	203	SS2	303	+5V
104	CPU_INT2	204	WHOAMI3	304	CPU_INT3
105	Reserved	205	BAA	305	WHOAMI0
106	BG3	206	GND	306	BAB
107	GND	207	STB2	307	GND
108	BB3	208	+5V	308	$\overline{\text{RR\_AB}}$
109	GND	209	C6	309	GND
110	STA1	210	GND	310	STA2
111	GND	211	C2	311	GND
112	DATA31	212	Reserved	312	C3
113	GND	213	ADDR29	313	GND
114	DATA28	214	GND	314	DATA29
115	+5V	215	ADDR26	315	+5V
116	DATA25	216	GND	316	DATA26
117	Reserved	217	ADDR23	317	GND
118	DATA22	218	GND	318	DATA23
119	GND	219	ADDR20	319	GND
120	DATA19	220	+5V	320	DATA20
121	GND	221	ADDR17	321	GND
122	DATA16	222	GND	322	DATA17
123	GND	223	ADDR14	323	GND
124	DATA13	224	Reserved	324	DATA14
125	GND	225	ADDR11	325	GND
126	DATA10	226	GND	326	DATA11
127	+5V	227	ADDR8	327	+5V
128	DATA7	228	GND	328	DATA0
129	Reserved	229	ADDR5	329	GND
130	DATA4	230	GND	330	DATA5
131	GND	231	ADDR2	331	GND
132	DATA1	232	+5V	332	DATA2

End of Appendix

Index

A

B

C

D

E

F

G

H

I

K

L

M

P

R

S

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   A. MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   B. Send your order form with payment to: Data General Corporation
   ATTN: Educational Services/TIPS G155
   4400 Computer Drive
   Westboro, MA 01581–9973

   C. TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   A. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   B. Check or Money Order – Make payable to Data General Corporation. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
Over 200 Items	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$0–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	**SHIP TO:** (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME_____	COMPANY NAME_____
ATTN:_____	ATTN:_____
ADDRESS_____	ADDRESS (NO PO BOXES)_____
CITY_____	CITY_____
STATE_____ ZIP_____	STATE_____ ZIP_____

Priority Code _____ (See label on back of catalog)

_____ _____ _____ _____ _____
Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

	ADD
☐ UPS	
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0–$149.99	0%
$150–$499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	–
SUB TOTAL	
Your local* sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is_____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number | Expiration Date

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508–870–1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation (“DGC”) provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC’s net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC’s sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC’S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC’S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC’s acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Programming
System Control
and I/O Registers:
AViiON® 4600 and
530 Series

014−002076−01

Cut here and insert in binder spine pocket