SYS68K/CPU-30 R4
Technical Reference Manual

Edition No. 2
February 1997

P/N 204030
FORCE COMPUTERS Inc./GmbH

TABLE OF CONTENTS

List of Figures

List of Tables

1 Introduction

1.1 Getting Started

This *SYS68K/CPU-30 R4 Technical Reference Manual* provides a comprehensive guide to the CPU-30 R4 board you purchased from FORCE COMPUTERS. In addition, each board delivered by FORCE includes an *Installation Guide.*

CAUTION: Before installing the board, please read the complete installation instructions.

1.1.1 SYS68K/CPU-30 R4 Technical Reference Manual Set

When purchased from FORCE, this set includes the *SYS68K/CPU-30 R4 Technical Reference Manual*, a copy of the circuit schematics, and copies of the following data sheets:

RTC 72421	FDC37C65C[1)]
DUSCC 68562	LANCE Am79C90
PI/T TS68230	SIA Am7992B
SCSI 87033/34	Motorola MC68030 and MC68882

1. The FDC37C65C is pin-to-pin compatible with Industry Standard WD37C65C

1.1.2 Overview of the Manual

Section 1 provides a brief overview of the product, the specifications, the ordering information, and the publication history of the manual. Information concerning the installation, default configuration, initialization, and connector pinouts is included in Section 2. A detailed hardware description is described in Section 3. The CPU board operates under the control of VMEPROM, which is described in Sections 5, 6, 7, 8, and 9. There is additional space allocated in the manual for user notes, modifications, etc.

NOTE: Please take a moment to examine the Table of Contents of the *SYS68K/CPU-30 R4 Technical Reference Manual* to see how this documentation is structured. This will be of value to you when looking for information in the future.

1.2 Overview of the SYS68K/CPU-30 R4

This CPU board is a high performance single-board computer based on the 68030 microprocessor and the VMEbus. The CPU board also includes an enhanced Floating Point Coprocessor 68882. The board design utilizes all of the features of the powerful FORCE Gate Array FGA-002.

The CPU-30 R4 provides an A32/D32 VMEbus interface including DMA, up to 32 Mbyte shared DRAM on-board, up to 8 Mbyte System Flash Memory, an Ethernet Interface, a single-ended SCSI interface, a floppy interface, four RS-232 serial I/O channels, up to 256 Kbyte SRAM and a Real-Time Clock, both with on-board battery backup.

Besides the CPU-30 R4, there will be a CPU-30Lite R4 without a coprocessor, a SCSI, an Ethernet, and a floppy disk interface.

SEE ALSO: Please refer to Table 2, “Ordering Information,” on page 9 for more detailed information.

The shared DRAM is accessible from the 68030 CPU, the FGA-002 DMA controller, and also from other VMEbus masters.

The CPU-30 R4 has an Ethernet port and three serial ports available on the front panel permitting a console port, download and data communication channels. One further serial port, as well as the SCSI interface and the Floppy interface are available via the 3-row VMEbus P2 connector. A 20-bit parallel interface and the three serial ports on the front panel are available via the optional 5-row VMEbus P2 connector.

1.2.1 Features of the CPU-30 R4

The main features of the SYS68K/CPU-30 R4 board are listed below.

Processor

- 68030 with 25 MHz frequency
- Flexible high bandwidth synchronous bus
- 68020 compatible integer unit
- Memory management unit
- Independent data and instruction memory management units
- Dual 256-byte on-chip caches for instructions and data
- 4-Gbyte addressing range
- Upward user object code compatible with the 68020

Coprocessor

- 68882 with 25 MHz frequency
- Pin- and SW-compatible with the MC68881

Main Memory

- 4, 8, 16, or 32 Mbyte of shared DRAM on-board
- Byte parity

VMEbus Interface

- Via FGA-002 in the 304-pin PQFP package
- Slot-1 detection switch enabled
- **Master:**

  A32, A24, A16: D8, D16, D32, ADO, UAT, RMW

  AM CODES:

  Standard supervisory data/program access

  Standard non-privileged data/program access

  Short supervisory access

  Short non-privileged access

  Extended supervisory data/program access

  Extended non-privileged data/program access
- **Slave:**

  A32: D8, D16, D32, ADO, UAT, RMW

  Access Address: SW programmable (FGA-002)

  AM CODES:

  Standard supervisory data/program access

  Standard non-privileged data/program access

  Extended supervisory data/program access

  Extended non-privileged data/program access

  SW programmable inside FGA-002 for DMA controller

  Single-level arbiter

  Request modes: ROR, RBCLR, REC, RAT

  IACK daisy chain driver

  FORCE Message Broadcast (FMB)

System Flash Memory

- 4 Mbyte Flash Memory is default configuration
- Up to 8 Mbyte Flash Memory
- 32-bit wide
- Reprogrammable on-board
- HW write protection

Ethernet Interface

- Via AM79C90
- Compatible with IEEE 802.3 Rev.0
- On-Chip DMA and buffer management
- 48-byte FIFO
- 24-bit wide linear addressing
- Network and packet error reporting
- Back-to-back reception with as little as 4.1 µs inter-packet gaptime
- AUI Ethernet available on the front panel via a standard 15-pin D-Sub connector.

SCSI Interface

- Via MB87033/34
- Full support for SCSI control
- Service of either initiator or target device
- 8-byte data buffer register incorporated
- Transfer byte counter (28-bit)
- Independent control and data transfer bus
- On-chip single-ended drivers/receivers
- SCSI interface available on the VMEbus P2 connector

Floppy Disk Interface

- Via FDC37C65C (the FCD37C65C is pin-to-pin compatible with Industry Standard WD37C65C)
- Available on VMEbus P2 connector

Boot ROM

- 128 -, 256 -, and 512 Kbyte of Flash Memory or up to 1 Mbyte OTP
- 8-bit wide
- Reprogrammable on-board in case of Flash Memory
- HW write protection in case of Flash Memory
- Two 32-pin PLCC sockets

Serial I/O Ports

- Via 68562 DUSCC
- Dual full-duplex asynchronous receiver and transmitter (programmable)
- Multi-protocol operation enabling support of bit- or character-oriented protocols
- Additional software allows the support of HDLC, SDLC, BISYNC, etc.
- Three ports available on the front panel via standard 9-pin D-Sub connectors
- One port available on standard 3-row VMEbus P2 connector.
- All channels support RS-232 or RS-422 via the FORCE hybrids FH-00x – as factory option also RS-485.
- RS-422 – and as factory option also RS-485 – terminations via cable resistors

Parallel I/O

- Available via two 68230 PI/T
- 20-bit user I/O available on optional 5-row VMEbus P2 connector, TTL level

SRAM

- 32-Kbyte or 128-Kbyte (assembly option) SRAM
- Optional 512-Kbyte SRAM in DIL package
- 8-bit wide
- On-board battery backup or backup via +5V-Stdby
- Socketed battery

RTC

- Real-Time Clock 72423
- IRQ capability
- Time of day and date counter included (year, month, week, day)
- Built-in quartz oscillator
- 12 hr/24 hr clock switch-over
- Automatic leap year setting
- CMOS design provides low power consumption during power-down mode
- On-board battery backup or backup via +5V-Stdby
- Socketed battery

Additional Features

- RESET/ABORT key
- Reset watchdog timer
- Status LEDs
- Rotary switches

1.3 Specifications

Table 1: Specifications for the CPU-30 R4 Board

CPU type	68030
CPU clock frequency	25 MHz
Shared DRAM capacity with parity CPU-30ZBE R4 CPU-30BE/8 R4 CPU-30BE/16 R4 factory option CPU-30Lite/4 R4 CPU-30Lite/8 R4	 4 Mbyte 8 Mbyte 16 Mbyte 32 Mbyte 4 Mbyte 8 Mbyte
SRAM capacity with on-board battery backup	32 Kbyte 128 Kbyte SRAM (optional)
No. of system EPROM sockets Data path	4 32-bit
Serial I/O interfaces RS-232/422/RS-485 compatible	3 via the front panel and via the 3-row VME P2 connector and 1 via the optional 5-row VME P2 connector via FORCE hybrids
Ethernet Interface Ethernet SRAM buffer	AM7990 64 Kbyte
Parallel I/O interface (optional)	Via 68230 PI/T 20 lines
Real-Time Clock with on-board battery backup	RTC 72423
SCSI interface	MB87033/34 Single-ended
Floppy disk interface	FDC37C65C[1)]
24-bit timer with 5-bit prescaler 8-bit timer	2 1
VMEbus interface A32,A24,A16:D8,D16,D32,ADO,UAT,RMW A32:D8,D16,D32,ADO,UAT,RMW	 Master Slave
ARBITER SYSCLK driver Mailbox interrupts	Single-level yes 8
FORCE Message Broadcast FMB-FIFO 0 FMB-FIFO 1	 8 byte 1 byte
VMEbus interrupter (level programmable) VMEbus and local interrupt handler Programmable IRQ levels for all sources Total number of IRQ sources	none 1 to 7 yes 42

Table 1: Specifications for the CPU-30 R4 Board (Continued)

RESET and ABORT switches	yes
VMEPROM firmware installed on all board versions	512 Kbyte
Power requirements + 5 V max +12 V max - 12 V max	 typical 2.3A typical 0.4A typical 0.1A
Operating temperature with forced air cooling Storage temperature Relative humidity (non-condensing in %) (With a battery installed, the storage temperature is -40 to +60˚C)	0˚C to +55˚C -40˚C to +85˚C 0 to 95%
Board dimensions	160 x 233 mm
No. of slots used	1

1. The FDC37C65C is pin-to-pin compatible with Industry Standard WD37C65C

1.4 Ordering Information

This page contains a list of the product names and their descriptions.

Table 2: Ordering Information

Product Name	Product Description
CPU-30ZBE Rev. 4	68030/68882 CPU, 25 MHz, 4 Mbyte shared DRAM, 4 Mbyte Flash, SCSI, Ethernet, Floppy disk, 4 serial I/O ports, 32-bit VME-bus interface, VMEPROM firmware. Installation Guide.
CPU-30BE/8 Rev. 4	68030/68882 CPU, 25 MHz, 8 Mbyte shared DRAM, 4 Mbyte Flash, SCSI, Ethernet, Floppy disk, 4 serial I/O ports, 32-bit VME-bus interface, VMEPROM firmware. Installation Guide.
CPU-30BE/16 Rev. 4	68030/68882 CPU, 25 MHz, 16 Mbyte shared DRAM, 4 Mbyte Flash, SCSI, Ethernet, Floppy disk, 4 serial I/O ports, 32-bit VME-bus interface, VMEPROM firmware. Installation Guide.
CPU-30Lite/4 Rev. 4	68030 CPU, 25 MHz, 4 Mbyte shared DRAM, 4 Mbyte Flash, 4 serial ports, 32-bit VMEbus interface, VMEPROM firmware, Installation Guide.
CPU-30Lite/8 Rev. 4	68030 CPU, 25 MHz, 8 Mbyte shared DRAM, 4 Mbyte Flash, 4 serial ports, 32-bit VMEbus interface, VMEPROM firmware, Installation Guide.
CPU-30/TM Rev. 4	Technical Reference Manual Set for the CPU-30 Rev. 4 including a detailed hardware description, a VMEPROM User's Manual and a FGA-002 User's Manual.
CABLE 9-25 SET	Four cable adapters DSub-9 to DSub-25.
IOBP-1	I/O back panel board for VMEbus P2 with flat cable connectors for SCSI, floppy and one serial I/O interface. Extends 4" behind P2.
IOBP-1 Serial Cable	IOBP-1 cable to 3U backpanel for the 4th serial port of the CPU-30.
IOPI-2	I/O back panel board for VMEbus P2 with flat cable connectors for SCSI and floppy interface. Extends 3" behind P2. Documentation included.
FH-002/SET	10 pcs. of Hybrid Modules for the Serial Interface to provide RS-232
FH-003/SET	10 pcs. of Hybrid Modules for the Serial Interface to provide RS-422.
FH-007/SET	10 pcs. of Hybrid Modules for the Serial Interface to provide RS-485.
VxWorks/DEV 68K	VxWorks development package for 68K based products.
VxWorks/BSP CPU-30	VxWorks board support package for CPU-30.
SYS68K/BusNet/VxWorks	BusNet runtime package for VxWorks on 68K VMEbus boards.

1.5 History of Manual Publication

Below is a description of the publication history of this *SYS68K/CPU-30 R4 Technical Reference Manual.*

Table 3: History of Manual

Edition No.	Description	Date
1	First Print	February 1996
2	Rotary switch description in the section “VMEPROM” has been corrected. “Ethernet address” has been changed. NETLOAD, NETSAVE commands have been extended. “Erase flash memory” description has been corrected. Boot flash devices 256K * 8 have been replaced by 128K * 8 devices.	February 1997

2 Installation

2.1 Introduction

This Installation Section provides guidelines for powering up the SYS68K/CPU-30 R4 board. The Installation Section, which you have in your hand now, appears both as Section 2 of the *SYS68K/CPU-30 R4 Technical Reference Manual* and as a stand-alone *Installation Guide*. This stand-alone Installation Guide is delivered by FORCE COMPUTERS with every board. *The SYS68K/CPU-30 R4 Technical Reference Manual* provides a comprehensive hardware and software guide to your board and is intended for those persons who require complete information.

2.1.1 Caution

CAUTION: Read the following safety note before handling the board.

To ensure proper functioning of the product over its usual lifetime, take the following precautions before handling the board.

Electrostatic discharge and incorrect board installation and uninstallation can damage circuits or shorten their lifetime.

- Before installing or uninstalling the board, read this *Installation* section.
- Before installing or uninstalling the board in a VME rack:
  - Check all installed boards for steps that you have to take before turning off the power.
  - Take those steps.
  - Finally turn off the power.
- Before touching integrated circuits, ensure that you are working in an electrostatic free environment.
- Ensure that the board is connected to the VMEbus via both connectors, the P1 and the P2 and that power is available on both.
- When operating the board in areas of strong electro-magnetic radiation, ensure that the board
  - is bolted on the VME rack
  - and shielded by closed housing.

2.1.2 Board Installation

The installation of the board is easy, requiring only a power supply and a VMEbus backplane. The power supply must meet the specifications described in Table 1, “Specifications for the CPU-30 R4 Board,” on page 7. The processor board requires +5 V supply voltage; ±12 V are needed for the RS-232 serial interface and the Ethernet Interface.

For the initial power up, a terminal can be connected to the 9-pin D-Sub microconnector of serial port 1, which is located on the front panel. The serial port provides RS-232 interface signal level.

SEE ALSO: Before powering up check that the default switch settings are correct as outlined in Section 2.3 ‘Default Switch Settings’.

2.2 Location Diagrams of the SYS68K/CPU-30 R4 Board

A location diagram showing the important components on the top side of the CPU-30 R4 appears on the next page. On the page next to it, there is a location diagram showing the bottom side of the CPU-30 R4.

SEE ALSO: Figure 2, “Diagram of the CPU-30 R4 (Bottom View),” on page 14 shows the location of all the switches on the board.

Both of these diagrams only show the components on the board which are of interest to the user.

2.2.1 Before Powering Up

Before powering up, please make sure that the default switch settings are all configured according to Table 4, “Default Switch Settings,” on page 15. Since the board is configured for power up according to these default settings, please check them *before* powering up your SYS68K/CPU-30.

NOTE: The battery backup for SRAM and RTC is disabled with the default switch setting. Stored data will be lost.

Figure 1: Diagram of the CPU-30 R4 (Top View)

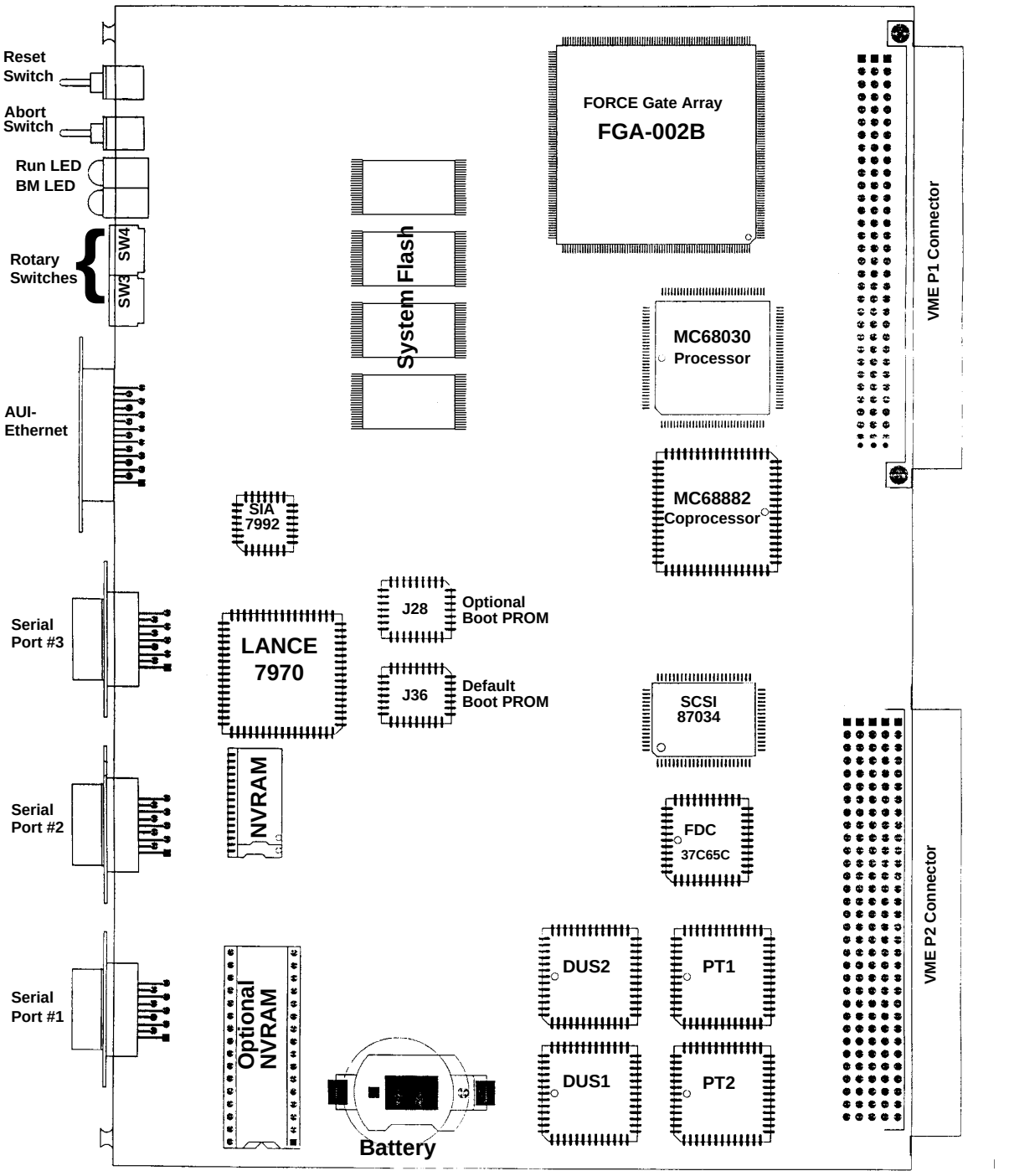
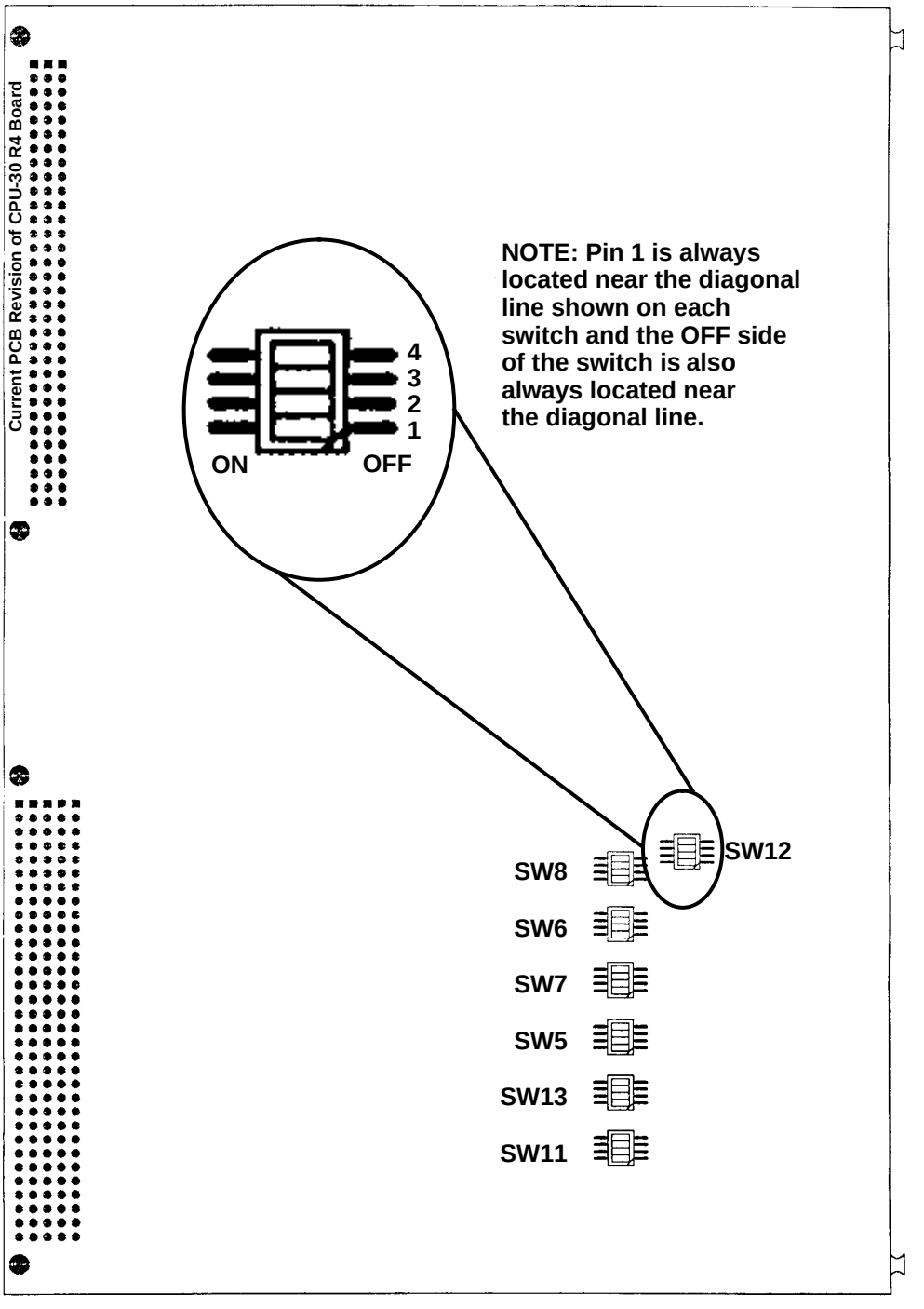


Figure 2: Diagram of the CPU-30 R4 (Bottom View)



2.3 Default Switch Settings

The following table shows the default settings for all the switches on the board. Please make sure you check the default settings before powering up the board.

SEE ALSO: For the position of the switches on your CPU-30 R4 board, please see Figure 2, “Diagram of the CPU-30 R4 (Bottom View),” on page 14.

Table 4: Default Switch Settings

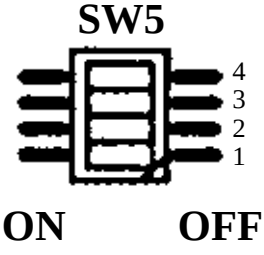
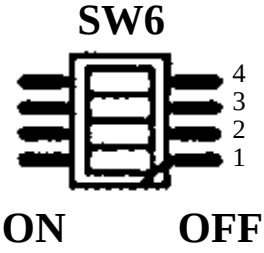
Diagram of Switch with Default Setting	Switches	Default Setting	Function
SWITCH 5			
	SW5-1	OFF	OFF = Boot PROM access to default Boot PROM and optional Boot PROM ON = Boot PROM access to optional Boot PROM only (Access to default Boot PROM is disabled)
	SW5-2	OFF	OFF = Optional Boot PROM Pinout for Flash PROM ON = Optional Boot PROM Pinout for EPROM
	SW5-3	OFF	OFF = Write to Boot PROM enabled ON = Write to Boot PROM disabled
	SW5-4	OFF	OFF = Write to System Flash Memory enabled ON = Write to System Flash Memory disabled
SWITCH 6			
	SW6-1	OFF	SW6-1 OFF = VME Bustimer bit 1=1 SW6-1 ON = VME Bustimer bit 1=0 **BUSTIMER (1:0)** **SW6-1 / SW6-2 / Time** OFF / OFF / 83.53ms OFF / ON / 1.30 ms ON / OFF / 81.6 µs ON / ON / 10.2 µs
	SW6-2	OFF	SW6-2 OFF = VME Bustimer bit 0=1 SW6-2 ON = VME Bustimer bit 0=0
	SW6-3	OFF	SW6-3 OFF = VME BRSEL bit 1=1 SW6-3 ON = VME BRSEL bit 1=0 **SLOT, BRSEL (1:0) : VME BR** SLOT-x detected, 11 : 3 SLOT-x detected, 10 : 2 SLOT-x detected, 01 : 1 SLOT-x detected, 00 : 0 SLOT-1 detected, -- : 3
	SW6-4	OFF	SW6-4 OFF = VME BRSEL bit 0=1 SW6-4 ON = VME BRSEL bit 0=0

Table 4: Default Switch Settings (Continued)

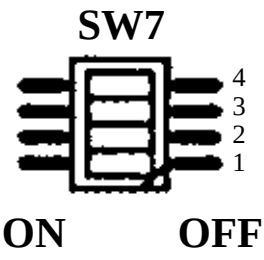
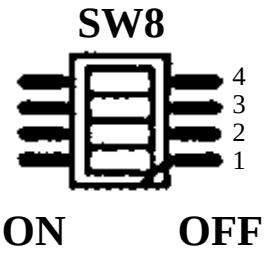
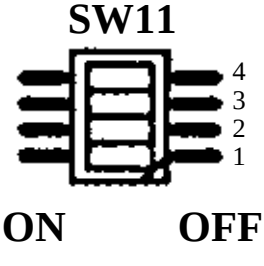
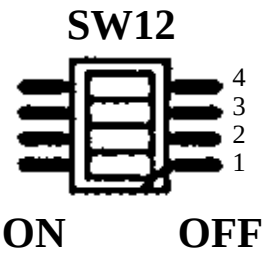
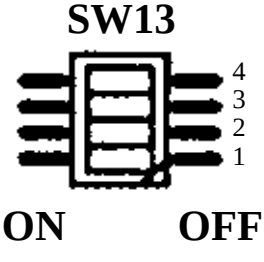
Diagram of Switch with Default Setting	Switches	Default Setting	Function
SWITCH 7			
	SW7-1	OFF	OFF = RESET Switch enabled ON = RESET Switch disabled
	SW7-2	OFF	OFF = ABORT Switch enabled ON = ABORT Switch disabled
	SW7-3	OFF	OFF = SCSI active termination enabled ON = SCSI active termination disabled
	SW7-4	OFF	OFF = additional VME Bustimer enabled if VME slot-1 function detected (otherwise disabled) ON = VME Bustimer disabled
SWITCH 8			
	SW8-1	OFF	OFF = VME slot-1 auto-detection enabled ON = VME slot-1 function disabled
	SW8-2	OFF	OFF = VME_SYSFAIL output enabled ON = VME_SYSFAIL output disabled
	SW8-3	OFF	OFF = VME_SYSRESET output enabled ON = VME_SYSRESET output disabled
	SW8-4	OFF	OFF = VME_SYSRESET input enabled ON = VME_SYSRESET input disabled
SWITCH 11			
	SW11-1	OFF	OFF = Power backup from battery disabled ON = Power backup from battery enabled
	SW11-2	OFF	OFF = Power Backup from VME STBY disabled ON = Power Backup from VME STBY enabled
	SW11-3	OFF	OFF = NVRAM supplied by Power Backup disabled ON = NVRAM supplied by Power Backup enabled
	SW11-4	OFF	OFF = Default NVRAM access only ON = Optional and default NVRAM access

Table 4: Default Switch Settings (Continued)

Diagram of Switch with Default Setting	Switches	Default Setting	Function
SWITCH 12			
	SW12-1	OFF	OFF = Serial channel 1 for RS-232 Hybrid FH-002 ON = Serial channel 1 for RS-422 Hybrid FH-003
	SW12-2	OFF	OFF = Serial channel 2 for RS-232 Hybrid FH-002 ON = Serial channel 2 for RS-422 Hybrid FH-003
	SW12-3	OFF	OFF = Serial channel 3 for RS-232 Hybrid FH-002 ON = Serial channel 3 for RS-422 Hybrid FH-003
	SW12-4	OFF	OFF = Serial channel 4 for RS-232 Hybrid FH-002 ON = Serial channel 4 for RS-422 Hybrid FH-003
SWITCH 13			
	SW13-1	OFF	OFF = Timer IRQ enabled ON = Timer IRQ disabled
	SW13-2	OFF	OFF = Watchdog reset disabled ON = Watchdog reset enabled
	SW13-3	OFF	Reserved (must be OFF)
	SW13-4	OFF	Reserved (must be OFF)

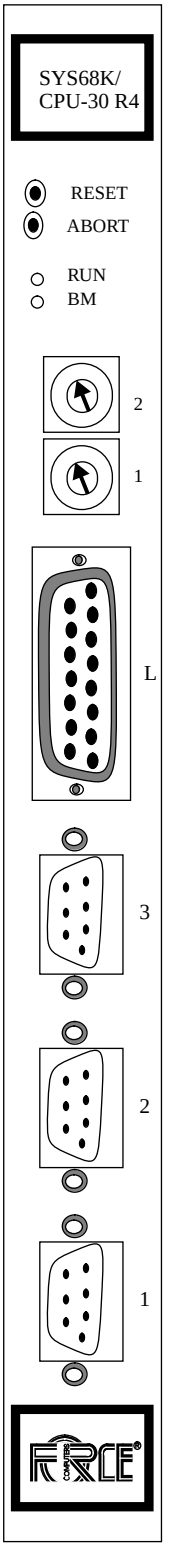
2.4 Front Panel

The table below outlines the layout on the front panel. Additionally, there is a drawing of the front panel on the next page. The front panel devices are briefly described on the pages following the drawing.

Table 5: Front Panel Layout

Device	Function	Name
Switch	Reset	RESET
Switch	Abort	ABORT
LED	RUN/HALT	RUN
LED	VME BM	BM
Rotary Switch	4-bit Input	2
Rotary Switch	4-bit Input	1
15-pin D-Sub connector	AUI-Ethernet Interface	L
9-pin D-Sub connector	Serial Interface	3
9-pin D-Sub connector	Serial Interface	2
9-pin D-Sub connector	Serial Interface	1

Figure 3: Front Panel



RESET and ABORT Keys

Status LEDs

Rotary Switches

15-pin D-Sub Connector

9-pin D-Sub Connector

9-pin D-Sub Connector

9-pin D-Sub Connector

2.4.1 RESET and ABORT Keys

The RESET key generates an on-board reset. The ABORT key generates an IRQ on a programmable level. Both keys can be disabled via the switches described below:

SW7-1	Description
OFF (default)	RESET key enabled
ON	RESET key disabled

SW7-2	Description
OFF (default)	ABORT key enabled
ON	ABORT key disabled

2.4.2 Status LEDs

The CPU-30 R4 includes two front panel LEDs: RUN/HALT LED and BM LED.

The RUN/HALT LED displays the condition that the processor is halted or reset is active and, in this case, the LED turns red. The RUN/HALT LED turns green on normal operation.

The bus master BM LED is used to indicate VMEbus mastership of the CPU-30 R4 and, in this case, the LED turns green.

2.4.3 Voltage Sensor

The voltage sensor generates a power-up reset if the voltage level is below 4.75 V.

2.4.4 Watchdog Timer

This timer can be enabled by software and will generate an NMI followed by a power-up reset, when it is not retriggered.

SW13-2	Description
OFF (default)	Watchdog reset disabled
ON	Watchdog reset enabled

2.4.5 Two Rotary Switches

Two software readable four-bit rotary switches are installed on the board and are accessible via the front panel.

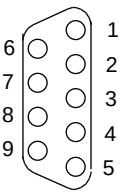
2.5 Serial I/O Channels

The CPU-30 R4 has three serial I/O channels available via 9-pin D-Sub connectors on the front panel. All channels will support RS-232, RS-422 and RS-485 interfaces via the FORCE hybrids FH-00x. The default configuration is RS-232.

The following table shows the pinout of the serial I/O channels for RS-232.

Table 6: 9-pin D-Sub Connector Pinout[1)] (RS-232)

Pin	Signal	Direction	Description
1	DCD	in	Data Channel Detector
2	RxD	in	Receive Data
3	TxD	out	Transmit Data
4	DTR	out	Data Terminal Ready
5	GND	-	Signal Ground
6	DSR	in	Data Set Ready
7	RTS	out	Request to Send
8	CTS	in	Clear to Send
9	GND*	-	Signal Ground



1. Default terminal port setup: 9600 Baud, 8 data bits, 1 stop bit, no parity.

**NOTE: ***With FH-002, this signal is provided by the hybrid being used. The signal DTR is always driven active and the signal DSR is always read active by software. The RS-232 interface on your current CPU-30 revision 4.x board is fully compatible to the RS-232 interface on the earlier CPU-30 revision 3.2 board. However, the default jumper settings prescribed for the earlier board must be used to obtain this functionality.

2.6 AUI-Ethernet

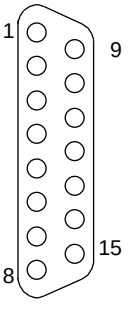
The AUI-Ethernet Interface is available on the front panel via a 15-pin D-Sub connector.

The unique Ethernet address is displayed by the banner when entering the FGA Boot debugger. FGA Boot also provides a utility function to get the CPU board's Ethernet address: "#40 (0x28) Get Ethernet Number".

The following table shows the pinout of the AUI-Ethernet connector.

Table 7: 15-pin AUI-Ethernet Connector

Pin	Description
1	GND
2	Collision Detect+
3	Transmit Data+
4	GND
5	Receive Data+
6	GND
7	Not connected
8	GND
9	Collision Detect-
10	Transmit Data-
11	GND
12	Receive Data-
13	+12V
14	GND
15	Not connected



2.7 SCSI

The MB87033/34 provides an 8-bit single-ended SCSI interface. It is routed to the VMEbus P2 connector.

The termination is switch selectable and "TERMPWR" is supported. The following switches control the SCSI termination.

SW7-3	Description
OFF (default)	SCSI active termination enabled
ON	SCSI active termination disabled

NOTE: TERMPWR is always supplied; if termination power is supplied externally by a source other than the VME connector, the active termination is still maintained, although the VME may not be powered.

2.8 Parallel I/O (Option)

The parallel I/O signals are only available with the optional 5-row VMEbus P2 connector.

2.9 Connector Pinout for VMEbus P2

Table 8: Signal Assignment of the VME P2 Connector

PIN	Row Z (factory option)	Row A	Row C	Row D (factory option)
1	PIT2 A0	SCSI Data 0	FDC RPM (TxD Port 2)	NC
2	GND	SCSI Data 1	FDC HLOAD (FDC EJECT) (RxD Port 2)	NC
3	PIT2 A1	SCSI Data 2	FDC DSEL2	TxD Port 1
4	GND	SCSI Data 3	FDC INDEX	RxD Port 1
5	PIT2 A2	SCSI Data 4	FDC DSEL1	RTS Port 1
6	GND	SCSI Data 5	FDC DSEL2	CTS Port 1
7	PIT2 A3	SCSI Data 6	FDC DSEL1	DTR Port 1
8	GND	SCSI Data 7	FDC MOTOR	DCD Port 1
9	PIT2 A4	SCSI DP	FDC DIREC	GND Port 1
10	GND	GND	FDC STEPX	TxD Port 2
11	PIT2 A5	GND	FDC WDATA	RxD Port 2
12	GND	GND	FDC WGATE	RTS Port 2
13	PIT2 A6	TERMPWR	FDC TRK00	CTS Port 2
14	GND	GND	FDC WPROT	DTR Port 2
15	PIT2 A7	GND	FDC RDATA	DCD Port 2
16	GND	SCSI ATN	FDC SDSEL	GND Port 2
17	PIT2 H1	GND	FDC RDY	TxD Port 3
18	GND	SCSI BSY	(RTS Port 2)	RxD Port 3
19	PIT2 H2	SCSI ACK	GND	RTS Port 3
20	GND	SCSI RST	GND	CTS Port 3
21	PIT2 H3	SCSI MSG	(CTS Port 2)	DTR Port 3
22	GND	SCSI SEL	GND	DCD Port 3
23	PIT2 H4	SCSI CD	GND	GND Port 3
24	GND	SCSI REQ	(TxD Port 3)	DSR Port 1
25	PIT1 H1	SCSI IO	(RxD Port 3)	DSR Port 2
26	GND	(RTS Port 1)	(RTS Port 3)	DSR Port 3
27	PIT1 H2	GND	(CTS Port 3)	PIT1 C0
28	GND	(CTS Port 1)	(TxD Port 1)	PIT1 C1
29	PIT1 H3	DSR Port 4	DCD Port 4 (RxD Port 1)	PIT1 C4
30	GND	RTS Port 4	RxD Port 4	PIT1 C7
31	PIT1 H4	CTS Port 4	TxD Port 4	NC
32	GND	GND Port 4	DTR Port 4	NC

NOTE: The signals marked in parenthesis are only available with the use of FH-002 hybrids, which are available at FORCE COMPUTERS.

2.10 Introduction to VMEPROM Firmware

The VMEPROM firmware is a full multitasking multiuser real-time system. It is stored in the on-board System Flash Memory and provides the following functionality:

- Configuration of the board
- Starting an application
- Application hooks
- Shell with over 80 commands
- Programming of Boot Flash devices.

2.10.1 Booting up VMEPROM

To start VMEPROM, the rotary switches must both be set to 'F':

Table 9: Rotary Switches

MODE 1	F
MODE 2	F

The different functions of the rotary switches are described in detail in the VMEPROM section of the *SYS68K/CPU-30 R4 Technical Reference Manual.*

Correct Operation

To test the correct operation of the CPU board, the following command must be typed in:

```
# SELFTEST <CR>
```

The selftest command tests some I/O devices, the main memory and the system timer tick interrupt. Depending on the size of the main memory, it may last a different amount of time (count about one minute per megabyte).
After all tests are done, the following message will appear on the terminal screen:

```
VMEPROM Hardware Selftest
-------------------------
I/O test ........ passed
Memory test ..... passed
Clock test ...... passed
```

2.11 The SYS68K/IOBP-1

FORCE COMPUTERS offers an IOBP-1 back panel for easy connection of I/O signals through the VMEbus P2 connector. This board can be plugged into the VMEbus P2 connector of a VMEbus board which carries the SCSI, FDC, and serial I/O signals on the VMEbus P2. It contains a SCSIbus connector (P2), a floppy disk interface connector (P3), and a serial I/O connector (P5). All VMEbus P2 connector row A and C pins are routed to the 64-pin male connector (P4). The pinout of these connectors is shown in the following table.

Table 10: SYS68K/IOBP-1 Pin Assignment

PIN No. IOBP-1 P1	PIN No. VMEbus P2	Row A Signal Mnemonic		Row B Signal Mnemonic	Row C Signal Mnemonic	
32	1	DB 0	SCSI	-		
31	2	DB 1	SCSI	GND		
30	3	DB 2	SCSI	-	Drive Select 4 (2)	FDC
29	4	DB 3	SCSI	-	Index	FDC
28	5	DB 4	SCSI	-	Drive Select 1	FDC
27	6	DB 5	SCSI	-	Drive Select 2	FDC
26	7	DB 6	SCSI	-	Drive Select 3 (1)	FDC
25	8	DB 7	SCSI	-	Motor On	FDC
24	9	DB P	SCSI	-	Direction In	FDC
23	10	GND		-	Step	FDC
22	11	GND		-	Write Data	FDC
21	12	GND		GND	Write Gate	FDC
20	13	TERMPWR	SCSI	-	Track 000	FDC
19	14	GND		-	Write Protect	FDC
18	15	GND		-	Read Data	FDC
17	16	ATN	SCSI	-	Side Select	FDC
16	17	GND		-	FDC READY	FDC
15	18	BSY	SCSI	-		
14	19	ACK	SCSI	-	GND	
13	20	RST	SCSI	-	GND	
12	21	MSG	SCSI	-		
11	22	SEL	SCSI	GND	GND	
10	23	C/D	SCSI	-	GND	
9	24	REQ	SCSI	-		
8	25	I/O	SCSI	-		
7	26			-		
6	27	GND		-	Reserved	
5	28			-	Reserved	
4	29	DSR	SER	-	DCD	SER

Table 10: SYS68K/IOBP-1 Pin Assignment (Continued)

PIN No. IOBP-1 P1	PIN No. VMEbus P2	Row A Signal Mnemonic		Row B Signal Mnemonic	Row C Signal Mnemonic	
3	30	RTS	SER	-	RXD	SER
2	31	CTS	SER	GND	TXD	SER
1	32	GND	SER	-	DTR	SER

3 Hardware Description

This CPU board is a high performance single-board computer based on the 68030 microprocessor and the VMEbus. The CPU board also includes an enhanced Floating Point Coprocessor 68882. The board design utilizes all of the features of the powerful FORCE Gate Array FGA-002.

Besides the CPU-30 R4, there will be a CPU-30Lite R4 without a coprocessor, a SCSI, an Ethernet, and a floppy disk interface.

SEE ALSO: Please refer to Table 2, “Ordering Information,” on page 9 for more detailed information.

The CPU-30 R4 provides an A32/D32 VMEbus interface including DMA, up to 32-Mbyte shared DRAM on-board, up to 8-Mbyte System Flash, an Ethernet Interface, a single-ended SCSI interface, a Floppy interface, four RS-232 serial I/O channels, up to 256-Kbyte SRAM and a Real-Time Clock, both with on-board battery backup.

The shared DRAM is accessible from the 68030 CPU, the FGA-002 DMA controller, and also from other VMEbus masters.

The CPU-30 R4 has an Ethernet port as well as three serial ports available on the front panel permitting a console port, download and data communication channels.

One serial port, as well as the SCSI interface and the Floppy interface are available via the 3-row VMEbus P2 connector.

A 20-bit parallel interface and the three serial ports from the front panel are available via the optional 5-row VMEbus P2 connector.

The CPU-30 R4 is fully software compatible to the CPU-30 R3 with the exception of the floppy controller FDC37C65C, which has replaced the WD1772.

3.1 SYS68K/CPU-30 R4 Memory Map

Table 11: SYS68K/CPU-30 R4 Memory Map

Address range	Device	VMEbus accessible	Cache	Burst	Access width
0000.0000_{16} …$00xF.FFFF_{16}$	DRAM: contributing to shared RAM, address range depends on memory capacity	Y	Y	Y	32/16/8
$00xx.0000_{16}$ …$F9FF.FFFF_{16}$	VME A32 extended address space (consecutive to DRAM)	n/a	N	N	32/16/8
$FB00.0000_{16}$ …$FBFE.FFFF_{16}$	VME A24 standard address space	n/a	N	N	32/16/8
$FBFF.0000_{16}$ …$FBFF.FFFF_{16}$	VME A16 short address space	n/a	N	N	32/16/8
$FC00.0000_{16}$ …$FCFE.FFFF_{16}$	VME A24 standard address space	n/a	N	N	16/8
$FCFF.0000_{16}$ …$FCFF.FFFF_{16}$	VME A16 short address space	n/a	N	N	16/8
$FD00.0000_{16}$ …$FEEF.FFFF_{16}$	reserved	n/a	n/a	n/a	n/a
$FEF0.0000_{16}$ …$FEF7.FFFF_{16}$???	LAN - RAM	N	N	N	32/16/8
$FEF80.0000_{16}$ …$FEFF.FFFF_{16}$???	LAN - Controller	N	N	N	16
$FF00.0000_{16}$ …$FF7F.FFFF_{16}$	The System PROM Area: address range depends on system flash capacity	N	N	N	32/16/8 RO 32 WO
$FF80.0000_{16}$ $FF80.0BFF_{16}$	reserved	n/a	n/a	n/a	n/a
$FF80.0C00_{16}$ …$FF80.0DFF_{16}$	PIT1	N	N	N	8
$FF80.0E00_{16}$ …$FF80.0FFF_{16}$	PIT2	N	N	N	8
$FF80.1000_{16}$ $FF80.1FFF_{16}$	reserved	n/a	n/a	n/a	n/a

Table 11: SYS68K/CPU-30 R4 Memory Map (Continued)

Address range	Device	VMEbus accessible	Cache	Burst	Access width
$FF80.2000_{16}$ …$FF80.21FF_{16}$	DUSCC1	N	N	N	8
$FF80.2200_{16}$ …$FF80.23FF_{16}$	DUSCC2	N	N	N	8
$FF80.2400_{16}$ …$FF80.2FFF_{16}$	reserved	n/a	n/a	n/a	n/a
$FF80.3000_{16}$ …$FF80.31FF_{16}$	Real-Time Clock – RTC 72423	N	N	N	8
$FF80.3200_{16}$ …$FF80.33FF_{16}$	reserved	n/a	n/a	n/a	n/a
$FF80.3400_{16}$ …$FF80.35FF_{16}$	SCSI-Controller	N	N	N	8
$FF80.3600_{16}$ …$FF80.37FF_{16}$	reserved	n/a	n/a	n/a	n/a
$FF80.3800_{16}$ …$FF80.397F_{16}$	The Floppy Disk Controller	N	N	N	8
$FF80.397F_{16}$ …$FF80.39FF_{16}$	Slot-1 status register (RO)	N	N	N	8 ro
$FFC0.0000_{16}$ …$FFCF.FFFF_{16}$	Local SRAM	N	Y	N	32/16/8
$FFD0.0000_{16}$ …$FFDF.FFFF_{16}$	FGA-002 Gate Array internal reg.	n/a	N	N	32/16/8
$FFE0.0000_{16}$ …$FFEF.FFFF_{16}$	Boot PROM	N	N	N	32/16/8
$FFF0.0000_{16}$ …$FFFF.FFFF_{16}$	reserved	n/a	n/a	n/a	n/a

3.2 The CPU 68030 Processor

3.2.1 Hardware Interface of the 68030

The 68030 uses a nonmultiplexed address and data bus. Asynchronous signals allow easy interfacing to the outside world; synchronous signals perform fast interaction.

The CPU drives the address signals (A0-A31), the size signals (SIZ0, SIZ1) and the function code signals (FC0-FC2) on every cycle, independent of a cache hit or miss. These signals are used to decode the memory map of the CPU board.

The hardware on the CPU board is notified by the address and data strobe signals that the current cycle is not a cache cycle, and that the decoding outputs are strobed to be valid.

The 32 data lines (D0-D31) are also driven from the processor on write cycles and sensed on read cycles.

The size of the data transfer is defined by the SIZE + A0 - A1 output signals (always driven from the CPU). During asynchronous cycles the data bus width is determined by the Data Size Acknowledge Input signals (DSACK0, DSACK1). Synchronous cycles acknowledged by the Synchronous Termination Input signal (STERM) acknowledge the indicated data width during writes, whereas during reads a 4-byte width is always acknowledged.

If a bus error occurs (BERR sensed from the CPU), exception handling starts because the current cycle has been aborted (illegal transfer or incorrect data).

On local bus operation, a bus error will be generated if a device does not respond correctly.

VMEbus transfers may also be aborted via a BERR.

3.2.2 The Instruction Set

For the 68030 instruction set and further information relative to programming, please refer to the *68030 User's Manual*.

3.2.3 Vector Table of the 68030

This table lists all vectors defined and used by the 68030 CPU.

Table 12: Exception Vector Assignments

Vector Number(s)	Vector Offset (Hex)	Assignment
0	000_{16}	Reset Initial Interrupt Stack Pointer
1	004_{16}	Reset Initial Program Counter
2	008_{16}	Access Fault (Bus Error)
3	$00C_{16}$	Address Error
4	010_{16}	Illegal Instruction
5	014_{16}	Integer Divide by Zero
6	018_{16}	CHK, CHK2 Instruction
7	$01C_{16}$	FTRAPcc, TRAPcc, TRAPV Instructions
8	020_{16}	Privilege Violation
9	024_{16}	Trace
10	028_{16}	Line 1010 Emulator (Unimplemented A-Line Opcode)
11	$02C_{16}$	Line 1111 Emulator (Unimplemented F-Line Opcode)
12	030_{16}	Unassigned, Reserved
13	034_{16}	Coprocessor Protocol Violation
14	038_{16}	Format Error
15	$03C_{16}$	Uninitialized Interrupt
16-23	040_{16}-$05C_{16}$	Unassigned, Reserved
24	060_{16}	Spurious Interrupt
25	064_{16}	Level 1 Interrupt Autovector
26	068_{16}	Level 2 Interrupt Autovector
27	$06C_{16}$	Level 3 Interrupt Autovector
28	070_{16}	Level 4 Interrupt Autovector
29	074_{16}	Level 5 Interrupt Autovector
30	078_{16}	Level 6 Interrupt Autovector
31	$07C_{16}$	Level 7 Interrupt Autovector
32-47	080_{16}-$0BC_{16}$	TRAP #0-15 Instruction Vectors
48	$0C0_{16}$	FPCP Branch or Set on Unordered Condition
49	$0C4_{16}$	FPCP Inexact Result
50	$0C8_{16}$	FPCP Divide by Zero
51	OCC_{16}	FPCP Underflow
52	ODO_{16}	FPCP Operand Error
53	$OD4_{16}$	FPCP Overflow
54	$0D8_{16}$	FPCP Signalling NAN
55	ODC_{16}	FPCP Unimplemented Data Type
56	$0E0_{16}$	MMU Configuration Error
57	$0E4_{16}$	Defined for 68852, not used by 68030
58	$0E8_{16}$	Defined for 68852, not used by 68030
59-63	$0EC_{16}$-$0FC_{16}$	Unassigned, Reserved
64-255	100_{16}-$3FC_{16}$	User Defined Vectors (192)

3.3 The Floating Point Coprocessor (FPCP)

The CPU board contains a Floating Point Coprocessor (FPCP 68882).

3.3.1 Features of the 68882

- 8 floating point data registers supporting 80-bit extended precision of real data (64-bit mantissa, 15-bit exponent, and one sign bit)
- Three registers for control, status and instruction address
- 67-bit arithmetic unit
- 67-bit barrel shifter
- 46 instructions with 35 arithmetic operations
- IEEE 754 compatible, including all requirements and suggestions
- Full set of trigonometric and transcendental functions
- Seven data types:
  - Byte Integer
  - Word Integer
  - Long Word Integer
  - Single Precision Real
  - Double Precision Real
  - Extended Precision Real
  - Packed Decimal Strings
- 22 constants available in the on-chip ROM, including Pi, e, and powers of 10
- Virtual memory/machine operations
- Efficient mechanism for procedure calls, context switches and interrupt handling

3.3.2 Interfacing to the 68882

The 68882 is a non-DMA type coprocessor which uses a subset of the general purpose coprocessor interface supported by the 68030.

Features of the interface implemented in the 68882 are as follows:

- Main processor and 68882 communicate via standard bus cycles.
- Main processor and 68882 communication is not dependent upon instruction sets or internal details of individual devices (e.g. instruction pipes or caches, addressing modes).
- The main processor and 68882 may operate at different clock speeds.
- 68882 instructions utilize all addressing modes provided by the main processor; all effective addresses are calculated by the main processor at the request of the coprocessor.
- All data transfers are performed by the main processor at the request of the 68882; thus memory management, bus errors, address errors, and bus arbitration function as if the 68882 instructions are executed by the main processor.
- Overlapped (concurrent) instruction execution enhances throughput while maintaining the programmer's model of sequential instruction execution.
- Coprocessor detection of exceptions which require a trap to be taken are serviced by the main processor at the request of the 68882; thus exception processing functions as if the 68882 instructions were executed by the main processor.
- Support of virtual memory/virtual machine systems is provided via the FSAVE and FRESTORE instructions.
- Up to eight coprocessors may reside in a system simultaneously; multiple coprocessors of the same type are also allowed.
- Systems may use software emulation of the 68882 without reassembling or relinking user software.

For further details, please refer to the *User's Manual of the 68881/68882*.

3.3.3 Addressing the 68882

The 68882 is addressed via the function codes of the 68030 and a part of the address bus. This is done automatically within the opcodes generated by most 68030/68882 floating-point compilers and assemblers.

The following table lists the conditions for addressing the 68882.

Signal	Value	Description
FC0 FC1 FC2	1 1 1	CPU Space Cycles
A13 A14 A15	1 0 0	Coprocessor ID = 1
A16 A17 A18 A19	0 1 0 0	Coprocessor Access Cycle

3.3.4 FPCP ID Number

All instructions for the FPCP must include the coprocessor ID (001). Please note that the VMEPROM Assembler supports this function by default.

3.3.5 Detection of the 68882

The SENSE pin of the FPCP is connected to PI/T #1. This allows automatic detection whether or not the 68882 is installed.

CAUTION: PI/T #1 pin PC6/PIACK must be programmed as an input.

PC6	Function
0	FPCP installed
1	FPCP not installed

3.3.6 Summary of the 68882

Allowed Function Codes	1 1 1 (CPU Space Cycle)
Coprocessor ID	0 0 1
Usable Data Bits	D0 - D31
Supported Transfer Types	Byte
	Word
	Long Word

3.4 The Local Bus

3.4.1 The FGA-002 Gate Array

The CPU board also contains the FGA-002 Gate Array with 24,000 gates and 304 pins.

The FGA-002 Gate Array controls the local bus and builds the interface to the VMEbus. It also includes a DMA controller, complete interrupt management, a message broadcast interface (FMB), timer functions, and mailbox locations.

The gate array monitors the local bus. This in turn signifies that if any local device is to be accessed, the gate array takes charge of all control signals in addition to used address and data signals.

The FGA-002 Gate Array serves as a manager for the VMEbus. All VMEbus address and data lines are connected to the gate array through the buffers. Additional functions such as the VMEbus interrupt handler and arbiter are also installed on the FGA-002 Gate Array.

The start address of the FGA-002 Gate Array registers is $FFD0.0000_{16}$. All registers of the gate array and associated functions are described in detail in the *FGA-002 Gate Array User's Manual*.

3.4.2 Shared DRAM

The CPU board contains a Shared dynamic RAM area with a capacity of 4, 8, 16 or 32 Mbytes. The Shared RAM area is optimized for fast accesses from the 68030 CPU and the DMA controller in the FGA-002 Gate Array. The Shared RAM is also accessible by other VMEbus masters.

The Shared RAM area is arranged in 36-bit wide memory banks. There may be one or two of these banks on the CPU board, depending on the overall memory capacity delivered. Each 36-bit wide bank is separated into 32 data bits and 4 parity bits. A parity bit checks every eight consecutive data bits (byte parity). Advanced on-board memory control logic routes data to and from the on-board 68030 CPU, the DMA controller, and the VMEbus interface.

For every read cycle, regardless of size (byte, word, long-word or cache line) and regardless of master (68030, DMA or VMEbus), all 32 bits of data and 4 bits of parity are read from the Shared RAM array. The 32 data and 4 parity bits are stored in transceivers.

Parity is regenerated in FGA-002 and compared to the parity bits read from memory. If a mismatch is found on an accessed byte, an error will be flagged. A synchronous termination signal (STERM) is asserted, and the cycle completes.

Write cycles are handled differently. In the case of a long-word access aligned to a 4-byte boundary, the DRAM can be written immediately. The parity info generated by FGA-002 will be written additionally to the DRAM. A synchronous termination signal (STERM) is asserted, and the cycle completed.

For all other write cycles (byte, word, long-word unaligned), the momentary valid parity info stored in DRAM must be read. Then the write to RAM Memory will be performed. Therefore, only the necessary data will be written, the remaining data already stored in DRAM will stay unmodified. Additionally, the new parity info generated by FGA-002 will be merged with the read parity info from DRAM and finally all four parity bits are written to DRAM. The synchronous termination signal (STERM) will be generated to complete the cycle.

All write cycles are terminated before they are fully processed to allow the master writing to DRAM to continue its operations (write posting).

3.4.2.1 Bank Selection of DRAM

The bank selection depends on memory size. The Dual-Banks architecture implements an interleaved organized DRAM (four consecutive bytes located in bank A, the next four consecutive bytes located in bank B, ...). The Single-Bank architecture implements a non-interleaved organized DRAM.

Table 13: Used Device Types for the Shared Memory

DRAM Device	Device Capacity	Total Capacity	Bank	Supported Product
1M * 4 FPM[1)]	9 * 1 Mbit * 4	4 Mbyte	1	CPU-30ZBE R4
1M * 4 FPM	18 * 1 Mbit * 4	8 Mbyte	2	CPU-30BE/8 R4
4M * 4 FPM	9 * 4 Mbit * 4	16 Mbyte	1	CPU-30BE/16 R4
4M * 4 FPM	18 * 4 Mbit * 4	32 Mbyte	2	factory option
1M * 4 FPM	9 * 1 Mbit * 4	4 Mbyte	1	CPU-30Lite/4 R4
1M * 4 FPM	18 * 1 Mbit * 4	8 Mbyte	2	CPU-30Lite/8 R4

1. FPM: Fast Page Mode

Shared RAM byte parity generation and check work for both local and VMEbus accesses. If a parity error is detected during a VMEbus slave read access, the CPU board drives BERR, informing the VMEbus master about the parity error. On all local accesses, a normal STERM will be generated, plus an interrupt on a software programmable level. The access address is stored inside the FGA-002 Gate Array allowing easy software controlled detection of the cycle which caused the parity error.

The Shared RAM is accessed from the VMEbus via FGA-002. The start and end access addresses are programmable in 4 Kbyte steps. The

defined memory range can be write protected in coordination with the VMEbus Address Modifier codes. For example, in privileged mode the memory could be read and written, while in non-privileged mode the memory could only be read, or a non-privileged access could be prohibited altogether.

When the gate array detects a VMEbus access cycle to the programmed address range of the Shared RAM, it requests local bus mastership from the CPU. After the CPU has granted local bus mastership to the FGA-002, the VMEbus access cycle is executed and all data is latched (read cycles), or stored to RAM (write cycles). The read and write cycle is then terminated and the FGA-002 immediately releases local bus mastership back to the CPU. Simultaneously, it completes the fully asynchronous VMEbus access cycle. The early completion of the memory read or write cycle allows the CPU to continue processing while the FGA-002 independently manages the VMEbus transaction overhead.

A programmable bit within the FGA-002 may be used to disable the early bus release option. With early release disabled, the FGA-002 retains local bus mastership until the VMEbus cycle is finished. This guarantees that no other local bus master (CPU or DMA controller) will access the Shared RAM until the VMEbus cycle is complete. In the case of a read-modify-write (RMW) cycle by another VMEbus master to the Shared RAM, the FGA-002 will perform both transactions (a read followed by a write) without releasing the local bus, thus guaranteeing that the cycle is indivisible.

In short, the early release option allows the CPU access to the Shared RAM sooner, but sacrifices the guaranteed indivisibility of VMEbus RMW cycles. Because the 68030 CPU includes an on-chip cache memory, this may not affect CPU performance at all.

3.4.3 Board Type with Memory Capacity

The following table lists the CPU board type with the memory capacity of the Shared RAM.

Board Type	CPU Frequency	DRAM Capacity
CPU-30ZBE R4	25 MHz	4 Mbytes
CPU-30BE/8 R4	25 MHz	8 Mbytes
CPU-30BE/16 R4	25 MHz	16 Mbytes
factory option	25 MHz	32 Mbytes
CPU-30Lite/4 R4	25 MHz	4 Mbytes
CPU-30Lite/8 R4	25 MHz	8 Mbytes

3.4.4 Reading the Shared RAM Capacity

The amount of Shared RAM may be read via three input pins from Port B of PI/T #2. The table below summarizes the encoding of these three bits.

B2	B1	B0	Memory Capacity
0	0	0	32 Mbytes
0	0	1	16 Mbytes
0	1	0	8 Mbytes
0	1	1	4 Mbytes
1	-	-	Reserved

SEE ALSO: Please refer to Section 3.10.13, ‘I/O Configuration of PI/T #2,’ on page 74 for more detailed information.

3.4.5 Shared RAM Addressing

The access address of the Shared RAM is programmable within the FGA-002 Gate Array. The default address range of the 4 Mbyte DRAM array is from 0000.0000_{16} to $003F.FFFF_{16}$. The default address range of the 32 Mbyte DRAM array is 0000.0000_{16} to $01FF.FFFF_{16}$. It is possible to program nearly any address range desired in the FGA-002.

Start Address	End Address	Memory Capacity
0000.0000_{16}	$01FF.FFFF_{16}$	32 Mbytes
0000.0000_{16}	$00FF.FFFF_{16}$	16 Mbytes
0000.0000_{16}	$007F.FFFF_{16}$	8 Mbytes
0000.0000_{16}	$003F.FFFF_{16}$	4 Mbytes

The access address of the Shared RAM from the VMEbus is also programmable via FGA-002. That is the address range that other VMEbus masters must use in order to access the Shared RAM on the CPU board. This is not necessarily the same address range used by the CPU for local accesses.

SEE ALSO: Please refer to Section 3.18, ‘VMEbus Slave Interface,’ on page 101 for more information.

3.4.6 Shared RAM Performance

The memory interface logic controlling the Shared RAM array is optimized for fast accesses from the 68030 CPU, providing the highest possible performance. Because the 68030 CPU contains an on-chip data and instruction cache, many CPU accesses are cache line "burst fills". These burst transactions attempt to read 16 consecutive bytes into the 68030, using four 4-byte cycles.

The first read cycle of such a burst usually requires 5 CPU clock cycles (200 nanoseconds at 25 MHz). Due to the optimized design of the memory control logic, each subsequent cycle only requires 1 CPU clock cycle (40 nanoseconds) to complete. This is commonly called a "5-1-1-1" burst transfer. Overall, the total cache line "burst fill" operation requires 8 clock cycles to transfer 16 bytes, providing a memory bandwidth of over 50 Mbytes/second.

Not all CPU accesses are burst transfers. Single read and write transactions are also supported at the fastest possible speed. A single read or write access (1, 2, or 4 bytes) requires 5 CPU clock cycles. Distributed asynchronous refresh is provided every 14 microseconds and an access during a pending refresh cycle may be delayed by a maximum of five additional clock cycles.

3.5 The System PROM Area

The first two read cycles after reset of the microprocessor are operand fetches of the Initial Interrupt Stack Pointer (ISP) and the Initial Program Counter (IPC). These operands are always fetched from addresses 0000.0000_{16} and 0000.0004_{16}, respectively.

3.5.1 Initialization

Special control logic in FGA-002 maps the Boot PROM (not the System Flash Memory) down to this address to allow the 68030 to boot from a single byte-wide PROM. This facilitates debugging and low-level program development. However, when the initialization routines in the Boot PROM are completed, control is transferred to the System Flash Memory in such a way that the 68030 appears to have been booted from the System Flash Memory, not the Boot PROM. For this reason, the System Flash Memory must also have the ISP and IPC loaded at address 0000.0000_{16} and 0000.0004_{16}, respectively.

0000.0000_{16} in System Flash Memory: Initial Interrupt Stack Pointer

0000.0004_{16} in System Flash Memory: Initial Program Counter

3.5.2 Memory Organization of the System PROM Area

The data path of the System Flash Memory is 32-bit wide, separated into 4-byte paths. Each byte path is connected to one Flash Memory device.

3.5.3 Read/Write to the System Flash Memory

Read cycles with any port size are allowed. Write cycles are flagged by the FGA-002 Gate Array with BERR. A programmable bit within the FGA-002 may be used to enable write operation to the System Flash Memory. In this case the FGA-002 will respond with an asynchronous data acknowledge (DSACK0, DSACK1). The write takes affect to the System Flash Memory depending on switch SW5-4.

SW5-4	Description
OFF (default)	Write to System Flash Memory enabled
ON	Write to System Flash Memory disabled

The status of SW5-4 is connected to PI/T #2.

Pin PC4 on the PI/T #2 interface signal protects the write to the System Flash Memory and must be programmed as an input.

The encoding of these bits is shown in the table below.

PC4	Function
0	Write to System Flash Memory unprotected
1	Write to System Flash Memory protected

CAUTION: Writes to the System Flash Memory must always be performed with a 4-byte wide port size and must be aligned on 4-byte boundaries.

3.5.4 Programming the System Flash Memory

For correct programming, a communication sequence to the Flash devices must be used which is defined by the manufacturer of the Flash device.

SEE ALSO: For further details, please refer to the data sheets for Flash devices in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

This communication sequence has to be performed on every byte path.

Besides the communication sequence, a programming voltage Vpp of 12V must be applied to the Flash device. The Vpp is generated by the CPU-30 R4 and is controlled by PI/T #2.

CAUTION: PI/T #2 pin PC6 must be programmed as an output.

PC6	Function
0	Vpp on
1	Vpp off

The Vpp generator is shared between the System Flash Memory and the default Boot PROM socket.

3.5.5 Device Types for the System Flash Memory

The following device types or equivalent are used by the System Flash Memory:

Table 14: Device Types used for System Flash Memory

Device	Device Capacity	Total Capacity	Device Speed	Default Configuration
28F008SA	1M * 8	4 Mbytes	120 ns	x
29F016	2M * 8	8 Mbytes	120 ns	

The default configuration using 28F008SA devices is provided for programming VMEPROM.

3.5.6 Address Map of the System PROM Area

The start address of the System PROM Flash Area is mapped via the FGA-002 Gate Array and cannot be changed. The size of this memory area depends on the memory capacity of the devices used. The following table lists the address map for the usable device types.

Table 15: Address Map of the PROM Area

Start Address	End Address	Used Device	Total Capacity	Default Configuration
$FF00.0000_{16}$	$FF3F.FFFF_{16}$	28F008SA	4 Mbytes	x
$FF00.0000_{16}$	$FF7F.FFFF_{16}$	29F016	8 Mbytes	

3.5.7 Summary of the PROM Area

Not Allowed Access with Function Code	111
Usable Data Bits	D00 - D31
Supported Port Size (read)	Long, Word, Byte
Supported Port Size (write)	Long (aligned!)
No. of Devices	4
Capacity	4 Mbytes
Default Configuration for	28F008A Devices
Default Access Time	120 ns
Access Address Range	$FF00.0000_{16}$ - $FF3F.FFFF_{16}$

3.6 The Boot PROM

The CPU board contains one or two 32-pin PROMs which are used to boot up the processor and initialize register contents of the FGA-002 Gate Array. This program finishes in such a manner that the 68030 microprocessor appears to have booted from the System Flash Memory. The Boot PROM devices are located in 32-pin PLCC sockets at location J28 or J36.

Because the 68030 will unconditionally boot from the Boot PROM memory after every power up or reset, there must always be a working Boot PROM device installed in the CPU board.

During the bootup procedure, the FGA-002 will map all addresses to the Boot PROM memory with the exception of FGA-002 internal registers. After bootup the Boot PROM will be accessible at address $FFE0.0000_{16}$. The start address of the Boot PROM is fixed and cannot be changed.

3.6.1 The Boot PROM Sockets

The Boot PROM area is located in two 32-pin PLCC sockets. One socket, the default Boot PROM socket, allows the usage of 12V programmable Flash devices. The second socket, the optional Boot PROM socket, allows the usage of 5V programmable Flash devices. Alternatively, EPROM devices (e.g. OTP) can be used.

SEE ALSO: For information on which devices may be used, please see the tables in Section 3.6.2, ‘The Boot PROM Address Map,’ on page 47.

3.6.1.1 Boot PROM Selection

The Boot PROM socket selection is controlled by switch SW5-1.

SW5-1	Description
OFF (default)	Default Boot socket J36 enabled with start address at $FFE0.0000_{16}$, Optional Boot socket J28 enabled with start address at $FFE8.0000_{16}$
ON	Default Boot socket J36 disabled, Optional Boot socket J28 enabled with start address at $FFE0.0000_{16}$

3.6.1.2 Device Type Selection for Optional Boot PROM (Socket J28)

The optional Boot PROM socket supports the use of

- Programmable 5V Flash devices
- EPROM (OTP)
- EEPROM

This is controlled by switch SW5-2.

SW5-2	Description
OFF (default)	Optional Boot PROM (Socket J28) supports 5V Flash device (programmable) and 12V Flash device (non-programmable)
ON	Optional Boot PROM (Socket J28) supports EPROM (OTP) or EEPROM (writeable)

3.6.1.3 Programming the Boot PROM Devices

The programming of Boot PROM devices is supported for the default Boot PROM (socket J36) for 12V Flash devices and for the optional Boot PROM socket J28 for 5V Flash or EEPROM devices.

For correct programming, a communication sequence to the Boot PROM devices must be used which is defined by the manufacturer of the device.

SEE ALSO: For further details, please refer to the data sheets in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

The programming to the Boot PROM devices is dependent on switch SW5-3.

SW5-3	Description
OFF (default)	Write to Boot PROM devices enabled
ON	Write to Boot PROM devices disabled

The status of switch SW5-3 is connected to the PI/T #2.

CAUTION: PI/T #2 pin PC2 must be programmed as an input.

PC2	Function
0	Write to Boot PROM unprotected
1	Write to Boot PROM protected

3.6.1.4 Programming Flash Devices

Besides the communication sequence, a programming voltage Vpp of 12V must be applied to the default Boot PROM (socket J36). Vpp is generated by the CPU-30 R4 and is controlled by PI/T #2.

PC6	Function
0	Programming voltage enabled (Vpp = ON)
1	Programming voltage disabled (Vpp = OFF)

The Vpp generator is shared between the System Flash Memory and the default Boot PROM socket.

CAUTION: PI/T #2 pin PC6 must be programmed as an output.

3.6.2 The Boot PROM Address Map

3.6.2.1 Address Map of the Default Boot PROM Socket J36

Boot PROM access to default Boot PROM and optional Boot PROM (Boot PROM selection switch SW5-1 in position OFF)

Used Device	Device Type	Start Address	End Address	Total Capacity	Default Configuration
28F512A	12V Flash	$FFE0.0000_{16}$	$FFE0.FFFF_{16}$	64K * 8	
28F010A	"	$FFE0.0000_{16}$	$FFE1.FFFF_{16}$	128K * 8	x
28F020A	"	$FFE0.0000_{16}$	$FFE3.FFFF_{16}$	256K * 8	

CAUTION: The default Boot PROM socket is disabled with "ON"-position of the Boot PROM selection switch SW5-1.

3.6.2.2 Opt. Boot PROM Addresses (J28), SW5-1=OFF

Boot PROM access to default Boot PROM and optional Boot PROM (Boot PROM selection switch SW5-1 in position OFF)

Used Device	Device Type	Start Address	End Address	Total Capacity	Default Configuration
29F010	5V Flash	$FFE8.0000_{16}$	$FFE9.FFFF_{16}$	128K * 8	Optional
29F040	"	$FFE8.0000_{16}$	$FFEF.FFFF_{16}$	512K * 8	
28C512	12V Flash	$FFE8.0000_{16}$	$FFE8.FFFF_{16}$	64K * 8	
28C010	"	$FFE8.0000_{16}$	$FFE9.FFFF_{16}$	128K * 8	
28F512A	"	$FFE8.0000_{16}$	$FFE8.FFFF_{16}$	64K * 8	
28F010A	"	$FFE8.0000_{16}$	$FFE9.FFFF_{16}$	128K * 8	
28F020A	"	$FFE8.0000_{16}$	$FFEB.FFFF_{16}$	256K * 8	
27C010	EPROM/OTP	$FFE8.0000_{16}$	$FFE9.FFFF_{16}$	128K * 8	
27C020	"	$FFE8.0000_{16}$	$FFEB.FFFF_{16}$	256K * 8	
27C040	"	$FFE8.0000_{16}$	$FFEF.FFFF_{16}$	512K * 8	
27C080	"	$FFE8.0000_{16}$	$FFEF.FFFF_{16}$ [1)]	1M * 8	

1. The upper 512K of the 1M * 8 are not accessible in this mode.

3.6.2.3 Opt. Boot PROM Addresses (J28), SW5-1=ON

Boot PROM access to optional Boot PROM only. Access to default Boot PROM is disabled. (Boot PROM selection switch SW5-1 in position ON)

Used Device	Device Type	Start Address	End Address	Total Capacity	Default Configuration
29F010	5V Flash	$FFE0.0000_{16}$	$FFE1.FFFF_{16}$	128K * 8	Optional
29F040	"	$FFE0.0000_{16}$	$FFE7.FFFF_{16}$	512K * 8	
28C512	12V Flash	$FFE0.0000_{16}$	$FFE0.FFFF_{16}$	64K * 8	
28C010	"	$FFE0.0000_{16}$	$FFE1.FFFF_{16}$	128K * 8	
28F512A	"	$FFE0.0000_{16}$	$FFE0.FFFF_{16}$	64K * 8	
28F010A	"	$FFE0.0000_{16}$	$FFE1.FFFF_{16}$	128K * 8	
28F020A	"	$FFE0.0000_{16}$	$FFE3.FFFF_{16}$	256K * 8	
27C010	EPROM/OTP	$FFE0.0000_{16}$	$FFE1.FFFF_{16}$	128K * 8	
27C020	"	$FFE0.0000_{16}$	$FFE3.FFFF_{16}$	256K * 8	
27C040	"	$FFE0.0000_{16}$	$FFE7.FFFF_{16}$	512K * 8	
27C080	"	$FFE0.0000_{16}$	$FFEF.FFFF_{16}$	1M * 8	

3.6.3 Summary of the Boot PROM Area

Not Allowed Access with Function Code	111
Supported Port Size	Byte
Maximum Capacity	1 Mbyte
Default Access Time	200 ns
Access Address	$FFE0.0000_{16}$ - $FFEF.FFFF_{16}$
No. of Devices to be Installed	1 or 2

3.7 The Local SRAM Memory

The SRAM memory is dedicated to the on-board default SRAM and an optional SRAM.

For the optional SRAM there are two 32-pin DIL socket positions available. The socket position J87 allows the usage of a 28-pin SRAM or a 32-pin SRAM with a high active chip select for pin 30. The socket position J88 allows the usage of a 32-pin SRAM. The default SRAM is located in socket position J86.

The SRAM address map between the default and the optional SRAM can be controlled by switch SW11-4.

SW11-4	Start Address of	
	Default SRAM	Optional SRAM
OFF	$FFC0.0000_{16}$	Disabled
ON	$FFC8.0000_{16}$	$FFC0.0000_{16}$

The SRAM memory at sockets J86, J87 and J88 allows the user to retain data when the power supply is switched off. A backup provides the current for the SRAM standby mode.

3.7.1 Memory Organization SRAM

The local SRAM memory is connected to the local 8-bit data bus, providing a byte-wide port. Succeeding bytes seen by the microprocessor are handled in the same manner as succeeding bytes for the local SRAM memory.

Byte, word, and long-word accesses are managed by the dynamic bus sizing of the microprocessor. For further details, please refer to the manual of the microprocessor.

Data can be read from and written to any address; odd, even or unaligned in byte, word, or long-word format.

Example of Data Transfers:

The following instruction is fully supported from the SRAM memory area:

```
MOVE.X  ($FFC0 000Y), D0

  X = B = Byte                1 Byte
  X = W = Word                2 Bytes
  X = L = Long Word           4 Bytes

  Y = 0
  Y = 1
  Y = 2
  Y = 3
    .
    .
    .
```

All combinations of the listed instructions are allowed and possible.

3.7.2 Used Devices for SRAM Area

The default SRAM capacity is 32 Kbyte. The following low power device types (marked with -L or -LL) are supported by the J87 socket position. The package type must be either 28-pin DIL or 32-pin DIL.

Device	Device Capacity	Default Configuration
KM 62256L[1)]	32K * 8	Optional
MB 84256L[1)]	32K * 8	
M5M 5256L[1)]	32K * 8	
KM 681000L	128K * 8	
M5M 510008L	128K * 8	

1. These devices must be installed with pin 1, 2, 31, 32 left free.

The following DIL-device types are supported by the J88 socket position.

Device	Device Capacity	Default Configuration
M5M 5408L	512K * 8	Optional

CAUTION: A device can only be assembled in either socket J87 or J88.

NOTE: The setup parameters of FGA-002 are stored in the SRAM located in the lower half of the SRAM address space. If the optional SRAM is enabled, the setup parameters of the optional SRAM will be used.

3.7.3 Access Time Selection of the SRAM Area

The access time of the SRAM area is software programmable in the FGA-002 Gate Array. Four fixed access times are available: 2.0, 1.0, 0.5 and 0.25 microseconds.

3.7.4 Backup Power for the SRAM Area

There are two sources for backup power. One from the VMEbus +5VSTDBY line and the other one from a CR2032-type lithium battery installed in the battery socket at location BAT 1.

Backup power from the VMEbus +5VSTDBY line is enabled by SW11-2. Backup power from the battery is enabled by SW11-1.

The SRAM memory, both the default SRAM (on-board) and the optional SRAM (sockets J87 and J88) are powered by backup power circuitry. This maintains the power supply for the SRAM memory to retain its non-volatile storage.

Under normal operation the backup power circuitry connects the +5V power supply to the SRAM memory. When the main +5V supply fails, backup power may be supplied from one of two alternate sources. The VMEbus +5VSTDBY line may be used to provide backup power under power-fail conditions. The switchover from normal +5V to +5VSTDBY is fully automatic; whichever voltage is higher will be available to the SRAM memory.

As a second alternative, the backup power may be supplied by battery. Should the +5V and/or +5VSTDBY supplies drop below approximately +3.3 volts, the on-board battery will be used.

This is controlled for +5VSTDBY by switch SW11-2.

SW11-2	Description
OFF (default)	Backup from +5VSTDBY for RTC and SRAM disabled
ON	Backup from +5VSTDBY for RTC and SRAM enabled

Backup from the battery is controlled by switch SW11-1 and SW11-3.

SW11-1	SW11-3	Default Position	Description
OFF	OFF	x	Backup from battery disabled for RTC and SRAM
OFF	ON		
ON	OFF		Backup from battery enabled for RTC, but disabled for SRAM
ON	ON		Backup from battery enabled for RTC and SRAM

3.7.5 Summary of the SRAM Area

Not Allowed Access with Function Code	111
Supported Port Size	Byte
Default Access Time	100ns
Access Address	$FFC0.0000_{16}$ - $FFCF.FFFF_{16}$
Capacity of Default SRAM	32 Kbytes
Maximum Capacity of Optional SRAM	512 Kbytes

3.8 The Real-Time Clock (RTC) 72423

There is an RTC 72423 installed on the CPU board, containing its own crystal to maintain accurate time and date. A battery is provided on the CPU board to allow the RTC to run even under power-down conditions.

3.8.1 Address Map of the RTC Registers

The RTC 72423 has a 4-bit data bus. It must be accessed in byte mode and the upper four bits (4..7) are "don't care" during read and write accesses. The base address of the RTC is FF80.3000$_{16}$. The following table shows the register layout of the RTC 72423.

Table 16: RTC Register Layout

Default I/O Base Address: $FF80 0000 Offset: $0000 3000 Name: RTC			
Address (HEX)	**Offset**	**Label**	**Description**
FF803000	00	RTC1SEC	1 Second Digit Register
FF803001	01	RTC10SEC	10 Second Digit Register
FF803002	02	RTC1MIN	1 Minute Digit Register
FF803003	03	RTC10MIN	10 Minute Digit Register
FF803004	04	RTC1HR	1 Hour Digit Register
FF803005	05	RTC10HR	PM/AM and 10 Hour Digit Register
FF803006	06	RTC1DAY	1 Day Digit Register
FF803007	07	RTC10DAY	10 Day Digit Register
FF803008	08	RTC1MON	1 Month Digit Register
FF803009	09	RTC10MON	10 Month Digit Register
FF80300A	0A	RTC1YR	1 Year Digit Register
FF80300B	0B	RTC10YR	10 Year Digit Register
FF80300C	0C	RTCWEEK	Week Register
FF80300D	0D	RTCCOND	Control Register D
FF80300E	0E	RTCCONE	Control Register E
FF80300F	0F	RTCCONF	Control Register F

3.8.2 RTC Programming

The following programming example shows how to read from or write to the RTC. Please note that the RTC must be stopped prior to reading the date and time registers.

SEE ALSO: For further details, please refer to the RTC 72423 data sheet in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

3.8.3 RTC Programming Example

```
/*************************************
**  read RTC 72423 and load to RAM      **
**      30-Oct-87  M.S.                 **
*************************************/

setclock(sy)
register struct SYRAM *sy;
{
register struct rtc7242 *rtc = RTC2;
register long count=100000l;

rtc->dcontrol = 1;                      /* hold clock */
while(count--)
    if(rtc->dcontrol&0x02)
          break;
if(!count)
   { printf("\nCannot read Realtime Clock");
     rtc->dcontrol = 0;
     return; }
sy->_ssec[0] = (unsigned char)((rtc->sec10reg&0x07)*10 +
(rtc->sec1reg&0x0f));
sy->_smin    = (unsigned char)((rtc->min10reg&0x07)*10 +
(rtc->min1reg&0x0f));
sy->_shrs    = (unsigned char)((rtc->hou10reg&0x03)*10 +
(rtc->hou1reg&0x0f));
sy->_syrs[0] = (unsigned char)((rtc->yr10reg&0x0f)*10 +
(rtc->yr1reg&0x0f));
sy->_sday    = (unsigned char)((rtc->day10reg&0x03)*10 +
(rtc->day1reg&0x0f));
sy->_smon    = (unsigned char)((rtc->mon10reg&0x01)*10 +
(rtc->mon1reg&0x0f));
rtc->dcontrol = 0;                      /* start clock */
}
```

```
/*************************************
**     write RTC 72423 from RAM         **
**      30-Oct-87  M.S.                 **
*************************************/

writeclock(sy)
register struct SYRAM *sy;
{
register struct rtc7242 *rtc = RTC2;
register long count=100000l;

rtc->dcontrol = 1;                      /* hold clock */
while(count--)
   if(rtc->dcontrol&0x02)
          break;
if(!count)
   { printf("\nCannot read Realtime Clock");
     rtc->dcontrol = 0;
     return; }
```

```
rtc->fcontrol = 5;
rtc->fcontrol = 4;                      /* 24-hour clock */
rtc->sec10reg = sy->_ssec[0]/10;
rtc->sec1reg  = sy->_ssec[0]%10;
rtc->min10reg = (char)(sy->_smin/10);
rtc->min1reg  = (char)(sy->_smin%10);
rtc->hou10reg = (char)(sy->_shrs/10);
rtc->hou1reg  = (char)(sy->_shrs%10);
rtc->yr10reg  = sy->_syrs[0]/10;
rtc->yr1reg   = sy->_syrs[0]%10;
rtc->day10reg = sy->_sday/10;
rtc->day1reg  = sy->_sday%10;
rtc->mon10reg = sy->_smon/10;
rtc->mon1reg  = sy->_smon%10;
rtc->dcontrol = 0;                      /* start clock */
}
```

3.8.4 Backup Power for the RTC

There are two sources for backup power. One from the VMEbus +5VSTDBY line and the other one from a CR2032-type lithium battery installed in the battery socket at location BAT 1.

Backup power from the VMEbus +5VSTDBY line is enabled by SW11-2. Backup power from the battery is enabled by SW11-1.

The RTC is powered by backup power circuitry. This circuitry maintains power supply for the RTC to guarantee continuous operation.

Under normal operation the backup power circuitry connects the +5V power supply to the RTC. When the main +5V supply fails, backup power may be supplied from one of two alternate sources. The VMEbus +5VSTDBY line may be used to provide backup power under power-fail conditions. The switchover from normal +5V to +5VSTDBY is fully automatic; whichever voltage is higher will be available to the RTC.

As a second alternative, the backup power may be supplied by an on-board lithium battery. Should the +5V and/or +5VSTDBY supplies drop below approximately +3.3 volts, the on-board battery will be used.

This is controlled for +5VSTDBY by switch SW11-2.

SW11-2	Description
OFF (default)	Backup from +5VSTDBY for RTC and SRAM disabled
ON	Backup from +5VSTDBY for RTC and SRAM enabled

Backup from the battery is controlled by switch SW11-1 and SW11-3.

SW11-1	SW11-3	Default Position	Description
OFF	OFF	x	Backup from battery disabled for RTC and SRAM
OFF	ON		
ON	OFF		Backup from battery enabled for RTC, but disabled for SRAM
ON	ON		Backup from battery enabled for RTC and SRAM

3.8.5 Summary of the RTC

Device	72423 RTC
Access Address	$FF80.3000_{16}$
Access Mode	Byte only
Supported Transfers	Byte only, 4 LSB
Battery Type	CR2032
Interrupt Request Level	Software programmable
FGA-002 Interrupt Request Channel	Local IRQ #0

3.9 The DUSCC 68562

The Dual Universal Serial Communications Controller 68562 (DUSCC) is a single-chip MOS-LSI communications device that provides two independent, multiprotocol, full duplex receiver/transmitter channels in a single package. Each channel consists of a receiver, a transmitter, a 16-bit multifunction counter/timer, a digital phase locked loop (DPLL), a parity/CRC generator and checker, and associated control circuits.

3.9.1 Features of the DUSCC

- Dual full-duplex synchronous/asynchronous receiver and transmitter
- Multiprotocol operation consisting of:

  BOP: HDLC/ADCCP, SDLC, SDLC Loop, X.25 or X.75 link level

  COP: BISYNC, DDCMP, X.21

  ASYNC: 5-8 bit plus optional parity
- Programmable data encoding formats: NRZ, NRZI, FM0, FM1, Manchester
- 4 character receiver and transmitter FIFOs
- Individual programmable baud rate for each receiver and transmitter
- Digital phase locked loop
- User programmable counter/timer
- Programmable channel modes full/half duplex, auto echo, local loopback
- Modem control signals for each channel: RTS, CTS, DCD
- CTS and DCD programmable auto enables for receiver and transmitter
- Programmable interrupt on change of CTS or DCD

3.9.2 Address Map of DUSCC #1 Registers

The following tables contain the complete register map of DUSCC #1. The first table pertains only to registers for Port #4, the second table for Port #1, and the third table for registers common to both Port #1 and Port #4.

Table 17: Serial I/O Port #4 (DUSCC #1) Register Address Map

Port Base Address: $FF80 2000					
Address (HEX)	**Offset (HEX)**	**Reset Value**	**Mode**	**Label**	**Description**
$FF802000	00	00	R/W	DUSCMR1	Channel Mode Reg 1
$FF802001	01	00	R/W	DUSCMR2	Channel Mode Reg 2
$FF802002	02	--	R/W	DUSSS1R	SYN1/Secondary Adr Reg 1
$FF802003	03	--	R/W	DUSS2R	SYN2/Secondary Adr Reg 2
$FF802004	04	00	R/W	DUSTPR	Transmitter Parameter Reg
$FF802005	05	--	R/W	DUSTTR	Transmitter Timing Reg
$FF802006	06	00	R/W	DUSRPR	Receiver Parameter Reg
$FF802007	07	--	R/W	DUSRTR	Receiver Timing Reg
$FF802008	08	--	R/W	DUSCTPRH	Counter/Timer Preset Reg H
$FF802009	09	--	R/W	DUSCTPRL	Counter/Timer Preset Reg L
$FF80200A	0A	--	R/W	DUSCTCR	Counter/Timer Control Reg
$FF80200B	0B	00	R/W	DUSOMR	Output and Miscellaneous Reg
$FF80200C	0C	--	R	DUSCTH	Counter/Timer High
$FF80200D	0D	--	R	DUSCTL	Counter/Timer Low
$FF80200E	0E	00	R/W	DUSPCR	Pin Configuration Reg
$FF80200F	0F	--	R/W	DUSCCR	Channel Command Reg
$FF802010	10				
$FF802011	11				
$FF802012	12	--	W	DUSTFIFO	Transmitter FIFO
$FF802013	13				
$FF802014	14				
$FF802015	15				
$FF802016	16	--	R	DUSRFIFO	Receiver FIFO
$FF802017	17				
$FF802018	18	00	R/W	DUSRSR	Receiver Status Reg
$FF802019	19	00	R/W	DUSTRSR	Transmitter/Receiver Stat Reg
$FF80201A	1A	--	R/W	DUSICTSR	Input + Counter/Timer Stat Reg
$FF80201C	1C	00	R/W	DUSIER	Interrupt Enable Reg

Table 18: Serial I/O Port #1 (DUSCC #1) Register Address Map

Port Base Address: $FF80 2020

Address (HEX)	Offset (HEX)	Reset Value	Mode	Label	Description
$FF802020	00	00	R/W	DUSCMR1	Channel Mode Reg 1
$FF802021	01	00	R/W	DUSCMR2	Channel Mode Reg 2
$FF802022	02	--	R/W	DUSSS1R	SYN1/Secondary Adr Reg 1
$FF802023	03	--	R/W	DUSS2R	SYN2/Secondary Adr Reg 2
$FF802024	04	00	R/W	DUSTPR	Transmitter Parameter Reg
$FF802025	05	--	R/W	DUSTTR	Transmitter Timing Reg
$FF802026	06	00	R/W	DUSRPR	Receiver Parameter Reg
$FF802027	07	--	R/W	DUSRTR	Receiver Timing Reg
$FF802028	08	--	R/W	DUSCTPRH	Counter/Timer Preset Reg H
$FF802029	09	--	R/W	DUSCTPRL	Counter/Timer Preset Reg L
$FF80202A	0A	--	R/W	DUSCTCR	Counter/Timer Control Reg
$FF80202B	0B	00	R/W	DUSOMR	Output and Miscellaneous Reg
$FF80202C	0C	--	R	DUSCTH	Counter/Timer High
$FF80202D	0D	--	R	DUSCTL	Counter/Timer Low
$FF80202E	0E	00	R/W	DUSPCR	Pin Configuration Reg
$FF80202F	0F	--	R/W	DUSCCR	Channel Command Reg
$FF802030	10				
$FF802031	11				
$FF802032	12	--	W	DUSTFIFO	Transmitter FIFO
$FF802033	13				
$FF802034	14				
$FF802035	15				
$FF802036	16	--	R	DUSRFIFO	Receiver FIFO
$FF802037	17				
$FF802038	18	00	R/W	DUSRSR	Receiver Status Reg
$FF802039	19	00	R/W	DUSTRSR	Transmitter/Receiver Stat Reg
$FF80203A	1A	--	R/W	DUSICTSR	Input + Counter/Timer Stat Reg
$FF80203C	1C	00	R/W	DUSIER	Interrupt Enable Reg

Table 19: Ports #1 and #4 (DUSCC #1) Common Register Address Map

Port Base Address: $FF80 2000

Address (HEX)	Offset (HEX)	Reset Value	Mode	Label	Description
$FF80201B	1B	00	R/W	DUSGSR	General Status Register
$FF80201E	1E	0F	R/W	DUSIVR	Interrupt Vec Reg Unmodified
$FF80201F	1F	00	R/W	DUSICR	Interrupt Control Register
$FF80203E	3E	0F	R	DUSIVRM	Interrupt Vec Reg Modified

3.9.3 Address Map of DUSCC #2 Registers

The following tables contain the complete register map of DUSCC #2. The first table pertains only to registers for Port #2, the second table for Port #3, and the third table for registers common to both Port #2 and Port #3.

Table 20: Serial I/O Port #2 (DUSCC #2) Register Address Map

Port Base Address: $FF80 2200					
Address (HEX)	**Offset (HEX)**	**Reset Value**	**Mode**	**Label**	**Description**
$FF802200	00	00	R/W	DUSCMR1	Channel Mode Reg 1
$FF802201	01	00	R/W	DUSCMR2	Channel Mode Reg 2
$FF802202	02	--	R/W	DUSSS1R	SYN1/Secondary Adr Reg 1
$FF802203	03	--	R/W	DUSS2R	SYN2/Secondary Adr Reg 2
$FF802204	04	00	R/W	DUSTPR	Transmitter Parameter Reg
$FF802205	05	--	R/W	DUSTTR	Transmitter Timing Reg
$FF802206	06	00	R/W	DUSRPR	Receiver Parameter Reg
$FF802207	07	--	R/W	DUSRTR	Receiver Timing Reg
$FF802208	08	--	R/W	DUSCTPRH	Counter/Timer Preset Reg H
$FF802209	09	--	R/W	DUSCTPRL	Counter/Timer Preset Reg L
$FF80220A	0A	--	R/W	DUSCTCR	Counter/Timer Control Reg
$FF80220B	0B	00	R/W	DUSOMR	Output and Miscellaneous Reg
$FF80220C	0C	--	R	DUSCTH	Counter/Timer High
$FF80220D	0D	--	R	DUSCTL	Counter/Timer Low
$FF80220E	0E	00	R/W	DUSPCR	Pin Configuration Reg
$FF80220F	0F	--	R/W	DUSCCR	Channel Command Reg
$FF802210	10				
$FF802211	11				
$FF802212	12	--	W	DUSTFIFO	Transmitter FIFO
$FF802213	13				
$FF802214	14				
$FF802215	15				
$FF802216	16	--	R	DUSRFIFO	Receiver FIFO
$FF802217	17				
$FF802218	18	00	R/W	DUSRSR	Receiver Status Reg
$FF802219	19	00	R/W	DUSTRSR	Transmitter/Receiver Stat Reg
$FF80221A	1A	--	R/W	DUSICTSR	Input + Counter/Timer Stat Reg
$FF80221C	1C	00	R/W	DUSIER	Interrupt Enable Reg

Table 21: Serial I/O Port #3 (DUSCC #2) Register Address Map

Port Base Address: $FF80 2220					
Address (HEX)	**Offset (HEX)**	**Reset Value**	**Mode**	**Label**	**Description**
$FF802220	00	00	R/W	DUSCMR1	Channel Mode Reg 1
$FF802221	01	00	R/W	DUSCMR2	Channel Mode Reg 2
$FF802222	02	--	R/W	DUSSS1R	SYN1/Secondary Adr Reg 1
$FF802223	03	--	R/W	DUSS2R	SYN2/Secondary Adr Reg 2
$FF802224	04	00	R/W	DUSTPR	Transmitter Parameter Reg
$FF802225	05	--	R/W	DUSTTR	Transmitter Timing Reg
$FF802226	06	00	R/W	DUSRPR	Receiver Parameter Reg
$FF802227	07	--	R/W	DUSRTR	Receiver Timing Reg
$FF802228	08	--	R/W	DUSCTPRH	Counter/Timer Preset Reg H
$FF802229	09	--	R/W	DUSCTPRL	Counter/Timer Preset Reg L
$FF80222A	0A	--	R/W	DUSCTCR	Counter/Timer Control Reg
$FF80222B	0B	00	R/W	DUSOMR	Output and Miscellaneous Reg
$FF80222C	0C	--	R	DUSCTH	Counter/Timer High
$FF80222D	0D	--	R	DUSCTL	Counter/Timer Low
$FF80222E	0E	00	R/W	DUSPCR	Pin Configuration Reg
$FF80222F	0F	--	R/W	DUSCCR	Channel Command Reg
$FF802230	10				
$FF802231	11				
$FF802232	12	--	W	DUSTFIFO	Transmitter FIFO
$FF802233	13				
$FF802234	14				
$FF802235	15				
$FF802236	16	--	R	DUSRFIFO	Receiver FIFO
$FF802237	17				
$FF802238	18	00	R/W	DUSRSR	Receiver Status Reg
$FF802239	19	00	R/W	DUSTRSR	Transmitter/Receiver Stat Reg
$FF80223A	1A	--	R/W	DUSICTSR	Input + Counter/Timer Stat Reg
$FF80223C	1C	00	R/W	DUSIER	Interrupt Enable Reg

Table 22: Ports #2 and #3 (DUSCC #2) Common Register Address Map

Port Base Address: $FF80 2200					
Address (HEX)	**Offset (HEX)**	**Reset Value**	**Mode**	**Label**	**Description**
$FF80221B	1B	00	R/W	DUSGSR	General Status Register
$FF80221E	1E	0F	R/W	DUSIVR	Interrupt Vec Reg Unmodified
$FF80221F	1F	00	R/W	DUSICR	Interrupt Control Register
$FF80223E	3E	0F	R	DUSIVRM	Interrupt Vec Reg Modified

3.9.4 Configuration of Serial I/O Ports

Serial ports #1 and #4 are controlled by DUSCC #1 at location J90. Serial ports #2 and #3 are controlled by DUSCC #2 at location J89. In both cases, the DUSCC is connected to a local 8-bit data bus and is accessible via byte transfers.

The RS-232 interfaces of port #1, #2, #3 and #4 are identical except that port #4 is wired directly to the VMEbus P2 connector, while ports #1 through #3 are wired to 9-pin D-Sub connectors labeled "1" through "3" on the front panel of the CPU board and to the optional 5-row VMEbus P2 connector. As a factory option, ports #1 through #3 may also be connected to the default assembled 3-row VMEbus P2 connector.

In the sections that follow the figures will refer to the hardware configuration switches and to the serial driver/receiver hybrid sockets. For reference, the switches and hybrids are assigned to the serial ports according to the following table.

Table 23: Switches & Module Assignment for Serial Port Configuration

DUSCC	Channel	Hybrid	Switch SW12-	Port
#1	A	J118	4	"4"
#1	B	J119	1	"1"
#2	A	J123	3	"2"
#2	B	J124	2	"3"

3.9.5 RS-232 and RS-422/485 Driver Modules

In order to conserve board space and to simplify varying the serial interfaces, FORCE Computers has developed RS-232 and RS-422/485 hybrid modules: the FH-002 and FH-003.

These 21-pin single in-line (SIL) modules are installed in sockets so that they may be easily changed to meet specific application needs. Please refer to the preceding table for the assignment of hybrid modules to serial ports.

3.9.6 RS-232 Configuration of Serial Ports

By default, all four serial ports are configured for RS-232 compatibility. For proper RS-232 operation, the correct driver/receiver hybrid module must be installed (FH-002), and the serial interface switches SW12-1 – SW12-4 must be configured for RS-232. The default switch setting is detailed below.

The serial ports may be configured for RS-232 or RS-422/485.

SW12-1	Description
OFF (default)	RS-232 support for port #1 FH-002 hybrid must be installed for J118
ON	RS-422/485 support for port #1 FH-003 hybrid must be installed for J118

SW12-2	Description
OFF (default)	RS-232 support for port #3 FH-002 hybrid must be installed for J124
ON	RS-422/485 support for port #3 FH-003 hybrid must be installed for J124

SW12-3	Description
OFF (default)	RS-232 support for port #2 FH-002 hybrid must be installed for J123
ON	RS-422/485 support for port #2 FH-003 hybrid must be installed for J123

SW12-4	Description
OFF (default)	RS-232 support for port #4 FH-002 hybrid must be installed for J126
ON	RS-422/485 support for port #4 FH-003 hybrid must be installed for J126

When a serial port is configured for RS-232 operation, its I/O signals will be connected to the front panel DB-9 connector as follows (ports #1, #2 and #3 only):

Signal	Input[1)]	Output[1)]	9-pin D-Sub Connector	Description
DCD	x		1	Data Carrier Detect
RXD	x	x	2	Receive Data
TXD		x	3	Transmit Data
DTR		(x)	4	Data Terminal Ready
GND		x	5	Signal GND (supplied by FH-002)
DSR	(x)	(x)	6	Data Set Ready
RTS			7	Request to Send
CTS	x		8	Clear to Send
GND			9	Signal GND

1. As a factory option signals marked in brackets may be connected to the 9-pin D-Sub connector.

When serial port #4 is configured for RS-232 operation, its I/O signals will be connected to the VME P2 connector as follows:

Signal	Input[1)]	Output[1)]	VME Connector P2	Description
DCD	x		C29	Data Carrier Detect
RXD	x	x	C30	Receive Data
TXD		x	C31	Transmit Data
DTR		(x)	C32	Data Terminal Ready
GND			A32	Signal GND (supplied by FH-002)
DSR	(x)	(x)	A29	Data Set Ready
RTS			A30	Request to Send
CTS	x		A31	Clear to Send

1. As a factory option signals marked in brackets may be connected to the 9-pin D-Sub connector.

NOTE: For the connection to the IOBP-1 back panel, please refer to Section 2.11, 'The SYS68K/IOBP-1,' on page 26.

When the serial ports #1, #2 and #3 are configured for RS-232 operation, its I/O signals will be connected to the optional 5-row VME P2 connector as follows:

Port	Signal	Input[1)]	Output[1)]	VME Connector P2	Description
1	DCD	x		D8	Data Carrier Detect
	RXD	x	x	D4	Receive Data
	TXD		x	D3	Transmit Data
	DTR		(x)	D7	Data Terminal Ready
	GND		x	D9	Signal GND (supplied by FH-002)
	DSR	(x)	(x)	D24	Data Set Ready
	RTS			D5	Request to Send
	CTS	x		D6	Clear to Send
2	DCD	x		D15	Data Carrier Detect
	RXD	x	x	D11	Receive Data
	TXD		x	D10	Transmit Data
	DTR		(x)	D14	Data Terminal Ready
	GND		x	D16	Signal GND (supplied by FH-002)
	DSR	(x)	(x)	D25	Data Set Ready
	RTS			D12	Request to Send
	CTS	x		D13	Clear to Send
3	DCD	x		D22	Data Carrier Detect
	RXD	x	x	D18	Receive Data
	TXD		x	D17	Transmit Data
	DTR		(x)	D21	Data Terminal Ready
	GND		x	D23	Signal GND (supplied by FH-002)
	DSR	(x)	(x)	D26	Data Set Ready
	RTS			D19	Request to Send
	CTS	x		D20	Clear to Send

1. As a factory option, signals marked in brackets may be connected to the 9-pin D-Sub connector.

3.9.7 RS-422/RS-485 Hardware Configuration of Serial Ports

It is possible to configure any or all of the four serial ports to be RS-422 compatible or – as factory option – RS-485 compatible. By default, all four serial ports are configured for RS-232 operation. For proper RS-422/485 operation, the correct driver/receiver hybrid module must be installed (FH-003), and the serial interface switches must be properly configured.

SEE ALSO: For a list of the serial interface switches on the CPU board, please refer to Section 3.9.6, ‘RS-232 Configuration of Serial Ports,’ on page 63.

The RS-422 compatible interface supports TXD, RXD, RTS, CTS with differential outputs and inputs. Each port occupies the same nine pins of the D-Sub connector as in the RS-232 compatible configuration, but with a different signal association.

When a serial port is configured for RS-422 operation, its I/O signals will be connected to the front panel DB-9 connector as follows (ports #1, #2 and #3 only):

Signal	Input	Output	9-pin D-Sub Connector	Description
TXD-		x	1	Transmit Data
RTS-		x	2	Request to Send
CTS+	x		3	Clear to Send
RXD+	x		4	Receive Data
RXD-	x		5	Receive Data
TXD+		x	6	Transmit Data
RTS+		x	7	Request to Send
CTS-	x		8	Clear to Send
GND-			9	Signal GND

When serial port #4 is properly configured for RS-422 operation, its I/O signals will be connected to the VME P2 connector as follows:

Signal	Input	Output	VME Connector P2	Description
TXD-		x	C29	Transmit Data
RTS-		x	C30	Request to Send
CTS+	x		C31	Clear to Send
RXD+	x		C32	Receive Data
RXD-	x		A32	Receive Data
TXD+		x	A29	Transmit Data
RTS+		x	A30	Request to Send
CTS-	x		A31	Clear to Send

When the serial ports #1, #2 and #3 are configured for RS-422 operation, its I/O signals will be connected to the optional 5-row VME P2 connector as follows:

Port	Signal	Input	Output	VME Connector P2	Description
1	TXD-		x	D8	Transmit Data
	RTS-		x	D4	Request to Send
	CTS+	x		D3	Clear to Send
	RXD+	x		D7	Receive Data
	RXD-	x		D9	Receive Data
	TXD+		x	D24	Transmit Data
	RTS+		x	D5	Request to Send
	CTS-	x		D6	Clear to Send
2	TXD-		x	D15	Transmit Data
	RTS-		x	D11	Request to Send
	CTS+	x		D10	Clear to Send
	RXD+	x		D14	Receive Data
	RXD-	x		D16	Receive Data
	TXD+		x	D25	Transmit Data
	RTS+		x	D12	Request to Send
	CTS-	x		D13	Clear to Send
3	TXD-		x	D22	Transmit Data
	RTS-		x	D18	Request to Send
	CTS+	x		D17	Clear to Send
	RXD+	x		D21	Receive Data
	RXD-	x		D23	Receive Data
	TXD+		x	D26	Transmit Data
	RTS+		x	D19	Request to Send
	CTS-	x		D20	Clear to Send

3.9.8 Termination Resistors for RS-422/RS-485 Configuration

If termination resistors are necessary to compensate for various cable lengths and to reduce signal reflections, it must be done externally from the CPU-30 R4 (e.g. via a cable connector).

The resistor value is user application dependent, but a recommended value is 1000 Ohms.

3.9.9 Summary of DUSCC #1

Device	68562 DUSCC
Access Address	$FF80.2000_{16}$
Port Width	Byte
Interrupt Request Level	Software programmable
FGA-002 Interrupt Level	Local IRQ #4

3.9.10 Summary of DUSCC #2

Device	68562 DUSCC
Access Address	$FF80.2200_{16}$
Port Width	Byte
Interrupt Request Level	Software programmable
FGA-002 Interrupt Channel	Local IRQ #5

3.10 The PI/T 68230

The MC68230 Parallel Interface/Timer provides versatile double buffered parallel interfaces and an operating system oriented timer. The parallel interfaces operate in unidirectional or bidirectional modes, either 8 or 16 bits wide. The PI/T contains a 24-bit wide counter and a 5-bit prescaler.

3.10.1 Features of the PI/T

- MC68000 Bus Compatible
- Port Modes Include:
  - Bit I/O
  - Unidirectional 8 bit and 16 bit
  - Bidirectional 8 bit and 16 bit
- Selectable Handshaking Options
- 24-bit Programmable Timer
- Software Programmable Timer Modes
- Contains Interrupt Vector Generation Logic
- Separate Port and Timer Interrupt Service Requests
- Registers are Read/Write and Directly Addressable

3.10.2 Address Map of the PI/T #1 Registers

PI/T #1 is accessible via the 8-bit local I/O bus (byte mode). The following table shows the register layout of the PI/T #1.

Table 24: PI/T #1 Register Layout

Default
I/O Base Address: $FF80 0000
Offset: $0000 0C00
Name: PI_T1

Address (HEX)	Offset (HEX)	Reset Value	Label	Description
FF800C00	00	00	PIT1 PGCR	Port General Control Register
FF800C01	01	00	PIT1 PSRR	Port Service Request Register
FF800C02	02	00	PIT1 PADDR	Port A Data Direction Register
FF800C03	03	00	PIT1 PBDDR	Port B Data Direction Register
FF800C04	04	00	PIT1 PCDDR	Port C Data Direction Register
FF800C05	05	00	PIT1 PIVR	Port Interrupt Vector Register
FF800C06	06	00	PIT1 PACR	Port A Control Register
FF800C07	07	00	PIT1 PBCR	Port B Control Register
FF800C08	08	--	PIT1 PADR	Port A Data Register
FF800C09	09	--	PIT1 PBDR	Port B Data Register
FF800C0A	0A	--	PIT1 PAAR	Port A Alternate Register
FF800C0B	0B	--	PIT1 PBAR	Port B Alternate Register
FF800C0C	0C	--	PIT1 PCDR	Port C Data Register
FF800C0D	0D	--	PIT1 PSR	Port Status Register
FF800C10	10	00	PIT1 TCR	Timer Control Register
FF800C11	11	0F	PIT1 TIVR	Timer Interrupt Vector Register
FF800C12	12	--	PIT1 CPR	Counter Preload Register
FF800C13	13	--	"	"
FF800C14	14	--	"	"
FF800C15	15	--	"	"
FF800C16	16	--	PIT1 CNTR	Count Register
FF800C17	17	--	"	"
FF800C18	18	--	"	"
FF800C19	19	--	"	"
FF800C1A	1A	00	PIT1 TSR	Timer Status Register

3.10.3 I/O Configuration of PI/T #1

The following table lists all I/O signals connected to PI/T #1. The functions of these signals are described in the corresponding chapter.

SEE ALSO: Additional information is provided in the PI/T data sheet, included in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

Table 25: PI/T #1 Interface Signals

Pin	Function	In/Out
PA0	Rotary Switch 1	I
PA1	"	I
PA2	"	I
PA3	"	I
PA4	Rotary Switch 2	I
PA5	"	I
PA6	"	I
PA7	"	I

Table 25: PI/T #1 Interface Signals (Continued)

Pin	Function	In/Out
H1	User I/O via optional B5 or optional 5-row VME P2 connector	I
H2	"	I/O
H3	"	I
H4	"	I/O
PB0	Floppy Disk Drive Control	O
PB1	"	O
PB2	"	O
PB3[1)]	"	I
PB4[1)]	"	I
PB5	"	O
PB6	DMA Controller Control	O
PB7	"	O
PC0	User I/O via optional B5 or optional 5-row VME P2 connector	I/O
PC1	"	I/O
PC2	Not Used, Reserved	-
PC3	Timer Interrupt Request	O
PC4	User I/O via optional B5 or optional 5-row VME P2 connector	I/O
PC5	Port Interrupt Request	O
PC6	68882 FPCP Sense	I
PC7	User I/O via optional B5 or optional 5-row VME P2 connector	I/O

1. reserved

3.10.4 Rotary Switches at PI/T #1

PA0-PA7 lines:

There are two rotary switches installed on the front panel of the CPU board. The position of each switch can be read in via port A of PI/T #1.

Each rotary switch provides four bits of data. Therefore, each switch has 16 possible positions and the value printed on the switch (i.e., 0-9 and A-F) can be read from the lines PA0-PA3 (SW1) and PA4-PA7 (SW2) of PI/T #1.

The following table lists the input signals of PI/T #1 in relation to the rotary switch signals.

Table 26: Rotary Switch Signals Assignment

Port A Bit	Switch	Bit
A0	"1"	0
A1	"1"	1
A2	"1"	2
A3	"1"	3
A4	"2"	0
A5	"2"	1
A6	"2"	2

Table 26: Rotary Switch Signals Assignment (Continued)

Port A Bit	Switch	Bit
A7	"2"	3

NOTE: The rotary switches serve a special function in conjunction with the RESET and ABORT switches. This functionality is built into the Boot EPROM and is described in detail in the Boot Software description of the *FGA-002 User's Manual*.

For application programs, the rotary switches can be used as a general purpose input channel for diagnostics, configuration selection, or automatic system boot with different configurations.

VMEPROM uses the rotary switches for automatic configuration.

3.10.5 Floppy Disk Drive Control Lines at PI/T #1

PB0-PB5:

These lines control the FDC37C65C floppy disk drive interface. They perform the functions listed in the table below.

Pin	Signal Name
PB0	DCHGEN
PB1	DRV
PB2	PCVAL
PB3	Reserved
PB4	Reserved
PB5	EJECT[1)]

1. factory option

Pins PB3 and PB4 of PI/T #1 are reserved. They must be programmed for input.

As a factory option the EJECT line (PB5) may be used as an output. This signal is shared with the HLOAD signal of the Floppy Controller.

SEE ALSO: Please refer to Section 3.12, 'The Floppy Disk Controller,' on page 84 for more information.

3.10.6 DMA Control Lines at PI/T #1

PB6, PB7:

These lines control the local peripheral bus when the DMA controller is active. The signal PC6 controls the direction of the DMA transfer (read vs. write). The PC7 signal selects the target of the DMA transfer (SCSI controller or Floppy controller). The following table clarifies all four possible combinations.

PB7	PB6	Function
0	0	DMA write to SCSI
0	1	DMA read from SCSI
1	0	DMA write to FDC
1	1	DMA read from SCSI

NOTE: An additional DMA control line is described in Section 3.10.21, ‘DMA Control Line at PI/T #2,’ on page 78.

3.10.7 8-Bit User Defined I/O Port at PI/T #1

PC0, PC1, PC4, PC7, H1-H4: (FACTORY OPTION)

This 8-bit input port may be available (as a factory option) at the 16-pin connector B5 and may be available (as a factory option) at the 5-row VME P2 connector. Four bits are connected to port C of PI/T #1 and can be used as inputs or outputs. The remaining four bits are connected to the handshake pins of PI/T #1. Handshake pins H2 and H4 may be used as either inputs or outputs. Handshake pins H1 and H3 may only be used as inputs.

PI/T #1	B5	VME P2	I/O
H1	4	Z25	I
H2	3	Z27	I/O
H3	2	Z29	I
H4	1	Z31	I/O
C0	8	D27	I/O
C1	7	D28	I/O
C4	6	D29	I/O
C7	5	D30	I/O

The connector B5 (factory option) provides, besides the 8-bit user port signals, a power signal GND on pins 9, 10, 11, 12, 13, 14, 15, and 16.

3.10.8 Interrupt Request Signals of PI/T #1

TOUT:

The PI/T #1 pin PC3 is used as an interrupt request output. The 24-bit timer can generate interrupt requests at a software programmable level. This interrupt request line is connected to the IRQ #2 of the FGA-002.

PIRQ:

The PI/T #1 pin PC5 may be used as a port interrupt request output. Normally, this pin is not used. However, PC5 may be used to generate an additional interrupt if switch SW13-1 is OFF. In this case, the Port Interrupt Request and the Timer Interrupt Request will use the same FGA-002 interrupt channel. Therefore, they will generate interrupts at the same interrupt priority level, and the user's software may need to poll the PI/T device to determine the actual cause of the interrupt. For compatibility with earlier boards, this pin is normally not used.

SW13-1	Function
OFF (default)	Port Interrupt Disabled
ON	Port Interrupt Enabled

NOTE: There is another timer interrupt signal at PI/T #2 which doesn't have anything to do with these signals of PI/T #1.

3.10.9 Floating Point Coprocessor Sense Line at PI/T #1

PC6:

This line reports whether or not an FPCP is installed on the CPU board.

PC6	Function
0	FPCP Installed
1	FPCP Not Installed

3.10.10 Reserved Line at PI/T #1

PC2:

This line is not used. In order to retain compatibility with earlier and future versions, this line should not be used in any applications.

3.10.11 Summary of PI/T #1

Device	68230 PI/T
Access Address	$FF80.0C00_{16}$
Port Width	Byte
Interrupt Request Level	Software programmable
FGA-002 Interrupt Channel (Timer IRQ)	Local IRQ #2

3.10.12 Address Map of the PI/T #2 Registers

The PI/T2 is accessible via the 8-bit local I/O bus (byte mode). The following table shows the register layout of PI/T2.

Table 27: PI/T #2 Register Layout

Default
I/O Base Address: $FF80 0000
Offset: $0000 0E00
Name: PI_T2

Address (HEX)	Offset (HEX)	Reset Value	Label	Description
FF800E00	00	00	PIT2 PGCR	Port General Control Register
FF800E01	01	00	PIT2 PSRR	Port Service Request Register
FF800E02	02	00	PIT2 PADDR	Port A Data Direction Register
FF800E03	03	00	PIT2 PBDDR	Port B Data Direction Register
FF800E04	04	00	PIT2 PCDDR	Port C Data Direction Register
FF800E05	05	00	PIT2 PIVR	Port Interrupt Vector Register
FF800E06	06	00	PIT2 PACR	Port A Control Register
FF800E07	07	00	PIT2 PBCR	Port B Control Register
FF800E08	08	--	PIT2 PADR	Port A Data Register
FF800E09	09	--	PIT2 PBDR	Port B Data Register
FF800E0A	0A	--	PIT2 PAAR	Port A Alternate Register
FF800E0B	0B	--	PIT2 PBAR	Port B Alternate Register
FF800E0C	0C	--	PIT2 PCDR	Port C Data Register
FF800E0D	0D	--	PIT2 PSR	Port Status Register
FF800E10	10	00	PIT2 TCR	Timer Control Register
FF800E11	11	0F	PIT2 TIVR	Timer Interrupt Vector Register
FF800E12	12	--	PIT2 CPR	Counter Preload Register
FF800E13	13	--	"	"
FF800E14	14	--	"	"
FF800E15	15	--	"	"
FF800E16	16	--	PIT2 CNTR	Count Register
FF800E17	17	--	"	"
FF800E18	18	--	"	"
FF800E19	19	--	"	"
FF800E1A	1A	00	PIT2 TSR	Timer Status Register

3.10.13 I/O Configuration of PI/T #2

The following table lists all I/O signals connected to PI/T #2. The functions of these signals are described in the corresponding chapter.

SEE ALSO: Additional information is provided in the PI/T data sheet, included in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

Table 28: PI/T #2 Interface Signals

Pin	Function	In/Out
PA0	User I/O via optional B6 or optional 5-row VME P2 connector	I/O
PA1	"	I/O
PA2	"	I/O
PA3	"	I/O
PA4	"	I/O
PA5	"	I/O
PA6	"	I/O
PA7	"	I/O
H1	User I/O via optional B6 or optional 5-row VME P2 connector	I
H2	"	I/O
H3	"	I
H4	"	I/O
PB0	Memory Size	I
PB1	"	I
PB2	"	I
PB3	Board ID	I
PB4	"	I
PB5	"	I
PB6	"	I
PB7	"	I
PC0	Hardware ID	I
PC1	Hardware ID	I
PC2	Status of write protection for (default and optional) Boot PROMs	I
PC3	Timer Interrupt Request	O
PC4	Status of write protection for SYSTEM-Flash Memory	I
PC5	DMA control	O
PC6	Flash programming voltage control	O
PC7	Unused, reserved	I

3.10.14 12-Bit User I/O Port at PI/T #2

PA0-PA7, H1-H4:

This 12-bit I/O port may be available (as a factory option) at a 16-pin connector B6 and may be available (as a factory option) at the 5-row VME P2 connector, providing a convenient connection for a flat cable. Eight bits are connected to port A of PI/T #2 and can be used as inputs or outputs. The remaining four bits are connected to the handshake pins of PI/T #2. This port can be used to build a Centronics type interface.

PI/T #2	Connection B6	VME-P2
PA0	10	Z1
PA1	7	Z3
PA2	11	Z5
PA3	6	Z7
PA4	12	Z9
PA5	5	Z11
PA6	13	Z13
PA7	4	Z15
H1	14	Z17
H2	3	Z19
H3	15	Z21
H4	2	Z23

The connector B6 (factory option) provides, besides the 12-bit user port signals, power signals. The +5V power on pin 1 and pin 16 is protected by a non destroyable 1A-fuse. The GND power is connected at pin 8 and pin 9.

3.10.15 Memory Size Identification at PI/T #2

PB0-PB2:

From these lines, the on-board Shared RAM capacity can be read in by software. The following assignment is defined:

B2	B1	B0	Memory Capacity
0	0	0	32 Mbytes
0	0	1	16 Mbytes
0	1	0	8 Mbytes
0	1	1	4 Mbytes
1	0	0	Reserved

3.10.16 Board Identification at PI/T #2

PB3-PB7:

From these lines, the CPU board identification number can be read in by software. Every CPU board has a unique number. Different versions of one CPU board (i.e. different speeds, capacity of memory, or modules) contain the same identification number. In the case of the CPU-30 R4, the number is ten ("10" decimal or $0A_{16}$ hexadecimal "01010" binary).

3.10.17 Interrupt Request Signal of PI/T #2

TOUT:

PI/T #2 pin PC3 is used as a Timer Interrupt Request output. The 24-bit timer can generate interrupt requests at a software programmable interval. The Timer Interrupt Request line is connected to the local IRQ #3 of the FGA-002. Optionally, the Timer Interrupt Request may be used as a "watchdog" reset generator.

If the PI/T #2 Timer Interrupt Request line is connected as a watchdog reset generator, then the user can program the 24-bit timer to generate a local hardware reset whenever the timer counts down to zero. This allows the user to implement a simple watchdog timer. In normal operation, the timer would never expire. Instead, the user's software (for example, a background system task) would constantly restart the timer. Should the timer fail to be restarted, the PI/T will generate a low level on the Timer Interrupt Request output pin. That will trigger the Reset Generator, resetting the entire CPU board. The watchdog reset generator option is enabled and disabled via a switch at location SW13-2.

SW13-2	Description
OFF (default)	Watchdog Timer Disabled
ON	Watchdog Timer Enabled

NOTE: There is another timer interrupt signal at PI/T #1 which doesn't have anything to do with these signals of PI/T #2.

3.10.18 PC0-PC1 Hardware ID at PI/T #2

To allow simple detection of different hardware implementations, a hardware ID-number can be read in on the PI/T #2 pins PC0-PC1.

PC1	PC0	Description
1	1	Revision 1, 2, 3
1	0	Revision 4
0	1	Reserved
0	0	Reserved

3.10.19 Floppy Drive Ready Signal at PI/T #2

PC2:

From this line the status of the write protection for the Boot PROM devices may be monitored.

CAUTION: PI/T #2 pin PC2 must be programmed as an input.

PC2	Function
0	Write to Boot PROM unprotected
1	Write to Boot PROM protected

3.10.20 Floppy Drive Write Protect Signal at PI/T #2

PC4:

From this line the status of the write protection for the System Flash Memory may be monitored.

CAUTION: PI/T #2 pin PC4 must be programmed as an input.

PC4	Function
0	Write to System Flash Memory unprotected
1	Write to System Flash Memory protected

CAUTION: Writes to the System Flash Memory must always be performed with a 4-byte wide port size and must be aligned on 4-byte boundaries.

3.10.21 DMA Control Line at PI/T #2

PC5:

This line controls the local devices when a DMA transfer is in progress. When the DMA controller accesses a local device (SCSI or FDC), this line must be set to "0".

SEE ALSO: There are two other DMA Control Lines (controlling the local peripheral bus) which are described in Section 3.10.6, ‘DMA Control Lines at PI/T #1,’ on page 72.

3.10.22 Flash Programming Control at PI/T #2

PC6:

A 12V-Vpp-Generator must be controlled to support programming of the System Flash Memory and the Boot PROM memory in the default Boot PROM socket.

CAUTION: PI/T #2 pin PC6 must be programmed as an output.

PC6	Function
0	Vpp on
1	Vpp off

The Vpp generator is shared between the System Flash Memory and the default Boot PROM socket.

SEE ALSO: For further information please refer to Section 3.5, ‘The System PROM Area,’ on page 42 and to Section 3.6.1.4, ‘Programming Flash Devices,’ on page 47.

3.10.23 Reserved Lines at PI/T #2

PC7:

These lines are not used. In order to retain compatibility with earlier and future versions, these lines should not be used in any applications.

3.10.24 Summary of PI/T #2

Device	68230 PI/T
Access Address	$FF80.0E00_{16}$
Port Width	Byte
Interrupt Request Level	Software programmable
FGA-002 Interrupt Channel Timer IRQ:	Local IRQ #3

3.11 SCSIbus Controller MB 87033/34

The MB 87033/34 SCSI Controller, with its up to 4 Mbytes/s data transfer rate, is installed on the CPU to interface directly to SCSI Winchester disks, optical drives or tape streamers.

All I/O signals are available on the user defined pins of the VMEbus P2 connector. The I/O signal assignment is compatible with the SYS68K/ ISCSI-1 Controller which allows the use of the SYS68K/IOBP-1 for interconnection to mass storage devices.

SEE ALSO: For further information please refer to Section 9.1.5, ‘SYS68K/ISCSI-1 Disk Controller,’ on page 153 and to Section 2.11, ‘The SYS68K/IOBP-1,’ on page 26.

The SCSI Controller on the CPU board is fully supported by the installed real-time monitor debugger VMEPROM.

3.11.1 Features of the 87033/34 SCSI Controller

- Functional superset of the MB87031 SCSI Controller
- Direct interface to SCSI bus devices with on-chip drivers
- Full support for SCSI control
- Service of either initiator or target device
- Eight byte data buffer register incorporated
- Transfer byte counter (24 bit)
- Independent control and data transfer bus
- Asynchronous data transfer speed of 2 Mbytes/s
- Synchronous data transfer speed up to 4 Mbytes/s

3.11.2 Address Map of MB 87033/34 Registers

The registers of the MB 87033/34 are accessible via the 8-bit local data bus (byte mode).

SEE ALSO: Additional information is provided in the MB 87033/34 data sheet, included in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

3.11.3 The SCSI DMA Controller

The 8-bit DMA channel of the SCSI Controller is directly connected to the installed DMA Controller (inside FGA-002 Gate Array) allowing data transfer with a maximum speed of 4 Mbyte/s.

The DMA Controller includes a 32-byte FIFO which waits until the 32 bytes are filled and then requests local bus mastership for an eight cycle data transfer (32 bit in parallel).

In addition to the 32-byte DMA FIFO, the DMA channel includes a second FIFO (8 bytes deep) to fill the DMA FIFO if the DMA transfer to main memory is taking place. This allows continuous data transfer on the local DMA bus with a data rate of 4 Mbyte/s without any timing gaps in between.

This technique permits the CPU to perform in a real time capacity because the ratio of CPU and DMA operation at the maximum SCSI data transfer rate of 4 Mbyte/s is 73% for the CPU, 20% for the DMA Controller and 7% for arbitration overhead. If the data transfer rate is less than 4 Mbyte/s, the percentage range of CPU operation increases and the DMAC range decreases while the overhead of 7% remains unchanged.

3.11.3.1 DMA Control Lines

Two output pins of PI/T #1 and one output pin of PI/T #2 are used to control the data direction, to start the DMA Controller, and select the local peripheral devices.

PI/T #1 Port B bits 6 and 7 control the local devices when DMA transfers are initiated. The following table shows which bit selects the direction of the DMA transfer and which bit selects the FDC or SCSI Controller.

PB7	PB6	Function
0	0	DMA write to SCSI
0	1	DMA read from SCSI
1	0	DMA write to FDC
1	1	DMA read from FDC

Port C bit 5 of PI/T #2 controls the local devices in case of direct memory access (DMA) as shown in the following table.

PC5	Function
0	DMA active
1	DMA is not active

The listing below provides a programming example for DMA transfers.

3.11.3.2 DMA Transfer Programming Example

```
*-------------------------------------------------------------
* DATA OUTPUT TO SCSI TARGET, USING THE MB87033/34 AND THE
* FGA-002 DMA CHANNEL
*-------------------------------------------------------------

                :
         BCLR    #$07,PBDR+PI_T      ;SELECT DMA WORKS WITH SCSI
         BCLR    #$06,PBDR+PI_T      ;SELECT TRANSFER DMA TO SCSI
  * THE FOLLOWING AUX VALUES ARE ONLY VALID FOR THE CPU-30 !
         MOVE.B  #$00,FGA02+AUXPINC  ;AUX PIN CONTROL
         MOVE.B  #$30,FGA02+AUXDST   ;AUXDSTSTART
         MOVE.B  #$45,FGA02+AUXDTE   ;AUXDSTTERM
         MOVE.B  #$00,FGA02+AUXDSTR  ;AUXDSTWEX
         MOVE.L  #BCOUNT,FGA02+DMABCNT ;BYTE COUNT
         MOVE.L  #SADDR,FGA02+DMASADR  ;DMA SOURCE ADDR
         MOVE.B  #$C5,FGA02+DMASATR  ;DMA SOURCE ATTRIBUTE (DPR)
         MOVE.B  #$C8,FGA02+DMADATR  ;DMA DEST ATTRIBUTE (AUX)
         MOVE.B  #$41,FGA02+DMAGEN   ;DMA GENERAL CONTROL
                 :
         MOVE.B  #$00,SCSI+TMODREG   ;SET SCSI TRANSFER MODE
         MOVE.B  #$80,SCSI+SCMDREG   ;SCSI COMMAND
         BCLR    #$00,PCDR+PI_T2     ;SET TO DMA ACTIVE
         MOVE.B  #$01,FGA02+DMARUNC  ;START DMA CONTROLLER
  WAIT   TST.B   FGA02+DMARUNC       ;POLL ON DMA READY
         BMI.B   WAIT
         BSET    #$00,PI_T2+PCDR     ;SET TO CPU ACTIVE
                 :
```

3.11.4 The SCSIbus

3.11.4.1 SCSIbus Configuration

Communication on the SCSIbus is only allowed between two SCSI devices at any given time. There may be a maximum of eight SCSI devices. Each SCSI device has a SCSI ID bit assigned. When two SCSI devices communicate on the SCSIbus, one acts as an initiator, and the target performs the operation. A SCSI device usually has a fixed role as an initiator or target, but some devices may be able to assume either role.

An initiator may address up to seven peripheral devices that are connected to a target. An option allows the addressing of up to 2048 peripheral devices per target using extended messages.

Of the eight SCSI devices supported on the SCSIbus, there can be any combination of initiators and targets. Certain SCSIbus functions are assigned to the initiator and other functions are assigned to the target. The initiator may arbitrate for the SCSIbus and select a particular target. The target may request the transfer of COMMAND, DATA, STATUS, or other information on the data bus, and in some cases, it may arbitrate for the SCSIbus and reselect an initiator for the purpose of continuing an operation.

Information transfers on the data bus are asynchronous and follow a defined REQ/ACK handshake protocol. One byte of information may be transferred with each handshake. An option is defined for synchronous data transfer.

3.11.4.2 SCSIbus Signal Termination

Each SCSIbus signal should be terminated at the physical start and the physical end of the SCSIbus. Therefore, the CPU-30 R4 provides circuitry for active termination.

Active termination can be controlled by switch SW7-3.

SW7-3	Description
OFF	Active SCSI termination on
ON	Active SCSI termination off

3.11.4.3 SCSIbus Terminator Power

The power for the terminator of any SCSI device will be provided from the CPU board directly, or from the SCSIbus itself. If termination power is not delivered from any other SCSI device, it is delivered from the CPU board.

The TERMPWR (terminator power) supply from the CPU board is protected by a self resetting fuse (1A max) and a diode in series, as defined in the SCSI specification.

The on-board terminators will draw power from the SCSIbus TERMPWR.

3.11.5 Summary of the SCSIbus Controller

Device	MB 87034
Access Address	$FF80.3400_{16}$
Port Width	Byte
Interrupt Request Level	Software Programmable
FGA-002 Interrupt Request Channel	Local IRQ #7

3.12 The Floppy Disk Controller

The CPU board contains a single chip floppy controller, the FDC37C65C. The FDC is connected to the DMA controller of the FGA-002 Gate Array. The installed driver/receiver circuits allow direct connection of 3 1/2" and 5 1/4" inch floppy drives. All I/O signals are available on the VMEbus P2 connector. The I/O signal assignment is compatible to the SYS68K/ISCSI-1 controller, which allows the use of the SYS68K/IOBP-1 and IOPI-2 for interconnection to mass storage devices.

3.12.1 Features of the FDC37C65C Controller

- Built-in data separator
- Built-in write precompensation
- 128-, 256-, 512- or 1024-byte sector lengths
- 3 1/2" or 5 1/4" single and double density
- Programmable stepping rate (2 to 6 ms)

3.12.2 Address Map of the FDC

The registers of the FDC are accessible via the 8 bit local I/O bus (byte mode). The following table shows the register layout of the FDC37C65C for the CPU-30 R4.

SEE ALSO: Additional information is provided in the FDC37C65C data sheet included in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

$FF80.3800_{16}$	Read Main Status Register (Write is illegal)
$FF80.3801_{16}$	Read Data Register Write Date Register
$FF80.3880_{16}$	Read DCHG Register Write Data Rate Selection Register
$FF80.3900_{16}$	Write Digital Output Register (Read is illegal)

3.12.3 Data Rate Support

The FDC37C65C allows two Data Rate Selection Options controlled via the Data Rate Selection Register. The CPU-30 R4 supports the 16 MHz and 9.6 MHz options.

SEE ALSO: For further details please refer to Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

3.12.4 Drive Select Support

The CPU-30 R4 supports two drive selects - DSEL 1 and DSEL 2 which are generated by the FDC37C65C-Controller.

3.12.5 Motor-On Support

The FDC37C65C Floppy Controller provides two signals for motor control. On the CPU-30 R4 they are tied together to build the Motor-On Signal.

3.12.6 DMA Control Lines

Two output pins of PI/T #1 and one output pin of PI/T #2 are used to control the data direction, to start the DMA Controller, and to control the used local devices.

PI/T #1 Port B bits 6 and 7 control the local devices in case of direct memory access (DMA). The following table shows which bit selects the direction of the DMA transfer and which bit selects the FDC or SCSI Controller.

PB7	PB6	Function
0	0	DMA write to SCSI
0	1	DMA read from SCSI
1	0	DMA write to FDC
1	1	DMA read from FDC

Port C bit 5 of PI/T #2 controls the local devices in case of direct memory access (DMA) as shown in the following table.

PC5	Function
0	DMA active
1	DMA is not active

SEE ALSO: The listing in Section 3.12.7.1, ‘DMA Transfer Programming Example,’ on page 86 provides a programming example for DMA transfers.

3.12.7 Floppy Disk Connector Assignment

Signal	VME P2 Connector
RDY	C17
SDSEL	C16
RDATA	C15

Signal	VME P2 Connector
WPROT	C14
TRU00	C13
WGATE	C12
WDATA	C11
STEPX	C10
DIRBC	C9
MOTOR	C8
DSEL1	C5, (C7)[1)]
DSEL2	C6, (C3)[2)]
INDEX	C4
HLOAD (EJECT)[3)]	C2
RPM	C1

1. Backward compatibility to DSEL3
2. backward compatibility to DSEL4
3. factory option

NOTE: For the connection to the IOBP-1 back panel, please refer to Section 2.11, ‘The SYS68K/IOBP-1,’ on page 26.

3.12.7.1 DMA Transfer Programming Example

```
*---------------------------------------------------------------
* READ DATA FROM FLOPPY, USING THE FDC AND THE FGA-002
* DMA CHANNEL
*---------------------------------------------------------------
                :
          BSET    #$07,PI_T1+PBDR(A4) ;SELECT DMA WORKS WITH FDC
          BSET    #$06,PI_T1+PBDR(A4) ;SELECT TRANSFER FDC TO DMA
  * THE FOLLOWING AUX VALUES ARE FOR CPU-30 ONLY !!!
          MOVE.B  #$00,FGA02+AUXPINC  ;AUX PIN CONTROL
          MOVE.B  #$07,FGA02+AUXSST   ;AUXSRCSTART
          MOVE.B  #$03,FGA02+AUXSTE   ;AUXSRCTERM
          MOVE.B  #$0F,FGA02+AUXSRCW  ;AUXSRCWEX
          MOVE.L  #BCOUNT,FGA02+DMABCNT ;BYTE COUNT
          MOVE.L  #DADDR,FGA02+DMADADR  ;DESTINATION ADDRESS
          MOVE.B  #$C8,FGA02+DMASATR  ;DMA SOURCE ATTRIBUTE (AUX)
          MOVE.B  #$C5,FGA02+DMADATR  ;DMA DEST ATTRIBUTE (DPR)
          MOVE.B  #$81,FGA02+DMAGEN   ;DMA GENERAL CONTROL
          MOVE.B  #$01,FGA02+DMARUNC  ;START DMA CONTROLLER
          MOVE.B  #$88,FDC+FCMDREG    ;READ SECTOR COMMAND
          BCLR    #$00,PI_T2+PCDR     ;SET TO DMA ACTIV
  WAIT    TST.B   FGA02+DMARUNC       ;WAIT UNTIL DMA IS READY
          BMI.S   WAIT
          BSET    #$05,PI_T2+PCDR     ;SET TO CPU ACTIVE
                  :
```

3.12.8 Jumper Setting on the Floppy Disk Drive

CAUTION: If the floppy disk drive contains a jumper which connects the floppy disk drive frame electrically with DC ground, insertion of this jumper is not allowed and can cause damage.

3.12.9 Summary of the Floppy Disk Controller

Device	FDC37C65C
Access Address	$FF80.3800_{16}$
Port Width	Byte
Interrupt Request Level	Software programmable
FGA-002 Interrupt Request Channel	Local IRQ #1

3.13 The Local Area Network Interface

The CPU board offers a Local Area Network (LAN) interface. The LAN interface consists of an Am7990 LANCE (Local Area Network Controller for Ethernet), an Am7992 SIA (Serial Interface Adapter), two 32-Kbyte static RAMs, and control logic. The SRAMs provide a 64-Kbyte data buffer between the LANCE and the local 68030 CPU, thus improving overall performance and reducing the risk of network overruns or underruns. Logic on the CPU board arbitrates which device - CPU or LANCE - gains mastership of the SRAM buffer.

3.13.1 Features of the Ethernet Interface

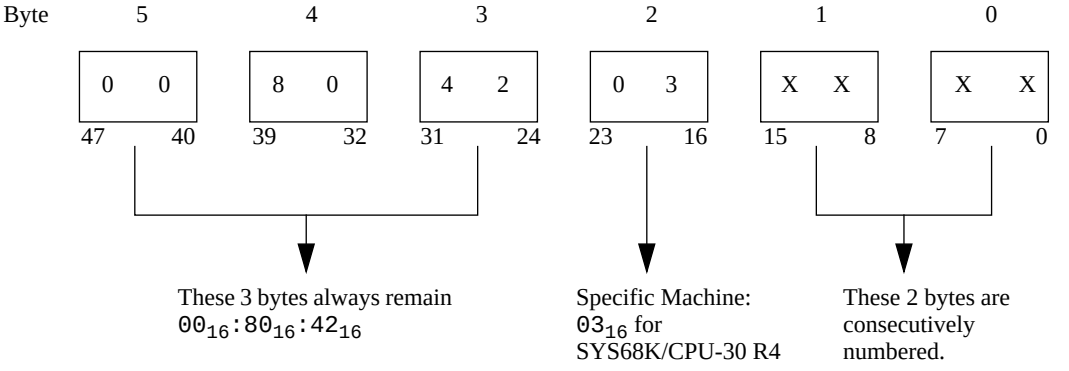
- Compatibility to IEEE 802.3/Ethernet
- Data rate of 10 Mbit per second
- DMA capability
- Interrupt generation
- 64 Kbytes of Local Buffer RAM

3.13.1.1 Ethernet Address

A 48-bit Ethernet address has been assigned to your CPU-30 R4 board. This Ethernet address has a general structure which ensures a unique number assignment for every board and is calculated from the board serial number.

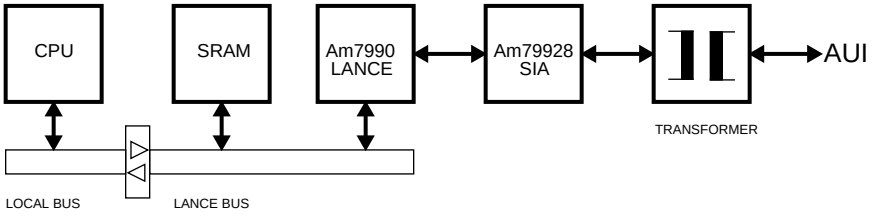
The unique Ethernet address is displayed by the banner when entering the FGA Boot debugger. FGA Boot also provides a utility function to get the CPU board's Ethernet address (see "#40 (0x28) Get Ethernet Number" on page 196).

Figure 4: The 48-bit (6-byte) Ethernet address



The figure below shows a simplified functional block diagram of the Ethernet Interface.

Figure 5: Functional Block Diagram of the Ethernet Interface



3.13.2 The Am7990 LANCE

The LANCE is a 68-pin VLSI device that provides many functions for the connection of a microprocessor to an Ethernet network.

In the transmitting mode the LANCE transfers data from the buffer SRAM to an internal FIFO (called a SILO). The serial output of the SILO is connected to the Am7992 SIA, where the data is sent to the Ethernet cable.

In the receiving mode the SIA transfers the received data from the Ethernet cable to the serial SILO input. The LANCE then transfers the data from the SILO to the buffer SRAM in 8-word (16 byte) bursts.

SEE ALSO: For detailed information please refer to the AM7990 LANCE data sheet in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

3.13.2.1 Address Map of the LANCE Registers

The LANCE contains one Register Address Pointer (RAP) and four Control/Status Registers (CSR[0..3]). To read or write to CSR1, CSR2, or CSR3, the LANCE must be stopped by setting the stop bit in CSR0. Therefore, CSR0 and RAP can be accessed at any time that the LANCE is in the slave mode. To read or write to CSR1 through CSR3, the number of the CSR must be written to the RAP first. The address map is shown in the following table.

Table 29: LANCE Register Layout

Default
I/O Base Address: $FEF8 0000
Offset: $0000 0000

Address (HEX)	Offset (HEX)	Reset Value	Label	Description
FEF80002	2	0000	RAP	Register Address Pointer
FEF80000	0	0004	CSR0	Control and Status Register 0
FEF80000	0	N/A	CSR1	Control and Status Register 1
FEF80000	0	N/A	CSR2	Control and Status Register 2
FEF80000	0	N/A	CSR3	Control and Status Register 3

N/A = NOT APPLICABLE OR NOT EFFECTED

3.13.2.2 The LANCE Interrupt

The LANCE is able to interrupt the CPU on a programmable level. It is connected to the Interrupt Request Channel 6 (IRQ #6) of the FGA-002 Gate Array. Like all on-board interrupts, the interrupt priority level of the LANCE may be selected by programming the FGA-002.

3.13.2.3 Summary of the LANCE

Device	AM7990
Access Address	$FEF8.0000_{16}$
Access Mode	Word Only
Interrupt Request Level	Programmable
FGA-002 Interrupt Request Channel	Local IRQ #6

3.13.3 The Am7992B Serial Interface Adapter (SIA)

The Am7992B Serial Interface Adapter (SIA) is a Manchester Encoder/Decoder compatible with IEEE-802.3 and Ethernet specifications.

3.13.4 Features of the Am7992B SIA

- Compatible with Ethernet/Cheapernet/IEEE-802.3 specifications
- Crystal controlled Manchester Encoder
- Manchester Decoder acquired clock & data within four bit times with an accuracy of +/-3 ns
- Guaranteed carrier and collision detection squelch threshold limits
- Carrier/collision detected for inputs greater than -275 mV
- No carrier/collision for inputs less than -175 mV
- Input signal conditioning reject transient noise
- Transients <10 ns for collision detector inputs
- Transients <20 ns for carrier detector inputs
- Receiver decodes Manchester data with worst case +/-19 ns of clock jitter (at 10 MHz)
- TTL compatible host interface
- Transmit accuracy +/-0.01% (without adjustments)

3.13.4.1 The Am7992B Transmitter

The transmitter encodes serial data coming from the LANCE into a Manchester II Code. The output signal is sent onto the Ethernet cable. An oscillator on the CPU board provides the 10 MHz reference frequency for the LANCE.

SEE ALSO: For further details please refer to the Am7992B data sheet in Section 4, ‘Circuit Schematics and Data Sheets,’ on page 115.

3.13.4.2 The Am7992B Receiver

The receiver has the responsibility of signalling to the LANCE when there is a message to receive. It decodes the Manchester-encoded data stream from the Ethernet cable and sends the data to the LANCE.

SEE ALSO: For further information please refer to the Am7992B data sheet in Section 4, 'Circuit Schematics and Data Sheets,' on page 115.

3.13.4.3 Network Interface Configuration

The LAN interface uses a standard 15-pin D-Sub connector providing a quick and secure connection to an Ethernet cable.

3.13.5 The LAN Buffer RAM

LAN Buffer RAM is accessible from addresses $FEF0.0000_{16}$ to $FEF0.FFFF_{16}$ and the port width is 16 bits (word).

The LAN RAM consists of two 32K * 8-bit SRAMs. These devices can be accessed by both the CPU and the LANCE after each device arbitrates for bus mastership.

With this detached memory, it is not necessary for the CPU to stop processing while the LANCE is bus master and accesses the LAN Buffer RAM. Thus, the real-time capabilities of the CPU board are preserved.

3.13.6 Summary of the LAN RAM

Devices	32K * 8 (2x)
Access Address	$FEF0.0000_{16}$ to $FEF0.FFFF_{16}$
Capacity	64 Kbytes
Port Width	16 bits (word)
FGA-002 Interrupt Request Channel	Local IRQ #6

3.14 Function Switches and Indication LEDs

3.14.1 RESET Function Switch

A reset of all on-board I/O devices, the FPCP and the CPU is performed when the RESET switch is pushed to the "UP" position. RESET is held active until the switch is in "DOWN" position.

In addition, a local timer guarantees a minimum reset time of 200-300ms. Power fail and power up also force a reset (200-300ms) to start the board if the supply voltage is out of range (below approximately 4.7 Volts).

If enabled, the reset is also driven to the VMEbus.

SEE ALSO: For more information, please refer to Section 3.22.2, ‘The SYSRESET* Signal,’ on page 110 and Section 3.23, ‘Reset Generation,’ on page 112. In combination with the ABORT switch, the RESET switch has a special function which is described in the Boot Software description of the *FGA-002 User's Manual*.

The Reset Function can be controlled by switch SW7-1.

SW7-1	Description
OFF (default)	Reset Key enabled
ON	Reset Key disabled

3.14.2 ABORT Function Switch

An interrupt on a software programmable level is provided on the board to allow an abort of the current program, to trigger a self-test or to start a maintenance program. The ABORT switch is activated in "UP" position and deactivated in "DOWN" position. In combination with the RESET switch, the ABORT switch has a special function which is described in the Boot Software description of the *FGA-002 User's Manual*.

The Abort Function can be controlled by switch SW7-2.

SW7-2	Description
OFF (default)	Abort Key enabled
ON	Abort Key disabled

3.14.3 "RUN" LED

The uppermost LED (below the RESET and ABORT switches) is the RUN LED.

This bicolor LED is green when the processor is not in HALT state. It is red during the reset phase, or when the processor is in HALT state.

3.14.4 "BM" LED

If the CPU board is the current VMEbus master, the green BM LED is lit. This provides the user with a convenient visual indication of the status of the VMEbus.

3.14.5 Rotary Switches

There are two rotary switches (SW1 and SW2) which are each four-bit hexadecimal encoded. These switches are completely under software control. The default setting is FF_{16}.

SEE ALSO: For a detailed description of the use of these switches under VMEPROM, please refer to Section 5.4.3, ‘Control Switches (Rotary Switches),’ on page 127.

In combination with the RESET and ABORT switches, the rotary switches have a special function which is described in the Boot Software description of the *FGA-002 User's Manual*.

3.14.6 Reserved Switches

Switches SW13-3 and SW13-4 are reserved and must be in the default OFF position.

3.15 The CPU Board Interrupt Structure

The FGA-002 Gate Array on the CPU board monitors all local and VMEbus interrupts. Each interrupt request from the local bus (SCSI and floppy disk controllers DUSCCs, RTC, PI/T timers, etc) as well as the FGA-002 specific interrupt requests are combined with seven VMEbus interrupt requests.

Each and every interrupt source, including the VMEbus IRQs, can be programmed to interrupt the CPU on an individually programmable priority level, from 1 through 7.

The Gate Array may supply the interrupt vector, or it may initiate an interrupt vector fetch from the I/O device or from the VMEbus.

In addition to local interrupts, the ACFAIL* and SYSFAIL* signals from the VMEbus can be programmed to interrupt the CPU on a software programmable level.

Interrupt vectors supplied by the FGA-002 all share a basic vector and a fixed vector offset for each source. The basic vector is software programmable.

The table below shows the connection between local devices and the local interrupt request of the FGA-002.

Device	Function(s)	IRQ
RTC	Various	0
FDC	Various	1
PI/T #1	Port and Timer Interrupt Requests (IRQs)	2
PI/T #2	Timer IRQ	3
DUSCC #1	Various	4
DUSCC #2	Various	5
LANCE	Various	6
SCSI	Various	7

3.16 VMEbus Interface

The CPU board contains a complete VMEbus interface which is compatible with the IEEE 1014 standard (VMEbus Revision C).

The VMEbus interface supports 8, 16, 32 bit, and unaligned data transfers. The extended, standard, and short I/O address modifier codes are implemented to interface to all existing VMEbus products.

Read-Modify-Write cycles on the VMEbus are also supported. The address strobe signal is held low during this cycle while the data strobe signals are driven low twice, once for the read cycle and once for the write cycle, and high between the both of them.

All seven VMEbus interrupt request signals are monitored by the FGA-002 which can optionally map every level and then interrupt the local CPU. A single level bus arbiter together with several release functions are implemented with all "Slot-1" system controller functions such as SYSRESET* driver and receiver, SYSCLK driver, and IACK daisy-chain driver.

The following sections describe the functions of the interface parts in detail.

For the connections to the IOBP-1 back panel, please refer to Section 2.11, ‘The SYS68K/IOBP-1,’ on page 26.

3.17 VMEbus Master Interface

3.17.1 Data Transfer Size of the VMEbus Interface

The memory map of the CPU board divides the 4-Gbyte address space of the 68030 microprocessor into areas of local memory, local I/O, FGA-002 internal registers, and VMEbus space. The VMEbus space is by far the largest portion of the address map. It is divided into ranges where different address bus widths (A32/A24/A16) and different data bus widths (D32/D16) will be used. The VMEbus interface also contains address ranges where the data transfer size is software programmable to be 16 or 32 bits wide.

In address ranges where the data transfer bus width is limited to 16 bits and a 32-bit transfer is attempted, the hardware on the CPU board will perform two consecutive transfers automatically, so that no overhead in software is necessary.

The following table lists the VMEbus address ranges and their associated address and data bus sizes in detail.

Table 30: Data Bus Size of the VMEbus (Master Interface)

Start Address	End Address	Address Size	Data Size
$xxxx.0000_{16}$ [1)]	$FAFF.FFFF_{16}$	A32	Prog.
$FB00.0000_{16}$	$FBFE.FFFF_{16}$	A24	Prog.
$FBFF.0000_{16}$	$FBFF.FFFF_{16}$	A16	Prog.
$FC00.0000_{16}$	$FCFE.FFFF_{16}$	A24	16
$FCFF.0000_{16}$	$FCFF.FFFF_{16}$	A16	16

1. VMEbus start address is dependent on the amount of on-board shared DRAM.

SEE ALSO: For further information, please refer to Section 3.4.5, ‘Shared RAM Addressing,’ on page 40.

VMEPROM includes a command (MEM) to set up the data bus transfer size of the three programmable areas. For example:

MEM displays the current data bus transfer size.

MEM 16 sets 8/16-bit data transfer size.

MEM 32 sets 8/16/32-bit data transfer size.

In addition, VMEPROM uses one bit of the rotary switches available on the front panel to select the data bus size of the VMEbus after RESET or power up.

This default configuration is useful if a user program or an operating system is started, and additional memory boards with known data sizes are installed.

SEE ALSO: For details on the usage of the rotary switches, please refer to Section 5.4.3, ‘Control Switches (Rotary Switches),’ on page 127.

Table 31: Defined VMEbus Transfer Cycles (D32 Mode)

Transfer Type	D31-D24	D23-D16	D15-D08	D07-D00
Byte Byte			X	X
Word			X	X
Long Word	X	X	X	X
Unaligned Word Unaligned Long Word A Unaligned Long Word B	 X	X X X	X X X	X
RMW[1)] Byte RMW Byte RMW Word RMW Long Word	 X	 X	X X X	X X X

1. RMW=Read-Modify-Write

Table 32: VMEbus Transfer Cycles (D16 Mode)

Transfer Type	D31-D24	D23-D16	D15-D08	D07-D00
Byte Byte			X	X
Word			X	X
RMW[1)] Byte RMW Byte RMW Word			X X	X X

1. RMW=Read-Modify-Write

3.17.2 Address Modifier Implementation

The VMEbus defines three different Address Modifier ranges as shown in the following table:

Table 33: Address Ranges

Mode	Address Lines Used	Short Form
Extended Addressing	A1-A31	A32
Standard Addressing	A1-A24	A24
Short I/O	A1-A15	A16

All allowed and defined Address Modifier (AM) Codes are listed in the next table. Those codes that are supported by the CPU board are marked with an asterisk (*).

The 4-Gbyte address range of the microprocessor is split into several areas to support all of the listed AM codes. The following table lists the address ranges and the supported AM codes for this range.

All VMEbus slave boards which will be addressed by the CPU board must recognize one or more of the AM codes in the table to guarantee proper operation.

Table 34: Address Modifier Codes

Code	Address Modifier						Function
	5	4	3	2	1	0	
3F	H	H	H	H	H	H	Standard Supervisory Block Transfer
*3E	H	H	H	H	H	L	Standard Supervisory Program Access
*3D	H	H	H	H	L	H	Standard Supervisory Data Access
3C	H	H	H	H	L	L	Reserved
3B	H	H	H	L	H	H	Standard Non-Privileged Block Transfer
*3A	H	H	H	L	H	L	Standard Non-Privileged Program Access
*39	H	H	H	L	L	H	Standard Non-Privileged Data Access
38	H	H	H	L	L	L	Reserved
37	H	H	L	H	H	H	Reserved
36	H	H	L	H	H	L	Reserved
35	H	H	L	H	L	H	Reserved
34	H	H	L	H	L	L	Reserved
33	H	H	L	L	H	H	Reserved
32	H	H	L	L	H	L	Reserved
31	H	H	L	L	L	H	Reserved
30	H	H	L	L	L	L	Reserved
2F	H	L	H	H	H	H	Reserved
2E	H	L	H	H	H	L	Reserved
*2D	H	L	H	H	L	H	Short Supervisory Access
2C	H	L	H	H	L	L	Reserved
2B	H	L	H	L	H	H	Reserved
2A	H	L	H	L	H	L	Reserved
*29	H	L	H	L	L	H	Short Non-Privileged Access
28	H	L	H	L	L	L	Reserved
27	H	L	L	H	H	H	Reserved
26	H	L	L	H	H	L	Reserved
25	H	L	L	H	L	H	Reserved
24	H	L	L	H	L	L	Reserved
23	H	L	L	L	H	H	Reserved
22	H	L	L	L	H	L	Reserved
21	H	L	L	L	L	H	Reserved
20	H	L	L	L	L	L	Reserved

Table 34: Address Modifier Codes (Continued)

Code	Address Modifier						Function
	5	4	3	2	1	0	
1F	L	H	H	H	H	H	User Defined
1E	L	H	H	H	H	L	User Defined
1D	L	H	H	H	L	H	User Defined
1C	L	H	H	H	L	L	User Defined
1B	L	H	H	L	H	H	User Defined
1A	L	H	H	L	H	L	User Defined
19	L	H	H	L	L	H	User Defined
18	L	H	H	L	L	L	User Defined
17	L	H	L	H	H	H	User Defined
16	L	H	L	H	H	L	User Defined
15	L	H	L	H	L	H	User Defined
14	L	H	L	H	L	L	User Defined
13	L	H	L	L	H	H	User Defined
12	L	H	L	L	H	L	User Defined
11	L	H	L	L	L	H	User Defined
10	L	H	L	L	L	L	User Defined
0F	L	L	H	H	H	H	Extended Supervisory Block Transfer
*0E	L	L	H	H	H	L	Extended Supervisory Program Access
*0D	L	L	H	H	L	H	Extended Supervisory Data Access
0C	L	L	H	H	L	L	Reserved
0B	L	L	H	L	H	H	Extended Non-Privileged Block Transfer
*0A	L	L	H	L	H	L	Extended Non-Privileged Program Access
*09	L	L	H	L	L	H	Extended Non-Privileged Data Access
08	L	L	H	L	L	L	Reserved
07	L	L	L	H	H	H	Reserved
06	L	L	L	H	H	L	Reserved
05	L	L	L	H	L	H	Reserved
04	L	L	L	H	L	L	Reserved
03	L	L	L	L	H	H	Reserved
02	L	L	L	L	H	L	Reserved
01	L	L	L	L	L	H	Reserved
00	L	L	L	L	L	L	Reserved

L = low signal level, H = high signal level

The table below lists what Address Modifier (AM) codes the CPU board will drive for each address range. Note that this table is a combination of the tables listed previously.

Table 35: Address Modifier Codes Used by the CPU Board

Addresses	Range	AM 543210	HEX	Code
$0xx0.0000_{16}$: : $F9FF.FFFF_{16}$	VMEbus (Extended Access) A32: D32, D24, D16, D8 (Shared Memory Dependent)	001110 001101 001010 001001	$0E_{16}$ $0D_{16}$ $0A_{16}$ 09_{16}	SPA[1)] SDA[2)] NPA[3)] NDA[4)]

Table 35: Address Modifier Codes Used by the CPU Board

Addresses	Range	AM 543210	HEX	Code
$FA00.0000_{16}$: : $FAFF.FFFF_{16}$	FORCE Message Broadcast Range	001101 001001	 $0D_{16}$ 09_{16}	SDA NDA
$FBFF.0000_{16}$: : $FBFE.FFFF_{16}$	VMEbus (Standard Access) A24: D32, D24, D16, D8	111110 111101 111010 111001	$3E_{16}$ $3D_{16}$ $3A_{16}$ 39_{16}	SPA SDA NPA NDA
$FBFF.0000_{16}$: : $FBFF.FFFF_{16}$	VMEbus (Short I/O Access) A16: D32, D24, D16, D8	101101 101001	$2D_{16}$ 29_{16}	SDA NDA
$FC00.0000_{16}$: : $FCFE.FFFF_{16}$	VMEbus (Standard Access) A24: D16, D8	111110 111101 111010 111001	$3E_{16}$ $3D_{16}$ $3A_{16}$ 39_{16}	SPA SDA NPA NDA
$FCFF.0000_{16}$: : $FCFF.FFFF_{16}$	VMEbus (Short I/O Access) A16: D16, D8	101101 101001	$2D_{16}$ 29_{16}	SDA NDA

1. SPA = Supervisor Program Access
2. SDA = Supervisor Data Access
3. NPA = Non-Privileged Program Access
4. NDA = Non-Privileged Data Access

3.18 VMEbus Slave Interface

3.18.1 The Access Address

The on-board shared RAM of the CPU board is also accessible from the VMEbus by another VMEbus master. Both the beginning and ending address of the Shared RAM are programmable in 4-Kbyte increments inside the FGA-002.

3.18.2 Data Transfer Size of the Shared RAM

The VMEbus slave interface for the Shared RAM is 32 bits wide. It supports 32-bit, 16-bit, and 8-bit as well as unaligned (UAT) and read-modify-write (RMW) transfers.

3.18.3 Address Modifier Decoding

For slave access to the Shared RAM from the VMEbus only Extended Address (A32) accesses are allowed.

The on-board logic allows accesses in the Privileged (supervisor) or Non-Privileged (user) mode for both data and program accesses. Each access mode can be enabled or disabled separately within the FGA-002. Thus, for example, read and write permission can be enabled for supervisor accesses, and read permission only for user accesses.

The following table shows the allowed AM Codes for VMEbus accesses to the Shared RAM.

Table 36: VMEbus Slave AM Codes

AM Code	Address Modifier						Function
	5	4	3	2	1	0	
0E	L	L	H	H	H	L	Extended Supervisory Program Access
0D	L	L	H	H	L	H	Extended Supervisory Data Access
0A	L	L	H	L	H	L	Extended Nonprivileged Program Access
09	L	L	H	L	L	H	Extended Nonprivileged Data Access

3.19 The VMEbus Interrupt Handler

All seven VMEbus interrupt request (IRQ) signals are connected to the interrupt handling logic on the FGA-002 Gate Array. Each of the VMEbus IRQ signals can be separately enabled or disabled. The FGA-002 Gate Array allows high-end multiprocessor environment board usage with distributed interrupt handling.

The FGA-002 Gate Array acts as a D08(O) interrupt handler in accordance with the VMEbus specification. It does not support 16-bit interrupt vectors.

In addition, every VMEbus interrupt request level can be mapped to cause an interrupt to the processor on a different level. So, for example, a VMEbus interrupt request on level 2 (IRQ2*) can be mapped to cause an interrupt request to the processor on level 5.

The complete VMEbus interrupt management is done inside the FGA-002.

3.19.1 VMEbus IACK Daisy Chain Driver

In accordance with the VMEbus specification, the CPU board includes an IACK daisy-chain driver. In the case of the CPU-board located in slot-1, the board acts as an IACK daisy-chain driver. Located in any other slots, the board closes the IACKIN-IACKOUT path according to the VMEbus specification.

NOTE: On the backplane the jumper for IACKIN-IACKOUT-Bypass must be removed for proper operation. This is not necessary on active backplanes.

3.20 VMEbus Arbitration

Each transfer to/from an off-board address causes a VMEbus access cycle. The VMEbus defines an arbitration mechanism to arbitrate for bus mastership. The CPU board includes a VMEbus requester, so that it may access external VMEbus resources, and a VMEbus arbiter, so that it may optionally act as "Slot-1" system controller.

3.20.1 Single-Level VMEbus Arbiter

The CPU board contains a single level arbiter which can be enabled/disabled through software. No additional control of the arbiter is required.

NOTE: In accordance with the VMEbus specification, the arbiter must be enabled if the CPU board is located in the first slot of the VMEbus backplane, and that it must be disabled if the CPU board is located in any other slot.

When the on-board single-level VMEbus arbiter is enabled, all other VMEbus masters (if any) must request VMEbus master ship using only bus request level 3 (BR3*), or they will not be recognized by the CPU board.

3.20.2 VMEbus Requester

The CPU board includes a VMEbus requester so that it may access external VMEbus resources. The level of the request may be selected by switches if the CPU-30 R4 did not detect slot-1.

SW6-3	SW6-4	Default Position	Description
OFF	OFF	x	Request level 3
OFF	ON		Request level 2
ON	OFF		Request level 1
ON	ON		Request level 0

In the case that the CPU-30 R4 detects slot-1, the request level 3 will be used automatically.

SEE ALSO: For a detailed description of the slot-1 detection please refer to Section 3.21, ‘Slot-1 Detection,’ on page 107.

NOTE: Note that the selection of the VMEbus request level has no effect upon the VMEbus arbiter located in the FGA-002 Gate Array.

3.20.3 VMEbus Release Modes

The CPU board contains several different software-selectable VMEbus release functions to relinquish VMEbus mastership. The bus release operation is independent of whether or not the on-board VMEbus arbiter is enabled and independent of the VMEbus request level. Easy handling and usage of the bus release functions is provided through the FGA-002 Gate Array. A Read-Modify-Write (RMW) cycle in progress is always completed before the bus is released. VMEPROM allows the user to change the release function through the ARB command. The modes are defined in the following subsections.

SEE ALSO: Please refer to Section 8.1, ‘ARB - Set the Arbiter of the CPU Board,’ on page 137 for details.

3.20.3.1 Release Every Cycle (REC)

The REC mode causes a release of VMEbus mastership after every VMEbus transfer cycle has been completed. A normal read or write cycle is terminated after the address and data strobes are driven high (inactive state). A Read Modify Write cycle (RMW) is terminated after the write cycle is completed by the CPU, through deactivation of the address and data strobes. If the REC mode is enabled, all other bus release functions have no impact ("don't care").

The REC mode is only for CPU cycles to the VMEbus and not for cycles initiated by the on-board DMA controller. The programming of the REC mode is described in the *FGA-002 Gate Array User's Manual*.

3.20.3.2 Release on Request (ROR)

The ROR mode is defined as a release of bus mastership if another VMEbus board has requested bus mastership while the CPU board is the current bus master. For these purposes, the on-board DMA controller can also be the requestor causing such a bus release.

The ROR mode applies only to CPU cycles to the VMEbus and not for cycles initiated by the on-board DMA controller. The ROR mode cannot be disabled, but it is programmable how long the CPU stays VMEbus master in spite of a pending bus request. Programming of the ROR mode is described in the *FGA-002 Gate Array Manual*.

3.20.3.3 Release After Timeout (RAT)

After every VMEbus access, a 100 microsecond timer within the FGA-002 begins running. Should this timer run out, the CPU board will automatically release its VMEbus mastership. The purpose of the timer is to hold the VMEbus for a short time after every VMEbus transfer, so that if the CPU makes another VMEbus request within this time period, the overhead of VMEbus arbitration will be avoided.

The timer is restarted after every VMEbus access, but not until the ROR timer has expired. Therefore, the actual time that the CPU board will hold

the bus is approximately equal to the programmed ROR delay time (see above) plus 100 microseconds. This function cannot be disabled.

The timer is only effective for CPU cycles to the VMEbus and not for cycles initiated by the on-board DMA controller. Programming of the RAT mode is described in the *FGA-002 Gate Array Manual*.

3.20.3.4 Release on Bus Clear (RBCLR)

The RBCLR function allows the VMEbus mastership release if an external arbiter asserts the BCLR* signal of the VMEbus. This function then overrides the ROR function timing limitations.

The RBCLR mode is only effective for CPU cycles to the VMEbus and not for cycles initiated by the on-board DMA controller. Programming of the RBCLR mode is described in the *FGA-002 Gate Array User's Manual*.

3.20.3.5 Release When Done (RWD)

The DMA Controller within the FGA-002 Gate Array can also become VMEbus master. It always operates in transfer bursts (maximum 32 transfers). The bus is always released after completion of such a transfer burst. The other bus release functions are for CPU mastership to the VMEbus only.

3.20.3.6 Release on ACFAIL (ACFAIL)

If the CPU board is programmed to be the ACFAIL Handler for the VMEbus system, and if the ACFAIL* signal from the VMEbus is asserted, the CPU will not release the VMEbus if it is already the VMEbus master. That is, REC, ROR, RAT, and RBCLR do not operate in this case. If the board is not ACFAIL Handler and the ACFAIL* signal is asserted, the board will release the VMEbus immediately.

3.20.3.7 Summary of Release Modes

Table 37: Bus Release Functions

Function	Enabled	VME is Released
REC ROR RAT RBCLR	Yes Always Always *1)	Every Cycle
REC ROR RAT RBCLR	No Always Always No	BRx* Active or after Timeout
REC ROR RAT RBCLR	No Always Always Yes	BRx* Active or after Timeout or BCLR* Active

1. * = "Don't Care"

3.20.4 VMEbus Grant Driver

In the case the CPU-30 R4 detects slot-1, the CPU-30 R4 will automatically use bus grant level 3 (BG3*) and drive the three remaining Bus Grant signals (BG0*, BG1*, BG2*) to a high level.

SEE ALSO: For a detailed description of the slot-1 detection please refer to Section 3.21, 'Slot-1 Detection,' on page 107.

3.21 Slot-1 Detection

The CPU-30 R4 may be used as System Controller when plugged into slot-1.

When the board is a slot-1 device, the hardware of the CPU-30 R4 setups the required system controller functions. These are:

- drive SYSCLOCK to VME
- use Bus Grant Level 3 (instead of the level selected by SW6-3 and SW6-4 as shown here)

SW6-3	OFF (default)	**SLOT, BRSEL (1:0): VME BR** SLOT-x detected, 11: 3 SW6-3 OFF = VME BRSEL bit 1=1 SLOT-x detected, 10: 2 SW6-3 ON = VME BRSEL bit 1=0 SLOT-x detected, 01: 1 SLOT-x detected, 00: 0
SW6-4	OFF (default)	SW6-4 OFF = VME BRSEL bit 0=1 SLOT-1 detected, -- : 3 SW6-4 ON = VME BRSEL bit 0 =0

- drive floating Bus Request Levels 0, 1 and 2 to a high level signal
- allow the additional Bus Timer to terminate VME cycles (timeout), if it is enabled by switch 7-4

SEE ALSO: For a description of the additional Bus Timer, please see Section 3.21.5, ‘VMEbus Timer,’ on page 109.

The System Controller functions are only enabled when the CPU-30 R4 is detected as a slot-1 device. The board’s slot-1 auto-detection mechanism probes the Bus Grant In Level 3 pin (BG3IN) from VME during power up to see whether it is possible to pull this signal down to a low signal level.

When the CPU-30 R4 is plugged into slot-1, it will succeed in pulling the VME signal to a low signal level, because Bus Grant In Level 3 is floating on slot-1. Hence, the CPU-30 R4 detects slot-1.

When the CPU-30 R4 is not plugged into slot-1, it will receive the Bus Grant In Level 3 (BG3IN) from a board plugged into a lower slot. It will fail trying to pull the VME signal to a low signal level. Hence, the CPU-30 R4 does not detect slot-1.

3.21.1 Special Slot-1 Situation

Please remember that a system begins with the highest daisy-chain priority at slot-1, the left most slot. As the slots move right they lose daisy-chain priority, so slot-2 has higher daisy-chain priority over slot-3, and slot-3 has higher daisy-chian priority over slot-4, and so on. After powering up, auto-detection may fail when another board is plugged into a slot with lower daisy-chain priority. This results in the board

(incorrectly) not driving its Bus Grant Out Level 3 (BG3OUT) on VME to the high signal level as defined by the VME specification. In this case the CPU-30 R4 will probe its Bus Grant In Level 3 at a low signal level and conclude that slot-1 is detected. This is the incorrect conclusion for the system setup which is actually there. For this situation, you can disable the auto-detection by switch SW8-1.

SW8-1	Description
OFF (default)	Auto detection enabled, slot 1 detected as a function of probing BG3IN on VME
ON	Auto detection disabled, slot-1 not detected

NOTE: For proper operation, the jumper on the backplane connecting BG3IN and BG3OUT must be removed for the CPU-30 R4 slot. This is not necessary on active backplanes.

NOTE: For proper operation, the jumpers for BGIN and BGOUT on lower (and higher) slots must be assembled on the backplane. This is not necessary on active backplanes.

3.21.2 Slot-1 Status Register

The status of the slot-1 detection may be read via the Slot-1 Status Register at FF80.3980$_{16}$. It is a read-only register.

CAUTION: Writing to the register is forbidden and may cause malfunctions of the CPU-30 R4.

Register	Access	Description
Slot-1 Status	R	Bit 0: 1 slot-1 detected 0 slot-1 not detected Bit 1-7: undefined
	W	Forbidden

3.21.3 Enabling the Arbiter

The arbiter of the FGA-002 will not automatically be set by hardware when detecting slot-1 by switch setting or auto-detection. It must be enabled by software if the board is system controller (e.g., FGA Boot enables the arbiter automatically). For more information on the FGA-002 arbiter, please see the *FGA-002 User's Manual*.

3.21.4 The SYSCLK Driver

The CPU board contains all the necessary circuitry to support the SYSCLK signal. The output signal is a stable 16 MHz signal with a 50% duty cycle. The driver circuitry for the SYSCLK signal has a current driver capacity of 64mA.

The SYSCLK signal will be enabled if the board detects slot-1.

3.21.5 VMEbus Timer

The FGA-002 Gate Array has implemented a Bus Timer to terminate VME transfers which initiates a bus error when no acknowledge can be detected after a timeout-period.

In addition to the FGA-002 Gate Array Bus Timer, the CPU-30 R4 provides an additional VMEbus Timer. This timer can be enabled when the CPU-30 R4 provides System Controller functions (slot-1 detected). The additional VMEbus Timer is controlled by switch SW7-4.

SW7-4	Description
OFF (default)	Additional VMEbus Timer enabled if slot-1 detected (otherwise disabled)
ON	Additional VMEbus Timer disabled

The timeout period for the additional VMEbus Timer can be configured with the switches SW6-1 and SW6-2.

SW6-1	SW6-2	Default Position	Description
OFF	OFF	x	Additional Bus Timer timeout = 83.53 ms
OFF	ON		Additional Bus Timer timeout = 1.3 ms
ON	OFF		Additional Bus Timer timeout = 81.6 µs
ON	ON		Additional Bus Timer timeout = 10.2 µs

3.22 Exception Signals

The VMEbus specification includes the signals SYSFAIL*, SYSRESET* and ACFAIL* for signaling exceptions or status. The SYSFAIL*, SYSRESET* and ACFAIL* signals are connected to the CPU board through buffers, switches and the FGA-002 Gate Array.

3.22.1 The SYSFAIL* Signal

The FGA-002 may be programmed to generate local interrupts when the SYSFAIL* signal is active. The VMEPROM firmware monitors the SYSFAIL* line during the initialization of external intelligent I/O boards.

The FGA-002 drives the SYSFAIL* signal after reset and until initialization of the board is completed. For compatibility with other VME boards this signal can be enabled and disabled by the switch SW8-2.

SW8-2	Description
OFF (default)	VME SYSFAIL output enabled
ON	VME SYSFAIL output disabled

3.22.2 The SYSRESET* Signal

The SYSRESET* signal from the VMEbus board will be monitored by the CPU board depending on the setting of switch SW8-4.

SW8-4	Description
OFF (default)	VME SYSRESET input monitored
ON	VME SYSRESET input not monitored

The SYSRESET* signal may be generated to the VMEbus for a number of reasons and is controlled by switch SW8-3.

SW8-3	Description
OFF (default)	VME SYSRESET output generated
ON	VME SYSRESET output not generated

A SYSRESET* is generated for any one of the following conditions:

- Active front panel RESET switch
- RESET instruction executed by the local 68030 CPU
- Access to the reset register within the FGA-002
- Optional watchdog timer from PI/T #2 expires
- Power-up condition
- Voltage monitor module detects a low voltage condition on the CPU-30 R4

3.22.3 The ACFAIL* Signal

The ACFAIL* line is ignored by VMEPROM firmware. The VMEbus requester logic in the FGA-002 monitors the ACFAIL* signal and may force a release of VMEbus mastership when it is asserted.

The CPU board can never drive the ACFAIL* signal.

3.23 Reset Generation

There is an IEEE 1014 compatible SYSRESET* driver installed on the CPU board. The reset generation circuitry operates when the power supply voltage Vcc reaches approximately 3 volts. The local reset signal will be asserted (low) for any one of the following conditions:

- Front panel RESET switch toggled
- Voltage Sensor unit detects Vcc below limit
- Execution of the RESET instruction by the local 68030
- Optional watchdog reset timer from PI/T #2 expires

An asserted reset signal will be held low (active) for at least 200 milliseconds after all the above conditions are removed.

NOTE: The VME SYSRESET Generation must be enabled (controlled by SW8-3). For further information, please refer to Section 3.22.2, ‘The SYSRESET* Signal,’ on page 110.

3.23.1 Front Panel Reset Switch

The RUN LED is red while the reset generator drives the on-board reset signal. After reset, the LED light will change to green.

The upper switch on the front panel of the CPU board is the RESET switch. Toggling it results in a reset of all on-board devices, independent of all jumper options.

The function of the reset switch is controlled by SW7-1.

SW7-1	Description
OFF	Panel Reset Switch enabled
ON	Panel Reset Switch disabled

3.23.2 The RESET Instruction

The RESET instruction of the microprocessor is designed to reset peripherals under program control, without resetting the processor itself. This instruction is fully supported by the CPU board. The RESET instruction triggers the reset generator and resets all peripherals on the board driving reset to low. At this point the processor on the CPU itself will not be reset. Therefore, program execution will go on with the next operation code. If another board asserts SYSRESET* before this instruction-triggered reset is finished, then the processor will still not be reset because of lockout logic.

3.23.3 Voltage Sensor Unit

The voltage sensor unit serves as a reset generator. Power up reset is provided by this sensor, as soon as the supply voltage Vcc has reached approximately 3 volts. The local reset signal will be asserted if Vcc subsequently falls below 4.7 volts.

The reset signal will stay asserted for at least 200 milliseconds after the supply voltage has passed the threshold.

4 Circuit Schematics and Data Sheets

4.1 Circuit Schematics of SYS68K/CPU-30 R4

Copies of the CPU-30 R4 schematics are found on the next page. The schematics contain the signal and unit cross references as well as the history of the schematics.

4.2 List of Data Sheets

This is a list of the data sheets which are relevant to the SPARC CPU-30 R4. Copies of these data sheets are found on the following pages.

RTC 72421

DUSCC 68562

PI/T TS68230

SCSI 87033/34

FDC37C65C

LANCE Am79C90

SIA Am7992B

Motorola MC68030 and MC68882

4.2.1 RTC 72421

4.2.2 DUSCC 68562

4.2.3 PI/T TS68230

4.2.4 SCSI 87033/34

4.2.5 FDC37C65C

4.2.6 LANCE Am79C90

4.2.7 SIA Am7992B

4.2.8 Motorola MC68030 and MC68882

5 VMEPROM

5.1 General Information

This CPU board operates under the control of VMEPROM, a ROM resident real-time multiuser multitasking monitor program. VMEPROM provides the user with a debugging tool for single and multitasking real-time applications. This manual describes those parts of VMEPROM which pertain to the hardware of the CPU. All general commands and system calls are described in the *VMEPROM User's Manual*.

5.2 Features of VMEPROM

- Line assembler/disassembler.
- Numerous commands for program debugging, including breakpoints, tracing, processor register display and modify.
- Display and modify floating point data registers.
- S-record up/downloading from any port defined in the system.
- Time stamping of user programs.
- Built-in Benchmarks.
- Support of RAM-disk and Winchester disks, also allowing disk formatting and initialization.
- Disk support for ISCSI-1 cards.
- Serial I/O support for up to two SIO-1/2 or ISIO-1/2 boards in the system.
- On-board Flash Memory programming utility.
- EPROM programming utility using the SYS68K/RR-2/3 boards.
- Full Screen Editor.
- Numerous commands to control the PDOS kernel and file manager.
- Complete task management.
- I/O redirection to files or ports from the command line.
- Over 100 system calls to the kernel supported.
- Data conversion and file management functions.
- Task management system calls in addition to terminal I/O functions.
- File management functions.

5.3 Power-up Sequence

After power up, the processor retrieves the initial stack pointer and program counter from address locations 0_{16} and 4_{16}. These locations are the first 8 bytes of the Boot ROM area. They are mapped down to address 0_{16} for a defined start after reset or power up. Control is transferred to the BIOS modules to perform all the necessary hardware initialization of the CPU. The real-time kernel is started and the user interface of VMEPROM is invoked as the first task. This sequence also reads the Real-Time Clock (RTC) of the CPU board and initializes the software clock of the kernel. If a terminal is connected to the terminal port of the CPU board, the VMEPROM banner and the VMEPROM prompt ("?") will be displayed upon power up or reset.

The default terminal port setup is as follows:

- Asynchronous communication
- 9600 Baud
- 8 data bits
- 1 stop bit
- No parity
- Hardware handshake protocol

If the above message does not appear, check the following:

1. Baud rate and character format setting of the terminal (default upon delivery of the CPU board is 9600 Baud, 8 data bits, 1 stop bit, no parity).
2. Cable connection from the CPU board to the terminal
3. Power supply, +5V, +12V, -12V must be present.

SEE ALSO: Refer to Section 2.5, ‘Serial I/O Channels,’ on page 21 and to Section 3.9, ‘The DUSCC 68562,’ on page 58 for the pinning of the D-Sub connector and the required handshake signals.

See Table 1, “Specifications for the CPU-30 R4 Board,” on page 7 for the power consumption of the CPU board.

If everything goes well, the header and prompt are displayed on the terminal and VMEPROM is now ready to accept commands.

5.4 Front Panel Switches

5.4.1 RESET Switch

Pressing the RESET switch on the front panel causes all programs to terminate immediately and resets the processor and all I/O devices.

When the VMEPROM kernel is started, it overwrites the first word in the user memory after the task control block with an EXIT system call. If breakpoints were defined and a user program was running when the RESET button was pressed, the user program could possibly be destroyed.

Pressing reset while a program is running should only be used as a last resort when all other actions (such as pressing ^C twice) have failed.

5.4.2 ABORT Switch

The ABORT switch is defined by VMEPROM to cause a level 7 interrupt. This interrupt cannot be disabled and is therefore the appropriate way to terminate a user program and return to the command level of VMEPROM.

If ABORT is pressed while a user program is under execution, all user registers are saved at the current location of the program counter and the message "Aborted Task" is displayed along with the contents of the processor register.

If ABORT is pressed while a built-in command is executed or the command interpreter is waiting for input, only the message is displayed and control is transferred to the command interpreter. The processor registers are not modified and are not displayed in this case.

NOTE: Tasks with port 0 as its input port will not be aborted.

5.4.3 Control Switches (Rotary Switches)

The two rotary switches on the front panel of the CPU board define the default behavior and actions taken by VMEPROM after power up or RESET.

The default definition of some of these switches can be patched in the Boot Flash ROMs for the user's convenience.

SEE ALSO: Please refer to Section 9, ‘Appendix to VMEPROM,’ on page 149 for a description of the memory locations to be patched.

The switch settings are read in by VMEPROM after reset and control various options.

The following describes the software definition for the rotary switches:

Table 38: Upper Rotary Switch (SW2)

Bit 3:	This bit indicates whether the RAM disk should be initialized after reset. If this bit is set to "0" (settings 0-7), the RAM disk is initialized as defined by bit 0 and 1. When the disk is initialized, all data on the disk is lost.
Bit 2:	This bit defines the default data size on the VMEbus. If the bit is set to "0", 16 bits are selected, if it is set to "1", 32 bits are selected.
Bit 1 and Bit 0:	These two bits define the default RAM disk. See Table 40, “RAM Disk Usage,” on page 128 for a detailed description. If AUTOBOOT is set by bit 2 and 3 of SW1, bit 1 and 0 of SW2 define which operating system will be booted. See Table 42, “Boot an Operating System (if AUTOBOOT is selected),” on page 129 for detailed description.

Table 39: Lower Rotary Switch (SW1)

Bit 3 and Bit 2:	These two bits define which program is to be invoked after reset. Please refer to Table 41, “Program After Reset,” on page 129 for a detailed description.
Bit 1:	If this switch is "0" (settings 0,1,4,5,8,9,C,D), VMEPROM tries to execute a start-up file after reset. The default filename is SY$STRT. If the bit is "1", VME-PROM comes up with the default banner.
Bit 0:	If this switch is set to "0" (settings 0,2,4,6,8,A,C,E), VMEPROM checks the VMEbus for available hardware after reset. In addition VMEPROM waits for SYSFAIL to disappear from the VMEbus. The following hardware can be detected: **Contiguous memory** **ASCU-1/2** **ISIO-1/2** **SIO-1/2** **ISCSI-1** **WFC-1** Please refer to Section 8.2, ‘CONFIG - Search VMEbus for Hardware,’ on page 138 for details.

Table 40: RAM Disk Usage

Bit 1	Bit 0	Upper Switch (SW 2)	selected on
1	1	RAM DISK AT TOP OF MEMORY (32 Kbytes)	3,7,B,F
1	0	RAM DISK AT $FC80.0000_{16}$ (512 Kbytes)	2,6,A,E
0	1	RAM DISK AT 4070.0000_{16} (512 Kbytes)	1,5,9,D
0	0	RAM DISK AT 4080.0000_{16} (512 Kbytes)	0,4,8,C

Table 41: Program After Reset

Bit 3	Bit 2	Lower Switch (SW 1)	selected on
1	1	VMEPROM	C,D,E,F
1	0	USER PROGRAM AT 4070.0000_{16}	8,9,A,B
0	1	AUTOBOOT SYSTEM	4,5,6,7
0	0	USER PROGRAM AT 4080.0000_{16}	0,1,2,3

Table 42: Boot an Operating System (if AUTOBOOT is selected)

Bit 1	Bit 0	Upper Switch (SW 2)	selected on
1	1	reserved	3,7,B,F
1	0	Boot UNIX/PDOS 4.x	2,6,A,E
0	1	Boot another operating system	1,5,9,D
0	0	Setup for UNIX mailbox driver	0,4,8,C

Table 43: Examples in Using the Rotary Switches

Rotary Switches		Description
Upper	Lower	
F	F	No RAM Disk initialization will be done. The VMEbus data size is 32 bits. The RAM Disk is on top of memory. VMEPROM will be started. No start file will be executed. The available hardware on the VMEbus will not be checked.
4	C	RAM Disk initialization will be done. The VMEbus data size is 32 bits. The RAM Disk is located at address 4080.0000_{16}. VMEPROM will be started. VMEPROM tries to execute a startup file. The available hardware on the VMEbus will be checked.
A	7	The VMEbus data size is 16 bits. AUTOBOOT System is enabled. UNIX/PDOS 4.x will be booted.

5.4.4 Default Memory Usage of VMEPROM

By default, VMEPROM uses the following memory assignment for the CPU board:

Table 44: Main Memory Layout

Start Address	End Address	Type
0000.0000_{16}	$0000.03FF_{16}$	Vector Table
0000.0400_{16}	$0000.0FFF_{16}$	System Configuration Data
0000.1000_{16}	$0000.5FFF_{16}$	SYRAM
0000.6000_{16}	$0000.6FFF_{16}$	VMEPROM internal use
0000.7000_{16}	$0000.7FFF_{16}$	Task Control Block 0
0000.8000_{16}	.....	User Memory of Task 0
.....	.....	Mail Array
.....	.....	RAM Disk (optional)
.....	End of local memory	Hashing buffers for Disk I/O

NOTE: The size of the first task cannot be extended beyond the highest on-board memory address. If more memory is available (on VMEbus), it can only be used for data storage, but not for tasking memory.

5.4.5 Default ROM Usage of VMEPROM

The following table shows the usage of the System Flash Memory when it contains VMEPROM. Note that only the first 512 Kbytes will be used by VMEPROM, all remaining space is available for user applications.

Table 45: Layout of System Flash Memory

Start Address	End Address	Type
$FF00.0000_{16}$	$FF00.0003_{16}$	Initial Supervisor Stackpointer
$FF00.0004_{16}$	$FF00.0007_{16}$	Initial Program Counter
$FF00.0008_{16}$	$FF00.000B_{16}$	Pointer to VMEPROM Initialization
$FF00.000C_{16}$	$FF00.000F_{16}$	Pointer to User Alterable Locations
$FF00.0010_{16}$	.....	Pointer to VMEPROM Shell
(Initial Program Counter)	.....	BIOS Modules, Kernel, File Manager
.....	.....	ROM resident installable devices and tables
(Pointer to Initialization)	.....	VMEPROM Initialization Code
(Ptr to alterable Locations)	.....	User Alterable Memory Locations
.....	.....	System Tools

Table 45: Layout of System Flash Memory (Continued)

(Pointer to Shell)	.....	VMEPROM Shell, System Tools, Debugging Tools, Line Assembler/ Disassembler
$FF03.6000_{16}$	$FF03.CFFF_{16}$	UNIX V.3 / PDOS Boot Program
$FF03.D000_{16}$	$FF03.FFFF_{16}$	reserved
$FF04.0000_{16}$	$FF07.FFFF_{16}$	Flash Programming Utility
$FF08.0000_{16}$	$FF3F.FFFF_{16}$	Unused System Flash Memory

6 Devices and Interrupts used by VMEPROM

6.1 Addresses of the On-board I/O Devices

The following table shows the on-board I/O devices and their addresses:

Table 46: On-board I/O Devices

Base Address	Device
$FF80.0C00_{16}$	PI/T1 68230
$FF80.0E00_{16}$	PI/T2 68230
$FF80.2000_{16}$	DUSCC1 68562
$FF80.2200_{16}$	DUSCC2 68562
$FF80.3000_{16}$	RTC 72423
$FF80.3400_{16}$	SCSI 87033
$FF80.3800_{16}$	FDC37C65C
$FFD0.0000_{16}$	FGA-002

6.2 On-board Interrupt Sources

The following table is used for the on-board interrupt sources and levels which are defined by VMEPROM. All interrupt levels and vectors of the on-board I/O devices are software programmable via the FGA-002 Gate Array.

Table 47: On-board Interrupt Sources

Device	Interrupt Level	Vector Number	Vector Address
Abort Switch	7	232 ($E8_{16}$)	$3A0_{16}$
PI/T1 (Timer Tic)	5	242 ($F2_{16}$)	$3C8_{16}$
DUSCC1	4	244 ($F4_{16}$)	$3D0_{16}$
DUSCC2	4	245 ($F5_{16}$)	$3D4_{16}$

6.3 Off-board Interrupt Sources

VMEPROM supports several VMEbus boards. As these boards are interrupt driven, the level and vectors must be defined for VMEPROM to work properly. The following table shows the default setup of the interrupt levels and vectors of the supported hardware.

SEE ALSO: For a detailed description of the hardware setup of the boards, please refer to Section 9, 'Appendix to VMEPROM,' on page 149.

The supported I/O boards together with the base addresses and the interrupt level and vector are summarized in Table 48. In order to ensure that these boards work correctly with VMEPROM, the listed interrupt vectors must not be used.

Table 48: Off-board Interrupt Sources

Board	Interrupt Level	Vector Number	Vector Address	Board Base Address
SIO-1/2	4	64-75 (40_{16}-$4B_{16}$)	100_{16}-$12C_{16}$	$FCB0.0000_{16}$
ISIO-1/2	4	76-83 ($4C_{16}$-53_{16})	130_{16}-$14C_{16}$	$FC96.0000_{16}$
WFC-1	3	119 (77_{16})	$1DC_{16}$	$FCB0.1000_{16}$
ISCSI-1	4	119 (77_{16})	$1DC_{16}$	$FCA0.0000_{16}$
ASCU-1/2	7	31 ($1F_{16}$)	$7C_{16}$	$FCB0.2000_{16}$

6.4 The On-board Real-Time Clock

During the power-up sequence, the on-board Real-Time Clock of the CPU board is read and loaded in the VMEPROM. This sequence is done automatically and requires no user intervention. If the software clock of VMEPROM is set by the ID command as described in the *VMEPROM User's Manual*, the RTC is automatically set to the new time and date values.

7 Concept of VMEPROM

7.1 Getting Started

After power up or after RESET has been pressed, VMEPROM prints a banner showing the version and revision being used and prints the prompt ("?").

If the above message does not appear, check the following:

1. Baud rate and character format setting of the terminal (default upon delivery of the CPU board is 9600 Baud, 8 data bits, 1 stop bit, no parity).
2. Cable connection from the CPU board to the terminal.
3. Power supply, +5V, +12V, -12V must be present.

SEE ALSO: Refer to Section 2.5, ‘Serial I/O Channels,’ on page 21 and to Section 3.9, ‘The DUSCC 68562,’ on page 58 for the pinning of the D-Sub connector and the required handshake signals.

See Table 1, “Specifications for the CPU-30 R4 Board,” on page 7 for the power consumption of the CPU board.

If everything goes well, the header and prompt are displayed on the terminal and VMEPROM is now ready to accept commands.

7.2 Command Line Syntax

All valid VMEPROM commands consist of the following:

```
? command<cr>
```

or

```
? command parameters<cr>
```

The underlined areas must be entered by the user. If more than one parameter will be entered, they must be separated by a space or a comma.

For a detailed description of all functions of the command interpreter please refer to chapter 3 of the *VMEPROM User's Manual*.

7.3 VMEPROM Commands

VMEPROM supports many commands. All of these commands are resident and are available at any time. Most of these commands are common for all versions of VMEPROM. All the common commands of VMEPROM are described in detail in the *VMEPROM User's Manual*. Those commands which are specific for the hardware of the CPU board are described in the following paragraphs of this manual. For a short description of one or all VMEPROM commands, the HELP command can be used. Enter 'HELP<cr>' for a description of all commands, or enter 'HELP command<cr>' for a description of a particular command.

8 Special VMEPROM Commands for CPU Boards

The following commands are implemented on the CPU board in addition to those listed in the *VMEPROM User's Manual*.

8.1 ARB - Set the Arbiter of the CPU Board

Format: ARB

The ARB command allows the user to set the arbitration and release modes of the CPU board for the VMEbus. Additionally, the VMEbus interrupts can be enabled or disabled.

Example:

```
? ARB
Set arbiter mode for VME-BUS:

STATUS :  ROR & RAT & RBCLR & FAIR
SET    :  Release on bus clear (RBCLR)       (Y/N) ? Y
SET    :  Fair VME-BUS arbitration (FAIR)    (Y/N) ? N
-----------------------------------------------------


Enable(1) / Disable(0) VMEbus interrupts by level:

STATUS :                     Level: 7 6 5 4 3 2 1
                                    1 1 1 1 1 1 1
SET    :  Enter new interrupt mask: 1 1 1 1 1 1 0

? _
```

8.2 CONFIG - Search VMEbus for Hardware

Format: CONFIG

This command searches the VMEbus for available hardware. It is useful if VMEPROM is started and bit 0 of the lower rotary switch on the front panel is set to "1", so that VMEPROM does not check the configuration by default.

In addition this command allows the user to install additional memory in the system. Additional memory can ONLY be installed with this command.

The following hardware is detected:

1. ASCU-1/2
2. ISIO-1/2
3. SIO-1/2
4. ISCSI-1
5. WFC-1
6. Contiguous memory

The board must be set to the default address for 32-bit systems.

SEE ALSO: This setup is summarized for all supported boards in Section 9, ‘Appendix to VMEPROM,’ on page 149.

Additional memory must be contiguous to the on-board memory of the CPU board. This memory is cleared by the config command to allow DRAM boards with parity to be used. Please remember that the installation of additional memory does not effect the RAM size of the running task. However, VMEPROM identifies this installed memory area and every time memory is required (e.g. with CT or FM) it is taken from this area as long as there is enough free space.

The CONFIG command also installs Winchester disks in the system and initializes the disk controller (if available). So if a SYSFAIL is active on the VMEbus (which can come for example from the ISIO-1/2 or ISCSI-1 controller during self-test), the command is suspended until the SYSFAIL signal is no longer active.

Example:

```
? CONFIG
UART FORCE ISIO-1/2 (U3) INSTALLED
ISIO-1/2:  1 boards available

? _
```

8.3 FERASE - Erase Flash Memories

Format: FERASE <flashbank>
FERASE <flashbank>,<flashoffset>,<length>

The FERASE command allows erasing Flash Memory banks.

The first format of the command erases the whole Flash Memory bank.

The second format allows specifying a region to erase.

NOTE: This must exactly match the page boundaries of the flash devices. For example, if the SYS_FLASH bank consists of four 28F008 (1 M * 8 bit) devices in parallel with a page size of 64 Kbyte each, the minimum size of one erasable region is 256 Kbyte (64 KB * 4).

The parameters are used as follows:

<flashbank> Symbolic name or base address of the Flash Memory bank that should be erased. The following symbolic names are currently supported:

BOOT_FLASH	(first) BOOT FLASH
BOOT_FLASH1	first BOOT FLASH
BOOT_FLASH2	second BOOT FLASH
SYS_FLASH	SYSTEM FLASH

<flashoffset> Optional relative byte offset within the flash bank.

<length> Optional length in bytes. If flashoffset and length are not specified, the whole bank will be erased.

Example:

```
? FERASE
Usage: FERASE <flashbank>,[<flashoffset>,<length>]
    Parameter <flashbank> is the base address of the flash bank
    or one of the following defines:
        BOOT_FLASH1     BOOT_FLASH2     SYS_FLASH1

? FERASE BOOT_FLASH2
Erasing flash memory ...   done.

? _
```

8.4 FGA - Change Boot Setup for Gate Array

Format: FGA

Some registers of the gate array are definable by the user. The contents of these registers is stored in the on-board battery SRAM in a short form.

The boot software for the gate array will take these values after reset to initialize the gate array. The FGA command may be used to enter an interactive node for changing this boot table in the batter SRAM.

The FGA command will show the actual value stored in the battery SRAM. To change any value, a new one has to be entered in binary format. If only a <CR> is entered, no change will be made. To step backward a minus has to be entered. If a <.> or <ESC> is given, the FGA command returns to the shell.

NOTE: The command uses cursor positioning codes of the selected terminal. Use the ST command to set the correct terminal.

Example:

```
? FGA

>>>   Setup for FGA-002 BOOTER   <<<

Register       FGA offset     value in SRAM  changed value

SPECIAL        $0420          %00100000      %00100000
CTL_01         $0238          %00000111      %00000111
CTL_02         $023C          %00001011      %00001011
CTL_05         $0264          %00001100      %00001100
CTL_12         $032C          %00110011      %00110011
CTL_14         $0354          %01111110      %01111110
CTL_15         $0358          %01000000      %01000000
CTL_16         $035C          %00100000      %00100000
MBX_00         $0000          %00000000      %.
MBX_01         $0004          %00000000
MBX_02         $0008          %00000000
MBX_03         $000C          %00000000
MBX_04         $0010          %00000000
MBX_05         $0014          %00000000
MBX_06         $0018          %00000000
MBX_07         $001C          %00000000

? _
```

8.5 FLUSH - Set Buffered Write Mode

Format: FLUSH
FLUSH ?
FLUSH ON
FLUSH OFF

This command flushes all modified hashing buffers for disk write or enable/disable buffered write mode for the local SCSI controller.

If no argument is entered, all modified hashing buffers are flushed. If an argument of "ON" or "OFF" is given, the buffered write mode will be enabled or disabled. By entering a question mark as an argument, only a message will be displayed, whether the buffered write mode is enabled or disabled.

Example:

```
? FLUSH

All modified buffers are flushed

? FLUSH ON

Buffered write is enabled
```

8.6 FMB - FORCE Message Broadcast

Format: FMB <slotlist>,<FMB channel>,<message>
FMB [<FMB channel>]

The FMB command allows sending a byte message to individual slots in the backplane, broadcast to all the boards, and getting a pending message.

The first format is used to send a message. With this the first parameter is used to select the slots to which a message should be sent. Each slot number can be separated with a '/' sign; a '-' defines a range of slot numbers. Slot numbers can range from 0 to 21. A slot number of 0 sends the message to all slots. The second parameter defines which FMB channel should be used. It can be '0'or '1'. The message is the byte to be deposited into the FMB channel(s).

The second format is used to get messages. If no parameter is given, one message of each FMB channel is fetched and displayed. If a channel is specified only this channel is addressed and the message will be displayed.

Example:

```
? FMB
FMB channel 0 is empty
FMB channel 1 is empty

? FMB 1-21,0,$EF

? FMB 1-21,1,%10100001

? FMB
FMB channel 0 = $EF
FMB channel 1 = $A1
? FMB 1-21,1,$77

? FMB
FMB channel 0 is empty
FMB channel 1 = $77

? FMB 1/2/5/7-19/21,0,$1

? FMB
FMB channel 0 = $01
FMB channel 1 is empty

? _
```

8.7 FPROG - Program Flash Memories

Format: FPROG <flashbank>,<source>
FPROG <flashbank>,<source>,<flashoffset>
FPROG <flashbank>,<source>,<flashoffset>,<length>

The FPROG command allows programming Flash Memory banks.

If the flash memory is not empty, it must be erased before reprogramming it (see section 8.3 “FERASE - Erase Flash Memories” on page 139).

The first format of the command programs the whole Flash Memory bank with the data stored at the specified source address.

The second format additionally allows specifying a destination offset within the Flash Memory bank and programs all the remaining space (from offset to end of flash bank).

The third format of the command also specifies the number of bytes to program.

The parameters are used as follows:

<flashbank> Symbolic name or base address of the Flash Memory bank that should be programmed. The following symbolic names are currently supported:

BOOT_FLASH	(first) BOOT FLASH
BOOT_FLASH1	first BOOT FLASH
BOOT_FLASH2	second BOOT FLASH
SYS_FLASH	SYSTEM FLASH

<source> Source address of the data to program.

<flashoffset> Optional relative byte offset within the flash bank. If no offset is specified, 0 is assumed.

<length> Optional length in bytes. If no length is specified, all the remaining space of the flash bank will be programmed.

Example: Partly programming the second Boot Flash

```
? FPROG BOOT_FLASH2,100000,0,1375
Programming flash memory
  0 |################################################| 100%
Done.
? _
```

8.8 FUNCTIONAL - Perform Functional Test

Format: FUNCTIONAL

NOTE: This command is not designed for the user, but instead for internal purposes by FORCE COMPUTERS.

8.9 MEM - Set Data Bus Width of the VMEbus

Format: MEM
MEM 16
MEM 32

This command can display or set the data bus width of the CPU board on the VMEbus.

If no argument is entered, the current data bus width is displayed. If an argument of '16' or '32' is given, the data bus width is set to 16 or 32 bits respectively.

Example:

```
? MEM
Data bus width is set to 32 bits

? MEM 16

? MEM
Data bus width is set to 16 bits

? MEM 32

? MEM
Data bus width is set to 32 bits

? _
```

8.10 SELFTEST - Perform On-board Selftest

Format: SELFTEST

This command performs a test of the on-board functions of the CPU board. It may only run if no other tasks are created. If there are any other tasks, no selftest will be made and an error will be reported. The selftest tests the memory of the CPU board and all devices on the board.

The following tests are performed in this order:

1. **I/O Test**

   This function tests the access to and the interrupts from the DUSCC. If the DUSCC cannot generate interrupts, an error will be reported. This test also checks if reading from and writing to the floppy disk controller and the SCSI controller proceeds as expected.

2. **Memory Test on the Memory of the Current Task**

   The following procedures are performed:

   1) Byte Test
   2) Word Test
   3) Longword Test

   All passes of the memory test perform pattern reading and writing as well as bit shift tests. If an error occurs while writing to or reading from memory, it will be reported. Dependent on the size of the main memory, this test may last a different amount of time (count about one minute per megabyte).

3. **Clock Test**

   If the CPU does not receive timer interrupts from the PI/T 68230, an error will be displayed. This ensures that VMEPROM could initialize the PI/T 68230 properly and the interrupts from the PI/T are working

CAUTION: During this process, all memory is cleared.

Example:

```
? SELFTEST
VMEPROM Hardware Selftest
-------------------------
I/O test ........ passed
Memory test ..... passed
Clock test ...... passed

? _
```

8.11 Installing a New Hard Disk

The FRMT command of VMEPROM may be used to set all hard disk parameters, to format the Winchester and to divide the disk into logical partitions. Before starting the FRMT command, the number of the last logical block of the Winchester must be known. The number of physical blocks per track must be 32, the number of bytes per sector must be 256. By using the following equation:

(# of Heads) * (# of Cylinders) * (Blocks/Track) = # of Last logical block

the number of Heads and the number of Cylinders may be calculated.

NOTE: The maximum number of Heads is 16. The number of large and floppy partitions is definable by the user.

The following example aids in formatting a CDC 94211-5 Winchester.

```
? FRMT
68K PDOS Force Disk Format Utility
  Possible Disk Controllers in this System are:
     Controller #1 is not defined
     Controller #2 is a FORCE WFC-1
     Controller #3 is a FORCE ISCSI-1
     Controller #4 is an onboard SCSI
     Controller #5 is not defined
     Controller #6 is a FORCE IBC
  Drives that are currently defined in system are:
     F0  is controller #4 , drive select $82
     F1  is controller #4 , drive select $83
     W0  is controller #4 , drive select $00

     All not named drives are undefined

  Select Menu: W,W0-W15=Winch; F,F0-F8=Floppy; Q=Quit
    Select Drive: W
  W0 Main Menu: 1)Parm 2)BadT 3)Form 4)Veri 5)Part 6)Writ
                P)Togl Q)Quit
    Command: 1

  W0 Parameters Menu: A)lter, D)isplay, R)ead file, Q)uit
    Command: A
                 # of Heads = 10
             # of Cylinders = 1022
  Physical Blocks per Track = 32
   Physical Bytes per Block = 256
          Shipping Cylinder = 0
                  Step rate = 0
  Reduced write current cyl = 0
    Write Precompensate cyl = 0
```

```
Current Winch Drive 0 Parameters:
              # of Heads = 10
          # of Cylinders = 1022
Physical Blocks per Track = 32
 Physical Bytes per Block = 256
        Shipping Cylinder = 0
                Step rate = 0
Reduced write current cyl = 0
  Write Precompensate cyl = 0`

W0 Parameters Menu: A)lter, D)isplay, R)ead file, Q)uit
  Command: Q
W0 Main Menu: 1)Parm 2)BadT 3)Form 4)Veri 5)Part 6)Writ
              P)Togl Q)Quit
Command: 3
Sector Interleave = 0
Physical Tracks to FORMAT = 0,10219
Ready to FORMAT Winchester Drive 0 ? Y
Sector Interleave Table:  0,1,2,3,4,5,6,7,8,9,10,11,12,
                          13,14,15,16,17,18,19,20,21,22,
                          23,24,25,26,27,28,29,30,31

Issuing Format Drive Command.
FORMAT SUCCESSFUL !


W0 Main Menu: 1)Parm 2)BadT 3)Form 4)Veri 5)Part 6)Writ
              P)Togl Q)Quit
  Command: 5
W0 Partitions Menu: A)lter, D)isplay, R)ecalc, Q)uit
  Command: A
    # of Large partitions = 6
   # of Floppy Partitions = 15
First track for PDOS Parts = 0
 Last track for PDOS Parts = 10219
        First PDOS disk # = 2

Current Winch Drive 0 Partitions:
    # of Large partitions = 6
   # of Floppy Partitions = 15
First track for PDOS Parts = 0
 Last track for PDOS Parts = 10219
        First PDOS disk # = 2
 Total # of Logical Tracks = 10220
```

```
 Disk #    Logical Trks  Physical Trks  PDOS sectors
            Base,Top      Base,Top       Total/{boot}
     2      0,1502        0,1502         48064/47872
     3      1503,3005     1503,3005      48064/47872
     4      3006,4508     3006,4508      48064/47872
     5      4509,6011     4509,6011      48064/47872
     6      6012,7514     6012,7514      48064/47872
     7      7515,9017     7515,9017      48064/47872
     9      9018,9097     9018,9097      2528/2336
     10     9098,9177     9098,9177      2528/2336
     11     9178,9257     9178,9257      2528/2336
     12     9258,9337     9258,9337      2528/2336
     13     9338,9417     9338,9417      2528/2336
     14     9418,9497     9418,9497      2528/2336
     15     9498,9577     9498,9577      2528/2336
     16     9578,9657     9578,9657      2528/2336
     17     9658,9737     9658,9737      2528/2336
     18     9738,9817     9738,9817      2528/2336
     19     9818,9897     9818,9897      2528/2336
     20     9898,9977     9898,9977      2528/2336
     21     9978,10057    9978,10057     2528/2336
     22     10058,10137   10058,10137    2528/2336
     23     10138,10217   10138,10217    2528/2336

  W0 Partitions Menu: A)lter, D)isplay, R)ecalc, Q)uit
    Command: Q

  W0 Main Menu: 1)Parm 2)BadT 3)Form 4)Veri 5)Part 6)Writ
                P)Togl Q)Quit
     Command: 6
  Write to Disk Y)es, N)o, F)ile : Y
  Write to file (Y/N)?N
  W0 Main Menu: 1)Parm 2)BadT 3)Form 4)Veri 5)Part 6)Writ
                P)Togl Q)Quit
     Command: Q
  Exit to Select Drive.  Update Param RAM (Y/N) ? Y
  System Parameter RAM Updated!!
  Select Menu: W,W0-W15=Winch; F,F0-F8=Floppy; Q=Quit
     Select Drive: Q
```

After formatting the disk, all logical partitions must be initialized using the INIT command.

The example below may be used to initialize the large logical partition number two.

```
? INIT
Enter Disk # :2
Directory Entries :1024
Number of sectors :47776
Disk Name :SYSTEM
Init:  Disk # 2
       Directory entries: 1024
       Number of sectors: 47776
       Disk name: SYSTEM
Initialize disk ? Y

? _
```

9 Appendix to VMEPROM

9.1 Driver Installation

This appendix summarizes the changes to be made to the default setup of additional VMEbus boards so that they are VMEPROM compatible. Drivers described in Appendices 9 through 9.1.6 are available in the Boot ROM, but are not installed. All drivers may be installed with the INSTALL command. When INSTALL followed by a question mark is entered, the following will appear:[1)]

```
? INSTALL ?
THE FOLLOWING UARTS AND DISK DRIVER ARE ALREADY IN EPROM:

UART DRIVER    FORCE CPU-30/DUSCC       ADDR: $FF004800
UART DRIVER    FORCE SIO-1/2            ADDR: $FF004B00
UART DRIVER    FORCE ISIO-1/2           ADDR: $FF004F00
UART DRIVER    FORCE IBC/ME             ADDR: $FF005400
UART DRIVER    FORCE UNIX MAIL          ADDR: $FF007200
DISK DRIVER    FORCE SCSI CPU-30        ADDR: $FF007A00
DISK DRIVER    FORCE ISCSI-1            ADDR: $FF009700
DISK DRIVER    FORCE IBC/ME             ADDR: $FF009F00
DISK DRIVER    FORCE WFC-1              ADDR: $FF00D100
```

9.1.1 VMEbus Memory

In general, every FORCE memory board can be used together with VMEPROM.

The base address must be set correctly in order to use the board within the tasking memory of VMEPROM. That is, the board base addresses of any additional memory boards must be set to be contiguous to the on-board memory.

9.1.2 SYS68K/SIO-1/2

These two serial I/O boards are set to the VME base address $B0.0000_{16}$ by default.

VMEPROM expects the first SIO-1/SIO-2 boards at $FCB0.0000_{16}$. This is in the standard VME address range (A24, D16, D8) with the address $B0.0000_{16}$. The address modifier decoder (AM-Decoder) of the SIO-1/2 boards must be set to:

Standard Privileged Data Access

Standard Nonprivileged Data Access

1. Please note that the printed UART and Disk Driver addresses are only examples. They may alternate according to software versions.

Please refer to the *SIO User's Manual* for setup. If a second SIO-1/2 board will be used, the base address must be set to $FCB0.0200_{16}$. The AM-decoder setup described above must again be used. Please refer to the *SIO User's Manual* for the address setup of the second SIO board. Before using the driver for the SIO-1/2 board, the driver must be installed by using the INSTALL command. The following must be entered:

```
? INSTALL U2,$FF004B00
```

In order to install one of the ports of the SIO boards in VMEPROM, the BP command can be used. The SIO-1/2 boards use the driver type 2. To install the first port of a SIO board with a 9600 baud rate, the following command line can be used:

```
? BP 5,9600,2,$FCB00000
```

The port can then be used as port number 5.

NOTE: The hardware configuration must be detected before a port can be installed. This can be done with the CONFIG command or by setting a front panel switch on the CPU Board and pressing RESET. Please refer to the command description in the *VMEPROM User's Manual* for a detailed description of the CONFIG and BP commands.

The base addresses of all ports of a SIO-1/2 board which must be specified with the BP command are as follows:

SIO Port #	Address
1 (first SIO board)	$FCB0.0000_{16}$
2	$FCB0.0040_{16}$
3	$FCB0.0080_{16}$
4	$FCB0.00C0_{16}$
5	$FCB0.0100_{16}$
6	$FCB0.0140_{16}$
1 (second SIO board)	$FCB0.0200_{16}$
2	$FCB0.0240_{16}$
3	$FCB0.0280_{16}$
4	$FCB0.02C0_{16}$
5	$FCB0.0300_{16}$
6	$FCB0.0340_{16}$

VMEPROM supports up to two serial I/O boards. These can be either the SIO-1/2 board, the ISIO-1/2 board, or a mixture of both.

NOTE: The first board of every type must be set to the first base address. In using one SIO-1 board and one ISIO-1 board, the base address of the boards must to be set to:

SIO-1	$FCB0.0000_{16}$
ISIO-1	$FC96.0000_{16}$

9.1.3 SYS68K/ISIO-1/2

These serial I/O boards are set to the address 96.0000_{16} in the standard VME address range by default. VMEPROM awaits this board at this address ($FC96.0000_{16}$ for the CPU-30 R4); no changes need to be made to the default setup. An optional second board may be used. When used, the address must be set to 98.0000_{16}. Read the *SYS68K/ISIO-1/2 User's Manual* for a description of the base address setup. Before using the driver for the ISIO-1/2 board, the driver must be installed by using the INSTALL command. The following must be entered:

```
? INSTALL U3,$FF004F00
```

In order to install one of the ports of an ISIO board in VMEPROM, the BP command can be used. The ISIO-1/2 boards are driver type 3. In order to install the first port of an ISIO board with a 9600 baud rate, the following command line can be used:

```
? BP 5,9600,3,$FC968000
```

The port number is five. The hardware configuration must be detected before a port can be installed. This is done with the CONFIG command, or by setting a front switch on the CPU board and pressing RESET. Read the command description in the *VMEPROM User's Manual* for a description of the CONFIG and BP commands. The base address of all ISIO-1/2 ports, specified by the BP command, is as follows:

ISIO Port #	Address
1 (first ISIO board)	$FC96.8000_{16}$
2	$FC96.8020_{16}$
3	$FC96.8040_{16}$
4	$FC96.8060_{16}$
5	$FC96.8080_{16}$
6	$FC96.80A0_{16}$
7	$FC96.80C0_{16}$
8	$FC96.80E0_{16}$
1 (second ISIO board)	$FC98.8000_{16}$
2	$FC98.8020_{16}$
3	$FC98.8040_{16}$
4	$FC98.8060_{16}$
5	$FC98.8080_{16}$
6	$FC98.80A0_{16}$
7	$FC98.80C0_{16}$
8	$FC98.80E0_{16}$

VMEPROM supports two serial I/O boards. These can be the SIO-1/2 or ISIO-1/2 board or a mixture of both. The first board of each type must be set to the first base address. When using one SIO-1 and one ISIO-1 board, the base address of the boards must be set to:

SIO-1	$FCB0.0000_{16}$
ISIO-1	$FC96.0000_{16}$

9.1.4 SYS68K/WFC-1 Disk Controller

VMEPROM supports up to two floppy disk drives and three Winchester disk drives together with the WFC-1 disk controller.

The floppy drives must be jumpered to drive select 3 and 4 and can be accessed as disk number 0 and 1 out of VMEPROM. The floppy drives are installed automatically when a WFC-1 controller is detected by the CONFIG command or after reset when the front panel switch of the CPU board is set to detect the hardware configuration. Usable floppy drives must support 80 tracks/side, double sided and double density. The step rate is 3 ms.

The Winchester drives are not installed automatically. The VMEPROM FRMT command must be used for defining the following factors:

- The physical drive structure (i.e. number of heads, number of cylinders, drive select number, etc.)
- The bad block of the Winchester drive
- The partitions to be used

If this setup is done once for a particular drive, the data is stored in the first sector of the Winchester and is loaded automatically when the disk controller is installed in VMEPROM. The driver for the WFC-1 may be installed by using the INSTALL command. The following must be entered:

```
? INSTALL W,$FF00D100
```

The default base address of the WFC-1 controller must be set to $FCB0.1000_{16}$. To do so, the address comparison for 32-bit address has to be enabled and the setup of the most significant 8 addresses must be jumpered.

VMEPROM supports termination interrupt of the WFC-1 controller. If you want to use the WFC-1 with interrupts, the corresponding jumper must be set to enable the interrupt.

Please refer to the data sheet of the WFC-1 controller for a detailed description of the address setup and termination interrupt.

9.1.5 SYS68K/ISCSI-1 Disk Controller

VMEPROM supports up to two floppy disk drives and three Winchester disk drives together with the ISCSI-1 disk controller. The floppy drives must be jumpered to drive select 3 and 4 and can be accessed as disk number 0 and 1 out of VMEPROM. The floppy drives are installed automatically when an ISCSI-1 controller is detected by the CONFIG command or after pressing RESET when the front panel switch of the CPU board is set to detect the hardware configuration. Usable floppy drives must support 80 tracks/side, and must be double-sided/double density. The step rate used is 3 ms. The Winchester drives are not installed automatically. The VMEPROM FRMT command must be used for defining the following factors:

- The physical structure of the drive (i.e., number of heads, number of cylinders, drive select number, etc.)
- The bad block of the Winchester drive
- The partitions to be used

If this setup is done once for a particular drive, the data is stored in the first sector of the Winchester and is loaded automatically when the disk controller is installed in VMEPROM. The driver for the ISCSI-1 may be installed by using the INSTALL command. The following must be entered:

```
? INSTALL W,$FF009700
```

The default base address of the ISCSI-1 controller is $A0.0000_{16}$ in the standard VME address range. This is the address $FCA0.0000_{16}$ for the CPU board and no changes have to be made to this setup. The ISCSI-1 driver uses interrupts by default. This cannot be disabled. Please make sure that the interrupt daisy chain is closed so that the controller can work properly.

9.1.6 Local SCSI Controller

VMEPROM supports up to three Winchester disk drives together with the local SCSI Controller.

The Winchester drives are not installed automatically.

The VMEPROM FRMT command must be used for defining the following factors:

- The physical structure of the drive (i.e., number of heads, number of cylinders, drive select number, etc.)
- The bad blocks of the Winchester drive
- The partitions to be used

If this setup is done once for a particular drive, the data is stored in the first sector of the Winchester and is loaded automatically when the disk controller is installed in VMEPROM. Upon viewing the VMEPROM Banner, the driver for the local SCSI controller is already installed. For this driver, memory is needed for hashing. The storage for the hashing buffers is allocated at the end of memory.

9.2 S-Record Formats

9.2.1 S-Record Types

Eight types of S-records have been defined to accommodate the needs of the encoding, transportation and decoding functions. VMEPROM supports S0, S1, S2, S3, S7, S8 and S9 records (S7 and S8 on load only).

An S-record format module may contain S-records of the following types:

S0 The header record for each block of S-records.

S1 A record containing code/data and the 2-byte address at which the code/data is to reside.

S2 A record containing code/data and the 3-byte address at which the code/data is to reside.

S3 A record containing code/data and the 4-byte address at which the code/data is to reside.

S5 A record containing the number of S1, S2 and S3 records transmitted in a particular block. The count appears in the address field. There is no code/data field. Not supported by VMEPROM.

S7 A termination record for a block of S3 records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.

S8 A termination record for a block of S2 records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.

S9 A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed.

Only one termination record is used for each block of S-records. S7 and S8 records are usually used only when control is to be passed to a 3- or 4-byte address. Normally, only one header record is used, although it is possible for multiple header records to occur.

Example:

```
S21402000000000444000201466000OCB241F8044CB1
S214020010203C0000020E428110C1538066FA487AE4
S214020020001021DF0008487A001221DF000C4E750E
S21402003021FC42555320003060082lFC41444452C2


                                          XX-  Check-sum
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX---  Data
    0200XX------------------------------------  24-bit Address
  14------------------------------------------  Byte Count
S2------------------------------------------  Record Type



S9030000FC

        FC------------------------------------  Check-sum
    0000------------------------------------  Data
  03------------------------------------------  Byte Count
S9------------------------------------------  Record Type
```

9.3 System RAM Definitions

```
/* SYRAM:H -- DEFINITION OF SYRAM BLOCK OF MEMORY
   05-Jan-88  Revised to correspond to PDOS 3.3
   BRIAN C. COOPER, EYRING RESEARCH INSTITUTE, INC.
   Copyright 1985-1988
*/
#define  NT   64                  /* number of tasks                   */
#define  NM   ((NT+3)&0xFC)       /* number of task messages           */
#define  NP   16                  /* number of task message pointers   */
#define  ND   ((NT+3)&0xFC)       /* number of delay events            */
#define  NC   8                   /* number of active channel buffers  */
#define  NF   64                  /* number of file slots              */
#define  NU   15                  /* number of I/O UART ports          */
#define  IZ   6                   /* input buffer size (2^p2p.         */
#define  MZ   0x4000000           /* maximum memory size               */
#define  TZ   64                  /* task message size                 */

#define  NTB  NT
#define  NTM  NM
#define  NTP  NP
#define  NCB  NC
#define  NFS  NF
#define  NEV  ND
#define  NIE  (ND/2)
#define  NPS  (NU+1)
#define  P2P  IZ
#define  MMZ  MZ
#define  TMZ  TZ

#define  IMK  (0xFF>>(8-P2P))     /* input buffer wrap around mask     */
#define  NCP  ((1<<P2P)+2)        /* (# characters/port) + 2           */
#define  MPZ  2048                /* memory page size                  */
#define  MBZ  (MMZ/MPZ)           /* memory bitmap size                */
#define  NMB  (MBZ/8)             /* number of map bytes               */
#define  FSS  38                  /* file slot size                    */
#define  TQB  2                   /* TCB index                         */
#define  TQM  (TQB+4)             /* map index                         */
#define  TQE  (TQM+2)             /* event #1 / event #2               */
#define  TQS  (TQE+2)             /* scheduled event                   */
#define  TBZ  (TQS+2+4)           /* TASK entry size                   */
#define  BPS  256                 /* bytes per sector                  */
#define  NRD  4                   /* number of RAM disks               */

struct SYRAM{
/*000*/ char *_bios;              /* address of bios rom               */
/*004*/ char *_mail;              /* *mail array address               */
/*008*/ unsigned int _rdkn;       /* *ram disk #                       */
/*00A*/ unsigned int _rdks;       /* *ram disk size                    */
/*00C*/ char *_rdka;              /* *ram disk address                 */
/*010*/ char _bflg;               /* basic present flag                */
/*011*/ char _dflg;               /* directory flag                    */
/*012*/ int  _f681;               /* 68000/68010 flag                  */
```

```
/*014*/ char *_sram;              /* run module B$SRAM                  */
/*018*/ int spare1;               /* reserved for expansion             */
/*01A*/ int  _fcnt;               /* fine counter                       */
/*01C*/ long _tics;               /* 32 bit counter                     */
/*020*/ unsigned char _smon;      /* month                              */
/*021*/ unsigned char _sday;      /* day                                */
/*022*/ unsigned char _syrs[2];   /* year                               */
/*024*/ unsigned char _shrs;      /* hours                              */
/*025*/ unsigned char _smin;      /* minutes                            */
/*026*/ unsigned char _ssec[2];   /* seconds                            */
/*028*/ char _patb[16];           /* input port allocation table        */
/*038*/ char _brkf[16];           /* input break flags                  */
/*048*/ char _f8bt[16];           /* port flag bits                     */
/*058*/ char _utyp[16];           /* port uart type                     */
/*068*/ char _urat[16];           /* port rate table                    */
/*078*/ char _evtb[10];           /* 0-79 event table                   */
/*082*/ char _evto[2];            /* 80-95 output events                */
/*084*/ char _evti[2];            /* 96-111 input events                */
/*086*/ char _evts[2];            /* 112-127 system events              */
/*088*/ char _ev128[16];          /* task 128 events                    */
/*098*/ long _evtm[4];            /* events 112-115 timers              */
/*0A8*/ long _bclk;               /* clock adjust constant              */
/*0AC*/ char *_tltp;              /* task list pointer                  */
/*0B0*/ char *_utcb;              /* user tcb ptr                       */
/*0B4*/ int  _suim;               /* supervisor interrupt mask          */
/*0B6*/ int  _usim;               /* user interrupt mask                */
/*0B8*/ char _sptn;               /* spawn task no. (** must be even **)*/
/*0B9*/ char _utim;               /* user task time                     */
/*0BA*/ char _tpry;               /* task priority (** must be even **) */
/*0BB*/ char _tskn;               /* current task number                */
/*0BC*/ char spare2;              /* reserved                           */
/*0BD*/ char _tqux;               /* task queue offset flag/no          */
/*0BE*/ char _tlck[2];            /* task lock/reschedule flags         */
/*0C0*/ char _e122;               /* batch task #                       */
/*0C1*/ char _e123;               /* spooler task #                     */
/*0C2*/ char _e124;
/*0C3*/ char _e125;
/*0C4*/ long _cksm;               /* system checksum                    */
/*0C8*/ int  _pnod;               /* pnet node #                        */
/*0CA*/ char bser[6];             /* bus error vector                   */
/*0D0*/ char iler[6];             /* illegal vector                     */
/*0D6*/ char ccnt[16];            /* control C count                    */
/*0E6*/ char *_wind;              /* window id's                        */
/*0EA*/ char *_wadr;              /* window addresses                   */
/*0EE*/ char *_chin;              /* input stream                       */
/*0F2*/ char *_chot;              /* output stream                      */
/*0F6*/ char *_iord;              /* i/o redirect                       */
/*0FA*/ char _fect;               /* file expand count                  */
/*0FB*/ char _pidn;               /* processor ident byte               */
/*0FC*/ long *_begn;              /* abs addr of K1$BEGN table          */
/*100*/ int  _rwcl[14];           /* port row/col 1..15                 */
/*11C*/ char *_opip[15];          /* output port pointers 1..15         */
/*158*/ char *_uart[16];          /* uart base addresses 1..15          */
/*198*/ long _mapb;               /* memory map bias                    */
```

```
/*                                                                    */
/*  the following change with different configurations:               */
/*  configuration for VMEPROM is defined to:                          */
/*    NT = 64, NF = 64, MZ = $400000                                  */
/*                                                                    */
/*  NOTE: the offset on top of each line is calculated only for this  */
/*        configuration                                               */
/*                                                                    */
/*019C*/ char _maps[NMB];          /* system memory bitmap            */
/*119C*/ char _port[(NPS-1)*NCP];  /* character input buffers         */
/*157A*/ char _iout[(NPS-1)*NCP];  /* character output buffers        */
/*1958*/ char rdtb[16];            /* redirect table                  */
/*1968*/ int  _tque[NTB+1];        /* task queue                      */
/*19EA*/ char _tlst[NTB*TBZ];      /* task list                       */
/*1DEA*/ char _tsev[NTB*32];       /* task schedule event table       */
/*25EA*/ long _tmtf[NTM];          /* to/from/INDEX.W                 */
/*26EA*/ char _tmbf[TMZ*NTM];      /* task message buffers            */
/*36EA*/ char _tmsp[NTP*6];        /* task message pointers           */
/*374A*/ char _deiq[2+8+NIE*10];   /* delay event insert queue        */
/*3894*/ char _devt[2+NEV*10];     /* delay events                    */
/*3B16*/ int  _bsct[32];           /* basic screen command table      */
/*3B56*/ int  _xchi[NCB];          /* channel buffer queue            */
/*3B66*/ char _xchb[NCB*BPS];      /* channel buffers                 */
/*4366*/ char _xfsl[NFS*FSS];      /* file slots                      */
/*4CE6*/ char _l2lk;               /* level 2 lock (file prims, evnt 120)*/
/*4CE7*/ char _l3lk;               /* level 3 lock (disk prims, evnt 121)*/
/*4CE8*/ long _drvl;               /* driver link list entry point    */
/*4CEC*/ long _utll;               /* utility link list entry point   */
/*4CF0*/ int  _rdkl[NRD*4 + 1];    /* RAM disk list                   */
};
```

9.4 Task Control Block Definitions

```
#define MAXARG     10                   /* max argument count of the cmd line        */
#define MAXBP      10                   /* max 10 breakpoints                        */
#define MAXNAME     5                   /* max 5 names in name buffer                */
#define TMAX       64                   /* Max number of tasks                       */
#define ARGLEN     20                   /* maximum argument length                   */

/* special system flags for VMEPROM                                                  */

#define SOMEREG 0x0001                  /* display only PC,A7,A6,A5                  */
#define T_DISP  0x0002                  /* no register display during trace(TC>1)    */
#define T_SUB   0x0004                  /* trace over subroutine set                 */
#define T_ASUB  0x0008                  /* trace over subroutine active              */
#define T_RANG  0x0010                  /* trace over range set                      */
#define REG_INI 0x0020                  /* no register initialization if set         */
#define RE_DIR  0x0040                  /* output redirection into file and          */
                                        /* console at the same time                  */

/* the registers are stored in the following order:                                  */
#define VBR     0
#define SFC     1
#define DFC     2
#define CACR    4
#define PC      5
#define SR      6
#define USTACK  7
#define SSTACK  8
#define MSTACK  9
#define D0      10                      /*  10-17  =  D0-D7                          */
#define A0      18                      /*  18-24  =  A0-A6                          */

#define N_REGS  25

#define BYTE    unsigned char
#define WORD    unsigned int
#define LWORD   unsigned long

struct TCB{

/*000*/ char _ubuf[256];                /* 256 byte user buffer                      */
/*100*/ char _clb[80];                  /* 80 byte monitor command line buffer       */
/*150*/ char _mwb[32];                  /* 32 byte monitor parameter buffer          */
/*170*/ char _mpb[60];                  /* monitor parameter buffer                  */
/*1AC*/ char _cob[8];                   /* character out buffer                      */
/*1B4*/ char _swb[508];                 /* system work buffer/task pdos stack        */
/*3B0*/ char *_tsp;                     /* task stack pointer                        */
/*3B4*/ char *_kil;                     /* kill self pointer                         */
/*3B8*/ long _sfp;                      /* RESERVED FOR INTERNAL PDOS USE            */
/*3BC*/ char _svf;                      /* save flag -- 68881 support (x881)         */
/*3BD*/ char _iff;                      /* RESERVED FOR INTERNAL PDOS USE            */
/*3BE*/ long _trp[16];                  /* user TRAP vectors                         */
/*3FE*/ long _zdv;                      /* zero divide trap                          */
/*402*/ long _chk;                      /* CHCK instruction trap                     */
/*406*/ long _trv;                      /* TRAPV Instruction trap                    */
```

```
/*40A*/ long _trc;                      /* trace vector                              */
/*40E*/ long _fpa[2];                   /* floating point accumulator                */
/*416*/ long *_fpe;                     /* fp error processor address                */
/*41A*/ char *_clp;                     /* command line pointer                      */
/*41E*/ char *_bum;                     /* beginning of user memory                  */
/*422*/ char *_eum;                     /* end user memory                           */
/*426*/ char *_ead;                     /* entry address                             */
/*42A*/ char *_imp;                     /* internal memory pointer                   */
/*42E*/ int  _aci;                      /* assigned input file ID                    */
/*430*/ int  _aci2;                     /* assigned input file ID's                  */
/*432*/ int  _len;                      /* last error number                         */
/*434*/ int  _sfi;                      /* spool file id                             */
/*436*/ BYTE _flg;                      /* task flags (bit 8=command line echo)      */
/*437*/ BYTE _slv;                      /* directory level                           */
/*438*/ char _fec;                      /* file expansion count                      */
/*439*/ char  _spare1;                  /* reserved for future use                   */
/*43A*/ char _csc[2];                   /* clear screen characters                   */
/*43C*/ char _psc[2];                   /* position cursor characters                */
/*43E*/ char _sds[3];                   /* alternate system disks                    */
/*441*/ BYTE _sdk;                      /* system disk                               */
/*442*/ char *_ext;                     /* XEXT address                              */
/*446*/ char *_err;                     /* XERR address                              */
/*44A*/ char _cmd;                      /* command line delimiter                    */
/*44B*/ BYTE _tid;                      /* task id                                   */
/*44C*/ char _ecf;                      /* echo flag                                 */
/*44D*/ char _cnt;                      /* output column counter                     */
/*44E*/ char _mmf;                      /* memory modified flag                      */
/*44F*/ char _prt;                      /* input port #                              */
/*450*/ char _spu;                      /* spooling unit mask                        */
/*451*/ BYTE _unt;                      /* output unit mask                          */
/*452*/ char _u1p;                      /* unit 1 port #                             */
/*453*/ char _u2p;                      /* unit 2 port #                             */
/*454*/ char _u4p;                      /* unit 4 port #                             */
/*455*/ char _u8p;                      /* unit 8 port #                             */
/*456*/ char _spare2[26];               /* reserved for system use                   */

/*****************************************************************************/
/*      VMEPROM variable area area                                           */
/*****************************************************************************/

/*470*/ char  linebuf[82];              /* command line buffer                       */
/*4C2*/ char  alinebuf[82];             /* alternate line buffer                     */
/*514*/ char  cmdline[82];              /* alternate cmdline for XGNP                */
/*566*/ int   allargs, gotargs;         /* argc save and count for XGNP              */
/*56A*/ int   argc;                     /* argument counter                          */
/*56C*/ char  *argv[MAXARG];            /* pointer to arguments of the cmd line      */
/*594*/ char  *odir, *idir;             /* I/O redirection args from cmd line        */
/*59C*/ int   iport,oport;              /* I/O port assignments                      */
/*5A0*/ char  *ladr;                    /* holds pointer to line in_mwb              */
/*5A4*/ LWORD offset;                   /* base memory pointer                       */
/*5A8*/ int   bpcnt;                    /* num of defined breakpoints                */
/*5AA*/ LWORD bpadr[MAXBP];             /* breakpoint address                        */
/*5D2*/ WORD  bpinst[MAXBP];            /* breakpoint instruction                    */
/*5E6*/ char  bpcmd[MAXBP][11];         /* breakpoint command                        */
```

```
/*654*/ WORD  bpocc[MAXBP];             /* # of times the breakpoint should be       */
                                        /* skipped                                   */
/*668*/ WORD  bpcocc[MAXBP];            /* # of times the breakpoint is already      */
                                        /* skipped                                   */
/*67C*/ LWORD bptadr;                   /* temp. breakpoint address                  */
/*680*/ WORD  bptinst;                  /* temp. breakpoint instruction              */
/*682*/ WORD  bptocc;                   /* # of times the temp. breakpoint should    */
                                        /* be skipped                                */
/*684*/ WORD  bptcocc;                  /* # of times the temp. breakpoint is        */
                                        /* already skipped                           */
/*686*/ char  bptcmd[11];               /* temp. breakpoint command                  */
/*691*/ char  outflag;                  /* output messages (yes=1,no=0)              */
/*692*/ char  namebn[MAXNAME][8];       /* Name buffer, name                         */
/*6BA*/ char  namebd[MAXNAME][40];      /* Name buffer, data                         */
/*782*/ WORD  errcnt;                   /* error counter for test ..                 */
/*784*/ LWORD times,timee;              /* start/end time                            */
/*78C*/ LWORD pregs[N_REGS];            /* storage area of processor regs            */
/*7F0*/ WORD  tflag;                    /* trace active flag                         */
/*7F2*/ WORD  tcount;                   /* trace count                               */
/*7F4*/ WORD  tacount;                  /* active trace count                        */
/*7F6*/ WORD  bpact;                    /* break point active flag                   */
/*7F8*/ LWORD savesp;                   /* save VMEprom stack during GO/T etc        */
/*7FC*/ char  VMEMSP[202];              /* Master stack, handle w/ care              */
/*8C6*/ char  VMESSP[802];              /* supervisor stack, handle w/ care          */
/*BE8*/ char  VMEPUSP[802];             /* vmeprom internal user stack               */
/*F0A*/ LWORD f_fpreg[3*8];             /* floating point data regs                  */
/*F6A*/ LWORD f_fpcr;                   /* FPCR reg                                  */
/*F6E*/ LWORD f_fpsr;                   /* FPSR reg                                  */
/*F72*/ LWORD f_fpiar;                  /* FPIAR reg                                 */
/*F76*/ BYTE  f_save[0x3c];             /* FPSAVE for null and idle                  */
/*FB2*/ BYTE  cleos[2];                 /* clear to end of screen parameter          */
/*FB4*/ BYTE  cleol[2];                 /* clear to end of line parameters           */
/*FB6*/ char  u_prompt[10];             /* user defined prompt sign                  */
/*FC0*/ long  c_save;                   /* save Cache control register               */
/*FC4*/ long  exe_cnt;                  /* execution count                           */
/*FC8*/ BYTE  nokill;                   /* kill task with no input port              */
/*FC9*/ BYTE  u_mask;                   /* unit mask for echo                        */
/*FCA*/ WORD  sysflg;                   /* system flags used by VMEPROM              */
                                        /* bit 0: display registers short form       */
                                        /* bit 1: trace without reg. display         */
                                        /* bit 2: trace over subroutine              */
                                        /* bit 3: trace over subroutine active       */
                                        /* bit 4: trace over range                   */
                                        /* bit 5: no register initialization         */
                                        /* bit 6: output redirection into file       */
                                        /*        and console at the same time       */
/*FCC*/ LWORD t_range[2];               /* start/stop PC for trace over range        */
/*FD4*/ LWORD ex_regs;                  /* pointer to area for saved regs            */
/*FD8*/ BYTE  sparend[0x1000-0xFD8];    /* make tcb size $1000 bytes                 */
        char _tbe[0];                   /* task beginning                            */
};
```

9.5 Interrupt Vector Table of VMEPROM

Vector Number	Vector Address	Assignment
0 (00_{16})	000_{16}	Reset: Initial Supervisor Stack Pointer
1 (01_{16})	004_{16}	Reset: Initial Program Counter
2 (02_{16})	008_{16}	Bus Error
3 (03_{16})	$00C_{16}$	Address Error
4 (04_{16})	010_{16}	Illegal Instruction
5 (05_{16})	014_{16}	Zero Divide
6 (06_{16})	018_{16}	CHK, CHK2 Instruction
7 (07_{16})	$01C_{16}$	FTRAPcc, TRAPcc, TRAPV Instructions
8 (08_{16})	020_{16}	Privilege Violation
9 (09_{16})	024_{16}	Trace
10 ($0A_{16}$)	028_{16}	VMEPROM System Calls
11 ($0B_{16}$)	$02C_{16}$	Coprocessor Instructions
12 ($0C_{16}$)	030_{16}	(Unassigned, Reserved)
13 ($0D_{16}$)	034_{16}	Coprocessor Protocol Violation
14 ($0E_{16}$)	038_{16}	Format Error
15 ($0F_{16}$)	$03C_{16}$	Uninitialized Interrupt
16-23 (10_{16}-17_{16})	040_{16}-$05C_{16}$	(Unassigned, Reserved)
24 (18_{16})	060_{16}	Spurious Interrupt
25 (19_{16})	064_{16}	AV1
26 ($1A_{16}$)	068_{16}	AV2
27 ($1B_{16}$)	$06C_{16}$	AV3
28 ($1C_{16}$)	070_{16}	AV4
29 ($1D_{16}$)	074_{16}	AV5
30 ($1E_{16}$)	078_{16}	AV6
31 ($1F_{16}$)	$07C_{16}$	AV7
32-47 (20_{16}-$2F_{16}$)	080_{16}-$0BC_{16}$	TRAP #0-15 Instruction Vectors
48 (30_{16})	$0C0_{16}$	FP Branch or Set on Unordered Condition
49 (31_{16})	$0C4_{16}$	FP Inexact Result
50 (32_{16})	$0C8_{16}$	FP Divide by Zero
51 (33_{16})	$0CC_{16}$	FP Underflow
52 (34_{16})	$0D0_{16}$	FP Operand Error

Vector Number	Vector Address	Assignment
53 (35_{16})	$0D4_{16}$	FP Overflow
54 (36_{16})	$0D8_{16}$	FP Signaling NAN
55 (37_{16})	$0DC_{16}$	(Unassigned, Reserved)
56 (38_{16})	$0E0_{16}$	PMMU Configuration
57 (39_{16})	$0E4_{16}$	PMMU Illegal Operation
58 ($3A_{16}$)	$0E8_{16}$	PMMU Access Level Violation
59-63 ($3B_{16}$-$3F_{16}$)	$0EC_{16}$-$0FC_{16}$	(Unassigned, Reserved)
64-75 (40_{16}-$4B_{16}$)	100_{16}-$12C_{16}$	SIO-1/2 Interrupt Vectors, Port # 1 - 6
76-83 ($4C_{16}$-53_{16})	130_{16}-$14C_{16}$	ISIO-1/2 Interrupt Vectors, Port # 1,2 - 15,16
84-118 (54_{16}-76_{16})	150_{16}-$1D8_{16}$	User Defined Vectors
119 (77_{16})	$1DC_{16}$	Disk Interrupt Vector (ISCSI-1)
120-191 (78_{16}-BF_{16})	$1E0_{16}$-$2FC_{16}$	User Defined Vectors
192 ($C0_{16}$)	300_{16}	Mailbox 0
193 ($C1_{16}$)	304_{16}	Mailbox 1
194 ($C2_{16}$)	308_{16}	Mailbox 2
195 ($C3_{16}$)	$30C_{16}$	Mailbox 3
196 ($C4_{16}$)	310_{16}	Mailbox 4
197 ($C5_{16}$)	314_{16}	Mailbox 5
198 ($C6_{16}$)	318_{16}	Mailbox 6
199 ($C7_{16}$)	$31C_{16}$	Mailbox 7
200-223 ($C8_{16}$-DF_{16})	320_{16}-37C	(Unassigned, Reserved)
224 ($E0_{16}$)	380_{16}	Timer
225 ($E1_{16}$)	384_{16}	Reserved
226 ($E2_{16}$)	388_{16}	Reserved
227 ($E3_{16}$)	$38C_{16}$	Reserved
228 ($E4_{16}$)	390_{16}	FMB1 Refused
229 ($E5_{16}$)	394_{16}	FMB0 Refused
230 ($E6_{16}$)	398_{16}	FMB1 Message
231 ($E7_{16}$)	$39C_{16}$	FMB0 Message
232 ($E8_{16}$)	$3A0_{16}$	ABORT
233 ($E9_{16}$)	$3A4_{16}$	ACFAIL
234 (EA_{16})	$3A8_{16}$	SYSFAIL
235 (EB_{16})	$3AC_{16}$	DMA Error

Vector Number	Vector Address	Assignment
236 (EC_{16})	$3B0_{16}$	DMA Normal
237 (ED_{16})	$3B4_{16}$	PARITY Error
238 (EE_{16})	$3B8_{16}$	Reserved
239 (EF_{16})	$3BC_{16}$	Reserved
240 ($F0_{16}$)	$3C0_{16}$	LOCAL1
241 ($F1_{16}$)	$3C4_{16}$	LOCAL2
242 ($F2_{16}$)	$3C8_{16}$	LOCAL3
243 ($F3_{16}$)	$3CC_{16}$	LOCAL4
244 ($F4_{16}$)	$3D0_{16}$	LOCAL5
245 ($F5_{16}$)	$3D4_{16}$	LOCAL6
246 ($F6_{16}$)	$3D8_{16}$	LOCAL7
247 ($F7_{16}$)	$3DC_{16}$	LOCAL8
248-254 ($F8_{16}$-FE_{16})	$3E0_{16}$-$3F8_{16}$	(Unassigned, Reserved)
255 (FF_{16})	$3FC_{16}$	Empty Interrupt

9.6 Benchmark Source Code

```
************************************************************
** Module name: Assembler benchmarks   Version: 1.0        **
** date started: 20-Apr-87 M.S. last update: 23-Apr-87  M.S. **
**      Copyright (c) 1986/87 FORCE Computers GmbH Munich   **
************************************************************
*
       section 0
        opt     alt,P=68020,P=68881
        xdef    .benchex
        xdef    .BEN1BEG,.BEN1END
        xdef    .BEN2BEG,.BEN2END
        xdef    .BEN3BEG,.BEN3END
        xdef    .BEN4BEG,.BEN4END
        xdef    .BEN5BEG,.BEN5END
        xdef    .BEN6BEG,.BEN6END
        xdef    .BEN7BEG,.BEN7END
        xdef    .BEN8BEG,.BEN8END
        xdef    .BEN9BEG,.BEN9END
        xdef    .BEN10BEG,.BEN10END
        xdef    .BEN11BEG,.BEN11END
        xdef    .BEN12BEG,.BEN12END
        xdef    .BEN13BEG,.BEN13END
        xdef    .BEN14BEG,.BEN14END
        page
*
* benchmark execution: benchex(address)
*
        movem.l d1-a6,-(a7)
        move.l  15*4(a7),a0
        jsr     (a0)
        movem.l (a7)+,d1-a6
        rts

*
* BENCH #1: DECREMENT LONG WORD IN MEMORY 10.000.000 TIMES
*
        LEA.L   @010(PC),A0
        MOVE.L  #10000000,(A0)
@020    SUBQ.L  #1,(A0)
        BNE.S   @020
        RTS
@010    DS.L    1

*
* BENCH #2: PSEUDO DMA 1K BYTES 50.000 TIMES
*
        MOVE.L  #50000,D2     ; DO 50000 TRANSFERS
@001    MOVE.W  #$FF,D3       ; EACH IS 1K BYTES
        LEA.L   @010(PC),A1   ; A1 POINTS TO SOURCE AND DESTINATION
@002    MOVE.L  (A1),(A1)+
        DBRA    D3,@002
        SUBQ.L  #1,D2
        BNE.S   @001
        RTS
        NOP
@010    NOP
        PAGE
```

```
*
* BENCH #3: SUBSTRING CHARACTER SEARCH 100.000 TIMES TAKEN FROM EDN 08/08/85
*
*
        MOVE.L  #100000,D4
@002    MOVE.L  #15,D0
        MOVE.L  #120,D1
        LEA.L   EDN1DAT(PC),A1
        LEA.L   EDN1DAT1(PC),A0
        BSR.S   EDN1
        SUBQ.L  #1,D4
        BNE.S   @002
        RTS
*
****** BEGIN EDN BENCH #1 *******
EDN1    MOVEM.L D3/D4/A2/A3,-(A7)
        SUB.W   D0,D1
        MOVE.W  D1,D2
        SUBQ.W  #2,D0
        MOVE.B  (A0)+,D3
@010    CMP.B   (A1)+,D3
@012    DBEQ    D1,@010
        BNE.S   @090
        MOVE.L  A0,A2
        MOVE.L  A1,A3
        MOVE.W  D0,D4
        BMI.S   @030
@020    CMP.B   (A2)+,(A3)+
        DBNE    D4,@020
        BNE.S   @012
@030    SUB.W   D1,D2
@032    MOVEM.L (A7)+,D3/D4/A2/A3
        RTS
@090    MOVEQ.L #-1,D2
        BRA.S   @032

******* END EDN BENCH #1 *******

EDN1DAT  DC.B    '00000000000000000000000000000000'
         DC.B    '00000000000000000000000000000000'
EDN1DAT1 DC.B    'HERE IS A MATCH000000000000000'
         PAGE

*
* BENCH #4: BIT TEST/SET/RESET 100.000 TIMES TAKEN FROM EDN 08/08/85
*
        MOVE.L  #100000,D4
        LEA.L   EDN2DAT(PC),A0
@010    MOVEQ.L #1,D0           ; TEST
        MOVEQ.L #10,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVEQ.L #11,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVE.W  #123,D1
        BSR.S   EDN2
        MOVEQ.L #2,D0           ; SET
```

```
 MOVEQ.L #10,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVEQ.L #11,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVE.W  #123,D1
        BSR.S   EDN2
        MOVEQ.L #3,D0           ; RESET
MOVEQ.L #10,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVEQ.L #11,D1
        BSR.S   EDN2
        MOVEQ.L #1,D0
        MOVE.W  #123,D1
        BSR.S   EDN2
        SUBQ.L  #1,D4
        BNE.S   @010
        RTS

*
EDN2    SUB.W   #2,D0
        BEQ.S   @020
        SUBQ.W  #1,D0
        BEQ.S   @030

@010
*       BFTST   (A0){D1:1}
        DC.W    $E8D0
        DC.W    $0841
        SNE     D2
        RTS

@020
*       BFSET   (A0){D1:1}
        DC.W    $EED0
        DC.W    $0841
        SNE     D2
        RTS

@030
*       BFTST   (A0){D1:1}
        DC.W    $E8D0
        DC.W    $0841
        SNE     D2
        RTS
EDN2DAT DC.L    0,0,0,0
        PAGE

*
* BENCH #5: BIT MATRIX TRANSPOSITION 100.000 TIMES
*           TAKEN FROM EDN 08/08/85
*
        MOVE.L  #100000,D4
        LEA.L   EDN3DAT(PC),A0
@002    MOVE.L  #7,D0
        MOVEQ.L #0,D1
        BSR.S   EDN3
        SUBQ.L  #1,D4
        BNE.S   @002
        RTS
```

```
*
EDN3    MOVEM.L D1-D7,-(A7)
        MOVE.L  D1,D2
        MOVE.W  D0,D7
        SUBQ.W  #2,D7
@010    ADDQ.L  #1,D1
        MOVE.L  D1,D3
        ADD.L   D0,D2
        MOVE.L  D2,D4
@020
        BFEXTU  (A0){D3:1},D5
        BFEXTU  (A0){D4:1},D6
        BFINS   D5,(A0){D4:1}
        BFINS   D6,(A0){D3:1}
        ADD.L   D0,D3
        ADDQ.L  #1,D4
        CMP.L   D3,D4
        BNE.S   @020
        DBRA    D7,@010
       MOVEM.L (A7)+,D1-D7
        RTS
EDN3DAT DC.B    %01001001
        DC.B    %01011100
        DC.B    %10001110
        DC.B    %10100101
        DC.B    %00000001
        DC.B    %01110010
        DC.B    %10000000
        EVEN
        PAGE

*
* BENCH #6: CACHE TEST - 128KB PROGRAM IS EXECUTED 1000 TIMES
*           CAUTION: THIS BENCHMARK NEEDS 128 KBYTE MEMORY
*
        LEA.L   @010(PC),A2
        MOVE.L  #$203A0000,D1        ; OPCODE FOR MOVE.L ($0,PC),D0
        MOVE.L  #$20000/4,D2         ; LENGTH IS 128 KBYTE
@004    MOVE.L  D1,(A2)+             ; LOAD OPCODE TO MEMORY
        SUBQ.L  #1,D2
        BNE.S   @004
        MOVE.W  #$4E75,(A2)          ; APPEND RTS
* PROGRAM IS NOW LOADED -- START 1000 TIMES
        MOVE.L  #1000,D3
@008    BSR.S   @010
        SUBQ.L  #1,D3
        BNE.S   @008
        RTS
*
@010    DC.L    0                    ; PROGRAM WILL START HERE
        PAGE

*
* BENCH #7: FLOATING POINT 1.000.000 ADDITIONS
*
        MOVE.L  #1000000,D5
        FMOVE.L #0,FP0
        FMOVE.L #1,FP1
@010    FADD.X  FP0,FP1
        SUBQ.L  #1,D5
        BNE.S   @010
        RTS
```

```
*
* BENCH #8: FLOATING POINT 1.000.000 SINUS
*
        MOVE.L  #1000000,D5
        FMOVE.L #1,FP1
@010    FSIN.X  FP1
        SUBQ.L  #1,D5
        BNE.S   @010
        RTS
        PAGE

*
* BENCH #9: FLOATING POINT 1.000.000 MULTIPLICATIONS
*
        MOVE.L  #1000000,D5
        FMOVE.L #1,FP0
        FMOVE.L #1,FP1
@010    FMUL.X  FP0,FP1
        SUBQ.L  #1,D5
        BNE.S   @010
        RTS
        PAGE

*
* PDOS BENCHMARK #1: CONTEXT SWITCHES
*
        MOVE.L  #100000,D6
@000    XSWP                    ;CONTEXT SWITCH
        SUBQ.L  #1,D6           ;DONE?
          BGT.S @000            ;N
        RTS
        PAGE

*
* PDOS BENCHMARK #2: EVENT SET
*
        MOVEQ.L #32,D1          ;SELECT EVENT 32
        MOVE.L  #100000,D6
*
@000    XSEV                    ;SET EVENT
        SUBQ.L  #1,D6           ;DONE?
          BGT.S @000            ;N
        RTS
        PAGE

*
* PDOS BENCHMARK #3: CHANGE TASK PRIORITY
*
        MOVEQ.L #-1,D0          ;SELECT CURRENT TASK
        MOVEQ.L #64,D1          ;SET PRIORITY TO 64
        MOVE.L  #100000,D6
*
@000    XSTP                    ;SET PRIORITY
        SUBQ.L  #1,D6           ;DONE?
          BGT.S @000            ;N
        RTS
```

```
*
* PDOS BENCHMARK #4: SEND TASK MESSAGE
*
        CLR.L   D0              ;SELECT TASK #0
        LEA.L   MES01(PC),A1    ;POINT TO MESSAGE
        MOVE.L  #100000,D6
*
@000    XSTM                    ;SEND MESSAGE
        XKTM                    ;READ MESSAGE BACK
        SUBQ.L  #1,D6           ;DONE?
          BGT.S @000            ;N
        RTS
MES01   DC.B    'BENCH #13',0
        EVEN
        PAGE

*
* PDOS BENCHMARK #5: READ TIME OF DAY
*
        MOVE.L  #100000,D6
@000    EQU     *
        XRTP
        SUBQ.L  #1,D6           ;DONE?
        BGT.S @000              ;N
        RTS
        end
```

9.7 Modifying Special Locations in ROM

The following table describes some special locations in the VMEPROM binary image.

These locations define the default setup of the name of the start-up file, user program location and RAM disk addresses. These options can be selected by front panel switches.

Some of the locations shown in the table can be changed by the user to adapt VMEPROM to every environment. The following procedure describes how to reprogram the on-board System Flash memory;

If there is another CPU or memory board available in the VME System, it should be used to save the content of the System Flash into a file. First copy the binary image to the local RAM of the other CPU board via the VMEbus:

```
BM FF000000,FF400000,<destination>
```

Then save it to disk. Now the on-board System Flash can be reprogrammed:

1. Enter the boot software by asserting the Reset and Abort switch simultaneously and then releasing the Reset switch.
2. Type INIT <CR> to initialize the FGA-002 and to make the main memory available.
3. Check whether switch SW5/4 is OFF to enable writing to the System Flash memory.
4. Copy the VMEPROM binary image of the System Flash into RAM:

   ```
   BM FF000000,FF040000,0
   ```
5. Modify the code of the VMEPROM image in RAM.
6. Erase the page in System Flash Memory where VMEPROM is stored (these are the first 256KBytes):

   ```
   FERASE SYS_FLASH,0,40000
   ```
7. Reprogram the Flash Memory by using the FPROG command:

   ```
   FPROG SYS_FLASH,0,0,40000
   ```
8. Reboot the system to test the changes.

The address of the following table is located at offset $000C_{16}$ relative to the beginning of the VMEPROM image:

Table 49: User's Patch Table

Offset	Size	Default	Description
00_{16}	DS.B 22	'SY$STRT',0	Name of the start-up file. It has to be a 0-terminated string.
16_{16}	DS.W 1 DS.W 1 DS.L 1 DS.W 1 DS.W 1 DS.L 1 DS.W 1 DS.W 1 DS.L 1	8 2048 4080.0000_{16} 8 2048 4070.0000_{16} 8 2048 $FC80.0000_{16}$	Disk no. of first RAM disk entry. (Upper Rotary, bits 1/0 = 00) No. of 256 byte sectors. Start address of first RAM disk. Disk no. of second RAM disk entry. (Upper Rotary, bits 1/0 = 01) No. of 256 byte sectors. Start address of second RAM disk. Disk no. of third RAM disk entry. (Upper Rotary, bits 1/0 = 10) No. of 256 byte sectors. Start address of third RAM disk.
$2E_{16}$	DS.B 18	'SY$DSK',0	Default name of initialized RAM disk. It must be a 0-terminated string.
40_{16}	DS.L 1 DS.L 1 DS.L 1 DS.L 1	4080.0000_{16} ..... 4070.0000_{16}	These four entries contain the address which is jumped to after kernel initialization. This can be selected by bits 3 and 2 of the Lower Rotary Switch. The second entry contains the address of the BOOT command. The fourth address is the start address of the VMEPROM shell. These values depend on the VMEPROM version.
50_{16}	DS.B 4	'USER'	Disk drivers need this ident to make sure that below data is valid.
54_{16}	DS.B 1	03_{16}	Bit 0: If this bit is "0", no message occurs indicating that VMEPROM is waiting until the hard disk is up to speed. This bit is only considered if bit 1 is set to "1". Bit 1: If it is "0", VMEPROM will not wait until hard disk is up to speed. Bit 2: Reserved, should be "0". Bit 3: Reserved, should be "0". Bit 4: Reserved, should be "0". Bit 5: Reserved, should be "0". Bit 6: Reserved, should be "0". Bit 7: Reserved, should be "0".
55_{16}	DS.B 1	00_{16}	Reserved
56_{16}	DS.B 1	07_{16}	SCSI Controller ID
57_{16}	DS.B 5	$5 * FF_{16}$	Reserved
$5C_{16}$	DS.W 1	16	This entry defines the number of hashing buffers. Valid entries are numbers from 1 to 32. The hashing buffers are used to improve disk access speed. Each buffer can hold 16 Kbytes of data.
$5E_{16}$	DS.L 1	0000.0000_{16}	Reserved

Example of how to find this table:

```
? M FF00000C L
FE00000C   FE00E000 : .

? MD FE00E000 70
FF00E000:  53 59 24 53 54 52 54 00  00 00 00 00 00 00 00 00   SY$STRT.........
FF00E010:  00 00 00 00 00 00 00 08  08 00 40 80 00 00 00 08   ..........@.....
FF00E020:  08 00 40 70 00 00 00 08  08 00 FC 80 00 00 53 59   ...p..........SY
FF00E030:  24 44 53 4B 00 00 00 00  00 00 00 00 00 00 00 00   $DSK............
FF00E040:  40 80 00 00 FF 01 4D 42  40 70 00 00 FF 00 DE B6   @.....MB.p......
FF00E050:  55 53 45 52 03 00 07 FF  FF FF FF FF 00 10 00 00   USER............
FF00E060:  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................

? _
```

9.8 Binding Applications to VMEPROM

9.8.1 General Information

In general, there are two ways to bind an application program in Flash Memory to the VMEPROM kernel. In all cases the application program is executed in user mode. The XSUP system call can be used to switch to supervisor mode. The first way keeps the original flash memory contents unchanged, the second needs to reprogram System Flash.

9.8.2 Using External Memory

The application can be put into an external RR-2 or RR-3 EPROM board on the VMEbus. In this case, the front panel switches of the CPU board must be set so that the application program is started after VMEPROM is booted. In this instance, the user stack is located at the top of the tasking memory and the supervisor stack is located within the task control block. The supervisor stack has a size of 500 bytes. No registers are predefined. If the reserved supervisor stack space is not sufficient, the stack pointer has to be set to point to an appropriate address in RAM.

9.8.3 Using System Flash Memory

Since VMEPROM image needs about 512 Kbytes of the System Flash memory, there are still 3.5 Mbytes of memory available. These can be used to hold a user's application.

SEE ALSO: The reprogramming of the flash memory is described in Section 9.7, 'Modifying Special Locations in ROM,' on page 171.

9.8.4 Binding the Application

Enter the boot software and copy the System Flash memory contents into RAM.

Now merge your own application with the VMEPROM code in RAM.

Then alter the necessary entries in the VMEPROM binary image. Depending on the time the application should be called, this will be the 'Pointer to VMEPROM Initialization' (early exit) or 'Pointer to VMEPROM Shell' (late exit, replacement of the shell)[1)]. Be sure to use the correct addresses (must be calculated for System Flash, not for RAM !).

---

1. In this case the application will be called with the address of the TCB and SYRAM on the stack:
   4(A7) Long word containing the begin address of the TCB
   8(A7) Long word containing the begin address of the system RAM (SYRAM).
   A C-program at this address could look like this:

```
main (struct TCB *pTCB, struct SYRAM *pSYRAM)
{ }
```

SEE ALSO: For further information please refer to Table 45, “Layout of System Flash Memory,” on page 130.

Now the modified image can be programmed into System Flash as described in Section 9.7, ‘Modifying Special Locations in ROM,’ on page 171.

10 Special FGA Boot Commands

The booter on this CPU board is the FGA-002 Boot Software.

At first FGA Boot initializes the devices on the board and checks if the board is System Controller (slot-1). If so, it enables the FGA-002 arbiter. For more details about the slot-1 functionality, please refer to “Slot-1 Detection” on page 107.

Then it initializes the devices on the board and starts the firmware in the second Boot Flash (at address $FFE8.0000_{16}$) or the System Flash Memory (at address $FF00.0000_{16}$). It is also possible to specify an address of a program in the battery buffered SRAM (see SETUP command).

NOTE: The binary images on these locations must be program modules, i.e. must provide an SSP (stack pointer) at offset 0 and a PC (program counter) at offset 4.

If no program modules are found, the debugger will be started instead.
To start the debugger manually, both Rotary Switches must be set to ’F’ and the Abort Switch must be kept asserted while reset.

The standard commands of the debugger are described in section ‘FGA Boot Software’ of the *FGA-002 User’s Manual*. Additional commands available on this version of the booter are described in the following chapters.

A short description of all commands will be listed on screen when ’? <cr>’ is typed in.

Line Editor

The shell knows the following control characters for line editing:

ESC or CONTROL C	Break current command line.
CONTROL A	Recall previous command.
CONTROL B	Go to begin of line.
CONTROL E	Go to end of line.
CONTROL H	Move cursor one character left.
CONTROL L	Move cursor one character right.
CONTROL D	Delete character under cursor.
DEL	Delete character left from cursor.
CONTROL \	Delete from cursor until end of line.
CONTROL O	Delete whole line.
CONTROL I	Toggle between insert/overwrite mode.
ENTER or RETURN	Execute command line.

Figure 6: Boot up procedure

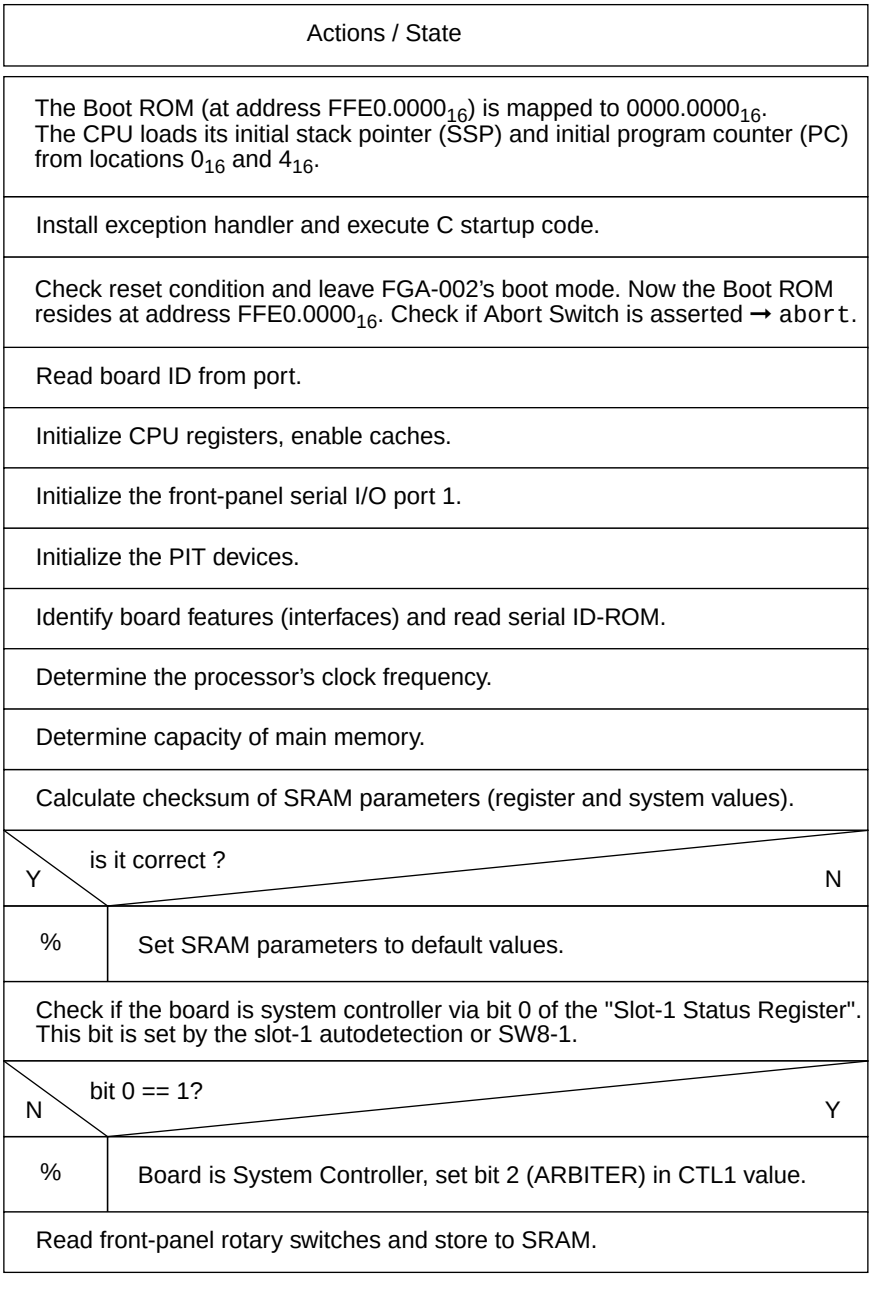
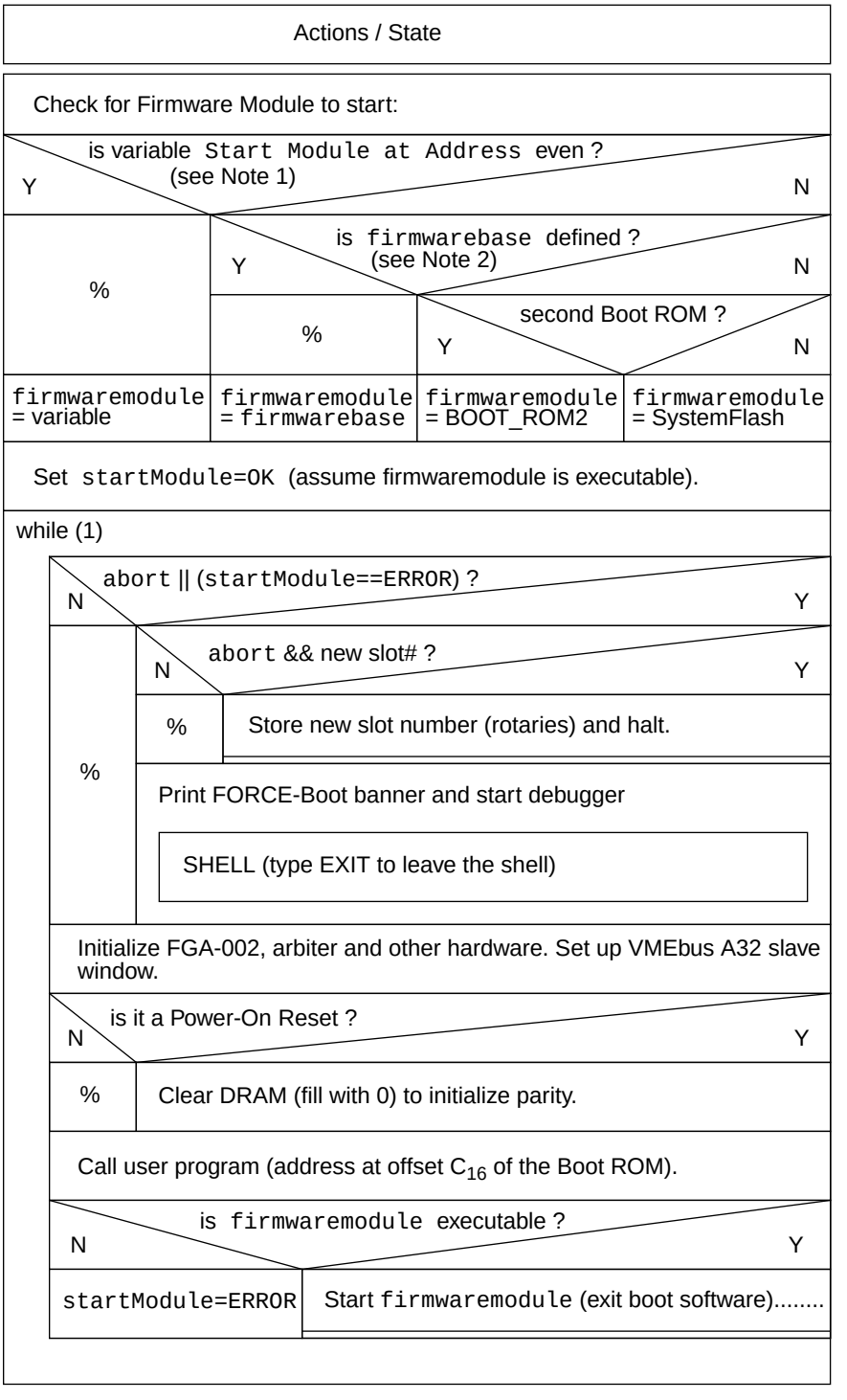


Figure 7: Boot up procedure (continued)



Note 1: This variable can be set with the **SETUP S** command.
Note 2: This is an entry at offset $2C_{16}$ of the Boot ROM which can be patched by the user.

10.1 AS - Line Assembler

Format: AS <address>

The AS command invokes the line assembler of the FGA-002 Boot Software. It can assemble and disassemble all 68020/30/40 mnemonics and 68881/82 floating point instructions.

The AS command, when invoked, displays the current address and disassembles the opcode at this location.

After the prompt on the next line, the user can enter one of the following:

1. A valid 680x0 mnemonic. Some addressing modes allow omission of arguments. These addressing modes can be entered by omitting the argument and typing the dividing character ",".

   Examples:
   ```
   CLR.W ([$1,A0],D0.W,$2)
   CLR.W ([$1,A0],,$2)
   CLR.W ([,A0],,)
   ```

2. A '#' sign followed by the new address changes the address counter to this absolute address.

3. An '=' disassembles the same location again.

4. A '+' or <Return> disassembles the next location.

5. A '+' or '-' sign followed by the number of bytes increases/decreases the address counter.

6. A '.' or <ESC> allow to exit the line assembler and return control to the command interpreter.

7. <Ctrl-A> copies the current disassembled opcode in the line buffer. This allows editing the current mnemonic.

All immediate values, addresses, and offsets used within mnemonics are assumed to be entered in decimal. Hex values have to be specified with a dollar sign '$'. In addition, binary values can be entered by a preceding percent sign '%', octal values by an at sign '@'. The disassembler displays all values in hex representation.

The line assembler accepts also pseudo opcodes of the form `DC.B`, `DC.W` and `DC.L` to define constant data storage. An ASCII pattern can be stored by using `DC.B` with the format '`DC.B "text`'. All characters after the '"' will be interpreted as ASCII characters and stored into memory.

The disassembler displays all illegal or unknown opcodes as `DC.W`.

Example:

```
FORCE-BOOT> AS 8000
$00008000 : ORI.B #0,D0
          : MOVE.L #$123,D1
$00008006 : ORI.B #0,D0
          : -6                    move 6 bytes back
$00008000 : MOVE.L #$123,D1
          : <Ctrl-A>              recall line
          : MOVE.L #$123,D1
$00008006 : ORI.B #0,D0
          : ADDI.L #20,D1
$0000800c : ORI.B #0,D0
          : .                     leave assembler
FORCE-BOOT> _
```

10.2 CONT - Continue with Calling Routine

Format: CONT

The CONT command allows to leave the debugger after it was entered from a user's application via BSR (entry address stored at offset 30_{16} of the booter) or via an exception (for setting a vector, use the address stored at offset 34_{16} of the booter image).

All registers will be restored before leaving the debugger via an RTS or RTE instruction.

Example:

```
FORCE-BOOT> CONT
```

10.3 DI - Disassembler

Format: DI <address>
DI <address>,<count>

The DI command causes the disassembler to be invoked and displays the mnemonic, starting at the specified address. If the count parameter is given, the specified number of lines (mnemonics) will be displayed. If count is omitted, a full page is displayed on the terminal and the user is prompted to continue disassembly (enter <Return>) or to abort (enter any other key).

The disassembler supports all 68020/30/40 mnemonics and the 68881/82 floating point instructions.

Example:

```
FORCE-BOOT> DI 8000 5
00008000   MOVE.L #$123,D1
00008006   ADDI.L #$14,D1
0000800c   ORI.B #0,D0
00008010   ORI.B #0,D0
00008014   ORI.B #0,D0
FORCE-BOOT> _
```

10.4 DRAMINIT - Initialize DRAM

Format: DRAMINIT

The DRAMINIT command fills the complete main memory with 0 if dynamic RAM with parity is used on the board. This forces the parity bits to be correct and prevents parity errors when reading from memory locations that have not been written previously.

Example:

```
FORCE-BOOT> DRAMINIT
FORCE-BOOT> _
```

10.5 FERASE - Erase Flash Memories

Format: FERASE <flashbank>
FERASE <flashbank>,<flashoffset>,<length>

The FERASE command allows to erase Flash Memory banks.
The first format of the command erases the whole Flash Memory bank.
The second format allows to specify a region to erase.

NOTE: This must exactly match the page boundaries of the flash devices. E.g. if the SYS_FLASH bank consists of four 28F008 (1 M * 8 bit) devices in parallel with a page size of 64 Kbyte each, the minimum size of one erasable region is 256 Kbyte (64 KB * 4).

The parameters are used as follows:

<flashbank> Symbolic name or base address of the Flash Memory bank that should be erased. The following symbolic names are currently supported:

BOOT_FLASH	(first) BOOT FLASH
BOOT_FLASH1	first BOOT FLASH
BOOT_FLASH2	second BOOT FLASH
SYS_FLASH	SYSTEM FLASH

<flashoffset> Optional relative byte offset within the flash bank.

<length> Optional length in bytes. If flashoffset and length are not specified, the whole bank will be erased.

Example:

```
FORCE-BOOT> FERASE
Usage: FERASE <flashbank>,[<flashoffset>,<length>]
    Parameter <flashbank> is the base address of the flash bank
    or one of the following defines:
        BOOT_FLASH1     BOOT_FLASH2     SYS_FLASH1

FORCE-BOOT> FERASE BOOT_FLASH1
Do not reprogram BOOT_FLASH1, this would destroy the booter
Device is write protected

FORCE-BOOT> FERASE SYS_FLASH 80000 40000
Erasing flash memory ...   done.
FORCE-BOOT> _
```

10.6 FPROG - Program Flash Memories

Format: FPROG <flashbank>,<source>
FPROG <flashbank>,<source>,<flashoffset>
FPROG <flashbank>,<source>,<flashoffset>,<length>

The FPROG command allows to program Flash Memory banks.

The first format of the command programs the whole Flash Memory bank with the data stored at the specified source address.

The second format additionally allows to specify a destination offset within the Flash Memory bank and programs all the remaining space (from offset to end of flash bank).

The third format of the command also specifies the number of bytes to program.

The parameters are used as follows:

<flashbank> Symbolic name or base address of the Flash Memory bank that should be programmed. The following symbolic names are currently supported:

BOOT_FLASH	(first) BOOT FLASH
BOOT_FLASH1	first BOOT FLASH
BOOT_FLASH2	second BOOT FLASH
SYS_FLASH	SYSTEM FLASH

<flashoffset> Optional relative byte offset within the flash bank. If no offset is specified, 0 is assumed.

<length> Optional length in bytes. If no length is specified, all the remaining space of the flash bank will be programmed.

Example:

```
FORCE-BOOT> FPROG

Usage:  FPROG <flashbank>,<source>[,<flashoffset>[,<length>]]
    Parameter <flashbank> is the base address of the flash bank
    or one of the following defines:
        BOOT_FLASH1     BOOT_FLASH2     SYS_FLASH1

FORCE-BOOT> FPROG BOOT_FLASH1,100000
Do not reprogram BOOT_FLASH1, this would destroy the booter
Device is write protected

FORCE-BOOT> FPROG BOOT_FLASH2,100000
Programming flash memory

  0 |################################################| 100%

Done.

FORCE-BOOT> _
```

10.7 GO - Go to Subroutine

Format: GO <address>

The GO command calls a subroutine at the specified address. To get back into the debugger an RTS instruction must be executed by the routine.

Example:

```
FORCE-BOOT> AS 0
$00000000 : ORI.B #0,D0
          : RTS
$00000002 : ORI.B #0,D0
          : .
FORCE-BOOT> GO 0
FORCE-BOOT> _
```

10.8 LO - Load S-Records to Memory

Format: LO [<host commands>]
LO <offset> [,<host commands>]
LO V [,<host commands>]
LO <offset>,V [,<host commands>]
LO E

The LO command allows to load S-Records from the console port into memory and to verify the memory contents.

The optional parameter '<host commands>' allows to specify a list of commands that will be sent to the host to initiate the data transfer, e.g. 'cat testfile'.

The first format of the command is a standard download. Data will be loaded to the absolute addresses of the S-Records.

The second format contains parameter <offset> that specifies the value that is added to the absolute addresses of the S-Records. This allows to modify the storage address while the download.

The next two formats are the same as previously described, except that no data will be loaded to memory, but a compare takes place between the memory contents and the S-Record data. This allows to verify the data.

Command 'LO E' displays the number of errors that occurred during the last download.

Example:

The following program originally located at address 10.0000_{16} should be loaded to 10.0200_{16} via a 'tip' connection.

Original program:

```
00100000   MOVE.L #$123456,D0
00100006   NOP
00100008   SUBQ.L #1,D0
0010000a   BNE.B $100006
0010000c   RTS
```

S-Record file 'test.x':

```
S0030000FC
S212100000203C001234564E71538066FA4E7530
S804000000FB
```

```
FORCE-BOOT> LO 200              add offset 200₁₆ to addresses
                                use '~C' to execute local command
~CLocal command? cat test.x     use 'cat test.x' to transfer file

away for 2 seconds
!

FORCE-BOOT> DI 100200 5         list program
00100200   MOVE.L #$123456,D0
00100206   NOP
00100208   SUBQ.L #1,D0
0010020a   BNE.B $100206
0010020c   RTS
FORCE-BOOT> _
```

10.9 NETLOAD - Load File via Network to Memory

Format: NETLOAD <filename>,<start address>
[,<ethernet number>]
[, <target IP#>, <server IP#>]

The NETLOAD command loads the specified binary file via Ethernet to memory. It uses the TFTP (Trivial File Transfer Protocol) to connect to the server where the file is located. Therefore a TFTP server must exist.

What happens in detail:

- **RARP:**
  1. If no IP (Internet Protocol) numbers are specified, NETLOAD sends a RARP packet (Reverse Address Resolution Protocol) to translate the board's Ethernet number into an IP address. A RARP server and a translation table are required for this task. E.g. on a UNIX system the file '/etc/ethers' must contain the board's Ethernet number.
  2. After the board has received its own IP number, a TFTP request is sent to this server which has replied the RARP. The server now starts sending the requested file. On a UNIX system, the file must be located in the '/tftpboot' directory.

- **ARP:**
  1. When the board's own target IP and server IP number are given by <target IP#> and <server IP #>, a standard ARP request is broadcasted to get the Ethernet number of the server.
  2. After the board has received the ARP reply, a TFTP request is sent to the specified server. This server now starts sending the requested file. On a UNIX system, the file must be located in the '/tftpboot' directory.

Example:

File 'test' is located in the '/tftpboot' directory of a UNIX system. It contains the text "This is a test". The configuration file '/etc/ethers' has an entry with the board's Ethernet number and the name of the board (its IP address): e.g. "0:80:42:3:88:88 board1".

```
FORCE-BOOT> NETLOAD test 100000 00:80:42:03:88:88
 LAN-controller at address FEF80000 set to Ethernet 00:80:42:03:88:88
 Transmitting RARP-REQUEST...
 LAN-controller at address FEF80000 set to Ethernet 00:80:42:03:88:88
 Transmitting RARP-REQUEST...
 Reception of RARP-REPLY
 Transmitting TFTP-REQUEST...
 PACKET:1 - loaded $00100000..$0010000E (15 bytes)
FORCE-BOOT> MD 100000 10
00100000:  54 68 69 73 20 69 73 20  61 20 74 65 73 74 0a 00   This is a test..
FORCE-BOOT> _
```

10.10 NETSAVE - Save Data via Network to File

Format: NETSAVE <filename>,<start>,<end>,[<ethernet number>] [, <target IP#>, <server IP#>]

The NETSAVE command saves the specified memory region into a file located on a server. Similar to NETLOAD this is done via the TFTP protocol.

NOTE: This file cannot be created. It must already exist with correct write permissions!

Example:

The following commands save the memory region 10.0100_{16} to $10.011F_{16}$ into the file 'test'. The content of the file will be overwritten.

```
FORCE-BOOT> BF 100100 100120 "# Another Test #" P
FORCE-BOOT> MD 100100 20
00100100:  23 20 41 6e 6f 74 68 65  72 20 54 65 73 74 20 23   # Another Test #
00100110:  23 20 41 6e 6f 74 68 65  72 20 54 65 73 74 20 23   # Another Test #
FORCE-BOOT> NETSAVE test 100100 10011F 00:80:42:03:88:88
 LAN-controller at address FEF80000 set to Ethernet 00:80:42:03:88:88
 Transmitting RARP-REQUEST...
 LAN-controller at address FEF80000 set to Ethernet 00:80:42:03:88:88
 Transmitting RARP-REQUEST...
 Reception of RARP-REPLY
 Transmitting TFTP-REQUEST...
 PACKET:1 - saved $00100100..$0010011F (32 bytes)
FORCE-BOOT> _
```

10.11 SETUP - Change Initialization Values

Format: SETUP F or SETUP
SETUP S

The SETUP command is used to change SRAM parameters. 'SETUP F' and 'SETUP' allow to modify the initialization values of the FGA-002 as described in the *FGA-002 User's Manual*. 'SETUP S' allows to set up additional system and application values which are also stored in the battery buffered SRAM. These values can be read by an application software via the utility interface.

SEE ALSO: For further information please refer to Section 11, 'The FGA Boot Utility Interface,' on page 192.

After modifying any entries, the INIT command must be executed to recalculate the SRAM checksum and validate the new values.

The following entries exist:

Name	Default Value
Start Module at Address	$FFFF.FFFF_{16}$
The user can specify the address of a program module here. A module must provide a SSP (stack pointer) at offset 0 and a PC (program counter) at offset 4. If a value of $FFFF.FFFF_{16}$ is set (this is the default), this entry will be ignored and the FGA Boot Software starts the firmware in the second Boot ROM or the System Flash Memory. If the entry contains a valid module base address, the specified program will be executed instead.	
Application Flags	0000_{16}
This entry is reserved for an application and will not be used by the FGA Boot Software. Utility call #39 allows to read it.	
Application Value	0000.0000_{16}
This entry is reserved for an application and will not be used by the FGA Boot Software. Utility call #39 allows to read it.	

Example:

```
FORCE-BOOT> SETUP S
Modify system values:
(<ESC> or <^C> terminates the input)
Start Module at Address   = $FFFFFFFF
Application Flags         = $0000
Application Value         = $00000000
Use the INIT command to recalculate the SRAM checksum !
FORCE-BOOT> INIT
FORCE-BOOT> _
```

10.12 SLOT - Change Slot Number and VMEbus Slave Address

Format: SLOT <slot number>

The SLOT command allows to modify the VMEbus slave address, the VMEbus address of the Mailbox Array (register MYVMEPAGE of the FGA-002) and the FMB slot number (register FMBCTL) as follows:

VMEbus slave address (A32)= 8000.0000_{16} + 0400.0000_{16} * (slot# - 1)
Mailbox base address (A16) = 8000_{16} + 0400_{16} * (slot# - 1)
FBM slot number = slot#

The size of the VMEbus slave window will be set as large as the local memory size of the board.

This is exactly the same as setting the rotary switches and asserting the Abort Key while reset, except the arbiter cannot be enabled or disabled. This can be done by modifying the value of CTL1 via command SETUP.

After setting a new slot number the INIT command must be executed to recalculate the SRAM checksum and validate the new values.

Example:

```
FORCE-BOOT> SLOT 5
Use the INIT command to recalculate the SRAM checksum !
FORCE-BOOT> INIT
FORCE-BOOT> _
```

10.13 VMEADDR - Change VMEbus Slave Address

Format: VMEADDR <slave address>
VMEADDR <slave address>,<slave window>

The VMEADDR command allows modifying the VMEbus slave address without affecting any other settings.

VMEbus slave address (A32)= <slave address>

With the first format of the command, the size of the VMEbus slave window will be set as large as the local memory size of the board.

The second format allows to specify the size of the VMEbus slave window manually.

After setting a new VMEbus slave address the INIT command must be executed to recalculate the SRAM checksum and validate the new values.

Example:

The following example sets the VMEbus slave address of the board to 8300.0000_{16} and the window size to 1 Mbyte. It can now be accessed from 8300.0000_{16} to $830F.FFFF_{16}$.

```
FORCE-BOOT> VMEADDR 83000000,100000
Use the INIT command to recalculate the SRAM checksum !
FORCE-BOOT> INIT
FORCE-BOOT> _
```

11 The FGA Boot Utility Interface

The FGA Boot Software provides several utility functions which can be called from within a user's application. This section describes all additional functions not listed in the *FGA-002 User's Manual*.

The interface expects C-like calling conventions, thus all parameters have to be placed as **long values** on the **supervisor stack**. The first parameter (which must be pushed onto the stack as last one), must contain the number of the requested function, the rest of the parameters depends on the specified function.

NOTE: The utility interface must be called in **supervisor mode.** It will NOT install its own stack but will use the application's stack!

Stack frame:

A7⇒	return address
+4	function number
+8	first fct. argument
+12	second fct. argument
.....	.....

To do a utility call, proceed as follows:

- retrieve the entry address from location $\texttt{BootROM+0008}_{16}$
- load function number and specific arguments onto the stack
- call the FGA-Boot Utility Interface in supervisor mode (JSR)
- retrieve return code from the interface (register D0)
- clean up stack (arguments are still on the stack)

From programming language C, the call could be made as follows (i.e., return version string):

```
#define BootROM 0xFFE00000

main()
{
long    (*util)(long fctNo, ...);
long    ret;


    util = *((long **) (BootROM + 0x0008));

    ret = util (......);
}
```

Utility Functions in Detail:

#36 (0x24) Extended Flash Memory Programming

This routine allows partial programming of Flash Memories.

Syntax: **long util (36, flashbank, source, offset, length)**

Parameters:	**flashbank**	Base address of the Flash Memory bank that should be programmed.
	source	Source address of the data to program.
	offset	Relative byte offset within the flash bank.
	length	Length in bytes.

Returns:	**0**	OK
	1	CLEAR_ERROR
	2	INVAL_PARMS
	3	ERASE_ERROR
	4	WRITE_ERROR
	5	ILL_WIDTH
	6	UNKNOWN_ID
	7	CAPACITY
	8	WRITEPROTECT
	9	NO_VPP
	10	SELECT_ERROR
	11	UNIMP_CMD
	12	UNSUP_DEV

#37 (0x25) Erase Flash Memories

The function allows partial erasing of Flash Memory banks if the devices support page erasing mode.

Syntax: **long util (37, flashbank, offset, length)**

Parameters:

flashbank	Base address of the Flash Memory bank that should be erased.
offset	Relative byte offset within the flash bank.
length	Length in bytes. If length is 0, all the remaining space of the flash bank will be erased.

If **offset** and **length** are both set to 0, the whole flash bank is erased.

Returns:

0	OK
1	CLEAR_ERROR
2	INVAL_PARMS
3	ERASE_ERROR
4	WRITE_ERROR
5	ILL_WIDTH
6	UNKNOWN_ID
7	CAPACITY
8	WRITEPROTECT
9	NO_VPP
10	SELECT_ERROR
11	UNIMP_CMD
12	UNSUP_DEV

Parameters **offset** and **length** must exactly match the page boundaries of the flash devices. For example, if the System Flash bank consists of four 28F008 (1 M * 8 bit) devices in parallel with a page size of 64 Kbyte each, the minimum size of one erasable region is 256 Kbyte (64 KB * 4).

#38 (0x26) Get System Values in SRAM

This function sets a pointer to the base address of the System Values stored in the SRAM. It also returns the size of this structure (#of bytes).

Syntax: **long util (38, SYS_VALUES **pSysValues)**

Parameter: **pSysValues** Address (!) of a pointer to structure SYS_VALUES. This pointer will be set by the routine to that location where the System Values start in SRAM.

```
typedef packed struct
    {
        ULONG   startModule;
    } SYS_VALUES;
```

Returns: **size of struct in bytes**

#39 (0x27) Get Application Values in SRAM

This function sets a pointer to the base address of the System Values stored in the SRAM. It also returns the size of this structure (#of bytes).

Syntax: **long util (39, APPL_VALUES **pApplValues)**

Parameter: **pApplValues** Address (!) of a pointer to structure APPL_VALUES. This pointer will be set by the routine to that location where the Application Values start in SRAM.

```
typedef packed struct
    {
        UWORD   applFlags;
        ULONG   applValue;
    } APPL_VALUES;
```

Returns: **size of struct in bytes**

#40 (0x28) Get Ethernet Number

This function copies the board's Ethernet number (6 bytes) to the specified buffer. The return value contains the status of this operation.

Syntax: **long util (40, intfNumb, pEtherAdr)**

Parameters:	**intfNumb**	Interface number, must be set to 0.
	pEtherAdr	Pointer to buffer where the Ethernet number should be stored into.
Returns:	**0**	OK
	(-1)	ERROR, no Ethernet number available.