SPARC CPU-5V
Technical Reference Manual

Material Number: 203651

Part Number 14028101 420 000 AA

NOTICE

The information in this document has been carefully checked and is believed to be entirely reliable. FORCE COMPUTERS makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. FORCE COMPUTERS reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

FORCE COMPUTERS assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of FORCE COMPUTERS GmbH/Inc.

FORCE COMPUTERS does not convey to the purchaser of the product described herein any license under the patent rights of FORCE COMPUTERS GmbH/Inc. nor the rights of others.

FORCE COMPUTERS Inc.
2001 Logic Drive
San Jose, CA 95124-3468
U.S.A.

Tel.: (408) 369-6000
FAX: (408) 371-3382

FORCE COMPUTERS GmbH
Prof.-Messerschmitt-Str.1
D-85579 Neubiberg/Munich
Germany

Tel.: (089) 608 14-0
FAX: (089) 609 77 93

FORCE COMPUTERS France S.A.R.L.
Le Volta
17-19 rue Jeanne Braconnier
F-92366 Meudon La Forêt Cédex
France

Tel.: (1) 41 07 95 15
FAX: (1) 45 37 06 19

FORCE COMPUTERS U.K. Ltd.
Alton House Office Park
Gatehouse Way
Aylesbury, Bucks. HP19 3XU
United Kingdom

Tel.: (01296) 310400
FAX: (01296) 310420

FORCE COMPUTERS strives to contribute to the safety and preservation of the environment with the same care it applies to the performance and quality of its products.

To support this effort, this Product Manual has been printed on paper that is completely chlorine free and conservation friendly.

List of Tables

This page intentionally left blank

List of Figures

SECTION 1 INTRODUCTION

1. Getting Started

This *SPARC CPU-5V Technical Reference Manual* provides a comprehensive guide to the SPARC CPU-5V board you purchased from FORCE COMPUTERS. In addition, each board delivered by FORCE includes an *Installation Guide.*

Please take a moment to examine the Table of Contents of the *SPARC CPU-5V Technical Reference Manual* to see how this documentation is structured. This will be of value to you when looking for information in the future.

1.1 The SPARC CPU-5V Technical Reference Manual Set

When purchased from FORCE, this set includes the *SPARC CPU-5V Technical Reference Manual* as well as two additional books. These two books are listed here:

Set of Data Sheets for the SPARC CPU-5V

OPEN BOOT PROM 2.0 MANUAL SET

The *Set of Data Sheets for the SPARC CPU-5V* contains the following data sheets.

- NCR SBus I/O Chipset Data Manual
- microSPARC-II User's Manual (STP1012PGA)
- SGS-THOMSON MK48T08(B)-10/12/15/20
- AMD Flash EPROM (AM28F020)
- Intel Flash Memory (28F008SA-L)

The *OPEN BOOT PROM 2.0 MANUAL SET* contains the following three sections.

- Open Boot 2.0 Quick Reference
- Open Boot 2.0 Command Reference
- FCODE Programs

1.2 Summary of the SPARC CPU-5V

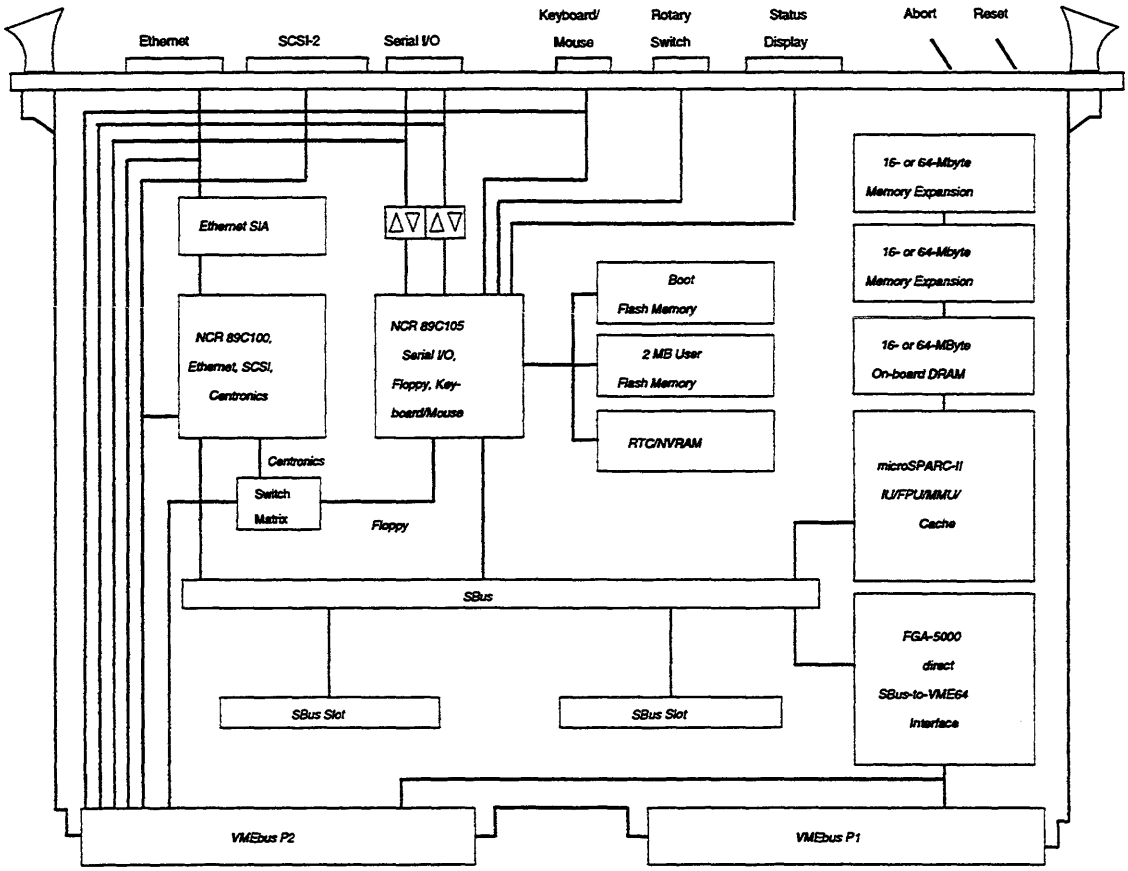
The SPARC CPU-5V addresses embedded applications where processing performance is as important as VMEbus throughput. Based on FORCE COMPUTERS FGA-5000 VMEbus to SBus interface gate array, the SPARC CPU-5V provides high speed VMEbus transfer capabilities for standard transfers and extended 64-bit MBLT transfers. In addition, the SPARC CPU-5V implements the capabilities of Sun Microsystems' SPARCstation 5 workstation on a single-slot VMEbus board.

The SPARC CPU-5V is powerd by the microSPARC-II processor, which delivers a sustained processing performance of 76 SPECint92 and 65 SPECfp92. The complete suite of I/O functions includes fast SCSI-2, Ethernet, floppy disk, serial I/O, Centronics parallel I/O and keyboard/mouse ports making the SPARC CPU-5V the ideal solution for computing and VME transfer intensive embedded applications.

A complete 64-bit VMEbus interface and two standard SBus slots enable the expansion of I/O memory and processing performance with a broad range of off-the-shelf solutions. The software support for the SPARC CPU-5V ranges from Solaris, the most popular implementation of the UNIX operating system on a RISC architecture, to sophisticated real-time operating systems such as VxWorks.

The SPARC CPU-5V is a single board computer combining workstation performance and functionality with the ruggedness and expandability of an industry-standard single-slot 6U VMEbus board.

FIGURE 1. Block Diagram of the SPARC CPU-5V



1.3 Specifications

Below is a table outlining the specifications of the SPARC CPU-5V board.

Table 1: Specifications of the SPARC CPU-5V

Processor	85 or 110 MHz microSPARC-II 64.0 / 76 SPECint92 54.6 / 65 SPECfp92
Memory Management Unit	SPARC Reference MMU
Data/Instruction Cache	8 Kbyte/16 Kbyte
Main Memory	16-or 64-Mbyte base board DRAM, expandable to 192 MB with mezzanine modules
SBus Slots	2
SCSI-2 with DMA to SBus	10 Mbytes/sec fast SCSI-2 I/O on front panel and P2
Ethernet with DMA to SBus	10 Mbits/sec, AM7990 compatible AUI port on front panel or P2
Parallel Port with DMA to SBus	3,4 Mbytes/sec I/O on P2 via switch matrix
Floppy Disk Interface	250, 300, 500 Kbytes/sec and 1 Mbyte/sec I/O on P2 via switch matrix
Serial I/O	2 RS-232 ports, RS-422/485 option via hybrid modules, I/O on front panel and P2
Keyboard/Mouse Port	Sun compatible, on front panel and P2
Counters/Timers	Two 22-bit, 500 ns resolution
Boot Flash Memory	512 Kbyte, on-board programmable Hardware write protection
User Flash Memory	2 Mbytes, on-board programmable Hardware write protection
RTC/NVRAM/Battery	M48T08

Table 1: Specifications of the SPARC CPU-5V (Continued)

VMEbus Interface	64-bit master/slave
Master	A32, A24, A16 D64, D32, D16, D8 MBLT, BLT
Slave	A32, A24, A16 D64, D32, D16, D8 MBLT, BLT, UAT
Additional Features	Reset and Abort switches Status LEDs, HEX display, Rotary switch, Power-on reset circuitry, Voltage sensor
Firmware	OpenBoot with diagnostics
Power Consumption (no SBus Modules installed)	+5V 5.0 A +12V / -12V 0.7 / 0.2A
Environmental Conditions Temperature (Operating) Temperature (Storage) Humidity	 0° C to +55° C -40° C to +85° C 0% to 95% noncondensing
Board Size	Single-Slot 6U form factor 160.00 x 233.35 mm 6.29 x 9.18 inches

1.3.1 Ordering Information

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

Table 2: Ordering Information

Catalog Name	Product Description
CPU-5V/16-110-2	110 MHz microSPARC-II CPU board with 16-Mbyte base board DRAM, 2-Mbyte User Flash Memory, SCSI, Ethernet, floppy disk, parallel and 2 serial I/O ports, 64-bit VMEbus interface, 2 SBus slots, OpenBoot firmware. Installation guide included.
CPU-5V/16-85-2	as above, except 85 MHz microSPARC-II.
CPU-5V/64-110-2	as above, except 110 MHz microSPARC-II and 64-Mbyte base board DRAM.
CPU-5V/64-85-2	as above, except 85 MHz microSPARC-II.
MEM-5/16	16-Mbyte mezzanine memory module for use on the SPARC CPU-5V. Up to two memory modules can be used.
MEM-5/64	64-Mbyte mezzanine memory module for use on the SPARC CPU-5V. Up to two memory modules can be used.
SBus Modules	
SBus/GX	Color 2-D and 3-D wireframe accelerator 1152x900, 8 bits per pixel, single SBus slot.
SBus/TGX	Color 2-D and 3-D wireframe high performance graphics accelerator up to 1152x900, 1-Mbyte VRAM, 8 bits per pixel, single SBus slot.
SBus/TGX+	Color 2-D and 3-D wireframe high performance graphics accelerator up to 1600x1280, 4-Mbyte VRAM, 8 bits per pixel, double buffering, single SBus slot.
SBus/FP	6U front panel for up to 2 SBus cards.
Accessories	
CPU-5V/TM	Technical Reference Manual Set for CPU-5V including OpenBoot User's Manual and a detailed hardware description.
IOBP-10	I/O backpanel on VMEbus P2 with flat cable connectors for Ethernet, SCSI, serial I/O and parallel/floppy disk interface for use with the CPU-5V.
Serial-2CE	Serial adapter cable 26-pin micro D-Sub to 25-pin D-Sub for use with the CPU-5V.

Table 2: Ordering Information (Continued)

Catalog Name	Product Description
FH003/SET	Hybrid modules for RS-422 serial I/O configuration.
FH005/SET	Hybrid modules for RS-485 serial I/O configuration.
Software	
Solaris 2.x/CPU-5V	Solaris 2.x package with Desktop Right-To-Use license, VMEbus driver on tape. Please contact your local sales representative for current version information.
Solaris 2.x/Client-RTU	Solaris 2.x Desktop Right-To-Use license. Without media. Please contact your local sales representative for current version information.
Solaris 2.x/Server-RTU-up	Solaris 2.x Desktop to Workgroup Server Right-To-Use upgrade license. Without media. Please contact your local sales representative for current version information.
Solaris 2.x/UM	Solaris 2.x operating system user manual. Please contact your local sales representative for current version information.
Solaris 1.x/CPU-5V	Solaris 1.x package with Right-To-Use license, VMEbus driver on tape. Please contact your local sales representative for current version information.
Solaris 1.x/CPU-5V/RTU	Solaris 1.x Right-To-Use license. Without media. Please contact your local sales representative for current version information.
Solaris 1.x/CPU-5V/UU-RTU	Solaris 1.x multiuser Right-To-Use license. Without media. Please contact your local sales representative for current version information.
Solaris 1.x/UM	Solaris 1.1 operating system user manual. Please contact your local sales representative for current version information.
VxWorks/DEV SPARC products	VxWorks development package for SPARC host and target. Please contact your local sales representative for current version information.
VxWorks/BSP CPU-5V	VxWorks board support package for CPU-5V. Please contact your local sales representative for current version information.

1.4 History of the Manual

Below is a description of the publication history of this *SPARC CPU-5V Technical Reference Manual.*

Table 3: History of Manual

Revision No.	Description	Date
1	First Print	April 1995

SECTION 2 INSTALLATION

2. Introduction

This Installation Section provides guidelines for powering up the SPARC CPU-5V board. The Installation Section, which you have in your hand now, appears both as Section 2 of the *SPARC CPU-5V Technical Reference Manual* and as a stand-alone *Installation Guide*. This stand-alone Installation Guide is delivered by FORCE COMPUTERS with every board. *The SPARC CPU-5V 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 Caution

Please read this Installation Section before installing the board. Take a moment to examine the Table of Contents to see how this documentation is structured. This will be of value to you when looking for specific information in the future.

2.2 Location Diagram of the SPARC CPU-5V Board

A location diagram showing the important components on the top side of the CPU-5V appears on the next page. On the page next to it, there is a location diagram showing the bottom side of the CPU-5V. Both of these diagrams show only the components on the board which are of interest to the user.

FIGURE 2. Diagram of the CPU-5V (Top View)

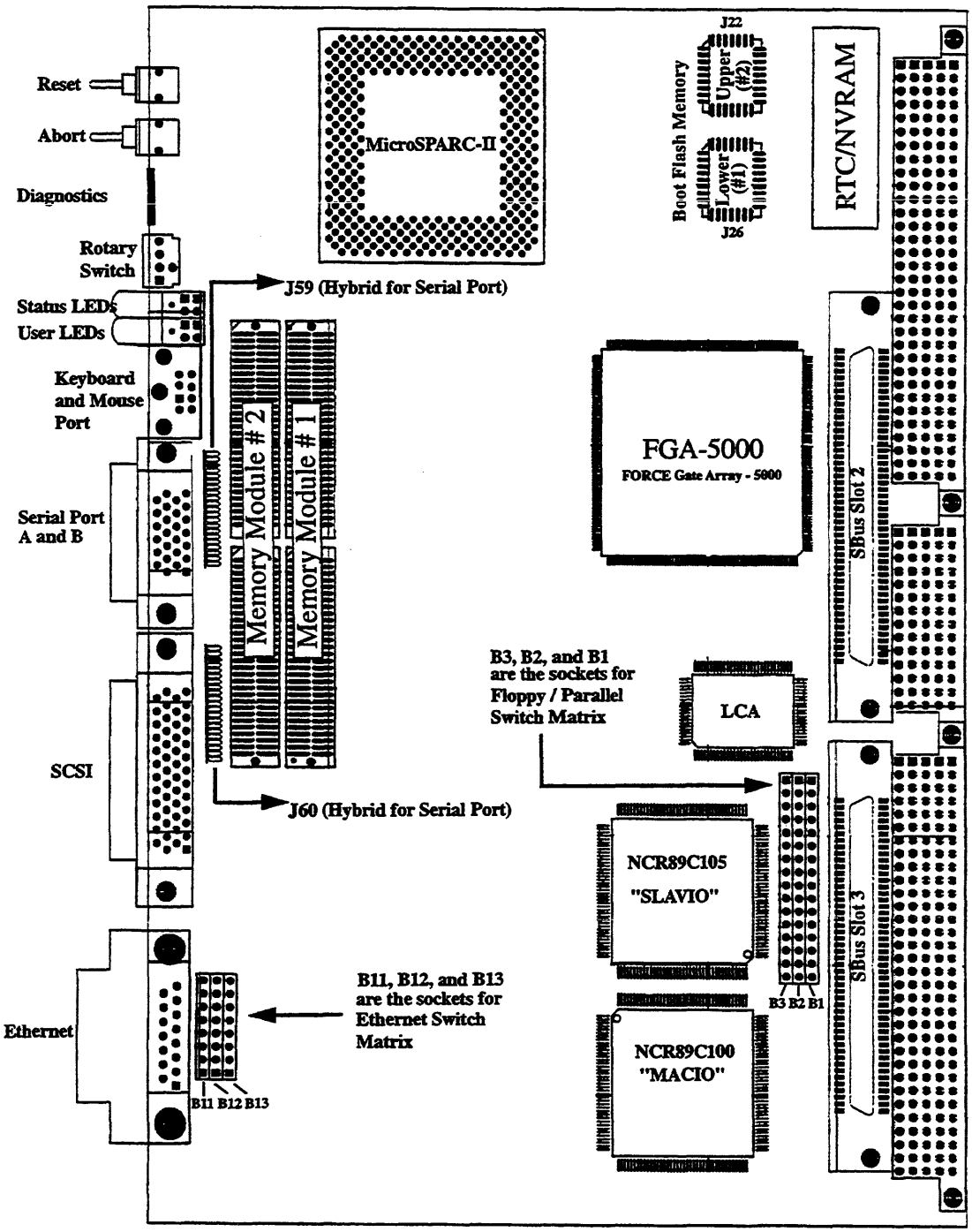
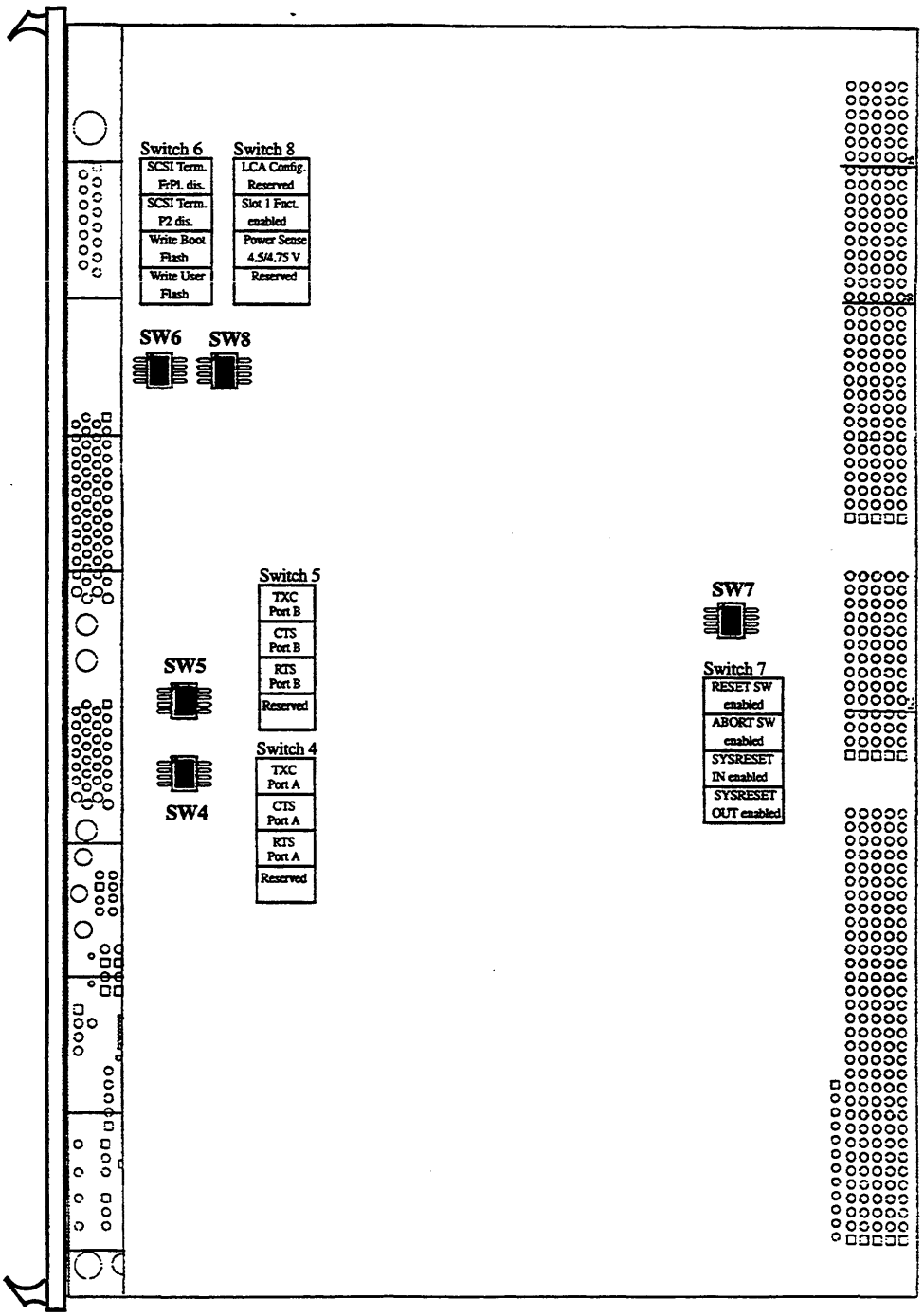


FIGURE 3. Diagram of the CPU-5V (Bottom View)



2.3 Before Powering Up

Before powering up, please make sure that the default switch settings are all set according to the table below. Check these switch settings *before* powering up the SPARC CPU-5V because the board is configured for power up according to these default settings. For the position of the switches on the board, please see "Diagram of the CPU-5V (Top View)" on page 10.

2.3.1 Default Switch Settings

Table 4: Default Switch Settings

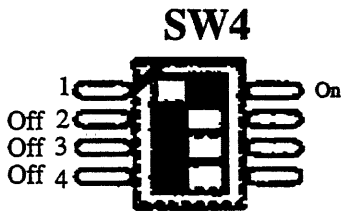
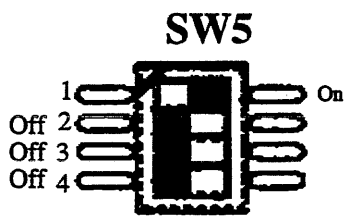
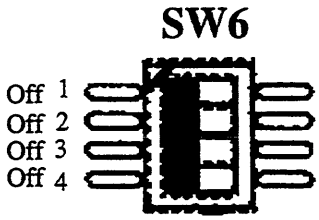
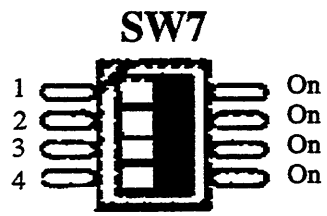
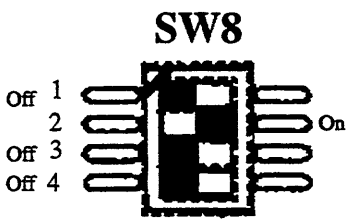
Diagram of Switch with Default Setting	Switches	Default Setting	Function
SWITCH 4 (Serial A Configuration)			
	SW4-1	ON	On = SER_TRXCA to TXC_A_CONN (Pin 24) Off = SER_RTS_A to TXC_A_CONN (Pin 24)
	SW4-2	OFF	On = CTS_A_CONN (Pin 5) to SER_RTXCA and Pullup to SER_CTSA Off = RTXC_A_CONN (Pin 17) to SER_RTXCA and CTS_A_CONN (Pin 5) to SER_CTSA
	SW4-3	OFF	On = SER_TRXCA to RTS_A_CONN (Pin 4) Off = SER_RTS_A to RTS_A_CONN (Pin 4)
	SW4-4	OFF	RESERVED
SWITCH SW5 (Controls Serial Channel B)			
	SW5-1	ON	On = SER_TRXCB to TXC_B_CONN (Pin 25) Off = SER_RTS_B to TXC_B_CONN (Pin 25)
	SW5-2	OFF	On = CTS_B_CONN (Pin 13) to SER_RTXCB and Pullup to SER_CTSB Off = RTXC_B_CONN (Pin 22) to SER_RTXCB and CTS_B_CONN (Pin 13) to SER_CTSB
	SW5-3	OFF	On = SER_TRXCB to RTS_B_CONN (Pin 19) Off = SER_RTS_B to RTS_B_CONN (Pin 19)
	SW5-4	OFF	RESERVED

Table 4: Default Switch Settings (Continued)

Diagram of Switch with Default Setting	Switches	Default Setting	Function
SWITCH 6			
	SW6-1	OFF	On = SCSI-Term Front Panel disabled Off = SCSI-Term Front Panel automatic
	SW6-2	OFF	On = SCSI-Term VME P2 disabled Off = SCSI-Term VME P2 enabled
	SW6-3	OFF	On = Write Boot Flash enabled Off = Write Boot Flash disabled
	SW6-4	OFF	On = Write User Flash enabled Off = Write User Flash disabled
SWITCH 7			
	SW7-1	ON	On = RESET Switch enabled Off = RESET Switch disabled
	SW7-2	ON	On = ABORT Switch enabled Off = ABORT Switch disabled
	SW7-3	ON	On = VME_SYSRESET input enabled Off = VME_SYSRESET input disabled
	SW7-4	ON	On = VME_SYSRESET output enabled (See "VMEbus SYSRESET Enable/Disable" on page 15) Off = VME_SYSRESET output disabled
SWITCH 8			
	SW8-1	OFF	On = LCA Configuration Mode Download Off = LCA Configuration Mode Serial PROM
	SW8-2	ON	On = VME Slot 1 Function enabled Off = VME Slot 1 Function disabled (See "VMEbus SYSRESET Enable/Disable" on page 15)
	SW8-3	OFF	On = Power Sense 4.5V Off = Power Sense 4.75V
	SW8-4	OFF	Reserved

2.4 Powering Up

The initial power up can easily be done by connecting a terminal to ttya (serial port A). The advantage of using a terminal is that no frame buffer, monitor, or keyboard is used for initial power up, which facilitates a simple start up.

Please see the chapter "OpenBoot Firmware" on page 20 for more detailed information on booting the system.

2.4.1 VME Slot-1 Device

The SPARC CPU-5V can be plugged into any VMEbus slot; however, the default configuration sets the board as a VME slot-1 device, which functions as VME system controller. To configure your CPU-5V in order that it is not a VME slot-1 device, the default configuration must be changed so that SW8-2 is OFF. In that case, it would also be necessary to change the SW7-4 to OFF, so that the VME_SYSRESET output is disabled.

CAUTION

Before installing the SPARC CPU-5V in a miniforce chassis, please first disable the VMEbus System Controller function by setting switch SW8-2 to OFF and also setting SW7-4 to OFF.

2.4.2 VMEbus SYSRESET Enable/Disable

2.4.2.1 SYSRESET Input

An external SYSRESET generates an on-board RESET in the default switch setting, i.e., SW7-3 is ON. When SW7-3 is OFF, the external SYSRESET does not generate an on-board RESET.

2.4.2.2 SYSRESET Output

An on-board RESET drives the SYSRESET signal to the VMEbus to low in the default switch setting, i.e., SW7-4 is ON. When SW7-4 is OFF, an on-board RESET doesn't drive the SYSRESET signal to the VMEbus to low.

CAUTION

Do not switch SW7-4 (SYSRESET output) to ON and SW8-2 (VMEbus Slot-1 device) to OFF at the same time.

The VMEbus Specification requires that if SYSRESET is driven, the SYSRESET signal shall be driven low for at least 200 ms. However, when the CPU-5V is not a VMEbus slot-1 device and the SYSRESET output signal is enabled, then the CPU-5V no longer conforms with this rule.

By default, the SYSRESET output is enabled. In this case it generates the SYSRESET signal to the VMEbus.

2.4.3 Serial Ports

By default, both serial ports are configured as RS-232 interfaces. It is also possible to configure both ports as RS-422 or RS-485 interfaces. This optional configuration is achieved with the special FORCE Hybrids FH-003 and FH-005.

The chapter "Default Switch Settings" on page 12 shows the necessary switch settings for RS-232 operation, where SW4 controls serial port A and SW5 controls serial port B. Please check that the switches are set accordingly.

2.4.4 RESET and ABORT Key Enable

To enable the RESET and the ABORT functions on the front panel, set switches SW7-1 (RESET) and SW7-2 (ABORT) to ON.

2.4.5 SCSI Termination

2.4.5.1 SCSI Termination at the Front Panel

Termination at the front panel for the SCSI interface is automatic when SW6-1 is OFF. This is the default setting. Automatic means that when a SCSI cable is plugged into the front panel connector, the termination is automatically disabled. When there is no SCSI cable plugged into the front panel, then the termination is automatically enabled.

2.4.5.2 SCSI Termination at P2

Termination at the VMEbus P2 for the SCSI interface is enabled when SW6-2 is OFF. This is the default setting.

CAUTION

When installing the SPARC CPU-5V in a MICROFORCE chassis, please first disable the SCSI termination by switching SW6-1 and SW6-2 to ON.

2.4.6 Boot Flash Memory Write Protection

Both of the Boot Flash Memory devices are write protectable via the switch SW6-3. When SW6-3 is OFF, the devices are write protected.

2.4.7 User Flash Memory Write Protection

The User Flash Memory devices are write protectable via SW6-4. When SW6-4 is OFF, the User Flash Memory devices are write protected.

2.4.8 Reserved Switches

SW4-4, SW5-4, and SW8-4 are reserved for test purposes.

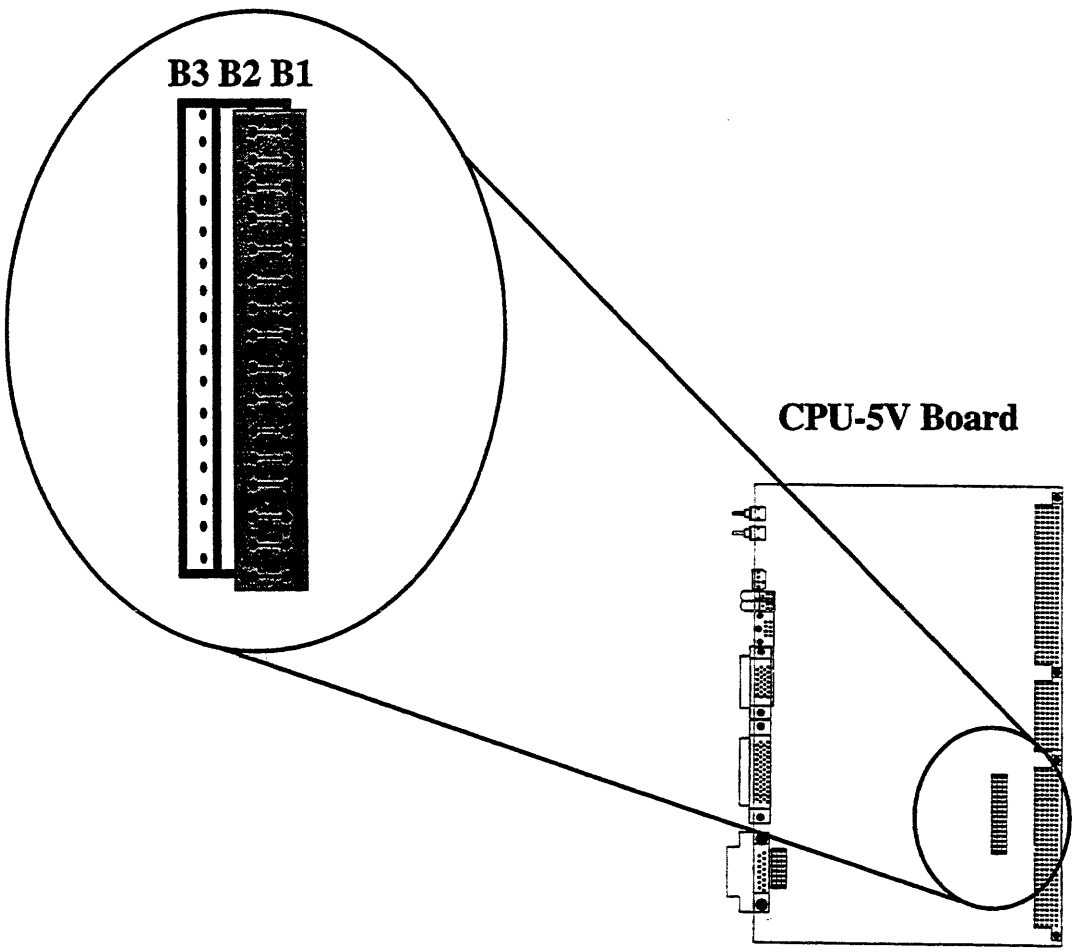
2.4.9 Parallel Port or Floppy Interface via VME P2 Connector

Via a 16-pin configuration switch matrix, it is possible for either the parallel port interface or the floppy interface to be available on the VME P2 connector.

The default setting enables the floppy interface via the VME P2 connector, with the configuration switch matrix plugged into B1 and B2. This means, of course, that by default the parallel port interface is not available via the VMEbus P2 connector.

To enable the parallel port interface via the VME P2 connector, plug the configuration switch matrix in sockets B2 and B3.

FIGURE 4. Floppy Interface Via VME P2 Connector



2.4.10 Ethernet via Front Panel or VME P2 Connector

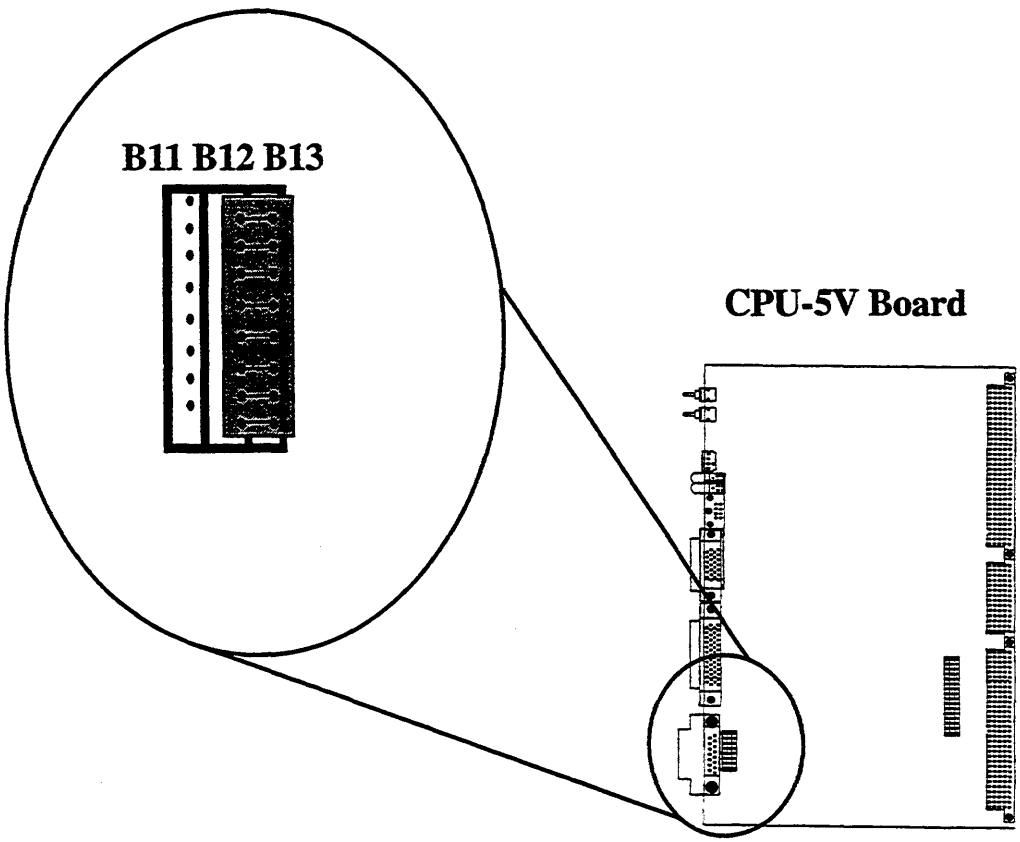
Via an 8-pin configuration switch matrix, it is either possible for the Ethernet interface to be available via the front panel or the VME P2 connector. The default configuration provides the Ethernet through the front panel connector.

In order to have the Ethernet interface accessible via the VME P2 connector, the default configuration must be changed.

By default, the Ethernet interface is available through the front panel with the configuration switch matrix plugged into connectors B12 and B13.

To configure the Ethernet interface to be accessible from the VMEbus P2 connector, the configuration switch matrix must be plugged into connectors B11 and B12.

FIGURE 5. Ethernet Interface via Front Panel



WARNING: When the Ethernet interface is configured via VMEbus P2, do not connect the Ethernet at the front panel.

2.5 OpenBoot Firmware

This chapter describes the use of OpenBoot firmware. Specifically, you will read how to perform the following tasks.

- Boot the System
- Run Diagnostics
- Display System Information
- Reset the System
- OpenBoot Help

For detailed information concerning OpenBoot, please see the *OPEN BOOT PROM 2.0 MANUAL SET.* This manual is included in the *SPARC CPU-5V Technical Reference Manual* Set.

2.5.1 Boot the System

The most important function of OpenBoot firmware is booting the system. Booting is the process of loading and executing a stand-alone program such as the operating system. After it is powered on, the system usually boots automatically after it has passed the Power On SelfTest (POST). This occurs without user intervention.

If necessary, you can explicitly initiate the boot process from the OpenBoot command interpreter. Automatic booting uses the default boot device specified in non-volatile RAM (NVRAM); user initiated booting uses either the default boot device or one specified by the user.

To boot the system from the default boot device, type the following command at the Forth Monitor prompt.

```
ok boot
```

or, if you are at the Restricted Monitor Prompt, you have to type the following:

```
> b
```

The boot command has the following format:

boot [device-specifier] [filename] [-ah]

The optional parameters are described as follows.

[device-specifier]	The name (full path or alias) of the boot device. Typical values are cdrom, disk, floppy, net or tape.
[filename]	The name of the program to be booted. *filename* is relative to the root of the selected device. If no filename is specified, the boot command uses the value of *boot-file* NVRAM parameter. The NVRAM parameters used for booting are described in the following chapter.
[-a]	-a prompt interactively for the device and name of the boot file.
[-h]	-h halt after loading the program.

NOTE: These options are specific to the operating system and may differ from system to system.

To explicitly boot from the internal disk, type:

```
ok boot disk
```

or at the Restricted Monitor prompt:

```
> b disk
```

To retrieve a list of all device alias definitions, type *devalias* at the Forth Monitor command prompt. The following table lists some typical device aliases:

Table 5: Device Alias Definitions

Alias	Boot Path	Description
disk	/iommu/sbus/espdma/esp/sd@3,0	Default disk (1st internal) SCSI-ID 3
disk3	/iommu/sbus/espdma/esp/sd@3,0	First internal disk SCSI-ID 3
disk2	/iommu/sbus/espdma/esp/sd@2,0	Additional internal disk SCSI-ID 2
disk1	/iommu/sbus/espdma/esp/sd@1,0	External disk SCSI-ID 1
disk0	/iommu/sbus/espdma/esp/sd@0,0	External disk SCSI-ID 0
tape	/iommu/sbus/espdma/esp/st@4,0	First tape drive SCSI-ID 4
tape0	/iommu/sbus/espdma/esp/st@4,0	First tape drive SCSI-ID 4
tape1	/iommu/sbus/espdma/esp/st@5,0	Second tape drive SCSI-ID 5
cdrom	/iommu/sbus/espdma/esp/sd@6,0:d	CD-ROM partition d, SCSI-ID 6
net	/iommu/sbus/ledma/le	Ethernet
floppy	/obio/SUNW,fdtwo	Floppy drive

2.5.2 NVRAM Boot Parameters

The OpenBoot firmware holds configuration parameters in NVRAM. At the Forth Monitor prompt, type *printenv* to see a list of all available configuration parameters. The OpenBoot command *setenv* may be used to set these parameters.

setenv [configuration parameter] [value]

This information refers only to those configuration parameters which are involved in the boot process. The following table lists these parameters.

Table 6: Setting Configuration Parameters

Parameter	Default Value	Description
auto-boot?	true	If true, boot automatically after power on or reset
boot-device	disk	Device from which to boot
boot-file	empty string	File to boot
diag-switch?	false	If true, run in diagnostic mode
diag-device	net	Device from which to boot in diagnostic mode
diag-file	empty string	File to boot in diagnostic mode

When booting an operating system or another stand-alone program, and neither a boot device nor a filename is supplied, the boot command of the Forth Monitor takes the omitted values from the NVRAM configuration parameters. If the parameter diag-switch? is false, boot-device and boot-file are used. Otherwise, the OpenBoot firmware uses diag-device and diag-file for booting.

For a detailed description of all NVRAM configuration parameters, please refer to the *OPEN BOOT PROM 2.0 MANUAL SET.*

2.5.3 Diagnostics

At power on or after reset, the OpenBoot firmware executes POST. If the NVRAM configuration parameter diag-switch? is true for each test, a message is displayed on a terminal connected to the first serial port. In case the system is not working correctly, error messages indicating the problem are displayed. After POST, the OpenBoot firmware boots an operating system or enters the Forth Monitor if the NVRAM configuration parameter auto-boot? is false.

The Forth Monitor includes several diagnostic routines. These on-board tests let you check devices such as network controller, SCSI devices, floppy disk system, memory, clock and installed SBus cards. User installed devices can be tested if their firmware includes a selftest routine.

The table below lists several diagnostic routines.

Table 7: Diagnostic Routines

Command	Description
probe-scsi	Identify devices connected to the on-board SCSI bus
probe-scsi-all [*device-path*]	Perform probe-scsi on all SCSI buses installed in the system below the specified device tree node. (If *device-path* is omitted, the root node is used.)
test *device-specifier*	Execute the specified device's selftest method. device-specifier may be a device path name or a device alias. For example: test net - test network connection test /memory - test number of megabytes specified in the selftest-#megs NVRAM parameter or test all of memory if diag-switch? is true
test-all [*device-specifier*]	Test all devices (that have a built-in selftest method) below the specified device tree node. (If *device-path* is omitted, the root node is used.)
watch-clock	Monitor the clock function
watch-net	Monitor network connection

To check the on-board SCSI bus for connected devices, type:

```
ok probe-scsi
Target 3
        Unit 0 Disk MICROP 1684-07MB1036511AS0C1684
ok
```

To test all the SCSI buses installed in the system, type:

```
ok probe-scsi-all
/iommu@0,10000000/sbus@0,10001000/esp@2,100000
Target 6
        Unit 0 Disk Removable Read Only Device SONY CD-ROM CDU-8012 3.1a

/iommu@0,10000000/sbus@0,10001000/espdma@4,8400000/esp@4,8800000
Target 3
        Unit 0 Disk MICROP 1684-07MB1036511AS0C1684

ok
```

The actual response depends on the devices on the SCSI buses.

To test a single installed device, type:

```
ok test device-specifier
```

This executes the device method name selftest of the specified device node.
device-specifier may be a device path name or a device alias as described in Table 5, "Device Alias Definitions," on page 22.The response depends on the selftest of the device node.

To test a group of installed devices, type:

```
ok test-all
```

All devices below the root node of the device tree are tested. The response depends on the devices that have a selftest routine. If a device specifier option is supplied at the command line, all devices below the specified device tree node are tested.

When you use the memory testing routine, the system tests the number of megabytes of memory specified in the NVRAM configuration parameter selftest-#megs. If the NVRAM configuration parameter diag-switch? is true, all memory is tested.

```
ok test /memory
testing 32 megs of memory at addr 0 27
ok
```

The command test-memory is equivalent to test /memory. In the example above, the first number (0) is the base address of the memory bank to be tested, the second number (27) is the number of megabytes remaining. If the CPU board is working correctly, the memory is erased and tested and you will receive the ok prompt. If the PROM or the on-board memory is not working, you receive one of a number of possible error messages indicating the problem.

To test the clock function, type:

```
ok watch-clock
Watching the 'seconds' register of the real time clock chip.
It should be 'ticking' once a second.
Type any key to stop.
22
ok
```

The system responds by incrementing a number once a second. Press any key to stop the test.

To monitor the network connection, type:

```
ok watch-net
Using AUI Ethernet Interface
Lance register test -- succeeded.
Internal loopback test -- succeeded.
External loopback test -- succeeded.
Looking for Ethernet packets.
'.' is a good packet. 'X' is a bad packet.
Type any key to stop.
.....X.....X.....
ok
```

The system monitors the network traffic, displaying "." each time it receives a valid packet and displaying "X" each time it receives a packet with an error that can be detected by the network hardware interface.

2.5.4 Display System Information

The Forth Monitor provides several commands to display system information. These commands let you display the system banner, the Ethernet address for the Ethernet controller, the contents of the ID PROM, and the version number of the OpenBoot firmware.

The ID PROM contains information specific to each individual machine, including the serial number, date of manufacture, and assigned Ethernet address.

The following table lists these commands.

Table 8: Commands to Display System Information

Command	Description
banner	Display system banner.
show-sbus	Display list of installed and probed SBus devices.
.enet-addr	Display current Ethernet address.
.idprom	Display ID PROM contents, formatted.
.traps	Display a list of SPARC trap types.
.version	Display version and date of the Boot PROM.
show-devs	Display a list of all device tree nodes.
devalias	Display a list of all device aliases.

2.5.5 Reset the System

If your system needs to be reset, you either press the reset button on the front panel or, if you are in the Forth Monitor, type **reset** on the command line.

```
ok reset
```

The system immediately begins executing the Power On SelfTest (POST) and initialization procedures. Once the POST finishes, the system either boots automatically or enters the Forth Monitor, just as it would have done after a power-on cycle.

2.5.6 OpenBoot Help

The Forth Monitor contains an on-line help. To get this, type:

```
ok help
Enter 'help command-name' or 'help category-name' for more help
(Use ONLY the first word of a category description)
Examples: help select -or- help line
Main categories are:
File download and boot
Resume execution
Diag (diagnostic routines)
Power on reset
>-prompt
Floppy eject
Select I/O devices
Ethernet
System and boot configuration parameters
Line editor
Tools (memory,numbers,new commands,loops)
Assembly debugging (breakpoints,registers,disassembly,symbolic)
Sync (synchronize disk data)
Nvramrc (making new commands permanent)
ok
```

A list of all available help categories is displayed. These categories may also contain subcategories. To get help for special forth words or subcategories just type help [name]. An example is shown on the next page.

An example of how to get help for special forth words or subcategories:

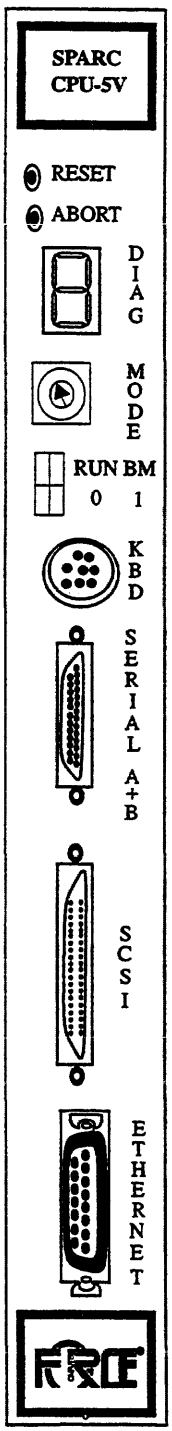
```
ok help tools
Category: Tools (memory,numbers,new commands,loops)
Sub-categories are:
Memory access
Arithmetic
Radix (number base conversions)
Numeric output
Defining new commands
Repeated loops
ok
ok help memory
Category: Memory access
dump ( addr length -- ) display memory at addr for length bytes
fill ( addr length byte -- ) fill memory starting at addr with byte
move ( src dest length -- ) copy length bytes from src to dest address
map? ( vaddr -- ) show memory map information for the virtual address
l? ( addr -- ) display the 32-bit number from location addr
w? ( addr -- ) display the 16-bit number from location addr
c? ( addr -- ) display the 8-bit number from location addr
l@ ( addr -- n ) place on the stack the 32-bit data at location addr
w@ ( addr -- n ) place on the stack the 16-bit data at location addr
c@ ( addr -- n ) place on the stack the 8-bit data at location addr
l! ( n addr -- ) store the 32-bit value n at location addr
w! ( n addr -- ) store the 16-bit value n at location addr
c! ( n addr -- ) store the 8-bit value n at location addr
ok
```

The on-line help shows you the forth word, the parameter stack before and after execution of the forth word (before -- after), and a short description.

The on-line help of the Forth Monitor is located in the boot PROM, so there is not an online help for all forth words.

2.6 Front Panel

FIGURE 6. Diagram of the Front Panel



2.6.1 Features of the Front Panel

The features listed below are described in detail in Section 3 of the *SPARC CPU-5V Technical Reference Manual.*

Table 9: Front Panel Layout

Device	Function	Name
Switch	Reset	RESET
Switch	Abort	ABORT
HEX. Display	Diagnostic	DIAG
Rotary Switch	User defined	MODE
LED/LED	RUN/HALT VME Busmaster/SYSFAIL	RUN BM
LED/LED	Software programmable	0 1
Mini DIN Connector	Keyboard/Mouse	KBD
Serial Connector	Serial Interfaces	SERIAL A+B
SCSI Connector	SCSI Interface	SCSI
D-Sub Connector	Ethernet	ETHERNET

2.7 SPARC CPU-5V Connectors

The connectors on the SPARC CPU-5V are listed in the following table.

Table 10: SPARC CPU-5V Connectors

Function	Location	Type	Manufacturer Part Number
Ethernet	Front Panel	15-pin D-Sub	AMP 747845-4
Serial Port A + B	Front Panel	26-pin Fine Pitch	AMP 749831-2
SCSI	Front Panel	50-pin Fine Pitch	AMP 749831-5
Keyboard/Mouse	Front Panel	8-pin Mini DIN	AMP 749232-1
SBus Slot2 (SBus Slave Select 1)	P3	96-pin SMD	FUJITSU FCN-234J096-G/V
SBus Slot3 (SBus Slave Select 2)	P4	96-pin SMD	FUJITSU FCN-234J096-G/V
VMEbus P1	P1	96-pin VGA	Various
VMEbus P2	P2	96-pin VGA	Various

The following pages show the pinouts of the connectors.

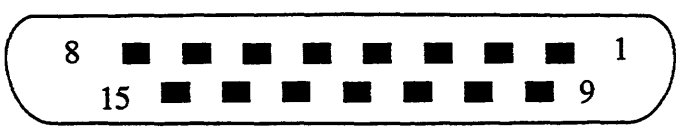
2.7.1 Ethernet Connector Pinout

The following table is a pinout of the Ethernet connector. The figure below shows the Ethernet connector and pin numbers.

Table 11: Ethernet Connector Pinout

Pin	Function
1	Analog GND
2	Collision+
3	Transmit Data+
4	Analog GND
5	Receive Data+
6	Analog GND
7	N.C.
8	Analog GND
9	Collision-
10	Transmit Data-
11	Analog GND
12	Receive Data-
13	+12VDC
14	Analog GND
15	N.C.

FIGURE 7. Pinout of the Ethernet Cable Connector



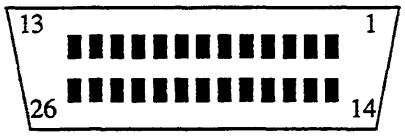
2.7.2 Serial Port A and B Connector Pinout

The following table is a pinout of the serial port connector. The figure on the next page shows the serial port connector and location of the pin numbers.

Table 12: Serial Port A and B Connector Pinout

Pin	Signal	Direction	Port	Description
1	none	none	A	Not connected
2	TD	output	A	Transmit Data
3	RD	input	A	Receive Data
4	RTS	output	A	Request To Send
5	CTS	input	A	Clear To Send
6	DSR	input	A	Data Set Ready
7	SG	none	A	Signal Ground
8	DCD	input	A	Data Carrier Detect
9	none	none	Not connected	
10	none	none	Not connected	
11	SDTR	output	B	Secondary Data Terminal Ready
12	SDCD	input	B	Secondary Data Carrier Detect
13	SCTS	input	B	Secondary Clear To Send
14	STD	output	B	Secondary Transmit Data
15	TC	input	A	Transmit Clock: DCE source
16	SRD	input	B	Secondary Receive Data
17	RC	input	A	Receive Clock
18	STC	input	B	Secondary Transmit Clock
19	SRTS	output	B	Secondary Request To Send
20	DTR	output	A	Data Terminal Ready
21	SDSR	input	B	Secondary Data Terminal Ready*
22	SRC	input	B	Secondary Receive Clock*
23	SSG	none	B	Secondary Signal Ground
24	TC	output	A	Transmit Clock: DTE source
25	STC	output	B	Transmit Clock: DTE source

FIGURE 8. Serial Ports A and B Connector Pinout



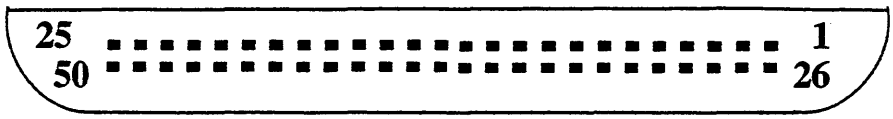
2.7.3 SCSI Connector Pinout

The following table is a pinout of the SCSI connector. The figure on the next page shows the SCSI connector and location of the pin numbers.

Table 13: SCSI 50-Pin Connector

Pin No.	Signal	Pin No.	Signal
1	GND	26	SCSI Data 0
2	GND	27	SCSI Data 1
3	GND	28	SCSI Data 2
4	GND	29	SCSI Data 3
5	GND	30	SCSI Data 4
6	GND	31	SCSI Data 5
7	GND	32	SCSI Data 6
8	GND	33	SCSI Data 7
9	GND	34	SCSI DP
10	GND	35	GND
11	GND	36	DISABLE TERM
12	N.C.	37	N.C.
13	N.C.	38	TERMPWR
14	N.C.	39	N.C.
15	GND	40	GND
16	GND	41	SCSI ATN
17	GND	42	GND
18	GND	43	SCSI BSY
19	GND	44	SCSI ACK
20	GND	45	SCSI RST
21	GND	46	SCSI MSG
22	GND	47	SCSI SEL
23	GND	48	SCSI CD
24	GND	49	SCSI REQ
25	GND	50	SCSI IO

FIGURE 9. Pinout of SCSI Connector



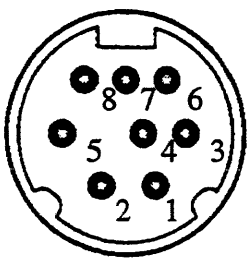
2.7.4 Keyboard/Mouse Connector Pinout

The following table is a pinout of the keyboard/mouse connector.The keyboard and mouse port is available on the front panel via a Mini DIN connector.

Table 14: Keyboard/Mouse Connector Pinout

Pin	Function
1	GND
2	GND
3	+5VDC
4	Mouse In
5	Keyboard Out
6	Keyboard In
7	Mouse Out
8	+5VDC

FIGURE 10. Keyboard/Mouse Connector



2.7.5 VME P2 Connector Pinout

The following table is a pinout of the VME P2 connector.

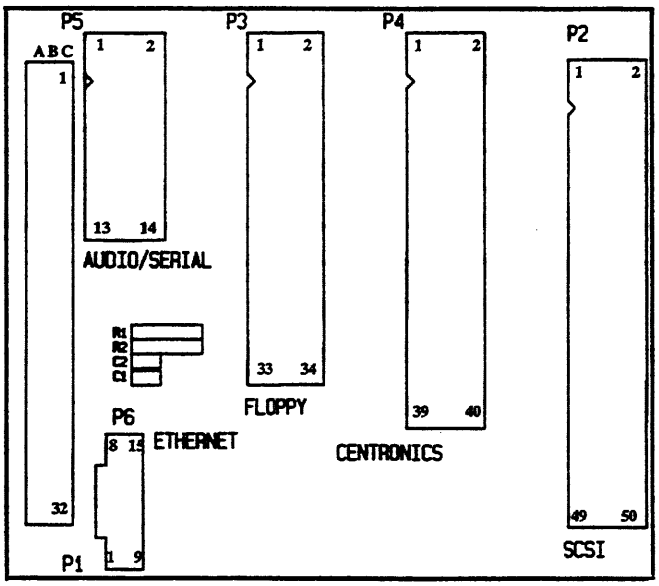
Table 15: VME P2 Connector Pinout

PIN	Row A	Row C (Floppy drive available via switch matrix connected to sockets B1 & B2)	Row C (Parallel Port available via switch matrix connected to sockets B2 & B3)
1	SCSI Data 0	FLPY DENSEL	CENTR DS
2	SCSI Data 1	FLPY DENSENSE	CENTR D0
3	SCSI Data 2	CENTR D1	CENTR D1
4	SCSI Data 3	FLPY INDEX	CENTR D2
5	SCSI Data 4	FLPY DRVSEL	CENTR D3
6	SCSI Data 5	CENTR D4	CENTR D4
7	SCSI Data 6	CENTR D5	CENTR D5
8	SCSI Data 7	FLPY MOTEN	CENTR D6
9	SCSI DP	FLPY DIR	CENTR D7
10	GND	FLPY STEP	CENTR ACK
11	GND	FLPY WRDATA	CENTR BSY
12	GND	FLPY WRGATE	CENTR PE
13	TERMPWR	FLPY TRACK0	CENTR AF
14	GND	FLPY WRPROT	CENTR INIT
15	GND	FLPY RDDATA	CENTR ERR
16	SCSI ATN	FLPY HEADSEL	CENT SLCT IN
17	GND	FLPY DISKCHG	CENTR SLCT
18	SCSI BSY	FLPY EJECT	N.C.
19	SCSI ACK	ETH +12V	ETH +12V
20	SCSI RST	GND	GND
21	SCSI MSG	GND	GND
22	SCSI SEL	ETH REC+ 2)	ETH REC+ 2)
23	SCSI CD	ETH REC- 2)	ETH REC- 2)
24	SCSI REQ	ETH TRA+ 2)	ETH TRA+ 2)
25	SCSI IO	ETH TRA- 2)	ETH TRA- 2)
26	Mouse IN	ETH COL+ 2)	ETH COL+ 2)
27	Keyboard OUT	ETH COL- 2)	ETH COL- 2)
28	Keyboard IN	GND	GND
29	TXD Port A	TXD Port B	TXD Port B
30	RXD Port A	RXD Port B	RXD Port B
31	RTS Port A	RTS Port B	RTS Port B
32	CTS Port A	CTS Port B	CTS Port B

2.7.6 The IOBP-10 Connectors

The IOBP-10 is an I/O back panel on VMEbus P2 with flat cable connectors for SCSI, serial I/O, Centronics/floppy interface, and a micro D-Sub connector for an Ethernet interface. This back panel can be plugged into the VMEbus P2 connector. The diagram below shows all the connectors. This IOBP-10 back panel is especially designed for the SPARC CPU-5V. Do not use any other I/O back panels on the SPARC CPU-5V, for example, the IOBP-1.

FIGURE 11. The IOBP-10



The pinouts of the connectors (P1) ... (P6) are shown in the following tables.

CAUTION

This IOBP-10 back panel is especially designed for the SPARC CPU-5V. Do not use any other I/O back panels on the SPARC CPU-5V, for example, the IOBP-1.

Table 16: IOBP-10 P1 Pinout

ROW A	Signal	ROW B	Signal	ROW C	Signal for Floppy Interface [1]	Signal for Parallel Port Interface [1]
1	SCSI Data 0	1	N.C.	1	FPY DENSEL	CENTR DS
2	SCSI Data 1	2	GND	2	FPY DENSENS	CENTR Data 0
3	SCSI Data 2	3	N.C..	3	N.C.	CENTR Data 1
4	SCSI Data 3	4	N.C.	4	FPY INDEX	CENTR Data 2
5	SCSI Data 4	5	N.C.	5	FPY DRVSEL	CENTR Data 3
6	SCSI Data 5	6	N.C.	6	N.C.	CENTR Data 4
7	SCSI Data 6	7	N.C.	7	N.C.	CENTR Data 5
8	SCSI Data 7	8	N.C.	8	FPY MOTEN	CENTR Data 6
9	SCSI DP	9	N.C.	9	FPY DIR	CENTR Data 7
10	GND	10	N.C.	10	FPY STEP	CENTR ACK
11	GND	11	N.C.	11	FPY WRDATA	CENTR BSY
12	GND	12	GND	12	FPY WRGATE	CENTR PE
13	TERMPWR	13	N.C.	13	FPY TRACK0	CENTR AF
14	GND	14	N.C.	14	FPY WRPROT	CENTR INIT
15	GND	15	N.C.	15	FPY RDDATA	CENTR ERR
16	SCSI ATN	16	N.C.	16	FPY HEADSEL	CENTR SLCT IN
17	GND	17	N.C.	17	FPY DISKCHG	CENTR SLCT
18	SCSI BSY	18	N.C.	18	FPY EJECT	RESERVED
19	SCSI ACK	19	N.C.	19	+12VDC [2]	+12VDC [2]
20	SCSI RST	20	N.C.	20	GND	GND
21	SCSI MSG	21	N.C.	21	GND	GND
22	SCSI SEL	22	GND	22	ETH REC+ [2]	ETH REC+ [2]
23	SCSI CD	23	N.C.	23	ETH REC- [2]	ETH REC- [2]
24	SCSI REQ	24	N.C.	24	ETH TRA+ [2]	ETH TRA+ [2]
25	SCSI IO	25	N.C.	25	ETH TRA- [2]	ETH TRA- [2]
26	Mouse IN	26	N.C.	26	ETH COL+ [2]	ETH COL+ [2]

Table 16: IOBP-10 P1 Pinout (Continued)

ROW A	Signal	ROW B	Signal	ROW C	Signal for Floppy Interface [1]	Signal for Parallel Port Interface [1]
27	Keyboard Out	27	N.C.	27	ETH COL- [2]	ETH COL- [2]
28	Keyboard In	28	N.C.	28	GND	GND
29	TxD Port A	29	N.C.	29	TxD Port B	TxD Port B
30	RxD Port A	30	N.C.	30	RxD Port B	RxD Port B
31	RTS Port A	31	GND	31	RTS Port B	RTS Port B
32	CTS Port A	32	N.C.	32	CTS Port B	CTS Port B

1) For further information, see "Floppy Interface Via VME P2 Connector" on page 18
2) For further information, please see "Ethernet Interface via Front Panel" on page 19

Table 17: IOBP-10 P2 Pinout (SCSI)

Pin No.	Signal	Pin No.	Signal
1	GND	2	SCSI Data 0
3	GND	4	SCSI Data 1
5	GND	6	SCSI Data 2
7	GND	8	SCSI Data 3
9	GND	10	SCSI Data 4
11	GND	12	SCSI Data 5
13	GND	14	SCSI Data 6
15	GND	16	SCSI Data 7
17	GND	18	SCSI DP
19	GND	20	GND
21	GND	22	GND
23	GND	24	GND
25	GND	26	TERMPWR
27	N.C.	28	GND
29	GND	30	GND
31	GND	32	SCSI ATN
33	GND	34	GND
35	GND	36	SCSI BSY
37	GND	38	SCSI ACK
39	GND	40	SCSI RST
41	GND	42	SCSI MSG
43	GND	44	SCSI SEL
45	GND	46	SCSI CD
47	GND	48	SCSI REQ
49	GND	50	SCSI IO

Table 18: IOBP-10 P3 Pinout (Floppy)

Pin No.	Signal	Pin No.	Signal
1	FPY EJECT	2	FPY DENSEL
3	GND	4	FPY DENSENS
5	GND	6	N.C.
7	GND	8	FPY INDEX
9	GND	10	FPY DRVSEL
11	GND	12	N.C.
13	GND	14	N.C.
15	GND	16	FPY MOTEN
17	GND	18	FPY DIR
19	GND	20	FPY STEP
21	GND	22	FPY WRDATA
23	GND	24	FPY WRGATE
25	GND	26	FPY TRACK0
27	N.C.	28	FPY WRPROT
29	GND	30	FPY RDDATA
31	GND	32	FPY HEADSEL
33	GND	34	FPY DISKCHG

Table 19: IOBP-10 P4 Pinout (Centronics)

Pin No.	Signal	Pin No.	Signal
1	CENTR DS	2	GND
3	CENTR Data 0	4	GND
5	CENTR Data 1	6	GND
7	CENTR Data 2	8	GND
9	CENTR Data 3	10	GND
11	CENTR Data 4	12	GND
13	CENTR Data 5	14	GND
15	CENTR Data 6	16	GND
17	CENTR Data 7	18	GND
19	CENTR ACK	20	GND
21	CENTR BSY	22	GND
23	CENTR PE	24	GND
25	CENTR SLCT	26	CENTR INIT
27	CENTR AF	28	CENTR ERR
29	N.C.	30	GND
31	GND	32	N.C.
33	N.C.	34	N.C.
35	N.C.	36	CENTR SLCT IN
37	N.C.	38	N.C.
39	N.C.	40	N.C.

Table 20: IOBP-10 P5 Pinout (Serial)

Pin No.	Signal	Pin No.	Signal
1	GND	2	Keyboard In
3	Mouse In	4	Keyboard Out
5	TxD Port B	6	TxD Port A
7	RxD Port B	8	RxD Port A
9	RTS Port B	10	RTS Port A
11	CTS Port B	12	CTS Port A
13	GND	14	GND

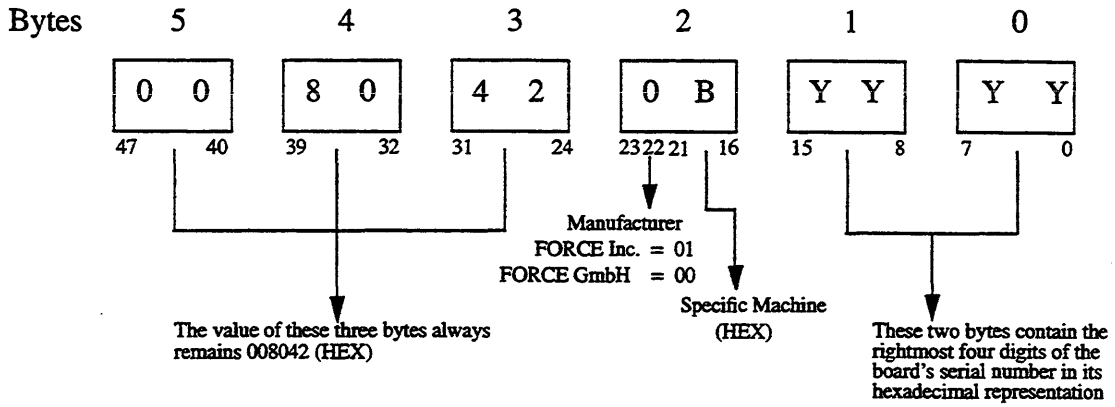
Table 21: IOBP-10 P6 Pinout (Ethernet)

Pin	Function
1	GND
2	Collision+
3	Transmit Data+
4	GND
5	Receive Data+
6	GND
7	N.C.
8	N.C.
9	Collision-
10	Transmit Data-
11	GND
12	Receive Data-
13	+12VDC
14	GND
15	N.C.

2.8 How to Determine the Ethernet Address and Host ID

This information explains how an Ethernet number and a host ID number are determined on a CPU-5V board.

The 48-Bit (Six Byte) Ethernet Address



The Ethernet Address

The Ethernet Address consists of a 48-bit (6-byte) number. The value of the most significant 24 bits is always 008042 (HEX). The value of the least significant 24 bits is calculated as follows.

The bits 23 and 22 identify whether the product is designed by FORCE COMPUTERS GmbH or by FORCE COMPUTERS Inc. The value is 00 for GmbH products and the value is 01 for Inc. products.

The bits 21 through 16 (least significant six bits of the third byte) identify a specific machine. The value of a CPU-5V board is 11 (decimal). Thus, the value of the most significant 32 bits (most significant four bytes) of a CPU-5V board from GmbH is 0080420B.

The least significant 16 bits (least significant two bytes) contain the rightmost four digits of the board's serial number in its hexadecimal representation.

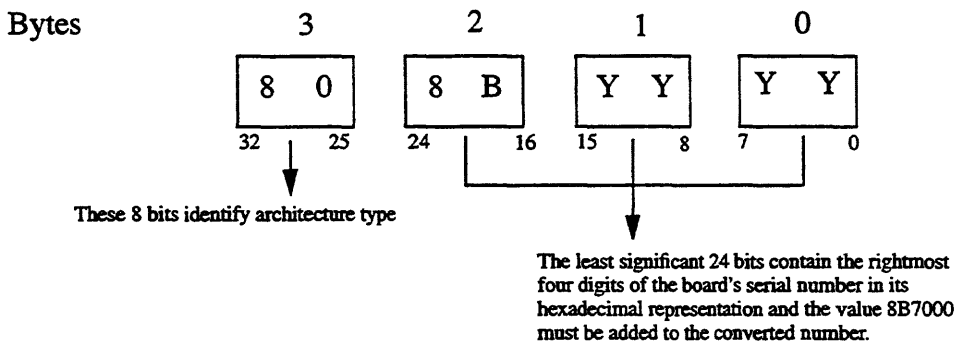
```
Sample Serial Number :    2910 (decimal) = > 0B5E (hexadecimal)

Sample Ethernet Address: 00 80 42 0B 0B 5E (hexadecimal)
```

The Host ID Number

The host ID is a 32-bit (4-byte) number and the eight most significant bits identify the architecture of the machine. The value representing the architecture of CPU-5V is 80 (HEX). The least significant 24 bits (least significant three bytes) contain the rightmost four digits of the board's serial number in its hexadecimal representation and the value 8B7000 (HEX) must be added to the converted serial number.

The Host ID Number




```
Sample Serial Number: 2910 (decimal) => 0B5E (hexadecimal)

                      80 8B 7000
                      +     0B5E
                      ----------
Sample Host ID:       80 8B 7B5E (hexadecimal)
```

SECTION 3 HARDWARE DESCRIPTION

3. Overview

Based on FORCE COMPUTERS FGA-5000 VMEbus to SBus interface gate array, the SPARC CPU-5V provides high speed VMEbus transfer capabilities for standard transfers and extended 64-bit MBLT transfers. In addition, the SPARC CPU-5V implements the capabilities of Sun Microsystems' SPARCstation 5 workstation on a single-slot VMEbus board.

The SPARC CPU-5V is powered by the microSPARC-II processor, which delivers a sustained processing performance of 76 SPECint92 and 65 SPECfp92. The complete suite of I/O functions includes fast SCSI-2, Ethernet, floppy disk, serial I/O, Centronics parallel I/O and keyboard/mouse ports making the SPARC CPU-5V the ideal solution for computing and VME transfer intensive embedded applications.

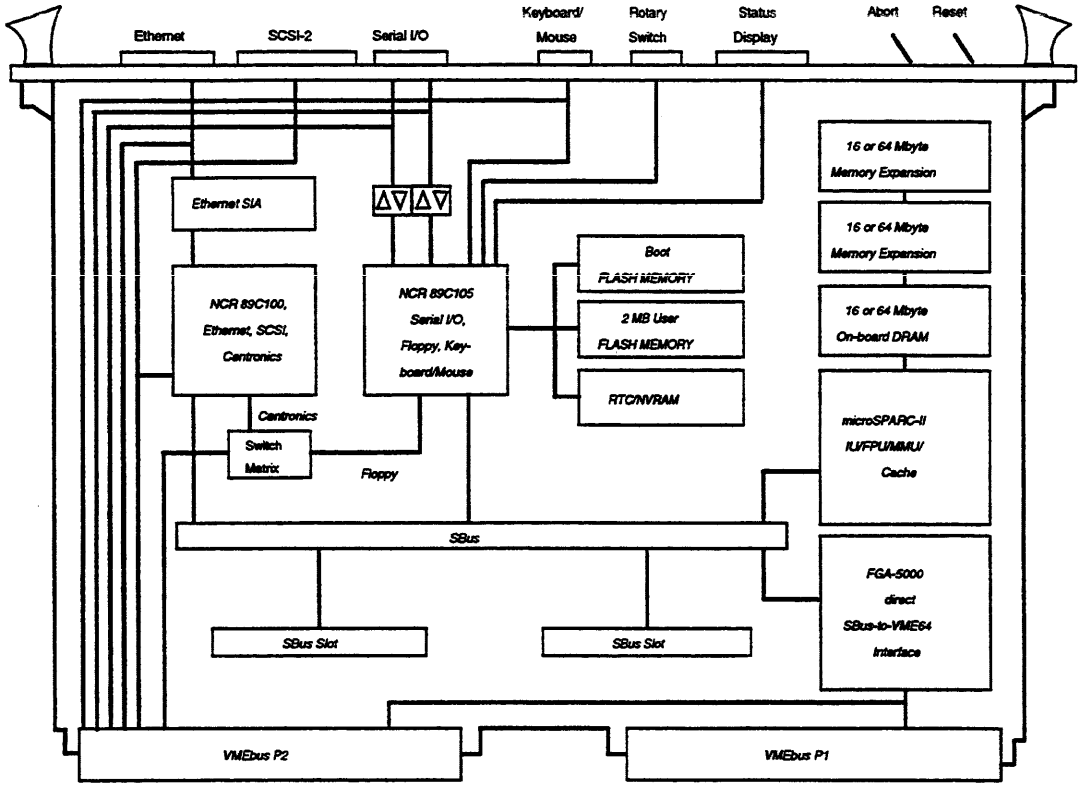
A complete 64-bit VMEbus interface and two standard SBus slots enable the expansion of I/O memory and processing performance with a broad range of off-the-shelf solutions. The software support for the SPARC CPU-5V ranges from Solaris, the most popular implementation of the UNIX operating system on a RISC architecture, to sophisticated real-time operating systems such as VxWorks.

The SPARC CPU-5V is a single board computer combining workstation performance and functionality with the ruggedness and expandability of an industry-standard single-slot 6U VMEbus board.

3.1 Block Diagram

A block diagram showing a functional overview of the CPU-5V is shown on the next page.

FIGURE 12. Block Diagram of the SPARC CPU-5V



3.2 The microSPARC-II Processor

The microSPARC-II CPU chip is at the core of the SPARC CPU-5V. This chip is realized in a 321-pin CPGA package. A Floating Point Unit, an Integer Unit, an MMU, an Instruction Cache, and a Data Cache are integrated in the microSPARC-II processor. Please see the microSPARC-II User's Manual (STP1012PGA) for further information.

3.2.1 Features of the microSPARC-II Processor

- microSPARC-II chip running at 85 MHz and 110 MHz
- Integer Unit with 5-stage pipeline
- Floating Point Unit
- SPARC Reference Memory Management Unit
- A 16 Kbyte instruction cache and an 8 Kbyte data cache, directly mapped
- Memory interface which supports up to 256 Mbyte DRAM
- SBus controller supports up to five SBus slots plus one "master-only" slot

3.2.2 Address Mapping for microSPARC-II

The table below lists the physical addresses of the microSPARC-II processor.

Table 22: Physical Memory Map of microSPARC-II

Address	Function	SBus Slot #	Select #
0000 0000 -> 0FFF FFFF	User Memory		
1000 0000 -> 1FFF FFFF	Control Space		
2000 0000 -> 2FFF FFFF	AFX Frame buffer	SBus Slot 0	
3000 0000 -> 3FFF FFFF		SBus Slot 1	SBus Slave Select 0
4000 0000 -> 4FFF FFFF		SBus Slot 2	SBus Slave Select 1
5000 0000 -> 5FFF FFFF		SBus Slot 3	SBus Slave Select 2
6000 0000 -> 6FFF FFFF		SBus Slot 4	SBus Slave Select 3
7000 0000 -> 7FFF FFFF		SBus Slot 5	SBus Slave Select 4

3.3 The Shared Memory

The microSPARC-II chip interfaces directly to a 64-bit wide DRAM on one side and to the SBus on the other side. The shared DRAM is 64-bit wide with one parity bit for 32-bit data. The SPARC CPU-5V provides 16- or 64-Mbyte DRAM which is assembled on the board itself. There are 4-Mbit devices used to realize 16 Mbytes and there are 16-Mbit devices to realize 64 Mbytes.

The microSPARC-II chip supports up to eight memory banks (bank 0 to bank 7). Two of the eight memory banks are used on the SPARC CPU-5V base board (bank 0 and bank 1). The signals for the remaining memory banks are routed to the memory module connectors for module #1 and module #2.

Memory connector for memory module #1 supports banks 2, 3, 4 and 5. Memory connector for memory module #2 supports banks 4, 5, 6 and 7. Memory modules with up to 4 memory banks can be used. As shown in the table below, the memory bank structure is organized so that memory modules with a bank count from 1 to 4 (if available) can be used in any combination. Each module has up to 4 banks, only up to 8 banks in total are allowed. A memory module can contain bank A, or banks A and B, or banks A, B and C, or bank A, B, C and D.

Table 23: Bank Selection

Bank Select from Processor	Base-board		Module on Connector #1				Module on Connector #2			
	Bank A	Bank B	Bank A	Bank B	Bank C	Bank D	Bank A	Bank B	Bank C	Bank D
0	x									
1		x								
2			x							
3				x						
4					x					x
5						x			x	
6								x		
7							x			

The shaded area above shows an example of how the banks are selected by the processor. In other words, the processor can select Bank B of the module on connector #1 by its own bank select 3. **CAUTION:** Do not connect more than one physical memory bank to one bank select from the processor. In other words, you can connect either bank C of the module on connector # 1, or bank D of module on connector # 2 to bank select 4, or you can connect either bank D of the module on connector # 1, or bank C of the module on connector # 2 to bank select 5.

The table below shows the base board memory capacity and the memory banks used by the microSPARC-II.

Table 24: CPU-5V Memory Banks

CPU-5V Memory Capacity	Memory Banks 0 and 1 are Used
16 Mbytes	X
64 Mbytes	X

3.4 Memory Module MEM-5

It is possible to upgrade your CPU-5V board with one or two memory modules (the remaining banks 2 to 7). The memory modules are available in different variants.

The MEM-5 provides 16- or 64-Mbyte DRAM. There are 4-Mbit devices used to realize 16 Mbytes and there are 16-Mbit devices to realize 64 Mbytes.

The table below shows the board memory capacity and the memory banks used on the microSPARC-II.

To understand the structure of the memory make sure you read "The Shared Memory" on page 55.

Table 25: MEM-5 Memory Banks

MEM-5 Memory Capacity	Memory Banks A and B are Used
16 Mbytes	X
64 Mbytes	X

Installing the memory modules is described in the document "How to Install MEM-5."

3.5 SBus Participants

There are two SBus slots located on the component side of the board. SBus Slot # 2 is located at connector P3 and SBus Slot # 3 is located at connector P4. A diagram of the board is located in the figure "Diagram of the CPU-5V (Top View)" on page 10.

The microSPARC-II chip supports up to 5 SBus slots plus an additional "master-only" slot. The SBus controller is inside the microSPARC-II chip.

The following table shows the microSPARC-II physical address map including all of its SBus slots and their functions on the SPARC CPU-5V.

3.5.1 Address Mapping for SBus Slots on the SPARC CPU-5V

Table 26: Physical Memory Map of SBus on SPARC CPU-5V

Address	Function	SBus Slot #	Select #
2000 0000 -> 2FFF FFFF	AFX Frame buffer	SBus Slot 0	
3000 0000 -> 3FFF FFFF	VMEbus Interface	SBus Slot 1	SBus Slave Select 0
4000 0000 -> 4FFF FFFF	SBus Module P3 VMEbus Interface	SBus Slot 2	SBus Slave Select 1
5000 0000 -> 5FFF FFFF	SBus Module P4 VMEbus Interface	SBus Slot 3	SBus Slave Select 2
6000 0000 -> 6FFF FFFF	VMEbus Interface	SBus Slot 4	SBus Slave Select 3
7000 0000 -> 77FF FFFF	NCR89C105 (SLAVIO) chip	SBus Slot 5	SBus Slave Select 4
7800 0000 -> 7DFF FFFF	NCR89C100 (MACIO) chip	SBus Slot 5	SBus Slave Select 4
7E00 0000 -> 7FFF FFFF	VMEbus Interface	SBus Slot 5	SBus Slave Select 4

If there are no SBus modules installed, SBus Slot 1, 2, 3 and 4, together with SBus Slot 5 address range: 7E00 0000-7FFF FFFF, are available for the VMEbus Interface.

3.6 NCR89C100 (MACIO)

The NCR89C100 is located on SBus Slave Select 4 at physical address $7800 0000. This chip drives the SCSI, Ethernet and Centronics parallel port.

The NCR89C100 SBus master integrates high performance I/O macrocells and logic including an Ethernet controller core, a fast 53C9X SCSI core, a high-speed parallel port, a DMA2 controller and an SBus interface.

The Ethernet core is compatible with the industry standard 7990 Ethernet controller. The SCSI core is a superset of the industry standard NCR53C90A which has been modified to support fast SCSI. The uni/bi-directional parallel port is Centronics compliant and can operate in either programmed I/O or DMA mode.

The DMA2 block comprises the logic used to interface each of these functions to the SBus. It provides buffering for each of the functions. Buffering takes the form of a 64-byte data cache and 16-bit wide buffer for the Ethernet channel, and a 64-byte FIFO for both the SCSI channel and the parallel port. The DMA2 incorporates as improved cache and FIFO draining algorithm which allows better SBus utilization than previous DMA implementations.

3.6.1 Features of the NCR89C100 on the SPARC CPU-5V

- Fast 8-bit SCSI
  - Supports fast SCSI mode
  - Backward compatible to 53C90A
- 7990-compatible Ethernet
- Parallel Port
  - I/O or DMA programmable modes
  - Centronics compatibility
- LS64854-compatible DMA2 Controller
- Glueless SBus Interface clocked with 21.25 MHz @ 85 MHz processor frequency
- Glueless SBus Interface clocked with 22.00 MHz @ 110 MHz processor frequency
- Concurrently supports:
  - 10 MB/sec SCSI transfers
  - 3.4 MB/sec Parallel port transfers
  - 1.25 MB/sec Ethernet transfers
- 64-byte FIFO for SCSI and Parallel Port data
- Supports SBus burst modes
  - 4-word, 8-word and "no/burst"
- Packaged in 160-pin PQFP

For further information about the NCR89100, please see *NCR SBus I/O Chipset Data Manual.*

3.6.2 SCSI

The SCSI interface provides a standard interface to a wide variety of mass storage devices, such as hard disks, tapes and CD-ROMs. The SCSI transfers up to 10 Mbytes per second.

The SPARC CPU-5V board's SCSI is realized via the NCR89C100. The NCR89C100 has on-chip 48 mA drivers and therefore provides direct drive of single-ended SCSI bus. The SCSI core is a superset of the industry standard NCR53C90A which has been modified to support fast SCSI.

The SCSI interface is single ended and supports "TERMPWR". The NCR89C100 DMA2 core is able to transfer the data to and from the shared main memory.

All signals of the SCSI interface are routed to a standard connector on the front panel and to the VME P2 connector. This 2nd connection is compatible to the CPU-2CE, CPU-3CE and the CPU-5CE. Please see the "VME P2 Connector Pinout" on page 41 where the SCSI signals on the VME P2 connector are shown.

3.6.3 SCSI Termination

When only one of the SCSI connectors is used, the other one is an end point. In that case, the SCSI bus must be terminated near the unused connector. This is supported on the CPU-5V board through one termination at the front panel and one termination at VME P2.

The front panel termination of the SCSI can be configured via the on-board switch SW6-1. If SW6-1 is ON, the front panel termination is disabled. If SW6-1 is OFF, the termination is automatic. Automatic means that when a SCSI cable is plugged into the front panel connector, the termination is automatically disabled. When there is no SCSI cable plugged into the front panel, the termination is automatically enabled.

The VME P2 termination of the SCSI interface can be enabled or disabled via the switch SW6-2. If SW6-2 is OFF, the termination is enabled. If SW6-2 is ON, the termination is disabled.

Please see "Diagram of the CPU-5V (Top View)" on page 10 for the location of the switches on the board.

CAUTION

When installing the SPARC CPU-5V in a MICROFORCE chassis, please first disable the SCSI termination by switching SW6-1 and SW6-2 to ON.

3.6.4 Ethernet

The NCR89C100 DMA controller enables the Ethernet interface to transfer data to and from the shared main memory. The Ethernet core is register level compatible with the AMD Am7990, Revision F, standard Ethernet controller, which is capable of transferring Ethernet data up to 10 Mbit/sec.

An 8-pin configuration switch matrix selects whether the Ethernet interface is available via the front panel or the VME P2 connector. By default, the Ethernet Interface is available through the front panel connector with the I/O bridge plugged into connectors B12 and B13.
It is possible to have the Ethernet interface accessible on the VME P2 connector by changing the default configuration. In order that the Ethernet interface is accessible from the VMEbus P2 connector, the I/O bridge array must be plugged into connectors B11 and B12.

Please see the figure "Ethernet Interface via Front Panel" on page 19 for information about changing the Ethernet configuration.

CAUTION

When the Ethernet is configured via P2, do not connect the Ethernet at the front panel.

3.6.5 Parallel Port

The parallel port is centronics compliant and provides uni/bi-directional communication. It operates in either programmed I/O or DMA mode.

The default configuration enables the floppy interface via the VME P2 connector, with the configuration switch matrix plugged into B1 and B2. This means, of course, that by default the parallel port interface is not available via the VMEbus P2 connector. It is the floppy disk interface which is available on the VMEbus P2 connector by default.

In order to configure the parallel port to be accessible from the VMEbus P2 connector, the switch matrix must be plugged into connectors B2 and B3. Please see the figure "Floppy Interface Via VME P2 Connector" on page 18 for information about changing the configuration.

3.7 NCR89C105 (SLAVIO)

The NCR89C105 SBus slave integrates most of the 8-bit system I/O functions including two dual channel 8530-compatible serial controllers, a high speed 8277AA-1-compatible floppy disk controller, counter/timers, interrupt controllers, and system reset logic. It also provides an SBus interface for several other byte-wide peripherals through an external expansion bus.

The primary serial controller is 8530-compatible and can be used as two general purpose serial ports.

The second serial controller is subset of the 8530 standard and is dedicated for the keyboard/ mouse connection.

The 8277AA-1-compatible floppy disk controller supports up to 1 Mbit/sec data transfer rate.

To reduce part count and system cost, a glueless interface to the SBus is provided. The slave I/O also includes an 8-bit expansion bus with control to support RTC/NVRAM, EPROM and generic 8-bit devices externally.

3.7.1 Features of the NCR89C105 on the SPARC CPU-5V

- Dual-channel serial ports (8530-compatible)
- Keyboard/ mouse port
- 82077AA-1 floppy disk controller
- 8-bit expansion bus for Flash Memory/TOD/NVRAM
- Glueless SBus interface clocked with 21.25 MHz @ 85 MHz and 22.0 MHz @ 110 MHz processor frequency
- Interrupt controller
- System reset control
- Programmable 22-bit counters & timers
- Auxiliary I/O registers
- Packaged in 160-pin PQFP

For further information about the NCR89100, please refer to the *NCR SBus I/O Chipset Data Manual.*

3.7.2 Address Map of Local I/O Devices on SPARC CPU-5V

The following table lists the physical addresses for all local I/O devices and the accesses permitted ((B)yte, (H)alf Word and (W)ord).

Table 27: NCR89C105 Chip Address Map

Physical Address	Device	Access
7000 0000 -> 70FF FFFF	Boot Flash Memory and User Flash Memory	B,H,W
7100 0000 -> 711F FFFF	Keyboard, Mouse, and Serial Ports	B
7100 0000	Mouse Control Port	
7100 0002	Mouse Data Port	
7100 0004	Keyboard Control Port	
7100 0006	Keyboard Data Port	
7110 0000	TTYB Control Port	
7110 0002	TTYB Data Port	
7110 0004	TTYA Control Port	
7110 0006	TTYA DATA Port	
7120 0000 -> 712F FFFF	RTC/NVRAM	B,H,W
7130 0000 -> 7137 FFFF	Boot Flash Memory and User Flash Memory Programming	B
7138 0000 -> 713F FFFF	Additional Registers	B
7140 0000 -> 714F FFFF	Floppy Controller	B
7140 0002	Digital Output Register (DOR)	
7140 0004	Main Status Register (MSR, Read Only)	
7140 0004	Datarate Select Register (DSR, Write Only)	
7140 0005	FIFO	
7140 0006	Reserved (Test mode select)	
7140 0007	Digital Input Register (DIR, Read Only)	
7140 0007	Configuration Control Register (CCR, Write Only)	
7150 0000 -> 7170 0000	Reserved	
7180 0000	89C105 Configuration Register	B
7190 0000 -> 719F FFFF	Auxiliary I/O Registers	B
7190 0000	Aux 1 Register (Miscellaneous System Functions)	
7191 0000	Aux 2 Register (Software Powerdown Control)	

3.7.3 Serial I/O Ports

The two serial I/O ports are available on the front panel via one 26-pin shielded connector .

Both of the two ports are available via the VMEbus P2 connector, each with four signals (RXD, TXD, RTS, CTS). Each of the two serial I/O ports are independent full-duplex ports.

The 8530 SCC block is functionally compatible with the standard NMOS 8530 and therefore provides two fully independent full-duplex ports.

The physical address map for the serial ports is shown in "NCR89C105 Chip Address Map" on page 63.

3.7.4 RS-232, RS-422 or RS-485 Configuration

Both serial ports can be configured as RS-232, RS-422 or RS-485. By default, the FH-002 hybrid module is installed for RS-232 operation.

In order to simplify changing the serial interfaces, FORCE COMPUTERS has developed RS-232, RS-422 and RS-485 hybrid modules: the FH-002, FH-003 and FH-005. These 21-pin SIL modules are installed in sockets so that they may be easily changed to meet specific application needs.

To change the configuration of serial port A, insert the respective hybrid in socket J59. To change the configuration of serial port B, insert the respective hybrid in socket J60. For the position of the sockets on the board, please see "Diagram of the CPU-5V (Top View)" on page 10.

Table 28: RS-232, RS-422 or RS-485 Configuration

Hybrid	Configuration	Socket for Serial Port A	Socket for Serial Port B	Default
FH-002	RS-232	J59	J60	*
FH-003	RS-422	J59	J60	
FH-005	RS-485	J59	J60	

3.7.5 RS-232 Hardware Configuration

The serial ports A and B are configured by default for RS-232 operation. The following individual I/O signals are available for serial ports A and B on the front panel connector.

Table 29: Serial Ports A and B Pinout List (RS-232)

Pin		Transmitted Signals	Pin		Received Signals
2	14	TXD-Transmit Data	3	16	RXD-Receive Data
4	19	RTS-Request to Send	5	13	CTS-Clear to Send
7	23	Ground	6	21	SYNC
20	11	DTR-Data Terminal Ready	8	12	DCD-Data Carrier Detect
24	25	TRXC-DTE Transmit Clock	15	18	TRXD-DCE Transmit Clock
			17	22	RTXC-DCE Receive Clock

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

The table below shows the switch settings for each port.

Table 30: Switch Settings for Ports A and B (RS-232)

Port A	Port B	Default	Function for RS-232
SW4-1	SW5-1	ON	TRXC is available on front panel connectors, pin 24
SW4-3	SW5-3	OFF	RTS is available on front panel connectors, pin 4
SW4-2	SW5-2	OFF	CTS is available on front panel connectors, pin 5

Please see the "Diagram of the CPU-5V (Bottom View)" on page 11 for the location of the switches on the board.

3.7.6 RS-422 Hardware Configuration

It is possible to reconfigure serial ports A and B for RS-422 operation. In order to configure the serial ports to RS-422, the hybrid module FH-003 must be used. Termination resistors can be installed to adapt various cable lengths and reduce reflections.

Table 31: Serial Ports A and B Pinout List (RS-422)

Pin		Transmitted Signals	Pin		Received Signals
24	25	TXD+ Transmit Data	20	11	RXD+ Receive Data
8	12	TXD- Transmit Data	7	23	RXD- Receive Data
4*	19*	RTS+ Request to Send	2*	14*	CTS+ Clear to Send
3*	16*	RTS- Request to Send	5*	13*	CTS- Clear to Send
4*	19*	TRXC+ Transmit Clock	2*	14*	RTXC+ Receive Clock
3*	16*	TRXC- Transmit Clock	5*	13*	RTXC- Receive Clock

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

* Signals RTS and TRXC can be switched so that they are available on connector pins 3 and 4 (16, 19). Signals CTS and RTXC can also be switched so that they are available on connector pins 2 and 5 (14, 93). This is done by switch SW4 for port A and by switch SW5 for port B.

The table on the next page shows the corresponding switch settings.

Table 32: Switch Settings for Ports A and B (RS-422)

Port A	Port B	Configuration	Function for RS-422
SW4-1	SW5-1	ON	ON for RS-422
SW4-3	SW5-3	ON	TRXC +/- on front panel connectors, pins 3/16 and 4/19 available
SW4-3	SW5-3	OFF	RTS +/- on front panel connectors, pins 3/16 and 4/19 available
SW4-2	SW5-2	OFF	CTS +/- on front panel connectors, pins 2/14 and 5/13 available
SW4-2	SW5-2	ON	RTXC +/- on front panel connectors, pins 2/14 and 5/13 available

Please see the "Diagram of the CPU-5V (Bottom View)" on page 11 for the location of the switches on the board.

3.7.7 RS-485 Hardware Configuration

It is possible to reconfigure serial ports A and B to be RS-485 compatible by using the hybrid module FH-005.

The following I/O signals are available on the front panel connectors of both serial ports.

Table 33: Serial Ports A and B Pinout List (RS-485)

Pin		Signals
7	23	RXTX+ Receive/Transmit Data
20	11	RXTX- Receive/Transmit Data

The pinout for serial port A is shown in the white area and the pinout for serial port B is shown in the grey area.

The Receive-Enable (REN) and Transmit-Enable (TEN) of the hybrid module FH-005 are controlled via the NCR89C105 serial I/O signals DTR (REN) and RTS(TEN).

The following table shows the corresponding switch settings

Table 34: Switch Settings for Ports A and B (RS-485)

Port A	Port B	Configuration	Function for RS-485
SW4-1	SW5-1	OFF	RTS functions as TEN
SW4-3	SW5-3	OFF	No function for RS-485
SW4-2	SW5-2	OFF	No function for RS-485

Please see the "Diagram of the CPU-5V (Bottom View)" on page 11 for the location of the switches on the board.

3.7.8 Keyboard and Mouse Port

The keyboard and mouse port is available on the front panel via an 8-pin mini DIN connector and on VME P2.

The serial port controller used for the keyboard and mouse port is compatible with the NMOS 8530 controller.

The pinout of the keyboard and mouse port is described in Section 2, Installation.

The physical address for the keyboard and mouse port is shown in "NCR89C105 Chip Address Map" on page 63.

3.7.9 Floppy Disk Interface

The floppy disk interface is available on the P2 connector by default, with the configuration switch matrix plugged into B1 and B2. The floppy disk configuration is described in the chapter "Parallel Port or Floppy Interface via VME P2 Connector" on page 18.

The floppy disk interface is 82077AA-1 compatible. It is able to transfer data rates of 250, 300, 500 Kbytes/sec, and 1 Mbyte/sec.

The floppy disk controller block is functionally compatible with the Intel 82077AA-1. It integrates drivers, receivers, data separator, and a 16-byte bidirectional FIFO. The floppy disk controller supports all standard disk formats (typically 720 K and 1.44 M floppies). It is also compatible with the 2.88 MB floppy format.

3.7.10 8-Bit Local I/O Devices

The following local I/O devices are interfaced via the NCR89C105

Table 35: 8-Bit Local I/O Devices

Function	IRQ	Physical Base Address
Boot Flash Memory Device # 1 256 Kbyte (default)	No	$7000 0000-> $7003 FFFF
Boot Flash Memory Device # 2 256 Kbyte (default)	No	$7004 0000 -> $7007 FFFF
User Flash Memory 1 Mbyte Device # 1	No	$7010 0000 -> $701F FFFF
User Flash Memory 1 Mbyte Device # 2	No	$7020 0000 -> $702F FFFF
RTC/NVRAM	No	$7120 0000 -> $712F FFFF
Flash Memory Programming Area	No	$7130 0000 -> $7137 FFFF
Additional Registers	No	$7138 0000 -> $713F FFFF

3.7.11 Boot Flash Memory

The boot flash memory consists of two 2-Mbit or 4-Mbit flash memory devices. In the default configuration, there are two 2-Mbit devices installed. The 4-Mbit devices are an additional assembly option.

The boot flash memory devices can be reprogrammed on-board and can also be write protected via hardware switch SW6-3.

When SW6-3 is ON, write accesses are possible. The devices are write protected when SW6-3 is OFF.

The boot flash memory devices are installed in sockets at location J26 (device #1) and J22 (device #2). This permits programming them in a standard programmer. This may be necessary if the power fails during reprogramming. In this case, the contents of the Boot Flash Memory would be lost and the board would not be able to boot.

Table 36: Boot Flash Memory Capacity

Devices	Count	Capacity	Default
256 K * 8	2	512 Kbyte	X
512 K* 8	2	1 Mbyte	

The on-board programming of the boot flash memory devices requires setting some bits in the Flash Memory Programming Voltage Control Register and Flash Memory Programming Control Register 1 and 2. These registers are shown on the following pages.

3.7.12 User Flash Memory

The user flash memory area consists of a maximum of two 8-Mbit flash memory devices, providing a capacity of 2 Mbytes. The capacity of user flash memory is outlined in the product nomenclature, which can be seen in the table "Ordering Information" on page 6.

This area can be used to store ROMable operating systems as well as application specific code.

Table 37: User Flash Memory Capacity

Devices	Count	Capacity
1M*8	2	2 Mbyte

The user flash memory devices can be reprogrammed on-board and can also be write protected via hardware switch SW6-4. When SW6-4 is ON, write accesses are possible. When SW6-4 is OFF, the devices are write protected.

The on-board programming of the user flash memory devices requires setting some bits in the Flash Memory Programming Voltage Control Register and Flash Memory Programming Control Register 1 and 2. These registers are shown on the following pages.

3.7.13 Programming the On-board Flash Memories

Both areas of flash memories, the Boot area and the User area, can be reprogrammed on-board. Please see "Flash Memory Support" on page 162 for details about programming the on-board memories.

The address range in which the flash memory devices can be programmed is located in a 512 Kbyte page (programming window) of the Generic Port area of the NCR89C105 (SLAVIO). The physical address range is $7130 0000 .. $7137 FFFF.

Please note the following steps for programming the on-board flash memory devices.

- Disable hardware write protection in order to program the flash memory devices. The switch SW6-3 must be ON in order to program the boot flash memory and the switch SW6-4 must be ON in order to program the user flash memory. For the location of the switches on the board please see "Diagram of the CPU-5V (Bottom View)" on page 11.
- Switch the programming voltage ON by setting the appropriate bit.
- Set address lines A[21:19] to the requested address range. Set the device number of the device to be selected.
- Select either the user flash memory or the boot flash memory for programming.
- After the flash memory devices have been programmed, we recommend that you return to the default settings of SW6-3 and SW6-4. This protects the flash memory devices from being programmed by accident.

In order to enable programming and to decide which area is to be mapped to the programming window, the following six bits VPP_ON, A[21:19], SEL_ROM and SEL_BOOT are used to control this.

3.7.13.1 Flash Memory Programming Voltage Control Register

To enable the programming of the flash memory devices, the +12V programming voltage must be switched ON. This is done by setting bit VPP ON in the Flash Memory Programming Voltage Control Register.

Initialization: VPP ON is cleared on reset. This inhibits the programming of the flash memory devices.

Physical Address: 7138 000A$_{16}$

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	VPP ON

VPP_ON (**RW**) This bit is used to turn the +12V programming voltage for **all** flash memories on or off. When the bit is set (1) then the programming voltage is turned on; and the programming voltage is turned off by clearing (0) this bit.

3.7.13.2 Flash Memory Programming Control Register 1

Physical Address: 7138 0002_{16}

7	6	5	4	3	2	1	0
1	1	1	1	A[21:19]			SEL ROM

A[21:19] (**RW**) The outputs of these three register bits are directly connected with the address pins A21, A20, and A19 of both USER flash memories which allows to address flash memories with up to 4 Mbyte size.
Because the flash memories are accessible only in the physical range 7130.0000_{16} to $7137.FFFF_{16}$, software has to modify these bits to make a specific 512 Kbyte *page* available in this address range.

A[21:19]	Page	Accessible Area of Flash Memory (offset)
000_2	0	00.0000_{16} ... $07.FFFF_{16}$
001_2	1	08.0000_{16} ... $0F.FFFF_{16}$
010_2	2	10.0000_{16} ... $17.FFFF_{16}$
011_2	3	18.0000_{16} ... $1F.FFFF_{16}$
100_2	4	20.0000_{16} ... $27.FFFF_{16}$
101_2	5	28.0000_{16} ... $2F.FFFF_{16}$
110_2	6	30.0000_{16} ... $37.FFFF_{16}$
111_2	7	38.0000_{16} ... $3F.FFFF_{16}$

SEL_ROM (**RW**) This bit and the SEL_BOOT bit in the Flash Memory Programming Control Register 2 are used to select one of four flash memory devices to be accessible in the physical address range 7130.0000_{16} to $7137.FFFF_{16}$.

3.7.13.3 Flash Memory Programming Control Register 2

Physical Address: 7138 0009$_{16}$

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	SEL BOOT

SEL_BOOT (**RW**) This bit and the SEL_ROM bit in the Flash Memory Programming Control Register 1 are used to select one of four flash memories to be accessible in the physical address range 7130.0000$_{16}$ to 7137.FFFF$_{16}$. The following table shows all possible configurations:

		SEL_BOOT	
		0	1
SEL_ROM	0	first USER flash memory device accessible	first BOOT flash memory device accessible
	1	second USER flash memory device accessible	second BOOT flash memory device accessible

3.7.14 RTC/NVRAM

The MK48T08 combines an 8 K x 8 full CMOS SRAM, a bytewide accessible Real Time Clock, a crystal, and a long-life lithium carbon monofluoride battery, all in a single plastic DIP package. The MK48T08 is a nonvolatile pin and functionally equivalent to any Jedec standard 8 K x 8 SRAM

For a detailed description of the RTC/NVRAM, please see the respective Data Sheet.

3.8 VMEbus Interface

The CPU-5V utilizes the FGA-5000 chip to provide fully SBus and VMEbus compliant interfaces. Supported functions include master and slave data transfer capabilities, VMEbus interrupt handling and arbitration functions. Additional VMEbus utility functions and a special loop-back cycle for stand-alone testing of the interface are provided.

Features of the FGA-5000

- VMEbus Master Interface
- VMEbus Slave Interface
- DMA Controller
- Interrupts
- VMEbus Arbiter
- FORCE Message Broadcast
- Mailboxes and Semaphores
- Reset Functions
- System Controller Functions
- Timers

A complete description of the FGA-5000 chip is found in the *FGA-5000 Technical Reference Manual,* available from FORCE COMPUTERS.

3.8.1 Address Mapping for the VMEbus Interface FGA-5000

The table below lists the physical addresses of the VMEbus interface FGA-5000.

Table 38: Physical Memory Map of VMEbus Interface on SPARC CPU-5V

Address	Function	SBus Slot #	Select #
3000 0000 -> 3FFF FFFF	VMEbus Interface	SBus Slot 1	SBus Slave Select 0
4000 0000 -> 4FFF FFFF	VMEbus Interface (SBus Module)	SBus Slot 2	SBus Slave Select 1
5000 0000 -> 5FFF FFFF	VMEbus Interface (SBus Module)	SBus Slot 3	SBus Slave Select 2
6000 0000 -> 6FFF FFFF	VMEbus Interface	SBus Slot 4	SBus Slave Select 3
7E00 0000 -> 7FFF FFFF	VMEbus Interface	SBus Slot 5	SBus Slave Select 4 (SB_SEL<5>)

The FGA-5000 can be selected with up to six SBus select input signals SSEL<5..0>. The microSPARC-II CPU chip supports only 5 select signals SLVSEL<4..0>. The remaining select signal of the FGA-5000 can be used to expand the VMEbus address area.

The I/O chips NCR89C100 (MACIO) and NCR89C105 (SLAVIO) are selected in SBus Slot 5 by SBus Slave Select 4. The upper part in this range is not used by these chips. So we decided to split the SBus Slave Select 4 into two signals: SB_SEL<4> and SB_SEL<5>. Now the I/O chips are selected by SB_SEL<4> and the VMEbus interface FGA-5000 is selected by SB_SEL<5>. With this expansion, the VMEbus interface FGA-5000 shares SBus Slot 5 with the I/O chips and can use an additional address range of up to 32 Mbyte in this SBus Slot.

On the base board, SBus Slot 2 (SBus Slave Select 1) and SBus Slot 3 (SBus Slave Select 2) are provided for SBus modules. When no SBus modules are installed, you can use these SBus slots to expand the VMEbus address range again. In this case, you gain 256 Mbyte with every additional SBus Slot.

When using all address range resources for the VMEbus interface the microSPARC-II CPU can access the VMEbus interface FGA-5000, internal registers and VMEbus slaves, in an address area of 1056 Mbyte (1GByte + 32 Mbyte).

3.8.2 Adaptation of the FGA-5000

Some aspects of the VMEbus interface chip FGA-5000 require a small amount of glue logic be built around this chip in order to use the chip on this base board.

In the case where the FGA-5000 is not VMEbus master, the VMEbus input signal BERR has not been directly routed to the FGA-5000. This masking is required for normal VMEbus transfers. However, during FMB transfers the VMEbus slave should see the input signal BERR.

When the FGA-5000 is one of several selected slaves during an FMB cycle, and one or more of the other slaves acknowledges the transfer with Bus Error, then the FGA-5000 doesn't recognize that the message is invalid. In order to enable the software to discard this invalid message, additional registers have been implemented in a separate programmable device on the base board.

How to access and interpret the contents of the added registers can be found in the chapter "Additional Registers" on page 90.

For information about the FMB implementation in the FGA-5000 chip, please refer to the *FGA-5000 Technical Reference Manual.*

3.8.3 VMEbus SYSRESET Enable/Disable

3.8.3.1 SYSRESET Input

An external SYSRESET generates an on-board RESET in the default switch setting, i.e., SW7-3 is ON. When SW7-3 is OFF, the external SYSRESET does not generate an on-board RESET.

3.8.3.2 SYSRESET Output

An on-board RESET drives the SYSRESET signal to the VMEbus to low in the default switch setting, i.e., SW7-4 is ON. When SW7-4 is OFF, an on-board RESET doesn't drive the SYSRESET signal to the VMEbus to low.

CAUTION

Do not switch SW7-4 (SYSRESET output) to ON and SW8-2 (VMEbus Slot-1 device) to OFF at the same time.

The VMEbus Specification requires that if SYSRESET is driven, the SYSRESET signal shall be driven low for at least 200 ms. However, when the CPU-5V is not a VMEbus Slot-1 device and the SYSRESET output signal is enabled, then the CPU-5V no longer conforms with this rule.

By default, the SYSRESET output is enabled. In this case it generates the SYSRESET signal to the VMEbus.

3.9 On-board Control Registers (System Configuration)

The following table shows the physical address of the registers used for system configuration. The registers are described in their respective functional chapters, for example, the USER LEDs are described in the chapter "Front Panel Status LEDs" on page 85.

Address	Reset Value	Size	Description
7138.0000_{16}	$F0_{16}$	8 bit	USER LED 1 Control Register
7138.0001_{16}	$F0_{16}$	8 bit	USER LED 2 Control Register
7138.0002_{16}	$F0_{16}$	8 bit	Flash Memory Programming Control Register 1
7138.0003_{16}	FX_{16}	8 bit	Rotary Switch Status Register
7138.0004_{16}	XX_{16}	8 bit	Reserved
7138.0005_{16}	XX_{16}	8 bit	Reserved
7138.0006_{16}	XX_{16}	8 bit	Reserved
7138.0007_{16}	XX_{16}	8 bit	Reserved
7138.0008_{16}	FE_{16}	8 bit	Boot ROM Size Control Register
7138.0009_{16}	FE_{16}	8 bit	Flash Memory Programming Control Register 2
$7138.000A_{16}$	FE_{16}	8 bit	Flash Memory Programming Voltage Control Register
$7138.000B_{16}$	XX_{16}	8 bit	Seven- Segment LED Display Control Register
$7138.000C_{16}$	FE_{16}	8 bit	FMB Channel 0 Data Discard Status Register
$7138.000D_{16}$	FE_{16}	8 bit	FMB Channel 1 Data Discard Status Register
$7138.000E_{16}$	XX_{16}	8 bit	reserved
$7138.000F_{16}$	FX_{16}	8 bit	LCA Identification Register

3.10 Front Panel

The Reset and Abort functions, the Hex display, the Rotary switch, and the LEDs are described on the following pages. The pinouts for the connectors shown in grey below are described in Section 2, Installation.

Table 39: Front Panel Layout

Device	Function	Name
Switch	Reset	RESET
Switch	Abort	ABORT
HEX. Display	Diagnostic	DIAG
Rotary Switch	User defined	MODE
LED/LED	RUN/HALT VME Busmaster/SYSFAIL	RUN BM
LED/LED	Software programmable	0 1
MiniDIN Connector	Keyboard/Mouse	KBD
Serial Connector	Serial Interfaces	SERIAL A+B
SCSI Connector	SCSI Interface	SCSI
D-Sub Connector	Ethernet	ETHERNET

3.10.1 RESET and ABORT Keys

The front panel on the SPARC CPU-5V has two mechanical switches which directly influence the system.

3.10.1.1 The RESET Key

The **RESET** key enables the user to reset the whole board. If the board is VMEbus system controller (Slot-1 device), the SYSRESET signal of the VMEbus also becomes active with the RESET key. This resets the complete VMEbus system. The switch SW7-4 can be used to disable driving the VMEbus SYSRESET signal (see "VMEbus SYSRESET Enable/Disable" on page 81). With on-board switch SW7-1, it is possible to deactivate the RESET key. When SW7-1 is ON, the RESET key works and when SW7-1 is OFF, toggling the RESET key has no effect.

Please see also "VMEbus SYSRESET Enable/Disable" on page 81.

3.10.1.2 The ABORT Key

The **ABORT** key on the front panel can be used to generate a nonmaskable interrupt (level 15). The ABORT key function is controlled by switch SW7-2. When SW7-2 is ON, the key works and when SW7-2 is OFF, toggling the ABORT key has no effect. If the ABORT key produces a nonmaskable interrupt, the pending signal can be read in the Miscellaneous Control and Status Register 0 (MCSR0 register). The ABORT Interrupt Request Mapping Register (ABORT_IRQ_MAP) is used to map and enable an interrupt, generated by assertion of the ABKEY signal. Please see the *FGA-5000 Technical Reference Manual* for further information.

3.10.2 Front Panel Status LEDs

There are 4 single LEDs on the front panel.

- The RUN/RESET LED
- The VME BM LED (Bus Master)
- 2 STATUS LEDs

The RUN/RESET LED is either red, green or blinking. This LED is red when any reset signal on the board is active. This LED begins blinking when SB_SEL<4> is inactive for more than 0,5s in order to signal a hang up. In all other cases, this LED is green.

The BM LED reflects all VMEbus master activities on the CPU-5V. When the board accesses the VMEbus, the BM LED lights up green. The BM LED turns red when the CPU-5V is asserting SYSFAIL to the VMEbus.

There are 2 additional STATUS LEDs, which are freely programmable LEDs controlled by accessing registers in the LCA.

3.10.2.1 USER LED 1 Control Register

Physical Address 7138 0000_{16}

7	6	5	4	3	2	1	0
1	1	1	1	BLINK_FREQ		COLOUR	

COLOUR (**RW**) These two bits are used to turn the first USER LED on or off, and to control the colour of the LED. The table below lists all possible values:

COLOUR	Colour of the first USER LED (LED #0)
00_2	USER LED is turned off
01_2	USER LED is turned on and shines green
10_2	USER LED is turned on and shines red
11_2	USER LED is turned on and shines yellow

BLINK_FREQ (**RW**) These two bits control the frequency at which the first USER LED is blinking. The table below lists all possible values and the corresponding blink frequency:

BLINK_FREQ	f_{blink} of the first USER LED (LED #0)
00_2	USER LED is not blinking
01_2	USER LED is blinking at 1/2 Hz
10_2	USER LED is blinking at 1 Hz
11_2	USER LED is blinking at 2 Hz

3.10.2.2 USER LED 2 Control Register

Physical Address 7138 0001_{16}

7	6	5	4	3	2	1	0
1	1	1	1	BLINK_FREQ		COLOUR	

COLOUR (**RW**) These two bits are used to turn the second USER LED on or off, and to control the colour of the LED. The table below lists all possible values:

COLOUR	Colour of the second USER LED (LED #1)
00_2	USER LED is turned off
01_2	USER LED is turned on and shines green
10_2	USER LED is turned on and shines red
11_2	USER LED is turned on and shines yellow

BLINK_FREQ (**RW**) These two bits control the frequency at which the second USER LED is blinking. The table below lists all possible values and the corresponding blink frequency:

BLINK_FREQ	f_{blink} of the second USER LED (LED #1)
00_2	USER LED is not blinking
01_2	USER LED is blinking at 1/2 Hz
10_2	USER LED is blinking at 1 Hz
11_2	USER LED is blinking at 2 Hz

3.10.3 Diagnostic LED (Hex Display)

A freely programmable LED display on the front panel provides diagnostic features. It can be accessed via the Seven Segment LED Display Control Register.

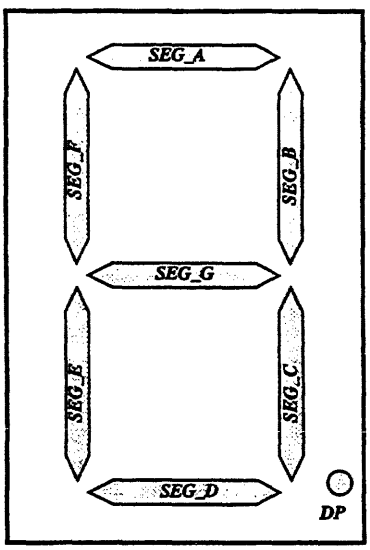
3.10.3.1 Seven Segment LED Display Control Register

Physical Address 7138 000B$_{16}$

7	6	5	4	3	2	1	0
DP	SEG_G	SEG_F	SEG_E	SEG_D	SEG_C	SEG_B	SEG_A

The following figure shows the hex display with the segments named in accordance to their bits in the Seven Segment LED Display Control Register. To switch a specific segment on, the corresponding bit must be set to one.

FIGURE 13. Segments of the Hex Display



3.10.4 Rotary Switch

The CPU-5V provides an additional rotary switch for user selectable settings. See the "Diagram of the CPU-5V (Top View)" on page 10 for the position of the rotary switch on the board. It is a hexadecimal rotary switch, decoded with 4 bits. The status of the rotary switch can be read in the Rotary Switch Status Register.

The table below shows the rotary switch settings and the corresponding values of the bits ROT[3..0], which you can read from the Rotary Switch Status Register.

3.10.4.1 Rotary Switch Status Register

Physical Address 7138 0003$_{16}$

7	6	5	4	3	2	1	0
1	1	1	1	ROTARY_SWITCH[3:0]			

ROTARY_SWITCH[3:0] (R) These bits reflect the current state of the rotary switch. On the SPARC CPU-5V the rotary switch is connected in such a way to the LCA that the signals ROTARY_SWITCH[3:0] are inverted!

Rotary Switch	ROTARY SWITCH [3:0]	Rotary Switch	ROTARY SWITCH [3:0]
0_{16}	1111_2	8_{16}	0111_2
1_{16}	1110_2	9_{16}	0110_2
2_{16}	1101_2	A_{16}	0101_2
3_{16}	1100_2	B_{16}	0100_2
4_{16}	1011_2	C_{16}	0011_2
5_{16}	1010_2	D_{16}	0010_2
6_{16}	1001_2	E_{16}	0001_2
7_{16}	1000_2	F_{16}	0000_2

3.11 Additional Registers

The following additional registers are provided on the CPU-5V to increase functionality.

3.11.1 FMB Channel 0 Data Discard Status Register

FMB channel 0 consists of an 8-stage FIFO and so does the FMB Channel 0 Data Discard Status Register. Read accesses to this register switch the internal read pointer one step ahead in the FIFO. Whenever your software needs to perform elementary functions as such, we recommend coordinating accesses to this register and the related FGA-5000 registers so that synchronisation of both FIFOs be not broken.

Physical Address 7138 000C$_{16}$

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	MSG VALID

MSG_VALID (R) The state of this bit indicates whether to discard the data in the FMB channel 0 of the SPARC FGA-5000. When the bit is cleared (0) then the data in the FMB channel **must** be discarded. In the case that this bit is set (1) the data in the FMB channel is valid.

3.11.2 FMB Channel 1 Data Discard Status Register

Physical Address 7138 000D$_{16}$

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	MSG VALID

MSG_VALID (R) The state of this bit indicates whether to discard the data in the FMB channel 1 of the SPARC FGA-5000. When the bit is cleared (0) then the data in the FMB channel **must** be discarded. In the case where this bit is set (1) the data in the FMB channel is valid.

A complete description of the FGA-5000 chip is found in the *FGA-5000 Technical Reference Manual,* available from FORCE COMPUTERS.

SECTION 4 OpenBoot

4. Software

4.1 OpenBoot

This section describes the enhancements to the standard OpenBoot firmware that have been done for the SPARC CPU-5V. For a description of standard OpenBoot firmware features, please see the *OPEN BOOT PROM 2.0 MANUAL SET.*

Besides the commands already provided by the standard OpenBoot firmware, the OpenBoot firmware available on the SPARC CPU-5V includes further words for the following:

- accessing and controlling the VMEbus Interface,
- accessing and programming available flash memories,
- controlling the operating mode of the Watchdog Timer, and
- making use of the Diagnostics.

The following subsections describe these words in detail, and examples are given when it seems necessary to convey the usage of a particular or a group of words. In general, each word is described using the notation stated below:

`name` (*stack-comment*) *description*

The `name` field identifies the name of the word being described.

The stack parameters passed to and returned from a word are described by the *stack-comment* notation — enclosed in parentheses —, and show the effect of the word on the evaluation stack. The notation used is:

parameters before execution — parameters after execution

The parameters passed and returned to the word are separated by the "—".

The *description* body describes the semantics of the word and conveys the purpose and effect of the particular word.

The OpenBoot ported to the SPARC CPU-5V is based upon the OpenBoot 2.15 obtained from Sun Microsystems.

4.2 VMEbus Interface

The VMEbus Interface on the SPARC CPU-5V consists of FORCE COMPUTERS' **SPARC FGA-5000** (VSI) chip. The FORCE Gate Array-5000 is a VMEbus to SBus interface chip.

4.2.1 Generic Information

The variables — which are declared as **`value`** — described below are used to retrieve generic information about the VMEbus interface:

`vsi-va` (— *vaddr*) returns the virtual base address *vaddr* of the registers included in the SPARC FGA-5000.

`vsi-pa` (— *paddr*) returns the physical base address *paddr* of the registers included in the SPARC FGA-5000.

`vsi-sbus-slot#` (— *sbus-slot#*) returns the number of the SBus slot *sbus-slot#* where the registers of the SPARC FGA-5000 are accessible.

`vsi-offset` (— *offset*) returns the *offset* within the particular SBus slot at which the registers, included in the SPARC FGA-5000, are accessible.

The base address of the SPARC FGA-5000, which is specified by the values `vsi-sbus-slot#` and `vsi-offset`, may be modified by the command described below:

`vsi-base-addr!` (*offset sbus-slot#* —) sets the base address of the SPARC FGA-5000 according to the given SBus slot number *sbus-slot#* and the offset *offset* within the specified SBus slot. The values *sbus-slot#* and *offset* are stored in the appropriate variables `vsi-sbus-slot#` and `vsi-offset`.
Furthermore, the command sets the variables `vsi-pa` and `vsi-va` according to the given parameters.

On the SPARC CPU-5V the SBus slots are utilized as stated in the table below.

sbus-slot#	SSEL	Address Range	Description
0	—	2000.0000_{16}...$2FFF.FFFF_{16}$	SBus Slot #0 (AFX)
1	0	3000.0000_{16}...$3FFF.FFFF_{16}$	SBus Slot #1 (reserved for VMEbus accesses through the SPARC FGA-5000)
2	1	4000.0000_{16}...$4FFF.FFFF_{16}$	SBus Slot #2 (Sbus Card 1)
3	2	5000.0000_{16}...$5FFF.FFFF_{16}$	SBus Slot #3 (Sbus Card 2)
4	3	6000.0000_{16}...$6FFF.FFFF_{16}$	SBus Slot #4 (Sbus Card 3)
5	4	7000.0000_{16}...$7FFF.FFFF_{16}$	SBus Slot #5 (MACIO,SLAVIO, SPARC FGA-5000 Registers)

`vsi-base-addr@` (— *offset sbus-slot#*) returns the base address of the SPARC FGA-5000 represented by the SBus slot number *sbus-slot#* and the offset *offset* within the specific SBus slot.

4.2.2 Register Addresses

The commands described below are used to obtain the virtual addresses of specific registers in the SPARC FGA-5000:

`vsi-sbus-base` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Base Address Register**.

`vsi-id` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Identification Register**.

`vsi-vme-range` (*range#* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Address Decoding And Translation Register** identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.

`vsi-vme-master-cap` (*range#* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Master Capability Register** identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

`vsi-vme-cap` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Capability Register**.

`vsi-sbus-ssel` (*range#* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Slave Slot Select Register** identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

`vsi-sbus-master-cap` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Master Capability Register**.

`vsi-sbus-retry-time-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Retry Time Control Register**.

`vsi-sbus-rerun-limit-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Rerun Limit Control Register**.

`vsi-swpar` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Write Posting Error Address Register.**

`vsi-vwpar` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Write Posting Error Address Register.**

`vsi-slerr` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Late Error Address Register.**

`vsi-slerr-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SBus Late Error Interrupt Level Select and Enable Register.**

`vsi-iack-emu` (*level* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **IACK Cycle Emulation Register** associated with the given *level.* The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vsi-vme-base` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Base Address Register.**

`vsi-sbus-range` (*range#* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Address Decoding and Translation Register** identified by its range number *range#.* The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

`vsi-vme-ext` (*range#* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Address Extension Register** identified by its range number *range#.* The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.

`vsi-reset-stat` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Reset Source Register.**

`vsi-intr-stat` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Interrupt Status Register.**

`vsi-irq-map` (*level* — *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **VMEbus Interrupt Level Select and Enable Register** associated with the given *level.* The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vsi-mbox-irq-map` (*mailbox# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Mailbox Interrupt Level Select and Enable Register** identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

`vsi-dma-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Interrupt Level Select and Enable Register.**

`vsi-wpe-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Write Posting Error Interrupt Level Select and Enable Register.**

`vsi-arb-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Arbiter Timeout Interrupt Level Select and Enable Register.**

`vsi-wdt-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Watchdog Timer Interrupt Level Select and Enable Register.**

`vsi-acfail-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **ACFAIL Interrupt Level Select and Enable Register.**

`vsi-sysfail-irq-map0` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SYSFAIL Assert Interrupt Level Select and Enable Register.**

`vsi-sysfail-irq-map1` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **SYSFAIL Negate Interrupt Level Select and Enable Register.**

`vsi-abort-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Abort Interrupt Level Select and Enable Register.**

`vsi-arb-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Arbiter Control Register.**

`vsi-req-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Requester Control Register.**

`vsi-bus-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Bus Capture Control Register.**

`vsi-mbox` (*mailbox# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Mailbox Register** identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vsi-mbox-stat` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's

Mailbox Status Register.

`vsi-sem` (*semaphore# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Semaphore Register** identified by its semaphore number *semaphore#*. The value of *semaphore#* may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

`vsi-fmb-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Message Broadcast Control Register.**

`vsi-fmb-irq-map` (*channel# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Message Broadcast Interrupt Level Select and Enable Register** identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

`vsi-fmb-addr` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Message Broadcast Address Register.**

`vsi-fmb-stat` (*channel# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Message Broadcast Status Register** identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

`vsi-fmb-msg` (*channel# — vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Message Broadcast Register** identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Registers.

`vsi-gcsr` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Global Control and Status Register.**

`vsi-wdt-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Watchdog Timer Control Register.**

`vsi-wdt-restart` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Watchdog Restart Register.**

`vsi-mcsr0` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Miscellaneous Control and Status Register 0.**

`vsi-mcsr1` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Miscellaneous Control and Status Register 1.**

`vsi-dma-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Control Register.**

`vsi-dma-mode` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Mode Register.**

`vsi-dma-stat` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Status Register.**

`vsi-dma-src` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Source Address Register.**

`vsi-dma-dest` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Destination Address Register.**

`vsi-dma-cap` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **DMA Capability and Transfer Count Register.**

`vsi-ibox-irq-map` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Interrupt Box Interrupt Level Select and Enable Register.**

`vsi-ibox-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Interrupt Box Control Register.**

`vsi-ibox-addr` (— *vaddr*) returns the virtual address *vaddr* of the SPARC FGA-5000's **Interrupt Box Address Register.**

The following commands are available to get the virtual addresses of the System Configuration Registers.

`sysconfig-va` (— *vaddr*) returns the virtual base address *vaddr* of the System Configuration Registers.

`led1-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the **First User LED Control Register.**

`led2-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the **Second User LED Control Register.**

`flash-ctrl1` (— *vaddr*) returns the virtual address *vaddr* of the **Flash Memory Control Register 1.**

`rotary-switch-stat` (— *vaddr*) returns the virtual address *vaddr* of the **Rotary Switch Status Register.**

`boot-rom-size-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the **Boot ROM Size Control Register.**

`flash-ctrl2` (— *vaddr*) returns the virtual address *vaddr* of the **Flash Memory Control Register 2.**

`flash-vpp-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the **Flash Memory Programming Voltage Control Register.**

`led-display-ctrl` (— *vaddr*) returns the virtual address *vaddr* of the **LED Display Control Register.**

`fmb-0-data-discard` (— *vaddr*) returns the virtual address *vaddr* of the **FMB Channel 0 Data Discard Status Register.**

`fmb-1-data-discard` (— *vaddr*) returns the virtual address *vaddr* of the **FMB Channel 1 Data Discard Status Register.**

`lca-id` (— *vaddr*) returns the virtual address *vaddr* of the **LCA ID Register.**

4.2.3 Register Accesses

The commands described below are used to read data from and to store data in specific registers of the SPARC FGA-5000:

`vsi-sbus-base@` (— *long*) returns the contents — a 32-bit data — of the SBus Base Address Register.

`vsi-sbus-base!` (*long* —) stores the 32-bit data *long* in the SBus Base Address Register.

`vsi-id@` (— *id-code*) returns the contents — the 32-bit data *id-code* — of the Identification Register.

`vsi-vme-range@` (*range#* — *long*) returns the contents — a 32-bit data — of the SBus Address Decoding And Translation Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.

`vsi-vme-range!` (*long range#* —) stores the 32-bit value *long* in the SBus Address Decoding And Translation Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Address Decoding and Translation Registers.

`vsi-vme-master-cap@` (*range# — byte*) returns the contents — an 8-bit data — of the VMEbus Master Capability Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

`vsi-vme-master-cap!` (*byte range# —*) stores the 8-bit data *byte* in the VMEbus Master Capability Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 VMEbus Master Capability Registers.

`vsi-vme-cap@` (*— byte*) returns the contents — an 8-bit data — of the VMEbus Capability Register.

`vsi-vme-cap!` (*byte —*) stores the 8-bit data *byte* in the VMEbus Master Capability Register.

`vsi-sbus-ssel@` (*range# — byte*) returns the contents — an 8-bit data — of the SBus Slave Slot Select Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

`vsi-sbus-ssel!` (*byte range# —*) stores the 8-bit data *byte* in the SBus Slave Slot Select Register identified by its register number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 SBus Slave Slot Select Registers.

`vsi-sbus-cap@` (*— byte*) returns the contents — an 8-bit data — of the SPARC FGA-5000's SBus Capability Register.

`vsi-sbus-cap!` (*byte —*) stores the 8-bit data *byte* in the SPARC FGA-5000's SBus Capability Register.

`vsi-sbus-retry-time-ctrl@` (*— byte*) returns the contents — an 8-bit data — of the SPARC FGA-5000's SBus Retry Time Control Register.

`vsi-sbus-retry-time-ctrl!` (*byte —*) stores the 8-bit data *byte* in the SPARC FGA-5000's SBus Retry Time Control Register.

`vsi-sbus-rerun-limit-ctrl@` (*— word*) returns the contents — a 16-bit data — of the SPARC FGA-5000's SBus Rerun Limit Control Register.

`vsi-sbus-rerun-limit-ctrl!` (*word —*) store the 16-bit data *word* in the SPARC FGA-5000's SBus Rerun Limit Control Register.

`vsi-swpar@` (— *long*) returns the contents — a 32-bit data — of the SBus Write Posting Error Address Register.

`vsi-vwpar@` (— *long*) returns the contents — a 32-bit data — of the VMEbus Write Posting Error Address Register.

`vsi-slerr@` (— *long*) returns the contents — a 32-bit data — of the SBus Late Error Address Register.

`vsi-slerr-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the Late Error Interrupt Level Select and Enable Register.

`vsi-slerr-irq-map!` (*byte* —) stores the 8-bit data *byte* in the Late Error Interrupt Level Select and Enable Register.

`vsi-iack-emu@` (*level* — *byte*) returns the contents — an 8-bit data — of the IACK Cycle Emulation Register associated with the given *level*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vsi-vme-base@` (— *byte*) returns the contents — an 8-bit data — of the VMEbus Base Address Register.

`vsi-vme-base!` (*byte* —) stores the 8-bit data *byte* in the VMEbus Base Address Register.

`vsi-sbus-range@` (*range#* — *long*) returns the contents — a 32-bit data — of the VMEbus Address Decoding and Translation Register identified by its range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

`vsi-sbus-range!` (*long range#* —) stores the 32-bit data *long* in the VMEbus Address Decoding and Translation Register identified by its range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Decoding and Translation Registers.

`vsi-vme-ext@` (*range#* — *byte*) returns the contents — an 8-bit data — of the VMEbus Address Extension Register identified by its range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.

`vsi-vme-ext!` (*byte range#* —) stores the 8-bit data *byte* in the VMEbus Address Extension Register identified by its range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three VMEbus Address Extension Registers.

`vsi-reset-stat@` (— *byte*) returns the contents — an 8-bit data — of the Reset Source Register.

`vsi-intr-stat@` (— *long*) returns the contents — a 32-bit data — of the Interrupt Status Register.

`vsi-intr-stat!` (*long* —) stores the 32-bit data *long* in the Interrupt Status Register.

`.vsi-intr-stat` (—) displays the actual contents of the Interrupt Status Register. The contents of the register is displayed as shown below:

```
ok .vsi-intr-stat
VME-IRQ1: 0  VME-IRQ2: 0  VME-IRQ3: 0  VME-IRQ4: 0  VME-IRQ5: 0
VME-IRQ6: 0  VME-IRQ7: 0  VME-IACK: 0  FMB1    : 0  FMB0    : 0
IBOX    : 0  LERR    : 0  WDOG    : 0  DMATERM : 0  VWPERR  : 0
SWPERR  : 0  MAILBOX : 0  ARBTOUT : 0  ABORT   : 0  SYSFAIL+: 0
SYSFAIL-: 0  ACFAIL  : 0
ok
```

When an interrupt is pending the command displays the one (1); otherwise it displays the zero (0) to indicate that the interrupt is not pending.

Note! The state of the entry `SYSFAIL-` reports the occurrence of a *negative* edge of the VMEbus SYSFAIL* signal which indicates that the SYSFAIL* signal has been **asserted**. The state of the entry `SYSFAIL+` reports the occurrence of a *positive* edge of the VMEbus SYSFAIL* signal which indicates that the SYSFAIL* signal has been **negated**.

`vsi-irq-map@` (*level— byte*) returns the contents — an 8-bit data — of the VMEbus Interrupt Level Select and Enable Register associated with the given *level*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vsi-irq-map!` (*byte level* —) stores the 8-bit data *byte* in the VMEbus Interrupt Level Select and Enable Register associated with the given *level*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vsi-mbox-irq-map@` (*mailbox#* — *byte*) returns the contents — an 8-bit data — of the Mailbox Interrupt Level Select and Enable Register identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

`vsi-mbox-irq-map!` (*byte mailbox#* —) stores the 8-bit data *byte* in the Mailbox Interrupt Level Select and Enable Register identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Interrupt Level Select and Enable Registers.

`vsi-dma-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the DMA Interrupt Level Select and Enable Register.

`vsi-dma-irq-map!` (*byte* —) stores the 8-bit data in the DMA Interrupt Level Select and Enable Register.

`vsi-wpe-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the Write Posting Error Interrupt Level Select and Enable Register.

`vsi-wpe-irq-map!` (*byte* —) stores the 8-bit data *byte* in the Write Posting Error Interrupt Level Select and Enable Register.

`vsi-arb-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the Arbiter Timeout Interrupt Level Select and Enable Register.

`vsi-arb-irq-map!` (*byte* —) stores the 8-bit data *byte* in the Arbiter Timeout Interrupt Level Select and Enable Register.

`vsi-wdt-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the Watchdog Timer Interrupt Level Select and Enable Register.

`vsi-wdt-irq-map!` (*byte* —) stores the 8-bit data *byte* in the Watchdog Timer Interrupt Level Select and Enable Register.

`vsi-acfail-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the ACFAIL Interrupt Level Select and Enable Register.

`vsi-acfail-irq-map!` (*byte* —) stores the 8-bit data *byte* in the ACFAIL Interrupt Level Select and Enable Register.

`vsi-sysfail-irq-map0@` (— *byte*) returns the contents — an 8-bit data — of the SYSFAIL Assert Interrupt Level Select and Enable Register.

`vsi-sysfail-irq-map0!` (*byte* —) stores the 8-bit data *byte* in the SYSFAIL Assert Interrupt Level Select and Enable Register.

`vsi-sysfail-irq-map1@` (— *byte*) returns the contents — an 8-bit data — of the SYSFAIL Negate Interrupt Level Select and Enable Register.

`vsi-sysfail-irq-map1!` (*byte* —) stores the 8-bit data *byte* in the SYSFAIL Negate Interrupt Level Select and Enable Register.

`vsi-arb-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Arbiter Control Register.

`vsi-arb-ctrl!` (*byte* —) stores the 8-bit data *long* in the Arbiter Control Register.

`vsi-req-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Requester Control Register.

`vsi-req-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Requester Control Register.

`vsi-bus-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Bus Capture Control Register.

`vsi-bus-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Bus Capture Control Register.

`vsi-mbox@` (*mailbox#* — *byte*) returns the contents — an 8-bit data — of the Mailbox Register identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vsi-mbox!` (*byte mailbox#* —) stores the 8-bit data *byte* in the Mailbox Register identified by its mailbox number *mailbox#*. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vsi-mbox-stat@` (— *word*) returns the contents — a 16-bit data — of the Mailbox Status Register.

`vsi-mbox-stat!` (*word* —) stores the 16-bit data *long* in the Mailbox Status Register.

`vsi-sem@` (*semaphore#* — *byte*) returns the contents — an 8-bit data — of the Semaphore Register identified by its semaphore number *semaphore#*. The value of *semaphore#* may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

`vsi-sem!` (*byte semaphore#* —) stores the 8-bit data *byte* in the Semaphore Register identified by its semaphore number *semaphore#*. The value of *semaphore#* may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

`vsi-fmb-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Message Broadcast Control Register.

`vsi-fmb-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Message Broadcast Control Register.

`vsi-fmb-irq-map@` (*channel#* — *byte*) returns the contents — an 8-bit data — of the Message Broadcast Interrupt Level Select and Enable Register identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

`vsi-fmb-irq-map!` (*byte channel#* —) stores the 8-bit data *byte* in the Message Broadcast Interrupt Level Select and Enable Register identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Interrupt Level Select and Enable Registers.

`vsi-fmb-addr@` (— *byte*) returns the contents — an 8-bit data — of the Message Broadcast Address Register.

`vsi-fmb-addr!` (*byte* —) stores the 8-bit data *byte* in the Message Broadcast Address Register.

`vsi-fmb-stat@` (*channel#* — *byte*) returns the contents — an 8-bit data — of the Message Broadcast Status Register identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

`vsi-fmb-stat!` (*byte channel#* —) stores the 8-bit data *byte* in the Message Broadcast Status Register identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Status Registers.

`vsi-fmb-msg@` (*channel#* — *long true* | *false*) returns the contents — a 32-bit data — of the Message Broadcast Register identified by its channel number *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two Message Broadcast Registers.

`vsi-gcsr@` (— *byte*) returns the contents — an 8-bit data — of the Global Control and Status Register.

`vsi-gcsr!` (*byte* —) stores the 8-bit data *byte* in the Global Control and Status Register.

`vsi-mcsr0@` (— *byte*) returns the contents — an 8-bit data — of the Miscellaneous Control and Status Register 0.

`vsi-mcsr0!` (*byte* —) stores the 8-bit data *byte* in the Miscellaneous Control and Status Register 0.

`vsi-mcsr1@` (— *byte*) returns the contents — an 8-bit data — of the Miscellaneous Control and Status Register 1.

`vsi-mcsr1!` (*byte* —) stores the 8-bit data *byte* in the Miscellaneous Control and Status Register 1.

`vsi-wdt-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Watchdog Timer Control Register.

`vsi-wdt-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Watchdog Timer Control Register.

`vsi-wdt-restart@` (— *byte*) returns the contents — an 8-bit data — of the Watchdog Restart Register.

`vsi-wdt-restart!` (*byte* —) stores the 8-bit data *byte* in the Watchdog Restart Register.

`vsi-dma-ctrl@` (— *word*) returns the contents — a 16-bit data — of the DMA Control Register.

`vsi-dma-ctrl!` (*word* —) stores the 16-bit data *word* in the DMA Control Register.

`vsi-dma-mode@` (— *byte*) returns the contents — an 8-bit data — of the DMA Mode Register.

`vsi-dma-mode!` (*byte* —) stores the 8-bit data *byte* in the DMA Mode Register.

`vsi-dma-stat@` (— *byte*) returns the contents — an 8-bit data — of the DMA Status Register.

`vsi-dma-stat!` (*byte* —) stores the 8-bit data *byte* in the DMA Status Register.

`vsi-dma-src@` (— *long*) returns the contents — a 32-bit data — of the DMA Source Address Register.

`vsi-dma-src!` (*long* —) stores the 32-bit data *long* in the DMA Source Address Register.

`vsi-dma-dest@` (— *long*) returns the contents — a 32-bit data — of the DMA Destination Address Register.

`vsi-dma-dest!` (*long* —) stores the 32-bit data *long* in the DMA Destination Address Register.

`vsi-dma-cap@` (— *long*) returns the contents — a 32-bit data — of the DMA Capability and Transfer Count Register.

`vsi-dma-cap!` (*long* —) stores the 32-bit data *long* in the DMA Capability and Transfer Count Register.

`vsi-ibox-irq-map@` (— *byte*) returns the contents — an 8-bit data — of the Interrupt Box Interrupt Level Select and Enable Register.

`vsi-ibox-irq-map!` (*byte* —) stores the 32-bit data *long* in the Interrupt Box Interrupt Level Select and Enable Register.

`vsi-ibox-ctrl@` (— *word*) returns the contents — a 16-bit data — of the Interrupt Box Control Register.

`vsi-ibox-ctrl!` (*word* —) stores the 16-bit data *word* in the Interrupt Box Control Register.

`vsi-ibox-addr@` (— *word*) returns the contents — a 16-bit data — of the Interrupt Box Address Register.

`vsi-ibox-addr!` (*word* —) stores the 16-bit data *word* in the Interrupt Box Address Register.

The following commands are available to read data from and store data in the System Configuration Registers.

`led1-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the First User LED Control Register.

`led1-ctrl!` (*byte* —) stores the 8-bit data *byte* in the First User LED Control Register.

`led2-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Second User LED Control Register.

`led2-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Second User LED Control Register.

`flash-ctrl1@` (— *byte*) returns the contents — an 8-bit data — of the Flash Memory Control Register 1.

`flash-ctrl1!` (*byte* —) stores the 8-bit data *byte* in the Flash Memory Control Register 1.

`rotary-switch-stat@` (— *byte*) returns the contents — an 8-bit data — of the Rotary Switch Status Register.

`boot-rom-size-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Boot ROM Size Control Register.

`boot-rom-size-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Boot ROM Size Control Register.

`flash-ctrl2@` (— *byte*) returns the contents — an 8-bit data — of the Flash Memory Control Register 2.

`flash-ctrl2!` (*byte* —) stores the 8-bit data *byte* in the Flash Memory Control Register 2.

`flash-vpp-ctrl@` (— *byte*) returns the contents — an 8-bit data — of the Flash Memory Programming Voltage Control Register.

`flash-vpp-ctrl!` (*byte* —) stores the 8-bit data *byte* in the Flash Memory Programming Voltage Control Register.

`led-display-ctrl@` (*byte* —) returns the contents — an 8-bit data — of the LED Display Control/Status Register. Because the LED Display Control Register is only writable, the command returns the contents of the LED Display Control **Shadow** Register.

`led-display-ctrl!` (— *byte*) stores the 8-bit data *byte* in the LED Display Control/ Status Register. Because the LED Display Control Register is only writable, the command stores the given data in the LED Display Control **Shadow** Register, too.

`fmb-0-data-discard@` (— *byte*) returns the contents — an 8-bit data — of the FMB Channel 0 Data Discard Status Register.

`fmb-1-data-discard@` (— *byte*) returns the contents — an 8-bit data — of the FMB Channel 1 Data Discard Status Register.

`lca-id@` (— *byte*) returns the contents — an 8-bit data — of the LCA ID Register.

4.2.4 VMEbus Interrupt Handler

`vme-intr-pending?` (*level* — *true* | *false*) checks whether an interrupt is pending on a given interrupt request *level* and returns a *flag*. When an interrupt is pending the *flag* is *true*; otherwise it is *false*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.
The command verifies the state of the interrupt pending bit in the Interrupt Status register associated with the given *level*. When the corresponding status bit is set then no VMEbus interrupt is pending and the command returns *false*. Otherwise — the status bit is cleared — the value *true* is returned.

`vme-iack@` (*level* — *vector*) initiates an interrupt acknowledge cycle at the given VMEbus interrupt request *level* and returns the obtained 8-bit *vector*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Typically, the *vector* returned is within the range 0 through 255, but when no interrupt is pending, and therefore no interrupt has to be acknowledged, the value -1 is returned. Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vme-intr-ena` (*mapping level* —) enables the interrupt to be generated upon the receipt of a VMEbus interrupt at *level*. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of *mapping* may be one of the values in the range one through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table below lists all allowed mappings.

The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *mapping* and *level* are considered. When *level* is zero then the command treats it as if the value "one" has been passed to the command.

mapping	Constant	**Interrupt generated by SPARC FGA-5000**
0	`vsi-nmi`	INT (connected with *nonmaskable* interrupt)
1	`vsi-sbus-irq-1`	SINT1 (connected with SBus IRQ1)
2	`vsi-sbus-irq-2`	SINT2 (connected with SBus IRQ2)
3	`vsi-sbus-irq-3`	SINT3 (connected with SBus IRQ3)
4	`vsi-sbus-irq-4`	SINT4 (connected with SBus IRQ4)
5	`vsi-sbus-irq-5`	SINT5 (connected with SBus IRQ5)
6	`vsi-sbus-irq-6`	SINT6 (connected with SBus IRQ6)
7	`vsi-sbus-irq-7`	SINT7 (connected with SBus IRQ7)
The words listed in the second column of the table may be used to specify a valid *interrupt mapping*.		

Table 40: Interrupt Mapping.

`vme-intr-dis` (*level* —) disables the interrupt to be generated when the specified VMEbus interrupt request at *level* is asserted. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vme-irq-ena` (*level* —) enables the interrupt to be generated upon the receipt of a VMEbus interrupt at *level*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vme-irq-dis` (*level* —) disables the interrupt to be generated upon the receipt of a VMEbus interrupt at *level*. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vme-irq-map@` (*level* — *mapping*) returns the interrupt asserted by the SPARC FGA-5000 when the VMEbus interrupt request *level* is asserted. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`vme-irq-map!` (*mapping level* —) defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of *mapping* may be one of the values in the range one through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.
The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *mapping* and *level* are considered. When *level* is zero then the command treats it as if the value "one" has been passed to the command.

`install-vme-intr-handler` (*mapping level* —) installs the interrupt service routine dealing with the given VMEbus interrupt *level*. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.
The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels. The address of the interrupt service routine currently in effect is preserved.
Only the least significant three bits of *mapping* and *level* are considered. When *level* is zero then the command treats it as if the value "one" has been passed to the command.

`uninstall-vme-intr-handler` (*level* —) removes the interrupt service routine dealing with the given VMEbus interrupt *level* and installs the *old* interrupt service routine. The value of *level* may be one of the values in the range one through seven. Each value specifies one of the seven VMEbus interrupt request levels.
Only the least significant three bits of *level* are considered and when *level* is zero then the command treats it as if the value "one" has been passed to the command.

`.vme-vectors` (—) displays the VMEbus interrupt vectors received during the last interrupt acknowledge cycle.
OpenBoot maintains seven variables called `vme-intr{1|2|3|4|5|6|7}-vec-`

tor which are modified by the VMEbus interrupt handlers. In general, the interrupt handlers store the vector obtained during an interrupt acknowledge cycle in the appropriate variable.

4.2.5 VMEbus Arbiter

The commands listed below are available to control the arbiter:

`vme-slot1-ena` (—) enables the board to act as the system controller. In particular the command calls `vme-slot1!` — passing the value *true* to it — to enable the system controller function.

`vme-slot1-dis` (—) disables the board to act as the system controller. In particular the command calls `vme-slot1!` — passing the value *false* to it — to disable the system controller function.

`vme-slot1!` (*true* | *false* —) enables or disables the board's function to operate as the system controller. When the value *true* is passed to the command the board acts as the system controller. Otherwise — the value *false* is passed to the command — the system controller function is disabled.

`vme-arb-mode@` (— *mode*) returns the *mode* the arbiter is currently operating in. The value of *mode* may range from zero to three. Each value specifies a particular mode: the values zero and three indicate that the arbiter is operating in the *priority* mode; the value one specifies the *round-robin* mode; and the value two specifies the *prioritized-round-robin* mode.
Three constants are available to specify one of the three bus arbitration modes: **`pri`** — prioritized — (3_{10}), **`rrs`** — round robin select— (1_{10}), **`prr`** — prioritized round robin — (6_{10}).

`vme-arb-mode!` (*mode* —) selects the arbiter mode specified by *mode*. The value of *mode* may range from zero to three. Each value specifies a particular mode: the values zero and three indicate that the arbiter operates in the *priority* mode; the value one specifies the *round-robin* mode; and the value two specifies the *prioritized-round-robin* mode.

`vme-arb-irq-map!` (*mapping* —) selects the interrupt to be generated by the arbiter when the arbitration timeout expired. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`vme-arb-irq-ena` (—) enables the interrupt to be generated by the arbiter when the arbitration timeout expired. In particular the command calls `vme-arb-irq!` — passing the value *true* to it — to enable the interrupt.

`vme-arb-irq-dis` (—) disables the interrupt to be generated by the arbiter when the arbitration timeout expired. In particular the command calls `vme-arb-irq!` — passing the value *false* to it — to disable the interrupt.

`vme-arb-irq!` (*true* | *false* —) enables or disables the interrupt to be generated by the arbiter when the arbitration timeout expired. When the value *true* is passed to the command the interrupt is enabled. Otherwise — the value *false* is passed to the command — the interrupt is disabled.

`.vsi-arb-ctrl` (—) displays the current contents of the VMEbus Arbiter Control Register.

4.2.6 VMEbus Requester

The commands listed below are available to control the requester and to obtain some information about the requesters's operational state:

`vme-bus-request-level@` (— *level*) returns the VMEbus request *level* in use when the VMEbus interface tries to gain the ownership of the VMEbus. The value of *level* may be one of the values in the range one through three. Each value specifies one of the four VMEbus request levels.

`vme-bus-request-level!` (*level* —) selects the bus-request *level* to be used when the VMEbus is being accessed. The value of *level* may be one of the values in the range one through three. Each value specifies one of the four VMEbus request levels

`vme-bus-request-mode@` (— *mode*) returns the VMEbus request *mode* in use when the VMEbus interface tries to gain the ownership of the VMEbus.

`vme-bus-request-mode!` (*mode* —) selects the bus-request *mode* to be used when the VMEbus is being accessed.
Two constants are available to specify one of the two request modes: **`fair`** (0_{10}) and **`unfair`** (1_{10}).

`vme-bus-release-mode@` (— *mode*) returns the VMEbus release *mode* in use when the VMEbus interface has gained the ownership of the VMEbus.

`vme-bus-release-mode!` (*mode* —) selects the release *mode* to be used when the VMEbus interface has gained the ownership of the VMEbus
Four constants are available to specify one of the four release modes: **`ror`** — release on request — (3_{10}), **`roc`** — release on bus clear — (5_{10}), **`rat`** — release after timeout — (6_{10}), and **`rwd`** — release when done — (7_{10}).

Because the SPARC FGA-5000 allows to consider more than one bus release mode simultaneously — however, the combination of the release modes should be reasonable — the following two examples show how to use the available constants and command to specify the release mode:

```
ok ror rat and vme-bus-release-mode!
ok
```

In this example the VMEbus will be released either when another master requests the VMEbus, or after a *fixed* timeout expired. In the example below the VMEbus is released either when the BBSY* signal is negated, or after a *fixed* timeout expired.

```
ok roc rat and vme-bus-release-mode!
ok
```

`vme-early-release!` (*true* | *false* —) allows or prevents the requester from releasing the VMEbus early. When the value *true* is passed to the command the requester *releases* the VMEbus before the cycle has been terminated completely. Otherwise — the value *false* is passed to the command — the requester releases the VMEbus only when the cycle has been terminated completely.

`vme-bbsy-filter!` (*true* | *false* —) enables or disables the BBSY* glitch filter. When the value *true* is passed to the command the BBSY* glitch filter is enabled. Otherwise — the value *false* is passed to the command — the BBSY* glitch filter is disabled.

`.vsi-req-ctrl` (—) displays the current contents of the VMEbus Requester Control Register.

`vme-bus-capture!` (*true* | *false* —) enables or disables the *bus-capture-and-hold* capability of the SPARC FGA-5000. If the value *true* is passed to the command the VMEbus Interface starts to capture the bus and when it gains the ownership of the bus it holds the as long as the bus is released. The bus is released when the command is called and the value *false* is passed to it.

`vme-bus-captured?` (— *true* | *false*) determines whether the VMEbus interface gains the ownership of the bus. The value *true* is returned when the VMEbus interface gains the ownership of the VMEbus. Otherwise the value *false* is returned to indicated that the VMEbus interface has not gained the ownership of the bus.
In general this command is called immediately after a *capture-and-hold* cycle has been initiated as shown in the example below:

```
ok true vme-bus-capture!
ok begin vme-bus-captured? until
ok ...

ok false vme-bus-capture!
ok
```

4.2.7 VMEbus Status Signals

The commands listed below are available to access and control the VMEbus status signals.

`vme-sysfail-set` (—) asserts (sets) the VMEbus SYSFAIL* signal.

`vme-sysfail-clear` (—) negates (clears) the VMEbus SYSFAIL* signal.

`vme-sysfail?` (— *true* | *false*) determines the state of the VMEbus SYSFAIL* signal and returns a *flag* set according to the signal's state. When the SYSFAIL* signal is asserted the *flag* returned is *true*; otherwise its value is *false*.

`vme-sysfail-assert-irq-map!` (*mapping* —) selects the interrupt to be generated when the VMEbus SYSFAIL* signal is asserted. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the SYSFAIL* signal is asserted. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`vme-sysfail-assert-irq-ena` (—) allows the VMEbus interface to generate an interrupt upon the assertion of the VMEbus SYSFAIL* signal.

`vme-sysfail-assert-irq-dis` (—) disables the interrupt to be generated upon the assertion of the VMEbus SYSFAIL* signal.

`vme-sysfail-assert-ip?` (— *true* | *false*) checks whether an interrupt is pending due to the assertion of the VMEbus SYSFAIL* signal and returns a *flag* set according to the appropriate interrupt pending flag. The *flag* is *true* when the interrupt is pending; otherwise its value is *false*.

`vme-sysfail-assert-ip-clear` (—) clears a pending interrupt generated by the assertion of the VMEbus SYSFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.

`vme-sysfail-negate-irq-map!` (*mapping* —) selects the interrupt to be generated when the VMEbus SYSFAIL* signal is negated. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the SYSFAIL* signal is negated. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`vme-sysfail-negate-irq-ena` (—) allows the VMEbus interface to generate an interrupt upon the negation of the VMEbus SYSFAIL* signal.

`vme-sysfail-negate-irq-dis` (—) disables the interrupt to be generated upon the negation of the VMEbus SYSFAIL* signal.

`vme-sysfail-negate-ip?` (— *true* | *false*) checks whether an interrupt is pending due to the negation of the VMEbus SYSFAIL* signal and returns a *flag* set according to the appropriate interrupt pending flag. The *flag* is *true* when the interrupt is pending; otherwise its value is *false*.

`vme-sysfail-negate-ip-clear` (—) clears a pending interrupt generated by the negation of the VMEbus SYSFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.

`vme-acfail?` (— *true* | *false*) determines the state of the VMEbus ACFAIL* signal and returns a *flag* set according to the signal's state. When the ACFAIL* signal is asserted the *flag* returned is *true*; otherwise it is *false*.

`vme-acfail-assert-irq-map!` (*mapping* —) selects the interrupt to be generated when the VMEbus ACFAIL* signal is asserted. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the ACFAIL* signal is asserted. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings

`vme-acfail-assert-irq-ena` (—) allows the VMEbus interface to generate an interrupt upon the assertion of the VMEbus ACFAIL* signal.

`vme-acfail-assert-irq-dis` (—) disables the interrupt to be generated upon the assertion of the VMEbus ACFAIL* signal.

`vme-acfail-assert-ip?` (— *true* | *false*) checks whether an interrupt is pending due to the assertion of the VMEbus ACFAIL* signal and returns a *flag* set according to the appropriate interrupt pending flag. The *flag* is *true* when the interrupt is pending; otherwise its value is *false*.

`vme-acfail-assert-ip-clear` (—) clears a pending interrupt generated by the assertion of the VMEbus ACFAIL* signal. This command clears on the corresponding interrupt pending bit in the Interrupt Status Register of the SPARC FGA-5000.

4.2.8 VMEbus Master Interface

The SPARC FGA-5000 provides 16 sets of registers to control any VMEbus master operation. Each set may be used to address a certain address range within the VMEbus' address space. A register set is identified by an unique number, the *range number* (*range#*), in the range zero through 15.
When the VMEbus is being accessed the part of the SPARC FGA-5000 connected with the SBus is considered as the **SBus slave** device, whereas the part of the SPARC FGA-5000 that is connected with the VMEbus is operating as **VMEbus master**. This fact is reflected in the names of the commands available to control the VMEbus master interface.
The commands listed and described in the following are available to initialize and control the VMEbus master interface:

`#vme-ranges` (— *#vme-ranges*) returns the number *#vme-ranges* of available register sets which are used to control accesses to the VMEbus.

`vme-master-ena` (*range#* —) enables the address decoding associated with the range number *range#* to access the VMEbus. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`vme-master-dis` (*range#* —) enables the address decoding associated with the range number *range#* to access the VMEbus. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`vme-master-wp-ena` (*range#* —) enables *write posting* within the VMEbus address range associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`vme-slave-wp-dis` (*range#* —) disables *write posting* within the VMEbus address range associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`vme-supervisor!` (*true* | *false* —) selects the mode in which the VMEbus is being accessed. When the value *true* is passed to the command, the VMEbus is accessed in the *previleged* mode. Otherwise — the value *false* is passed to the command — the VMEbus is accessed in the *non-previleged* mode.
The mode selected with this command applies to **all** ranges used to access the VMEbus.

`vme-master-cap@` (*range# — data-capability address-capability*) returns the address- and data capabilities associated with the range number *range#* which are used when

the VMEbus is accessed. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VME-bus master operation.
The value of *data-capability* and *address-capability* may be one of the values listed in the table below.

`vme-master-cap!` (*data-capability address-capability range#* —) defines the address- and data capabilities associated with the range number *range#* which are used when the VMEbus is accessed. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VME-bus master operation.
The value of *data-capability* and *address-capability* may be one of the values listed in the table below:

value	data-capability	address-capability
000_2	cap-d8	cap-a16
001_2	cap-d16	cap-a24
010_2	cap-d32	cap-a32
011_2	cap-blt	reserved
100_2	cap-mblt	cap-a64
101_2	reserved	reserved
110_2	reserved	reserved
111_2	reserved	reserved

`sbus-slot-sel@` (*range#* — *sbus-slot#*) returns the number of the SBus slot *sbus-slot#* that is associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`sbus-slot-sel!` (*sbus-slot# range#* —) sets the number of the SBus slot *sbus-slot#* that is associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`sbus-slot#>ssel#` (*sbus-slot#* — *ssel#*) converts the logical SBus slot number *sbus-slot#* to the corresponding SBus slave select number *ssel#*.

`ssel#>sbus-slot#` (*ssel#* — *sbus-slot#*) converts the SBus slave select number *ssel#* to the corresponding logical SBus slot number *sbus-slot#*.

`/sbus-range` (*range# — size*) returns the *size* of the range associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`sbus-slave-range@` (*range# — offset sbus-slot# size*) returns the SBus slave parameters associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.
The parameters returned by the command specify the SBus address range to be accessed to reach the VMEbus. The address range is represented by the triple *offset*, *sbus-slot#*, and *size*.

`sbus-slave-range!` (*offset sbus-slot# size range# —*) sets the SBus slave parameters associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.
The parameters passed to the command specify the SBus address range to be accessed to reach the VMEbus. The address range is represented by the triple *offset*, *sbus-slot#*, and *size*.

`vme-master-range@` (*range# — addr data-capability address-capability size*) returns the VMEbus master capabilities associated with the range number identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.
The VMEbus address range being accessed is represented by the *addr-size* pair, where *addr* specifies the physical VMEbus address and *size* identifies the address range covered. The value of *data-capability* and *address-capability* may be one of the values listed in the table below.

`vme-master-range!` (*addr data-capability address-capability size range# —*) sets the VMEbus master capabilities associated with the range number identified by *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.
The VMEbus address range being accessed is represented by the *addr-size* pair, where *addr* specifies the physical VMEbus address and *size* identifies the address range covered. The value of *data-capability* and *address-capability* may be one of the values listed in the table below.

value	data-capability	address-capability
000_2	cap-d8	cap-a16
001_2	cap-d16	cap-a24

value	data-capability	address-capability
010_2	cap-d32	cap-a32
011_2	cap-blt	reserved
100_2	cap-mblt	cap-a64
101_2	reserved	reserved
110_2	reserved	reserved
111_2	reserved	reserved

`.vme-master-range` (*range#* —) displays the current settings of the VMEbus master interface asociated with the range number *range#*. The value of *range#* may be one of the values in the range zero through 15. Each value specifies one of the 16 register sets controlling any VMEbus master operation.

`.vme-master-ranges` (—) displays the current settings of **all** register sets of the VME-bus master interface.

`.vme-cap` (—) displays the contents of the VMEbus Capability register.

`vme-master-map` (*range#* — *vaddr*) makes the physical address range, as defined by the contents of the range register set specified by the range number *range#*, available to the processor's virtual address space and returns the virtual address *vaddr*.
Because the command obtains all information from the specific range register set, the particular range register **must** be initialized before.

`vme-master-unmap` (*vaddr range#* —) removes the physical address range, as defined by the contents of the range register set specified by the range number *range#*, from the processor's virtual address space.

`addr,size>sbus-compare-code` (*address size* — *compare-code*) returns the SBus *compare-code* which corresponds with the given *address* and *size* pair.

`sbus-compare-code>addr,size` (*compare-code* — *address size*) converts the SBus *compare-code* to the corresponding *address* and *size* pair.

The examples on the following pages describe how to initialize the VMEbus interface for subsequent VMEbus master accesses.

The example below shows how to access a 1 MByte area within the *extended* address space (A32) of the VMEbus beginnig at address 4080.0000_{16} . The register set associated with the range number zero (*range#* is 0) is used to access the VMEbus area mentioned above.
The first commands initializes the VMEbus master interface. It sets the data- and address

capabilities, as well as the VMEbus address and the size of the area being accessed. The *data capability* is defined using the predefined constant `cap-d32` which enables the VMEbus master interface to access bytes (8bit data), half-words (16bit data), and words (32-bit data) within the VMEbus area. The *address capability* is defined using the predefined constant `cap-a32` that enables the VMEbus interface to access the *extended* address space (A32) of the VMEbus.
The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets $A0.0000_{16}$ to $BF.FFFF_{16}$ which corresponds to the VMEbus addresses in the range 4080.0000_{16} to $409F.FFFF_{16}$ of the *extended* address space (A32).

```
ok h# 4080.0000 cap-d32 cap-a32 1Meg 2 * 0 vme-master-range!
ok 1Meg d# 10 * 1 1Meg 2 * 0 sbus-slave-range!
ok 0 vme-master-ena
ok 0 vme-master-map value vmebus
ok
```

Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.

```
ok vmebus 0 vme-master-unmap
ok
```

When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor's virtual address space to access the VMEbus must be released **before** the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

In the next example the VMEbus interface is initialized to allow accesses to the *standard* address space (A24) of the VMEbus beginnig at address 98.0000_{16} . The size of this area is 512KByte and the register set associated with the range number one (*range#* is 1) is used to access this VMEbus area.
The first commands initializes the VMEbus master interface. It sets the data- and address capabilities, as well as the VMEbus address and the size of the area being accessed. The *data capability* is defined using the predefined constant `cap-d16` which enables the VMEbus master interface to access bytes (8bit data), and half-words (16bit data) within the VMEbus area. The *address capability* is defined using the predefined constant `cap-a24` that enables the VMEbus interface to access the *standard* address space (A24) of the VMEbus.
The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets 120.0000_{16} to $127.FFFF_{16}$ which corresponds to the VMEbus addresses in the range 98.0000_{16} to $9F.FFFF_{16}$ of the *standard* address space (A24).

```
ok h# 98.0000 cap-d16 cap-a24 1Meg 2 / 1 vme-master-range!
ok 1Meg d# 18 * 2 1Meg 2 / 1 sbus-slave-range!
ok 1 vme-master-ena
ok 1 vme-master-map value vmebus
ok
```

Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.

```
ok 1 vme-master-map value vmebus
ok
```

When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor's virtual address space to access the VMEbus must be released **before** the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

The last example describes how to initialized the VMEbus interface to allow accesses to the *short* address space (A16) of the VMEbus beginnig at address 0000_{16} . The size of this area is 64KByte and therefore covers the entired *short* address space. The register set associated with the range number two (*range#* is 2) is used to access this VMEbus area.
Again, the first commands initializes the VMEbus master interface. It sets the data- and address capabilities, as well as the VMEbus address and the size of the area being accessed. The *data capability* is defined using the predefined constant `cap-d8` which limits the VMEbus master interface to access only bytes (8bit data) within the VMEbus area. The *address capability* is defined using the predefined constant `cap-a16` that enables the VMEbus interface to access the *standard* address space (A16) of the VMEbus.
The SBus slave interface is initialized by the second command which specifies that the VMEbus is accessed when the SBus slot one (1) is being accessed at offsets 400.0000_{16} to $400.FFFF_{16}$ which corresponds to the VMEbus addresses in the range 0000_{16} to $FFFF_{16}$ of the *short* address space (A16).

```
ok h# 0000 cap-d8 cap-a16 h# 1.0000 2 vme-master-range!
ok 1Meg d# 64 * 3 h# 1.0000 2 sbus-slave-range!
ok 2 vme-master-ena
ok 2 vme-master-map value vmebus
ok
```

Finally, the third command enables any access to the VMEbus. The fourth command maps the physical address area to be accessed in order to address the VMEbus to the virtual address space of the processor and stores the virtual address in the variable `vmebus`. This variable may be used to access the VMEbus area using the commands to read and write data provided by OpenBoot.

```
ok 2 vme-master-map value vmebus
ok
```

When the translation (SBus to VMEbus) defined by the contents of the register set associated with the range number zero is no longer used, then the memory mapped to the processor's virtual address space to access the VMEbus must be released **before** the contents of this register set are modified. This has to be done with the command `vme-master-unmap` as stated above.

Assumed the first three register sets have been used to access the VMEbus address spaces as described in the examples above, then the following command may be used to display the settings of the registers sets:

```
ok .vme-master-ranges
```

The following commands are available to control the various operating modes of the SPARC FGA-5000 SBus interface.

`sbus-burst-length@` (— *#burst-lenght*) returns the maximum length of an SBus burst that is generated by the SPARC FGA-5000. The value of *#burst-length* is in the range zero through three. Each value specifies one of four possible burst lengths as stated in the table below.

`sbus-burst-length!` (*#burst-lenght* —) sets the maximum length of an SBus burst that is generated by the SPARC FGA-5000. The value of *#burst-length* may be in the range zero through three. Each value specifies one of four possible burst lengths as stated in the table below. The command considers only the least significant two bits of the value *#burst-length.*

#burst-length	Burst Length
0	8-byte burst
1	16-byte burst
2	32-byte burst
3	64-byte burst

`sbus-master-read-stop-point@` (— *#read-stop-point*) returns the SBus master read stop point currently in effect. The value of *#read-stop-point* is in the range zero through three. Each value specifies one of four possible read stop points as stated in the table below.

`sbus-master-read-stop-point!` (*#read-stop-point* —) sets the SBus master read stop point used by the SPARC FGA-5000. The value of *#read-stop-point* may be in the range zero through three. Each value specifies one of four possible read stop points as stated in the table below. The command considers only the least significant two bits of the value *#read-stop-point.*

#read-stop-point	Read Stop Point
0	Stop at 8-byte boundary
1	Stop at 16-byte boundary
2	Stop at 32-byte boundary
3	Stop at 64-byte boundary

`sbus-retry-time@` (— *#retry-time*) returns the number of SBus clocks before an SBus cycle is terminated with a retry by the SPARC FGA-5000. The value of *#retry-time* is in the range zero through 255 and specifies the number of SBus clocks.

`sbus-retry-time!` (*#retry-time* —) sets the number of SBus clocks before an SBus cycle is terminated with a retry by the SPARC FGA-5000. The value of *#retry-time* may be in the range zero through 255 and specifies the number of SBus clocks. The command treats the value of *#retry-time* as a modulo 256 number.
When the command is called it verifies whether the given number of SBus clocks falls below the limit specified by `min-retry-time`. If this value falls below the given limit, then the commands uses the value of `min-retry-time` instead. This ensures that the SBus interface is operating properly.

`sbus-rerun!` (*true* | *false*) enables or disables the SPARC FGA-5000's capability to generate reruns on the SBus. When the value *true* is passed to the command the SPARC FGA-5000 will initiate SBus rerun if necessary. Otherwise — the value *false* is passed to the command — the SPARC FGA-5000's capability to initiate SBus reruns is disabled.

`sbus-rerun-ena` (—) enables the SPARC FGA-5000's capability to generate reruns on the SBus.

`sbus-rerun-dis` (—) disables the SPARC FGA-5000's capability to generate reruns on the SBus.

`sbus-rerun-limit@` (— *#rerun-limit*) returns the number of reruns before the SPARC FGA-5000 terminates an SBus cycle with an error. The value of *#rerun-limit* is in the range zero through 255 and specifies the number of reruns.

`sbus-rerun-limit!` (*#rerun-limit* —) sets the number of reruns before the SPARC FGA-5000 terminates an SBus cycle with an error. The value of *#rerun-limit* may be in the range zero through 255 and specifies the number of reruns. The command treats the value of *#rerun-limit* as a modulo 256 number.
When the command is called it verifies whether the given number of reruns falls below the limit specified by `min-rerun-limit`. If this value falls below the given limit, then the commands use the value of `min-rerun-limit` instead. This ensures that the SBus interface is operating properly.

`sbus-burst-ena` (—) enables the SPARC FGA-5000's capability to transfer data using SBus burst transfers.

`sbus-burst-dis` (—) disables the SPARC FGA-5000's capability to transfer data using SBus burst transfers.

`sbus-hidden-arb-ena` (—) enables the SPARC FGA-5000's capability to perform *hidden* arbitration.

`sbus-hidden-arb-dis` (—) disables the SPARC FGA-5000's capability to perform *hidden* arbitration.

`sbus-split-flow-ena` (—) enables split flowthrough.

`sbus-split-flow-dis` (—) disables split flowthrough.

`sbus-split-ena` (—) enables the SPARC FGA-5000's capability to split SBus cycles.

`sbus-split-dis` (—) disables the SPARC FGA-5000's capability to split SBus cycles.

`.sbus-cap` (—) displays the current contents of the SBus Master Capability Register as shown below:

```
ok .sbus-cap
Split: 1   Split Flow: 1   Arbiter: 1   Burst: 1
Master Read Stop Point: 32 bytes  Max. Burst Length: 32 bytes
ok
```

`.sbus-retry-time-ctrl` (—) displays the current contents of the SPARC FGA-5000's SBus Retry Time Control Register as stated below:

```
ok .sbus-retry-time-ctrl
Retry time: 10
ok
```

`.sbus-rerun-limit-ctrl` (—) displays the current contents of the SBus Rerun Limit Control Register as depicted below:

```
ok .sbus-rerun-limit-ctrl
Enable Reruns: 0   Rerun limit: 255
ok
```

4.2.9 VMEbus Slave Interface

The SPARC FGA-5000 provides three sets of registers to control any VMEbus slave access. Each set may be used to make a certain slave address range — *standard-* (A24) or *extended* (A32) slave address range — available to the VMEbus' address space. A register set is identified by a unique number, the *range number* (*range#*), in the range zero through two. Only the A24 and A32 slave mode allows a VMEbus master to access the memory of the SPARC CPU-5V. In the A16 slave mode all VMEbus master accesses are limited to the registers of the SPARC FGA-5000 which are accessible from the VMEbus.

When the VMEbus interface is being accessed from the VMEbus, then the part of the SPARC FGA-5000 connected with the VMEbus is considered as **VMEbus slave** device. Whereas the part of the SPARC FGA-5000 that is connected with the SBus is operating as the **SBus master**. This fact is reflected in the names of the commands available to control the VMEbus master interface.

The commands listed and described in the following are available to initialize and control the A16 VMEbus slave interface:

`vme-a16-slave-ena` (—) enables the capability to access the SPARC FGA-5000 registers from the VMEbus in the *short* address space (A16).

`vme-a16-slave-dis` (—) disables the capability to access the SPARC FGA-5000 registers from the VMEbus in the *short* address space (A16).

`vme-a16-slave-addr@` (— *addr*) returns the 16-bit address *addr* at which the registers of the SPARC FGA-5000 are accessible within the *short* address space (A16).

`vme-a16-slave-addr!` (*addr* —) defines the 16-bit address *addr* at which the registers of the SPARC FGA-5000 are accessible within the *short* address space (A16).
The least significant nine bits of the address *addr* are ignored by the command — the command treats them as if they are cleared —, because the SPARC FGA-5000 is only accessible from the VMEbus beginning at 512 Byte boundaries.

The commands listed and described in the following are available to initialize and control the A24 and A32 VMEbus slave interface:

`vme-slave-ena` (*range#* —) enables the address decoding associated with the range number *range#* to allow accesses from the VMEbus. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`vme-slave-dis` (*range#* —) disables the address decoding associated with the range number *range#* to allow accesses from the VMEbus. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`vme-slave-wp-ena` (*range#* —) enables *write posting* within the VMEbus slave address range associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`vme-slave-wp-dis` (*range#* —) disables *write posting* within the VMEbus slave address range associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`/vme-a24-range` (*range#* — *size*) returns the *size* of the *standard* (A24) slave interface associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`sbus-a24-master-range@` (*range#* — *vaddr size*) returns the SBus master parameters associated with the A24 slave interface identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
The parameters *vaddr* and *size* identify the virtual address range within the SBus, into which all A24 slave accesses are translated.

`sbus-a24-master-range!` (*vaddr size range#* —) defines the SBus master parameters associated with the A24 slave interface identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
The parameters *vaddr* and *size* identify the virtual address range within the SBus, into which all A24 slave accesses are translated.

`vme-a24-slave-range@` (*range#* — *paddr size*) returns the VMEbus base address *paddr* and the size *size* of the A24 slave window associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`vme-a24-slave-range!` (*paddr size range#* —) sets the VMEbus base address *paddr* and the size *size* of the A24 slave window associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`/vme-a32-range` (*range# — size*) returns the *size* of the *extended* (A32) slave interface associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`sbus-a32-master-range@` (*range# — vaddr size*) returns the SBus master parameters associated with the A32 slave interface identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
The parameters *vaddr* and *size* identify the virtual address range within the SBus, into which all A32 slave accesses are translated.

`sbus-a32-master-range!` (*vaddr size range# —*) defines the SBus master parameters associated with the A32 slave interface identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.
The parameters *vaddr* and *size* identify the virtual address range within the SBus, into which all A32 slave accesses are translated.

`vme-a32-slave-range@` (*range# — paddr size*) returns the VMEbus base address *paddr* and the size *size* of the A32 slave window associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`vme-a32-slave-range!` (*paddr size range# —*) sets the VMEbus base address *paddr* and the size *size* of the A32 slave window associated with the range identified by *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`.vme-slave-range` (*range# —*) displays the current settings of the VMEbus slave interface associated with the range number *range#*. The value of *range#* may be one of the values in the range zero through two. Each value specifies one of the three register sets controlling any VMEbus slave access.

`.vme-slave-ranges` (—) displays the current settings of **all** register sets controlling any VMEbus slave access.

`addr, size>vme-compare-code` (*address size — compare-code*) returns the VMEbus *compare-code* which corresponds with the given *address* and *size* pair.

`vme-compare-code>addr, size` (*compare-code — address size*) converts the VMEbus *compare-code* to the corresponding *address* and *size* pair.

The following example lists all steps to be taken, to initialize the VMEbus interface for A32 accesses from the VMEbus beginning at address 2340.0000_{16} and ranging to $235F.FFFF_{16}$. The register set associated with the *range number* zero (0) is used to control this particular VMEbus slave interface.

```
ok h# 2340.0000 1Meg 2 * 0 vme-a32-slave-range!
ok h# ffe0.0000 1Meg 2 * 0 sbus-a32-master-range!
ok h# 10.0000 obmem h# ffe0.0000 1Meg 2 * iomap-pages
ok 0 vme-slave-ena
ok
```

As shown above the first command defines the VMEbus slave interface's base address and size of the slave window. The second command defines that any A32 access is translated to an access of the SBus beginning at SBus address $FFE0.0000_{16}$. And the third command creates all necessary entries within the IOMMU to translate the SBus access to an access of the on-board memory beginning at physical address 10.0000_{16}.
Finally, the VMEbus slave interface is enabled using the fourth command.

The next example lists all steps to be taken, to initialize the VMEbus interface for A24 accesses from the VMEbus beginning at address $C0.0000_{16}$ and ranging to $CF.FFFF_{16}$. The register set associated with the *range number* one (1) is used to control this particular VMEbus slave interface.

```
ok h# c0.0000 1Meg 1 vme-a24-slave-range!
ok h# fff0.0000 1Meg 1 sbus-a24-master-range!
ok h# 20.0000 obmem h# fff0.0000 1Meg iomap-pages
ok 1 vme-slave-ena
ok
```

As shown above the first command defines the VMEbus slave interface's base address and size of the slave window. The second command defines that any A24 access is translated to an access of the SBus beginning at SBus address $FFF0.0000_{16}$. And the third command creates all necessary entries within the IOMMU to translate the SBus access to an access of the on-board memory beginning at physical address 20.0000_{16}.
Finally, the VMEbus slave interface is enabled using the fourth command.

4.2.10 VMEbus Device Node

The OpenBoot device tree contains the device node for the VMEbus interface and is called "**vme**". It is a *child device* of the device node "**/iommu**" (The full pathname of the VMEbus interface device node is displayed by the command **show-devs**). The device *alias* **vme** is available as a shorthand representation of the VMEbus interface device-path.

The vocabulary of the VMEbus device includes the standard commands recommended for a *hierarchical* device. The words of this vocabulary are only available when the VMEbus device has been selected as shown below:

```
ok cd vme
ok words
selftest     reset     close     open ...
... list of further methods of the device node
ok selftest .
0
ok device-end
ok
```

The example listed above, selects the VMEbus device and makes it the current node. The word **words** displays the names of the *methods* of the VMEbus device. And the third command calls the method **selftest** and the value returned by this method is displayed. The last command *unselects* the current device node, leaving no node selected.
The following methods are defined in the vocabulary of the VMEbus device:

`open` (— *true*) prepares the package for subsequent use. The value *true* is always returned.

`close` (—) frees all resources allocated by open.

`reset` (—) puts the VMEbus Interface into *quiet* state.

`selftest` (— *error-number*) performs a test of the VMEbus interface, and returns an *error-number* to report the course of the test. In the case that the device has been tested successfully the value zero is returned; otherwise it returns a specific error number to indicate a certain fail state.

`decode-unit` (*addr len* — *low high*) converts the *addr* and *len*, a text string representation, to *low* and *high* which is a numerical representation of a physical address within the address space defined by the package.

`map-in` (*low high size* — *vaddr*) creates a mapping associating the range of physical address beginning at *low*, extending for *size* bytes, within the package's physical address space, with a processor virtual address *vaddr*.

`map-out` (*vaddr size* —) destroys the mapping set by map-in at the given virtual address *vaddr* of length *size*.

`dma-alloc` (*size* — *vaddr*) allocates a virtual address range of length *size* bytes that is suitable for direct memory access by a bus master device. The memory is allocated according to the most stringent alignment requirements for the bus. The address of the acquired virtual memory *vaddr* is returned via the stack.

`dma-free` (*vaddr size* —) releases a given virtual memory, identified by its address *vaddr* and *size*, previously acquired by `dma-alloc`.

`dma-map-in` (*vaddr size cachable?* — *devaddr*) converts a given virtual address range, specified by *vaddr* and *size*, into an address *devaddr* suitable for direct memory access on the bus. The virtual memory must be allocated already by `dma-alloc`. The SPARC CPU-5V does not support caching. Thus the *cachable?* flag is ignored.

`dma-map-out` (*vaddr devaddr size* —) removes the direct memory access mapping previously created by `dma-map-in`.

`dma-sync` (*vaddr devaddr size* —) synchronizes memory caches associated with a given direct memory access mapping, specified by its virtual address *vaddr*, the *devaddr* and its *size* that has been established by dma-map-in.

4.2.11 VMEbus NVRAM Configuration Parameters

The NVRAM configuration parameters listed below are available to control the initialisation and operation of the VMEbus Interface. The current state of these configuration parameters are displayed using the `printenv` command, and are modified using either the `setenv`, or the `set-default` command provided by OpenBoot.

`vme-sysfail-clear?` when the value of the configuration parameter is `true` the SYSFAIL* signal will be cleared by OpenBoot. In the case that the configuration parameter is `false` OpenBoot will not clear the SYSFAIL* signal, but the operating system which is loaded has to clear it. (default: `true`)
The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-bus-timer?` controls whether the VMEbus transaction timer in the SPARC FGA-5000 is used to watch each VMEbus access. When the flag is `true` the transaction timer is enabled. If the flag is `false` the transaction timer is disabled. (default: `true`)
The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-bus-timeout` contains the timeout value of the SPARC FGA-5000 VMEbus transaction timer and is a value in the range one to three. Each value selects a particular timeout period. Independent of the state of the configuration parameter `vme-bus-timer?` the timeout value is stored in the appropriate register. When the value of this configuration parameter is not in the range one through three, then the value three is used instead. (default: 3_{10})
The state of this NVRAM configuration parameter is considered independent of the state of the `vme-init?` configuration parameter.

`vme-slot#` specifies the *logical* VMEbus slot number assigned to the SPARC CPU-5V board. The value may be in the range one through 255, but preferably should be set in such a way that it corresponds with the number of an available VMEbus slot.
The state of this configuration parameter does **not** control whether the VMEbus interface is operating as system controller when the configuration parameter's value is one. (default: 1_{10})

`vme-fair-req?` specifies whether the VMEbus requester operates in the *fair* mode when requesting the VMEbus. When the value of the configuration parameter is `true`, the VMEbus requester operates in the *fair* mode. Otherwise — the value of the configuration parameter is `false` — the requester does not operate **not** in the fair mode. (default: `true`)

`vme-init?` controls whether the VMEbus interface is initialized by OpenBoot. When this flag is `true` the VMEbus interface is initialized according to the state of the NVRAM parameter listed below. In the case that the flag is `false` the VMEbus interface is not initialized. The VMEbus interface is initialized **after** OpenBoot set up the main memory. (default: `true`)

The state of the NVRAM configuration parameters listed in the following are only considered by OpenBoot when the configuration parameter `vme-init?` is `true`!

`vme-intr1` controls whether the VMEbus interrupt request level 1 has to be enabled. When the value is 255 then the VMEbus interrupt request level 1 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 1 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 1 occurs. (default: 255_{10})

`vme-intr2` controls whether the VMEbus interrupt request level 2 has to be enabled. When the value is 255 then the VMEbus interrupt request level 2 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 2 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 2 occurs. (default: 255_{10})

`vme-intr3` controls whether the VMEbus interrupt request level 3 has to be enabled. When the value is 255 then the VMEbus interrupt request level 3 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 3 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 3 occurs. (default: 255_{10})

`vme-intr4` controls whether the VMEbus interrupt request level 4 has to be enabled. When the value is 255 then the VMEbus interrupt request level 4 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 4 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 4 occurs. (default: 255_{10})

`vme-intr5` controls whether the VMEbus interrupt request level 5 has to be enabled. When the value is 255 then the VMEbus interrupt request level 5 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 5 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 5 occurs. (default: 255_{10})

`vme-intr6` controls whether the VMEbus interrupt request level 6 has to be enabled. When the value is 255 then the VMEbus interrupt request level 6 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 6 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 6 occurs. (default: $\mathbf{255}_{10}$)

`vme-intr7` controls whether the VMEbus interrupt request level 7 has to be enabled. When the value is 255 then the VMEbus interrupt request level 7 is not enabled. In the case that the value is within the range one to seven, the corresponding interrupt handler is activated and the VMEbus interrupt request level 7 is enabled. The values one to seven specify the SPARC FGA-5000 interrupt request line to be asserted when a VMEbus interrupt request level 7 occurs. (default: 255_{10})

`vme-sysfail-assert?` controls whether a nonmaskable interrupt is generated upon the assertion of the VMEbus signal SYSFAIL*. When the flag is `true` an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the assertion of the SYSFAIL* signal is enabled. In the case that the flag is `false` the ability to generate a nonmaskable interrupt upon the assertion of the SYSFAIL* signal is enabled. (default: `false`)

`vme-sysfail-negate?` controls whether a nonmaskable interrupt is generated upon the negation of the VMEbus signal SYSFAIL*. When the flag is `true` an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the negation of the SYSFAIL* signal is enabled. In the case that the flag is `false` the ability to generate a nonmaskable interrupt upon the negation of the SYSFAIL* signal is enabled. (default: `false`)

`vme-acfail-assert?` controls whether a nonmaskable interrupt is generated upon the assertion of the VMEbus signal ACFAIL*. When the flag is `true` an interrupt handler, dealing with this interrupt, is installed and the ability to generate a nonmaskable interrupt upon the assertion of the ACFAIL* signal is enabled. In the case that the flag is `false` the ability to generate a nonmaskable interrupt upon the assertion of the ACFAIL* signal is enabled. (default: `false`)

`vme-ibox-addr` the least significant 16 bits of this 32-bit configuration parameter define the address at which the interrupt box (IBOX) of the SPARC FGA-5000 is accessible within the *short* address space (A16). Only the least significant 16 bits of this configuration parameter are considered, and the state of the remaining bits is ignored. Independent of the configuration parameter `vme-ibox-ena?` OpenBoot will set the address of the IBOX. (default: 0_{16})

`vme-ibox-ena?` indicates whether the interrupt box (IBOX), accessible in the *short* (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the IBOX is enabled. In the case that the NVRAM configura-

tion parameter is `false` the IBOX is not enabled.
The default value of this NVRAM configuration parameter is `false`.

`fmb-init?` controls whether the FMB system is initialized by OpenBoot. When this flag is `true` the FMB system is initialized according to the state of the NVRAM parameter listed below. In the case that the flag is `false` the FMB system is not initialized. The FMB system is initialized only during the initialization of the VMEbus interface, which means that the `vme-init?` configuration parameter must be `true`, in order to set up the FMB system. (default: `true`)

`fmb-slot#` specifies the *logical* slot number assigned to the FMB channels of the SPARC CPU-5V board. The value may be in the range one through 21, and preferably should be set in such a way that it corresponds with the number of an available VMEbus slot. (default: 1_{10})

`fmb-addr` specifies the address — the most significant eight bits of a 32-bit address — where the FMB system resides in the *extended* address space (A32) of the VMEbus. (default: fa_{16})

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the ***short*** **(A16)** address range.

`vme-a16-slave-addr` specifies the base address of the slave interface accessible in the *short* (A16) address range of the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a16-slave-size` specifies the size of the memory which is made available to the *short* (A16) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the `vme-a16-slave-ena?` configuration parameter is `true`!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a16-slave-ena?` indicates whether the slave interface, accessible in the *short* (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.
The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter `vme-init?` is `true` and the OpenBoot will initialize the slave interface according to the configuration parameters described above. When the `vme-a16-slave-ena?` configuration parameter is `true`, then OpenBoot will

initialize the VMEbus slave interface according to the NVRAM configuration parameters `vme-a16-slave-addr` and `vme-a16-slave-size`. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The *virtual* base address of the physical on-board memory provided for VMEbus slave accesses is stored in the variable `vme-a16-slave-mem`.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding MMU and IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter `vme-a16-slave-ena?` must be set to `false` to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Supposed that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

Note! The SPARC CPU-5V does **not** provide the ability to access its on-board memory from the VMEbus within the *short* (A16) address range. Therefore, the NVRAM configuration parameters associated with the A16 slave interface, control the access to the registers of the SPARC FGA-5000, which are accessible within the *short* address range. The configuration parameter `vme-a16-slave-size` is not of any importance and will be ignored.

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the ***standard*** **(A24)** address range.

`vme-a24-slave-addr` specifies the base address of the slave interface accessible in the *standard* (A24) address range of the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-slave-size` specifies the size of the memory which is made available to the *standard* (A24) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the `vme-a24-slave-ena?` configuration parameter is `true`!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-slave-ena?` indicates whether the slave interface, accessible in the *standard* (A24) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.
The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter vme-init? is true and the OpenBoot will initialize the slave interface according to the configuration parameters described above.
When the vme-a24-slave-ena? configuration parameter is true, then OpenBoot will initialize the VMEbus slave interface according to the NVRAM configuration parameters vme-a24-slave-addr and vme-a24-slave-size. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The *virtual* base address of the physical on-board memory provided for VMEbus slave accesses is stored in the variable vme-a24-slave-mem.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter vme-a24-slave-ena? must be set to false to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Supposed that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

The NVRAM configuration parameters listed below are associated with the slave interface accessible in the ***extended*** **(A32)** address range.

vme-a32-slave-addr specifies the base address of the slave interface accessible in the *extended* (A32 address range of the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a32-slave-size specifies the size of the memory which is made available to the *extended* (A32) address range of the VMEbus. When the value of this configuration parameter is zero OpenBoot will not initialize the slave interface, even if the vme-a24-slave-ena? configuration parameter is true!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a32-slave-ena? indicates whether the slave interface, accessible in the *extended* (A32) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the VMEbus slave interface is enabled. In the case that the NVRAM configuration parameter is false the VMEbus slave interface is not enabled, and any attempt to access the VMEbus slave interface from the VMEbus will lead to an error termination on the VMEbus.
The default value of this NVRAM configuration parameter is false.

In the case that the NVRAM configuration parameter vme-init? is true and the OpenBoot

will initialize the slave interface according to the configuration parameters described above. When the vme-a32-slave-ena? configuration parameter is true, then OpenBoot will initialize the VMEbus slave interface according to the NVRAM configuration parameters vme-a16-slave-addr and vme-a32-slave-size. It will provide the required amount of physical on-board memory and builds up the necessary MMU and IOMMU settings to make the memory available to the VMEbus. The *virtual* base address of the physical onboard memory provided for VMEbus slave accesses is stored in the variable vme-a32-slave-mem.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the slave interface completely according to the NVRAM configuration parameters associated with the slave interface.

In addition, this mechanism allows to report the parameters of the slave interface to an operating system loaded, which in turn provides its own memory and the corresponding IOMMU settings. In this case the VMEbus device driver is responsible for the access to the slave interface from the VMEbus. In general, the configuration parameter vme-a32-slave-ena? must be set to false to prevent OpenBoot from initialising and enabling the slave interface when an operating system will be loaded. Supposed that the slave interface is initialized and enabled by OpenBoot prior to loading the operating system, any access from the VMEbus to the slave interface while loading the operating system may alter memory and cause severe damage.

The NVRAM configuration parameters listed below are associated with the master interface to access the ***short*** **(A16)** address range.

vme-a16-master-addr specifies the base address of the *short* (A16) address range to be accessed on the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a16-master-size specifies the size of the area in the *short* (A16) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the vme-a16-master-ena? configuration parameter is true! If the specified size exceeds the size of the *short* (A16) address range, then it limits the specified size to 64 Kbyte. Due to the capabilities of the SPARC FGA-5000 OpenBoot will always adjust the specified size to 64 Kbyte.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

vme-a16-master-ena? indicates whether the master interface, to access the *short* (A16) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is true then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is false the VMEbus master interface is not enabled.
The default value of this NVRAM configuration parameter is false.

In the case that the NVRAM configuration parameter `vme-init?` is `true` OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a16-master-ena?` configuration parameter is `true`, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a16-master-mem`.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a16-master-ena?` must be set to `false` to prevent OpenBoot from initialising and enabling the master interface when an operating system will be loaded.

The NVRAM configuration parameters listed below are associated with the master interface to access the ***standard*** **(A24)** address range.

`vme-a24-master-addr` specifies the base address of the *standard* (A24) address range to be accessed on the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-master-size` specifies the size of the area in the *standard* (A24) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the `vme-a24-master-ena?` configuration parameter is `true`! If the specified size exceeds the size of the *standard* (A24) address range, then it limits the specified size to 16 Mbyte.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a24-master-ena?` indicates whether the master interface, to access the *standard* (A24) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus master interface is not enabled.
The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter `vme-init?` is `true` OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a24-master-ena?` configuration parameter is `true`, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a24-master-mem`.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the

NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a24-master-ena?` must be set to `false` to prevent OpenBoot from initializing and enabling the master interface when an operating system will be loaded.

The NVRAM configuration parameters listed below are associated with the master interface to access the ***extended*** **(A32)** address range.

`vme-a32-master-addr` specifies the base address of the *extended* (A32) address range to be accessed on the VMEbus.
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a32-master-size` specifies the size of the area in the *standard* (A24) address range of the VMEbus which will be accessed. When the value of this configuration parameter is zero OpenBoot will not initialize the master interface, even if the `vme-a32-master-ena?` configuration parameter is `true`!
The default value of this 32-bit NVRAM configuration parameter is zero (0).

`vme-a32-master-ena?` indicates whether the master interface, to access the *extended* (A32) address range of the VMEbus, should be enabled. When this NVRAM configuration parameter is `true` then the VMEbus master interface is enabled. In the case that the NVRAM configuration parameter is `false` the VMEbus master interface is not enabled.
The default value of this NVRAM configuration parameter is `false`.

In the case that the NVRAM configuration parameter `vme-init?` is `true` OpenBoot will initialize the master interface according to the configuration parameters described above. When the `vme-a24-master-ena?` configuration parameter is `true`, then OpenBoot will initialize the necessary registers in the master interface and provides the virtual memory to access the VMEbus. The *virtual* base address necessary to access the VMEbus is stored in the variable `vme-a24-master-mem`.
Thus, applications executed within the OpenBoot environment may benefit from this mechanism, because OpenBoot will initialize the master interface completely according to the NVRAM configuration parameters associated with the master interface.

In addition, this mechanism allows to report the parameters of the master interface to an operating system loaded, which in turn provides its own virtual memory to access the VMEbus. In this case the VMEbus device driver is responsible for providing the necessary virtual address range to access the VMEbus. In general, the configuration parameter `vme-a32-master-ena?` must be set to `false` to prevent OpenBoot from initializing and enabling the master interface when an operating system will be loaded.

4.2.12 DMA Controller Support

The commands listed below are available to control the DMA controller of the SPARC FGA-5000, as well as to get information about the actual state of the DMA controller.

`dma-irq-map!` (*mapping* —) selects the interrupt to be generated by the DMA controller when the DMA process terminated successfully or due to an error. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the certain VMEbus interrupt request level is asserted. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`dma-irq!` (*true* | *false*) enables or disables the interrupt to be generated by the DMA controller when the DMA process terminated successfully or due to an error. When the value *true* is passed to the command the interrupt is enabled. Otherwise — the value *false* is passed to the command — the interrupt is disabled.

`dma-ip?` (— *true* | *false*) checks whether an interrupt is pending because a DMA process has been terminated. The value *true* is returned when an interrupt is pending due to the termination of a DMA process. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`dma-ena` (—) enables the DMA controller and starts a DMA process.

`dma-dis` (—) disables the DMA controller and stops the DMA process currently running.

`dma-halt` (—) halts the DMA process currently running.

`dma-resume` (—) resumes the DMA process that has been halted before.

`dma-src-cap@` (— *data-capability address-capability*) returns the *data-capability* and *address-capability* currently defined for the source of the DMA process.

`dma-src-cap!` (*data-capability address-capability* —) sets the *data-capability* and *address-capability* for the source of the DMA process.
The constants listed below are available to specify the *data-capability* and the *address-capability*:

value	data-capability	address-capability
000_2	cap-d8	cap-a16
001_2	cap-d16	cap-a24
010_2	cap-d32	cap-a32

value	data-capability	address-capability
011_2	cap-blt	reserved
100_2	cap-mblt	reserved
101_2	reserved	reserved
110_2	reserved	reserved
111_2	reserved	reserved

`dma-dest-cap@` (— *data-capability address-capability*) returns the *data-capability* and *address-capability* currently defined for the destination of the DMA process.

`dma-dest-cap!` (*data-capability address-capability* —) sets the *data-capability* and *address-capability* for the destination of the DMA process.
The constants listed below are available to specify the *data-capability* and the *address-capability*:

value	data-capability	address-capability
000_2	cap-d8	cap-a16
001_2	cap-d16	cap-a24
010_2	cap-d32	cap-a32
011_2	cap-blt	reserved
100_2	cap-mblt	reserved
101_2	reserved	reserved
110_2	reserved	reserved
111_2	reserved	reserved

`dma-count@` (— *transfer-count*) returns the current state of the transfer count. The value *transfer-count* indicates the number of bytes to be transfer by the DMA controller.
Because the DMA controller only transfers a multiple of 32-bit data (*longword*, which is a *word* in the SPARC terminology), the command returns the appropriate number of *words* to be transferred.

`dma-count!` (*transfer-count* —) sets the number of bytes — *transfer-count* — to be transferred by the DMA controller.
Because the DMA controller only transfers a multiple of 32-bit data (*longword*, which is a *word* in the SPARC terminology), the command calculates the appropriate number of *words* to be transferred. The *transfer-count* is considered to be a modulo 4 Mbyte less four bytes number.

`dma-running?` (— *true* | *false*) checks whether the DMA controller is in the *running* state. The value *true* is returned when the DMA controller is currently running. Otherwise the value *false* is returned to indicate that the DMA controller is disabled.

`dma-waiting?` (— *true* | *false*) checks whether the DMA controller is in the *waiting* state. The value *true* is returned when the DMA controller is currently waiting, which means that it has been halted. Otherwise the value *false* is returned to indicate that the DMA controller is not waiting.

`dma-normal-terminated?` (— *true* | *false*) checks whether the DMA process has been terminated successfully. It returns the value *true* when the DMA process has been terminated successfully. Otherwise the value *false* is returned to indicate that the DMA process has been terminated due to a fail state, or because the DMA process is still in progress.

`dma-error-terminated?` (— *true* | *false*) checks whether the DMA process has been terminated unsuccessfully. It returns the value *true* when the DMA process has been terminated due to a fail state. Otherwise the value *false* is returned to indicate that the DMA process has been terminated due to normal termination, or because the DMA process is still in progress.

`.dma-stat` (—) displays the current state of the DMA Status Register.

```
ok .dma-stat
ERR:3 NT:0 HALT:0 RUN:0
ok
```

The fields **NT**, **HALT**, and **RUN** reflect the current state of the DMA controller. When the **NT** field is set to one (1), then the DMA controller terminated successfully (*normal termination*). In the case that the **HALT** field is set to one (1), then the DMA controller is halted — in general, this field is set along with the **RUN** field. The DMA controller is running when the **RUN** field is set to one (1). When one of the fields described previously is cleared (0), the DMA controller is **not** in the particular state.

Typically, the **ERR** field indicates the course of the DMA controller operation and may indicate the fail states listed in the table below:

Error Code	Description
0	Error occurred on source bus
1	Error occurred on destination bus
2	No error termination

Error Code	Description
3	No error termination

The following two commands used to initiate a DMA transfer do not set the data- and address capabilities of the source area and destination area. The capabilities **must** be appropriately set with the `dma-src-cap!` and `dma-dest-cap!` commands before the DMA transfer is started.

`dma-mem>vme` (*src-addr dest-addr count — true | false*) initiates a DMA transfer from the SBus to the VMEbus and **awaits** the termination of the DMA process.
The amount of bytes given by *count* is transferred from *src-addr* — an address area on the SBus (*virtual* address) — to *dest-addr* — an address area on the VMEbus (*physical* address). The command returns the value *true* when all data have been transferred successfully. Otherwise the value *false* is returned to indicate that an error occurred during the DMA process.
Because the DMA controller only transfers a multiple of 32-bit data (*longword*, which is a *word* in SPARC terminology), the command calculates the appropriate number of *words* to be transferred. Furthermore, the *count* is considered to be a modulo 4 Mbyte less four bytes number.

`dma-vme>mem` (*src-addr dest-addr count — true | false*) initiates a DMA transfer from the VMEbus to the SBus and **awaits** the termination of the DMA process.
The amount of bytes given by *count* is transferred from *src-addr* — an address area on the VMEbus (*physical* address) — to *dest-addr* — an address area on the SBus (*virtual* address). The command returns the value *true* when all data have been transferred successfully. Otherwise the value *false* is returned to indicate that an error occurred during the DMA process.
Because the DMA controller only transfers a multiple of 32-bit data (*longword*, which is a *word* in the SPARC terminology), the command calculates the appropriate number of *words* to be transferred. Furthermore, the *count* is considered to be a modulo 4 Mbyte less four bytes number.

4.2.13 Mailboxes and Semaphores

The commands described in this section control the mailboxes, the semaphores, and the interrupt box (IBOX).

`vme-mbox-take` (*mailbox#* — *true* | *false*) takes the mailbox semaphore specified by *mailbox#* and returns the value *true* when the mailbox semaphore has been taken successfully. The value *false* is returned when the mailbox semaphore has been taken already.
The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers

`vme-mbox-give` (*mailbox#* —) gives — releases — the mailbox semaphore specified by *mailbox#*.
The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vme-mbox-irq-map!` (*mapping mailbox#* —) selects the interrupt to be generated when the mailbox semaphore specified by *mailbox#* is taken. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the mailbox semaphore is taken. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`vme-mbox-irq-ena` (*mailbox#* —) allows the VMEbus interface to generate an interrupt when the mailbox specified by *mailbox#* is taken. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vme-mbox-irq-dis` (*mailbox#* —) disables the interrupt to be generated when the mailbox specified by *mailbox#* is taken. The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vme-mbox-ip?` (*mailbox* — *true* | *false*) checks whether an interrupt is pending because the mailbox semaphores specified by *mailbox#* have been taken. The value *true* is returned when an interrupt is pending because the mailbox semaphore has been taken. Otherwise the value *false* is returned to indicate that no interrupt is pending.
The value of *mailbox#* may be one of the values in the range zero through 15. Each value specifies one of the 16 Mailbox Registers.

`vme-sem-take` (*semaphore#* — *true* | *false*) takes a semaphore specified by *mailbox#* and returns the value *true* when the semaphore has been taken successfully. The value

false is returned when the semaphore has been taken already.
The value of *semaphore#* may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

`vme-sem-give` (*semaphore#* —) gives — releases — the semaphore specified by *semaphore#*.
The value of *semaphore#* may be one of the values in the range zero through 47. Each value specifies one of the 48 Semaphore Registers.

The Interrupt Box is **only** accessible from the VMEbus within the *short* address space (A16). Any byte access — reading or writing — may lead the SPARC FGA-5000 to generate an interrupt. The address of the interrupt box within the *short* address space may be any byte location in the range 0000_{16} through $FFFF_{16}$.
The commands listed below are available to control and initialize the Interrupt Box.

`vme-ibox-irq-map!` (*mapping* —) selects the interrupt to be generated when the interrupt box is being accessed.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the interrupt box is accessed. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`vme-ibox-irq-ena` (—) allows the VMEbus interface to generate an interrupt when the interrupt box is accessed.

`vme-ibox-irq-dis` (—) disables the interrupt to be generated when the interrupt box is accessed.

`vme-ibox-ip?` (— *true* | *false*) checks whether an interrupt is pending because the interrupt box has been accessed. The value *true* is returned when an interrupt is pending because the interrupt box has been accessed. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`vme-ibox-ena` (—) enables the interrupt box.

`vme-ibox-dis` (—) disables the interrupt box.

`vme-ibox-addr@` (— *addr*) returns the physical address *addr* of the interrupt box.

`vme-ibox-addr!` (*addr* —) sets the physical address *addr* of the interrupt box.

As shown in the example below the first command sets the address of the interrupt box. The interrupt box is accessible at the address 4002_{16} within the VMEbus *short* address space. An SBus IRQ 5 is generated by the SPARC FGA-5000 whenever the interrupt box is accessed from the VMEbus. The fourth command enables the interrupt box.

```
ok h# 4002 vme-ibox-addr!
ok 5 vme-ibox-irq-map!
ok vme-ibox-irq-ena
ok vme-ibox-ena
ok
```

4.2.14 FORCE Message Broadcast

The commands listed below are available to control the FORCE Message Broadcast (FMB) system and to obtain status information about the state of the FMB system.

`fmb-super-only` (*true* | *false* —) allows or prevents the FMB message registers from being accessed in the *non-privileged* mode. When the value *true* is passed to the command the FMB message register is accessible in the *privileged* mode, as well as in the *non-privileged* mode. Otherwise — the value *false* is passed to the command — the FMB message registers are accessible in the *privileged* mode only.

`fmb-ena` (*channel#* —) enables the FMB channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two FMB channels.

`fmb-dis` (*channel#* —) disables the FMB channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero to one. Each value specifies one of the two FMB channels.

`fmb!` ([*true* | *false*] *channel#* —) enables or disables the FMB channel specified by *channel#*. When the value *true* is passed to the command the FMB channel is enabled. Otherwise — the value *false* is passed to the command — the FMB channel is disabled.

`fmb-slot@` (— *slot#*) returns the slot number *slot#* assigned to the FMB channels.

`fmb-slot!` (*slot#* —) assigns the slot number *slot#* to the FMB channels. The value of *slot#* may be one of the values in the range zero to 21. Each value specifies a specific slot.

`fmb-addr@` (— *fmb-space*) returns the most significant eight bits — the *fmb-space* — of the 32-bit VMEbus address the FMB will respond to when an FMB transaction on the VMEbus is detected.

`fmb-addr!` (*fmb-space* —) sets the most significant eight bits — the *fmb-space* — of the 32-bit VMEbus address the FMB will respond to when an FMB transaction on the VMEbus is detected.

`fmb-irq-map!` (*mapping channel#* —) selects the interrupt to be generated when the FMB message has been accepted, or rejected by the channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the FMB message is accepted or rejected. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings

`fmb-irq!` ([*true* | *false*] *channel#* —) enables or disables the interrupt to be generated when either an FMB message has been accepted or rejected by the channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
When the value *true* is passed to the command the interrupt is enabled. Otherwise — the value *false* is passed to the command — the interrupt is disabled.

`fmb-ip?` (*channel#* — *true* | *false*) checks whether an interrupt is pending because an FMB message has been accepted or rejected by the channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
The value *true* is returned when an interrupt is pending. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`fmb-accepted-ip?` (*channel#* — *true* | *false*) checks whether an interrupt is pending because an FMB message has been accepted by the channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
The value *true* is returned when an interrupt is pending because a message has been accepted. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`fmb-rejected-ip?` (*channel#* — *true* | *false*) checks whether an interrupt is pending because an FMB message has been accepted by the channel specified by *channel#*. The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
The value *true* is returned when an interrupt is pending because an message has been rejected. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`fmb-rejected-ip-clear` (*channel#* —) clears a pending *message rejected* interrupt generated by the channel specified by *channel#*.

`fmb-msg@` (*channel#* — *message true* | *false*) fetches a message — a 32-bit data — from the FMB channel specified by *channel#*. The *message* and the value *true* are returned when an FMB is available. Otherwise the value *false* is returned to indicated that no FMB message is available.

`fmb-msg!` (*message slot-list channel# — true | false*) sends the *message* — a 32-bit data — to all FMB channels identified by the *slot-list* and *channel#*. The value *true* is returned when the message has been sent out successfully. Otherwise the value *false* is returned to indicate that one or more FMB channels have rejected the message.
The value of *channel#* may be one of the values in the range zero through one. Each value specifies one of the two FMB channels.
The value of *slot-list* identifies the hosts participating in the FMB transaction. Each bit of the slot list is associated with a host identified by a *unique* FMB slot number. The first bit — bit 0 — relates to the host with the FMB slot number one (1); the second bit — bit 1 — relates to the host with the FMB slot number two (2); and so forth.
Because the FMB system allows only up to 21 hosts, the command considers only the least significant bits of the parameter *slot-list* (bit 0 through 20).

`fmb-init` (*slot# fmb-space* —) performs all rudimentary steps to initialize the SPARC FGA-5000 in such a way that the subsequent FMB cycles are carried out using the `fmb-msg!` command.
The slot number *slot#* specifies the slot number the FMB channels are associated with. The value of *slot#* may be one of the values in the range zero to 21. Each value specifies a specific slot.
The last available register set in the SPARC FGA-5000 is initialized to carry out an FMB cycle on the VMEbus within the appropriate VMEbus address area that has been specified by *fmb-space*. The parameter *fmb-space* defines the most significant eight bits (one of 256 16-Mbyte pages) of the VMEbus address where the FMB area is located. The capabilities of this VMEbus master range are A32/D32 and write posting is disabled. The variable `fmb-va` contains the virtual address to be accessed to execute an FMB cycle on the VMEbus.

The example below assigns the slot number 15_{10} to the FMB channels available (all other hosts must have a different FMB slot number). The FMB address space is set to FA_{16} which means that the FMB system is accessed when the address $FAXX.XXXX_{16}$ appears on the VMEbus address lines (the least significant 24 bits are used to select a specific FMB channel and specific hosts). And the second command enables the second FMB channel.

```
ok d# 15 h# fa fmb-init
ok true 1 fmb!
ok h# 1234AA55 h# 0010.800f 1 fmb-msg!
ok 1 fmb-msg@
ok .s 2drop
1234AA55 ffffffff
ok 1 fmb-msg@
ok .s drop
0
ok
```

Finally the message $1234.AA55_{16}$ is sent to the second FMB channel available on the hosts with the FMB slot number one, two, three, four, 15, and 20. Because the message is sent to the host with the FMB slot number 15 — the host that sent the message —, the message is read from the second FMB channel on the host, as shown by the fourth command. When the FMB channel is read again, and supposed the host did not receive another FMB message, the command `fmb-msg@` will return the value *false* to indicate that no more messages are available.

4.2.15 Diagnostic

The commands listed and described in this section are used to obtain various error status information from the SPARC FGA-5000 in the case of write posting errors and SBus errors.

`wperr-irq-map!` (*mapping* —) selects the interrupt to be generated when a write posting error occurs on the SBus or VMEbus. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the write post error occures. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`wperr-irq!` (*true* | *false* —) enables or disables the interrupt to be generated when a write posting error occurs on the SBus or VMEbus. When the value *true* is passed to the command the interrupt is enabled. Otherwise — the value *false* is passed to the command — the interrupt is disabled.

`vme-wperr-ip?` (— *true* | *false*) checks whether an interrupt is pending because a write posting error occurred on the VMEbus. The value *true* is returned when an interrupt is pending due to a write posting error. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`sbus-wperr-ip?` (— *true* | *false*) checks whether an interrupt is pending because a write posting error occurred on the SBus. The value *true* is returned when an interrupt is pending due to a write posting error. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`sbus-wperr-clear` (— *error-addr*) reads the SBus Write Posting Error Address Register and returns the address *error-addr*.

`vme-wperr-clear` (— *error-addr*) reads the VMEbus Write Posting Error Address Register and returns the address *error-addr*.

`slerr-irq-map!` (*mapping* —) selects the interrupt to be generated when a late error occurs on the SBus. The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the late error occurs. The value of *mapping* may be one of

the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`slerr-irq!` (*true* | *false* —) enables or disables the interrupt to be generated when a late error occurs on the SBus.When the value *true* is passed to the command the interrupt is enabled. Otherwise — the value *false* is passed to the command — the interrupt is disabled.

`slerr-ip?` (— *true* | *false*) checks whether an interrupt is pending because a late error occurred on the SBus. The value *true* is returned when an interrupt is pending due to a late error. Otherwise the value *false* is returned to indicate that no interrupt is pending.

`slerr-clear` (— *error-addr*) reads the SBus Late Error Address Register and returns the address *error-addr*.

4.2.16 Miscellanea

The commands listed in this section are used to control miscellaneous functions in the SPARC FGA-5000.

`freeze-intr-mapping` (—) prevents the SYSFAIL*, ACFAIL* and ABORT Interrupt Select and Enable Registers from being modified by setting the *freeze* bit in the Miscellaneous Control and Status Register. This mechanism is intended to prevent the appropriate Interrupt Control and Status Register from being modified after it has been initialized once.

`dtb-driver-ena` (—) enables all VMEbus DTB drivers.

`dtb-driver-dis` (—) disables all VMEbus DTB drivers.

`vme-timer-ena` (—) enables the VMEbus transaction timer.

`vme-timer-dis` (—) disables the VMEbus transaction timer.

`vme-timeout@` (— *timeout*) returns the VMEbus transaction timer timeout value in use. The value of *timeout* may be one of the values in the range one through three. Each value identifies a particular timeout period as shown in the table below.

`vme-timeout!` (*timeout* —) sets the VMEbus transaction timer timeout according to the given *timeout*. The value of *timeout* may be one of the values in the range one through three. When the value being specified is not in the range one through three, then the command selects the longest timeout period automatically.
The values select a particular timeout period. The table below lists all possible values:

timeout	$t_{transaction-timeout}$
1	32 us
2	128 us
3	512 us

Table 41: VMEbus Transaction Timer Timeout Values

4.3 Standard Initialization of the VMEbus Interface

Besides the initialization performed according to the state of the NVRAM configuration parameters, the VMEbus interface — mainly the SPARC FGA-5000 — is initialized as described in the subsections below.

4.3.1 SPARC FGA-5000 Registers

The registers of the SPARC FGA-5000 are accessible beginning at offset $0FFF.FE00_{16}$ within the SBus slot 5 and occupy the last 512 bytes in this slot ($0FFF.FE00_{16}$ ... $0FFF.FFFF_{16}$). This corresponds with the physical address range $7FFF.FE00_{16}$ through $7FFF.FFFF_{16}$.

The area in the range $0FE0.0000_{16}$ through $0FFF.FDFF_{16}$ is available for any application. Preferably, this area may be used to access the *standard* (A24, max 16 MB) and *short* (A16, max 64 KB) address space of the VMEbus.

4.3.2 VMEbus Transaction Timer

The SPARC FGA-5000 contains a VMEbus transaction timer which is disabled after a RESET. This timer is enabled during the initialisation phase of OpenBoot and the transaction timeout period is set to the longest possible value (512 us).

4.3.3 SBus Rerun Limit

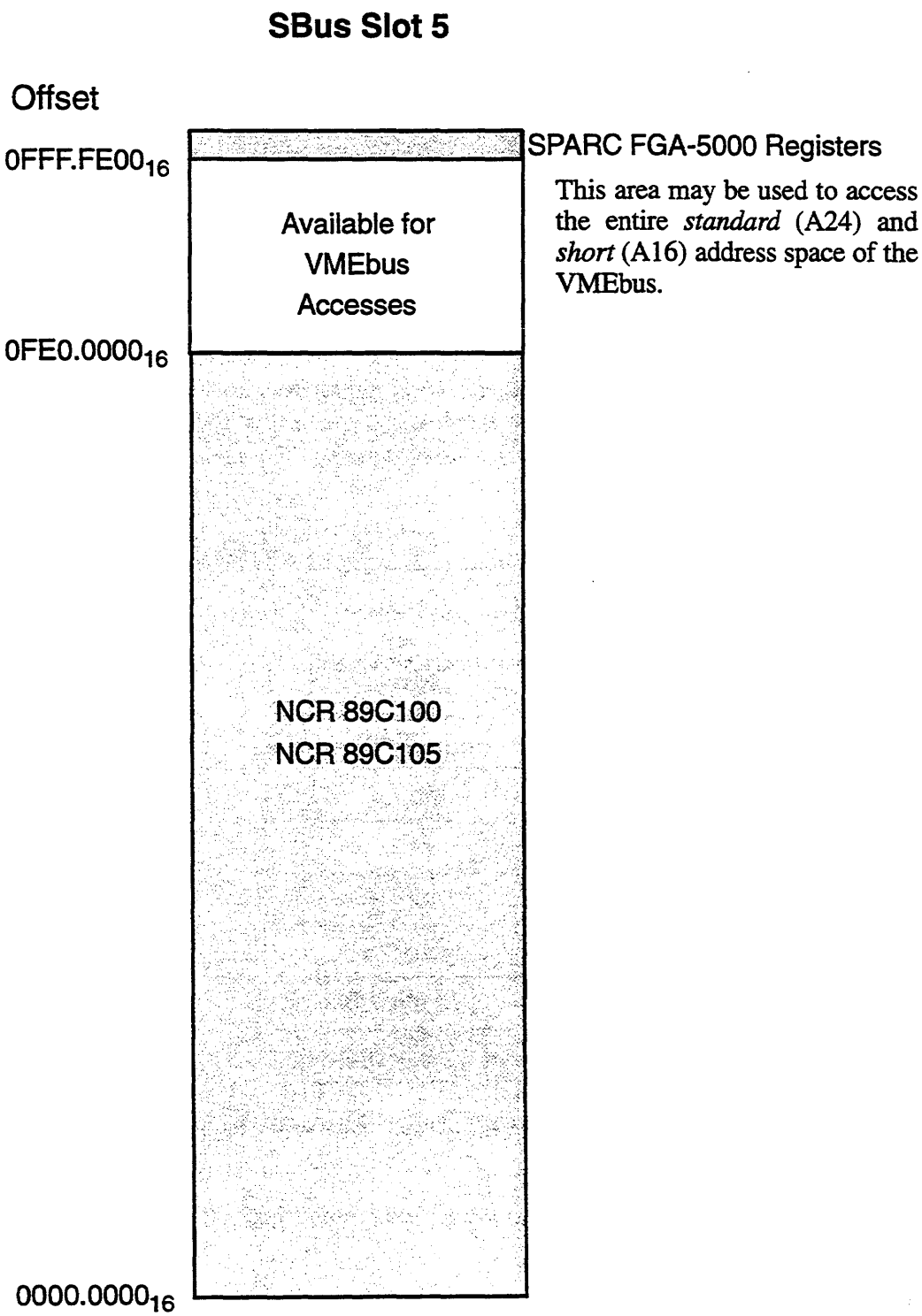
The SBus Rerun Limit counter, within the SPARC FGA-5000, is **disabled** to avoid any unproper behaviour of the system.

4.3.4 Interrupts

The SPARC FGA-5000 is initialized in such a way that in the case of the occurrence of one of the events listed below, a nonmaskable interrupt (level 15 interrupt) is generated:

1.) Pressing the ABORT switch

4.3.5 SBus Slot 5 Address Map



4.4 System Configuration

4.4.1 Watchdog Timer

`wd-ena` (—) enables and starts the watchdog timer.

`wd-dis` (—) stops and disables the watchdog timer.

`wd-timeout@` (— *timeout*) returns the watchdog timer's reference value in use. The value of *timeout* may be one of the values in the range zero through seven. Each value identifies a particular timeout period as shown in the table below.

`wd-timeout!` (*timeout* —) sets the watchdog timer's reference value for timeout according to the given *timeout*. The value of *timeout* may be one of the values in the range zero through seven. Only the least significant three bits of the value *timeout* are considered. The values select a particular timeout period. The table below lists all possible values:

timeout	$t_{wd\text{-}timeout\text{-}min}$
0	408 ms
1	1.68 s
2	6.7 s
3	26.8 s
4	1 min 48 s
5	7 min 9 s
6	28 min 38 s
7	1 h 54 min

Table 42: Watchdog Timer Timeout Values

`wd-nmi-ena` (—) allows an interrupt to generate when half of the watchdog time has expired.

`wd-nmi-dis` (—) disables the interrupt's ability to generate when half of the watchdog time has expired.

`wd-irq-map!` (*mapping* —) selects the interrupt to be generated when half of the watch-

dog time has expired.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when half of the watchdog time has expired. The value of *mapping* may be one of the values in the range of zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`wd-nmi-clear` (—) clears a pending interrupt caused by the watchdog timer when half of the watchdog time has expired.

`wd-ip?` (— *true* | *false*) checks whether an interrupt is pending due to an interrupt generated by the watchdog timer when half of the watchdog time has expired. The value *true* is returned when the interrupt is pending; otherwise the value *false* is returned.

`wd-restart` (—) resets the watchdog timer and starts a new time count. In particular the command invokes one of the commands **`vsi-wdt-restart@`** or **`vsi-wdt-restart!`** to restart the watchdog timer.

The watchdog timer is started by the commands listed below:

```
ok 3 wd-timeout!
ok vsi-nmi wd-irq-map!
ok wd-nmi-ena
ok wd-ena
ok
```

In this example the *watchdog timer timeout* is set to 26.8 seconds, and a **nonmaskable** interrupt is generated whenever half of the watchdog time has expired. The OpenBoot already contains an interrupt handler dealing with the interrupt generated by the watchdog timer, and this interrupt handler increments an internal variable by one, whenever the watchdog timer emits an interrupt. The state of this variable is determined by:

```
ok wdnmi-occurred? ?
6
ok
```

This variable is cleared — set to zero — by

```
ok wdnmi-occurred? off
ok
```

`wd-reset?` (— *true* | *false*) determines whether a reset has been generated because the watchdog timer has expired. If a reset has been generated because the watchdog timer reached the timeout value, then the value *true* is returned; otherwise the value *false* is returned.

4.4.2 Watchdog Timer NVRAM Configuration Parameters

The NVRAM configuration parameters listed below are available to control the initialisation and operation of the watchdog timer. The current state of these configuration parameters are displayed using the `printenv` command, and are modified using either the `setenv`, or the `set-default` command provided by OpenBoot.

`wd-ena?` controls whether the watchdog timer has to be started. When the flag is `true`, then the watchdog timer is started after it has been initialized according to the configuration parameter `wd-timeout`. If the flag is `false` the watchdog timer is not started, but the watchdog timer registers are initialized according to the configuration parameter `wd-timeout`. (default: `false`)

`wd-timeout` contains the timeout value of the watchdog timer and is a value in the range 0 to 7. Each value selects a particular timeout period. Independent of the state of the configuration parameter `wd-ena?` the timeout value is stored in the appropriate watchdog timer register. (default: 7_{10})

4.4.3 Abort Switch

`abort-switch?` (— *true* | *false*) determines the current state of the abort switch. The value *true* is returned when the abort switch is pressed. And the value *false* is returned when the abort switch is released.

`abort-irq-map!` (*mapping* —) selects the interrupt to be generated when the abort switch is pressed.
The parameter *mapping* defines the interrupt asserted by the SPARC FGA-5000 when the abort switch is presssed. The value of *mapping* may be one of the values in the range zero through seven. Each value specifies one of the eight SPARC FGA-5000 interrupt request lines. The table "Interrupt Mapping." on page 109 lists all allowed mappings.

`abort-nmi-ena` (—) allows an interrupt to generate when the abort switch is pressed.

`abort-nmi-dis` (—) disables the interrupt's ability to generate when the abort switch is being pressed.

`abort-ip?` (— *true* | *false*) checks whether an interrupt is pending because the abort switch has been pressed. The value *true* is returned when the interrupt is pending; otherwise the vallue *false* is returned.

`abort-nmi-clear` (—) clears a pending interrupt caused by the abort switch.

4.4.4 Abort Switch NVRAM Configuration Parameter

The NVRAM configuration parameter listed below is available to control the initialisation and operation of the abort switch. The current state of these configuration parameters are displayed using the `printenv` command, and are modified using either the `setenv`, or the `set-default` command provided by OpenBoot.

`abort-ena?` controls whether the abort switch has to be enabled. When this flag is `true` the abort switch is enabled and has the same effect as pressing the **STOP-A** key on an available keyboard. If the flag is `false` then the abort switch is disabled. (default: `false`)

4.4.5 LEDs, Seven-Segment Display and Rotary Switch

The commands described below are available to control the seven-segment LED display, the user LEDs, and are used to retrieve information about the state of the rotary switch.

`diag-led!` (*byte* —) stores the data *byte* passed to the command in the register used to control the seven-segment display.

`>7-seg-code` (*u* — *7-seg-code*) converts the value *u* to its corresponding seven-segment code *7-seg-code*. Only the least significant four bits of the value *u* are considered.

`led!` (*colour freq led#* —) controls the user LED identified by *led#*. The value of *led#* may be either zero or one. The value zero specifies the **first** user LED, and the value one specifies the **second** user LED. The command only considers the state of the bit 0 of the value *led#*.
The parameters *colour* and *freq* define the colour of the LED and the frequency at which the LED is blinking. The following constants are defined to specify the *colour*: **`black`**, **`green`**, **`red`**, and **`yellow`**. When the colour **`black`** is specified the LED is turned off.
The constants **`no-blinking`**, **`slow`**, **`moderate`**, and **`fast`** are available to specify a frequency. The constant **`no-blinking`** causes the LED to be turned on permanently.

The following example shows how to let the second user LED blink at about 2 Hz (moderate) in red

```
ok red moderate 1 led!
ok
```

`led-on` (*led#* —) turns the user LED identified by *led#* on. The value of *led#* may be either zero or one. The value zero specifies the **first** user LED, and the value one specifies the **second** user LED. The command only considers the state of the bit 0 of the value

led#.

`led-off` (*led#* —) turns the user LED identified by *led#* off. The value of *led#* may be either zero or one. The value zero specifies the **first** user LED, and the value one specifies the **second** user LED. The command only considers the state of the bit 0 of the value *led#*.

`led?` (*led#* — *true* | *false*) determines the state of the LED identified by *led#*, and returns either *true* or *false* to indicate if the LED is turned on or off. The value of *led#* may be either zero or one. The value zero specifies the **first** user LED, and the value one specifies the **second** user LED. The command only considers the state of the bit 0 of the value *led#*.
When the LED is turned on, then the value *true* is returned; otherwise the value *false* is returned.

`toggle-led` (*led#* —) determines the state of the user LED identified by *led#*, and turns the LED on or off. The LED is turned on when it was turned off before, and vice versa. The value of *led#* may be either zero or one. The value zero specifies the **first** user LED, and the value one specifies the **second** user LED. The command only considers the state of the bit 0 of the value *led#*.

`rotary-switch@` (— *byte*) returns the current state of the rotary switch. The value of *byte* may be one of the values in the range zero through 15. The value zero corresponds to the position 0 of the rotary switch, the value one corresponds to position 1, and so forth.

4.4.6 Reset

The command listed below are available to initiate various RESETs, and to obtain information about a previous RESET.

`vme-sysreset` (—) asserts the VMEbus SYSRESET* signal and thus causes a *system* reset.

`reset-call` (—) forces a *local* reset. This command provides the same function as the OpenBoot command `reset`.

`vme-sysreset-in!` (*true* | *false*) allows or prevents the board from being reset by the assertion of the VMEbus SYSRESET* signal. When the value *true* is passed to the command the board will be reset whenever the VMEbus SYSRESET* signal is asserted. Otherwise — the value *false* is passed to the command — the board will not be reset by the assertion of the SYSRESET* signal.

`sbus-reset?` (— *true* | *false*) determines whether the last reset occurred was due to an SBus reset. The value *true* is returned when the last reset was because of an SBus reset. Otherwise it returns the value *false* to indicate that the last reset was **not** because of an SBus reset.

`wdt-reset?` (— *true* | *false*) determines whether a reset has been generated because the watchdog timer has expired. If a reset has been generated because the watchdog timer reached the timeout value, then the value *true* is returned; otherwise the value *false* is returned.

`vme-sysreset?` (— *true* | *false*) determines whether the last reset occurred was due to the assertion of the VMEbus SYSRESET* signal. The value *true* is returned when the last reset was a VMEbus SYSRESET* reset. Otherwise it returns the value *false* to indicate that the last reset was **not** a VMEbus SYSRESET* reset.

`vme-sysreset-call?` (— *true* | *false*) determines whether the last reset occurred was due to a VMEbus SYSRESET* call. The value *true* is returned when the last reset was because of a VMEbus SYSRESET* call. Otherwise it returns the value *false* to indicate that the last reset was **not** a VMEbus SYSRESET* call.

A VMEbus SYSRESET* call is done by clearing the SYSRESET bit in the SPARC FGA-5000's Miscellaneous Control and Status Register.

`reset-call?` (— *true* | *false*) determines whether the last reset occurred was due to a local reset call. The value *true* is returned when the last reset was because of a local reset call. Otherwise it returns the value *false* to indicate that the last reset was **not** a local reset call.

A local reset call is done by clearing the RESET bit in the SPARC FGA-5000's Miscellaneous Control and Status Register.

`vme-reset-call?` (— *true* | *false*) determines whether the last reset occurred was due to a reset call initiated by an access via the VMEbus. The value *true* is returned when the last reset was because of a reset call. Otherwise it returns the value *false* to indicate that the last reset was **not** because of a reset call initiated by an access via the VMEbus.

A reset call is done by clearing the LOCRESET bit in the SPARC FGA-5000's GLobal Control and Status Register.

4.5 Flash Memory Support

4.5.1 Flash Memory Programming

The commands listed below are available to access and program the flash memories available on the SPARC CPU-5V.

`flash-messages` (— *vaddr*) returns the virtual address of the *variable* `flash-messages`. The state of this variable controls whether the words to erase and program the flash memories will display messages while erasing or programming the flash memories. Messages will not be displayed after *turning off* this variable by `flash-messages off`, and are displayed after *turning on* this variable by `flash-messages on`.

`flash-va` (— *vaddr*) returns the virtual base address *vaddr* of the flash memory programming window. The virtual address returned is only valid when the flash memories have been previously prepared for accessing using the `select-flash` word.

`boot-flash-va` (— *vaddr*) returns the virtual base address *vaddr* of the BOOT flash memory.

`user-flash-va` (— *vaddr*) returns the virtual base address *vaddr* of the USER flash memory. When the USER flash memory is not accessible directly, but only through the flash memory programming window, then the address returned is zero. On the SPARC CPU-5V the USER flash memory is accessible only through the flash memory programming window. Thus, the commands described above have to be used to access the USER flash memory.

`select-flash` ("USER<eol>" | "BOOT<eol>" —) prepares either the BOOT flash memories, or the USER flash memories for programming. In detail, the number and size of the available flash memories are determined, as well as the size of the flash memory programming window. The flash memory programming window is mapped and the virtual base address of the window is stored internally, and may be obtained by using the word `flash-va`.

`user-flash?` (— *true* | *false*) checks whether the BOOT flash memory or the USER flash memory is accessible through the flash memory programming window. It returns *true* in the case that the USER flash memory is accessible through the programming window; otherwise it returns *false*.

`move>flash` (*source-addr dest-addr count* —) programs the selected flash memory beginning at *dest-addr* with a number of bytes, specified by *count*, stored at *source-addr*.

`flash>move` (*source-addr dest-addr count* —) copies a number of bytes, specified by *count*, from the selected flash memory beginning at *source-addr* to *dest-addr*. The flash memory is accessed through the flash memory programming window for reading data from the memory. Thus, the flash memory has to be prepared for accessing using the

command `select-flash`.

`fill-flash` (*dest-addr count pattern* —) fills the selected flash memory beginning at *dest-addr* with a particular *pattern*. The number of bytes to be programmed in the flash memory is given by *count*.

`erase-flash` (*device-number* —) erases a flash memory device identified by its *device-number*. The devices are numbered beginning from zero (0).

`c!-flash` (*byte addr* —) stores the *byte* at the location within the selected flash memory identified by *addr*.

`w!-flash` (*half-word addr* —) stores the *half-word* (16 bits) at the location within the selected flash memory identified by *addr*.

`l!-flash` (*word addr* —) stores the *word* (32 bits) at the location within the selected flash memory identified by *addr*.

The USER flash memory is prepared for programming by:

```
ok select-flash USER
USER flash memory is selected for programming
Flash memory programming window at $ffe98000 size 512 Kbyte
512 Kbyte BOOT flash memory is available at $ffe58000.
2048 Kbyte USER flash memory is available.
ok
```

As shown above, the word `select-flash` informs the user that the USER flash memory has been made accessible through the flash memory programming window. It displays the base address (*virtual* address) of the window and its size.
The total amount of the available BOOT flash memory and USER flash memory is displayed, too. After the USER flash memory has been prepared for programming, all commands described above operate on the USER flash memory. And the BOOT flash memory is only read and programmed by these commands when the BOOT flash memory has been prepared for these operations by:

```
ok select-flash BOOT
BOOT flash memory is selected for programming
Flash memory programming window at $ffe98000 size 512 Kbyte
512 Kbyte BOOT flash memory is available at $ffe58000.
2048 Kbyte USER flash memory is available.
ok
```

To read data from the selected flash memory — in the current context from the USER flash memory — the command `flash>move` is used as follows:

```
ok flash-va h# 10.0000 h# 20.0000 flash>move
ok
```

The contents of the entire USER flash memory is copied to main memory beginning at address 10.0000_{16} . A specific area within the selected flash memory is read by:

```
ok flash-va h# 6.8000 + h# 10.0000 h# 5.8c00 flash>move
ok
```

and copies 363520 bytes beginning from address `flash-va` + 6.8000_{16} to main memory beginning at address 10.0000_{16} .

4.5.2 Flash Memory Device

The device tree of OpenBoot for the SPARC CPU-5V contains a device node associated with the USER flash memories. Thus, it is possible to load an executable image stored in the available USER flash into memory and start such an executable.

The device is called "**flash-memory@0,71300000**" and is attached to the device node "**/obio**". The device alias **flash** is available as an abbreviated representation of the flash memory device path.
The vocabulary of the flash memory device includes the standard commands recommended for a *byte* device. The words of this vocabulary are only available when the flash memory device has been selected as shown below:

```
ok cd flash
ok words
close          open          selftest      reset      load
write-blocks   read-blocks   seek          write      read
max-transfer   block-size
ok selftest .
0
ok device-end
ok
```

The example listed above, selects the flash memory device and makes it the current node. The word **words** displays the names of the *methods* of the VMEbus device. And the third command calls the method **selftest** and the value returned by this method is displayed. The last command *unselects* the current device node, leaving no node selected.
When the command **select-dev** is used to select the flash memory device, the NVRAM configuration parameters **bootflash-#megs** and **bootflash-#devices** have to be set properly, before the device can be selected.

The NVRAM configuration parameters listed below are available to control the loading of an image from the USER flash memory. The current state of these configuration parameters is displayed using the `printenv` command, and is modified using either the `setenv`, or the `set-default` command provided by OpenBoot.

`bootflash-#megs` specifies the amount of available USER flash memory in megabyte.

(default: 0 Megabyte)

`bootflash-#devices` specifies the number of available USER flash memory devices. (default: no devices)

`bootflash-load-base` specifies the address where the data loaded from the available flash memory are stored when the **`load`** or **`boot`** command, provided by OpenBoot, is used to load an image from the flash memory.
When this parameter is set to -1 — which is the parameter's default value — then the image loaded from the flash memory is stored beginning at the address *addr.* But when the value of the configuration parameter differs from -1, then the image loaded from the flash memory is stored beginning at the address specified by the configuration parameter **`bootflash-load-base`**. And the same address is stored in the variable **`load-base`** maintained by OpenBoot.

The methods listed below are available in the vocabulary of the flash memory device:

`open` (— *true*) prepares the package for subsequent use. The value `true` is returned when the device has been opened successfully; otherwise the value `false` is returned. Usually, the fail state is indicated when the NVRAM configuration parameters `bootflash-#megs` and `bootflash-#devices` are not consistent.

`close` (—) frees all resources allocated by open.

`reset` (—) puts the flash memory device into *quiet* state.

`selftest` (— *error-number*) always returns the value zero.

`read` (*addr lenth — actual*) reads at most *length* bytes from the flash memory device into memory beginning at address *addr.* If *actual* is zero or negative, the read failed. The value of *length* may not always be a multiple of the device's normal block size.

`write` (*addr length — actual*) discards the information passed to the command and always returns zero to indicate that the device does not support this function.

`seek` (*offset file# — error?*) seek to byte *offset* within the file identified by *file#.* The flash memory device package maintains an internal position counter that is updated whenever a method to read data from or to store data in the flash memories is called. If *offset* and *file#* are both zero, then the internal position counter is reset to offset zero, otherwise the value of *offset* is assigned to the internal position counter, and a subsequent access to the flash memories starts at the offset selected.
Because the flash memory device does not support any file system, the parameter *file#* is ignored, except in the case mentioned above.
When the seek succeeded the value of *error?* is zero, otherwise the value -1 is retun-

rned to indicate the fail state.

`read-blocks` (*addr block# #blocks — #read*) reads the number of blocks identified by *#blocks* of length *block-size* bytes, each from the device beginning at the device bvlock *block#*, into memory at address *addr*. It returns the number of blocks actually read (*#read*).

`write-blocks` (*addr block# #blocks — #written*) discards the information passed to the command and always returns zero to indicate that the device does not support this function.

`block-size` (— *bytes*) returns the size in bytes *bytes* of a block which is always the size of the flash memory programming window.

`max-transfer` (— *bytes*) returns the size in bytes *bytes* of the largest single transfer the device can perform. The command returns a multiple of `block-size`.

`load` (*addr — length*) reads a stand-alone program from the flash memory beginning at offset 0_{16} and stores it beginning at address *addr*. It returns the number of bytes *length* read from the flash memory.
This method considers the state of the NVRAM configuration parameter `boot-flash-load-base`: when this parameter is set to -1 — which is the parameter's default value — then the image loaded from the flash memory is stored beginning at the address *addr*. But when the value of the configuration parameter differs from -1, then the image loaded from the flash memory is stored beginning at the address specified by the configuration parameter `bootflash-load-base`. And the same address is stored in the variable `load-base` maintained by OpenBoot.

4.5.3 Loading and Executing Programs from USER Flash Memory

Besides the ability to load and execute an executable image from disk, or via a network, or other components, the OpenBoot for the SPARC CPU-5V provides a convenient way to load and execute an executable image from the available USER flash memory. The executable image to be loaded has to be either a **binary** image (a.out format), a **FORTH** program, or a **FCode** program.
As mentioned at the beginning of this section the device alias **`flash`** is available as an abbreviated representation of the flash memory device. The command listed below is used to explicitly load and execute an image from the flash memory:

```
ok boot flash
```

The following NVRAM configuration parameters can be modified to determine whether or not the system will load an executable image automatically after a power-up cycle or system reset:

```
auto-boot?
boot-device
```

Assuming, that the SPARC CPU-5V is equipped with one USER flash memory device which size is 1Mbyte, then commands listed in the following have to be used to load and execute an image from the flash memory automatically after a power-up cycle or system reset:

```
ok setenv bootflash-#devices 1
bootflash-#devices =  1
ok setenv bootflash-#megs 1
bootflash-#megs =     1
ok setenv boot-device flash
boot-device =         flash
ok setenv auto-boot? true
auto-boot? =          true
ok reset
```

4.5.4 Controlling the Flash Memory Interface

The commands listed below are available to control the flash memory interface. These commands are used to make a specific flash memory device available in the flash memory programming window, and to control the flash memory programming voltage.

`flash-vpp-on` (—) turns the programming voltage on.

`flash-vpp-off` (—) turns the programming voltage off.

`userprom-select-page` (*page* —) makes a *page* (one of a eight possible 512 KB pages) of a USER flash memory available in the *flash memory programming window*.

`bootprom-select-page` (*page* —) makes a *page* (one of a eight possible 512 KB pages) of a BOOT flash memory available in the *flash memory programming window*.

`select-bootprom-1` (—) makes the first BOOT flash memory device available in the *flash memory programming window*.

`select-bootprom-2` (—) makes the second BOOT flash memory device available in the *flash memory programming window*.

`select-bootprom` (*device-number* —) makes a BOOT flash memory device, identified by its *device-number*, available in the *flash memory programming window*. The devices are numbered beginning from zero (0).

`select-userprom-1` (—) makes the first USER flash memory device available in the

flash memory programming window.

`select-userprom-2` (—) makes the second USER flash memory device available in the *flash memory programming window.*

`select-userprom` (*device* —) makes a USER flash memory device, identified by its *device-number*, available in the *flash memory programming window.* The devices are numbered beginning from zero (0).

4.6 Onboard Interrupts

Besides the interrupt handlers already available in the standard OpenBoot, the OpenBoot of the SPARC CPU-5V provides further handlers that deal with the interrupts generated by following:

- one of the VMEbus interrupt levels one to seven;
- the assertion and negation of the SYSFAIL* signal;
- the assertion of the ACFAIL* signal;
- pressing the ABORT switch;
- the Watchdog Timer, when half the time has expired.

4.6.1 VMEbus Interrupts

The interrupt handlers for any VMEbus interrupt are not installed automatically by OpenBoot; however, appropriate words are available to *activate* and *deactivate* an interrupt handler serving a specific VMEbus interrupt. Such an interrupt handler is activated by:

```
ok 0 pil!
ok 3 5 install-vme-intr-handler
ok
```

The **pil!** command decreases the processor interrupt level to allow the processor to respond to all interrupts. By default, OpenBoot sets the mask to 13 and allows the processor to respond to interrupts above interrupt level 13. The second command installs the interrupt handler that deals with the VMEbus interrupt level 5. Furthermore, this command specifies that an SBus interrupt level 3 will be generated upon the occurrence of a VMEbus interrupt 5. Any of the seven SBus interrupt levels may be specified to be generated upon a VMEbus interrupt.
OpenBoot maintains seven variables called `vme-intr{1|2|3|4|5|6|7}-vector` which are modified by the VMEbus interrupt handlers. In general, the interrupt handlers store the vector obtained during an interrupt acknowledge cycle in the appropriate variable. The state of these variables is displayed by

```
ok .vme-vectors
1: --   2: --   3: --   4: --   5: 33   6: --   7: --
ok
```

By default, the value -1 (`true`) is assigned to these variables to indicate that no VMEbus interrupt occurred. So, the word `.vme-vectors`, as shown above, will display "`--`" indicating that no interrupt occurred; otherwise it shows the vector obtained (a value in the range 0 to FF_{16}).
Another way to display the state of a variable used to store the interrupt vector is

```
ok vme-intr5-vector ?
```

```
33
ok
```

and the variable is set to -1 (`true`) by

```
ok vme-intr5-vector on
ok
```

An interrupt handler is removed and the corresponding interrupt is disabled by

```
ok 5 uninstall-vme-intr-handler
ok
```

All interrupt handlers serving all VMEbus interrupts are installed by

```
ok 0 pil!
ok 8 1 do i i install-vme-intr-handler loop
ok
```

In this case, all interrupt handlers are installed and the VMEbus interrupt to SBus interrupt mapping is as follows: SBus interrupt level 1 is generated upon the occurrence of a VMEbus interrupt 1; SBus interrupt level 2 is generated upon the occurrence of a VMEbus interrupt 2; and so forth.

4.6.2 SYSFAIL Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated upon the assertion and negation of the SYSFAIL* signal. This handler need not to be installed because it is already installed by OpenBoot.
By default, the interrupts that will be emitted by a status change of the SYSFAIL* signal are disabled and have to be enabled by

```
ok vme-sysfail-assert-nmiena
ok vme-sysfail-negate-nmiena
ok
```

which enable the generation of a nonmaskable interrupt whenever the SYSFAIL* signal is asserted and negated.

When an nonmaskable interrupt occurred due to the assertion of the SYSFAIL* signal, then the appropriate interrupt handler increments the variable `sysfail-asserted?` by one to report the occurrence of such an interrupt. The variable `sysfail-negated?` is incremented by the interrupt handler when the SYSFAIL* signal has been negated and caused a non-maskable interrupt. The state of both variables are obtained by

```
ok sysfail-asserted? ?
```

```
0
ok
```

and

```
ok sysfail-negated? ?
1
ok
```

And these variables are cleared — set to zero — by

```
ok sysfail-asserted? off
ok sysfail-negated? off
ok
```

4.6.3 ACFAIL Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated upon the assertion of the ACFAIL* signal. This handler need not to be installed because it is already installed by OpenBoot.
By default, the interrupt that will be emitted by asserting the ACFAIL* signal is disabled and has to be enabled by

```
ok vme-acfail-assert-irq-ena
ok
```

which enables the generation of a nonmaskable interrupt whenever the ACFAIL* signal is asserted.

When a nonmaskable interrupt occurred due to the assertion of the ACFAIL* signal, then the appropriate interrupt handler increments the variable `acfail-asserted?` by one to report the occurrence of such an interrupt. The state of this variable is obtained by

```
ok acfail-asserted? ?
2
ok
```

And the variable is cleared — set to zero — by

```
ok acfail-asserted? off
ok
```

4.6.4 ABORT Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated by pressing the front panel abort switch. This handler need not be installed because it is already installed by OpenBoot.
By default, the interrupt that will be emitted when the abort switch has been pressed is disabled and has to be enabled by

```
ok abort-nmi-ena
ok
```

which enables the generation of a nonmaskable interrupt whenever the abort switch is pressed.

When a nonmaskable interrupt occurred due to pressing the abort switch, then the appropriate interrupt handler increments the variable `abort-occurred?` by one to report the occurrence of such an interrupt. The state of both variables are obtained by

```
ok abort-occurred? ?
7
ok
```

And these variables are cleared — set to zero — by

```
ok abort-occurred? off
ok
```

Besides the effects described above, the pressing of the abort switch has the same effect as giving the **Stop-A** keyboard command. The program currently running is aborted and the FORTH interpreter appears immediately.

4.6.5 Watchdog Timer Interrupt

OpenBoot for the SPARC CPU-5V already includes an interrupt handler to serve the non-maskable interrupt generated by the watchdog timer when half of the time has expired. This handler need not to be installed because it is already installed by OpenBoot.
By default, the interrupt that will be emitted by the watchdog timer is disabled — the watchdog timer is disabled — and has to be enabled by

```
ok wd-nmi-ena
ok wd-ena
ok
```

In this example a nonmaskable interrupt is generated whenever half of the watchdog time has expired. The interrupt handler included in OpenBoot restarts the watchdog timer to ensure that the watchdog time will not expire and cause a reset. Additionally, the interrupt handler

increments the variable wdnmi-occurred? by one whenever the watchdog timer emits an interrupt. The state of this variable is determined by

```
ok wdnmi-occurred? ?
6
ok
```

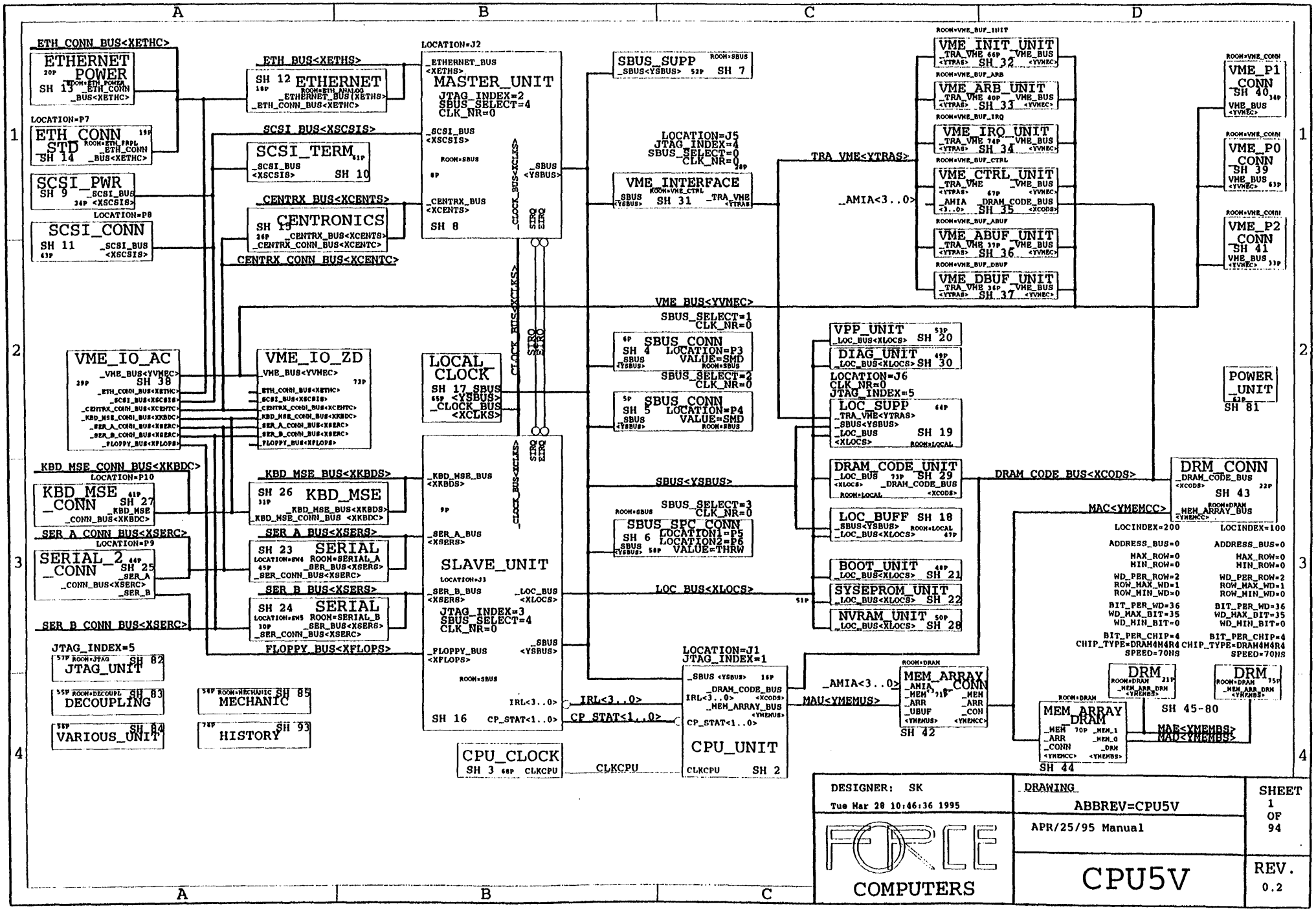
This variable is cleared — set to zero — by

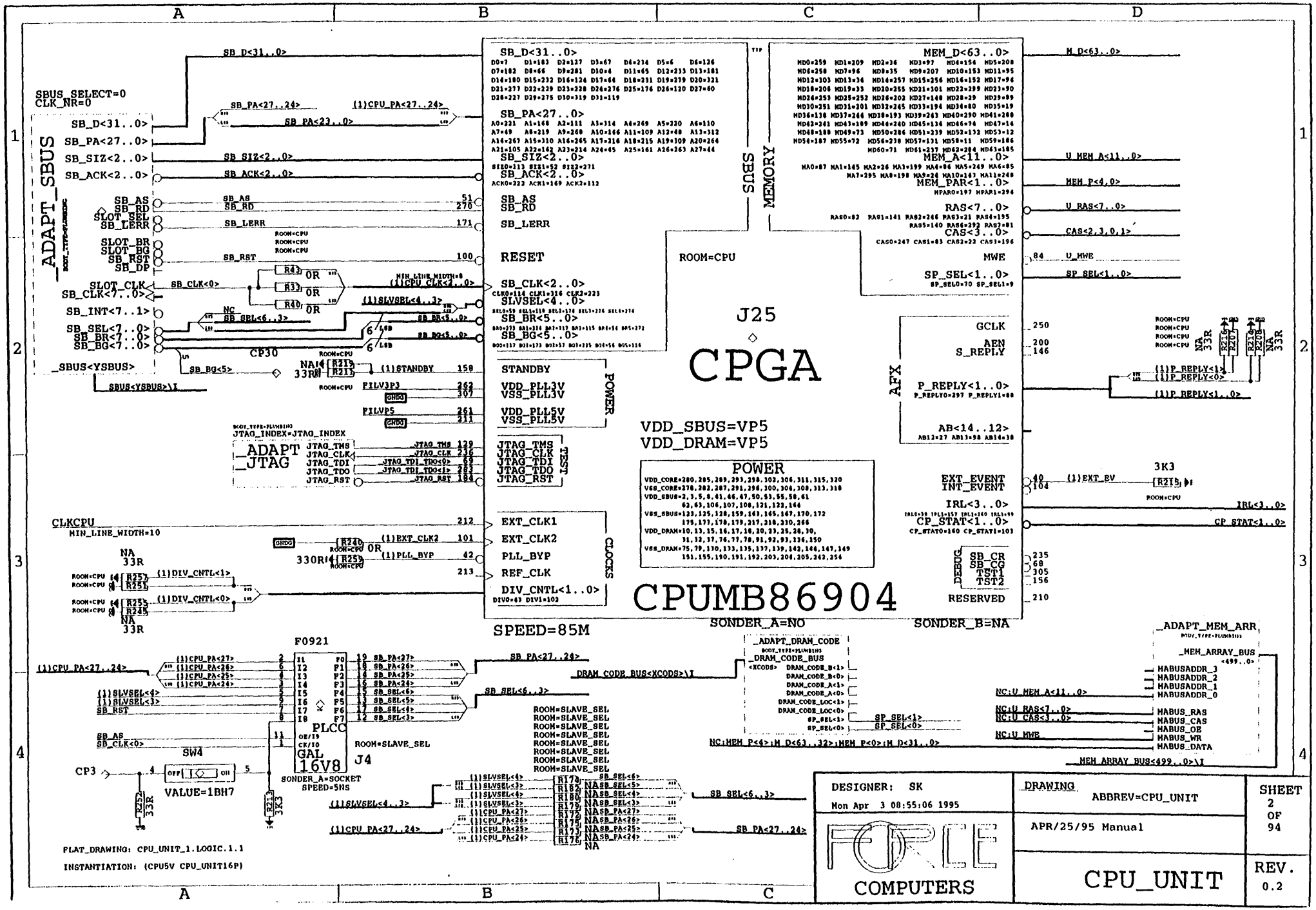
```
ok wdnmi-occurred? off
ok
```

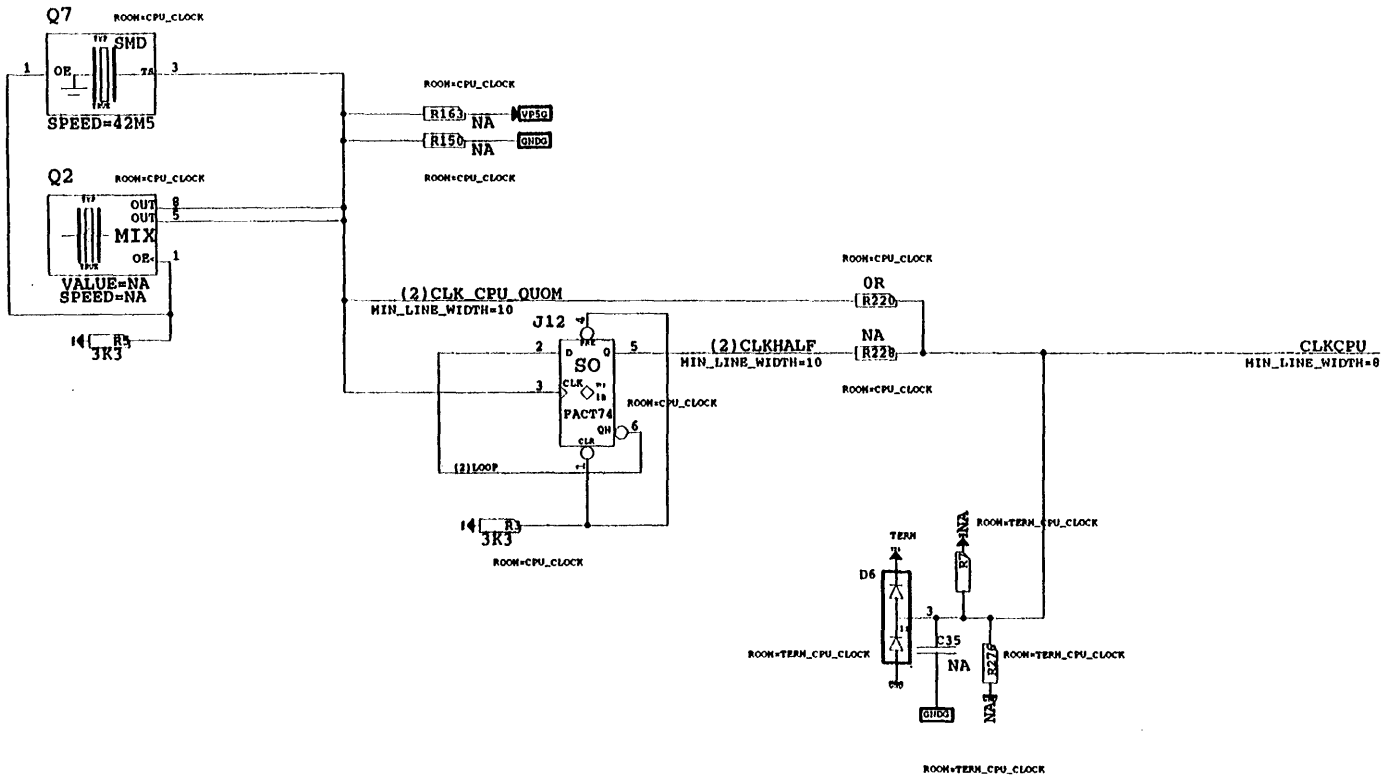
SECTION 5 CIRCUIT SCHEMATICS

5. CPU-5V Schematics

Copies of the CPU-5V schematics are found on the next page. The schematics contain the signal and unit cross references as well as the history of the schematics.


ETHERNET POWER SH 13
ETH_CONN_BUS<XETHC>
ETH_BUS<XETHS>
SH 12 ETHERNET
ETH_CONN_BUS<XETHC>
LOCATION=J2
MASTER_UNIT
JTAG_INDEX=2
SBUS_SELECT=4
CLK_NR=0
LOCATION=P7
ETH_CONN_STD SH 14
SCSI_BUS<XSCSIS>
SCSI_TERM SH 10
SCSI_PWR SH 9
SCSI_CONN SH 11
LOCATION=P8
CENTRX_BUS<XCENTS>
CENTRONICS SH 15
CENTRX_CONN_BUS<XCENTC>
SBUS_SUPP SH 7
LOCATION=J5
JTAG_INDEX=4
SBUS_SELECT=0
CLK_NR=0
VME_INTERFACE SH 31
TRA_VME<YTRAS>
_AMIA<3..0>
VME_INIT_UNIT SH 32
VME_ARB_UNIT SH 33
VME_IRQ_UNIT SH 34
VME_CTRL_UNIT SH 35
VME_ABUF_UNIT SH 36
VME_DBUF_UNIT SH 37
VME_P1 CONN SH 40
VME_P0 CONN SH 39
VME_P2 CONN SH 41
VME_BUS<YVMEC>
VME_IO_AC SH 38
VME_IO_ZD
LOCAL_CLOCK SH 17
CLOCK_BUS<XCLKS>
SBUS_SELECT=1
CLK_NR=0
SBUS_CONN SH 4
LOCATION=P3
VALUE=SMD
SBUS_SELECT=2
CLK_NR=0
SBUS_CONN SH 5
LOCATION=P4
VALUE=SMD
VPP_UNIT SH 20
DIAG_UNIT SH 30
LOCATION=J6
CLK_NR=0
JTAG_INDEX=5
LOC_SUPP SH 19
POWER_UNIT SH 81
KBD_MSE_CONN_BUS<XKBDC>
LOCATION=P10
KBD_MSE_CONN SH 27
KBD_MSE_BUS<XKBDS>
SH 26 KBD_MSE
SER_A_CONN_BUS<XSERC>
LOCATION=P9
SERIAL_2_CONN SH 25
SER_A_BUS<XSERS>
SH 23 SERIAL
SER_B_BUS<XSERS>
SH 24 SERIAL
SER_B_CONN_BUS<XSERC>
FLOPPY_BUS<XFLOPS>
SLAVE_UNIT
LOCATION=J3
JTAG_INDEX=3
SBUS_SELECT=4
CLK_NR=0
SBUS<YSBUS>
LOC_BUS<XLOCS>
IRL<3..0>
CP_STAT<1..0>
SH 16
SBUS_SPC_CONN SH 6
LOCATION1=P5
LOCATION2=P6
VALUE=THRW
DRAM_CODE_UNIT SH 29
DRAM_CODE_BUS<XCODS>
LOC_BUFF SH 18
BOOT_UNIT SH 21
SYSEPROM_UNIT SH 22
NVRAM_UNIT SH 28
DRM_CONN SH 43
MAC<YMEMCC>
LOCINDEX=200
LOCINDEX=100
ADDRESS_BUS=0
MAX_ROW=0
MIN_ROW=0
WD_PER_ROW=2
ROW_MAX_WD=1
ROW_MIN_WD=0
BIT_PER_WD=36
WD_MAX_BIT=35
WD_MIN_BIT=0
BIT_PER_CHIP=4
CHIP_TYPE=DRAH4M4R4
SPEED=70NS
DRM SH 45-80
MEM_ARRAY_CONN SH 42
MEM_ARRAY_DRAM SH 44
MAE<YMEMBS>
MAD<YMEMBS>
LOCATION=J1
JTAG_INDEX=1
CPU_UNIT
CPU_CLOCK SH 3
CLKCPU
JTAG_UNIT SH 82
DECOUPLING SH 83
MECHANIC SH 85
VARIOUS_UNIT SH 84
HISTORY SH 93
DESIGNER: SK
Tue Mar 28 10:46:36 1995
DRAWING
ABBREV=CPU5V
APR/25/95 Manual
COMPUTERS
CPU5V
SHEET 1 OF 94
REV. 0.2


CPUMB86904
CPGA
J25
SPEED=85M
SONDER_A=NO
SONDER_B=NA
VDD_SBUS=VP5
VDD_DRAM=VP5
ROOM=CPU
SBUS
MEMORY
AFX
POWER
CLOCKS
TEST
DEBUG
_ADAPT_SBUS
_ADAPT_JTAG
_ADAPT_DRAM_CODE
_ADAPT_MEM_ARR
SBUS_SELECT=0
CLK_NR=0
SB_D<31..0>
SB_PA<27..0>
SB_SIZ<2..0>
SB_ACK<2..0>
MEM_D<63..0>
MEM_A<11..0>
MEM_PAR<1..0>
RAS<7..0>
CAS<3..0>
SP_SEL<1..0>
SB_CLK<2..0>
SLVSEL<4..0>
SB_BR<5..0>
SB_BG<5..0>
STANDBY
VDD_PLL3V
VSS_PLL3V
VDD_PLL5V
VSS_PLL5V
JTAG_TMS
JTAG_CLK
JTAG_TDI
JTAG_TDO
JTAG_RST
RESET
GCLK
AEN
S_REPLY
P_REPLY<1..0>
AB<14..12>
EXT_EVENT
INT_EVENT
IRL<3..0>
CP_STAT<1..0>
RESERVED
EXT_CLK1
EXT_CLK2
PLL_BYP
REF_CLK
DIV_CNTL<1..0>
CLKCPU
F0921
PLCC
GAL
16V8
SW4
CP3
VALUE=1BH7
SONDER_A=SOCKET
SPEED=5NS
ROOM=SLAVE_SEL
J4
DESIGNER: SK
Mon Apr 3 08:55:06 1995
DRAWING
ABBREV=CPU_UNIT
APR/25/95 Manual
SHEET
2
OF
94
REV.
0.2
CPU_UNIT
FORCE
COMPUTERS



FLAT_DRAWING: CPU_CLOCK_2.LOGIC.1.1

INSTANTIATION: (CPU5V CPU_CLOCK68P)

DESIGNER: SK Mon Apr 3 08:54:45 1995	DRAWING ABBREV=CPU_CLOCK	SHEET 3 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	CPU_CLOCK	REV. 0.2

P3
VALUE=%VALUE SONDER_A=STAND_OFF

FSBU

1	GND	2	2 SB_BR<1>	49	49 SB_CLK<0>	50	50 SB_BG<1>
	3 SB_SEL<1>		4 SB_INT<1>		51 SB_AS		GND
	5 SB_D<0>		6 SB_D<2>		53 SB_D<1>		54 SB_D<3>
	7 SB_D<4>		8 SB_INT<2>		55 SB_D<5>		+5V
9	9 SB_D<6>	10	10 SB_D<8>	57	57 SB_D<7>	58	58 SB_D<9>
	11 SB_D<10>		12 SB_INT<3>		59 SB_D<11>		GND
	13 SB_D<12>		14 SB_D<14>		61 SB_D<13>		62 SB_D<15>
	15 SB_D<16>		16 SB_INT<4>		63 SB_D<17>		+5V
17	17 SB_D<19>	18	18 SB_D<21>	65	65 SB_D<18>	66	66 SB_D<20>
	19 SB_D<23>		20 SB_INT<5>		67 SB_D<22>		GND
	21 SB_D<25>		22 SB_D<27>		69 SB_D<24>		70 SB_D<26>
	23 SB_D<29>		24 SB_INT<6>		71 SB_D<28>		+5V
25	25 SB_D<31>	26	26 SB_SIZ<0>	73	73 SB_D<30>	74	74 SB_SIZ<1>
	27 SB_SIZ<2>		28 SB_INT<7>		75 SB_RD		GND
	29 SB_PA<0>		30 SB_PA<2>		77 SB_PA<1>		78 SB_PA<3>
	31 SB_PA<4>		32 SB_LERR		79 SB_PA<5>		+5V
33	33 SB_PA<6>	34	34 SB_PA<8>	81	81 SB_PA<7>	82	82 SB_PA<9>
	35 SB_PA<10>		36 SB_ACK<0>		83 SB_PA<11>		GND
	37 SB_PA<12>		38 SB_PA<14>		85 SB_PA<13>		86 SB_PA<15>
	39 SB_PA<16>		40 SB_ACK<1>		87 SB_PA<17>		+5V
41	41 SB_PA<18>	42	42 SB_PA<20>	89	89 SB_PA<19>	90	90 SB_PA<21>
	43 SB_PA<22>		44 SB_ACK<2>		91 SB_PA<23>		GND
	45 SB_PA<24>		46 SB_PA<26>		93 SB_PA<25>		94 SB_PA<27>
47	47 SB_DP	48	-12V	95	95 SB_RST	96	+12V

SBUS_SELECT=SBUS_SELECT
CLK_NR=CLK_NR

SB_D<31..0> — SB_D<31..0>
SB_PA<27..0> — SB_PA<27..0>
SB_SIZ<2..0> — SB_SIZ<2..0>
SB_ACK<2..0> — SB_ACK<2..0>

SB_AS — SB_AS
SB_RD — SB_RD
SB_SEL<1> — SLOT_SEL
SB_LERR — SB_LERR

SB_BR<1> — SLOT_BR
SB_BG<1> — SLOT_BG
SB_RST — SB_RST
SB_DP — SB_DP

SB_CLK<0> — SLOT_CLK
SB_CLK<7..0>

SB_INT<7..1> — SB_INT<7..1>

SB_SEL<7..0>
SB_BR<7..0>
SB_BG<7..0>

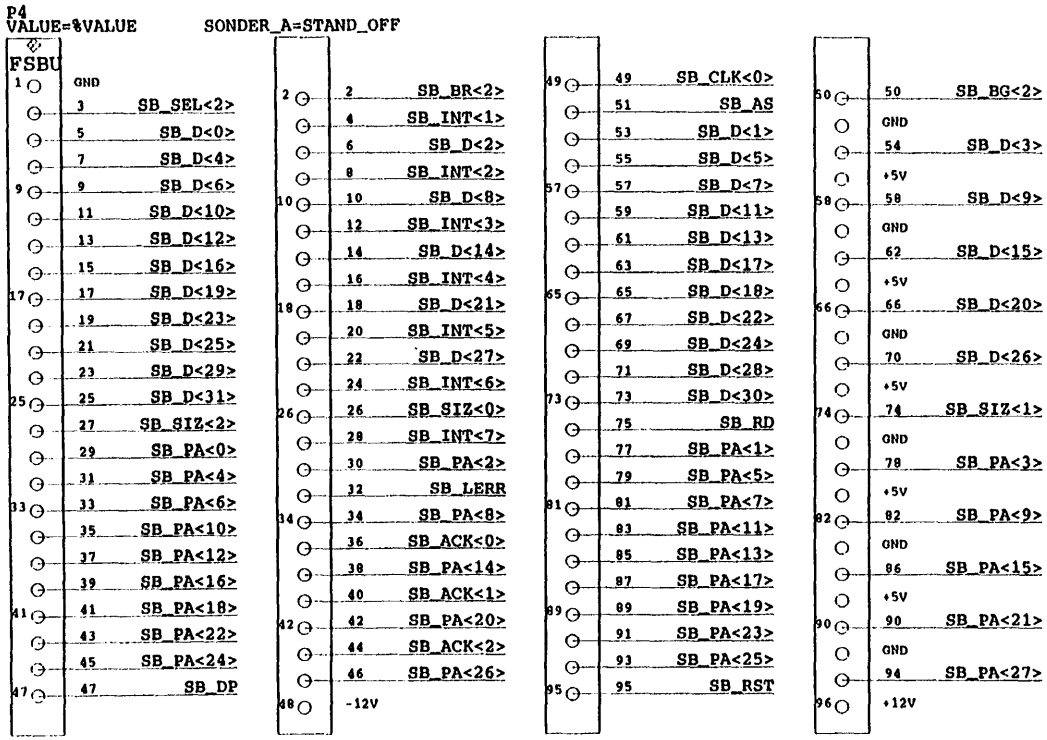
_SBUS<YSBUS>

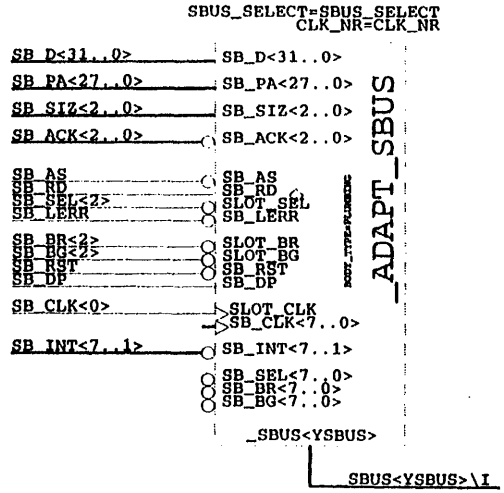
SBUS<YSBUS>\I

_ADAPT_SBUS

FLAT_DRAWING: SBUS_CONN_4.LOGIC.1.1
INSTANTIATION: (CPU5V SBUS_CONN6P)

DESIGNER: SK Mon Apr 3 00:54:31 1995	DRAWING ABBREV=SBUS_CONN	SHEET 4 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SBUS_CONN	REV. 0.2

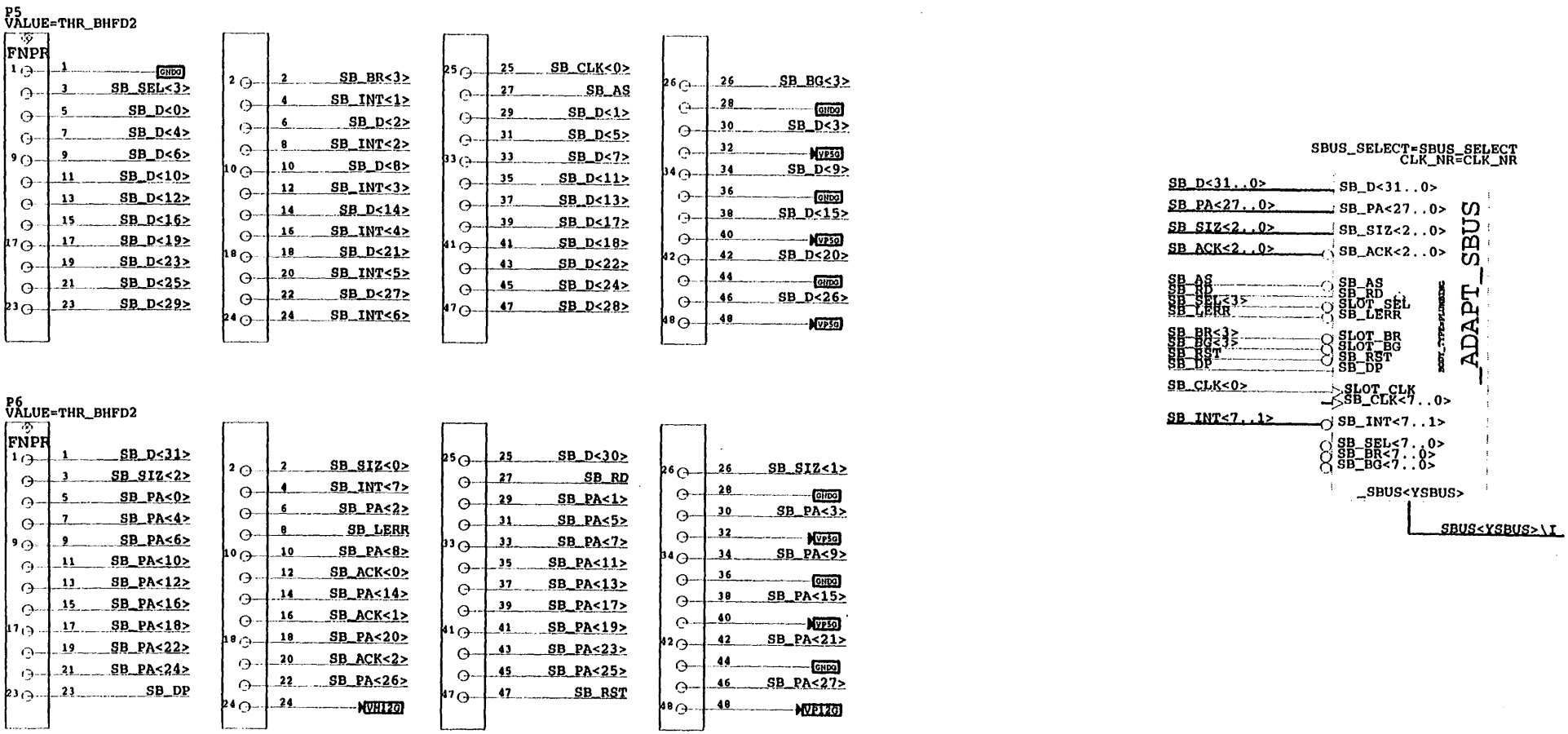




FLAT_DRAWING: SBUS_CONN_3.LOGIC.1.1

INSTANTIATION: (CPU5V SBUS_CONN5P)

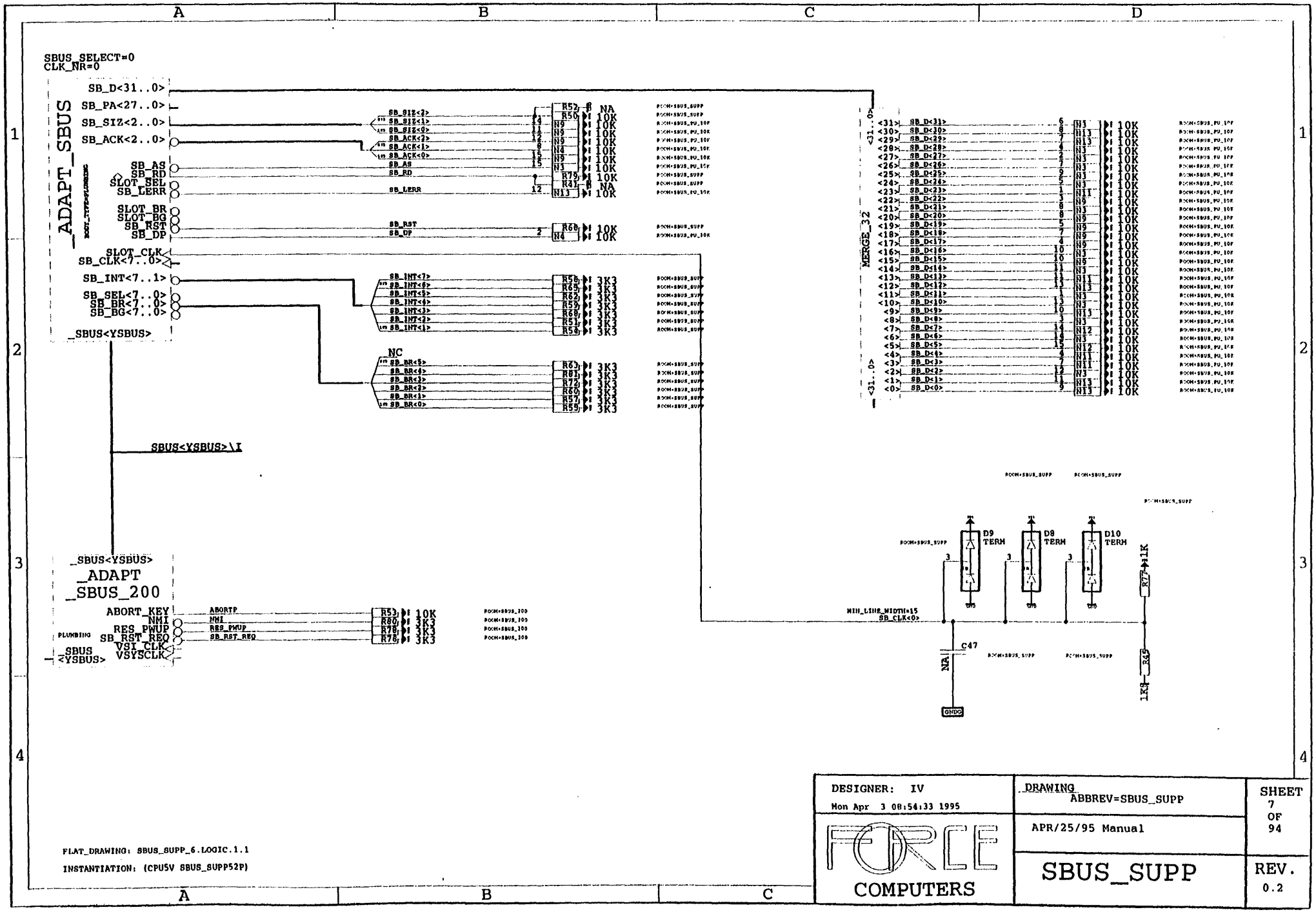
DESIGNER: SK Mon Apr 3 08:54:29 1995	DRAWING ABBREV=SBUS_CONN APR/25/95 Manual	SHEET 5 OF 94
FORCE COMPUTERS	SBUS_CONN	REV. 0.2



FLAT_DRAWING: SBUS_SPC_CONN_5.LOGIC.1.1

INSTANTIATION: (CPU5V SBUS_SPC_CONN58P)

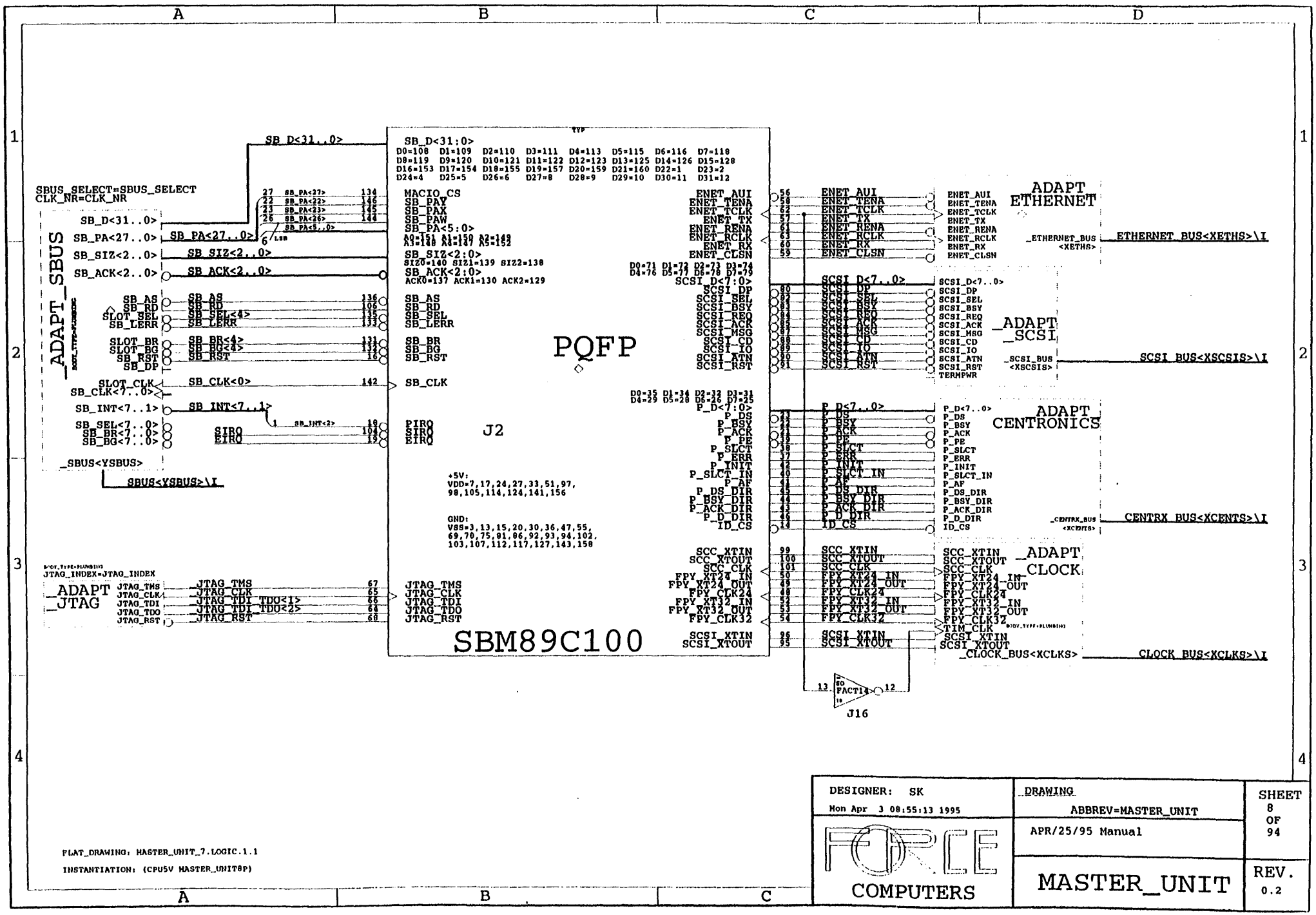
DESIGNER: SK Mon Apr 3 08:54:26 1995	DRAWING ABBREV=SBUS_SPC_CONN	SHEET 6 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SBUS_SPC_CONN	REV. 0.2



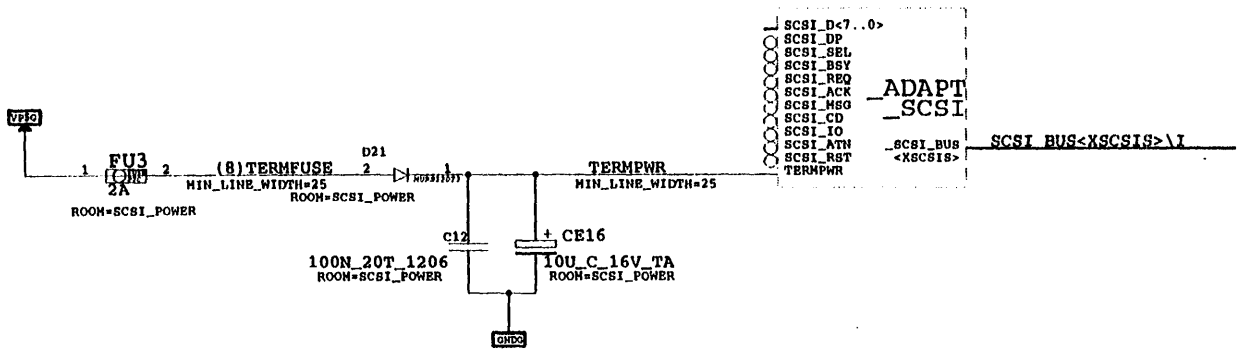
FLAT_DRAWING: SBUS_SUPP_6.LOGIC.1.1

INSTANTIATION: (CPU5V SBUS_SUPP52P)

DESIGNER: IV Mon Apr 3 00:54:33 1995	DRAWING ABBREV=SBUS_SUPP	SHEET 7 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SBUS_SUPP	REV. 0.2



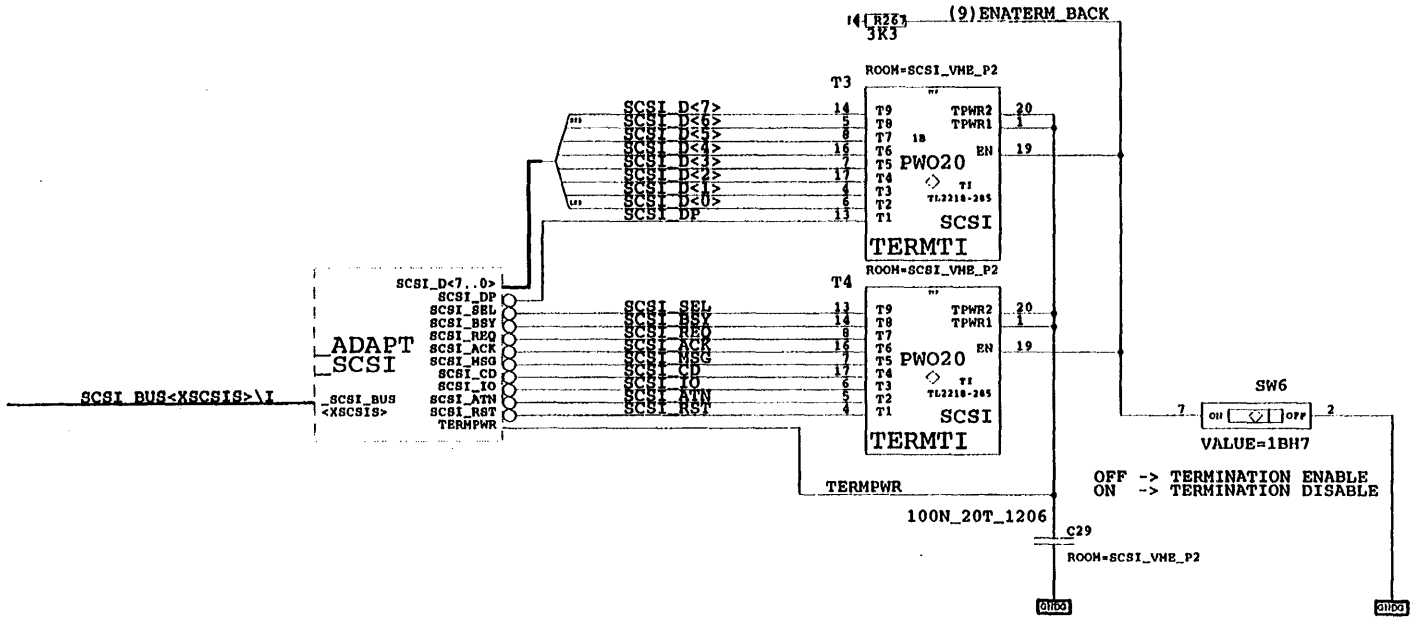
DESIGNER: SK Mon Apr 3 08:55:13 1995	DRAWING ABBREV=MASTER_UNIT APR/25/95 Manual	SHEET 8 OF 94
FORCE COMPUTERS	MASTER_UNIT	REV. 0.2



FLAT_DRAWING: SCSI_PWR_8.LOGIC.1.1

INSTANTIATION: (CPU5V SCSI_PWR24P)

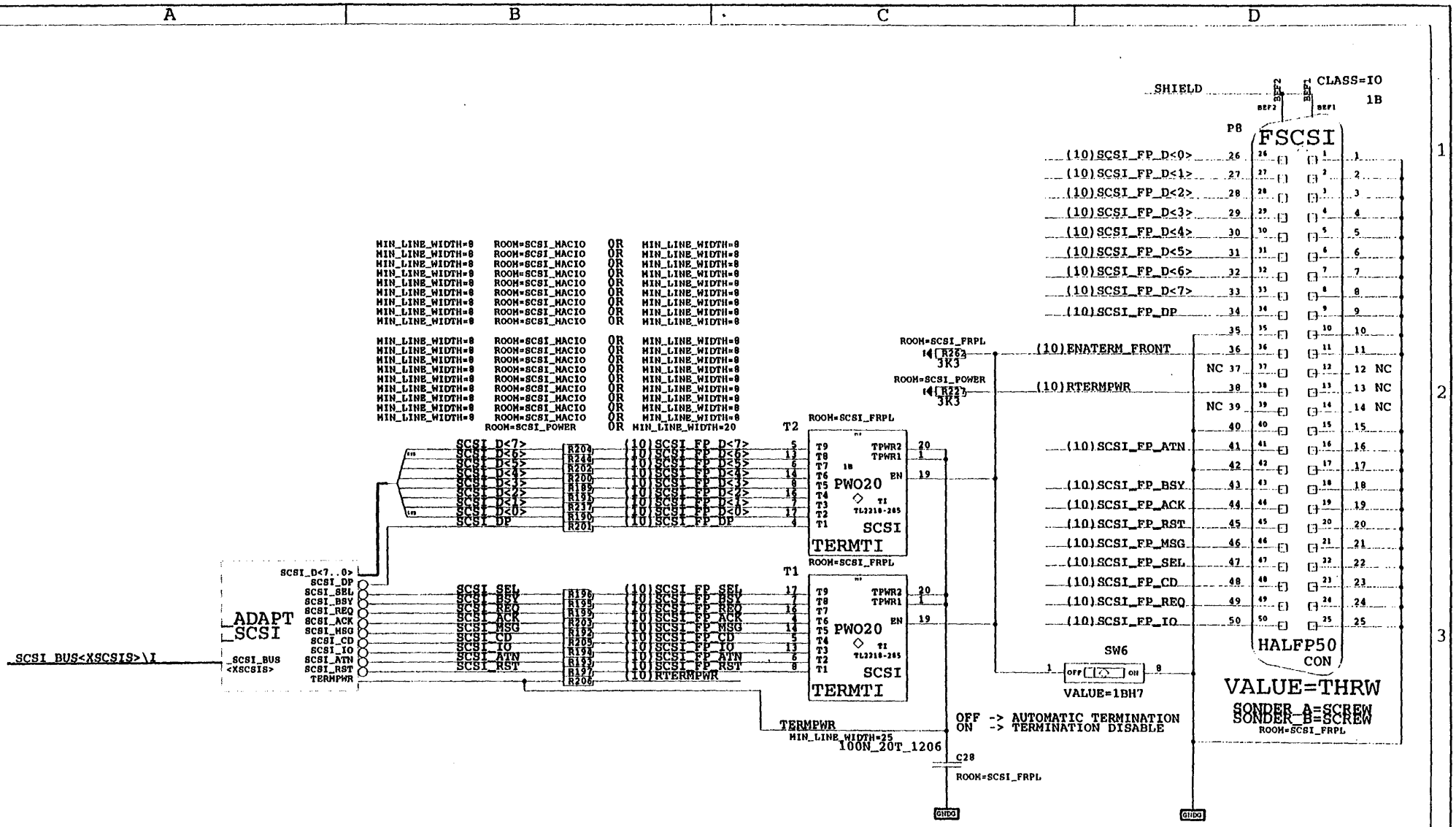
DESIGNER: SK Mon Apr 3 08:55:14 1995	DRAWING ABBREV=SCSI_PWR	SHEET 9 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SCSI_PWR	REV. 0.2



FLAT_DRAWING: SCSI_TERM_9.LOGIC.1.1

INSTANTIATION: (CPU5V SCSI_TERM61P)

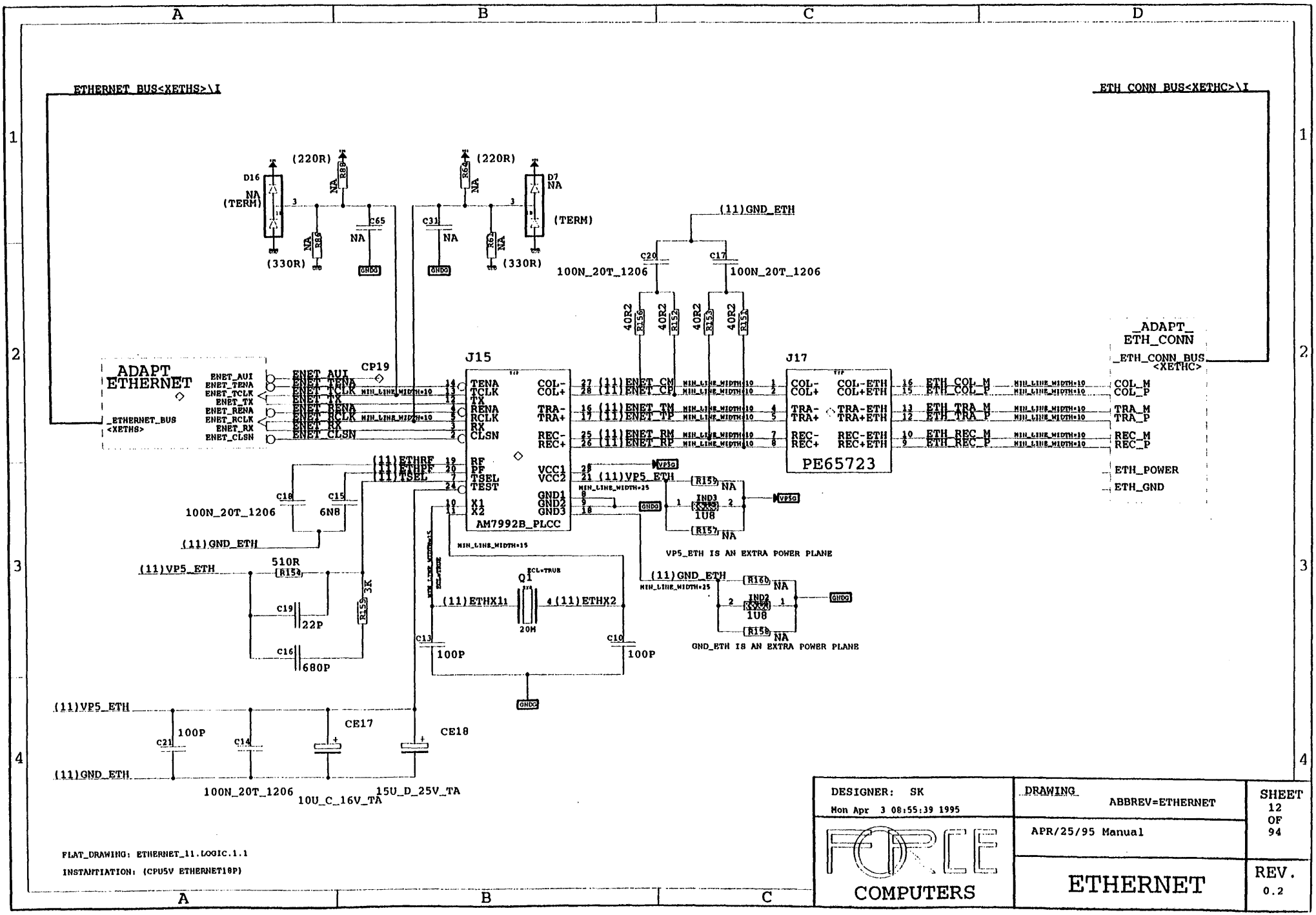
DESIGNER: SK Mon Apr 3 08:55:52 1995	DRAWING ABBREV=SCSI_TERM	SHEET 10 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SCSI_TERM	REV. 0.2

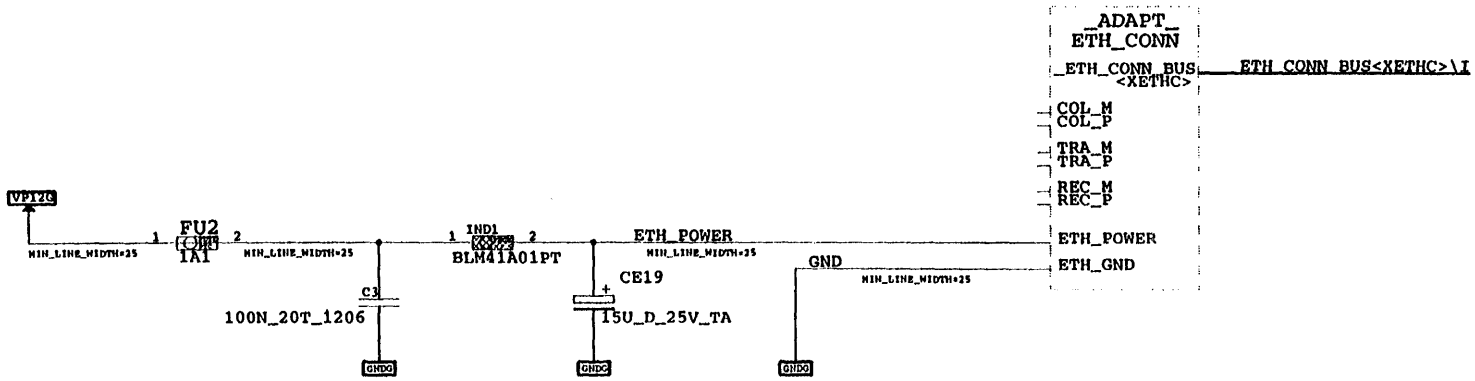


FLAT_DRAWING: SCSI_CONN_10.LOGIC.1.1

INSTANTIATION: (CPU5V SCSI_CONN43P)

DESIGNER: SK Mon Apr 3 08:55:32 1995	DRAWING ABBREV=SCSI_CONN APR/25/95 Manual	SHEET 11 OF 94
FORCE COMPUTERS	SCSI_CONN	REV. 0.2

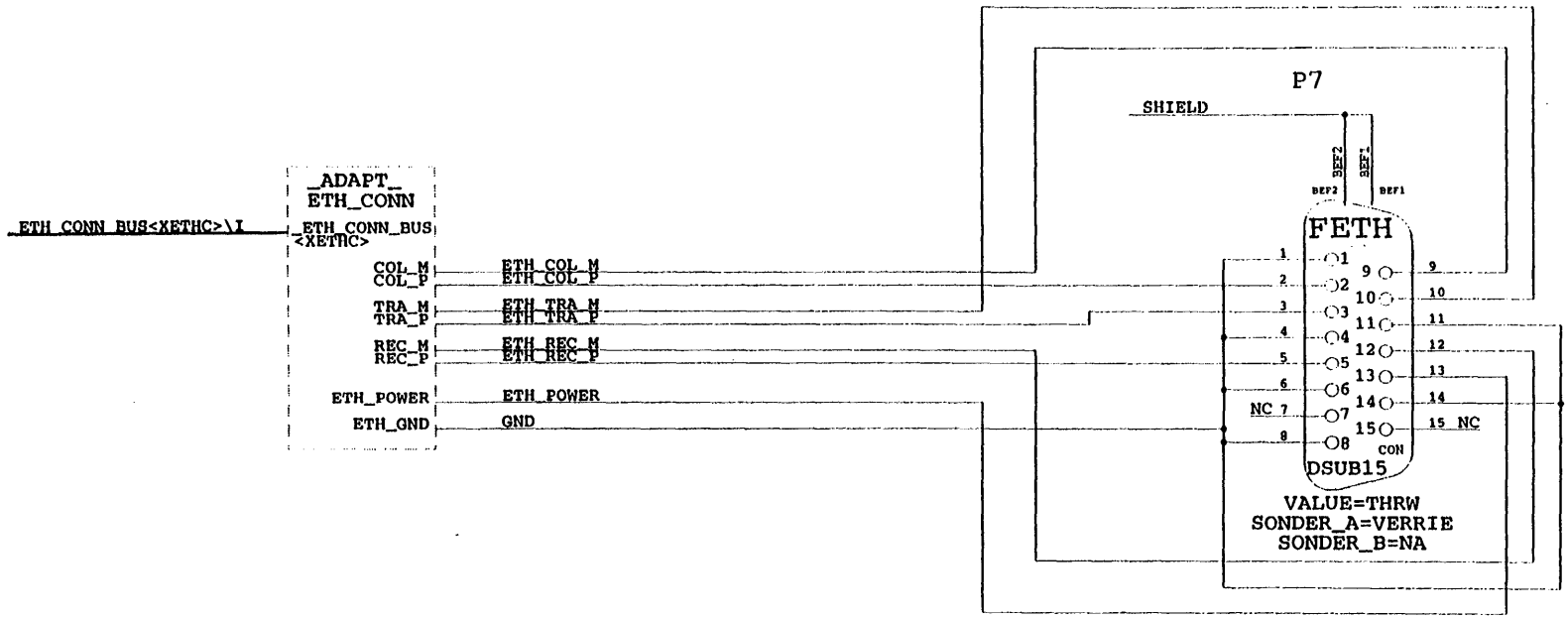

ETHERNET_BUS<XETHS>\I
ETH_CONN_BUS<XETHC>\I
ADAPT ETHERNET
_ETHERNET_BUS <XETHS>
ENET_AUI
ENET_TENA
ENET_TCLK
ENET_TX
ENET_RENA
ENET_RCLK
ENET_RX
ENET_CLSN
CP19
J15
AM7992B_PLCC
TENA
TCLK
TX
RENA
RCLK
RX
CLSN
RF
PF
TSEL
TEST
X1
X2
COL-
COL+
TRA-
TRA+
REC-
REC+
VCC1
VCC2
GND1
GND2
GND3
J17
PE65723
COL-ETH
COL+ETH
TRA-ETH
TRA+ETH
REC-ETH
REC+ETH
ADAPT ETH_CONN
_ETH_CONN_BUS <XETHC>
COL_M
COL_P
TRA_M
TRA_P
REC_M
REC_P
ETH_POWER
ETH_GND
(11)ENET_CM
(11)ENET_CP
(11)ENET_TM
(11)ENET_TP
(11)ENET_RM
(11)ENET_RP
(11)ETHRF
(11)ETHPF
(11)TSEL
(11)VP5_ETH
(11)GND_ETH
(11)ETHX1
(11)ETHX2
D16
D7
(TERM)
NA
(220R)
(330R)
R88
R86
R64
R65
C65
C31
C20
C17
100N_20T_1206
40R2
R156
R152
R153
R157
C18
C15
6N8
C19
22P
C16
680P
510R
R158
3K
R159
Q1
ECL-TRUE
20M
C13
100P
C10
100P
C21
C14
CE17
CE18
15U_D_25V_TA
10U_C_16V_TA
IND3
IND2
1U8
R150
R151
R160
R161
VP5_ETH IS AN EXTRA POWER PLANE
GND_ETH IS AN EXTRA POWER PLANE
MIN_LINE_WIDTH=10
MIN_LINE_WIDTH=15
MIN_LINE_WIDTH=25
FLAT_DRAWING: ETHERNET_11.LOGIC.1.1
INSTANTIATION: (CPU5V ETHERNET18P)
DESIGNER: SK
Mon Apr 3 08:55:39 1995
DRAWING
ABBREV=ETHERNET
APR/25/95 Manual
COMPUTERS
ETHERNET
SHEET 12 OF 94
REV. 0.2



PLAT_DRAWING: ETHERNET_POWER_12.LOGIC.1.1

INSTANTIATION: (CPU5V ETHERNET_POWER20P)

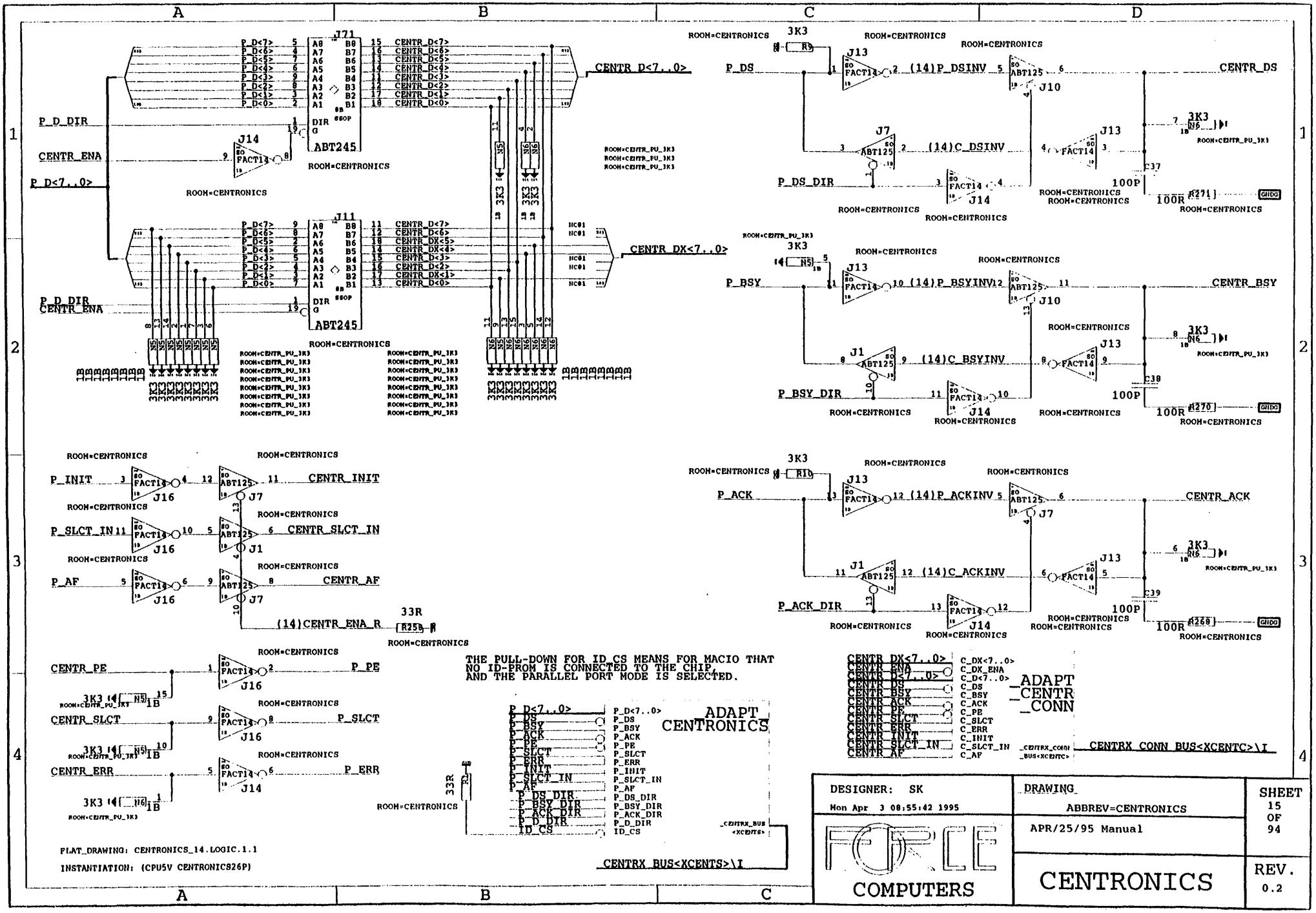
DESIGNER: SK Mon Apr 3 08:54:40 1995	DRAWING ABBREV=ETHERNET_POWER	SHEET 13 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	ETHERNET_POWER	REV. 0.2

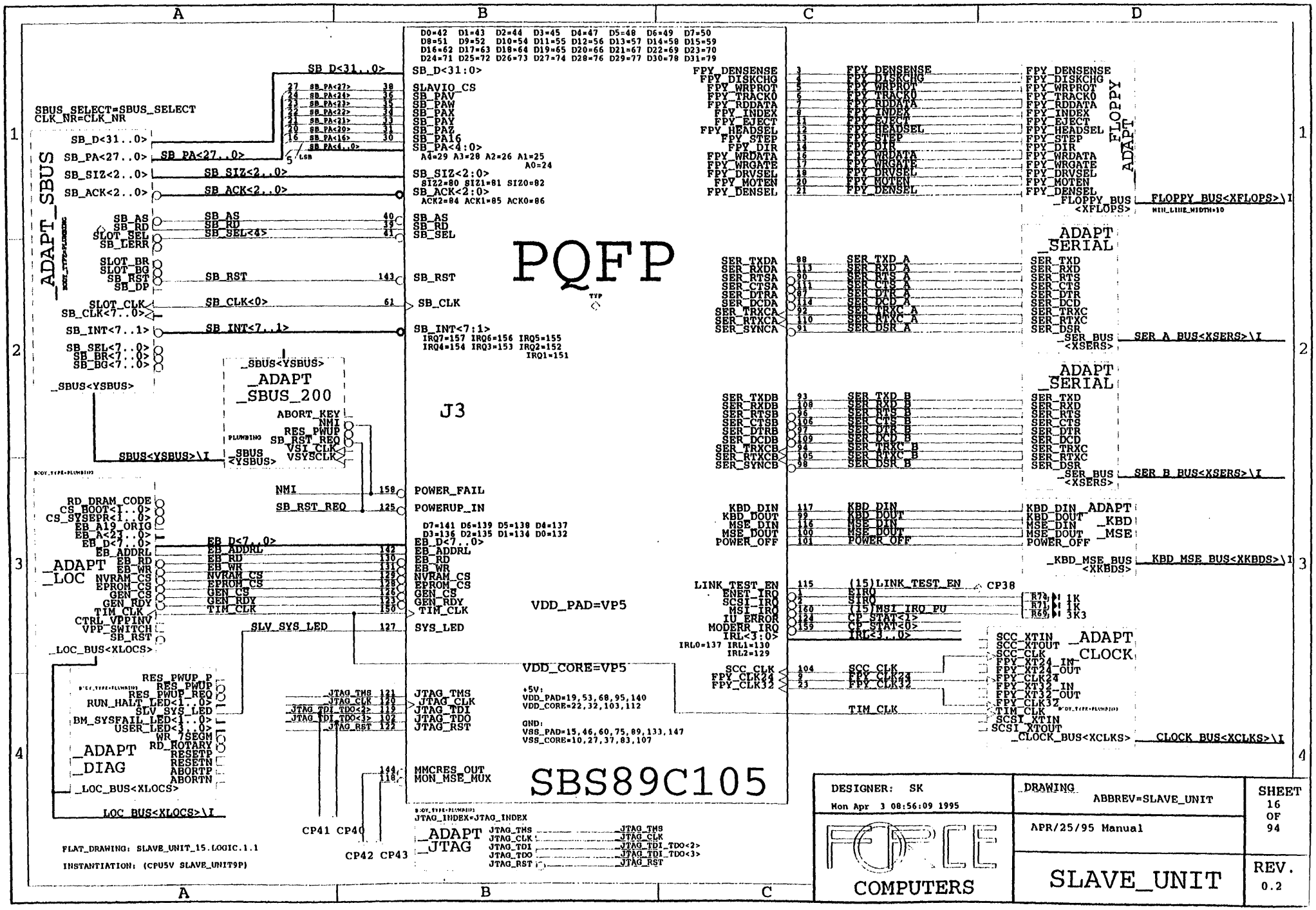


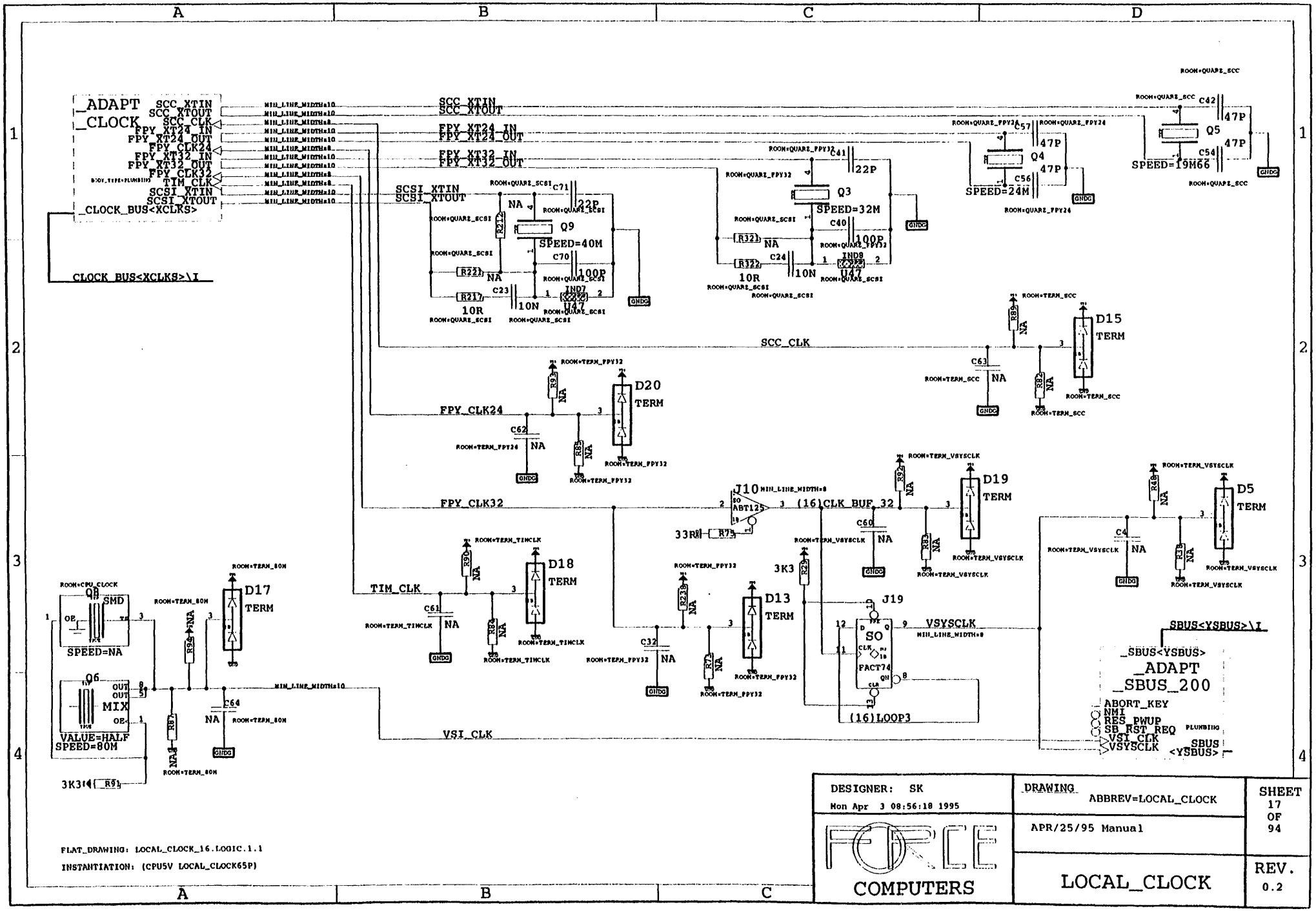
FLAT_DRAWING: ETH_CONN_STD_13.LOGIC.1.1

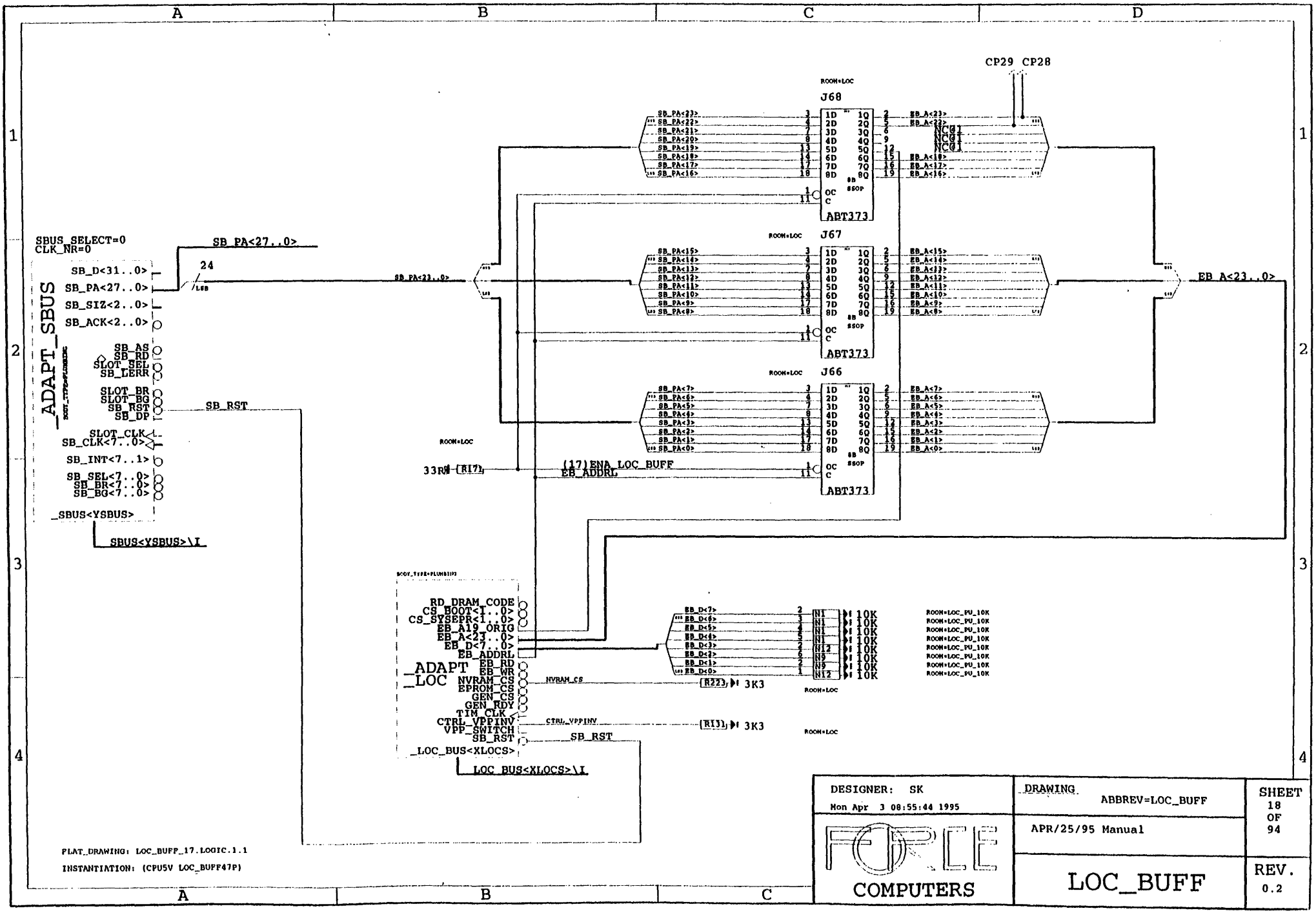
INSTANTIATION: (CPU5V ETH_CONN_STD19P)

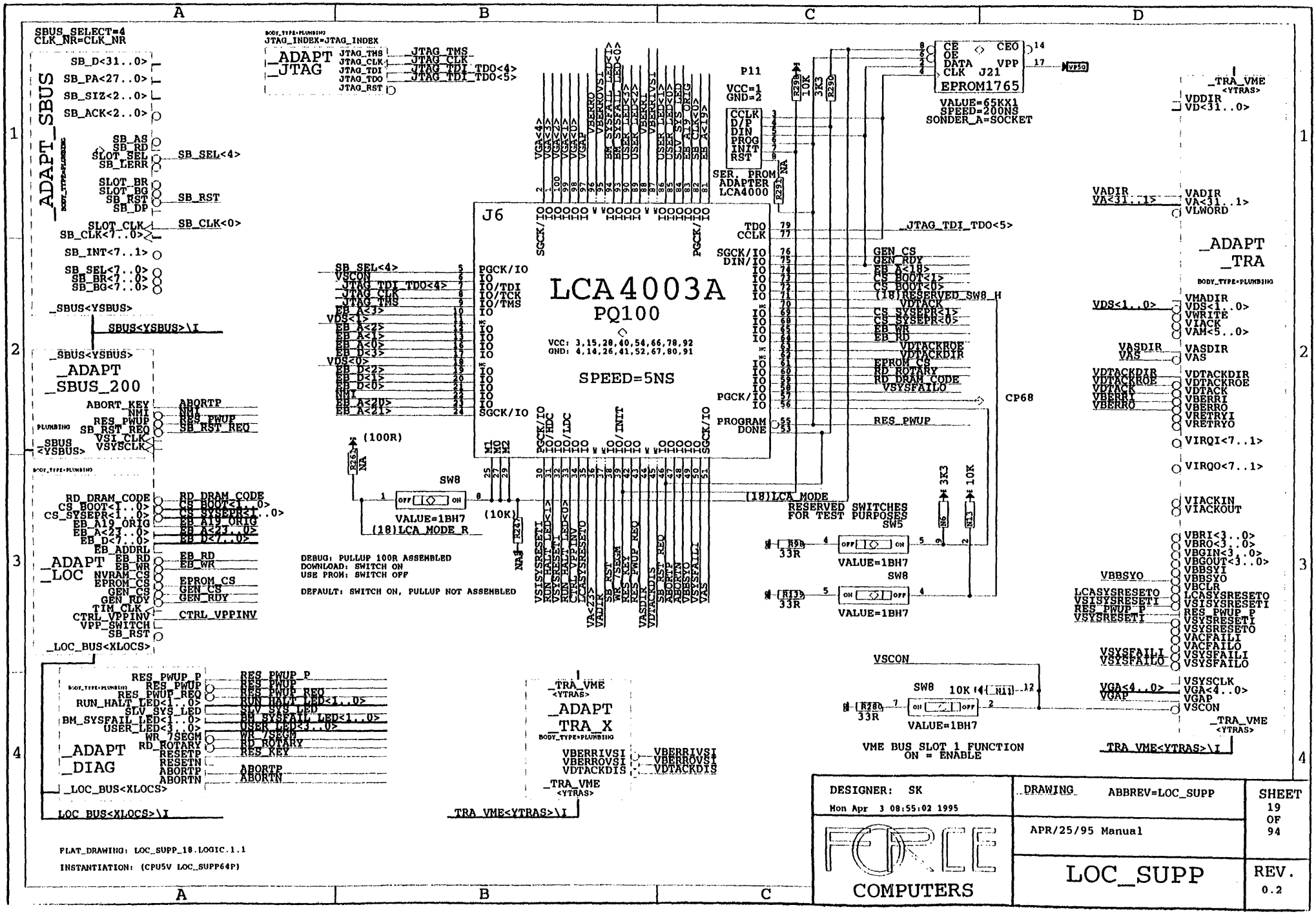
DESIGNER: SK Mon Apr 3 08:55:18 1995	DRAWING ABBREV=ETH_CONN_STD APR/25/95 Manual	SHEET 14 OF 94
FORCE COMPUTERS	ETH_CONN_STD	REV. 0.2

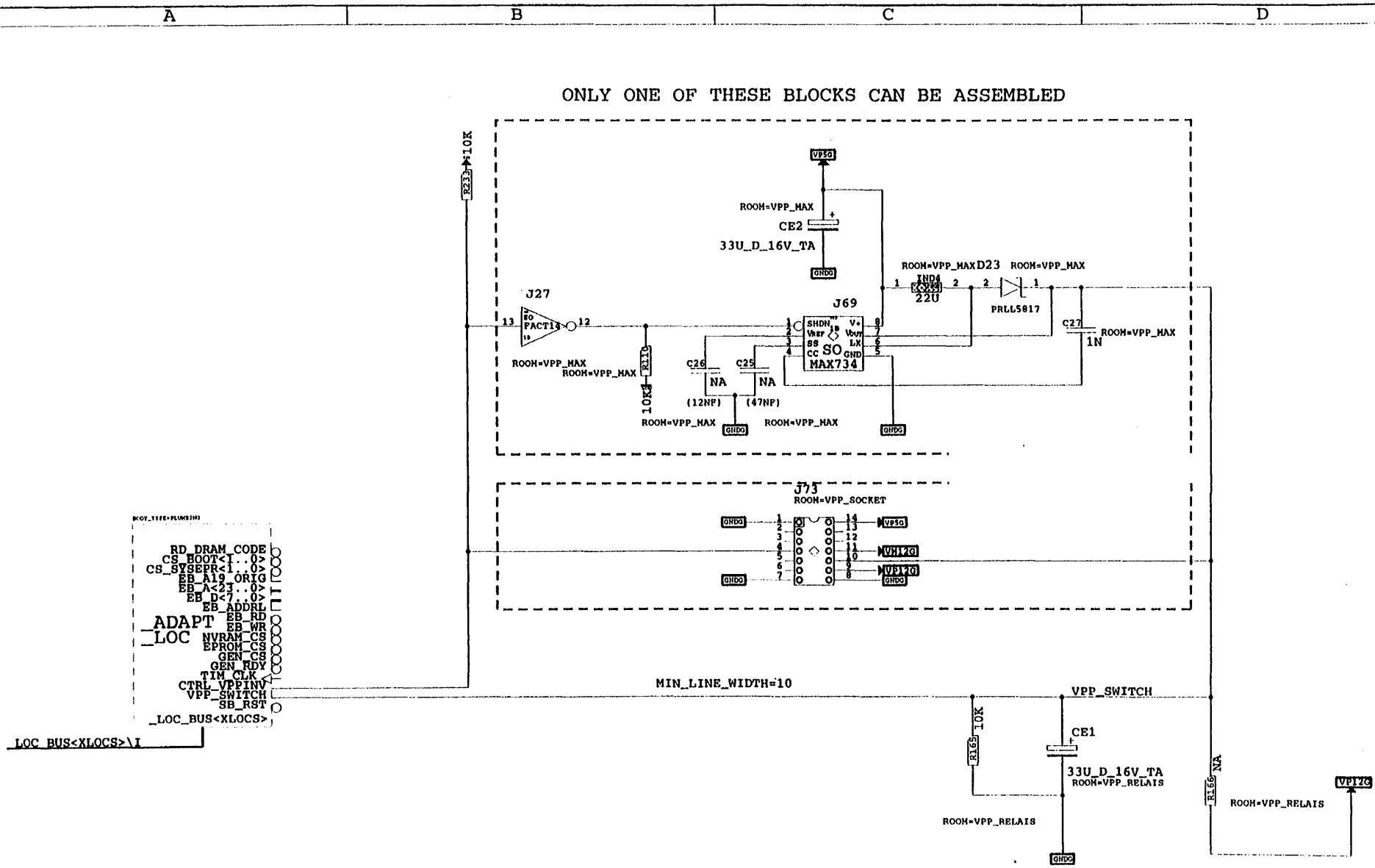

A
B
C
D
P_D<7>
P_D<6>
P_D<5>
P_D<4>
P_D<3>
P_D<2>
P_D<1>
P_D<0>
J71
ABT245
P_D_DIR
CENTR_ENA
P_D<7..0>
CENTR_D<7..0>
CENTR_D<7>
CENTR_D<6>
CENTR_D<5>
CENTR_D<4>
CENTR_D<3>
CENTR_D<2>
CENTR_D<1>
CENTR_D<0>
J14
FACT14
ROOM=CENTRONICS
J11
CENTR_DX<7..0>
CENTR_DX<6>
CENTR_DX<5>
CENTR_DX<4>
CENTR_DX<3>
CENTR_DX<2>
CENTR_DX<1>
ROOM=CENTR_PU_3K3
3K3
P_DS
CENTR_DS
J13
J10
J7
ABT125
(14)P_DSINV
(14)C_DSINV
P_DS_DIR
100P
100R
P_BSY
CENTR_BSY
(14)P_BSYINV
(14)C_BSYINV
P_BSY_DIR
J1
P_INIT
CENTR_INIT
J16
P_SLCT_IN
CENTR_SLCT_IN
P_AF
CENTR_AF
(14)CENTR_ENA_R
33R
P_ACK
CENTR_ACK
(14)P_ACKINV
(14)C_ACKINV
P_ACK_DIR
CENTR_PE
P_PE
CENTR_SLCT
P_SLCT
CENTR_ERR
P_ERR
THE PULL-DOWN FOR ID_CS MEANS FOR MACIO THAT
NO ID-PROM IS CONNECTED TO THE CHIP
AND THE PARALLEL PORT MODE IS SELECTED.
ADAPT
CENTRONICS
P_D<7..0>
P_DS
P_BSY
P_ACK
P_PE
P_SLCT
P_ERR
P_INIT
P_SLCT_IN
P_AF
P_DS_DIR
P_BSY_DIR
P_ACK_DIR
P_D_DIR
ID_CS
CENTRX_BUS<XCENTS>
CENTRX_BUS<XCENTS>\I
C_DX<7..0>
C_DX_ENA
C_D<7..0>
C_DS
C_BSY
C_ACK
C_PE
C_SLCT
C_ERR
C_INIT
C_SLCT_IN
C_AF
_ADAPT
_CENTR
_CONN
CENTRX_CONN_BUS<XCENTC>\I
PLAT_DRAWING: CENTRONICS_14.LOGIC.1.1
INSTANTIATION: (CPU5V CENTRONICS26P)
DESIGNER: SK
Mon Apr 3 08:55:42 1995
DRAWING
ABBREV=CENTRONICS
APR/25/95 Manual
SHEET 15 OF 94
REV. 0.2
FORCE COMPUTERS
CENTRONICS

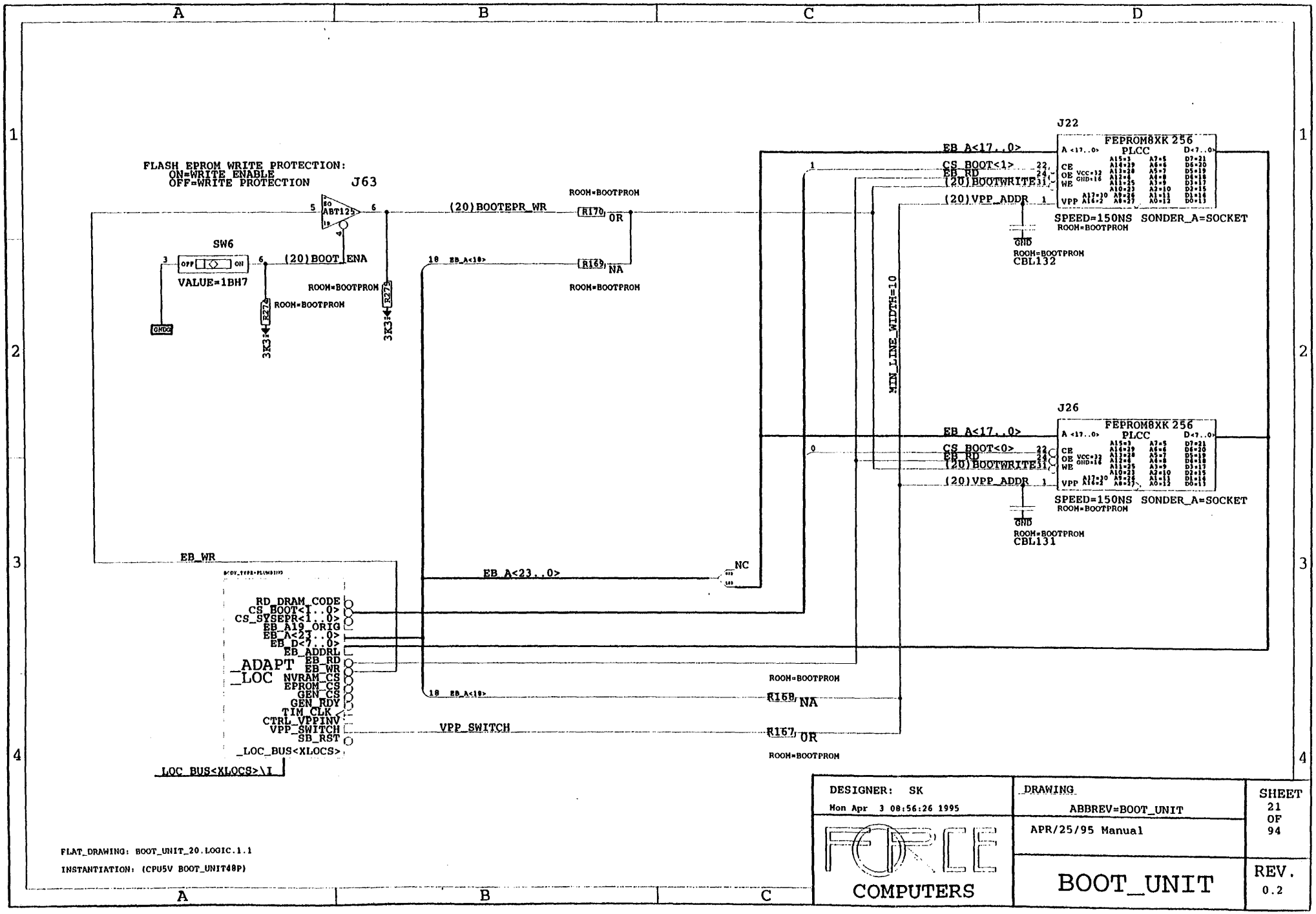

A
B
C
D
_ADAPT_SBUS
SBUS_SELECT=SBUS_SELECT
CLK_NR=CLK_NR
SB_D<31..0>
SB_PA<27..0>
SB_SIZ<2..0>
SB_ACK<2..0>
SB_AS
SB_RD
SLOT_SEL
SB_LERR
SLOT_BR
SLOT_BG
SB_RST
SB_DP
SLOT_CLK
SB_CLK<7..0>
SB_INT<7..1>
SB_SEL<7..0>
SB_BR<7..0>
SB_BG<7..0>
_SBUS<YSBUS>
SBUS<YSBUS>\I
_ADAPT
_SBUS_200
ABORT_KEY
NMI
RES_PWUP
SB_RST_REQ
VSI_CLK
VSYSCLK
PLUMBING
SBUS
<YSBUS>
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<27..0>
EB_D<7..0>
EB_ADDRL
EB_RD
EB_WR
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_ADAPT
_LOC
_LOC_BUS<XLOCS>
SLV_SYS_LED
RES_PWUP_P
RES_PWUP
RES_PWUP_REQ
RUN_HALT_LED<1..0>
SLV_SYS_LED
BM_SYSFAIL_LED<1..0>
USER_LED<3..0>
WR_7SEGM
RD_ROTARY
RESETP
RESETN
ABORTP
ABORTN
_ADAPT
_DIAG
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
FLAT_DRAWING: SLAVE_UNIT_15.LOGIC.1.1
INSTANTIATION: (CPU5V SLAVE_UNIT9P)
CP41 CP40
CP42 CP43
JTAG_TMS
JTAG_CLK
JTAG_TDI
JTAG_TDO
JTAG_RST
JTAG_TDI_TDO<2>
JTAG_TDI_TDO<3>
MMCRES_OUT
MON_MSE_MUX
JTAG_INDEX=JTAG_INDEX
_ADAPT
_JTAG
D0=42 D1=43 D2=44 D3=45 D4=47 D5=48 D6=49 D7=50
D8=51 D9=52 D10=54 D11=55 D12=56 D13=57 D14=58 D15=59
D16=62 D17=63 D18=64 D19=65 D20=66 D21=67 D22=69 D23=70
D24=71 D25=72 D26=73 D27=74 D28=76 D29=77 D30=78 D31=79
SLAVIO_CS
SB_PAV
SB_PAW
SB_PAX
SB_PAY
SB_PAZ
SB_PA16
A0=29 A1=28 A2=26 A3=25
A0=24
SIZ2=80 SIZ1=81 SIZ0=82
ACK2=84 ACK1=85 ACK0=86
IRQ7=157 IRQ6=156 IRQ5=155
IRQ4=154 IRQ3=153 IRQ2=152
IRQ1=151
PQFP
TYP
J3
POWER_FAIL
POWERUP_IN
D7=141 D6=139 D5=138 D4=137
D3=136 D2=135 D1=134 D0=132
IRL0=137 IRL1=130
IRL2=129
VDD_PAD=VP5
VDD_CORE=VP5
+5V:
VDD_PAD=19,53,68,95,140
VDD_CORE=22,32,103,112
GND:
VSS_PAD=15,46,60,75,89,133,147
VSS_CORE=10,27,37,83,107
SBS89C105
FPY_DENSENSE
FPY_DISKCHG
FPY_WRPROT
FPY_TRACK0
FPY_RDDATA
FPY_INDEX
FPY_EJECT
FPY_HEADSEL
FPY_STEP
FPY_DIR
FPY_WRDATA
FPY_WRGATE
FPY_DRVSEL
FPY_MOTEN
FPY_DENSEL
FLOPPY_BUS<XFLOPS>
FLOPPY_BUS<XFLOPS>\I
ADAPT
_FLOPPY
SER_TXDA
SER_RXDA
SER_RTSA
SER_CTSA
SER_DTRA
SER_DCDA
SER_TRXCA
SER_RTXCA
SER_SYNCA
SER_TXD_A
SER_RXD_A
SER_RTS_A
SER_CTS_A
SER_DTR_A
SER_DCD_A
SER_TRXC_A
SER_RTXC_A
SER_DSR_A
SER_BUS<XSERS>
SER_A_BUS<XSERS>\I
ADAPT
_SERIAL
SER_TXDB
SER_RXDB
SER_RTSB
SER_CTSB
SER_DTRB
SER_DCDB
SER_TRXCB
SER_RTXCB
SER_SYNCB
SER_TXD_B
SER_RXD_B
SER_RTS_B
SER_CTS_B
SER_DTR_B
SER_DCD_B
SER_TRXC_B
SER_RTXC_B
SER_DSR_B
SER_B_BUS<XSERS>\I
KBD_DIN
KBD_DOUT
MSE_DIN
MSE_DOUT
POWER_OFF
_KBD_MSE_BUS<XKBDS>
_KBD_MSE_BUS<XKBDS>\I
ADAPT
_KBD
_MSE
LINK_TEST_EN
ENET_IRQ
SCSI_IRQ
MSI_IRQ
IU_ERROR
MODERR_IRQ
IRL<3:0>
IRL<3..0>
(15)LINK_TEST_EN
(15)MSI_IRQ_PU
CP38
R70 1K
R71 1K
R69 3K3
CP_STAT<1>
CP_STAT<0>
SCC_CLK
FPY_CLK24
FPY_CLK32
TIM_CLK
SCC_XTIN
SCC_XTOUT
SCC_CLK
FPY_XT24_IN
FPY_XT24_OUT
FPY_CLK24
FPY_XT32_IN
FPY_XT32_OUT
FPY_CLK32
TIM_CLK
SCSI_XTIN
SCSI_XTOUT
_CLOCK_BUS<XCLKS>
CLOCK_BUS<XCLKS>\I
ADAPT
CLOCK
DESIGNER: SK
Mon Apr 3 08:56:09 1995
DRAWING
ABBREV=SLAVE_UNIT
APR/25/95 Manual
SHEET
16
OF
94
FORCE
COMPUTERS
SLAVE_UNIT
REV.
0.2


_ADAPT
_CLOCK
SCC_XTIN
SCC_XTOUT
SCC_CLK
FPY_XT24_IN
FPY_XT24_OUT
FPY_CLK24
FPY_XT32_IN
FPY_XT32_OUT
FPY_CLK32
TIM_CLK
SCSI_XTIN
SCSI_XTOUT
_CLOCK_BUS<XCLKS>
CLOCK_BUS<XCLKS>\I
SPEED=40M
SPEED=32M
SPEED=24M
SPEED=19M66
SPEED=NA
SPEED=80M
VALUE=HALF
MIX
SMD
TERM
D15
D20
D18
D17
D13
D19
D5
J10
ABT125
(16)CLK_BUF_32
(16)LOOP3
FACT74
VSYSCLK
SCC_CLK
FPY_CLK24
FPY_CLK32
TIM_CLK
VSI_CLK
_SBUS<YSBUS>
_ADAPT
_SBUS_200
SBUS<YSBUS>\I
ABORT_KEY
NMI
RES_PWUP
SB_RST_REQ
VSI_CLK
VSYSCLK
SBUS
<YSBUS>
10R
10N
22P
100P
47P
3K3
33R
NA
GNDG
FLAT_DRAWING: LOCAL_CLOCK_16.LOGIC.1.1
INSTANTIATION: (CPU5V LOCAL_CLOCK65P)
DESIGNER: SK
Mon Apr 3 08:56:18 1995
DRAWING
ABBREV=LOCAL_CLOCK
APR/25/95 Manual
SHEET
17
OF
94
COMPUTERS
LOCAL_CLOCK
REV.
0.2


A
B
C
D
1
2
3
4
SBUS_SELECT=0
CLK_NR=0
_ADAPT_SBUS
SB_D<31..0>
SB_PA<27..0>
SB_SIZ<2..0>
SB_ACK<2..0>
SB_AS
SB_RD
SLOT_SEL
SB_LERR
SLOT_BR
SLOT_BG
SB_RST
SB_DP
SLOT_CLK
SB_CLK<7..0>
SB_INT<7..1>
SB_SEL<7..0>
SB_BR<7..0>
SB_BG<7..0>
_SBUS<YSBUS>
SBUS<YSBUS>\I
SB_PA<27..0>
24
LSB
SB_PA<23..0>
SB_RST
ROOM=LOC
J68
J67
J66
ABT373
SB SSOP
CP29 CP28
NC@1
EB_A<23..0>
33R
(R17)
(17)ENA_LOC_BUFF
EB_ADDRL
BODY_TYPE=PLUMB1P3
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<23..0>
EB_D<7..0>
EB_ADDRL
EB_RD
EB_WR
_ADAPT
_LOC
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
EB_D<7..0>
10K
ROOM=LOC_PU_10K
(R22)
3K3
(R13)
3K3
PLAT_DRAWING: LOC_BUFF_17.LOGIC.1.1
INSTANTIATION: (CPU5V LOC_BUFF47P)
DESIGNER: SK
Mon Apr 3 08:55:44 1995
DRAWING
ABBREV=LOC_BUFF
APR/25/95 Manual
SHEET
18
OF
94
FORCE
COMPUTERS
LOC_BUFF
REV.
0.2


SBUS_SELECT=4
CLK_NR=CLK_NR
_ADAPT_SBUS
SB_D<31..0>
SB_PA<27..0>
SB_SIZ<2..0>
SB_ACK<2..0>
SB_AS
SB_DS
SLOT_SEL
SB_LERR
SB_SEL<4>
SLOT_BR
SLOT_BG
SB_RST
SB_RST
SB_DP
SLOT_CLK
SB_CLK<0>
SB_CLK<7..0>
SB_INT<7..1>
SB_SEL<7..0>
SB_BR<7..0>
SB_BG<7..0>
_SBUS<YSBUS>
SBUS<YSBUS>\I
_ADAPT
_SBUS_200
ABORT_KEY
NMI
RES_PWUP
SB_RST_REQ
VSI_CLK
VSYSCLK
ABORTP
NMI
RES_PWUP
SB_RST_REQ
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<23..0>
EB_D<7..0>
EB_ADDRL
EB_RD
EB_WR
EPROM_CS
GEN_CS
GEN_RDY
CTRL_VPPINV
_ADAPT
_LOC
NVRAM_CS
TIM_CLK
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
JTAG_INDEX=JTAG_INDEX
_ADAPT
_JTAG
JTAG_THS
JTAG_CLK
JTAG_TDI
JTAG_TDO
JTAG_RST
JTAG_TMS
JTAG_CLK
JTAG_TDI_TDO<4>
JTAG_TDI_TDO<5>
P11
VCC=1
GND=2
CCLK
D/P
DIN
PROG
INIT
RST
SER. PROM
ADAPTER
LCA4000
R293
NA
R239
10K
3K3
R236
CE
OE
DATA
CLK
CEO
VPP
J21
EPROM1765
VALUE=65KX1
SPEED=200NS
SONDER_A=SOCKET
VP5Q
J6
LCA4003A
PQ100
VCC: 3,15,28,40,54,66,78,92
GND: 4,14,26,41,52,67,80,91
SPEED=5NS
SB_SEL<4>
VSCON
JTAG_TDI_TDO<4>
JTAG_CLK
JTAG_TMS
EB_A<3>
VDS<1>
EB_A<2>
EB_A<1>
EB_A<0>
EB_D<3>
VDS<0>
EB_D<2>
EB_D<1>
EB_D<0>
NMI
EB_A<20>
EB_A<21>
PGCK/IO
IO
IO/TDI
IO/TCK
IO/TMS
IO
SGCK/IO
TDO
CCLK
SGCK/IO
DIN/IO
PGCK/IO
PROGRAM
DONE
_JTAG_TDI_TDO<5>
GEN_CS
GEN_RDY
EB_A<18>
CS_BOOT<1>
CS_BOOT<0>
(18)RESERVED_SW8_H
VDTACK
CS_SYSEPR<1>
CS_SYSEPR<0>
EB_WR
EB_RD
VDTACKROE
VDTACKDIR
EPROM_CS
RD_ROTARY
RD_DRAM_CODE
VSYSFAILO
RES_PWUP
CP68
(100R)
R66
NA
SW8
VALUE=1BH7
(18)LCA_MODE_R
(10K)
R247
NA
(18)LCA_MODE
RESERVED SWITCHES
FOR TEST PURPOSES
SW5
3K3
10K
R59
33R
VALUE=1BH7
SW8
R13
33R
VALUE=1BH7
DEBUG: PULLUP 100R ASSEMBLED
DOWNLOAD: SWITCH ON
USE PROM: SWITCH OFF
DEFAULT: SWITCH ON, PULLUP NOT ASSEMBLED
_TRA_VME
<YTRAS>
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
_ADAPT
_TRA
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VRETRYO
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRO<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
LCASYSRESETO
VSISYSRESETI
RES_PWUP_P
VSYSRESETI
VSYSRESETO
VACFAILI
VACFAILO
VSYSFAILI
VSYSFAILO
VSYSCLK
VGA<4..0>
VGAP
VSCON
_TRA_VME
<YTRAS>
TRA_VME<YTRAS>\I
VSCON
SW8
10K
R11
R280
33R
VALUE=1BH7
VME BUS SLOT 1 FUNCTION
ON = ENABLE
_ADAPT
_DIAG
RES_PWUP_P
RES_PWUP
RES_PWUP_REQ
RUN_HALT_LED<1..0>
SLV_SYS_LED
BM_SYSFAIL_LED<1..0>
USER_LED<3..0>
WR_/SEGM
RD_ROTARY
RESETP
RESETN
ABORTP
ABORTN
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
_TRA_VME
<YTRAS>
_ADAPT
_TRA_X
VBERRIVSI
VBERROVSI
VDTACKDIS
_TRA_VME
<YTRAS>
TRA_VME<YTRAS>\I
FLAT_DRAWING: LOC_SUPP_18.LOGIC.1.1
INSTANTIATION: (CPU5V LOC_SUPP64P)
DESIGNER: SK
Mon Apr 3 08:55:02 1995
DRAWING ABBREV=LOC_SUPP
APR/25/95 Manual
LOC_SUPP
SHEET 19 OF 94
REV. 0.2
FORCE COMPUTERS

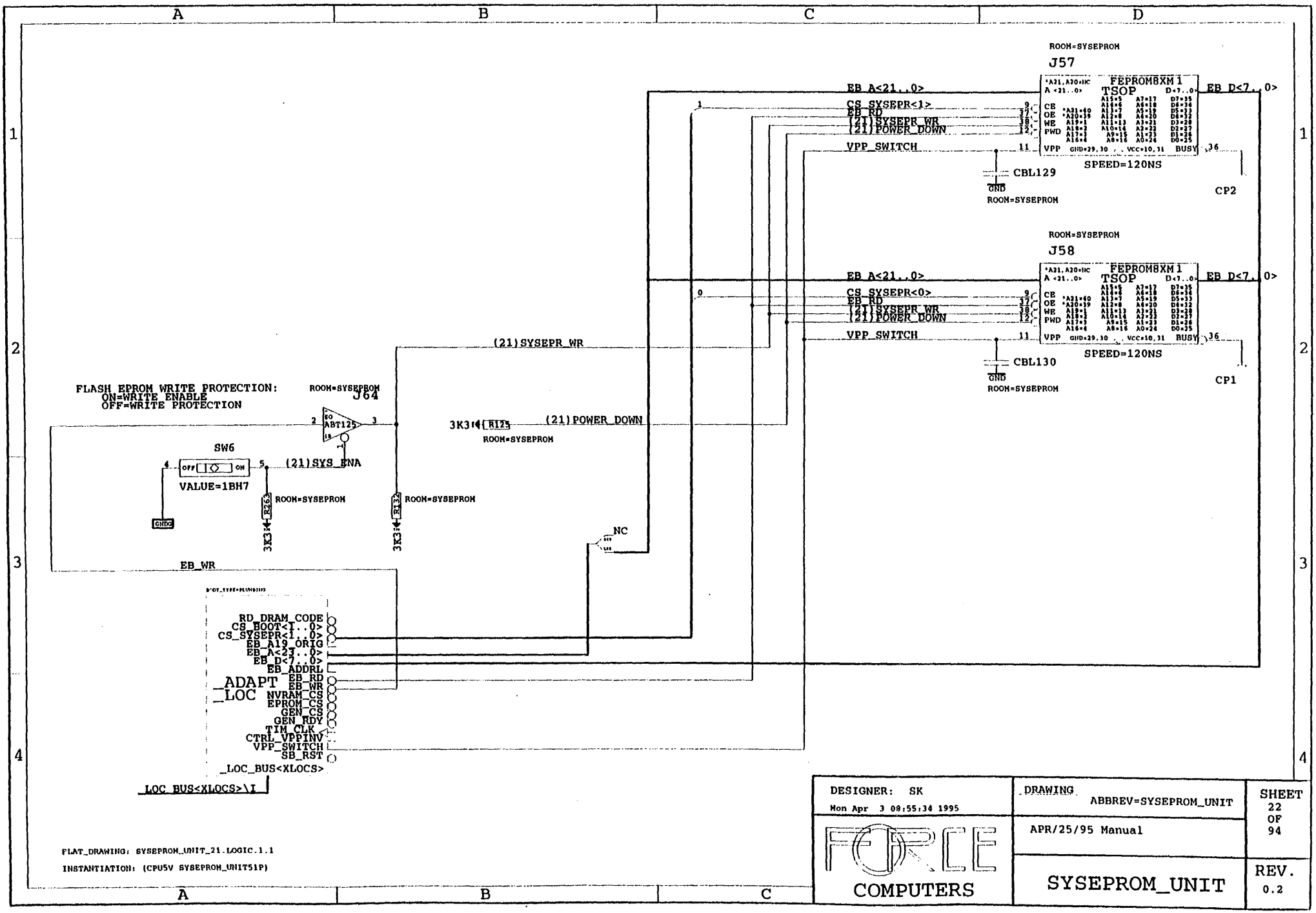

ONLY ONE OF THESE BLOCKS CAN BE ASSEMBLED
R233
10K
VP5G
ROOM=VPP_MAX
CE2
33U_D_16V_TA
GNDG
J27
FACT14
J69
MAX734
SHDN
Vref
SS
CC
SO
V+
Vout
LX
GND
D23
PRLL5817
IND4
220U
C27
1N
C26
NA
(12NP)
C25
NA
(47NP)
R110
10K
J73
ROOM=VPP_SOCKET
VH12G
VP12G
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<23..0>
EB_D<7..0>
EB_ADDRL
EB_RD
EB_WR
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
_ADAPT
_LOC
LOC_BUS<XLOCS>\I
MIN_LINE_WIDTH=10
VPP_SWITCH
R165
10K
CE1
33U_D_16V_TA
ROOM=VPP_RELAIS
R166
NA
VP12G
ROOM=VPP_RELAIS
PLAT_DRAWING: VPP_UNIT_19.LOGIC.1.1
INSTANTIATION: (CPU5V VPP_UNIT53P)
DESIGNER: SK
Mon Apr 3 08:55:50 1995
COMPUTERS
DRAWING
ABBREV=VPP_UNIT
APR/25/95 Manual
VPP_UNIT
SHEET 20 OF 94
REV. 0.2

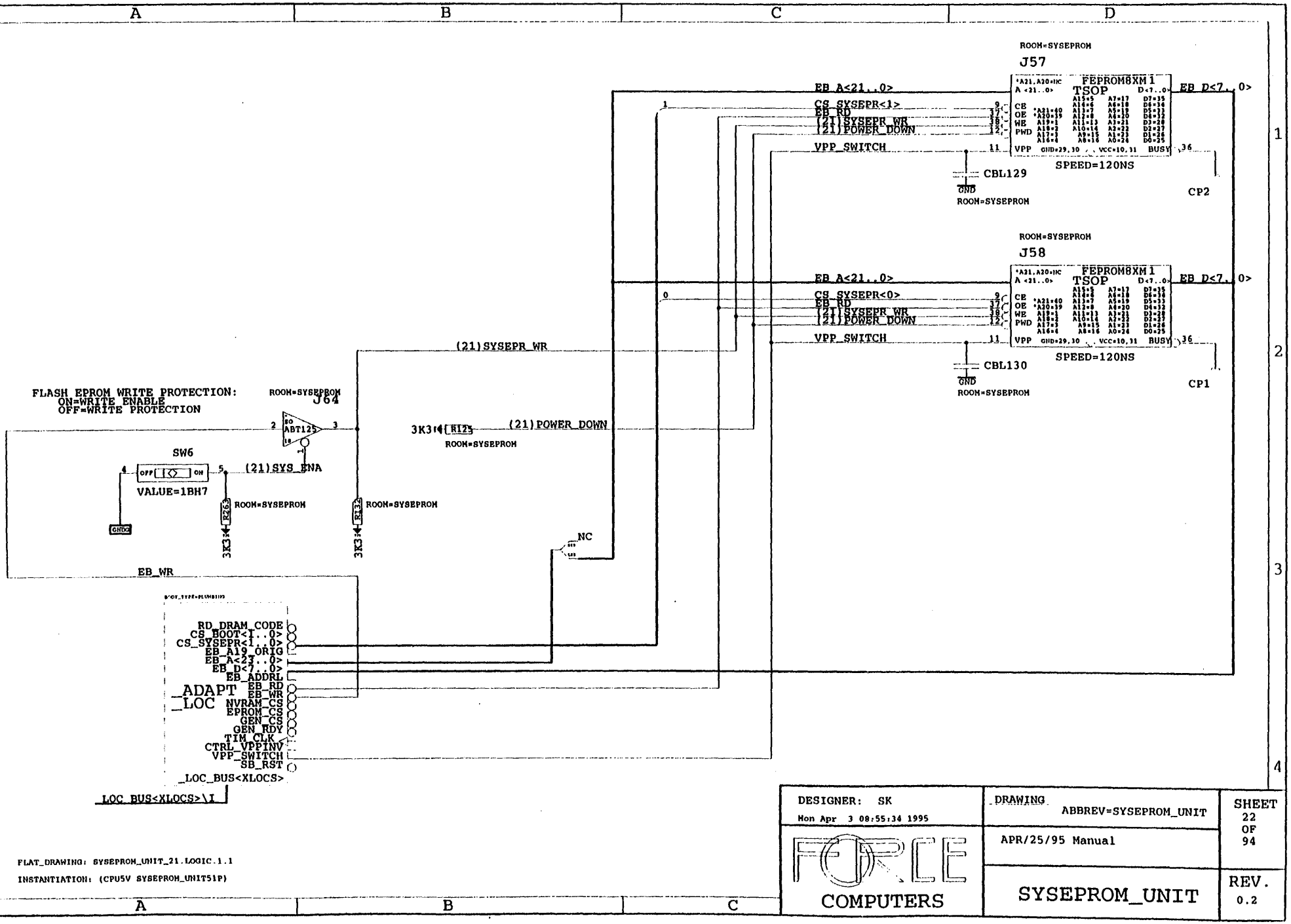


FLAT_DRAWING: BOOT_UNIT_20.LOGIC.1.1

INSTANTIATION: (CPU5V BOOT_UNIT48P)

DESIGNER: SK Mon Apr 3 08:56:26 1995	DRAWING ABBREV=BOOT_UNIT APR/25/95 Manual	SHEET 21 OF 94
FORCE COMPUTERS	BOOT_UNIT	REV. 0.2


ROOM=SYSEPROM
J57
EB A<21..0>
CS_SYSEPR<1>
EB_RD
(21)SYSEPR_WR
(21)POWER_DOWN
VPP_SWITCH
FEPROM8XM1
TSOP
EB D<7..0>
SPEED=120NS
CBL129
GND
CP2
J58
CS_SYSEPR<0>
CBL130
CP1
FLASH EPROM WRITE PROTECTION:
ON=WRITE ENABLE
OFF=WRITE PROTECTION
J64
ABT125
SW6
VALUE=1BH7
(21)SYS_ENA
R25
R133
3K3
NC
EB_WR
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<23..0>
EB_D<7..0>
EB_ADDRL
_ADAPT_LOC
EB_RD
EB_WR
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
FLAT_DRAWING: SYSEPROM_UNIT_21.LOGIC.1.1
INSTANTIATION: (CPU5V SYSEPROM_UNIT51P)
DESIGNER: SK
Mon Apr 3 08:55:34 1995
DRAWING
ABBREV=SYSEPROM_UNIT
APR/25/95 Manual
COMPUTERS
SYSEPROM_UNIT
SHEET 22 OF 94
REV. 0.2

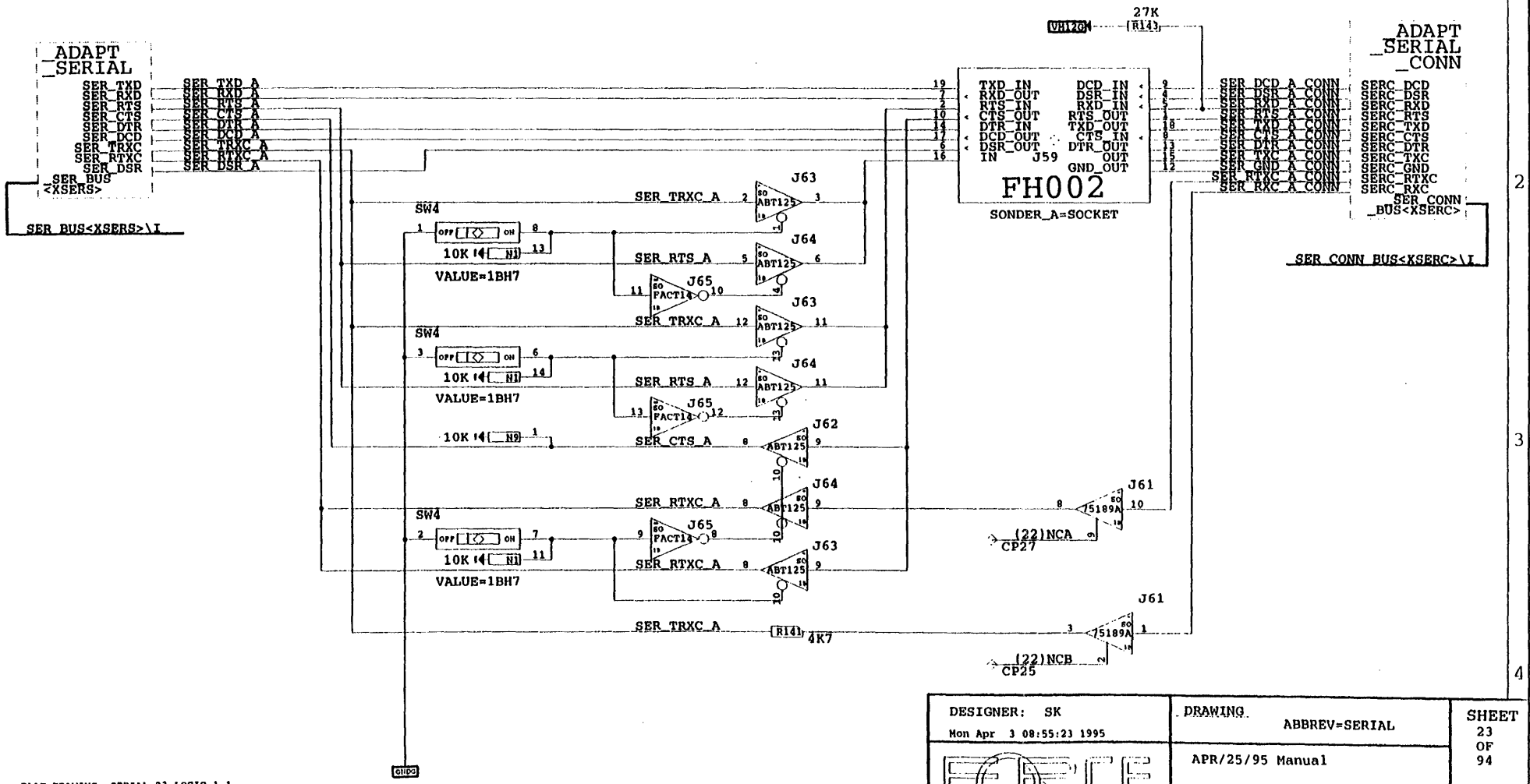

ROOM=SYSEPROM
J57
EB A<21..0>
CS_SYSEPR<1>
EB_RD
(21)SYSEPR_WR
(21)POWER_DOWN
VPP_SWITCH
FEPROM8XM1
TSOP
EB D<7..0>
SPEED=120NS
CBL129
GND
CP2
ROOM=SYSEPROM
J58
CS_SYSEPR<0>
CBL130
CP1
FLASH EPROM WRITE PROTECTION:
ON=WRITE ENABLE
OFF=WRITE PROTECTION
J64
SO ABT125
3K3 R125
SW6
VALUE=1BH7
(21)SYS_ENA
R26
R135
NC
EB_WR
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<21..0>
EB_D<7..0>
EB_ADDRL
_ADAPT_LOC
EB_RD
EB_WR
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
FLAT_DRAWING: SYSEPROM_UNIT_21.LOGIC.1.1
INSTANTIATION: (CPU5V SYSEPROM_UNIT51P)
DESIGNER: SK
Mon Apr 3 08:55:34 1995
DRAWING
ABBREV=SYSEPROM_UNIT
APR/25/95 Manual
SHEET 22 OF 94
COMPUTERS
SYSEPROM_UNIT
REV. 0.2

PORT A:

RS232 SCC	RS232 CONNECTOR	RS232 PIN #	RS422 SCC	RS422 CONNECTOR	RS422 PIN #	RS485 SCC	RS485 CONNECTOR	RS485 PIN #
TXDA	TXD	2	TXDA	TXD+	24	TXDA/	RXTX+	7
RXDA	RXD	3		TXD-	8	RXDA	RXTX-	20
RTSA	RTS	4	RTSA/	RTS+/TRXC+	4	REN=DTRA		
CTSA	CTS	5	TRXCA	RTS-/TRXC-	1	TEN=RTSA		
DTRA	DTR	20	CTSA/	CTS+/RTXC+	2			
DCDA	DCD	8	RTXCA	CTS-/RTXC-	5			
TRXCA	TXC/RXC	24/15	RXDA	RXD+	20			
RTXCA	RTXC	17		RXD-	7			
DSRA	DSR	6						

PORT B:

RS232 SCC	RS232 CONNECTOR	RS232 PIN #	RS422 SCC	RS422 CONNECTOR	RS422 PIN #	RS485 SCC	RS485 CONNECTOR	RS485 PIN #
TXDB	TXD	14	TXDB	TXD+	25	TXDB/	RXTX+	23
RXDB	RXD	16		TXD-	12	RXDB	RXTX-	11
RTSB	RTS	19	RTSB/	RTS+/TRXC+	19	REN=DTRB		
CTSB	CTS	13	TRXCB	RTS-/TRXC-	16	TEN=RTSB		
DTRB	DTR	11	CTSB/	CTS+/RTXC+	14			
DCDB	DCD	12	RTXCB	CTS-/RTXC-	11			
TRXCB	TXC/RXC	25/10	RXDB	RXD+	11			
RTXCB	RTXC	22		RXD-	23			
DSRB	DSR	21						



FLAT_DRAWING: SERIAL_23.LOGIC.1.1

INSTANTIATION: (CPU5V SERIAL45P)

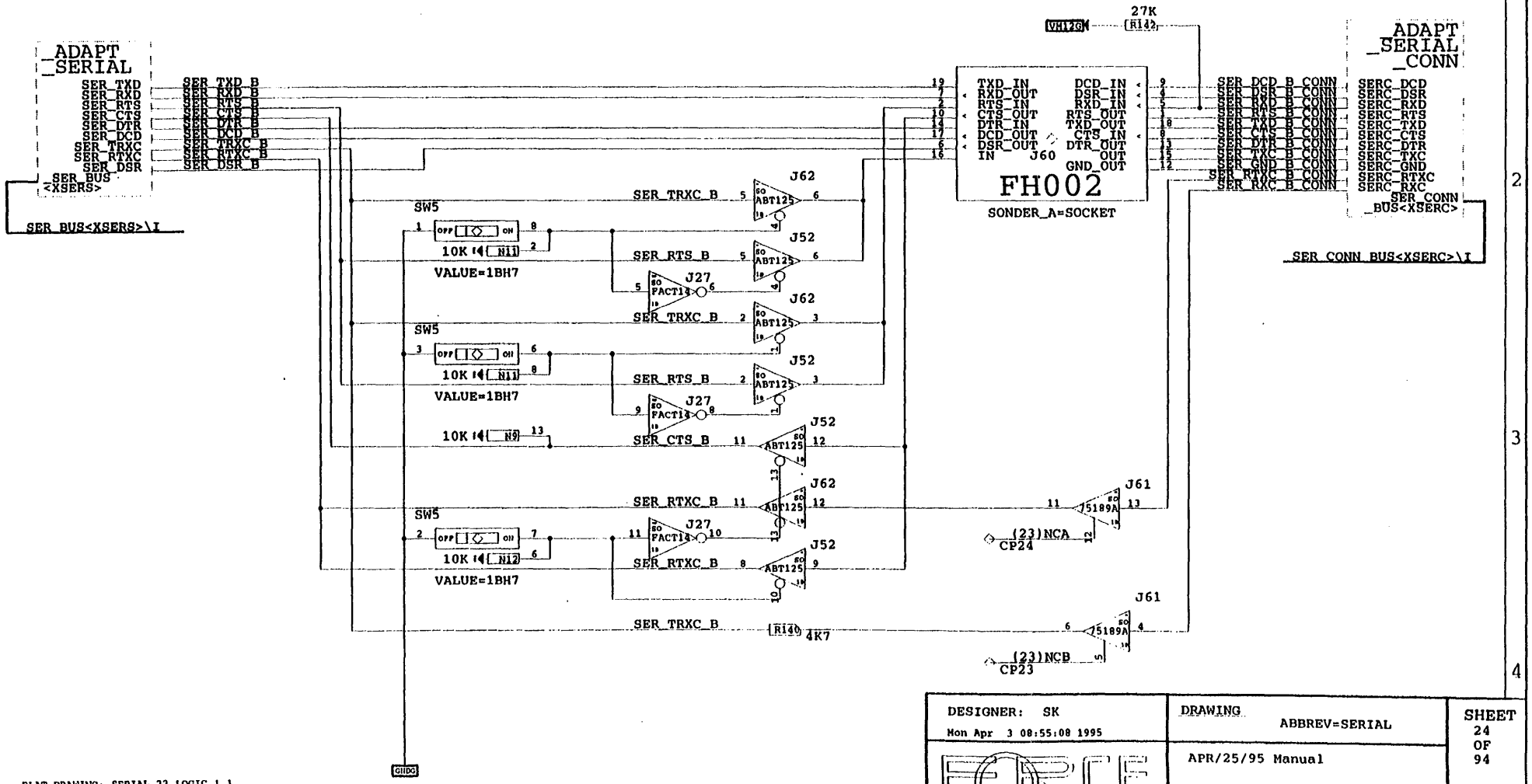
DESIGNER: SK Mon Apr 3 08:55:23 1995	DRAWING ABBREV=SERIAL	SHEET 23 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SERIAL	REV. 0.2

PORT A:

RS232 SCC	RS232 CONNECTOR	RS232 PIN #	RS422 SCC	RS422 CONNECTOR	RS422 PIN #	RS485 SCC	RS485 CONNECTOR	RS485 PIN #
TXDA	TXD	2	TXDA	TXD+	24	TXDA/	RXTX+	7
RXDA	RXD	3		TXD-	8	RXDA	RXTX-	20
RTSA	RTS	4	RTSA/	RTS+/TRXC+	4	REN=DTRA		
CTSA	CTS	5	TRXCA	RTS-/TRXC-	3	TEN=RTSA		
DTRA	DTR	20	CTSA/	CTS+/RTXC+	5			
DCDA	DCD	8	RTXCA	CTS-/RTXC-	22			
TRXCA	TXC/RXC	24/15	RXDA	RXD+	20			
RTXCA	RTXC	17		RXD-	7			
DSRA	DSR	6						

PORT B:

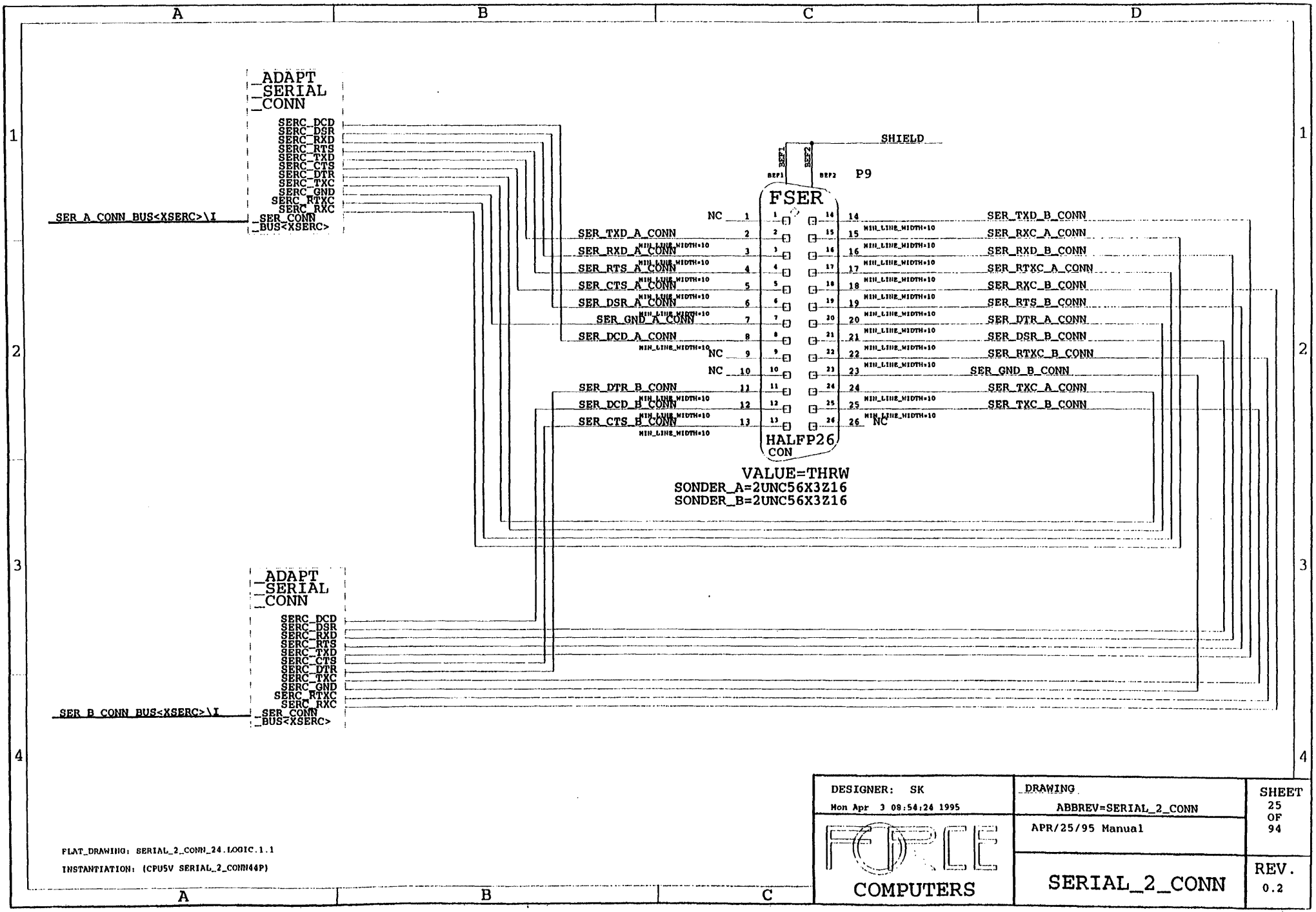
RS232 SCC	RS232 CONNECTOR	RS232 PIN #	RS422 SCC	RS422 CONNECTOR	RS422 PIN #	RS485 SCC	RS485 CONNECTOR	RS485 PIN #
TXDB	TXD	14	TXDB	TXD+	25	TXDB/	RXTX+	23
RXDB	RXD	16		TXD-	12	RXDB	RXTX-	11
RTSB	RTS	19	RTSB/	RTS+/TRXC+	19	REN=DTRB		
CTSB	CTS	13	TRXCB	RTS-/TRXC-	16	TEN=RTSB		
DTRB	DTR	11	CTSB/	CTS+/RTXC+	14			
DCDB	DCD	12	RTXCB	CTS-/RTXC-	13			
TRXCB	TXC/RXC	25/18	RXDB	RXD+	11			
RTXCB	RTXC	22		RXD-	23			
DSRB	DSR	21						



FLAT_DRAWING: SERIAL_22.LOGIC.1.1

INSTANTIATION: (CPU5V SERIAL30P)

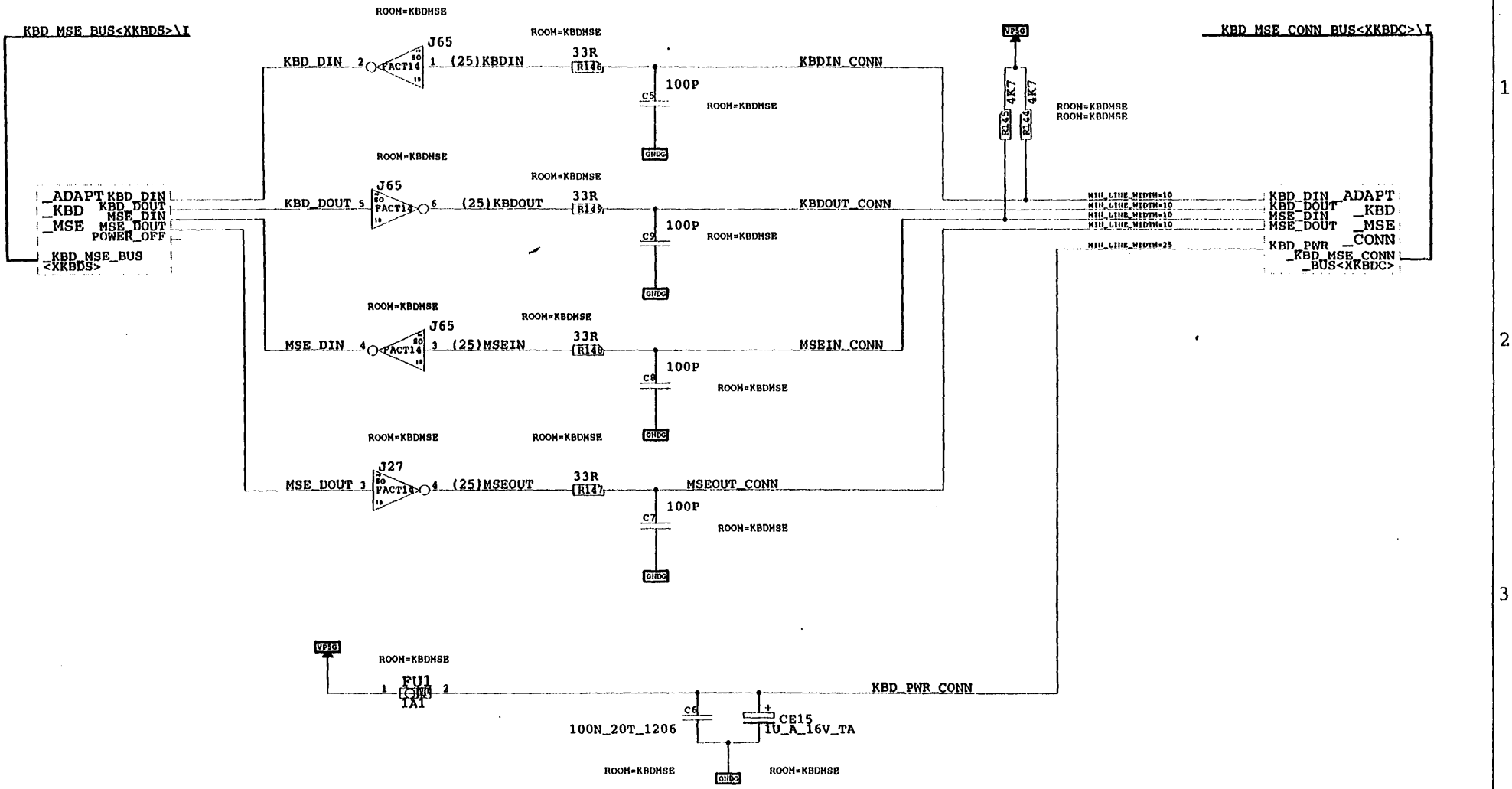
DESIGNER: SK Mon Apr 3 08:55:08 1995	DRAWING ABBREV=SERIAL	SHEET 24 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SERIAL	REV. 0.2



FLAT_DRAWING: SERIAL_2_CONN_24.LOGIC.1.1

INSTANTIATION: (CPU5V SERIAL_2_CONN44P)

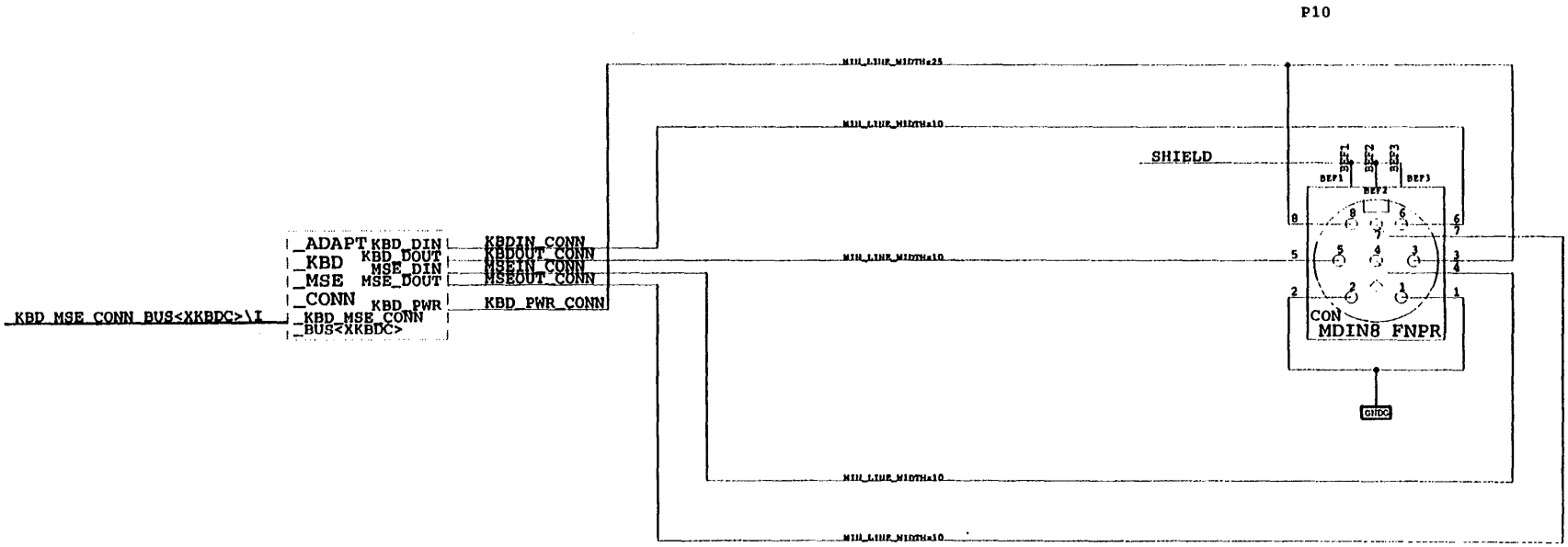
DESIGNER: SK Mon Apr 3 08:54:24 1995	DRAWING ABBREV=SERIAL_2_CONN	SHEET 25 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	SERIAL_2_CONN	REV. 0.2



FLAT_DRAWING: KBD_MSE_25.LOGIC.1.1

INSTANTIATION: (CPU5V KBD_MSE31P)

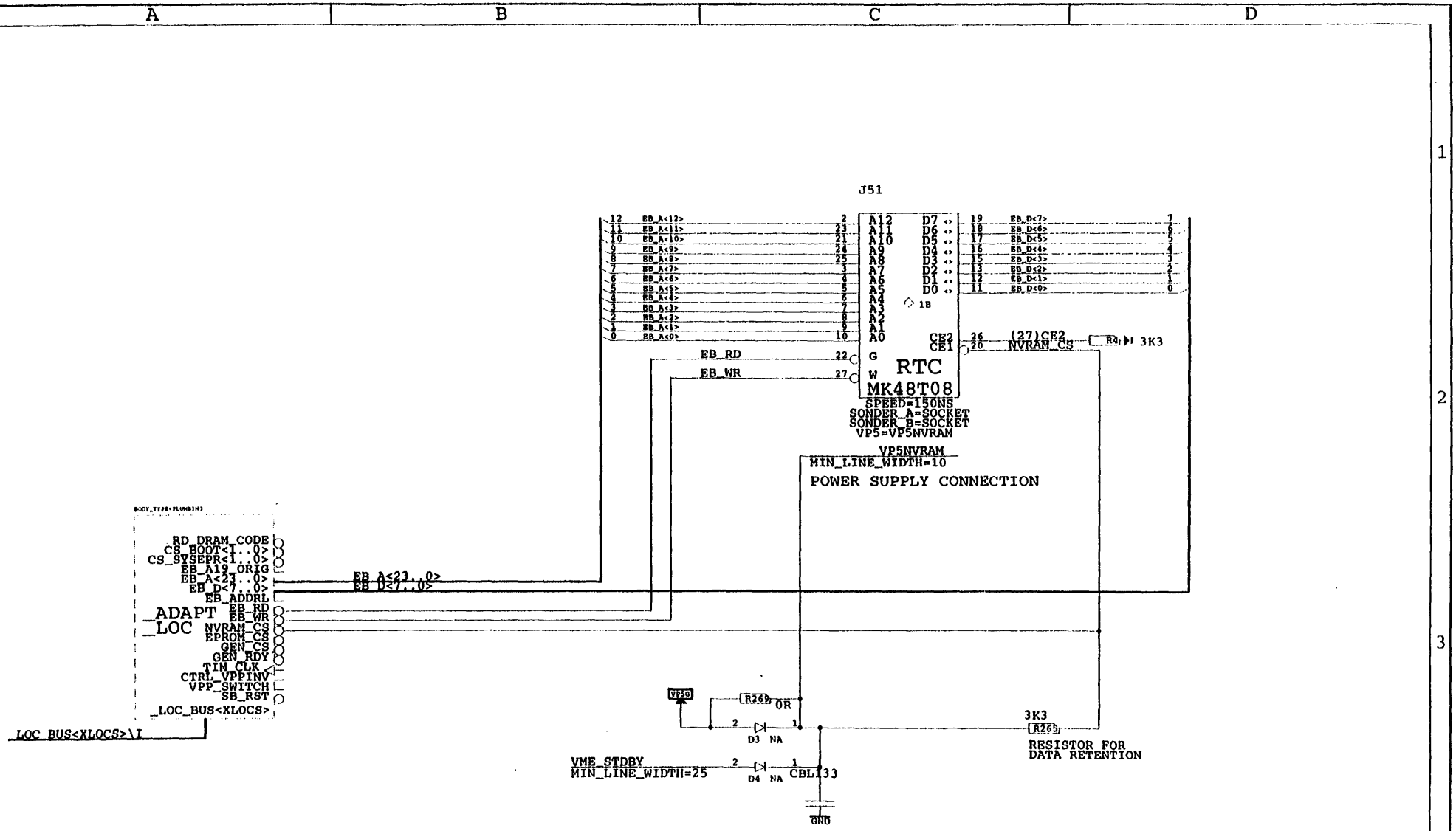
DESIGNER: SK Mon Apr 3 08:54:21 1995	DRAWING ABBREV=KBD_MSE	SHEET 26 OF 94
COMPUTERS	APR/25/95 Manual	
	KBD_MSE	REV. 0.2



FLAT_DRAWING: KBD_MSE_CONN_26.LOGIC.1.1

INSTANTIATION: (CPU5V KBD_MSE_CONN41P)

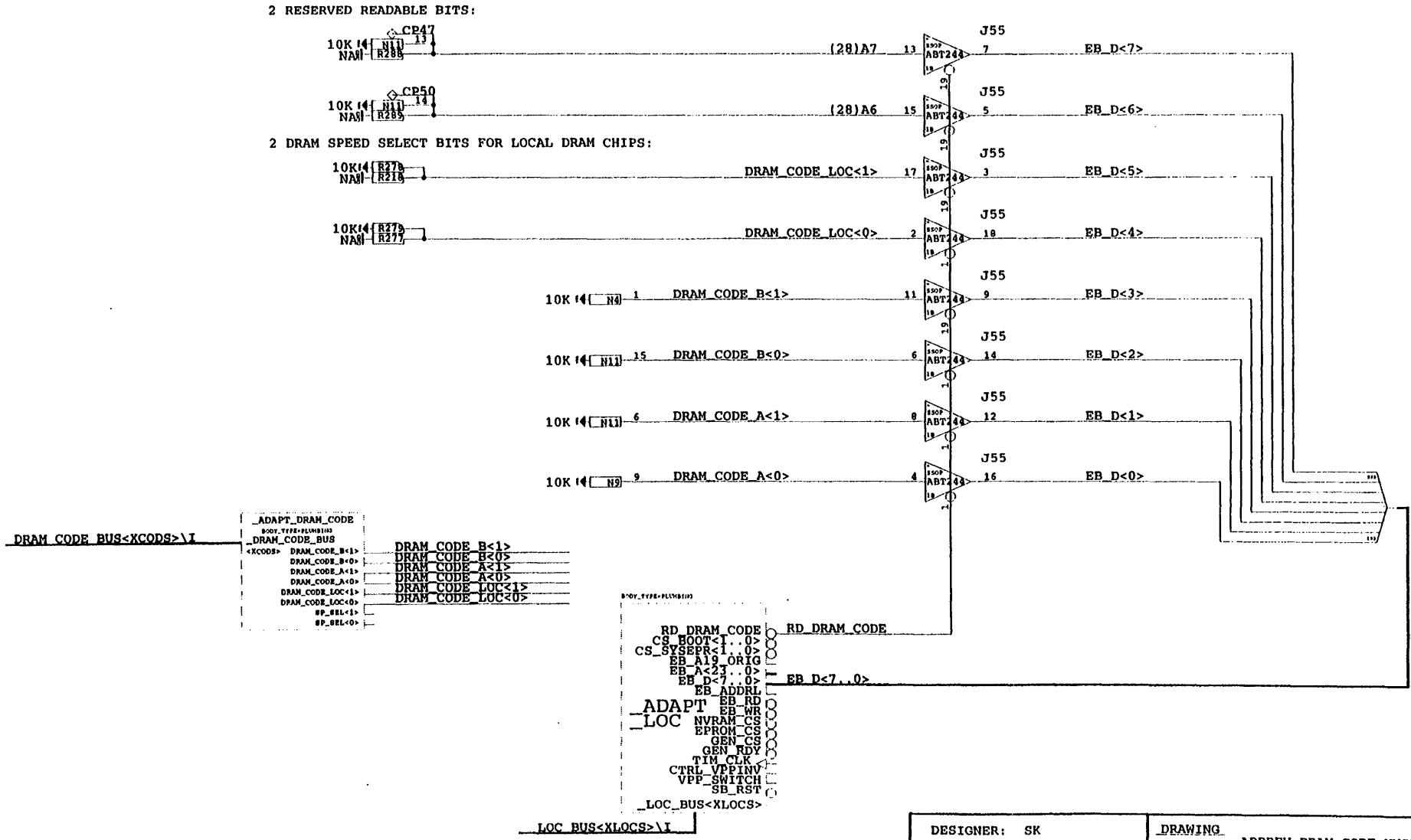
DESIGNER: SK Mon Apr 3 08:56:29 1995	DRAWING ABBREV=KBD_MSE_CONN APR/25/95 Manual	SHEET 27 OF 94
COMPUTERS	KBD_MSE_CONN	REV. 0.2



FLAT_DRAWING: NVRAM_UNIT_27.LOGIC.1.1

INSTANTIATION: (CPU5V NVRAM_UNIT50P)

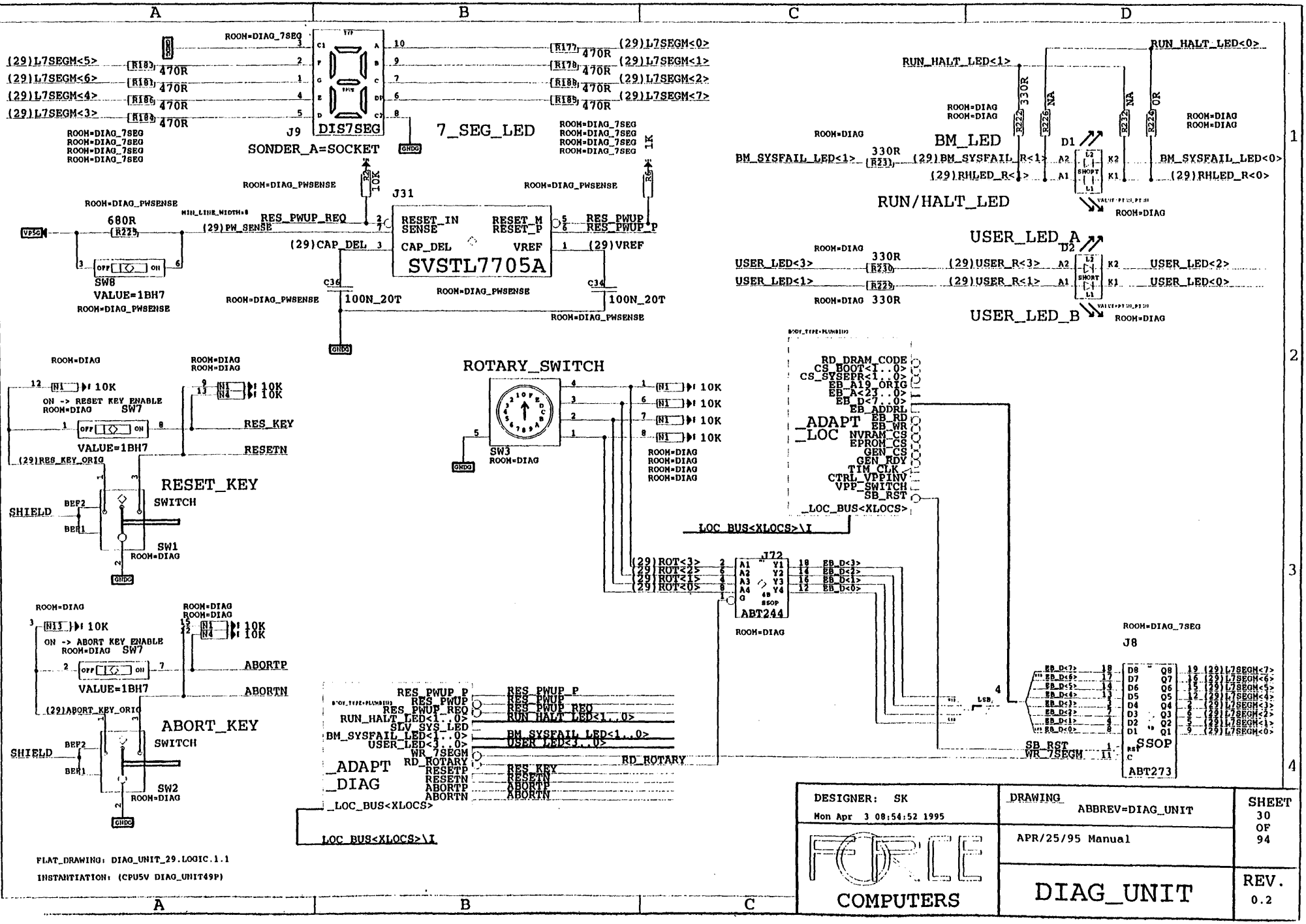
DESIGNER: SK Mon Apr 3 08:55:20 1995	DRAWING ABBREV=NVRAM_UNIT	SHEET 28 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	NVRAM_UNIT	REV. 0.2

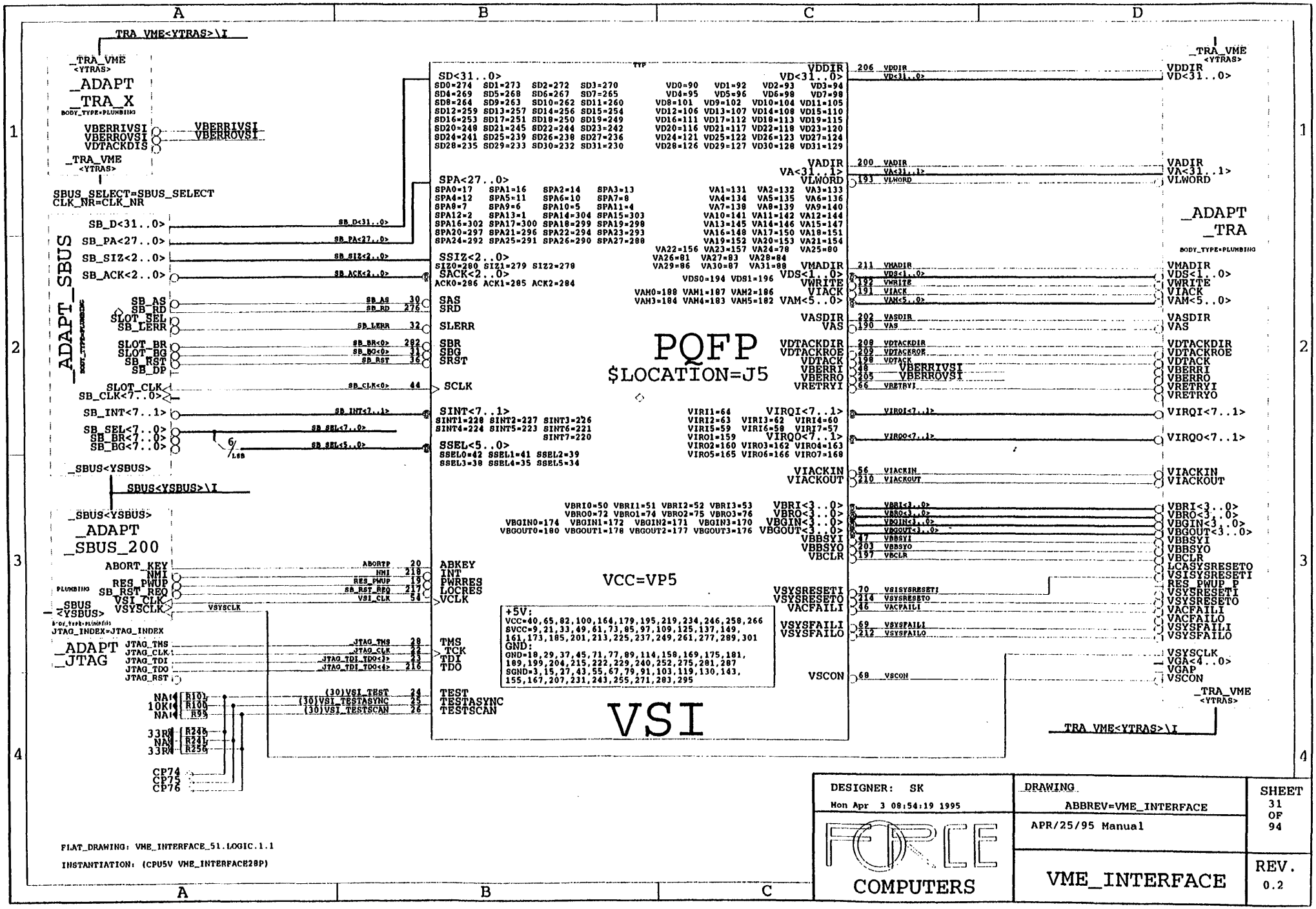


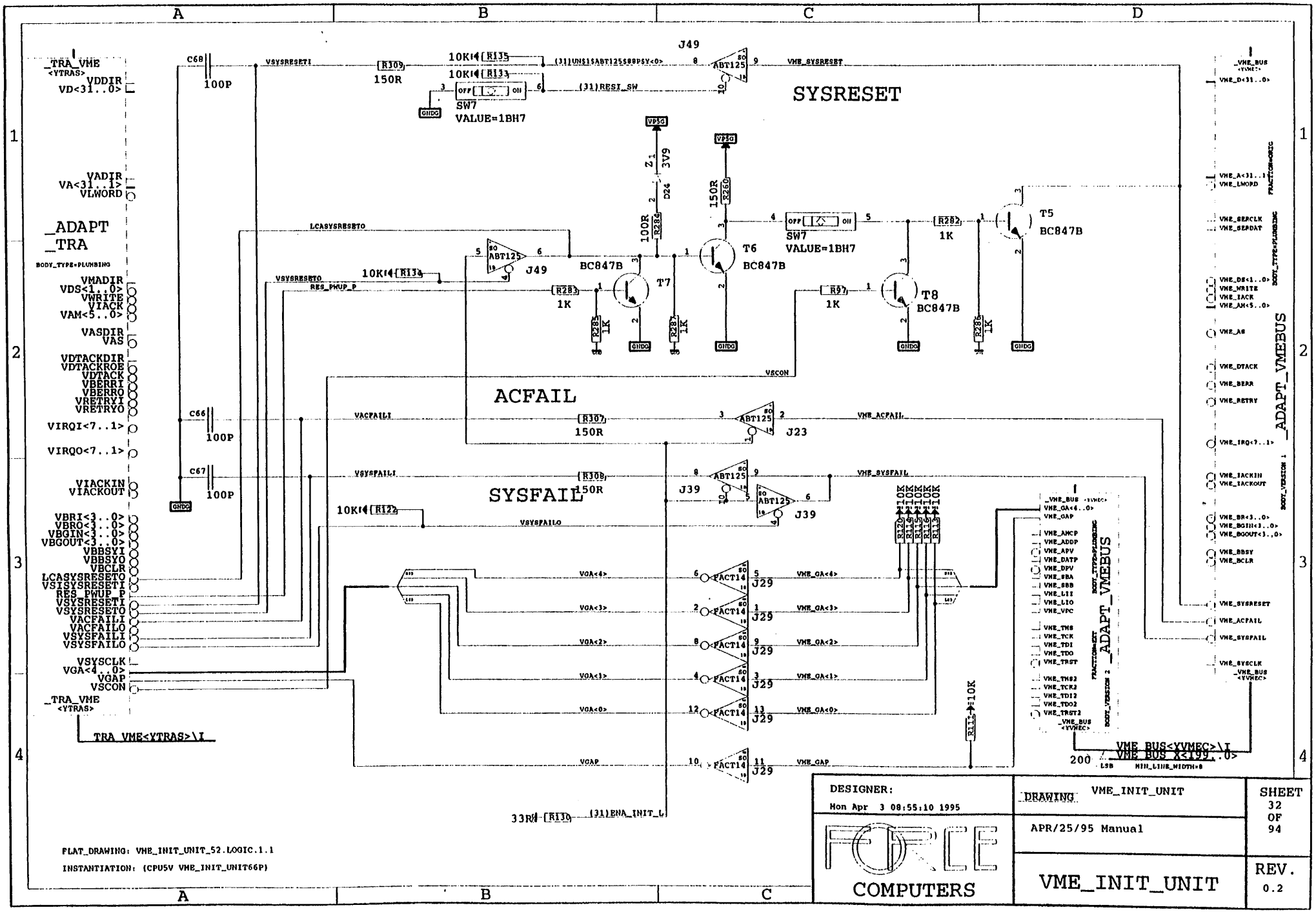
FLAT_DRAWING: DRAM_CODE_UNIT_28.LOGIC.1.1

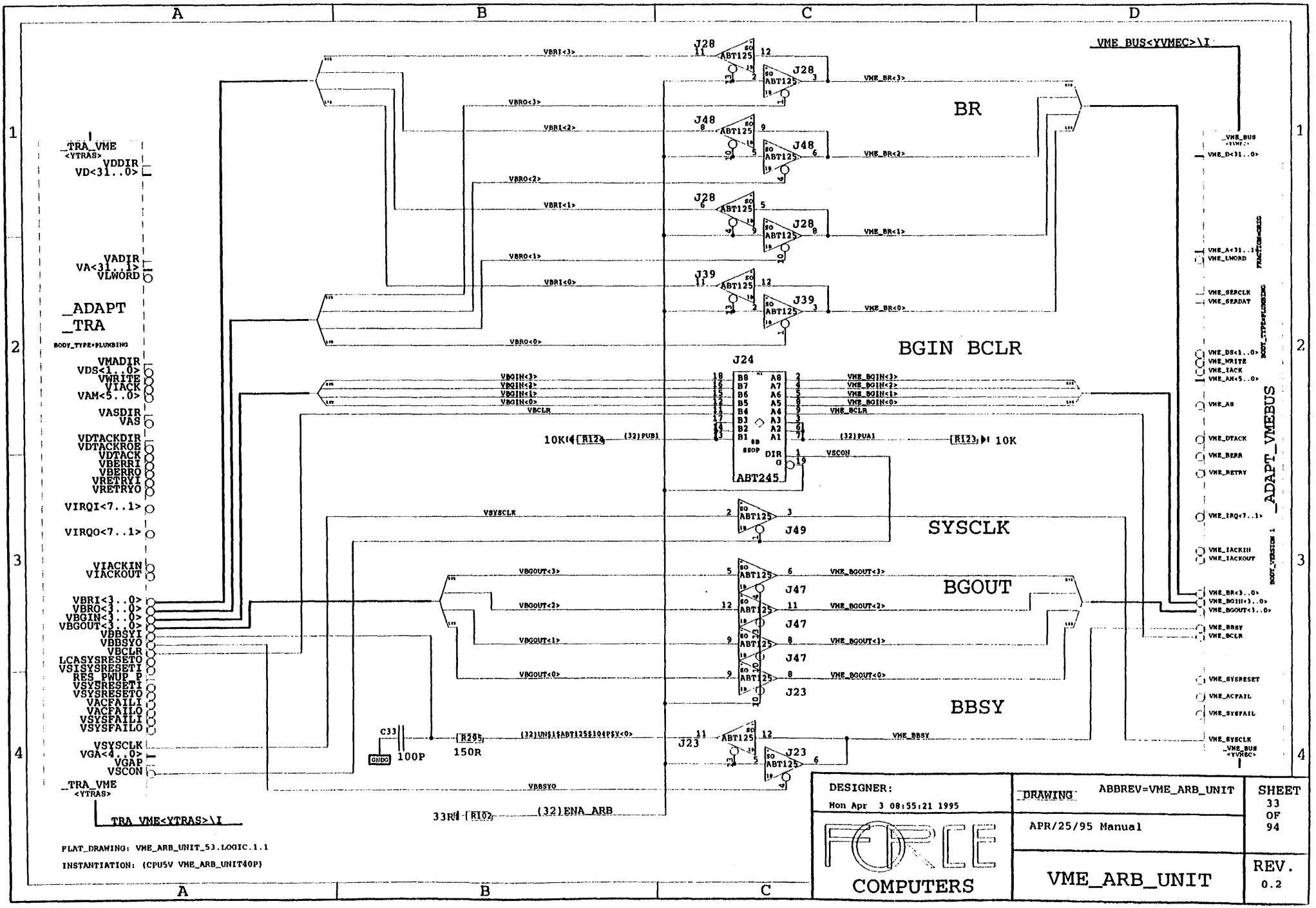
INSTANTIATION: (CPU5V DRAM_CODE_UNIT73P)

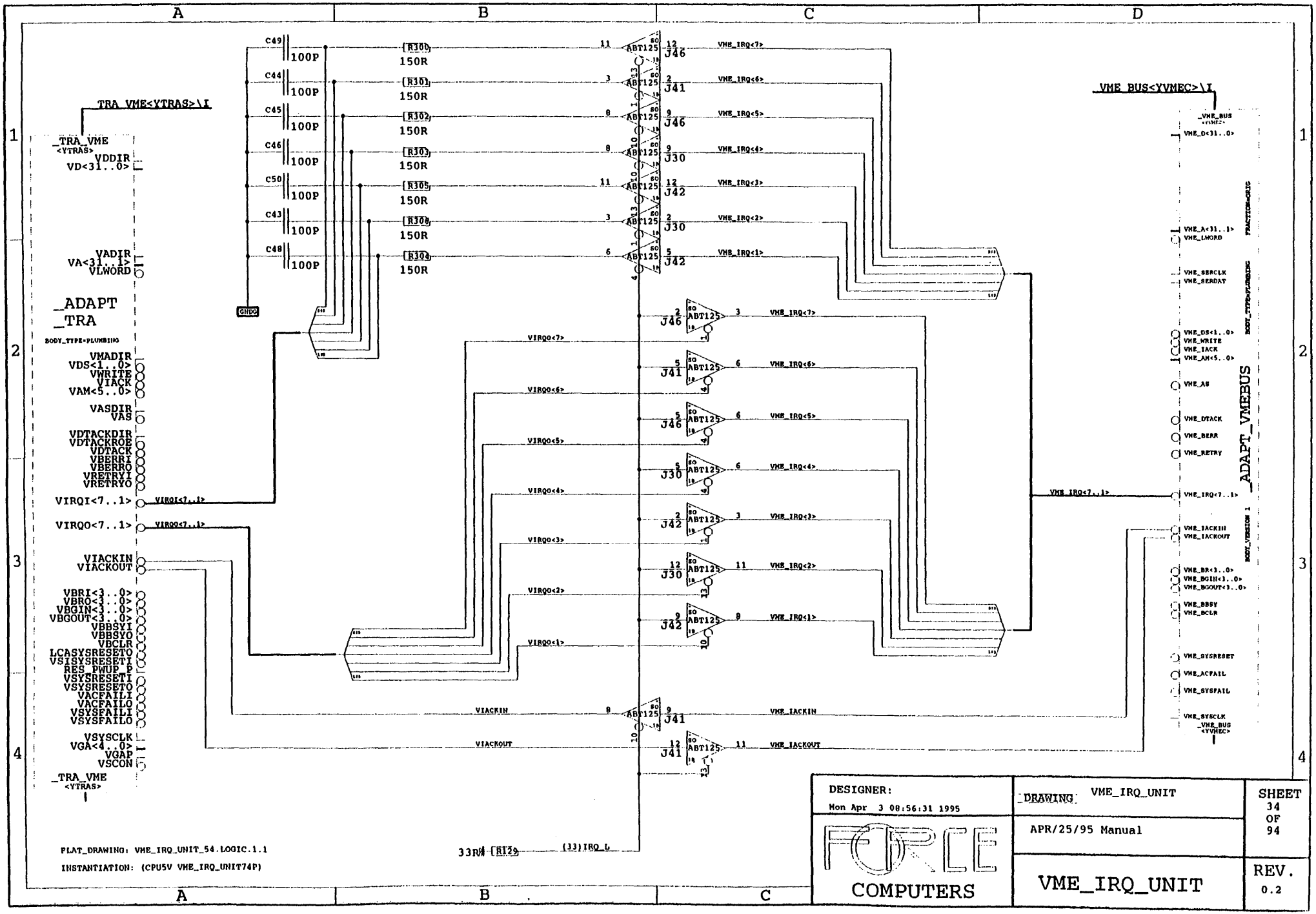
DESIGNER: SK Mon Apr 3 08:54:17 1995	DRAWING ABBREV=DRAM_CODE_UNIT	SHEET 29 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CODE_UNIT	REV. 0.2

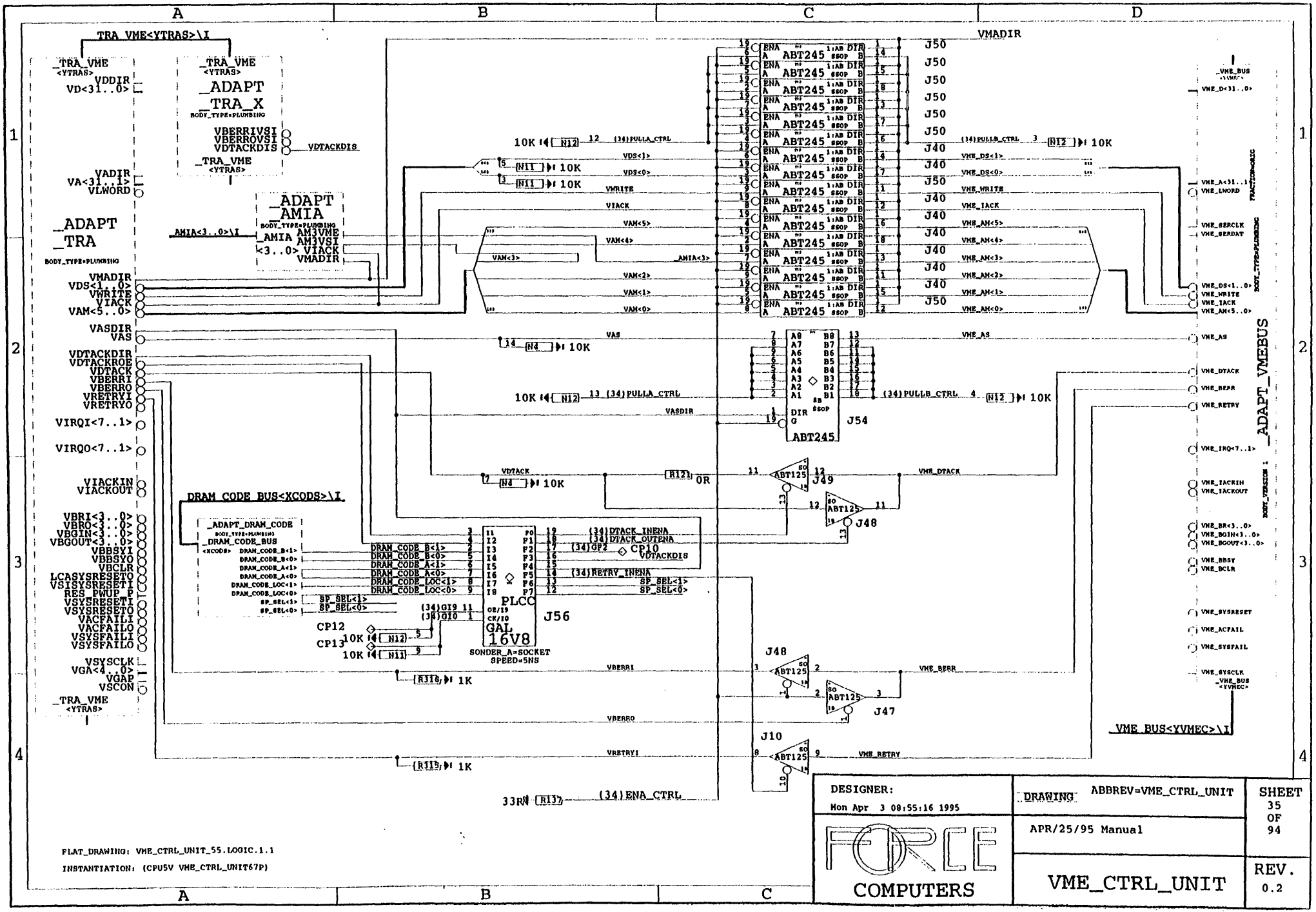

DIS7SEG
7_SEG_LED
SONDER_A=SOCKET
J9
J31
SVSTL7705A
RESET_IN SENSE
RESET_M RESET_P
CAP_DEL
VREF
RES_PWUP_REQ
RES_PWUP
RES_PWUP_P
(29)CAP_DEL
(29)VREF
(29)PW_SENSE
680R
VALUE=1BH7
SW6
100N_20T
ROOM=DIAG_PWSENSE
ROOM=DIAG_7SEG
(29)L7SEGM<0>
(29)L7SEGM<1>
(29)L7SEGM<2>
(29)L7SEGM<3>
(29)L7SEGM<4>
(29)L7SEGM<5>
(29)L7SEGM<6>
(29)L7SEGM<7>
470R
10K
1K
BM_LED
RUN/HALT_LED
USER_LED_A
USER_LED_B
BM_SYSFAIL_LED<1>
BM_SYSFAIL_LED<0>
RUN_HALT_LED<1>
RUN_HALT_LED<0>
(29)BM_SYSFAIL_R<1>
(29)RHLED_R<1>
(29)RHLED_R<0>
USER_LED<3>
USER_LED<2>
USER_LED<1>
USER_LED<0>
(29)USER_R<3>
(29)USER_R<1>
330R
ROOM=DIAG
D1
D2
SHORT
ROTARY_SWITCH
SW3
RESET_KEY
SWITCH
SW1
SHIELD
RESETP
RESETN
RES_KEY
ON -> RESET KEY ENABLE
(29)RES_KEY_ORIG
ABORT_KEY
SW2
ABORTP
ABORTN
ON -> ABORT KEY ENABLE
(29)ABORT_KEY_ORIG
ABT244
ABT273
SSOP
J72
J8
RD_DRAM_CODE
CS_BOOT<1..0>
CS_SYSEPR<1..0>
EB_A19_ORIG
EB_A<23..0>
EB_D<7..0>
EB_ADDRL
EB_RD
EB_WR
NVRAM_CS
EPROM_CS
GEN_CS
GEN_RDY
TIM_CLK
CTRL_VPPINV
VPP_SWITCH
SB_RST
_LOC_BUS<XLOCS>
LOC_BUS<XLOCS>\I
_ADAPT
_LOC
_DIAG
RES_PWUP_REQ
RUN_HALT_LED<1..0>
SLV_SYS_LED
BM_SYSFAIL_LED<1..0>
USER_LED<3..0>
WR_7SEGM
RD_ROTARY
ROT<3>
ROT<2>
ROT<1>
ROT<0>
EB_D<7>
EB_D<6>
EB_D<5>
EB_D<4>
EB_D<3>
EB_D<2>
EB_D<1>
EB_D<0>
FLAT_DRAWING: DIAG_UNIT_29.LOGIC.1.1
INSTANTIATION: (CPU5V DIAG_UNIT49P)
DESIGNER: SK
Mon Apr 3 08:54:52 1995
DRAWING
ABBREV=DIAG_UNIT
APR/25/95 Manual
SHEET 30 OF 94
COMPUTERS
DIAG_UNIT
REV. 0.2

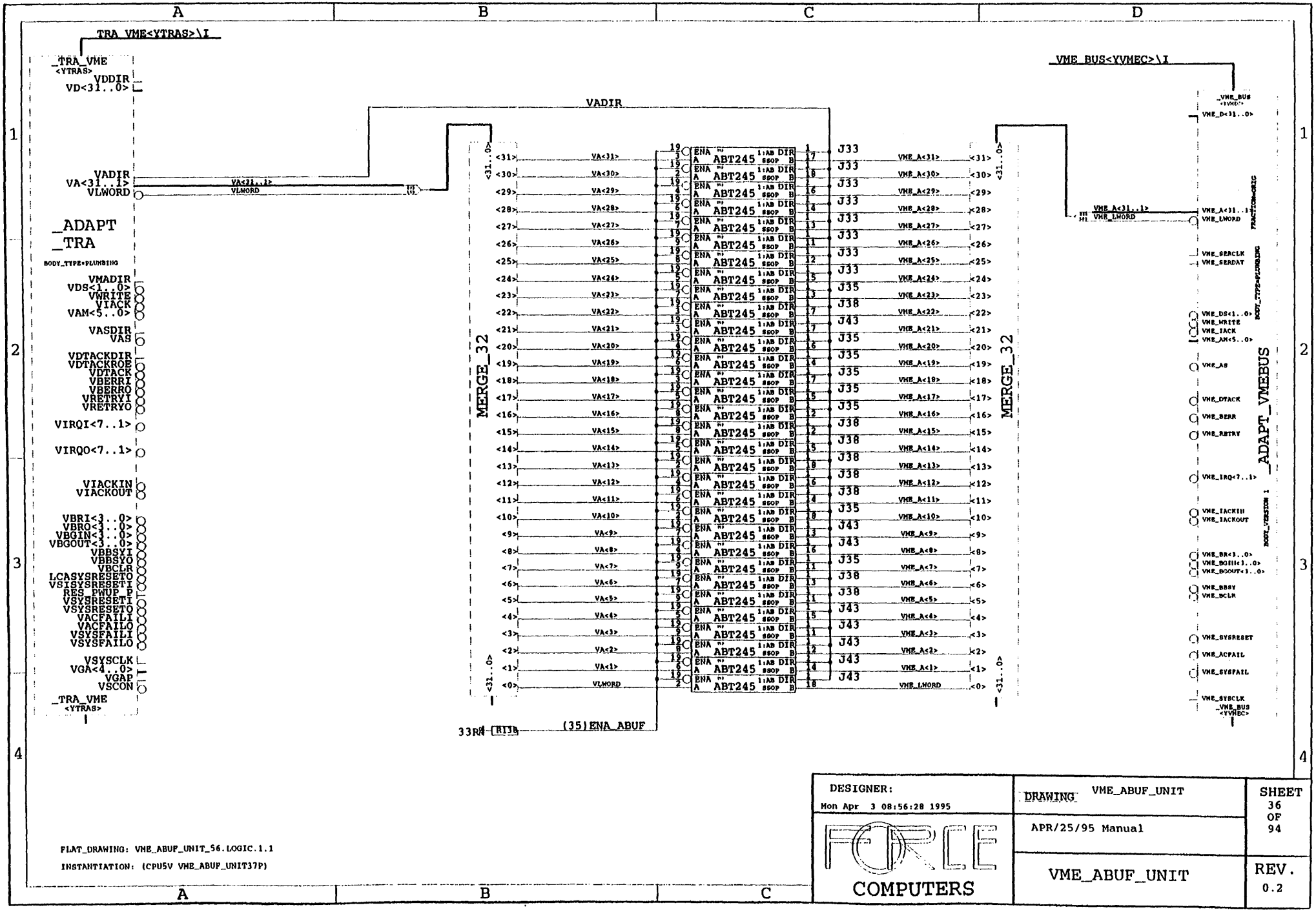

TRA VME<YTRAS>\I
_TRA_VME
<YTRAS>
_ADAPT
_TRA_X
BODY_TYPE=PLUMBING
VBERRIVSI
VBERROVSI
VDTACKDIS
SBUS_SELECT=SBUS_SELECT
CLK_NR=CLK_NR
ADAPT_SBUS
SB_D<31..0>
SB_PA<27..0>
SB_SIZ<2..0>
SB_ACK<2..0>
SB_AS
SB_RD
SLOT_SEL
SB_LERR
SLOT_BR
SLOT_BG
SB_RST
SB_DP
SLOT_CLK
SB_CLK<7..0>
SB_INT<7..1>
SB_SEL<7..0>
SB_BR<7..0>
SB_BG<7..0>
_SBUS<YSBUS>
SBUS<YSBUS>\I
_ADAPT
_SBUS_200
ABORT_KEY
NMI
RES_PWUP
SB_RST_REQ
VSI_CLK
VSYSCLK
JTAG_INDEX=JTAG_INDEX
_ADAPT
_JTAG
JTAG_TMS
JTAG_CLK
JTAG_TDI
JTAG_TDO
JTAG_RST
10K
33R
CP74
CP75
CP76
SD<31..0>
SPA<27..0>
SSIZ<2..0>
SACK<2..0>
SAS
SRD
SLERR
SBR
SBG
SRST
SCLK
SINT<7..1>
SSEL<5..0>
ABKEY
INT
PWRRES
LOCRES
VCLK
TMS
TCK
TDI
TDO
TEST
TESTASYNC
TESTSCAN
PQFP
$LOCATION=J5
VCC=VP5
+5V:
VCC=40,65,82,100,164,179,195,219,234,246,258,266
8VCC=9,21,33,49,61,73,85,97,109,125,137,149,
161,173,185,201,213,225,237,249,261,277,289,301
GND:
GND=18,29,37,45,71,77,89,114,158,169,175,181,
189,199,204,215,222,229,240,252,275,281,287
SGND=3,15,27,43,55,67,79,91,103,119,130,143,
155,167,207,231,243,255,271,283,295
VSI
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRO<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
VSYSRESETI
VSYSRESETO
VACFAILI
VSYSFAILI
VSYSFAILO
VSCON
_TRA_VME
<YTRAS>
_ADAPT
_TRA
BODY_TYPE=PLUMBING
LCASYSRESETO
VSISYSRESETI
RES_PWUP_P
VSYSCLK
VGA<4..0>
VGAP
TRA VME<YTRAS>\I
FLAT_DRAWING: VME_INTERFACE_51.LOGIC.1.1
INSTANTIATION: (CPU5V VME_INTERFACE28P)
DESIGNER: SK
Mon Apr 3 08:54:19 1995
DRAWING
ABBREV=VME_INTERFACE
APR/25/95 Manual
SHEET
31
OF
94
COMPUTERS
VME_INTERFACE
REV.
0.2


SYSRESET
ACFAIL
SYSFAIL
SW7
VALUE=1BH7
J49
T5
BC847B
T6
BC847B
T7
T8
BC847B
_ADAPT_VMEBUS
_TRA_VME
TRA_VME<YTRAS>\I
VME BUS<YVMEC>\I
FLAT_DRAWING: VME_INIT_UNIT_52.LOGIC.1.1
INSTANTIATION: (CPU5V VME_INIT_UNIT66P)
DESIGNER:
Mon Apr 3 08:55:10 1995
DRAWING VME_INIT_UNIT
APR/25/95 Manual
SHEET 32 OF 94
COMPUTERS
VME_INIT_UNIT
REV. 0.2


A
B
C
D
_TRA_VME
<YTRAS>
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
_ADAPT
_TRA
BODY_TYPE=PLUMBING
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VRETRYO
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRO<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
LCASYSRESETO
VSISYSRESETI
RES_PWUP_P
VSYSRESETI
VSYSRESETO
VACFAILI
VACFAILO
VSYSFAILI
VSYSFAILO
VSYSCLK
VGA<4..0>
VGAP
VSCON
_TRA_VME
<YTRAS>
TRA_VME<YTRAS>\I
FLAT_DRAWING: VME_ARB_UNIT_53.LOGIC.1.1
INSTANTIATION: (CPU5V VME_ARB_UNIT40P)
VME_BUS<YVMEC>\I
BR
BGIN BCLR
SYSCLK
BGOUT
BBSY
J28
J48
J39
J24
J49
J47
J23
ABT125
ABT245
VBRI<3>
VBRO<3>
VBRI<2>
VBRO<2>
VBRI<1>
VBRO<1>
VBRI<0>
VBRO<0>
VME_BR<3>
VME_BR<2>
VME_BR<1>
VME_BR<0>
VBGIN<3>
VBGIN<2>
VBGIN<1>
VBGIN<0>
VBCLR
VME_BGIN<3>
VME_BGIN<2>
VME_BGIN<1>
VME_BGIN<0>
VME_BCLR
10K
R124
R123
(32)PUB1
(32)PUA1
VSCON
VSYSCLK
VBGOUT<3>
VBGOUT<2>
VBGOUT<1>
VBGOUT<0>
VME_BGOUT<3>
VME_BGOUT<2>
VME_BGOUT<1>
VME_BGOUT<0>
C33
100P
GNDG
R205
150R
(32)UN1ABT125$104PSV<0>
VME_BBSY
VBBSYO
33R
R102
(32)ENA_ARB
_VME_BUS
<YVME>
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_SERCLK
VME_SERDAT
VME_DS<1..0>
VME_WRITE
VME_IACK
VME_AM<5..0>
VME_AS
VME_DTACK
VME_BERR
VME_RETRY
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BGIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLK
_VME_BUS
<YVMEC>
_ADAPT_VMEBUS
DESIGNER:
Mon Apr 3 08:55:21 1995
DRAWING
ABBREV=VME_ARB_UNIT
SHEET
33
OF
94
APR/25/95 Manual
COMPUTERS
VME_ARB_UNIT
REV.
0.2


A
B
C
D
TRA_VME<YTRAS>\I
_TRA_VME
<YTRAS>
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
_ADAPT
_TRA
BODY_TYPE=PLUMBING
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VRETRYO
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRO<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
LCASYSRESETO
VSISYSRESETI
RES_PWUP_P
VSYSRESETI
VSYSRESETO
VACFAILI
VACFAILO
VSYSFAILI
VSYSFAILO
VSYSCLK
VGA<4..0>
VGAP
VSCON
_TRA_VME
<YTRAS>
C49 100P
C44 100P
C45 100P
C46 100P
C50 100P
C43 100P
C48 100P
R300 150R
R301 150R
R302 150R
R303 150R
R305 150R
R308 150R
R304 150R
GNDC
ABT125
J46
J41
J30
J42
VME_IRQ<7>
VME_IRQ<6>
VME_IRQ<5>
VME_IRQ<4>
VME_IRQ<3>
VME_IRQ<2>
VME_IRQ<1>
VIRQO<7>
VIRQO<6>
VIRQO<5>
VIRQO<4>
VIRQO<3>
VIRQO<2>
VIRQO<1>
VIACKIN
VIACKOUT
VME_IACKIN
VME_IACKOUT
VME_IRQ<7..1>
VME BUS<YVMEC>\I
_VME_BUS
<YVMEC>
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_SERCLK
VME_SERDAT
VME_DS<1..0>
VME_WRITE
VME_IACK
VME_AM<5..0>
VME_AS
VME_DTACK
VME_BERR
VME_RETRY
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BGIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLK
_VME_BUS
<YVMEC>
_ADAPT_VMEBUS
BODY_TYPE=PLUMBING
FUNCTION=MISC
BODY_VERSION 1
33R4 [R129
(33)IRQ_L
PLAT_DRAWING: VME_IRQ_UNIT_54.LOGIC.1.1
INSTANTIATION: (CPU5V VME_IRQ_UNIT74P)
DESIGNER:
Mon Apr 3 08:56:31 1995
DRAWING VME_IRQ_UNIT
APR/25/95 Manual
SHEET 34 OF 94
FORCE COMPUTERS
VME_IRQ_UNIT
REV. 0.2

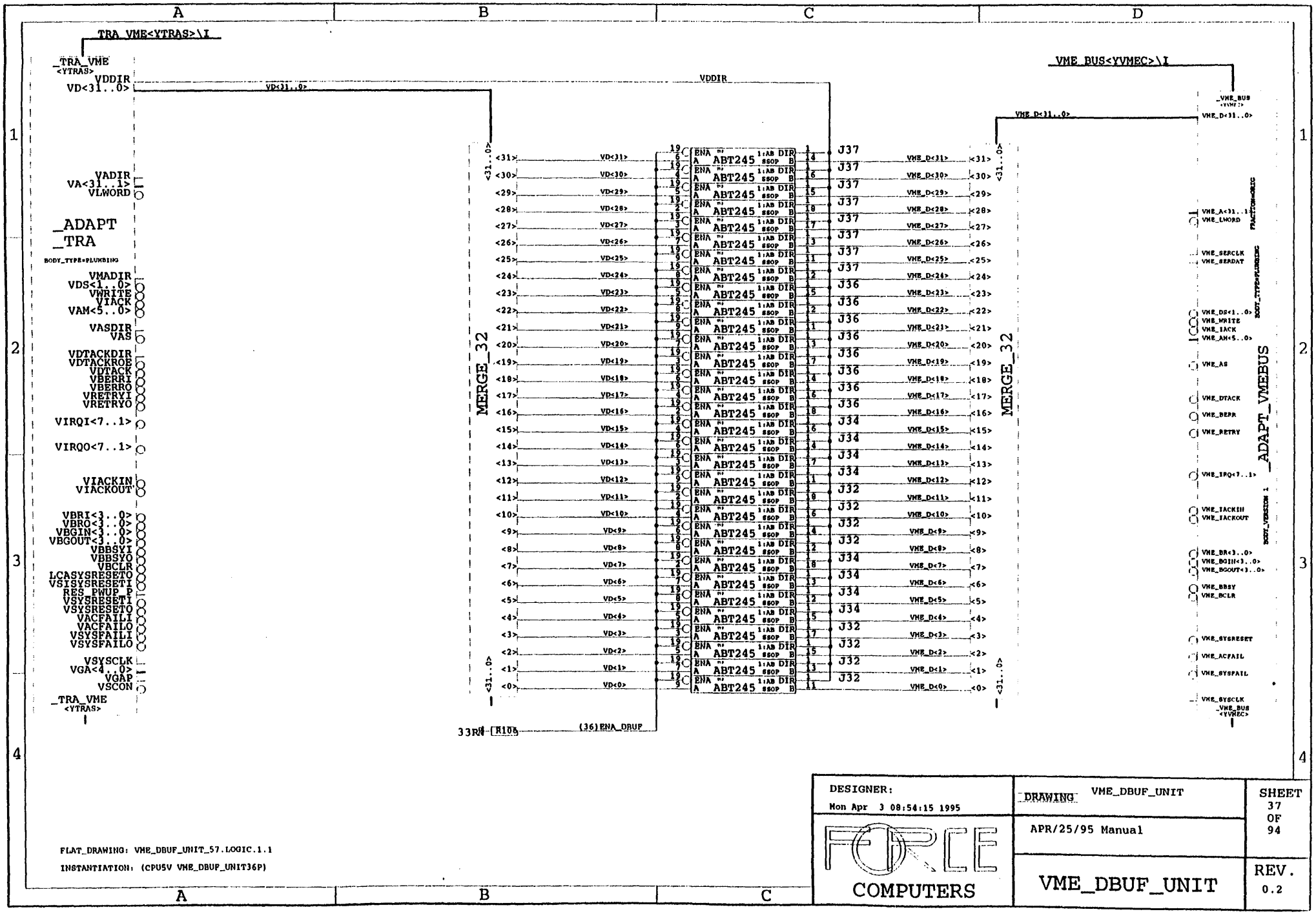

TRA VME<YTRAS>\I
_TRA_VME <YTRAS>
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
_ADAPT _TRA
BODY_TYPE=PLUMBING
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VRETRYO
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRO<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
LCASYSRESETO
VSISYSRESETI
RES_PWUP_P
VSYSRESETI
VSYSRESETO
VACFAILI
VACFAILO
VSYSFAILI
VSYSFAILO
VSYSCLK
VGA<4..0>
VGAP
VSCON
_ADAPT _X
VBERRIVSI
VBERROVSI
VDTACKDIS
_ADAPT _AMIA
AMIA<3..0>\I
AM3VME
AM3VST
<3..0> VIACK
VMADIR
ABT245
J50
J40
J54
J49
J48
J47
J10
ABT125
10K
0R
1K
33R
(34)PULLA_CTRL
(34)PULLB_CTRL
(34)ENA_CTRL
VDS<1>
VDS<0>
VWRITE
VIACK
VAM<5>
VAM<4>
VAM<3>
VAM<2>
VAM<1>
VAM<0>
VAS
VASDIR
VDTACK
VBERRI
VBERRO
VRETRYI
DRAM CODE BUS<XCODS>\I
_ADAPT_DRAM_CODE
_DRAM_CODE_BUS
DRAM_CODE_B<1>
DRAM_CODE_B<0>
DRAM_CODE_A<1>
DRAM_CODE_A<0>
DRAM_CODE_LOC<1>
DRAM_CODE_LOC<0>
SP_SEL<1>
SP_SEL<0>
PLCC
GAL 16V8
J56
SONDER_A=SOCKET
SPEED=5NS
(34)DTACK_INENA
(34)DTACK_OUTENA
(34)OP2
CP10
VDTACKDIS
(34)RETRY_INENA
(34)OI9
(34)OI0
CP12
CP13
VME_DTACK
VME_BERR
VME_RETRY
VME_AS
VME_DS<1>
VME_DS<0>
VME_WRITE
VME_IACK
VME_AM<5>
VME_AM<4>
VME_AM<3>
VME_AM<2>
VME_AM<1>
VME_AM<0>
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_BERCLK
VME_BERDAT
VME_DS<1..0>
VME_AM<5..0>
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BOIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLK
_VME_BUS <YVMEC>
VME_BUS<YVMEC>\I
_ADAPT_VMEBUS
FLAT_DRAWING: VME_CTRL_UNIT_55.LOGIC.1.1
INSTANTIATION: (CPU5V VME_CTRL_UNIT67P)
DESIGNER:
Mon Apr 3 08:55:16 1995
DRAWING ABBREV=VME_CTRL_UNIT
APR/25/95 Manual
SHEET 35 OF 94
FORCE COMPUTERS
VME_CTRL_UNIT
REV. 0.2

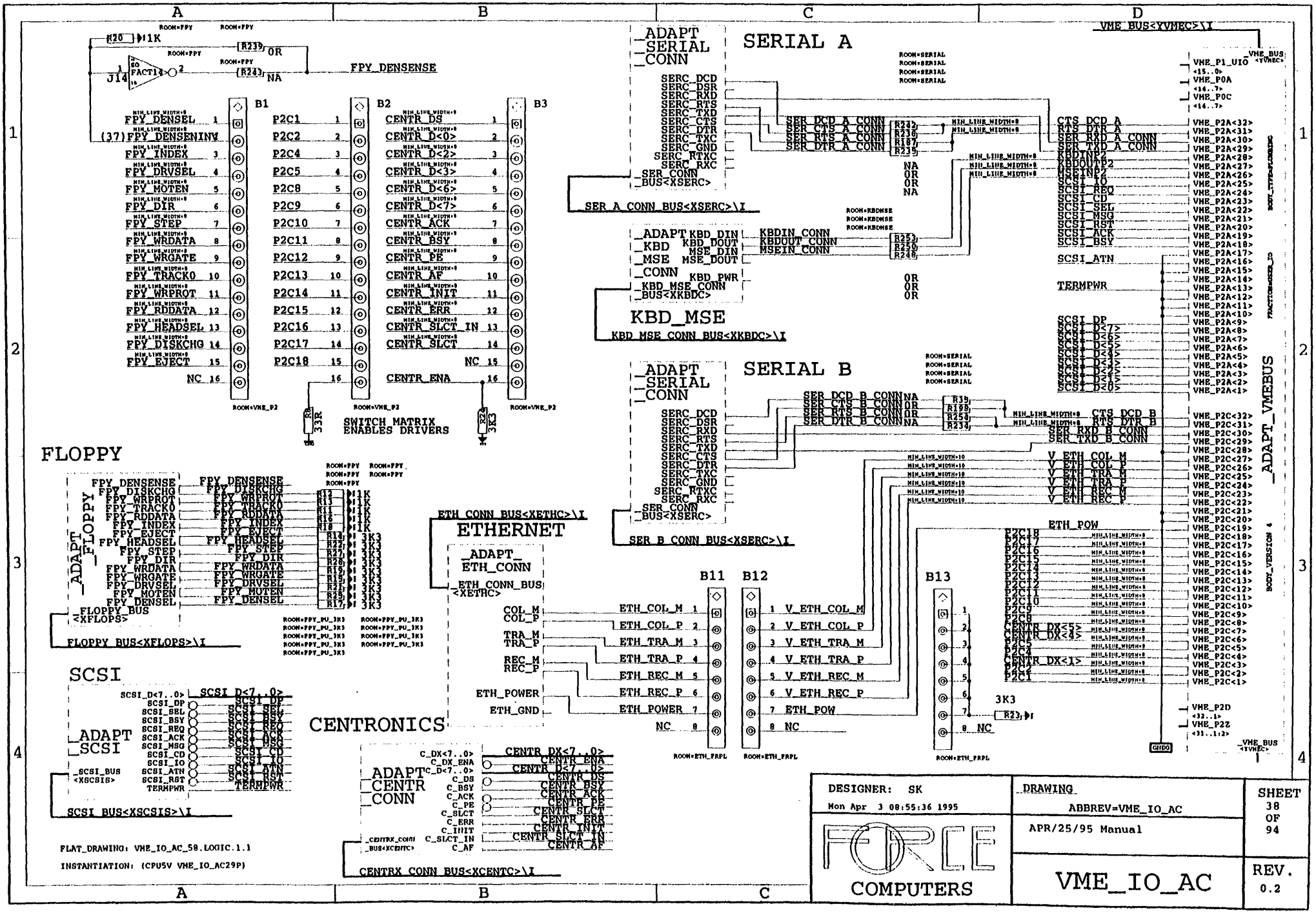


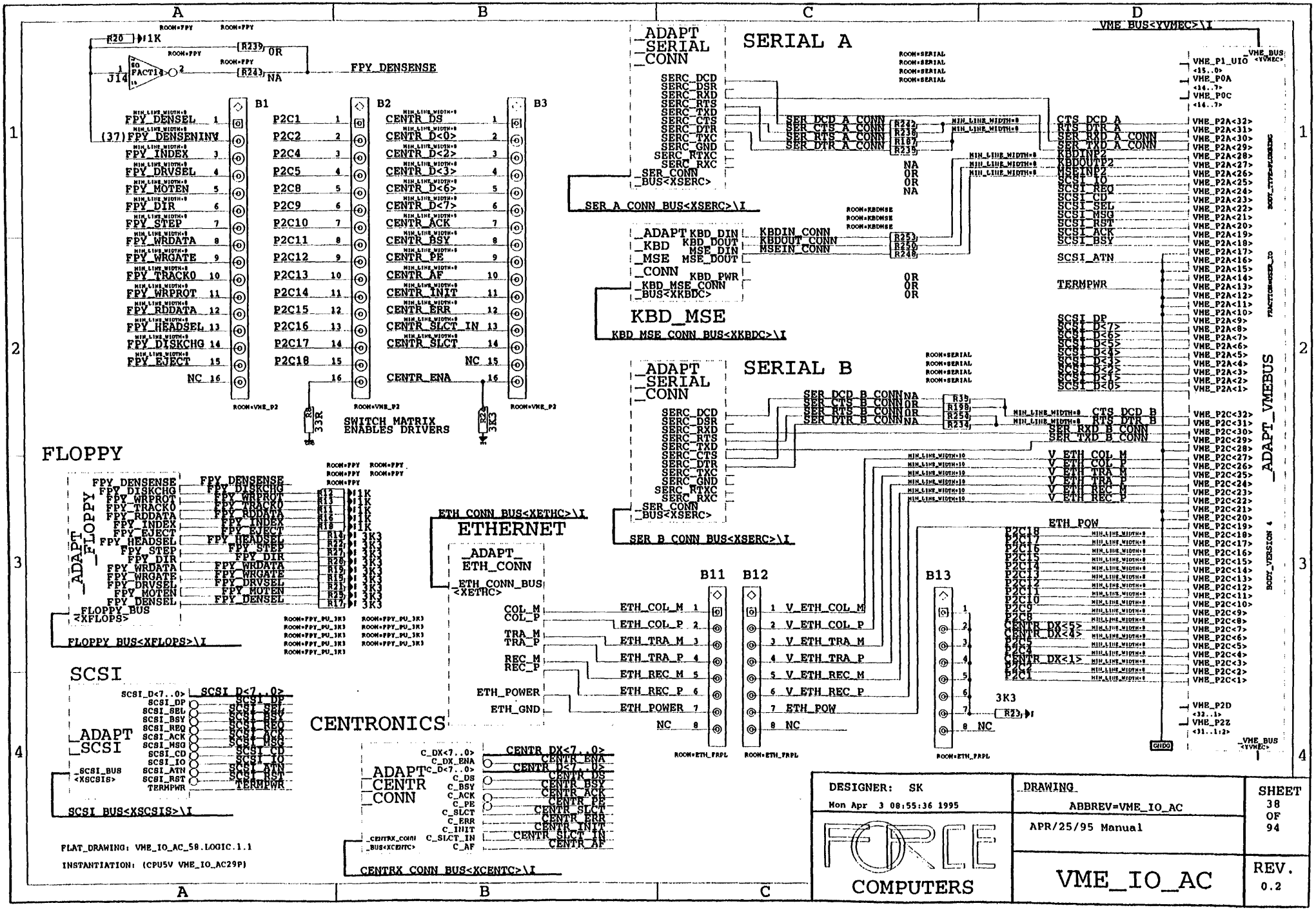
FLAT_DRAWING: VME_ABUF_UNIT_56.LOGIC.1.1

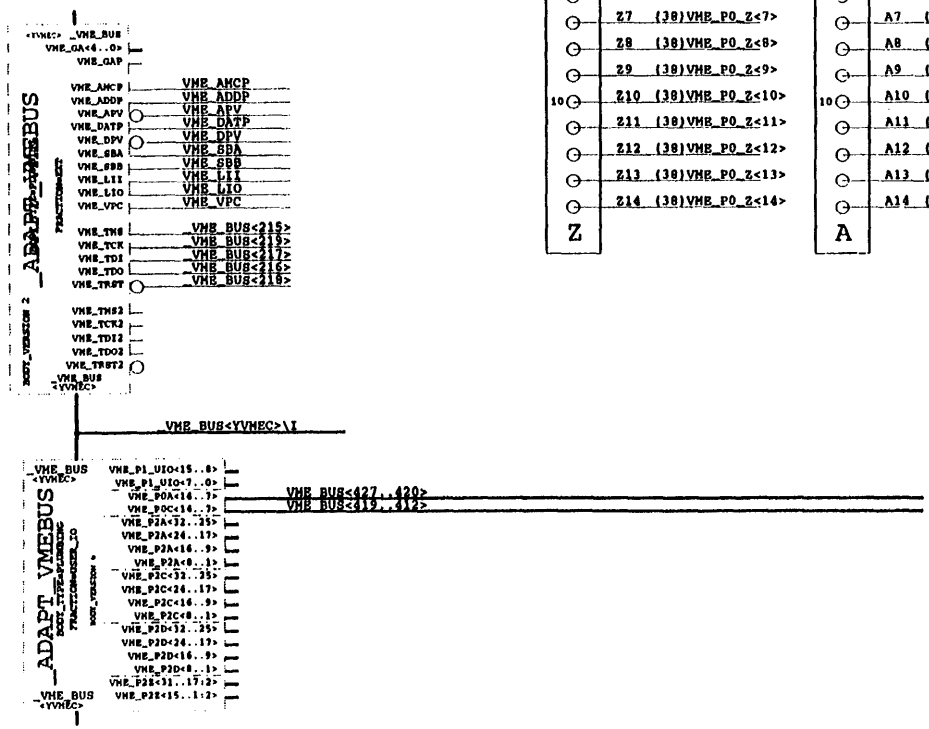
INSTANTIATION: (CPU5V VME_ABUF_UNIT37P)

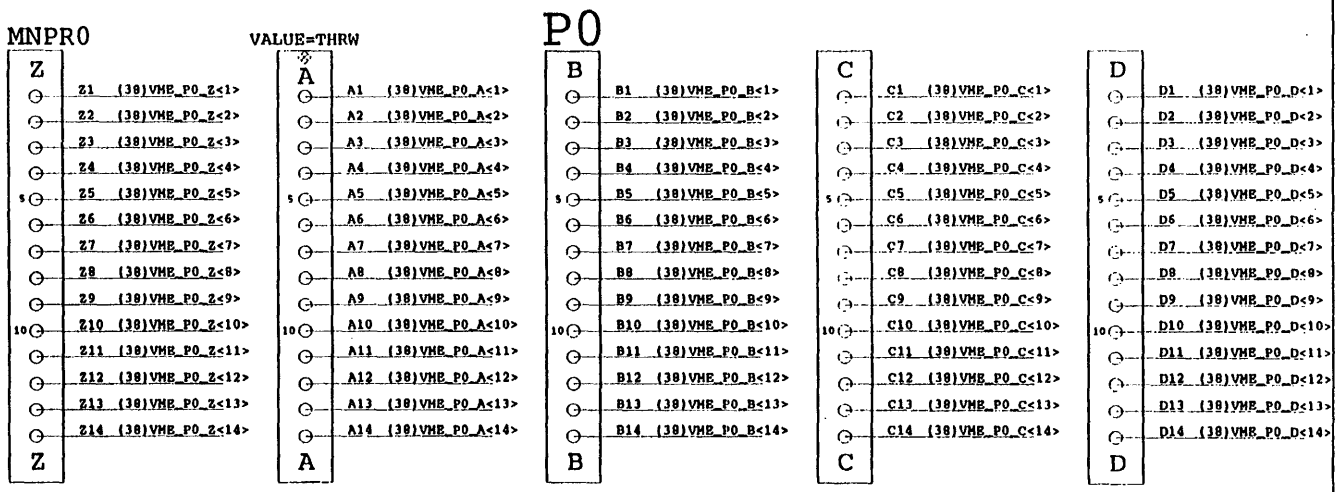
DESIGNER: Mon Apr 3 08:56:28 1995	DRAWING VME_ABUF_UNIT	SHEET 36 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	VME_ABUF_UNIT	REV. 0.2


TRA_VME<YTRAS>\I
_TRA_VME
<YTRAS>
VDDIR
VD<31..0>
VADIR
VA<31..1>
VLWORD
_ADAPT
_TRA
BODY_TYPE=PLUMBING
VMADIR
VDS<1..0>
VWRITE
VIACK
VAM<5..0>
VASDIR
VAS
VDTACKDIR
VDTACKROE
VDTACK
VBERRI
VBERRO
VRETRYI
VRETRYO
VIRQI<7..1>
VIRQO<7..1>
VIACKIN
VIACKOUT
VBRI<3..0>
VBRQ<3..0>
VBGIN<3..0>
VBGOUT<3..0>
VBBSYI
VBBSYO
VBCLR
LCASYSRESETO
VSISYSRESETI
RES_PWUP_F
VSYSRESETI
VSYSRESETO
VACFAILI
VACFAILO
VSYSFAILI
VSYSFAILO
VSYSCLK
VGA<4..0>
VGAP
VSCON
_TRA_VME
<YTRAS>
VD<31..0>
VDDIR
MERGE_32
ABT245
ENA
A
B
1:AB DIR
SSOP
J37
J36
J34
J32
33R
(R106
(36)ENA_DBUF
VME_BUS<YVMEC>\I
_VME_BUS
<YVMEC>
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_SERCLK
VME_SERDAT
VME_DS<1..0>
VME_WRITE
VME_IACK
VME_AM<5..0>
VME_AS
VME_DTACK
VME_BERR
VME_RETRY
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BOIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLK
_ADAPT_VMEBUS
FUNCTION=MISC
BOOT_TYPE=PLUMBING
BOOT_VERSION = 1
FLAT_DRAWING: VME_DBUF_UNIT_57.LOGIC.1.1
INSTANTIATION: (CPU5V VME_DBUF_UNIT36P)
DESIGNER:
Mon Apr 3 08:54:15 1995
COMPUTERS
DRAWING VME_DBUF_UNIT
APR/25/95 Manual
VME_DBUF_UNIT
SHEET 37 OF 94
REV. 0.2


SERIAL A
SERIAL B
KBD_MSE
FLOPPY
SCSI
ETHERNET
CENTRONICS
SWITCH MATRIX
ENABLES DRIVERS
_ADAPT_VMEBUS
FPY_DENSENSE
SER_A_CONN_BUS<XSERC>\I
SER_B_CONN_BUS<XSERC>\I
KBD_MSE_CONN_BUS<XKBDC>\I
FLOPPY_BUS<XFLOPS>\I
ETH_CONN_BUS<XETHC>\I
SCSI_BUS<XSCSIS>\I
CENTRX_CONN_BUS<XCENTC>\I
FLAT_DRAWING: VME_IO_AC_50.LOGIC.1.1
INSTANTIATION: (CPU5V VME_IO_AC29P)
DESIGNER: SK
Mon Apr 3 08:55:36 1995
DRAWING
ABBREV=VME_IO_AC
APR/25/95 Manual
SHEET 38 OF 94
REV. 0.2
VME_IO_AC
FORCE COMPUTERS


FLOPPY
SCSI
ETHERNET
CENTRONICS
SERIAL A
SERIAL B
KBD_MSE
SWITCH MATRIX
ENABLES DRIVERS
FPY_DENSENSE
SER_A_CONN_BUS<XSERC>\I
SER_B_CONN_BUS<XSERC>\I
KBD_MSE_CONN_BUS<XKBDC>\I
ETH_CONN_BUS<XETHC>\I
FLOPPY_BUS<XFLOPS>\I
SCSI_BUS<XSCSIS>\I
CENTRX_CONN_BUS<XCENTC>\I
VME_BUS<YVMEC>\I
_ADAPT_VMEBUS
FLAT_DRAWING: VME_IO_AC_58.LOGIC.1.1
INSTANTIATION: (CPU5V VME_IO_AC29P)
DESIGNER: SK
Mon Apr 3 08:55:36 1995
DRAWING
ABBREV=VME_IO_AC
APR/25/95 Manual
SHEET 38 OF 94
REV. 0.2
FORCE COMPUTERS
VME_IO_AC

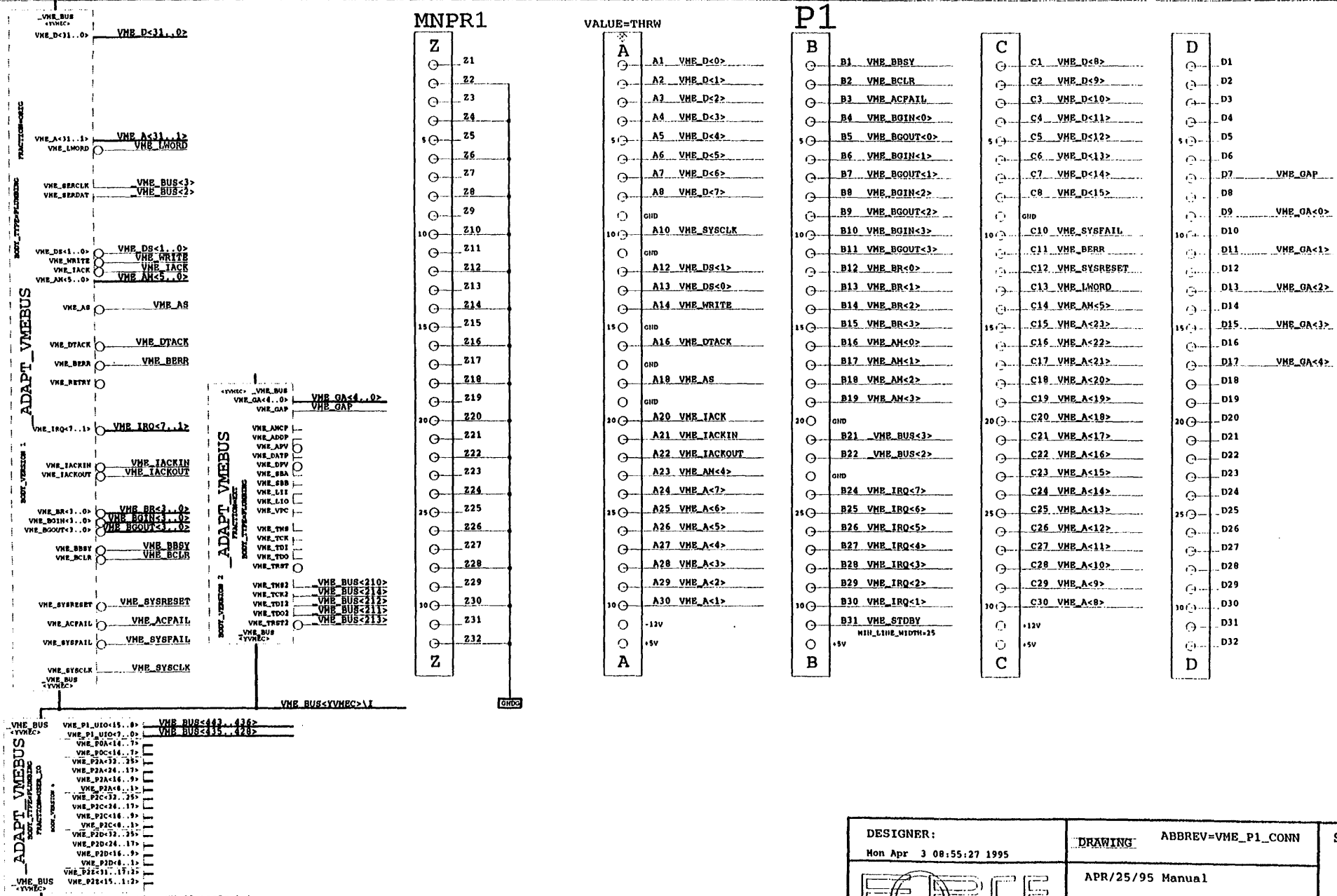


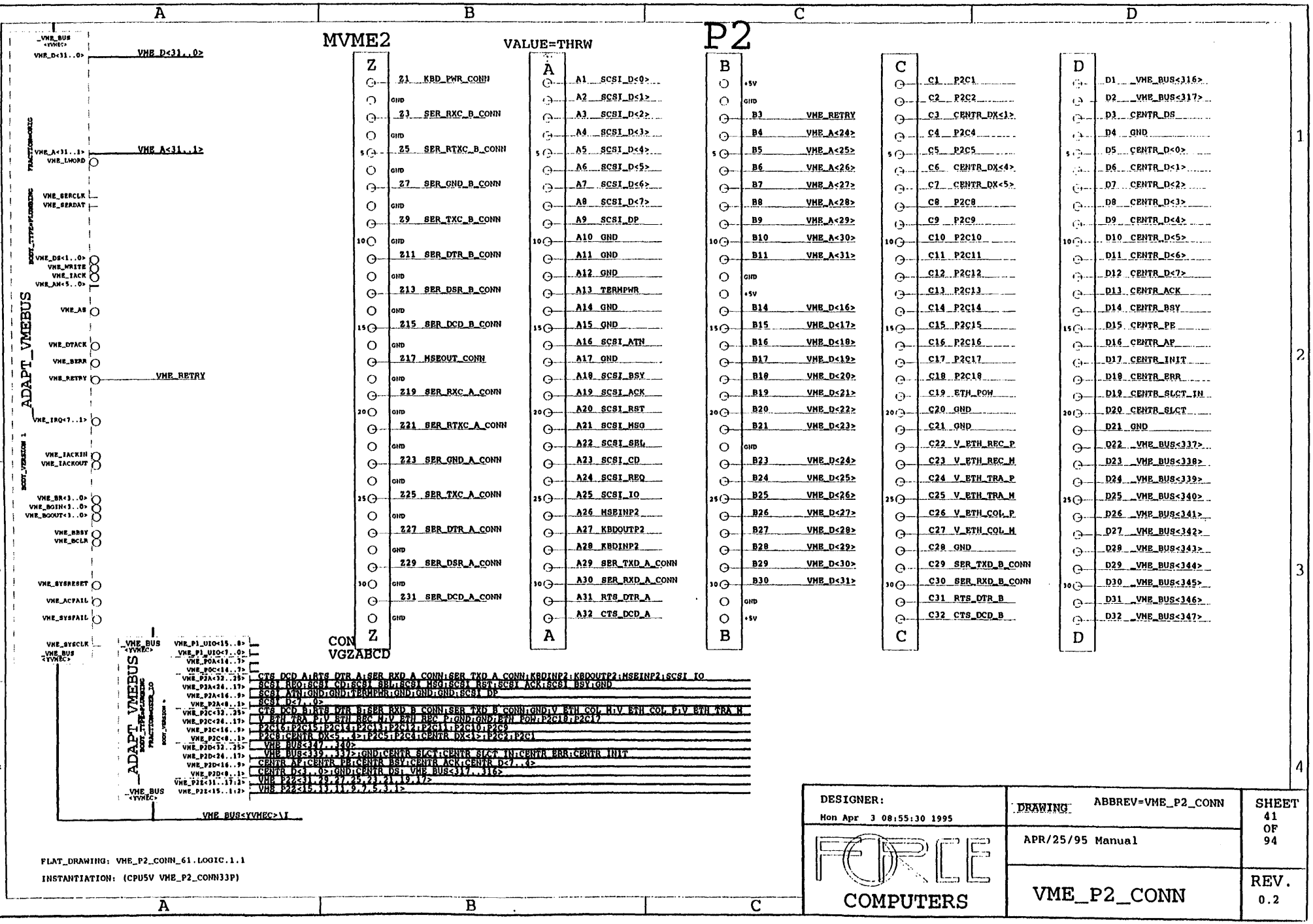


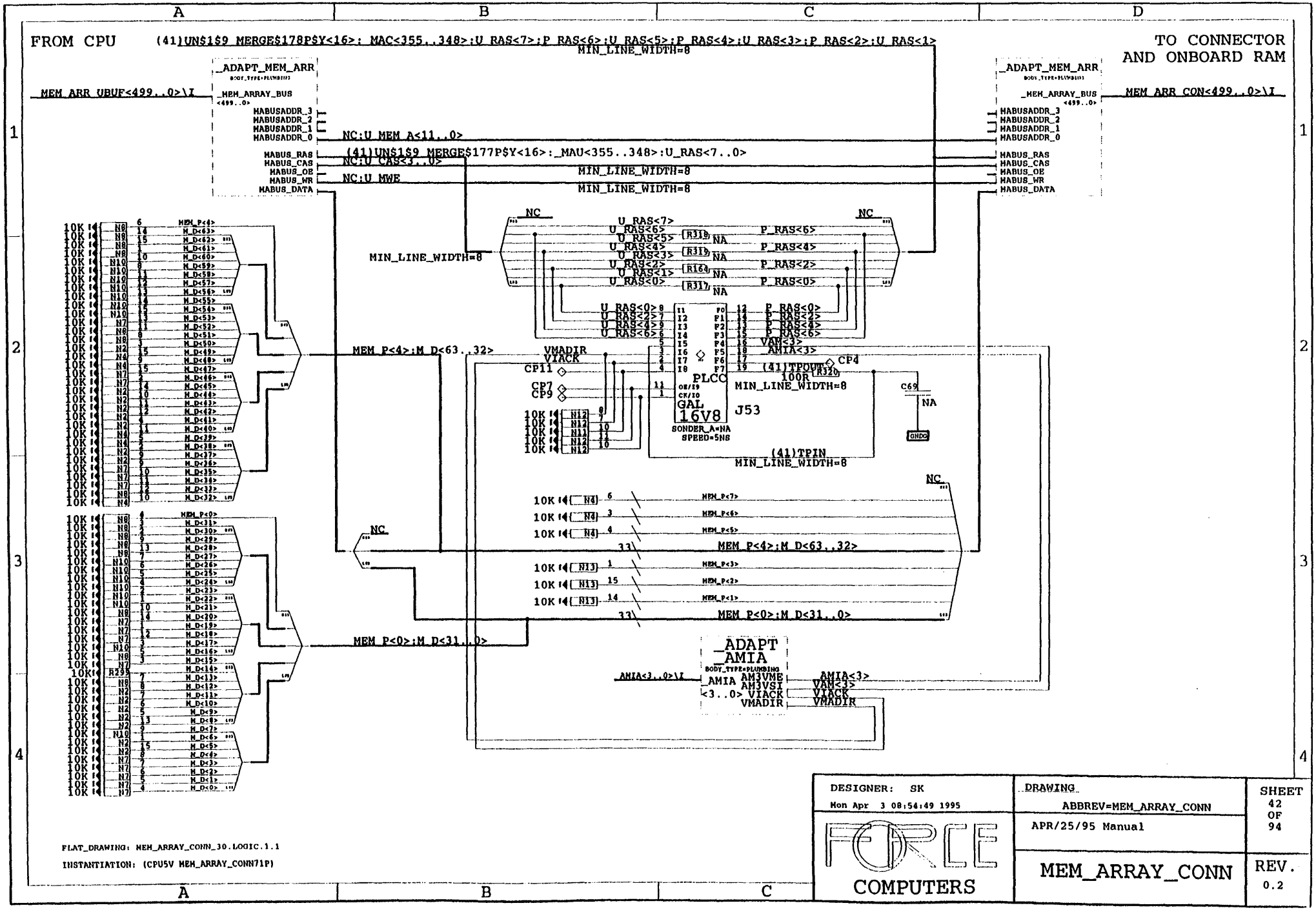
PLAT_DRAWING: VME_P0_CONN_59.LOGIC.1.1

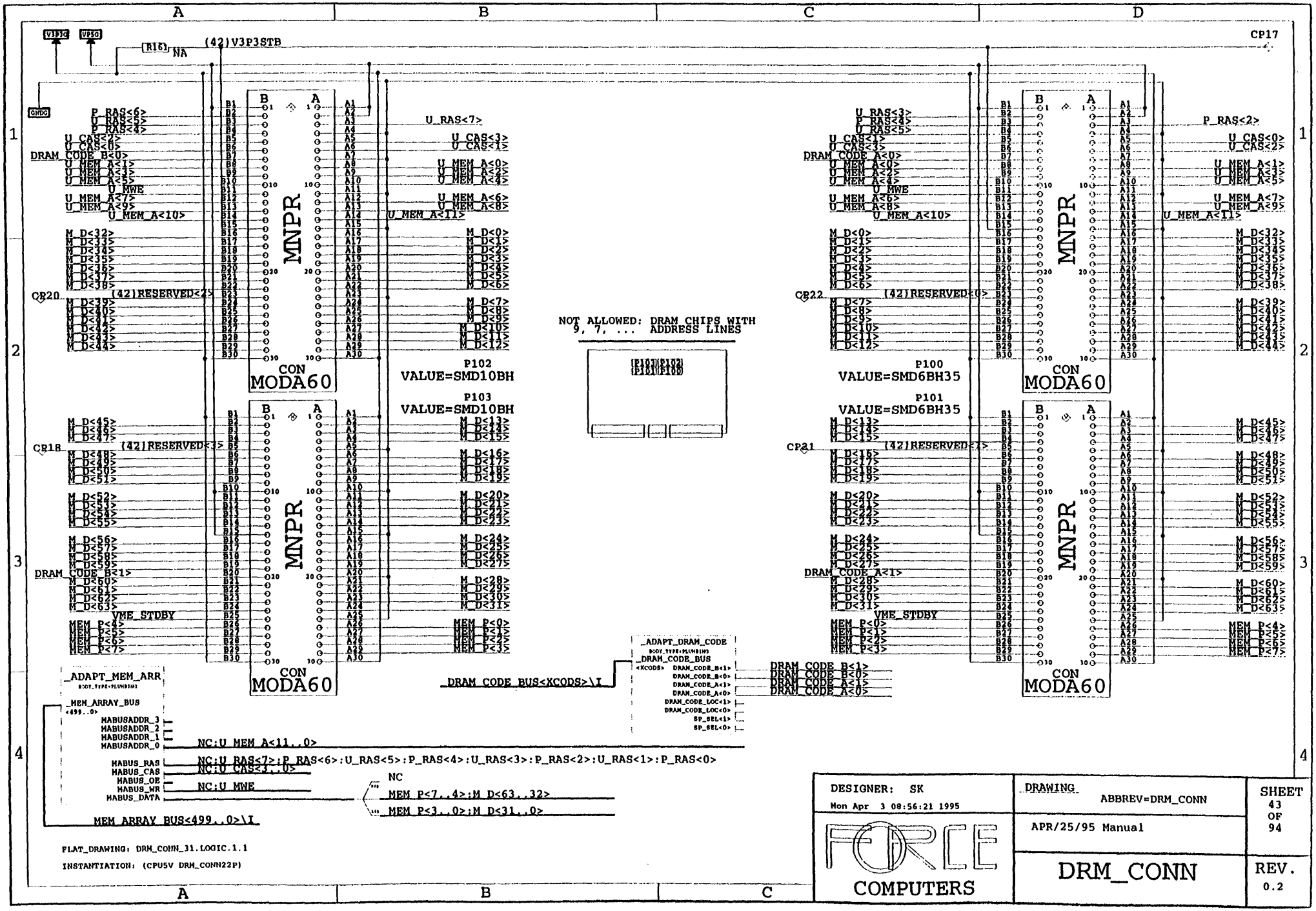
INSTANTIATION: (CPU5V VME_P0_CONN63P)

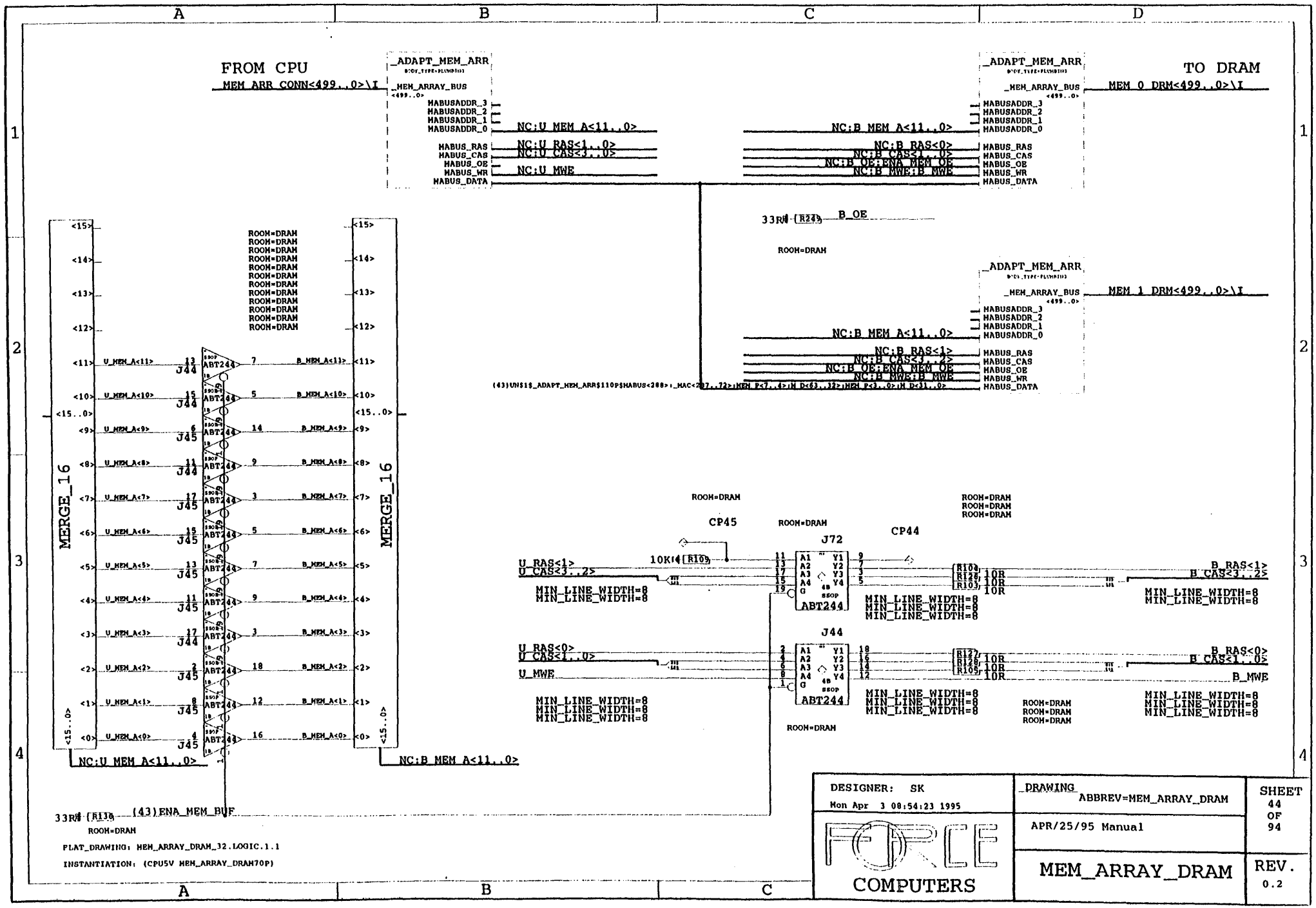
DESIGNER: Mon Apr 3 08:55:25 1995	DRAWING ABBREV=VME_P0_CONN	SHEET 39 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	VME_P0_CONN	REV. 0.2

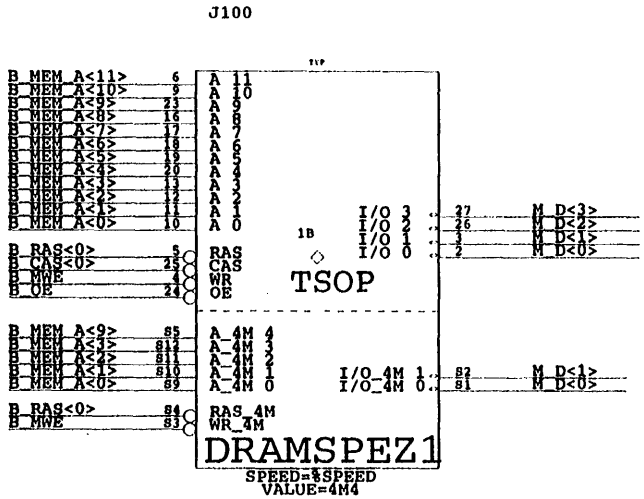

MNPR1
P1
VALUE=THRW
ADAPT_VMEBUS
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_BERCLK
VME_BERDAT
VME_DS<1..0>
VME_WRITE
VME_IACK
VME_AM<5..0>
VME_AS
VME_DTACK
VME_BERR
VME_RETRY
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BGIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLK
VME_GA<4..0>
VME_GAP
VME_BUS<210>
VME_BUS<214>
VME_BUS<212>
VME_BUS<211>
VME_BUS<213>
VME_BUS<YVMEC>\I
VME_P1_UIO<15..0>
VME_BUS<443..436>
VME_BUS<435..428>
FLAT_DRAWING: VME_P1_CONN_60.LOGIC.1.1
INSTANTIATION: (CPU5V VME_P1_CONN34P)
DESIGNER:
Mon Apr 3 08:55:27 1995
DRAWING
ABBREV=VME_P1_CONN
APR/25/95 Manual
SHEET 40 OF 94
REV. 0.2
COMPUTERS
VME_P1_CONN


MVME2
VALUE=THRW
P2
ADAPT_VMEBUS
VME_BUS <YVMEC>
VME_D<31..0>
VME_A<31..1>
VME_LWORD
VME_SERCLK
VME_SERDAT
VME_DS<1..0>
VME_WRITE
VME_IACK
VME_AM<5..0>
VME_AS
VME_DTACK
VME_BERR
VME_RETRY
VME_IRQ<7..1>
VME_IACKIN
VME_IACKOUT
VME_BR<3..0>
VME_BGIN<3..0>
VME_BGOUT<3..0>
VME_BBSY
VME_BCLR
VME_SYSRESET
VME_ACFAIL
VME_SYSFAIL
VME_SYSCLR
Z
A
B
C
D
Z1 KBD_PWR_CONN
Z3 SER_RXC_B_CONN
Z5 SER_RTXC_B_CONN
Z7 SER_GND_B_CONN
Z9 SER_TXC_B_CONN
Z11 SER_DTR_B_CONN
Z13 SER_DSR_B_CONN
Z15 SER_DCD_B_CONN
Z17 MSEOUT_CONN
Z19 SER_RXC_A_CONN
Z21 SER_RTXC_A_CONN
Z23 SER_GND_A_CONN
Z25 SER_TXC_A_CONN
Z27 SER_DTR_A_CONN
Z29 SER_DSR_A_CONN
Z31 SER_DCD_A_CONN
A1 SCSI_D<0>
A2 SCSI_D<1>
A3 SCSI_D<2>
A4 SCSI_D<3>
A5 SCSI_D<4>
A6 SCSI_D<5>
A7 SCSI_D<6>
A8 SCSI_D<7>
A9 SCSI_DP
A10 GND
A11 GND
A12 GND
A13 TERMPWR
A14 GND
A15 GND
A16 SCSI_ATN
A17 GND
A18 SCSI_BSY
A19 SCSI_ACK
A20 SCSI_RST
A21 SCSI_MSG
A22 SCSI_SEL
A23 SCSI_CD
A24 SCSI_REQ
A25 SCSI_IO
A26 MSEINP2
A27 KBDOUTP2
A28 KBDINP2
A29 SER_TXD_A_CONN
A30 SER_RXD_A_CONN
A31 RTS_DTR_A
A32 CTS_DCD_A
B3 VME_RETRY
B4 VME_A<24>
B5 VME_A<25>
B6 VME_A<26>
B7 VME_A<27>
B8 VME_A<28>
B9 VME_A<29>
B10 VME_A<30>
B11 VME_A<31>
B14 VME_D<16>
B15 VME_D<17>
B16 VME_D<18>
B17 VME_D<19>
B18 VME_D<20>
B19 VME_D<21>
B20 VME_D<22>
B21 VME_D<23>
B23 VME_D<24>
B24 VME_D<25>
B25 VME_D<26>
B26 VME_D<27>
B27 VME_D<28>
B28 VME_D<29>
B29 VME_D<30>
B30 VME_D<31>
C1 P2C1
C2 P2C2
C3 CENTR_DX<1>
C4 P2C4
C5 P2C5
C6 CENTR_DX<4>
C7 CENTR_DX<5>
C8 P2C8
C9 P2C9
C10 P2C10
C11 P2C11
C12 P2C12
C13 P2C13
C14 P2C14
C15 P2C15
C16 P2C16
C17 P2C17
C18 P2C18
C19 ETH_POW
C20 GND
C21 GND
C22 V_ETH_REC_P
C23 V_ETH_REC_M
C24 V_ETH_TRA_P
C25 V_ETH_TRA_M
C26 V_ETH_COL_P
C27 V_ETH_COL_M
C28 GND
C29 SER_TXD_B_CONN
C30 SER_RXD_B_CONN
C31 RTS_DTR_B
C32 CTS_DCD_B
D1 _VME_BUS<316>
D2 _VME_BUS<317>
D3 CENTR_DS
D4 GND
D5 CENTR_D<0>
D6 CENTR_D<1>
D7 CENTR_D<2>
D8 CENTR_D<3>
D9 CENTR_D<4>
D10 CENTR_D<5>
D11 CENTR_D<6>
D12 CENTR_D<7>
D13 CENTR_ACK
D14 CENTR_BSY
D15 CENTR_PE
D16 CENTR_AP
D17 CENTR_INIT
D18 CENTR_ERR
D19 CENTR_SLCT_IN
D20 CENTR_SLCT
D21 GND
D22 _VME_BUS<337>
D23 _VME_BUS<338>
D24 _VME_BUS<339>
D25 _VME_BUS<340>
D26 _VME_BUS<341>
D27 _VME_BUS<342>
D28 _VME_BUS<343>
D29 _VME_BUS<344>
D30 _VME_BUS<345>
D31 _VME_BUS<346>
D32 _VME_BUS<347>
CON
VGZABCD
VME_P1_UIO<15..8>
VME_P1_UIO<7..0>
VME_P0A<16..3>
VME_P0C<16..3>
VME_P2A<32..25>
VME_P2A<24..17>
VME_P2A<16..9>
VME_P2A<8..1>
VME_P2C<32..25>
VME_P2C<24..17>
VME_P2C<16..9>
VME_P2C<8..1>
VME_P2D<32..25>
VME_P2D<24..17>
VME_P2D<16..9>
VME_P2D<8..1>
VME_P2E<31..17:3>
VME_P2E<15..1:2>
CTS_DCD_A;RTS_DTR_A;SER_RXD_A_CONN;SER_TXD_A_CONN;KBDINP2;KBDOUTP2;MSEINP2;SCSI_IO
SCSI_REQ;SCSI_CD;SCSI_SEL;SCSI_MSG;SCSI_RST;SCSI_ACK;SCSI_BSY;GND
SCSI_ATN;GND;GND;TERMPWR;GND;GND;GND;SCSI_DP
SCSI_D<7..0>
CTS_DCD_B;RTS_DTR_B;SER_RXD_B_CONN;SER_TXD_B_CONN;GND;V_ETH_COL_H;V_ETH_COL_P;V_ETH_TRA_H
V_ETH_TRA_P;V_ETH_REC_H;V_ETH_REC_P;GND;GND;ETH_POW;P2C18;P2C17
P2C16;P2C15;P2C14;P2C13;P2C12;P2C11;P2C10;P2C9
P2C8;CENTR_DX<5..4>;P2C5;P2C4;CENTR_DX<1>;P2C2;P2C1
VME_BUS<347..340>
VME_BUS<339..337>;GND;CENTR_SLCT;CENTR_SLCT_IN;CENTR_ERR;CENTR_INIT
CENTR_AP;CENTR_PE;CENTR_BSY;CENTR_ACK;CENTR_D<7..4>
CENTR_D<3..0>;GND;CENTR_DS;_VME_BUS<317..316>
VME_P2E<31,29,27,25,23,21,19,17>
VME_P2E<15,13,11,9,7,5,3,1>
VME_BUS<YVMEC>\I
FLAT_DRAWING: VME_P2_CONN_61.LOGIC.1.1
INSTANTIATION: (CPU5V VME_P2_CONN33P)
DESIGNER:
Mon Apr 3 08:55:30 1995
DRAWING ABBREV=VME_P2_CONN
APR/25/95 Manual
SHEET 41 OF 94
REV. 0.2
FORCE COMPUTERS
VME_P2_CONN


FROM CPU
(41)UN$1$9_MERGE$178P$Y<16>:_MAC<355..348>:U_RAS<7>:P_RAS<6>:U_RAS<5>:P_RAS<4>:U_RAS<3>:P_RAS<2>:U_RAS<1>
MIN_LINE_WIDTH=8
TO CONNECTOR AND ONBOARD RAM
_ADAPT_MEM_ARR
MEM_ARR_UBUF<499..0>\I
_MEM_ARRAY_BUS <499..0>
MABUSADDR_3
MABUSADDR_2
MABUSADDR_1
MABUSADDR_0
MABUS_RAS
MABUS_CAS
MABUS_OE
MABUS_WR
MABUS_DATA
NC:U_MEM_A<11..0>
(41)UN$1$9_MERGE$177P$Y<16>:_MAU<355..348>:U_RAS<7..0>
NC:U_CAS<3..0>
NC:U_MWE
MEM_ARR_CON<499..0>\I
MEM_P<4>:M_D<63..32>
MEM_P<0>:M_D<31..0>
U_RAS<7>
U_RAS<6>
U_RAS<5>
U_RAS<4>
U_RAS<3>
U_RAS<2>
U_RAS<1>
U_RAS<0>
P_RAS<6>
P_RAS<4>
P_RAS<2>
P_RAS<0>
R318
R319
R164
R317
NA
NC
VMADIR
VIACK
CP11
CP7
CP9
CP4
PLCC
GAL 16V8
J53
SONDER_A=NA
SPEED=5NS
VAM<3>
AMIA<3>
(41)TPOUT
100R
R320
C69
GNDG
(41)TPIN
_ADAPT_AMIA
AMIA<3..0>\I
AMIA<3..0>
AM3VME
AM3VSI
VIACK
VMADIR
VAM<3>
FLAT_DRAWING: MEM_ARRAY_CONN_30.LOGIC.1.1
INSTANTIATION: (CPU5V MEM_ARRAY_CONN71P)
DESIGNER: SK
Mon Apr 3 08:54:49 1995
DRAWING
ABBREV=MEM_ARRAY_CONN
APR/25/95 Manual
SHEET 42 OF 94
FORCE COMPUTERS
MEM_ARRAY_CONN
REV. 0.2


V3P3G
V5P0G
CP17
(42)V3P3STB
R161
NA
CMOS
P_RAS<6>
U_RAS<5>
P_RAS<4>
U_CAS<2>
U_CAS<0>
DRAM_CODE_B<0>
U_MEM_A<1>
U_MEM_A<3>
U_MEM_A<5>
U_MWE
U_MEM_A<7>
U_MEM_A<9>
U_MEM_A<10>
M_D<32>
M_D<33>
M_D<34>
M_D<35>
M_D<36>
M_D<37>
M_D<38>
CR20
(42)RESERVED<2>
M_D<39>
M_D<40>
M_D<41>
M_D<42>
M_D<43>
M_D<44>
M_D<45>
M_D<46>
M_D<47>
CR18
(42)RESERVED<3>
M_D<48>
M_D<49>
M_D<50>
M_D<51>
M_D<52>
M_D<53>
M_D<54>
M_D<55>
M_D<56>
M_D<57>
M_D<58>
M_D<59>
DRAM_CODE_B<1>
M_D<60>
M_D<61>
M_D<62>
M_D<63>
VME_STDBY
MEM_P<4>
MEM_P<5>
MEM_P<6>
MEM_P<7>
B
A
MNPR
CON
MODA60
U_RAS<7>
U_CAS<3>
U_CAS<1>
U_MEM_A<0>
U_MEM_A<2>
U_MEM_A<4>
U_MEM_A<6>
U_MEM_A<8>
U_MEM_A<11>
M_D<0>
M_D<1>
M_D<2>
M_D<3>
M_D<4>
M_D<5>
M_D<6>
M_D<7>
M_D<8>
M_D<9>
M_D<10>
M_D<11>
M_D<12>
P102
VALUE=SMD10BH
P103
VALUE=SMD10BH
M_D<13>
M_D<14>
M_D<15>
M_D<16>
M_D<17>
M_D<18>
M_D<19>
M_D<20>
M_D<21>
M_D<22>
M_D<23>
M_D<24>
M_D<25>
M_D<26>
M_D<27>
M_D<28>
M_D<29>
M_D<30>
M_D<31>
MEM_P<0>
MEM_P<1>
MEM_P<2>
MEM_P<3>
NOT ALLOWED: DRAM CHIPS WITH
9, 7, ... ADDRESS LINES
P102
P103
P100
P101
U_RAS<3>
P_RAS<4>
U_RAS<5>
U_CAS<1>
U_CAS<3>
DRAM_CODE_A<0>
U_MEM_A<0>
U_MEM_A<2>
U_MEM_A<4>
U_MWE
U_MEM_A<6>
U_MEM_A<8>
U_MEM_A<10>
M_D<0>
M_D<1>
M_D<2>
M_D<3>
M_D<4>
M_D<5>
M_D<6>
CR22
(42)RESERVED<0>
M_D<7>
M_D<8>
M_D<9>
M_D<10>
M_D<11>
M_D<12>
P_RAS<2>
U_CAS<0>
U_CAS<2>
U_MEM_A<1>
U_MEM_A<3>
U_MEM_A<5>
U_MEM_A<7>
U_MEM_A<9>
U_MEM_A<11>
M_D<32>
M_D<33>
M_D<34>
M_D<35>
M_D<36>
M_D<37>
M_D<38>
M_D<39>
M_D<40>
M_D<41>
M_D<42>
M_D<43>
M_D<44>
M_D<45>
M_D<46>
M_D<47>
CP21
(42)RESERVED<1>
M_D<48>
M_D<49>
M_D<50>
M_D<51>
M_D<52>
M_D<53>
M_D<54>
M_D<55>
M_D<56>
M_D<57>
M_D<58>
M_D<59>
DRAM_CODE_A<1>
M_D<60>
M_D<61>
M_D<62>
M_D<63>
VME_STDBY
MEM_P<0>
MEM_P<1>
MEM_P<2>
MEM_P<3>
MEM_P<4>
MEM_P<5>
MEM_P<6>
MEM_P<7>
ADAPT_MEM_ARR
BODY_TYPE=PLUMBING
MEM_ARRAY_BUS
<499..0>
MABUSADDR_3
MABUSADDR_2
MABUSADDR_1
MABUSADDR_0
MABUS_RAS
MABUS_CAS
MABUS_OE
MABUS_WR
MABUS_DATA
MEM_ARRAY_BUS<499..0>\I
NC:U_MEM_A<11..0>
NC:U_RAS<7>:P_RAS<6>:U_RAS<5>:P_RAS<4>:U_RAS<3>:P_RAS<2>:U_RAS<1>:P_RAS<0>
NC:U_CAS<3..0>
NC:U_MWE
NC
MEM_P<7..4>:M_D<63..32>
MEM_P<3..0>:M_D<31..0>
DRAM_CODE_BUS<XCODS>\I
ADAPT_DRAM_CODE
BODY_TYPE=PLUMBING
DRAM_CODE_BUS
<XCODS>
DRAM_CODE_B<1>
DRAM_CODE_B<0>
DRAM_CODE_A<1>
DRAM_CODE_A<0>
DRAM_CODE_LOC<1>
DRAM_CODE_LOC<0>
SP_SEL<1>
SP_SEL<0>
DRAM_CODE_B<1>
DRAM_CODE_B<0>
DRAM_CODE_A<1>
DRAM_CODE_A<0>
FLAT_DRAWING: DRM_CONN_31.LOGIC.1.1
INSTANTIATION: (CPU5V DRM_CONN22P)
DESIGNER: SK
Mon Apr 3 08:56:21 1995
FORCE
COMPUTERS
DRAWING
ABBREV=DRM_CONN
APR/25/95 Manual
DRM_CONN
SHEET
43
OF
94
REV.
0.2


FROM CPU
MEM_ARR_CONN<499..0>\I
_ADAPT_MEM_ARR
_MEM_ARRAY_BUS
MABUSADDR_3
MABUSADDR_2
MABUSADDR_1
MABUSADDR_0
MABUS_RAS
MABUS_CAS
MABUS_OE
MABUS_WR
MABUS_DATA
NC:U_MEM_A<11..0>
NC:U_RAS<1..0>
NC:U_CAS<3..0>
NC:U_MWE
NC:B_MEM_A<11..0>
NC:B_RAS<0>
NC:B_CAS<1..0>
NC:B_OE:ENA_MEM_OE
NC:B_MWE:B_MWE
TO DRAM
MEM_0_DRM<499..0>\I
MEM_1_DRM<499..0>\I
33R# (R224)
B_OE
ROOM=DRAM
MERGE_16
U_MEM_A<11>
U_MEM_A<10>
U_MEM_A<9>
U_MEM_A<8>
U_MEM_A<7>
U_MEM_A<6>
U_MEM_A<5>
U_MEM_A<4>
U_MEM_A<3>
U_MEM_A<2>
U_MEM_A<1>
U_MEM_A<0>
B_MEM_A<11>
B_MEM_A<10>
B_MEM_A<9>
B_MEM_A<8>
B_MEM_A<7>
B_MEM_A<6>
B_MEM_A<5>
B_MEM_A<4>
B_MEM_A<3>
B_MEM_A<2>
B_MEM_A<1>
B_MEM_A<0>
J44
J45
ABT244
CP45
CP44
J72
10K# (R109)
U_RAS<1>
U_CAS<3..2>
U_RAS<0>
U_CAS<1..0>
U_MWE
B_RAS<1>
B_CAS<3..2>
B_RAS<0>
B_CAS<1..0>
B_MWE
10R
MIN_LINE_WIDTH=8
(43)ENA_MEM_BUF
33R# (R136)
PLAT_DRAWING: MEM_ARRAY_DRAM_32.LOGIC.1.1
INSTANTIATION: (CPU5V MEM_ARRAY_DRAM70P)
DESIGNER: SK
Mon Apr 3 08:54:23 1995
DRAWING
ABBREV=MEM_ARRAY_DRAM
APR/25/95 Manual
SHEET 44 OF 94
FORCE COMPUTERS
MEM_ARRAY_DRAM
REV. 0.2

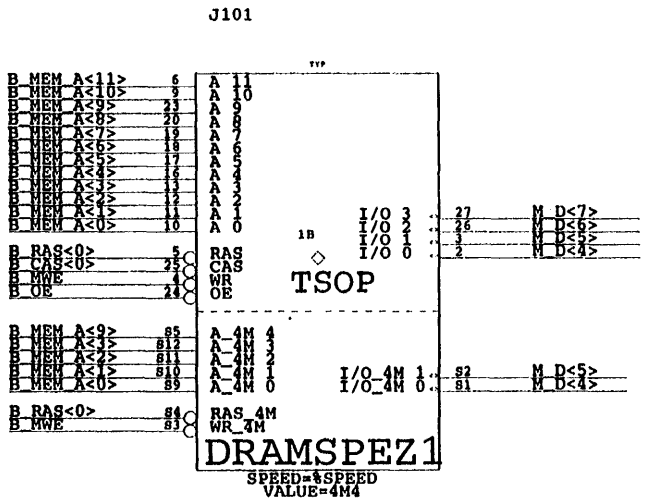


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_67.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:32 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 45 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

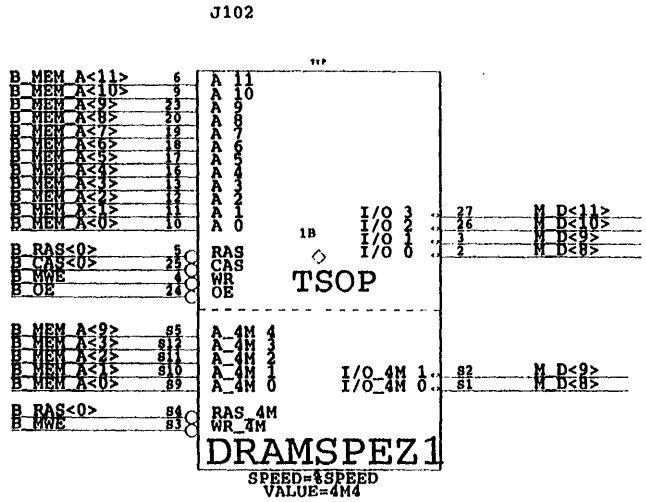


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_68.LOGIC.0.1

INSTANTIATION: (CPU5V DRM75P#1 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:04 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 46 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

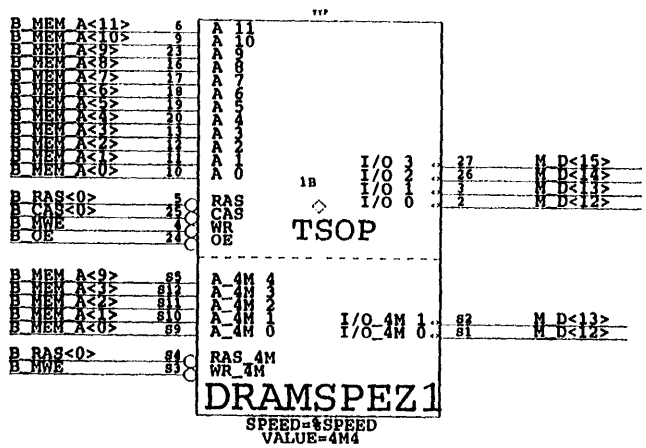


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_77.LOGIC.8.1

INSTANTIATION: (CPU5V DRH75P82 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:05 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 47 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

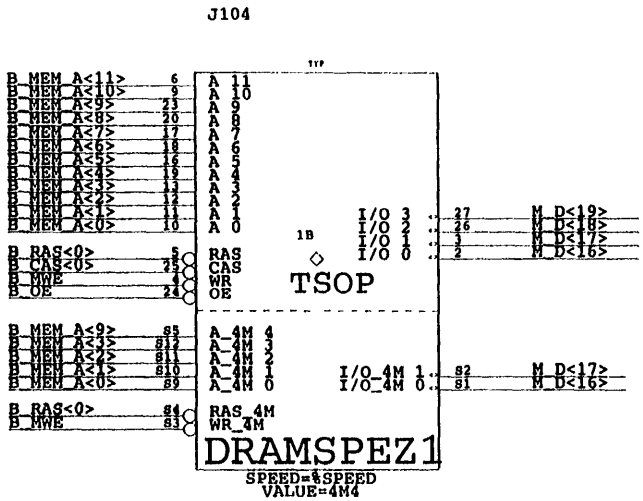


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_78.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P#3 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:10 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 48 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

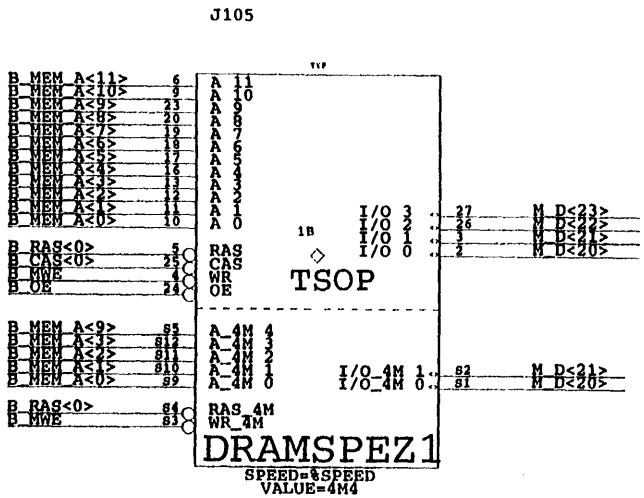


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_79.LOGIC.0.1

INSTANTIATION: (CPU5V DRH75P#4 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:11 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 49 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

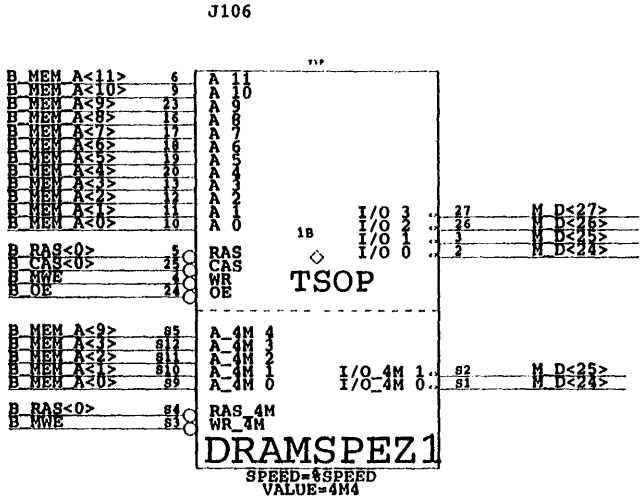


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_80.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P85 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:12 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 50 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

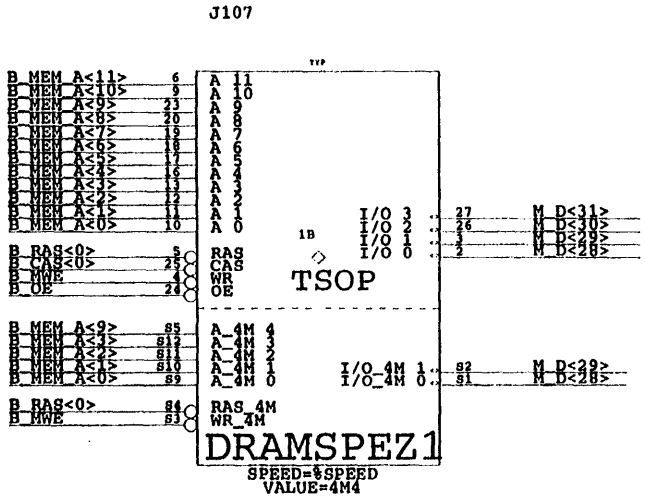


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_81.LOGIC.0.1

INSTANTIATION: (CPU5V DRM75P#6 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:13 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 51 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

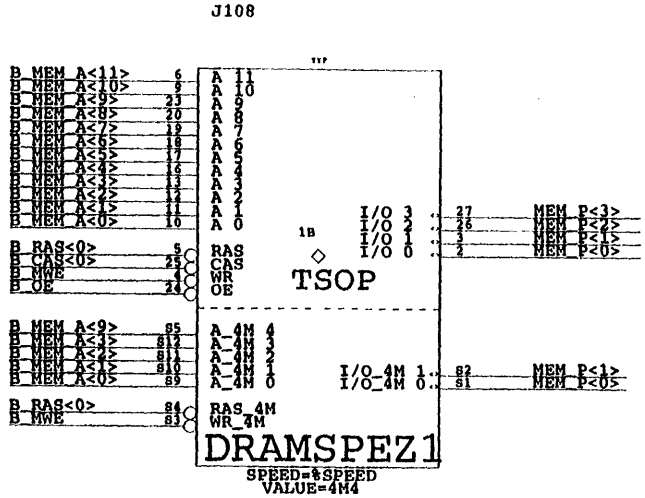


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_82.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P87 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:14 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 52 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

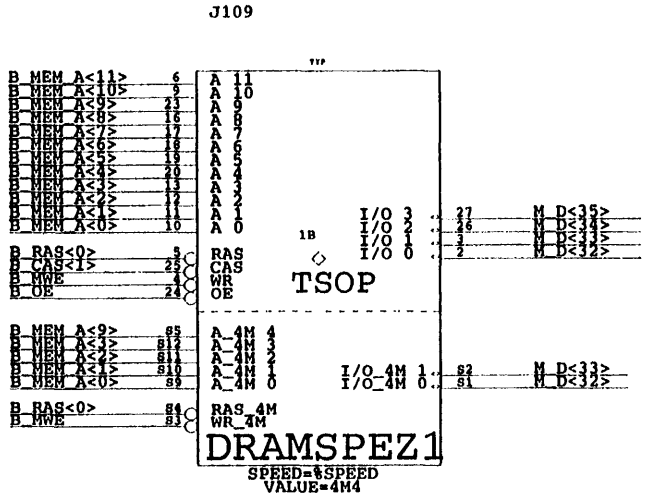


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_03.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P00 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:19 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 53 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

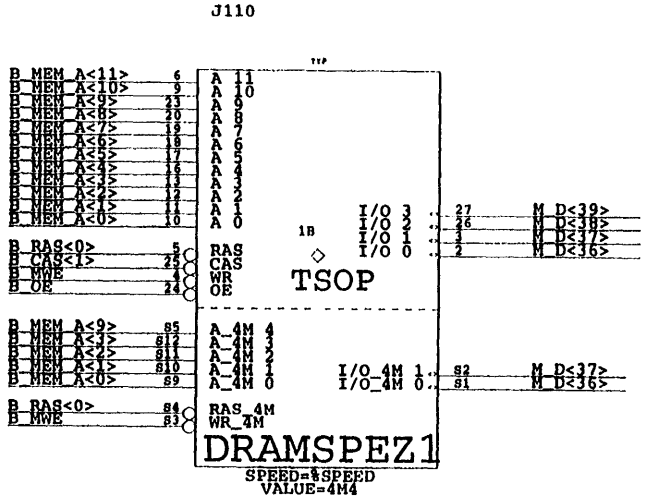


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_84.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P89 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:25 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 54 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

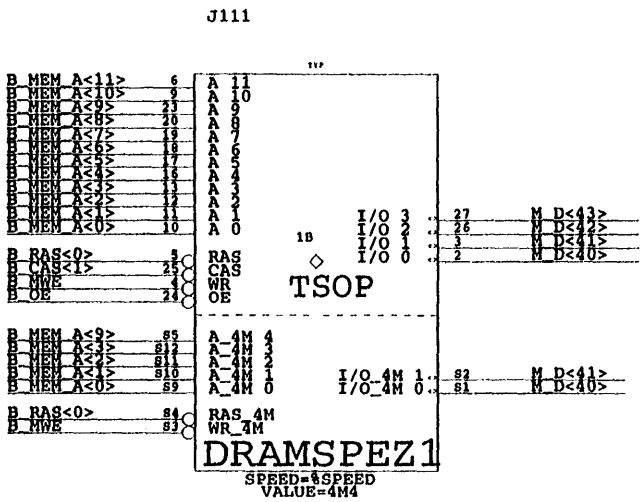


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_69.LOGIC.0.1

INSTANTIATION: (CPU5V DRM75P#10 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:41 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 55 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

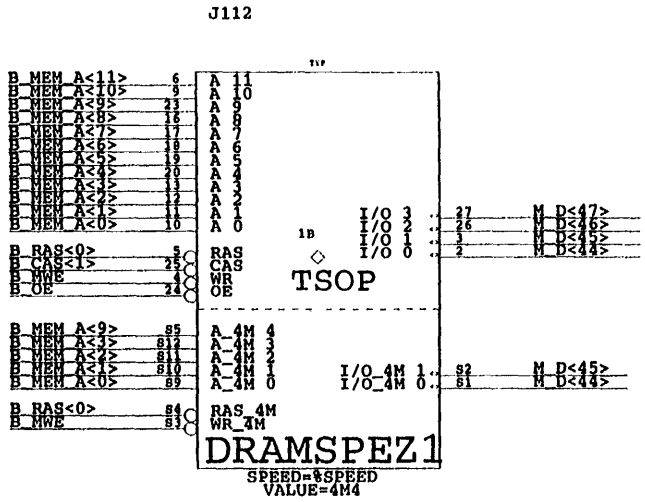


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_70.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P#11 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:43 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 56 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

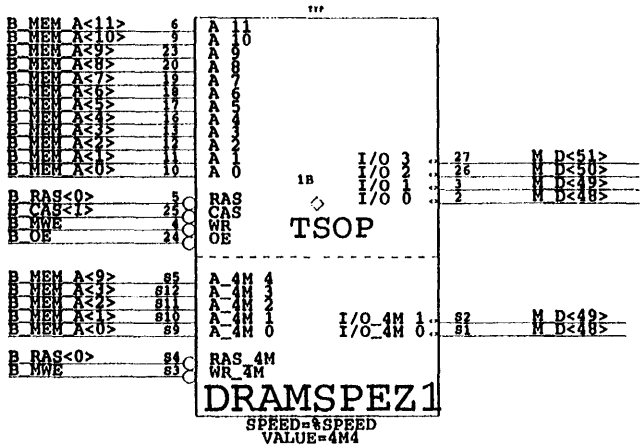


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_71.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P#12 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:46 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 57 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

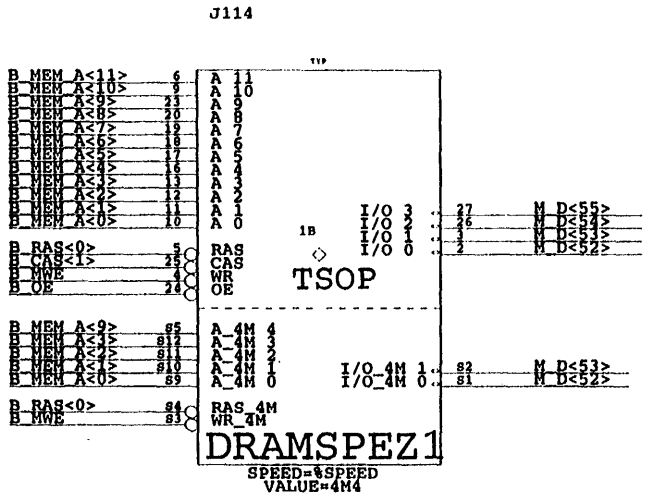


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_72.LOGIC.0.1

INSTANTIATION: (CPU5V DRM75P813 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:54 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 58 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

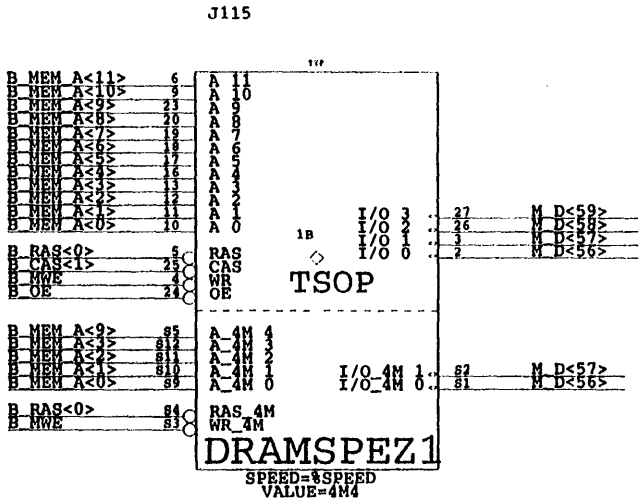


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_73.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P#14 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:54 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 59 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

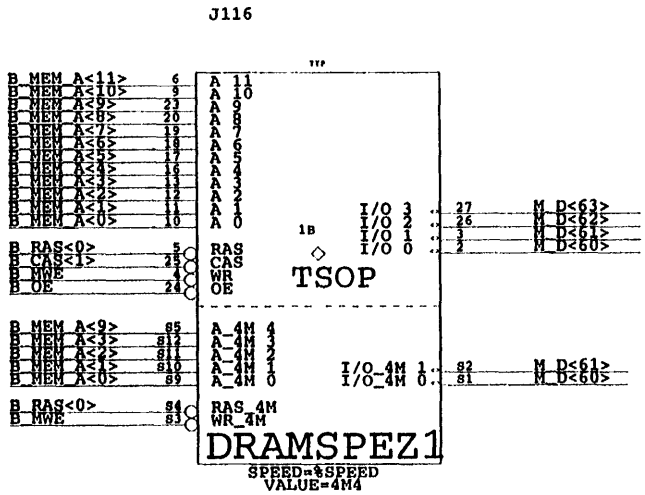


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_74.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P#15 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:55 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 60 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

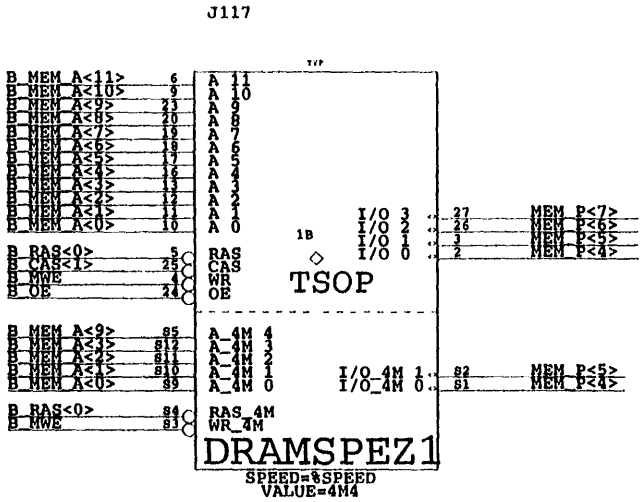


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_75.LOGIC.8.1

INSTANTIATION: (CPU5V DRM75P816 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:56 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 61 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

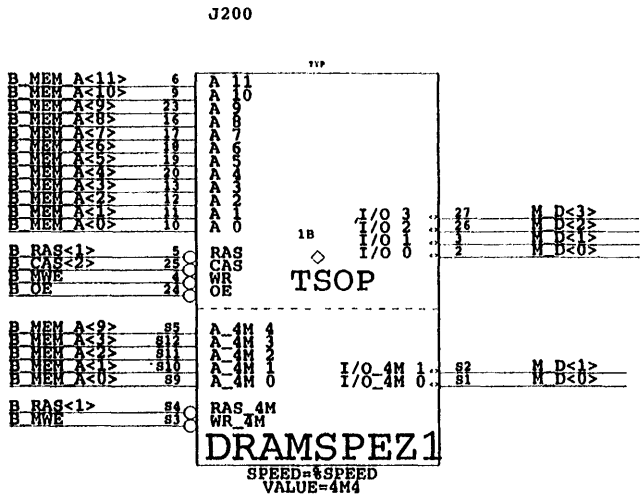


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_76.LOGIC.0.1

INSTANTIATION: (CPU5V DRM75P#17 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:00 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 62 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

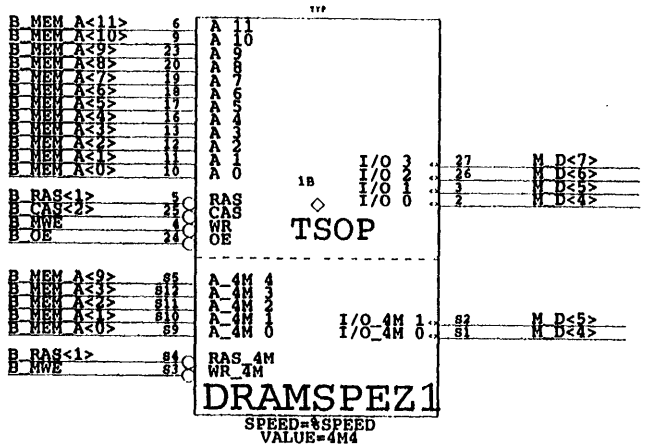


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_33.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:24 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 63 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

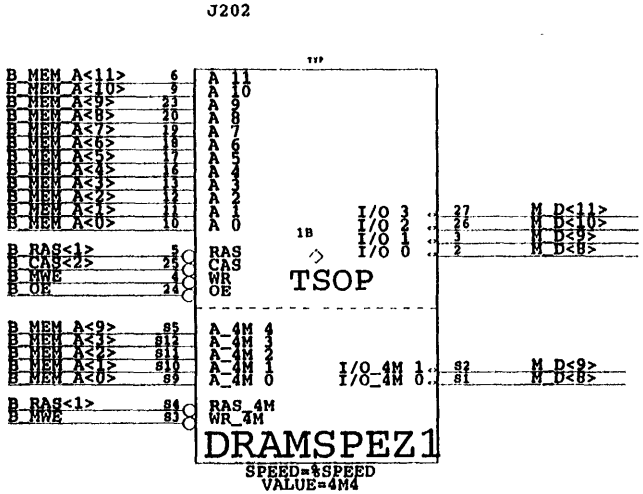


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_34.LOGIC.0.1

INSTANTIATION: (CPU5V DRH21P01 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:53 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 64 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

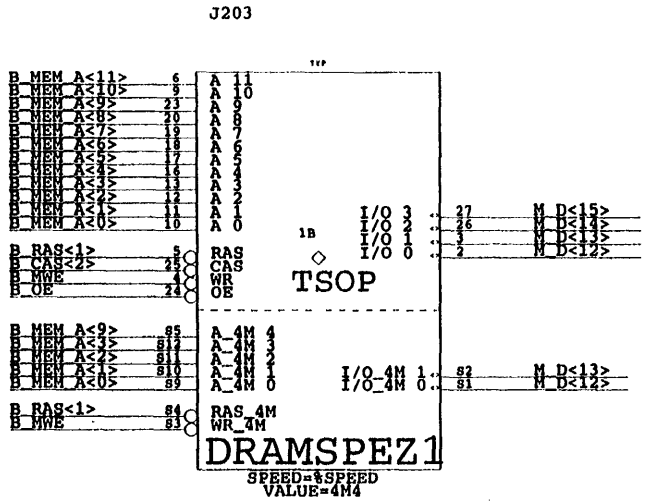


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_43.LOGIC.8.1

INSTANTIATION: (CPU5V DRH21P@2 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:54 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 65 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

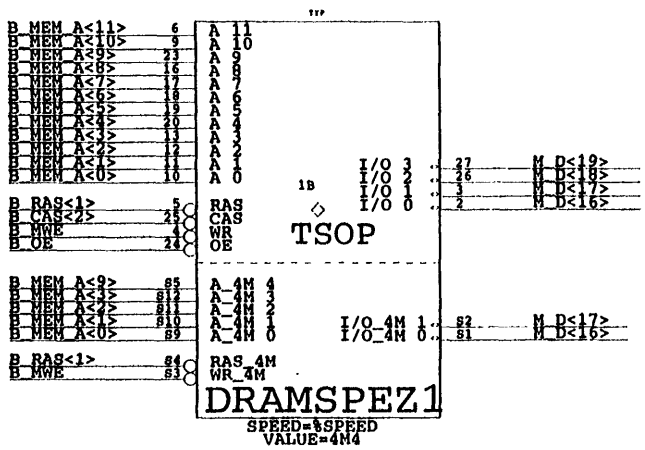


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_44.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P83 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:55 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 66 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

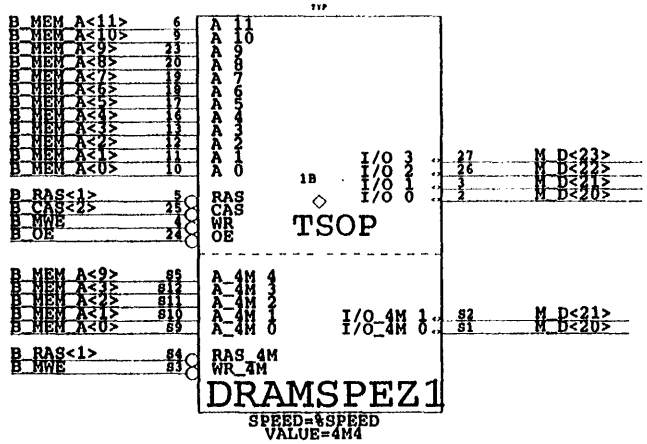


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_45.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P84 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:56 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 67 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

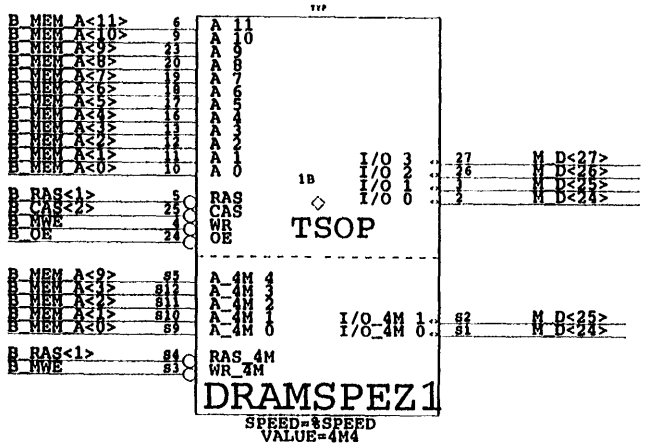


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_46.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P85 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:55:57 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 68 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

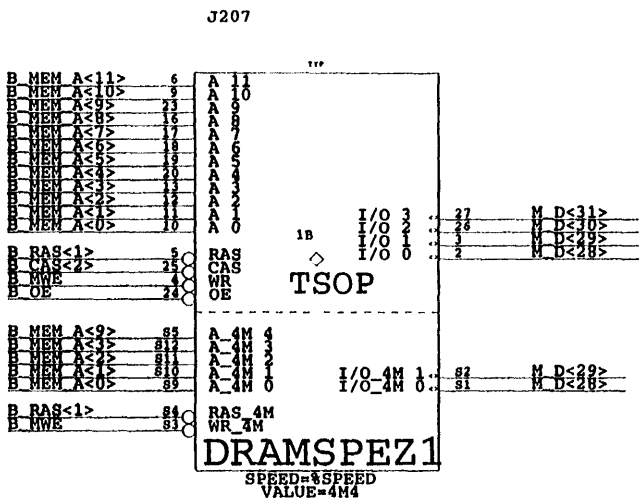


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_47.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P#6 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 00:56:01 1995	_DRAWING ABBREV=DRAM_CHIP	SHEET 69 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

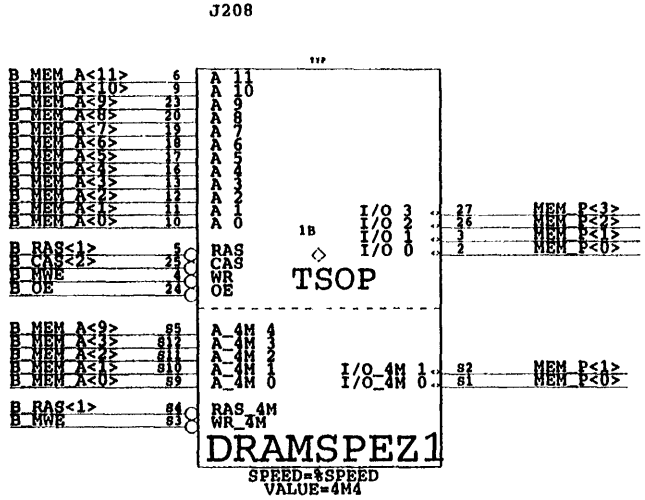


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_48.LOGIC.8.1

INSTANTIATION: (CPU5V DRH21P87 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:01 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 70 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

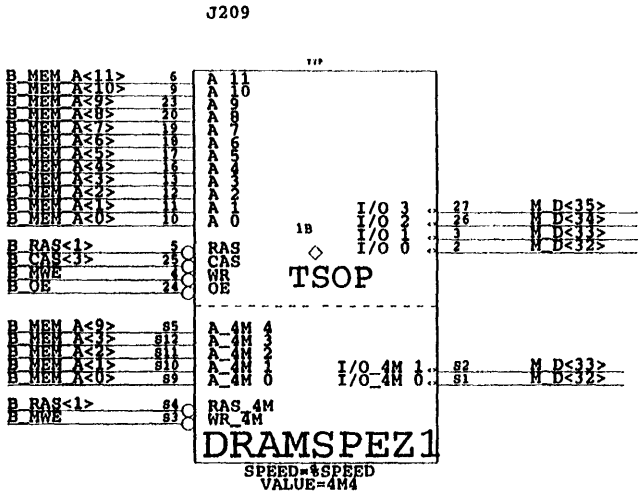


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_49.LOGIC.0.1

INSTANTIATION: (CPU5V DRM21P#0 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:02 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 71 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

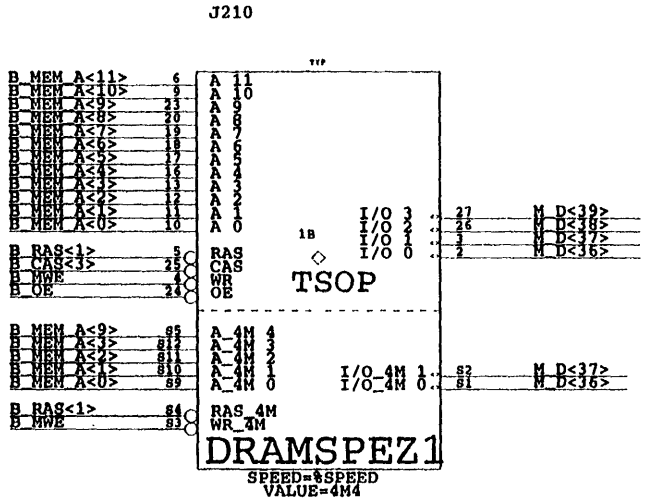


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_50.LOGIC.0.1

INSTANTIATION: (CPU5V DRM21P#9 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:56:03 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 72 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

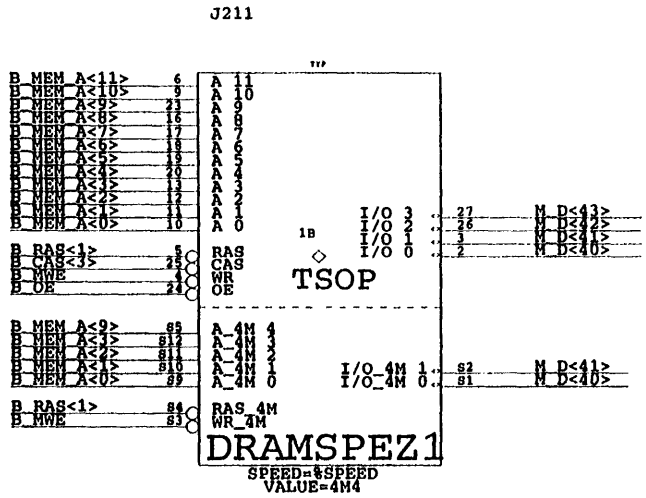


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_35.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P#10 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:27 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 73 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

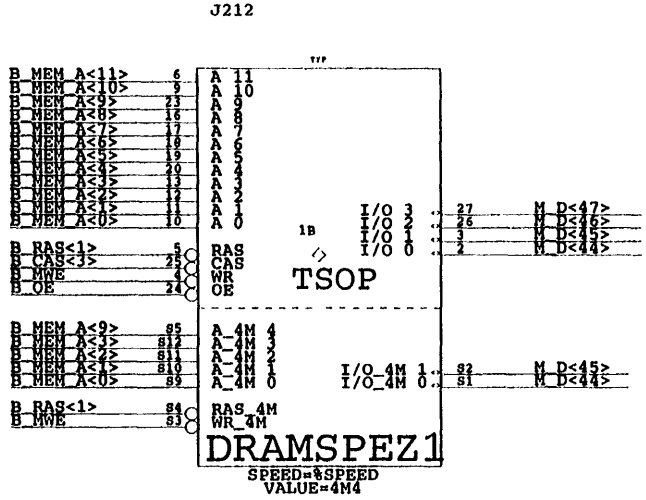


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_36.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P#11 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:30 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 74 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

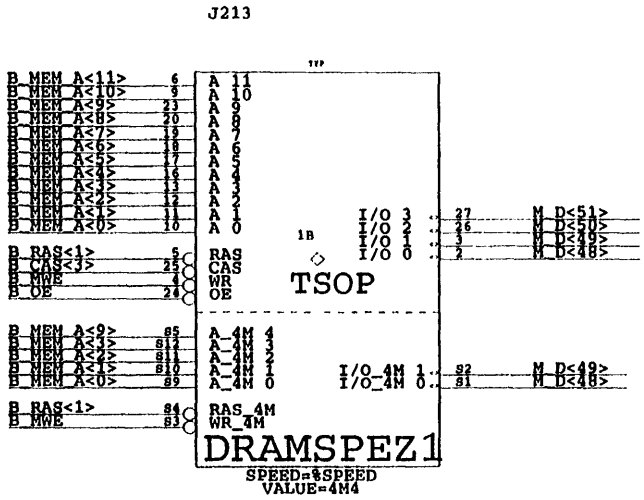


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_37.LOGIC.0.1

INSTANTIATION: (CPU5V DRM21P812 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:35 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 75 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

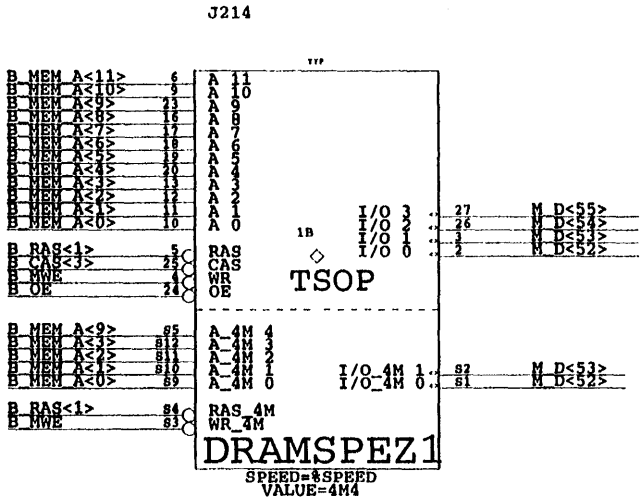


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_30.LOGIC.0.1

INSTANTIATION: (CPU5V DRM21P#13 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:36 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 76 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2



EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_39.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P#14 DRAM_CHIP3P)

DESIGNER: IV

Mon Apr 3 08:54:36 1995

DRAWING ABBREV=DRAM_CHIP

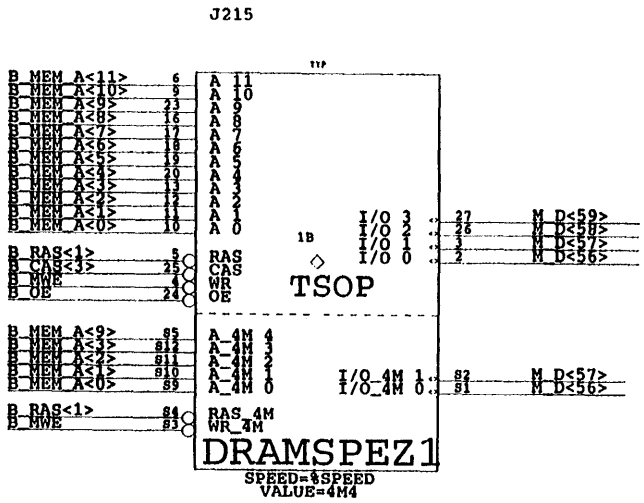
APR/25/95 Manual

SHEET 77 OF 94

FORCE COMPUTERS

DRAM_CHIP

REV. 0.2

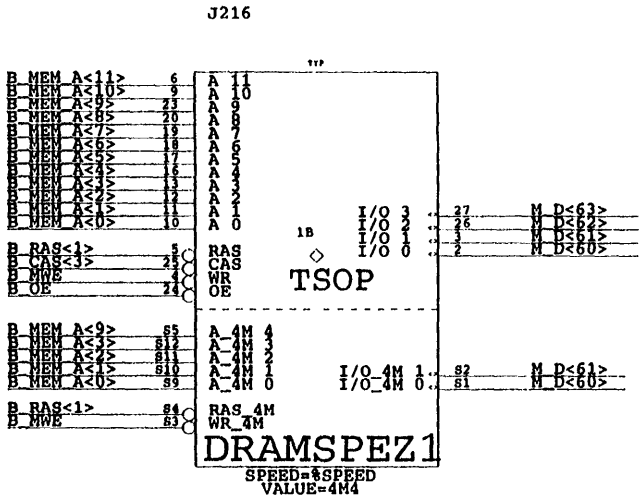


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_40.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P#15 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:37 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 78 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2



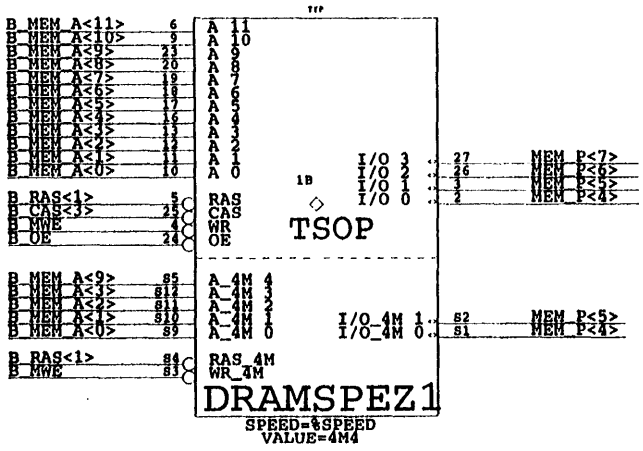
EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_41.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P816 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:38 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 79 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

J217

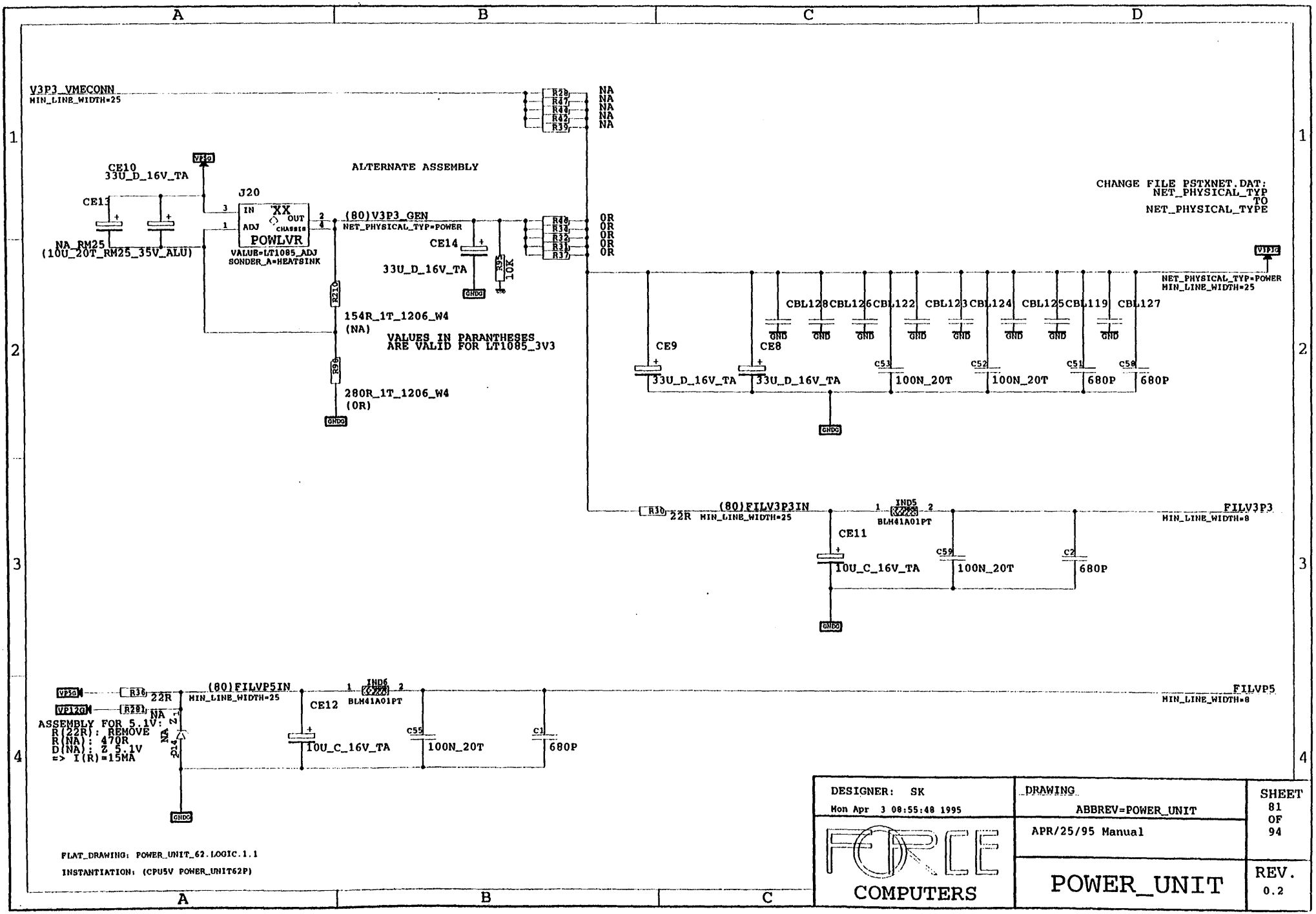


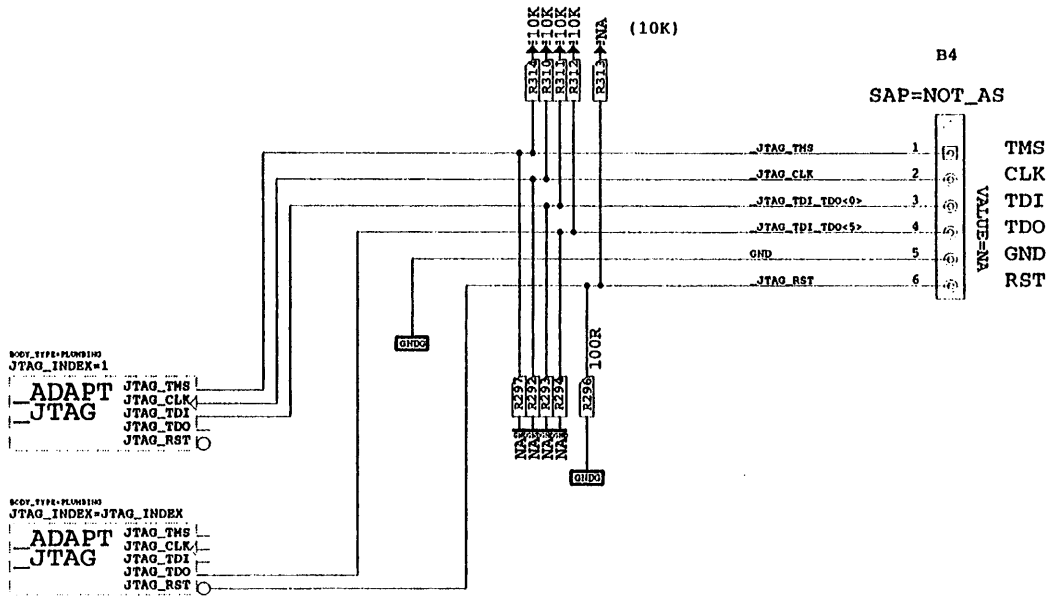
EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_42.LOGIC.8.1

INSTANTIATION: (CPU5V DRM21P817 DRAM_CHIP3P)

DESIGNER: IV Mon Apr 3 08:54:39 1995	DRAWING ABBREV=DRAM_CHIP	SHEET 80 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	DRAM_CHIP	REV. 0.2

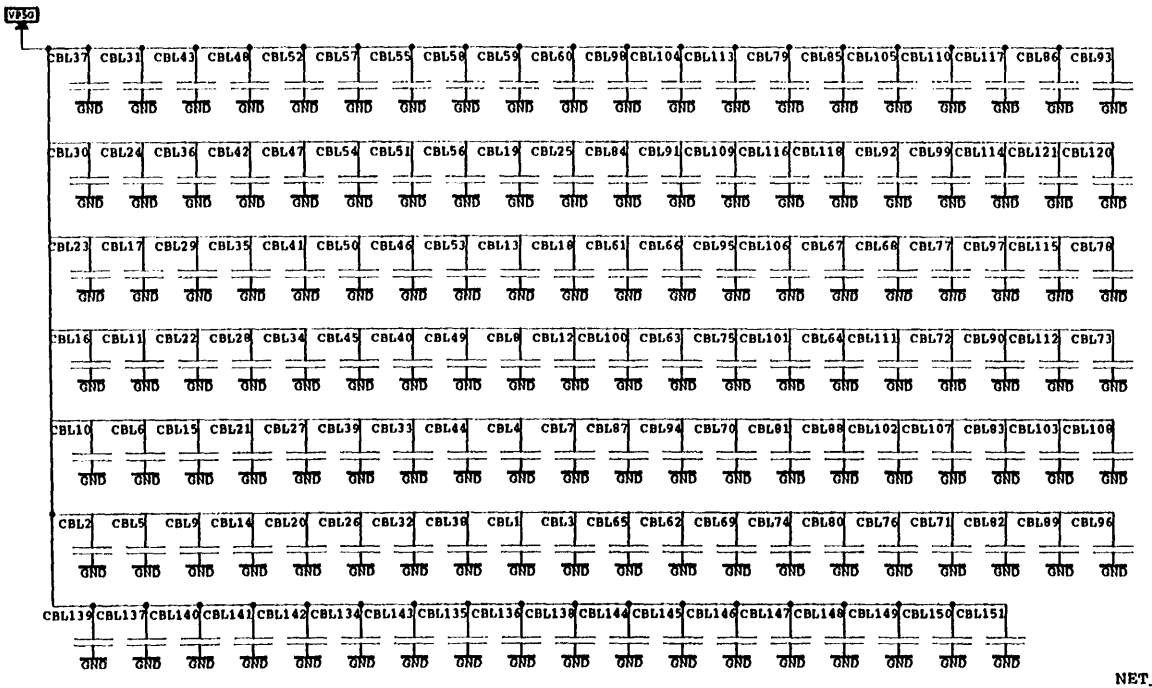

V3P3_VMECONN
MIN_LINE_WIDTH=25
R28 R47 R43 R42 R39
NA NA NA NA NA
ALTERNATE ASSEMBLY
CE10
33U_D_16V_TA
CE13
J20
IN
ADJ
XX
OUT
CHASSIS
POWLVR
VALUE=LT1085_ADJ
SONDER_A=HEATSINK
NA_RM25
(10U_20T_RM25_35V_ALU)
(80)V3P3_GEN
NET_PHYSICAL_TYP=POWER
R40 R34 R32 R31 R37
0R 0R 0R 0R 0R
CE14
33U_D_16V_TA
R95
10K
R210
154R_1T_1206_W4
(NA)
VALUES IN PARANTHESES
ARE VALID FOR LT1085_3V3
R99
280R_1T_1206_W4
(0R)
GNDQ
CHANGE FILE PSTXNET.DAT:
NET_PHYSICAL_TYP
TO
NET_PHYSICAL_TYPE
V3P3G
NET_PHYSICAL_TYP=POWER
MIN_LINE_WIDTH=25
CBL128 CBL126 CBL122 CBL123 CBL124 CBL125 CBL119 CBL127
GND
CE9
33U_D_16V_TA
CE8
33U_D_16V_TA
C53
100N_20T
C52
100N_20T
C51
680P
C58
680P
R30
22R
(80)FILV3P3IN
MIN_LINE_WIDTH=25
IND5
BLM41A01PT
FILV3P3
MIN_LINE_WIDTH=8
CE11
10U_C_16V_TA
C59
100N_20T
C2
680P
VP5G
R36
22R
VP12G
R28
NA
ASSEMBLY FOR 5.1V:
R(22R): REMOVE
R(NA): 470R
D(NA): Z 5.1V
=> I(R)=15MA
D14
(80)FILVP5IN
MIN_LINE_WIDTH=25
IND6
BLM41A01PT
CE12
10U_C_16V_TA
C55
100N_20T
C1
680P
FILVP5
MIN_LINE_WIDTH=8
FLAT_DRAWING: POWER_UNIT_62.LOGIC.1.1
INSTANTIATION: (CPU5V POWER_UNIT62P)
DESIGNER: SK
Mon Apr 3 08:55:48 1995
DRAWING
ABBREV=POWER_UNIT
APR/25/95 Manual
SHEET
81
OF
94
COMPUTERS
POWER_UNIT
REV.
0.2

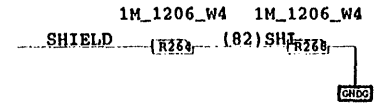


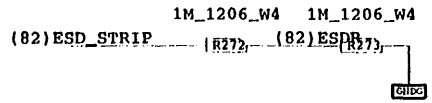
FLAT_DRAWING: JTAG_UNIT_63.LOGIC.1.1

INSTANTIATION: (CPU5V JTAG_UNIT57P)

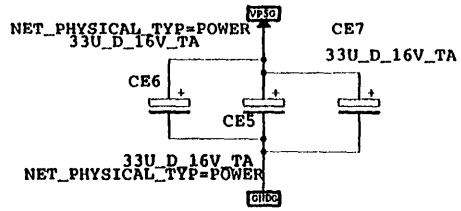
DESIGNER: GP/IV Mon Apr 3 08:55:46 1995	DRAWING JTAG_UNIT JTAG_UNIT	SHEET 82 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	JTAG_UNIT	REV. 0.2

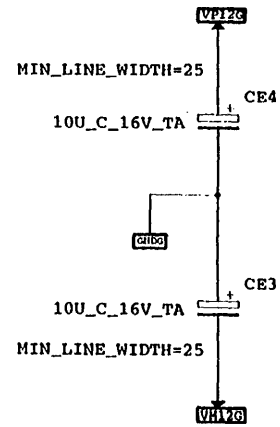


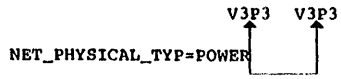




VME_STDBY SEE VME_P1_CONN PAGE
V3P3_VMECONN SEE POWER_UNIT PAGE
FILVP5 SEE POWER_UNIT PAGE
FILV3P3 SEE POWER_UNIT PAGE

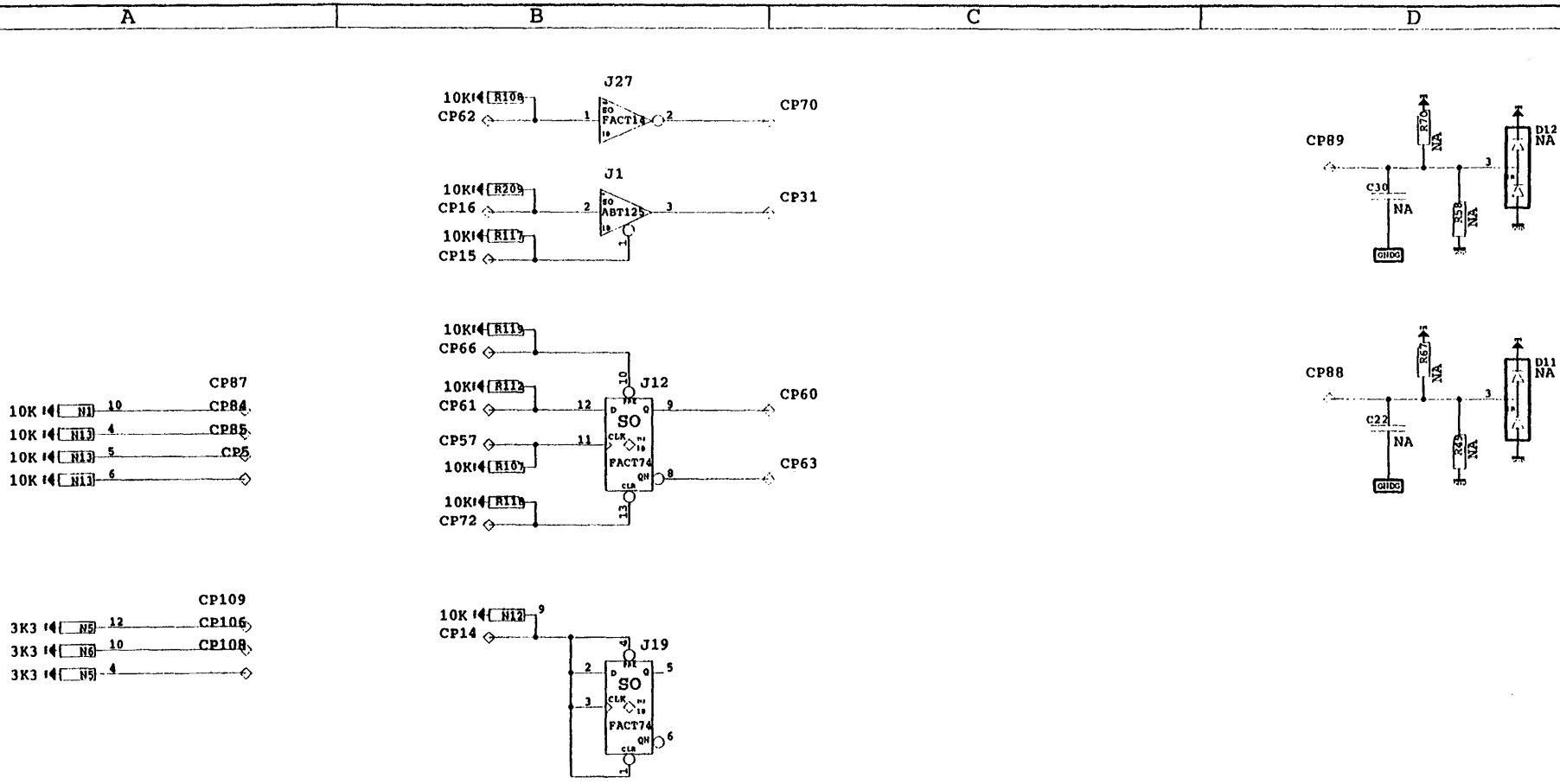






FLAT_DRAWING: DECOUPLING_64.LOGIC.1.1
INSTANTIATION: (CPU5V DECOUPLING55P)

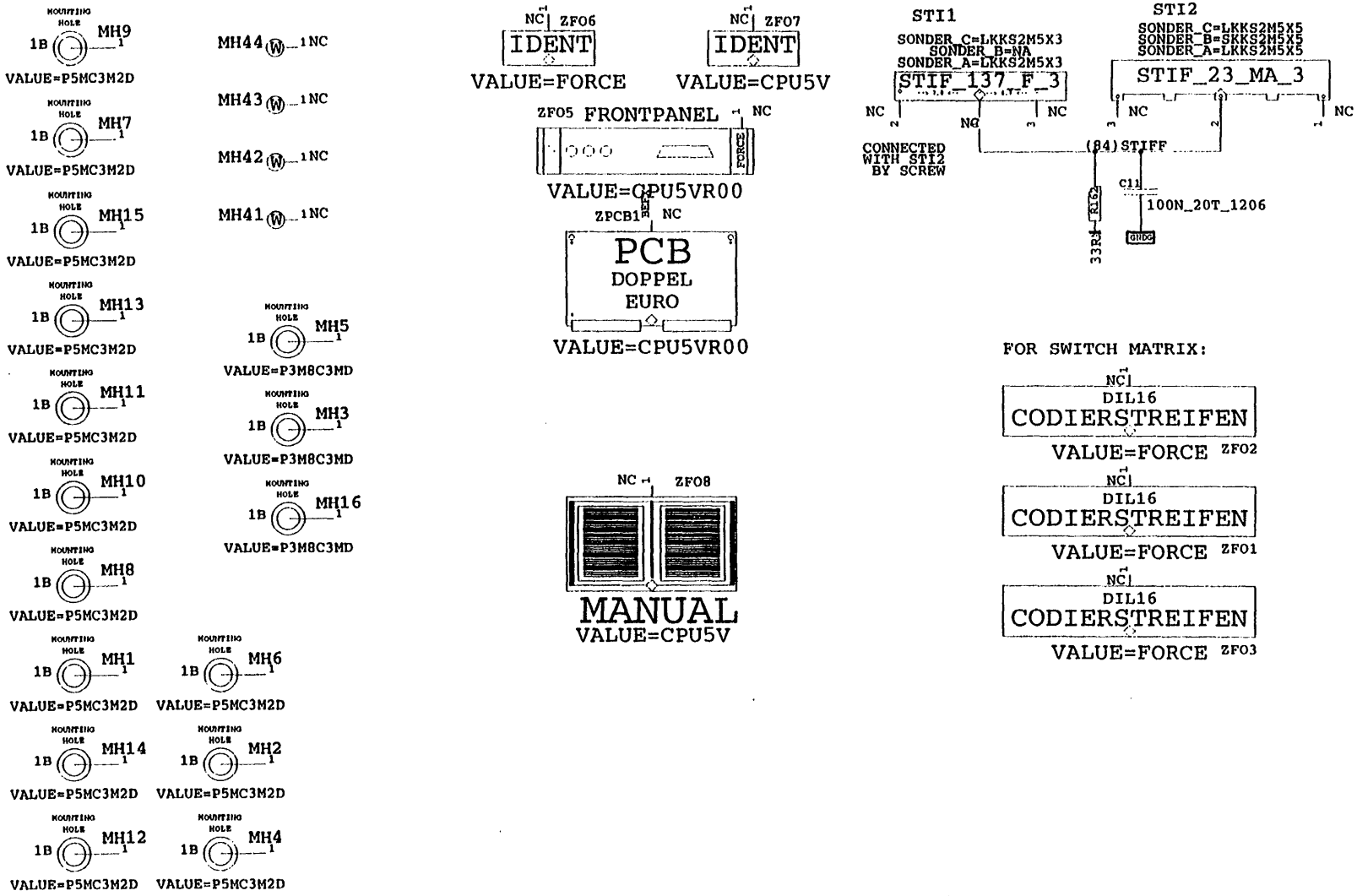
DESIGNER: SK Mon Apr 3 00:54:59 1995	DRAWING ABBREV=DECOUPLING	SHEET 83 OF 94
COMPUTERS	APR/25/95 Manual	
	DECOUPLING	REV. 0.2



FLAT_DRAWING: VARIOUS_UNIT_65.LOGIC.1.1

INSTANTIATION: (CPU5V VARIOUS_UNIT56P)

DESIGNER: SK Mon Apr 3 08:55:59 1995	DRAWING ABBREV=VARIOUS_UNIT	SHEET 84 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	VARIOUS_UNIT	REV. 0.2



FLAT_DRAWING: MECHANIC_66.LOGIC.1.1

INSTANTIATION: (CPU5V MECHANIC54P)

DESIGNER: SK Mon Apr 3 08:56:07 1995	DRAWING ABBREV=MECHANIC	SHEET 85 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	MECHANIC	REV. 0.2

```
      *** Signal Cross-Reference ***
        --- for the entire design --

(1)CPU_CLK <2..0> 2.2B
(1)CPU_PA <27..24> 2.1B 2.4A 2.4B
(1)DIV_CNTL <1..0> 2.3A
(1)EXT_CLK2       2.3B
(1)EXT_EV         2.3D
(1)PLL_BYP        2.3B
(1)P_REPLY <1..0> 2.2D
(1)SLVSEL <4..3> 2.2B 2.4A 2.4B
(1)STANDBY        2.2B
(2)CLKHALF        3.2B
(2)CLK_CPU_QUOH 3.2B
(2)LOOP           3.3B
(8)TERMFUSE       9.2A
(9)ENATERM_BACK 10.2C
(10)ENATERM_FRONT 11.2C
(10)RTERMPWR     11.2C 11.3C
(10)SCSI_FP_ACK 11.2C 11.3B
(10)SCSI_FP_ATN 11.2C 11.3B
(10)SCSI_FP_BSY 11.2C 11.3B
(10)SCSI_FP_CD  11.3B 11.3C
(10)SCSI_FP_D <7..0> 11.1C 11.2B 11.2C 11.3B
(10)SCSI_FP_DP  11.2C 11.3B
(10)SCSI_FP_IO  11.3B 11.3C
(10)SCSI_FP_MSG 11.3B 11.3C
(10)SCSI_FP_REQ 11.3B 11.3C
(10)SCSI_FP_RST 11.3B 11.3C
(10)SCSI_FP_SEL 11.3B 11.3C
(11)ENET_CH       12.2B
(11)ENET_CP       12.2B
(11)ENET_RH       12.2B
(11)ENET_RP       12.2B
(11)ENET_TH       12.2B
(11)ENET_TP       12.2B
(11)ETHPF         12.3B
(11)ETHRF         12.3B
(11)ETHX1         12.3B
(11)ETHX2         12.3B
(11)GND_ETH       12.1C 12.3A 12.3C 12.4A
(11)TSEL          12.3B
(11)VP5_ETH       12.3A 12.3B 12.4A
(14)CENTR_ENA_R 15.3A
(14)C_ACKINV      15.3D
(14)C_BSYINV      15.2D
(14)C_DSINV       15.1D
(14)P_ACKINV      15.3C
(14)P_BSYINV      15.2C
(14)P_DSINV       15.1C
(15)LINK_TEST_EN 16.3C
(15)HSI_IRQ_PU    16.3C
(16)CLK_BUF_32    17.3C
(16)LOOP3         17.4C
(17)ENA_LOC_BUFF 18.3B
(18)LCA_MODE      19.3C
(18)LCA_MODE_R    19.3B
(18)RESERVED_SW8_H 19.2C
(20)BOOTEPR_HR    21.1B
(20)BOOTWRITE     21.1C 21.3C
(20)BOOT_ENA      21.2A
(20)VPP_ADDR      21.1C 21.3C
(21)POWER_DOWN    22.1C 22.2B 22.2C
(21)SYSEPR_HR     22.1C 22.2B 22.2C
(21)SYS_ENA       22.3A
(22)NCA           24.3C
(22)NCB           24.4C
```

```
(23)NCA           23.3C
(23)NCB           23.4C
(25)KBDIN         26.1B
(25)KBDOUT        26.1B
(25)MSEIN         26.2B
(25)MSEOUT        26.3B
(27)CE2           28.2C
(28)A6            29.1C
(28)A7            29.1C
(29)ABORT_KEY_ORIG 30.4A
(29)BH_SYSFAIL_R <1..1> 30.1C
(29)CAP_DEL       30.2B
(29)L7SEGH <7..0> 30.1A 30.1C 30.4D
(29)PW_SENSE      30.1A
(29)RES_KEY_ORIG 30.3A
(29)RHLED_R <1..0> 30.1C 30.1D
(29)ROT <3..0>    30.3C
(29)USER_R <3..1> 30.2D
(29)VREF          30.2B
(30)VSI_TEST      31.4B
(30)VSI_TESTASYNC 31.4B
(30)VSI_TESTSCAN 31.4B
(31)ENA_INIT_L    32.4B
(31)RESI_SW       32.1B
(31)UN$1$ABT125$08P$Y <0..0> 32.1B
(32)ENA_ARB       33.4B
(32)PUA1          33.2C
(32)PUB1          33.2B
(32)UN$1$ABT125$104P$Y <0..0> 33.4B
(33)IRQ_L         34.4B
(34)DTACK_INENA 35.3B
(34)DTACK_OUTENA 35.3B
(34)ENA_CTRL      35.4B
(34)GF2           35.3B
(34)GI0           35.3B
(34)GI9           35.3B
(34)PULLA_CTRL    35.1B 35.2B
(34)PULLB_CTRL    35.1D 35.2C
(34)RETRY_INENA 35.3B
(35)ENA_ABUF      36.4B
(36)ENA_DBUF      37.4B
(37)FPY_DENSENINV 38.1A
(38)VME_P0_A <14..1> 39.2C 39.3C
(38)VME_P0_B <14..1> 39.2C 39.2D 39.3D
(38)VME_P0_C <14..1> 39.2D 39.3D
(38)VME_P0_D <14..1> 39.2D 39.3D
(38)VME_P0_E <14..1> 39.2B 39.3B
(41)TPIN          42.3C
(41)TPOUT         42.2C
(41)UN$1$9 MERGE$177P$Y <16..16> 42.1B
(41)UN$1$9 MERGE$178P$Y <16..16> 42.1C
(42)RESERVED <3..0> 43.2A 43.2C
(42)V3P3STB       43.1A
(43)ENA_MEM_BUF 44.4A
(43)UN$1$_ADAPT_MEM_ARR$110P$HABUS <288..288> 44.2C

(80)FILV3P3IN     81.3C
(80)FILVP5IN      81.4A
(80)V3P3_GEN      81.1B
(82)ESDR          83.1D
(82)ESD_STRIP     83.1C
(82)SHI           83.1D
(84)STIFF         85.1D
ABORTN           19.3C 19.4B 30.4A 30.4C
ABORTP           7.3A 19.2A 19.3C 19.4B 30.4A 30.4C
                 31.3B
```

```
BH_SYSFAIL_LED <1..0> 19.1B 19.4B 30.1C 30.1D 30.4C

B_CAS <3..0>     44.1C 44.2C 44.3D 44.4D 45.2B 46.2B
                 47.2B 48.2B 49.2B 50.2B 51.2B 52.2B
                 53.2B 54.2B 55.2B 56.2B 57.2B 58.2B
                 59.2B 60.2B 61.2B 62.2B 63.2B 64.2B
                 65.2B 66.2B 67.2B 68.2B 69.2B 70.2B
                 71.2B 72.2B 73.2B 74.2B 75.2B 76.2B
                 77.2B 78.2B 79.2B 80.2B
B_MEM_A <11..0> 44.1C 44.2B 44.2C 44.3B 44.4B 45.1B
                 45.2B 46.1B 46.2B 47.1B 47.2B 48.1B
                 48.2B 49.1B 49.2B 50.1B 50.2B 51.1B
                 51.2B 52.1B 52.2B 53.1B 53.2B 54.1B
                 54.2B 55.1B 55.2B 56.1B 56.2B 57.1B
                 57.2B 58.1B 58.2B 59.1B 59.2B 60.1B
                 60.2B 61.1B 61.2B 62.1B 62.2B 63.1B
                 63.2B 64.1B 64.2B 65.1B 65.2B 66.1B
                 66.2B 67.1B 67.2B 68.1B 68.2B 69.1B
                 69.2B 70.1B 70.2B 71.1B 71.2B 72.1B
                 72.2B 73.1B 73.2B 74.1B 74.2B 75.1B
                 75.2B 76.1B 76.2B 77.1B 77.2B 78.1B
                 78.2B 79.1B 79.2B 80.1B 80.2B
B_MWE            44.1C 44.2C 44.4D 45.2B 45.3B 46.2B
                 46.3B 47.2B 47.3B 48.2B 48.3B 49.2B
                 49.3B 50.2B 50.3B 51.2B 51.3B 52.2B
                 52.3B 53.2B 53.3B 54.2B 54.3B 55.2B
                 55.3B 56.2B 56.3B 57.2B 57.3B 58.2B
                 58.3B 59.2B 59.3B 60.2B 60.3B 61.2B
                 61.3B 62.2B 62.3B 63.2B 63.3B 64.2B
                 64.3B 65.2B 65.3B 66.2B 66.3B 67.2B
                 67.3B 68.2B 68.3B 69.2B 69.3B 70.2B
                 70.3B 71.2B 71.3B 72.2B 72.3B 73.2B
                 73.3B 74.2B 74.3B 75.2B 75.3B 76.2B
                 76.3B 77.2B 77.3B 78.2B 78.3B 79.2B
                 79.3B 80.2B 80.3B
B_OE             44.1C 44.2C 45.2B 46.2B 47.2B 48.2B
                 49.2B 50.2B 51.2B 52.2B 53.2B 54.2B
                 55.2B 56.2B 57.2B 58.2B 59.2B 60.2B
                 61.2B 62.2B 63.2B 64.2B 65.2B 66.2B
                 67.2B 68.2B 69.2B 70.2B 71.2B 72.2B
                 73.2B 74.2B 75.2B 76.2B 77.2B 78.2B
                 79.2B 80.2B
B_RAS <1..0>     44.1C 44.2C 44.3D 45.2B 46.2B 47.2B
                 48.2B 49.2B 50.2B 51.2B 52.2B 53.2B
                 54.2B 55.2B 56.2B 57.2B 58.2B 59.2B
                 60.2B 61.2B 62.2B 63.2B 64.2B 65.2B
                 66.2B 67.2B 68.2B 69.2B 70.2B 71.2B
                 72.2B 73.2B 74.2B 75.2B 76.2B 77.2B
                 78.2B 79.2B 80.2B
CAS <?..?>       2.1D
CENTR_ACK        15.3D 15.4C 38.1B 38.4B 41.2D 41.4C

CENTR_AF         15.3B 15.4C 38.2B 38.4B 41.2D 41.4C

CENTR_BSY        15.2D 15.4C 38.2B 38.4B 41.2D 41.4C

CENTR_D <7..0>   15.1B 15.1C 15.2B 15.4C 38.1B 38.4B
                 41.1D 41.2D 41.4C
CENTR_DS         15.1D 15.4C 38.1B 38.4B 41.1D 41.4C

CENTR_DX <7..0>  15.2B 15.2C 15.4C 38.3D 38.4B 38.4D
                 41.1D 41.4C
CENTR_ENA        15.1A 15.2A 15.4C 38.2B 38.4B
CENTR_ERR        15.4A 15.4C 38.2B 38.4B 41.2D 41.4C

CENTR_INIT       15.3B 15.4C 38.2B 38.4B 41.2D 41.4C
```

```
CENTR_PE         15.4A 15.4C 38.2B 38.4B 41.2D 41.4C

CENTR_SLCT       15.4A 15.4C 38.2B 38.4B 41.2D 41.4C

CENTR_SLCT_IN    15.3B 15.4C 38.2B 38.4B 41.2D 41.4C

CLKCPU           1.4B 2.3A 3.2D
CP_STAT <1..0>   1.4B 2.3D 16.3D
CS_BOOT <1..0>   19.2D 19.3A 21.1C 21.2C
CS_SYSEPR <1..0> 19.2D 19.3A 22.1C 22.2C
CTRL_VPPINV      18.4B 19.3A 19.3B
CTS_DCD_A        38.1D 41.3C 41.4C
CTS_DCD_B        38.2D 41.3D 41.4C
DRAM_CODE_A <1..0> 29.2B 29.3B 35.3B 43.1C 43.3C
                 43.4C
DRAM_CODE_B <1..0> 29.2B 29.3B 35.3B 43.1A 43.3A
                 43.4C
DRAM_CODE_LOC <1..0> 29.1C 29.2C 29.3B 35.3B
EB_A <23..0>     18.1C 18.2C 18.2D 19.1C 19.2B 19.2D
                 19.3A 21.1C 21.2B 21.2C 21.3B 21.4B
                 22.1C 22.2C 28.1B 28.2B 28.3B
EB_A19_ORIG      19.1C 19.3A
EB_ADDRL         16.3A 18.3B
EB_D <7..0>      16.3A 18.3C 18.4C 19.2B 19.3A 22.1D
                 22.2D 28.1C 28.2C 28.3B 29.1D 29.2D
                 29.3D 29.4C 30.3C 30.4D
EB_RD            16.3A 19.2D 19.3A 21.1C 21.3C 22.1C
                 22.2C 28.2C
EB_WR            16.3A 19.2D 19.3A 21.3A 22.3A 28.2C

EIRQ             1.2B 8.2A 16.3C
ENA_MEM_OE       44.1C 44.2C
ENET_AUI         8.1C 12.2A
ENET_CLSN        8.2C 12.2A
ENET_RCLK        8.1C 12.2A
ENET_RENA        8.1C 12.2A
ENET_RX          8.2C 12.2A
ENET_TCLK        8.1C 12.2A
ENET_TENA        8.1C 12.2A
ENET_TX          8.1C 12.2A
EPROM_CS         16.3A 19.2D 19.3A
ETH_COL_M        12.2C 14.2B 38.3C
ETH_COL_P        12.2C 14.2B 38.3C
ETH_POW          38.3D 38.4C 41.2D 41.4C
ETH_POWER        13.2B 14.2B 38.4C
ETH_REC_M        12.2C 14.2B 38.4C
ETH_REC_P        12.2C 14.2B 38.4C
ETH_TRA_M        12.2C 14.2B 38.3C
ETH_TRA_P        12.2C 14.2B 38.4C
FILV3P3          2.2B 81.3D
FILVP5           2.2B 81.4D
FPY_CLK24        8.3C 16.4C 17.2B
FPY_CLK32        8.3C 16.4C 17.3B
FPY_DENSEL       16.1C 38.1A 38.3A
FPY_DENSENSE     16.1C 38.1B 38.3A
FPY_DIR          16.1C 38.1A 38.3A
FPY_DISKCHG      16.1C 38.2A 38.3A
FPY_DRVSEL       16.1C 38.1A 38.3A
FPY_EJECT        16.1C 38.2A 38.3A
FPY_HEADSEL      16.1C 38.2A 38.3A
FPY_INDEX        16.1C 38.1A 38.3A
FPY_MOTEN        16.1C 38.1A 38.3A
FPY_RDDATA       16.1C 38.2A 38.3A
FPY_STEP         16.1C 38.1A 38.3A
FPY_TRACK0       16.1C 38.2A 38.3A
FPY_WRDATA       16.1C 38.2A 38.3A
FPY_WRGATE       16.1C 38.2A 38.3A
```

DESIGNER:		SHEET 86 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	CROSS REFERENCE	REV. 0.2

A	B	C	D

FPY_WRPROT	16.1C 38.2A 38.3A
FPY_XT24_IN	8.3C 17.1B
FPY_XT24_OUT	8.3C 17.1B
FPY_XT32_IN	8.3C 17.1B
FPY_XT32_OUT	8.3C 17.1B
GEN_CS	16.3A 19.2D 19.3A
GEN_RDY	16.3A 19.2D 19.3A
ID_CS	8.3C 15.4B
IRL <3..0>	1.4B 2.3D 16.3D
KBDINP2	38.1D 41.3C 41.4C
KBDIN_CONN	26.1C 27.2B 38.1C
KBDOUTP2	38.1D 41.3C 41.4C
KBDOUT_CONN	26.1C 27.2B 38.2C
KBD_DIN	16.3C 26.1A
KBD_DOUT	16.3C 26.1A
KBD_PWR_CONN	26.3C 27.2B 41.1B
LCASYSRESETO	19.3B 19.3D 32.1A
MEM_P <7..?>	2.1D 2.4C 42.1A 42.2B 42.3A 42.3B 42.3C 43.3A 43.3B 43.3C 43.3D 43.4B 44.2C 53.2C 62.2C 71.2C 80.2C
MSEINP2	38.1D 41.3C 41.4C
MSEIN_CONN	26.2C 27.2B 38.2C
MSEOUT_CONN	26.3B 27.2B 41.2B
MSE_DIN	16.3C 26.2A
MSE_DOUT	16.3C 26.3A
M_D <63..0>	2.1D 2.4C 42.1A 42.2A 42.2B 42.3A 42.3B 42.3C 42.4A 43.1A 43.1B 43.1C 43.1D 43.2A 43.2B 43.2C 43.2D 43.3A 43.3B 43.3C 43.3D 43.4B 44.2C 45.2C 46.2C 47.2C 48.2C 49.2C 50.2C 51.2C 52.2C 54.2C 55.2C 56.2C 57.2C 58.2C 59.2C 60.2C 61.2C 63.2C 64.2C 65.2C 66.2C 67.2C 68.2C 69.2C 70.2C 72.2C 73.2C 74.2C 75.2C 76.2C 77.2C 78.2C 79.2C
NC01	15.1B 15.2B 18.1C
NMI	7.3A 16.3A 19.2A 19.2B 31.3B
NVRAM_CS	16.3A 18.4B 28.2C
P2C1	38.1A 38.4D 41.1D 41.4C
P2C2	38.1A 38.4D 41.1D 41.4C
P2C4	38.1A 38.3D 41.1D 41.4C
P2C5	38.1A 38.3D 41.1D 41.4C
P2C8	38.1A 38.3D 41.1D 41.4C
P2C9	38.1A 38.3D 41.1D 41.4C
P2C10	38.1A 38.3D 41.1D 41.4C
P2C11	38.2A 38.3D 41.2D 41.4C
P2C12	38.2A 38.3D 41.2D 41.4C
P2C13	38.2A 38.3D 41.2D 41.4C
P2C14	38.2A 38.3D 41.2D 41.4C
P2C15	38.2A 38.3D 41.2D 41.4C
P2C16	38.2A 38.3D 41.2D 41.4C
P2C17	38.2A 38.3D 41.2D 41.4C
P2C18	38.2A 38.3D 41.2D 41.4C
POWER_OFF	16.3C
P_ACK	8.2C 15.3C 15.4B
P_ACK_DIR	8.3C 15.3C 15.4B
P_AF	8.3C 15.3A 15.4B
P_BSY	8.2C 15.2C 15.4B
P_BSY_DIR	8.3C 15.2C 15.4B
P_D <7..0>	8.2C 15.1A 15.2A 15.4B
P_DS	8.2C 15.1C 15.4B
P_DS_DIR	8.3C 15.1C 15.4B
P_D_DIR	8.3C 15.1A 15.2A 15.4B
P_ERR	8.3C 15.4B
P_INIT	8.3C 15.3A 15.4B
P_PE	8.2C 15.4B

P_RAS <6..0>	42.1C 42.2C 43.1A 43.1C 43.1D 43.4C
P_SLCT	8.2C 15.4B
P_SLCT_IN	8.3C 15.3A 15.4B
RD_DRAM_CODE	19.2D 19.3A 29.3C
RD_ROTARY	19.2D 19.4B 30.4C
RESETN	30.2A 30.4C
RES_KEY	19.3C 19.4B 30.2A 30.4C
RES_PWUP	7.3A 19.2A 19.2D 19.4B 30.1C 30.4C 31.3B
RES_PWUP_P	19.3D 19.4B 30.1C 30.4C 32.2A
RES_PWUP_REQ	19.3C 19.4B 30.1A 30.4C
RTS_DTR_A	38.1D 41.3C 41.4C
RTS_DTR_B	38.2D 41.3D 41.4C
RUN_HALT_LED <1..0>	19.3B 19.4B 30.1C 30.1D 30.4C
SB_ACK <2..0>	2.1A 4.2C 4.3B 5.2C 5.3B 6.2D 6.3B 7.1B 8.2A 16.1A 31.2B
SB_AS	2.1A 2.4A 4.1B 4.2C 5.1B 5.2C 6.1B 6.2D 7.1B 8.2A 16.1A 31.2B
SB_BG <5..0>	2.2A 2.2B 4.1C 4.2C 5.1C 5.2C 6.1C 6.2D 8.2A 31.2B
SB_BR <5..0>	2.2B 4.1B 4.2C 5.1B 5.2C 6.1B 6.2D 7.2B 8.2A 31.2B
SB_CLK <0..0>	2.2A 2.4A 4.1B 4.3C 5.1B 5.3C 6.1B 6.2D 7.3C 8.2A 16.2A 19.1A 19.1C 31.2B
SB_D <31..0>	2.1A 4.2A 4.2B 4.2C 5.2A 5.2B 5.2C 6.1A 6.1B 6.1C 6.2A 6.2B 6.2C 6.2D 6.3A 6.3B 7.1C 7.2C 8.1A 16.1A 31.1B
SB_DP	4.2C 4.3A 5.2C 5.3A 6.2D 6.3A 7.1B
SB_INT <7..1>	4.1B 4.2B 4.3C 5.1B 5.2B 5.3C 6.1B 6.2B 6.2D 6.3B 7.2B 8.2A 16.2A 31.2B
SB_LERR	2.1A 4.2C 4.3B 5.2C 5.3B 6.2D 6.3B 7.1B 8.2A 31.2B
SB_PA <27..0>	2.1A 2.4B 2.4C 4.2A 4.2B 4.2C 4.3A 4.3B 4.3C 5.2A 5.2B 5.2C 5.3A 5.3B 5.3C 6.2D 6.3A 6.3B 6.3C 8.1A 16.1A 18.1C 18.2A 18.2B 18.2C 31.2B
SB_RD	2.1A 4.2B 4.2C 5.2B 5.2C 6.2D 6.3B 7.1B 8.2A 16.1A 31.2B
SB_RST	2.2A 2.4A 4.2C 4.3B 5.2C 5.3B 6.2D 6.3B 7.1B 8.2A 16.2A 18.2A 18.4B 19.1A 19.3B 30.4D 31.2B
SB_RST_REQ	7.3A 16.3A 19.2A 19.3C 31.3B
SB_SEL <7..0>	2.2A 2.4B 2.4C 4.1A 4.2C 5.1A 5.2C 6.1A 6.2D 8.2A 16.1A 19.1A 19.2A 31.2A
SB_SIZ <2..0>	2.1A 4.2A 4.2B 4.2C 5.2A 5.2B 5.2C 6.2D 6.3A 6.3B 6.3C 7.1B 8.2A 16.1A 31.2B
SCC_CLK	8.3C 16.4C 17.2C
SCC_XTIN	8.3C 17.1B
SCC_XTOUT	8.3C 17.1B
SCSI_ACK	8.2C 10.3B 11.3B 38.1D 38.4A 41.2C 41.4C
SCSI_ATN	8.2C 10.3B 11.3B 38.2D 38.4A 41.2C 41.4C
SCSI_BSY	8.2C 10.2B 11.3B 38.2D 38.4A 41.2C 41.4C
SCSI_CD	8.2C 10.3B 11.3B 38.1D 38.4A 41.3C 41.4C
SCSI_D <7..0>	8.2C 10.2B 11.2B 11.3B 38.2D 38.4A 41.1C 41.4C
SCSI_DP	8.2C 10.2B 11.3B 38.2D 38.4A 41.1C 41.4C

SCSI_IO	8.2C 10.3B 11.3B 38.1D 38.4A 41.3C 41.4C
SCSI_MSG	8.2C 10.3B 11.3B 38.1D 38.4A 41.2C 41.4C
SCSI_REQ	8.2C 10.2B 11.3B 38.1D 38.4A 41.3C 41.4C
SCSI_RST	8.2C 10.3B 11.3B 38.1D 38.4A 41.2C 41.4C
SCSI_SEL	8.2C 10.2B 11.3B 38.1D 38.4A 41.2C 41.4C
SCSI_XTIN	8.3C 17.1B
SCSI_XTOUT	8.3C 17.1B
SER_CTS_A	16.2C 24.2A 24.3B
SER_CTS_A_CONN	24.2D 25.2B 38.1C
SER_CTS_B	16.2C 23.2A 23.3B
SER_CTS_B_CONN	23.2D 25.2B 38.2C
SER_DCD_A	16.2C 24.2A
SER_DCD_A_CONN	24.2D 25.2B 38.1C 41.3B
SER_DCD_B	16.2C 23.2A
SER_DCD_B_CONN	23.2D 25.2B 38.2C 41.2B
SER_DSR_A	16.2C 24.2A
SER_DSR_A_CONN	24.2D 25.2B 41.3B
SER_DSR_B	16.3C 23.2A
SER_DSR_B_CONN	23.2D 25.2D 41.2B
SER_DTR_A	16.2C 24.2A
SER_DTR_A_CONN	24.2D 25.2D 38.1C 41.3B
SER_DTR_B	16.2C 23.2A
SER_DTR_B_CONN	23.2D 25.2B 38.2C 41.2B
SER_GND_A_CONN	24.2D 25.2B 41.3B
SER_GND_B_CONN	23.2D 25.2D 41.1B
SER_RTS_A	16.2C 24.2A 24.2B 24.3B
SER_RTS_A_CONN	24.2D 25.2B 38.1C
SER_RTS_B	16.2C 23.2A 23.2B 23.3B
SER_RTS_B_CONN	23.2D 25.2D 38.2C
SER_RTXC_A	16.2C 24.2A 24.3B 24.4B
SER_RTXC_A_CONN	24.2D 25.2D 41.2B
SER_RTXC_B	16.3C 23.2A 23.3B 23.4B
SER_RTXC_B_CONN	23.2D 25.2D 41.1B
SER_RXC_A_CONN	24.2D 25.1D 41.2B
SER_RXC_B_CONN	23.2D 25.2D 41.1B
SER_RXD_A	16.2C 24.2A
SER_RXD_A_CONN	24.2D 25.2B 38.1D 41.3C 41.4C
SER_RXD_B	16.2C 23.2A
SER_RXD_B_CONN	23.2D 25.2D 38.2D 41.3D 41.4C
SER_TRXC_A	16.2C 24.2A 24.2B 24.3B 24.4B
SER_TRXC_B	16.2C 23.2A 23.2B 23.3B 23.4B
SER_TXC_A_CONN	24.2D 25.2D 41.3B
SER_TXC_B_CONN	23.2D 25.2D 41.1B
SER_TXD_A	16.2C 24.2A
SER_TXD_A_CONN	24.2D 25.1B 38.1D 41.3C 41.4C
SER_TXD_B	16.2C 23.2A
SER_TXD_B_CONN	23.2D 25.1D 38.2D 41.3D 41.4C
SHIELD	11.1D 14.2C 25.1C 27.2C 30.3A 30.4A 83.1C
SIRQ	1.2B 8.2A 16.3C
SLV_SYS_LED	16.3A 19.1C 19.4B
SP_SEL <1..0>	2.2D 2.4D 35.3B 35.3C
TERMPWR	9.2B 10.3C 11.3C 38.2D 38.4A 41.2C 41.4C
TIM_CLK	16.3A 16.4C 17.3B
USER_LED <3..0>	19.1C 19.4B 30.2C 30.2D 30.4C
U_CAS <3..0>	2.4D 42.1B 43.1A 43.1B 43.1C 43.1D 43.4A 44.1C 44.3B 44.4B
U_MEM_A <11..0>	2.1D 2.4D 42.1B 43.1A 43.1B 43.1C 43.1D 43.4C 44.1C 44.2A 44.3A 44.4A
U_MWE	2.2D 2.4D 42.1B 43.1A 43.1C 43.4A 44.1C 44.4B

U_RAS <7..0>	2.1D 2.4D 42.1B 42.1C 42.2C 43.1A 43.1B 43.1C 43.4C 44.1C 44.3B
V3P3_VMECONN	81.1A
VA <31..1>	19.1D 19.3B 31.1C 36.1A 36.1B 36.2B 36.3B 36.4B
VACFAILI	31.3C 32.2B
VADIR	19.1D 19.3B 31.1C 36.1B
VAM <5..0>	31.2C 35.1B 35.2B 42.2C 42.4C
VAS	19.2D 19.3C 31.2C 35.2B
VASDIR	19.2D 19.3C 31.2C 35.2C
VBBSYI	31.3C
VBBSYO	19.3C 19.3D 31.3C 33.4B
VBCLR	31.3C 33.2B
VBERRI	19.1C 19.2D 35.4B
VBERRIVSI	19.1C 19.4C 31.1A 31.2D
VBERRO	19.1B 19.2D 35.4B
VBERROVSI	19.1B 19.4C 31.1A 31.2D
VBGIN <3..0>	31.3C 33.2B
VBGOUT <3..0>	31.3C 33.3B 33.4B
VBRI <3..0>	31.3C 33.1B 33.2B
VBRO <3..0>	31.3C 33.1B 33.2B
VD <31..0>	31.1C 37.1A 37.1B 37.2B 37.3B 37.4B
VDDIR	31.1C 37.1C
VDS <1..0>	19.2A 19.2D 31.2C 35.1B
VDTACK	19.2D 31.2C 35.3B
VDTACKDIR	19.2D 31.2C
VDTACKDIS	19.3C 19.4C 35.1B 35.3C
VDTACKROE	19.2D 31.2C
VGA <4..0>	19.1B 19.4D 32.3B 32.4B
VGAP	19.1B 19.4D 32.4B
VIACK	31.2C 35.1B 42.2B 42.4C
VIACKIN	31.3C 34.4B
VIACKOUT	31.3C 34.4B
VIRQI <7..1>	31.2C 34.3A
VIRQO <7..1>	31.2C 34.2B 34.3A 34.3B
VLWORD	31.1C 36.1A 36.4B
VMADIR	31.2C 35.1D 42.2B 42.4C
VME_A <31..1>	36.1C 36.1D 36.2C 36.3C 36.4C 40.1A 40.2D 40.3C 40.3D 41.1A 41.1C 41.2C
VME_ACFAIL	32.2C 40.1C 40.3A
VME_ADDP	39.2A
VME_AM <5..0>	35.1D 35.2D 40.2A 40.2C 40.2D 40.3C
VME_AMCP	39.2A
VME_APV	39.2A
VME_AS	35.2D 40.2A 40.2C
VME_BBSY	33.4C 40.1C 40.3A
VME_BCLR	33.2C 40.1C 40.3A
VME_BERR	35.4C 40.2A 40.2D
VME_BGIN <3..0>	33.2C 40.1C 40.3A
VME_BGOUT <3..0>	33.3C 33.4C 40.1C 40.2C 40.3A
VME_BR <3..0>	33.1C 33.2C 40.2C 40.3A
VME_D <31..0>	37.1C 37.1D 37.2C 37.3C 37.4C 40.1A 40.1C 40.1D 41.1A 41.2C 41.3C
VME_DATP	39.2A
VME_DPV	39.2A
VME_DS <1..0>	35.1D 40.2A 40.2C
VME_DTACK	35.3C 40.2A 40.2C
VME_GA <4..0>	32.3C 32.4C 40.1D 40.2B 40.2D
VME_GAP	32.4C 40.1D 40.2B
VME_IACK	35.1D 40.2A 40.2C
VME_IACKIN	34.4C 40.2A 40.2C
VME_IACKOUT	34.4C 40.2C 40.3A
VME_IRQ <7..1>	34.1C 34.2C 34.3C 34.3D 40.2A 40.3C
VME_LII	39.2A

DESIGNER:	----------	SHEET 87 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	CROSS REFERENCE	REV. 0.2

```
VME_LIO          39.3A
VME_LWORD        36.1D 36.4C 40.1A 40.2D
VME_P2Z <?..?>   41.4C
VME_RETRY        35.4C 41.1C 41.2A
VME_SBA          39.2A
VME_SBB          39.2A
VME_STDBY        28.4B 40.3C 43.3A 43.3C
VME_SYSCLK       40.1C 40.3A
VME_SYSFAIL      32.3C 40.1D 40.3A
VME_SYSRESET     32.1C 40.2D 40.3A
VME_VPC          39.3A
VME_WRITE        35.1D 40.2A 40.2C
VP5NVRAM         28.2C
VPP_SWITCH       20.3D 21.4B 22.1C 22.2C
VRETRYI          31.2C 35.4B
VSCON            19.2B 19.4C 31.4C 32.2C 33.3C
VSISYSRESETI     19.3B 19.3D 31.3C
VSI_CLK          17.4B 31.3B
VSYSCLK          17.3C 31.3A 33.3B
VSYSFAILI        19.3C 19.3D 31.3C 32.3B
VSYSFAILO        19.2D 19.4D 31.3C 32.3B
VSYSRESETI       19.3B 19.3D 32.1A
VSYSRESETO       31.3C 32.2A
VWRITE           31.2C 35.1B
V_ETH_COL_M      38.3C 38.3D 41.3D 41.4C
V_ETH_COL_P      38.3C 38.3D 41.3D 41.4C
V_ETH_REC_M      38.3D 38.4C 41.3D 41.4C
V_ETH_REC_P      38.3D 38.4C 41.2D 41.4C
V_ETH_TRA_M      38.3C 38.3D 41.3D 41.4C
V_ETH_TRA_P      38.3D 38.4C 41.3D 41.4C
HR_7SEGH         19.3B 19.4B 30.4D
_AMIA <3..0>     1.1C 1.4C 35.1A 35.2C 42.2C 42.4B
                 42.4C
_CENTRX_BUS <?..?> 1.1A 8.3D 15.4B
_CENTRX_CONN_BUS <?..?> 1.2B 15.4D 38.4B
_CLOCK_BUS <?..?> 1.2B 8.3D 16.4D 17.2A
_DRAM_CODE_BUS <?..?> 1.3D 2.4B 29.3A 35.3B 43.4B
_ETHERNET_BUS <?..?> 8.1D 12.1A
_ETH_BUS <?..?> 1.1A
_ETH_CONN_BUS <?..?> 1.1A 12.1D 13.2D 14.2A 38.3B
_FLOPPY_BUS <?..?> 1.3A 16.1D 38.3A
_JTAG_CLK        2.3B 8.3A 16.4A 16.4C 19.1B 19.2B
                 31.3B 82.2C
_JTAG_RST        2.3B 8.3A 16.4A 16.4C 82.2C
_JTAG_TDI_TDO <5..0> 2.3B 8.3A 16.4A 16.4C 19.1B
                 19.1D 19.2B 31.4B 82.2C
_JTAG_THS        2.2B 8.3A 16.4A 16.4C 19.1B 19.2B
                 31.3B 82.2C
_KBD_MSE_BUS <?..?> 1.3A 16.3D 26.1A
_KBD_MSE_CONN_BUS <?..?> 1.3A 26.1D 27.2A 38.2C
_LOC_BUS <?..?> 1.3C 16.4A 18.4B 19.4A 20.3A 21.4A
                 22.4A 28.3A 29.4B 30.3C 30.4B
_MAC <?..?>      1.3D 42.1C 44.2C
_MAD <?..?>      1.4D
_MAE <?..?>      1.4D
_MAU <?..?>      1.4C 42.1B
_MEM_0_DRM <499..0> 44.1D
_MEM_1_DRM <499..0> 44.2D
_MEM_ARRAY_BUS <499..0> 2.4D 43.4A
_MEM_ARR_CON <499..0> 42.1D
_MEM_ARR_CONN <499..0> 44.1A
_MEM_ARR_UBUF <499..0> 42.1A
_SBUS <?..?>     1.3C 2.2A 4.3D 5.3D 6.3D 7.2A 8.3A
                 16.3A 17.3D 18.3A 19.2A 31.3A
_SCSI_BUS <?..?> 1.1A 8.2D 9.2D 10.3A 11.3A 38.4A
_SER_A_BUS <?..?> 1.3A 16.2D
_SER_A_CONN_BUS <?..?> 1.3A 25.1A 38.1C
```

```
_SER_BUS <?..?> 23.2A 24.2A
_SER_B_BUS <?..?> 1.3A 16.3D
_SER_B_CONN_BUS <?..?> 1.3A 25.4A 38.3C
_SER_CONN_BUS <?..?> 23.2D 24.2D
_TRA_VME <?..?> 1.1C 19.4B 19.4D 31.1A 31.4D 32.4A
                33.4A 34.1A 35.1A 36.1A 37.1A
_VME_BUS <?..?> 1.2C 32.4D 33.1D 34.1D 35.4D 36.1D
                37.1D 38.1D 39.3A 39.3C 39.4C 40.1A
                40.2C 40.3B 40.4A 40.4B 41.1D 41.2D
                41.3D 41.4A 41.4C
_VME_BUS_X <199..0> 32.4D
```

DESIGNER:		SHEET 88 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	CROSS REFERENCE	REV. 0.2

*** Unit Cross-Reference ***
--- for the entire design --

B1	CONSOC1X16	38.1A
B2	CONSOC1X16	38.1B
B3	CONSOC1X16	38.1B
B4	CONSOC1X6	82.1C
B11	CONSOC1X8	38.3C
B12	CONSOC1X8	38.3C
B13	CONSOC1X8	38.3C
C1	CAPC	81.4B
C2	CAPC	81.3D
C3	CAPC	13.3B
C4	CAPC	17.3D
C5	CAPC	26.1B
C6	CAPC	26.4B
C7	CAPC	26.3B
C8	CAPC	26.2B
C9	CAPC	26.2B
C10	CAPC	12.3B
C11	CAPC	85.1D
C12	CAPC	9.3B
C13	CAPC	12.3B
C14	CAPC	12.4A
C15	CAPC	12.3B
C16	CAPC	12.3A
C17	CAPC	12.2C
C18	CAPC	12.3A
C19	CAPC	12.3A
C20	CAPC	12.2C
C21	CAPC	12.4A
C22	CAPC	84.2D
C23	CAPC	17.2B
C24	CAPC	17.2C
C25	CAPC	20.2C
C26	CAPC	20.2C
C27	CAPC	20.2D
C28	CAPC	11.4C
C29	CAPC	10.3C
C30	CAPC	84.1D
C31	CAPC	12.1B
C32	CAPC	17.3C
C33	CAPC	33.4B
C34	CAPC	30.2B
C35	CAPC	3.3C
C36	CAPC	30.2B
C37	CAPC	15.1D
C38	CAPC	15.2D
C39	CAPC	15.3D
C40	CAPC	17.1C
C41	CAPC	17.1C
C42	CAPC	17.1D
C43	CAPC	34.1A
C44	CAPC	34.1A
C45	CAPC	34.1A
C46	CAPC	34.1A
C47	CAPC	7.3D
C48	CAPC	34.2A
C49	CAPC	34.1A
C50	CAPC	34.1A
C51	CAPC	81.2D
C52	CAPC	81.2D
C53	CAPC	81.2C
C54	CAPC	17.1D
C55	CAPC	81.4B
C56	CAPC	17.1D
C57	CAPC	17.1D
C58	CAPC	81.2D
C59	CAPC	81.3D
C60	CAPC	17.3C
C61	CAPC	17.3B
C62	CAPC	17.2B
C63	CAPC	17.2D
C64	CAPC	17.4A
C65	CAPC	12.1B
C66	CAPC	32.2A
C67	CAPC	32.3A
C68	CAPC	32.1A
C69	CAPC	42.2C
C70	CAPC	17.2B
C71	CAPC	17.1B
CBL1	CAPBLOCK	83.2B
CBL2	CAPBLOCK	83.2A
CBL3	CAPBLOCK	83.2B
CBL4	CAPBLOCK	83.2B
CBL5	CAPBLOCK	83.2A
CBL6	CAPBLOCK	83.2A
CBL7	CAPBLOCK	83.2B
CBL8	CAPBLOCK	83.2B
CBL9	CAPBLOCK	83.2A
CBL10	CAPBLOCK	83.2A
CBL11	CAPBLOCK	83.2A
CBL12	CAPBLOCK	83.2B
CBL13	CAPBLOCK	83.1B
CBL14	CAPBLOCK	83.2A
CBL15	CAPBLOCK	83.2A
CBL16	CAPBLOCK	83.2A
CBL17	CAPBLOCK	83.1A
CBL18	CAPBLOCK	83.1B
CBL19	CAPBLOCK	83.1B
CBL20	CAPBLOCK	83.2A
CBL21	CAPBLOCK	83.2A
CBL22	CAPBLOCK	83.2A
CBL23	CAPBLOCK	83.1A
CBL24	CAPBLOCK	83.1A
CBL25	CAPBLOCK	83.1B
CBL26	CAPBLOCK	83.2A
CBL27	CAPBLOCK	83.2A
CBL28	CAPBLOCK	83.2A
CBL29	CAPBLOCK	83.1A
CBL30	CAPBLOCK	83.1A
CBL31	CAPBLOCK	83.1A
CBL32	CAPBLOCK	83.2A
CBL33	CAPBLOCK	83.2A
CBL34	CAPBLOCK	83.2A
CBL35	CAPBLOCK	83.1A
CBL36	CAPBLOCK	83.1A
CBL37	CAPBLOCK	83.1A
CBL38	CAPBLOCK	83.2B
CBL39	CAPBLOCK	83.2A
CBL40	CAPBLOCK	83.2A
CBL41	CAPBLOCK	83.1A
CBL42	CAPBLOCK	83.1A
CBL43	CAPBLOCK	83.1A
CBL44	CAPBLOCK	83.2B
CBL45	CAPBLOCK	83.2A
CBL46	CAPBLOCK	83.1A
CBL47	CAPBLOCK	83.1A
CBL48	CAPBLOCK	83.1A
CBL49	CAPBLOCK	83.2B
CBL50	CAPBLOCK	83.1A
CBL51	CAPBLOCK	83.1A
CBL52	CAPBLOCK	83.1A
CBL53	CAPBLOCK	83.1B
CBL54	CAPBLOCK	83.1A
CBL55	CAPBLOCK	83.1A
CBL56	CAPBLOCK	83.1B
CBL57	CAPBLOCK	83.1A
CBL58	CAPBLOCK	83.1B
CBL59	CAPBLOCK	83.1B
CBL60	CAPBLOCK	83.1B
CBL61	CAPBLOCK	83.1B
CBL62	CAPBLOCK	83.2B
CBL63	CAPBLOCK	83.2B
CBL64	CAPBLOCK	83.2B
CBL65	CAPBLOCK	83.2B
CBL66	CAPBLOCK	83.1B
CBL67	CAPBLOCK	83.1B
CBL68	CAPBLOCK	83.1B
CBL69	CAPBLOCK	83.2B
CBL70	CAPBLOCK	83.2B
CBL71	CAPBLOCK	83.2C
CBL72	CAPBLOCK	83.2C
CBL73	CAPBLOCK	83.2C
CBL74	CAPBLOCK	83.2B
CBL75	CAPBLOCK	83.2B
CBL76	CAPBLOCK	83.2B
CBL77	CAPBLOCK	83.1C
CBL78	CAPBLOCK	83.1C
CBL79	CAPBLOCK	83.1B
CBL80	CAPBLOCK	83.2B
CBL81	CAPBLOCK	83.2B
CBL82	CAPBLOCK	83.2C
CBL83	CAPBLOCK	83.2C
CBL84	CAPBLOCK	83.1B
CBL85	CAPBLOCK	83.1B
CBL86	CAPBLOCK	83.1C
CBL87	CAPBLOCK	83.2B
CBL88	CAPBLOCK	83.2B
CBL89	CAPBLOCK	83.2C
CBL90	CAPBLOCK	83.2C
CBL91	CAPBLOCK	83.1B
CBL92	CAPBLOCK	83.1B
CBL93	CAPBLOCK	83.1C
CBL94	CAPBLOCK	83.2B
CBL95	CAPBLOCK	83.1B
CBL96	CAPBLOCK	83.2C
CBL97	CAPBLOCK	83.1C
CBL98	CAPBLOCK	83.1B
CBL99	CAPBLOCK	83.1C
CBL100	CAPBLOCK	83.2B
CBL101	CAPBLOCK	83.2B
CBL102	CAPBLOCK	83.2B
CBL103	CAPBLOCK	83.2C
CBL104	CAPBLOCK	83.1B
CBL105	CAPBLOCK	83.1B
CBL106	CAPBLOCK	83.1B
CBL107	CAPBLOCK	83.2C
CBL108	CAPBLOCK	83.2C
CBL109	CAPBLOCK	83.1B
CBL110	CAPBLOCK	83.1C
CBL111	CAPBLOCK	83.2B
CBL112	CAPBLOCK	83.2C
CBL113	CAPBLOCK	83.1B
CBL114	CAPBLOCK	83.1C
CBL115	CAPBLOCK	83.1C
CBL116	CAPBLOCK	83.1B
CBL117	CAPBLOCK	83.1C
CBL118	CAPBLOCK	83.1B
CBL119	CAPBLOCK	81.2D
CBL120	CAPBLOCK	83.1C
CBL121	CAPBLOCK	83.1C
CBL122	CAPBLOCK	81.2C
CBL123	CAPBLOCK	81.2C
CBL124	CAPBLOCK	81.2D
CBL125	CAPBLOCK	81.2D
CBL126	CAPBLOCK	81.2C
CBL127	CAPBLOCK	81.2D
CBL128	CAPBLOCK	81.2C
CBL129	CAPBLOCK	22.1D
CBL130	CAPBLOCK	22.2D
CBL131	CAPBLOCK	21.3D
CBL132	CAPBLOCK	21.2D
CBL133	CAPBLOCK	28.4C
CBL134	CAPBLOCK	83.2A
CBL135	CAPBLOCK	83.2B
CBL136	CAPBLOCK	83.2B
CBL137	CAPBLOCK	83.2A
CBL138	CAPBLOCK	83.2B
CBL139	CAPBLOCK	83.2A
CBL140	CAPBLOCK	83.2A
CBL141	CAPBLOCK	83.2A
CBL142	CAPBLOCK	83.2A
CBL143	CAPBLOCK	83.2A
CBL144	CAPBLOCK	83.2B
CBL145	CAPBLOCK	83.2B
CBL146	CAPBLOCK	83.2B
CBL147	CAPBLOCK	83.2B
CBL148	CAPBLOCK	83.2B
CBL149	CAPBLOCK	83.2B
CBL150	CAPBLOCK	83.2C
CBL151	CAPBLOCK	83.2C
CE1	CAPELKO	20.3D
CE2	CAPELKO	20.1C
CE3	CAPELKO	83.3D
CE4	CAPELKO	83.3D
CE5	CAPELKO	83.3C
CE6	CAPELKO	83.3C
CE7	CAPELKO	83.3D
CE8	CAPELKO	81.2C
CE9	CAPELKO	81.2C
CE10	CAPELKO	81.1A
CE11	CAPELKO	81.3C
CE12	CAPELKO	81.4A
CE13	CAPELKO	81.1A
CE14	CAPELKO	81.2B
CE15	CAPELKO	26.4C
CE16	CAPELKO	9.3B
CE17	CAPELKO	12.4B
CE18	CAPELKO	12.4B
CE19	CAPELKO	13.2B
CP1	CP	22.2D
CP2	CP	22.1D
CP3	CP	2.4A
CP4	CP	42.2C
CP5	CP	84.2A
CP7	CP	42.2B
CP9	CP	42.2B
CP10	CP	35.3C
CP11	CP	42.2B
CP12	CP	35.3B
CP13	CP	35.3B
CP14	CP	84.3B
CP15	CP	84.1B
CP16	CP	84.1B
CP17	CP	43.1D
CP18	CP	43.2A
CP19	CP	12.2B

DESIGNER:

COMPUTERS

APR/25/95 Manual

CROSS REFERENCE

SHEET 89 OF 94

REV. 0.2

CP20	CP	43.2A
CP21	CP	43.2C
CP22	CP	43.2C
CP23	CP	23.4C
CP24	CP	23.3C
CP25	CP	24.4C
CP27	CP	24.3C
CP28	CP	18.1D
CP29	CP	18.1D
CP30	CP	2.2A
CP31	CP	84.1C
CP38	CP	16.3D
CP40	CP	16.4B
CP41	CP	16.4A
CP42	CP	16.4B
CP43	CP	16.4B
CP44	CP	44.3C
CP45	CP	44.3C
CP47	CP	29.1B
CP50	CP	29.1B
CP57	CP	84.2B
CP60	CP	84.2C
CP61	CP	84.2B
CP62	CP	84.1B
CP63	CP	84.2C
CP66	CP	84.2B
CP68	CP	19.2D
CP70	CP	84.1C
CP72	CP	84.2B
CP74	CP	31.4A
CP75	CP	31.4A
CP76	CP	31.4A
CP84	CP	84.2A
CP85	CP	84.2A
CP87	CP	84.2A
CP88	CP	84.2D
CP89	CP	84.1D
CP106	CP	84.3A
CP108	CP	84.3A
CP109	CP	84.2A
D1	LEDDSHORT	30.1D
D2	LEDDSHORT	30.2D
D3	DIODE	28.3C
D4	DIODE	28.4C
D5	DIODETERM	17.3D
D6	DIODETERM	3.3C
D7	DIODETERM	12.1B
D8	DIODETERM	7.3D
D9	DIODETERM	7.3D
D10	DIODETERM	7.3D
D11	DIODETERM	84.2D
D12	DIODETERM	84.1D
D13	DIODETERM	17.3C
D14	DIODEZXXX	81.4A
D15	DIODETERM	17.2D
D16	DIODETERM	12.1A
D17	DIODETERM	17.3A
D18	DIODETERM	17.3B
D19	DIODETERM	17.3D
D20	DIODETERM	17.2C
D21	DIODE	9.2B
D23	DIODE	20.1C
D24	DIODEZXXX	32.1C
FU1	FUSEPTC	26.3B
FU2	FUSEPTC	13.2A
FU3	FUSEPTC	9.2A
IND1	IND	13.2B

IND2	IND	12.3C
IND3	IND	12.3C
IND4	IND	20.2C
IND5	IND	81.3C
IND6	IND	81.4B
IND7	IND	17.2B
IND8	IND	17.2C
J1	ABT125	1.3C 15.2C 15.3A 15.3C 84.1B
J2	SBM89C100	1.1B 8.2B
J3	SBS89C105	1.3B 16.2B
J4	GAL16V8	2.4B
J5	VSI	1.1C 31.2C
J6	LCA40XX_PQ100	1.2C 19.1B
J7	ABT125	15.1C 15.3A 15.3D
J8	ABT273	30.3D
J9	DIS7SEG	30.1A
J10	ABT125	15.1D 15.2D 17.3C 35.4C
J11	ABT245	15.1B
J12	FACT74	3.2B 84.2B
J13	FACT14	15.1C 15.1D 15.2C 15.2D 15.3C 15.3D
J14	FACT14	15.1A 15.1D 15.2D 15.3D 15.4A 38.1A
J15	LANAM7992B	12.2B
J16	FACT14	8.4C 15.3A 15.4A
J17	LANPTRANS	12.2C
J19	FACT74	17.3C 84.3B
J20	POWLVR	81.1A
J21	EPROM1765	19.1D
J22	FEPROM8XK	21.1D
J23	ABT125	32.2C 33.4C
J24	ABT245	33.2C
J25	CPUMB86904	2.2C
J26	FEPROM8XK	21.2D
J27	FACT14	20.2B 23.2B 23.3B 26.3B 84.1B
J28	ABT125	33.1C
J29	FACT14	32.3C 32.3C 32.3C 32.4C 32.4C
J30	ABT125	34.1C 34.3C
J31	SV6TL7705A	30.1B
J32	ABT245	37.3C 37.4C
J33	ABT245	36.1C 36.2C
J34	ABT245	37.2C 37.3C
J35	ABT245	36.2C 36.3C
J36	ABT245	37.2C
J37	ABT245	37.1C 37.2C
J38	ABT245	36.2C 36.2C 36.3C 36.3C 36.3C
J39	ABT125	32.3C 33.2C
J40	ABT245	35.1C 35.2C
J41	ABT125	34.1C 34.2C 34.4C
J42	ABT125	34.1C 34.2C 34.3C
J43	ABT245	36.2C 36.3C 36.3C 36.4C
J44	ABT244	44.2A 44.3A 44.3C
J45	ABT244	44.2A 44.3A 44.4A
J46	ABT125	34.1C 34.2C
J47	ABT125	33.3C 33.4C 35.4C
J48	ABT125	33.1C 35.3C
J49	ABT125	32.1C 32.2B 33.3C 35.3C
J50	ABT245	35.1C 35.2C
J51	RTCMK48T08	20.1C
J52	ABT125	23.2C 23.3C
J53	GAL16V8	42.2C
J54	ABT245	35.2C
J55	ABT244	29.1C 29.2C 29.3C
J56	GAL16V8	35.3B
J57	FEPROM8XH	22.1D
J58	FEPROM8XH	22.2D
J59	FH002	24.2C

J60	FH002	23.2C
J61	75189A	23.3D 23.4D 24.3D 24.4D
J62	ABT125	23.2C 23.3C 24.3C
J63	ABT125	21.1B 24.2C 24.3C
J64	ABT125	22.2B 24.2C 24.3C 24.3C
J65	FACT14	24.2B 24.3B 26.1B 26.2B
J66	ABT373	18.2C
J67	ABT373	18.1C
J68	ABT373	18.1C
J69	MAX734	20.2C
J71	ABT245	15.1B
J72	ABT244	30.3C 44.3C
J73	CONSOCDIL14	20.2C
J100	DRAMSPEZ1	45.1B
J101	DRAMSPEZ1	46.1B
J102	DRAMSPEZ1	47.1B
J103	DRAMSPEZ1	48.1B
J104	DRAMSPEZ1	49.1B
J105	DRAMSPEZ1	50.1B
J106	DRAMSPEZ1	51.1B
J107	DRAMSPEZ1	52.1B
J108	DRAMSPEZ1	53.1B
J109	DRAMSPEZ1	54.1B
J110	DRAMSPEZ1	55.1B
J111	DRAMSPEZ1	56.1B
J112	DRAMSPEZ1	57.1B
J113	DRAMSPEZ1	58.1B
J114	DRAMSPEZ1	59.1B
J115	DRAMSPEZ1	60.1B
J116	DRAMSPEZ1	61.1B
J117	DRAMSPEZ1	62.1B
J200	DRAMSPEZ1	63.1B
J201	DRAMSPEZ1	64.1B
J202	DRAMSPEZ1	65.1B
J203	DRAMSPEZ1	66.1B
J204	DRAMSPEZ1	67.1B
J205	DRAMSPEZ1	68.1B
J206	DRAMSPEZ1	69.1B
J207	DRAMSPEZ1	70.1B
J208	DRAMSPEZ1	71.1B
J209	DRAMSPEZ1	72.1B
J210	DRAMSPEZ1	73.1B
J211	DRAMSPEZ1	74.1B
J212	DRAMSPEZ1	75.1B
J213	DRAMSPEZ1	76.1B
J214	DRAMSPEZ1	77.1B
J215	DRAMSPEZ1	78.1B
J216	DRAMSPEZ1	79.1B
J217	DRAMSPEZ1	80.1B
MH1	HOLE	85.3A
MH2	HOLE	85.3A
MH3	HOLE	85.2B
MH4	HOLE	85.4A
MH5	HOLE	85.2B
MH6	HOLE	85.3A
MH7	HOLE	85.1A
MH8	HOLE	85.3A
MH9	HOLE	85.1A
MH10	HOLE	85.2A
MH11	HOLE	85.2A
MH12	HOLE	85.4A
MH13	HOLE	85.2A
MH14	HOLE	85.3A
MH15	HOLE	85.1A
MH16	HOLE	85.2B
MH41	MH4	85.1B
MH42	MH4	85.1B

MH43	MH4	85.1B
MH44	MH4	85.1B
N1	RESNV16SO	10.3C 24.2B 24.3B 24.4B 30.2A 30.2C 30.3A 84.2A
N2	RESNV16SO	42.2A 42.3A 42.4A
N3	RESNV16SO	7.1B 7.1D 7.2D
N4	RESNV16SO	7.1B 29.2B 30.2A 30.3A 35.2B 35.3B 42.2A 42.3A 42.3B
N5	RESNV16SO	15.1B 15.2A 15.2B 15.2C 15.4A 84.3A
N6	RESNV16SO	15.1B 15.1D 15.2B 15.2D 15.3D 15.4A 19.3C 84.3A
N7	RESNV16SO	42.2A 42.3A 42.4A
N8	RESNV16SO	42.1A 42.2A 42.2A 42.3A 42.4A
N9	RESNV16SO	7.1B 7.1D 7.2D 18.4C 23.3B 24.3B 29.3B
N10	RESNV16SO	42.2A 42.2A 42.3A 42.4A
N11	RESNV16SO	7.1D 7.2D 19.4D 23.2B 23.3B 29.1B 29.2B 35.1B 35.4B 42.2B
N12	RESNV16SO	7.2D 18.3C 18.4C 23.4B 35.1B 35.1D 35.2B 35.2D 35.3B 42.2B 84.3B
N13	RESNV16SO	7.1B 7.1D 7.2D 19.3D 30.3A 42.3B 84.2A
P0	CONVGZABCD_MNPR0	39.1C
P1	CONVGZABCD_MNPR1	40.1C
P2	CONVGZABCD_MVME2	41.1C
P3	CONSBUS_FSBU	1.2C 5.1A
P4	CONSBUS_FSBU	1.2C 4.1A
P5	CONPCS40_FNPR	6.1A
P6	CONPCS40_FNPR	6.2A
P7	CONDSUB15_FETH	1.1A 14.1D
P8	CONHALFP50	1.1A 11.1D
P9	CONHALFP26_FSER	1.3A 25.1C
P10	CONMDIN8	1.3A 27.1C
P11	CONSPROMAD4000	19.1C
P100	CONMODA60	43.2C
P101	CONMODA60	43.2C
P102	CONMODA60	43.2B
P103	CONMODA60	43.2B
Q1	QUARZ506	12.3B
Q2	QUOMTTLN	3.2A
Q3	QUARZ506	17.1C
Q4	QUARZ506	17.1D
Q5	QUARZ506	17.1D
Q6	QUOMTTLN	17.4A
Q7	QUOHCMOSPL	3.1A
Q8	QUOHCMOSPL	17.3A
Q9	QUARZ506	17.1B
R1	RESEG	15.4B
R2	RESEV	30.1B
R3	RESEV	3.3B
R4	RESEV	28.2D
R5	RESEV	3.2A
R6	RESEV	30.1C
R7	RESEV	3.3C
R8	RESEG	38.2A
R9	RESEG	15.1C
R10	RESEG	15.3C
R11	RESEV	38.3A
R12	RESEV	38.3A
R13	RESEV	38.3A
R14	RESEV	38.3B
R15	RESEV	38.3B
R16	RESEV	38.3A
R17	RESEV	38.3B
R18	RESEV	38.3A
R19	RESEV	38.3B

DESIGNER:

FORCE COMPUTERS

APR/25/95 Manual

CROSS REFERENCE

SHEET 90 OF 94

REV. 0.2

R20	RESEV	38.1A
R21	RESEV	38.3B
R22	RESEV	38.3B
R23	RESEV	38.4D
R24	RESEV	38.2B
R25	RESEV	38.3B
R26	RESEV	38.3B
R27	RESEV	38.3B
R28	RESE	81.1B
R29	RESEV	17.3C
R30	RESE	81.3C
R31	RESE	81.2B
R32	RESE	81.1B
R33	RESE	2.2A
R34	RESE	81.1B
R35	RESE	38.2D
R36	RESE	81.4A
R37	RESE	81.2B
R38	RESEG	17.3D
R39	RESE	81.1B
R40	RESE	2.2A
R41	RESEG	7.1B
R42	RESE	81.1B
R43	RESE	2.2A
R44	RESE	81.1B
R45	RESEG	7.3D
R46	RESE	81.1B
R47	RESE	81.1B
R48	RESEV	17.3D
R49	RESEG	84.2D
R50	RESEV	7.1B
R51	RESEV	7.2B
R52	RESEG	7.1B
R53	RESEV	7.3B
R54	RESEV	7.2B
R55	RESEV	7.2B
R56	RESEV	7.2B
R57	RESEV	7.2B
R58	RESEG	84.1D
R59	RESEV	7.2B
R60	RESEV	7.2B
R61	RESEG	12.1B
R62	RESEV	7.2B
R63	RESEV	7.2B
R64	RESEV	12.1B
R65	RESEV	7.2B
R66	RESEV	7.1B
R67	RESEV	84.2D
R68	RESEV	7.2B
R69	RESEV	16.3D
R70	RESEV	84.1D
R71	RESEV	16.3D
R72	RESEV	7.2B
R73	RESEG	17.3C
R74	RESEV	16.3D
R75	RESEG	17.3C
R76	RESEV	7.3B
R77	RESEV	7.3D
R78	RESEV	7.3B
R79	RESEV	7.1B
R80	RESEV	7.3B
R81	RESEV	7.2B
R82	RESEG	17.2D
R83	RESEG	17.3C
R84	RESEG	17.3B
R85	RESEG	17.2B
R86	RESEG	12.1A

R87	RESEG	17.4A
R88	RESEV	12.1B
R89	RESEV	17.2D
R90	RESEV	17.3B
R91	RESEV	17.4A
R92	RESEV	17.3C
R93	RESEV	17.2B
R94	RESEV	17.3A
R95	RESEG	81.2B
R96	RESE	81.2B
R97	RESE	32.2C
R98	RESEG	19.3C
R99	RESEV	31.4A
R100	RESEV	31.4A
R101	RESEV	31.4A
R102	RESEG	33.4B
R103	RESE	44.3D
R104	RESE	44.3D
R105	RESE	44.4D
R106	RESEG	37.4B
R107	RESEV	84.2B
R108	RESEV	84.1B
R109	RESEV	44.3C
R110	RESEG	20.2B
R111	RESEV	32.4D
R112	RESEV	84.2B
R113	RESEV	32.3C
R114	RESEV	32.3C
R115	RESEV	32.3C
R116	RESEV	32.3C
R117	RESEV	84.1B
R118	RESEV	84.2B
R119	RESEV	84.2B
R120	RESEV	32.3C
R121	RESE	35.3C
R122	RESEV	32.3B
R123	RESEV	33.2D
R124	RESEV	33.2B
R125	RESEV	22.2B
R126	RESE	44.3D
R127	RESE	44.4D
R128	RESE	44.4D
R129	RESEG	34.4B
R130	RESEG	32.4B
R131	RESEV	18.4C
R132	RESEV	22.3B
R133	RESEV	32.1B
R134	RESEV	32.2B
R135	RESEV	32.1B
R136	RESEG	44.4A
R137	RESEG	35.4B
R138	RESEG	36.4B
R139	RESEG	19.3C
R140	RESE	23.4C
R141	RESE	24.4C
R142	RESE	23.1D
R143	RESE	24.1D
R144	RESE	26.1C
R145	RESE	26.1C
R146	RESE	26.1B
R147	RESE	26.3B
R148	RESE	26.2B
R149	RESE	26.1B
R150	RESE	3.2B
R151	RESE	12.2C
R152	RESE	12.2C
R153	RESE	12.2C

R154	RESE	12.3A
R155	RESE	12.3B
R156	RESE	12.2C
R157	RESE	12.3C
R158	RESE	12.3C
R159	RESE	12.3C
R160	RESE	12.3C
R161	RESE	43.1A
R162	RESEG	85.1D
R163	RESE	3.1B
R164	RESE	42.2C
R165	RESE	20.3C
R166	RESE	20.4D
R167	RESE	21.4C
R168	RESE	21.4C
R169	RESE	21.2B
R170	RESE	21.1B
R171	RESEG	18.3B
R172	RESE	2.4B
R173	RESE	2.4B
R174	RESE	2.4B
R175	RESE	2.4B
R176	RESE	2.4B
R177	RESE	30.1B
R178	RESE	30.1B
R179	RESE	2.4B
R180	RESE	2.4B
R181	RESE	30.1A
R182	RESE	2.4B
R183	RESE	30.1A
R184	RESE	30.1A
R185	RESE	30.1B
R186	RESE	30.1A
R187	RESE	38.1C
R188	RESE	30.1B
R189	RESE	11.2B
R190	RESE	11.3B
R191	RESE	11.2B
R192	RESE	11.3B
R193	RESE	11.3B
R194	RESE	11.3B
R195	RESE	11.3B
R196	RESE	11.3B
R197	RESE	11.3B
R198	RESE	38.2D
R199	RESE	11.3B
R200	RESE	11.2B
R201	RESE	11.3B
R202	RESE	11.2B
R203	RESE	11.3B
R204	RESE	11.2B
R205	RESE	11.3B
R206	RESE	11.3B
R207	RESEG	2.2D
R208	RESEG	2.2D
R209	RESEV	84.1B
R210	RESE	81.2B
R211	RESEG	2.2B
R212	RESE	17.1B
R213	RESEV	2.4A
R214	RESEV	2.2D
R215	RESEV	2.3D
R216	RESEV	2.2D
R217	RESE	17.2B
R218	RESEG	29.1B
R219	RESEV	2.2B
R220	RESE	3.2C

R221	RESE	17.2B
R222	RESE	30.1D
R223	RESEV	18.4C
R224	RESE	30.1D
R225	RESE	30.1A
R226	RESE	30.1D
R227	RESEV	11.2C
R228	RESE	3.2C
R229	RESE	30.2C
R230	RESE	30.2C
R231	RESE	30.1C
R232	RESE	30.1D
R233	RESEV	20.1B
R234	RESE	38.2D
R235	RESE	38.1C
R236	RESE	38.1C
R237	RESE	11.2B
R238	RESEV	17.3C
R239	RESE	30.1A
R240	RESE	2.3B
R241	RESEG	31.4A
R242	RESE	38.1C
R243	RESE	38.1A
R244	RESE	11.2B
R245	RESEG	2.3A
R246	RESEG	31.4A
R247	RESEG	19.3B
R248	RESE	38.2C
R249	RESEG	44.1C
R250	RESE	38.2C
R251	RESEG	2.3A
R252	RESEG	2.4A
R253	RESE	38.1C
R254	RESE	38.2D
R255	RESEV	2.3A
R256	RESEG	31.4A
R257	RESEV	2.3A
R258	RESEG	15.3B
R259	RESEV	2.3B
R260	RESE	32.1C
R261	RESEV	19.2B
R262	RESEV	11.2C
R263	RESEV	22.3A
R264	RESE	83.1D
R265	RESE	28.3D
R266	RESE	83.1D
R267	RESEV	10.2C
R268	RESE	15.3D
R269	RESE	28.3C
R270	RESE	15.2D
R271	RESE	15.1D
R272	RESE	83.1D
R273	RESE	83.1D
R274	RESEV	21.2A
R275	RESEV	21.2B
R276	RESEG	3.3C
R277	RESEG	29.2B
R278	RESEV	29.1B
R279	RESEV	29.2B
R280	RESEG	19.4C
R281	RESE	81.4A
R282	RESE	32.1D
R283	RESE	32.2B
R284	RESE	32.1C
R285	RESEG	32.2B
R286	RESEG	32.2D
R287	RESEG	32.2C

DESIGNER:

FORCE COMPUTERS

APR/25/95 Manual

CROSS REFERENCE

SHEET 91 OF 94

REV. 0.2

R288	RESEG	29.1B
R289	RESEG	29.1B
R290	RESE	19.1C
R291	RESE	19.1C
R292	RESEG	82.2C
R293	RESEG	82.2C
R294	RESEG	82.2C
R295	RESE	33.4B
R296	RESE	82.2C
R297	RESEG	82.2C
R298	RESEV	19.1C
R299	RESEV	42.4A
R300	RESE	34.1B
R301	RESE	34.1B
R302	RESE	34.1B
R303	RESE	34.1B
R304	RESE	34.2B
R305	RESE	34.1B
R306	RESE	34.1B
R307	RESE	32.2B
R308	RESE	32.3B
R309	RESE	32.1B
R310	RESEV	82.1C
R311	RESEV	82.1C
R312	RESEV	82.1C
R313	RESEV	82.1C
R314	RESEV	82.1C
R315	RESEV	35.4B
R316	RESEV	35.4B
R317	RESE	42.2C
R318	RESE	42.1C
R319	RESE	42.2C
R320	RESE	42.2C
R321	RESE	17.1C
R322	RESE	17.2C
STI1	STIF137F3	85.1C
STI2	STIF23MA3	85.1D
SW1	SWI_TFT	30.3A
SW2	SWI_TFT	30.4A
SW3	SWIROTDILS	30.2B
SW4	SWSMD12RO4	1.3A 2.4A 24.2B 24.3B
SW5	SWSMD12RO4	1.3A 19.3C 23.2B 23.3B
SW6	SWSMD12RO4	10.3C 11.3D 21.2A 22.2A
SW7	SWSMD12RO4	30.2A 30.3A 32.1B 32.1C
SW8	SWSMD12RO4	19.3B 19.3C 19.4C 30.2A
T1	SCSITERMTI	11.3C
T2	SCSITERMTI	11.2C
T3	SCSITERMTI	10.2C
T4	SCSITERMTI	10.2C
T5	TRANSINPN	32.1D
T6	TRANSINPN	32.2C
T7	TRANSINPN	32.2C
T8	TRANSINPN	32.2C
ZFO1	CODIER16D	85.3D
ZFO2	CODIER16D	85.2D
ZFO3	CODIER16D	85.3D
ZFO5	FRONTPANEL	85.1B
ZFO6	IDENT	85.1B
ZFO7	IDENT	85.1C
ZFO8	MANUAL	85.2C
ZPCB1	PCBDOPEURO	85.1B

DESIGNER:		SHEET 92 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	CROSS REFERENCE	REV. 0.2

HISTORY FOR SOURCE SCHEMATIC CHANGES

Changes from CPU-5V Revision 0.0 to Revision 0.1:
(File: ../../doc/changes/history_0_1)

Standard serial PROM connection implemented:
-> serial PROM counter reset
-> reset key applicable
Changes on SH 18 J6.

Signal _JTAG_RST, _JTAG_TMS, JTAG_CLK, JTAG_TDI_TDO<4> pulled up with 10k and pulled down with 33R (33R NA).
Changes on SH 81 B4.

Signals EB_A<21..20> driven by LCA for Flash PROMs.
Changes on SH 17 J68, SH 18 J6.

CAS<2> and CAS<3> swapped and CAS<0> and CAS<1> swapped.
Changes on SH 1 J25.

PA<22> and PA<26> swapped for MACIO.
Changes on SH 7 J2.

SB_PA<15> SB_PA<20> SB_PA<26> connected with the correct pins of the SBus connectors.
Changes on SH 3 J4, SH 4 P3, SH 5 P6.

SP_SEL<1..0> generated in PAL.
Note: The signals named SP_SEL in the schematic have been renamed automatically to SP_SEM in the layout for historical reasons.
Changes on SH 1 J25, SH 28 J55, SH 34 J56, SH 42 adapter.

Boot Size Switch connected with correct LCA pin:
Changes on SH 18 J6.

P100 and P101 exchanged with P102 and P103 (height).
Changes on SH 42 P100, P101, P102, P103.

Position of LEDs L1 and L2 exchanged.
Changes on SH 29 D1.

Fuses replaced with PTCs.
Changes on SH 8, SH 12, SH 25.

Signals EB_D<3..0> connected with LCA instead of EB_D<7..4>.
Changes on SH 18 J6.

DRAM Data Bus pulled.
Changes on SH 42.

SYSRESETin through LCA to FGA-5000.
Changes on SH 18 J6, SH 30 J5.

SYSRESETout from LCA in addition.
Changes on SH 18 J6, SH 31 T7.

SYSRESETout driven by 3 transistors.
Changes on SH 18 J6, SH 29 J31, SH 30 J5, SH 31 J49.

Signals VSI_TEST, VSI_TESTASYNC and VSI_TESTSCAN pulled up and down.
Changes on SH 30 J5.

Signals ACFAIL, SYSFAIL, SYSRESET, BBSY, IRQ<7..0> glitch filtered.
Changes on SH 31, SH 32, SH 33.

5 row VMEbus connector P0 pins disconnected.
Changes on SH 38.

5 row VMEbus connector P1: Z only GND, D disconnected.
Changes on SH 39.

5 row VMEbus connector P2, Centronics: D1 .. D19 shifted to D3 .. D21.
Changes on SH 40.

Geographical address lines connected with CPs.
Changes on SH 31.

GND for serial interface connected with hybride GND.
Changes on SH 24.

Additional SMD footprint for Q6, VSI_CLK: QUOM Q8.
Changes on SH 16.

Disable VBERRI and VRETRYI when not being VMEbus master.
Changes on SH 18, 30, 34.

Delay negation of VMEbus signal BERR* when not being VMEbus master.
Changes on SH 18, 30, 34.

Additional Block Cs installed.
Changes on SH 82.

Voltage division resistors for 3.5V changed from 0805 to 1206.
Changes on SH 80.

MIN_LINE_WIDTH=8 addet to VMEbus signals.
Changes on SH 31.

Holes entered into schematics.
Changes on SH 84.

Replace LT1085_3V3 with LT1085_ADJ.
Changes on SH 80 J20, R96, R210.

RAS<0,2,4,6> assertion delayed.
Changes on SH 43.

Relais removed.
Changes on SH 19 J70, D22, R164.

Quom 80 MHz is used only for FGA-5000, J18 (FACT74) removed.
Changes on SH 16.

Clock signal for SCSI generated with new quartz Q9 instead of FACT74.
Changes on SH 16.

Signal ENET_TCLK used for SLAVIO signal TIM_CLK (10 MHz).
Changes on SH 7, 83.

MIN_LINE_WIDTH of quartz signals changed from 8 mil to 10 mil.
Changes on SH 16.

Signals ENET_TCLK and ENET_RCLK terminated on the destination end.
Changes on SH 11, 16, 83.

Changes from CPU-5V Revision 0.1 to Revision 0.2:
(File: ../../doc/changes/history_0_2)

Bus _AMIA<3..0> added.
Changes on SH 1.

History sheet number updated.
Changes on SH 1.

Serial PROM XC1736 replaced with XC1765.
Changes on SH 19 J21.

SYSFAILO connected with LCA.
Changes on SH 19 J6.

Parts added in order to support third overtone crystals.
Changes on SH 17 Rx, Rx, Rx, Cx, INDx.

DRAM_CODE_A<0> connected.
Changes on SH 29 _ADAPT_DRAM_CODE.

AM3VSI controlled by AM3VME, VIACK and VMADIR.
Changes on SH 35 _ADAPT_AMIA, SH 42 J53.

History updated.
Changes on SH 93.

DESIGNER: SK Thu Mar 23 08:09:42 1995	DRAWING ABBREV=HISTORY	SHEET 93 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	HISTORY	REV. 0.2

HISTORY FOR SOURCE SCHEMATIC CHANGES

Changes from CPU-5V Revision 0.1 to Revision 0.2:
(File: ../../doc/changes/history_0_2)

Bus _AMIA<3..0> added.
Changes on SH 1.

History sheet number updated.
Changes on SH 1.

Serial PROM XC1736 replaced with XC1765.
Changes on SH 19 J21.

SYSFAILO connected with LCA.
Changes on SH 19 J6.

Parts added in order to support third overtone crystals.
Changes on SH 17 Q9, R212, R221, R217, C23, IND7,
Q3, R321, R322, C24, IND8.

DRAM_CODE_A<0> connected.
Changes on SH 29 _ADAPT_DRAM_CODE.

AM3VSI controlled by AM3VME, VIACK and VMADIR.
Changes on SH 35 _ADAPT_AMIA, SH 42 J53.

SYSRESET is prohibited with an additional transistor and
resistor when the board is not a slot 1 device.
Changes on SH 32 T8, R97.

Keyboard and mouse are connected with VMEbus P2 by default.
Changes on SH 38 R248 R250 R253.

Resistors for JTAG changed.
Changes on SH 82 R313 R296.

JTAG daisy chain closed.
Changes on SH 1.

Resistor value changed for RAS timing.
Changes on SH 42 R320.

Capacitor removed.
Changes on SH 81 CE13.

Socket for RTC changed.
Changes on SH 28 J51.

Sockets for GALs added.
Changes on SH 2 J4, SH 35 J56.

PCB changed to Rev. 0.2.
Changes on SH 85 ZF04.

Stiffener body changed.
Changes on SH 85 ST11.

Parameter name changed: SB_SELECT instead of SB_SLOT.
Changes on SH 1, SH 2, SH 4, SH 5, SH 6, SH 7, SH 8,
SH 16, SH 18, SH 19, SH 31.

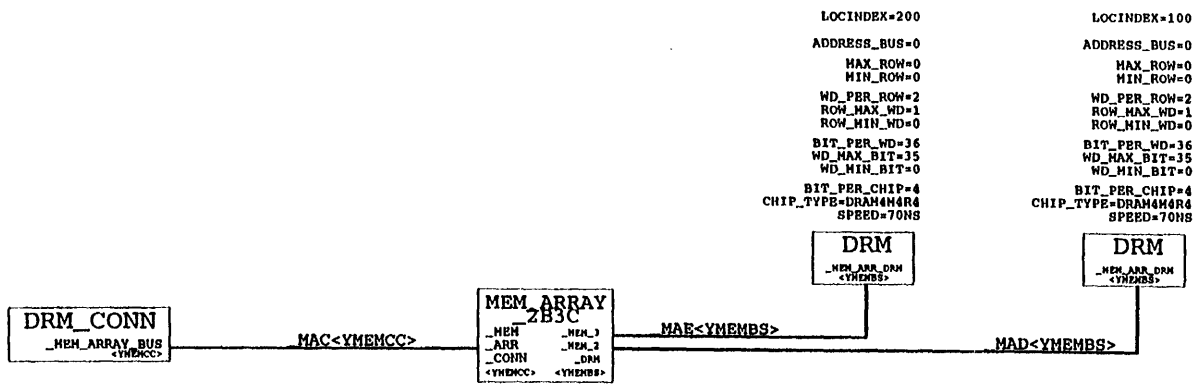
Switch SW 4-4 is connected with Select GAL.
Changes on SH 2 J4, R252, R213.

Switch SW 5-4 is connected with LCA.
Changes on SH 19 J6, R98, N6.

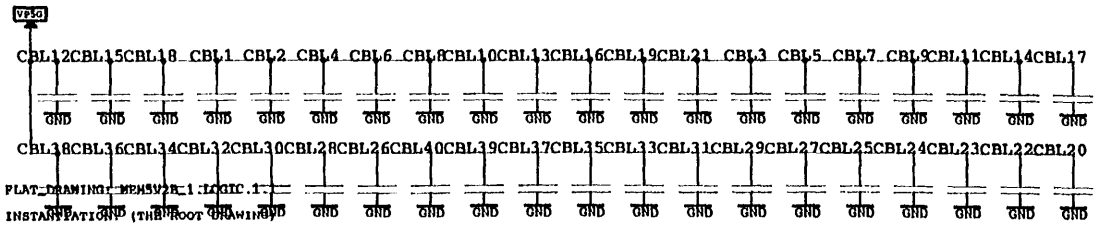
History updated.
Changes on SH 94.

DESIGNER: SK Mon Apr 3 08:12:55 1995	DRAWING ABBREV=HISTORY	SHEET 94 OF 94
FORCE COMPUTERS	APR/25/95 Manual	
	HISTORY	REV. 0.2

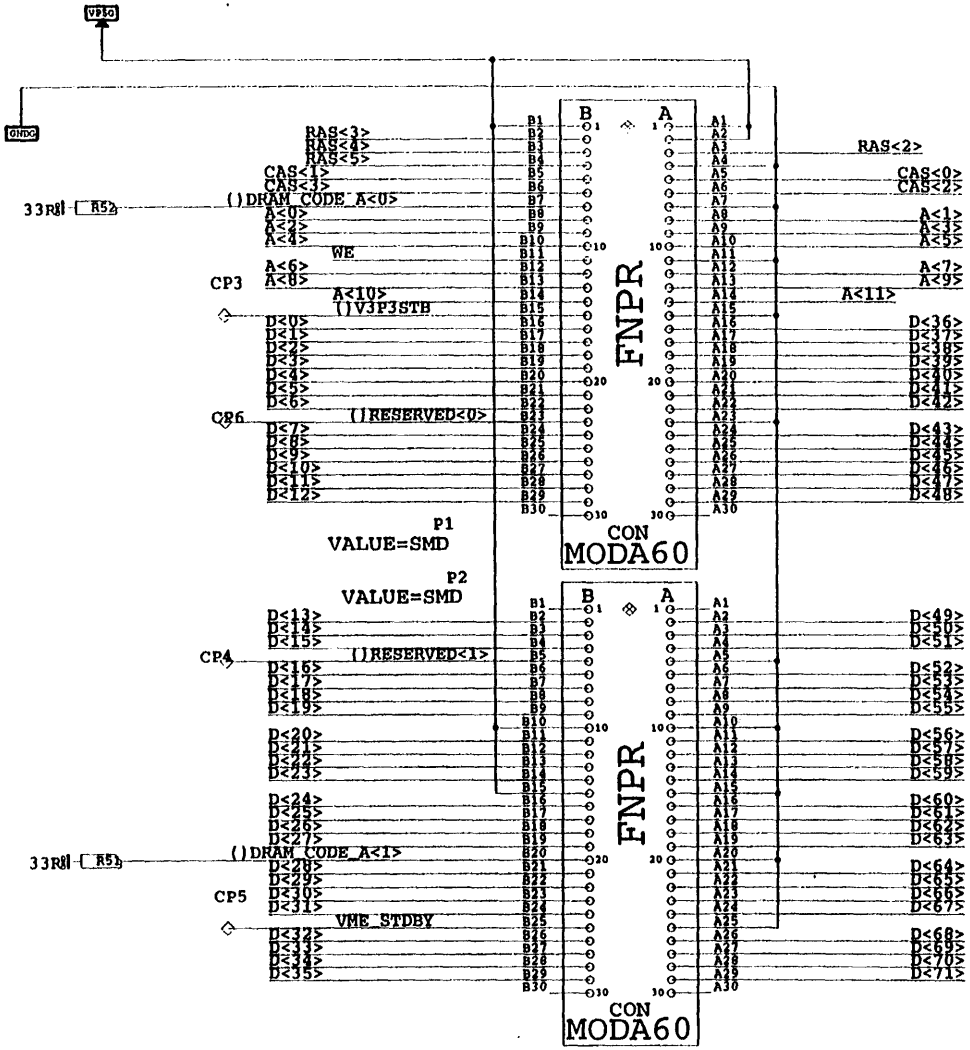
5.1 MEM-5 Schematics



60 NS DRAMS EINTRAGEN!!!



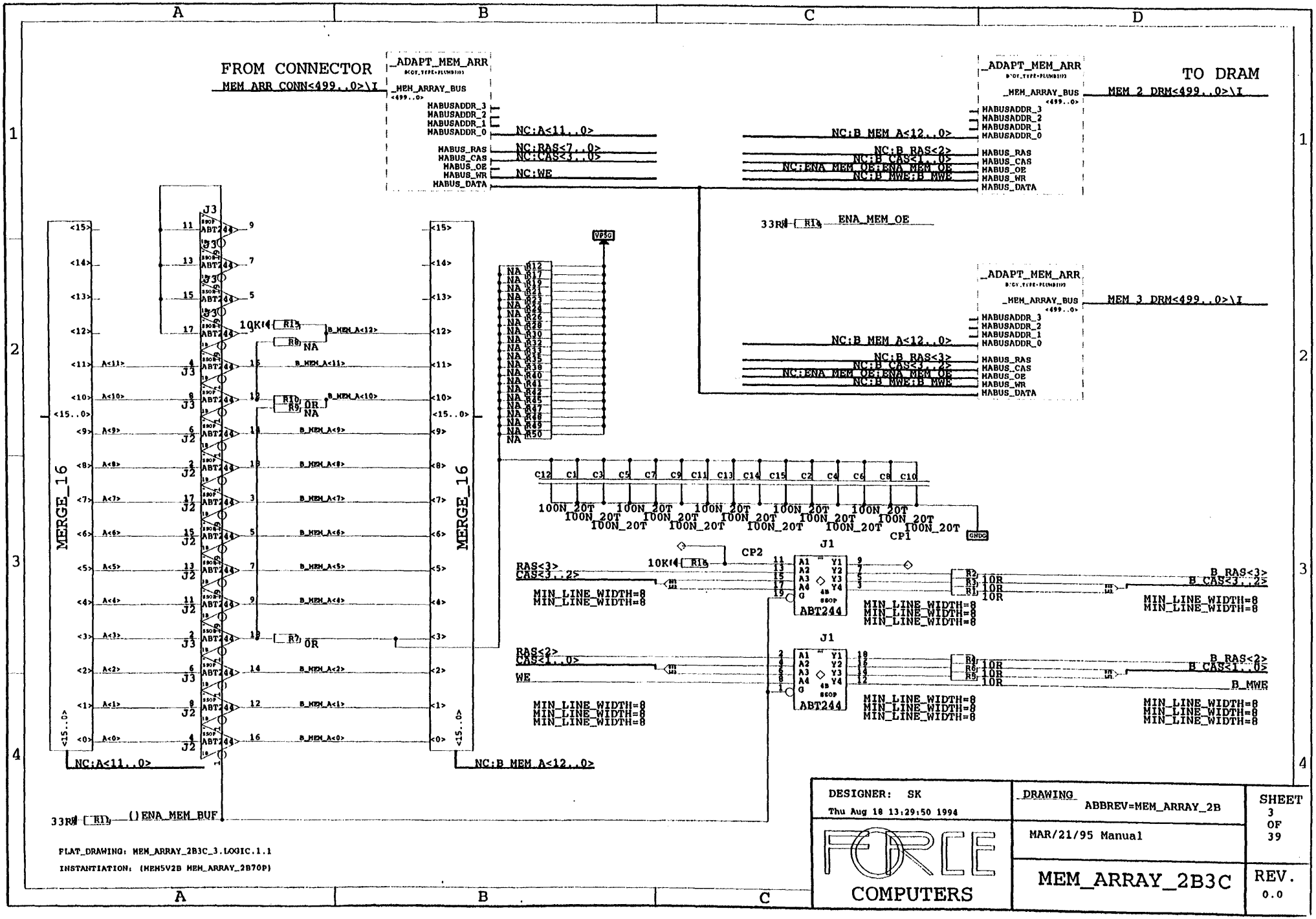
DESIGNER: SK Thu Aug 18 13:29:38 1994	DRAWING ABBREV=MEM5V2B MAR/21/95 Manual	SHEET 1 OF 39
FORCE COMPUTERS	MEM5V2B	REV. 0.0

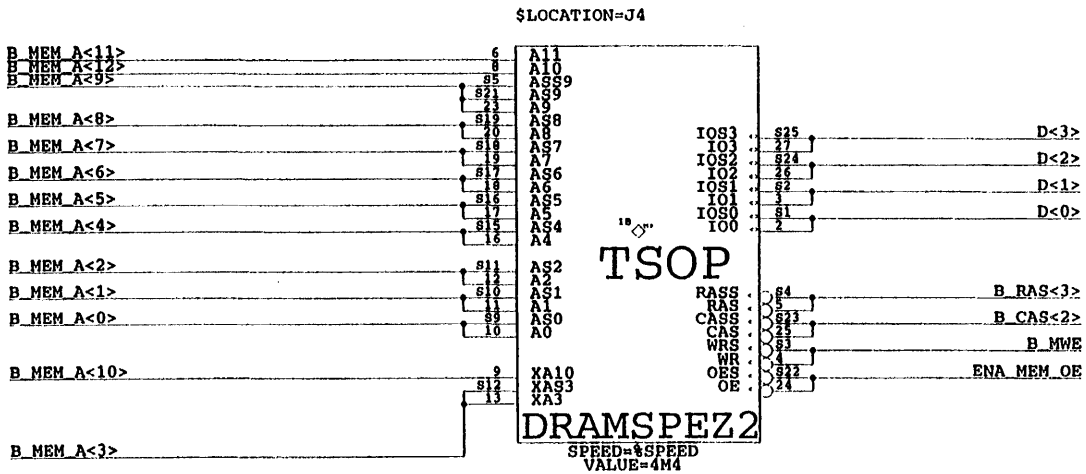


FLAT_DRAWING: DRM_CONN_2.LOGIC.1.1

INSTANTIATION: (HEM5V2B DRM_CONN22P)

DESIGNER: SK Thu Aug 18 13:30:10 1994	DRAWING ABBREV=DRM_CONN	SHEET 2 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRM_CONN	REV. 0.0


FROM CONNECTOR
MEM_ARR_CONN<499..0>\I
_ADAPT_MEM_ARR
_MEM_ARRAY_BUS
<499..0>
HABUSADDR_3
HABUSADDR_2
HABUSADDR_1
HABUSADDR_0
HABUS_RAS
HABUS_CAS
HABUS_OE
HABUS_WR
HABUS_DATA
NC:A<11..0>
NC:RAS<7..0>
NC:CAS<3..0>
NC:WE
NC:B_MEM_A<12..0>
NC:B_RAS<2>
NC:B_CAS<1..0>
NC:ENA_MEM_OE
NC:B_MWE
TO DRAM
MEM_2_DRM<499..0>\I
MEM_3_DRM<499..0>\I
ENA_MEM_OE
ENA_MEM_BUF
MERGE_16
B_MEM_A<12>
B_MEM_A<11>
B_MEM_A<10>
B_MEM_A<9>
B_MEM_A<8>
B_MEM_A<7>
B_MEM_A<6>
B_MEM_A<5>
B_MEM_A<4>
B_MEM_A<3>
B_MEM_A<2>
B_MEM_A<1>
B_MEM_A<0>
10K
33R
NA
OR
VPSG
GNDC
CP1
CP2
J1
ABT244
RAS<3>
CAS<3..2>
RAS<2>
CAS<1..0>
WE
B_RAS<3>
B_CAS<3..2>
B_RAS<2>
B_CAS<1..0>
B_MWE
10R
MIN_LINE_WIDTH=8
DESIGNER: SK
Thu Aug 18 13:29:50 1994
DRAWING
ABBREV=MEM_ARRAY_2B
MAR/21/95 Manual
SHEET 3 OF 39
MEM_ARRAY_2B3C
REV. 0.0
COMPUTERS
FLAT_DRAWING: MEM_ARRAY_2B3C_3.LOGIC.1.1
INSTANTIATION: (MEM5V2B MEM_ARRAY_2B70P)



EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_4.LOGIC.9.1

INSTANTIATION: (MEH5V2B DRM21P DRAM_CHIP3P)

DESIGNER: IV

Thu Aug 18 13:30:14 1994

DRAWING ABBREV=DRAM_CHIP

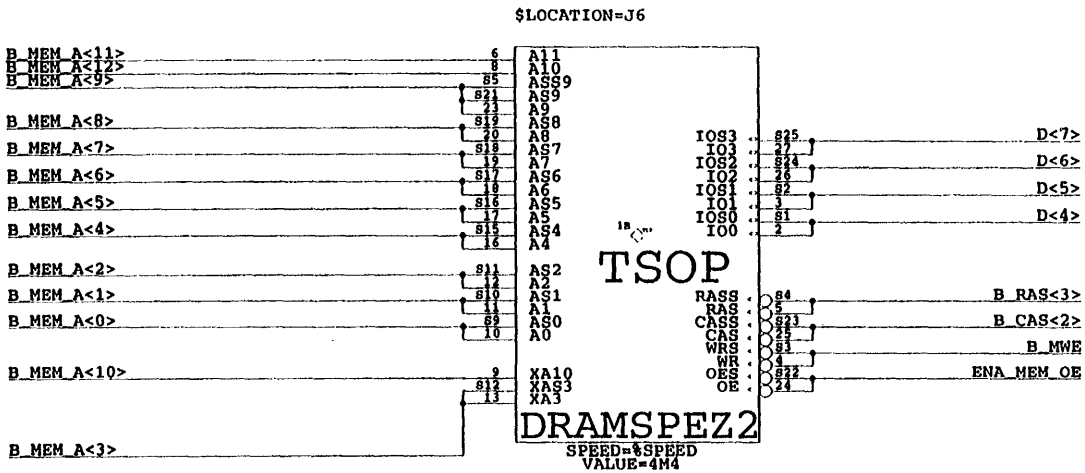
MAR/21/95 Manual

SHEET 4 OF 39

FORCE COMPUTERS

DRAM_CHIP

REV. 0.0

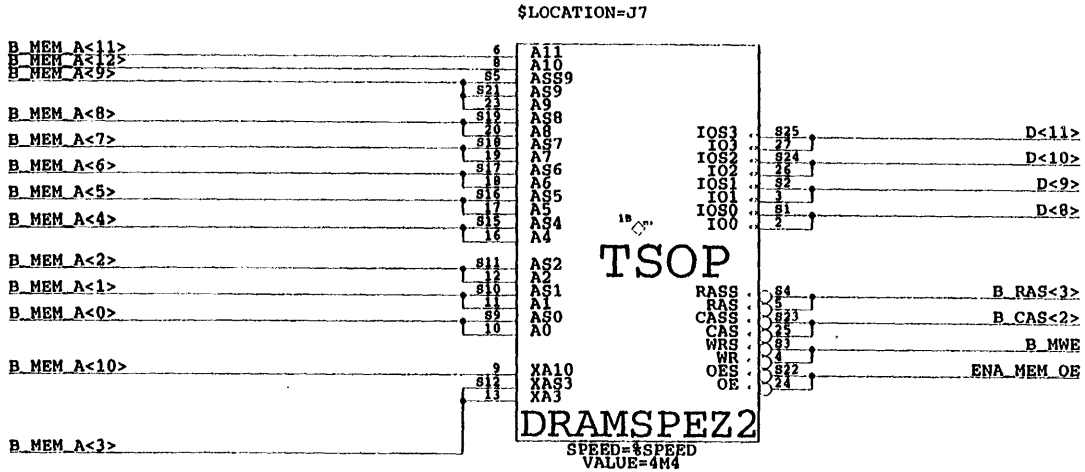


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_5.LOGIC.9.1

INSTANTIATION: (HEM5V2B DRH21P#1 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:39 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 5 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

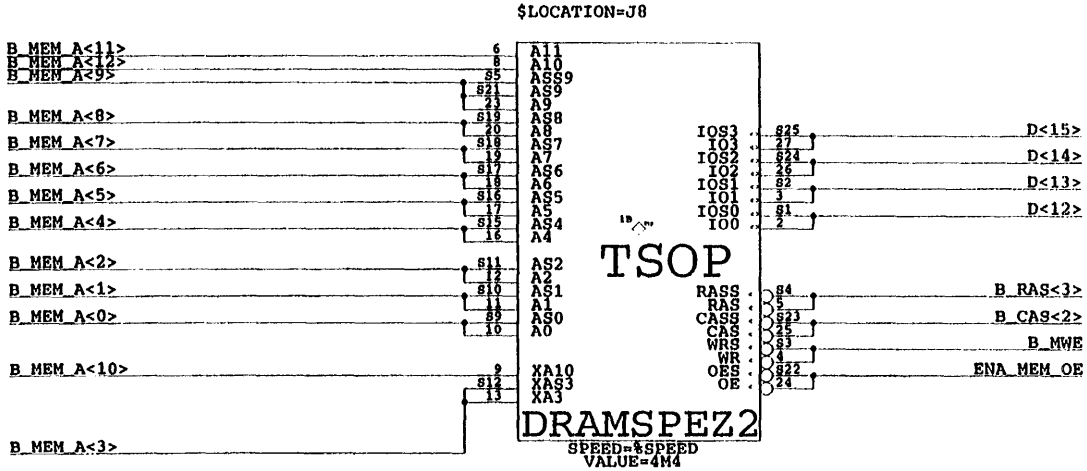


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_6.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P82 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:41 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 6 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

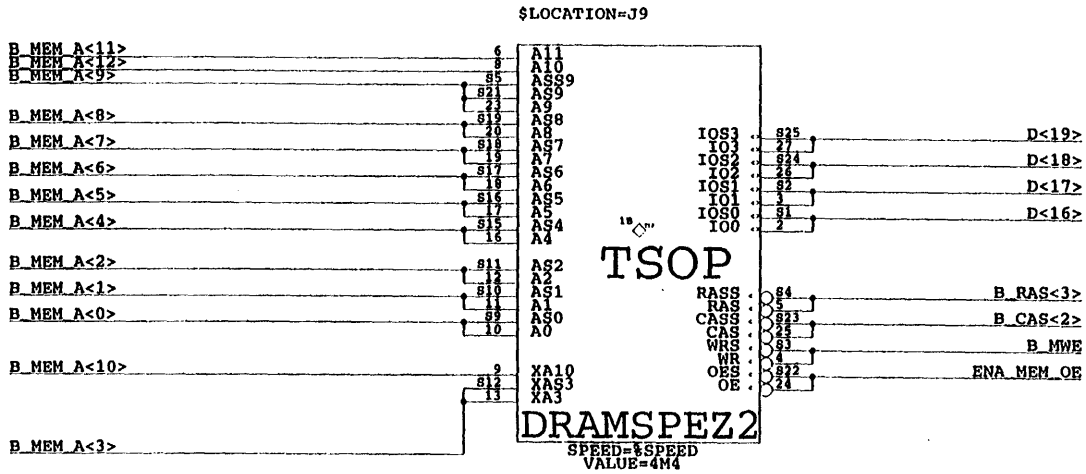


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_7.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P#3 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:43 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 7 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

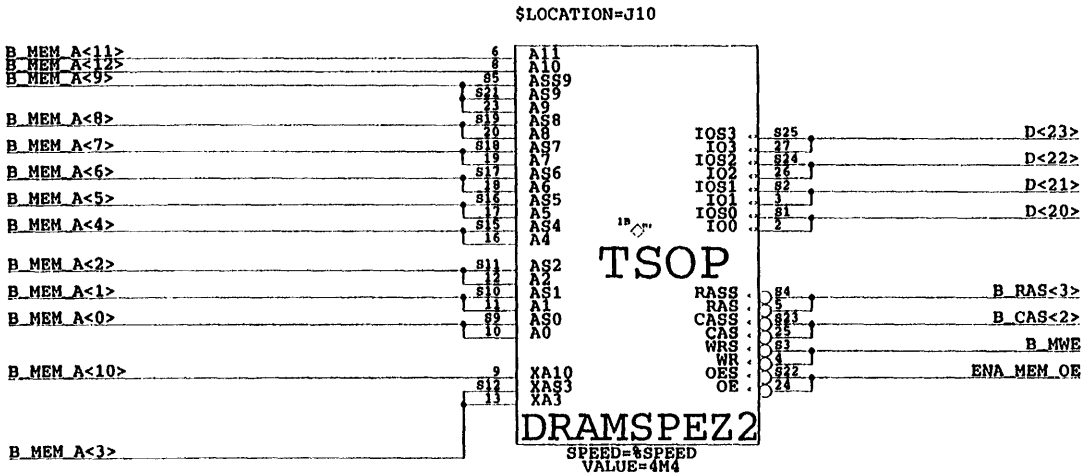


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_8.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P#4 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:44 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 8 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

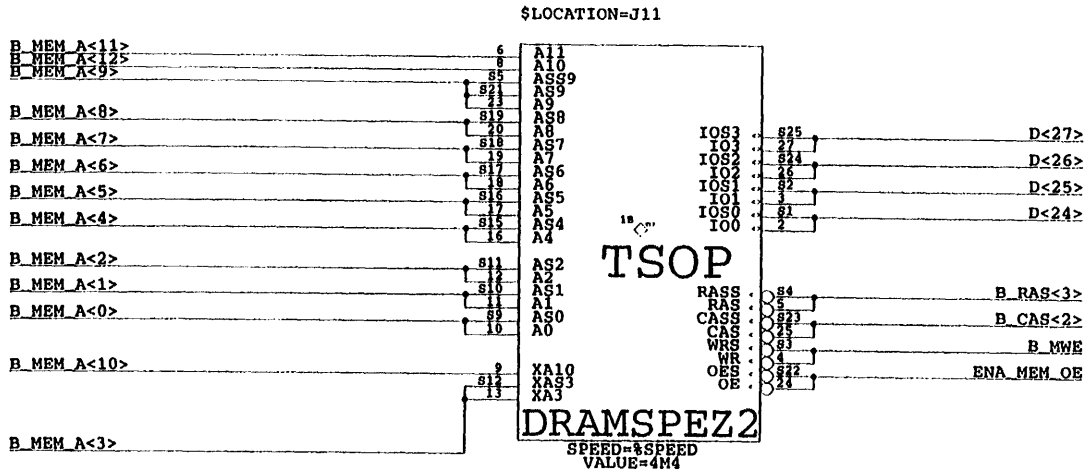


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_9.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P05 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:45 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 9 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

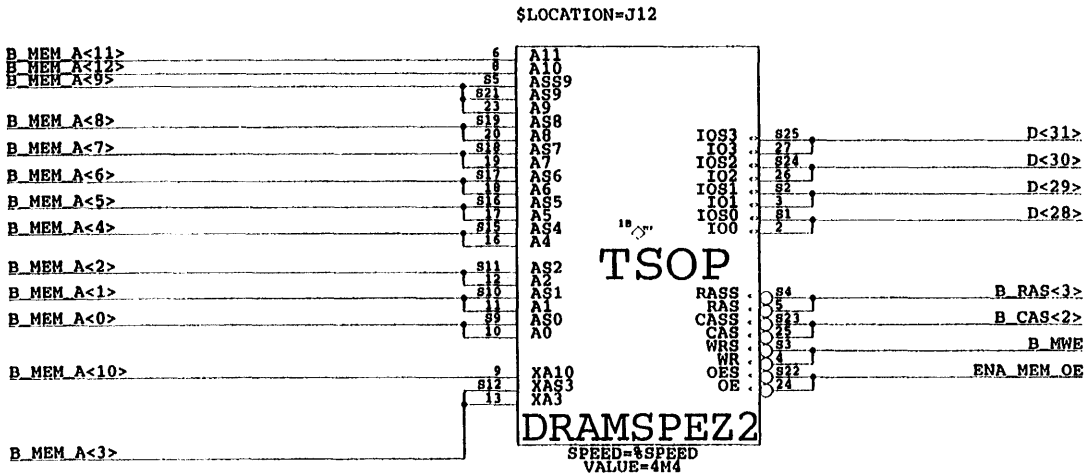


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_10.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P#6 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:47 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 10 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

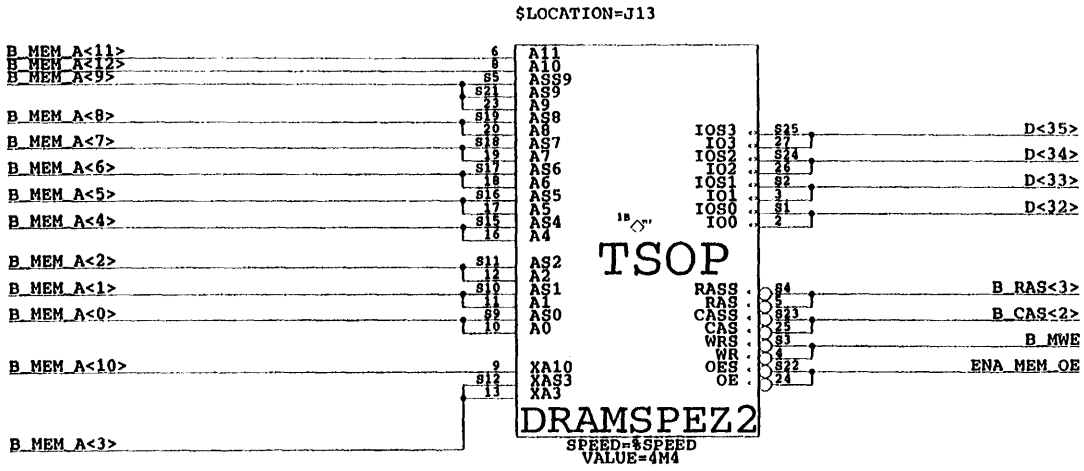


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_11.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P#7 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:52 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 11 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

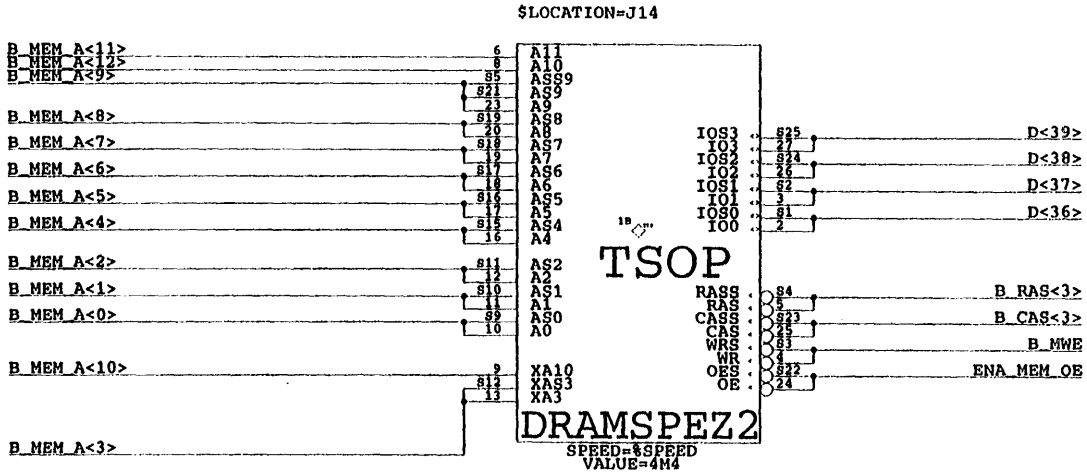


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_12.LOGIC.9.1

INSTANTIATION: (MEH5V2B DRH21P#8 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:54 1994	_DRAWING ABBREV=DRAM_CHIP	SHEET 12 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

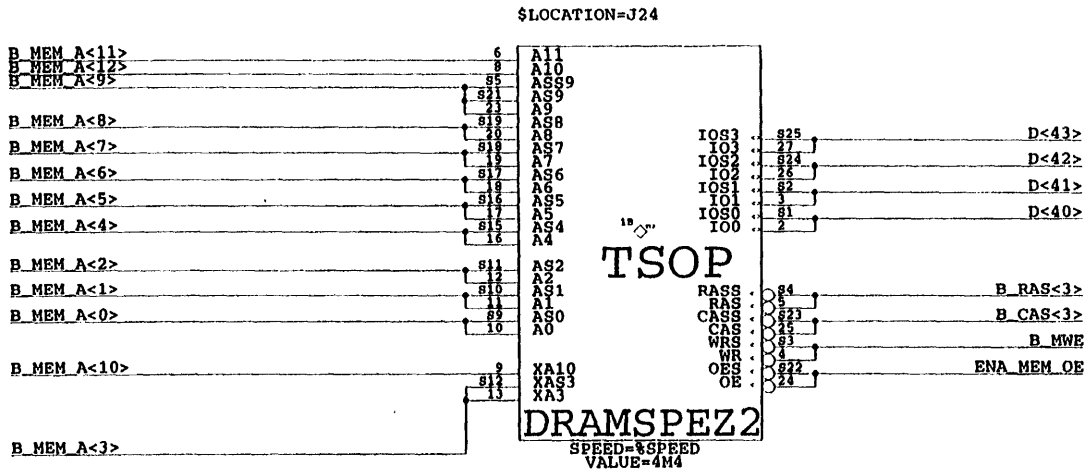


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_13.LOGIC.9.1

INSTANTIATION: (HEM5V2B DRH21P#9 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:56 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 13 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

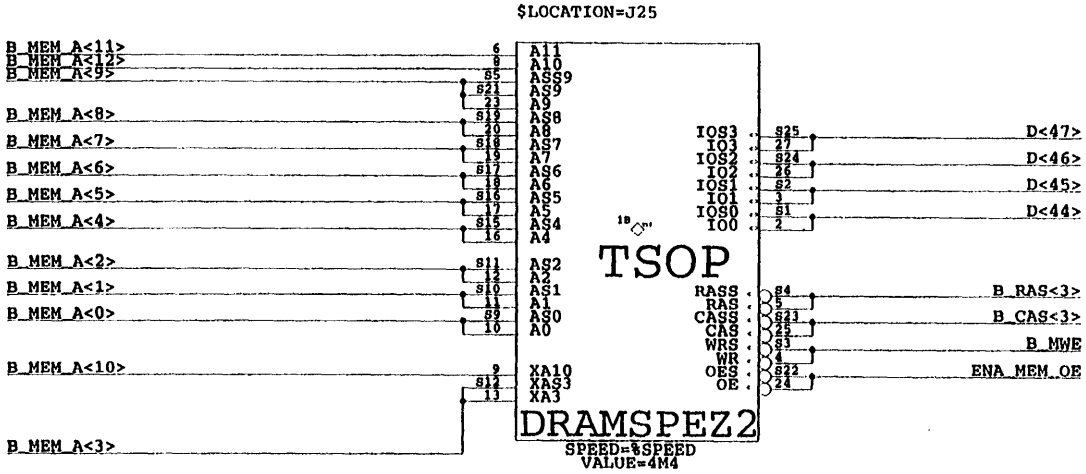


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_14.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH21P#10 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:16 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 14 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0



EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_15.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P811 DRAM_CHIP3P)

DESIGNER: IV

Thu Aug 18 13:29:18 1994

DRAWING ABBREV=DRAM_CHIP

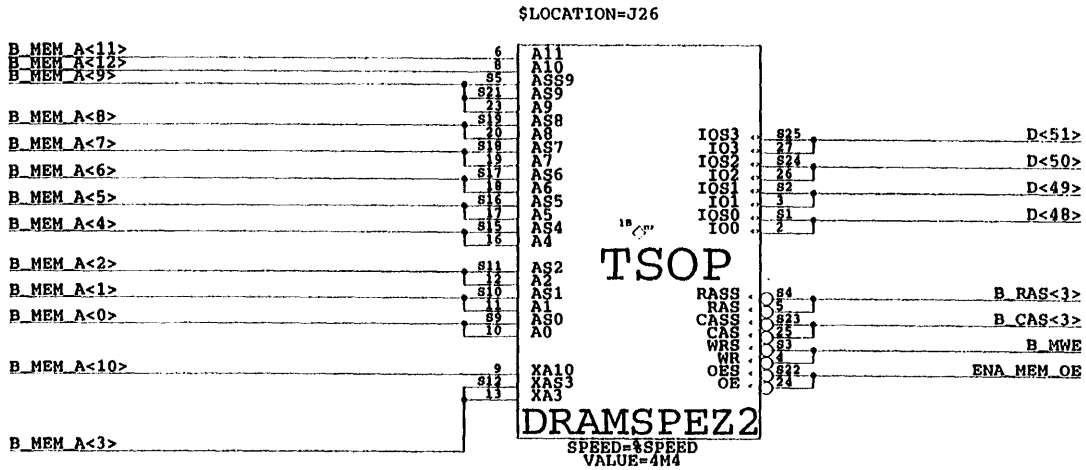
MAR/21/95 Manual

SHEET 15 OF 39

REV. 0.0

FORCE COMPUTERS

DRAM_CHIP

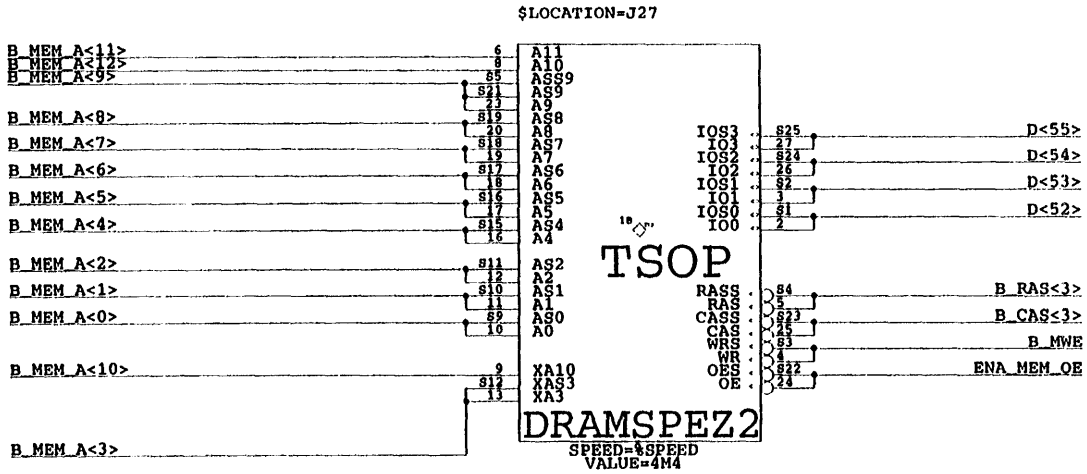


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_16.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH21P#12 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:19 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 16 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

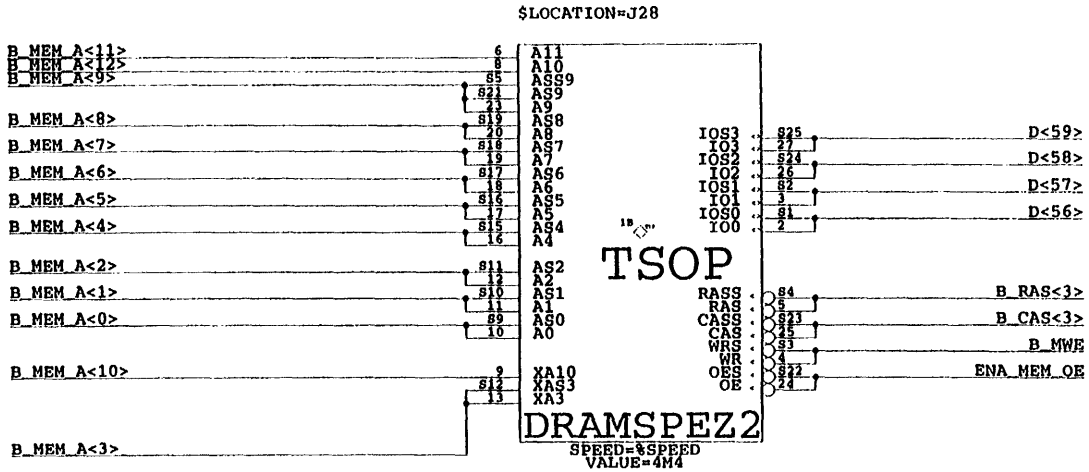


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_17.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH21P#13 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:20 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 17 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

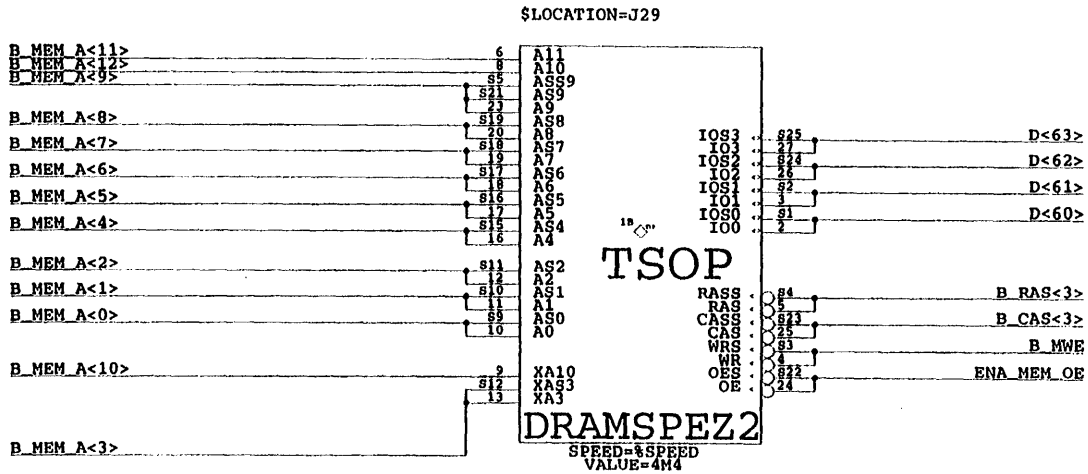


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_18.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH21P814 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:22 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 18 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

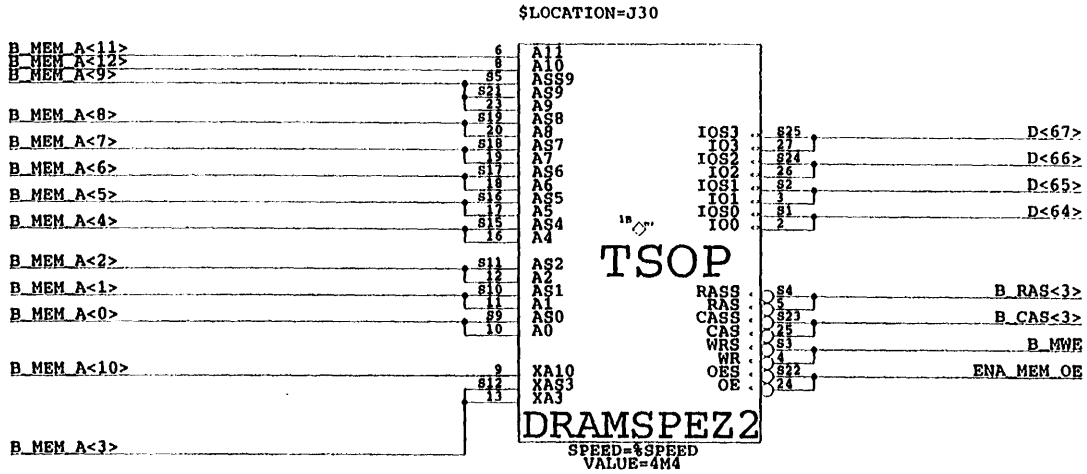


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_19.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM21P#15 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:23 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 19 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

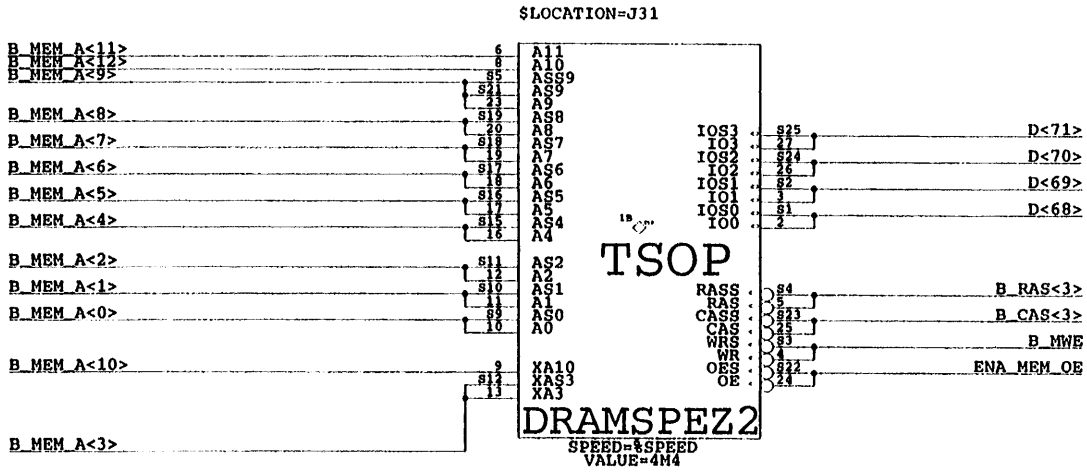


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_20.LOGIC.9.1

INSTANTIATION: (HEM5V2B DRH21P#16 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:24 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 20 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

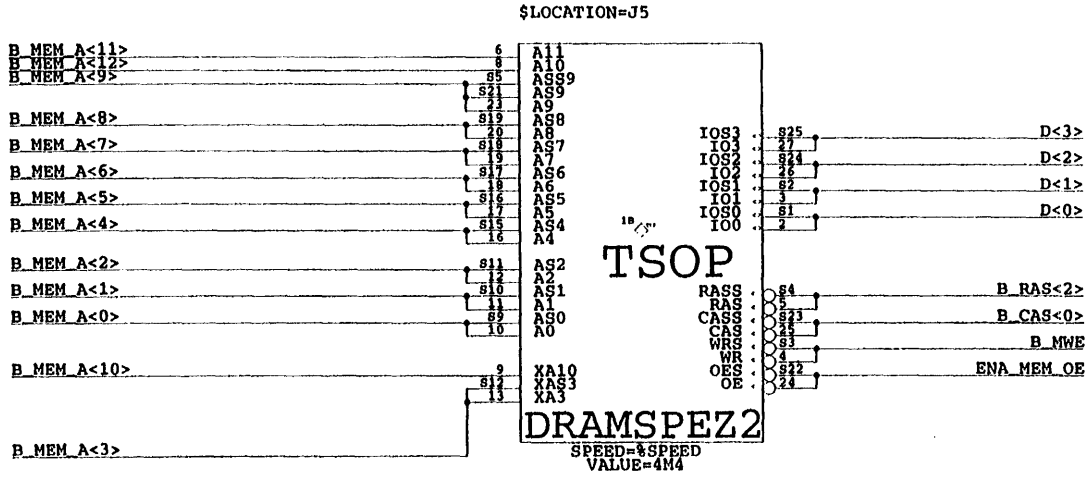


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_21.LOGIC.9.1

INSTANTIATION: (MEH5V2B DRM21P#17 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:26 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 21 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

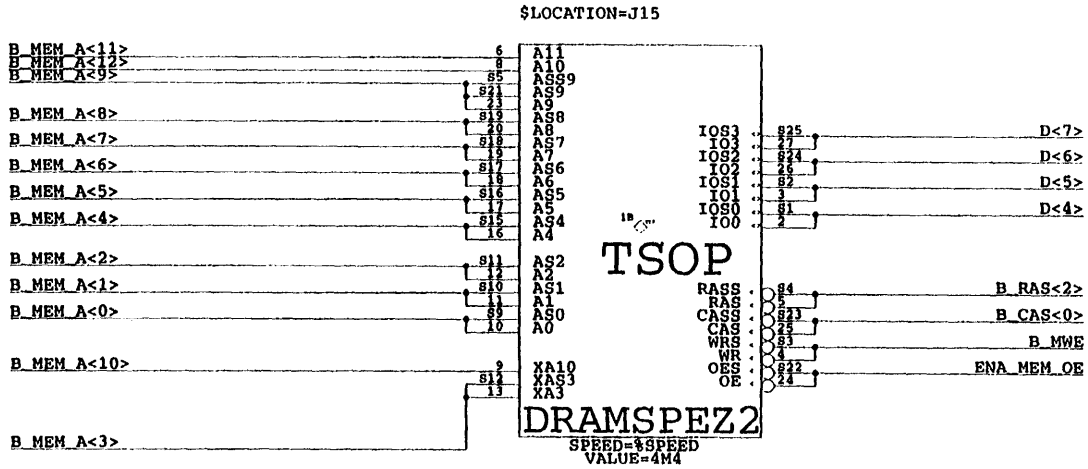


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_22.LOGIC.9.1

INSTANTIATION: (MEH5V2B DRM75P DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:15 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 22 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

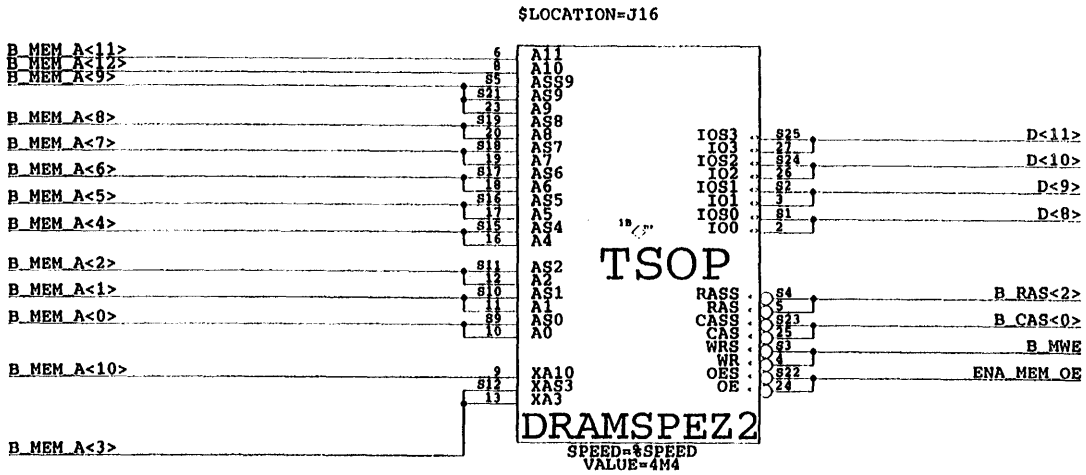


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_23.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P81 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:57 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 23 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

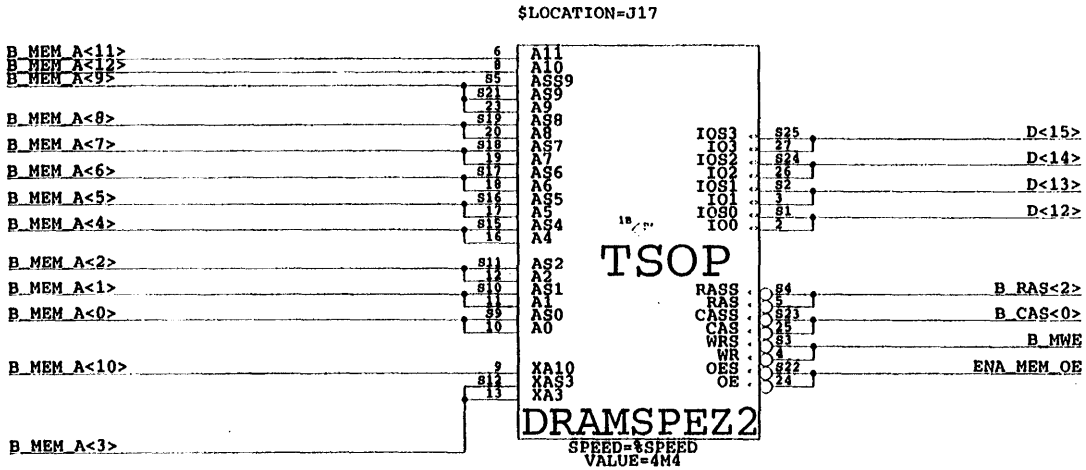


EXPR=(CHIP_TYPE=DRAM4M4R4)

DESIGNER: IV Thu Aug 18 13:29:59 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 24 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

FLAT_DRAWING: DRAM_CHIP_24.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#2 DRAM_CHIP3P)



EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_25.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH75P#3 DRAM_CHIP3P)

DESIGNER: IV

Thu Aug 18 13:30:00 1994

FORCE COMPUTERS

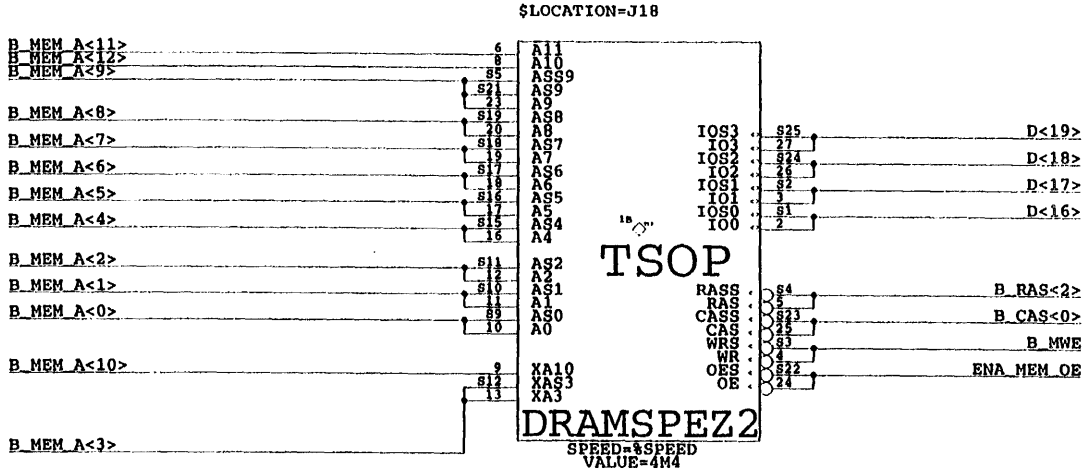
DRAWING ABBREV=DRAM_CHIP

MAR/21/95 Manual

DRAM_CHIP

SHEET 25 OF 39

REV. 0.0



EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_26.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#4 DRAM_CHIP3P)

DESIGNER: IV

Thu Aug 18 13:30:02 1994

DRAWING ABBREV=DRAM_CHIP

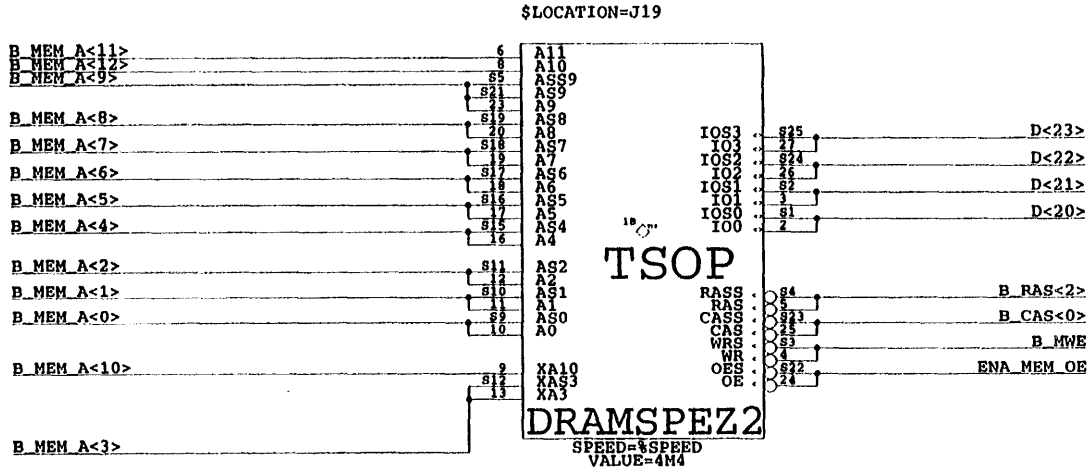
MAR/21/95 Manual

FORCE COMPUTERS

DRAM_CHIP

SHEET 26 OF 39

REV. 0.0

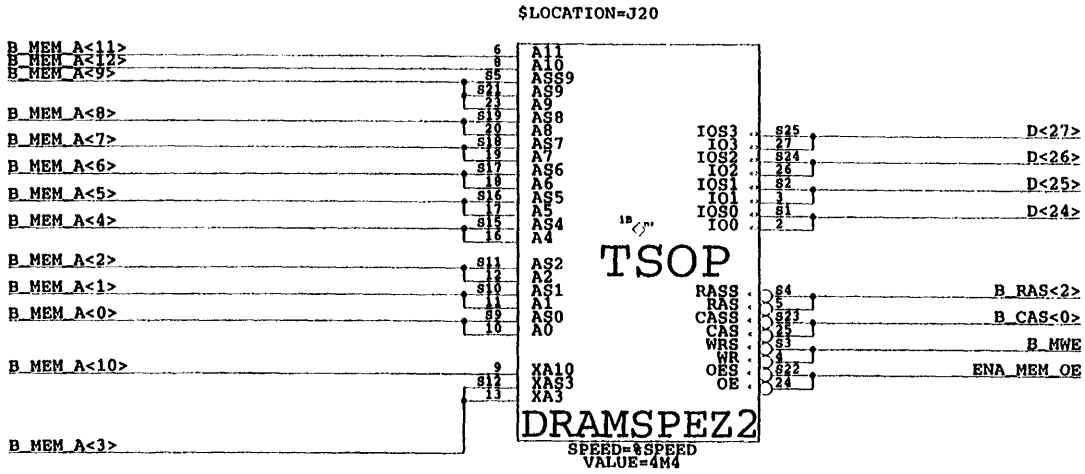


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_27.LOGIC.9.1

INSTANTIATION: (MEH5V2B DRH75P#5 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:30:04 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 27 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

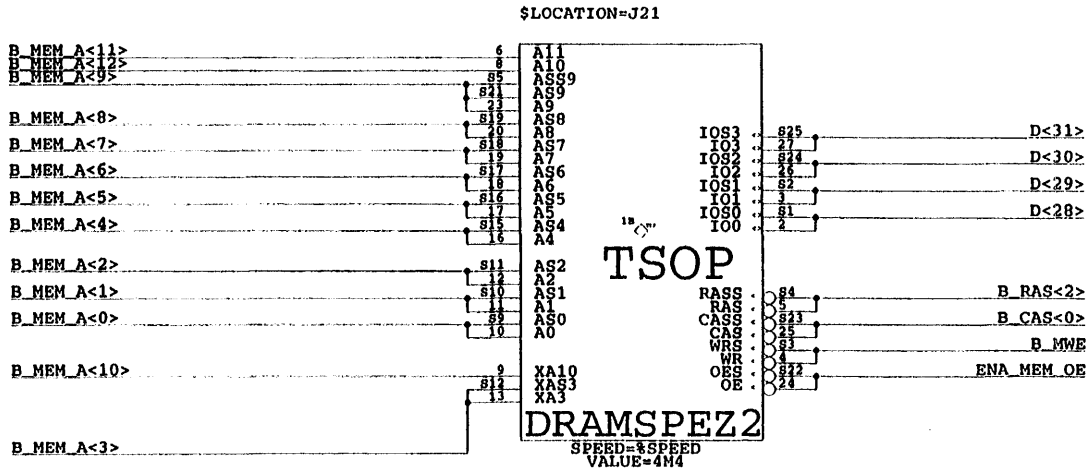


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_28.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P86 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:30:05 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 28 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

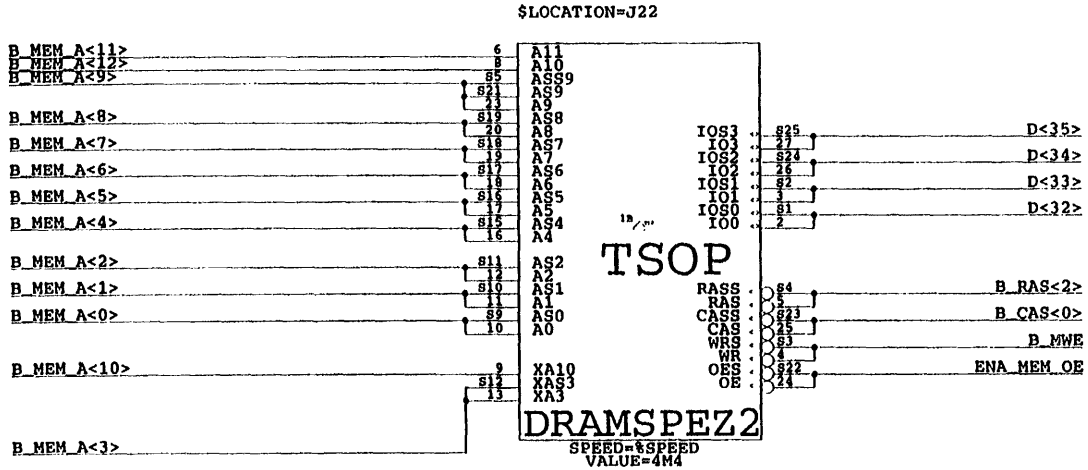


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_29.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#7 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:30:07 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 29 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

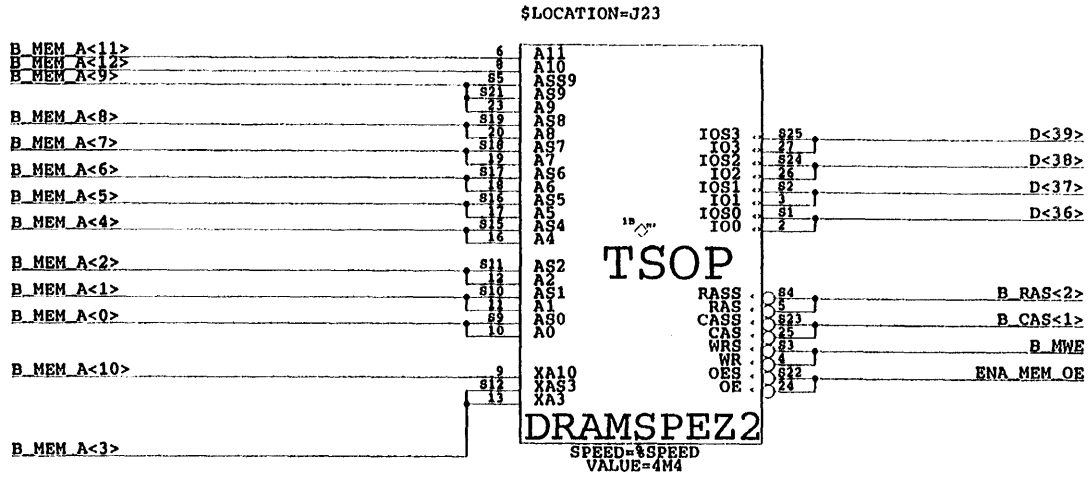


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_30.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH75P88 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:30:12 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 30 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

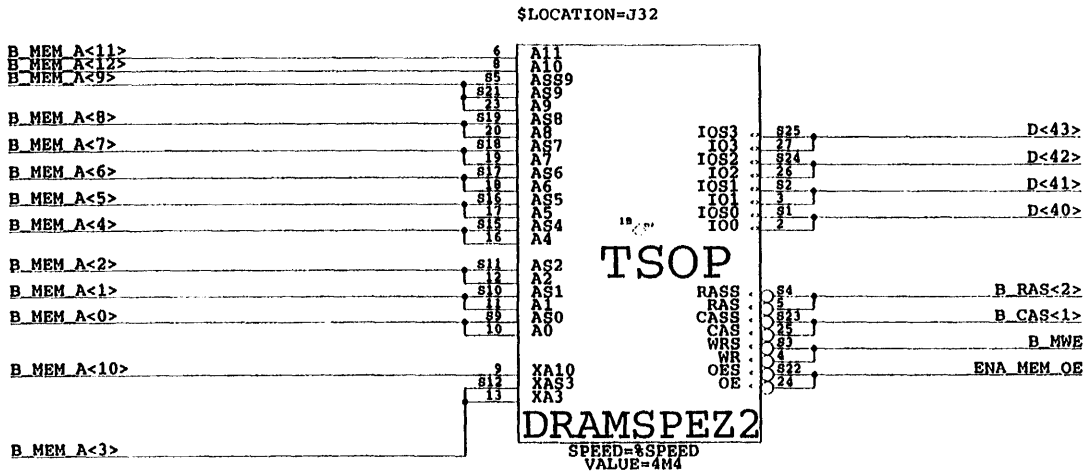


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_31.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#9 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:30:15 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 31 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

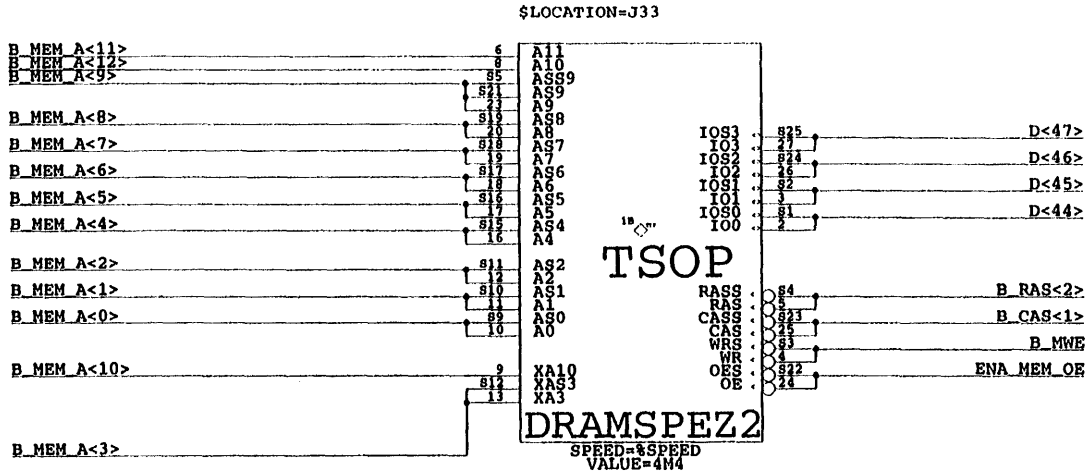


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_32.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P810 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:27 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 32 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	REV. 0.0
	DRAM_CHIP	

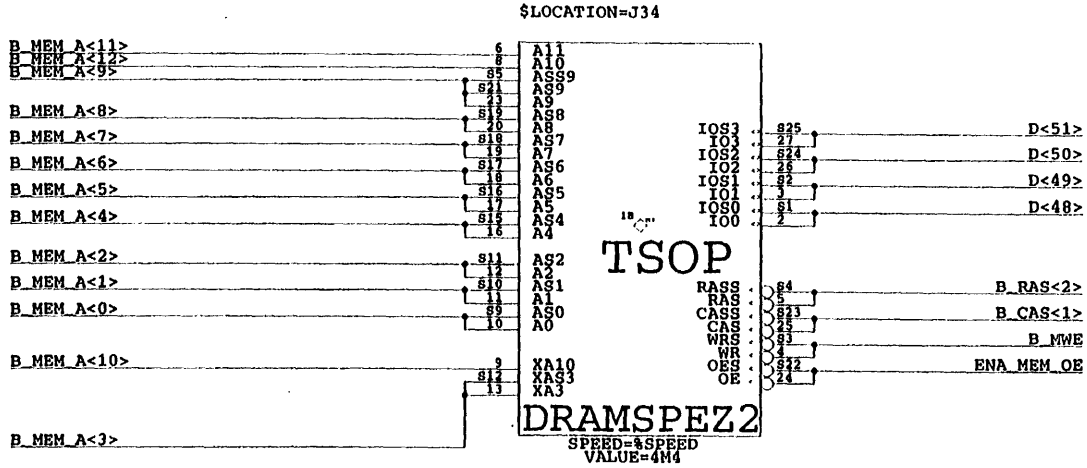


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_33.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P811 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:28 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 33 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

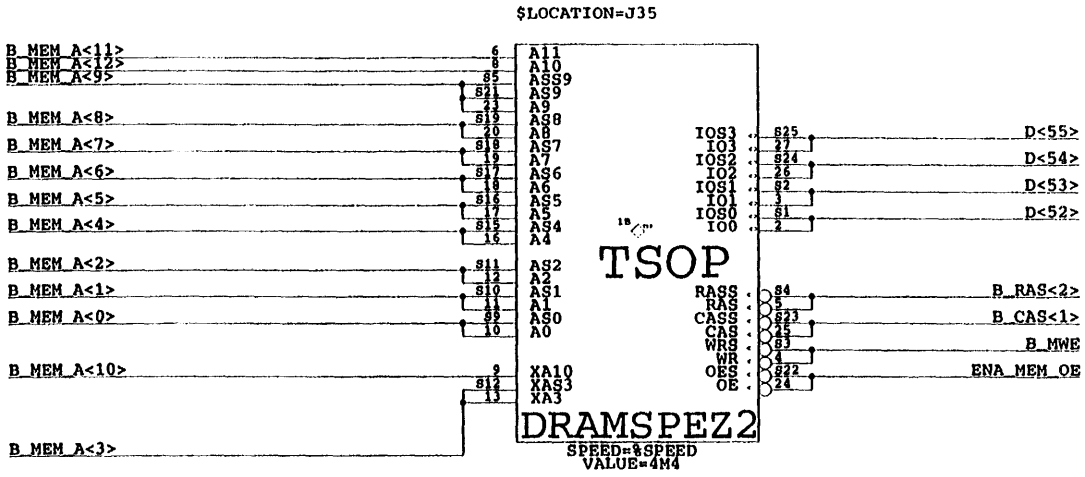


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_34.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#12 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:30 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 34 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

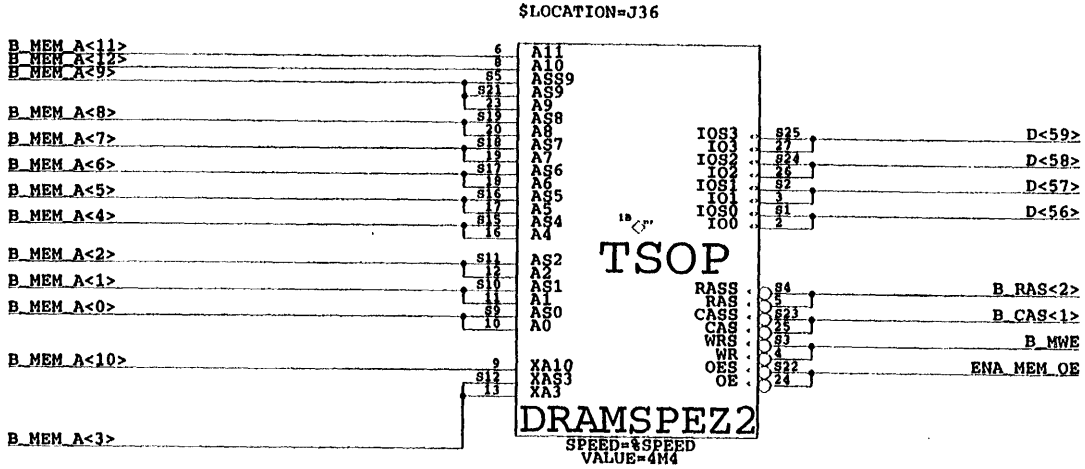


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_35.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#13 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:31 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 35 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

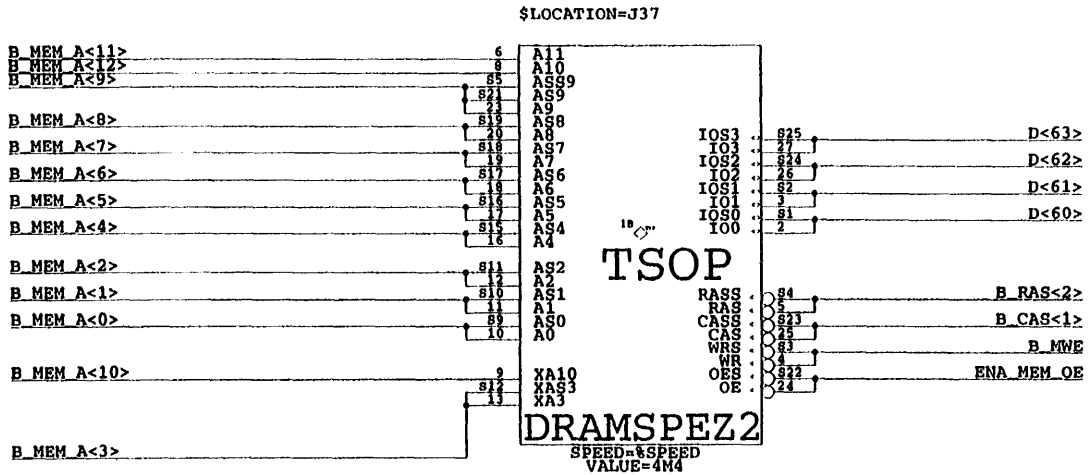


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_36.LOGIC.9.1

INSTANTIATION: (HEH5V2B DRH75P814 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:32 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 36 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

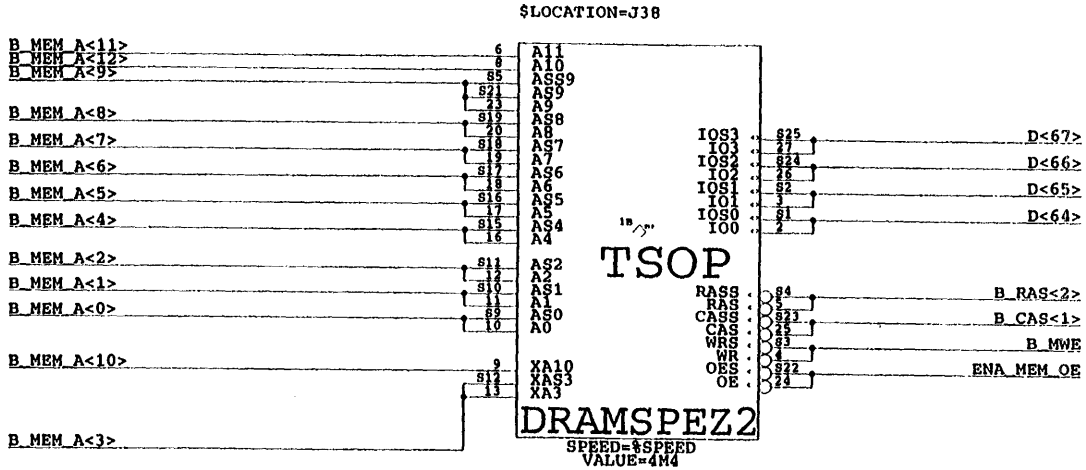


EXPR=(CHIP_TYPE=DRAM4M4R4)

FLAT_DRAWING: DRAM_CHIP_37.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRH75P#15 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:34 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 37 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	REV. 0.0
	DRAM_CHIP	

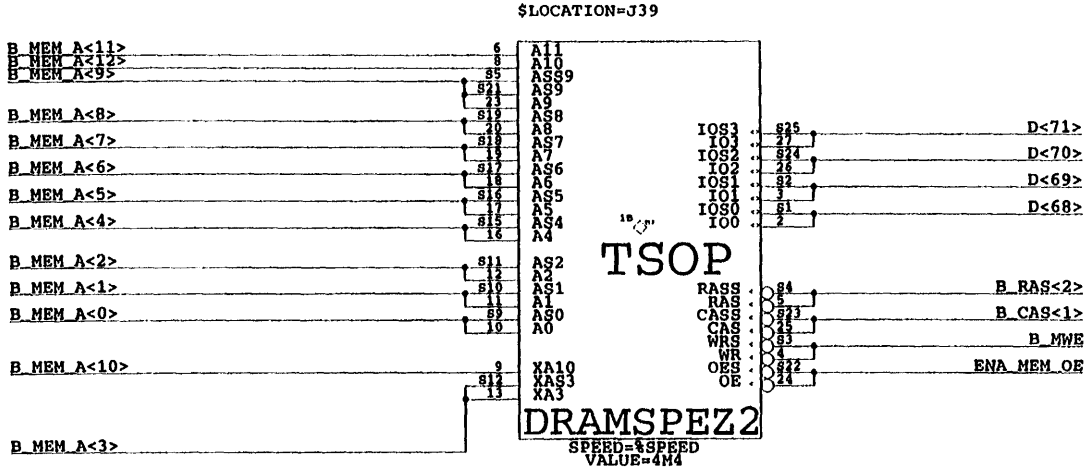


EXPR=(CHIP_TYPE=DRAM4M4R4)

PLAT_DRAWING: DRAM_CHIP_38.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#16 DRAM_CHIP3P)

DESIGNER: IV Thu Aug 18 13:29:35 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 38 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0



EXPR=(CHIP_TYPE=DRAM4M4R4)

DESIGNER: IV Thu Aug 18 13:29:36 1994	DRAWING ABBREV=DRAM_CHIP	SHEET 39 OF 39
FORCE COMPUTERS	MAR/21/95 Manual	
	DRAM_CHIP	REV. 0.0

FLAT_DRAWING: DRAM_CHIP_39.LOGIC.9.1

INSTANTIATION: (MEM5V2B DRM75P#17 DRAM_CHIP3P)

SECTION 6 SUN OPEN BOOT DOCUMENTATION

6. **Insert your** *OPEN BOOT 2.0 PROM MANUAL SET* here.

SECTION 7 APPENDIX

7. Product Error Report

Dear Customer,

Although FORCE COMPUTERS has achieved a very high standard of quality in products and documentation, suggestions for improvements are always welcome.

Customer feedback is always appreciated.

Please use the "Product Error Report" form on the next page for your comments and return it to one of our listed offices.

Sincerely,

FORCE COMPUTERS GmbH/Inc.

This page intentionally left blank

PRODUCT ERROR REPORT

HARDWARE/SOFTWARE/SYSTEMS

PRODUCT:	SERIAL NO.:
DATE OF PURCHASE:	ORIGINATOR:
COMPANY:	POINT OF CONTACT:
ADDRESS: _____ _____ _____ _____	TELEPHONE:_____ EXT:_____
PRESENT DATE:	

THIS AREA TO BE COMPLETED BY FORCE COMPUTERS:

DATE:_____
PR#:_____

RESPONSIBLE DEPT.:

___ ENGINEERING
___ MARKETING
___ PRODUCTION

AFFECTED PRODUCT:	**AFFECTED DOCUMENTATION:**
___ HARDWARE ___ SOFTWARE ___ SYSTEM	___ HARDWARE ___ SOFTWARE ___ SYSTEM

ERROR DESCRIPTION:

_____
_____
_____
_____

Please send this product error report to one of our nearest FORCE COMPUTERS offices:

FORCE COMPUTERS Inc.
2001 Logic Drive
San Jose, CA 95124-3468
U.S.A.

FORCE COMPUTERS GmbH
Prof.-Messerschmitt-Str.1
D-85579 Neubiberg/Munich
Germany

FORCE COMPUTERS France S.A.R.L.
Le Volta
17-19 rue Jeanne Braconnier
F-92366 Meudon La Forêt Cédex
France

FORCE COMPUTERS U.K. Ltd.
Alton House Office Park
Gatehouse Way
Aylesbury, Bucks. HP19 3XU
United Kingdom