

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

April, 1991

Mizar
MZ 8130 / MZ 8134
68030-based VMEbus CPU Boards

Hardware Reference Manual

Publication No. 4425241
Third Edition

Mizar
2410 Luna Road, Carrollton, Texas 75006
800-635-0200, 214-277-4600

MZ 8130/MZ 8134 Product Notes

For technical assistance, contact the Mizar Technical Support Center at 1-800-635-0200.

COPYRIGHT

Copyright © 1989, 1990, 1991 Mizar Digital Systems, Inc. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without written permission of Mizar. This document was prepared by Mizar and was printed in the United States of America.

UNPACKING

This board contains components that are susceptible to static discharge and should be handled with appropriate caution.

Upon receipt, inspect the board for missing, broken, or damaged components and for physical damage to the printed circuit board or connectors. This product was shipped in perfect condition. Any damage to the product is the responsibility of the shipping carrier and should be reported to the carrier's agent immediately.

WARRANTY

Mizar warrrants its computer board products to conform to the description on an accepted purchase order and to be free from defects in material and workmanship for twelve (12) months. These warranties are contingent upon proper use in the application for which the item was intended and do not cover equipment modified in any manner (with or without the knowledge of Mizar) or which was subjected to unusual physical or electrical stress. The liability of Mizar hereunder shall be limited to repair or replacement, at manufacturer's discretion, of any defective unit.

Except for the express warranties set forth above, Mizar makes no warranties, express, implied or statutory, as to any products sold, including, but not limited to, their merchantability or fitness for any particular purpose. Any affirmation of fact or promise made by Mizar shall not be deemed to create an express warranty that the products shall conform to the affirmation or promise, and any description or model of the products is for identification or illustrative purposes only and shall not be deemed to create an express warranty.

REVISIONS

The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, Mizar reserves the right to make changes, without prior notice, to any products herein to improve reliability, function, or design. Mizar does not convey any license under its patent rights or the patents of others.

TRADEMARKS

Mizar is a registered trademark and MXbus is a trademark of Mizar Digital Systems, Inc.
OS-9 is a trademark of Microware Systems Corporation.
VxWorks is a trademark of Wind River Systems.

Preface

This manual describes both the MZ 8130 and the MZ 8134 68030-based CPU VMEbus boards: installation, functional description, and firmware requirements. You should be familiar with the VMEbus specification, the MC68030/MC68EC030 CPU, electronics, and computers. This manual is not a tutorial in VMEbus, CPU functions, or protocol.

This manual contains the following sections:

Section 1: User's Guide describes the MZ 8130/MZ 8134 and its configuration and installation.

Section 2: Programming Guide describes the address maps, serial port chip, SmartWatch, CPU interrupts, and registers.

Section 3: Hardware Reference Guide describes the clocks, memory devices, timer, system controller, RS-232-C specifications, connector pin assignments, and the schematics.

Glossary defines words used in this manual.

Index

Manual Conventions

This manual uses the following conventions for notes, fonts, data, and signals.



This symbol identifies important information to remember.



This symbol identifies precautions you should take to prevent damage to equipment or data.



This symbol identifies precautions you should take to prevent injury to yourself or others.

Fonts

This manual uses **boldface** for section headings, defaults in tables, and commands you type in at the keyboard. It uses *italics* for the names of manuals.

Address and Data Prefixes

The prefix attached to an address or data identifies the type of number in the address or data.

%	A binary number
&	A decimal number
$	A hexadecimal number
None	(No notation) A decimal number

Electrical Signal Levels

A dagger (†) by a signal listed in a table indicates its state is "don't care". An asterisk (*) by a signal name indicates the signal is asserted low. Electrical and logical signal levels are defined in the glossary.

Related Publications

VMEbus Specification, Revision C.1, October 1985, available from VITA, 10229 N. Scottsdale Rd., Suite E, Scottsdale, Arizona 85253 USA, (602) 951-8866.

MXbus Specification available from Mizar Technical Support.

MC68030 Enhanced 32-Bit Microprocessor User's Manual, Second Edition MC68030 UM/AD, 1987, available from Motorola, Inc.

SCN68681 Dual Asynchronous Receiver/Transmitter (DUART) Product Specification, December 1986, available from Signetics Corporation, 811 E. Arques Ave., P.O. Box 3409, Sunnyvale, California 94088-3409, (408) 991-2000; or from any of their regional sales offices and distributors.

DS1216E/F SmartWatch Product Data Sheet, 1987, available from Dallas Semiconductor Corporation, 4350 Beltwood Parkway South, Dallas, Texas 75244-3219, (214) 450-0400.

Table of Contents

Section 1: User's Guide

1.1 General Description 1-1
1.2 VMEbus Interface 1-3
1.3 MXbus Interface 1-5
1.4 Unpacking the MZ 8130/MZ 8134 1-6
1.5 Configuring the Jumpers 1-7
1.5.1 K1: MXbus Expansion Header 1-9
1.5.2 K2: CPU ID Number Selection 1-10
1.5.3 K3: EPROM Size Selection 1-11
1.5.4 K4: CPU/DRAM Timing 1-12
1.5.5 K5: MXbus IACKIN/IACKOUT Daisy-Chain Bypass Jumper 1-12
1.5.6 K6: MXbus BIRQ1* and BIRQ2* VMEbus Request Level 1-13
1.5.7 K7, K8: VMEbus Request/Grant Level Selection 1-13
1.5.8 K9: VMEbus System Controller Selections 1-15
1.6 Installing the MZ 8130/MZ 8134 1-16
1.7 Controls and Indicators 1-17

Section 2: Programming Guide

2.1 Configuring the MZ 8130/MZ 8134 2-1
2.1.1 MZ 8130/MZ 8134 Addressing 2-1
2.1.1.1 A32 Address Decoding 2-1
2.1.1.2 A24 Master Side Card Address Decoding 2-2
2.1.2 Setting the Registers 2-3
2.1.2.1 Control Register 2-4
2.1.2.2 Status Register 2-9
2.1.3 MZ 8130/MZ 8134 Initialization 2-10
2.1.4 Configuring the VMEbus Interface 2-11
2.1.4.1 The MZ 8130/MZ 8134 as a VMEbus Master 2-11
2.1.4.2 CPU CAS and CAS2 Instruction VMEbus Incompatibility 2-13
2.1.4.3 The MZ 8130/MZ 8134 as a VMEbus Slave 2-13
2.1.4.4 VMEbus Address Modifier Codes 2-13
2.1.4.5 The MZ 8130/MZ 8134 as VMEbus Requester 2-14
2.1.5 Configuring the MXbus Side Card Interface 2-15
2.1.5.1 MXbus A24 Side Card Slave 2-15
2.1.5.2 MXbus A32 Side Card Slave 2-16
2.1.5.3 MXbus A24 Side Card Master 2-16
2.1.5.4 MXbus A32 Side Card Master 2-17
2.2 Using the SCN68681 DUART Serial Port Chip 2-17
2.2.1 Initializing the SCN68681 DUART 2-18
2.2.2 Transmitting and Receiving Characters 2-19
2.2.3 Using the Timer 2-19
2.2.4 Using Interrupts 2-20
2.2.5 Programming the SCN68681 2-20

2.3 Using the SmartWatch Time-of-Day Clock 2-23
2.3.1 Using the SmartWatch with 512K EPROMs 2-30
2.4 CPU Interrupts 2-30
2.4.1 ACFAIL*, and ABORT Autovectored Interrupts 2-30
2.4.2 Side Card S1IRQ1* and S1IRQ2* Autovectored Interrupts 2-31
2.4.3 VMEbus Mailbox Autovectored Interrupt 2-31
2.4.4 DUART Serial Port/Timer Vectored Interrupts 2-31
2.4.5 VMEbus IRQ 7 to IRQ 1 Vectored Interrupts 2-32
2.4.6 Side Card VMEbus BIRQ1* and BIRQ2* Interrupts 2-32

Section 3: Hardware Reference Guide

3.1 Introduction 3-1
3.2 CPU Central Processing Unit 3-2
3.3 Clocks 3-3
3.4 Memory Devices 3-3
3.4.1 EPROM 3-3
3.4.2 Dual-Access DRAM 3-3
3.5 DUART Serial Ports/Timer 3-4
3.5.1 RS-232-C Interface Specifications 3-4
3.5.2 Serial Port RS-232-C DCE Specifications 3-5
3.6 VMEbus Connector (P1) 3-6
3.7 MXbus Connector (S1) 3-8
3.8 System Controller 3-13
3.9 MZ 8130/MZ 8134 Specifications 3-15
3.10 Schematics 3-15

List of Figures

Figure 1-1. MZ 8130/MZ 8134 CPU Board 1-2
Figure 1-2. MZ 8130/MZ 8134 with MXbus Side Card 1-5
Figure 1-3. MZ 8130/MZ 8134 Jumper Block Location 1-8
Figure 1-4. K1: MXbus Expansion Header Signals 1-9
Figure 1-5. K2: CPU ID Number Selection 1-10
Figure 1-6. K3: EPROM Size Selection 1-11
Figure 1-7. K5: MXbus IACKIN/IACKOUT Daisy-Chain Bypass Jumper Block 1-12
Figure 1-8. K6: MXbus BIRQ1 & BIRQ2 VMEbus Interrupt Request Selection 1-13
Figure 1-9. K7 and K8: VMEbus Request/Grant Level Selection 1-14
Figure 1-10. K9: VMEbus System Controller and Time-Out Selection 1-15
Figure 1-11. Front Panel Controls and Indicators 1-17
Figure 3-1 MZ 8130/MZ 8134 Block Diagram 3-1

Schematics CPU Module ..... 1 of 10
VME Data Buffers ..... 2 of 10
VME Address and Strobes ..... 3 of 10
VMEbus Syscon, Bus Requester ..... 4 of 10
CPU Bus Arbitration, DRAM Control ..... 5 of 10
DRAM Memory, buffer, BERR, VME DRAM PAL ..... 6 of 10
CPU, EPROM, and Address Decoding ..... 7 of 10
Serial port, IACK Dvr, DSACK, and MXbus Control ..... 8 of 10
Interrupts, Status and Control Registers ..... 9 of 10
MXbus Interface ..... 10 of 10
P2 VMEbus Interface Example Schematics ..... 1 of 1

List of Tables

Table 1-1. Differences Between the MZ 8130 and the MZ 8134 1-3
Table 1-2. Jumper Block Defaults 1-7
Table 1-3. EPROM Access Times 1-11
Table 1-4. EPROM Size and Jumper Settings 1-11
Table 1-5. Request/Grant Level Selection and Jumpers 1-14
Table 2-1. MZ 8130 A32 Address Map 2-1
Table 2-2. MZ 8134 A32 Address Map 2-2
Table 2-3. MZ 8130 A24 Address Map 2-2
Table 2-4. MZ 8134 A24 Address Map 2-3
Table 2-5. Control Register 2-4
Table 2-6. Control Register Defaults 2-5
Table 2-7. Mailbox Interrupt VMEbus address 2-5
Table 2-8. Dual Access DRAM VMEbus base address 2-5
Table 2-9. System Controller Arbitration 2-6
Table 2-10. VMEbus Release Mechanisms 2-6
Table 2-11. Cache and Memory Selection 2-7
Table 2-12. Data Path Size Selection 2-8
Table 2-13. DRAM Address Space Selection 2-8
Table 2-14. Interrupt Handler Levels 2-9
Table 2-15. Status Register 2-9
Table 2-16. MZ 8130/MZ 8134 Defaults 2-10
Table 2-17. VMEbus Access for CPU and 32-bit Side Card Masters 2-12
Table 2-18. VMEbus Access for 24-bit Side Card Masters 2-12
Table 2-19. VME Address Modifier Codes 2-14
Table 2-20. A32 CPU Addresses Access Same A24 Address in A24 MXbus Slave Side Card 2-15
Table 2-21. MZ 8130 A32 Address Decoding 2-16
Table 2-22. MZ 8134 A32 Address Decoding 2-16
Table 2-23. SP0, SP1 - DCE Serial Port RS-232-C Pin Assignments 2-18
Table 2-24. SCN68681 Register Address Map 2-18
Table 2-25. CPU Interrupt Priorities 2-30
Table 2-26. Default STATUS/ID Locations for Side Card Generated VMEbus Interrupts 2-33
Table 3-1. Signal States 3-4
Table 3-2. DCE RS232-C Pin Assignment 3-5
Table 3-3. VMEbus P1 Pin Assignments 3-6
Table 3-4. Side Card VMEbus P2 Pin Assignments 3-7

Table 3-5. S1 Pin Assignments 3-8
Table 3-6. A24:D16 MXbus Loading/Driving Specifications 3-9
Table 3-7. A24:D16 MXbus Signal Description 3-10
Table 3-8. A32:D32 MXbus Loading/Driving Specifications 3-11
Table 3-9. A32:D32 MXbus Signal Description 3-11
Table 3-10. MXbus Expansion Header Loading/Driving Specifications 3-12
Table 3-11. MXbus Expansion Header Signals 3-12

Section 1: User's Guide

1.1 General Description

This section gives a brief overview of the MZ 8130/MZ 8134 in the following sections:

1.1 General Description
1.2 VMEbus Interface Capability
1.3 MXbus Interface Capability
1.4 Unpacking the MZ 8130/MZ 8134
1.5 Setting the Jumpers
1.6 Installing the MZ 8130/MZ 8134
1.7 Controls and Indicators

The MZ 8130/MZ 8134 (shown in Figure 1-1) is a high-performance single-height (3U x 160mm) VMEbus central processing unit (CPU) board incorporating:

- An MC68030 microprocessor with a memory management unit or an MC68EC030 microprocessor without a memory management unit
- One Mbyte of dual-access DRAM (MZ 8130) or four Mbytes of dual-access DRAM (MZ 8134)
- Up to one Mbyte of EPROM
- Two serial ports
- A time-of-day clock
- A VMEbus interface
- A jumper-selectable VMEbus system controller
- A full 16/32-bit MXbus™ master and slave side card interface
- VMEbus interrupt handler
- A mailbox interrupt
- A 16-bit timer interrupt
- Up to four Gbytes addressing range with optional side card

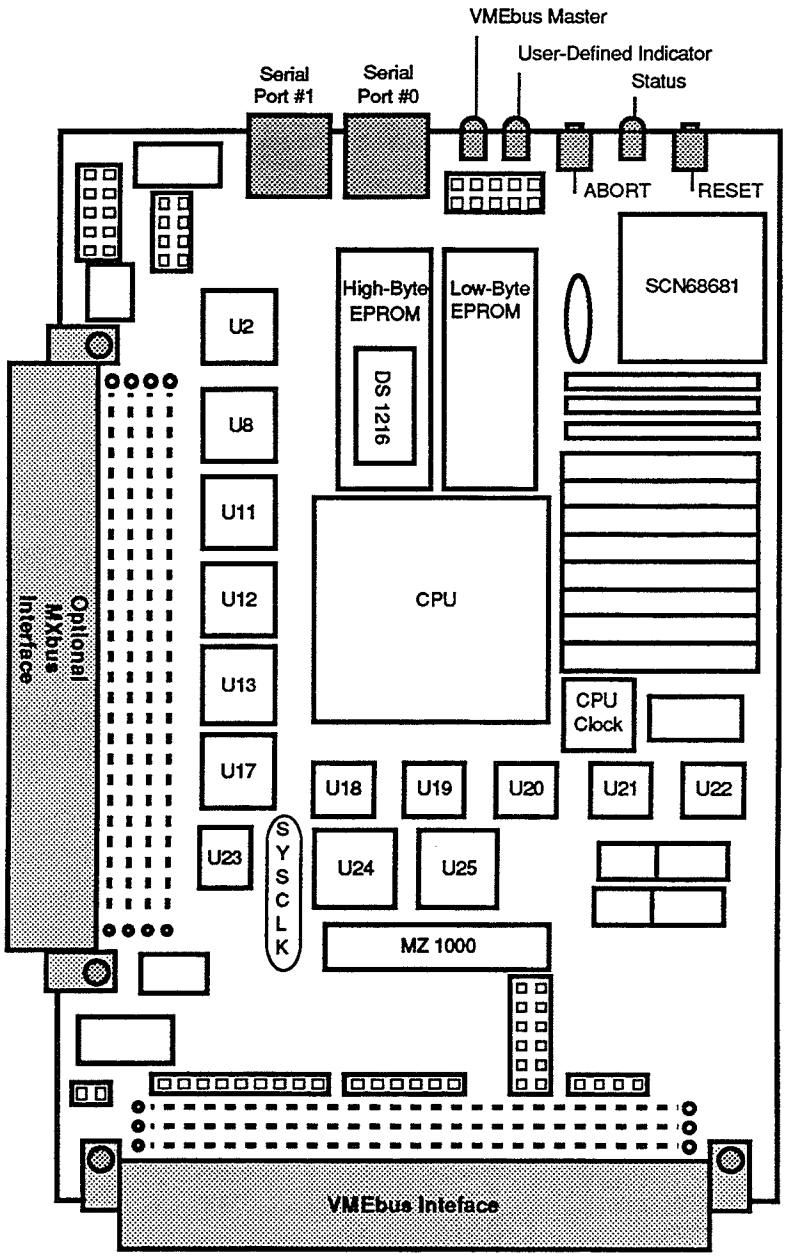


Figure 1-1. MZ 8130 CPU Board (MZ 8134 differs slightly)

Table 1-1 summarizes the differences between the MZ 8130 and the MZ 8134.

Table 1-1. Differences Between the MZ 8130 and the MZ 8134

Feature	MZ 8130	MZ 8134
CPU	MC68030 or MC68EC030	MC68030 or MC68EC030
CPU Clock	20 or 25 MHz	20 or 25 MHz
SmartWatch option	Yes	Yes
DRAM	1 Mbyte	4 Mbytes
DRAM address in Control Register	Bits 23 to 20	Bits 23, 22
BIRQ1 and BIRQ 2 STATUS/ID address	See Table 2-25	See Table 2-25
A32 address map	See Table 2-1	See Table 2-2
A24 address map	See Table 2-3	See Table 2-4
K4 Jumper	Yes	No
CPU DRAM timing	80 ns	80 ns
MXbus interface	Optional	Optional
DRAM address range	$0 - $FFFFF	$0 - $3FFFFF
MXbus address schemes	See Table 2-3	See Table 2-4
S1 connector	Optional	Optional
S1 pin 27, row D	Reserved	"Cacheable" signal

1.2 VMEbus Interface

The MZ 8130 and MZ 8134 fully conform to the *VMEbus Specification, Revision C.1*, including address pipelining (slave only), read-modify-write cycles, address-only cycles, early bus release, and VMEbus system controller with an IACK daisy-chain driver. The MZ 8130/MZ 8134 has VMEbus A24 addressing and VMEbus D16 data transfer capabilities. With an MXbus side card providing buffers for the VMEbus P2 connector address and data signals, the MZ 8130/MZ 8134 can have VMEbus A32 addressing and D32 data transfer capabilities. The VMEbus base address is selected by the control register.

Functional modules on the MZ 8130/MZ 8134 provide the following VMEbus features:

1) Mailbox interrupt: An auto-vectored interrupt is generated when the mailbox address is detected on the VMEbus. The mailbox interrupt address is selected by the control register and can be located on any 256 byte boundary in the VMEbus Short I/O address space (A16).

2) Master/slave modules:

   - The bus master functional module initiates and directs VME data transfer bus cycles to slave modules. The MZ 8130/MZ 8134 is a VMEbus A16:D16 and A24:D16 master.
   - The slave functional module detects and responds to VME data transfer bus cycles directed to it, allowing other VMEbus masters to access the MZ 8130/MZ 8134 DRAM. The DRAM is accessed as a VMEbus A24:D16 slave.

3) VMEbus System Controller:

- The MZ 1000 System Controller gate array provides all necessary VMEbus system controller functions, including four level bus arbitration, 16 MHz system clock, and VMEbus time-out signals.
- A four-level arbiter accepts requester signals BR0* to BR3* and grants the bus to one requester at a time. The arbitration method is specified by control register bit 18.
- An IACK daisy-chain driver generates an IACKIN* when an interrupt handler acknowledges a request.

4) Bus request module:

- A selectable VMEbus requester arbitrates for master status and accepts the appropriate bus grant signal (BGIN0 to BGIN3) acknowledging its request. The bus grant in is passed through to the bus grant out (BGOUT0 to BGOUT3) signal if the bus is being granted on a level other than its own or if it has no pending requests.
- The MZ 8130/MZ 8134's VMEbus requester circuit provides four VMEbus release mechanisms and early release of the VMEbus to enable bus arbitration to occur during the last MZ 8130/MZ 8134 bus cycle, increasing the overall VMEbus transfer rate. The bus release method is selected by control register bits 17-15.

5) Interrupt handler module:

- The interrupt handler can be configured to respond to up to all seven VMEbus interrupt levels.

6) Resets:

- Responds to SYSRESET* from the system controller.
- Generates the VMEbus SYSRESET signal in system controller mode when powered up or when the reset switch is activated.

1.3 MXbus Interface

The MXbus is an option on Mizar's family of single-height (3U) 680x0 CPU boards. The MXbus enables the CPU board to be expanded with an optional side card, such as an MZ 8800 series board as illustrated in Figure 1-2.

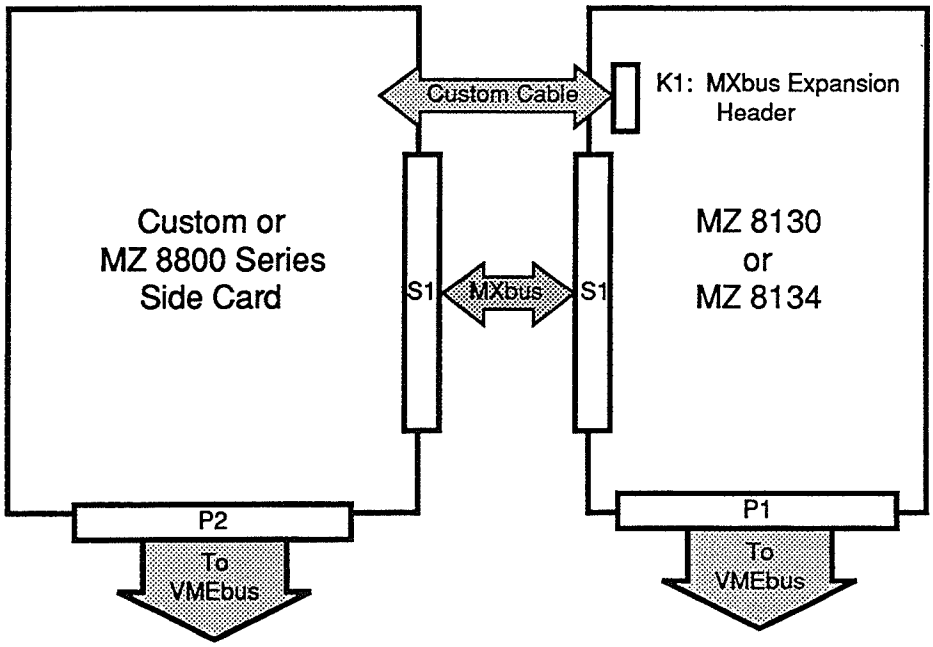


Figure 1-2. MZ 8130/MZ 8134 with MXbus Side Card

A full array of address lines, data lines, and interrupt lines, microprocessor function codes, bus request, and bus grant signals can be passed through the MXbus connector, S1.

A side card can generate two types of interrupts:

- Two autovectored, on-board local interrupts can be generated via signals S1IRQ1 and S1IRQ2
- Two VMEbus interrupts can be generated via the signals BIRQ1 and BIRQ2.

MXbus side cards can be used to add resources, such as proprietary serial or parallel ports, floppy disk controllers, or memory. An expanded board (the MZ 8130/MZ 8134 plus the side card) can use the side card's VMEbus interface to connect to the VMEbus upper address and data lines, as well as user-defined pins on the P2 connector.

1.4 Unpacking the MZ 8130/MZ 8134

Mizar supplies complete systems for end users as well as hardware and software for OEM and VAR buyers. For a custom system, there may be no standard package or prepackaged system to check your shipment against so you must check each item on the packing list against your purchase order and the actual, physical components. Mizar tests all systems, boards, and peripherals before shipping.

Inspect the shipping materials for major damage. Mishandling during shipping can affect the appearance or operation of equipment.

Carefully unpack and visually inspect all boards, chassis, and cables for any indication of physical damage before proceeding. Report any damage to the carrier used to ship the unit. Leave the system down and leave power off until all connections are complete.

Check the MZ 8130/MZ 8134 against your shipping list and purchase order to make sure the board has the options you requested. Inspect the male pins on all connectors for shipping damage. Pins should be straight and evenly spaced.

9 Do not apply power to any board that has been damaged. Applying power to a damaged board can cause additional damage as well as damage other boards in your VMEbus system.

Before using the MZ 8130/MZ 8134 board:

1) Configure the jumpers (Section 1.5, "Configuring the Jumpers")
2) Install the MZ 8130/MZ 8134 in your system (Section 1.7, "Installing the MZ 8130/MZ 8134")
3) Configure the MZ 8130/MZ 8134 software (Section 2, "Programming Guide")
4) Boot the software you plan to run, such as VxWorks™ or OS-9™. Refer to the appropriate software manual.

1.5 Configuring the Jumpers

You may have to reconfigure some jumper blocks for your specific system. Table 1-2 shows the default jumper configuration.

Table 1-2. Jumper Block Defaults

Jumper Block	Function	Default Setting
K1	MXbus Expansion Header	NO JUMPERS
	You should not attach anything to the MXbus Header unless it is appropriate for the side card you installed.	
K2	CPU ID	$F
K3	EPROM size selection	64K/128K EPROMs
K4	CPU/DRAM Timing	Factory Set
K5	MXbus IACKIN/IACKOUT	Bypassed (IACK daisy-chain bypass jumper installed)
K6	MXbus BIRQ1 and BIRQ2 Interrupts	No MXbus interrupts
K7, K8	K7: VMEbus request level K8: VMEbus grant level	Bus request level 0 Bus grant level 0
	K7 and K8 must select the same VMEbus request/grant level.	
K9	VMEbus system controller and VMEbus BERR	Controller enabled, BERR enabled
	The MZ 8130/MZ 8134 must be in VMEbus slot one if the system controller is enabled.	

Figure 1-3 shows the locations of the jumper blocks. Pin 1 on each jumper block is in the upper left-hand corner when the board is oriented component side up with the P1 connector toward you.

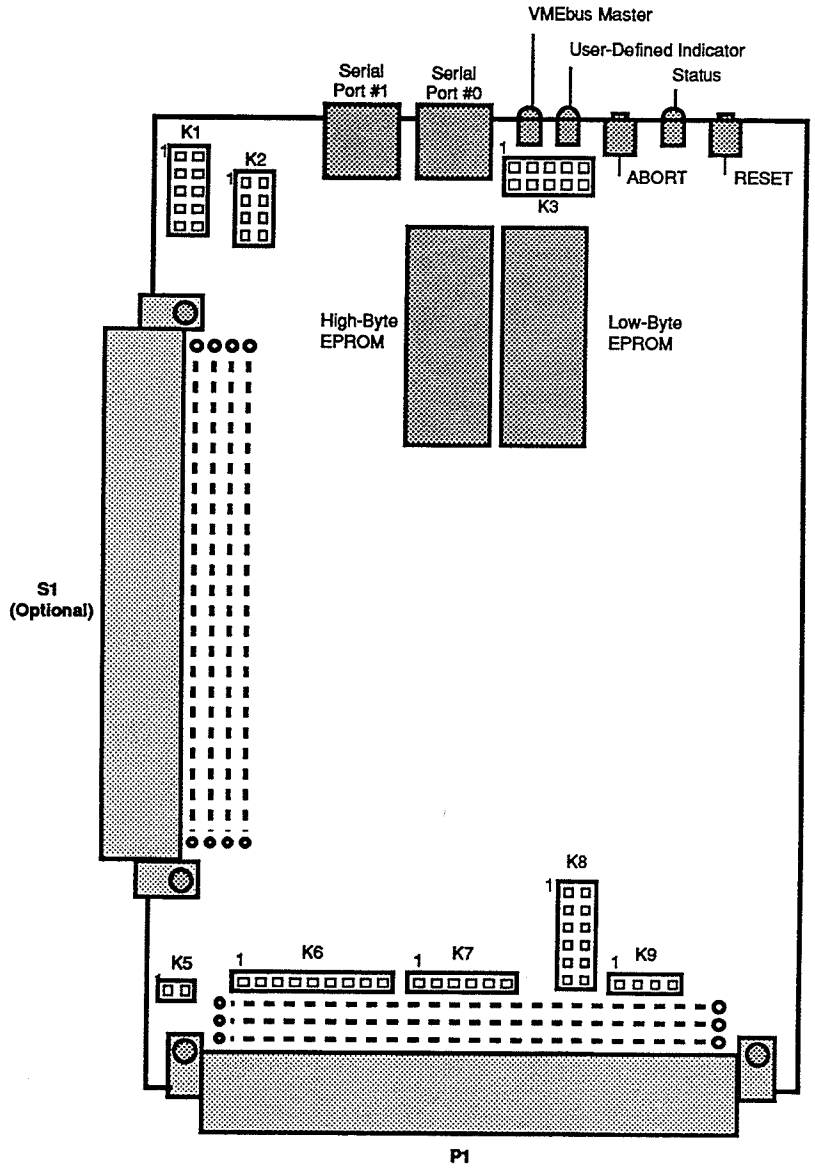


Figure 1-3. MZ 8130 Jumper Block Location (MZ 8134 differs slightly)

1.5.1 K1: MXbus Expansion Header

Block K1, the MXbus expansion header, makes the non-buffered CPU signals in Figure 1-4 available to side card designers. It enables MXbus side cards to expand the MZ 8130/MZ 8134's DRAM and/or enables additional zero-wait state memory to be available to the CPU.

For an MXbus side card, use a wire wrap tool for point-to-point wiring or a flat ribbon cable.

Refer to Section 1.3, "MXbus Interface," or to the *MXbus Specification* for more information about the MXbus.

9 You should not attach anything to the MXbus Header unless it is appropriate for the side card you installed.

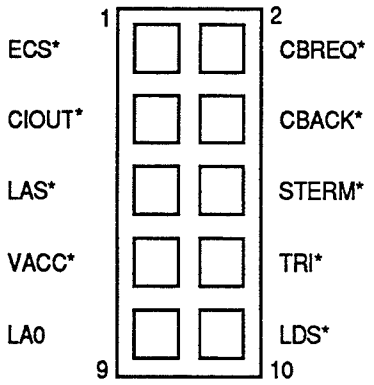
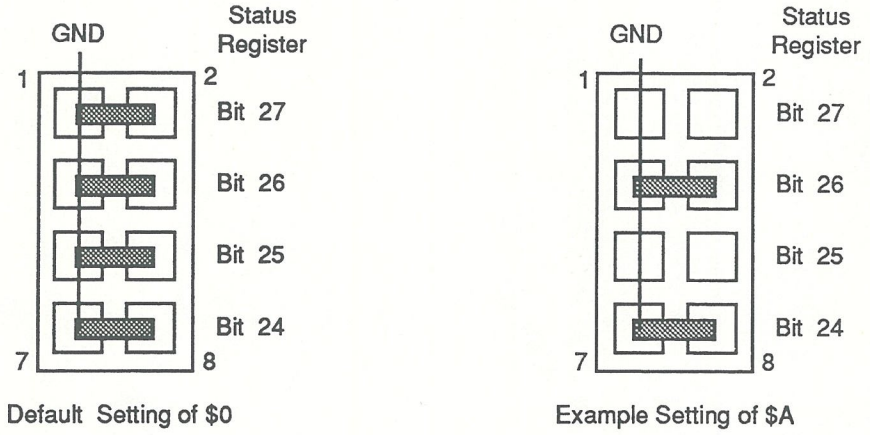


Figure 1-4. K1: MXbus Expansion Header Signals (* active low)

1.5.2 K2: CPU ID Number Selection

Jumper block K2 selects the CPU ID number. This jumper block is software-readable via status register bits 27 to 24 and can be used to identify specific CPUs in a multiple CPU system. This jumper block can also be used for any user-defined purpose. As shown in Figure 1-5, a jumper between the selected pins clears the corresponding status register bit. Unjumpered pins set the corresponding status bit.



BIN HEX
1 0 1 0 A
1 1 1 1 F

Figure 1-5. K2: CPU ID Number Selection

1.5.3 K3: EPROM Size Selection

Jumper block K3 configures the EPROM sockets for 64K, 128K, 256K, 512K, 1M, 2M or 4M EPROMs. EPROM size is normally specified in total number of bits, not bytes. The MZ 8130/MZ 8134 uses two EPROMs, organized for 16-bit accesses from the CPU. Section 2.2, "Configuring the Registers," describes the addressing. The MZ 8130/MZ 8134 requires a high-byte EPROM (HB, D31 to D24) and a low-byte EPROM (LB, D23 to D16) as shown in Figure 1-1. Both EPROMs must be the same size and the same speed. Table 1-3 indicates the EPROM access time. Table 1-4 shows the total number of bytes for each EPROM size and the jumpers for K3. Figure 1-6 shows the jumper settings for both 256K (the default) and 2M EPROM

9 Only EPROMs with access times equal to or less than those shown below should be used to insure proper operation.

AT27C010

1MB (128K x 8)

Table 1-3. EPROM Access Times

CPU Speed	EPROM Access Time
20 MHz	200 ns
25 MHz	150 ns

Table 1-4. EPROM Sizes and Jumper Settings

Total Bytes	Organization	EPROM	K3 Jumpers
16 KBytes	**8K x 8 (x 2)**	**64K**	**4-5**
32 KBytes	**16K x 8 (x 2)**	**128K**	**4-5**
64 KBytes	32K x 8 (x 2)	256K	1-6, 4-5
128 KBytes	64K x 8 (x 2)	512K	1-6, 2-7, 4-5
256 KBytes	128K x 8 (x 2)	1M	* 1-6, 2-7, 4-5 *
512 KBytes	256K x 8 (x 2)	2M	1-6, 2-7, 4-9
1 MByte	512K x 8 (x 2)	4M	1-6, 2-7, 3-8, 4-9

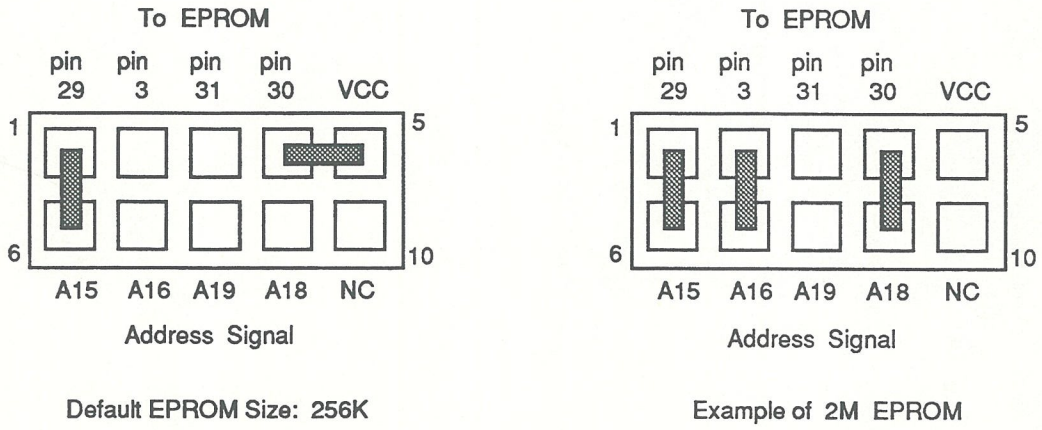


Figure 1-6 K3: EPROM Size Selection

1.5.4 K4: CPU/DRAM Timing

Jumper block K4 on the MZ 8130 is factory set and should not be changed.

This jumper block is not present on the MZ 8134.

1.5.5 K5: MXbus IACKIN/IACKOUT Daisy-Chain Bypass Jumper

Jumper block K5 accommodates a side card which generates VMEbus interrupts via the MXbus signals BIRQ1 and/or BIRQ2. (The side card may also generate local interrupts via S1IRQ1 or S1IRQ2). If a side card generates BIRQ*x*, K5 must be left unjumpered to enable the IACKIN and IACKOUT signals to pass through to the MXbus connector. If no side card is present or if the side card does not generate VMEbus interrupts, K5 **MUST** be jumpered to propagate the signal down the IACK daisy chain. Figure 1-7 shows the K5 jumper block with and without a jumper. Refer to Section 1.5.6, "BIRQ1* and BIRQ2* MXbus Side Card VMEbus Interrupts," for details about side card generated VMEbus interrupts.

The MZ 8130/MZ 8134 by itself is not a VMEbus interrupter.



Figure 1-7. K5: MXbus IACKIN/IACKOUT Daisy-Chain Bypass Jumper Block

1.5.6 K6: MXbus BIRQ1* and BIRQ2* VMEbus Request Level

Jumper block K6 selects the MXbus BIRQ1* and/or BIRQ2* side card VMEbus interrupt request level(s). You can leave this jumper block unconnected if the MXbus side card does not assert a BIRQ*x*. Figure 1-8 shows the pin assignments of jumper block K6 and an example configuration. Use a wire wrap tool for point-to-point wiring of a jumper from the BIRQ*x* pin to the VMEbus IRQ*x* pin.

If the side card will be asserting either BIRQ1 or B1RQ2 VMEbus interrupt request, remove the K5 jumper.

3 **Do NOT** configure the MZ 8130/MZ 8134 VMEbus interrupt handler, via the control register, to respond to either of the side card VMEbus interrupt levels selected by K6.

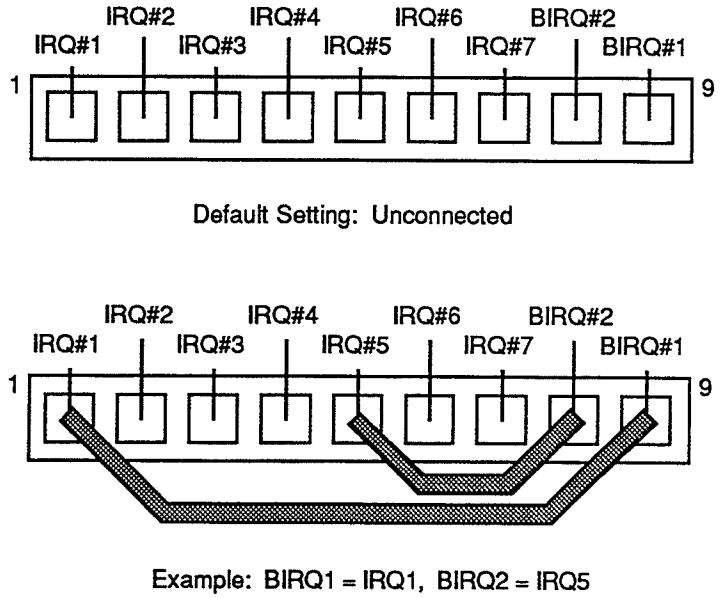


Figure 1-8. K6: MXbus BIRQ1 & BIRQ2 VMEbus Interrupt Request Selection

1.5.7 K7, K8: VMEbus Request/Grant Level Selection

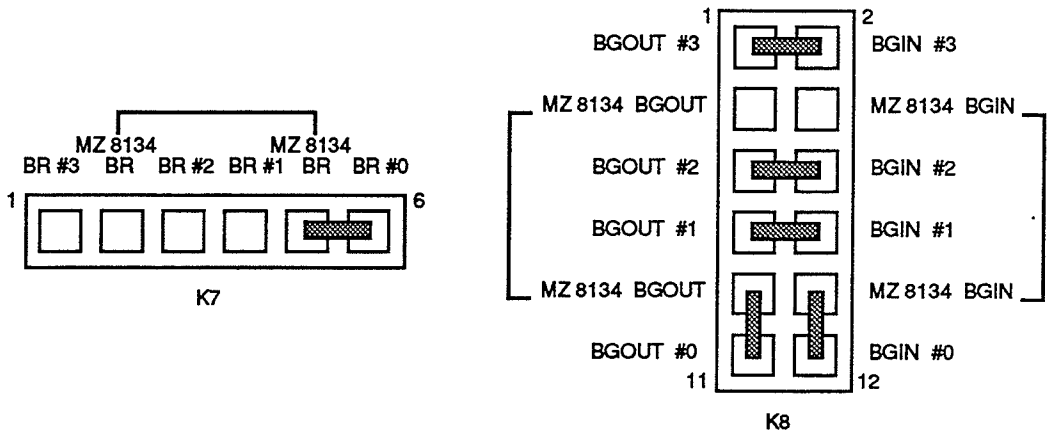
Jumper blocks K7 and K8 select one of four VMEbus request and grant levels. Each VMEbus master must reside on only one level, but any number of bus masters can reside on a level since these signals are daisy chained down the VMEbus backplane. K7 and K8 must be configured for the same level. Unused levels are passed through. The MZ 8130/MZ 8134 signals appear in two positions on the jumper block to enable use of a jumper rather than a wire wrap.

The *VMEbus Specification, Revision C.1* and Section 2.2.1, "Control Register," describe bus level priorities and arbitration methods.

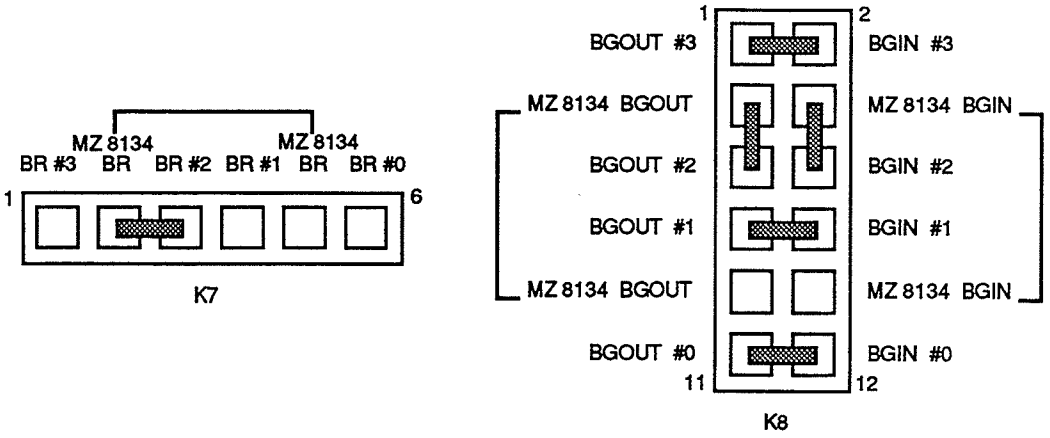
Table 1-5 shows the request/grant level selections and pin connections with the default setting in boldface. Figure 1-9 shows the jumper settings for K7 and K8.

Table 1-5. Request/Grant Level Selections and Jumpers

VMEbus Level	Priority Level	K7 Jumpers	K8 Jumpers
3	1st	1-2	1-3, 2-4, 5-6, 7-8, 11-12
2	2nd	2-3	1-2, 3-5, 4-6, 7-8, 11-12
1	3rd	4-5	1-2, 5-6, 7-9, 8-10, 11-12
0	**4th**	**5-6**	**1-2, 5-6, 7-8, 9-11, 10-12**



Default Setting: VMEbus Request and Grant on Level 0



Example Setting: VMEbus Request and Grant on Level 2

Figure 1-9. K7 and K8: VMEbus Request/Grant Level Selection

1.5.8 K9: VMEbus System Controller Selections

Jumper block K9 enables the MZ 1000 system controller chip and the MZ 1000's VMEbus time-out circuit (BERR). Every VMEbus system must have only one system controller enabled, and it must reside in slot 1, the leftmost slot on the VMEbus backplane. The VMEbus system controller is enabled by placing a jumper between pins 1 and 2 of the K9 jumper block.

When the MZ 8130/MZ 8134 is the system controller, pressing RESET asserts both a local and VMEbus system-wide reset.

When K9 enables the bus time-out circuit, the MZ 1000 system controller monitors the two VMEbus data strobes, DS0 and DS1, and generates a VMEbus BERR signal if either data strobe stays asserted beyond 64 to 128 µs from the assertion of the data strobe. During debugging, you can disable bus time-out to enable the VMEbus to "hang" so that you can examine the address and control signals on the VMEbus. To disable bus time-out, remove the jumper between pins 3 and 4. To enable bus time-out, place a jumper between these pins as shown in Figure 1-10. If the system controller function is disabled, the BERR jumper has no effect.

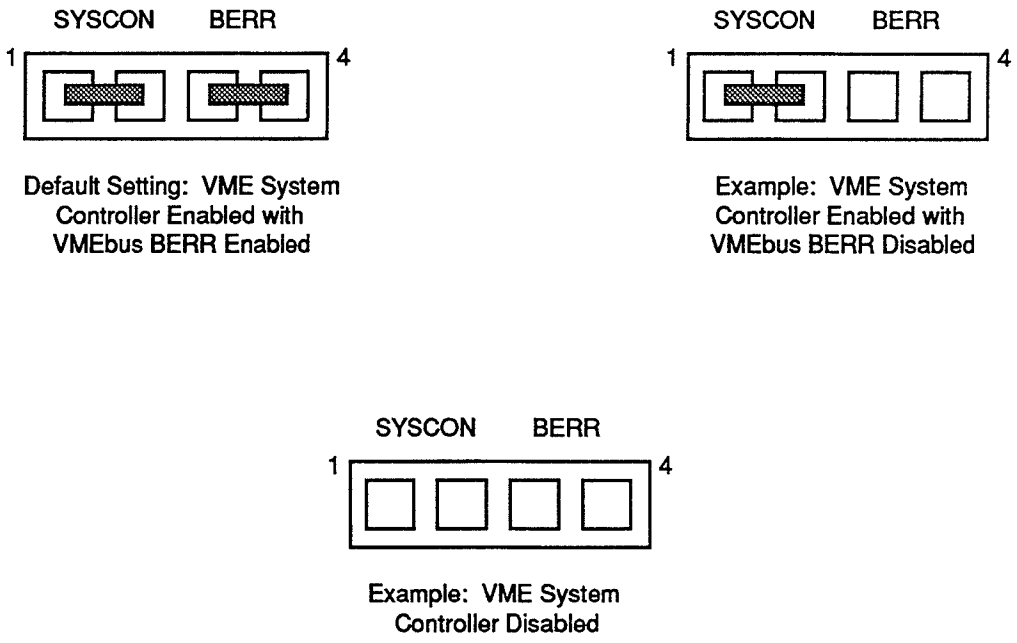


Figure 1-10. K9: VMEbus System Controller and BERR Selection

1.6 Installing the MZ 8130/MZ 8134

If the MZ 8130/MZ 8134 will be the VMEbus system controller, it must be installed in slot one on the VMEbus backplane. Make sure that all BGIN/BGOUT and IACKIN/IACKOUT jumpers on the backplane have been removed for the MZ 8130/MZ 8134's VMEbus card slot.

9 Make certain the power supply is off before inserting the MZ 8130/MZ 8134 into theVMEbus backplane.

Section 1.5, "Configuring the Jumpers," describes configuring the MZ 8130/MZ 8134 board to your particular VMEbus system. Verify that all on-board jumpers are correctly placed for your application. Verify that the serial port RJ11 connector cables are correct for your terminal type. RJ11 to DB25 cable adapters are available from Mizar.

Before inserting the MZ 8130/MZ 8134 into the VMEbus backplane, carefully check all power pins on the backplane using a DC voltmeter. All voltages must meet the following specifications:

+5VDC: +4.875 to +5.25VDC
+12VDC: +11.64 to +12.25VDC
-12VDC: -11.64 to -12.25VDC

Refer to Section 3.7, "VMEbus Connector," for the location of the VMEbus P1 and P2 connector power pins to be checked.

Insert the board firmly into the VMEbus slot. Secure the board into the card cage with the two captive screws at the top and bottom of the front panel if your front panel has them. After installing the board, apply power and carefully check the voltage levels on the VMEbus backplane again.

1.7 Controls and Indicators

Figure 1-11 shows the location of the controls and indicators (LEDs) defined below.

RESET: The RESET switch resets the MZ 8130/MZ 8134 and the CPU. When the board is the VMEbus system controller, the RESET switch also drives SYSRESET on the VMEbus, resetting all VMEbus boards in the system.

9 Pressing RESET aborts any processing in progress. The processing cannot be recovered.

STATUS (Red) The CPU halt signal controls this indicator. The LED glows brightly when the CPU is halted, and it glows dimly during normal use.

ABORT: The ABORT switch sends an autovectored nonmaskable level 7 interrupt to the CPU.

USER IND: (Yellow) The USER-defined INDicator glows when control register bit 17 is cleared.

BUS MSTR: (Green) The VMEbus master LED glows when the MZ 8130/MZ 8134 is the VMEbus master.

PORT 0: The RJ11 connector for serial port 0.

PORT 1: The RJ11 connector for serial port 1.

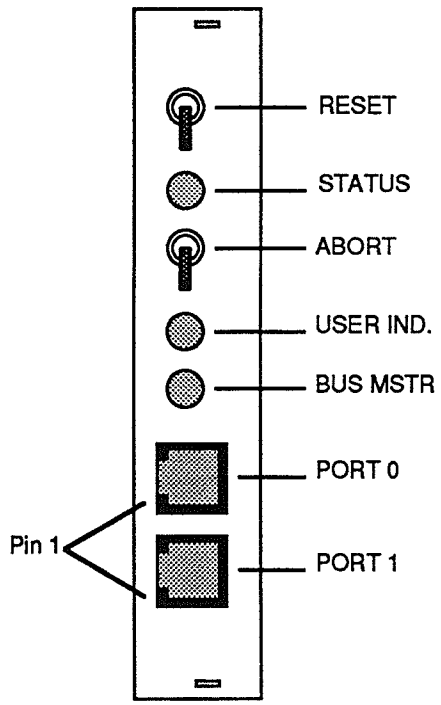


Figure 1-11. Front Panel Controls and Indicators

Section 2: Programming Guide

2.1 Configuring the MZ 8130/MZ 8134

This section describes the software requirements for the MZ 8130/MZ 8134:

2.1 Configuring the MZ 8130/MZ 8134
2.2 Using the SCN68681 DUART Serial Port Chip
2.3 Using the SmartWatch Time-of-Day Clock
2.4 Interrupting the CPU

2.1.1 MZ 8130/MZ 8134 Addressing

2.1.1.1 A32 Address Decoding

3 D16 MXbus masters cannot access the 32-bit control register.

The active low COMPAT32* MXbus signal must be asserted to enable MXbus A32 address decoding for side card masters. The 16- or 32-bit data path size of the VMEbus A16, A24, A32, and MXbus address spaces are selected by the control register. The SCN68681 DUART uses an 8-bit data path.

Table 2-1 shows the A32 address map used by the MZ 8130 CPU and MXbus A32 side card masters.

Table 2-1. MZ 8130 A32 Address Map

Address (Hex)			Port Size	Device
$00000000	-	$000FFFFF	32	Local DRAM
$00100000	-	$00FFFFFF	16/32	VMEbus Standard A24 Address Space
$01000000	-	$F7FFFFFF	16/32	VMEbus Extended A32 Address Space
$F8000000	-	$FEDFFFFF	16/32	MXbus Side Card Address Space
$FEE00000	-	$FEEFFFFF	16	ROM/EPROM, SmartWatch
$FEF00000	-	$FEFBFFFF	NA	Reserved
$FEFC0000	-	$FEFCFFFF	16	Status Register
$FEFD0000	-	$FEFDFFFF	32	Control Register
$FEFE0000	-	$FEFEFFFF	8	SCN68681 DUART Serial Port / Timer
$FEFF0000	-	$FEFFFFFF	NA	Reserved
$FF000000	-	$FFFEFFFF	16/32	MXbus Side Card Address Space
$FFFF0000	-	$FFFFFFFF	16/32	VMEbus Short A16 Address Space

Table 2-2 shows the A32 address map used by the MZ 8134 CPU and MXbus A32 side card masters.

Table 2-2. MZ 8134 A32 Address Map

Address (Hex)	Port Size	Device
$00000000 - $003FFFFF	32	Local DRAM
$00400000 - $00FFFFFF	16/32	VMEbus Standard A24 Address Space
$01000000 - $F7FFFFFF	16/32	VMEbus Extended A32 Address Space
$F8000000 - $FEDFFFFF	16/32	MXbus Side Card Address Space
$FEE00000 - $FEEFFFFF	16	ROM/EPROM, SmartWatch
$FEF00000 - $FEFBFFFF	NA	Reserved
$FEFC0000 - $FEFCFFFF	16	Status Register
$FEFD0000 - $FEFDFFFF	32	Control Register
$FEFE0000 - $FEFEFFFF	8	SCN68681 DUART Serial Port/Timer
$FEFF0000 - $FEFFFFFF	NA	Reserved
$FF000000 - $FFFEFFFF	16/32	MXbus Side Card Address Space
$FFFF0000 - $FFFFFFFF	16/32	VMEbus Short A16 Address Space

2.1.1.2 A24 Side Card Master Address Decoding

The MZ 8130/MZ 8134 decodes an address asserted by an MXbus A24 side card master when the MXbus COMPAT32* signal is not asserted by the side card, as shown by the address maps in Tables 2-3 and 2-4. Note that the MZ 8130/MZ 8134 decodes on-board devices on larger address boundaries than a side card does. Refer to Section 2.1.5, "MXbus Side Card Interface," for more information. Table 2-3 shows the A24 address map used by MZ 8130 MXbus A24 side card masters.

Table 2-3. MZ 8130 A24 Address Map

Address Range (Hex)	Device
$000000 - $0FFFFF	Local DRAM
$100000 - $1FFFFF	MXbus Side Card Address Space (Open)
$200000 - $BFFFFF	VMEbus Standard A24 Address Space
$C00000 - $DFFFFF	MXbus Side Card Address Space (Open)
$E00000 - $EFFFFF	ROM/EPROM, SmartWatch
$F00000 - $FBFFFF	Reserved
$FC0000 - $FCFFFF	Status Register
$FD0000 - $FDFFFF	Control Register
$FE0000 - $FEFFFF	SCN68681 DUART Serial Port / Timer
$FF0000 - $FFFFFF	VMEbus Short A16 Address Space

Table 2-4 shows the A24 address map used by MZ 8134 MXbus A24 side card masters.

Table 2-4. MZ 8134 A24 Address Map

Address Range (Hex)	Device
$000000 - $3FFFFF	Local DRAM
$400000 - $BFFFFF	VMEbus Standard A24 Address Space
$C00000 - $DFFFFF	MXbus Side Card Address Space (Open)
$E00000 - $EFFFFF	ROM/EPROM, SmartWatch
$F00000 - $FBFFFF	Reserved
$FC0000 - $FCFFFF	Status Register
$FD0000 - $FDFFFF	Control Register
$FE0000 - $FEFFFF	SCN68681 DUART Serial Port/Timer
$FF0000 - $FFFFFF	VMEbus Short A16 Address Space

2.1.2 Setting the Registers

The control and status registers allow software access to status information and enable software control of user-configurable functions.

2.1.2.1 Control Register

3 A master side card must have a 32-bit data bus to access the control register.

The control register is a 32-bit, write-only register allowing software control of the functions shown in Table 2-5. It resides at the A32 address $FEFD0000 or A24 address $FD0000 and can be altered only by long word write accesses.

Table 2-5. Control Register

Bit	Description	Bit	Description
31	Mailbox interrupt location	14	Cache memory, CACHERAM
30	Mailbox interrupt location	13	Cache instructions, CACHDIS
29	Mailbox interrupt location	12	Memory management unit MMUDIS
28	Mailbox interrupt location	11	Data path size, D32SIZ
27	Mailbox interrupt location	10	Data path size, D32DIS
26	Mailbox interrupt location	9	DRAM address space in A24 address space, A24DIS
25	Mailbox interrupt location		
24	Mailbox interrupt location	8	DRAM address space in A32 address space, A32DIS
23	DRAM address, MZ 8130/MZ 8134		
22	DRAM address, MZ 8130/MZ 8134	7	VMEbus interrupt level 7, IRQ7
21	DRAM address, MZ 8130 only	6	VMEbus interrupt level 6, IRQ6
20	DRAM address, MZ 8130 only	5	VMEbus interrupt level 5, IRQ5
19	User Indicator LED	4	VMEbus interrupt level 4, IRQ4
18	VMEbus arbitration, SYSARB	3	VMEbus interrupt level 3, IRQ3
17	VMEbus release mechanism, BCLR	2	VMEbus interrupt level 2, IRQ2
16	VMEbus release mechanism, ROR	1	VMEbus interrupt level 1, IRQ1
15	VMEbus release mechanism, RWD/PREL	0	SYSFAIL

The control register is not accessible from the VMEbus. Upon power-up or reset, all bits in the control register are cleared and must be initialized. Table 2-6 lists the power-up and reset defaults of the control register.

Table 2-6. Control Register Defaults

Bit(s)	Condition	Bit(s)	Condition
31-24	Mailbox location $0	11	D16 selected
23-22	DRAM base address at $0	10	D32 data size disabled
19	User IND LED illuminated	9	A24 access disabled
18	Round-robin arbitration	8	A32 access disabled
17-15	ROR VMEbus release	7-1	VMEbus interrupt handles enabled for all levels
14	DRAM cache disabled		
13	Instruction, data caches disabled	0	VMEbus SYSFAIL* asserted
12	MMU disabled		

Table 2-7 shows **bits 31** to **24** specifying VMEbus A16 mailbox interrupt address bits, MA15 to MA8 (Mailbox Address 15 to Mailbox address 8). See Section 2.4.3, "VMEbus Mailbox Autovectored Interrupt" for more information.

Table 2-7. Mailbox Interrupt VMEbus Address

Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
MA15	MA14	MA13	MA12	MA11	MA10	MA9	MA8

Table 2-8 shows **bits 23** to **20** specifying the VMEbus base address for the dual-access DRAM for both A24 and A32 addressing (RAMA23 to RAMA20). The memory resides in either A24 or A32 space selectable via the control register. The MZ 8130's memory is addressable on 1-Mbyte boundaries. The MZ 8134's memory is addressable on 4-Mbyte boundaries.

Table 2-8. Dual Access DRAM VMEbus Base Address

Description	Bit 23	Bit 22	Bit 21	Bit 20
RAM address	RAMA23	RAMA22	RAMA21	RAMA20
Board	MZ 8130	MZ 8130	MZ 8130	MZ 8130
	MZ 8134	MZ 8134		

For example, to set a starting address of 0x500000 on the MZ 8130, set bits 20 and 22. To set a starting address of 0xC00000 for either board, set bits 22 and 23.

Bit 19, IND LED, illuminates the user indicator on the front panel when cleared. Setting this bit turns the LED off.

Bit 18, SYSARB, selects the MZ 1000 VMEbus system controller for round robin or priority arbitration. Round-robin arbitration gives equal bus access to each bus request (BR) level on a rotating basis. Priority arbitration gives BR3 the highest bus access priority and BR0 the lowest. Table 2-9 summarizes bit 18.

Table 2-9. System Controller Arbitration

Bit 18	Arbitration
1	Round robin
0	Priority

Bit 17, BCLR, enables the Bus CLeaR VMEbus release mechanism when ROR and RWD/PREL are disabled. The BCLR release mechanism is used primarily during priority arbitration by the VMEbus system controller to remove a lower priority bus master when a higher bus request level is asserted. During a VMEbus access by a lower priority bus master, a higher priority master requests the VMEbus, the system controller asserts BCLR to remove the lower priority bus master from the bus.

After receiving a BCLR, the MZ 8130/MZ 8134 releases the VMEbus early by negating BBSY during the last VMEbus transfer or immediately if the MZ 8130/MZ 8134 is not currently performing a VMEbus cycle. Enable BCLR by setting bit 17 and disabling ROR and RWD/PREL.

Table 2-10 summarizes the selection of VMEbus release mechanisms.

Table 2-10. VMEbus Release Mechanisms († indicates "Don't Care")

VMEbus Release Mechanism	BCLR Bit 17	ROR Bit 16	RWD/PREL Bit 15
BCLR Release	1	1	0
ROR Release	1	0	0
RWD Release	†	†	1
PREL Release	0	1	0 - Keep
	0	1	1 - Release

Bit 16, ROR, enables the Release On Request VMEbus release mechanism when RWD/PREL is disabled. When using ROR, the MZ 8130/MZ 8134 relinquishes bus mastership when any VMEbus request signal is asserted. ROR is the most efficient bus release mechanism for a single processor configuration since the MZ 8130/MZ 8134 releases the VMEbus only when another bus master is requesting the VMEbus. At that time, it releases the bus almost immediately, allowing VMEbus arbitration to occur concurrently with the last VMEbus cycle. If the MZ 8130/MZ 8134 is not performing a VMEbus bus cycle, the VMEbus is released immediately. Enable ROR by clearing bit 16 and disabling RWD/PREL (bit 15 cleared). ROR is disabled when this bit is set.

Bit 15, RWD/PREL, enables the Release When Done with each VMEbus cycle and Program RELease VMEbus release mechanisms. In RWD mode, the MZ 8130/MZ 8134 releases the VMEbus early by negating BBSY during each VMEbus cycle. RWD is very useful if the VMEbus very active and the MZ 8130/MZ 8134 requires the VMEbus on a limited basis. RWD is enabled when bit 15 is set which overrides both BCLR and ROR bus release mechanisms.

Program RELease, PREL, is a software VMEbus release mechanism in which the MZ 8130/MZ 8134 does not release the VMEbus until control register bit 15 is set. This is very useful if your system has routines which must be completed before another master can be allowed to access the VMEbus. PREL is enabled by clearing bit 15 and disabling both ROR and BCLR. The MZ 8130/MZ 8134 retains VMEbus mastership until software clears bit 15 or until either ROR or BCLR is enabled.

3 Clearing bit 15 always releases VMEbus mastership immediately.

Table 2-11 shows bits 14 to 12 which affect the cache and memory management unit.

Table 2-11. Cache and Memory Selection

Bit 14	Bit 13	Bit 12
CACHERAM	CACHDIS*	MMUDIS*

Bit 14, CACHERAM, when set enables the on-board DRAM to be a CPU cacheable memory area by disabling all VMEbus write accesses to the DRAM. By preventing VMEbus write accesses, the instruction codes stored in the CPU's internal cache always match the same code stored in the MZ 8130/MZ 8134's DRAM ensuring cache coherency.

Since the MXbus is a direct extension of the MZ 8130/MZ 8134 CPU bus, master side cards have write capability to DRAM even if Bit 14 is set.

Bit 13, CACHDIS*, controls whether or not all CPU instruction bus transfers are cached according to the CPU's internal cache control register. Care must than be taken to be sure the cacheable instruction code cannot be changed by another VMEbus or MXbus master. If cacheable instruction code is altered by another master, the code in the CPU's internal instruction cache will not match the actual code for that address. Set this bit to enable the CPU instruction cache.

Bit 12, MMUDIS*, when set enables the MC68030's on-chip memory management unit. The MC68EC030 has no memory management unit.

Table 2-12 summarizes data path size selection.

Table 2-12. Data Path Size Selection

Bit 11 D32SIZ*	Bit 10 D32DIS*	Description
1	1	D32 data path size
0	0	D16 data path size
A16 and A24	A32 and slave mode DRAM	Access type

Bit 11, D32SIZ, selects the data path size for VMEbus master accesses in the A16 and A24 address spaces. Clear this bit to select D16 capability and set it to select D32 capability. See Section 2.1.4.1, "The MZ 8130/MZ 8134 as a VMEbus Master," for more information.

To utilize D32 transfers, the MXbus side card must provide the circuitry to connect to the upper 16 VMEbus data lines on the P2 connector.

Bit 10, D32DIS*, selects the data path size for VMEbus master accesses in the A32 address space, and for external VMEbus accesses to the dual access DRAM. Clear this bit to select D16 capability and set it to select D32 capability.

To utilize A32/D32 transfers, the MXbus side card must provide the circuitry to connect to the upper 8 VMEbus address lines and the upper 16 VMEbus data lines on the P2 connector.

Bit 9, A24DIS*, when set selects the MZ 8130/MZ 8134's DRAM to reside in the VMEbus A24 address space. Control register bits RAMA23 and RAMA22 (bits 23 and 22) select the DRAM's VMEbus address on any 4 Mbyte boundary. When this bit is cleared, the MZ 8130/MZ 8134's DRAM is not accessible in the VMEbus A24 address space.

Bit 8, A32DIS*, when set selects the MZ 8130/MZ 8134's DRAM to reside in the VMEbus A32 address space. The MXbus side card must decode the upper VMEbus A31 to A24 address lines and assert the MXbus signal A32MATCH* when a match occurs on these lines. Control register bits RAMA23 and RAMA22, (bits 23 and 22) are still used to select the MZ 8134's DRAM A32 address on any 4-Mbyte boundary; control register bits RAMA 23-20 select the MZ 8130's address on any 1-Mbyte boundary. If bit 9 and bit 8 are cleared, the DRAM cannot be accessed by the VMEbus. If bits 8 and 9 are set, the DRAM is accessible in both A24 and A32 address spaces. Table 2-13 summarizes DRAM address space selection.

Table 2-13. DRAM Address Space Selection

Bit 9 A24DIS	Bit 8 A32DIS	Results
1	0	A24 only
0	1	A32 only
1	1	Both A24 and A32
0	0	Inaccessible from VMEbus

Table 2-14 summarizes interrupt handler level selection.

Table 2-14. Interrupt Handler Levels

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
VMEbus IRQ7	VMEbus IRQ6	VMEbus IRQ5	VMEbus IRQ4	VMEbus IRQ3	VMEbus IRQ2	VMEbus IRQ1

Bits 7 to 1, VMEbus IRQ7 to VMEbus IRQ1, select the VMEbus interrupts handled by the MZ 8130/ MZ 8134. To handle a VMEbus interrupt level, clear this bit. To enable another VMEbus master to handle a VMEbus interrupt level, set the corresponding bit.

3

Only ONE VMEbus master can handle a particular VMEbus interrupt level.

The MZ 8130/MZ 8134 VMEbus interrupt handler must not be configured via the control register to respond to either of the side card VMEbus interrupt levels selected by jumper block K6.

Bit 0, SYSFAIL*, asserts the VMEbus SYSFAIL* signal when cleared. When this bit is set, VMEbus SYSFAIL* is negated.

Bit 0 must be set for normal operation.

2.1.2.2 Status Register

The 8-bit on-board, read-only status register allows software access to the status of ACFAIL* and SYSFAIL* VMEbus signals, the side card sense signal, and the CPU ID number selected by jumper block K2. The register is located at word address A32 $FEFC0000 or A24 $FC0000 on data lines D31 to D24.

The status register is not accessible from the VMEbus.

Table 2-15 describes the status register.

Table 2-15. Status Register

31	30	29	28	27-24
ACFAIL*	SYSFAIL*	MXSENSE*	Reserved	CPU ID

Bit 31, ACFAIL*, reflects the status of the VMEbus ACFAIL* signal. This bit is cleared when the VMEbus ACFAIL* signal is asserted. The VMEbus ACFAIL* signal also initiates a CPU non-maskable autovectored level 7 interrupt.

Bit 30, SYSFAIL*, reflects the status of the VMEbus SYSFAIL* signal. This bit is cleared when the VMEbus SYSFAIL* signal is asserted by any VMEbus board.

Bit 29, MXSENSE*, reflects the state of the MXSENSE* signal, which is pulled high by the MZ 8130/MZ 8134. If you design a side card, you can pull this signal low to indicate any user-defined status.

Bit 28 is reserved and should be ignored.

Bits 27 to 24, CPU ID, reflect the CPU ID selected by jumper block K2.

2.1.3 MZ 8130/MZ 8134 Initialization

The power-up and reset defaults of the MZ 8130/MZ 8134 are shown in Table 2-16.

Table 2-16. MZ 8130/MZ 8134 Defaults

Feature	Default
Mailbox location base address	$0
DRAM base address	$0
User IND LED	Illuminated
DRAMaccess	Disabled
Instruction cache	Disabled
Data cache	Disabled
MMU	Disabled
Data size	D16
A24 and A32 Access	Disabled
MXbus	Disabled
VMEbus:	
System controller	Enabled
Bus time-out	Enabled
Arbitration	Round-robin
VMEbus release	ROR
Interrupt handler	Enabled for all levels
VMEbus SYSFAIL*	Asserted

After power-up or reset, the following devices must be initialized:

- Control register:
  - Negate the VMEbus SYSFAIL signal
  - Turn-off the USER IND LED
  - Set the DRAM and mailbox VMEbus addresses
  - Select the VMEbus release mechanism
  - Disable any or all VMEbus interrupt handlers
  - Select the MXbus transfer size.
- SCN68681 DUART (serial ports/timer)

2.1.4. Configuring the VMEbus Interface

The MZ 8130/MZ 8134 provides A16 and A24 VMEbus addressing and D16 data paths to and from the VMEbus. When a VMEbus A32:D32 MXbus side card is present, the MZ 8130/MZ 8134 can function with full 32-bit master and slave VMEbus interface capabilities, including VMEbus un-aligned transfers if the side card is so equipped. The DRAM is addressed in the VMEbus A24 address space on a 4-Mbyte boundary for the MZ 8134 and a 1-Mybte boundary for the MZ 8130, while the mailbox interrupt is addressed in the VMEbus A16 address space on a 256-byte boundary. Both VME addresses are selected with the control register.

If an MXbus side card that decodes the upper VMEbus addresses is present, the DRAM can also reside in the VMEbus A32 address space. The MZ 8130/MZ 8134's DRAM can be completely isolated from the VMEbus.

The MZ 8130/MZ 8134's VMEbus requester circuit provides four VMEbus release mechanisms: Release On Request (ROR), Release When Done (RWD), Release on Bus CLeaR (BCLR), and Programmed RELease (PREL). The VMEbus BuSY (BBSY) signal is released early to allow bus arbitration to occur during the last MZ 8130/MZ 8134 bus access, allowing higher VMEbus transfer rates.

2.1.4.1 The MZ 8130/MZ 8134 as a VMEbus Master

When the CPU address bus asserts a VMEbus address, the MZ 8130/MZ 8134 requests the bus and performs a VMEbus access. The MZ 8130/MZ 8134 alone can address both the A16 and A24 VMEbus address spaces with D16 transfers. With an MXbus side card with the appropriate address buffers on the side card's VMEbus P2 connector address signals, the MZ 8130/MZ 8134 can perform VMEbus A32 accesses.

If the VMEbus A32 space is accessed when there is no VMEbus A32 MXbus side card, a VMEbus A32:D16 cycle takes place with the upper eight VMEbus address lines pulled-up ($FF) by the VMEbus backplane.

The MZ 8130/MZ 8134 can also perform VMEbus D32 transfers with an MXbus side card which has data buffers on the side card's VMEbus P2 data signals. The VMEbus D32 capability is enabled by D32DIS* (control register bit 10) and D32SIZ* (control register bit 11). The D32DIS* control register bit enables VMEbus D32 transfers for VMEbus A32 accesses. The D32SIZ* control register bit enables D32 transfters for A16 and A24 accesses.

MXbus A24 and A32 master side cards access the VMEbus by first becoming bus master of the local CPU bus. Two separate address schemes are used for MXbus A24 and A32 side card masters, as shown in Table 2-17.

Table 2-17. VMEbus Access for CPU and 32-bit side card masters († indicates "Don't Care")

32-bit Address	VMEbus Access	Bit 10 D32DIS*	Bit 11 D32SIZ*
MZ 8130			
$FFFF0000 – $FFFFFFFF	A16:D16	†	0
$FFFF0000 – $FFFFFFFF	A16:D32	†	1
$00100000 – $00FFFFFF	A24:D16	†	0
$00100000 – $00FFFFFF	A24:D32	†	1
$01000000 – $F7FFFFFF	A32:D16	0	†
$01000000 – $F7FFFFFF	A32:D32	1	†
MZ 8134			
$FFFF0000 – $FFFFFFFF	A16:D16	†	0
$FFFF0000 – $FFFFFFFF	A16:D32	†	1
$00400000 – $00FFFFFF	A24:D16	†	0
$00400000 – $00FFFFFF	A24:D32	†	1
$01000000 – $F7FFFFFF	A32:D16	0	†
$01000000 – $F7FFFFFF	A32:D32	1	†

Table 2-18. VMEbus Access for 24-bit side card masters († indicates "Don't Care")

32-bit Address	VMEbus Access	Bit 10 D32DIS*	Bit 11 D32SIZ*
MZ 8130			
$FF0000 – $FFFFFF	A16:D16	†	0
$FF0000 – $FFFFFF	A16:D32	†	1
$200000 – $BFFFFF	A24:D16	†	0
$200000 – $BFFFFF	A24:D32	†	1
MZ 8134			
$FF0000 – $FFFFFF	A16:D16	†	0
$FF0000 – $FFFFFF	A16:D32	†	1
$400000 – $BFFFFF	A24:D16	†	0
$400000 – $BFFFFF	A24:D32	†	1

2.1.4.2 CPU CAS and CAS2 Instruction VMEbus Incompatibility

The CAS and CAS2 instructions of the CPU can execute one large indivisible cycle with several reads and writes at different addresses. The VMEbus, however, does not support this type of operation. The *VMEbus Specification, Revision C.1* states that the address must not change during the course of any read/write indivisible cycle.

The guideline for using this instruction is that CAS and CAS2 instructions can be executed ONLY if the operands read/written reside entirely within the on-board or MXbus side card DRAM.

2.1.4.3 The MZ 8130/MZ 8134 as a VMEbus Slave

The dual access DRAM resides in the VMEbus A24 address space when enabled by the A24DIS* control register bit. When a A32 MXbus side card with address decoding of the upper VMEbus P2 connector address signals is used, the DRAM can also reside in the VMEbus A32 address space. VMEbus A32 is enabled by the A32DIS* control register bit.

If an MXbus side card has VMEbus data buffers for the VMEbus P2 connector data signals and D32DIS* (control register bit 10) is enabled, the DRAM can then be accessed with D32 transfers, including VMEbus unaligned transfers.

If an MXbus side card has both a VMEbus P2 address comparator and data buffers, the DRAM can be accessed by A24:D16, A24:D32, A32:D16, and A32:D32 transfers.

If control register bit 13 is set to allow only VMEbus read accesses to the DRAM, data in the DRAM can be cacheable in the CPU instruction cache (control register bit 14). The default register setting is cache disabled.

3 If both control register bits A24DIS* and A32DIS* are cleared, VMEbus DRAM accesses cannot occur.

2.1.4.4 VMEbus Address Modifier Codes

An address modifier (AM) code is asserted with each VMEbus address broadcast. The AM code determines which VMEbus address space is accessed: Short I/O (A16), Standard (A24), or Extended (A32). Table 2-19 indicates the VMEbus address modifier codes asserted for the various address spaces.

Table 2-19. VME Address Modifier Codes

VMEbus Standard (A24):	
39:	A24 user data access
3A:	A24 user instruction access
3D:	A24 supervisor data access
3E:	A24 supervisor instruction access
VMEbus Short (A16):	
29:	A16 user access
2D:	A16 supervisor access
VMEbus Extended (A32):	
09:	A32 user data access
0A:	A32 user instruction access
0D:	A32 supervisor data access
0E:	A32 supervisor instruction access

When the MZ 8130/MZ 8134's DRAM is being accessed by another VMEbus master, the VMEbus address modifier codes are checked according to control register bits 9 and 8 (A24DIS and A32DIS). When the A24DIS control register bit is enabled and a VMEbus address decoding checks for a standard (A24) address modifier code. When the A32DIS control register bit is enabled, indicating a VMEbus A32 MXbus side card is present, the VMEbus decoding checks for an extended (A32) VMEbus address modifier code. The mailbox interrupt uses short I/O (A16).

2.1.4.5 The MZ 8130/MZ 8134 as VMEbus Requester

When the local CPU bus asserts a VMEbus address, the MZ 8130/MZ 8134 requests to become the VMEbus master. The bus requester circuit then asserts a VMEbus request on the level determined by jumper block K7. Upon receiving the bus request, the VMEbus system controller asserts a bus grant signal after the current bus master relinquishes the bus. When the MZ 8130/MZ 8134 receives the bus grant signal as selected by jumper block K8 (must be on the same level as K7), the MZ 8130/MZ 8134 acknowledges bus mastership by asserting the VMEbus busy, BBSY, signal. The MZ 8130/MZ 8134 then performs at least one VMEbus cycle.

Control register bits 15 to 13 select one of four MZ 8130/MZ 8134 bus release schemes:

- Release On Request (ROR) of the VMEbus by another VMEbus master
- Programmed RELease (PREL) in which bus release is software controlled
- Release When Done (RWD) in which the VMEbus is released during each VMEbus cycle
- Release on Bus CLeaR (BCLR)

Refer to Section 2.1.2.1, "Control Register," for details of bus release mechanisms.

2.1.5 Configuring the MXbus Side Card Interface

The MZ 8130/MZ 8134 implements the MXbus for D16 and/or D32 bit data paths for A32 or A24 master/slave side cards. The MXbus side card interface connector, S1, is located on the side of the MZ 8130/MZ 8134 board. The interface allows you to expand the CPU bus with MXbus master and slave side cards, such as the MZ 8800 Series. Thirty-two address signals, thirty-two data signals, CPU control signals, CPU clock, two CPU autovectored interrupt signals, two VMEbus interrupt signals, and MZ 8130/MZ 8134 control signals are available via the MXbus side card interface.

The MXbus expansion header, K1, provides additional signals to allow CPU high-speed memory to be placed on an MXbus side card. See Section 1.5.1, "K1: MXbus Expansion Header," for details.

2.1.5.1 MXbus A24 Side Card Slave

The MXbus structure is derived from the MC68030/MC68EC030 microprocessor interface. By leaving the MXbus signal COMPAT32* unasserted, the MXbus can be configured for A24 MXbus slave cards, such that the address and data strobes are decoded to match the MC68000 microprocessor interface. The address and data strobes are asserted only for accesses in side card address spaces $F8000000 to $FEDFFFFF and $FEFF0000 to $FFFEFFFF and during VMEbus A16 accesses $FFFF0000 to $FFFFFFFF.

Because the CPU is an A32 device, it accesses the A24 MXbus address space at several different A32 addresses. Table 2-20 below shows all CPU addresses that result in the same A24 address for MXbus A24 slave side cards.

Table 2-20. A32 CPU Addresses That Access Same A24 Address on A24 MXbus Slave Side Card

A32 Addresses Access A24: $000000 - $FFFFFF	
A32: $F8000000 -	$F8FFFFFF
A32: $F9000000 -	$F9FFFFFF
...	...
A32: $FD000000 -	$FDFFFFFF
A32: $FE000000 -	$FEDFFFFF
A32: $FF000000 -	$FFFFFFFF

The data width for MXbus A24 slaves is controlled by the data size acknowledge signals (DSACK0*, DSACK1*). A24:D16 slaves using MC68000 signals may treat DSACK1* as DTACK* for 16-bit data transfers with the CPU.

2.1.5.2 MXbus A32 Side Card Slave

MXbus A32 slave side cards can have D16 and/or D32 data widths in accordance with the assertion of MXbus signals DSACK0-1*. A32 slaves are not required to assert the MXbus COMPAT32* signal to use the full A32 MXbus address spaces $F8000000 to $FEDFFFFF and $FF000000 to $FFFEFFFF.

Asserting the active low COMPAT32* MXbus signal allows the MXbus address and data strobes to be asserted during **ALL** CPU bus cycles and converts MXbus data strobe UDS* into address signal A0. MXbus A32 slave side cards that assert the COMPAT32* signal are required to decode all 32 address bits for **ALL** CPU cycles.

2.1.5.3 MXbus A24 Side Card Master

The MXbus COMPAT32* signal, pulled high by the MZ 8130/MZ 8134, selects A24 address decoding for MXbus A24 master accesses. The MZ 8130/MZ 8134's address decoding is redefined when an MXbus A24 side card master becomes bus master of the CPU microprocessor bus as shown in Tables 2-21 (MZ 8130) and 2-22 (MZ 8134). The CPU microprocessor bus signals AS*, DS*, A0, SIZ0, and SIZ1 are encoded during each A24 access.

Table 2-21. MZ 8130 A32 Address Decoding

Address (Hex)	Ports	Device
$00000000 - $000FFFFF	32	Local DRAM
$00100000 - $00FFFFFF	16/32	VMEbus Standard A24 Address Space
$01000000 - $F7FFFFFF	16/32	VMEbus Extended A32 Address Space
$F8000000 - $FEDFFFFF	16/32	MXbus Side Card Address Space
$FEE00000 - $FEEFFFFF	16	ROM/EPROM, SmartWatch
$FEF00000 - $FEFBFFFF	NA	Reserved
$FEFC0000 - $FEFCFFFF	16	Status Register
$FEFD0000 - $FEFDFFFF	32	Control Register
$FEFE0000 - $FEFEFFFF	8	SCN68681 DUART Serial Port / Timer
$FEFF0000 - $FEFFFFFF	NA	Reserved
$FF000000 - $FFFEFFFF	16/32	MXbus Side Card Address Space
$FFFF0000 - $FFFFFFFF	16/32	VMEbus Short A16 Address Space

Table 2-22. MZ 8134 A32 Address Decoding

Address (Hex)	Device
$000000 - $3FFFFF	Local DRAM
$400000 - $BFFFFF	VMEbus Standard A24 Address Space
$C00000 - $DFFFFF	MXbus Side Card Address Space (Open)
$E00000 - $EFFFFF	ROM/EPROM, SmartWatch
$F00000 - $FBFFFF	Reserved
$FC0000 - $FCFFFF	Status Register
$FD0000 - $FDFFFF	Control Register
$FE0000 - $FEFFFF	SCN68681 DUART Serial Port/Timer
$FF0000 - $FFFFFF	VMEbus Short A16 Address Space.

MXbus A24 master side cards can use D16 or D32 data path widths by using LDS* and UDS*. In order to take advantage of the MC68030 and MC68EC030 dynamic bus sizing feature, the side card must drive the SIZE0 and SIZE1 signals and monitor DSACK1 and DSACK0. The simpler MC68000-type interface may be employed, but without the advantage of dynamic bus sizing.

Control register bit D32DIS* selects either D16 or D32 data paths for the VMEbus accesses to/from the MZ 8130/MZ 8134. To use VMEbus transfers, the side card must provide the circuitry to connect to the upper 16 data lines on the P2 connector.



D16 MXbus masters cannot access the 32-bit control register.

2.1.5.4 MXbus A32 Side Card Master

MXbus A32 master side cards use the CPU microprocessor address bus the same way the CPU does. A32 master side cards must assert the MXbus COMPAT32* MXbus signal to enable A32 address decoding. The assertion of COMPAT32* signal also requires the side card to assert DS*, A0, the SIZ0-1 size signals, and RMC* as required by the MC68030 and MC68EC030 (consult the *MC68030 User's Manual* for more informtion).

To allow accesses to the VMEbus A32 address space, the side card must buffer the VMEbus P2 connector address signals.

MXbus A32 master side cards can also have D16 or D32 capability for VMEbus accesses. D32 accesses require appropriate circuitry to connect to the VMEbus data lines on the P2 connector.

2.2 Using the SCN68681 DUART Serial Port Chip

The SCN68681 DUal Asynchronous Receiver/Transmitter (DUART) serial port chip provides two independent full-duplex asynchronous RS-232-C serial ports (RJ11 connectors) on the front panel of the MZ 8130/MZ 8134. A 3.6864 MHz oscillator allows the use of 18 standard asynchronous baud rates from 50 baud to 38.4K baud. An SCN68681 internal 16-bit programmable timer can be used to generate custom baud rates or as a timer interrupt to the CPU. The SCN68681 DUART can assert a level 3 vectored interrupt to the CPU.

Use NOP instructions to provide the required recovery time of 1 µs between accesses.

When using the SCN68681 in polled mode operation, the break and error bits in the status register must be checked only after a character has been received, indicated by bit-0 of the SCN68681 status register being set.

Table 2-23 shows the SP0, SP1 - DCE Serial Port RS-232-C pin assignments. Refer to Section 3.5.2 for the RS-232-C interface specifications.

Table 2-23. SP0, SP1 - DCE Serial Port RS-232-C Pin Assignments

Pin	Direction	Designation	Function
1	Receives	CTS	Clear To Send
2	Receives	RD	Receive Data
3	Supplies	GND	Ground
4	Supplies	GND	Ground
5	Supplies	TD	Transmit Data
6	Supplies	RTS	Request To Send

2.2.1 Initializing the SCN68681 DUART

The SCN68681 DUART's base address is $FEFE0000 in the A32 address space and $FE0000 in the A24 address space. The SCN68681 is a byte-wide port and requires data to be transferred on data lines D7 to D0 (byte 3). Table 2-24 shows the address of each SCN68681 register.

Table 2-24. Addresses of SCN68681 Registers

Address	Read_Accesses	Write_Accesses
$FE FE0000	Mode Register A (MR1A,MR2A)	Mode Register A (MR1A, MR2A)
$FE FE0002	Status Register A (SRA)	Clock-Select Register A (CSRA)
$FE FE0004	*Reserved*	Command Register A (CRA)
$FE FE0006	Rx Holding Register A (RHRA)	Tx Holding Register A (THRA)
$FE FE0008	Input Port Register (IPCR)	Aux. Control Register (ACR)
$FE FE000A	Interrupt Status Register (ISR)	Interrupt Mask Register (IMR)
$FE FE000C	Cntr/Tmr High Register (CTR)	Cntr/Tmr Upper Register (CTUR)
$FE FE000E	Cntr/Tmr Low Register (CTL)	Cntr/Tmr Lower Register (CTLR)
$FE FE0010	Mode Register B (MR1B,MR2B)	Mode Register B (MR1B, MR2B)
$FE FE0012	Status Register B (SRB)	Clock-Select Register B (CSRB)
$FE FE0014	*Reserved*	Command Register B (CRB)
$FE FE0016	Rx Holding Register B (RHRB)	Tx Holding Register B (THRB)
$FE FE0018	Interrupt-Vector Register (IVR)	Interrupt-Vector Register (IVR)
$FE FE001A	Input Port	Output Port Conf. Register (OPCR)
$FE FE001C	Start-Counter Command	Set Output Port Bits Command
$FE FE001E	Stop-Counter Command	Reset Output Port Bits Command

Note that read and write accesses to the same address can access different registers and that MR1*x* and MR2*x* registers have the same address. An internal pointer in the SCN68681 points to MR1*x* after a reset or power-up. An access to the same address results in the pointer being set to MR2*x*. Accessing MR2*x* does not change the pointer's position, but writing the reset pointer command, $10, into the CR*x* register, resets the pointer to MR1*x*.

Initializing the SCN68681 DUART for polled mode operation requires initializing the Mode Registers (MR1*x*, MR2*x*), Clock Select Register (CSR*x*), Command Register (CR*x*), Interrupt Mask Register (IMR), and Output Port Bits Command Register properly. These registers configure the serial port for data length, parity, baud rate, mask interrupts, de-assert RTS, CTS, DTR, and DSR and enable the serial port.

For example, to configure port 0 for 8 bits, no parity, one stop bit, 9600 baud, and to disable RTS, CTS, DTR, and DCD, follow the steps below:

Step 1. Write $13 into MR1A to set 8 bits, no parity.
Step 2. Write $7 into MR2A to set 1 stop bit.
Step 3. Write $BB into CSRA to set baud rate to 9600.
Step 4. Write $5 into CRA to enable transmitter and receiver.
Step 5. Write $0 into IMR to clear interrupt masks register.

Section 2.2.5, "Programming the SCN68681," provides more information on programming.

2.2.2 Transmitting and Receiving Characters

Once the SCN68681 is initialized, bit 2 of each port's status register (SRx) may be polled to determine whether the Tx Holding Register (THRx) for that port is ready for a character. Once the bit is set, a character may be sent by writing it to THRx. The SCN68681 then transmits the character over the corresponding port.

When a character is received from a serial port, bit 0 of the corresponding status register is set and the character is available in the Rx Holding Register (RHR*x*). A simple serial port driver polls bit 0 of the status register of the specified serial port. When the bit is set, the received character can be read from the Rx Holding Register.

2.2.3 Using the Timer

The SCN68681 DUART includes a programmable counter/timer. In counter mode, the timer can be read, stopped, or started under software control. In the timer mode, it can interrupt the CPU on vectored interrupt level 3 when the counter reaches its terminal value. Refer to the *SCN68681 Dual Asynchronous Receiver/Transmitter (DUART) Product Specification* for more information.

2.2.4 Using Interrupts

The SCN68681 DUART can interrupt the CPU, allowing it to perform other tasks until an SCN68681 serial port or timer requires service.

To use an interrupt routine, you must set up an interrupt vector in the SCN68681 IVR register which will be asserted during an SCN68681 interrupt acknowledge cycle. This interrupt vector points to the location in the interrupt vector table containing the address location of the SCN68681 interrupt handler routine. This routine performs the actual reading or writing of the serial port or timer. The interrupt handler must read the ISR register to determine the source of the interrupt if more than one SCN68681 interrupt is possible.

For example, you can program the SCN68681 to send a vectored level 3 interrupt to the CPU when an Rx holding register receives a character or becomes empty or when the timer reaches the terminal count condition.

2.2.5 Programming the SCN68681

The following programming example for an OS-9™ system initializes the SCN68681 serial port 0 for polled mode operation and transmits a short message. The program then waits for a character to be received by polling bit 0 of the status register A. When the bit 0 is set, the received character is moved from the Rx holding register into register d0 in the CPU.

```
**************************************************************
*
* Procedure:    SCN68681 polled mode software program.
*               All code is written in 680x0 assembly in OS-9 Macro Assembler syntax.
*
* Description:  This program initializes the serial port 0 (SP0) and 1 (SP1) for polled mode
operation
*               at 9600 baud, 1 stop bit, no parity, no handshaking, and no
*               interrupts. It tests SP0 by sending a short message "Serial port 0 is
*               up". It shows how a character can be received from SP0 and
*               placed into register d0 and how a character in register d1 can be transmitted
*               via SP0 in a simple program that is not executed.                    *
**************************************************************************
OS-9 System Level Operands:
**************************************************************************
use /ho/defs/oskdefs.d
Editionequ    1
Typ_Lang      equ     (Prgrm<<8)+Objct
Attr_Rev      equ     (ReEnt<<8)+0
        psect serial,Typ_Lang,Attr_Rev,Edition,0,serial
*
```

```
*****************************************************************

serbase:  equ=    $fefe0000       68681 base address
mra       equ     serbase+0       mode register A (mr1a, mr2a)
sra       equ     serbase+$2      status register A (sra)
csra      equ     serbase+$2      clock select register A (csra
cra       equ     serbase+$4      command register A (cra)
rhra      equ     serbase+$6      Rx holding register A (rhra)
thra      equ     serbase+$6      Tx holding register A (thra)
imr       equ     serbase+$a      interrupt mask register (imr)
mrb       equ     serbase+$10     mode register B (mr1b,mr2b)
srb       equ     serbase+$12     status register B (srb)
csrb      equ     serbase+$12     clock select register B (csrb)
crb       equ     serbase+$14     command register B (crb)
rhrb      equ     serbase+$16     Rx holding register B (rhra)
thrb      equ     serbase+$16     Tx holding register B (rhra)
setout    equ     serbase+$1c     set outputs port bits command register

*****************************************************************
Macro Definitions
*****************************************************************
*
bugcall MACRO

        trap #15
        dc.w \1
        ENDM
*
*****************************************************************
Initialize Serial Ports 0 and 1
*****************************************************************
*
initports:  move.b   #$13   ,mra    SP0: 8 bits, no parity
            move.b   #$7    ,mra    SP0: 1 stop bit
            move.b   #$13   ,mrb    SP1: 8 bits, no parity
            move.b   #$7    ,mrb    SP1: 1 stop bit
            move.b   #$bb   ,csra   SP0: 9600 baud
            move.b   #$bb   ,csrb   SP1: 9600 baud
            move.b   #$5    ,cra    SP0: enable transmitter and receiver
            move.b   #$5    ,crb    SP1: enable transmitter and receiver
            move.b   #$f    ,setout disable RTS, DTR, and DCD
            move.b   #$0    ,imr    clear interrupt mask register
```

```
**************************************************************************
 Print "Serial port 0 is up" to SP0
**************************************************************************
*
        lea mess1,a0                point a0 to beginning of message

loop
        move.b (a0)+,d0             get next character in message
        beq.s ender                 test for end of message =0

txwait
        btst.b #$4 ,sra             is SP0 ready to transmit
        beq.s txwait                loop until SP0 is ready

        move.b d0 ,thra             send a character to SP0
        bra.s loop                  loop for next character in message

ender
        bugcall return              return to monitor
*
**************************************************************************
Example:  Receive a character from SP0 and place it in register d0
**************************************************************************
*
getchar
        btst.b #$1 ,sra         is a character available at SP0?
        beq.s getchar           loop until character is available
        move.b rhra ,d0         put character into register d0

**************************************************************************
Example:  Take a character in register do and transmit it out SP0
**************************************************************************
*
putchar
        btst.b #$4 ,sra         is SP0 ready to transmit?
        beq.s putchar           loop until SP0 is available
        move.b d1 , thra        send the character in register d1 to SP0

**************************************************************************
Data
**************************************************************************
*
align
mess1   dc.b cr , lf, "Serial port 0 is up" , 0

align

**************************************************************************
```

2.3 Using the SmartWatch Time-of-Day Clock

The DS1216 SmartWatch is an option only on the 20 MHz MZ 8130. It is not an option for 25 MHz boards. The SmartWatch provides backed-up timekeeping information and maintains the date in year, month, day, hour, minute, second, and hundredths of seconds format. It automatically corrects for 28-, 29-, 30-, and 31-day months. The SmartWatch operates in either 24-hour format or 12-hour format with an AM/PM indicator.

The SmartWatch is located under an EPROM socket and is accessed by a highly structured sequence of 64 EPROM read cycles to both local address locations $10 and $14 as defined by the *DS1216E/F SmartWatch Product Data Sheet.* The odds of the recognition sequence occurring in EPROM code and causing an inadvertent access to the SmartWatch is less than 1 in 10 to the 19th power.

3 The code to access the SmartWatch must not reside in EPROM.

Prior to executing the first 64 read cycles, an EPROM read cycle to address location $0 must be performed to reset the SmartWatch's register pointer. After a correct pattern of address accesses to both locations is completed, the SmartWatch is enabled and the next 64 EPROM read cycles to locations $10 and $14 cause the SmartWatch to either receive data or transmit data on local data line d0.

Cycles to other address locations outside of the EPROM space can be interleaved without interrupting the pattern recognition or data transfer sequence to the SmartWatch. The SmartWatch data is in BCD format.

A lithium energy cell embedded in the circuit board maintains the time and date automatically for more than ten years when there is no external power to the MZ 8130/MZ 8134. The energy cell cannot be replaced.

The SmartWatch programming example below sets the time and date in the DS1216 SmartWatch and then reads the newly set time and date from the SmartWatch. The program requires you to place the new time into the CPU's d0 register and the new date into register d1.

This example uses the MZ 8930 debug monitor's serial port print routines to print the newly set time/ date to a terminal attached to serial port 0 in an OS-9 system. For more details on programming the SmartWatch, refer to the Dallas Semiconductor *DS1216 E/F SmartWatch Product Data Sheet* (see "Related Publications" in the Preface).

```
*****************************************************************
* Procedure:    Read and write to DS1216 SmartWatch example.
*               All code is written in 680x0 assembly in OS-9 Macro Assembler syntax.
*
* Description:  This example sets the time/date and then reads the new time/date.
*               The time and date are set by entering the values into d0 and d1 before
*               the program is run.
*
*               Enter the time in d0 using this format: d0=$HHMMSSss:
*               For example, to set 9:34:20.43 PM in 24- hour mode, enter:
*                       d0 = $21342043
*
*               For example, to set 9:34:20.43 PM in 12-hour mode, enter:
*                       d0=$A9342043
*
*               Enter the date in d1 using this format:  d1=$YYMMdd0w
*               For example, to set the date to Jan 7,1991, day 5, enter:
*                       d1=$91010705
*
* NOTE:
* Because this code depends on a pattern being recognized by the SmartWatch, COMPLETE
* control of EPROM must be given to this routine, and it cannot be interrupted.
* This code MUST not reside in EPROM
* at the time of execution (it can be copied from EPROM elsewhere, then executed.)
*
*****************************************************************
* OS-9 System Operands
*****************************************************************
*
use /h0/defs/oskdefs.d
Edition  equ 1
Typ_Lang equ (Prgrm<<8)+Objct
Attr_Rev equ (ReEnt<<8)+0
psect  setclk,Typ_Lang,Attr_Rev,Edition,0,setclk
*
*****************************************************************
*   Assembler constants
*****************************************************************
readclk         equ     $fe810008       set reset bit (A16), and R/W bit (A3) for read.
writeclk        equ     $fe810000       set reset bit and clear R/W bit for write.
resetclk        equ     $fe800008       clear reset bit to reset SmartWatch.
*
*   NOTE:
*   If pin 1 of the EPROM is tied to VCC, a reset of theSmartWatch is not possible by
*   pulling pin 1 low. To assure a reset is done, a 'read' of the SmartWatch
*   must be done.  This is why the read bit of this constant is set.
*
```

```
cr          equ     13
lf          equ     10
write       equ     $0023
writeln     equ     $0024
pcrlf       equ     $0026
return      equ     $0063

*************************************************************************
*   Macro Definitions
*************************************************************************
*
bugcall MACRO
        trap #15
        dc.w \1
        ENDM
*
*       External routines:      binascii:  converts hex values to ascii characters (0-9, A-F)
*
*************************************************************************
* Code Start
*************************************************************************
*
setclk
        move.l d0,time          save time and date from registers set up
        move.l d1,date          prior to execution

        pea mess1               print start message
        bugcall writeln

        bsr writeclock          set current date/time
        bsr readclock           read current date/time
        bsr printdate           print date to verify setting
        bsr printtime           print time to verify setting

        bugcall return          return to debugger
*
*************************************************************************
* Prints date as found in location 'date'
*************************************************************************
*
printdate
        pea datemess            print initial message
        bugcall write

        move.l date,d0          d0 is temp.
        rol.l #8,d0             d0.b is now year
        bsr printd0byte         print year
```

```
        pea slash           print slash
        bugcall write
        rol.l #8,d0         d0.b is now month
        bsr printd0byte     print the month
        pea slash           print a slash
        bugcall write
        rol.l #8,d0         d0.b is now date
        bsr printd0byte     print date
        pea daymess         print 'day of week' message
        bugcall write
        rol.l #8,d0         d0.b is now day of week
        and.b #$07,d0       mask off upper bits of day
        bsr printd0byte     print day of week
        bugcall pcrlf
        rts

*************************************************************************
Prints time as found in location 'time'
*************************************************************************
printtime
        pea timemess        print time message
        bugcall write
        move.l time,d0      d0 is temp.
        rol.l #8,d0         d0.b is now hours stuff
        btst.l #7,d0        check for 12-24 format
        beq.s hour24        24 hour format
        move.l d0,-(sp)     save d0 for just a bit
        andi.b #$1f,d0      get rid of am-pm and 12-24 bit
        bsr printd0byte     print hours (12 hour format0
        move.l (sp)+,d0     restore d0 from save
        bra.s moretime

hour24
        bsr printd0byte     print hours (24 hour format)
moretime
        pea colon           print colon
        bugcall write
        rol.l #8,d0         d0.b is now minutes
        bsr printd0byte     print minutes
        pea colon           print colon
        bugcall write
        rol.l #8,d0         d0.b is now seconds
        bsr printd0byte     print seconds
        pea period          print '.'
        bugcall write
        rol.l #8,d0         d0.b is now .01 seconds
        bsr printd0byte     print .01 seconds
        btst.l #31,d0       check for 12-24 format
```

```
        beq.s ender             if 24 format, then all done
        btst.l #29,d0           check for AM,PM
        beq.s pram
        pea pmmess              print PM
        bugcall write
        bra.s ender
pram
        pea ammess              print AM
        bugcall write
ender
        bugcall pcrlf           print '<cr><lf>'
        rts

*************************************************************************
* Print Least Significant, LS, Byte of Register d0
*************************************************************************
*
printd0byte
        move.l d0,-(a7)         set up to print LS byte of d0
        move.l #$1,-(a7)
        pea string
        bsr binascii            convert hex to acsii
        pea string
        bugcall write           print LS byte of d0
        rts

*************************************************************************
* SmartWatch sequence generator:    Sends recognition sequence to SmartWatch
*                                   to enable reading and writing it.
*************************************************************************
*
sequence                        d0,d1,d2,a0 are temp.
        move.w resetclk,d0      dummy read to reset compare counter
        move.l #$5ca33ac5,d0    SmartWatch recognition sequence
        move.l #$3f,d1          64 operations (adjust -1 for dbra instruction)
nextptbit
        move.l d0,d2            move pattern into d2
        andi.l #$00000001,d2    mask all but one bit of pattern
        lsl.l #1,d2             shift bit up to address line A1,which corresponds to address
*                               line A0 of the SmartWatch
        move.l #writeclk,a0     load R/W bit in a0 for write to SmartWatch
        add.l d2,a0             add pattern bit to EPROM/SmartWatch base address
        move.w (a0),a0          read EPROM (does a write to theSmartWatch)
        ror.l #1,d0             rotate pattern right one bit
        dbra d1,nextptbit       loop for all 64 pattern bits
        rts
```

```
*************************************************************************
* SmartWatch Write Routine:  Sets the SmartWatch to date in locations 'time' and 'date'.
*************************************************************************
writeclock                  branch SmartWatch sequence generator routine
        bsr sequence        to enable SmartWatch for write access
        move.l time,d0      place time into register d0
        move.l date,d1      place date into register d1
        move.l #$1f,d2      32 operations (adj. -1 for dbra instruction)
        nextwrbit1
        move.l d0,d3        move time into register d3
        andi.l #$00000001,d3 mask all but one bit 1 (date bit of SmartWatch)
        lsl.l #1,d3         shift data to bit 1 (data bit of SmartWatch)
        move.l #writeclk,a0 clear R/W bit in a0 for write to SmartWatch
        add.l d3,a0         add data bit EPROM/SmartWatch base address
        move.w (a0),a0      read to EPROM (does write to SmartWatch)
        lsr.l #1,d0         shift for next bit of time
        dbra d2,nextwrbit1  loop for all 32 data bits

        move.l #$1f,d2      32 operations (adj. -1 for dbra inst.)
nextwrbit2
        move.l d1,d3        move date into register d3
        andi.l #$00000001,d3 mask all but one bit of date
        lsl.l #1,d3         shift data to bit 1 (data bit of SmartWatch)
        move.l #writeclk,a0 clear R/W bit in a0 for write to SmartWatch
        add.l d3,a0         add data bit EPROM/SmartWatch base address
        move.w (a0),a0      read EPROM (does write to SmartWatch)
        lsr.l #1,d1         shift for all 32 date bits
        dbra d2,nextwrbit2  loop for all 32 date bits
        rts
*
*************************************************************************
* SmartWatch Read routine:  Reads time and date in the  SmartWatch and places
                            them in locations time' and ' date'.
*************************************************************************
readclock                   branch to sequence generator routine bsr sequence
*                           to enable SmartWatch for read access
        clr.l d0            d0,d1,d2 are temp

        move.l #$1f,d1      32 operations (adj. -1 for dbra instruction)
nextrdbit1
        move.w readclk,d2   read one time bit from SmartWatch data is on LS bit of
*                           upper byte of word, so:
        andi.l #$00000100,d2 mask off all but SmartWatch data bit
        lsr.l #8,d2         shift data bit to LS byte of d2
        or.l d2,d0          'or' data into destination d0 register
        ror.l #1,d0         rotate data one place
        dbra d1,nextrdbit1  loop for all 32 bits
        move.l d0,time      place read time into location 'time'
```

```
        clr.l d0
        move.l #$1f,d1          32 operations (adj. -1 for dbra instruction)
nextrdbit2
        move.w readclk,d2       read one date bit from SmartWatch data is on LS bit of upper
*                                byte of word, so:
        andi.l #$00000100,d2    mask off all but SmartWatch data bit
        lsr.l #8,d2             get data to LS byte of d2
        or.l d2,d0              put data into destination (d0)
        ror.l #1,d0             rotate data one place
        dbra d1,nextrdbit2      loop for all 32 bits
        move.l d0,date          place read date into location 'date'
        rts
*************************************************************************
* Data
*************************************************************************
*
align

date            dc.l 0                  save location for date
time            dc.l 0                  save location for time
string          dc.b 8,'********'       used by 'binascii' for character conversion.

mess1           dc.b mess1end-*-1,cr,lf,"SmartWatch set to following parameters:"
mess1end        equ *
slash           dc.b slashend-*-1,'/'
slashend        equ *
datemess        dc.b dateend-*-1,"Date: "
dateend         equ *
daymess         dc.b dayend-*-1," Day of week: "
dayend          equ *
timemess        dc.b timend-*-1,"Time: "
timend          equ *
colon           dc.b colonend-*-1,":"
colonend        equ *
period          dc.b periodend-*-1,"."
periodend       equ *
ammess          dc.b amend-*-1," AM"
amend           equ *
pmmess          dc.b pmend-*-1," PM"
pmend           equ *
        align

        ends
```

2.3.1 Using the SmartWatch with 512K EPROMs

When using the SmartWatch with 512K EPROMs, pin 1 of the EPROM sockets (SmartWatch reset pin) is connected to an address line. The SmartWatch should be programmed to ignore pin 1 by setting the reset bit (bit 4 of the day register.) This must be done while the board is configured for an EPROM size other than 512K. Once the bit is set, the board can be reconfigured and used with 512K EPROMs, as long as care is taken never to clear the reset bit when writing to the SmartWatch.

2.4 CPU Interrupts

When the CPU receives an interrupt request, it temporarily stops execution of the current process and enters the exception processing state. After exception processing is completed, the CPU resumes execution at the point where the interrupt occurred.

Interrupt handling routines can also be interrupted by a higher priority interrupt depending on the interrupt mask of the interrupt handler.

Refer to the *MC68030 User's Manual* or the *VMEbus Specification, Revision C.1* for more information about interrupts.

The MZ 8130/MZ 8134 prioritizes the 13 separate interrupt request sources to the CPU shown in Table 2-25. When two interrupts are on the same interrupt level, the autovectored interrupt has a higher priority than the vectored interrupt. The vectored serial port interrupt has priority over the vectored VMEbus IRQ 3 interrupt.

Table 2-25. CPU Interrupt Priorities

	Autovectored		Vectored	
Level	**Priority**	**Description**	**Priority**	**Description**
7	1	ABORT, ACFAIL*	2	VMEbus IRQ7
6	3	S1IRQ1*	4	VMEbus IRQ6
5	5	MAILBOX*	6	VMEbus IRQ5
4	7	S1IRQ2*	8	VMEbus IRQ4
3			9	SCN68681 DUART
3			10	VMEbus IRQ3
2			11	VMEbus IRQ2
1			12	VMEbus IRQ1

2.4.1 ACFAIL* and ABORT Autovectored Interrupts

A non-maskable autovectored level 7 interrupt is asserted to the CPU when the ABORT switch is toggled or when the MZ 8130/MZ 8134 receives an ACFAIL* signal from the VMEbus backplane.

The ACFAIL* and ABORT autovector interrupt handler target address must be written into the DRAM at offset $7C in the MC68030 exception vector table. The interrupt handler routine located at the target address should read status register bit 31 to determine if an ACFAIL* or ABORT interrupt occurred. Bit 31 is cleared during a ACFAIL* interrupt and set during an ABORT interrupt.

2.4.2 Side Card S1IRQ1* and S1IRQ2* Autovectored Interrupts

The MXbus side card autovectored interrupts enable side cards to interrupt the CPU. S1IRQ1* asserts an autovectored interrupt request on level 6 while S1IRQ2* asserts an autovectored interrupt on level 4. An interrupt handling routine must be written for each interrupt level used. The interrupt handler target address for the S1IRQ1* interrupt is located in DRAM at offset $78 in the CPU exception vector table. The interrupt handler target address for the S1IRQ2* interrupt is located at offset $70 in the CPU exception vector table.

The MXbus side card is responsible for removing the interrupt request during the interrupt acknowledge cycle for that level.

2.4.3 VMEbus Mailbox Autovectored Interrupt

The mailbox autovectored level 5 interrupt enables any VMEbus master to interrupt the CPU by performing a write operation to the mailbox VMEbus address location. Use control register bits 31 to 24 to specify the VMEbus address on any 256-byte boundary in the VMEbus short A16 address space. See Section 2.1.2.1, "Control Register," for more information about the mailbox address. The MZ 8130/MZ 8134 can also initiate its own mailbox interrupt via the VMEbus. The request is latched until the interrupt is acknowledged, at which time it clears itself.

The mailbox's level 5 interrupt handler target address is located in DRAM at offset $74 in the CPU exception vector table.

2.4.4 DUART Serial Port/Timer Vectored Interrupts

The SCN68681 DUART can be programmed to send a vectored level 3 interrupt to the MC68030/MC68EC030 CPU when an Rx holding register receives a character, a Tx holding register becomes empty, or the timer reaches terminal count.

The interrupt vector which the SCN68681 asserts during an interrupt acknowledge cycle must be placed into the IMR register. This interrupt vector points to a location in the interrupt vector table which has the address location of the SCN68681 interrupt handler routine. The interrupt handler routine performs the actual reading or writing of the serial port or timer. The interrupt handler must read the ISR register to determine the cause of the interrupt if the SCN68681 is programmed to generate an interrupt for more than one condition.

2.4.5 VMEbus IRQ 7 to IRQ 1 Vectored Interrupts

Control register bits 7 to 1 (VMEbus IRQ7 to VMEbus IRQ1) determine the handling of up to all seven of the VMEbus interrupt request levels. The VMEbus interrupts are vectored interrupts requiring an interrupt vector from the interrupter during the MZ 8130/MZ 8134's VMEbus interrupt acknowledge cycle. These interrupt vectors point to the interrupt vector table which must have the interrupt handler target address (the address where the interrupt handler starts) for that VMEbus interrupt request level.

3 Only one VMEbus master can handle a specific VMEbus interrupt level.

If a side card is present and is configured to generate a VMEbus interrupt via BIRQ1 or BIRQ2, the VMEbus interrupt handler must not be configured via the control register to respond to either of the side card VMEbus interrupt levels selected by jumper block K6.

2.4.6 Side Card VMEbus BIRQ1* and BIRQ2* Interrupts

A side card can generate two distinct VMEbus interrupts using the active low MXbus signals BIRQ1*, BIRQ2*, MXIACKIN*, VMEIACKOUT*, and MXLIACK*. On-board circuitry translates these signals into the proper VMEbus interrupt signals, routes the interrupt level encoded on A1 to A3 to the corresponding side card address lines, and provides the STATUS/ID to the interrupt handler, completing the interrupt acknowledge cycle.

The side card initiates a VMEbus interrupt by asserting one of the BIRQ*x* signals. These signals are wired directly to the VMEbus IRQ*x* signals via jumper block K6 which selects the IRQ level for each. With the MXbus IACKIN/IACKOUT IACK daisy-chain jumper, K5, removed, the IACKIN* signal from the VMEbus becomes the MXbus signal MXIACKIN*. Whenever this signal is detected, the side card circuitry must determine whether it has any VMEbus interrupts pending, and if so, whether the interrupt level encoded on A1 to A3 matches the level of any pending interrupts. If there is no pending interrupt, or if the level of the current interrupt acknowledge Cycle does not match any pending request, VMEIACKOUT* is asserted to pass the IACK daisy-chain signal to the next position on the VMEbus backplane.

If a match is found, the side card releases the corresponding BIRQ*x* signal and asserts LIACK. When the MZ 8130/MZ 8134 detects LIACK*, the data in a predetermined location in DRAM is placed on the VMEbus in response to the interrupt handler's request for the STATUS/ID. This completes the interrupt acknowledge cycle.

The STATUS/ID is commonly used to derive the address of the interrupt handling routine. The locations corresponding to the levels selected by jumper block K6 should be loaded with the appropriate value BEFORE the assertion of the BIRQ*x* signal so that the interrupt handler can take appropriate action.

Table 2-26 shows the predetermined STATUS/ID location for each VMEbus interrupt level in the MZ 8130 and the MZ 8134.

Table 2-26. Default STATUS/ID Locations for Side Card VMEbus Interrupts

Interrupt Level	MZ 8130	MZ 8134
Level 1	$FFFF3	$3FFFF3
Level 2	$FFFF5	$3FFFF5
Level 3	$FFFF7	$3FFFF7
Level 4	$FFFF9	$3FFFF9
Level 5	$FFFFB	$3FFFFB
Level 6	$FFFFD	$3FFFFD
Level 7	$FFFFF	$3FFFFF

Section 3: Hardware Reference Guide

3.1 Introduction

This section provides a functional description of the MZ 8130/MZ 8134 in the following sections:

3.1 Introduction
3.2 Central Processing Unit
3.3 Clocks
3.4 Memory Devices
3.5 DUART Serial Ports/Timer
3.6 VMEbus Connector (P1)
3.7 MXbus Connector (S1)
3.8 System Controller
3.9 MZ 8130/MZ 8134 Specifications
3.10 MZ 8130 Schematics

A block diagram of the MZ 8130/MZ 8134 is shown in Figure 3-1.

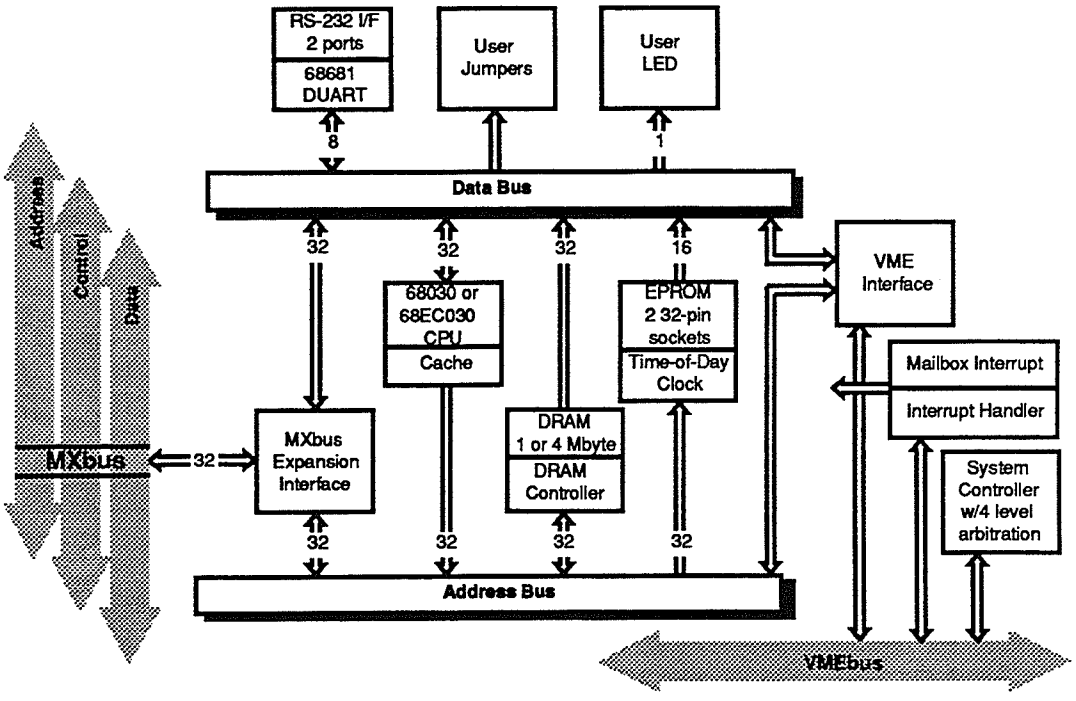


Figure 3-1. MZ 8134 Block Diagram

3.2 Central Processing Unit

The MZ 8130 provides a Motorola MC68030 CPU, while the MZ 8134 provides an MC68EC030 CPU. The major difference between the processors is that the MC68030 has a memory management unit (MMU); the MC68EC030 does not. Both are high-performance 32-bit CPUs, implemented in CMOS technology, and described in the *MC68030 Enhanced 32-Bit Microprocessor User's Manual.* The on-chip 256-byte instruction and 256-byte data caches improve overall performance.

The CPU performs all functions required in a 32-bit microprocessing system, including system address generation, control, memory access, and processing. The MC68030/MC68EC030 has:

- Separate 32-bit address and data buses along with local bus arbitration signals. Data can be read or written over the 32-bit bus in several lengths:
  - Long word (32 bits)
  - Word (16 bits)
  - Byte (8 bits)
- A coprocessor interface
- Sixteen 32-bit general purpose registers
- Two 32-bit supervisor stack pointers
- One user stack pointer
- Ten special purpose control registers
- A 256-byte instruction cache
- A 256-byte data cache
- Pipelined instruction architecture
- Dynamic bus sizing with unaligned transfer capability
- Four Gbytes addressing range with an optional side card
- Seven interrupt levels
- A full instruction set particularly well suited for memory-managed operating systems
- A memory management unit (MMU) (MC68030 only). Most real-time systems do not use the MMU.

3.3 Clocks

The main clock source for the MZ 8130/MZ 8134 is an oscillator running at a CPU speed of 25 MHz. The VMEbus 16 MHz SYSCLK signal, derived from a 16 MHz crystal, is driven by the MZ 1000 system controller when it is enabled. The SCN68681 DUART serial port chip has an internal clock derived from a 3.6864 MHz crystal.

The optional DS1216 SmartWatch is a completely self-contained, backed-up, time-of-day clock. The DS1216:

- Maintains the date in year, month, day, hour, minute, second, and hundredths of a second format
- Automatically corrects for 28-, 29-, 30-, and 31-day months
- Operates in either a 24-hour or 12-hour format with an AM/PM indicator
- Maintains better than ±1 minute per month accuracy at 25° C
- Has a lithium energy cell that maintains the time for more than ten years in the absence of electrical power source

3.4 Memory Devices

The MZ 8130/MZ 8134 contains two types of memory: up to one Mbyte of EPROM, EEPROM, ROM, or PROM memory and one or four Mbytes of dual-access DRAM.

3.4.1 EPROM

The two byte-wide 32-pin JEDEC EPROM sockets on the MZ 8130/MZ 8134 can accommodate 64K, 128K, 256K, 512K, 1M, 2M, or 4M EPROMs as configured by jumper block K3 for up to 1 Mbyte of EPROM. Refer to Section 1.5.3, "EPROM Size Selection," for more information.

3.4.2 Dual-Access DRAM

The dual-access DRAM supports CPU 32-, 16-, and 8-bit accesses and 32-bit unaligned transfers. The DRAM can be accessed by the local CPU and by the MXbus A32:D32 and A24:D16 side card master via the MXbus. The MZ 8130 has one Mbyte of DRAM located at $0 to $FFFFF. The MZ 8134 has four Mbtes of DRAM located at $000000 to $3FFFFF It can also be accessed by the VMEbus at the VMEbus A24/A32 address programmed into the control register. The control register also permits the DRAM to be cacheable to the CPU by controlling the type of VMEbus DRAM accesses allowed.

The DRAM performs VMEbus data transfers to and from memory in compliance with the *VMEbus Specification Revision C.1*, including read-modify-write cycles, address-only cycles, and address pipelining.

3.5 DUART Serial Ports/Timer

The SCN68681 Dual Asynchronous Receiver/Transmitter, DUART features include

- Two independent on-board, full-duplex, asynchronous RJ11 RS-232-C serial ports located on the front panel
- A 3.6864 MHz oscillator
- An internal multifunction16-bit programmable timer that can generate a timer interrupt to the CPU or a custom baud rate
- Independently programmed receiver and transmitter baud rates allowing 18 standard asynchronous baud rates from 50 to 38,400 baud
- Programmable data format with 5 to 8 data bits plus parity, odd, even, none, or forced, 1, 1.5, or 2 stop bits programmable in 1/16-bit increments
- Parity, framing, and overrun error detection.
- A versatile interrupt system that can generate a level 3 vectored CPU interrupt request when a character is received, the transmit buffer becomes empty, or the 16-bit timer reaches its count according to the programming of the SCN68681.

Refer to Section 2.2.5 for information about programming the SCN68681 for transmission or to the *SCN68681 Dual Asynchronous Receiver/Transmitter (DUART) Product Specification* from Signetics for more detail.

3.5.1 RS-232-C Interface Specifications

The MZ 8130/MZ 8134's RS-232-C interface circuits meet EIA standard specifications No. RS-232-C. All RS-232-C signals generated by the MZ 8130/MZ 8134 are approximately 10 volts when measured across a 3K load. The RS-232-C receiving circuits accept signals in the 3 to 25 volt range. The input impedances of all MZ 8130/MZ 8134 RS-232-C circuits accept signals with DC resistances of 4.7K. Consult the EIA RS-232-C specifications for more detailed information.

Table 3-1 describes the signal states used by the MZ 8130/MZ 8134. The negative threshold voltage is less (more negative) than -3 volts with respect to signal ground. The positive threshold voltage is greater than +3 volts with respect to signal ground.

Table 3-1. Signal States

RS-232-C Signal States	Negative	Positive
Binary State	One	Zero
Signal Condition	Mark	Space
Control and Timing Functions	Off	On

3.5.2 Serial Port RS-232-C DCE Specifications

The MZ 8130/MZ 8134 serial ports are DCE ports, generally used with RS-232-C terminals. Table 3-2 lists the RS-232-C interface pins present on the serial ports RJ11 connectors located on the front panel. All unassigned pins are unconnected. Pin 1 of the RJ11 connector is the bottom-most pin when the MZ 8130/MZ 8134 is viewed with the RESET switch at the top.

Table 3-2. DCE RS-232-C Pin Assignment

Pin	Direction	Designation	Function
1	Receives	CTS	Clear To Send
2	Receives	RD	Receive Data
3	Supplies	GND	Ground
4	Supplies	GND	Ground
5	Supplies	TD	Transmit Data
6	Supplies	RTS	Request To Send

The signals are described below.

Clear To Send — Pin 1
This signal is asserted to the external device to indicate that the MZ 8130/MZ 8134 has data to transmit via the TD signal (pin 5). CTS and RTS can operate in handshaking mode.

Received Data — Pin 2
The MZ 8130/MZ 8134 receives data on this pin. When no data is being sent, the external device should put this signal in the marking.

Ground — Pins 3 and 4
These leads are connected to signal ground of the MZ 8130/MZ 8134. It establishes the common ground reference for all other interface signals.

Transmit Data — Pin 5
The MZ 8130/MZ 8134 provides data on this pin. A positive signal is a space (binary 0) and a negative signal is a mark (binary 1). The MZ 8130/MZ 8134 holds this signal in the marking state when no data is being transmitted and between characters or words.

Request To Send — Pin 6
This signal received indicates that the external device has data ready to transmit.

3.6 VMEbus Connector (P1)

The VMEbus interface on the MZ 8130/MZ 8134 is the P1 connector, a 96-pin connector organized in the three rows. Table 3-3 identifies the pin assignments of this physical interface between the MZ 8130/MZ 8134 and the VMEbus. Section 1.2, "VMEbus Interface Capability," and Section 2.1.4, "Configuring the VMEbus Interface," describe the VMEbus interface.

Table 3-3. VMEbus P1 Pin Assignments (* active low)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D0	BBSY*	D8
2	D1	BCLR*	D9
3	D2	ACFAIL*	D10
4	D3	BG0IN*	D11
5	D4	BG0OUT*	D12
6	D5	BG1IN*	D13
7	D6	BG1OUT*	D14
8	D7	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK	A17
22	IACKOUT*	SERDAT*	A16
23	AM4	GND	A15
24	A7	IRQ7*	A14
25	A6	IRQ6*	A13
26	A5	IRQ5*	A12
27	A4	IRQ4*	A11
28	A3	IRQ3*	A10
29	A2	IRQ2*	A9
30	A1	IRQ1*	A8
31	-12V+5	STDBY	+12V
32	+5V	+5V	+5V

The VMEbus interface of a side card is P2, a 96-pin connector organized in three rows labeled A, B, and C, is the physical interface between a side card (if installed) and the VMEbus. Table 3-4 identifies the pin assignments.

Table 3-4. Side Card's VMEbus P2 Pin Assignments (* active low)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	User Defined	+5V	User Defined
2	User Defined	GND	User Defined
3	User Defined	Reserved	User Defined
4	User Defined	A24	User Defined
5	User Defined	A25	User Defined
6	User Defined	A26	User Defined
7	User Defined	A27	User Defined
8	User Defined	A28	User Defined
9	User Defined	A29	User Defined
10	User Defined	A30	User Defined
11	User Defined	A31	User Defined
12	User Defined	GND	User Defined
13	User Defined	+5V	User Defined
14	User Defined	D16	User Defined
15	User Defined	D17	User Defined
16	User Defined	D18	User Defined
17	User Defined	D19	User Defined
18	User Defined	D20	User Defined
19	User Defined	D21	User Defined
20	User Defined	D22	User Defined
21	User Defined	D23	User Defined
22	User Defined	GND	User Defined
23	User Defined	D24	User Defined
24	User Defined	D25	User Defined
25	User Defined	D26	User Defined
26	User Defined	D27	User Defined
27	User Defined	D28	User Defined
28	User Defined	D29	User Defined
29	User Defined	D30	User Defined
30	User Defined	D31	User Defined
31	User Defined	GND	User Defined
32	User Defined	+5V	User Defined

3.7 MXbus Connector (S1)

The MXbus interface on the MZ 8130/MZ 8134 is the optional S1 side connector, a 128-pin connector organized in the four rows labeled A, B, C, and D in Table 3-5 .

Table 3-5. S1 Pin Assignments (* active low)

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic	Row D Signal Mnemonic
1	LA11	LA10	LBERR*	LA24
2	LA16	LA14	VCC	LA25
3	LA12	LA13	SYSCLK	LA26
4	LA15	LA17	Reserved	LA27
5	LA19	LA18	HALFCLK	LA28
6	LA7	LA20	BR*	LA29
7	LA9	LA1	BGACK*	LA30
8	LA8	LA3	DSACK1*	LA31
9	LA4	LA2	AS*	LD0
10	LA21	LA23	LA5	LD1
11	LA22	LA6	GND	LD2
12	-12V	BG*	LD16	LD3
13	LD17	LD18	LD26	LD4
14	LD19	LD20	LD25	LD5
15	RESET	LD21	GND	LD6
16	+12V	LD22	LD23	LD7
17	FC1*	FC2*	FC0*	LD8
18	COMPAT32*	LD24	GND	LD9
19	R/W*	LDS* (DS*)	VAEN*	LD10
20	Reserved	UDS* (A0)	GND	LD11
21	Reserved	VCC	LD31	LD12
22	A32MATCH*	LD28	GND	LD13
23	LD30	LD27	LD29	LD14
24	IPL2*	BAV*	MXSENSE*	LD15
25	IPL0*	BIRQ2*	BIRQ1*	VDIR*
26	VCC	VDS0*	Reserved	VDEN*
27	IPL1*	VDS1*	BAS*	Cacheable* (MZ 8134)
28	LIACK*	BIACK*	GND	CLK
29	BA1	LWORD*	GND	DSACK0*
30	BA2	IACKOUT*	S1IRQ2*	RMC*
31	BA3	IACKIN*	S1IRQ1*	SIZE0
32	Reserved	SPACE*	AVEC*	SIZE1

Table 3-6 describes the A24:D16 MXbus loading and driving specifications (* active low).

Table 3-6. A24:D16 MXbus Loading/Driving Specifications

Signal Name	Direction From MXbus	MXbus Must Drive IOH / IOL	MXbus Max. Load IIH / IIL
LA1	I/O	135 µA / 735 µA	265 µA / 2.4 mA
LA2-LA3	I/O	150 µA / 800 µA	250 µA / 2.4 mA
LA4	I/O	130 µA / 710 µA	280 µA / 2.5 mA
LA5-LA14	I/O	120 µA / 700 µA	290 µA / 2.5 mA
LA15	I/O	120 µA / 1.0 mA	280 µA / 2.2 mA
LA16	I/O	170 µA / 1.6 mA	230 µA / 1.6 mA
LA17-LA18	I/O	170 µA / 1.2 mA	230 µA / 2.6 mA
LA19	I/O	120 µA / 1.6 mA	230 µA / 1.6 mA
LA20	I/O	125 µA / 575 µA	275 µA / 2.6 mA
LA21-LA22	I/O	100 µA / 320 µA	300 µA / 2.8 mA
LA23	I/O	100 µA / 660 µA	300 µA / 2.4 mA
FC0-FC2	I/O	90 µA / 1.6 mA	330 µA / 1.7 mA
AM5	——	Not used	Not used
LWORD*	IN	——	14.9 mA / 63.4 mA
LONG*	——	Not used	Not used
LD0-LD15	I/O	120 µA / 730 µA	290 µA / 1.3 mA
AS*	I/O	140 µA / 2.1 mA	3.1 mA / 22.0 mA
LDS*-UDS*	I/O	25 µA / 250 µA	3.2 mA / 24.0 mA
R/W*	I/O	130 µA / 3.2 mA	275 µA / 2.1 mA
DTACK*	I/O	100 µA / 5.1 mA	3.2 mA / 19.0 mA
BERR*	I/O	28 µA / 5.3 mA	3.2 mA / 19.0 mA
BAS*	IN	——	900 µA / 18.8 mA
VDS0-1*	IN	——	355 µA / 7.3 mA
BIRQ1-2*	OUT	O.C. / 48.0 mA	——
BIACK*	IN	——	14.9 mA / 63.2 mA
BA1-3	IN	——	15.0 mA / 64.0 mA
IACKIN*	IN	——	400 µA / 8.0 mA
IACKOUT*	OUT	400 µA / 8.0 mA	——
S1IRQ1-2*	OUT	25 µA / 835 µA	——
IPL0-2	IN	——	780 µA / 16.0 mA
LIACK*	OUT	45 µA / 1.6 mA	——
AVEC*	IN	——	3.2 mA / 16.0 mA
VMA*	——	Not used	Not used
E	——	Not used	Not used
BR*	OUT	25 µA / 5.3 mA	——
BG*	IN	——	3.2 mA / 24.0 mA
BGACK*	OUT	100 µA / 6.0 mA	——
BAV*	IN	——	3.0 mA / 21.8 mA
RESET*	IN	——	4.6 mA / 46..6 mA
HALT*	I/O	10 µA / 5.3 mA	3.2 mA / 19.0 mA
HALFCLK	IN	——	980 µA / 19.4 mA
SYSCLK	IN	——	50 µA / 600 µA
SPACE*	IN	70 µA / 800 µA	——

Table 3-7 describes the A24:D16 MXbus signals (* active low).

Table 3-7. A24:D16 MXbus Signal Description

Signal Name	Signal Description
LA1-LA23	Unbuffered CPU A1 - A23 Address Lines
FC0-FC2	Unbuffered CPU Function Code Signals
LWORD*	Buffered VMEbus LWORD* Signal
LD0-LD16	Unbuffered CPU D0 - D16 Data Lines
AS*	Buffered CPU Address Strobe
UDS*-LDS*	Encoded/Decoded CPU Data Strobes
R/W*	Unbuffered CPU Read/Write Signal
DTACK*	Unbuffered DSACK1* Data Transfer Acknowledge Signal
BERR*	MXbus Bus Error Signal (unbufferd to CPU)
BAS*	Buffered VMEbus Address Strobe Signal AS*
VDS0-1*	Buffered VMEbus Data Strobes DS0* and DS1*
BIRQ1-2*	Unbuffered VMEbus Interrupt Request Signals
BIACK*	Buffered VMEbus Interrupt Acknowledge Signal IACK*
BA1-3	Buffered VMEbus Address Lines A1-A3.
IACKIN*	Buffered VMEbus Interrupt Acknowledge
IACKOUT*	Unbuffered VMEbus Interrupt Acknowledge Out Daisy Chain Signal
S1IRQ1*	CPU Autovector Interrupt Level 6 Request Signal
S1IRQ2*	CPU Autovector Interrupt Level 4 Request Signal
IPL0-2*	Unbuffered CPU Encoded Interrupt Request Signals
LIACK*	VMEbus Interrupt Acknowledge Signal that must be asserted to the MZ 8130/MZ 8134 to allow the VMEbus interrupt vector to be fetched from the MZ 8130/MZ 8134's DRAM and placed on the VMEbus.
VPA*,VMA*,E	Are not supported by the CPUs (used only by the MZ 8115 and MZ 8120)
AVEC*	Unbuffered CPU Autovector Interrupt Signal
BR*	Buffered CPUbus/MXbus Request Signal
BG*	Buffered CPUbus/MXbus Grant Signal
BGACK*	Buffered CPUbus/MXbus Bus Grant Acknowledge Signal
BAV*	MZ 8130/MZ 8134 is the VMEbus master
RESET*	Buffered VMEbus RESET* or MZ 8130/MZ 8134 RESET Switch
HALT*	Unbuffered MXbus HALT* Signal
HALFCLK	Buffered CPU Clock Divided by 2
SYSCLK	Unbuffered VMEbus SYSCLK Signal
SPACE*	Unbuffered CPU Space Cycle Signal

Table 3-8 describes the A32:D32 MXbus loading and driving specifications (* active low).

Table 3-8. A32:D32 MXbus Loading/Driving Specifications

Signal Name	Direction From MXbus	MXbus Must Drive IOH / IOL	MXbus Max. Load IIH / IIL
LA24-LA31	I/O	30 µA / 600 µA	275 µA / 2.6 mA
SIZE0-1	I/O	75 µA / 600 µA	325 µA / 2.6 mA
RMC*	I/O	75 µA / 1.5 mA	325 µA / 3.1 mA
LD16-LD23	I/O	130 µA / 740 µA	280 µA / 1.3 mA
LD24-LD31	I/O	210 µA / 910 µA	200 µA / 1.2 mA
DSACK0-1*	I/O	100 µA / 5.1 mA	3.2 mA / 19.0 mA
VDIR	IN	——	3.0 mA / 19.2 mA
VDEN*	IN	——	3.1 mA / 21.6 mA
VAEN*	IN	——	940 µA / 19.9 mA
A32MATCH*	OUT	25 µA / 1.0 mA	——
COMPAT32*	OUT	25 µA / 1.0 mA	——
MXSENSE*	OUT	70 µA / 800µA	——
CLK	IN	——	1.0 mA / 1.0 mA

Table 3-9 describes the A32:D32 MXbus signals (* active low).

Table 3-9. A32:D32 MXbus Signal Description

Signal Name	Signal Description
LA23-LA31	Unbuffered CPU Address Lines
SIZE0-1	Unbuffered CPU Data Size Signals
RMC*	Unbuffered CPU Read-Modify-Cycle Signal
LD16-LD31	Unbuffered CPU Data Lines
DSACK0-1*	Unbuffered CPU Data Size Acknowledge Signal
VDIR	MZ 8130/MZ 8134 VMEbus D16-D31 Data Buffer Direction Signal
VDEN*	MZ 8130/MZ 8134 VMEbus D16-D31 Data Buffer Enable Signal
VAEN*	MZ 8130/MZ 8134 VMEbus A24-A31 Address Buffer Enable Signal
A32MATCH*	VMEbus A32 MZ 8130/MZ 8134 DRAM Address Decoding Match Signal
COMPAT32*	MXbus A24 / A32 Enable/Disable Signal
MXSENSE*	MZ 8130/MZ 8134 Side Card Status Register Signal (User defined)
CLK	Unbuffered CPU Clock

Table 3-10 describes the MXbus expansion header loading and driving specifications (* active low).

Table 3-10. MXbus Expansion Header Loading/Driving Specifications

Signal Name	Direction From MXbus	MXbus Must Drive IOH / IOL	MXbus Max. Load IIH / IIL
ECS*	IN	——	400 µA / 2.0 mA
LA0	IN	——	325 µA / 2.9 mA
LAS*	IN	——	210 µA / 1.9 mA
LDS*	IN	——	250 µA / 3.2 m
TRI*	OUT	20 µA / 250 µA	——
VACC*	IN	——	950 µA / 19.4 mA

Table 3-11 describes the MXbus expansion header signals (* active low).

Table 3-11. MXbus Expansion Header Signal Description

Signal Name	Signal Description
ECS*	Unbuffered MC68030/MC68EC030 Early Chip Signal
LA0	Unbuffered MC68030/MC68EC030 A0 Address Line
LAS*	Unbuffered MC68030/MC68EC030 Address Strobe
LDS*	Unbuffered MC68030/MC68EC030 Data Strobe
TRI*	Tristates AS*, UDS-LDS*, and LA0 MXbus Signals to allow direct assertion via the K1 expansion header.
VACC*	VMEbus Access to MZ 8130/MZ 8134's MC68030/MC68EC030 CPUBus/MXbus

Section 1.5.1, "K1: MXbus Expansion Header," describes the MXbux Expansion Header.

3.8 System Controller

VMEbus system controller functions are provided by the MZ 1000 gate array. The MZ 1000 system controller provides:

- Four-level VMEbus prioritized or round robin arbitration
- 16 MHz SYSCLK
- 400 ms SYSRESET
- A bus grant daisy-chain driver
- An IACK driver
- A VMEbus time-out watchdog timer (asserts BERR after the time-out period of 65-128µs)
- Error detection and correction on the arbitration circuit

To use the MZ 8130/MZ 8134 as the system controller, you must:

- Install it in slot 1 of the VMEbus backplane and
- Enable the system controller functions jumper block K9. It can be disabled if another CPU card with system controller capabilities is enabled.

The MZ 1000's bus arbitration circuit arbitrates VMEbus mastership between VMEbus masters in one of two modes as selected by SYSARB (control register bit 18):

- Priority (PRI) or
- Round robin (RRS)

The arbitration circuit uses a synchronous state machine that operates with a 16 MHz clock. The bus request inputs, BR0 to BR3, and VMEbus busy, BBSY, are clocked on a rising edge. The bus grant outputs, BG0 to BG3, are then clocked on the next rising edge. Clocking both BR*x* inputs and the BG*x* outputs alleviates the possibility of metastability and glitch conditions on the BG outputs.

The MZ 1000 issues a VMEbus grant signal which propagates down the bus grant daisy chain. The VMEbus master which accepts the grant asserts BBSY to inform the MZ 1000 that it is the current bus master. The MZ 1000 monitors BBSY and does not arbitrate the bus until it is negated and a bus request is received.

To perform bus arbitration, the MZ 1000 requires two to three 16 MHz clock edges. The number of edges is dependent upon the state of the MZ 1000 when BBSY is negated. At 16 MHz, this results in a bus request to a bus grant time period of 100 to 200 ns. Typically, arbitration takes 150 ns.

When the MZ 1000 is in priority mode (SYSARB is set), each bus master is assigned a bus priority level based upon the bus request level on which it resides. Bus request level 3, BR3, has the highest priority, while level BR0 has the lowest. When two or more masters request the VMEbus between the same rising edges of the 16 MHz clock, the MZ 1000 grants the bus to the master residing on the highest priority bus request level. When a low priority bus master has bus mastership and a higher priority master requests the bus, the MZ 1000 issues a BCLR interrupt to remove the lower priority bus master.

In round robin mode (SYSARB is not set), each bus request level is given equal priority. The VMEbus masters are granted the VMEbus on a rotating basis, BR3-BR2-BR1-BR0-BR3-etc. When a bus master releases the bus by negating BBSY, the next active bus request level in the chain is granted the bus by the arbiter. All bus masters are then granted equal access to the VMEbus. In this mode, the MZ 1000 asserts a BCLR interrupt when another master requests the bus. The ROR bus release mechanism is commonly used with round robin arbitration. A ROR master releases the bus when it sees any bus request and doesn't necessarily need to monitor BCLR.

The MZ 1000 has a BERR watchdog timer which notifies a bus master when the address it has placed on the VMEbus is not responded to within 64 to 128 µs. The MZ 1000 monitors the VMEbus data strobes, DS0 and DS1. If either data strobe stays asserted for more than 64 to 128 µs, the MZ 1000 asserts BERR. A common use of the watchdog timer is performed by the operating system to determine the amount of memory available when the system boots. During debugging, you may want to disable BERR and allow the bus to hang so that the address and control signals can be examined. The BERR signal is disabled by removing the jumper on jumper block K9.

To reduce the possibility of a VMEbus system crash due to noise, spurious bus requests, or errors in the bus grant daisy chain, the MZ 1000 incorporates error detection and correction on the arbitration circuit. The MZ 1000 monitors BBSY and both data strobes. If neither are asserted within 16 to 32 µs after a bus grant is asserted, the MZ 1000 rearbitrates the bus.

Noise in electronic systems is unavoidable and never can be eliminated completely. TTL buses in general are noisy, and the VMEbus is no exception. To help alleviate these problems, the MZ 1000 incorporates three ground pins: Schmidt triggers on BBSY and BR0 to BR3 and a digital filter triggers on the glitch-prone BBSY signal. The digital filter on BBSY filters out any erroneous BBSY glitches which last less than 25 ns. The MZ 8130/MZ 8134 also has an inductor on its BBSY driver to slow the rise time of BBSY and help eliminate glitches when other system controllers are used.

In addition to the MZ 1000's system controller functions, the MZ 8130/MZ 8134 has a IACK driver which ensures that the VMEbus data strobes have been valid on the bus for 40 ns before the IACKIN/IACKOUT daisy chain is asserted. This allows much better control of the interrupt acknowledge cycle and is required by *VMEbus Specifications, Revision C.1*.

When the system controller functions on the MZ 8130/MZ 8134 are enabled, the front panel RESET switch sends a 400 ms VMEbus RESET signal to all boards on the VMEbus backplane. The MZ 1000 also asserts a 400 ms reset pulse upon power-up.

3.9 MZ 8130/MZ 8134 Specifications

Processor:	MC68030 or MC68EC030 CPU.
DRAM:	1 Mbyte (MZ 8130) or 4 Mbyte (MZ 8134) of dual access DRAM with hardware refresh and software programmable VMEbus address.
ROM/EPROM:	16 Kbytes to 1 MByte (28 or 32 pins).
Serial ports/timer:	SCN68681 DUART dual RS-232-C ports, 50 to 38.4K baud; RS-232-C signals TD, RD, CTS, and RTS; 16-bit timer interrupt.
Time-of-day clock:	DS1216 SmartWatch backed-up time-of-day clock (optional).

VMEbus data transfer modes

Without VMEbus A32:D32 side card:	A16:A24:D16 master/slave
With VMEbus A32:D32 side card:	A16:A24:A32:D16:D32 master/slave.

MXbus data transfer modes

A24:D16 side card:	A24:D16 master/slave.
A32:D32 side card:	A32:D32 master/slave.

VMEbus system controller:	On-board MZ 1000 gate array, four level VMEbus arbiter, IACK driver.
VMEbus interrupt handler:	Up to all seven VMEbus levels, BCLR, and ACFAIL.
MXbus interrupts:	Two MXbus autovectored interrupts.
Mailbox interrupt:	Software programmable, VMEbus-addressed CPU autovectored interrupt.
Indicators (LEDs):	Reset, CPU status, and one user-programmable LED.
Software support:	Ported to several popular operating systems. Contact Mizar Technical Support for details.
Operating conditions:	Temperature: 0° to 55° C.
Humidity:	10 - 90% noncondensing.
Power consumption:	3.5 A typical, 5.3 A maximum.
Physical size:	3U 100mm x 160mm Eurocard.

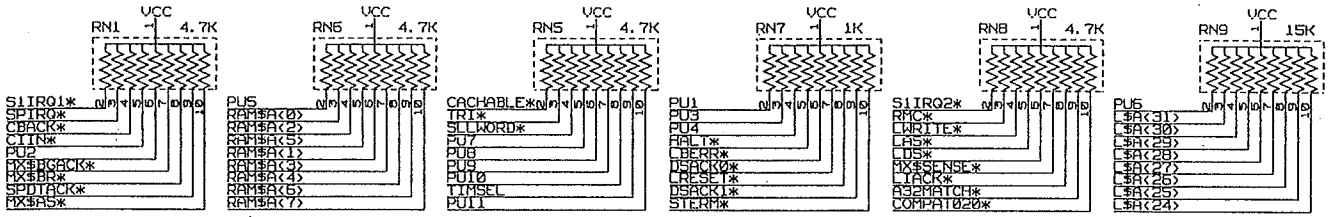
3.10 MZ 8134 Schematics

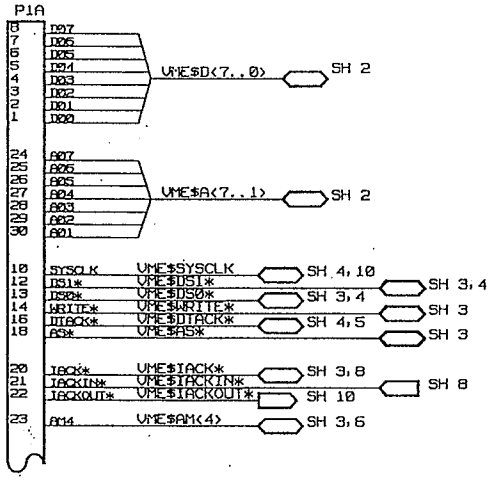
The following pages contain the schematics of the MZ 8134. The main difference between the MZ 8130 and the MZ 8134 is the size of the DRAM.

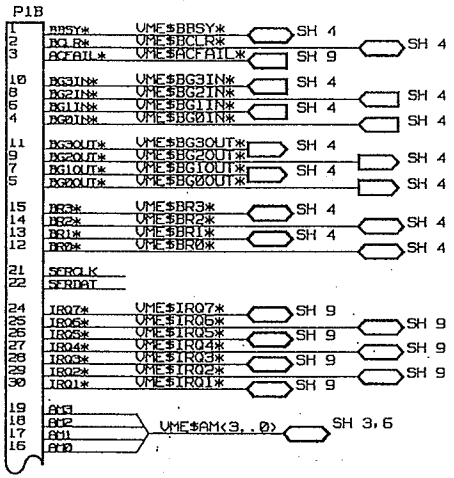
ALL INFORMATION DISCLOSED BY THIS DOCUMENT IS CONSIDERED PROPRIETARY BY MIZAR INCORPORATED. ALL DESIGN, MANUFACTURE, USE, REPRODUCTION AND SALE RIGHTS ARE RESERVED BY MIZAR INCORPORATED.

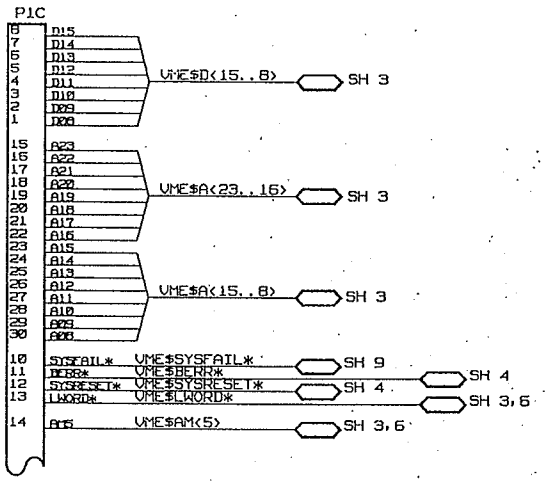
SCHEMATIC REVISIONS

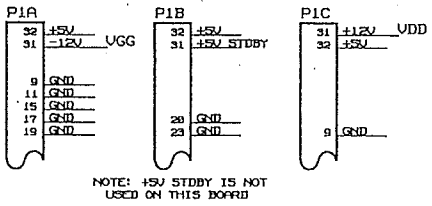
REV	DESCRIPTION	APPD	DATE



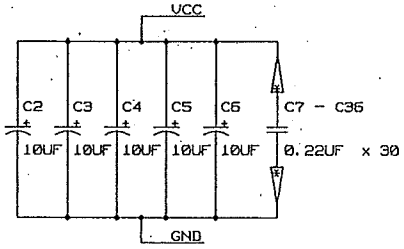








NOTE: +5V STDBY IS NOT USED ON THIS BOARD



VMEBUS CONNECTORS	PAGE 1	SERIAL PORT IACK DRVR DSACK/MXBUS CNTRL	PAGE 8
VME DATA BUFFERS	PAGE 2	INTERRUPTS, STATUS & CONTROL REGISTERS	PAGE 9
VME ADDRESS & STROBES	PAGE 3	MXBUS INTERFACE	PAGE 10
VMEBUS SYSCON, BUS REQESTER	PAGE 4		
DRAM CONTROL, CPU BUS ARBITRATION	PAGE 5		
DRAM MEM, BUFFER, BERR/VME DRAM PALS	PAGE 6		
CPU, EPROM, AND ADDRESS DECODING	PAGE 7		

DRAWING TITLE=8134 ABBREV=8134

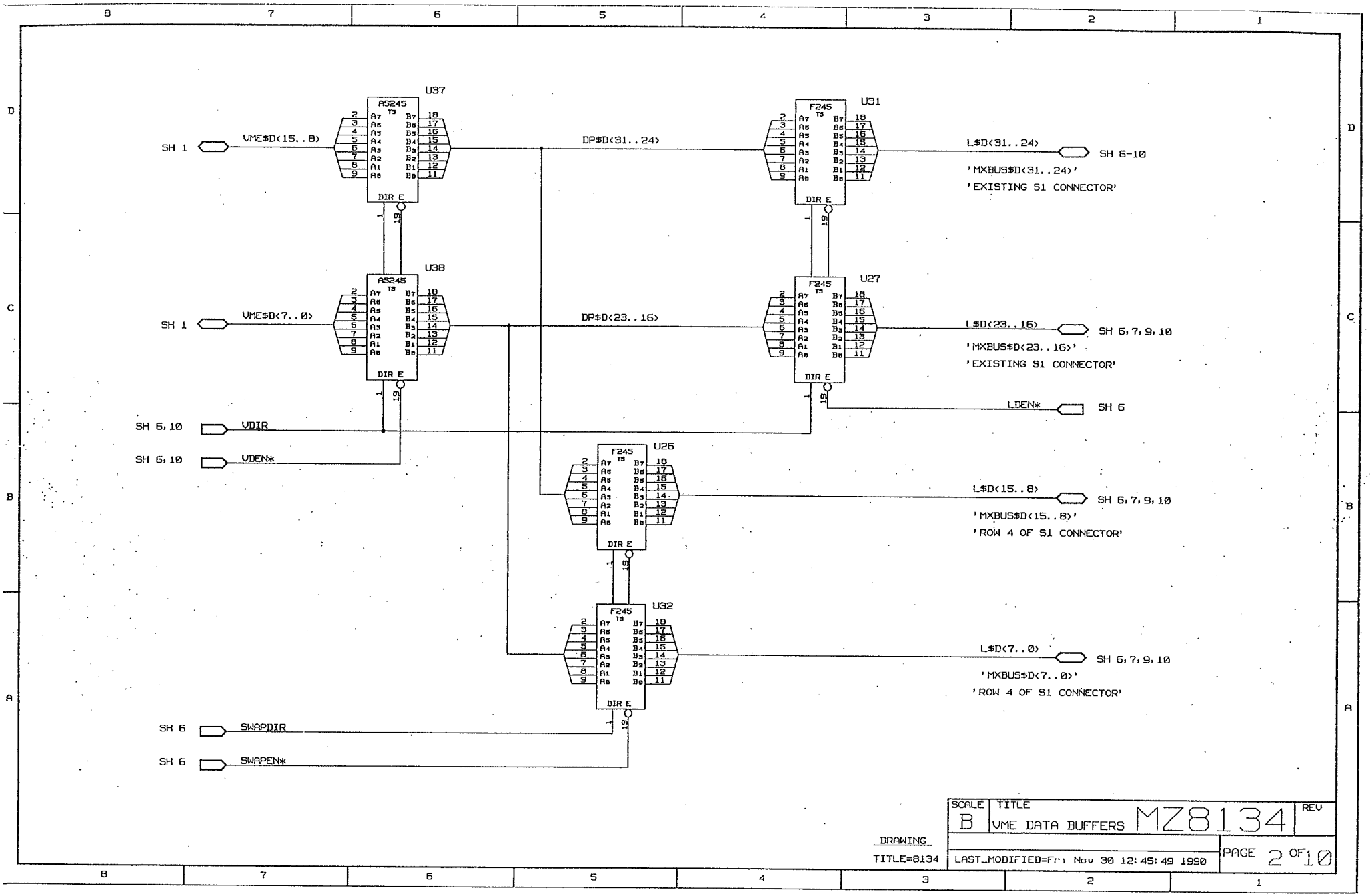
PROJECT ENGINEER	JMF	DATE	11/30/90
DRAWN BY			
CHECKED BY			
APPROVED BY			

MIZAR

SCALE	TITLE	SCHEM. REV
B	MZ8134	

MC68030 CPU MODULE

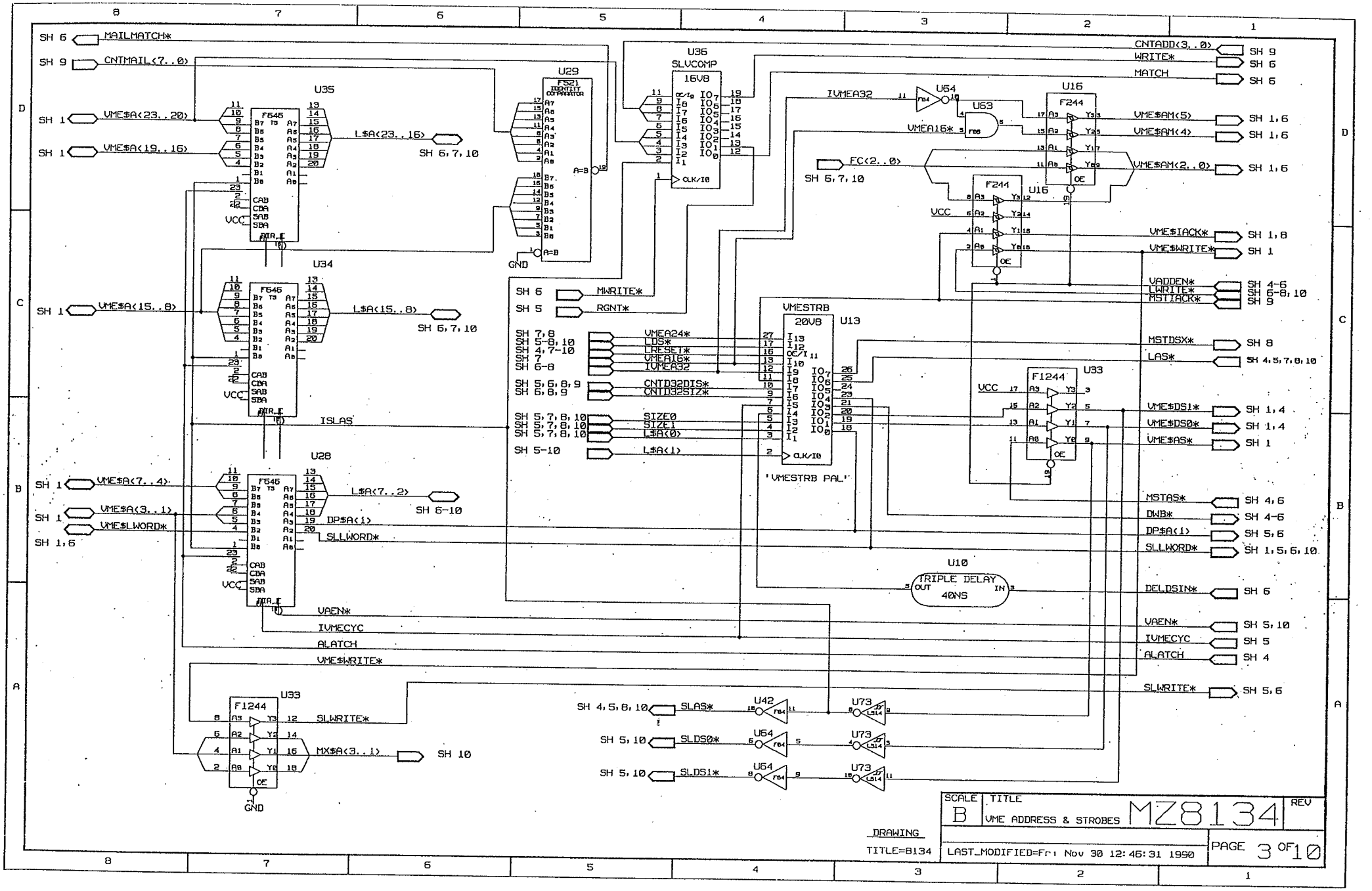
Thu Jan 10 10:08:40 1991

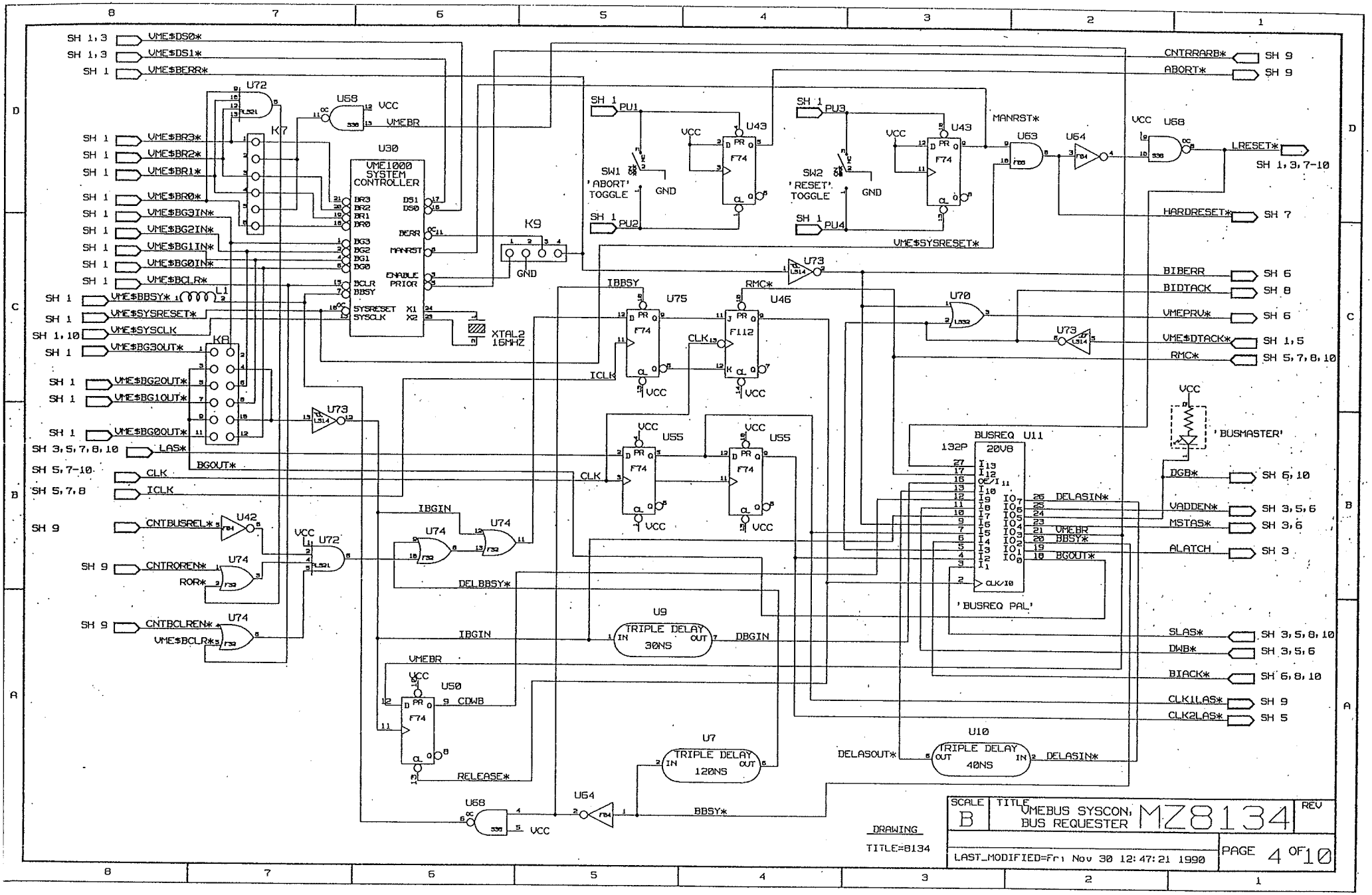


SCALE	TITLE	REV
B	VME DATA BUFFERS MZ8134	

DRAWING
TITLE=8134

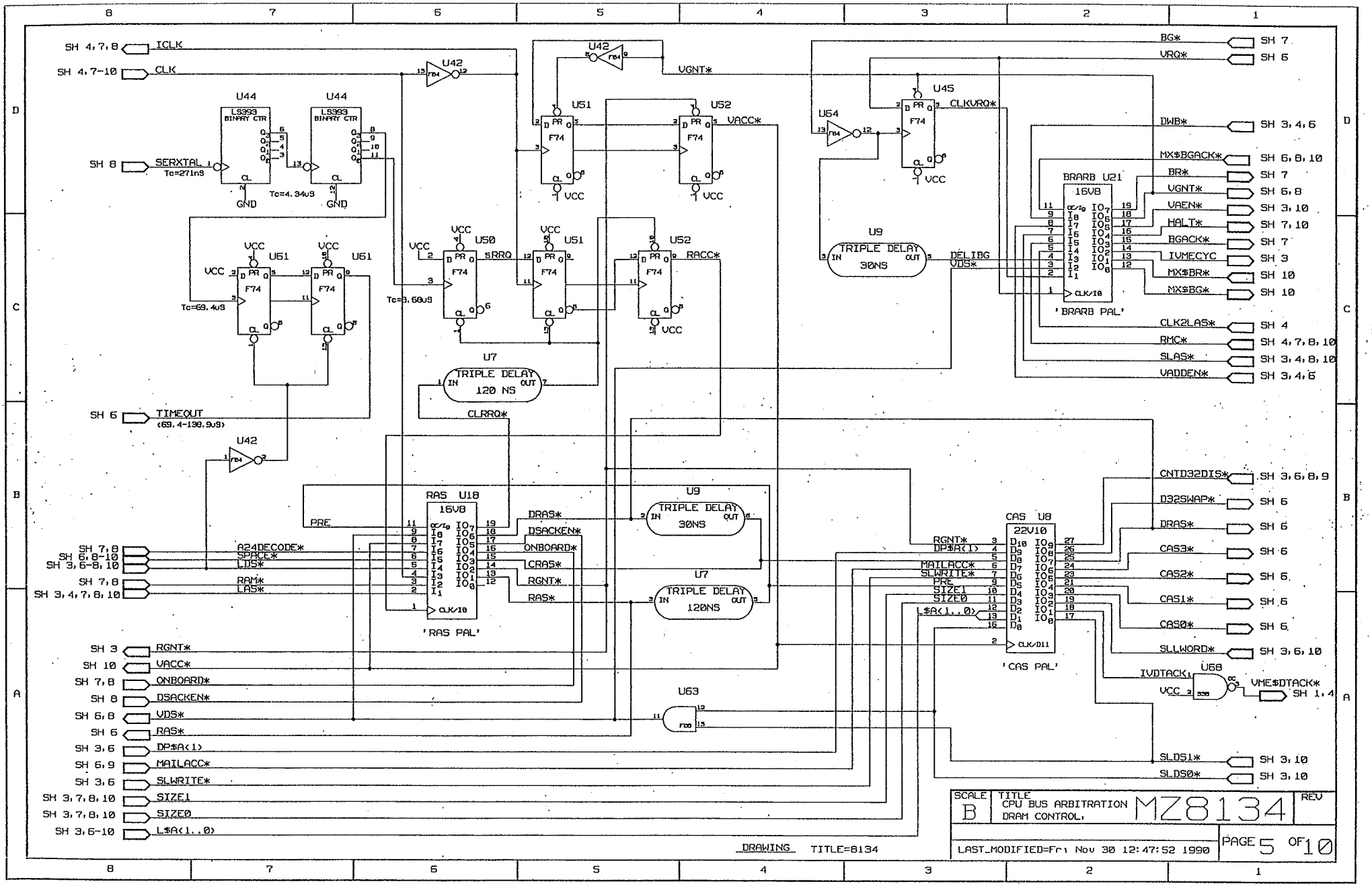
LAST_MODIFIED=Fri Nov 30 12:45:49 1990

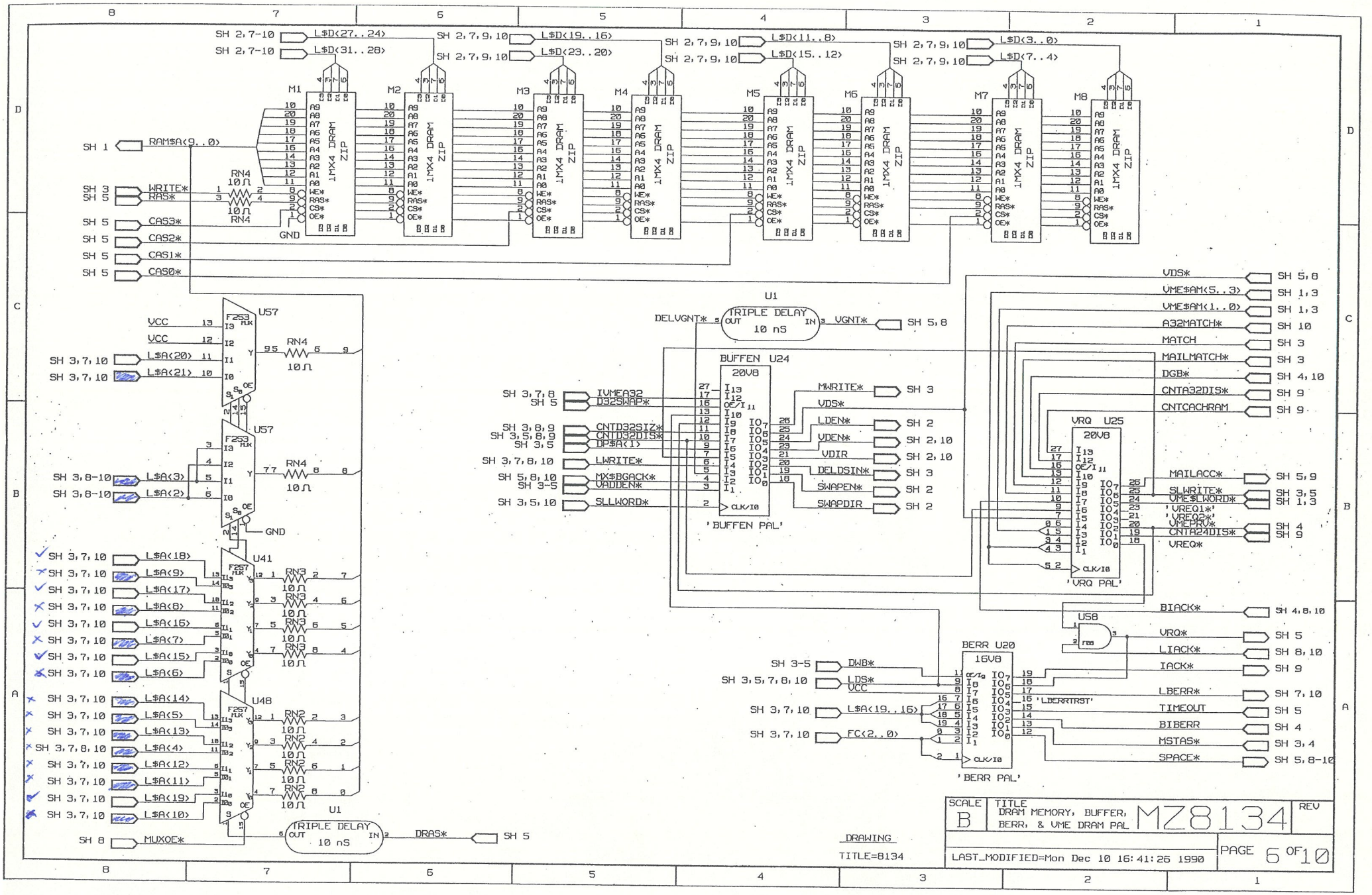

SH 6 MAILMATCH*
SH 9 CNTMAIL<7..0>
SH 1 VME$A<23..20>
SH 1 VME$A<19..16>
SH 1 VME$A<15..8>
SH 1 VME$A<7..4>
SH 1 VME$A<3..1>
SH 1,6 VME$LWORD*
U35
U34
U28
F645
L$A<23..16>
L$A<15..8>
L$A<7..2>
SH 6,7,10
SH 6-10
ISLAS
DP$A<1>
SLLWORD*
VAEN*
IVMECYC
ALATCH
VME$WRITE*
U29
FS21
IDENTITY COMPARATOR
A=B
GND
U36
SLVCOMP
16V8
CLK/I0
SH 6 MWRITE*
SH 5 RGNT*
SH 7,8 VMEA24*
SH 5-8,10 LDS*
SH 4,7-10 LRESET*
SH 7 VMEA16*
SH 6-8 IVMEA32
SH 5,6,8,9 CNTD32DIS*
SH 6,8,9 CNTD32SIZ*
SH 5,7,8,10 SIZE0
SH 5,7,8,10 SIZE1
SH 5,7,8,10 L$A<0>
SH 5-10 L$A<1>
VMESTRB
20V8
U13
'VMESTRB PAL'
IVMEA32
VMEA16*
U64
U53
U16
F244
FC<2..0>
SH 5,7,10
VCC
VME$AM<5> SH 1,6
VME$AM<4> SH 1,6
VME$AM<2..0> SH 1,6
VME$IACK* SH 1,8
VME$WRITE* SH 1
VADDEN* SH 4-6
LWRITE* SH 6-8,10
MSTIACK* SH 9
MSTDSX* SH 8
LAS* SH 4,5,7,8,10
U33
F1244
VME$DS1* SH 1,4
VME$DS0* SH 1,4
VME$AS* SH 1
MSTAS* SH 4,6
DWB* SH 4-6
DP$A<1> SH 5,6
SLLWORD* SH 1,5,6,10
U10
TRIPLE DELAY
OUT
IN
40NS
DELDSIN* SH 6
VAEN* SH 5,10
IVMECYC SH 5
ALATCH SH 4
SLWRITE* SH 5,6
SH 4,5,8,10 SLAS*
SH 5,10 SLDS0*
SH 5,10 SLDS1*
U42
U73
LS14
MX$A<3..1> SH 10
CNTADD<3..0> SH 9
WRITE* SH 6
MATCH SH 6
SCALE B
TITLE VME ADDRESS & STROBES
MZ8134
REV
DRAWING TITLE=8134
LAST_MODIFIED=Fri Nov 30 12:46:31 1990
PAGE 3 OF 10



SCALE	TITLE	REV
B	VMEBUS SYSCON, BUS REQUESTER MZ8134	
	LAST_MODIFIED=Fri Nov 30 12:47:21 1990	PAGE 4 OF 10

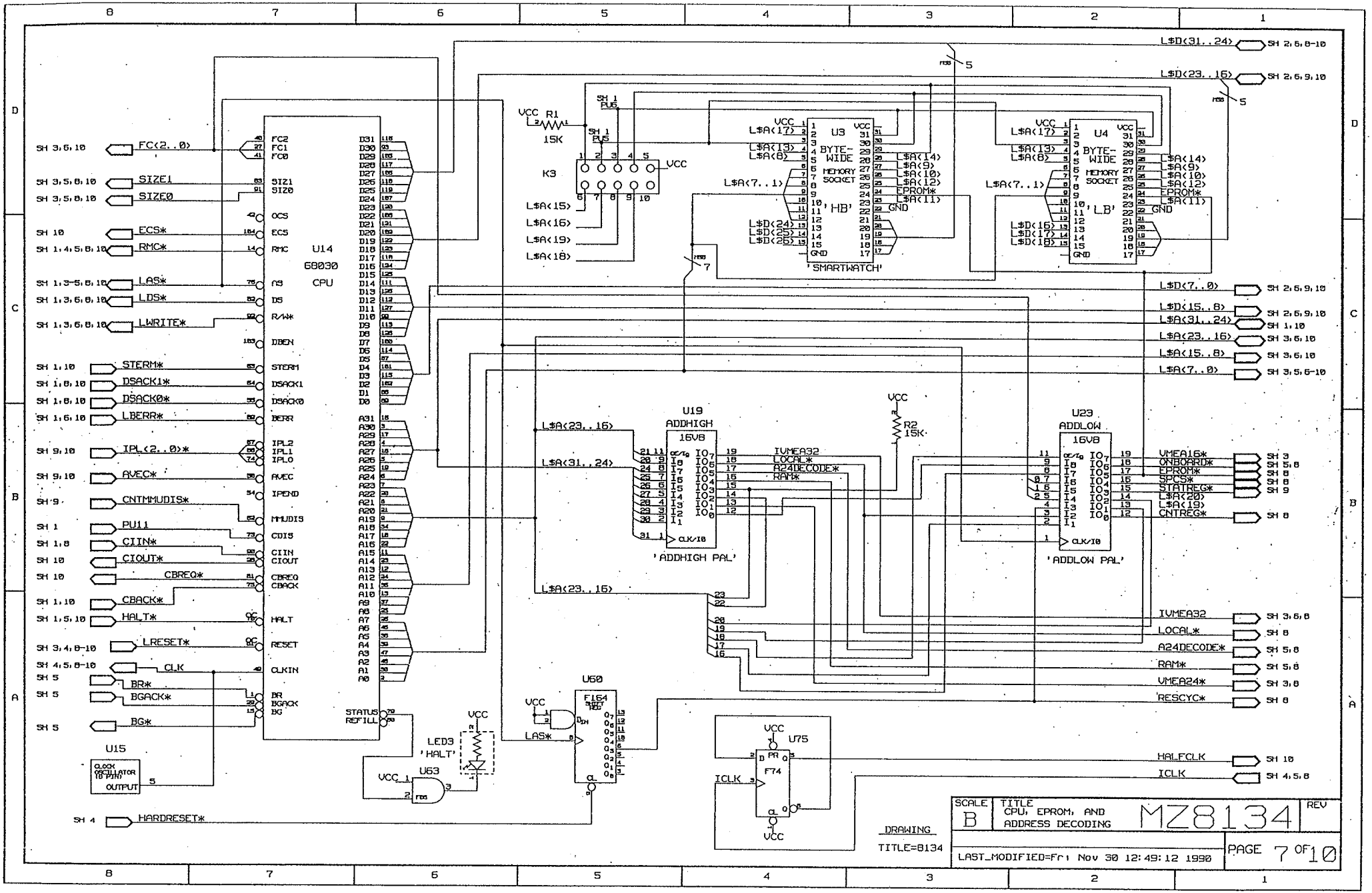
DRAWING
TITLE=8134

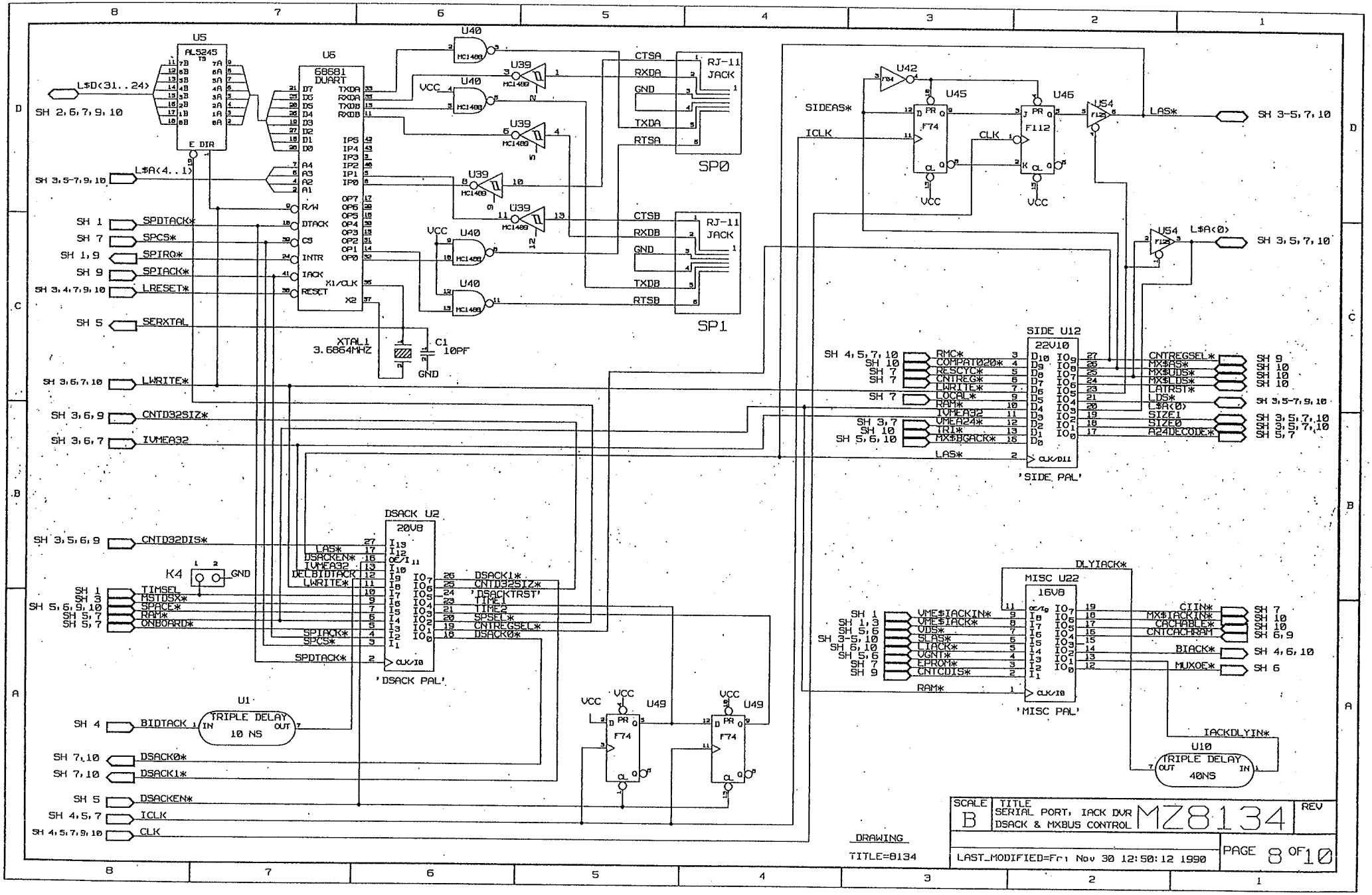

SH 4,7,8 ICLK
SH 4,7-10 CLK
SH 8 SERXTAL Tc=271nS
U44 LS393 BINARY CTR
Tc=4.34uS
U42
U51
U52
VGNT*
VACC*
U54
U45
CLKVRQ*
BG* SH 7
VRQ* SH 6
DWB* SH 3,4,6
MX$BGACK* SH 6,8,10
BR* SH 7
VGNT* SH 6,8
VAEN* SH 3,10
HALT* SH 7,10
BGACK* SH 7
IVUMECYC SH 3
MX$BR* SH 10
MX$BG* SH 10
BRARB U21 16V8
'BRARB PAL'
U9 TRIPLE DELAY 30NS
DELIBG
VDS*
U61
Tc=69.4uS
U50
SRRQ
RACC*
Tc=3.68uS
U7 TRIPLE DELAY 120 NS
CLRRQ*
SH 6 TIMEOUT (69.4-138.9uS)
CLK2LAS* SH 4
RMC* SH 4,7,8,10
SLAS* SH 3,4,8,10
VADDEN* SH 3,4,6
RAS U18 16V8
'RAS PAL'
PRE
DRAS*
DSACKEN*
ONBOARD*
CRAS*
RGNT*
RAS*
A24DECODE*
SPACE*
LDS*
RAM*
LAS*
SH 7,8
SH 6,8-10
SH 3,6-8,10
SH 7,8
SH 3,4,7,8,10
CAS U8 22V10
'CAS PAL'
RGNT*
DP$A(1)
MAILACC*
SLWRITE*
PRE
SIZE1
SIZE0
L$A(1..0)
CNTD32DIS* SH 3,6,8,9
D32SWAP* SH 6
DRAS* SH 6
CAS3* SH 6
CAS2* SH 6
CAS1* SH 6
CAS0* SH 6
SLLWORD* SH 3,6,10
U68
IVDTACK
VME$DTACK* SH 1,4
U63
SLDS1* SH 3,10
SLDS0* SH 3,10
SH 3 RGNT*
SH 10 VACC*
SH 7,8 ONBOARD*
SH 8 DSACKEN*
SH 6,8 VDS*
SH 6 RAS*
SH 3,6 DP$A(1)
SH 6,9 MAILACC*
SH 3,6 SLWRITE*
SH 3,7,8,10 SIZE1
SH 3,7,8,10 SIZE0
SH 3,6-10 L$A(1..0)
DRAWING TITLE=8134
SCALE B
TITLE CPU BUS ARBITRATION DRAM CONTROL,
MZ8134
REV
LAST_MODIFIED=Fri Nov 30 12:47:52 1990
PAGE 5 OF 10

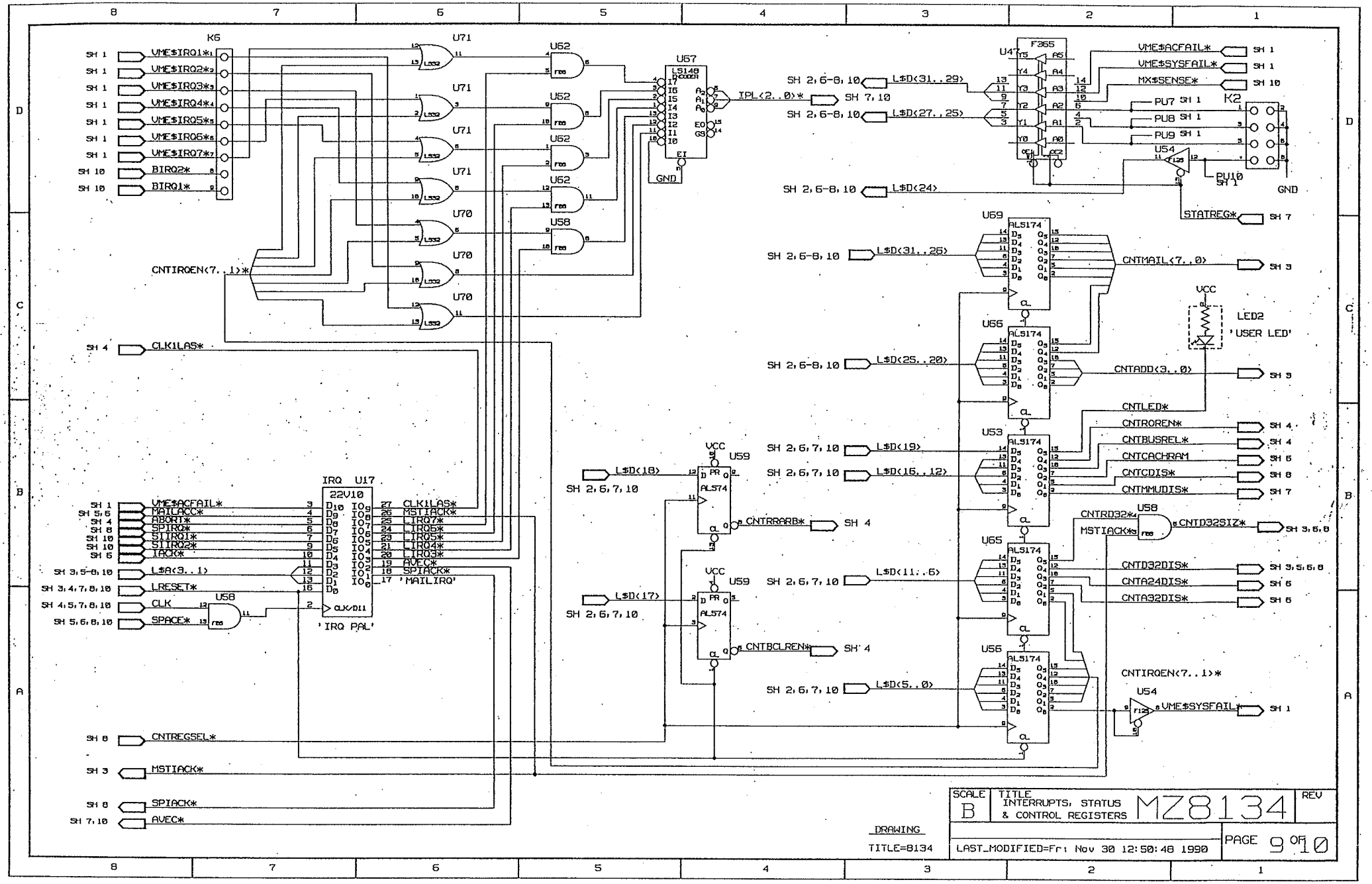


DRAWING
TITLE=8134

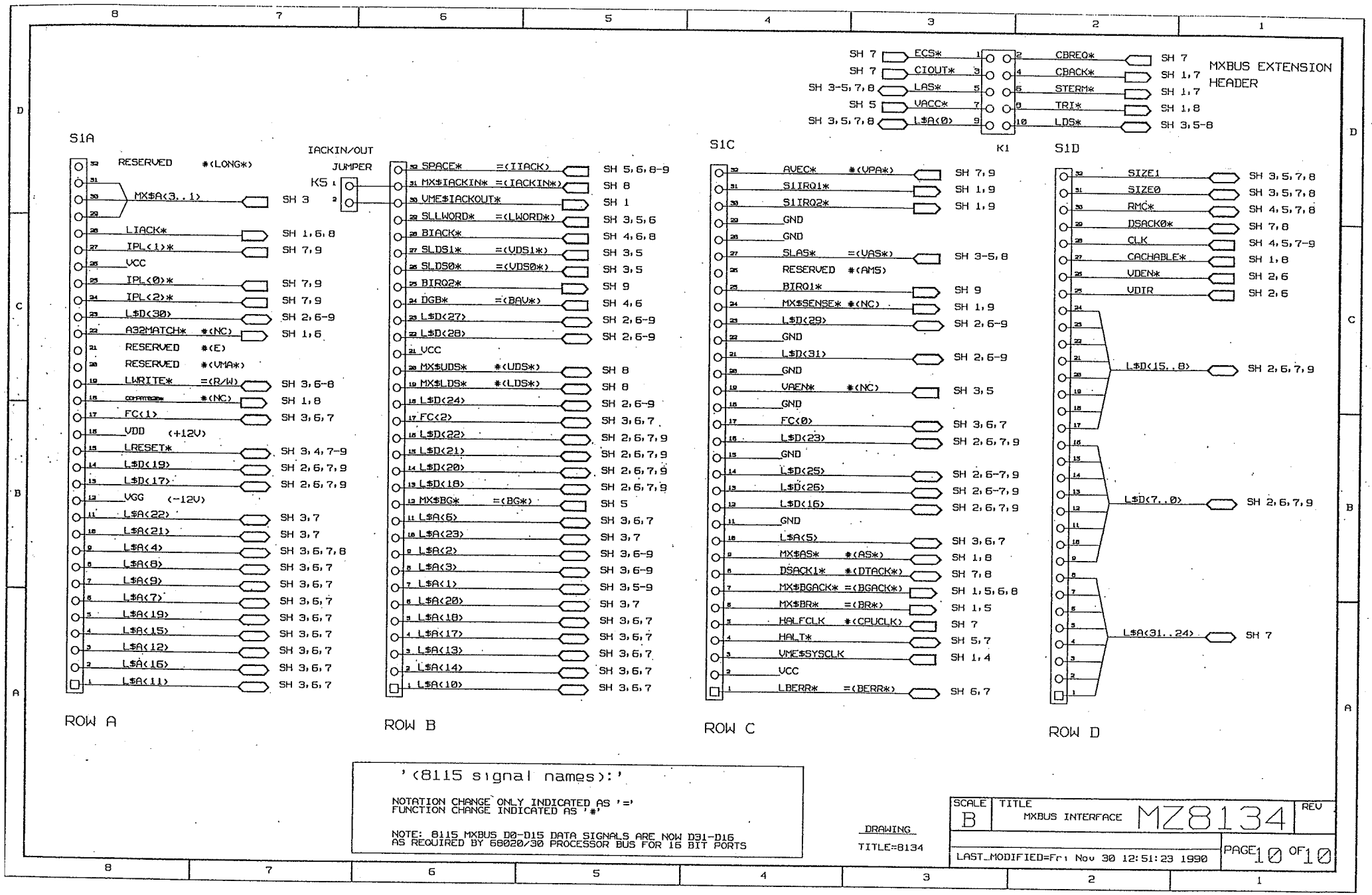
SCALE	TITLE		REV
B	DRAM MEMORY, BUFFER, BERR, & VME DRAM PAL	MZ8134	
	LAST_MODIFIED=Mon Dec 10 16:41:26 1990	PAGE 6 OF 10	

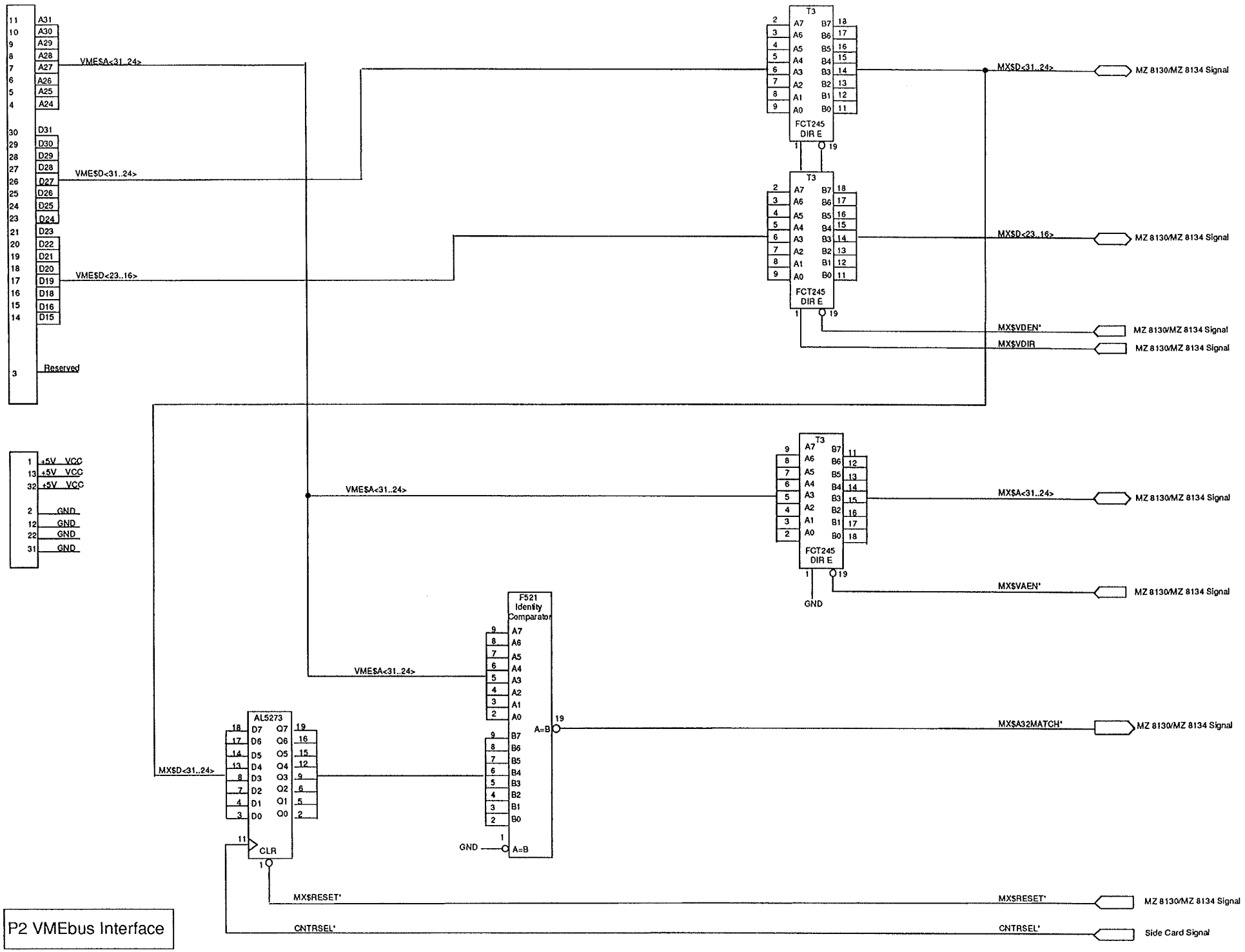

U14
68030
CPU
U15
CLOCK OSCILLATOR (8 PIN) OUTPUT
U3
BYTE-WIDE MEMORY SOCKET
'HB'
'SMARTWATCH'
U4
'LB'
K3
R1
15K
R2
15K
U19
ADDHIGH
16V8
'ADDHIGH PAL'
U23
ADDLOW
16V8
'ADDLOW PAL'
U60
F164
U63
U75
F74
LED3
'HALT'
FC<2..0>
SIZE1
SIZE0
ECS*
RMC*
LAS*
LDS*
LWRITE*
STERM*
DSACK1*
DSACK0*
LBERR*
IPL<2..0>*
AVEC*
CNTMMUDIS*
PU11
CIIN*
CIOUT*
CBREQ*
CBACK*
HALT*
LRESET*
CLK
BR*
BGACK*
BG*
HARDRESET*
STATUS
REFILL
L$D<31..24>
L$D<23..16>
L$D<7..0>
L$D<15..8>
L$A<31..24>
L$A<23..16>
L$A<15..8>
L$A<7..0>
IVMEA32
LOCAL*
A24DECODE*
RAM*
VMEA24*
RESCYC*
HALFCLK
ICLK
VMEA16*
ONBOARD*
EPROM*
SPCS*
STATREG*
CNTREG*
DRAWING
TITLE=8134
SCALE B
TITLE CPU, EPROM, AND ADDRESS DECODING
MZ8134
REV
LAST_MODIFIED=Fri Nov 30 12:49:12 1990
PAGE 7 OF 10


U5
ALS245
L$D<31..24>
SH 2,6,7,9,10
E DIR
U6
68681
DUART
L$A<4..1>
SH 3,5-7,9,10
SH 1 SPDTACK*
SH 7 SPCS*
SH 1,9 SPIRQ*
SH 9 SPIACK*
SH 3,4,7,9,10 LRESET*
SH 5 SERXTAL
XTAL1
3.6864MHZ
C1
10PF
GND
SH 3,6,7,10 LWRITE*
SH 3,6,9 CNTD32SIZ*
SH 3,6,7 IVMEA32
U40
U39
VCC
CTSA
RXDA
GND
TXDA
RTSA
RJ-11
JACK
SP0
CTSB
RXDB
TXDB
RTSB
SP1
U42
U45
U46
U54
SIDEAS*
ICLK
F74
F112
CLK
LAS*
SH 3-5,7,10
L$A<0>
SH 3,5,7,10
SIDE U12
22V10
SH 4,5,7,10 RMC*
SH 10 COMPAT020*
SH 7 RESCYC*
CNTREG*
LWRITE*
SH 7 LOCAL*
RAM*
IVMEA32
SH 3,7 VMEA24*
SH 10 IRI*
SH 5,6,10 MX$BGACK*
LAS*
CLK/D11
'SIDE PAL'
CNTREGSEL*
MX$AS*
MX$UDS*
MX$LDS*
LATRST*
LDS*
L$A(0)
SIZE1
SIZE0
A24DECODE*
SH 9
SH 10
SH 10
SH 10
SH 3,5-7,9,10
SH 3,5,7,10
SH 3,5,7,10
SH 5,7
DSACK U2
20V8
SH 3,5,6,9 CNTD32DIS*
LAS*
DSACKEN*
IVMEA32
DELBIDTACK
LWRITE*
SPIACK*
SPCS*
SPDTACK*
CLK/I8
'DSACK PAL'
DSACK1*
CNTD32SIZ*
'DSACKTRST'
TIME1
TIME2
SPSEL*
CNTREGSEL*
DSACK0*
K4
GND
SH 1 TIMSEL
SH 3 MSTDSX*
SH 5,6,9,10 SPACE*
SH 5,7 RAM*
SH 5,7 ONBOARD*
U1
TRIPLE DELAY
10 NS
SH 4 BIDTACK
SH 7,10 DSACK0*
SH 7,10 DSACK1*
SH 5 DSACKEN*
SH 4,5,7 ICLK
SH 4,5,7,9,10 CLK
U49
DLYIACK*
MISC U22
16V8
SH 1 VME$IACKIN*
VME$IACK*
SH 1,3
SH 5,6
VDS*
SH 3-5,10 SLAS*
SH 6,10 LIACK*
SH 5,6 VGNT*
SH 7 EPROM*
SH 9 CNTCDIS*
RAM*
CLK/I8
'MISC PAL'
CIIN*
MX$IACKIN*
CACHABLE*
CNTCACHRAM
BIACK*
MUXOE*
SH 7
SH 10
SH 10
SH 6,9
SH 4,6,10
SH 6
IACKDLYIN*
U10
TRIPLE DELAY
40NS
SCALE B
TITLE
SERIAL PORT, IACK DVR
DSACK & MXBUS CONTROL
MZ8134
REV
DRAWING
TITLE=8134
LAST_MODIFIED=Fri Nov 30 12:50:12 1990
PAGE 8 OF 10



SCALE	TITLE		REV
B	INTERRUPTS, STATUS & CONTROL REGISTERS	MZ8134	
DRAWING TITLE=8134	LAST_MODIFIED=Fri Nov 30 12:50:48 1990	PAGE 9 OF 10	


MXBUS EXTENSION HEADER
K1
SH 7 ECS* 1
SH 7 CIOUT* 3
SH 3-5,7,8 LAS* 5
SH 5 VACC* 7
SH 3,5,7,8 L$A(0) 9
2 CBREQ* SH 7
4 CBACK* SH 1,7
6 STERM* SH 1,7
8 TRI* SH 1,8
10 LDS* SH 3,5-8
S1A
IACKIN/OUT JUMPER
K5
RESERVED *(LONG*)
MX$A(3..1) SH 3
LIACK* SH 1,6,8
IPL(1)* SH 7,9
VCC
IPL(0)* SH 7,9
IPL(2)* SH 7,9
L$D(30) SH 2,6-9
A32MATCH* *(NC) SH 1,6
RESERVED *(E)
RESERVED *(VMA*)
LWRITE* =(R/W) SH 3,6-8
*(NC) SH 1,8
FC(1) SH 3,6,7
VDD (+12V)
LRESET* SH 3,4,7-9
L$D(19) SH 2,6,7,9
L$D(17) SH 2,6,7,9
VGG (-12V)
L$A(22) SH 3,7
L$A(21) SH 3,7
L$A(4) SH 3,6,7,8
L$A(8) SH 3,6,7
L$A(9) SH 3,6,7
L$A(7) SH 3,6,7
L$A(19) SH 3,6,7
L$A(15) SH 3,6,7
L$A(12) SH 3,6,7
L$A(16) SH 3,6,7
L$A(11) SH 3,6,7
ROW A
SPACE* =(IIACK) SH 5,6,8-9
MX$IACKIN* =(IACKIN*) SH 8
VME$IACKOUT* SH 1
SLLWORD* =(LWORD*) SH 3,5,6
BIACK* SH 4,6,8
SLDS1* =(VDS1*) SH 3,5
SLDS0* =(VDS0*) SH 3,5
BIRQ2* SH 9
DGB* =(BAV*) SH 4,6
L$D(27) SH 2,6-9
L$D(28) SH 2,6-9
VCC
MX$UDS* *(UDS*) SH 8
MX$LDS* *(LDS*) SH 8
L$D(24) SH 2,6-9
FC(2) SH 3,6,7
L$D(22) SH 2,6,7,9
L$D(21) SH 2,6,7,9
L$D(20) SH 2,6,7,9
L$D(18) SH 2,6,7,9
MX$BG* =(BG*) SH 5
L$A(6) SH 3,6,7
L$A(23) SH 3,7
L$A(2) SH 3,6-9
L$A(3) SH 3,6-9
L$A(1) SH 3,5-9
L$A(20) SH 3,7
L$A(18) SH 3,6,7
L$A(17) SH 3,6,7
L$A(13) SH 3,6,7
L$A(14) SH 3,6,7
L$A(10) SH 3,6,7
ROW B
S1C
AVEC* *(VPA*) SH 7,9
SIIRQ1* SH 1,9
SIIRQ2* SH 1,9
GND
GND
SLAS* =(VAS*) SH 3-5,8
RESERVED *(AMS)
BIRQ1* SH 9
MX$SENSE* *(NC) SH 1,9
L$D(29) SH 2,6-9
GND
L$D(31) SH 2,6-9
GND
VAEN* *(NC) SH 3,5
GND
FC(0) SH 3,6,7
L$D(23) SH 2,6,7,9
GND
L$D(25) SH 2,6-7,9
L$D(26) SH 2,6-7,9
L$D(16) SH 2,6,7,9
GND
L$A(5) SH 3,6,7
MX$AS* *(AS*) SH 1,8
DSACK1* *(DTACK*) SH 7,8
MX$BGACK* =(BGACK*) SH 1,5,6,8
MX$BR* =(BR*) SH 1,5
HALFCLK *(CPUCLK) SH 7
HALT* SH 5,7
VME$SYSCLK SH 1,4
VCC
LBERR* =(BERR*) SH 6,7
ROW C
S1D
SIZE1 SH 3,5,7,8
SIZE0 SH 3,5,7,8
RMC* SH 4,5,7,8
DSACK0* SH 7,8
CLK SH 4,5,7-9
CACHABLE* SH 1,8
VDEN* SH 2,6
VDIR SH 2,6
L$D(15..8) SH 2,6,7,9
L$D(7..0) SH 2,6,7,9
L$A(31..24) SH 7
ROW D
'(8115 signal names):'
NOTATION CHANGE ONLY INDICATED AS '='
FUNCTION CHANGE INDICATED AS '*'
NOTE: 8115 MXBUS D0-D15 DATA SIGNALS ARE NOW D31-D16
AS REQUIRED BY 68020/30 PROCESSOR BUS FOR 16 BIT PORTS
DRAWING
TITLE=8134
SCALE B
TITLE MXBUS INTERFACE MZ8134
REV
LAST_MODIFIED=Fri Nov 30 12:51:23 1990
PAGE 10 OF 10


P2 VMEbus Interface
VME$A<31..24>
VME$D<31..24>
VME$D<23..16>
Reserved
+5V VCC
GND
T3
FCT245
DIR E
MX$D<31..24>
MX$D<23..16>
MX$VDEN*
MX$VDIR
MX$A<31..24>
MX$VAEN*
F521
Identity
Comparator
A=B
MX$A32MATCH*
AL5273
CLR
MX$D<31..24>
MX$RESET*
CNTRSEL*
MZ 8130/MZ 8134 Signal
Side Card Signal

Glossary

A16 — A type of module that provides or decodes an address on address lines A01 through A15.

A24 — A type of module that provides or decodes an address on address lines A01 through A23.

A32 — A type of module that provides or decodes an address on address lines A01 through A31.

active high — A signal that is logically active (true) when in a TTL high voltage state (< +2.0 V).

active low — A signal that is logically active (true) when in a TTL low voltage state (< +0.8 V).

address-only cycle — A data transfer bus cycle that consists of an address broadcast, but no data transfer. Slaves do not acknowledge address-only cycles and masters terminate the cycle without waiting for an acknowledgment.

ANSI — Acronym for the American National Standards Institute.

ANSI Standards — American National Standards Institute standards widely used and modified by firms in the United States.

ANSI Standard C — A specification for the C language conforming to ANSI standards.

ARP — Acronym for Address Resolution Protocol which is used to resolve a hardware-level address from an internet address.

ASCII — Acronym for American Standard Code for Information Interchange, a seven-bit digital code for the transmission of data. Extended ASCII uses eight bits to define more characters.

asserted — A signal that is logically active (true).

autovectored interrupt — An interrupt not requiring a user vector to be sent to the processor via the data bus, such as ABORT, BCLR, S1IRQ1, and S1IRQ2 in the MZ 8130 and MZ 8134.

autoboot — See **automatic boot**.

automatic boot — An automatic loading and starting of a program without specific attention from anyone.

backplane — The hardware to which the boards physically attach; the main board in a computer. In a VMEbus system, the VMEbus backplane is a printed circuit board with 96-pin connectors and signal paths that bus the connector pins.

Binary-Coded Decimal (BCD) — A system of writing numbers that assigns a 4-bit binary code to each number between 0 and 10. If a number contains more than one digit, each digit uses four bits. For example, the BCD equivalent in base-10 of 189 is 0001 1000 1001.

Binary-Coded Number (BCN) — A number written in the binary system, sometimes called base 2, using only zero and one to represent any number. For example, the binary equivalent of 189 is 10111101. Binary-coded numbers are easily handled as "off" and "on" conditions by calculators and computers.

benchmark — A standard by which something can be measured.

booting — Starting the operating system or other application.

bootstrapping — See **booting**.

breakpoint — A location in a program where the program pauses. A debugger allows you to set breakpoints in a program. When the program pauses, you can use the debugger to examine the program's progress and modify its execution.

bus — The format and design of signal arrangements on the backplane, the set of channels through which signals pass from board to board. A standardized format and design for the electrical and mechanical interconnection of printed circuit boards and the backplane.

A set of parallel wires over which data is communicated; a common set of electrical lines shared between electrical components. Buses can be internal to a processor, can connect a processor to main or secondary memory, or connect a computer system to outside peripherals.

bus master — The device that controls the bus. Every device on a bus that is capable of becoming a master is assigned a priority. The device with the higher priority receives control when both request control simultaneously.

byte — A generic term to indicate a measurable portion of consecutive binary digits. For example, in the MZ 8134, it refers to the 8 bits data accessed as a group that can appear at any address in memory. Its bits are numbered from right to left, starting with 0, the least significant bit (LSB), and ending with 7, the most significant bit (MSB).

cache — A small, extremely high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed.

card — An easily replaced printed circuit board.

card cage — The physical structure that actually holds the printed circuit boards.

central processing unit (CPU) — The brain of the system; it executes UNIX system calls and commands. See **cluster node**, **diskless node**, **server**, and **VxWorks target**.

cluster — More than one CPU in a backplane, implying a multiprocessing environment. See **cluster node**.

cluster node — A set of boards running UNIX that occupy a portion of the card cage in a Mizar Optimum system. A cluster node can be a graphics node (workstation), or it can be a non-graphics node (compute node) with two serial lines for terminal connections (console and Port 0).

configuration — The arrangement of the software and/or hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.

controller — A hardware device that regulates the transmission of data over the data links of a network, between peripherals and the system.

CPU — See **central processing unit**.

CPU ID — A method of identifying a CPU in a multiple CPU system. The CPU ID of an MZ 8134 is set with the K2 jumper block.

D16 — A master or slave that sends and receives data 16 bits at a time over data lines D00 to D15. An interrupter that sends STATUS/IDs 16 bits at a time over data lines D00 to D15. An interrupt handler that receives STATUS/IDs 16 bits at a time over data lines D00 to D15.

D32 — A master or slave that sends and receives data 32 bits at a time over data lines D00 to D31. An interrupter that sends STATUS/IDs 32 bits at a time over data lines D00 to D31. An interrupt handler that receives STATUS/IDs 32 bits at a time over data lines D00 to D31.

data transfer bus — One of the four buses provided by the VMEbus backplane. It allows masters to direct the transfer of binary data between themselves and slaves.

data word size — The specific length of data word, such as word or byte, that a particular microprocessor is designed to handle. Generally, the longer the word length, the higher the data throughput, the greater the precision for arithmetic operation, and the greater the memory addressing capability.

daughter card — A full-height printed circuit board that attaches to the front of a full-height CPU board.

decimal number — A numbering system based on 10.

default The value used by software or hardware when none is supplied.

device A physical input-output unit.

dhrystones An integer operations-oriented benchmark frequently used for board-level performance and compiler performance measurements.

diagnostic output The output produced by a program that reports errors or problems the program encountered when executing, usually error messages displayed on the screen.

diagnostics A set of utilities to help locate hardware problems.

disk A magnetic storage medium for large amounts of data. UNIX refers to each disk as a separate device signified by two letters and a number. The number represents the first drive, the second drive, etc. Each disk has seven or eight logical storage areas (partitions labeled a to g or h). These partitions (software) are not cylinders (hardware). SMD-type (sm) disks are numbered **0** for the first one, **2** for the second SMD-type device, **4** for the third SMD-type device, and so on.

diskless node A system with no disk of its own, connecting through an Ethernet network to a server node. The diskless node can be a graphics node (workstation), or it can be a non-graphics node (compute node).

display Generally refers to information displayed on screen or terminal. In a non-X Window System environment, the screen or terminal. In an X environment, a set of graphics screen(s) driven by a single X server.

DMA Acronym for Direct Memory Access, a procedure to gain direct access to main storage to achieve data transfer without involving the CPU. Access to physical memory without CPU control.

DRAM Acronym for Dynamic Random Access Memory.

driver A software package that enables communication between the operating system and a specific peripheral.

DTB See **data transfer bus.**

DUART Acronym for the Signetics SCN68681 Dual Asynchronous Receiver/Transmitter serial port chip providing two independent full-duplex asynchronous RS-232-C serial ports to two RJ11 connectors on the MZ 8134.

EPROM Acronym for Electrically Programmable Read Only Memory; it can be erased and reprogrammed by an electrical signal.

Ethernet A local area network protocol commonly used to network computers. A Carrier Sense Multi-Access with Collision Detection (CSMA/CD) network that conforms to industry specifications. A network in which all stations monitor the network during their own transmissions, terminating transmission immediately if a collision is detected.

falling edge A signal that is changing from a high level to a low level.

file A collection of related records treated as a unit – a basic unit of information storage under UNIX. Each file is treated as a linear array of bytes. A file can contain data and/or programs.

file system A collection of files that can be mounted on a block special file. The collection of all files on a computer. Or a directory usually within the root directory (/) that is assigned its own area on a disk and that functions independently of other such file systems.

ftp A file transfer protocol for file transfer across a network.

functional module A collection of electronic circuitry that resides on one VMEbus board and works together to accomplish a task.

graceful shutdown An orderly shutdown following the procedures to stop all processes and flush all disk buffers.

graphics device A physical graphics input-output unit, such as a graphics workstation. A graphics device requires a graphics controller attached to a CPU. Prior to Mizar's Release 5.2 of 4.3 BSD UNIX, each graphics device needed its own CPU.

graphics workstation An input/output device capable of displaying bit-mapped graphics.

hexadecimal number A numbering system based on 16. The numbers 0 through 9 are the same in hexadecimal notation. However, the numbers 10 through 16 are represented by single characters, usually the letters A through F.

high signal A high signal is in a TTL high voltage level. For example, the MZ 8134 is high when it is < +2.0V.

host The primary computer attached to a network providing services such as computation, database access, special programs, or programming languages. An information processor providing supporting services. A computer configured to share resources.

hostname The official system name for any node.

ICMP Acronym for Internet Control Message Protocol; used for reporting errors and controlling Internet Protocol.

interface A common boundary between parts of a system. See **user interface.**

internet A network using the Internet protocol to support connectivity.

internet protocol (IP) A protocol developed by the United States Department of Defense to support communication among networks.

internet address An address consisting of four fields identifying the location of any node or device on the network.

interrupt An external event causing a change in the normal flow of instructions in a currently executing process. In UNIX or VxWorks, interrupts are generally sent by hardware devices external to the CPU.

IP Acronym for the Internet Protocol.

IPC Interprocess communication utility. UNIX and VxWorks provide a variety of ways for tasks to communicate with each other.

jumper A conductor placed between the ends of two electrical contacts to act as a shunt or switch. The most common jumper is a plastic-covered metal piece that is placed across two pins on a printed circuit board.

LSB Acronym for the Least Significant Bit. The bit in the rightmost position in a byte.

library A collection of tested programs, routines, and subroutines available for use by other programs.

logic state The binary values (0 or 1) at the nodes of logic elements and ICs at a specific time. Also known as "logic signals."

long word The longest computer word which a particular computer system can handle. For example, in the MZ 8134, it refers to 32 bits of data accessed as a group that can appear at any address in memory evenly divisible by four. Its bits are numbered from right to left, starting with 0, the least significant bit (LSB), and ending with 31, the most significant bit (MSB).

low signal A low signal is in a TTL low voltage state. For example, the MZ 8134 is low when it is < +0.8 V.

master A functional module that initiates data transfer bus (DTB) cycles in order to transfer data between itself and a slave module.

memory In UNIX, physical memory represents the available space in main memory. Programs are either swapped or paged into physical memory for execution. Virtual memory management permits programs to treat disk storage as an

	extension of main memory.
module	See **functional module**.
MSB	Acronym for the Most Significant Bit. The bit in the leftmost position in a byte.
mount	To associate the root of a file system with a directory entry in an already-mounted file system.
MTBF	Acronym for Mean Time Between Failures.
multiprocessing	An environment with two or more CPUs in the same backplane.
multitasking	An environment where CPU time is shared between several processes simultaneously.
MXbus	Mizar's proprietary backplane bus that enables a card to be used as a side card. It is a standardized VME expansion interface which allows bus mastership of the base CPU module. It has a wide range of uses, from simple I/O ports to complex parallel processor expansion modules.
negated	A signal that is logically inactive (false).
network	The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.
Network File System	(NFS) A utility facilitating communication between a variety of nodes on a local network.
node	Any system in a local network with its own CPU board. For example, in a cluster network, the server system and each cluster are nodes.
NFSstones	Benchmark for network file servers using NFS.
no-op	A shortened form of no-operation; an instruction that does not execute an operation, but merely advances execution to the next instruction.
octal number	A numbering system based on 8, sometimes used as shorthand for long binary numbers.
OS-9	A multiuser, multitasking real-time operating system with self-hosted software development.
packet	basic unit of information transmitted over the network.
page	The fixed size unit of measurement used to divide an address space for memory management.
panic	An unrecoverable system failure, usually causing the machine to reboot.

pathname	The directions for traversing the file system tree. An absolute pathname gives directions from the root to the file. A relative pathname gives directions from the current directory to the file.
physical memory	The actual physical total of random access memory (RAM) available to a system
pipe	An interprocess communication mechanism for one-way reliable data transfer. A pipe directs the output of one command to the input of another command.
print	The hardcopy output of a printer, the printed page.
process	A program in execution that can be scheduled by the CPU. An activity carried out by the system, including computing, assembling, compiling, etc. Each process is assigned a number unique to the system to track its progress until completion.
Programmable Read-Only (PROM) Chips	Programmable Read-Only Memory (PROM) chips on the CPU contain a monitor and loading facility for UNIX or VxWorks. These PROMs bring up the system and execute the larger body of UNIX from disk or tape.
prompt	Any text string requesting input from the user. A combination of letters, numbers, and/or symbols at which users type commands in the shell. A common C-shell prompt is **%**.
protocol	A set of conventions (format, error detection, etc.) used between communicating processes, metaphorically, a common language.
Qbus	A backplane bus by Digital Equipment Corporation. A standardized format and design for the electrical and mechanical interconnection of printed circuit boards and the backplane.
RARP	Acronym for Reverse Address Resolution Protocol which is used to resolve an interenet address from a hardware-level network address.
read-modify-write cycle	A data transfer bus cycle used to both read from and write to a slave location without permitting any other master to access that location.

RFS Acronym for Remote File Sharing; a networking facility in UNIX System V.

rising edge A signal that is changing from a low level to a high level.

ROM Acronym for Read-Only Memory; nonvolatile memory that retains its data even when power is off.

root The directory / at the top of the UNIX file system.

screen The display surface of a monitor for text and graphics. A monitor screen can contain several X11 screens.

SCSI Acronym for Small Computer Systems Interface. A protocol and set of commands used in communication between some peripherals and their drivers. A peripheral bus between the CPU and the hard, floppy, and tape drives.

server A server is always the first CPU in the system. In a non-X environment, the server node. In an X Window System environment, the program that provides X11's basic windowing mechanism. The server, also known as the X server, manages clients.

server node The central node for a network of other nodes without disks of their own. For diskless networks, the server node must have an Ethernet controller.

side card A half-height printed circuit board that attaches to the side of a half-height CPU board.

signal See **UNIX signal.**

single step Execute one line of source code at a time.

single-user mode A state of the operating system in which only one user is supported.

single-user prompt # The prompt displayed when in single-user mode.

slave A functional module that detects data transfer bus (DTB) cycles initiated by a master and, when those cycles specify its participation, transfers data between itself and the master.

SmartWatch The Dallas Semiconductor DS1216 self-contained, battery backed-up, time-of-day clock used on several Mizar boards.

socket In BSD 4.3, an interprocess communication mechanism and its associated data structure and system call.

source-level debugger A program that enables you to examine a program's source code while changing the program's object (executable) file.

stack That portion of computer memory or registers used to hold information temporarily. A dynamic, sequential data list, usually contained in system memory, having special provisions for program access from one end or the other. Storage and retrieval of data from the stack is generally automatically performed by the processor. Items in a stack are frequently processed in a Last-In-First-Out order. Adding an item to a stack is called "pushing" because the new item pushes all of the other items down the stack. Deleting an item from a stack is called "popping" because the stack pops up to fill the gap left by the removed item.

stand alone A device or program that can operate independently. Software, such as the diagnostics, that can run without the operating system.

standard input The default means of sending input to a command, usually by typing it in at the keyboard.

standard output The default means of receiving output from a command, usually by displaying it on the screen.

symbol table The table produced by a compiler or an interpreter listing each variable, its attributes, and its absolute or relative location in memory.

system administrator The individual performing the necessary maintenance and troubleshooting procedures to keep computer systems up and running.

system console The terminal or workstation plugged into the console port of the UNIX or VxWorks system.

system controller See **controller.**

system name See **hostname.**

target See **VxWorks target.**

TCP Acronym for Transmission Control Protocol, a standard internet protocol.

TCP/IP Acronym for Transmission Control Protocol/Internet Protocol. A combination of two network protocols which provide addressing, routing, forwarding, error detection, and error correction utilities.

terminal An input/output device for communicating with UNIX. To UNIX, a terminal is simply a special file. UNIX writes to the file (things appear on your screen) and reads from the file (you type something on your keyboard).

terminal line — You must add terminals, printers, and modems manually by changing special *tty* files. You must tell UNIX what you are using, what settings (such as baud rate and parity), and where you are plugging it into the system.

termination — The provisions for ending a transmission line and connecting to a bus bar or other terminating device.

TFTP — Acronym for Trivial File Transfer Protocol which is frequently used for booting diskless workstations. It uses the User Datagram Protocol (UDP) protocol to transfer files over the internet.

trace — A debugging routine or program that lists every instruction in a program, along with the value that results from the execution of each instruction. **trace** also is the execution of such a routine or program.

tristated — A signal that can assume three output states: high, low, or high impedance. In the MZ 8134, a signal that is neither low nor high, but is seen as a high impedance to the signal line.

TTL — Acronym for Transistor-Transistor Logic. A family of integrated circuit logic elements with a specific output structure.

UNIX — A multiuser operating system. Features include portability, multitasking, timesharing, and easy implementation of graphics and multiprocessing utilities.

UNIX engine — A CPU board and peripheral controllers used as the main processor for a UNIX system. This board is always seated in the first slot. It controls the traffic for all UNIX and VxWorks processors in the system.

UNIX signal — A communication sent from one process to another to notify the process of an event taking place. The receiving process can either ignore the signal or handle it through a process called the interrupt handler.

user interface — The visual and mechanical features reflecting how the user communicates with a product or system.

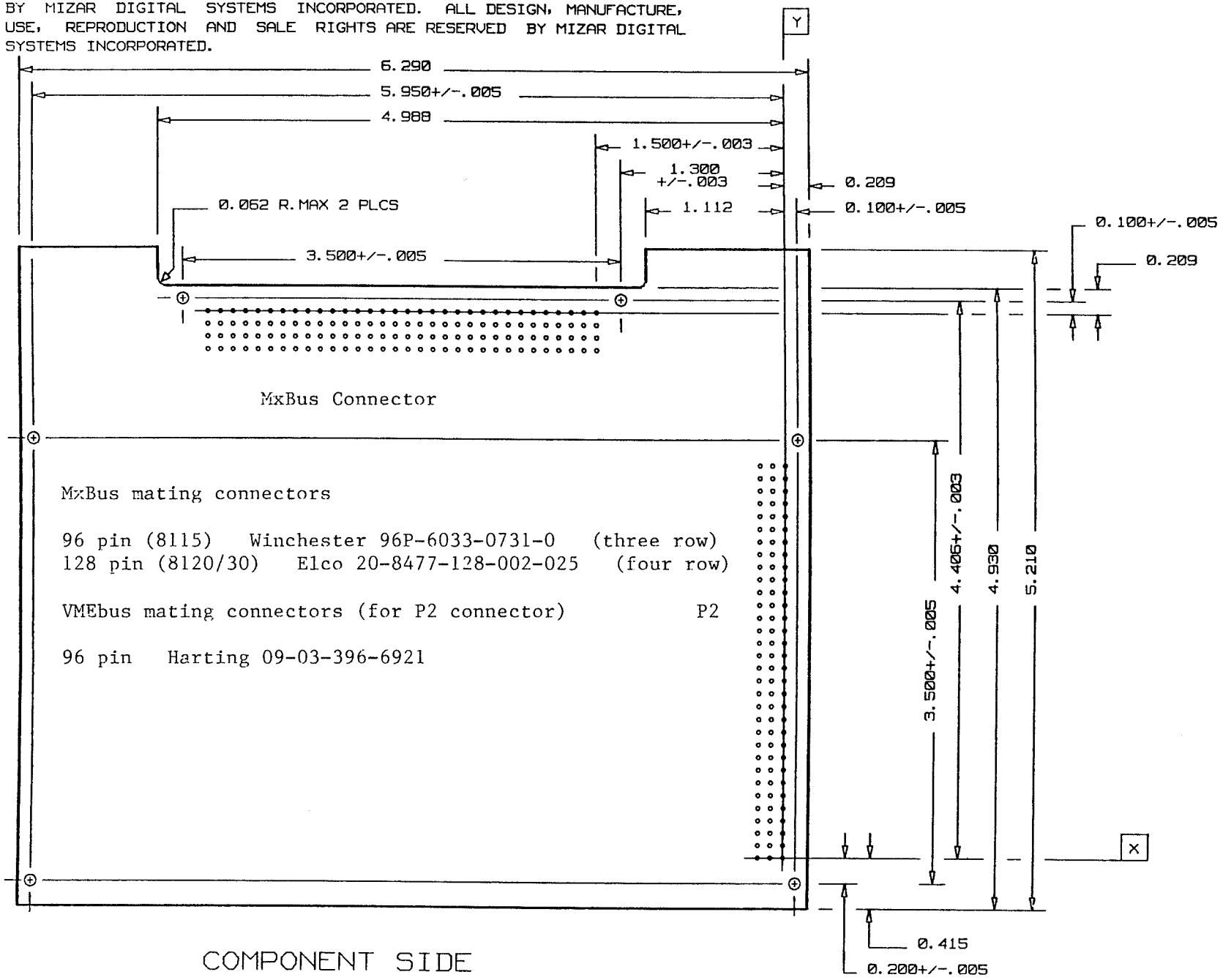
VDMA — Virtual Direct Memory Access. Accessing virtual memory directly and transparently to peripherals.

virtual memory — A memory mapping facility extending the range of available memory addresses beyond the actual physical memory address limits.

VMEbus — VersaModule Europa 16/32-bit backplane bus. A standardized format and design for the electrical and mechanical interconnection of printed circuit boards and the backplane.

VxWorks	A real-time multitasking operating system from Wind River Systems.
whetstones	A benchmark program consisting of floating point and integer arithmetic, functional calls, array indexing, conditional jumps, and transcendental functions, used primarily to measure the performance of a floating point library.
VxWorks target	A board or set of boards running VxWorks. The CPU on which VxWorks runs.
word	A unit of data. A set of characters that occupies one storage location and is treated by computer circuits as a unit and transported as such. For example, in the MZ 8134, it refers to 16 bits of data accessed as a group that can appear at any address in memory evenly divisible by two. Its bits are numbered from right to left, starting with 0, the least significant bit (LSB), and ending with 16, the most significant bit (MSB).
workstation	A node with a graphics subsystem.
X Window System	MIT's large, versatile windowing graphics interface providing a standardized windowing ability for a graphics workstation. In a windowing system, documents, programs, and tools are contained within windows. The windows accept commands and input from both mouse and keyboard.
X25	A wide area network communications protocol often used for synchronous communications.
YP	Acronym for Yellow Pages, a network service.

ALL INFORMATION DISCLOSED BY THIS DOCUMENT IS CONSIDERED PROPRIETARY BY MIZAR DIGITAL SYSTEMS INCORPORATED. ALL DESIGN, MANUFACTURE, USE, REPRODUCTION AND SALE RIGHTS ARE RESERVED BY MIZAR DIGITAL SYSTEMS INCORPORATED.



COMPONENT SIDE

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP