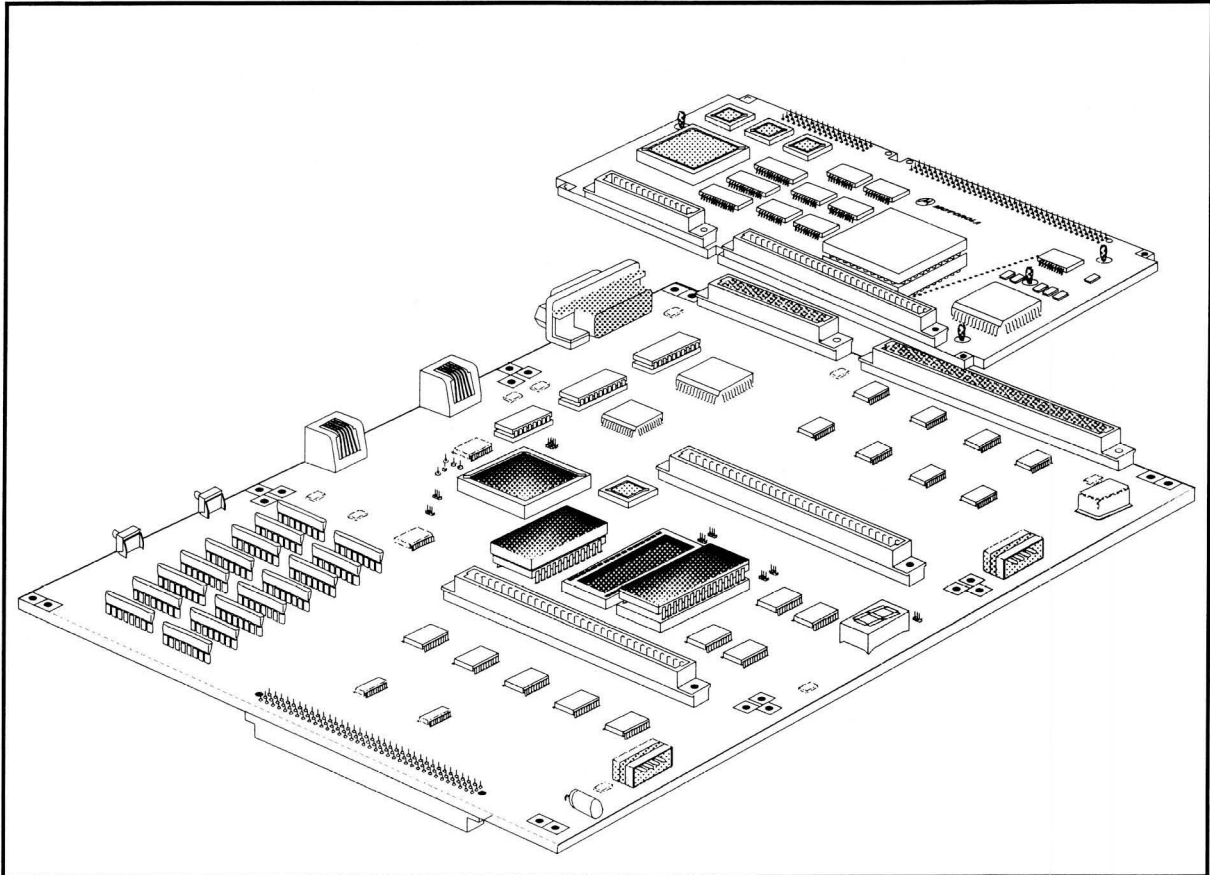


I.D.P.



M68EC0x0
Integrated Development Platform
Users Manual
Rev 3.0




MOTOROLA



M68EC0x0 Integrated Development Platform Users Manual

Revision 3.0
October 1993

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

PREFACE

The M68EC0x0IDP Integrated Development Platform Users Manual along with a M68EC0x0 CPU Module User's Manual provide the complete documentation for the M68EC0x0IDP Integrated Development Platform. The M68EC0x0IDP Users Manual is organized as follows:

Section 1	Introduction
Section 2	Installation and Operation
Section 3	Hardware Configuration
Section 4	ROM68K Configuration and Command Set
Section 5	MON68 Configuration and Command Set
Appendix A	M68EC0x0IDP System Calls
Appendix B	MC68230 Programming Example
Appendix C	M68EC0x0IDP Schematics

Trademarks

MON68 is a trademark of Intermetrics.

ROM68K is a trademark of Integrated Systems Inc.

XRAY+ is a registered trademark of Integrated Systems Inc.

pSOS+ is a registered trademark of Integrated Systems Inc.

XDB Debugger is a registered trademark of Intermetrics

DEC VT220 is a registered trademark of Digital Equipment Corp.

IBM-PC is a registered trademark of International Business Machines

Macintosh is a registered trademark of Apple Computers Inc.

— Sales Offices —

UNITED STATES

ALABAMA , Huntsville	(205) 464-6800
ARIZONA , Tempe	(602) 897-5056
CALIFORNIA , Agoura Hills	(818) 706-1929
CALIFORNIA , Los Angeles	(310) 417-8848
CALIFORNIA , Irvine	(714) 753-7360
CALIFORNIA , Roseville	(916) 922-7152
CALIFORNIA , San Diego	(619) 541-2163
CALIFORNIA , Sunnyvale	(408) 749-0510
COLORADO , Colorado Springs	(719) 599-7497
COLORADO , Denver	(303) 337-3434
CONNECTICUT , Wallingford	(203) 949-4100
FLORIDA , Maitland	(407) 628-2636
FLORIDA , Pompano Beach/Fort Lauderdale	(305) 486-9776
FLORIDA , Clearwater	(813) 538-7750
GEORGIA , Atlanta	(404) 729-7100
IDAHO , Boise	(208) 323-9413
ILLINOIS , Chicago/Hoffman Estates	(708) 490-9500
INDIANA , Fort Wayne	(219) 436-5818
INDIANA , Indianapolis	(317) 571-0400
INDIANA , Kokomo	(317) 457-6634
IOWA , Cedar Rapids	(319) 373-1328
KANSAS , Kansas City/Mission	(913) 451-8555
MARYLAND , Columbia	(410) 381-1570
MASSACHUSETTS , Marlborough	(508) 481-8100
MASSACHUSETTS , Woburn	(617) 932-9700
MICHIGAN , Detroit	(313) 347-6800
MINNESOTA , Minnetonka	(612) 932-1500
MISSOURI , St. Louis	(314) 275-7380
NEW JERSEY , Fairfield	(201) 808-2400
NEW YORK , Fairport	(716) 425-4000
NEW YORK , Hauppauge	(516) 361-7000
NEW YORK , Poughkeepsie/Fishkill	(914) 473-8102
NORTH CAROLINA , Raleigh	(919) 870-4355
OHIO , Cleveland	(216) 349-3100
OHIO , Columbus Worthington	(614) 431-8492
OHIO , Dayton	(513) 495-6800
OKLAHOMA , Tulsa	(800) 544-9496
OREGON , Portland	(503) 641-3681
PENNSYLVANIA , Colmar Philadelphia/Horsham	(215) 997-1020 (215) 957-4100
TENNESSEE , Knoxville	(615) 690-5593
TEXAS , Austin	(512) 873-2000
TEXAS , Houston	(800) 343-2692
TEXAS , Plano	(214) 516-5100
VIRGINIA , Richmond	(804) 285-2100
WASHINGTON , Bellevue Seattle Access	(206) 454-4160 (206) 622-9960
WISCONSIN , Milwaukee/Brookfield	(414) 792-0122

Field Applications Engineering Available
Through All Sales Offices

CANADA

BRITISH COLUMBIA , Vancouver	(604) 293-7605
ONTARIO , Toronto	(416) 497-8181
ONTARIO , Ottawa	(613) 226-3491
QUEBEC , Montreal	(514) 731-6881

INTERNATIONAL

AUSTRALIA , Melbourne	(61-3)887-0711
AUSTRALIA , Sydney	(61-2)906-3855
BRAZIL , Sao Paulo	55(11)815-4200
CHINA , Beijing	86 505-2180

FINLAND , Helsinki	358-0-35161191
Car Phone	358(49)211501
FRANCE , Paris/Vanves	33(1)40 955 900
GERMANY , Langenhagen/ Hanover	49(511)789911
GERMANY , Munich	49 89 92103-0
GERMANY , Nuremberg	49 911 64-3044
GERMANY , Sindelfingen	49 7031 69 910
GERMANY , Wiesbaden	49 611 761921
HONG KONG , Kwai Fong	852-4808333
Tai Po	852-6668333
INDIA , Bangalore	(91-812)627094
ISRAEL , Tel Aviv	972(3)753-8222
ITALY , Milan	39(2)82201
JAPAN , Aizu	81(241)272231
JAPAN , Atsugi	81(0462)23-0761
JAPAN , Kumagaya	81(0485)26-2600
JAPAN , Kyushu	81(092)771-4212
JAPAN , Mito	81(0292)26-2340
JAPAN , Nagoya	81(052)232-1621
JAPAN , Osaka	81(06)305-1801
JAPAN , Sendai	81(22)268-4333
JAPAN , Tachikawa	81(0425)26-2340
JAPAN , Tokyo	81(03)3440-3311
JAPAN , Yokohama	81(045)472-2751
KOREA , Pusan	82(51)4635-035
KOREA , Seoul	82(2)554-5188
MALAYSIA , Penang	60(4)374514
MEXICO , Mexico City	52(5)282-2864
MEXICO , Guadalajara	52(36)21-8977
Marketing	52(36)21-9023
Customer Service	52(36)669-9160
NETHERLANDS , Best	(31)49988 612 11
PUERTO RICO , San Juan	(809)793-2170
SINGAPORE	(65)2945438
SPAIN , Madrid	34(1)457-8204
or	34(1)457-8254
SWEDEN , Solna	46(8)734-8800
SWITZERLAND , Geneva	41(22)7991111
SWITZERLAND , Zurich	41(1)730 4074
TAIWAN , Taipei	886(2)717-7089
THAILAND , Bangkok	(66-2)254-4910
UNITED KINGDOM , Aylesbury	44(296)395-252

FULL LINE REPRESENTATIVES

CALIFORNIA , Loomis	
Galena Technology Group	(916) 652-0268
COLORADO , Grand Junction	
Cheryl Lee Whitley	(303) 243-9658
KANSAS , Wichita	
Melinda Shores/Kelly Greiving	(316) 838 0190
NEVADA , Reno	
Galena Technology Group	(702) 746 0642
NEW MEXICO , Albuquerque	
S&S Technologies, Inc.	(505) 298-7177
UTAH , Salt Lake City	
Utah Component Sales, Inc.	(801) 561-5099
WASHINGTON , Spokane	
Doug Kenley	(509) 924-2322
ARGENTINA , Buenos Aires	
Argonics, S.A.	(541) 343-1787

HYBRID COMPONENTS RESELLERS

Elmo Semiconductor	(818) 768-7400
Minco Technology Labs Inc.	(512) 834-2022
Semi Dice Inc.	(310) 594-4631

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
Section 1		
Introduction		
1.1	M68EC0x0IDP Functionality	1-2
1.1.1	Microprocessor Evaluation	1-2
1.1.2	Software Development	1-2
1.1.3	Hardware Development	1-3
1.2	Supporting Documentation	1-3
1.2	Vendors Supporting IDP	1-3
Section 2		
Installation and Operation		
2.1	Unpacking	2-1
2.2	Equipment Requirements	2-1
2.3	Assembly And Power-On	2-1
2.4	ASCII Terminal/Host Computer Connection	2-3
2.5	Software Interface	2-3
2.6	Troubleshooting	2-4
Section 3		
Hardware Configuration		
3.1	Connector Pinouts	3-1
3.1.1	Power	3-2
3.1.2	Serial Input/Output	3-3
3.1.3	Parallel Input/Output	3-4
3.1.4	Jumper Settings	3-6
3.2	M68EC0x0IDP Motherboard	3-7
3.2.1	M68EC0x0IDP Bus Interface	3-8
3.2.2	M68EC0x0IDP Bus Arbitration	3-8
3.2.3	M68EC0x0IDP Interrupts	3-9
3.2.4	M68EC0x0 Motherboard Memory Map	3-9
3.2.5	M68EC0x0 Motherboard 68K-Based CPU Module Slot	3-9
3.2.6	M68EC0x0 Motherboard I/O Slots	3-12
3.2.7	DRAM Array	3-12
3.2.7.1	DRAM Memory Configuration	3-12
3.2.7.2	DRAM Memory Map Division	3-12
3.2.8	32-Pin EPROM Sockets	3-13

TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
Section 3 Hardware Configuration		
3.2.8.1	User ROM Autoboot	3-14
3.2.8.2	EPROM Configuration	3-14
3.2.9	Real-Time Clock/Calendar	3-15
3.2.9.1	Clock/Calendar	3-16
3.2.9.2	Battery-Backed SRAM Usage	3-16
3.2.10	Dual RS-232C Serial Ports	3-16
3.2.11	Parallel Interface/Timer	3-18
3.3.12	Status LED	3-20
3.2.13	Reset Switch	3-21
3.2.14	Abort Switch	3-22
3.2.15	Power-On LED	3-22
3.2.16	Power Connectors	3-22
3.2.17	MC88916 Clock	3-22
3.2.17.1	Jumper J1	3-23
3.2.17.2	Jumper J3	3-23
3.2.17.3	Oscillator Y1	3-23
3.2.17.4	$\overline{\text{CLKEN}}$	3-23
3.3	IDP BUS Specification.	3-24
3.3.1	I/O Slot Connector Pinouts	3-25
3.3.2	Bus Signals.....	3-25
3.3.3	Electrical Specifications	3-27
3.3.3.1	Power Supply	3-27
3.3.3.2	Bus Connector	3-27
3.3.3.3	Bus Driver Characteristics	3-28
3.3.3.4	Bus Receiver Characteristics	3-28
3.3.3.5	Transceiver Characteristics	3-28
3.3.4	Bus Timing	3-29
3.3.5	Memory Map	3-29
3.3.6	ID ROM Format	3-30
3.4	Mechanical Specifications	3-30
3.4.1	IDP Bus Connector	3-30
3.4.2	I/O Module Drawing	3-30

Section 4 ROM68K Configuration and Command Set

4.1	Overview	4-1
4.2	Single CPU	4-1
4.2.1	Downloading Via RS232	4-1

TABLE OF CONTENTS (Concluded)

Paragraph Number	Title	Page Number
Section 4		
ROM68K Configuration and Command Set		
4.2.2	CPU With An Ethernet Interface	4-2
4.3	ROM68K Command Set.....	4-3
4.3.1	Memory Manipulation Commands	4-4
4.3.2	Register Manipulation Commands	4-4
4.3.3	Execution/Breakpoint Commands	4-4
4.3.4	File Loading/Bootting Commands	4-5
4.3.5	Miscellaneous Commands	4-5
Section 5		
MON68 Configuration and Command Set		
5.1	General Description	5-1
5.2	Troubleshooting	5-1
5.3	Starting MON68 With XDB	5-2
5.4	Using MON68 With XDB	5-2
5.4.1	Transparency Mode	5-2
5.4.2	Trace Mode	5-3
5.4.3	Data Breakpoints	5-3
5.5	Using MON68 With A Dumb Terminal.....	5-3
5.6	MON68 Commands	5-4
Appendix A		
M68EC0x0IDP System Calls		
Appendix B		
MC68230 Programming Example		
Appendix C		
M68EC0x0IDP Schematics		

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
2-1	PC, Dec VT-220, Or Macintosh Connections	2-3
3-1	M68EC0x0 Motherboard	3-1
3-2	M68EC0x0 Motherboard Connectors	3-2
3-3	P3/P4 Connectors	3-2
3-4	P7/P8 Connectors	3-3
3-5	P10/P9 Connectors	3-3
3-6	P5 Connector	3-4
3-7	P6 Connector	3-5
3-8	Jumper Diagram	3-7
3-9	M68EC0x0 Motherboard Block Diagram	3-8
3-10	EPROM Configuration	3-15
3-11	Serial I/O Connector Pinout	3-17
3-12	P5 Connector Pinout	3-20
3-13	LED Segments	3-21
3-14	P16/P17 Connectors	3-22
3-15	M68EC0x0 IDP IO Slot Connector	3-25
3-16	Memory Map	3-29
3-17	Mechanical Dimensions	3-31
B-1	MC68230 Programming Example	B-1

LIST OF TABLES

Table Number	Title	Page Number
2-1	PC, VT-220, and Macintosh RJ-11/DB25 Connections	2-3
3-1	P3/P4 Connector Pinout	3-2
3-2	P7/P8 Connector Pinout	3-3
3-3	P10/P9 Connector Pinout	3-4
3-4	P5 Connector Pinout	3-5
3-5	P6 Connector Pinout	3-6
3-6	Jumper Description	3-7
3-7	Bus Request Inputs	3-9
3-8	M68EC0x0IDP Interrupts	3-9
3-9	Motherboard Memory Map	3-10
3-10	M68EC0X0 Motherboard P1 Connector Pinout	3-11
3-11	M68EC0X0 Motherboard P2 Connector Pinout	3-11
3-12	Motherboard DRAM Memory Map Division	3-13
3-13	Motherboard EPROM Access Widths	3-13
3-14	Motherboard EPROM Jumper Settings	3-14
3-15	MK48TO2 Memory Map	3-15
3-16	Non-Volatile RAM Memory Map	3-16
3-17	MC68681 Memory Map	3-17
3-18	P7/P8 Connector Pinout	3-18
3-19	MC68230 Memory Map	3-19
3-20	MC68230 to P5 Printer Connections	3-20
3-21	LED Display Values	3-21
3-22	P16/P17 Connector Pinout	3-22
3-23	M68EC0x0IDP I.O Slot Pinout	3-25
3-24	Bus Parameters	3-28
3-25	Electrical Requirements	3-28
3-26	Receiver Characteristics	3-28
4-1	ROM68K Commands Set	4-3
5-1	MON68 Commands	5-4

APPENDIX A

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

SECTION 1

INTRODUCTION

This section provides general and detailed information on the M68EC000 family integrated development platform (IDP). Specific information for the applicable 68K-based CPU Module can be found in the accompanying 68K-based CPU Module user's manual. Please read both user's manuals before applying power to the unit.

The M68EC0x0IDP is a board set designed to provide a low-cost, yet flexible environment for developing both software and hardware for MC68EC0x0-based products. In addition, the platform provides the means for MC68EC0x0 processor and tool evaluation which enables users to properly select the microprocessor and associated tools for their next application.

The M68EC0x0IDP consists of an interchangeable 68K--based CPU Module as well as a generic M68EC0x0 motherboard designed to support each 68K--based CPU module. The M68EC0x0IDP also includes two software monitor programs (Integrated Systems ROM68k™ and Intermetric's MON68™). This configuration allows the user to take advantage of an entire suite of software features including tracing, assembling, and disassembling, that are offered by the two monitors.

The M68EC0x0 motherboard consist of the following features:

- CPU Slot for 68k—based CPU Module
- Five IDP I/O Expansion Slots
- 2 Mbytes of Socketed Page Mode DRAM Standard (User upgradeable to 5 or 8 M bytes)
- Two 32-Pin EPROM Sockets, up to 8 Mbits per Socket
- Battery-Backed Clock/Calendar with 2040 Bytes of SRAM
- Two RS232C Serial Ports Using MC68681 DUART
- 24-Bit Timer and Parallel Port Using the MC68230
- Seven-Segment Status LED
- Reset Switch, Abort Switch, and Power-On LED
- Two Power Connectors
- Two ROM Monitor/Debuggers

The 68K--based CPU Module consist of the following features:

- A MC68EC0x0, MC680X0 or MC683XX CPU
- 25-MHz MC68882 FPU (for the MC68EC020 and MC68EC030 CPU Modules Only)
- Address Translation Map External To The Processor
- IDP Bus Arbitration
- IDP Interrupt Prioritization
- Local Expansion Bus (Connects to M68EC0X0SRM performance analyzer)

The MC68EC0x0IDP bus provides a low-cost high-performance alternative to the VME and other popular embedded control buses. The major features of the MC68EC0x0IDP bus are:

- 32-Bit Data Bus with Byte Write Capability
- 28-Bit Address Bus with Slot Addressing
- 12.5 or 25-MHz Synchronous Transfers
- Two-Clock Burst Transfers
- Per Module Bus Request/Acknowledge and Interrupt Request/Acknowledge
- Compact 3.5" by 7.8" Module Size
- Reliable Four-Point Parallel Mounting using Low-Cost DIN Connectors

1.1 M68EC0X0IDP FUNCTIONALITY

The M68EC0x0IDP was designed to provide a flexible, expandable platform for MC68EC0x0 evaluation, software development, and hardware development support. The basic functionality of the MC68EC0x0IDP can be supplemented with a variety of standard M68EC0x0IDP bus products as well as customer defined modules.

1.1.1 Microprocessor Evaluation

With complete access to the unbuffered CPU signals, an optional high speed memory board (M68EC0X0SRM) connected to the local bus connector provides a mechanism for measuring the performance of the microprocessor. The memory on board can be configured such that the user can emulate different types of memory (instruction or data areas), and different sized devices (8, 16 or 32-bit widths). A software selectable feature allows the user to designate areas of memory to have specific numbers of wait states. In addition, dynamic bus sizing is supported by a feature that allows the user to specify different size configurations. A 32-bit timer on board provides profiling functionality. The user can monitor activities including the total number of times certain areas of memory are accessed. Specifically, the number of read accesses, the number of write accesses and the number of times the bus is utilized can all be monitored. Together, these features provide a powerful tool to assist the user in selecting optimal processor/memory combinations prior to the actual design while considering important factors such as price/performance tradeoffs.

1.1.2 Software Development

Two monitor routines reside in the ROM on the motherboard. Together, the monitors provide the user with a generous set of commands for software development and debug. The M68EC0x0IDP provides a convenient way to switch from one monitor to the other while maintaining a consistent development configuration.

The M68EC0x0IDP boots with Integrated Systems' ROM68k. This monitor features commands for reading and writing memory, viewing and/or changing CPU registers, assembling, disassembling and downloading code via serial port or network connection. In addition, the monitor can be configured to automatically load an OS from a host system and start it running anytime the target CPU is powered on or reset. Integrated Systems also offers XRAY+, a powerful source level debugger and pSOS+, a real-time operating system. XRAY+ is highly integrated with Integrated Systems' real time operating system allowing the user to debug on an operating system level as well as on a source level. Together, these two development tools offer a complete real-time operating system solution for the user. Both XRAY+ and pSOS+ are offered as options with the M68EC0x0IDP. pSOS+ is available through Motorola.

The second ROM monitor on board is "MON68" provided by Intermetrics. MON68 features commands to allow the user to set and display memory and registers. In addition, MON68 implements a trace function that allows the user to set breakpoints and examine traces of executed code. MON68 interfaces directly with Intermetrics' XDB source level debugger. XDB's unique features include actual target level access and direct control of program execution at the source statement or machine instruction level. The user can single step by source lines, machine instructions, and into and over procedure calls. XDB also has an assertion mechanism for testing a program under user defined conditions. XDB is offered as an option with the MC68EC0x0IDP and can be obtained through Motorola.

Other compatible operating systems, compilers as well as debuggers are offered by many third-party vendors giving the user the option to tailor the development system specifically for his or her needs.

1.1.3 Hardware Development

With an optional interface board and hardware connected to the 68K—based CPU Module local bus connector, the user may plug directly into the CPU socket of the target system. The re-mapping features of the M68EC0x0IDP allow the user to map out the portions of M68EC0x0IDP memory that conflict with target memory. Bus arbitration and interrupt control may be disabled on the M68EC0x0IDP, allowing the target system to be responsible for those functions.

1.2 SUPPORTING DOCUMENTATION

Users are encouraged to refer to the schematic diagrams for detailed data on the hardware design.

For information on the MC68EC000 CPU, please refer to the *M68000 Family User's Manual*, Motorola order number M68000UM/AD Revision 8.

For information on the MC68EC020 CPU, please refer to the *M68020 32-Bit Embedded Controller User's Manual*, Motorola order number M68020UM/AD.

For information on the MC68EC030 CPU, please refer to the *MC68EC030 32-Bit Embedded Controller User's Manual*, Motorola order number MC68EC030UM/AD.

For information on the MC68EC040 CPU, please refer to the *M68040 User's Manual*, Motorola order number M68040UM/AD Revision 2.

For information on the MC68882 FPU, please refer to the *MC68881/882 Floating-Point Coprocessor User's Manual*, Motorola order number MC68881UM/AD.

For information on the MC68681 DUART, please refer to the *Dual Asynchronous Receiver/Transmitter*, Motorola order number MC68681/D.

For information on the MC68230 PI/T, please refer to the *Parallel Interface/Timer*, Motorola order number MC68230/D.

For information on the MK48T02 RTC, please refer to the *Memory Products Databook*, SGS-Thompson order code DBMEMORYST/1US.

For information on the MC88916 Clock Drivers, please refer to the *Timing Solutions*, Motorola order number BR1333/D.

1.3 VENDORS SUPPORTING IDP

The Major leading compiler and real time O/S vendors have supported the IDP board. A user can purchase an off the shelf compiler and real time O/S from the following vendors and make it work with the IDP board without any difficulty.

The compiler vendors are: Avocet Systems, BSO/Tasking, Cygnus Support, Diab Data, Intermetrics Corp, Manx Software Systems, Microtec Research, Green Hills Inc., Production Language Corp., Sierra Systems, Software Development ssystems..

The real time O/S vendors are: A. T. Barrett & Assoc., Byte-BOS, Eyring Inc., Integrated Systems Inc., JMI Software Consultants, Microware Systems, Ready Systems, US Software, Wind River Systems.

SECTION 2 INSTALLATION AND OPERATION

This section provides information on the installation and operation of M68EC0x0IDP integrated development platform (IDP).

CAUTION

The M68EC0x0IDP uses electronic components that are sensitive to static electricity. When handling and/or installing boards, you must take care to prevent the components from being damaged by static currents. Always work in an area of low static electricity. Connect an anti-static bracelet to a grounded surface to prevent static discharge.

2.1 UNPACKING

The M68EC0x0IDP is packaged using anti-static materials. Upon receipt, inspect the board for missing, broken, or damaged components as well as for physical damage to the printed circuit board. Report any damage immediately to the carrier or their authorized agent.

2.2 EQUIPMENT REQUIREMENTS

Certain equipment and materials are required to use the IDP. The following items are included:

- M68EC0x0 IDP Motherboard
- Applicable 68K-based CPU Module
- 2-Foot, Dual Connector, Power Supply Cable
- 7-Foot, 6-Conductor, Dual-Terminated RJ-11 Cable For Serial Port
- RJ-11 to DB-25 Female Adapter

Items required but not included are:

- RS-232 ASCII Terminal or Host Computer with RS-232 Serial Port
- 5 V @ 3-10 Amps Power Supply

2.3 ASSEMBLY AND POWER-ON

Perform the following steps to assemble and power-up the M68EC0x0IDP. Refer to **Section 3 Hardware Configuration** as needed for terminal and connector locations.

1. Attach standoffs (4 on each side) to the M68EC0x0 IDP motherboard with screws and sleeves included in plastic pack.
2. Place the M68EC0x0IDP on a firm flat work surface.
3. Ensure that the power supply is turned off and attach the pig-tailed end of the red/black power supply cable to a 5-volt, 3 to 10 amp power supply. An M68EC0X0IDP motherboard and a 68k CPU Module will require upto 3 amps. Additional cards on the slot connectors and the CPU module local bus will require additional current.
4. Attach the two keyed Molex connectors of the other end of the power supply cable into each "male" Molex connector marked "POWER" (P4 and P3) on the M68EC0x0 motherboard.
5. In order to connect the M68EC0x0IDP to your host computer running a terminal emulator or terminal, the RJ-11/DB25 adapter will have to be properly wired. Refer to **2.4 ASCII Terminal/Host Computer Connection** for additional details.
6. Attach the RS-232 cable with RJ-11 connector to the M68EC0x0 motherboard in the RJ-11 adapter marked SIO 0 (P7). SIO 1 is not polled in hardware and therefore requires a user defined device driver in order to access.
7. Attach the other end of the RS-232 cable to the RJ-11/DB25 adapter. The RJ-11/DB25 connector will then attach to the communication port of your host computer or terminal.
8. After all connections are made, it is important to make sure that the terminal or the host running the terminal emulator application is set to transmit at 9600 baud, 8 data bits, 1 stop bit and no parity bit. If a host computer is being used, there should be some mechanism within the terminal emulator application that allows the transmit parameters to be configured. To set the baud rate, the HC command is used (see section 4). The baud rates available for use by the monitors is 9600, 19200, or 38400 baud. The configured baud rate takes effect when the RESET button is pushed. Upon reset, the LED displays the baud rate for 1 second. A "1" represents 19200, "9" represents 9600, and "3" represents 38400. If the user wishes to change the baud rate back to 9600 without going through the HC command, press the ABORT button during the 1 second window in which the baud rate appears on the LED display. The monitor then boots up at 9600 baud.
9. Turn on the power supply.
10. The following prompt should appear on the terminal.

```

Copyright 1993 Motorola Inc. All rights reserved
Copyright 1993 Software Components Group. All rights reserved
IDP ROM Version 3.0
Processor Type is: M68040
DRAM Size: 2 Megs

ROM68K :->

```

11. If the BI command is executed at the ROM68K prompt, Intermetrics™ MON68 will be invoked and the following screen will be displayed:

```

SmartMON      Version: M68EC40          version 2.5.0

Copyright (c) EMBEDDED SUPPORT TOOLS CORPORATION 1992
>

```

NOTE

The BI command is explained in **Section 4 ROM68K Configuration and Command Set** which describes the transition from ROM68K to the MON68; however, the M68EC0x0IDP requires a reset to transition back to ROM68K from the MON68.

2.4 ASCII TERMINAL/HOST COMPUTER CONNECTION

Due to the variety of terminals and host computers, three examples are given in Figure 2-1. The IBM-PC, DEC VT-220 and the Macintosh™ use a male DB-25 for the communications port. Figure 2-1 and Table 2-1 show the proper connection using the included RJ-11 to DB-25 adapter. For other applications, refer to the information supplied with the terminal or host computer to determine the style of connector and its appropriate connections.

NOTE

The IBM-PC requires a terminal emulation software program, such as ProComm™, Flashlink™, Windows™ 3.0 Terminal, PC-VT, etc. Connecting to a Macintosh requires terminal emulation software and a modem cable.

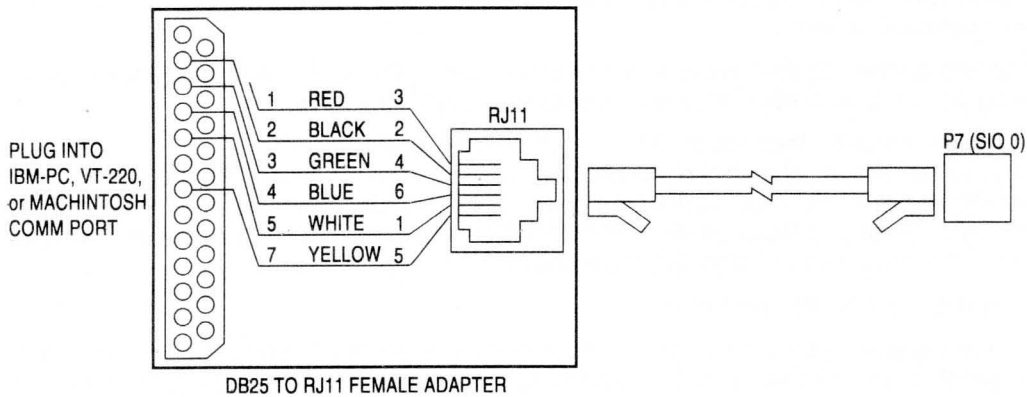


Figure 2-1. PC, DEC VT-220 , or Macintosh Connection

Table 2-1. PC, VT-220, and Macintosh RJ-11/DB25 Connections

Pin DB25	Color	Pin RJ11	Mnemonic	Description
1	Red	3	GND	Do Not Connect
2	Black	2	RXD	Receive Data (From Terminal)
3	Green	4	TXD	Transmit Data (To Terminal)
4	Blue	6	CTS	Clear To Send (From Terminal)
5	White	1	RTS	Request to Send (to Terminal)
7	Yellow	5	GND	Ground

2.5 SOFTWARE INTERFACE

Two unique ROM monitors reside on the M68EC0x0IDP. Together, these monitors provide a complementary command set allowing the user to assemble, disassemble, trace, read and modify memory and registers. The M68EC0x0IDP "powers up" with ROM68K; however, the M68EC0x0IDP provides the means to transition to and operate in the MON68 environment without compromising the configuration of the M68EC0x0IDP. Refer to **Section 4 ROM68K** and **Section 5 MON68** for detailed information on the ROM monitors. To set the baud rate, the HC command is used (see section 4). The baud rates available for use by the monitors is 9600, 19200, or 38400 baud. The configured baud rate takes effect when the RESET button is pushed. Upon reset, the LED displays the baud rate for 1 second. A "1" represents 19200, "9" represents 9600, and "3" represents 38400. If the user wishes to change the baud rate back to 9600 without going through the HC command, press the ABORT button during the 1 second window in which the baud rate appears on the LED display. The monitor then boots up at 9600 baud.

2.6 TROUBLESHOOTING

If any problems are encountered during start-up, please check the following items for help in solving the problem:

1. IDP board sparks, shorts out or smokes.

Power down the IDP and remove the power connector. Ensure that all socketed parts (Oscillator [Y1], ROMs [U41, U42], DRAM [U18 to U33] and control logic [U11 and U12 on IDP motherboard; various U numbers on CPU module]) are firmly seated, correctly oriented and no pins are bent. Ensure the CPU module is firmly connected to the IDP motherboard. Ensure all jumpers (J1 through J9) are in the correct position (Refer to **Chapter 3** Hardware configuration for jumper settings). Remove any cards in motherboard slots or on the CPU module local bus. Inspect the IDP motherboard and CPU module for any extraneous material, wires, etc. Ensure the power supply is supplying the correct voltage (4.75V to 5.25 V) and current (3 to 10 amps). An M68EC0X0IDP motherboard and a 68k CPU Module will require upto 3 amps. Additional cards on the slot connectors and the CPU module local bus will require additional current.

If everything appears to be correct, plug the power connector back in and apply power again. be ready to quickly power down if the IDP sparks, shorts out or smokes again.

If the problem persists, then fax to "Attention: IDP Technical Support" at 512 - 891 - 4568. Please have the serial number of both the M68EC0X0IDP motherboard (located at the bottom right; next to the DRAM) and the CPU module (located on the CPU module), the revision number and code of each control logic device and ROM on the M68EC0X0IDP and CPU module (printed on the label of the device) and a description of what is happening on power-up.

2. IDP board does not respond at all.

Check that the green power-on LED (D1) next to P3 is illuminated. If not, check to ensure that your power supply is plugged in, turned on, and supplying the correct voltage (4.75 V to 5.25 V) and current (3 to 10 amps). An M68EC0X0IDP motherboard and a 68k CPU Module will require upto 3 amps. Additional cards on the slot connectors and the CPU module local bus will require additional current.

If the green power LED is on check the seven segment LED (U48) diagnostic delay at the top end of the board between P3 and P4. The seven segment LED will display a number briefly on reset and then it will be blank. The number displayed depends upon the baud rate; 9 for 9600, 1 for 19200, and 3 for 38400 baud. Power down the IDP and remove any cards in motherboard slots or on the CPU module local bus and apply power again.

Press and release the reset button (SW1). This should cause the seven segment LED to operate as described above. If the LED is not momentarily displaying 9, 1 or 3, shield the seven segment LED from light, press and release the reset button (SW1).

If no number is displayed or any number is continuously displayed, then the M68EC0X0IDP failed the onboard diagnostics. Ensure that all socketed parts (Oscillator [Y1], ROMs [U41, U42], DRAM [U18 to U33] and control logic [U11 and U12 on IDP motherboard; various U numbers on CPU module]) are firmly seated, correctly oriented, the correct part and no pins are bent. Ensure that DRAM 0 (U18 to U25) has the same amount of memory as DRAM 1 (U26 - U33). Ensure that both banks of DRAM (DRAM 0 and DRAM 1) are populated. Ensure the CPU module is firmly connected to the IDP motherboard. Ensure all jumpers (J1 through J9) are in the correct position (Refer to **Chapter 3** Hardware Configuration for jumper settings). Ensure the CPU module can operate at the frequency set by the jumpers (J1 and J3) and the oscillator (Y1) value.

If the seven segment LED and/or power LED still does not respond as described above, then fax to "Attention: IDP Technical Support" at 512 - 891 - 4568. Please have the serial number of both the M68EC0X0IDP motherboard (located at the bottom right; next to the DRAM) and the CPU module (located on the CPU module), the revision number and code of each control logic device and ROM on the M68EC0X0IDP and CPU module (printed on the label of the device) and a description of what is working and failing on power-up.

3. Board shows power is on and seven segment LED momentarily flashes 9, 1 or 3 but the terminal or host computer is showing nothing.

Ensure that the serial port cable is wired correctly, plugged into P7 of the M68EC0x0IDP, and plugged into the appropriate communications port of the terminal or host computer. Ensure the software of the terminal or host computer is accessing the correct port. The most common problems are being on the wrong port of the terminal or host computer and having the RJ11 to DB25 adapter incorrectly wired. Ensure that the terminal or host computer is plugged in and powered on.

4. Terminal or host computer displays garbage.

Check the terminal baud rate, bits/character, stop bits and parity settings (9600 baud, 8 data bits, 1 stop bit and no parity bit). Check the baud rate of the IDP by pushing the reset button (SW1). Ensure that the serial port cable is wired correctly, plugged into P7 of the M68EC0x0IDP, and plugged into the appropriate communications port of the terminal or host computer. Ensure the software of the terminal or host computer is accessing the correct port and settings (9600 baud, 8 data bits, 1 stop bit and no parity bit).

If the IDP baud rate is not supported by your terminal or host computer, upon reset the seven segment LED displays the baud rate for 1 second. A "1" represents 19200, "9" represents 9600, and "3" represents 38400. To change the baud rate back to 9600 without going through the HC command, press the ABORT button during the 1 second window in which the baud rate appears on the LED display. The monitor then boots up at 9600 baud. Set your terminal or host computer to 9600 and try again. If your terminal or host computer does not support 9600 baud, then a different monitor with appropriate baud rate should be placed in the ROM 1 socket [U42] (Refer to **Chapter 3 Hardware Configuration** for details on how to install a ROM for boot up).

If the terminal or host computer is still displaying only garbage, then fax to "Attention: IDP Technical Support" at 512 - 891 - 4568. Please have the serial number of both the M68EC0x0IDP motherboard (located at the bottom right; next to the DRAM) and the CPU module (located on the CPU module), the revision number and code of each control logic device and ROM on the M68EC0x0IDP and CPU module (printed on the label of the device) and a description of what is working and failing on power-up.

5. Data is corrupted.

If everything is working, except data placed in user DRAM space becomes corrupted the problem can be conflicts with system memory use, system memory checking, or operating the M68EC0x0IDP system out of specifications. The system DRAM space (The M68EC0x0 IDP system reserves \$0000 0000 to \$0003 FFFF). The system memory checking setting is viewed and configured with the HC command (see section 4). The M68EC0x0IDP system specifications are in section 3. Both banks of DRAM (DRAM0 and DRAM 1) should be populated with MCM54400AZ70 or MCM524256AZ70 (or their equivalent).

6. Downloading data doesn't work, but other terminal/host communication works.

The download command (see section 4) will operate at the baud rate the system monitor is operates. In some cases, this may work fine for communicating to the terminal or host, but the interface can be affected during downloads. The solution is to switch to a slower baud rate (which will slow down the download) or to change the terminal or host computer settings to insert a delay after every line (which also slows the download, but not as much). When using the M68EC0x0SRM board, the MP command must first be used to remap the memory (this is done for you by the M68EC0x0SRM software).

7. Data is corrupted.

If everything is working, except data placed in user DRAM space becomes corrupted the problem can be conflicts with system memory use, system memory checking, or operating the M68EC0x0IDP system out of specifications. The system DRAM space (The M68EC0x0 IDP system reserves \$0000 0000 to \$0003 FFFF). The system memory checking setting is viewed and configured with the HC command (see section 4). The M68EC0x0IDP system specifications are in section 3. Both banks of DRAM (DRAM0 and DRAM 1) should be populated with MCM54400AZ70 or MCM524256AZ70 (or their equivalent).

SECTION 3 HARDWARE CONFIGURATION

This section describes the hardware configuration, bus specifications, bus timing, memory mapping, and mechanical specifications for Rev. 3.0 M68EC0X0IDP motherboards. Figure 3-1 shows an isometric drawing of the M68EC0X0IDP motherboard.

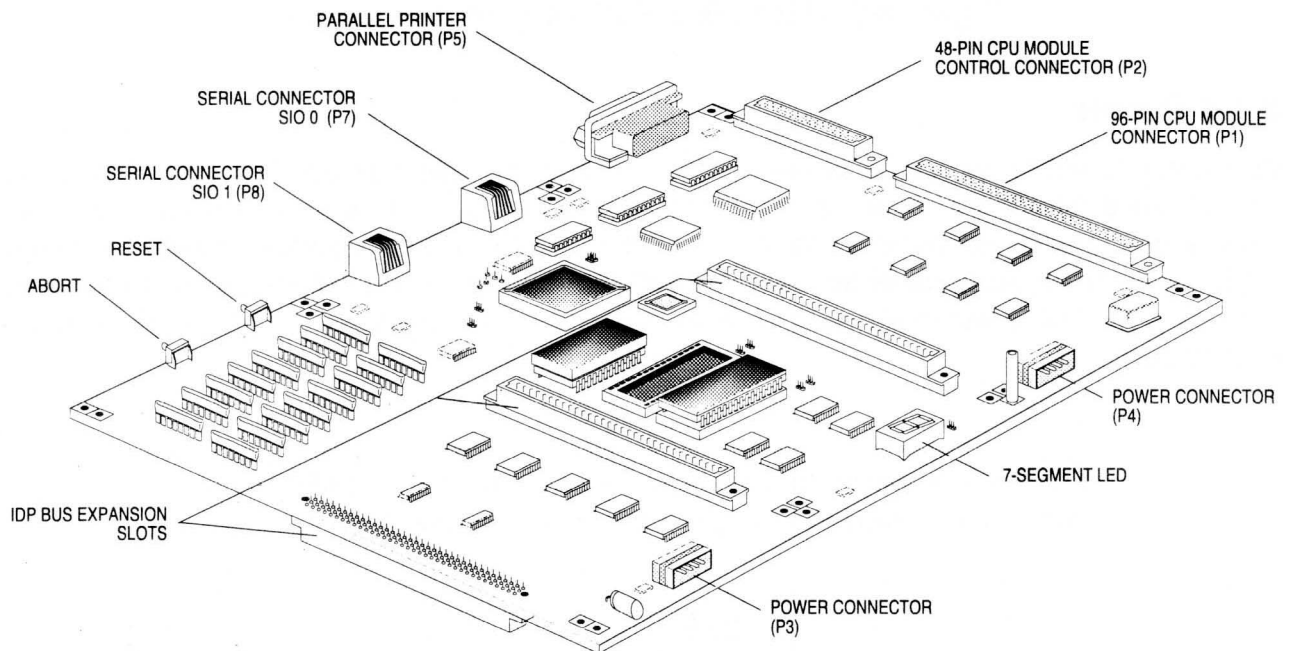


Figure 3-1. M68EC0X0IDP Motherboard

3.1 CONNECTOR PINOUTS

The following paragraphs describe the connector pinouts for power, serial input/output, parallel input/output and jumpers. Figure 3-2 shows the approximate locations of the various connectors on the M68EC0X0IDP motherboard.

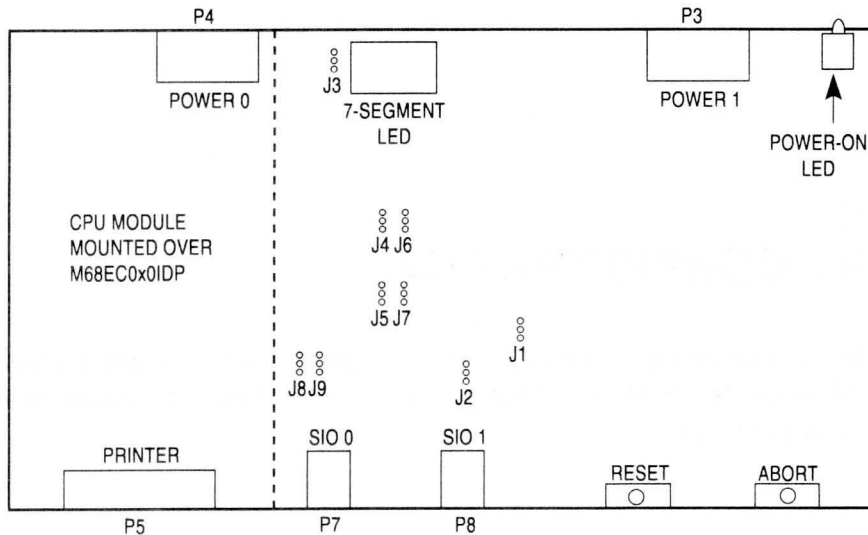


Figure 3-2. M68EC0x0 Motherboard Connectors

3.1.1 Power

Connectors P3 and P4 are used to provide +5 V and ground to the M68EC0X0IDP. These connectors (see Figure 3-3 and Table 3-1) are commonly used for 5.25" floppy disks as well as 5.25" hard drives. Most any power supply used for disk purposes can be used with the M68EC0X0IDP providing it meets the current requirements. The current required for the MC68EC0X0IDP depends on the CPU module and I/O slot cards installed. A M68EC040 system with no I/O slots will require about 3 amps. A fully loaded system will require up to 10 amps.

NOTE

Pin 1 on connectors P3 and P4 can be identified on the M68EC0x0IDP motherboard silkscreen by the pin numbers for pins 1 and 4.

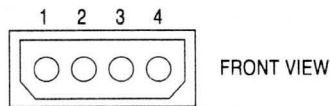


Figure 3-3. P3/P4 Connectors

Table 3-1. P3/P4 Connector Pinout

Pin	Description
1	No Connect (OK if connected to +12 V)
2	Ground
3	Ground
4	+5 V

3.1.2 Serial Input/Output

Connector P7 is for serial channel 0, and P8 is for serial channel 1. They connect to telephone style RJ-11, 6 conductor connectors. serial channel 0 (P7) is the channel used by the on board ROM monitors. Four-wire conductor cable and connectors may be used if hardware handshaking (RTS and CTS) is not a requirement. Pin 1 of P7/P8 is the left most pin when facing the connector. If four-wire conductor cable is used, pins 2,3,4 and 5 would be connected. Refer to Figure 3-4 and Table 3-2.

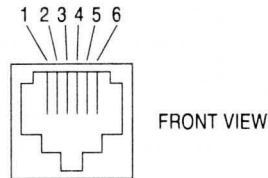


Figure 3-4. P7/P8 Connectors

Table 3-2. P7/P8 Connector Pinout

Pin	Mnemonic	P7 MC68681 Signal	P8 MC68681 Signal	Description
1	RTS	OP0	OP1	Request to Send (to Terminal)
2	RXD	RxDA	RxDB	Receive Data (from Terminal)
3	GND	—	—	Ground
4	TXD	TxDA	TxDB	Transmit Data (to Terminal)
5	GND	—	—	Ground
6	CTS	IP0	IP1	Clear to Send (from Terminal)

Connectors P7 and P8 each have an unpopulated shadow connector (P10 shadows P7, P9 shadows P8). The signals on the RJ-11 connector are also available on the shadow connector. Pin 1 and pin 2 of P10/P9 is identified on the M68EC0X0IDP silkscreen. Figure 3-5 shows the pin arrangement of P10/P9. Table 3-3 describes the pinout of P10/P9.

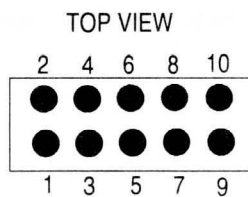


Figure 3-5. P10/P9 Connectors

Table 3-3. P10/P9 Connector Pinout

Pin	Mnemonic	P10 MC68681 Signal	P9 MC68681 Signal	Description
1	NC	–	–	No Connection on M68EC0X0IDP
2	RXD	RxDA	RxDB	Receive Data (from Terminal)
3	TXD	TxDA	TxDB	Transmit Data (to Terminal)
4	NC	–	–	No Connection on M68EC0X0IDP
5	GND	–	–	Ground
6	NC	–	–	No Connection on M68EC0X0IDP
7	RTS	OP0	OP1	Request to Send (to Terminal)
8	CTS	IP0	IP1	Clear to Send (from Terminal)
9	NC	–	–	No Connection on M68EC0X0IDP
10	NC	–	–	No Connection on M68EC0X0IDP

3.1.3 Parallel Input/Output

The connector P5 provides bidirectional parallel access to the M68EC0X0IDP motherboard. P5 connects to DB-25 male connectors. Pin 1 is the upper-rightmost pin and pin 25 is the lower-leftmost pin on P5 as you face the connector. Pin 1 is also identified on the M68EC0X0IDP silkscreen with a number and triangle behind the connector. The direction of the signals on P5 depends on the setting of jumper J8/J9 (Marked as PC and LPT on M68EC0X0IDP silkscreen). Jumpers J8 and J9 should always be set the same (i.e. both connecting pin 1 and pin 2 of their jumper or pin 2 and pin 3 of their jumper). Figure 3-6 shows the pin arrangement of P5. Table 3-4 describes the pinout of P5.

NOTE

Signals that are overlined are considered active low. Signals without an overline are considered active high. For example, \overline{STB} is active low (<0.8 V), D0 is active high (>2.0 V).

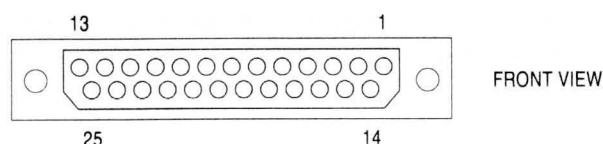


Figure 3-6. P5 Connector

Table 3-4. P5 Connector Pinout

Pin	Mnemonic	P8/P9 = PC	P8/P9 = LPT	MC68230 Signal	Description
1	STB	to M68EC0X0IDP	from M68EC0X0IDP	H1(PC) / H2(LPT)	Strobe
2	D0	from M68EC0X0IDP	to M68EC0X0IDP	Port A0	Data bit 0
3	D1	from M68EC0X0IDP	to M68EC0X0IDP	Port A1	Data bit 1
4	D2	from M68EC0X0IDP	to M68EC0X0IDP	Port A2	Data bit 2
5	D3	from M68EC0X0IDP	to M68EC0X0IDP	Port A3	Data bit 3
6	D4	from M68EC0X0IDP	to M68EC0X0IDP	Port A4	Data bit 4
7	D5	from M68EC0X0IDP	to M68EC0X0IDP	Port A5	Data bit 5
8	D6	from M68EC0X0IDP	to M68EC0X0IDP	Port A6	Data bit 6
9	D7	from M68EC0X0IDP	to M68EC0X0IDP	Port A7	Data bit 7
10	ACK	from M68EC0X0IDP	to M68EC0X0IDP	H2(PC) / H1(LPT)	Acknowledge
11	BUSY	to M68EC0X0IDP	from M68EC0X0IDP	Port B3	Busy
12	PAPER_ERR	to M68EC0X0IDP	from M68EC0X0IDP	Port B2	Paper Error
13	SLCT	to M68EC0X0IDP	from M68EC0X0IDP	Port B1	Select
14	AF	from M68EC0X0IDP	to M68EC0X0IDP	Port B5	Auto Feed
15	ERR	to M68EC0X0IDP	from M68EC0X0IDP	Port B0	Error
16	NC	-	-	-	No Connection on M68EC0X0IDP
17	SLCT_IN	from M68EC0X0IDP	to M68EC0X0IDP	Port B4	Init
18	GND	-	-	-	Ground
19	GND	-	-	-	Ground
20	GND	-	-	-	Ground
21	GND	-	-	-	Ground
22	GND	-	-	-	Ground
23	GND	-	-	-	Ground
24	GND	-	-	-	Ground
25	GND	-	-	-	Ground

Connector P5 has an unpopulated shadow connector P6. The signals on the DB-25 connector are also available on the shadow connector. Pin 1 and pin 2 of P6 is identified on the M68EC0X0IDP silkscreen. Figure 3-7 shows the pin arrangement of P6. Table 3-4 describes the pinout of P6.

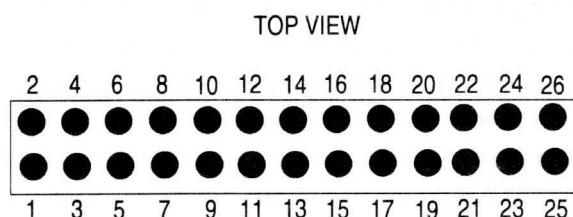


Figure 3-7. P6 Connector

Table 3-5. P6 Connector Pinout

Pin	Mnemonic	P8/P9 = PC	P8/P9 = LPT	MC68230 Signal	Description
1	STB	to M68EC0X0IDP	from M68EC0X0IDP	H1(PC) / H2(LPT)	Strobe
2	D0	from M68EC0X0IDP	to M68EC0X0IDP	Port A0	Data bit 0
3	D1	from M68EC0X0IDP	to M68EC0X0IDP	Port A1	Data bit 1
4	D2	from M68EC0X0IDP	to M68EC0X0IDP	Port A2	Data bit 2
5	D3	from M68EC0X0IDP	to M68EC0X0IDP	Port A3	Data bit 3
6	D4	from M68EC0X0IDP	to M68EC0X0IDP	Port A4	Data bit 4
7	D5	from M68EC0X0IDP	to M68EC0X0IDP	Port A5	Data bit 5
8	D6	from M68EC0X0IDP	to M68EC0X0IDP	Port A6	Data bit 6
9	D7	from M68EC0X0IDP	to M68EC0X0IDP	Port A7	Data bit 7
10	ACK	from M68EC0X0IDP	to M68EC0X0IDP	H2(PC) / H1(LPT)	Acknowledge
11	BUSY	to M68EC0X0IDP	from M68EC0X0IDP	Port B3	Busy
12	PAPER_ERR	to M68EC0X0IDP	from M68EC0X0IDP	Port B2	Paper Error
13	SLCT	to M68EC0X0IDP	from M68EC0X0IDP	Port B1	Select
14	AF	from M68EC0X0IDP	to M68EC0X0IDP	Port B5	Auto Feed
15	ERR	to M68EC0X0IDP	from M68EC0X0IDP	Port B0	Error
16	NC	-	-	-	No Connection on M68EC0X0IDP
17	SLCT_IN	from M68EC0X0IDP	to M68EC0X0IDP	Port B4	Init
18	GND	-	-	-	Ground
19	GND	-	-	-	Ground
20	GND	-	-	-	Ground
21	GND	-	-	-	Ground
22	GND	-	-	-	Ground
23	GND	-	-	-	Ground
24	GND	-	-	-	Ground
25	GND	-	-	-	Ground
26	NC	-	-	-	No Connection on M68EC0X0IDP

3.1.4 Jumper Settings

The M68EC0X0IDP has several jumpers which need to be properly set for correct operation. Each jumper is identified by a number (J1 to J9). Pin 1 is identified by double brackets (see Figure 3-8) on the M68EC0X0IDP silkscreen. Pin 2 is always the middle pin and pin 3 is the end pin with a single bracket on the M68EC0X0IDP silkscreen. In addition, each jumper has text associated with the jumper to indicate what happens with each position of the jumper. The jumpers are set by the location of the jumper plug. For all possible jumper settings, pin 1 or pin 3 is connected to pin 2 by the jumper plug. Every jumper should have a jumper plug. Figure 3-8 shows a sample jumper. Table 3-6 list the jumpers and describes their function.

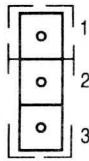


Figure 3-8. Jumper Diagram

Table 3-6. Jumper Description

Jumper	Pin 1 Text	Pin 3 Text	Description	Additional Information
J1	25 MHz	12.5 MHz	Selects biasing resistor for M68EC0X0IDP operation above 20 MHz (pin 1) or below 20 MHz (pin3)	Section 3.2.17
J2	2 OR 8 M DRAM	5 M DRAM	Configures M68EC0X0IDP board for 2 or 8 Megabytes of DRAM (pin1) or 5 Megabytes of DRAM (pin 2)	Section 3.2.17
J3	25 MHz	12.5 MHz	Selects clock for M68EC0X0IDP operation equal to half crystal oscillator (Y1) frequency (pin 1) or the same as crystal oscillator (Y1) frequency (pin3)	Section 3.2.17
J4	Vcc	A17	Selects signal applied to pin 30 of ROM 0 (U41); U41-pin 30 connected to Vcc (pin1) or A17 (pin 3). This is used to select different ROM sizes.	Section 3.2.8
J5	Vcc	A17	Selects signal applied to pin 30 of ROM 1 (U42); U42-pin 30 connected to Vcc (pin1) or A17 (pin 3). This is used to select different ROM sizes.	Section 3.2.8
J6	Vcc	A19	Selects signal applied to pin 1 of ROM 0 (U41); U41-pin 1 connected to Vcc (pin1) or A19 (pin 3). This is used to select different ROM sizes.	Section 3.2.8
J7	Vcc	A19	Selects signal applied to pin 1 of ROM 1 (U42); U42-pin 1 connected to Vcc (pin1) or A19 (pin 3). This is used to select different ROM sizes.	Section 3.2.8
J8	PC	LPT	Selects parallel pinout as sender (pin 1) or receiver (pin 3) Always set the same as J9.	Section 3.2.11
J9	PC	LPT	Selects parallel pinout as sender (pin 1) or receiver (pin 3) Always set the same as J8.	Section 3.2.11

3.2 M68EC0X0IDP MOTHERBOARD

The M68EC0X0IDP motherboard provides the core memory and I/O functions for the system as well as allowing expansion via the M68EC0X0IDP bus. In addition, the M68EC0x0IDP motherboard provides slot decoding plus interrupt and bus arbitration routing. A block diagram of the M68EC0x0IDP motherboard is shown in Figure 3-9.

NOTE

Signals that are overlined are considered active low. Signals without an overline are considered active high. For example, \overline{AS} is active low (<0.8 V), D0 is active high (>2.0 V).

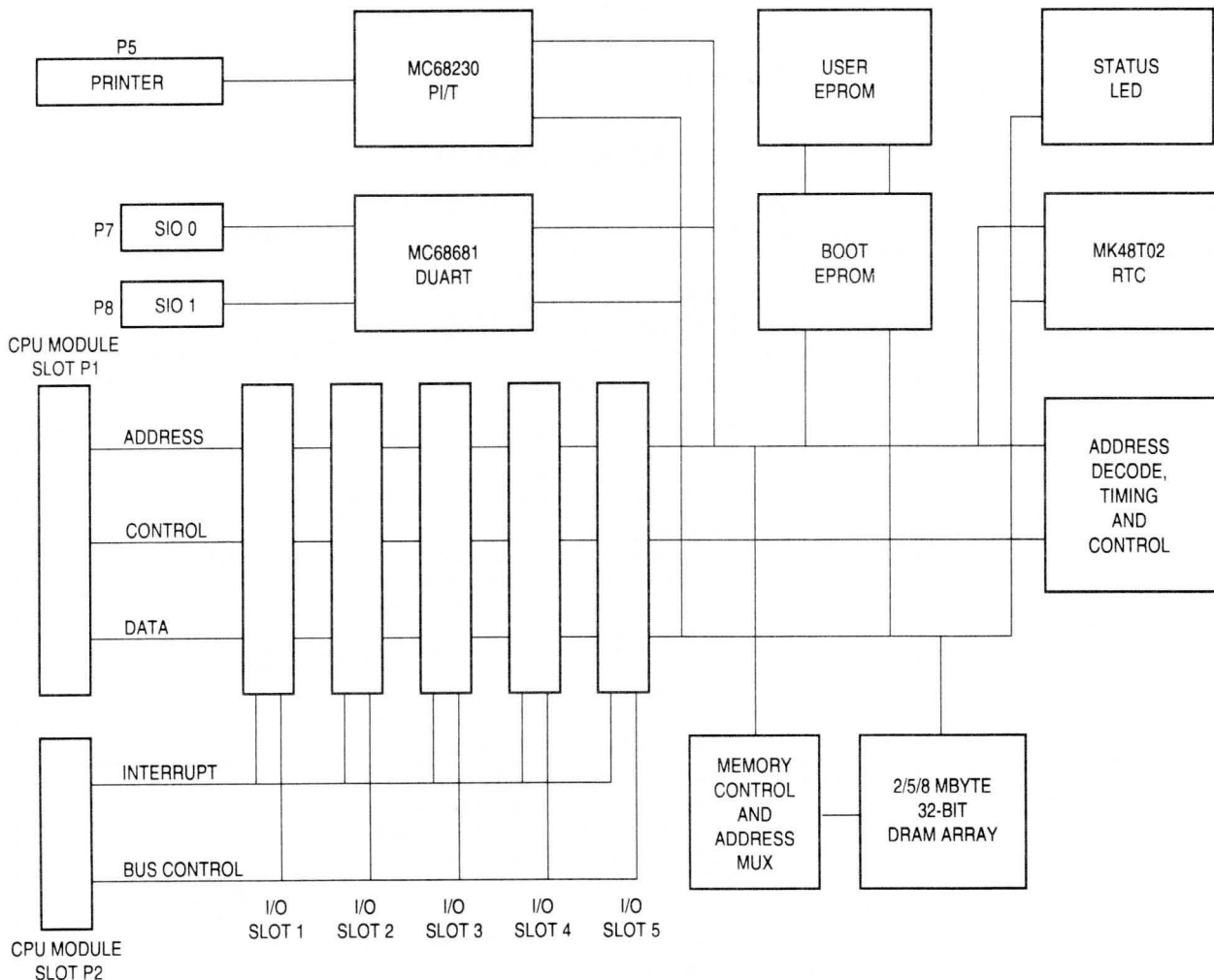


Figure 3-9. M68EC0x0 Motherboard Block Diagram

3.2.1 M68EC0X0IDP Bus Interface

The M68EC0X0IDP bus is a 32-bit data/28-bit address bus. Transfers take place synchronous to a 25-MHz or 12.5-MHz clock. The M68EC0X0IDP bus supports 32-bit transfers with byte write capability. All devices in the M68EC0X0IDP space are on 32-bit boundaries. The M68EC0X0IDP bus is independent of the CPU bus on the CPU module. A CPU module translates the CPU bus to the M68EC0X0IDP bus.

3.2.2 M68EC0X0IDP Bus Arbitration

One of the responsibilities of the 68K-based CPU Module is to perform bus arbitration for the M68EC0X0IDP bus. The \overline{BREQx} signals come into the 68K-based CPU Module where they are used to arbitrate the M68EC0X0IDP bus away from the CPU. When the CPU releases the M68EC0X0IDP bus, the appropriate M68EC0x0IDP \overline{BACKx} signal is generated, \overline{BGACK} is asserted and \overline{BDEN} is negated by the

CPU module. This three-states the M68EC0X0IDP data, address and control bus, allowing the M68EC0X0IDP I/O slot to become bus master and access the M68EC0X0IDP motherboard and other I/O slots. There is no way for the I/O slot bus master to access the CPU module. The CPU may continue to access CPU module resources (e.g. the CPU module local bus connector) while the M68EC0X0IDP bus has been arbitrated away. The bus arbitration function is disabled when the BRQDIS bit in control register zero (CREG0) on the CPU module is set (1). Table 3-7 outlines the bus request inputs and their priorities.

Table 3-7. Bus Request Inputs

Bus Request	Source	Priority
$\overline{\text{BREQ1}}$	I/O Slot 1	Highest
$\overline{\text{BREQ2}}$	I/O Slot 2	
$\overline{\text{BREQ3}}$	I/O Slot 3	
$\overline{\text{BREQ4}}$	I/O Slot 4	
$\overline{\text{BREQ5}}$	I/O Slot 5	Lowest

3.2.3 M68EC0X0IDP Interrupts

There are eight interrupts that are routed from the M68EC0x0 motherboard to the 68K-based CPU Module and one interrupt from the local bus: $\overline{\text{NMI}}$, $\overline{\text{LNMI}}$, $\overline{\text{TIRQ}}$, $\overline{\text{PIRQ}}$, and $\overline{\text{IREQ1}}$ – $\overline{\text{IREQ5}}$ (see Table 3-8). Circuitry on the 68K-based CPU Module prioritizes these interrupts and sends the appropriate interrupt request to the microprocessor. When the CPU responds with an interrupt acknowledge cycle, this circuit generates an $\overline{\text{IACK}}$ signal for each of the requests. The exceptions to this are $\overline{\text{NMI}}$ and $\overline{\text{LNMI}}$, both of which are acknowledged with $\overline{\text{AVEC}}$. All interrupts are disabled when the IRQDIS bit in control register zero (CREG0) on the CPU module is set (1).

Table 3-8. M68EC0X0IDP Interrupts

Interrupt	Source	Level	Priority	Response Signal	Response Cycle
NMI	Motherboard Abort Switch	7	Highest	–	AVEC cycle
$\overline{\text{LNMI}}$	CPU Module Local Bus NMI	7		–	AVEC cycle
$\overline{\text{TIRQ}}$	MC68230 Timer	6		$\overline{\text{TIACK}}$	IACK Cycle
$\overline{\text{PIRQ}}$	MC68681 Serial	5		$\overline{\text{PIACK}}$	IACK Cycle
$\overline{\text{PIRQ}}$	MC68230 Parallel Port	5		$\overline{\text{PIACK}}$	IACK Cycle
$\overline{\text{IREQ1}}$	I/O Slot 1	4		$\overline{\text{IACK1}}$	IACK Cycle
$\overline{\text{IREQ2}}$	I/O Slot 2	3		$\overline{\text{IACK2}}$	IACK Cycle
$\overline{\text{IREQ3}}$	I/O Slot 3	3		$\overline{\text{IACK3}}$	IACK Cycle
$\overline{\text{IREQ4}}$	I/O Slot 4	2		$\overline{\text{IACK4}}$	IACK Cycle
$\overline{\text{IREQ5}}$	I/O Slot 5	2	Lowest	$\overline{\text{IACK5}}$	IACK Cycle

3.2.4 M68EC0X0 Motherboard Memory Map

The M68EC0X0IDP motherboard resources are located at \$0000 0000 to \$00FF FFFF and duplicate shadows of the motherboard memory map are located at \$0C00 0000 to \$0CFF FFFF and \$0D00 0000 to \$0DFF FFFF. The duplicate shadows mean the same memory location can be accessed at any of three memory addresses. This allows for memory selectable CPU caching schemes (i.e. part of the DRAM is

cache inhibited and located at \$00XX XXX and part of the DRAM is cacheable and located at \$0CXX XXXX). CPUs with 24 bit address busses must use the memory remapping function of the CPU module to access \$0CXX XXXX and \$0DXX XXXX. The memory map is described in Table 3-9. The resident monitors use \$0n00 0000 to \$0n03 FFFF of DRAM. If the MP command (refer to **Section 4**) is never used, then the system will only use \$0000 0000 to \$0000 FFFF. DRAM memory from \$0n04 0000 to \$0n7F FFFF is available to the user. For more information on the allocation of the memory map within each segment (e.g. MC68681 register locations), refer to the appropriate following paragraphs.

Table 3-9. Motherboard Memory Map

Start Address		End Address		Description
\$0n00	0000	\$0n03	FFFF	DRAM (reserved for system use)
\$0n04	0000	\$0n7F	FFFF	DRAM (available to user)
\$0n80	0000	\$0n8F	FFFF	ROM0, Boot ROM Socket
\$0n90	0000	\$0n9F	FFFF	ROM1, User ROM Socket
\$0nA0	0000	\$0nAF	FFFF	MK48T02, Battery-Backed Clock/Calendar
\$0nB0	0000	\$0nBF	FFFF	MC68681, Serial I/O Controller
\$0nC0	0000	\$0nCF	FFFF	MC68230, Parallel Interface/Timer
\$0nD0	0000	\$0nDF	FFFF	STATUS LED, seven-segment LED display
\$0nE0	0000	\$0nEF	FFFF	Used To Access CPU Module Functions
\$0nF0	0000	\$0nFF	FFFF	No-Ack Space

n= \$0, \$C or \$D

3.2.5 M68EC0X0 Motherboard 68K-Based CPU Module Slot

The M68EC0X0 motherboard has one 68K-based CPU Module slot, using connectors P1 and P2. This slot may have any of several compatible 68K-based CPU Modules installed. A 96-pin connector carries the applicable M68EC0X0IDP bus signals; a 48-pin connector carries the 68K-based CPU Modules slot-specific signals.

NOTE

The pin identification convention between 96-pin DIN male and female connectors is inconsistent. A1 on the male connector matches up to A32 on the female 96-pin DIN connector. For this reason, there is confusion between the way that the pinout is listed in Table 3-6 and the pinout described by the schematics in Appendix C. When a female 96-pin DIN connector is attached to the male 96-pin DIN connector on the motherboard, the pinout listed in Table 3-6 is consistent with the pinout markings on the back of the female DIN connector. The schematics in Appendix C represent the pinout when looking at the male connector on the motherboard (with no other connectors attached) assuming that "A1" is the closest pin to the notched corner of the male connector. Pin A1 on all connectors on the motherboard (male and female) is located near the notch on the connector. This means the corresponding pin A1 of the connector plugging into a motherboard slot is on the same side but opposite end of the connector.

The pinout for the 96-pin P1 connector is listed in Table 3-10 and the pinout for the 48-pin P2 connector is listed in Table 3-11.

Table 3-10. M68EC0X0 Motherboard P1 Connector Pinout

Pin	Row A	Row B	Row C	Pin	Row A	Row B	Row C
1	VCC	BCLK	VCC	17	A4	A3	A2
2	GND	GND	GND	18	A7	A6	A5
3	NC	NC	NC	19	A10	A9	A8
4	D1	D0	NC	20	A13	A12	A11
5	D4	D3	D2	21	A16	A15	A14
6	D7	D6	D5	22	GND	GND	GND
7	D10	D9	D8	23	A19	A18	A17
8	D13	D12	D11	24	A22	A21	A20
9	D16	D15	D14	25	A25	A24	A23
10	D19	D18	D17	26	R/W	A27	A26
11	GND	GND	GND	27	BYTE1	BYTE2	BYTE3
12	D22	D21	D20	28	BRST	BRSTA	BYTE0
13	D25	D24	D23	29	TACK	CINH	AS
14	D28	D27	D26	30	NC	NC	RST
15	D31	D30	D29	31	GND	GND	GND
16	VCC	VCC	VCC	32	VCC	NC	VCC

Table 3-11. M68EC0X0 Motherboard P2 Connector Pinout

Pin	Row A	Row B	Row C
1	GND	GND	GND
2	BGACK	NOACK	CPUIO
3	BREQ1	BREQ2	BREQ3
4	VCC	VCC	VCC
5	BREQ4	BREQ5	NC
6	BACK1	BACK2	BACK3
7	BACK4	BACK5	NC
8	GND	GND	GND
9	NMI	TIRQ	PIRQ
10	REQ1	IREQ2	IREQ3
11	REQ4	REQ5	NC
12	VCC	VCC	VCC
13	BDEN	TIACK	PIACK
14	IACK1	IACK2	IACK3
15	IACK4	IACK5	CLKEN
16	GND	GND	GND

3.2.6 M68EC0X0 Motherboard I/O Slots

The M68EC0X0 motherboard has five slots for I/O expansion, using connectors P11 to P15. Each slot is designed to accept a single M68EC0X0IDP-compatible I/O card. The M68EC0X0 motherboard also contains address decoding circuitry to provide slot enable and ID ROM enable signals to each slot. CPUs with 24 bit address busses must use the memory remapping function of the CPU module to access I/O slots. Input/output cards such as Ethernet, SCSI, floppy and hard disk controllers may be designed to work with the M68EC0X0IDP. Contact Motorola for status of available I/O cards.

3.2.7 DRAM Array

The M68EC0X0 motherboard has two banks of 32-bit wide DRAM. The size of these banks may be 256K x 32 (1 Mbyte) or 1M x 32 (4 Mbytes) for a total of 2 Mbytes, 5 MBytes or 8 MBytes. The M68EC0X0IDP is shipped with both banks populated for a total of 2 Mbytes. Both banks of DRAM (DRAM0 and DRAM 1) should be populated with MCM54400AZ70 or MCM524256AZ70 (or their equivalent).

The performance of the M68EC0X0 motherboard DRAM is five clocks for a non-burst access. Burst accesses complete in five clocks for the first 32-bit word and two clocks for each subsequent access. The circuitry can transfer up to four 32-bit words per burst access. DRAM access may be delayed if a DRAM refresh is occurring when the DRAM access is initiated. The DRAM is refreshed about every 15 μ S off the 3.6864 MHz clock (Y2) of the MC68681. Changing the system clock oscillator (Y1) or the system frequency (J1 and J3) does not affect refreshing. The DRAM continue to be refreshed while the M68EC0X0IDP is in reset if the voltage and current remain within specifications.

3.2.7.1 DRAM MEMORY CONFIGURATION

The amount of DRAM installed in the board must match the jumper J2 setting. If the memory does not match the J2 setting or the jumper has no plug installed, the board may fail to work or the memory may be corrupted. For two or eight Mbytes of DRAM installed jumper J2 plug should be set to connect pin1 and pin 2 (see **paragraph 3.1.4** for jumper orientation description). This setting is marked by the words "2 or 8 M DRAM" on the M68EC0X0IDP silkscreen. For five Mbytes of DRAM installed jumper J2 plug should be set to connect pin 2 and pin 3. This setting is marked by the words "5 M DRAM" on the M68EC0X0IDP silkscreen. When installing additional memory, DRAM0 (U18 to U25) should always have as much or more memory as DRAM1 (U26 - U33).

3.2.7.2 DRAM MEMORY MAP DIVISION

The resident monitors use \$0000 0000 to \$0003 FFFF. \$0004 0000 to \$007F FFFF is available to the user. If the MP command (refer to **Section 4**) is never used, the resident monitors use only \$0000 0000 to \$0000 FFFF. Table 3-12 shows how the DRAM memory is subdivided based on the J2 setting. DRAM0 and DRAM1 indicate which bank of DRAM is used for that memory address. SHADOW0 and SHADOW1 indicate the address is shadowed onto DRAM0 and DRAM1. Whatever the setting, the memory logically appears from \$0000 0000 to \$001F FFFF (2 Mbytes), \$0000 0000 to \$004F FFFF (5 Mbytes) or \$0000 0000 to \$007F FFFF (8 Mbytes).

Table 3-12. Motherboard DRAM Memory Map Division

Start Address	End Address	2 Mbytes	5 Mbytes	8 Mbytes
\$0n00 0000	\$0n0F FFFF	DRAM0	DRAM0	DRAM0
\$0n10 0000	\$0n1F FFFF	DRAM1	DRAM0	DRAM1
\$0n20 0000	\$0n2F FFFF	SHADOW0	DRAM0	DRAM0
\$0n30 0000	\$0n3F FFFF	SHADOW1	DRAM0	DRAM1
\$0n40 0000	\$0n4F FFFF	SHADOW0	DRAM1	DRAM0
\$0n50 0000	\$0n5F FFFF	SHADOW1	SHADOW1	DRAM1
\$0n60 0000	\$0n6F FFFF	SHADOW0	SHADOW1	DRAM0
\$0n70 0000	\$0n7F FFFF	SHADOW1	SHADOW1	DRAM1

n= \$0, \$C or \$D

3.2.8 32-Pin EPROM Sockets

The M68EC0X0 motherboard has two 32-pin sockets for EPROM. ROM0 (U41), the boot ROM, provides the reset and stack vectors upon power-up. ROM1 (U42) is the user ROM. Control logic allows the ROMs to be executable by providing a full 32-bit long word or 16-bit word for each access. The width of the ROM access depends on the BYTE3 - BYTE0 settings. Table 3-13 shows the possible BYTE requests and the corresponding width of the ROM access.

Table 3-13. Motherboard EPROM Access Widths

BYTE3	BYTE2	BYTE1	BYTE0	D31 - D16	D15 - D0	Size
Asserted	Asserted or Negated	Negated	Negated	Valid	Invalid	Word
Asserted or Negated	Asserted	Negated	Negated	Valid	Invalid	Word
Negated	Negated	Asserted	Asserted or Negated	Invalid	Valid	Word
Negated	Negated	Asserted or Negated	Asserted	Invalid	Valid	Word
Asserted	Asserted	Asserted	Asserted	Valid	Valid	Long Word
Asserted	Asserted or Negated	Asserted	Asserted or Negated	Valid	Valid	Long Word
Asserted	Asserted or Negated	Asserted or Negated	Asserted	Valid	Valid	Long Word
Asserted or Negated	Asserted	Asserted	Asserted or Negated	Valid	Valid	Long Word
Asserted or Negated	Asserted	Asserted or Negated	Asserted	Valid	Valid	Long Word

The byte ordering is big-Endian. The access time for each long word is 18 clocks for a long word and 10 clocks for a word. The ROMs themselves must be 150 ns or faster. Big-Endian is defined as follows:

Most Significant			Least Significant
D31			D0
BYTE 3	BYTE 2	BYTE 1	BYTE 0

3.2.8.1 USER ROM AUTOBOOT

The standard M68EC0X0IDP ROM monitor provides an autoboot capability. After the power-on diagnostics have successfully completed, the software will look at the first long word of the user ROM (\$0090 0000). If the pattern \$4AFC 4AFC is present, the software will jump to the address pointed to by the third long word in the user ROM (\$0090 0008). This allows software developers to insert ROMs that override the normal boot sequence of the system.

3.2.8.2 EPROM CONFIGURATION

A configuration header allows the use of 512K, 1M, 2M, 4M or 8M-bit EPROM devices. Table 3-14 shows the jumper connection settings for different ROM sizes. J4 and J6 are associated with ROM0 (U41). J5 and J7 are associated with ROM1 (U42). Improperly setting the jumper plug or not installing the jumper plug may result in damage to the M68EC0X0IDP or the ROM. Each jumper has silkscreen text associated with it to assist in selecting the correct configuration. J4/J5 has silkscreen markings of Vcc for pin 1 and A17 for pin 3. J4-pin 2 is connected to ROM0 (U41) pin 30. J5-pin 2 is connected to ROM1 (U42) pin 30. J6/J7 has silkscreen markings of Vcc for pin 1 and A19 for pin 3. J6-pin 2 is connected to ROM0 (U41) pin 1. J7-pin 2 is connected to ROM1 (U42) pin 1.

Table 3-14. Motherboard EPROM Jumper Settings

ROM SIZE	J4/J5 SETTING	J6/J7 SETTING
512K-bit	Pin 1 to Pin 2	Pin 1 to Pin 2
1M-bit	Pin 1 to Pin 2	Pin 1 to Pin 2
2M-bit	Pin 2 to Pin 3	Pin 1 to Pin 2
4M-bit	Pin 2 to Pin 3	Pin 1 to Pin 2
8M-bit	Pin 2 to Pin 3	Pin 2 to Pin 3

Figure 3-10 shows the placement and orientation of the EPROM devices.

NOTE

Ensure that when installing the EPROM, that the notch on the EPROM matches with the notch on the socket.

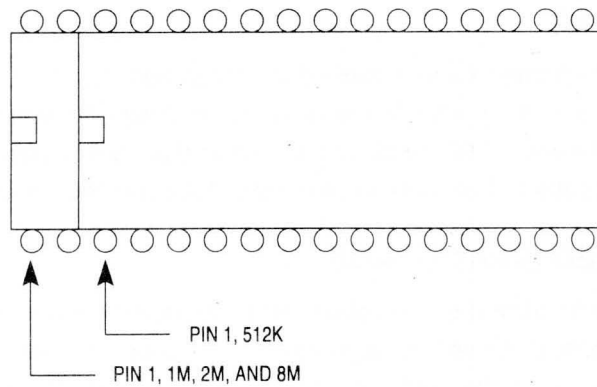


Figure 3-10. EPROM Configuration

3.2.9 Real-Time Clock/Calendar

An MK48T02 provides the M68EC0X0IDP with battery-backed clock/calendar functions. In addition, the device has 2040 bytes of battery-backed RAM. The MK48T02 is an 8 bit device located on the least significant byte (BYTE0) of the data bus. Bytes 3, 2 and 1 have nothing in them, so reads and writes to these locations are not valid and accesses to bytes 3, 2 or 1 may corrupt the data in MK49T02 on byte 0. The MK48T02 memory map repeats every \$2000 long words from \$00A0 0000 to \$00AF FFFF. The memory map for the MK48T02 is listed in Table 3-15.

Table 3-15. MK48T02 Memory Map

Address	Description
\$00A0 0003	Non-volatile RAM Byte 0
\$00A0 0007	Non-volatile RAM Byte 1
\$00A0 000B	Non-volatile RAM Byte 2
\$00A0 000F	Non-volatile RAM Byte 3
\$00A0 0013	Non-volatile RAM Byte 4
—	—
—	—
—	—
\$00A0 1FD3	Non-volatile RAM Byte 2037
\$00A0 1FD7	Non-volatile RAM Byte 2038
\$00A0 1FDB	Non-volatile RAM Byte 2039
\$00A0 1FDF	Non-volatile RAM Byte 2040
\$00A0 1FE3	Clock Control Register
\$00A0 1FE7	Clock Seconds Register
\$00A0 1FEB	Clock Minutes Register
\$00A0 1FEF	Clock Hours Register
\$00A0 1FF3	Clock Day Register
\$00A0 1FF7	Clock Date Register
\$00A0 1FFB	Clock Month Register
\$00A0 1FFF	Clock Year Register

3.2.9.1 CLOCK/CALENDAR

The MK48T02 provides a sophisticated battery-backed clock/calendar function. For detailed operation and programming information, users should refer to the *Memory Products Databook* (SGS-Thompson order code DBMEMORYST/1US). The MK48T02 needs to be initialized upon first powerup with an HC command (refer to **Section 4** for details on the HC command) and again if the real-time clock is used as a timer.

3.2.9.2 BATTERY-BACKED SRAM USAGE

To provide maximum functionality from the non-volatile RAM, guidelines have been established for its use. The RAM is divided into sections. Each section is assigned to one of the potential I/O slots. There are five I/O slots available. Table 3-16 outlines the memory map for the non-volatile RAM.

Table 3-16. Non-Volatile RAM Memory Map

Address Start	Address End	Size	Use
\$00A0 0000	\$00A0 01FF	128 Bytes	Motherboard and CPU Module Configuration Data
\$00A0 0200	\$00A0 02FF	128 Bytes	I/O Slot 1 Configuration Data
\$00A0 0400	\$00A0 04FF	128 Bytes	I/O Slot 2 Configuration Data
\$00A0 0600	\$00A0 06FF	128 Bytes	I/O Slot 3 Configuration Data
\$00A0 0800	\$00A0 08FF	128 Bytes	I/O Slot 4 Configuration Data
\$00A0 0A00	\$00A0 0AFF	128 Bytes	I/O Slot 5 Configuration Data
\$00A0 0C00	\$00A0 0CFF	128 Bytes	Reserved
\$00A0 0E00	\$00A0 0EFF	128 Bytes	Reserved
\$00A0 1000	\$00A0 10FF	128 Bytes	Reserved
\$00A0 1200	\$00A0 12FF	128 Bytes	Reserved
\$00A0 1400	\$00A0 1FDF	760 Bytes	Operating System Configuration Data

The first 32 bytes of an I/O slot's configuration data consist of a copy of the ID ROM data for the module installed in that slot. For a detailed description of the ID ROM format refer to **3.5.6 ID ROM Format**. System software compares this copy with the existing ID ROM at initialization. A mismatch indicates that the configuration of the slot has changed and any configuration data in the non-volatile RAM is invalid. When this occurs, the OS will copy the new ID ROM data into the configuration data area for that slot. The remaining 96 bytes may be used by the OS to hold driver-specific information.

3.2.10 Dual RS-232C Serial Ports

An MC68681 DUART is used to implement a dual RS-232C serial I/O interface. For detailed information on the MC68681 operation, refer to *MC68681* (MOTOROLA order number MC68681/D). The MC68681 is an 8 bit device located on the least significant byte (BYTE0) of the data bus. Bytes 3, 2 and 1 have nothing in them, so reads and writes to these locations are not valid and accesses to bytes 3, 2 or 1 may corrupt the MC68681 on byte 0. The MC68681 memory map repeats every \$40 long words from \$00B0 0000 to \$00BF FFFF. The register map for the MC68681 is outlined in Table 3-17.

Table 3-17. MC68681 Memory Map

Register Address	Read Function	Write Function
\$00B0 0003	Mode A	Mode A
\$00B0 0007	Status A	Clock Select A
\$00B0 000B	Reserved	Command A
\$00B0 000F	Receive Data A	Transmit Data A
\$00B0 0013	Input Port Change	Auxiliary Control
\$00B0 0017	Interrupt Status	Interrupt Mask
\$00B0 001B	C/T Upper Byte	C/T Reload Upper Byte
\$00B0 001F	C/T Lower Byte	C/T Reload Lower Byte
\$00B0 0023	Mode B	Mode B
\$00B0 0027	Status B	Clock Select B
\$00B0 002B	Reserved	Command B
\$00B0 002F	Receive Data B	Transmit Data B
\$00B0 0033	Interrupt Vector	Interrupt Vector
\$00B0 0037	Input Port (Real Time)	Output Port Configuration
\$00B0 003B	Start Counter Command	Set Output Port Bits
\$00B0 003F	Stop Counter Command	Reset Output Port Bits

The interrupt request output of the MC68681 is sent to the MC68230 H3 input. When responding to an MC68681 interrupt, the interrupt handler must clear the H3 interrupt condition in the MC68230 as well as clearing the MC68681 interrupt condition. The ROM68K polls the serial port, so therefore does not use interrupts for the MC68681. The MON68 monitor uses interrupts for serial communication, taking care of the MC68681 and MC68230. For detailed programming information, refer to the MC68681 data sheet and **Appendix B MC68230 Software Example**. The M68EC0x0 motherboard uses a pair of RJ-11 connectors, P7 and P8, for the serial interface. The resident monitors use port A (P7) as their communication port and do not use port B (P8). To set the baud rate, the HC command is used (refer to **Section 4**). The baud rates available for use by the monitors is 9600, 19200, or 38400 baud. The configured baud rate takes effect when the RESET button is pushed. Upon reset, the LED displays the baud rate for 1 second. A "1" represents 19200, "9" represents 9600, and "3" represents 38400. If the user wishes to change the baud rate back to 9600 without going through the HC command, press the ABORT button during the 1 second window in which the baud rate appears on the LED display. The monitor then boots up at 9600 baud. Figure 3-11 shows the orientation and pin numbering for both connectors. Table 3-18 gives a pin description.

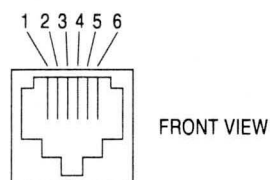


Figure 3-11. Serial I/O Connector Pinout

Table 3-18. P7/P8 Connector Pinout

Pin	Mnemonic	P7 MC68681 Signal	P8 MC68681 Signal	Description
1	RTS	OP0	OP1	Request to Send (to Terminal)
2	RXD	RxDA	RxDB	Receive Data (from Terminal)
3	GND	–	–	Ground
4	TXD	TxDA	TxDB	Transmit Data (to Terminal)
5	GND	–	–	Ground
6	CTS	IP0	IP1	Clear to Send (from Terminal)

3.2.11 Parallel Interface/Timer

An MC68230 provides a 24-bit timer and a parallel printer interface. For detailed information on the MC68230 operation, refer to *MC68230* (MOTOROLA order number MC68230/D). The MC68230 is an 8 bit device located on the least significant byte (BYTE0) of the data bus. Bytes 3, 2 and 1 have nothing in them, so reads and writes to these locations are not valid and accesses to bytes 3, 2 or 1 may corrupt the MC68230 on byte 0. The MC68230 memory map repeats every \$80 long words from \$00C0 0000 to \$00CF FFFF. The memory map for the MC68230 is outlined in Table 3-19.

Table 3-19. MC68230 Memory Map

Register Address		Register Function
\$00C0	0003	Port General Control
\$00C0	0007	Port Service Request
\$00C0	000B	Port A Data Direction
\$00C0	000F	Port B Data Direction
\$00C0	0013	Port C Data Direction
\$00C0	0017	Port Interrupt Vector
\$00C0	001B	Port A Control
\$00C0	001F	Port B Control
\$00C0	0023	Port A Data
\$00C0	0027	Port B Data
\$00C0	002B	Port A Alternate
\$00C0	002F	Port B Alternate
\$00C0	0033	Port C Data
\$00C0	0037	Port Status
\$00C0	003B	Reserved
\$00C0	003F	Reserved
\$00C0	0043	Timer Control Register
\$00C0	0047	Timer Interrupt Vector
\$00C0	004B	Reserved
\$00C0	004F	Counter Preload High Byte
\$00C0	0053	Counter Preload Middle Byte
\$00C0	0057	Counter Preload Low Byte
\$00C0	005B	Reserved
\$00C0	005F	Counter High Byte
\$00C0	0063	Counter Middle Byte
\$00C0	0067	Counter Low Byte
\$00C0	006B	Timer Status

The MC68681 (serial I/O) interrupt request output is not sent directly to the 68K-based CPU module. Instead, it is routed to the MC68230 H3 control input (internal wiring on motherboard). The MC68681 is programmed normally. Whenever the MC68230 is set up to generate an interrupt when the H3 line goes low. The interrupt handler for the serial ports must clear the H3 interrupt as well as the serial I/O interrupt. The resident MON68 monitor performs the programming needed to use the MC68681 in interrupt mode.

Support circuitry has been added to allow the MC68230 to perform a parallel printer port function. The pinout and signal assignment is compatible with the Centronics printer interface. Since the MC68230 is programmable, driver software is needed to implement the actual function. The MC68230 operates at 1/4 the oscillator frequency (Y2). This is 6.25 MHz for the standard 25 MHz oscillator. Changing the motherboard clock frequency does not affect the MC68230 clock frequency.

Figure 3-12 shows the orientation and pin numbering for the P5 printer connector .

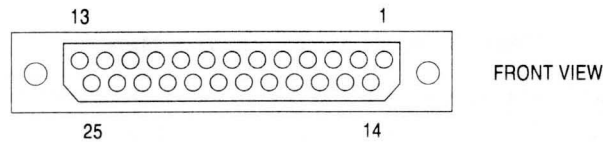


Figure 3-12. P5 Connector Pinout

Table 3-20 listed the connections between the P5 printer connector and the MC68230.

Table 3-20. MC68230 to P5 Printer Connections

MC68230 Signal	P5 Pin	Mnemonic	P8/P9 = PC	P8/P9 = LPT	Description
H1(PC) / H2(LPT)	1	STB	to M68EC0X0IDP	from M68EC0X0IDP	Strobe
Port A0	2	D0	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 0
Port A1	3	D1	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 1
Port A2	4	D2	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 2
Port A3	5	D3	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 3
Port A4	6	D4	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 4
Port A5	7	D5	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 5
Port A6	8	D6	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 6
Port A7	9	D7	from M68EC0X0IDP	to M68EC0X0IDP	Data bit 7
H2(PC) / H1(LPT)	10	ACK	from M68EC0X0IDP	to M68EC0X0IDP	Acknowledge
Port B0	15	ERR	to M68EC0X0IDP	from M68EC0X0IDP	Error
Port B1	13	SLCT	to M68EC0X0IDP	from M68EC0X0IDP	Select
Port B2	12	PAPER_ERR	to M68EC0X0IDP	from M68EC0X0IDP	Paper Error
Port B3	11	BUSY	to M68EC0X0IDP	from M68EC0X0IDP	Busy
Port B4	17	SLCT_IN	from M68EC0X0IDP	to M68EC0X0IDP	Init
Port B5	14	AF	from M68EC0X0IDP	to M68EC0X0IDP	Auto Feed
Port B0	15	ERR	to M68EC0X0IDP	from M68EC0X0IDP	Error
Port B6	–	CNTL_DIR	Program to 1	Program to 0	Control signals direction set by user
Port B7	–	DATA_DIR	-	-	Data Direction (1 = in, 0 = out)
Port C0	–	OE	-	-	P5 output enable (1 = disabled, 0 = enabled)
Port C1	–	H_OE	-	-	Handshake output enables (1 = disabled, 0 = enabled)

3.2.12 Status LED

A seven-segment LED display provides the user with status information. The standard ROM monitor uses the LED to display the state of the power-on diagnostic tests. The LED is controlled by a write-only 8-bit register located at \$00D0 0003. The eighth bit (D7) is not used and a 1 should always be written to it. The register is shadowed every long word for \$00DX XXXX. Each LED segment is controlled by their own bits. Writing a 1 to a bit turns the associated segment off; writing a 0 turns the segment on. The state of this register is unknown after reset.

The bit assignments for the register are as follows:

Bit	7	6	5	4	3	2	1	0
Segment	–	g	f	e	d	c	b	a

Table 3-21 list the values written to the register for different displays.

Table 3-21. LED Display Values

Display	Data
Blank	\$FF
0	\$C0
1	\$F9
2	\$A4
3	\$B0
4	\$99
5	\$92
6	\$82
7	\$F8
8	\$80
9	\$98

The segment locations are shown in Figure 3-13.

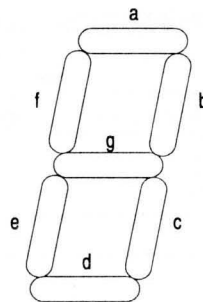


Figure 3-13. LED Segments

3.2.13 Reset Switch

A momentary contact, push-button switch, when pressed, allows the user to generate a hardware reset signal to the system. The reset is denounced through the MC88916 and an R-C network. The MC88916 will assert reset to the rest of the system for ~1000 M68EC0X0IDP clock periods after the later of the reset switch being released, the MC88916 phase lock loop achieving lock or the voltage being within specifications. In addition to the reset switch and MC88916, the MC34064 asserts a reset to the MC88916 if the voltage goes outside of the M68EC0X0IDP voltage specifications.

The reset switch also has two additional, unpopulated holes in the M68EC0X0IDP motherboard to allow the user to add remote reset capability. The holes are next to the reset switch and marked as TP6 ($\overline{\text{RESET}}$) and TP7(GND) on the M68EC0X0IDP motherboard silkscreen.

3.2.14 Abort Switch

A momentary contact, push-button switch, when pressed, allows the user to generate a non-maskable interrupt (NMI) to the installed 68K-based CPU module. The interrupt is autovectorred by the CPU module.

The abort switch also has two additional, unpopulated holes in the M68EC0X0IDP motherboard to allow the user to add remote reset capability. The holes are next to the abort switch and marked as TP8 ($\overline{\text{NMI}}$) and TP9(GND) on the M68EC0X0IDP motherboard silkscreen.

3.2.15 Power-On LED

A single LED display, when illuminated, indicates that power has been applied to the board.

3.2.16 Power Connectors

Two four-pin right-angle connectors, P3 and P4, are used to provide power to the M68EC0x0 motherboard. Figure 3-14 and Table 3-22 provide the orientation and pin numbering for both connectors.

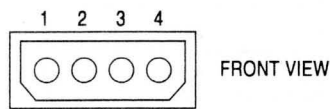


Figure 3-14. P16/P17 Connectors

Table 3-22. P16/P17 Connector Pinout

Pin	Use
1	No Connect (OK if connected to +12 V)
2	Ground
3	Ground
4	+5V

3.2.17 MC88916 CLOCK

Because the M68EC0X0IDP bus is a synchronous bus, the clock used on the motherboard, CPU modules and I/O slot cards must be aligned. This clock synchronization is achieved with the MC88916. The BCLK is routed to the CPU modules and I/O slots. On the motherboard, the CPU module and the I/O slots, a MC88916 uses the BCLK to create a local version of the clock that the M68EC0X0IDP is running at the M68EC0X0IDP bus frequency. This local copy of the clock will be within 3 ns of any other version of the M68EC0X0IDP clock created by another MC88916. The MC88916 takes the half frequency BCLK and provides a clock that is twice the frequency of the M68EC0X0IDP bus, three clocks that are equal to the frequency of the M68EC0X0IDP bus and a clock that is half the frequency of the M68EC0X0IDP bus.

The MC88916 requires a different external analog loop filter depending on the frequency of the M68EC0X0IDP bus (above or below 20 MHz). To allow the user to change the clock frequency of the M68EC0X0IDP, the CPU module provides a clock enable ($\overline{\text{CLKEN}}$) signal, the oscillator is socketed and two jumpers are provided. The MC88916 input clock must be at least 5 MHz, putting a minimum frequency of M68EC0X0IDP bus operation of 10 MHz.

3.2.17.1 JUMPER J1

Jumper J1 changes the external analog loop filter of the MC88916. Pin 1 of J1 is marked 25 MHz on the silkscreen. Pin 3 of J1 is marked 12.5 MHz on the silkscreen. These settings of J1 correspond to the frequency used for the M68EC0X0IDP bus. Connecting pin 1 and pin 2 of J1 with the jumper plug is for running the M68EC0X0IDP bus at frequencies above 20 MHz. Connecting pin 2 and pin 3 of J1 with the jumper plug is for running the M68EC0X0IDP bus at frequencies below 20 MHz. If the jumper plug is not installed on J1, the IDP will not function properly.

3.2.17.2 JUMPER J3

Jumper J3 changes whether the crystal oscillator (Y1) is divided by 2 or divided by 4 before becoming BCLK. Pin 1 of J3 is marked 12.5 MHz on the silkscreen. Pin 3 of J3 is marked 25 MHz on the silkscreen. These settings of J3 correspond to the division of the crystal oscillator to achieve the frequency of the M68EC0X0IDP bus. Connecting pin 1 and pin 2 of J3 with the jumper plug is for running the M68EC0X0IDP bus at frequency of the crystal oscillator frequency. Connecting pin 2 and pin 3 of J3 with the jumper plug is for running the M68EC0X0IDP bus at half the frequency of the crystal oscillator frequency. If the jumper plug is not installed on J3, the IDP will not function properly unless the clock enable overrides the motherboard oscillator. The setting of J3 does not affect the frequency of the MC68230.

3.2.17.3 OSCILLATOR Y1

The M68EC0X0IDP bus frequency is determined by the frequency of the crystal oscillator (Y1) socketed on the motherboard. Depending on the settings of J1, J2 and the signal $\overline{\text{CLKEN}}$, the M68EC0X0IDP bus will operate at equal to or half the frequency of the crystal oscillator frequency. The minimum input frequency of the MC88916 is 5 MHz. If the oscillator clock is enabled ($\overline{\text{CLKEN}}$ is 0), then the minimum frequency of the crystal oscillator (Y1) is 10 MHz. The crystal oscillator also determines the frequency of the MC68230. The MC68230 is clocked at one fourth the frequency of the crystal oscillator. The minimum frequency for the MC68230 is 2 MHz. If the crystal oscillator clock is disabled ($\overline{\text{CLKEN}}$ = 1), then the minimum frequency of the oscillator (Y1) is 8 MHz. The M68EC0X0IDP is designed to operate at frequencies upto 25 MHz, so the maximum frequency of the crystal oscillator (Y1) is 25 MHz.

3.2.17.4 $\overline{\text{CLKEN}}$

The CPU module asserts $\overline{\text{CLKEN}}$ to the motherboard through connector P2. If $\overline{\text{CLKEN}}$ is asserted ($\overline{\text{CLKEN}}$ = 0), then the BCLK to the CPU module through connector P1 is an input to the CPU module. If the $\overline{\text{CLKEN}}$ is negated ($\overline{\text{CLKEN}}$ = 1), then the BCLK to the CPU module through connector P1 is an output from the CPU module and will be equal to half the frequency of the M68EC0X0IDP bus. The M68EC0X0IDP I/O slot cards will always take the BCLK as an input, whether it is crystal oscillator (Y1) or the CPU module driving BCLK.

3.3 IDP BUS SPECIFICATION.

The following paragraphs provide the specification for the M68EC0x0IDP bus. This specification has been written to ensure complete compatibility between manufacturers of the M68EC0x0IDP I/O modules.

Synchronous Transfers

An MC88916 clock synchronizer is present on each I/O module to synchronize clocks to the bus clock (BCLK). This allows the M68EC0x0IDP bus to have a guaranteed clock skew of 3 ns or less.

Slot Decoding

Each I/O slot has its own slot enable ($\overline{\text{SLOT}}$) signal generated by the M68EC0X)IDP motherboards based on A27-A24, which reduces the address decoding overhead for the I/O module. The slot size is 16 Mbytes. The slots are located at \$0nXX XXXX, where n = the slot number and X can be any hex value.

ID ROM

An ID ROM enable ($\overline{\text{IDROM}}$) signal selects the on board ID ROM. This ROM allows software to determine the current configuration of the system. While the ID ROM space is 1 Mbyte, the ID ROM definition covers just the first 32 bytes of the ROM. The remaining space may be used as the board designer sees fit. Both $\overline{\text{SLOT}}$ and $\overline{\text{IDROM}}$ assert for accesses to the IDROM.

Individual Interrupt Control

Each slot has its own interrupt request ($\overline{\text{IREQ}}$) and interrupt acknowledge ($\overline{\text{IACK}}$) signals. No prioritization or acknowledge decoding is required of the I/O module, again significantly reducing the bus interface overhead. An interrupt acknowledge cycle is indicated to the I/O slot by the CPU module asserting $\overline{\text{IACK}}_n$ (where n = the I/O slot number) and $\overline{\text{AS}}$.

Individual Bus Control

Each slot has its own bus request ($\overline{\text{BREQ}}$) and bus acknowledge ($\overline{\text{BACK}}$) signals. These signals allow bus masters to access the M68EC0x0IDP bus without the need for complex bus arbitration circuitry.

Control Signal Pull-ups

The $\overline{\text{BREQ}}$, $\overline{\text{BACK}}$, $\overline{\text{IREQ}}$, $\overline{\text{IACK}}$, $\overline{\text{TACK}}$, $\overline{\text{BRSTA}}$, $\overline{\text{CINH}}$ and $\overline{\text{AS}}$ are pulled high with a 4.7K Ω resistor. $\overline{\text{RST}}$ is pulled high with a 1K Ω resistor. $\overline{\text{SLOT}}$ and $\overline{\text{IDROM}}$ are always actively driven by the M68EC0XoIDP motherboard.

Compact, Flexible Module Size

The slot size of an I/O module is 3.5" by 7.8". However, depending upon the slot used on the motherboard, I/O modules may be two (7.1" by 7.8") or even three (10.7" by 7.8") module widths in size.

Reliable Parallel Mounting

Unlike most buses, M68EC0x0IDP I/O modules mount parallel to the M68EC0x0 motherboard. This significantly reduces the need for mechanical strengthening of the system and also provides a very low profile for space critical applications.

3.3.1 I/O Slot Connector Pinouts

Each I/O slot has five signals unique to the slot ($\overline{\text{SLOT}}$, $\overline{\text{BREQ}}$, $\overline{\text{BACK}}$, $\overline{\text{IREQ}}$, and $\overline{\text{IACK}}$). The remaining signals are identical for each I/O slot. Figure 3-15 and Table 3-23 provide the pinout for the M68EC0x0IDP I/O slots.

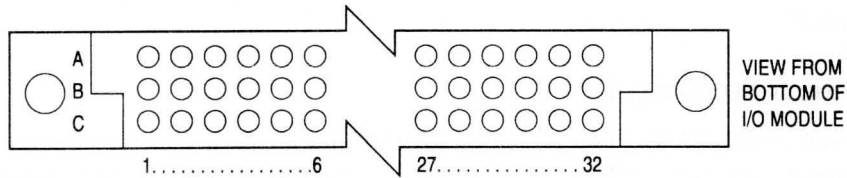


Figure 3-15. M68EC0x0IDP I/O Slot Connector

Table 3-23. M68EC0x0IDP I/O Slot Pinout

Pin	Row A	Row B	Row C	Pin	Row A	Row B	Row C
1	V _{CC}	IDROM	V _{CC}	17	V _{CC}	V _{CC}	V _{CC}
2	GND	GND	GND	18	D31	D30	D29
3	$\overline{\text{SLOT}}$	NC	$\overline{\text{RST}}$	19	D28	D27	D26
4	$\overline{\text{TACK}}$	$\overline{\text{CINH}}$	$\overline{\text{AS}}$	20	D25	D24	D23
5	$\overline{\text{BRST}}$	$\overline{\text{BRSTA}}$	$\overline{\text{BYTE0}}$	21	D22	D21	D20
6	BYTE1	BYTE2	BYTE3	22	GND	GND	GND
7	R/ $\overline{\text{W}}$	A27	A26	23	D19	D18	D17
8	A25	A24	A23	24	D16	D15	D14
9	A22	A21	A20	25	D13	D12	D11
10	A19	A18	A17	26	D10	D9	D8
11	GND	GND	GND	27	D7	D6	D5
12	A16	A15	A14	28	D4	D3	D2
13	A13	A12	A11	29	D1	D0	$\overline{\text{BREQ}}$
14	A10	A9	A8	30	$\overline{\text{BACK}}$	$\overline{\text{IREQ}}$	$\overline{\text{IACK}}$
15	A7	A6	A5	31	GND	GND	GND
16	A4	A3	A2	32	V _{CC}	BCLK	V _{CC}

3.3.2 Bus Signals

The following paragraphs provide a brief description of each M68EC0x0IDP bus signal. Each signal also has the direction indicated. If a single direction is indicated, then that signal is always unidirectional. If the signal is designated as an Input/Output, the signal is an input when the I/O slot is a slave and an output when the I/O slot is a master. If the signal is designated as an Output/Input, the signal is an output when the I/O slot is a slave and an input when the I/O slot is a master. The only exception to this is the reset and data bus which is bidirectional as both a slave and a master (data bus direction based on the R/ $\overline{\text{W}}$ and bus mastership status).

Bus Clock (BCLK; Input)

The M68EC0x0IDP I/O modules will synchronize to this totem pole bus signal and produce the required local module clock (SCLK). BCLK is one-half the frequency used by the system. For 25-MHz systems, BCLK is 12.5-MHz. The MC88916 clock synchronizes on the I/O modules double BCLK and create the phase-aligned 25-MHz clocks as well as a 50-MHz and 12.5-MHz clock. Refer to the MC88916 data sheet for more information on the clock synchronization. Note that all timing occurs in relation to the recovered clock SCLK.

Data Bus (D0–D31; Input/Output)

These 32 three-state, bidirectional lines are used for transferring data between the current bus master and the selected slave. All data transfers are on 32-bit boundaries. D0 is the least significant bit.

Address Bus (A0–A27; Input/Output)

These 26 three-state lines are driven by the current bus master to select one of the 64MB 32-bit words to transfer. A0 and A1 are implicitly encoded in the four byte enable lines ($\overline{\text{BYTE0}}$ – $\overline{\text{BYTE3}}$). A27 - A24 are used to address the motherboard (\$0, \$C or \$D) or I/O slots (\$1, \$2, \$3, \$4, \$5, or \$E). All other combinations of A27 - A24 are reserved. A2 to A23 are used to address resources on the motherboard or I/O slot.

Address Strobe ($\overline{\text{AS}}$; Input/Output)

This three-state signal is used to indicate that a transfer is taking place. Slave modules sample this signal on the rising edge of SCLK.

Transfer Acknowledge ($\overline{\text{TACK}}$; Output/Input)

This three-state signal is returned by the addressed slave device to indicate that data will be accepted or available on the next rising edge of SCLK.

Burst Request ($\overline{\text{BRST}}$; Input/Output)

This three-state signal indicates that the current bus master desires to perform a burst cycle.

Burst Acknowledge ($\overline{\text{BRSTA}}$; Output/Input)

This three-state signal is driven by the addressed slave to indicate that it will respond to a burst cycle.

Cache Inhibit ($\overline{\text{CINH}}$; Output/Input)

This three-state signal is driven by the addressed slave to indicate that the data should not be cached by the CPU module.

Read/Write ($\overline{\text{R/W}}$; Input/Output)

When high, this three-state signal indicates that the current bus master is reading data from the addressed slave. When low, the master is writing data to the slave.

Byte Enables ($\overline{\text{BYTE0}}$ – $\overline{\text{BYTE3}}$; Input/Output)

These four three-state signals are driven by the master to indicate which of the four bytes in the 32-bit word are being transferred. They allow the master to write selected bytes within a 32-bit word. On reads, at least the bytes requested must be provided (the other bytes not requested may or may not be valid).

Slot Enable ($\overline{\text{SLOT}}$; Input)

This signal is driven by the motherboard address decoder and enables the slot for transfers. This signal is qualified by $\overline{\text{AS}}$.

ID ROM Space ($\overline{\text{IDROM}}$; Input)

This signal is driven by the motherboard address decoder and indicates that the master is accessing the slot ID ROM space. It is qualified with $\overline{\text{SLOT}}$ and $\overline{\text{AS}}$.

Reset ($\overline{\text{RST}}$; Input/Output)

This open collector signal is used to reset the M68EC0X0IDP bus. It may be driven by the motherboard, CPU module or I/O slot. When asserted, the $\overline{\text{RST}}$ must remain asserted for at least 1000 M68EC0X0IDP clock periods.

Interrupt Request ($\overline{\text{IREQ}}$; Output)

This signal is driven by the I/O module to indicate to the CPU module that an interrupt condition exists. It is removed when the motherboard asserts $\overline{\text{TACK}}$ or when as part of the interrupt handler.

Interrupt Acknowledge ($\overline{\text{TACK}}$; Input)

This signal is driven by the motherboard and indicates that the module should place its vector on the data bus (D0–D7). This signal is qualified with $\overline{\text{AS}}$.

Bus Request ($\overline{\text{BREQ}}$; Output)

This signal is driven by a bus master I/O slot card when it desires access to the bus. It is removed after the module has completed all desired transfers and three-stated the $\overline{\text{AS}}$, BRSTA, CINH, TACK, address and data bus.

Bus Acknowledge ($\overline{\text{BACK}}$; Input)

This signal is driven by the CPU module to indicate that the requesting I/O slot now has ownership of the bus. It is removed when the module releases $\overline{\text{BREQ}}$.

3.3.3 Electrical Specifications

The following paragraphs define the electrical characteristics of the signals and components of the M68EC0x0IDP bus.

3.3.3.1 POWER SUPPLY

The M68EC0x0IDP bus requires +5 V and ground. The tolerance for +5 V as measured to ground is $\pm 5\%$ over the temperature range of 0 to 50° C.

3.3.3.2 BUS CONNECTOR

The M68EC0x0IDP bus uses DIN 41612-CS connectors that are characterized by the IEC DIN 41610 specification. The parameters shown in Table 3-24 are extracted from the IEC 603-2 specification class 2.

Table 3-24. Bus Parameters

Parameters	Specification
Current per Contact	2 A (20C)
Resistance per Contact	0.02 Ω maximum
Insertion Cycles	400 minimum.
Test Voltage	100 VAC @ 60 Hz
Temperature Range	-55 to +125°C
Isolation Resistance	10 Mg Ω minimum.

3.3.3.3 BUS DRIVER CHARACTERISTICS

The M68EC0x0IDP bus uses three types of drivers, totem-pole, open collector and three-state. The totem-pole drivers are used for non-shared signals whereas the three-state drivers are used for signals that may be driven from multiple sources. Only one driver can be active at a time for a given signal. Table 3-25 defines the electrical requirements for both driver types.

Table 3-25. Electrical Requirements

Driver Type	Output Low Current— I_{OL}	Output High Current— I_{OH}	Input Low Voltage Max.	Input High Voltage Min.	Max. Output Capacitance
Totem Pole	8 mA	400 μ A	0.6 V @ 8 mA	2.4 V @ 400 μ A	15pF
Open Collector	12 ma	—	0.8 V @ 12 ma	2.0 V	4.5 pF
Three-State	48 mA	3mA	0.6 V @ 48 mA	2.4 V @ 3 mA	18pF

Recommended three-state driver devices are 74F24X, 74F64X, 74F125 or equivalent. Recommended totem pole driver devices are 74F00, 74F04, or equivalent. The recommended open collector device is the MC88916. Active negation of signals is recommended for three-state driver devices.

3.3.3.4 BUS RECEIVER CHARACTERISTICS

Receivers should have an input diode forcing a maximum input clamp voltage of -1.5 V. Table 3-26 lists the receiver characteristics.

Table 3-26. Receiver Characteristics

Comments	Input Low Current I_{IL}	Input High Current I_{IH}	Input Low Voltage Max.	Input High Voltage Min.	Max. Input Capacitance
Recommended Value	400 μ A	25 μ A	2V	0.8V	10pf
Acceptable Limit	600 μ A	50 μ A	2V	0.8V	15pf

3.3.3.5 TRANSCEIVER CHARACTERISTICS

Transceivers must have the same characteristics as those defined separately for the drivers and receivers, except that the maximum combined input and output capacitance is 18pF.

3.3.4 Bus Timing

For information on M68EC0X0IDP bus timing, write to the following address:

Motorola
 High-Performance MPU Division
 ATTN: IDP Technical Support
 Mail Stop OE-33
 6501 William Cannon Drive West
 Austin, Texas 78735-8598 USA

3.3.5 Memory Map

The M68EC0X0IDP bus has 26 address lines providing 64MB 32-bit long words or 256 Mbytes of total memory space. This space is divided into sixteen 16-Mbyte slots. Slot \$0, \$C and \$D are reserved for the motherboard; the next 5 slots are used for I/O modules. Slot \$E is the ID ROM slot, and slot \$F is guaranteed to be empty. Slots \$6 through \$B are reserved.

Slot addressing eliminates the problems associated with setting address or option jumpers. An identification mechanism is provided to allow software to determine the exact configuration of the system at run time. Each I/O slot has a corresponding 1 Mbyte block in the ID ROM slot where the ID ROM for a module plugged into that slot would appear.

Figure 3-16 illustrates the top-level memory map and the ID ROM map for the M68EC0X0IDP bus.

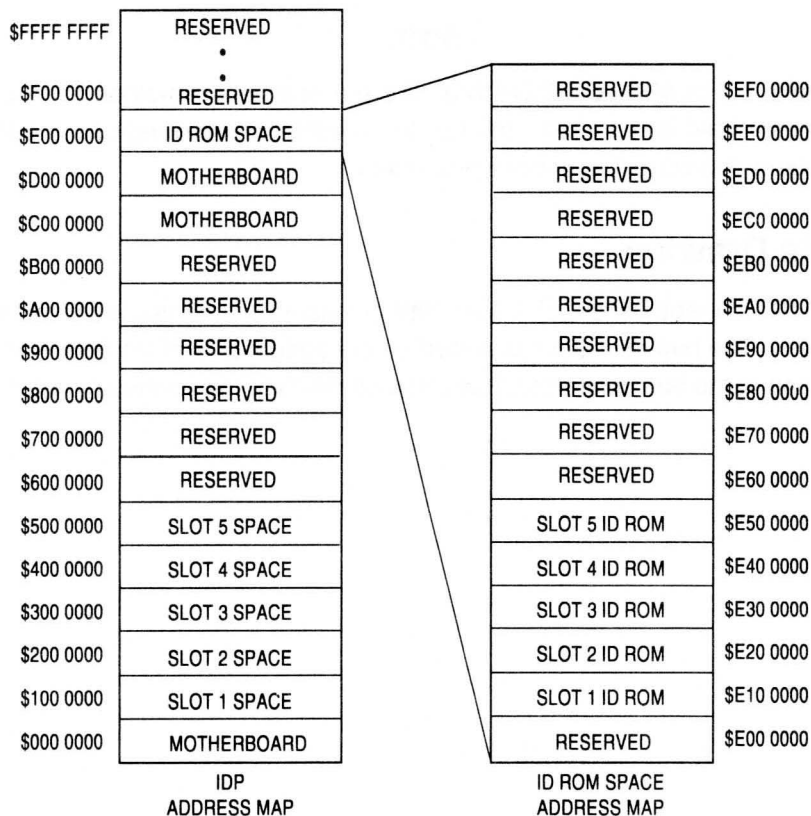


Figure 3-16. Memory Map

3.3.6 ID ROM Format

If the user desires to make a I/O module commercially available, the user should contact Motorola by writing to the address listed in **3.3.4 Bus Timing**. Vendor ID codes, module ID codes, etc. will be assigned by Motorola.

For users developing their own I/O modules and not making them commercially available, the following information is provided to avoid conflict with commercial modules.

The vector numbers 64–79 and 253–254 are reserved for Motorola use. Vector numbers 80–252 are available for third party hardware and software vendors and customers. The vendors and customers are requested to call Motorola and reserve these vector numbers. Vector numbers 80 - 90 will not be assigned to third party hardware vendors, so are ideal for customer use that will not interfere with commercially available cards.

3.4 MECHANICAL SPECIFICATIONS

This section defines the mechanical dimensions of the M68EC0x0IDP bus I/O Modules.

3.4.1 IDP Bus Connector

The M68EC0x0IDP I/O modules use a DIN41612-CS female-type connector. The recommended connector is a Robinson-Nugent Part No. DIN-96CSC-S1-TR.

NOTE

As used in the M68EC0x0IDP bus, the pin orientation molded onto the connector itself is not valid. Refer to the mechanical drawings for the I/O module for the correct connector placement.

3.4.2 I/O Module Drawing

The basic M68EC0x0IDP I/O module is 3.5" x 7.8". The mechanical dimensions for the single-size module are shown in Figure 3-14. The bus connector is placed on the bottom side of the board. Refer to Figure 3-17 for the correct orientation of the bus connector. Pay close attention to the connector notch and keying.

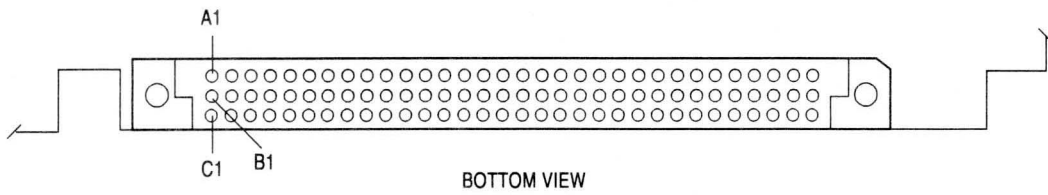
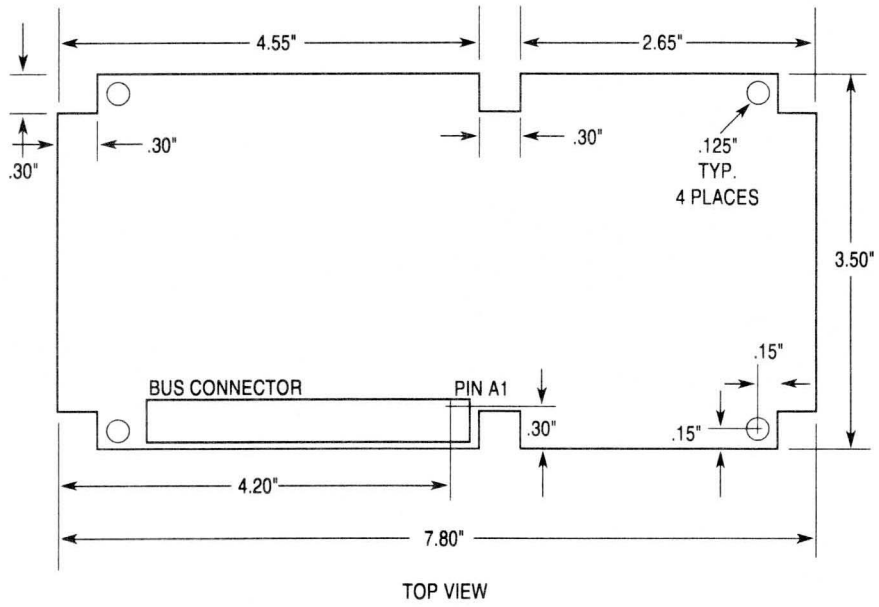


Figure 3-17. Mechanical Dimensions

SECTION 4

ROM68K CONFIGURATION AND COMMAND SET

This section provides detailed information on Integrated Systems' ROM68K installed on the M68EC0x0IDP integrated development platform (IDP) and the ROM68K command set.

4.1 OVERVIEW

The capabilities of the ROM68K are as follows:

- Serves as a basic interactive ROM monitor with commands for reading and writing memory, viewing and/or changing CPU registers, and executing code. ROM68K will also catch exception conditions that occur during execution.
- Provides a means of downloading code (the running start OS) from the host system where it is built. Downloading can be done either through the board's RS-232 serial port or via a network connection. Network downloading uses the TFTP protocol.
- Can be configured to automatically load an OS from a host system and start it running anytime the target CPU is powered on or reset. Thus, it can be part of a finished system, providing autoboot from a host computer.

The ROM68K supports two different hardware configurations. Before attempting to use the ROM68K, identify the configuration that matches your requirements. The configurations are:

1. A single CPU with no network connection
2. A single CPU with an Ethernet connection to a host system

4.2 SINGLE CPU

The simplest configuration is a target CPU board connected via RS-232 to a host computer system running a terminal emulation program. The terminal emulation program must be able to transfer text files over the serial channel. Most terminal emulators have this capability.

When using this configuration, the **DC** command tells the ROM68K to expect an S-record file, and then uses the terminal emulator to send the S-record file to the console. After the file has been loaded, a start can be executed with the **GO** command. The following paragraphs explain how to use the ROM68K to download and start an OS on a single-target CPU.

4.2.1 Downloading via RS232

Once the ROM68K is running, your OS can be downloaded through the serial channel. The **DC** command tells the ROM68K that an S-record file is downloading through the serial channel. The ROM68K then displays the message: `waiting for S-records from host....` At this point, the S-record file containing the operating system (`os.hex`) must be sent through the serial connection. Normally, terminal

emulators have an escape sequence or special keypad function to send a file. For example, the UNIX `tip` program uses `~>` as the escape sequence to send a file. When these two characters are entered, `tip` prompts for the name of a file to send.

At the conclusion of the download, **DC** prints out a message stating that the download has completed. To transfer control to the OS, you use the **GO** command. **GO** will need a parameter specifying the address at which to start execution. The OS entry point is `0x40008`. The following example demonstrates a loading and starting an OS through `tip`:

```
ROM68K>DC
Waiting for S-records from host... ~>Local file name? os.hex
14148 lines transferred in 11 minutes 1 second
!

14148 records read

ROM68K>go 40008
```

4.2.2 CPU With An Ethernet Interface

There are advantages to configuring a network interface on your board: 1) downloading is much faster over the network and 2) the ROM68K autoboot feature can be used. Once configured, autoboot will automatically load and start your OS after every board reset.

An ethernet card which works with the IDP board will be available from Motorola sales offices in Q4 1993. The part number for ordering the ethernet card is `M68EC0X0ENT`. The details on how to configure the ethernet card are described in a separate manual which is shipped with the Ethernet card.

4.3 ROM68K COMMAND SET

The ROM68K commands can be divided according to function into the five groups listed in Table 4-1.

Table 4-1. ROM68K Command Set

Memory Manipulation Commands	
AS	Assemble Into Memory
CM	Compare Memory
DI	Disassemble Memory
DM	Display Memory
FM	Fill Memory
MM	Move Memory
MP	Map Logical Address
PM	Patch Memory
SM	Search Memory
Register Manipulation Commands	
DO	Display Offset Registers
DR	Display Registers
PR	Patch Register
Execution/Breakpoint Commands	
CB	Clear Breakpoint(S)
DB	Define Breakpoint
GO	Start Execution
LB	List Breakpoints
ST	Single Step
File Loading/Booting Commands	
BI	Boot Intermetric MON68
BO	Boot from Network
DC	Download S-Record File Through Console
Miscellaneous Commands	
EC	Evaluate Constants
HC	Hardware Configuration
HE	Help

Those familiar with pROBE⁺/68K should have no difficulty using commands from the first three groups because they are almost identical to their pROBE⁺ counterparts. They only differ from pROBE⁺ in that any pSOS-related options, such as task qualifiers, have been removed.

4.3.1 Memory Manipulation Commands

These commands are used to remap, examine, and/or modify memory. The user can optionally specify the width of the data element on which the command operates by appending a period and a width specifier to the command name. Valid width specifiers are:

B	Byte (1 byte—default)
W	Word (2 bytes)
L	Long (4 bytes)

For example, DM.L 40000 displays memory in long words. This width specification is used in two ways. First, it determines the size of the command operands. For example FM.B fills memory with a byte value; whereas, FM.W fills memory with a word value. Second, it determines the access size used in the memory cycles. For example, PM.W should be used to patch memory that can only be accessed in word mode. The default width is byte.

Many memory display/modify commands require you to enter an address range. An address range can be entered two ways:

1. A starting and ending address separated by two periods. For example, to display memory from 40000 through 40FFF:

```
DM.L 40000..40FFF
```

2. A starting address and count. If a count is entered, the ending address is determined by first scaling the count by the size of the command operand (B, W, or L), and then adding the scaled count, less one, to the start address. For example, to display 40000 through 40FFF:

```
DM.L 40000 400
```

Note that the address range specified is inclusive of both the starting and ending addresses.

An address can be either an absolute number, as in the examples, or a combination of register contents and an optional offset. To tell the ROMs to use the contents of a register, enter a slash (/) followed by the name of the register. For example, 14/A0 means take the current content of register A0 and add 14 (hex) to it. Thus, if register A0 currently contains 4E000, 14/A0 evaluates to 4E014. An offset of zero need not be explicitly specified—i.e. /A0 is valid.

4.3.2 Register Manipulation Commands

The register manipulation commands allow the user to view or change the contents of the processor's registers.

The ROM68K also include eight pseudo-registers known as offset registers. These registers can be used anywhere in a command that a processor register can be used, such as when specifying an address. The offset registers are named F0–F7.

4.3.3 Execution/Breakpoint Commands

These commands can be used to define, clear, and display breakpoints and to start or resume the OS.

4.3.4 File-Loading/Booting Commands

These commands load S-record files into memory, either from the network or the console channel, and boot the system.

4.3.5 Miscellaneous Commands

The remaining ROM68K commands are used for changing the operating parameters, converting between decimal and hexadecimal numbers, and obtaining on-line help.

Usage: AS [.<width>] <address>

Options:

Description:

The AS command displays and allows you to optionally modify the contents of memory using standard Motorola 680X0 instruction mnemonics. AS first disassembles the instruction at <address>, displaying both the hexadecimal op-code and the instruction mnemonic, and then uses an equal (=) sign to prompt for a new instruction:

```
<address> <old instruction>  
= <your input>
```

<old instruction> represents the current contents of <address>. <Your input> can be:

- (a) a new instruction
- (b) a period to terminate the command
- (c) a carriage return to advance to the next instruction

When the user enters a new instruction, AS will confirm the input by again disassembling the current location.

The following rules should be observed when entering new instructions:

1. Use standard Motorola 680X0 mnemonics.
2. Addresses in PC relative operands (e.g., branches) must be entered in absolute form. AS automatically calculates the correct relative offset.
3. Constants are entered in AS differently than in the other ROM68K commands. Decimal is the default base. Hexadecimal constants must be preceded by a dollar (\$) sign.
4. Non-instructions may be entered using the pseudo-opcode DC.W.
5. Only instructions for the correct processor type are allowed. For instance, MC68020 instructions will be rejected on an MC68000 processor.

Usage: BI

Options: None

Description:

BI is used to switch from ROM68K to MON68. The system configuration is unchanged when using this command—i.e., all memory remapping and hardware configuration changes done while in ROM68K will remain configured if BI is used.

CB

Clear Breakpoint(S)

CB

Usage: CB <break_index>

Options: <break_index> the index of the break, which can be seen in the display produced by the DB and LB commands

Description:

The CB command is used to remove one or all of the break definitions from the breakpoint table. All breaks in the table can be removed by specifying the wild-card character (*) for <break_index>.

CM

Compare Memory

CM

Usage: CM [.<width>] <range> <address>

Options:

Description:

The CM command compares data in the memory region specified by <range> with the data starting at <address>. Comparisons are done on an element-by-element (byte, word, or long word) basis, and all mismatches are reported.

Usage: DB <address> [<count>]

Options:

Description:

DB sets an instruction breakpoint. When the program being debugged encounters an instruction breakpoint, execution stops just before the instruction on which the breakpoint is set to be executed. An instruction break may be optionally qualified by a pass count so that the break occurs only after the instruction has been executed a specified number of times.

If qualified by <count>, the monitor decrements a pass counter each time the specified instruction is executed. Execution stops only on the <count> pass. If a break occurs for some other reason, before exhausting the count, breakpoint displays will show the count remaining. If omitted, a <count> of 1 is assumed.

DB verifies that <address> can be written, which precludes breaks at nonexistent, odd, or ROM addresses.

NOTE

A breakpoint stops immediately before execution of the instruction at the designated location.

Usage: DC [`.<width>`] [`<offset>` | `PREV`]

Options:

<code>width</code>	specifies the type of memory access to use
a. <code>B</code>	Byte(default)
b. <code>W</code>	Word
c. <code>L</code>	Long Word
<code><offset></code>	specifies an amount to add to the load address of each S-record before loading it to memory.

Description:

DC loads an S-record file into memory. The S-records are assumed to always arrive via the console channel. After this command is entered, the following message is displayed:

```
Waiting for S-records from the host
```

After this message all further input to the console is assumed to be valid S-records until either a terminating S-record or an RS-232 break is received. When the download terminates, a message is printed stating the number of records loaded.

If the optional parameter `PREV` is used, DC prints the final message from the previous DC command.

NOTES

Downloading files via the serial port at 9600 BAUD takes approximately 1 second per 1KB of code

If the `MP` command is used, the first 256K bytes of RAM is reserved for system use (first user physical address space available is \$40000). Otherwise, if the `MP` command is not used, only the first 64K bytes of RAM is reserved for system use (first usable address is \$10000).

To ensure that all download information stays uncorrupted in memory if the reset button is pushed, refer to `HC` command and disable memory test.

The following text provides downloading examples using PC-compatible, Macintosh, and Unix-based systems.

The download procedures differ depending on the type of host that is being used. The examples shown in the following text represent the most common hardware/software configurations.

IBM-PC and Compatibles:

A communication program such as PROCOMM may be used to download an ASCII file on to the target board. Steps to download S-records are:

1. While inside the M68EC0x0IDP monitor, type DC at the prompt. The screen should appear as such:

```
ROM68K-> dc
```

The monitor then responds with:

```
Waiting for S-records from the host . . .
```

2. At this time, it is PROCOMM's responsibility to send a text stream into the console port. This may be achieved by pressing the PgUp key on the keyboard. A menu then appears to prompt the user for the correct file type. Choose 7) for ASCII. PROCOMM then prompts the user for the file to be sent to the M68EC0x0IDP .

For Windows 3.0 systems:

The utility program "Terminal" is included in the Windows 3.0 release. This program may be used to download S-records onto the M68EC0x0IDP target system.

1. While inside the M68EC0x0IDP monitor, type DC at the prompt. The screen should appear as such:

```
ROM68K-> dc
```

The monitor then responds with:

```
Waiting for S-records from the host . . .
```

2. Click on the TRANSFER menu and choose SEND TEXT FILE. A menu appears to prompt the user to select the file to be sent.

For The Macintosh:

A communication program such as Whiteknight may be used.

1. While inside the M68EC0x0IDP monitor, type DC at the prompt. The screen should appear as such:

```
ROM68K-> dc
```

The monitor then responds with:

```
Waiting for S-records from the host . . .
```

2. Click on the FILE menu and choose SEND TEXT FILE. A menu appears to prompt the user to select the file to be sent.

For Unix Systems:

The utility TIP or CU may be used to communicate with the M68EC0x0IDP Target System. Please note that there may be some software initialization needed to properly configure one of the serial ports as a direct connection to the M68EC0x0IDP

1. Ensure that the `/etc/getty` daemon is not running on the serial port that is chosen to connect to the M68EC0x0IDP. There should be an entry within the `/etc/ttyab` file that looks something like:

```
ttya "usr/etc/getty std.9600" unknown off local secure
```

The fourth field for the port you are using contains the word "off" which disables `/etc/getty` from allowing a login on this port.

If it is necessary to modify the `/etc/ttytab` file, execute a `kill -1` to signal the OS to re-examine the `/etc/ttytab` file.

2. Ensure that the read and write permissions on the device are enabled. This may easily be checked by typing

```
$ ls -l /dev/ttya
```

on the Unix command prompt. The output of the above command should appear as follows:

```
crw-rw-rw- 1 root 12, 0 Oct 8 1991 /dev/ttya
```

If not, use the `chmod` command to add the necessary read and write permissions.

3. Add an entry in the `/etc/remote`:

```
idp:dv=/dev/ttya:br#9600
```

4. To log into the M68EC0x0IDP system, type:

```
$ tip idp
```

NOTE

Please note that hardware handshaking is not implemented on the serial device of the IDP. Some terminal emulation programs may require signals such as "DCD" (Data Carrier Detect) to be present. In those cases, please ensure that those options be set appropriately. Also, it may be necessary to pause for at least 1/10 second (or more) after each line of S-record transferred.

If the SPARC types "connected", the software is hooked up correctly. If nothing happens, ensure that the M68EC0x0IDP power supply is turned on and connected to the M68EC0x0IDP, the M68EC0x0IDP serial port 0 is connected to ttya of the SUN and the baud rate in the `/etc/remote` is configured correctly. Otherwise, press the reset button on the M68EC0x0IDP. The prompt should appear as:

```
ROM68K->
```

5. While inside the M68EC0x0IDP monitor, type DC at the prompt.

```
ROM68K-> dc
```

The monitor then responds with:

```
Waiting for S-records from the host . . .
```

Without typing a carriage return, type `~$cat filename` such that the line appears as such:

```
Waiting for S-records from the host . . . ~$cat filename
```

You can also try `~<filename` and then hit return. If you are unable to transfer the file using these commands, then the problem lies with the S-record file.

Press carriage return to initiate the download. Note that the file must be an ASCII file. For more information in the usage of the TIP or CU utility, use the on-line man pages. The command "`~.`" is useful in ending the UNIX- M68EC0x0IDP communication.

Special Topics:

Please note that it is possible to add an offset to the `dc` command. For instance if the S-Records start at address \$0, but a "\$40000" is appended to the "dc" command, then the S-record download starts at address $\$0 + \$40000 = \$40000$. So DC \$40000 would download S-Records that normally would go to address \$0 to \$40000.

Usage: DI [.<width>] [<address> [<count>]]

Options:

Description:

The DI command disassembles the contents of memory, using standard Motorola 680X0 instruction mnemonics. Disassembly begins at <address> or, if <address> is not specified, at the current PC. <count> specifies the number of instructions to disassemble. If not specified, <count> defaults to 8. DI can be repeated by hitting the ESCAPE key, which causes disassembly to continue with the next block of instructions.

Memory contents which cannot be disassembled are shown as:

DC.W <hex number>

Note that DI displays constants differently from other ROM68K commands. Within the disassembled instructions, the default base is decimal. Hexadecimal constants, must be, preceded by a dollar sign. The reason that decimals are often used in the instructions is that the corresponding hex values can be easily discerned in the hexadecimal display of the opcode.

Usage: DM [.<width>] {<address> | <range>}

Options:

Description:

The DM command displays memory in hexadecimal and ASCII form. The user may specify a complete memory range or simply the starting address, in which case a default count of 80 bytes is assumed. DM always displays an integral multiple of 16 bytes, irrespective of the specified address range. If hardware limitations require that less than 16 bytes be displayed (e.g. at the end of physical memory), then the DM command should be used instead.

DM can be repeated by hitting the ESCAPE key, which causes the display to continue to the next block of memory locations.

The address fields in the display are shown as either a relocatable register and offset or a hexadecimal number. One of the two formats is automatically chosen to match that of the <address> input on the command line.

DO

Display Offset Registers

DO

Usage: DO

Options: None

Description:

The DO command displays the contents of the eight offset registers, F0–F7.

DR

Display Registers

DR

Usage: DR

Options: None

Description:

The DR command displays the current contents of the CPU registers. In addition, DR always shows the most recent break condition.

EC

Evaluate Constant

EC

Usage: EC <value>

Options: None

Description:

The EC command evaluates a 32-bit expression and displays the resulting value in hexadecimal, decimal, and relocatable (offset register relative) forms.

FM

Fill Memory

FM

Usage: FM [**<width>**] **<range>** **<value>**

Options:

Description:

The FM command fills memory with the specified value. **<value>** must be consistent with **<width>**.

Usage: GO [<new_PC>] [,<tmp_bkpt1> [,<tmp_bkpt2>]]

Options:

Description:

The GO command starts/resumes execution of the application. If <new_PC> is entered, then the PC register will be loaded with <new_PC> prior to execution. Otherwise, execution resumes at the current PC.

Two temporary breakpoints can be set. These are similar to regular breakpoints, with two exceptions. First, as their name implies, they are temporary—as soon as a break occurs for *any* reason, they are removed. Second, they cannot be qualified by a pass count.

Like regular breakpoints, temporary breaks at illegal, odd, or ROM addresses are rejected.

To return to the monitor after executing a GO command, the code being executed should end with an exception (e.g. an illegal instruction [\$4AFC]). This will cause

Usage: HC <config item> <data/action>

Options:

Description:

HC is used for three purposes. The first is to set date, day and time. The second is to disable or enable the bootup memory test. The third is to set the baud rate for data download. The set time/date feature of the command is formatted as follows:

```
HC T yr mo dt dy hr mi
```

<config item> is T (time) and it is followed by <data> defined as follows:

yr—year in which user selects 00 through 99

mo—month in which user selects 01 through 12

dt—date in which user selects 01 through 31

dy—day in which user selects 01 through 07 (Mon = 01)

hr—hour in which user selects 00 through 23 (1 pm = 13)

mi—minute in which user selects 00 through 60.

To set the baud rate for downloading data, the following command sequence is used:

```
HC B <bps>
```

where <config item> is B (baud rate) and <bps> is 9600, 19200, or 38400. The configured baud rate takes effect when the RESET button is pushed. Upon reset, the LED displays the baud rate for 1 second. A "1" represents 19200, "9" represents 9600, and "3" represents 38400. If the user wishes to change the baud rate back to 9600 without going through the HC command, press the ABORT button during the 1 second window in which the baud rate appears on the LED display. The monitor then boots up at 9600 baud.

NOTE

The user should disable the memory test after bootup to ensure validity of downloaded code (see DC command). Otherwise, DRAM may be corrupted if a RESET is initiated.

To disable the bootup memory test the HC command is used as follows:

```
HC M D
```

To enable the bootup memory test the HC command is used as follows:

```
HC M E.
```

HE

Help

HE

Usage: HE [<cmd>]

Options: None

Description:

The HE command displays general information about the ROMs and a complete list of all the available commands. If a specific command name is specified, the special help information about that command is displayed. The commands HELP and ? are equivalent to HE.

LB

List Breakpoints

LB

Usage: LB

Options: None

Description:

The LB command displays all currently defined breakpoints.

MM

Move Memory

MM

Usage: MM [.<width>] <range> <address>

Options:

Description:

The MM command copies data from the memory locations specified by <range> to an area of memory starting at <address>.

Usage: MP [<action>] [<address>]

Options: <action> can be any of the following:

- S Must also be followed by an <address>. Stores the translation table starting at <address> into NVRAM. An MPQ is also initiated.
- Q Queries the contents of NVRAM.
- C Compiles the translation table from NVRAM without actually executing the remap function.
- G Executes the specified remapping.

<address> denotes the starting address in NVRAM for the address translation table. This option is used with the "S" option.

Description:

The MP command is used to access the address translation table to remap M68EC0x0IDP memory. This command is used after the translation table has been set up by the user. MP used alone automatically maps unused logical addresses to the No-Acknowledge physical memory space.

Since the M68EC0x0IDP is used for both hardware debugging and processor evaluation, it is necessary to understand how to remap memory to avoid resource contention when interfacing hardware to the local bus connector. Physical address remapping is accomplished by the hardware while logical address remapping is a function of the monitor. The default motherboard logical addresses are equivalent to the physical addresses.

Since one of the features of the M68EC0x0IDP includes an unbuffered CPU local bus, the user may attach hardware to that connector for the purposes of debugging a target system or running a benchmark with the optional memory board. When hardware is connected to the local bus, it may become necessary to remap the resources on the motherboard to avoid resource contention with the target system hardware. The "MP" command exists for this reason. Upon reset and up to the time the "MP G" command is issued, the M68EC0x0IDP keeps the target system from responding by keeping the MAPEN signal negated (low). The main function of the MP command is to move M68EC0x0IDP system resources that may possibly conflict with the target hardware's memory map to a user-defined area. The term "logical" address for this discussion refers to the user-defined addresses assigned to the M68EC0x0IDP resources. The term "physical" address refers to the default motherboard addresses. There are some restrictions as to how the M68EC0x0IDP is remapped. The MP command software relies on these restrictions to ensure that the debug monitor still runs under a different map.

An 8-bit binary number called the "System Byte" is needed to describe the starting address of a block of system resources. The starting address is given by the System Byte shifted to the left by 20 bits. For instance, if the System Byte is "5A", the starting logical address of the system block is "05A00000". Note that for processors that support only 24-bits of addressing, the upper byte is truncated; therefore, the remap flexibility is limited.

Assuming that the System Byte is "RR", the logical to physical translation is as follows:

Logical	Physical	Description	Size (bytes)
0RR0xxxx	000xxxxx	System/RAM(0)	64K
0RR1xxxx	001xxxxx	System/RAM(1)	64K
0RR2xxxx	002xxxxx	System/RAM(2)	64K
0RR3xxxx	003xxxxx	System/RAM(3)	64K
0RR4xxxx	reserved	No Acknowledge	
0RR5xxxx	reserved	No Acknowledge	
0RR6xxxx	reserved	No Acknowledge	
0RR7xxxx	reserved	No Acknowledge	
0RR8xxxx	reserved	No Acknowledge	
0RR9xxxx	reserved	No Acknowledge	
0RRAxxxx	00Axxxxx	RTC/NVRAM	64K
0RRBxxxx	00Bxxxxx	DUART	64K
0RRCxxxx	00Cxxxxx	PIT	64K
0RRDxxxx	00Dxxxxx	LED	64K
0RRExxxx	00Exxxxx	RAM(4)	64K
0RRFxxxx	00Fxxxxx	ROM1(0)	64K

Note that the above table does not account for all possible M68EC0x0IDP resources. For example, only the first 64K bytes of the "user" ROM (ROM1(0)) is made available through the above table. To add to the above translations, the remap command supports the use of 21 descriptors to further define the memory map from what is provided by system byte. Each descriptor consists of the starting logical address, the starting physical address, the number of 64K-byte blocks mapped by that descriptor, and an extra byte reserved for future use. If the number of blocks is 0, then that descriptor is ignored. All logical addresses not mapped by the System Byte or any of the 21 descriptors are automatically mapped to the "No Acknowledge" space. The System Byte resources are of higher priority than those remapped by the descriptors; therefore, all added descriptors that conflict with the System Byte resources will be overwritten by the System Byte resources.

The remap command also provides a means by which the interrupt logic, bus error time-out logic and the bus arbitration logic can be disabled. These bits should always be enabled to guarantee the operation of the debug monitor. If any of these bits are disabled, it is the responsibility of the target hardware to provide these functions. Careful consideration is recommended prior to disabling any of these bits.

The following example defines an address translation table written in "assembly". To actually create the table, the user must first define the starting logical address of the system resources (SYSTEMBYTE). Assuming that SYSTEMBYTE equals \$AB (arbitrary value), the starting logical address of the system resources is:

```
$0AB00000 for machines that support 32-bit addressing.
$ B00000 for machines that support 24-bit addressing.
```

```
SYSTEMBYTE equ $ab ;set log addr of system resources
```

The M68EC0x0IDP provides the flexibility to allow the user to disable bus arbitration, interrupt control and bus error logic in order to assign these responsibilities to the target system when developing hardware. For all other cases, these system hardware resources must be enabled for the M68EC0x0IDP to function properly. In other words, the default status of bus arbitration, interrupt control and bus error logic on the M68EC0x0IDP should be ENABLED.

```
ENABLED equ 0
DISABLED equ 1

BUS_ARB equ ENABLED ;default is enabled
INTERRUPTS equ ENABLED ;default is enabled
BERR equ ENABLED ;default is enabled
```

If the user wishes to map the physical locations \$00900000-\$0093FFFF to logical addresses \$0F180000-\$0F1BFFFF, the following is done. The number of 64k byte blocks (size) must be between 0 and 255. A size of 0 implies that the descriptor is invalid.

```
LOG_0 equ $0F180000 ;starting logical address
PHY_0 equ $00900000 ;starting physical address
SIZ_0 equ 4 ;4 * 64k bytes = 256k bytes to be translated
```

NOTE

On an M68EC000IDP logical address \$0 is used regardless of how the system byte is defined. Furthermore, the logical address \$FFFFxxxx is reserved and must not be used if IRQDIS bit = 0 (default case).

If descriptors 1 through 20 are not used, then LOG_x, PHY_x and SIZ_x may be equated to zero. This ensures that no extraneous translations are performed. To complete the assembly code file, the following assembly code is used.


```

dc.b  BERR<<2+BUS_ARB<<1+INTERRUPTS ;inter status in bit position 0
*                                     ;bus arb status in bit position 1
*                                     ;bus err status in bit position 2
dc.b  SYSTEMBYTE                      ;starting logical addr. for system
*                                     ;hardware resources (32-bit)
*
dc.l  LOG_0,PHY_0                      ;define logical and physical start
*                                     ;addresses
dc.b  SIZ_0,0                          ;define number of 64k byte blocks
*                                     ;to be translated
*
dc.l  LOG_1,PHY_1                      ;define logical and physical start
*                                     ;addresses
dc.b  SIZ_1,0                          ;define number of 64k byte blocks
*                                     ;to be translated
*
.
.
.
dc.l  LOG_20,PHY_20                   ;define logical and physical start
*                                     ;addresses
dc.b  SIZ_20,0                        ;define number of 64k byte blocks
*                                     ;to be translated

```

Once this assembly file is created, it is then assembled and linked to create an S-record file. This S-record file is then downloaded to the M68EC0x0IDP via the "DC" command. It may be downloaded to anywhere in memory. Once downloaded into memory it is then transferred to NVRAM via the "MP S" command. Once in NVRAM, this table is accessible to the other MP functions without having to download the original S-record file.

NOTE

Please note that if the MP command is used, the first 256K bytes of RAM is reserved for system use (first user-usable physical address is \$40000). Otherwise, if the MP command is not used, only the first 64K bytes of RAM is reserved for system use (first usable address is \$10000).

Usage: PM [**.<width>**] **<address>** [**SKIP**] [**<value>**]

Options:

Description:

The PM command is used to view and optionally change the value of one or more memory locations. It operates in one of two modes—immediate or interactive. If **<value>** is entered on the command line, the value at **<address>** is changed and the command is complete. Otherwise, PM enters interactive mode, allowing the user to scroll through and alter successive memory locations. The interactive display is formatted as follows:

```
<address> <current value> = [<value>] <patch delimiter>
```

The left-hand side of the display is generated by PM. The right-hand side (in bold) represents user input. If **<value>** is omitted, the contents of that memory location are left unchanged.

<patch delimiter> can be any of the following:

.	Terminate command
\	Back up to the previous element
=	Redisplay the current element
(null)	Advance to the next element

In all cases, a RETURN is needed to terminate the command line.

The SKIP option causes PM to skip every other data element (only in interactive mode). This option is especially useful when patching memory mapped I/O devices whose byte-wide registers are defined at odd or even-only locations.

Usage: PR <reg> [<value>]

Options:

Description:

The PR command is used to view and optionally change the contents of the CPU registers and the eight offset registers.

PR operates in two modes—immediate and interactive. If <value> is entered as part of the command line, then <reg> is changed, and the command is complete. Otherwise, PR enters the interactive mode, allowing the user to scroll through and change one or more registers.

CPU registers are divided into three separate groups: 1) control registers (e.g., PC, SR, VBR), 2) data registers and 3) address registers. The eight offset registers form a fourth group.

In the interactive mode, PR will scroll through only the affected group. To patch a register in a different group, a new PR command must be entered.

The interactive display is formatted as follows:

```
<reg> <current value> = [<new value>] <patch delimiter>
```

The left-hand side of the display is generated by PR. The right-hand side is the user input. <new value> should agree with the <reg> data size. If <new value> is omitted, then <current value> is retained. <patch delimiter> can be any of the following:

.	Terminate command
\	Back up to the previous element
=	Redisplay the current element
(null)	Advance to the next element

SM

Search Memory

SM

Usage: SM [.<width>] <range> [NOT] <value>

Options:

Description:

The SM command searches the specified address range and reports all matching occurrences of <value>. If a width other than .B is specified, then matching will occur only on aligned boundaries—i.e., for SM.W, matching will only be tested on word (even address) boundaries. The NOT operator reverses the matching sense and reports all non-matching occurrences of <value>.

ST

Step

ST

Usage: ST [<count>]

Options: <count> represents the number of instructions to execute. (defaults to 1)

Description:

The ST command is used to single step one or more instructions.

Prior to its execution, the ST command fully disassembles each instruction, giving its PC, hexadecimal object code, and mnemonic equivalent. After the last step, ST also outputs the standard break display, which includes the next instruction to be executed.

NOTE

If the interrupt mask in the current SR allows a pending interrupt to be serviced, the interrupt service code will not be single-stepped. It will execute transparently and silently during the ST command.

SECTION 5

MON68 CONFIGURATION

This section provides a general description of the MON68 and MON68 command set, as well as general guidelines for using the MON68 in standalone mode and together with the optional XDB debugger.

5.1 GENERAL DESCRIPTION

MON68 is a ROM-based software device that can be used to evaluate and debug programs as they run on development board systems built around the M68000 family microprocessors. In addition, MON68 directly supports board-based source-level debugging in conjunction with the Intermetrics XDB debugger. MON68 provides the following capabilities:

- Initializes the system
- Single-steps programs or executes in real-time
- Sets breakpoints on conditions, code, and data
- Examines memory and the trace buffer
- Displays and sets memory and register values
- Maintains a trace buffer for debugging purposes

MON68 is equipped with an extensive command language which is described in detail in the following paragraphs. The pre-configured interface to the XDB debugger allows you to use your development system for C-language source-level debugging. Since MON68 and the XDB debugger allow you to test, integrate, and debug application programs at both the C and assembly levels, these tools virtually eliminate the need to use more expensive analysis instrumentation.

5.2 TROUBLESHOOTING

Most problems in starting up MON68 result from improperly setting up the development board or from an improper connection between the host computer system and the development board.

Some of the common problems are:

- Specifying a baud rate different from the one the UART is configured to expect
- Not supplying power to the target system
- Using the wrong kind of RS-232 cable
- Plugging the cable into an incorrect port on the target or host
- Some target boards and hosts have several ports

5.3 STARTING MON68 WITH XDB

Upon startup, the XDB debugger tries to establish contact with the MON68, using the settings specified in the command line or by environment variables.

XDB has the ability to determine where the user code is located and what procedures are currently on the stack. When XDB first communicates with MON68, the following communication takes place:

1. XDB issues an **IN** command to determine the processor type.
2. XDB requests the first instruction of user code, looking for a LINK instruction, so it can correctly synchronize the source code with machine instructions.
3. XDB requests the following registers: program counter (PC), A5, A6, and A7. If the PC is within the bounds of the code to be debugged, the appropriate source code is displayed.
4. XDB then looks at the value of A6, the frame pointer. If this value is not zero, XDB will follow the frame pointer to determine what procedures have been called. XDB will follow either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the value of the status register. This allows XDB to build a stack window showing the current status of the stack. It is very important that A6 is set to zero in RMAIN to ensure that XDB does not have to chase down an invalid frame pointer.

When a reset and run is issued, the following communication takes place between XDB and MON68:

1. XDB issues a **DC** command to find the vector base register (VBR).
2. XDB issues a display memory at vector zero of the VBR. This is the initial stack pointer for the system and is set up by RMAIN. XDB sets either the USP or SSP to the value contained at that location.
3. XDB sends a command to reset all the registers to zero.
4. XDB sets the SR to 2700.
5. XDB goes to the symbol table and finds `__MAIN` and loads that address into the PC.
6. XDB then sets any breakpoints.
7. XDB then issues a **GO** command.

If the XDB debugger has trouble communicating with MON68, the following message is displayed:

```
Sorry, the monitor is not responding.
```

This indicates that no reply was detected after sending data to the target system. The debugger will try to reset MON68. If the XDB debugger can then get a response, you will be given another chance to communicate in transparency mode. If XDB still cannot get any response, it will give up and exit.

5.4 USING MON68 WITH XDB

The following paragraphs contain information about running MON68 with XDB, including MON68 and XDB's transparency mode, trace mode, and data breakpoints

5.4.1 Transparency Mode

XDB's transparency mode allows you to communicate directly with MON68. Most of the time, XDB can handle all low-level communications, freeing you to concentrate on the high-level C code. You may,

however, wish to communicate with MON68 to perform assembly-oriented tests, block fills, or other target system functions.

To enter transparency mode, type a lower-case letter "o" at the XDB prompt as follows:

```
xdb>o
```

Now all commands will be received directly by MON68.

To exit transparency mode, type ctrl-D (the end-of-file character), which brings you back to the XDB prompt.

When entering transparency mode, XDB removes any breakpoints it may have established in the target code to ensure that you can access unmodified target code. If necessary XDB reestablishes breakpoints when transparency mode ends.

Upon exit from transparency mode, the XDB reestablishes control over certain aspects of MON68 that it needs. As long as you avoid the XDB's own breakpoint trap, you may establish breakpoint conditions in transparency mode. These breakpoints will remain active after you exit transparency mode. If one of these breakpoints is hit, then XDB will issue at the following message:

```
Stopped on a breakpoint not set by the debugger.
```

5.4.2 Trace Mode

Trace mode sets the trace bit on the M68000 family microprocessors. When tracing is enabled, the program counter is stored in the trace buffer after execution of each instruction. The contents of the buffer may be checked to determine program flow. Since the PC has to be stored after each instruction when this tracing is enabled, the code is not running in real time. The trace buffer is configured to 1K of RAM.

5.4.3 Data Breakpoints

The MON68 version of XDB does not support the XDB's data breakpoint commands. However, by entering transparency mode, you may issue the MON68's set data breakpoint (**SBD**) and set data range breakpoint (**SBR**) commands to monitor the value of particular addresses.

5.5 USING MON68 WITH A DUMB TERMINAL

The following control codes may be typed to change a command line:

ctrl-X (line delete)

The cursor is backspaced to the beginning of the line. This control character will also terminate any option and return to the MON68.

ctrl- (backspace)

The cursor is moved back one space and the character at that position is deleted.

ctrl-C (Interrupt)

This character will terminate any operation and return to the MON68. If the user program is operating with the serial port interrupt disabled, the command will not be seen until interrupts are enabled.

5.6 MON68 COMMANDS

The MON68 commands can be divided according to function into the six groups listed in Table 5-1. MON68 supports commands in either standalone mode or through the XDB's transparency mode. A description of each command is included in the following pages.

Table 5-1. MON68 Commands

Command	Definition
Startup Commands	
DC	Display Configuration
IN	Initialize
Register Commands	
DR	Display Registers
SR	Set Registers
Memory Commands	
DM	Display Memory
SM	Set Memory
Breakpoint Commands	
DB	Display Breakpoint
RB	Remove Breakpoint
SB	Set Conditional Breakpoint
SBC	Set Address Breakpoint
SBD	Set Data Breakpoint
SBR	Set Data Range Breakpoint
Trace Functions	
DT	Display Trace
TE	Trace Enable
TD	Trace Disable
Execution Commands	
GO	Start Execution
SI	Single Step
SO	Step out of Range

Usage: DB

Example: Display the current breakpoints.

Command: >DB

Results: Code Breakpoints

1. 010000 count = 0002 actual = 0000 enabled
2. 010010 count = 0001 actual = 0000 enabled

Data Breakpoint

1. 0EA0000 data mask = 000ff cmp_flags = enabled
Word_cmp_BEQ

Data Range Breakpoint

1. 0EA1234 Low = 0007 High = 0009 cmp_flags = enable outside
check

Description:

This command displays all software breakpoints and their status, including whether the breakpoint is enabled or disabled, and the conditions on which the breakpoint is activated. Each breakpoint is assigned a number which may be used when specifying the breakpoint.

In the example above, the first code breakpoint was set to break after reaching this address a second time (count = 2). The second code breakpoint was set to break when the instruction at this address is executed. The data breakpoint was set to break if the word at address 0EA0000 is equal to 000FF. The data range breakpoint was set to break if the value at address 0EA1234 is lower than 0007 or higher than 0009 (hex).

The DB command may be repeated by pressing the ENTER key.

Usage: DC

Example: Display the current configuration.

Command: >DC

Results: The ROM Monitor uses the following resources

Micro Type	= M68010
VBR	= 00E80000
RAM Start	= 00E90000
RAM End	= 00E91800
INT Mask	= 0400
Break Trap	= 14
I/O Trap	= 13
Trace Buffer	= 1K

Description:

This command displays the overall configuration of the monitor, including the target processor type, address of the vector base register, and the beginning and ending RAM space. In the example above, the processor is a Motorola M68010, with a vector base register at address E80000. The monitor's RAM space, including the trace buffer, starts at E90000 and ends at E91800. Interrupts greater than level 4 are allowed to run. The breakpoint trap will be set to trap 14, and the I/O Trap will be set to trap 13. A 1K trace buffer has been established for storing trace information.

Usage: DM[B|L|W] [base_address] [length_number]

Options:

B	display information as bytes
L	display information as longs
W	display information as words (default)
base_address	the starting address for displaying memory
length_number	the length of the memory display, in terms of the units chosen. The default is 8.

Example 1: Display 5 word-sized data starting at address 200010. Notice that the ASCII representation is shown to the right.

Command: >DMW 200010 5

Result: 200010: 1253 1214 1500 3456 1234 .S....4V.4
>

Example 2: Display 5 byte-sized data starting at address 200010.

Command: >DMB 200010 5

Result: 200010: 12 53 12 14 15 .S...

Description:

This command displays the contents of a specified portion of memory, in byte (B), word (W, the default) or long word (L) format.

Usage: DR [register_name1 register_name2...]

Options: register_name name of the register to be displayed

Example: Display the contents of all registers (default) of a 68010.

Command: >DR

Results:

D0 = 00000000	D1 = 00000000	D2 = 00000000
D3 = 00000000	D4 = 00000000	D5 = 00000000
D6 = 00000000	D7 = 00000000	
A0 = 00000000	A1 = 00000000	A2 = 00000000
A3 = 00000000	A4 = 00000000	A5 = 00000000
A6 = 00000000	A7 = 00E84000	
USP = 00E85000	SSP* = 00E84000	PC = 00F00086
SR = 2000		
VBR = 00E80000	SFC = 0007	DFC = 0007

Description:

This command displays the contents of all, some, or one register. The active stack is indicated by an asterisk. If no specific register(s) is specified, then display all registers.

DT

Display Trace

DT

Usage: DT[B|F][F] [number_of_instructions]

Options:

- B display contents in a backward direction (default)
- F display contents in a forward direction
- F as a second possible F option, trace contents will be displayed with full data movements

number_of_instructions
the number of machine instructions of trace information to display. The default is 20.

Example: Display the trace buffer in a backwards direction for 20 instructions.

Command: >DTB

Results:

19.	00ea6070	2080	move.b	d1,(a0)
18.	00ea6074	10BC00AA	move.l	#\$AA,(a0)
			.	
			.	
			.	
0.	00ea607a	2210	move.l	(a0),d1>

Description:

This command displays the contents of the trace buffer if tracing has been enabled using the TE command or by setting a data breakpoint. The trace buffer contains the program and program counter history up to the point where MON68 became active.

Usage:	GO[D T] [address]
Options:	D execute with tracing disabled
	T execute with tracing enabled
	address the hex address at which execution will resume. The default is the current PC location.
Example:	Start execution at address 10000. A code breakpoint was set previously at address 1000E.
Command:	>GO 10000
Results:	..RUNNING REAL-TIME WITH BREAKPOINTS !BREAK! - Breakpoint at 001000E

Description:

This command starts real-time execution, which continues until either a breakpoint is taken or an interrupt is received from the host. The T and D options combine the GO command with the effects of the TE and TD commands.

If breakpoints are enabled, a TRAP instruction is inserted into the target code at each breakpoint address. This allows real-time execution of target code while allowing for breakpoints to be taken. If the location where execution begins contains a breakpoint TRAP, then that breakpoint is temporarily disabled until the program is stepped off the breakpoint. If an assertion has been set, then the GO command will enable the trace buffer and checks for break conditions after each instruction is executed.

After the GO command has been issued, MON68 will display a message showing one of the following modes of operation:

1. Real time
2. Real time with code breakpoints
3. Tracing (PC only)
4. Tracing (PC only) with assertions
5. Tracing (PC and data movements)
6. Tracing (PC and data movements) with assertions

The last function performed before starting execution of user code is to call `sys_go`.

IN

Initialize

IN

Usage: IN

Example: Initialize monitor

Command: >IN

Results: EST_RMbug Version: M68010
Revision 1.1 Copyright (c) Embedded Support Tools
Corporation 1989, 1992

Description:

This command identifies the MON68 version number, sets the monitor to default configuration parameters, and puts the monitor into command mode. Note that initialization of MON68 does not imply initialization of the target system.

RB

Remove Breakpoint

RB

Usage: RB[C|D|RT] [address]

Options:

C	remove code breakpoint (default)
D	remove data breakpoint
R	remove data range breakpoint
address	address of the breakpoint to be removed

Example: Remove code breakpoint at address 10000

Command: >RBC 10000

Description:

This command removes code (C), data (D) or range (R) software breakpoint codes at the specified address. If no type is specified, then all breakpoints are removed.

SBC

Set Address Breakpoint

SBC

Usage: SB[C] [address] [count]

Options:

C	set address breakpoint (default)
address	the address of the breakpoint
count	the number of times that the address must be reached before the breakpoint is taken

Example: Set a code breakpoint at address 10000 and take the breakpoint after the code has been executed 16 times (10 hex)

Command: >SBC 10000 10

Description:

This command sets a breakpoint at the address specified and takes the breakpoint if the address is reached a specified number of times.

SBD

Set Data Breakpoint

SBD

Usage: SBD [address] [data] [size_test]

Options:

address	the address of the breakpoint
data	the value to be tested against the value of the specified address
size_test	a compound test consisting of: <ul style="list-style-type: none">(a) the size of the operands:<ul style="list-style-type: none">B byteL longW word(b) the test to be performed:<ul style="list-style-type: none">C compareA and(c) the condition to satisfy:<ul style="list-style-type: none">E equalN not equal

the default size_test is LCE

Example: Set a data breakpoint at address E00000; take the breakpoint if the variable at this address is overwritten and the word (W) compare (C) does not equal (N) FFFF

Command: >SBD E00000 FFFF WCN

Description:

This command compares the value of the data at the address specified with a given value and takes the breakpoint if the values meet the conditions.

Usage: SBR [address] [low_value] [high_value] [E|N]

Options:

address	the value to be compared against a range of values
low_value	the lower boundary of the data range
high_value	the upper boundary of the data range
E	break if the data is outside of the specified range; do not break on boundaries (default)
N	break if the data is inside of the specified range; break on boundaries

Example: Set a data breakpoint at address E00000, and take the breakpoint if the value at this address is outside of the range (E) between 9FFF and B000

Command: >SBR E00000 9FFF B000 E

Description:

This command sets a breakpoint if the value at a specified location is inside or outside a given range. A test can be set to determine if the value at the specified address is equal (E) or not equal (N) to values outside a given range. This command only operates on the long word at the address specified. To use a byte or word at this address a mask must be used.

Usage: SB [address] [count] [>] [R|D|TD|TE] [value] [data] [size_test]

Options:

address	the address of the breakpoint														
count	the number of times that the address must be reached before the breakpoint is taken														
R	specifies that the value which follows will be a register														
D	specifies that the value which follows will be data														
TD	disable trace mode (value, data and size_test do not apply)														
TE	enable trace mode (value, data and size_test do not apply)														
value	the name of the register (with the R option) or the address of the data (with the D option)														
data	the value which will be tested against the value of the register or data														
size_test	a compound test consisting of: <ul style="list-style-type: none"> (a) the size of the operands: <table> <tr> <td>B</td> <td>byte</td> </tr> <tr> <td>L</td> <td>long</td> </tr> <tr> <td>W</td> <td>word</td> </tr> </table> (b) the test to be performed: <table> <tr> <td>C</td> <td>compare</td> </tr> <tr> <td>A</td> <td>and</td> </tr> </table> (c) the condition to satisfy: <table> <tr> <td>E</td> <td>equal</td> </tr> <tr> <td>N</td> <td>not equal</td> </tr> </table> <p>the default size_test is LCE</p>	B	byte	L	long	W	word	C	compare	A	and	E	equal	N	not equal
B	byte														
L	long														
W	word														
C	compare														
A	and														
E	equal														
N	not equal														

Example 1: Set a conditional breakpoint at address 10000 and take the breakpoint if register (R) D0 has the value FFFFFFFF.

Command: >SB 10000 > R D0 FFFFFFFF

Example 2: Set a conditional breakpoint at address 5072 and take the breakpoint if the data value (D) at address 5FEE is word (W) compare (C) equal (E) to 0000.

Command: >SB 5072 > D 5FEE 0000 WCE

Example 3: Enable tracing between addresses 5058 and 5072.

Commands: >SB 5058 > TE
>SB 5072 > TD

Description:

This command sets a conditional breakpoint at the address specified and takes the breakpoint after the associated conditions have been met. A TE or TD breakpoint stops execution at the specified address. Trace mode will either be enabled (TE) or disabled (TD), and execution will resume immediately. This type of breakpoint can be used to enable tracing through selected sections of code.

Usage: SI[DR] count

Options: DR display the register contents at each step
count the number of instructions to step

Example 1: Step 9 machine instructions.

Command: >SI 9

Example 2: Step one machine instruction and display registers.

Command: >SIDR

Results: !BREAK at 1000!

D0 = 00000000	D1 = 00000000	D2 = 00000000
D3 = 00000000	D4 = 00000000	D5 = 00000000
D6 = 00000000	D7 = 00000000	
A0 = 00000000	A1 = 00000000	A2 = 00000000
A3 = 00000000	A4 = 00000000	A5 = 00000000
A6 = 00000000	A7 = 00E84000	
USP = 00E85000	SSP = 00E84000	PC = 00F00086
SR = 2000		
VBR = 00E80000	SFC = 0007	DFC = 0007

Description:

This command causes the MON68 to execute the specified number of instructions with tracing enabled. After a single target instruction, no further action occurs unless the specified count has not decremented to zero. This command can be repeated by pressing the ENTER key.

SM

Set Memory

SM

Usage: SM[B|L|W] [base_address] [hex_byte(s)]

Options:

B	set memory with bytes
L	set memory with longs
W	set memory with words (default)
base_address	the address at which memory will start to be filled
hex_byte	the value(s) used to fill memory

Example: Fill three bytes of memory, starting at address 200010, with the values 0B, 7F, and 34.

Command: >SMB 200010 0B 7F 34

Description:

Starting at the specified address, This command fills memory with the values specified.

SO

Step Out of Address Range

SO

Usage: SO [first_address] [last_address]

Options: first_address the first address of the address range

last_address the last address of the address range

Example: Assuming that PC=1000, break if execution steps out of the range between 1000 and 1fff.

Command: >SO 0001000 0001fff

Result: !BREAK OUTSIDE RANGE AT \$3000!

Description:

This command starts execution, which continues until a breakpoint is encountered, an instruction outside of the specified range is about to be executed, or an interrupt is received from the host.

Usage: SR[B|L|W] register_name1 hex_string1 [register_name2 hex_string2]

Options:

B	bytes
L	longs
W	words
register_name	the name of the address register (Ax), data register (Dx), or all registers (all)
hex_string	the value to be placed in the register

Example: Set address register A0 to 005B and data register D2 to 7F34

Command: >SRW A0 005B D2 7F34

Description:

This command sets the specified registers to the specified hex values.

TD

Trace Disable

TD

Usage: TD

Example: Disable tracing.

Command: >TD

Description:

This command disables the tracing function. Tracing can only be disabled if no assertions are set. If no assertions are set, a GO command will have the program run in real time.

TE

Trace Enable

TE

Usage: TE

Example: Enable tracing.

Command: >TE

Description:

This command enables the tracing function, which captures PC history (the default) or full data movements if the CF command was used.

APPENDIX A M68EC0x0IDP SYSTEM CALLS

The following system calls are used for the M68EC0x0IDP. The monitor used is MON68.

IN_CHAR

Input	DO = IN_CHAR (0010)
Output	DO = Return Code D1 [7:0] = Character
Return Codes	0000 = RET_OK Successful return. 0001 = NO_CHAR
Description	When this system call is issued the ROM monitor will read a character from the input buffer. The ROM monitor returns the next character from its input buffer. If there is no character present in the buffer, IN_CHAR returns the error code NO_CHAR.
Examples	MOVE. W #IN_CHAR, D0 TRAP #15 ;trap
Note	Trap #15 Causes a break back into the monitor.

OUT_CHAR

Input	DO = OUT_CHAR (0013) D1 = Character to be transmitted
Output	DO = Return Code
Return Codes	0000 = RET_OK Successful return.
Description	When this system call is issued the ROM monitor will write a character out to the serial port.
Examples	MOVE. B CHAR, D1 MOVE.W #OUT_CHAR, D0 TRAP #15 ;trap
Note	Trap #15 Causes a break back into the monitor.

Motorola M68EC0x0IDP

Microprocessor with 128K Bytes of On-Chip EPROM

Page 11

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Page 12

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

Microprocessor with 128K Bytes of On-Chip EPROM

APPENDIX B

MC68230 PROGRAMMING EXAMPLE

The following example initializes both the PIT and the DUART. In addition, it provides the code that handles an interrupt request generated by the DUART and routed to the PIT.

```

org      $40000
vect     equ      64          this should be the vector number of
*                               the receiver.
h1vec    equ      0
h2vec    equ      1
h3vec    equ      2
h4vec    equ      3

duart    equ      $00b00003
pit      equ      $00c00003

```

*duart values

```

regoff   equ      4          this should be the difference between
ser1     equ      duart      terminal
ser2     equ      duart+8*regoff  serial port2 address

```

*symmetric duart offsets

```

mr1x     equ      0          mode register 1
mr2x     equ      0          mode register 2
srx      equ      1*regoff   status register
csrx     equ      1*regoff   clock select register
crx      equ      2*regoff   command register
rbx      equ      3*regoff   receiver buffer
tbx      equ      3*regoff   transmitter buffer

```

*non-symmetric duart offsets

```

ipcr     equ      4*regoff   input port change register
acr      equ      4*regoff   auxiliary control register
isr      equ      5*regoff   interrupt control register
imr      equ      5*regoff   interrupt mask register
cmsb     equ      6*regoff   current counter/timer most
*                               significant byte
ctur     equ      6*regoff   counter/timer upper register
clsb     equ      7*regoff   current counter/timer least
*                               significant byte
ctlr     equ      7*regoff   counter/timer lower register
ivr      equ      12*regoff  interrupt vector register
ip       equ      13*regoff  input port (unlatched)
opcr     equ      13*regoff  output port configuration register
strc     equ      14*regoff  start counter command
btst     equ      14*regoff  output port register bit set command
stpc     equ      15*regoff  stop-counter command

```

btrst equ 15*regoff output port register bit reset command

*pit values

pgcr	equ	0	port general control register
psrr	equ	1*regoff	port service request register
paddr	equ	2*regoff	port a data direction register
pbddr	equ	3*regoff	port b data direction register
pcddr	equ	4*regoff	port c data direction register
pivr	equ	5*regoff	port interrupt vector register
pacr	equ	6*regoff	port a control register
pbcr	equ	7*regoff	port b control register
padr	equ	8*regoff	port a data register
pbdr	equ	9*regoff	port b data register
paar	equ	10*regoff	port a alternate register
pbar	equ	11*regoff	port b alternate register
pcdr	equ	12*regoff	port c data register
pitsr	equ	13*regoff	port status register
tcr	equ	16*regoff	timer control register
tivr	equ	17*regoff	timer interrupt vector register
cprh	equ	19*regoff	counter preload high
cprm	equ	20*regoff	counter preload mid
cppl	equ	21*regoff	counter preload low
cntrh	equ	23*regoff	counter high
cntrm	equ	24*regoff	counter mid
cntrl	equ	25*regoff	counter low
tsr	equ	26*regoff	timer status register

*This initialization routine assumes that the transmitter is polled, but the receiver is interrupt-driven. This example uses channel A as the main i/o serial channel.

*

*initialize vector table entry for receive handler assume that vector base register=\$0

*

```
lea.l   rcv_hdlr,a0
move.l  a0,(vect+h3vec)<<2
```

*initialize pit

```
lea.l   pit,a0
move.b  #$00,pgcr(a0)   set mode to 0
move.b  #$18,psrr(a0)  set up pirq and piack
move.b  #$00,pbddr(a0) all pins on port b are inputs
move.b  #$82,pbcr(a0)  submode 1x, h3 interrupt enabled
move.b  #vect,pivr(a0) setup pivr
```

```
    move.b    #$20,pgcr(a0)    turn on the ports
```

*now initialize the duart registers

```
    lea      ser1,a0          get address of port a
    move.b   #$30,CRX(a0)     reset tx, channel a
    move.b   #$20,CRX(a0)     reset rx, channel a
    move.b   #$10,CRX(a0)     reset mr pointer, channel a
```

*init the general registers of the duart

```
    move.b   #$0f,ivr(a0)     init ivr
    move.b   #$02,imr(a0)     init imr
    move.b   #$00,acr(a0)     init acr
    move.b   #$00,ctur(a0)    init ctur
    move.b   #$02,ctlr(a0)    init ctlr
    move.b   #$00,opcr(a0)    init opcr
    move.b   #$01,btst(a0)    init cts
```

*init the actual serial port

```
    move.b   #$bb,csrx(a0)    init csrx
    move.b   #$13,mr1x(a0)    init mr1x
    move.b   #$07,mr2x(a0)    init mr2x
    move.b   #$05,crx(a0)     enable port
```

```
    move.w   #$2000,sr        turn on all interrupts
endless bra    endless       enter endless loop
```

*

* In this sample interrupt handler, once a character is received, it is immediately echoed out.

*

rcv_hdlr:

```
    move.l   a0,-(sp)         save registers
    move.l   d0,-(sp)
    move.l   d1,-(sp)

    lea.l   pit,a0
    bset.b   #2,pitsr(a0)     clear pit interrupt

    lea.l   ser1,a0

    move.b   srx(a0),d1
    and.b    #1,d1            extract rcvrdy
```

```

        beq.b    oops          if not ready, must be error
*
*                               if error, just get out so that
                               receive fifo is not disturbed

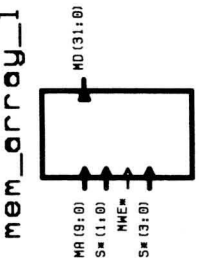
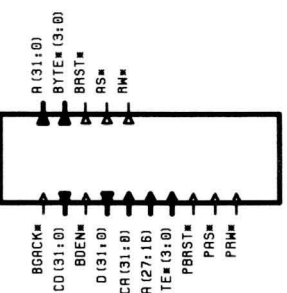
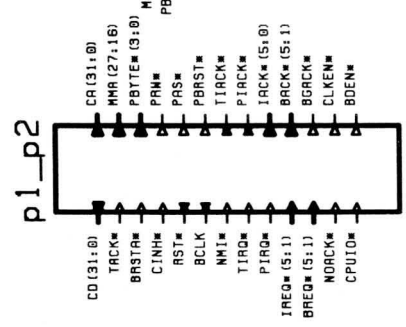
        move.b  rbx(a0),d0    read character into d0
*
waittxrdy:
        move.b  srx(a0),d1
        and.b   #4,d1        extract txdy
        beq.b   waittxrdy
        move.b  d0,tbx(a0)    echo character found in d0
*
oops:
        move.l  (sp)+,d1      restore registers
        move.l  (sp)+,d0
        move.l  (sp)+,a0
        rte

end

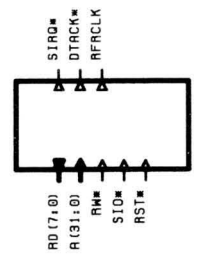
```


APPENDIX C SCHEMATICS

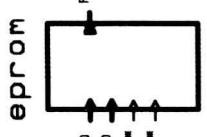
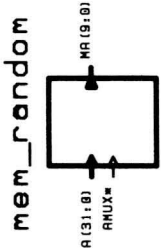
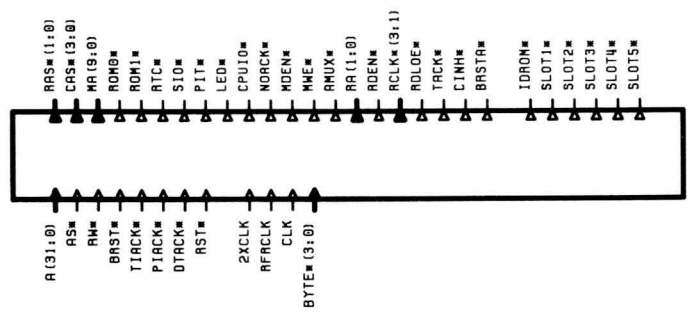
slot_interface



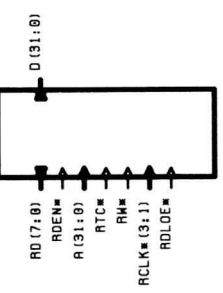
sio_rs232



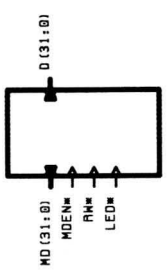
fpga



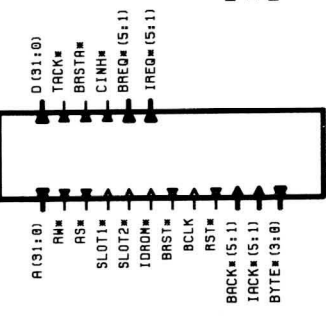
rtc



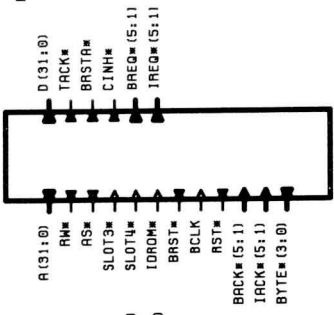
mdata_buff



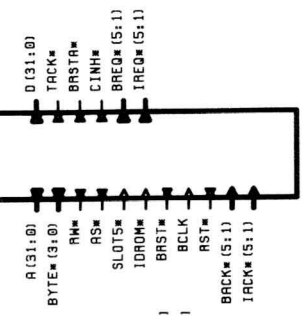
iomod_a



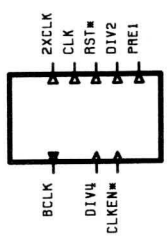
iomod_b



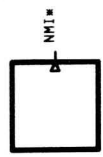
iomod_c



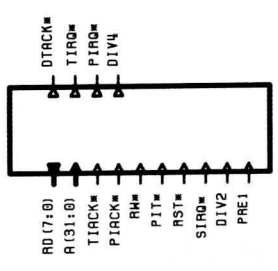
clock_sync

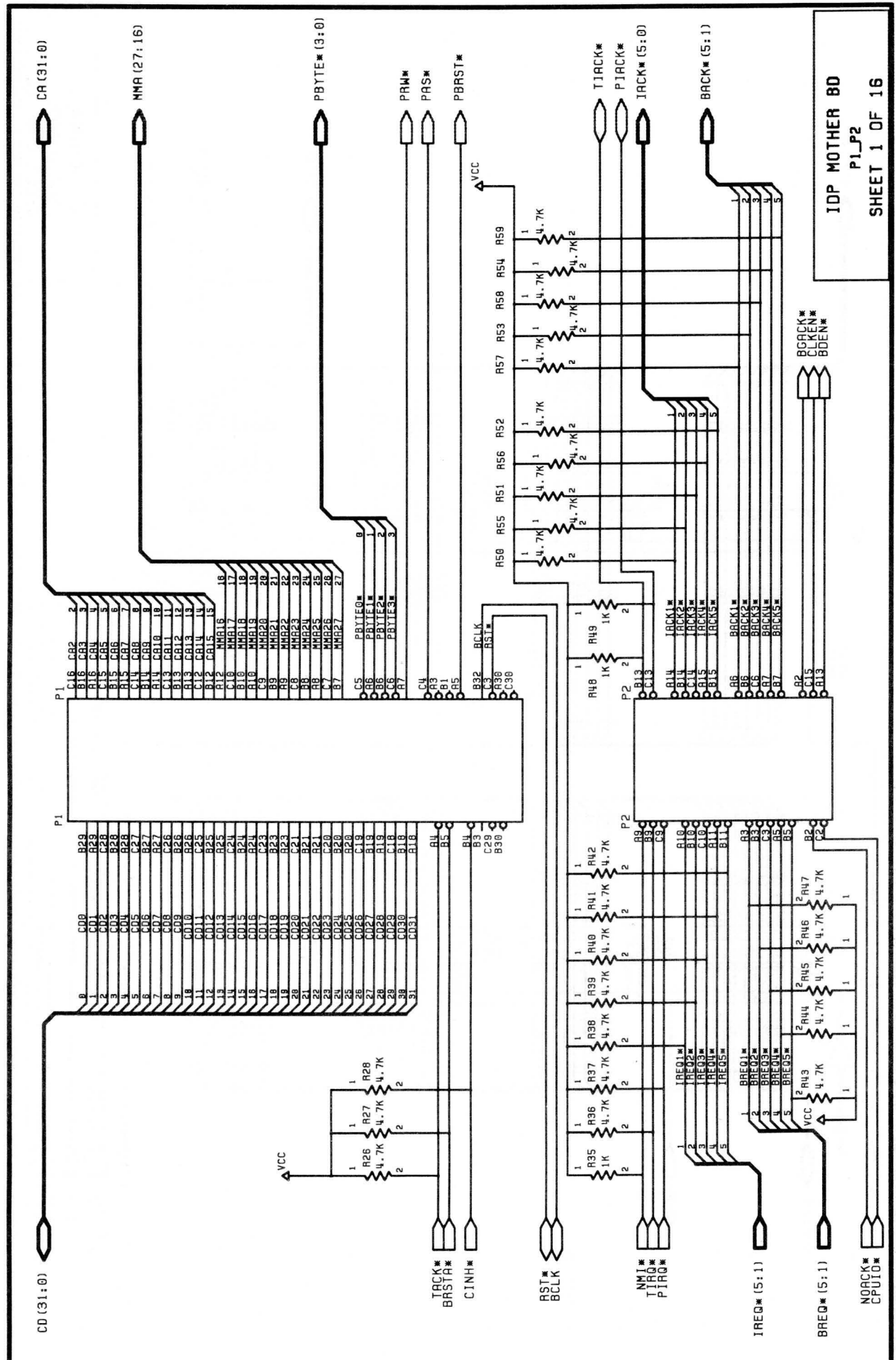


power_caps

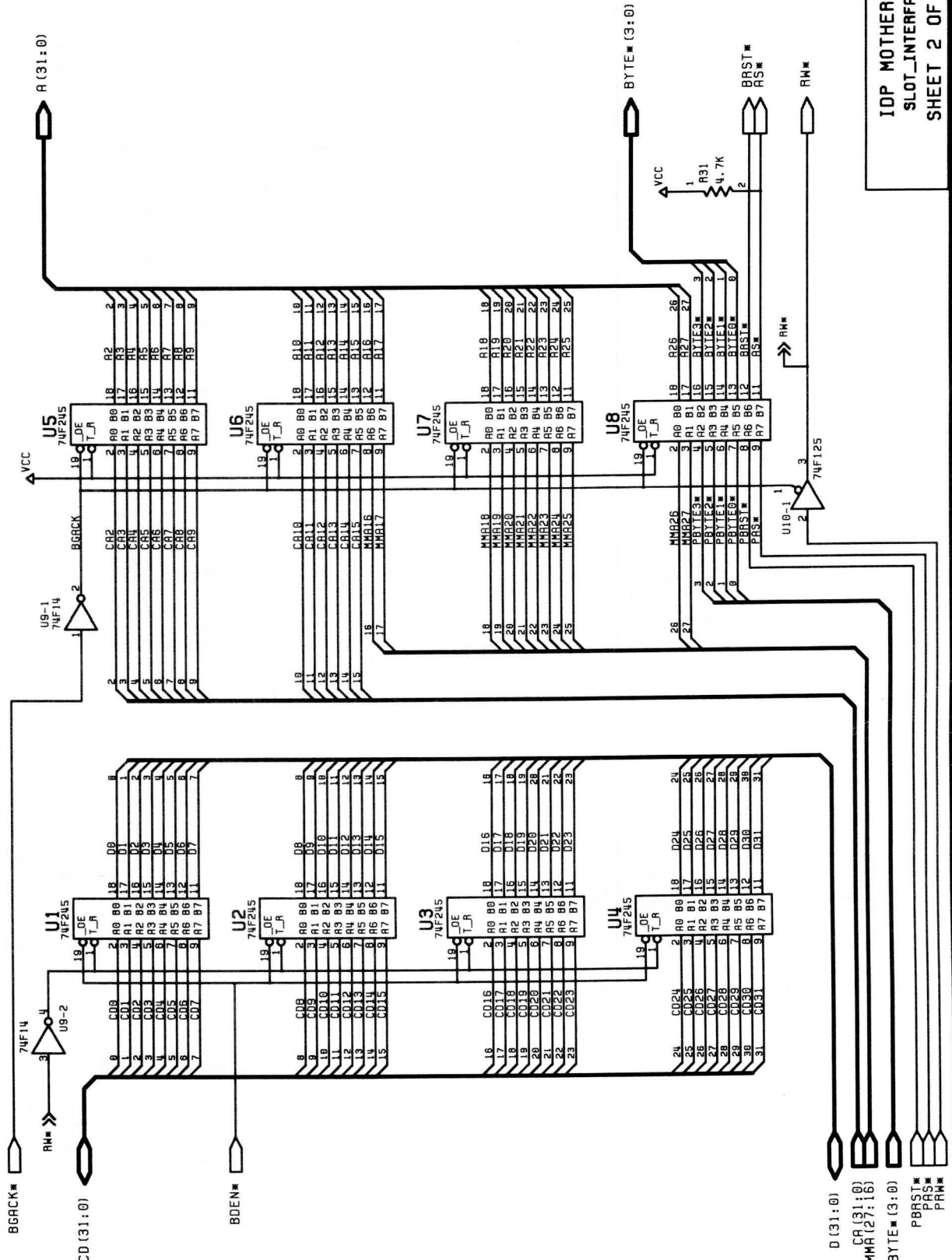


parallel_io

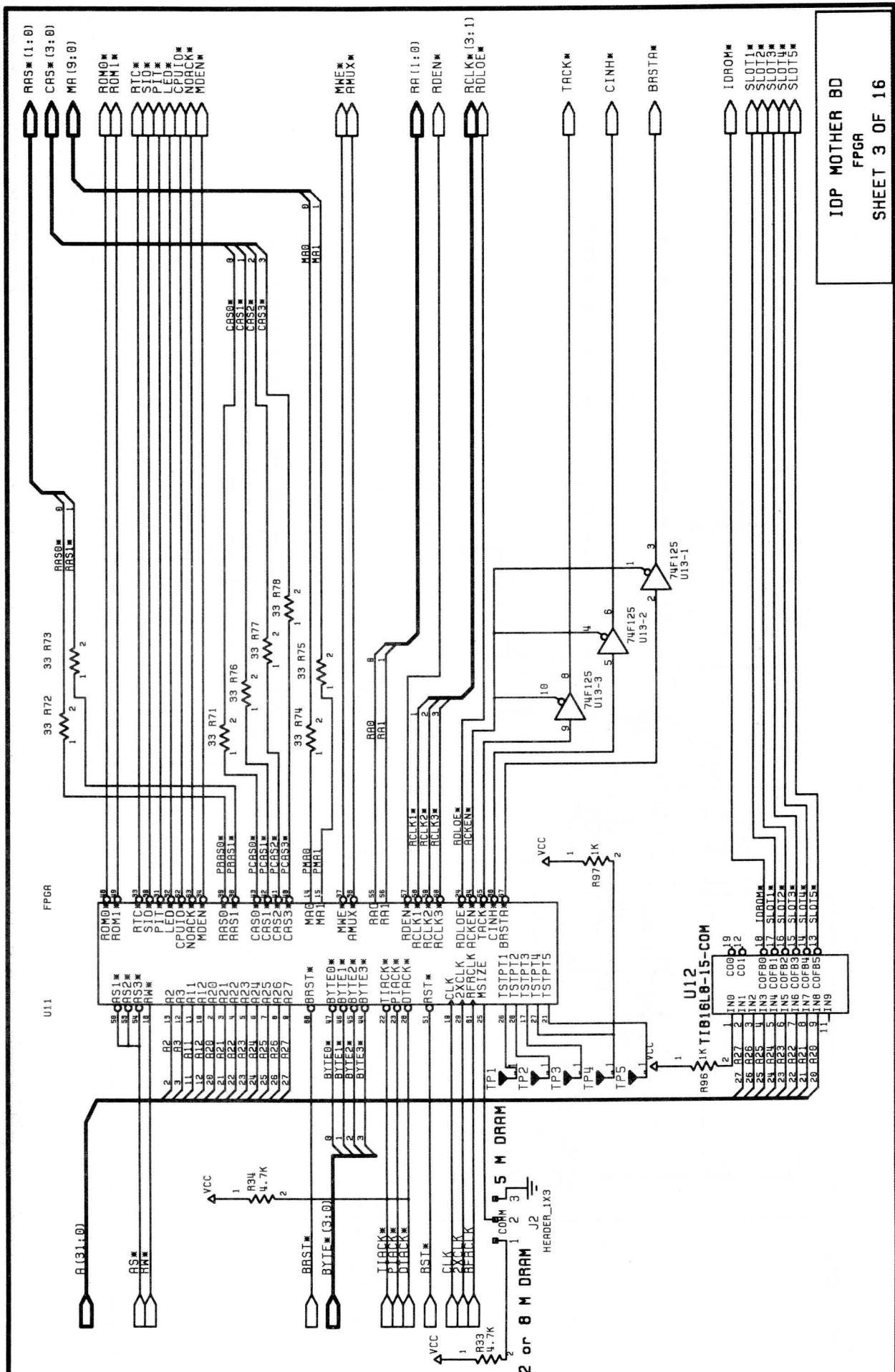


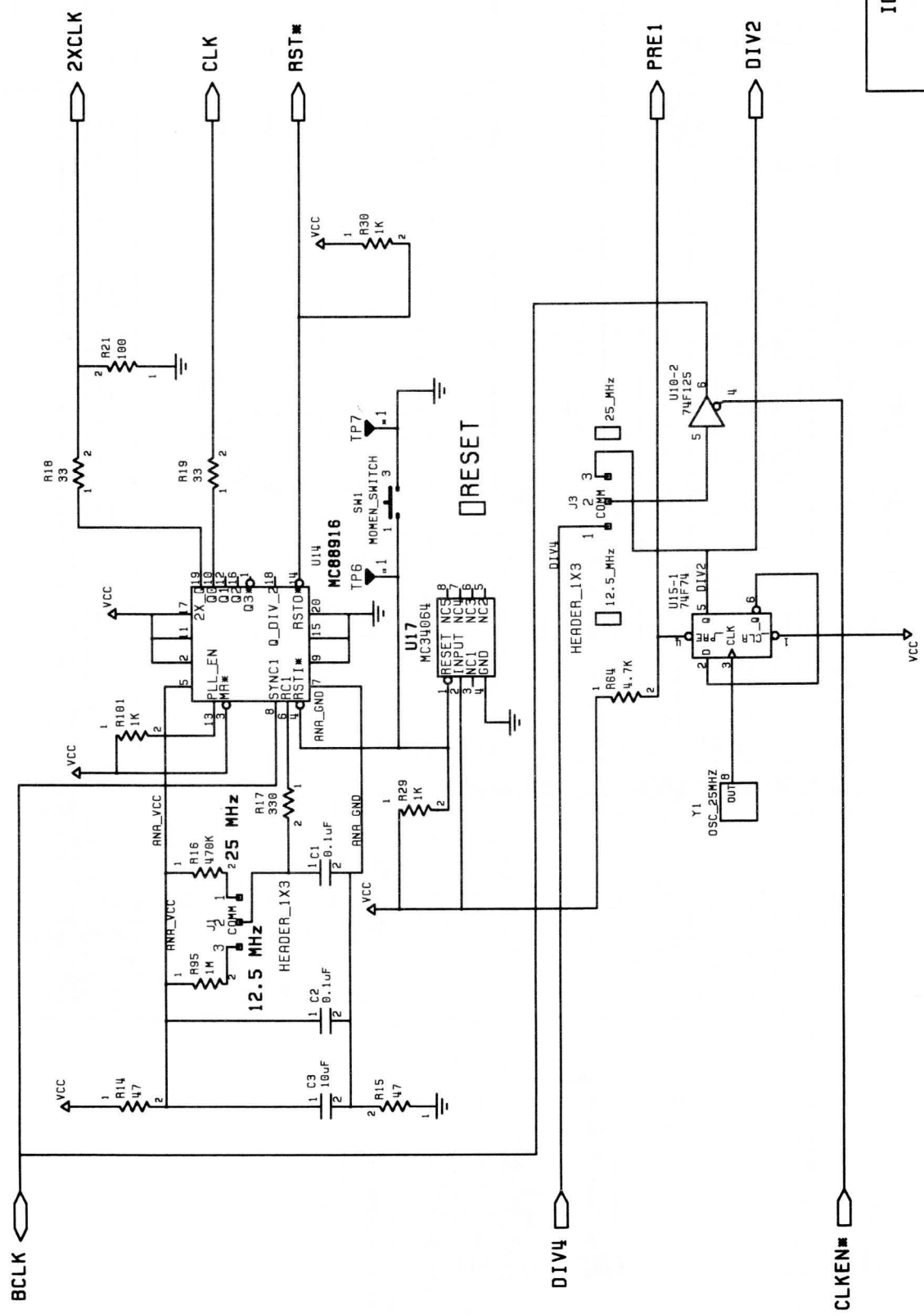


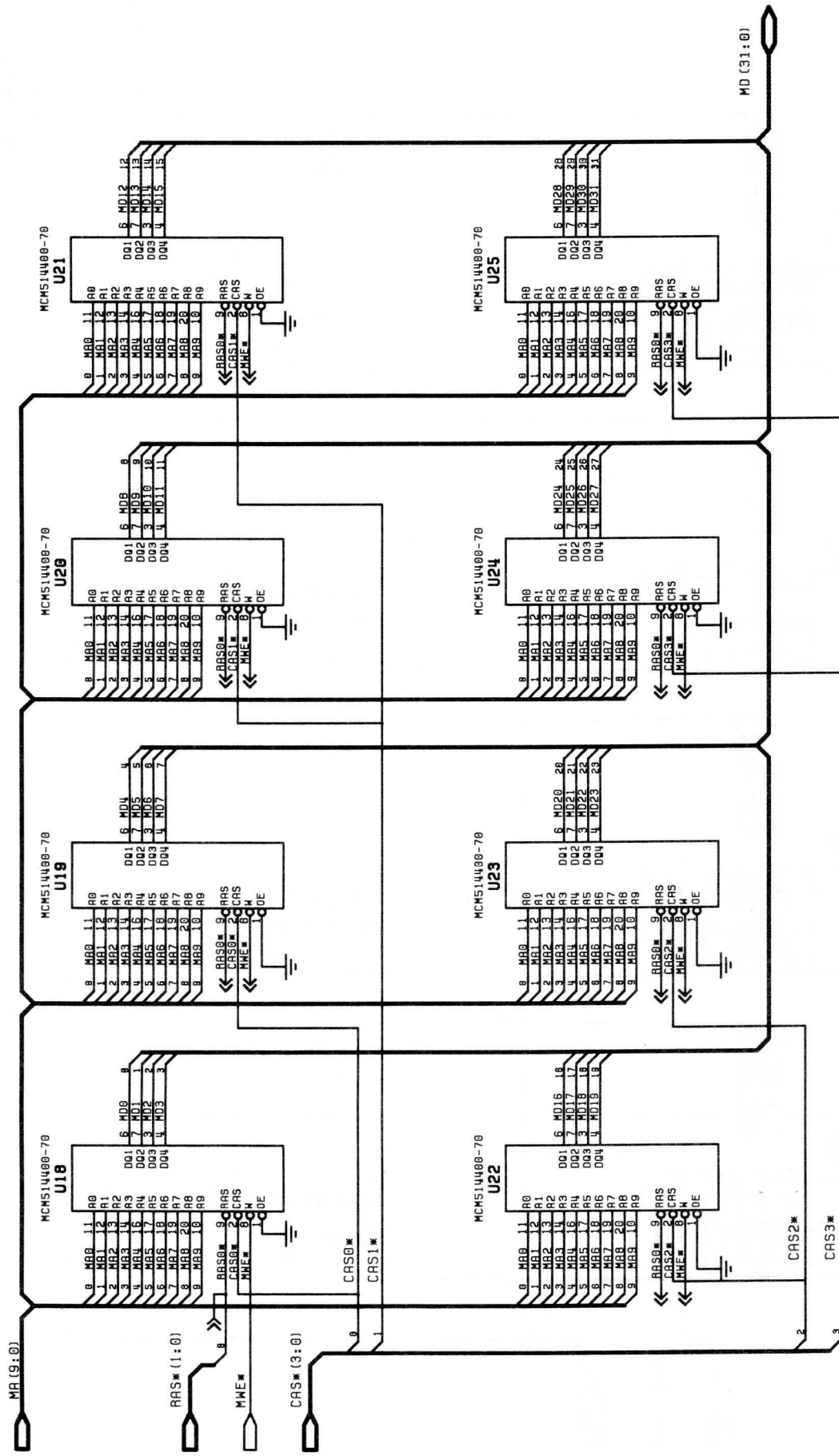
IDP MOTHER BD
P1_P2
SHEET 1 OF 16



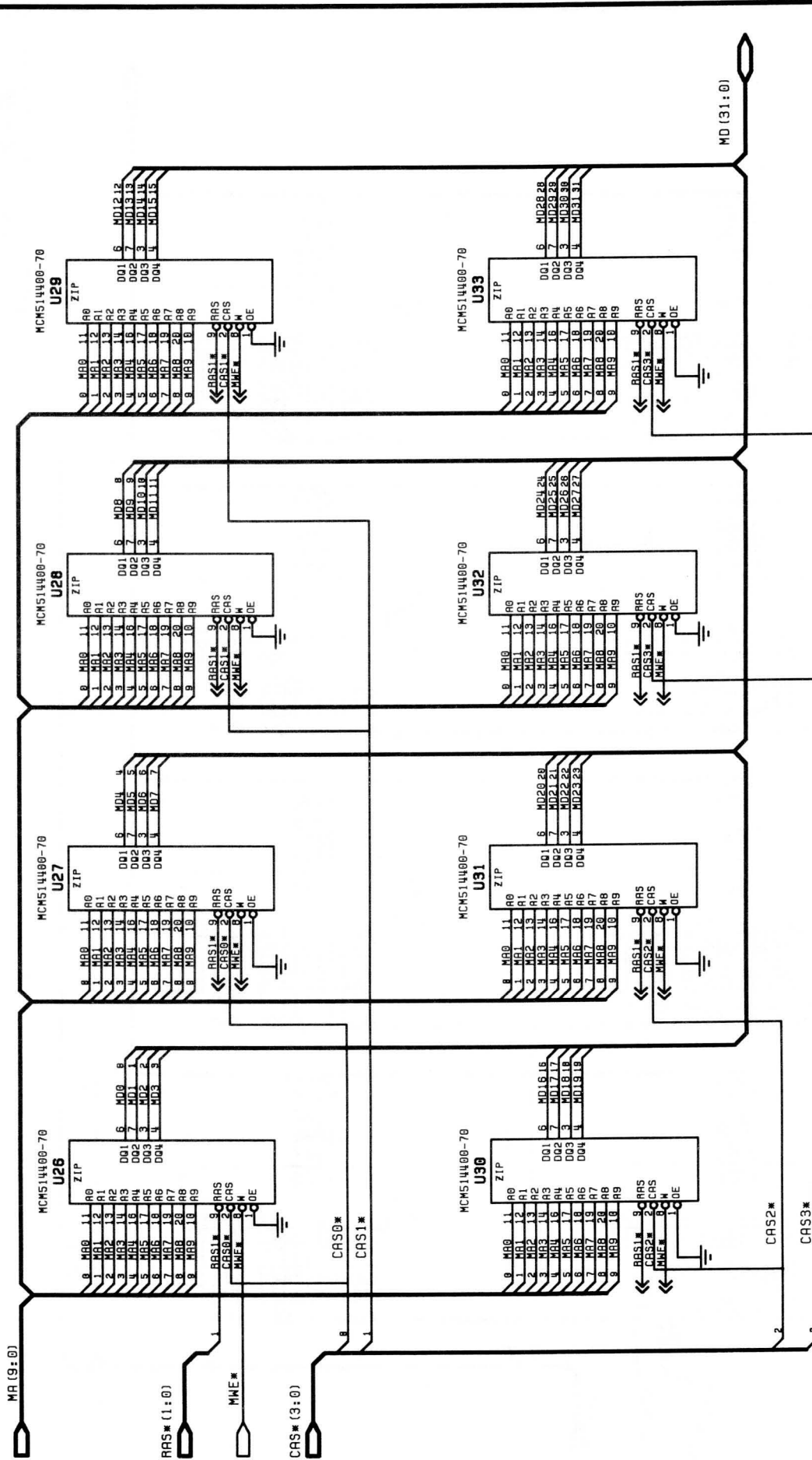
IDP MOTHER BD
SLOT_INTERFACE
SHEET 2 OF 16





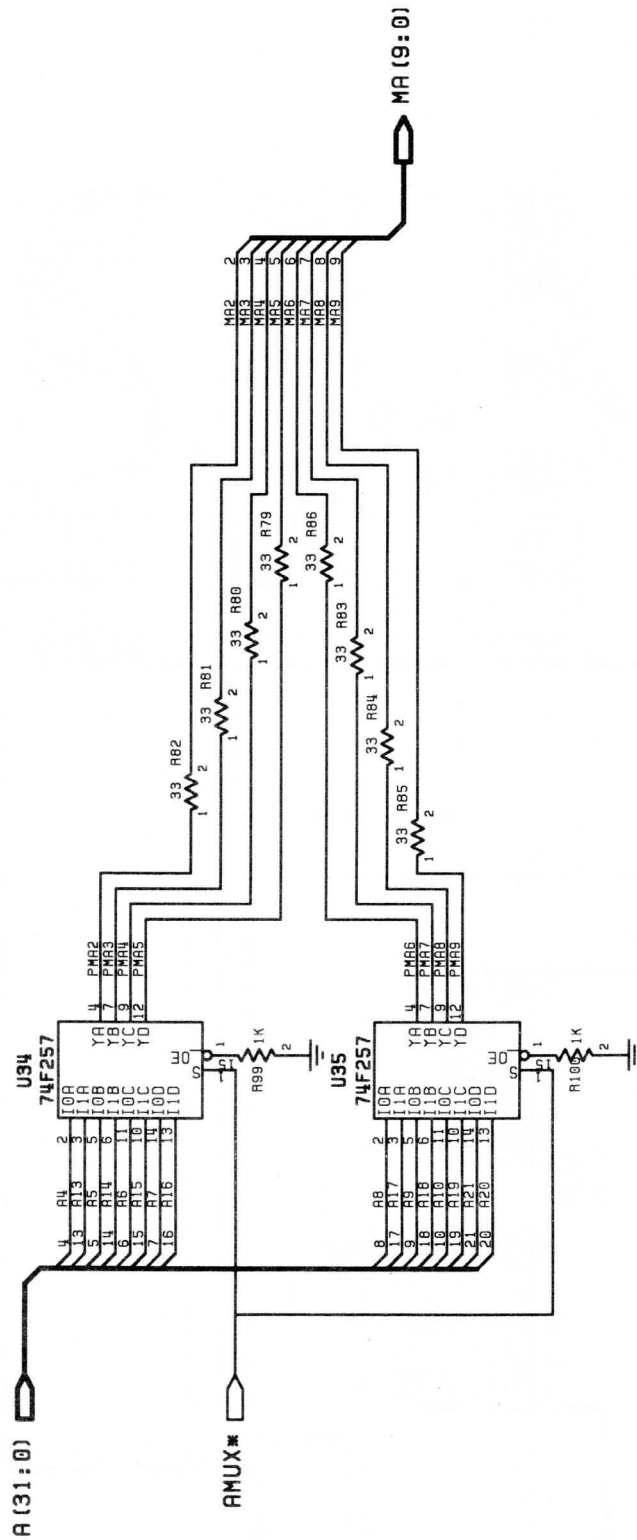


MD (31:0)



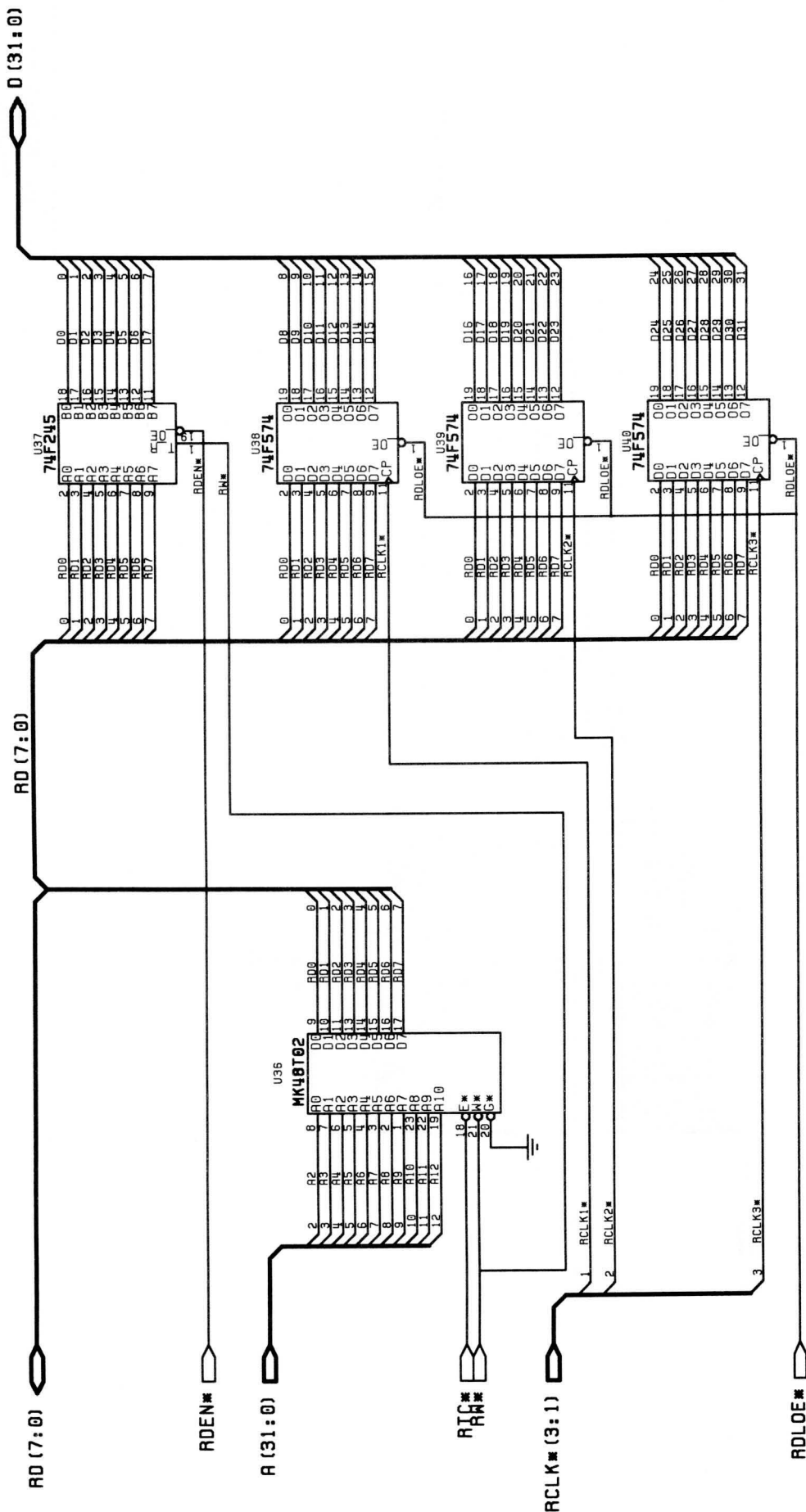
IDP MOTHER BD
MEM_ARRAY_2
SHEET 6 OF 16

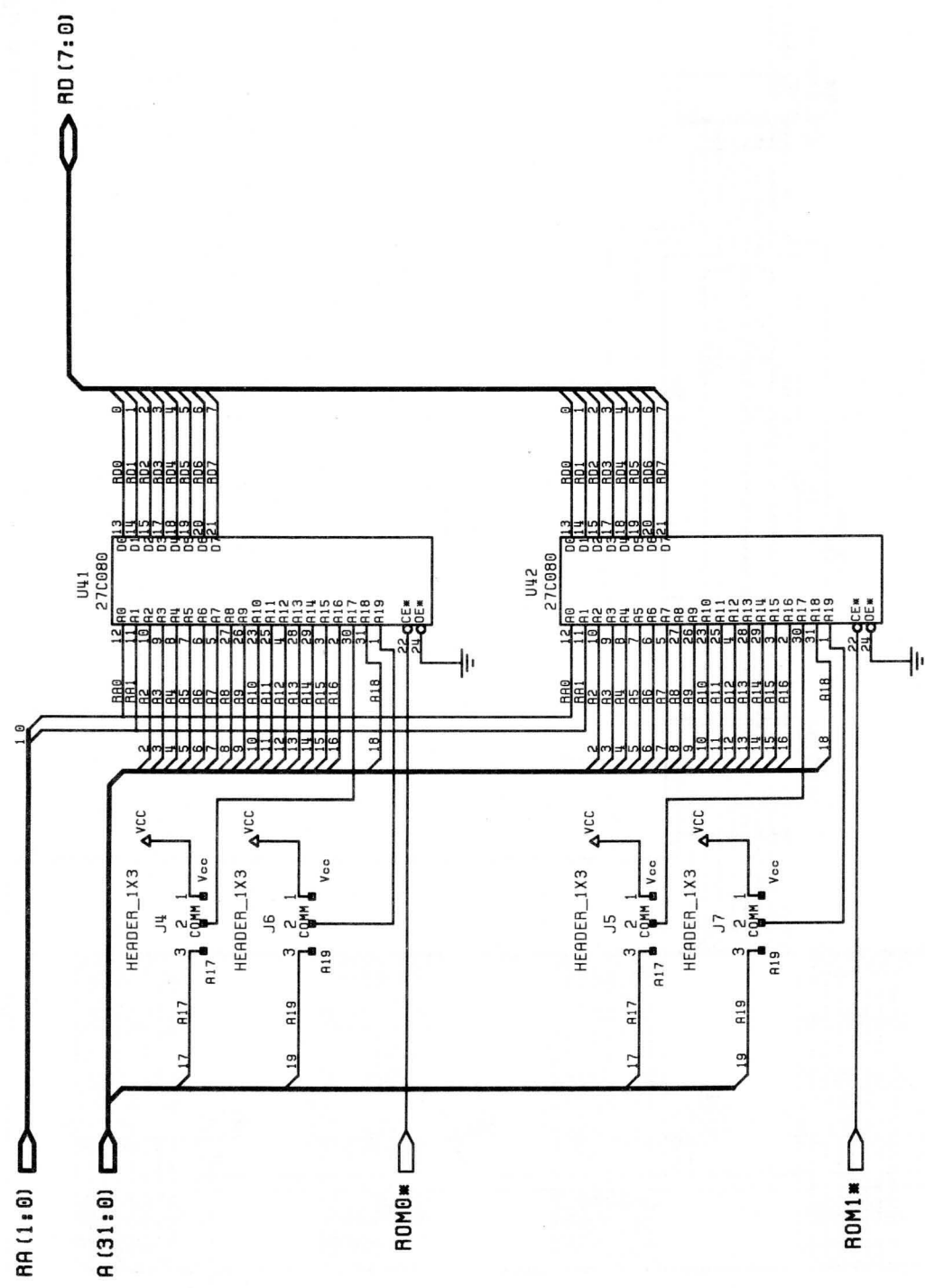
NULL



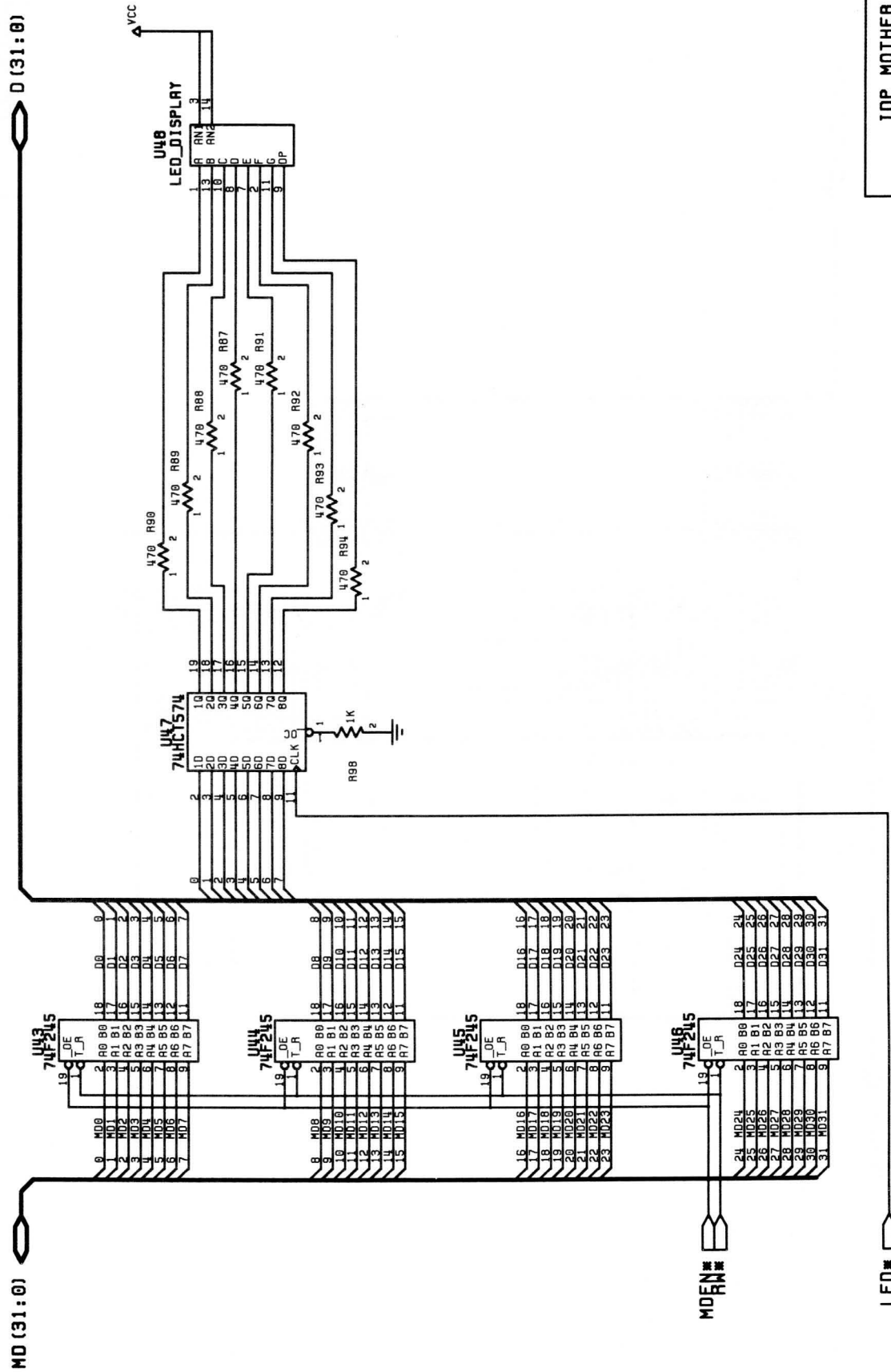
IDP MOTHER BD
MEM_RANDOM
SHEET 7 OF 16

NULL





IDP MOTHER BD
 EPROM
 SHEET 9 OF 16



MD (31:0)

D (31:0)

MDEN

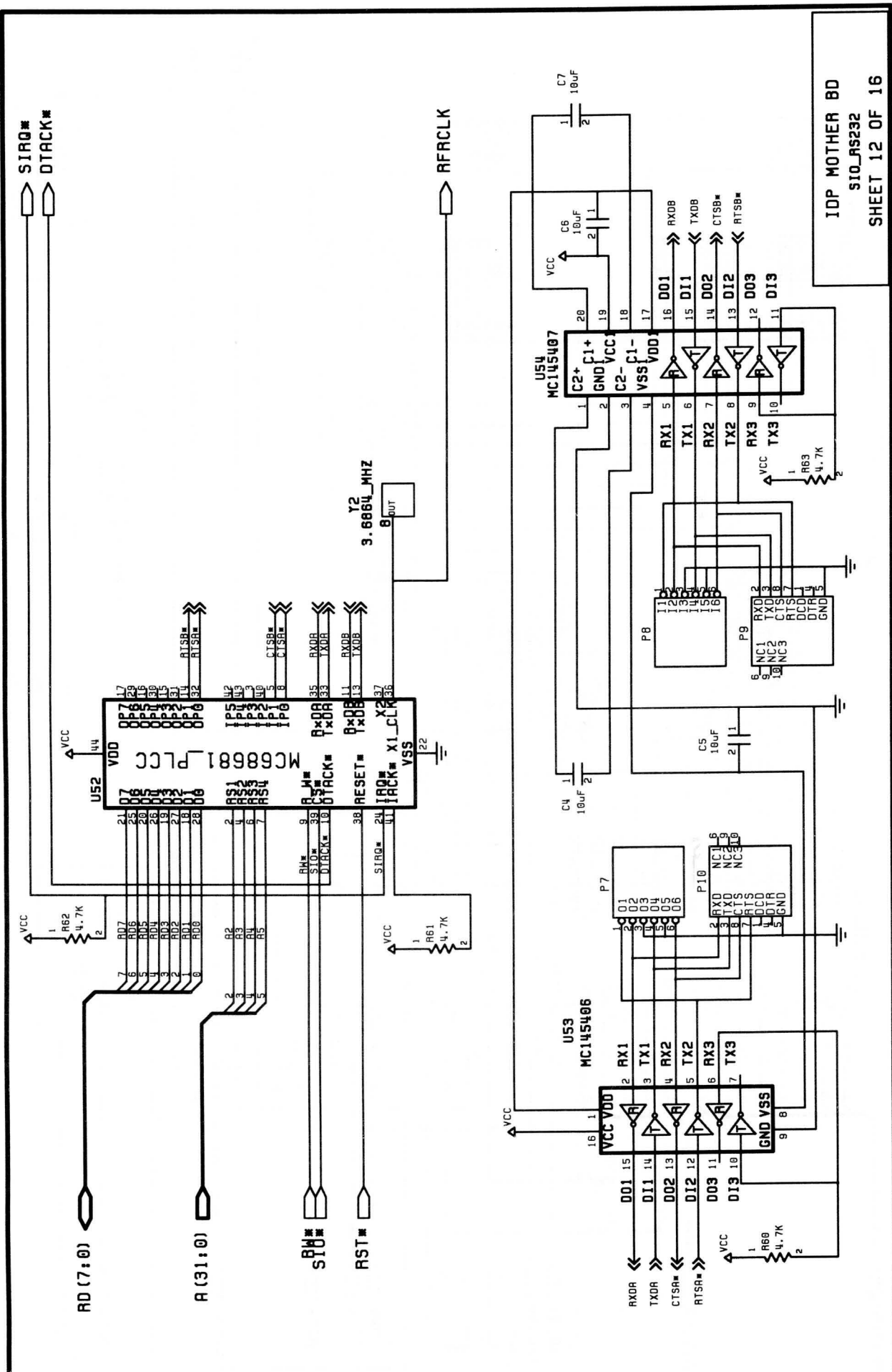
LED*

MDEN#

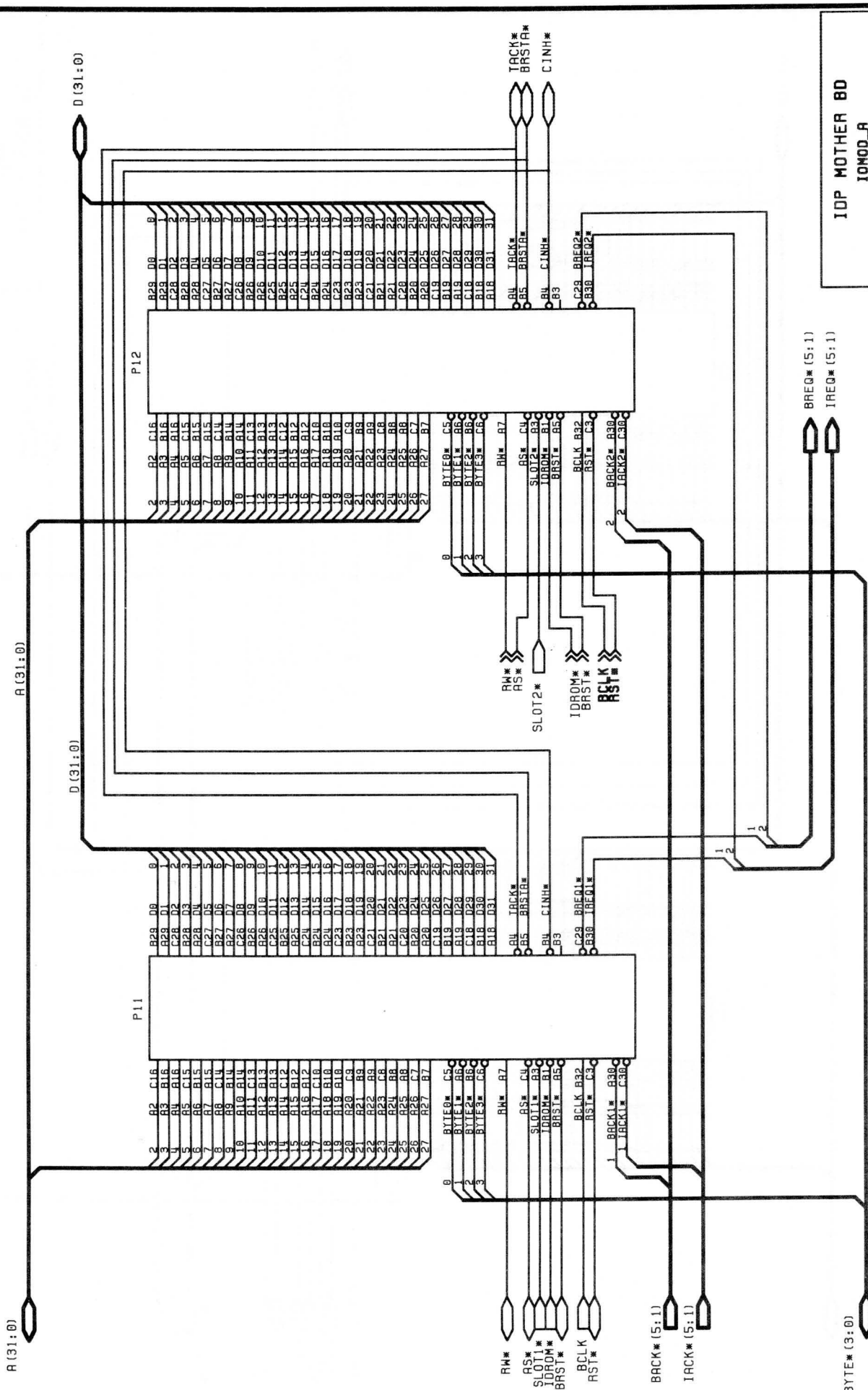
MDATA_BUFF

SHEET 10 OF 16

NULL

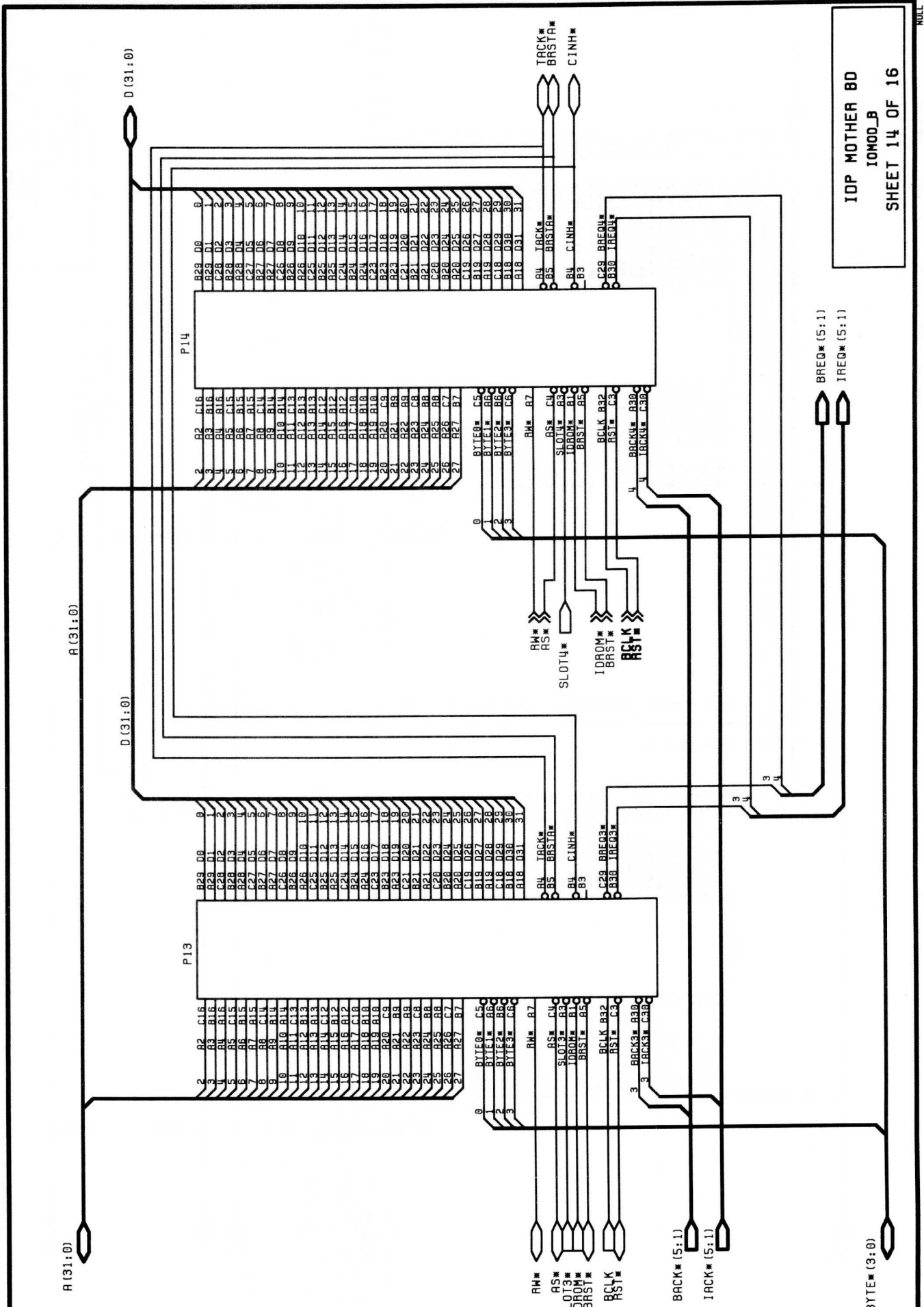


IDP MOTHER BD
S10_AS232
SHEET 12 OF 16



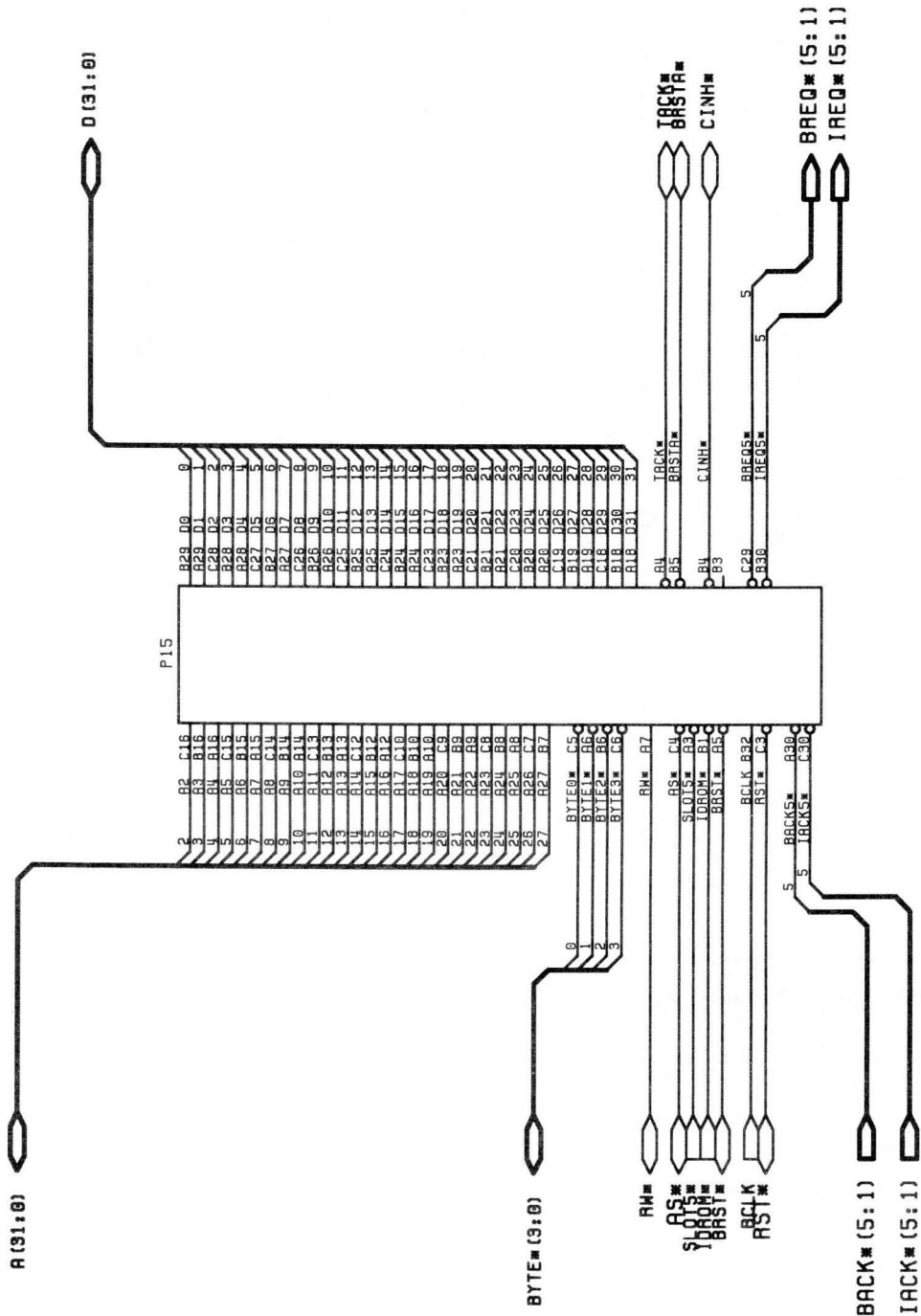
IOP MOTHER BD
IOMOD_A
SHEET 13 OF 16

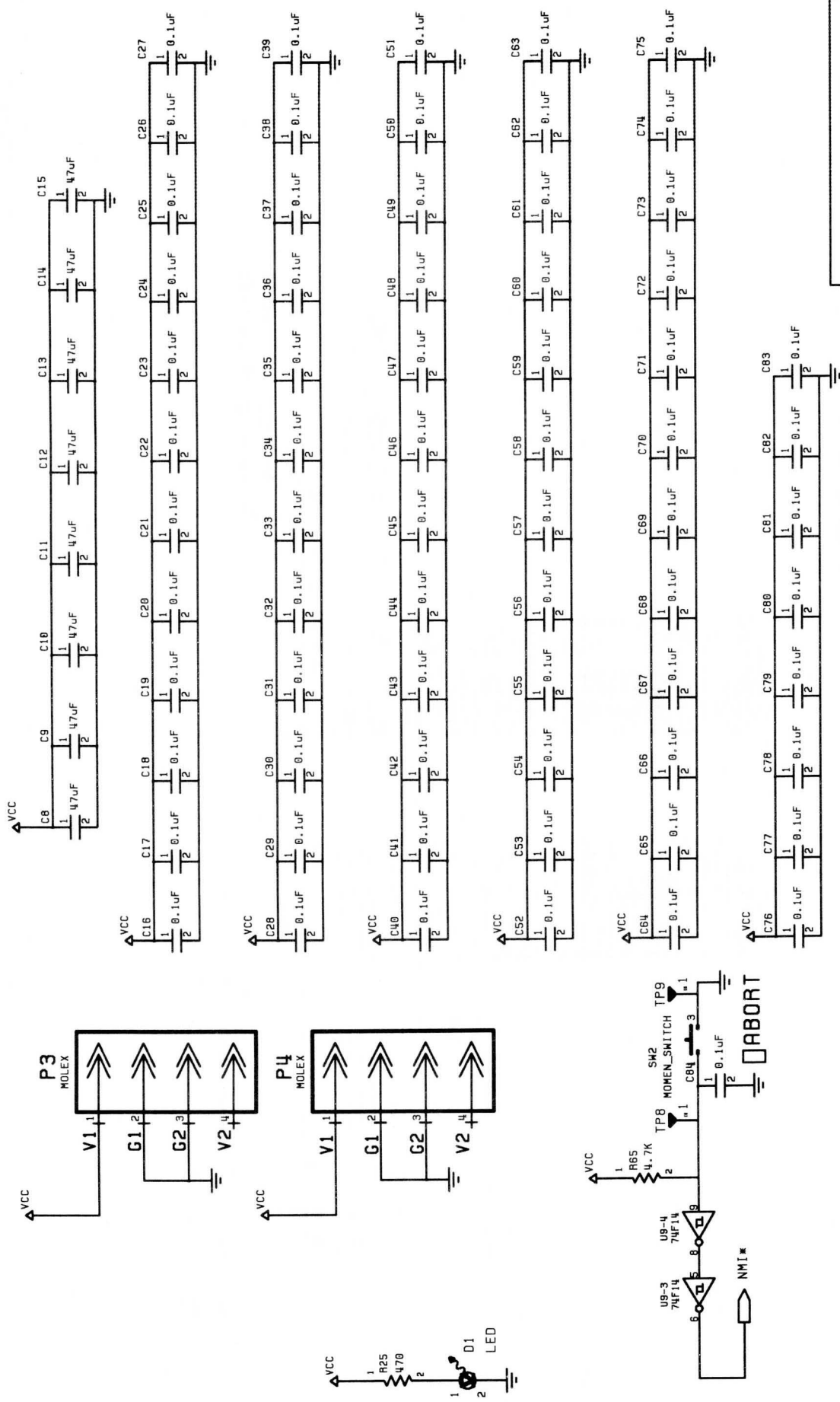
NOT



IDP MOTHER BD
 IOMOD_B
 SHEET 14 OF 16

NULL





IDP MOTHER BD
POWER_CAPS
SHEET 16 OF 16



Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No.2 Dai King Street, Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong.