M5206EC3 USER'S MANUAL
REVISION 1.2

Cadre III

A Framework for Solutions
4150 Freidrich Lane Suite D
Austin, Texas 78744
Support: (USA only): (800) 410-2031
(512) 326-9455
Email: support@cadreiii.com
Web: www@cadreiii.com

LIMITED WARRANTY

Cadre III warrants this product against defects in material and workmanship for a period of sixty (60) days from the original date of purchase. This warranty extends to the original customer only and is in lieu of all other warrants, including implied warranties of merchantability and fitness. In no event will the seller be liable for any incidental or consequential damages. During the warranty period, Cadre III will replace, at no charge, components that fail, provided the product is returned (properly packed and shipped prepaid) to Cadre III at the address below. Dated proof of purchase, such as a copy of the invoice, must be enclosed with the shipment. We will return the shipment prepaid via UPS.

This warranty does not apply if, in the opinion of Cadre III, the product has been damaged by accident, misuse, neglect, misapplication, or as a result of service or modification (other than specified in the manual) by others.

Please send the board and cables with a complete description of the problem to:

Cadre III
4150 Freidrich Lane, Suite D
Austin, Texas 78744

HELPFUL INFORMATION

Information for the MCF5206e processor and evaluation board is updated frequently at the following URL: http://www.motorola.com/ColdFire.
Visit http://www.motorola.com/ColdFire to obtain the follow information.

1. Source code for the assembler
2. Most current User manual for the MCF5206e processor
3. Most current User manual for the M5206EC3
4. Addendum for the MCF5206e
5. Application notes
6. Example code for the MCF5206e
7. B.O.M. for the M5206EC3

These example files are also available on the web site: S-REC, COFF, ELF.

Refer to the electronic version of this user manual at www.motorola.com/ColdFire for the most current information.

Want to find out how others are using ColdFire integrated microprocessors in their applications? Sign up at www.wildrice.com and follow the instructions.

Disclaimer

The information in this manual has been carefully examined and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, Motorola reserves the right to make changes to any product(s) herein to improve reliability, function, or design. The M5206EC3 board is not intended for use in life and/or property critical applications. Here, such applications are defined to be any situation in which any failure, malfunction, or unintended operation of the board could, directly, or indirectly, threaten life, result in personal injury, or cause damage to property. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible. Motorola Inc. will not assume responsibility for any damages incurred or generated by this product. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights, if any, or the rights of others.

Warning

This board generates, uses, and can radiate radio frequency energy and, if not installed properly, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for class a computing devices pursuant to Subpart J of Part 15 of the FCC rules, which provide reasonable protection against such interference. Operation of this product in a residential area is likely to cause interference, in which case users, at their own expense, will be required to correct the interference.

TABLE OF CONTENTS

TABLES

FIGURES

1 INTRODUCTION TO THE M5206EC3 BOARD

1.1 OVERVIEW

The M5206EC3 is a versatile single-board computer based on the MCF5206e ColdFire® processor, which you can use as a powerful microprocessor-based controller in a variety of applications. With the addition of a terminal, the M5206EC3 serves as a complete microcomputer for development/evaluation, training, and educational use. You just have to connect an RS-232-compatible terminal (or a personal computer with terminal emulation software) and a power supply to have a fully functional system.

Provisions have been made to connect this board to additional user-supplied boards via the Microprocessor Expansion Bus connectors to expand memory and I/O capabilities. Additional boards may require bus buffers to compensate for added bus loading.

Furthermore, the PC-board permits configuration in a way that best suits an application. Available features include: as much as 4 MBytes DRAM, 1 MByte of SRAM (not included), timer, serial and parallel I/O, Ethernet, DMA, I-cache, internal SRAM, chip select module, and 1 MByte of Flash. In addition, all of the signals are easily accessible to any logical analyzer with mictor probes or berg connectors to assist with debug. All of the processor's signals are also available via connectors J8 and J9 for expansion purposes.

1.2 GENERAL HARDWARE DESCRIPTION

The M5206EC3 board provides the RAM, Flash ROM, on-board NE2000-compatible Ethernet interface (10 Mbit/sec), RS-232, and all the built-in I/O functions of the MCF5206e for learning and evaluating the attributes of the MCF5206e. The MCF5206e ~~a~~ member of the ColdFire Family of processors ~~is a~~ 32-bit processor with 32 bits of addressing and 32 lines of data. The processor has eight 32-bit data registers, eight 32-bit address registers, a 32-bit program counter, and a 16-bit status register.

The MCF5206e has a System Integration Module (SIM) that incorporates many system design functions, such as programmable chip-select logic, system protection logic, general-purpose I/O, and interrupt controller logic. The chip-select logic can select as many as eight memory banks or peripherals and the DRAM controller allows a glueless interface to two banks of DRAMs. The chip-select logic also allows a programmable number of wait states for using slower memory (refer to *MCF5206e User's Manual* , downloadable at http://www.Motorola.com/ColdFire, for detail information about the SIM.) The M5206EC3 only uses three of the chip selects to access the Flash ROMs, SRAM (which is not populated on board, but you can add later) and the Ethernet. The DRAM controller controls one SIMM module, 4 MBytes of DRAM, both RAS lines, and all four CAS lines are used. All other functions of the SIM are available.

A hardware watchdog timer (bus monitor) circuit is included in the SIM that monitors the bus activities. If a bus cycle is not terminated within a programmable time, the watchdog timer will assert an internal transfer error signal to terminate the bus cycle. The ROM monitor never uses the hardware watchdog timer feature but it is available to enable it in your code.

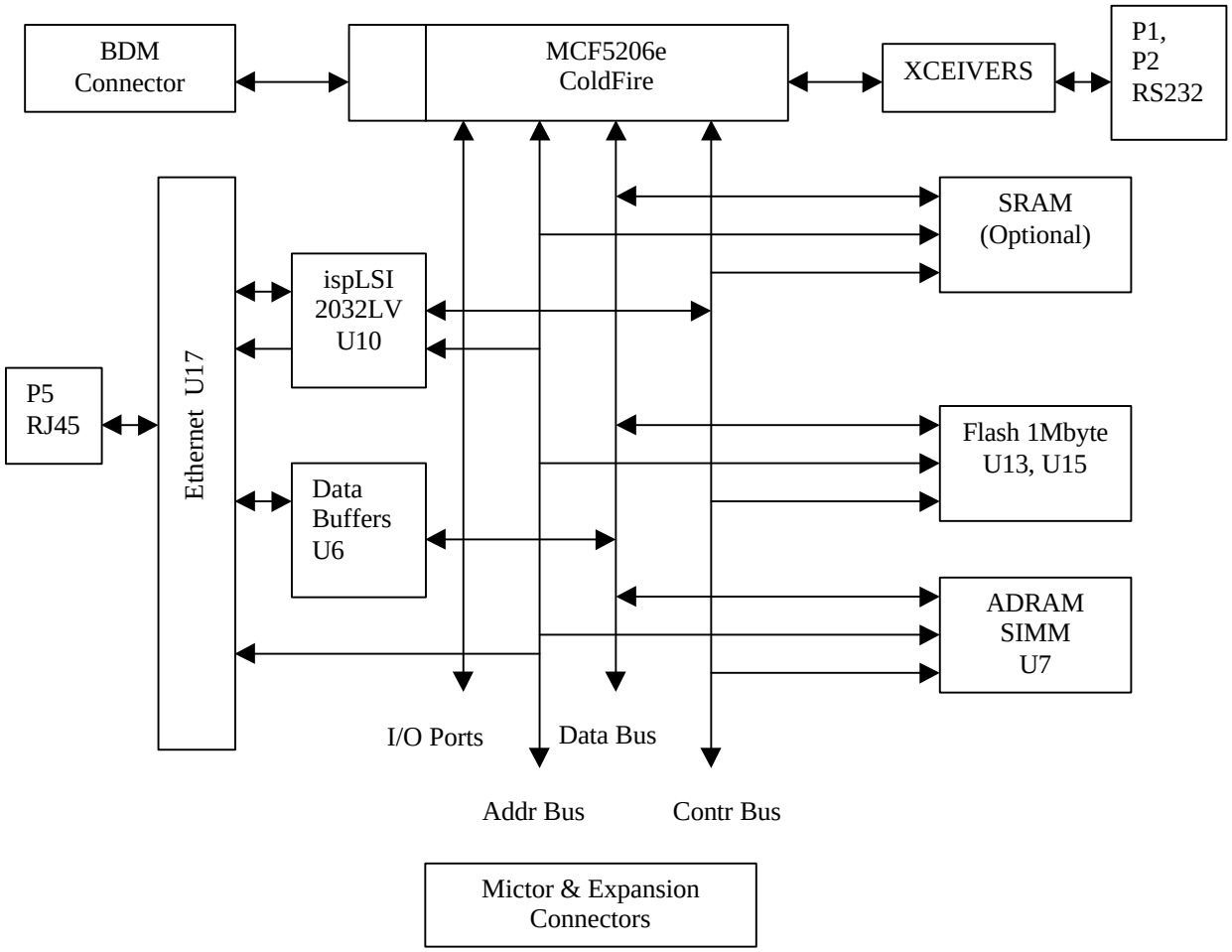


Figure 1. Block Diagram of the M5206EC3 Board

1.3 SYSTEM MEMORY

There are two on-board Flash ROMs (U13, U15). U13 is the most significant byte; U15, the least significant byte. The M5206EC3 comes with two 29LV004 Flash ROMs programmed with a debugger/monitor firmware. Both AM29LV004DT Flash are 4 Mbits, each giving a total of 1 MByte of Flash memory. The Debug only supports 29LV004 Flash ROM.

The one 72-pin SIMM socket for ADRAM supports as much as 32 MBytes of 3.3V ADRAM. The board comes with 4 MBytes of 3.3V ADRAM installed.

The MCF5206e has 8 KBytes organized as 2048x32 bits of internal SRAM.

The internal cache of the MCF5206e is a nonblocking 4 KByte direct-mapped instruction cache. The ROM monitor currently does not use the cache, but user code can enable and use the I-cache.

1.4 SERIAL COMMUNICATION CHANNELS

The MCF5206e has two built-in UARTs with independent baud-rate generators. The signals of channel one are passed through external driver/receivers to make the channel RS-232 compatible. The debugger uses UART1 to let you access with a terminal. In addition, the signals of both channels are available on the mictor connectors LA1 and LA3 to be viewed by a logic analyzer. The UART1 channel is the TERMINAL channel the debugger uses for communicating with the external terminal/PC. The TERMINAL baud rate is set at 19200.

The MCF5206e also incorporates the M-Bus, which is compatible with I^2C bus standard.

1.5 PARALLEL I/O PORTS

The MCF5206e offers one 8-bit general-purpose parallel I/O port. Each pin can be individually programmed as input or output. The parallel port bits PP(3:0) are multiplexed with PST(3:0) and PP(7:4) are multiplexed with DDATA(3:0). The Pin Assignment Register (PAR) controls both nibbles of the parallel port. After reset, all pins are configured as general-purpose parallel I/O. The ROM monitor configures the pins as PST(3:0) and DDATA(3:0).

1.6 PROGRAMMABLE TIMER/COUNTER

The MCF5206e has two built-in general-purpose 16-bit timer/counters. The MCF5206EC3 ROM monitor does not use these timers, so they are available for you to use. The signals for the timer are available on the LA1 and J2.

1.7 ON-BOARD ETHERNET

The M5206EC3 has an on-board Ethernet (NE2000 compatible) operating at 10 Mbits. The on-board ROM monitor is programmed to perform fast downs from a network to memory in S-Record, COFF, or ELF. See the Ethernet section in the appendix for more information.

1.8 SYSTEM CONFIGURATION

The M5206EC3 board requires only the following items for minimum system configuration (see Figure 3):

1. The M5206EC3 board (provided)
2. Power supply, 7.5V to 9V with minimum of 1.5 amp
3. RS-232C-compatible terminal or any computer with terminal emulation software and an RS-232 port
4. Communication cable (provided)

1.9 INSTALLATION AND SETUP

The following sections describe all the steps needed to prepare the board for operation. Please read the following sections carefully before using the board. When you are preparing the board for the first time, be sure to check that all jumpers are in the default locations. After the board is functional in its standard configuration, you can use the Ethernet by following the instructions provided in the following sections.

1.9.1 Unpacking

1. Unpack the computer board from its shipping box.
2. Save the box for storing or reshipping.
3. Refer to the following list and verify that all the items are present. You should have received:

    a. M5206EC3 single board computer
    b. M5206EC3 user's manual, this documentation
    c. One serial (RS-232) communication cable
    d. One Computer Systems BDM wiggler cable

WARNING
AVOID TOUCHING THE MOS DEVICES. STATIC DISCHARGE CAN AND WILL DAMAGE THESE DEVICES.

Once you verified that all the items are present:

1. Remove the board from its protective jacket.
2. Check the board for any visible damage and ensure that there are no broken, damaged, or missing parts. If you have not received all the items listed above or they are damaged, please contact Cadre III immediately in order to correct the problem.

Cadre III
4150 Freidrich Lane, Suite D
Austin, Texas 78744
Support: (USA only): (800) 410-2031
(512) 326-9455

1.9.2 Preparing the Board for Use

The board as shipped is ready to be connected to a terminal and the power supply without any modification. However, follow the steps below to ensure proper operation from the first time you apply the power. Figure 4 shows the placement of the jumpers and the connectors, and section 1.11 explains the default jumper settings.

1.9.3 Providing Power to the Board

The board accepts two types of power supply connections. Connector P4 is a 2.1mm barrel connector power jack with center positive and P3 is a lever-actuated connector for bare-wire insertion. The board accepts 7.5V to 9V DC (regulated or unregulated) at 1.5 amp through either one of the connectors (see below). Power supplied to the processor passes through jumpers J6 and J7 (**note: power connected to the pullup resistors attached to the processor does not pass through J6 or J7**). Both J6 and J7 are in parallel with each other and can perform power analysis.

Note: On boards labeled "Rev 1.2," the silkscreen for D9 and D10 are incorrect. D9 should be labeled +3.3V and D10 should be labeled +5V. For those board revisions higher than Rev 1.2, ignore this note.

Contact NO.	Voltage
1	+7.5-9V
2	Ground

1.9.4 Selecting Terminal Baud Rate

The MCF5206e serial channel used for serial communication has a built-in timer the ROM monitor uses to generate the baud rate for terminal communication. You can program the serial channel to several baud rates. After power-up or a manual reset, the ROM monitor firmware configures the UART channel 1 for 19200 baud. Once the ROM monitor is running, you can issue the SET command to choose any baud rate the ROM monitor supports. Refer to Chapter 2 for more information on this command.

1.9.5 The Terminal Character Format

The character format of the communication channel is fixed at power -up or reset. The character format is 8 bits per character, no parity, and one stop bit. Make sure your terminal is set to this format. Handshaking is set to none.

1.9.6 Connecting the Terminal

Use the RS-232 serial cable to connect the terminal to the M5206EC3. The cable has a 9-pin female D subconnector at one end and a 9-pin male D subconnector at the other end (see Figure 2). Connect the 9-pin male connector to P1 connector on M5206EC3. Connect the 9-pin female connector to one of the available serial communication channels normally referred to as COMx (COM1, COM2, etc.) on the IBM PC or compatible machine. Depending on the kind of serial connector on the back of your PC, that connector may be a male 25-pin or 9-pin. 9-pin-to-25-pin adapters are available at most electronics stores.

1.9.7 Using a Personal Computer as a Terminal

You can use your personal computer as a terminal provided you also have installed terminal emulation software such as PROCOMM, KERMIT, QMODEM, Windows 95® Hyper Terminal or similar packages. Connect as described in **1.9.6 Connecting the Terminal**.

Once the connection to the PC is made, you are ready to power-up the PC and run the terminal emulation software. When the PC is in terminal mode, you need to select the baud rate and the character format for the channel. Most terminal emulation software packages provide a command known as "Alt-p" (press the p key while pressing the Alt key) to choose the baud rate and character format. Select 8 bits, no parity, one stop bit. Then, select the baud rate as 19200. Now apply power to the board.

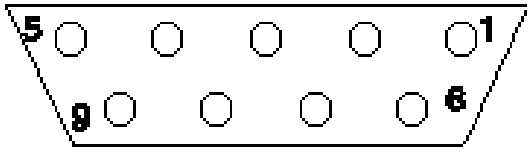


Figure 2. Pin assignment for P1 (Terminal) connector

1. Data Carrier Detect, Output (shorted to pins 4 and 6)
2. Receive Data, Output from board (receive refers to terminal side)
3. Transmit Data, Input to board (transmit refers to terminal side)
4. Data Terminal Ready, input (shorted to pin 1 and 6)
5. Signal Ground
6. Data Set Ready, Output (shorted to pins 1 and 4).
7. Request to Send, input
8. Clear to send, output
9. Not connected

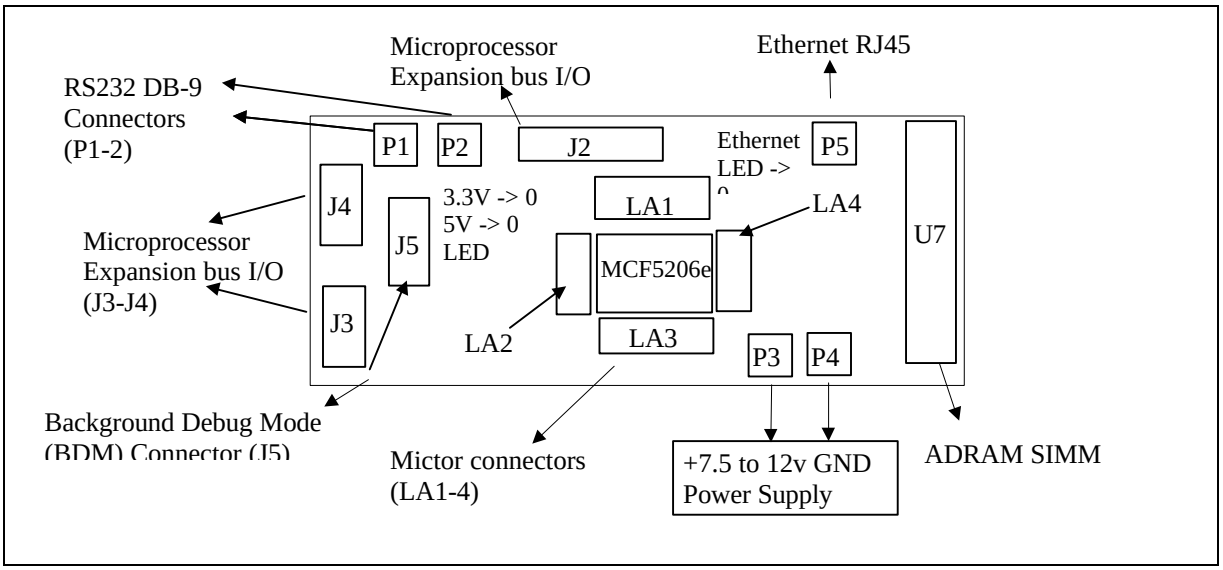


Figure 3. System Configuration

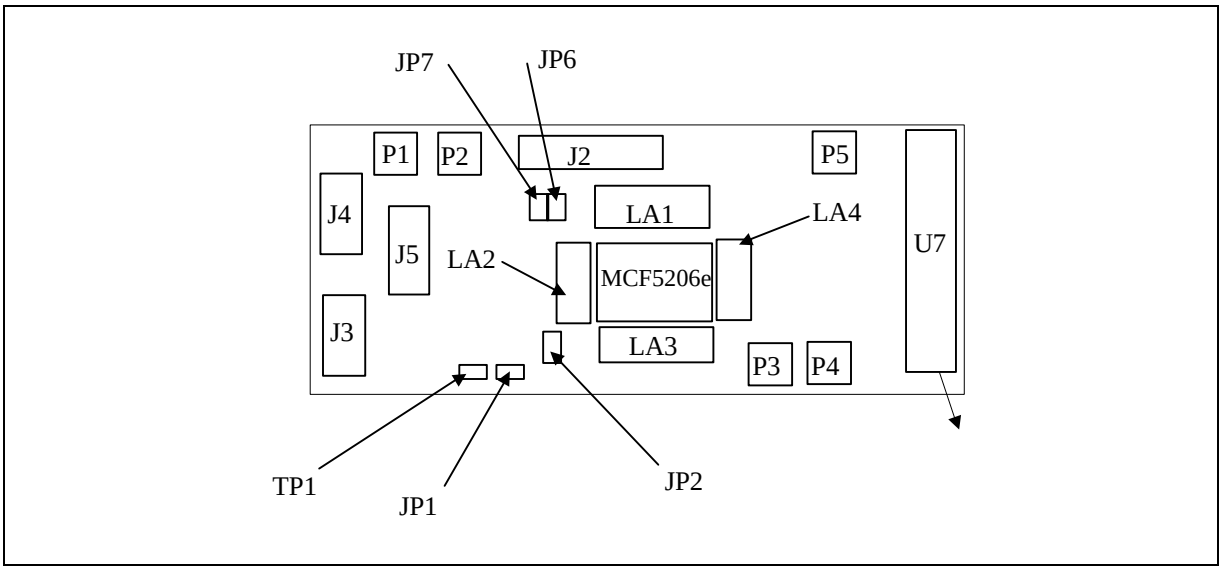


Figure 4. Jumper and Connector Placement

1.10 SYSTEM POWER-UP AND INITIAL OPERATION

Now that you have connected all the cables, you can apply power to the board. After power is applied, dBUG initializes the board then displays the power-up message on the terminal, which includes the amount of the memory present.

```
Hard Reset
DRAM Size: 4M

Copyright 1997-1998 Motorola, Inc.  All Rights Reserved.
ColdFire® MCF5206e EVS Debugger V1.4.1 (JUL 1998 12:10:48:)
Enter ‘help’ for help.

dBUG>
```

Note: You can download from the web any updates to the ROM Monitor.

The board is now ready for operation under the control of the debugger as described in Chapter 2. If you do not receive the above response, perform the following checks:

1. Make sure that the power supply is properly set and connected to the board.
2. Check that both LEDs D9 and D10 are lit (the board requires a minimum of 7.5 to 9 V DC).
3. Check that the terminal and board are set for the same character format and baud rate.
4. Press the black RESET button to ensure that the board has been initialized properly.

If you still are not receiving the proper response, your board may have been damaged in shipping. Contact Cadre III for further instructions.

1.11 M5206EC3 JUMPER SETUP

The jumpers on the board are discussed in Chapter 3. However, here's a brief discussion of the jumper settings.

1.11.1 **Jumper JP1**

This jumper selects between -CS0 to Flash or a header.

Table 1. JP1, -CS0 Select

JP1	FUNCTION
1 and 2	Flash (default)
2 and 3	Header (TP1)

1.11.2 Jumper JP2 - Flash Upper Half/Lower Half Boot

This jumper allows the MCF5206e to boot from the lower or upper half of the flash. The default is the lower half. Refer to Section 3.3.1 for information on using this jumper.

Table 2. JP2, Upper/Lower Half BOOT

JP2	FUNCTION
1 and 2	Lower (default)
2 and 3	Upper

1.11.3 Jumper J6 and J7 - CPU Power JP6 and 7

These jumpers pass power to the ColdFire CPU. Without a minimum of one jumper, the CPU will not get any power.

JP6	JP7	FUNCTION
ON	ON	Power (default)
OFF	OFF	No Power

1.12 USING THE BDM

The MCF5206e has a built-in debug mechanism referred to as BDM that uses the J5 header.

The BDM cable (provided) is to be used with third-party developer software tools such as SDS or P&E. For a current list of third-party development tools, visit the Motorola ColdFire web site at http://www.motorola.com/ColdFire. The BDM cable connects to the parallel port of a computer and to the MC5206EC3 J5 header.

IMPORTANT: There is no key to protect the BDM cable from being rotated and plugged in incorrectly. To prevent hooking up the BDM cable incorrectly, notice pin 1 on the cable and the notation on the board. A red strip on the ribbon cable normally notes which side of the cable pin 1 is located. There is also a pin-1 marking on the board near the connector.

2 USING THE MONITOR/DEBUG FIRMWARE

The M5206EC3 computer board has a resident firmware package that provides a self-contained programming and operating environment. The firmware, named dBUG, provides you with monitor/debug, disassembly, program download, and I/O control functions. This chapter explains how to use the dBUG package, including the user interface and command structure.

2.1 WHAT IS dBUG?

The dBUG package is a resident firmware package for the ColdFire® Family evaluation boards. The firmware (stored in two 512Kx8 Flash ROM devices) provides a self-contained programming and operating environment. The dBUG package interacts with you through predefined commands that are entered from the terminal.

The user interface to dBUG is the command line. A number of features have been implemented to achieve an easy and intuitive command line interface.

The dBUG package assumes that an 80x24 character dumb terminal is used to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit, 8N1. The baud rate is 19200 but can be changed after the power-up using the SET command (see Section 2.4.21).

The command line prompt is "dBUG>". Enter any dBUG command from this prompt. Command lines cannot exceed 80 characters in length. Wherever possible, dBUG displays data in 80 columns or less. The dBUG echoes each character as you type them, eliminating the need for any "local echo" on the terminal side.

In general, dBUG is not case sensitive. You can enter commands in either upper or lower case. Only symbol names require the exact case.

Most commands can be recognized by using an abbreviated name. For instance, entering "h" is the same as entering "help." Therefore, it is not necessary to type the entire command name.

The commands DI, GO, MD, STEP and TRACE are used repeatedly when debugging. The dBUG recognizes these commands and allows for repeated execution of these commands with minimal typing. After a command is entered, simply press <RETURN> or <ENTER> to invoke the command again. The command is executed as if no command line parameters were provided.

An additional function called the "TRAP 15 handler" lets you program various routines within dBUG. The TRAP 15 handler is discussed at the end of this chapter.

The operational mode of dBUG is demonstrated in Figure 5. After system initialization, the board waits for a command-line input from the user terminal. When a proper command is entered, the operation continues in one of the two basic modes. If the command causes execution of the user program, the dBUG firmware may or may not be re-entered, depending on programming of the user program (see Section 2.5.4). For the other mode, the command will be executed under

control of the dBUG firmware, and after command completion, the system will return to command-entry mode.

During command execution, additional user input may be required depending on the command function.

For commands that accept an optional <width> to modify the memory access size, the valid values are as follows:

.B	8-bit (byte) access
.W	16-bit (word) access
.L	32-bit (long) access

When no <width> option is provided, the default width is .W, 16-bit.

The dBUG maintains the core ColdFire register set. These are listed below.

A0-A7
D0-D7
PC
SR

All control registers on ColdFire are not readable by the supervisor-programming model, and thus not accessible via dBUG. User code can change these registers, but be careful as changes may render dBUG useless.

A reference to "SP" actually refers to "A7."

2.2 OPERATIONAL PROCEDURE

2.2.1 System Power-Up

a. Be sure the power supply is connected properly prior to power-up.
b. Make sure the terminal is connected to the terminal (P1) connector.
c. Turn power on to the board.

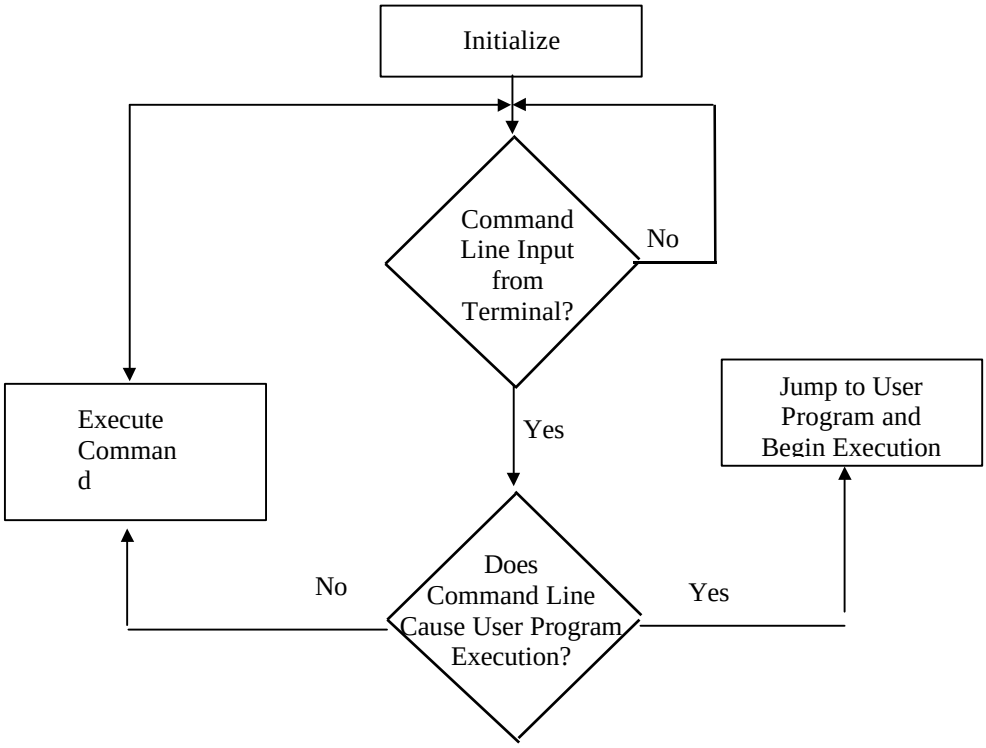


Figure 5. Flow Diagram of dBUG Operational Mode

2.2.2 System Initialization

Powering up the board will initialize the system. The processor is reset and dBUG is invoked.

The dBUG performs the following configurations of internal resources during the initialization. The instruction cache is invalidated and disabled. The Vector Base Register, VBR, points to the Flash. However, a copy of the exception table is made at address $00000000 in ADRAM. To take over an exception vector, place the address of the exception handler in the appropriate vector in the vector table located at 0x00000000, and then point the VBR to 0x00000000. The software watchdog timer is disabled, bus monitor is enabled, and internal timers are placed in a stop condition. Interrupt controller registers are initialized with unique interrupt level/priority pairs.

After initialization, the terminal will display the following:

```
Hard Reset
DRAM Size: 4M
NE2000: 0x300

Copyright 1997-1998 Motorola, Inc.  All Rights Reserved.
ColdFire® MCF5206e EVS Debugger Vx.x.x (xxx 199x xx:xx:xx:)
Enter ‘help’ for help.

dBUG>
```

If you did not receive this response, recheck the setup. Refer to **Section 1.10 System Power-Up And Initial Operation**. Note, the date `"xxx 199x xx:xx:xx"` may vary in different revisions.

You can re-initialize the M5206EC3 computer board firmware using other methods, which are discussed in the following paragraphs.

2.2.2.1 HARD RESET BUTTON

Hard RESET is the red button located in the lower right side of the board. Depressing this button terminates all processes, resets the MCF5206e processor and board logic, and restarts the dBUG firmware. Pressing the RESET button would be the appropriate action if all else fails.

2.2.2.2 ABORT BUTTON

ABORT is the black button located next to RESET button on the right side of the board. The abort function interrupts the present processing (a level 7 interrupt on MCF5206e) and passes control to the dBUG firmware. This action differs from RESET in that no processor register or memory contents are changed, the processor and peripherals are not reset, and dBUG is not restarted. Also, in response to depressing the ABORT button, the contents of the MCF5206e core internal registers are displayed.

The abort function is most appropriate when software is being debugged. You can interrupt the processor without destroying the present state of the system.

2.2.2.3 SOFTWARE RESET COMMAND

The dBUG does have a command that restarts the dBUG as if a hardware reset was invoked. The command is "RESET."

2.2.2.4 USER MEMORY ADDRESS

User memory is located at addresses $00020000-$xxxxxxxx, where $xxxxxxxx is the maximum RAM address of the memory installed in the board. When first learning the system, you should limit your activities to this area of the memory map. The dBUG uses the address range $00000000-$0001FFFF. The memory map is documented in detail in Section 3.1.8.

2.3 TERMINAL CONTROL CHARACTERS

The command line editor uses a history buffer to remember the last five commands issued. These commands can be recalled and then executed using control keys.

Several keys serve as a command line edit and control functions. It is best to become familiar with these functions before working with the system**.** These functions include:

a. RETURN (carriage- return) - will enter the command line and initiates processing
b. Delete (Backspace) key or CTRL-H - will delete the last character entered
c. CTRL-D - Go down in the command history buffer; you can modify, then press enter
d. CTRL-U - Go up in the command history buffer; you can modify, then press enter
e. CTRL-R - Recall and execute the last command entered; does not require pressing RETURN

For characters requiring the control key (CTRL), the CTRL should be held down while the other key is pressed.

2.4 dBUG COMMAND SET

Table 3 lists the dBUG commands. Each of the individual commands is described in the following pages.

Table 3. dBUG Commands

COMMAND MNEMONIC	DESCRIPTION	SYNTAX	PAGE
AS	Assemble	AS <addr> <instruction>	2-6
BC	Block Compare	BD <1ST ADDR> <2ND ADDR> <LENGTH>	2-7
BF	Block Fill	BF<WIDTH> BEGIN END DATA	2-8
BM	Block Move	BM BEGIN END DEST	2-9
BS	Block Search	BS <WIDTH> BEGIN END DATA	2-11
BR	Breakpoint	BR ADDR <-R> <-C COUNT> <-T TRIGGER>	2-10
DATA	Data Convert	DATA VALUE	2-12
DI	Disassemble	DI <ADDR>	2-13
DL	Download Serial	DL <OFFSET>	2-14
DN	Download Network	DN <-C> <-E> <-S> <-I> <-O OFFSET> <FILENAME>	2-15
GO	Execute	GO <ADDR>	2-16
GT	Go TILL BREAKPOINT	GT <ADDR>	2-17
HELP	Help	HELP <COMMAND>	2-18
IRD	Internal Register Display	IRD <MODULE.REGISTER>	2-19
IRM	Internal Register Modify	IRM <MODULE.REGISTER> <DATA>	2-20
MD	Memory Display	MD <WIDTH> <BEGIN> <END>	2-21
MM	Memory Modify	MM <WIDTH> ADDR <DATA>	2-22
RD	Register Display	RD <REG>	2-23
RM	Register Modify	RM REG DATA	2-24
RESET	Reset	RESET	2-25
SET	Set Configurations	SET OPTION <VALUE>	2-26
SHOW	Show Configurations	SHOW OPTION	2-28
STEP	Step (Over)	STEP	2-29
SYMBOL	Symbol Management	SYMBOL <SYMB> <-A SYMB VALUE> <-R SYMB> <-C | L | S>	2-30
TRACE	Trace(Into)	TRACE <NUM>	2-31
UPDBUG	Update Dbug	UPDBUG	2-32
UPUSER	Update User Flash	UPUSER	2-33
VERSION	Show Version	VERSION	2-34

Note:
If a command causes the system to access an unused address (i.e., no memory or peripheral devices are mapped at that address), a bus trap error will occur, which results in a trap error message and reveals the contents of all the MCF5206e core registers. Control is returned to the dBUG monitor.

Parameters enclosed in < > symbols are optional.

2.4.1 AS - Assemble AS

Usage: AS <addr> <instruction>

The AS command assembles instructions. The value for addr can be an absolute address specified as a hexadecimal value or a symbol name. The instruction can be any valid instruction for the target processor.

The assembler keeps track of the address where the last instructionŝ opcode was written. If no address is provided to the AS command and the AS command has not been used since system reset, then AS defaults to the beginning address of user space for the target board.

If no instruction is forwarded to the AS command, then AS prompts with the address where opcode will be written, and continues to assemble instructions until you terminate the AS command by inputting a period (.).

The inline assembler allows the use of case-sensitive symbols defined by equate statements and labels that are stored in the symbol table. The syntax for defining symbols and labels is as follows:

```
Symbol   equ   value
Symbol:  equ   value
Symbol   .equ  value
Symbol:  .equ  value
Label:   instruction
Label:
```

Constants and operands may be input in several different bases:

0x	followed by hexadecimal constant
$	followed by hexadecimal constant
@	followed by octal constant
%	followed by binary constant
digit	decimal constant

The assembler also supports the different syntax allowed for the indexed, displacement and immediate addressing modes:

(12,An)	or	12(An)
(4,PC,Xn)	or	4(PC,Xn)
(0x1234).L	or	0x1234.L

Examples:

To assemble one fmove'instruction at the next assemble address, the command is:

```
as      move.l  #0x25,d0
```

To assemble multiple lines at 0x12000, the command is:

```
as      12000
```

then:

```
0x00012000:  start:  nop
0x00012002:  nop
0x00012004:  lsr.l   #1,d0
0x00012006:  cmp     #4,d0
0x00012008:  beq     start
0x0001200A:
```

2.4.2 BC - Compare Blocks of Memory BC

Usage: BC first second length

The BC command compares two contiguous blocks of memory; the first block starting at address "first," the second block starting at address "second," both of length "length." If the blocks are not identical, the addresses of the first mismatch are displayed. The value for addresses "first" and "second" can be an absolute address specified as a hexadecimal value or a symbol name. The value for length can be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

Examples:

To verify that the code in the first block of user FLASH space (128K) is identical to the code in user ADRAM space, the command is,

```
bc  20000 FFE20000 20000        .
```

2.4.3 BF - Block of Memory Fill BF

Usage: BF<width> begin end data

The BF command fills a contiguous block of memory starting at address "begin," stopping at address "end," with the value, "data." Width modifies the size of the data that is written.

The value for addresses "begin" and "end" can be an absolute address specified as a hexadecimal value, or a symbol name. The value for "data" can be a symbol name or a number converted according to the user defined radix, normally hexadecimal.

This command first aligns the starting address for the data access size and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly aligned memory accesses.

Examples:

To fill a memory block starting at 0x00010000 and ending at 0x00040000 with the value 0x1234, the command is:

```
bf      10000 40000 1234
```

To fill a block of memory starting at 0x00010000 and ending at 0x0004000 with a byte value of 0xAB, the command is:

```
bf.b    10000 40000 AB
```

To zero out the BSS section of the target code (defined by the symbols bss_start and bss_end), the command is:

```
bf      bss_start bss_end 0
```

2.4.4 BM - Block Move BM

Usage: BM begin end dest

The BM command moves a contiguous block of memory starting at address "begin," stopping at address "end," to the new address, "dest." The BM command copies memory as a series of bytes and does not alter the original block.

The value for addresses "begin," "end," and "dest" can be an absolute address specified as a hexadecimal value or a symbol name. If the destination address overlaps the block defined by "begin" and "end," an error message is produced and the command exits.

Examples:

To copy a block of memory starting at 0x00040000 and ending at 0x00080000 to the location 0x00200000, the command is:

```
bm      40000 80000 200000
```

To copy the target code's data section (defined by the symbols data_start and data_end) to 0x00200000, the command is:

```
bm      data_start data_end 200000
```

2.4.5 BR - Breakpoint BR

Usage: BR addr <-r> <-c count> <-t trigger>

The BR command inserts or removes breakpoints at address addr. The value for addr can be an absolute address specified as a hexadecimal value, or a symbol name. Count and trigger are numbers converted according to the user-defined radix, normally hexadecimal. If no argument is provided to the BR command, a listing of all defined breakpoints is displayed.

The -r option to the BR command removes a breakpoint defined at address addr. If no address is specified in conjunction with the -r option, all breakpoints are removed.

Each time a breakpoint is encountered during the execution of target code, its count value is incremented by one. By default, the initial count value for a breakpoint is zero, but the -c option allows setting the initial count for the breakpoint.

Each time a breakpoint is encountered during the execution of target code, the count value is compared against the trigger value. If the count value is equal to or greater than the trigger value, a breakpoint is encountered and control returned to dBUG. By default, the initial trigger value for a breakpoint is one, but the -t option allows setting the initial trigger for the breakpoint.

If no address is specified in conjunction with the -c or -t options, then all breakpoints are initialized to the values specified by the -c or -t option.

Examples:

To set a breakpoint at the C function main(), the command is:

```
br      _main
```

When the target code is executed and the processor reaches main(), control will returned to dBUG.

To set a breakpoint at the C function bench() and set its trigger value to 3, the command is:

```
br      _bench -t 3
```

When the target code is executed, the processor must try to execute the function bench() a third time before returning control back to dBUG.

To remove all breakpoints, the command is:

```
br      -r
```

2.4.6 BS - Block Search BS

Usage: BS<width> begin end data

The BS command searches a contiguous block of memory starting at address "begin," stopping at address "end," for the value, "data." "Width" modifies the size of the data that is compared during the search.

The value for addresses "begin" and "end" can be an absolute address specified as a hexadecimal value or a symbol name. The value for "data" can be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly aligned memory accesses.

Examples:

To search for the 16-bit value 0x1234 in the memory block starting at 0x00040000 and ending at 0x00080000 the command is:

```
bs      40000 80000 1234
```

This reads the 16-bit word located at 0x00040000 and compares it against the 16-bit value 0x1234. If no match is found, the address increments to 0x00040002 and the next 16-bit value is read and compared.

To search for the 32-bit value 0xABCD in the memory block starting at 0x00040000 and ending at 0x00080000, the command is:

```
bs.l    40000 80000 ABCD
```

This reads the 32-bit word located at 0x00040000 and compares it against the 32-bit value 0x0000ABCD. If no match is found, the address increments to 0x00040004 and the next 32-bit value is read and compared.

To search the BSS section (defined by the symbols bss_start and bss_end) for the byte value 0xAA, the command is:

```
bs.b    bss_start bss_end AA
```

2.4.7 DATA - Data Conversion DATA

Usage: DATA data

The DATA command displays data in both decimal and hexadecimal notation.

The value for data can be a symbol name or an absolute value. If an absolute value passed into the DATA command is prefixed by "0x", data is interpreted as a hexadecimal value. Otherwise, data is interpreted as a decimal value. All values are treated as 32-bit quantities.

Examples:

To display the decimal equivalent of 0x1234, the command is:

```
data    0x1234
```

To display the hexadecimal equivalent of 1234, the command is:

```
data    1234
```

2.4.8 DI - Disassemble DI

Usage: DI <addr>

The DI command disassembles target code pointed to by addr. The value for addr can be an absolute address specified as a hexadecimal value or a symbol name.

Wherever possible, the disassembler will use information from the symbol table to produce a more meaningful disassembly. This is especially useful for branch target addresses and subroutine calls.

The DI command tries to track the address of the last disassembled opcode. If no address is provided to the DI command, the DI command uses the address of the last disassembled opcode.

Examples:

To disassemble code that starts at 0x00040000, the command is:

```
di      40000
```

To disassemble code of the C function main(), the command is:

```
di      _main
```

2.4.9 DL - Download Serial DL

Usage: DL <offset>

The DL command performs an S-record download of data obtained from the serial port. The value for offset is converted according to the user-defined radix, normally hexadecimal.

If offset is provided, the destination address of each S-record is adjusted by offset. The DL command checks the destination address for validity. If the destination is an address below the defined user space (0x00000000-0x00020000), an error message is displayed and downloading aborted.

If the S-record file contains the entry point address, the program counter is set to reflect this address.

Examples:

To download an S-record file through the serial port, the command is:

```
dl
```

To download an S-record file through the serial port and adjust the destination address by 0x40, the command is:

```
dl      0x40
```

2.4.10 DN - Download Network DN

Usage: DN <-c> <-e> <-i> <-s> <-o offset> <filename>

The DN command downloads code from the network. The DN command handle files that are S-record, COFF, or ELF formats. The DN command uses Trivial File Transfer Protocol, TFTP, to transfer files from a network host.

In general, the type of file to be downloaded and the name of the file must be specified to the DN command. The -c option indicates a COFF download, the -e option indicates an ELF download, -I option indicates an image download, and the -s indicates an S-record download. The -o option works only in conjunction with the -s option to indicate and optional offset for S-record download. The filename is passed directly to the TFTP server and, therefore, must be a valid filename on the server.

If neither of the -c, -e, -i, -s or filename options are specified, then a default filename and file type will be used. Default filename and file type parameters are manipulated using the SET and SHOW commands.

The DN command checks the destination address for validity. If the destination is an address below the defined user space, an error message is displays and downloading is aborted.

For ELF and COFF files, which contain symbolic debug information, the symbol tables are extracted from the file during download and used by dBUG. Only global symbols are kept in dBUG.

Note: The dBUG symbol table is not cleared prior to downloading, so it is your responsibility to clear the symbol table as necessary prior to downloading.

If an entry point address is specified in the S-record, COFF, or ELF file, the program counter is set accordingly.

Examples:

To download an S-record file with the name "srec.out", the command is:

dn -s srec.out

To download a COFF file with the name "coff.out", the command is:

dn -c coff.out

To download a file using the default file type with the name "bench.out", the command is:

dn bench.out

To download a file using the default filename and file type, the command is:

dn

This command requires proper Network address and parameter setup. Refer to Appendix A for this procedure.

2.4.11 Go - Execute GO

Usage: GO <addr>

The GO command executes target code starting at address addr. The value for addr can be an absolute address specified as a hexadecimal value or a symbol name. If no argument is provided, the GO command begins executing instructions at the current program counter.

When the GO command is executed, all user-defined breakpoints are inserted into the target code, and the context is switched to the target program. Control is regained only when the target code encounters a breakpoint, illegal instruction, or other exception that hands control back to dBUG.

Examples:

To execute code at the current program counter, the command is:

```
go
```

To execute code at the C function main(), the command is:

```
go _main
```

To execute code at the address 0x00040000, the command is:

```
go 40000
```

2.4.12 GT - Execute Till a Temporary Breakpoint GT

Usage: GT <addr>

The GT command executes the target code starting at the address in PC (whatever the PC has) until a temporary breakpoint as given in the command line is reached.

Example:

To execute code at the current program counter and stop at breakpoint address 0x10000, the command is:

GT 10000

2.4.13 HELP - Help HE

Usage: HELP <command>

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of where user code can start is given. If <command> is provided, a brief listing of the syntax of the specified command is displayed.

Examples:

To obtain a listing of all the commands available within dBUG, the command is:

```
help
```

The help list is longer than one page. The help command displays one full screen and asks for an input to display the rest of the list.

To obtain help on the breakpoint command, the command is:

```
help br
```

2.4.14 IRD - Internal Registers Display IRD

Usage: IRD <module.register>

This commands displays the internal registers of different modules inside the MCF5206e. In the command line, the module refers to the module name where the register is located and the register refers to the specific register needed.

The registers are organized according to the module to which they belong. The available modules on the MCF5206e are SIM, UART1, UART2, TIMER, DMA, M-Bus, DRAMC, and Chip Select. Refer to the MCF5206e Userś Manual for more details.

Example:

```
ird     sim.sypcr           ;display the SYPCR register in the SIM module.
```

2.4.15 IRM - Internal Registers MODIFY IRM

Usage: IRM module.register data

This commands modifies the contents of the internal registers of different modules inside the MCF5206e. In the command line, the module refers to the module name where the register is located, register refers to the specific register needed, and data is the new value to be written into that register.

The registers are organized according to the module to which they belong. The available modules on the MCF5206e are SIM, UART1, UART2, TIMER, M-Bus, DRAMC, Chip-Select. Refer to MCF5206e User's Manual for more information.

Example:

```
irm     timer.tmr1  0021        ;write 0021 into TMR1 register in the TIMER module.
```

2.4.16 MD - Memory Display MD

Usage: MD<width> <begin> <end>

The MD command displays a contiguous block of memory starting at address "begin" and stopping at address "end". The value for addresses "begin" and "end" can be an absolute address specified as a hexadecimal value or a symbol name. "Width" modifies the size of the data that is displayed.

Memory display starts at the address "begin". If no beginning address is provided, the MD command uses the last displayed address. If no ending address is provided, MD will display memory up to an address that is 128 beyond the starting address.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly aligned memory accesses.

Examples:

To display memory at address 0x00400000, the command is:

```
md 400000
```

To display memory in the data section (defined by the symbols data_start and data_end), the command is:

```
md data_start
```

To display a range of bytes from 0x00040000 to 0x00050000, the command is:

```
md.b  40000 50000
```

To display a range of 32-bit values starting at 0x00040000 and ending at 0x00050000, the command is:

```
md.l  40000 50000
```

This command can be repeated by pressing the RETURN key. It will continue with the address after the last display address.

2.4.17 MM - Memory Modify MM

Usage: MM<width> addr <data>

The MM command modifies memory at the address addr. The value for address addr can be an absolute address specified as a hexadecimal value or a symbol name. "Width" changes the size of the data that is modified. The value for data may be a symbol name or a number converted according to the user defined radix, normally hexadecimal.

If a value for data is provided, the MM command immediately sets the contents of addr to data. If no value for data is provided, the MM command enters into a loop. The loop obtains a value for data, sets the contents of the current address to data, increments the address according to the data size, and repeats. The loop terminates when an invalid entry for the data value is entered, i.e., a period.

This command first aligns the starting address for the data access size then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly aligned memory accesses.

Examples:

To set the byte at location 0x00010000 to be 0xFF, the command is:

```
mm.b  10000 FF
```

To interactively modify memory beginning at 0x00010000, the command is:

```
mm    10000
```

2.4.18 RD - Register Display RD

Usage: RD <reg>

The RD command displays the register set of the target. If no argument for reg is provided, then all registers are displayed. Otherwise, the value for reg is displayed.

Examples:

To display all the registers and their values, the command is:

```
rd
```

To display only the program counter, the command is:

```
rd      pc
```

2.4.19 RM - Register Modify RM

Usage: RM reg data

The RM command modifies the contents of the register "reg to data." The value for "reg" is the name of the register, and the value for "data" can be a symbol name or it is converted according to the user defined radix, normally hexadecimal.

The dBUG preserves the registers by storing a copy of the register set in a buffer. The RM command updates the copy of the register in the buffer. The actual value will not be written to the register until target code is executed.

Examples:

To change register D0 to contain the value 0x1234, the command is:

```
rm      D0 1234
```

2.4.20 RESET - Reset the board and dBUG RESET

Usage: RESET

The RESET command tries to reset the board and dBUG to their initial power-up states.

The RESET command executes the same sequence of code that occurs at power up. This code tries to initialize the devices on the board and dBUG data structures. If the RESET command fails to reset the board to your satisfaction, cycle power or press the reset button.

Examples:

To reset the board and clear the dBUG data structures, the command is:

```
reset
```

2.4.21 SET - Set Configuration SET

Usage: SET option <value>
SET

The SET command allows the setting of user-configurable options within dBUG. The options are listed below. If the SET command is issued without an option, it will show the available options and values.

See Appendix A for information on configuring dBUG for network downloads. **The board needs a RESET after this command in order for the new option(s) to take effect.**

baud - This is the baud rate for the first serial port on the board. All communication between dBUG and you occur using either 9600 or 19200 bps, eight data bits, no parity, and one stop bit, 8N1. Do not choose 38400 baud.

base - This is the default radix for use in converting number from their ASCII text representation to the internal quantity used by dBUG. The default is hexadecimal (base 16), and other choices are binary (base 2), octal (base 8), and decimal (base 10).

client - This is the network Internet Protocol (IP) address of the board. For network communications, the client IP must be set to a unique value, usually assigned by your local network administrator.

server - This is the network IP address of the machine that contains files accessible via TFTP. Your local network administrator will have this information and can assist in properly configuring a TFTP server if one does not exist.

gateway - This is the network IP address of the gateway for your local subnetwork. If the client IP address and server IP address are not on the same subnetwork, then this option must be properly set. Your local network administrator will have this information.

netmask - This is the network address mask to determine if use of a gateway is required. This field must be properly set. Your local network administrator will have this information.

filename - This is the default filename to be used for network download if no name is provided to the DN command.

filetype - This is the default file type to be used for network download if no type is provided to the DN command. Valid values are: "s-record," "coff," "image," and "elf."

autoboot - This option allows for the automatic download and execution of a file from the network. You can use this option to automatically boot an operating system from the network. Valid values are: "on" and "off." This option is not implemented on the current of dBUG.

nicbase - This is base address of the network interface. This command is **used to inform the dBUG** of the address of the network interface. The default value shows 0x0000. However, this parameter is hard coded to 0x300. **DO NOT CHANGE THIS OPTION.**

macaddr - This is the Ethernet MAC address of the board. For network communications, the MAC address must be set to a unique value. Any address that is not already in use is suitable.

Examples:

To see all the available options and supported choices, the command is:

```
set
```

To set the baud rate of the board to be 19200, the command is:

```
set       baud 19200
```

Now press the RESET button (RED) or RESET command for the new baud to take effect. This baud will be programmed in Flash ROM and will be used during the power up.

2.4.22 SHOW - Show Configuration SHOW

Usage: SHOW option
SHOW

The SHOW command displays the settings of the user-configurable options within dBUG. The SHOW command can display most configurable options via the SET commands. If the SHOW command is issued without any option, it will show all options.

Examples:

To display all the current options, the command is:

```
show
```

To display the current baud rate of the board, the command is:

```
show        baud
```

To display the TFTP server IP address, the command is:

```
show        server
```

2.4.23 STEP - Step Over ST

Usage: STEP

You can use the ST command to "step over" a subroutine call rather than trace every instruction in the subroutine. The ST command sets a breakpoint one instruction beyond the current program counter and then executes the target code.

You can also use the ST command for BSR and JSR instructions. The ST command will work for other instructions as well, but note that if the ST command is used with an instruction that will not return, i.e. BRA, the temporary breakpoint may never be encountered and thus dBUG may not regain control.

Example:

To pass over a subroutine call, the command is:

```
step
```

2.4.24 SYMBOL - Symbol Name Management SYMBOL

Usage: SYMBOL <symb> <-a symb value> <-r symb> <-c|l|s>

The SYMBOL command adds or removes symbol names from the symbol table. If only a symbol name is provided to the SYMBOL command, the symbol table is searched for a match on the symbol name and its information displayed.

The **-a** option adds a symbol name and its value into the symbol table. The **-r** option removes a symbol name from the table.

The **-c** option clears the entire symbol table; the **-l** option lists the contents of the symbol table; and the **-s** option displays usage information for the symbol table.

Symbol names contained in the symbol table are truncated to 31 characters. Any symbol table lookups, either by the SYMBOL command or by the disassembler, will use only the first 31 characters. Symbol names are case sensitive.

Examples:

To define the symbol "main" to have the value 0x00040000, the command is:

```
symbol      -a main 40000
```

To remove the symbol "junk" from the table, the command is:

```
symbol      -r junk
```

To see how full the symbol table is, the command is:

```
symbol      -s
```

To display the symbol table, the command is:

```
symbol      -l
```

2.4.25 TRACE - Trace Into TR

Usage: TRACE <num>

The TRACE command allows single instruction execution. If <num> is provided, then <num> instructions are executed before control is returned to dBUG. The value for <num> is a decimal number.

The TRACE command sets bits in the processors'supervisor registers to achieve single instruction execution, and the target code executed. Control returns to dBUG after a single instruction execution of the target code.

Examples:

To trace one instruction at the program counter, the command is:

```
tr
```

To trace 20 instructions from the program counter, the command is:

```
tr      20
```

2.4.26 UPDBUG - Update the dBUG Image UPDBUG

Usage: UPDBUG

The UPDBUG command update the dBUG image in Flash. When updates to the MCF5206e EVS dBUG are available, the updated image is downloaded to address 0x00020000. The new image is placed into Flash using the UPDBUG command. You are prompted for verification before performing the operation. Use this command with extreme caution as any error can render dBUG
--and thus the board--useless.

2.4.27 UPUSER - Update User Code In Flash UPUSER

Usage: UPUSER <number of sectors>

The UPUSER command places user code and data into space allocated for you in Flash. Six sectors of 128K each are available as user space. To place code and data in user Flash, the image is downloaded to address 0x00020000 and the UPUSER command issued. This command programs all six sectors of user Flash space. You access this space starting at address 0xFFE20000. To program less than six sectors, supply the number of sectors you want to program after the UPUSER command.

Examples:

To program all 6 sectors of user FLASH space, the command is:

upuser or upuser 6

To program only 128K of user FLASH space, the command is:

upuser 1

2.4.28 VERSION - Display dBUG Version VERSION

Usage: VERSION

The VERSION command displays the version information for dBUG. The dBUG version number and build date are both given.

The version number is separated by a decimal, for example, "1.1.1." The first number indicates the version of the CPU specific code, the second indicates the version of the board specific code, and the third indicates the version of the board-specific code.

The version date is the day and time at which the entire dBUG monitor was compiled and built.

Examples:

To display the version of the dBUG monitor, the command is:

```
version
```

2.5 TRAP #15 Functions

An additional utility within the dBUG firmware is a function called the TRAP 15 handler. The user program to use various routines within the dBUG, to perform a special task, and to return control to the dBUG can call this function. This section describes the TRAP 15 handler and how it is used.

There are four TRAP #15 functions. These are: OUT_CHAR, IN_CHAR, CHAR_PRESENT, and EXIT_TO_dBUG.

2.5.1 OUT_CHAR

This function (function code 0x0013) sends a character, which is in lower 8 bits of D1, to terminal.

Assembly example:

```
       /* assume d1 contains the character */
        move.l            #$0013,d0    Selects the function
       TRAP         #15          The character in d1 is sent to terminal
```

C example:

```
void board_out_char (int ch)
{
       /* If your C compiler produces a LINK/UNLK pair for this routine,
        * then use the following code which takes this into account
       */
#if  l
       /* LINK a6,#0 -- produced by C compiler */
       asm (" move.l       8(a6),d1");  /* put 'ch'into d1 */
       asm (" move.l       #0x0013,d0");  /* select the function */
                              /* make the call */
       /* UNLK a6  -- produced by C compiler */
#else
       /*  If C compiler does not produce a LINK/UNLK pair, the use
        *  the following code.
       */
        asm (" move.l      4(sp),d1");  /* put 'ch'into d1 */
       asm (" move.l       #0x0013,d0");  /* select the function */
                              /* make the call */
#endif
}
```

2.5.2 IN_CHAR

This function (function code 0x0010) returns an input character (from terminal) to the caller. The returned character is in D1.

Assembly example:

```
       move.l #$0010,d0    Select the function
       trap   #15          Make the call, the input character is in d1.
```

C example:

```
int board_in_char (void)
{
       asm (" move.l       #0x0010,d0");             /* select the function */
       asm (" trap  #15");              /* make the call */
       asm (" move.l       d1,d0");           /* put the character in d0 */
```

```
}
```

2.5.3 CHAR_PRESENT

This function (function code 0x0014) checks if an input character is present to receive. A value of zero is returned in D0 when no character is present. A non-zero value in D0 means a character is present.

Assembly example:

```
      move.l #$0014,d0    Select the function
      trap   #15          Make the call,  d0 contains the response (yes/no).
```

C example:

```
int board_char_present (void)
{
      asm (" move.l       #0x0014,d0");             /* select the function */
      asm (" trap  #15");             /* make the call */
}
```

2.5.4 EXIT_TO_dBUG

This function (function code 0x0000) transfers the control back to the dBUG, by terminating the user code. The register context are preserved.

Assembly example:

```
      move.l #$0000,d0    Select the function
      trap   #15          Make the call,  exit to dBUG.
```

C example:

```
void board_exit_to_dbug (void)
{
      asm (" move.l       #0x0000,d0");       /* select the function */
      asm (" trap       #15");          /* exit and transfer to dBUG */
```

3 HARDWARE DESCRIPTION AND RECONFIGURATION

This chapter provides a functional description of the M5206EC3 board hardware (the schematics are located in Appendix C). In this manual, an active low signal is indicated by a dash ("-") preceding the signal name.

3.1 PROCESSOR AND SUPPORT LOGIC

This part of the chapter discusses the ColdFire processor and general supporting logic on the M5206EC3 board.

3.1.1 Processor

Refer to Section 1.2 for a detailed description of the MCF5206e processor.

The MCF5206e has an IEEE JTAG-compatible port and BDM port. These signals are available at port J5. The processor also has the logic to generate as many as eight chip selects (-CS0 to -CS7) and supports ADRAM. The processor defaults to the BDM debug mode.

3.1.2 Reset Logic

The reset logic provides system initialization under two modes (under system power-up and when the RESET switch, S1 [red switch], is active). The power-on generates the master RESET by asserting the -RSTI and -HIZ that reset the total system. The RESET switch generates normal reset that resets the entire processor except for the DRAM controller.

U11 and U12 produce active-low power-on RESET signals that feed the LSI2032V (U10) along with the push-button RESET. The U4 device generates the system reset (-RESET), FLASH and Ethernet RESET signals.

DBUG configures internal resources during the initialization by invalidating and disabling the instruction cache. The Vector Base Register, VBR, points to the Flash. However, a copy of the exception table is made at address $00000000 in DRAM. To take over an exception vector, you must place the address of the exception handler in the appropriate vector in the vector table located at $00000000, and then point the VBR to $00000000.

The software watchdog timer is disabled and internal timers are placed in a stop condition. Interrupt controller registers are initialized with unique interrupt level/priority pairs. The parallel I/O port is configured for input.

3.1.3 -HIZ Signal

The -HIZ signal is actively driven by the LSI2032V (U10). This signal is available for monitoring on LA1 and J4. However, you should not drive this signal. If you need to drive -HIZ, it should be done through the -HIZ_INLOW signal that is available on J2 pin 26. The -HIZ_INLOW signal feeds the U10 that drives the -HIZ signal to the processor.

3.1.4 Clock Circuitry

The M5206EC3 uses a 54 MHz oscillator (U4) to provide the clock to the CLK pin of the processor. In addition to U4, a 20-MHz oscillator (U18) feeds into the Ethernet chip. U4 also drives the LSI2032 internal clock requirements. The 20-MHz oscillator is not used by the internal logic of the LSI2032 although it is connected in the schematic.

3.1.5 Watchdog Timer (Bus Monitor)

The processor initiates a bus cycle that provides the necessary information for the bus cycle (e.g. address, data, control signals, etc.) and asserting the -CS or -RAS low. Then, the processor waits for an acknowledgment (-TA signal) from the addressed device before it can complete the bus cycle. It is possible (due to incorrect programming) that the processor can try to access part of the address space that physically does not exist. In this case, the bus cycle will continue forever because there is no memory or I/O device to provide an acknowledgment signal, leaving the processor in an infinite wait state. The MCF5206e has the necessary logic built into the chip to watch the duration of the bus cycle. If the cycle is not terminated within the preprogrammed duration, the logic will internally assert a transfer-error signal. In response, the processor will terminate the bus cycle and an access fault exception (trap) will occur.

The duration of the watchdog is selected by BMT0-1 bits in System Protection Register. The dBUG initializes this register with the value 00, which provides for 1024 system clock time-out.

3.1.6 Interrupt Sources

The ColdFire Family of processors can receive interrupts for seven levels of interrupt priorities. When the processor receives an interrupt that has higher priority than the current interrupt mask (in status register), it will perform an interrupt-acknowledge cycle at the end of the current instruction cycle. This interrupt-acknowledge cycle indicates to the source of the interrupt that the request is being acknowledged and the device should provide the proper vector number to indicate where the service routine for this interrupt level is located. If the source of the interrupt can't provide a vector, its interrupt should be set up as autovector interrupt, which directs the processor to a predefined entry into the exception table (refer to *the MCF5206e User's Manual*).

The processor goes to a service routine via the exception table. This table is in the Flash and the VBR points to it. A copy of this table is made in the RAM starting at $00000000. To set an exception vector, place the address of the exception handler in the appropriate vector in the vector table located at $00000000, and then point the VBR to $00000000.

The MCF5206e has three external interrupt request lines. You can program the external interrupt request pins to a interrupt priority-level signals (-IPL[2:0]) or predefined interrupt request pins (-IRQ7, -IRQ4, -IRQ1). The M5206EC3 configures these lines as predefined interrupt request pins. The only interrupt signal used on the M5206EC3 is -IRQ4 and -IRQ7 for the Ethernet. By changing the external interrupt request pins to a interrupt priority-level signal, the Ethernet will no longer function. There are also six internal interrupt requests from Timer1, Timer2, software watchdog timer, UART1, UART2, and MBUS. You can program each external and internal for any priority level. In case of similar priority level, a second relative priority between 0 to 3 will be assigned.

The software watchdog is programmed for Level 7, priority 2, and uninitialized vector. The UART1 is programmed for Level 3, priority 2, and autovector. The UART2 is programmed for Level 3, priority 1, and autovector. The M-Bus is at Level 3, priority 0, and autovector. The timers are at Level 5 with Timer 1 with priority 3 and Timer 2 with priority 2 and both for autovector.

The M5206EC3 uses -IRQ7 to support the ABORT function using the ABORT switch S1 (red switch). This switch forces a nonmaskable interrupt (level 7, priority 3) if your program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT). Because the ABORT switch can't generate a vector in response to level 7 interrupt acknowledge from the processor, the debugger programs this request for autovector mode. Additional circuitry prevents debouncing of the switch, which causes the ABORT signal to assert for a minimum of 30µ seconds.

The -IRQ1 line of the MCF5206e is not used on this board. However, the -IRQ1 is programmed for Level 1 with priority 1 and autovector. You can use this line for external interrupt request. Refer to *MCF5206e User's Manual* for more information about the interrupt controller.

3.1.7 Internal SRAM

The MCF5206e has 8 KBytes of internal memory. This memory, which is mapped to 0x00400000, is not used by the dBUG but is available to users.

3.1.8 MCF5206e Registers and Memory Map

The memory and I/O resources of the M5206EC3 are divided into three groups, MCF5206e internal, external resources, and the Ethernet controller. All the I/O registers are memory mapped.

The MCF5206e has built-in logic and as many as eight chip-select pins (-CS0 to -CS7) that enable external memory and I/O devices. In addition, two -RAS lines are available for DRAMs and registers to specify the address range, type of access, and the method of -TA generation for each chip-select and -RAS pins. The dBUG programs these registers to map the external memory and I/O devices.

The M5206EC3 uses chip-select zero (-CS0) to enable the Flash ROMs (refer to Section 3.3). The M5206EC3 uses -RAS1, -RAS2, -CAS0, -CAS1, -CAS2, and -CAS3 to enable the ADRAM SIMM module (refer to Section 3.2), -CS2 for SRAM (not populated), and -CS3 for Ethernet bus I/O space.

The chip-select mechanism of the MCF5206e allows the memory mapping to be defined based on the memory space required (user/supervisor, program/data spaces).

The MBAR register maps all MCF5206e internal registers, configuration registers, parallel I/O port registers, UART registers, and system control registers at any 1 KByte boundary. The dBUG maps MBAR to 0x10000000. For a complete map of these registers, refer to the *MCF5206e User's Manual*.

The M5206EC3 board can have as much as 32 MBytes of 3.3V ADRAM installed. Refer to Section 3.2 for a discussion of RAM. The dBUG is programmed in two 29LV004B Flash ROMs that only occupy 1 MBytes of the address space. ROM monitor uses the first 128 KBytes. The following six 128 KByte sectors are available to users. Refer to section 3.3.

The Ethernet bus interface maps all the I/O space of the Ethernet bus to the MCF5206e memory at address $40000000. Refer to section 3.6.

Table 4. ROM Monitor Default M5206EC3 Memory Map

ADDRESS RANGE	SIGNAL and DEVICE
$00000000-$003FFFFF	-RAS1, -RAS2, 4 MBytes of ADRAMs
$00400000-$00401FFF	Internal SRAM (8 KBytes)
$10000000-$100003FF	Internal Module registers
$30000000-$3007FFFF[1]	-CS2, External SRAM (512 KBytes)
$40000000-$4000FFFFF	-CS3, 1M Byte Ethernet Bus area
$FFE00000-$FFEFFFFF	-CS0, 1 MBytes of Flash ROM

1. Not installed. Level 2 cache footprint accepts Motorola's MCM69F737TQ chip and any other SRAM with the same electrical specifications and package.

All the unused area of the memory map is available to users.

3.1.9 Reset Vector Mapping

After reset, the processor attempts to get the initial stack pointer and initial program counter values from locations $000000-$000007 (the first eight bytes of memory space). This operation requires the board to have a nonvolatile memory device in this range with proper information. However, in some systems, it is preferable to have RAM starting at address $00000000. In the MCF5206e, the -CS0 responds to any accesses after reset until the CSMR0 is written. Because -CS0 is connected to the Flash ROMs, they appear to be at address $00000000, which provides the initial stack pointer and program counter (the first eight bytes of the Flash ROM). The initialization routine, however, programs the chip-select logic and locates the Flash ROMs to start at $FFE00000 and the DRAMs to start at $00000000.

3.1.10 -TA Generation

The processor starts a bus cycle by providing the necessary information (address, R/-W, etc.) and asserting the -TS. The processor then waits for an acknowledgment (-TA) by the addressed device before it can complete the bus cycle. This -TA is used not only to indicate the presence of a device, it also allows devices with different access time to properly communicate with the processor . The MCF5206e, as part of the chip-select logic, has a built-in mechanism to generate the -TA for all external devices that do not have the capability to generate the -TA on their own. The Flash ROMs and DRAMs can not generate the -TA. Their chip-select logic is programmed by the ROM monitor to generate the -TA internally after a preprogrammed number of wait states. In order to support the future expansion of the board, the -TA input of the processor is also connected to the Processor Expansion Bus, J3, which allows the expansion boards to assert this line to indicate its
-TA to the processor. On the expansion boards, however, this signal should be generated through an open collector buffer with no pullup resistor (a pullup resistor is included on the board). All the -TAs from the expansion boards should be connected to this line.

3.1.11 Wait State Generator

The Flash ROMs and ADRAM SIMM on the board may require some adjustments on the cycle time of the processor to make them compatible with processor speed. To extend the CPU bus cycles for the slower devices, you can program the chip-select logic of the MCF5206e to generate the -TA after a given number of wait states. Refer to Sections 3.2 and 3.3 for information about wait state requirements of ADRAMs and Flash ROMs, respectively.

3.2 ADRAM SIMM

The M5206EC3 has one 168-pin SIMM socket (U23) for ADRAM SIMM at 3.3 V. This socket supports DRAM SIMMs of 250Kx32, 1Mx32, 2Mx32, 4Mx32, and 8Mx32. No special configurations are needed. The dBUG will detect the total memory installed on power-up. The SIMM speed should be 60ns. (Note: NOT SV ADRAM SIMM).

3.3 FLASH ROM

There are two 512 KByte Flash ROMs on the M5206EC3: U13 (high, even byte), and U15 (low, odd byte).

The board is shipped with two 29LV004, 512 KByte FLASH ROMs for a total of 1 MBytes. The first 128 KBytes and last 128 KBytes are reserved by the ROM monitor firmware. 768 KBytes are available to users. The chip-select signal generated by the MCF5206e (-CS0) enables both chips.

You can program the MCF5206e chip-select logic to generate the -TA for -CS0 signal after a certain number of wait states. The dBUG programs this parameter to three wait states.

3.3.1 JP2 Jumper and User's Program

When the jumper is set between pins 1 and 2, the behavior is normal. When the jumper is set between pins 2 and 3, the board boots from the second half of Flash virtual address of (0x80000) physical address of (0xFFE80000).

Procedure:

1. Compile and link as though the code was to be placed at the base of the Flash, but set up so that it will download to the ADRAM starting at address 0x80000. You need to refer to the compiler for this, as the procedure will depend on the compiler used (in Diab Data, a shadow in the linker file is used).

2. Set up the jumper for Normal operation, pin1 connected to pin 2.

3. Download to ADRAM (if using serial or Ethernet, start ROM monitor first. If using BDM via wiggler, download first, then start ROM monitor by pointing PC to 0xffe00400 and run.)

4. In ROM Monitor, run "puser" command.

5. Move the jumper to pins 2 and 3 and press "reset" to execute the code in user space.

3.4 SERIAL COMMUNICATION CHANNELS

The M5206EC3 offers a number of serial communications that are discussed in this section.

3.4.1 MCF5206e Two UARTs

The MCF5206e has two built-in UARTs, each with its own software-programmable baud-rate generators. Only one channel serves as the ROM monitor-to-terminal output and the other is available to users. The ROM monitor, however, programs the interrupt level for UART1 to Level 3, priority 2, and autovector mode of operation. The interrupt level for UART2 is programmed

to Level 3, priority 1, and autovector mode of operation. The signals of these channels are available on port LA1 and J2. The signals of UART1 and UART2 are also passed through the RS-232 driver/receiver and are available on DB-9 connectors P1 and P2. Refer to *the MCF5206e User's Manual* for programming and the register map.

3.4.2 Motorola Bus (M-Bus) Module

The MCF5206e has a built-in M-Bus module that allows interchip bus interface for a number of I/O devices. It is compatible with industry-standard I^2C Bus. The M5206EC3 does not use this module and it is available to users. The two M-Bus signals are SDA and SCL that are available at the LA1 and J2 connector. These signals are open-collector signals. However, they have pullup resistors on the M5206EC3. These signals are connected to the ADRAM SIMM module I^2C interface but not used by the debugger. The interrupt control register for M-Bus is set for Level 3, priority 0, and autovector.

3.5 PARALLEL I/O PORT

The MCF5206e has one 8-bit parallel port. All the pins have dual functions and can be configured as I/O or their alternate function via the Pin Assignment Register. All pins are configured as I/O pins by the ROM monitor.

3.6 ONBOARD ETHERNET LOGIC

The M5206EC3 includes the necessary logic, drivers, and the NE2000-compatible Ethernet chip to allow 10 Mbit transfer rate on a network. The Ethernet space addresses are located starting at 0x40000000. The interface base address is 0x300 and uses the ColdFire IRQ4. Thus, the address of the chip is 0x40000300.

Note that all registers should be addressed as WORD (even though the registers are bytes) and that the even address registers are addressed as they are (no change); the read word will have the byte of the data in the lower byte of the word.

For odd-addressed bytes, the address is mapped to 0x400083xx-1. Note that odd bytes are addressed as even addresses but increased by 0x8000. Still the read byte will be in the lower byte of the read word.

Below is an example of the data structure used to define the registers. For the description of the registers, refer to the Data Sheet for Davicom DM9008, a copy of this document ion is available on the ColdFire web site at http://www.mot.com/ColdFire.

```
typedef struct
{
      NATURAL16    CR;
      union
      {
            struct
            {
                  /* Even registers */
                  NATURAL16    CLDA1; /* CLDA1 (rd) PSTOP (wr) */
                  NATURAL16    TSR;         /* TSR (rd) TPSR (wr) */
                  NATURAL16    FIFO;        /* FIFO (rd) TBCR1 (wr) */
                  NATURAL16    CRDA0; /* CRDA0 (rd) RSAR0 (wr) */
                  NATURAL16    RBCR0; /* Remote Byte Count 0 (wr) */
                  NATURAL16    RSR;         /* RSR (rd) RCR (wr) */
                  NATURAL16    CNTR1; /* CNTR1 (rd) DCR (wr) */
```

```
                NATURAL16    DATAPORT;

                NATURAL16    reserved[(0x10000-0x0012)/2];

                /* Odd registers */
                NATURAL16    CLDA0; /* CLDA0 (rd) PSTART (wr) */
                NATURAL16    BNRY;        /* Boundary pointer (rd wr) */
                NATURAL16    NCR;         /* NCR (rd) TBCR0 (wr) */
                NATURAL16    ISR;         /* Interrupt Status Register (rd wr) */
                NATURAL16    CRDA1; /* CRDA1 (rd) RSAR1 (wr) */
                NATURAL16    RBCR1; /* Remote Byte Count 1 (wr) */
                NATURAL16    CNTR0; /* CNTR0 (rd) TCR (wr) */
                NATURAL16    CNTR2; /* CNTR2 (rd) IMR (wr) */
          } page0;
          struct
          {
                /* Even registers */
                NATURAL16    PAR1;        /* Physical Address Byte 1 */
                NATURAL16    PAR3;        /* Physical Address Byte 3 */
                NATURAL16    PAR5;        /* Physical Address Byte 5 */
                NATURAL16    MAR0;        /* Multicast Address Byte 0 */
                NATURAL16    MAR2;        /* Multicast Address Byte 2 */
                NATURAL16    MAR4;        /* Multicast Address Byte 4 */
                NATURAL16    MAR6;        /* Multicast Address Byte 6 */

                NATURAL16    reserved[(0x10000-0x0010)/2];

                /* Odd registers */
                NATURAL16    PAR0;        /* Physical Address Byte 0 */
                NATURAL16    PAR2;        /* Physical Address Byte 2 */
                NATURAL16    PAR4;        /* Physical Address Byte 4 */
                NATURAL16    CURR;        /* Current Page Register (rd wr) */
                NATURAL16    MAR1;        /* Multicast Address Byte 1 */
                NATURAL16    MAR3;        /* Multicast Address Byte 3 */
                NATURAL16    MAR5;        /* Multicast Address Byte 5 */
                NATURAL16    MAR7;        /* Multicast Address Byte 7 */
          } page1;
          struct
          {
                /* Even registers */
                NATURAL16    PSTOP; /* PSTOP (rd) CLDA1 (wr) */
                NATURAL16    TPSR;        /* Transmit Page Start Address (rd) */
                NATURAL16    ACU;         /* Address Counter Upper */
                NATURAL16    reserved0;
                NATURAL16    reserved2;
                NATURAL16    RCR;         /* Receive Configuration Register (rd)
*/
                NATURAL16    DCR;         /* Data Configuration Register (rd) */

                NATURAL16    reserved[(0x10000-0x0010)/2];

                /* Odd registers */
                NATURAL16    PSTART;      /* PSTART (rd) CLDA0 (wr) */
                NATURAL16    RNPP;        /* Remote Next Packet Pointer */
                NATURAL16    LNPP;        /* Local Next Packet Pointer */
                NATURAL16    ACL;         /* Address Counter Lower */
                NATURAL16    reserved1;
                NATURAL16    reserved3;
                NATURAL16    TCR;         /* Transmit Configuration Register (rd)
*/
                NATURAL16    IMR;         /* Interrupt Mask Register (rd) */
          } page2;
     } regs;
} NS8390;
```

The main purpose for this setup is to allow the use of the Ethernet card (NE2000-compatible) to facilitate network download. Refer to Chapter 2 for the network download command (DN). The dBUG driver is 100 percent NE2000-compatible.

The Ethernet Bus interrupt request line is hardwired to the ColdFire IRQ4.

The onboard ROM monitor lets you download files from a network to memory in different formats. The current formats supported are S-Record, COFF, ELF, or Image.

3.7 CONNECTORS AND THE EXPANSION BUS

There are 10 connectors on the M5206EC3 that connect the board to external I/O devices and or expansion boards. This section provides a brief discussion and the pin assignments of the connectors.

3.7.1 The Terminal Connector P1

The signals on UART1 that run through the RS-232 driver/receivers drive the terminal. The M5206EC3 uses a 9-pin D-sub female connector P1 for connecting the board to a terminal or a PC with terminal emulation software. The available signals are a working subset of the RS-232C standard. Table 5, the P1 (Terminal) Connector pin assignment, shows the pin assignment.

Table 5. P1 (Terminal) Connector Pin Assignment

PIN NO.	DIRECTION	SIGNAL NAME
1	Output	Data Carrier Detect (shorted to 4 & 6)
2	Output	Receive data
3	Input	Transmit data
4	Input	Data Terminal Ready (shorted to 1 & 6)
5		Signal Ground
6	Output	Data Set Ready (shorted to 1 & 4)
7	Input	Request to Send
8	Output	Clear to Send
9		Not Used

3.7.2 The Auxiliary Serial Communication Connector P2

The MCF5206e has two built-in UARTs. One channel on port P2, which is not used by the M5206EC3 ROM monitor, is available to users. The available signals form a working subset of the RS-232C standard. Table 6, the P2 Connector Pin Assignment, shows the pin assignment for P2.

Table 6. P2 Connector Pin Assignment

PIN NO.	DIRECTION	SIGNAL NAME
1	Output	Data Carrier Detect (shorted to 4 & 6)
2	Output	Receive data
3	Input	Transmit data
4	Input	Data Terminal Ready (shorted to 1 & 6)
5		Signal Ground
6	Output	Data Set Ready (shorted to 1 & 4)
7	Input	Request to Send
8	Output	Clear to Send
9		Not Used

3.7.3 Logical Analyzer Connectors LA1-4 and Processor Expansion Bus J2, J3, and J4

All the processor signals are available on four mictor connectors (LA1-4). You can refer to the data sheets for the major parts and the schematic at the end of this manual to get an accurate

loading capability. Subsets of the signals are available on J2, J3, and J4 for easier access. Tables 7 through 13 show the pin assignment for J2, J3, J4, LA1, LA2, LA3, and LA4, respectively.

Table 7. J2 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	-CS0	2	-CTS1
3	-CS1	4	TXD1
5	-CS2	6	RXD1
7	-CS3	8	-RTS1
9	-IRQ4	10	-CTS2
11	-BR	12	-RTS2
13	-BD	14	RXD2
15	-BG	16	TXD2
17	SDA	18	TIN0
19	SCL	20	TIN1
21	-IRQ7	22	TOUT0
23	+5V	24	TOUT1
25	GND	26	HIZ_INLOW
27	+5V	28	+5V

Table 8. J3 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	A0	2	D16
3	A1	4	D17
5	A2	6	D18
7	A3	8	D19
9	A4	10	D20
11	A5	12	D21
13	A6	14	D22
15	A7	16	D23
17	A8	18	D24
19	A9	20	D25
21	A10	22	D26
23	A11	24	D27
25	A13	26	D28
27	A13	28	D29
29	A14	30	D30
31	A15	32	D31
33	A16	34	TT0
35	A17	36	TT1
37	A18	38	ATM
39	A19	40	SIZ0
41	A20	42	SIZ1
43	A21	44	R/-W
45	A22	46	-TS
47	A23	48	-TA
49	A24	50	-TEA
51	A25	52	-ATA
53	A26	54	-RSTI
55	A27	56	-IRQ1
57	54 MHz CLK	58	+5V
59	GND	60	GND

Table 9. J4 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	TCLK	2	D0
3	DSCLK	4	D1
5	DSDI	6	D2
7	DSDO	8	D3
9	BKPT	10	D4
11	+5V	12	D5
13	GND	14	D6
15	PP0	16	D7
17	PP1	18	D8
19	PP2	20	D9
21	PP3	22	D10
23	PP4	24	D11
25	PP5	26	D12
27	PP6	28	D13
29	PP7	30	D14
31	-HIZ	32	D15
33	MTMOD	34	-CAS0
35	-RAS0	36	-CAS1
37	-RAS1	38	-CAS2
39	-DRAMW	40	-CAS3

Table 10. LA1 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	NC	4	PP4
5	PP5	6	PP6
7	PP7	8	SCL
9	SDA	10	TOUT0
11	TOUT1	12	TIN1
13	TIN0	14	TXD1
15	TXD1	16	-RTS1
17	-CTS1	18	TXD2
19	RXD2	20	NC
21	NC	22	NC
23	NC	24	NC
25	NC	26	NC
27	-TRST/DSCLK	28	TCLK
29	TDO/DSO	30	-HIZ
31	TDI/DSI	32	TMS/-BKPT
33	MTMOD	34	-CTS2
35	-RTS2	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 11. LA2 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	54 MHz CLK	4	A27
5	A26	6	A25
7	A24	8	A23
9	A22	10	A21
11	A20	12	A19
13	A18	14	A17
15	A16	16	A15
17	A14	18	A13
19	A12	20	PP3
21	PP2	22	PP1
23	PP0	24	A0
25	A1	26	A2
27	A3	28	A4
29	A5	30	A6
31	A7	32	A8
33	A9	34	A10
35	A11	36	20 MHz CLK
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 12. LA3 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	NC	4	-DRAW
5	-CAS3	6	-CAS2
7	-CAS1	8	-CAS0
9	-RAS1	10	-RAS0
11	-BG	12	-BD
13	-BR	14	-IRQ1
15	-IRQ4	16	-IRQ7
17	-ATA	18	-RSTI
19	-TS	20	NC
21	NC	22	NC
23	NC	24	-CS0
25	-CS1	26	-CS2
27	-CS3	28	TT0
29	TT1	30	ATM
31	SIZ0	32	SIZ1
33	R/-W	34	-TA
35	-TEA	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 13. LA4 Connector Pin Assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	NC	4	D0
5	D1	6	D2
7	D3	8	D4
9	D5	10	D6
11	D7	12	D8
13	D9	14	D10
15	D11	16	D12
17	D13	18	D14
19	D15	20	D31
21	D30	22	D29
23	D28	24	D27
25	D26	26	D25
27	D24	28	D23
29	D22	30	D21
31	D20	32	D19
33	D18	34	D17
35	D16	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

3.7.4 Debug Connector J5

The MCF5206e has a background debug port, real-time trace support, and real-time debug support available at connector J5. Table 14 (The J5 Connector Pin Assignment) shows the pin assignment.

Table 14. J5 Connector Pin Assignment

PIN NO.	SIGNAL NAME
1	No Connect
2	-BKPT
3	Ground
4	DSCLK
5	Ground
6	No Connect
7	-RESET
8	DSI
9	+3.3 Volts
10	DSO
11	Ground
12	PST3
13	PST2
14	PST1
15	PST0
16	DDAT3
17	DDAT2
18	DDAT1
19	DDAT0
20	Ground
21	10K pull down
22	No Connect
23	Ground
24	54 MHz CLK
25	3.3 Volts
26	-TA

Appendix A CONFIGURING dBUG FOR NETWORK DOWNLOADS

The dBUG module can perform downloads over an Ethernet network using the Trivial File Transfer Protocol (TFTP). Before using this feature, several parameters are required for network downloads to occur. This information and the steps for configuring dBUG are described below.

A.1 Required Network Parameters

For performing network downloads, dBUG requires six parameters: four that are network-related; two that are download-related. These parameters are listed below with the dBUG designation following in parenthesis.

All computers connected to an Ethernet network running the IP protocol need three network-specific parameters. These parameters are:

- Internet Protocol, IP, address for the computer (client IP)
- IP address of the Gateway for non-local traffic (gateway IP)
- Network netmask for flagging traffic as local or non-local (netmask)

In addition, the dBUG network download command requires the following three parameters:

- IP address of the TFTP server (server IP)
- Name of the file to download (filename)
- Type of the file to download (file type of S-record, COFF, ELF, or Image)

Your local system administrator can assign a unique IP address for the board and also provide you with the IP addresses of the gateway, netmask, and TFTP server. Fill out the lines below with this information.

Client IP:	___.___.___.___	(IP address of the board)
Server IP:	___.___.___.___	(IP address of the TFTP server)
Gateway:	___.___.___.___	(IP address of the gateway)
Netmask:	___.___.___.___	(Network netmask)

A.2 Configuring dBUG Network Parameters

Once the network parameters have been obtained, you must configure the ROM monitor. The following commands are used to configure the network parameters.

set client <client IP>
set server <server IP>
set gateway <gateway IP>
set netmask <netmask>
set Macaddr <macaddr>

For example, the TFTP server is named "santafe" and has IP address 123.45.67.1. The board is assigned the IP address of 123.45.68.15. The gateway IP address is 123.45.68.250, and the netmask is 255.255.255.0. The commands to dBUG are as follows:

set client 123.45.68.15
set server 123.45.67.1
set gateway 123.45.68.250
set netmask 255.255.255.0
set Macaddr 00:00:00:00:00:00

The last step is to inform dBUG of the name and type of the file to download. Prior to giving the name of the file, keep in mind the following:

1. Most, if not all, TFTP servers will permit access only to files starting at a particular sub-directory. (This is a security feature that prevents reading of arbitrary files by unknown persons.) For example, SunOS uses the directory /tftp_boot as the default TFTP directory. When specifying a filename to a SunOS TFTP server, all filenames are relative to /tftp_boot. As a result, you normally will be required to copy the file to download into the directory the TFTP server uses.

2. A default filename for network downloads is maintained by dBUG. To change the default filename, use the command:

set filename <filename>

3. When using the Ethernet network for download, S-record, COFF, ELF, or Image files can be downloaded. A default file type for network downloads is maintained by dBUG as well. To change the default file type, use the command:

set filetype <srecord|coff|elf|image>

4. Continuing with the above example, the compiler produces an executable COFF file, "a.out." This file is copied to the /tftp_boot directory on the server with the command:

rcp a.out santafe:/tftp_boot/a.out

5. Change the default filename and file type with the commands:

set filename a.out
set filetype coff

6. Finally, perform the network download with the "dn" command. The network download process uses the configured IP addresses and the default filename and file type for initiating a TFTP download from the TFTP server.

A.3 Troubleshooting Network Problems

Most problems related to network downloads are a direct result of improper configuration. Verify that all IP addresses configured into dBUG are correct (use the "show" command).

Using an IP address already assigned to another machine will cause the dBUG network download to fail, and probably create other severe network problems. Make certain the client IP address is unique for the board.

Check for proper insertion or connection of the network cable. Is the status LED lit indicating that network traffic is present?

Check for proper configuration and operation of the TFTP server. Most Unix workstations can execute a command named "tftp" that can connect to the TFTP server as well. Is the default TFTP root directory present and readable?

The "ICMP_DESTINATION_UNREACHABLE" or similar ICMP message signifies a serious error has occurred. Reset the board, and wait one minute for the TFTP server to time out, and terminate any open connections. Verify that the IP addresses for the server and gateway are correct.

Appendix B FPLA CODE

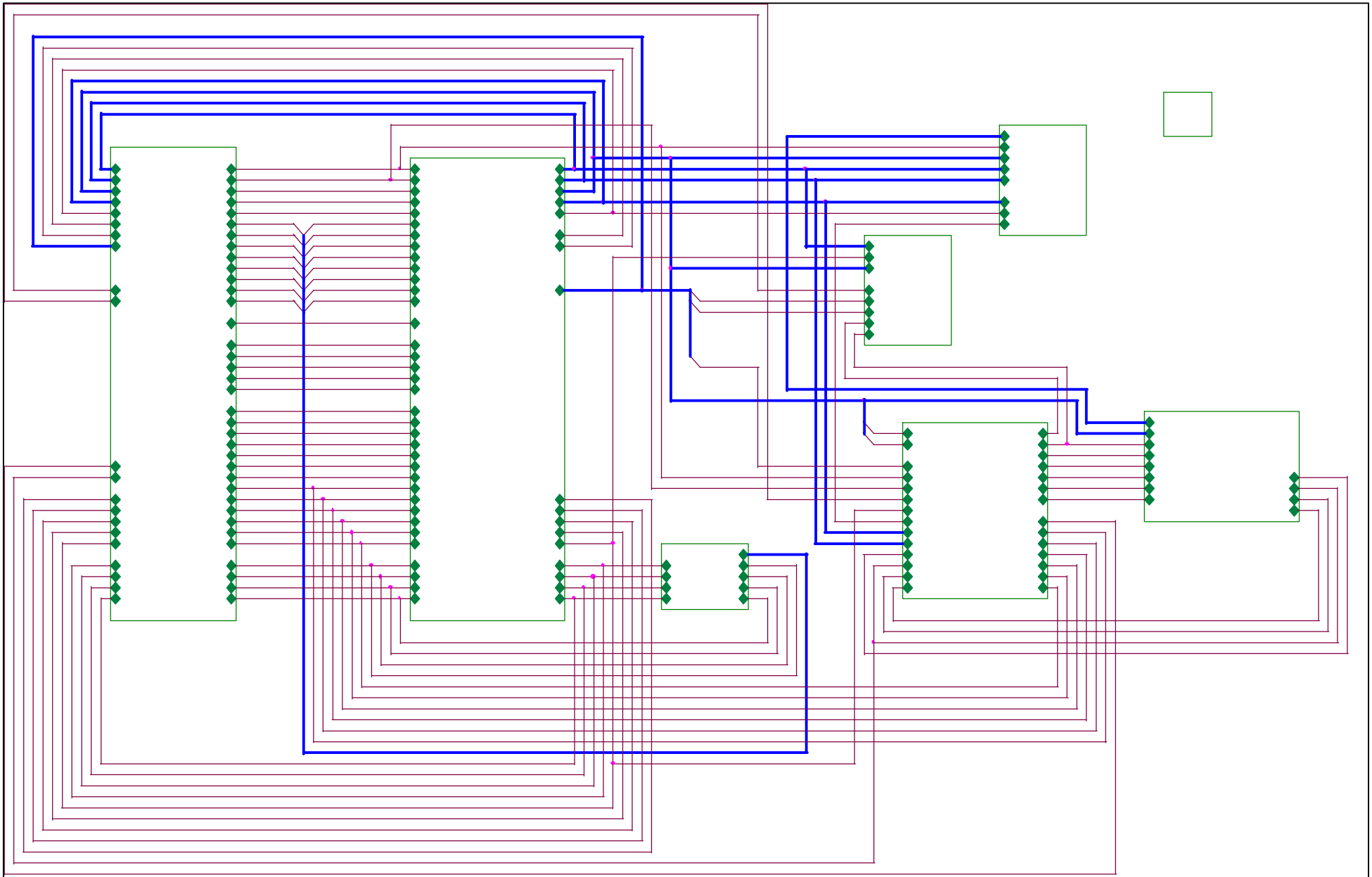
```
module isa2
title 'isa  controller'
"Feb 26 '98 version v1 of the 5206e
"isa2 device 'ispLSI';
;"***************************************************"
;"This abel file contains the code for a NE2000 compatible ethernet"
;"for the 55206e Coldfire processor as well as reset and IRQ7 (abort)"
;"It was targeted to Lattice ispLSI 2032 fpga  "
;"CS: B3D3                                     "
;"***************************************************"
;"***************************************************"
;"Declaration Section                             "
;"***************************************************"
;" constants"
   C,P,X,Z,H,L = .C.,.P.,.X.,.Z.,1,0;
;"***************************************************"
DLYIOCHRDY0 node ISTYPE 'reg_d,buffer';
DLYIOCHRDY,ENDIT,END16,END8 node;
STARTISA node ISTYPE 'reg_d,buffer';
SBHE,IOR,IOW,ISAOE node;
DA,DLYDA node ISTYPE 'reg_d,buffer';
ABORTML,DAOE,CLK16MHZ,CLK8MHZ node ISTYPE 'reg_d,buffer';


CLK2MHZ      node ISTYPE 'reg_d,buffer';

RSTMH node;
BCLK0 node ISTYPE 'reg_d,buffer';
BCLK1 node ISTYPE 'reg_d,buffer';
BCLK2 node ISTYPE 'reg_d,buffer';


DSL, ISASWAPL, SBHEL, ISAOEL, DTACKL, ISASELL node;


ABORTOL      pin 3 ISTYPE 'reg_d, buffer';
RST_L        pin 4;              "Output - to ColdFire reset
DB_CS_L            pin 5;              "Output - Data buffer enable for ethernet
A0IN         pin 6;              "INPUT  - A0 received from CF through buffers
IOCHRDY            pin 7;              "Input  - asserted by ethernet
IOCS16L            pin 9;              "Input  - asserted by ethernet
SIZ1         pin 10;
XCLK0        pin 11;                   "Input  - global clock
IOWL         pin 15;                   "Input  - write signal from ethernet
RD           pin 16;                   "INPUT  - R/-W from the ColdFire
CLK4MHZ      pin 17 ISTYPE 'reg_d,buffer';
BALE         pin 18;                   "Output - address latch enable
A0           pin 19;                   "OUTPUT - A0 sent to the ethernet
CS3_L        pin 22;                   "Input  - From ColdFire
HIZ_INLOW    pin 25;                   "Input  - From Header to allow a HIZ
PORIN_L      pin 26;                   "Input  - Suppy Voltage Supervisor

EIRQ         pin 28;                   "Input  - Ethernet IRQ 3
IRQ4_L       pin 29;                   "Output - IRQ 3 into the ColdFire
ETHER_RST    pin 30;                   "Input  - Hard Reset switch
ABORTIL      pin 31;                   "INPUT  - abort signal received from the
Abort swith
HIZ_L        pin 32;                   "Output - to ColdFire *HIZ
IORL         pin 37;                   "Input  - read signal from ethernet
OE_FLASH_L   pin 38;                   "Output - Flash output enable
A16          pin 39;
TAL          pin 40;                   "Input / Output - Transfer acknowledge
BDM_RST            pin 41;                   "Input - Reset from the BDM
SIZ0         pin 43;
"BDM_RST_L   pin 44;             Input  - BDM reset input

XCLK1        pin 35;                   "Clock - 20MHz
```

```
; "*******************************"
; " Lattice attributes            "
; "*******************************"
pLSI property 'CLK CLK54 CLK0 ';
pLSI property 'CLK XCLK0 CLK1 ';
pLSI property 'CLK CLK8MHZ SLOWCLK ';
pLSI property 'ISP ON';
pLSI property 'PULLUP ON';
pLSI property 'Y1_AS_RESET OFF';

; "-------------------------------"
; " Output inverter macro         "
; "-------------------------------"
OB21 MACRO (XO0, A0)
        {
        ?XO0 = !?A0;
};

; "-------------------------------"
; " Tristate Output inverter macro "
; "-------------------------------"
OT21 MACRO (XO0, A0, OE)
        {
        ?XO0.OE = ?OE;
        ?XO0 = !?A0;
};


CBU43 MACRO (Q0,Q1,Q2,CLK,EN,CS)
{
        [?Q0..?Q2].clk = ?CLK;
        ?Q0.D = ?Q0.Q & !?CS $ ?EN & !?CS ;
        ?Q1.D = ?Q1.Q & !?CS $ ( ?Q0.Q & ?EN & !?CS  );
        ?Q2.D = ?Q2.Q & !?CS $ ( ?Q0.Q & ?Q1.Q &  ?EN & !?CS );
};

equations


;"##########################################"
;"Bidirectional circuit equations"
;"##########################################"

OT21 (TAL, DA, DAOE)
OB21 (IORL, IOR)
OB21 (IOWL, IOW)

"new for the M5206e

OE_FLASH_L = !RD;

"Same as the M5206e

IRQ4_L =  !EIRQ # !PORIN_L;

"used to enable the data buffers to the ethernet
!DB_CS_L = !ETHER_RST & !CS3_L;

ABORTML := ABORTIL ;

ABORTML.clk  = CLK4MHZ ;

ABORTOL := ABORTML ;

ABORTOL.clk  = CLK4MHZ ;

"Ethernet reset
ETHER_RST = !PORIN_L # !BDM_RST;
```

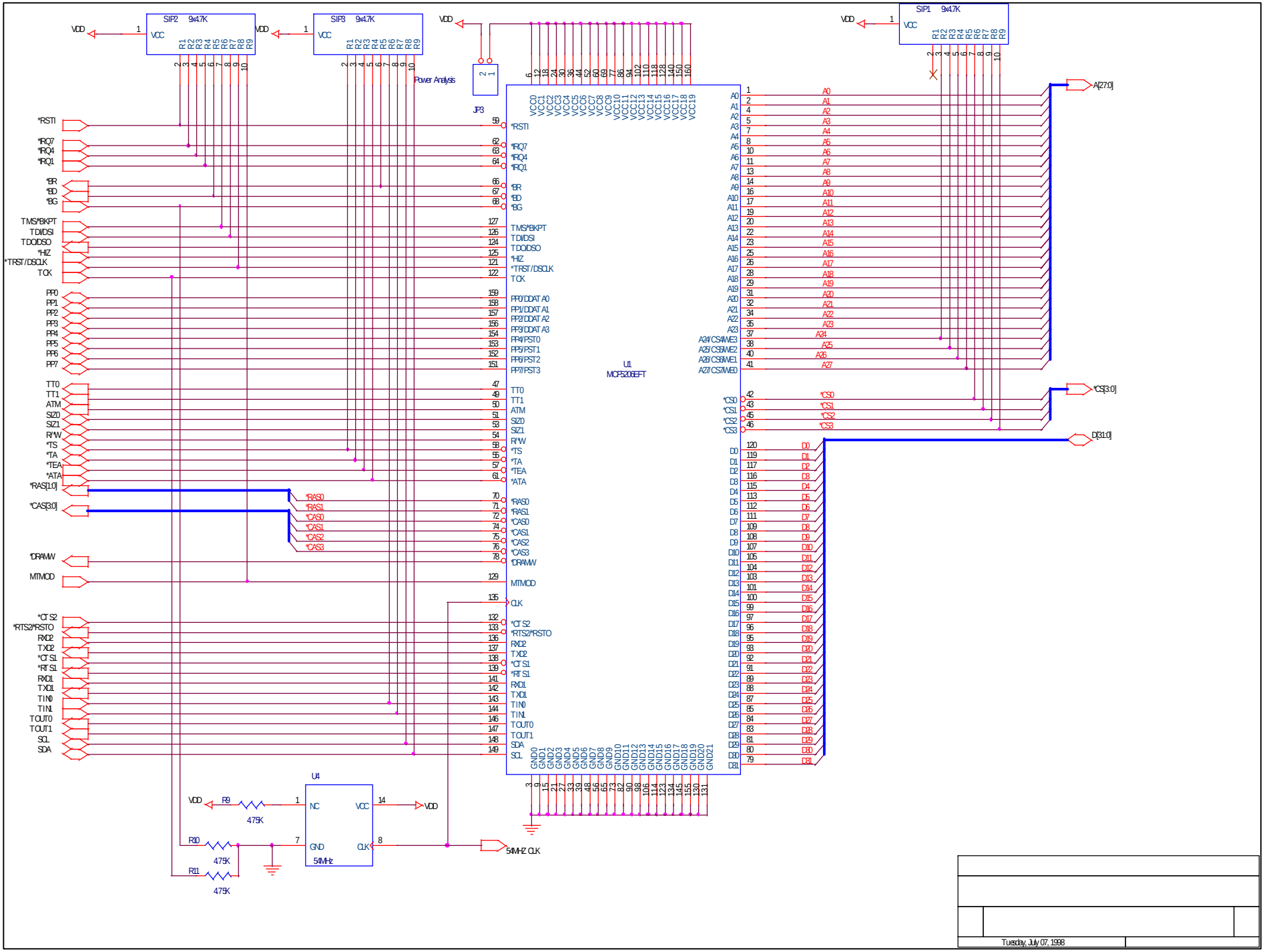
```
"CPU reset
!RST_L = !PORIN_L # !BDM_RST;

!HIZ_L = !RST_L # !HIZ_INLOW; " # !BDM_RST;

DAOE :=      !CS3_L # DA;

DAOE.clk = XCLK0 ;

A0 =  !SIZ1 & SIZ0 & !A0IN # A16 ;

SBHE =  STARTISA & !SIZ1 &  SIZ0 & !A0IN #
        STARTISA &  SIZ1 & !SIZ0 & !A0IN #
       STARTISA & !SIZ1 & !SIZ0 & !A0IN ;

CLK16MHZ := !CLK16MHZ ;

CLK16MHZ.clk = XCLK0 ;

CLK8MHZ := CLK8MHZ & !CLK16MHZ #
       !CLK8MHZ & CLK16MHZ ;

CLK8MHZ.clk = XCLK0 ;

CLK4MHZ := CLK4MHZ $ ( CLK16MHZ & CLK8MHZ );

CLK4MHZ.clk = XCLK0 ;

CLK2MHZ := CLK2MHZ $ ( CLK4MHZ & CLK16MHZ & CLK8MHZ );

CLK2MHZ.clk = XCLK0 ;

DA :=  !CS3_L & END16 & ENDIT & !IOCS16L & RD & !CLK4MHZ & SBHE #
       !CS3_L & END8 & ENDIT & RD & !CLK4MHZ #
         DLYDA  & !CS3_L #
         DA & !CS3_L;

DA.clk=XCLK0;

DLYDA :=!CS3_L & END16 & ENDIT & !IOCS16L & !RD & !CLK4MHZ & SBHE #
       !CS3_L & END8 & ENDIT & IOCS16L & !RD & !CLK4MHZ #
       !CS3_L & END8 & ENDIT & !SBHE & !RD & !CLK4MHZ ;

DLYDA.clk=XCLK0;

STARTISA := !CS3_L & !ENDIT ;

STARTISA.clk  = CLK4MHZ ;

CBU43 (BCLK0,BCLK1,BCLK2,CLK4MHZ,STARTISA,!STARTISA)

BALE   = STARTISA & !CLK4MHZ &  !BCLK2 & !BCLK1 & !BCLK0 & !IOR & !IOW ;

IOR    = STARTISA & !BCLK2 & !BCLK1 & BCLK0 & !CLK4MHZ & RD #
         IOR & !CS3_L ;

IOW    = STARTISA & !BCLK2 & !BCLK1 & BCLK0 & !CLK4MHZ & !RD #
         IOW & STARTISA ;

END16 = !BCLK2 & BCLK1 & !BCLK0 & !CLK4MHZ#
       END16 & STARTISA ;

END8 = BCLK2 & !BCLK1 & BCLK0 & !CLK4MHZ #
       END8 & STARTISA ;

ENDIT =      END16 & !IOCS16L & IOCHRDY & DLYIOCHRDY0 & DLYIOCHRDY & SBHE &
STARTISA#
       END8  & IOCS16L & IOCHRDY & DLYIOCHRDY0 & DLYIOCHRDY & STARTISA #
       END8  & !SBHE   & IOCHRDY & DLYIOCHRDY0 & DLYIOCHRDY & STARTISA ;

DLYIOCHRDY0:= IOCHRDY;

DLYIOCHRDY0.clk  = CLK4MHZ ;
```

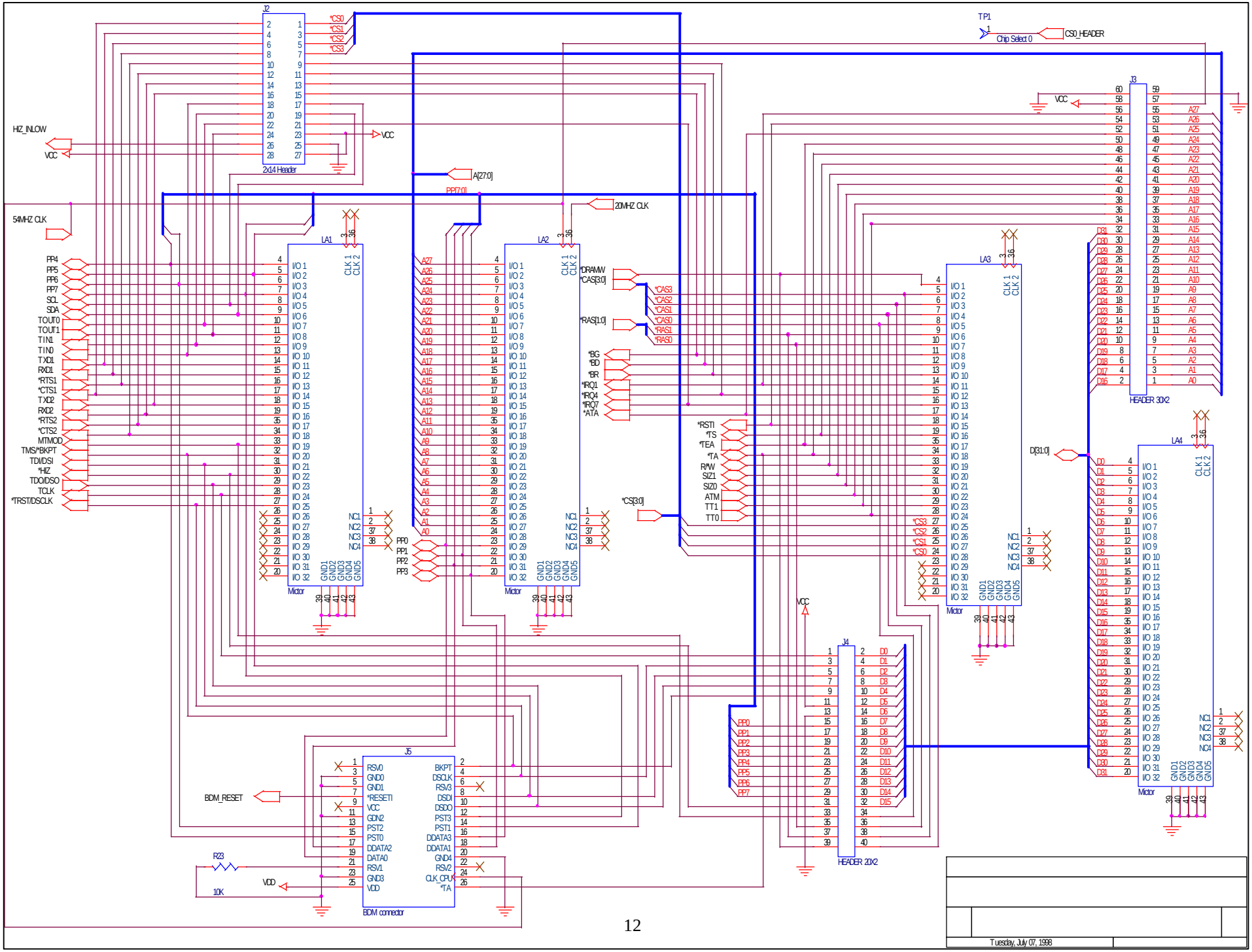
```
DLYIOCHRDY = IOCHRDY & CLK4MHZ #
          DLYIOCHRDY & !CLK4MHZ ;
```

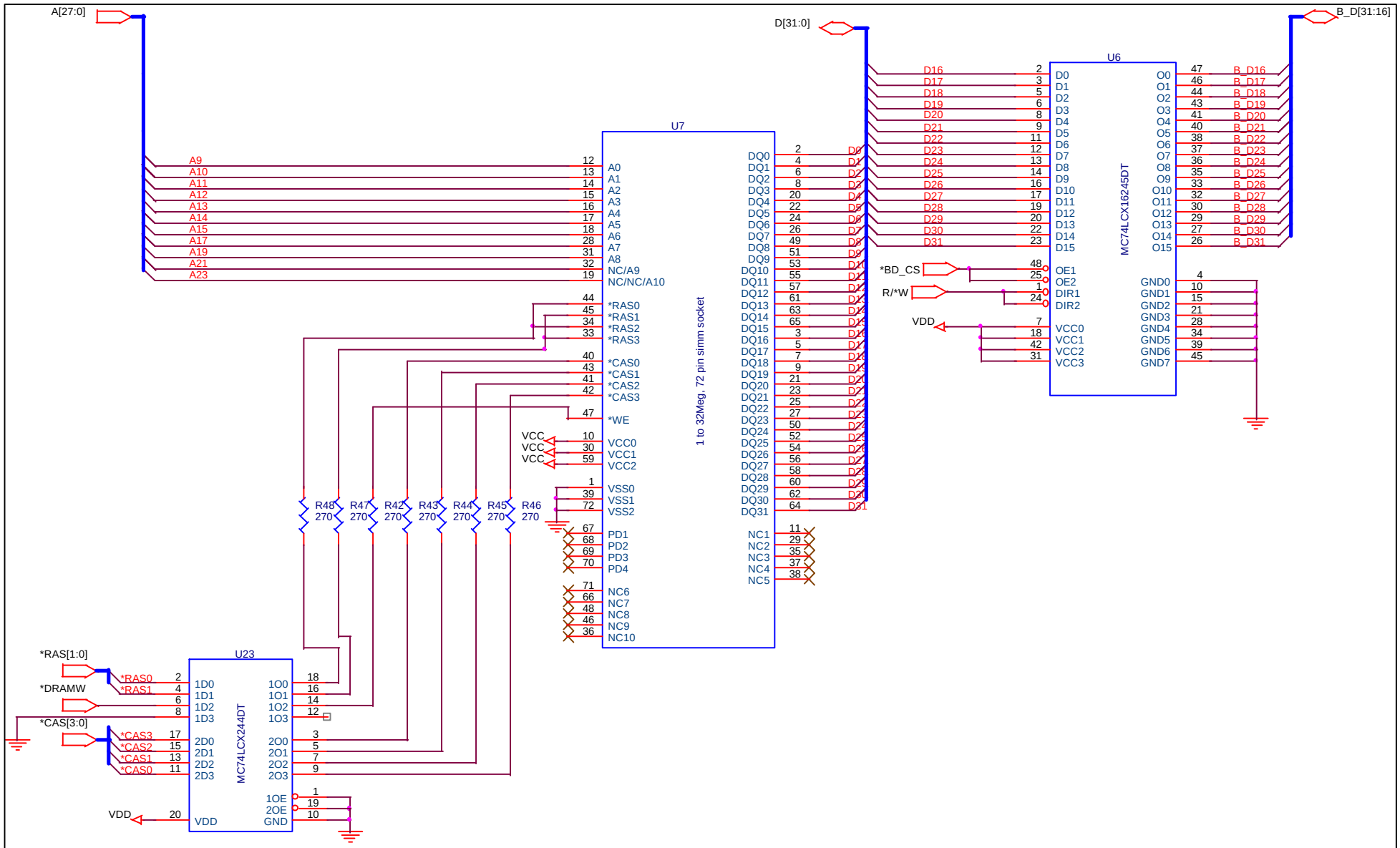
Appendix C SCHEMATICS



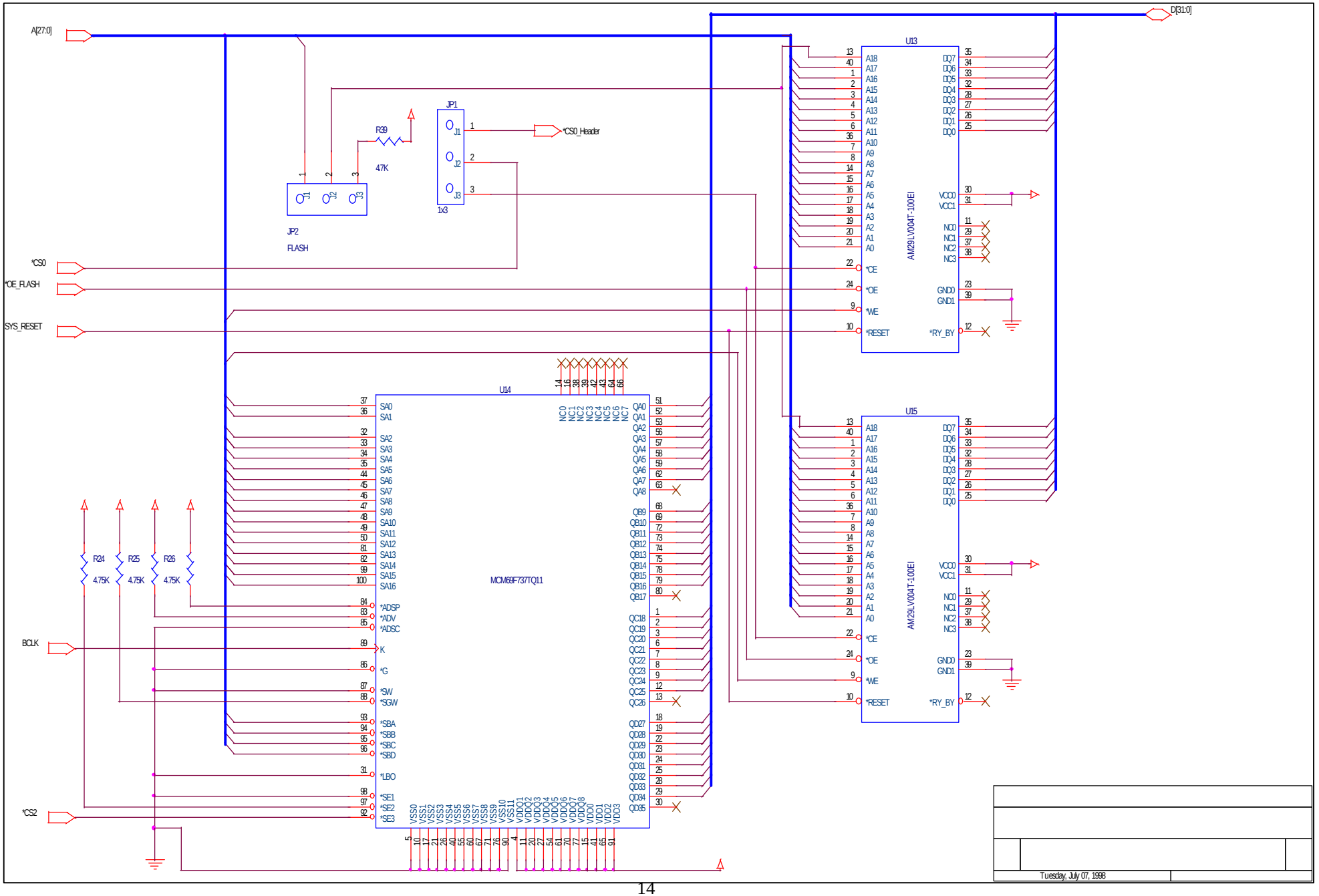
Tuesday, July 07, 1998

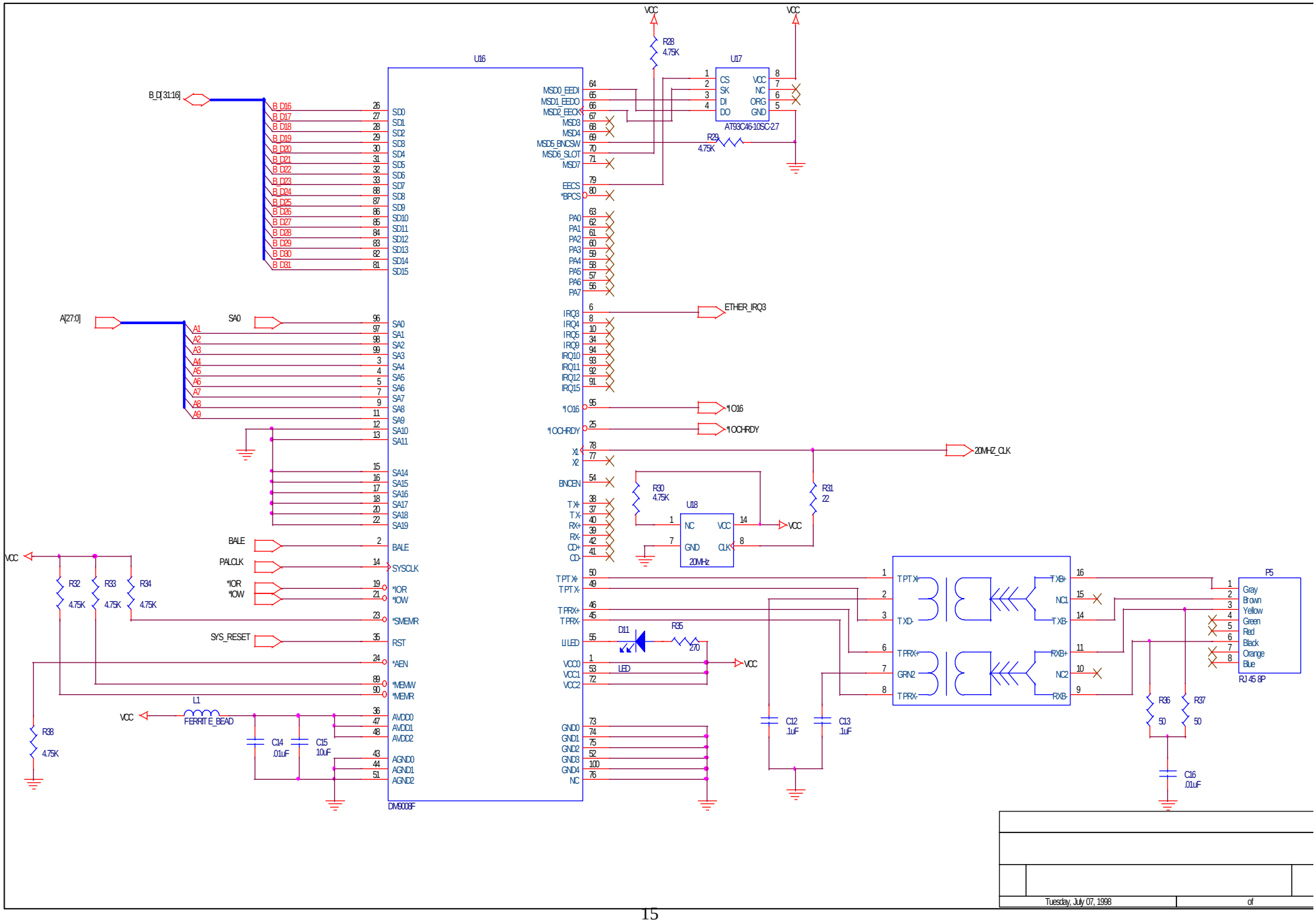

SIP2 9x4.7K
SIP3 9x4.7K
SIP1 9x4.7K
VDD
VCC
Power Analysis
JP3
U1
MCP5206EFT
*RSTI
*IRQ7
*IRQ4
*IRQ1
*BR
*BD
*BG
TMS/*BKPT
TDI/DSI
TDO/DSO
*HIZ
*TRST/DSCLK
TCK
PP0
PP1
PP2
PP3
PP4
PP5
PP6
PP7
TT0
TT1
ATM
SIZ0
SIZ1
R/W
*TS
*TA
*TEA
*ATA
*RAS[1:0]
*CAS[3:0]
*DRAMW
MTMOD
CLK
*CTS2
*RTS2/*RSTO
RXD2
TXD2
*CTS1
*RTS1
RXD1
TXD1
TIN0
TIN1
TOUT0
TOUT1
SCL
SDA
A[27:0]
*CS[3:0]
D[31:0]
U4
NC
GND
CLK
54MHz
R9
R10
R11
475K
54MHZ CLK
Tuesday, July 07, 1998

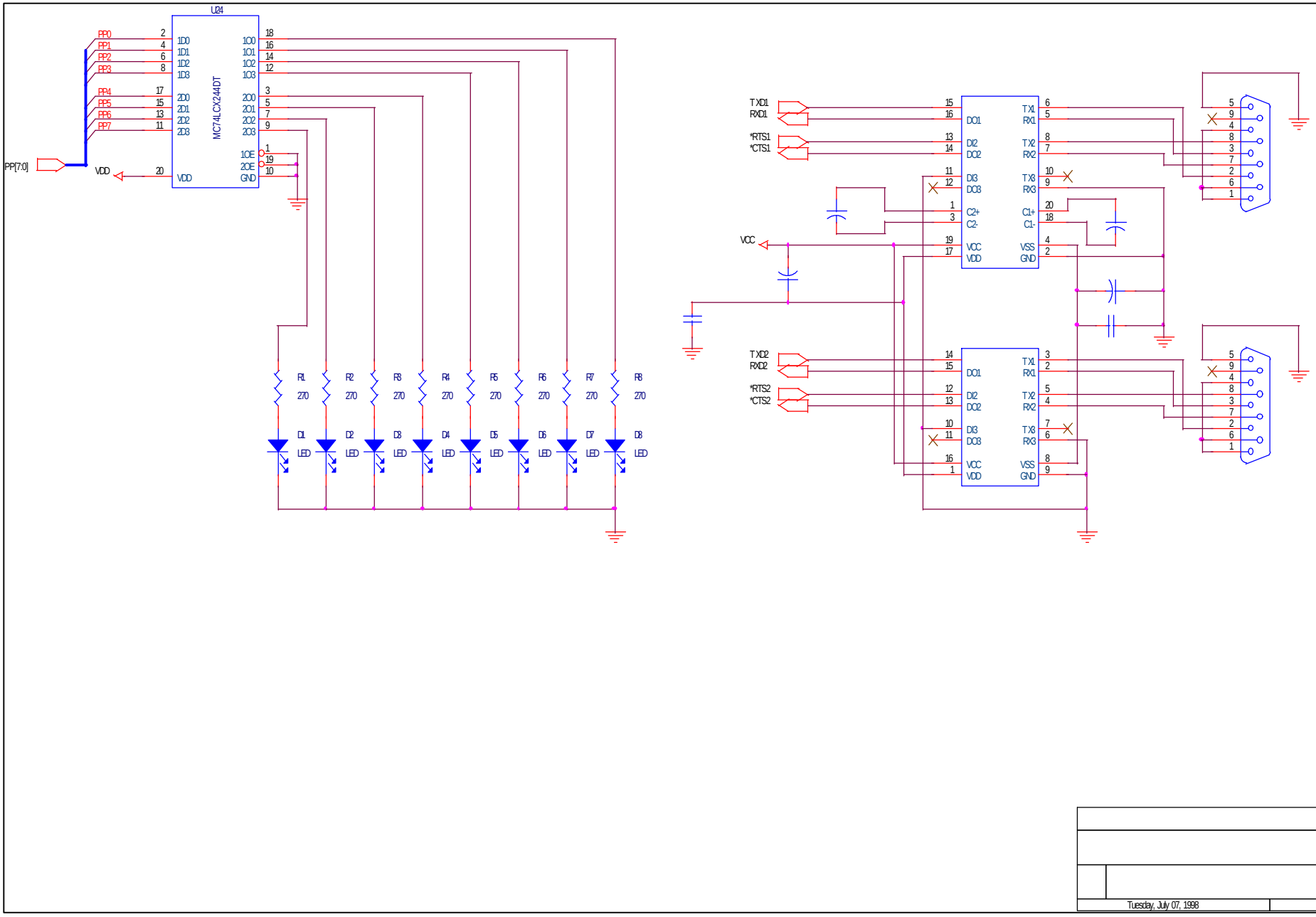

J2
2x14 Header
*CS0
*CS1
*CS2
*CS3
HIZ_INLOW
VCC
A[27:0]
PP[7:0]
54MHZ CLK
20MHZ CLK
LA1
LA2
LA3
LA4
Micter
PP4
PP5
PP6
PP7
SCL
SDA
TOUT0
TOUT1
TIN1
TIN0
TXD1
RXD1
*RTS1
*CTS1
TXD2
RXD2
*RTS2
*CTS2
MTMOD
TMS/*BKPT
TDI/DSI
*HIZ
TDO/DSO
TCLK
*TRST/DSCLK
PP0
PP1
PP2
PP3
*DRAMW
*CAS[3:0]
*CAS3
*CAS2
*CAS1
*CAS0
*RAS[1:0]
*RAS1
*RAS0
*BG
*BD
*BR
*IRQ1
*IRQ4
*IRQ7
*ATA
*RSTI
*TS
*TEA
*TA
R/W
SIZ1
SIZ0
ATM
TT1
TT0
*CS[3:0]
TP1
Chip Select 0
CS0_HEADER
J3
HEADER 30X2
D[31:0]
J4
HEADER 20X2
J5
BDM connector
BDM_RESET
R23
10K
VDD
RSV0
GND0
GND1
*RESET!
VCC
GDN2
PST2
PST0
DDATA2
DATA0
RSV1
GND3
VDD
BKPT
DSCLK
RSV3
DSDI
DSDO
PST3
PST1
DDATA3
DDATA1
GND4
RSV2
CLK_CPU
*TA
Tuesday, July 07, 1998

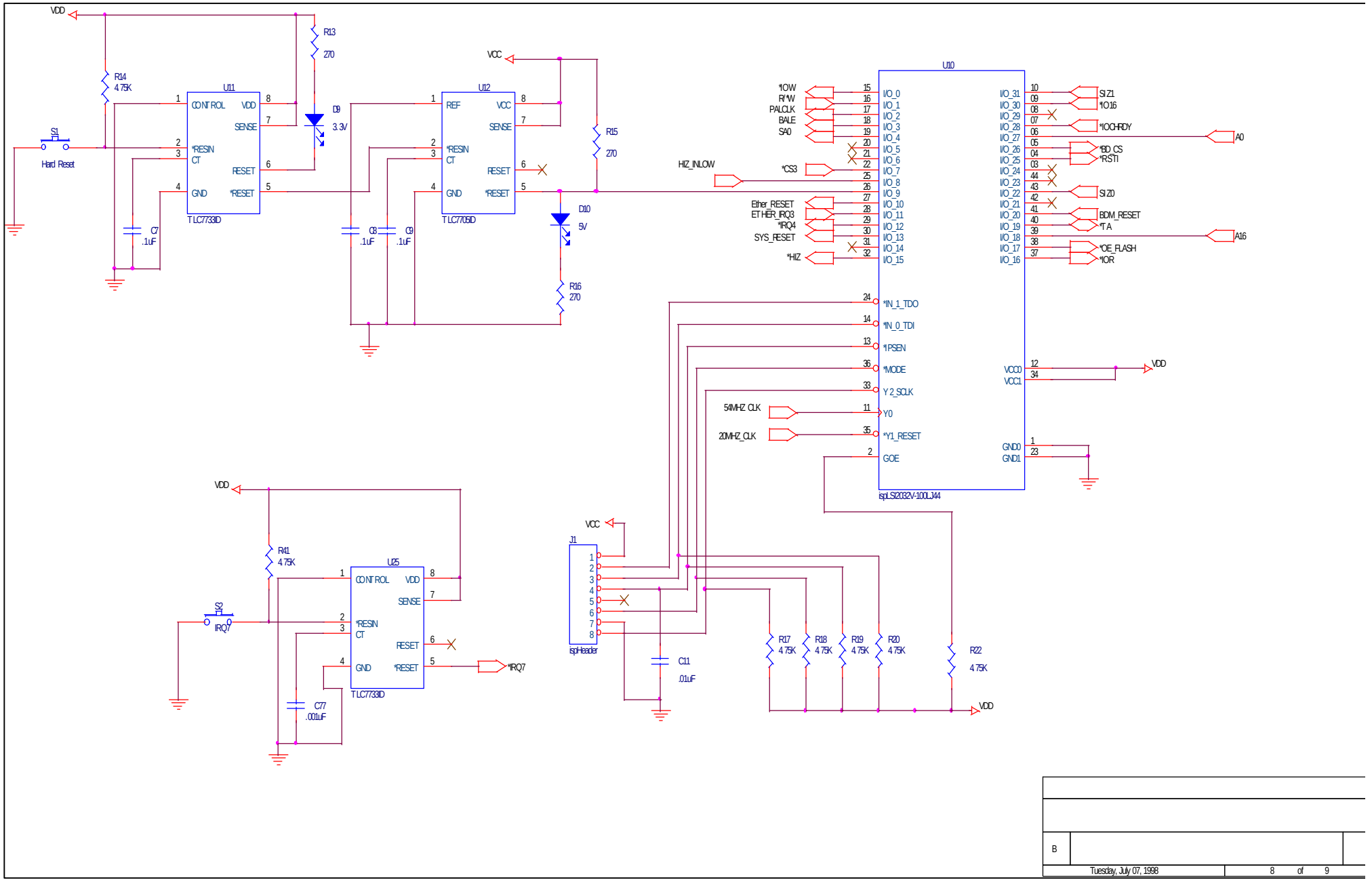


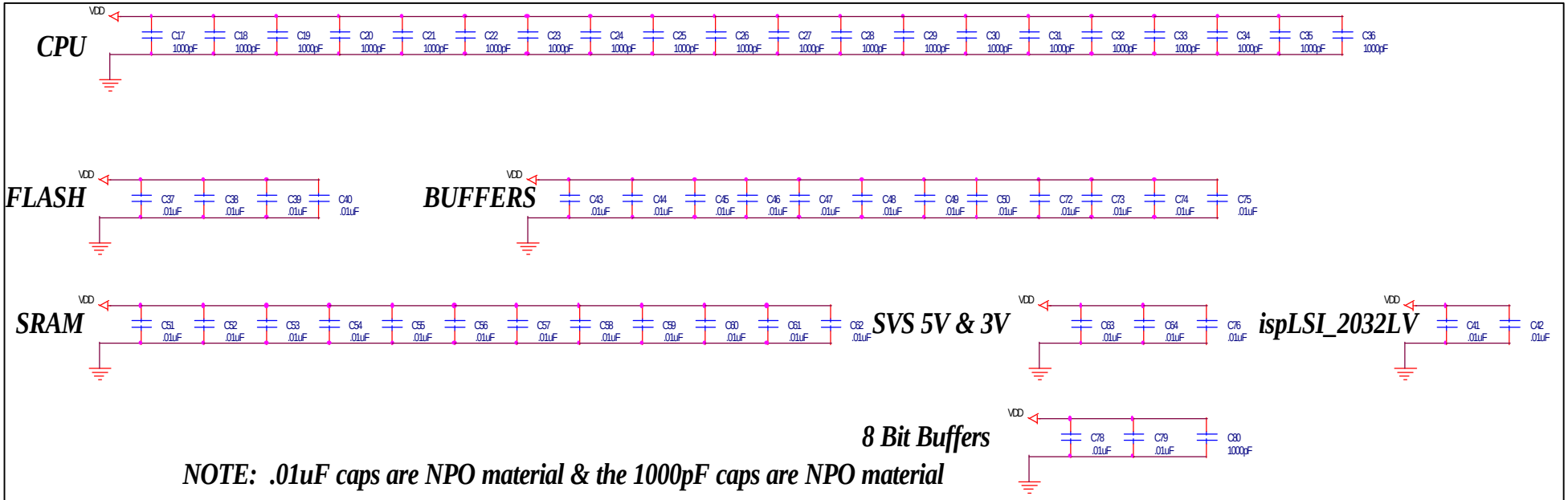
Tuesday, July 07, 1998


A[27:0]
D[31:0]
JP1
J1
J2
J3
1X3
*CS0_Header
R39
47K
JP2
FLASH
*CS0
*OE_FLASH
SYS_RESET
U13
U15
AM29LV004T-100EI
U14
MCM69F737TQ11
R24
R25
R26
4.75K
BCLK
*CS2
*ADSP
*ADV
*ADSC
*SW
*SGW
*SBA
*SBB
*SBC
*SBD
*LBO
*SE1
*SE2
*SE3
*CE
*OE
*WE
*RESET
*RY_BY
Tuesday, July 07, 1998

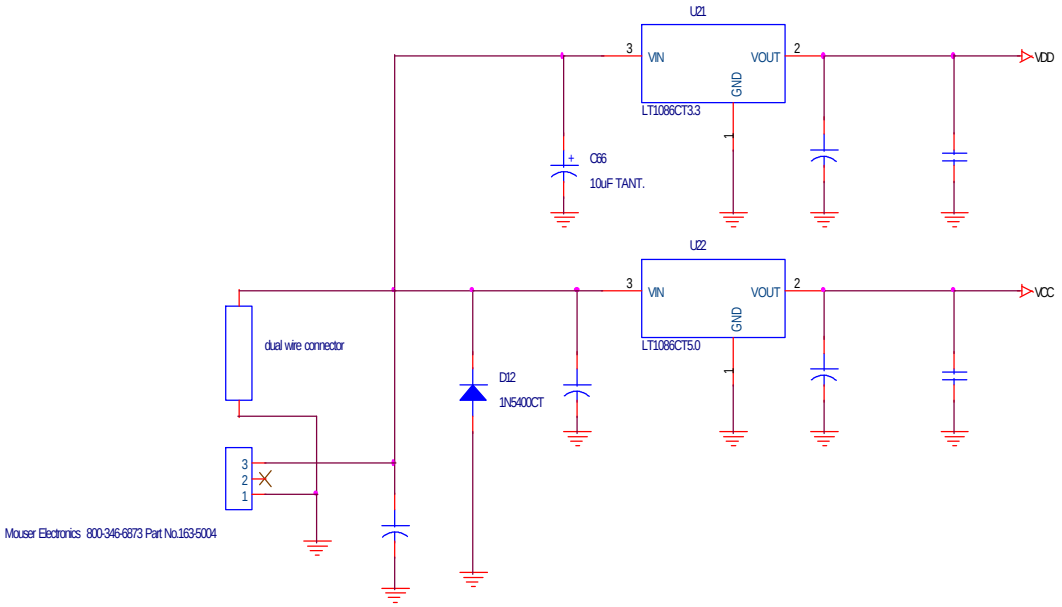

U16
DM9008F
U17
AT93C46-10SC-2.7
U18
20MHz
B_D[31:16]
A[27:0]
SA0
VCC
R28 4.75K
R29 4.75K
R30 4.75K
R31 22
R32 4.75K
R33 4.75K
R34 4.75K
R38 4.75K
L1
FERRITE_BEAD
C14 .01uF
C15 10uF
C12 .1uF
C13 .1uF
C16 .01uF
D11
R35 270
LED
ETHER_IRQ3
20MHZ_CLK
*IO16
*IOCHRDY
BALE
PALCLK
SYSCLK
*IOR
*IOW
*SMEMR
SYS_RESET
RST
*AEN
*MEMW
*MEMR
P5
RJ45 8P
Gray
Brown
Yellow
Green
Red
Black
Orange
Blue
R36 50
R37 50
Tuesday, July 07, 1998
of


U24
MC74LCX244DT
PP0
PP1
PP2
PP3
PP4
PP5
PP6
PP7
PP[7:0]
VDD
1D0
1D1
1D2
1D3
2D0
2D1
2D2
2D3
1OE
2OE
GND
R1 270
R2 270
R3 270
R4 270
R5 270
R6 270
R7 270
R8 270
D1 LED
D2 LED
D3 LED
D4 LED
D5 LED
D6 LED
D7 LED
D8 LED
TXD1
RXD1
*RTS1
*CTS1
TXD2
RXD2
*RTS2
*CTS2
VCC
DO1
DI2
DO2
DI3
DO3
C2+
C2-
VCC
VDD
TX1
RX1
TX2
RX2
TX3
RX3
C1+
C1-
VSS
GND
Tuesday, July 07, 1998


VDD
R13
270
R14
4.75K
U11
CONTROL
VDD
SENSE
*RESIN
CT
RESET
GND
*RESET
TLC7733ID
S1
Hard Reset
C7
.1uF
D9
3.3V
VCC
U12
REF
VCC
SENSE
*RESIN
CT
RESET
GND
*RESET
TLC7705ID
C8
.1uF
C9
.1uF
R15
270
D10
5V
R16
270
HIZ_INLOW
U10
*IOW
R/*W
PALCLK
BALE
SA0
*CS3
Ether_RESET
ETHER_IRQ3
*IRQ4
SYS_RESET
*HIZ
I/O_0
I/O_1
I/O_2
I/O_3
I/O_4
I/O_5
I/O_6
I/O_7
I/O_8
I/O_9
I/O_10
I/O_11
I/O_12
I/O_13
I/O_14
I/O_15
I/O_31
I/O_30
I/O_29
I/O_28
I/O_27
I/O_26
I/O_25
I/O_24
I/O_23
I/O_22
I/O_21
I/O_20
I/O_19
I/O_18
I/O_17
I/O_16
SIZ1
*IO16
*IOCHRDY
A0
*BD_CS
*RSTI
SIZ0
BDM_RESET
*TA
A16
*OE_FLASH
*IOR
*IN_1_TDO
*IN_0_TDI
*PSEN
*MODE
Y2_SCLK
54MHZ_CLK
Y0
20MHZ_CLK
*Y1_RESET
GOE
VCC0
VCC1
GND0
GND1
VDD
ispLSI2032V-100LJ44
R41
4.75K
U25
CONTROL
VDD
SENSE
*RESIN
CT
RESET
GND
*RESET
TLC7733ID
S2
IRQ7
*IRQ7
C77
.001uF
J1
ispHeader
C11
.01uF
R17
4.75K
R18
4.75K
R19
4.75K
R20
4.75K
R22
4.75K
B
Tuesday, July 07, 1998
8 of 9



NOTE: .01uF caps are NPO material & the 1000pF caps are NPO material



Title		
Size B	Document Number	Rev
Date: Tuesday, July 07, 1998	Sheet 9 of 9	

Appendix D MC5206EC3 BILL OF MATERIALS

ITEM	ASSY	PART NO.	MFG	VENDOR/ PART NO.	DESC.	REF. DES.
1	5	10uF 16V		DIG/PCE3031CT-ND	10uF	C1 - C4; C15
2	9	.1uF		VENKEL/C0805X7R500-104KNE	.1uF	C5 - C9; C12, C13, C67, C70
3	38	.01uF		VENKEL/C0805x7R500-103KNE	.01uF	C11, C14, C16, C37-C64; C72 - C76; C78,C79
4	22	1000pF		VENKEL/C0805COG500-102JNE	1000pF	C17 - C36; C77, C80
5	4	10uF TANT.		VENKEL/TA016TCM106KBR	10uF TANT	C65, C66, C68, C69
6	1	200 uF		DIG/PCE2048CT-ND	200 uF	C71
7	11	90T HSMG-T400	CM	NEW/09THSMG-T400	LED, GRN	D1 - D11
10	2	1 X 3	Berg	DIG/S1011-03-ND	1 X 3	JP1, JP2
11	1	ispHeader 1x 8	Berg	DIG/S1011-13-ND	ispHeader	J1
12	1	2 x 13 Header	Berg	DIG/S2011-13-ND	2 X 13 header	J2
13	1	Header 30 x 2	Berg	DIG/S2011-30-ND	Header 30 x 2	J3
14	1	Header 20 x 2	Berg	DIG/S2011-20-ND	Header 20 x 2	J4
15	1	BDM connector 13 x 2	Berg	DIG/S2011-13-ND	BDM connector	J5
16	4	767054-1	AMP	TIME/767054-1	Mictor 767054-1	LA1 - LA4
17	1	BLM31A700SPT	Murata Erie	NEWARK/BLM31A700SPT	Ferrite_Bead	L1
18	2	748875-1	AMP	NEWARK/66F1579	Conn, 9, D, Female, RA	P1, P2
19	1	2SV-02	Augut	NEWARK/46F897	Conn, 2-pin pwr	P3
20	1	RAPC722	SWC	NEWARK/93F7715	Power conn 3-pin	P4
21	12	270		VENKEL/CR0805-10W-271JT	270	R1-R8; R13, R15, R16, R35
22	21	4.75K		VENKEL/CR0805-10W-4751FT	4.75K	R9 - R11; R14, R17 - R20; R22, R24 - R30; R32 - R34; R38, R41
23	7	22 ohm		VENKEL/CF1206-8W-220JT	220 ohm 1206	R42 - R48
24	1	10K		VENKEL/CR0805-10W-103JT	10K	R23
25	1	22		VENKEL/CR0805-10W-220JT	22	R31
26	2	50 805 5% .1W		VENKEL/CR0805-10W-470JT	50	R36, R37
27	3	0 X 4.7K		DIG/4310R-1-472-ND	RPAK, 4.7K, 10-pin 9RES	SIP1-SIP3
28	1	Hard Reset - KS11R23CQD	C&K	ARROW/KS11R23CQD	Hard Reset-KS11R23CQD	S1

29	1	IRQ7 - KS11R22CQD	C&K	ARROW/KS11R22CQD	IRQ7 - KS11R23CQD	S2
30	1	Chip-Select 0 1 x 1 CONN		DIG/S1011-01-ND	Chip Select 0	TP1
31	1	MC145407DW	Motorola	NEWARK/MC145407DW	MC145407DW	U3
32	1	MC145406DW	Motorola	NEW/08T MC145406DW	MC145406DW	U2
33	1	MCF5206EFT	Motorola	BRAD	MCF5206EFT	U1
34	1	P1100-HCV 54 MHz	PLET	OMNIPRO/P1100-HC-53 125 MHz	XTAL, 54 MHz	U4
36	1	MC74LCX16245DT	Motorola	NEW/83F4287	MC72LCS16245DT	U6
37	1	822019-4	AMP	MOUSER/571-8220194	SKT, SIMM, 72-pin	XU7
38	1	ispLSI2032LV-100LJ ispLSI2032LV-110LJ ispLSI2032V-110LJ ispLSI2032V-100LJ	Lattice	INSIGHT ELECTRONICS	ispLSI2032LV-100LJ ispLSI2032LV-110LJ ispLSI2032V-110LJ ispLSI2032V-100LJ	U10
39	2	TL7733ID or TL7733IP	TI	LINTECH/TLC7733ID	TL7733ID OR TL7733IP	U11, U25
40	1	TL7705ID or TI7705IP	TI	LINTECH/TLC7705ID	TL7705ID or TI7705IP	U12
41	2	AM29LV004T-100EI	AMD	ARROW/AM29LV004T-100EC	AM29LV004T-100EI	U13, U15
42	1	MCM69F737TQ11	Motorola	BRAD/NO POP	MCM69F737TQ11	U14
43	1	DM9008F	Davicom	DAVICOM/DM9008F/B	DM9008F	U16
44	1	AT93C46-10SC-2.7	Atmel	BRAD/NO POP	AT93C46-10SC-2.7	U17
45	1	P1100-HVC 20 MHz	PLET	OMNIPRO/P1100-HC-2000MHz	XTAL, 20MHz	U18
46	1	FD22-101G	Halo EL	ERIC/FD22-101G	FD22-101	U19
47	1	555153-1	AMP	MOUSER/571-5551641	RJ 45 8P	P5
48	1	LT1086CT3.3		DIG/LT1086CM-3.3-ND	LT1086CM3.3	U21
49	1	LT1086CT5.0		DIG/LT1086CT-5-ND	LT1086CT5.0	U22
50	2	IC197-4004-2000	YAM	ERIC/IC197-4004-2000	SKT, TSOP44	XU13, XU15
51	1	1N5400		DIG/1N5400CT-ND	IN5400	D12
52	1	822275-1	AMP	DIG/A2142-ND	SKT, PLCC-44	XU10
53	1	1MX32 3.3V EDO 60ns	SMART MOD.	BRAD/SM532013091X656	SIMM	U7
54	1	.001UF 805		VENDEL/C0805-COG500-102JNE		C77
57	1	2 PIN CONNECTOR, 1 X 2		DIG/S1011-02-ND		JP3
58	3	SHORT JUMPER UNPLATED		DIG/929950-00-ND		
59	1	PAN PACIFIC SERIAL CABLE	PAN PACIFIC	S-9MF-6		
60	1	4-40 HEX NUTS	DIG	DIG/H216-ND		
61	1	4-40 X 1/4 SCREWS	DIG	DIG/H142-ND		
62	5	TAPERED SQ. WHITE PAD	DIG	DIG/SJ5518-9-ND		