M5272C3 User's Manual

M5272C3UM/D
Rev. 1.4, 08/2001

LIMITED WARRANTY

Matrix Design 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, Matrix Design will replace, at no charge, components that fail, provided the product is returned (properly packed and shipped prepaid) to Matrix Design at 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 Matrix Design, 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:

Matrix Design & Manufacturing, Inc.
2914 Montopolis Drive #290
Austin, TX 78741
Phone: (512) 385-9210
Fax: (512) 385-9224
http://www.cadreiii.com

The M5272C3 Evaluation Board is CE certified.

CONTENTS

Paragraph Number	Title	Page Number
	Chapter 1 M5407C3 Board	
1.1	General Hardware Description	1-1
1.2	System Memory	1-4
1.3	Serial Communication Channels	1-4
1.4	Parallel I/O Ports	1-4
1.5	Programmable Timer/Counter	1-5
1.6	PCI Controller	1-5
1.7	On Board Ethernet	1-5
1.8	System Configuration	1-5
1.9	Installation And Setup	1-7
1.9.1	Unpacking	1-7
1.9.2	Preparing the Board for Use	1-7
1.9.3	Providing Power to the Board	1-8
1.9.4	Selecting Terminal Baud Rate	1-8
1.9.5	The Terminal Character Format	1-8
1.9.6	Connecting the Terminal	1-8
1.9.7	Using a Personal Computer as a Terminal	1-8
1.10	System Power-up and Initial Operation	1-11
1.11	M5407C3 Jumper Setup	1-11
1.12	Using The BDM Port	1-13
	Chapter 2 Using the Monitor/Debug Firmware	
2.1	What Is dBUG?	2-1
2.2	Operational Procedure	2-3
2.2.1	System Power-up	2-3
2.2.2	System Initialization	2-4
2.2.2.1	Hard RESET Button.	2-5
2.2.2.2	ABORT Button.	2-5
2.2.2.3	Software Reset Command.	2-6
2.3	Command Line Usage	2-6
2.4	Commands	2-6

CONTENTS

Paragraph Number	Title	Page Number
2.5	TRAP #15 Functions	2-39
2.5.1	OUT_CHAR	2-39
2.5.2	IN_CHAR	2-39
2.5.3	CHAR_PRESENT	2-40
2.5.4	EXIT_TO_dBUG	2-40

Chapter 3
Hardware Description and Reconfiguration

3.1	The Processor and Support Logic	3-1
3.1.1	Processor	3-1
3.1.2	Reset Logic	3-1
3.1.3	HIZ Signal	3-2
3.1.4	Clock Circuitry	3-2
3.1.5	Watchdog Timer	3-2
3.1.6	Interrupt Sources	3-2
3.1.7	Internal SRAM	3-3
3.1.8	The MCF5407 Registers and Memory Map	3-4
3.1.9	Reset Vector Mapping	3-5
3.1.10	TA Generation	3-5
3.1.11	Wait State Generator	3-6
3.1.12	SDRAM DIMM	3-6
3.1.13	Flash ROM	3-7
3.1.14	JP15 Jumper and User's Program	3-7
3.2	Serial Communication Channels	3-7
3.2.1	MCF5407 UARTs	3-7
3.2.2	I2C Module	3-8
3.3	Real-Time Clock	3-8
3.4	Parallel I/O Port	3-8
3.5	On-Board Ethernet Logic	3-8
3.6	Connectors and Expansion Bus	3-11
3.6.1	Expansion Connectors - J1 and J2	3-11
3.6.2	The Debug Connector J5	3-13

Appendix A
Configuring dBUG for Network Downloads

Appendix B
ColdFire to ISA, IRQ7 and Reset Logic Abel Code

Appendix C

CONTENTS

Paragraph Number	Title	Page Number
	SDRAM MUX PAL Equation	
	Appendix D Evaluation Board BOM	
	Appendix E Schematics	
	Appendix F Errata	

TABLES

Table Number	Title	Page Number
1-1	Power Supply Connections	1-8
1-2	Jumper Settings	1-11
1-3	Jumper Settings -CS Databus Width	1-13
1-4	Jumper Settings Ethernet Controller Slew Rates	1-13
1-5	Jumper Settings Ethernet Controller Operation Modes	1-13
2-1	dBUG Command Summary	2-7
3-1	The M5272C3 Memory Map	3-5
3-2	J2 Connector Pin Assignment	3-12
3-3	J3 Connector pin assignment	3-13
D-1	MCF5272EVM_BOM	D-1

ILLUSTRATIONS

Figure Number	Title	Page Number
1-1	MCF5272C3 Block Diagram	1-3
1-2	Minimum System Configuration	1-6
1-3	Pin assignment for female P4 (Terminal) connector.	1-9
1-4	Jumper Locations	1-10
2-1	Flow Diagram of dBUG Operational Mode.	2-4
3-1	The J4 Connector pin assignment	3-15

Chapter 1
M5272C3 Board

The M5272C3 is a versatile single board computer based on the MCF5272 ColdFire® Processor. It may be used as a powerful microprocessor based controller in a variety of applications. With the addition of a terminal, it serves as a complete microcomputer system for reference design, development/evaluation, training and educational use. The user need only connect an RS-232 compatible terminal (or a personal computer with terminal emulation software) and power supply to have a fully functional system.

Provisions have been made to connect this board to additional user supplied peripherals, via the Microprocessor Expansion Bus connectors J2 & J3 (on the schematic diagram), to expand memory and I/O capabilities. Additional peripherals may require bus buffers to minimize additional bus loading.

The board has been designed to be configured specifically for a user's application. The user may upgrade the memory to 512K FSRAM, via the U9 footprint, or connect to a PC via the JR1 USB connector as a device.

1.1 General Hardware Description

The M5272C3 board provides SDRAM, Flash ROM, an Ethernet interface (10/100M bit/sec), RS232 and all the built-in I/O functions of the MCF5272 device for programming and evaluating the attributes of the microprocessor. The MCF5272 device is a member of the ColdFire® family of processors. It is a 32-bit processor with a 23-bit address bus 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 MCF5272 has a System Integration Module referred to as the SIM. This module incorporates many of the functions needed for system design. These include programmable chip-select logic, system protection logic, general purpose I/O and interrupt controller logic. The chip-select logic can select up to eight memory banks and peripherals in addition to two banks of DRAMs. The chip-select logic also allows the insertion of a programmable number of wait-states to allow slower memory or memory mapped peripherals to be used (refer to *MCF5272 User's Manual* by Motorola for detailed information about the SIM.). The M5272C3 uses five of the eight chip selects to access the Flash ROMs (-CS0), FSRAM (-CS2) (which is not populated on the board but may be added by the user) and SDRAM (-CS7). -CS5 is used to generate control signals for PLI socket 0 (J5) and -CS6 is used to

generate control signals for PLI socket 1 (J6). The PLI sockets J5 & J6 allow ISDN or POTs daughter cards to be connected to the system. If the user chooses not to populate either of the PLI connectors, -CS5 and -CS6 become available to the user. The DRAM controller is used to control two SDRAM devices providing 4MB of SDRAM memory configured as 1MBx32 longwords. All other functions of the SIM are available to the user.

Figure 1-1 shows the M5272C3 block diagram.

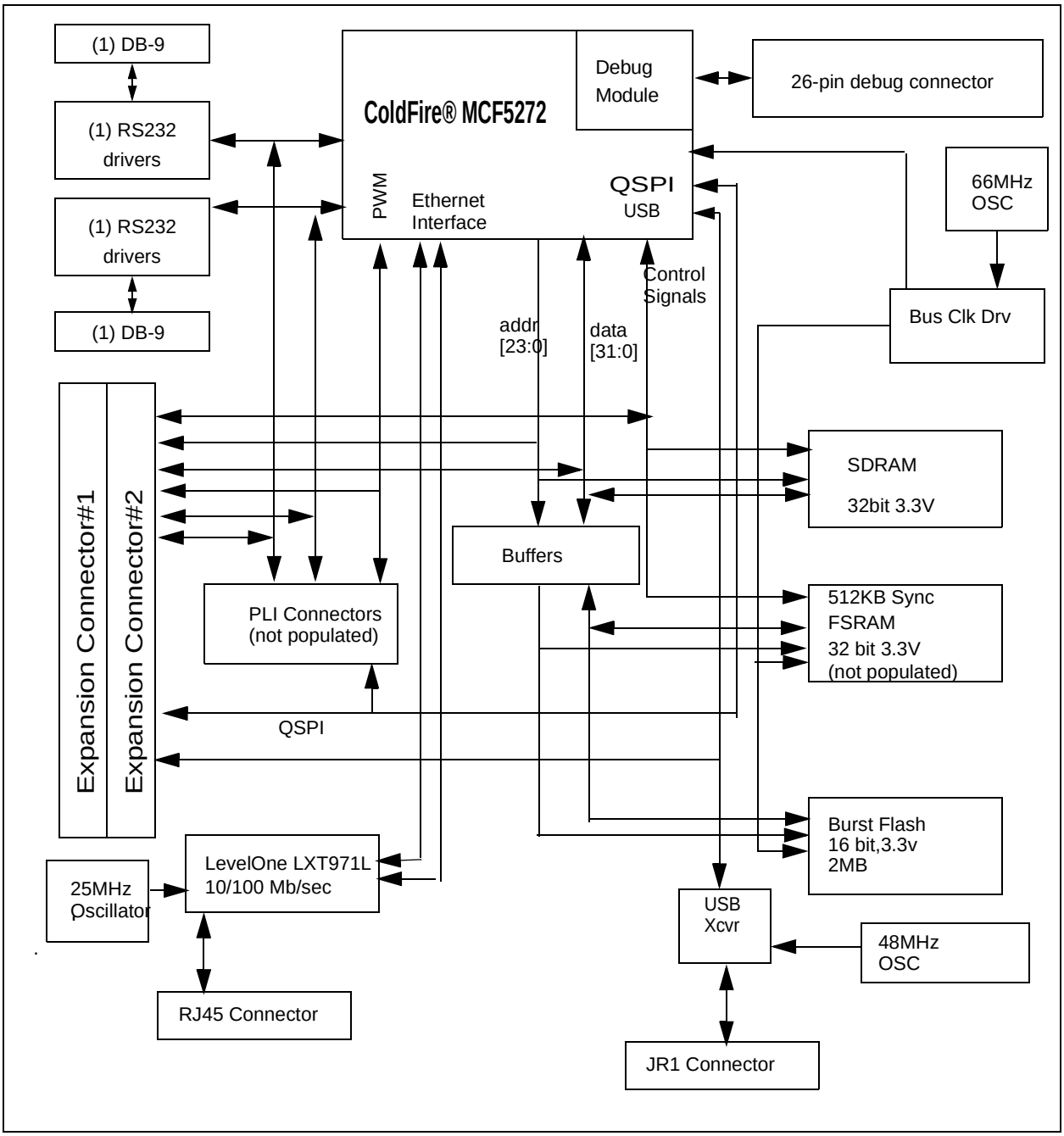


Figure 1-1. M5272C3 Block Diagram

1.2 System Memory

One on-board Flash ROM (U8) is used to store the M5272C3 dBUG debugger/monitor firmware in the lower 256 KBytes. The AM29PL160C-XX device contains 16Mbits of non-volatile storage (16 bits by 1 MByte) giving a total of 2MBytes of Flash memory.

The MCF5272 has 4KBytes of internal SRAM organized as 1KBx32bits. The SRAM can be used for either data or instruction space.

There are two SDRAM devices on the PCB. The system ships with 2 x 1M x 16 of SDRAM totalling 4MBytes of volatile memory. Various SDRAM manufacturers devices, as detailed on the schematics, are supported.

The internal cache of the MCF5272 is non-blocking. The instruction cache is 1 KByte with a 16-byte line size. The ROM Monitor currently does not utilize the cache, but programs downloaded with the ROM Monitor can initialise and use the cache.

The M5272C3 evaluation board has a foot print for a 512 KByte FSRAM but it is unpopulated.

1.3 Serial Communication Channels

The MCF5272 has 2 built-in UARTs (UART0 and UART1) with independent baud rate generators. The signals of both channels pass through external Driver/Receivers to make the channels RS232 compatible. An RS232 serial cable with DB9 connectors is included with the board. UART0 which is connected to P3 is used by the ROM monitor debugger dBUG for the user to access with a terminal. In addition, the signals of both channels are available on the 120 pin expansion connector J3. UART0 channel is the “TERMINAL” channel used by the debugger for communication with external terminal/PC. The “TERMINAL’ baud rate defaults to 19200.

1.4 Parallel I/O Ports

The MCF5272 offers 48-bits of general-purpose parallel I/O of which eight GPIO lines will be available at all times. Each pin can be individually programmed as input or output. The GPIO signals are configured as three ports each having up to 16 signals. These three GPIO ports are shared with other signals as follows :

Port A bits 6:0 are multiplexed with the signals required to interface to an external USB transceiver.

PA7 is multiplexed with SPI_CS3 and DOUT3.

Port A bits 15:8 are multiplexed with the pins of PLI TDM Ports 0 &1 and USART2

Port B bits 7:0 are multiplexed with the USART1 signals and the bus control signal /TA

Port B bits 15:8 are multiplexed with the Ethernet controller signals

Port C bits 15:0 are multiplexed with data bus signal D15:D0 and are only available when the device is configured for 16 bit data bus mode using the WSEL signal at power up.

1.5 Programmable Timer/Counter

The MCF5272 has four built-in general purpose timers/counters and a software watchdog timer. All four timers are available to the user. The signals for each timer are available on the 120 pin expansion connector J3.

1.6 USB Controller

The MCF5272 connects to the USB transceiver (U26) or directly via JR1 using the on-chip USB transceiver interface. The MCF5272 functions as a device on the USB bus.

1.7 On Board Ethernet

The M5272C3 has an on board Ethernet controller (Level One LXT971L) operating at 10M bits/sec or 100M bits/sec. The on-board dBUG ROM monitor is programmed to allow a user to download files from a network to memory in different formats. The current compiler formats supported are S-Record, COFF, ELF, or Image (raw binary). Refer to Appenix A for details on how to configure the board for network download.

1.8 System Configuration

The M5272C3 board requires only the following items for minimum system configuration:

- The M5272C3 board (provided).
- Power supply, +5V to 14V DC with minimum of 1.0 Amp.
- RS232C compatible terminal or a PC with terminal emulation software.
- RS232 Communication cable (provided).

Refer to Section 2.2.2, “System Initialization” for initial system setup.

Figure 1-2 displays minimum system configuration.

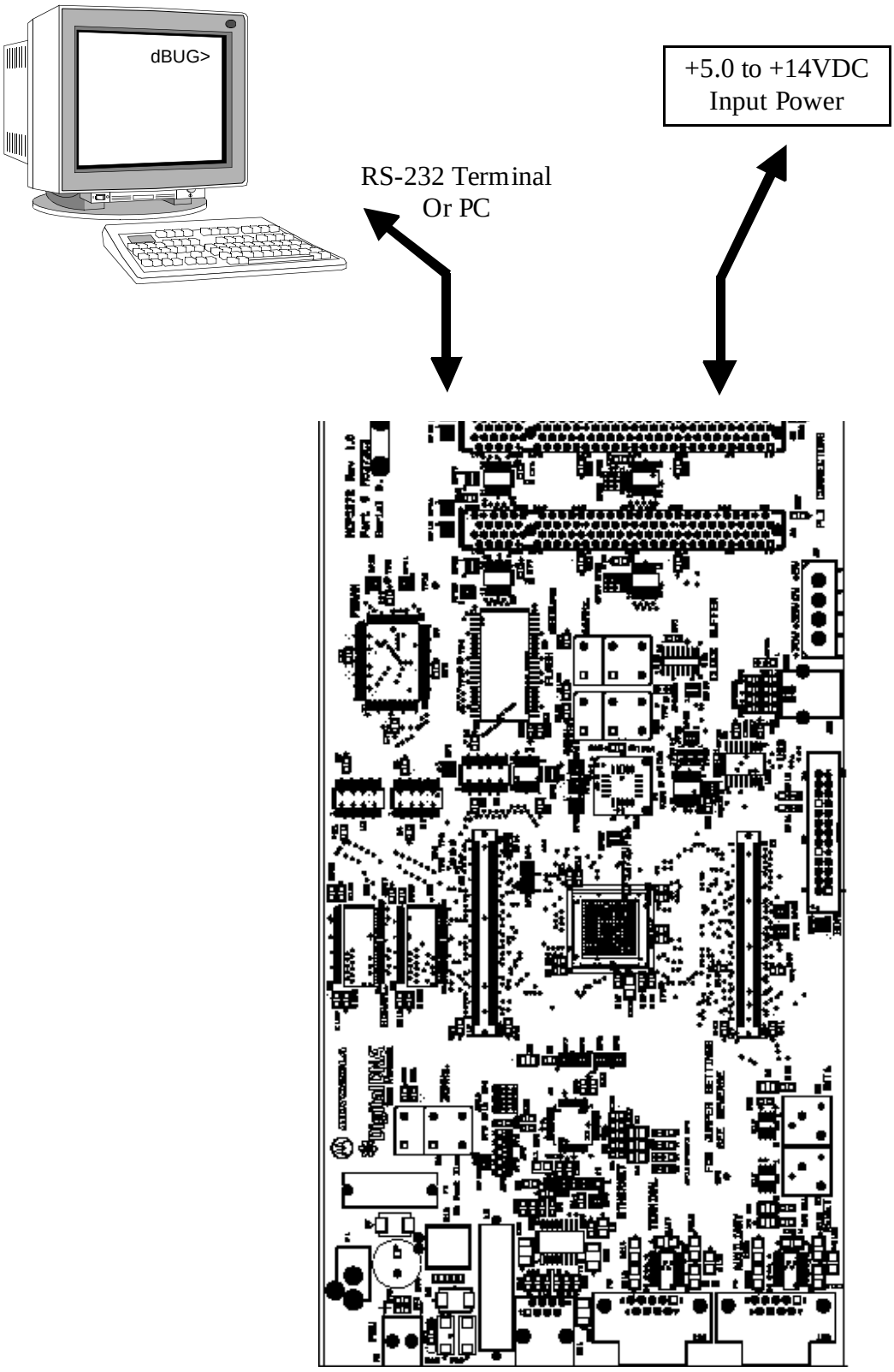


Figure 1-2. Minimum System Configuration

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. Default jumper markings are on the board next to the individual jumpers and a master jumper table is provided on the underside of the board. After the board is functional in its default mode, you may use the Ethernet by following the instructions provided in Appendix A.

1.9.1 Unpacking

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

- M5272C3 Single Board Computer
- M5272C3 User's Manual, this document
- One RS232 communication cable
- One BDM (Background Debug Mode) wiggler cable
- ColdFire® Programmers Reference Manual
- A selection of Third Party Developer Tools and Literature

NOTE:

Avoid touching the MOS devices. Static discharge can and will damage these devices.

Once you have verified that all the items are present, remove the board from its protective jacket and anti-static bag. Check the board for any visible damage. 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 Matrix Design immediately - for contact details please see the front of this manual.

1.9.2 Preparing the Board for Use

The board, as shipped, is ready to be connected to a terminal and power supply without any need for modification. Figure1.4 - Jumper Locations - shows the position of the jumpers and connectors.

1.9.3 Providing Power to the Board

The board accepts two means of power supply connection, either P1 or P2. Connector P1

is a 2.1mm power jack and P2 a lever actuated connector. The board accepts 7V to 14V DC at 1.5 Amp via either one of the connectors.

Table 1-1. Power Supply Connections on P2

Contact Number	Voltage
1	+7V to +14V DC
2	Ground

1.9.4 Selecting Terminal Baud Rate

The serial channel UART0 of MCF5272 is used for serial communication and has a built in timer. This timer is used by the dBUG ROM monitor to generate the baud rate used to communicate with a serial terminal. A number of baud rates can be programmed. On power-up or manual RESET, the dBUG ROM monitor firmware configures the channel for 19200 baud. Once the dBUG ROM monitor is running, a SET command may be issued to select any baud rate supported by the ROM monitor. Refer to Chapter 2 for the discussion of this command.

1.9.5 The Terminal Character Format

The character format of the communication channel is fixed at power-up or RESET. The default character format is 8 bits per character, no parity and one stop bit. It is neccessary to ensure that the terminal or PC is set to this format.

1.9.6 Connecting the Terminal

The board is now ready to be connected to a terminal. Use the RS232 male/female DB-9 serial cable to connect the PC to the M5272C3. The cable has a 9-pin female D-sub terminal connector at one end and a 9-pin male D-sub connector at the other end. Connect the 9-pin male connector to P3 connector on M5272C3. Connect the 9-pin female connector to one of the available serial communication channels normally referred to as COM1 (COM2, etc.) on the IBM PC or compatible. Depending on the kind of serial connector on the back of your PC, the connector on the PC may be a male 25-pin or 9-pin. It may be neccessary to obtain a 25pin-to-9pin adapter to make the connection. If an adapter is required, refer to Figure 1-3 which shows the pin assignment for the 9-pin connector on the board.

1.9.7 Using a Personal Computer as a Terminal

A personal computer may be used as a terminal provided a terminal emulation software package is available. Examples of this software are PROCOMM, KERMIT, QMODEM, Windows 95/98/2000 Hyper Terminal or similar packages. The board should then be connected as described in section 1.9.6, “Connecting the Terminal.”

Once the connection to the PC is made, power may be applied to the PC and the terminal emulation software can be run. In the terminal mode, it is neccessary to select the baud rate and 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. The character format should be 8 bits, no parity, one stop bit. (see section 1.9.5 The Terminal Character Format.) The baud rate should be set to 19200. Power can now be applied to the board.

Figure 1-3 shows pin assignments for a female terminal connector.

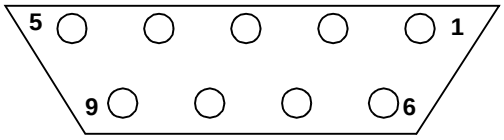


Figure 1-3. Pin assignment for female (Terminal) connector.

Pin assignments are as follows.

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 1-4 shows jumper locations.

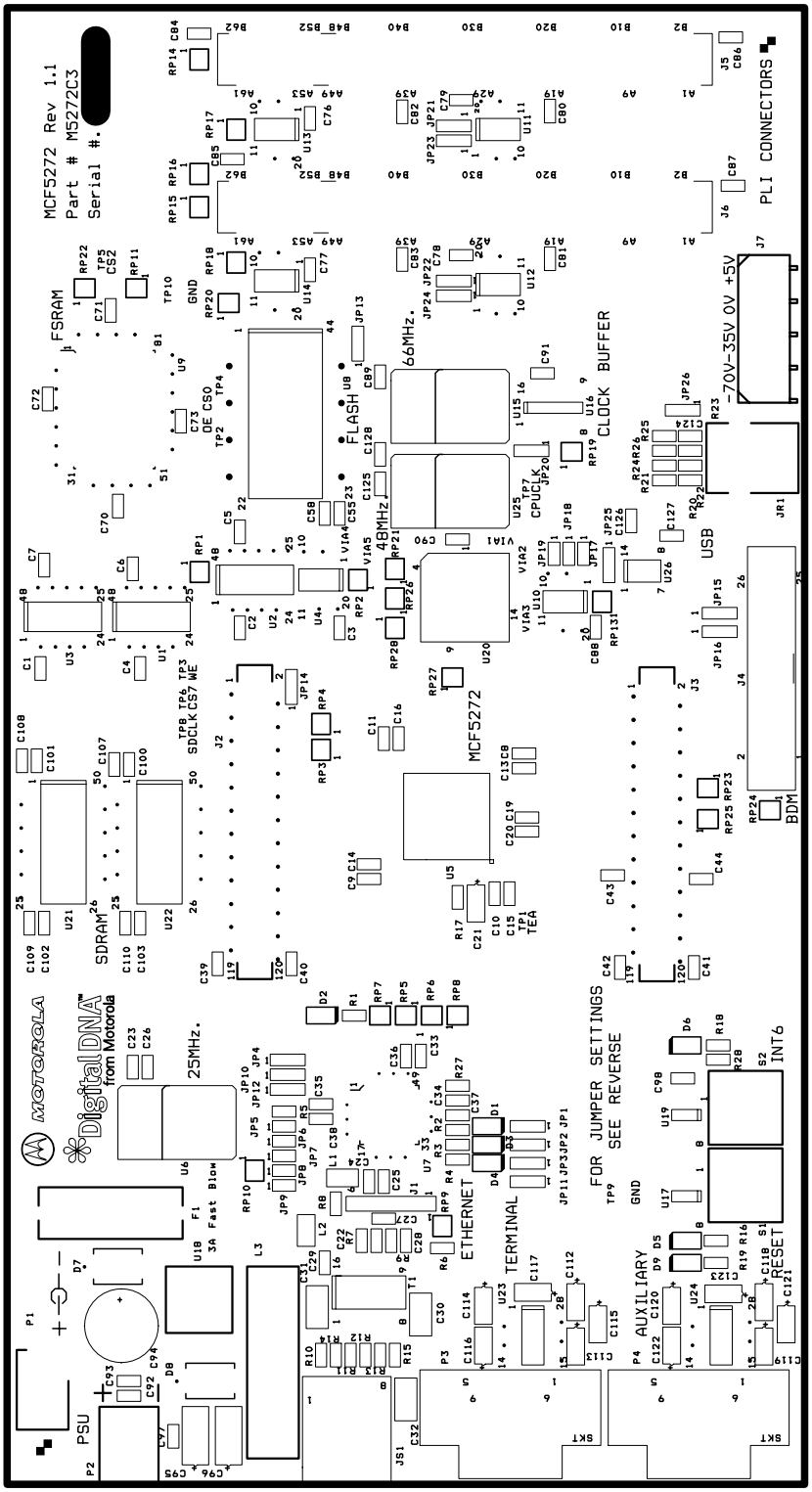


Figure 1-4. Jumper Locations

1.10 System Power-up and Initial Operation

When all of the cables are connected to the board, power may be applied. The dBUG ROM Monitor initialises the board and then displays a power-up message on the terminal, which includes the amount of memory present on the board.

```
Hard Reset
DRAM Size: 4M

Copyright 1995-2000 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5272 EVS Firmware v2e.1a.xx (Build XXX on XXX  XX 20XX
xx:xx:xx)

Enter 'help' for help.


dBUG>
```

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

1. Make sure that the power supply is properly configured for polarity, voltage level and current capability (~1A) and is connected to the board.
2. Check that the terminal and board are set for the same character format and baud.
3. Press the RESET button to insure 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 Matrix Design for further instructions, please see the beginning of this manual for contact details.

1.11 M5272C3 Jumper Setup

Jumper settings are as follows:

Note '*' is used to indicate that default setting.
'**' is used to indicate mandatory setting for proper operation.

Table 1-2. Jumper Settings

Jumper Setting		Function
JP1	* 1-2	Ethernet controller (U7) autonegotiate enabled
	2-3	Ethernet controller (U7) autonegotiate disabled
JP2	* 1-2	Ethernet controller (U7) handles BOTH 10 and 100 Base T operation
	2-3	Ethernet controller (U7) handles EITHER 10 OR 100 Base T Ethernet connection
JP3	*1-2	Ethernet controller (U7) Full Duplex operation
	2-3	Ethernet controller (U7) HalfDuplex operation
JP4	*1-2	Ethernet controller (U7) Management Data Enabled

Table 1-2. Jumper Settings (Continued)

Jumper Setting		Function
	2-3	Ethernet controller (U7) Management Data Disabled
JP5:9	Not applicable	Ethernet controller Device Address (Default 0)
JP10	*1-2	Set slew rate (in conjunction with JP12 - see Table 1-4)
JP11	*1-2	Ethernet Pause capability Disabled
JP12	*1-2	Set slew rate (in conjunction with JP10 - see Table 1-4)
JP13	**1-2	Flash EEPROM (U8) - boot into dBUG ROM monitor
	2-3	Flash EEPROM (U8) - boot into Flash user space ignoring first 256K
JP14	*1-2	MCF5272 SDRAM controller (U5) - enable RESET
	2-3	MCF5272 SDRAM controller (U5) - disable RESET
JP15	*1-2	MCF5272 (U5) Normal Operation
	2-3	MCF5272 (U5) Test Mode
JP16	*1-2	MCF5272 (U5) - BDM mode
	2-3	MCF5272 (U5) - JTAG mode
JP17	**ON	Databus width - 32 bits
	OFF	Databus width - 16 bits
JP18	**ON	Set /CS0 databus width - see Table 1-3
JP19	**OFF	Set /CS0 databus width - see Table 1-3
JP20	* 1-2	CPU Clock - 66MHz
	2-3	CPU Clock - 48MHz
JP21	* 1-2	Routes BD27 onto PLI 0 connector J5
	2-3	Routes Mux'd signal UART2CTS/SPI_CS2 5272 pin (K2) onto PLI 0 connector J5
JP22	* 1-2	Routes BD27 onto PLI 1 connector J6
	2-3	Routes Mux'd signal BUSW0/SPI_CS0 5272 pin (M5) onto PLI 1 connector J6
JP23	* 1-2	Routes BD30 onto PLI 0 connector J5
	2-3	Routes Mux'd signal PA11/SPI_CS1 5272 pin (L1) onto PLI 0 connector J5
JP24	*1-2	Routes BD30 onto PLI 1 connector J6
	2-3	Routes Mux'd signal DOUT3/SPI_CS3 - 5272 pin (P1) onto PLI 1 connector J6
JP25	*1-2	USB tranceiver (U26) - non-forced SEO USB operation
	2-3	USB tranceiver (U26) - forced SEO USB operation
JP26	*Not fitted	USB interface JR1 - supplies either 3.3V or 5.0V if fitted
	1-2	USB interface JR1 - board supplies 3.3V
	2-3	USB interface JR1 - board supplies 5.0V via J7

Table 1-3. -CS0 Databus width Jumper Setting

-CS Databus width

JP19	JP18	-CS0 Databus width
OFF	OFF	32-bit
OFF	ON	8-bit
ON	OFF	16bit
ON	ON	Reserved

Table 1-4. Ethernet Controller Slew Rate Jumper Settings

JP12	JP10	Ethernet Controller Slew Rate
1-2	1-2 (0)	2.5nS
1-2 (0)	2-3 (1)	3.1nS
2-3 (1)	1-2 (0)	3.7nS
2-3 (1)	2-3 (1)	4.3nS

Table 1-5. Ethernet Controller Operation Modes

AutoNegotiation	Speed MBPS	Duplex	JP1	JP2	JP3
Disabled	10	Half	L	L	L
Disabled	10	Full	L	L	H
Disabled	100	Half	L	H	L
Disabled	100	Full	L	H	H
Enabled	100 ONLY	Half	H	L	L
Enabled	100 ONLY	Full	H	L	H
Enabled	10/100	Half	H	H	L
Enabled	10/100	Full/Half	H	H	H

1.12 Using The BDM Port

The MCF5272 has a built in debug mechanism referred to as BDM (background debug module). The M5272C3 has the Motorola defined debug module connector, J4, to facilitate this connection. In order to use the BDM, simply connect the 26-pin connector at the end of the BDM wiggler cable provided by Motorola from P&E Microcomputer Systems to the J4 connector. No special setting is needed. Refer to the ColdFire® User's Manual BDM Section for additional instructions.

NOTE:

BDM functionality and use is supported via third party developer software and hardware tools, details for some of which may be found in this kit on CD-ROM.

Chapter 2
Using the Monitor/Debug Firmware

The M5272C3 single board computer has a resident firmware package that provides a self-contained programming and operating environment. The firmware, named dBUG, provides the user with monitor/debug interface, inline assembler and disassembly, program download, register and memory manipulation, and I/O control functions. This Chapter is a how-to-use description of the dBUG package, including the user interface and command structure.

2.1 What Is dBUG?

dBUG is a traditional ROM monitor/debugger that offers a comfortable and intuitive command line interface that can be used to download and execute code. It contains all the primary features needed in a debugger to create a useful debugging environment.

dBUG is a resident firmware package for the ColdFire® family single board computers. The firmware (stored in one 1Mx16 Flash ROM device) provides a self-contained programming and operating environment. dBUG interacts with the user through pre-defined commands that are entered via the terminal. These commands are defined in Section 2.4, “Commands.”

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.

dBUG assumes that an 80x24 character dumb-terminal is utilized to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit, 8N1. The default baud rate is 19200 but can be changed after the power-up.

The command line prompt is “dBUG> “. Any dBUG command may be entered from this prompt. dBUG does not allow command lines to exceed 80 characters. Wherever possible, dBUG displays data in 80 columns or less. dBUG echoes each character as it is typed, eliminating the need for any “local echo” on the terminal side.

In general, dBUG is not case sensitive. Commands may be entered either in upper or lower case, depending upon the user’s equipment and preference. Only symbol names require that the exact case be used.

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

The commands DI, GO, MD, STEP and TRACE are used repeatedly when debugging. dBUG recognizes this 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" allows the user program to utilize 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 2-1. After the 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 the discretion of the user. For the alternate case, the command will be executed under control of the dBUG firmware, and after command completion, the system returns 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:

- B8-bit (byte) access
- W16-bit (word) access
- L32-bit (long) access

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

The core ColdFire® register set is maintained by dBUG. 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 may change these registers, but caution must be exercised as changes may render dBUG inoperable.

A reference to “SP” (stack pointer) actually refers to general purpose address register seven, “A7."

2.2 Operational Procedure

System power-up and initial operation are described in detail in Chapter 1. This information is repeated here for convenience and to prevent possible damage.

2.2.1 System Power-up

- Be sure the power supply is connected properly prior to power-up.
- Make sure the terminal is connected to TERMINAL (P4) connector.
- Turn power on to the board.

Figur 2-1shows the dUBG operational mode.

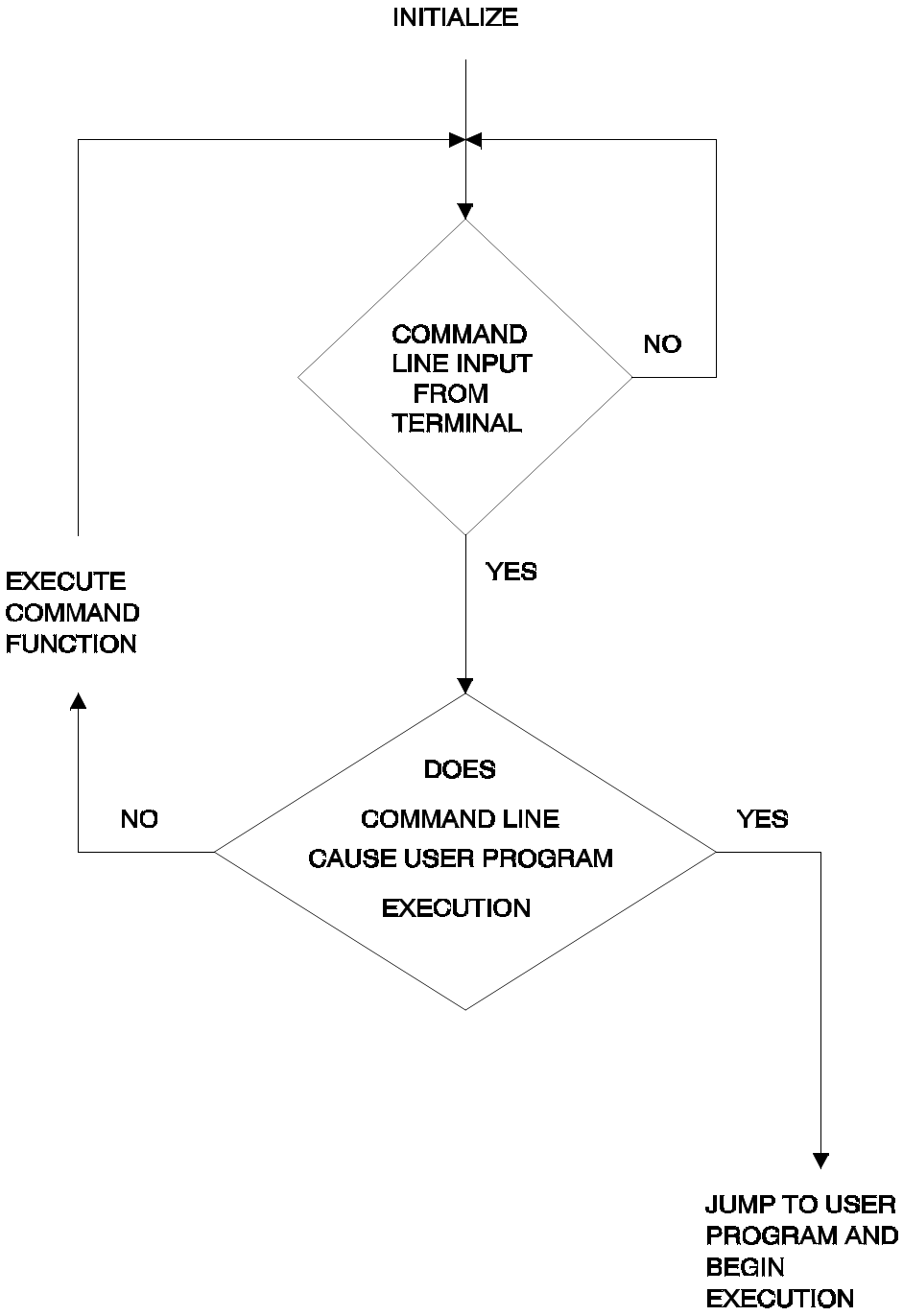


Figure 2-1. Flow Diagram of dBUG Operational Mode.

2.2.2 System Initialization

The act of powering up the board will initialize the system. The processor is reset and dBUG is invoked.

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 SDRAM. To take over an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at 0x00000000, and then points the VBR to 0x00000000.

The Software Watchdog Timer is disabled and internal timers are placed in a stop condition. Interrupt controller registers initialized with unique interrupt level/priority pairs. Please refer to the dBUG source files on theColdFire website (www.motorola.com/coldfire) for the complete initialization code sequence.

After initialization, the terminal will display:

```
Hard Reset
DRAM Size: 4M

Copyright 1995-2001 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5272 EVS Firmware v2e.1a.1a (Build XXX on XXX)

Enter 'help' for help.

dBUG>
```

If you did not get this response check the setup, refer to Section 1.10 System Power-Up and Initial Operation.

Other means can be used to re-initialize the M5272C3 Computer Board firmware. These means are discussed in the following paragraphs.

2.2.2.1 Hard RESET Button.

Hard RESET (S1) is the button. Depressing this button causes all processes to terminate, resets the MCF5272 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 (S2) is the button located next to RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5272) and gives 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 MCF5272 core internal registers are displayed.

The abort function is most appropriate when software is being debugged. The user can interrupt the processor without destroying the present state of the system. This is accomplished by forcing a non-maskable interrupt that will call a dBUG routine that will save the current state of the registers to shadow registers in the monitor for display to the user. The user will be returned to the ROM monitor prompt after exception handling.

2.2.2.3 Software Reset Command.

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

2.3 Command Line Usage

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.

dBUG assumes that an 80x24 ASCII 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 default is19200 bps — a speed commonly available from workstations, personal computers and dedicated terminals.

The command line prompt is: dBUG>

Any dBUG command may be entered from this prompt. dBUG does not allow command lines to exceed 80 characters. Wherever possible, dBUG displays data in 80 columns or less. dBUG echoes each character as it is typed, eliminating the need for any local echo on the terminal side.

The <Backspace> and <Delete> keys are recognized as rub-out keys for correcting typographical mistakes.

Command lines may be recalled using the <Control> U, <Control> D and <Control> R key sequences. <Control> U and <Control> D cycle up and down through previous command lines. <Control> R recalls and executes the last command line.

In general, dBUG is not case-sensitive. Commands may be entered either in uppercase or lowercase, depending upon the user’s equipment and preference. Only symbol names require that the exact case be used.

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

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

2.4 Commands

This section lists the commands that are available with all versions of dBUG. Some board or CPU combinations may use additional commands not listed below.

Table 2-1. dBUG Command Summary

MNEMONIC	SYNTAX	DESCRIPTION
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
BM	bm begin end dest	Block Move
BR	br addr <-r> <-c count> <-t trigger>	Breakpoint
BS	bs <width> begin end data	Block Search
DC	dc value	Data Convert
DI	di<addr>	Disassemble
DL	dl <offset>	Download Serial
DN	dn <-c> <-e> <-i> <-s <-o offset>> <filename>	Download Network
GO	go <addr>	Execute
GT	gt addr	Execute To
HELP	help <command>	Help
IRD	ird <module.register>	Internal Register Display
IRM	irm module.register data	Internal Register Modify
LR	lr<width> addr	Loop Read
LW	lw<width> addr data	Loop Write
MD	md<width> <begin> <end>	Memory Display
MM	mm<width> addr <data>	Memory Modify
MMAP	mmap	Memory Map Display
RD	rd <reg>	Register Display
RM	rm reg data	Register Modify
RESET	reset	Reset
SD	sd	Stack Dump
SET	set <option value>	Set Configurations
SHOW	show <option>	Show Configurations
STEP	step	Step (Over)
SYMBOL	symbol <symb> <-a symb value> <-r symb> <-C|l|s>	Symbol Management
TRACE	trace <num>	Trace (Into)
UPDBUG	updbug	Update dBUG
UPUSER	upuser <bytes>	Update User Flash
VERSION	version	Show Version

ASM Assembler

Usage: ASM <<addr> stmt>

The ASM command is a primitive assembler. The <stmt> is assembled and the resulting code placed at <addr>. This command has an interactive and non-interactive mode of operation.

The value for address <addr> may be an absolute address specified as a hexadecimal value, or a symbol name. The value for stmt must be valid assembler mnemonics for the CPU.

For the interactive mode, the user enters the command and the optional <addr>. If the address is not specified, then the last address is used. The memory contents at the address are disassembled, and the user prompted for the new assembly. If valid, the new assembly is placed into memory, and the address incremented accordingly. If the assembly is not valid, then memory is not modified, and an error message produced. In either case, memory is disassembled and the process repeats.

The user may press the <Enter> or <Return> key to accept the current memory contents and skip to the next instruction, or a enter period to quit the interactive mode.

In the non-interactive mode, the user specifies the address and the assembly statement on the command line. The statement is the assembled, and if valid, placed into memory, otherwise an error message is produced.

Examples:

To place a NOP instruction at address 0x00010000, the command is:

```
asm     10000 nop
```

To interactively assembly memory at address 0x00400000, the command is:

```
asm     400000
```

BC Block Compare

Usage: BC addr1 addr2 length

The BC command compares two contiguous blocks of memory on a byte by byte basis. The first block starts at address addr1 and the second starts at address addr2, both of length bytes.

If the blocks are not identical, the address of the first mismatch is displayed. The value for addresses addr1 and addr2 may be an absolute address specified as a hexadecimal value or a symbol name. The value for length may be a symbol name or a number converted according to the user defined radix (hexadecimal by default).

Example:

To verify that the data starting at 0x20000 and ending at 0x30000 is identical to the data starting at 0x80000, the command is:

```
bc          20000 80000 10000
```

BF Block Fill

Usage: BF<width> begin end data <inc>

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. If no <width> is specified, the default of word sized data is used.

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

The optional value <inc> can be used to increment (or decrement) the data value during the fill.

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 0x00020000 and ending at 0x00040000 with the value 0x1234, the command is:

```
bf      20000 40000 1234
```

To fill a block of memory starting at 0x00020000 and ending at 0x0004000 with a byte value of 0xAB, the command is:

```
bf.b    20000 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
```

To fill a block of memory starting at 0x00020000 and ending at 0x00040000 with data that increments by 2 for each <width>, the command is:

```
bf      20000 40000 0 2
```

BM Block Move

Usage: BM begin end dest

The BM command moves a contiguous block of memory starting at address begin and 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 values for addresses begin, end, and dest may be absolute addresses specified as hexadecimal values, or symbol names. 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
```

NOTE:

Refer to "upuser" command for copying code/data into Flash memory.

BR Breakpoints

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

The BR command inserts or removes breakpoints at address addr. The value for addr may 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, then 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() (symbol _main; see “symbol” command), the command is:

```
br          _main
```

When the target code is executed and the processor reaches main(), control will be 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 attempt to execute the function bench() a third time before returning control back to dBUG.

To remove all breakpoints, the command is:

```
br          -r
```

BS Block Search

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. If no <width> is specified, the default of word sized data is used.

The values for addresses begin and end may be absolute addresses specified as hexadecimal values, or symbol names. The value for data may 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:

bs40000 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, then the address is incremented 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:

bs.l40000 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, then the address is incremented to 0x00040004 and the next 32-bit value is read and compared.

DC Data Conversion

Usage: DC data

The DC command displays the hexadecimal or decimal value data in hexadecimal, binary, and decimal notation.

The value for data may be a symbol name or an absolute value. If an absolute value passed into the DC command is prefixed by ‘0x’, then 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 and binary equivalent of 0x1234, the command is:

```
dc          0x1234
```

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

```
dc          1234
```

DI Disassemble

Usage: DI <addr>

The DI command disassembles target code pointed to by addr. The value for addr may 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 attempts to track the address of the last disassembled opcode. If no address is provided to the DI command, then the DI command uses the address of the last opcode that was disassembled.

The DI command is repeatable.

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
```

DL Download Console

Usage: DL <offset>

The DL command performs an S-record download of data obtained from the console, typically a serial port. The value for offset is converted according to the user-defined radix, normally hexadecimal. Please reference the ColdFire Microprocessor Family Programmer's Reference Manual for details on the S-Record format.

If offset is provided, then the destination address of each S-record is adjusted by offset.

The DL command checks the destination download address for validity. If the destination is an address outside the defined user space, then an error message is displayed and downloading aborted.

If the S-record file contains the entry point address, then 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 add an offset to the destination address of 0x40, the command is:

```
dl          0x40
```

DN Download Network

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

The DN command downloads code from the network. The DN command handle files which are either S-record, COFF, ELF or Image 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, the -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 an 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 filetype will be used. Default filename and filetype parameters are manipulated using the SET and SHOW commands.

The DN command checks the destination download address for validity. If the destination is an address outside the defined user space, then an error message is displayed and downloading 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. The dBUG symbol table is not cleared prior to downloading, so it is the user's 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 filetype with the name "bench.out", the command is:

```
dn bench.out
```

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

```
dn
```

GO Execute

Usage: GO <addr>

The GO command executes target code starting at address addr. The value for addr may 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 only regained when the target code encounters a breakpoint, illegal instruction, trap #15 exception, or other exception which causes control to be handed back to dBUG.

The GO command is repeatable.

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

GT Execute To

Usage: GT addr

The GT command inserts a temporary breakpoint at addr and then executes target code starting at the current program counter. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name.

When the GT command is executed, all breakpoints are inserted into the target code, and the context is switched to the target program. Control is only regained when the target code encounters a breakpoint, illegal instruction, or other exception which causes control to be handed back to dBUG.

Examples:

To execute code up to the C function bench(), the command is:

```
gt _bench
```

IRD Internal Register Display

Usage: IRD <module.register>

This command displays the internal registers of different modules inside the MCF5272. In the command line, module refers to the module name where the register is located and register refers to the specific register to display.

The registers are organized according to the module to which they belong. The available modules on the MCF5272 are CS, DMA0, DMA1, DMA2, DMA3, DRAMC, PP, MBUS, SIM, TIMER1, TIMER2, UART0 and UART1. Refer to the MCF5272 user's manual for more information on these modules and the registers they contain.

Example:

```
ird          sim.rsr
```

IRM Internal Register Modify

Usage: IRM module.register data

This command modifies the contents of the internal registers of different modules inside the MCF5272. In the command line, module refers to the module name where the register is located and register refers to the specific register to modify. The data parameter specifies the new value to be written into the register.

The registers are organized according to the module to which they belong. The available modules on the MCF5272 are CS, DMA0, DMA1, DMA2, DMA3, DRAMC, PP, MBUS, SIM, TIMER1, TIMER2, UART0 and UART1. Refer to the MCF5272 user's manual for more information on these modules and the registers they contain.

Example:

To modify the TMR register of the first Timer module to the value 0x0021, the command is:

```
irm          timer1.tmr    0021
```

HELP Help

Usage: HELP <command>

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of where user code may start is given. If command is provided, then 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
```

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

```
help br
```

LR Loop Read

Usage: LR<width> addr

The LR command continually reads the data at addr until a key is pressed. The optional <width> specifies the size of the data to be read. If no <width> is specified, the command defaults to reading word sized data.

Example:

To continually read the longword data from address 0x20000, the command is:

```
lr.l        20000
```

LW Loop Write

Usage: LW<width> addr data

The LW command continually writes data to addr. The optional width specifies the size of the access to memory. The default access size is a word.

Examples:

To continually write the longword data 0x12345678 to address 0x20000, the command is:

```
lw.l    20000 12345678
```

Note that the following command writes 0x78 into memory:

```
lw.b    20000 12345678
```

MD Memory Display

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

The MD command displays a contiguous block of memory starting at address begin and stopping at address end. The values for addresses begin and end may be absolute addresses specified as hexadecimal values, or symbol names. Width modifies the size of the data that is displayed. If no <width> is specified, the default of word sized data is used.

Memory display starts at the address begin. If no beginning address is provided, the MD command uses the last address that was displayed. If no ending address is provided, then 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:

```
md.l    40000 50000
```

MM Memory Modify

Usage: MM<width> addr <data>

The MM command modifies memory at the address addr. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name. Width specifies the size of the data that is modified. If no <width> is specified, the default of word sized data is used. 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, then the MM command immediately sets the contents of addr to data. If no value for data is provided, then 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, 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 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
```

MMAP Memory Map Display

Usage: mmap

This command displays the memory map information for the M5272C3 evaluation board. The information displayed includes the type of memory, the start and end address of the memory, and the port size of the memory. The display also includes information on how the Chip-selects are used on the board.

Here is an example of the output from this command:

```
      Type            Start         End      Port Size
-------------------------------------------------------
SDRAM             0x00000000   0x003FFFFF   32-bit
Vector Table      0x00000000   0x000003FF   32-bit
USER SPACE        0x00020000   0x003FFFFF   32-bit
MBAR              0x10000000   0x100003FF   32-bit
Internal SRAM     0x20000000   0x20000FFF   32-bit
External SRAM     0x30000000   0x3007FFFF   32-bit
Flash             0xFFE00000   0xFFFFFFFF   16-bit

Chip Selects
----------------
CS0  Flash
CS1  not in use
CS2  Ext SRAM
CS3  not in use
CS4  not in use
CS5  not in use
CS6  not in use
CS7  SDRAM
```

RD Register Display

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.

dBUG preserves the registers by storing a copy of the register set in a buffer. The RD command displays register values from the register buffer.

Examples:

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

```
rd
```

To display only the program counter:

```
rd      pc
```

Here is an example of the output from this command:

```
PC: 00000000 SR: 2000 [t.Sm.000...xnzvc]
An: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01000000
Dn: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

RM Register Modify

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 may be a symbol name, or it is converted according to the user-defined radix, normally hexadecimal.

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 on MC68000 and ColdFire to contain the value 0x1234, the command is:

```
rm          D0 1234
```

RESET Reset the Board and dBUG

Usage: RESET

The RESET command resets the board and dBUG to their initial power-on states.

The RESET command executes the same sequence of code that occurs at power-on. If the RESET command fails to reset the board adequately, cycle the power or press the reset button.

Examples:

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

```
reset
```

SET Set Configurations

Usage: SET <option value>

The SET command allows the setting of user-configurable options within dBUG. With no arguments, SET displays the options and values available. The SHOW command displays the settings in the appropriate format. The standard set of options is listed below.

- baud - This is the baud rate for the first serial port on the board. All communications between dBUG and the user occur using either 9600 or 19200 bps, eight data bits, no parity, and one stop bit, 8N1.
- base - This is the default radix for use in converting a number from its 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 is required to be set to a unique value, usually assigned by your local network administrator.
- server - This is the network IP address of the machine which 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: “srecord”, “coff”, and “elf”.
- mac - This is the ethernet Media Access Control (MAC) address (a.k.a hardware address) for the evaluation board. This should be set to a unique value, and the most significant nibble should always be even.

Examples:

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

```
set         baud 19200
```

NOTE:

See the SHOW command for a display containing the correct formatting of these options.

SHOW Show Configurations

Usage: SHOW <option>

The SHOW command displays the settings of the user-configurable options within dBUG. When no option is provided, SHOW displays all options and values.

Examples:

To display all options and settings, the command is:

```
show
```

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

```
show baud
```

Here is an example of the output from a show command:

```
dBUG> show
  base: 16
  baud: 19200
 server: 192.0.0.1
 client: 192.0.0.2
gateway: 0.0.0.0
netmask: 255.255.255.0
filename: test.srec
filetype: S-Record
  mac: 00:CF:54:07:C3:01
```

STEP Step Over

Usage: STEP

The STEP command can be used to “step over” a subroutine call, rather than tracing every instruction in the subroutine. The ST command sets a temporary breakpoint one instruction beyond the current program counter and then executes the target code.

The STEP command can be used to “step over” BSR and JSR instructions.

The STEP command will work for other instructions as well, but note that if the STEP command is used with an instruction that will not return, i.e. BRA, then the temporary breakpoint may never be encountered and dBUG may never regain control.

Examples:

To pass over a subroutine call, the command is:

```
step
```

SYMBOL Symbol Name Management

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, then 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 only use the first 31 characters. Symbol names are case-sensitive.

Symbols can also be added to the symbol table via in-line assembly labels and ethernet downloads of ELF formatted files.

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
```

TRACE Trace Into

Usage: TRACE <num>

The TRACE command allows single-instruction execution. If num is provided, then num instructions are executed before control is handed back 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.

This command is repeatable.

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
```

UPDBUG Update dBUG

Usage: updbug

The updbug command is used to update the dBUG image in Flash. When updates to the M5272C3 dBUG are available, the updated image is downloaded to address 0x00020000. The new image is placed into Flash using the UPDBUG command. The user is prompted for verification before performing the operation. Use this command with extreme caution, as any error can render dBUG useless!

UPUSER Update User Flash

Usage: UPUSER <bytes>

The UPUSER command places user code and data into space allocated for the user in Flash. The optional parameter bytes specifies the number of bytes to copy into the user portion of Flash.If the bytes parameter is omitted, then this command writes to the entire user space. There are seven sectors of 256K each available as user space. Users access this memory starting at address 0xFFE40000.

Examples:

To program all 7 sectors of user Flash, the command is:

```
upuser
```

To program only 1000 bytes into user Flash, the command is:

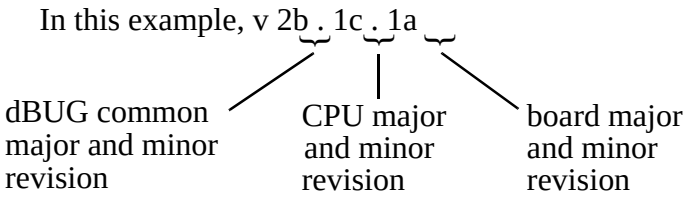
```
upuser 1000
```

VERSION Display dBUG Version

Usage: VERSION

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

The version number is separated by a decimal, for example, “v 2b.1c.1a”.



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. This function can be called by the user program to utilize various routines within the dBUG, to perform a special task, and to return control to the dBUG. 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 1
        /* LINK a6,#0 -- produced by C compiler */
        asm (" move.l8(a6),d1");                    /* put 'ch'into d1 */
        asm (" move.l#0x0013,d0");  /* select the function */
        asm (" trap #15");                          /* 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.l4(sp),d1");                   /* put 'ch'into d1 */
        asm (" move.l#0x0013,d0");  /* select the function */
        asm (" trap #15");                          /* 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.ld1,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 */
}
```

Chapter 3
Hardware Description and Reconfiguration

This chapter provides a functional description of the M5272C3 board hardware. With the description given here and the schematic diagram in Appendix C, the user can gain a good understanding of the board's design. In this manual, an active low signal is indicated by a "-" preceeding the signal name in the text and a bar over the signal name in the schematics.

3.1 The Processor and Support Logic

This part of the chapter discusses the CPU and general support logic on the M5272C3 board.

3.1.1 Processor

The microprocessor used on the M5272C3 is the highly integrated Motorola ColdFire® MCF5272, 32-bit processor. The MCF5272 implements a ColdFire Version 2 core with 1-KByte instruction cache, two UART channels, four timers, 4-KBytes of SRAM, a QSPI (Queued Serial Peripheral Interface) module, three 16-bit wide parallel I/O ports (which are multiplexed with other signals) and the system integration module (SIM). All of the core processor registers are 32 bits wide except for the Status Register (SR) which is 16 bits wide. This processor communicates with external devices over a 32-bit wide data bus, D[31:0] with support for 8 and 16-bit ports. The width of this data bus is configurable for 16 or 32 bits at Power on Reset (POR) using the WSEL pin. When programmed for a 16-bit external databus width, the signals D[15:0] become GPIO Port C. The chip can address 16-MBytes of memory space using a 23-bit wide address bus and internal chip-select logic. All the processor's signals are available through the expansion connectors (J2 and J3). Refer to section 3.3.1 for their pin assignments.

The MCF5272 processor has the capabiliy to support both an IEEE JTAG-compatible port and a BDM port. These ports are multiplexed on to the same pins and can be used with third party tools to allow the user to download code to the board. The board is configured to boot up in the normal/BDM mode of operation. The BDM signals are available at port J4. The processor also has the logic to generate up to eight (8) chip selects, -CS0 to -CS7, and support for 1 bank of ADRAM (not included on the evaluation board) or 1 bank of SDRAM (included on the evaluation board 4-Mbytes in total configured as 1Mx32). The -CS7 signal is used to provide selection and control of this bank of SDRAM.

3.1.2 Reset Logic

The reset logic provides system initialisation. Reset occurs during power-on or via assertion of the signal -RSTI which causes the MCF5272 to reset. Reset is also triggered by the reset switch (S1) which resets the entire processor/system.

A hard reset and voltage sense controller (U17) is used to produce an active low power-on RESET signal. The reset switch S1 is fed into U17 which generates the signal which is fed to the MCF5272 reset, -RSTI. The -RSTI signal is an open collector signal and so can be wire OR'ed with other reset signals from additional peripherals.

dBUG configures the MCF5272 microprocessor internal resources during initialization. The instruction cache is invalidated and disabled. The Vector Base Register, VBR, contains an address which initially points to the Flash memory. The contents of the exception table are written to address $00000000 in the SDRAM. The Software Watchdog Timer is disabled, Bus Monitor enabled and internal timers are placed in a stop condition. The interrupt controller registers are initialised with unique interrupt level/priority pairs. A memory map for the entire board can be seen in Table 3-1.

3.1.3 HIZ Signal

The assertion of the -HIZ signal during reset forces all output drivers to a high-impedance state. On the M5272C3 board the high impedance signal is pulled to +3.3V via a 4.7K pull-up resistor, ensuring that the output drivers will not be in a high-impedance state during reset. -HIZ is also available to the user on connector J2.

3.1.4 Clock Circuitry

The M5272C3 board uses a 66MHz oscillator (U15 on schematics) to provide the clock to the clock driver chip (U16). The clock driver provides buffered clocks for the MCF5272 processor (U5), the FSRAM (U9) (not fitted) and the PAL (U20). In addition to the 66MHz oscillator, there is also a 25MHz oscillator (U6) which feeds the Ethernet chip (U7) and a USB 48MHz oscillator (U25) which feeds the USBEXTCLK signal on MCF5272 to clock the USB module therein.

3.1.5 Watchdog Timer

The duration of the Watchdog is selected by the REF[15:1] bits in the Watchdog Reset Reference Register (WRRR). The dBUG monitor initialises this register with the value 0xFFFE, which provides the maximum time-out period but dBUG does **NOT** enable the watchdog timer.

3.1.6 Interrupt Sources

The ColdFire® family of processors can receive seven levels of interrupt priorities. When

the processor receives an interrupt which has a higher priority than the current interrupt mask (in the 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 interrupt is not capable of providing a vector, it's interrupt should be set up as an autovector interrupt which directs the processor to a predefined entry in the exception table (refer to the MCF5272 User's Manual).

The processor goes to an exception routine via the exception table. This table is stored in the Flash EEPROM. The address of the table location is stored in the VBR. The dBUG ROM monitor writes a copy of the exception table into the RAM starting at $00000000. To set an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at $00000000 and then points the VBR to $00000000.

The MCF5272 microprocessor has six external interrupt request lines -INT[6:1], three of which (-INT[6:4]) are multiplexed with other functions. The interrupt controller is capable of providing up to 32 interrupt sources. These sources are :-

- External interrupt signals -INT[6:1]
- Software watchdog timer
- Four general purpose timers
- Two UART's
- Ethernet controller
- PLI (Physical Layer Interface) time division multiplexer interface controller
- Memory to memory DMA
- QSPI module
- USB module

All external interrupt inputs are edge sensitive. The active edge is programmable. An interrupt request must be held valid for at least three consecutive CPU clock cycles, to be considered a valid input. Each interrupt input can have it's priority programmed by setting the xIPL[2:0] bits in the Interrupt Control Registers.

NOTE:

No interrupt sources should have the same level and priority as another. Programming two interrupt sources with the same level and priority can result in undefined operation.

The M5272C3 hardware uses -INT6 to support the ABORT function using the ABORT switch S2. This switch is used to force an interrupt (level 6, priority 3**) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT). Since the ABORT switch is not capable of generating a vector in

response to level six interrupt acknowledge from the processor, the dBUG programs this interrupt request for autovector mode.

The -INT1 line of the MCF5272 processor is connected to PLI connector pin B33 (J6).

The -INT3 line of the MCF5272 processor is connected to PLI connector pin A34 (J5).

Refer to MCF5272 User's Manual for more information about the interrupt controller.

3.1.7 Internal SRAM

The MCF5272 processor has 4-KBtyes of internal memory which may be programmed as data or instruction memory. This memory is mapped to 0x20000000** and configured as data space but is not used by the dBUG monitor except during system initialisation. After system initialisation is complete the internal memory is available to the user. The memory is relocatable to any 4-KByte boundary.

3.1.8 The MCF5272 Registers and Memory Map

The memory and I/O resources of the M5272C3 hardware are divided into two groups, MCF5272 internal and external resources. All the I/O registers are memory mapped.

The MCF5272 processor has built in logic and up to eight chip-select pins (-CS[7:0]) which are used to enable external memory and I/O devices. In addition there are -RAS and -CAS lines available for controlling DRAMs. There are registers to specify the address range, type of access and the method of -TA generation for each chip-select and the -RAS pin. These registers are programmed by the dBUG monitor to map the external memory and I/O devices.

The M5272C3 uses the following signals to select external peripherals :-

-CS0 to enable the Flash ROM (refer to Section 3.1.13)

-RAS0, -CAS0 and -CS7 to enable the SDRAM (refer to Section 3.1.12)

-CS2 for the FSRAM (not populated)

-CS5 & -CS6 for the PLI I/O space (refer to section 3.2.8)

The chip select mechanism of the MCF5272 processor allows the memory mapping to be defined based on the required memory space (User/Supervisor, Program/Data spaces).

All of the MCF5272 internal registers, configuration registers, parallel I/O port registers, UART registers and system control registers are mapped by the MBAR register at any 1-KByte boundary. The MBAR register is mapped to 0x10000000** by the dBUG monitor. For a complete map of these registers refer to the MCF5272 User's Manual.

The M5272C3 board has 4-MBytes of SDRAM installed. Refer to Section 3.1.12 for a discussion of the SDRAM on the board. The dBUG ROM monitor is programmed in one

AMD Am29PL160BC-XX Flash ROM which occupies 2-MBytes of the address space. The first 256-KBytes, i.e the first sector, are used by ROM Monitor and the remainder is left for the user. Refer to section 3.1.13.

Table 3-1 shows the M5272C3 memory map.

Table 3-1. The M5272C3 Memory Map

Address Range	Signal and Device	Memory Access Time
$00000000-$00020000	SDRAM space for dBug ROM monitor use	refer to manufacturer spec
$00020000-$003FFFFF	SDRAM space	refer to manufacturer spec
$10000000-$100003FF	System Integration Module (SIM) registers	internal access
$20000000-$20000FFF	SRAM	internal access (1 clock)
$30000000-$3007FFFF[1]	-CS2, External FSRAM (not fitted)	2-1-1-1
$50000000-$5001FFFF	-CS5, PLI Connector I/O J5	max. number of wait states is 30
$60000000-$6001FFFF	-CS6, PLI Connector I/O J6	max. number of wait states is 30
$FFE00000-$FFFFFFFF	-CS0, 2M Flash ROM	8-7-7-7

[1] Not installed. SRAM footprint accepts Motorola's MCM69F737TQ chip and any other SRAM with the same electrical specifications and package, alternatives are shown on the schematics Appendix **.

* dBUG Monitor does not program the -CS5, -CS6 chip selects (addresses shown are for illustrative purposes only)

*Connectors J5 and J6 are not populated on the evaluation board

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

3.1.9 Reset Vector Mapping

After reset, the processor attempts to read the initial stack pointer and program counter values from locations $00000000 & $00000007 (the first eight bytes of memory space). This requires the board to have a non-volatile memory device in this range with the correct information stored in it. In some systems, however, it is preferred to have RAM starting at address $00000000. The MCF5272 processor chip-select zero (-CS0) responds to any accesses after reset until the CSMR0 is written. Since -CS0 (the global chip select) is connected to the Flash ROM (U8), the Flash ROM initially appears at address $00000000 which provides the initial stack pointer and program counter (the first 8 bytes of the Flash ROM). The initialisation routine then programs the chip-select logic, locates the Flash ROM to start at $FFE00000 and configures the rest of the internal and external peripherals.

3.1.10 TA Generation

The processor starts a bus cycle by asserting -CSx with other control signals. The processor then waits for a transfer acknowledgment (-TA) either from within (Auto acknowledge -

AA mode) or from the externally addressed device before it can complete the bus cycle. -TA is used to indicate the completion of the bus cycle. It also allows devices with different access times to communicate with the processor properly (i.e. asynchronously). The MCF5272 processor, as part of the chip-select logic, has a built-in mechanism to generate -TA for all external devices which do not have the capability to generate this signal. The Flash ROM and FSRAM can not generate -TA.. The chip-select logic is programmed by the dBUG ROM Monitor to generate -TA internally after a pre-programmed number of wait states. In order to support future expansion of the M5272C3 board, the -TA input of the processor is also connected to the Processor Expansion Bus (J3 pin 64). This allows the expansion boards to assert this line to indicate their -TA signal to the processor. On the expansion boards this signal should be generated through an open collector buffer with no pull-up resistor; a pull-up resistor is included on this board. All -TA signals from expansion boards should be connected to this line.

3.1.11 Wait State Generator

The Flash ROM and SDRAM on the board may require some adjustments to the cycle time of the processor to make them compatible with the processor's external bus speed. To extend the CPU bus cycles for the slower devices, the chip-select logic of the MCF5272 processor can be programmed to generate an internal -TA after a given number of wait states. Refer to Table 3-1 for information about the address space of the memory and refer to the manufacturers specification for wait state requirements of the SDRAM and Flash ROM.

3.1.12 SDRAM

The M5272C3 has two 16-MBit devices on the board, in a 32-bit wide data bus configuration. The MCF5272 processor supports one bank of SDRAM, which on this board is represented by SDRAM devices U21 & U22. These are connected to the MCF5272 to provide 1Mx32 of memory.

3.1.13 Flash ROM

There is one 2-MByte Flash ROM on the M5272C3, U8.

The board is shipped with one AMD Am29PL160C, 2-MByte Flash ROM. The first 256-Kbytes of the Flash contains the ROM Monitor firmware dBUG. The remaining Flash memory is available to the user via use of jumper 13.

The MCF5272 chip-select logic can be programmed to generate the -TA for -CS0 signal after a certain number of wait states (i.e. auto acknowledge mode). The dBUG monitor programs this parameter to be six wait-states.

3.1.14 JP13 Jumper and the User's Program

Jumper 13 allows users to test code from boot/POR without having to overwrite the ROM Monitor.

When the jumper is set between pins 1 and 2, the behavior of the system is normal, dBUG boots and then runs from 0xFFE00000. When the jumper is set between pins 2 and 3, the board boots from the second half of the Flash (0xFFF00000).

Procedure:

1. Compile and link as though the code was to be placed at the base of the flash, but setup so that it will download to the SDRAM starting at address 0xE0000. The user should refer to their compiler documentation for this, since it will depend upon the compiler used.
2. Set up the jumper (JP13) for Normal operation, pin1 connected to pin 2.
3. Download to SDRAM (If using serial or ethernet, start the ROM Monitor first. If using BDM via a wiggler cable, download first, then start ROM Monitor by pointing the PC to 0x7FE00400 and run.)
4. In the ROM Monitor, execute the 'upuser' command.
5. Move jumper (JP13) to pin 2 connected to pin 3 and push the reset button (S1). User code should now be running.

3.2 Serial Communication Channels

The M5272C3 offers a number of serial communications. They are discussed in this section.

3.2.1 MCF5272 UARTs

The MCF5272 device has two built in UARTs, each with its own software programmable baud rate generators. One channel is the ROM Monitor to Terminal output and the other is available to the user. The ROM Monitor programs the interrupt level for UART0 to Level 3, priority 2 and autovector mode of operation**. The interrupt level for UART1 is programmed to Level 3, priority 1 and autovector mode of operation**. The signals from these channels are available on expansion connector J3. The signals of UART0 and UART1 are also passed through the RS-232 driver/receivers U23 & U24 and are available on DB-9 connectors P3 and P4. Refer to the MCF5272 User's Manual for programming the UART's and their register maps.

3.2.2 QSPI Module

The QSPI (Queued Serial Peripheral Interface) module provides a serial peripheral interface with queued transfer capability. It will support up to 16 stacked transfers at one

time, minimising CPU intervention between transfers. Transfer RAMs in the QSPI are indirectly accessible using address and data registers.

Functionality it is very similar, but not identical, to the QSPI portion of the QSM (Queued Serial Module) implemented in the MC68332.

- Programmable queue to support up to 16 transfers without user intervention
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chip-select lines for control of up to 15 devices
- Baud rates from 129.4-Kbps to 33-Mbps at 66MHz.
- Programmable delays before and after transfers
- Programmable clock phase and polarity
- Supports wrap-around mode for continuous transfers

Please see the MCF5272 Users Manual for more detail. The QSPI signals from the MCF5272 device are brought out to expansion connector J3. Some of these signals are multiplexed with other functions.

3.2.3 PWM (Pulse Width Modulation) Module

This section describes the Pulse Width Modulation unit for use in control applications. With a suitable low-pass filter, the PWM can be used as a simple digital to analog converter. This module generates a synchronous series of pulses. The duty cycle of the pulses is under software control. A summary of the main features include :-

- Double-buffered width register
- Variable-divide prescaler
- Three independent PWM modules
- Byte-wide width register-provides programmable control of duty cycle.

The PWM is a simple free-running counter implemented along with a width register and a comparator such that the output is cleared when the counter value exceeds the value of the width register. When the counter “wraps around,” the counter value is less than or equal to the value of the width register, and the output is set high.

The width register is double-buffered so that a new value can be loaded for the next cycle without disturbing the current cycle. At the beginning of each period the contents of the width buffer register are loaded into the width register. The width register feeds the comparator for the purpose of comparison during the next cycle. The prescaler contains a variable divider that can divide the incoming clock by certain values between 1 and 32768. The PWM signals are brought to expansion connector J3, please see the MCF5272 User manual for more detail.

3.2.4 Parallel I/O Port

The MCF5272 device provides up to 48 general-purpose I/O signals depending on device configuration. Eight general-purpose I/O (GPIO) pins will be available at all times. The functions of all I/O pins are individually programmable, since they are multiplexed with other pin functions. All general-purpose I/O pins can be individually selected as input or output pins. After reset, all software configurable multi-function GPIO pins default to general purpose input pins. At the same time, all multifunction pins that are not shared with a GPIO pin default to high impedance. Internal pullup resistors avoid unknown read values in order to reduce power consumption. They remain active until the corresponding port direction registers are programmed.

The general-purpose I/O signals are configured as three ports, each having up to sixteen signals. These three general-purpose I/O ports are shared with other signals as follows :-

- Port A bits [6:0] are multiplexed with the signals required to interface to an external USB transceiver.
- PA7 is multiplexed with SPI_CS3 and DOUT3.
- Port A bits [15:8] are multiplexed with the pins of PLI TDM Ports 0 and 1 and UART1.
- Port B bits [7:0] are multiplexed with the UART0 signals and the bus control signal $\overline{TA}$.
- Port B bits [15:8] are multiplexed with the Ethernet controller signals.
- Port C bits [15:0] are multiplexed with data bus signals D15–D0 and are only available when the device is configured for 16-bit data bus mode using the WSEL signal.

Control registers are provided for each port to select the function (GPIO or peripheral pin) assigned to each pin individually. Pins can have from 1 to 4 functions including GPIO. There is no configuration register for GPIO port C because its pins are configured by the WSEL signal during device reset.

An additional port, Port D, has only a control register which is used to configure the pins that are not multiplexed with any GPIO signals. Please see the MCF5272 User's manual for more detail. All of these signals are brought out to expansion connectors J2 & J3.

3.2.5 Ethernet Module

The MCF5272 device performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The MCF5272 Ethernet Controller requires an external interface adaptor and transceiver function to complete the interface to the ethernet media. The MCF5272 Ethernet module also features an integrated fast (100baseT) Ethernet media access controller (MAC).

The Fast Ethernet controller (FEC) incorporates the following features :-

- Full compliance with the IEEE 802.3 standard
- Support for three different physical interfaces:
  — 100 Mbps 802.3 media independent interface (MII)
  — 10 Mbps 802.3 MII
  — 10 Mbps seven-wire interface
- Half-duplex 100-Mbps operation at system clock frequency Š 50 MHz
- 448 bytes total on-chip transmit and receive FIFO memory to support a range of bus latencies
  Note: the total FIFO size is 448 bytes. It is not intended to hold entire frames but only to compensate for external bus latency. The FIFO can be partitioned on any 32-bit boundary between receive and transmit, for example, 32 x 56 receive and 32 x 56 transmit.
- Retransmission from transmit FIFO following a collision, no processor bus used
- Automatic internal flushing of the receive FIFO for runts and collisions with no processor bus use

For more details see the MCF5272 Users manual, this module's signals are also brought to expansion connector J3.

The on board ROM MONITOR is programmed to allow a user to download files from a network to memory in different formats. The current compiler formats supported are S-Record, COFF, ELF or Image.

3.2.6 USB (Universal Serial Bus) Module

The MCF5272 processor includes the following features :-

- Supports full-speed 12-Mbps USB devices and low-speed 1.5-Mbps devices
- Full compliance with the *Universal Serial Bus Specification, Revision 1.1* with the addition of an external USB transceiver
- Automatic processing of USB standard device requests: CLEAR_FEATURE, GET_CONFIGURATION, GET_DESCRIPTOR, GET_INTERFACE, GET_STATUS, SET_ADDRESS, SET_CONFIGURATION, SET_FEATURE, and SET_INTERFACE
- Supports either internal or external USB transceiver
- Programmable 512-byte receive and 512-byte transmit FIFO buffers
- USB device controller with protocol control and administration for up to eight endpoints, 16 interfaces, and 16 configurations
- Programmable endpoint types with support for up to eight control, interrupt, bulk, or isochronous endpoints
- Independent interrupts for each endpoint

- Supports remote wakeup
- Detects start-of-frame and missed start-of-frame for isochronous endpoint synchronization
- Notification of start-of-frame, reset, suspend, and resume events
- Operates in “Device” mode only

All the USB signals are brought out to USB connector JR1 and expansion connector J3. For further details please refer to the MCF5272 User’s manual.

3.2.7 ROM Module

- 16-Kbyte ROM, organized as 4K x 32 bits
- Contains tabular data for soft HDLC, DTMF detection & tone generation
- Single-cycle access
- Physically located on ColdFire core's high-speed local bus
- Byte, word, longword address capabilities
- Programmable memory mapping

The ROM module contains tabular data that the ColdFire core can access in a single cycle. The ROM can be located on any 16-Kbyte address boundary in the 4-Gbyte address space.

Depending on configuration information, instruction fetches can be sent simultaneously to the instruction cache and the ROM block. If the fetch address is mapped into the region defined by the ROM, the ROM provides the data back to the processor and any instruction cache data is discarded. Accesses from the on-chip ROM are not cached.

3.2.8 PLI (Physical Layer Interface) Module

The physical layer interface (PLI) allows the MCF5272 device to connect at a physical level with external CODECs, TDM controllers and other peripheral devices which utilize either the general circuit interface (GCI), or interchip digital link (IDL), physical layer protocols. These PLI slots are unpopulated on the M5272C3 board and have been built on to the PCB to allow customers to add their own POTs (Plain Old Telephone System) and ISDN (Integrated Digital Subscriber Network) hardware.

The MCF5272 PLI has four ports, port[3:0], connected through three physical interfaces, numbered 0, 1, 2 and 3. A port can service, read or write any 2B + D channel. Port 0 connects through interface 0, and Ports 1 and 2 both connect through interface 1. Port 3 can use either interface 1 or 3. In the case of interface 1, which connects multiple ports, delayed frame sync generators are provided for each port. These generators delay the active slots within a port with respect to a reference clock.

For more details on the PLI port configurations and programming these ports please refer to the MCF5272 User’s manual. All these siganls appear on expansion connector J3 as well

as PLI connectors J5 & J6.

3.3 Connectors and Expansion Bus

There are 2 expansion connectors on the M5272C3 (J2 and J3) which are used to connect the board to external I/O devices and/or expansion boards.

3.3.1 Expansion Connectors - J2 and J3

Table 3-2 shows pin assignments for the J2 connector.

Table 3-2. J2 Connector Pin Assignment

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	GND	2	GND	61	D17	62	D18
3	CPU_Ext_CLK	4	-RESET	63	D19	64	GND
5	GND	6	-DRESETEN	65	GND	66	D20
7	-BS0	8	-BS1	67	D21	68	D22
9	-BS2	10	-BS3	69	D23	70	GND
11	-OE	12	GND	71	GND	72	D24
13	-CS0	14	-CS1	73	D25	74	D26
15	-CS2	16	-CS3	75	D27	76	GND
17	-CS4	18	-CS5	77	GND	78	D28
19	-CS6	20	GND	79	D29	80	D30
21	DRQ	22	DACK/HIZ	81	D31	82	GND
23	TC/BYPASS	24	GND	83	GND	84	A0
25	A10_PRECHG	26	-WE	85	A1	86	A2
27	SDBA0	28	SDBA1	87	A3	88	GND
29	-RAS0	30	-CAS0	89	GND	90	A4
31	SDCLK	32	-SDWE	91	A5	92	A6
33	SDCLKE	34	-CS7	93	A7	94	GND
35	GND	36	D0	95	GND	96	A8
37	D1	38	D2	97	A9	98	A10
39	D3	40	GND	99	A11	100	GND
41	GND	42	D4	101	GND	102	A12
43	D5	44	D6	103	A13	104	A14
45	D7	46	GND	105	A15	106	GND
47	GND	48	D8	107	GND	108	A16

Table 3-2. J2 Connector Pin Assignment (Continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
49	D9	50	D10	109	A17	110	A18
51	D11	52	GND	111	A19	112	GND
53	GND	54	D12	113	GND	114	A20
55	D13	56	D14	115	A21	116	A22
57	D15	58	GND	117	+3.3V	118	+3.3V
59	GND	60	D16	119	+3.3V	120	+3.3V

Table 3-3 shows the pin assignments of the J3 connector.

Table 3-3. J3 Connector pin assignment

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	GND	2	GND	61	GND	62	GND
3	DS0	4	DSI	63	UXBEXTCLK	64	PB5/-TA
5	GND	6	-BKPT	65	GND	66	GND
7	CPU_CLK	8	DSCLK	67	PB6	68	USBLineH
9	GND	10	-DTEA	69	TIN1	70	USBLineL
11	-TEST	12	MTMOD	71	TOUT1	72	GND
13	GND	14	GND	73	-INT1	74	-INT2
15	DDATA0	16	DDATA1	75	-INT3	76	WSEL
17	DDATA2	18	DDATA3	77	+3.3V	78	QSPI_DIN
19	PST0	20	PST1	79	BUSW1	80	BUSW0
21	PST2	22	PST3	81	+3.3V	82	PWMOUT1
23	GND	24	GND	83	PWMOUT2/TOUT2	84	PWMOUT3/TIN2
25	PA0/USB_TP	26	PA1/USB_RP	85	+3.3V	86	+3.3V
27	PA3/USB_TN	28	PA2/USB_RN	87	EXTCLK	88	ETXD0
29	PA4/USB_SUSP	30	PA5/USB_TXEN	89	ECOL	90	ERXDV
31	PA6/USBRXD	32	-RST0	91	ERXCLK	92	ERXD0
33	GND	34	GND	93	ETXEN	94	ETXD3
35	FSC0/FSR0	36	DGNT0	95	GND	96	GND
37	DCL0/USRT2CLK	38	DIN0/USRT2RXD	97	ETXD2	98	ETXD1

Table 3-3. J3 Connector pin assignment (Continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
39	USRT2CTS/SPI_CS2	40	USRT2RTS/-INT5	99	-CS6_RD	100	-CS5_RD
41	DOUT0/USRT2TXD	42	DREQ0	101	ERXD3	102	ERXD2
43	SPI_CS1	44	GND	103	GND	104	GND
45	GND	46	DFSC2	105	ERXD1	106	ERXERR
47	DFSC3	48	FSC1-FSR1/DFCS1	107	-CS6_WR	108	-CS5_WR
49	DCL1/GEN_DCL_OUT	50	DREQ1	109	EMDC	110	EMDIO
51	DGNT1/-INT6	52	DOUT1	111	GND	112	GND
53	DIN1	54	SPI-CS3/DOUT3	113	ETXERR	114	ECRS
55	DIN3/-INT4	56	USRT1TXD	115	+5.0V	116	+5.0V
57	USRT1RXD	58	USRT1CTS	117	+5.0V	118	+5.0V
59	USRT1RTS	60	USRT1CLK	119	+3.3V	120	+3.3V

.

3.3.2 The Debug Connector J4

The MCF5272 processor has a Background Debug Mode (BDM) port, which supports Real-Time Trace Support and Real-Time Debug. The signals which are neccessary for debug are available at connector J4. Figure 3-1 shows the J4 Connector pin assignment.

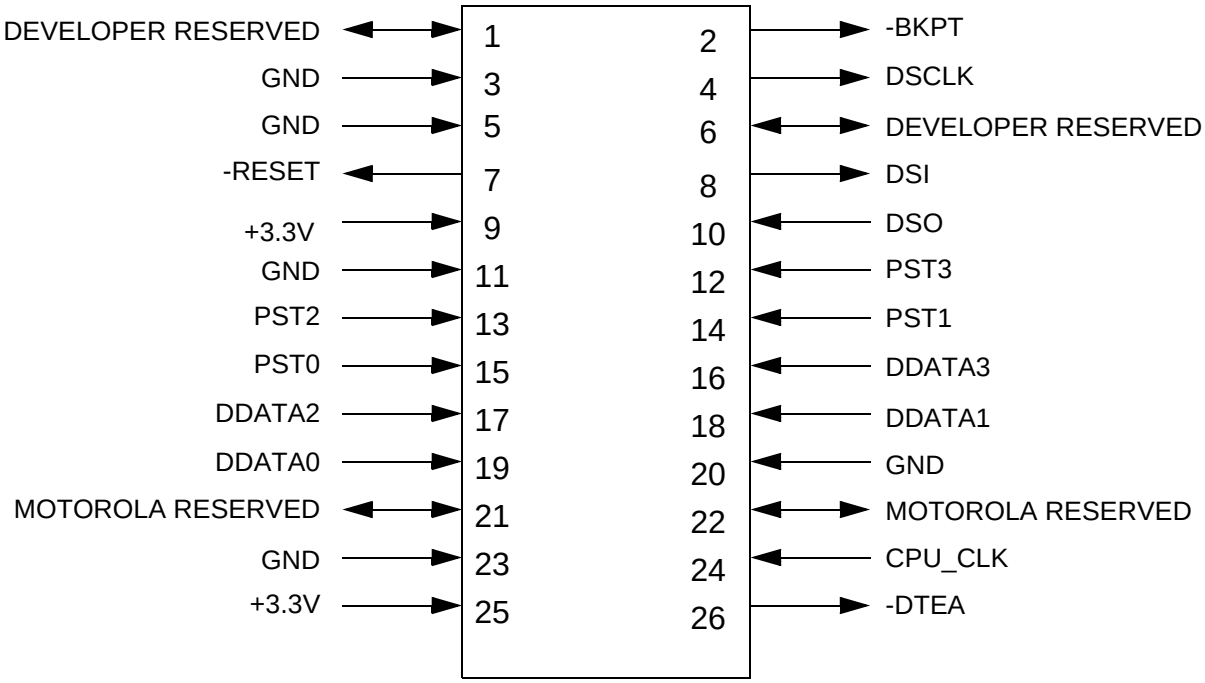


Figure 3-1. The J4 Connector pin assignment

Appendix A
Configuring dBUG for Network Downloads

The dBUG module has the ability to perform downloads over an Ethernet network using the Trivial File Transfer Protocol, TFTP. Prior to using this feature, several parameters are required for network downloads to occur. The information that is required and the steps for configuring dBUG are described below.

A.1 Required Network Parameters

For performing network downloads, dBUG needs 6 parameters; 4 are network-related, and 2 are download-related. The parameters are listed below, with the dBUG designation following in parenthesis.

All computers connected to an Ethernet network running the IP protocol need 3 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), and
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 (filetype of S-record, COFF, ELF, or Image).

Your local system administrator can assign a unique IP address for the board, and also provide you 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, the dBUG Rom Monitor must be configured. 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 mac <addr>

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 MAC address is chosen arbitrarily and is unique. The commands to dBUG are:

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 mac 00:CF:52:72:C3:01

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.

Most, if not all, TFTP servers will only permit access to files starting at a particular sub-directory. (This is a security feature which 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 used by the TFTP server.

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

set filename <filename>

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

set filetype <srecord|coff|elf|image>

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

Change the default filename and filetype with the commands:

```
set filename a.out
set filetype coff
```

Finally, perform the network download with the ‘dn’ command. The network download process uses the configured IP addresses and the default filename and filetype 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. This is accomplished via the ‘show ’command.

Using an IP address already assigned to another machine will cause dBUG network download to fail, and probably 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’ which can be used to connect to the TFTP server as well. Is the default TFTP root directory present and readable?

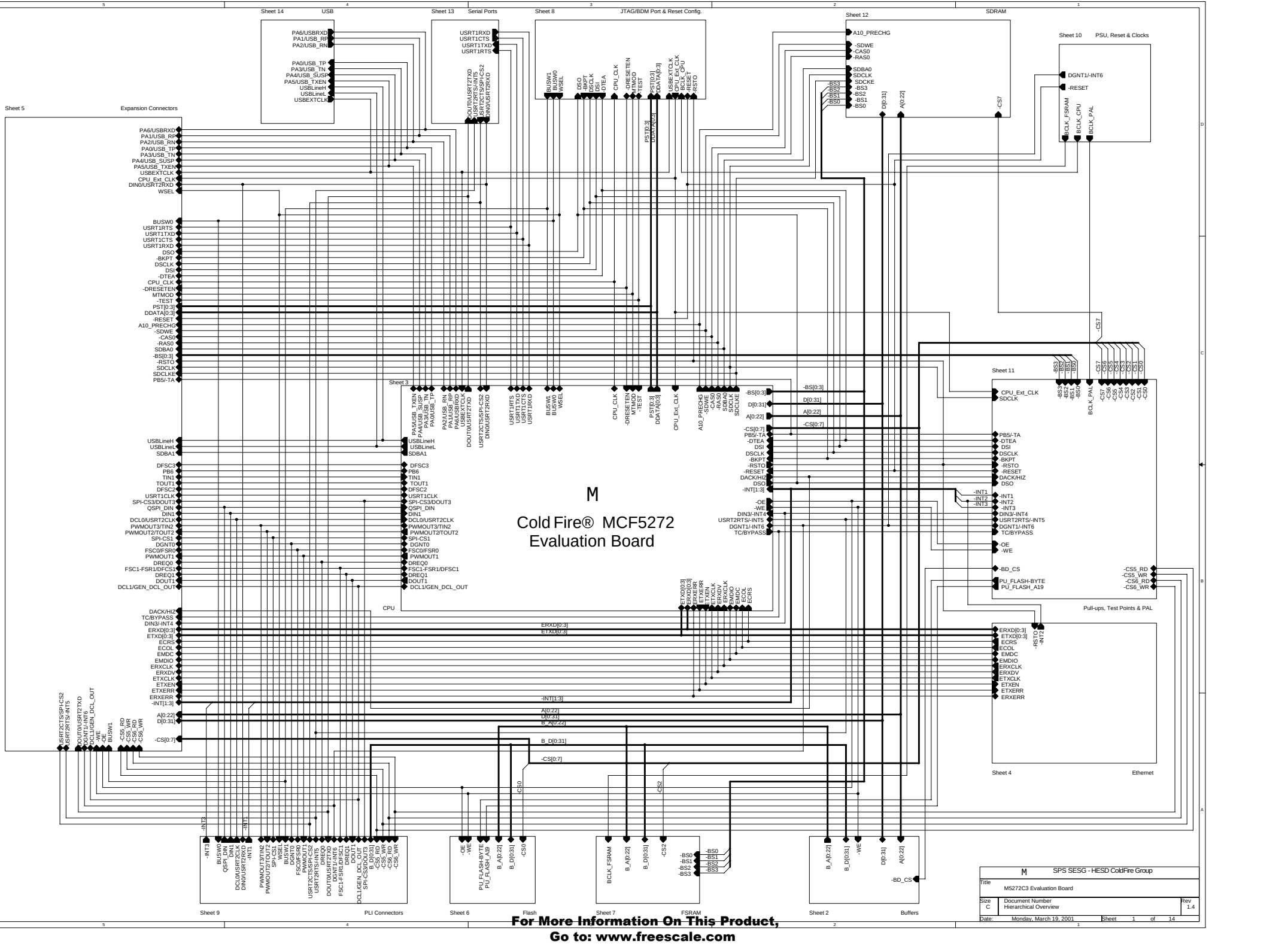
If ‘ICMP_DESTINATION_UNREACHABLE’ or similar ICMP message appears, then a serious error has occurred. Reset the board, and wait one minute for the TFTP server to time out and terminate any open connections. Verify that the IP addresses for the server and gateway are correct. Also verify that a TFTP server is running on the server.

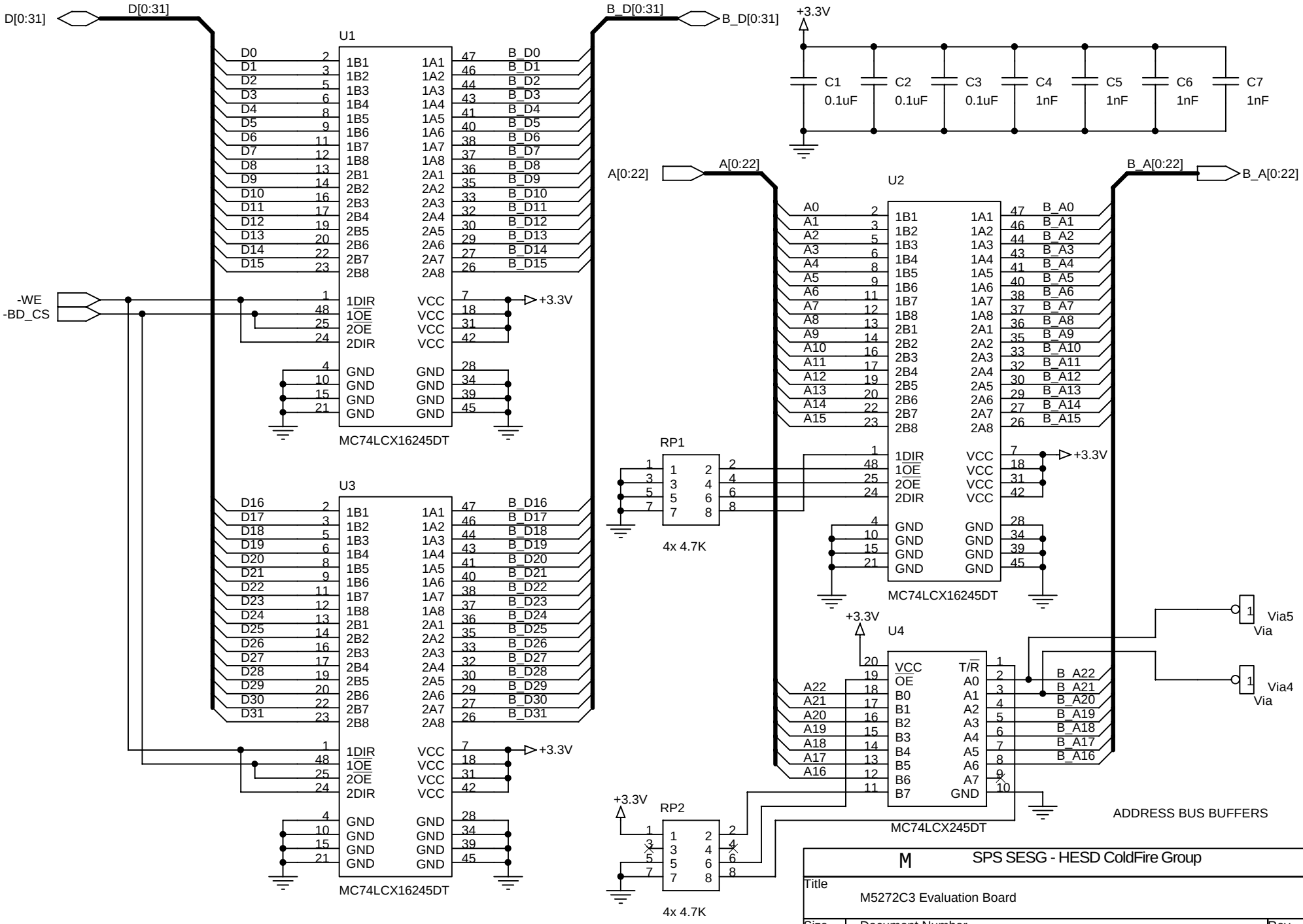
Appendix B
PAL Equations

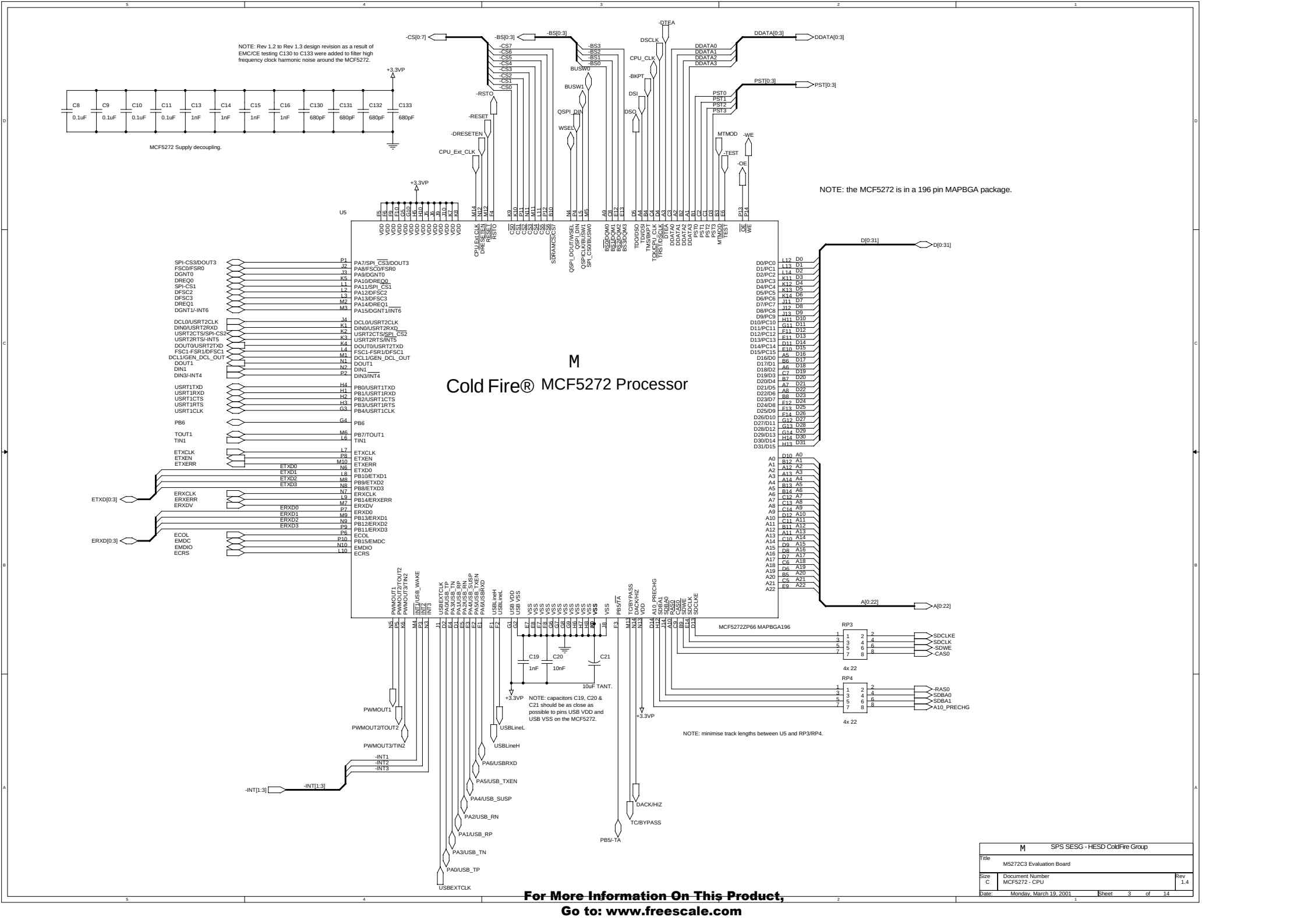
The PAL equations listed below provide some simple decode logic for the address and data buffers (sheet 2 of the schematics) and some memory mapped I/O (sheet 9 of the schematics). The first equation which defines generation of the BD_CS signal is a simple OR'ing of chip select signals -CS0 (Flash), -CS2 (FSRAM) and -CS5 & -CS6 (PLI connectors). This signal is then used to control the output enable -OE signal of buffers U1 and U3 that drive the 32-bit wide data bus. The next four equations define read and write access to some memory mapped I/O (octal D-type flip flops). These equations generate 8 extra input and 8 extra output lines for each PLI connector. This extra I/O is a backup to the I/O already available on the MCF5272 which may be used by some customers for alternate functions, as many of the pins on the MCF5272 have 2 or 3 multiplexed functions. Each I/O equation uses a chip select signal (-CS5 & -CS6) and the -OE signal from the MCF5272 to define the control signal to the D-types.

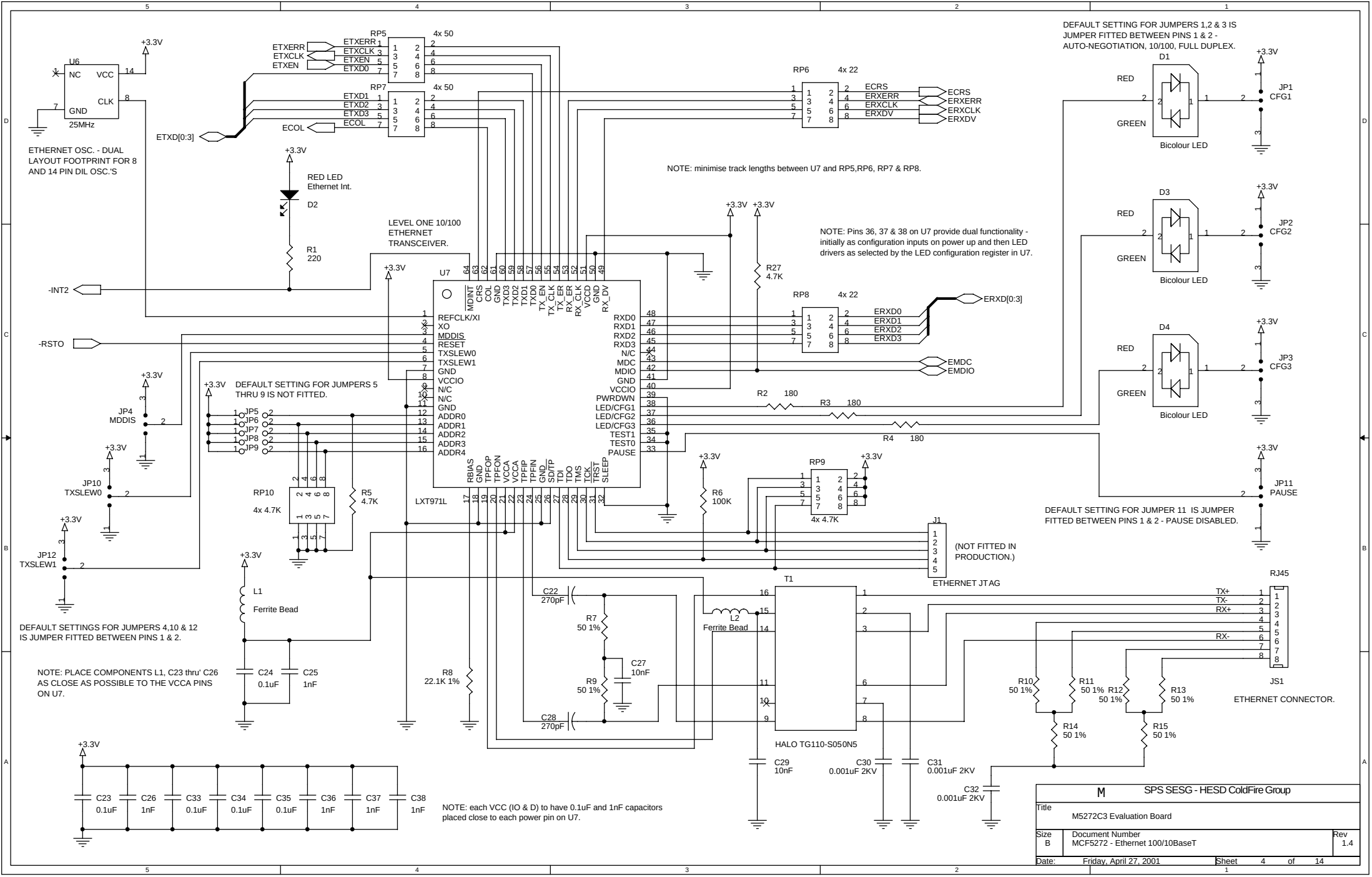
```
TITLE           U20
PATTERN         P00001
REVISION        1
DATE            3rd November 2000
AUTHOR          Pete Highton
COMPANY         Motorola SPS (c) 2000

CHIP    U20      PALCE16V8

PIN     1        CLK            COMBINATORIAL; I/P
PIN     2       /CS0            COMBINATORIAL; I/P
PIN     3       /CS2            COMBINATORIAL; I/P
PIN     4       /CS5            COMBINATORIAL; I/P
PIN     5       /CS6            COMBINATORIAL; I/P
PIN     6       /OE             COMBINATORIAL; I/P
PIN     7        NC
PIN     8        NC
PIN     9        NC
PIN     10       GND
```

```
PIN     11       NC
PIN     12      /BD_CS          COMBINATORIAL   ; O/P
PIN     13      /CS5_RD         COMBINATORIAL   ; O/P
PIN     14      /CS5_WR         COMBINATORIAL   ; O/P
PIN     15      /CS6_RD         COMBINATORIAL   ; O/P
PIN     16      /CS6_WR         COMBINATORIAL   ; O/P
PIN     17       NC
PIN     18       NC
PIN     19       NC
PIN     20       VCC

EQUATIONS

BD_CS = CS0 + CS2 + CS5 + CS6   ; Generic CS derived for data buffers.

CS5_RD = CS5 * OE               ; CS5 & OE used to read from PLI 0 connector.
                                ; /OE = read, OE = write.
CS5_WR = CS5 * /OE              ; CS5 & OE used to write to PLI 0 connector.

CS6_RD = CS6 * OE               ; CS6 & OE used to read from PLI 1 connector.

CS6_WR = CS6 * /OE              ; CS6 & OE used to write to PLI 1 connector.
```

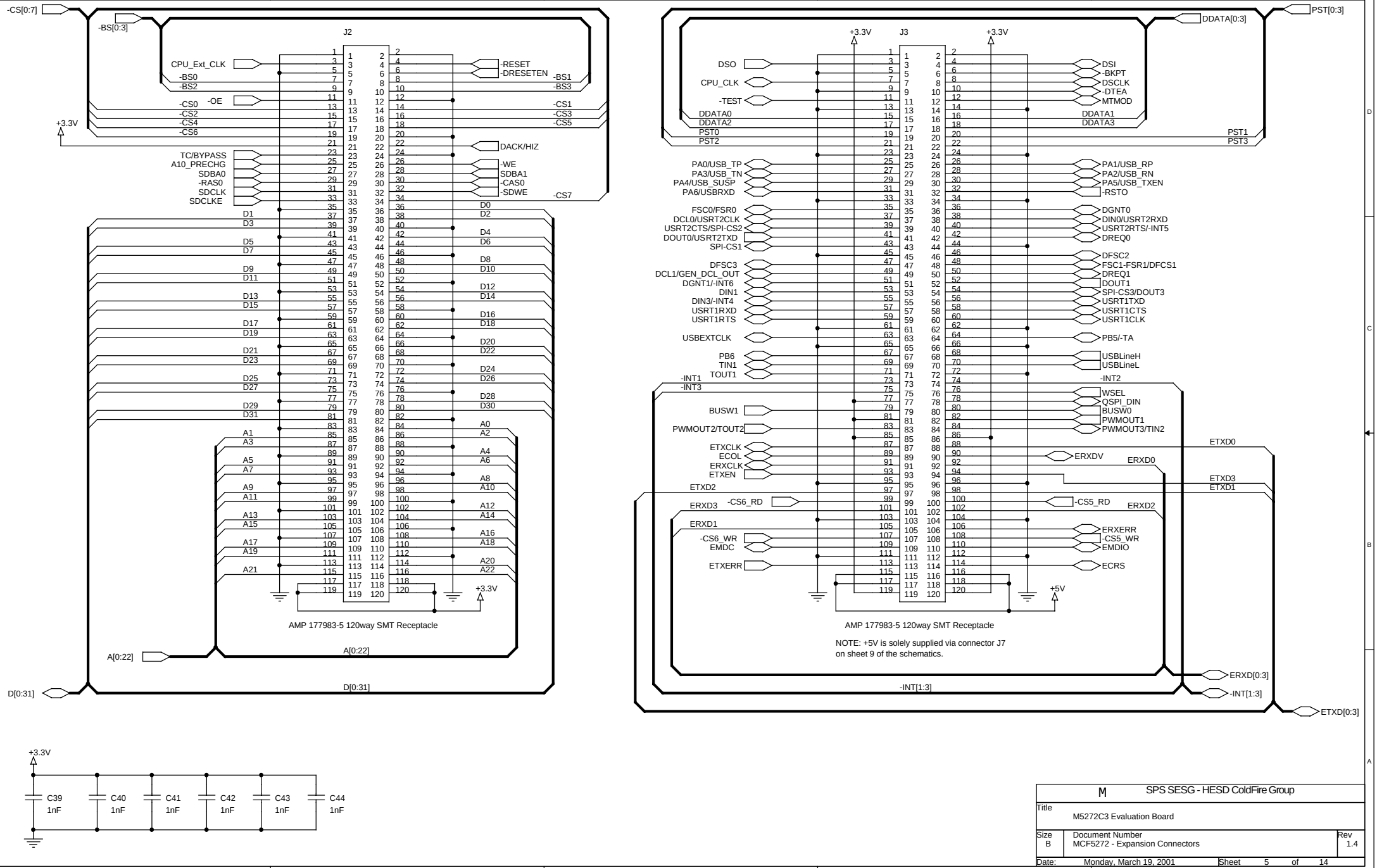




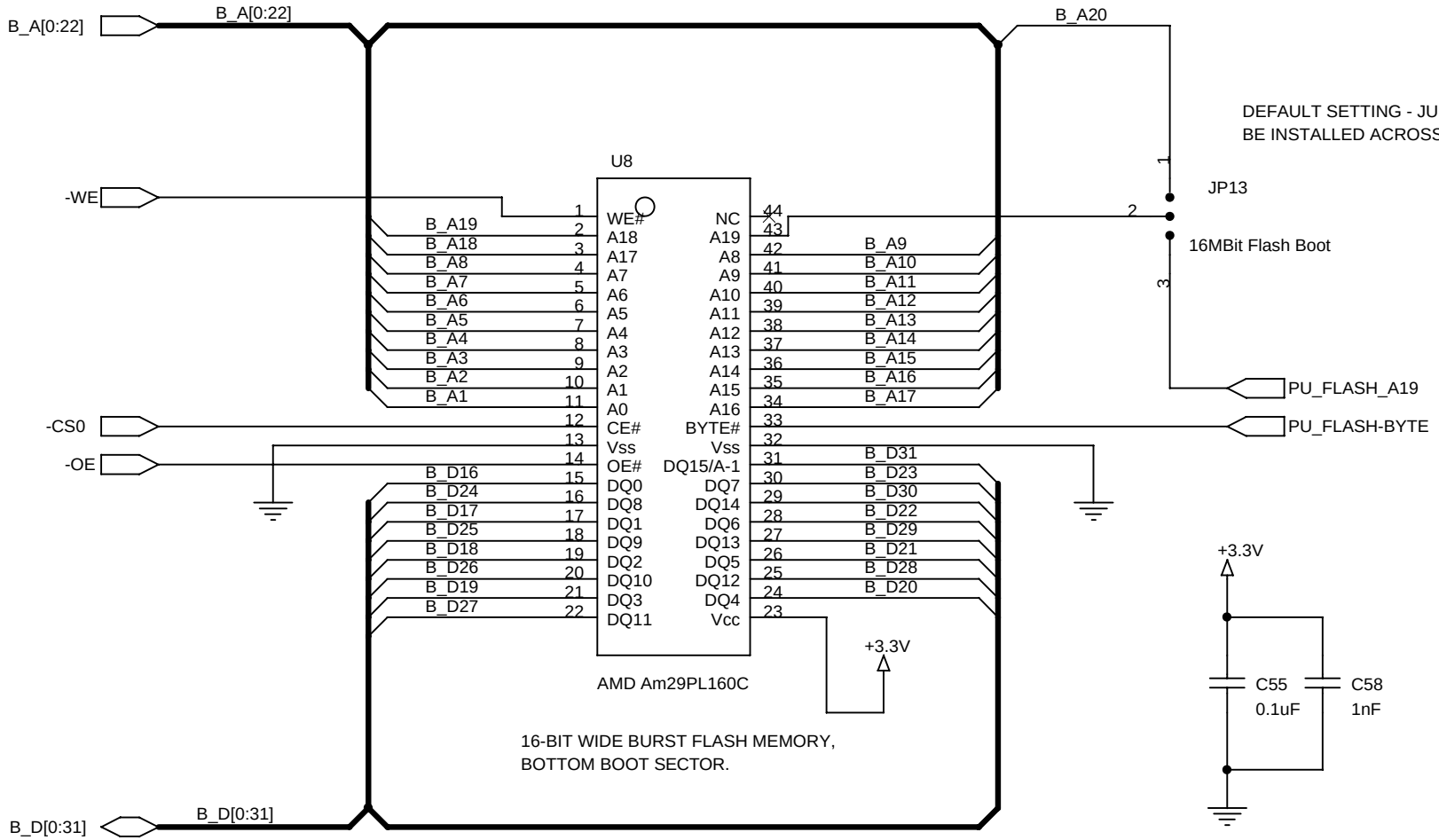




For More Information On This Product,
Go to: www.freescale.com

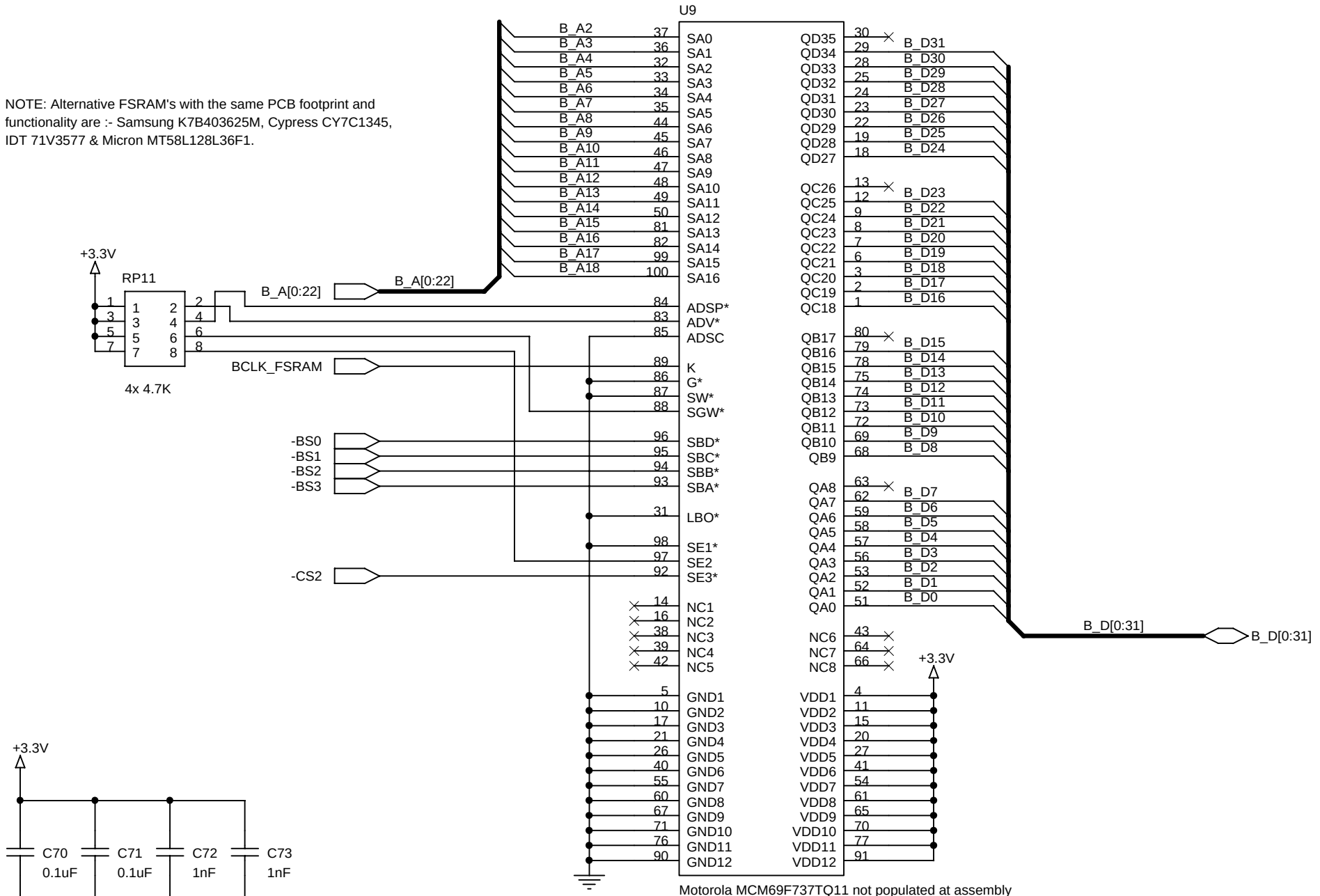


M	SPS SESG - HESD ColdFire Group	
Title	M5272C3 Evaluation Board	
Size B	Document Number MCF5272 - Expansion Connectors	Rev 1.4
Date: Monday, March 19, 2001	Sheet 5 of 14	



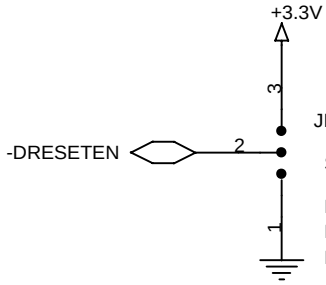
M	SPS SESG - HESD ColdFire Group	
Title: M5272C3 Evaluation Board		
Size: A	Document Number: MCF5272 - Flash Memory	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 6 of 14	

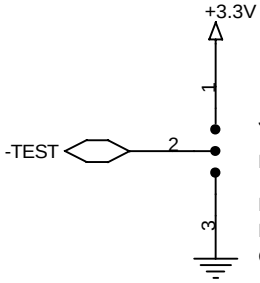
NOTE: Alternative FSRAM's with the same PCB footprint and functionality are :- Samsung K7B403625M, Cypress CY7C1345, IDT 71V3577 & Micron MT58L128L36F1.

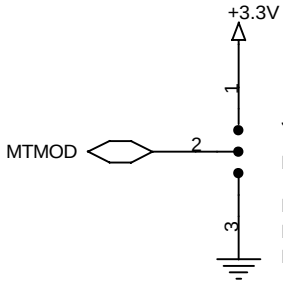


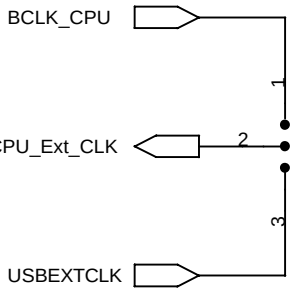
Motorola MCM69F737TQ11 not populated at assembly

M	SPS SESG - HESD ColdFire Group	
Title	M5272C3 Evalution Board	
Size: A	Document Number: MCF5272 - FSRAM	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 7 of 14	



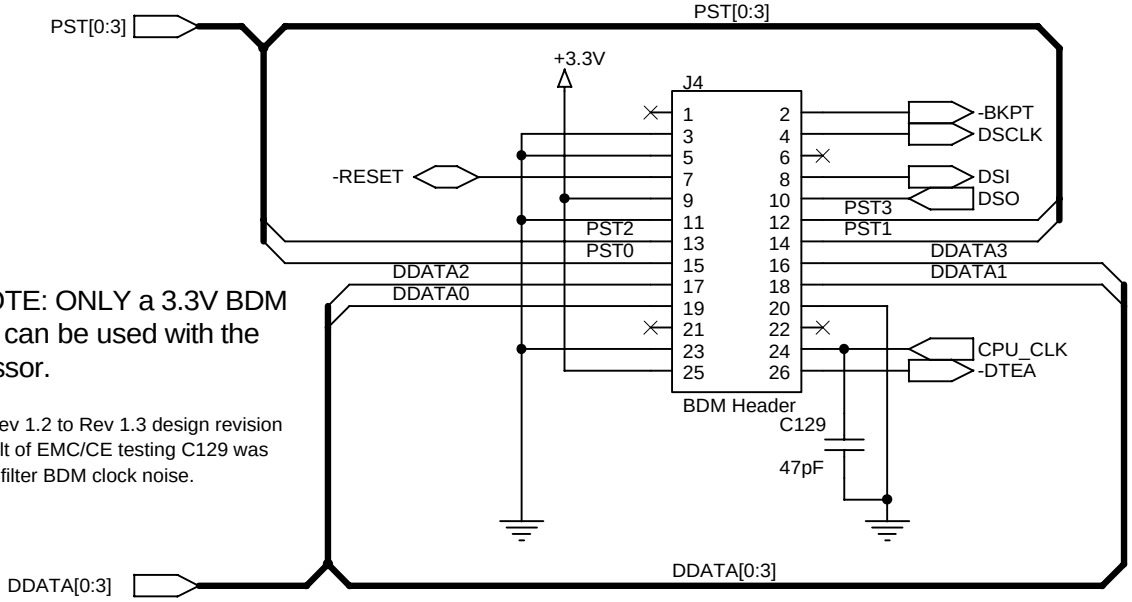






IMPORTANT NOTE: ONLY a 3.3V BDM debugging cable can be used with the MCF5272 processor.

NOTE: Rev 1.2 to Rev 1.3 design revision as a result of EMC/CE testing C129 was added to filter BDM clock noise.



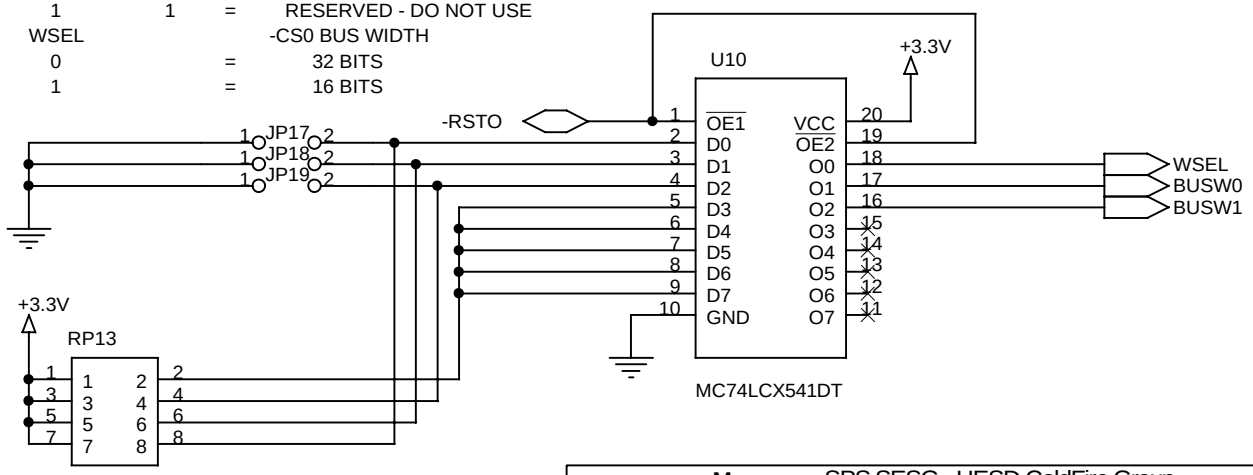
NOTE: 4.7K pull up resistors are used on signals -BKPT, DSCLK, DSI, DSO, -RESET & -DTEA. See page 11 of the schematics.

DEFAULT FOR JP17/WSEL IS FITTED (0).
DEFAULT FOR JP18/BUSW0 IS FITTED (0).
DEFAULT FOR JP19/BUSW1 IS NOT FITTED (1).

BUSW1	BUSW0		DATA BUS WIDTH
0	0	=	32 BITS
0	1	=	8 BITS
1	0	=	16 BITS
1	1	=	RESERVED - DO NOT USE

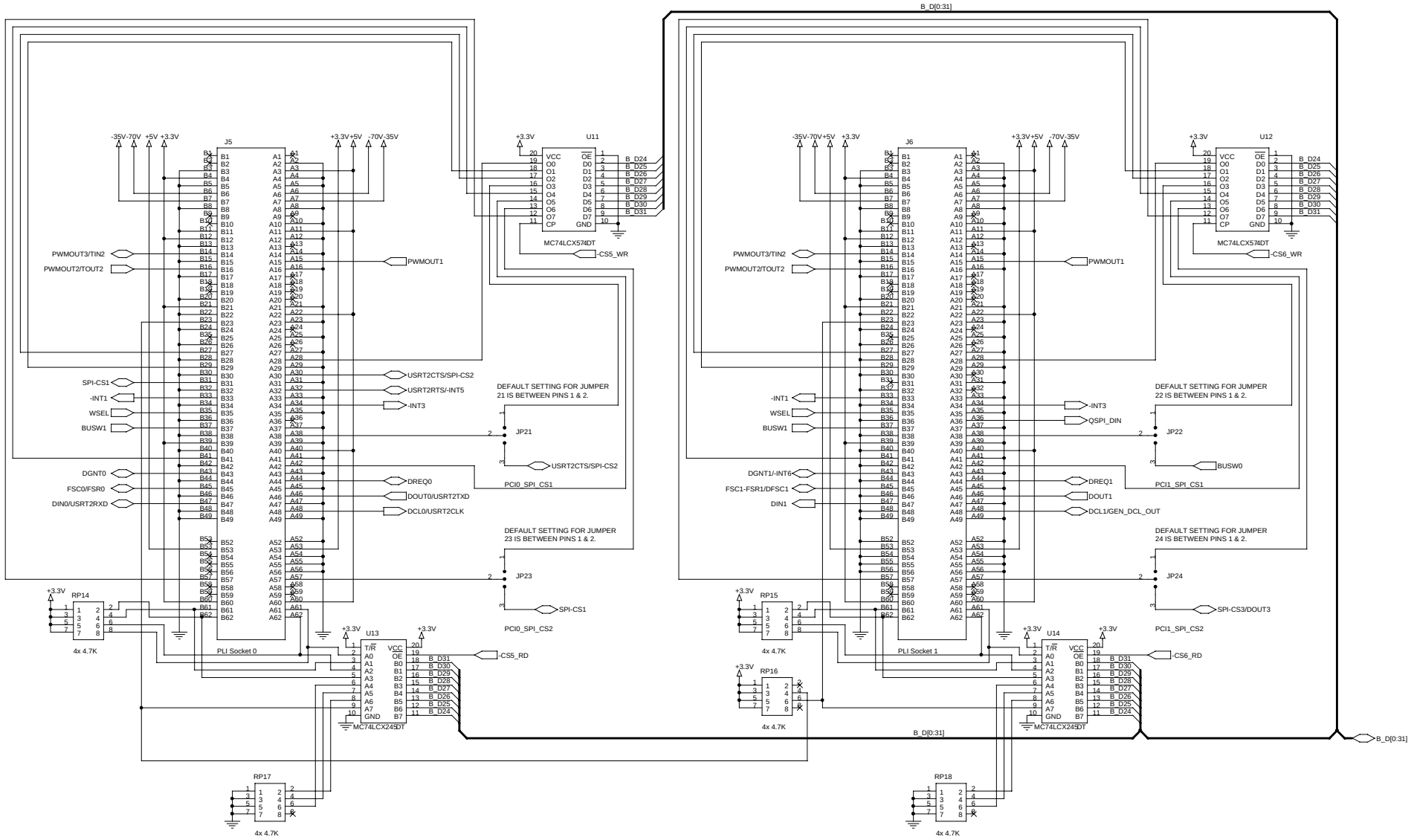
WSEL		-CS0 BUS WIDTH
0	=	32 BITS
1	=	16 BITS

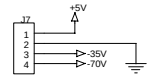
IMPORTANT NOTE: THE -RSTO SIGNAL MUST BE USED TO DRIVE THE OUTPUT ENABLE PINS OF U10 TO ALLOW THE WSEL, BUSW0 & BUSW1 SIGNALS TO BE LATCHED CORRECTLY BY THE MCF5272 - U5.



M	SPS SESG - HESD ColdFire Group	
Title	M5272C3 Evaluation Board	
Size A	Document Number MCF5272 Background Debug Mode (BDM) Port & Reset Config.	Rev 1.4
Date: Monday, March 19, 2001	Sheet 8 of 14	

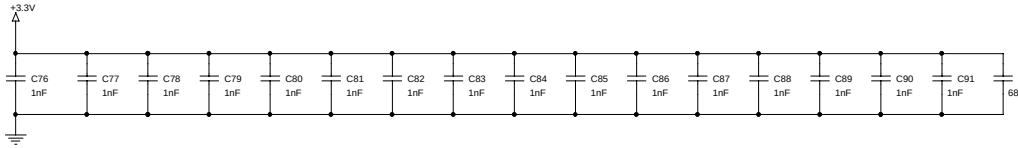
NOTE: PLI Sockets 0 & 1 are not populated during assembly.





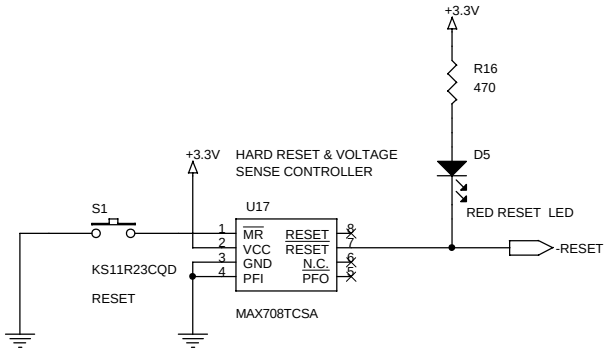
+5V & HIGH VOLTAGE CONN.

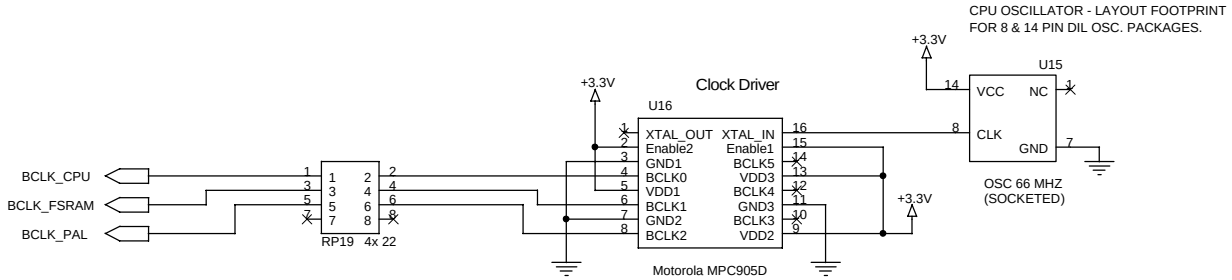
NOTE: all volatge supplies on J5 must be regulated and stable. +5V for the whole board is solely suppled via connector J7.



NOTE: Rev 1.2 to Rev 1.3 design revision as a result of EMC/CE testing C134 was added to filter high frequency clock harmonic noise near U16 (See schematic sheet 10).

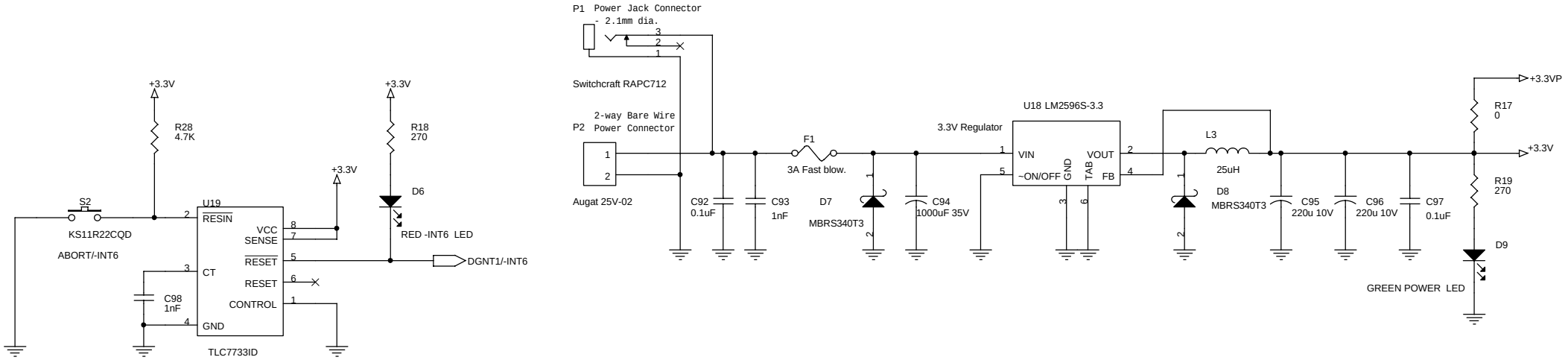
M	SPS SESG - HESD ColdFire Group	
Title	M5272C3 Evaluation Board	
Size C	Document Number MCF5272 - PLI Connectors	Rev 1.4
Date: Monday, March 19, 2001	Sheet 9 of 14	





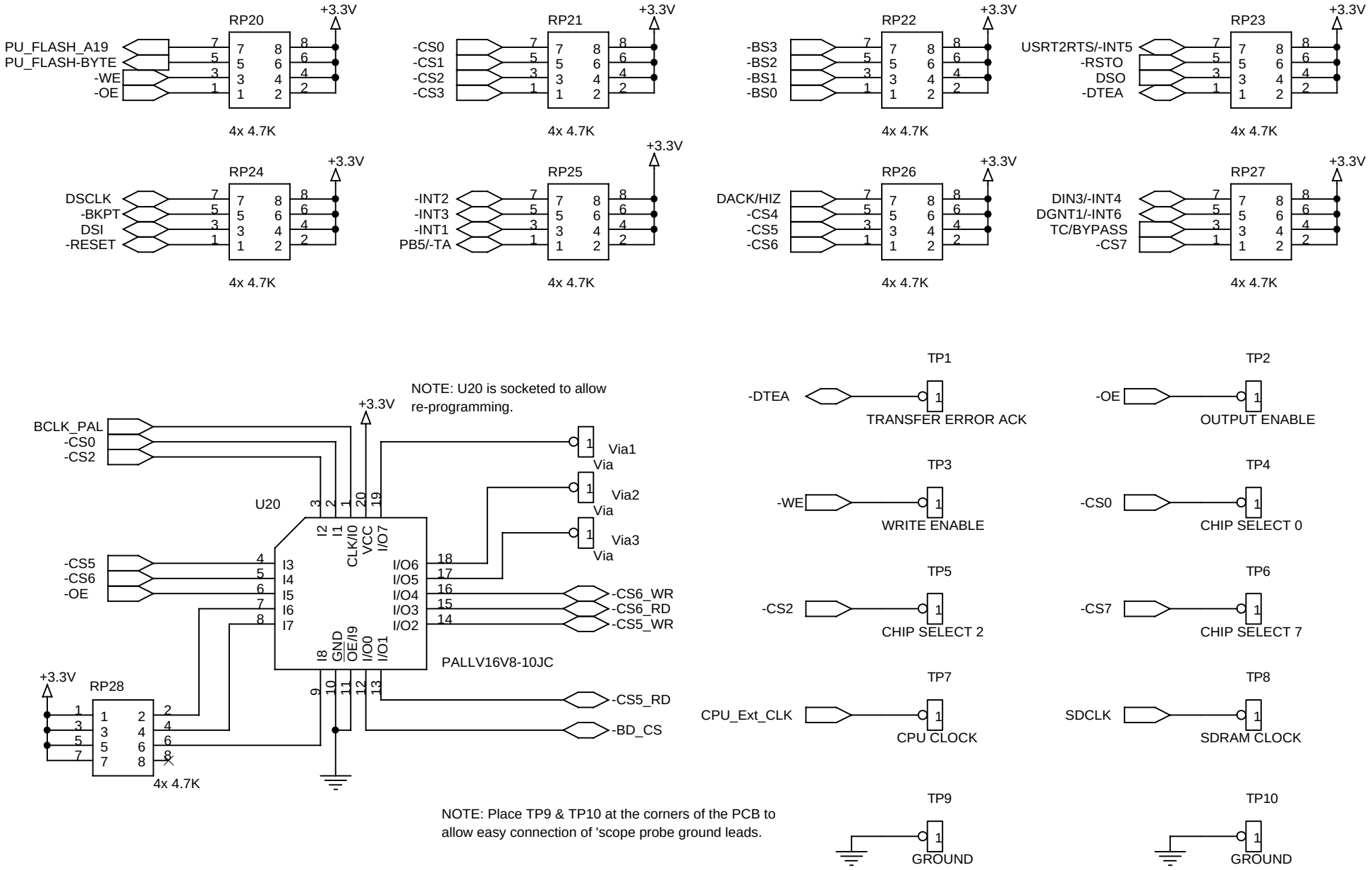
NOTE: minimise track lengths between U16 & RP19.

DC Voltage Input range +5V to +14V.



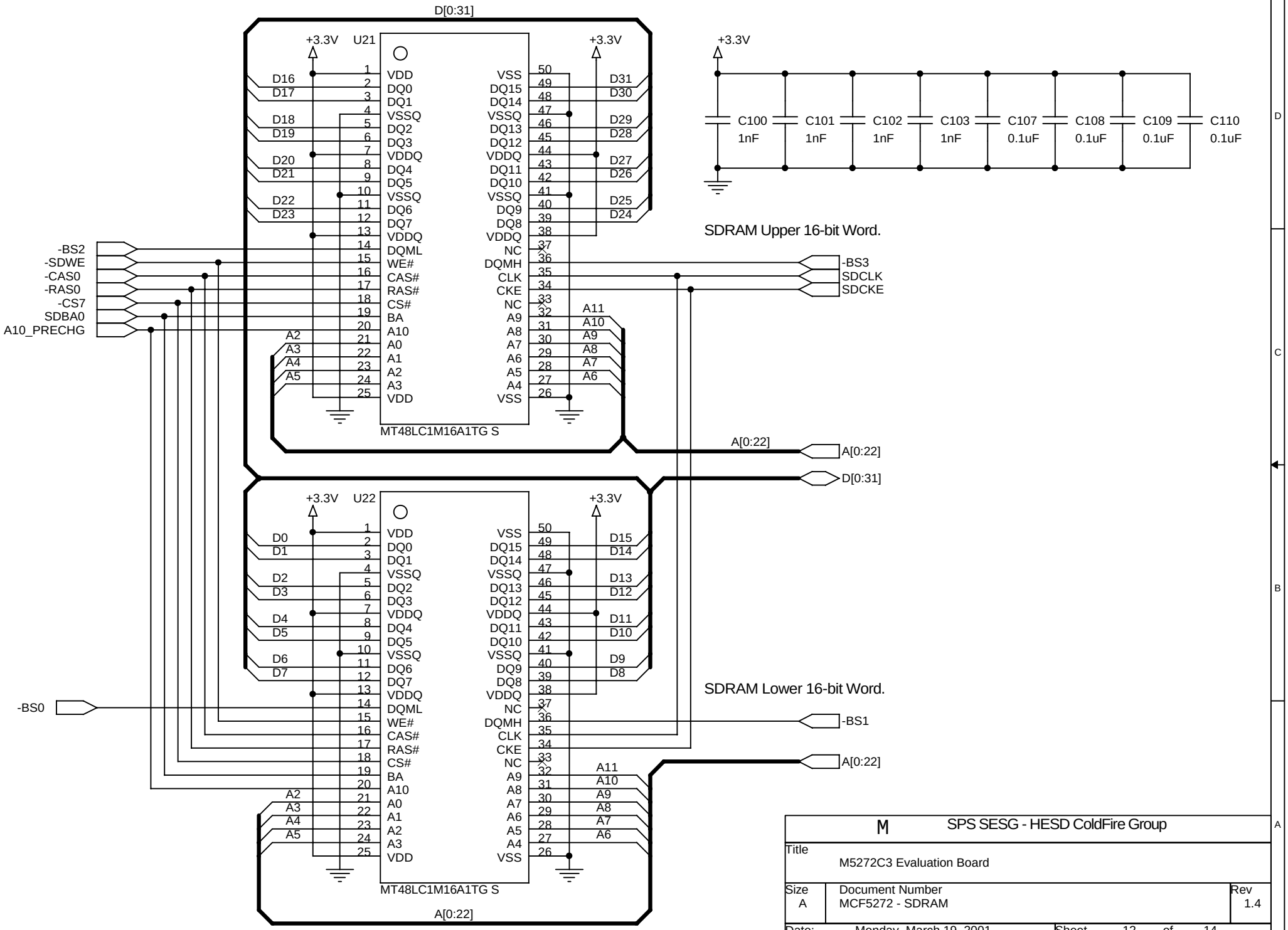
Debounced -INT6 Signal

SPS SESG - HESD ColdFire Group		
Title: M5272C3 Evaluation Board		
Size: B	Document Number: MCF5272 - PSU, Reset & Clocks	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 10 of 14	

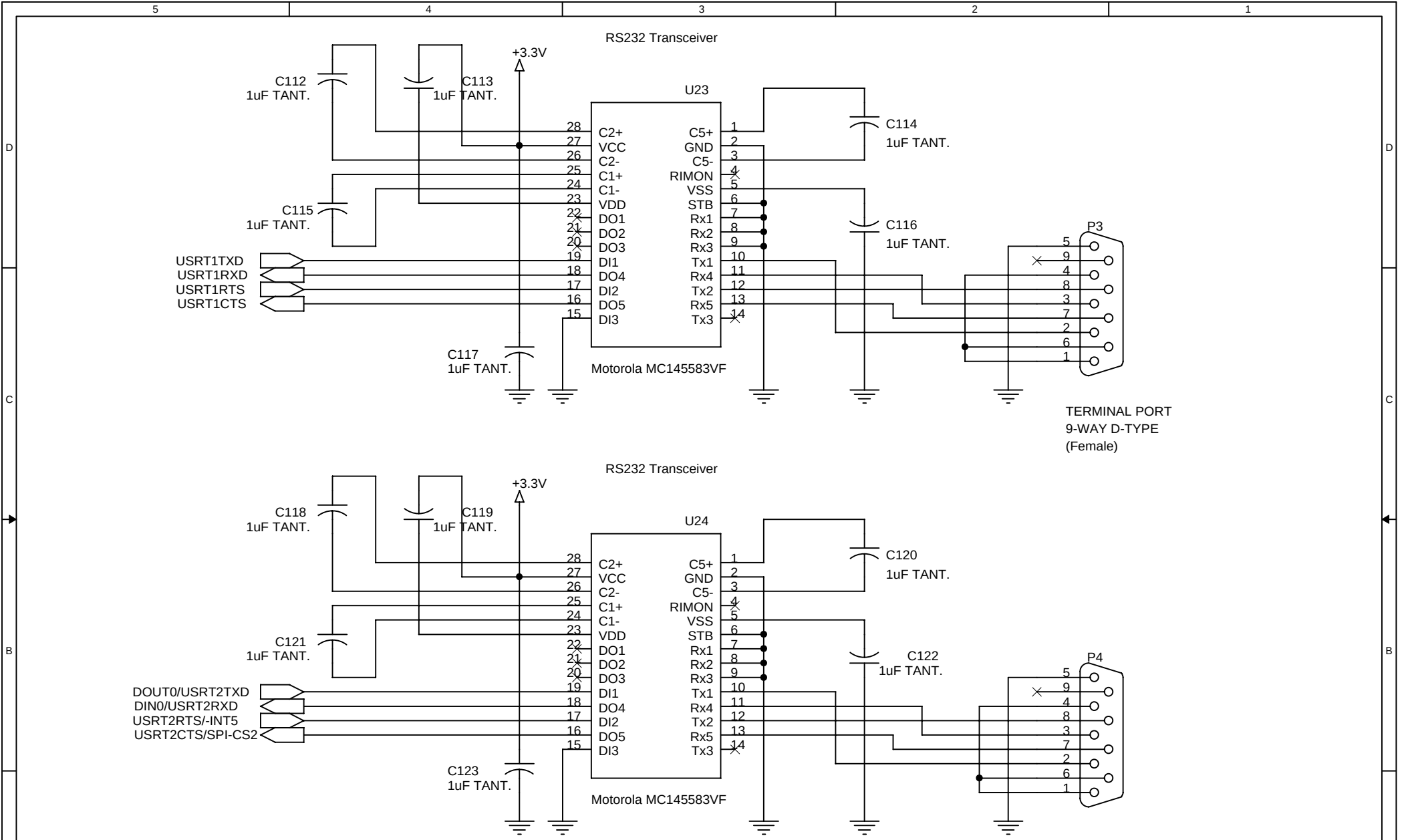


NOTE: Place TP9 & TP10 at the corners of the PCB to allow easy connection of 'scope probe ground leads.

M	SPS SESG - HESD ColdFire Group	
Title	M5272C3 Evaluation Board	
Size: A	Document Number: MCF5272 Pull-ups, Test points and PAL.	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 11 of 14	



M	SPS SESG - HESD ColdFire Group	
Title: M5272C3 Evaluation Board		
Size: A	Document Number: MCF5272 - SDRAM	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 12 of 14	



USB OSCILLATOR - LAYOUT FOOTPRINT
FOR 8 & 14 PIN DIL OSC. PACKAGES.

+3.3V

U25
NC VCC
CLK
GND
48MHz
(Not socketed)
USBEXTCLK

+3.3V JP25
USB MODE
DEFAULT JUMPER 25 SETTING
BETWEEN PINS 1 & 2.

DEFAULT JUMPER 26 SETTING IS NOT
FITTED. +5V WILL ONLY BE SUPPLIED
VIA J7 ON SCHEMATIC SHEET 9 FOR
OPERATION AS A USB HUB. NORMAL
OPERATION IS AS A USB DEVICE.

U26
MODE VCC
OE# VMO/FSEO
RCV VPO
VP D+
VM D-
SUSPND SPEED
GND NC

PA5/USB_TXEN
PA6/USBRXD
PA1/USB_RP
PA2/USB_RN
PA4/USB_SUSP

PA3/USB_TN
PA0/USB_TP

PDIUSBP11APW

USB TRANSCEIVER

+3.3V +5V JP26
USB POWER

R20
1.5K

USB PORT B

Default setting for the USBLineH
signal is R21 fitted R24 removed.

R21 0
R22 33
R23 33
R24 0
R25 0
R26 0

C124
10nF

JR1

Default setting for the USBLineL
signal is R25 fitted R26 removed.

USBLineL
USBLineH

+3.3V

C125 0.1uF
C126 0.1uF
C127 1nF
C128 1nF

M	SPS SESG - HESD ColdFire Group	
Title: M5272C3 Evaluation Board		
Size: A	Document Number: MCF5272 - USB Interface	Rev: 1.4
Date: Monday, March 19, 2001	Sheet 14 of 14	

Appendix D
Evaluation Board BOM

Table D-1. MCF5272EVM_BOM

Item	Qty	Reference	Part	Function
1	29	C1,C2,C3,C8,C9,C10,C11,C12, C23,C24,C33,C34,C35,C55,C56 ,C57,C69,C70,C71,C92,C97,C9 9,C107,C108,C109,C110,C111,C 125,C126	0.1 UF	SMT Decoupling Capacitors
2	75	C4,C5,C6,C7,C13,C14,C15,C16, C17,C18,C19,C25,C26,C36,C37 ,C38,C39,C40,C41,C42,C43,C4 4,C45,C46,C47,C48,C49,C50,C 51,C52,C53,C54,C58,C59,C60, C61,C62,C63,C64,C65,C66,C67 ,C68,C72,C73,C74,C75,C76,C7 7,C78,C79,C80,C81,C82,C83,C 84,C85,C86,C87,C88,C89,C90, C91,C93,C98,C100,C101,C102, C103,C104,C105,C106,C127,C1 28,C129	1nF	SMT Decoupling Capacitors
3	4	C20,C27,C29,C124	10nF	SMT Capacitors
4	1	C21	10 uF TANT.	SMT Capacitor
5	2	C22,C28	270pF COG or NPO only	SMT Capacitors
6	3	C30,C31,C32	0.001uF 2KV	SMT Capacitors
7	1	C94	Rubycon 1000uF 35V	SMT Capacitor
8	2	C95,C96	AVX TPSE220M10RLM	SMT Capacitors
9	12	C112,C113,C114,C115,C116,C1 17,C118,C119,C120,C121,C122, C123	Size B 1uF TANT.	SMT Capacitors
10	3	D1,D3,D4	LSGT670 Bicolour LED's	Infineon SMT LEDs
11	3	D2,D5,D6	LST670 Red LED's	Infineon SMT LEDs
12	2	D7,D8	Motorola MBRS340T3	SMT Schottky Power Diodes
13	1	D9	LGT670 Green LED	Infineon SMT LED

Table D-1. MCF5272EVM_BOM (Continued)

Item	Qty	Reference	Part	Function
14	1	F1	MULTICOMP MCHTE-15M	Fuse
14A	1	F1	Littelfuse F920-ND 5A/250V	PSU Fast Acting Fuse
15	17	JP1,JP2,JP3,JP4,JP10,JP11,JP12,JP13,JP14,JP15,JP16,JP20,JP21,JP22,JP23,JP24,JP25	Harwin M22-2010305	3-way Jumpers
16	8	JP5,JP6,JP7,JP8,JP9,JP17,JP18,JP19	Harwin M22-2010205	2-way Jumpers
17	1	JR1	AMP 787780-1	USB PortB connector
18	1	JS1	Molex	RJ45 socket
19	1	J1	Berg 0.1” SIL	Ethernet JTAG connector
20	2	J2,J3	AMP177983-5	120-way SMT Receptacle expansion connectors
21	1	J4	Thomas&Betts 609-2627	BDM 26-way header
22	1	J5	Molex 89177A PCI socket	PLI SOCKET 0
23	1	J6	Molex 89177A PCI socket	PLI SOCKET 1
24	1	J7	AMP 350211-1	+5V & HIGH VOLTAGE CONNECTOR
25	2	L1,L2	CTCB1210-600-HC	Central Technology Ferrite Bead
26	1	L3	Siemens B82111-B-C24	25uH INDUCTOR
27	1	P1	Switchcraft RAPC712	PSU barrel connector
28	1	P2	Augat 25V-02	2-way bare wire power connector
29	2	P3,P4	Molex DB9 connector	RS232 9way DType thru board
30	21	RP1,RP2,RP9,RP10,RP11,RP12,RP13,RP14,,RP15,RP16,RP17,RP18,RP20,RP21,RP22,RP23,RP24,RP25,RP26,RP27,RP28	Philips	SMT 4K7x4 resistor packs
31	7	RP3,RP4,RP6,RP8,RP19	Philips	SMT 22x4 resistor packs
32		RP5,RP7	Philips	SMT 50x4 resistor packs
33	1	R1	220R	SMT 220 0805 resistor
34	3	R2,R3,R4	180R	SMT 180 0805 resistors
35	1	R5,R27,R28	4K7	SMT 4K7 0805 resistors
36	1	R6	100K	SMT 100K 0805 resistor
37	8	R7,R9,R10,R11,R12,R13,R14,R15	50R 1%	SMT 50 0805 resistors
38	1	R8	22K1 1%	SMT 22K1 0805 resistors

Table D-1. MCF5272EVM_BOM (Continued)

Item	Qty	Reference	Part	Function
39	3	R16,R18,R19	270R	SMT 270 0805 resistors
40	5	R17,R21,R24,R25,R26	0R	SMT 0 0805 resistors
41	1	R20	1K5	SMT 1.5K 0805 resistor
42	3	R22,R23	33R	SMT 33 0805 resistors
43	1	S1	C&K KS11R23CQD	Hard reset red push-button switch
44	1	S2	C&K KS11R22CQD	/IRQ7 black push-button switch
45	10	TP1,TP2,TP3,TP4,TP5,TP6,TP7 ,TP8,TP9,TP10	Hughes 100-103	Test Points
46	1	T1	Halo TG110-S050N5	Ethernet isolation transformer
47	3	U1,U2,U3,	MC74LCX16245DT	16-bit wide bus transceivers
48	3	U4,U13,U14	MC74LCX245DT	8-bit wide bus transceivers
49	1	U5	MCF5272ZP66	MCF5272 ColdFire processor
50	1	U6	Pletronics Osc 25 MHz	Oscillator for ethernet transceiver
51	1	U7	LXT971L	Level One Ethernet transceiver
52	1	U8	Am29PL160C	AMD 1Mx16 burst Flash memory
53	1	U9 *	MCM69F737TQ11	Burst FSRAM (not populated)
54	1	U10	MC74LCX541DT	8-bit wide bus buffer
55	1	U11,U12	MC74LCX574DT	8-bit wide Octal D-type flip-flop
56	1	U15	Pletronics Osc 66MHz	Oscillator for the MCF5272
57	1	U16	MPC905D	Clock driver
58	1	U17	Maxim MAX708TCSA	System reset & voltage sense controller
59	1	U18	LM2596S-3.3	National Semi DCtoDC switcher
60	1	U19	TLC7733ID	TI switch debounce circuit
61	1	U20	PALLV16V8-10JC	Lattice 3.3V erasable PAL
62	2	U21,U22	MT48LC1M16A1TG-10JC	Micron Tech. SDRAMs 16Mbit
63		U23,U24	MC145583VF	RS232 transceiver - 3.3V supply
64	1	U25	Pletronics Osc 48MHz	Oscillator for the USB module
65	1	U26	PDIUSB11APW	Philips USB transceiver
66	1	SKT	3M 220-7160-75-1157	PLCC socket for U20

*Alternate Parts - Samsung K7B403625M, GalvantechGVT71128E36,ISSI IS61SF12836, Micron MT58L128L36F1, GSI GS84036A, IDT 71V3577, Cypress CY7C1345