M5249C3 Reference Board User's Manual

Document Number: M5249C3UM
Rev. 1
07/2006

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 26668334
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

M5249C3UM
Rev. 1
07/2006

WARNING

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

Chapter 1
M5249C3 Board

1.1 General Hardware Description 1-1
1.2 System Memory 1-2
1.3 Serial Communication Channels 1-3
1.4 Parallel I/O Ports 1-3
1.5 Programmable Timer/Counter 1-3
1.6 Ethernet Controller 1-3
1.7 System Configuration 1-3
1.8 Installation and Setup 1-4
1.8.1 Unpacking 1-5
1.8.2 Preparing the Board for Use 1-5
1.8.3 Providing Power to the Board 1-5
1.8.4 Selecting Terminal Baud Rate 1-5
1.8.5 The Terminal Character Format 1-6
1.8.6 Connecting the Terminal 1-6
1.8.7 Using a Personal Computer as a Terminal 1-6
1.9 System Power-up and Initial Operation 1-8
1.10 M5249C3 Jumper Setup 1-9
1.11 Using The BDM Port 1-10

Chapter 2
Using the Monitor/Debug Firmware

2.1 What Is dBUG? 2-1
2.2 Operational Procedure 2-2
2.2.1 System Power-up 2-2
2.2.2 System Initialization 2-3
2.3 Command Line Usage 2-4
2.4 Commands 2-5
2.5 TRAP #15 Functions 2-38
2.5.1 OUT_CHAR 2-38
2.5.2 IN_CHAR 2-38
2.5.3 CHAR_PRESENT 2-39
2.5.4 EXIT_TO_dBUG 2-39

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 MCF5249 Registers and Memory Map 3-3
3.1.9 Reset Vector Mapping 3-4
3.1.10 TA Generation 3-5
3.1.11 Wait State Generator 3-5
3.1.12 SDRAM 3-5
3.1.13 Flash ROM 3-5
3.1.14 JP12 Jumper and the User's Program 3-6
3.2 Serial Communication Channels 3-6
3.2.1 MCF5249 UARTs 3-6
3.2.2 QSPI Module 3-6
3.2.3 General Purpose I/O Pins 3-7
3.2.4 Ethernet Controller 3-7
3.2.5 Audio Module 3-8
3.2.6 I2C Module 3-8
3.2.7 Analog to Digital Converter (ADC) Module 3-8
3.2.8 Flash Memory Card/IDE Interface Module 3-9
3.3 Connectors and Expansion Bus 3-9
3.3.1 Expansion Connectors - J4 and J5 3-9
3.3.2 The Debug Connector J2 3-11

Appendix A
Configuring dBUG for Network Downloads

A.1 Required Network Parameters A-1
A.2 Configuring dBUG Network Parameters A-1
A.3 Troubleshooting Network Problems A-2

Appendix B
PAL Equations

Appendix C
Schematics

Appendix D
Evaluation Board BOM

Chapter 1
M5249C3 Board

The M5249C3 is a versatile single board computer based on the MCF5249 ColdFire® processor. It may be used as a powerful microprocessor-based controller in a variety of applications. 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 a power supply to have a fully functional system.

This board can be connected to external peripherals supplied by the user to expand memory and I/O capabilities via the Expansion Bus connectors J4 & J5 (see schematic diagram). Buffers may be required to compensate for bus loading if external peripherals are connected to the system.

1.1 General Hardware Description

The M5249C3 board provides SDRAM, Flash ROM, an Ethernet interface (10/100BaseT), and RS-232 in addition to the built-in I/O functions of the MCF5249 device for programming and evaluating the attributes of the microprocessor. The MCF5249 device is a member of the ColdFire® family of processors. It is a 32-bit processor with a 24-bit address bus and 16 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 MCF5249 processor 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 four 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 *MCF5249 User's Manual* for detailed information about the chip selects). Two of the four chip selects are used to access devices in the system. One chip select ($\overline{CS0}$) is used to access the Flash ROM and the other ($\overline{CS1}$) is used to access the Ethernet controller (U4 on the schematics).The DRAM controller is used to control one SDRAM device providing 8MB of SDRAM memory configured as 4MBx16 words. All other functions of the SIM are available to the user.

Figure 1-1 shows the M5249C3 block diagram.

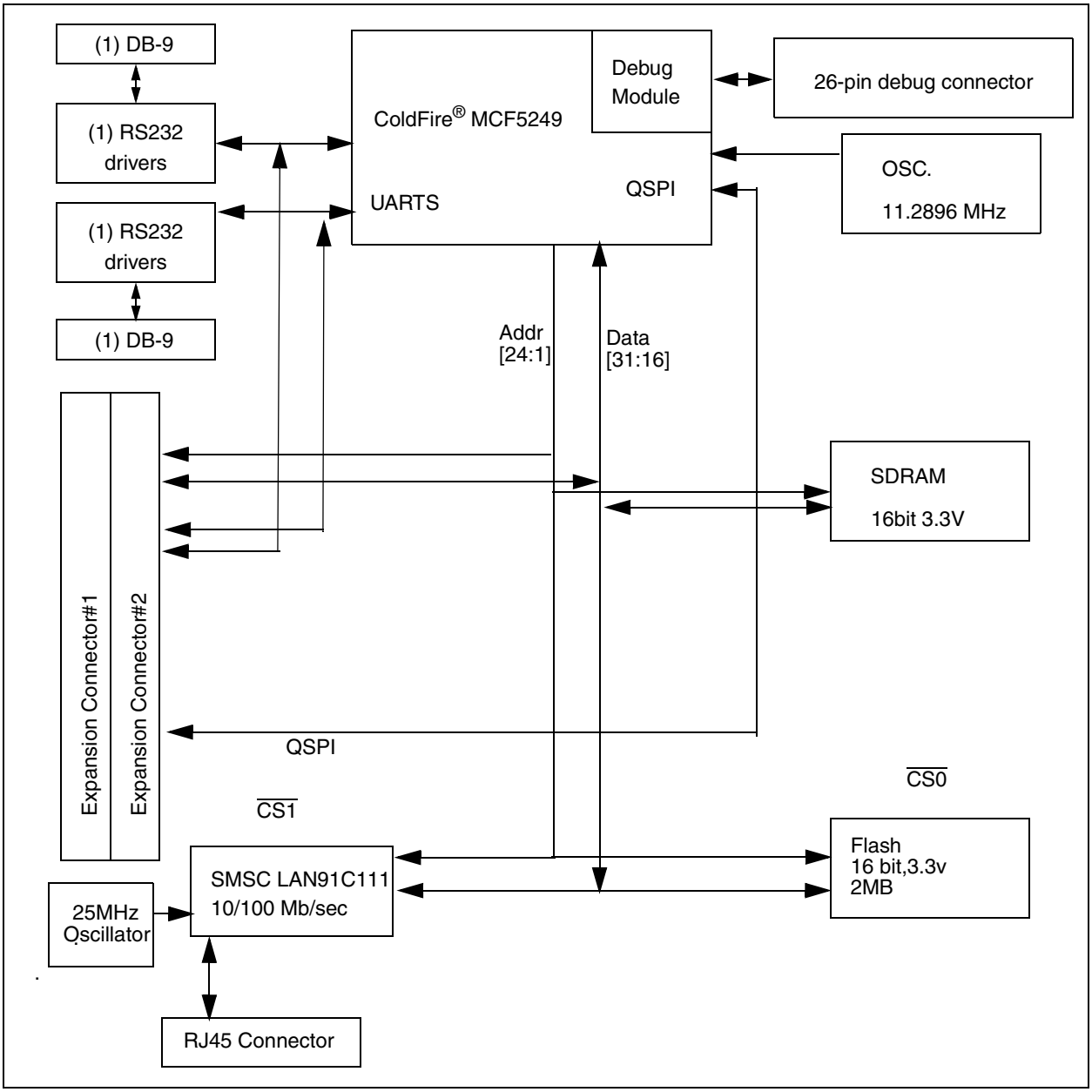


Figure 1-1. M5249C3 Block Diagram

1.2 System Memory

One on-board Flash ROM (U6) is used in the system. The Am29LV160DB-XX device contains 16Mbits of non-volatile storage (1 MByte x 16) giving a total of 2MBytes of Flash memory. The lower 256 KBytes are used to store the M5249C3 dBUG debugger/monitor firmware.

The MCF5249 processor has 96KBytes of internal SRAM organized as 1 bank of 64Kbytes and 1 bank of 32KBytes. The SRAM can be used for either data or instruction space.

There is one SDRAM (U7) device on the PCB. The system ships with 1x 4M x 16 of SDRAM totalling 8MBytes of volatile memory.

The internal cache of the MCF5249 is non-blocking. The instruction cache is 8 KBytes 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.

1.3 Serial Communication Channels

The MCF5249 processor 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 RS-232 compatible. An RS-232 serial cable with DB9 connectors is included with the board. The signals of both channels are available on the 120 pin expansion connector (J5). UART0 channel is the "TERMINAL" channel used by dBUG for communication with an external terminal/PC. The "TERMINAL" baud rate defaults to 19200.

1.4 Parallel I/O Ports

The MCF5249 offers 47-lines of general-purpose I/O of which 11 are dedicated inputs and 10 are dedicated outputs. Eight of the GPIO lines are also available as edge sensitive interrupt inputs.

1.5 Programmable Timer/Counter

The MCF5249 has two built-in general purpose timers/counters and a software watchdog timer. All timers are available to the user. The signals for each timer are available on the 120 pin expansion connector (J5).

1.6 Ethernet Controller

The M5249C3 PCB has an Ethernet controller (SMSC LAN91C111 U4) operating at 10M bits/sec or 100Mbits/sec. The dBUG ROM monitor is programmed to allow a user to download files over a network to memory in different formats. The compiler formats currently 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.

1.7 System Configuration

The M5249C3 board requires the following items for minimum system configuration:

- The M5249C3 board (provided).
- Power supply, +7V 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 the minimum system configuration.

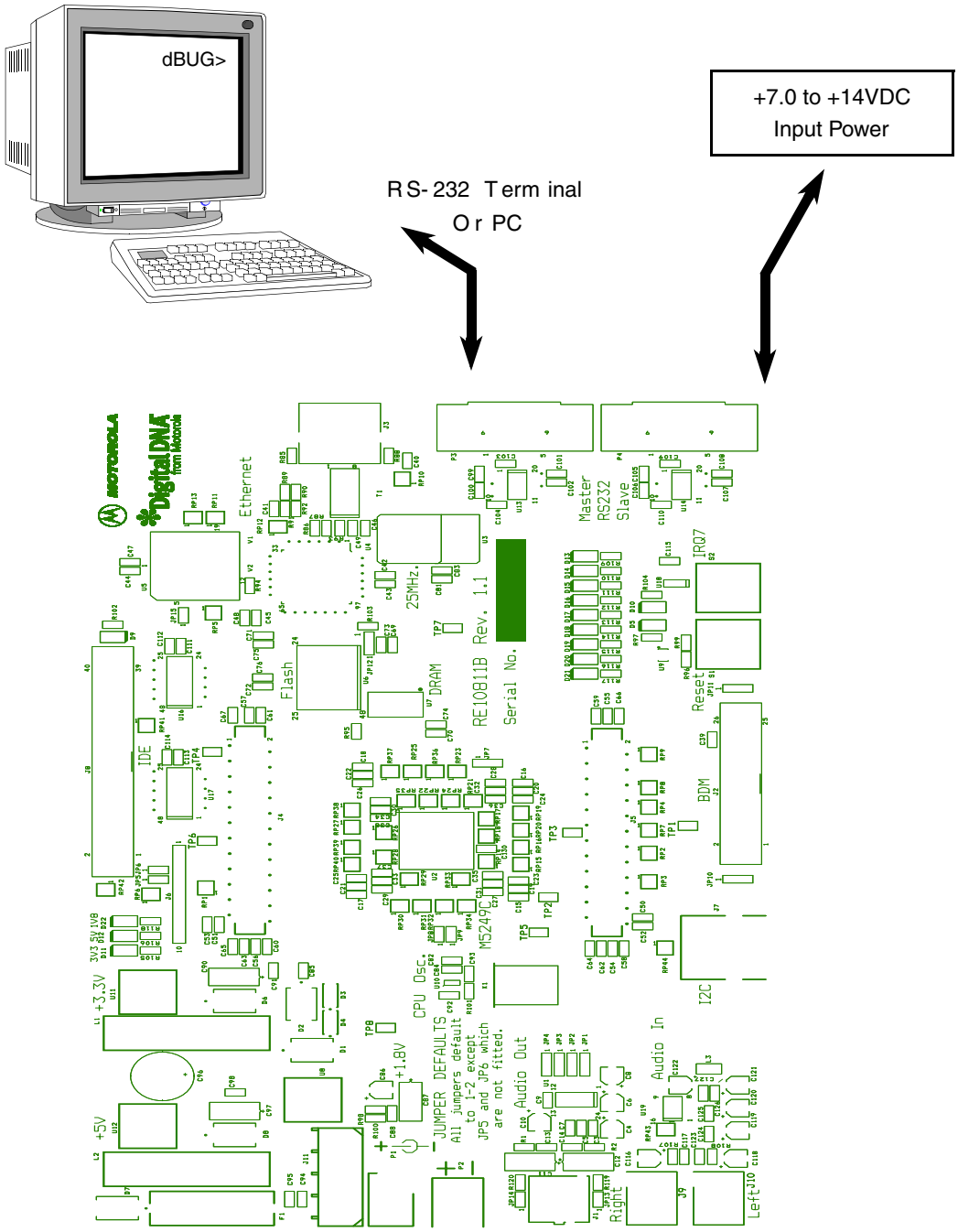


Figure 1-2. Minimum System Configuration

1.8 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 documented on the master jumper table and printed on the underside of the board (see Table 1-2). After the board is

functional in its default mode, the Ethernet interface may be used by following the instructions provided in Appendix A, "Configuring dBUG for Network Downloads."

1.8.1 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:

- M5249C3 Single Board Computer
- *M5249C3 User's Manual* (this document)
- One RS232 communication cable
- One BDM (Background Debug Mode) "wiggler" cable
- *MCF5249UM ColdFire Integrated Microprocessor User Manual*
- *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 Rapid PCB immediately; for contact details please see the front of this manual.

1.8.2 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. Figure 1-4 shows the position of the jumpers and connectors.

1.8.3 Providing Power to the Board

The board accepts three means of power supply connection, either P1, P2 or J11. Connector P1 is a 2.1mm power jack, P2 a lever actuated connector and J11 is a PC disk drive type power connector. The board accepts +7V to +14V DC at 1.0 Amp via either of the connectors.

Table 1-1. Power Supply Connections on P2

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

1.8.4 Selecting Terminal Baud Rate

The serial channel UART0 of the MCF5249 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, "Using the Monitor/Debug Firmware," for the discussion of this command.

1.8.5 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 with no flow control. It is neccessary to ensure that the terminal or PC is set to this format.

1.8.6 Connecting the Terminal

The board is now ready to be connected to a PC/terminal. Use the RS232 serial cable to connect the PC/terminal to the M5249C3 PCB. 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 connector P3 on the M5249C3 board. Connect the 9-pin female connector to one of the available serial communication channels normally referred to as COM1 (COM2, etc.) on the PC running terminal emulation software. The connector on the PC/terminal may be either male 25-pin or 9-pin. It may be neccessary to obtain a 25pin-to-9pin adapter to make this connection. If an adapter is required, refer to Figure 1-3 which shows the pin assignment for the 9-pin connector on the board.

1.8.7 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 1.8.6, "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 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 1.8.5, "The Terminal Character Format") The baud rate should be set to 19200. Power can now be applied to the board.

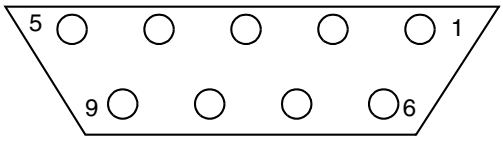


Figure 1-3. Pin assignment for female (Terminal) connector

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 on the next page shows the jumper locations for the board.

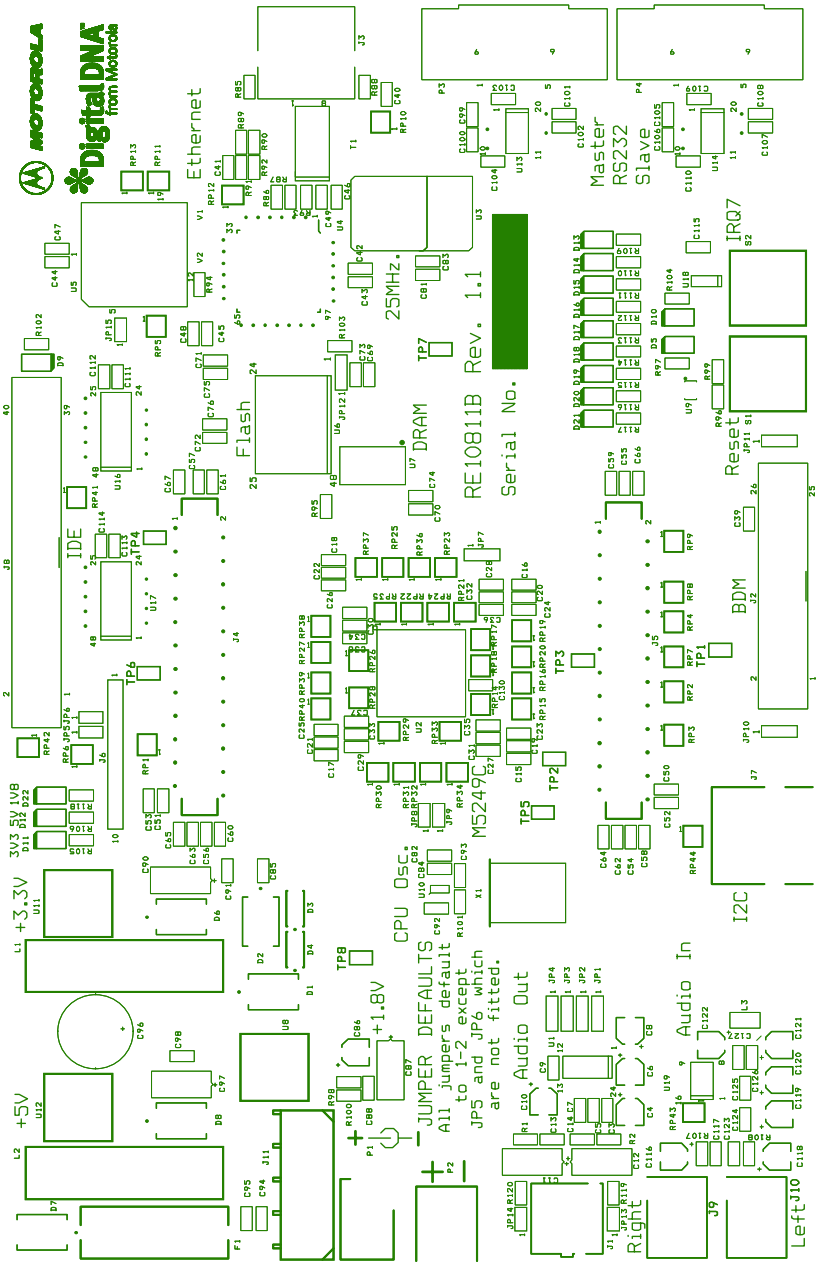


Figure 1-4. Jumper Locations

1.9 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: 8M
```

```
Copyright 1995-2002 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5249 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, "Using the Monitor/Debug Firmware." 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. Contact Rapid PCB for further instructions; please see the beginning of this manual for contact details.

1.10 M5249C3 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	Audio DAC AK4360VF U1 De-emphasis on
	2-3	Audio DAC AK4360VF U1 De-emphasis off
JP2	* 1-2	Audio DAC AK4360VF U1 Boost -high pass correction on
	2-3	Audio DAC AK4360VF U1 Boost - high pass correction off
JP3	*1-2	Audio DAC AK4360VF U1 Clock sample rate = 384fs
	2-3	Audio DAC AK4360VF U1 Clock sample rate = 256fs
JP4	*1-2	Audio DAC AK4360VF U1 Audio format I^2S 16/18/20 bit
	2-3	Audio DAC AK4360VF U1 Audio format 16-bit LSB justified
JP5	1-2	Not connected - I2C channel 0 pull-up SDA0, MCF5249 U2
JP6	1-2	Not connected - I2C channel 0 pull-up SCL0, MCF5249 U2
JP7	*1-2	BDM mode selected at power-on/reset (POR)
	2-3	JTAG mode selected at power-on/reset (POR)
JP8	*1-2	Current measurement for the CPU core +1.8V
JP9	*1-2	Current measurement for the CPU I/O pads +3.3V
JP10	*1-2	BDM connector (J2) I/O or Pad voltage +3.3V

Table 1-2. Jumper Settings (continued)

Jumper Setting		Function
	2-3	BDM connector (J2) I/O or Pad voltage +1.8V
JP11	*1-2	BDM connector (J2) Core voltage +3.3V
	2-3	BDM connector (J2) Core voltage +1.8V
JP12	**1-2	Flash EEPROM (U6) - boot into dBUG ROM monitor at power-on/reset (POR)
	2-3	Flash EEPROM (U6) - boot into Flash ignoring first 256K sector at POR
JP13	*1-2	Audio DAC AK4360VF U1 left audio channel load
JP14	*1-2	Audio DAC AK4360VF U1 right audio channel load
JP15	*1-2	Routes $\overline{CS1}$ from U2 (MCF5249) to PAL U5 and hence to Ethernet controller U4. Removed this isolates $\overline{CS1}$ signal for use with the expansion connectors (J4 & J5)

1.11 Using The BDM Port

The MCF5249 microprocessor has a built in debug module referred to as BDM (background debug module). In order to use BDM, simply connect the 26-pin debug connector on the board, J2, to the P&E BDM wiggler cable provided in the kit. 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 tools. Details may be found on CD-ROM included in this kit.

Chapter 2
Using the Monitor/Debug Firmware

The M5249C3 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 with no flow control. 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, at the discretion of the user's program. 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:

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

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

The 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, "M5249C3 Board." 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 2-1 shows the dUBG operational mode.

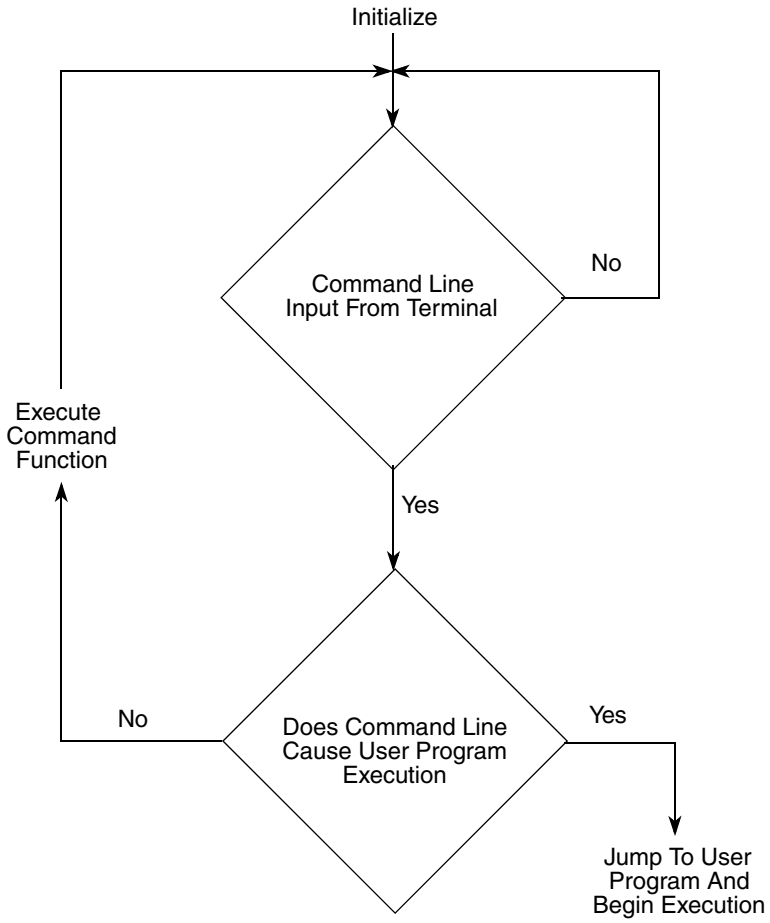


Figure 2-1. Flow Diagram of dBUG Operational Mode

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 the ColdFire website (http://www.freescale.com/coldfire) for the complete initialization code sequence.

After initialization, the terminal will display:

```
Hard Reset
DRAM Size: 8M
```

```
Copyright 1995-2002 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5249 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 1.9, “System Power-up and Initial Operation.”

Other means can be used to re-initialize the M5249C3 Computer Board firmware. These means are discussed in the following paragraphs.

2.2.2.1 Hard RESET Button

Hard RESET (S1) is the red button. Depressing this button causes all processes to terminate, resets the MCF5249 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 the RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5249) 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 MCF5249 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 is 19200 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

Table 2-1. dBUG Command Summary (continued)

Mnemonic	Syntax	Description
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
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> -Clls>	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:

```
bs      40000 80000 1234
```

This reads the 16-bit word located at 0x00040000 and compares it against the 16-bit value 0x1234. If no match is found, 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.l    40000 80000 ABCD
```

This reads the 32-bit word located at 0x00040000 and compares it against the 32-bit value 0x0000ABCD. If no match is found, 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 MCF5xxx. 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 MCF5xxx are CS, DMA0, DMA1, DMA2, DMA3, DRAMC, PP, MBUS, SIM, TIMER1, TIMER2, UART0 and UART1. Refer to the MCF5407 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 MCF5xxx. 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 MCF5xxx are CS, DMA0, DMA1, DMA2, DMA3, DRAMC, PP, MBUS, SIM, TIMER1, TIMER2, UART0 and UART1. Refer to the MCF5407 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 M5249C3 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      Ethernet controller
CS2      not in use
CS3      not in use
```

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, with no flow control.
- 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:52:49: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 M5249C3 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:

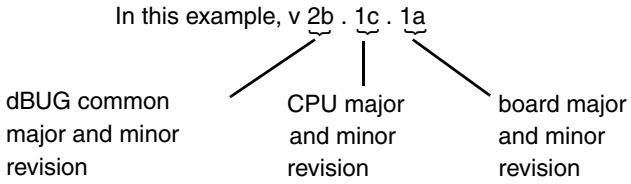
```
upuser 1000
```

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 M5249C3 board hardware. With the description given here and the schematic diagrams in Appendix C, "Schematics," the user can gain a good understanding of the board's design. In this manual, an active low signal is indicated by a bar over the signal name.

3.1 The Processor and Support Logic

This part of the chapter discusses the CPU and general support logic on the M5249C3 board.

3.1.1 Processor

The microprocessor used on the M5249C3 is the highly integrated ColdFire MCF5249, 32-bit processor. The MCF5249 implements a ColdFire Version 2 core with 8-KByte instruction cache, two UART channels, two timers, 96-KBytes of SRAM, a QSPI (Queued Serial Peripheral Interface) module, an I^2C module, 4x I^2S modules, an IDE module, a Flash memory stick interface, 64 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 16-bit wide data bus, D[31:16]. The chip can address 64-MBytes of memory space using a 25-bit wide address bus and internal chip-select logic. All the processor's signals are available through the expansion connectors (J4 and J5). Refer to Section 3.3.1, "Expansion Connectors - J4 and J5," for their pin assignments.

The MCF5249 processor has the capability to support both an IEEE JTAG-compatible port and a BDM debug port. These ports are multiplexed 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 (J2). The processor also has the logic to generate up to four (4) chip selects, $\overline{CS0}$ to $\overline{CS3}$, and support for 2 banks of SDRAM (included on the evaluation board as 8-Mbytes in total configured as 4Mx16). The $\overline{SDRAM\_CS1}$ 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 $\overline{RESET}$ which causes the MCF5249 to reset. Reset is also triggered by the reset switch (S1) which resets the entire processor/system.

A hard reset and voltage sense controller (U9) is used to produce an active low power-on RESET signal. The reset switch S1 is fed into U9 which generates the signal which is fed to the MCF5249 reset, $\overline{RESET}$.

The $\overline{\text{RESET}}$ signal is an open collector signal and so can be wire OR'ed with other reset signals from additional peripherals.

dBUG configures the MCF5249 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, the Bus Monitor is enabled, and the 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 $\overline{\text{HIZ}}$ signal forces all output drivers to a high-impedance state. On the M5249C3 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. $\overline{\text{HIZ}}$ is also available to the user on connector (J5).

3.1.4 Clock Circuitry

The M5249C3 board uses a 11.2896MHz crystal (X1 on the schematics) to provide the clock to the clock driver chip (U10). The clock driver provides a buffered clock for the MCF5249 processor (U2). In addition to the 11.2896MHz crystal, there is also a 25MHz oscillator (U3) which feeds the Ethernet chip (U4).

3.1.5 Watchdog Timer

The duration of the Watchdog is selected by the SWT[1:0] bits in the System Protection and Control Register (SYPCR), SWT[1:0] = 11 gives a maximum timeout period of 2^{28}/System frequency. The dBUG monitor initialises these bits with the value 0x11, which provides the maximum time-out period, but dBUG does **NOT** enable the watchdog timer via the SYPCR register SWE bit.

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, its interrupt should be set up as an autovector interrupt which directs the processor to a predefined entry in the exception table (refer to the MCF5249 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 MCF5249 microprocessor has eight external interrupt request lines $\overline{\text{INT}}$[7:0], all of which are multiplexed with other functions. The interrupt controller is capable of providing up to 32 interrupt sources. These sources are:

- External interrupt signals $\overline{\text{INT}}$[7:0]
- Software watchdog timer module
- Two general purpose timer modules
- UART module
- I^2C module
- Audio interface modules
- DMA module
- QSPI module

All external interrupt inputs are edge sensitive. The active level is programmable. An interrupt request must be held valid until an IACK cycle starts to guarantee correct processing. 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 M5249C3 hardware uses $\overline{\text{INT7}}$ to support the ABORT function using the ABORT switch (S2). This switch is used to force an interrupt (level 7, priority 3) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2, "Using the Monitor/Debug Firmware," for more information on ABORT). Since the ABORT switch is not capable of generating a vector in response to a level seven interrupt acknowledge from the processor, the dBUG programs this interrupt request for autovector mode.

Refer to MCF5249 User's Manual for more information about the interrupt controller.

3.1.7 Internal SRAM

The MCF5249 processor has 96-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 32-KByte boundary.

3.1.8 The MCF5249 Registers and Memory Map

The memory and I/O resources of the M5249C3 hardware are divided into two groups, MCF5249 internal and external resources. All the I/O registers are memory mapped.

The MCF5249 processor has built in logic and up to four chip-select pins ($\overline{\text{CS}}$[3:0]) which are used to enable external memory and I/O devices. In addition there are $\overline{\text{SDRAS}}$ and $\overline{\text{SDCAS}}$ lines available for controlling SDRAMs. There are registers to specify the address range, type of access and the method of $\overline{\text{TA}}$ generation for each chip-select. These registers are programmed by the dBUG monitor to map the external memory and I/O devices.

The M5249C3 uses the following signals to select external peripherals:

- $\overline{CS0}$ to enable the Flash ROM (refer to Section 3.1.13, “Flash ROM”)
- $\overline{SDRAS}$, $\overline{SDCAS}$ and $\overline{SDRAM\_CS1}$ to enable the SDRAM (refer to Section 3.1.12, “SDRAM”)
- $\overline{CS1}$ for the Ethernet controller

The chip select mechanism of the MCF5249 processor allows the memory mapping to be defined for the required memory space (User/Supervisor, Program/Data spaces).

All of the MCF5249 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 MBAR1 register is mapped to 0x1000_0000 and MBAR2 mapped to 0x8000_0000 by the dBUG monitor. For a complete map of these registers refer to the *MCF5249 User's Manual*.

The M5249C3 board has 8-MBytes of SDRAM installed. Refer to Section 3.1.12, “SDRAM,” for a discussion of the SDRAM on the board. The dBUG ROM monitor is programmed in one AMD Am29LV160DB-90 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, “Flash ROM.”

Table 3-1 shows the M5249C3 memory map.

Table 3-1. M5249C3 Memory Map

Address Range	Signal and Device	Memory Access Time
$0000_0000 – $0002_0000	SDRAM space for dBug ROM monitor use	Refer to manufacturer spec.
$0002_0000 – $007F_FFFF	SDRAM space	Refer to manufacturer spec.
$1000_0000 – $1000_03FF	System Integration Module (SIM) registers	Internal access
$1000_0000 – $1000_0054	MBAR—Module Base Addres Reg.	Refer to MCF5249UM SIM section
$2000_0000 – $2000_FFFF	SRAM1	Internal access (1 cycle)
$2001_0000 – $2001_7FFF	SRAM0	Internal access (1 cycle)
$3000_0000 – $3007_FFFF	$\overline{CS1}$, External Ethernet controller	8-7-7-7
$8000_0000 – $BFFF_FFFF	MBAR2—Module Base Address Reg. 2	Refer to MCF5249UM SIM section
$FFE0_0000 – $FFFF_FFFF	$\overline{CS0}$, 2M Flash ROM	8-7-7-7

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 & $00000004 (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 $00000000. The MCF5249 processor chip-select zero ($\overline{CS0}$) responds to any accesses after reset until the CSMR0 is written. Since $\overline{CS0}$ (the global chip select) is connected to the Flash ROM (U6), the Flash ROM initially appears at address $00000000 which provides the initial stack pointer and program counter (the first eight 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 $\overline{CSx}$ with the other control signals. The processor then waits for a transfer acknowledgment ($\overline{TA}$) either from within (Auto acknowledge - AA mode) or from the externally addressed device before it can complete the bus cycle. $\overline{TA}$ 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) like the Ethernet controller (U4). The MCF5249 processor, as part of the chip-select logic, has a built-in mechanism to generate $\overline{TA}$ for all external devices which do not have the capability to generate this signal. For example the Flash ROM cannot generate a $\overline{TA}$.signal. The chip-select logic is programmed by the dBUG ROM Monitor to generate $\overline{TA}$ internally after a pre-programmed number of wait states. In order to support future expansion of the M5249C3 board, the $\overline{TA}$ input of the processor is also connected to the Processor Expansion Bus (J5, pin 66). This allows any expansion boards to assert this line to provide a $\overline{TA}$ 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 $\overline{TA}$ 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 MCF5249 processor can be programmed to generate an internal $\overline{TA}$ 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 M5249C3 has one 64-MBit device on the board, in a 16-bit wide data bus configuration. The MCF5249 processor supports one bank of SDRAM, which on this board is represented by SDRAM device, (U7). These are connected to the MCF5249 to provide 4Mx16 of memory.

3.1.13 Flash ROM

There is one 2-MByte Flash ROM on the M5249C3, (U6).

The board is shipped with one AMD Am29LV160DB, 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 12.

The MCF5249 chip-select logic can be programmed to generate the $\overline{TA}$ for $\overline{CS0}$ 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 JP12 Jumper and the User's Program

Jumper 12 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 (JP12) 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 program counter (PC) to 0x7FE00400 and run.)
4. In the ROM Monitor, execute the 'upuser' command.
5. Move jumper (JP12) to pin 2 connected to pin 3 and push the reset button (S1). User code should now be running from reset/POR.

3.2 Serial Communication Channels

The M5249C3 offers two serial communications channels. They are discussed in this section.

3.2.1 MCF5249 UARTs

The MCF5249 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 (J5). The signals of UART0 and UART1 are also passed through the RS-232 driver/receivers (U13) & (U14) and are available on DB-9 connectors (P3) and (P4). Refer to the MCF5249 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 is very similar, but not identical, to the QSPI portion of the QSM (Queued Serial Module) implemented in the MC68332 processor.

- 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 274.5-Kbps to 17.5-Mbps at 140MHz.
- Programmable delays before and after transfers
- Programmable clock phase and polarity
- Supports wrap-around mode for continuous transfers

Please see the MCF5249 Users Manual for more detail. The QSPI signals from the MCF5249 device are brought out to expansion connector (J4). Some of these signals are multiplexed with other functions.

3.2.3 General Purpose I/O Pins

The MCF5249 offers 64-bits of general-purpose I/O of which 11 are dedicated general purpose inputs and 10 are dedicated general purpose outputs. Eight of the GPIO lines are also available as edge sensitive interrupt inputs. The functions of all I/O pins are individually programmable, since they are multiplexed with other pin functions. All general-purpose I/O pins (unless dedicated as either only input or output) 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.

Control registers are provided for each pin to select the function (GPIO or peripheral pin) assigned to each pin individually. Pins can have from 1 to 4 functions including GPIO.

Please see the MCF5249 User's manual for more detail. All of these signals are brought out to expansion connectors (J4) & (J5).

3.2.4 Ethernet Controller

The MCF5249 device has an Ethernet controller, SMSC LAN91C111, memory mapped into the address space using $\overline{\text{CS1}}$. The Ethernet controller performs both the MAC & PHY functions and allows 10BaseT or 100BaseT operation. This controller is clocked from a standalone 25MHz oscillator independent of the CPU clock. The interface between the MCF5249 and the SMSC LAN91C111 is therefore asynchronous.

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

- Full integration and compliance with the IEEE 802.3/802.3u 100Base-TX/10Base-T physical layer standards
- Dual speed - 10/100Mbps
- 8Kbytes of internal Rx & Tx FIFO buffers
- Burst transfers are supported
- Single 25MHz operation for both the MAC & PHY
- On-chip wave shaping
- On-chip adaptive equaliser
- Baseline wander correction

For more details see the LAN91C111 Users manual at http://www.smsc.com.

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.

3.2.5 Audio Module

The MCF5249 processor's audio module includes the following features:

- Support for reception and transmission of digital audio over serial interfaces IIS/EIAJ and digital interface IEC958
- 4x IIS/EIAJ interfaces
- 2x IEC958 receivers (4x multiplexed inputs)
- 1x IEC958 transmitter - two outputs - one with professional subcoding, one with consumer subcoding
- Allows direct transmission of received audio to an audio transmitter without CPU intervention.
- IEC958 receivers and transmitter support main audio, plus handling of IEC958 C, U and V sub-channels
- Frequency measurement block - precise measurement of the incoming sample frequency

All the Audio signals are brought out to expansion connectors (J4) & (J5). For further details please refer to the MCF5249 User's manual.

3.2.6 I^2C Module

The MCF5249 processor's I^2C module includes the following features:

- Compatibility with the I^2C bus standard
- Multimaster operation
- Software programmable for one of 64 different clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte by byte data transfer
- Arbitration-lost interrupt with auto mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation and detection
- Repeated start signal generation
- Acknowledge bit generation and detection
- Bus busy detection

3.2.7 Analog to Digital Converter (ADC) Module

The MCF5249 processor's ADC module includes the following features:

- Sigma-Delta based ADC with 12-bit resolution

- Four multiplexed inputs - EBUIN3_ADIN0_GPI38, EBUIN4_ADIN1_GPI39, RXD2_ADIN2_GPI28 and CTS2_ADIN3_GPI31
- The digital portion of the ADC is on-chip, an analog comparator must be sourced externally
- Single output - TOUT1_ADOUT_GPO35, provides the reference voltage which requires an external comparator (resistor/capacitor circuit).
- Software interrupt provided when the ADC measurement is complete

3.2.8 Flash Memory Card/IDE Interface Module

The MCF5249 processor's Flash Memory Card/IDE module includes the following features:

- TBA

3.3 Connectors and Expansion Bus

There are 2 expansion connectors on the M5249C3 (J4 and J5) which are used to connect the board to external I/O devices and/or expansion boards.

3.3.1 Expansion Connectors - J4 and J5

Table 3-2 shows pin assignments for the (J4) connector.

Table 3-2. J4 Connector Pin Assignment

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	+1.8V	2	+1.8V	61	$\overline{CS1}$	62	A22
3	+1.8V	4	GND	63	NC	64	A23
5	NC	6	A1	65	SCL0/QSPICLK	66	GND
7	NC	8	A2	67	GND	68	A24
9	GND	10	A3	69	SDA0/QSPIDIN	70	NC
11	$\overline{CS0}$	12	+1.8V	71	BCLKE	72	CMDSDIO2/GPIO34
13	NC	14	A4	73	BCLK	74	+3.3V
15	+3.3V	16	A5	75	+3.3V	76	NC
17	D16	18	A6	77	SDUDQM	78	SCLKOUT/GPIO15
19	D17	20	GND	79	$\overline{SDRAS}$	80	EF/GPIO19
21	GND	22	A7	81	$\overline{SDRAM\_CS1}$	82	GND
23	D18	24	A8	83	GND	84	SDATA0_SDI01/GPIO54
25	D19	26	+3.3V	85	$\overline{SDWE}$	86	SDATA_BS2/RSTO
27	+3.3V	28	A9	87	$\overline{SDCAS}$	88	BUFENB1/GPIO57
29	D20	30	A10	89	SDLDQM	90	+3.3V
31	D21	32	A11	91	+3.3V	92	SDATA1_BS1/GPIO9

Table 3-2. J4 Connector Pin Assignment (continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
33	D22	34	GND	93	SDATA3/GPIO56	94	QSPI_CS0/GPIO29
35	GND	36	A12	95	+5V	96	QSPI_CS1/GPIO24
37	D23	38	A13	97	GND	98	GND
39	D24	40	A14	99	$\overline{\text{SDRAM\_CS2}}$	100	QSPI_DOUT/GPIO26
41	D25	42	+3.3V	101	GPIO5	102	QSPI_CS2/GPIO21
43	+3.3V	44	A15	103	GPIO6	104	QSPI_CS3/GPIO22
45	D26	46	A16	105	+3.3V	106	+1.8V
47	D27	48	A17	107	GND	108	GND
49	D28	50	GND	109	GND	110	GND
51	GND	52	A18	111	+1.8V	112	+1.8V
53	D29	54	A19	113	+5V	114	+5V
55	D30	56	A20	115	+5V	116	+5V
57	D31	58	+3.3V	117	GND	118	GND
59	+3.3V	60	A21	119	GND	120	GND

Table 3-3 shows the pin assignments of the J5 connector.

Table 3-3. J5 Connector pin assignment

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	+1.8V	2	+1.8V	61	R/$\overline{\text{W}}$	62	GND
3	GND	4	GND	63	XTRIM/GPO38	64	GND
5	TXD0/GPO27	6	DDATA0	65	CL11/GPO39	66	$\overline{\text{TA}}$
7	RXD0/GPI27	8	DDATA1	67	GND	68	$\overline{\text{RESET}}$
9	LRCK3/GPIO45	10	+3.3V	69	BUFENB2/GPIO17	70	+1.8V
11	+1.8V	12	DDATA2	71	CRIN	72	SDATAI1
13	SRE/GPIO11	14	DDATA3	73	SCLK4/GPIO50	74	GND
15	$\overline{\text{TA\_IN}}$	16	PST0	75	+3.3V	76	SDATAI3/GPI41
17	RXD1/GPI28	18	GND	77	SCLK1	78	+1.8V
19	GND	20	PST1	79	SDATAO1/GPIO25	80	SDATAI4/GPI42
21	SCLK3/GPIO49	22	PST2	81	LRCK1	82	GND
23	$\overline{\text{RTS0}}$/GPO30	24	PST3	83	GND	84	LRCK4/GPIO46
25	TXD1/GPO28	26	+1.8V	85	SDATAO2	86	+3.3V
27	+3.3V	28	DSO	87	LRCK2/GPIO44	88	TIN1/GPIO23
29	$\overline{\text{RTS1}}$/GPO31	30	DSI	89	SCL1/GPIO3	90	EBUIN1/GPI36

Table 3-3. J5 Connector pin assignment (continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
31	TOUT0/GPO33	32	GND	91	+3.3V	92	EBUIN2/GPI37
33	NC	34	DSCLK	93	$\overline{OE}$	94	EBUIN3/GPI38
35	GND	36	$\overline{BKPT}$	95	SDA1/GPIO55	96	GND
37	SWE/GPIO12	38	+3.3V	97	TOUT1/GPO35	98	SCLK2/GPIO48
39	NC	40	EBUIN4/GPI39	99	GND	100	CL16/GPO42
41	PSTCLK	42	$\overline{CTS0}$/GPI30	101	EBUOUT2/GPO37	102	NC
43	+1.8V	44	GND	103	CFLG/GPIO18	104	NC
45	SFSY/GPIO52	46	$\overline{CTS1}$/GPI31	105	GND	106	+1.8V
47	RCK/GPIO51	48	GND	107	+1.8V	108	+1.8V
49	SUBR/GPIO53	50	+1.8V	109	+3.3V	110	+3.3V
51	GND	52	TIN0/GPI33	111	EBUOUT1/GPO36	112	NC
53	IDEDIOL/GPIO13	54	GND	113	+5V	114	+5V
55	IDEIORDY/GPIO16	56	HI-Z	115	+5V	116	+5V
57	IDEDIOW/GPIO14	58	TCK	117	GND	118	GND
59	+3.3V	60	+1.8V	119	GND	120	GND

3.3.2 The Debug Connector J2

The MCF5249 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 (J2). Figure 3-1 shows the (J2) Connector pin assignment.

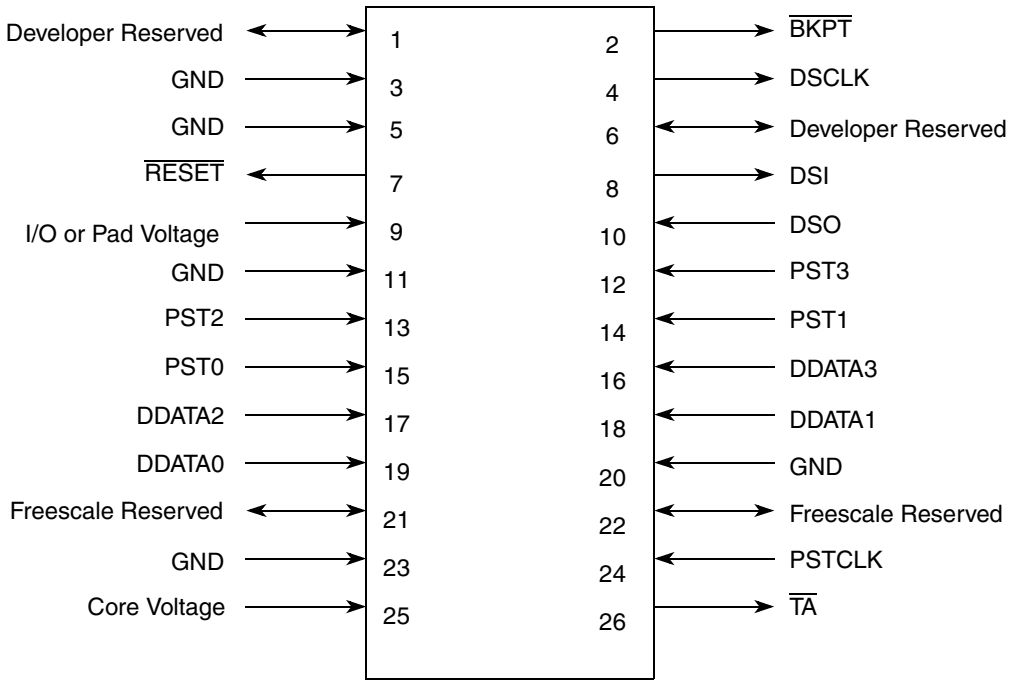


Figure 3-1. The J2 Connector pin assignment

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 (NOTE: this requires a TFTP server to be running on the host attached to the board). 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:52:49: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 simple logic equations for the memory mapped interface to the Ethernet controller U4 (sheet 5 of the schematics). The first equation inverts the interrupt signal from the LAN91C111 and generates an IRQ6 signal to the MCF5249. The next equation inverts the R/W signal from the MCF5249 to create the W/R signal required by the Ethernet controller. The next two equations create the positive read (RD) and write (WR) control signals required by the LAN91C111 using the R/W and CS1 signals from the MCF5249. Finally the reset logic of the LAN91C111 requires a positive logic RESET signal which is a simple inversion of RESET signal applied to the MCF5249 from either the BDM port, reset switch (S1) or power on reset (POR).

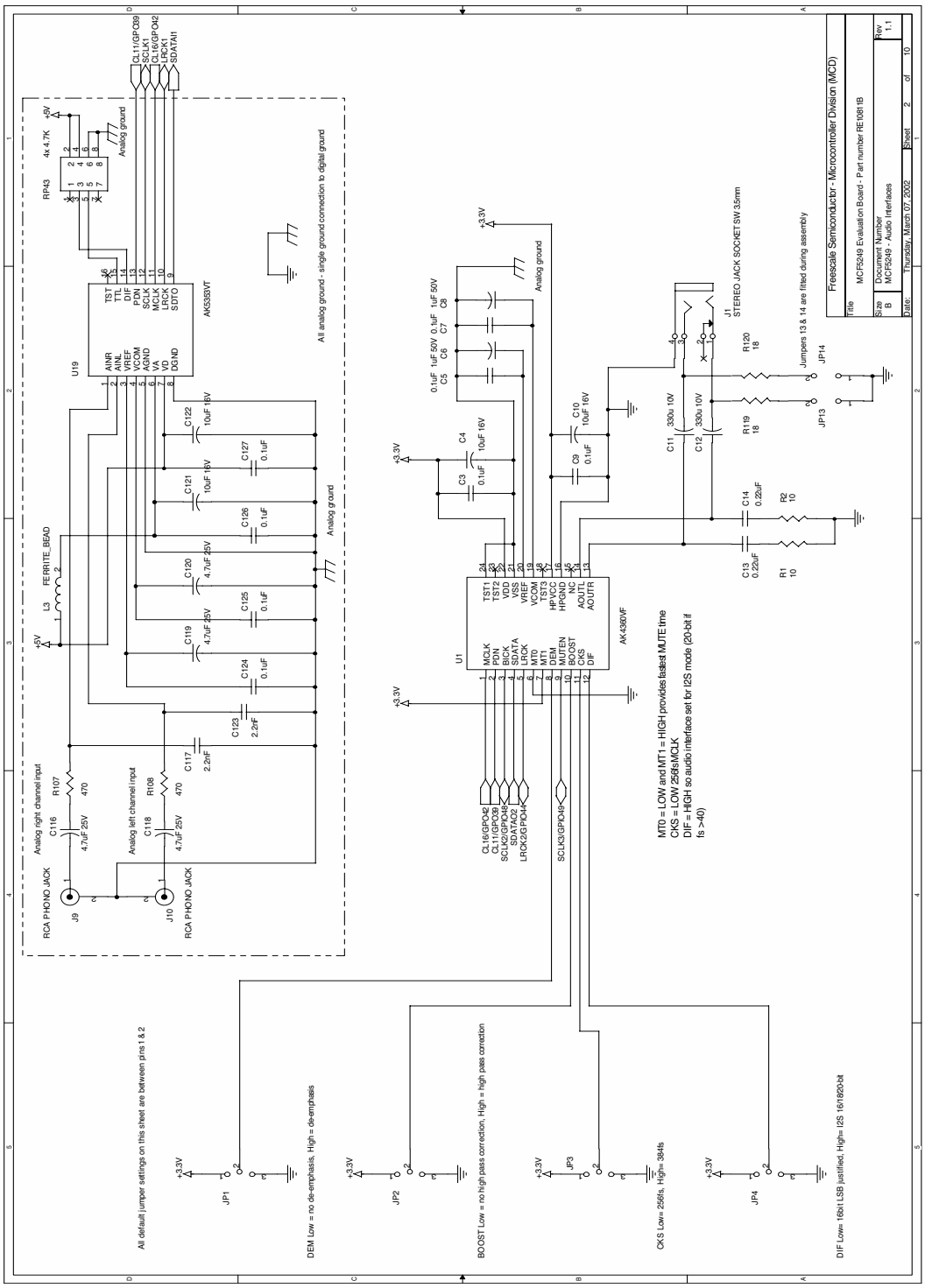
Important Note the " symbol at the start of some of the signal definitions comments out the signal. Intially an asynchronous TA terminated interface was considered for the ethernet controller, which is why these signals have been brought out to the PAL. After initial debug of the board a simple asynchronous auto-acknowledge interface with wait states set up in the chip select control register sufficed.

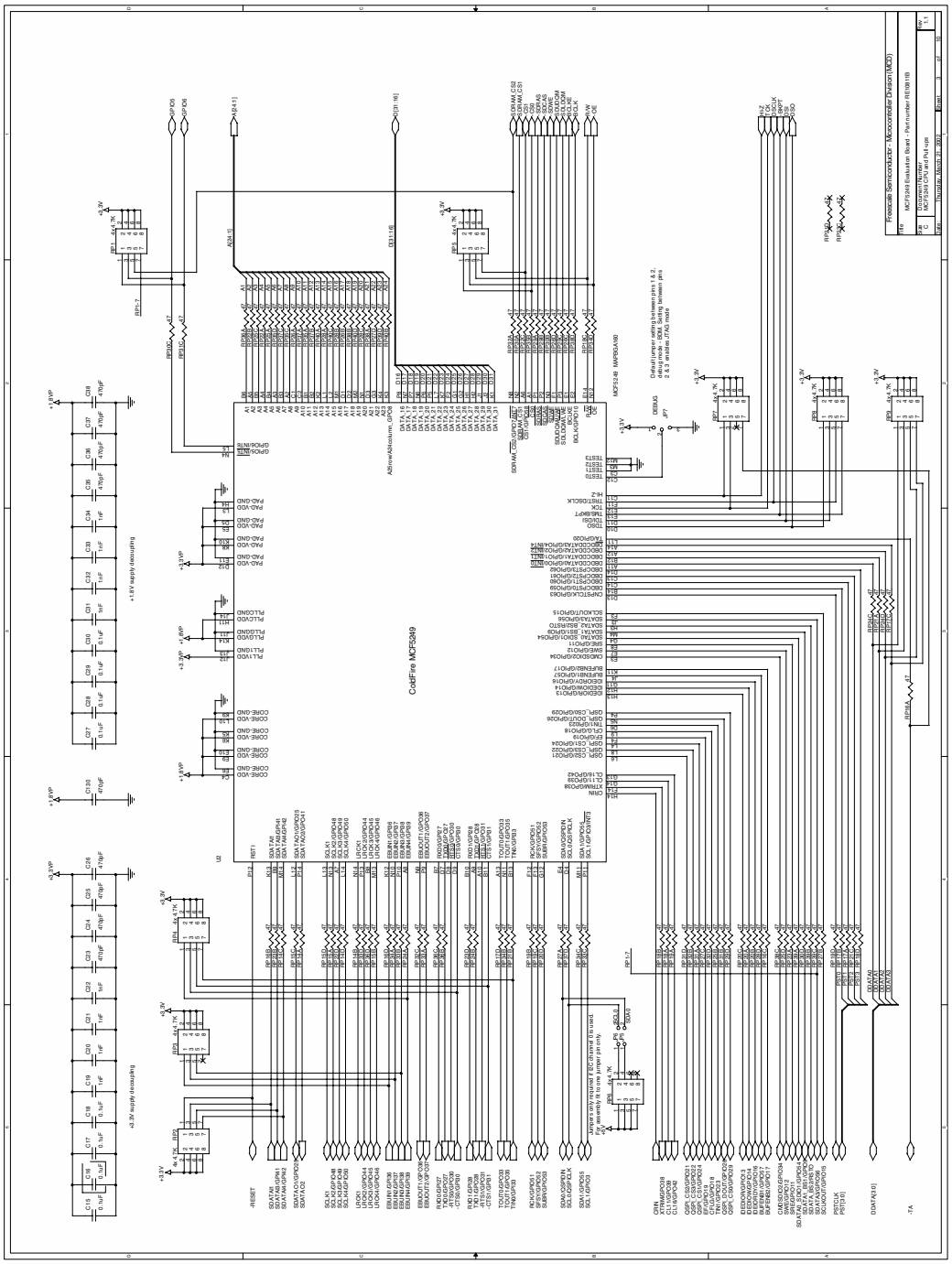
```
module EthernetIF
title 'Ethernet Interface logic for the M5249C3 board'
"March 2 2002 Revision 1.0 of the code"
"EthernetIF device 'ispLSI22LV10';
;"*****************************************************"
;"This abel file contains the code to interface the SMSC"
;"10/100baseT Ethernet controller LAN91C111-NE to the"
;"MCF5249 ColdFire processor"
;"It was targeted to Lattice ispLSI 22LV10 PAL"
;"CS:380E "
;"*****************************************************"


;"*****************************************************"
;"Declaration Section                                  "
;"*****************************************************"


" Inputs
BCLK            PIN     2;              " Bus clock input to the 22V10 from MCF5249
RESET           PIN     3;              " /RESET Input from MCF5249
!CS1            PIN     4;              " /CS1 Chip Select 1 input from MCF5249
R_W             PIN     5;              " Read not Write input from MCF5249
"!TA_IN         PIN     6;              "" /TA Transfer Ack. input from expansion
                                        connector
"!OE            PIN     7;              "" /OE Output Enable input from MCF5249
"!SRDY          PIN     9;              "" /SRDY Synchronous Ready input from LAN91C111
INTR0           PIN     10;             " Interrupt 0 input from LAN91C111
"ARDY           PIN     11;             "" ARDY Asynchronous ready input from LAN91C111

" Outputs
"!ANRDYREG      PIN     17 ISTYPE 'reg';  "" Registered ARDY
```

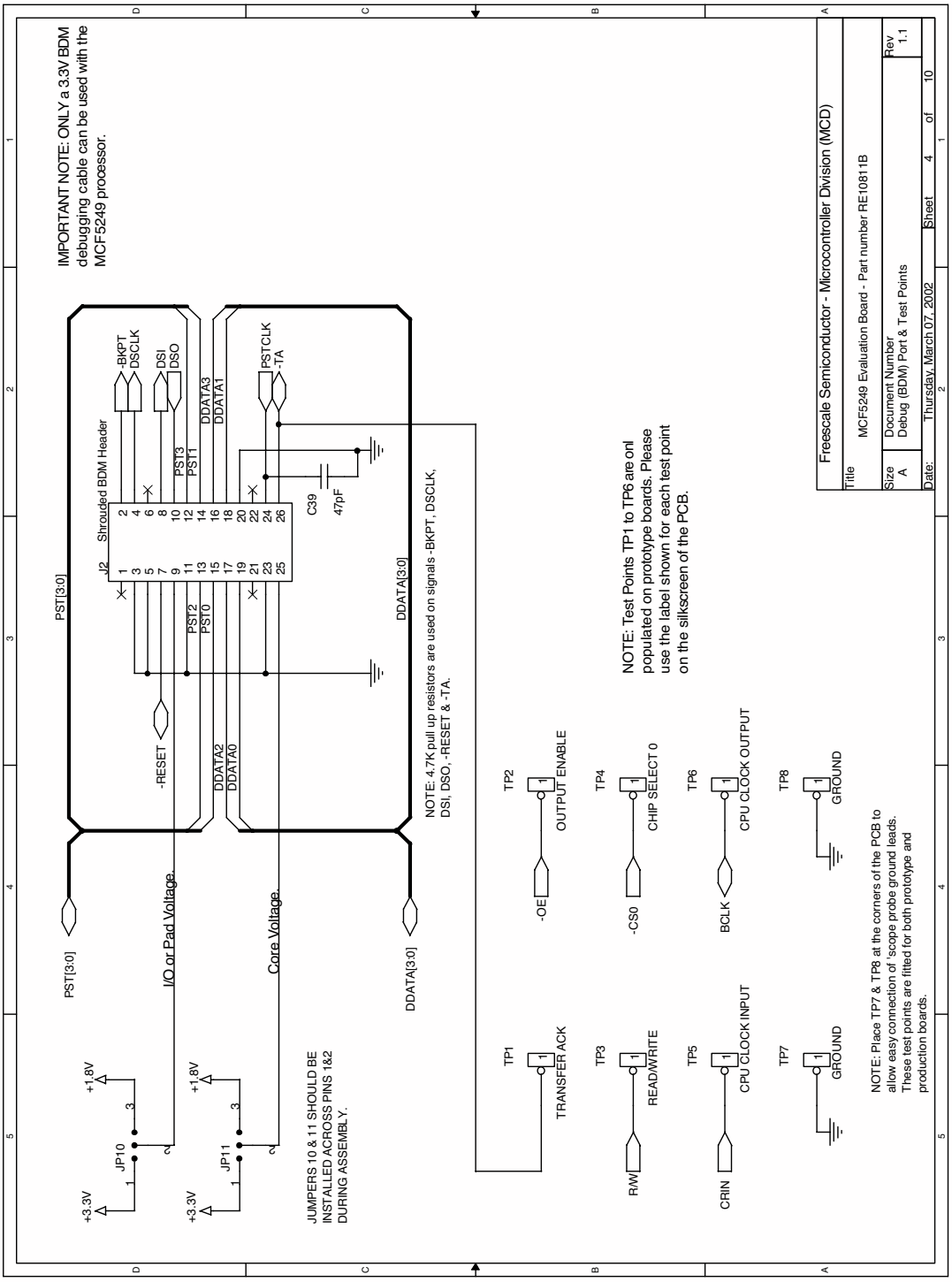
```
"!ANRDYDLY        PIN      18 ISTYPE 'reg';  "" Delayed, registered ARDY
!WR               PIN      19;               " /WR Write Output to LAN91C111
!RD               PIN      20;               " /RD Read Output to LAN91C111
"!RDYRTN          PIN      21;               "" /RDYRTN Ready Return Output to LAN91C111
"!ADS             PIN      23;               "" /ADS Address Data Strobe Output to LAN91C111
W_R               PIN      24;               " Write not Read Output to LAN91C111
RESET_OUT         PIN      25;               " RESET Output to LAN91C111
!IRQ6             PIN      26;               " Interrupt Request 6 Output
"!TA              PIN      27;               "" /TA Output to the MCF5249


;"***************************************************"
;"Equations Section                                   "
;"***************************************************"
equations

" Invert interrupt from Ethernet controller to GPIO6/IRQ6
IRQ6 = !INTR0;

" Generate inverted R_W
W_R = !R_W;

" Generate Read and Write to controller for asynchronous access
RD = CS1 & R_W;
WR = CS1 & !R_W;

" Generate inverted reset
RESET_OUT = !RESET ;


end
```

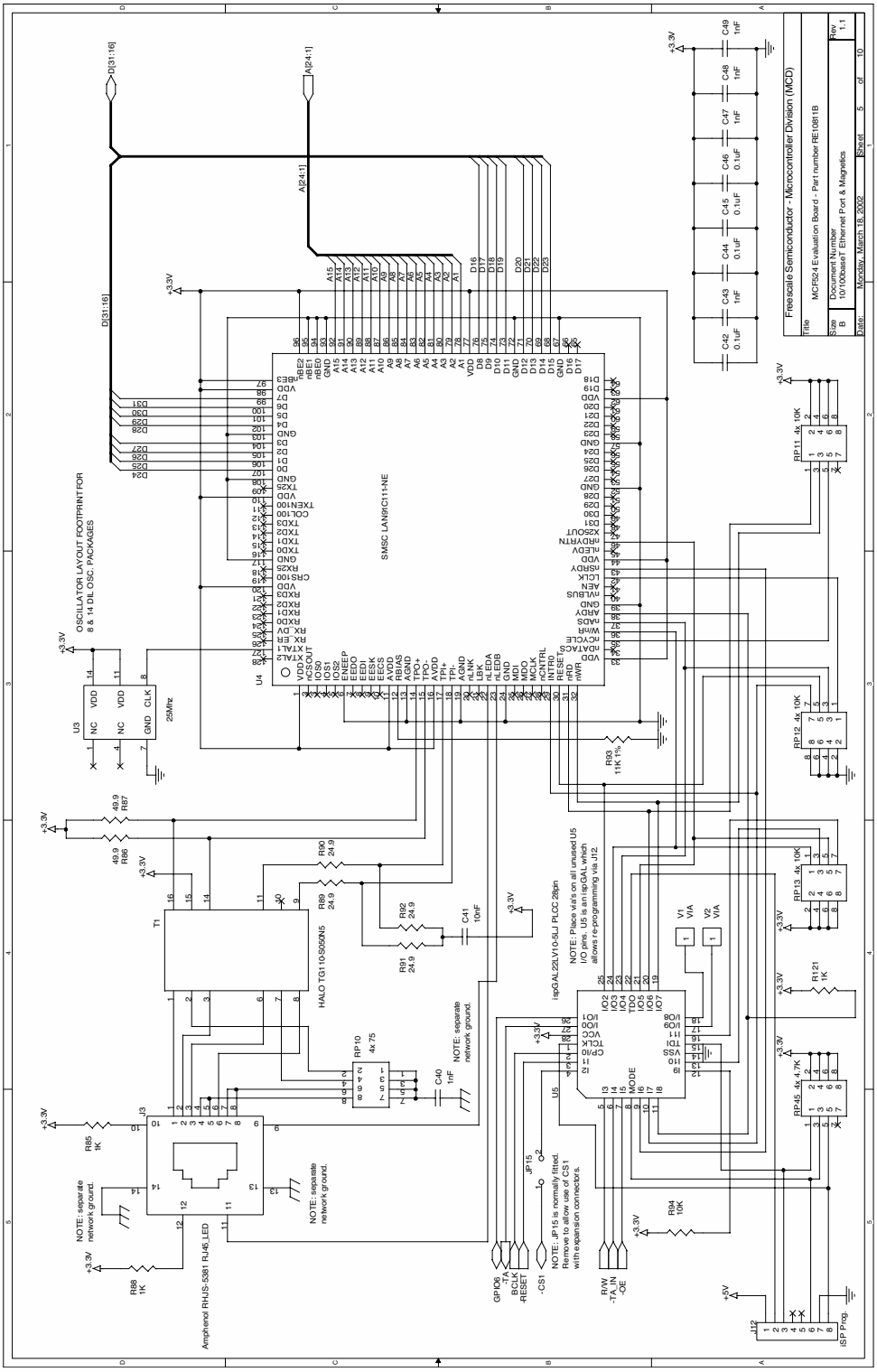
Appendix C
Schematics

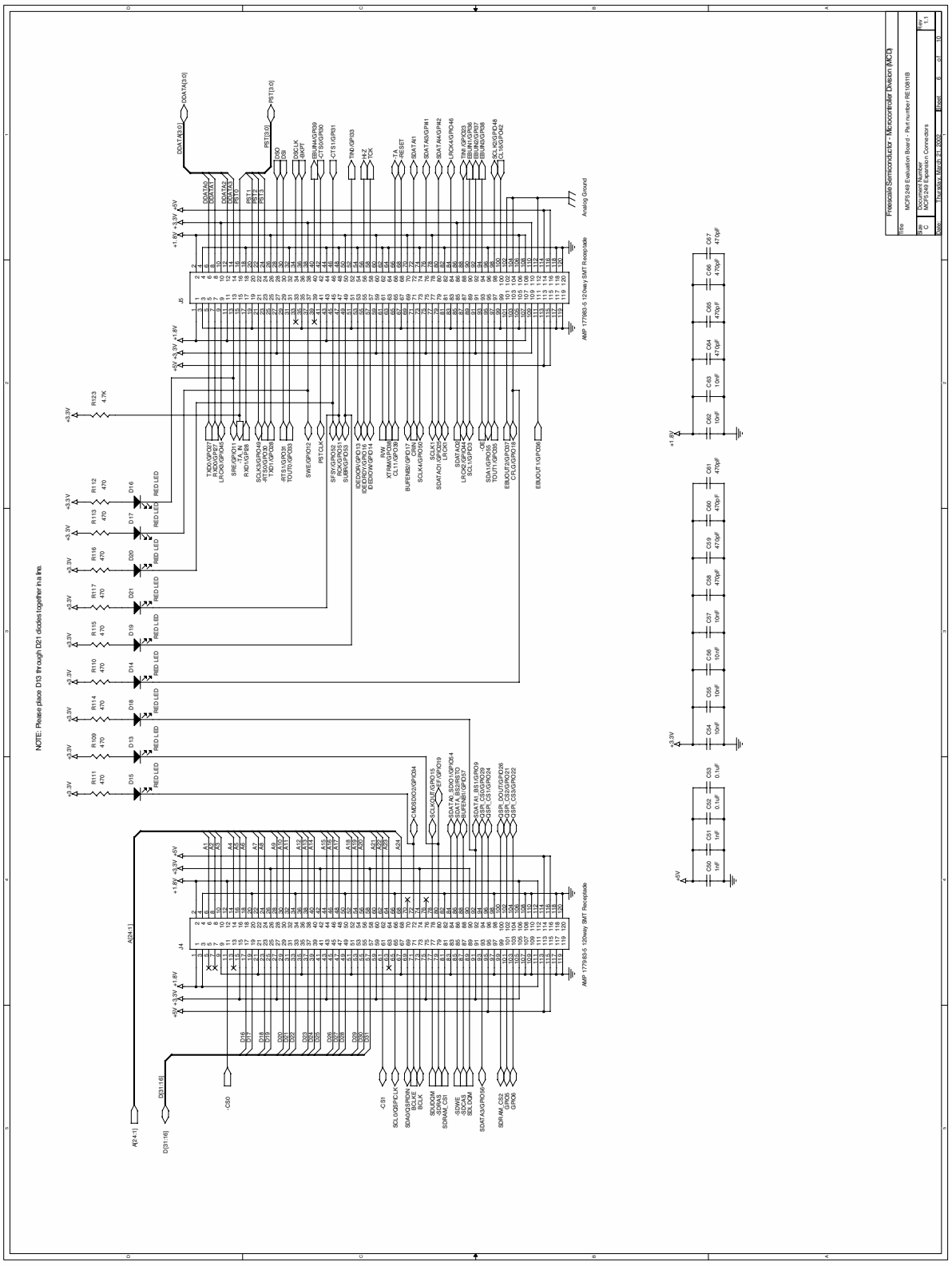

Freescale Semiconductor - Microcontroller Division (MCD)
MCF5249 Evaluation Board - Part number RE1061B
Document Number:
Hardware Block Diagram
Rev. 1.1
Thursday, March 07, 2002
Sheet 1 of 10
ColdFire MCF5249
Sheet 7
Memory
Sheet 10
DE
Sheet 2
Audio
Sheet 5
Ethernet
Sheet 4
Debug_Test Ports
Sheet 3
OPU
Sheet 9
Serial_I2C_QSPI
Sheet 8
PSU_Osc
Sheet 6
Expansion Connectors

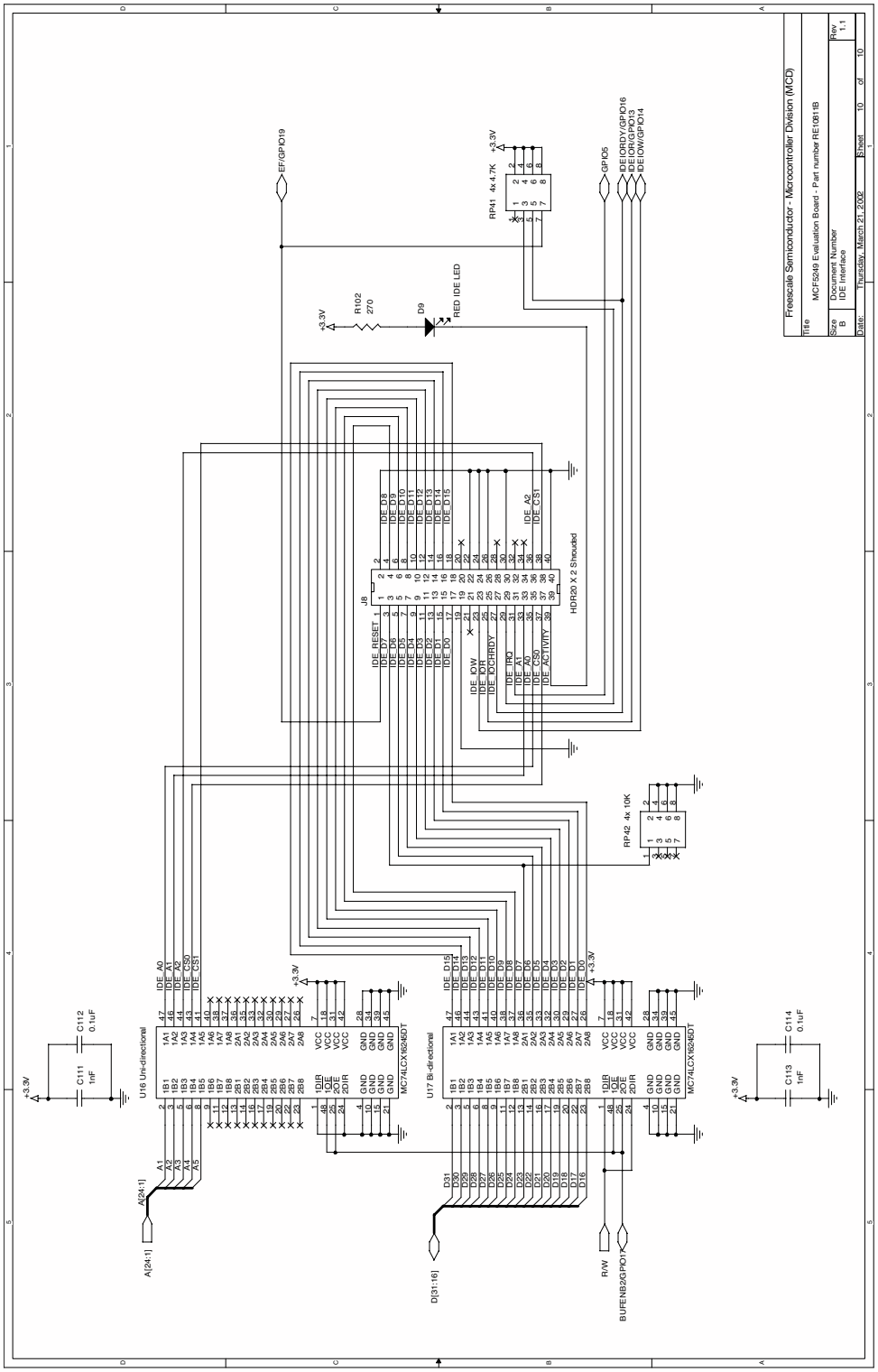

Freescale Semiconductor - Microcontroller Division (MCD)
MCF5249 Evaluation Board - Part number RE1081B
MCF5249 - Audio Interfaces
Thursday, March 07, 2002
All analog ground - single ground connection to digital ground
Analog ground
Analog right channel input
Analog left channel input
RCA PHONO JACK
AK4360VF
AK5353VT
STEREO JACK SOCKET SW 3.5mm
Jumpers 13 & 14 are fitted during assembly
All default jumper settings on this sheet are between pins 1 & 2
DEM Low = no de-emphasis, High = de-emphasis
BOOST Low = no high pass correction, High = high pass correction
CKS Low= 256fs, High= 384fs
DIF Low= 16bit LSB justified, High= I2S 16/18/20bit
MT0 = LOW and MT1 = HIGH provides fastest MUTE time
CKS = LOW 256fsMCLK
DIF = HIGH so audio interface set for I2S mode (20-bit if fs >40)

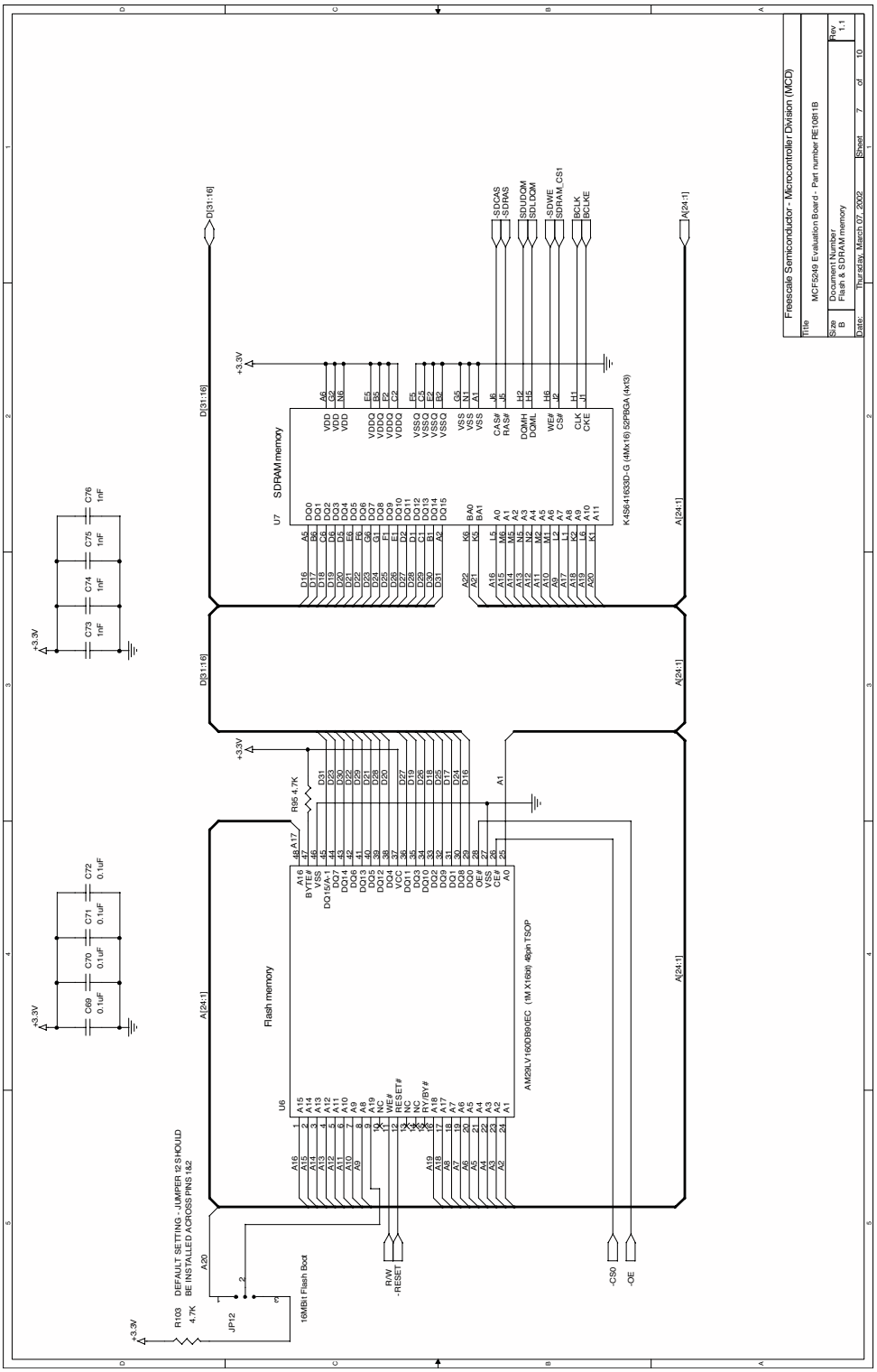


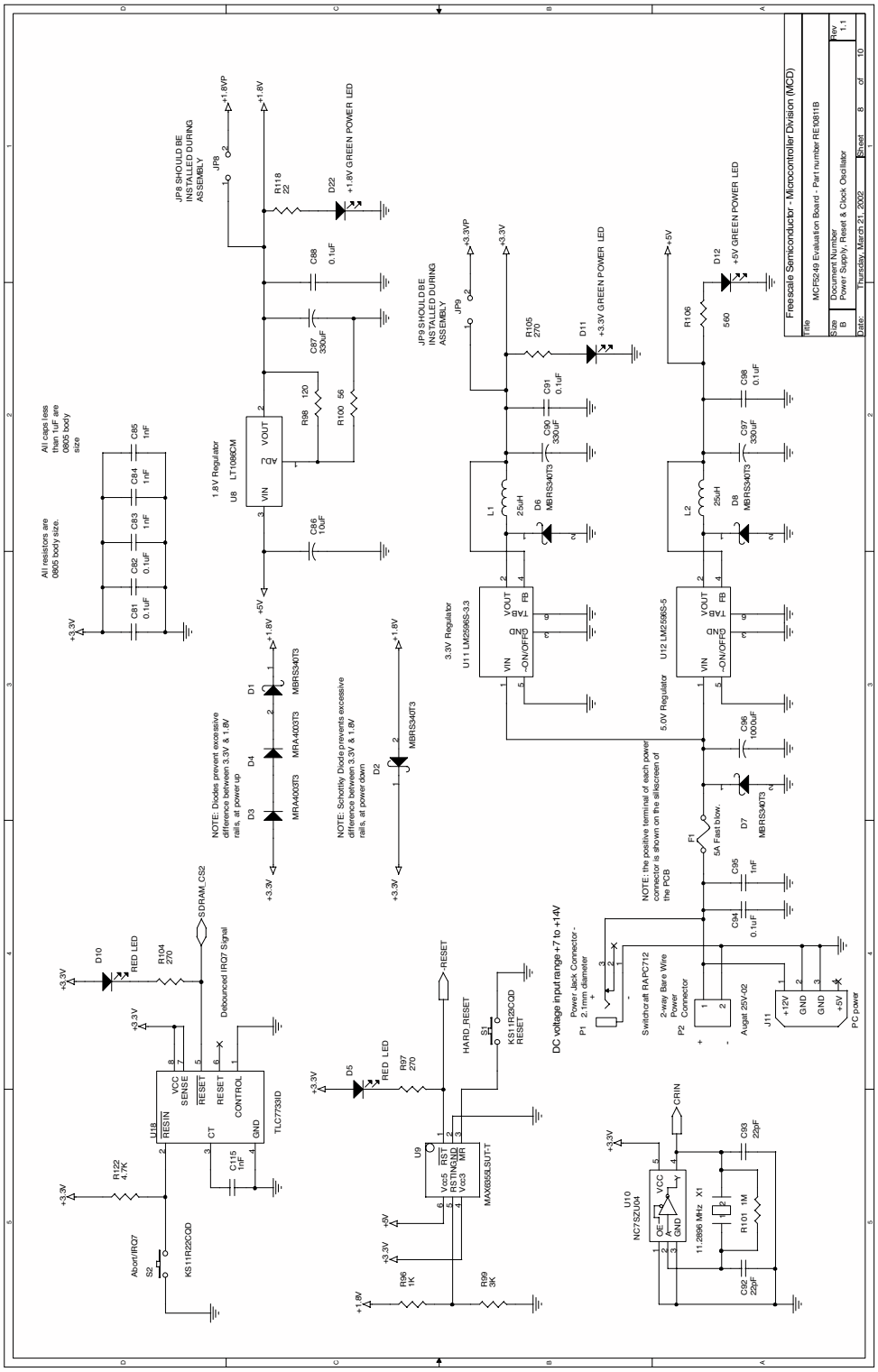

IMPORTANT NOTE: ONLY a 3.3V BDM debugging cable can be used with the MCF5249 processor.
Shrouded BDM Header
J2
-BKPT
DSCLK
DSI
DSO
PSTCLK
-TA
PST3
PST1
DDATA3
DDATA1
PST2
PST0
DDATA2
DDATA0
-RESET
C39
47pF
PST[3:0]
DDATA[3:0]
NOTE: 4.7K pull up resistors are used on signals -BKPT, DSCLK, DSI, DSO, -RESET & -TA.
I/O or Pad Voltage.
Core Voltage.
+1.8V
+3.3V
JP10
JP11
JUMPERS 10 & 11 SHOULD BE INSTALLED ACROSS PINS 1&2 DURING ASSEMBLY.
TP1
TRANSFER ACK
TP2
OUTPUT ENABLE
-OE
TP3
READ/WRITE
R/W
TP4
CHIP SELECT 0
-CS0
TP5
CPU CLOCK INPUT
CRIN
TP6
CPU CLOCK OUTPUT
BCLK
TP7
GROUND
TP8
GROUND
NOTE: Test Points TP1 to TP6 are onl populated on prototype boards. Please use the label shown for each test point on the silkscreen of the PCB.
NOTE: Place TP7 & TP8 at the corners of the PCB to allow easy connection of 'scope probe ground leads. These test points are fitted for both prototype and production boards.
Freescale Semiconductor - Microcontroller Division (MCD)
Title
MCF5249 Evaluation Board - Part number RE10811B
Size A
Document Number
Debug (BDM) Port & Test Points
Rev 1.1
Date: Thursday, March 07, 2002
Sheet 4 of 10

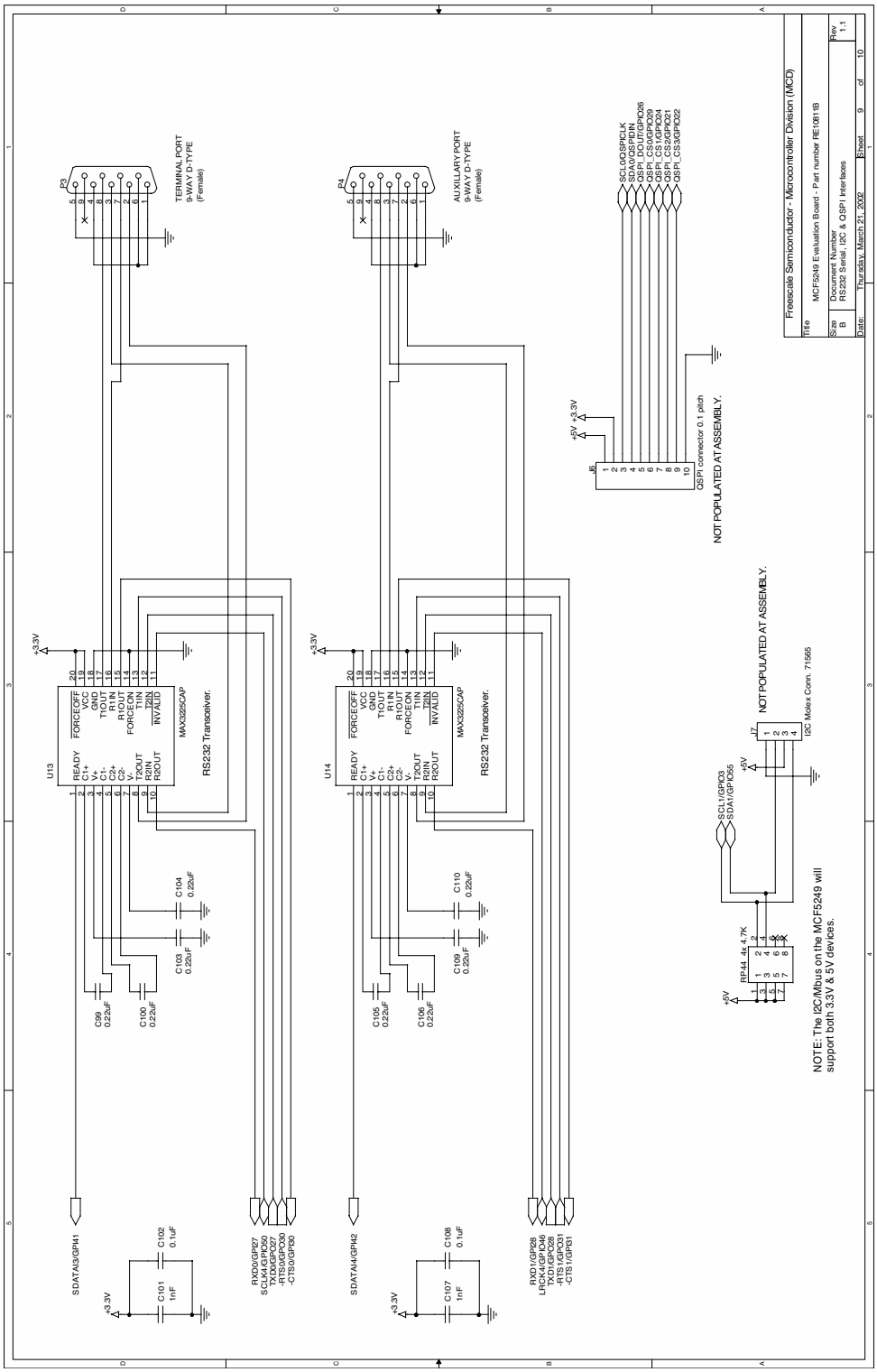


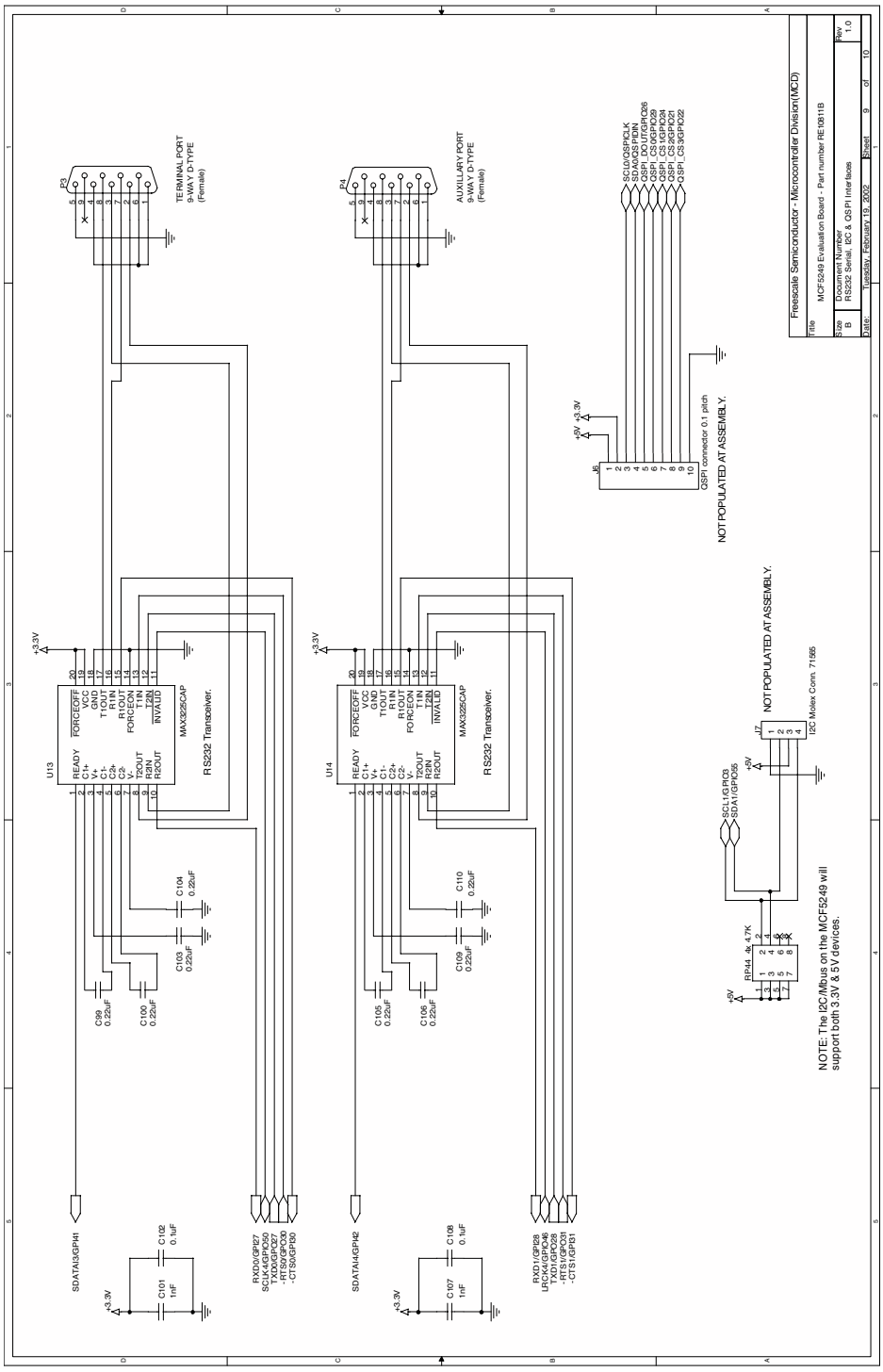

Freescale Semiconductor - Microcontroller Division (MCD)
Title
MCF5249 Evaluation Board - Part number RE1061R
Size
C
Document Number
MCF5249 Expansion Connectors
Rev
1.1
Date
Thursday, March 31, 2005
Sheet
6
of
10
NOTE: Please place D3 through D21 diodes together in a line.
Analog Ground
AMP 177983-5 120way SMT Receptacle
AMP 177983-5 120way SMT Receptacle


Freescale Semiconductor - Microcontroller Division (MCD)
Title MCF5249 Evaluation Board - Part number RE1281B
Size B
Document Number IDE Interface
Rev 1.1
Date: Thursday, March 21, 2002
Sheet 10 of 10
EF/GPIO19
GPIO5
DEIORDY/GPIO16
DEIOR/GPIO13
DEIOW/GPIO14
RP41 4x 4.7K
+3.3V
R102 270
D9
RED IDE LED
J8
HDR/40 X 2 Shrouded
IDE_RESET
IDE_D7
IDE_D6
IDE_D5
IDE_D4
IDE_D3
IDE_D2
IDE_D1
IDE_D0
IDE_IOW
IDE_IOR
IDE_IOCHRDY
IDE_IRQ
IDE_A1
IDE_A0
IDE_CS0
IDE_ACTIVITY
IDE_D8
IDE_D9
IDE_D10
IDE_D11
IDE_D12
IDE_D13
IDE_D14
IDE_D15
IDE_A2
IDE_CS1
RP42 4x 10K
U116 Uni-directional
U117 Bi-directional
MC74LCX16245DT
C111 1nF
C112 0.1uF
C113 1nF
C114 0.1uF
A[24:1]
D[31:16]
R/W
BUFENB2/GPIO17


Freescale Semiconductor - Microcontroller Division (MCD)
MCF5249 Evaluation Board - Part number RE1081B
Document Number
Flash & SDRAM memory
Thursday, March 07, 2002
SDRAM memory
K4S641633D-G (4Mx16) 52PBGA (4x13)
Flash memory
AM29LV160DB90EC (1M X16bit) 48pin TSOP
DEFAULT SETTING - JUMPER 12 SHOULD BE INSTALLED ACROSS PINS 1&2
16MBit Flash Boot
R/W
-RESET
-CS0
-OE


Freescale Semiconductor - Microcontroller Division (MCD)
MCF5249 Evaluation Board - Part number RE9811B
Document Number
Power Supply, Reset & Clock Oscillator
Rev 1.1
Thursday, March 21, 2002
JP8 SHOULD BE INSTALLED DURING ASSEMBLY
JP9 SHOULD BE INSTALLED DURING ASSEMBLY
All resistors are 0805 body size.
All caps less than 1uF are 0805 body size
1.8V Regulator
3.3V Regulator
5.0V Regulator
NOTE: Diodes prevent excessive difference between 3.3V & 1.8V rails, at power up
NOTE: Schottky Diode prevents excessive difference between 3.3V & 1.8V rails, at power down
NOTE: the positive terminal of each power connector is shown on the silkscreen of the PCB
DC voltage input range +7 to +14V
Power Jack Connector - 2.1mm diameter
Switchcraft RAPC712
2-way Bare Wire Power Connector
Augat 2SV-02
PC power
+1.8V GREEN POWER LED
+3.3V GREEN POWER LED
+5V GREEN POWER LED
RED LED
HARD_RESET
RESET
SDRAM_CS2
Debounced IRQ7 Signal
Abort/IRQ7
5A Fast blow
11.2896 MHz
MBRS340T3
MRA4003T3
MAX6895LSUT-T
TLC7733ID
K511R22COD
K511R26COD


Freescale Semiconductor - Microcontroller Division (MCD)
Title MCF5249 Evaluation Board - Part number RE1801B
Size B Document Number RS232 Serial, I2C & QSPI Interfaces Rev 1.1
Date: Thursday, March 21, 2002 Sheet 9 of 10
TERMINAL PORT 9-WAY D-TYPE (Female)
AUXILLARY PORT 9-WAY D-TYPE (Female)
MAX3225CAP RS232 Transceiver.
SCL0/QSPICLK
SDA0/QSPIDIN
QSPI_DOUT/GPIO26
QSPI_DIN/GPIO27
QSPI_CS1/GPIO24
QSPI_CS2/GPIO21
QSPI_CS3/GPIO22
QSPI connector 0.1 pitch
NOT POPULATED AT ASSEMBLY.
I2C Molex Conn. 71565
SCL1/GPIO3
SDA1/GPIO55
RP44 4x 4.7K
NOTE: The I2C/Mbus on the MCF5249 will support both 3.3V & 5V devices.


Freescale Semiconductor - Microcontroller Division(MCD)
Title
MCF5249 Evaluation Board - Part number RE10811B
Size
B
Document Number
RS232 Serial, I2C & QSPI Interfaces
Rev
1.0
Date:
Tuesday, February 19, 2002
Sheet
9
of
10
P3
TERMINAL PORT
9-WAY D-TYPE
(Female)
P4
AUXILLARY PORT
9-WAY D-TYPE
(Female)
U13
U14
MAX3225CAP
RS232 Transceiver.
READY
FORCEOFF
VCC
GND
V+
C1+
C1-
C2+
C2-
V-
T2OUT
R2IN
R2OUT
T1IN
T2IN
INVALID
R1OUT
FORCEON
R1IN
T1OUT
C98 0.22uF
C100 0.22uF
C103 0.22uF
C104 0.22uF
C105 0.22uF
C106 0.22uF
C109 0.22uF
C110 0.22uF
C101 1nF
C102 0.1uF
C107 1nF
C108 0.1uF
+3.3V
+5V
SDATA13/GPI41
SDATA14/GPI42
RXD0/GPI27
SCL0/4/GPI25
TXD0/GPI27
-RTS0/GPO30
-CTS0/GPI30
RXD1/GPI28
TXD1/GPI26
-RTS1/GPO31
-CTS1/GPI31
SCL0/QSPICLK
SDA0/QSPIDIN
QSPI_DOUT/GPIO26
QSPI_CS1/GPIO29
QSPI_CS2/GPIO30
QSPI_CS3/GPIO31
QSPI_CS0/GPIO22
J5
QSPI connector 0.1 pitch
NOT POPULATED AT ASSEMBLY.
SCL1/GPIO3
SDA1/GPIO55
J7
I2C Molex Conn. 71565
NOT POPULATED AT ASSEMBLY.
RP44 4k 4.7K
NOTE: The I2C/Mbus on the MCF5249 will
support both 3.3V & 5V devices.

Appendix D
Evaluation Board BOM

Table D-1. M5249C3 Bill of Materials

Item	Qty	Reference	Part	Function
1	36	C3,C5,C7,C9,C15,C16,C17,C18,C27,C28,C29,C30,C42,C44, C45,C46,C52,C53,C69,C70, C71,C72,C81,C82,C88,C91,C94,C98,C102,C108,C112,C114, C124,C125,C126, C127	0.1uF 25V	SMT Decoupling Capacitors
2	5	C4,C10,C86,C121,C122	10uF 16V Electrolytic (Al)	SMT Capacitors
3	2	C6,C8	1uF 50V Electrolytic (Al)	SMT Capacitors
4	5	C11,C12,C87,C90,C97	330uF 10V Tant. AVX TPSE337K10CLR	SMT Decoupling Capacitors
5	10	C13,C14,C99,C100,C103,C104,C105,C106,C109,C110,	0.22uF 16 or 25V	SMT Capacitors
6	28	C19,C20,C21,C22,C31,C32,C33,C34,C40,C43,C47,C48,C49, C50,C51,C73,C74,C75,C76,C83,C84,C85,C95,C101,C107,C111,C113,C115	1nF 50V COG	SMT Decoupling Capacitors
7	17	C23,C24,C25,C26,C35,C36,C37,C38,C58,C59,C60,C61,C64, C65,C66,C67,C130	470pF 50V COG	SMT Decoupling Capacitors
8	1	C39	47pF 50V COG	SMT Capacitors
9	7	C41,C54,C55,C56,C57,C62,C63	10nF 25V X7R	SMT Capacitors
10	2	C92,C93	22pF 50V COG	SMT Capacitors
11	1	C96	Rubycon 35V 1000uF	Thru' hole Capacitor
12	4	C116,C118,C119,C120	4.7uF 16V X7R	SMT Capacitors
13	2	C117,C123	2.2nF 50V X7R	SMT Capacitors
14	5	D1,D2,D6,D7,D8	MBRS340T3	SMT Schottky Power Diodes
15	2	D3,D4	MRA4003T3	SMT Power Diodes
16	3	D5,D9,D10	Liteon LTL-94PURK-TA	Red SMT LEDs
17	3	D11,D12,D22	Liteon LTL-94PGK-TA	Green SMT LEDs
18	9	D13,D14,D15,D16,D17,D18,D19, D20,D21	Liteon LTL-94PURK-TA	Red SMT LEDs

Table D-1. M5249C3 Bill of Materials (continued)

Item	Qty	Reference	Part	Function
19	1	F1	Multicomp MCHTE-15M	Fuse
20	1	JP1	Harwin M22-2010305	3-way jumper - de-emphasis select, low = no, high = yes
21	1	JP2	Harwin M22-2010305	3-way jumper - high pass correction, low = no, high = yes
22	1	JP3	Harwin M22-2010305	3-way jumper - clock sample freq., low = 256fs, high = 384fs
23	1	JP4	Harwin M22-2010305	3-way jumper - audio bit stream, low = 16-bit LSB justified, high = I^2S 16/18/20-bit
24	1	JP5	Harwin M22-2010205	2-way jumper - I^2C SDA0 pull-up
25	1	JP6	Harwin M22-2010205	2-way jumper - I^2C SCL0 pull-up
26	1	JP7	Harwin M22-2010305	3-way jumper - BDM/JTAG select
27	3	JP8, JP9, JP15	Harwin M22-2010205	2-way jumpers - 1.8V uP power, 3.3V uP power & -CS1 for expansion connector not E'net
28	3	JP10,JP11,JP12	Harwin M22-2010305	3-way jumpers - I/O voltage select, Core voltage select & dBUG or user code select for Flash
29	2	JP13,JP14	Harwin M22-2010205	2-way jumpers - audio output load impedance select
30	1	J1	Schurter 4832.2320	3.5mm Stereo jack socket (SW)
31	1	J2	2x13 0.1 pitch connector	Shrouded BDM header
32	1	J3	Amphenol RHJS-5381	RJ45 connector+integrated LEDS
33	2	J4,J5	AMP 177983-5	120 way SMT receptacle - expansion connectors
34	1	J6	1x10 0.1 pitch connector	QSPI connector (not fitted)
35	1	J7	Molex 71565	I^2C connector (not fitted)
36	1	J8	2x20 0.1 pitch connector	IDE connector
37	2	J9,J10	Marushin Electric/Schurter MR551L	RCA Phono Audio jack socket
38	1	J11	PC disk drive power conn.	Alternate EVB power connector
39	1	J12	1x8 0.1 pitch SIL conn.	iSP GAL programming connector
40	2	L1,L2	Siemens B82111-B-C24	DC-DC Power supply inductors
41	1	L3	Murata BLM31AJ601SN1L	Ferrite bead inductor
42	1	P1	Switchcraft RAPC712	2.1mm barrel power connector
43	1	P2	Augat 25V-02	2-way bare wire power connector

Table D-1. M5249C3 Bill of Materials (continued)

Item	Qty	Reference	Part	Function
44	2	P3,P4	McMurdo SDEX9SNT	RS232 ports - 9 way thru' hole
45	13	RP1, RP2, RP3, RP4, RP5, RP6, RP7, RP8, RP9, RP41, RP43, RP44, RP45	Philips ARC241-4K7	4x 4.7K ohm SMT resistor packs
46	1	RP10	Philips ARC241-75R	4x 75 ohm SMT resistor pack
47	4	RP11,RP12,RP13,RP42	Philips ARC241-10K	4x 10K ohm SMT resistor packs
48	27	RP14,RP15,RP16,RP17,RP18, RP19,RP20,RP21,RP22,RP23, RP24,RP25,RP26,RP27,RP28, RP29,RP30,RP31,RP32,RP33, RP34,RP35,RP36,RP37,RP38, RP39, RP40	Philips ARC241-47R	4x 47 ohm SMT resistor packs
49	2	R1,R2		SMT 10 ohm 0805 resistors
50	4	R85,R88,R96,R121		SMT 1K ohm 0805 resistors
51	2	R86,R87		SMT 49.9 ohm 1% 0805 resistors
52	4	R89,R90,R91,R92		SMT 24.9 ohm 1% 0805 resistors
53	1	R93		SMT 11K ohm 1% 0805 resistor
54	1	R94		SMT 10K ohm 0805 resistor
55	4	R95,R103,R122,R123		SMT 4.7K ohm 0805 resistors
56	4	R97,R102,R104,R105		SMT 270 ohm 0805 resistors
57	1	R98		SMT 120 ohm 0805 resistor
58	1	R99		SMT 3K ohm 0805 resistor
59	1	R100		SMT 56 ohm 0805 resistor
60	1	R101		SMT 1M ohm 0805 resistor
61	1	R106		SMT 560 ohm 0805 resistor
62	11	R107,R108,R109,R110,R111, R112,R113,R114,R115,R116, R117		SMT 470 ohm 0805 resistors
63	1	R118		SMT 22 ohm 0805 resistor
64	2	R119,R120		SMT 18 ohm 0805 resistors
65	1	S1	C&K KS11R23CQD	Red reset switch
66	1	S2	C&K KS11R22CQD	Black abort/Int. switch
67	8	TP1,TP2,TP3,TP4,TP5,TP6,TP7,TP8	Keystone Electrical Components Cat. No.5015	SMT Test points - only TP7 & TP8 (GND) fitted for production
68	1	T1	Halo TG110-S050N5	Ethernet isolation transformer
69	1	U1	AKM AK4360VF	DAC and stereo amplifier
70	1	U2	Freescale XCF5249VF140	ColdFire V2 CPU - 140MHz

Table D-1. M5249C3 Bill of Materials (continued)

Item	Qty	Reference	Part	Function
71	1	U3	Pletronics	25MHz oscillator for LAN91C111
72	1	U4	SMSC LAN91C111	Ethernet controller 10/100BaseT
73	1	U5	Lattice ispGAL22LV1-5LJ	PAL22V10 3.3V in-system programmable (isp)
74	1	U6	AMD Am29LV160DB90EC	1Mx16 48pin TSSOP Flash EEPROM memory
75	1	U7	Samsung K4S641633D-G	Synchronous DRAM 4Mx16 52PBGA (4x13)
76	1	U8	Linear Technology LT1086CM	DC to DC regulator (+5V to +1.8V)
77	1	U9	Maxim MAX6355LSUT-T	Reset controller and 3 rail voltage sensor (+5V, +3.3V & +1.8V)
78	1	U10	National Semiconductor NC7SZU04	CMOS unbuffered inverter driving oscillator circuit
79	1	U11	National Semiconductor LM2596S-3.3	DC to DC regulator (+7V/+14V to +3.3V)
80	1	U12	National Semiconductor LM2596S-5	DC to DC regulator (+7v/+14V to +5V)
81	2	U13, U14	Maxim MAX3225CAP	+3.3V RS232 transceivers
82	2	U16, U17	On Semiconductor MC74LCX16245DT	16bit wide bus transceivers
83	1	U18	TI TLC7733ID	Abort switch debounce circuit
84	1	U19	AKM AK5353VT	Stereo audio ADC via I^2S I/F
85	1	X1	HC49US case	11.2896MHz quartz crystal