MCF5206eLITE
Evaluation Board
User’s Manual

Rev. 2
DISCLAIMER

The information in this manual has been carefully examined and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, Motorola reserves the right to make changes to any product(s) herein to improve reliability, function, or design. The M5206eLITE board is not intended for use in life and/or property critical applications. Here, such applications are defined to be any situation in which any failure, malfunction, or unintended operation of the board could, directly, or indirectly, threaten life, result in personal injury, or cause damage to property. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Motorola Inc. will not assume responsibility for any damages incurred or generated by this product. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights, if any, or the rights of others.

WARNING

THIS BOARD GENERATES, USES, AND CAN RADIATE RADIO FREQUENCY ENERGY AND, IF NOT INSTALLED PROPERLY, MAY CAUSE INTERFERENCE TO RADIO COMMUNICATIONS. AS TEMPORARILY PERMITTED BY REGULATION, IT HAS NOT BEEN TESTED FOR COMPLIANCE WITH THE LIMITS FOR CLASS A COMPUTING DEVICES PURSUANT TO SUBPART J OF PART 15 OF 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.

Motorola is a registered trademark of Motorola Inc.
IBM PC and IBM AT are registered trademarks of IBM Corp.
All other trademark names mentioned in this manual are the registered trademark of their respective owners.
TABLE OF CONTENTS

CHAPTER 1  1-1
1.1 INTRODUCTION .............................................................................................................. 7
1.2 GENERAL HARDWARE DESCRIPTION ............................................................................ 7
1.3 SYSTEM MEMORY .......................................................................................................... 8
1.4 SERIAL COMMUNICATION CHANNELS ......................................................................... 8
1.5 PARALLEL I/O PORTS .................................................................................................. 8
1.6 PROGRAMMABLE TIMER/COUNTER .......................................................................... 8
1.7 SYSTEM CONFIGURATION .......................................................................................... 8
1.8 INSTALLATION AND SETUP ....................................................................................... 9
1.9 SYSTEM POWER-UP AND INITIAL OPERATION ....................................................... 12
1.10 M5206eLITE JUMPER SETUP .................................................................................... 12
   1.10.1. Jumper JP2 - This jumper selects between -CS0 to Flash or off-board connector J1 .......... 12
   1.10.2. Jumper JP1 – DRAM SIMM voltage selection – 5V or 3.3V .............................................. 12
   1.10.3. Jumper JP3 - This jumper selects between BDM & JTAG operation of the MCF5206e ...... 13
   1.10.4. Jumper JP4 - This jumper selects between 3.3V or 5V supply to the BDM cable ............... 13
   1.10.5. Jumper JP5 - This jumper allows current measurements of the MCF5206e to be taken .... 13
1.11 USING THE BDM .................................................................................................... 13

CHAPTER 2  2-1
2.1 WHAT IS dBUG? .......................................................................................................... 14
2.2 OPERATIONAL PROCEDURE ....................................................................................... 15
2.2.1. System Power-up .................................................................................................... 15
2.2.2. System Initialization ............................................................................................. 16
2.2.2.1. Hard RESET Button .......................................................................................... 16
2.2.2.2. Software Reset Command ............................................................................... 16
2.2.2.3. USER Program ................................................................................................. 16
2.2.3. System Operation .................................................................................................. 17
2.3 TERMINAL CONTROL CHARACTERS ..................................................................... 17
2.4 dBUG COMMAND SET ............................................................................................. 18
2.4.1. AS - Assembly AS ............................................................................................... 18
2.4.2. BC - Compare Blocks of Memory BC ................................................................. 20
2.4.3. BF - Block of Memory Fill BF .............................................................................. 20
2.4.4. BM - Block Move BM ......................................................................................... 21
2.4.5. BR - Breakpoint BR ............................................................................................ 22
2.4.6. BS - Block Search BS .......................................................................................... 23
2.4.7. DATA - Data Conversion DATA ......................................................................... 24
2.4.8. DI - Disassemble DI ............................................................................................ 24
2.4.9. DL - Download Serial DL .................................................................................... 25
2.4.10. Go - Execute GO ............................................................................................... 25
2.4.11. GT - Execute Until a Temporary Breakpoint GT ................................................ 26
2.4.12. HELP - Help HE ............................................................................................... 26
2.4.13. IRD - Internal Registers Display IRD ................................................................. 26
2.4.14. IRM - Internal Registers MODIFY IRM ............................................................. 27
2.4.15. MD - Memory Display MD ................................................................................ 27
2.4.16. MM - Memory Modify MM ................................................................................ 28
2.4.17. RD - Register Display RD .................................................................................. 28
2.4.18. RM - Register Modify RM .................................................................................. 29
2.4.19. SET - Reset the board and dBUG RESET ......................................................... 29
2.4.20. SHOW - Show Configuration SHOW .................................................................. 30
2.4.21. SHOW - Show Configuration SHOW .................................................................. 30
CHAPTER 3 3-1

3.1 THE PROCESSOR AND SUPPORT LOGIC .......................................................................................... 37
  3.1.1. The Processor ......................................................................................................................... 37
  3.1.2. The Reset Logic ...................................................................................................................... 37
  3.1.3. The -HIZ Signal ...................................................................................................................... 37
  3.1.4. The Clock Circuitry .................................................................................................................. 37
  3.1.5. Watchdog Timer (BUS MONITOR) ......................................................................................... 38
  3.1.6. Interrupt Sources ................................................................................................................... 38
  3.1.7. Internal SRAM ......................................................................................................................... 38
  3.1.8. The MCF5206e Registers and Memory Map ......................................................................... 38
  3.1.9. Reset Vector Mapping ............................................................................................................ 39
  3.1.10 -TA Generation ...................................................................................................................... 39
  3.1.11 Wait State Generator ............................................................................................................. 40

3.2 THE ASYNCHRONOUS DRAM SIMM ............................................................................................ 40

3.3 FLASH ROM .................................................................................................................................. 40
  3.3.1. JP2 Jumper and User’s Program ............................................................................................... 40

3.4 THE SERIAL COMMUNICATION CHANNELS ................................................................................ 40
  3.4.1. The MCF5206e UARTs ........................................................................................................... 40
  3.4.2. Motorola Bus (M-Bus) Module ............................................................................................... 41

3.5 THE PARALLEL I/O PORT ............................................................................................................. 41

3.6 THE CONNECTORS AND THE EXPANSION BUS ......................................................................... 41
  3.6.1. The Terminal Connector J9 ...................................................................................................... 41
  3.6.2. The Auxiliary Serial Communication Connector P2 ............................................................... 41
  3.6.3. The M-Bus/PC Connector J5 .................................................................................................... 42
  3.6.4. Processor Expansion Bus J1 & J2 ............................................................................................ 42
  3.6.5. The Debug Connector J3 ......................................................................................................... 45
  3.6.6. The 5V Tolerant GPIO Connector J10 .................................................................................... 46
  3.6.7 The GPIO Open Collector Driver Connector J11 ..................................................................... 46
  3.6.8. Asynchronous DRAM SIMM Connections CN1 ..................................................................... 47

APPENDIX A PALLV16V8 CODE – PALASM4 ....................................................................................... 49

APPENDIX B SCHEMATICSS ............................................................................................................... 51
TABLES

TABLE 1 – JP2, CS0 SELECT ............................................................................................................................... 12
TABLE 2 – JP1, DRAM SIMM VOLTAGE SELECTION .......................................................................................... 12
TABLE 3 – JP3, BDM OR JTAG OPERATION ...................................................................................................... 13
TABLE 4 – JP4, 3.3V OR 5V SUPPLY FOR THE BDM I/F .................................................................................. 13
TABLE 5 – JP5, CURRENT MEASUREMENT OF THE MCF5206e ........................................................................ 13
TABLE 6 - dBUG COMMANDS ............................................................................................................................... 18
TABLE 7 - THE M5206eLITE MEMORY MAP .................................................................................................. 39
TABLE 8 - THE J9 (TERMINAL) CONNECTOR PIN ASSIGNMENT ................................................................. 41
TABLE 9 - THE J4 CONNECTOR PIN ASSIGNMENT ............................................................................................ 42
TABLE 10 - THE J5 CONNECTOR PIN ASSIGNMENT .......................................................................................... 42
TABLE 11 - THE J1 CONNECTOR PIN ASSIGNMENT .......................................................................................... 43
TABLE 12 - THE J2 CONNECTOR PIN ASSIGNMENT .......................................................................................... 44
TABLE 13 - THE J3 CONNECTOR PIN ASSIGNMENT .......................................................................................... 45
TABLE 14 - THE J10 CONNECTOR PIN ASSIGNMENT ....................................................................................... 46
TABLE 15 - THE J11 CONNECTOR PIN ASSIGNMENT ....................................................................................... 46
TABLE 16 - THE CN1 CONNECTOR PIN ASSIGNMENT ....................................................................................... 47

FIGURES

FIGURE 1 BLOCK DIAGRAM OF THE BOARD .................................................................................................. 7
FIGURE 2 PIN ASSIGNMENT FOR J4 (TERMINAL) CONNECTOR ....................................................................... 10
FIGURE 3 SYSTEM CONFIGURATION ........................................................................................................... 11
FIGURE 4 JUMPER AND CONNECTOR PLACEMENT ......................................................................................... 11
FIGURE 5 FLOW DIAGRAM OF dBUG OPERATIONAL MODE ............................................................................ 15
INTRODUCTION TO THE M5206eLITE BOARD

1.1 INTRODUCTION

The M5206eLITE is a versatile single board computer based on MCF5206e ColdFire® Processor. It may be used as a powerful microprocessor based controller in a variety of applications. With the addition of a terminal, it serves as a complete microcomputer system for development/evaluation, training and educational use. To have a fully functional system, all that is required is an RS-232 terminal (or a PC with terminal emulation software) and a regulated 5V power supply.

It is possible to expand the memory and I/O capabilities of this board by connecting additional hardware via the Microprocessor Expansion Bus connectors, although it may be necessary to add bus buffers to accommodate additional bus loading.

Furthermore, provisions have been made on the printed circuit board (PCB) to permit configuration of the board in a way which best suits an application. Options available are: up to 32M bytes of ADRAM (FPM or EDO – not fitted), 1M byte (256Kx32) FSRAM, Timers, general purpose I/O, an MBus(I2C) slave device (real-time clock) and 1M byte (512Kx16) of Flash EEPROM. All of the processor’s signals are also available via connectors J1 and J2 for expansion purposes.

1.2 GENERAL HARDWARE DESCRIPTION

The M5206eLITE board provides FSRAM, Flash ROM, RS232 and all the built-in I/O functions of the MCF5206e for learning and evaluating the attributes of the MCF5206e. The MCF5206e is a member of the ColdFire® family of processors. It is a 32-bit processor with up to 32 bits of addressing and 32 lines of data. The processor has eight 32-bit data registers, eight 32-bit address registers, a 32-bit program counter and a 16-bit status register.

The MCF5206e has a System Integration Module referred to as SIM. The module incorporates many of the functions needed for system design. These include programmable chip-select logic, system protection logic, general purpose I/O and interrupt controller logic. The chip-select logic can select up to eight memory banks or peripherals in addition to two banks of ADRAMs. The chip-select logic also allows a programmable number of wait-states to allow the use of slower access memory (refer to MCF5206e User’s Manual by Motorola for detailed information about configuration of the SIM - system integration module.) The M5206eLITE board only uses three of the available chip selects to access the Flash EEPROM, FSRAM and the extra GPIO. The on-chip ADRAM controller can be used to control one ADRAM SIMM module of up to 32M bytes of ADRAM (the ADRAM SIMM is not fitted to the board, but can easily be added by the user), both -RAS lines and all four –CAS lines are used. All other functions of the SIM are available to the user.

A hardware watchdog timer (Bus Monitor) circuit is included in the SIM that monitors the bus activities. If a bus cycle is not terminated within a programmable time, the watchdog timer will assert an internal transfer error signal to terminate the bus cycle. A block diagram of the board is shown in Figure 1.

![Figure 1 Block Diagram of the board](image-url)
1.3 SYSTEM MEMORY

There is one on board Flash EEPROM (U4), which is configured in hardware to be 16-bits wide. The M5206eLITE comes with the AM29LV800BB Flash EEPROM programmed with debugger/monitor firmware (dBUG). The AM29LV800BB Flash EEPROM is 8Mbits giving a total of 1M byte (512Kx16) of Flash memory. This version of dBUG monitor only supports AM29LV800BB Flash EEPROM.

There is one 72-pin SIMM socket for ADRAM (ADRAM SIMM not fitted).

The MCF5206e processor has 8K bytes organized as 2048x32 bits of internal SRAM.

The internal cache of the MCF5206e is a non-blocking 4k-Byte Direct-Mapped Instruction Cache. The ROM Monitor currently does not utilize the cache, but programs downloaded with the ROM Monitor can use the cache.

1.4 SERIAL COMMUNICATION CHANNELS

The MCF5206e has 2 UART’s with independent baud rate generators. The signals of channel one are passed through external Driver/Receivers to make the channel compatible with RS-232. UART1 is used by the debugger for the user to access with a terminal. The UART1 channel is the “TERMINAL” channel used by the debugger for communication with external terminal/PC. The “TERMINAL” baud rate is set at 19200. The signals of channel two are passed directly to connector J4, where they can be used for 5V serial communications.

The MCF5206e also incorporates the M-Bus module, which is compatible with the MBus(FC) standard. Connected to the MBus(FC) bus as a slave device, is a Dallas Semiconductor real time clock and NVRAM (64 bytes) device, U17 – DS1307Z. At manufacture this device is programmed with the time and date which is battery backed up (BT1). This can be read back or written, using the M5206eLITE MBus(FC) sample code provided on the ColdFire web site – http://www.motorola.com/ColdFire/.

1.5 PARALLEL I/O PORTS

The MCF5206e processor offers one 8-bit general-purpose parallel I/O port. Each pin can be individually programmed as input or output. The parallel port bits PP(3:0) are multiplexed with PST(3:0) and PP(7:4) are multiplexed with DDATA(3:0). The Pin Assignment Register (PAR) controls both nibbles of the parallel port. After reset, all pins are configured as PST and DDATA pins to allow real time trace and debug. Apart from the on-chip GPIO there are two memory mapped bus transceivers (U14 & U15) controlled via chip select 3. Transceiver U14 is configured for output only and drives a 7-segment LED display, which provides status information about the board. One line of U14 (A7) also drives the direction control signal of transceiver U15. This allows the developer to use U15 for input or output. Input is limited to 5V signals via connector J10 on the board. Output however can be either 3.3V at connector J10, or a much higher voltage supplied by external hardware on pin 9 of J11. Connector J11 is driven by an open collector device, U16 – ULN2803A, allowing high drive I/O.

1.6 PROGRAMMABLE TIMER/COUNTER

The MCF5206e has two built in general-purpose 16-bit timer/counters. These timers are available to the user. The signals for the timers are available on the J1 and J2 connectors. The signals for timer/counter 0 are multiplexed with the DMA request signals DREQ0 (TIN1) and DREQ1 (TOUT1). The functionality of these pins is programmed via the PAR register (Pin Assignment Register) during initialization of the processor. However, timer 0 is available to the user as an internal counter/timer, e.g. for an RTOS system clock.

1.7 SYSTEM CONFIGURATION

The M5206eLITE board requires the following items for minimum system configuration (Figure 3):

1. The M5206eLITE board (provided).
2. Regulated power supply, 5V with minimum of 1.5 Amps.
3. RS-232C compatible terminal or a PC with terminal emulation software.
4. Serial communication cable (not provided).
5. Macraigor BDM cable (part of the Mentor toolkit provided with the board).

Refer to next sections for the initial setup.
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 the board is set up for the first time, ensure that all jumpers are in the default locations. The standard configuration does not require any modifications. After the board is setup in its standard configuration, you may use the BDM cable by following the instructions provided in the following sections.

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:

- M5206eLITE Single Board Computer
- M5206eLITE User’s Manual, this documentation
- One Macraigor BDM cable (part of the Mentor toolkit)
- MCF5206e User’s Manual
- Motorola Literature CDROM
- Getting Started Documentation

From participating Third Party Tool Developers - faxback trial forms & trial software on CD
- Mentor Graphics - Microtec Product Division - BDM cable, Trial Compiler & “XRAY4” Debugger
- Software Development Systems - Trial “SingleStep” Debugging Software
- Noral Micrologics - Faxback trial form for a BDM based debugging cable & a mouse mat
- Diab Data - Trial compiler tool chain
- Embedded Support Tools (EST) - Faxback trial form for debugging hardware & “VisionClick” software
- WindRiver Systems - trial compiler toolchain & RTOS
- Green Hills Software - Trial compiler toolchain & debugging suite
- Accelerated Technology - Trial “Nucleus+” RTOS

WARNING

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

Once you verified that all the items are present, remove the board from its protective jacket. 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 Williams Electronic Design immediately (irbwilliams@compuserve.com). Each board carries a 12 months return to manufacturer warranty – whether the board is repaired or replaced is at the manufacturers discretion.

1.8.2. Preparing the Board for Use

The board as shipped is ready to be connected to a terminal and a power supply without any need for modification. However, follow the steps below to ensure proper operation from the first time you apply the power. Figure 4 shows the placement of the jumpers and the connectors, which you need to refer to in the following sections. The steps to be taken are:

a. Connecting the power supply.
b. Connecting the terminal.

1.8.3. Providing Power to the Board

Connector J8 is a screw terminal connector. The board accepts 5V DC (regulated) at 1.5 Amps via this connector. Power supplied to the processor passes through jumper JP5. This does not include resistors used to pull-up signals attached to the processor. Jumper JP5 can be removed and with the use of a current meter, can be used to perform power analysis of the MCF5206e.

<table>
<thead>
<tr>
<th>Contact No.</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
</tr>
</tbody>
</table>
1.8.4. Selecting Terminal Baud Rate

The serial channel of MCF5206e that is used for serial communication has a built-in timer used by the ROM MONITOR to generate the baud rate used to communicate with a terminal. It can be programmed to a number of baud rates. After the power-up or a manual RESET, the ROM Monitor firmware configures the channel for 19200 baud. After the ROM Monitor is running, the SET command may be issued to choose any baud rate supported by the ROM Monitor. Refer to Chapter 2 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 of the board. The character format is 8 bits per character, no parity, and one stop bit. You need to ensure that your terminal or PC is set to this format.

1.8.6. Connecting the Terminal

The board is now ready to be connected to a terminal. Use an RS-232 serial cable to connect the PC to the M5206eLITE. The cable should have a 9-pin female D-sub connector at one end and a 9-pin male D-sub connector at the other end. Connect the 9-pin male connector to the J9 connector on the M5206eLITE. Connect the 9-pin female connector to one of the available serial communication channels normally referred to as COM1 (COM2, etc.) on the IBM PC or compatible. Depending on the kind of serial connector on the back of your PC, the connector on your PC may be a male 25-pin or 9-pin. You may need to obtain a 9-pin-to-25-pin adapter to make the connection. If you need to build an adapter, refer to Figure 2 that shows the pin assignment for the 9-pin connector on the board.

1.8.7. Using a Personal Computer as a Terminal

You may use your personal computer as a terminal provided you also have a terminal emulation software such as PROCOMM, KERMIT, QMODEM, Windows 95 Hyper Terminal or similar packages. Then connect as described in 1.8.6 Connecting the Terminal.

Once the connection to the PC is made, you are ready to power-up the PC and run the terminal emulation software. When you are in the terminal mode, you need to select the baud rate and the character format for the channel. Most terminal emulation software packages provide a command known as “Alt-p” (press the p key while pressing the Alt key) to choose the baud rate and character format. Make sure you select 8 bits, no parity, one stop bit, see section 1.8.5 The Terminal Character Format. Then, select the baud rate as 19200. Now you are ready to apply power to the board.

![Figure 2 Pin assignment for J4 (Terminal) connector.](image)

1. Data Carrier Detect, Output (shorted to pin 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 (not connected – could be shorted to pins 1 & 6).
5. Signal Ground.
6. Data Set Ready, Output (shorted to pin 1).
7. Request to Send, input.
8. Clear to send, output.
Figure 3 System Configuration

Figure 4 Jumper and connector placement
1.9 SYSTEM POWER-UP AND INITIAL OPERATION

When the cables are connected, power may be applied to the board. After power is applied the dBUG monitor program initializes the board, then displays the power-up message on the terminal. The amount of memory present will also be displayed.

**Hard Reset**

*FSRAM Size: 1M*

Copyright 1997-1999 Motorola, Inc. All Rights Reserved.

ColdFire® MCF5206e EVS Debugger Vx.x.x (xxx 199x xx:xx:xx)

Enter ‘help’ for help.

*dBUG>*

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

1. Make sure that the power supply is set to the correct voltage and current levels and is properly connected to the board.
2. Check that the terminal and board are set for the same character format and baud.
3. Press the black RESET (S1) button to ensure that the board has been initialized properly.

If you are not receiving the proper response, your board may have been damaged. Contact Williams Electronic Design (irbwilliams@compuserve.com) for further help.

1.10 M5206eLITE Jumper Setup

The jumpers on the board are discussed in Chapter 3. However, a brief discussion of the jumper settings follows:

1.10.1. **Jumper JP2 - This jumper selects between -CS0 to Flash or off-board connector J1**

<table>
<thead>
<tr>
<th>JP2 Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2 Flash ROM U4 (default)</td>
</tr>
<tr>
<td>2 and 3 Header (J1)</td>
</tr>
</tbody>
</table>

1.10.2. **Jumper JP1 – ADRAM SIMM voltage selection – 5V or 3.3V**

This jumper allows the selection of either 5V or 3.3V supply to the ADRAM SIMM. The default is 5V.

<table>
<thead>
<tr>
<th>JP1 Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2 5V supply (default)</td>
</tr>
<tr>
<td>2 and 3 3.3V supply</td>
</tr>
</tbody>
</table>
1.10.3. Jumper JP3 - This jumper selects between BDM & JTAG operation of the MCF5206e

<table>
<thead>
<tr>
<th>JP3</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>BDM operation (default)</td>
</tr>
<tr>
<td>2 and 3</td>
<td>JTAG operation</td>
</tr>
</tbody>
</table>

1.10.4. Jumper JP4 - This jumper selects between 3.3V or 5V supply to the BDM cable

As the Macraigor BDM cable, supplied as part of this evaluation kit, is a 3.3V cable, 3.3V is the default setting.

<table>
<thead>
<tr>
<th>JP4</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>5V supply</td>
</tr>
<tr>
<td>2 and 3</td>
<td>3.3V supply (default)</td>
</tr>
</tbody>
</table>

1.10.5. Jumper JP5 - This jumper allows MCF5206e current consumption to be measured

With the jumper fitted, current is supplied to the MCF5206e processor as normal. With the jumper removed, a current meter can be connected to pins 1 & 2 of the jumper to measure the current required by the MCF5206e.

<table>
<thead>
<tr>
<th>JP3</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 and 2</td>
<td>Jumper fitted (default)</td>
</tr>
</tbody>
</table>

1.11 USING THE BDM

The MCF5206e has a built in debug mechanism referred to as BDM (Background Debug Mode). The M5206eLITE board has the necessary connector, J3, to facilitate this connection.

In order to use BDM, simply connect the 26-pin IDC header to the J3 connector. No special setting is needed.

Refer to the BDM section of the MCF5206e User’s Manual for additional instructions.

IMPORTANT: There is no key to protect the BDM cable from being rotated and plugged in incorrectly. To prevent hooking up the BDM cable incorrectly, the position of pin 1 is clearly marked on the board and is denoted by a red strip on the cable.
CHAPTER 2

USING THE MONITOR/DEBUG FIRMWARE

The M5206eLITE Computer Board has a resident firmware package that provides a self-contained programming and operating environment. The firmware, named dBUG, provides the user with monitor/debug, disassembly, program download, and I/O control functions. This Chapter is a description of the dBUG package, including the user interface and command structure.

2.1 WHAT IS dBUG?

dBUG is a resident firmware package for the ColdFire® family Computer Boards. The firmware (stored in one 512Kx16 Flash ROM device) provides a self-contained programming and operating environment. The user interacts with dBUG through pre-defined commands that are entered via an RS232 terminal.

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

dBUG assumes that an 80x24 character dumb-terminal is utilized to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit, 8N1. The baud rate is 19200 baud which can be changed after 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. Therefore dBUG 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 5. After the system initialization, the board waits for a command-line input from the user terminal. When a proper command is entered, the operation continues in one of the two basic modes. If the command causes execution of the user program, the dBUG firmware may or may not be re-entered, depending on the operation of the user’s code. In 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 readable only via 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 useless.

A reference to “SP” actually refers to “A7”.

2.2 OPERATIONAL PROCEDURE

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

2.2.1. System Power-up

a. Be sure the power supply is connected properly prior to power-up.
b. Make sure the terminal is connected to TERMINAL (J9) connector.
c. Turn power on to the board.

Figure 5 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 $30000000 in FSRAM. 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 0x30000000, and then points the
VBR to 0x30000000.

The Software Watchdog Timer is disabled, Bus Monitor enabled, and internal timers are placed in a stop condition.
Interrupt controller registers initialized with unique interrupt level/priority pairs.

After initialization, the terminal will display:

Hard Reset
FSRAM Size: 1M

Copyright 1997-1999 Motorola, Inc. All Rights Reserved.
ColdFire® MCF5206e EVS Debugger Vx.x.x (xxx 199x xx:xx:xx)
Enter ‘help’ for help.
dBUG>

If you did not get this response check the setup. Refer to Section 1.9 SYSTEM POWER-UP AND INITIAL
OPERATION. Note, the date ‘xxx 199x xx:xx:xx’ may vary in different revisions.

Other means can be used to re-initialize the M5206eLITE Computer Board firmware. These means are discussed
in the following paragraphs.

2.2.2.1. Hard RESET Button.

Hard RESET is the black push button switch (S1) located in the corner of the board. Depressing this switch causes
all processes to terminate, resets the MCF5206e processor and board logic and restarts the dBUG firmware.
Pressing the RESET button would be the appropriate action if all else fails.

2.2.2.2. 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”.

2.2.2.3. USER Program.

The user can return control of the system to the firmware by recalling dBUG via his/her program. Instructions can
be inserted into the user program to call dBUG via the TRAP 15 handler.
2.2.3. System Operation

After system initialization, the terminal will display:

**Hard Reset**
**FSRAM Size: 1M**

*Copyright 1997-1999 Motorola, Inc. All Rights Reserved.*

*ColdFire® MCF5206e EVS Debugger Vx.x.x (xxx 199x xx:xx:xx:)*

Enter ‘help’ for help.

dBUG>

and waits for a command.

The user can call any of the commands supported by the firmware. A standard input routine controls the system while the user types a line of input. Command processing begins only after the line has been entered and followed by a carriage-return.

### NOTES

1. The user memory is located at addresses $30020000-$300FFFFF, $300FFFFF is the maximum FSRAM address of the memory installed on the board. When first learning the system, the user should limit his/her activities to this area of the memory map. Address range $30000000-$3001FFFF is used by dBUG.

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

#### 2.3 TERMINAL CONTROL CHARACTERS

The command line editor remembers the last five commands, in a history buffer, which were issued. They can be recalled and then executed using control keys.

Several keys are used as a command line edit and control functions. It is best to be familiar with these functions before exercising the system. These functions include:

a. **RETURN** (carriage-return) - will enter the command line and causes processing to begin.

b. **Delete** (Backspace) key or **CTRL-H** - will delete the last character entered on the terminal.

c. **CTRL-D** - Go down in the command history buffer, you may modify then press enter key.

d. **CTRL-U** - Go up in the command history buffer, you may modify then press enter key.

e. **CTRL-R** - Recall and execute the last command entered, does not need the enter key to be pressed.

For characters requiring the control key (CTRL), the CTRL should be pushed and held down and then the other key (H,D,U or R) should be pressed.
2.4 dBUG COMMAND SET

Table 6 lists the dBUG commands. Each of the individual commands is described in the following pages.

Table 6 - dBUG Commands

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
<th>SYNTAX</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS</td>
<td>ASSEMBLE</td>
<td>AS &lt;addr&gt; &lt;instruction&gt;</td>
<td>18</td>
</tr>
<tr>
<td>BC</td>
<td>BLOCK COMPARE</td>
<td>BC&lt;FIRST&gt;&lt;SECOND&gt;&lt;LENGTH&gt;</td>
<td>20</td>
</tr>
<tr>
<td>BF</td>
<td>BLOCK FILL</td>
<td>BF&lt;WIDTH&gt; BEGIN END DATA</td>
<td>20</td>
</tr>
<tr>
<td>BM</td>
<td>BLOCK MOVE</td>
<td>BM BEGIN END DEST</td>
<td>21</td>
</tr>
<tr>
<td>BS</td>
<td>BLOCK SEARCH</td>
<td>BS &lt;WIDTH&gt; BEGIN END DATA</td>
<td>22</td>
</tr>
<tr>
<td>BR</td>
<td>BREAKPOINT</td>
<td>BR ADDR &lt;-R&gt; &lt;-C COUNT&gt; &lt;-T TRIGGER&gt;</td>
<td>23</td>
</tr>
<tr>
<td>DATA</td>
<td>DATA CONVERT</td>
<td>DATA VALUE</td>
<td>24</td>
</tr>
<tr>
<td>DI</td>
<td>DISASSEMBLE</td>
<td>DI &lt;ADDR&gt;</td>
<td>24</td>
</tr>
<tr>
<td>DL</td>
<td>DOWNLOAD SERIAL</td>
<td>DL &lt;OFFSET&gt;</td>
<td>25</td>
</tr>
<tr>
<td>GO</td>
<td>EXECUTE</td>
<td>GO &lt;ADDR&gt;</td>
<td>25</td>
</tr>
<tr>
<td>GT</td>
<td>Go TILL BREAKPOINT</td>
<td>GT &lt;ADDR&gt;</td>
<td>26</td>
</tr>
<tr>
<td>HELP</td>
<td>HELP</td>
<td>HELP &lt;COMMAND&gt;</td>
<td>26</td>
</tr>
<tr>
<td>IRD</td>
<td>INTERNAL REGISTER DISPLAY</td>
<td>IRD &lt;MODULE.REGISTER&gt;</td>
<td>26</td>
</tr>
<tr>
<td>IRM</td>
<td>INTERNAL REGISTER MODIFY</td>
<td>IRM &lt;MODULE.REGISTER&gt; &lt;DATA&gt;</td>
<td>27</td>
</tr>
<tr>
<td>MD</td>
<td>MEMORY DISPLAY</td>
<td>MD &lt;WIDTH&gt; &lt;BEGIN&gt; &lt;END&gt;</td>
<td>27</td>
</tr>
<tr>
<td>MM</td>
<td>MEMORY MODIFY</td>
<td>MM &lt;WIDTH&gt; ADDR &lt;DATA&gt;</td>
<td>28</td>
</tr>
<tr>
<td>RD</td>
<td>REGISTER DISPLAY</td>
<td>RD &lt;REG&gt;</td>
<td>28</td>
</tr>
<tr>
<td>RM</td>
<td>REGISTER MODIFY</td>
<td>RM REG DATA</td>
<td>29</td>
</tr>
<tr>
<td>RESET</td>
<td>RESET</td>
<td>RESET</td>
<td>29</td>
</tr>
<tr>
<td>SET</td>
<td>SET CONFIGURATIONS</td>
<td>SET OPTION &lt;VALUE&gt;</td>
<td>30</td>
</tr>
<tr>
<td>SHOW</td>
<td>SHOW CONFIGURATIONS</td>
<td>SHOW OPTION</td>
<td>30</td>
</tr>
<tr>
<td>STEP</td>
<td>STEP (OVER)</td>
<td>STEP</td>
<td>31</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>SYMBOL MANAGEMENT</td>
<td>SYMBOL &lt;SYMB&gt; &lt;-A SYMB VALUE&gt; &lt;-R SYMB&gt; &lt;-C</td>
<td>L</td>
</tr>
<tr>
<td>TRACE</td>
<td>TRACE(INTO)</td>
<td>TRACE &lt;NUM&gt;</td>
<td>32</td>
</tr>
<tr>
<td>UPDBG</td>
<td>UPDATE DBUG</td>
<td>UPDBG</td>
<td>32</td>
</tr>
<tr>
<td>UPUSER</td>
<td>UPDATE USER FLASH</td>
<td>UPUSER</td>
<td>33</td>
</tr>
<tr>
<td>VERSION</td>
<td>SHOW VERSION</td>
<td>VERSION</td>
<td>33</td>
</tr>
</tbody>
</table>

2.4.1. AS - Assemble

Usage: AS <addr> <instruction>

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

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

If no instruction is passed to the AS command, then AS prompts with the address where opcode will be written, and continues to assemble instructions until the user terminates the AS command by inputting a period, “.”.
The inline assembler permits the use of case-sensitive symbols defined by equate statements and labels which are stored in the symbol table. The syntax for defining symbols and labels is as follows:

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

Constants and operands may be input in several different bases:

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

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

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

Examples:

To assemble one ‘move’ instructions at the next assemble address, the command is:

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

To assemble multiple lines at 0x30112000, the command is:

```plaintext
as 12000
then:
0x30112000: start: nop
0x30112002: nop
0x30112004: lsr.l #1,d0
0x30112006: cmp #4,d0
0x30112008: beq start
0x3011200a:
```
2.4.2. **BC - Compare Blocks of Memory**

Usage: BC first second length

The BC command compares two contiguous blocks of memory the first block starting at address ‘first’, the second block starting at address ‘second’, both of length ‘length’. If the blocks are not identical, then the addresses of the first mismatch are displayed. The value for addresses ‘first’ and ‘second’ 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, normally hexadecimal.

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

```
bc 20000 FFE20000 30020000
```

2.4.3. **BF - Block of Memory Fill**

Usage: BF<width> begin end data

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

The value for addresses begin and end 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.

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 0x30010000 and ending at 0x30040000 with the value 0x1234, the command is:

```
bf 30010000 30040000 1234
```

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

```
bf.b 30010000 30040000 AB
```

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

```
bf bss_start bss_end 0
```
2.4.4. BM - Block Move

**BM**

Usage: BM begin end dest

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

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

Examples:

To copy a block of memory starting at 0x30040000 and ending at 0x30080000 to the location 0x00200000, the command is:

```
bm 30040000 30080000 200000
```

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

```
bm data_start data_end 200000
```
2.4.5. \textbf{BR - Breakpoint}

Usage: \texttt{BR addr <-r> <-c count> <-t trigger>}

The BR command inserts or removes breakpoints at address \texttt{addr}. The value for \texttt{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 \texttt{-r} option to the BR command removes a breakpoint defined at address \texttt{addr}. If no address is specified in conjunction with the \texttt{-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 \texttt{-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 \texttt{-t} option allows setting the initial trigger for the breakpoint.

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

Examples:
To set a breakpoint at the C function \texttt{main()}, the command is:

\begin{verbatim}
br _main
\end{verbatim}

When the target code is executed and the processor reaches \texttt{main()}, control will be returned to dBUG.

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

\begin{verbatim}
br _bench -t 3
\end{verbatim}

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

To remove all breakpoints, the command is:

\begin{verbatim}
br -r
\end{verbatim}
2.4.6. **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.

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.

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 0x30040000 and ending at 0x30080000 the command is:

```
bs 30040000 30080000 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 0x30040002 and the next 16-bit value is read and compared.

To search for the 32-bit value 0xABCD in the memory block starting at 0x30040000 and ending at 0x30080000, the command is:

```
bs.l 30040000 30080000 ABCD
```

This reads the 32-bit word located at 0x30040000 and compares it against the 32-bit value 0x0000ABCD. If no match is found, then the address is incremented to 0x30040004 and the next 32-bit value is read and compared.

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

```
bs.b bss_start bss_end AA
```
2.4.7. DATA - Data Conversion

Usage: DATA data

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

The value for data may be a symbol name or an absolute value. If an absolute value passed into the DATA 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 equivalent of 0x1234, the command is:

```
data 0x1234
```

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

```
data 1234
```

2.4.8. DI - Disassemble

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.

Examples:
To disassemble code that starts at 0x30040000, the command is:

```
di  30040000
```

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

```
di  _main
```
2.4.9. **DL - Download Serial**

**Usage:** DL <offset>

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

If offset is provided, then the destination address of each S-record is adjusted by offset. The DL command checks the destination address for validity. If the destination is an address below the defined user space (0x00000000-0x00020000), 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 adjust the destination address by 0x40, the command is:

```
dl 0x40
```

2.4.10. **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, or other exception, which causes control to be handed back to dBUG.

**Examples:**

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

```
go
```

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

```
go _main
```

To execute code at the address 0x30040000, the command is:

```
go 30040000
```
2.4.11. **GT - Execute Till a Temporary Breakpoint**

**Usage:**

GT <addr>

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

**Example:**

To execute code at the current program counter and stop at breakpoint address 0x30010000, the command is:

```bash
GT 30010000
```

2.4.12. **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
```

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

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

```
help br
```

2.4.13. **IRD - Internal Registers Display**

**Usage:**

IRD <module.register>

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

The registers are organized according to the module to which they belong. The available modules on the MCF5206e are SIM, UART1, UART2, TIMER, M-Bus, ADRAMC, and Chip-Select. Refer to MCF5206e User’s Manual.

**Example:**

```
ird sim.sypcr ;display the SYP CR register in the SIM module.
```
2.4.14. IRM - Internal Registers MODIFY

Usage: IRM module.register data

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

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

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

2.4.15. 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 value for addresses begin and end may be an absolute address specified as a hexadecimal value, or a symbol name. Width modifies the size of the data that is displayed.

Memory display starts at the address begin. If no beginning address is provided, the MD command uses the last 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 0x30040000, the command is:

md 30040000

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 0x30040000 to 0x30050000, the command is:

md.b 30040000 30050000

To display a range of 32-bit values starting at 0x30040000 and ending at 0x30050000, the command is:

md.l 30040000 30050000

This command may be repeated by simply pressing the carriage-return (Enter) key. It will continue with the address after the last display address.
2.4.16. MM - Memory Modify

Usage: MM<width> addr <data>

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

If a value for data is provided, 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 0x30010000 to be 0xFF, the command is:

\[ \text{mm.b 30010000 FF} \]

To interactively modify memory beginning at 0x30010000, the command is:

\[ \text{mm 30010000} \]

2.4.17. 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.

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

\[ \text{rd} \]

To display only the program counter, the command is:

\[ \text{rd pc} \]
2.4.18. **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 to contain the value 0x1234, the command is:

```
rm D0 1234
```
2.4.20. SET - Set Configuration

**SET**

Usage: SET option <value>

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

The board needs a **RESET** after this command in order for the new option(s) to take effect.

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

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

Examples:

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

```
set
```

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

```
set baud 19200
```

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

2.4.21. SHOW - Show Configuration

**SHOW**

Usage: SHOW option

SHOW

The SHOW command displays the settings of the user configurable options within dBUG. Most options configurable via the **SET** command can be displayed with the **SHOW** command. If the **SHOW** command is issued without any option, it will show all options.

Examples:

To display all the current options, the command is:

```
show
```

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

```
show baud
```
2.4.22. **STEP - Step Over**

Usage: **STEP**

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

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

Examples:

To pass over a subroutine call, the command is:

```
step
```

---

2.4.23. **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.

Examples:

To define the symbol “main” to have the value 0x30040000, the command is:

```
symbol -a main 30040000
```

To remove the symbol “junk” from the table, the command is:

```
symbol -r junk
```

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

```
symbol -s
```

To display the symbol table, the command is:

```
symbol -l
```
2.4.24. **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.

Examples:

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

    tr

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

    tr 20

2.4.25. **UPDBUG - Update the dBUG Image**  

Usage: UPDBUG

The UPDBUG command is used for updating the dBUG image in Flash. When updates to the MCF5206e EVS dBUG are available, the updated image is downloaded to address 0x30020000. 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, and thus the board, useless!
2.4.26. UPUSER - Update User Code In Flash

Usage: UPUSER <number of sectors>

The UPUSER command places user code and data into space allocated for the user in Flash. There are six sectors of 128K each available as user space. To place code and data in user Flash, the image is downloaded to address 0x30020000, and the UPUSER command issued. This command programs all six sectors of user Flash space. Users access this space starting at address 0xFFE20000. To program less than six sectors, supply the number of sectors you wish to program after the UPUSER command.

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

```
upuser or upuser 6
```

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

```
upuser 1
```

2.4.27 VERSION - Display dBUG Version

Usage: VERSION

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

The version number is separated by a decimal, for example, “v1.1”. The first number indicates the version of the CPU specific code, and the second number indicates the version of the board specific code.

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

Examples:

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

```
version
```

2.5 TRAP #15 Functions

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

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

2.5.1. OUT_CHAR

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

Assembly example:

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

C example:

```c
void board_out_char (int ch)
{
    /* If your C compiler produces a LINK/UNLK pair for this routine,
    * then use the following code which takes this into account
    */

    #if l
    /* LINK a6,#0 — produced by C compiler */
    asm (“ move.l 8(a6),d1”);  /* put ‘ch’ into d1 */
    asm (“ move.l #0x0013,d0”); /* select the function */
    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.l 4(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:

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

C example:

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

2.5.3. **CHAR_PRESENT**

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

Assembly example:

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

C example:

```c
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:

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

C example:

```c
void board_exit_to DBG (void)
{
    asm (" move.l #0x0000,d0"); /* select the function */
    asm (" trap #15"); /* exit and transfer to DBG */
}
```
CHAPTER 3

HARDWARE DESCRIPTION AND RECONFIGURATION

This chapter provides a functional description of the M5206eLITE board hardware. With the description given here and the schematic diagrams provided at the end of this manual, the user can gain a good understanding of the board’s design. In this manual, an active low signal is indicated by a “-” preceding the signal name.

3.1 THE PROCESSOR AND SUPPORT LOGIC

This part of the Chapter discusses the ColdFire processor and general supporting logic on the M5206eLITE board.

3.1.1. The Processor

The microprocessor used in the M5206eLITE is the highly integrated MCF5206e, 32-bit processor. The MCF5206e uses a ColdFire® processor as the core with 4k-Byte Direct-Mapped Instruction Cache, two UART channels, two 16-bit Timers, 8K bytes of SRAM, Motorola M-Bus Module supporting the MBus(I^2C) bus, one-byte wide parallel I/O port and the supporting integrated system logic. All the registers of the core processor are 8-bit, 16-bit, or 32 bits wide. All the data and address registers are 32 bit wide. This processor communicates with external devices over a 32-bit wide data bus, D31-D0 with support for 8 and 16-bit ports. This chip can address the entire 4G Bytes of memory space using internal chip-select logic that can mask memory block sizes from 64K to 2G individually. All the processor’s signals are available through 80way Samtec connectors, J1 and J2. Refer to section 3.6 for pin assignments.

The MCF5206e has an IEEE JTAG-compatible port and BDM port. These signals are available at port J3. The processor also has the logic to generate up to eight (8) chip selects, -CS0 to -CS7, and support for ADRAM (FPM or EDO).

3.1.2. The Reset Logic

The reset logic provides system initialization under two modes. Under system power-up and when the RESET switch, S1 (black switch), is active. The power-on generates the Master RESET by asserting -RSTI and –HIZ that causes total system reset. The RESET switch also generates a Master Reset that resets the entire processor.

U8 is used to produce an active low power-on RESET signal which feeds the MCF5206e (U7) along with the push-button RESET. The MAX708TCSA (U8) device generates the system reset (-RESET).

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 $30000000 in FSRAM. 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 $30000000, and then points the VBR to $30000000.

The Software Watchdog Timer is disabled, Bus Monitor enabled, and internal timers are placed in a stop condition. Interrupt controller registers initialized with unique interrupt level/priority pairs. The parallel I/O port is configured for PST and DDATA to allow some third party BDM cables to give real-time trace and debug functionality.

3.1.3. The -HIZ Signal

The –HIZ signal is actively driven by the MAX708TCSA (U8). This signal is available for monitor on J1. The user should be very careful before driving this signal as it puts all processor signals in to a tri-state condition. This can only really be useful if the user wants to completely disable the MCF5206e and allow another device to drive the bus, which might more elegantly be achieved via the bus arbitration signals –BR, -BG and –BD, all also available on connector J1.
3.1.4. The Clock Circuitry

The M5206eLITE uses a 54MHZ-oscillator (U9) to provide the clock to CLK pin of the processor and connector J1 for any synchronous external hardware that might require the same clock as the MCF5206e.

3.1.5. Watchdog Timer (BUS MONITOR)

A bus cycle is initiated by the processor providing the necessary information for the bus cycle (e.g. address, data, control signals, etc.) and asserting the -CS or -RAS low. Then, the processor waits for an acknowledgment (-TA or -ATA signal) from the addressed device before it can complete the bus cycle. It is possible (due to incorrect programming) that the processor attempts to access part of the address space that physically does not exist. In this case, the bus cycle will go on forever, since there is no memory or I/O device to provide an acknowledgment signal, and the processor will be in an infinite wait state. The MCF5206e has the necessary logic built into the chip to watch the duration of the bus cycle. If the cycle is not terminated within the preprogrammed duration the logic will internally assert a Transfer Error signal. In response, the processor will terminate the bus cycle and an access fault exception (trap) will take place.

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

3.1.6. Interrupt Sources

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

The processor goes to a service routine via the exception table. This table is in the Flash and the VBR points to it. However, a copy of this table is made in the RAM starting at $30000000. To set an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at $30000000, and then points the VBR to $30000000.

The MCF5206e has three external interrupt request lines. You can program the external interrupt request pins to a interrupt priority-level signals (-IPL[2:0]) or predefined interrupt request pins (-IRQ7, -IRQ4, -IRQ1). The M5206eLITE configures these lines as predefined interrupt request pins. There are also eight internal interrupt requests from DMA0, DMA1, Timer1, Timer2, Software watchdog timer, UART1, UART2, and MBUS. Each interrupt source, external and internal, can be programmed for any priority level. In the case of identical priority levels, a second relative priority between 0 to 3 will be assigned.

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

The -IRQ1, -IRQ4 & -IRQ7 lines of the MCF5206e are not used on this board. However, the -IRQ1 is programmed for Level 1 with priority 1 and autovector. The user may use this line for an external interrupt request. Refer to MCF5206e User’s Manual for more information about the interrupt controller.

3.1.7. Internal SRAM

The MCF5206e has 8K bytes of internal memory. This memory is mapped to 0x20000000 and is not used by the dBUG. It is available to the user.

3.1.8. The MCF5206e Registers and Memory Map

The memory and I/O resources of the M5206eLITE are divided into two groups, MCF5206e Internal and External resources. All the I/O registers are memory mapped.
The MCF5206e has built in logic and up to eight chip-select pins (-CS0 to -CS7) which are used to enable external memory and I/O devices. In addition there are two -RAS lines for ADRAM’s. There are registers to specify the address range, type of access, and the method of -TA generation for each chip-select and -RAS pin. These registers are programmed by dBUG to map the external memory and I/O devices.

The M5206eLITE uses chip-select zero (-CS0) to enable the Flash ROM (refer to Section 3.3.). The M5206eLITE uses -RAS1, -RAS2, -CAS0, -CAS1, -CAS2, and -CAS3 to enable the ADRAM SIMM module (not populated - refer to Section 3.2), -CS2 for FSRAM, and -CS3 for GPIO space.

The chip select mechanism of the MCF5206e allows the memory mapping to be defined based on the memory space desired (User/Supervisor, Program/Data spaces).

All the MCF5206e internal registers, configuration registers, parallel I/O port registers, DUART registers and system control registers are mapped by the MBAR register at any 1M-byte boundary. It is mapped to 0x10000000 by dBUG. For a complete map of these registers refer to the MCF5206e User’s Manual.

The M5206eLITE board can have up to 32M bytes of 3.3V or 5V ADRAM installed. Refer to Section 3.2 for a discussion of RAM. The dBUG is programmed in one AM29LV800BB Flash ROM which occupies 1M byte of the address space. The ROM Monitor uses the first 128K bytes. The following thirteen 128K byte sectors are available for the user. Refer to section 3.3.

<table>
<thead>
<tr>
<th>ADDRESS RANGE</th>
<th>SIGNAL and DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00000000-$003FFFFF</td>
<td>-RAS1, -RAS2, 4M bytes of ADRAM’s</td>
</tr>
<tr>
<td>$10000000-$100003FF</td>
<td>Internal Module registers</td>
</tr>
<tr>
<td>$20000000-$20001FFF</td>
<td>Internal SRAM (8K bytes)</td>
</tr>
<tr>
<td>$30000000-$300FFFFFE*</td>
<td>-CS2, External FSRAM (1M byte – 256Kx32)</td>
</tr>
<tr>
<td>$40000000-$40000FFFFF</td>
<td>-CS3, 64K bytes of GPIO</td>
</tr>
<tr>
<td>$FFE00000-$FFEFFFFFF</td>
<td>-CS0, 1M byte of Flash EEPROM (512Kx16)</td>
</tr>
</tbody>
</table>

* Installed – the level 2 cache footprint accepts Motorola’s MCM69F737TQ device and any other FSRAM with the same electrical specifications and pinout.

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

3.1.9. Reset Vector Mapping

After reset, the processor attempts to get the initial stack pointer and initial program counter values from locations $0000000-$0000007 (the first eight bytes of memory space). This requires the board to have a nonvolatile memory device in this range with proper information. However, in some systems, it is preferred to have RAM starting at address $00000000. In the MCF5206e, the -CS0 responds to any accesses after reset until the CSMR0 is written. Since -CS0 is connected to Flash EEPROM’s, the Flash EEPROMs appear to be at address $00000000 which provides the initial stack pointer and program counter (the first 8 bytes of the Flash ROM). The initialization routine, however, then programs the chip-select logic and locates the Flash EEPROM’s to start at $FFEO0000 and the ADRAMs to start at $00000000.

3.1.10. -TA Generation

The processor starts a bus cycle by providing the necessary information (address, R/-W, etc.) and asserting the -TS. The processor then waits for an acknowledgment (-TA) by the addressed device before it can complete the bus cycle. This -TA is used not only to indicate the presence of a device, it also allows devices with different access time to communicate with the processor properly. The MCF5206e, as part of the chip-select logic, has a built in mechanism to generate the -TA for all external devices that do not have the capability to generate the -TA signal on their own. The Flash EEPROM’s and ADRAM’s can not generate the -TA. Their chip-select logic is programmed by the ROM Monitor to generate the -TA internally after a pre-programmed number of wait states.
In order to support the future expansion of the board, the -TA input of the processor is also connected to the Processor Expansion Bus connector, J1. This allows any expansion boards to assert this line to indicate their -TA to the processor. On the expansion boards, however, this signal should be generated through an open collector buffer with no pull-up resistor, a pull-up resistor is included on the M5206eLITE board. All the -TA's from any expansion boards should be connected to this line.

3.1.11. Wait State Generator
The Flash EEPROM’s and ADRAM SIMM on the board may require some adjustments to match the cycle time of the processor to make them compatible with the processor speed. To extend the CPU bus cycles for the slower devices, the chip-select logic of the MCF5206e can be programmed to generate the -TA after a given number of wait states. Refer to Sections 3.2 and 3.3 information about wait state requirements of ADRAM’s and Flash EEPROM’s respectively.

3.2 THE ADRAM SIMM
The M5206eLITE has one 72-pin SIMM socket (CN1) for ADRAM SIMM. This socket supports ADRAM SIMM’s of 256Kx32, 1Mx32, 2Mx32, 4Mx32, and 8Mx32. No special configurations are needed. The dBUG will detect the total memory installed on power-up. The SIMM access speed should be 60ns.

3.3 FLASH ROM
There is one 1Mbyte Flash EEPROM on the M5206eLITE, U4 which is 16 bits wide.
The board is shipped with one AM29LV800BB, 512Kx16-word, FLASH EEPROM for a total of 1M bytes. The first 128K and last 128K are reserved by the ROM Monitor firmware. 768Kbytes are available to the user. The chip-select signal generated by the MCF5206e (-CS0) enables this chip.
The MCF5206e chip-select logic can be programmed to generate the -TA for -CS0 signal after a certain number of wait states. The dBUG programs this parameter to three wait-states.

3.3.1. JP2 Jumper and User’s Program
This jumper allows users to test code from boot without having to overwrite the ROM Monitor.
When the jumper is set between pins 1 and 2, the behavior is normal. When the jumper is set between pins 2 and 3, the board boots from any external memory connected to connector J1. The code in the external hardware will have to “mimic” the set up of the original hardware on the M5206eLITE board, it will also have to be the correct width (data port size) and speed to run with a 54MHz MCF5206e.

3.4 THE SERIAL COMMUNICATION CHANNELS
The M5206eLITE offers a number of serial communications. They are discussed in this section.

3.4.1. The MCF5206e Two UARTs
The MCF5206e has two built in UART’s, each with its own software programmable baud rate generators, only one channel is the ROM Monitor to Terminal output and other is available to the user at 3.3V levels. The ROM Monitor, however, programs the interrupt level for UART1 to Level 3, priority 2 and autovector mode of operation. The interrupt level for UART2 to Level 3, priority 1 and autovector mode of operation. The signals of this channel are available on connector J4. The signals of UART1 are also passed through the RS-232 driver/receiver and are available on a DB-9 connector J9. Refer to the MCF5206e User’s Manual for the programming and register map of the DUART module.
3.4.2. **Motorola Bus (M-Bus) Module**

The MCF5206e has a built-in M-Bus module that allows inter-chip bus interface for a number of I/O devices. It is compatible with industry-standard MBus(FC) Bus. The M5206eLITE uses this module, but it is available to the user on connectors J1 & J2. The two M-Bus signals are SDA and SCL that are available on the J5 connector. These signals are open-collector signals. However, they have pull-up resistors on the M5206eLITE. These signals are connected to the DS1307Z real-time clock device via the MBus(FC) bus which is used during production to initialize the date & time into the RTC. However, the MBus(FC) bus is not used by the dBug ROM monitor. The interrupt control register for M-Bus is set for Level 3, priority 0 and autovector.

3.5 **THE PARALLEL I/O Port**

The MCF5206e has one 8-bit parallel port. All the pins have dual functions. They can be configured as I/O or their alternate function via the Pin Assignment register. All pins are configured as DDATA and PSTx pins by the ROM Monitor to allow some third party developer BDM cables to give real-time trace and debug information.

3.6 **THE CONNECTORS AND THE EXPANSION BUS**

There are 10 connectors on the M5206eLITE board which are used to connect the board to external I/O devices and or expansion boards. This section provides a brief discussion and the pin assignments of the connectors.

3.6.1. **The Terminal Connector J9**

The signals on UART1 that run through RS-232 driver/receivers are used to drive the Terminal. The M5206eLITE uses a 9-pin D-sub female connector J9 for connecting the board to a terminal or a PC with terminal emulation software. The available signals are a working subset of the RS-232C standard. Table 8 - The J9 (Terminal) Connector pin assignment shows the pin assignment.

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>DIRECTION</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Output</td>
<td>Data Carrier Detect (shorted to 6)</td>
</tr>
<tr>
<td>2</td>
<td>Output</td>
<td>Receive data</td>
</tr>
<tr>
<td>3</td>
<td>Input</td>
<td>Transmit data</td>
</tr>
<tr>
<td>4</td>
<td>Input</td>
<td>Not Connected (shorted to 1 &amp; 6 if nec.)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Signal Ground</td>
</tr>
<tr>
<td>6</td>
<td>Output</td>
<td>Data Set Ready (shorted to 1 &amp; 4)</td>
</tr>
<tr>
<td>7</td>
<td>Input</td>
<td>Request to Send</td>
</tr>
<tr>
<td>8</td>
<td>Output</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Not Used</td>
</tr>
</tbody>
</table>

3.6.2. **The Auxiliary Serial Communication Connector P2**

The MCF5206e has two built-in UART’s. One channel is not used by the M5206eLITE ROM Monitor and is available to the user. These signals are available on connectors J2 & J4. The available signals form a working subset of the RS-232C standard. Table 9 - The J4 Connector pin assignment shows the pin assignment for J4.
### Table 9 - The J4 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>DIRECTION</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3.3V</td>
</tr>
<tr>
<td>2</td>
<td>Output</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>3</td>
<td>Input</td>
<td>Request to Send</td>
</tr>
<tr>
<td>4</td>
<td>Output</td>
<td>Receive Data</td>
</tr>
<tr>
<td>5</td>
<td>Input</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Signal Ground</td>
</tr>
</tbody>
</table>

### 3.6.3. The Mbus/I2C Connector J5

The MCF5206e has a built-in Mbus/I2C module. These signals (SDA/SCL) are available on connector J5. The available signals form a working Mbus/I2C serial connection. Table 9 - The J4 Connector pin assignment shows the pin assignment for J5.

### Table 10 - The J5 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>DIRECTION</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>5V</td>
</tr>
<tr>
<td>2</td>
<td>Output</td>
<td>Serial Clock - SCL</td>
</tr>
<tr>
<td>3</td>
<td>Bi-Directional</td>
<td>Serial Data - SDA</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>N.C.</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>N.C.</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Signal Ground</td>
</tr>
</tbody>
</table>

### 3.6.4. Processor Expansion Bus J1 & J2

All the processor signals are available on 2 Samtec connectors J1 & J2. The user may refer to the data sheets for the major parts and the schematic at the end of this manual to obtain an accurate loading capability. Tables 11-12 show the pin assignment for J1 & J2 respectively.
Table 11 - The J1 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A0</td>
<td>80</td>
<td>BKPT</td>
</tr>
<tr>
<td>2</td>
<td>A1</td>
<td>79</td>
<td>DSO</td>
</tr>
<tr>
<td>3</td>
<td>A2</td>
<td>78</td>
<td>DSCLK</td>
</tr>
<tr>
<td>4</td>
<td>A3</td>
<td>77</td>
<td>DSI</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>76</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>A4</td>
<td>75</td>
<td>-RESET</td>
</tr>
<tr>
<td>7</td>
<td>A5</td>
<td>74</td>
<td>TCK</td>
</tr>
<tr>
<td>8</td>
<td>A6</td>
<td>73</td>
<td>SDA</td>
</tr>
<tr>
<td>9</td>
<td>A7</td>
<td>72</td>
<td>TT1</td>
</tr>
<tr>
<td>10</td>
<td>A8</td>
<td>71</td>
<td>TT0</td>
</tr>
<tr>
<td>11</td>
<td>A9</td>
<td>70</td>
<td>ATM</td>
</tr>
<tr>
<td>12</td>
<td>A10</td>
<td>69</td>
<td>-TS</td>
</tr>
<tr>
<td>13</td>
<td>A11</td>
<td>68</td>
<td>-ATA</td>
</tr>
<tr>
<td>14</td>
<td>A12</td>
<td>67</td>
<td>-TA</td>
</tr>
<tr>
<td>15</td>
<td>A13</td>
<td>66</td>
<td>SIZ0</td>
</tr>
<tr>
<td>16</td>
<td>A14</td>
<td>65</td>
<td>SIZ1</td>
</tr>
<tr>
<td>17</td>
<td>A15</td>
<td>64</td>
<td>R/-W</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>63</td>
<td>GND</td>
</tr>
<tr>
<td>19</td>
<td>A16</td>
<td>62</td>
<td>CLK</td>
</tr>
<tr>
<td>20</td>
<td>A17</td>
<td>61</td>
<td>-HIZ</td>
</tr>
<tr>
<td>21</td>
<td>A18</td>
<td>60</td>
<td>N.C.</td>
</tr>
<tr>
<td>22</td>
<td>A19</td>
<td>59</td>
<td>-CS0_OFF</td>
</tr>
<tr>
<td>23</td>
<td>A20</td>
<td>58</td>
<td>-DREQ0</td>
</tr>
<tr>
<td>24</td>
<td>A21</td>
<td>57</td>
<td>-IPL2</td>
</tr>
<tr>
<td>25</td>
<td>A22</td>
<td>56</td>
<td>-IPL1</td>
</tr>
<tr>
<td>26</td>
<td>A23</td>
<td>55</td>
<td>-IPL0</td>
</tr>
<tr>
<td>27</td>
<td>D0</td>
<td>54</td>
<td>-BR</td>
</tr>
<tr>
<td>28</td>
<td>D1</td>
<td>53</td>
<td>-BR_HW</td>
</tr>
<tr>
<td>29</td>
<td>GND</td>
<td>52</td>
<td>GND</td>
</tr>
<tr>
<td>30</td>
<td>D2</td>
<td>51</td>
<td>-BG_HW</td>
</tr>
<tr>
<td>31</td>
<td>D3</td>
<td>50</td>
<td>-CS3</td>
</tr>
<tr>
<td>32</td>
<td>D4</td>
<td>49</td>
<td>-CS2</td>
</tr>
<tr>
<td>33</td>
<td>D5</td>
<td>48</td>
<td>-CS1</td>
</tr>
<tr>
<td>34</td>
<td>D6</td>
<td>47</td>
<td>-CS0</td>
</tr>
<tr>
<td>35</td>
<td>D7</td>
<td>46</td>
<td>PST0</td>
</tr>
<tr>
<td>36</td>
<td>D8</td>
<td>45</td>
<td>D15</td>
</tr>
<tr>
<td>37</td>
<td>D9</td>
<td>44</td>
<td>D14</td>
</tr>
<tr>
<td>38</td>
<td>D10</td>
<td>43</td>
<td>D13</td>
</tr>
<tr>
<td>39</td>
<td>D11</td>
<td>42</td>
<td>D12</td>
</tr>
<tr>
<td>40</td>
<td>GND</td>
<td>41</td>
<td>GND</td>
</tr>
<tr>
<td>PIN NO.</td>
<td>SIGNAL NAME</td>
<td>PIN NO.</td>
<td>SIGNAL NAME</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>D16</td>
<td>80</td>
<td>N.C.</td>
</tr>
<tr>
<td>2</td>
<td>D17</td>
<td>79</td>
<td>N.C.</td>
</tr>
<tr>
<td>3</td>
<td>D18</td>
<td>78</td>
<td>N.C.</td>
</tr>
<tr>
<td>4</td>
<td>D19</td>
<td>77</td>
<td>N.C.</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>76</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>D20</td>
<td>75</td>
<td>N.C.</td>
</tr>
<tr>
<td>7</td>
<td>D21</td>
<td>74</td>
<td>N.C.</td>
</tr>
<tr>
<td>8</td>
<td>D22</td>
<td>73</td>
<td>N.C.</td>
</tr>
<tr>
<td>9</td>
<td>D23</td>
<td>72</td>
<td>SCL</td>
</tr>
<tr>
<td>10</td>
<td>D24</td>
<td>71</td>
<td>TOUT1</td>
</tr>
<tr>
<td>11</td>
<td>D25</td>
<td>70</td>
<td>-JTAG</td>
</tr>
<tr>
<td>12</td>
<td>D26</td>
<td>69</td>
<td>N.C.</td>
</tr>
<tr>
<td>13</td>
<td>D27</td>
<td>68</td>
<td>VCC</td>
</tr>
<tr>
<td>14</td>
<td>D28</td>
<td>67</td>
<td>VCC</td>
</tr>
<tr>
<td>15</td>
<td>D29</td>
<td>66</td>
<td>VCC</td>
</tr>
<tr>
<td>16</td>
<td>D30</td>
<td>65</td>
<td>VCC</td>
</tr>
<tr>
<td>17</td>
<td>D31</td>
<td>64</td>
<td>N.C.</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>63</td>
<td>GND</td>
</tr>
<tr>
<td>19</td>
<td>A24</td>
<td>62</td>
<td>N.C.</td>
</tr>
<tr>
<td>20</td>
<td>A25</td>
<td>61</td>
<td>N.C.</td>
</tr>
<tr>
<td>21</td>
<td>A26</td>
<td>60</td>
<td>N.C.</td>
</tr>
<tr>
<td>22</td>
<td>A27</td>
<td>59</td>
<td>N.C.</td>
</tr>
<tr>
<td>23</td>
<td>N.C.</td>
<td>58</td>
<td>TIN1</td>
</tr>
<tr>
<td>24</td>
<td>N.C.</td>
<td>57</td>
<td>-DREQ1</td>
</tr>
<tr>
<td>25</td>
<td>N.C.</td>
<td>56</td>
<td>N.C.</td>
</tr>
<tr>
<td>26</td>
<td>N.C.</td>
<td>55</td>
<td>-CTS2</td>
</tr>
<tr>
<td>27</td>
<td>PST1</td>
<td>54</td>
<td>-RTS2</td>
</tr>
<tr>
<td>28</td>
<td>-TEA</td>
<td>53</td>
<td>TXD2</td>
</tr>
<tr>
<td>29</td>
<td>GND</td>
<td>52</td>
<td>GND</td>
</tr>
<tr>
<td>30</td>
<td>PST2</td>
<td>51</td>
<td>RXD2</td>
</tr>
<tr>
<td>31</td>
<td>PST3</td>
<td>50</td>
<td>-CTS1</td>
</tr>
<tr>
<td>32</td>
<td>-BG</td>
<td>49</td>
<td>-RTS1</td>
</tr>
<tr>
<td>33</td>
<td>DDATA0</td>
<td>48</td>
<td>TXD1</td>
</tr>
<tr>
<td>34</td>
<td>DDATA1</td>
<td>47</td>
<td>RXD1</td>
</tr>
<tr>
<td>35</td>
<td>DDATA2</td>
<td>46</td>
<td>-DRAMW</td>
</tr>
<tr>
<td>36</td>
<td>DDATA3</td>
<td>45</td>
<td>-CAS3</td>
</tr>
<tr>
<td>37</td>
<td>-BD</td>
<td>44</td>
<td>-CAS2</td>
</tr>
<tr>
<td>38</td>
<td>-RAS0</td>
<td>43</td>
<td>-CAS1</td>
</tr>
<tr>
<td>39</td>
<td>-RAS1</td>
<td>42</td>
<td>-CAS0</td>
</tr>
<tr>
<td>40</td>
<td>GND</td>
<td>41</td>
<td>GND</td>
</tr>
</tbody>
</table>
The MCF5206e has a Background Debug Port, Real-Time Trace Support, and Real-Time Debug Support. The necessary signals are available at connector J3. Table 13 - The J3 Connector pin assignment table shows the pin assignment.

### Table 13 - The J3 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TCK</td>
</tr>
<tr>
<td>2</td>
<td>-BKPT</td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
</tr>
<tr>
<td>4</td>
<td>DSCLK</td>
</tr>
<tr>
<td>5</td>
<td>Ground</td>
</tr>
<tr>
<td>6</td>
<td>No Connect</td>
</tr>
<tr>
<td>7</td>
<td>-RESET</td>
</tr>
<tr>
<td>8</td>
<td>DSI</td>
</tr>
<tr>
<td>9</td>
<td>+3.3 or 5 Volts (see Jumper 4)</td>
</tr>
<tr>
<td>10</td>
<td>DSO</td>
</tr>
<tr>
<td>11</td>
<td>Ground</td>
</tr>
<tr>
<td>12</td>
<td>PST3</td>
</tr>
<tr>
<td>13</td>
<td>PST2</td>
</tr>
<tr>
<td>14</td>
<td>PST1</td>
</tr>
<tr>
<td>15</td>
<td>PST0</td>
</tr>
<tr>
<td>16</td>
<td>DDAT3</td>
</tr>
<tr>
<td>17</td>
<td>DDAT2</td>
</tr>
<tr>
<td>18</td>
<td>DDAT1</td>
</tr>
<tr>
<td>19</td>
<td>DDAT0</td>
</tr>
<tr>
<td>20</td>
<td>Ground</td>
</tr>
<tr>
<td>21</td>
<td>4.7K pull down</td>
</tr>
<tr>
<td>22</td>
<td>No Connect</td>
</tr>
<tr>
<td>23</td>
<td>Ground</td>
</tr>
<tr>
<td>24</td>
<td>54MHz CLK</td>
</tr>
<tr>
<td>25</td>
<td>+3.3 or 5 Volts (see Jumper 4)</td>
</tr>
<tr>
<td>26</td>
<td>-TA</td>
</tr>
</tbody>
</table>
3.6.6. **The 5V Tolerant GPIO Connector J10**

Eight extra GPIO lines are made available to the user via a memory mapped MC74LCX646 device (U15). This device is controlled via chip select 3, and supports inputs at 5V or 3.3V levels and outputs at 3.3V only. The necessary signals are available at connector J10. The value read or written to J10 appears on bits D16 to D23 of the data bus. The direction of the data on J10 is controlled by D31 at the time of the read/write to J10. Table 14 - The J10 Connector pin assignment shows the pin assignment.

### Table 14 - The J10 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA 0</td>
</tr>
<tr>
<td>2</td>
<td>DATA 1</td>
</tr>
<tr>
<td>3</td>
<td>DATA 2</td>
</tr>
<tr>
<td>4</td>
<td>DATA 3</td>
</tr>
<tr>
<td>5</td>
<td>DATA 4</td>
</tr>
<tr>
<td>6</td>
<td>DATA 5</td>
</tr>
<tr>
<td>7</td>
<td>DATA 6</td>
</tr>
<tr>
<td>8</td>
<td>DATA 7</td>
</tr>
</tbody>
</table>

3.6.7 **The GPIO Open Collector Driven Connector J11**

Connector J11 is driven via an open collector driver. This connector is purely an output port, which via pin 9 of connector J11, can drive high voltage/ high drive loads. The necessary signals are available at connector J11. Table 15 - The J11 Connector pin assignment shows the pin assignment.

### Table 15 - The J11 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA 0</td>
</tr>
<tr>
<td>2</td>
<td>DATA 1</td>
</tr>
<tr>
<td>3</td>
<td>DATA 2</td>
</tr>
<tr>
<td>4</td>
<td>DATA 3</td>
</tr>
<tr>
<td>5</td>
<td>DATA 4</td>
</tr>
<tr>
<td>6</td>
<td>DATA 5</td>
</tr>
<tr>
<td>7</td>
<td>DATA 6</td>
</tr>
<tr>
<td>8</td>
<td>DATA 7</td>
</tr>
<tr>
<td>9</td>
<td>Open Collector Voltage Input</td>
</tr>
<tr>
<td>10</td>
<td>R.T.C. – Square Wave Output</td>
</tr>
</tbody>
</table>
3.6.8. ADRAM SIMM Connections CN1

All the ADRAM signals are available on 1x 72 pin connector CN1. The user may refer to the data sheets for various manufacturers SIMM’s and the schematic at the end of this manual to obtain an accurate loading capability. Table 16 - The CN1 Connector pin assignment shows the pin assignment for CN1.

Table 16 - The CN1 Connector pin assignment

<table>
<thead>
<tr>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
<th>PIN NO.</th>
<th>SIGNAL NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VSS</td>
<td>72</td>
<td>VSS</td>
</tr>
<tr>
<td>2</td>
<td>DQ0</td>
<td>71</td>
<td>N.C.</td>
</tr>
<tr>
<td>3</td>
<td>DQ16</td>
<td>70</td>
<td>PD4</td>
</tr>
<tr>
<td>4</td>
<td>DQ1</td>
<td>69</td>
<td>PD3</td>
</tr>
<tr>
<td>5</td>
<td>DQ17</td>
<td>68</td>
<td>PD2</td>
</tr>
<tr>
<td>6</td>
<td>DQ2</td>
<td>67</td>
<td>PD1</td>
</tr>
<tr>
<td>7</td>
<td>DQ18</td>
<td>66</td>
<td>N.C.</td>
</tr>
<tr>
<td>8</td>
<td>DQ3</td>
<td>65</td>
<td>DQ15</td>
</tr>
<tr>
<td>9</td>
<td>DQ19</td>
<td>64</td>
<td>DQ31</td>
</tr>
<tr>
<td>10</td>
<td>VCC</td>
<td>63</td>
<td>DQ14</td>
</tr>
<tr>
<td>11</td>
<td>N.C.</td>
<td>62</td>
<td>DQ30</td>
</tr>
<tr>
<td>12</td>
<td>A0</td>
<td>61</td>
<td>DQ13</td>
</tr>
<tr>
<td>13</td>
<td>A1</td>
<td>60</td>
<td>DQ29</td>
</tr>
<tr>
<td>14</td>
<td>A2</td>
<td>59</td>
<td>VCC</td>
</tr>
<tr>
<td>15</td>
<td>A3</td>
<td>58</td>
<td>DQ28</td>
</tr>
<tr>
<td>16</td>
<td>A4</td>
<td>57</td>
<td>DQ12</td>
</tr>
<tr>
<td>17</td>
<td>A5</td>
<td>56</td>
<td>DQ27</td>
</tr>
<tr>
<td>18</td>
<td>A6</td>
<td>55</td>
<td>DQ11</td>
</tr>
<tr>
<td>19</td>
<td>A10</td>
<td>54</td>
<td>DQ26</td>
</tr>
<tr>
<td>20</td>
<td>DQ4</td>
<td>53</td>
<td>DQ10</td>
</tr>
<tr>
<td>21</td>
<td>DQ20</td>
<td>52</td>
<td>DQ25</td>
</tr>
<tr>
<td>22</td>
<td>DQ5</td>
<td>51</td>
<td>DQ9</td>
</tr>
<tr>
<td>23</td>
<td>DQ21</td>
<td>50</td>
<td>DQ24</td>
</tr>
<tr>
<td>24</td>
<td>DQ6</td>
<td>49</td>
<td>DQ8</td>
</tr>
<tr>
<td>25</td>
<td>DQ22</td>
<td>48</td>
<td>-N.C.</td>
</tr>
<tr>
<td>26</td>
<td>DQ7</td>
<td>47</td>
<td>-W</td>
</tr>
<tr>
<td>27</td>
<td>DQ23</td>
<td>46</td>
<td>N.C.</td>
</tr>
<tr>
<td>28</td>
<td>A7</td>
<td>45</td>
<td>-RAS1</td>
</tr>
<tr>
<td>29</td>
<td>N.C.</td>
<td>44</td>
<td>-RAS0</td>
</tr>
<tr>
<td>30</td>
<td>VCC</td>
<td>43</td>
<td>-CAS1</td>
</tr>
<tr>
<td>31</td>
<td>A8</td>
<td>42</td>
<td>-CAS3</td>
</tr>
<tr>
<td>32</td>
<td>A9</td>
<td>41</td>
<td>-CAS2</td>
</tr>
<tr>
<td>33</td>
<td>-RAS3</td>
<td>40</td>
<td>-CAS0</td>
</tr>
<tr>
<td>34</td>
<td>-RAS2</td>
<td>39</td>
<td>VSS</td>
</tr>
<tr>
<td>35</td>
<td>N.C.</td>
<td>38</td>
<td>N.C.</td>
</tr>
<tr>
<td>36</td>
<td>N.C.</td>
<td>37</td>
<td>N.C.</td>
</tr>
</tbody>
</table>
APPENDIX A  PALLV16V8 code – PALASM4

TITLE  U10_BUS_ARBITRATION&_GPIO
PATTERN  P00001
REVISION  1
DATE  19th January 1999
AUTHOR  Pete Highton
COMPANY  Motorola SPS (c) 1999

CHIP  U10  PALCE16V8

PIN  1  CLK  COMBINATORIAL
PIN  2  /BR  COMBINATORIAL
PIN  3  /BR_HW  COMBINATORIAL
PIN  4  /BD  COMBINATORIAL
PIN  5  /CS3  COMBINATORIAL
PIN  6  U15OP  COMBINATORIAL
PIN  7  NC  COMBINATORIAL
PIN  8  NC  COMBINATORIAL
PIN  9  /WR  COMBINATORIAL
PIN 10  GND
PIN 11  NC
PIN 12  /BG_HW  COMBINATORIAL  ; O/P
PIN 13  /BG  COMBINATORIAL  ; O/P
PIN 14  /IORD  COMBINATORIAL  ; O/P
PIN 15  /IOWR  COMBINATORIAL  ; O/P
PIN 16  /U15_OE  COMBINATORIAL  ; O/P
PIN 17  NC  COMBINATORIAL
PIN 18  NC  COMBINATORIAL
PIN 19  /RD  COMBINATORIAL  ; O/P
PIN 20  VCC

EQUATIONS

RD = /WR

IORD = CS3*/WR  ; GPIO Read enable

IOWR = CS3*WR  ; GPIO write enable

U15_OE = U15OP + IORD  ; If device is O/P, permanent OE when read

; Bus arbitration...

BG = (BD + (BR */BR_HW))  ; External hardware has priority

BG_HW = BR_HW */BG  ; Bus grant to the target H/W
NOTE: CN1 is a vertical 72-pin EDO or FPM DRAM SIMM connector.

NOTE: The DRAM SIMM VCC is switchable between 5V & 3.3V.
NOTE: JFCS is /CS for Flash EEPROM, the /CSS OFF signal is provided for any external boot hardware.
NOTE: CON4 & CON5 are vias to be grouped together for user access.