# SBC5206 USER'S MANUAL
# REVISION 1.1



Arnewsh Inc.
P.O. Box 270352
Fort Collins, CO 80527-0352
Phone: (970) 223-1616
Fax: (970) 223-9573

## COPYRIGHT



## 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, Arnewsh reserves the right to make changes to any product(s) herein to improve reliability, function, or design. The SBC5206 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, Arnewsh Inc. will not assume responsibility for any damages incurred or generated by this product. Arnewsh 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.**

## LIMITED WARRANTY

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

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

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

Arnewsh Inc.
P.O. Box 270352
Fort Collins, CO 80527-0352
Phone: (970) 223-1616
Fax : (970) 223-9573

## TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION TO THE SBC5206 BOARD

## 1.1 INTRODUCTION

The SBC5206 is a versatile single board computer based on MCF5206 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 for development/evaluation, training and educational use. The user must only connect an RS-232 compatible terminal (or a personal computer with terminal emulation software) and a power supply to have a fully functional system.

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

Furthermore, provisions have been made in the PC-board to permit configuration of the board in a way which best suits an application. Options available are: up to 32M of DRAM, Timer, I/O, ISA bus interface, and up to 1M bytes of Flash or 2M bytes of EPROM. In addition, all of the I/O functions of the MCF5206 are available for the user.

## 1.2 GENERAL HARDWARE DESCRIPTION

The SBC5206 board provides the RAM, Flash ROM, optional Ethernet interface (ISA bus), RS232, and all the built-in I/O functions of the MCF5206 for learning and evaluating the attributes of the MCF5206. The MCF5206 is a member of the ColdFire family of processors. It is a 32-bit processor with 32 bits of addressing and 32 lines of data. The processor has eight 32-bit data registers, eight 32-bit address registers, a 32-bit program counter, and a 16-bit status register.

The MCF5206 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 DRAM's. The chip-select logic also allows programmable number of wait-state to allow the use of slower memory (refer to MCF5206 User's Manual by Motorola for detail information about the SIM.) The SBC5206 only uses four of the chip selects to access the Flash ROM's, MC68HC901, ISA bus interface, and the IACK for MC68HC901. The DRAM controller is used to control one SIMM module of up to 32M bytes of DRAM, both -RAS 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 which 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 1.1.

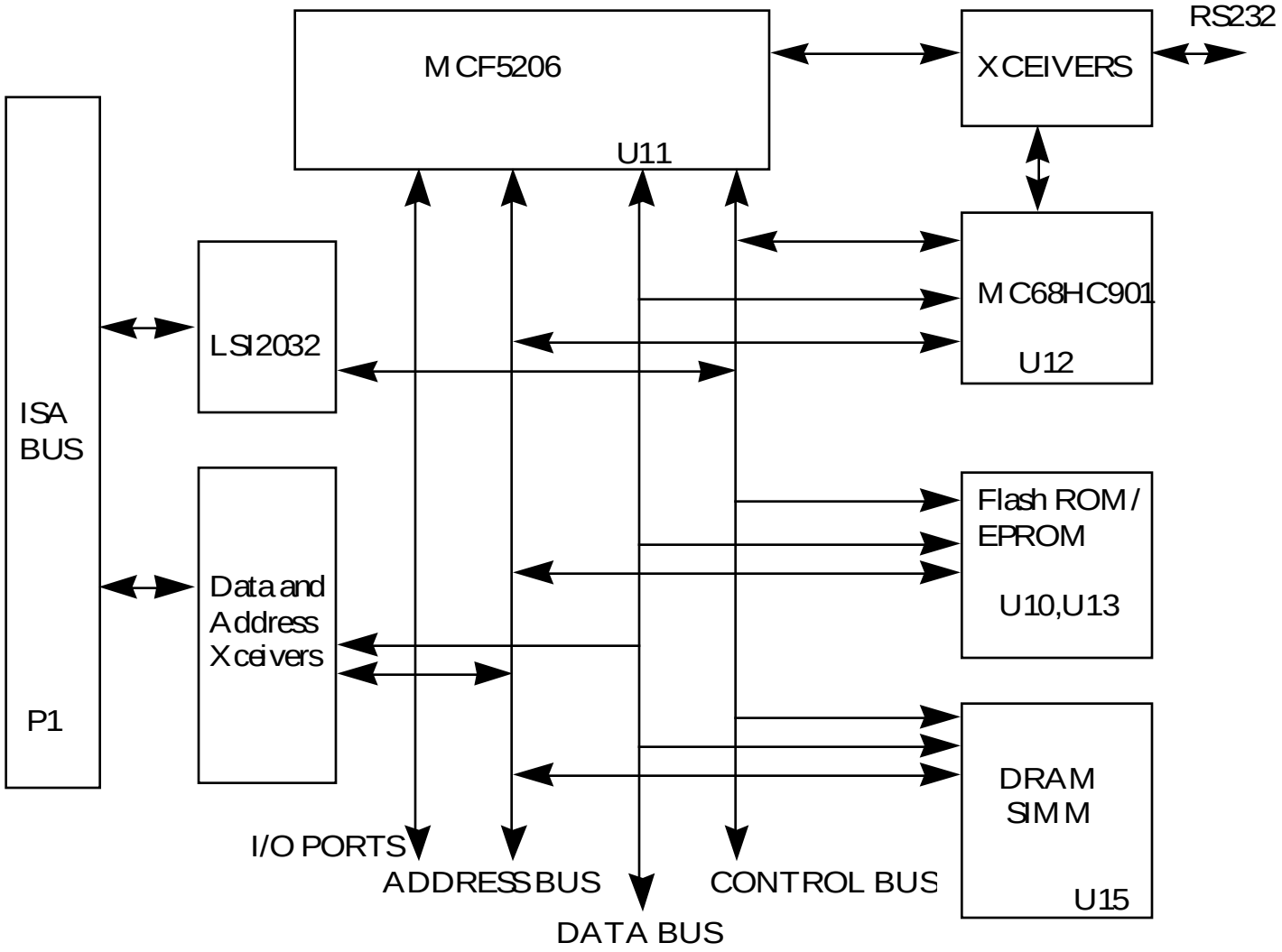

MCF5206
U11
XCEIVERS
RS232
MC68HC901
U12
LSI2032
ISA
BUS
P1
Data and
Address
Xceivers
Flash ROM /
EPROM
U10,U13
DRAM
SIMM
U15
I/O PORTS
ADDRESS BUS
CONTROL BUS
DATA BUS

Figure 1.1

## 1.3 SYSTEM MEMORY

There are two 32-pin sockets on the board for EPROM's or Flash ROM's (U10, U13), U13 is the most significant byte and the U10 is the least significant byte. The EPROM sockets can be set up via jumpers (JP2, and JP3) to accept 27C512, 27C010, 27C020, 27C040, and 27C080 EPROM's. or 29F010, and 29F040. The SBC5206 comes with two 29F010 Flash ROM's which are programmed with a debugger/monitor firmware. The dBUG only supports 29F010 flash ROM.

There is one 72-pin SIMM socket for DRAM which can accept 256Kx32 to 8Mx32 SIMM modules. The debugger detects the DRAM size installed.

## 1.4 SERIAL COMMUNICATION CHANNELS

The MCF5206 has two built-in UART's with baud rate generator. The signals of channel one are passed through external Driver/Receivers to make the channel compatible with RS-232. These channels are not used by the debugger and are available to user. In addition, the signals of both channels are available at connector J7. The SBC5206, however, has one MC68HC901 which has four timers and a serial communication port. One timer channel is used as baud rate generator for the serial channel. The SI and SO lines are passed through external Driver/Receiver to make this channel compatible with RS-232C level (Note: only 2 main signals are available, SI and SO signals). This channel is the "TERMINAL" channel used by the debugger for communication with external terminal/PC. The MCF5206 also incorporate the M-Bus which is compatible with $I^2C$ Bus standard.

## 1.5 PARALLEL I/O PORTS

MCF5206 offers one 8-bit general purpose parallel I/O port. Each pin can be individually programmed as input or output. The parallel port is multiplexed with PST(3:0) and DDATA(3:0) debug signals. The default is parallel I/O function after reset.

## 1.6 PROGRAMMABLE TIMER/COUNTER

The MCF5206 has two built in general purpose timer/counters. These timers are not used by the debugger and are available to the user. The signals for the timer are available on the connector J7. There are also three timers in MC68HC901 which are available to user.

## 1.7 ISA BUS CONNECTOR

The SBC5206 has one ISA bus connector to allow the use of off-the-shelf ISA I/O cards. The main reason for this connector is to install an Ethernet card to support down-load via network.

## 1.8 SYSTEM CONFIGURATION

The SBC5206 board requires only the following items for minimum system configuration (Fig. 1.2):

a. The SBC5206 board (provided).
b. Power supply ( +5 Vdc regulated or 7.5V to 12V DC), minimum of 0.4 Amp.
c. RS-232C compatible terminal or a PC with terminal emulation software.

d. Communication cable (provided).

Refer to next sections for initial setup.

## 1.9 INSTALLATION AND SETUP

The following sections describe all the steps needed to prepare the board for operation. Please read the following sections carefully before using the board. When you are preparing the board for the first time, do not use the optional features (Ethernet, ISA BUS). The minimum configuration does not require any modifications. After the board is functional in its minimal configuration, you may use other features by following the instructions provided in the following sections.

### 1.9.1 Unpacking

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

a. SBC5206 Single Board Computer.

b. SBC5206 User's Manual, this documentation.

c. One communication cable.

**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 Arnewsh Inc. immediately in order to correct the problem.

### 1.9.2 Preparing the Board for Use

The board as shipped is ready to be connected to a terminal and the power supply without any need for modification. However, follow the steps below to insure proper operation from the first time you apply the power. Figure 1.3 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.

### 1.9.3 Providing Power to the Board

The board accepts two means of power supply connections. Connector J3 is a 2.1mm power jack and J4 lever actuated connector. The board accepts either +5V regulated supply **or** +7.5V to 12V DC (regulated or unregulated), less than one Amp via either connectors. Jumper JP1 selects between +5 and +7.5-12V options. ***Make sure the jumper JP1 is in proper location for your option***. Connect power supply as marked on the board and shown below (do not turn the power supply on yet):

| Contact NO. | Voltage |
|---|---|
| 1 | +5 Vdc or +7.5-12V |
| 2 | Ground |

**Jumper JP1.**

| Jumper Pin | Function |
|---|---|
| 1 and 2 | +5V regulated |
| 2 and 3 | +7.5-12V DC regulated or unregulated (default) |

### 1.9.4 Selecting Terminal Baud Rate

The serial channel of MC68HC901 which is used for serial communication channel has a built in software programmable baud rate generator (timer). It can be programmed to a number of baud rates. After the power-up or a manual RESET, the dBUG firmware configures the channel for 19200 baud. After the dBUG is running, you may issue the SET command to choose any baud rate supported by the dBUG. Refer to Chapter 2 for the discussion of this command.

### 1.9.5 The Terminal Character Format

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

### 1.9.6 Connecting the Terminal

The board is now ready to be connected to a terminal. Use the communication cable provided to connect the terminal to the SBC5206. The cable has a 9-pin female D-sub connector at one end and a 9-pin male D-sub connector at the other end. Attach the 9-pin male connector to J5 connector on the board. Attach the 9-pin female connector to a 9-pin-to-25-pin adapter, if necessary, to make it compatible with the connector on the back of the terminal.

### 1.9.7 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, or similar packages. Use the communication cable provided to connect the PC to the SBC5206. The cable has 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 J5 connector on SBC5206. 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's 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 1.4 which shows the pin assignment for the 9-pin connector on the board.

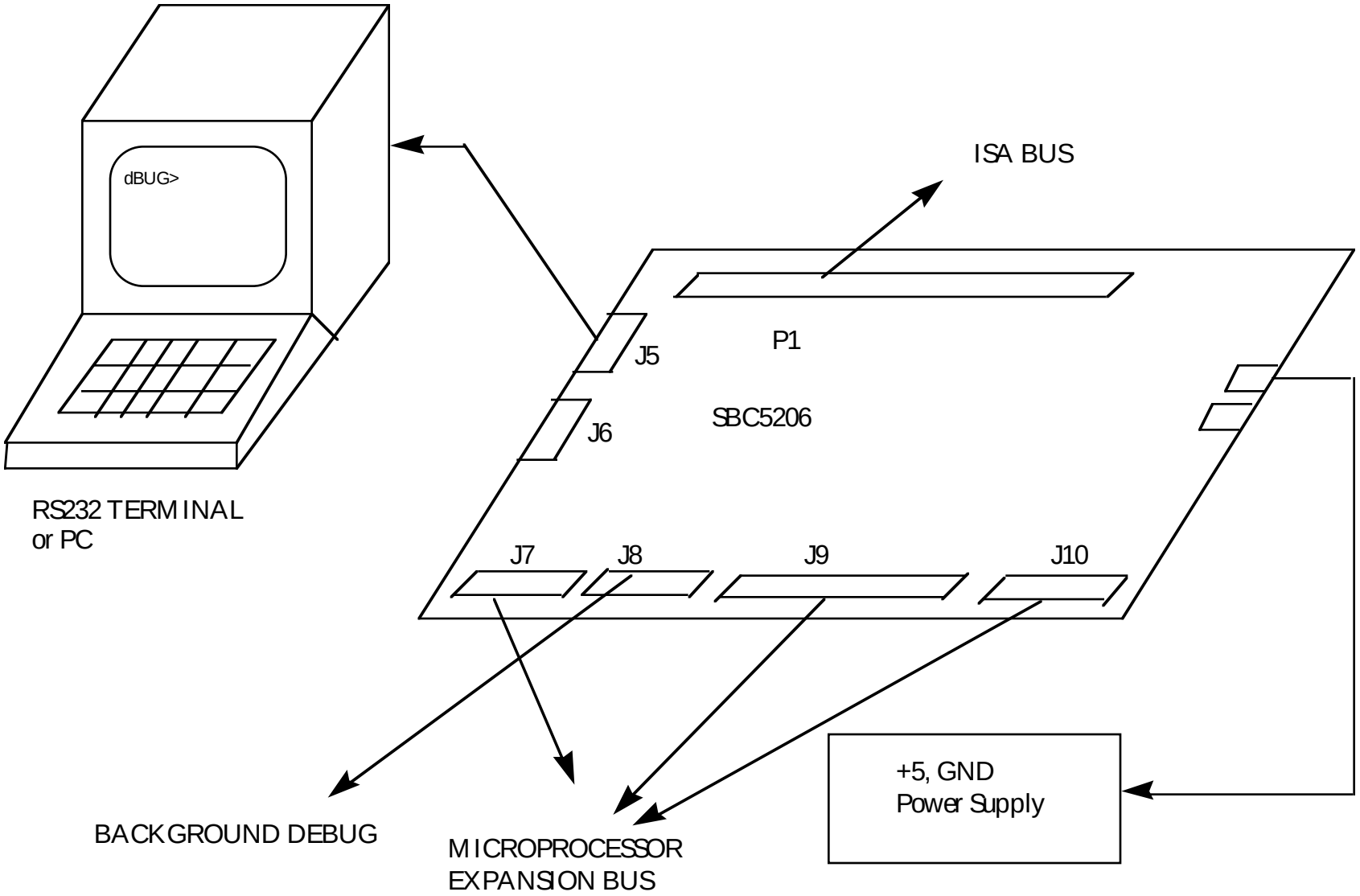
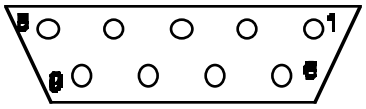



Figure 1.2. System Configuration

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 1.9.5). Then, select the baud rate as 19200. Now you are ready to apply power to the board.




1. Data Carrier Detect, Output (shorted to pins 6 and 8).
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 used).
5. Signal Ground.
6. Data Set Ready, Output (shorted to pins 1 and 8).
7. Request to Send, input.
8. Clear to send, output (shorted to pins 1 and 6).
9. Not connected.

**Figure 1.4. Pin assignment for the J5 (Terminal) connector.**

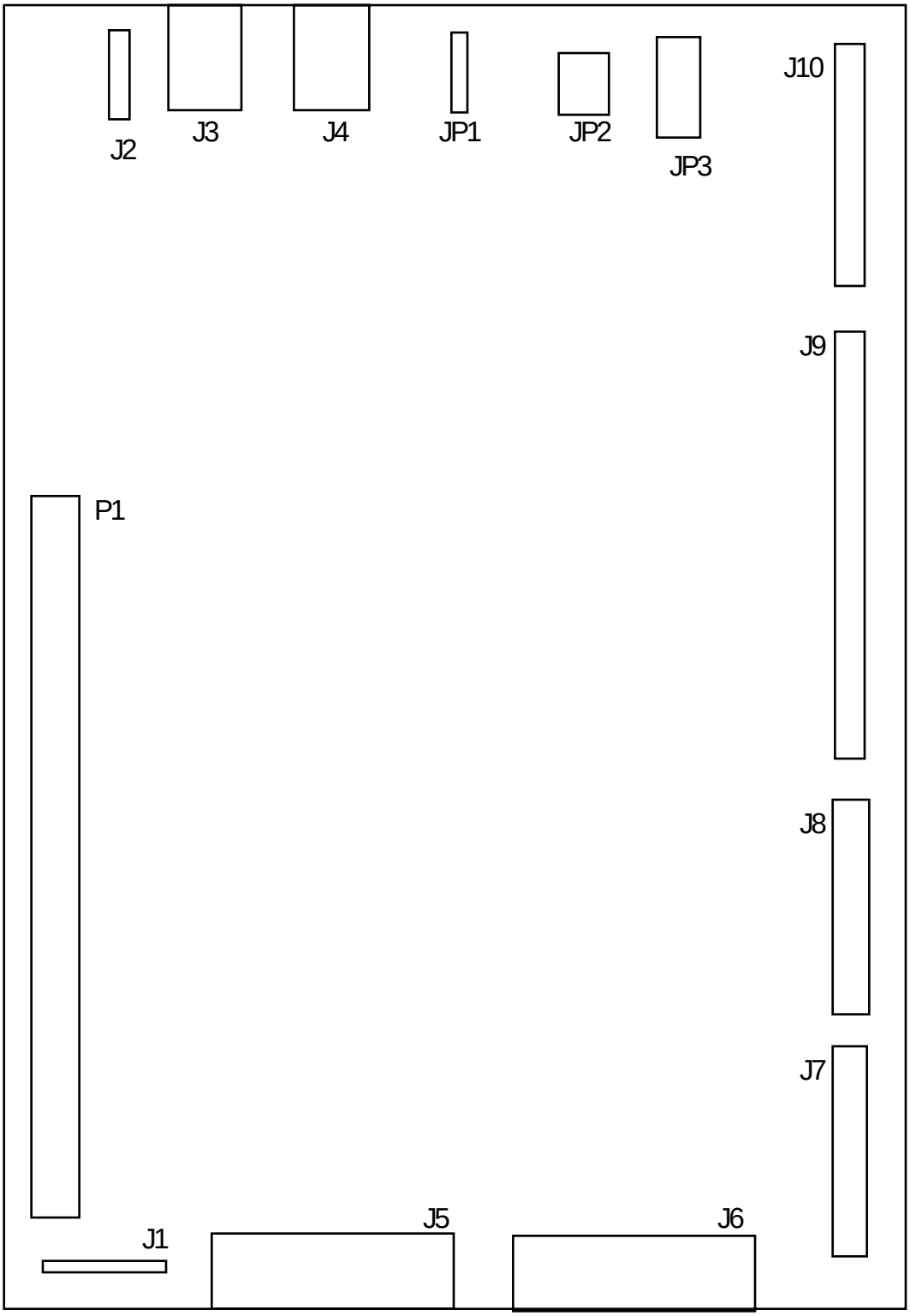



**Figure 1.3. Jumper and connector placement.**

## 1.10 SYSTEM POWER-UP AND INITIAL OPERATION

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

***Hard Reset***

***DRAM Size: 1M***

***Copyright 1995-1996 Motorola, Inc. All Rights Reserved.***
***ColdFire MCF5206 EVS Debugger V1.1 (xxx 1996 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 properly set and connected to the board.
2. Check that the terminal and board are set for the same character format and baud.
3. Press the red RESET (red switch) button to insure that the board has been initialized properly.

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

## 1.11 SBC5206 JUMPER SETUP

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

1. Jumper JP1. This jumper selects the power supply selection.

| Jumper Pin | Function |
|---|---|
| 1 and 2 | +5V regulated |
| 2 and 3 | +7.5-12V DC regulated or unregulated (default) |

2. Jumper JP2. This jumper selects between Flash and EPROM.

| Jumper Pin | Function |
|---|---|
| 3 to 5<br>and<br>4 to 6 | Select Flash (default) |

3. Jumper JP3. This jumper selects the size of EPROM or Flash.

| Jumper Pin | Function |
|---|---|
| 7 to 9<br>and<br>8 to 10 | Selects 128Kx8 EPROM/Flash |

### 1.12 USING THE BDM

The MCF5206 has a built in debug mechanism referred to as BDM. The SBC5206 has the necessary connector, J8, to facilitate this connection.

In order to use the BDM, simply connect the 26-pin IDC header at the end of the BDM cable provided by the BDM development tool (third party tool) to the J8 connector. No special setting is needed. Refer to the BDM User's Manual for additional instructions.

# CHAPTER 2

# USING THE MONITOR/DEBUG FIRMWARE

The SBC5206 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 how-to-use 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 two 128Kx8 Flash ROM devices) provides a self-contained programming and operating environment. dBUG interacts with the user through pre-defined commands that are entered via the terminal.

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

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 but can be changed after the power-up.

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

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

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

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

An additional function called the "TRAP 15 handler" allows the user program to utilize various routines within dBUG. The TRAP 15 handler is discussed at the end of this chapter.

The operational mode of dBUG is demonstrated in Figure 2-1. After the system initialization, the board waits for a command line input from the user terminal. When a proper command is entered, the operation continues in one of the two basic modes. If the command causes execution of the user program, the dBUG firmware may or may not be re-entered, depending on the discretion of the user. For the alternate case, the command will be executed under control of the dBUG firmware, and after command completion, the system returns to command entry mode.

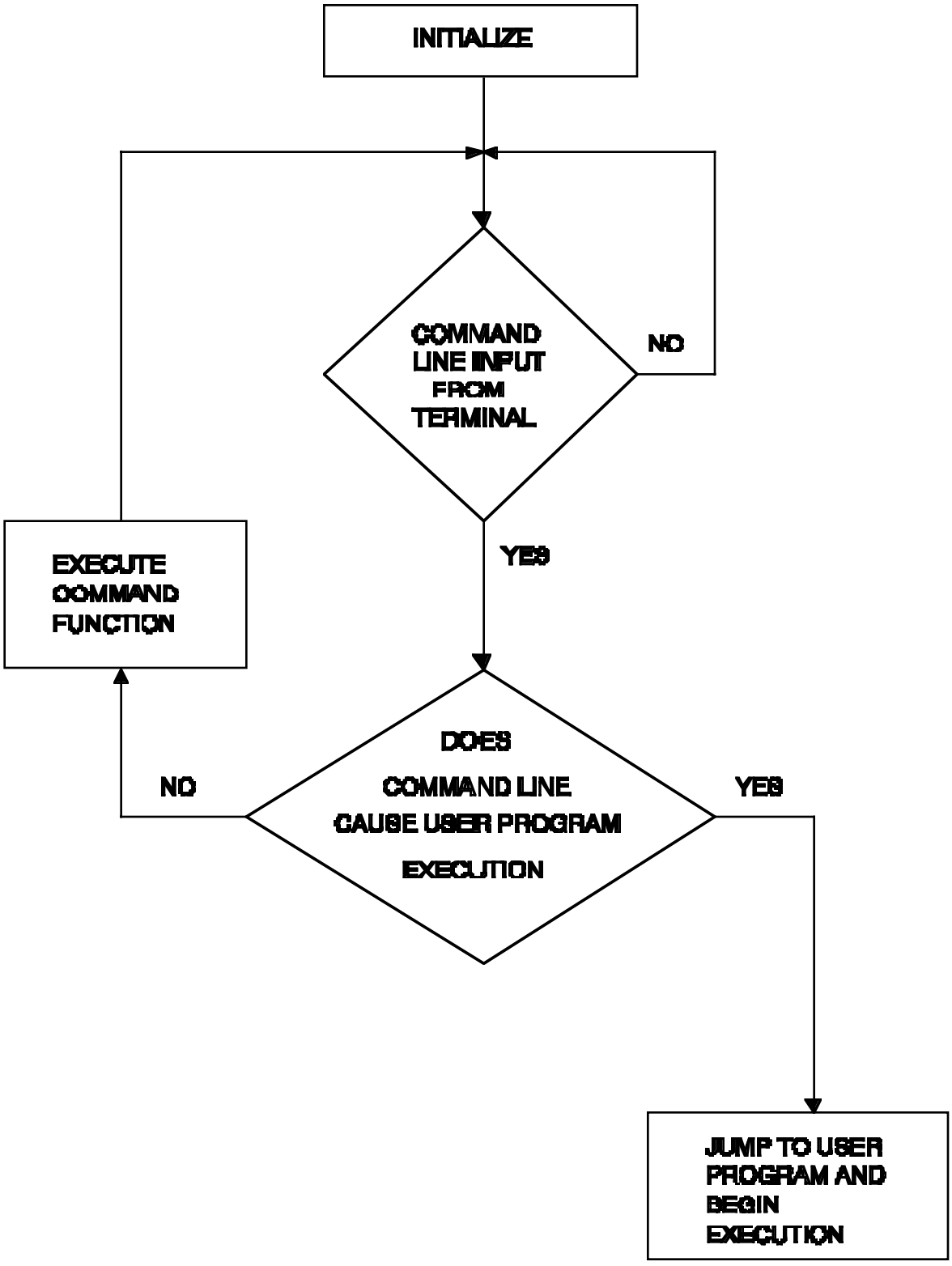



**Figure 2-1. Flow Diagram of dBUG Operational Mode.**

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

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

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

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

The core ColdFire register set is maintained by dBUG. These are listed below:

A0-A7
D0-D7
PC
SR

All control registers on ColdFire are not readable by the supervisor programming model, and thus not accessible via dBUG. User code may change these registers, but caution must be exercised as changes may render dBUG 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 (J5) connector.
c. Turn power on to the board.

### 2.2.2 System Initialization

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

dBUG performs the following configurations of internal resources during the initialization. The instruction cache is invalidated and disabled. The Vector Base Register, VBR, points to the Flash. However, a copy of the exception table is made at address $00000000 in DRAM. To take over an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at 0x00000000, and then points the VBR to 0x00000000.

The Software Watchdog Timer is disabled, Bus Monitor enabled, and internal timers are placed in a stop condition. Interrupt controller registers initialized with unique interrupt level/priority pairs. The system pins are configured for -CS4, -CS5, -WE0, -WE1, -IRQ1, -IRQ4, -IRQ7.

After initialization, the terminal will display:

***Hard Reset***

***DRAM Size: 1M***

***Copyright 1995-1996 Motorola, Inc. All Rights Reserved.***
***ColdFire MCF5206 EVS Debugger V1.1 (xxx 1996 xx:xx:xx:)***
***Enter 'help' for help.***

***dBUG>***

If you did not get this response check the setup. Refer to Section 1.10. Note, the date '***xxx 1996 xx:xx:xx***' may vary in different revisions.
Other means can be used to re-initialize the SBC5206 Computer Board firmware. These means are discussed in the following paragraphs.

**2.2.2.1 SOFT RESET Button.** SOFT RESET is the red button located in the middle side of the board. Depressing this button causes all processes to terminate, resets the MCF5206 processor and board logic's and restarts the dBUG firmware. Pressing the RESET button would be the appropriate action if all else fails. The Reset button issues a soft reset and therefore, the memory controller remains functional. If total reset is needed, the board should be powered down and up again. During the power-up, both the -RSTI and -HIZ are asserted to cause Hard Reset.

**2.2.2.2 ABORT Button.** ABORT is the black button located next to RESET button in the middle side of the board. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5206) and gives control to the dBUG firmware. This action differs from RESET in that no processor register or memory contents are changed, the processor and peripherals are not reset, and dBUG is not restarted. Also, in response to depressing the ABORT button, the contents of the MCF5206 core internal registers are displayed.

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

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

**2.2.2.4 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***

***DRAM Size: 1M***

***Copyright 1995-1996 Motorola, Inc. All Rights Reserved.***
***ColdFire MCF5206 EVS Debugger V1.1 (xxx 1996 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 $00010000-$xxxxxxxx, $xxxxxxxx is the maximum RAM address of the memory installed in the board. When first learning the system, the user should limit his/her activities to this area of the memory map. Address range $00000000-$0000FFFF 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 MCF5206 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) should be pressed.

## 2.4 dBUG COMMAND SET

Table 2-1 lists the dBUG commands. Each of the individual commands is described in the following pages.

**TABLE 2-1. dBUG Commands.**

| COMMAND MNEMONIC | DESCRIPTION | SYNTAX | PAGE |
|---|---|---|---|
| BF | BLOCK FILL | BF<WIDTH> BEGIN END DATA | 2-7 |
| BM | BLOCK MOVE | BM BEGIN END DEST | 2-8 |
| BS | BLOCK SEARCH | BS <WIDTH> BEGIN END DATA | 2-9 |
| BR | BREAKPOINT | BR ADDR <-R> <-C COUNT> <-T TRIGGER> | 2-10 |
| DATA | DATA CONVERT | DATA VALUE | 2-11 |
| DI | DISASSEMBLE | DI <ADDR> | 2-12 |
| DL | DOWNLOAD SERIAL | DL <OFFSET> | 2-13 |
| DN | DOWNLOAD NETWORK | DN <-C> <-E> <-S> <-I> <-O OFFSET> <FILENAME> | 2-14 |
| GO | EXECUTE | GO <ADDR> | 2-15 |
| GT | Go TILL BREAKPOINT | GT <ADDR> | 2-16 |
| HELP | HELP | HELP <COMMAND> | 2-17 |
| IRD | INTERNAL REGISTER DISPLAY | IRD <MODULE.REGISTER> | 2-18 |
| IRM | INTERNAL REGISTER MODIFY | IRM <MODULE.REGISTER> <DATA> | 2-19 |
| MD | MEMORY DISPLAY | MD <WIDTH> <BEGIN> <END> | 2-20 |
| MM | MEMORY MODIFY | MM <WIDTH> ADDR <DATA> | 2-21 |
| RD | REGISTER DISPLAY | RD <REG> | 2-22 |
| RM | REGISTER MODIFY | RM REG DATA | 2-23 |
| RESET | RESET | RESET | 2-24 |
| SET | SET CONFIGURATIONS | SET OPTION <VALUE> | 2-25 |
| SHOW | SHOW CONFIGURATIONS | SHOW OPTION | 2-27 |
| STEP | STEP (OVER) | STEP | 2-28 |
| SYMBOL | SYMBOL MANAGEMENT | SYMBOL <SYMB> <-A SYMB VALUE> <-R SYMB> <-C \| L \| S> | 2-29 |
| TRACE | TRACE(INTO) | TRACE <NUM> | 2-30 |
| UPDBUG | UPDATE DBUG | UPDBUG | 2-31 |
| UPUSER | UPDATE USER FLASH | UPUSER | 2-32 |
| VERSION | SHOW VERSION | VERSION | 2-33 |

### 2.4.1 BF - Block of Memory Fill BF

Usage: BF<width> begin end data

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

The value for addresses begin and end 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 0x00010000 and ending at 0x00040000 with the value 0x1234, the command is:

```
bf      10000 40000 1234
```

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

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

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

```
bf      bss_start bss_end 0
```

### 2.4.2 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 0x00040000 and ending at 0x00080000 to the location 0x00200000, the command is:

```
bm      40000 80000 200000
```

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

```
bm      data_start data_end 200000
```

### 2.4.3 BR - Breakpoint BR

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

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

The -r option to the BR command removes a breakpoint defined at address addr. If no address is specified in conjunction with the -r option, then all breakpoints are removed.
Each time a breakpoint is encountered during the execution of target code, its count value is incremented by one. By default, the initial count value for a breakpoint is zero, but the -c option allows setting the initial count for the breakpoint.

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

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

Examples:

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

```
br      _main
```

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

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

```
br      _bench -t 3
```

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

To remove all breakpoints, the command is:

```
br      -r
```

### 2.4.4 BS - Block Search BS

Usage: BS<width> begin end data

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

The value for addresses begin and end 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 0x00040000 and ending at 0x00080000 the command is:

```
bs      40000 80000 1234
```

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

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

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

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

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

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

### 2.4.5 DATA - Data Conversion DATA

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
```

### 2.4.6 DI - Disassemble DI

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 0x00040000, the command is:

```
di      40000
```

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

```
di      _main
```

### 2.5.7 DL - Download Serial DL

Usage: DL <offset>

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

If offset is provided, 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 (0x00000000-0x00010000), 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
```

### 2.4.8 DN - Download Network DN

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

The DN command downloads code from the network. The DN command handle files which are either S-record, COFF or ELF formats. The DN command uses Trivial File Transfer Protocol, TFTP, to transfer files from a network host. This command only works with 100% NE2000 compatible boards.

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

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

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

For ELF and COFF files which contain symbolic debug information, the symbol tables are extracted from the file during download and used by dBUG. Only global symbols are kept in dBUG. The dBUG symbol table is not cleared prior to downloading, so it is the user's responsibility to clear the symbol table as necessary prior to downloading.

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

Examples:

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

```
dn -s srec.out
```

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

```
dn -c coff.out
```

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

```
dn bench.out
```

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

```
dn
```

**This command requires proper Network address and parameter setup. Refer to Appendix A for this procedure. Also refer to "SET" command to setup the base address and the IRQ for the card.**

### 2.4.9 Go - Execute GO

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 0x00040000, the command is:

```
go 40000
```

### 2.4.10 GT - Execute Till a Temporary Breakpoint GT

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 0x10000, the command is:

GT 10000

### 2.4.11 HELP - Help HE

Usage: HELP <command>

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of where user code 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 screen full and ask for an input to display the rest of the list.

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

```
help br
```

### 2.4.12 IRD - Internal Registers Display IRD

Usage: IRD <module.register>

This commands displays the internal registers of different modules inside the MCF5206. 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 MCF5206 are SIM, UART1, UART2, TIMER, M-Bus, DRAMC, and Chip-Select. Refer to MCF5206 User's Manual.

Example:

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

### 2.4.13 IRM - Internal Registers MODIFY IRM

Usage: IRM module.register data

This commands modifies the contents of the internal registers of different modules inside the MCF5206. 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 MCF5206 are SIM, UART1, UART2, TIMER, M-Bus, DRAMC, Chip-Select. Refer to MCF5206 User's Manual.

Example:

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

### 2.4.14 MD - Memory Display MD

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

The MD command displays a contiguous block of memory starting at address begin and stopping at address end. The value for addresses begin and end 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 0x00400000, the command is:

```
md 400000
```

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

```
md data_start
```

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

```
md.b    40000 50000
```

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

```
md.l    40000 50000
```

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

### 2.2.15 MM - Memory Modify MM

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 0x00010000 to be 0xFF, the command is:

```
mm.b    10000 FF
```

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

```
mm      10000
```

### 2.4.16 RD - Register Display RD

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 0x00010000 to be 0xFF, the command is:

```
mm.b    10000 FF
```

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

```
mm      10000
```

### 2.4.17 RM - Register Modify RM

Usage: RM reg data

The RM command modifies the contents of the register reg to data. The value for reg is the name of the register, and the value for data 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
```

### 2.4.18 RESET - Reset the board and dBUG
### RESET

Usage: RESET

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

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

Examples:

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

```
reset
```

### 2.4.19 SET - Set Configuration SET

Usage: SET option <value>
SET

The SET command allows the setting of user configurable options within dBUG. The options are listed below. If the SET command is issued without 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).

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

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

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

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

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

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

autoboot - This option allows for the automatic downloading and execution of a file from the network. This option can be used to automatically boot an operating system from the network. Valid values are: “on” and “off”. This option is not implemented on the current of dBUG.

nicbase - this is base address of the network interface card. When using network card, the base address of that card is needed by the dBUG in order to communicate with it. This command is used *to inform the dBUG* of the address of the card. dBUG does not set or configure the interface card. It only uses this address to access the card. The user should provide this information to dBUG.

nicirq - this is the IRQ used in the network interface card. When using network card, the IRQ used by that card is needed by the dBUG in order to communicate with it. This command is used *to inform the dBUG* of the IRQ of the card. dBUG does not set or configure the interface card. It only uses this IRQ to access the card. The user should provide this information to dBUG.

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.

In order to use the KNE2000TLC Ethernet ISA card in the system, the debugger need to know its IRQ and its base address. The Kingston Technology Corporation Ethernet card KNE2000TLC has a default base address of $300 and uses IRQ3. To set up the debugger for Ethernet communication, the following commands should be issued first.

```
set     nicbase   300
set     nicirq     3
```

### 2.4.20 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
```

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

```
show server
```

### 2.4.21 STEP - Step Over ST

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
```

### 2.4.22 SYMBOL - Symbol Name Management SYMBOL

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

The SYMBOL command adds or removes symbol names from the symbol table. If only a symbol name is provided to the SYMBOL command, 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 0x00040000, the command is:

```
symbol      -a main 40000
```

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

```
symbol      -r junk
```

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

```
symbol      -s
```

To display the symbol table, the command is:

```
symbol      -l
```

### 2.4.23 TRACE - Trace Into TR

Usage: TRACE <num>

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

### 2.4.24 UPDBUG - Update the dBUG Image UPDBUG

Usage: UPDBUG

The UPDBUG command is used for updating the dBUG image in Flash. When updates to the MCF5206 EVS dBUG are available, the updated image is downloaded to address 0x00010000. 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!

### 2.4.25 UPUSER - Update User Code In Flash UPUSER

Usage: UPUSER

The UPUSER command places user code and data into space allocated for the user in Flash, the last 128K of Flash ROM. To place code and data in user Flash, the image is downloaded to address 0x00010000, and the UPUSER command issued. This commands programs the entire upper 128K of Flash. Users access this space starting at address 0xFFE20000.

### 2.4.26 VERSION - Display dBUG Version 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. This function can be called by the user program to utilize various routines within the dBUG, to perform a special task, and to return control to the dBUG. This section describes the TRAP 15 handler and how it is used.

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

### 2.5.1 OUT_CHAR

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

Assembly example:

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

C example:

```
void board_out_char (int ch)
{
        /* If your C compiler produces a LINK/UNLK pair for this routine,
         * then use the following code which takes this into account
        */
#if  1
        /* LINK a6,#0 -- produced by C compiler */
        asm (" move.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:

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

C example:

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

```
        asm (" move.l   d1,d0");                /* put the character in d0 */
}
```

### 2.5.3 CHAR_PRESENT

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

Assembly example:

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

C example:

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

### 2.5.4 EXIT_TO_dBUG

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

Assembly example:

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

C example:

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

# CHAPTER 3

# HARDWARE DESCRIPTION AND RECONFIGURATION

This chapter provides a functional description of the SBC5206 board hardware. With the description given here and the schematic diagram 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 CPU and general supporting logic on the SBC5206 board.

### 3.1.1 The Processor

The microprocessor used in the SBC5206 is the highly integrated MCF5206, 32-bit processor. The MCF5206 uses a ColdFire processor as the core with 512 bytes of instruction cache, two UART channels, two Timers, 512 bytes of SRAM, Motorola M-Bus Module supporting the $I^2C$, one-byte wide parallel I/O port, and the supporting integrated system logic. All the registers of the core processor are 32 bits wide except for the Status Register (SR) which is 16 bits wide. This processor communicates with external devices over a 32-bit wide data bus, D0-D31 with support for 8 and 16-bit ports. This chip can address the entire 4 G Bytes of memory space using internal chip-select logic. However, it provides only 28 address lines, A0-A27. All the processor's signals are available at J7, J9, and J10 for off the board expansion. Refer to section 3.7 for pin assignment.

The MCF5206 has an IEEE JTAG-compatible port and BDM port. These signals are available at J8. The processor also has the logic to generate up to eight (8) chip selects, -CS0 to -CS8 and two banks of DRAM's.

### 3.1.2 The Reset Logic

The reset logic provides system initialization under two modes. Under system power-up and when the RESET switch, S2 (red switch), is activated. The power-on generates the Master RESET by asserting the -RSTI and -HIZ which causes total system reset. The RESET switch generates Normal Reset which resets the entire processor except the DRAM controller.

U9 is used to produce active low power-on RESET signal which feeds the LSI2032 (U4) along with the Push-button RESET. The U4 device generates the system reset (-RESET) and ISA bus RESET signals.

dBUG performs the following configurations of internal resources during the initialization. The instruction cache is invalidated and disabled. The Vector Base Register, VBR, points to the Flash. However, a copy of the exception table is made at address $00000000 in DRAM. 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 $00000000, and then points the VBR to $00000000.

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 I/O.

### 3.1.3 The -HIZ Signal

*The -HIZ signal is actively driven by the LSI2032 (U4). This Signal is available for monitor on J10. However, this signal should not be driven by the user. If the user need to drive the -HIZ, it should be*

*done through -HIZ_INLOW signal which is available on J7. The -HIZ_INLOW signal feeds the U4 which drives the -HIZ signal to the processor.*

### 3.1.4 The Clock Circuitry

The SBC5206 uses a 25MHZ oscillator (U8) to provide the clock to CLK pin of the processor. This clock also feeds to LSI2032 for its internal use and to produce clock for the ISA timings and MC68HC901 (1/4 system clock).

### 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 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 which physically does not exist. In this case, the bus cycle will go on for ever, 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 MCF5206 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 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 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 status register), it will perform an interrupt acknowledge cycle at the end of the current instruction cycle. This interrupt acknowledge cycle indicates to the source of the interrupt that the request is being acknowledged and the device should provide the proper vector number to indicate where the service routine for this interrupt level is located. If the source of interrupt is not capable of providing a vector, its interrupt should be set up as autovector interrupt which directs the processor to a predefined entry into the exception table (refer to the MCF5206 User's Manual).

The processor goes to different 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 $00000000. 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 $00000000, and then points the VBR to $00000000.

The MCF5206 has three external interrupt request lines (-IRQ1, -IRQ4, -IRQ7) which can be set up either as -IPL0-2 lines for encoded interrupt request or dedicated -IRQ lines (1, 4, and 7). The SBC5206 configures these lines as -IRQ lines. There are also six internal interrupt requests from Timer1, Timer2, Software watchdog timer, UART1, UART2, and MBUS. Each interrupt source , external and internal, can be programmed for any priority level. In case of similar priority level, a second relative priority between 1 to 3 will be assigned.
On SBC5206, the internal Timers, Software Watchdog Timer, M-Bus, and UART's are disabled and not used. However, 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 SBC5206 uses -IRQ7 to support the ABORT function using the ABORT switch S1 (black switch). This switch is used to force a non-maskable interrupt (level 7, priority 3) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT). Since

the ABORT switch is not capable of generating a vector in response to level seven interrupt acknowledge from the processor, the debugger programs this request for autovector mode.

The MC68HC901 reports its interrupt request on -IRQ4 line which is set for Level 4, priority 2. It uses the vectored mode for acknowledgment. The chip-select -CS1 is used to generate the -IACK signal for MC68HC901. The MC68HC901 is programmed to generate vectors $F0 to $FF. This should not be changed.

The -IRQ1 line of the MCF5206 is 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 external interrupt request. Refer to MCF5206 User's Manual for more information about the interrupt controller.

### 3.1.7 Internal SRAM

The MCF5206 has 512 bytes of internal memory. This memory is mapped to $20000000 and is not used by the dBUG. It is available to the user.

### 3.1.8 The MCF5206 Registers and Memory Map

The memory and I/O resources of the SBC5206 are divided into three groups, MCF5206 Internal, External resources, and the ISA Bus address. All the I/O registers are memory mapped.

The MCF5206 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 DRAM's. There are eighteen (32) registers to specify the address range, type of access, and the method of -TA generation for each chip-select and -RAS pins. These registers are programmed by dBUG to map the external memory and I/O devices.

The SBC5206 uses chip-select zero (-CS0) to enable the EPROM/Flash ROM ( refer to Section 3.3.) The SBC5206 uses -RAS1 and -RAS2 to enable the DRAM SIMM module (refer to Section 3.2), -CS2 for enabling the MC68HC901, -CS1 for Interrupt acknowledge of MC68HC901, and -CS3 for ISA Bus I/O space.

The chip-select signals -CS4, -CS5, -CS6, and -CS7 share their pins with address lines A24, A25, A26, and A27 and the write-enable lines -WE3, -WE2, -WE1, and -WE0. The pins for -CS6 and -CS7 are programmed as write enable line -WE1 and -WE0 respectively to support the on-board Flash ROM. The pins for -CS4 and -CS5 are programmed as chip select lines.

The chip select mechanism of the MCF5206 allows the memory mapping to be defined based on the memory space desired (User/Supervisor, Program/Data spaces).

All the MCF5206 internal registers, configuration registers, parallel I/O port registers, DUART registers and system control registers are mapped by MBAR register at 1K-byte boundary. It is mapped to $10000000 by dBUG. For complete map of these registers refer to the MCF5206 User's Manual.

The SBC5206 board can have up to 32M bytes of DRAM installed. The first 32M bytes are reserved for this memory. Refer to Section 3.2 for a discussion of RAM. The dBUG is programmed in two 29F010 Flash ROM's which only occupies 256K bytes of the address space. The first 128K bytes are used by dBUG and the second half is left for user. Refer to section 3.3.

The MC68HC901 is used as dBUG serial communication, baud rate generator, and ISA Bus interrupt request. Refer to section 3.4.2.

The ISA Bus interface maps all the I/O space of the ISA bus to the MCF5206 memory at address $04000000. Refer to section 3.6.

**TABLE 3.1. The SBC5206 memory map.**

| ADDRESS RANGE | SIGNAL and DEVICE |
|---|---|
| $00000000-$01FFFFFF[1] | -RAS1, -RAS2, Up to 32M bytes of DRAM's. |
| $10000000-$100003FF | Internal Module registers |
| $20000000-$200001FF | Internal SRAM |
| $30000000-$300FFFFF | -CS2, 1M space for MC68HC901. -CS1 is used for IACK. |
| $40000000-$400FFFFF | -CS3, 1M ISA Bus area |
| $FFE00000-$FFE3FFFF | -CS0, 256K bytes of Flash ROM. |

1 Refer to the text for more detail.

All the unused area 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 $000000-$000007 (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 MCF5206, the -CS0 responds to any accesses after reset until the CSMR0 is written. This includes the reset vector range. Since -CS0 is connected to Flash ROM's, the Flash ROM's 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, programs the chip-select logic and locates the Flash ROM's to start at $FFE00000 and the DRAM's 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 MCF5206, as part of the chip-select logic, has a built in mechanism to generate the -TA for all external devices which do not have the capability to generate the -TA on their own. The Flash ROM's and DRAM's can not generate the -TA. Their chip-select logic's are programmed by dBUG to generate the -TA internally after a preprogrammed number of wait states. In order to support the future expansion of the board, the -TA input of the processor is also connected to the Processor Expansion Bus, J9. This allows the 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 board. All the -TA's from the expansion boards should be connected to this line.

### 3.1.11 Wait State Generator

The Flash ROM's and DRAM SIMM on the board may require some adjustments on the cycle time of the processor to make them compatible with processor speed. To extend the CPU bus cycles for the slower devices, the chip-select logic of the MCF5206 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 DRAM's and Flash ROM's respectively.

### 3.2 THE DRAM SIMM

The SBC5206 has one 72-pin SIMM socket (U15) for DRAM SIMM. This socket supports DRAM SIMM's of 256Kx32, 1Mx32, 2Mx32, 4Mx32, and 8Mx32. No special configuration is needed. The dBUG will detect the total memory installed on power-up. The SIMM speed should be 70ns. The DRAM Access timing is 3,2,2,2.

### 3.3 THE EPROM/FLASH ROM

There are two sockets for EPROM's/Flash ROM's on the SBC5206, U13 (high, even byte) and U10 (low, odd byte). These sockets support 64K, 128K, 256K, 512K, and 1M-byte EPROM's such as 27C256, 27C512, 27C010, 27C020, 27C040, and 27C080 chips for a total of up to 2M bytes. The sockets also support the Flash ROM's such as 29F010 and 29F040 which are 5-volt only devices.

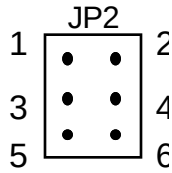
If the user wishes to modify the size or the type of the memory chips, the jumpers JP2 and JP3 should be modified to accommodate different size and type of memory chips. Refer to Figure 3.1 for jumper selection. Although, the board supports different types of Flash ROM's and EPROM's, the dBUG Flash driver is setup for 29F010 device only.

The board is shipped with two 29F010, 128K-byte, FLASH ROM's for a total of 256K bytes. The first 128K of the Flash contains dBUG firmware. The second half (last 128K) is available to the user. The high byte (even address) chip is installed in U13 socket and the low byte (odd address) chip is installed in U10 socket. The chip-select signal generated by the MCF5206 (-CS0) enables both chips.

The MCF5206 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.

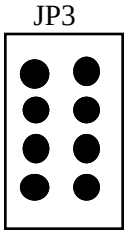
JP2 Configuration

| MEMORY TYPE | JUMPER SETUP |
|---|---|
| FLASH ROM | Connect 3 to 5 and 4 to 6 (default) |
| EPROM | Connect 1 to 3 and 2 to 4 |




JP3 Configuration for EPROM

| MEMORY SIZE | JUMPER SETUP |
|---|---|
| 27C512 (512K EPROM) | 6 to 8 |
| 27C010 (1M EPROM) | 3 to 5 and<br>4 to 6 and 8 to 10 |
| 27C020 (2M EPROM) | 3 to 5 and<br>4 to 6 and 8 to 10 |
| 27C040 (4M EPROM) | 3 to 5 and<br>2 to 4 and 8 to 10 |
| 27C080 (8M EPROM) | 1 to 3 and<br>2 to 4 and 8 to 10 |




JP3 Configuration for FLASH ROM

| MEMORY SIZE | JUMPER SETUP |
|---|---|
| 29F010 (1M Flash) and 29F040 (4M Flash) | 7 to 9 and 8 to 10 |

Note: Only connect the pins specified. Leave the rest open.

**Figure 3.1. Jumper setup for the Flash/EPROM sockets.**

## 3.4 THE SERIAL COMMUNICATION CHANNELS

The SBC5206 offers a number of serial communications. They are discussed in this section.

### 3.4.1 The MCF5206 DUART

The MCF5206 has two built in UART's, These serial channels with software programmable baud rate generators are not used by the SBC5206 or dBUG and are available to the user. The dBUG, 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 these channels are available on J7. The signals of UART1 are also passed through the RS-232 driver/receiver and are available on DB-9 connector J6. Refer to the MCF5206 User's Manual for programming and the register map.

### 3.4.2 MC68HC901

To provide the board with one independent serial communication channel for dBUG communication with terminal or PC, an MC68HC901 is used. This device provides four timer channels (A, B, C, and D), one serial communication channel, and 8 input lines. Channel D timer is used as the baud rate generator for the serial communication channel.

The clock source for the timers is the 2.4576MHZ crystal. The clock signal to drive the MC68HC901 logic is one-fourth of the processor's clock.

The TXD (SO) signal and the RXD (SI) signal are passed through RS-232 driver/receiver and are available on J5. The eight input lines are used to report the ISA Bus interrupts (IRQ3, IRQ4, IRQ5, IRQ6, IRQ7, IRQ9, IRQ10, and IRQ11). The interrupt from MC68HC901 is reported to MCF5206 on -IRQ4 of the MCF5206. The interrupt level for the MC68HC901 is set for Level 4 with priority 2. The vectors used for MC68HC901 are $F0 to $FF. It generates 16 vectors. This should not be changed.

The -CS2 is used to access the MC68HC901 internal registers, it is mapped to $30000000. The -CS1 is programmed to generate an Interrupt Acknowledge signal to drive the -IACK of the MC68HC901. Refer to MC68HC901 User's Manual for functional description and the programming model.

### 3.4.3 Motorola Bus (M-Bus) Module

The MCF5206 has a built in M-Bus module which allows interchip bus interface for a number of I/O devices. It is compatible with industry-standard $I^2C$ Bus. The SBC5206 does not use this module and it is available to the user. The two M-Bus signals are SDA and SCL which are available at J7 connector. These signals are open-collector signals. However, they have pull-up resistors on the SBC5206. The interrupt control register for M-Bus is set for Level 3, priority 0 and autovector.

## 3.5 THE PARALLEL I/O Port

The MCF5206 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 I/O pins.

### 3.6 THE ISA BUS LOGIC

The SBC5206 includes the necessary logic, drivers, and the connector (P1) to allow the use of off-the-shelf ISA Bus I/O cards. The slot can be used with 8- or 16-bit ISA cards. The ISA-space addresses are located starting at $40000000.

The main purpose for this setup is to allow the use of Ethernet card (NE2000 compatible) to facilitate network down load, refer to chapter 2 for network download command (DN). The dBUG driver only accepts 100% NE2000 compatible cards.

The ISA Bus interrupt request lines IRQ3, IRQ4, IRQ5, IRQ6, IRQ7, IRQ9, IRQ10, and IRQ11 are connected to I0 to I7 of the MC68HC901. The requested interrupt is then routed to -IRQ4 of the MCF5206.

### 3.7 THE CONNECTORS AND THE EXPANSION BUS

There are eleven connectors on the SBC5206 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.

#### 3.7.1 The Programming Connector J1

The J1 connector is used to program the ispLSI2032. This Connector is not a user connector.

**TABLE 3.2. The J1 Connector Pin Assignment.**

| PIN NO. | SIGNAL NAME |
|---|---|
| 1 | +5 Volts |
| 2 | -SDO |
| 3 | -SDI |
| 4 | -ISPEN |
| 5 | No Connect (key) |
| 6 | -MODE |
| 7 | GND |
| 8 | SCLK |

#### 3.7.2 The ISA Bus Auxiliary Power Connector J2

The ISA Bus requires +/-12 and -5 as well as +5 volts supply. Since they are not always needed the connector used for these is a simple burg connector. Table 3.3 shows the Pin assignment for J2.

**TABLE 3.3. The J2 Connector pin assignment.**

| PIN NUMBER | SIGNAL NAME |
|---|---|
| 1 | +12 Volts |
| 2 | Ground |
| 3 | -12 Volts |
| 4 | -5 Volts |

#### 3.7.3 The Power Supply Connectors J3 and J4

The SBC5206 needs +5 volts supply (less than an Amp.). The power can be +5 Volts regulated ***or*** +7.5 to +12 Volts DC (regulated or unregulated) which utilizes the on board regulator U14. Jumper JP1 (Table 3.6) makes the selection between +5 Volts regulated and the +7.5-12 Volts supply. If pins 1 and 2 are

connected, the board needs external +5 Volts regulated supply. If pins 2 and 3 are connected, then a DC supply of +7.5 to +12 volts may be used. In either case, the power may be connected to the board through J3 (2.1mm power jack) or the J4 two-contact lever actuated terminal block. On J3 the center pin (pin 1) is the plus supply and the body (pin 3) is the ground. On J4 the pin 1 is the plus supply and pin 2 is the ground. Tables 3.4 and 3.5 show the Pin assignment for J3 and J4.

**TABLE 3.4. The J3 Connector pin assignment.**

| PIN NUMBER | SIGNAL NAME |
|---|---|
| 1 (center pin) | Plus Supply |
| 2 (body) | Ground |

**TABLE 3.5. The J4 Connector pin assignment.**

| PIN NUMBER | SIGNAL NAME |
|---|---|
| 1 | Plus Supply |
| 2 | Ground |

**TABLE 3.6. The Jumper JP1.**

| Jumper Pins | Selection |
|---|---|
| 1 to 2 | regulated +5 Volts |
| 2 to 3 | +7.5 to +12 DC regulated or unregulated |

### 3.7.4 The Terminal Connector J5

The SBC5206 uses a 9-pin D-sub female connector J5 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 3.7 shows the pin assignment.

**TABLE 3.7. The J5 (TERMINAL) Connector pin assignment.**

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

### 3.7.5 The Auxiliary Serial Communication Connector J6

The MCF5206 has two built-in UART's. These channels are not used by the SBC5206 dBUG and they are available to the user. The signals of these channels are available on J7. The signals on UART1 are also run trough RS-232 driver/receivers and are available on J6. The available signals form a working subset of the RS-232C standard. Table 3.8 shows the pin assignment for J6.

**TABLE 3.8. The J6 Connector pin assignment.**

| PIN NO. | DIRECTION | SIGNAL NAME |
|---|---|---|
| 1 | Output | Connected to pin 6 and 8 |
| 2 | Output | Receive Data |
| 3 | Input | Transmit Data |
| 4 | | No connect |
| 5 | | Signal Ground |
| 6 | Output | Connected to 1 and 8 |
| 7 | Input | Clear to Send |
| 8 | Output | Request to Send (connected to 1 & 6) |
| 9 | | Not Used |

### 3.7.6 The Processor Expansion Bus J7, J9, and J10

All the processors signals are available on three burg headers J7, J9 and J10 for future expansion. Although these signals are not buffered, they can drive at least one TTL load with some having more than one TTL load capability. 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. All the primary signals to/from the processor needed for simple memory expansion are available on J9. The remaining expansion signals and I/O signals are available on J7 and J10. Therefore, a single 60-wire flat ribbon cable with the IDC connectors may be used for most of future expansions. Tables 3.9, 3.11 and 3.12 show the pin assignment for J7, J9, and J10 respectively.

### 3.7.7 The Debug Connector J8

The MCF5206 does have background Debug Port, Real-Time Trace Support, and Real-Time Debug Support. The necessary signals are available at connector J8. Table 3.10 shows the pin assignment.

### 3.7.8 The ISA Bus Connector P1

The SBC5206 can utilize the ISA Bus 16-bit I/O cards. The P1 connector is ISA Bus compatible connector. Table 3.13 shows the pin assignment.

**TABLE 3.9. The J7 Connector pin assignment.**

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

**TABLE 3.10. The J8 Connector pin assignment.**

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

**TABLE 3.11. The J9 Connector pin assignment.**

| PIN NO. | SIGNAL NAME |
|---|---|
| 1 | A0 |
| 2 | D16 |
| 3 | A1 |
| 4 | D17 |
| 5 | A2 |
| 6 | D18 |
| 7 | A3 |
| 8 | D19 |
| 9 | A4 |
| 10 | D20 |
| 11 | A5 |
| 12 | D21 |
| 13 | A6 |
| 14 | D22 |
| 15 | A7 |
| 16 | D23 |
| 17 | A8 |
| 18 | D24 |
| 19 | A9 |
| 20 | D25 |
| 21 | A10 |
| 22 | D26 |
| 23 | A11 |
| 24 | D27 |
| 25 | A12 |
| 26 | D28 |
| 27 | A13 |
| 28 | D29 |
| 29 | A14 |
| 30 | D30 |
| 31 | A15 |
| 32 | D31 |
| 33 | A16 |
| 34 | TT0 |
| 35 | A17 |
| 36 | TT1 |
| 37 | A18 |
| 38 | ATM |
| 39 | A19 |
| 40 | SIZ0 |
| 41 | A20 |
| 42 | SIZ1 |
| 43 | A21 |
| 44 | R/-W |
| 45 | A22 |
| 46 | -TS |
| 47 | A23 |
| 48 | -TA |
| 49 | A24/-CS4/-WE3 |
| 50 | -TEA |
| 51 | A25/-CS5/-WE2 |
| 52 | -ATA |
| 53 | A26/-CS6/-WE1 |
| 54 | -RESET |
| 55 | A27/-CS7/-WE0 |
| 56 | -IRQ1 |
| 57 | CLK |
| 58 | +5 Volts |
| 59 | Ground |
| 60 | Ground |

**Table 3.12. The J10 Connector pin assignment.**

| PIN NO. | SIGNAL NAME |
|---|---|
| 1 | TCK |
| 2 | D0 |
| 3 | DSCLK |
| 4 | D1 |
| 5 | DSI |
| 6 | D2 |
| 7 | DSO |
| 8 | D3 |
| 9 | -BKPT |
| 10 | D4 |
| 11 | +5 Volts |
| 12 | D5 |
| 13 | Ground |
| 14 | D6 |
| 15 | PP0/DDATA0 |
| 16 | D7 |
| 17 | PP1/DDATA1 |
| 18 | D8 |
| 19 | PP2/DDATA2 |
| 20 | D9 |
| 21 | PP3/DDATA3 |
| 22 | D10 |
| 23 | PP4/PST0 |
| 24 | D11 |
| 25 | PP5/PST1 |
| 26 | D12 |
| 27 | PP6/PST2 |
| 28 | D13 |
| 29 | PP7/PST3 |
| 30 | D14 |
| 31 | -HIZ |
| 32 | D15 |
| 33 | MTMOD0 |
| 34 | -CAS0 |
| 35 | -RAS0 |
| 36 | -CAS1 |
| 37 | -RAS1 |
| 38 | -CAS2 |
| 39 | -DRAMW |
| 40 | -CAS3 |

**TABLE 3.13. The P1 Connector pin assignment.**

| PIN NO. | SIGNAL NAME | PIN NO. | SIGNAL NAME |
|---|---|---|---|
| 1 | GND | 2 | IOCHK* |
| 3 | RESET | 4 | SD7 |
| 5 | +5V | 6 | SD6 |
| 7 | IRQ9 | 8 | SD5 |
| 9 | -5V | 10 | SD4 |
| 11 | DRQ2 | 12 | SD3 |
| 13 | -12V | 14 | SD2 |
| 15 | ZWS* | 16 | SD1 |
| 17 | +12V | 18 | SD0 |
| 19 | GND | 20 | IOCHRDY |
| 21 | SMFMW* | 22 | AEN |
| 23 | SMFMR* | 24 | SA19 |
| 25 | IOW* | 26 | SA18 |
| 27 | IOR* | 28 | SA17 |
| 29 | DACK3* | 30 | SA16 |
| 31 | DRQ3 | 32 | SA15 |
| 33 | DACK1* | 34 | SA14 |
| 35 | DRQ1 | 36 | SA13 |
| 37 | REFSH | 38 | SA12 |
| 39 | SYSCLK | 40 | SA11 |
| 41 | IRQ7 | 42 | SA10 |
| 43 | IRQ6 | 44 | SA9 |
| 45 | IRQ5 | 46 | SA8 |
| 47 | IRQ4 | 48 | SA7 |
| 49 | IRQ3 | 50 | SA6 |
| 51 | DACK2* | 52 | SA5 |
| 53 | TC | 54 | SA4 |
| 55 | BALE | 56 | SA3 |
| 57 | +5V | 58 | SA2 |
| 59 | OSC | 60 | SA1 |
| 61 | GND | 62 | SA0 |
| 63 | MEMCS16* | 64 | SBHE* |
| 65 | IOCS16 | 66 | LA23 |
| 67 | IRQ10 | 68 | LA22 |
| 69 | IRQ11 | 70 | LA21 |
| 71 | IRQ12 | 72 | LA20 |
| 73 | IRQ15 | 74 | LA19 |
| 75 | IRQ14 | 76 | LA18 |
| 77 | DACK0* | 78 | LA17 |
| 79 | DRQ0 | 80 | MEMB* |
| 81 | DACK5* | 82 | MEMW* |
| 83 | DRQ5 | 84 | SD8 |
| 85 | DACK6* | 86 | SD9 |
| 87 | DRQ6 | 88 | SD10 |
| 89 | DACK7* | 90 | SD11 |
| 91 | DRQ7 | 92 | SD12 |
| 93 | +5V | 94 | SD13 |
| 95 | MASTER* | 96 | SD14 |
| 97 | GND | 98 | SD15 |

### 3.8 THE SBC5206 JUMPERS

There are a total of three jumpers on the SBC5206 board to configure the board for different setup. Table 3.14 shows what these jumpers are for and the section where more information can be found.

**TABLE 3.14. The SBC5206 Jumpers.**

| Jumper No. | Function (section) |
|---|---|
| JP1 | Power Supply Selection, (section 3.7.3) |
| JP2 | Flash/EPROM type selection (section 3.3) |
| JP3 | Flash/EPROM size selection (section 3.3) |

# Appendix A

## A.1 Configuring dBUG for Network Downloads

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

### A1.1 Required Network Parameters

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

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

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

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

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

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

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

### A.1.2 Configuring dBUG Network Parameters

Once the network parameters have been obtained, dBUG must be configured. The following commands are used to configure the network parameters.

**set client <client IP>**
**set server <server IP>**
**set gateway <gateway IP>**
**set netmask <netmask>**

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

set client 123.45.68.15
set server 123.45.67.1
set gateway 123.45.68.250
set netmask 255.255.255.0

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

Most, if not all, TFTP servers will only permit access to files starting at a particular sub-directory. (This is a security feature which prevents reading of arbitrary files by unknown persons.) For example, SunOS uses the directory /tftp_boot as the default TFTP directory. When specifying a filename to a SunOS

TFTP server, all filenames are relative to /tftp_boot. As a result, you normally will be required to copy the file to download into the directory used by the TFTP server.

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

**set filename <filename>**

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

**set filetype <srecord|coff|elf|image>**

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

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

Change the default filename and filetype with the commands:

set filename a.out
set filetype coff

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

### A.1.3 Troubleshooting Network Problems

Most problems related to network downloads are a direct result of improper configuration. Verify that all IP addresses configured into dBUG are correct. This is accomplished via the 'show' command.

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

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

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

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