SBC5307 USER'S MANUAL
REVISION 2.0

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, Motorola reserves the right to make changes to any product(s) herein to improve reliability, function, or design. The LAB5307rev2 board is not intended for use in life and/or property critical applications. Here, such applications are defined to be any situation in which any failure, malfunction, or unintended operation of the board could, directly, or indirectly, threaten life, result in personal injury, or cause damage to property. Although every effort has been made to make the supplied software and its documentation as accurate and functional as possible, Motorola Inc. will not assume responsibility for any damages incurred or generated by this product. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights, if any, or the rights of others.

WARNING

THIS BOARD GENERATES, USES, AND CAN RADIATE RADIO FREQUENCY ENERGY AND, IF NOT INSTALLED PROPERLY, MAY CAUSE INTERFERENCE TO RADIO COMMUNICATIONS. AS TEMPORARILY PERMITTED BY REGULATION, IT HAS NOT BEEN TESTED FOR COMPLIANCE WITH THE LIMITS FOR CLASS A COMPUTING DEVICES PURSUANT TO SUBPART J OF PART 15 OF FCC RULES, WHICH ARE DESIGNED TO PROVIDE REASONABLE PROTECTION AGAINST SUCH INTERFERENCE. OPERATION OF THIS PRODUCT IN A RESIDENTIAL AREA IS LIKELY TO CAUSE INTERFERENCE, IN WHICH CASE THE USER, AT HIS/HER OWN EXPENSE, WILL BE REQUIRED TO CORRECT THE INTERFERENCE.

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

TABLES

FIGURES

CHAPTER 1

INTRODUCTION TO THE SBC5307 BOARD

1.1 INTRODUCTION

The SBC5307 is a versatile single board computer based on MCF5307 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 8M SDRAM, SRAM, Timer, I/O, Ethernet, and 1M bytes of Flash. In addition, all of the signals are easily accessible to any logical analyzer with mictor probes to assist in debugging. Most of the processor's signals are also available via connectors J8 and J9 for expansion purposes.

1.2 GENERAL HARDWARE DESCRIPTION

The SBC5307 board provides the RAM, Flash ROM, on board NE2000 compatible Ethernet interface (10M bit/sec), RS232, and all the built-in I/O functions of the MCF5307 for learning and evaluating the attributes of the MCF5307. The MCF5307 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 MCF5307 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 MCF5307 User's Manual by Motorola for detail information about the SIM.) The SBC5307 only uses three of the chip selects to access the Flash ROM's, SRAM (which is not populated on board, may be added by the user) and the Ethernet. The DRAM controller is used to control one SIMM or one DIMM module 8M bytes of DRAM, both -RAS lines and all four –CAS lines are used. All other functions of the SIM are available to the user.

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

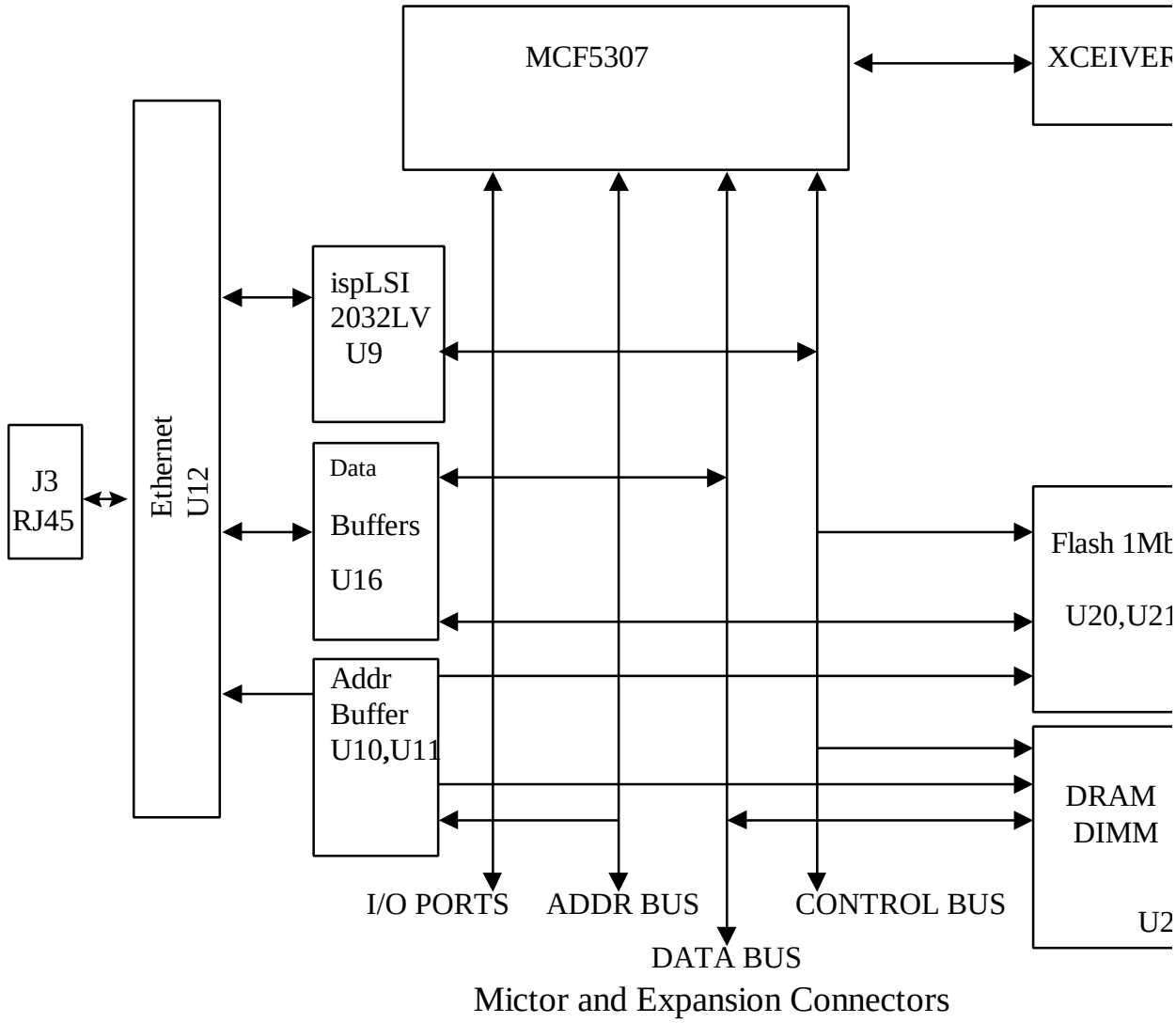


Figure 1 Block Diagram of the board

1.3 SYSTEM MEMORY

There are two on board Flash ROM's (U20, U21), U20 is the most significant byte and the U21 is the least significant byte. The SBC5307 comes with two 29LV004 Flash ROM's programmed with a debugger/monitor firmware. Both AM29LV004DT Flash are 4Mbits each giving a total of 1Mbyte of Flash memory. The dBUG only supports 29LV004 flash ROM.

There is one 168-pin DIMM socket for SDRAM. It currently supports 1M x 4 Bank x 16-Bits SDRAM totaling 8M of RAM.

The MCF5307 has 4K bytes organized as 1024x32 bits of internal SRAM.

The internal cache of the MCF5307 is a non-blocking, 8kbyte, 4-way set-associative, unified (instruction and data cache with a 16-byte line size. The ROM Monitor currently does not utilize the cache, but programs downloaded with the ROM Monitor can use the cache.

1.4 SERIAL COMMUNICATION CHANNELS

The MCF5307 has 2 built-in UART's with independent baud rate generators. The signals of channel one are passed through external Driver/Receivers to make the channel compatible with RS-232. UART1 is used by the debugger for the user to access with a terminal. In addition, the signals of both channels are available on the mictor connectors LA1 and LA3 to be viewed by a logic analyzer. UART1 channel is the "TERMINAL" channel used by the debugger for communication with external terminal/PC. The "TERMINAL' baud rate is set at 19200. The MCF5307 also incorporate the M-Bus, which is compatible with I^2C Bus standard. The I^2C bus is connected to the DIMM socket with an ID# = 0, however, the debugger does not use this feature.

1.5 PARALLEL I/O PORTS

MCF5307 offers one 16-bit general-purpose parallel I/O port. Each pin can be individually programmed as input or output. The parallel port bits PP (7:0) is multiplexed with TT (1:0), TM (2:0), DREQ (1:0), and XTIP. The second set of parallel port bits PP (15:8) is multiplexed with address bus bits A (31:24). Both bytes of the parallel port are controlled by the Pin Assignment Register (PAR). The pins are programmable on a pin by pin basis. The setting of the multiplex pins are determined by the configuration byte during reset. After reset, all pins are configured as general-purpose parallel I/O.

1.6 PROGRAMMABLE TIMER/COUNTER

The MCF5307 has two built in general purpose timer/counters. These timers are available to the user. The signals for the timer are available on the LA4 to be viewed by a logic analyzer.

1.7 ON BOARD ETHERNET

The SBC5307 has an on board Ethernet (NE2000 compatible) operating at 10M bits. The on board ROM MONITOR is programmed to allow a user to download files from a network to memory in different formats. The current formats supported are S-Record, COFF, ELF, or Image.

1.8 SYSTEM CONFIGURATION

The SBC5307 board requires only the following items for minimum system configuration (Figure 3):

1. The SBC5307 board (provided).
2. Power supply, 7.5V to 9V with minimum of 1.5 Amp.
3. RS-232C compatible terminal or a PC with terminal emulation software.
4. 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, be sure to check that all jumpers are in the default locations. The standard configuration does not require any modifications. After the board is functional in its standard configuration, you may use the Ethernet 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. SBC5307 Single Board Computer

b. SBC5307 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 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 4 shows the placement of the jumpers and the connectors, which you need to refer to in the following sections. The steps to be taken are:

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

1.9.3. Providing Power to the Board

The board accepts two means of power supply connections. Connector J5 is a 2.1mm power jack and J6 lever actuated connector. The board accepts 7.5V to 9V DC (regulated or unregulated) at 1.5 Amp via either one of the connectors.

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

1.9.4. Selecting Terminal Baud Rate

The serial channel of MCF5307 which is used for serial communication has a built in timer used by the ROM MONITOR to generate the baud rate used to communicate with a terminal.. It can be programmed to a number of baud rates. After the power-up or a manual RESET, the ROM Monitor firmware configures the channel for 19200 baud. After the ROM Monitor is running, you may issue the SET command to choose any baud rate supported by the ROM Monitor. Refer to Chapter 2 for the discussion of this command.

1.9.5. The Terminal Character Format

The character format of the communication channel is fixed at 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 RS-232 serial cable to connect the PC to the SBC5307. 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 J4 connector on SBC5307. 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 2 which shows the pin assignment for the 9-pin connector on the board.

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, Windows 95 Hyper Terminal or similar packages. Then connect as described in 1.9.6 Connecting the Terminal .

Once the connection to the PC is made, you are ready to power-up the PC and run the terminal emulation software. When 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 The Terminal Character Format . Then, select the baud rate as 19200. Now you are ready to apply power to the board.

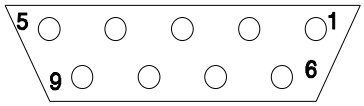


Figure 2 Pin assignment for J4 (Terminal) connector.

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

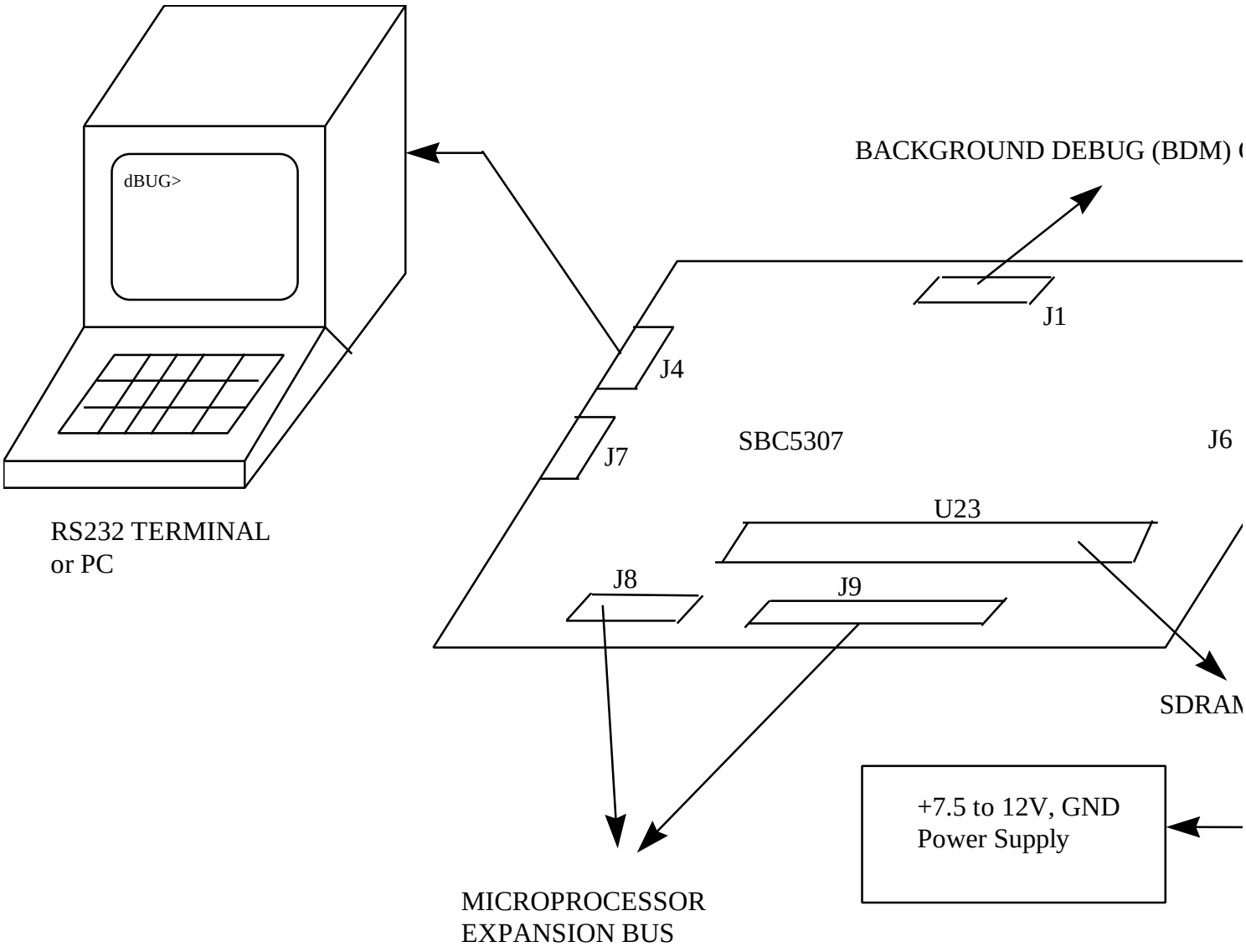


Figure 3 System Configuration

Figure 4 Jumper and connector placement

1.10 SYSTEM POWER-UP AND INITIAL OPERATION

Now that you have connected all the cables, you 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: 8M
NE2000: 0x300

Copyright 1997-1998 Motorola, Inc. All Rights Reserved.
ColdFire® MCF5307 EVS Debugger Vx.x.x (xxx 199x xx:xx:xx:)
Enter 'help' for help.

dBUG>

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

1. Make sure that the power supply is 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 black RESET button to insure that the board has been initialized properly.

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

1.11 SBC5307 Jumper Setup

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

1.11.1. Jumper JP1- Flash Upper Half/Lower Half Boot

This jumper allows the MC5307 to boot from the lower or upper half of the flash. The default is the lower half.

Table 1 – JP1, Upper/Lower Half BOOT

JP1	Function
1 and 2	Lower (default)
2 and 3	Upper

1.11.2. Jumper JP2 - This jumper selects between /CS0 to Flash or a header

Table 2 – JP2, /CS0 select

JP3	Function
1 and 2	Flash (default)
2 and 3	header

1.12 USING THE BDM

The MCF5307 has a built in debug mechanism referred to as BDM. The SBC5307 has the necessary connector, J1, 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 J1 connector. No special setting is needed. Refer to the BDM User's Manual for additional instructions.

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

CHAPTER 2

USING THE MONITOR/DEBUG FIRMWARE

The SBC5307 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 512Kx8 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 5. After the system initialization, the board waits for a command-line input from the user terminal. When a proper command is entered, the operation continues in one of the two basic modes. If the command causes execution of the user program, the dBUG firmware may or may not be re-entered, depending on the discretion of the user. For the alternate case, the

command will be executed under control of the dBUG firmware, and after command completion, the system returns to command entry mode.

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

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

.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 (J4) connector.
c. Turn power on to the board.

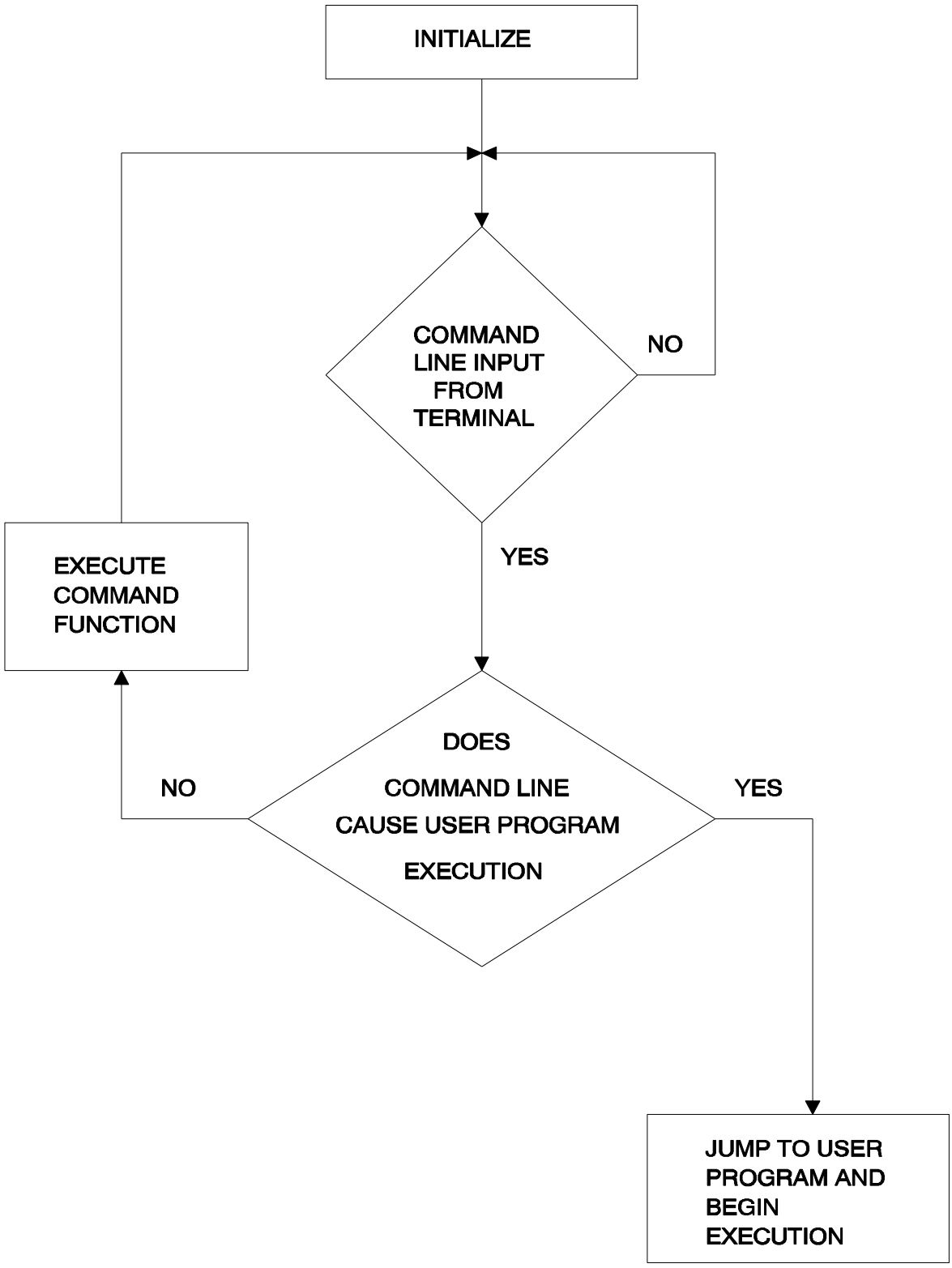


Figure 5 Flow Diagram of dBUG Operational Mode.

2.2.2. System Initialization

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

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

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

After initialization, the terminal will display:

Hard Reset
DRAM Size: 8M
NE2000: 0x300

Copyright 1997-1998 Motorola, Inc. All Rights Reserved.
ColdFire® MCF5307 EVS Debugger Vx.x.x (xxx 199x xx:xx:xx:)
Enter 'help' for help.

dBUG>

If you did not get this response check the setup. Refer to Section 1.10 SYSTEM POWER-UP AND INITIAL OPERATION. Note, the date ***'xxx 199x xx:xx:xx'*** may vary in different revisions.

Other means can be used to re-initialize the SBC5307 Computer Board firmware. These means are discussed in the following paragraphs.

2.2.2.1. Hard RESET Button.

Hard RESET is the red button located in the lower right side of the board. Depressing this button causes all processes to terminate, resets the MCF5307 processor and board logic's and restarts the dBUG firmware. Pressing the RESET button would be the appropriate action if all else fails.

2.2.2.2. ABORT Button.

ABORT is the black button located next to RESET button on the right side of the board. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5307) 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 MCF5307 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: 8M
NE2000: 0x300

Copyright 1997-1998 Motorola, Inc. All Rights Reserved.
ColdFire® MCF5307 EVS Debugger Vx.x.x (xxx 199x xx:xx:xx:)
Enter 'help' for help.

dBUG>

and waits for a command.

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

NOTES

1. The user memory is located at addresses $00020000-$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-$0001FFFF 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 MCF5307 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 3 lists the dBUG commands. Each of the individual commands is described in the following pages.

Table 3 - dBUG Commands

COMMAND MNEMONIC	DESCRIPTION	SYNTAX	PAGE
AS	ASSEMBLE	AS <addr> <instruction>	2-8
BF	BLOCK FILL	BF<WIDTH> BEGIN END DATA	2-11
BM	BLOCK MOVE	BM BEGIN END DEST	2-12
BS	BLOCK SEARCH	BS <WIDTH> BEGIN END DATA	2-14
BR	BREAKPOINT	BR ADDR <-R> <-C COUNT> <-T TRIGGER>	2-13
DATA	DATA CONVERT	DATA VALUE	2-15
DI	DISASSEMBLE	DI <ADDR>	2-16
DL	DOWNLOAD SERIAL	DL <OFFSET>	2-17
DN	DOWNLOAD NETWORK	DN <-C> <-E> <-S> <-I> <-O OFFSET> <FILENAME>	2-18
GO	EXECUTE	GO <ADDR>	2-19
GT	Go TILL BREAKPOINT	GT <ADDR>	2-20
HELP	HELP	HELP <COMMAND>	2-21
IRD	INTERNAL REGISTER DISPLAY	IRD <MODULE.REGISTER>	2-22
IRM	INTERNAL REGISTER MODIFY	IRM <MODULE.REGISTER> <DATA>	2-23
MD	MEMORY DISPLAY	MD <WIDTH> <BEGIN> <END>	2-24
MM	MEMORY MODIFY	MM <WIDTH> ADDR <DATA>	2-25
RD	REGISTER DISPLAY	RD <REG>	2-26
RM	REGISTER MODIFY	RM REG DATA	2-27
RESET	RESET	RESET	2-28
SET	SET CONFIGURATIONS	SET OPTION <VALUE>	2-29
SHOW	SHOW CONFIGURATIONS	SHOW OPTION	2-31
STEP	STEP (OVER)	STEP	2-32
SYMBOL	SYMBOL MANAGEMENT	SYMBOL <SYMB> <-A SYMB VALUE> <-R SYMB> <-C | L | S>	2-33
TRACE	TRACE(INTO)	TRACE <NUM>	2-34
UPDBUG	UPDATE DBUG	UPDBUG	2-35
UPUSER	UPDATE USER FLASH	UPUSER	2-36
VERSION	SHOW VERSION	VERSION	2-37

2.4.1. AS - Assemble *AS*

Usage: AS <addr> <instruction>

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

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

If no instruction is passed to the AS command, then AS prompts with the address where opcode will be written, and continues to assemble instructions until the user terminates the AS command by inputting a period, ".".

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

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

Constants and operands may be input in several different bases:

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

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

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

Examples:

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

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

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

```
                as      12000
```

then:

```
            0x00012000: start:  nop
```

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

2.4.2. BC - Compare Blocks of Memory *BC*

Usage: BC first second length

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

Examples:

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

```
bc    20000 FFE20000 20000    .
```

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

Usage: BF<width> begin end data

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

The value for addresses begin and end 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
```

2.4.4. *BM - Block Move* **BM**

Usage: BM begin end dest

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

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

Examples:

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

```
bm      40000 80000 200000
```

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

```
bm      data_start data_end 200000
```

2.4.5. *BR - Breakpoint* **BR**

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

The BR command inserts or removes breakpoints at address addr. The value for addr 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
```

2.4.6. BS - Block Search *BS*

Usage: BS<width> begin end data

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

The value for addresses begin and end 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
```

2.4.7. *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
```

2.4.8. *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
```

2.4.9. *DL - Download Serial* **DL**

Usage: DL <offset>

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

If offset is provided, then the destination address of each S-record is adjusted by offset. The DL command checks the destination address for validity. If the destination is an address below the defined user space (0x00000000-0x00020000), then an error message is displayed and downloading aborted.
If the S-record file contains the entry point address, then the program counter is set to reflect this address.

Examples:

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

```
dl
```

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

```
dl      0x40
```

2.4.10. *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.

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

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

The DN command checks the destination address for validity. If the destination is an address below the defined user space, 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.

2.4.11. *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
```

2.4.12. *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

2.4.13. *HELP - Help* HE

Usage: HELP <command>

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of where user code 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
```

2.4.14. *IRD - Internal Registers Display* **IRD**

Usage: IRD <module.register>

This commands displays the internal registers of different modules inside the MCF5307. 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 MCF5307 are SIM, UART1, UART2, TIMER, M-Bus, DRAMC, and Chip-Select. Refer to MCF5307 User's Manual.

Example:

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

2.4.15. *IRM - Internal Registers MODIFY* **IRM**

Usage: IRM module.register data

This commands modifies the contents of the internal registers of different modules inside the MCF5307. 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 MCF5307 are SIM, UART1, UART2, TIMER, M-Bus, DRAMC, Chip-Select. Refer to MCF5307 User's Manual.

Example:

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

2.4.16. MD - Memory Display **MD**

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

The MD command displays a contiguous block of memory starting at address begin and stopping at address end. The value for addresses begin and end 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.

2.4.17. *MM - Memory Modify* **MM**

Usage: MM<width> addr <data>

The MM command modifies memory at the address addr. The value for address addr 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
```

2.4.18. *RD - Register Display* **RD**

Usage: RD <reg>

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

Examples:

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

```
rd
```

To display only the program counter, the command is:

```
rd      pc
```

2.4.19. *RM - Register Modify* **RM**

Usage: RM reg data

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

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

Examples:

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

```
rm      D0 1234
```

2.4.20. *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
```

2.4.21. *SET - Set Configuration* SET

Usage: SET option <value>
SET

The SET command allows the setting of user configurable options within dBUG. The options are listed below. If the SET command is issued without 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. This command is used *to inform the dBUG* of the address of the network interface. The default value shows 0x0000. However, this parameter is hard coded to 0x300. ***DO NOT CHANGE THIS OPTION.***

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

Examples:

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

```
set
```

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

```
set     baud 19200
```

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

2.4.22. *SHOW - Show Configuration* **SHOW**

Usage: SHOW option
SHOW

The SHOW command displays the settings of the user configurable options within dBUG. 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
```

2.4.23. *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
```

2.4.24. *SYMBOL - Symbol Name Management* **SYMBOL**

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

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

2.4.25. *TRACE - Trace Into* **TR**

Usage: TRACE <num>

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

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

Usage: UPDBUG

The UPDBUG command is used for updating the dBUG image in Flash. When updates to the MCF5307 EVS dBUG are available, the updated image is downloaded to address 0x00020000. The new image is placed into Flash using the UPDBUG command. The user is prompted for verification before performing the operation. Use this command with extreme caution, as any error can render dBUG, and thus the board, useless!

2.4.27. *UPUSER - Update User Code In Flash* **UPUSER**

Usage: UPUSER <number of sectors>

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

Examples:

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

upuser or upuser 6

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

upuser 1

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 SBC5307 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 SBC5307 board.

3.1.1. The Processor

The microprocessor used in the SBC5307 is the highly integrated MCF5307, 32 -bit processor. The MCF5307 uses a ColdFire® processor as the core with 8K bytes of unified cache, two UART channels, two Timers, 4K 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. All the processor's signals are available through mictor connectors, LA1, LA2, LA3, LA4 and LA5. Refer to section 3.7 for pin assignment.

The MCF5307 has an IEEE JTAG-compatible port and BDM port. These signals are available at port J1. The processor also has the logic to generate up to eight (8) chip selects, -CS0 to -CS7, and support ADRAM or SDRAM.

3.1.2. The Reset Logic

The reset logic provides system initialization. The reset occurs during power-on and asserts the *RSTI which causes total system reset. The reset is also triggered by the red reset switch and resets the entire processor.

U5 is used to produce active low power-on RESET signal which feeds into the ispLSI2032 (U9 where the Push-button RESET also goes. The U9 device generates the system reset (-RESET) and Ethernet RESET signals.

ROM Monitor 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 the SDRAM.

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

3.1.3. The -HIZ Signal

The -HIZ signal is actively driven by the LSI2032 (U9). This Signal is available for monitor on connector LA3. However, this signal should not be driven by the user.

3.1.4. The Clock Circuitry

The SBC5307 uses a 45MHZ oscillator (U22) to provide the clock to CLK pin of the processor. In addition to U22, there also exist a 20MHz oscillator which feeds into the Ethernet chip. The bus clock out of the MCF5307 drives a clock buffer chip which is fed into the edge select pin of the MCF5307, the ispLSI2032 for Ethernet timing (1/4 bus clock), SRAM (U19), and SDRAM (U23).

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 MCF5307 has the necessary logic built into the chip to watch the duration of the bus cycle. If the cycle is not terminated within the preprogrammed duration the logic will internally assert a Transfer Error signal. In response, the processor will terminate the bus cycle and an access fault exception (trap) will take place.

The duration of the Watchdog is selected by BMT0-1 bits in System Protection Register. The dBUG initializes this register with the value 00, which provides for 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 MCF5307 User's Manual).

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

The MCF5307 has four external interrupt request lines. You can program the external interrupt request pins to level 1, 3, 5, and 7 or levels 2, 4, 6, and 7. The SBC5307 configures these lines as level 1, 3, 5, and 7. 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 0 to 3 will be assigned.

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 SBC5307 uses -IRQ7 to support the ABORT function using the ABORT switch S1 (red 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 -IRQ1 line of the MCF5307 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 MCF5307 User's Manual for more information about the interrupt controller.

3.1.7. Internal SRAM

The MCF5307 has 4K bytes of internal memory. This memory is mapped to 0x00800000 and is not used by the dBUG. It is available to the user.

3.1.8. The MCF5307 Registers and Memory Map

The memory and I/O resources of the SBC5307 are divided into three groups, MCF5307 Internal, External resources, and the ethernet controller. All the I/O registers are memory mapped.

The MCF5307 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 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 SBC5307 uses chip-select zero (/CS0) to enable the Flash ROM's (refer to Section 3.3.) The SBC5307 uses /RAS1, /RAS2, /CAS0, /CAS1, /CAS2, and /CAS3 to enable the SDRAM DIMM module (refer to Section 3.2), /CS2 for SRAM (not populated), and /CS3 for Ethernet Bus I/O space.

The chip select mechanism of the MCF5307 allows the memory mapping to be defined based on the memory space desired (User/Supervisor, Program/Data spaces).

All the MCF5307 internal registers, configuration registers, parallel I/O port registers, DUART registers and system control registers are mapped by MBAR register at any 1K-byte boundary. It is mapped to 0x10000000 by dBUG. For complete map of these registers refer to the MCF5307 User's Manual.

The SBC5307 board can have up to 8M bytes of SDRAM installed. The first 8M bytes are reserved for this memory. Refer to Section 3.2 for a discussion of RAM. The dBUG is programmed in two 29LV004B Flash ROM's which only occupies 1M bytes of the address space. The first 128K bytes are used by ROM Monitor and the second half is left for user. Refer to section 3.3.

The Ethernet Bus interface maps all the I/O space of the Ethernet bus to the MCF5307 memory at address $FE600000. Refer to section 3.6.

Table 4 - The SBC5307 memory map

ADDRESS RANGE	SIGNAL and DEVICE
$00000000-$007FFFFF	/RAS1, /RAS2, 8M bytes of SDRAM's.
$00800000-$00800FFF	Internal SRAM (4K bytes)
$10000000-$100003FF	Internal Module registers
$FE400000-$FE47FFFF[1]	External SRAM (512K bytes)
$FE600000-$FE7FFFFF	/CS3, 2M Ethernet Bus area
$FFE00000-$FFEFFFFF	/CS0, 1M bytes of Flash ROM.

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

All the unused area of the memory map is available to 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 MCF5307, the /CS0 responds to any accesses after reset until the CSMR0 is written. Since /CS0 is connected to Flash ROM's, the Flash ROMs 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 DRAMs to start at $00000000.

3.1.10. /TA Generation

The processor starts a bus cycle by providing the necessary information (address, R/*W, etc.) and asserting the /TS. The processor then waits for an acknowledgment (/TA) by the addressed device before it can complete the bus cycle. This /TA is used not only to indicate the presence of a device, it also allows devices with different access time to communicate with the processor properly. The MCF5307, 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 ROM Monitor to generate the /TA internally after a preprogrammed number of wait states. In order to support the future expansion of the board, the /TA input of the processor is also connected to the Processor Expansion Bus, 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 SDRAM DIMM 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 MCF5307 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 SDRAM's and Flash ROM's respectively.

3.2 THE SDRAM DIMM

The SBC5307 has one 168-pin DIMM socket (U23) for SDRAM DIMM. This socket supports SDRAM DIMM's of 1M x 4 x 16-Bits SDRAM x 2. No special configuration is needed. The DIMM speed should be a minimum of 70ns. The SDRAM Access timing is 2,4,2,1,-1 for t_{RCD}, t_{RAS}, t_{RP}, t_{RWL}, and t_{EP} respectfully. These timings determine how long the data is delayed after the /CAS signal (or the read command) is asserted during a SDRAM access. This corresponds to the t_{RCD} specifications in most SDRAM's. The other timings that correspond to the SDRAM are the active command to precharge command (t_{RAS}), precharge command to active command (t_{RP}), last data input to precharge command (t_{RWL}), and last data out to early precharge (t_{EP}).

3.3 FLASH ROM

There are two 512Kbyte Flash ROM's on the SBC5307, U20 (high, even byte) and U21 (low, odd byte).

The board is shipped with two 29LV004, 512K-byte, FLASH ROM's for a total of 1M bytes. The first 128K of the Flash contains ROM Monitor firmware. The last 896K is available to the user. The chip-select signal generated by the MCF5307 (/CS0) enables both chips.

The MCF5307 chip-select logic can be programmed to generate the /TA for /CS0 signal after a certain number of wait states. The dBUG programs this parameter to three wait-states.

3.3.1. JP1 Jumper and User's Program

This jumper allows users to test code from the boot without having to overwrite the ROM Monitor.
When the jumper is set between pins 1 and 2, the behavior is normal. When the jumper is set between pins 2 and 3, the board boots from the second half of Flash (0x80000).

Procedure:

1. Compile and link as though the code was to be place at the base of the flash, but setup so that it will download to the SDRAM starting at address 0x80000. The user need to refer to the compiler for this, since it will depend upon the compiler used (in Diab Data, a shadow in the linker file is used).

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

3. Download to SDRAM (If using serial or ethernet, start ROM Monitor first. If using BDM via wiggler, download first, then start ROM Monitor by pointing PC to 0xffe00400 and run.)

4. In ROM Monitor, run 'upuser' command.

5. Move jumper to 3.3V and reset, pin 2 connected to pin 3. User code should be running.

3.4 THE SERIAL COMMUNICATION CHANNELS

The SBC5307 offers a number of serial communications. They are discussed in this section.

3.4.1. The MCF5307 DUART

The MCF5307 has two built in UART's, each with its own software programmable baud rate generators, only one channel is the ROM Monitor to Terminal output and other is available to the user. The ROM Monitor, however, programs the interrupt level for UART1 to Level 3, priority 2 and autovector mode of operation. The interrupt level for UART2 to Level 3, priority 1 and autovector mode of operation. The signals of these channels are available on port LA1 and LA3. The signals of UART1 and UART2 are also passed through the RS-232 driver/receiver and are available on DB-9 connectors J4 and J7. Refer to the MCF5307 User's Manual for programming and the register map.

3.4.2. Motorola Bus (M-Bus) Module

The MCF5307 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 SBC5307 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 LA4 connector. These signals are open-collector signals. However, they have pull-up resistors on the SBC5307. These signals are connected to the SDRAM DIMM module I^2C interface but not used by the debugger. The interrupt control register for M-Bus is set for Level 3, priority 0 and autovector.

3.5 THE PARALLEL I/O Port

The MCF5307 has one 16-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 by the ROM Monitor

3.6 ON BOARD ETHERNET LOGIC

The SBC5307 includes the necessary logic, drivers, and the NE2000 compatible Ethernet chip to allow 10M bit transfer rate on a network. The Ethernet-space addresses are located starting at 0xFE600000.

The interface base address is 0x300 and uses IRQ3. However, the Ethernet base address in our system as mentioned earlier is 0xFE600000. Which brings the address of chip to 0xFE600300. Note that all registers should be addressed as WORD (eventhough the registers are bytes) also note that the even address registers are addressed as they are (no change), the read word will have the byte of the data in the lower byte of the word.

For odd addressed bytes, the address is mapped to 0xFE6083xx-1. Note that odd-bytes are addressed as even addresses but increased by 0x8000. Still the read byte will be in the lower byte of the read word Below is an example of the data structure used to define the registers. For the description of the registers refer to the Data Sheet for Davicom DM9008, a copy of this document in on Coldfire Website.

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

                        NATURAL16   DATAPORT;

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

                        /* Odd registers */
                        NATURAL16   CLDA0;      /* CLDA0 (rd) PSTART (wr) */
                        NATURAL16   BNRY;       /* Boundary pointer (rd wr) */
                        NATURAL16   NCR;        /* NCR (rd) TBCR0 (wr) */
                        NATURAL16   ISR;        /* Interrupt Status Register (rd wr) */
                        NATURAL16   CRDA1;      /* CRDA1 (rd) RSAR1 (wr) */
                        NATURAL16   RBCR1;      /* Remote Byte Count 1 (wr) */
                        NATURAL16   CNTR0;      /* CNTR0 (rd) TCR (wr) */
                        NATURAL16   CNTR2;      /* CNTR2 (rd) IMR (wr) */
                } page0;
                struct
                {
                        /* Even registers */
                        NATURAL16   PAR1;       /* Physical Address Byte 1 */
                        NATURAL16   PAR3;       /* Physical Address Byte 3 */
```

```
                NATURAL16   PAR5;           /* Physical Address Byte 5 */
                NATURAL16   MAR0;           /* Multicast Address Byte 0 */
                NATURAL16   MAR2;           /* Multicast Address Byte 2 */
                NATURAL16   MAR4;           /* Multicast Address Byte 4 */
                NATURAL16   MAR6;           /* Multicast Address Byte 6 */

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

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

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

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

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

The Ethernet Bus interrupt request line is hardwired to IRQ3.

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

3.7 THE CONNECTORS AND THE EXPANSION BUS

There are 8 connectors on the SBC5307 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 Terminal Connector J4

The signals on UART1 that runs through RS-232 driver/receivers are used to drive the Terminal. The SBC5307 uses a 9-pin D-sub female connector J4 for connecting the board to a terminal or a PC with terminal emulation software. The available signals are a working subset of the RS-232C standard. Table 5 - The J4 (Terminal) Connector pin assignment shows the pin assignment.

Table 5 - The J4 (Terminal) Connector pin assignment

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

3.7.2. The Auxiliary Serial Communication Connector J7

The MCF5307 has two built-in UART's. One channel is not used by the SBC5307 ROM Monitor and is available to the user. This signal is available on port J7. The available signals form a working subset of the RS-232C standard. Table 6 - The J7 Connector pin assignment shows the pin assignment for J7.

Table 6 - The J7 Connector pin assignment

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

3.7.3. Logical Analyzer connectors LA1-5 and Processor Expansion Bus J8 & J9

All the processors signals are available on 5 mictor connectors LA1-5. 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. A subset of the signals are available on J8 and J9 for easier access. Tables 7-14 show the pin assignment for J8, J9, LA1, LA2, LA3, LA4 and LA5 respectively.

Table 7 - The J8 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	TIN1	2	TT0_PP0
3	TOUT1	4	TT1_PP1
5	TIN0	6	TM0_PP2
7	TOUT0	8	TM1_PP3
9	SCL	10	TM2_PP4
11	SDA	12	DREQ1_PP5
13	/IRQ1	14	DREQ0_PP6
15	/IRQ5	16	XTIP_PP7
17	/CS0_HEADER	18	A24_PP8
19	/BWE0	20	A25_PP9
21	/BWE1	22	A26_PP10
23	/BWE2	24	A27_PP11
25	/BWE3	26	A28_PP12
27	/OE	28	A29_PP13
29	/CS4	30	A30_PP14
31	/CS5	32	A31_PP15
33	/RTS0	34	GND

Table 8 - The J9 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	D1	2	D2
3	D0	4	D3
5	A0	6	D4
7	A1	8	D5
9	A2	10	D6
11	A3	12	D7
13	A4	14	D8
15	A5	16	D9
17	A6	18	D10
19	A7	20	D11
21	A8	22	D12
23	A9	24	D13
25	A10	26	D14
27	A11	28	D15
29	A12	30	D16
31	A13	32	D17
33	A14	34	D18
35	A15	36	D19
37	A16	38	D20
39	A17	40	D21
41	A18	42	D22
43	A19	44	D23
45	A20	46	D24
47	A21	48	D25
49	A22	50	D26
51	A23	52	D27
53	R/-W	54	D28
55	-AS	56	D29
57	-TA	58	D30
59	GND	60	D31

Table 9 - The LA2 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	NC	4	A0
5	A1	6	A3
7	A5	8	A7
9	A9	10	A11
11	A13	12	A15
13	A17	14	A19
15	A21	16	A23
17	A25_PP9	18	A27_PP11
19	A29_PP13	20	A31_PP15
21	A30_PP14	22	A28_PP12
23	A26_PP10	24	A24_PP8
25	A22	26	A20
27	A18	28	A16
29	A14	30	A12
31	A10	32	A8
33	A6	34	A4
35	A2	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 10 - The LA1 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	PSTCLK	4	TXD1
5	TXD2	6	NC
7	NC	8	SIZ0
9	/CS0	10	/BG
11	NC	12	/IRQ3
13	/CF_RSTI	14	/TA
15	/CS4	16	/CS3
17	/CS2	18	/AS
19	/IRQ5	20	/BR
21	/TS	22	/CS5
23	/CS6	24	/CS7
25	R/-W	26	/IRQ7
27	/IRQ1	28	/BD
29	/CS1	30	/OE
31	SIZ1	32	NC
33	NC	34	NC
35	NC	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 11 - The LA3 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	NC	4	PST3
5	PST0	6	PST1
7	PST2	8	DREQ0_PP6
9	DREQ1_PP5	10	MTMOD0
11	MTMOD1	12	RXD2
13	/RTS2	14	RXD1
15	MTMOD2	16	DSCLK_/TRST
17	DSDO_TDI	18	TCK
19	/BKPT_TMS	20	/HIZ
21	DSDO_TDO	22	/CTS2
23	/CTS1	24	/RTSI
25	MTMOD3	26	DDATA0
27	DDATA1	28	DDATA2
29	DDATA3	30	XTIP_PP7
31	TMO_PP2	32	TM1_PP3
33	TT0_PP3	34	TM2_PP4
35	TT1_PP1	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 12 - The LA4 Connector pin assignment

PIN NO.	SIGNAL NAME	PIN NO.	SIGNAL NAME
1	NC	2	NC
3	BCLK0	4	/R_RAS0
5	/R_CAS2	6	/R_RAS1
7	/R_CAS0	8	/R_CAS1
9	/R_CAS3	10	/R_SRAS
11	/R_DRAMW	12	/R_SCAS
13	R_SCKE	14	TIN0
15	TOUT0	16	PPLTPA
17	EDGSEL	18	SCL
19	SDA	20	NC
21	NC	22	NC
23	NC	24	NC
25	NC	26	NC
27	NC	28	NC
29	NC	30	/BWE0
31	/BWE3	32	/BEW2
33	/BWE1	34	/TIN1
35	TOUT1	36	NC
37	NC	38	NC
39	GND	40	GND
41	GND	42	GND
43	GND		

Table 13 - The LA5 Connector pin assignment

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

3.7.4. The Debug Connector J1

The MCF5307 does have background Debug Port, Real-Time Trace Support, and Real-Time Debug Support. The necessary signals are available at connector J1. Table 14 - The J1 Connector pin assignment shows the pin assignment.

Table 14 - The J1 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	No Connect
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	+3.3 Volts
26	No Connect

APPENDIX A (Configuring dBUG for Network Downloads)

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

A.1 Required Network Parameters

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

All computers connected to an Ethernet network running the IP protocol need 3 network-specific parameters. These parameters are:

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

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

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

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

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

A.2 Configuring dBUG Network Parameters

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

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

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

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

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

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

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

set filename <filename>

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

set filetype <srecord|coff|elf|image>

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

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

Change the default filename and filetype with the commands:

set filename a.out
set filetype coff

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

A.3 Troubleshooting Network Problems

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

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

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

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

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

APPENDIX B (FPLA code)

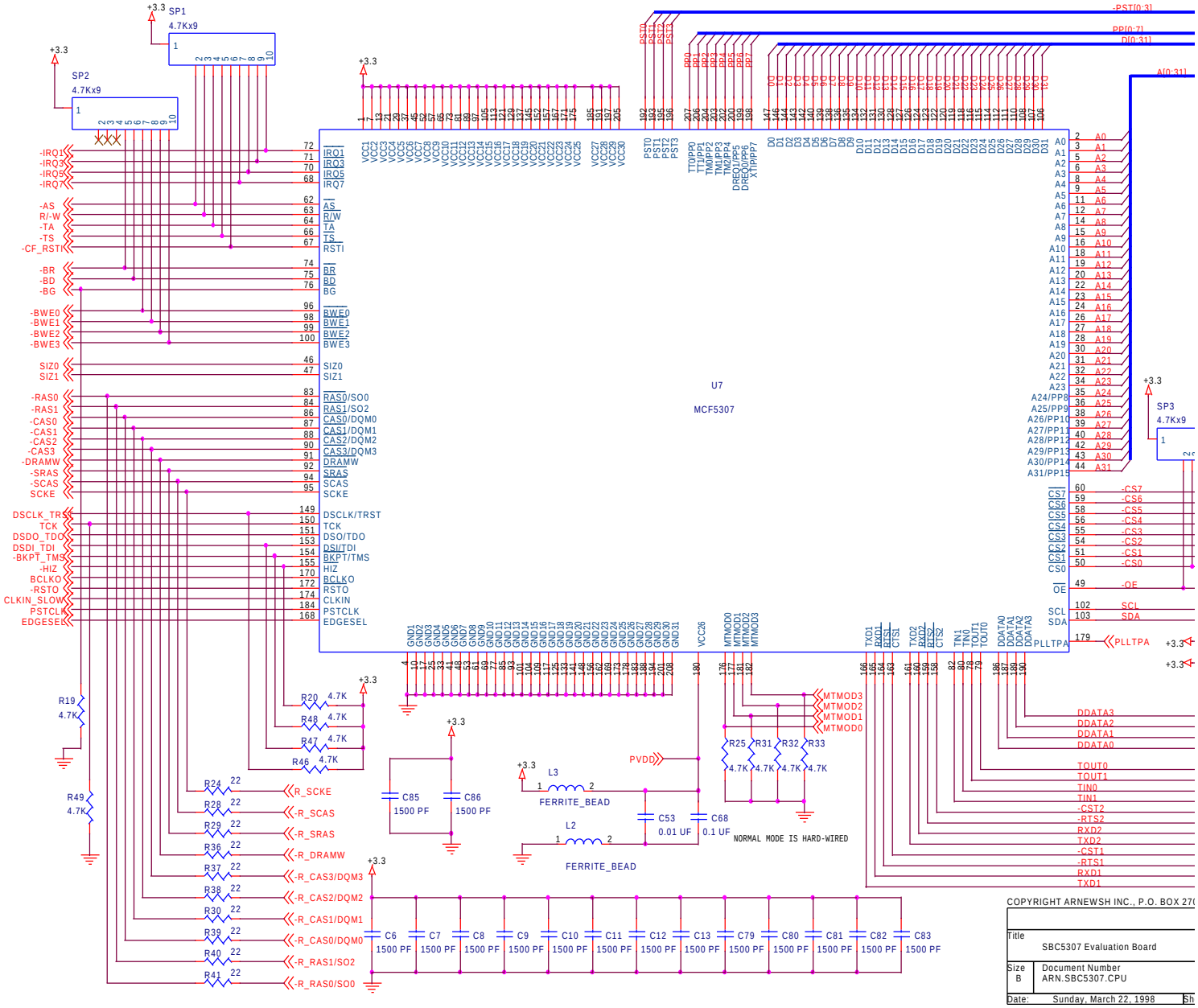
```
module isa2
title 'Ethernet  controller'
"Feb 26 '98 version v1 of the 5307
"isa2 device 'ispLSI';
;"*************************************************"
;"This abel file contains the code for a NE2000 compatible Ethernet"
;"for the 55307 Coldfire processor as well as reset and IRQ7 (abort)"
;"It was targeted to Lattice ispLSI 2032 fpga  "
;"CS: B3D3                                     "
;"*************************************************"
;"*************************************************"
;"Declaration Section                            "
;"*************************************************"
;" constants"
   C,P,X,Z,H,L = .C.,.P.,.X.,.Z.,1,0;
;"*************************************************"
DLYIOCHRDY0 node ISTYPE 'reg_d,buffer';
DLYIOCHRDY,ENDIT,END16,END8 node;
STARTISA node ISTYPE 'reg_d,buffer';
SBHE,IOR,IOW,ISAOE node;
DA,DLYDA node ISTYPE 'reg_d,buffer';
ABORTML,DAOE,CLK16MHZ node ISTYPE 'reg_d,buffer';

CLK4MHZ     node ISTYPE 'reg_d,buffer';

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


ABORTOL     pin 3 ISTYPE 'reg_d, buffer';
RST_L       pin 4;                  "Output - to ColdFire reset
DB_CS_L           pin 5;                  "Output - Data buffer enable for
ethernet
A0IN        pin 6;                  "INPUT  - A0 received from CF through
buffers
IOCHRDY           pin 7;                  "Input  - asserted by ethernet
IOCS16L           pin 9;                  "Input  - asserted by ethernet
SIZ1        pin 10;
XCLK0             pin 11;                 "Input  - global clock
IOWL        pin 15;                 "Input  - write signal from ethernet
RD          pin 16;                 "INPUT  - R/W* from the ColdFire
CLK8MHZ     pin 17 ISTYPE 'reg_d,buffer';
BALE        pin 18;                 "Output - address latch enable
A0          pin 19;                 "OUTPUT - A0 sent to the ethernet
PORIN_L     pin 26;                 "Input  - Suppy Voltage Supervisor
CS3_L             pin 22;                 "Input  - From ColdFire
RSTIN_L           pin 27;                 "Input  - Hard Reset switch
ETHER_IRQ   pin 28;                 "Input  - Ethernet IRQ 3
IRQ3        pin 29;                 "Output - IRQ 3 into the ColdFire
RST_H       pin 30;                 "Output - to the Ethernet
ABORTIL     pin 31;                 "INPUT  - abort signal received from the
Abort swith
HIZ_L       pin 32;                 "Output - to ColdFire *HIZ
IORL        pin 37;                 "Input  - read signal from ethernet
A16         pin 39;
TAL         pin 40;                 "Input / Output - Transfer acknowledge
SBHEL             pin 41;                 "Output - sent to the ethernet
```

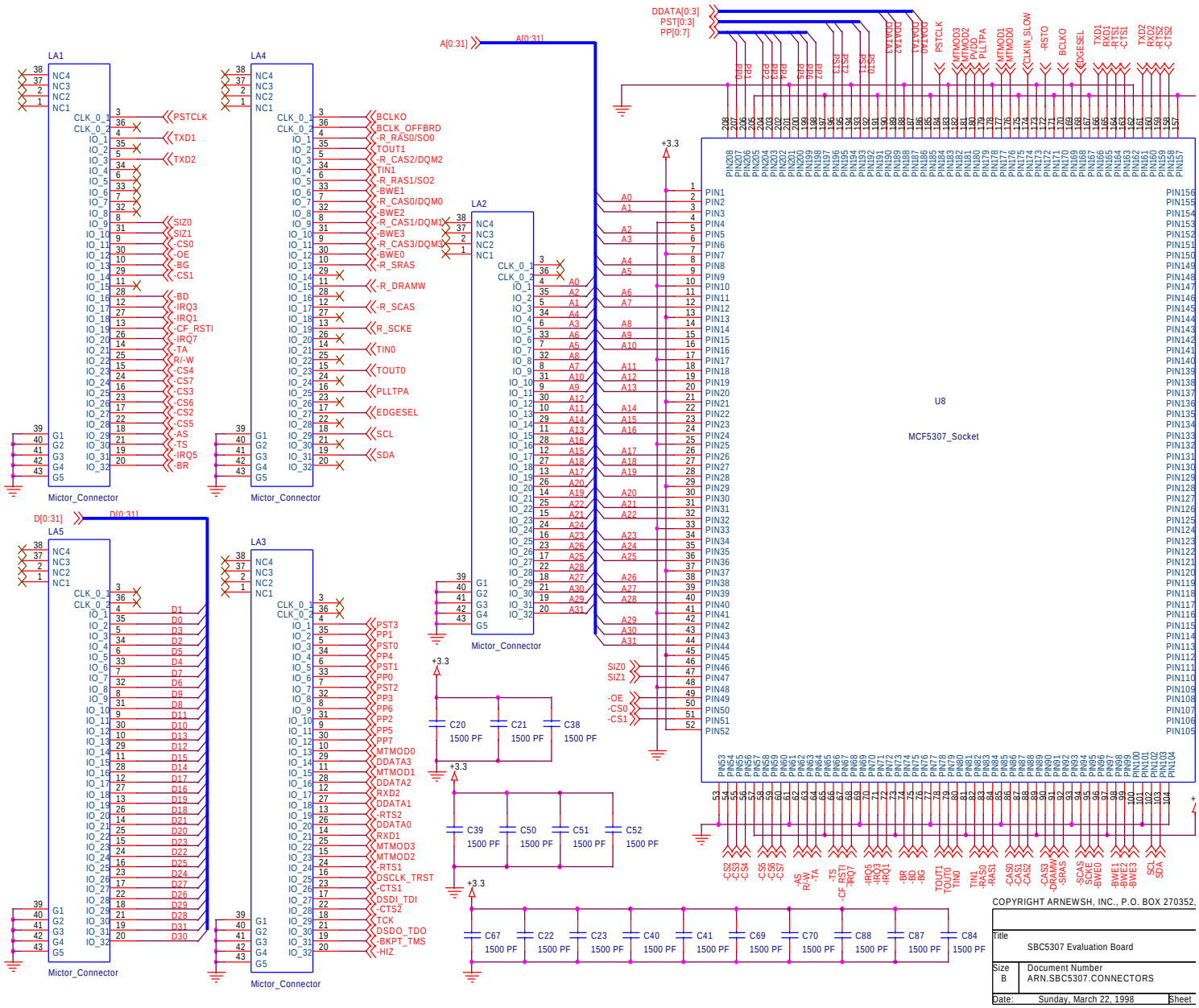
```
SIZ0        pin 43;
BDM_RST_L   pin 44;           "Input  - BDM reset input



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

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

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


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

equations

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

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

SBHEL = 1;

IRQ3 = !ETHER_IRQ;

!DB_CS_L = !RST_H & !CS3_L;

ABORTML := ABORTIL ;

ABORTML.clk  = CLK8MHZ ;

ABORTOL := ABORTML ;
```

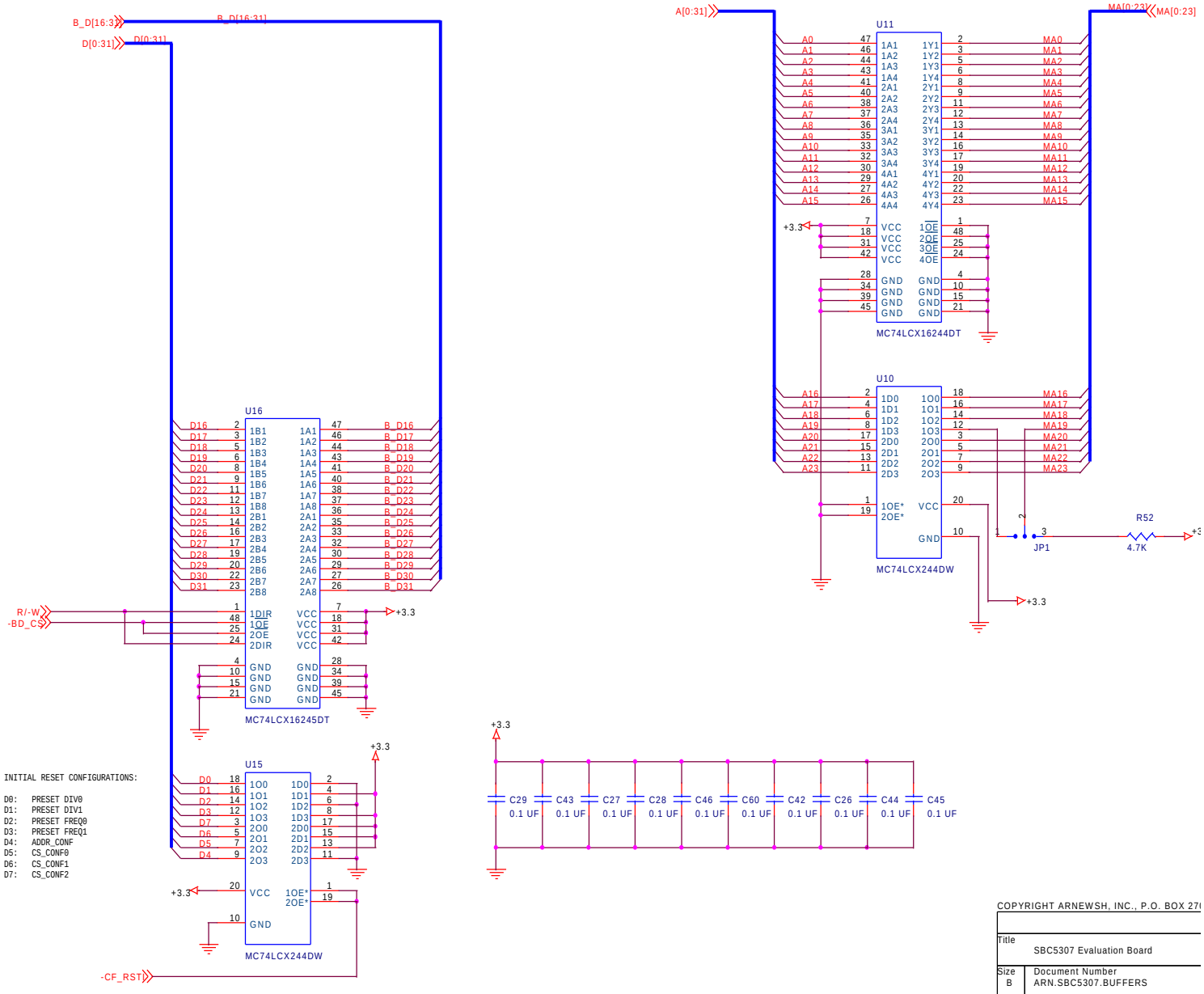
```
ABORTOL.clk  = CLK8MHZ ;

RSTMH = !RSTIN_L ;

RST_H = RSTMH # !PORIN_L # !BDM_RST_L;

!HIZ_L = !RST_L;

DAOE :=     !CS3_L # DA;

DAOE.clk = XCLK0 ;

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

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

CLK16MHZ := !CLK16MHZ ;

CLK16MHZ.clk = XCLK0 ;

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

CLK8MHZ.clk = XCLK0 ;

CLK4MHZ := CLK4MHZ $ ( CLK16MHZ & CLK8MHZ );

CLK4MHZ.clk = XCLK0 ;

DA := !CS3_L & END16 & ENDIT & !IOCS16L & RD & !CLK8MHZ & SBHE #
      !CS3_L & END8 & ENDIT & RD & !CLK8MHZ #
        DLYDA  & !CS3_L #
        DA & !CS3_L;

DA.clk=XCLK0;

DLYDA :=!CS3_L & END16 & ENDIT & !IOCS16L & !RD & !CLK8MHZ & SBHE #
      !CS3_L & END8 & ENDIT & IOCS16L & !RD & !CLK8MHZ #
      !CS3_L & END8 & ENDIT & !SBHE & !RD & !CLK8MHZ ;
DLYDA.clk=XCLK0;

STARTISA := !CS3_L & !ENDIT ;

STARTISA.clk  = CLK8MHZ ;

CBU43 (BCLK0,BCLK1,BCLK2,CLK8MHZ,STARTISA,!STARTISA)

BALE  = STARTISA & !CLK8MHZ &  !BCLK2 & !BCLK1 & !BCLK0 & ! IOR & !IOW ;

IOR   = STARTISA & !BCLK2 & !BCLK1 & BCLK0 & !CLK8MHZ & RD #
        IOR & !CS3_L ;

IOW   = STARTISA & !BCLK2 & !BCLK1 & BCLK0 & !CLK8MHZ & !RD #
        IOW & STARTISA ;

END16 = !BCLK2 & BCLK1 & !BCLK0 & !CLK8MHZ#
      END16 & STARTISA ;

END8 =      BCLK2 & !BCLK1 & BCLK0 & !CLK8MHZ #
      END8 & STARTISA ;
```

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

DLYIOCHRDY0:= IOCHRDY;

DLYIOCHRDY0.clk  = CLK8MHZ ;

DLYIOCHRDY = IOCHRDY & CLK8MHZ #
           DLYIOCHRDY & !CLK8MHZ ;

;"**************************************************"
;" Test Vector Section"
;"**************************************************"
test_vectors 'HIZ_L  Test Vector'
([XCLK0, RSTIN_L ,PORIN_L,BDM_RST_L,CS3_L]->[RST_H])
 [P,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,0,1,1]->[X];
 [C,1,0,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,0,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,0]->[X];
 [C,0,1,1,0]->[X];
 [C,0,1,1,0]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,0,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,0]->[X];
 [C,1,1,1,0]->[X];
 [C,1,1,0,0]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
 [C,1,1,1,1]->[X];
end
```

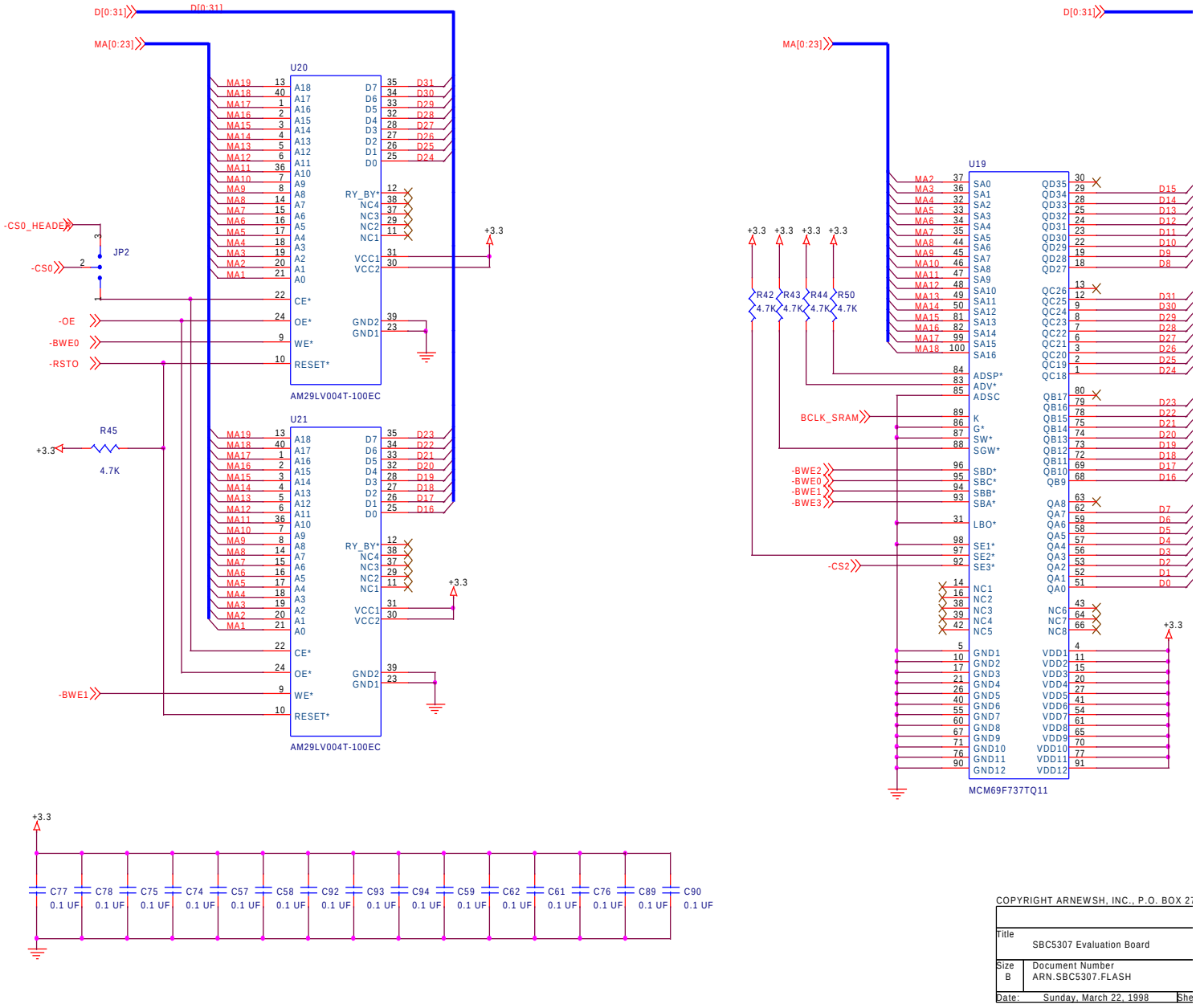
APPENDIX C (Schematics)


SP1
4.7Kx9
SP2
4.7Kx9
SP3
4.7Kx9
U7
MCF5307
-PST[0:3]
PP[0:7]
D[0:31]
A[0:31]
-IRQ1
-IRQ3
-IRQ5
-IRQ7
-AS
R/-W
-TA
-TS
-CF_RSTI
-BR
-BD
-BG
-BWE0
-BWE1
-BWE2
-BWE3
SIZ0
SIZ1
-RAS0
-RAS1
-CAS0
-CAS1
-CAS2
-CAS3
-DRAMW
-SRAS
-SCAS
SCKE
DSCLK_TRST
TCK
DSDO_TDO
DSDI_TDI
-BKPT_TMS
-HIZ
BCLKO
-RSTO
CLKIN_SLOW
PSTCLK
EDGESEL
-CS7
-CS6
-CS5
-CS4
-CS3
-CS2
-CS1
-CS0
-OE
SCL
SDA
PLLTPA
MTMOD3
MTMOD2
MTMOD1
MTMOD0
DDATA3
DDATA2
DDATA1
DDATA0
TOUT0
TOUT1
TIN0
TIN1
-CST2
-RTS2
RXD2
TXD2
-CST1
-RTS1
RXD1
TXD1
R19 4.7K
R20 4.7K
R48 4.7K
R47 4.7K
R46 4.7K
R49 4.7K
R24 22
R28 22
R29 22
R36 22
R37 22
R38 22
R30 22
R39 22
R40 22
R41 22
R_SCKE
-R_SCAS
-R_SRAS
-R_DRAMW
-R_CAS3/DQM3
-R_CAS2/DQM2
-R_CAS1/DQM1
-R_CAS0/DQM0
-R_RAS1/SO2
-R_RAS0/SO0
C85 1500 PF
C86 1500 PF
L3
FERRITE_BEAD
L2
FERRITE_BEAD
PVDD
C53 0.01 UF
C68 0.1 UF
NORMAL MODE IS HARD-WIRED
R25 4.7K
R31 4.7K
R32 4.7K
R33 4.7K
C6 1500 PF
C7 1500 PF
C8 1500 PF
C9 1500 PF
C10 1500 PF
C11 1500 PF
C12 1500 PF
C13 1500 PF
C79 1500 PF
C80 1500 PF
C81 1500 PF
C82 1500 PF
C83 1500 PF
COPYRIGHT ARNEWSH INC., P.O. BOX 27(
Title
SBC5307 Evaluation Board
Size
B
Document Number
ARN.SBC5307.CPU
Date:
Sunday, March 22, 1998


LA1
LA2
LA3
LA4
LA5
Mictor_Connector
U8
MCF5307_Socket
A[0:31]
D[0:31]
DDATA[0:3]
PST[0:3]
PP[0:7]
+3.3
C20 1500 PF
C21 1500 PF
C38 1500 PF
C39 1500 PF
C50 1500 PF
C51 1500 PF
C52 1500 PF
C67 1500 PF
C22 1500 PF
C23 1500 PF
C40 1500 PF
C41 1500 PF
C69 1500 PF
C70 1500 PF
C88 1500 PF
C87 1500 PF
C84 1500 PF
COPYRIGHT ARNEWSH, INC., P.O. BOX 270352,
Title
SBC5307 Evaluation Board
Size B
Document Number ARN.SBC5307.CONNECTORS
Date: Sunday, March 22, 1998
Sheet

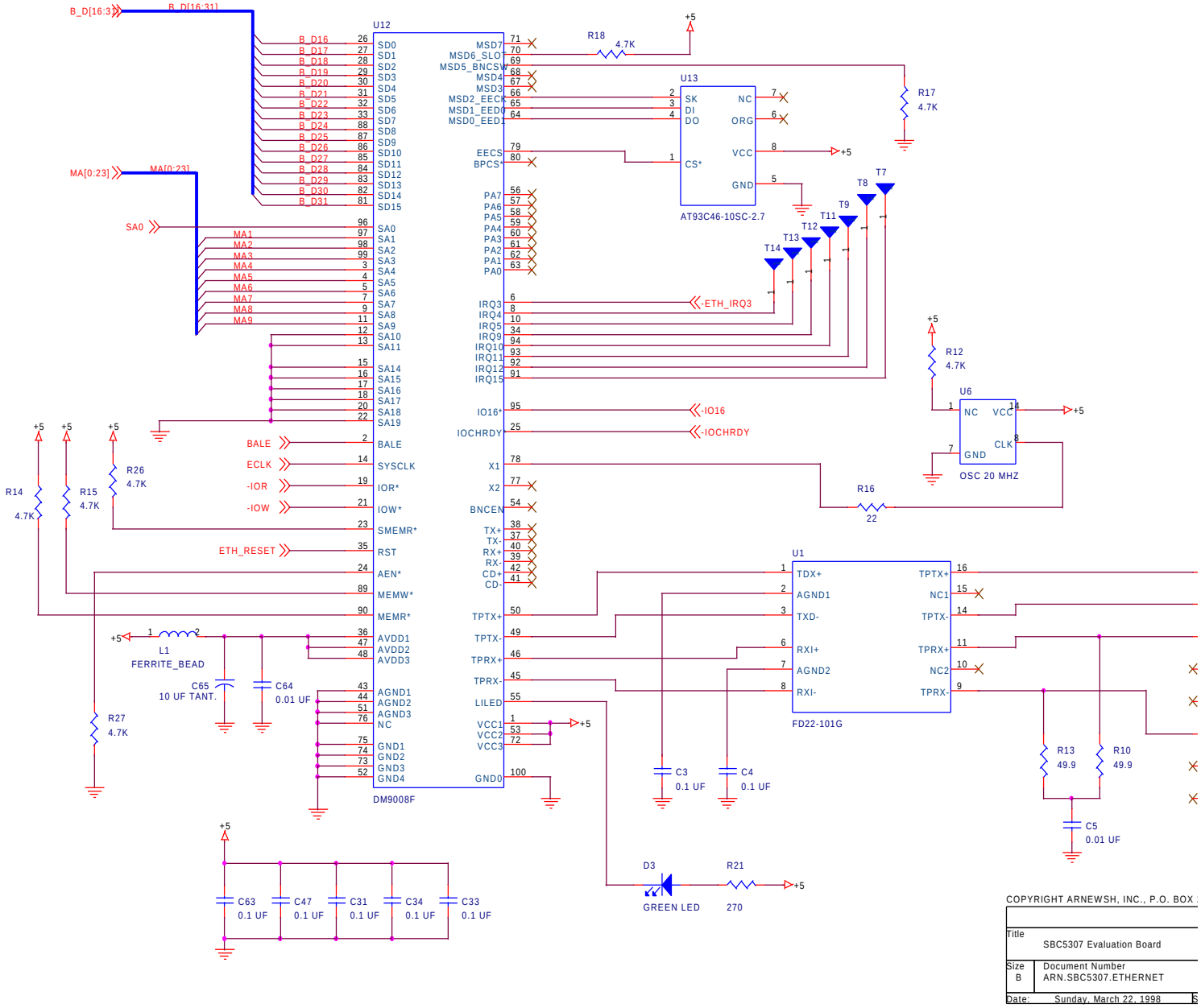


Title	SBC5307 Evaluation Board
Size B	Document Number ARN.SBC5307.BUFFERS
Date: Sunday, March 22, 1998	Shee



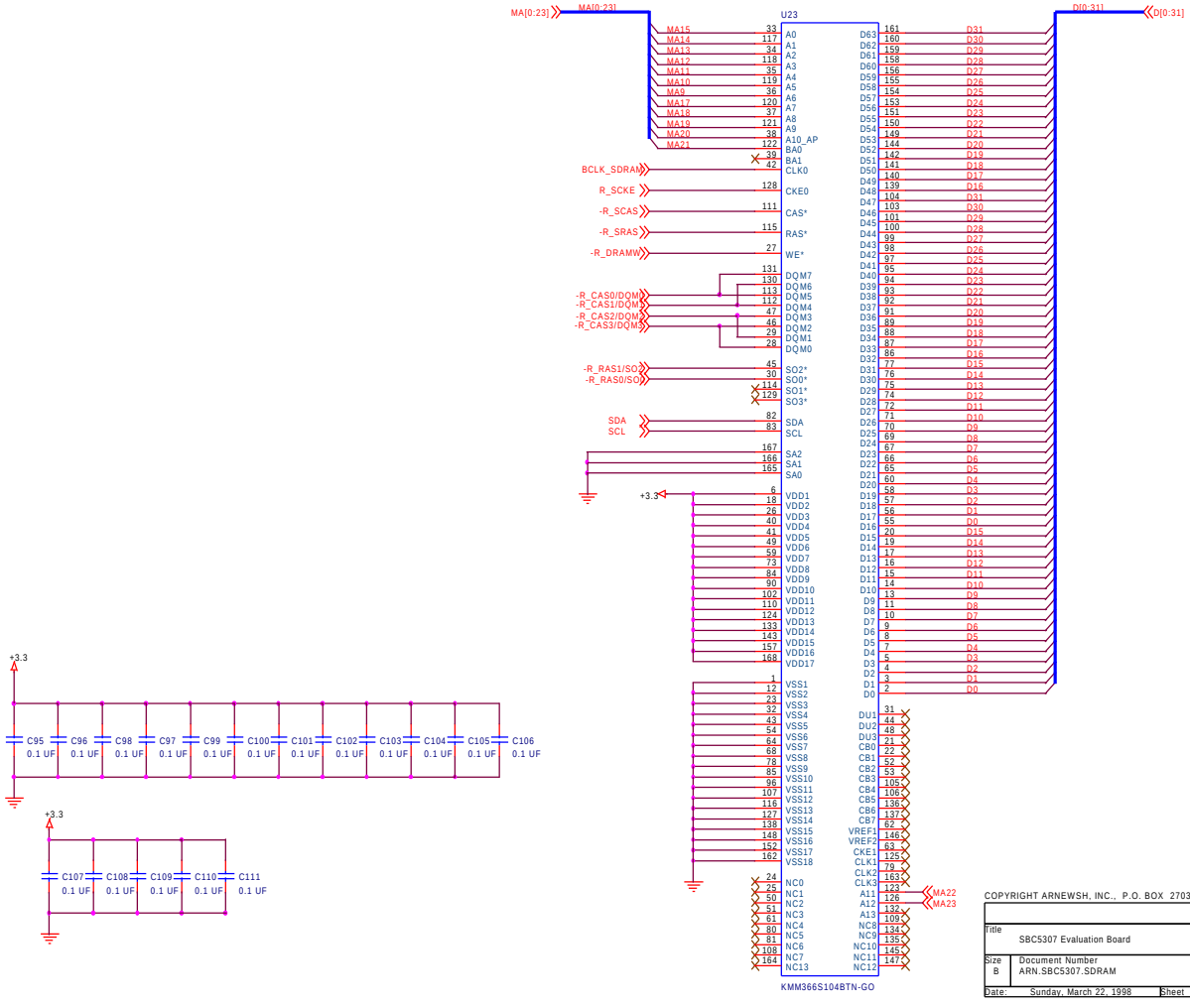
COPYRIGHT ARNEWSH, INC., P.O. BOX 2

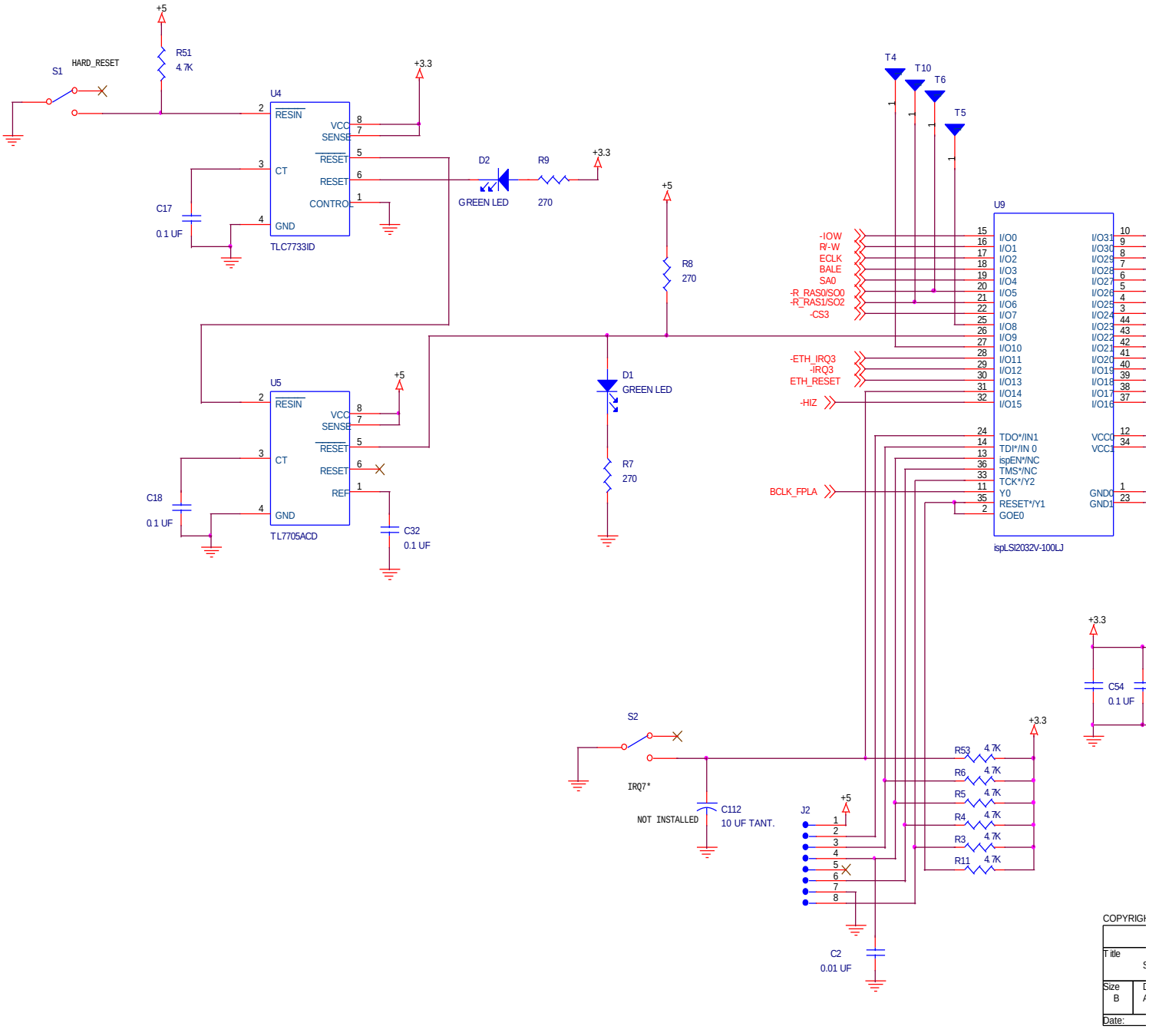
Title	SBC5307 Evaluation Board
Size B	Document Number ARN.SBC5307.FLASH
Date: Sunday, March 22, 1998	She

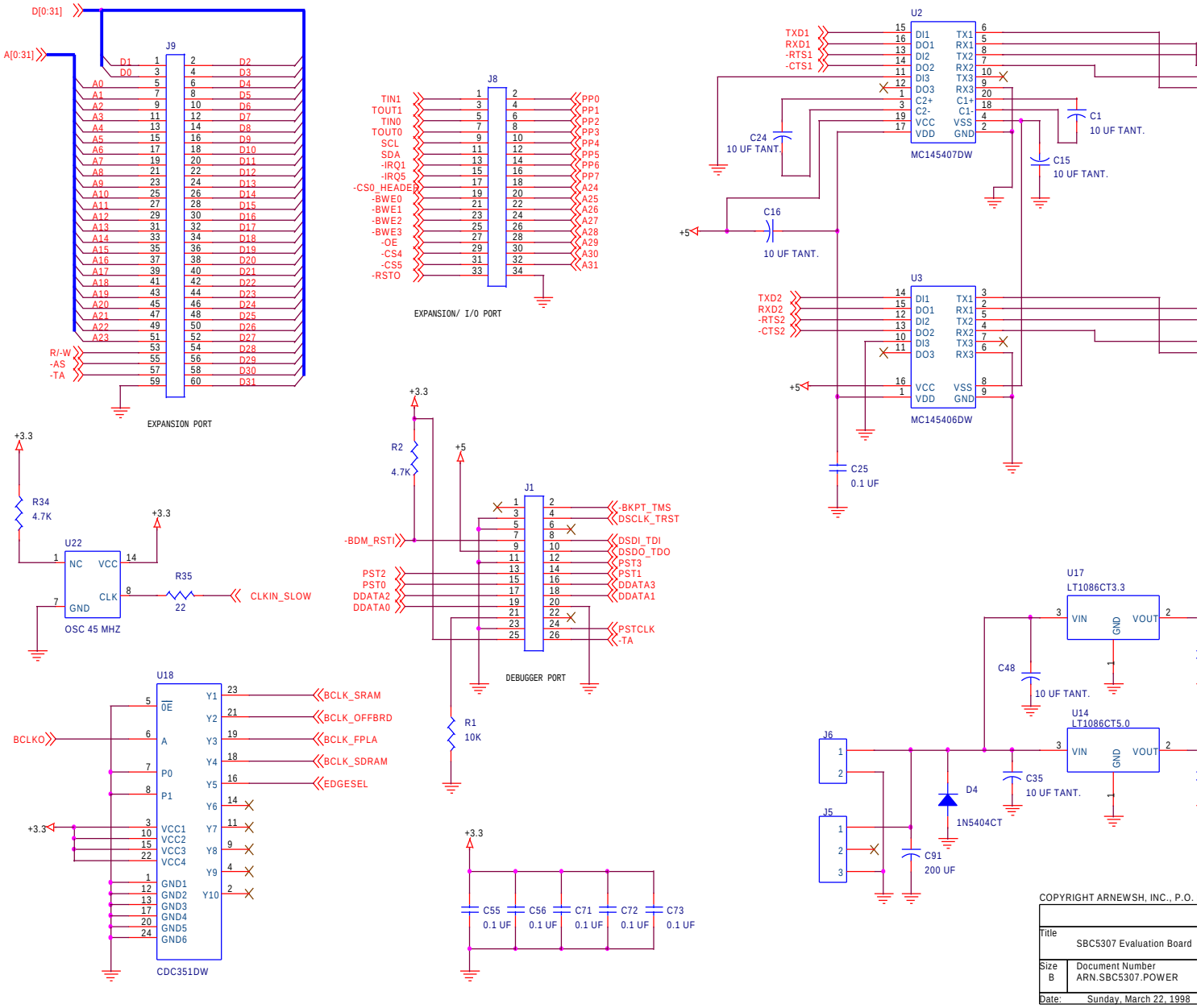


COPYRIGHT ARNEWSH, INC., P.O. BOX

Title	SBC5307 Evaluation Board
Size B	Document Number ARN.SBC5307.ETHERNET
Date: Sunday, March 22, 1998	S


MA[0:23]
MA[0:23]
U23
D[0:31]
D[0:31]
MA15
MA14
MA13
MA12
MA11
MA10
MA9
MA17
MA18
MA19
MA20
MA21
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10_AP
BA0
BA1
CLK0
CKE0
CAS*
RAS*
WE*
BCLK_SDRAM
R_SCKE
-R_SCAS
-R_SRAS
-R_DRAMW
DQM7
DQM6
DQM5
DQM4
DQM3
DQM2
DQM1
DQM0
-R_CAS0/DQM
-R_CAS1/DQM
-R_CAS2/DQM
-R_CAS3/DQM
SO2*
SO0*
SO1*
SO3*
-R_RAS1/SO
-R_RAS0/SO
SDA
SCL
SA2
SA1
SA0
+3.3
VDD1
VDD2
VDD3
VDD4
VDD5
VDD6
VDD7
VDD8
VDD9
VDD10
VDD11
VDD12
VDD13
VDD14
VDD15
VDD16
VDD17
VSS1
VSS2
VSS3
VSS4
VSS5
VSS6
VSS7
VSS8
VSS9
VSS10
VSS11
VSS12
VSS13
VSS14
VSS15
VSS16
VSS17
VSS18
NC0
NC1
NC2
NC3
NC4
NC5
NC6
NC7
NC13
DU1
DU2
DU3
CB0
CB1
CB2
CB3
CB4
CB5
CB6
CB7
VREF1
VREF2
CKE1
CLK1
CLK2
CLK3
A11
A12
A13
NC8
NC9
NC10
NC11
NC12
MA22
MA23
KMM366S104BTN-GO
D63
D62
D61
D60
D59
D58
D57
D56
D55
D54
D53
D52
D51
D50
D49
D48
D47
D46
D45
D44
D43
D42
D41
D40
D39
D38
D37
D36
D35
D34
D33
D32
D31
D30
D29
D28
D27
D26
D25
D24
D23
D22
D21
D20
D19
D18
D17
D16
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
+3.3
C95 0.1 UF
C96 0.1 UF
C98 0.1 UF
C97 0.1 UF
C99 0.1 UF
C100 0.1 UF
C101 0.1 UF
C102 0.1 UF
C103 0.1 UF
C104 0.1 UF
C105 0.1 UF
C106 0.1 UF
+3.3
C107 0.1 UF
C108 0.1 UF
C109 0.1 UF
C110 0.1 UF
C111 0.1 UF
COPYRIGHT ARNEWSH, INC., P.O. BOX 270352 FORT COLLINS, CO 80527
Title
SBC5307 Evaluation Board
Size B
Document Number ARN.SBC5307.SDRAM
Rev 1.0
Date: Sunday, March 22, 1998
Sheet 6 of 8


+5
R51
4.7K
S1
HARD_RESET
U4
RESIN
VCC
SENSE
RESET
CT
RESET
CONTROL
GND
TLC7733ID
+3.3
C17
0.1 UF
D2
GREEN LED
R9
270
+3.3
+5
R8
270
T4
T10
T6
T5
U9
-IOW
R/-W
ECLK
BALE
SA0
-R_RAS0/SO0
-R_RAS1/SO2
-CS3
-ETH_IRQ3
-IRQ3
ETH_RESET
-HIZ
BCLK_FPLA
I/O0
I/O1
I/O2
I/O3
I/O4
I/O5
I/O6
I/O7
I/O8
I/O9
I/O10
I/O11
I/O12
I/O13
I/O14
I/O15
TDO*/IN1
TDI*/IN 0
ispEN*/NC
TMS*/NC
TCK*/Y2
Y0
RESET*/Y1
GOE0
VCC0
VCC1
GND0
GND1
ispLSI2032V-100LJ
U5
RESIN
VCC
SENSE
RESET
CT
RESET
REF
GND
TL7705ACD
+5
C18
0.1 UF
C32
0.1 UF
D1
GREEN LED
R7
270
+3.3
C54
0.1 UF
S2
IRQ7*
NOT INSTALLED
C112
10 UF TANT.
+5
J2
+3.3
R53 4.7K
R6 4.7K
R5 4.7K
R4 4.7K
R3 4.7K
R11 4.7K
C2
0.01 UF
COPYRIGH
Title
Size
B
Date:



Title	SBC5307 Evaluation Board
Size B	Document Number ARN.SBC5307.POWER
Date:	Sunday, March 22, 1998

APPENDIX D (Pin Array Layout)

This layout represents the pattern for the MCF5307 socket if used on the SBC5307 board. The pin numbers corresponds to the MCF5307 processor.