KEIL SOFTWARE

MCB251SB Evaluation Board

MCS® 251 Microcontroller Target Board and Development Tools

User's Guide 01.96

Every effort was made to ensure accuracy in this manual and to give appropriate credit to persons, companies, and trademarks referenced herein.

A47 D11/19/98

Preface

This manual describes the Keil Software MCB251SB Evaluation Board and the MCS® 251 microcontroller software development tools. The following chapters are included:

"Chapter 1. Introduction" gives an overview of this user's guide and provides a quick start table.

"Chapter 2. Setup" describes how to connect and configure the board.

"Chapter 3. Theory of Operation" provides detailed information about the different sections of the circuit board.

"Chapter 4. Programming" gives details about how to use our tools to generate programs for the MCB251SB evaluation board.

"Appendix A. Schematics" contains the schematic drawings for the MCB251SB.

"Appendix B. PAL Equations" lists the logic equations for the PLDs used on the board.

"Appendix C. DIP Switch Settings" provides detailed information about the DIP switches on the board.

NOTE
This manual assumes that you are familiar with Microsoft Windows and the hardware and instruction set of the MCS® 251 microcontroller.

Document Conventions

This document uses the following conventions:

Examples	Description
README.TXT	Bold capital text is used for the names of executable programs, data files, source files, environment variables, and commands you enter at the MS-DOS command prompt. This text usually represents commands that you must type in literally. For example: **CLS** **DIR** **BL51.EXE** Note that you are not required to enter these commands using all capital letters.
Courier	Text in this typeface is used to represent information that displays on screen or prints at the printer. This typeface is also used within the text when discussing or describing command line items.
Variables	Text in italics represents information that you must provide. For example, *projectfile* in a syntax string means that you must supply the actual project file name. Occasionally, italics are also used to emphasize words in the text.
Elements that repeat…	Ellipses (…) are used to indicate an item that may be repeated.

Omitted code
.
.
.
	Vertical ellipses are used in source code listings to indicate that a fragment of the program is omitted. For example:
	`void main (void) {`
	`.`
	`.`
	`.`
	`while (1);`

⟦*Optional Items*⟧	Optional arguments in command-line and option fields are indicated by double brackets. For example: **C51 TEST.C PRINT ⟦(*filename*)⟧**
{ *opt1* | *opt2* }	Text contained within braces, separated by a vertical bar represents a group of items from which one must be chosen. The braces enclose all of the choices and the vertical bars separate the choices. One item in the list must be selected.
Keys	Text in this sans serif typeface represents actual keys on the keyboard. For example, “Press **Enter** to continue.”

Contents

Chapter 1. Introduction

Thank you for letting Keil Software provide you with the MCB251SB evaluation board and software for the MCS® 251 microcontroller family. With this kit you can generate code and then operate it on the MCB251SB evaluation board. This hands-on process helps you determine hardware and software needs for current and future product development.

The MCB251SB evaluation board lets you become familiar with the different operating modes of the Intel 8xC251SB microcontroller. The board supports all operating modes of the 251. Additionally, you may use an 8051 with this board. By generating and testing code for the various operating modes of the 251 and 8051, you can evaluate code and processor performance. These factors can be weighed against other production parameters to help choose the optimum code and processor combination. Alternatively, you may choose just to play with the board, make it flash the LEDs, and write "Hello World" out the serial port.

This user's guide describes the hardware of the MCB251SB evaluation board and contains the operating instructions for the 251 monitor (MON251) and the MON251 terminal program (MON251.EXE). MON251 is a configurable monitor which is installed in the 87C251SB-16 processor of the MCB251SB board. It lets you communicate between your PC and the MCB251SB evaluation board so that you can download and run your 251 programs.

The MCB251SB kit includes the following items:

- MCB251SB Evaluation Board User's Guide (this manual),
- MCB251SB Evaluation Board,
- MCB251SB Evaluation Board Software,
- 9 Volt DC Wall Transformer,
- 9-pin Serial Cable,
- and a 251 Evaluation Kit which includes a 2K compiler.

Quick Start

Use the following list to quickly locate important information about the MCB251SB evaluation board.

To…	See…
Connect power to the MCB251SB board.	“Connecting the MCB251SB” on page 4.
Connect the MCB251SB to your PC.	“Connecting the MCB251SB” on page 4.
Read about the default configuration settings.	“Configuring the MCB251SB” on page 5.
Create a simple program to blink the LEDs.	“FLASH Example Program” on page 30.
Write code to use the external UART.	“External UART Example” on page 41.
Learn more about the µVision IDE.	“Using µVision to Create the FLASH Program” on page 32.
Learn more about the dScope debugger.	“Using dScope to Debug the FLASH Program” on page 37.
Learn about the MON251 terminal program.	“Chapter 5. Using the 251 Monitor” on page 45.
Read about the DIP switch settings.	“Appendix C. DIP Switch Settings” on page 59.
Configure the RAM/ROM memory.	“Memory Map” on page 26.
See the MCB251SB schematics.	“Appendix A. Schematics” on page 49.
See the MCB251SB PAL equations.	“Appendix B. PAL Equations” on page 55.

Chapter 2. Setup

Setting up the MCB251SB evaluation board is easy. The board requires power and a serial connection to a PC running the MON251 terminal program. Before you start, make sure you have satisfied the following hardware and software requirements.

Hardware Requirements

- The MCB251SB Evaluation Board.
- A serial cable, 9-pin male to 9-pin female, 1-2 meters long, wired one-to-one.
- A power supply capable of providing 9-12VDC at 300-500mA. The power cable should terminate with a 5.5mm barrel plug with a 2.5mm center hole. Polarity is irrelevant.
- An IBM compatible PC with an available RS-232 port. If the port has a 25-pin connector, a 9-pin male to 25-pin female adapter may be required.
- A device programmer is required to program the 251 or 8051 and to program any EPROMs or other programmable devices. Contact our technical support group for a list of compatible device programmers.

Software Requirements

- The Keil MON251 terminal program (MON251.EXE).
- The Keil MON251 software/firmware (MON251.HEX). The monitor is already programmed into the provided 251 CPU (U1).
- Microsoft or PC DOS version 3.1 or later.
- Microsoft Windows version 3.1, 3.11 or Windows 95.

Connecting the MCB251SB

To use the MCB251SB evaluation board, you must:

- Connect it to your PC using the supplied serial cable,
- Connect power using the supplied wall transformer.

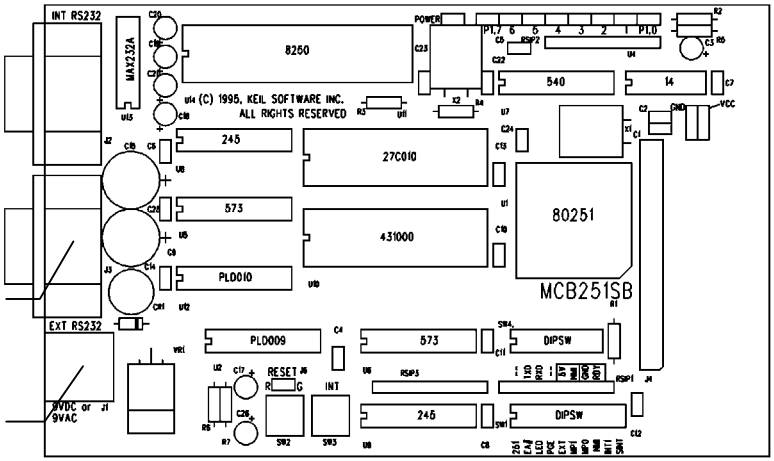
The serial cable lets your PC download program code and debug your target applications. The power cable provides power to the MCB251SB evaluation board. The MCB251SB does not get power from the PC via the serial cable.

Perform the following steps to connect the MCB251SB evaluation board.

1. Connect the MCB251SB to your PC using the provided serial cable. Connect the EXT RS232 jack on the MCB251SB to an available serial port on your PC.

2. Connect the supplied transformer to a source of 120 volts AC. Plug the jack from the transformer into socket J1 on the board. The jack is the power switch for the board. Simply remove the jack from the socket to turn the unit off and replace it to power the unit on.

The external serial port jack (EXT RS232) and the power supply jack (J1) are shown in the following illustration.



External Serial Jack (EXT RS232)

Power Supply Jack (J1)

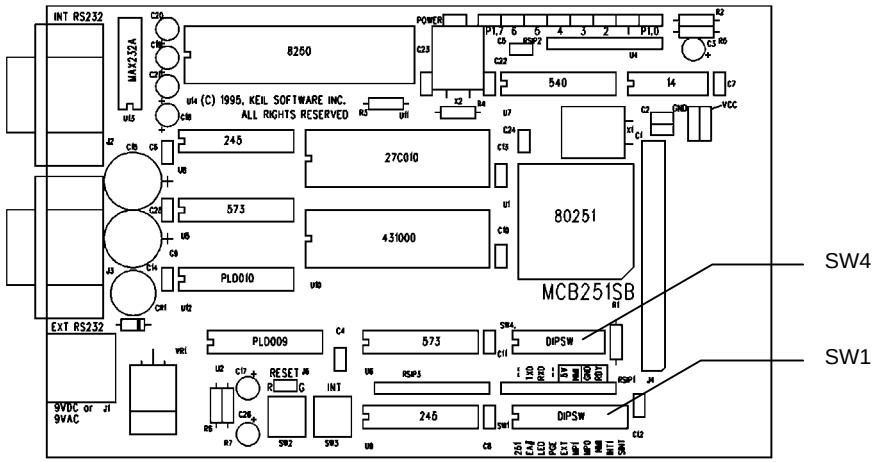
Configuring the MCB251SB

You configure how the MCB251SB evaluation board works using the DIP switches (labeled SW1 and SW4) on the board and the configuration bytes in the 251SB microcontroller. The MCB251SB evaluation board is shipped with the following configuration:

- Intel 87C251SB-16 microcontroller,
- 251 Source Mode,
- Page Mode,
- 1 Wait State,
- 128 Kbyte Address Space,
- 4-Bytes Pushed on Interrupts,
- Internal EPROM from FF:0000h to FF:3FFFh,
- External RAM from FE:0000h to FE:FFFFh and FF:4000h to FF:FBFFh,
- Port 1 connected to LEDs,
- Port pins 3.0 and 3.1 used for serial I/O,
- Port pin 3.2 used for external UART interrupt generation,
- Port pin 3.3 used for input from the INT push-button (SW3).

DIP Switches

The following figure shows the location of the DIP switches on the MCB251SB board.



The default DIP switch settings are shown in the following tables.

SW1	1	2	3	4	5	6	7	8	9	10
ON	X		X	X	X				X	X
OFF		X				X	X	X		

SW4	1	2	3	4	5	6	7	8
ON		X	X		X		X	
OFF	X			X		X		X

For more information about the meanings of the DIP switch settings, refer to “Appendix C. DIP Switch Settings” on page 59.

Configuration Registers

The Intel 251SB has several internal configuration registers that you must program to configure the part. The configuration bytes of the 251SB microcontroller are listed in the following tables.

CONFIG0 Configuration Register

Bit	Label	Function
7 , 6	**—**	Reserved: Set these bits (1) when writing CONFIG0.
5	**WSA**	Wait State A: Set this bit (1) for no wait states. Clear this bit (0) to set 1 wait state for regions 00:, FE:, and FF:.
4	**XALE**	Extend ALE: Set this bit (1) for $T_{ALE} = T_{OSC.}$ Clear this bit (0) for $T_{ALE} = 3T_{OSC}$ (adds one external wait state).
3 , 2	**RD1 , RD0**	RD# and PSEN# Function Select Bits: (0,1)
1	**PAGE**	Page Mode: Set this bit for non-page mode (P2=A8-15, P0=AD0-7). Clear this bit for page mode (P2=A8-15/D0-7, P0=A0-7).
0	**SRC**	Source/Binary Mode: Set this bit for Source Mode Operation. Clear this bit for Binary Mode Operation.

RD1	RD0	RD# Range	PSEN# Range	Features
0	0	Reserved	Reserved	Reserved
0	1	RD# = A16	All addresses	128KB external address space
1	0	P3.7 only	All addresses	One additional port pin
1	1	≤7F:FFFFh	≥80:0000h	Compatible w/MCS® 51 devices

CONFIG1 Configuration Register

Bit	Label	Function
7-5	**—**	Reserved: Set these bits (1) when writing CONFIG1.
4	**INTR**	Interrupt Mode: Set this bit (1) to make interrupts push 4 bytes onto the stack (PC register (3 bytes), and PSW1 register (1 byte)). Clear this bit (0) to make interrupts push only the lower two bytes of the PC register onto the stack.
3	**WSB**	Wait State B: Set this bit (1) for no wait states. Clear this bit (0) for one external wait state in region 01:.
2 , 1	**—**	Reserved: Set these bits (1) when writing CONFIG1.
0	**EMAP**	EPROM Map: Set this bit to map the upper 8 Kbytes of on-chip code memory to FF:2000h-FF:3FFFh (normal location). Clear this bit to map the upper 8 Kbytes of on-chip code memory to 00:E000h-00:FFFFh.

The 251SB included with the MCB251SB evaluation board has been programmed as follows:

Register	Value
CONFIG0	0D5h
CONFIG1	0F7h

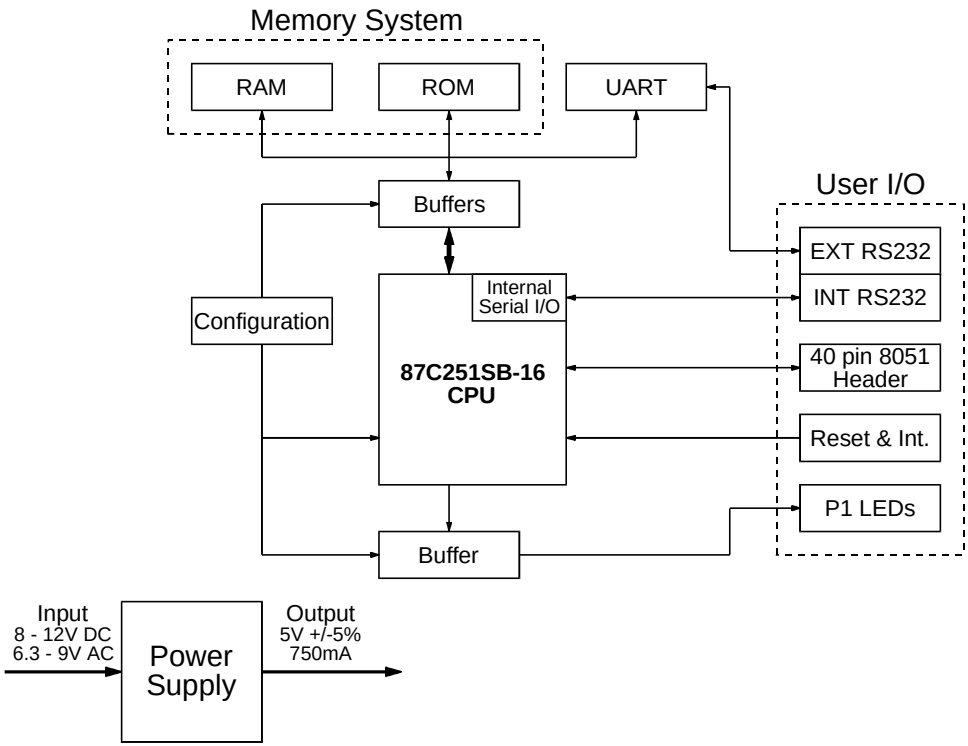
For more information about the configuration registers, refer to the Intel *8XC251SB Embedded Microcontroller User's Manual*.

Chapter 3. Theory of Operation

The MCB251SB is designed to be a very flexible evaluation board that you can use to compare the effectiveness of the 251 versus the 8051. We have attempted to support all features of the 251 while remaining moderately compatible with the 8051. We have also tried to provide a board that can be expanded to support your own hardware prototypes.

This chapter describes logical sections of the MCB251SB and also provides a circuit description of each. The descriptions here will help you understand how the MCB251SB board works and how you can easily interface to the various I/O devices available.

The following block diagram shows the various memory, I/O, configuration, and power systems that compose the board.



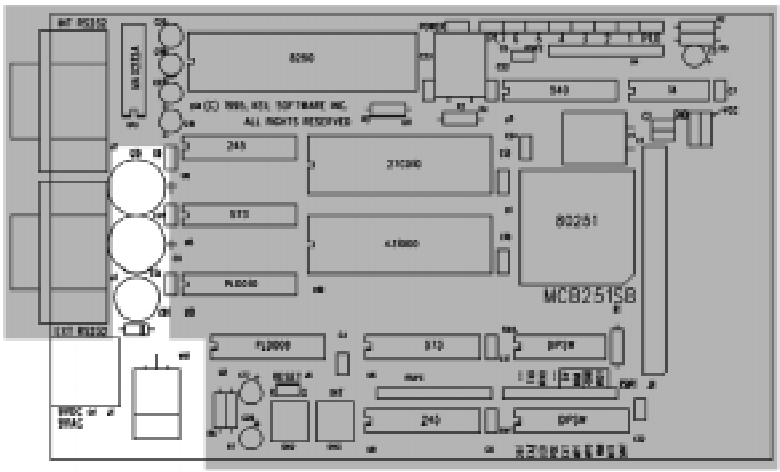
The MCB251SB evaluation board can easily be divided into the following logical sections.

- Power Supply
- CPU
- Configuration
- Buffers and Decode Logic
- Memory
- PORT1 (LEDs)
- Push Buttons
- Serial Ports
- Prototyping Area

These logical sections, the components used, and the circuit locations are described on the following pages.

Power Supply Circuitry

The following figure shows the power supply section of the MCB251SB.

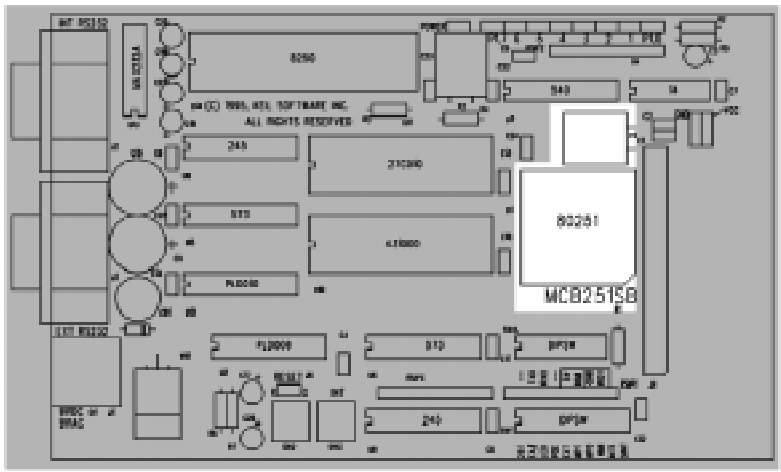


Power is supplied to the MCB251SB from an external 9-12 Volt AC or DC power supply which is capable of providing 300-500mA. Connection is made using a standard 5.5mm barrel plug with a 2.5mm center hole.

The barrel jack located at J1 accepts such a plug. Polarity is not important since a full-wave bridge rectifier is used. 5 Volts DC is provided to the board by the 7805 voltage regulator at VR1.

CPU Circuitry

The following figure shows the CPU section of the MCB251SB.



The Intel 87C251SB16 provided with the MCB251SB is a 251 derivative with 16 Kbytes of internal EPROM space. This part is located at U1.

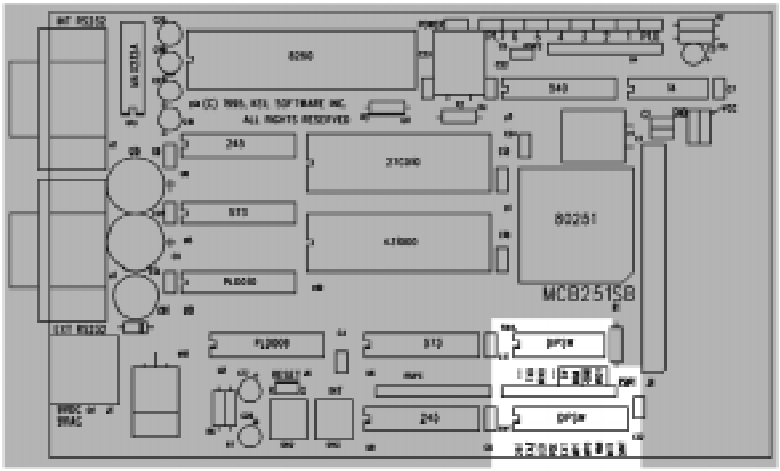
The configuration bytes of the CPU provided with the board are already programmed for source mode, page mode, 1 wait state, and 4-byte interrupt frames. The configuration byte values are as follows:

Register	Value
CONFIG0	0D5h
CONFIG1	0F7h

A 15.000 MHz oscillator at X1 provides the clock signal for the CPU.

Configuration Circuitry

The following figure shows the configuration section of the MCB251SB.



The MCB251SB is a very flexible evaluation board. You can change the operation of the board using the 2 sets of DIP switches (SW1 and SW4). Features such as CPU type, external or internal code memory, LEDs, page mode, memory map, push button inputs, external UART interrupts, and on-chip serial port can all be configured using these switches.

Note
DIP switch settings must coincide with the configuration bytes programmed into the 251 CPU.

The default settings for the DIP switches are shown in the following tables.

SW1	1	2	3	4	5	6	7	8	9	10
ON	X		X	X	X				X	X
OFF		X				X	X	X		

SW4	1	2	3	4	5	6	7	8
ON		X	X		X		X	
OFF	X			X		X		X

Caution

Improper DIP switch settings may cause the MCB251SB evaluation board to malfunction. In particular, the following settings could cause component failure:

- *Setting both SW4-7 and SW4-8 ON could damage the RDY signal source (if present).*
- *Setting SW4-5, SW4-6, and SW1-8 ON while pressing SW3 will damage U4.*

These settings are not likely to be found in any normal configuration of the MCB251SB.

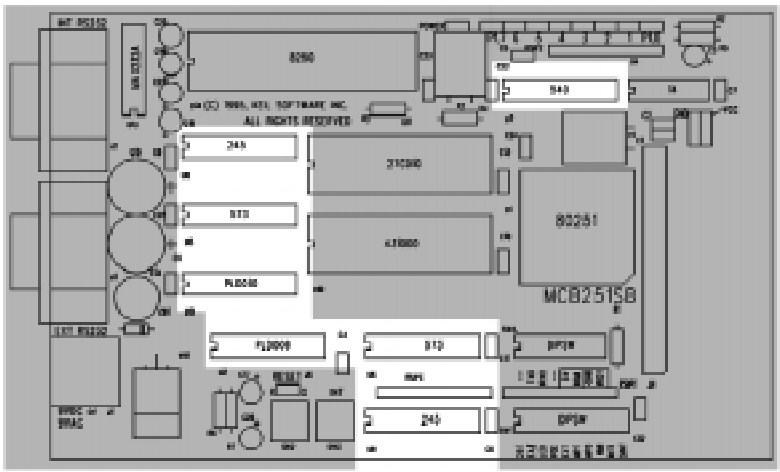
Note

You must RESET the MCB251SB after changing the state of any DIP switch.

Refer to "Appendix C. DIP Switch Settings" on page 59 for a complete description of the DIP switch settings.

Buffers and Decode Logic Circuitry

The following figure shows the buffers and decode logic section of the MCB251SB.



Buffers

The MCB251SB provides complete support for 251 page mode and non-page mode. To do this, the MCB251SB includes 2 sets of buffers and latches for the data/address bus.

In non-page mode, the 251 multiplexes the address bus (A0-A7) and data bus (D0-D7) on port 0. This is the same way the 8051 works. To support non-page mode, the **PGE** DIP switch must be off. In non-page mode, the address bus (port 0) is latched using the 74HCT573 at U6. The data bus (port 0) is buffered using the 74HCT245 at U9. Config byte 0, bit 1 must be set (1) for non-page mode to function. Refer to "Configuration Registers" on page 7 for details about the CPU configuration registers.

In page mode, the 251 multiplexes the address bus (A8-A15) and data bus (D0-D7) on port 2. To support page mode, the **PGE** DIP switch must be on. In page mode, the address bus (port 2) is latched using the 74HCT573 at U5. The data bus (port 2) is buffered using the 74HCT245 at U8. Config byte 0, bit 1 must be clear (0) for page mode to function. Refer to "Configuration Registers" on page 7 for details about the CPU configuration registers.

In addition to the address latches and data buffers, a 74HCT540 at U7 drives the 8 red LEDs from the port 1 output signals. You may disable the LEDs by setting the **LED** DIP switch off.

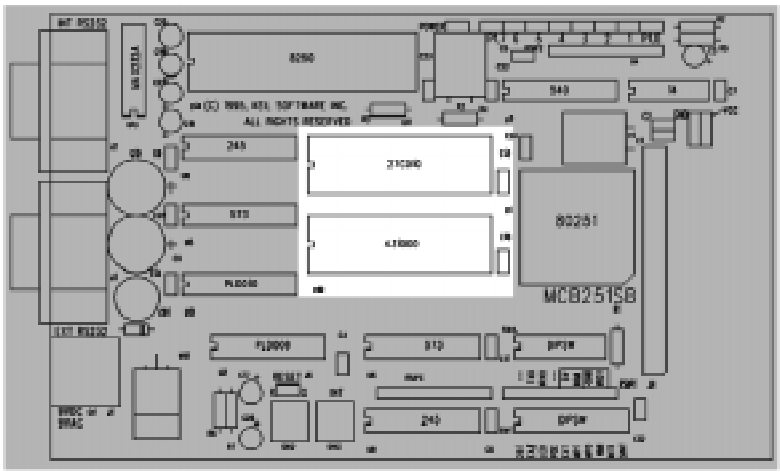
Decode Logic

All memory address decode logic as well as other signal conversion is performed by the 16V8 PALs at U2 and U12.

Refer to "Appendix B. PAL Equations" on page 55 for a complete listing of the PAL equations used.

Memory Circuitry

The following figure shows the memory section of the MCB251SB.



The MCB251SB maps three devices into the memory address space of the 251: RAM at U10, ROM at U11, and the external UART at U14. 32-pin DIP sockets are provided for the RAM and ROM.

You may use 27-series EPROMs in the ROM socket at U11. EPROMs with 28 pins must be installed with pin 14 of the EPROM in pin 16 of the socket. The following parts are compatible with the MCB251SB:

Part Number	Size	Pins
27128	16K × 8	28
27256	32K × 8	28
27512	64K × 8	28
27010	128K × 8	32

The MCB251SB is provided with a 128K × 8 SRAM installed in U10.

The external UART, RAM, and ROM are memory mapped according to the DIP switch settings of **MP0**, **MP1**, and **251**.

When the **251** DIP switch is on, the MCB251SB is configured for the Intel 251SB CPU and the following settings apply to the **MP0** and **MP1** switches.

251 Memory Map				
MP1 (SW1-6)	MP0 (SW1-7)	SRAM	EPROM	16450 UART
OFF	OFF	0:0000h–1:FBFFh	—	1:FE00h–1:FFFFh
OFF	ON	—	0:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	OFF	0:0000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	ON	0:8000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh

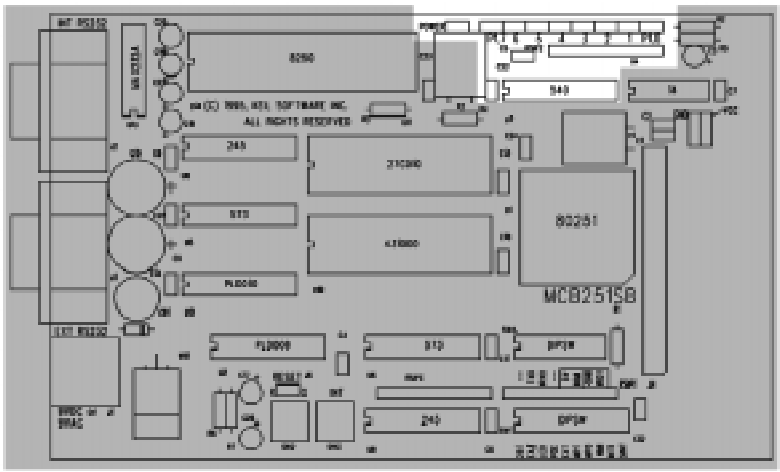
When the **251** DIP switch is off, the MCB251SB is configured for a standard 8051 CPU and the following settings apply to the **MP0** and **MP1** switches.

8051 Memory Map				
MP1 (SW1-6)	MP0 (SW1-7)	SRAM	EPROM	16450 UART
OFF	OFF	C:0000h–C:FFFFh X:0000h–X:FFFFh	—	—
OFF	ON	—	C:0000h–C:DFFFh X:0000h–X:DFFFh	—
ON	OFF	C:0000h–C:FFFFh X:0000h–X:FFFFh	—	—
ON	ON	C:8000h–C:FFFFh X:8000h–X:FFFFh	—	—

Note
Even though the MCB251SB supports the 8051 CPU, many of the features of the board are only available when the 251 CPU is used.

PORT 1 (LEDs) Circuitry

The following figure shows the LED section of the MCB251SB.

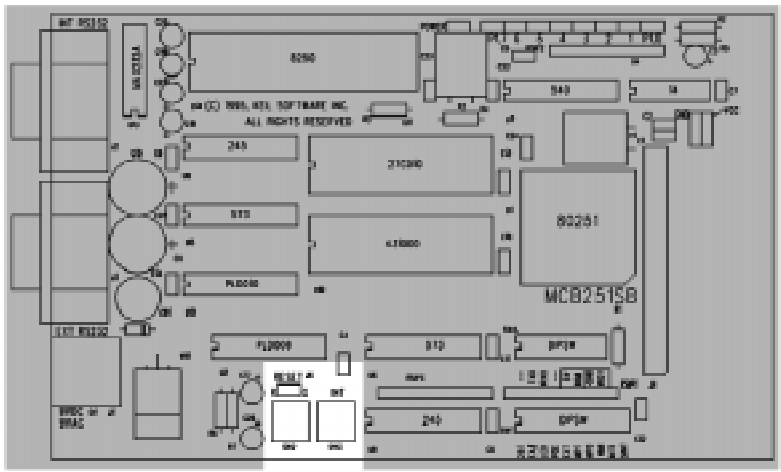


The MCB251SB has a single green power LED which indicates the power to the board is on.

Eight red LEDs are optionally connected to the port 1 outputs through a 74HCT540 at U7. You may disable the LED driver by setting the **LED** DIP switch (SW1-3) off.

Push Button Circuitry

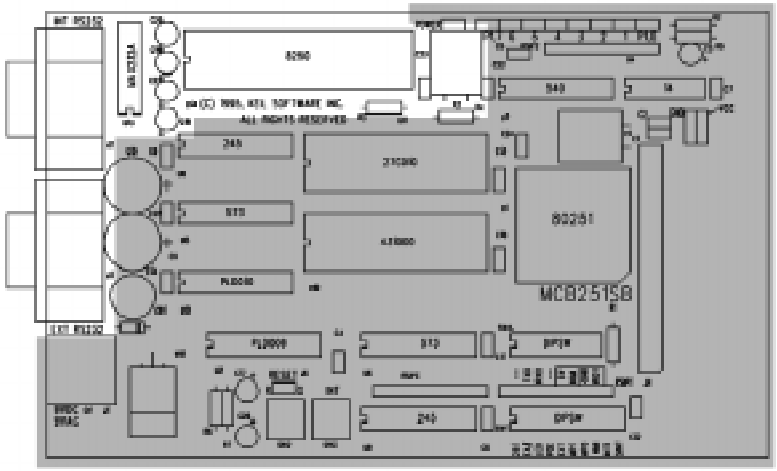
The following figure shows the push buttons section of the MCB251SB.



The MCB251SB provides two push buttons. The first push button, labeled RESET and located at SW2, is connected to the reset circuit and may be used to reset the CPU. The second button, labeled INT and located at SW3, is optionally connected to the INT1 (port 3.3) of the CPU. When the **INT1** DIP switch is on, the INT push button is connected to the INT1 pin of the CPU.

Serial Port Circuitry

The following figure shows the serial ports section of the MCB251SB.



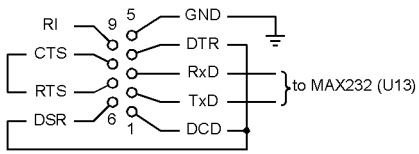
The MCB251SB supports both the on-chip serial port and an external UART. Both of these use the MAX232, at U13, to convert RS-232 voltage levels.

Internal Serial Port

You may use the **TXD** and **RXD** DIP switches to connect and disconnect the on-chip serial port to the MAX232. Set these switches on to connect the transmit and receive lines of the on-chip serial port to the MAX232.

The internal serial port is derived from the internal serial functions of the 8051/80251 (P3.0/RXD and P3.1/TXD).

The port is configured as a standard 3-wire interface only. The handshaking signals are connected to loop the PC's signals back. Refer to the figure at the right to determine how the DB9 connector for this port is wired.



The baud rate for the internal serial port is derived from the system clock. The following table lists the standard BPS rates for the MCB251SB board using a 15.000 MHz crystal.

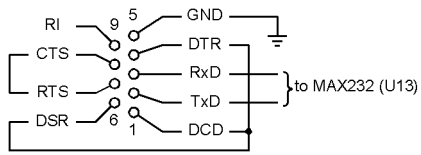
	SMOD = 0		SMOD = 1	
BPS	**TH1**	**Actual BPS**	**TH1**	**Actual BPS**
300	07Eh	300.48	N/A	N/A
600	0BFh	600.96	07Eh	600.96
1200	0E0h	1220.70	0BFh	1201.92
2400	0F0h	2441.41	0E0h	2441.41
4800	0F8h	4882.81	0F0h	4882.81
9600	0FCh	9765.63	0F8h	9765.63

Use the DB9 connector at J2 for interfacing to the internal serial port.

External UART

The external UART, at U14, is implemented using an 8250/16450/16550 device. This is the same chip that is used in most IBM PC compatible computers. A 1.8432 MHz oscillator, at X2, provides the input frequency to the UART. The interrupt output of the UART is optionally connected to the INT0 pin (port 3.2) of the CPU. When the **SINT** DIP switch is on, the UART interrupt is connected to the INT0 CPU pin.

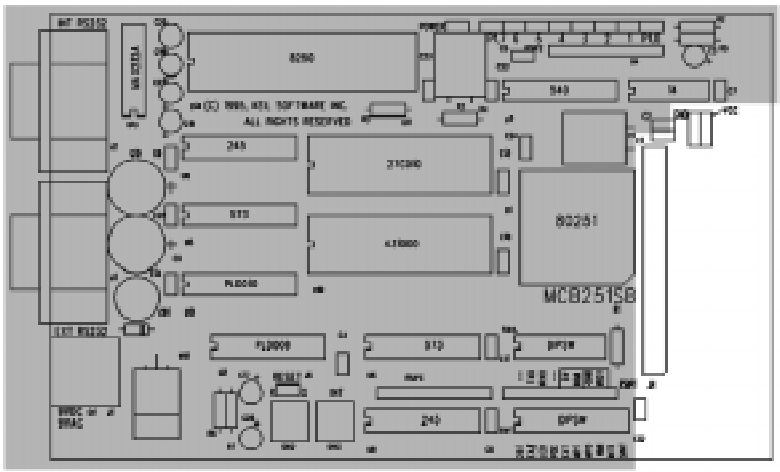
The external UART is configured as a 3-wire interface. The handshaking signals loop the PC's signals back. The Keil 251 Monitor (MON251) is configured to use this serial port. This gives you free access to the internal serial port.



Use the DB9 connector at J3 for interfacing to the external UART.

Prototyping Area

The following figure shows the prototyping section of the MCB251SB.

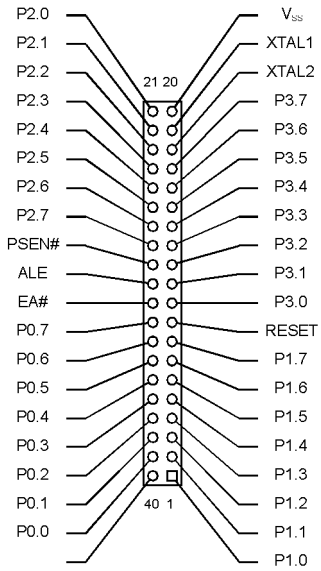


A perforated area is provided on the MCB251SB for prototyping your own hardware. The prototyping area also provides blocks of +5 volts (VCC) and ground (GND).

The connector at J4 provides the signals necessary to create a 40-pin DIP plug for emulating an 8051 or 251 in a 40-pin DIP package. All signals are provided at the connector except for V_{CC}. If you need V_{CC} at this connector, you must modify the MCB251SB evaluation board.

The figure at the right shows the pin labels and pin numbers for the connector. This connector is numbered and wired to correspond with a 40-pin DIP 8051 package.

The signals provided to the J4 connector are driven directly by the CPU. Exercise caution to avoid overloading these signal lines.



Note

Since the MCB251SB supports both the page mode and non-page mode, and because separate data buffers are used to implement these modes, it is not possible to connect memory-mapped devices directly to the data bus on the J4 connector. Instead, you should connect the address bus to the output of the 74HCT573s (at U5 or U6) and the data bus to the input/output of the 74HCT245s (at U8 or U9).

Chapter 4. Programming

Writing programs for the MCB251SB is relatively simple. However, before you get started, there are a few points you should keep in mind.

- The 251SB is the first of a series of new chips that support a new hardware architecture that is different from that of the 8051.
- The 251SB begins executing instructions at address 0FF0000h. The MCB251SB is shipped with the Keil 251 monitor programmed into the 251SB. If you leave the **EA#** DIP switch off, the monitor program begins running when you apply power to the board.
- If you use the built-in 251 monitor for loading and running programs via the external serial port on the MCB251SB, your programs must begin at address 0FF4000h. If you do not use the built-in monitor, your programs must begin at address 0FF0000h.
- The addresses 0FE0000h and 0:0000h and the addresses 0FF0000h and 1:0000h are synonymous and are used interchangeably in this manual. Refer to the Intel *8XC251SB Embedded Microcontroller User's Manual* for details on how the 251 CPU maps these memory areas.

This chapter describes everything you need to know to write programs for the MCB251SB evaluation board.

Internal Program Memory

The 251 CPU provided with the MCB251SB has 16K of internal program memory. As provided, the 251 CPU is programmed with the Keil 251 monitor program. This program lets you download and debug your programs using with the Keil dScope debugger for Windows or the MON251 terminal program (MON251.EXE). The first 16K of code space (1:0000h-1:3FFFh) comes from the internal program memory in the 251 CPU. You can disable the on-chip memory using the **EA#** DIP switch.

Memory Map

The MCB251SB evaluation board supports 4 different memory maps for the on-board RAM, ROM, and external UART. The memory map is selected using the **MP0** and **MP1** DIP switches. The possible combinations are listed in the following table.

251 Memory Map				
MP1 (SW1-6)	MP0 (SW1-7)	SRAM	EPROM	16450 UART
OFF	OFF	0:0000h–1:FBFFh	—	1:FE00h–1:FFFFh
OFF	ON	—	0:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	OFF	0:0000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	ON	0:8000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh

The 251 CPU provided with the MCB251SB evaluation board is programmed so that the RD# signal provides the A16 address line. This lets the 251 address up to 128K of external address space.

You may program your own 251 so that the RD# line is available only as a port pin. In this case, you must set the **EXT** DIP switch off to disable the external memory system.

You may program your own 251 so that the RD# line is 100% compatible with the 8051. In this case, you must set the **251** DIP switch to off to configure the board for 8051-style signals.

The following descriptions apply to the 251 CPU supplied with the MCB251SB.

External RAM

When you use the built-in 251 monitor to download and test your 251 programs, you should switch **MP0** and **MP1** to off. When these DIP switches are off, RAM is mapped into the 0:0000h-1:FBFFh address range of the 251. This setting gives you the greatest flexibility when developing new programs since you can easily download and test them using the 251 monitor and the dScope debugger.

External EPROM

When you want the entire external address space devoted to program code, set the **MP1** switch off and the **MP0** switch on. This maps the external EPROM into the 0:0000h-1:DFFFh address range of the 251. This setting gives you the largest amount of program space. However, you must program an EPROM with your program code.

External EPROM and External RAM

When you want half of the external address space devoted to EPROM and half devoted to RAM, set the **MP1** switch on and the **MP0** switch off. This setting maps the external RAM into the 0:0000h-0:FFFFh address range of the 251 and the external EPROM into the 1:0000h-1:DFFFh address range. This gives you a full 64K of RAM and 56K of EPROM space.

External EPROM, External RAM, and Memory-Mapped I/O

When you want half of the external address space devoted to EPROM but only 32K devoted to RAM, set the **MP1** switch and the **MP0** switch to on. This setting maps the external RAM into the 0:8000h-0:FFFFh address range of the 251 and the external EPROM into the 1:0000h-1:DFFFh address range. This gives you a full 32K of RAM and 56K of EPROM space. You may use the lower 32K of the address space for your own memory-mapped I/O devices.

Writing Programs for the 251 Monitor

Compiling and linking programs for use with the 251 monitor on the MCB251SB board requires only two steps.

First, compile your applications as you normally would but use the **MODSRC** directive to specify 251 source mode. For example, you may use the following command line.

```
C251 MYCODE.C MODSRC
```

Second, when you link your object files, use the **CLASSES** directive shown in the following example command line.

```
L251 MYCODE.OBJ CLASSES(CODE($0FF4000h,0FF4000h))
```

In this command line, $0FF4000h tells the debugger where the program begins and the 0FF4000h tells the linker where to begin locating the CODE segment.

You may use the OH251 utility to create an Intel HEX file from the absolute object module created by the linker. Use the following command line to create a HEX file:

```
OH251 MYFILE
```

You may use either Intel HEX files or absolute object modules with the dScope debugger and the MON251 terminal program. Absolute object modules include source-level debugging information but HEX files do not.

Writing Programs for EPROM

Compiling and linking programs for programming into EPROM is very similar to creating programs for use with the monitor.

Compile your applications as you normally would using the **MODSRC** directive to specify 251 source mode. For example:

```
C251 MYCODE.C MODSRC
```

When you build programs for EPROM, you do not need to tell the linker or the debugger about the program's beginning address or where the CODE segment is located (0FF0000h is assumed).

Finally, use the OH251 utility to create an Intel HEX file from the OMF251 absolute object module the linker creates. You may program your EPROM using the created HEX file.

FLASH Example Program

The following simple program, FLASH, is an exercise you may use to test the MCB251SB and verify that you can use the tools provided to generate a working program.

FLASH does nothing more than blink the LEDs on the MCB251SB evaluation board. The complete source listing for the program is as follows:

```
/* FLASH.C - LED Flasher for the Keil MCB251SB Evaluation Board */

#include <reg251s.h>                  /* Include 251SB header file */

void main (void)
{
for (;;)                              /* Loop forever */
  {
  register unsigned int i;            /* Delay var */
  register unsigned char j;           /* LED var */

  for (j = 0x01; j < 0x80; j <<= 1)   /* Blink LED 0, 1, 2, 3, 4, 5, 6 */
    {
    P1 = j;                           /* Output to LED Port */
    for (i = 0; i < 1000; i++);       /* Delay for 1000 Counts */
    }

  for (j = 0x80; j > 0x01; j >>=1)    /* Blink LED 7, 6, 5, 4, 3, 2, 1 */
    {
    P1 = j;                           /* Output to LED Port */
    for (i = 0; i < 1000; i++);       /* Delay for 1000 Counts */
    }
  }
}
```

You may build the flash example program either using the 251 tools from the MS-DOS command line or you may use the µVision integrated development environment and dScope. Both methods are described below.

Using the MS-DOS Command Line Tools

Use the following command line to compile the FLASH example program:

```
C251 FLASH.C DEBUG MODSRC
```

Use the following command line to link the FLASH example program for use with the 251 monitor:

```
L251 FLASH.OBJ CLASSES(CODE($0FF4000h,0FF4000h))
```

To link the FLASH example program for EPROM, use the following command line:

```
L251 FLASH.OBJ
```

Using the Windows-Based Tools

This section leads you step-by-step through the process of creating the FLASH example program and testing it using the µVision IDE and dScope debugger.

Using µVision to Create the FLASH Program

To create the FLASH example program using µVision, you need to perform the following steps:

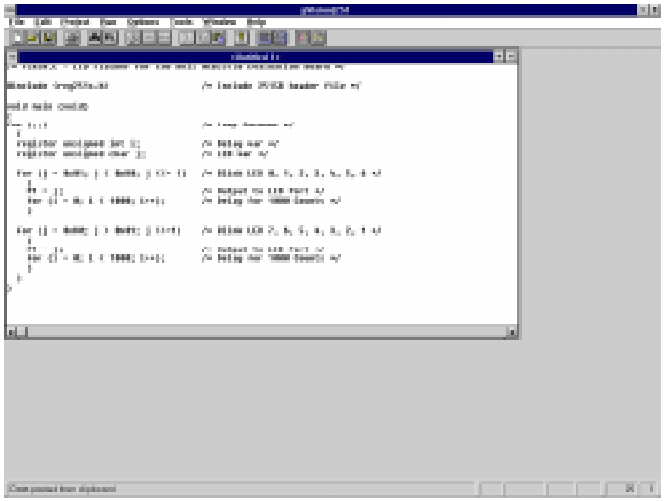
- Create the FLASH.C source file.
- Create the FLASH project file.
- Include FLASH.C in the project.
- Set the C251 compiler options for the project.
- Set the L251 linker options for the project.
- Set the path specifications for the 251 tools (if necessary).
- Set the make options for the project.
- Build the project.

Each of these steps is described in detail below.

Start µVision by double-clicking on the icon in the 251 Tools group.

When µVision starts, select the New command from the File menu and µVision opens a new text window in which you may create the FLASH program.

Enter the FLASH example program shown on page 30. You screen should look something like the following figure.



You should save the FLASH program after you enter it. Select the Save As… command from the File menu and µVision displays the Save As dialog box shown below.

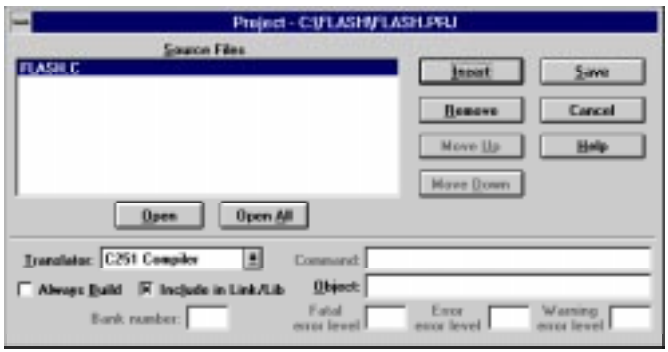
To save the program as FLASH.C, enter FLASH.C in the File Name text box at the top of the Save As dialog box. You may want to save the flash program and the other files you create in a separate directory. This example uses C:\FLASH for the source files and project files.

After you save FLASH.C, you should create a FLASH project file. A project file contains a list of all the source files in your project as well as the options to use for the compiler, assembler, linker, and make facility. Additionally, the project manager helps you compile, link, and test your target program.

To create a project file for FLASH, select the New Project… command from the Project menu. µVision displays the dialog box shown below. Enter the name for the project in the File Name text box. This example uses FLASH.PRJ.

NOTE
*You should always use **PRJ** as the file extension for project files.*

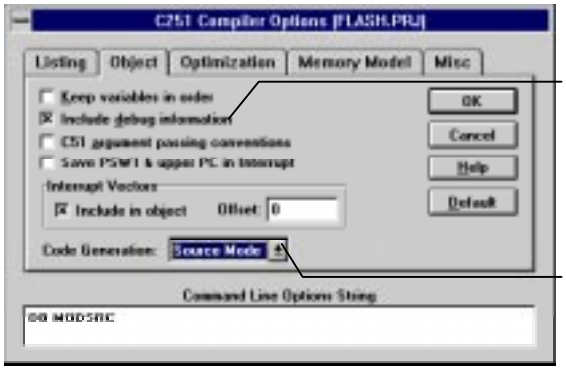
When the project file is created, µVision displays the Project Manager dialog box. Here, you select the source files to include in your project.



Click the Insert button and choose the FLASH.C file you previously saved. Then, click the save button to save your changes to the project file. To return to the Project Manager dialog box, select the Edit Project… command from the Project menu.

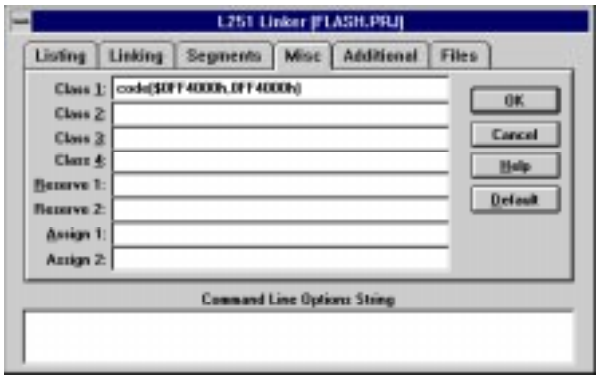
When you have created a project file and inserted the source files into the project, you are ready to set the options for the compiler, linker, and other tools.

To set the C251 compiler options, select the C251 Compiler… command from the Options menu and µVision displays the C251 Compiler Options dialog box. The only options you need to set for the FLASH example are Include debug information and Source Mode for Code generation. These controls are shown in the following figure.



NOTE
In any of the option dialog boxes, you may click the Default button to set the controls to the default settings.

To set the L251 linker options, select the L251 Linker… command from the Options menu. µVision displays the L251 Linker Options dialog box shown below.

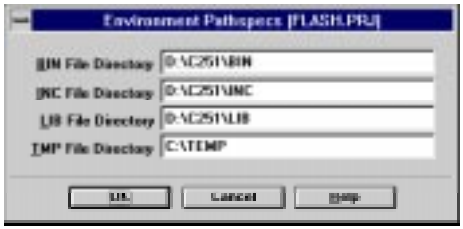


Enter the following text in one of the Class text boxes on the Misc tab shown above.

```
CODE($0FF4000h,0FF4000h)
```

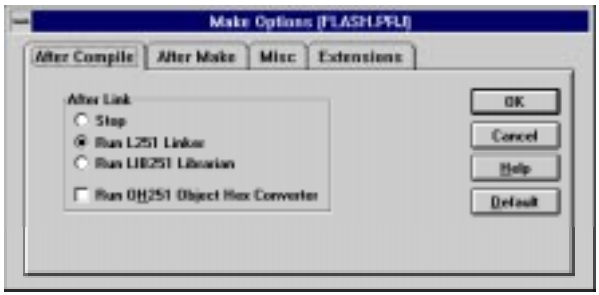
This creates the class assignment that tells the linker that code space starts at 0FF40000h. This is required to download the FLASH program to the 251 monitor.

You may wish to specify the path to the 251 tools directly in µVision rather than rely on the settings in AUTOEXEC.BAT. You may do this in the Environment Pathspecs dialog box. Open this dialog box using the Environment Pathspecs… command in the Options menu.



You may specify the path to BIN directory, the INC directory, and the LIB directory. Additionally, you may specify a temporary directory for the compiler and linker to use when compiling and linking.

Finally, the make options control how µVision processes the files in your project. Open the Make Options dialog box by selecting the Make… command in the Options menu.



Make sure you select the Run L251 Linker radio button. This links your source files after compiling them.

Now, you are ready to build the FLASH project. Select the Make: Build Project command from the Project menu to begin compiling and linking. µVision responds by compiling the FLASH.C source file and linking it with the appropriate library files.

This process is called *making* the project. While the make is running, µVision displays the status as shown below.

If errors occur during the make process, a message window appears. If there were warnings or errors in your source file, you may interactively select the error and see the corresponding line in your source file.

When make completes successfully, you are ready to begin debugging the FLASH program.

Using dScope to Debug the FLASH Program

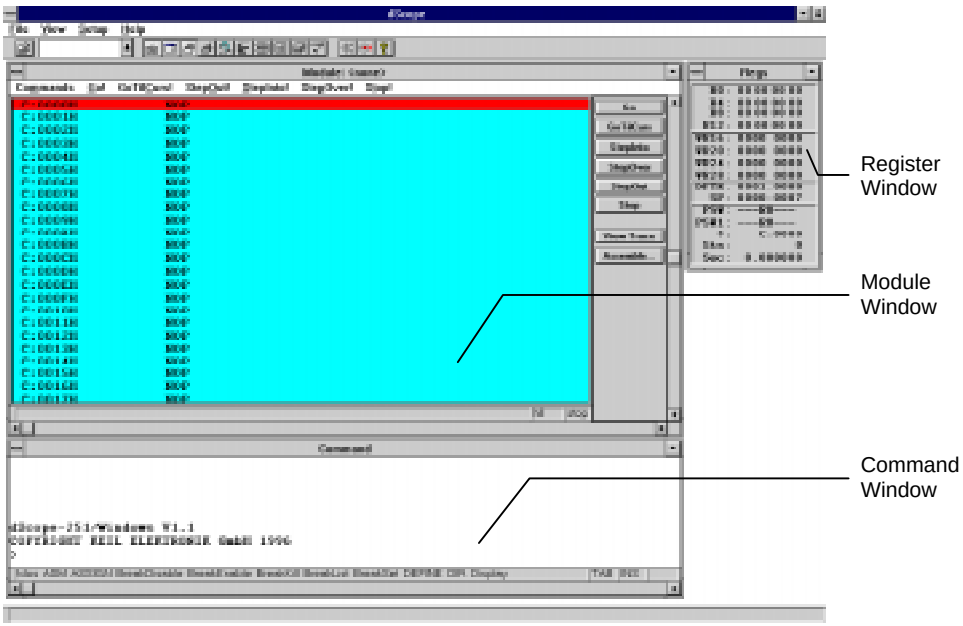
To load the FLASH in the MCB251SB evaluation board using dScope, you need to perform the following steps:

- Load the MON251.DLL CPU driver.
- Configure the CPU driver for the appropriate COM port and baud rate.
- Configure the CPU driver for serial break.
- Load the FLASH program.
- Step through the FLASH program.

Each of these steps is described in detail below.

Start dScope by selecting the dScope Debugger… command from the Run menu. This loads dScope and sets the current path to the path in which your project file is saved.

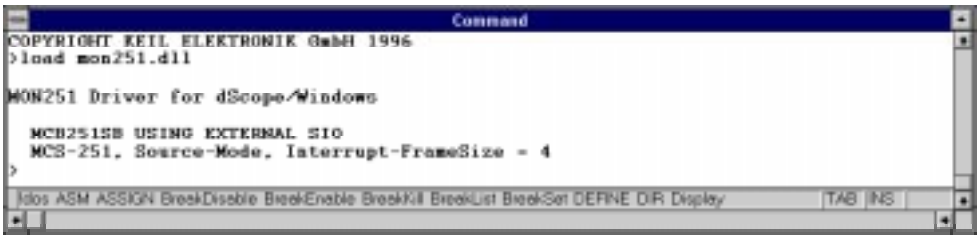
When dScope starts, a screen similar to the following displays.



To load the MON251.DLL CPU driver, type the following in the command window.

```
load mon251.dll
```

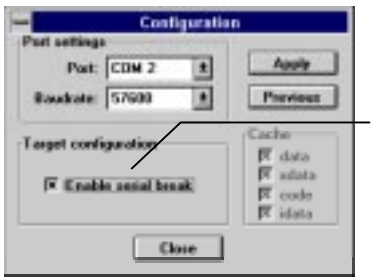
This is shown in the following figure.



Typically, the first time you load the CPU driver for the 251 monitor, you must set the COM port and baudrate. If dScope cannot determine the com port and the baud rate automatically, the following dialog box appears informing you that dScope could not find the target system.

When using the MCB251SB evaluation board, the baudrate should be set for 57,600 baud. You must determine which COM port you are using.

Next, you should configure the CPU driver to enable serial breaks. To do this, select the Configuration command from the Peripherals menu. dScope displays the Configuration dialog box shown below.

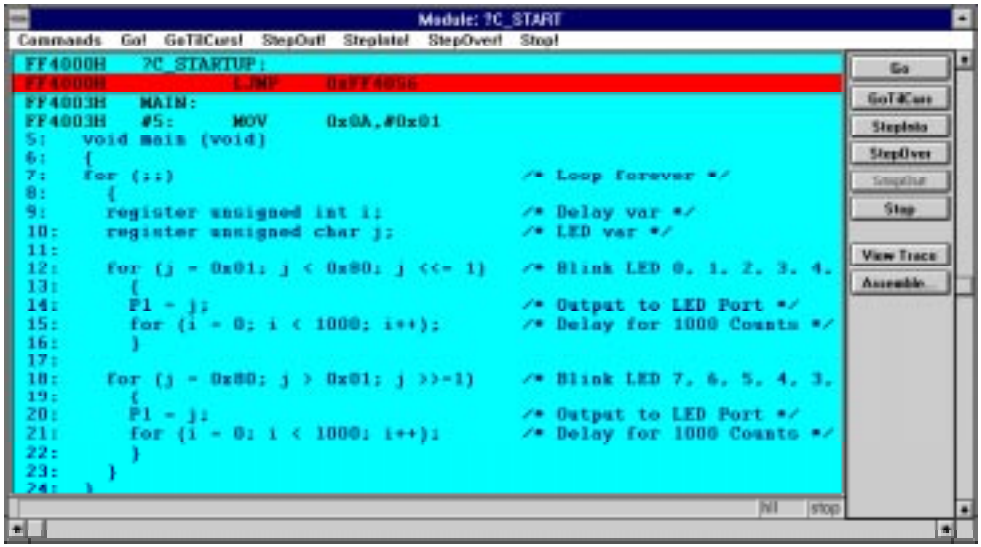


Set Enable serial break to let dScope stop programs running on the MCB251SB.

Finally, you are ready to load the FLASH program. To do so, type the following in the command window.

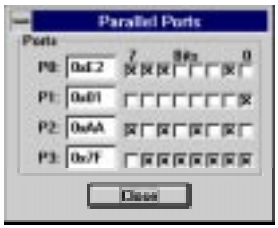
```
load flash                              /* this loads the flash program */

g,main                                 /* this steps over the startup code */
                                     /* and stops on the first line of main */
```

Once FLASH is loaded, the module window displays the FLASH program as shown in the following figure.



You may now single step through the FLASH program by clicking on the StepOver button on the right side of the module window. As you step through the program, you should see the LEDs on the MCB251SB changing.

You may display the status of the I/O ports in dScope. Select the I/O-Ports command from the Peripherals menu and dScope displays the following dialog box.



As you step through the FLASH program, you will see how the Parallel Ports dialog box mirrors the LEDs on the MCB251SB evaluation board.

When you are ready to exit dScope, click on the Stop button in the module window and select the Exit command in the File menu.

NOTE
You must stop the simulation of your target program before you can exit dScope.

External UART Example

The following example program shows you how to use the external UART to send and receive serial data. The **_getkey** and **putchar** routines are included for use with the stream functions like **printf**.

```
#include <reg251s.h>
#include <stdio.h>

#define DELAY   50000L

#define EXTSIO_ADDR    0xFFFF00UL

#define EXTSIO_RBR     ((volatile unsigned char far *) (EXTSIO_ADDR))
#define EXTSIO_THR     ((volatile unsigned char far *) (EXTSIO_ADDR))
#define EXTSIO_IER     ((volatile unsigned char far *) (EXTSIO_ADDR + 1))
#define EXTSIO_IIR     ((volatile unsigned char far *) (EXTSIO_ADDR + 2))
#define EXTSIO_LCR     ((volatile unsigned char far *) (EXTSIO_ADDR + 3))
#define EXTSIO_MCR     ((volatile unsigned char far *) (EXTSIO_ADDR + 4))
#define EXTSIO_LSR     ((volatile unsigned char far *) (EXTSIO_ADDR + 5))
#define EXTSIO_MSR     ((volatile unsigned char far *) (EXTSIO_ADDR + 6))
#define EXTSIO_SCR     ((volatile unsigned char far *) (EXTSIO_ADDR + 7))
#define EXTSIO_DLL     ((volatile unsigned char far *) (EXTSIO_ADDR + 0))
#define EXTSIO_DLM     ((volatile unsigned char far *) (EXTSIO_ADDR + 1))

#define EXTSIO_CLK     1843200UL

/*-----------------------------------------------
This function sets the baudrate for the external
UART on the MCB251SB.
-----------------------------------------------*/
void extsio_baudrate (
  unsigned long baudrate)
{
unsigned long baud_divisor;
unsigned char lcr_save;

lcr_save = *EXTSIO_LCR;                                  /* Save LCR */
*EXTSIO_LCR = 0x80;                                      /* Set DLAB */

baud_divisor = (EXTSIO_CLK / 16UL) / baudrate;

*EXTSIO_DLL = (unsigned char) (baud_divisor & 0xFF);
*EXTSIO_DLM = (unsigned char) ((baud_divisor >> 8) & 0xFF);

*EXTSIO_LCR = lcr_save;                                /* Restore LCR */
}
```

```
/*---------------------------------------------
This function performs a general initialization of
the external UART on the MCB251SB.
---------------------------------------------*/
void extsio_setup (
  unsigned long baudrate)
{
unsigned char dummy;

extsio_baudrate (baudrate);

*EXTSIO_LCR = 0x03;                          /* 8-bit, 1 stop, no parity */

*EXTSIO_IER = 0x00;                            /* disable all interrupts */

*EXTSIO_MCR = 0x00;

dummy = *EXTSIO_LSR;
dummy = *EXTSIO_MSR;
dummy = *EXTSIO_RBR;
}

/*---------------------------------------------
Send a character out the external UART.
---------------------------------------------*/
void extsio_putchar (
  unsigned char c)
{
while ((*EXTSIO_LSR & 0x20) == 0);

*EXTSIO_THR = c;
}

/*---------------------------------------------
Receive a character from the external UART.  If
no characters have been received, -1 is returned.
---------------------------------------------*/
int extsio_getchar (void)
{
if (*EXTSIO_LSR & 0x01)
  return ((unsigned) *EXTSIO_RBR);

return (-1);
}

/*---------------------------------------------
_getkey replacement for external UART.
---------------------------------------------*/
char _getkey (void)
{
int c;

while ((c = extsio_getchar ()) == -1);

return (c);
}
```

```
/*-----------------------------------------------
putchar replacement for external UART.
-----------------------------------------------*/
char putchar (char c)
{
extsio_putchar (c);
return (c);
}

/*-----------------------------------------------
Setup the external UART for 38400 baud, write the
string "Hello World nnn", and repeat.
-----------------------------------------------*/
void main (void)
{
unsigned int counter = 0;

extsio_setup (38400);

for (counter = 0; ; counter++)
  {
  unsigned long i;

  printf ("Hello World %u\r\n", (unsigned) counter % 100);
  for (i = 0; i < DELAY; i++);
  }
}

/*-----------------------------------------------
-----------------------------------------------*/
```

Use the following command line to compile the external UART example program:

```
C251 EXTSIO.C DEBUG MODSRC
```

Since this example program uses the external UART, you must generate a HEX file and program an EPROM. USe the following command line to link:

```
L251 EXTSIO.OBJ
```

and the following command line to generate an Intel HEX file:

```
OH251 EXTSIO
```

You may then use the EXTSIO.HEX file to program an EPROM.

Chapter 5. Using the 251 Monitor

The MCB251SB comes ready-programmed with the Keil 251 monitor (MON251). The monitor programmed in the 251 CPU is setup to run at 57,600 baud with a 15MHz oscillator.

You may use the MON251 DOS terminal program (MON251.EXE) or the dScope for Windows debugger/simulator to quickly download and test your programs.

MON251 Terminal Program

The MON251 terminal program communicates with the 251 monitor using one of your PC's serial ports. MON251 lets you:

- Display the contents of the 251's memory areas in ASCII and hexadecimal,
- Interactively change memory contents,
- Display and change register contents,
- Initialize memory with constant values,
- Disassemble the code area,
- Assemble in-line code,
- Run programs with breakpoints in real-time,
- Create up to 50 program breakpoints,
- Single-step through your program,
- Step over subroutines,
- Upload and download Intel HEX files and OMF-251 object files,
- Display an online command help menu.

Starting the MON251 Terminal Program

Use the following command line to start the MON251 terminal program.

```
MON251 [COM1|COM2|COM3|COM4] [INT14|NOINT] [BAUDRATE(n)]
```

where:

COM1: … COM4: Specifies the serial port to use. You may use the COM port number as an abbreviation. COM1 is used by default.

INT14 Specifies that the serial interface is accessed through the BIOS software interrupt 14H. Direct hardware interrupts are not used. You may use **I** as an abbreviation for **INT14**.

NOINT Specifies that the serial interface is polled. Hardware interrupts are not used. You may use **N** as an abbreviation for **NOINT**.

BAUDRATE(n) Specifies the baud rate to use. If this option is omitted, MON251 defaults to 9600 baud. The value ***n*** specifies the baud rate. Valid baud rates are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, and 57600. You may use **BR** as an abbreviation for **BAUDRATE**.

The command-line parameters of the MON251 terminal program may be set using the environment variable **MON251**. You may set this environment variable using the DOS SET command. If no parameters are included on the command line, MON251 uses the **MON251** environment variable settings. If the environment variable settings are used, the MON251 terminal program displays the following message:

```
ENVIRONMENT STRING: MON251=<parameters>
```

Utility Commands

The MON251 terminal program supports the following commands:

- **`Exit`** Exit MON251 and return to DOS.
- **`F1`** Exit MON251 and return to DOS.
- **`F2`** Transmit the contents of a file.
- **`F3`** Echo the contents of the screen to a file.
- **`Help`** Display the help menu.
- **`;`** Comment lines.

Exit: Exit MON251 and Return to DOS

This command closes all files and returns to DOS. For example:

```
#EXIT
```

F1: Exit MON251 and Return to DOS

When you enter **`F1`** or **`Alt+1`**, MON251 responds with the following prompt.

```
EXIT MON251 (y or [n])
```

Enter **`Y`** to exit MON251, close all files, and return to DOS.

F2: Transmit the Contents of a File

When you enter **`F2`** or **`Alt+2`**, MON251 responds with the following prompt.

```
Input File:
```

Here, you may enter a file from which MON251 inputs and interprets commands. For example:

```
Input File:          MYINIT.CMD <cr>
```

directs MON251 to read the contents of the file MYINIT.CMD and interpret them as commands. You may press **`Ctrl+C`** to stop retrieving commands from the file.

F3: Echo the Contents of the Screen to a File

When you enter **F3** or **Alt+3**, MON251 responds with the following prompt.

```
Output File:
```

Here, you may enter a file where the contents of the screen will be copied. For example:

```
Output File:      DEBUG.PRN
```

directs MON251 to write the screen contents to DEBUG.PRN. Press **F3** or **Alt+3** to stop writing screen changes.

If you enter the name of a file that already exists, MON251 responds with the followin prompt.

```
Overwrite existing file (y or [n])?
```

Enter **Y** to overwrite the file.

HELP: Display the Help Menu

When you enter the help command, MON251 displays a brief description of all 251 monitor commands.

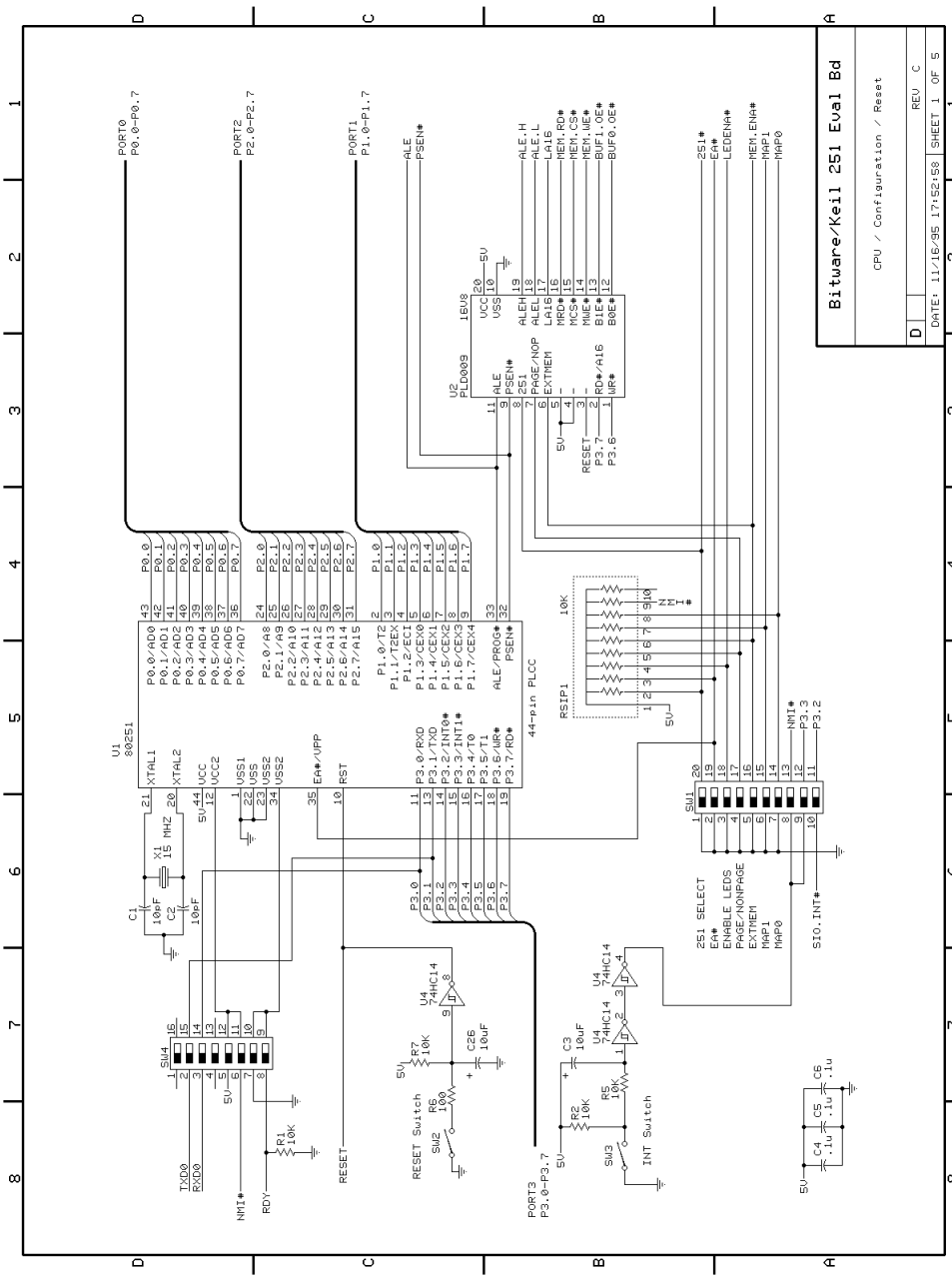
;: Comment Lines

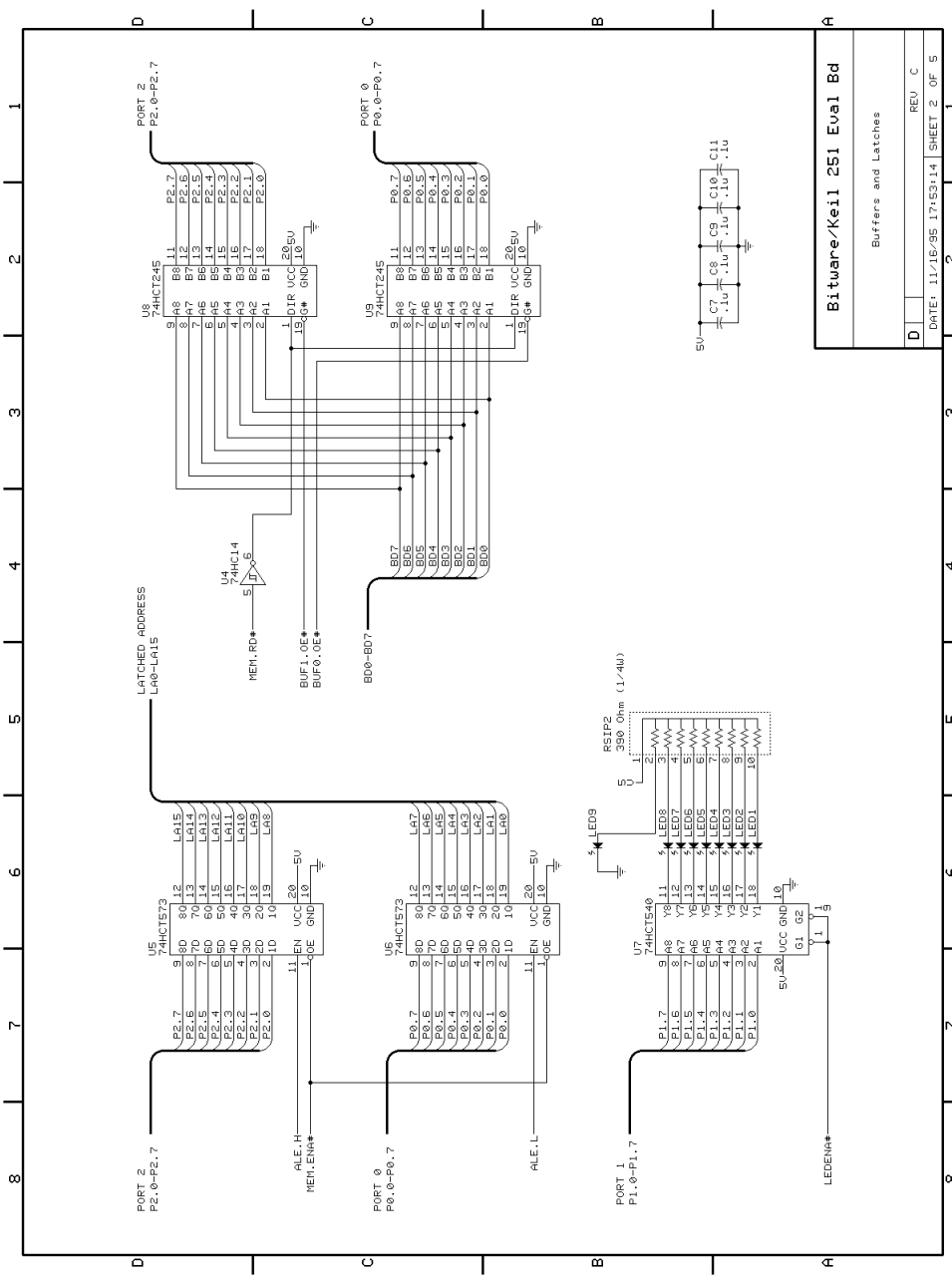
Any line starting with a semicolon is a comment. Comments may be entered after a command. For example,

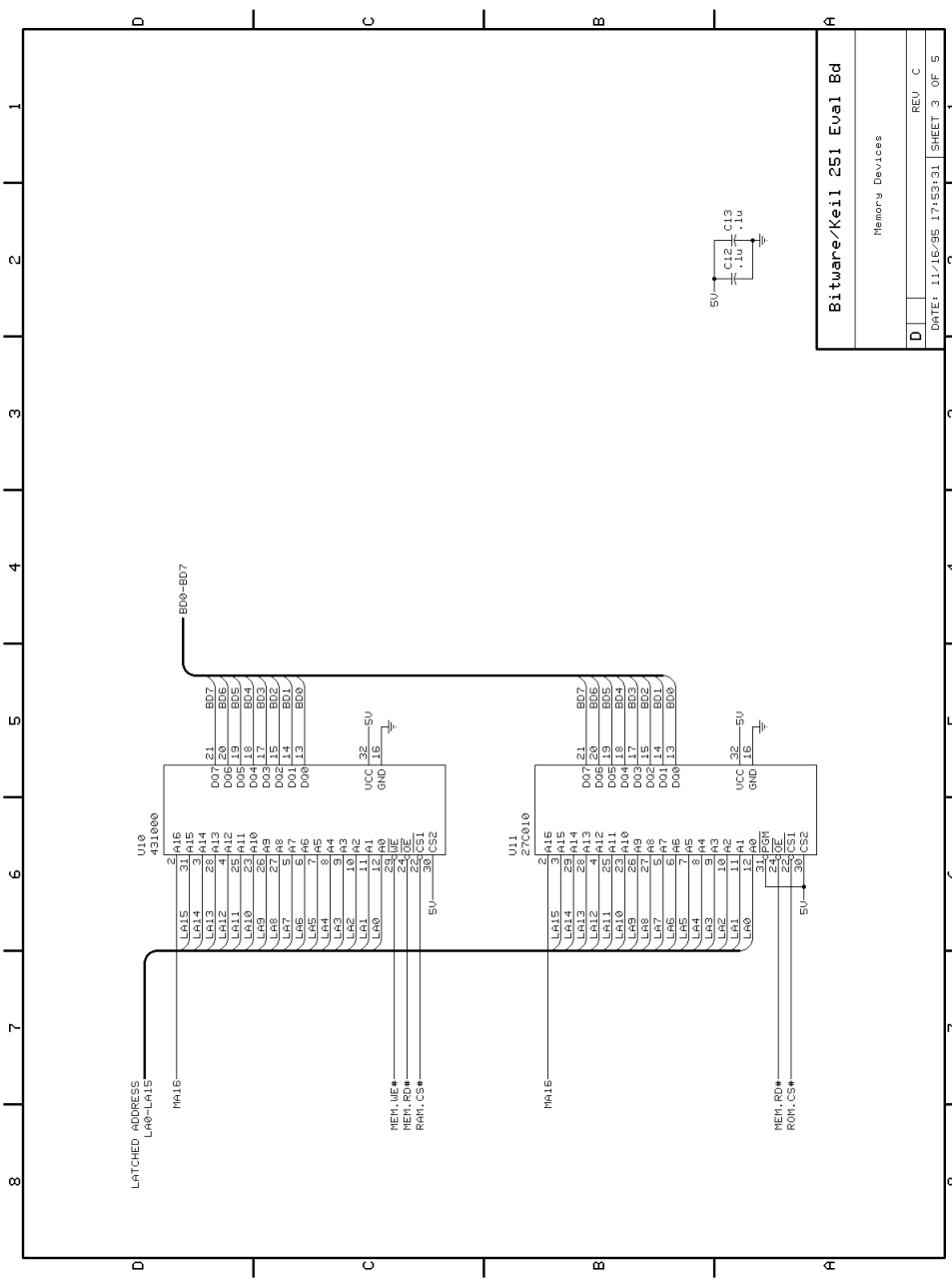
```
#D C:0x4000    ; Show code at 0xFF4000
```

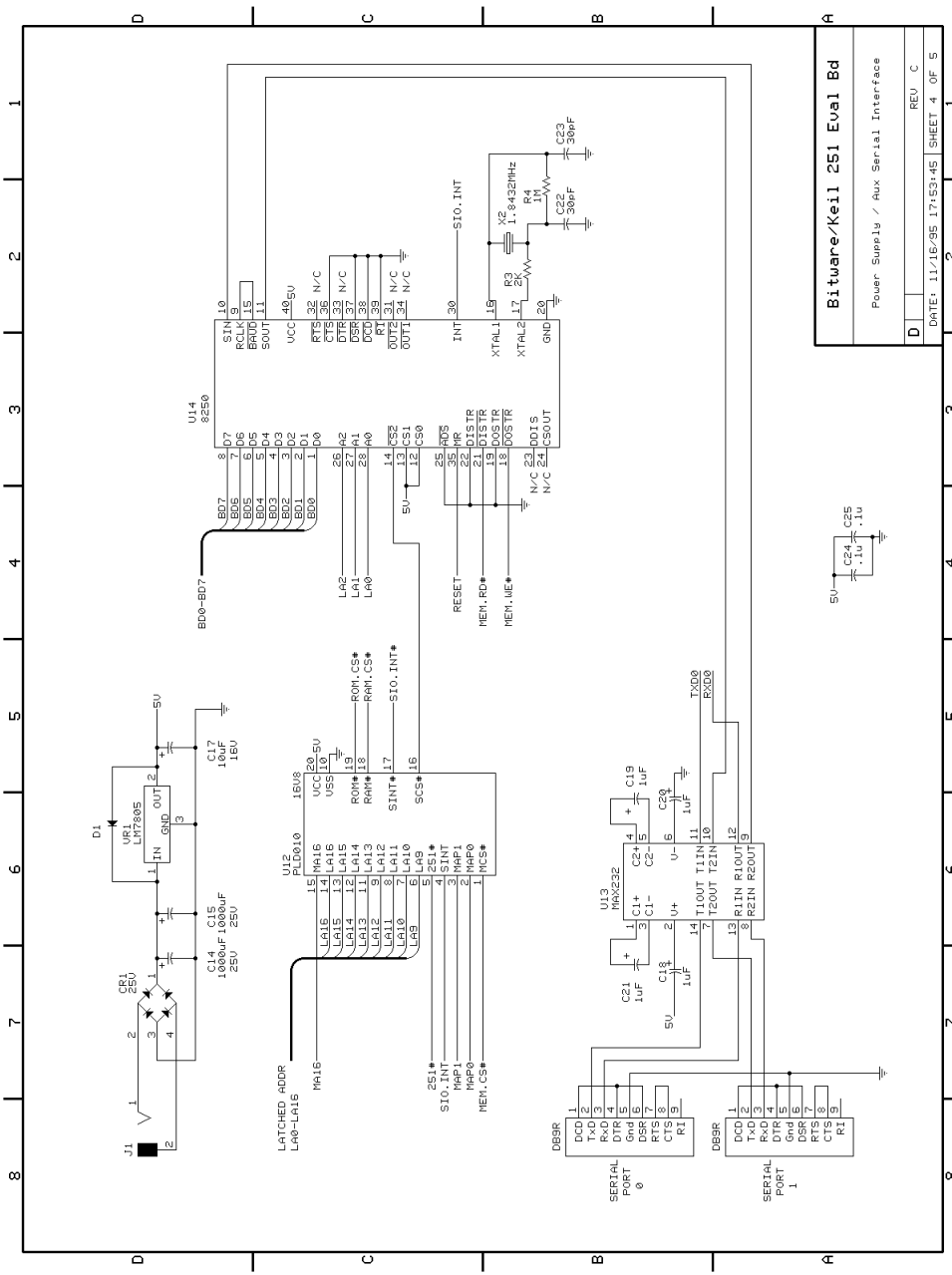
MON251 ignores any text after the semicolon. Comments are useful when you use **F2** to transmit the contents of a command file.

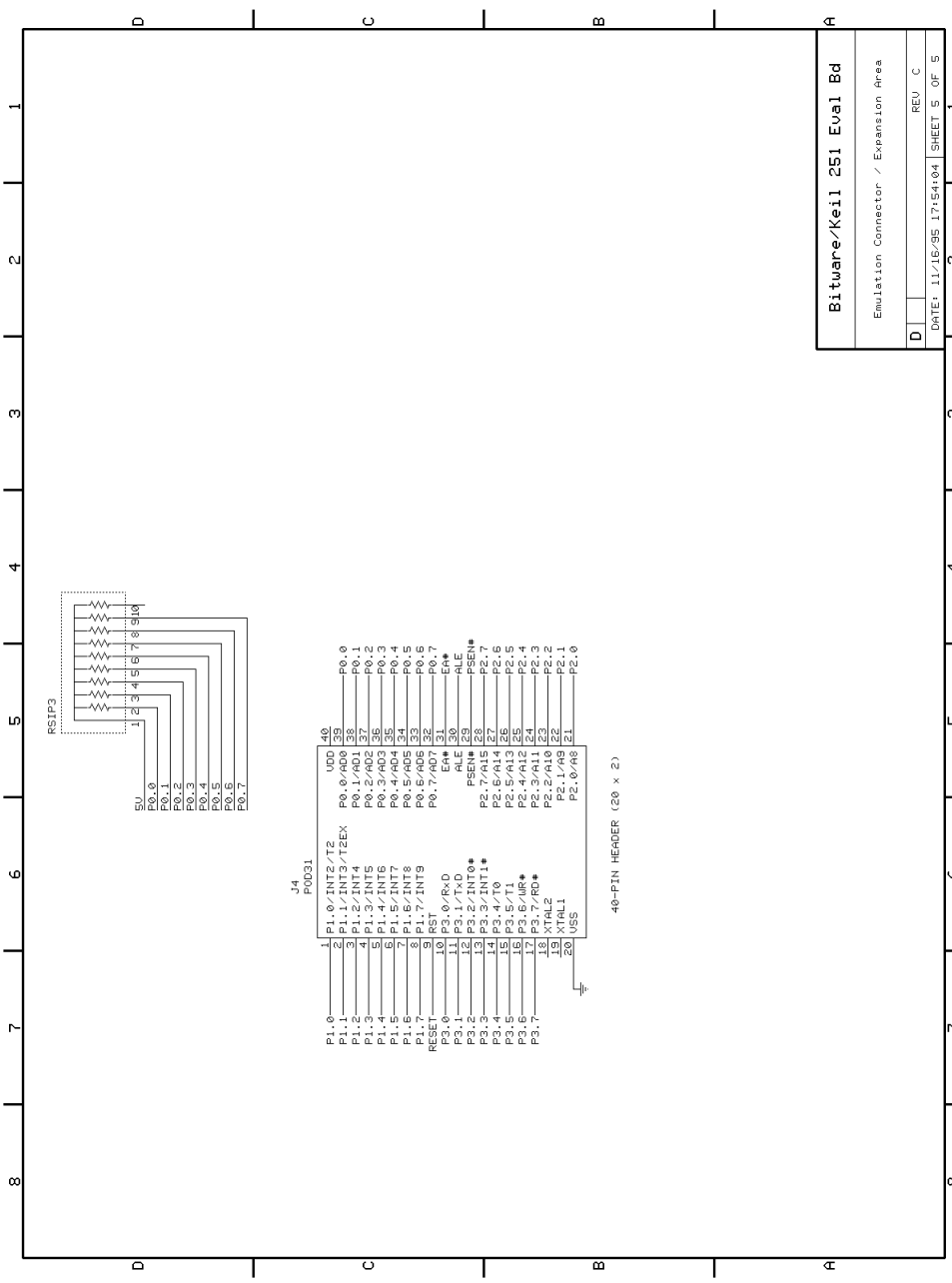
Appendix A. Schematics




Bitware/Keil 251 Eval Bd
Buffers and Latches
REV C
DATE: 11/16/95 17:53:14
SHEET 2 OF 5
PORT 2
P2.0-P2.7
PORT 0
P0.0-P0.7
LATCHED ADDRESS
LA0-LA15
U8
74HCT245
U9
74HCT245
U4
74HC14
U5
74HCT573
U6
74HCT573
U7
74HCT540
RSIP2
390 Ohm (1/4W)
LED9
MEM.RD#
BUF1.OE#
BUF0.OE#
BD0-BD7
ALE.H
MEM.ENA#
ALE.L
LEDENA#
C7
C8
C9
C10
C11
.1u
5V


Bitware/Keil 251 Eval Bd
Memory Devices
DATE: 11/16/95 17:53:31
SHEET 3 OF 5
REV C
U10
431000
U11
27C010
BD0-BD7
MA16
LATCHED ADDRESS
LA0-LA15
MEM.WE#
MEM.RD#
RAM.CS#
ROM.CS#
5V
C12 .1u
C13 .1u


Bitware/Keil 251 Eval Bd
Power Supply / Aux Serial Interface
REV C
SHEET 4 OF 5
U14 8250
U12 PLD010
U13 MAX232
VR1 LM7805
X2 1.8432MHz
SERIAL PORT 0
SERIAL PORT 1
LATCHED ADDR LA0-LA16
BD0-BD7
RESET
MEM.RD#
MEM.WE#
SIO.INT
ROM.CS#
RAM.CS#
MEM.CS#
TXD0
RXD0


Bituware/Keil 251 Eval Bd
Emulation Connector / Expansion Area
D
REV C
DATE: 11/16/95 17:54:04
SHEET 5 OF 5
RSIP3
5V
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
J4
POD31
40-PIN HEADER (20 x 2)
P1.0/INT2/T2
P1.1/INT3/T2EX
P1.2/INT4
P1.3/INT5
P1.4/INT6
P1.5/INT7
P1.6/INT8
P1.7/INT9
RST
P3.0/RxD
P3.1/TxD
P3.2/INT0#
P3.3/INT1#
P3.4/T0
P3.5/T1
P3.6/WR#
P3.7/RD#
XTAL2
XTAL1
VSS
VDD
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
EA#
ALE
PSEN#
P2.7/A15
P2.6/A14
P2.5/A13
P2.4/A12
P2.3/A11
P2.2/A10
P2.1/A9
P2.0/A8

Appendix B. PAL Equations

This appendix lists the PAL equations for the 16V8 logic devices at U2 and U12.

PLD009E (U2) Equations

```
/*-------------------------------------------------------------------------------
MCB251SB PLD009e
Copyright (c) 1995 Bitware and Rainbolt & Associates
All rights reserved.
-------------------------------------------------------------------------------*/
Name       PLD009e;
Partno     MCB251SB-PLD009e;
Date       08/16/95;
Revision   06;
Designer   Rainbolt/Ward;
Company    R&A & Bitware;
Assembly   MCB251SB-C;
Location   U2;
Device     G16V8A;

/*-------------------------------------------------------------------------------
Inputs
-------------------------------------------------------------------------------*/
Pin 1   = !wr           ;       /* WR from 251 */
Pin 2   = !rd           ;       /* RD from 251 */
Pin 3   = reset         ;       /* RST to CPU */

Pin 6   = extmem        ;       /* EXT from SW1 */
Pin 7   = !page         ;       /* PGE from SW1 */
Pin 8   = harvard       ;       /* RAM from SW1 */
Pin 9   = !psen         ;       /* PSEN from 251 */
Pin 11  = ale           ;       /* ALE from 251 */

/*-------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------*/
Pin 12  = !buf0_oe      ;       /* non-page mode data buffer */
Pin 13  = !buf1_oe      ;       /* page mode data buffer */
Pin 14  = !mem_we       ;       /* external memory write enable */
Pin 15  = !mem_cs       ;       /* external memory chip select */
Pin 16  = !mem_rd       ;       /* external memory read enable */
Pin 17  = la16          ;       /* latched A16 */
Pin 18  = ale_l         ;       /* non-page mode ale */
Pin 19  = ale_h         ;       /* page mode ale */

/*-------------------------------------------------------------------------------
Output Equations
-------------------------------------------------------------------------------*/
ale_h   = !extmem & (!page # ( page & ale));
ale_l   = !extmem & ( page # (!page & ale));

buf1_oe = !extmem &  page;
buf0_oe = !extmem & !page;
la16    = !extmem & !harvard & (!rd);

mem_we  = !extmem & !reset & wr;
mem_rd  = !extmem & (psen # (harvard & rd));
mem_cs  = !extmem & (psen # wr # (harvard & rd));
```

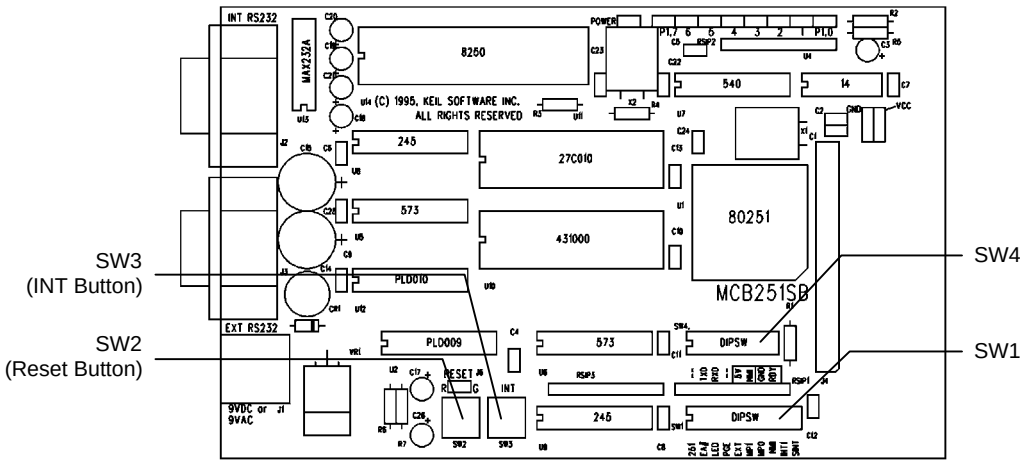
PLD010E (U12) Equations

```
/*-------------------------------------------------------------------------------
MCB251SB PLD010e
Copyright (c) 1995 Bitware and Rainbolt & Associates
All rights reserved.
-------------------------------------------------------------------------------*/
Name       PLD010e;
Partno     MCB251SB-PLD010e;
Date       08/16/95;
Revision   06;
Designer   Rainbolt/Ward;
Company    R&A & Bitware;
Assembly   MCB251SB-C;
Location   U12;
Device     G16V8A;

/*-------------------------------------------------------------------------------
Inputs
-------------------------------------------------------------------------------*/
Pin 1   = !mem_cs       ;       /* external memory chip select */
Pin 2   = !map0         ;       /* MP0 from SW1 */
Pin 3   = !map1         ;       /* MP1 from SW1 */
Pin 4   = sio_int       ;       /* INT from U14, 8250-30 */
Pin 5   = harvard       ;       /* RAM from SW1 */
Pin 6   = la9           ;       /* latched A9 */
Pin 7   = la10          ;       /* latched A10 */
Pin 8   = la11          ;       /* latched A11 */
Pin 9   = la12          ;       /* latched A12 */
Pin 11  = la13          ;       /* latched A13 */
Pin 12  = la14          ;       /* latched A14 */
Pin 13  = la15          ;       /* latched A15 */
Pin 14  = la16          ;       /* latched A16 */

/*-------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------*/
Pin 15  = ma16          ;       /* latched A16 after mapping */
Pin 16  = !sio_cs       ;       /* SIO chip select */
Pin 17  = !cpu_int      ;       /* SIO interrupt to CPU */
Pin 18  = !ram_cs       ;       /* RAM chip select */
Pin 19  = !rom_cs       ;       /* ROM chip select */

/*-------------------------------------------------------------------------------
                  251 Memory Map
-----------------------------------------------------------
MP1  MP0  RAM             ROM             SIO
-----------------------------------------------------------
 0    0   0:0000-1:FBFF   -               1:FE00-1:FFFF
 0    1   -               0:0000-1:DFFF   1:FE00-1:FFFF
 1    0   0:0000-0:FFFF   1:0000-1:DFFF   1:FE00-1:FFFF
 1    1   0:8000-0:FFFF   1:0000-1:DFFF   1:FE00-1:FFFF
-----------------------------------------------------------


                  8051 Memory Map
-----------------------------------------------------------
MP1  MP0  XDATA           CODE
-----------------------------------------------------------
 0    0   RAM 0000-FFFF   RAM 0000-FFFF
 0    1   ROM 0000-DFFF   ROM 0000-DFFF
 1    0   RAM 0000-FFFF   RAM 0000-FFFF
 1    1   RAM 8000-FFFF   RAM 8000-FFFF
-----------------------------------------------------------
-------------------------------------------------------------------------------*/
```

```
field ADD1 = [la16..la10];
field ADD2 = [la16..la9];
field ADD3 = [la16..la13];

field MAPM = [map1..map0];

B0      = ADD1:[00000..1FBFF];
B1      = ADD1:[00000..0FFFF];
B2      = ADD1:[08000..0FFFF];
B3      = ADD1:[10000..1FBFF];

B5      = ADD2:[1FE00..1FFFF];
B6      = ADD2:[0FE00..0FFFF];

B7      = ADD3:[00000..1DFFF];
B8      = ADD3:[10000..1DFFF];

/*-------------------------------------------------------------------------------
Output Equations
-------------------------------------------------------------------------------*/
cpu_int = sio_int;

sio_cs  = mem_cs & B5 & !harvard;

ram_cs  = MAPM:[0] & mem_cs & B0
        # MAPM:[2] & mem_cs & B1
        # MAPM:[3] & mem_cs & B2
        ;

rom_cs  = MAPM:[1] & mem_cs & B7
        # MAPM:[2] & mem_cs & B8
        # MAPM:[3] & mem_cs & B8
        ;

ma16    = MAPM:[0] & la16
        # MAPM:[1] & la16
        ;
```

Appendix C. DIP Switch Settings

You configure the MCB251SB using the on-board DIP switches labeled SW1 and SW4. These switches are shown in the following figure.



The default switch settings are shown in the following tables. When you receive your MCB251SB evaluation board, the DIP switches should be set as indicated.

SW1	1	2	3	4	5	6	7	8	9	10
ON	X		X	X	X				X	X
OFF		X				X	X	X		

SW4	1	2	3	4	5	6	7	8
ON		X	X		X		X	
OFF	X			X		X		X

The following sections describe each of the configuration switches.

251 (SW1-1): Default ON

The **251** switch selects whether an 8051 or a 251 microcontroller is installed in the U1 socket. If an 8051 is installed, **251** should be OFF. If a 251 is installed, **251** should be ON unless the 251 is directly emulating an 8051.

EA# (SW1-2): Default OFF

The **EA#** switch selects whether or not the CPU executes internal program code. When **EA#** switch is ON, program code stored in external ROM is executed. When **EA#** is OFF, the CPU executes program code from the internal ROM.

LED (SW1-3): Default ON

The **LED** switch selects whether or not port 1 is connected to the 8 LEDs in the upper right portion of the board. When **LED** is ON, the LEDs on port 1 are enabled. When **LED** is OFF, the LEDs are not connected to port 1.

PGE (SW1-4): Default ON

The **PGE** switch selects whether the MCB251SB board operates in page mode or non-page mode. When **PGE** is ON, the address/data multiplex/demultiplex logic for 251 page mode is enabled. When **PGE** is OFF, the logic for 251 non-page mode is enabled.

In page mode, the data and upper address byte are multiplexed on port 2 (this mode is available only on the 251). In non-page mode, the data and lower address byte are multiplexed on port 0 (this mode is available on the 251 and on the 8051). This switch should be set to coincide with the CPU configuration byte settings. Refer to your 8051 or 251 data sheets for more information about these modes.

EXT (SW1-5): Default ON

The **EXT** switch selects whether or not the external memory bus is used. If **EXT** is ON, the port 0 and port 2 are used for the address/data bus and the signals on these ports are latched using U5, U6, U8, and U9. If **EXT** is OFF, external memory addressing is not used and you cannot access the external SRAM (U10), EPROM (U11), or 16450 UART (U14).

MP1 (SW1-6): Default OFF
MP0 (SW1-7): Default OFF

The **MP1** and **MP0** switches select the memory map for the MCB251SB. The settings for these switches depend on whether you use a 251 microcontroller or an 8051 microcontroller. The settings for both processors are shown in the following tables.

251 Memory Map				
MP1 (SW1-6)	MP0 (SW1-7)	SRAM	EPROM	16450 UART
OFF	OFF	0:0000h–1:FBFFh	—	1:FE00h–1:FFFFh
OFF	ON	—	0:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	OFF	0:0000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh
ON	ON	0:8000h–0:FFFFh	1:0000h–1:DFFFh	1:FE00h–1:FFFFh

8051 Memory Map				
MP1 (SW1-6)	MP0 (SW1-7)	SRAM	EPROM	16450 UART
OFF	OFF	C:0000h–C:FFFFh X:0000h–X:FFFFh	—	—
OFF	ON	—	C:0000h–C:DFFFh X:0000h–X:DFFFh	—
ON	OFF	C:0000h–C:FFFFh X:0000h–X:FFFFh	—	—
ON	ON	C:8000h–C:FFFFh X:8000h–X:FFFFh	—	—

NMI (SW1-8): Default OFF

The **NMI** switch selects whether or not the INT push-button (SW3) is connected to the NMI pin on the 251 microcontroller. If **NMI** is on, the INT push-button is connected to the NMI pin on the 251. This switch is mutually exclusive with **INT1 (SW1-9)**. Only one of these two switches may be ON. This feature is available only on some varieties of the 251.

INT1 (SW1-9): Default ON

The **INT1** switch selects whether or not the INT push-button (SW3) is connected to port pin 3.3 on the microcontroller. If the **INT1** switch is ON, the INT push-button is connected to port 3.3. This switch is mutually exclusive with **NMI (SW1-8)**. Only one of these two switches may be ON.

SINT (SW1-10): Default ON

The **SINT** switch selects whether or not the external 16450 UART generates an interrupt on the microcontroller. If **SINT** is ON, the interrupt output from the 16450 is connected to port pin 3.2. If **SINT** is OFF, the interrupt output from the 16450 is not connected to the microcontroller. This switch is useful when using interrupt driven serial I/O with the external 16450 UART.

– – (SW4-1): Default OFF

This switch is unused at this time.

TXD (SW4-2): Default ON

The **TXD** switch controls whether or not the microprocessor's internal serial port input is connected to the MAX232A and to the internal serial port connector. When **TXD** is ON, output from port pin 3.1 is connected to the MAX232A RS-232 line driver. When **TXD** is OFF, port pin 3.1 is disconnected from the MAX232A.

RXD (SW4-3): Default ON

The **RXD** switch controls whether or not the microprocessor's internal serial port input is connected to the MAX232A and to the internal serial port connector. When **RXD** is ON, output from port pin 3.0 is connected to the MAX232A RS-232 line driver. When **RXD** is OFF, port pin 3.0 is disconnected from the MAX232A.

– – (SW4-4): Default OFF

This switch is reserved.

5V (SW4-5): Default ON

The **5V** switch provides power to pin 12 (VCC2) of the 251 microcontroller. When **5V** is ON, 5 volts are provided to pin 12 of the 251. This switch is mutually exclusive with **NMI (SW4-6)**. Only one of these two switches may be ON.

NMI (SW4-6): Default OFF

The **NMI** switch, when ON, connects pin 12 (NMI) of the 251 microcontroller to the NMI signal from SW1-8. This switch is mutually exclusive with **5V (SW4-5)**. Only one of these two switches may be ON.

GND (SW4-7): Default ON

The **GND** switch provides ground to pin 34 (VSS2) of the 251 microcontroller. When **GND** is ON, ground is provided to pin 34 of the 251. This switch is mutually exclusive with **RDY (SW4-8)**. Only one of these two switches may be ON.

RDY (SW4-8): Default OFF

The **RDY** switch provides ground to pin 34 (VSS2) of the 251 microcontroller through a 10K Ohm resistor. When **RDY** is ON, ground is provided to pin 34 of the 251 through the resistor. This switch is mutually exclusive with **GND (SW4-7)**. Only one of these two switches may be ON.

Index

V

W