KEIL SOFTWARE

MCB251 Evaluation Board

MCS®251 Microcontroller Target Board for Intel 251SB, Temic 251G1/A1, and Intel 151 and 8051 Derivatives

User's Guide 07.97

Preface

This manual describes the Keil Software MCB251 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 and provides detailed information about the DIP switches and configuration jumpers.

"Chapter 3. Hardware" provides detailed information about hardware including the schematic drawings for the MCB251 board, the logic equations for the PLD and the memory locations of the different Monitor versions contained in the EPROM.

"Chapter 4. Programming" gives details about how to use our tools to generate programs for the MCB251 evaluation board.

"Chapter 5. Using the 251 Monitor" gives a quick overview about the MON251 terminal program.

NOTE
This manual assumes that you are familiar with Microsoft Windows and the hardware and instruction set of the 8051 and 251 microcontrollers.

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** **L251.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: **`C251 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 MCB251 evaluation board and software for the MCS®251 microcontroller family. With this kit you can generate code and then operate it on the MCB251 evaluation board. This hands-on process helps you determine hardware and software needs for current and future product development.

The MCB251 evaluation board supports all operating modes of the 8051, 151, and 251 microcontrollers and lets you become familiar with the different modes of these devices. You may use a standard 8051 or the Dallas Semiconductor 80C320 or 8xC520 with this board. By generating and testing code for the various operating modes of the 8051, 151, and 251, you can evaluate code and processor performance. These factors can be weighed against other production parameters to help you 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 MCB251 evaluation board and contains the operating instructions for the monitor programs (MON251, MON51) and the terminal programs (MON251.EXE, MON51.EXE). Several different configurations of Monitor 251 and Monitor 51 are installed in the EPROM on the MCB251 board. The monitor programs let you communicate between your PC and the MCB251 evaluation board and let you download and run your 251 or 8051 programs.

The MCB251 kit includes the following items:

- MCB251 Evaluation Board User's Guide (this manual),
- MCB251 Evaluation Board,
- MCB251 Evaluation Board Software,
- 9-pin Serial Cable,
- 9 VDC, 500mA wall adapter power supply,
- and a 251 Evaluation Kit which includes a 2K compiler.

Quick Start

Use the following list to quickly locate important information about the MCB251 evaluation board.

To…	See…
Connect power to the MCB251 board.	“Using the MCB251” on page 4.
Connect the MCB251 to your PC.	“Using the MCB251” on page 4.
Read about the default configuration settings.	“Configuring the MCB251” on page 5.
Create a simple program to blink the LEDs.	“BLINKY Example Program” on page 31.
Write code to use the external UART.	“External UART Example” on page 46.
Learn more about the µVision IDE.	“Using µVision to Create the BLINKY Program” on page 32.
Learn more about the dScope debugger.	“Using dScope to Debug the BLINKY Program” on page 42.
Learn about the MON251 terminal program.	“Chapter 5. Using the 251 Monitor” on page 49.
Read about the DIP switch settings.	“DIP Switches” on page 6.
Read about the configuration jumpers.	“Configuration Jumpers” on page 9.
Configure the RAM/ROM memory.	“Monitor Memory Map” on page 27.
See the MCB251 schematics.	“Schematics” on page 18.
See the MCB251 PAL equations.	“PAL Equations” on page 23.

Chapter 2. Setup

The MCB251 evaluation 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 MCB251 Evaluation Board.
- A serial cable, 9-pin male to 9-pin female, 1-2 meters long, wired one-to-one.
- A 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 any EPROMs or other programmable devices.

Software Requirements

- The Keil MON251 terminal program (MON251.EXE) or the Keil dScope for Windows with the MON251.DLL driver. The Keil MON251 software/firmware is already programmed into the EPROM provided on the MCB251 board.
- Microsoft Windows version 3.1, 3.11, Windows 95, or Windows NT.

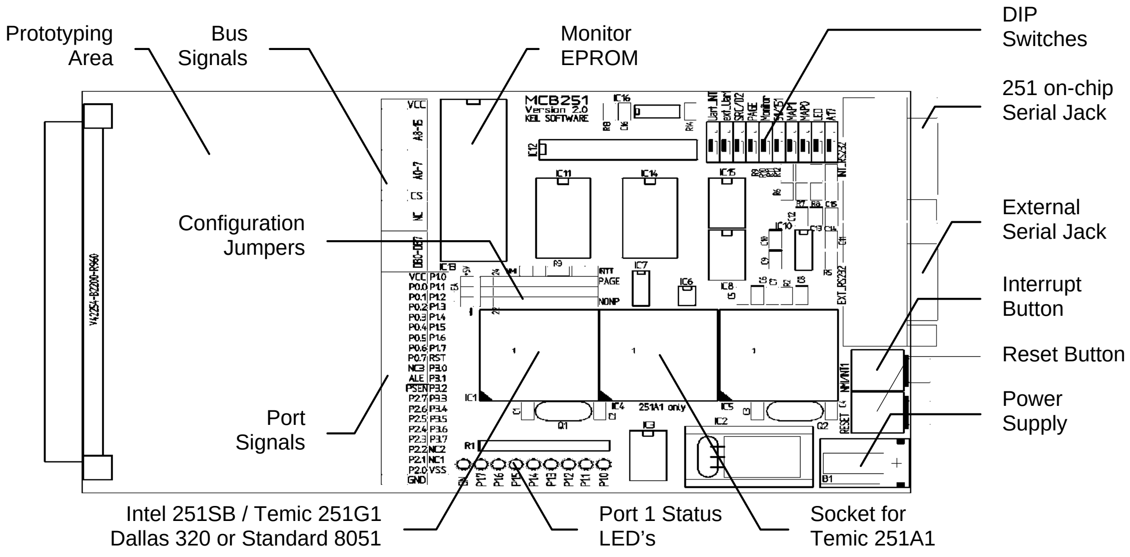
Using the MCB251

To use the MCB251 evaluation board, you must:

- Connect the external serial port jack (EXT RS232) to a serial port on your PC using the supplied serial cable.
- Connect power using a power supply provided.

The serial cable lets your PC download program code and debug your target applications. The power cable provides power to the MCB251 evaluation board. The MCB251 does not get power from the PC via the serial cable.

The following illustration shows MCB251 board and the important interface and hardware components.



Configuring the MCB251

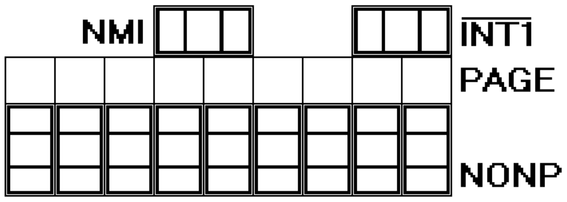
You configure the MCB251 evaluation board with the DIP switches and the configuration jumpers. The MCB251 evaluation board is shipped with the following configuration:

- Intel 87C251SB or Temic 251G1 microcontroller,
- Source Mode with 0 Wait States,
- Non-Page Mode.

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

Switch	Uart Int	ext Uart	SRC/ D2	Page	Moni tor	51/ 151	MAP 1	MAP 2	LED	A17
ON	X	X	X		X				X	X
OFF				X		X	X	X		

The default settings of the configuration jumpers are shown in the following figure. The **NONP** jumper selects the Non-Page Mode of the 251 CPU. The **INT1** and **NMI** jumpers are set to connect the Interrupt Button to the INT1 source.



DIP Switches

The following sections describe each of the DIP switches of the MCB251 board.

Uart_INT: Default ON

The **Uart_INT** switch selects whether or not the external 16550 UART generates an interrupt on the microcontroller. If **Uart_INT** is ON, the interrupt output from the 16550 is connected to port pin 3.2. If **Uart_INT** is OFF, the interrupt output from the 16550 is not connected to the microcontroller. This switch is useful when using interrupt-driven serial I/O with the external UART. You must set **Uart_INT** to ON, if you want to halt execution of a user program within dScope.

ext_Uart : Default ON

The **ext_Uart** switch determines whether or not the chip select signal for the external 16550 UART is generated. If **ext_Uart** is ON, the chip select signal is generated and the Monitor program uses the external serial interface for communication. The communication baud rate is 57600 BPS. If **ext_Uart** is OFF, the UART chip select is not generated and the Monitor program uses the on-chip serial interface for communication. In this case, the communication baud rate is 19200 BPS.

SRC/D2: Default ON

The **SRC/D2** switch selects either SOURCE mode or BINARY mode on the 251 CPU. If **SRC/D2** is ON, SOURCE mode is selected. If **SRC/D2** is OFF, BINARY mode is selected.

When you use a 151 or 8051 device, the **SRC/D2** must be OFF. When you use the Dallas 320/520 device, the **SRC/D2** must be ON to select a Monitor version which supports the Dual DPTR registers of that CPU. Refer to “CPU Selection” on page 10 for a complete description on how to select the CPU type and the related Monitor version

If you operate the MCB251 with a user application in EPROM, you must set the **SRC/DP2** switch to OFF.

PAGE: Default OFF

The **PAGE** switch selects whether or not the 251 or 151 Monitor operates in page mode or non-page mode. When **PAGE** is ON, a page mode Monitor version is selected. When **PAGE** is OFF, a non-page mode Monitor version is enabled. For correct operation of the MCB251 board, you must also set the configuration jumpers to the correct position: **PAGE** for page mode or **NONP** for non-page mode.

If you operate the MCB251 with a user application in EPROM, you must set the **PAGE** switch OFF. However, you may still use the jumpers to select page mode or non-page mode.

Monitor: Default ON

The **Monitor** switch selects if the Monitor is used for debugging or if the MCB251 operates with a user application EPROM at IC13. When **Monitor** is ON, the Monitor Memory Mapping is enabled and you must operate the MCB251 board with the supplied Monitor EPROM. When **Monitor** is OFF, the User Memory Mapping is selected and you may insert an EPROM with your target application.

If you operate the MCB251 with a user application EPROM, you must set the DIP switches **51/151**, **PAGE** and **SRC/D2** to OFF.

51/151: Default ON

The **51/151** switch selects whether a 251 or a 151/8051 compatible microcontroller is installed in the IC1 socket. If a 151/8051 compatible device is installed, **51/151** should be ON. If a 251 compatible device is installed, **51/151** should be OFF (unless the 251 is emulating an 8051 or 151 device).

A17: Default ON

The **A17** switch selects whether the 251 CPU uses P1.7 as the A17 address line or whether P1.7 is used as I/O port line. When **A17** is ON, the A17 line is used and you can address up to 256KB memory with the 251 device. When **A17** is OFF P1.7 can be used as I/O port line. Note that the CONFIG BYTES of the 251 CPU must be programmed correctly to use this feature.

The Monitor EPROM which is supplied in the MCB251 board is configured for 256KB memory. Refer to the *251 or 151 Microcontroller User's Manual* for more information.

If you use the MCB251 board with an 8051 compatible device, such as the Intel 151 or Dallas 320, the **A17** DIP switch must be OFF.

MAP1, MAP0: Default OFF, OFF

The **MAP1** and **MAP0** switches select the memory map for the MCB251. The memory map used on the MCB251 board depends on the setting of the Monitor DIP switch. The memory for both settings are shown in the following tables.

Monitor Memory Mapping (Monitor DIP Switch is ON)				
Map1	**Map0**	**RAM**	**Monitor EPROM**	**UART CS & USER CS**
OFF 251 Mode	OFF	00:0000h–01: FFFFh FE:0000h–FF: DFFFh	FF:E800h–FF:FFFFh	UART: FF:E400h–FF:E7FFh USER: FF:E000h–FF:E3FFh
OFF	ON	illegal	illegal	illegal
ON	OFF	illegal	illegal	illegal
ON 8051 Mode	ON	X:0000h–X:DFFFh C:0000h-C:DFFFh (von Neumann mapped)	C:E800h–C:FFFFh	UART: X:E400h–X:E7FFh USER: X:E000h–X:E3FFh (von Neumann mapped)

User Memory Mapping (Monitor DIP Switch is OFF) Note: DIP Switch 51/151, Page and SRC/D2 must be also OFF				
Map1	**Map0**	**RAM**	**User EPROM**	**UART CS & USER CS**
OFF 251 Mode	OFF	00:0000h–01:FFFFh FE:0000h–FE:FFFFh	FF:0000h–FF:DFFFh FF:E800h–FF:FFFFh	UART: FF:E400h–FF:E7FFh USER: FF:E000h–FF:E3FFh
OFF 251 Mode	ON	00:0000h–01:FFFFh	FE:0000h–FF:DFFFh FF:E800h–FF:FFFFh	UART: FF:E400h–FF:E7FFh USER: FF:E000h–FF:E3FFh
ON 251 Mode	OFF	00:0000h–00:7FFFh	00:8000h–01:FFFFh FE:0000h–FF:DFFFh FF:E800h–FF:FFFFh	UART: FF:E400h–FF:E7FFh USER: FF:E000h–FF:E3FFh
ON 8051 Mode	ON	X:0000h–X:DFFFh	C:0000h–C:FFFFh	UART: X:E400h–X:E7FFh USER: X:E000h–X:E3FFh

LED: 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. If the **Monitor** DIP switch ON and **LED** is OFF, the Monitor does not modify the port 1 lines on power-up.

Configuration Jumpers

The following sections describe each of the configuration jumpers of the MCB251 board.

NMI: Default ON

The **NMI** configuration jumper selects whether or not the INT push-button is connected to the NMI pin on the Temic 251G1 microcontroller. If **NMI** is OFF, the INT push-button is connected to the NMI pin on the Temic 251G1. If **NMI** is ON, pin 34 is connected to GND.

INT1: Default ON

The **INT1** configuration jumper selects whether or not the INT push-button 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. If **INT1** is OFF, the INT push-button is not connected to port 3.3.

PAGE / NONP: Default NONP

These nine configuration jumpers select whether the MCB251 board operates in page mode or non-page mode. When these jumpers are in the **PAGE** position, the address/data bus is configured for 251/151 page mode. When the jumpers are in the **NONP** position, the standard 8051 bus or 251/151 non-page mode is selected.

In page mode the data and upper address byte are multiplexed on port 2 (this mode is available only on the 251/151). In non-page mode the data and lower address byte are multiplexed on port 0 (this mode is available on the 251, 151, and 8051). This jumper setting should be set to coincide with the CPU CONFIG BYTE settings. Refer to the *251 or 151 Microcontroller User's Manual* for more information about these modes.

CPU Selection

The MCB251 board can operate with various CPU types. The following table gives you an overview of the various CPU operating modes. It lists values for the configuration bytes of the 251 / 151 CPU used by the Monitor program. For more information about the configuration registers, refer to the *251 or 151 Microcontroller User's Manual*.

NOTE
*The settings in the following table are only relevant only for the Monitor EPROM (**Monitor** DIP switch ON). When the **Monitor** DIP switch is OFF, you may select the Memory Mapping and the CPU type with the **MAP0** and **MAP1** DIP switches.*

CPU Type	DIP Switch Setting					CPU Config Bytes	
and Operating Mode	Page	Src/D2	Map0	Map1	A17	Config0	Config1
251SB, 251G1 or 251A1 Page Mode, Source Mode	ON	ON	OFF	OFF	ON	0F1h	097h
251SB, 251G1 or 251A1 Page Mode, Binary Mode	ON	OFF	OFF	OFF	ON	0F0h	097h
251SB, 251G1 or 251A1 Non-page Mode, Source Mode	OFF	ON	OFF	OFF	ON	0F3h	097h
251SB, 251G1 or 251A1 Non-page Mode, Binary Mode	OFF	OFF	OFF	OFF	ON	0F2h	097h
Intel 151, Page Mode	ON	OFF	ON	ON	OFF	0FCh	0E7h
Intel 151, Non-page Mode	OFF	OFF	ON	ON	OFF	0FEh	0E7h
Standard 8051 Derivatives	OFF	OFF	ON	ON	OFF	N/A	N/A
Dallas 320/520	OFF	ON	ON	ON	OFF	N/A	N/A

NOTE
*The Temic 251A1 and 251G1 are currently available only as A-step devices. The operating mode of this device is programmed inside the CPU and cannot be changed in an external EPROM. If you use one of these devices you must set the **PAGE** DIP switch OFF and the **SRC/D2** DIP switch ON. This restriction does not apply when the 251A1 and 251G1 are available in C-step technology.*

The Intel 251SB CPU may be used to emulate the Intel 151 CPU. This lets you test 151 applications without replacing the 251SB CPU.

Monitor Modes

The MCB251 board comes with a Monitor EPROM which contains several different Monitor configurations. Most of the Monitor configurations are required to run the different CPU types on the board. The following Monitor settings are independent of the CPU type used in the MCB251 board.

DIP Switch	Monitor Mode
ext_Uart	**ON**: use **EXT RS232** interface @ 57600 BPS. for Monitor communication. **OFF**: use **INT RS232** interface @ 19200 BPS. for Monitor communication.
LED	**ON**: display Monitor status on Port 1 LED’s (see below). **OFF**: Monitor status is not displayed and Port 1 is not affected.
Monitor	**ON**: use Monitor program for debugging **OFF**: User EPROM as IC13. Refer to “MAP1 and MAP0 DIP Switch” on page 7 to for the User EPROM Memory Map.

NOTE
Some derivatives of the 8051 and 251 microcontroller family do not have a compatible serial interface or baud rate generator for the internal serial interface. For such devices it is impossible to use the ***INT RS232*** *interface. Currently, the Temic 251A1 and standard 8051 CPU are not compatible. With these devices you can only use the Monitor with the EXT RS232 interface—the* ***ext_Uart*** *DIP switch must be* ***ON****.*

Monitor Status Display

During the reset phase the Monitor program displays status information on the Port 1 LEDs. This status information reflects the board configuration and signals that the board is working correctly. The status byte is shifted to the left, which flashes LED’s during the reset phase. The following table lists the first phase of the LED pattern.

Monitor Version	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Monitor 251 and 251 CPU	**A17**	**OFF**	**OFF**	**OFF**	**OFF**	**OFF**	**OFF**	**ON†**
Monitor 51 for 151 or 8051 CPU	**–**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON**	**ON†**
Monitor 51 for Dallas 320/520	**–**	**ON**	**OFF**	**ON**	**OFF**	**ON**	**OFF**	**ON†**

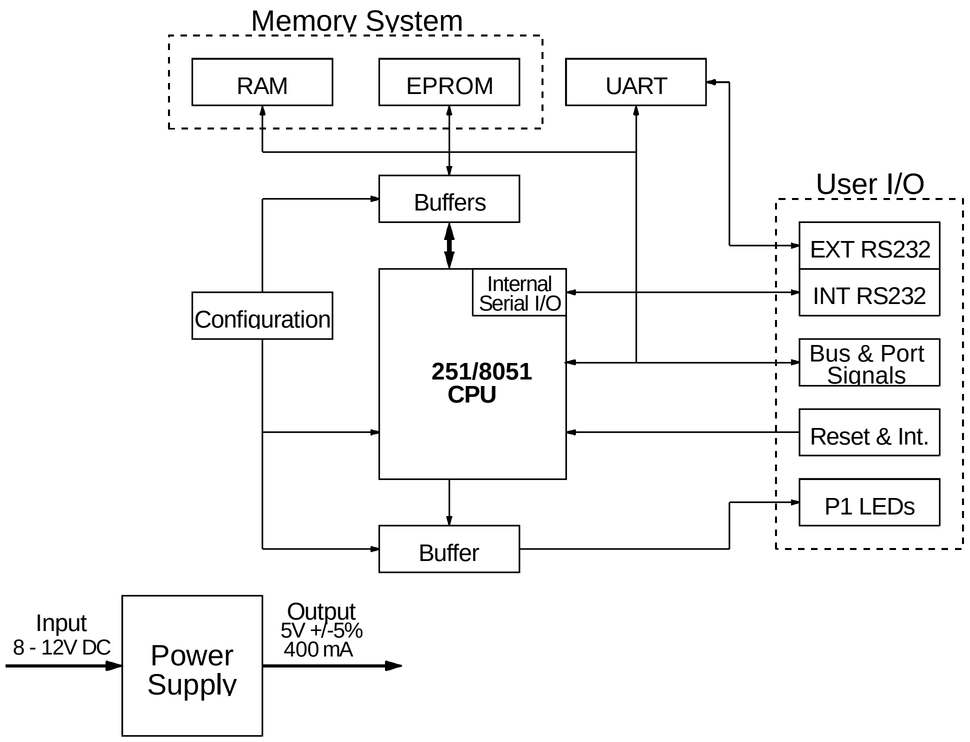
† The P1.0 status LED is ON when the INT RS232 interface is used for the Monitor/PC interface and OFF when the EXT RS232 interface is used for the Monitor/PC interface.

Chapter 3. Hardware

The MCB251 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 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 MCB251 and also provides a circuit description. The descriptions here will help you understand how the MCB251 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.



Power Supply

Power is supplied to the MCB251 from an external 8-12 Volt 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 center hole provides positive voltage. 5 Volts DC is generated and provided to the board by a 7805 voltage regulator at IC2. To reduce the noise for the 251A1 A/D converter a second 78L05 voltage regulator is provided at IC6.

251/8051 CPU

The Intel 251SB or Temic 251G1 provided with the MCB251 are 251 derivatives with 1 Kbytes of internal RAM. The CPU is located at IC1. The IC1 socket can accommodate a Dallas 320 or 520 CPU, an Intel 151, or a standard 8051 microcontroller.

You may use the Temic 251A1 CPU in the IC4 socket. However, you must remove the CPU in socket IC1 when you use the 251A1.

A 12.000 MHz crystal provides the clock signal for the CPU.

NOTE
You may insert only one CPU into the board at a time. It is not possible to operate with a Temic 251A1 in socket IC4 and another 251/8051 CPU in socket IC1 simultaneously.

Configuration

The MCB251 is a very flexible evaluation board. You can change the operation of the board using the DIP switches and the configuration jumpers. Features such as CPU type, LEDs, page mode, memory map, push button inputs, external UART interrupts, and on-chip serial port can all be configured using these switches.

NOTE
You must RESET the MCB251 after changing the state of any DIP switch.

Refer to “DIP Switches” on page 6 and “Configuration Jumpers” on page 9 for a complete description of the DIP switches and configuration jumpers.

Buffers

The MCB251 provides complete support for 251 page mode and non-page mode. To do this, the MCB251 includes 2 latches for the data/address bus.

In non-page mode, the 251/151 multiplexes the address bus (A0-A7) and data bus (D0-D7) on port 0. This is the same way the standard 8051 works. To support non-page mode, the configuration jumpers must be in the **NONP** position. In non-page mode, the address bus (port 0) is latched using the 74HC373 at IC8.

In page mode, the 251/151 multiplexes the address bus (A8-A15) and data bus (D0-D7) on port 2. To support page mode, the configuration jumpers must be in the **PAGE** position. In page mode, the address bus (port 2) is latched using the 74HC373 at IC15.

NOTE
*The CPU configuration bytes must be set according the board configuration for page or non-page mode. Refer to the 251/151 data book for more information about the CPU configuration bytes. The Monitor EPROM contains pre-configured Monitor versions for page and non-page mode. These are selected with the **PAGE** DIP switch. To select a Monitor version for PAGE mode, the **PAGE** DIP switch must be ON.*

Decode Logic

All memory address decode logic and other signal conversions are performed by the 20V8 PAL at IC12. Refer to " PAL Equations" on page 23 for a complete listing of the PAL equations used.

Memory and I/O Devices

The MCB251 maps three memory devices into the address space of the CPU: the RAM at IC14, the RAM at IC15, and the EPROM at IC13. The 20V8 PAL at IC12 provides the chip select signal for the external UART at IC5 and a user chip select signal.

The MCB251 board comes with a Monitor EPROM at IC13 which can support up to 16 different Monitor versions. You may replace the Monitor EPROM at IC14 with a 27C1001 or 27C2001 for your target program. If you insert an EPROM with a user application program the DIP switches **PAGE**, **SCR/D2**, **51/151**, and **Monitor** must be set to OFF. If the **Monitor** DIP switch is OFF, you may select different memory maps for the user application using the **MAP0** and **MAP1** DIP switches. For more information refer to the MAP1 and MAP0 DIP switch descriptions on page 7.

Status LEDs

The MCB251 has a single power LED labeled ON which indicates the power to the board is on. Eight LEDs are optionally connected to the Port 1 outputs through a 74HC373 at IC3. The Port 1 LEDs are flashed during reset and display the configuration status of the Monitor EPROM. For more information refer to “Monitor Status Display” on page 11. You may disable the LED driver by setting the **LED** DIP switch OFF. The this DIP switch is OFF, the Monitor does not affect the Port 1 Pins during the reset phase.

Push Buttons

The MCB251 provides two push buttons. The first push button, labeled RESET (S1), is connected to the reset input of the CPU. The second button, labeled INT (S2), is connected to the INT1 (port 3.3) of the CPU when the configuration jumper **INT1** (J5) is set. When the configuration jumper **NMI** is not set, you may generate an NMI signal to the Temic 251G1 device.

Serial Port

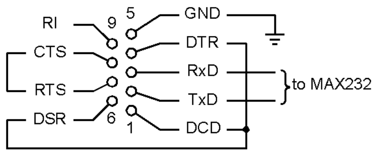
The MCB251 supports both the on-chip serial port and an external UART. Both use the MAX232, at IC10, to convert TTL to RS-232 voltage levels.

You can configure the Monitor to use either the on-chip serial port or the external UART for the communication with the PC. When the **ext_Uart** DIP switch is ON, the external UART is used at 57600 BPS for the Monitor communication. Otherwise the on-chip serial port is used at 19200 BPS for the Monitor communication.

The internal serial port is derived from the internal serial functions of the 8051/251 (P3.0/RXD and P3.1/TXD).

The external UART (IC5) is implemented using an 8250, 16450, or 16550 device. This is the same chip that is used in most PCs. A 1.8432 MHz oscillator 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 **Uart_INT** DIP switch is ON, the UART interrupt is connected to the INT0 CPU pin.

Both serial ports are configured as a standard 3-wire interface. 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.

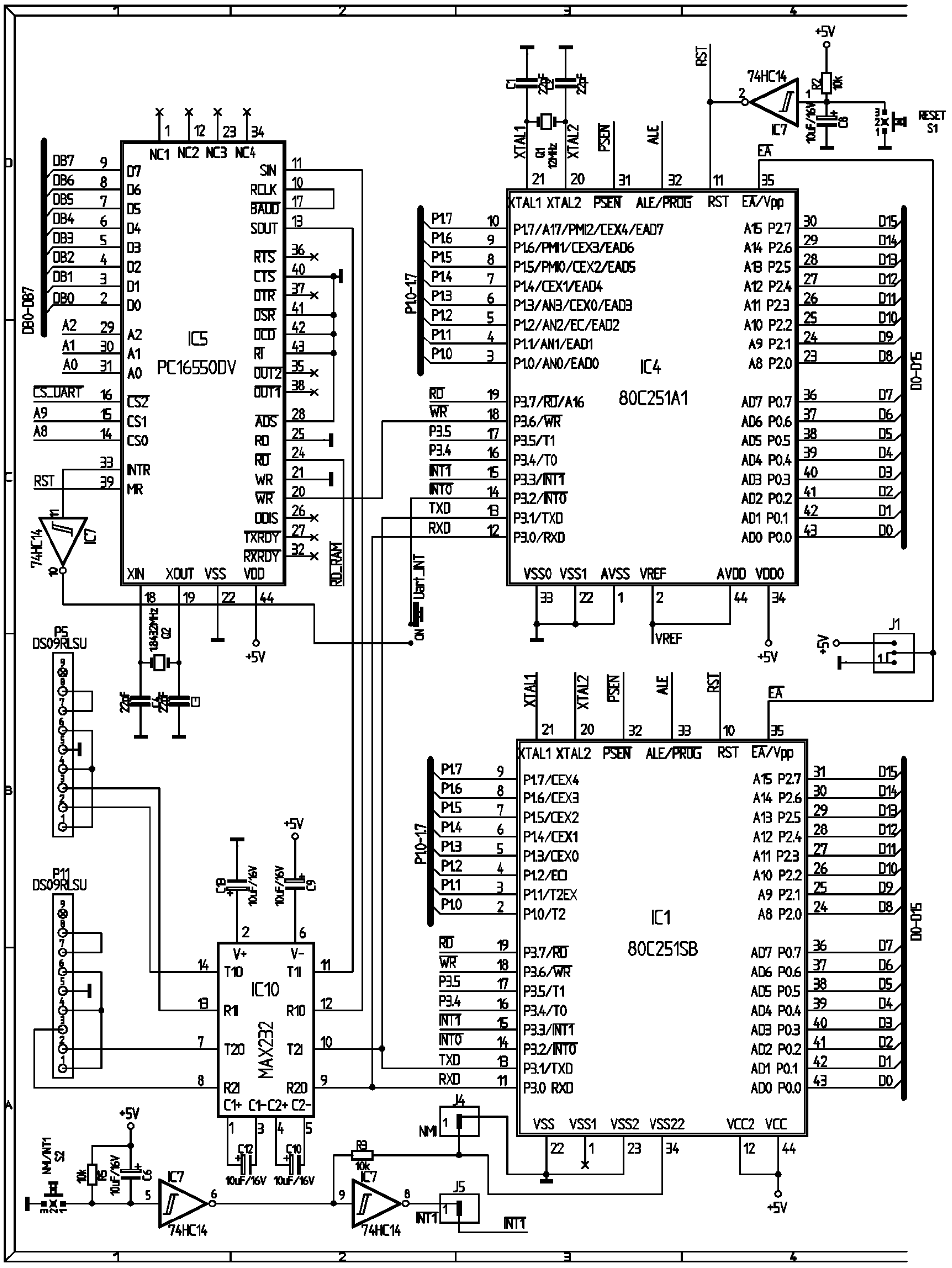


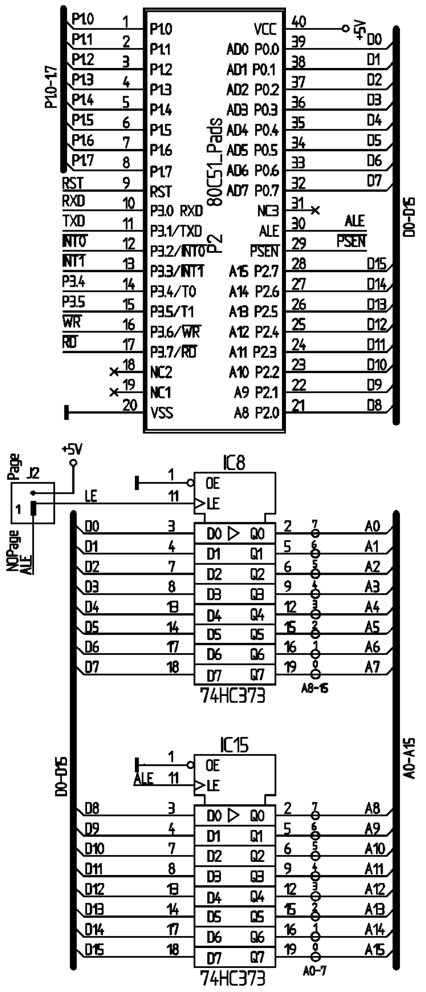
Prototyping Area

A perforated area is provided on the MCB251 for prototyping your own hardware. All CPU signals necessary to create a 40-pin DIP plug for emulating an 8051 or 251 in a 40-pin DIP package are connected to this prototyping area. In addition, the data and address bus signals and the user chip select signal are connected to this area. The signals provided are driven directly by the CPU. Exercise caution to avoid overloading these signal lines. Refer to "DIP Switches" on page 6 for the pin-out of the CPU and bus signals.

NOTE
Since the MCB251 supports both the page mode and non-page mode you should connect memory-mapped devices directly to the data bus DB0-DB7 and A0-A15.

Schematics


IC5
PC16550DV
IC4
80C251A1
IC1
80C251SB
IC10
MAX232
IC7
74HC14
RESET S1
P5 DS09RLSU
P11 DS09RLSU
J1
J4
J5
NMI/INT1 S2
DB0-DB7
D0-D15
P10-P17
1.8432MHz Q2
12MHz Q1
+5V
VREF
RD_RAM
Uart_INT



Monitor Modes

ext. Uart	ON = use ext. RS232 @ 57600bps
	OFF = use int. RS232 @ 19200bps

LED	ON = LED’s display Port 1 Status
	OFF = LED’s off

Monitor	ON = use Monitor for Debugging
	OFF = User EPROM see table Memory Map

Monitor Selection (Monitor = ON)

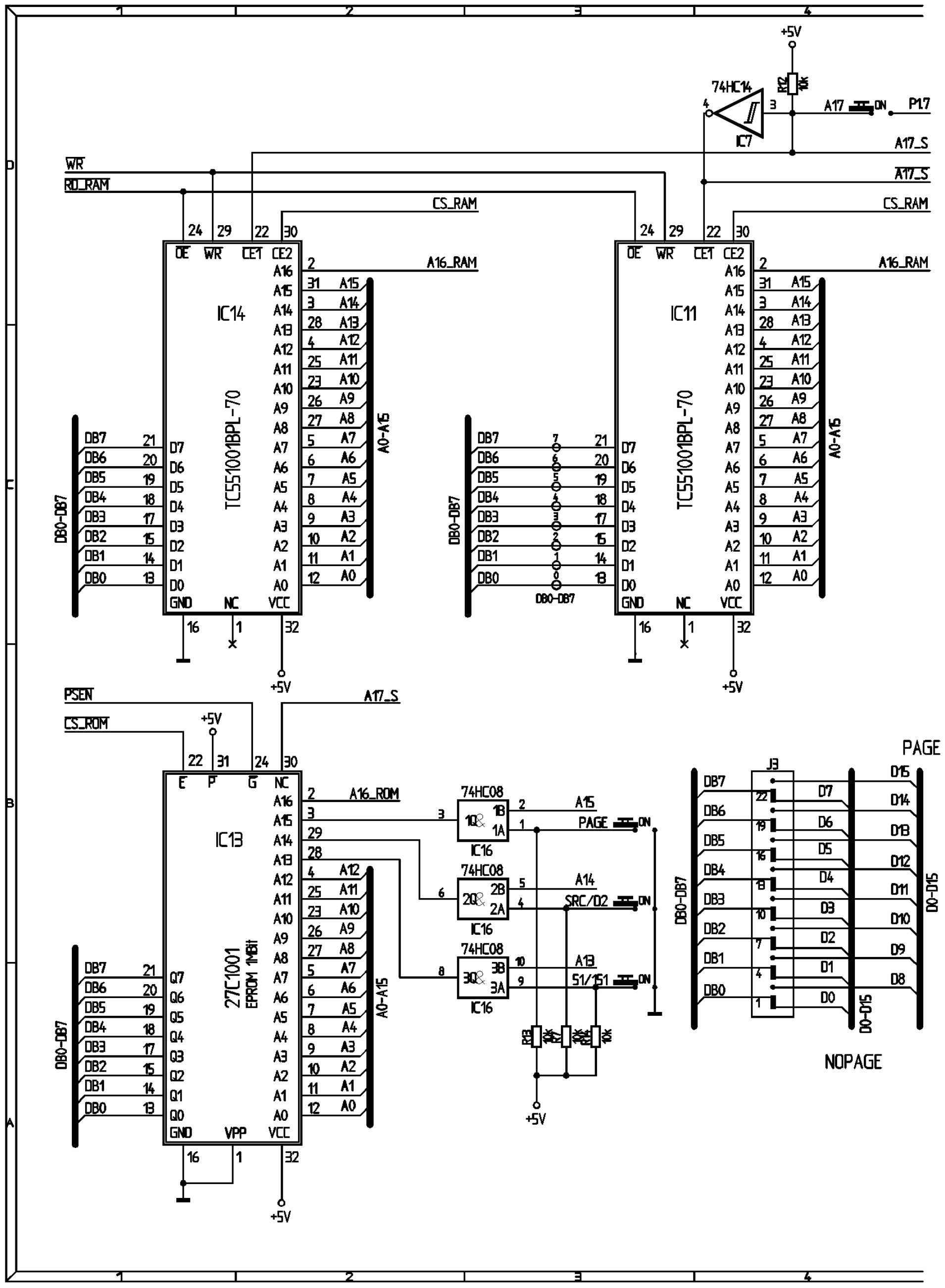
CPU	51/151	Page	SRC/DP2	MAP0	MAP1
251	OFF	ON=Page Mode OFF=8051 Bus	ON=Source Mode OFF=Binary Mode	OFF	OFF
151	ON		OFF	ON	ON
8051 Derivates	ON	OFF	OFF	ON	ON
80320	ON	OFF	ON	ON	ON

Memory Map for User EPROM (Monitor = OFF)

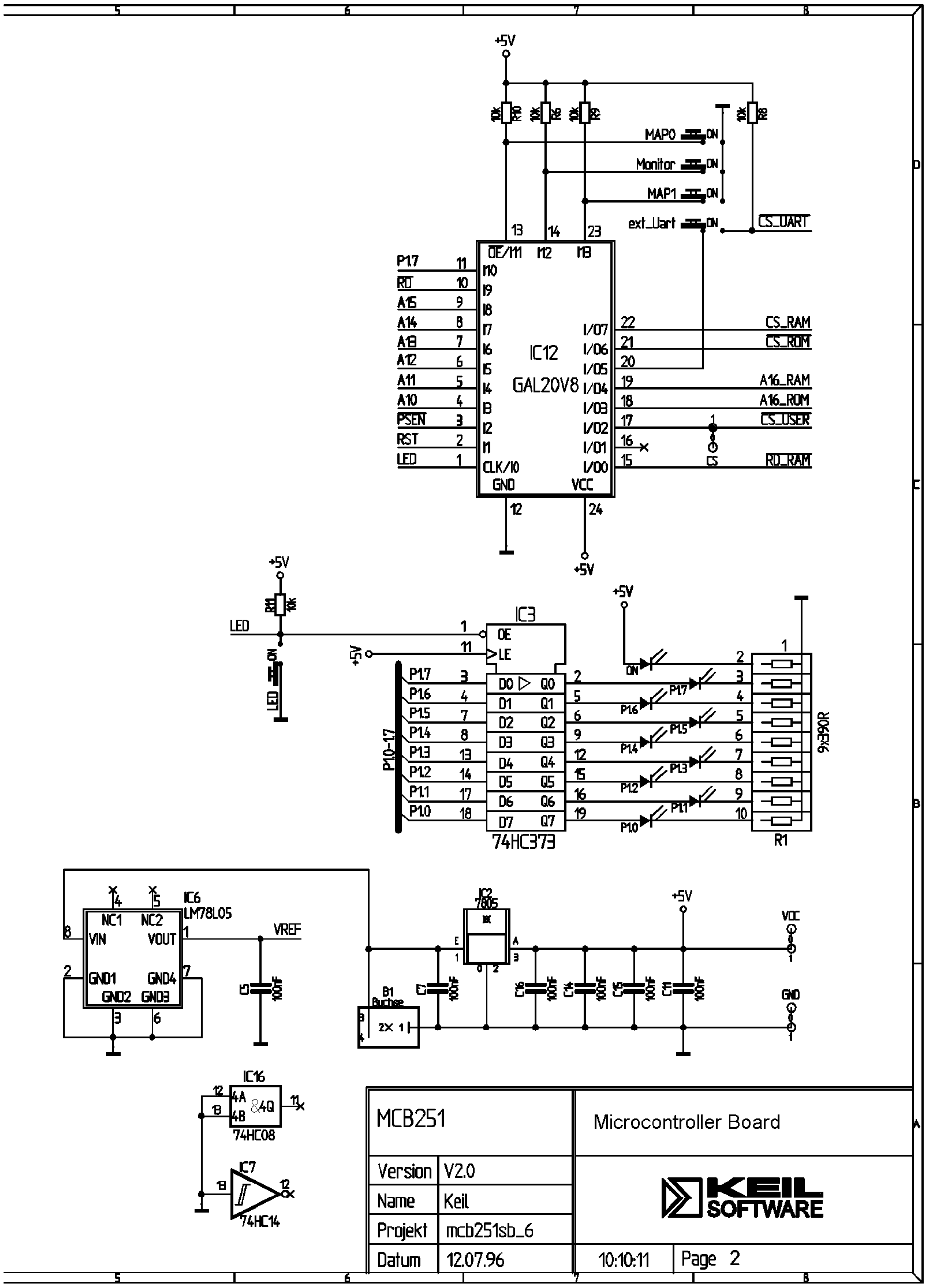
51/151=OFF Page=OFF SRC/DP2=OFF			
MAP0	MAP1	ROM	RAM
OFF	OFF	64kB FF0000...FFFFFF (*)	192kB 0...02FFFF
ON	OFF	128kB FE0000...FFFFFF (*)	128kB 0...01FFFF
OFF	ON	224kB 008000...00FFFF FD0000...FFFFFF (*)	32kB 0...007FFF
ON	ON	8051 Mode 64kB C:0...C:FFFF	8051 Mode 64kB X:0...X:DFFF (*)
* Memory Map excludes USER & UART areas			
USER CS		FFE000...FFE3FF	X:E000...X:E3FF
UART CS		FFE400...FFE7FF	X:E400...X:E7FF

MCB251	
Version	V2.0
Name	Keil
Projekt	mcb251sb_6
Datum	12.07.96

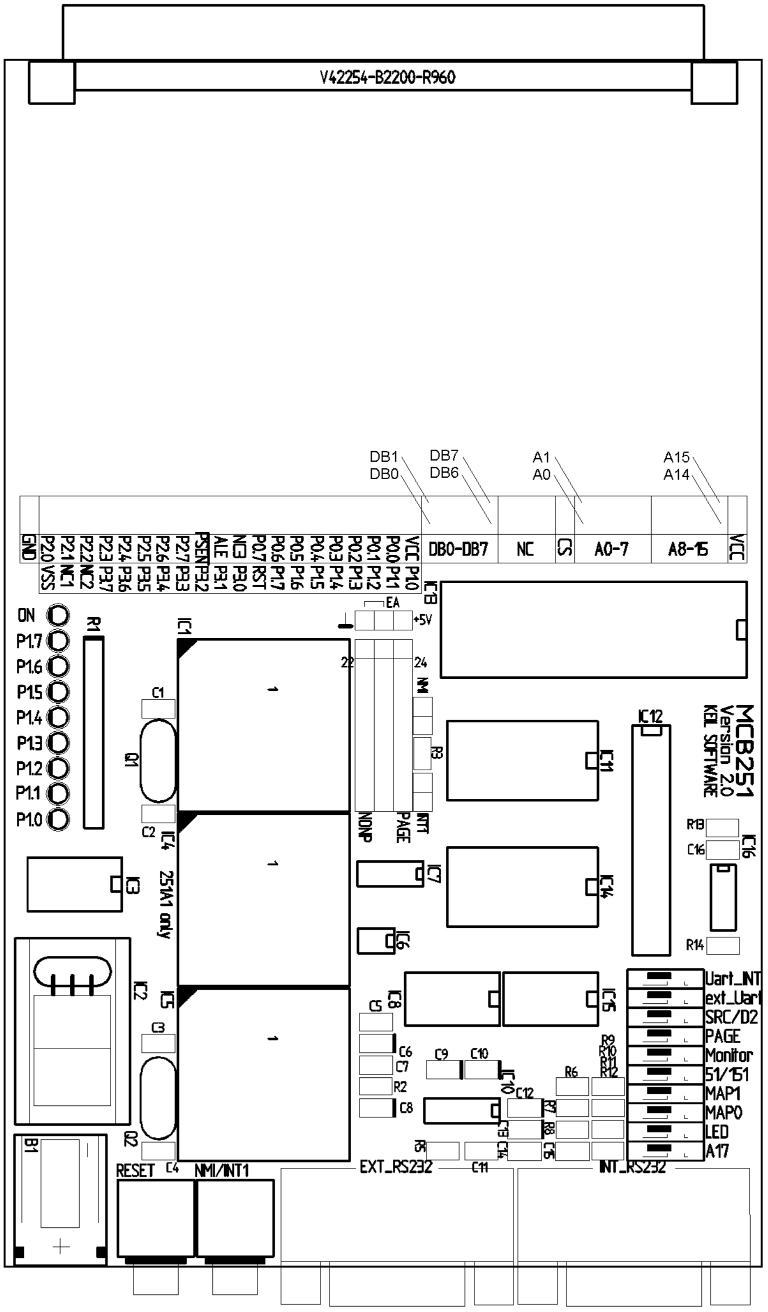
10:10:11 Page 1



WR
RD_RAM
CS_RAM
A16_RAM
IC14
TC551001BPL-70
IC11
74HC14
IC7
R12 10k
+5V
A17
P17
A17_S
DB0-DB7
A0-A15
PSEN
CS_ROM
A16_ROM
IC13
27C1001 EPROM 1MBit
74HC08
IC16
A15
PAGE
A14
SRC/D2
A13
51/151
R13
J3
PAGE
NOPAGE
D0-D15


+5V
MAP0
Monitor
MAP1
ext_Uart
CS_UART
IC12
GAL20V8
P1.7
RD
A15
A14
A13
A12
A11
A10
PSEN
RST
LED
CS_RAM
CS_ROM
A16_RAM
A16_ROM
CS_USER
RD_RAM
IC3
74HC373
P1.0-1.7
R1
9x390R
IC6
LM78L05
VREF
IC2
7805
B1
Buchse
VCC
GND
IC16
74HC08
IC7
74HC14
MCB251
Microcontroller Board
Version V2.0
Name Keil
Projekt mcb251sb_6
Datum 12.07.96
10:10:11
Page 2
KEIL SOFTWARE

Printed Board Assembly



Technical Data

Supply Voltage:	8V-12V DC
Supply Current:	typical: 300mA
System Clock:	12 MHz
Memory:	256 Kbytes RAM 128 Kbytes Monitor EPROM (optional: 256 Kbytes User EPROM)
CPU:	Intel 80C251SB or Temic 80C251G1 (optional: Dallas 80C320, Intel 80C151, Intel 80C51FC, or compatible device)
Peripherals:	2 × RS232 Interfaces

PAL Equations

This following lists the PAL equations for the 20V8 logic device at IC12.

```
;PALASM Design Description

;---------------------------------- Declaration Segment ------------
TITLE    GAL for MCB251SB Prototype board
PATTERN  Decoder and Boot Logic
REVISION 1.0
AUTHOR   Hans Schneebauer
COMPANY  Keil Elektonik GmbH
DATE     10/06/96

CHIP  IC6  PALCE20V8

;---------------------------------- PIN Declarations ---------------
PIN  1          LED                 COMBINATORIAL ; INPUT
PIN  2          RST                 COMBINATORIAL ; INPUT
PIN  3          /PSEN               COMBINATORIAL ; INPUT
PIN  4..11      A[10..17]           COMBINATORIAL ; INPUT
PIN  12         GND                               ; INPUT
PIN  13         /MAP[0]             COMBINATORIAL ; INPUT
PIN  14         /MON_ON             COMBINATORIAL ; INPUT
PIN  15         /RD_RAM             COMBINATORIAL ; OUTPUT
PIN  16         MON_RUN             COMBINATORIAL ; OUTPUT
PIN  17         /CS_USER            COMBINATORIAL ; OUTPUT
PIN  18         A16_ROM             COMBINATORIAL ; OUTPUT
PIN  19         A16_RAM             COMBINATORIAL ; OUTPUT
PIN  20         /CS_UART            COMBINATORIAL ; OUTPUT
PIN  21         /CS_ROM             COMBINATORIAL ; OUTPUT
PIN  22         CS_RAM              COMBINATORIAL ; OUTPUT
PIN  23         /MAP[1]             COMBINATORIAL ; INPUT
PIN  24         VCC                               ; INPUT
```

```
;---------------------------------- Boolean Equation Segment ------
EQUATIONS

IF (MON_ON) THEN                    ; Monitor switched on
  BEGIN
    A16_ROM = LED
    IF (MAP[1..0] = #b11) THEN
      BEGIN
        RD_RAM  = PSEN + /A[16]      ; 8051 Mode
        A16_RAM = 0;
      END
    ELSE
      BEGIN
        RD_RAM  = PSEN
        A16_RAM = A[16]
      END
    IF (MON_RUN) THEN
      BEGIN
        MON_RUN = /RST
        IF (MAP[1..0] = #b11) THEN            ; 8051 Mode ?
          BEGIN
            CS_RAM = /(A[15] * A[14] * A[13]) ; Address 0 - dfff
          END
        ELSE
          BEGIN
            CS_RAM = /(A[17] * A[16] *  A[15] * A[14] * A[13]) ; Address 0 - 3dfff
          END

        CS_ROM =  A[17] *  A[16] *  A[15] *  A[14] * ; Address 3e800 - 3efff
                  A[13] * /A[12] *  A[11] +
                  A[17] *  A[16] *  A[15] *  A[14] * ; Address 3f000 - 3ffff
                  A[13] *  A[12]
      END
    ELSE
      BEGIN
        CS_ROM = 1                            ; ROM always enabled
        CS_RAM = 0                            ; RAM always disabled
        IF (A[17..10] = #b11111010) THEN      ; Address 3e800 - 3ebff
          BEGIN
            MON_RUN = PSEN * /RST
          END
        ELSE
          BEGIN
            MON_RUN = 0
          END
      END
  END
ELSE                                ; Monitor switched off
  BEGIN
    CASE (MAP[1..0])
      BEGIN
      0: BEGIN
         CS_RAM  =/(A[17] * A[16])                         ; RAM 00000 - 2ffff
         CS_ROM  =  A[17] * A[16] * /CS_UART * /CS_USER    ; ROM 30000 - 3dfff
         A16_ROM =  A[16]
         A16_RAM =  A[16]
         RD_RAM  =  PSEN
         END
      1: BEGIN
         CS_RAM  = /A[17]                          ; RAM 00000 - 1ffff
         CS_ROM  =  A[17] * /CS_UART * /CS_USER     ; ROM 20000 - 3dfff
         A16_ROM =  A[16]
         A16_RAM =  A[16]
         RD_RAM  =  PSEN
         END
      2: BEGIN
         CS_RAM  = /A[17] * /A[16] * /A[15]         ; RAM 0000 - 07FFF
         CS_ROM  = /CS_RAM * /CS_UART * /CS_USER    ; ROM 00000 - 3ffff
         A16_ROM = A[16]
         A16_RAM = A[16]
         RD_RAM  = PSEN
         END
      3: BEGIN                              ; 8051 Mode
         CS_RAM = /(A[15] * A[14] * A[13])  ; RAM 00000 - 0dfff  8051 Mode
         CS_ROM  = 1;                       ; ROM 00000 - 0ffff
         A16_ROM = 0
         A16_RAM = 0
         RD_RAM  = /A[16]
         END
      END

  END

IF (MAP[1..0] = #b11) THEN                   ; 8051 Mode?
  BEGIN
    IF (A[15..10] = #b111000) THEN           ; ADDRESS e000 - e3ff
      BEGIN
        CS_USER = 1
      END
    ELSE
      BEGIN
        CS_USER = 0
      END

    IF (A[15..10] = #b111001) THEN           ; ADDRESS e400 - e7ff
      BEGIN
        CS_UART = 1
      END
    ELSE
      BEGIN
        CS_UART = 0
      END
  END
ELSE
  BEGIN
    IF (A[17..10] = #b11111000) THEN           ; ADDRESS 3e000 - 3e3ff
      BEGIN
        CS_USER = 1
      END
    ELSE
      BEGIN
        CS_USER = 0
      END

    IF (A[17..10] = #b11111001) THEN           ; ADDRESS 3e400 - 3e7ff
      BEGIN
        CS_UART = 1
      END
    ELSE
      BEGIN
        CS_UART = 0
      END
  END

;-------------------------------------------------------------------
```

Monitor EPROM Addresses

This following table lists the EPROM addresses in the Monitor EPROM for the different monitor versions and contains the EPROM addresses of the 251/151 configuration bytes CONFIG0 and CONFIG1. At address 0 and address 10000H the Monitor EPROM contains a LJMP 0E800H instruction.

Monitor Version	EPROM Addresses Monitor Program Code	CONFIG0	CONFIG1
8051/151 Monitor, PAGE Mode with Flash LED’s	00800H – 01FFFH	01FF8H	01FF9H
251 Monitor, Source Mode, PAGE, with Flash LED's	02800H – 03FFFH	03FF8H	03FF9H
8051/151 Monitor, PAGE Mode with Flash LED's	04800H – 05FFFH	05FF8H	05FF9H
251 Monitor, Binary Mode, PAGE, with Flash LED's	06800H – 07FFFH	07FF8H	07FF9H
8051 Monitor, Dallas 320 Version with Flash LED's251	08800H – 09FFFH	–	–
251 Monitor, Source Mode, NOPAGE, with Flash LED's	0A800H – 0BFFFH	0BFF8H	0BFF9H
8051/151 Monitor, NOPAGE, with Flash LED's	0C800H – 0DFFFH	0DFF8H	0DFF9H
251 Monitor, Binary Mode, NOPAGE, with Flash LED's	0E800H – 0FFFFH	0FFF8H	0FFF9H
8051/151 Monitor, PAGE Mode no Flash LED's	10800H – 11FFFH	11FF8H	11FF9H
251 Monitor, Source Mode, PAGE, no Flash LED's	12800H – 13FFFH	13FF8H	13FF9H
8051/151 Monitor, PAGE Mode no Flash LED's	14800H – 15FFFH	15FF8H	15FF9H
251 Monitor, Binary Mode, PAGE, no Flash LED's	16800H – 17FFFH	17FF8H	17FF9H
8051 Monitor, Dallas 320 Version, no Flash LED's251	18800H – 19FFFH	–	–
251 Monitor, Source Mode, NOPAGE, no Flash LED's	1A800H – 1BFFFH	1BFF8H	1BFF9H
8051/151 Monitor, NOPAGE, no Flash LED's	1C800H – 1DFFFH	1DFF8H	1DFF9H
251 Monitor, Binary Mode, NOPAGE, no Flash LED's	1E800H – 1FFFFH	1FFF8H	1FFF9H

Chapter 4. Programming

Writing programs for the MCB251 is relatively simple. However, before you get started, there are a few points you should keep in mind.

- The 251 supports a new hardware architecture that is different from that of the 8051. The 251 begins executing instructions at address 0FF0000h. The MCB251 is shipped with the Keil 251 monitor programmed into an EPROM on the MCB251 board. The MCB251 board has Monitor boot logic which is coded in the PLD device. At Reset the Monitor EPROM is addressable at address 0FF0000h. After the execution of the first Monitor instruction, the Monitor EPROM is addressed at 0FFE800h-0FFFFFFh.
- If you use the monitor for loading and running programs via the serial port on the MCB251, your programs still start at address 0FF0000h or C:0000. This is the same address as if you do not use a monitor program.

This following sections provide more information about writing programs for the MCB251 evaluation board.

Monitor Memory Map

The MCB251 evaluation board supports 248KB RAM memory which is addressable by for user application programs. The memory map depends on how you use the A17 address line. Refer to the *251 Microcontroller User's Manual* for more information about the memory map of the 251 CPU.

		Monitor Memory Mapping (Monitor DIP Switch is ON)		
Map1	**Map0**	**RAM**	**Monitor EPROM**	**UART CS & USER CS**
OFF	OFF 251 Mode	00:0000h–01: FFFFh FE:0000h–FF: DFFFh	FF:E800h–FF:FFFFh	UART: FF:E400h–FF:E7FFh USER: FF:E000h–FF:E3FFh
OFF	ON	illegal	illegal	illegal
ON	OFF	illegal	illegal	illegal
ON	ON 8051 Mode	X:0000h–X:DFFFh C:0000h-C:DFFFh (von Neumann mapped)	C:E800h–C:FFFFh	UART: X:E400h–X:E7FFh USER: X:E000h–X:E3FFh (von Neumann mapped)

The 251 CPU provided with the MCB251 evaluation board is programmed so that the RD# signal provides the A16 address line. This lets the 251 address up to 256K of external address space. The memory mapping during the test phase allows you to use the maximum address space of that CPU.

Monitor Data & Interrupt Vectors

The Monitors provided with the MCB251 use a minimal amount of your target's resources. The following sections describe what is required for the 251 monitor and for the 8051 monitor.

Monitor 251

For debugging, the 251 monitor uses the TRAP interrupt vector of the 251 CPU for handling breakpoints. In addition, you can enable the serial interrupt so that you can use dScope to halt a program that is running. The Temic 251G1 allows you to stop program execution with the NMI interrupt. The interrupt vector addresses listed in the following table should not be used by the application under test.

The 251 monitor uses the address space 0xFFDFC0-0xFFDFFF for monitor data. The user application should not modify these locations. Other than the interrupt vectors and the monitor data, the monitor program does not use and other memory resources of the 251 CPU.

You must reserve the following address spaces with the L251 linker RESERVE directive. When you reserve these memory areas, the linker prevents your target program from using them.

Monitor Data Space and Interrupt Vectors			
Monitor Mode	**Interrupt Vectors**		**Monitor Data**
Serial Interrupt disabled	**TRAP:** **NMI:**	**FF:007Bh–FF:007Dh,** **FF:003Bh–FF:003Dh**	**FF:DFC0h–FF:DFFFh**
EXT RS232 **Serial Interrupt enabled**	**INT0:** **NMI:** **TRAP:**	**FF:0003h–FF:0005h** **FF:003Bh–FF:003Dh** **FF:007Bh–FF:007Dh**	**FF:DFC0h–FF:DFFFh**
INT RS232 **Serial Interrupt enabled**	**SINT:** **NMI:** **TRAP:**	**FF:0023h–FF:0025h** **FF:003Bh–FF:003Dh** **FF:007Bh–FF:007Dh**	**FF:DFC0h–FF:DFFFh**

Monitor 51

The 8051 monitor does not require the use of any interrupt vectors. The serial interrupt is not required for serial communication. However, if you enable the serial interrupt, you must reserve the interrupt vector space for the internal interrupt 0 or serial interrupt.

The 8051 monitor uses the address space X:0xDF00-X:0xDFFF for monitor data. If you enable “Record Trace”, then the address space X:0xCC00-X:0xDFFF is used for monitor data. The complete CODE/XDATA space is von-Neumann wired so that accesses to the XDATA space modify the data in the CODE space.

Writing Programs for the 251 Monitor

Compiling and linking programs for use with the 251 monitor on the MCB251 board requires only two steps.

First, compile your applications as you normally would. Make sure that you have selected the correct mode (**MODBIN** or **MODSRC**) directive to specify 251 operating mode you have configured on the MCB251 board. For example, you may use the following command line.

```
C251 MYCODE.C MODSRC
```

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

```
L251 MYCODE.OBJ RESERVE (0FF0003H-0FF0005H, 0FF003BH-0FF003DH,
                                                        0FF007BH-0FF007DH)
```

In this command line, the RESERVE directive tells the linker not to use the specified address space for the user application.

NOTE
*The only difference between writing programs for the Monitor 251 compared to other user applications is the **RESERVE** directive for the L251 linker. However, you can still use the **RESERVE** directive, if you want to program your application into an EPROM.*

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.

BLINKY Example Program

The following simple program, BLINKY, is an exercise you may use to test the MCB251 and verify that you can use the tools provided to generate a working program.

BLINKY blinks the LEDs on the MCB251 evaluation board. The complete source listing for the program is shown below:

```
/* BLINKY.C - LED Flasher for the Keil MCB251 Evaluation Board */

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

void wait (void)  {                  /* wait function */
  ;                                  /* only to delay for LED flashes */
}

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

  while (1) {                        /* Loop forever */
    for (j=0x01; j< 0x40; j<<=1)  {   /* Blink LED 0, 1, 2, 3, 4, 5, 6 */
      P1 &= 0x80;                    /* do not change A17 port line   */
      P1 |= j;                       /* Output to LED Port */
      for (i = 0; i < 10000; i++)  {  /* Delay for 1000 Counts */
        wait ();                     /* call wait function */
      }
    }

    for (j=0x40; j> 0x01; j>>=1)  {   /* Blink LED 6, 5, 4, 3, 2, 1 */
      P1 &= 0x80;                    /* do not change A17 port line   */
      P1 |= j;                       /* Output to LED Port */
      for (i = 0; i < 10000; i++)  {  /* Delay for 10000 Counts */
        wait ();                     /* call wait function */
      }
    }
  }
}
```

You may build the BLINKY 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 BLINKY example program:

```
C251 BLINKY.C DEBUG MODSRC
```

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

```
L251 BLINKY.OBJ RESERVE (0FF0003H-0FF0005H, 0FF003BH-0FF003DH,
                                                         0FF007BH-0FF007DH)
```

Using the Windows-Based Tools

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

Using µVision to Create the BLINKY Program

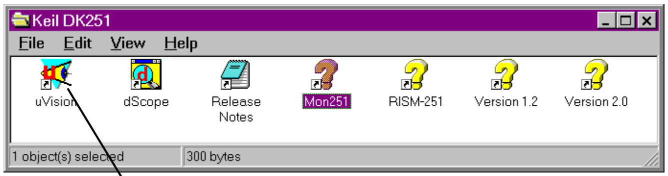
To create the BLINKY example program using µVision, you need to perform the following steps:

- Create the BLINKY.C source file.
- Create the BLINKY project file.
- Include BLINKY.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.

Create the BLINKY.C Source File

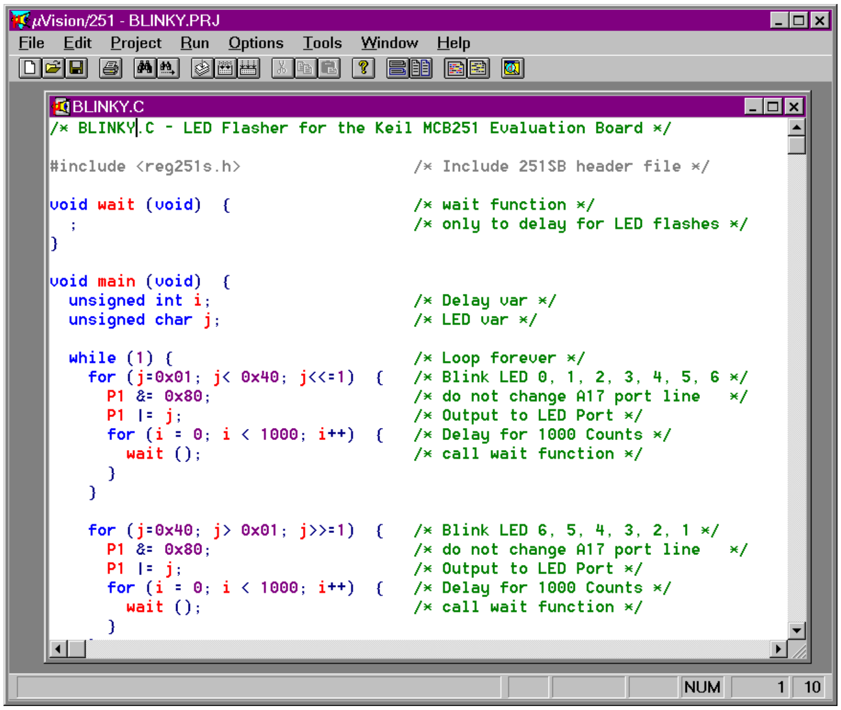
Start µVision by double-clicking on the icon in the 251 Tools group.



µVision Program Icon

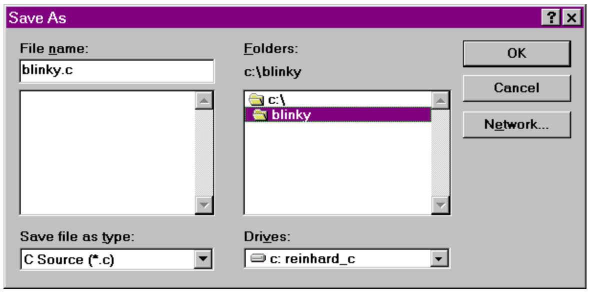
When µVision starts, select the New command from the File menu and µVision opens a new text window in which you may create the BLINKY program.

Enter the BLINKY example program shown on page 31. Your screen should look something like the following figure.



```
/* BLINKY.C - LED Flasher for the Keil MCB251 Evaluation Board */

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

void wait (void)  {                   /* wait function */
  ;                                   /* only to delay for LED flashes */
}

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

  while (1) {                         /* Loop forever */
    for (j=0x01; j< 0x40; j<<=1)  {   /* Blink LED 0, 1, 2, 3, 4, 5, 6 */
      P1 &= 0x80;                     /* do not change A17 port line   */
      P1 |= j;                        /* Output to LED Port */
      for (i = 0; i < 1000; i++)  {   /* Delay for 1000 Counts */
        wait ();                      /* call wait function */
      }
    }

    for (j=0x40; j> 0x01; j>>=1)  {   /* Blink LED 6, 5, 4, 3, 2, 1 */
      P1 &= 0x80;                     /* do not change A17 port line   */
      P1 |= j;                        /* Output to LED Port */
      for (i = 0; i < 1000; i++)  {   /* Delay for 1000 Counts */
        wait ();                      /* call wait function */
      }
```

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

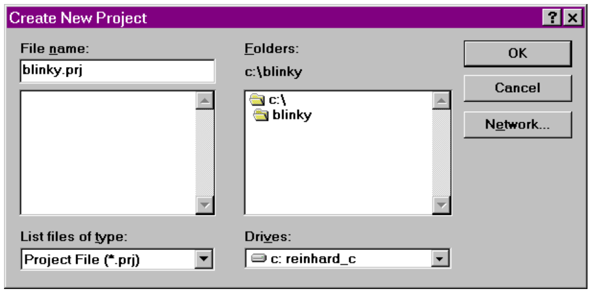


To save the program as BLINKY.C, enter BLINKY.C in the File Name text box at the top of the Save As dialog box. You may want to save the flash program and the other files you create in a separate directory. This example uses C:\BLINKY for the source files and project files.

Create the BLINKY Project File

After you save BLINKY.C, you should create a BLINKY 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.

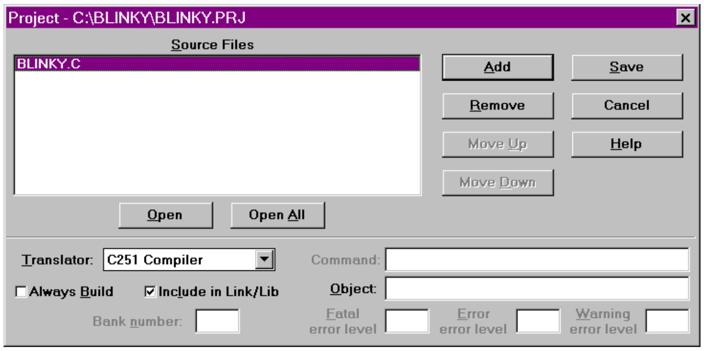
To create a project file for BLINKY, select the New Project… command from the Project menu. µVision displays the dialog box shown below. Enter the name for the project in the File Name text box. This example uses BLINKY.PRJ.



NOTE
Always use ***PRJ*** *as the file extension for project files.*

Include BLINKY.C in the Project

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


Project - C:\BLINKY\BLINKY.PRJ

Source Files: BLINKY.C

Translator: C251 Compiler

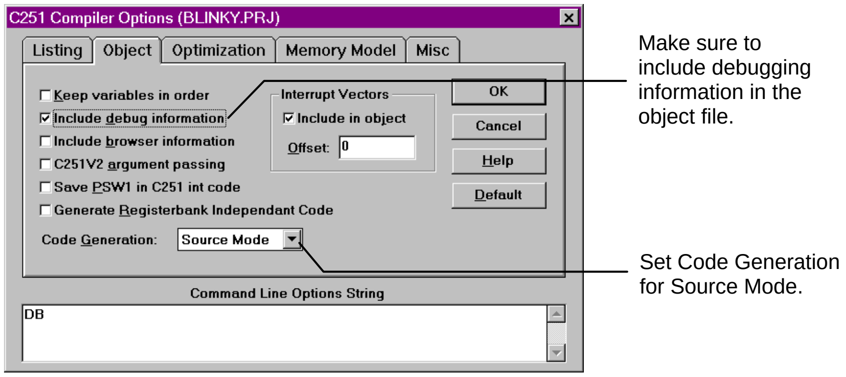
☐ Always Build ☑ Include in Link/Lib

Click the Insert button and choose the BLINKY.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.

Set the C251 Compiler Options for the Project

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 BLINKY example are Include debug information and Source Mode for Code generation. These controls are shown in the following figure.


C251 Compiler Options (BLINKY.PRJ)
Listing
Object
Optimization
Memory Model
Misc
OK
Cancel
Help
Default
Command Line Options String

- ☐ Keep variables in order
- ☑ Include debug information
- ☐ Include browser information
- ☐ C251V2 argument passing
- ☐ Save PSW1 in C251 int code
- ☐ Generate Registerbank Independant Code

Code Generation: Source Mode

Interrupt Vectors
- ☑ Include in object
- Offset: 0

DB

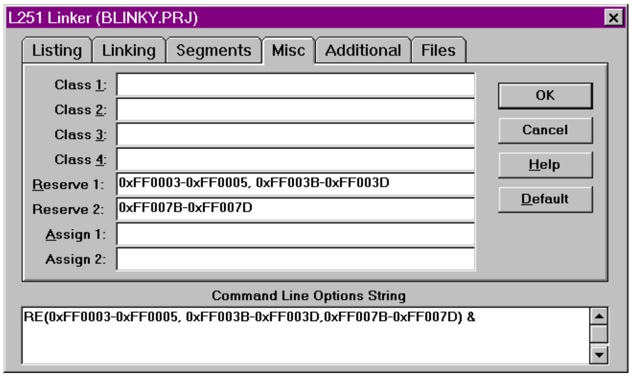
Make sure to include debugging information in the object file.

Set Code Generation for Source Mode.

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

Set the L251 Linker Options for the Project

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.



Class 1:
Class 2:
Class 3:
Class 4:
Reserve 1: 0xFF0003-0xFF0005, 0xFF003B-0xFF003D
Reserve 2: 0xFF007B-0xFF007D
Assign 1:
Assign 2:

Command Line Options String

```
RE(0xFF0003-0xFF0005, 0xFF003B-0xFF003D,0xFF007B-0xFF007D) &
```

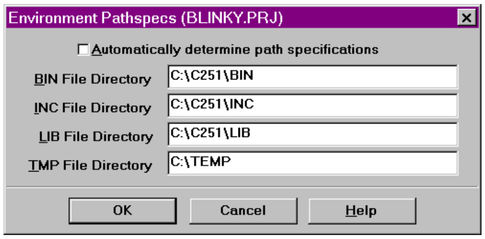
Enter the following text in the Reserve text boxes on the Misc tab shown above.

```
0xFF0003-0xFF0005, 0xFF003B-0xFF003D, 0xFF007B-0xFF007D
```

This creates the reserve directive that tells the linker that memory areas 0xFF0003-0xFF0005, 0xFF003B-0xFF003D and 0xFF007B-0xFF007D should not be used for program code. This reserves the space for the interrupt vectors.

Set the Path Specifications for the 251 Tools

You may wish to specify the path to the 251 tools directly in µVision. You may do this in the Environment Pathspecs dialog box. Open this dialog box using the Environment Pathspecs… command in the Options menu.


Environment Pathspecs (BLINKY.PRJ)

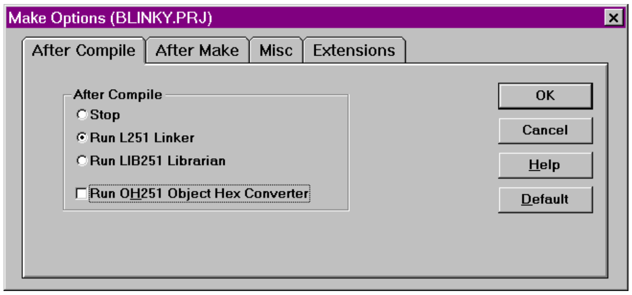
☐ Automatically determine path specifications

BIN File Directory	C:\C251\BIN
INC File Directory	C:\C251\INC
LIB File Directory	C:\C251\LIB
TMP File Directory	C:\TEMP

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.

Set the Make Options for the Project

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 Options (BLINKY.PRJ)

After Compile | After Make | Misc | Extensions

After Compile
- Stop
- [✓ Run L251 Linker]
- Run LIB251 Librarian

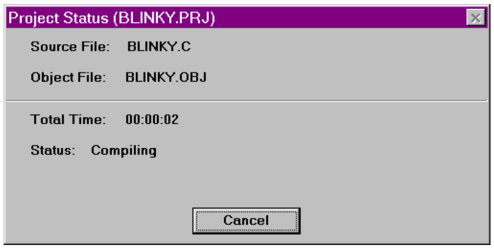
☐ Run OH251 Object Hex Converter

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

Build the Project

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

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


Project Status (BLINKY.PRJ)

Source File: BLINKY.C

Object File: BLINKY.OBJ

Total Time: 00:00:02

Status: Compiling

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 BLINKY program.

Using dScope to Debug the BLINKY Program

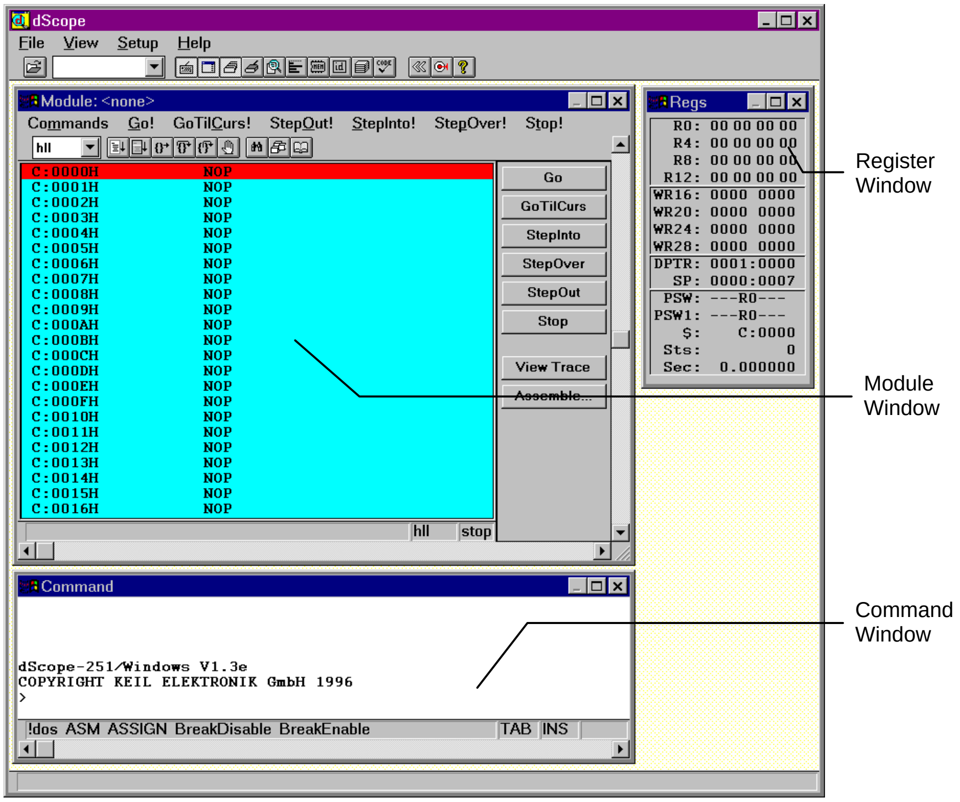
To load the BLINKY in the MCB251 evaluation board using dScope, you need to perform the following steps:

- 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 BLINKY program.
- Step through the BLINKY 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.



```
C:0000H      NOP
C:0001H      NOP
C:0002H      NOP
C:0003H      NOP
C:0004H      NOP
C:0005H      NOP
C:0006H      NOP
C:0007H      NOP
C:0008H      NOP
C:0009H      NOP
C:000AH      NOP
C:000BH      NOP
C:000CH      NOP
C:000DH      NOP
C:000EH      NOP
C:000FH      NOP
C:0010H      NOP
C:0011H      NOP
C:0012H      NOP
C:0013H      NOP
C:0014H      NOP
C:0015H      NOP
C:0016H      NOP
```

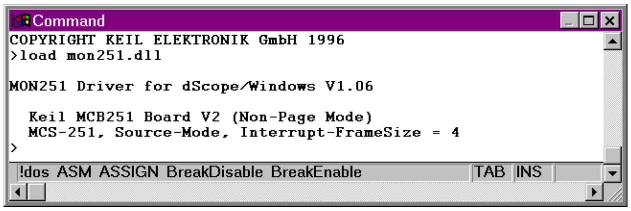
```
  R0: 00 00 00 00
  R4: 00 00 00 00
  R8: 00 00 00 00
 R12: 00 00 00 00
WR16: 0000 0000
WR20: 0000 0000
WR24: 0000 0000
WR28: 0000 0000
DPTR: 0001:0000
  SP: 0000:0007
 PSW: ---R0---
PSW1: ---R0---
   $:    C:0000
 Sts:         0
 Sec:  0.000000
```

```
dScope-251/Windows V1.3e
COPYRIGHT KEIL ELEKTRONIK GmbH 1996
>
```

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

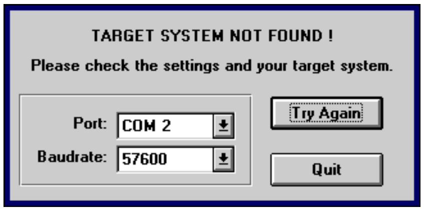
```
load mon251.dll
```

This is shown in the following figure.



```
COPYRIGHT KEIL ELEKTRONIK GmbH 1996
>load mon251.dll

MON251 Driver for dScope/Windows V1.06

  Keil MCB251 Board V2 (Non-Page Mode)
  MCS-251, Source-Mode, Interrupt-FrameSize = 4
>
```

Typically, the first time you load the CPU driver for the 251 monitor, you must set the COM port and baud rate. 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.



TARGET SYSTEM NOT FOUND !

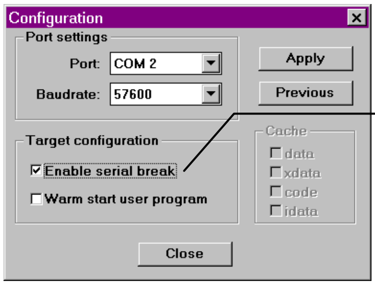
Please check the settings and your target system.

Port: COM 2

Baudrate: 57600

When using the MCB251 evaluation board, the baud rate should be set for 57,600 baud. You must determine which COM port you use.

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.



Configuration

Port settings

Port: COM 2

Baudrate: 57600

Target configuration

[✓] Enable serial break

[ ] Warm start user program

Cache

[ ] data

[ ] xdata

[ ] code

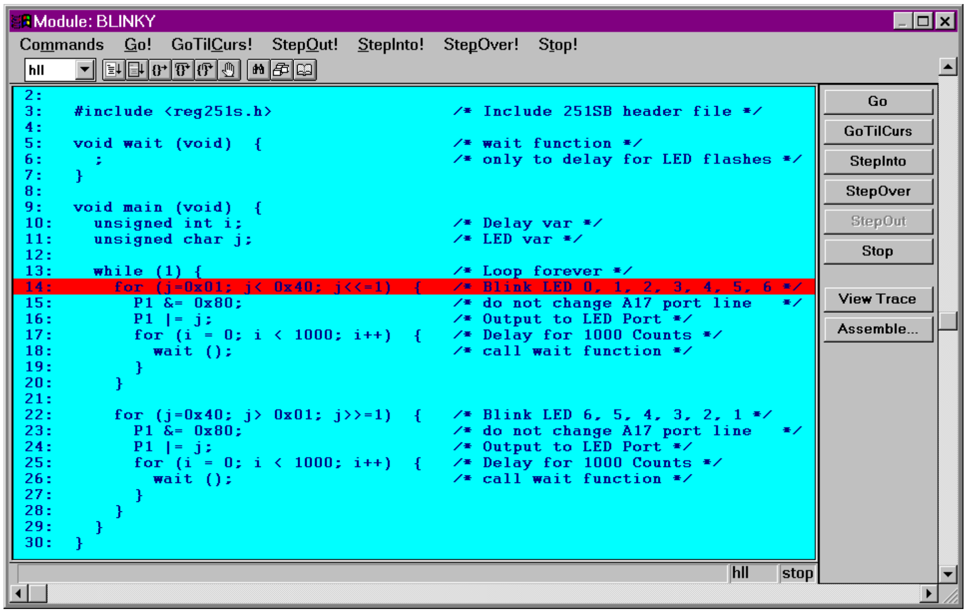
[ ] idata

Set Enable serial break to let dScope stop programs running on the MCB251.

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

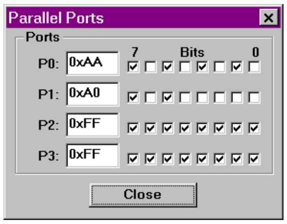
```
load blinky                         /* this loads the blinky program */

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

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



```

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

void wait (void)  {                   /* wait function */
  ;                                   /* only to delay for LED flashes */
}

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

  while (1) {                         /* Loop forever */
    for (j=0x01; j< 0x40; j<<=1)  {   /* Blink LED 0, 1, 2, 3, 4, 5, 6 */
      P1 &= 0x80;                     /* do not change A17 port line    */
      P1 |= j;                        /* Output to LED Port */
      for (i = 0; i < 1000; i++)  {   /* Delay for 1000 Counts */
        wait ();                      /* call wait function */
      }
    }

    for (j=0x40; j> 0x01; j>>=1)  {   /* Blink LED 6, 5, 4, 3, 2, 1 */
      P1 &= 0x80;                     /* do not change A17 port line    */
      P1 |= j;                        /* Output to LED Port */
      for (i = 0; i < 1000; i++)  {   /* Delay for 1000 Counts */
        wait ();                      /* call wait function */
      }
    }
  }
}
```

You may now single step through the BLINKY 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 MCB251 changing.



		7			Bits				0
P0:	0xAA	✓	☐	✓	☐	✓	☐	✓	☐
P1:	0xA0	✓	☐	✓	☐	☐	☐	☐	☐
P2:	0xFF	✓	✓	✓	✓	✓	✓	✓	✓
P3:	0xFF	✓	✓	✓	✓	✓	✓	✓	✓

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 dialog box shown on the right.

As you step through the BLINKY program, you will see how the Parallel Ports dialog box mirrors the LEDs on the MCB251 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 program execution 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    0xFFE700UL

#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 MCB251.
-----------------------------------------------*/
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 MCB251.
-----------------------------------------------*/
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);
}

/*------------------------------------------------
Wait dummy function
------------------------------------------------*/
void wait (void)
{
  ;                 /* just to waits some time */
}

/*------------------------------------------------
Setup the external UART for 57600 baud, write the
string "Hello World nnn", and repeat.
------------------------------------------------*/
void main (void)
{
unsigned int counter = 0;

extsio_setup (57600);

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

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

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

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 MCB251 comes ready-programmed with the Keil 251 monitor (MON251). The monitor programmed in the Monitor EPROM is setup to communicate at 57,600 baud using the EXT_RS232 interface and 19,200 baud using the INT_RS232 interface with a 12MHz 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 (1, 2, 3, or 4) 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 rather than interrupt-driven. 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. For example:

```
SET MON251=COM1 BAUDRATE(1200)
```

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>
```

Terminal 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 following 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.

Index