FE-51CC03

Development System

for Atmel

AT89C51CC03 Microcontrollers

User's Manual

Rev. 01/19

CONTENTS

PREFACE

CHAPTER 1. DESCRIPTION OF THE SYSTEM

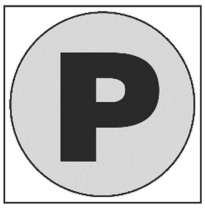
CHAPTER 2. INSTALLATION AND QUICK START-UP

CHAPTER 3. ABOUT THE WINDOWS DEBUGGER

CHAPTER 4. WINDOWS DEBUGGING SESSION

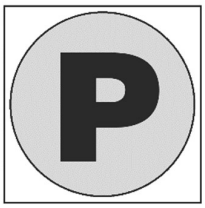
CHAPTER 5. WINDOWS MENUS AND COMMANDS

PREFACE



FE-51RD2/CC03 Development System

PREFACE



FE-51RD2/CC03 Development System

The FE-51RD2 *Development System* for ATMEL AT89C51RD2/CC03 Microcontrollers derivatives with 6/12 clocks/cycle is very effective in meeting the diverse demands of emulation operations.

P.1. Features

- Emulates 89C51RD2/CC03 Derivatives with 6/12 Clocks/Cycle
- 60K Code Memory
- Real-Time Emulation
- Frequency up to 20MHz/3V, 33MHz/5V
- ISP and X2 Mode Support
- MS-Windows Debugger for C and Assembler
- Keil µVision Debugger Compatible
- Emulation Header and Signal Testpoints
- Target Board and Programmer Included
- Serially Linked to PC at 115Kbaud

P.2. The Technology

FE-51RD2/CC03 uses ATMEL standard devices for real-time and transparent emulation.

The system accepts devices operating with 6 clocks per cycle and also 12 clocks per cycle without any special setup.

Furthermore, the microcontrollers can be used in “emulation mode” where I/O ports are not affected by the emulation process.

FE-51RD2 is not frequency or voltage restricted, so it can be used to emulate the microcontroller in the complete range of parameters defined by the device.

P.3. The Software

The FE-51RD2 is supplied with a MS-Windows Debugger.

For the complete description of the Windows debugger, please refer to Chapter 3, 4 and 5. The available working modes:

- Emulation - this is a real time mode

- Simulator - this is not a real time mode

are introduced in Chapter 1.

P.4. Emulation Restrictions and Troubleshooting

You must read carefully Chapter 1, especially the ***Emulation Support and Emulation Restrictions*** paragraphs before using the hardware system. These paragraphs will explain how to prepare your project to take full advantage of the system.

Chapter 8 gives some ***troubleshooting*** recommendations to follow in case that you are experiencing problems with your hardware or software.

P.5. About the Manual

1.Description of the System

Describes the system and its capabilities, as well as detailing its specifications and applications.

2.Installation

Provides step-by-step instructions on software and hardware installation procedures. It is recommended that this chapter be read in its entirety prior to connecting the system.

3.About the Windows Debugger

Includes the basic information you will need to begin using the Ceibo Windows Debugger.

4.Windows Debugging Session

Gives a session example for a quick introduction to the Windows Debugger capabilities.

5.Windows Menus and Commands

Details the use of the Windows Debugger menus and their related commands.

6.Real Time Emulation

Gives all the information regarding the system in real-time Emulation Mode.

7.On-Line Assembler

Describes the syntax used by the On-line Assembler command. It also gives a complete list of examples of the instruction set.

8.System Errors and Troubleshooting

Lists the errors detected while operating FE-51RD2 and gives common questions and answers for troubleshooting.

CHAPTER 1

Description of the System

CHAPTER 1

Description of the System

1.1. Introduction

This chapter describes the capabilities and applications of the system. The technical specifications and the emulation restrictions are detailed in the following paragraphs.

1.2. General Description

Ceibo FE-51RD2/CC03 is a development system that supports ATMEL AT89C51RD2/CC03 microcontrollers with 6/12 clocks/cycle at any frequency allowed by the devices. It is serially linked to a PC or compatible systems and can emulate the microcontrollers using either the built-in clock generator or any other clock source connected to the microcontroller.

Emulation is carried out by loading the system with the user software and an embedded monitor program. FE-51RD2/CC03 locates the monitor in the upper 4K of the code memory space.

Two working modes are available: real-time and simulator.

In the ***real-time mode*** the user software is executed transparently and without interfering with the microcontroller speed. Breakpoints can be added to stop program execution at a specific address.

The ***simulation mode does not implement all the chip options*** and it is intended only for software debugging of the basic 8051 functions. ***Use the system in emulation mode.*** The simulation mode is used to debug the software without any hardware. FE-51RD2/CC03 may be disconnected while using the simulation mode.

The software includes C and Assembler Source Level Debugger, On-line Assembler and Disassembler, Trace, Conditional Breakpoints and many other features.

The system is supplied with Windows debugger software, RS-232 cable and a power supply.

The system includes 3 main hardware boards:

1. FE-51RD2/CC03 - In-circuit Emulator

2. DB-51RD2 - Development Board

3. MP-51RD2 - Microcontroller Programmer

This chapter describes the three parts of the systems.

1.3. Applications

The main applications of FE-51RD2/CC03 Development System are:

- Emulation of microcontrollers
- Demonstration of microcontroller capabilities
- Development of microprocessor based systems
- Hardware and software debugging purposes
- Training in the field of microprocessors

1.4. Specifications

System Memory

FE-51RD2 provides 64K of code memory. However, only 60K are available for user's programs while emulating because the system comes with an embedded monitor program that uses the upper 4K of the memory space. Code memory is mapped as belonging to the FE-51RD2 Emulator.

Breakpoints

Breakpoints allow real-time program execution until an opcode is executed at a specified address.

Windows Debugger

The FE-51RD2 software includes a source level debugger for Assembler and high-level languages C and others with the capability of executing lines of the program while displaying the state of any variable. The debugger uses symbols contained in the absolute file generated by the most commonly used Assemblers and High Level Language Compilers. The CEIBO Windows Debugger runs only under Windows 95 or later and also under Windows NT.

Supported Microcontrollers

The system supports ATMEL AT89C51RD2/CC03 microcontrollers and other derivatives that will be announced in the future. The standard supported package for emulation is PLCC.

Frequency

FE-51RD2/CC03 runs from the clock source supplied by the user hardware. The minimum and maximum frequencies are determined by the emulated chip characteristics, while the emulator maximum frequency is 40MHz. The hardware has been upgraded and now includes a programmable clock generator. The target frequency is according to the setup in Options/ Architecture/ XTAL menu.

Host Characteristics

PC or compatible systems with Windows 7 or later.

Input Power

5V, 1.5A power supply supplied.

Mechanical Dimensions

10cm x 10cm.

Items Supplied as Standard

Development system including emulator, programmer and development board, PLCC emulation header, Windows software with source level debugger, on-line assembler and disassembler, user's manual, RS-232 cable, USB adapter and power supply.

Options

QFP and DIP adapters.

1.5. Hardware Description - FE-51RD2 Emulator

The first step required to work with the FE-51RD2 board is to be able to identify its different parts and to understand how the electronics function. This will help you to take full advantage of all the FE-51RD2 capabilities.

On the bottom side of the FE-51RD2 you will see a phone jack. It is used to serially link the FE-51RD2 to your computer and to utilize the software in emulation mode instead of just the simulator. The emulation mode is used to interact with the hardware while the simulator is independent of any hardware connection. The RS-232 cable connections are given separately. A supplied USB adapter can be used too.

On the right side of the RS-232 connector you will see a POWER LED indicating whether or not the power is applied to the system. ***The emulator must be connected to a target to be powered and to receive the clock input, otherwise it will not work. If you do not have your target system, you can use DB-51RD2 Board that is included in the system.***

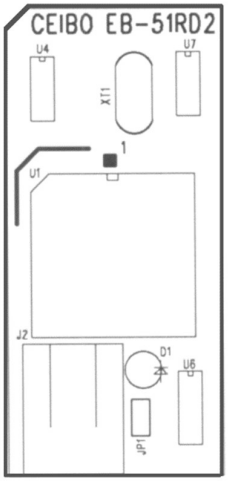


FIGURE 1.1: ***FE-51RD2 Emulator***

U1 is the microcontroller used for your emulation. It is placed on a 44-PLCC socket.

On the other side of the board there is a PLCC plug that you must connect to a target board. ***Make sure that pin #1 of your target is aligned to pin #1 of the emulator***, which is clearly marked and has the same direction as U1.

JP1 is a connector necessary only for firmware updates. Make sure that the cap is not placed on it to work properly. If any firmware update will be necessary, follow the instructions that will come with it in **www.ceibo.com**.

1.6. Hardware Description - DB-51RD2 Development Board

This board is supplied to be used as a possible target and to test your software.

It provides the following functionality:

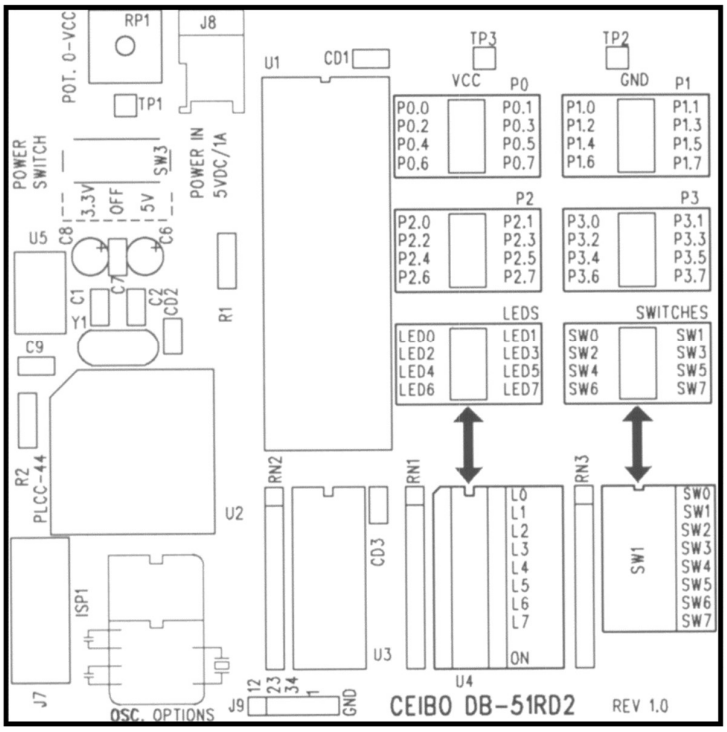


FIGURE 1.2: ***DB-51RD2 Development Board***

1. Testpoints for Port Lines. Please note that ***Port 0 is open-drain*** and if you want to see a high level on these lines you must ***add pull-up resistors***. The system includes a spare resistor network that can be plugged into the 40-pin socket of the target board to provide the pull-ups. This is shown in the following figure. Make sure that the common of the resistor network (1 common + pull-ups) is connected to Pin 40, which is the Vcc line.

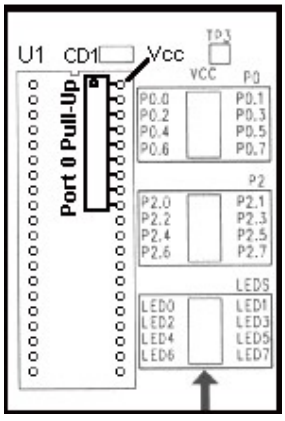


FIGURE 1.3: ***Adding Pull-Ups to Port 0 on DB-51RD2***

2. LEDs. One LED is the ON/Off indicator while L0 to L7 can be used to display high-levels while connecting them to the Ports or other signals. An 8-pin jumper is supplied for that connection.

3. Switches. The SWITCHES connector provides logic states to input ports. The same supplied 8-wire ribbon cable can be used to connect the ports to the Switches, thus allowing to easily test your program.

4. Sockets. DIP and PLCC for microprocessors can be connected to this board. ***You may connect FE-51RD2 emulator to PLCC-44 socket.*** In such a case, make sure that U1 socket does not have a microprocessor on it.

5. Power Switch. The power switch is on the left side of the board. It has three positions: OFF, 3.3V and 5V. Although the input voltage is always 5V, the system has a built-in voltage regulator (U5) that drops the voltage to 3.3V if the switch is set accordingly. ***Use the power supply included in the system*** and connect it to the Power In jack (J8).

6. Potentiometer. This 50KOhm potentiometer is used to get on TP1 analog input from 0 to Vcc (which is 3.3V or 5V according to the power switch position).

7. ISP Connector. J7 is an ISP connector that can be used directly with MP-51RD2 programmer to program the microcontroller installed on the 40-DIP or 44-PLCC socket. A 10-pin ribbon cable is supplied for this purpose.

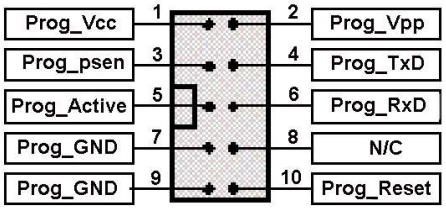


FIGURE 1.4: ***ISP Connector***

8. Oscillator Options. This board can be used to connect a full size crystal oscillator (16-DIP), half size (8-DIP) or a 2-pin crystal with capacitors as shown on the board.

9. Future Options. The board is prepared to be used with future derivatives, which have pin signals not supported by the 89C51RD2 and as shown on J9 connector. Additional frequency options for new derivatives are prepared with C1, C2 and Y1 crystal placements.

1.7. Hardware Description -MP-51RD2 Programmer

This small board contain the RS-232 interface to ISP signals. Use it together with the supplied 10-pin ribbon cable and the software driver available from ATMEL W&M website.

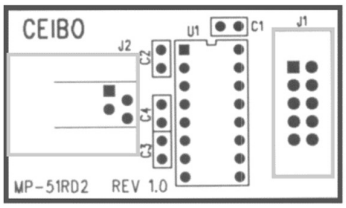


FIGURE 1.5: ***MP-51RD2 Programmer***

How to Use the MP-51RD2 Programmer:

1. Connect it to your COM port using the RS-232 cable supplied with the system (same as for the FE-51RD2 emulator).

2. Connect the ISP cable supplied with the system to the programmer and also to the DB-51RD2 target board.

3. Place a microcontroller in one of the DB-51RD2 sockets (DIP or PLCC).

4. Make sure that the crystal on the DB-51RD2 is installed.

5. Power up the DB-51RD2 board.

6. Invoke the ISP software. This is not Ceibo's debugger, but ISP utilities provided by Atmel W&M and available in their web site.

1. 8. Emulation Restrictions

The following restrictions are valid for FE-51RD2:

1. FE-51RD2 Monitor Program shares 4 KByte of the 64K memory code space. Therefore, user programs can be up to 60 KBytes.

2. Code memory cannot be mapped external and always belongs to the emulator system.

3. The program also uses 4 Bytes of the internal stack memory.

4. The stack pointer may not be defined below address 7.

5. The first instruction (address 0000h) must be 3-bytes long. For example: use LJMP and not SJMP or AJMP as the first instruction.

6. The UART is shared with the system and interrupts should not be disabled. Also the related timer to the serial port must not be stopped.

7. Breakpoints cannot be set to the Reset Vector (address 0000h-0002h), Serial Interrupt (address 0023h-0025h) or to the 4K of the reserved memory space (address F000h-FFFFh). Any other location is allowed, included inside interrupt service routines.

CHAPTER 2

Installation and Quick Start-Up

CHAPTER 2

Installation and Quick Start-Up

2.1. Introduction

This chapter describes hardware and software installation procedures and details the connection steps required before starting up.

2.2. RS-232C Interface

FE-51RD2 is serially linked to a host computer through an RS-232C interface. A supplied USB adapter can be used too. A standard phone cable is used to connect the FE-51RD2 to COM1, 2, 3 or 4 in the host computer. This cable has a 9-pin female connector to fit into the COM1 (or other) connector, and a phone plug to connect to the FE-51RD2 serial interface jack.

If your computer has a 25-pin connector for the RS-232 interface, use an additional 9-pin to 25-pin adapter.

The cable characteristics are given in Table 2.1.

Signal Name	Phone Jack (FE-51RD2)	9-Pin Female Connector (PC Side)
Receive Data	pin 1	pin 2
Transmit Data	pin 2	pin 3
Signal Ground	pin 4	pin 5

TABLE 2.1: ***RS-232 Cable***

2.3. Connecting the FE-51RD2 Components

No special tools are required to install the FE-51RD2.

For proper installation the host must be properly configured and the power should be OFF.

Carry out the following steps to connect the system components:

a) Connect the FE-51RD2 Emulator to DB-51RD2 Development Board or any other target you may have.

b) Connect the serial cable into the serial jack on FE-51RD2.

c) Connect the other end of the serial cable to host PC serial channel COM1 (COM2, COM3 or COM4).

d) Plug the power supply into the power connector on the DB-51RD2 Development Board or any other target you may have.

e) Plug the other end of the cable into a power outlet and turn on the power switch.

2.4. Installing the Software

Follow the steps described below to install your FE-51RD2 software:

- Turn on the host computer.
- Run Windows.
- Run de latest debugger available in www.ceibo.com

The Windows Setup creates the Ceibo Windows Debugger icon.

2.5. Starting Up

Turn the FE-51RD2 power supply on. Set the power switch to the 5V position.

Double click the Debugger icon.

The system will display a copyright screen after successfully invoking the software. Now the system is ready for operation.

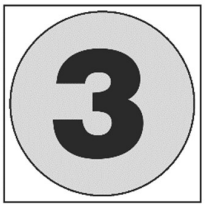
If the software responds with a message indicating that the system is not connected, check the following:

1. Power supply - the Power LED must be on.

2. RS-232 - check the connections to your computer.

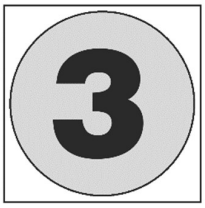
3. Communication port - verify that the connected serial port corresponds to the setup of the system.

CHAPTER 3



About the Windows Debugger

CHAPTER 3



About the Windows Debugger

3.1. Introduction

This chapter includes the basic information you will need to begin using the Ceibo Windows Debugger program and to understand the meaning of the different menus.

3.2. Debug Capabilities

The Windows Debugger is used to load a program, execute it in real-time, simulate the software environment as well as many other functions.

You may be familiar with other debuggers' nomenclature. Nevertheless, the following review is useful to understand some terms used by The Ceibo Windows Debugger.

Tracing

A program may be executed one line at a time. You can trace a program using high-level language lines or assembly instructions.

Stepping

This is like tracing but program execution steps over CALL instructions without leaving the current procedure.

Viewing

Ceibo Windows Debugger opens special windows showing your program state from various perspectives: variables and their values, breakpoints, a source file, CPU registers, memory, peripheral registers, etc.

Inspecting

The debugger can delve deeper into your program and show you the variable contents. This function provides more information about the variable than viewing it in the Watches window, such as full variable information and details (address, type of variable, type components for complex types, etc.).

Changing

The current value of a variable can be replaced with your specified value.

Watching

Program variables can be isolated and their values kept track of while the program runs.

3.3. Global Menus

A Global Menu is the list of commands easily accessible from a bar which runs along the top of the window.

A pull-down menu is available for each item on the menu bar and allows the following:

- Execute a command.
- Open a pop-up menu. Pop-up menus appear when a menu item is chosen followed by a menu icon (▶).
- Open a dialog box. If a menu item is followed by (**...**), a Dialog box appears when this particular item is selected.
- Check an option to select it.

Global menus are accessed by pressing F10 and using the arrow keys, pressing Alt and typing the first letter of the menu name or clicking the option.

Some of the menu commands have hot key shortcuts that are available from any part of the Debugger.

3.4. Local Menus

The Windows Debugger is context-sensitive and uses Local Menus specifying different windows. Local menus are tailored to the particular window you are in. It is important not to confuse them with global menus.

To prompt a local menu press Alt-F10 or click the right button of your mouse.

Menu placement and contents depends on which window or pane you are in and where your cursor is.

Contents may vary from one local menu to another. Many local commands appear in almost all local menus. The results of these similarly-named commands may differ, depending on the context.

Every command on a local menu has a hot key shortcut consisting of Ctrl plus the underscored letter in the command.

Because of this setup, a hot key, like Ctrl-C might mean one thing in one context but something quite different in another. The core commands are still consistent across local menus. For example, the Go To command and the Search command always do the same thing, even when they are invoked from different windows.

3.5. Input Boxes

Many of the Windows Debugger command options are available in input boxes. An input box prompts you to type in a string. All your entries are recorded in a history buffer, so you can pick up any entry just by selecting it with the arrows.

3.6. Windows

The Windows Debugger displays all information and data in both global and local menus, dialog boxes (where options are set and information entered) and windows.

There are many window types, depending on the kind of information it holds.

Windows may be opened and closed using menu commands (or hot key shortcuts for those commands).

After a window has been opened, you can move, resize, close, and otherwise manage them with commands from the Window and System menus.

3.7. Using the Menus

There are four ways to open the menus:

1. Press F10, use left or right arrow to get to the desired menu, press Enter.
2. Press F10, then the first letter of the menu name (F,V,R,B,D,O,W,H).
3. Press Alt plus the first letter of any menu bar command. For example, wherever you are in the system, Alt-F takes you to the File menu.
4. Click in the menu bar command with the mouse.

To navigate within the global system:

1. Use left and right arrows to move from one pull-down menu to another.
2. Use up and down arrow to scroll through the commands in a specific menu.

3. Highlight a menu command and press Enter to move to a lower-level (pop-up) menu or dialog box.

To get out of a menu or the menu system:

1. Press Esc to exit a lower-level menu and return to the previous menu.
2. Click the active window with the mouse to leave the menu system and return to the active window.
3. Press F10 to return to the current active window.

Some menu commands have a shortcut "hot key" that you press to execute them. The hot key appears in the menu to the right of these commands.

3.8. Toolbar

FIGURE 3.1: ***Toolbar***

The buttons on the ***Toolbar*** are the commands you need to operate the most useful functions:

 get help information

 open a file dialog box

 open the list of Modules dialog box

 select the CPU window

 select the Watches window

 run a program

 instruction step

 step with skip calls

 go to cursor

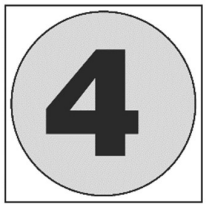
halt a program

3.9. Status Line

FIGURE 3.2: ***Status Line***

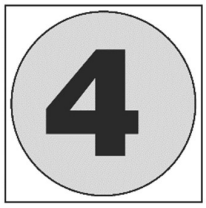
The ***status line*** on the bottom of the main application window displays messages related to the cursor position in the Module window, chip type, operating mode (simulation, emulation or in-circuit simulation) and current status (program running, ready, error). It also provides on-line help information on selected menus.

CHAPTER 4



Windows Debugging Session

CHAPTER 4



Windows Debugging Session

4.1. Introduction

Follow the steps of the session example for a quick introduction to the Windows Debugger capabilities.

The complete explanation of menus and commands is given in the following chapters.

Following the steps explained in this chapter will give you a better understanding of the debugger environment.

4.2. Preparing the Software

1. Do not apply power to the FE-51RD2. It is not necessary for the first stage which mostly explains the software capabilities.

2. Invoke the Windows Debugger by double clicking the Ceibo icon.

4.3. Selecting the Simulation Mode

Select the Simulation Mode from the Options Menu and the Architecture Command.

The bar on the top of the screen is the global menu.

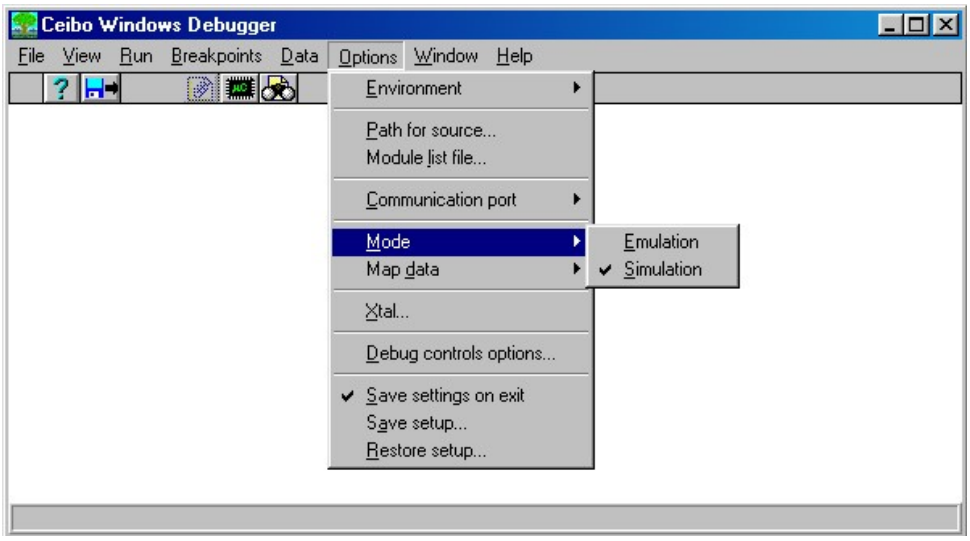


FIGURE 4.1: ***Simulation Mode***

4.4. Accessing the Global Menu

FIGURE 4.2: ***Global Menus***

1. The Global menu commands may be activated by simultaneously pressing the ALT and the command first letter keys.

2. After invoking the program, press Alt-V to open the View command. Select CPU and observe the new window added to the screen. The CPU window becomes the active window.

3. Open the Port Windows from the Target Command in the View Menu.

4. Press CTRL-F6 several times to select a different active window.

4.5. Accessing the Local Menu

1. The Local menu command may be accessed by pressing Alt-F10 or clicking the right button.

2. A direct access to a local menu command is possible by holding down the Ctrl key and pressing the letter that identifies the command.

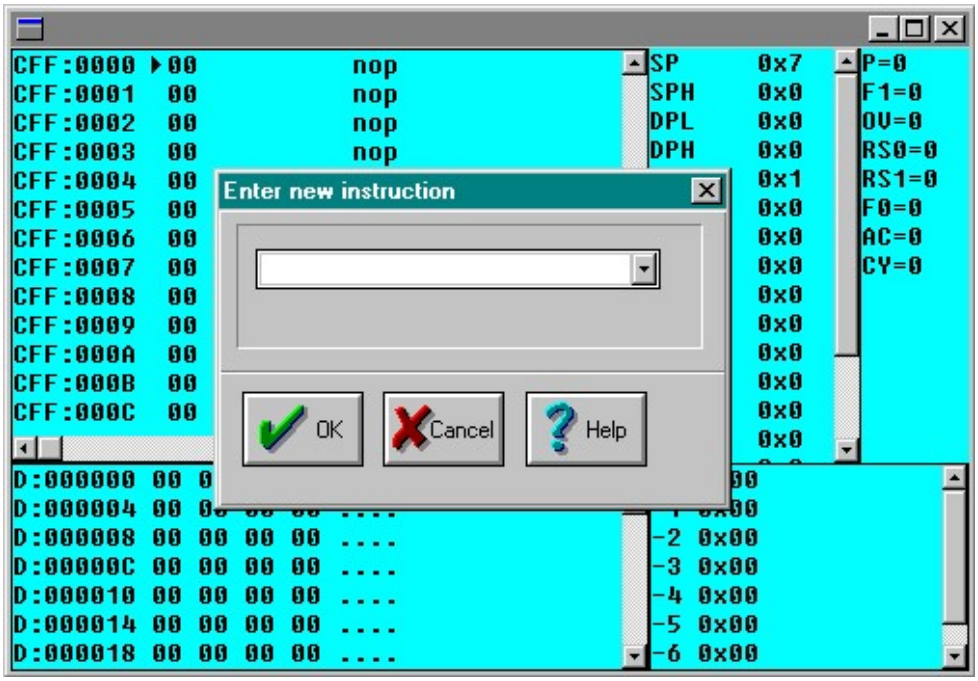


FIGURE 4.3: ***CPU Assembly Command***

3. Select the CPU window and check its Local Menu. The default is Assemble, meaning that you can enter any assembly instruction directly.

4. Move the cursor to any line and type MOV R0,#3 directly. Observe how the code has been changed.

4.6. Adding Watches

Open the Watches Window by selecting the View Menu and the Watch command.

Press the <INS> key or click the right button while the cursor points to somewhere inside the Watches Window. You may also add a watch by clicking the Watches button on the toolbar.

Type P1 and press the Enter key. Then, Port 1 will be added to the Watches Window.

Note the your entry is case sensitive and P1 is not the same as p1.

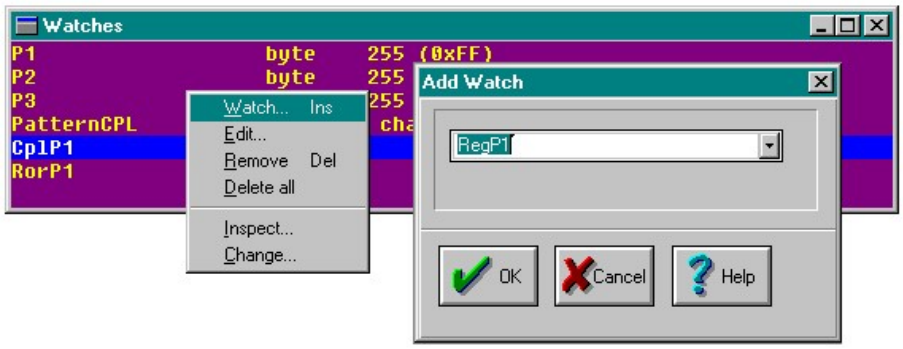


FIGURE 4.4: ***Adding Watches***

4.7. Changing Watches

1. If you want to change the Port value, invoke the Local menu again and select the Change command.

   A selection may be done either by moving the arrows until the Change command is highlighted or by pressing the C key.

   Type 55 and press the Enter key. Observe that the Watches Window has an updated value for Port 1.

2. Click the OK button.

3. You can also change the watches by positioning the cursor on the variable and then pressing Ctrl-C.

4. The Language Command in the Options and Environment Menus determines the base and syntax of your entries. For example, if you choose C Language, 55 is a decimal value and 0x55 is a hexadecimal number. In case the Assembler is selected, you should type 55h to enter the same hexadecimal value. The syntax of your inputs is explained in the next chapter.

The Integer Display Format Command in the Options and Environment Menus defines the base of the display in the Watches Window. You can select hexadecimal or decimal display of your variables.

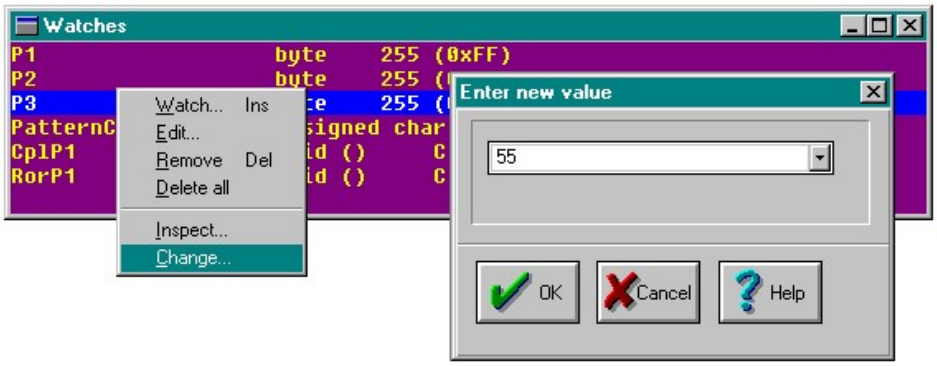


FIGURE 4.5: ***Changing Watches***

4.8. Watching Memory Spaces

1. You can add to the Watches Window any memory space as a succession of values. That may be achieved by specifying the type, initial address and length.

2. Add to the Watches Window the first 10 bytes of the RAM by typing D:0,10.

3. Add to the Watches 10 bytes of the code memory. Your entry may be C:1000,10.

4. Display 3 bits of the Bit addressable space. Type B:0,3.

5. Add to the Watches Window any SFR by entering the absolute address. Your entry may be P:90h or P:0x90. You may also type 0x90,2 to display Port 1 and the following SFR (address 90H and 91H). 0x90 is 90H if you selected C language in the Language Command of the Options Menu. If your selection is Assembler, just type P:90H,2.

4.9. Displaying a Memory Space

1. Open the Data Window by selecting the Memory Space Command in the View Menu.

2. Change the contents of this memory. Click the right button and select the Change Command.

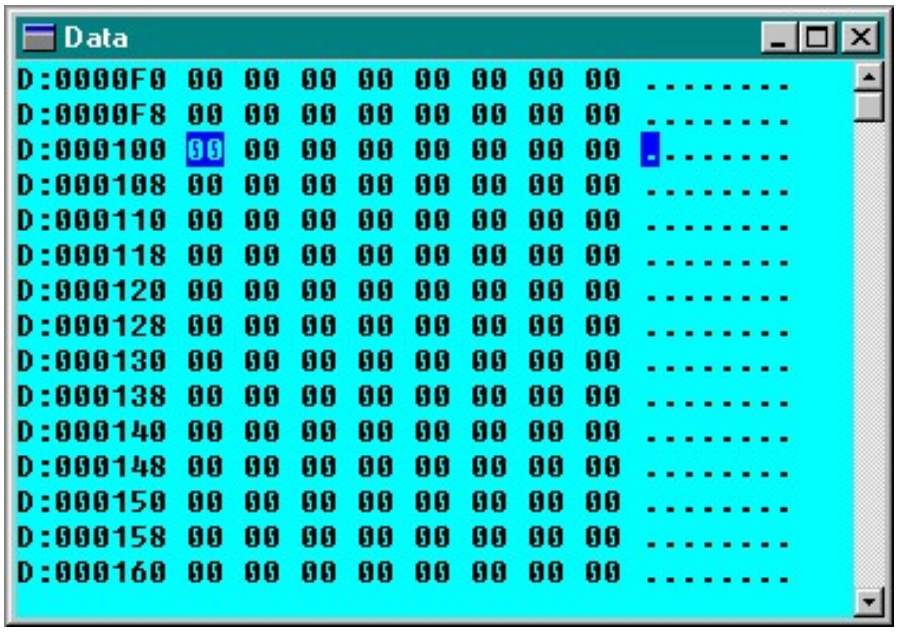


FIGURE 4.6: ***Data Window***

3. Enter successive new values separated by spaces or commas: 11,22,33,44.

4. Check the changes in the Data Window.

4.10. Changing the Windows

1. Press Alt-W to select the Window menu, that permits changing the windows. Try all the options given in this menu.

2. A window may be resized by moving the borders with the left button.

3. The arrows surrounding the window borders can be moved to position the desired information on the screen.

4.11. Loading a File

1. Press Alt-F to activate the File menu.

2. Set the cursor to highlight the Load option and press the Enter key.

3. Select the CTESTS.ABS sample file to load code and symbols.

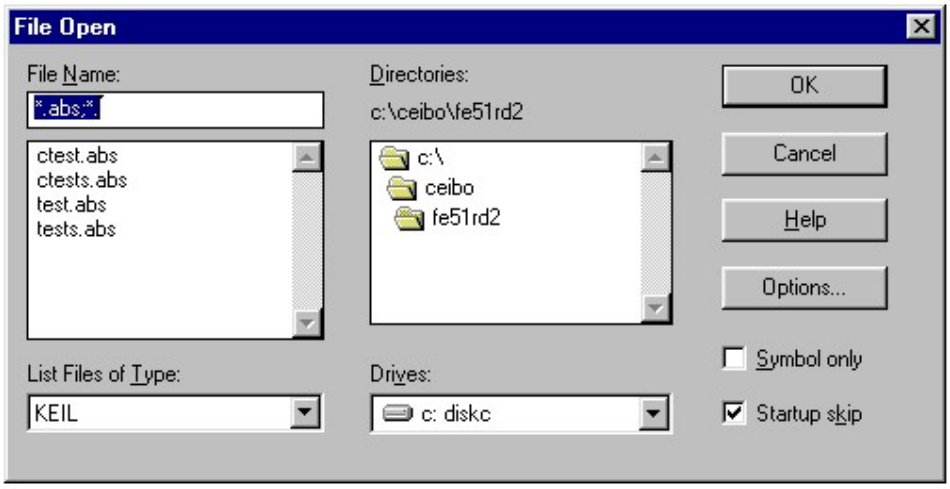


FIGURE 4.7: ***Loading a File***

4. Select *Keil* in *List Files of Type.* Note that it is very important to define the List Files of Type to get the proper symbol information. CTEST.ABS and CTESTS.ABS are Keil files. You may also load a Hex file without symbol information, but in that case the symbolic debugger will not function. After successfully loading the CTESTS.ABS file, the Module and Watches windows will be opened. If you try with a different file with incomplete debug information, the CPU window will be opened instead of the Module and Watches windows.

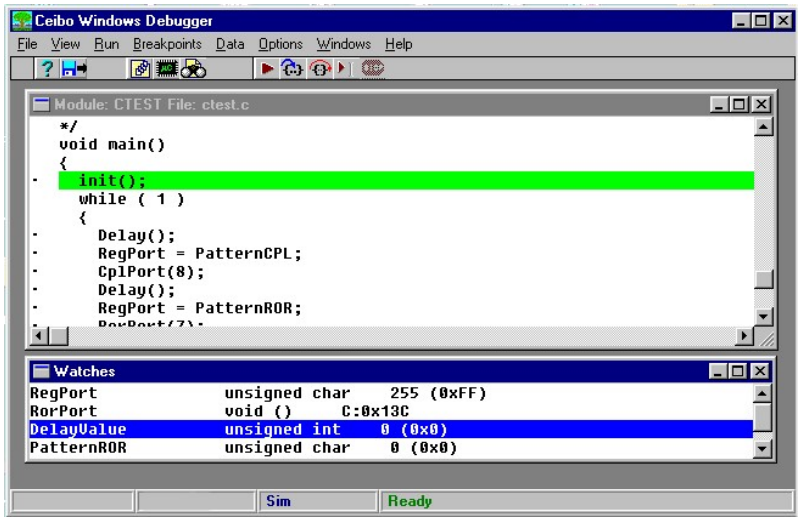


FIGURE 4.8: ***Default Screen***

4.12. Capturing Watches

You can capture the name of a variable and include it in the Watches Window.

1. Open the Module Window with your source code.

2. Position the cursor on the variable name in any part of your source code.

   Press Ctrl-W or use the right button to include the variable in the Watches Window.

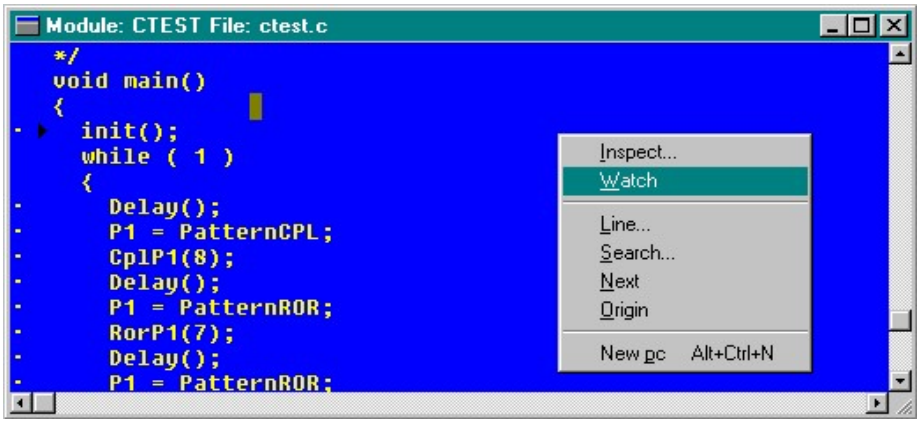


FIGURE 4.9: ***Capturing Watches***

4.13. Debugging the Program

1. Position the cursor on the Reg_P1 variable and click the left button. Click the right button and select the Watch command or press Ctrl-W to add the variable to the Watches Window.

2. Move and resize the three windows according to your convenience.

3. Open the CPU window.

4. Select the RUN Menu and execute a Program Reset. That can be done directly by pressing Ctrl-F2.

5. Execute a few assembly steps. These are available from the Run menu and the command name is Instruction Trace. Press the Alt-F7 keys several times.

6. Execute the program. Press the F9 key.

7. Halt the program execution by pressing Ctrl-Break.

8. Display the executed instructions from the View Menu, using the Trace Buffer command.

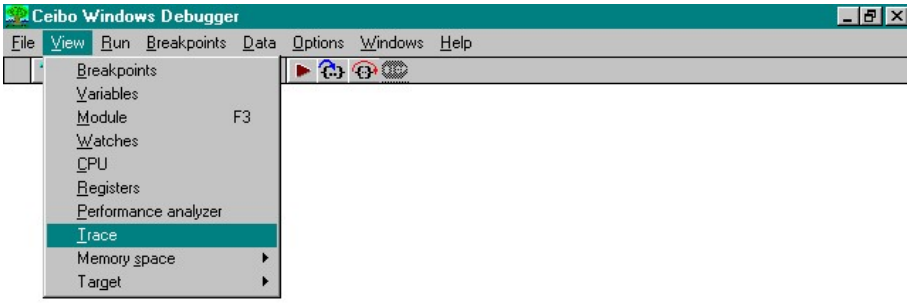


FIGURE 4.10: ***View Menu***

9. Set a Breakpoint by moving the cursor in the Module window and pressing the F2 key. Execute the program by pressing the F9 key.

10. Execute a high-level language step by pressing the F7 key. Repeat that operation several times. Use F8 when you want to step over function calls (the program will execute the function and then stop).

11. Continue the high-level-language stepping by pressing F7. Note that the Modules are changed in accordance with the actual program counter value.

CHAPTER 5

Windows Menus and Commands

CHAPTER 5

Windows Menus and Commands

5.1. Introduction

The previous chapter gave you a general approach to the Windows Debugger menus and commands. You will find a detailed description of the menus and their related commands in the following paragraphs. ***As the debugger is general for all the 8051 derivatives, some specific functions may not apply to all the derivatives.***

5.2. File Menu

The File menu options deal with operations external to the Ceibo Debugger, such as loading programs for debugging, and leaving the application.

The available commands are: Load, Get Info, New and Exit.

This menu also provides a list of the last opened files, so you may load a file just by clicking on the desired file name.

Load

The Load command loads a program for debugging from a disk. You may select the directory and the file name to be loaded, as well as the format of the file to achieve complete debug information compatibility.

The supported file formats are: Intel Hex, OMF51, Keil, IAR-UBROF, Tasking-IEEE695 and others.

Call Ceibo for additional format support. Utility programs and software updates may be released in the future to support new compilers with different formats.

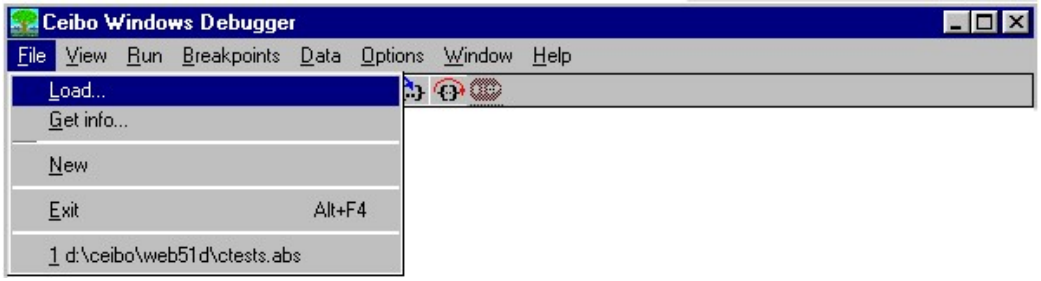


FIGURE 5.1: ***File Menu***

From the File Menu you can specify the Symbol Only option, thus loading only the symbol information in the file for debugging ROM applications. Code is not loaded in this case.

The Startup Skip option is used to run the program automatically until the first line of your main program is reached. If selected the program will run from 0000h to the Main label while debugging C.

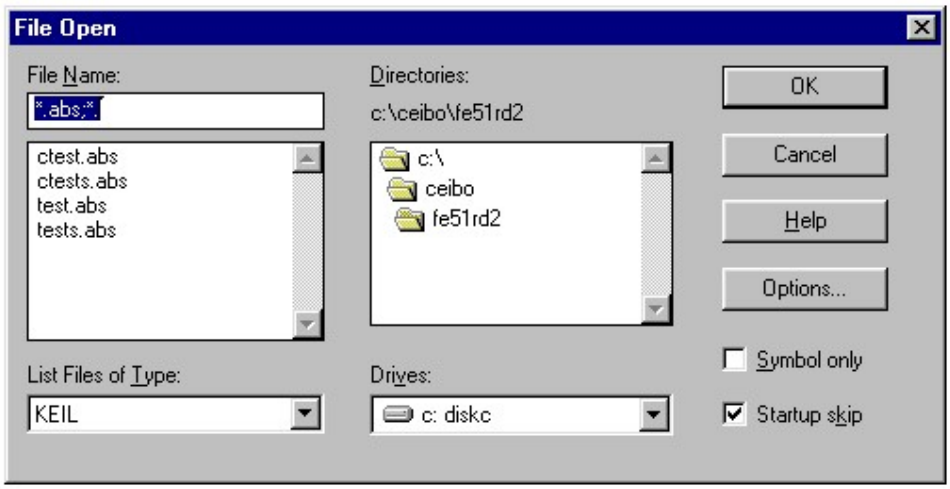


FIGURE 5.2: ***File Open Dialog***

The Options button accesses the Set Options Dialog box, which lists the list of files that may be loaded automatically for special hardware support.

Get Info

The Get Info command displays a window showing the current state of your computer resources software and system versions.

New

This command deletes all the symbol information loaded by the Load Command, thus clearing symbols and code.

Exit

The Exit command terminates the debugging session. The hot key Alt-F4 can also be used to leave the debugger. The Windows Alt-Tab key sequence may also be used to leave temporarily the session without closing it.

5.3. View Menu

The View menu commands open windows that display different aspects of the program being debugged. The available commands are: Breakpoints, Variables, Module, Watch, CPU, Registers, Performance Analyzer, Trace, Memory Space and Target.

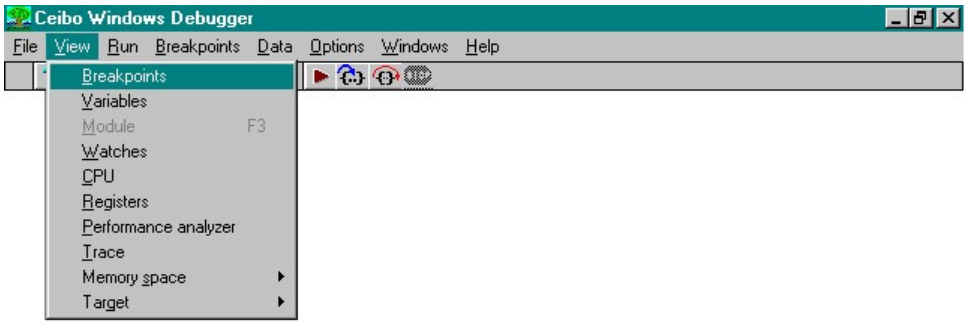


FIGURE 5.3: ***View Menu***

Breakpoints

The Breakpoints menu commands let breakpoints be displayed, set and cleared. The different options may be accessed through the Local menu of this window; type Alt-F10 or click the right button.

The different breakpoint options available from the Local menu are:

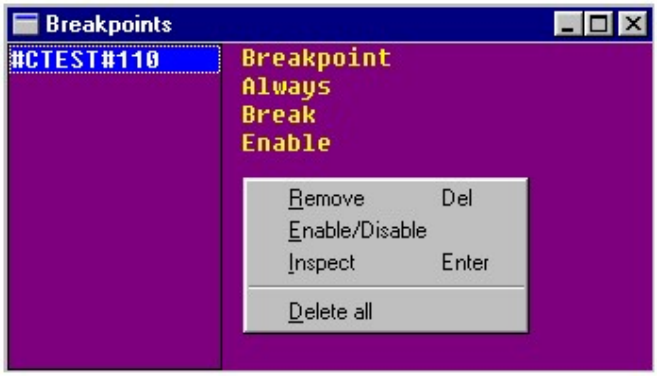


FIGURE 5.4: ***Breakpoints Local Menu***

Remove: Use this command to cancel a selected breakpoint.

Enable/Disable: Sets the status of the selected breakpoint without removing it or affecting its definition.

Inspect: Displays the information regarding the code location, such as module name and CPU address.

Delete All: Removes all the defined breakpoints.

Variables

The Variables command opens a Variables window displaying a list of the program global (or public) and local symbols, and their locations. The window is split into two sections. The upper area shows the global symbols while the lower one displays the local symbols.

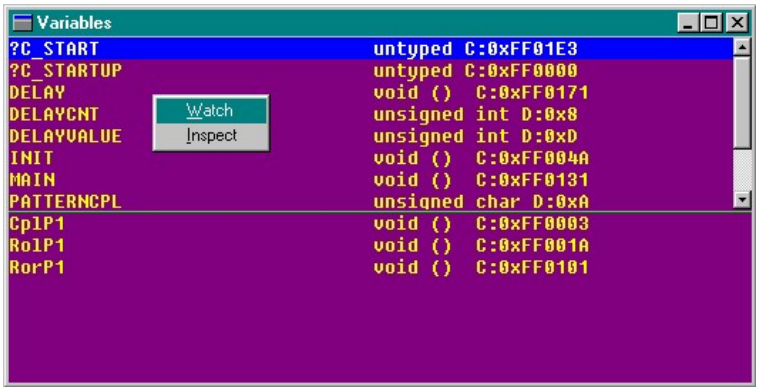


FIGURE 5.5: ***Variables Window***

The Local menu of the Variables window has two useful commands:

Watch**:** Adds a variable to the Watches window. Click first the desired variable to highlight it and then you may include it to the Watches Window by using the Local Menu.

Inspect**:** Displays all the available information about the highlighted variable.

Module

The Module command opens a Module window showing the list file of the selected module.

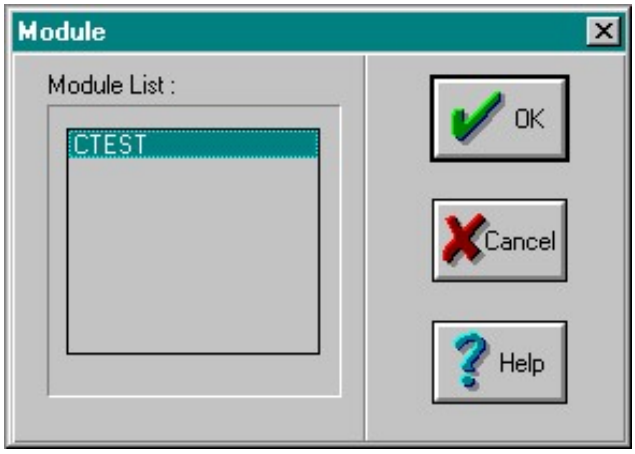


FIGURE 5.6: ***Module List***

A module to be viewed can be picked from a list of the current program available modules. Only modules which have a corresponding list file will appear in this pick list.

This command will be disabled if no debug information has been loaded.

F3 or the button in the Speed Bar are the hot keys for this command.

The Module window title indicates the module name currently being viewed.

List and source files may be in different directories and the software should know about it. The Options menu gives the possibility to define the path for the files and filenames.

From the Module window you may open the Local menu with the following options:

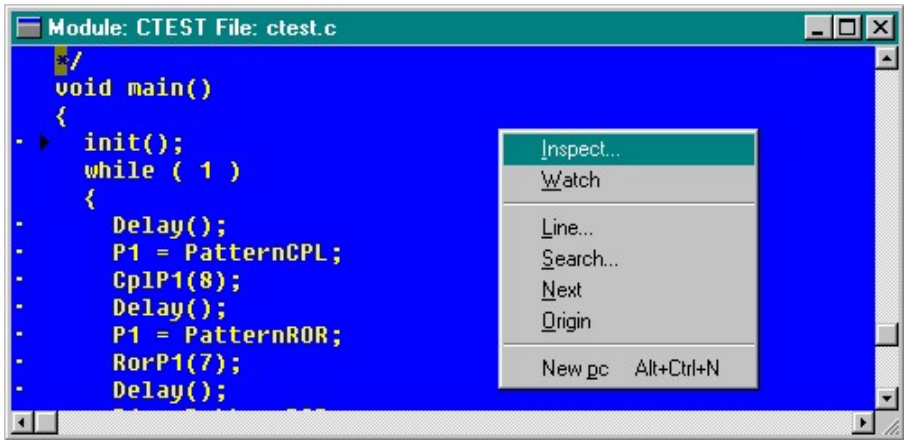


FIGURE 5.7: ***Module Local Menu***

Inspect: This command provides all the available information on the selected variable.

Watch: Use this command to add the variable pointed by the cursor to the Watches window.

Line: Specifies the line number of the source file to be displayed in the Module window.

Search: The Search command may be used to locate any string in the Module window.

Next: This command searches the next location of the string specified by the Search command.

Origin: The Origin command refreshes the screen to the current position of the program counter.

New PC: This command sets the program counter to the current position of the cursor. Stack operations and variable values may be affected by this redefinition of the program counter.

Watches

The Watches command opens a Watches window showing the value of variables specified from the Data menu. Different Local menus may also be used to enter new variables.

These Local menus are available in the Watches, Module and Variables windows.

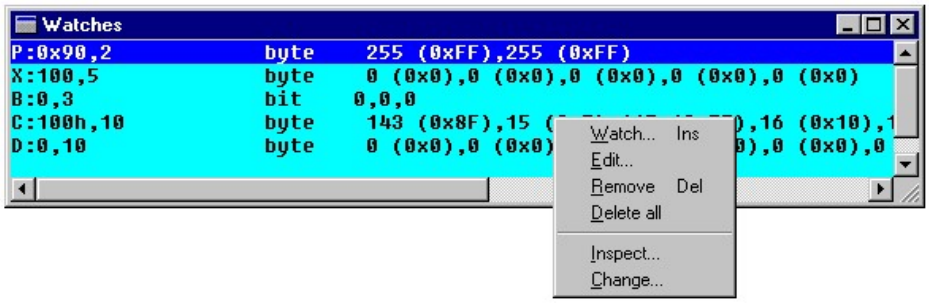


FIGURE 5.8: ***Watches Local Menu***

The Local menu of the Watches window has the following options:

Watch: Use this command to add a new variable to the window. You may enter any predefined symbol like P1, P1.0, R3, etc. or make an absolute reference using the first letter to define the variable type followed by a colon and the address. Furthermore, absolute references may be expressed with a length.

The syntax is:

absolute_reference:address,length

Absolute references are D for the on-chip RAM, X, F for ports and any SFR (special function registers), C for code memory and B for bit memory . Some examples are:

D:100,5

B:0x80,5

B:P1.0,8

F:0x431

Addresses are entered using the syntax of the selected language in the Options menu. Length is always decimal.

Edit: This command is used to modify the selected watch name.

Remove: The Remove command deletes the selected variable from the Watches window.

Delete All: This command clears the Watches window.

Inspect: The Inspect command may be selected to get the address information of the selected variable.

Change: The Change command permits the modification of variable contents. Your entries use the syntax of the selected language in the Options menu. For example, if you select C and you want to enter 9Fh, just type 0x9F. The 9FH entry is recognized if the selected language is ASM. In case that you are using Pascal, enter $9F. A string may be changed by entering values separated by commas or blank spaces.

CPU

The CPU command opens a CPU window displaying the disassembled instructions of your program, the Stack, the internal Registers and any memory space according to the Local menu selection. *If you get only the disassembly instructions, select* ***restore*** *from the local menu to have the other items.*

An instruction may be displayed with symbol information, and mixed with source code lines. You may also patch-up code using the built-in assembler.

The CPU window is divided into five sections: disassembled code, registers, status bits, stack and memory. Each of them has its own Local menu.

From the disassemble section the Local menu enables the following commands:

Go to: This command is used to set the cursor to any desired address.

Origin: The Origin command refreshes the screen to the current position of the program counter.

Toggle Source: This command toggles between pure disassembled code and code embedded with source line numbers.

Assemble: The Assemble command is used to on-line modify your code. The syntax of this command is fully explained in the On-Line Assembler chapter.

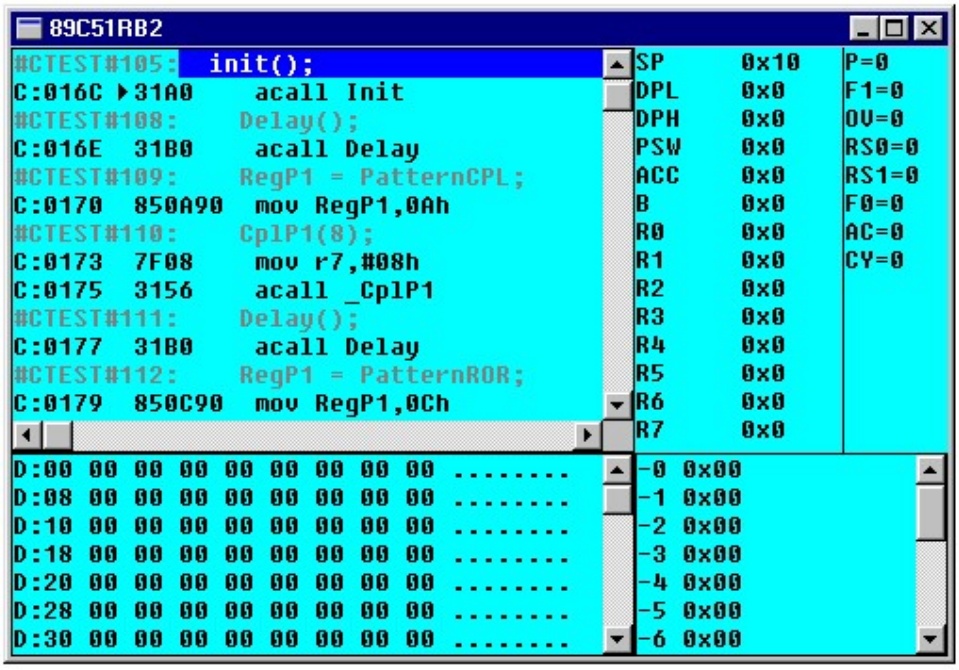


FIGURE 5.9: ***CPU Window***

New PC: This command sets the program counter to the current position of the cursor. Stack operations and variable values may be affected by this redefinition of the program counter.

Print to File: Use this command to save any portion of your code in ASCII format to a disk file.

The registers and status bits sections in the CPU window are similar to the Register window. The Local menu is explained in the description of the Registers window.

STACK

The stack section of the CPU window shows the stack bytes with the associated addresses relative to the stack pointer. For example, the address -2 means stack pointer contents minus 2. The Local menu of this section has the following commands:

Go to: This command is used to set the cursor to any stack position.

Origin: The Origin command refreshes the screen to the current position of the program counter.

Change: You can use this command to modify the stack contents.

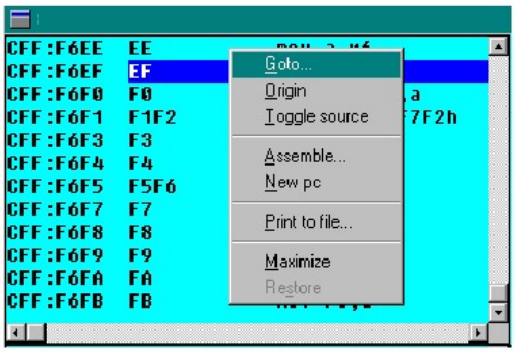


FIGURE 5.10: ***Disassembly Local Menu***

FIGURE 5.11: ***Stack Local Menu***

The memory section of the CPU window has the same Local menu as the Memory Space windows in the View menu, with the addition of the capability to change the type of memory displayed in the CPU window: external data, on-chip RAM or code. The explanation of the Local menu commands is given in the Memory Space windows description.

Registers

The Registers command opens a Registers window where the flags and current CPU registers state appear.

The Registers are displayed according to the selected microcontroller in the Options menu, Architecture and Chip commands.

The Local menu of this window has the following commands:

Toggle: This command is available for the register bits and clears or sets the selected bit state.

Increment: Adds one to the current value of the selected register byte.

Decrement: Subtracts one from the current value of the selected register byte.

Zero: Clears the selected register byte.

Read: The Read command forces the reading of the specified register.

Change: Use this command to modify the selected register.

Update: Reads all the registers to update the window.

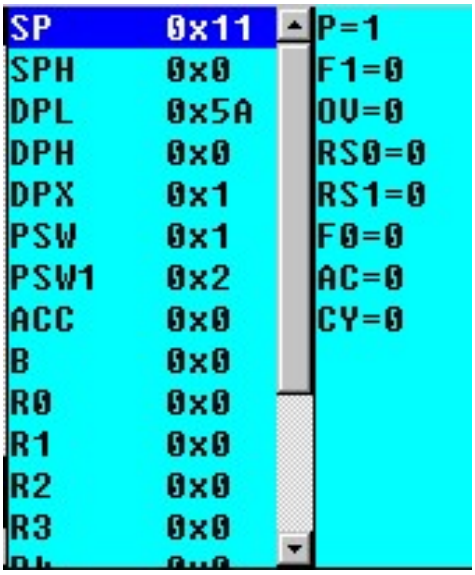


FIGURE 5.12: ***Registers Window***

Performance Analyzer

This command processes the information recorded in the trace buffer and provides a graphics representation of the executed modules and the percentage of time spent in each of them.

The local menu of this window may be used to get more useful information about your software performance.

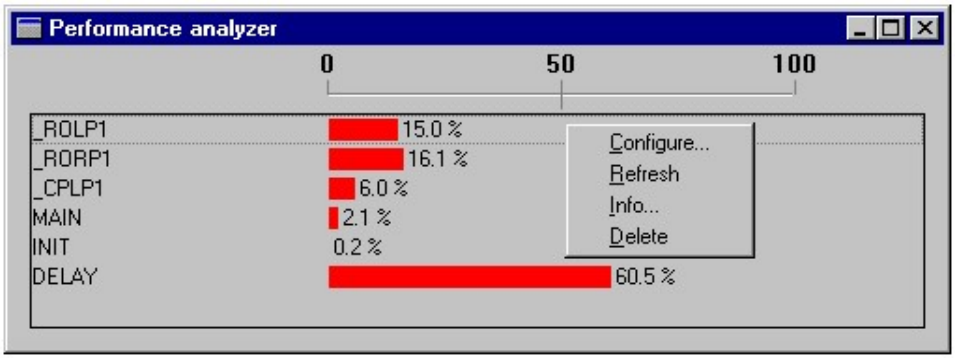


FIGURE 5.13: ***Performance Analyzer***

Configure: A dialog box allows the selection of the functions belonging to the module that you may want to include in the performance analysis. Just select the desired module and then the respective functions. Furthermore, if you desire to define your own address range, click the [*user defined*...] line and the Add button. Then, you will be able to select a customized address range to analyze.

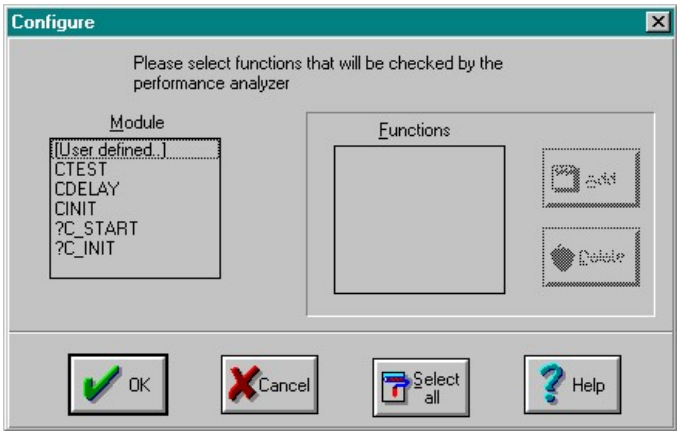


FIGURE 5.14: ***User Defined Dialog Box***

Refresh: This command reads again the trace buffer and recalculates the performance of the modules.

Info: Displays information about the module selected by the cursor in the window.

Trace

The Trace menu allows the current Trace window to be opened, as well as viewing the trace status.

Trace functions are enabled in simulation mode only.

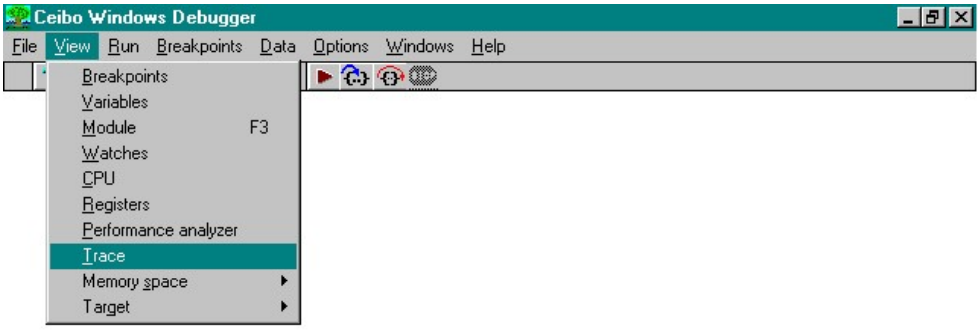


FIGURE 5.15: ***Trace Menu***

The Trace window allows the current trace buffer to be viewed, display different formats for the trace selected, filter data from the trace display and search data patterns in the buffer.

***Trace Dump*:** The Local menu of the Trace Dump windows provides many useful functions to setup the operation of the trace and manipulate the accumulated information.

These functions are:

Go to: Sets the cursor to the specified frame number.

Origin: Displays the window starting from the first recorded frame.

Inspect: You may display additional information about the variables recorded in the trace buffer.

Display Mode: A selection of source code, disassembled instruction or mixed source and disassembled code is available.

Time Stamps: You can display the absolute cycles (accumulated number of cycles), absolute time (accumulated time according to the XTAL selection in the Options menu) and relative cycles (number of cycles of each frame).

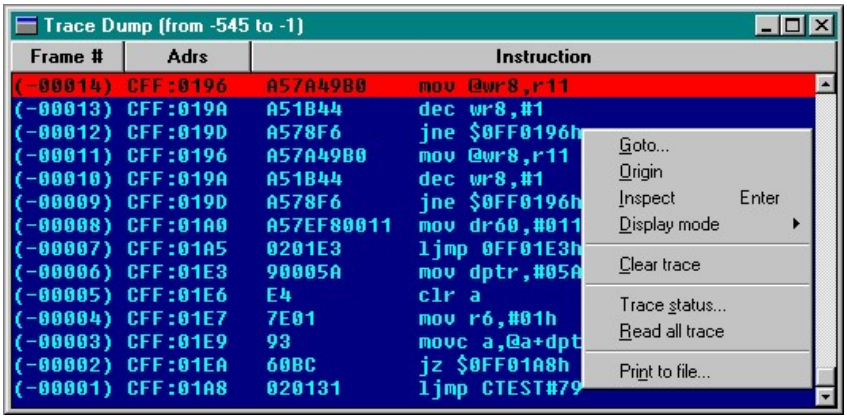


FIGURE 5.16: ***Trace Dump***

Filters: Defines the trace filters of the displayed data. You may specify which instructions or sequences are of your interest.

Clear Trace: Deletes all the accumulated data.

Trace Status: This command is the same as available from the Trace Menu and it is explained separately.

Read all Trace: Gets all the recorded data from the trace buffer for performance analysis, absolute time stamp calculations, etc.

Print to File: Saves the trace buffer in a disk file.

Trace Status: The Trace Status command displays a window showing the current state of the trace. This includes trace state (recording/halted), trace overflow indication and number of frames currently in the trace buffer.

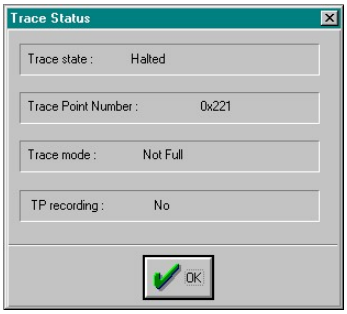


FIGURE 5.17: ***Trace Status***

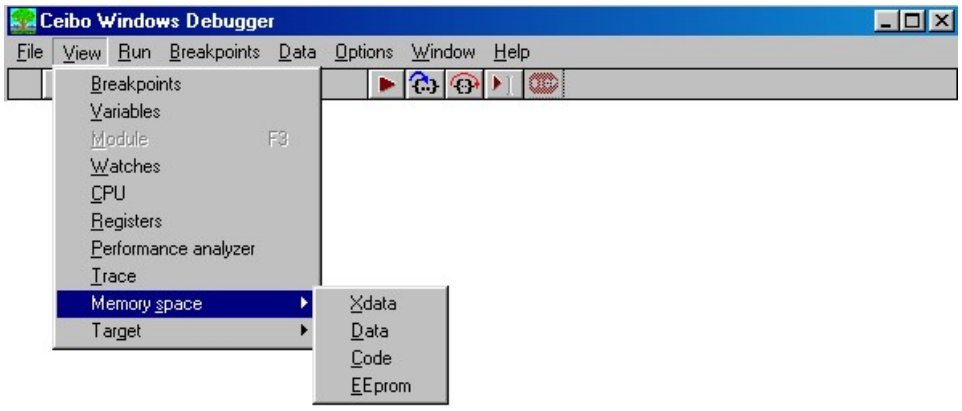


FIGURE 5.18: ***Memory Local Menu***

Memory Space

The Memory Space command opens up to three windows where the specified areas of memory are displayed. The data can be viewed as raw hex bytes with their corresponding ASCII representation. From this command you may open a

window with internal data memory (Data), external data memory (XData), EEPROM (also XData) or the code memory (Code).

Important: ***XData RAM*** belongs to the on-chip RAM or external RAM according to the *Options/Map data* selection. Regarding the ***EEPROM*** space, the *EETIM* SFR *must be set* to write to this space; this is done automatically by defining the frequency in *Options/Xtal* dialog.

Any of the three memory spaces has a Local Menu with the following commands.

Go to: This command is used to set the cursor to any desired address.

Search: Use this command to find a data value in the window.

Next: Locates the cursor in the next finding of the data value specified by the Search command.

Change: This command modifies the contents of the selected data memory. These data bytes may be written sequentially with blank spaces or commas between them, if you need to specify a string, i.e. 11, 22, 33, 44, 55, where the base is specified according to the selected language in the Options menu.

Block: The Block command is very useful to manipulate the data. You can clear all the data, set it to any value, move portions of the memory space, read a file and put its contents into the memory and write any portion of the memory to a disk file.

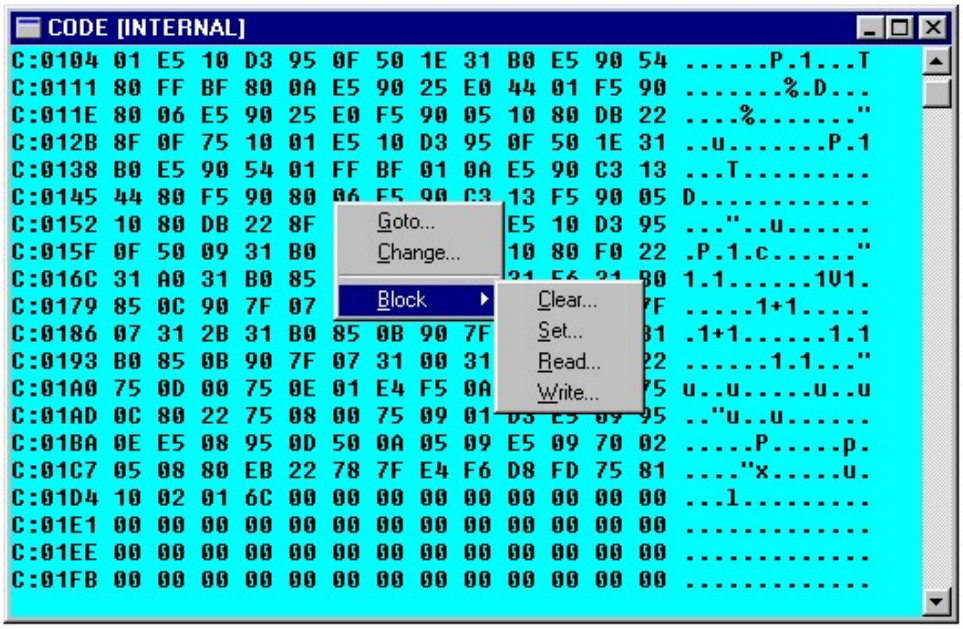


FIGURE 5.19: ***Block Menu***

Target

This command opens different windows that are related to the target microcontroller emulated or simulated by the debugger.

The available windows that may be opened are Port, Interrupt, Serial, Miscellaneous, Timer, Power, PWM, Watchdog and others depending on the selected chip type.

From any of there windows you may use the Local menu to carry out the same functions described in the Registers window: Increment, Decrement, Zero, Read, Change and Update.

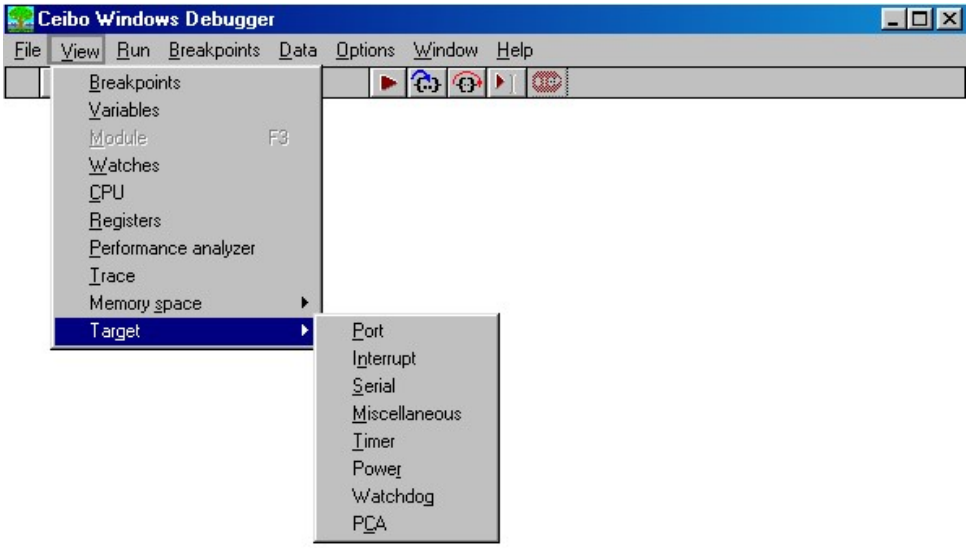


FIGURE 5.20: ***Target Menu***

5.4. Run Menu

The Run menu commands execute the program being debugged. The following options are available: Run, Execute Forever, Go to Cursor, Trace Into, Execute to, Step Over, Animate, Instruction Trace, Continuous Run, Halt and Program Reset.

Run

The Run command executes the program continuously until either the program is halted with the Halt key, or a breakpoint is reached. The F9 key is the hot key that executes this command.

Execute Forever

The Execute Forever command executes the program being debugged continuously until halted with the Halt key.

Any breakpoints set are temporarily deleted, until halting program execution.

This is useful for temporarily running the program without interruptions, and without having to delete all the previously set breakpoints.

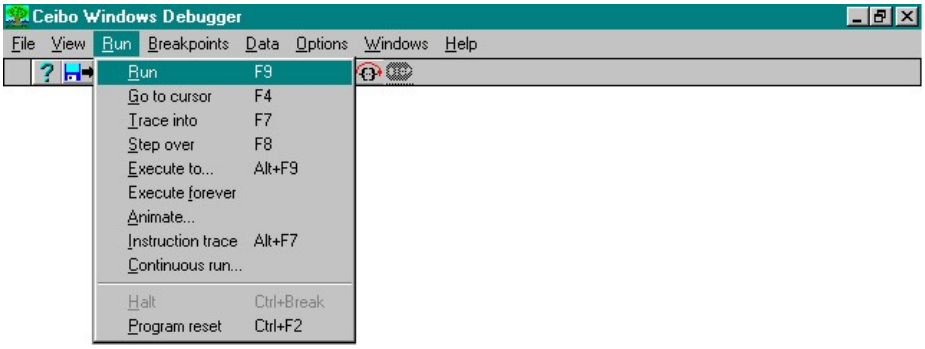


Figure 5.21: *Run Menu*

Go to Cursor

The Go to Cursor command executes the program until the instruction or source line pointed by the cursor is reached.

The current window must be a CPU window or a Module window in order to determine which location to execute.

The F4 key is the hot key that executes this command.

Trace Into

The Trace Into command executes a high-level language source line or a single machine instruction.

If your code module does not include debug information, the Trace into command executes a single machine instruction that may be observed in the CPU window.

In case you have a code module with debug information, this command executes a complete line step.

The F7 key is the hot key that executes this command.

Execute To

The Execute To command executes the program and stops the program at a specific location. The address can be entered in any of the valid address formats.

Alt-F9 is the hot key that executes this command.

Step Over

The Step over command executes a high-level language source line or a single machine instruction.

If your code module does not include debug information, the Step over command executes a single machine instruction that may be observed in the CPU window. The only exception occurs when your code has a CALL instruction; then the program execution continues until it returns to the line following the CALL instruction. If the program does not return to the next line it will keep running.

The F8 key is the hot key that executes this command.

Animate

The Animate command is a self-repeating Trace into command.

The source lines or instructions are continuously executed until any key is pressed. CEIBO Debugger shows changes reflecting the current program state between each step; this allows you to watch the program control flow.

The Animate Speed dialog box prompts you for the rate at which animated steps will be executed.

Figure 5.22: ***Animate Speed***

Instruction Trace

The Instruction Trace command executes a single machine instruction.

Use this command for the following: trace into a function in a module that was not compiled with debug information or watch execution of instructions which belong to a source line.

Alt-F7 is the hot key for this command.

Continuous Run

This command executes the program and breaks automatically to refresh all the windows according to the halt intervals defined in the dialog box.

Halt

The Halt command stops the currently running program.

This command will be disabled if there is no program currently running. Ctrl-Break is the hot key for this command.

Program Reset

The Program Reset command issues a hardware reset to the emulated Microcontroller, causing all registers and on chip peripherals to return to their reset state. If you loaded a program with the Startup Skip option, the program will execute the startup code as well.

Ctrl-F2 is the hot key for this command.

5.5. Breakpoints Menu

The Breakpoints menu commands let breakpoints be set and cleared.

The following commands are available: Toggle and Delete All.

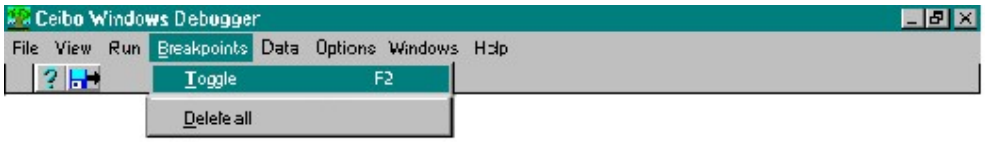


FIGURE 5.23: ***Breakpoints Menu***

Toggle

The Toggle command sets or clears a breakpoint at whatever address the cursor is pointing to in the CPU window or in the Module window.

The program will stop each time it reaches a line where a breakpoint has been set, or a global or hardware breakpoint occurs.

The F2 key is the hot key that executes this command.

Delete All

The Delete All command deletes all the breakpoints from the program. This include software, hardware, or expression true global breakpoints.

This command is used when debugging is to be continued without stopping the program at any previously set breakpoint location.

5.6. Data Menu

The Data menu commands permit the examination of variables and symbols in the program.

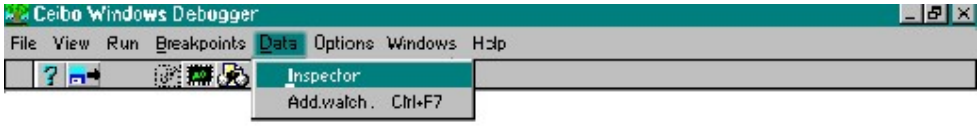


FIGURE 5.24: ***Data Menu***

The following commands are available: Inspector and Add Watch.

Inspector

The Inspect command prompts you for a symbol name to be inspected, if the specified symbol is found in the current program debug information, the relevant information on this symbol is displayed, this includes type, memory space reference and location.

You cannot change the inspected variable value from this command. Use the Watch Change command from the Watches window for modifying variable values.

Add watch

The Add Watch command prompts you for a variable name, or a memory space reference, and places it on the watch list displayed in the Watches window.

5.7. Options Menu

The Options menu allows adjustment of some options that have a global effect on the conduct of the Windows Debugger, and the remote emulator system.

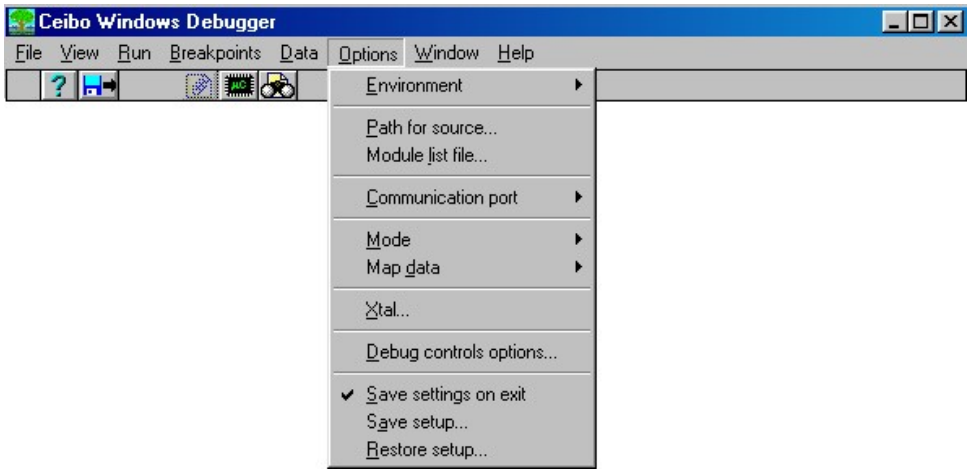


FIGURE 5.25: ***Options Menu***

The following are the available options: Environment, Path for Source, Module List File, Communication Port, Communication Baud, Mode, Map Data, Xtal, Debug Control Options, Save Settings on Exit, Save Setup and Restore Setup.

Environment

The Environment option allows you to control the general environment parameters of Windows Debugger.

The following options are available: Language, Integer Display Format and Beep on Breakpoint.

Language: The Language command determines the base and syntax of your entries. For example, if you choose C Language, 55 is a decimal value and 0x55 is a hexadecimal number. In case the Assembler is selected, you should type 55h to enter the same hexadecimal value.

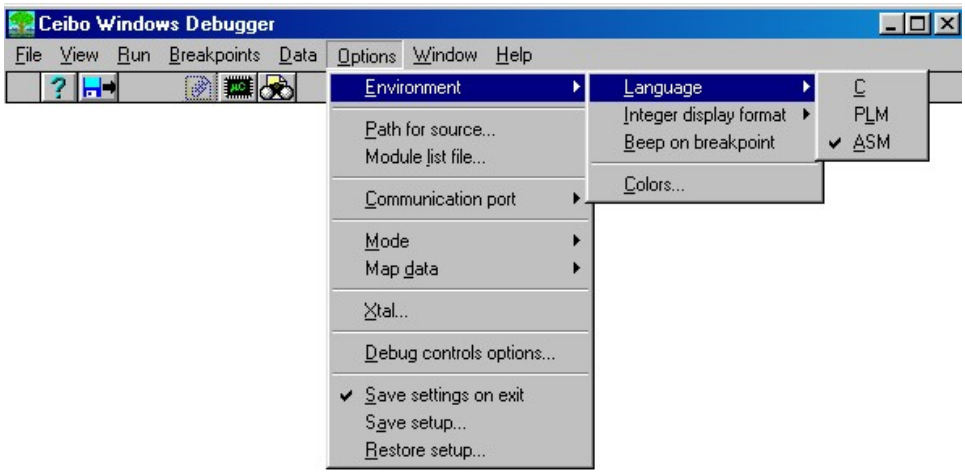


FIGURE 5.26: ***Environment Options***

Integer Display Format: The Integer Display Format command defines the base of the display in the Watches Window. You can select hexadecimal, decimal or both bases for displaying your variables.

Beep on Breakpoint: This option toggles On and Off the beep sound generation whenever a breakpoint is reached.

Colors: With this command you may select your preferred color for each window, both background and foreground colors as well as text types.

Path for Source

The Path for Source List option defines the directory trees in which the debugger will search for the source list files of your program.

The source list files are first searched for in the directories specified by this command, followed by the current directory and finally in the directory from which the program was loaded.

The syntax for this command is: *directory1;directory2;...,* thus allowing definition of several paths.

Module List File

The Module List File option is used to set the list file name associated with each module of the program being debugged.

The Module command will only allow access to modules with valid list files found in your disk.

Use the File Find button to redefine the list file names and paths. First click on a module name to select it. Then, use the File Find button to open the Module List Files Dialog Box.

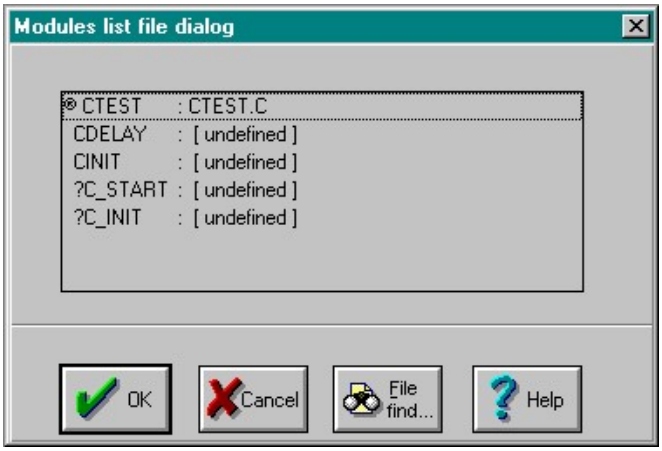


FIGURE 5.27: ***Module List Dialog Box***

Whenever loading an ASM program for the first time, the default setting will be the module_name.LST. It is therefore recommended to keep the module name equal to the list file name, as it will prevent you from having to configure this setting. Otherwise you will need to assign the list file name for each module after loading a new program to debug for the first time.

Communication Port

The Communication port option allows selection of the host PC COM port number to be used for communication with the remote emulator system.

Make sure that there are no resident programs hooked up to the selected COM port, when being used for remote emulation interface.

Mode

The Mode option allows you to select the operation mode of Windows Debugger.

The following options are available: Emulation and Simulation.

Emulation: The Emulation Mode option sets the Debugger to operate in full emulation mode. In this mode your program will be executed in real time on the remote emulator system.

This mode requires the use of FE-51RD2 connected to your PC. Communication error will result if the emulator is not found on the selected COM port.

Simulation: The Simulation Mode option sets the Windows Debugger to operate in full simulation mode. In this mode your program instruction execution will be simulated by the debugger built-in simulator. This mode can be operated without connecting any remote emulator system, thus allowing software debugging to be done while the emulator is used for hardware debugging, or other projects.

This is ***not a real-time mode***; only basic functions are supported and not all SFRs belonging to particular derivatives. ***Set the system to emulation mode***, while using the debugger with FE-51RD2.

Map Data

The Map Data command allows you to define the data memory as belonging to the on-chip RAM or to your target hardware. Data memory is accessed by MOVX instructions.

Xtal

Use this command to tell the software which crystal frequency you are using. With this value the system will be able to calculate time events as displayed in the Trace window. It also sets EETIM SFR accordingly to access the on-chip EEPROM. ***This command does not set frequency***, which is defined by hardware (crystal on the probe).

Debug Controls

Debug Controls dialog is used to define the following:

Reload Setting**:** you may define whether the last loaded code will be automatically reloaded or not. Automatic reloading is possible while powering up the system or when a program reset is executed.

Origin Enable**:** this option specifies where the cursor will point to while stopping the emulation. If the box is checked, that enables the origin and the cursor will point to the actual program counter value in the CPU and Module windows. If it is not checked, those windows will remain in the same state as they have been while running the program.

Save Settings on Exit

Select this command if you want to save the setup while leaving the debugger. This setup will be restored while invoking again the debugger.

Save Setup

The Save Setup command allows you to save the options set in the options and window layouts at any time and with any filename.

Restore Setup

The Restore Setup command allows you to load a configuration file from disk. The configuration file should have been previously saved by using the Save Setup command.

5.8. Windows Menu

The Window menu commands allow various operations on the currently open windows with the following commands: Tile, Cascade, Close all and Restore Standard.

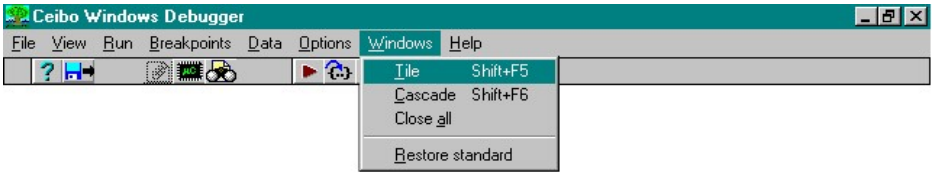


FIGURE 5.28: ***Windows Menu***

5.9. Help Menu

The Help menu commands open a help window for whichever subject will be selected from the menu. This menu offers the following options: Index, Topic Search and About.

FIGURE 5.29: Help Menu

Index

The Index command displays a list of help topics. Not all the help contexts are listed, only the useful subjects and starting points. Shift-F1 is the hot key that executes this command.

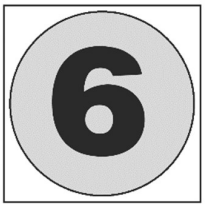
Topic Search

The Topic Search command find specific information about the debugger features and commands. Alt-F1 is the hot key that executes this command.

About

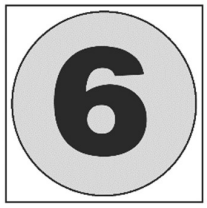
The About command displays the debugger software version.

CHAPTER 6



Real Time Emulation

CHAPTER 6



Real Time Emulation

6.1. Introduction

The Emulation Mode option sets the Debugger to operate in real time using the FE-51RD2 emulator.

As the system uses some of the microcontroller resources in Emulation Mode, you must learn how to use it and how to prepare your code in order to avoid operational problems. Table 6.1 provides a list of used resources and a quick reference about the actions that you have to carry out while working in real-time with FE-51RD2 .

Please read carefully this chapter before using FE-51RD2 in Emulation Mode.

6.2. Emulation Memory

1. ***Address F000h-FFFFh:*** Real time emulation is carried out by downloading your code into the Flash memory of FE-51RD2 . The upper 4KByte of the 64 KByte code memory is automatically loaded with a Monitor program to control the emulation, so do not uses it for your code..

2. ***Address 0000h-0002h:*** Before executing your program in real-time, some areas of your code are changed automatically. The Reset vector content is modified by an LJMP Monitor instruction, so your own code must begin with an LCALL or LJMP instructions. The same usage of the Reset vector does not allow to write JMP or CALL instructions with address 0001h or 0002h as destination.

3. ***Address 0023h-0025h:*** This is the serial interrupt vector. You may use it only for a valid “LJMP to interrupt service routine”. Otherwise it must be left reserved by NOPs.

4. ***Address 0000h-EFFFh:*** Code memory cannot be mapped external and always belongs to the emulator system.

6.3. Stack Pointer

1. ***System Stack:*** The emulator uses 4 bytes of the stack while halting real-time emulation.

2. ***Stack Pointer:*** This pointer must not be set to an address below 07h.

6.4. Breakpoints

Breakpoints in real time emulation are implemented by replacing the user code at the breakpoint location with an LCALL Monitor instruction.

Since an LCALL instruction is used it requires three address locations to be changed. Therefore, setting a breakpoint on a one or two byte instruction may also alter the next one or two consecutive instructions.

The following restrictions apply to breakpoints:

1. ***Breakpoint address:*** The address ***must*** belong to an opcode, otherwise undetermined operation will result.

2. ***Consecutive instructions:*** Setting a breakpoint to a one or two byte instruction will cause alteration of the next one or two consecutive instructions as well. This will be indicated on the screen with a bright red line marking. The original breakpoint instruction will be marked with a red line. If one of these instructions are executed before the originally marked instruction, a NOP will be executed instead of the original opcode. These instructions ***must not*** be executed before the breakpoint instruction because an undetermined operation may result.

3. ***Expression True Global Breakpoints:*** These Breakpoints will force execution to be done in Simulation Mode.

4. ***Reset and Interrupt Vectors:*** Breakpoints cannot be set on the Reset vector (address 0000h to 0002h) or on the Serial Interrupt vector (address (0023h to 0025h).

6.5. Halting the Emulation

The FE-51RD2 uses the UART to Halt the user program while executing it in real-time, so that the state of the program can be examined and/or altered.

If you try to stop your program with the Halt command (Ctrl-Break), and the debugger continues running, this means that the interrupts have been disabled or the timer used to operate the UART is not running.

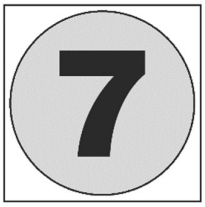
The first rule is not disabling the interrupt. EA bit (IE.7) which must be set as well as the enable interrupt bit of the UART.

The second consideration is to handle the interrupt vector addresses. If you reserve the interrupt for the emulator, the first 3 addresses of the interrupt vector

must be filled with NOPs, so you will be sure that your compiler did not use these addresses for your application code. The debugger will give an interrupt warning message whenever code other than 00h is found on the reserved interrupt vector locations.

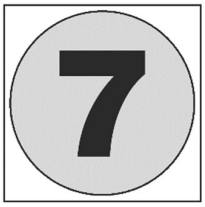
When using shared interrupts, the system replaces the interrupt vector by an LCALL instruction. Therefore, your code must never jump directly to the two address locations following the first address of the interrupt vector.

CHAPTER 7



On-Line Assembler

CHAPTER 7



On-Line Assembler

7.1. Introduction

FE-51RD2 software includes two useful functions: On-line Assembler and Disassembler. Both functions permit the user to translate instructions from and into mnemonics.

The On-line Assembler command assembles a single instruction mnemonic into the code memory.

After writing an instruction mnemonic, press the <ENTER> key If an invalid instruction is entered, a syntax error message is displayed.

7.2. Assembler Syntax

Certain restrictions are placed on the type of instructions that might be entered in this mode.

- Use a valid symbol format. The Debugger requires operands in hexadecimal or symbols.
- An H must follow the hexadecimal operands.
- The comma is used to separate operands.
- A dollar symbol ($) is used to specify absolute jump addresses. If it is omitted, the specified address is relative.

The following are examples of valid expressions:

JBC 12H,$123H

CPL P1.1

AJMP 0H

The Disassembler function allows the user to disassemble memory values, into instruction mnemonics.

This function assumes that the starting address points to the first byte of an instruction and takes second and third bytes if they are necessary to be used as operands.

Stepping down through the CPU window assumes the next instruction is an opcode.

Stepping up in the CPU window may result with erroneous mnemonic display.

7.3. Instruction Set

The following pages detail the syntax of the mnemonics and operands used by the On-line Assembler and Disassembler functions.

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
00	1	NOP	NOP
01	2	AJMP page-addr	AJMP 0010H
02	3	LJMP code-addr	LJMP 00F3H
03	1	RR A	RR A
04	1	INC A	INC A
05	2	INC byte-addr	INC 1FH
06	1	INC @R0	INC @R0
07	1	INC @R1	INC @R1
08	1	INC R0	INC R0
09	1	INC R1	INC R1
0A	1	INC R2	INC R2
0B	1	INC R3	INC R3
0C	1	INC R4	INC R4
0D	1	INC R5	INC R5
0E	1	INC R6	INC R6
0F	1	INC R7	INC R7
10	3	JBC bit-addr, rel-offset	JBC 02H, A0H
11	2	ACALL page-addr	ACALL 0000H
12	3	LCALL code-addr	LCALL 0110H
13	1	RRC A	RRC A
14	1	DEC A	DEC A
15	2	DEC byte-addr	DEC FAH
16	1	DEC @R0	DEC @R0
17	1	DEC @R1	DEC @R1
18	1	DEC R0	DEC R0
19	1	DEC R1	DEC R1
1A	1	DEC R2	DEC R2
1B	1	DEC R3	DEC R3
1C	1	DEC R4	DEC R4
1D	1	DEC R5	DEC R5
1E	1	DEC R6	DEC R6
1F	1	DEC R7	DEC R7

bit	addr	:	00H	- FFH	-	8051 bit address
byte	addr	:	00H	- FFH	-	On-chip 8-bit RAM address
code	addr	:	0000H	- FFFFH	-	Absolute code address
page	addr	:	0000H	- 07FFH	-	11-bit code address
rel	offset	:	00H	- FFH	-	8-bit 2's complement addr
data	byte	:	00H	- FFH	-	Immediate data byte
data	word	:	0000H	- FFFFH	-	Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
20	3	JB bit-addr, rel-offset	JB 01H, F5H
21	2	AJMP page-addr	AJMP 010FH
22	1	RET	RET
23	1	RL A	RL A
24	2	ADD A, #data-byte	ADD A, #22H
25	2	ADD A, byte-addr	ADD A, 32H
26	1	ADD A, @R0	ADD A, @R0
27	1	ADD A, @R1	ADD A, @R1
28	1	ADD A, R0	ADD A, R0
29	1	ADD A, R1	ADD A, R1
2A	1	ADD A, R2	ADD A, R2
2B	1	ADD A, R3	ADD A, R3
2C	1	ADD A, R4	ADD A, R4
2D	1	ADD A, R5	ADD A, R5
2E	1	ADD A, R6	ADD A, R6
2F	1	ADD A, R7	ADD A, R7
30	3	JNB bit-addr, rel-offset	JNB 0CH, 57H
31	2	ACALL page-addr	ACALL 0143H
32	1	RETI	RETI
33	1	RLC A	RLC A
34	2	ADDC A, #data-byte	ADDC A, #87H
35	2	ADDC A, byte-addr	ADDC A, ACH
36	1	ADDC A, @R0	ADDC A, @R0
37	1	ADDC A, @R1	ADDC A, @R1
38	1	ADDC A, R0	ADDC A, R0
39	1	ADDC A, R1	ADDC A, R1
3A	1	ADDC A, R2	ADDC A, R2
3B	1	ADDC A, R3	ADDC A, R3
3C	1	ADDC A, R4	ADDC A, R4
3D	1	ADDC A, R5	ADDC A, R5
3E	1	ADDC A, R6	ADDC A, R6
3F	1	ADDC A, R7	ADDC A, R7

bit	addr	:	00H	-	FFH	-	8051 bit address
byte	addr	:	00H	-	FFH	-	On-chip 8-bit RAM address
code	addr	:	0000H	-	FFFFH	-	Absolute code address
page	addr	:	0000H	-	07FFH	-	11-bit code address
rel	offset	:	00H	-	FFH	-	8-bit 2's complement addr
data	byte	:	00H	-	FFH	-	Immediate data byte
data	word	:	0000H	-	FFFFH	-	Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
40	2	JC rel-offset	JC BBH
41	2	AJMP page-addr	AJMP 0260H
42	2	ORL byte-addr, A	ORL 32H, A
43	3	ORL byte-addr, #data-byte	ORL 23H, #C5H
44	2	ORL A, #data-byte	ORL A, #67H
45	2	ORL A, byte-addr	ORL A, 43H
46	1	ORL A, @R0	ORL A, @R0
47	1	ORL A, @R1	ORL A, @R1
48	1	ORL A, R0	ORL A, R0
49	1	ORL A, R1	ORL A, R1
4A	1	ORL A, R2	ORL A, R2
4B	1	ORL A, R3	ORL A, R3
4C	1	ORL A, R4	ORL A, R4
4D	1	ORL A, R5	ORL A, R5
4E	1	ORL A, R6	ORL A, R6
4F	1	ORL A, R7	ORL A, R7
50	2	JNC rel-offset	JNC BAH
51	2	ACALL page-addr	ACALL 0220H
52	2	ANL byte-addr, A	ANL 04H, A
53	3	ANL byte-addr, #data-byte	ANL 23H, #C7H
54	2	ANL A, #data-byte	ANL A, #AEH
55	2	ANL A, byte-addr	ANL A, 03H
56	1	ANL A, @R0	ANL A, @R0
57	1	ANL A, @R1	ANL A, @R1
58	1	ANL A, R0	ANL A, R0
59	1	ANL A, R1	ANL A, R1
5A	1	ANL A, R2	ANL A, R2
5B	1	ANL A, R3	ANL A, R3
5C	1	ANL A, R4	ANL A, R4
5D	1	ANL A, R5	ANL A, R5
5E	1	ANL A, R6	ANL A, R6
5F	1	ANL A, R7	ANL A, R7

bit	addr	:	00H	-	FFH	- 8051 bit address
byte	addr	:	00H	-	FFH	- On-chip 8-bit RAM address
code	addr	:	0000H	-	FFFFH	- Absolute code address
page	addr	:	0000H	-	07FFH	- 11-bit code address
rel	offset	:	00H	-	FFH	- 8-bit 2's complement addr
data	byte	:	00H	-	FFH	- Immediate data byte
data	word	:	0000H	-	FFFFH	- Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
60	2	JZ rel-offset	JZ 04H
61	2	AJMP page-addr	AJMP 03F0H
62	2	XRL byte-addr, A	XRL DEH, A
63	3	XRL byte-addr, #data-byte	XRL E9, #77H
64	2	XRL A, #data-byte	XRL A, #55H
65	2	XRL A, byte-addr	XRL A, 01H
66	1	XRL A, @R0	XRL A, @R0
67	1	XRL A, @R1	XRL A, @R1
68	1	XRL A, R0	XRL A, R0
69	1	XRL A, R1	XRL A, R1
6A	1	XRL A, R2	XRL A, R2
6B	1	XRL A, R3	XRL A, R3
6C	1	XRL A, R4	XRL A, R4
6D	1	XRL A, R5	XRL A, R5
6E	1	XRL A, R6	XRL A, R6
6F	1	XRL A, R7	XRL A, R7
70	2	JNZ rel-offset	JNZ 56H
71	2	ACALL page-addr	ACALL 0323H
72	2	ORL C, bit-addr	ORL C, 7FH
73	1	JMP @A+DPTR	JMP @A+DPTR
74	2	MOV A, #data-byte	MOV A, #56H
75	3	MOV byte-addr, #data-byte	MOV 34H, #45H
76	2	MOV @R0, #data-byte	MOV @R0, #89H
77	2	MOV @R1, #data-byte	MOV @R1, #23H
78	2	MOV R0, #data-byte	MOV R0, #25H
79	2	MOV R1, #data-byte	MOV R1, #12H
7A	2	MOV R2, #data-byte	MOV R2, #FDH
7B	2	MOV R3, #data-byte	MOV R3, #15H
7C	2	MOV R4, #data-byte	MOV R4, #13H
7D	2	MOV R5, #data-byte	MOV R5, #E9H
7E	2	MOV R6, #data-byte	MOV R6, #A7H
7F	2	MOV R7, #data-byte	MOV R7, #14H

bit	addr	:	00H	-	FFH	- 8051 bit address
byte	addr	:	00H	-	FFH	- On-chip 8-bit RAM address
code	addr	:	0000H	-	FFFFH	- Absolute code address
page	addr	:	0000H	-	07FFH	- 11-bit code address
rel	offset	:	00H	-	FFH	- 8-bit 2's complement addr
data	byte	:	00H	-	FFH	- Immediate data byte
data	word	:	0000H	-	FFFFH	- Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
80	2	SJMP rel-offset	SJMP 0013H
81	2	AJMP page-addr	AJMP 0402H
82	2	ANL C, bit-addr	ANL C, 09H
83	1	MOVC A, @A+PC	MOVC A, @A+PC
84	1	DIV AB	DIV AB
85	3	MOV byte-addr, byte-addr	MOV 01H, 34H
86	2	MOV byte-addr, @R0	MOV 06H, @R0
87	2	MOV byte-addr, @R1	MOV 21H, @R1
88	2	MOV byte-addr, R0	MOV 15H, R0
89	2	MOV byte-addr, R1	MOV 05H, R1
8A	2	MOV byte-addr, R2	MOV 24H, R2
8B	2	MOV byte-addr, R3	MOV 09H, R3
8C	2	MOV byte-addr, R4	MOV 01H, R4
8D	2	MOV byte-addr, R5	MOV 20H, R5
8E	2	MOV byte-addr, R6	MOV 34H, R6
8F	2	MOV byte-addr, R7	MOV 67H, R7
90	3	MOV DPTR, #data-word	MOV DPTR, #AAAAH
91	2	ACALL page-addr	ACALL 0445H
92	1	MOV bit-addr, C	MOV 05H, C
93	2	MOVC A, @A+DPTR	MOVC A, @A+DPTR
94	2	SUBB A, #data-byte	SUBB A, #33H
95	1	SUBB A, byte-addr	SUBB A, 32H
96	1	SUBB A, @R0	SUBB A, @R0
97	1	SUBB A, @R1	SUBB A, @R1
98	1	SUBB A, R0	SUBB A, R0
99	1	SUBB A, R1	SUBB A, R1
9A	1	SUBB A, R2	SUBB A, R2
9B	1	SUBB A, R3	SUBB A, R3
9C	1	SUBB A, R4	SUBB A, R4
9D	1	SUBB A, R5	SUBB A, R5
9E	1	SUBB A, R6	SUBB A, R6
9F	1	SUBB A, R7	SUBB A, R7

bit	addr	:	00H	-	FFH	-	8051 bit address
byte	addr	:	00H	-	FFH	-	On-chip 8-bit RAM address
code	addr	:	0000H	-	FFFFH	-	Absolute code address
page	addr	:	0000H	-	07FFH	-	11-bit code address
rel	offset	:	00H	-	FFH	-	8-bit 2's complement addr
data	byte	:	00H	-	FFH	-	Immediate data byte
data	word	:	0000H	-	FFFFH	-	Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
A0	2	ORL C, /bit-addr	ORL C, /04H
A1	2	AJMP page-addr	AJMP 05FEH
A2	2	MOV C, bit-addr	MOV C, 7FH
A3	1	INC DPTR	INC DPTR
A4	1	MUL AB	MUL AB
A5		Reserved	Reserved
A6	2	MOV @R0, byte-addr	MOV @R0, 45H
A7	2	MOV @R1, byte-addr	MOV @R1, 33H
A8	2	MOV R0, byte-addr	MOV R0, FFH
A9	2	MOV R1, byte-addr	MOV R1, 00H
AA	2	MOV R2, byte-addr	MOV R2, E5H
AB	2	MOV R3, byte-addr	MOV R3, 34H
AC	2	MOV R4, byte-addr	MOV R4, 12H
AD	2	MOV R5, byte-addr	MOV R5, 67H
AE	2	MOV R6, byte-addr	MOV R6, 89H
AF	2	MOV R7, byte-addr	MOV R7, 32H
B0	2	ANL C, /bit-addr	ANL C, /23H
B1	2	ACALL page-addr	ACALL 0503H
B2	2	CPL bit-addr	CPL 02H
B3	1	CPL C	CPL C
B4	3	CJNE A, #data-byte, rel-offset	CJNE A, #32H, 23H
B5	3	CJNE A, byte-addr, rel-offset	CJNE A, 12H, 56H
B6	3	CJNE @R0, #data-byte, rel-offset	CJNE @R0, #66H, 23H
B7	3	CJNE @R1, #data-byte, rel-offset	CJNE @R1, #34H, 12H
B8	3	CJNE R0, #data-byte, rel-offset	CJNE R0, #23H, 56H
B9	3	CJNE R1, #data-byte, rel-offset	CJNE R1, #00H, 10H
BA	3	CJNE R2, #data-byte, rel-offset	CJNE R2, #56H, 23H
BB	3	CJNE R3, #data-byte, rel-offset	CJNE R3, #76H, 56H
BC	3	CJNE R4, #data-byte, rel-offset	CJNE R4, #23H, 45H
BD	3	CJNE R5, #data-byte, rel-offset	CJNE R5, #80H, ACH
BE	3	CJNE R6, #data-byte, rel-offset	CJNE R6, #BAH, CAH
BF	3	CJNE R7, #data-byte, rel-offset	CJNE R7, #09H, 14H

bit	addr	:	00H	FFH	-	8051 bit address
byte	addr	:	00H	FFH	-	On-chip 8-bit RAM address
code	addr	:	0000H	FFFFH	-	Absolute code address
page	addr	:	0000H	07FFH	-	11-bit code address
rel	offset	:	00H	FFH	-	8-bit 2's complement addr
data	byte	:	00H	FFH	-	Immediate data byte
data	word	:	0000H	FFFFH	-	Immediate data word

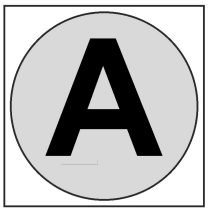
HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
C0	2	PUSH byte-addr	PUSH 23H
C1	2	AJMP page-addr	AJMP 0604H
C2	2	CLR bit-addr	CLR 04H
C3	1	CLR C	CLR C
C4	1	SWAP A	SWAP A
C5	2	XCH A, byte-addr	XCH A, 12H
C6	1	XCH A, @R0	XCH A, @R0
C7	1	XCH A, @R1	XCH A, @R1
C8	1	XCH A, R0	XCH A, R0
C9	1	XCH A, R1	XCH A, R1
CA	1	XCH A, R2	XCH A, R2
CB	1	XCH A, R3	XCH A, R3
CC	1	XCH A, R4	XCH A, R4
CD	1	XCH A, R5	XCH A, R5
CE	1	XCH A, R6	XCH A, R6
CF	1	XCH A, R7	XCH A, R7
D0	2	POP byte-addr	POP 32H
D1	2	ACALL page-addr	ACALL 0634H
D2	2	SETB bit-addr	SETB 03H
D3	1	SETB C	SETB C
D4	1	DA A	DA A
D5	3	DJNZ byte-addr, rel-offset	DJNZ 23H, 34H
D6	1	XCHD A, @R0	XCHD A, @R0
D7	1	XCHD A, @R1	XCHD A, @R1
D8	2	DJNZ R0, rel-offset	DJNZ R0, 02H
D9	2	DJNZ R1, rel-offset	DJNZ R1, 23H
DA	2	DJNZ R2, rel-offset	DJNZ R2, 43H
DB	2	DJNZ R3, rel-offset	DJNZ R3, 87H
DC	2	DJNZ R4, rel-offset	DJNZ R4, ADH
DD	2	DJNZ R5, rel-offset	DJNZ R5, EAH
DE	2	DJNZ R6, rel-offset	DJNZ R6, 34H
DF	2	DJNZ R7, rel-offset	DJNZ R7, F9H

bit	addr	:	00H	FFH	-	8051 bit address
byte	addr	:	00H	FFH	-	On-chip 8-bit RAM address
code	addr	:	0000H	FFFFH	-	Absolute code address
page	addr	:	0000H	07FFH	-	11-bit code address
rel	offset	:	00H	FFH	-	8-bit 2's complement addr
data	byte	:	00H	FFH	-	Immediate data byte
data	word	:	0000H	FFFFH	-	Immediate data word

HEX CODE	NUMBER OF BYTES	SYNTAX	EXAMPLE
E0	1	MOVX A, @DPTR	MOVX A, @DPTR
E1	2	AJMP page-addr	AJMP 0734H
E2	1	MOVX A, @R0	MOVX A, @R0
E3	1	MOVX A, @R1	MOVX A, @R1
E4	1	CLR A	CLR A
E5	2	MOV A, byte-addr	MOV A, 12H
E6	1	MOV A, @R0	MOV A, @R0
E7	1	MOV A, @R1	MOV A, @R1
E8	1	MOV A, R0	MOV A, R0
E9	1	MOV A, R1	MOV A, R1
EA	1	MOV A, R2	MOV A, R2
EB	1	MOV A, R3	MOV A, R3
EC	1	MOV A, R4	MOV A, R4
ED	1	MOV A, R5	MOV A, R5
EE	1	MOV A, R6	MOV A, R6
EF	1	MOV A, R7	MOV A, R7
F0	1	MOVX @DPTR, A	MOVX @DPTR, A
F1	2	ACALL page-addr	ACALL 0703H
F2	1	MOVX @R0, A	MOVX @R0, A
F3	1	MOVX @R1, A	MOVX @R1, A
F4	1	CPL A	CPL A
F5	2	MOV byte-addr, A	MOV 34H, A
F6	1	MOV @R0, A	MOV @R0, A
F7	1	MOV @R1, A	MOV @R1, A
F8	1	MOV R0, A	MOV R0, A
F9	1	MOV R1, A	MOV R1, A
FA	1	MOV R2, A	MOV R2, A
FB	1	MOV R3, A	MOV R3, A
FC	1	MOV R4, A	MOV R4, A
FD	1	MOV R5, A	MOV R5, A
FE	1	MOV R6, A	MOV R6, A
FF	1	MOV R7, A	MOV R7, A

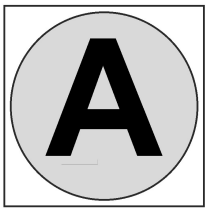
bit	addr	:	00H	-	FFH	- 8051 bit address
byte	addr	:	00H	-	FFH	- On-chip 8-bit RAM address
code	addr	:	0000H	-	FFFFH	- Absolute code address
page	addr	:	0000H	-	07FFH	- 11-bit code address
rel	offset	:	00H	-	FFH	- 8-bit 2's complement addr
data	byte	:	00H	-	FFH	- Immediate data byte
data	word	:	0000H	-	FFFFH	- Immediate data word

FE-51CC03



Appendix

FE-51CC03



Appendix

A.1. Introduction

Ceibo FE-51RD2 and FE-51CC03 are based on the same technology, so for the information please refer to the FE-51RD2 User's Manual. This Appendix describes the differences and steps you should follow to use the FE-51CC03 emulator.

A.2. General Description

The system includes the FE-51CC03 emulator that is very similar to the FE-51RD2 as described in the manual.

The major difference is the chip installed on the emulator; it must be a AT89C51CC03.

The emulator needs clock and power from the target to work, so a mechanical adapter - **ADP-51CC03D** - is provided to convert the AT89C51CC03 microcontroller pinout to a standard 8051. This is due to the rotated configuration of the chip. Therefore, if you do not have a target ready to run the emulator, you may use the supplied DB-51RD2 board with this PLCC-44 to DIP-40 adapter. This adapter provides connection to the clock source, power (Vcc and GND) as well as Port 0, 1, 2 and 3. Please note that Port 0 is open-drain and needs pull-up resistors to get the high level.

Use the emulator with your target or **DB-51RD2 and the provided adapter**.

Warning! ***Do not connect the emulator directly to U2 (44-PLCC socket) on DB-51RD2 Board. This socket is for standard 8051 pinout and NOT for the 89C51CC01/3 in 44-PLCC package.***

A.3. Installing the Software and the System

Ceibo Windows debugger can be used to operate many systems, so a selection of the desired device (AT89C51CC03) is required. Follow these instructions:

1. Install the software. It is not necessary to connect the system to your computer.

2. Connect the system to your PC and target and make sure that the ***LED on FE-51CC03 is ON***.

3. Invoke the debugger.

4. You may get ***error messages*** at this point. Press Abort button in the Warning Dialog of the debugger.

5. Select in the Options Menu, ***Architecture, Chip, AT89C51CC03***.

Now you are ready to use the system.

A.4. Replacing the T89C51CC03 Microcontroller

If you need to replace the PLCC micrcontroller on the emulator:

1. Install the new chip in the emulator.

2. Invoke the debugger. A warning message will be displayed to update the Monitor program. Press OK to update it.

3. If something is not working, please make sure that the selected chip in the Options Menu is 89C51CC03.

A.5. Programmable Clock Generator

The hardware has been upgraded and now includes a programmable clock generator. The target frequency is according to the setup in Options/ Architecture/ XTAL menu.

Use the XTAL command to tell the software which crystal frequency you are using. With this value the system will be able to calculate time events as displayed in the Trace window. Your selection causes the built-in programmable clock to set the desired frequency. The default is Hz, although you can add after your entry Hz, KHz or MHz. Your entry is not case sensitive. The frequency will be set to the nearest value allowed by the programmable clock generator.

X1 is disconnected in the PLCC emulation header. X2 is connected and can be used in your target as a clock output signal.

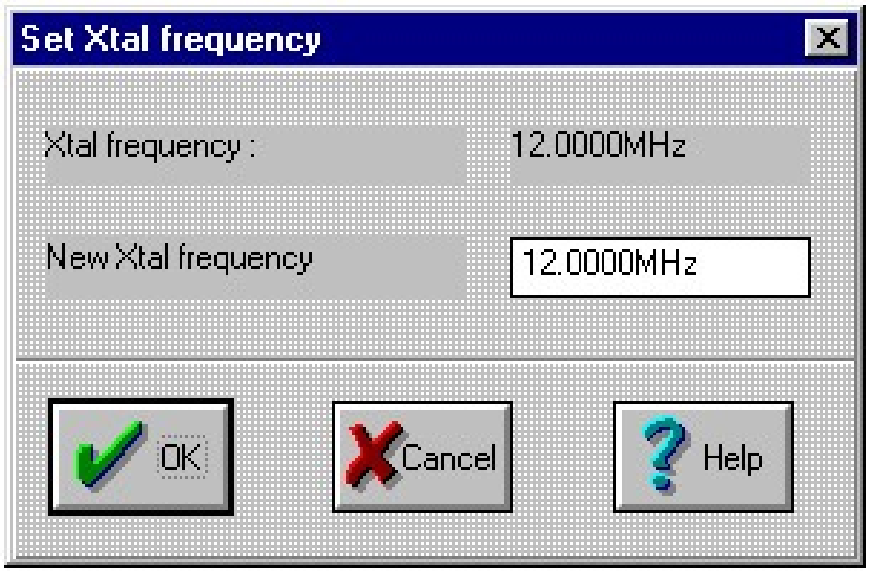


FIGURE: ***Frequency Selection***

The new hardware revision of the emulator also adds a jumper for clock source selection (**INT/EXT**):

INT means the clock source is taken from the emulator programmable clock generator.

EXT means the clock source is taken from the user target board.

Make sure that the jumper cap is installed. Othewise, the emulator will not work.

A.6. Xtal on DB-51RD2 Development Board

Although the emulator does not need the crystal on the target board, it is still necessary while using the DB-51RD2 development board to program a microcontroller using the MP-51RD2 programmer.

A.7. Trace Capabilities

The latest software version enables the emulator trace function. The readme document in the software update explains the use of the trace.

A.8. Software Updates

Check always www.ceibo.com for software updates and improvements.

Index

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z