

DS-251 Development System



User's Manual

© COPYRIGHT BY CEIBO

Rev. 10/13 - V1.09

CONTENTS

PREFACE

P.1. Features	I
P.2. The Technology	II
P.3. The Software	II
P.4. Emulation Restrictions and Troubleshooting	II
P.5. About The Manual	II

CHAPTER 1. DESCRIPTION OF THE SYSTEM

1.1. Introduction.....	1-1
1.2. General Description	1-1
1.3. Applications.....	1-2
1.4. Specifications	1-2
1.5. Hardware Description.....	1-4
1.6. Trace Option	1-8
1.7. ROM/ROMless Support	1-8
1.8. Power Supply	1-9
1.9. Emulation Restrictions.....	1-9

CHAPTER 2. INSTALLATION AND QUICK START-UP

2.1. Introduction.....	2-1
2.2. RS-232C Interface.....	2-1
2.3. Connecting The DS-251 Components	2-2
2.4. Installing The Software.....	2-2
2.5. Starting Up.....	2-2
2.6. Quick Start-Up.....	2-3
2.7. Trace Clips	2-4
2.8. Reset Pushbutton.....	2-5
2.9. Power Supply Connector	2-6

CHAPTER 3. ABOUT THE WINDOWS DEBUGGER

3.1. Introduction.....	3-1
3.2. Debug Capabilities.....	3-1
3.3. Global Menus	3-2
3.4. Local Menus.....	3-3

3.5. Input Boxes	3-3
3.6. Windows.....	3-3
3.7. Using the Menus	3-4
3.8. Toolbar.....	3-4
3.9. Status Line.....	3-5

CHAPTER 4. WINDOWS DEBUGGING SESSION

4.1. Introduction	4-1
4.2. Preparing the Software.....	4-1
4.3. Accessing the Global Menu.....	4-1
4.4. Accessing the Local Menu	4-2
4.5. Selecting the Simulation Mode	4-3
4.6. Adding Watches.....	4-3
4.7. Changing Watches	4-4
4.8. Watching Memory Spaces.....	4-5
4.9. Displaying a Memory Space	4-6
4.10. Changing the Windows	4-6
4.11. Loading a File	4-7
4.12. Capturing Watches.....	4-8
4.13. Debugging the Program	4-9

CHAPTER 5. WINDOWS MENUS AND COMMANDS

5.1. Introduction	5-1
5.2. File Menu.....	5-1
Load.....	5-1
Get Info	5-3
New.....	5-3
Exit.....	5-3
5.3. View Menu	5-3
Breakpoints.....	5-4
<i>Set Options</i>	5-4
<i>Cycle</i>	5-4
<i>Address Match</i>	5-4
<i>Address</i>	5-4
<i>Passcount</i>	5-4
<i>Add</i>	5-4
<i>Remove</i>	5-4
<i>Enable/Disable</i>	5-4
<i>Inspect</i>	5-5
<i>Delete All</i>	5-5

Variables.....	5-5
<i>Watch</i>	5-6
<i>Inspect</i>	5-6
Module	5-6
<i>Inspect</i>	5-7
<i>Watch</i>	5-7
<i>Line</i>	5-7
<i>Search</i>	5-7
<i>Next</i>	5-7
<i>Origin</i>	5-7
<i>New PC</i>	5-7
Watches.....	5-8
<i>Watch</i>	5-8
<i>Edit</i>	5-9
<i>Remove</i>	5-9
<i>Delete All</i>	5-9
<i>Inspect</i>	5-9
<i>Change</i>	5-9
CPU.....	5-9
<i>Disassembly</i>	5-9
<i>Go To</i>	5-9
<i>Origin</i>	5-9
<i>Toggle Source</i>	5-10
<i>Assemble</i>	5-10
<i>New PC</i>	5-10
<i>Print to File</i>	5-10
<i>Stack</i>	5-11
<i>Go To</i>	5-11
<i>Origin</i>	5-11
<i>Change</i>	5-11
Registers.....	5-11
<i>Toggle</i>	5-12
<i>Increment</i>	5-12
<i>Decrement</i>	5-12
<i>Zero</i>	5-12
<i>Read</i>	5-12
<i>Change</i>	5-12
<i>Update</i>	5-12
Performance Analyzer.....	5-13
<i>Configure</i>	5-13
<i>Refresh</i>	5-14
<i>Info</i>	5-14
Trace.....	5-14

	<i>Go to</i>	5-14
	<i>Origin</i>	5-14
	<i>Display Mode</i>	5-15
	<i>Time Stamps</i>	5-15
	<i>Filters</i>	5-15
	<i>Inspect</i>	5-15
	<i>Clear Trace</i>	5-15
	<i>Trace Status</i>	5-15
	<i>Read All Trace</i>	5-15
	<i>Print to File</i>	5-15
	<i>Trace Status</i>	5-15
	Memory Space	5-16
	<i>Go To</i>	5-17
	<i>Search</i>	5-17
	<i>Next</i>	5-17
	<i>Block</i>	5-17
	Target	5-17
5.4. Run Menu		5-18
	Run	5-18
	Execute Forever	5-19
	Go to Cursor	5-19
	Trace Info	5-19
	Execute To	5-19
	Step Over	5-19
	Animate	5-20
	Instruction Trace	5-20
	Continuous Run	5-21
	Halt	5-21
	Program Reset	5-21
5.5. Breakpoints Menu.....		5-21
	Toggle	5-22
	Delete All	5-22
5.6. Data Menu		5-22
	Inspector	5-22
	Evaluate	5-23
	Add Watch	5-23
5.7. Options Menu		5-23
	Environment	5-23
	<i>Language</i>	5-24
	<i>Integer Display Format</i>	5-24
	<i>Beep on Breakpoint</i>	5-24
	Path for Source	5-24

Module List File	5-25
Communication Port	5-25
Communication Baud	5-26
Mode.....	5-26
<i>Emulation</i>	5-26
<i>Simulation</i>	5-26
Architecture	5-27
<i>Memory Map</i>	5-27
<i>Chip Configuration</i>	5-28
Save Settings on Exit	5-28
Debug Controls	5-28
Save Setup	5-28
Restore Setup	5-28
5.8. Windows Menu	5-29
5.9. Help Menu.....	5-29
Index.....	5-29
Topic Search	5-30
About	5-30

CHAPTER 6. ON-LINE ASSEMBLER

6.1. Introduction.....	6-1
6.2. Notation for Instruction Operands	6-2
6.3. Bit Instructions	6-8

CHAPTER 7. SYSTEM ERRORS AND TROUBLESHOOTING

7.1. Introduction.....	7-1
7.2. Error Description.....	7-1
7.3 Common Questions and Answers.....	7-5

PREFACE



DS-251 Development System

PREFACE



DS-251 Development System

DS-251 is an development system dedicated to Intel MCS[®]251 microcontrollers. It is used as in-circuit emulator in your design of a microcontroller embedded application based on the MCS[®]251 architecture.

P.1. Features

- Real-Time and Transparent In-Circuit Emulator for MCS[®]251
- Standard 256K Emulation Memory
- Real-Time Trace up to 128K Frames Deep, 128 Bits Wide
- Complex Hardware Breakpoints
- Supports Both Binary Mode and Source Mode
- MS-Windows Debugger
- High-Level Support for Popular C-Compilers
- Full Support of Local and Global Variables
- On-Line Assembler and Disassembler
- Performance Analyzer
- Serially Linked to IBM PC at 115 KBaud

P.2. The Technology

DS-251 uses Intel bond-out technology for real-time and transparent emulation. This hi-tech concept frees all the microcontroller resources to the user application. All I/O ports are not affected by the emulation process. Furthermore, the system 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

DS-251 is supplied with a Windows Real-Time Debugger, which is based on a MS-Windows platform (i.e. multi-tasking and multi-window display).

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 will explain how to prepare your project to take full advantage of the system.

Chapter 7 gives some *troubleshooting* recommendations to follow in case that you are experiencing problems with your hardware or software. Also *common questions* are answered in Chapter 7.

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 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.

7 System Errors and Troubleshooting

Lists the errors detected while operating DS-251 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 DS-251 is a development system that supports Intel MCS[®]251 microcontrollers 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 oscillator or any other clock source connected to the microcontroller.

Emulation is carried out by loading the system with the user software. As the system uses Intel bond-out technology, emulation is carried out transparently and in real-time without affecting port or register states.

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

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

1.3. Applications

The main applications of DS-251 Development System are:

- Emulation of microcontrollers
- Development of microprocessor based systems
- Hardware and software debugging purposes

1.4. Specifications

EMULATOR MEMORY

DS-251 provides 256 KBytes of code memory with software mapping. Memory mapping has a resolution of 1 Byte and can be set to the target user or belonging to the emulator.

HARDWARE BREAKPOINTS

Breakpoints allow real-time program execution until an opcode is executed at a specified address. Breakpoints on data read or write and an AND/OR combination of two external signals are also implemented. A selection of breakpoints according to rising/falling edge or high/low level of external events is available.

CONDITIONAL BREAKPOINTS

A complete set of conditional breakpoints permit halting program emulation on code addresses, source code lines, access to external and on-chip memory, port and register contents.

LANGUAGES AND FILE FORMATS

DS-251 accepts files generated by compilers from many vendors according to Intel standard OMF-251 format (Keil, IAR, Tasking, etc.). Also 8051 compilers in many different formats can be used with this emulator. Assemblers and high-level languages such as C and PLM are fully supported.

SOURCE-LEVEL DEBUGGER

DS-251 software comes with an MS-Windows debugger. The debugger includes commands which allow the user to get all the information necessary for testing the programs and hardware in real-time. The commands permit setting

breakpoints on high-level language lines, adding a watch window with the symbols and variables of interest, modifying variables, displaying floating point values, showing the trace buffer, executing assembly steps and many more useful functions.

TRACE AND LOGIC ANALYZER

The trace memory is used to record the microprocessor activities. Eight lines are user selectable test points. Trigger inputs and conditions are available for starting and stopping the trace recording. The trace buffer can be viewed in disassembled instructions or high level language lines embedded with the related instructions. The maximum trace memory is 128K frames of 128-bit each including bus activity, program counter, trace clips and 32-bit time stamps with a maximum resolution of 40ns.

PERFORMANCE ANALYZER

This useful function checks the real-time trace buffer and provides time statistics on modules and procedures as a percentage of the total execution time.

PERSONALITY PROBES

DS-251 uses Intel and Temic bond-out microcontrollers for hardware and software emulation. The selection of a different microcontroller is made by replacing the probe or just by software as some bond-outs emulate a family of devices.

<i>Probe</i>	<i>Supported Devices</i>
C251SX	N8xC251SA/B/P/Q
C251TX	N8xC251TA/B/P/Q
C251GX	TSC8x251G1/2, TSC8x251G1D

As the list of supported devices and available probes is continuously evolving, call Ceibo for the latest update.

FREQUENCY

The personality probes run at the frequency of the crystal on them or from the clock source supplied by the user hardware. Therefore, the same probe may be

adapted to the frequency requirements. The minimum and maximum frequencies are determined by the emulated chip characteristics and limited by the bondout capabilities.

HOST CHARACTERISTICS

IBM PC or compatible systems with 8 MByte of RAM and one RS-232 port.

INPUT POWER

5VDC/3.5A.

1.5. ROM/ROMless Support

ROMless operation means the microcontroller P0 and P2 ports are used as bus to fetch instructions. The code memory is located in an external EPROM or other memory devices not belonging to the on-chip resources. EA set to "0" during Reset determines this operating mode.

ROMed operation means the microcontroller P0 and P2 ports may be used as I/O lines or as bus to access external data Ram, but not to fetch instructions. The code memory is located in the on-chip EPROM or ROM. EA set to "1" during Reset defines the ROMed mode.

ROMless or ROMed modes are selected by means of software, by choosing the corresponding device in the Options|Architecture|Chip menu.

1.6. Emulation Restrictions

The following restrictions are valid for DS-251:

- 1 Trap function is reserved by the system.
- 2 Only emulation at $V_{cc} = 5V \pm 5\%$ is possible with standard probes.

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

DS-51 is serially linked to a host computer through an RS-232C interface. A standard phone cable is used to connect the DS-251 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 9-pin male connector to connect to the DS-251 serial interface.

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	9-Pin Male Connector (DS-251)	9-Pin Female Connector (PC Side)
Receive Data	pin 3	pin 3
Transmit Data	pin 2	pin 2
Signal Ground	pin 5	pin 5

TABLE 2.1: RS-232 Cable

2.3. Connecting the DS-251 Components

No special tools are required to install the DS-251.

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 serial cable into the serial jack on DS-251.
- b) Connect the other end of the serial cable to host PC serial channel COM1 (COM2, COM3 or COM4).
- c) Plug the power supply into the power connector.
- d) Plug the other end of the cable into a power outlet.

2.4. Installing the Software

Follow the steps described below to install your DS-251 software:

- a Turn on the host computer.
- b Insert your DS-251 disk into Drive A or B.
- c Run Windows.
- d From Program Manager select the drive with the Windows Debugger disk and run SETUP. The Windows Setup creates the Ceibo Windows Debugger icon.

2.5. Starting Up

Turn the DS-251 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 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.

2.6. Quick Start-Up

Before using the system for emulation purposes, you must initialize the configuration bits and select the emulated chip.

Selecting the Microcontroller

Click on the Option Menu and Chip Command. Select the desired ROM or ROMless device from this menu.

Configuration Bit Assignments

You have to initialize all the parameters of the Options Menu/Chip Configuration.

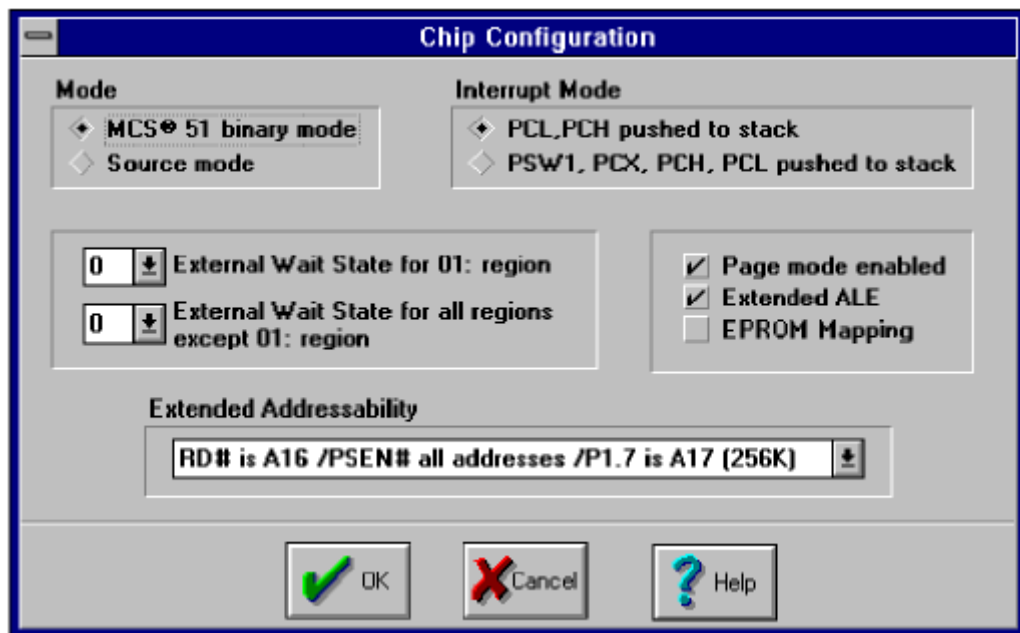


Figure 2.1: *Chip Configuration Window*

The available options are:

-
- 1 Source or binary compatibility,
 - 2 Interrupt mode 51 or 251,
 - 3 Number of unit states,
 - 4 Page mode, ALE type and EPROM mapping,
 - 5 Extended addressability.

2.7. Trace Clips

Trace Clips must be connected to the Trace Connector located on the header as follows:

TOP VIEW					
Testpoint #0	1	> 0	0	2	Testpoint #1
Testpoint #2	3	0	0	4	Testpoint #3
Testpoint #4	5	I 0	0	6	Testpoint #5
Testpoint #6 - Ext. BP1	7	0	0	8	Testpoint #7 - Ext. BP2
Trace Start Trigger	9	0	0	10	Trace Stop Trigger

FIGURE 2.1: *Trace Connector*

The trace clips are used to record external events in real time.

Two of the trace clips may be used as external triggers that allow the trace recording to be started and stopped upon external events.

External triggers are connected as follows:

Start Trigger: Test Point #9 connected to the Red clip with the white wire is also the start trigger signal.

Stop Trigger: Test Point #0 connected to the Red clip with the black wire is the stop trigger signal.

Trace Triggers are restricted to real time operation. During a single step operation all the instructions are recorded.

If any Hardware Breakpoint is set the triggers are automatically disabled.

The software allows the trace trigger levels to be selected. They may be either level or edge for the external start and stop trigger signals.

External triggers are disabled by not selecting neither the edge nor the level states.

The active mode selected is for both start and stop trigger signals.

In the edge mode the trace will record from the valid start edge to a valid stop edge where edge means low to high transition.

In the low level mode the trace will record only when both start and stop triggers are low.

In the high level mode the trace will record only when one or both of the start or stop triggers are high.

Trace **Testpoints #6 and #7** may be used as **External Breakpoints on Hardware Events** if enabled.

Breakpoint trigger may also be the Trace Full condition. Then the emulation will stop once it is full.

Selecting Level or Edge enables the External Breakpoint Triggers on TP 6 and 7.

The Logic Operator commands allow selection of AND/OR combinations of the two external breakpoints.

The options are both low, any one is low, both are high, only one is high, both are leading edge, etc.

2.8. Reset Pushbutton

The Reset Pushbutton has the following functions:

1. One click. If the system is halted the program counter is reset and all the registers are set to the initial state. If the system is executing a program, the microcontroller is reset and the program execution then continues from address 0000h.
2. Two clicks. The system halts at address 0000h.
3. Three clicks. All the emulator hardware is reset.

2.9. Power Supply Connector

The power supply connector is used to apply a regulated DC voltage to the board. You should use the power supply included with the system. If you are using another power supply, the plug must be according to the following connections:

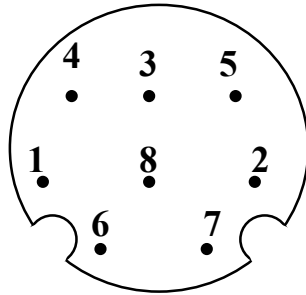


FIGURE 2.2: *Power Connector*

The connections are:

+5V: Pin 4, 3, 5 and 8

GND: Pin 1, 2, 6 and 7

Pin 1, 8 and 2 may be left disconnected in the plug because they are internally connected in the emulator board.

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.

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. Dialog boxes appear when a menu item is chosen and they are indicated as (...).
- 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



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 or emulation) 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 DS-251. 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. Accessing the Global Menu



FIGURE 4.1: *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.

4.4. Accessing the Local Menu

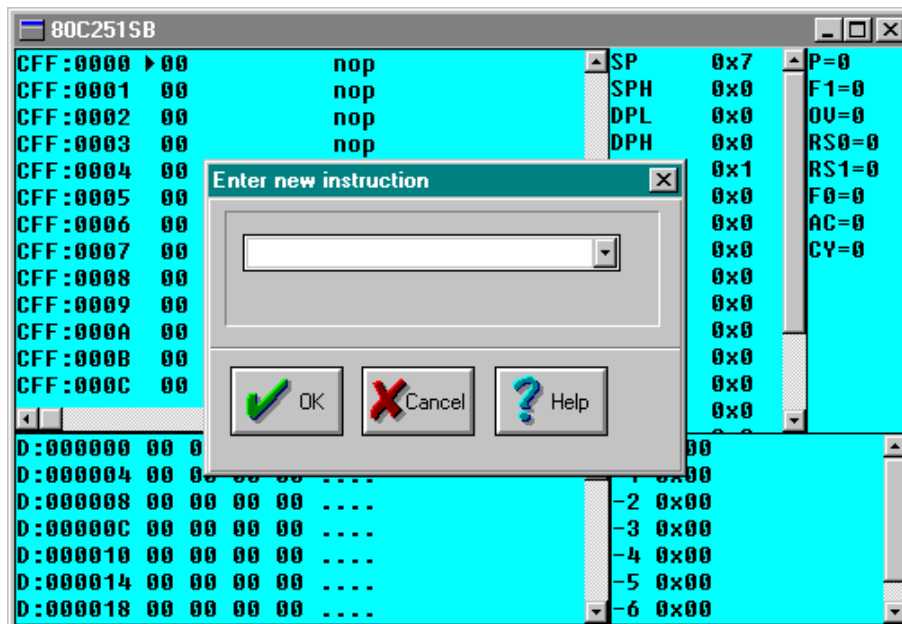


FIGURE 4.2: CPU Assembly Command

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

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

4.5. Selecting the Simulation Mode

Select the Simulation Mode from the Options Menu and the Architecture Command. This mode may not be implemented yet in your current software version. If this is the case, use the emulation mode with the DS-251 connected to your PC.

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

The bottom bar displays the software status.

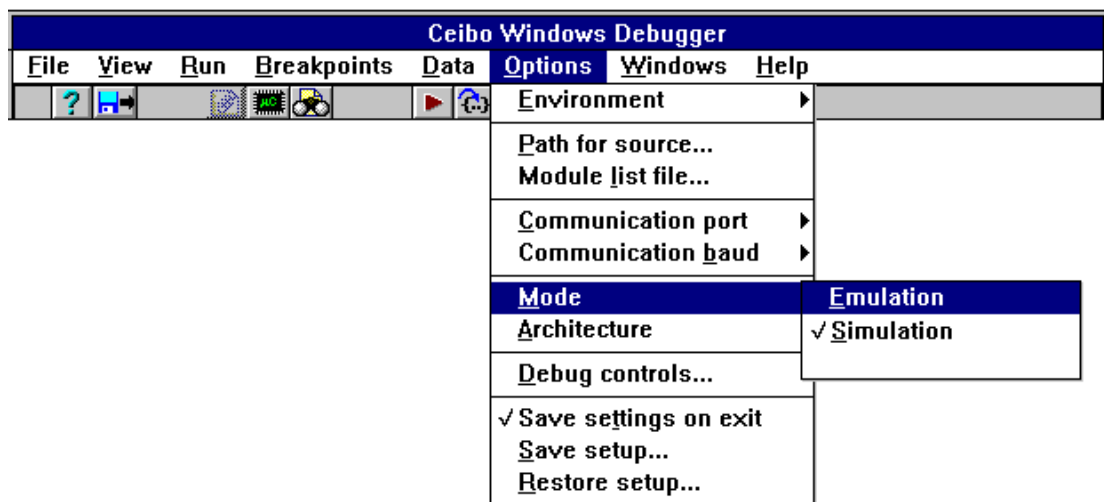


FIGURE 4.3: *Simulation Mode*

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

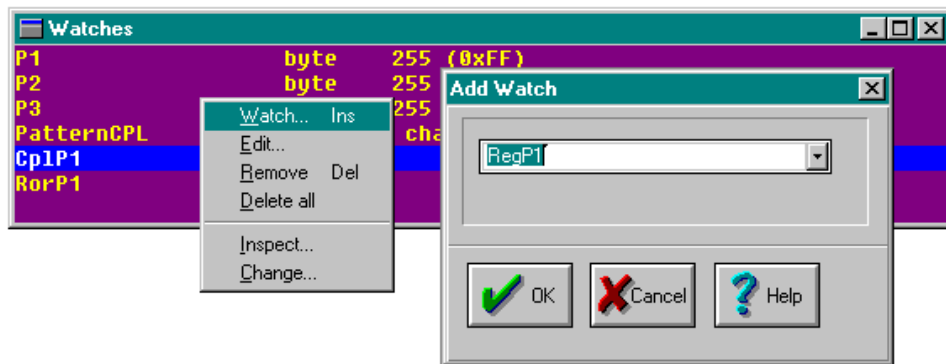


FIGURE 4.4: Adding Watches

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.

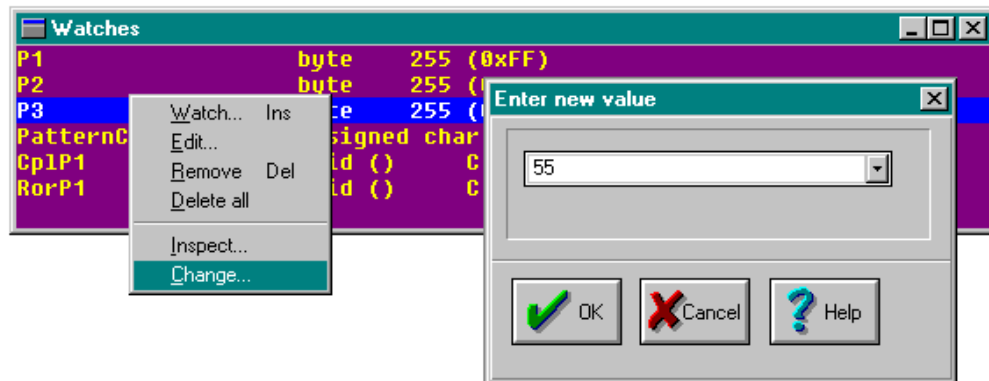


FIGURE 4.5: *Changing Watches*

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 F:401h.

For example, type P: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

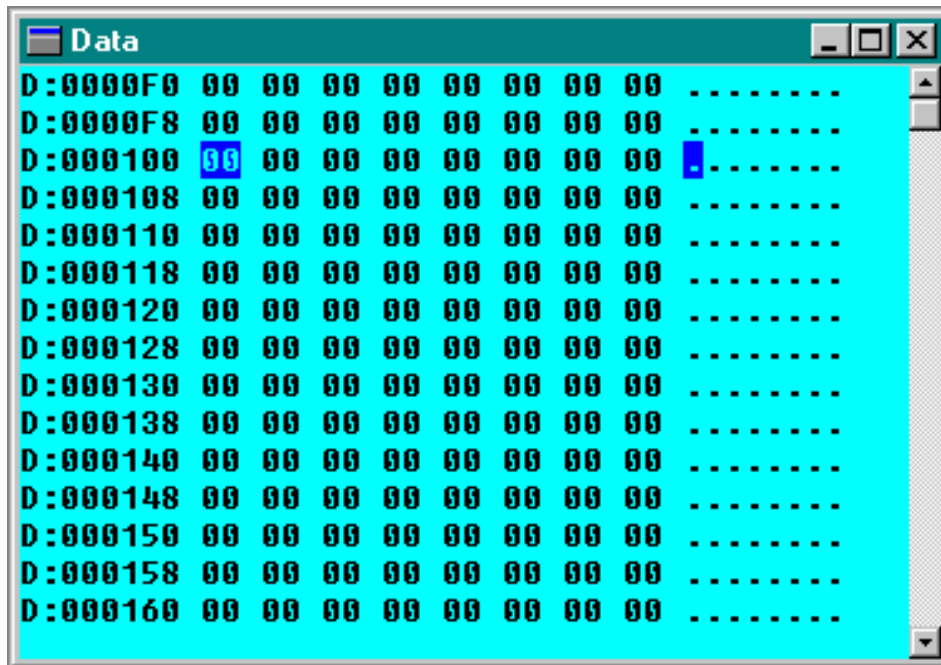


FIGURE 4.6: *Data Window*

- 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.
- 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.

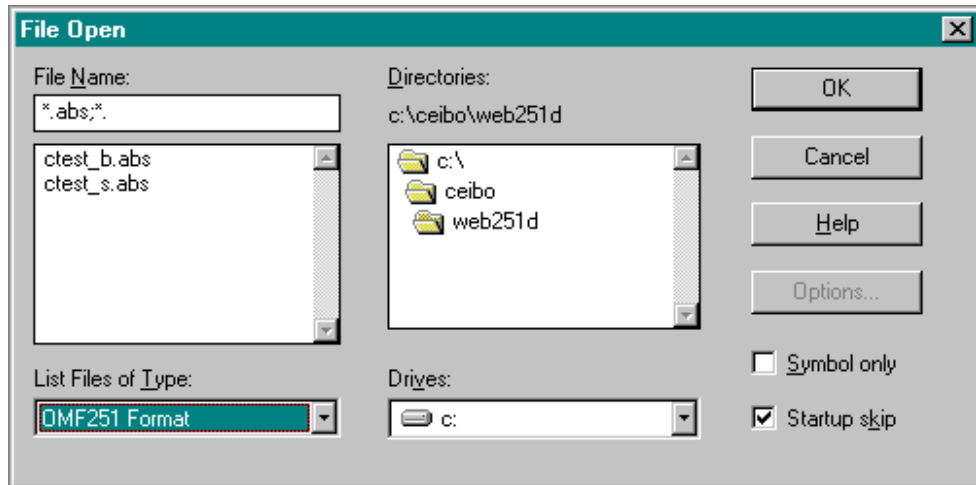


FIGURE 4.7: Loading a File

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

Note that it is very important to define the type of list files to get the proper symbol information. CTEST.ABS and CTESTS.ABS are Keil files in OMF251 format.

You may also load a Hex file without symbol information, but in that case the symbolic debugger will not function.

- 4 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.

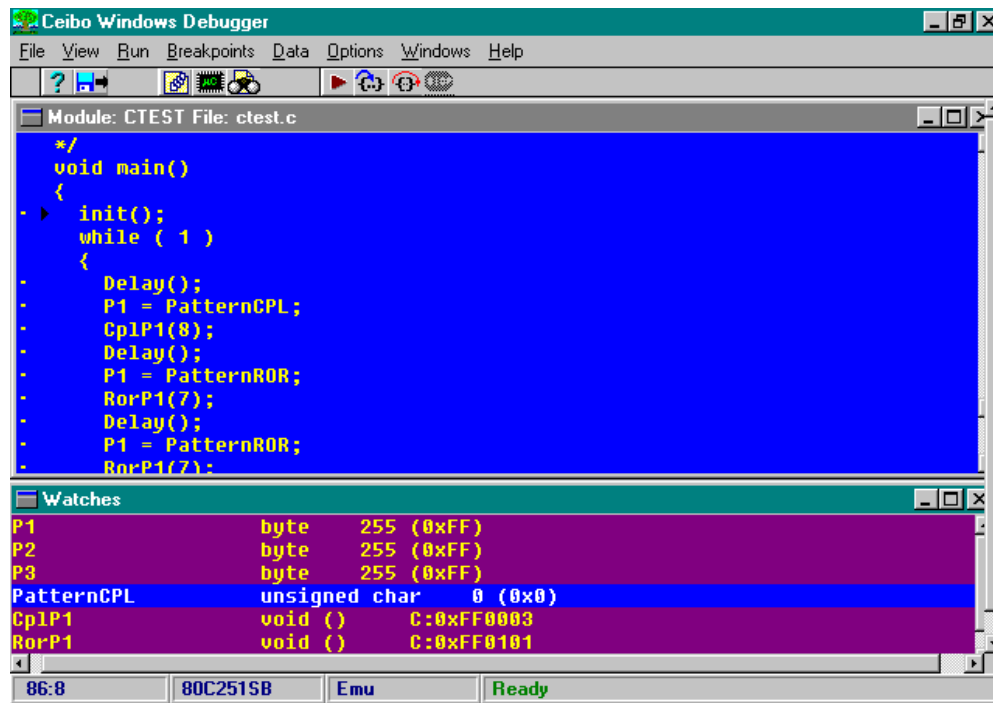


FIGURE 4.8: Default Screen

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

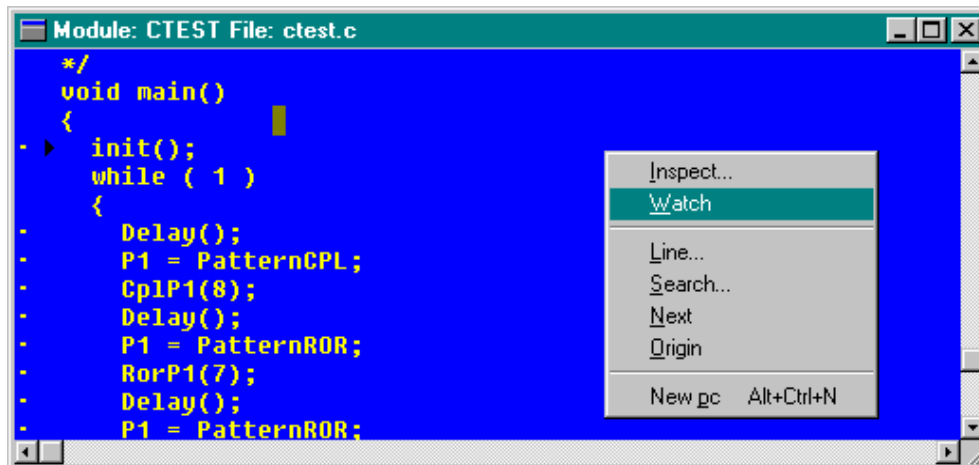


FIGURE 4.9: *Capturing Watches*

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.

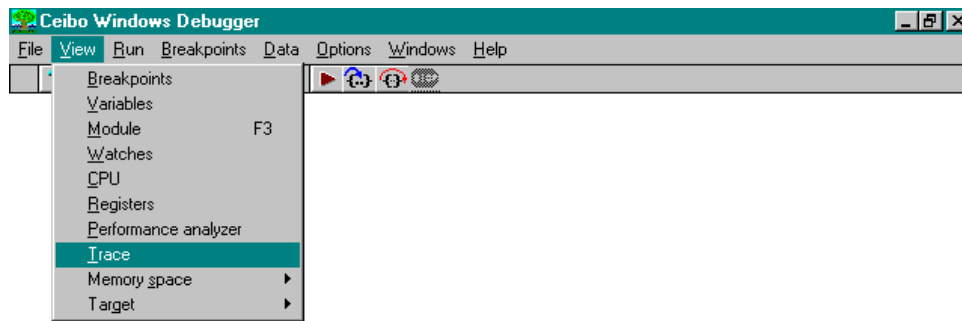


FIGURE 4.10: *View Menu*

- 1 Set a Breakpoint by moving the cursor in the Module window and pressing the F2 key. Execute the program by pressing the F9 key.
- 2 Execute a high-level language step by pressing the F8 key. Repeat that operation several times.
- 3 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.

5.2. File Menu

The File menu options deal with operations external to the Windows 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 and Intel OMF251.

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

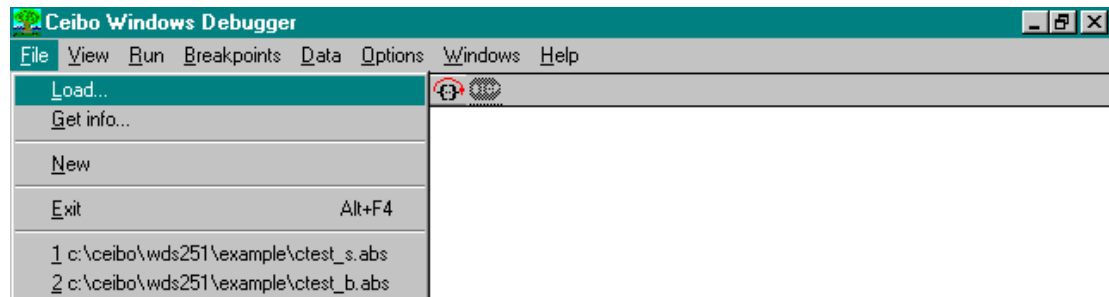


FIGURE 5.1: *File Menu*

From the File Menu you can specify the Symbol Only option, thus loading only the symbol information in the file for debugging ROM applications.

The Startup Skip option is used to run the program automatically until the first line of your main program is reached.

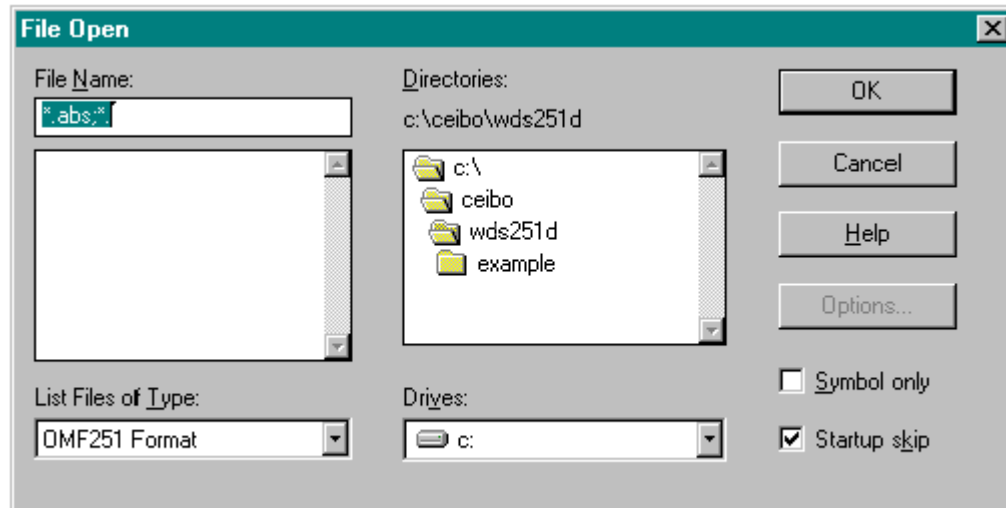


FIGURE 5.2: *File Open Dialog*

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.

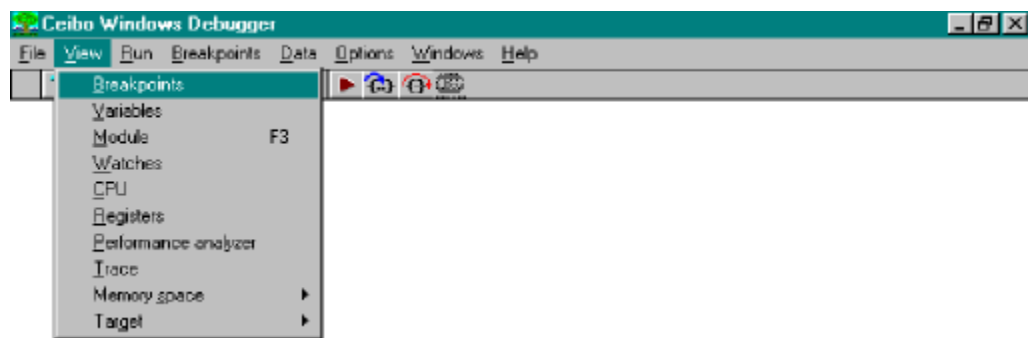


FIGURE 5.3: *View Menu*

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:

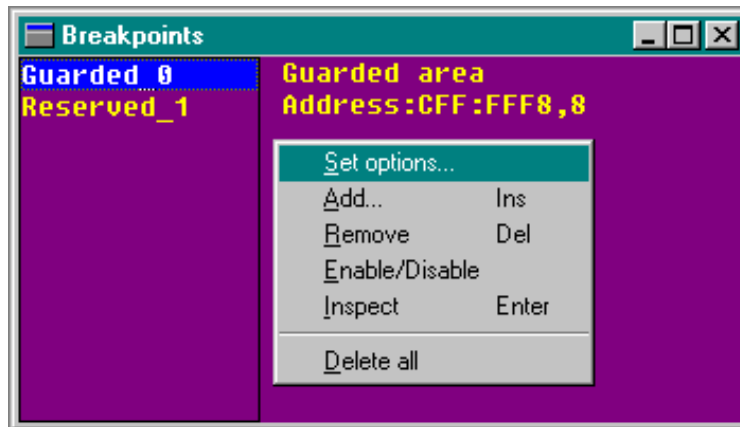


FIGURE 5.4: Breakpoints Local Menu

Set Options: This command is used to redefine the type of bus cycle, address match condition, passcount and address of the selected breakpoint. Click first one of the breakpoints on the left side of the window and then select the Set Options command. Use this command to define the cycle type, address match condition, address range and passcount.

Cycle: Use this option to specify the bus cycle or execution state of a breakpoint.

Address match: Selects the address qualification mode.

Address: Address ranges may be defined by selecting range and length and entering the corresponding addresses separated by a comma or a blank space. The format for an address range is: **address, length**.

Passcount: From the Set Options Dialog Box you may define the passcount value, thus selecting the number of occurrences before the program actually stops.

Add: This command is used to set a new breakpoint. The operation is similar to the above Set Options, with the difference that you do not need to select an existing breakpoint from the window.

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.

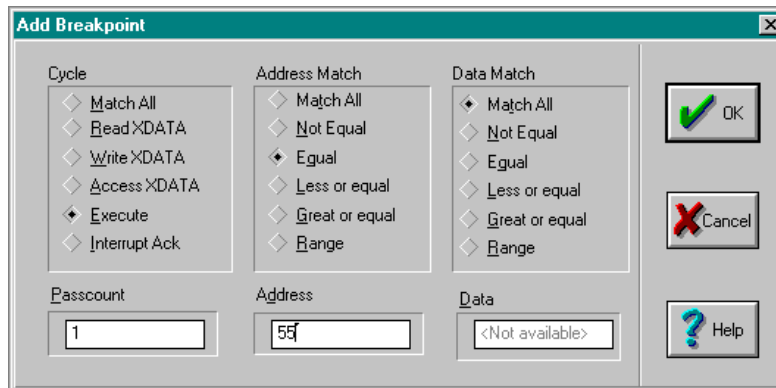


FIGURE 5.5: Add Breakpoint

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.

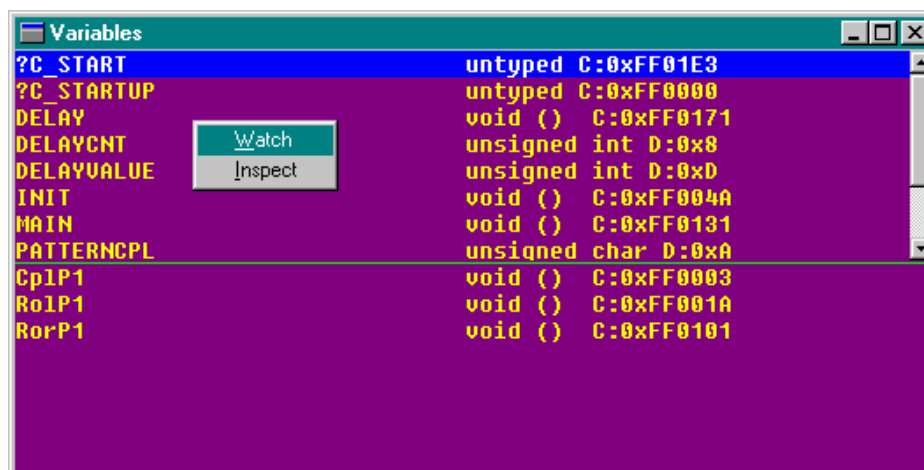


FIGURE 5.6: Variables Window

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.

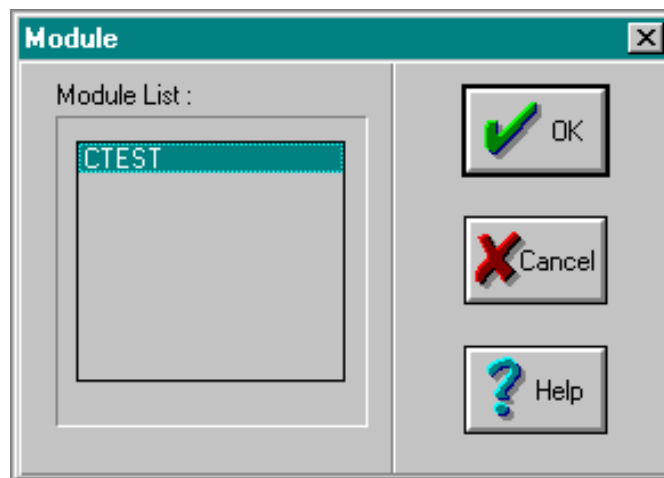


FIGURE 5.7: *Module List*

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:

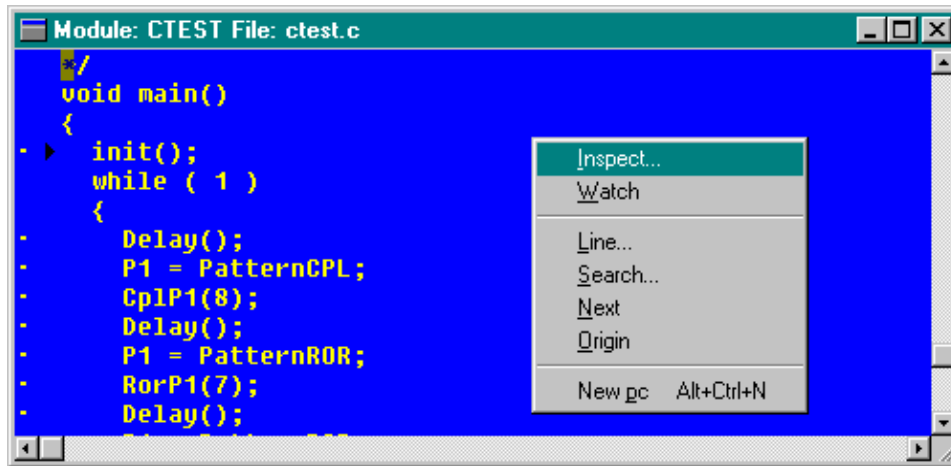


FIGURE 5.8: *Module Local Menu*

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.

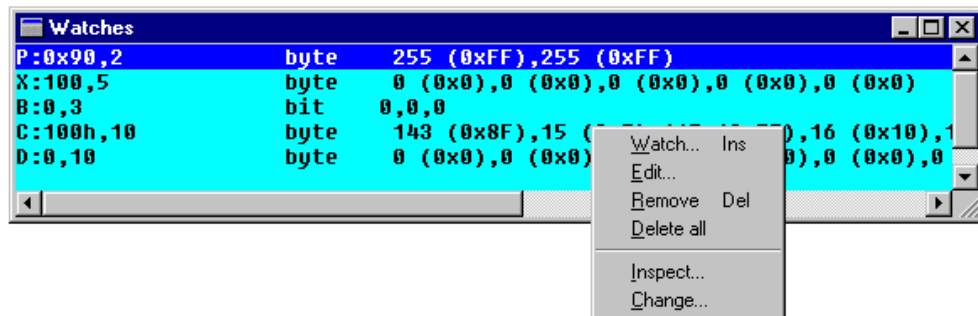


FIGURE 5.9: Watches Local Menu

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.

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.

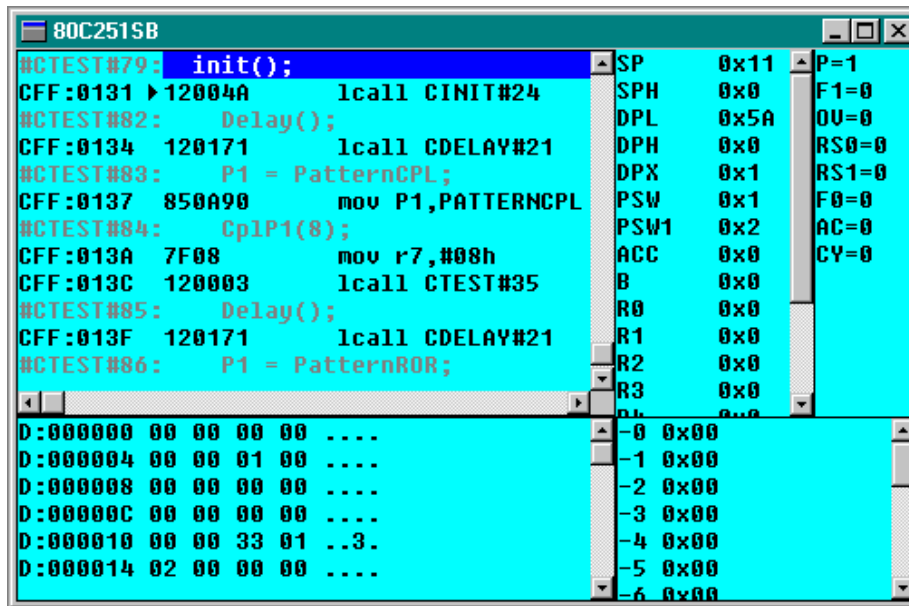


FIGURE 5.10: CPU Window

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.

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.

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:

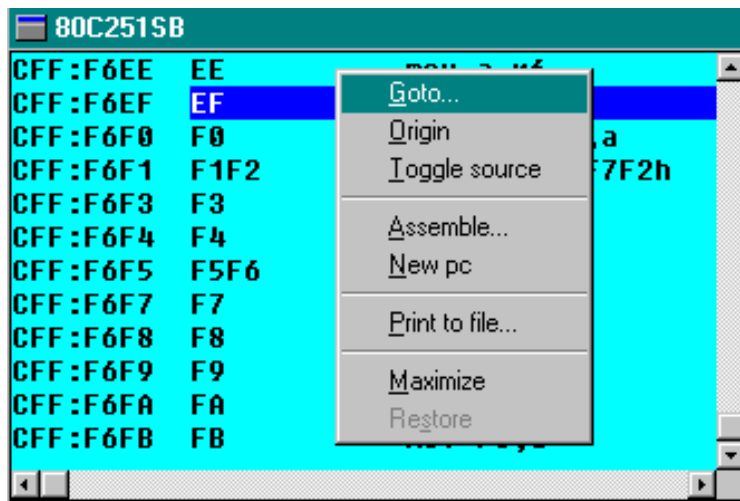


FIGURE 5.11: Disassembly Local Menu

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.

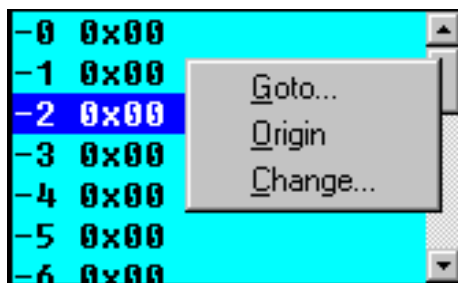


FIGURE 5.12: Stack Local Menu

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.

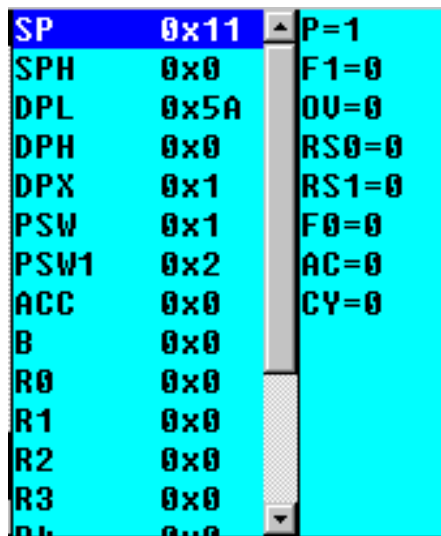
Decrement: Subtracts one from the current value of the selected register byte.

Zero: Clears the selected register byte.

Read: The Read command forces the reading of the specified register. This is a useful command in case that the register is affected by the reading operation.

Change: Use this command to modify the selected register.

Update: Reads all the registers to update the window.



The screenshot shows a window titled 'Registers' with a cyan background. It displays a list of registers and flags. The registers are listed on the left, and their current values are shown in the middle. The flags are listed on the right, and their current states are shown in the middle. A vertical scrollbar is visible between the registers and flags columns.

SP	0x11	P=1
SPH	0x0	F1=0
DPL	0x5A	OV=0
DPH	0x0	RS0=0
DPX	0x1	RS1=0
PSW	0x1	F0=0
PSW1	0x2	AC=0
ACC	0x0	CY=0
B	0x0	
R0	0x0	
R1	0x0	
R2	0x0	
R3	0x0	
R4	0x0	

FIGURE 5.13: *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.

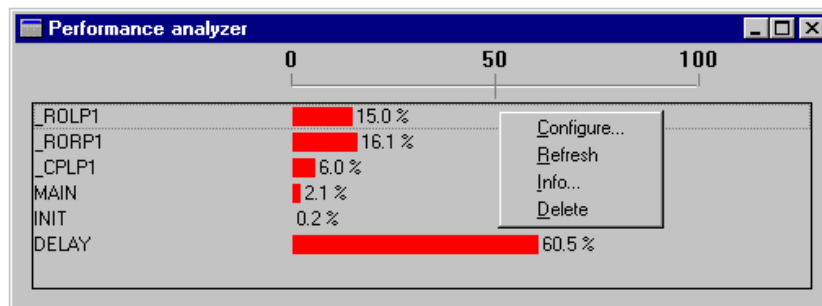


FIGURE 5.14: *Performance Analyzer*

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.

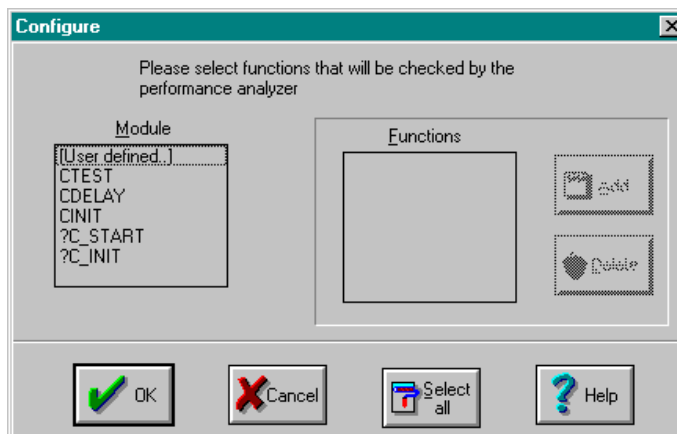


FIGURE 5.15: *User Defined Dialog Box*

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. The Trace functions are enabled in simulation modes only.

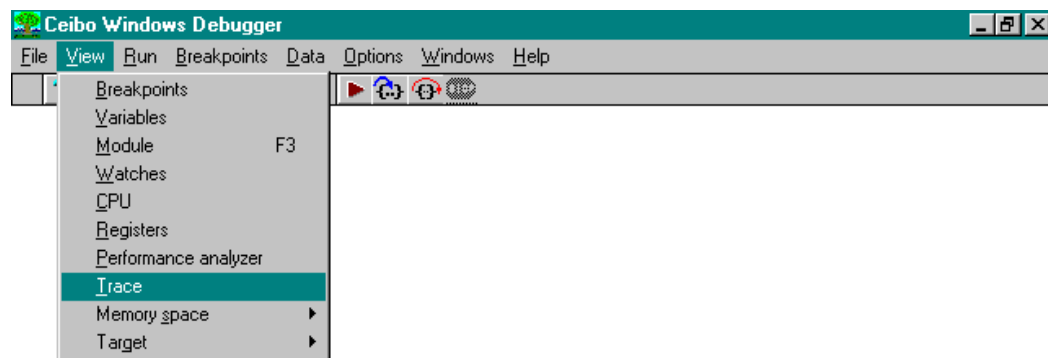


FIGURE 5.16: Trace Menu

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.

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.

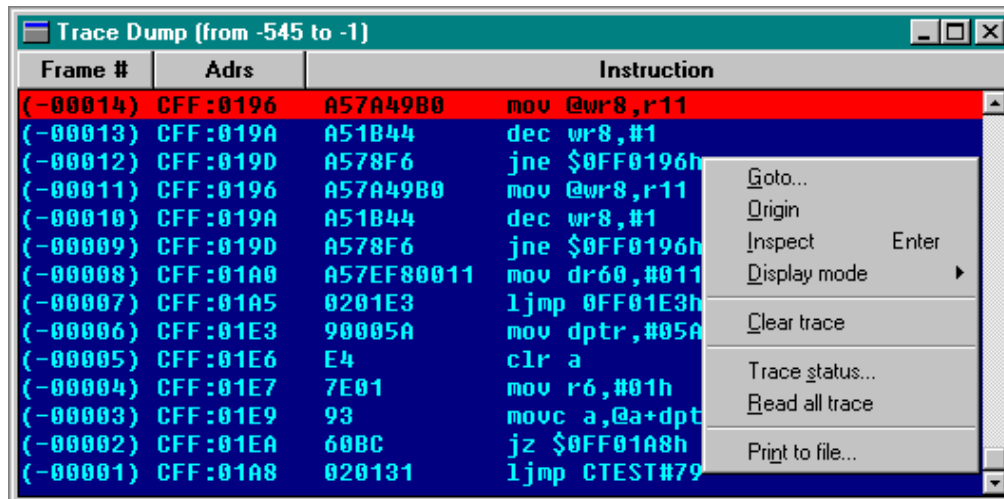


FIGURE 5.17: Trace Dump

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).

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.

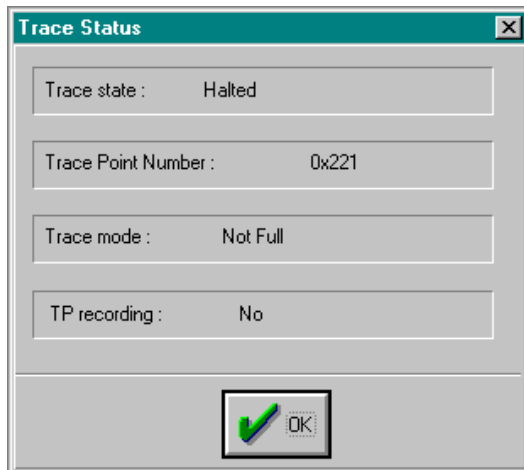


FIGURE 5.18: *Trace Status*

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) or the code memory (Code).

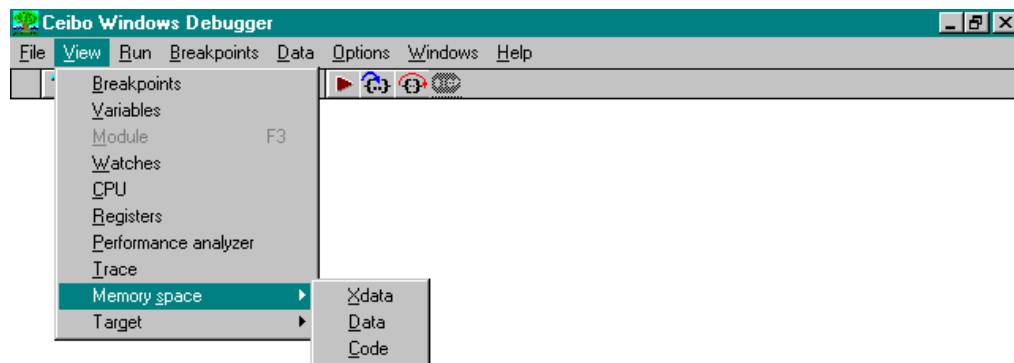


FIGURE 5.19: *Memory Local Menu*

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.

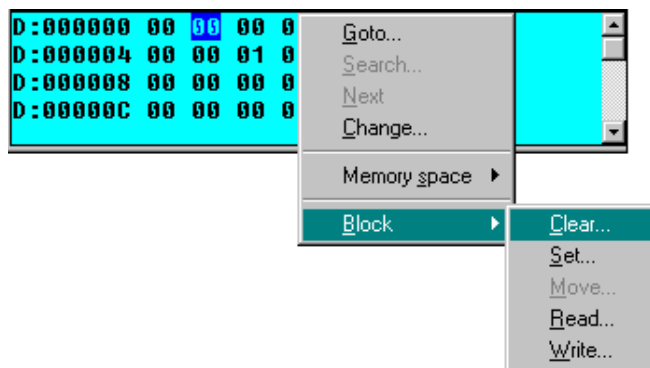


FIGURE 5.20: Block Menu

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.

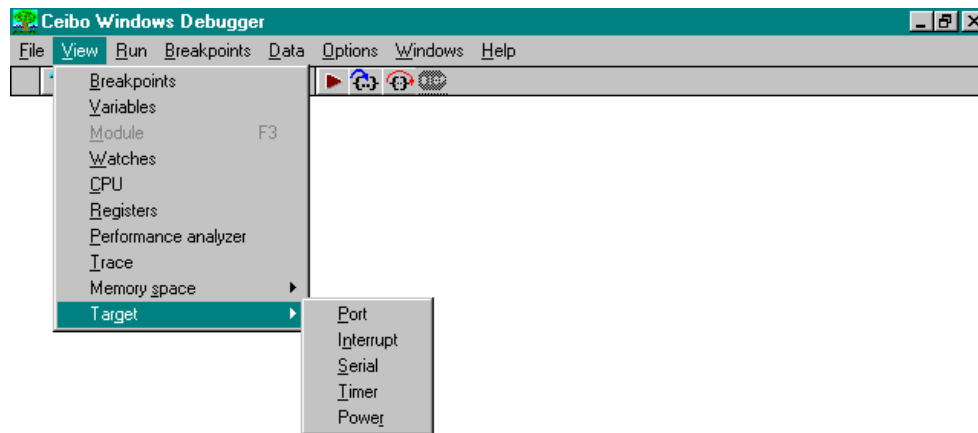


FIGURE 5.21: *Target Menu*

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.

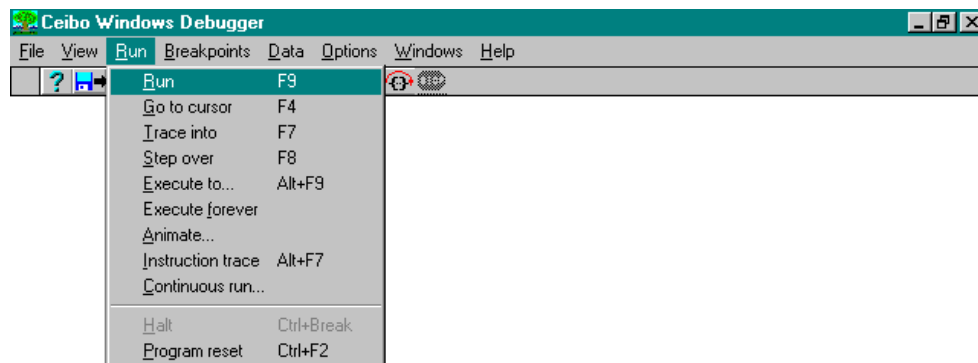


Figure 5.22: *Run Menu*

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.

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.

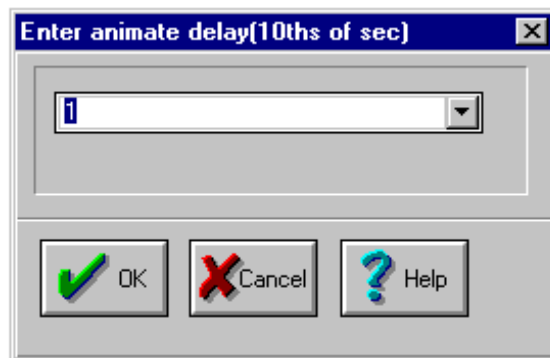


Figure 5.23: *Animate Speed*

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. breakpoints stop the emulation "after" executing the first assembly instruction of the specified address. You may also stop the emulation "before" execution by enabling Software breakpoints in the Debug Control Options.

The following commands are available: Toggle, Breakpoint at, Change Memory Global, Expression True Global and Delete All.

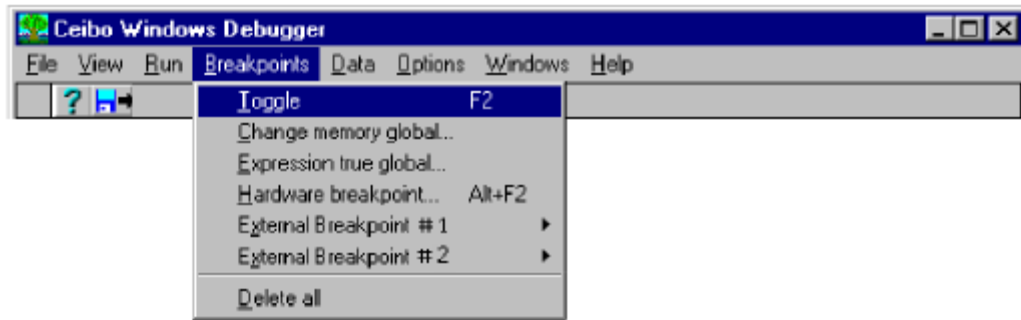


FIGURE 5.29: Breakpoints Menu

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.

Expression True Global

The Expression True Global command allows setting a breakpoint that will occur when the value of the specified memory space location matches the specified data value. This command is available in simulation modes only.

Change Memory Global

This command may be used to set a breakpoint on access to any specified memory space, regardless of the data contents. The Change Memory Global command is available in simulation modes only.

Hardware Breakpoint

This command opens a dialog box to set a breakpoint according to the cycle, address and number of occurrences. The Add Breakpoint Dialog box has been explained in the Set Options Dialog box of the View menu and Breakpoints Local menu.

Hardware breakpoints stop emulation "after" executing the first assembly instruction of the specified address. Use Software breakpoints to stop "before" execution (see Debug Control Options).

External Breakpoint

This command is used to stop the emulation upon external signals connected to the Trace Clips (see Chapter 2, paragraph 2.7). A selection of different logic levels and edges is available. The logic operator is used to select an AND/OR combination of external breakpoints.

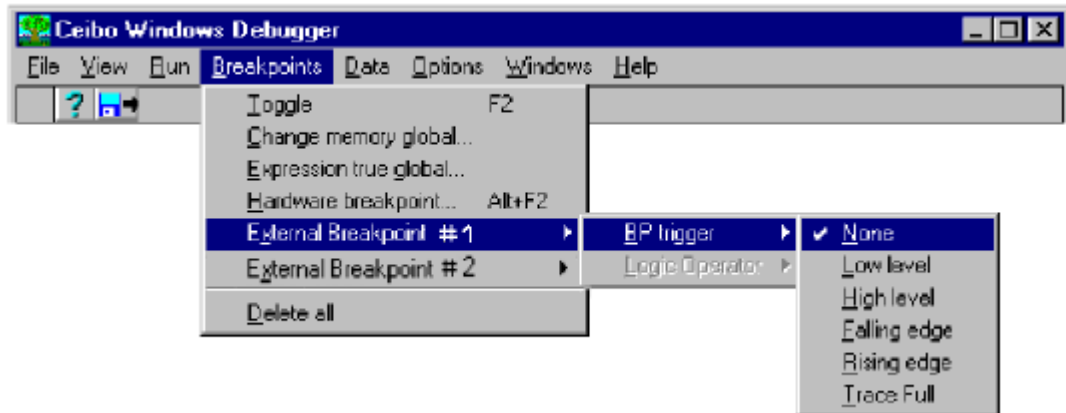


FIGURE 5.30: *External Breakpoint*

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.

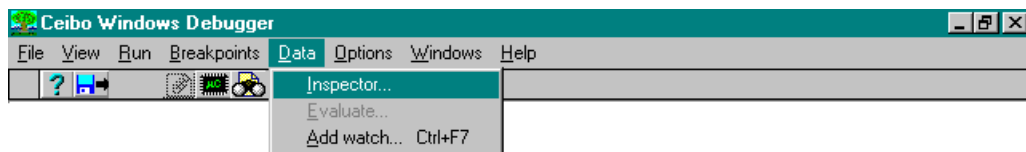


FIGURE 5.25: *Data Menu*

The following commands are available: Inspector, Evaluate 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.

Evaluate

The Evaluate command opens a dialog box which prompts you for an expression to evaluate. Then evaluates it according to the selected language in the Options menu.

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.

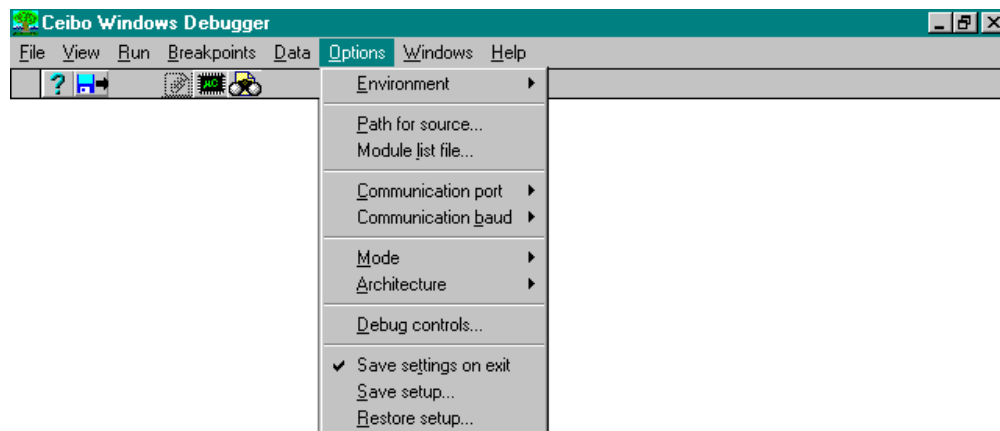


FIGURE 5.26: Options Menu

The following are the available options: Environment, Path for Source, Module List File, Communication Port, Communication Baud, Mode, Architecture, 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.

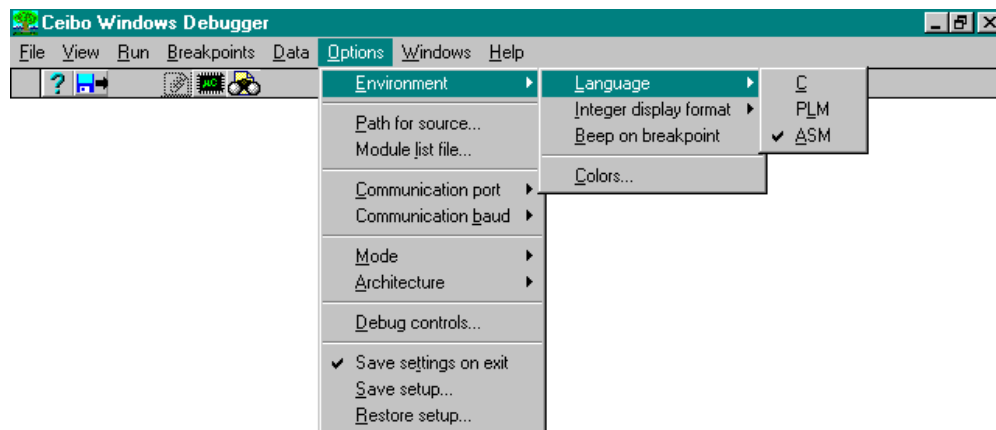


FIGURE 5.27: Environment Options

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.

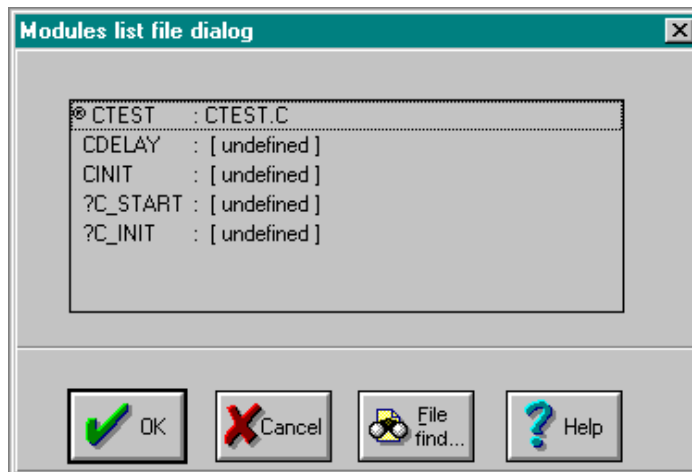


FIGURE 5.28: Module List Dialog Box

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.

Communication Baud

The Communication Baud option allows selection of the RS-232 interface baud rate to be used for communication with the remote emulator system.

You can toggle between High and Low options. Low baud rate is set to 9600 baud, and High baud rate is set to the maximum baud rate possible at time of selection, up to the maximum of 115K baud.

The baud rate synchronization is done in the software. There is no need to change any setting on the remote emulator system, whenever changing the baud rate.

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 DS-251 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 and also not implemented in the current software version; only basic functions are supported and not all SFRs belonging to particular derivatives. Set the system to emulation mode, while using the debugger with DS-251.

Architecture

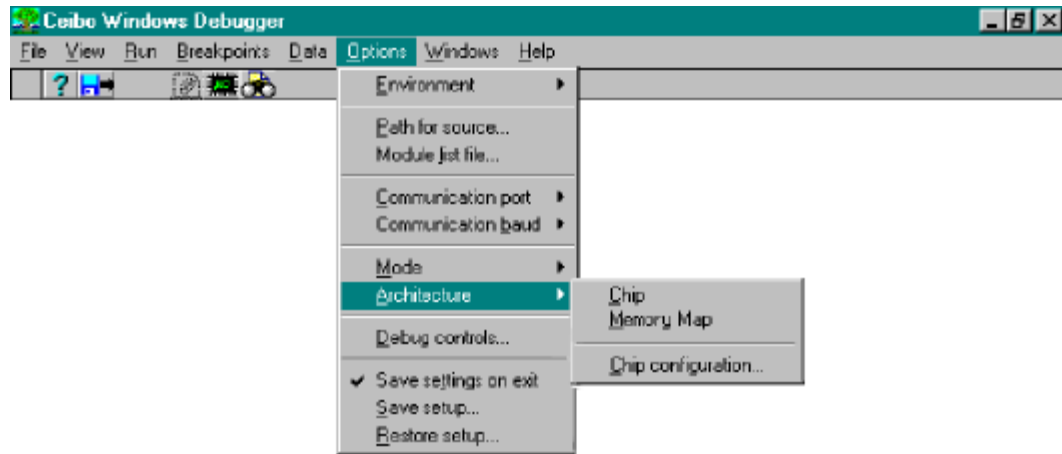


FIGURE 5.29: Architecture Menu

The Architecture option allows you to control and configure the remote emulator system options.

The available options are: Chip, Memory Map and Chip Configuration.

Chip: This command selects the emulated microcontroller type.

Memory Map: The Memory Map command allows you to define the memory spaces as belonging to the emulator or to your target hardware.

Chip Configuration: You can define the chip configuration with this command and initialize all the parameters of the Options Menu/Chip Configuration.

The available options are:

- 1 Source or binary compatibility,
- 2 Interrupt mode 51 or 251,
- 3 Number of unit states,

-
- 4 Page mode, ALE type and EPROM mapping,
 - 5 Extended addressability.

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.

Debug Control

Determines special hardware setup. Definines in which case a program is automatically reloaded. Sets the origin while halting the emulation.

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.

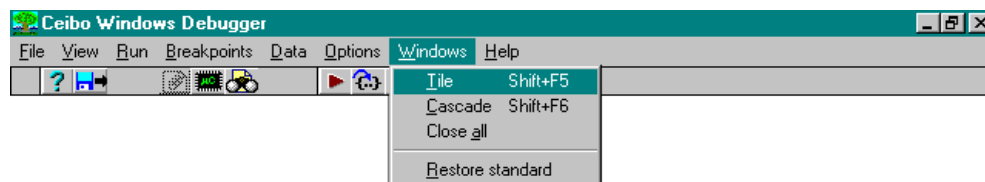


FIGURE 5.30: *Windows Menu*

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.

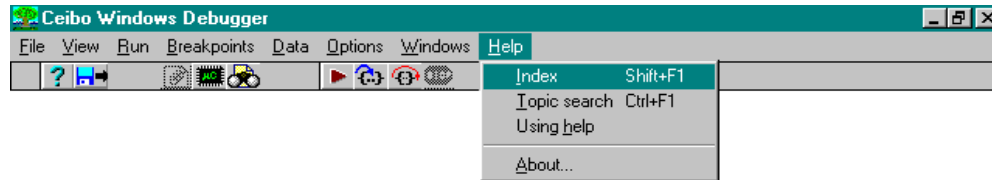


FIGURE 5.31: Help Menu

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



On-Line Assembler

CHAPTER 6



On-Line Assembler

6.1. Introduction

DS-251 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.

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.

6.2. Notation for Instruction Operands

Register Notation		MCS@251 Arch.	MCS 51 Arch.
@Ri	A memory location (00HFFH) addressed indirectly via byte register R0 or R1		✓
Rn	Byte register R0R7 of the currently selected register bank		
n	Byte register index: n = 07		✓
r r r r	Binary representation of n		
Rm	0R15 of the currently selected register file		
Rmd	Destination register		
Rms	Source register	✓	
m, md, ms	Byte register index: m, md, ms = 015		
s s s s	Binary representation of m or md		
S S S S	Binary representation of ms		
WRj	Word register WR0, WR2, ..., WR30 of the currently selected register file		
WRjd	Destination register		
WRjs	Source register		
@WRj	A memory location (00:0000H-00:FFFFH) addressed indirectly through word register WR0WR30		
@WRj +dis16	Data RAM location (00:0000H-00:FFFFH) addressed indirectly through a word register (WR0WR30) + displacement value	✓	
j, jd, js	Word register index: j, jd, js = 030		
t t t t	Binary representation of j or jd		
T T T T	Binary representation of js		
DRk	Dword register DR0, DR4, ..., DR28, DR56, DR60 of the currently selected register file		
DRkd	Destination Register		
DRks	Source Register		
@Drk	A memory location (00:0000H-FF:FFFFH) addressed Indirectly through dword register DR0DR28, DR56, DR60		

@DRk +dis24	Data RAM location (00:0000HFF:FFFFH) addressed indirectly through a dword register (DR0DR28, DR56, DR60) + displacement value	✓	
k, kd, ks	Dword register index: k, kd, ks = 0, 4, 8, ..., 28, 56, 60		
u u u u	Binary representation of k or kd		
U U U U	Binary representation of ks		

TABLE 6.1: *Notation for Register Operands*

Direct Address.	Description	MCS®251 Arch.	MCS 51 Arch.
dir8	An 8-bit direct address. This can be a memory address (00:0000H-00:00FFH) or an SFR address (S:00H - S:FFH).	✓	✓
dir16	A 16-bit memory address (00:0000H-00:FFFFH) used in direct addressing.	✓	

TABLE 6.2: *Notation for Direct Addresses*

ata	Description	MCS®251 Arch.	MCS 51 Arch.
#data	An 8-bit constant that is immediately addressed in an instruction.	✓	✓
#data16	A 16-bit constant that is immediately addressed in an instruction.	✓	
#0data16 #1data16	A 32-bit constant that is immediately addressed in an instruction. The upper word is filled with zeros (#0data16) or ones (#1data16).	✓	
#short v v	A constant, equal to 1, 2, or 4, that is immediately addressed in an instruction. Binary representation of #short.	✓	

TABLE 6.3: *Notation for Immediate Addressing*

Bit Address	Description	MCS®251 Arch.	MCS 51Arch.
bit y y y	A directly addressed bit in memory locations 00:0020H-00:007FH or in any defined SFR. A binary representation of the bit number (0-7) within a byte.	✓	
bit51	A directly addressed bit (bit number =00H-FFH) in memory or an SFR. Bits 00H-7FH are the 128 bits in byte locations 20H-2FH in the on-chip RAM. Bits 80H-FFH are the 128 bits in the 16SFR's with addresses that end in 0H or 8H: S:80H, S:88H, S:90H, . . . , S:F0H, S:F8H.		✓

TABLE 6.4: *Notation for Bit Addressing*

Destination Address	Description	MCS®251 Arch.	MCS 51Arch.
rel	A signed (two complement) 8-bit relative address. The destination is -128 to +127 bytes relative to first byte of the next instruction.	✓	✓
addr11	An 11-bit destination address. The destination is in the same 2-Kbyte block of memory as the first byte of the next instruction.	✓	✓
addr16	A 16-bit destination address. A destination can be anywhere within the same 64-Kbyte region as the first byte of the next instruction.	✓	✓
addr24	A 24-bit destination address. A destination can be anywhere within the 16-Mbyte address space.	✓	

TABLE 6.5: *Notation for Destinations in Control Instructions*

Bin.	0	1	2	3	4	5	6-7	8-F
Src.	0	1	2	3	4	5	A5x6A5x7	A5x8A5xF
0	NOP	AJMP addr11	LJMP 16	RR A	INC A	INC dir8	INC @Ri	INC Rn
1	JBC bit,rel	ACALL addr11	LCALL 16	RRC A	DEC A	DEC dir8	DEC @Ri	DEC Rn
2	JB bit,rel	AJMP addr11	RET	RLA	ADD A,#data	ADD A,dir8	ADD A,@Ri	ADD A,Rn
3	JNB bit,rel	ACALL addr11	RETI	RLCA	ADDC A,#data	ADDC A,dir8	ADDC A,@Ri	ADDC A,Rn
4	JC rel	AJMP addr11	ORL dir8,A	ORL dir8,#data	ORL A,#data	ORL A,dir8	ORL A,@Ri	ORL A,Rn
5	JNC rel	ACALL addr11	ANL dir8,A	ANL dir8,#data	ANL A,#data	ANL A,dir8	ANL A,@Ri	ANL A,Rn
6	JZ rel	AJMP addr11	XRL dir8,A	XRL dir8,#data	XRL A,#data	XRL A,dir8	XRL A,@Ri	XRL A,Rn
7	JNZ rel	ACALL addr11	ORL CY,bit	JMP @A+DPTR	MOV A,#data	MOV 8,#data a	MOV @Ri,#data	MOV Rn,#data
8	SJMP rel	AJMP addr11	ANL CY,bit	MOVC A,@A+PC	DIV AB	MOV dir8,dir8	MOV dir8,@Ri	MOV dir8,Rn
9	MOV DPTR, #data16	ACALL addr11	MOV bit,CY	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir8	SUBB A,@Ri	SUBB A,Rn
A	ORL CY,bit	AJMP addr11	MOV CY,bit	INC DPTR	MUL AB	ESC	MOV @Ri,dir8	MOV Rn,dir8
B	ANL CY,bit	ACALL addr11	CPL bit	CPL CY	CJNE A,#data,rel	CJNE A,dir8,rel	CJNE 1	CJNE 1
C	PUSH dir8	AJMP addr11	CLR bit	CLR CY	SWAP A	XCH A,dir8	XCH A,@Ri	XCH A,Rn
D	POP dir8	ACALL addr11	SETB bit	SETB CY	DA A	DJNZ dir8,rel	XCHD A,@Ri	DJNZ Rn,rel
E	MOVX A, @DPTR	AJMP addr11	MOVX A,@Ri		CLR A	MOV A,dir8	MOV A,@Ri	MOV A,Rn
F	MOV A	addr11	MOVX @Ri,A		CPL A	MOV dir8,A	MOV @Ri,A	MOV Rn,A

TABLE 6.6: Instructions for MCS® 51 Microcontrollers

Bin	A5x8	A5x9	A5xA	A5xB	A5xC	A5xD	A5xE	A5xF
Src	x8	x9	xA	xB	xC	xD	xE	xF
0	JSLE rel	MOV +dis	j,Rm	INC R,#short (1) MOV reg,ind			SRA reg	
1	JSG rel	MOV +dis ,Rm	j,Rm	DEC R,#short (1) MOV ind,reg			SRL reg	
2	JLE rel	MOV +dis			ADD Rm,Rm	ADD WRj,WRj	ADDr eg,op2 (2)	ADD DRk,DRk
3	JG rel	MOV +dis ,Rm					SLL reg	
4	JSL rel	MOV +dis			ORL Rm,Rm	ORL WRj,WRj	ORL reg,op2 (2)	
5	JSGE rel	MOV +dis ,WRj			ANL Rm,Rm	ANL WRj,WRj	ANL reg,op2 (2)	
6	JE rel	MOV +dis			XRL Rm,Rm	XRL WRj,WRj	XRL reg,op2 (2)	
7	JNE rel	MOV +dis ,WRj	MOV op1,reg (2)		MOV Rm,Rm	MOV WRj,WRj	MOV reg,op2 (2)	MOV DRk,DRk
8		LJMP @Wrj EJMP @DRk	EJMP addr24		DIV Rm,Rm	DIV WRj,WRj		
9		LL @DRk	ECALL addr24		SUB Rm,Rm	SUB WRj,WRj	SUB reg,op2 (2)	SUB DRk,DRk
A		Bit Instruc- tions (3)	ERET		MUL Rm,Rm	MUL WRj,WRj		
B		TRAP			CMP Rm,Rm	CMP WRj,WRj	CMP reg,op2 (2)	CMP DRk,DRk
C			PUSH op1 (4) MOV DRk,PC					
D			POP op1 (4)					
E								
F								

TABLE 6.7: New Instructions for the MCS®251 Architecture

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
Oper Rmd,Rms	x	C	md	ms	#data		#data (low)
Oper WRjd,WRjs	x	D	jd/2	js/2			
Oper DRkd,DRks	x	F	kd/4	ks/4			
Oper Rm,#data	x	E	m	0000			
Oper WRj,#data16	x	E	j/2	0100			
Oper DRk,#data16	x	E	k/4	1000	#data (high)		#data (low)
MOV Drk(h),#data16 MOV Drk,#1data16 CMP DRk,#1data16	77B	AEE	k/4	1100	#data (high)		#data (low)
Oper Rm,dir8	x	E	m	0001	dir8 addr		dir16 addr (low)
Oper WRj,dir8	x	E	j/2	0101	dir8 addr		
Oper DRk,dir8	x	E	k/4	1101	dir8 addr		
Oper Rm,dir16	x	E	m	0011	dir16 addr (high)		
Oper WRj,dir16	x	E	j/2	0111	dir16 addr (high)		
Oper Drk,dir16 †	x	E	k/4	1111	dir16 addr (high)		dir16 addr (low)
Oper Rm,@WRj	x	E	j/2	1001	m	00	dir16 addr (low)
Oper Rm,@DRk	x	E	k/4	1011	m	00	

TABLE 6.8: Data Instructions

x	Operation	Notes
2	ADD reg,op2	All addressing modes are supported.
9	SUB reg,op2	All addressing modes are supported.
B	CMP reg,op2	Does not support reg,op2 = DRk,direct16.
4	ORL reg,op2	Does not support double-word operations.
5	ANL reg,op2	Does not support double-word operations.
6	XRL reg,op2	Does not support double-word operations.
7	MOV reg,op2	All addressing modes are supported.
8	DIV reg,op2	Two modes only:reg,op2 = Rmd,Rms reg,op2 = Wjd,Wjs
A	MUL reg,op2	Two modes only:reg,op2 = Rmd,Rms reg,op2 = Wjd,Wjs

TABLE 6.9: High Nibble, Byte 0 of Data Instructions

6.3. Bit Instructions

All of the bit instructions in the MCS 251 architecture ("New Instructions for the MCS 251 Architecture") have opcode A9, which serves as an escape byte (similar to A5). The high nibble of byte 1 specifies the bit instruction, as given in Bit Instructions.

Instruction		Byte 0(x)		Byte 1			Byte 2	Byte 3
1	Bit Instr (dir8)	A	9	xxxx	0	bit	dir8 addr	

TABLE 6.10: Bit Instructions

xxxx	Bit Instruction
0001	JBC bit
0010	JB bit
0011	JNB bit
0111	ORL CY,bit
1000	ANL CY,bit
1001	MOV bit,CY
1010	MOV CY,bit
1011	CPL bit
1100	CLR bit
1101	SETB bit
1110	ORL CY, /bit
1111	ANL CY, /bit

TABLE 6.11:Byte 1 (High Nibble) for Bit Instructions

Instruction	Byte 0(x)		Byte 1		Byte 2	Byte 3
PUSH #data	C	A	0000	0010	#data	
PUSH #data16	C	A	0000	0110	#data16 (high)	#data16 (low)
PUSH Rm	C	A	m	1000		
PUSH WRj	C	A	j/2	1001		
PUSH DRk	C	A	k/4	1011		
MOV DRk,PC	C	A	k/4	0001		
POP Rm	C	A	m	1000		
POP WRj	C	A	j/2	1001		
POP DRk	C	A	k/4	1011		

TABLE 6.12: PUSH/POP Instructions

Instruction	Byte 0(x)		Byte 1		Byte 2	Byte 3
EJMP addr24	8	A	addr[23:16]		addr[15:8]	addr[7:0]
ECALL addr24	9	A	addr[23:16]		addr[15:8]	addr[7:0]
LJMP @WRj	8	9	j/2	0100		
LCALL @WRj	9	9	j/2	0100		
EJMP @DRk	8	9	k/4	1000		
ECALL @DRk	8	9	k/4	1000		
ERET	A	A				
JE rel	8	8	rel			
JNE rel	7	8	rel			
JLE rel	2	8	rel			
JG rel	3	8	rel			
JSL rel	4	8	rel			
JSGE rel	5	8	rel			
JSLE rel	0	8	rel			
JSG rel	1	8	rel			
TRAP	B	9				

TABLE 6.13: Control Instructions

Instruction	Byte 0		Byte 1		Byte 2	Byte 3
MOV Rm,@WRj+dis	0	9	m	j/2	dis[15:8]	dis[7:0]
MOV WRk,@WRj+dis	4	9	j/2	k2	dis[15:8]	dis[7:0]
MOV Rm,@DRk+dis	2	9	m	k/4	dis[15:8]	dis[7:0]
MOV WRj,@DRk+dis	6	9	j/2	k/4	dis[15:8]	dis[7:0]
MOV @WRj+dis,Rm	1	9	m	j/2	dis[15:8]	dis[7:0]
MOV @WRj+dis,WRk	5	9	j/2	k2	dis[15:8]	dis[7:0]
MOV @DRk+dis,Rm	3	9	m	k/4	dis[15:8]	dis[7:0]
MOV @DRk+dis,WRj	7	9	j/2	k/4	dis[15:8]	dis[7:0]
MOVS WRj,Rm	1	A	j/2	m		
MOVZ WRj,Rm	0	A	j/2	m		
MOV WRj,@WRj	0	B	j/2	1000	j/2	0
MOV WRj,@DRk	0	B	k/4	1010	j/2	

					0	
MOV @WRj,WRj	1	B	j/2	1000	j/2	0
MOV @DRk,WRj	1	B	k/4	1010	j/2	0
MOV dir8,Rm	7	A	m	0001	dir8 addr	
MOV dir8,WRj	7	A	j/2	0101	dir8 addr	
MOV dir8,DRk	7	A	k/4	1101	dir8 addr	
MOV dir16,Rm	7	A	m	0011	dir16 addr (high)	
MOV dir16,WRj	7	A	j/2	0111	dir16 addr (high)	
MOV dir16,DRk	7	A	k/4	1111	dir16 addr (high)	
MOV @WRj,Rm	7	A	j/2	1001	m	0
MOV @DRk,Rm	7	A	k/4	1011	m	0

dir16 addr (low)
dir16 addr (low)
dir16 addr (low)

TABLE 6.14: Displacement/Extended MOVs

	Instruction	Byte 0		Byte 1		
1	INC Rm,#short	0	B	m	00	ss
2	INC WRj,#short	0	B	j/2	01	ss
3	INC DRk,#short	0	B	k/4	11	ss
4	DEC Rm,#short	1	B	m	00	ss
5	DEC WRj,#short	1	B	j/2	01	ss
6	DEC DRk,#short	1	B	k/4	11	ss

TABLE 6.15: *INC/DEC*

ss	#short
00	1
01	2
10	4

TABLE 6.16: *Encoding for INC/DEC*

	Instruction	Byte 0		Byte 1	
1	SRA Rm	0	E	m	0000
2	SRA WRj	0	E	j/2	0100
3	SRL Rm	1	E	m	0000
4	SRL WRj	1	E	j/2	0100
5	SLL Rm	3	E	m	0000
6	SLL WRj	3	E	j/2	0100

TABLE 6.17: *Shifts*

CHAPTER 7



System Errors and Troubleshooting

CHAPTER 7



System Errors and Troubleshooting

7.1. Introduction

This chapter describes the errors detected while operating DS-251 and answers the most common questions related to hardware and software. Please read carefully the error message and follow the indications that appear on the screen to solve the problem.

7.2. Error Description

Error #2 - Reset

This error indicates that the emulated microcontroller could not be properly reset.

Error #3 - Run

The Run Error means that the system is not able to start executing your program.

Error #4 - Halt

This error appears when the system cannot stop the program execution.

Error #5 - Update

The system failed to update register contents.

Error #6 - Hardware

An internal hardware failure occurred.

Error #7 - Hardware

An internal hardware failure occurred.

Error #8 - XTAL

The crystal oscillator of the emulated microcontroller does not operate.

Error #9 - Power Down

The emulated microcontroller is in the Power Down Mode.

Error #10 - Idle

The emulated microcontroller is in the Idle Mode.

Error #11 - Power Down or Idle

The emulated microcontroller is in the Idle or Power Down Mode.

Error #20 - Communications

The PC cannot communicate with DS-251.

Error #21 - Communications

An invalid command has been transmitted to DS-251.

Error #22 - Communications

An invalid command has been transmitted to DS-251.

Error #32 - Communications

DS-251 does not respond to the PC command.

Error #33 - Communications

There are missing bytes in the transmission.

Error #34 - Communications

This is a time-out error.

Error #35 - Communications

An unexpected system status has been received.

Error #36 - Communications

The system identifier is wrong.

Errors #37-#255 - Communications

The PC does not receive any response from DS-251.

Error #256 - Load Error

Wrong File Format.

Error #257 - Load Error

Your program uses banked memory.

Error #300 - User's Entry Error

Address is not from XDATA memory.

Error #301 - User's Entry Error

Address is not from CODE memory.

Error #302 - User's Entry Error

Undefined cycle.

Error #303 - User's Entry Error

Unrecognized input

Error #304 - User's Entry Error

Unrecognized end of line.

Error #305 - User's Entry Error

Trace point does not exist.

Error #306 - User's Entry Error

Path is not correct.

Error #307 - User's Entry Error

Out of range error.

Error #308 - User's Entry Error

The watch value cannot be changed.

Error #309 - User's Entry Error

The trigger event must be set.

Error #310 - User's Entry Error

Invalid passcount entered; Passcount must be 1 to 32767.

Error #311 - User's Entry Error

Invalid memory space identifier.

Error #312 - User's Entry Error

Symbol not found.

Error #313 - User's Entry Error

Search expression not found.

Error #314 - User's Entry Error

Space cannot be dynamically updated.

Error #315 - User's Entry Error

The option cannot be provided while Running.

Error #316 to #450 - User's Entry Error

Error detected while entering a value or symbol.

Error #800 & #801 - System Error

This is an internal error; call CEIBO for technical assistance.

Error #802

Demo version can load only 1K code.

Error #803

Out of memory space.

Error #804

File operation failed.

Error #805 - to #900 - System Error

This is an internal error; call CEIBO for technical assistance.

7.3. Common Questions and Answers

ROMless Operation

1. How does the system work in ROMless mode?

ROMless operation means that the chip selected is for example 80C251SB, and ports lines are the bus. Instructions are fetched from the DS-251 emulation memory if addresses are below the boundary defined in the Options Menu (4K, 8K, 16K, 32K and 64K). Above the boundary, instructions are fetched from your target circuit, so you must have the emulation header connected to a working circuitry.

ROMed Operation

2. *How does the system work in ROMed mode?*

ROMled operation means that the chip selected is for example 87C251SB, and ports lines are not affected by instruction fetch cycles.

Ports

3. *I cannot access (read or write) Port 0 and 2.*

Select chip type ROMed (like 87C251SB) and not ROMless (like 80C251SB).

Timers

4. *How many timer ticks are lost while stopping the emulation and reassuming it?*

As the system uses a bond-out chip, no ticks are lost while stopping the emulation.

System Problems

5. *Yesterday I worked with the system and it was OK. Today the ports are not showing any activity.*

The software has been invoked without a DS-251 system connected to the power supply; then the mode has changed from Emulation to Simulation and therefore you are working only with the simulator.

6. *I cannot establish a communication between the system and my PC, although the serial port of my computer works with another system as well as the serial cable.*

1. A cable from another system may be the problem. Use only the cable supplied with the system.
2. Your PC does not support the high baud rate. Try to set the baud rate to low in the Options Menu.

7. *The system shows always Error #6 - crystal problems.*

Set the crystal jumper to Internal.

8. *The system does not work at 24 MHz.*

The Intel bond-out chip may not always support that frequency.

Software

9. *Why cannot Port 0 and 2 be accessed in the watches window, etc. ?*

Select chip type ROMed (like 87C251SB) and not ROMless (like 80C251SB).

10. *Why some options are grayed out?*

Not available yet in the current software version.

11. *I cannot open the Module Window.*

You did not load a file with DEBUG information. Check again how the file has been generated and if you selected the appropriate vendor in the Load command.

12. *Windows is reporting error code out of memory, etc., when I try to open any dialog or window.*

The error code may be caused by incomplete setup. Please copy BWCC.DLL from ceibo directory to your Windows system directory (i.e. WINDOWS\SYSTEM or WIN95\SYSTEM).

Index

A

- About
 - the Manual, III
 - the Windows Debugger, 3-1
- About Command, 5-35
- Add Watch, 5-27
- Add, 5-5
- Address, 5-4
 - Match, 5-4, 5-19
- Animate, 5-24
- Answers, Common, 7-5
- Applications, 1-2
- Architecture, 5-31
- Assemble, 5-11
- Assembler, 1-2, 6-1, II
 - On-Line, 1-2, 6-1

B

- B, 4-5, 5-9
- Baud, 5-30
- Beep, 5-28
- Block, 5-20
- Breakpoint Vector, 1-12
- Breakpoints, 1-3, 5-4, 5-25

C

- C, II, 1-2, 4-5, 5-9
- Capturing Watches, 4-8
- Change, 5-9, 5-12, 5-20
- Change, Memory Global, 5-26
- Changing, 3-2, 4-4, 4-7
- Chip, 5-31
- Clock Oscillator, 1-1, 1-6

- Code Memory, 1-2
- Code, 5-31,
- COM, 2-1, 2-2
- Common Answers, 7-5
- Communication Port, 5-30
- Components, 2-2
- Configure, 5-14
- Connectors, 1-8
- Continuous Run, 5-24
- CPU, 5-10
- Cycle, 5-4, 5-19

D

- D, 4-5, 5-9
- Data 1-10, 5-9, 5-31
- Data Menu, 5-26
- Data Support, 1-10
- DC Power Jack, 1-6
- Debug Capabilities, 3-1
- Debug Controls, 5-32
- Debugger
 - Source-Level, 1-3
 - Symbolic, 1-2
 - Windows, II, 1-2, 2-3, 3-1
- Debugging
 - the Program, 4-9
 - Windows Session,
- Decrement, 5-12
- Delete All, 5-5, 5-9, 5-26
- Description of the System, III, 1-1
- Dialog Box, 3-3
- Disassembler, 6-1
- Display Mode, 5-16
- Dual Event, 5-18

E

- Edit, 5-9

- Emulation
 - Header, 1-7 - 1-8
 - Memory, 5-1
 - Real-Time, II, 7-1
 - Restrictions, II, 1-9
- Environment, 5-28
- Errors, II, 7-1
- Evaluate, 5-27
- Execute
 - Forever, 5-22
 - To Cursor, 5-23
- Exit, 5-3
- Expression True Global, 5-26, 7-3

F

- Features, I
- File, 4-7, 5-1
- Filters, 5-16
- Frequency, 1-1, 1-4, 1-6
- From Event, 5-17

G

- Get Info, 5-3
- Global Menus, 3-2, 4-2
- Globals, 5-5, 5-26
 - Change Memory, 5-26
 - True Expression, 5-26
- GND, 1-6
- Goto, 5-10, 5-12, 5-16, 5-20

H

- Halt, 5-25
 - Ends, 5-17
- Hardware,
 - Description, 1-5

- Header, 2-6
- Help, 5-34
- Hex, 5-1
- High-Level Language, 6-2
- Host Characteristics, 1-4

I

- Icon, 4-1
- Increment, 5-12
- Index, 5-35
- Info, 5-15
- Input
 - Boxes, 3-3
 - Power, 1-4
- Inspect, 5-16
- Inspect, 5-5, 5-7, 5-9
- Inspecting, 3-2
- Inspector, 5-27
- Installation, III, 2-1, 2-3
- Instruction,
 - Consecutive, 7-3
 - Trace, 5-24
- Integer Format, 5-28

L

- Lines, 5-8
- Load, 5-1
 - a File, 4-7
- Locals, 5-5

M

- Map, 5-31
- Memory
 - Code, 1-3, 1-9, 5-31
 - External, 4-5
 - Internal, 4-5

- Spaces, 4-5, 4-6, 5-20
- System, 1-2
- Menus, 5-1
 - Breakpoints, 5-4
 - Data, 5-26
 - File, 5-1
 - Global, 3-2, 4-3
 - Help, 5-34
 - Local, 3-3, 4-2
 - Options, 5-27
 - Run, 5-22
 - Using the, 3-4
 - View, 5-3
 - Windows, 5-34
- Microcontrollers
 - Selection, 1-3
- Mode, 5-30
 - Emulation, 5-30
 - Real-Time, 1-1
 - Simulation, 1-2, 4-3, 5-21
- Module, 5-6
 - List, 5-6, 5-29
 - Local Menu, 5-7

N

- New PC, 5-8, 5-11
- New, 5-3
- Next, 5-8, 5-20

O

- OMF251, 5-1
- Options, 5-4, 5-27
 - Interrupt, 5-32
 - Restore, 5-34
 - Save, 5-34
- Origin, 5-8, 5-11, 5-12, 5-16
- Oscillator,
 - Clock, 1-1, 1-4

P

- P, 4-6, 5-9
- Passcount, 5-4, 5-19
- Path for Source, 5-29
- Performance Analyzer, 5-13
- Polled, 5-33
- Power Supply, 1-9
- Print to File, 5-11, 5-17
- Program Reset, 5-25
- Publics, 5-5

R

- RAM, 1-2, 1-3, 4-5
- Real-Time Emulation, II, 7-1
- Refresh, 5-15
- Registers, 5-12
- Remove, 5-5, 5-9
- Reset, 5-25, 5-35, 7-3, 7-5
- Restore, 5-34
- Restrictions, III, 1-9
- ROM/ROMless Support, 1-8
- RS-232
 - Cable, 2-1
 - Interface, 2-1
- Run Begins, 5-17
- Run, 5-22

S

- Save, 5-34
- Search, 5-8, 5-20
- Set Options, 5-4
- SFR, 4-5, 5-9
- SFR, 6-1
- Simulation, II, 1-1

Simulator, 1-1, 4-3, 5-30

Software

Installation, 2-2

Packages, II

Preparing the, 4-1

Trace, 1-2

User, 1-3

Specifications, 1-2

Stack Pointer, 7-3

Start Event, 5-18

Starting-Up, 2-3

Startup Skip, 5-2

Status Line, 3-5

Stepping, 3-1

Stop Event, 5-18

Stop Over, 5-23

Symbols, 5-2, 5-9

Syntax, 5-9

System,

Errors, 7-1

Memory, 1-2

T

Target, 5-21

Topic, 5-35

Search, 5-35

Trace, 1-3, 5-15

Buffer, 5-15

Clear, 5-16

Dump, 5-15

Instructions, 5-24

Into, 5-23

Read All, 5-17

Software, 1-2, 1-3

Status, 5-16

Tracing, 3-1

Triggers, 5-16, 5-17, 5-19

Troubleshooting, III, 7-1

U

Until Event, 5-18

Using the Menus, 3-4

V

Variables, 5-5

View Menu, 5-3

Viewing, 3-2

W

Watches, 4-3, 4-4, 5-6, 5-7, 5-8

Adding, 4-3, 5-27

Capturing, 4-8

Changing, 4-4, 4-7

Local Menu, 5-8

Watching, 3-2, 4-5

Windows, 3-3, 4-7, 4-8

Changing, 4-7

Debugging Session, II, 4-1

Menu, 5-34

Menus and Commands, II, 5-1

Z

Zero, 5-13