DS-51 Development System

User's Manual

Rev. 05/14 - V1.1

POWER UP SEQUENCE

Vcc from the emulator is disconnected in the emulator plug. For example, Pin 40 for the DIP package is disconnected in the emulator plug on the emulation header. Therefore, follow the instructions given below before connecting the power supply to your target or to the emulator.

1. Power On Sequence

The emulator comes with its own power supply.

Vcc in the emulator header is disconnected.

The emulator Ground is connected to the emulator header.

Therefore, the emulator can work stand alone, meaning without any target.

2. Emulator Without Target

If you will use the ***emulator without a target***, proceed to turn on the power supply.

3. Emulator With Target

While using the ***emulator with a target board*** follow these steps:

a. Make sure that all the power supplies you may have are OFF.

b. You should have two power supplies: one for the emulator and the other for your target. **Turn on both power supplies from a common power switch**. If you do not have it, turn on first the target and then the emulator power supply**.**

4. Emulator With Target and ONCE Support

While using the ***emulator with a target board*** that needs to enter in ONCE mode (a soldered Microcontroller on the board has to be set to 3-state mode), turn on **first the target** and then the emulator power supply.

Do not leave the emulator powered while connected to a target unpowered

CONTENTS

PREFACE

CHAPTER 1. DESCRIPTION OF THE SYSTEM

CHAPTER 2. INSTALLATION

CHAPTER 3. ABOUT THE WINDOWS DEBUGGER

CHAPTER 4. WINDOWS DEBUGGING SESSION

CHAPTER 5. WINDOWS MENUS AND COMMANDS

CHAPTER 6. WORKING WITH THE HARDWARE AND REAL-TIME EMULATION

CHAPTER 7. UTILITIES, SOFTWARE SUPPORT AND SYNTAX

CHAPTER 8. ON-LINE ASSEMBLER

CHAPTER 9. SYSTEM ERRORS AND TROUBLESHOOTING

PREFACE

DS-51 Development System

PREFACE

DS-51 Development System

The DS-51 *Development System* for 80C51 Microcontrollers and derivatives is very effective in meeting the diverse demands of emulation operations.

P.1. Features

- Real-time and Transparent In-Circuit Emulator
- Supports Most of the 8051 Derivatives
- Emulates 1.5V to 6V Microcontrollers
- Maximum Frequency of 42MHz
- 128K/512K of Internal Memory
- Banking Support
- 32K Trace Memory "on the Fly"
- 64K Hardware and Conditional Breakpoints
- DOS and MS-Windows Debuggers
- Source Level Debugger for Assembler, PLM and C
- On-line Assembler and Disassembler
- Performance Analyzer
- Serially linked to IBM PC at 115 KBaud

P.2. The Software

The DS-51 is supplied with two software packages:

- C51D DOS Debugger
- Windows Debugger

Both debuggers have similar functions, although the Windows debugger is wholly based on a Windows platform (e.g., multi-tasking and multi-window display).

Description of the DOS debugger is given only in the on-line help.

Windows debugger functions are described in Chapters 3, 4 and 5.

The three working modes:

- Real time
- Simulator
- In-circuit simulator

are introduced in Chapter 1.

P.3. 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 10 gives some ***troubleshooting*** recommendations to follow in case that you are experiencing problems with your hardware or software.

P.4. About the Manual

1. Description of the System

Describes the system and Personality Probes, 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. Working with the Hardware and Real-Time Emulation

Provides a session example and gives all the information regarding the system in real-time Emulation Mode.

7. Utilities, Software Support and Syntax

Explains the use of utilities provided to adapt different assemblers and compilers to Ceibo's Debuggers.

8. 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.

9. System Errors and Troubleshooting

Lists the errors detected while operating DS-51 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 of the system and personality probes, as well as the emulation restrictions are detailed in the following paragraphs.

1.2. General Description

Ceibo DS-51 is a development system that supports most of the 8051 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.

Three working modes are available: real-time, simulator and in-circuit 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*** is used to debug the software without any hardware. Therefore, DS-51 may be disconnected while using the simulation mode.

In the ***in-circuit simulation mode*** an additional microprocessor is used to take control of the microcontroller lines and to simulate its operation but not in real-time. This operating mode allows access to all the microcontroller functions (I/O, timers, etc.) and interacts with the hardware according to the user software execution or directly by means of emulator commands sent from the host computer. This mode is available as part of Ceibo 8051 Debugger and it is ***not useful*** with DS-51 since all functions are supported in real-time.

The combination of all the available working modes allows an easy way to debug hardware and software functions.

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

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

1.3. Applications

The main applications of DS-51 Development System are:

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

1.4. Specifications

Emulator Memory

DS-51 provides 64K of user code memory and with some personality probes also 64K of user data memory. This RAM memory permits downloading and modifying of user programs and variables.

DS-51/512K system has a universal memory bank switching configuration for any 32K or 64K banks and up to 32 memory pages.

Code Memory

Code memory is mapped as belonging to the DS-51 Development System or to the target board. Also memory splitted mode is supported.

Data Memory

Data memory is accessed by MOVX instructions and can be mapped as belonging to the emulator or to the target circuitry. Some Personality Probes include 64 KBytes of RAM to be used as data memory.

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.

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.

User Software

The CEIBO DOS Debugger may be installed to run under DOS or Windows 3.X or later. The program is based on pull-down menus. The CEIBO Windows Debugger runs only under MS-Windows 98 or later.

Symbolic Debugger

DS-51 allows symbolic debugging of assembler or high-level languages. The symbolic debugger uses symbols contained in the absolute file generated by the most commonly used Assemblers and High Level Language Compilers.

Source Level Debugger

The DS-51 software includes a source level debugger for Assembler and High-Level Languages (PLM, C and C++) with the capability of executing lines of the program while displaying the state of any variable.

Real-Time Trace

This real-time memory is 32K deep and is used to record the microprocessor activities. Eight lines are user selectable test points and two lines provide start and stop triggers. Emulation is not stopped to display the trace buffer.

Supported Microcontrollers

The supported microcontrollers are most of the 80C51 microcontrollers and derivatives in both ROMless and ROMed versions. DS-51 also has different personality probes to emulate the microcontroller derivatives.

Microcontroller Selection

DS-51 uses standard and bond-out microcontrollers for hardware and software emulation. The selection of a different microcontroller is made by replacing the microcontroller on the probe or changing the probe.

Frequency

The DS-51 probes have an internally connected crystal. The XTAL jumper on the probe permits the use of a different clock rate supplied by the user circuitry. Maximum frequency is defined by the microcontroller or bondout installed on the probe.

Host Characteristics

PC or compatible systems with 8 MByte of RAM, one RS-232C interface card and MS-Windows 98 or later.

Input Power

5V, 1.5A power supply supplied.

Mechanical Dimensions

26 mm x 151mm x 195 mm.

Items Supplied as Standard

Development tool with 64K Breakpoints, 64K internal Code Memory, 32K Real-Time Trace, Personality Probe C51 for 8051 microcontrollers, Software including source level debugger, on-line assembler and disassembler, User Manual, RS-232 Cable and Power Supply.

Options

Personality probes and emulation headers for the different microcontrollers. Adapters for PLCC, QFP and other devices.

1.5. Emulation Restrictions

The following restrictions are valid for DS-51:

1. Only emulation at Vcc = 5V± 5% is possible with standard probes. The variable voltage probes allow emulation from 1.5V to 6V.

2. ALE cannot be disabled in emulation mode.

3. DS-51 uses 3 bytes of the internal stack memory to save register status when program execution is halted. The user has to leave this additional stack space unused.

4. Programs loaded into the emulator are stored in a RAM memory. Therefore, if a power failure occurs after the code has been loaded, code information is altered and the load operation must be repeated.

5. The simulator can simulate timers only in Timer Mode and while the Gate control is 0. Some other specific hardware and real-time functions

may not be supported by the simulator and in any case the ***emulation mode is always recommended***.

6. The simulation mode supports only the basic functions of the 8051 microcontrollers. Special registers and functions of some derivatives may be not supported in the simulation modes. Use the real-time ***emulation mode*** for fully support of dedicated functions.

1.6. Personality Probes

1.6.1. Probe P-C51

Most of the 80C51 devices are emulated by this personality probe. The technology used consists of using standard devices operating in a special licensed emulation mode. Therefore, you may use the same probe for many derivatives just by replacing the microcontroller on the probe. The maximum frequency of this probe is related to emulation mode and it is 20MHz to 33MHz, although it may change according to the innovations of the microcontrollers and derivatives. Both the probe and the emulator are prepared to run up to 42MHz. It emulates microcontrollers at 5V only and with 12 clocks/cycle. An example of the capabilities of a personality probe is given in the following table:

Microcontroller on the Probe	*Supported Microcontrollers*
P80C32 or P87C52	80C31/2, 8xC51/2/4/8, 89C55, 89C536/8
P87C51FB or P87L51FB	8xC51FA/B/C
P87C51RD+	8xC51RA/B/C/D+
P87C550 (DIP)	8xC550 (DIP)
P87C654	8xC652/4

Probe P-C51 and Supported Microcontrollers

Probe Adapter - it is the board installed ***inside the emulator box***. You need to open the emulator box to access this board (see next paragraph). It also has to be attached to the 100-pin ribbon cable.

Clock is jumper selectable from the built-in crystal oscillator installed on a socket (the user may replace it by any other) or any external source.

The clock jumper and the emulated microcontroller are on the Probe Adapter.

Emulation Header - it is the board attached to the ribbon cable and that has the plug with the mechanical characteristics of the emulated microprocessor. It has to be plugged into the target board socket.

The standard probe has a 40-DIP header, as shown in the figure. It has to be attached to the 100-pin ribbon cable.

Adapters for 44-PLCC and 44-QFP are available.

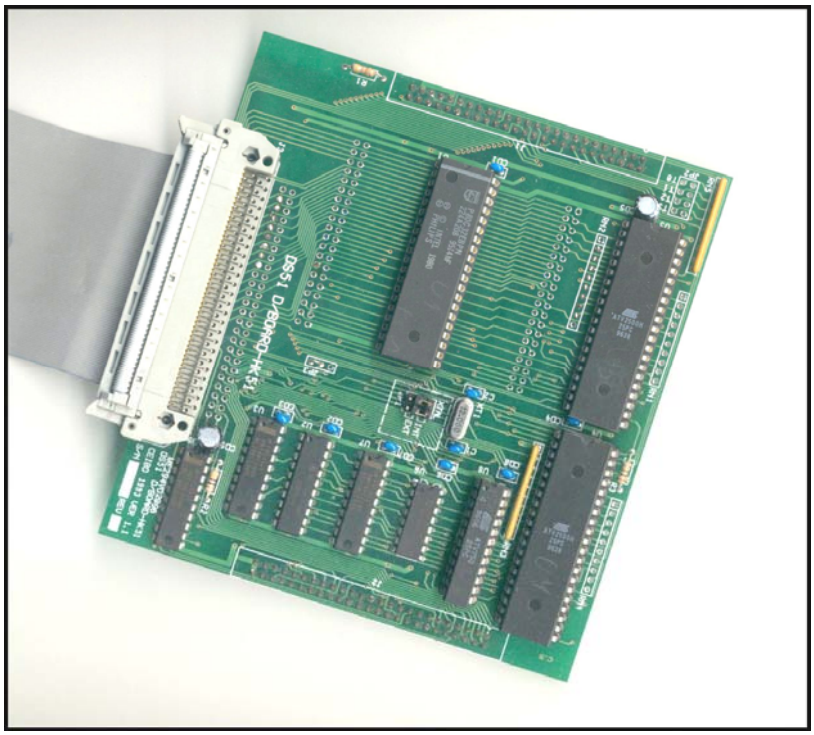

Figure 1.1: P-*C51 - Probe Adapter*

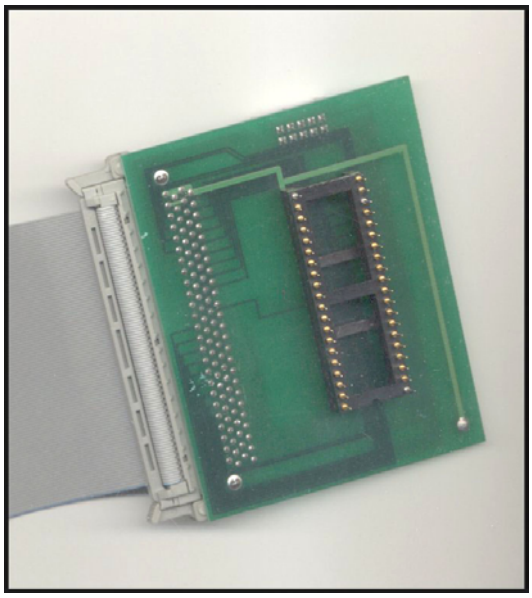

Figure 1.2: P-*C51 - Emulation Header*

The emulation header supplied with the probe is:

Emulation Header	*Description*
H-51-40D	40-pin DIP

Support for AT89C1051/2051/4051 20-pin devices is provided using Philips P80C32 or P87C52 microcontroller on the probe and ADP-20D 40-pin DIP to 20-pin DIP adapter. Also 20-SO adapters are available. The analog comparator option of AT89C1051/2051/4051 cannot be emulated.

1.6.2. Probe P-C51LV and P-C51xD2

Most of the 80C51 devices are emulated by this personality probe and has all the features of Probe C51. Additionally it emulates the devices with 6 clock/cycle such as 89C51RD2 and many Philips/Atmel/NXP other devicess.

The main differences between Probe C51 and probe P-C51LV/P-C51xD2 are:

Feature	*Probe P-C51*	*Probe P-C51LV/xD2*
Voltage	5V	3V and 5V
Frequency	33MHz	42MHz
Clock Source	Crystal	Programmable
Off-chip Xdata	no	64K
12 clock/cycle	yes	Yes
6 clock/cycle	no	Yes
Special Emulation Mode (*)	yes	Yes
Devices w/o Emulation Mode (*)	no	Yes

(*) see technologies.pdf in www.ceibo.com

Probe P-C51LV and Probe P-C51 - Differences

The technology used consists of using standard devices operating in a special licensed emulation mode to support ROM/ROMless applications. Also any devices without this special mode can be emulated; in that case the chip will work in ROMless mode only, where Port 0 and 2 are bus and not I/O (this is not important if the user code has MOVX instructions to access off chip memory or peripherals). More information about this can be found in technologies.pdf available in www.ceibo.com. Therefore, you may use the same probe for many derivatives just by replacing the microcontroller on the probe. The maximum frequency of this probe is related to emulation mode and it is 42MHz, although it may change according to the innovations of the microcontrollers and derivatives. Probe P-C51LV has an optional 64K data memory that can be mapped to the emulator and accessed by MOVX instructions. It supports 3V and 5V (jumper selectable), as well as 6 and 12clocks/cycle microcontrollers. An example of the capabilities of a personality probe is given in the following table:

Microcontroller on the Probe	*Supported Microcontrollers*
P80C32 or P87C52	80C31/2, 8xC51/2/4/8, 89C55, 89C536/8
P87C51FB or P87L51FB	8xC51FA/B/C
P87C51RD+	8xC51RA/B/C/D+
P87C654	8xC652/4
P89C51RD2	8xC51RA2/RB2/RC2/RD2
T89C51RD2	T8xC51RA2/RB2/RC2/RD2 (*)
AT89C51RD2	AT8xC51RA2/RB2/RC2/RD2 (*)
P89C668	8xC66X

(*) see technologies.pdf in www.ceibo.com

Probe P-C51LV and Supported Microcontrollers

The standard emulation header is 44-pin PLCC and adapters for 40-DIP and 44-QFP are available.

Clock is jumper selectable from the programmable clock source (controlled by software) or any external source.

The emulation header supplied with the probe is:

Emulation Header	*Description*
H-51-44P/C51LV	44-pin PLCC

Support for AT89C1051/2051/4051 20-pin devices is provided using Philips 80C32 or 87C52 microcontroller on the probe and ADP-20D 40-pin DIP to 20-pin DIP adapter. The analog comparator option of AT89C1051/2051/4051 cannot be emulated.

Probe C51LV has 3 jumpers:

ROM/ROMLESS - When the jumper cap is placed, the probe works in ROMless mode. Removing the jumper cap causes the probe to work in ROM mode.
ROM: means Port 0 and 2 are I/O and not bus.

ROMless: means Port 0 and 2 are bus only.

XTAL - This jumper in used to select the clock source. INT means the clock is programmable by a built-in clock generator; select the frequency from the options menu (debugger version V1.18 or later). EXT means belonging to your target. Please note that always 2 jumpers must be set.

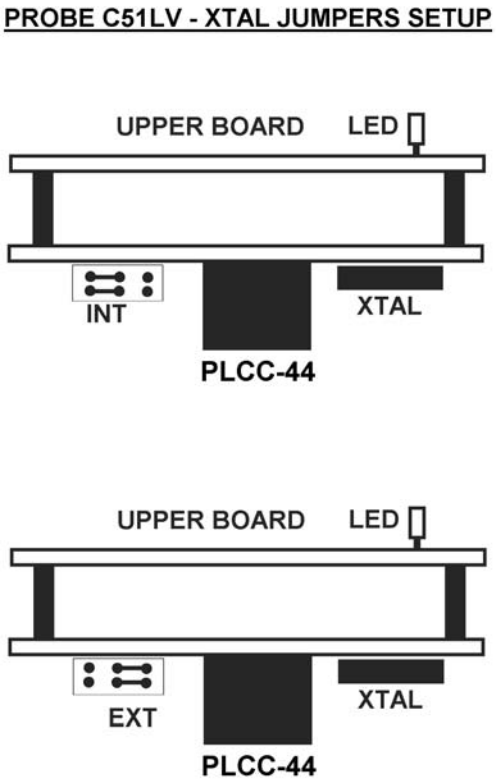

Figure 1.3: P-*C51LV - Jumper Setup*

3/5V - JP1 selects 3.3V or 5V emulation.

Emulated Microcontroller: It is on a socket, between the two boards and on a PLCC socket.

Maximum Frequency: depends on the Emulated Microcontroller, Voltage selection and Mode (X1 or X2).

Probe Adapter - it is the board installed ***inside the emulator box***. You need to open the emulator box to access this board. It also has to be attached to the 60-pin ribbon cable.

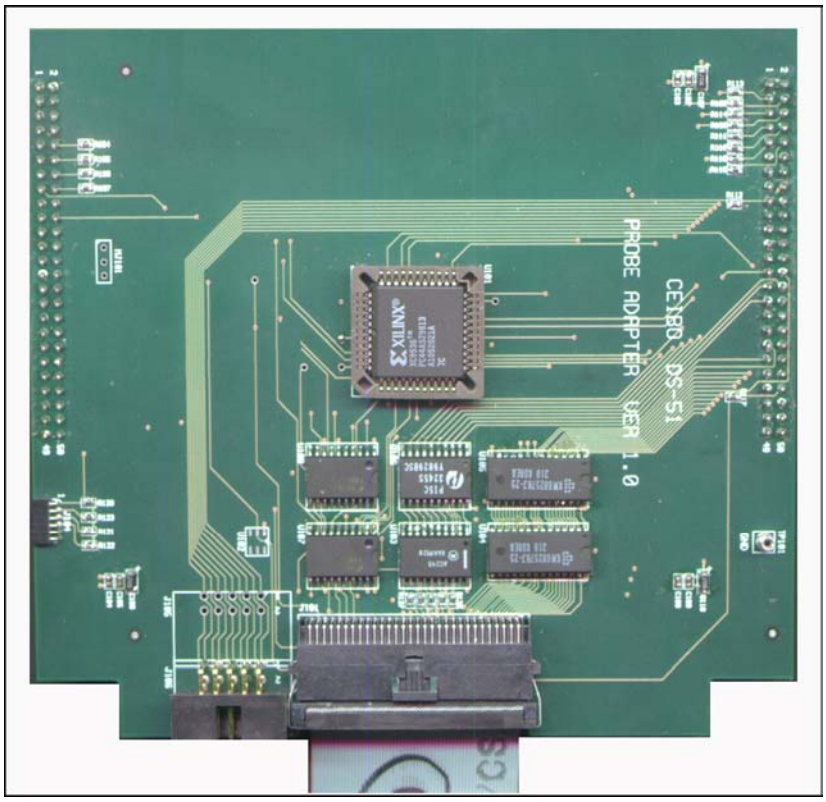

Figure 1.4: P-*C51LV - Adapter*

To connect this board to DS-51, open he emulator box. Then, align and press the adapter connectors on the J1 and J2 motherboard connectors until they are firmly plugged together.

1.6.3. PROBE P-C51XD2

Probe P-C51XD2 consists of the emulation header and probe adapter. Optionally 40-DIP and 44-QFP adapters are available.

Emulation Header - it is a single board attached to the ribbon cable and has a PLCC-44 plug on the bottom.

The board connects the 60-pin ribbon cable to the emulator. There are a few jumpers to select clock source, ROM/ROMless mode and voltage. The LED on it provides a power on indication.

ROM/ROMLESS - When the jumper cap is placed, the probe works in ROMless mode. Removing the JP1 jumper cap causes the probe to work in ROM mode.

ROM: means Port 0 and 2 are I/O and not bus. P3.6 and P3.7 are I/O and not RD/WR signals. Select this mode if you do not use MOVX instructions to access Xdata out of the chip.

ROMless: means Port 0 and 2 are bus only. P3.6/P3.7 are RD/WR lines.

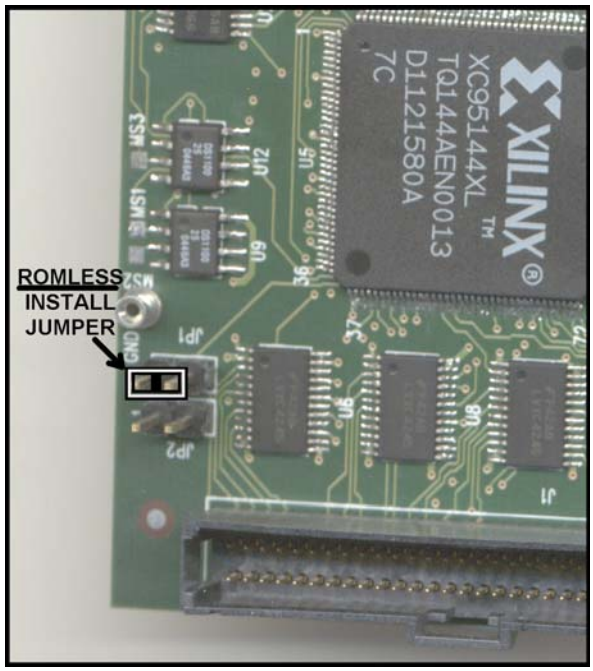

Figure 1.5: JP1 - *ROMless Jumper*

XTAL - This jumper in used to select the clock source. INT means the clock is programmable by a built-in clock generator; select the frequency from the options menu. EXT means belonging to your target. Please note that always 2 jumpers must be set and ***horizontally***.

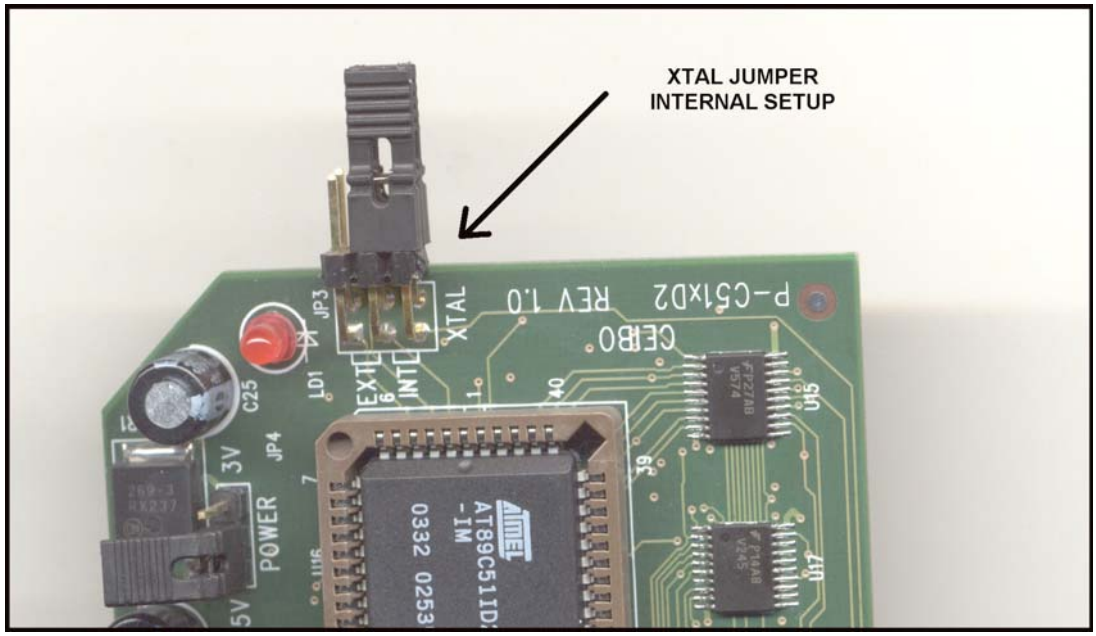

Figure 1.6*: JP3 - XTAL Jumpers*

The two jumper setup is more clearly shown in the following figure.

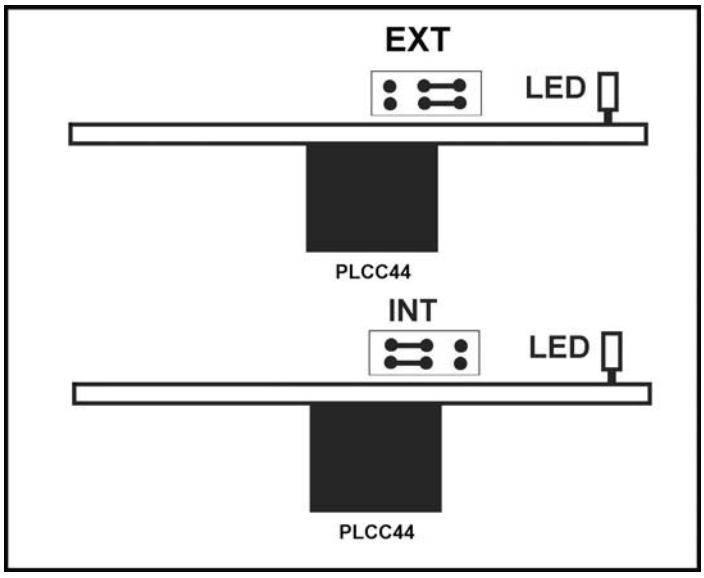

Figure 1.7: ***JP3 - XTAL INT and EXT Setup***

3/5V - JP1 selects 3.3V or 5V emulation.

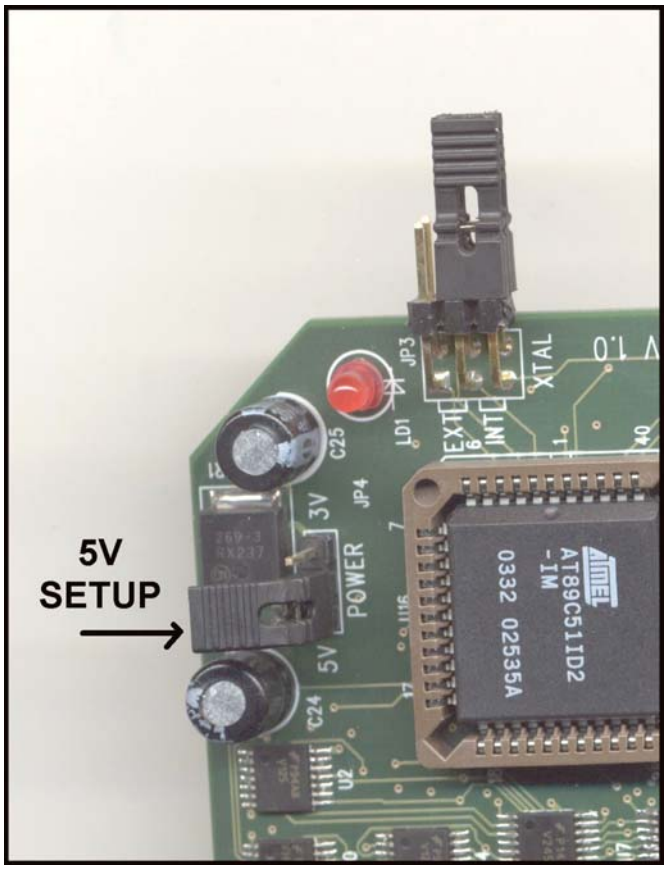

Figure 1.8: ***JP4 - 5V Setup***

Emulated Microcontroller: It is on a PLCC socket. Your system is supplied with an Atmel AT89C51ID2 microcontroller which supports also many other derivative.

Maximum Frequency: depends on the selected mode (ROM/ROMless), Voltage selection and frequency mode (X1 or X2).

ROM/ROMless	X1/X2	Voltage (V)	F Max (MHz)
ROM Mode - Port 0,2 and 3.6/3.7 are I/O	X1 - 12 clk/cycle	5	37
		3.3	33
	X2 - 6 clk/cycle	5	18
		3.3	16
ROMless Mode - Used for MOVX instructions that activate bus lines	X1 - 12 clk/cycle	5	47
		3.3	46
	X2 - 6 clk/cycle	5	23
		3.3	23

Probe Adapter - it is the board installed inside the emulator box. You need to open the emulator box to access this board. It also has to be attached to the 60-pin ribbon cable.

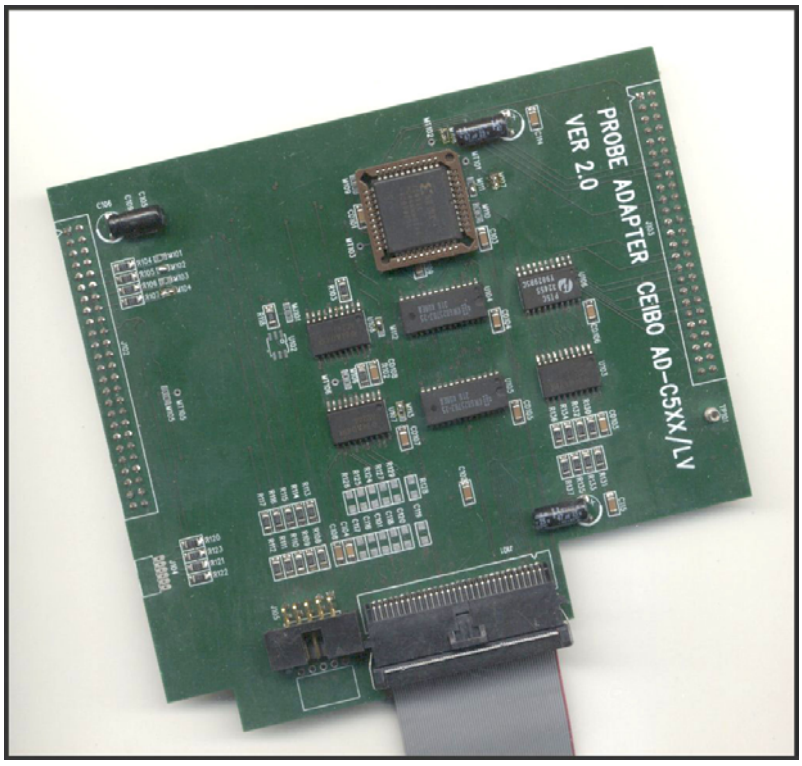

Figure 1.9: ***P-C51XD2 - Adapter***

To connect this board to DS-51, open he emulator box. Then, align and press the adapter connectors on the J1 and J2 motherboard connectors

until they are firmly plugged together. ***Trace clips are connected to J105.***

QFP Adapter - this is a soldering block that has on the top a PLCC socket. Solder this block to your target, and connect then the emulation header.

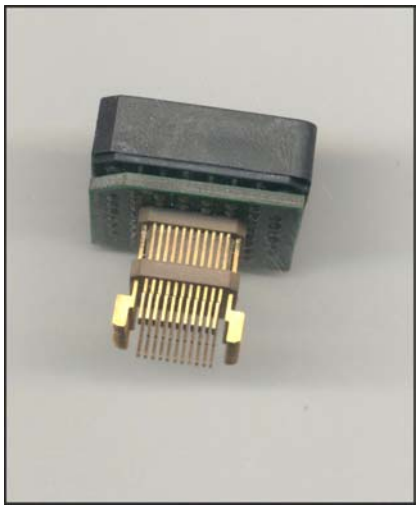

Figure 1.10: ***P- C51XD2 - QFP Soldering Block***

1.6.4. Probe P-C32

This probe is similar to C51, but does not use the microcontrollers in emulation mode. The advantage is that frequency is completely defined by the microcontroller on the probe. You may replace the microcontroller by any derivative from any vendor. As this probe does not require the special emulation mode, microcontrollers are emulated in ROMless mode only and up to 42MHz if the chip on the probe allows it. The standard emulation header is 40-pin DIP and adapters for 44-PLCC and 44-QFP are available.

Clock is jumper selectable from the built-in oscillator or any external source. Probe P-C32 includes 64 KByte data memory that can be mapped to the emulator and accessed by MOVX instructions. Emulation is at 5V only.

The emulation header supplied with the probe is:

Emulation Header	*Description*
H-51-40D	40-pin DIP

1.6.5. Probe P-C591

This probe is similar in features to P-C51LV and dedicated to Philips P8xC591, running at 6 clock/cycle.

The emulation header supplied with the probe is:

Emulation Header	*Description*
H-591-44P	44-pin PLCC

1.6.6. Probe P-C554

This probe is similar in features to P-C51LV and dedicated to Philips P8xC554, running at 6 or 12 clock/cycle and with QFP-80 package.

The emulation header supplied with the probe is:

Emulation Header	*Description*
H-554-80Q	80-pin QFP

1.6.7. Probe P-C323

This probe supports emulation of the Dallas 80C323 microcontroller at 3.3V, using the standard 80C323 device (ROMless mode). The maximum frequency is only limited by the 80C323 microcontroller on the probe. Clock is jumper selectable from the built-in oscillator or any external source.

The emulation header supplied with this probe is:

Emulation Header	*Description*
H-51-40D	40-pin DIP

1.6.8. Probes P-C168 and P-C434

These Probes use special bond-out devices for emulation. The maximum frequency of this probe is related to emulation mode and it is 20MHz to 25MHz, although it may change according to the innovations of the microcontrollers and derivatives. Both the probe and the emulator are prepared to run up to 42MHz. Clock is jumper selectable from the built-in oscillator or any external source.

The emulation headers supplied with the probes are:

Emulation Header	*Description*
H-168-64D	64-pin SDIP
H-434-42D	42-pin SDIP

1.6.9. Probes P-C410, P-C580 and P-C782

These personality probes incorporate a variable voltage technology that allows emulation of the microcontrollers from 1.5V to 6V and in the complete frequency range specified by Philips. Special bond-out chips are used for the emulation.

The user may select the power source either from the internal 5V power supply or by any external power source. This selection is defined by a Jumper. This

probe has another XTAL Jumper that allows selecting the clock source from the crystal on the probe or from an external source. The crystal on the probe is mounted on a socket and may also be replaced by any other one within the frequency range specified by the emulated microcontroller. In some cases a crystal will not be appropriate for low frequencies and a crystal oscillator is required. The probe allows a crystal oscillator to be replaced by another.

Probes P-C410 and P-C782 come with a 40-pin DIP emulation header. 44-PLCC and 44-QFP adapters are also available.

Probe P-C580 is supplied with a 56-pin VSO or 64-pin QFP adapter that must be soldered on your target board.

The emulation headers supplied with the probes are:

Emulation Header	*Description*
H-51-40D	40-pin DIP
H-580-56V	56-pin VSO
H-580-64Q	64-pin QFP

1.6.10. Probes P-C451, P-C552, P-C558, P-C575, P-C592, P-C598, P-C055

All these probes use standard devices operating in a special emulation mode. The maximum frequency of this probe is related to emulation mode and it is 20MHz to 25MHz, although it may change according to the innovations of the microcontrollers and derivatives. Clock is jumper selectable from the built-in oscillator or any external source.

The emulation headers supplied with the probes are:

Emulation Header	*Description*
H-552-68P	68-pin PLCC
H-51-40D	40-pin DIP
H-451-68P	68-pin PLCC
H-558-80R	Row of 80 pins, requires ADP-80Q
H-592-68P	68-pin PLCC
H-598-80R	Row of 80 pins, requires ADP-80Q
H-055-42S	42-pin SDIP

1.6.11. Probes P-C501, P-C504, P-C505C, P-C505CA, P-C505L, P-C508, P-C509Q, P-C513, P-515P, P-515Q, P-517P, P-517Q

These probes are dedicated to emulation of Siemens / Infineon SAB-C500 and 80C5xx family of microcontrollers.

All these probes use standard Siemens devices operating in a special emulation mode.

The maximum frequency of this probe is related to emulation mode and it is 20MHz to 40MHz, although it may change according to the innovations of the microcontrollers and derivatives.

Clock is jumper selectable from the built-in oscillator or any external source.

The emulation headers for the probes are:

Probe	*Header Description*
P-C501	44-pin PLCC
P-C504	44-pin QFP
P-C505C	44-pin QFP
P-C505CA	44-pin QFP
P-C505L	80-pin QFP
P-C508	64-pin QFP
P-C509Q	100-pin QFP
P-C513	44-pin PLCC
P-C515P	68-pin PLCC
P-C515Q	80-pin QFP
P-C517P	84-pin PLCC
P-C517Q	100-pin QFP

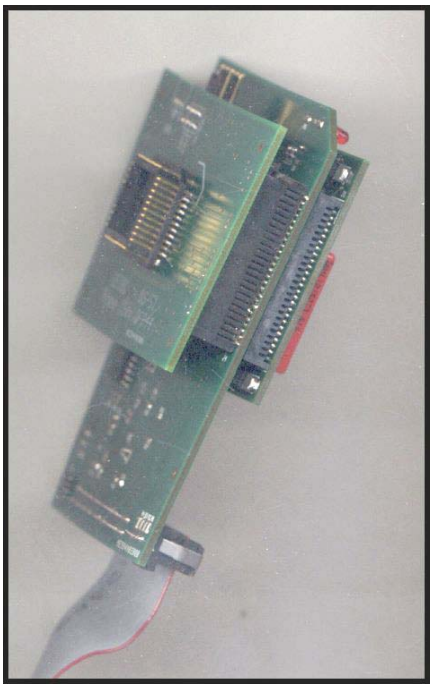

Figure 1.11: P-*C5xx* - *Emulation Header*

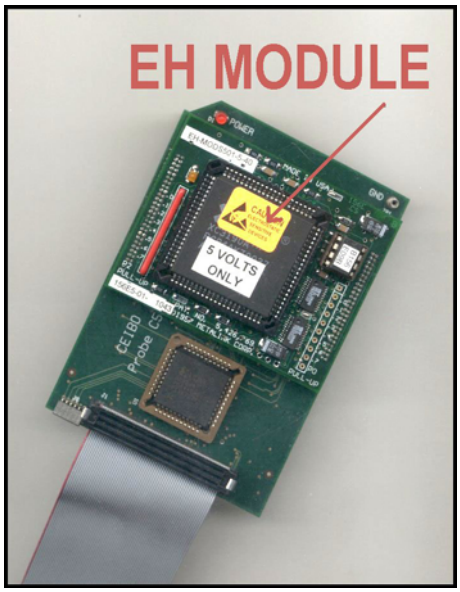

Figure 1.12: P-*C5xx - EH Module*

The bottom board carries on the emulated chip. It also has a socket for the crystal to determine the emulated frequency and a jumper to select the clock source.

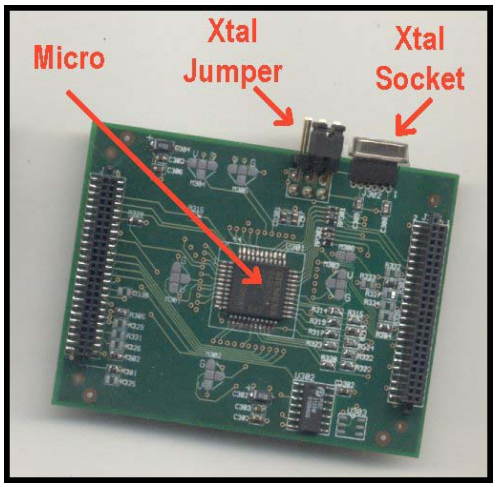

Figure 1.13: P-*C5xx - Microcontroller and Crystal*

The **XTAL** jumper is used to select the clock source.

INT means the clock is defined by the crystal on the socket..

EXT means belonging to your target.

Please note that always 2 jumpers must be set and **horizontally**.

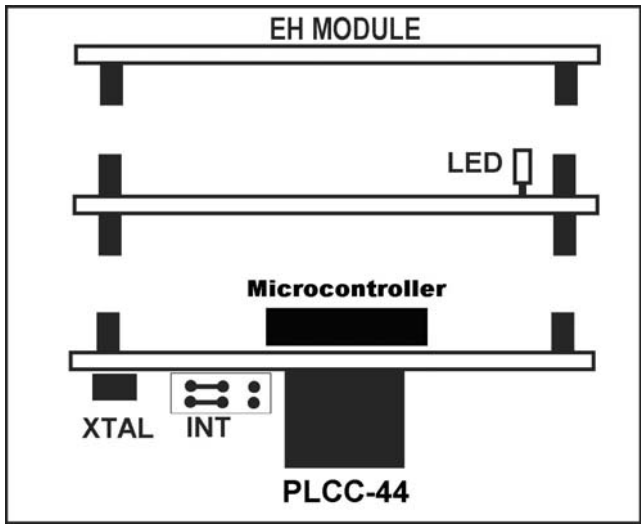

Figure 1.14: P-*C5xx - Crystal Setup*

Probe Adapter - it is the board installed ***inside the emulator box***. You need to open the emulator box to access this board. It also has to be attached to the 60-pin ribbon cable.

To connect this board to DS-51, open he emulator box. Then, align and press the adapter connectors on the J1 and J2 motherboard connectors until they are firmly plugged together.

QFP Adapter - this is a soldering block that has on the top a PLCC socket. Solder this block to your target, and connect then the emulation header.

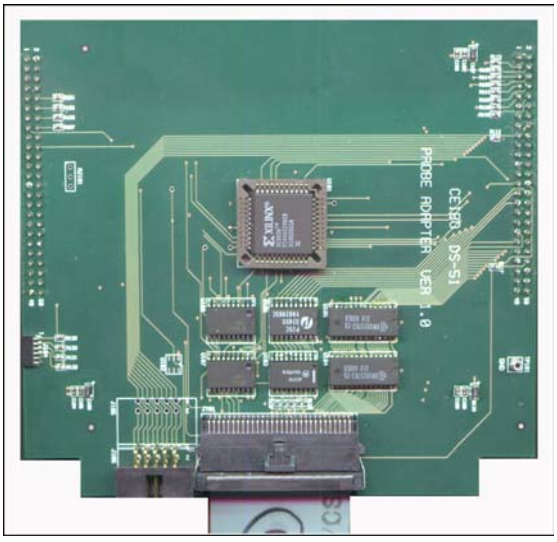

Figure 1.15: P-*C5xx - Adapter*

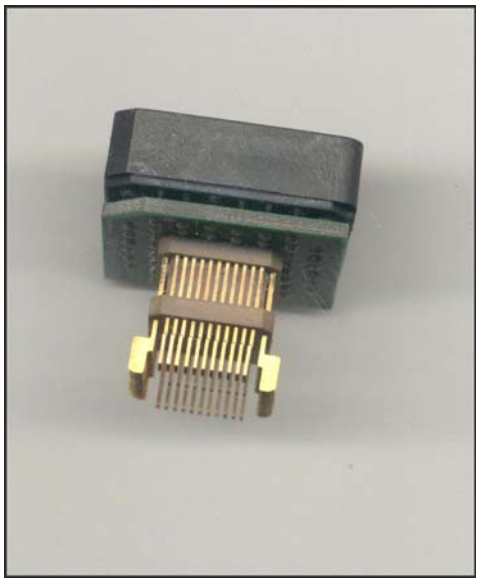

Figure 1.16: P-*C5xx - QFP Soldering Block*

1.6.12. Probe P-C530

Dallas microcontrollers are supported by probe P-C530, which is based on a special Dallas bond-out chip. The maximum frequency of the probe is 33 MHz and this is the limit of the current Dallas bond-out chip. Clock is jumper selectable from the built-in oscillator or any external source. Different derivatives are emulated by just using the appropriate emulation header: 40-pin DIP, 44-pin PLCC and 52-pin PLCC.

The emulation headers supplied with the probe are:

Emulation Header	*Description*
H-530-40D	40-pin DIP
H-530-44P	44-pin PLCC
H-530-52P	52-pin PLCC

Although Dallas 80C320 is supported by a different dedicated bond-out device, the new P-C530 bond-out chip supports it with minor timing differences, not relevant in the most common designs. However, Probe P-C320 based on the old bond-out chip is also available from Ceibo, but with support for 80C320 only. Probe P-C530 includes 64 KByte data memory that can be mapped to the emulator and accessed by MOVX instructions.

1.6.13. Probe P-C550

This probe is for Philips P8xC550 and not Dallas DS87C550. Although P8xC550 is supported by probe C51, for PLCC packages this special probe is required. The electrical characteristics are similar to those of probe C51.

The emulation header supplied with this probe is:

Emulation Header	*Description*
H-550-44P	44-pin PLCC

1.6.14. Probe P-DSC550R

This probe is for Dallas DS87C550 and not Philips P8xC550. As a bond-out is not available, the chip will work in ROMless mode only, where Port 0 and 2 are bus and not I/O (this is not important if the user code has MOVX instructions to access off chip memory or peripherals). More information about this can be found in technologies.pdf available in www.ceibo.com.

The emulation header supplied with this probe is:

Emulation Header	*Description*
H-DS550-68P	68-pin PLCC

1.6.15. Probe P-C752

This personality probe supports the 8xC750/1/2 and 8xC748/9 microcontrollers by using a standard 87C752 with the security bits programmed. This probe has a XTAL Jumper that allows selecting the clock source from the crystal on the probe or from an external source.

The crystal on the probe is mounted on a socket and may also be replaced by any other one within the frequency range specified by the emulated microcontroller. While selecting the chip type 8xC749 or 8xC752 from the software menu, the AVcc and AVss signals are rerouted to the target board and should be connected to the appropriate voltages in order to use Port 1.

Probe P-C752 uses a standard 87C752 operating in emulation mode to emulate 8xC748 and 8xC750/1 microcontrollers. The main software difference between both types of chips is found in the Interrupt Enable Register (IE - address A8h). The ETI bit of the IE Register has different addresses. The 87C752 locates the ETI bit at bit address ADh, while 8xC748 and 8xC750/1 chips have that bit at address ABh.

Therefore, when emulating an 8xC748 or 8xC750/1 with 87C752, the assembler or high-level language compiler must be set up so that the target chip is an 8xC752 to generate the file for the emulator. Once software debugging is complete, the target chip must be redefined and recompiled to to generate the file for the 8xC748 or 8xC750/1 programmer.

The above manipulations are only necessary if the software is using the ETI register bit for an 8xC750/1. The 8xC748 is equivalent to the 8xC750 and the 8xC749 is equivalent to the 8xC752. Frequency is limited by the 87C752 operating in emulation mode.

Probe P-C752 comes with two emulation headers, 28-pin DIP and 24-pin DIP. Adapters for 28-pin PLCC are also available.

Emulation Header	*Description*
H-752-28D	28-pin DIP
ADP-24D	24-pin DIP

1.6.16. Probe P-W77

This probe supports W77xxx Turbo microcontrollers (4 clocks per cycle). DS-51 maximum frequency for this configuration is up to 35-40 MHz, depending on the normal deviations of the chip characteristics.

Two modes are implemented: Romed and Romless.

a. Romed mode selection can be done via Ceibo Windows debugger. If the selected chip in the menu has any ROM/Flash, then the emulator configures the probe in Romed mode. In this mode P0 and P2 are ports only. Up to 32 kBytes of On-Chip code can to be used (0000-7FFFh) in this mode.

   Two modes are available for this setup: Normal and Advanced.

   In Normal mode up to 32 kBytes of On-Chip code can to be used (0000-7FFFh). 64 kBytes of internal/external xData available and can be accessed by MOVX @DPRT and MOVX @Ri instructions.

   In Advanced mode up to 64kBytes of On-Chip code can to be used. XData access is via MOVX @Ri only.

b. Romless mode selection can to be done via Ceibo Windows debugger. If the selected chip does not contain any ROM/Flash, the emulator configures the probe in Romless mode. In this mode P0 and P2 are address/data bus only.

   MOVX @DPRT and MOXV @Ri can be used to access 64K Xdata. Also 64K/512K code can be accessed in this mode.

The standard package is PLCC-44. Ceibo provides also adapters for DIP-40 and QFP-44. Maximum frequency is supported. 3V and 5V emulation is also possible as the probe has voltage translators.

Clk/Cycle	Fmax	ROMed				ROMless	
		NORMAL		ADVANCED			
		CODE	XDATA	CODE	XDATA	CODE	XDATA
4	35-40	32K	64K any MOVX	64K	MOVX @Ri only	64k/512K	64K any MOVX

1.6.17. Probe P-W78

This probe supports W78xxx microcontrollers working in x1 mode (12 clocks per cycle, as the standard 8051 core). DS-51 maximum frequency for this configuration is up to 40-50 MHz, depending on the normal deviations of the chip characteristics.

Two modes are implemented: Romed and Romless.

a. Romed mode selection can be done via Ceibo Windows debugger. If the selected chip in the menu has any ROM/Flash, then the emulator configures the probe in Romed mode. In this mode P0 and P2 are ports only.

   Only external XData access via short MOVX @Ri and up to 32kBytes of On-Chip code can to be used (0000-7FFFh) in this mode.

b. Romless mode selection can to be done via Ceibo Windows debugger. If the selected chip does not contain any ROM/Flash, the emulator configures the probe in Romless mode. In this mode P0 and P2 are address/data bus only.

   MOVX @DPRT and MOXV @Ri can be used to access 64K Xdata. Also 64K/512K code can be accessed in this mode.

The standard package is PLCC-44. Ceibo provides also adapters for DIP-40 and QFP-44. Maximum frequency is supported. 3V and 5V emulation is also possible as probe has voltage translators.

Clk/Cycle	Fmax	ROMed				ROMless	
		NORMAL		ADVANCED			
		CODE	XDATA	CODE	XDATA	CODE	XDATA
12	40-50	32K	MOVX @Ri only	32K	MOVX @Ri only	64k/512K	64K any MOVX

1.7. Trace Filters and Triggers

The debugger provides many functions to select the desired data to be recorded in the trace buffer and also when to start and stop recording. ***Chapter 5*** provides more information about the possibilities and software menues. The main software and hardware capabilities are below described.

Trace Filters: Defines the trace filters of the displayed data. You may specify which instructions or sequences are of your interest. Selected regions are IN or OUT, where IN means enable and OUT means disable recording of marked lines in the window. The IN Select button toggles the window from all IN to OUT and vice versa. The last line of the window is All Range; this line is used to select or cancel all the memory to include or omit libraries, assembler modules and other code not defined in the debug information of your compiler. You may also add or delete user defined regions.

TP Recording: Enables or disables the recording of testpoints.

Trace Triggers: This command sets the trace control to the selected option to start and stop the recording. The different buttons and functions are:

Run begins: When execution is started, the trace buffer will capture data from the start program execution until the trace buffer is filled. This mode automatically selects the *Stop when Full* option.

Halt ends: Trace buffer capture is enabled and data is continuously collected until the break condition occurs. In this mode, the trace buffer will be filled with bus cycles immediately preceding the break condition.

From event: Trace capture begins when the Start event trigger condition is satisfied and continues until the program stops due to a breakpoint or Halt command. In this mode, the trigger event will be found in the beginning of the trace buffer contents. After clicking this button you must select the Start event button to complete the trigger definition.

Until event: Trace capture is enabled and the trace buffer filled until the Stop event trigger condition is satisfied. In this mode, the trigger event will be found in the end of the trace buffer. After clicking this button you must select the Stop event button to complete the trigger definition.

Dual event: This option combines both From and Until events, therefore you have to complete the selection by defining Start and Stop events.
Start event: Select the Start event button to define the trigger conditions for trace modes using a start event. This button will be dimmed if the selected trace mode does not require a start event.

Stop event: Select the Stop event button to define the trigger conditions for trace modes using a stop event. This button will be dimmed if the selected trace mode does not require a stop event.

Hardware Trace Triggers: These options are used to start and stop trace recording according to the testpoint signals connected to the clips. A selection of logic levels and edges is available. **See Trace Clips according to the description given in paragraph 1.9.**

After selecting the Start or Stop event buttons, the Set Trace Trigger dialog box will be displayed.

A Trace Trigger selects the criteria for capturing an execution trace in the target system trace buffer. Setting a trace event and a hardware breakpoint are in fact the same operation. The only difference between the two operations is that action for a testpoint is to turn the trace buffer on or off while breakpoints are used to implement the various breakpoint actions.

The Trace Trigger definition is used to define the condition used to start trace data collection. When this option is selected, a dialog box containing field for selecting the bus cycle type, address and data bus contents is displayed and you are prompted to fill in the conditions used to trigger the trace collection hardware in the target system.

When a trace trigger is defined, each field of the dialog box is combined to specify the event. Trace triggers operate on the recognition of user-defined events, using combinations of the following controls.

Cycle: Use this option to specify the bus cycle or execution state to be used in defining the trace trigger condition.

Address Match: Selects the address qualification mode. If any mode other than Match All is selected, the Address input box must be filled in with the address or address range to be recognized by the trace trigger. Addresses can be entered as symbols, modules and line numbers or as physical addresses.

Data Match: Selects the data qualification mode. If any mode other than Match All is selected, the Data input box must be filled in with the data or data range to be recognized by the trace trigger. Data can be entered as symbols.

Passcount: Selects the number of times the event will be recognized before the trace trigger occurs. When all the condition have been specified, select OK to confirm the trace trigger setup.

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.

Trace Filter Operating Instructions

Open the Trace Filter window by invoking the View|Trace|Trace filter menu.

The window will display the current address range setting of the Trace filter, address range marked as IN will be recorded in the trace when the trace is running while address range marked as OUT will not be recorded. The default setting of the Trace filter is 0000-FFFF IN, that is the entire 64K address range will be recorded in the Trace buffer. (Note that the Trace filter operates in Real-Time while your program is running).

You may enter up to 10 different address ranges. The display will be automatically updated and show the filtered ranges in a memory map format from address 0000 till FFFF.

Use the local menu (Alt-F10) in order to Add new ranges, Accept changes made, Cancel changes made, Restore filter or Clear filter. You must exit this window before resuming normat operation. (You may use the INS, Enter, Esc keys instead of the local menu).

The format for entering a new range is as follows:

StartAddress, StopAddress, IN/OUT

i.e.

123,234,IN

345,456,OUT

You may enter symbols or line numbers reference in the standard symbol format.

i.e.

.delay,:tdelay#7,OUT

Normaly you would wish to mask out from the test commonly called routines (such as delay routines etc.) for this you should first Clear the Trace filter (i.e. set 0000,FFFF,IN) and then enter the ranges to be masked as OUT.

If on the other hand you wish only to record special routines of your code in the Trace buffer (i.e Interrupt routines) you should first set the Trace filter to 0000,FFFF,OUT and then enter the ranges to be recorded as IN (i.e. 200,220,IN).

More About Trace Filters - selecting address ranges

The ranges that appear in the configuration windows reflect the modules that your assembler or compiler generates.

The ranges can be enabled or disabled just by clicking on with the left button or space bar.

If you check the IN button, the highlighted lines are recorded in the trace. The "selected regions are IN" legend appears on the top of the window.

If the IN button is not checked, marking the lines works in the opposite way, meaning that all the highlighted lines will NOT be recorded in the trace. The "selected regions are OUT" legend appears on the top of the window.

You can also define any routine, module, line numbers and ranges to be recorded or not in the trace regardless if they are a module or not. Use the ADD button to define your customized range.

Delete button in Trace Filter Configuration Window is only for user defined areas.

The debugger shows in that window all the available ranges as contained in the file you are loading. You can enable or disable them, but not erase them with the Delete button.

Following is a sequence that will help you to use the trace filter capabilities:

1. Load your code
2. Open the Trace Dump window.
3. Open the local menu by clicking the left button.
4. Select Hardware Trace Filters
5. Click the left button of the mouse while pointing to the line: #EVAL2#MAIN [3h,24h] - this is just an example.

   After doing that the line #EVAL2#MAIN [3h,24h] is disabled (not erased).
6. Click the Add button.
7. Fill Alias with "all code"
8. Fill Start with 0 (this is the default).
9. Fill Stop with 0xffff
10. Click OK.
11. Click Now all the code is selected and the trace will record all.
12. Click the left button of the mouse while pointing to the line: "all code" Now all the code is disabled and the trace will not record anything.
13. Click the Add button.
14. Fill Alias with "my area"
15. Fill Start with 0 (this is the default).
16. Fill Stop with 100
17. Click OK.
18. Click Now all the code is selected and the trace will record only when the program counter points to 0-100.

1.8. Hardware Breakpoints

Emulation can be stopped upon occurrence of external events, such as logic level transitions in your hardware.
Breakpoint on External Events

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.

1.9. 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

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.

1.10. Banking Support - DS51/512K Emulators Only

Address lines are available through the trace clips to support the different bank configurations. Addresses are provided by the emulator while loading a program and by the target circuit while executing it.

Connect the trace clips to address lines in your target circuit before loading and executing your program. The trace clips have the following meaning:

Banking Type: 32K

EA15 = TP8 (GRAY)

EA16 = TP7 (VIOLET)

EA17 = TP6 (BLUE)

EA18 = TP5 (GREEN)

Banking Type: 64K

EA16 = TP8 (GRAY)

EA17 = TP7 (VIOLET)

EA18 = TP6 (BLUE)

EA15 to EA18 are the addresses produced by your decoder in your target. These lines are connected in your target to EPROM/Flash or other memory used as the code memory.

Your hardware must work properly to generate EA15 to EA18. Please check it. It is recommended to use a small testing program and verify that all works properly.

Set also the banking configuration (Options | Debug Controls) prior to loading your code.

Banking Type	TP8 (GRAY)	TP7 (PURPLE OR VIOLET)	TP6 (BLUE)	TP5 (GREEN)
128K (2x64K)	A16	-	-	-
256K (4x64K)	A16	A17	-	-
512K (8x64K)	A16	A17	A18	-
128K (4x32K)	A15	A16	A17	-
256K (8x32K)	A15	A16	A17	-
512K (16x32K)	A15	A16	A17	A18

TABLE 1.1: ***Banking Support***

Keil Software is supported directly. Please do the following changes in your project in order to produce absolute file with a banked code:

1. Select Options | C51 Compiler.. menu item

   1.0 C51 Compiler - Object category.
   "Keep variables in order", "Include debug information",
   "Include extended debug information" boxes should be checked.

2. Select Options | A51 assembler.. menu item

   2.1 A51 assembler - Object category.
   "Include debug information" box should be checked.
   "Exclude line numbers" should be unchecked.

3. Select Options | BL51 Code banking Linker.

   3.1 Linking category : "Code Banking" should be checked.

   3.2 Size/Location category: "Bank Area" field should contain:

   0,0FFFFh (64K banking)

   OR

8000h,0FFFFh (32K banking)

4. Select Project | Edit Project menu item

4.1 Select the banked Module in "Source Files" list and specify a "Bank Number" for this module. Bank numbering is **zero-based for 64K banking** and **one-based for 32K banking**.

IAR compiler needs to select in the Load Dialog, List Files of Type, IAR BANKED. Use -r of the DOS C-compiler option in order to add debug information or for the Windows version check also the following boxes:

Project | Options | ICC8051 | Debug | Generate Debug Info

Project | Options | Xlink | Output | Debug Info

The simulation mode is not for bank switching. Use the emulator in ***emulation mode only***.

Only Hardware Breakpoint are supported. Do not check Software Breakpoints in (Options | Debug Controls).

Ceibo Debugger V1.17 or later must be used for bank switching support.

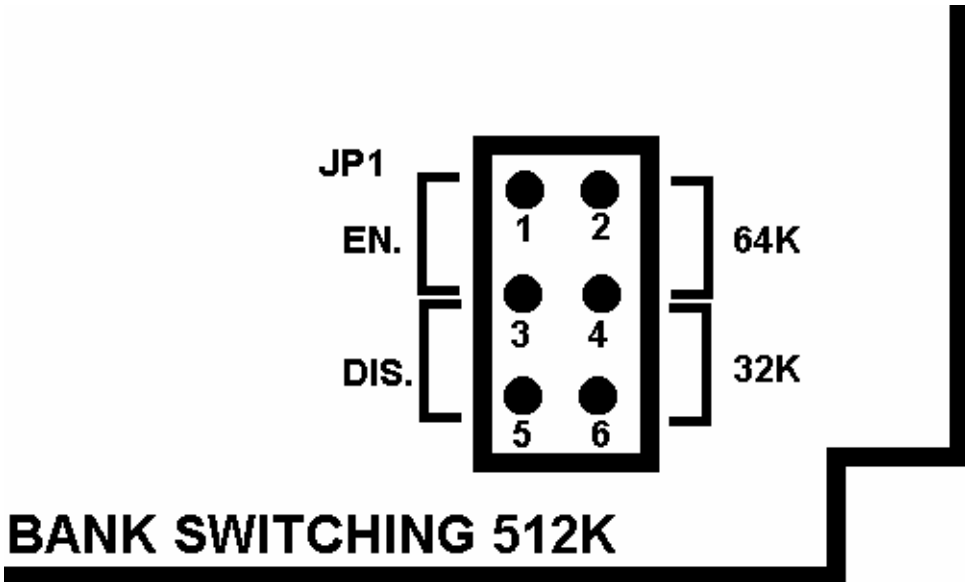

FIGURE 1.17: ***Banking Support - Jumper Setup***

DS-51 motherboard (inside the emulator box) has a jumper to select the following options:

SETUP	EN.	DIS.	64K	32K
NO BANKING		ON	ON	
64K BANKING		ON		
32K BANKING	ON			ON

TABLE 1.2: ***Banking Setup***

ON: means place the cap.

64K BANKING: only one cap must be placed; the other one (32K/64K) must be removed.

NO BANKING and **32K BANKING**: need two caps.

1.11. Watchdog Support

You may enable or disable the watchdog feature available in many microcontrollers.

If you use the Windows software, select Debug Controls in the Options Menu and enable/disable the watchdog through the watchdog setting command.

If you are using the DOS Debugger, invoke the software with the command line: C51D -ENAWD.

1.12. Idle Mode Support with Probe C51LV

Probe C51LV supports Idle Mode with Ceibo Windows Debugger V1.18 or later.

Stopping the emulation with Halt Command (Ctrl-Break)

When the user halts the emulator (Ctrl-Break) and the target micrococontroller is in Idle state,the Ceibo Windows Debugger will wait about 10ms to allow the exiting this mode by means of a user interrupt. Then, emulation will stop.

If at that time the microcontroller still remains in Idle Mode, emulation will be stopped by a Reset state. The debugger will display in this case:

Power down or idle mode detected.
Emulation halted by reset.
Check trace to see sequence of instructions.

Then the user can read the trace to see the sequence of instructions that lead to Idle Mode.

Stopping the emulation with External Breakpoints

An additional way to the stop emulation while the microcontroller is in Idle Mode is by using the DS-51 External Breakpoint and trace clips.

1. In the Ceibo Windows debugger menu select :

   Breakpoints-> External Breakpoint -> BP trigger -> Rising edge

   Breakpoints-> External Breakpoint -> Logic Operator -> Or

2. If you want to stop the emulation when the microcontroller exists the Idle mode, connect GRAY or VIOLET cable to T2 (testpoint on the Probe C51LV).

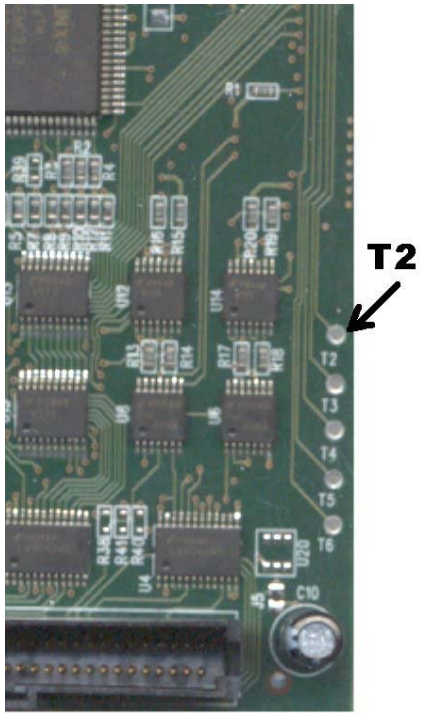

FIGURE 1.18: ***Exiting Idle Mode with External Breakpoint***

1.13. ALE Noise

Some conditions may cause noise or ringing on ALE signal: using external clock, poor grounging in the target board, etc.

This may be detected by filling a block of XDATA RAM (from the Memory Space, XDATA window) with different patterns (0,55,AA,FF). If the block is only partially written correctly and shows holes in various locations changing each time the block is read then it is clearly an ALE ringing/noise problem.

This can be solved either by using a CMOS technology latch (373/573) on your target board (since CMOS has higher noise immunity) like HC family (and not HCT), or by placing a small (100pF) capacitor on the ALE line directly on your target board latch.

It is also recommended to use clock internal.

1.14. 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.

1.15. ADP-44P - 44 PIN PLCC to 40 PIN DIP Adapter

ADP-44P converts 44-pin PLCC to 40-pin DIP sockets, thus allowing the use of popular 40-pin DIP devices with systems designed for 44-pin PLCC devices. This is a common application during the debugging cycle while using in-circuit emulators.

ADP-44P is compatible with most of the popular 80C51 family of microcontrollers. Among the supported devices are: 8xC31/2, 8xC51/2, 8xC51FA/B/C, 8xC524, 8xC528, 8xC652, 8xC654 and others.

Philips P8xC550 is not compatible with ADP-44P.

CHAPTER 2

Installation

CHAPTER 2

Installation

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-51 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-51 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-51)	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. 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:

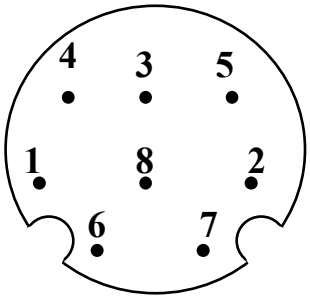

FIGURE 2.2: ***Power Connector***

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.

2.4. Connecting the DS-51 Components

No special tools are required to install the DS-51.

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-51.

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.

2.5. Installing the Software

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

Turn on the host computer.

Insert your DS-51 disk or CD into the Drive.

From the Program Manager select the drive with the Windows Debugger disk or CD and run SETUP.

Complete Setup utilities are supplied with the CEIBO Debugger. All the Debugger files are compressed and expanded during installation. The Windows Setup creates the Ceibo Windows Debugger icon.

Follow the instructions on the screen. This will automatically create and update groups and icons the next time you run Windows.

2.6. Starting Up

Turn the DS-51 power supply on.

If the emulator is connected to a target board, turn on first the emulator and then the power supply of the target.

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.

2.7. Changing The Personality Probe

Follow the steps described below to replace the personality probe:

a) Ensure that the power is OFF to DS-51.

b) Loosen the four screws in the bottom of the box. No tools are needed and this operation can be done manually.

c) To connect a different personality probe to DS-51, align and press the J1 and J2 connectors together until they are plugged into the DS-51 main board.

FIGURE 2.3: ***Opening the Emulator Box***

J1 and J2 motherboard connectors are shown in the following figure.

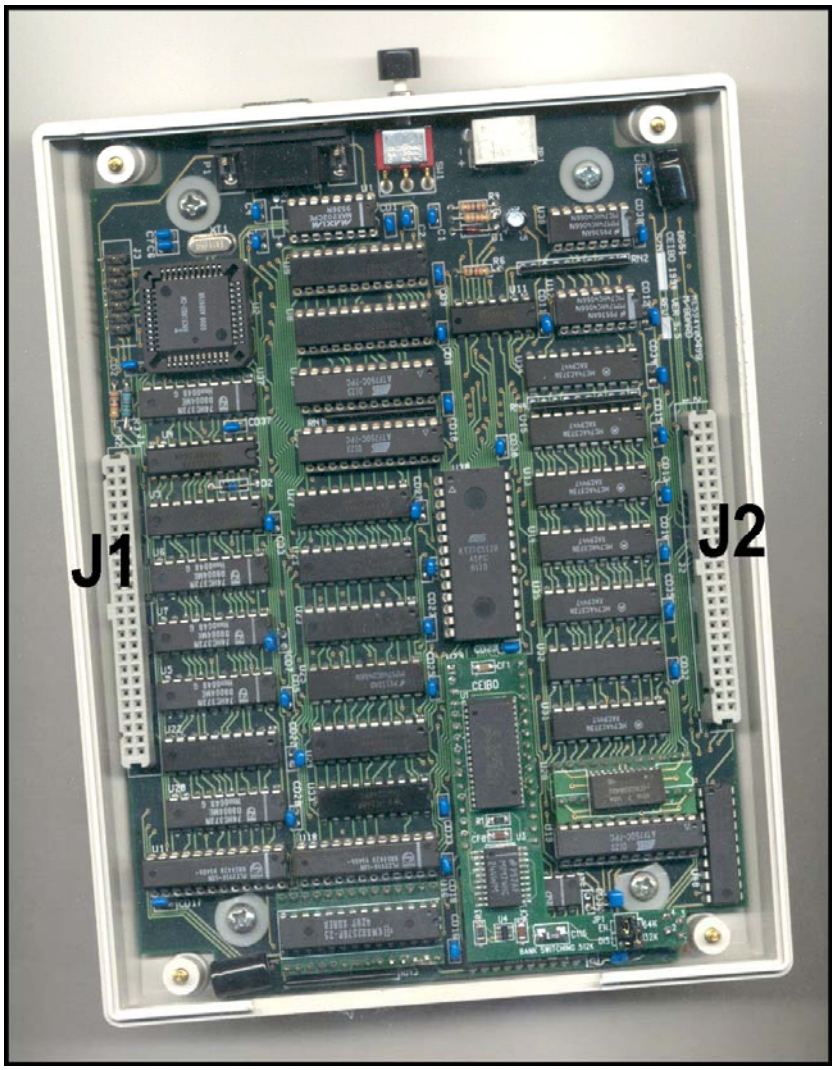

FIGURE 2.4: ***Locating the Adapter Connectors***

2.8. Changing the Header

DS-51 has different headers to emulate the different packages and pin-outs of the microcontrollers.

Follow the steps described below to replace the header:

a) Ensure that the power is OFF to DS-51.

b) Use the ejectors to remove the header from the ribbon cable.

c) To connect a different header to the DS-51, align and press the connectors firmly together until the ribbon cable is firmly plugged into DS-51.

2.9. Changing The Microcontroller

Some Personality Probes allow the user to replace the Microcontroller to emulate a different derivative.

Probe C5xx

The following figure shows the location of the Microcontroller.

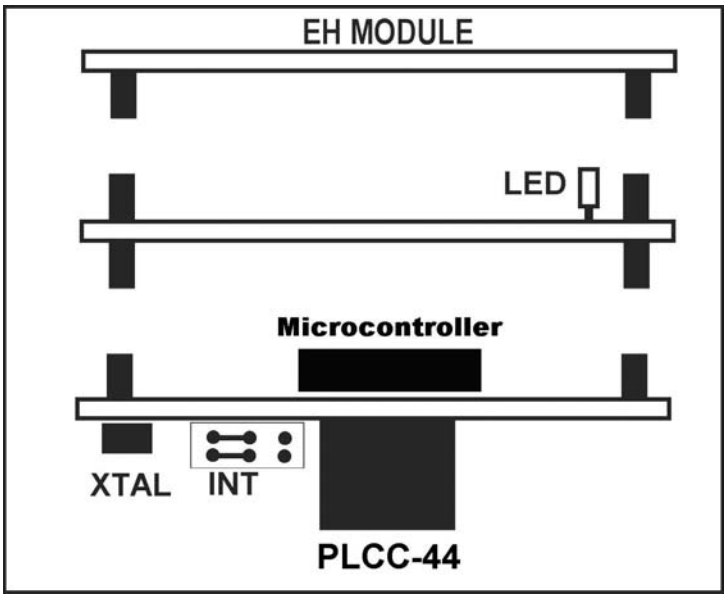

FIGURE 2.5: ***P-C5xx Microcontroller***

Probe C51LV

This is similar to the above without the EH-Module. It is a 44-PLCC Microcontroller

Probe C51

The emulated Microcontroller is on the Probe Adapter, inside the emulator box and as shown in the prefious chapter. It is a 40-DIP Microcontroller.

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


load a new program to debug


open a module window


open a CPU window


add a new expression to watches window


run program


execute a single source line or instruction


execute a single source line or instruction skipping calls

stop running the 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 or buttons.

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-51. It is not necessary for the first stage which mostly explains the software capabilities.

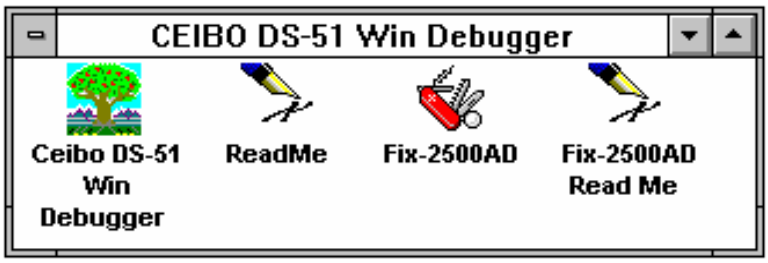

FIGURE 4.1: ***Debugger Icon***

2. Invoke the Windows Debugger by double clicking the Ceibo icon. You may get a communication error if the system is not connected.

4.3. Accessing the Global Menu

FIGURE 4.2: ***Global Menus***

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

The bottom bar displays the software status.

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

4.4. Selecting the Simulation Mode

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

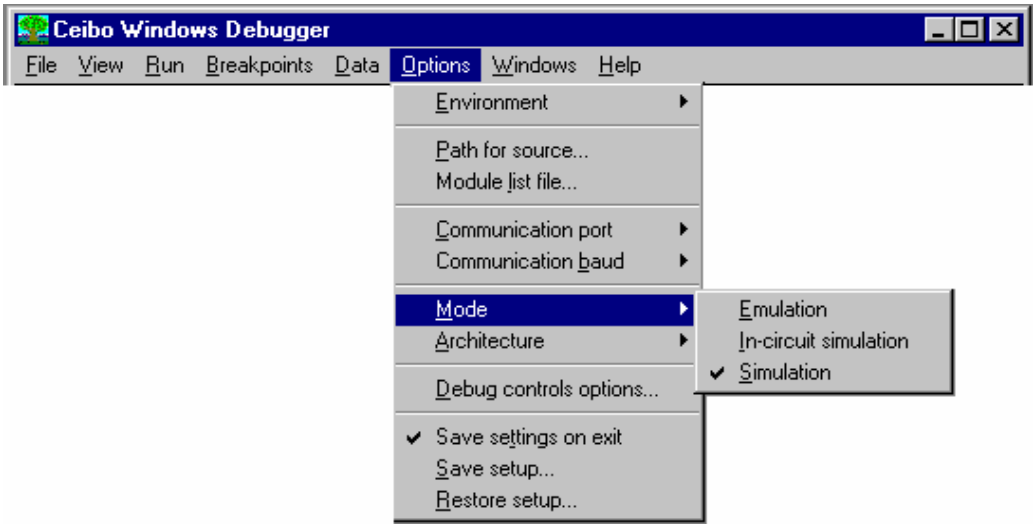

FIGURE 4.3: ***Simulation Mode***

4.5. Opening Windows

1. 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.

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

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

4.6. Accessing the Local Menu

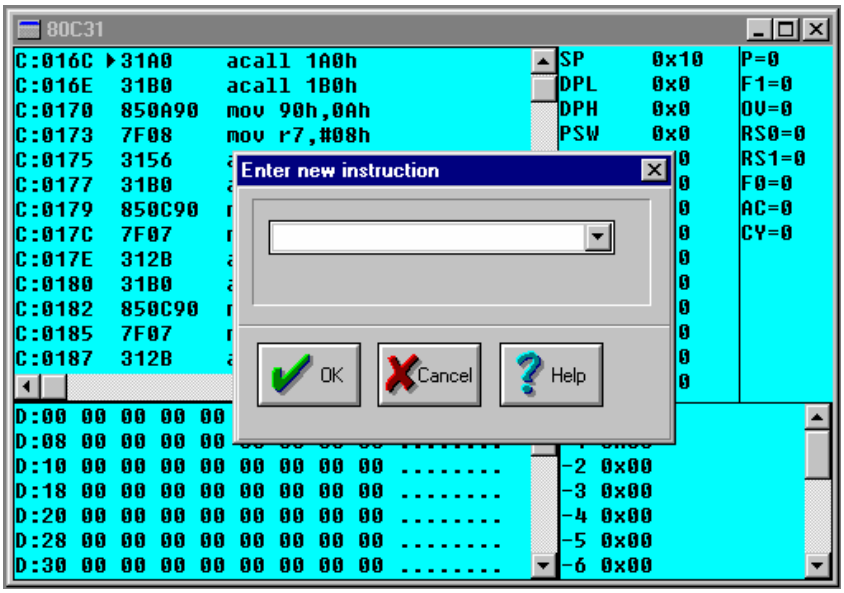

FIGURE 4.4: ***CPU Assembly Command***

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 A,3 directly. Observe how the code has been changed.

4.7. 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 or from the Data Menu.

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.

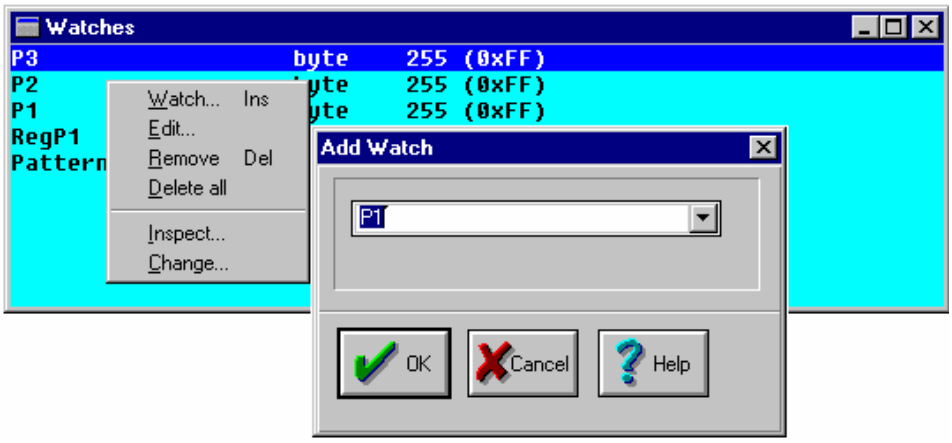

FIGURE 4.5: ***Adding Watches***

4.8. 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.

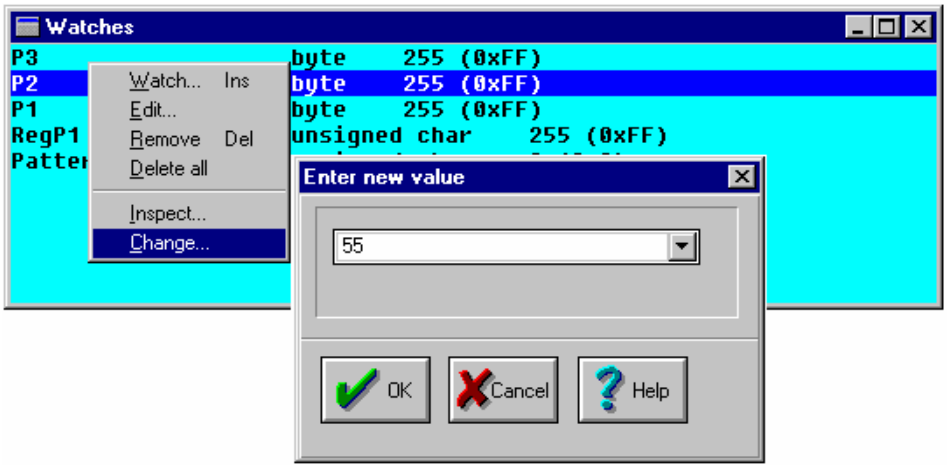

FIGURE 4.6: ***Changing Watches***

4.9. 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 on-chip 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 internal RAM. Type B:0,3.

5. Display 5 bytes of the external memory space accessed by MOVX instruction. Type X:100,5.

6. Add to the Watches Window any SFR by entering the absolute address.

   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.

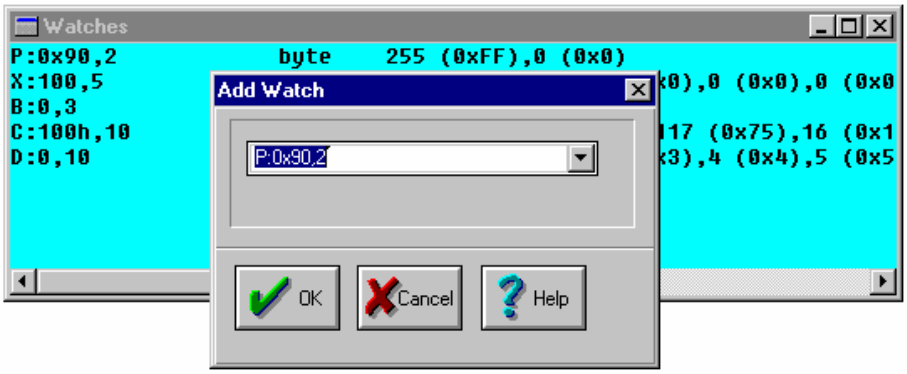

FIGURE 4.7: ***Adding Strings to the Watches***

4.10. Displaying a Memory Space

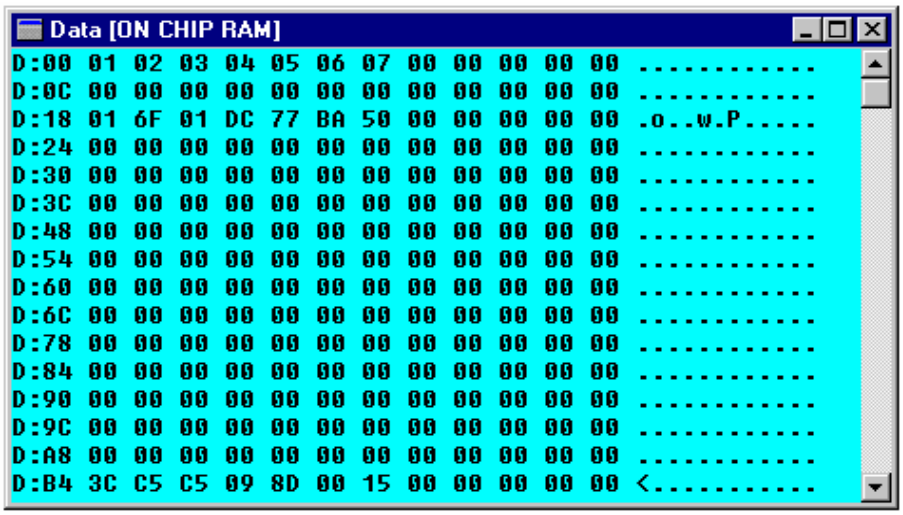

FIGURE 4.8: ***Data Window***

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.

4.11. 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.

4.12. 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.

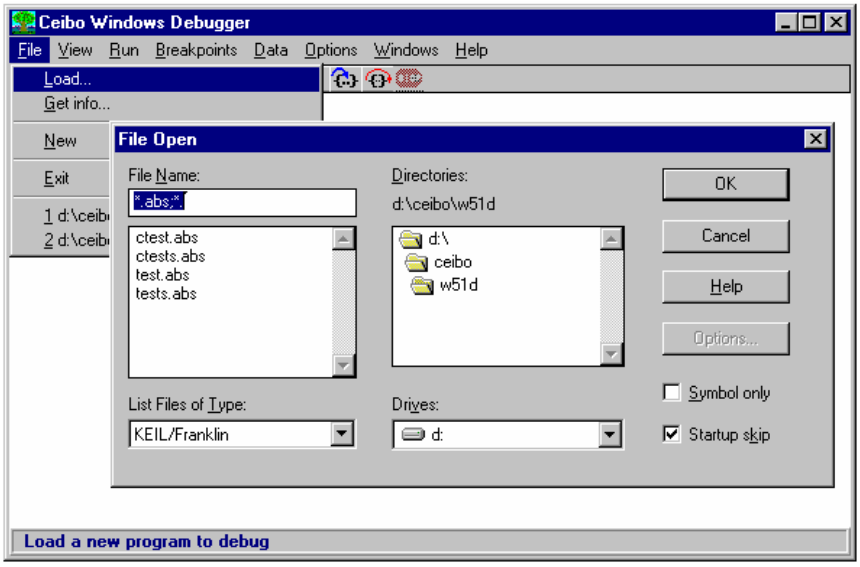

FIGURE 4.9: ***Loading a File***

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

Note that it is very important to define the type of list files to get the proper symbol information. TEST.ABS and TESTS.ABS are Intel ASM/PLM51 files. 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.

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.

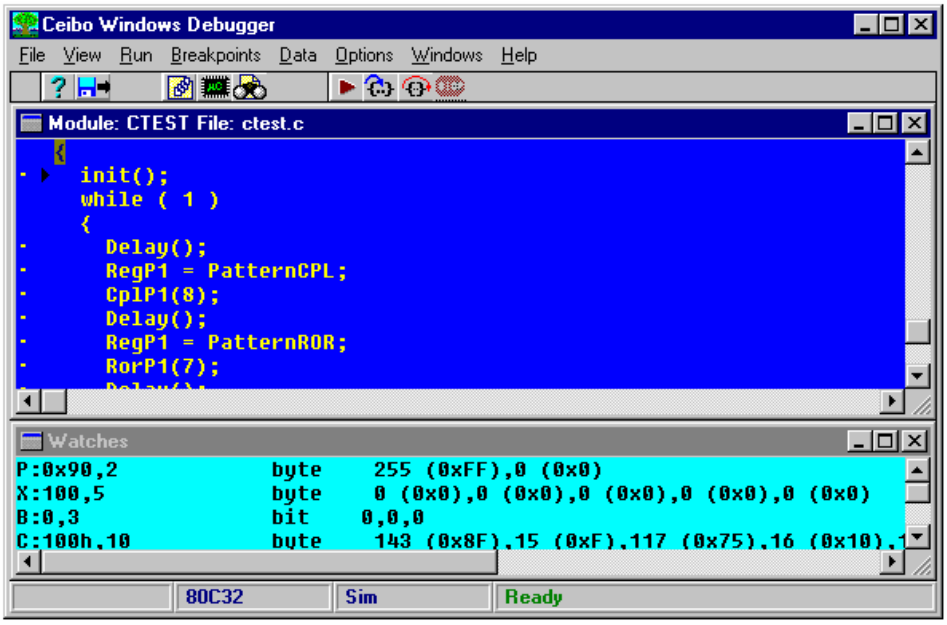

FIGURE 4.10: ***Default Screen***

4.13. 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.

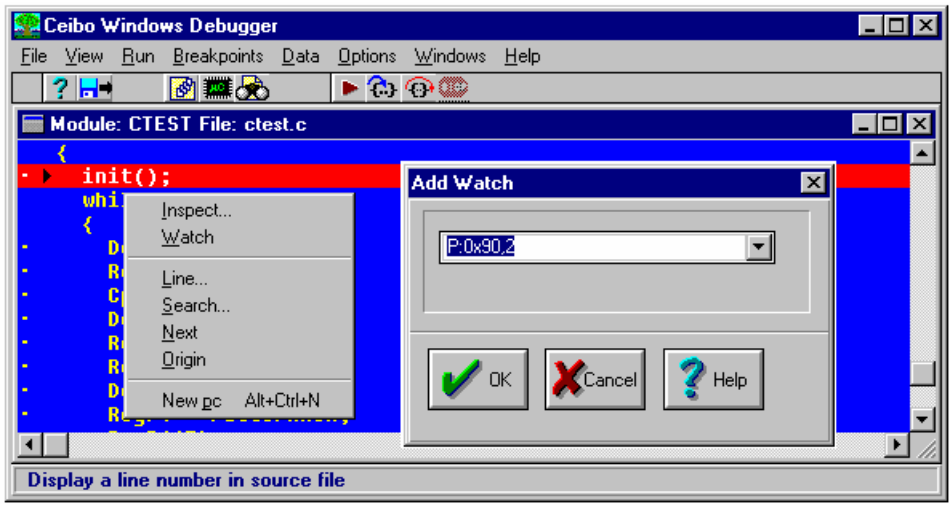

FIGURE 4.11: ***Capturing Watches***

4.14. Debugging the Program

1. Position the cursor on the RegP1 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.

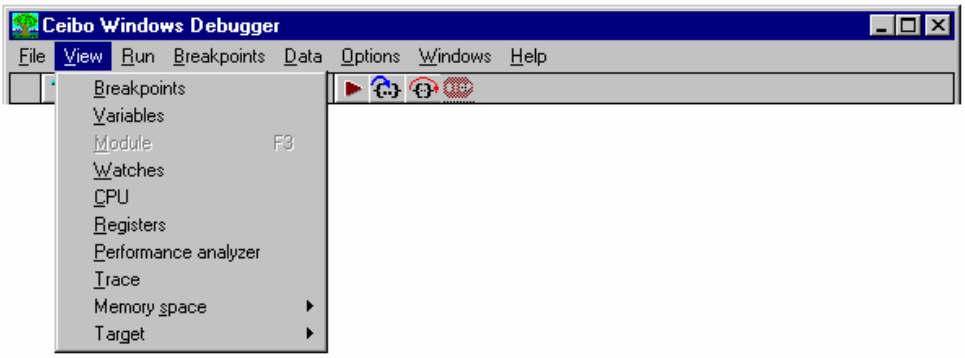

FIGURE 4.12: ***View Menu***

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 F8 key. Repeat that operation several times.

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.

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, Intel ASM51 and PLM51 OMF51,

Keil extended OMF, IAR, UBROF, BSO/Tasking OMF51 and IEEE, old BSO, 2500AD, Metalink ASM51, MCC Software and general OMF51. Call Ceibo for additional format support.

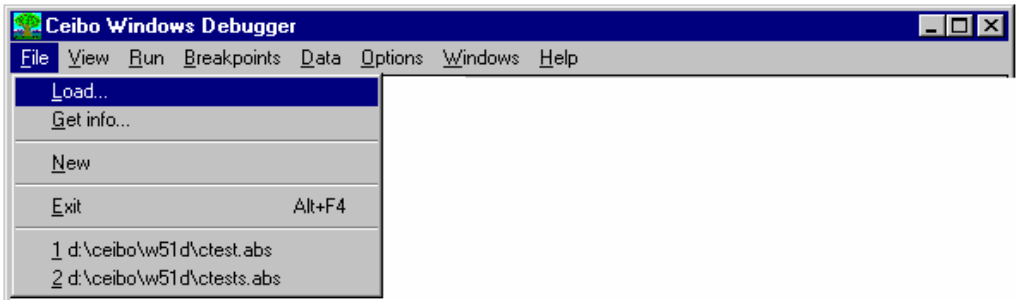

FIGURE 5.1: ***File Menu***

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.

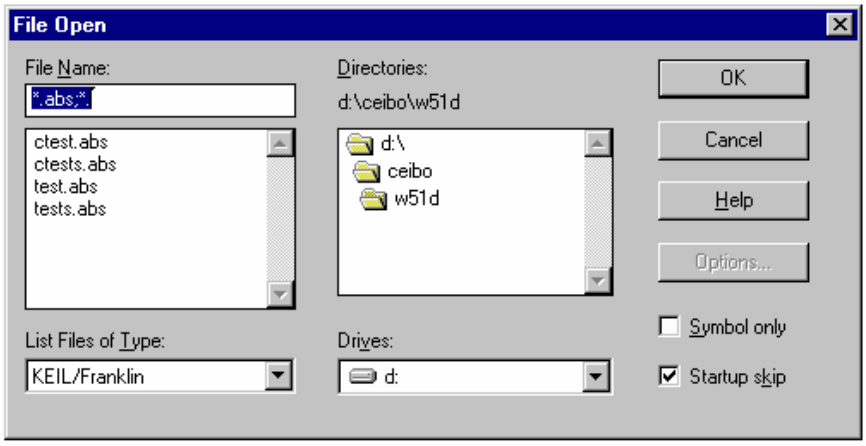

FIGURE 5.2: ***File Open Dialog***

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, Watches, CPU, Registers, Performance Analyzer, Trace, Memory Space and Target.

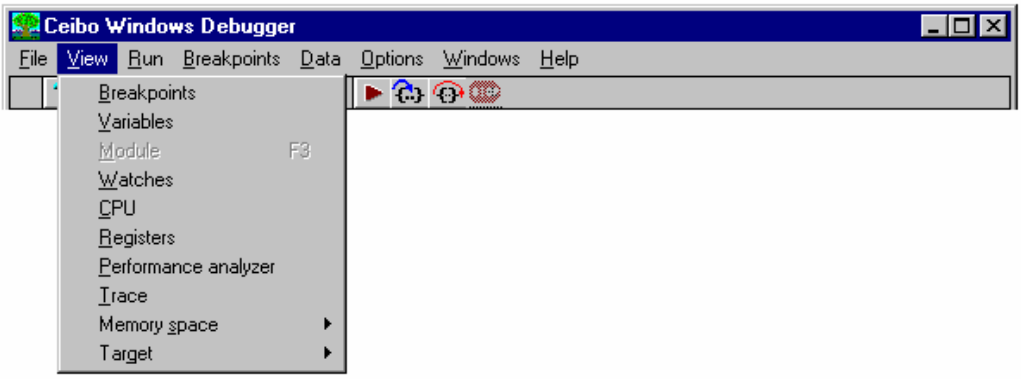

FIGURE 5.3: ***View Menu***

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:

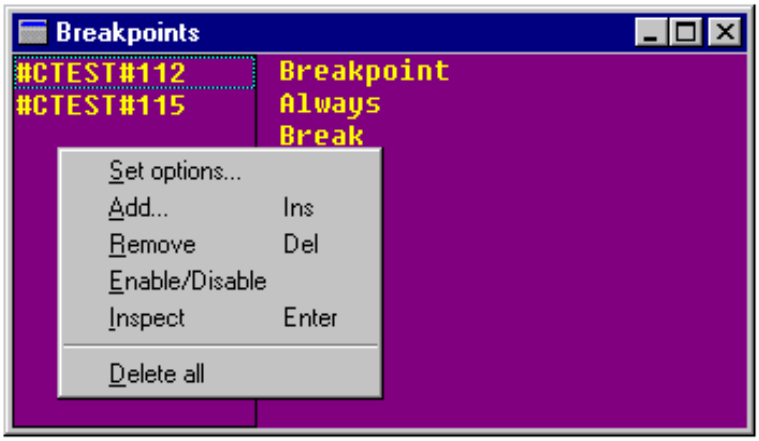

FIGURE 5.4: ***Breakpoints Local Menu***

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.

Data match: Defines the data match condition.

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.***

Data value: Sets the data value. The format is any expression of supported languages.

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.

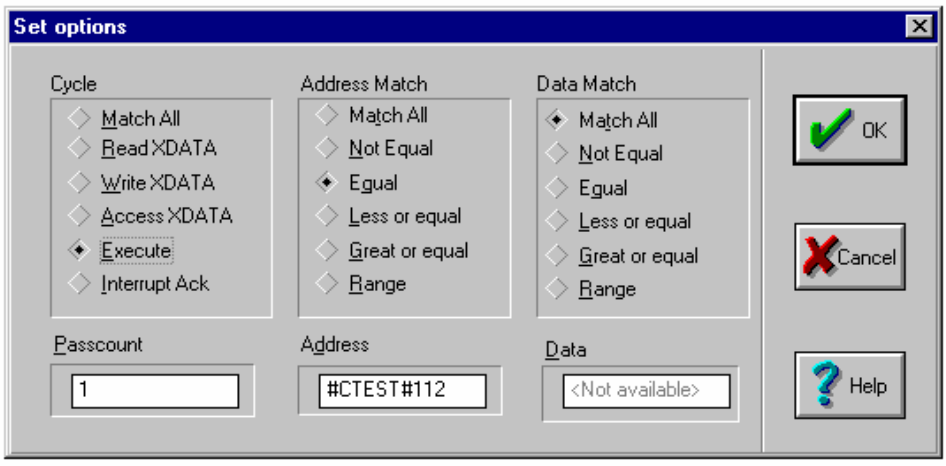

FIGURE 5.5: ***Set Options***

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. The cursor in the CPU or Module window will point to the selected value.

Delete All: Removes all the defined breakpoints.

Variables

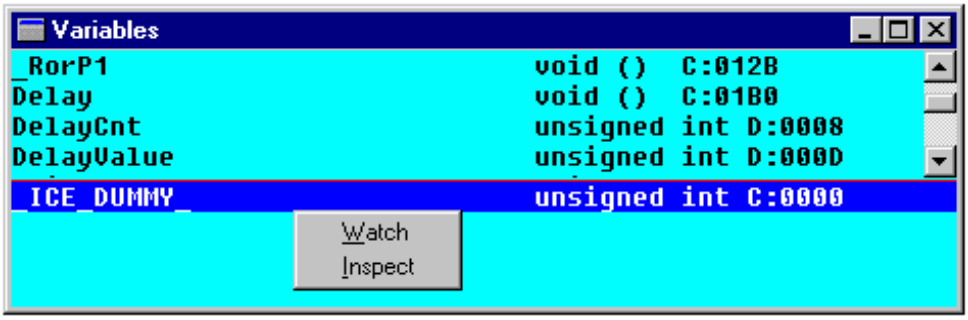

FIGURE 5.6: ***Variables Window***

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.

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.

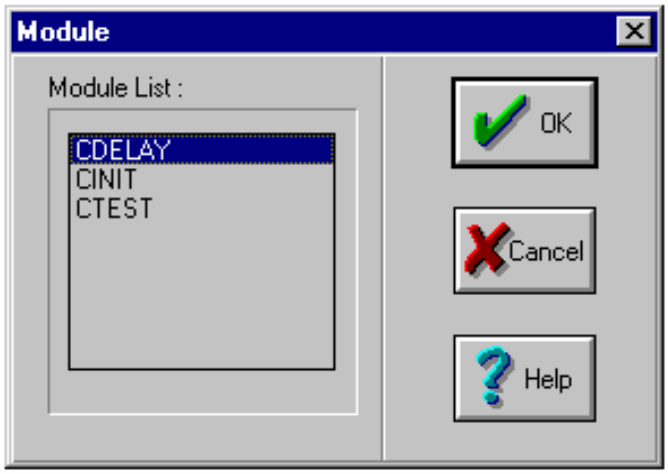

FIGURE 5.7: ***Module List***

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:

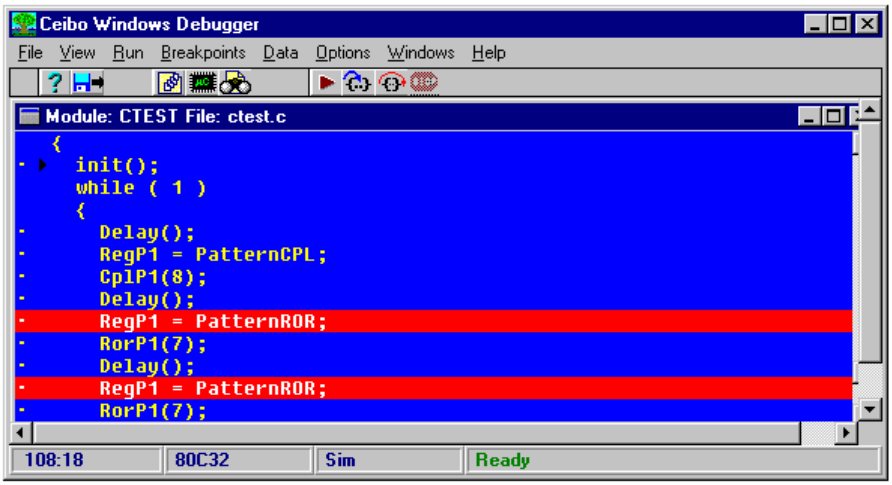

FIGURE 5.8: ***Module Local Menu***

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.

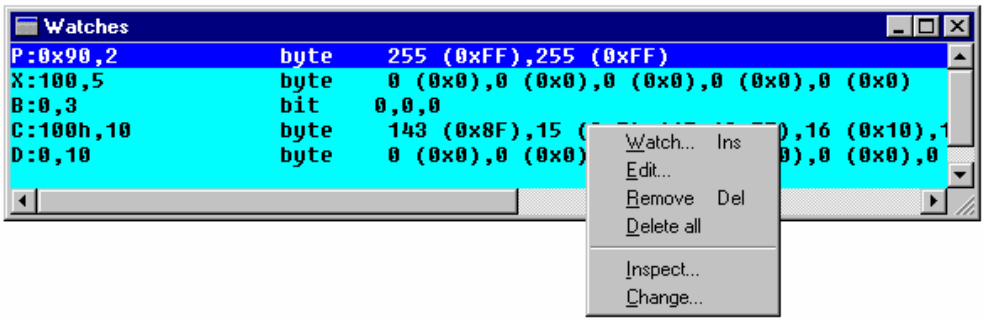

FIGURE 5.9: ***Watches Local Menu***

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, ACC, 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 for external RAM accessed by MOVX instructions, P for ports and any SFR (special function registers), C for code memory and B for bit memory (below 80H is for RAM bits and above that value represents SFR bits). Some examples are:

D:100,5

X:MYSYMBOL,2

B:0x80,5

B:P1.0,8

P:0xFE

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.

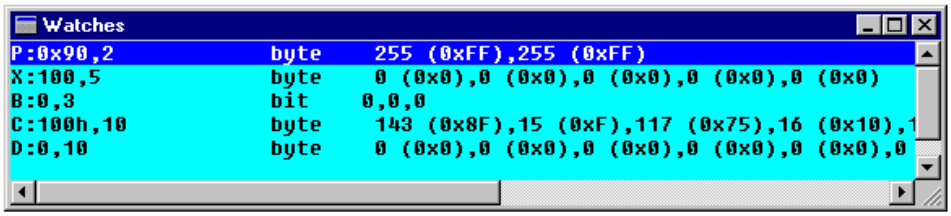

FIGURE 5.10: ***Watches Window***

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 or PLM. 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.

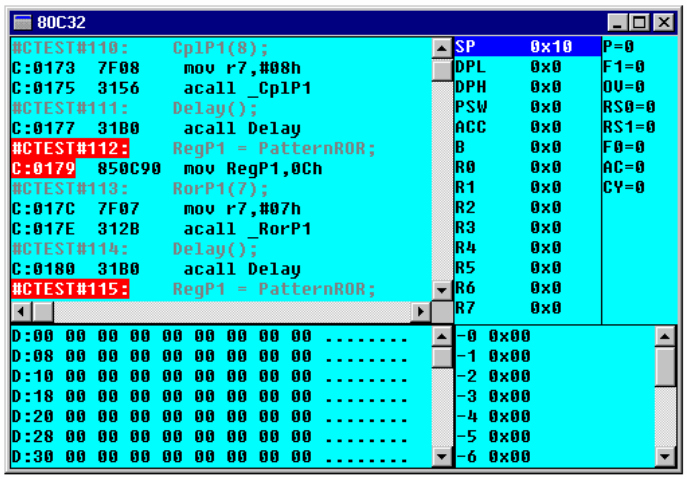

FIGURE 5.11: ***CPU Window***

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.

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.

Maximize: Increases the size of the window. This function is used for example to display the disassembled instructions without the other associated windows (stack, registers, memory).

Restore: Sets the screen to the default size.

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.

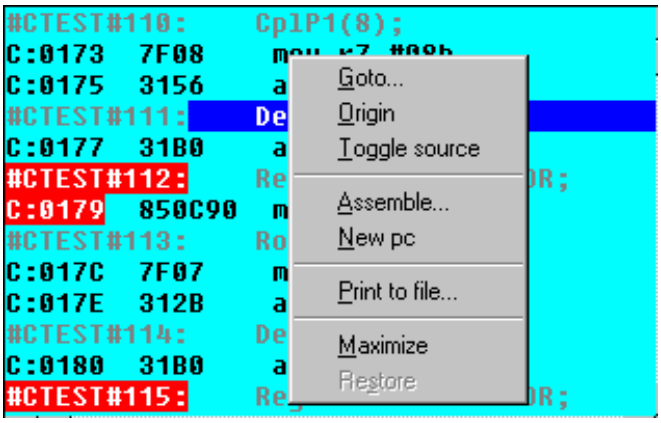

FIGURE 5.12: ***Disassembly Local Menu***

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.

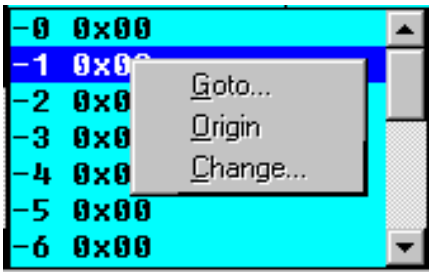

FIGURE 5.13: ***Stack Local Menu***

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.

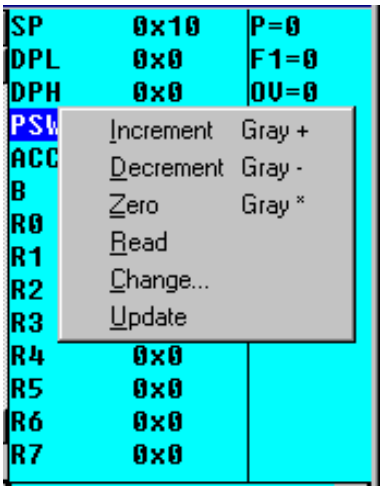

FIGURE 5.14: ***Registers Window***

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.

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.

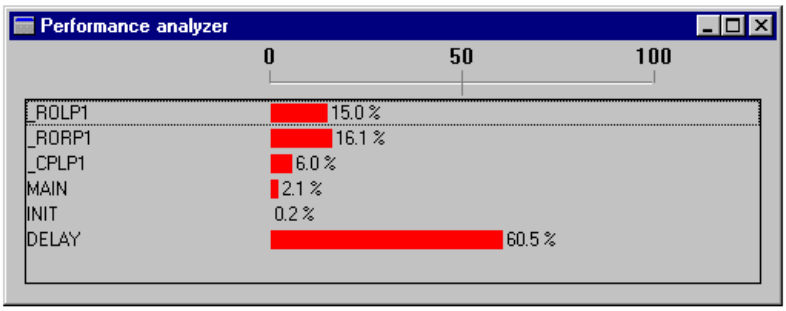

FIGURE 5.15: ***Performance Analyzer***

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

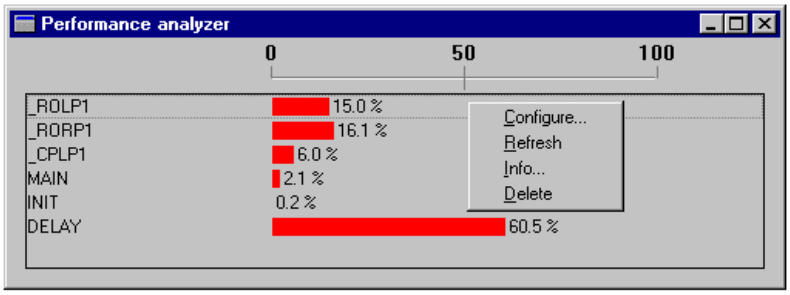

FIGURE 5.16: ***Performance Analyzer - Local Menu***

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.

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

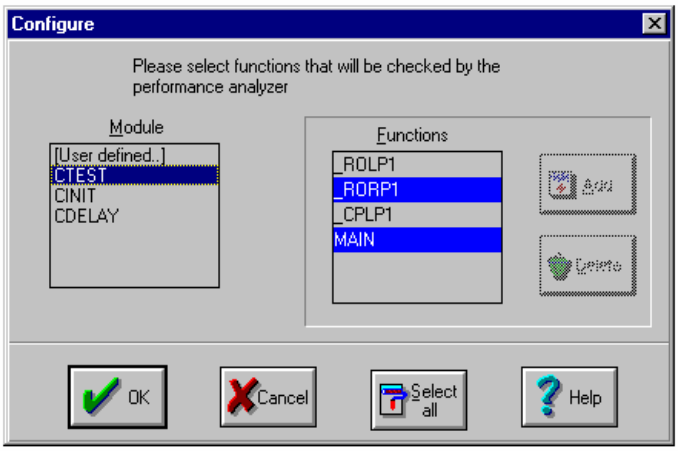

FIGURE 5.17: User Defined Dialog Box

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

Delete: Deletes the selected function from the screen.

Trace

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

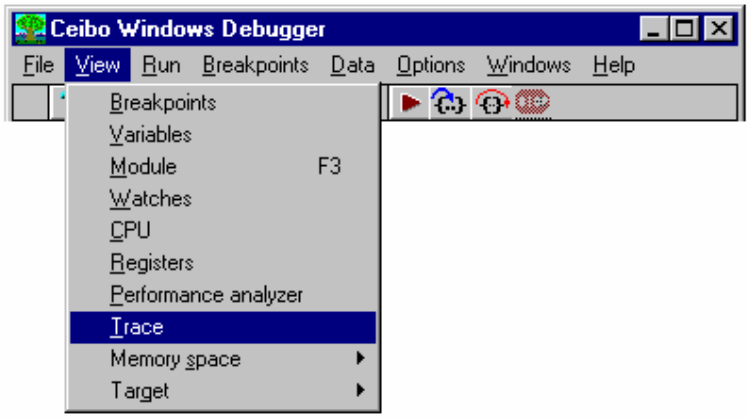

FIGURE 5.18: *Trace Menu*

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. The cursor in the CPU or Module window will point to the selected value.

Display Mode: A selection of source code, disassembled instruction or mixed source and disassembled code is available. If you change the setup, select read all trace to refresh the window.

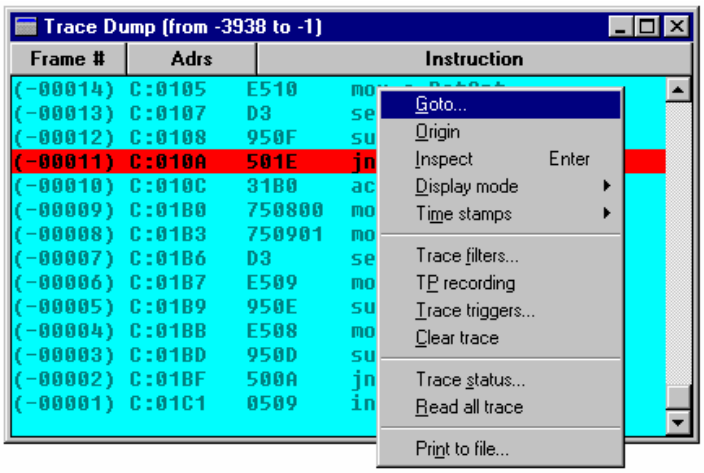

FIGURE 5.19: ***Trace Dump***

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). If you change the setup, select read all trace to refresh the window. Time stamps are displayed only if the selected Display Mode is All.

Trace Filters: Defines the trace filters of the displayed data. You may specify which instructions or sequences are of your interest. Selected regions are IN or ýOUT, where IN means enable and OUT means disable recording of marked lines in the window. The IN Select button toggles the window from all IN to OUT and vice versa. The last line of the window is All Range; this line is used to select or cancel all the memory to include or omit libraries, assembler modules and other code not defined in the debug information of your compiler. You may also add or delete user defined regions.

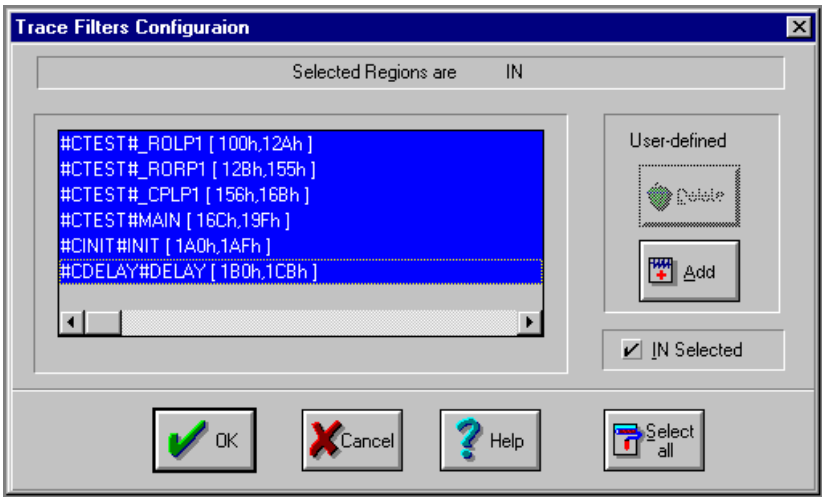

FIGURE 5.20: ***Trace Filters Configuration***

TP Recording: Enables or disables the recording of testpoints.

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

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 Triggers: This command sets the trace control to the selected option to start and stop the recording.

The different buttons and functions are:

Run begins: When execution is started, the trace buffer will capture data from the start program execution until the trace buffer is filled. This mode automatically selects the *Stop when Full* option.

Halt ends: Trace buffer capture is enabled and data is continuously collected until the break condition occurs. In this mode, the trace buffer will be filled with bus cycles immediately preceding the break condition.

From event: Trace capture begins when the Start event trigger condition is satisfied and continues until the program stops due to a breakpoint or Halt command. In this mode, the trigger event will be found in the beginning of the trace buffer contents. After clicking this button you must select the Start event button to complete the trigger definition.

Until event: Trace capture is enabled and the trace buffer filled until the Stop event trigger condition is satisfied. In this mode, the trigger event will be found in the end of the trace buffer. After clicking this button you must select the Stop event button to complete the trigger definition.

Dual event: This option combines both From and Until events, therefore you have to complete the selection by defining Start and Stop events.

Start event: Select the Start event button to define the trigger conditions for trace modes using a start event. This button will be dimmed if the selected trace mode does not require a start event.

Stop event: Select the Stop event button to define the trigger conditions for trace modes using a stop event. This button will be dimmed if the selected trace mode does not require a stop event.

Hardware Trace Triggers: These options are used to start and stop trace recording according to the testpoint signals connected to the clips. A selection of logic levels and edges is available. See Trace Clips explanation in Chapter 1.

After selecting the Start or Stop event buttons, the Set Trace Trigger dialog box will be displayed.

A trace trigger is the means for selecting the criteria for capturing an execution trace in the target system trace buffer. As we will see, setting a trace event and a hardware breakpoint are in fact the same operation. The only difference between the two operations is that action for a tracepoint is to turn the trace buffer on or off while breakpoints are used to implement the various breakpoint actions.

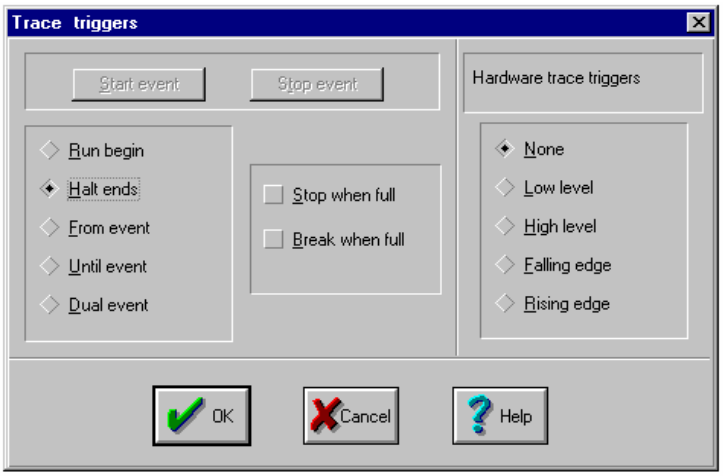

FIGURE 5.21: ***Trace Triggers***

The Trace Trigger definition is used to define the condition used to start trace data collection. When this option is selected, a dialog box containing field for

selecting the bus cycle type, address and data bus contents is displayed and you are prompted to fill in the conditions used to trigger the trace collection hardware in the target system.

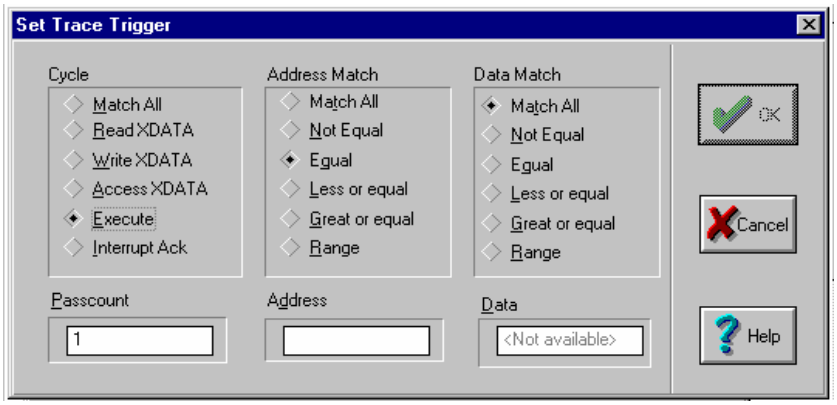

FIGURE 5.22: ***Set Trace Trigger Dialog Box***

When a trace trigger is defined, each field of the dialog box is combined to specify the event. Trace triggers operate on the recognition of user-defined events, using combinations of the following controls.

Cycle: Use this option to specify the bus cycle or execution state to be used in defining the trace trigger condition.

Address Match: Selects the address qualification mode. If any mode other than Match All is selected, the Address input box must be filled in with the address or address range to be recognized by the trace trigger. Addresses can be entered as symbols, modules and line numbers or as physical addresses.

Data Match: Selects the data qualification mode. If any mode other than Match All is selected, the Data input box must be filled in with the data or data range to be recognized by the trace trigger. Data can be entered as symbols.

Passcount: Selects the number of times the event will be recognized before the trace trigger occurs.

When all the condition have been specified, select OK to confirm the trace trigger setup.

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.

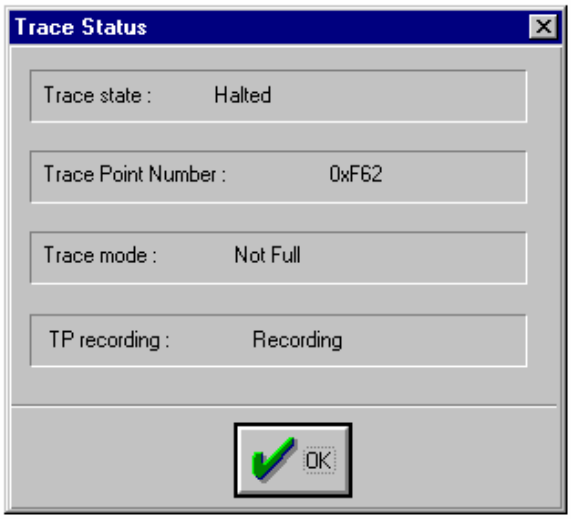

FIGURE 5.23: ***Trace Status***

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).

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

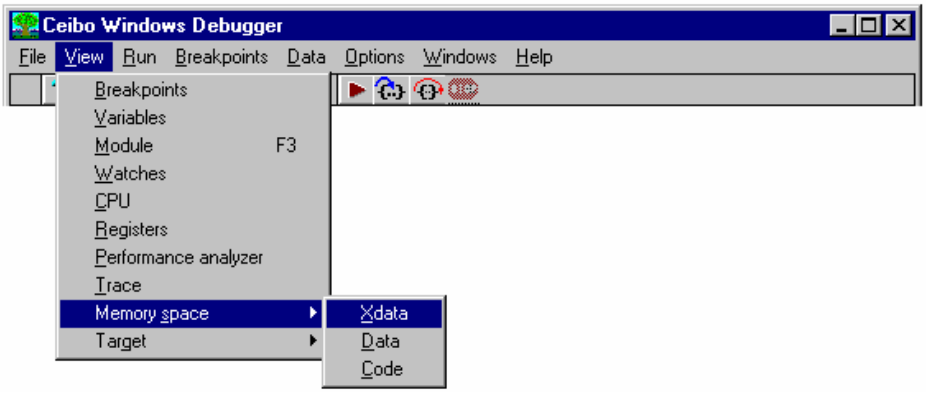

FIGURE 5.24: ***Memory Local Menu***

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.

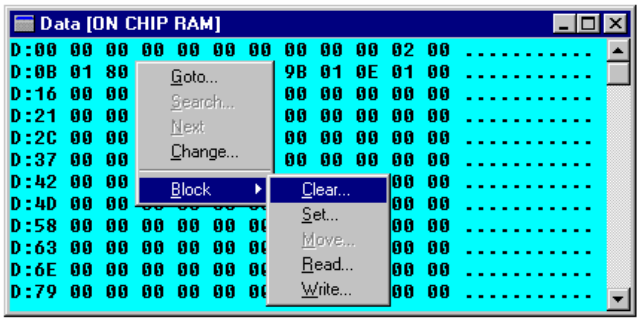

FIGURE 5.25: ***Block Menu***

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. The All option opens automatically all the available target windows.

FIGURE 5.26: ***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.

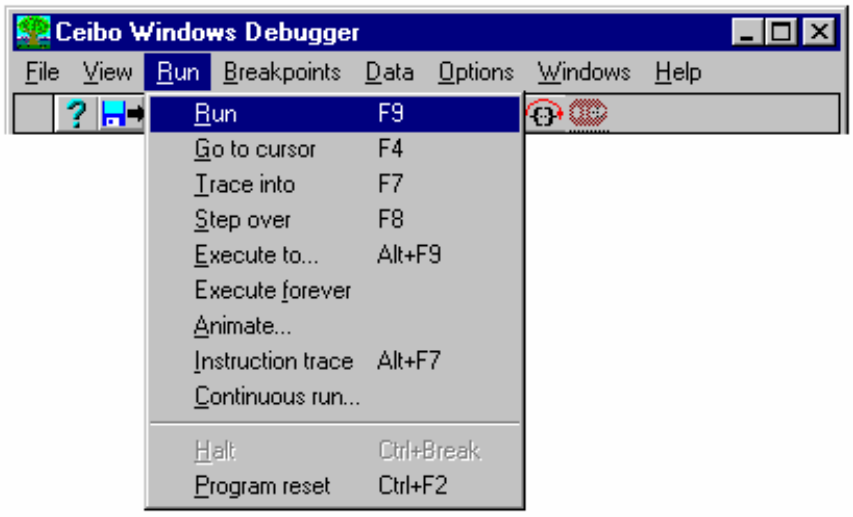

FIGURE 5.27: ***Run Menu***

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.

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.

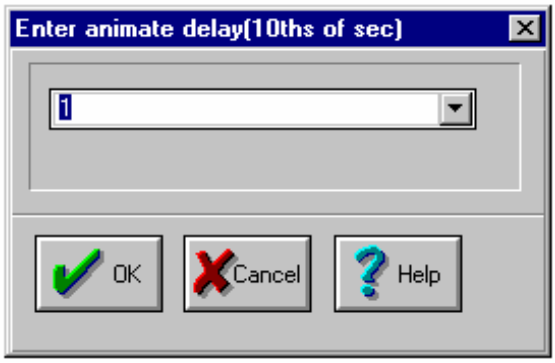

FIGURE 5.28: ***Animate Speed***

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.

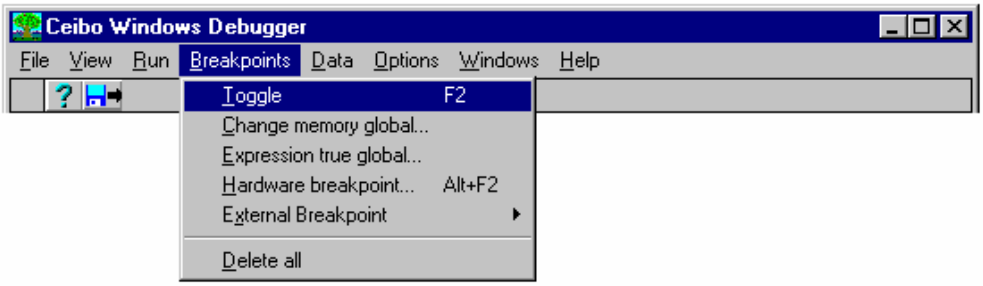

FIGURE 5.29: ***Breakpoints Menu***

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 Chapters 1 and 2). A selection of different logic levels and edges is available. The logic operator is used to select an AND/OR combination of external breakpoints.

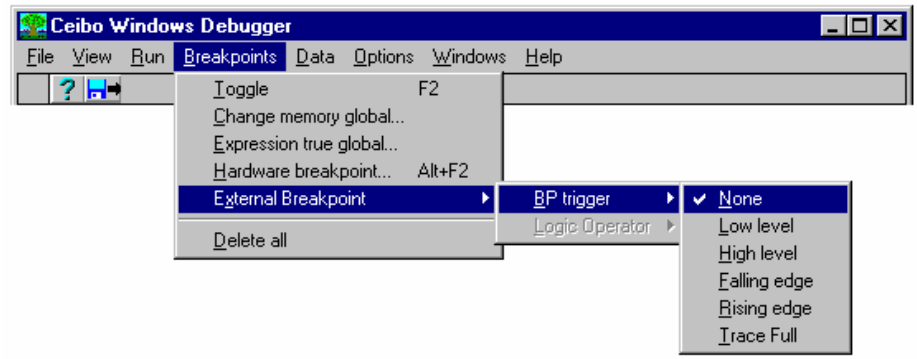

FIGURE 5.30: ***External Breakpoint***

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.

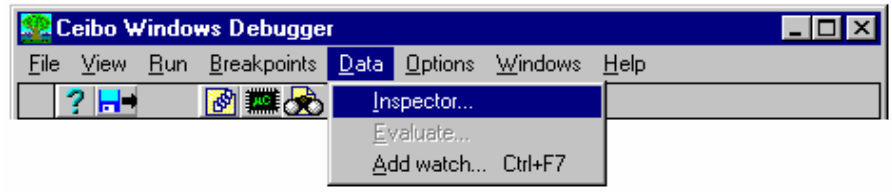

FIGURE 5.31: ***Data Menu***

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.

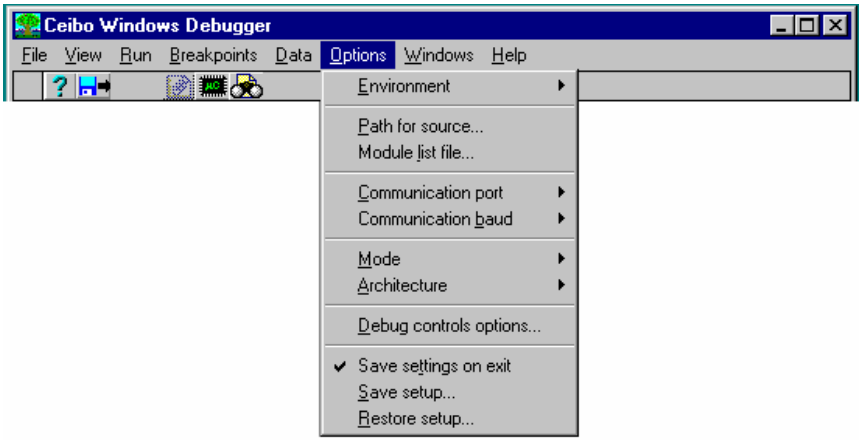

FIGURE 5.32: ***Options Menu***

The following are the available options: Environment, Path for Source, Module List File, Communication Port, Communication Baud, Mode, Architecture, Debug Control, 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, Beep on Breakpoint and Colors.

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.

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: Use this command to customize the colors of all the windows according to your preference.

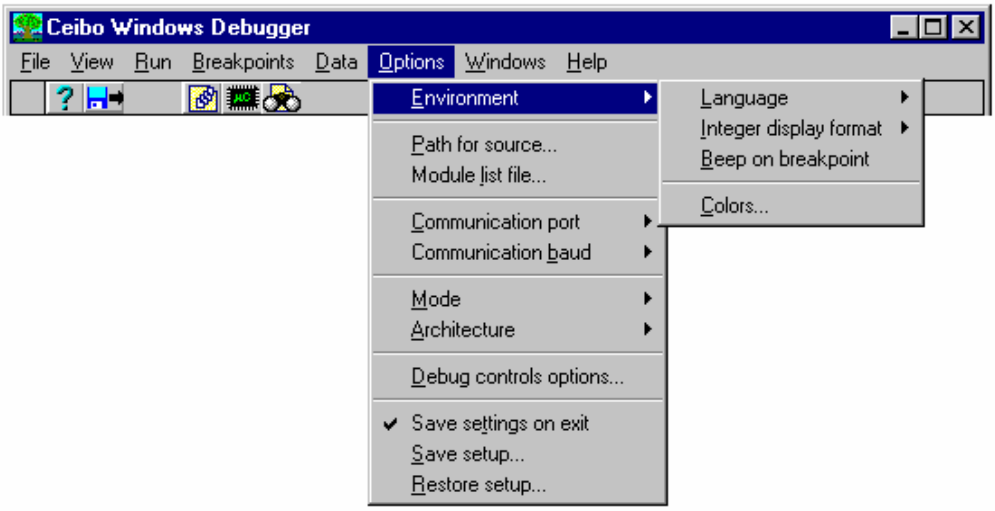

FIGURE 5.33: ***Environment Options***

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.

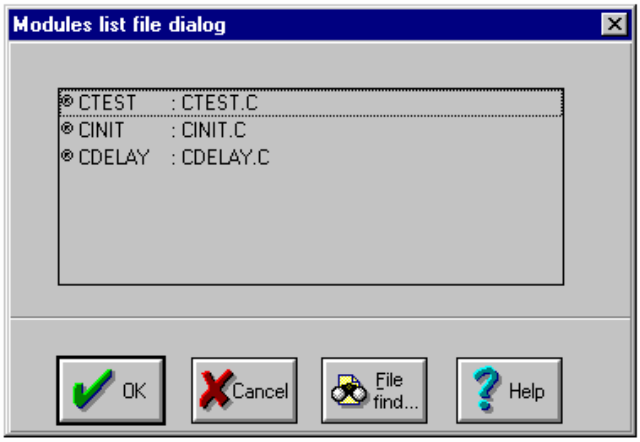

FIGURE 5.34: ***Module List Dialog Box***

Whenever loading a PLM or 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, In-Circuit Simulation 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 Ceibo's real time emulator systems. Communication error will result if the emulator is not found on the selected COM port.

In-Circuit Simulation: The In-Circuit Simulation Mode option sets the Windows Debugger to operate in a special simulation mode.

In this mode your program instructions will be simulated by the debugger built-in simulator, but any SFR references will be transferred to the remote emulator system, thus causing ports and other SFRs to change while simulating.

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

This is not a real-time mode; only basic 8051 functions are supported and not all SFRs belonging to particular derivatives. Set the system to emulation mode, while using the debugger with DS-51.

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 8051 functions are supported and not all SFRs belonging to particular derivatives. Set the system to emulation mode, while using the debugger with DS-51.

Architecture

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

The available options are: Chip, Map Code, Map Data, Xtal and Reset.

Chip: This command selects the emulated microcontroller type.

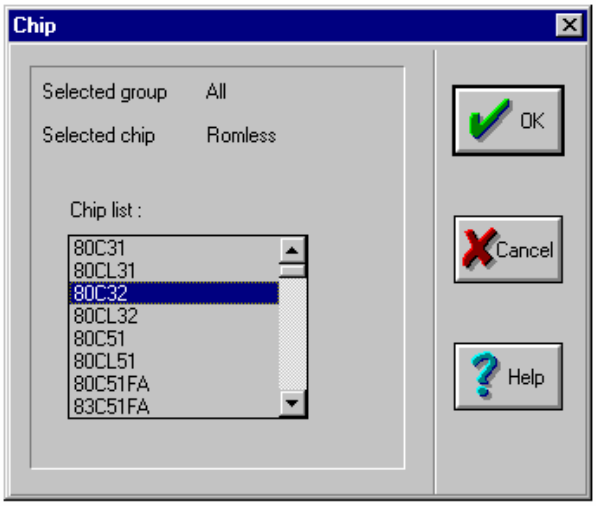

FIGURE 5.35: ***Chip Dialog Box***

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

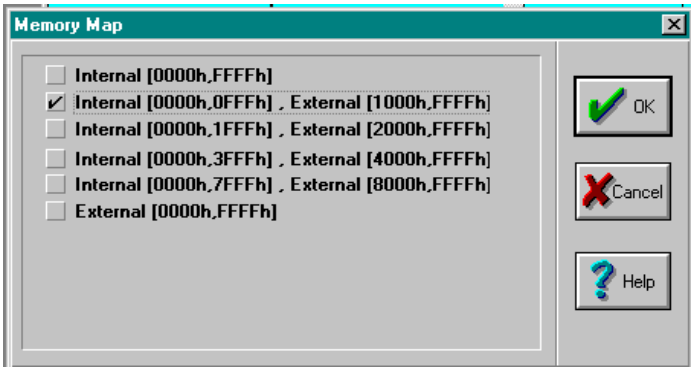

FIGURE 5.36: ***Map Code Window***

Map Data: The Map Data command allows you to define the data memory as belonging to the emulator 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. This command does not set frequency, which is defined by hardware (crystal on the probe).

User Reset: Enables or disables the reset signal generated by a target circuit.

Debug Control Options

These options are used to set many hardware functions related to the emulator.

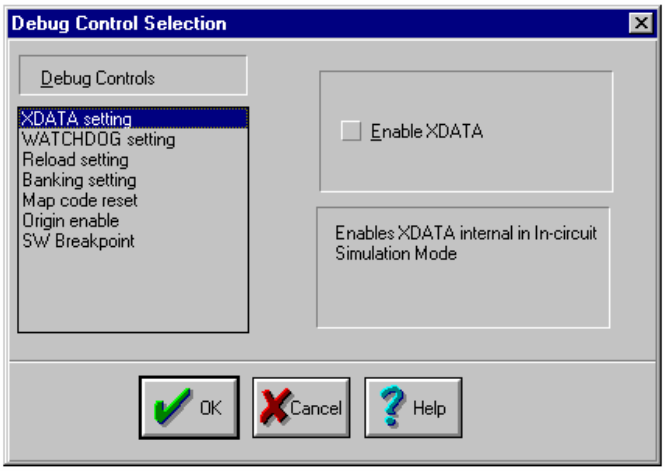

FIGURE 5.37: ***Debug Control Options***

From this option you may define the data map in the In-Circuit Simulation Mode, enable or disable the internal watchdog present in many microcontrollers, define when to reload your program, establish the type of banking if supported by your system. It allows or avoids resetting the program, while changing the code map, enables setting the cursor to the original position after halting the emulation and other special features.

A Software breakpoint can be defined only in RAM location, thus meaning in the internal code memory of the emulator. Set code memory to internal, if you want to use software breakpoints (Options|Architecture|Map code). Enable the Software Breakpoints on the Options/Debug control and use F2 in the source file to set breakpoints. These Software breakpoints stop the emulation "before" executing the assembly instruction of the specified address.

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.

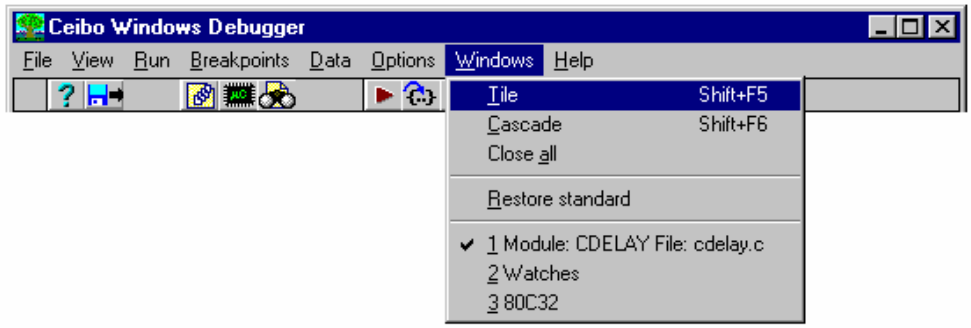

FIGURE 5.38: *Windows Menu*

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.

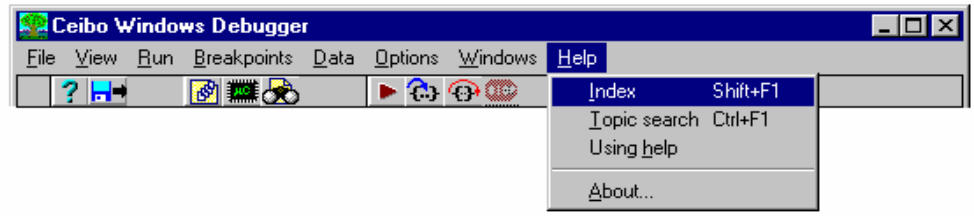

FIGURE 5.39: Help Menu

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

Working with the Hardware and Real-Time Emulation

CHAPTER 6

Working with the Hardware and Real-Time Emulation

6.1. Introduction

The In-Circuit Simulation Mode option executed your Program instructions with the simulator, but affects the ports and hardware registers of the microcontroller. Thus, the ports will be altered while executing your program. This mode is not real-time and is available as part of Ceibo's debuggers. It is not really useful with DS-51 since all the functions are supported in real-time. Only basic 8051 functions are supported and not all SFRs belonging to particular derivatives.

The Emulation Mode option sets the Debugger to operate in full emulation mode. In this mode your program will be executed in real-time.

6.2. Starting up

1. Connect the RS-232 cable to DS-51 and your computer COM Port.

2. Invoke the debugger.

3. Select the Options Menu, choose the right Communication Port and set the Mode to Emulation.

4. Apply a Reset by pressing Ctrl-F2. Now you are ready to operate the system in the Emulation Mode.

6.3. Port Testing

The following steps describe how to test a port connected to a target hardware. The simple program you are going to try is a counter on Port 1 acting as output.

1. Select the View Menu and open the Watches Window.

2. Add P1 (Port 1) to the watches.

3. Select the View Menu and open the CPU Window.

4. Use the on-line assembler to modify the CPU and change the code as follows:

   Address 0000h ..... INC P1

   Address 0002h ..... AJMP $0h

   The first instruction increments Port 1. The second instruction is a jump to the absolute address 0h. The $ symbol indicates that the numeric value is an absolute address and not the offset.

5. Press Ctrl-Break to halt the program execution. Observe that the Watches Window has been updated with the actual Port 1 value.

6. Select the View Menu and choose the Trace command to display the history of the executed instructions.

6.4. High-Level Language Debugging

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 CTEST.ABS sample file and Keil Type. This file includes code and symbols.

4. The file generates a sequence that is an output to the microcontroller port lines.

5. After successfully loading the CTEST.ABS file, the Module window will be opened with the updated code.

6. Open the Watches window.

7. Type P1 to add Port 1 to the Watches window.

8. Select the View Menu and the Modules command to choose the CTEST Module.

9. Select the Run Menu and execute a Program Reset. This can be done directly by pressing Ctrl-F2.

10. Execute the program by pressing the F9 key and observe with an oscilloscope the port outputs as they are changing while running the program.

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

12. Press Alt-X to leave the Debugger.

6.5. Special Hardware Considerations

Check that all the setup options are according to your needs, especially in the Mode, Architecture and Debug Controls of the Options Menu. Otherwise the system may not respond properly to your software and/or hardware stimulus.

CHAPTER 7

Utilities, Software Support and Syntax

CHAPTER 7

Utilities, Software Support and Syntax

7.1 Introduction

Two utilities are included in the software for assemblers and C compilers. FIXASM and FIXC are the utilities provided to support the listing files which are not fully compatible with Intel format. This chapter explains the use of these programs. The support for different software vendors and the Windows Symbolic Debugger syntax are covered in this chapter as well.

7.2 FIXASM Utility

This utility modifies the Assembler listing files to be accepted by the DOS Debugger. The Windows Debugger does not require this utility. Output file overwrites the input file, although you may select a different output filename.

Run FIXASM without parameters for command line syntax and options.

FIXASM acts as follows: it scans the input file and replaces the beginning of every valid line containing the assembler directives ' RSEG ' and ' CSEG ' with the pattern '----', as it is defined by Intel format.

- Intel ASM51 Assembler does not need to be fixed up, because it generates the listing file in the above mentioned format.
- IAR Assembler does not need to be fixed up, because it generates line information.
- Keil A51 Assembler needs to be fixed up because it lacks the above pattern in the ' RSEG ' directive line.

- MCC MA51 Assembler needs to be fixed up because it lacks the above pattern in the ' CSEG ' directive line.

7.3. FIXC Utility

This utility modifies the C listing files to be accepted by the DOS Debugger. The Windows Debugger does not require this utility.

The modification is intended to support two C Compilers: MCC and IAR.

Output file is by default input file name with .LST extension although it may be redirected to another output file.

Run FIXC without parameters for command line syntax and options.

FIXC with /M option is needed while using MCC MCC51 C Compiler package. This option adds line numbers to your C source file (i.e. filename.c). Output filename is by default the input filename with .LST extension. Output file must differ from input file. The generated output file is a line by line copy of the input file with a line counter added to the beginning of each source line. A total of 6 characters are added to the beginning of each line with the format: 'xxxx '.

FIXC with /A option is needed while using IAR ICC8051 C Compiler package. With this option FIXC expects the input file to be the listing output file of the compiler.

Output filename is by default input filename of your C source with .LST extension. The generated output file is a line by line copy of the input file with the line counter format changed to the following format 'xxxx '.

This modification is not mandatory. The debugger will fully function without changing the IAR listing files, although significant speed performance improvement will be achieved with large files after running the utility.

- Keil C51 Compiler does not need any modification.

- MCC MCC51 Compiler needs to be fixed up because it does not generate a valid list file with the correct line numbering format. Invoke the FIXC program with the /M option and with the source filename that was entered into the compiler.

- IAR ICC8051 Compiler fix-up is recommended for improved speed performance with large list files.

7.4. Debugging Assembler Files

Ceibo's Debuggers can be used with files generated by Intel, Signetics, Philips, BSO/Tasking and other Assemblers that generate Intel OMF files and compatible listings.

The first thing you have to do is to prepare the assembler files. Use the DEBUG option while assembling your program to include symbol information in the object file. The LIST option is also mandatory.

The Debug Option of your Assembler generates symbol information such as Line Numbers, Public and Local references, Labels and others. Ceibo's software recognizes those symbols.

The List Option creates an ASCII file that contains the source written by the user and references to Line Numbers. Ceibo's software allows invoking the List of your program and debugging it according to assembler language lines, while showing all the variables you want to watch.

The normal way to prepare your program for debugging is as follows:

1. Use an Editor to write your source code.

2. Assemble you sources with an Assembler program using the Debug and List options.

3. Link and locate your object files with the Intel RL51 program or a similar linker and locator that generates an Intel compatible object file.

If you are using an Assembler with linking and relocating capabilities (Intel, Keil, BSO/Tasking , etc.) you must write all your code under a CSEG or RSEG directive.

If the files are written to be linked with other programs, you may use the RSEG directive. Some assembler programs do not generate the required debugging information to support Relocatable Code Segments. Therefore, it is necessary to use the following syntax with RSEG:

myseg SEGMENT CODE

RSEG myseg

mylabel:

The only difference is the Label added to the third line, that is required to recognize the address of the relocatable code segment.

7.5. Using IAR Systems Compiler Package

Like any other compiler, working with IAR Systems compiler requires three major steps:

1. Writing a program in any given editor.

2. Compiling with ICC8051 compiler.

3. When no errors are detected, linking the programs to a desired memory location.

Compiling

The compiler must produce the following files :

1. File with extension *.R03 - object file
2. File with extension *.LST - listing file

Required compiler options:

-*r* : includes debug information for C source files

-*L* : produces list file

S : includes debug information for C assembler files

F : fixes listing for C assembler files

Linking

While linking object file with IAR Xlink the *.XCL file must be created. The required linker options are:

-*FAOMF8051* : Produce file in OMF-51 format for Ceibo DOS Debugger only.

-*FDEBUG* : Produce file in UBROF format for Ceibo Windows Debugger only.

-Y# : Use this switch, if you are using Ceibo's Debugger V1.09 only. It is not necessary for V1.10 or later.

When linking file with the IAR System environment Link, Options must be set to the following:

debug option 1: None

format 2: *aomf8051* for Ceibo DOS Debugger only or *debug* for Ceibo Windows Debugger

Preparing for Downloading

Before downloading your absolute file into the DOS Debugger only, you have to fix all your list files with the FIXC utility program.

The Windows Debugger does not require the FIXC utility.

The command line for running FIXC is :

FIXC <your module file name>.LST /A

Loading the file with the DOS Debugger

Load the file resulting from the Linker. The default extension of the file is *.A03*.

Loading the file with the Windows Debugger

Specify the vendor and load the file resulting from the Linker. The default extension of the file is *.DBG*.

7.6. Using Keil Software

This software is supported with the OMF format.

Use the following command lines with *noamake* in both C51 and L51:

c51 hello.c *debug objectextend noamake*

l51 hello.obj, my_libraries *noamake*

The *noamake* switch is necessary only for the Ceibo DOS Debugger.

While using Windows Debugger, select the vendor Keil in the Load Dialog Box. Load the output file from the L51 or BL51 linker. The file has no extension as default, although you may change the extension in the linker command.

7.7. Using BSO/Tasking Software

To prepare a file for debugging use the following options:

Compiler Switches

-g: include debug information

i.e. *cc51 -Ms -g -C51* sieve.c

Do not use the *-l* switch for proper source-level debugging.

ASM Switches

debug: include debug information

i.e. *asm51* sieve.src *debug* noprint

LINK Switches

The Linker does not need any special options.

Generate a file in OMF-51 Format

i.e.

DOS:
omf51 sieve.out sieve.abs

Windows:
ieee51 sieve.out sieve.iee

7.8. Using MCC Software

To prepare a file for debugging use the following options:

Compiler Switches

mcc51 hello.c */l1/t/w0* ;do not use spaces between options

ASM Switches

ma51 hello.src *debug nolist noprint format(momf)*

LINK Switches

ml51 hello.obj+*mcm51.lib format(iomf)*

Generating valid Debug Information

mcsu hello

Generating Valid Listing File for DOS Debugger with FIXC

FIXC hello.c hello.lst */m*

7.9. Using Avocet/Hitech Software

This software is supported by an external utility that converts the debug information to standard OMF51 Format. If you use Avocet of Hitech software, call Ceibo to receive the utility. Set the "List File of Type" to OMF51 Format in the Load command.

7.10. Windows Debugger Syntax

Predefined Names

DS-51 accepts the following symbols that you may invoke directly while adding a watch, setting a breakpoint or assembling an instruction. The complete list of

predefined Ports and Register names may be found in the Microcontroller Data Book.

Ports: P0, P1, P3, etc.

Port Bits: P0.0 to P0.2, P1.0 to P1.7, etc.

Registers: ACC, B, etc.

Register Bits: ACC.0 to ACC.7, B.0 to B.7, etc.

Absolute References

You can 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 for external RAM accessed by MOVX instructions, P for ports and any SFR (special function registers), C for code memory and B for bit memory (below 80H is for RAM bits and above that value represents SFR bits). Some examples are:

D:100,5

X:MYSYMBOL,2

B:0x80,5

B:P1.0,8

P:0xFE

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

Direct Access to Local Variables and Executable Lines

The general syntax for line numbers is:

#module name#line number

Local variables of modules are invoked as:

#module_name#variable

A local variable of a procedure has the following syntax:

#module_name#procedure_name#variable

Examples:

#test#12	means line number 12 belonging to module TEST.
#tcdelayt#delay#DelayCnt	means variable DelayCnt belonging to procedure delay and module tcdelayc.

CHAPTER 8

On-Line Assembler

CHAPTER 8

On-Line Assembler

8.1. Introduction

DS-51 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.

8.2. Assembler Syntax

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

- Use a valid symbol format. The DOS Debugger requires operands in hexadecimal or symbols.
- An H must follow the hexadecimal operands, while using the DOS Debugger.
- 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.

8.3. 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

CHAPTER 9

System Errors and Troubleshooting

CHAPTER 9

System Errors and Troubleshooting

9.1. Introduction

This chapter describes the errors detected while operating DS-51 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.

9.2. 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. The personality probe may be faulty or without clock.

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. Check the system with XTAL jumper set to internal.

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-51.

Error #21 - Communications

An invalid command has been transmitted to DS-51. This is actually not an error and may appear while your computer is trying to synchronize the baud rate with the emulator.

Error #22 - Communications

An invalid command has been transmitted to DS-51.

Error #32 - Communications

DS-51 does not respond to the PC command. The COM port is not working or something is wrong with the cable connections. Try a different COM port or another computer. Check the selected COM in the Options Menu. Check that the emulator is powered. Set baud rate to low.

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-51.

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.

9.3. Common Questions and Answers

ROMless Operation

1. How does the system work in ROMless mode?

ROMless operation means that the chip selected is 80C31, 80C51FA, etc., and ports 0 and 2 are the bus.

Port 0 can only be used as AD0-7 address/data bus lines.

Port 2 can only be used as A8-15 address lines.

ROMed Operation

2. How does the system work in ROMed mode?

ROMed operation means that the chip selected is 83Cxx, 87Cxx, 80C51, etc., and ports 0 and 2 are I/O ports.

Port 0 can only be used as I/O port.

Port 2 can only be used as I/O port.

Ports

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

Select chip type ROMed (like 87C51) and not ROMless (like 80C31).

Microcontroller

4. I replaced the microcontroller and the system does not work.

DS-51 may accept many different derivatives. DS-51 uses standard microcontrollers working in emulation mode. This special mode used by the emulator in ROMed mode, outputs the internal buses and it is not documented in the Philips Microcontroller Data Book. Programming the security bits is the operation required by the chip to accept entering in the emulation mode.

If you cannot program the security bits or you are not using a Philips microcontroller, select in the software menu a ROMless type (i.e. 80C32,

80C51FA, etc.) and not ROMed (i.e. 87C51FB, 87C...., etc.). However, microcontrollers without security bits programmed will work only with certain probes, like Probe C32.

5. Which security bits should I program, if I replace the microcontroller ?

Use Philips microcontrollers and program lock bit 1 and 2 only. Do not program lock bit 3.

6. How can I program security bits ?

Use Ceibo MP-51 or any other programmer with similar capabilities.

Power Supply

7. Should I turn on first the target or the emulator power supply ?

Turn on first the emulator power supply. Do not leave the emulator plugged to an unpowered target.

Low Voltage

8. How can I emulate the 87L51FB at 3V ?

Use Probe C51LV. Call Ceibo for more information.

9. Can I emulate the 8051 from 1.5V to 6V ?

Yes, use Probe C410.

Frequency

10. I set the software to XTAL 12MHz, but the clock did not change.

The XTAL command in the Options menu tells the software the working frequency for timing calculations in the Trace, but it does not set it. Use the crystal jumper or replace the crystal on the probe for different frequency settings.

Breakpoints

11. What is the difference between Hardware and Software Breakpoints ?

A Hardware Breakpoint may be set to any address location (ROM or RAM). It also stops the emulation "after" executing the assembly instruction of the specified address.

A Software breakpoint can be defined only in RAM location, thus meaning in the internal code memory of the emulator. Set code memory to internal,

if you want to use software breakpoints (Options|Architecture|Map code). Enable the Software Breakpoints on the Options/Debug control and use F2 in the source file to set breakpoints. These Software breakpoints stop the emulation "before" executing the assembly instruction of the specified address.

Trace

12. How should I use the trace triggers and testpoints ?

Connect the trace clips to your hardware (see Chapter 2 for the connections) and then use the local menu of the trace window for software setup.

13. Can I save the trace in a disk file ?

Yes, use the Print to file command in the local menu of the trace window.

14. I cannot display the time stamps in the trace.

Change the display mode and select "Read all" to refresh the trace window.

Banking Support

15. How is the banking support defined ?

Use the Options|Debug controls to define the setup. Your system must be factory prepared for banking support.

ALE Disable

16. How can I disable the ALE signal ?

ALE is necessary in the emulation mode and cannot be disabled.

System Problems

17. 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 system connected to the power supply; then the mode has changed from Emulation to Simulation and therefore you are working only with the simulator.

18. 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 black 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.

19. The system shows always Error #6 - crystal problems.

Set the crystal jumper to Internal.

20. I get a communication error with the DOS debugger but with the Windows debugger everything works.

Check the COM port setup in the Options Menu.

21. I get error #4 and #5 with the DOS debugger but with the Windows debugger everything works.

Check the Chip type setup in the Options Menu.

Software

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

Select chip type ROMed (like 87C51) and not ROMless (like 80C31).

23. Why some options are grayed out?

Not available yet in the current software version.

24. I am using BSO/Tasking assembler and when I try to enter a symbol in the Watches window, I get Error #311 and the values of the variable are represented as question marks (?????).

This assembler and many others are case sensitive, so enter the variable manually and always in upper case.

25. 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.

26. Probe C752 does not work with my interrupt routine.

Probe C752 uses an 87C752 to emulate 8XC750/1 microcontrollers. The main software difference between both types of chips is found in the Interrupt Enable Register (IE - address A8h). The ETI bit of the IE Register has different addresses. DS-752 locates the ETI bit at bit address ADh, while 8XC750/1 chips have that bit at address ABh. Therefore, if you are emulating an 8XC750/1 with the 87C752 you have to tell your assembler or

high-level language compiler that the target chip is an 8XC752 to generate the file for the emulator. After you finish to debug the software redefine the target chip as an 8XC750/1 and recompile it to generate the file for the 8XC750/1 programmer.

Again, the above manipulations are only necessary if your software is using the ETI register bit for an 8XC750/1. The 87C748 is equivalent to the 87C750 and the 87C749 is equivalent to the 87C752.

27. Which file should I load while using Keil software.

Load the output file from the L51 or BL51 linker. The file has no extension as default, although you may change the extension in the linker command.

28. Why the Code Dump Window displays several times the same addresses ?

Adjust the window size. The emulated microprocessor also sets the program counter to 0 while overflowing its maximum address (FFFFh or less according to the chip type).

29. Interrupts and Timers are not working in simulation modes.

Use real-time emulation mode for Interrupts and Timers. This support is not fully implemented yet in your current software version.

30. Why does the Code Dump Window display several times the same address ?

Adjust the window size. The emulated microprocessor also sets the program counter to 0,while overflowing its maximum address (FFFFh or less according to the chip type).

31. Which are the special Linker Options for IAR Version 2.01. ?

Add -Y# in the XCL file if your Ceibo Debugger is v1.09. Otherwise, it is not necessary.

32. Does the debugger run under Windows NT?

The current version does not support Windows NT. Check www.ceibo.com for the latest software update.

33. 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

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z