

Z8051[™] Family of 8-Bit Microcontrollers

Z8051 On-Chip Debugger and In-System Programmer

User Manual

UM024002-0512



Copyright ©2012 Zilog[®], Inc. All rights reserved. <u>www.zilog.com</u>



ii

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

©2012 Zilog, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z8051 is a trademark or registered trademark of Zilog, Inc. All other product or service names are the property of their respective owners.

Revision History

Each instance in the Revision History table below reflects a change to this document from its previous version.

Date	Revision Level	Description	Page No
May 2012	02	Updated for improved usability.	All
Mar 2012	01	Original issue.	All



Table of Contents

Revision History
List of Figures
Introduction
The Z8051 On-Chip Debugger
Features
Install the Z8051 OCD Software and Documentation
Z8051 OCD Driver Installation 4
Understanding the OCD Menu Functions 12
File Menu
Emulation Menu
Break/Configure Menu 17
View Menu
Window Menu
Child Windows
Z8051 Basic Registers Window
Code Disassemble Window
Code Dump Window
XDATA Dump Window 44
IRAM Dump Window
SFR Dump Window
Watch Global Window 52
Watch Local Window 54
Text File Window 57
The Z8051 OCD In-System Programmer
Features
Connect the Hardware
Apply Power
Understanding the OCD ISP Menu Functions
HexData Menu
Program Menu
Window Menu



Child Windows	79
Code Dump Window	80
XData Dump Window 8	81
Appendix A. OCD Driver Installation on Windows Vista Systems	82
Appendix B. OCD Driver Installation on Windows XP Systems	85
Customer Support	88



List of Figures

Figure 1.	On-Chip Debugger Screen
Figure 2.	Install Device Driver Dialog, Windows 7 5
Figure 3.	Browse For Driver Dialog, Windows Vista
Figure 4.	Can't Verify Publisher Dialog, Windows Vista
Figure 5.	Successfully Installed Dialog, Windows Vista
Figure 6.	Unsuccessful Installation, Scenario 1
Figure 7.	Unsuccessful Installation, Scenario 2 10
Figure 8.	Device Manager Dialog, Windows 7
Figure 9.	The OCD's File Menu 12
Figure 10.	Object File Dialog
Figure 11.	Open File Dialog
Figure 12.	Break Debug Dialog 14
Figure 13.	The OCD's Emulation Menu 15
Figure 14.	The OCD's Break/Configure Menu 17
Figure 15.	Break Control Dialog 18
Figure 16.	Break BEFORE Timing Diagram 18
Figure 17.	Break AFTER Timing Diagram 19
Figure 18.	Peripheral Control Dialog 19
Figure 19.	Z51F0811 MCU Configuration Example 20
Figure 20.	The OCD's View Menu
Figure 21.	The Basic Registers Dialog 23
Figure 22.	Code Disassembler Dialog 24
Figure 23.	Code Dump Dialog
Figure 24.	XDATA Dump Dialog
Figure 25.	IRAM Dump Dialog
Figure 26.	SFR Dump Dialog
Figure 27.	Global Variables Dialog
Figure 28.	Local Function Dialog
Figure 29.	A Sample Text File
Figure 30.	The OCD's Window Menu 31
Figure 31.	Cascaded Windows
Figure 32.	Tiled Windows
Figure 33.	Using the Basic Registers Function, #1 of 6 34



Figure 34.	Using the Basic Registers Function, #2 of 6 35
Figure 35.	Using the Basic Registers Function, #3 of 6 35
Figure 36.	Using the Basic Registers Function, #4 of 6 36
Figure 37.	Using the Basic Registers Function, #5 of 6 36
Figure 38.	Using the Basic Registers Function, #6 of 6 37
Figure 39.	Using the Code Disassembler Function, #1 of 3 38
Figure 40.	Using the Code Disassembler Function, #2 of 3 38
Figure 41.	Using the Code Assembler Function, #3 of 3 39
Figure 42.	Using the PC Break Toggle Function
Figure 43.	Using the Code Dump Function, #1 of 2 41
Figure 44.	Using the Code Dump Function, #2 of 2 42
Figure 45.	The Code Dump Function's Goto/Input Dialog
Figure 46.	The Code Dump Function's Pattern Load Dialog
Figure 47.	The Code Dump Function's Pattern Save Dialog 44
Figure 48.	The Code Dump Function's Pattern Fill Dialog
Figure 49.	Using the XDATA Dump Function, #1 of 2 45
Figure 50.	Using the XDATA Dump Function, #2 of 2 46
Figure 51.	The XDATA Dump Function's Goto/Input Dialog
Figure 52.	The XDATA Dump Function's Pattern Load Dialog
Figure 53.	The XDATA Dump Function's Pattern Save Dialog
Figure 54.	The XDATA Dump Function's Pattern Fill Dialog
Figure 55.	Using the IRAM Dump Function, #1 of 2 49
Figure 56.	Using the IRAM Dump Function, #2 of 2 50
Figure 57.	Using the SFR Dump Function, #1 of 3 51
Figure 58.	Using the SFR Dump Function, #2 of 3 51
Figure 59.	Using the SFRDump Function, #3 of 3 52
Figure 60.	The Watch Global Function's Global Variables Dialog
Figure 61.	Adding A Global Symbol 53
Figure 62.	Editing A Global Symbol 54
Figure 63.	The Watch Local Function Dialog 55
Figure 64.	Editing A Local Symbol 55
Figure 65.	Example Watch Local Function, #1 of 2 56
Figure 66.	Example Watch Local Function, #2 of 2 57
Figure 67.	Using the Text File Function, #1 of 5
Figure 68.	Using the Text File Function, #2 of 5
Figure 69.	Using the Text File Function, #3 of 5



Figure 70.	Text File Child Window 60
Figure 71.	Code Disassembler Child Window 61
Figure 72.	Using the Text File Function, #4 of 5 62
Figure 73.	Using the Text File Function, #5 of 5
Figure 74.	Example On-Chip Debugger ISP Screen 64
Figure 75.	OCD Hardware ISP Pin Assignment (Bottom View)
Figure 76.	The OCD ISP's File Menu 67
Figure 77.	Device Select Dialog
Figure 78.	Fill Buffer Dialog69
Figure 79.	File Open Dialog
Figure 80.	OCD ISP Dialog
Figure 81.	Select Device To Read Dialog
Figure 82.	Most Recently Used Files
Figure 83.	The OCD ISP's Program Menu
Figure 84.	Configuration Dialog
Figure 85.	The OCD ISP's Window Menu
Figure 86.	Open CODE Dump Child Window
Figure 87.	Open XData Dump Child Window
Figure 88.	Cascading Child Windows 78
Figure 89.	Tiled Child Windows
Figure 90.	CODE Dump Child Window 80
Figure 91.	XData Dump Child Window 81
Figure 92.	Found New Hardware Dialog, Windows Vista
Figure 93.	Install Device Driver Dialog, Windows Vista
Figure 94.	Couldn't Find Driver Dialog, Windows Vista
Figure 95.	The Found New Hardware Wizard Welcome Screen
Figure 96.	The Found New Hardware Wizard's Browse Screen

Introduction

The Z8051 On-Chip Debugger (OCD) and In-System Programmer (ISP) applications have been developed to support Zilog's Z8051 8-bit MCUs. This document describes how to set up and use the Z8051 OCD and ISP programs with your Z8051 Development Kit.



2

The Z8051 On-Chip Debugger

The Z8051 On-Chip Debugger enables a development PC to communicate with your target Z8051-based MCU. The OCD interface is used to connect the development PC and the Z8051 MCU. The OCD controls the Z8051 MCU's internal debugging logic, including emulation, step run, monitoring, etc., and can read or change the value of the Z8051 MCU's internal memory and I/O peripherals.

The Z8051 OCD supports emulation and debugging at the maximum frequency of the MCU and can support In-System Programming (ISP), thereby eliminating the requirement for an expensive emulator system.

The Z8051 OCD Debugger works with the Microsoft Windows XP, Vista (32/64) and Windows 7 (32/64) operating systems.



See the example On-Chip Debugger Screen shown in Figure 1.

Figure 1. On-Chip Debugger Screen



Features

The key features of the Z8051 On-Chip Debugger are:

- Supports Zilog's 8-bit Z8051 Family of MCUs
- Loads HEX and map/symbol files
- Allows symbolic debugging
- Supports the internal code memory of the target MCU
- Supports *In-System Programming-only* tools
- Displays code space using a disassembler
- Supports line assembly functions
- Toggles Program Counter (PC) breakpoints
- Supports the display and modification of RAM, SFR, registers, etc.
- Displays code, XDATA area using dump format
- Device autodetect:
 - Device configuration is not required
- Operating frequency:
 - Supports the maximum frequency of the target MCU
- Operating voltage:
 - Supports the entire voltage range of the target MCU
- Clock source:
 - Supports all X_{IN}, internal/external RCs, etc.
- Display emulation clock:
 - Counts and displays executed machine cycles
- Emulation and debugging:
 - Supports free run, step run, autostep run, etc.
- Save and load the development environment

Install the Z8051 OCD Software and Documentation

The Z8051 On-Chip Debugger (OCD) interface is the interface by which your PC will communicate with the Z8051 MCU to download and execute code. In addition to the OCD, software such as development tools and sample applications will be installed.



Note: If you have already installed the Z8051 software and documentation from the CD-ROM following the procedure on the paper insert in your kit (FL0138 or FL0139), skip this section and proceed to the Z8051 OCD Driver Installation section, below.

Observe the following procedure to install the Z8051 On-Chip Debugger software and documentation on your computer.

- 1. Ensure that the OCD interface hardware is not connected to your PC.
- 2. Insert the Z8051 Software and Documentation CD into your computer's CD-ROM drive. The setup program launches automatically. If the setup program does not launch automatically, open Windows Explorer, browse to your CD-ROM drive, and double-click the file labeled Z8051_<version_number>.exe.

Note: In this filename, <version_number> refers to the version number of the OCD Software and Documentation release. For example, this version number may be 1.0.

3. Follow the on-screen instructions to complete the OCD software installation.

Z8051 OCD Driver Installation

The driver programs for the Z8051 On-Chip Debugger are copied during the software and documentation installation. In the following procedure for PCs running Windows 7 32- and 64-bit operating systems, ensure that the target side of the OCD will remain unconnected while you install these drivers.

- ► Note: If you are running Windows Vista, see <u>Appendix A</u> on page 82 to install your device drivers. If you are running Windows XP, see <u>Appendix B</u> on page 85.
 - 1. Connect the OCD hardware to the USB port of your PC by connecting the A (male) end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port. Connect the Mini-B end to the OCD device.
 - 2. After the PC detects the new hardware, it will display the *Installing device driver software* dialog shown in Figure 2.





Figure 2. Install Device Driver Dialog, Windows 7

IMPORTANT NOTE: If you should encounter the scenarios presented in Figures 6 or 7, right-click your mouse on **ZILOG OCD I/F** (highlighted in Figure 6) or **Unknown device** (highlighted in Figure 7) and select **Update Driver Software...**

- 3. Select **Browse my computer for driver software (advanced)** to display the dialog shown in Figure 3, which prompts you to key in or browse for the location of the .inf file. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse** button to navigate to one of the following paths, then click the **Next** button.
 - On 32-bit machines, use the following path:

<Z8051 Installation>\Z8051_<version_number>\device drivers\OCD USB\x32

- On 64-bit machines, use the following path:

<Z8051 Installation>\Z8051_<version_number>\device drivers\OCD USB\x64



Bro	Found New Hardw	oftware on your	computer			
Sear	ch for driver softwar	e in this location:				
C:\/	Program Files\Zilog\	Z8051_1.0\device driv	ers\OCD USB\x32	-	Browse	
 Ir	nclude subfolders					

Figure 3. Browse For Driver Dialog, Windows Vista



4. When Windows prompts you whether to install or not install, as shown in Figure 4, click **Install this driver software anyway** and wait until the installation is completed (Windows may prompt you more than once).



Figure 4. Can't Verify Publisher Dialog, Windows Vista



An DIXYS Company 8

5. When the installation is complete, the screen shown in Figure 5 will appear. Click **Close** to exit the OCD driver installation.

Found New Hardware - Zilog Z8051 USB OCD	
The software for this device has been successfully installed	
Windows has finished installing the driver software for this device:	
Zilog Z8051 USB OCD	
	Close

Figure 5. Successfully Installed Dialog, Windows Vista

Note: On some installations, the Found New Hardware screen shown in Figure 5 may also display the text string, Zilog Z8051 USB OCD - No Firmware. This occurrence is normal and can be disregarded.

>



File Action View Help	
a 🚔 bpak-b7yc8n1	
Batteries	
Bluetooth Radios	
> 📲 Computer	
Disk drives	
🕟 🌉 Display adapters	
DVD/CD-ROM drives	
Human Interface Devices	
D IDE ATA/ATAPI controllers	
🔋 拱 IEEE 1284.4 compatible printer	
> 📾 IEEE 1284.4 devices	
👂 🟺 IEEE 1394 Bus host controllers	
🔊 \overline a Imaging devices	
> - Keyboards	
>- 💾 Mice and other pointing devices	
Monitors	
Network adapters	
Other devices	
HEWLETT-PACKARD DESKJET 990C	
ZILOG OCD I/F)	
Portable Devices	
Ports (COM & LPT)	
Processors	
Sound, video and game controllers	
> 🛀 System devices	
a 🏺 Universal Serial Bus controllers	
Generic USB Hub	
🖨 Generic USB Hub	
🚽 员 Generic USB Hub	
🚽 🖣 Generic USB Hub	
HP Color LaserJet 2840 EWS	
Intel(R) 5 Series/3400 Series Chipset Family USB	Enhanced Host Controller - 3B34
Intel(R) 5 Series/3400 Series Chipset Family USB	Enhanced Host Controller - 3B3C
USB Composite Device	
USB Composite Device	
🟺 USB Composite Device	Device driver software was not successfully installed & X
🟺 USB Composite Device	Click here for details

Figure 6. Unsuccessful Installation, Scenario 1



🐣 Device Manager	E X
<u>File Action View Help</u>	
⊿ 🚔 bpak-b7yc8n1	
🔉 🎲 Batteries	
p 🚯 Bluetooth Radios	
⊳ - 1 Sector Computer	
🖻 👝 Disk drives	
b. Number of the second sec	
DVD/CD-ROM drives	
🗄 🖓 Human Interface Devices	
IDE ATA/ATAPI controllers	
🔋 🦏 IEEE 1284.4 compatible printer	
Definition IEEE 1284.4 devices	
p - 🖉 IEEE 1394 Bus host controllers	
Imaging devices	
b — Keyboards	
Mice and other pointing devices	
Monitors	
Network adapters	
The Heleneum device	
Detable Devices	
Dete (COM & LDT)	
Sound video and game controllers	
Sound, Made and game conditions	
Iniversal Serial Bus controllers	
Generic USB Hub	
Generic USB Hub	
Generic USB Hub	
- 🗍 Generic USB Hub	
🚽 🗑 Generic USB Hub	
🛶 🏺 Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34	
🛶 🏺 Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C	
🛶 📮 USB Composite Device	
USB Composite Device	
USB Composite Device	
USB Composite Device	
USB Mass Storage Device	
USB Root Hub	
USB Root Hub	
	_

Figure 7. Unsuccessful Installation, Scenario 2

An DXYS Company

6. If *Zilog Z8051 USB OCD* appears in the Device Manager (as highlighted in Figure 8), the OCD driver software has been successfully installed.



Figure 8. Device Manager Dialog, Windows 7



Understanding the OCD Menu Functions

This section describes the operation of the File, Emulation, Break/Configure, View, Window menus.

File Menu

The File menu enables you to perform basic commands in the debugger environment. Its two commands, Load Hex and Save Hex, are described in this section.

- The Load Hex command is used to load user code to the target MCU's code space.
- The Save Hex command is used to save the contents of the target MCU's code space to a file on your computer.

The OCD's File menu is shown in Figure 9.



Figure 9. The OCD's File Menu

Observe the following procedure to load a user hex code file to the target MCU's code space.

 Run the Z8051 OCD software. From the Start menu, navigate to All Programs → Zilog Z8051 Software and Documentation <version_number> → Zilog Z8051 OCD <version_number>.

Note: For a free download of the latest version of the OCD software, visit <u>the Zilog website</u> and navigate via the **Tools and Software** menu to **Software Downloads**.

2. From the **File** menu of the Debugger, select **Load Hex**. The Object File dialog box appears, as shown in Figure 10.



Hexa file name		Common area size
Bank0 amples\Z51F0811\Demo\demo.hex	Browse	✓ No common area 0 ~ 0x7FFF (32KB)
Bank1	Browse	0 ~ 0x3FFF (16KB) 0 ~ 0x8FFF (48KB)
Bank2	Browse	Symbol file
Bank3	Browse	Sympoline Terringland we late a feature to the second
Bank4	Browse	Do VERIFICATION after download
Bank5	Browse	Job
Bank6	Browse	
Bank7	Braviee	
Bank8	Browse	
Bank9	Browse	
Bank10	Browse	
Bank11	Browse	
Bank12	Browse	
Bank13	Brawse	
Bank14	Browse	
Bank15	Browse	

Figure 10. Object File Dialog

3. The **Hex file name** panel, located on the left side of the Object File dialog, displays 16 banks. If you are using the Z8051 MCU's LINEAR ADDRESS Mode, you are not required to select additional banks; LINEAR ADDRESS Mode uses only Bank 0. Click the **Browse** button for Bank 0 to display the Open File dialog shown in Figure 11.



Open					?×
Look in:	Demo		-) 🕫 😕 🛄 🗸	
My Recent Documents Desktop My Documents My Computer	Clock.h demo.hex eeprom.h leds.h main.h switches.h timer.h wt.h				
Flaces	File <u>n</u> ame:	demo.hex		•	<u>O</u> pen
	Files of type:	Hex file (*.H*)		•	Cancel

Figure 11. Open File Dialog

- 4. In the Open File dialog, select the hex file that you wish to load into the memory space of the target MCU, and click **OK**.
- 5. If previous PC breakpoints exist in the debugger environment, the Break Debug dialog box will appear, as shown in Figure 12. Click **Yes** if you wish to remove these breakpoints, or **No** if you prefer to retain them.

MCS51_dbg	×
Clear previ	ous PC break(s) ?
Yes	No

Figure 12. Break Debug Dialog



- 6. The debugger will automatically search for map and symbol files associated with the hex file and load these files to memory.
- 7. After the map/symbol files are loaded into memory, the debugger resets the target MCU and moves the MCU's program counter to 0000h.
- 8. Save the current debugging environment to the hard drive of your development PC and exit the Debugger by selecting **Exit** from the **File** menu.

Emulation Menu

The Emulation menu, shown in Figure 13, lists the controls for starting or stopping an emulation routine. Use the Emulation menu to control the flow of code execution for debugging purposes.



Figure 13. The OCD's Emulation Menu

The remainder of this section describes the features of the Emulation menu.

Reset & Go

This menu selection starts an emulation from the 0000h address upon a reset of the target MCU, and functions in a manner similar to a Power-On Reset. Emulation continues until a breakpoint occurs or the user stops the emulation process. The Reset & Go menu is disabled (greyed out) in the Emulation menu during emulation.



Go From

The Go From menu selection starts emulation from a user-specified address, and is used to debug each software module. The user is prompted to enter an emulation start address, as follows:

- Using LINEAR ADDRESS Mode, enter a 20-bit address directly.
- Using BANKED ADDRESS Mode, enter 4 bits of bank and 16 bits of address. Each bank size is smaller than or equal to 64KB.

The Go From function is disabled (greyed out) in the Emulation menu during emulation.

Go

The Go function begins emulation from the *current address*, which can be characterized as:

- The last known (stopped) address that was held in the Program Counter from a previous emulation session
- The point at which a break occurs, such that:
 - If a break occurs before the breakpoint, the current address is the PC breakpoint address
 - If a break occurs after the breakpoint, the current address is the next execution address of the PC breakpoint address
- If the target MCU was reset, the reset address is 0000h.

The Go function is disabled (greyed out) in the Emulation menu during emulation.

Step

The Step function is used to debug each instruction flow and process one step at a time; the target MCU program flow will execute only one instruction at a time, then halt.

If the MCU receives a CALL instruction, it executes a Step run into the subroutine. If MCU is in STOP Mode, the Step run is ignored. The Step function is disabled (greyed out) in the Emulation menu during emulation.

Step Over

The Step Over function is used to check main program flow when each subroutine had been tested already. This function is similar to the Step function, with the exception of its subroutine call. If the MCU receives a CALL instruction, the debugger assumes the CALL and its subroutine to be one instruction, even if the subroutines are nested.



If the Step Over function reaches a PC breakpoint condition, emulation is halted. This function is disabled (greyed out) in the Emulation menu during emulation.

Step Auto

Using the Step Auto function, a step run is executed every 100ms; execution will continue unless the user halts it. This function is disabled (greyed out) in the Emulation menu during emulation.

Break

Using the Break function, emulation is halted immediately, even if the MCU is in STOP Mode. This function is disabled (greyed out) in the Emulation menu during emulation.

Reset

The Reset function releases a hardware reset signal to the target MCU, which is then reinitialized. Emulation is not halted when the MCU is emulating; however, this function has no effect when the target MCU is idle. The Reset function is enabled in the Emulation menu whether an emulation is running or is idle.

Break/Configure Menu

The Break/Configure menu, shown in Figure 14, lists PC breakpoint control, device configuration and hardware test functions.



Figure 14. The OCD's Break/Configure Menu

Clear ALL Break

The Clear ALL Break function immediately clears all PC breakpoints. This menu is disabled (greyed out) in the Break/Configure menu during emulation.



Break BEFORE (AFTER)

The Break BEFORE (AFTER) function prompts the user to select a PC breakpoint event either before or after execution. When selecting this menu option, the Break Control dialog box appears, prompting the user to choose one of these two conditions; see Figure 15.



Figure 15. Break Control Dialog

Selecting **Break before execution** causes a PC breakpoint when the PC reaches the PC breakpoint address; however, a PC breakpoint position will not be executed, as illustrated in the timing diagram shown in Figure 16.





Selecting **break after execution** causes a PC breakpoint to occur when the PC reaches the PC breakpoint address, and a PC breakpoint position is executed, as illustrated in the timing diagram shown in Figure 17.





Figure 17. Break AFTER Timing Diagram

This Break BEFORE (AFTER) function is disabled (greyed out) in the Break/Configure menu during emulation.

Peripheral Control

Selecting the Peripheral Control function from the Break/Configure menu prompts the user to determine whether the target MCU's internal peripheral functions should continue to operate or remain idle, as shown in Figure 18. These peripherals are always running during emulation by default.



Figure 18. Peripheral Control Dialog

The purpose of the Peripheral Control function is to tell the OCD whether the peripherals should be stopped or continuously run during Break (Debug) Mode. All peripherals,



including the PLL and ADC functions, will be stopped when selecting **Stopped concurrently**. The Peripheral Control menu selection is disabled (greyed out) in the Break/Configure menu during emulation.

Note: The Peripheral Control function does not control each peripheral individually.

Chip Configuration

>

The Chip Configuration function is used to configure the target MCU's I/O pin function, oscillation method, code protection, etc. Each device series features different configurations. If a configuration changes, the user must turn off power to the target MCU, then power it on again. As a result, configurations can be influenced when power rises to operational voltage.

The Configuration dialog box shown in Figure 19 offers an example configuration for the Z51F0811 device.

Configuration dialog	×
	Boot area
Lock code area	✓ 256B : 0100 ~ 01FF
🔽 Lock xdata area	768B : 0100 ~ 03FF
Disable /RESET input	☐ 1.7KB : 0100 ~ 07FF
Enable Xin, Xout	☐ 3.8KB : 0100 ~ OFFF
Enable Sub-Xin, Xout	Enable HARD lock
Configuration dum	p
0000 : 00 E8 13 03 08 0008 : 00 00 00 00 00	00 00 00 00 Write
0010:00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 Close

Figure 19. Z51F0811 MCU Configuration Example

The Chip Configuration menu selection is disabled (greyed out) in the Break/Configure menu during emulation.



Hardware Test to Repair

The Hardware Test to Repair function is used for OCD emulator testing and repairing. Its subfunctions are not available to the user.

View Menu

The View menu, shown in Figure 20, supports the opening of child windows. The Debugger offers the following nine types of child windows:

- MCS51 basic registers
- Code disassembly
- Code dump
- XDATA dump
- IRAM dump
- SFR dump
- Watch Global
- Watch Local
- Text file





Figure 20. The OCD's View Menu

Each of the View menu's functions are described in this section.

Toolbar

The Toolbar menu selection displays or hides the debugger's toolbar. This toolbar is located on the upper left side of the debugger frame. The toolbar displays frequently used menu buttons for the user's convenience.

Emulation Toolbar

This menu selection displays or hides the emulation toolbar, which is located to the right of the main toolbar described above. The emulation toolbar displays frequently used emulation control menu buttons for the user's convenience.

Window Open Bar

This menu selection displays or hides the window open bar, which is located to the right side of the emulation toolbar described above. The window open bar displays menu buttons that can be used to open child windows.



Status Bar

This menu selection displays or hides the status bar, which is located at the bottom of the debugger frame. The status bar displays simple help features, the emulation clock count, etc.

Z8051 Basic Registers

This menu selection opens a window that displays the Z8051 Series' basic registers. If this window is already open, selecting the **Z8051 Basic Registers** option will cause this window to appear at the top-most level of the debugger frame. See Figure 21.

🚑 Basic regi	sters							. 🗆 🗙
	RO	R1	R2	RЗ	R4	R5	R6	R7
Reg #0	68	00	00	00	00	77	16	00
Reg #1	00	00	00	00	00	00	00	00
Reg #2	96	00	00	00	00	00	00	00
Reg #3	00	00	00	00	2C	01	00	00
	CY	AC	FO	RS1	RSO	OV	F1	PA
PSW	0	0	0	0	0	0	0	0
ACC	00	D	PTR	0	000			
SP	07		_	Mod	lify		Cance	el le

Figure 21. The Basic Registers Dialog

The Z8051 Basic Registers menu selection is disabled (greyed out) in the View menu during emulation.

Code Disassembly

This menu selection opens a window which displays the memory spaces containing disassembled code. If this window is already open, selecting **Code Disassemble** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 22.



🗁 Code d	disassemble	er		×
Bank # 0) 🔻 🗌	Goto	PC 00000 Change	
0 0000	0201BA	LJMP	001BA	
0_0003	7BFF	MOV	R3,#0FF	
0_0005	E4	CLR	A	
0_0006	FD	MOV	R5,A	
0_0007	7F01	MOV	R7,#001	
0 0009	8B08	MOV	008,R3	
0_000B	8A09	MOV	009,R2	
0_000D	890A	MOV	00A,R1	
0_000F	EF	MOV	A,R7	
0_0010	14	DEC	A	
0_0011	600F	JZ	00022	
0_0013	14	DEC	A	
0_0014	6011	JZ	00027	
0_0016	14	DEC	A	
0_0017	6013	JZ	0002C	
0_0019	2403	ADD	A,#003	
0_{01B}	7012	JNZ	0002F	
0_001D	750D80	MOV	00D,#080	
0 0020	800D	SJMP	በበበ2ፑ	-
•				►

Figure 22. Code Disassembler Dialog

The Code Disassemble menu selection is disabled (greyed out) in the View menu during emulation.

Code Dump

This menu selection opens a window which displays the contents of code memory in a *dumped* format. If this window is already open, selecting **Code Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 23.



🗖 0	ODE	Е:В	ank_	0 00	04B :	: Pag	je C	S 50	5F									x
Bank	(# C	00	•		Goto)		Patte	rn	Lo	ad		Save	•		Fill		
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0_00	00	02	01	BA	7B	FF	E 4	FD	7 F	01	8B	08	8A	09	89	ΟA	EF	
0_00	10	14	60	0 F	14	60	11	14	60	13	24	03	70	12	75	0D	80	
0_00	20	80	0 D	75	0 D	C0	80	80	75	0 D	94	80	03	75	0 D	D4	ΕD	
0_00	30	25	0 D	F5	0 D	C2	00	F5	0 E	12	01	80	<u>E</u> 4	F5	0 B	AB	80	
0_00	40	AA	09	Α9	0 A 0	85	0B	82	75	83	00	12	00	E2	F5	0C	60	
0_00	50	11	D2	00	85	00	0 E	12	01	80	05	0B	E5	0B	C3	94	40	
0_00	60	40	DC	22	C2	88	C2	89	C2	8 A	12	01	5 F	7 F	20	12	01	
0_00	70	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
0_00	80	12	01	Α9	12	01	AO	7 F	10	12	01	Α9	7 F	00	7E	10	12	
0_00	90	01	63	E 4	12	01	Α9	7 F	60	12	01	Α9	C2	00	75	0 E	01	
0_002	0A	12	01	80	C2	00	75	0E	28	02	01	80	75	A0	FF	75	98	
0_00	B0	FF	12	00	63	7B	FF	7A	01	79	3B	E 4	FD	FF	12	00	09	
0_00	CO	7A	01	79	44	12	0.0	03	12	01	5 F	7B	FF	7A	01	79	4 D	
0_00	D0 I	E 4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5 F	
0_001	ΕO	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	ЗA	F5	83	Ε0	
0_00	FO	22	50	06	E 9	25	82	F8	E6	22	BB	FE	06	E 9	25	82	F8	

Figure 23. Code Dump Dialog

The Code Dump menu selection is disabled (greyed out) in the View menu during emulation.

XDATA Dump

This menu selection opens a window which displays the contents of XDATA memory in a dumped format. The term *XDATA* refers to the external data memory contained in Z8051 Series devices. If this window is already open, selecting **XDATA Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 24.



É	XDATA : Bank_0 00CF : Page CS 0096													•	x			
	Bank #	00	•		Goto			Patte	rn	Lo	ad		Save			Fill		
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0	0000	12	34	50	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0	_00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Figure 24. XDATA Dump Dialog

The XDATA Dump menu selection is disabled (greyed out) in the View menu during emulation.

IRAM Dump

This menu selection opens a window which displays the contents of IRAM memory in a dumped format. The term *IRAM* refers to the internal data memory contained in Z8051 Series devices. If this window is already open, selecting **IRAM Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 25.



ीं IRAM : 0C5															x			
	Pattern		Save	•														
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
	0000	EC	8B	FB	E3	Β7	4A	52	00	18	B2	F4	62	00	0C	AE	11	
	0010	4 D	95	83	EC	EΒ	66	ED	Α9	Ε1	83	BB	80	FF	3A	FF	42	
	0020	DF	02	AC	01	30	80	4B	20	4C	81	FA	BA	00	0C	D1	40	
	0030	55	53	C1	80	73	9A	22	10	8B	DC	40	66	68	49	F8	AA	
	0040	D9	Α6	F9	99	CO	CO	D9	21	42	75	39	F6	88	66	В5	43	
	0050	32	28	ΒA	D4	62	12	66	8A	D1	11	7C	73	0C	7C	C2	0 E	
	0060	C2	C 9	Α3	3B	F4	C3	СВ	00	AC	0 E	D7	C1	0 D	0C	31	80	
	0070	DO	E3	62	ЗA	E 9	56	D2	E 9	94	20	FC	7 F	E 9	EΒ	89	ΕE	
	0080	00	C2	80	В8	6 F	FB	5B	E3	51	12	Α7	В3	D2	FB	DC	AE	
	0090	0 B	ΕE	00	8B	71	7B	35	6D	1 F	E6	34	26	E6	41	98	F1	
	00A0	DD	7 F	ВD	8D	B8	EВ	2 F	ED	Β4	1D	9D	C5	5 D	Β6	D9	ΕE	
	00B0	44	6D	2 D	85	49	<u>C7</u>	55	5E	29	3F	76	65	7E	DD	77	BF	
	00C0	AD	06	26	43	AA	D9	F4	3E	82	91	32	4B	F9	Β6	1E	F5	
	00D0	7 F	C1	E5	C5	FF	B8	Β6	AE	7 F	Α3	2E	A0	Β7	2 D	DD	DA	
	00E0	E5	BE	C1	В0	FF	FC	CF	D5	43	9C	27	76	4 E	5 F	38	В5	
	00F0	Ε7	1E	B2	70	BB	BE	В9	BF	ΕO	9 F	E1	69	77	5E	CD	ED	

Figure 25. IRAM Dump Dialog

The IRAM Dump menu selection is disabled (greyed out) in the View menu during emulation.

SFR Dump

This menu selection opens a window which displays the contents of the SFR peripherals in a dumped format. The term *SFR* refers to the special function registers contained in Z8051 Series devices. If this window is already open, selecting **SFR Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 26.



🛠 SFR : 0A8																		x
	Pattern	n <u>R</u>	efre	sh	S	ave												
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
	0080	02	07	00	00	00	00	D9	00	03	00	04	87	FB	00	00	DF	
	0090	04	00	00	00	00	00	00	00	00	00	8 F	01	FO	00	00	00	
	00A0	00	CO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	00B0	00	00	00	00	00	00	00	00	00	00	7 F	7 F	00	00	00	00	
	00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	00D0	00	00	00	00	00	00	42	00	00	00	00	00	3F	3F	01	FF	
	00E0	00	00	00	00	00	80	FF	00	00	00	00	03	80	00	00	00	
	00F0	00	00	00	00	00	00	00	00	00	00	03	C0	60	08	00	00	

Figure 26. SFR Dump Dialog

The SFR Dump menu selection is disabled (greyed out) in the View menu during emulation.

Watch Global

This menu selection opens a window that displays global variables. If this window is already open, selecting **Watch Global** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 27.

🔒 Global variables					×
Add symbol	Remove	symbol			
Туре	Attribute	Name	Value	Address	•
BIT	BIT	LCD_E	0	0x88.2	
BIT	BIT	LCD_RS	1	0x88.0	
BIT	BIT	LCD_RW	1	0x88.1	Ξ
unsigned char	DATA	R0_io	0x0	0x98	
unsigned char	DATA	R0_port	0x2	0x80	
BIT	BIT	R03	0	0x80.3	
unsigned char	DATA	R1_io	0x0	0xA0	Ŧ
•					•

Figure 27. Global Variables Dialog

The Watch Global menu selection is disabled (greyed out) in the View menu during emulation.


Watch Local

This menu selection opens a window that displays local variables. If this window is already open, selecting **Watch Local** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 28.

🛄 Local function() : LCD_BUSY							
Туре	Attribute	Name	Value	Address			
BIT	BIT	test	0	0x20.1			
int	DATA	i	0x1060	0x6			

Figure 28. Local Function Dialog

The Watch Local menu selection is disabled (greyed out) in the View menu during emulation.

Text File

This menu selection opens a window which displays the contents of a text file.

If a selected text file is already open, selecting **Text File** from the View menu will cause the window containing the text file to appear at the top-most level of the debugger frame; otherwise, selecting **Text File** will open a new window. See the example text file in Figure 29.



🧧 main.c	
Goto line	
0_0180 LCD_busy();	
0_0183 LCD_E = LCD_RW = 0;	// E=0, RW=0
0_0187 LCD_RS = TYPE;	<pre>// RS=INSTRUCTION(0), CHARAC</pre>
127	
0_018B R0_port = uch >> 4;	// output data High
0_{0192} LCD_E = 1;	
0_0194 LCD_E = 0;	
0.0196 B0 port - uch.	// output data Tow
0_{0199} KO_DOLC = dell,	// output data now
0 0198 LCD E = 0;	
0 019D LCD RS = 0:	
0 019F }	
137	
138 void LCD_busy()	
139 {	
140 bit test;	
141 int i;	
142	
0 010F LCD E = LCD RS = 0;	// E=0, RS=0
$\begin{array}{ccc} 0 & 0113 \\ 0 & 0113 \\ \end{array} RO & 10 \\ 0 & 0110 \\ \end{array}$	// R03~00 : change to input
0_0116 DCD_RW = 1;	// Rw=1
0_0110 IOP(1=0; 1(1000; 1++) {	// 8-1
0.011D test = $R03$:	// Get busy flag
0 0121 LCD E = 0:	// E=0
0 0123 LCD E = 1;	// E=1 : Skip AC3
0 0125 LCD E = 0;	// E=0 +
- m	F .:

Figure 29. A Sample Text File

The Text File menu selection is disabled (greyed out) in the View menu during emulation.



Window Menu

The Window menu, shown in Figure 30, can be used to modify the arrangement of child windows or to directly select a child window.

Win	dow
	Cascade
	Tile
	Close
	1 Basic registers
	2 CODE : Bank_0 0000 : Page CS 0200
	3 XDATA : Bank_0 0000 : Page CS 0096
	4 IRAM : 000
	5 STARTUP.A51
	6 Code disassembler
	7 main.c
•	8 SFR : 080

Figure 30. The OCD's Window Menu



Cascade

This menu selection arranges opened child windows in a stepped visual sequence, as shown in Figure 31.

*
<u>^</u>

Figure 31. Cascaded Windows



33

Tile

This menu selection arranges opened child windows in a partitioned display, as shown in Figure 32.



Figure 32. Tiled Windows

Close

This menu selection closes the top-most child window appearing in the debugger frame.

Windows 1, 2, 3, Etc.

This menu selection assigns a sequential number (e.g., 1, 2, 3...) to each child window in the order in which it is opened. Users can directly select any open child window by its number. In Figure 30 on page 31, for example, selecting **6** from the **Window** menu will display the Code Disassembler window as the top-most window in the Debugger screen.



Child Windows

Child windows are secondary windows that are displayed within the main OCD window.

Z8051 Basic Registers Window

The Z8051 Basic Registers window allows users to edit the contents of the Z8051 Series registers. Figure 33 shows an example Z8051 Basic Registers window.

🕘 Basic regist	ers							• X
	R0	R1	R2	R3	R4	R5	R6	R7
Reg #0	12	34	FB	E3	B7	4A	42	00
Reg #1	18	92	F4	62	10	04	AE	01
Reg #2	4D	95	8B	E8	EB	66	ED	A9
🗌 Reg #3	E1	83	BA	08	FF	30	FD	42
PSW	CY 0	AC 0	F0 0	RS1	RS0 0	0V 0	F1 0	PA 0
ACC B	00	D	PTR	0	000			
SP	07			Mod	lify		Cance	el

Figure 33. Using the Basic Registers Function, #1 of 6

Edit

The **Modify** button is disabled (greyed out) by default. Changing the value of a register enables the **Modify** button. New register values are downloaded to the target MCU upon clicking the **Modify** button.

In Figure 34, the current register bank is highlighted in the red area. Users can change register banks by selecting or deselecting any of the registers in this current register bank.



📇 Basic regist	ers							•	×
$\square \land$	R0	R1	R2	R3	R4	R5	R6	R7	
Reg #0	12	34	FB	E3	B7	4A	42	00	
🗌 Reg #1	18	92	F4	62	10	04	AE	01	
🗌 Reg #2	4D	95	8B	E8	EB	66	ED	A9	
🗌 Reg #3	E1	83	BA	08	FF	30	FD	42	
	CY	AC	F0	RS1	RS0	ov	F1	PA	
PSW	0	0	0	0	0	0	0	0	
ACC	00	D	PTR	0	000				
SP	07			Mod	lify		Cano	el	

Figure 34. Using the Basic Registers Function, #2 of 6

In Figure 35, the R0–R7 registers are highlighted in the red area. These registers map to the same area as IRAM addresses in the range 00h–1Fh. Users can change these values by entering 8-bit hexadecimal formats.

📇 Basic regist	ters						• 💌
	R0 R1	L R2	R3	R4	R5	R6	R7
🔽 Reg #0	12 34	FB	E3	B7	4 A	42	00
🗌 Reg #1	18 92	F4	62	10	04	AE	01
🗌 Reg #2	4D 95	8B	E8	EB	66	ED	A9
🗌 Reg #3	E1 83	BA	08	FF	30	FD	42
PSW	CY AC	F0	RS1 0	RS0 0	0V 0	F1 0	PA 0
ACC B	00	DPTR	Mod	000	1	Cance	4
SP	107	_	1,50	/			

Figure 35. Using the Basic Registers Function, #3 of 6



In Figure 36, the red area highlights the Program Status Word (PSW), in which bit units can be changed.

😬 Basic regist	ers							•	<
	R0	R1	R2	R3	R4	R5	R6	R7	
Reg #0	12	34	FB	E3	B7	4A	42	00	
🗌 Reg #1	18	92	F4	62	10	04	AE	01	
🗌 Reg #2	4D	95	8B	E8	EB	66	ED	A9	
🗌 Reg #3	E1	83	BA	08	FF	30	FD	42	
	CY	AC	F0	RS1	RS0	ov	F1	PA	٦
PSW	0	0	0	0	0	0	0	0	J
ACC	00	D	PTR	0	000				
В	00								
SP	07			Mod	lify		Cance	el	

Figure 36. Using the Basic Registers Function, #4 of 6

In Figure 37, the red area highlights the Accumulator (ACC), the B Register (B) and the Stack Pointer (SP) registers. Enter a number in n 8-bit hexadecimal format to change any of these values.

🖶 Basic regist	ers							• 💌	
	R0	R1	R2	R3	R4	R5	R6	R7	
🔽 Reg #0	12	34	FB	E3	B7	4A	42	00	
🗌 Reg #1	18	92	F4	62	10	04	AE	01	
🗌 Reg #2	4D	95	8B	E8	EB	66	ED	A9	
🗌 Reg #3	E1	83	BA	08	FF	30	FD	42	
	CY	AC	F0	RS1	RS0	ov	F1	PA	
PSW	0	0	0	0	0	0	0	0	
ACC B	00		PTR	0	000				
SP	07	J		Mod	lify		Cance	el	

Figure 37. Using the Basic Registers Function, #5 of 6



In Figure 38, the red area highlights the DPTR Register which displays, and can be edited by, entering numbers in the 16-bit hexadecimal format. If the target MCU features more than two DPTRs, the DPTR field in this dialog shows the currently selected register. If each DPTR resides at a different address, Zilog recommends using the SFR window instead.

🖶 Basic regist	iers 🗖 🗖 💌
	R0 R1 R2 R3 R4 R5 R6 R7
Reg #0	12 34 FB E3 B7 4A 42 00
Reg #1	18 92 F4 62 10 04 AE 01
🗌 Reg #2	4D 95 8B E8 EB 66 ED A9
🗌 Reg #3	E1 83 BA 08 FF 30 FD 42
PSW	CY AC F0 RS1 RS0 OV F1 PA 0 0 0 0 0 0 0 0 0
ACC B SP	O0 DPTR O000 00 00 00 07 Modify Cancel

Figure 38. Using the Basic Registers Function, #6 of 6

Code Disassemble Window

The Code Disassemble window displays the contents of code memory by using a disassemble format. All operand values must be entered in hexadecimal format. Figure 39 shows an example Code Disassembler window.



🗁 Code	disassembl	er		×
Bank #	00 👻	Goto	PC 00000 Change	
0_0009	8B08	MOV	008,R3	
0_000E	8A09	MOV	009,R2	
0_000D	890A	MOV	00A,R1	
0_000F	EF	MOV	A, R7	
0_0010	14	DEC	A	
0_0011	600F	JZ	00022	
0_0013	14	DEC	A	
0_0014	6011	JZ	00027	
0_0016	14	DEC	A	
0_0017	6013	JZ	0002C	
0_0019	2403	ADD	A,#003	
0_001E	7012	JNZ	0002F	
0_001D	750D80	MOV	00D,#080	
0_0020	800D	SJMP	0002F	
0_0022	750DC0	MOV	00D,#0C0	Ŧ
◀ 📃			•	

Figure 39. Using the Code Disassembler Function, #1 of 3

If map/symbol files are already loaded, the affected source lines are highlighted by boxes, as shown in Figure 40. Double-click any of these highlighted boxes to open its source file and move to the appropriate address line.

🗁 Code d	lisassemble	er				ĸ
Bank # 00	•	Goto	PC	00000	Change	
0 0009	8B08	MOV	008,R	:3		
0_000B	8A09	MOV	009,R	2		
0_000D	890A	MOV	00A, R	1		
0 000F	EF	MOV	A, R7			
0_0010	14	DEC	A			
0_0011	600F	JZ	00022			
0_0013	14	DEC	A			
0_0014	6011	JZ	00027			
0_0016	14	DEC	A			
0_0017	6013	JZ	0002C			
0_0019	2403	ADD	A,#00	3		
0_001B	7012	JNZ	0002F	•		
0_001D	750D80	MOV	00D,#	080		
0_0020	800D	SJMP	0002F	•		
0 0022	750DC0	MOV	00D,#	0C0		Ŧ
•	11				Þ	

Figure 40. Using the Code Disassembler Function, #2 of 3



Line Assemble

The Line Assemble function supports a line assembly function in which users can change the code space with assembly language. This function can directly change the target MCU code space, but it does not change the source program file.

With your mouse, move the cursor to a line that you wish to change, and right-click to open an edit field for the contents of that line, as shown in Figure 41. Change the contents of the line by entering an instruction, operand, etc., in hexadecimal format.

🗁 Code d	lisassemble	er	- • •	
Bank # 00		Goto	PC 00000 Change	
0 0009	8B08	MOV	008,R3	
0_000B	8A09	MOV	009,R2	a.
0_000D	890A	MOV	00A,R1	-
0 000F	EF	MOV	A, R7	
0_0010	14	DEC	A	
0_0011	600F	JZ	00022	
0_0013	14	DEC	A	
0_0014	6011	JZ	00027	
0_0016	14	DEC	A	
0_0017	6013	JZ	0002C	
0_0019	2403	ADD	A,#003	
0_001B	7012	117	00025	
0 001D	MOV 000),#080	Modify	
0_0020	0000	SJME	0002r	
0_0022	750DC0	MOV	00D,#0C0 -	r
•				

Figure 41. Using the Code Assembler Function, #3 of 3

PC Break Toggle

The target MCU's internal Program Counter (PC), sets or clears all PC breakpoint settings. The PC breakpoint count differs in each device in the Z8051 Series; normally, eight breakpoints can be set. In Figure 42, the red line represents a program counter breakpoint in the line, and the blue line represents the current program counter.



🗁 Code disassemb	ler				x
Bank # 00 💌	Goto	PC (0016F	Change	
0_016E 07	INC	@R1			
0_016F EB	MOV	A, R3			
0_0170 1B	DEC	R3			
0_0171 70F8	JNZ	0016B			
0_0173 1A	DEC	R2			
0_0174 80F5	SJMP	0016B			
0 0176 ED	MOV	A, R5			
0_0177 1D	DEC	R5			
0_0178 7001	JNZ	0017B			
0_017A 1C	DEC	R4			
0_017B ED	MOV	A, R5			
0_017C 4C	ORL	A, R4			
0_017D 70E8	JNZ	00167			
0 017F 22	RET				
0 0180 12010E	' LCALL	0010F			Ŧ
< <u> </u>				•	

Figure 42. Using the PC Break Toggle Function

To set or clear a PC breakpoint, set your cursor on a selected line and double-click the mouse's left button.

Code Dump Window

Code dump windows display each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. Each 256-byte page resides at an address in the range xx00-xxFFh, in which xx is the number of the page.

The upper side of the Code Dump window displays the address of the current cursor position and the checksum of the current page. The current page number is displayed as a watermark in the center of this window. In Figure 43, for example, the page number is 00.



🗀 COD	E : B	ank_	0 00)5A	: Pag	je CS 5D5F										•	x
Bank #	00	•		Goto)		Patte	rn _	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0_0000	02	01	ΒA	7B	FF	E 4	FD	7 F	01	8B	80	8A	09	89	0A	EF	
0_0010	14	60	0 F	14	60	11	14	60	13	24	03	70	12	75	0 D	80	
0_0020	80	0 D	75	0 D	C0	80	80	75	0 D	94	80	03	75	0 D	D4	ED	
0_0030	25	0 D	F5	0 D	C2	00	F5	0 E	12	01	80	E 4	F5	0B	AB	80	
0_0040	AA	09	Α9	0 A 0	85	0B	82	75	83	00	12	00	E2	F5	0C	60	
0_0050	11	D2	00	85	0C	0E	12	01	80	05	0B	E5	0B	C3	94	40	
0_0060	40	DC	22	C2	88	C2	89	C2	8 A	12	01	5 F	7 F	20	12	01	
0_0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
0_0080	12	01	Α9	12	01	Α0	7 F	10	12	01	Α9	7 F	00	7E	10	12	
0_0090	01	63	E 4	12	01	Α9	7 F	60	12	01	Α9	C2	00	75	0E	01	
0_00A0	12	01	80	C2	00	75	0E	28	02	01	80	75	Α0	FF	75	98	
0_00B0	FF	12	00	63	7B	FF	7A	01	79	3B	E 4	FD	FF	12	00	09	
0_00C0	7A	01	79	44	12	00	03	12	01	5 F	7B	FF	7A	01	79	4 D	
0_00D0	E 4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5 F	
0_00E0	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	3A	F5	83	Ε0	
0_00F0	22	50	06	E 9	25	82	F8	E 6	22	BB	FE	06	E 9	25	82	F8	

Figure 43. Using the Code Dump Function, #1 of 2

Edit

Users can change data values in the Code Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 44.



🗀 COD	E : B	ank_	0 00	035	: Pag	je C	je CS 5D71										x
Bank #	00	•		Got	D		Patte	rn _	Lo	ad		Save	9		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0_0000	02	01	ΒA	7B	FF	E 4	FD	7 F	01	8B	80	8A	09	89	0 A 0	EF	
0_0010	12	34	32	51	20	11	14	60	14	12	34	70	12	75	0 D	80	
0_0020	80	00	75	0D	<u>C0</u>	-80	08	75	0 D	94	80	03	75	0 D	D4	ΕD	
0_0030	25	AB	39	13	44	d٥	F5	0 E	12	01	80	E 4	F5	0B	AB	08	
0_0040	AA`	09	Α9	0A	85	0B	82	75	83	00	12	00	E2	F5	0C	60	
0_0050	11	D2	00	85	00	0 E	12	01	80	05	0B	E5	0B	C3	94	40	
0_0060	40	DC	22	C2	88	C2	89	C2	8 A	12	01	5 F	7 F	20	12	01	
0_0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
0_0080	12	01	Α9	12	01	AO	7 F	10	12	01	Α9	7 F	00	7E	10	12	
0_0090	01	63	E 4	12	01	Α9	7 F	60	12	01	Α9	C2	00	75	0 E	01	
0_00A0	12	01	80	C2	00	75	0 E	28	02	01	80	75	Α0	FF	75	98	
0_00B0	FF	12	00	63	7B	FF	7A	01	79	3B	E 4	FD	FF	12	00	09	
0_00C0	7A	01	79	44	12	0.0	03	12	01	5 F	7B	FF	7A	01	79	4 D	
0_00D0	E 4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5 F	
0_00E0	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	3A	F5	83	Ε0	
0_00F0	22	50	06	E 9	25	82	F8	E 6	22	BB	FE	06	E 9	25	82	F8	

Figure 44. Using the Code Dump Function, #2 of 2

If you wish to cancel your inputs, press the Escape (Esc) key. Press the Enter key to save your changes, and note that the red color of your changed character pair has changed back to black.

Bank

The devices in the Z8051 Series use a linear addressing method, and display page units in the 64KB range. To overcome this 64KB limit, the user can employ banked addresses, in which a bank is the upper 4 bits of a 20-bit address.

Goto

Click the **Goto** button to view memory locations in any 16-bit segments within the 0000h-FFFFh address range in the Code Dump window or edit these memory locations by entering an address in hexadecimal format. See the example Input dialog in Figure 45.



put dialog box		
<u>A</u> ddress 0000 [•] FFFF	0300	
ок		
Cancel		

Figure 45. The Code Dump Function's Goto/Input Dialog

Load

Click the **Load** button to display the Pattern Load dialog, in which you can load a data pattern or code hex file to the code space; see Figure 46.

OK
Cancel
OG Browse

Figure 46. The Code Dump Function's Pattern Load Dialog

Alternatively, users can download code by choosing **Load Hex** from the **File** menu. However, this method is used to load user-specified data patterns only; it does not clear the entire code space. A data pattern can be either a small code segment or complete program code.

Save

Click the **Save** button to display the Pattern Save dialog, in which you can save a code space as a pattern file; see the example in Figure 47.



Code bank #	00 💌	OK
Start address	0300	Cancel
End address	03FF	
File name		Browse

Figure 47. The Code Dump Function's Pattern Save Dialog

Fill

Click the **Fill** button to display the Pattern Fill dialog, in which you can write a common value in all code memory spaces in a specified address range; see the example in Figure 48.

attern Fill di	alog	>
Code bank #	00 💌	OK
Start address	0300	Cancel
End address	03FF	-
Hexa value	00	

Figure 48. The Code Dump Function's Pattern Fill Dialog

XDATA Dump Window

The XDATA Dump window displays each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. Each 256-byte page resides at an address in the range xx00-xxFFh, in which xx is the number of the page.

The upper side of the XDATA Dump window displays the address of the current cursor position and the checksum of the current page. The current page number is displayed as a watermark in the center of this window. In Figure 49, for example, the page number is 00.



	TA :	Bank	<u>_</u> 0 0	OCE	: Pa	ige (CS 5	BD5								•	x
Bank #	00	•		Goto)	I	Patte	rn _	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Е	F	
0_0000	02	11	FA	02	12	39	FD	7 F	01	8B	80	8A	09	89	0A	EF	
0_0010	14	60	0 F	02	12	3A	14	60	13	24	03	70	12	75	0 D	80	
0_0020	80	0 D	75	02	12	33	08	75	0D	94	80	03	75	0D	D4	ED	
0_0030	25	0 D	F5	02	12	36	F5	0 E	12	01	80	02	12	3B	AB	80	
0_0040	AA	09	Α9	02	0 F	5E	82	75	83	00	12	02	12	3C	0C	60	
0_0050	11	D2	00	85	00	0 E	12	01	80	05	0B	E 5	0B	C3	94	40	
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5 F	7 F	20	12	01	
0_0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
0_0080	21	BD	41	06	41	78	41	CO	61	0C	61	54	01	EΒ	01	E4	
0_0090	02	63	01	96	01	C1	12	01	35	12	01	18	70	22	75	84	
0_00A0	00	75	85	02	74	00	85	C6	Β6	05	B6	F0	D5	Β6	FC	D2	
0_00B0	C1	20	C1	FD	12	03	7A	12	01	5E	75	84	00	75	85	00	
0_00C0	22	75	84	00	75	85	02	90	80	00	43	C1	01	75	B 6	40	
0_00D0	74	00	F0	D5	Β6	FC	D2	C0	20	C0	FD	63	C1	01	75	85	
0_00E0	00	74	00	22	63	D9	10	D2	C4	80	FΕ	75	81	BF	75	A 8	
0_00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	A0	80	

Figure 49. Using the XDATA Dump Function, #1 of 2

Edit

Users can change data values in the Code Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 50.



	TA :	Bank	_0 C	028	: Pa	ige (CS 5	BD5								•	×
Bank #	00	•		Goto)		Patte	rn _	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0_0000	02	11	FA	02	12	39	FD	7 F	01	8B	80	8A	09	89	ΟA	EF	
0_0010	14	60	0 F	02	12-	3A	14	60	-1,3	24	03	70	12	75	0D	80	
0_0020	80	0D	75	02	12	98	13	24	0D	94	80	03	75	0D	D4	ΕD	
0_0030	25	0 D	F5	02	12-	36	F5	0E	-12	01	80	02	12	3B	AB	08	
0_0040	AA	09	Α9	02	0 F	5E	82	75	83	00	12	02	12	3C	0C	60	
0_0050	11	D2	00	85	00	0 E	12	01	80	05	0B	E5	0B	C3	94	40	
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5 F	7 F	20	12	01	
0_0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
0_0080	21	BD	41	06	41	78	41	C0	61	0C	61	54	01	EΒ	01	E 4	
0_0090	02	63	01	96	01	C1	12	01	35	12	01	18	70	22	75	84	
0_00A0	00	75	85	02	74	00	85	C6	B6	05	B6	FO	D5	B6	FC	D2	
0_00B0	C1	20	C1	FD	12	03	7A	12	01	5E	75	84	00	75	85	00	
0_00C0	22	75	84	00	75	85	02	90	80	00	43	C1	01	75	Β6	40	
0_00D0	74	00	F0	D5	B6	FC	D2	C0	20	C0	FD	63	C1	01	75	85	
0_00E0	00	74	00	22	63	D9	10	D2	C4	80	FΕ	75	81	BF	75	A8	
0_00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	A0	80	

Figure 50. Using the XDATA Dump Function, #2 of 2

Bank

The devices in the Z8051 Series use a linear addressing method, and display page units in the 64KB range. To overcome this 64KB limit, the user can employ banked addresses, in which a bank is the upper 4 bits of a 20-bit address.

Goto

Click the **Goto** button to view memory locations in any 16-bit segments within the 0000h-FFFFh address range in the XDATA Dump window or edit these memory locations by entering an address in hexadecimal format. See the example in Figure 51.

put dialog box		
Address	0300	
ОК		
Cancel	J	

Figure 51. The XDATA Dump Function's Goto/Input Dialog



Load

Click the **Load** button to display the Pattern Load dialog, in which you can load a data pattern or code hex file to the XDATA area. However, this command does not clear the XDATA area; see Figure 52.



Figure 52. The XDATA Dump Function's Pattern Load Dialog

Save

Click the **Save** button to display the Pattern Save dialog, in which you can save the XDATA area as a pattern file; see Figure 53.

Pattern save	dialog	×
XRAM bank #	00 💌	OK
Start address	0000	Cancel
End address	OOFF	
File name		Browse
I		

Figure 53. The XDATA Dump Function's Pattern Save Dialog

Fill

Click the **Fill** button to display the Pattern Fill dialog, in which you can write a common value in all XDATA memory spaces in a specified address range; see the example in Figure 54.



XRAM bank #	00 💌	OK
Start address	0000	Cancel
End address	OOFF	-
Hexa value	00	

Figure 54. The XDATA Dump Function's Pattern Fill Dialog

IRAM Dump Window

The IRAM Dump window displays each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. Each 256-byte page resides at an address in the range xx00-xxFFh, in which xx is the number of the page.

The upper side of the IRAM Dump window displays the address of the current cursor position and the checksum of the current page. A watermark, displayed as *iR*, appears in the center of this window, as shown in Figure 55.

Figure 55 also shows IRAM addresses in the range 00h-7Fh, which represent the direct area; the characters representing these addresses are colored black. IRAM addresses in the range 80h-FFh represent the indirect area; these characters are colored cyan.



🕐 IRAN	1:0	B9														•	x
Pattern		Save	•														
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0000	E8	8B	FB	E 3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11	
0010	4 D	95	8B	ΕA	EΒ	66	ED	Α9	F1	83	BB	80	FF	3A	7 D	62	
0020	DF	22	2C	09	34	80	5B	20	45	81	EΕ	FA	81	0E	D5	60	
0030	55	53	C1	80	73	98	6A	10	AB	D4	40	66	68	49	B8	AA	
0040	D9	Α2	Β9	99	CO	CO	D9	21	40	64	79	F6	88	66	B5	43	
0050	22	68	AA	D4	62	12	63	0A	F5	11	7C	73	04	7C	C2	0 E	
0060	CO	C 9	Α3	3B	F4	С3	CF	00	вс	0 E	D5	C3	0C	2C	11	80	
0070	DO	Ε3	62	ЗA	AD	56	D2	E 9	94	80	FC	37	E 9	EΒ	89	ΕE	
0080	00	C2	80	В8	6E	FB	59	A3	51	12	Α3	В3	92	FB	DC	AE	
0090	4B	ΕA	04	8 F	71	7B	37	6D	1 F	E6	Β4	26	E6	41	98	F1	
00A0	DD	7 F	ВD	8D	B8	ΕA	2 F	EF	BC	1D	DD	E5	5D	В0	D1	ΕE	
00B0	48	6D	2 D	85	4 D	C7	55	5E	2B	ЗF	76	6D	7E	DD	77	BF	
00C0	2 D	06	26	73	AA	D9	F4	BE	82	91	32	4B	F8	B6	1E	E5	
00D0	7 F	C1	E5	C4	BF	B8	Β6	AC	7 F	63	2E	A0	Β7	2 D	FD	DA	
00E0	E5	ΒE	C5	В0	FF	FC	CF	D4	43	98	27	76	48	7 F	3E	В5	
00F0	EF	1F	B2	70	BB	BE	B9	BF	E1	8 F	E1	79	77	56	CD	ED	

Figure 55. Using the IRAM Dump Function, #1 of 2

To learn more about direct and indirect memory areas, please refer to the product specification for your particular Z8051 device.

Edit

Users can change data values in the IRAM Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 56.



$\langle \hat{U} \rangle$	IRAN	1:0	9D														•	x
1	Pattern		Save															
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0	000	E8	8B	FB	E 3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11	
0	010	4 D	95	8B	ΕA	EB	66	ЕÐ	<u>A9</u>	F1	-8,3	BB	80	FF	3A	7 D	62	
0	020	DF	22	2C	09	34	80	518	12	34	8þ	ΕE	FA	81	0 E	D5	60	
0	030	55	53	C1	80	73	98	6A	10	ÀΒ	Ð4	40	66	68	49	B8	AA	
0	040	D9	Α2	В9	99	CO	CO	D9	21	40	64	79	F6	88	66	В5	43	
0	050	22	68	AA	D4	62	12	63	0A	F5	11	7C	73	04	7C	C2	0E	
0	060	C0	C 9	Α3	3B	F4	С3	CF	00	вс	0 E	D5	С3	0C	2C	11	80	
0	070	DO	E3	62	ЗA	AD	56	D2	E 9	94	80	FC	37	E 9	EB	89	ΕE	
0	080	00	C2	80	В8	6E	FB	59	A3	51	12	A3	B3	92	-FB	DC	AE	
0	090	4B	ΕA	04	8 F	71	7B	37	6D	1	98	76	BE	DA	41	98	F1	
0	0A0	DD	7 F	BD	8D	B8	EA	2 F	EF	BC	1D	DD	E5	50	-180	D1	EE	
0	0B0	48	6D	2 D	85	4 D	C7	55	5E	2B	3F	76	6D	7 E	DD	77	BF	
0	0C0	2 D	06	26	73	AA	D9	F4	BE	82	91	32	4 B	F8	B6	1E	E5	
0	OD0	7 F	C1	E5	C4	BF	B8	B6	AC	7 F	63	2E	A0	В7	2 D	FD	DA	
0	OE0	E5	BE	C5	в0	FF	FC	CF	D4	43	98	27	76	48	7 F	3E	B5	
0	010	ΕF	1 F	В2	70	вВ	BE	В9	BF	E1	8 F	Ε1	79	17	56	CD	ED	

Figure 56. Using the IRAM Dump Function, #2 of 2

If you wish to cancel your inputs, press the Escape (Esc) key. Press the Enter key to save your changes, and note that the red color of your changed character pair has changed back to black.

Save

Click the **Save** button to save the IRAM area as a pattern file.

SFR Dump Window

The Special Function Register (SFR) Dump window displays each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. The upper side of the SFR Dump window displays the address of the current cursor position and the check-sum of the current page.

Figure 57 shows SFR addresses in the range 80h-FFh, which represent the direct area of IRAM.



🕺 SFR :	: 0CI	D															×
Pattern	R	efres	sh	S	ave												
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0080	29	22	4 C	01	00	00	D9	00	00	00	04	87	76	00	00	DF	
0090	05	00	00	00	00	00	00	00	FF	00	8 F	01	FO	00	00	00	
00A0	7 F	C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00B0	00	00	00	00	00	00	00	00	00	00	7 F	7 F	00	00	00	00	
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00D0	00	00	00	00	00	00	42	00	00	00	00	00	3F	3F	01	FF	
00E0	EF	00	00	00	00	80	FF	00	00	00	00	03	80	00	00	00	
00F0	00	00	00	00	00	00	00	00	00	00	03	C0	60	08	00	00	

Figure 57. Using the SFR Dump Function, #1 of 3

The special function registers differ in each Z8051 Series device. To learn more about special function registers, please refer to the product specification for your particular Z8051 device.

Edit

Users can change data values in the SFR Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 58.

1	🛠 SFR	: 0A 6	5														• (x
	Pattern	1 <u>R</u>	efres	sh	S	ave												
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
Γ	0800	29	22	4 C	01	00	00	D9	00	00	00	04	87	76	00	00	DF	
l	0090	05	00	0,0	00	00	00	0,0	00	FF	00	8 F	01	F0	00	00	00	
L	00A0	7 F	C0	00	97	FE	D5	00	00	00	00	00	00	00	00	00	00	
1	00B0	00	00	00	00	00	00	00	00	00	00	7 F	7 F	00	00	00	00	
L	00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
L	00D0	00	00	00	00	00	00	42	00	00	00	00	00	3F	3F	01	FF	
L	00E0	EF	00	00	00	00	80	FF	00	00	00	00	03	80	00	00	00	
	00F0	00	00	00	00	00	00	00	00	00	00	03	C0	60	80	00	00	

Figure 58. Using the SFR Dump Function, #2 of 3



If you wish to cancel your inputs, press the Escape (Esc) key. Press the Enter key to save your changes, and note that the red color of your changed character pair has changed back to black.

Refresh

The SFR area includes static registers such as a stack pointer, an accumulator, etc. However, most SFRs are dynamic registers such as timers, I/Os, etc. Clicking the **Refresh** button (highlighted in Figure 59) redisplays all current data.

🛠 SFR	: 001	D		_													x
Pattern	• <u>R</u>	efres	sh	S	ave												
(0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Е	F	
0080	29	22	4 C	01	00	00	D9	00	00	00	04	87	76	00	00	DF	
0090	05	00	00	00	00	00	00	00	FF	00	8 F	01	F0	00	00	00	
00A0	7 F	C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00B0	00	00	00	00	00	00	00	00	00	00	7 F	7 F	00	00	00	00	
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00D0	00	00	00	00	00	00	42	00	00	00	00	00	3F	3F	01	FF	
00E0	EF	00	00	00	00	80	FF	00	00	00	00	03	80	00	00	00	
00F0	00	00	00	00	00	00	00	00	00	00	03	C0	60	08	00	00	

Figure 59. Using the SFRDump Function, #3 of 3

Save

Clicking the **Save** button saves an SFR area as a pattern file.

Watch Global Window

The Watch Global window displays and supports the modification of global variables within the user's C language-based source program. Each variable is located within the Code, XDATA, IRAM, SFR dump spaces. If users could easily determine a variable's location, they could edit the variable directly. However, finding a global variable across these many memory dump spaces is often perceived to be a tedious process.

The Watch Global window alleviates this problem by employing a map/symbol file; see Figure 60.



				. • 💌
Remove s	symbol			
Attribute	Name	Value	Address	
BIT	LCD_E	0	0x88.2	
BIT	LCD_RS	1	0x88.0	
BIT	LCD_RW	1	0x88.1	
DATA	R1_io	0x0	0xA0	
DATA	R0_io	0x0	0x98	
DATA	R0_port	0x4	0x80	
BIT	R03	0	0x80.3	
	Attribute BIT BIT BIT DATA DATA DATA BIT	Remove symbol Attribute Name BIT LCD_E BIT LCD_RS BIT LCD_RW DATA R1_io DATA R0_io DATA R0_port BIT R03	Remove symbol Attribute Name Value BIT LCD_E 0 BIT LCD_RS 1 BIT LCD_RW 1 DATA R1_io 0x0 DATA R0_io 0x0 DATA R0_port 0x4 BIT R03 0	Remove symbol Attribute Name Value Address BIT LCD_E 0 0x88.2 BIT LCD_RS 1 0x88.0 BIT LCD_RW 1 0x88.1 DATA R1_io 0x0 0xA0 DATA R0_io 0x0 0x98 BIT R03 0 0x80.3

Figure 60. The Watch Global Function's Global Variables Dialog

Add Symbol

Clicking the **Add Symbol** button displays the Global Symbol Add/Remove dialog, in which you can add a global variable to the Watch Global display list, shown in Figure 61.

Global symbol add/remove o	lialog 📃
R0 port	ADD
LCD_E LCD_RS LCD_RW R0_jo R0_port R03 R1_jo	Cancel

Figure 61. Adding A Global Symbol

Remove Symbol

Clicking the **Remove Symbol** button removes a global variable from the Watch Global display list.



Edit

Users can change data values in the Watch Global window at any time, except during emulation. This editing method is quite simple; just place the cursor where you wish to make an edit, and double-click the left button on your mouse to display a pop-up dialog in which you can change the data and click the **Modify** pop-up button to incorporate the change, as shown in Figure 62.

🖳 Global variables				- • •
Add symbol	Remove	symbol		
Туре	Attribute	Name	Value	Address
BIT	BIT	LCD_E	0	0x88.2
BIT	BIT	LCD_RS	1	0x88.0
BIT	BIT	LCD_RW	1	0x88.1
unsigned char	DATA	R1_io	0x0	0xA0
unsigned char	DATA	R0_io	0x0	0x98
unsigned char	DATA	R0_port	0x4	Modify
BIT	BIT	R03	0	0X80.3
ļ				

Figure 62. Editing A Global Symbol

Watch Local Window

The Watch Global window displays and supports the modification of local variables within the user's C language-based source program. Each variable is located within the Code, XDATA, IRAM, SFR dump spaces.

Much like the issue with finding global variables, users could edit these local variables directly if finding them was not so tedious. The Watch Local window, shown in Figure 63, alleviates this problem by employing a map/symbol file, as described in the previous section.



U Local function() : LCD_OUT										
Туре	Attribute	Name	Value	Address						
BIT	BIT	TYPE	0	0x20.0						
unsigned char	DATA	uch	0x1	OxE						
•				•						

Figure 63. The Watch Local Function Dialog

Edit

Users can change data values in the Watch Local window at any time, except during emulation. This editing method is quite simple; just place the cursor where you wish to make an edit, and double-click the left button on your mouse to display a pop-up dialog in which you can change the data and click the **Modify** pop-up button to incorporate the change, as shown in Figure 64.

Local function() :	LCD_OUT			
Туре	Attribute	Name	Value	Address
BIT	BIT	TYPE	0	0x20.0
unsigned char	DATA	uch	0x1	Modify
•				•

Figure 64. Editing A Local Symbol

Add or Remove Symbol

Locals variables are dynamic; therefore, adding or removing a symbol will depend on each program module.

In the Debugger, the user can check the current C module and find its local variables automatically so that the user is not required to add or remove the symbol.



Figure 65 shows an example C source program module. The current program counter is located in the delay(UINT uCnt) function module (highlighted in the upper half of the figure), and the Local Variable window displays the name of the module and its variable (highlighted in the lower half of the figure).

🧧 main.c				
Goto line				
70 71 void de 72 { 73 UIN 74	lay(UINT uCnt T tmp, uu;)		*
0_0163 for 0_0167 0_0168 0_0176 } 0_0176 } 0_017F } 80	<pre>(tmp=10; tmp! uu = uCnt; while(uu !=</pre>	=0; tmp) { 0) uu;		
81 // 82 // LCD 83 // 84 void LC 85 { 0_0063 LCD 0_0069 del	Function D_init() _E = LCD_RW = ay(0xfff);	LCD_RS = 0;		// E=0 // min _
4	m			¥ .,
U Local function	0 :_DELAY	1		
Туре	Attribute	Name	Value	Address
unsigned int unsigned int unsigned int	DATA DATA DATA	uCnt tmp uu	OxFFFF OxO OxO	0x6 0x4 0x2
		III		

Figure 65. Example Watch Local Function, #1 of 2



If program flow is changed to another module, then the Local Variable list will be changed, as shown in Figure 66.

📕 main.c			[
Goto line				
116 0_01AF R0_pc 0_01B5 LCD_F 0_01B7 LCD_F 0_01B9 }	<pre>prt = uch >> E = 1; E = 0;</pre>	4;	_	// out *
121 122 void LCD	out (bit TYP	E, UCHAR uch)		
0_0180 LCD_1	ousy();			-
0_0183 LCD H 0_0187 LCD_H	E = LCD RW = RS = TYPE;	0;		// E=0 // RS=
127 0_018B R0_pc 0_0192 LCD_F 0_0194 LCD_F	<pre>prt = uch >> E = 1; E = 0;</pre>	4;		// out E
131 0_0196 R0_pc 0_0199 LCD_H	prt = uch; E = 1;			// out
A TOP F	m.			K
Local function()	LCD_OUT		1	
Туре	Attribute	Name	Value	Address
BIT unsigned char	BIT DATA	TYPE uch	0 0x1	0x20.0 0xE
2				

Figure 66. Example Watch Local Function, #2 of 2

Text File Window

The Text File window displays text files, but does not support the editing of text files. If you have loaded a map/symbol file, the source program file will display an actual hard-ware address in the line number area. To provide a visual understanding of this displayed data, the following two examples offer a comparison.

Example 1. If a map/symbol file has not been loaded, or if the file does not include symbol information, only the line number is displayed, as highlighted in Figure 67.



main.c		
Goto lin	e	
84 V	pid LCD_init()	
85 1		
86	$LCD_E = LCD_RW = LCD_RS = 0;$	// E=0
87	delay(0xffff);	// min
88		
89	<pre>// Function set (4bit long)</pre>	
90	LCD_out_Upper((UCHAR)0x20);	// 0
91	LCD_out_Upper((UCHAR)0x20);	// 0
92	LCD_out_Upper((UCHAR)0x80);	// N
93	delay(0x10);	// min
94		
95	// Display on / off	
96	LCD_out_Upper((UCHAR)0x00);	// 0
97	LCD_out_Upper((UCHAR)0xc0);	// 1
98	delay(0x10);	// min
99		
100	// Display clear	
101	LCD out Upper((UCHAR)0x00);	1/ 0
107	ICD out Honer //HCHADIOv101.	11 0
	iii)	

Figure 67. Using the Text File Function, #1 of 5

Example 2. If a map/symbol file has been loaded and the file includes symbol information, then the line number and address are displayed, as highlighted in Figure 68.

📕 main.c		
Goto lin	ne	
84 V	oid LCD_init()	*
0_0063	<pre>LCD_E = LCD_RW = LCD_RS = 0; delay(0xfff);</pre>	// E=0 // min
89 0_006C 0_0071 0_0076 93 94	<pre>// Function set (4bit long) LCD_out_Upper((UCHAR)0x20); LCD_out_Upper((UCHAR)0x20); LCD_out_Upper((UCHAR)0x80); delay(0x10);</pre>	// 0 // 0 // N // min
95 0_007B 0_007E 98	<pre>// Display on / off LCD_out_Upper((UCHAR)0x00); LCD_out_Upper((UCHAR)0xc0); delay(0x10);</pre>	// 0 // 1 // min
100 0_0083 0_0086	// Display clear LCD_out_Upper((UCHAR)0x00); LCD_out_Upper(/UCHAR)0x10);	// o -

Figure 68. Using the Text File Function, #2 of 5



Goto Line

Clicking the **Goto Line** button displays the Get Decimal Number dialog box, which allows users to jump to another line in a text file; see Figure 69. Map/symbol information is not required.



Figure 69. Using the Text File Function, #3 of 5

Disassemble Window Linkage

If a map/symbol file has been loaded and a text file is displayed (see the example in Figure 70), the text file will show addresses instead of line numbers. In this Text File Child Window, and with your mouse, set your cursor in an address area (the left-most column) and double-click the left button to launch the Code Disassemble dialog, which will highlight the address.



🝠 ma	🚽 main.c						
Got	o line	Find		Up	Down		
0_0206		if(clockState	== CLK_32KH	IZ)			
186		{	-				
187		unsigned	short 1;				
0_0200		switch_to	_orun2();		Vacmo do	low for YTML stabilization	
0 0200		unute(")n	$r \in 32000; I$	8MH7\n\	/SOme de	Tay LOL ATAL SCADITIZACION	
191		i upuco (in	trionming di	. 01112 1111	- 11		
192		, else					
193		{					
0_02E8		uputs ("\n	\rP31 was pr	essed. L	EDs dire	ction is changed\n\r");	
0_02F1		change_le	ds_direction	n();			
196		}					
0_0310	}						
198							
199	/***	*					
200		Function: tim	erU_isr				
201	* /	vescription:	timer u serv	rice inte	rrupt oc	curs on every ims. malicks	13
0 0911	moid	TIMEDO isr/m	oid) interr	mt 7 TO	VECT		
204	1 I	TITERO_ISE(*	ord, meerre	.pc 2_10_	VLCI		_
0 0324	1 3	msTicks++;					
0 0330		ledTicks++;					
207		and a survey of the					
208		//update leds	every 100ms	3			-
1				1			

Figure 70. Text File Child Window

Example. Double-click the left button on your mouse at address 0_02C6. The Code Disassembler child window appears, showing the 0_02C6 location at the top of the dialog; see Figure 71.



🔤 Code disasser	nbler		_ 🗆 ×
Bank #00 💌	Goto	PC 00000 Change	
0_02B3 C006 0_02B5 C005 0_02B7 C004 0_02B9 C003 0_02BB C002 0_02BD C001 0_02BF C000 0_02C1 C0D0 0_02C3 75D000 0_02C3 75D000 0_02C9 E0 0_02C9 E0 0_02C9 E0 0_02C9 FF 0_02C8 701B 0_02C9 T01B 0_02C9 7F7D 0_02D2 7F7D 0_02D4 1E 0_02D5 BEFF01 0_02D8 1F 0_02D8 1F 0_02D9 EE 0_02D8 4F 0_02D9 EE 0_02D8 70F7 0_02D8 70F7 0_02D8 70F7 0_02D8 70F7	PUSH PUSH PUSH PUSH PUSH PUSH PUSH MOV MOV MOV MOV MOV JNZ LCALL MOV DEC CJNE DEC MOV ORL JNZ MOV	006 005 004 003 002 001 000 0D0,#000 DPTR,#00093 A,@DPTR R7,A 002E8 00B8C R6,#000 R7,#07D R6 R6,#0FF,002D9 R7 A,R6 A,R7 002D4 DPTR,#016A7	
8			- 11

Figure 71. Code Disassembler Child Window

Break Toggle

If a map/symbol file has been loaded and a text file is displayed, the text file will show addresses instead of line numbers. With your mouse, set your cursor in the text area and double-click the left button to toggle all PC breakpoints.



Figure 72 shows an example of a segment of source code in which the color of the PC breakpoint line is red.

ł	🍠 mair	.c	
	Goto	line	
Г	84	void LCD_init()	*
L	85	{	
0	_0063	LCD_E = LCD_RW = LCD_RS = 0;	// E=0
0	_0069	<pre>delay(0xffff);</pre>	// min
L	88		
L	89	<pre>// Function set (4bit long)</pre>	
0	_006C	LCD_out_Upper((UCHAR)0x20);	// 0
0	0071	LCD_out_Upper((UCHAR)0x20);	// 0
0	0076	LCD_out_Upper((UCHAR)0x80);	// N
L	93	delay(0x10);	// min =
L	94		
L	95	// Display on / off	
0	007B	LCD_out_Upper((UCHAR)0x00);	// 0
0	007E	LCD_out_Upper((UCHAR)0xc0);	// 1
L	98	delay(0x10);	// min
L	99		
	100	// Display clear	
0	0083	LCD_out_Upper((UCHAR)0x00);	// 0 _
	-nnee	ICD out Honer ((HCHAD) 0x10) .	// n *
L	•		h. A

Figure 72. Using the Text File Function, #4 of 5



🦉 mair	1.0	
Goto	bline	
84	<pre>void LCD_init()</pre>	*
85 0_0063 0_0069	<pre>{ LCD_E = LCD_RW = LCD_RS = 0; delay(0xffff);</pre>	// E=0 // min
88 89	// Function set (4bit long)	(/)
0_0000 0_0071 0_0076 93	LCD_out_Upper((UCHAR)0x20); LCD_out_Upper((UCHAR)0x80); delay(0x10);	// 0 // N // min =
94 95 0_007B	<pre>// Display on / off LCD_out_Upper((UCHAR)0x00); LCD_out_Upper((UCHAR)0x00);</pre>	// 0
98	delay(0x10);	// min
100 0_0083	<pre>// Display clear LCD_out_Upper((UCHAR)0x00); LCD_out_Upper((UCHAP)0x10); III</pre>	+ 0 // + 0 //

In Figure 73, the color of the current program counter address line is blue.

Figure 73. Using the Text File Function, #5 of 5



The Z8051 OCD In-System Programmer

The Z8051 On-Chip Debugger (OCD) In-System Programmer (ISP) has been developed to support Zilog's Z8051 8-bit MCUs. This document describes how to set up and use the Z8051 On-Chip Debugger ISP with your Z8051 Development Kit. The OCD ISP is used to program the Flash or EEPROM memory spaces of a target Z8051 MCU using Zilog's On-Chip Debugger. The OCD interface uses only two I/O lines¹ and does not require a socket adapter or specified power circuit. An example ISP screen is shown in Figure 74.

🛷 ZILOG	OCD ISP - XDATA address : 0000000	0		- 🗆 ×		
HexaData	Program <u>W</u> indow <u>H</u> elp					
Select	Z51F0811RFX/RHX/RJX/QUX F Erase XDATA (data EEPROM)					
*****	1 Read completed					
File F3	2	Config. 1'st 8bytes				
	3	00 E8 13 03 08 00 00 00				
Read	4	Checksum D21B				
	5	Passed 0				
Write	6	Flancad 0.55c				
F12		Lidpsed 0,555				
Verify	CODE address 10000000	×	XXDATA address : 00000000	×		
8	Edit NO 0 1 2 3 4 5 6 7	89ABCDEF	Edit NO 0 1 2 3 4 5 6 7 8 9 A B C D	EF		
Retard	000000 02 01 03 32 00 00 00 00 00 0 000010 00 00 00 32 00 00 00 00 00 0			-		
3	000020 00 00 00 32 00 00 00 00 00 0 000030 00 00 00 02 0F E8 00 00 00 0					
Plank						
9	000060 00 00 00 02 13 80 00 00 00 0					
	000080 00 00 00 32 00 00 00 00 00 00	0 00 02 06 5F 00 00				
Config.	000000 00 00 00 32 00 00 00 00 00 00 00 00 00 00 00 00 00	0 00 32 00 00 00 00 00	000000 00 00 00 00 00 00 00 00 00 00 00			
× 1	000080 00 00 00 32 00 00 00 00 00 00 0 000000 00 00 00 32 00 00 00 00 00 0	0 00 32 00 00 00 00 00 00 0 00 00 00 00 00 00	000000 00 00 00 00 00 00 00 00 00 00 00			
	000000 00 00 00 32 00 00 00 00 00 0 000000 00 00 00 02 12 C4 00 00 00 0	0 00 32 00 00 00 00 0 00 02 13 0C 00 00	0000D0 00 00 00 00 00 00 00 00 00 00 00			
	0000F0 00 00 00 32 00 00 00 00 00 0 000100 69 80 FE 75 81 21 12 15 BD E	0 00 02 02 41 12 01 5 82 60 03 02 00 FE				
	000110 79 0A E9 44 00 60 1B 7A 01 9	0 1C 7A 78 8D 75 A0				
	000130 A0 FF E4 78 FF F6 D8 FD 78 0	0 E8 44 00 60 0A 79				
	000140 00 75 A0 00 E4 F3 09 D8 FC 7 000150 79 01 90 00 00 E4 F0 A3 D8 F	8 8D E8 44 00 60 0C C D9 FA 90 00 7E E4				
	000160 F0 90 00 7F E4 F0 02 00 FE A 000170 90 00 56 74 80 F0 A3 74 25 F	F A8 53 07 7F 8F A8 0 A3 E4 F0 A3 F0 90	000160 00 00 00 00 00 00 00 00 00 00 00 00 0			
	000180 12 00 75 F0 7A E4 12 0D 50 1	2 14 21 12 12 B4 12 5 C1 75 F0 80 12 0E	00 00 00 00 00 00 00 00 00 00 00 00 00	-		
	Ready	and the second se		1		

Figure 74. Example On-Chip Debugger ISP Screen

Note: If your system V_{CC} is lower than device specifications, the OCD cannot program the device.

^{1.} The Z8051 OCD ISP requires a two-connection pin in your target system.


Features

The key features of the Z8051 On-Chip Debugger ISP are:

- Supports Zilog's 8-bit Z8051 Family of MCUs
- Uses the Intel HEX file format
- Display the Code and XData areas in an editable hexadecimal dump format
- Display and edit device configurations
- Autodetects target devices
- Can program eight devices simultaneously
- Performs post-programming device verification
- Supports all programming functions

Connect the Hardware

After installing the OCD software and drivers, hardware connections can be established. The pin positions of the Z8051 USB OCD interface are shown in Figure 75. Confirm the target device's pin positioning, and connect this interface to the USB port of your PC.

2	1	Pin No.	Function
4	3	2	V <sub- scriptTa- ble>CC</sub-
6	5	4	GND
8	7	6	OCD S <sub- scriptTa- ble>CLK</sub-
10	9	8	OCD S <sub- scriptTa- ble>DATA</sub-

Figure 75. OCD Hardware ISP Pin Assignment (Bottom View)



Apply Power

Observe the following procedure to complete your hardware connection to the Z8051 USB OCD interface.

1. Ensure that the power is off to the target MCU and that the MCU is soldered properly onto the target board.

Caution: If your target MCU is already powered on prior to connecting the USB OCD interface, it may not be able to recognize which mode the OCD is operating in. The target MCU is identified at power-up whether it is in OCD or User Mode.

- 2. Power on your PC.
- 3. Connect the OCD hardware to your PC.
- 4. Connect the OCD hardware to your target system.
- 5. Apply power to the target system.
- 6. Use the OCD In-System Programmer to perform your programming tasks.
- 7. When your programming work is complete, power off the target system.

Understanding the OCD ISP Menu Functions

This section describes the operation of the HexData, Program, Window and Child menus.

HexData Menu

The HexData menu, shown in Figure 76, allows users to load their hexadecimal code to a target device for programming. Because each device operates on its own programming algorithm and features a different memory map, ISP functions are enabled only after a target device has been selected.





Figure 76. The OCD ISP's File Menu

Select Device

Observe the following procedure to select a target device.

- Run the Z8051 ISP software. Navigate via the Windows Start menu to All Programs
 → Zilog Z8051 Software and Documentation <version_number> → Zilog Z8051
 ISP <version_number>.
- 2. From the **HexData** menu of the ISP, choose **Select Device**. The Device Select dialog box appears and displays a list of potential target devices, as shown in Figure 77.





Figure 77. Device Select Dialog

Load Code HEX File

Observe the following procedure to load and read a hexadecimal data file.

Note: All hexadecimal files follow the Intel HEX format.



1. Select **Load Code HEX File** from the **HexData** menu to load a hexadecimal file from the host PC to a code buffer space generated by the In-System Programmer. The Fill Buffer dialog appears, as shown in Figure 78.

ill buffer		
<u>S</u> tart address	10000	Fill
<u>E</u> nd address	FFFFF	Don't care
Fill <u>d</u> ata	FF ÷	Cancel



• Note: Loading a hexadecimal file into this code buffer space does not affect the memory space of the target device.

The Fill Buffer dialog prompts the user to enter starting and ending addresses that define the range of the code buffer space, plus the data pattern to fill the buffer space before loading the hexadecimal file.

- Clicking the Fill button performs the task of filling the code buffer with specified data values.
- Clicking the **Don't Care** button will cause the buffer to remain loaded with the data values that it currently contains.
- Clicking the Cancel button cancels the file loading tasks and closes the Fill Buffer dialog.
- 2. Click either the **Fill** button or the **Don't Care** button to open the **File Open** dialog box, which is shown in Figure 79. Next, select the hexadecimal file that you wish to load into the buffer, and click **Open**.



Open			×
Look in:	Demo	💽 🔕 🖉 🛤 🗸	
My Recent Documents Desktop My Documents My Computer My Network	demo.hex		
Places	File <u>n</u> ame: Files of <u>type</u> :	demo.hex Qpen Hexafile (*.HEX) Cancel	

Figure 79. File Open Dialog

3. The OCD_ISP dialog box appears, as shown in Figure 80. After a hexadecimal file has been loaded, this dialog displays the name of the target device and a data check-sum value, as highlighted in the figure.



🛷 ZILOG	OCD ISP	×
HexaData	Program <u>W</u> indow <u>H</u> elp	
Select	Z51F0811RFX/RHX/RJX/QUX	ase XDATA (data EEPROM)
F1	0 OK	
File	1	Config. 1'et Skyter
F3	3	
Read	4	Chastering 14D4
	5	Pacead 0
Write	6	Flansed 0.00s
FIZ		
Verify		
STAGE S		
Blunk		
9		
Config.		
?		
	Decel.	
-	neauy	

Figure 80. OCD ISP Dialog

Note: The ISP cannot calculate the checksum without a defined code buffer range (see <u>Step 1</u>). Therefore, if you have not yet selected a target device yet, the Checksum field will display ????.

Load XData HEX File

Selecting **Load XData HEX File** from the **HexData** menu loads a hexadecimal file from the host PC to the XData buffer of the ISP software; this hexadecimal file is in Intel HEX format. Loading this file to the XData buffer space does not affect the memory space of the target device.



Read Device

Selecting Load XData HEX File from the HexData menu causes the target device to be read by the OCD hardware. When the host PC detects two or more hardware devices, it prompts the user to select which device to read, as indicated in Figure 81.

💷 Select dev	ice to read 🛛 🔯
0CD_0 0CD_1 0CD_2 0CD_3	
0CD_4 0CD_5 0CD_6 0CD_7	OK

Figure 81. Select Device To Read Dialog

If the selected target device is unlocked, the OCD hardware will read the code, XData and configuration values, then update the display and the checksum.

If the selected target device is locked, the OCD hardware will display the term LOCK and prompt the user to read the configuration only.

Most Recent Files

As the user opens and closes files, these files will appear in the Hex Data menu, and can be selected at any time in a current or future session. The maximum number of most recentlyused files that will appear in the Hex Data menu is eight. See the example in Figure 82, which shows that the user has recently opened only three files; the third file (selected in the figure) is the demo.hex file.





Figure 82. Most Recently Used Files

Exit

Choosing Exit from the HexData menu immediately terminates the OCD ISP.

Program Menu

The Program menu, shown in Figure 83, lists all of the OCD ISP's programming main control functions, each of which is described in this section. This menu is enabled after the user selects a target device.



Figure 83. The OCD ISP's Program Menu



Write

Selecting **Write** from the **Program** menu applies the entire programming sequence. This sequence is listed below.

- 1. The host PC detects the OCD hardware and its connection with the target device.
- 2. All connected target devices are processed simultaneously.
- 3. The ISP erases the target device with a bulk erase algorithm.
- 4. The ISP next programs the code area.
- 5. The ISP verifies the code area.
- 6. The ISP programs the configuration area.
- 7. The ISP verifies the configuration area.

Verify

Selecting **Verify** from the **Program** menu initiates a comparison of the contents of the ISP programmer's buffer with the contents of the target device's memory. This verification sequence is described below.

- 1. The host PC detects the OCD hardware and its connection with the target device.
- 2. All connected target devices are processed simultaneously.
- 3. The ISP checks whether the target device is locked or unlocked. If the target is locked, verification is canceled.
- 4. The ISP verifies the code area.
- 5. The ISP verifies the configuration area.

Erase

Selecting **Erase** from the **Program** menu causes the entire contents of the target device's memory, including configurations, to be erased. This erasure sequence is described below.

- 1. The host PC detects the OCD hardware and its connection with the target device.
- 2. All connected target devices are processed simultaneously.
- 3. The ISP erases the target device, whether it is locked or unlocked.

Blank Check

The Blank Check function determines if the target device is blank (entirely erased) after an erasure. The sequence of this Blank Check function is described below.

- 1. The host PC detects the OCD hardware and its connection with the target device.
- 2. All connected target devices are processed simultaneously.



- 3. The ISP checks whether the target device is locked or unlocked. If the target is locked, the Blank Check function is canceled.
- 4. The ISP determines if the code area is entirely erased.
- 5. The ISP determines if the configuration area is entirely erased.

Set Configuration

Because each device in the Z8051 Series is configured differently, use the Set Configuration function to configure the target device's I/O pin functions, oscillation method, code protection, etc.; see Figure 84 for an example.

Configuration dialog	×
	Boot area
Lock code area	▼ 256 : 0100 ~ 01FF
C Lock xdata area	768 : 0100 ~ 03FF
Disable /RESET input	☐ 1792 : 0100 ~ 07FF
Enable Xin, Xout	5 3840 : 0100 ~ 0FFF
Enable Sub-Xin, Xout	Enable HARD lock
Configuration dum	ıp .
0000 : 00 E8 13 03 08 0008 : 00 00 00 00 00 0010 : 00 00 00 00 00 0018 : 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 84. Configuration Dialog

Window Menu

The Window menu, shown in Figure 85, can be used to modify the arrangement of child windows or to directly select a child window.





Figure 85. The OCD ISP's Window Menu

Open CODE Dump

Selecting **Open CODE Dump** from the **Window** menu opens a window which displays code memory in a dump format, as shown in Figure 86. If this window is already open, the window will move to the top-most level of the debugger frame.

C co	DE	ad	dre	ss :	00	000	000	0											- 1	×
Edit C	K I)	1	2	3	4	5	E	6	7	8	9	A	E	3 (¢	D	Е	F	
000000	02	01	03	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF			-	
000010	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000020	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000030	FF	FF	FF	02	OF	E8	FF	FF	FF	FF	FF	02	OF	68	FF	FF				
000040	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000050	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000060	FF	FF	FF	02	13	80	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000070	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000080	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	02	06	5F	FF	FF				
0000090	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
00000A0	FF	FF	FF	02	0A	C1	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
0000B0	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
000000	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
0000000	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
0000E0	FF	FF	FF	02	12	C4	FF	FF	FF	FF	FF	02	13	0C	FF	FF				
000050	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	02	02	41	12	01				
000100	69	80	FE	75	81	21	12	15	BD	E 5	82	60	03	02	00	FE				
000110	79	0A	E9	44	00	60	1B	7A	01	90	10	7A	78	8D	75	AO				
000120	00	E4	93	F2	A3	08	B8	00	02	05	AO	D9	F4	DA	F2	75				
000130	AO	FF	E4	78	FF	F6	D8	FD	78	00	E8	44	00	60	0A	79				
000140	00	75	AO	00	E4	F3	09	D8	FC	78	8D	E8	44	00	60	OC				
000150	79	01	90	00	00	E4	FO	A3	D8	FC	D9	FA	90	00	7E	E4				
000160	FO	90	00	7 F	E4	FO	02	00	FE	AF	Α8	53	07	7 F	8F	A8			-	

Figure 86. Open CODE Dump Child Window



Open XData Dump

Selecting **Open XData Dump** from the Window menu opens a window which displays all external data (XData) memory in a dump format. An example is shown in Figure 87. If this window is already open, the window will move to the top-most level of the debugger frame.

Σ	XD.	AT/	\ a	ddro	ess	: 0	000	000	00	<u> </u>											×
	Edit N	0)	1	2	3	4	5	e	6	7	8	9	A	E	} (2	D	Е	F	12
0(00000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			-	
0(01000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				1000
0(00020	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00				
00	00030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	00040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0.0	00				
00	00050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0(00060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0	00070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	08000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0(00090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0(OAOOC	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	00000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0	DOODO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
UI	JUUEU	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
UI	UTUUE	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	00100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
00	0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
01	0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0	10140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0	0140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0	10160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				(mail)
1	0100		00	00	00	20		00		00	00	00	00	20	00	00	0.0				

Figure 87. Open XData Dump Child Window



Cascade

Selecting **Cascade** from the **Window** menu arranges opened child windows in a stepped (cascading) visual display, as shown in Figure 88.

XDATA	A adilite	se -	anon	0001												×		
Edit NO) 1	2 3	3 4	5	6 7	7 {	3	9	A	В	C	Ę)	E	F			
000000 00	C co	DE e	addre	ss : I	1000	0000	n											x
000010 00	Edit O	K h	1	2	3 1	E	6		7	8	a	Δ	P	1		n	E	F
000030 00	Con C			-	J 4		nnl				-				-		<u> </u>	- Incol
000040 00	000000				F FF	FF FF	FF	FF	FF	FF	FF FF	FF	FF	FF	FF			*
000050 00	000010	FFI	11 13 77 77	FFF	T TT	TT FF	FF	11	11	11	FF	FF	FF	TT	FF			
000060 00	000030	FF H	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000070 00	000040	FF I	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
0000000 00	000050	FF I	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
00 0A0000	000060	FF I	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
0000B0 00	000070	FF 1		FFF	F FF	FF FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
0000000 00	000000	FFI	11 17	FFF	11 1 77 7	FF	FF	TT	FF	TT	TT	TT	FF	TT	FF			
0000000 00	0A0000	FF H	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000020 00	0000B0	FF H	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000100 00	000000	FF H	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000110 00	000000	FF H	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000120 00	DODOED	FF 1	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000130 00	000070	1 11	11 17 77 77	T TT	11 1 77 7	11	FF	11	11	11	11 FF	11	T T F F	11 FF	FF			
000140 00	000110	FF	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000150 00	000120	FF H	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
000100 00	000130	FF H	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
	000140	FF H	FF FF	FFF	FFF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
	000150	FF H	FF FF	FFF	F FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			
	000100	FF 1	FF FF	FF F	r rr	FF	r r	r r	L.L.	r r	L.L.	F F	r r	L.L.	L.L.			

Figure 88. Cascading Child Windows



Tile

Selecting **Tile** from the **Window** menu arranges opened child windows in a partitioned (tiled) display, as shown in Figure 89.

C co	DE	ad	dre	ss :	00	00	000	0										×	X×D	AT/	A ai	ldri) e e	- ī	040	100	nn										×
Edit O	Κt)	1	2	3	4	E	5 (3	7	8	9	A	E	3 0) [D	Е	Edit N	10	0	1	2	3	4	5	6	6	7	8	9	A	E	1 0	; D		E
000000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1		000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1	Sec.
000020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000020	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00		
000030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	F.F.	FF			000050	00	00	00	00	00	00	00	00	00	UU	00	00	00	00	00	00		
000060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	rr FF	FF	rr FF	FF	FF			000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000	TT FF	11	FF	T T FF	TT FF	r r FF	FF	TT FF	T T FF	FF	TT FF	11	TT FF	TT FF	TT FF	FF			0000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000030	FF	TT	FF	FF	FF	TT	FF	FF	FF	FF	FF	TT	FF	TT	FF	FF			0000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
ODODBO	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			ODOORO	00	00	00	0.0	00	00	00	00	00	00	00	00	00	00	00	nn		
0000000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			0000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			OCODEO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0000070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			ODOOFO	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000100	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000110	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000120	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000130	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000140	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000160	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000170	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000180	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF			000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000190	FF	FF	FF	r r FF	FF	FF	rr FF	rr FF	FF	r r FF	FF	r r FF	FF	r r FF	rr FF	rr FF			000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000180	L L	r r FF	FF	rr FF	T T FF	r r FF	L L L	r r FF	TT TT	r r FF	FF	rr FF	TT I	rr FF	TT FF	FF			0001R0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000100	FF	T T FF	FF	T T FF	TT FF	TT FF	L L L	T T FF	FF	11	FF	11	FF	11	FF	FF			000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
000100	FF	TT	FF	TT	FF	FF	FF	TT	FF	TT	FF	TT	FF	FF	FF	FF			000100	00	00	00	00	0.0	00	00	00	00	00	00	00	00	00	00	00		
000180	т. Т.	FF	FF	FF	FF	TT	FF	R.R.	FF	FF	FF	ਸ਼ਾਸ਼	FF	FF	FF	FF	_	•	000180	00	00	00	nn	00	00	00	nn	00	00	00	00	nn	00	00	nn		•

Figure 89. Tiled Child Windows

Close

Selecting **Close** from the **Window** menu closes the top-most child window that appears in the frame.

Windows 1, 2, 3, Etc.

This menu selection assigns a sequential number (e.g., 1, 2, 3...) to each child window in the order in which it is opened. Users can directly select any open child window by its number. In Figure 85 on page 76, for example, selecting **2** from the **Window** menu will display the XData Dump window as the top-most window in the debugger screen.

Child Windows

Child windows are secondary windows that are displayed within the main ISP window. The OCD ISP presents two child windows – the Code dump and XData windows.



Code Dump Window

Code dump windows display each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data.

The upper side of the Code Dump window displays the address of the current cursor position. The term Code is displayed as a watermark in the center of this window, as shown in Figure 90.

C	co	DE	ad	dre	ss :	00	000)50	1												×
E	dit O	K	3	1	2	3	4	5	6	6	7	8	9	A	E	3)	C	D	Е	F	
000	14A0	07	CO	06	CO	05	CO	04	CO	03	CO	02	12	0E	EB	90	17				
000	04B0	A7	75	FO	80	12	0E	3B	DO	02	DO	03	DO	04	DO	05	DO				1
000	14C0	06	DO	07	0A	DO	07	DO	06	80	AE	90	17	90	75	FO	80				
000	14D0	CO	07	CO	06	CO	05	12	0E	3B	DO	0.5	DO	06	DO	07	74				
000	14E0	10	2E	FE	E4	ЗF	FF	OD	02	04	ЗD	22	AF	83	E5	82	90				
000	14F0	00	OF	FO	A3	EF	FO	12	03	28	90	00	0F	EO	FE	A3	EO				
000	1500	88	74	FO	5E	FC	74	01	5F	FD	CO	07	CO	06	CO	05	CO				
000	0510	04	12	03	48	DO	04	DO	05	80	82	8D	83	12	03	52	12				
000	1520	03	00	12	03	48	DO	06	DO	07	8E	OB	74	01	5F	F 5	OC				
000	0530	90	00	OB	EO	F9	A3	ΕO	FA	A3	ΕO	FB	90	00	ÛE	EO	F8				
000	0540	75	0A	00	С3	E5	0A	64	80	88	FO	63	FO	80	95	FO	50				
000	0550	44	CO	00	E 5	0A	F8	33	95	EO	FD	E8	25	OB	F5	OD	ED				
000	0560	35	0C	F5	0E	90	00	8D	EO	FC	A3	EO	FD	E5	OD	20	F5				
000	0570	OD	E5	0E	ЗD	F5	0E	E 5	0A	29	F8	E4	ЗA	FC	8B	05	88				
000	0580	82	80	83	8D	FO	12	15	A1	85	OD	82	85	0E	83	FO	05				
000	1590	0A	DO	00	80	AE	8E	82	8F	83	12	03	7A	12	03	00	12				
000	DSA0	03	2F	22	AF	83	E5	82	90	00	12	FO	A3	EF	FO	12	03				
000	05B0	28	90	00	12	EO	FE	A3	ΕO	FF	74	FO	5E	FC	74	01	5F				
000	05C0	FD	8E	02	74	01	5F	FB	CO	07	CO	06	CO	0.5	CO	04	CO				
000	05D0	03	CO	02	12	03	48	DO	02	DO	03	DO	04	DO	05	80	82				
000	DSE0	8D	83	CO	03	CO	02	12	03	52	12	03	00	12	03	48	DO				
000	DSF0	02	DO	03	DO	06	DO	07	90	00	8D	ΕO	FC	A3	EO	FD	EA				
000	0600	20	FA	EB	ЗD	FB	90	00	11	EO	8A	82	8B	83	FO	8E	82				*

Figure 90. CODE Dump Child Window

Edit

Users can change data values in the Code Dump window at any time, except during programming execution. The editing method is quite simple: click the **Edit** button so that the **Edit OK** button (<u>Edit OK</u>) appears, place the cursor where you wish to make an edit, then write a new character pair in hexadecimal format. Upon changing any data, the changed value will appear after a checksum is computed.

To disable a change of values, click the **Edit OK** button so that the **Edit NO** button (Edit NO) appears.



Cursor Position

The position of the cursor can be moved either by mouse click or by keystroke. If you prefer using your keyboard, use the arrow keys (up, down, left, right) and/or the PageUp, PageDn, Home, and End keys. If you want to use your mouse, click the target position or use the scroll bar.

XData Dump Window

The XDATA Dump window displays each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. Each 256-byte page resides at an address in the range xx00–xxFFh, in which xx is the number of the page. The upper side of the XDATA Dump window displays the address of the current cursor position. The term XDATA is displayed as a watermark in the center of this window, as shown in Figure 91. Editing and cursor functions are the same for the XDATA Dump window as they are for the Code Dump window.



Figure 91. XData Dump Child Window



Appendix A. OCD Driver Installation on Windows Vista Systems

The driver programs for the Z8051 On-Chip Debugger are copied to the development PC during the software and documentation installation. In the following procedure for PCs running Windows Vista 32- and 64-bit operating systems, ensure that the target side of the OCD will remain unconnected while you install these drivers.

- 1. Connect the OCD hardware to the USB port of your PC by connecting the A (male) end of one of the two USB A (male)-to-Mini-B cables with the development PC's USB port. Connect the Mini-B end to the OCD device.
- 2. After the PC detects the new hardware, it will display the Found New Hardware Wizard dialog box, shown in Figure 92. Click Locate and install driver software (recommended).



Figure 92. Found New Hardware Dialog, Windows Vista

3. Depending on your development PC's User Account Control settings, Windows may ask for permission to continue the installation. Click **Continue**.





4. When the **Insert the Disc** dialog appears, as shown in Figure 93, select **I don't have the disc. Show me other options.** Click the **Next** button to display the dialog that follows, which is shown in Figure 94.

inser	t the disc that car	me with you	Ir ZILOG OCI	D I/F		
f you search	have the disc that cam the disc for driver soft	e with your dev ware.	vice, insert it nov	v. Windows will a	utomatically	
•	don't have the d	lisc. Show r	me other opt	ions.		

Figure 93. Install Device Driver Dialog, Windows Vista



Wind	Found New Hardware - Unknown Device ows couldn't find driver software for your device	
* (<u>Check for a solution</u> Windows will check to see if there are steps you can take to get your device vorking.	
• 1	B <u>r</u> owse my computer for driver software (advanced) .ocate and install driver software manually.	
		Cancel

Figure 94. Couldn't Find Driver Dialog, Windows Vista

5. Return to page 6 and follow <u>Steps 3 through 6</u>.



Appendix B. OCD Driver Installation on Windows XP Systems

The driver programs for the Z8051 On-Chip Debugger are copied during the software and documentation installation. On Windows XP systems, ensure that the target side of the OCD will remain unconnected while you install these drivers. *It is important that you observe the following procedure; do not skip ahead until the OCD drivers are installed.*

- 1. Connect the OCD hardware to the USB port of your PC by connecting the A-Male end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port, and connect the Mini-B end to the OCD device.
- 2. After the PC detects the new hardware, it will display the Found New Hardware Wizard dialog box, shown in Figure 95. Select Install from a list or specific location (Advanced); then click Next.



Figure 95. The Found New Hardware Wizard Welcome Screen



- 3. The next dialog box, shown in Figure 96, prompts you to enter a path or navigate to the directory in which the .inf file was installed. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse** button to navigate to one of the following paths and click the **Next** button, leaving all other selections at their default settings.
 - On 32-bit machines, use the following path:

<Z8051 Installation>\Z8051_<version_number>\device drivers\OCD USB\x32

– On 64-bit machines, use the following path:

<Z8051 Installation>\Z8051_<version_number>\device drivers\OCD USB\x64

Note: On some installations, the Found New Hardware screen shown in Figure 95 may also display the text string, Zilog Z8051 USB OCD - No Firmware. This occurrence is normal and can be disregarded.

ease	choose your search and installation options.
C	Search for the best driver in these locations.
	Use the check boxes below to limit or expand the default search, which includes local paths an removable media. The best driver found will be installed.
	Search removable media (floppy, CD-ROM)
	Include this location in the search:
	C\Program Files\Zilog\Z8051_1.0\device drivers\OCD UE
c	Don't search. I will choose the driver to install.
	Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Figure 96. The Found New Hardware Wizard's Browse Screen



4. When Windows prompts you whether to continue the installation or stop, click the **Continue Anyway** button and wait until the installation is completed (Windows may prompt you more than once). When the installation is complete, click **Finish**.



Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at <u>http://sup-port.zilog.com</u>.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the <u>Zilog Knowledge Base</u> or consider participating in the <u>Zilog Forum</u>.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at <u>http://www.zilog.com</u>.