

Z8051[™] Family of 8-Bit Microcontrollers

Z8051 On-Chip Debugger and In-System Programmer

User Manual

UM024001-0212



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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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	Revisio	n	Page
	Level	Description	No
Feb 2012	01	Original issue.	All

Z8051 Family of 8-Bit Microcontrollers Development Tools



Revision History



Table of Contents

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

Z8051 Family of 8-Bit Microcontrollers Development Tools



Child Windows	. 73
Code Dump Window	. 74
XData Dump Window	. 75
Customer Support	. 85



List of Figures

Figure 1.	On-Chip Debugger Screen
Figure 2.	An Example Setup [to be supplied]
Figure 3.	The Found New Hardware Wizard Welcome Screen
Figure 4.	The Found New Hardware Wizard's Browse Screen
Figure 5.	The OCD's File Menu
Figure 6.	Object File Dialog
Figure 7.	Open File Dialog
Figure 8.	Break Debug Dialog
Figure 9.	The OCD's Emulation Menu 10
Figure 10.	The OCD's Break/Configure Menu 12
Figure 11.	Break Control Dialog 13
Figure 12.	Break BEFORE Timing Diagram 13
Figure 13.	Break AFTER Timing Diagram 14
Figure 14.	Peripheral Control Dialog 14
Figure 15.	Z51F0811 MCU Configuration Example 15
Figure 16.	The OCD's View Menu 17
Figure 17.	The Basic Registers Dialog 18
Figure 18.	Code Disassembler Dialog 19
Figure 19.	Code Dump Dialog 20
Figure 20.	XDATA Dump Dialog
Figure 21.	IRAM Dump Dialog 22
Figure 22.	SFR Dump Dialog 23
Figure 23.	Global Variables Dialog 23
Figure 24.	Local Function Dialog
Figure 25.	A Sample Text File
Figure 26.	The OCD's Window Menu
Figure 27.	Cascaded Windows
Figure 28.	Tiled Windows
Figure 29.	Using the Basic Registers Function, #1 of 6 29
Figure 30.	Using the Basic Registers Function, #2 of 6 30
Figure 31.	Using the Basic Registers Function, #3 of 6 30
Figure 32.	Using the Basic Registers Function, #4 of 6 31
Figure 33.	Using the Basic Registers Function, #5 of 6 31

Z8051 Family of 8-Bit Microcontrollers Development Tools



D ' 0 4	
Figure 34.	Using the Basic Registers Function, #6 of 6
Figure 35.	Using the Code Disassembler Function, #1 of 3
Figure 36.	Using the Code Disassembler Function, #2 of 3
Figure 37.	Using the Code Assembler Function, #3 of 3
Figure 38.	Using the PC Break Toggle Function
Figure 39.	Using the Code Dump Function, #1 of 2
Figure 40.	Using the Code Dump Function, #2 of 2 37
Figure 41.	The Code Dump Function's Goto/Input Dialog
Figure 42.	The Code Dump Function's Pattern Load Dialog
Figure 43.	The Code Dump Function's Pattern Save Dialog
Figure 44.	The Code Dump Function's Pattern Fill Dialog
Figure 45.	Using the XDATA Dump Function, #1 of 2 40
Figure 46.	Using the XDATA Dump Function, #2 of 2
Figure 47.	The XDATA Dump Function's Goto/Input Dialog
Figure 48.	The XDATA Dump Function's Pattern Load Dialog
Figure 49.	The XDATA Dump Function's Pattern Save Dialog
Figure 50.	The XDATA Dump Function's Pattern Fill Dialog
Figure 51.	Using the IRAM Dump Function, #1 of 2
Figure 52.	Using the IRAM Dump Function, #2 of 2
Figure 53.	Using the SFR Dump Function, #1 of 3
Figure 54.	Using the SFR Dump Function, #2 of 3
Figure 55.	Using the SFRDump Function, #3 of 3
Figure 56.	The Watch Global Function's Global Variables Dialog
Figure 57.	Adding A Global Symbol
Figure 58.	Editing A Global Symbol
Figure 59.	The Watch Local Function Dialog
Figure 60.	
Figure 61.	Example Watch Local Function, #1 of 2
Figure 62.	•
Figure 63.	Using the Text File Function, #1 of 5
Figure 64.	Using the Text File Function, #2 of 5
Figure 65.	Using the Text File Function, #3 of 5
Figure 66.	Code Disassemble Dialog
Figure 67.	[<i>dialogX</i>]
Figure 68.	Using the Text File Function, #4 of 5
Figure 68.	Using the Text File Function, #4 of 5
rigule 09.	Using the reat Flie Function, #5 of 5



Figure 70.	Example On-Chip Debugger ISP Screen 59
Figure 71.	OCD Hardware ISP Pin Assignment (Bottom View) 60
Figure 72.	The OCD ISP's File Menu 61
Figure 73.	Device Select Dialog
Figure 74.	Fill Buffer Dialog63
Figure 75.	File Open Dialog
Figure 76.	OCD ISP Dialog
Figure 77.	Select Device To Read Dialog
Figure 78.	Most Recently Used Files 67
Figure 79.	The OCD ISP's Program Menu 67
Figure 80.	Configuration Dialog
Figure 81.	The OCD ISP's Window Menu 70
Figure 82.	Open CODE Dump Child Window
Figure 83.	Open XData Dump Child Window
Figure 84.	Cascading Child Windows 72
Figure 85.	Tiled Child Windows
Figure 86.	CODE Dump Child Window
Figure 87.	XData Dump Child Window
Figure 88.	Found New Hardware Dialog, Windows Vista
Figure 89.	Install Device Driver Dialog, Windows Vista
Figure 90.	Couldn't Find Driver Dialog, Windows Vista
Figure 91.	Browse For Driver Dialog, Windows Vista
Figure 92.	Can't Verify Publisher Dialog, Windows Vista
Figure 93.	Successfully Installed Dialog, Windows Vista
Figure 94.	Install Device Driver Dialog, Windows 7 82
Figure 95.	Driver Software Installation Dialog, Windows 7
Figure 96.	Device Manager Dialog, Windows 7

Z8051 Family of 8-Bit Microcontrollers Development Tools



List of Figures



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.

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.

Z8051 Family of 8-Bit Microcontrollers Development Tools



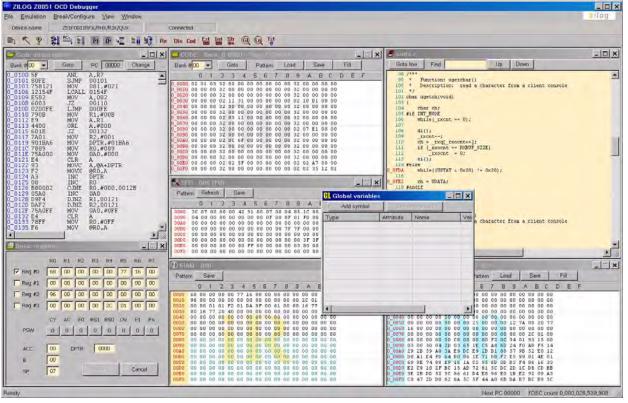


Figure 1. On-Chip Debugger Screen

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.



3

- 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

Z8051 OCD Software and Documentation Installation

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>.exe.

Note: In this filename, <version_number> refers to the version number of the OCD software. 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 Windows XP systems, ensure



that the target side of the OCD will remain unconnected while you install these drivers. Refer to Figure 2 for guidance.

Figure 2. An Example Setup [to be supplied]

Note: If you are running Windows Vista, see <u>Appendix A</u> on page 76 to install your device drivers. If you are running Windows 7, see <u>Appendix B</u> on page 82.

- 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 3. Select Install from a list or specific location (Advanced); then click Next.





Figure 3. The Found New Hardware Wizard Welcome Screen

- 3. The next dialog box, shown in Figure 4, prompts you to choose either the browse or entry options for where you will store 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 and click the **Next** button, leaving all other selections at their default settings. [For correct and current branding, please change instances of "ZiLOG" to "Zilog".]
 - On 32-bit machines, use the following path:

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

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

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



Found New Hardware Wizard
Please choose your search and installation options.
 Search for the best driver in these locations. Use the check boxes below to limit or expand the default search, which includes local paths and 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 L
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.
< <u>B</u> ack <u>N</u> ext > Cancel

Figure 4. 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**.

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.

7

DIXYS Company

- 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 5.

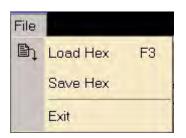


Figure 5. The OCD's File Menu

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 6.

Z8051 Family of 8-Bit Microcontrollers Development Tools



Object file di	ialog		x
Hexa file nar	me	-	Common area size
🔽 BankO	amples\Z51F0811\Demo\demo.hex	Browse	✓ No common area 0 ~ 0x7FFF (32KB)
🔽 Bank1		Browse	0 ~ 0x3FFF (16KB) 0 ~ 0x8FFF (48KB)
🔽 Bank2		Browse	Symbol file CMProgram Piles V2LOG\28051_1 D\samples\251P
🔽 Bank3		Browse	
F Bank4		Browse	Do VERIFICATION after download
🔽 Bank5		Browse	dot
🔽 Bank6		Browse	
F Bank7		Browse	
F Bank8		Browse	
🔽 Bank9		Browse	
F Bank10		Browse	
F Bank11		Browse	
F Bank12		Browse	
🔽 Bank13		Browse	
F Bank14		Browse	
Bank15		Brawse	save message to rile Download Close

Figure 6. 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 7.



ien					
Look in:	🔁 Demo		• 0	1 🕫 📰 🔻	
My Recent Documents Desktop My Documents My Computer My Network	 clock.h demo.hex eeprom.h leds.h main.h switches.h timer.h wt.h 				
Places	File <u>n</u> ame:	demo.hex		•	<u>O</u> pen
	Files of type:	Hex file (*.H*)		-	Cancel

Figure 7. 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 8. Click **Yes** if you wish to remove these breakpoints, or **No** if you prefer to retain them.

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

Figure 8. Break Debug Dialog

I

I

I

Z8051 Family of 8-Bit Microcontrollers Development Tools



- 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 9, 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 9. 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:

- A stopped address of [from the?] previous emulation [the last known address at the stopping of the last emulation that was run? Please clarify.].
- Where 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 *[new?]* 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.

Z8051 Family of 8-Bit Microcontrollers Development Tools



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 10, lists PC breakpoint control, device configuration and hardware test functions.

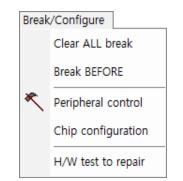


Figure 10. The OCD's Break/Configure Menu

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 11.



Figure 11. 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 12.

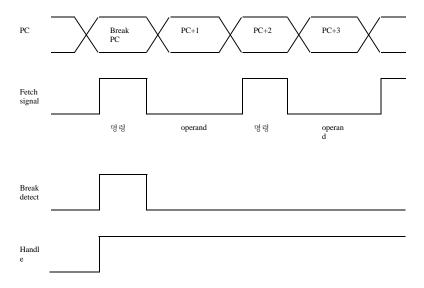


Figure 12. Break BEFORE Timing Diagram

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 13.

Z8051 Family of 8-Bit Microcontrollers Development Tools



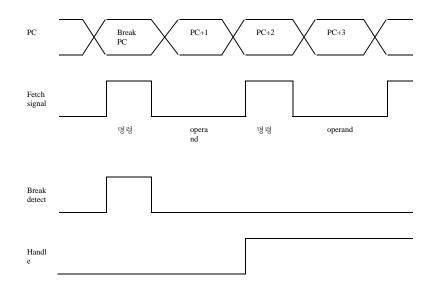


Figure 13. Break AFTER Timing Diagram

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 14. These peripherals are always running during emulation by default.

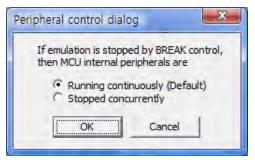


Figure 14. Peripheral Control Dialog

The Peripheral Control function *is used timer cycle measuring in interrupt subroutine with peripheral STOP.* [I'm afraid I cannot decipher this statement; please clarify.]



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 15 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	
0000 : 00 E8 13 03 08 0008 : 00 00 00 00 00	
0010:00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 Close

Figure 15. 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 16, 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 16. 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.

Z8051 Family of 8-Bit Microcontrollers Development Tools



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 17.

📒 Basic regi	sters	<u></u>						. 🗆 ×
1000	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 B	00	D	PTR	0	000			
SP	07			Mod	lify		Cance	el

Figure 17. 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 18.



🗁 Code disassemb	ler		- • •
Bank # 00 💌	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	MON	008,R3	
0_000B 8A09	MON	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_001B 7012	JNZ	0002F	
0 001D 750D80		00D, # 080	
<u>n nn2n 8nn</u> D	SIMP	0002F	
III			E. A

Figure 18. 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 19.

Z8051 Family of 8-Bit Microcontrollers **Development Tools**



Bank #	00	•		Goto)	I	Patte	rn	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0000	02	01	BA	7B	FF	E 4	FD	7 F	01	8B	80	8A	09	89	0A	EF	
0010	14	60	0 F	14	60	11	14	60	13	24	03	70	12	75	0D	80	
0020	80	0 D	75	0D	C0	80	80	75	0D	94	80	03	75	0 D	D4	ΕD	
0030	25	0 D	F5	0D	C2	00	F5	0 E	12	01	80	E 4	F5	0B	AB	80	
0040	AA	09	Α9	0A	85	0B	82	75	83	00	12	00	E2	F5	0C	60	
0050	11	D2	00	85	00	0E	12	01	80	05	0B	E5	0B	C3	94	40	
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	A0	7 F	C0	
0800_0	12	01	Α9	12	01	Α0	7 F	10	12	01	Α9	7 F	00	7E	10	12	
0090	01	63	E 4	12	01	Α9	7 F	60	12	01	Α9	C2	00	75	0E	01	
0A00_0	12	01	80	C2	00	75	0 E	28	02	01	80	75	Α0	FF	75	98	
0_00В0	FF	12	00	63	7B	FF	7A	01	79	3B	E 4	FD	FF	12	00	09	
0000	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	Ε6	22	BB	FE	06	Ε9	25	82	F8	

Figure 19. 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 20.



S XD	ATA :	Banl	k_0 (00CF	: Pa	ige (cs o	096								•	×
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_000) 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_00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0_00B	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	
0_00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0_00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0_00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Figure 20. 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 21.

Z8051 Family of 8-Bit Microcontrollers **Development Tools**



Patterr	۱	Save	•														
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0000	EC	8B	FB	E3	B7	4 A	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	B5	43	
0050	32	28	ΒA	D4	62	12	66	8A	D1	11	7C	73	0C	7C	C2	0E	
0060	C2	С9	A3	3B	F4	С3	СВ	00	AC	0 E	D7	C1	0D	0C	31	80	
0070	DO	Ε3	62	ЗA	E 9	56	D2	E 9	94	20	FC	7 F	E 9	EΒ	89	ΕE	
0800	00	C2	80	B8	6 F	FB	5B	E3	51	12	Α7	Β3	D2	FB	DC	AE	
0090	0B	ΕE	00	8B	71	7B	35	6D	1 F	E 6	34	26	Ε6	41	98	F1	
00A0	DD	7 F	BD	8 D	B8	EΒ	2 F	ED	Β4	1D	9D	C5	5D	B6	D9	EΕ	
00B0	44	6D	2 D	85	49	C7	55	5E	29	ЗF	76	65	7E	DD	77	BF	
00C0	AD	06	26	43	AA	D9	F4	3E	82	91	32	4B	F 9	B6	1E	F5	
00D0	7 F	C1	E5	C5	FF	B8	B6	AE	7 F	A3	2E	A0	Β7	2 D	DD	DA	
00E0	E 5	ΒE	C1	В0	FF	FC	CF	D5	43	9C	27	76	4 E	5 F	38	B5	
00F0	E7	1E	B2	70	BB	BE	В9	BF	ΕO	9 F	Ε1	69	77	5E	CD	ED	

Figure 21. 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 22.



🛠 SFR :	0A8	3															x
Pattern	R	efres	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	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	00	00	00	00	00	80	FF	00	00	00	00	03	80	00	00	00	
OOFO	00	00	00	00	00	00	00	00	00	00	03	C0	60	80	00	00	

Figure 22. 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 23.

Add symbol	Remov	e symbol			
Туре	Attribute	Name	Value	Address	1
BIT	BIT	LCD_E	0	0x88.2	
BIT	BIT	LCD_RS	1	0x88.0	
BIT	BIT	LCD_RW	1	0x88.1	1
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	

Figure 23. 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 24.

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

Figure 24. 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 25.



Goto lin	e	
0180	LCD_busy();	
0183	$LCD_E = LCD_RW = 0;$	// E=0, RW=0
_0187	LCD_RS = TYPE;	<pre>// RS=INSTRUCTION(0), CHARAC</pre>
127	and the second se	
_018B	$R0_port = uch >> 4;$	// output data High
_0192	$LCD_E = 1;$	
_0194 131	$LCD_E = 0;$	
_0196	R0_port = uch;	// output data Low
0199	$LCD_E = 1;$	
0198	$LCD_E = 0;$	
019D	$LCD{RS} = 0;$	
_019F }		
137		
	bid LCD_busy()	
139 { 140	bit test;	
141	int i;	
142	1110 1,	
010F	LCD E = LCD RS = 0;	// E=0, RS=0
0113	R0 io = 0xf0;	// R03~00 : change to input
0116	$LCD_RW = 1;$	// RW=1
0118	for(i=0; i<1000; i++) {	
_011B	$LCD_E = 1;$	// E=1
_011D	test = R03;	// Get busy flag
_0121	$LCD_E = 0;$	// E=0
_0123	$LCD_E = 1;$	// E=1 : Skip AC3
0125	LCD E = 0;	// E=0

Figure 25. 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 26, 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 26. The OCD's Window Menu



Cascade

This menu selection arranges opened child windows in a stepped visual sequence, as shown in Figure 27.

Basic registers		
CODE . Bank 0 0000 : Page CS 5	DSF	
B XDATA : Bank 0 0000 : Page	CS 0096	
B 🛠 SFR. 080		
L IRAM : OCB		
Code disassemb	fr	
0 0 Bank # 00 -	Goto PC 00000 Change	
0 0		

Figure 27. Cascaded Windows



Tile

This menu selection arranges opened child windows in a partitioned display, as shown in Figure 28.

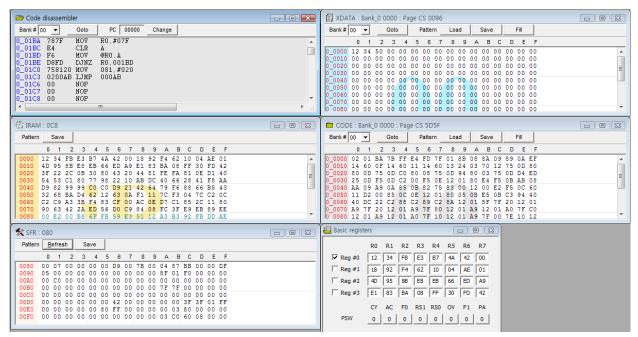


Figure 28. Tiled Windows

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 26 on page 26, 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 29 shows an example Z8051 Basic Registers window.

🖶 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
PSW	CY 0	AC 0	F0 0	RS1 0	RS0 0	0V 0	F1 0	PA 0
ACC B SP	00 00 07	D	PTR	Mod	DOO		Cance	el

Figure 29. 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 30, 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.



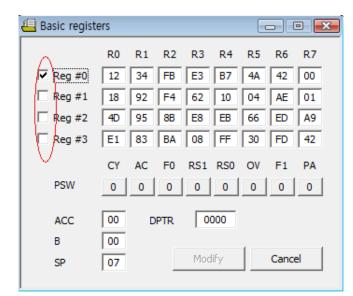


Figure 30. Using the Basic Registers Function, #2 of 6

In Figure 31, 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	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
Ì	CY	AC	F0	RS1	RS0	٥V	F1	PA
PSW	0	0	0	0	0	0	0	0
ACC	00	D	PTR	0	000			
в	00							
SP	07			Mod	lify		Cance	el

Figure 31. Using the Basic Registers Function, #3 of 6



In Figure 32, 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)
PSW	0	0	0	0	0	0	0	0	J
ACC	00	D	PTR	0	000				
В	00			2					
SP	07			Mod	lify		Cance	el	

Figure 32. Using the Basic Registers Function, #4 of 6

In Figure 33, 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 33. Using the Basic Registers Function, #5 of 6



In Figure 34, 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.

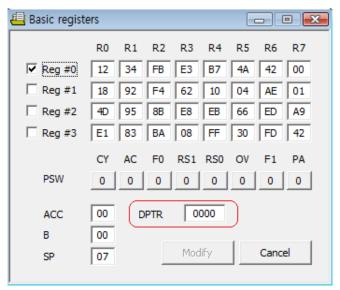


Figure 34. Using the Basic Registers Function, #6 of 6

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 35 shows an example Code Disassembler window.



🗁 Code d	disassemble	er		- • •
Bank # 0	0 🔻 🗌	Goto	PC 00000	Change
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_001B	7012	JNZ	0002F	
0_001D	750D80	MOV	00D,#080	
0_0020		SJMP	0002F	
0_0022	750DC0	MOV	00D,#0C0	Ψ.
•				► a

Figure 35. 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 36. Double-click any of these highlighted boxes to open its source file and move to the appropriate address line.

🗁 Code d	isassemble	er		x
Bank # 00	•	Goto	PC 00000 Change	
0 0009	8B08	MOV	008,R3	
0_000B	8A09	MOV	009,R2	m.
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	JNZ	0002F	
0 001D	750D80	MOV	00D,#080	
0_0020	800D	SJMP	0002F	
0 0022	750DC0	MOV	00D,#0C0	Ŧ
•	11		F	

Figure 36. Using the Code Disassembler Function, #2 of 3

I



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 37. Change the contents of the line by entering an instruction, operand, etc., in hexadecimal format.

🗁 Code d	disassembl	er		x
Bank # 0) 🔻	Goto	PC 00000 Change	
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_001B	7012	187	00028	
0 001D	MOV 000	D,#080	Modify	
0_0020	0000	SUME	0002r	
0 0022	750DC0	MOV	00D,#0C0	Ψ.
•	111			► ai

Figure 37. 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 38, the red line represents a program counter breakpoint in the line, and the blue line represents the current program counter.



Bank # 00 ▼ Goto PC 0016F Change 0_016E 07 INC @R1 ▲ 0_016F EB MOV A, R3 ■ 0_0170 1B DEC R3 ■
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 12010F LCALL 0010F 👻

Figure 38. 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 39, for example, the page number is 00.



Bank #	00	•		Goto		F	Patte	rn _	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0000	02	01	BA	7B	FF	E 4	FD	7 F	01	8B	08	8A	09	89	0A	EF	
0010	14	60	0 F	14	60	11	14	60	13	24	03	70	12	75	0D	80	
0020	80	0D	75	0 D	CO	80	08	75	0D	94	80	03	75	0D	D4	ED	
00030	25	0 D	F5	0 D	C2	00	F5	0 E	12	01	80	E 4	F5	0B	AB	08	
00040	AA	09	Α9	0A	85	0B	82	75	83	00	12	00	E 2	F5	0C	60	
0050	11	D2	00	85	00	0E	12	01	80	05	0B	E5	0B	C3	94	40	
0000	40	DC	22	C2	88	C2	89	C2	8A	12	01	5 F	7 F	20	12	01	
0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	AO	7 F	CO	
0800 0	12	01	Α9	12	01	A0	7 F	10	12	01	A 9	7 F	00	7E	10	12	
0000	01	63	E4	12	01	Α9	7 F	60	12	01	Α9	C2	00	75	0E	01	
OA00 C	12	01	80	C2	00	75	0E	28	02	01	80	75	A 0	FF	75	98	
0 00В0	FF	12	00	63	7B	FF	7A	01	79	3B	E 4	FD	FF	12	00	09	
0000	7A	01	79	44	12	0.0	03	12	01	5 F	7B	FF	7A	01	79	4 D	
0000	E 4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5 F	
000E0	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	3A	F5	83	E0	
0 OOFO	22	50	06	E 9	25	82	F8	E 6	22	BB	FE	06	E 9	25	82	F8	

Figure 39. 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 40.



	E : B	ank_	0 00	035	: Pag	je C	S 50	71								•	ζ.
Bank #	00	•		Goto)		Patte	rn _	Lo	ad		Save	9		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	08	8A	09	89	0A	EF	
0_0010	12	34	32	51	20	11	14	60	14	12	34	70	12	75	0D	80	
0_0020	80	00	75	00	CO	-80	08	75	0 D	94	80	03	75	0 D	D4	ED	
0_0030	25	AB	39	13	44	d٥	F5	0 E	12	01	80	E 4	F5	0B	AB	80	
0_0040	AA)	09	Α9	0A	85	OB	82	75	83	00	12	00	E2	F5	0C	60	
0_0050	11	D2	00	85	00	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	A 9	12	01	A0	7 F	CO	
0_0080	12	01	Α9	12	01	AO	7 F	10	12	01	A 9	7 F	00	7E	10	12	
0_0090	01	63	E 4	12	01	A 9	7 F	60	12	01	A 9	C2	00	75	0E	01	
0_00A0	12	01	80	C2	00	75	0E	28	02	01	80	75	AO	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	ΕO	
0_00F0	22	50	06	E 9	25	82	F8	E 6	22	BB	FE	06	E 9	25	82	F8	

Figure 40. 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 41.





Figure 41. 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 42.



Figure 42. 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 43.



lialog	
00 💌	OK
0300	Cancel
03FF	-
	Browse
	00 💌

Figure 43. 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 44.

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

Figure 44. 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 45, for example, the page number is 00.



Bank #	00	-		Goto)	1	Patte	rn _	Lo	ad		Save	•		Fill		
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0000_0	02	11	FA	02	12	39	FD	7 F	01	8B	08	8A	09	89	ΟA	EF	
0_0010	14	60	0 F	02	12	3A	14	60	13	24	03	70	12	75	0D	80	
0_0020	80	0 D	75	02	12	33	08	75	0 D	94	80	03	75	0 D	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	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	AO	7 F	C0	
0_0080	21	BD	41	06	41	78	41	CO	61	0C	61	54	01	EB	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	Β6	F0	D5	Β6	FC	D2	
0_00В0	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	B6	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	FE	75	81	BF	75	A 8	
0_00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	Α0	80	

Figure 45. 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 46.



É) XDA	TA :	Banl	<u>_</u> 0 (0028	: Pa	ige (CS 5	BD5									x
	Bank #	00	•		Goto	0	I	Patte	rn _	Lo	ad		Save	•		Fill		
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
	0000	02	11	FA	02	12	39	FD	7 F	01	8B	08	8A	09	89	ΟA	EF	
	0010	14	60	0 F	02	12-	3A	14	60	-1,3	24	03	70	12	75	0D	80	
	0020	80	0 D	75	02	12	98	13	24	0D	94	80	03	75	0 D	D4	ED	
	_0030	25	0 D	F5	02	12-	36	F5	0E	-1 2	01	80	02	12	3B	AB	08	
	_0040	AA	09	Α9	02	0 F	5E	82	75	83	00	12	02	12	3C	0C	60	
	0050	11	D2	00	85	00	0 E	12	01	80	05	0B	E5	0B	C3	94	40	
	0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5 F	7 F	20	12	01	
	_0070	Α9	7 F	20	12	01	Α9	7 F	80	12	01	Α9	12	01	Α0	7 F	C0	
	_0080	21	BD	41	06	41	78	41	CO	61	0C	61	54	01	EB	01	E 4	
	_0090	02	63	01	96	01	C1	12	01	35	12	01	18	70	22	75	84	
	_00A0	00	75	85	02	74	00	85	C6	Β6	05	B6	F0	D5	B6	FC	D2	
	_00B0	C1	20	C1	FD	12	03	7A	12	01	5E	75	84	00	75	85	00	
	00C0	22	75	84	00	75	85	02	90	80	00	43	C1	01	75	B6	40	
	00D0	74	00	FO	D5	Β6	FC	D2	C0	20	C0	FD	63	C1	01	75	85	
	_00E0	00	74	00	22	63	D9	10	D2	C4	80	FE	75	81	BF	75	A 8	
0	00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	A 0	80	

Figure 46. 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 47.

put dialog box		
Address	0300	
0000 ~ FFFF		
ок		
Cancel		

Figure 47. 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 48.



Figure 48. 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 49.

[
OK
Cancel
Browse

Figure 49. 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 50.



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

Figure 50. 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 51.

Figure 51 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.



Pattern		Save	•														
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Ε	F	
0000	E8	8B	FB	E3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11	
0010	4 D	95	8B	ΕA	EВ	66	ΕD	Α9	F1	83	BB	80	FF	3A	7 D	62	
0020	DF	22	2C	09	34	80	5B	20	45	81	ΕE	FA	81	0 E	D5	60	
0030	55	53	C1	80	73	98	6A	10	AB	D4	40	66	68	49	B8	AA	
0040	D9	Α2	В9	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	0E	
0060	C0	C 9	Α3	3B	F4	C3	CF	00	BC	0 E	D5	C3	0C	2C	11	80	
0070	DO	Ε3	62	3A	AD	56	D2	E 9	94	80	FC	37	E 9	EΒ	89	EE	
0080	00	C2	80	В8	6E	FB	59	Α3	51	12	Α3	В3	92	FB	DC	AE	
0090	4B	ΕA	04	8 F	71	7B	37	6D	1 F	E 6	Β4	26	E6	41	98	F1	
00A0	DD	7 F	BD	8D	B8	ΕA	2 F	EF	BC	1D	DD	E5	5D	B0	D1	EE	
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	F 4	BE	82	91	32	4B	F8	B6	1E	E5	
00D0	7 F	C1	E5	C4	BF	B8	B6	AC	7 F	63	2E	A0	Β7	2 D	FD	DA	
00E0	E5	BE	C5	В0	FF	FC	CF	D4	43	98	27	76	48	7 F	3E	B5	
00F0	EF	1 F	B2	70	BB	ΒE	Β9	BF	Ε1	8 F	Ε1	79	77	56	CD	ED	

Figure 51. 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 52.



1]) IRAN	1:0	9D															x
	Pattern		Save	•														
		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
Γ	0000	E8	8B	FB	E3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11	
	0010	4 D	95	8B	ΕA	EВ	66	ЕĻ	A9	F1	-8,3	BB	80	FF	3A	7 D	62	
	0020	DF	22	2C	09	34	80	5B	12	34	81	ΕE	FA	81	0 E	D5	60	
	0030	55	53	C1	80	73	98	6A	10	ÀΒ	D 4	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	C0	C 9	Α3	3B	F4	C3	CF	00	BC	0 E	D5	C3	0C	2C	11	80	
	0070	D0	Ε3	62	3A	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	A3	B3	92	-FB	DC	AE	
	0090	4B	ΕA	04	8 F	71	7B	37	6D	11	98	76	ΒE	DA	41	98	F1	
	00A0	DD	7 F	ВD	8D	B8	ΕA	2 F	EF	BC	1D	DD	E5	5D	-B0	D1	EE	
	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	F 4	BE	82	91	32	4B	F8	Β6	1E	E5	
	00D0	7 F	C1	E5	C4	BF	B8	Β6	AC	7 F	63	2E	A0	Β7	2 D	FD	DA	
	00E0	E5	BE	C5	В0	FF	FC	CF	D4	43	98	27	76	48	7 F	3E	B5	
	00F0	ΕF	1 F	B2	70	BB	BE	В9	BF	Ε1	8 F	E1	79	77	56	CD	ED	

Figure 52. 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 53 shows SFR addresses in the range 80h-FFh, which represent the direct area of IRAM.



Patterr	ו <u>R</u>	efres	sh	S	ave												
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	Е	F	
0800	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 53. 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 54.

🕺 SFR :	: 046	5															x
Pattern	R	efres	sh	S	ave												
	0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	
0800	29	22	4 C	01	00	00	D9	00	00	00	04	87	76	00	00	DF	
0090	05	00	0,0	00	00	00	6,0	00	FF	00	8 F	01	F0	00	00	00	
00A0	7 F	C0	00	97	FE	D5	00	00	00	00	00	00	00	00	00	00	
00B0	00	00	00	00	00	00	-0'0	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	8 0	00	00	

Figure 54. 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 55) redisplays all current data.

🛠 SFR	: 00	D		_													x
Patter	n <u>R</u>	efre	sh	s	ave												
	-0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	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	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	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 55. 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 56.



Add symbol	Remov	e 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	0x80	
BIT	BIT	R03	0	0x80.3	

Figure 56. 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 57.

R0_port	ADD
LCD_E LCD_RS LCD_RW R0_jo R0_port R03 R1_jo	Cancel

Figure 57. 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 58.

😗 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

Figure 58. 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 59, alleviates this problem by employing a map/symbol file, as described in the previous section.



Туре	Attribute	Name	Value	Address
BIT	BIT	TYPE	0	0x20.0
unsigned char	DATA	uch	0x1	0xE

Figure 59. 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 60.

U Local function()	: LCD_OUT			. • 💌
Туре	Attribute	Name	Value	Address
BIT	BIT	TYPE	0	0x20.0
unsigned char	DATA	uch	0×1	Modify

Figure 60. 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.

I



Figure 61 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).

Goto line				
72 [lay(UINT uCnt T tmp, uu;)		•
0163 for	(tmp=10; tmp!	=0; tmp) {		T
_0167 _016B _0176 } _017F } _80	<pre>uu = uCnt; while(uu !=</pre>	0) uu;		E
81 // 82 // LCD 83 //	Function			
	E = LCD_RW =	LCD_RS = 0;		// E=0
85 {	E = LCD_RW =	LCD_RS = 0;		// E=0 // min -
85 { _0063 LCD _0069 del	_E = LCD_RW = ay(0xffff);	LCD_RS = 0;		
85 { _0063 LCD _0069 de1	_E = LCD_RW = ay(0xffff);	LCD_RS = 0;	Value	// min -
85 { 0063 LCD 0069 de1	E = LCD_RW = ay(0xffff); m 0:_DELAY)	-	// min _
0063 LCD 0069 de1 0069 de1	DE = LCD_RW = ay(0xffff); m 0:_DELAY Attribute	Name	Value	// min
85 { 0063 LCD 0069 de1 0 Local function Type unsigned int	0 E = LCD_RW = ay(0xffff); m 0:_DELAY Attribute DATA	Name uCnt	Value 0xFFFF	// min

Figure 61. 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 62.

🦉 main.c				
Goto line				
116	a second a second			Sec. 1
	_port = uch >:	> 4;		// out
	$\mathbf{D}_{\mathbf{E}} = 1;$			
	$D_E = 0;$			
121 (
	CD and the TW	DE HOURD walks		
122 Void 1	CD_out (pit II)	PE, UCHAR uch)		
	Dhuard			
	D_busy(); D E = LCD RW =	- 0+		// E=0
	D RS = TYPE;	42		// RS=
127	D_ND - IIIL,			// 10-
	port = uch >	> 4:		// out :
-	D E = 1;			// ouo
-	D = 0;			
131				
0 0196 RO	port = uch;			// out
	D = 1;			
	n F - n.			
*	TH.			
Local functio	n(): LCD_OUT			
Туре	Attribute	Name	Value	Address
BIT	BIT	TYPE	0	0x20.0
unsigned char	DATA	uch	0x1	OxE
and a service of the	(Sectors)	101		8 (M2)

Figure 62. 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 hardware 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 63.



84 vo: 85 (86 87 88	<pre>id LCD_init() LCD_E = LCD_RW = LCD_RS = 0; delay(0xffff);</pre>	// E=0 // min
86 87		
87		
	<pre>delay(0xffff);</pre>	// min
88		
89	// Function set (4bit long)	
90	LCD_out_Upper((UCHAR)0x20);	11 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);	// 0
102	ICD out Unner//IICHAD(0v10).	// 0

Figure 63. 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 64.

🖉 main.c		
Goto lin	ie	
	oid LCD_init()	
85 { 0 0063 0 0069	<pre>LCD_E = LCD_RW = LCD_RS = 0; delay(0xffff);</pre>	// E=0 // min
89 006C 0071 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 _007B _007E _98 _99	<pre>// Display on / off LCD_out_Upper((UCHAR)0x00); LCD_out_Upper((UCHAR)0xc0); delay(0x10);</pre>	// 0 // 1 // min
100 _0083	// Display clear LCD_out_Upper((UCHAR)0x00); LCD_out_Upper/(UCHAR)0x10);	// o .

Figure 64. 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 65. Map/symbol information is not required.

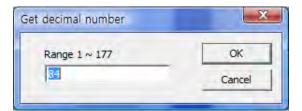


Figure 65. Using the Text File Function, #3 of 5

Disassemble Window Linkage

If a map/symbol file has been loaded and a text file is displayed, the text file will show addresses instead of line numbers. In the *[name]* dialog, 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; see Figure 66.



🎯 Code disasser	nbler	and the second second	_ 🗆 ×
Bank #00 💌	Goto	PC 00000 Change	
0_02B3 C006 0_02B5 C005 0_02B7 C004 0_02B9 C003 0_02BB C002 0_02BB C002 0_02BF C000 0_02C1 C0D0 0_02C3 75D000 0_02C3 75D000 0_02C9 E0 0_02CA FF 0_02CB 701B 0_02CB 701B 0_02CB 701B 0_02D2 7F7D 0_02D4 1E 0_02D5 BEFF01 0_02D5 BEFF01 0_02D8 1F 0_02D9 EE 0_02D8 4F 0_02D9 EE 0_02D8 70F7 0_02DB 70F7 0_02DB 70F7 0_02DB 70F7 0_02DB 70F7	PUSH PUSH PUSH PUSH PUSH PUSH PUSH MOV MOV MOV JNZ LCALL MOV DEC CJNE DEC CJNE DEC MOV ORL JNZ MOV	006 005 004 003 002 001 000 0D0,#000 0D0,#000 0D0,#000 0D7TR,#00093 A,@DPTR R7,A 002E8 00B8C R6,#000 R7,#07D R6 R6,#0FF,002D9 R7 A,R6 A,R7 002D4 DPTR,#016A7	

Figure 66. Code Disassemble Dialog

Example. Double-click the left button on your mouse at address 0_02C6 . If the text file is not open, the *[dialogX]* dialog box will open and show the 0_02C6 location at the top of the dialog; see Figure 67.



I

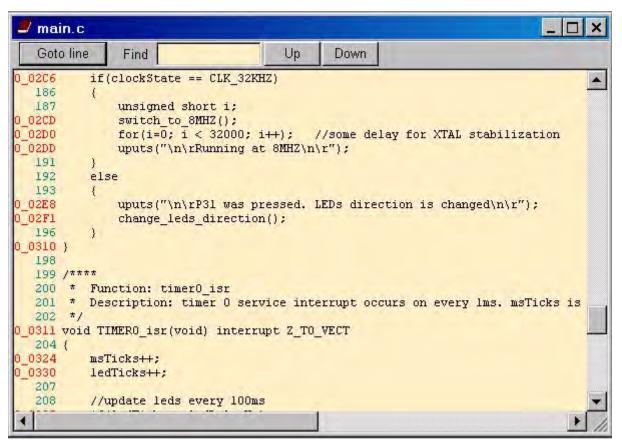


Figure 67. [dialogX]

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

🦉 main.c		- 0 🔀
Goto line		
84 vo:	id LCD_init()	*
85 {		
0_0063	LCD_E = LCD_RW = LCD_RS = 0;	// E=0
0_0069	delay(0xffff);	// min
88		
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
93	delay(0x10);	// min =
94		
95	// Display on / off	
0_007B	LCD_out_Upper((UCHAR)0x00);	// 0
0_007E	LCD_out_Upper((UCHAR)0xc0);	// 1
98	delay(0x10);	// min
99		
100	// Display clear	
0_0083	LCD_out_Upper((UCHAR)0x00);	// 0 _
0,0086	TCD out Honer (/HCHAD) 0v10) .	// 0
	III	►

Figure 68. Using the Text File Function, #4 of 5

I



In Figure 69, the color of the current program counter address line is blue.

🦉 main.c		- • •
Goto lin	e	
84 vo	oid LCD_init()	A
85 {		
0_0063	$LCD_E = LCD_RW = LCD_RS = 0;$	// E=0
0_0069	<pre>delay(0xffff);</pre>	// min
88		
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
93	delay(0x10);	// min =
94		
95	// Display on / off	
0_007B	LCD_out_Upper((UCHAR)0x00);	// 0
0_007E	LCD_out_Upper((UCHAR)0xc0);	// 1
98	delay(0x10);	// min
99		
100	// Display clear	
0_0083	LCD_out_Upper((UCHAR)0x00);	// 0 _
0,0086	ICD out Honer (/HCHAD) 0v10) .	// 0
		►

Figure 69. 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.

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.



	SP - XDATA address : 0000000 [.] am <u>W</u> indow <u>H</u> elp		
leot Z511 F1 0	F0811RFX/RHX/RJX/QUX	Erase XDATA (data EEPROM)	
F3 2 3		Config. 1'st 8bytes 00 E8 13 03 08 00 00 00	
ead 4 5		Checksum D21B	
rite 6 12 7		Passed 0 Elapsed 0,55s	
	DDE address 00000000	X XDATA address : 00000000	
ank p p p p p p p p p p p p p	NO D 1 2 3 4 5 6 7 0	00 00 32 00<	00
0000000 000100 000110 000120 000120 000140 000150 000160	00 00<	00 00 02 13 0C 00<	00 00 00 00 00 00 00 00 00 00 00 00 00

Figure 70. Example On-Chip Debugger ISP Screen

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



60

I

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 71. 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 71. 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 72, 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.

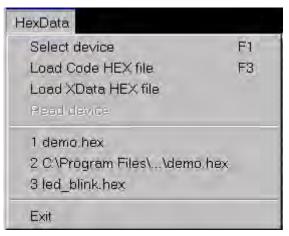


Figure 72. The OCD ISP's File Menu

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 73.



I

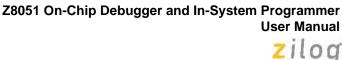


Figure 73. Device Select Dialog

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 74.

ll 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 75. Next, select the hexadecimal file that you wish to load into the buffer, and click **Open**.



I

Open	-				
Look in:	🔁 Demo		<u> </u>	2 🤌 🛄 🔻	
	demo.hex				
My Recent					
Documents					
Desktop					
My Documents					
1					
My Computer					
kilu Notwork					
My Network Places	1				
the second s	File <u>n</u> ame:	demo.hex		-	<u>O</u> pe
	I HA Hander				

Figure 75. File Open Dialog

3. The OCD_ISP dialog box appears, as shown in Figure 76. 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		
F3	3	Config. 1'st 8bytes 00 00 00 00 00 00 00 00	
Read	4		
	5	Checksum 14D4	
Write	6	Passed 0	
F12		Elapsed 0,00s	
Verify			
Erose			
<u></u>			
Blunk 9			
Config.			
	Ready		

Figure 76. 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 77.

💷 Select dev	ice to read
0CD_0 0CD_1 0CD_2 0CD_3	
© OCD_4 © OCD_5 © OCD_6	ОК
© OCD_7	Cancel

Figure 77. 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 78, which shows that the user has recently opened only three files; the third file (selected in the figure) is the demo.hex file.



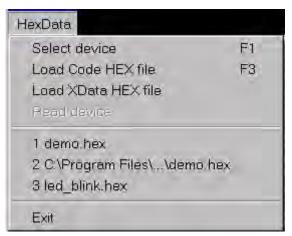


Figure 78. Most Recently Used Files

Exit

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

Program Menu

The Program menu, shown in Figure 79, 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.

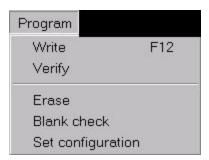


Figure 79. The OCD ISP's Program Menu



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 80 for an example.

Configuration dialog	×
	Boot area
🔲 Lock code area	▼ 256:0100~01FF
🔽 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	p .
	00 00 00 OK

Figure 80. Configuration Dialog

Window Menu

The Window menu, shown in Figure 81, can be used to modify the arrangement of child windows or to directly select a child window.



I

	Open CODE dump
	Open XDATA dump
	Cascade
	Tile
	Close
\checkmark	1 CODE address : 00000000
	2 XDATA address : 0000004D

Figure 81. 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 82. 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												×
Edit O	κ)	1	2	3	4	5	ł	5	7	8	9	A	E	} (¢	D	Е	F	
000000	02	01	03	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF	1			
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				
000090	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	1			
000000	FF	FF	FF	32	FF	FF	FF	FF	FF	FF	FF	32	FF	FF	FF	FF				
OBOODED	FF	FF	FF	02	12	C4	FF	FF	FF	FF	FF	02	13	00	FF	FF				
0000070	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	B 8	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	A8	53	07	7F	8F	A8	-			-

Figure 82. Open CODE Dump Child Window



Open XData Dump

I

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 83. If this window is already open, the window will move to the top-most level of the debugger frame.

Edit N	0)	1	2	3	4	5	Ð	5	7	8	9	A	E	} (2	D	Е	F	
000000	00	00	00	00	00	00	00	00	0.0	00	00	00	00	00	00	00				
000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000020	00	00	00	00	00	00	00	00	00	00	00	0.0	00	00	00	00				
000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000040	00	00	00	00	0.0	00	00	00	00	00	00	00	00	00	00	00				
000050		00	00	00	10.0	00	00	00	00	00	00	00	00	00	00	00				
000060		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000070	10.0	00	00	00	00	00	00	00	100	00	00	00	00	00	00	00				
000080		00	00	00	00	12.2	00	00	00	00	00	00	00	00	00	00				
000090		00		00	00	2.2	00	00	00	00	00	00	00	00	00	00				
0A0000	10.0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0000B0	10.0	00	00	00	00		00	00	00	00	00	00	00	00	00	00				
000000	22	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000000		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0000E0		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
0000F0	10.01	00	00	00	00		00	00	00	00	00	00	00	00	00	00				
000100		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000110	10.0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000120		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000130		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000140	2.2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				
000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00				

Figure 83. Open XData Dump Child Window



I

Cascade

Selecting **Cascade** from the **Window** menu arranges opened child windows in a stepped (cascading) visual display, as shown in Figure 84.

Contraction of the local division of the loc		, adili	esr	- 0	00A	aoc	10												×	4		
Edit N	0 0	1 1	2	3	4	5	6	7		8	9	A	В	C	Ę)	E	F				
000000	00	Co	DDE	ad	dro		00	nnr	100	n												×
000010	00		-	au	ure	55.	00	000	000	U	_			_							_	
000020	00	Edit	OK	0	1	2	3	4	5	i 1	3	7	8	9	A	E	3 (2	D	E	F	
000030	00	00000	0 FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000040	00	00001		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				-
000060	00	00002	0 FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	h			_
000070	00	00003		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1			
000080	00	00004		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1			
000090	00	00005		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
0A0000	00	00006		FF	FF	FF	FF	FF	FF	FF FF	FF	FF	FF	FF	FF	FF	FF	FF FF	1			
0000B0	00	00007		r r FF	rr FF	FF	FF	FF	FF	FF	FF	FF	rr FF	FF	FF	FF	FF	FF				
000000	00	000009		TT	TT	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
0000000	00	0000A		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
0000E0 0000F0	00	0000B	0 FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000100	00	00000	O FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000110	00	00000	0 FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000120	00	0000E	0 FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000130	00	0000F		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	1			
000140	00	00010	2.2	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000150	00	00011	-	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				
000160	00	00012		rr FF	rr FF	FF	FF	FF	rr FF	FF	FF	FF	FF	FF	FF	r r FF	FF	FF FF				
least so	-	00013		11	11	FF	FF	FF	TT	FF	FF	TT	TT TT	FF	FF	FF	TT	FF				
		00015		FF	FF	FF	FF	FF	FF	FF	FF	FF	TT	FF	FF	FF	FF	FF				
		00016	-		FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF				-

Figure 84. Cascading Child Windows



Tile

Selecting **Tile** from the **Window** menu arranges opened child windows in a partitioned (tiled) display, as shown in Figure 85.

CODE address : 00000000	XIXIATA address 1000000	×
Edit OK 0 1 2 3 4 5 6 7 8 9 A B C D	E Edit NO 0 1 2 3 4 5 6 7 8 9 A B C D E	E
000000 FF		
000010 FF		and the
000020 FF		
000030 FF		
000040 FF		
000050 FF		
000060 FF		
000070 FF		
000080 FF		
000090 FF		
OOOOAO FF	00 00 00 00 00 00 00 00 00 00 00 00 00	
OOOOBO FF	00 00 00 00 00 00 00 00 00 00 00 00 00	
OOOOCO FF		
OOOODO FF		
OOOOEO FF		
ODOOFO FF	0000F0 00 00 00 00 00 00 00 00 00 00 00	
000100 FF F		
000110 FF		
000120 FF		
000130 FF		
000140 FF		
000150 FF		
000160 FF		
000170 FF		
000180 FF	000180 00 00 00 00 00 00 00 00 00 00 00 00 0	
000190 FF		
0001A0 FF	00 00 00 00 00 00 00 00 00 00 00 00 00	
0001B0 FF	0001B0 00 00 00 00 00 00 00 00 00 00 00 00 0	
OOO1CO FF	000100 00 00 00 00 00 00 00 00 00 00 00	
0001D0 FF		-
निय यस यस समायस यस यस यस समायस समाय समाय त्र समाय त्र समाय ति	Ling on ou	1877

Figure 85. 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 81 on page 70, 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 86.

C co	DE	ad	dre	ss :	00	000	150	1												x
Edit O	K	3	1	2	3	4	5	1	3	7	8	9	A	E	3)	C	D	E	F	
0004A0	07	CO	06	CO	05	CO	04	CO	03	CO	02	12	0E	EB	90	17				
0004B0	A7	75	FO	80	12	0E	3B	DO	02	DO	03	DO	04	DO	05	DO				
000400	06	DO	07	0A	DO	07	DO	06	80	AE	90	17	90	75	FO	80				
000400	CO	07	CO	06	CO	05	12	0E	3B	DO	0.5	DO	06	DO	07	74				
0004E0	10	2E	FE	E4	3F	FF	OD	02	04	ЗD	22	AF	83	E5	82	90				
0004F0	00	OF	FO	A3	EF	FO	12	03	28	90	00	0F	EO	FE	A3	ΕO				
000500	88	74	FO	5E	FC	74	01	5F	FD	CO	07	CO	06	CO	05	CO				
000510	04	12	03	48	DO	04	DO	05	80	82	8D	83	12	03	52	12				
000520	03	00	12	03	48	DO	06	DO	07	8E	OB	74	01	5F	F 5	0C				
000530	90	00	OB	EO	F9	A3	ΕO	FA	A3	EO	FB	90	00	0E	ΕO	F8				
000540	75	0A	00	C3	E5	0A	64	80	88	FO	63	FO	80	95	FO	50				
000550	44	CO	00	E 5	0A	F8	33	95	EO	FD	E8	25	OB	F5	OD	ED				
000560	35	OC	F5	0E	90	00	8D	EO	FC	A3	EO	FD	E5	OD	20	F5				
000570	OD	E5	0E	ЗD	F5	0E	E.5	0A	29	F8	E4	ЗA	FC	8B	0.5	88				
000580	82	80	83	8D	FO	12	15	Al	85	OD	82	85	0E	83	FO	05				
000590	0A		00	80	AE	8E	82	8F	83	12	03	7A	12	03	00	12				
0005A0	03	2F	22	AF	83	E5	2.7	90	00	12	FO	A3	EF	FO	12	03				
0005B0	28	90	00	12	EO	FE	A3	EO	FF	74	FO	5E	FC	74	01	5F				
000500	FD	8E	02	74		5F	FB	CO	07	CO	06	CO	05	CO	04	10.5				
0005D0	03	CO	02	12	03	48	DO	02	DO	03	DO	04	DO	05	80	82				
0005E0	8D	83	CO	03	CO	02	12	03	52	12	03	00	12	03	48	DO				
0005F0	02	DO	03	DO	06	DO	07	90	00	8D	EO	FC	A3	EO	FD	EA				
000600	20	FA	EB	ЗD	FB	90	00	11	EO	88	82	8B	83	FO	8E	82				

Figure 86. 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 87. Editing and cursor functions are the same for the XDATA Dump window as they are for the Code Dump window.

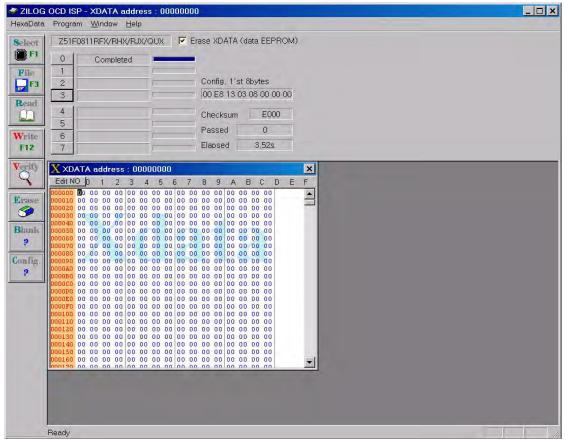


Figure 87. XData Dump Child Window



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. Refer to <u>Figure 2</u> on page 4 for guidance.

- 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 88. Click Locate and install driver software (recommended).



Figure 88. 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 you see the Installing Device Driver dialog shown in Figure 89, do not click **Close**. Instead, wait until you see the dialog that follows, which is shown in Figure 90.

U Driver Software Installation		x
Installing device driver software		
Unidentified Device	Searching Windows Update	
		Close

Figure 89. Install Device Driver Dialog, Windows Vista



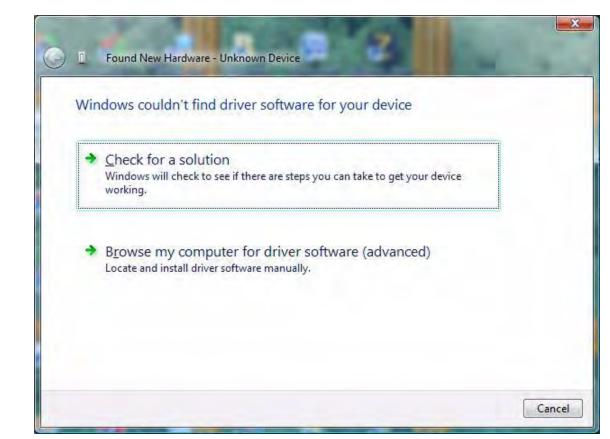


Figure 90. Couldn't Find Driver Dialog, Windows Vista

- 5. Select **Browse my computer for driver software (advanced)** to display the dialog shown in Figure 91, 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>\device drivers\OCD USB\x32

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

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



Found New Hardware - Unknown Device Browse for driver software on your computer	
Search for driver software in this location:	
C:\Program Files\ZiLOG\Z8051_1.0\device drivers\OCD USB\x32	Browse
	Next Cance

Figure 91. Browse For Driver Dialog, Windows Vista

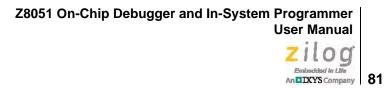
I



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



Figure 92. Can't Verify Publisher Dialog, Windows Vista



7. When the installation is complete, the screen shown in Figure 93 will appear. Click **Close** to exit the OCD driver installation.

Generation International State - ZILOG Z8051 USB OCD - No Firmware	×
The software for this device has been successfully installed	
Windows has finished installing the driver software for this device:	
ZILOG Z8051 USB OCD - No Firmware	
	Close

Figure 93. Successfully Installed Dialog, Windows Vista



Appendix B. OCD Driver Installation on Windows 7 Systems

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.

- 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 94. Click within this dialog to display the installation sequence of the driver software, which is diagrammed from top to bottom in Figure 95.



Figure 94. Install Device Driver Dialog, Windows 7



Installing device driver software	
ZILOG OCD I/F	Searching Windows Update
Obtaining device driver software from Windo Skip obtaining driver software from Window	
Skip obtaining driver software norm window	sopuate
	Close
Driver Software Installation	
Installing device driver software	
ZILOG OCD I/F	Searching preconfigured driver folders
	Close
Driver Software Installation	
Installing device driver software	
	2
ZILOG OCD I/F	OInstalling driver software
	Close
	Close
Driver Software Installation	
ZILOG Z8051 USB OCD installed	
ZILOG Z8051 USB OCD	Ready to use
	Close

Figure 95. Driver Software Installation Dialog, Windows 7

I



3. If *Zilog Z8051 USB OCD* appears in the Device Manager (as highlighted in Figure 96), the OCD driver software has been successfully installed.

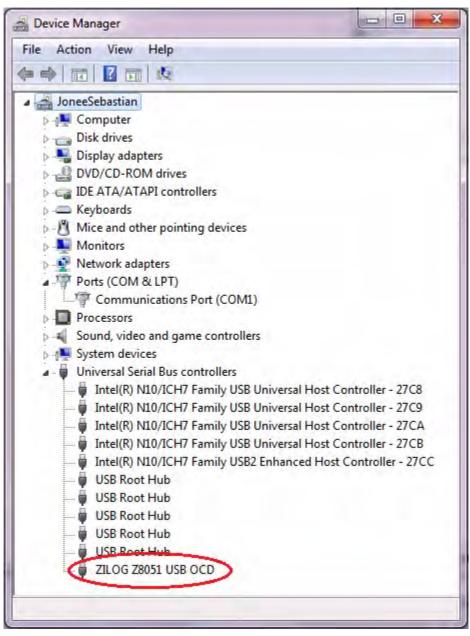


Figure 96. Device Manager Dialog, Windows 7



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>.



Customer Support