



***Z8051™ Family of 8-Bit Microcontrollers***

# **Z8051 On-Chip Debugger and In-System Programmer**

**User Manual**

UM024001-0312





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

---

## LIFE SUPPORT POLICY

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

### As used herein

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

### Document Disclaimer

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concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded.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

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

# Revision History

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

Date	Revision Level	Description	Page No
Mar 2012	01	Original issue.	All

# Table of Contents

Revision History .....	iii
List of Figures .....	vi
Introduction .....	1
The Z8051 On-Chip Debugger .....	2
Features .....	3
Install the Z8051 OCD Software and Documentation .....	3
Z8051 OCD Driver Installation .....	4
FTDI USB-to-UART Driver Installation .....	7
Understanding the OCD Menu Functions .....	9
File Menu .....	9
Emulation Menu .....	12
Break/Configure Menu .....	14
View Menu .....	18
Window Menu .....	28
Child Windows .....	31
Z8051 Basic Registers Window .....	31
Code Disassemble Window .....	34
Code Dump Window .....	37
XDATA Dump Window .....	41
IRAM Dump Window .....	45
SFR Dump Window .....	47
Watch Global Window .....	49
Watch Local Window .....	51
Text File Window .....	54
The Z8051 OCD In-System Programmer .....	61
Features .....	62
Connect the Hardware .....	62
Apply Power .....	63
Understanding the OCD ISP Menu Functions .....	63
HexData Menu .....	63
Program Menu .....	70

Window Menu .....	72
Child Windows .....	76
Code Dump Window .....	77
XData Dump Window .....	78
Appendix A. OCD Driver Installation on Windows Vista Systems .....	79
Appendix B. OCD Driver Installation on Windows 7 Systems .....	85
Customer Support .....	90

# List of Figures

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

Figure 34.	Using the Basic Registers Function, #5 of 6	33
Figure 35.	Using the Basic Registers Function, #6 of 6	34
Figure 36.	Using the Code Disassembler Function, #1 of 3	35
Figure 37.	Using the Code Disassembler Function, #2 of 3	35
Figure 38.	Using the Code Assembler Function, #3 of 3	36
Figure 39.	Using the PC Break Toggle Function	37
Figure 40.	Using the Code Dump Function, #1 of 2	38
Figure 41.	Using the Code Dump Function, #2 of 2	39
Figure 42.	The Code Dump Function's Goto/Input Dialog	40
Figure 43.	The Code Dump Function's Pattern Load Dialog	40
Figure 44.	The Code Dump Function's Pattern Save Dialog	41
Figure 45.	The Code Dump Function's Pattern Fill Dialog	41
Figure 46.	Using the XDATA Dump Function, #1 of 2	42
Figure 47.	Using the XDATA Dump Function, #2 of 2	43
Figure 48.	The XDATA Dump Function's Goto/Input Dialog	43
Figure 49.	The XDATA Dump Function's Pattern Load Dialog	44
Figure 50.	The XDATA Dump Function's Pattern Save Dialog	44
Figure 51.	The XDATA Dump Function's Pattern Fill Dialog	45
Figure 52.	Using the IRAM Dump Function, #1 of 2	46
Figure 53.	Using the IRAM Dump Function, #2 of 2	47
Figure 54.	Using the SFR Dump Function, #1 of 3	48
Figure 55.	Using the SFR Dump Function, #2 of 3	48
Figure 56.	Using the SFR Dump Function, #3 of 3	49
Figure 57.	The Watch Global Function's Global Variables Dialog	50
Figure 58.	Adding A Global Symbol	50
Figure 59.	Editing A Global Symbol	51
Figure 60.	The Watch Local Function Dialog	52
Figure 61.	Editing A Local Symbol	52
Figure 62.	Example Watch Local Function, #1 of 2	53
Figure 63.	Example Watch Local Function, #2 of 2	54
Figure 64.	Using the Text File Function, #1 of 5	55
Figure 65.	Using the Text File Function, #2 of 5	55
Figure 66.	Using the Text File Function, #3 of 5	56
Figure 67.	Text File Child Window	57
Figure 68.	Code Disassembler Child Window	58
Figure 69.	Using the Text File Function, #4 of 5	59

Figure 70.	Using the Text File Function, #5 of 5	60
Figure 71.	Example On-Chip Debugger ISP Screen	61
Figure 72.	OCD Hardware ISP Pin Assignment (Bottom View)	62
Figure 73.	The OCD ISP's File Menu	64
Figure 74.	Device Select Dialog	65
Figure 75.	Fill Buffer Dialog	66
Figure 76.	File Open Dialog	67
Figure 77.	OCD ISP Dialog	68
Figure 78.	Select Device To Read Dialog	69
Figure 79.	Most Recently Used Files	70
Figure 80.	The OCD ISP's Program Menu	70
Figure 81.	Configuration Dialog	72
Figure 82.	The OCD ISP's Window Menu	73
Figure 83.	Open CODE Dump Child Window	73
Figure 84.	Open XData Dump Child Window	74
Figure 85.	Cascading Child Windows	75
Figure 86.	Tiled Child Windows	76
Figure 87.	CODE Dump Child Window	77
Figure 88.	XData Dump Child Window	78
Figure 89.	Found New Hardware Dialog, Windows Vista	79
Figure 90.	Install Device Driver Dialog, Windows Vista	80
Figure 91.	Couldn't Find Driver Dialog, Windows Vista	81
Figure 92.	Browse For Driver Dialog, Windows Vista	82
Figure 93.	Can't Verify Publisher Dialog, Windows Vista	83
Figure 94.	Successfully Installed Dialog, Windows Vista	84
Figure 95.	Install Device Driver Dialog, Windows 7	85
Figure 96.	Driver Software Installation Dialog, Windows 7	86
Figure 97.	Unsuccessful Installation, Scenario 1	87
Figure 98.	Unsuccessful Installation, Scenario 2	88
Figure 99.	Device Manager Dialog, Windows 7	89



# 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.

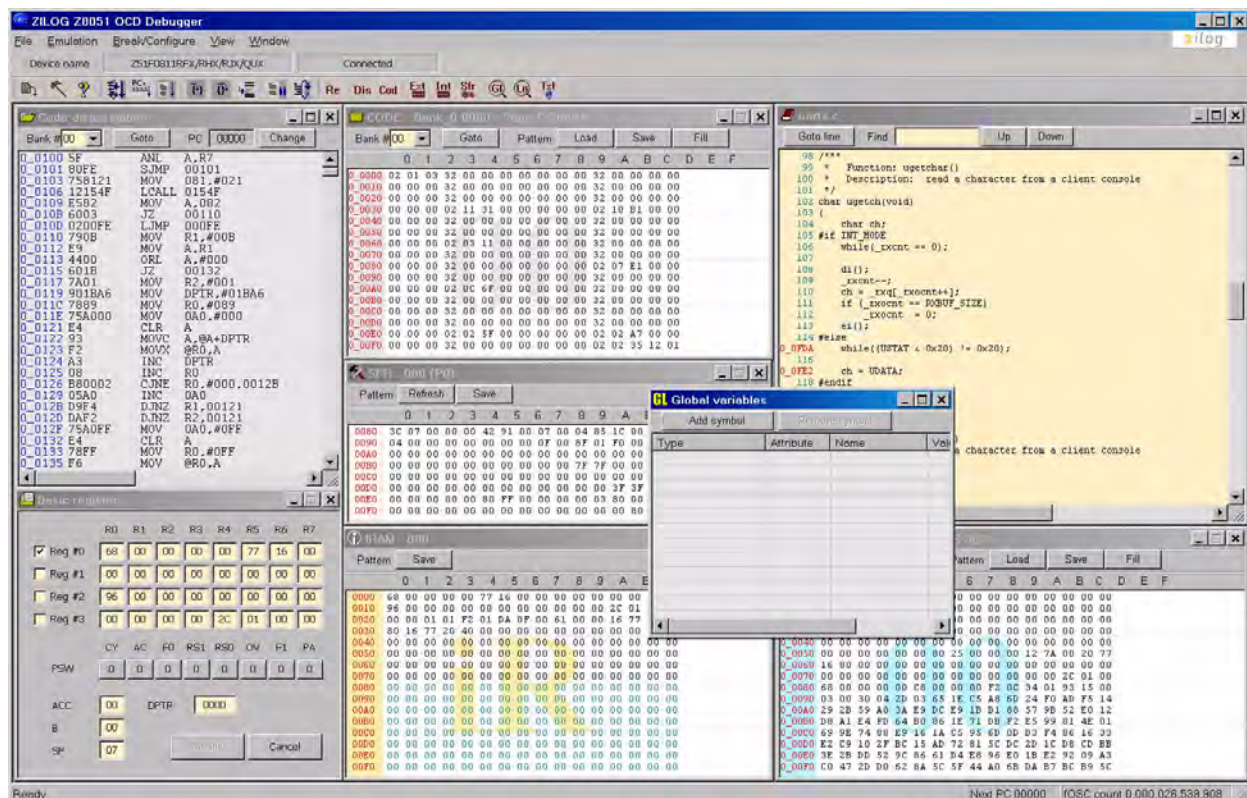


Figure 1. On-Chip Debugger Screen

## Features

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

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

## Install the Z8051 OCD Software and Documentation

The Z8051 On-Chip Debugger (OCD) interface is the interface by which your PC will communicate with the Z51F0811 MCU to download and execute code. In addition to the OCD, software such as development tools and sample applications will be installed.

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.

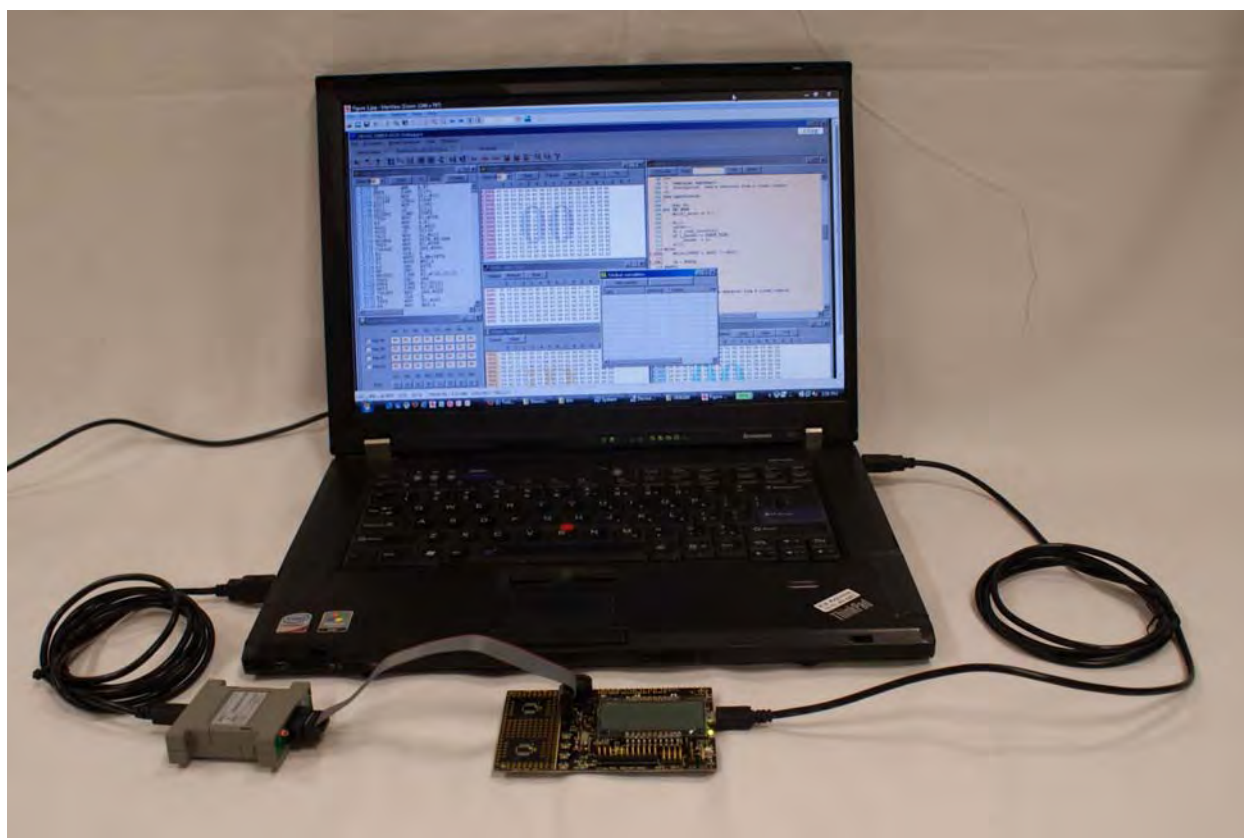


Figure 2. An Example Setup

---

► **Note:** If you are running Windows Vista, see [Appendix A](#) on page 79 to install your device drivers. If you are running Windows 7, see [Appendix B](#) on page 85.

---

1. Connect the OCD hardware to the USB port of your PC by connecting the A-Male end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port, 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**.

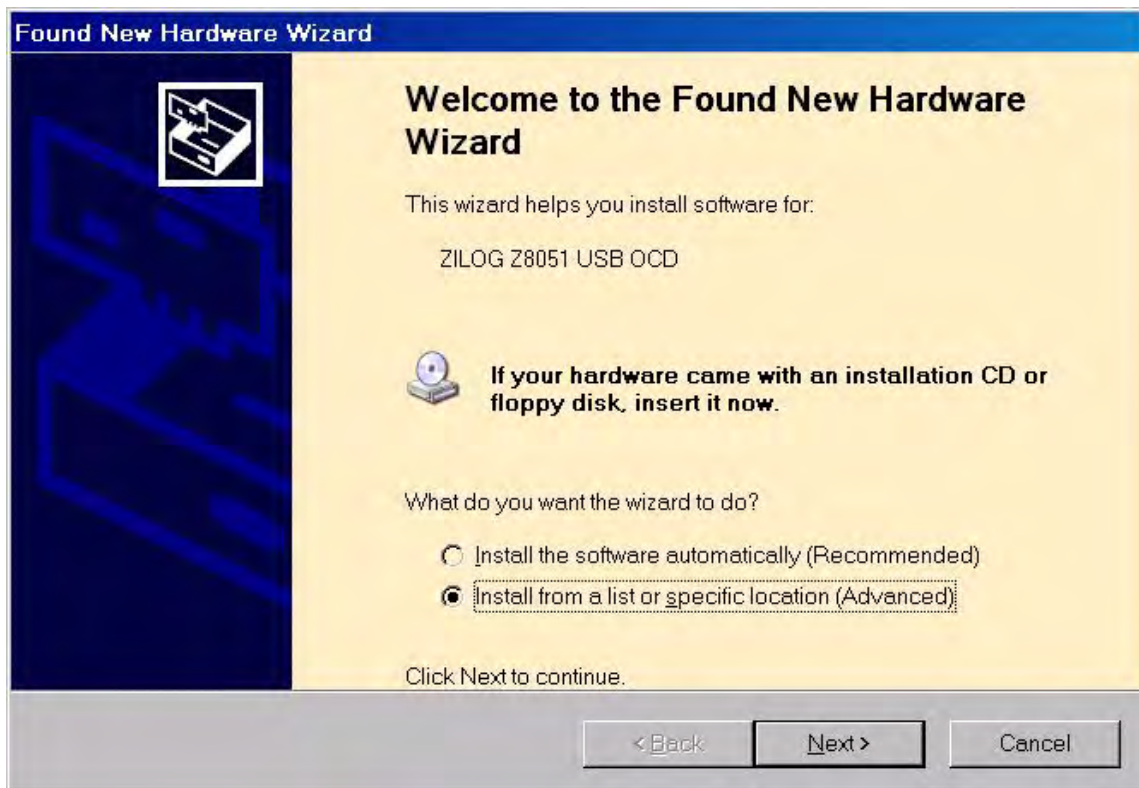
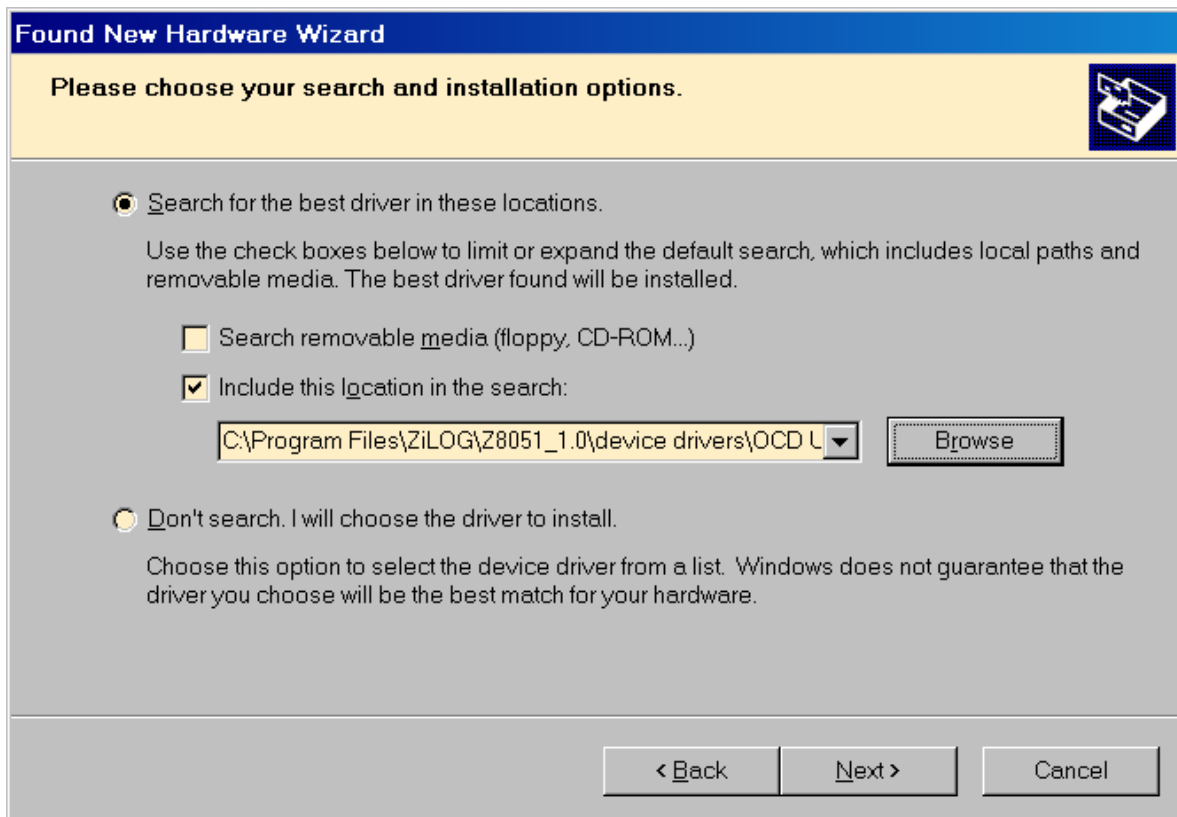


Figure 3. The Found New Hardware Wizard Welcome Screen

3. The next dialog box, shown in Figure 4, prompts you to enter a path or navigate to the directory in which the .inf file was installed. Depending on the type of computer you use (32-bit or 64-bit), use the **Browse** button to navigate to one of the following paths and click the **Next** button, leaving all other selections at their default settings.
  - On 32-bit machines, use the following path:  
<Z8051 Installation>\Z8051\_<version>\device drivers\OCD USB\x32
  - On 64-bit machines, use the following path:  
<Z8051 Installation>\Z8051\_<version>\device drivers\OCD USB\x64





**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**.

## FTDI USB-to-UART Driver Installation

An FTDI USB-to-UART driver is required to allow your PC to communicate through its USB port to the on-chip UART of the Z51F0811 MCU. Observe the following procedure to perform these connections.

1. Ensure that the USB cable is not plugged in to the Z51F0811 Evaluation Board's P1 connector.
2. Navigate to the following filepath and double-click the CDM20802\_setup.exe file to begin the driver installation.

<Z8051 Installation>\Z8051\_<version>\device drivers\FTDI Uart

3. The installation process will begin and you should observe output similar to the following messages on the screen of your PC:

```
32-bit OS detected  
<installation path>\dpinstx86.exe  
Installation driver  
FTDI CDM driver installation process completed.
```

4. When the installation is complete, plug in the Mini-B connector of the second USB cable into the Board, and the larger A connector into the USB port of your PC. Again, refer to [Figure 2](#) on page 5 for guidance.
5. If the driver installation was successful, the *Ports (COM & LPT)* section of the Device Manager will display *USB Serial Port (COMx)* or similar message, as highlighted in Figure 5.

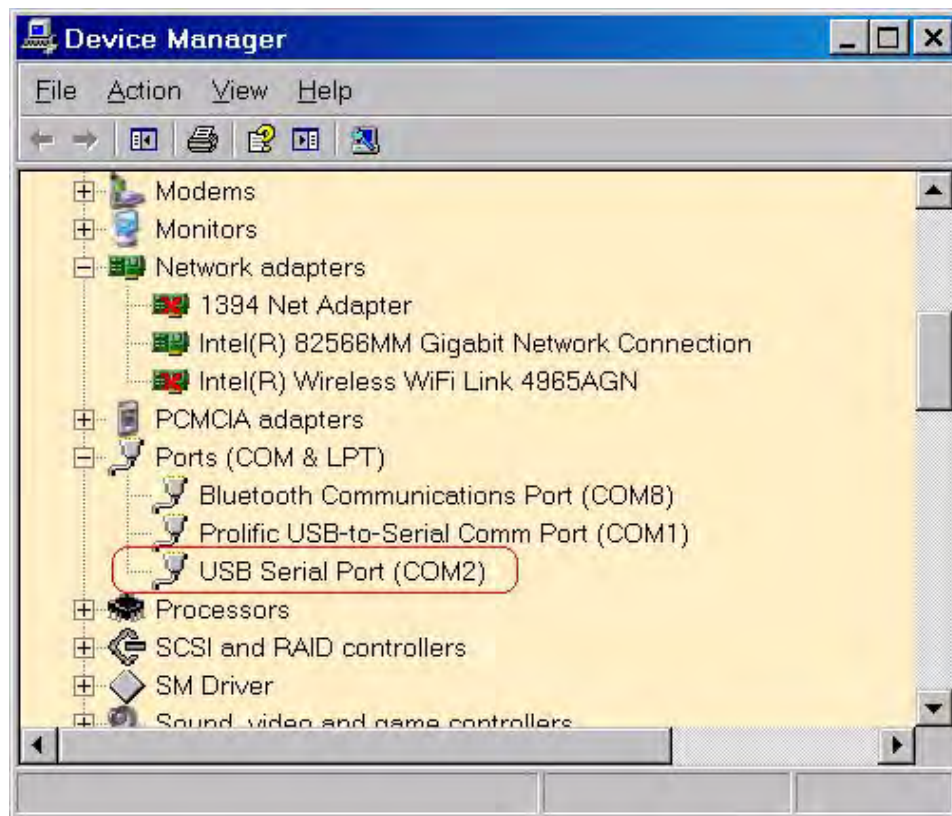


Figure 5. A Successful USB-to-UART Driver Installation



---

► **Note:** To launch the Device Manager on Windows 7 systems, launch the Start menu, enter *device manager* in the **Search programs and files** field, and press the Enter key.

To open the Device manager on earlier Windows systems, navigate via the following path:

Start → Control Panel → System → Hardware → Device Manager → Ports (COM& LPT)

---

## Understanding the OCD Menu Functions

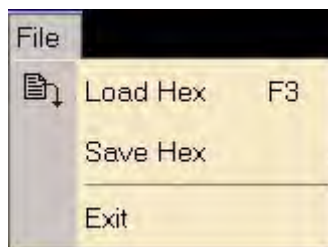
This section describes the operation of the File, Emulation, Break/Configure, View, Window menus.

### File Menu

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

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

The OCD's File menu is shown in Figure 6.



**Figure 6. The OCD's File Menu**

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

1. 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 [the Zilog website](#) 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 7.

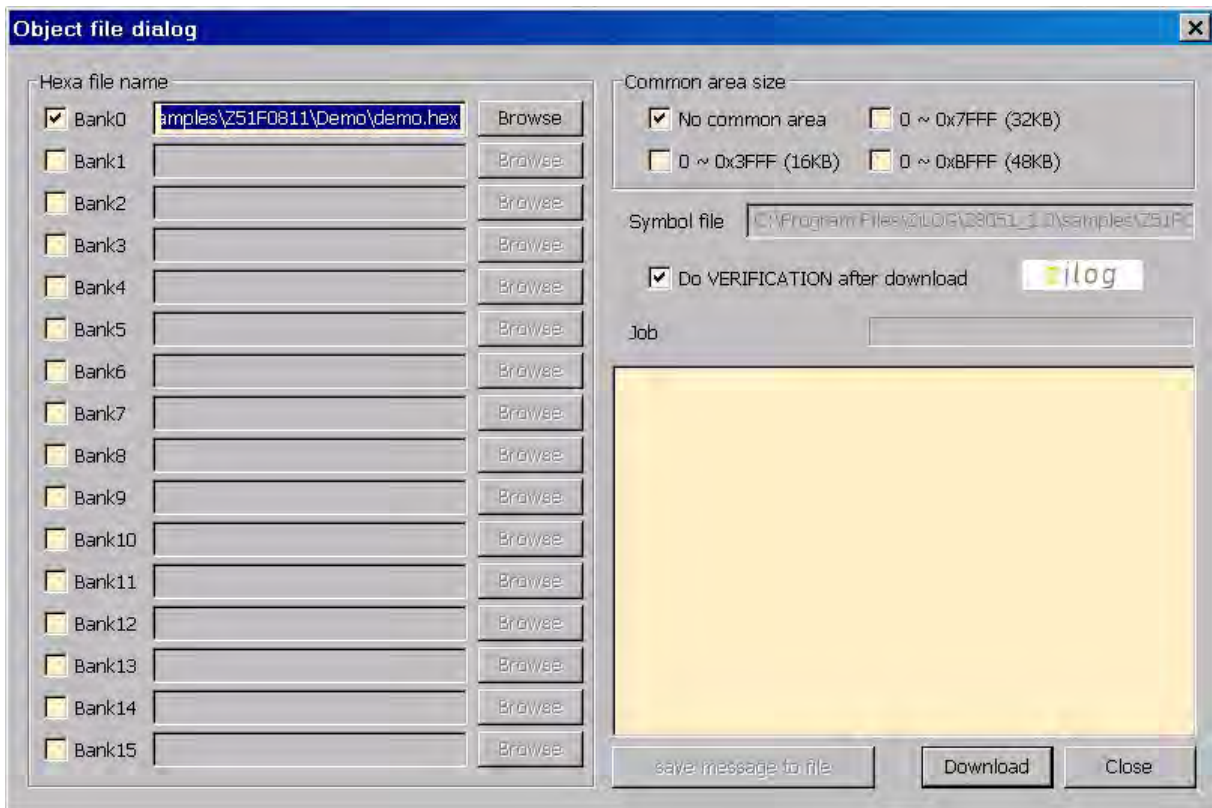
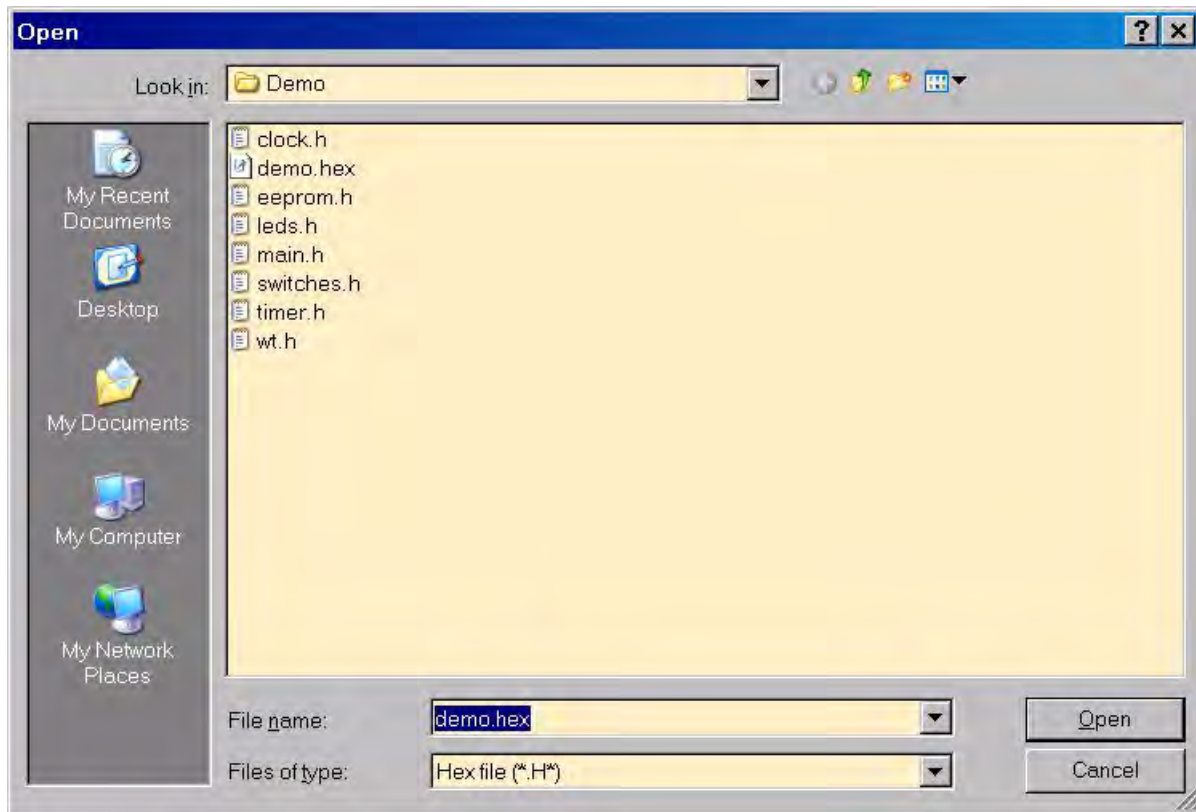
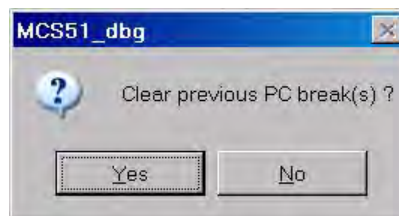


Figure 7. Object File Dialog

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 8.

**Figure 8. 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 9. Click **Yes** if you wish to remove these break-points, or **No** if you prefer to retain them.

**Figure 9. Break Debug Dialog**

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

## Emulation Menu

The Emulation menu, shown in Figure 10, lists the controls for starting or stopping an emulation routine. Use the Emulation menu to control the flow of code execution for debugging purposes.

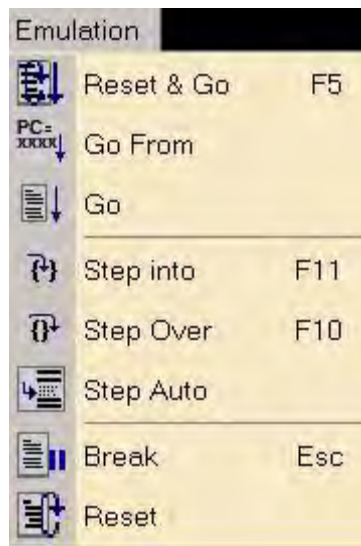


Figure 10. The OCD's Emulation Menu

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

### Reset & Go

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

## Go From

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

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

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

## Go

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

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

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

## Step

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

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

## Step Over

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

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

### Step Auto

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

### Break

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

### Reset

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

## Break/Configure Menu

The Break/Configure menu, shown in Figure 11, lists PC breakpoint control, device configuration and hardware test functions.

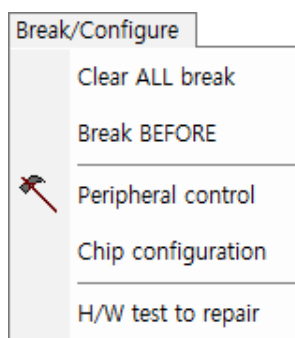


Figure 11. The OCD's Break/Configure Menu

### 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 12.



Figure 12. 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 13.

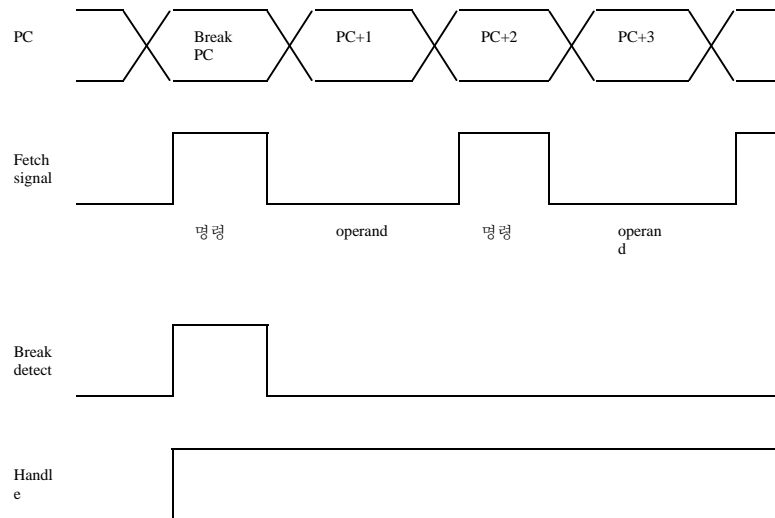
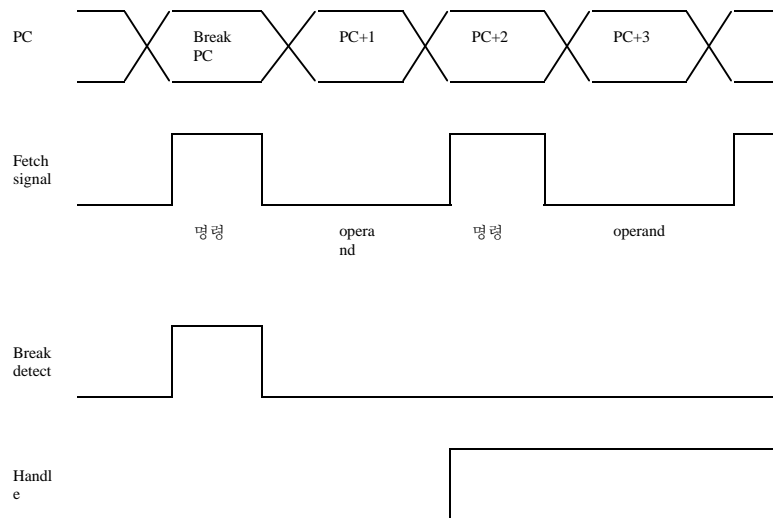


Figure 13. Break BEFORE Timing Diagram

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 14.

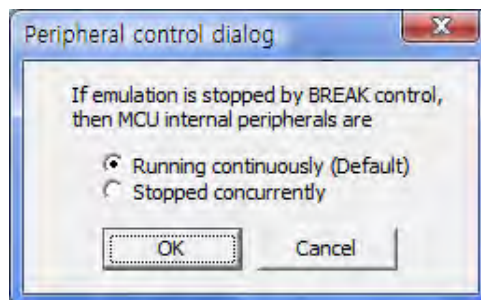


**Figure 14. Break AFTER Timing Diagram**

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



**Figure 15. Peripheral Control Dialog**

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



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

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

## Chip Configuration

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

The Configuration dialog box shown in Figure 16 offers an example configuration for the Z51F0811 device.

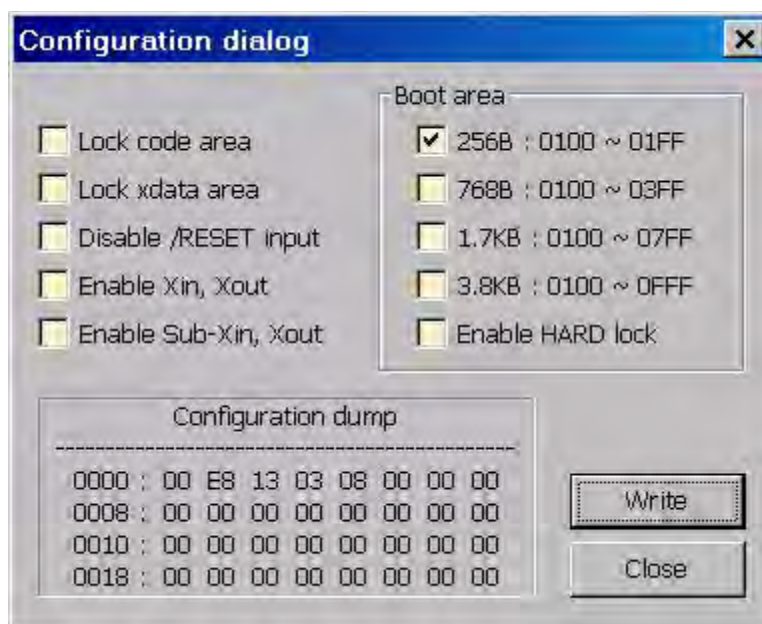


Figure 16. Z51F0811 MCU Configuration Example

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 17, 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 17. The OCD's View Menu**

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

### Toolbar

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

### Emulation Toolbar

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

### Window Open Bar

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

## Status Bar

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

## Z8051 Basic Registers

This menu selection opens a window that displays the Z8051 Series' basic registers. If this window is already open, selecting the **Z8051 Basic Registers** option will cause this window to appear at the top-most level of the debugger frame. See Figure 18.



Figure 18. 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 19.

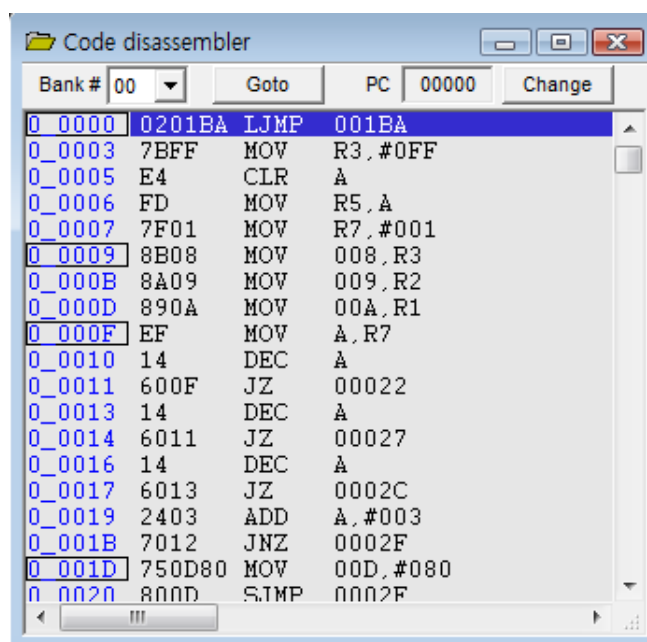


Figure 19. Code Disassembler Dialog

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 20.

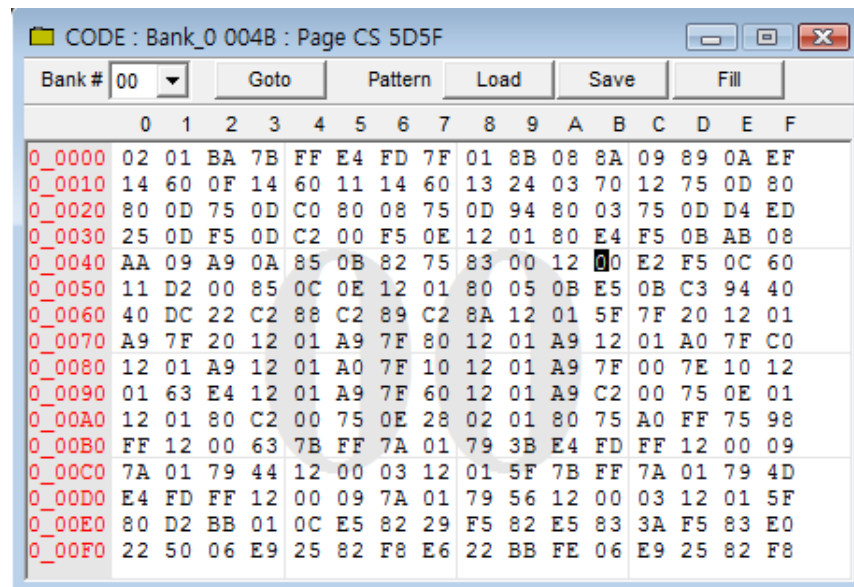


Figure 20. Code Dump Dialog

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 21.

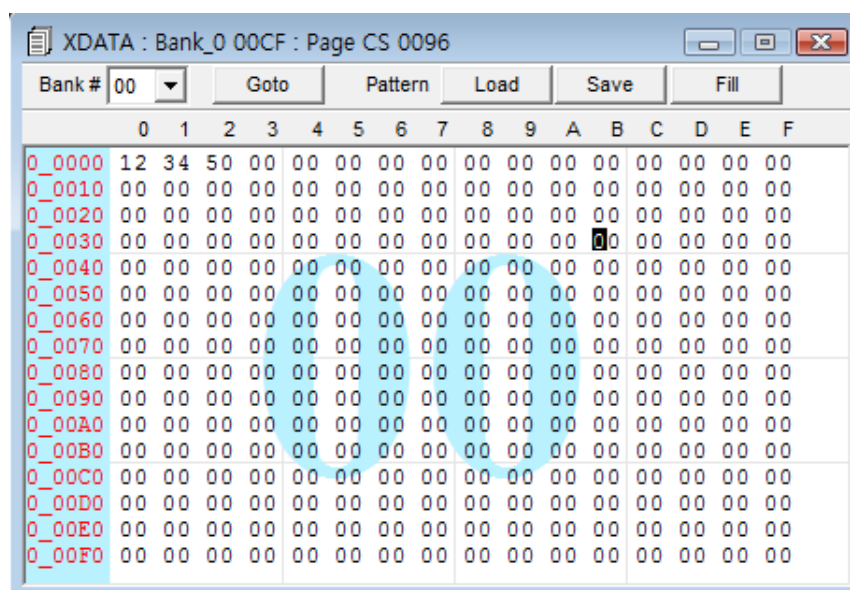
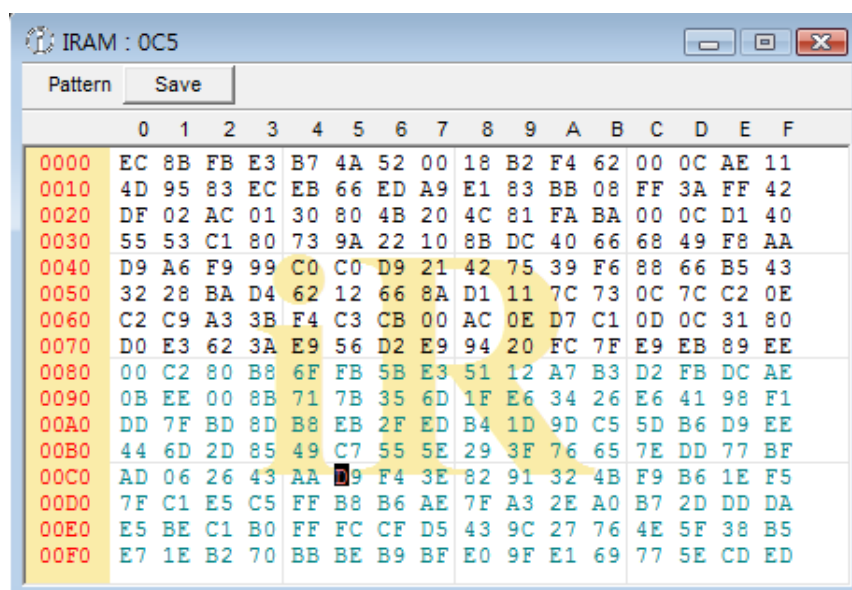


Figure 21. XDATA Dump Dialog

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 22.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	EC	8B	FB	E3	B7	4A	52	00	18	B2	F4	62	00	0C	AE	11
0010	4D	95	83	EC	EB	66	ED	A9	E1	83	BB	08	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	A6	F9	99	C0	C0	D9	21	42	75	39	F6	88	66	B5	43
0050	32	28	BA	D4	62	12	66	8A	D1	11	7C	73	0C	7C	C2	0E
0060	C2	C9	A3	3B	F4	C3	CB	00	AC	0E	D7	C1	0D	0C	31	80
0070	D0	E3	62	3A	E9	56	D2	E9	94	20	FC	7F	E9	EB	89	EE
0080	00	C2	80	B8	6F	FB	5B	E3	51	12	A7	B3	D2	FB	DC	AE
0090	0B	EE	00	8B	71	7B	35	6D	1F	E6	34	26	E6	41	98	F1
00A0	DD	7F	BD	8D	B8	EB	2F	ED	B4	1D	9D	C5	5D	B6	D9	EE
00B0	44	6D	2D	85	49	C7	55	5E	29	3F	76	65	7E	DD	77	BF
00C0	AD	06	26	43	AA	9	F4	3E	82	91	32	4B	F9	B6	1E	F5
00D0	7F	C1	E5	C5	FF	B8	B6	AE	7F	A3	2E	A0	B7	2D	DD	DA
00E0	E5	BE	C1	B0	FF	FC	CF	D5	43	9C	27	76	4E	5F	38	B5
00F0	E7	1E	B2	70	BB	BE	B9	BF	E0	9F	E1	69	77	5E	CD	ED

Figure 22. 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 23.



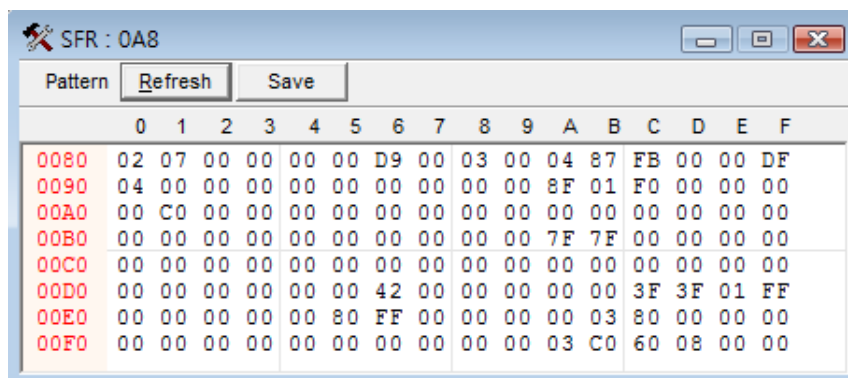


Figure 23. SFR Dump Dialog

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 24.

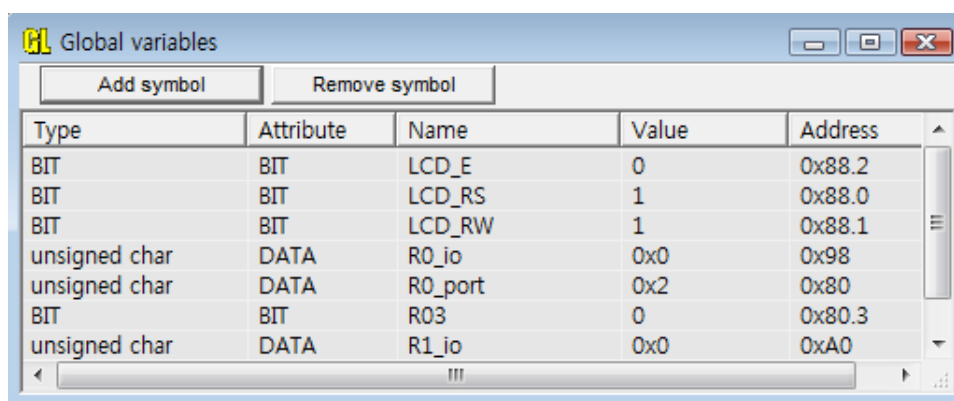
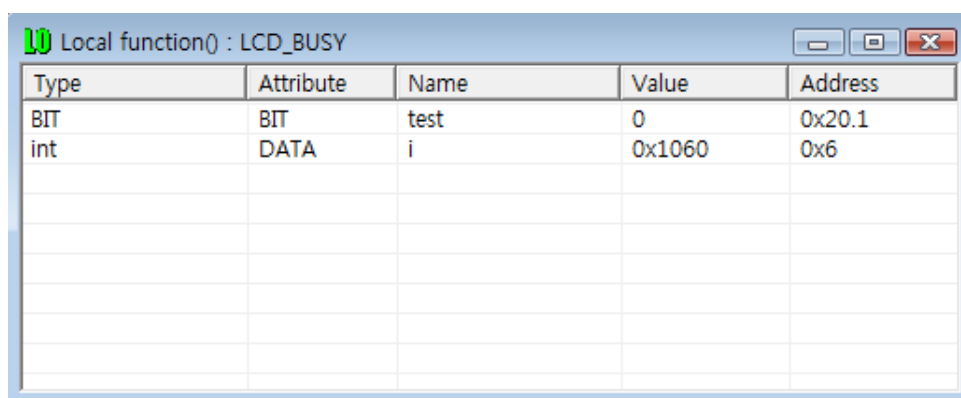


Figure 24. Global Variables Dialog

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 25.



Type	Attribute	Name	Value	Address
BIT	BIT	test	0	0x20.1
int	DATA	i	0x1060	0x6

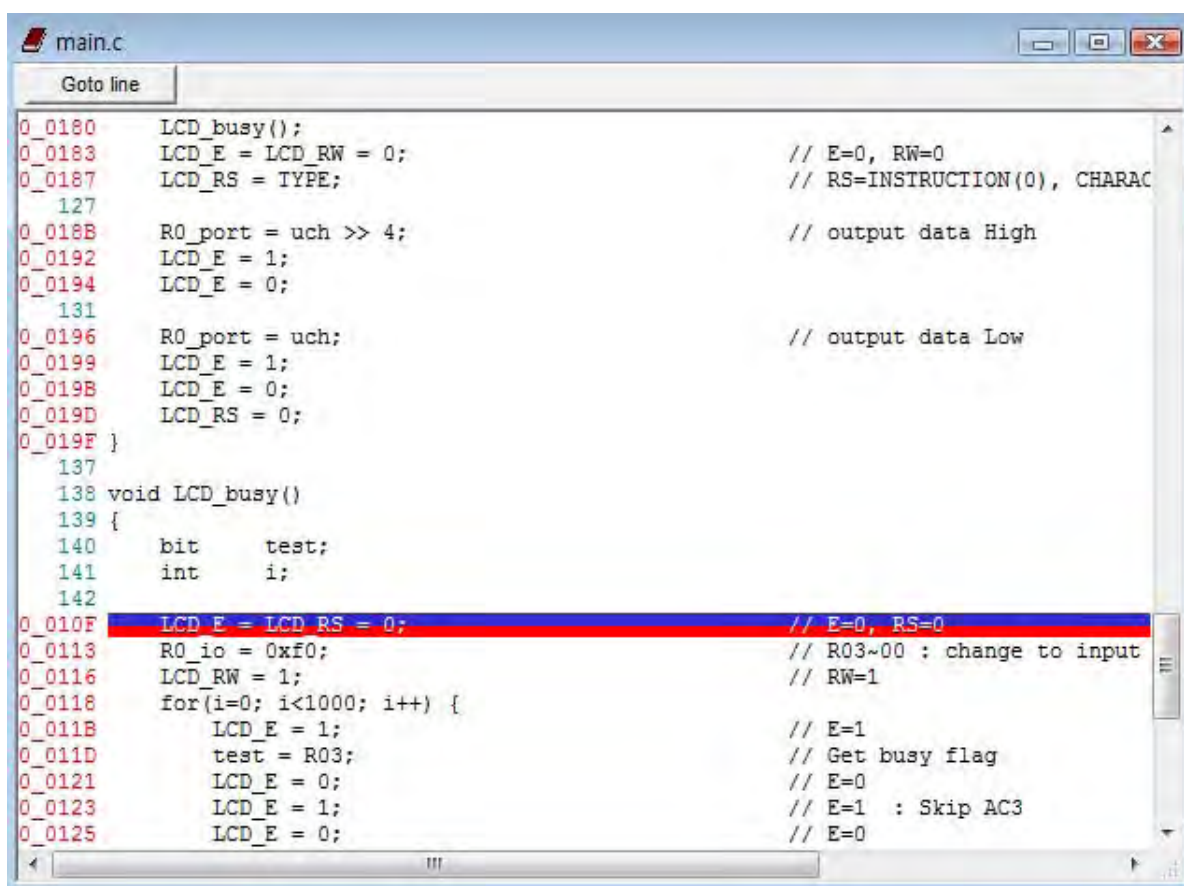
Figure 25. Local Function Dialog

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 26.



```

main.c
Goto line
0_0180 LCD_busy();
0_0183 LCD_E = LCD_RW = 0; // E=0, RW=0
0_0187 LCD_RS = TYPE; // RS=INSTRUCTION(0), CHARAC
127
0_018B R0_port = uch >> 4; // output data High
0_0192 LCD_E = 1;
0_0194 LCD_E = 0;
131
0_0196 R0_port = uch; // output data Low
0_0199 LCD_E = 1;
0_019B LCD_E = 0;
0_019D LCD_RS = 0;
0_019F }
137
138 void LCD_busy()
139 {
140     bit test;
141     int i;
142
0_010F LCD_E = LCD_RS = 0; // E=0, RS=0
0_0113 R0_io = 0xf0; // R03~00 : change to input
0_0116 LCD_RW = 1; // RW=1
0_0118 for(i=0; i<1000; i++) {
0_011B LCD_E = 1; // E=1
0_011D test = R03; // Get busy flag
0_0121 LCD_E = 0; // E=0
0_0123 LCD_E = 1; // E=1 : Skip AC3
0_0125 LCD_E = 0; // E=0
  
```

Figure 26. 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 27, can be used to modify the arrangement of child windows or to directly select a child window.

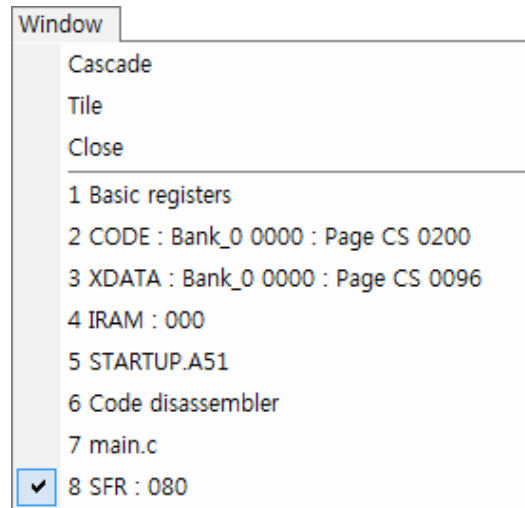


Figure 27. The OCD's Window Menu

## Cascade

This menu selection arranges opened child windows in a stepped visual sequence, as shown in Figure 28.

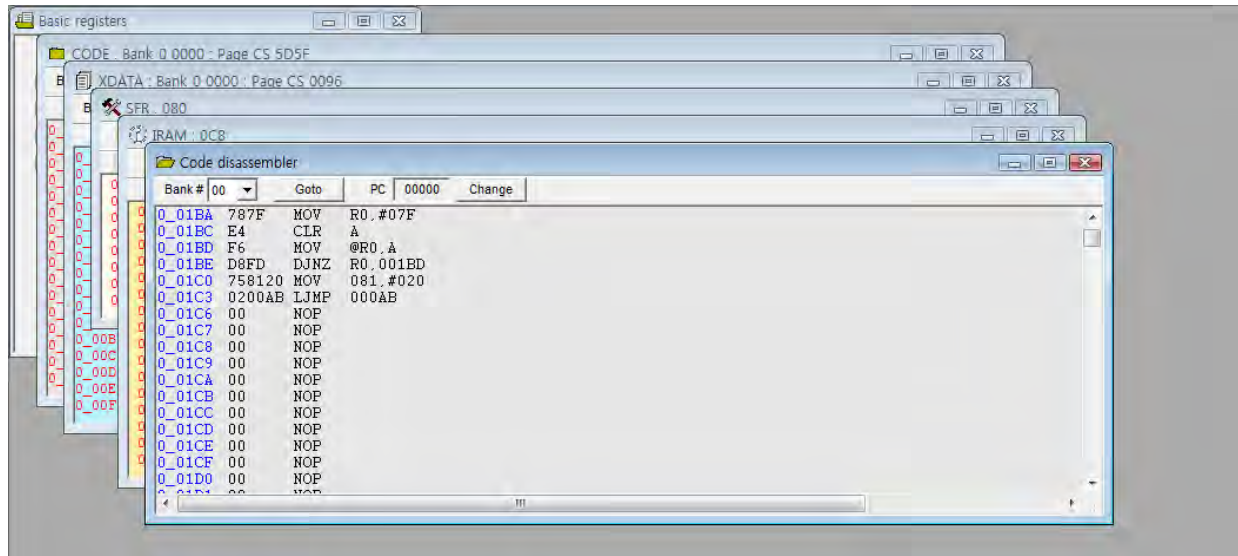


Figure 28. Cascaded Windows

## Tile

This menu selection arranges opened child windows in a partitioned display, as shown in Figure 29.

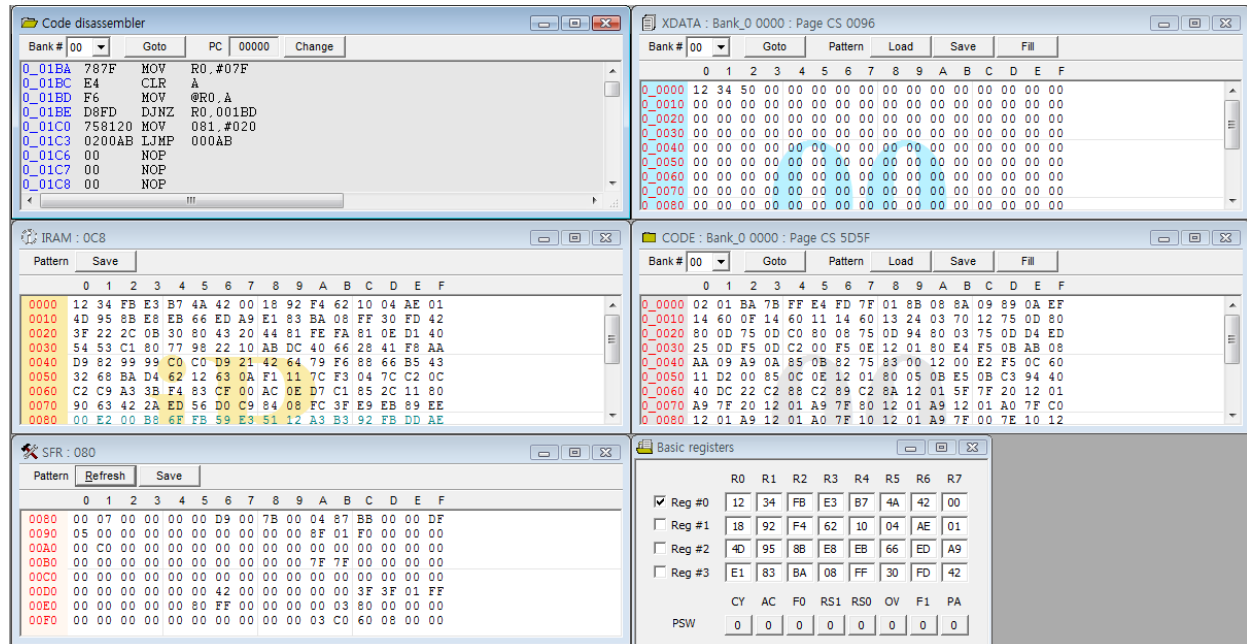


Figure 29. Tiled Windows

## 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 27](#) on page 28, 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 30 shows an example Z8051 Basic Registers window.

	R0	R1	R2	R3	R4	R5	R6	R7
<input checked="" type="checkbox"/> Reg #0	12	34	FB	E3	B7	4A	42	00
<input type="checkbox"/> Reg #1	18	92	F4	62	10	04	AE	01
<input type="checkbox"/> Reg #2	4D	95	8B	E8	EB	66	ED	A9
<input type="checkbox"/> Reg #3	E1	83	BA	08	FF	30	FD	42

	CY	AC	F0	RS1	RS0	OV	F1	PA
PSW	0	0	0	0	0	0	0	0

ACC	00	DPTR	0000
B	00		
SP	07		

Modify Cancel

Figure 30. 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 31, 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.

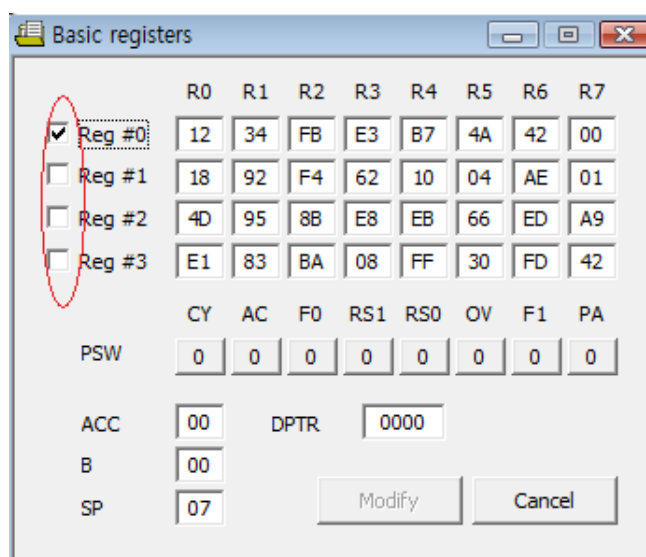


Figure 31. Using the Basic Registers Function, #2 of 6

In Figure 32, 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.

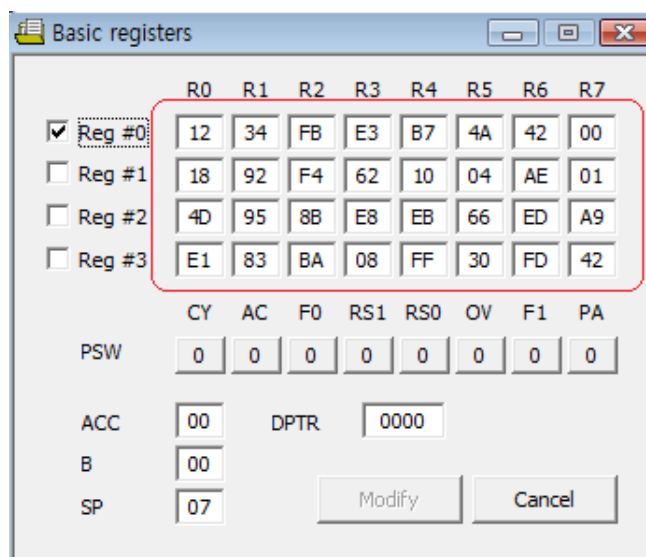


Figure 32. Using the Basic Registers Function, #3 of 6



In Figure 33, the red area highlights the Program Status Word (PSW), in which bit units can be changed.

The screenshot shows the 'Basic registers' window. It contains a table of registers R0 through R7. Below this table is the Program Status Word (PSW) section, which is highlighted with a red box. The PSW section includes fields for CY, AC, F0, RS1, RS0, OV, F1, and PA, all of which are currently set to 0. Below the PSW section are fields for ACC (00), B (00), and SP (07). At the bottom right are 'Modify' and 'Cancel' buttons.

	R0	R1	R2	R3	R4	R5	R6	R7
<input checked="" type="checkbox"/> Reg #0	12	34	FB	E3	B7	4A	42	00
<input type="checkbox"/> Reg #1	18	92	F4	62	10	04	AE	01
<input type="checkbox"/> Reg #2	4D	95	8B	E8	EB	66	ED	A9
<input type="checkbox"/> Reg #3	E1	83	BA	08	FF	30	FD	42

	CY	AC	F0	RS1	RS0	OV	F1	PA
PSW	0	0	0	0	0	0	0	0

ACC	00	DPTR	0000
B	00		
SP	07		

Figure 33. Using the Basic Registers Function, #4 of 6

In Figure 34, 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.

The screenshot shows the 'Basic registers' window. It contains a table of registers R0 through R7. Below this table is the Program Status Word (PSW) section. Below the PSW section are fields for ACC (00), B (00), and SP (07). These three fields are highlighted with a red box. At the bottom right are 'Modify' and 'Cancel' buttons.

	R0	R1	R2	R3	R4	R5	R6	R7
<input checked="" type="checkbox"/> Reg #0	12	34	FB	E3	B7	4A	42	00
<input type="checkbox"/> Reg #1	18	92	F4	62	10	04	AE	01
<input type="checkbox"/> Reg #2	4D	95	8B	E8	EB	66	ED	A9
<input type="checkbox"/> Reg #3	E1	83	BA	08	FF	30	FD	42

	CY	AC	F0	RS1	RS0	OV	F1	PA
PSW	0	0	0	0	0	0	0	0

ACC	00	DPTR	0000
B	00		
SP	07		

Figure 34. Using the Basic Registers Function, #5 of 6

In Figure 35, 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.

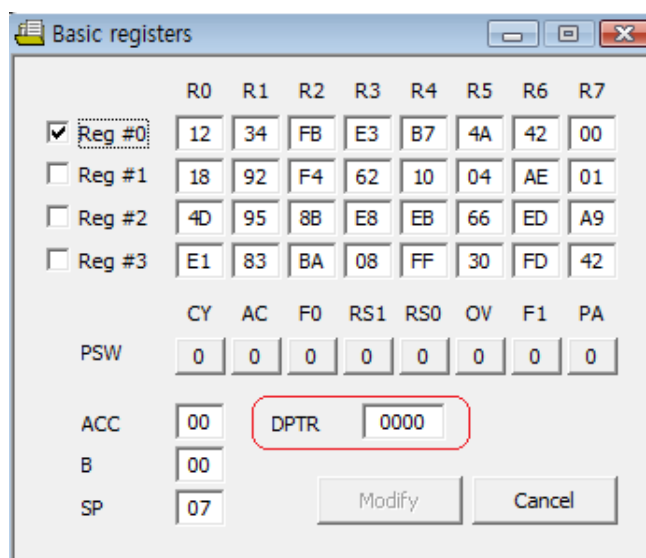
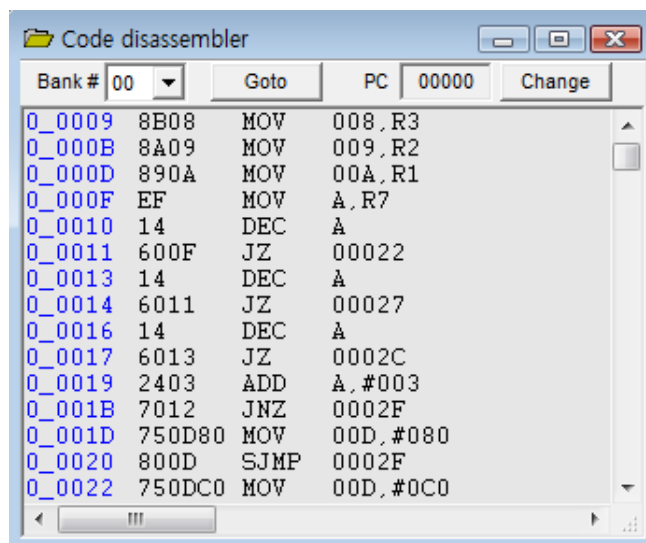


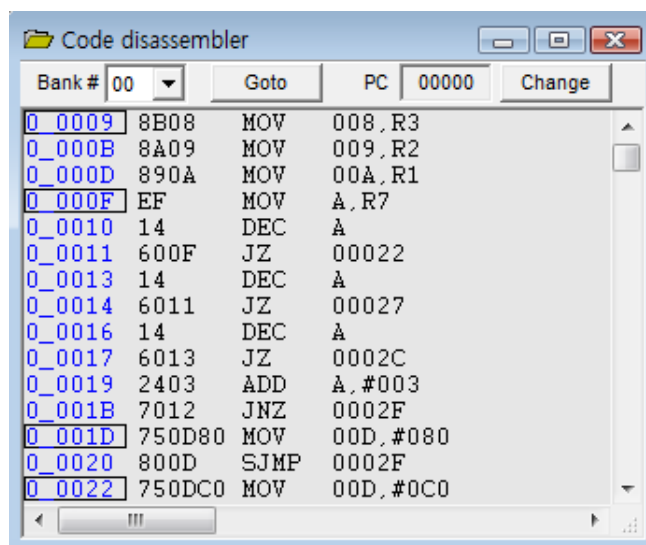
Figure 35. Using the Basic Registers Function, #6 of 6

## 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 36 shows an example Code Disassembler window.

**Figure 36. 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 37. Double-click any of these highlighted boxes to open its source file and move to the appropriate address line.

**Figure 37. Using the Code Disassembler Function, #2 of 3**

## Line Assemble

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

With your mouse, move the cursor to a line that you wish to change, and right-click to open an edit field for the contents of that line, as shown in Figure 38. Change the contents of the line by entering an instruction, operand, etc., in hexadecimal format.

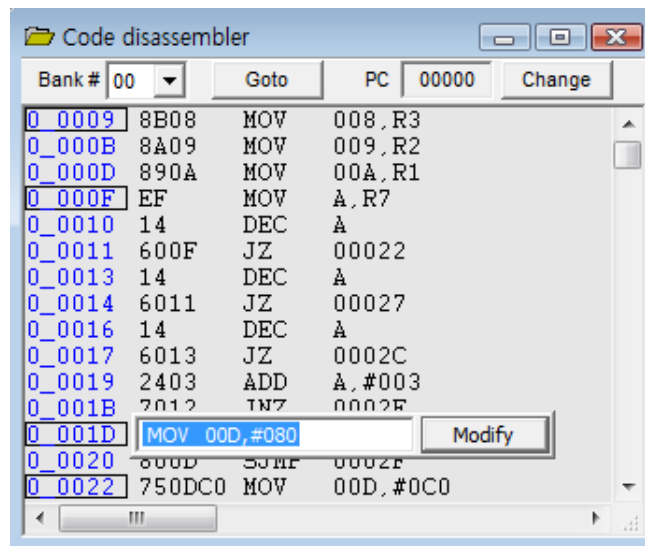
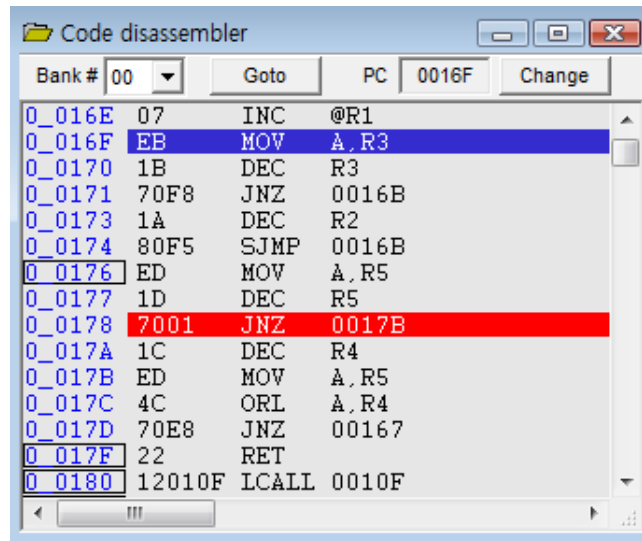


Figure 38. Using the Code Assembler Function, #3 of 3

## 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 39, the red line represents a program counter breakpoint in the line, and the blue line represents the current program counter.



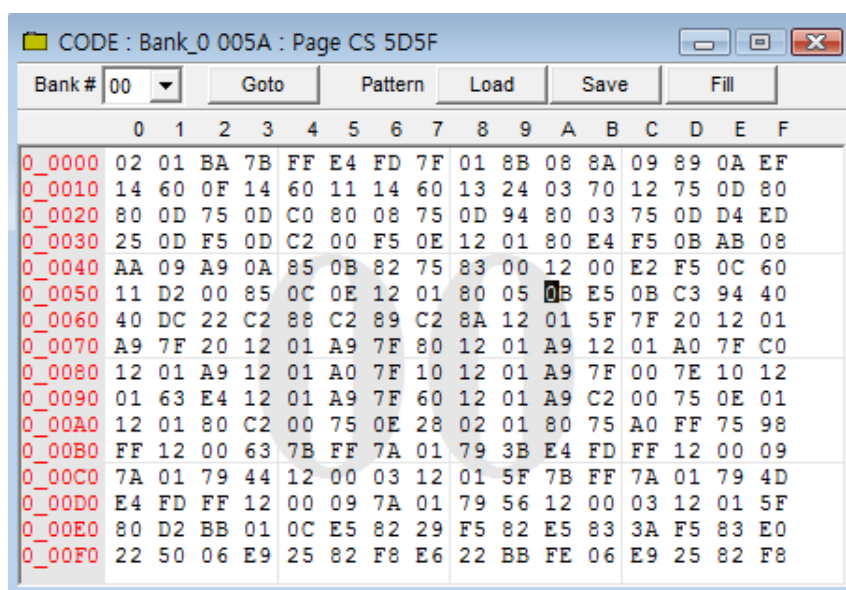
**Figure 39. Using the PC Break Toggle Function**

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 40, for example, the page number is 00.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0_0000	02	01	BA	7B	FF	E4	FD	7F	01	8B	08	8A	09	89	0A	EF
0_0010	14	60	0F	14	60	11	14	60	13	24	03	70	12	75	0D	80
0_0020	80	0D	75	0D	C0	80	08	75	0D	94	80	03	75	0D	D4	ED
0_0030	25	0D	F5	0D	C2	00	F5	0E	12	01	80	E4	F5	0B	AB	08
0_0040	AA	09	A9	0A	85	0B	82	75	83	00	12	00	E2	F5	0C	60
0_0050	11	D2	00	85	0C	0E	12	01	80	05	0B	E5	0B	C3	94	40
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5F	7F	20	12	01
0_0070	A9	7F	20	12	01	A9	7F	80	12	01	A9	12	01	A0	7F	C0
0_0080	12	01	A9	12	01	A0	7F	10	12	01	A9	7F	00	7E	10	12
0_0090	01	63	E4	12	01	A9	7F	60	12	01	A9	C2	00	75	0E	01
0_00A0	12	01	80	C2	00	75	0E	28	02	01	80	75	A0	FF	75	98
0_00B0	FF	12	00	63	7B	FF	7A	01	79	3B	E4	FD	FF	12	00	09
0_00C0	7A	01	79	44	12	00	03	12	01	5F	7B	FF	7A	01	79	4D
0_00D0	E4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5F
0_00E0	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	3A	F5	83	E0
0_00F0	22	50	06	E9	25	82	F8	E6	22	BB	FE	06	E9	25	82	F8

Figure 40. 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 41.

CODE : Bank\_0 0035 : Page CS 5D71

Bank #	00	Goto	Pattern	Load	Save	Fill										
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0_0000	02	01	BA	7B	FF	E4	FD	7F	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	0D	75	0D	C0	80	08	75	0D	94	80	03	75	0D	D4	ED
0_0030	25	AB	39	13	44	00	F5	0E	12	01	80	E4	F5	0B	AB	08
0_0040	AA	09	A9	0A	85	0B	82	75	83	00	12	00	E2	F5	0C	60
0_0050	11	D2	00	85	0C	0E	12	01	80	05	0B	E5	0B	C3	94	40
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5F	7F	20	12	01
0_0070	A9	7F	20	12	01	A9	7F	80	12	01	A9	12	01	A0	7F	C0
0_0080	12	01	A9	12	01	A0	7F	10	12	01	A9	7F	00	7E	10	12
0_0090	01	63	E4	12	01	A9	7F	60	12	01	A9	C2	00	75	0E	01
0_00A0	12	01	80	C2	00	75	0E	28	02	01	80	75	A0	FF	75	98
0_00B0	FF	12	00	63	7B	FF	7A	01	79	3B	E4	FD	FF	12	00	09
0_00C0	7A	01	79	44	12	00	03	12	01	5F	7B	FF	7A	01	79	4D
0_00D0	E4	FD	FF	12	00	09	7A	01	79	56	12	00	03	12	01	5F
0_00E0	80	D2	BB	01	0C	E5	82	29	F5	82	E5	83	3A	F5	83	E0
0_00F0	22	50	06	E9	25	82	F8	E6	22	BB	FE	06	E9	25	82	F8

Figure 41. 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 42.

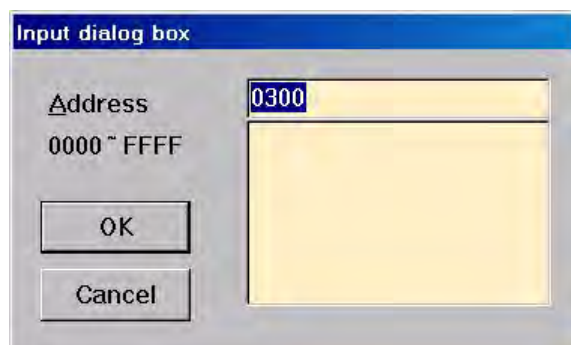


Figure 42. The Code Dump Function's Goto/Input Dialog

## 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 43.

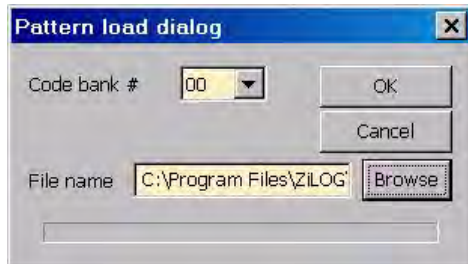


Figure 43. The Code Dump Function's Pattern Load Dialog

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 44.





Figure 44. 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 45.

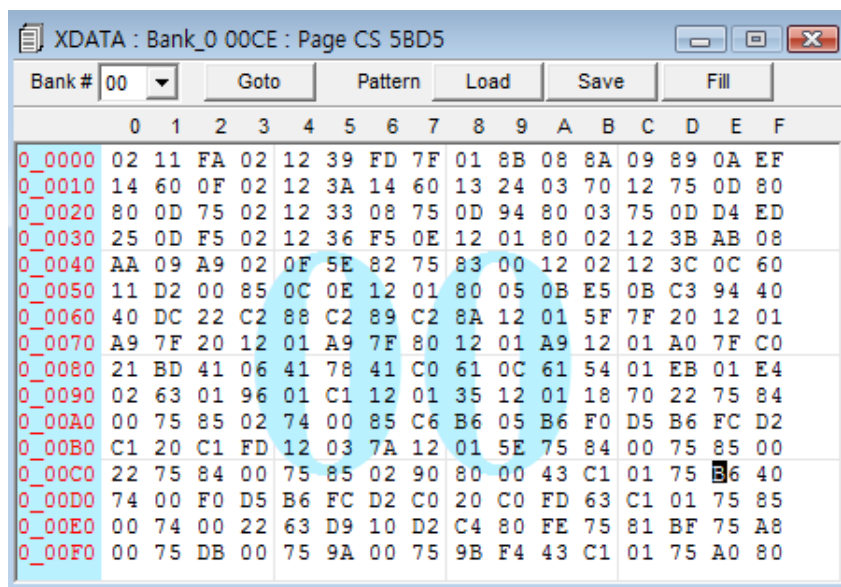


Figure 45. The Code Dump Function's Pattern Fill Dialog

## 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  $xx00\text{--}xxFFh$ , 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 46, for example, the page number is 00.

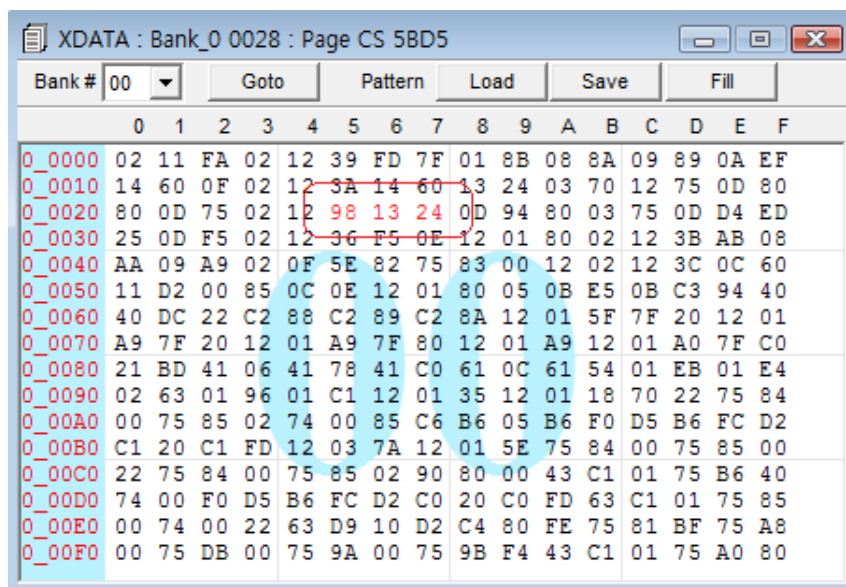


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0_0000	02	11	FA	02	12	39	FD	7F	01	8B	08	8A	09	89	0A	EF
0_0010	14	60	0F	02	12	3A	14	60	13	24	03	70	12	75	0D	80
0_0020	80	0D	75	02	12	33	08	75	0D	94	80	03	75	0D	D4	ED
0_0030	25	0D	F5	02	12	36	F5	0E	12	01	80	02	12	3B	AB	08
0_0040	AA	09	A9	02	0F	5E	82	75	83	00	12	02	12	3C	0C	60
0_0050	11	D2	00	85	0C	0E	12	01	80	05	0B	E5	0B	C3	94	40
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5F	7F	20	12	01
0_0070	A9	7F	20	12	01	A9	7F	80	12	01	A9	12	01	A0	7F	C0
0_0080	21	BD	41	06	41	78	41	C0	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	B6	05	B6	F0	D5	B6	FC	D2
0_00B0	C1	20	C1	FD	12	03	7A	12	01	5E	75	84	00	75	85	00
0_00C0	22	75	84	00	75	85	02	90	80	00	43	C1	01	75	86	40
0_00D0	74	00	F0	D5	B6	FC	D2	C0	20	C0	FD	63	C1	01	75	85
0_00E0	00	74	00	22	63	D9	10	D2	C4	80	FE	75	81	BF	75	A8
0_00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	A0	80

Figure 46. 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 47.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0_0000	02	11	FA	02	12	39	FD	7F	01	8B	08	8A	09	89	0A	EF
0_0010	14	60	0F	02	12	3A	14	60	13	24	03	70	12	75	0D	80
0_0020	80	0D	75	02	12	98	13	24	0D	94	80	03	75	0D	D4	ED
0_0030	25	0D	F5	02	12	36	F5	0E	12	01	80	02	12	3B	AB	08
0_0040	AA	09	A9	02	0F	5E	82	75	83	00	12	02	12	3C	0C	60
0_0050	11	D2	00	85	0C	0E	12	01	80	05	0B	E5	0B	C3	94	40
0_0060	40	DC	22	C2	88	C2	89	C2	8A	12	01	5F	7F	20	12	01
0_0070	A9	7F	20	12	01	A9	7F	80	12	01	A9	12	01	A0	7F	C0
0_0080	21	BD	41	06	41	78	41	C0	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	B6	05	B6	F0	D5	B6	FC	D2
0_00B0	C1	20	C1	FD	12	03	7A	12	01	5E	75	84	00	75	85	00
0_00C0	22	75	84	00	75	85	02	90	80	00	43	C1	01	75	B6	40
0_00D0	74	00	F0	D5	B6	FC	D2	C0	20	C0	FD	63	C1	01	75	85
0_00E0	00	74	00	22	63	D9	10	D2	C4	80	FE	75	81	BF	75	A8
0_00F0	00	75	DB	00	75	9A	00	75	9B	F4	43	C1	01	75	A0	80

Figure 47. 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 48.

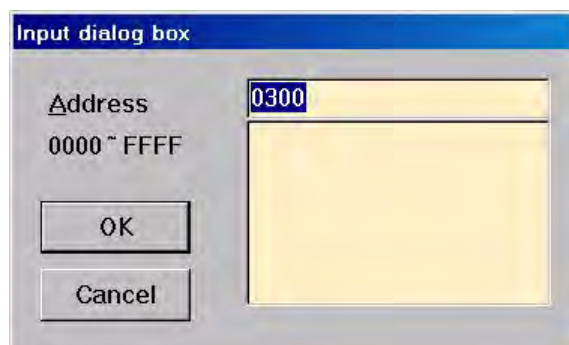


Figure 48. The XDATA Dump Function's Goto/Input Dialog

## 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 49.



Figure 49. 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 50.



Figure 50. 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 51.

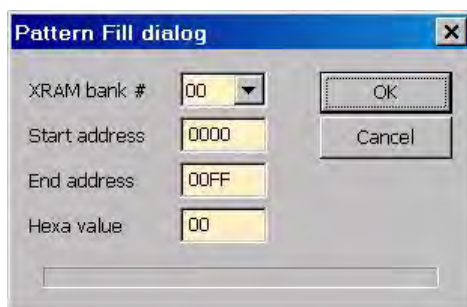


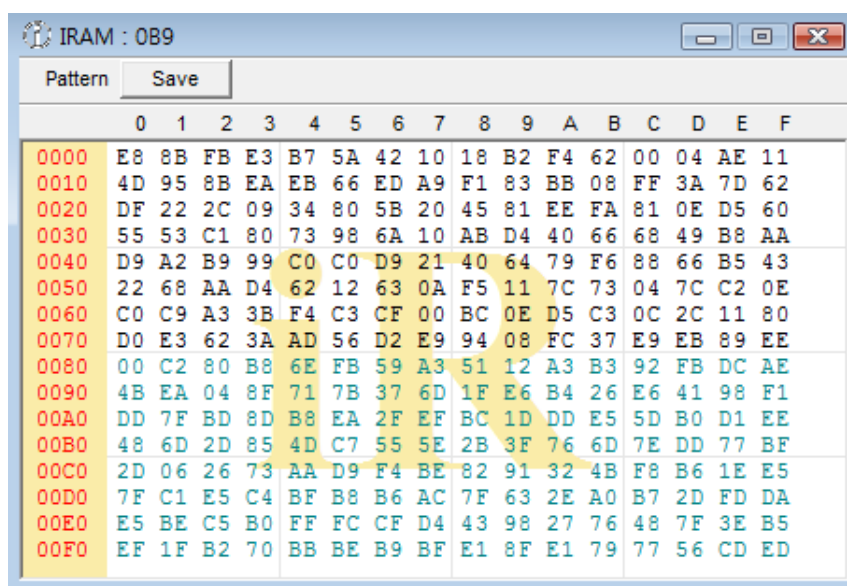
Figure 51. The XDATA Dump Function's Pattern Fill Dialog

## 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 52.

Figure 52 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.



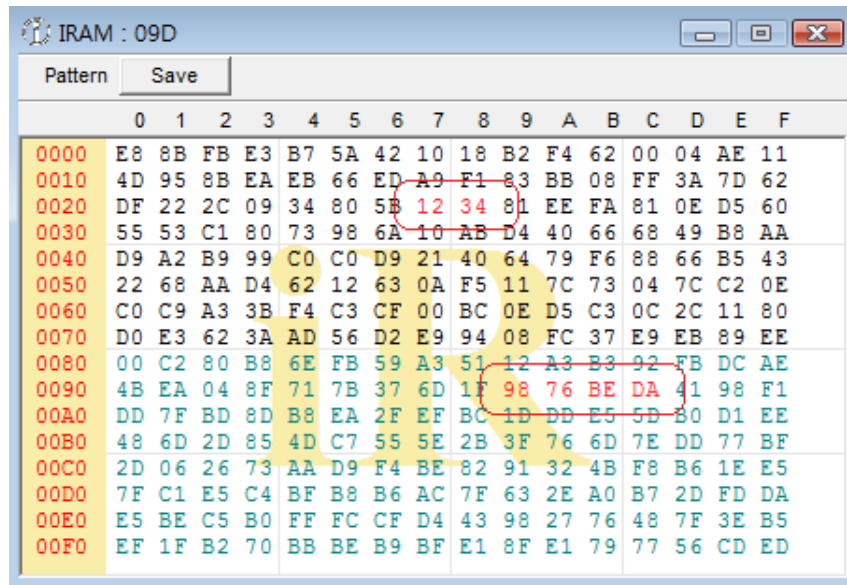
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	E8	8B	FB	E3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11
0010	4D	95	8B	EA	EB	66	ED	A9	F1	83	BB	08	FF	3A	7D	62
0020	DF	22	2C	09	34	80	5B	20	45	81	EE	FA	81	0E	D5	60
0030	55	53	C1	80	73	98	6A	10	AB	D4	40	66	68	49	B8	AA
0040	D9	A2	B9	99	C0	C0	D9	21	40	64	79	F6	88	66	B5	43
0050	22	68	AA	D4	62	12	63	0A	F5	11	7C	73	04	7C	C2	0E
0060	C0	C9	A3	3B	F4	C3	CF	00	BC	0E	D5	C3	0C	2C	11	80
0070	D0	E3	62	3A	AD	56	D2	E9	94	08	FC	37	E9	EB	89	EE
0080	00	C2	80	B8	6E	FB	59	A3	51	12	A3	B3	92	FB	DC	AE
0090	4B	EA	04	8F	71	7B	37	6D	1F	E6	B4	26	E6	41	98	F1
00A0	DD	7F	BD	8D	B8	EA	2F	EF	BC	1D	DD	E5	5D	B0	D1	EE
00B0	48	6D	2D	85	4D	C7	55	5E	2B	3F	76	6D	7E	DD	77	BF
00C0	2D	06	26	73	AA	D9	F4	BE	82	91	32	4B	F8	B6	1E	E5
00D0	7F	C1	E5	C4	BF	B8	B6	AC	7F	63	2E	A0	B7	2D	FD	DA
00E0	E5	BE	C5	B0	FF	FC	CF	D4	43	98	27	76	48	7F	3E	B5
00F0	EF	1F	B2	70	BB	BE	B9	BF	E1	8F	E1	79	77	56	CD	ED

Figure 52. 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 53.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	E8	8B	FB	E3	B7	5A	42	10	18	B2	F4	62	00	04	AE	11
0010	4D	95	8B	EA	EB	66	ED	A9	F1	83	BB	08	FF	3A	7D	62
0020	DF	22	2C	09	34	80	5B	12	34	81	EE	FA	81	0E	D5	60
0030	55	53	C1	80	73	98	6A	10	AB	D4	40	66	68	49	B8	AA
0040	D9	A2	B9	99	C0	C0	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	C9	A3	3B	F4	C3	CF	00	BC	0E	D5	C3	0C	2C	11	80
0070	D0	E3	62	3A	AD	56	D2	E9	94	08	FC	37	E9	EB	89	EE
0080	00	C2	80	B8	6E	FB	59	A3	51	12	A3	B3	92	FB	DC	AE
0090	4B	EA	04	8F	71	7B	37	6D	1F	98	76	BE	DA	41	98	F1
00A0	DD	7F	BD	8D	B8	EA	2F	EF	BC	1D	DD	E5	5D	80	D1	EE
00B0	48	6D	2D	85	4D	C7	55	5E	2B	3F	76	6D	7E	DD	77	BF
00C0	2D	06	26	73	AA	D9	F4	BE	82	91	32	4B	F8	B6	1E	E5
00D0	7F	C1	E5	C4	BF	B8	B6	AC	7F	63	2E	A0	B7	2D	FD	DA
00E0	E5	BE	C5	B0	FF	FC	CF	D4	43	98	27	76	48	7F	3E	B5
00F0	EF	1F	B2	70	BB	BE	B9	BF	E1	8F	E1	79	77	56	CD	ED

Figure 53. 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.

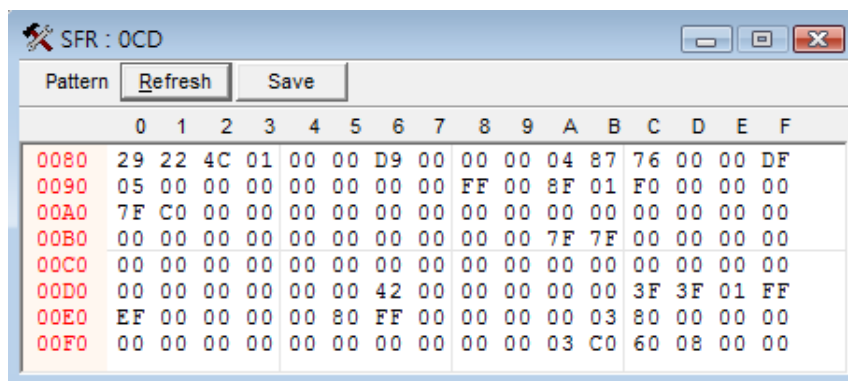
### Save

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 54 shows SFR addresses in the range 80h–FFh, which represent the direct area of IRAM.



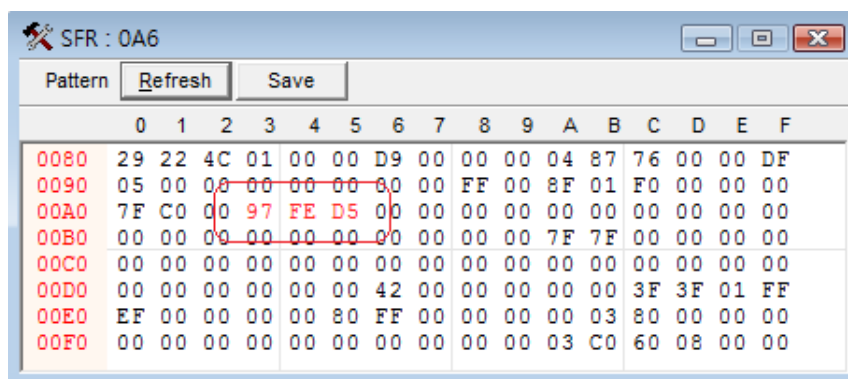
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0080	29	22	4C	01	00	00	D9	00	00	00	04	87	76	00	00	DF
0090	05	00	00	00	00	00	00	00	FF	00	8F	01	F0	00	00	00
00A0	7F	C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	7F	7F	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D0	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 54. 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 55.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0080	29	22	4C	01	00	00	D9	00	00	00	04	87	76	00	00	DF
0090	05	00	00	00	00	00	00	00	FF	00	8F	01	F0	00	00	00
00A0	7F	C0	00	97	FE	D5	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	7F	7F	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 55. 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 56) redisplay all current data.

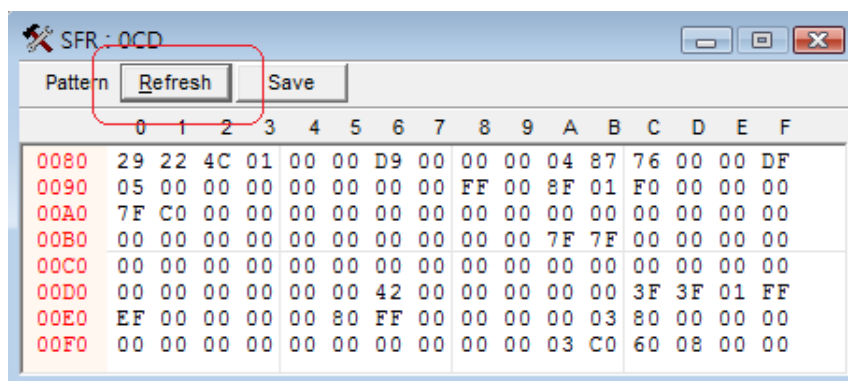


Figure 56. Using the SFRDump Function, #3 of 3

## 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 57.

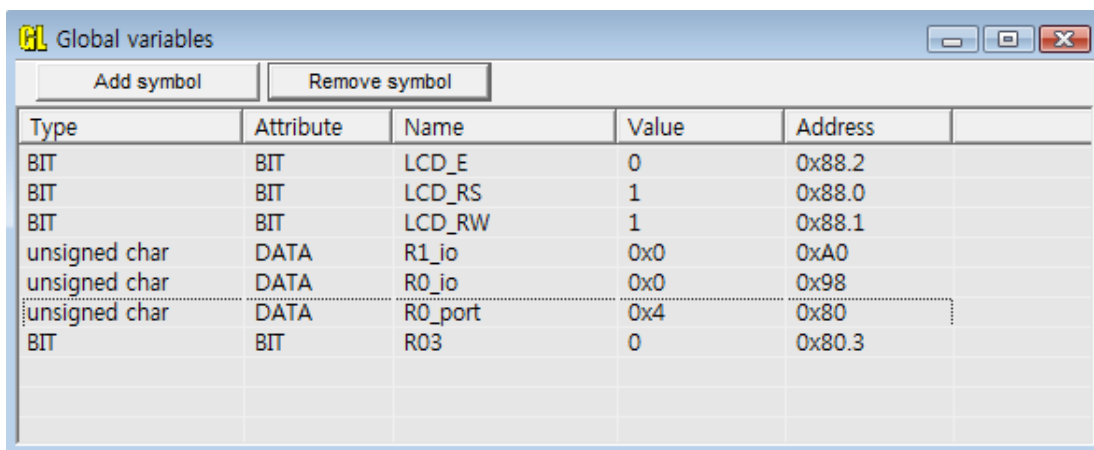


Figure 57. The Watch Global Function's Global Variables Dialog

### 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 58.

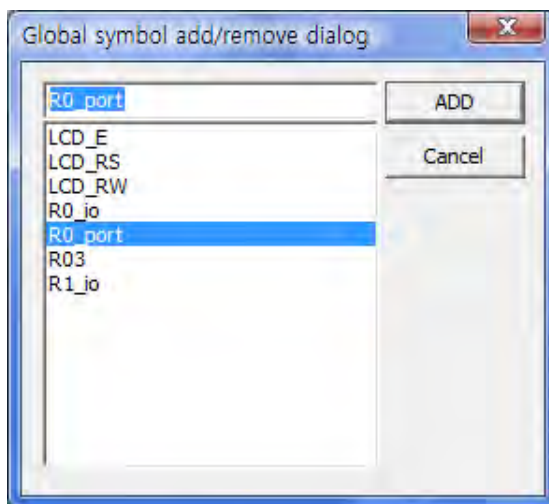


Figure 58. Adding A Global Symbol

### 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 59.

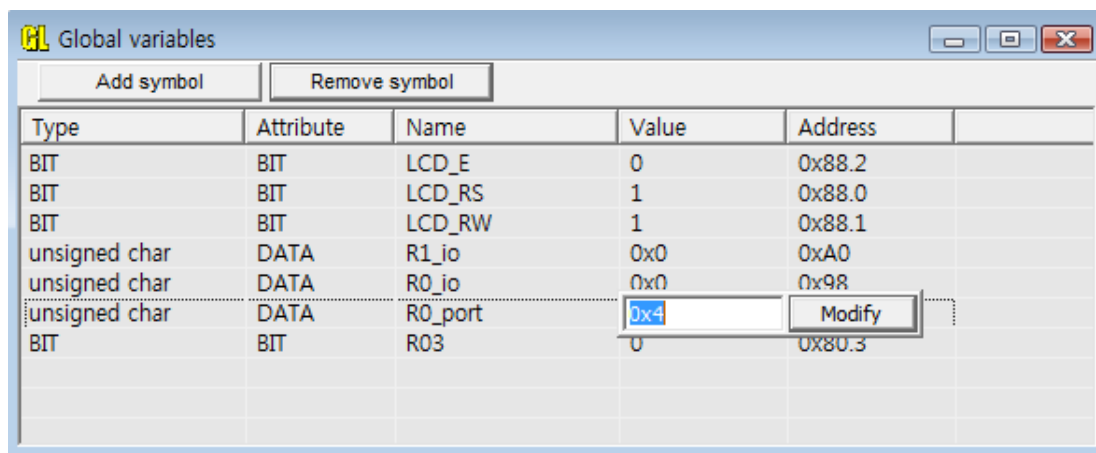


Figure 59. Editing A Global Symbol

## 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 60, alleviates this problem by employing a map/symbol file, as described in the previous section.

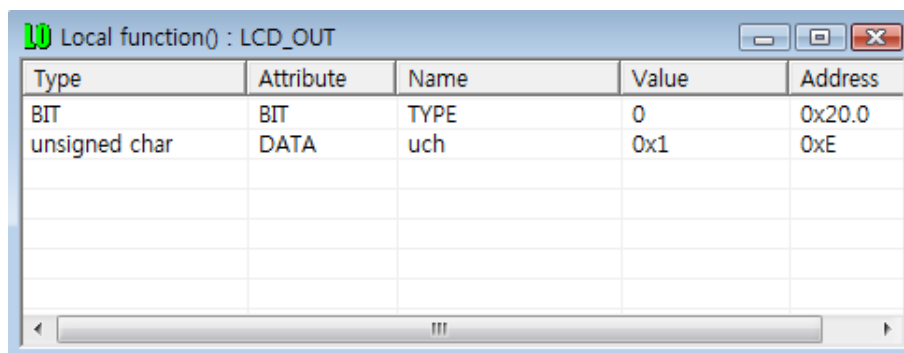


Figure 60. The Watch Local Function Dialog

## 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 61.

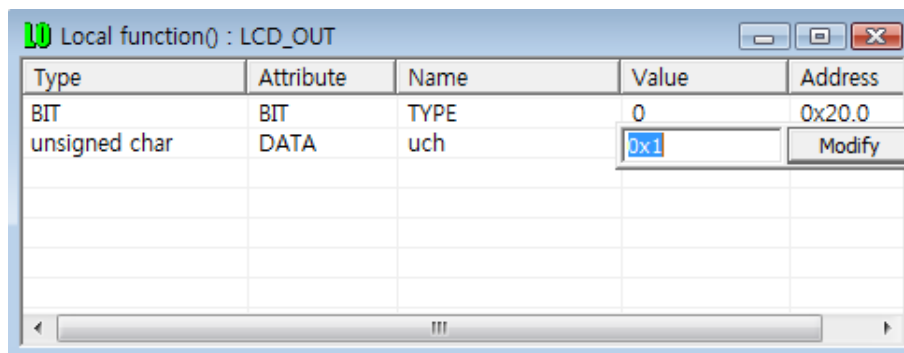


Figure 61. Editing A Local Symbol

## Add or Remove Symbol

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

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

Figure 62 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).

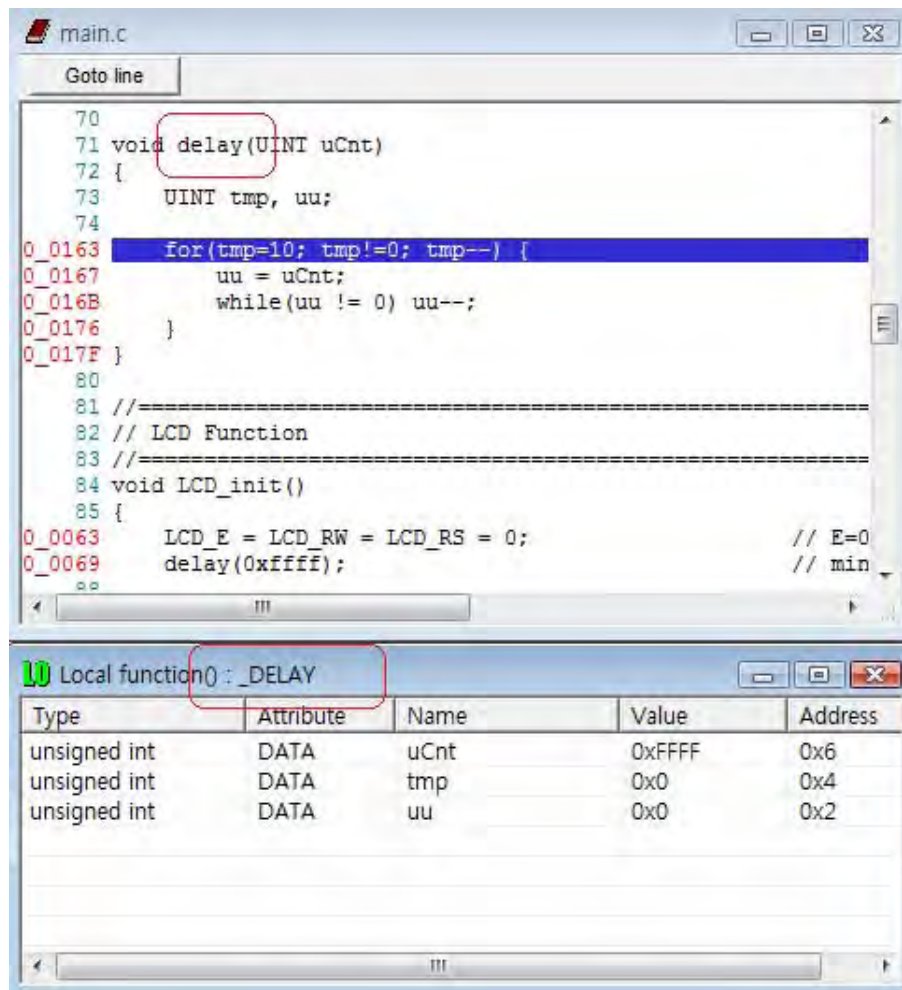


Figure 62. Example Watch Local Function, #1 of 2

If program flow is changed to another module, then the Local Variable list will be changed, as shown in Figure 63.

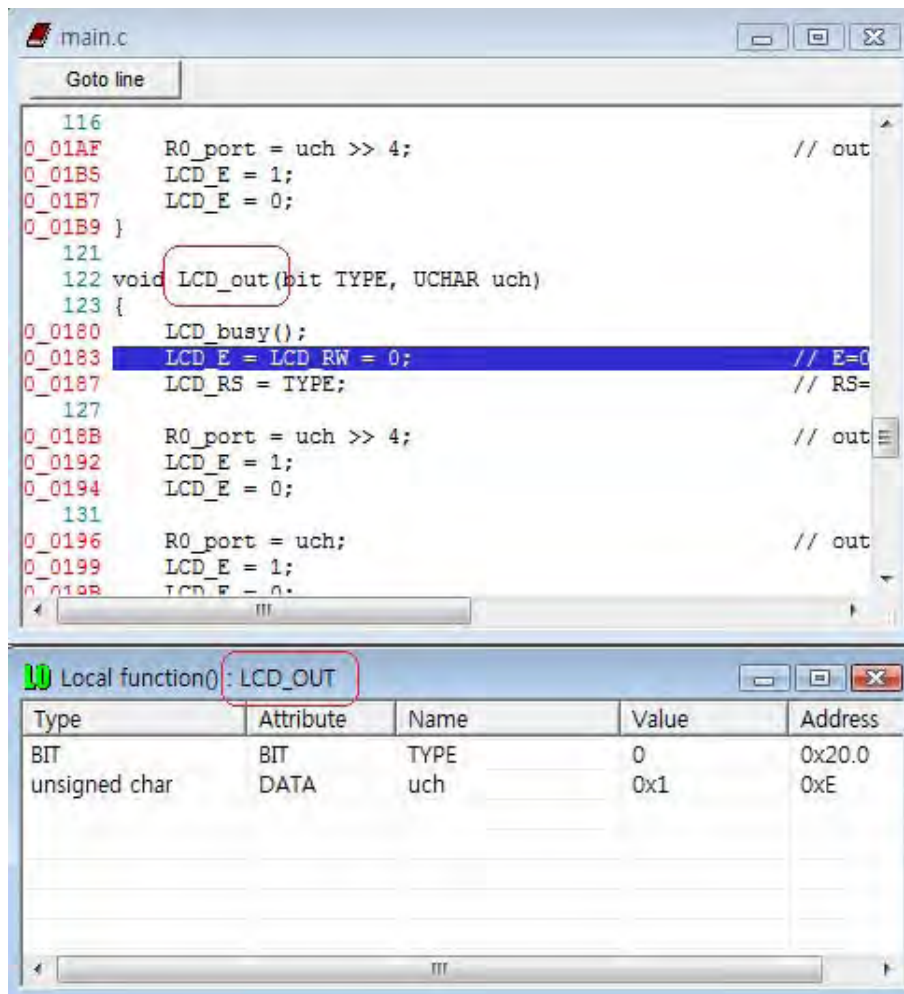
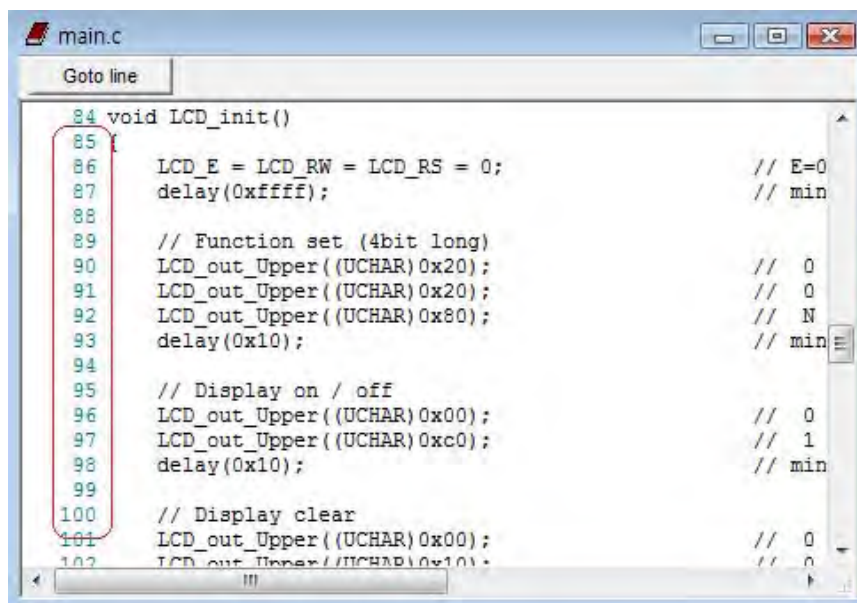


Figure 63. Example Watch Local Function, #2 of 2

## 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.

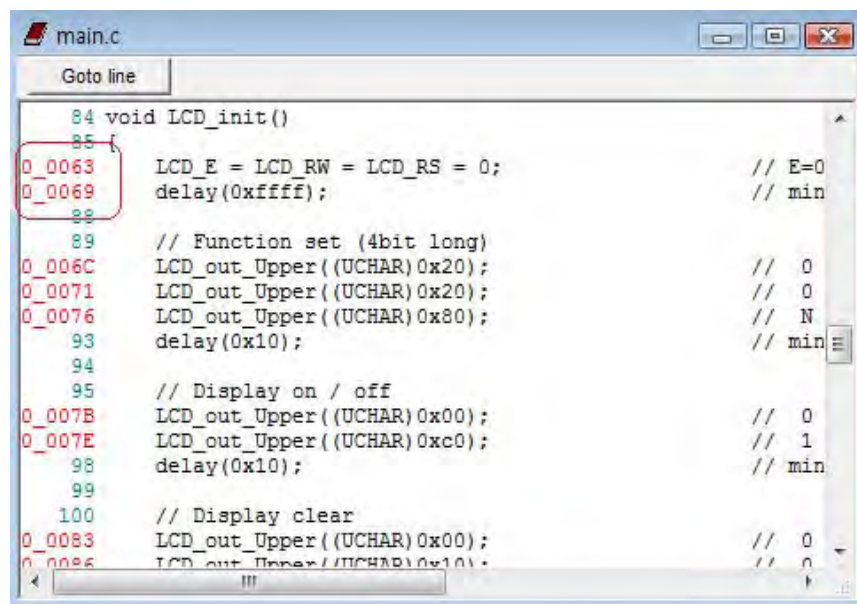
**Example 1.** If a map/symbol file has not been loaded, or if the file does not include symbol information, only the line number is displayed, as highlighted in Figure 64.



```
84 void LCD_init()
85 {
86     LCD_E = LCD_RW = LCD_RS = 0;           // E=0
87     delay(0xffff);                         // min
88
89     // Function set (4bit long)
90     LCD_out_Upper((UCHAR)0x20);             // 0
91     LCD_out_Upper((UCHAR)0x20);             // 0
92     LCD_out_Upper((UCHAR)0x80);             // N
93     delay(0x10);                           // min
94
95     // Display on / off
96     LCD_out_Upper((UCHAR)0x00);             // 0
97     LCD_out_Upper((UCHAR)0xc0);             // 1
98     delay(0x10);                           // min
99
100    // Display clear
101    LCD_out_Upper((UCHAR)0x00);             // 0
102    LCD_out_Upper((UCHAR)0x10);             // 0
```

Figure 64. 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 65.



```
84 void LCD_init()
85 {
0_0063 LCD_E = LCD_RW = LCD_RS = 0;           // E=0
0_0069 delay(0xffff);                         // min
88
89     // Function set (4bit long)
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 LCD_out_Upper((UCHAR)0x10);             // 0
```

Figure 65. 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 66. Map/symbol information is not required.

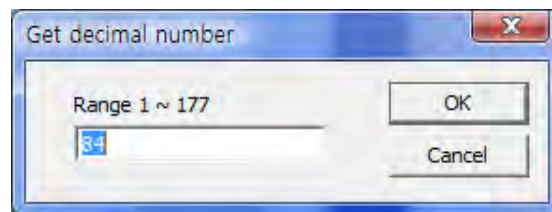
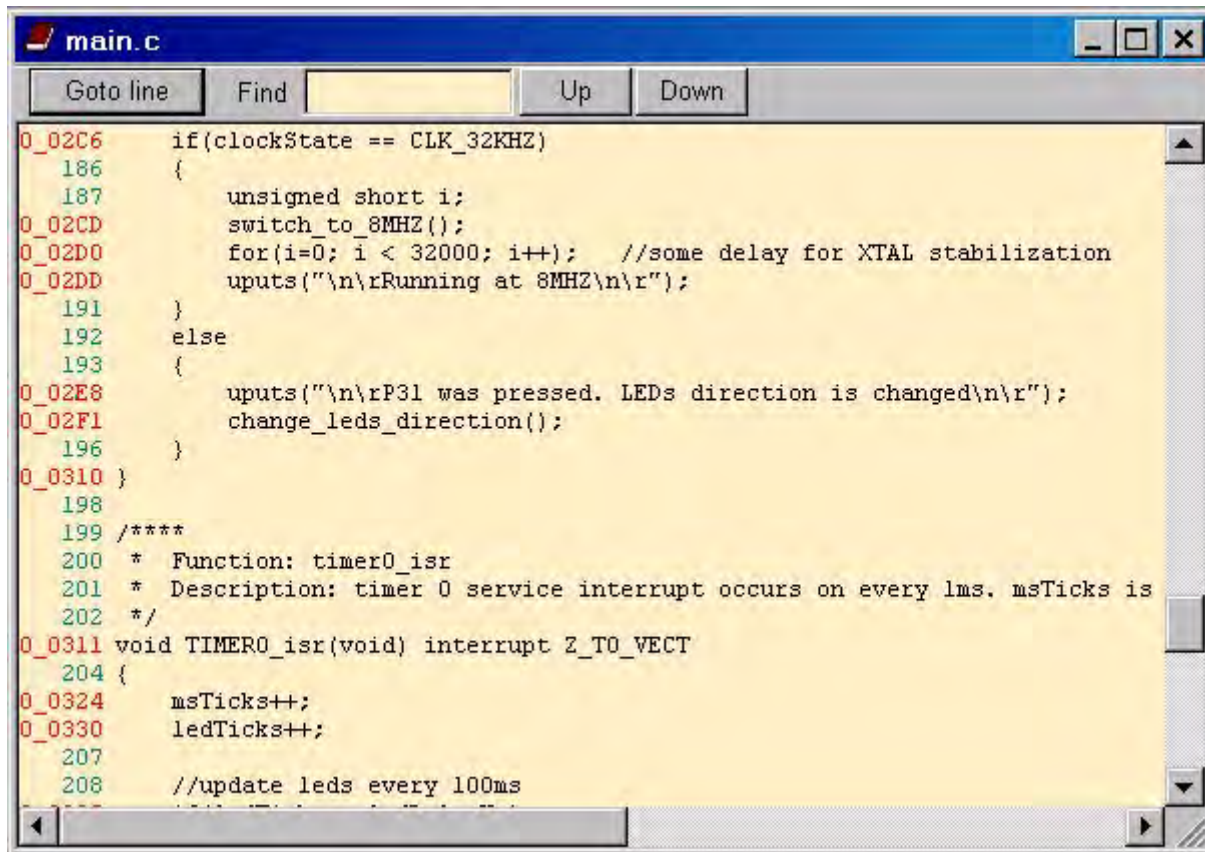


Figure 66. Using the Text File Function, #3 of 5

## Disassemble Window Linkage

If a map/symbol file has been loaded and a text file is displayed (see the example in Figure 67), the text file will show addresses instead of line numbers. In this Text File Child Window, and with your mouse, set your cursor in an address area (the left-most column) and double-click the left button to launch the Code Disassemble dialog, which will highlight the address.





```
0_02C6  if(clockState == CLK_32KHZ)
186    {
187        unsigned short i;
0_02CD    switch_to_8MHZ();
0_02D0    for(i=0; i < 32000; i++); //some delay for XTAL stabilization
0_02DD    uputs("\n\rRunning at 8MHZ\n\r");
191    }
192    else
193    {
0_02E8    uputs("\n\rP31 was pressed. LEDs direction is changed\n\r");
0_02F1    change_leds_direction();
196    }
0_0310 }
198
199 /****
200  * Function: timer0_isr
201  * Description: timer 0 service interrupt occurs on every 1ms. msTicks is
202  */
0_0311 void TIMER0_isr(void) interrupt Z_TO_VECT
204 {
0_0324    msTicks++;
0_0330    ledTicks++;
207
208    //update leds every 100ms
```

Figure 67. Text File Child Window

**Example.** Double-click the left button on your mouse at address 0\_02C6. The Code Disassembler child window appears, showing the 0\_02C6 location at the top of the dialog; see Figure 68.

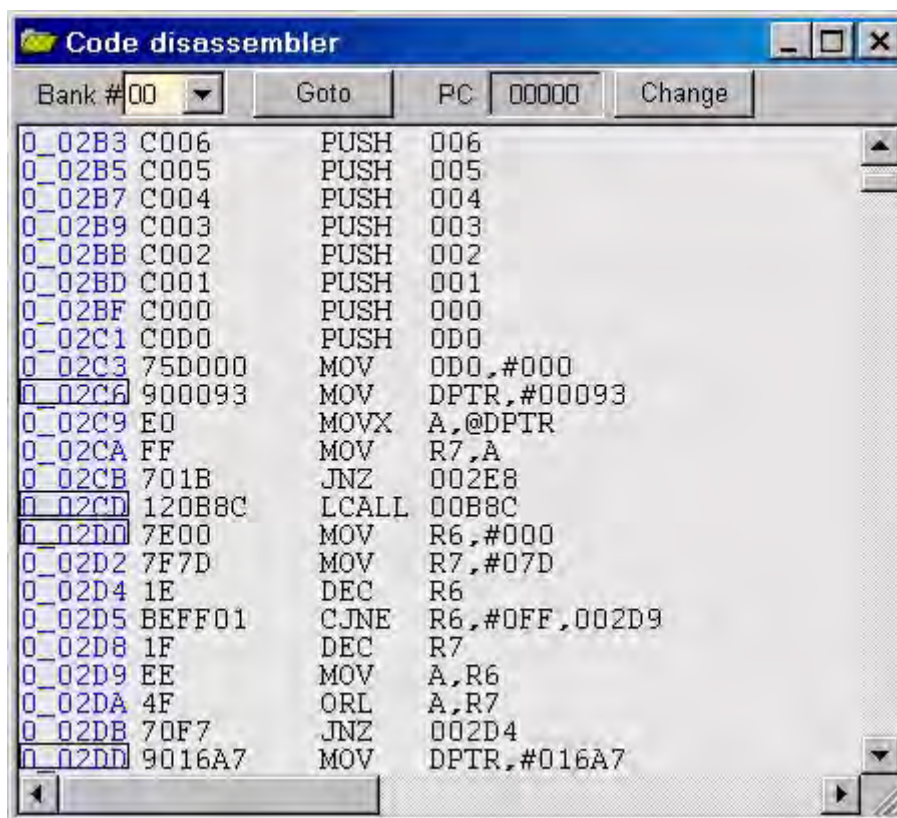
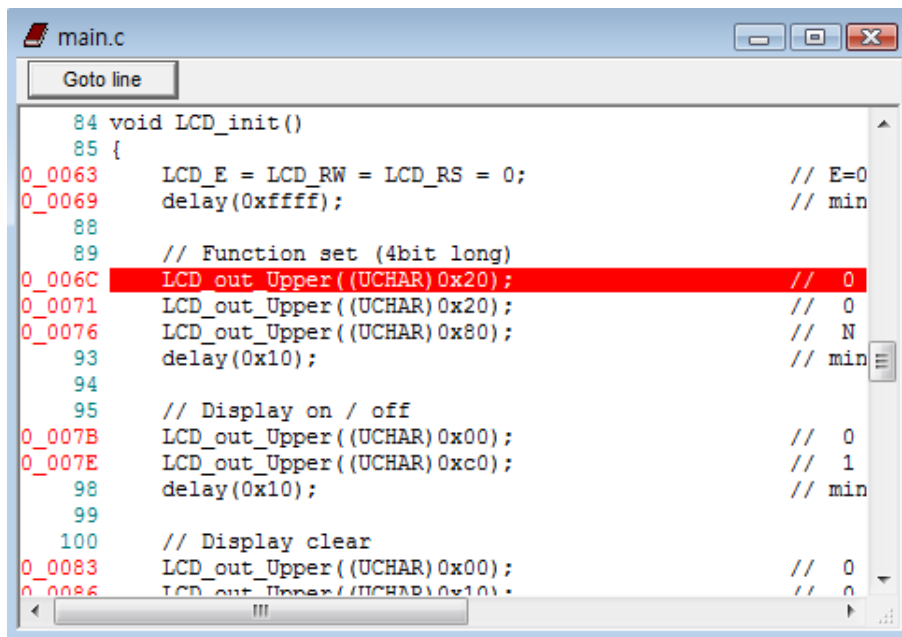


Figure 68. Code Disassembler Child Window

### Break Toggle

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

Figure 69 shows an example of a segment of source code in which the color of the PC breakpoint line is red.

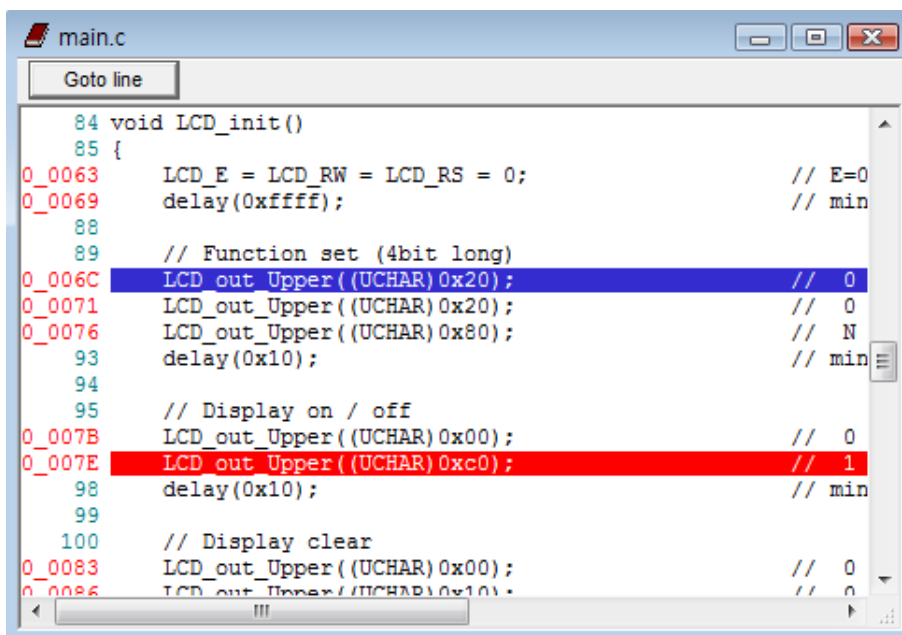


```
main.c
Goto line

84 void LCD_init()
85 {
0_0063     LCD_E = LCD_RW = LCD_RS = 0;           // E=0
0_0069     delay(0xffff);                       // min
88
89     // Function set (4bit long)
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     LCD_out_Upper((UCHAR)0x10);           // 0
```

Figure 69. Using the Text File Function, #4 of 5

In Figure 70, the color of the current program counter address line is blue.



```
main.c
Goto line

84 void LCD_init()
85 {
0_0063     LCD_E = LCD_RW = LCD_RS = 0;           // E=0
0_0069     delay(0xffff);                       // min
88
89     // Function set (4bit long)
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     LCD_out_Upper((UCHAR)0x10);           // 0
```

Figure 70. 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 lines<sup>1</sup> 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.

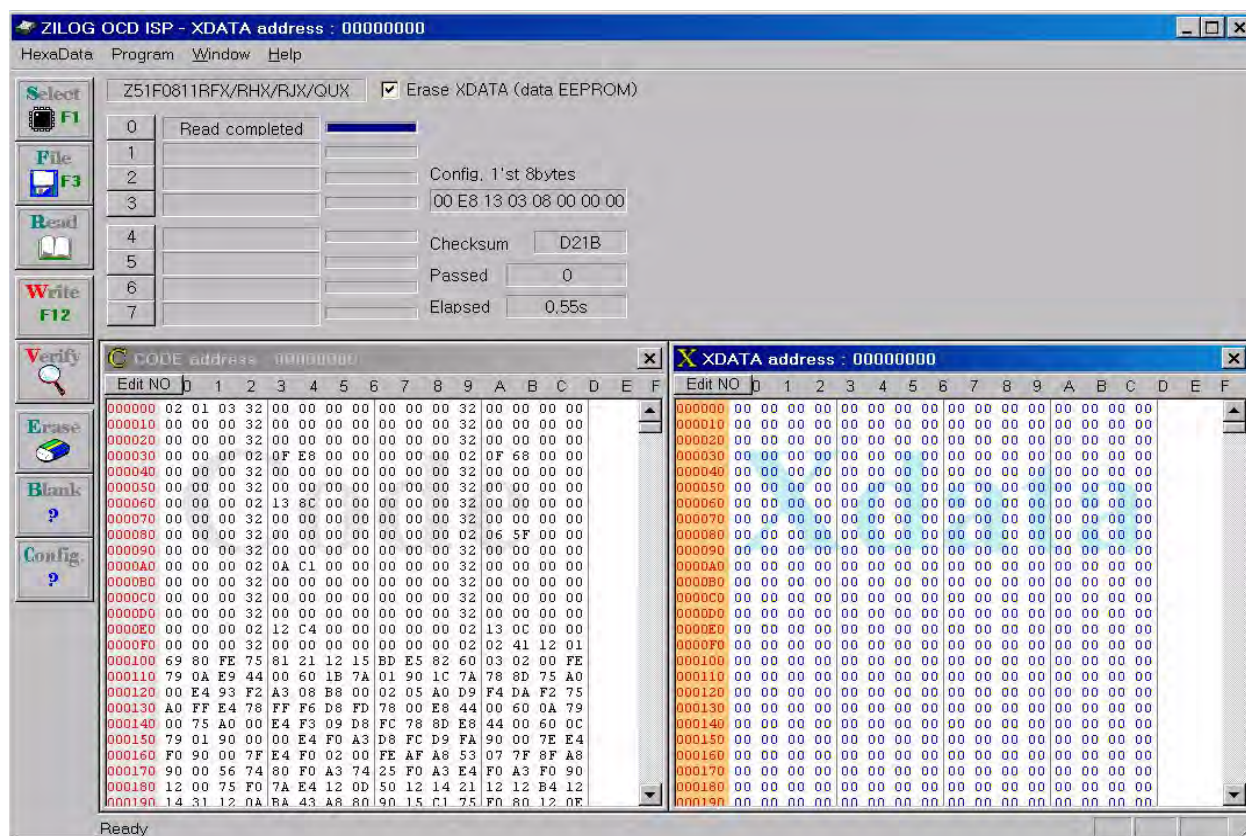


Figure 71. Example On-Chip Debugger ISP Screen

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



## Features

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

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

## Connect the Hardware

After installing the OCD software and drivers, hardware connections can be established. The pin positions of the Z8051 USB OCD interface are shown in Figure 72. Confirm the target device's pin positioning, and connect this interface to the USB port of your PC.

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

Figure 72. 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 73, 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.

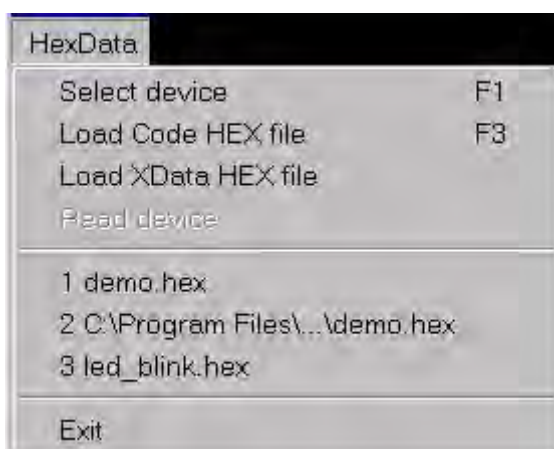


Figure 73. The OCD ISP's File Menu

## Select Device

Observe the following procedure to select a target device.

1. 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 74.





Figure 74. 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 75.

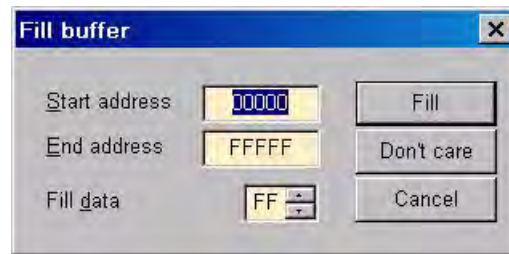


Figure 75. Fill Buffer Dialog

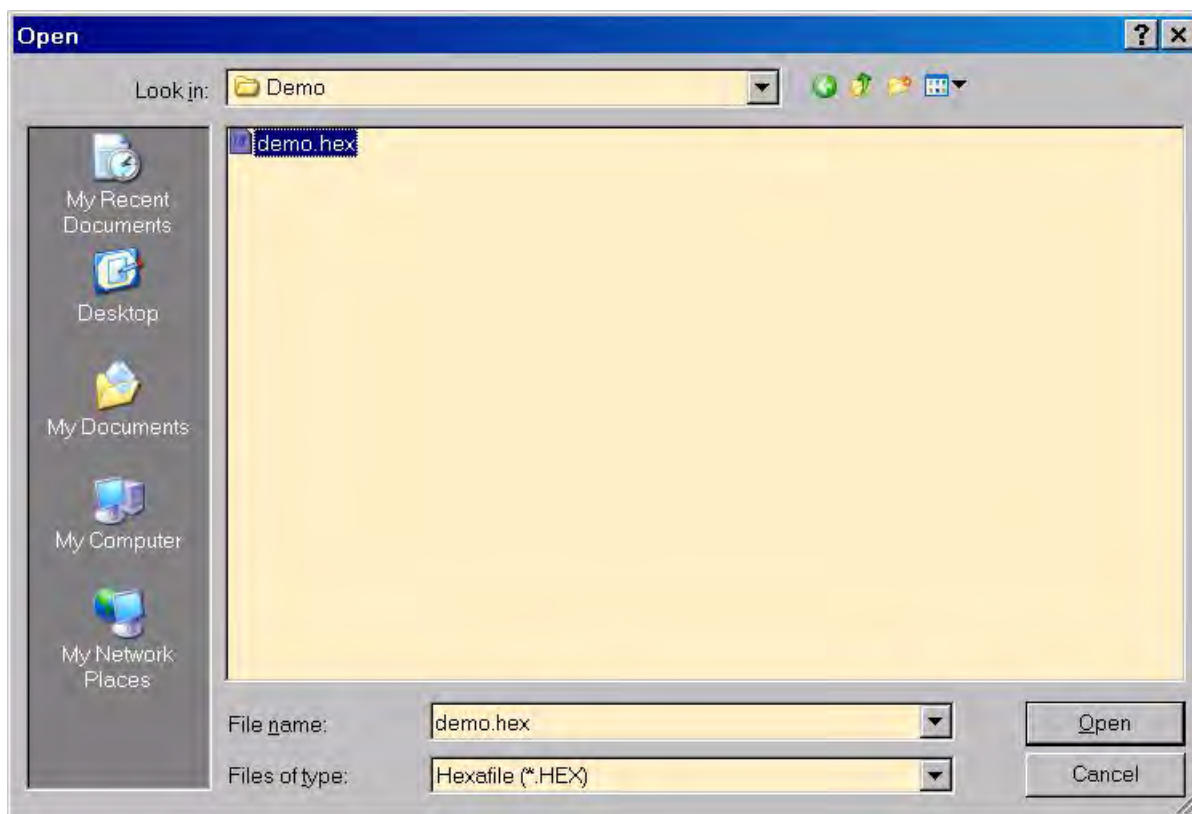
---

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



**Figure 76. File Open Dialog**

3. The OCD\_ISP dialog box appears, as shown in Figure 77. After a hexadecimal file has been loaded, this dialog displays the name of the target device and a data checksum value, as highlighted in the figure.

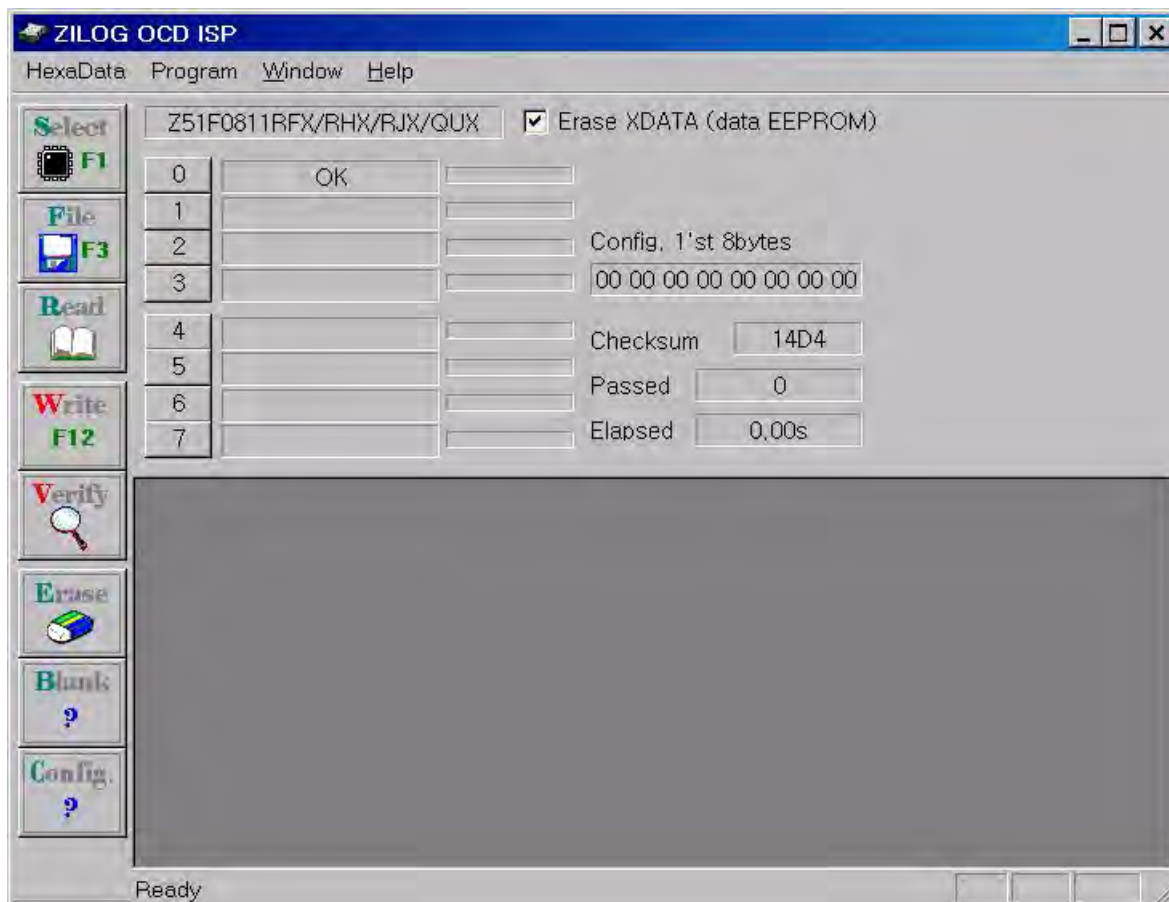


Figure 77. OCD ISP Dialog

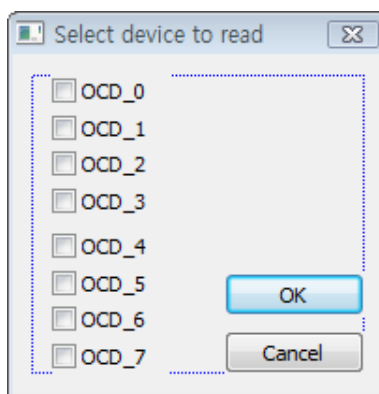
- 
- **Note:** The ISP cannot calculate the checksum without a defined code buffer range (see [Step 1](#)). Therefore, if you have not yet selected a target device yet, the Checksum field will display ????.
- 

### 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 78.



**Figure 78. 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 recently-used files that will appear in the Hex Data menu is eight. See the example in Figure 79, which shows that the user has recently opened only three files; the third file (selected in the figure) is the demo.hex file.

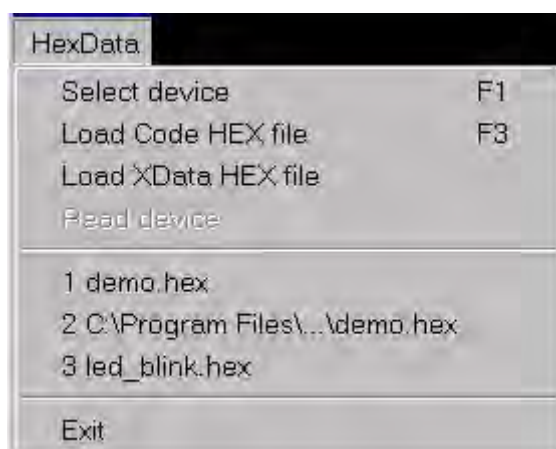


Figure 79. Most Recently Used Files

## Exit

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

## Program Menu

The Program menu, shown in Figure 80, 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.

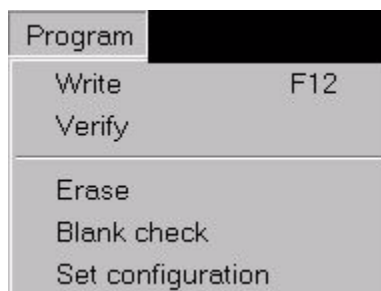


Figure 80. The OCD ISP's Program Menu

## 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 81 for an example.

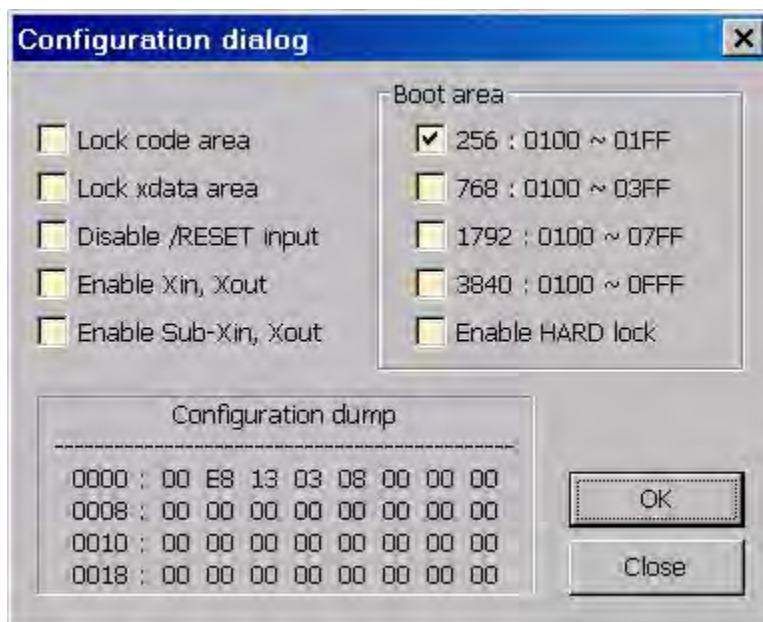


Figure 81. Configuration Dialog

## Window Menu

The Window menu, shown in Figure 82, can be used to modify the arrangement of child windows or to directly select a child window.



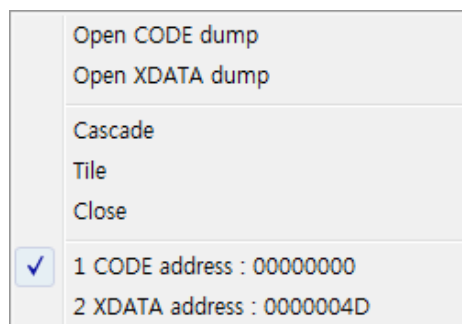


Figure 82. The OCD ISP's Window Menu

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

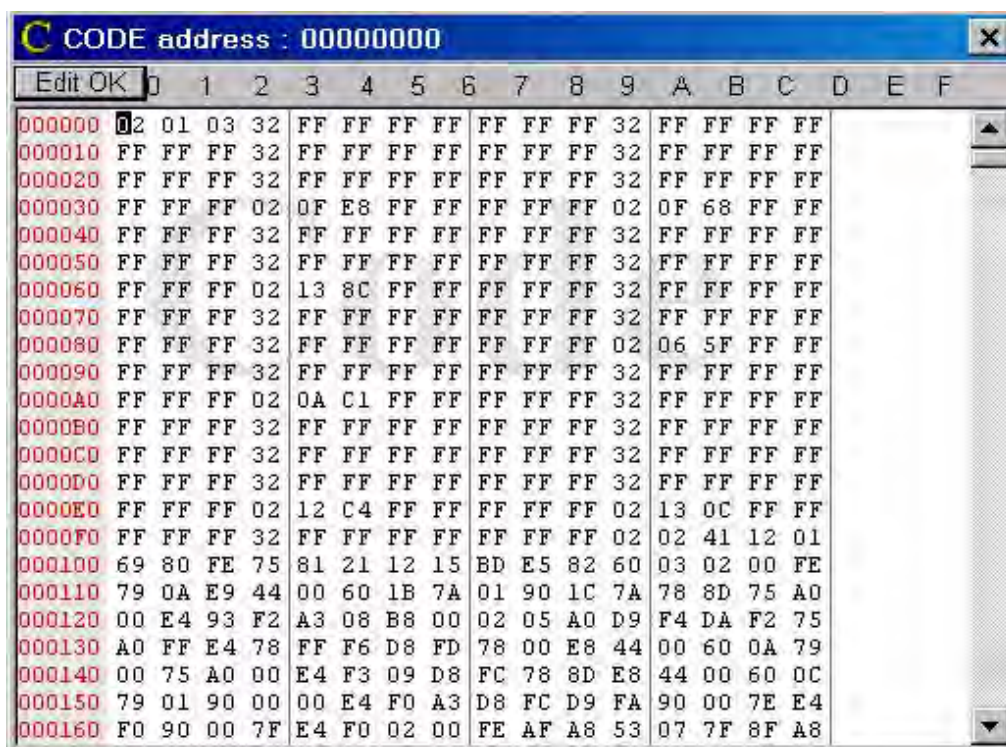


Figure 83. Open CODE Dump Child Window

## Open XData Dump

Selecting **Open XData Dump** from the Window menu opens a window which displays all external data (XData) memory in a dump format. An example is shown in Figure 84. If this window is already open, the window will move to the top-most level of the debugger frame.

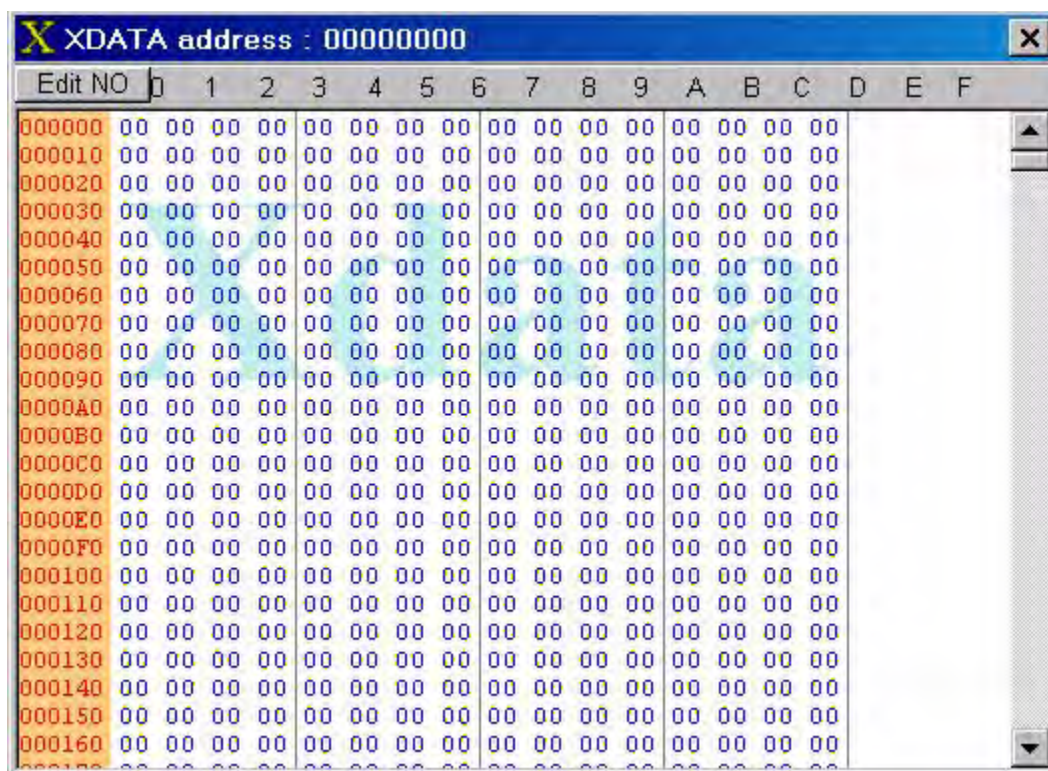


Figure 84. Open XData Dump Child Window

## Cascade

Selecting **Cascade** from the **Window** menu arranges opened child windows in a stepped (cascading) visual display, as shown in Figure 85.

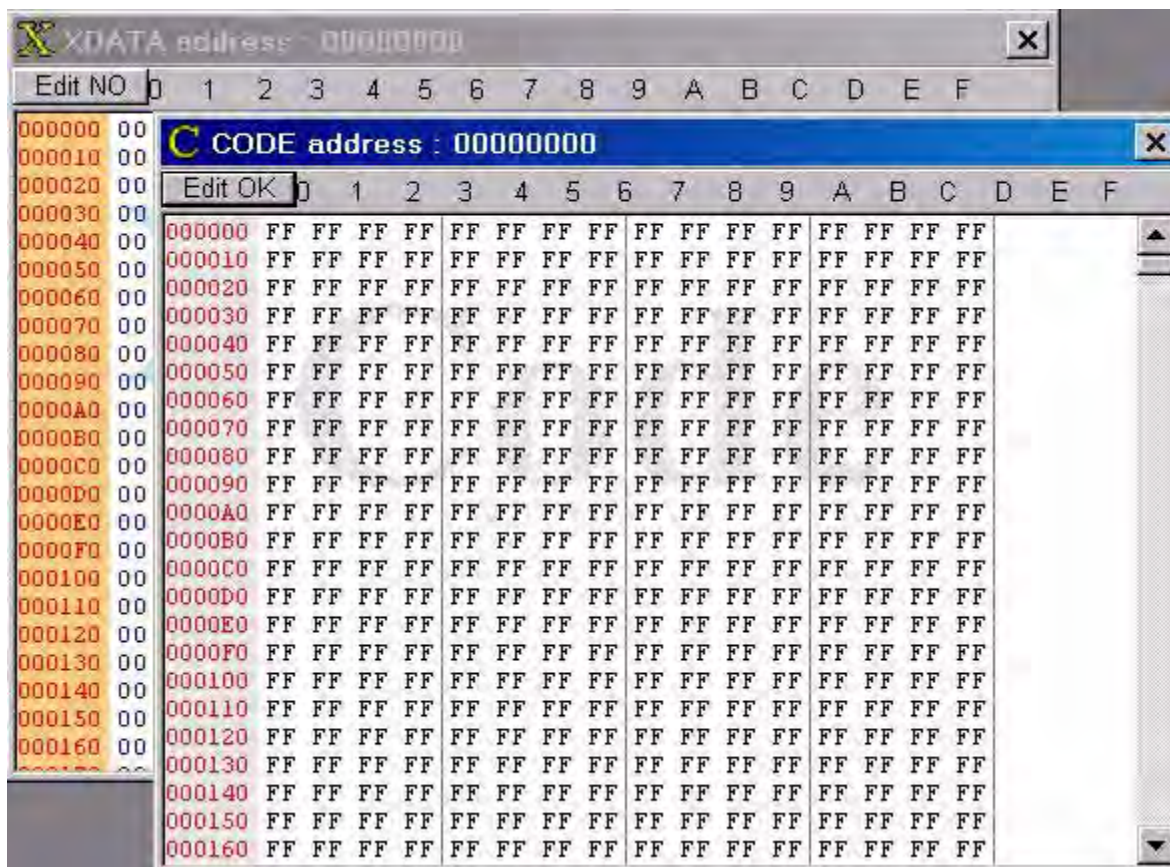


Figure 85. Cascading Child Windows



## Tile

Selecting **Tile** from the **Window** menu arranges opened child windows in a partitioned (tiled) display, as shown in Figure 86.

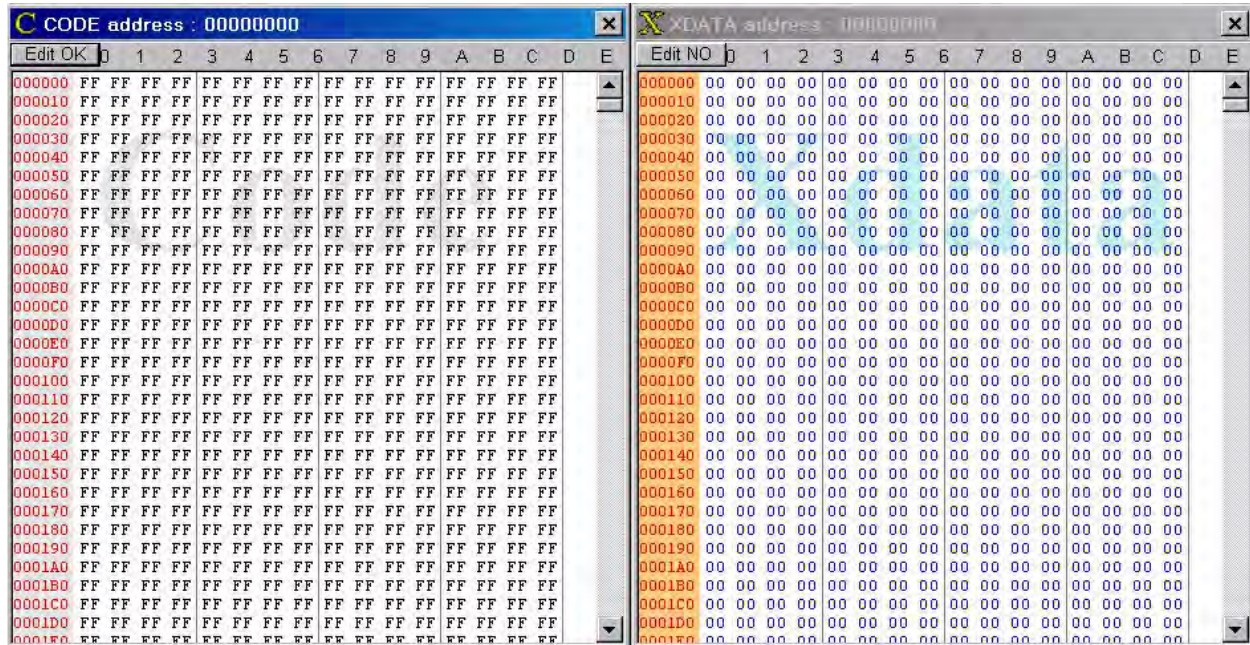


Figure 86. Tiled Child Windows

## 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 82](#) on page 73, 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 87.

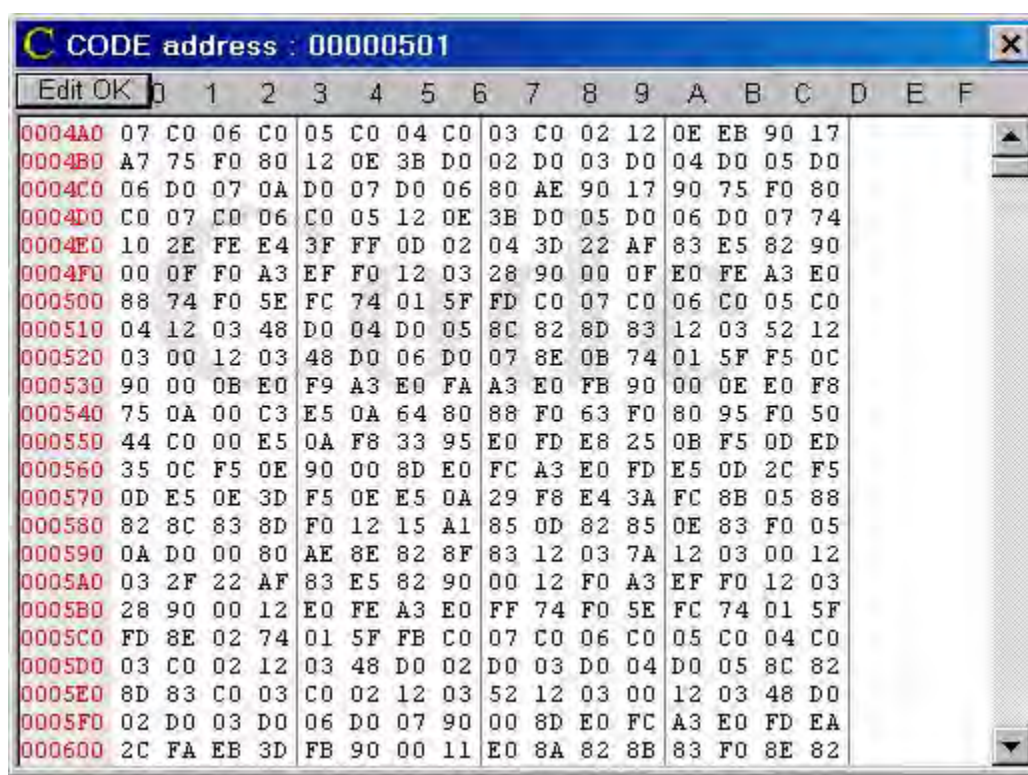
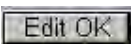


Figure 87. CODE Dump Child Window

### 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 (  ) 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 (  ) 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 88. Editing and cursor functions are the same for the XDATA Dump window as they are for the Code Dump window.

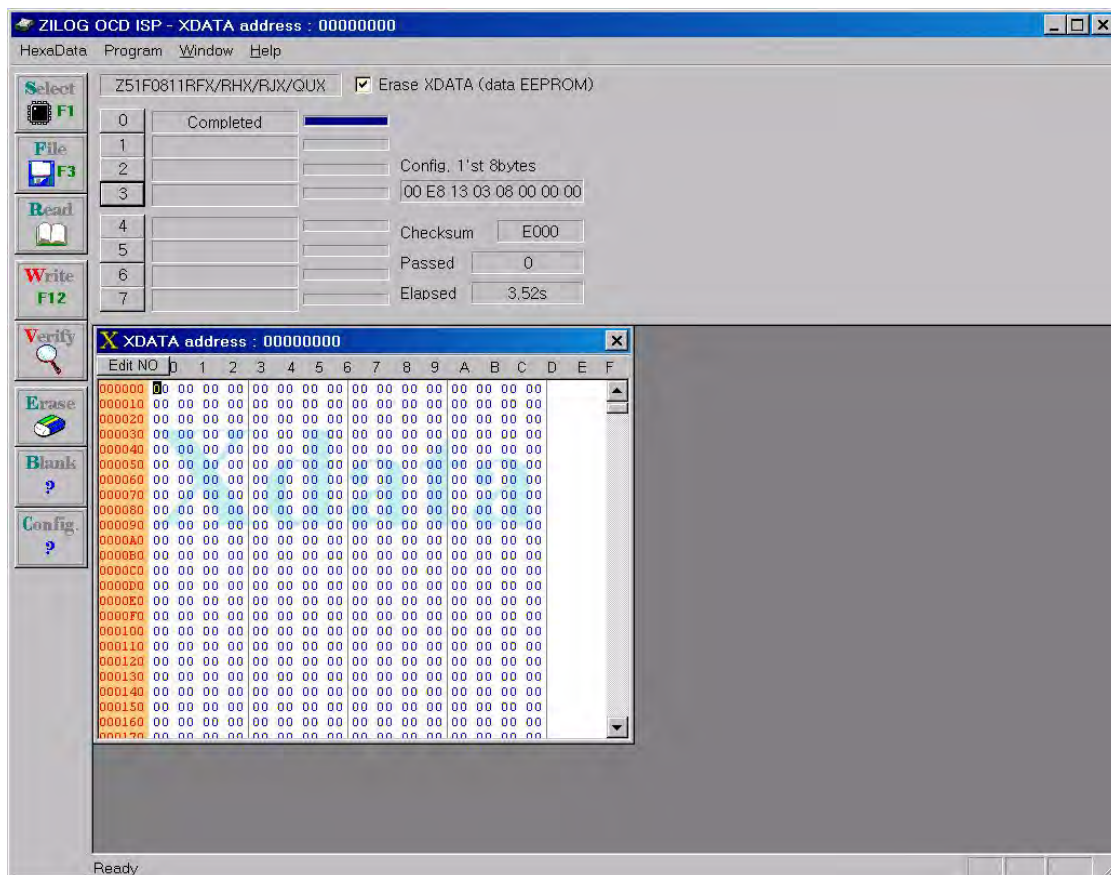


Figure 88. XData Dump Child Window



## 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 [Figure 2](#) on page 5 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 89. Click **Locate and install driver software (recommended)**.

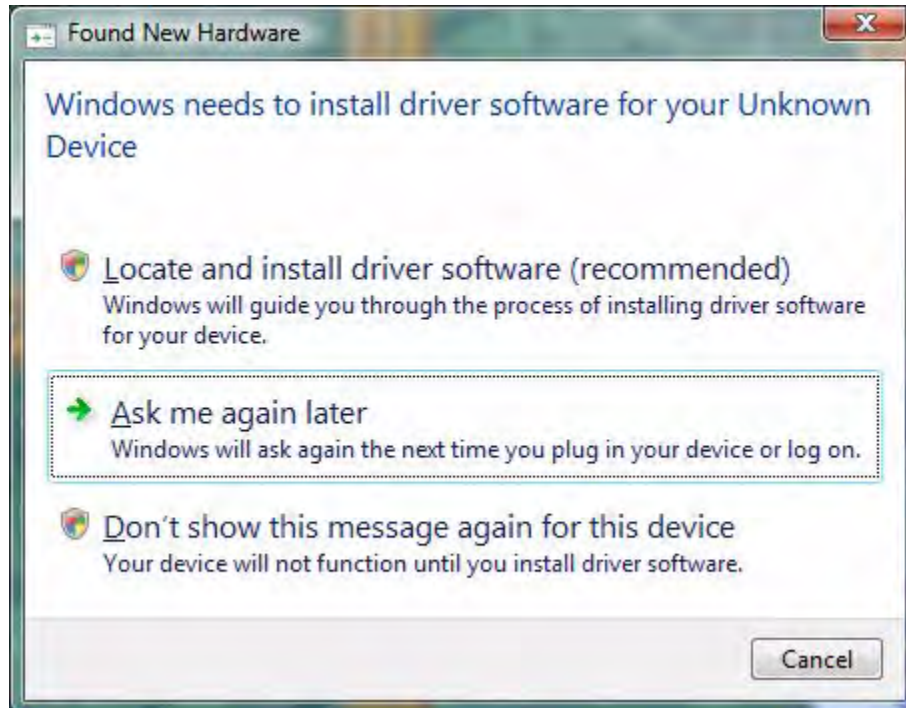


Figure 89. Found New Hardware Dialog, Windows Vista

3. Depending on your development PC's User Account Control settings, Windows may ask for permission to continue the installation. Click **Continue**.
4. When the **Insert the Disc** dialog appears, as shown in Figure 90, select **I don't have the disc. Show me other options**. Click the **Next** button to display the dialog that follows, which is shown in Figure 91.



Figure 90. Install Device Driver Dialog, Windows Vista



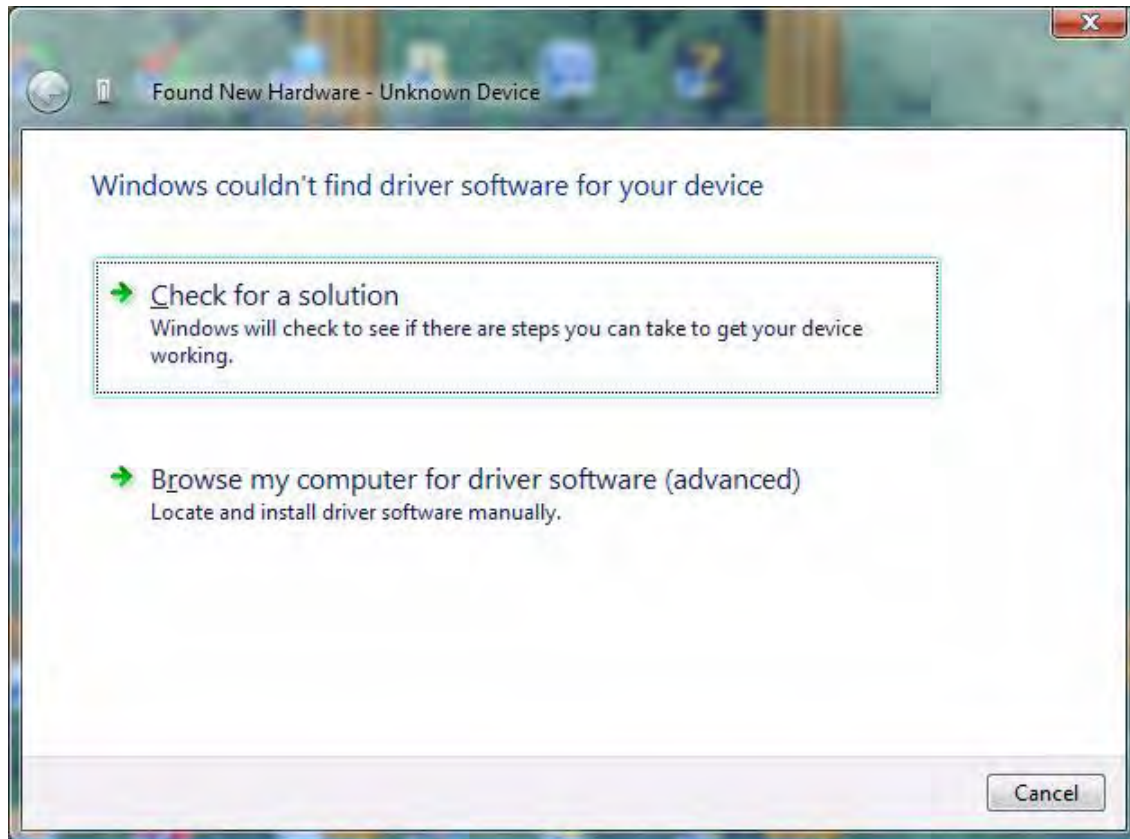


Figure 91. Couldn't Find Driver Dialog, Windows Vista

5. Select **Browse my computer for driver software (advanced)** to display the dialog shown in Figure 92, 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

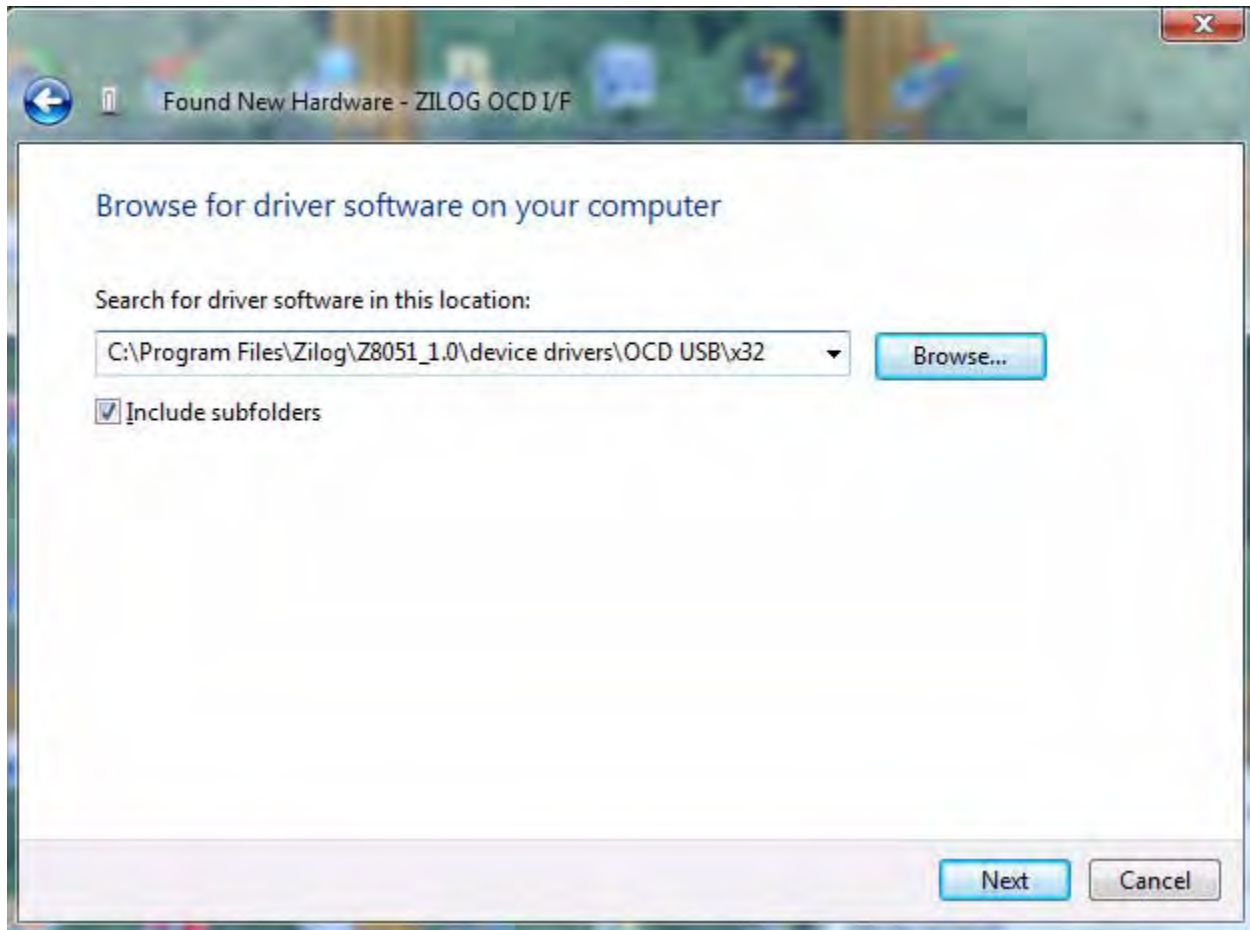


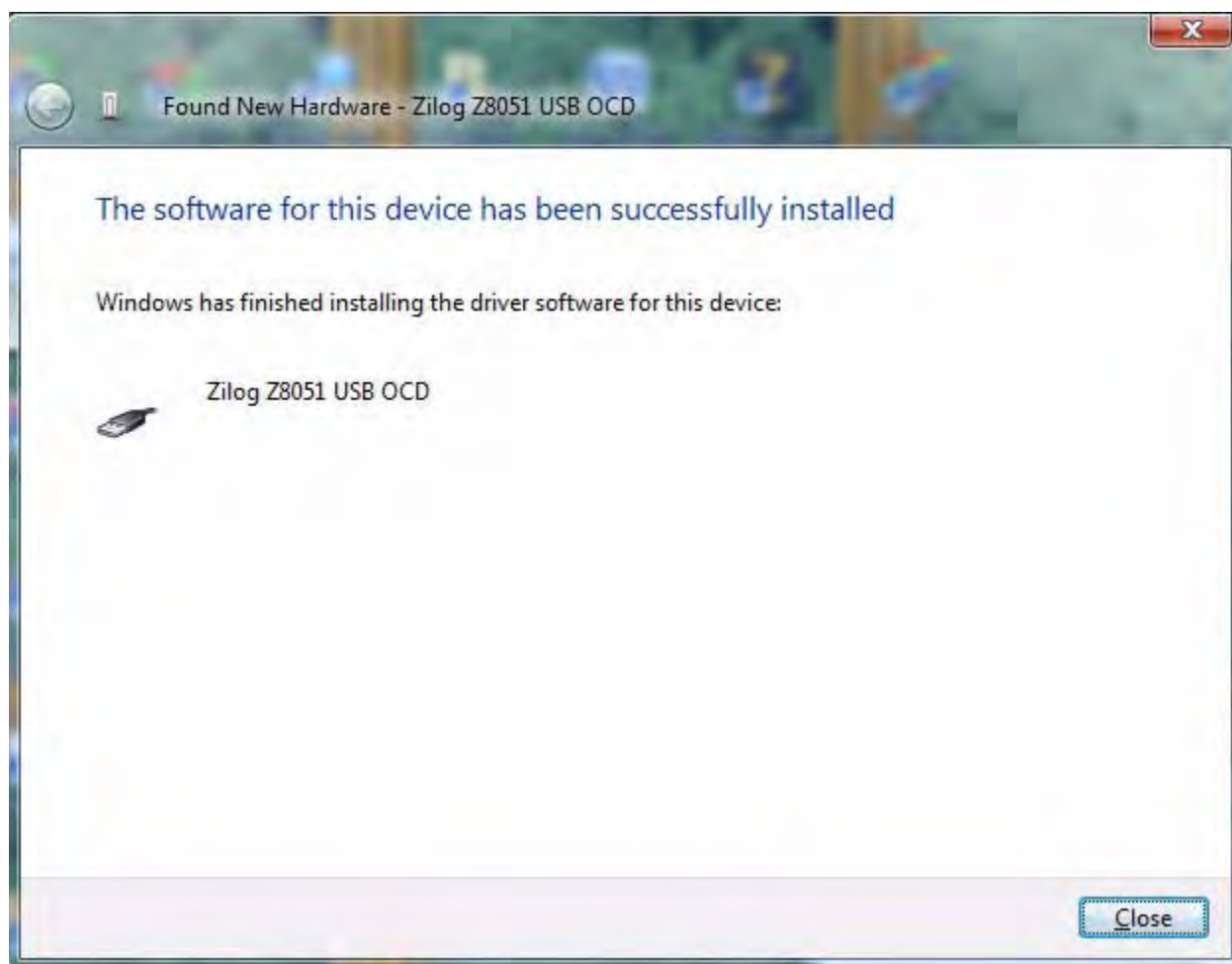
Figure 92. Browse For Driver Dialog, Windows Vista

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 93. Can't Verify Publisher Dialog, Windows Vista

7. When the installation is complete, the screen shown in Figure 94 will appear. Click **Close** to exit the OCD driver installation.



**Figure 94. Successfully Installed Dialog, Windows Vista**

---

► **Note:** On some installations, the Found New Hardware screen shown in Figure 94 may also display a `No Firmware` text string. This occurrence is normal and can be disregarded.

---

## 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 95. Click within this dialog to display the installation sequence of the driver software, which is diagrammed from top to bottom in Figure 96.



Figure 95. Install Device Driver Dialog, Windows 7

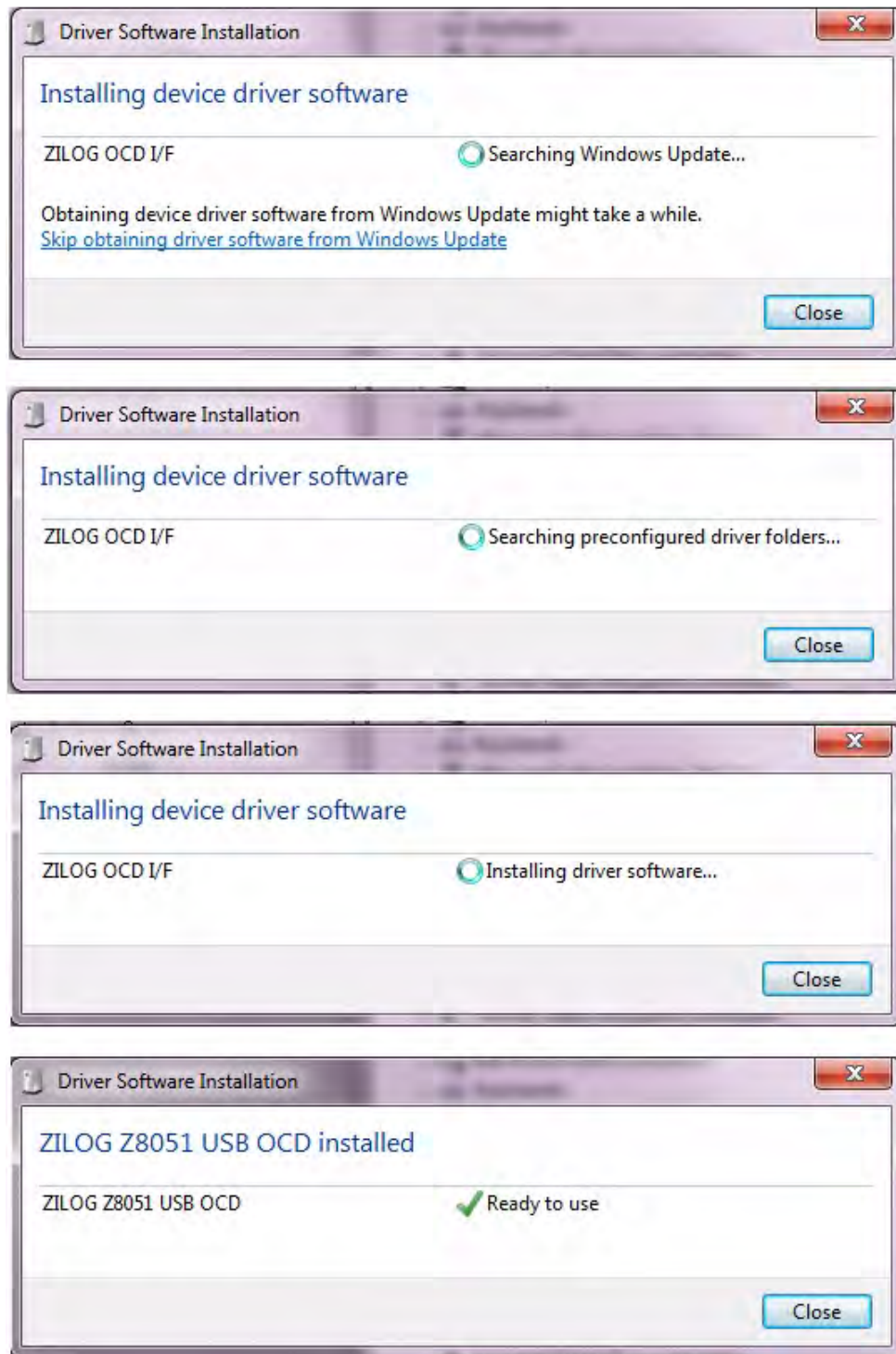


Figure 96. Driver Software Installation Dialog, Windows 7



- **Note:** If you should encounter the scenarios shown in Figures 97 or 98, right-click your mouse on **ZILOG OCD I/F** (highlighted in Figure 97) or **Unknown device** (highlighted in Figure 98) and select **Update Drive Software...** Then, follow [Steps 5 through 7](#) in [Appendix A](#) on page 79 to complete the driver installation.

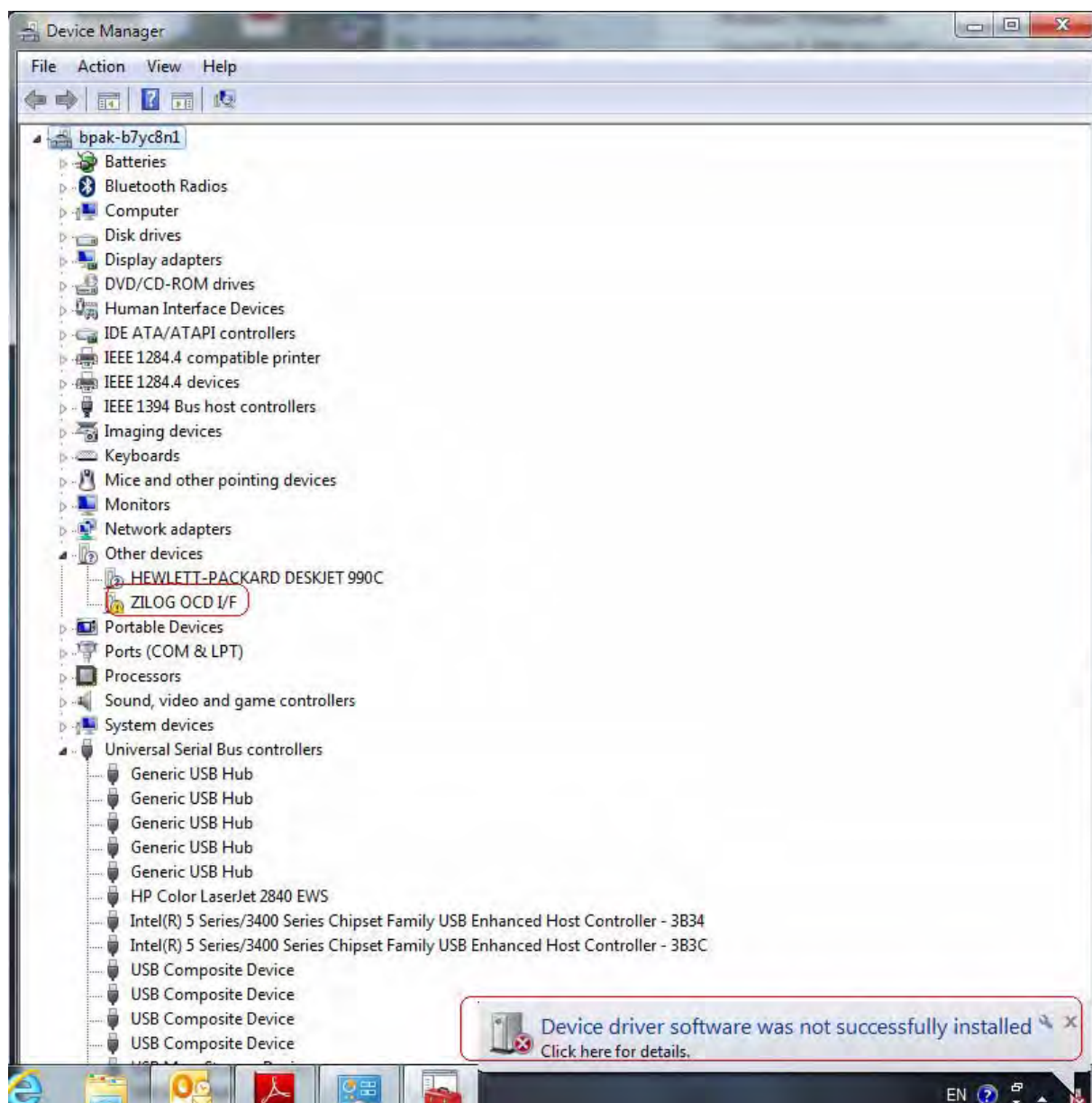


Figure 97. Unsuccessful Installation, Scenario 1

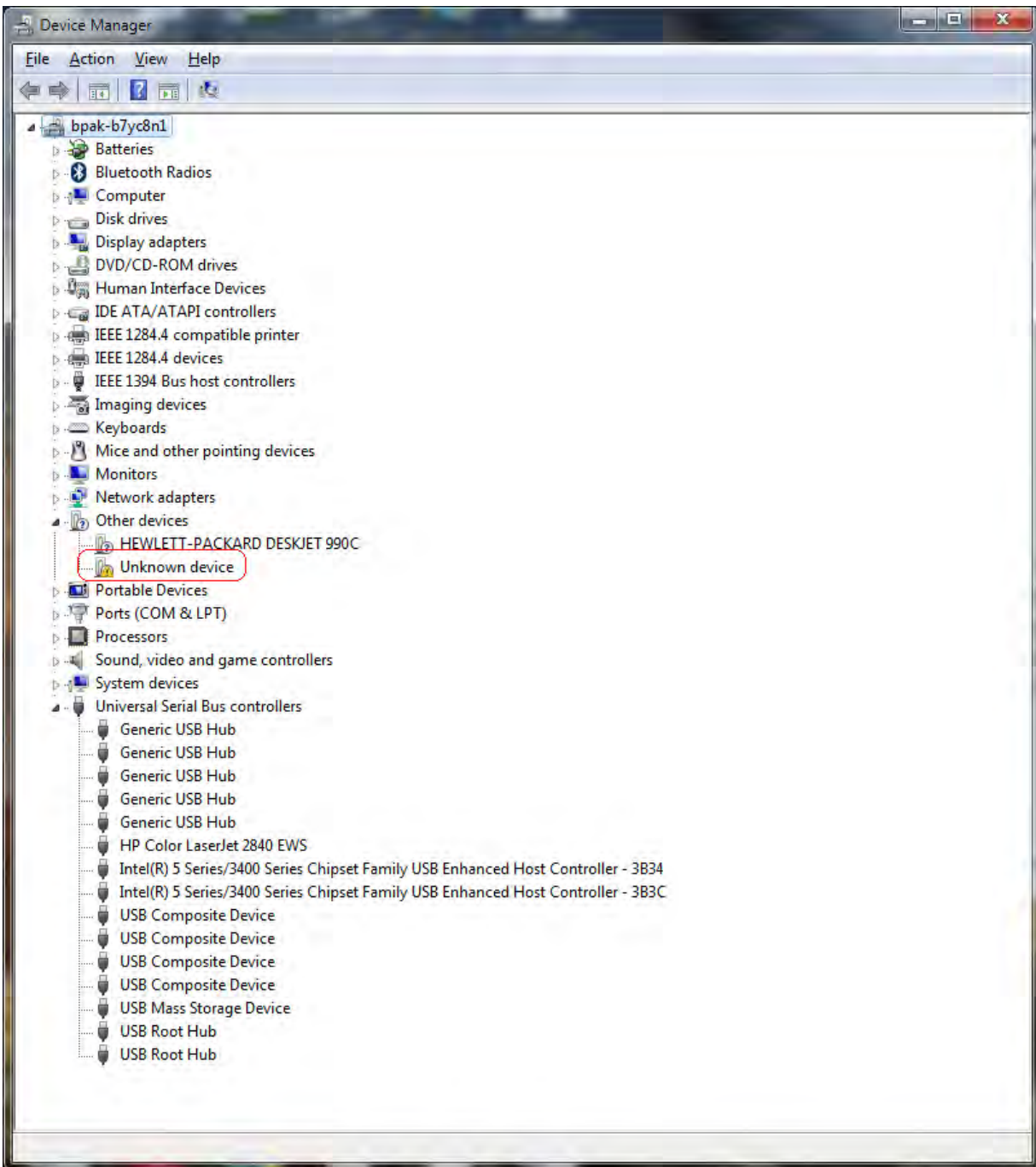


Figure 98. Unsuccessful Installation, Scenario 2



3. If *Zilog Z8051 USB OCD* appears in the Device Manager (as highlighted in Figure 99), the OCD driver software has been successfully installed.

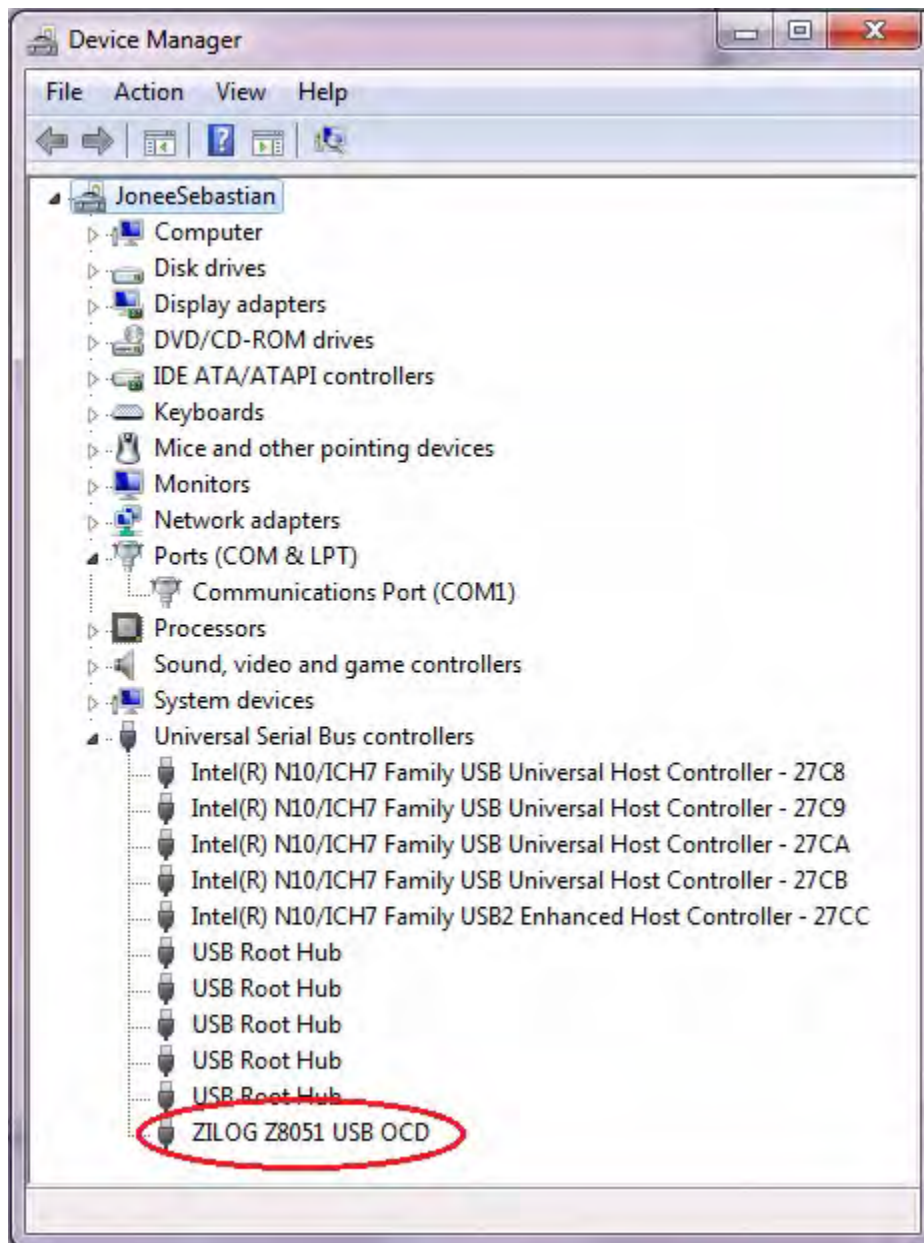


Figure 99. Device Manager Dialog, Windows 7

# 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 <http://support.zilog.com>.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the [Zilog Knowledge Base](#) or consider participating in the [Zilog Forum](#).

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