Z8051® Family of 8-Bit Microcontrollers

Z8051 On-Chip Debugger II

User Manual

UM027001-0215

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

LIFE SUPPORT POLICY

As used herein

Document Disclaimer

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
Feb 2015	01	Original issue.	All

Table of Contents

List of Figures

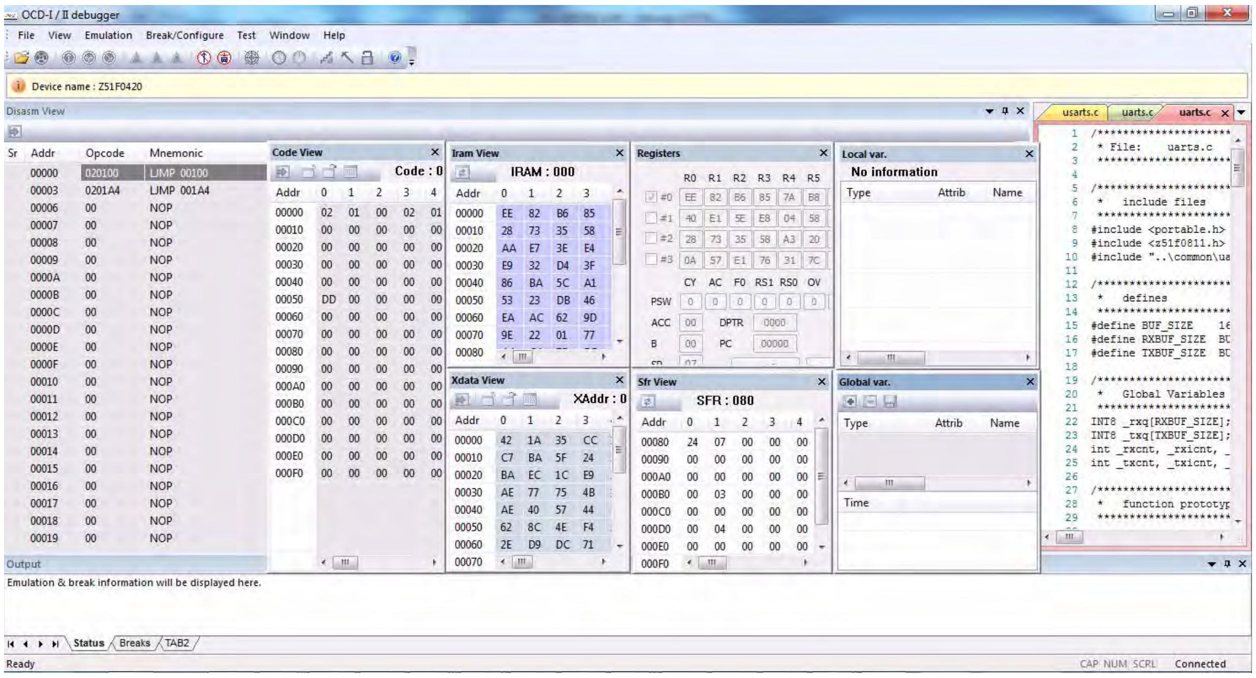
Introduction

Zilog's Z8051 On-Chip Debugger II (OCDII) application has been developed to support Zilog's Z8051 MCUs when application development will occur on PCs. This document describes how to set up and use the Z8051 OCDII program with your Z8051 Development Kit.

The Z8051 OCDII enables a development PC to communicate with your target Z8051-based MCU; its interface dongle (Z51FOCD2000ZACG) is used to connect this development PC to the Z8051 MCU. The OCDII 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 OCDII supports emulation and debugging at the maximum frequency of the MCU, thereby eliminating the requirement for an expensive emulator system.

The Z8051 OCDII Debugger works with the Microsoft Windows XP, Vista (32/64) and Windows 7 (32/64) operating systems. An example On-Chip Debugger II screen is shown in Figure 1.



Device name : Z51F0420

```
Sr  Addr    Opcode   Mnemonic
    00000   020100   LJMP  00100
    00003   0201A4   LJMP  001A4
    00006   00       NOP
    00007   00       NOP
    00008   00       NOP
    00009   00       NOP
    0000A   00       NOP
    0000B   00       NOP
    0000C   00       NOP
    0000D   00       NOP
    0000E   00       NOP
    0000F   00       NOP
    00010   00       NOP
    00011   00       NOP
    00012   00       NOP
    00013   00       NOP
    00014   00       NOP
    00015   00       NOP
    00016   00       NOP
    00017   00       NOP
    00018   00       NOP
    00019   00       NOP
```

```
Code : 0
Addr    0   1   2   3   4
00000   02  01  00  02  01
00010   00  00  00  00  00
00020   00  00  00  00  00
00030   00  00  00  00  00
00040   00  00  00  00  00
00050   DD  00  00  00  00
00060   00  00  00  00  00
00070   00  00  00  00  00
00080   00  00  00  00  00
00090   00  00  00  00  00
000A0   00  00  00  00  00
000B0   00  00  00  00  00
000C0   00  00  00  00  00
000D0   00  00  00  00  00
000E0   00  00  00  00  00
000F0   00  00  00  00  00
```

```
IRAM : 000
Addr    0   1   2   3
00000   EE  82  B6  85
00010   28  73  35  58
00020   AA  E7  3E  E4
00030   E9  32  D4  3F
00040   86  BA  5C  A1
00050   53  23  DB  46
00060   EA  AC  62  9D
00070   9E  22  01  77
00080
```

```
         R0  R1  R2  R3  R4  R5
[✓] #0   EE  82  B6  85  7A  B8
[ ] #1   40  E1  5E  E8  04  58
[ ] #2   28  73  35  58  A3  20
[ ] #3   0A  57  E1  76  31  7C
         CY  AC  F0  RS1 RS0 OV
PSW      0   0   0   0   0   0
ACC      00      DPTR   0000
B        00      PC     00000
SP       07
```

```
Local var.
No information
Type          Attrib     Name
```

```
XAddr : 0
Addr    0   1   2   3
00000   42  1A  35  CC
00010   C7  BA  5F  24
00020   BA  EC  1C  E9
00030   AE  77  75  4B
00040   AE  40  57  44
00050   62  8C  4E  F4
00060   2E  D9  DC  71
00070
```

```
SFR : 080
Addr    0   1   2   3   4
00080   24  07  00  00  00
00090   00  00  00  00  00
000A0   00  00  00  00  00
000B0   00  03  00  00  00
000C0   00  00  00  00  00
000D0   00  04  00  00  00
000E0   00  00  00  00  00
000F0
```

```
Global var.
Type          Attrib     Name
Time
```

```
/*********************
 * File:    uarts.c
 *********************

/*********************
 *   include files
 *********************
#include <portable.h>
#include <z51f0811.h>
#include "..\common\ua

/*********************
 *   defines
 *********************
#define BUF_SIZE    16
#define RXBUF_SIZE  BU
#define TXBUF_SIZE  BU

/*********************
 *   Global Variables
 *********************
INT8 _rxq[RXBUF_SIZE];
INT8 _txq[TXBUF_SIZE];
int _rxcnt, _rxicnt, _
int _txcnt, _txicnt, _

/*********************
 *   function prototyp
 *********************
```

Emulation & break information will be displayed here.

Ready CAP NUM SCRL Connected

Figure 1. On-Chip Debugger II Screen

On-Chip Debugger I Software and MCUs

Zilog's On-Chip Debugger I (OCDI) software, formerly referred to simply as *OCD software,* controls the OCDI dongle that is connected to each OCDI-compatible MCU. This dongle is the original OCDI interface module that bridges the OCDI software to all OCDI-compatible MCUs.

The following OCDI-compatible MCUs utilize the OCDI dongle and software:

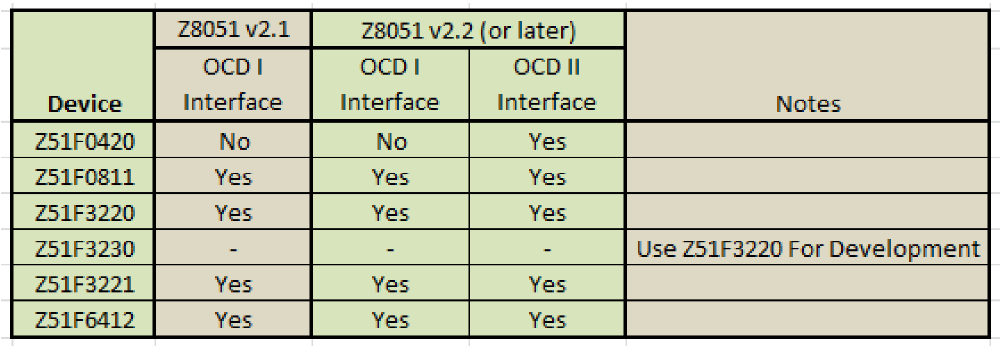
- Z51F6412
- Z51F3221
- Z51F3220
- Z51F0811

On-Chip Debugger II Software and MCUs

The On-Chip Debugger II (OCDII) software also controls the OCDI dongle that is connected to each OCDI-compatible MCU. Additionally, the OCDII software controls the OCDII dongle, which connects to all OCDI or OCDII MCUs. This dongle is an OCDII interface module that bridges the On-Chip Debugger II software to all OCDI-compatible and OCDII-compatible MCUs.

The Z51F0420 OCDII-compatible MCU utilizes the OCDII dongle and software.

Figure 2 shows the compatibility between Z8051 devices and software/OCD type interface.



	Z8051 v2.1	Z8051 v2.2 (or later)		
Device	OCD I Interface	OCD I Interface	OCD II Interface	Notes
Z51F0420	No	No	Yes	
Z51F0811	Yes	Yes	Yes	
Z51F3220	Yes	Yes	Yes	
Z51F3230	-	-	-	Use Z51F3220 For Development
Z51F3221	Yes	Yes	Yes	
Z51F6412	Yes	Yes	Yes	

Figure 2. Device, Software, and OCD Type Interface Compatibility

OCD II Features

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

- Supports Zilog's 8-bit Z8051 Family of MCUs
- Support for OCD I and OCD II MCUs and interface modules
- Loads HEX and map/symbol files
- Allows symbolic debugging
- Supports the internal code memory of the target MCU
- 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 II Software and Documentation

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

Note: If you followed the procedure on the paper insert in your kit and have already installed the Z8051 software and documentation, skip this section and proceed to the Z8051 OCD II Driver Installation section on the next page.

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

1. Ensure that the OCD II interface hardware is not connected to your PC.
2. In a web browser, download the Z8051 Software and Documentation v2.2 or later from the Zilog Store. When the download is complete, unzip the file to your hard drive and double-click to launch the installation file named `Z8051_<version_number>.exe`.

Note: In this filename, `<version_number>` refers to the version number of the OCD II Software and Documentation release. For example, this version number may be `2.2`.

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

Z8051 OCD II Driver Installation

The driver programs for the Z8051 On-Chip Debugger II 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 II will remain unconnected while you install these drivers.

Note: If you are running Windows Vista, see Appendix A on page 59 to install your device drivers. If you are running Windows XP, see Appendix B on page 62.

1. Connect the OCDII 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 OCDII device.
2. After the PC detects the new hardware, it will display the *Installing device driver software* dialog shown in Figure 3.

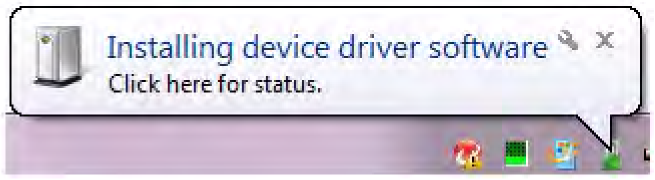

Installing device driver software
Click here for status.

Figure 3. The Install Device Driver Dialog, Windows 7

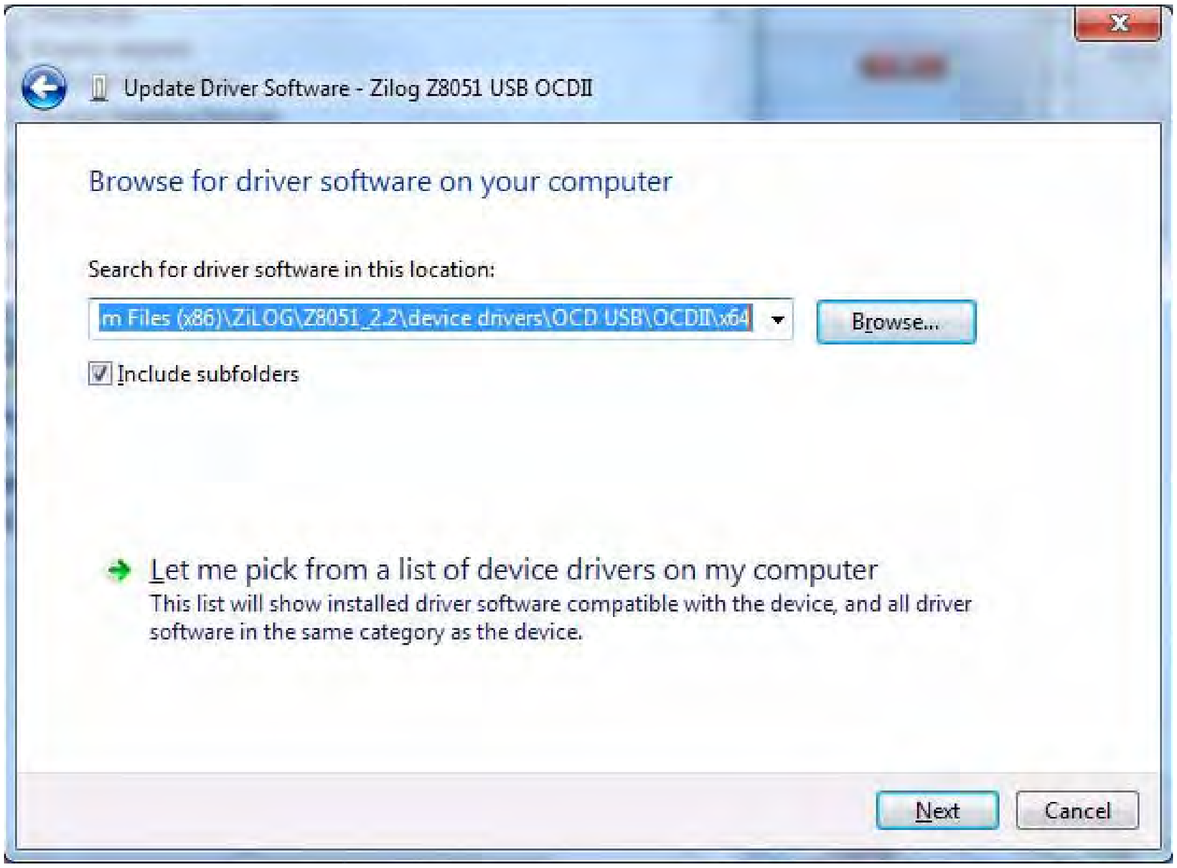
IMPORTANT NOTE: If you should encounter the scenarios presented in Figures 7 or 8, right-click your mouse on **ZILOG OCDII I/F** (highlighted in Figure 7) or **Unknown device** (highlighted in Figure 8) and select **Update Driver Software...**

3. Select **Browse my computer for driver software (advanced)** to display the dialog shown in Figure 4, 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 OCDI dongle, use the following driver path for either 32-bit or 64-bit machines:

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

   - On OCDII dongle, use the following driver path for either 32-bit or 64-bit machines:

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


Update Driver Software - Zilog Z8051 USB OCDII

Browse for driver software on your computer

Search for driver software in this location:

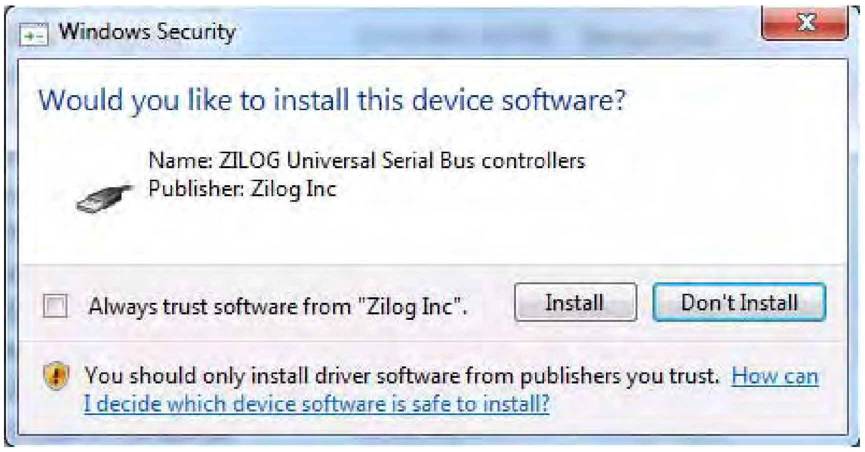
m Files (x86)\ZiLOG\Z8051_2.2\device drivers\OCD USB\OCDII\x64

[✓] Include subfolders

Browse...
Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.
Next
Cancel

Figure 4. The Browse For Driver Dialog

4. When Windows prompts you whether to install or not install, as shown in Figure 5, click **Install** and wait until the installation is completed.



Would you like to install this device software?

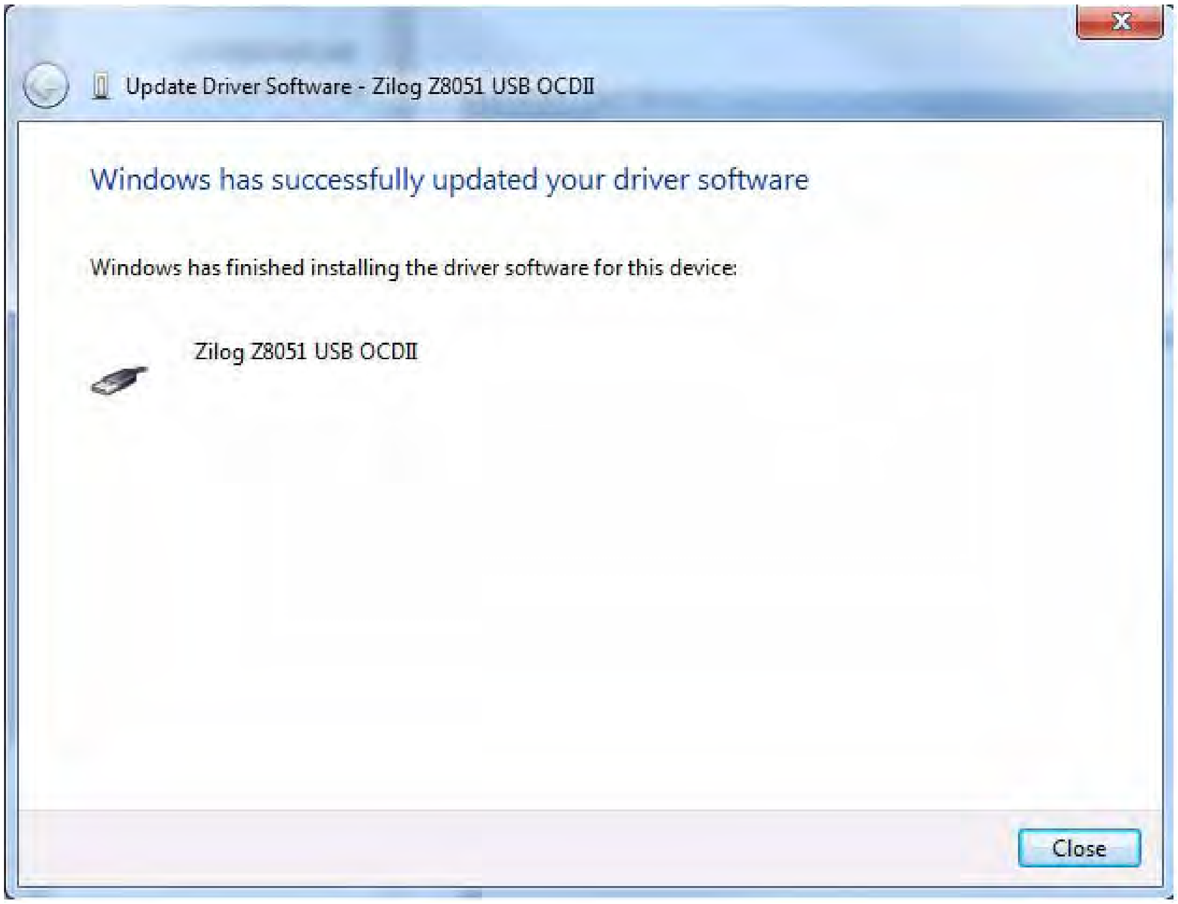
Name: ZILOG Universal Serial Bus controllers
Publisher: Zilog Inc

Always trust software from "Zilog Inc".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 5. The Would You Like to Install Dialog

5. When the installation is complete, the screen shown in Figure 6 will appear. Click **Close** to exit the OCDII driver installation.



Update Driver Software - Zilog Z8051 USB OCDII

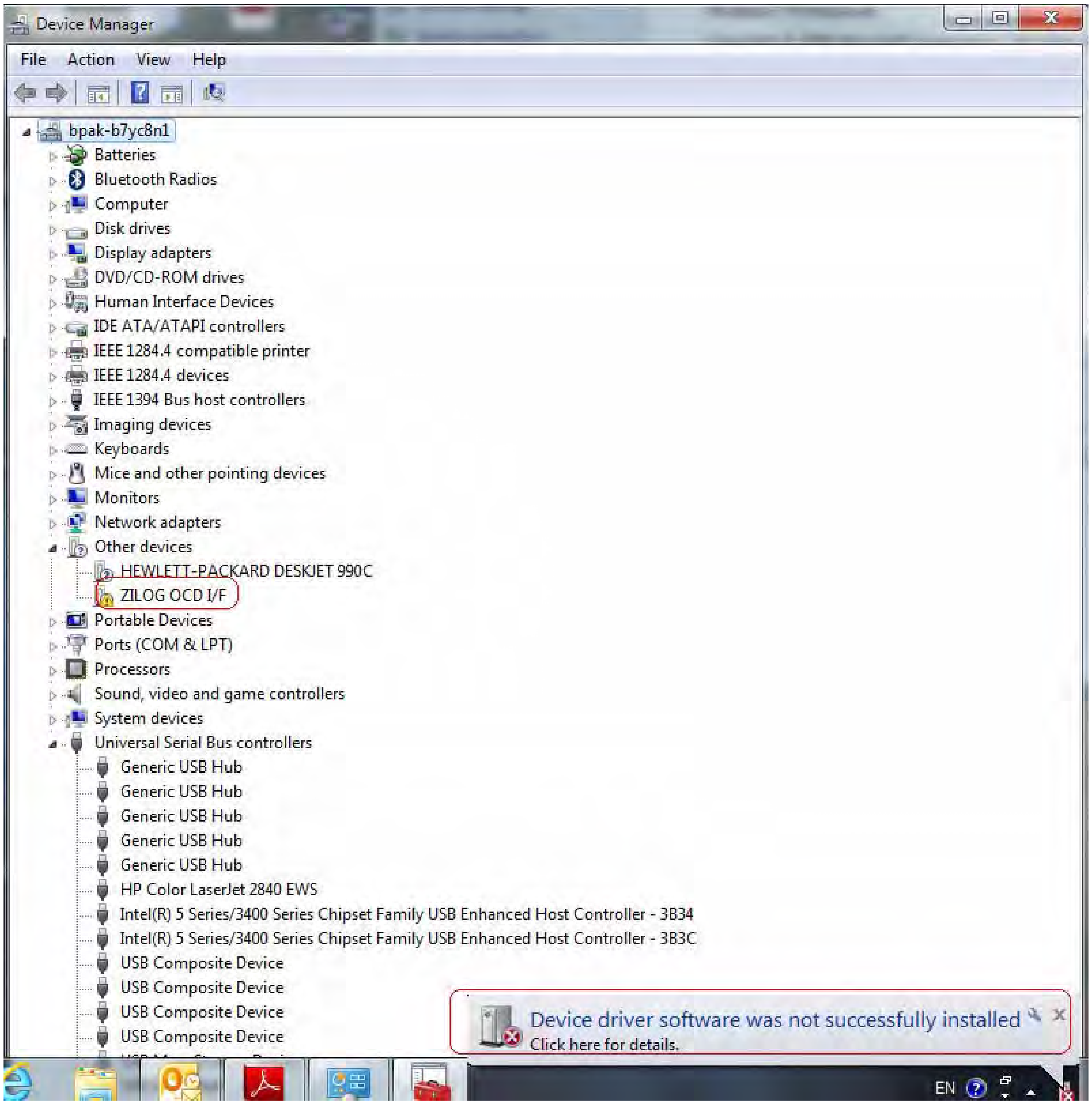
Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

Zilog Z8051 USB OCDII

Figure 6. The Successfully Installed Dialog

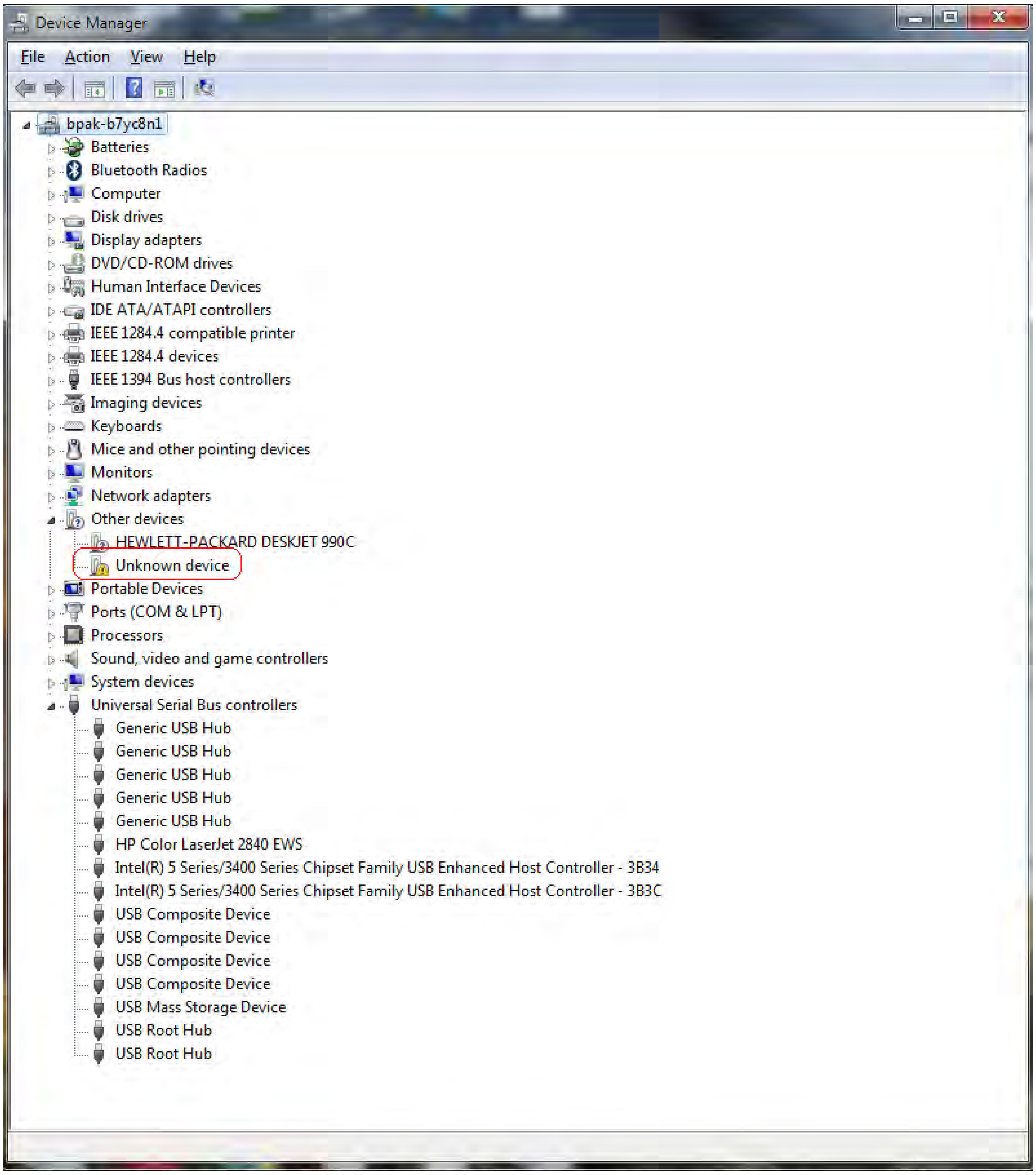
Note: On some installations, the Found New Hardware screen shown in Figure 6 may also display the text string, `Zilog Z8051 USB OCD - No Firmware`. This occurrence is normal and can be disregarded.



bpak-b7yc8n1
Batteries
Bluetooth Radios
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Human Interface Devices
IDE ATA/ATAPI controllers
IEEE 1284.4 compatible printer
IEEE 1284.4 devices
IEEE 1394 Bus host controllers
Imaging devices
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Other devices
HEWLETT-PACKARD DESKJET 990C
ZILOG OCD I/F
Portable Devices
Ports (COM & LPT)
Processors
Sound, video and game controllers
System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
Generic USB Hub
HP Color LaserJet 2840 EWS
Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34
Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C
USB Composite Device
USB Composite Device
USB Composite Device
USB Composite Device

Device driver software was not successfully installed
Click here for details.

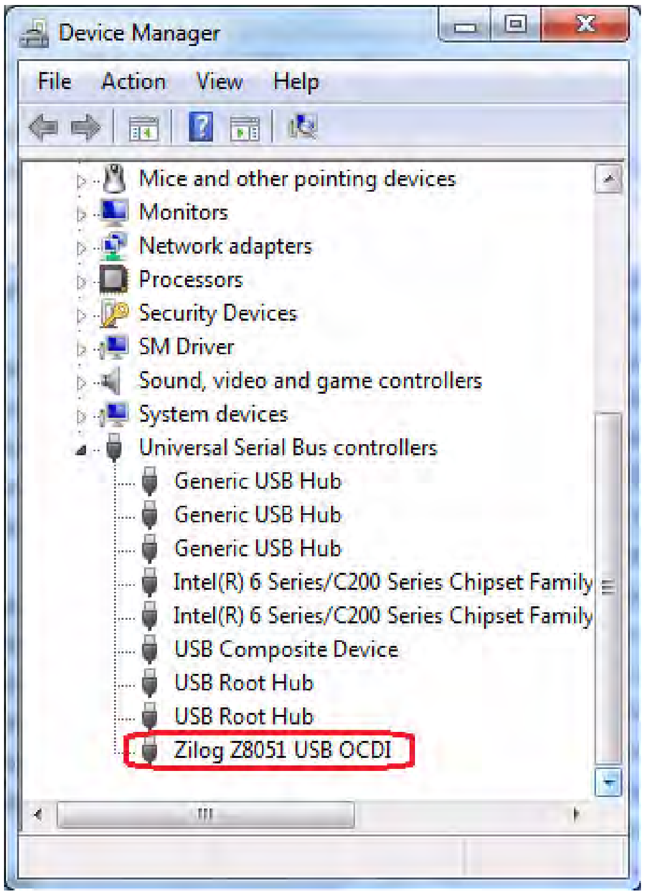
Figure 7. An Unsuccessful Installation, Scenario 1



bpak-b7yc8n1
- Batteries
- Bluetooth Radios
- Computer
- Disk drives
- Display adapters
- DVD/CD-ROM drives
- Human Interface Devices
- IDE ATA/ATAPI controllers
- IEEE 1284.4 compatible printer
- IEEE 1284.4 devices
- IEEE 1394 Bus host controllers
- Imaging devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Other devices
  - HEWLETT-PACKARD DESKJET 990C
  - Unknown device
- Portable Devices
- Ports (COM & LPT)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers
  - Generic USB Hub
  - Generic USB Hub
  - Generic USB Hub
  - Generic USB Hub
  - Generic USB Hub
  - HP Color LaserJet 2840 EWS
  - Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34
  - Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C
  - USB Composite Device
  - USB Composite Device
  - USB Composite Device
  - USB Composite Device
  - USB Mass Storage Device
  - USB Root Hub
  - USB Root Hub

Figure 8. An Unsuccessful Installation, Scenario 2

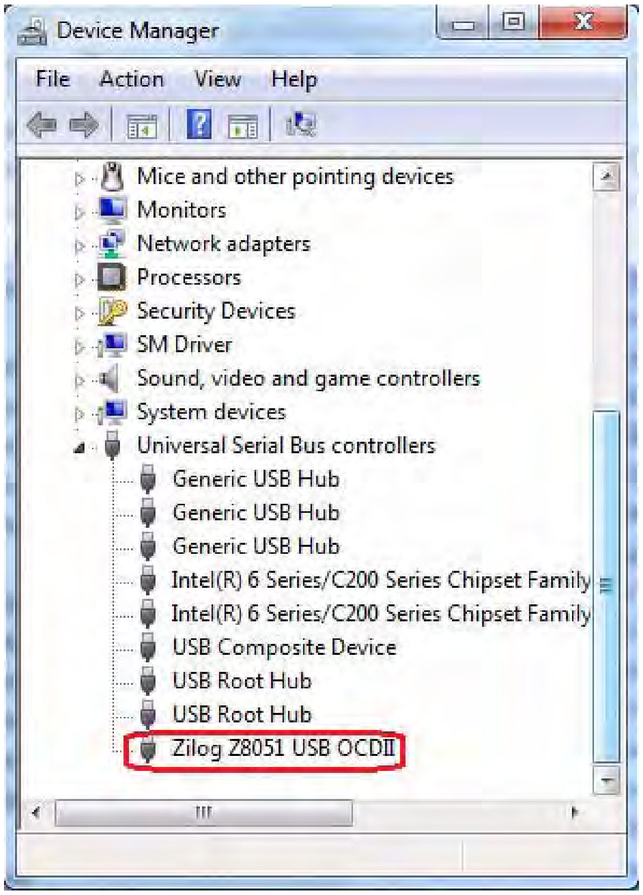
6. The OCDI or OCDII driver software has been successfully installed when either of the following two conditions occur:
   a. If *Zilog Z8051 USB OCDI* appears in the Device Manager when using the OCDI debug tool, as indicated in Figure 9.



Mice and other pointing devices
Monitors
Network adapters
Processors
Security Devices
SM Driver
Sound, video and game controllers
System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Generic USB Hub
Intel(R) 6 Series/C200 Series Chipset Family
Intel(R) 6 Series/C200 Series Chipset Family
USB Composite Device
USB Root Hub
USB Root Hub
Zilog Z8051 USB OCDI

Figure 9. The Device Manager Dialog for the OCDI Installation

   b. If *Zilog Z8051 USB OCDII* appears in the Device Manager when using the OCDII debug tool, as indicated in Figure 10.


Device Manager

File Action View Help

Mice and other pointing devices
Monitors
Network adapters
Processors
Security Devices
SM Driver
Sound, video and game controllers
System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Generic USB Hub
Intel(R) 6 Series/C200 Series Chipset Family
Intel(R) 6 Series/C200 Series Chipset Family
USB Composite Device
USB Root Hub
USB Root Hub
Zilog Z8051 USB OCDII

Figure 10. The Device Manager Dialog for the OCD II Installation

Understanding the OCD Menu Functions

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

File Menu

The File menu enables you to open files required for viewing. The OCD II File menu is shown in Figure 11.

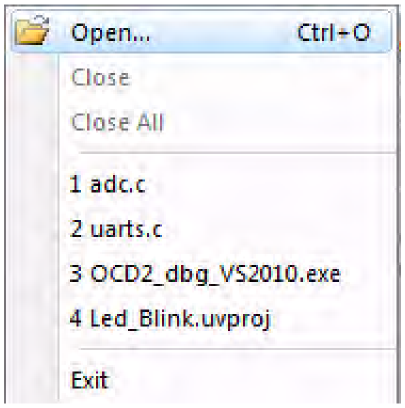


Figure 11. The OCD II File Menu

Emulation Menu

The Emulation menu, shown in Figure 12, lists the controls for starting or stopping an emulation routine. Use the Emulation menu to control the flow of code execution for debugging purposes.

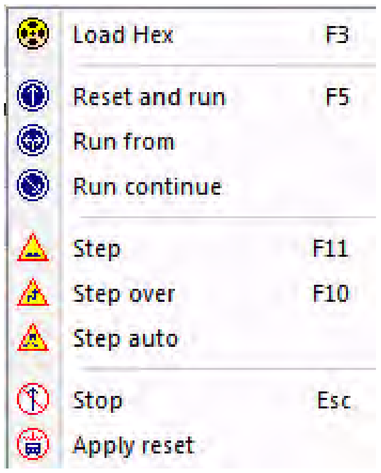


Figure 12. The OCD II Emulation Menu

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

Load Hex

This Emulation menu selection downloads your hex file to the target device; simply click the Download button in the Hex file download dialog box. An example is shown in Figure 70 on page 66.

Note: To load a hex file to the target MCU, see the procedure described in Appendix C on page 65.

Reset and Run

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 and Run menu is disabled (greyed out) in the Emulation menu during emulation.

Run From

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

Run Continue

The Run Continue 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 Run Continue 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.

Stop

Using the Stop 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.

Apply Reset

The Apply 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 Apply 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 13, lists PC breakpoint control, device configuration and hardware test functions.

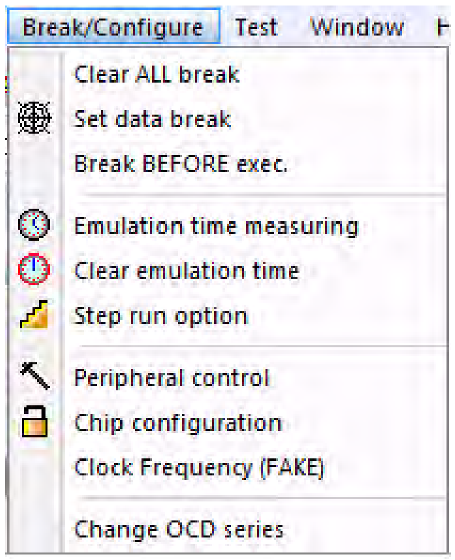


Figure 13. The OCDII Break/Configure Menu

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.

Set Data Break

This menu selection lets you add, remove, or change a data break (also called a breakpoint). See Figure 14.

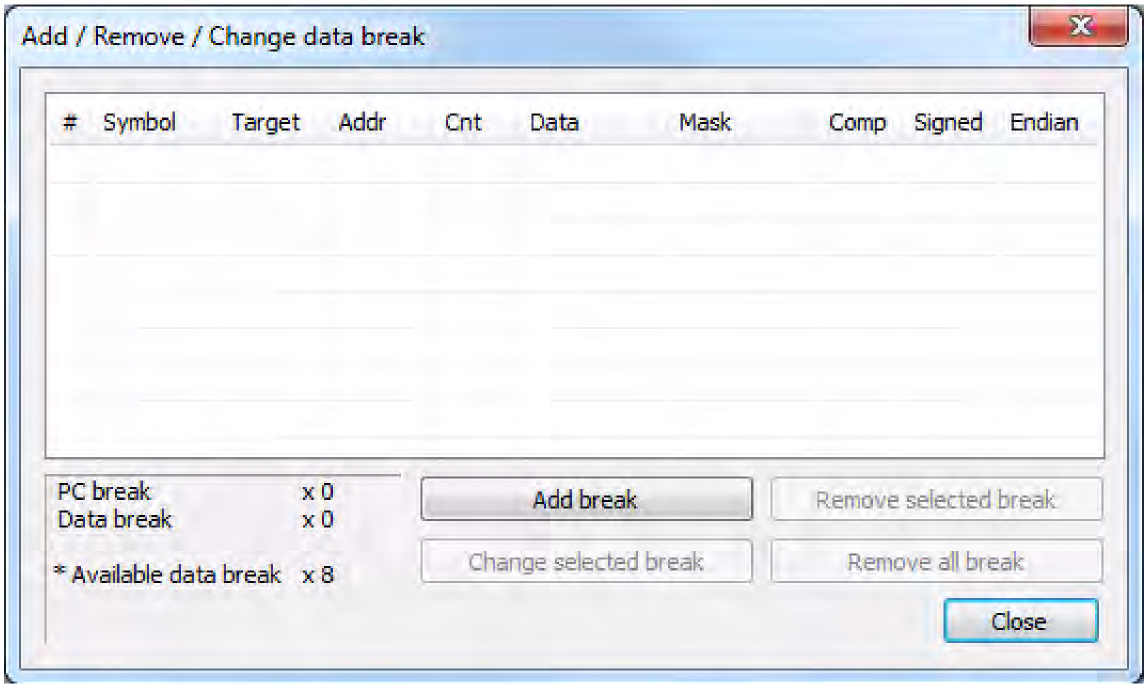


Figure 14. The Add/Remove/Change Data Break Dialog

Break BEFORE (AFTER)

The Break BEFORE (AFTER) function prompts the user to select a PC breakpoint event either before or after execution. If a break after execution is desired, select **Break BEFORE (AFTER) exec.** from the Break/Configure menu until a checkmark appears beside the Break AFTER exec. selection. Make this selection again to remove this checkmark if a break before execution is desired.

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 15.

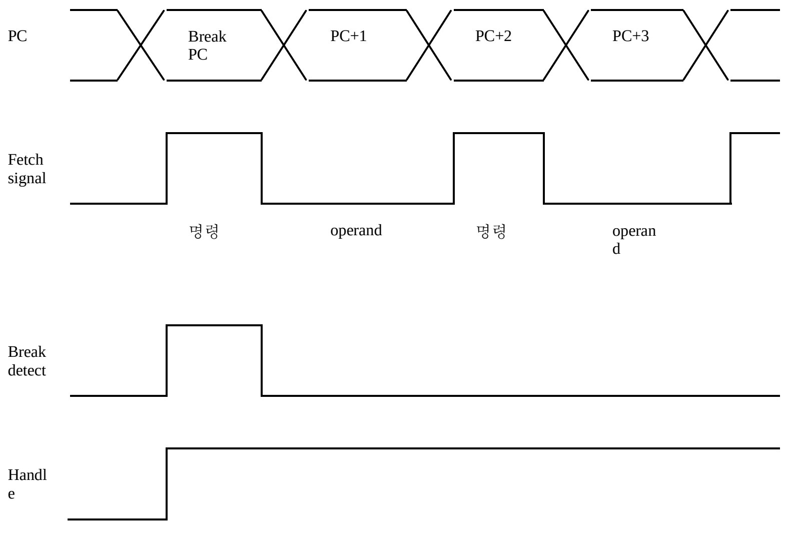


Figure 15. Break BEFORE Timing Diagram

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 16.

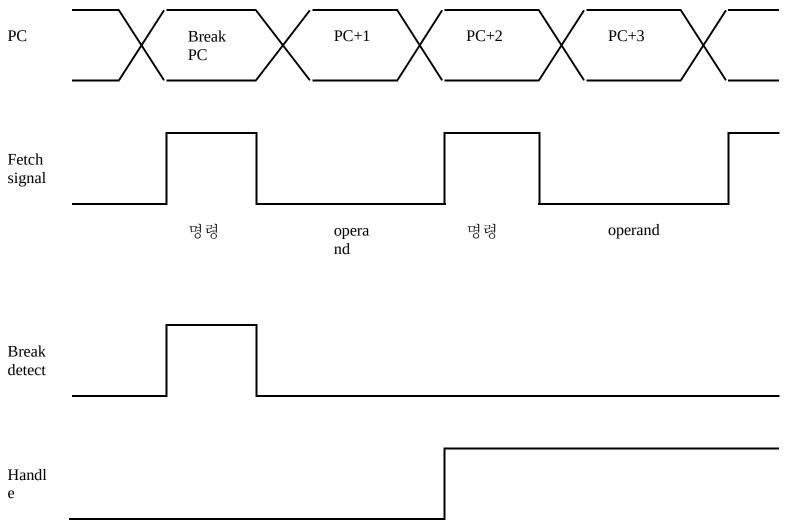


Figure 16. Break AFTER Timing Diagram

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

Emulation Time Measuring

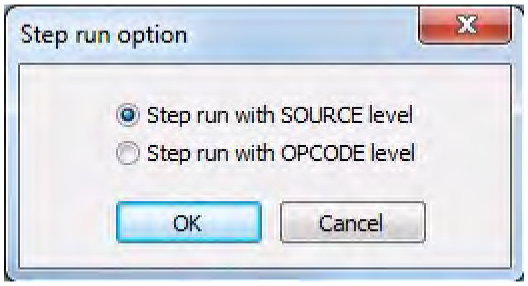
This menu, which works only for OCD II devices, changes the emulation time measuring option.

Clear Emulation Time

This menu, which works only for OCD II devices, clears the most recent emulation time and execution clock information.

Step Run Option

Selecting the Step run option function from the Break/Configure menu prompts the user to decide to step run to either of the SOURCE or OPCODE levels.

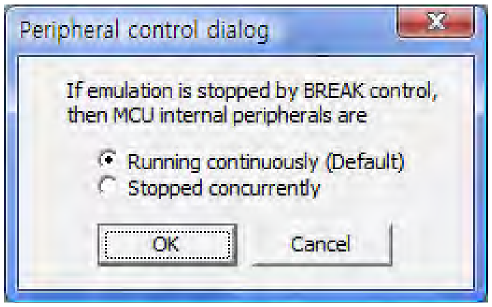


[✓ Step run with SOURCE level] Step run with OPCODE level

Figure 17. Step Run Option Dialog

Peripheral Control

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



If emulation is stopped by BREAK control, then MCU internal peripherals are

[✓ Running continuously (Default)]
Stopped concurrently

Figure 18. The Peripheral Control Dialog

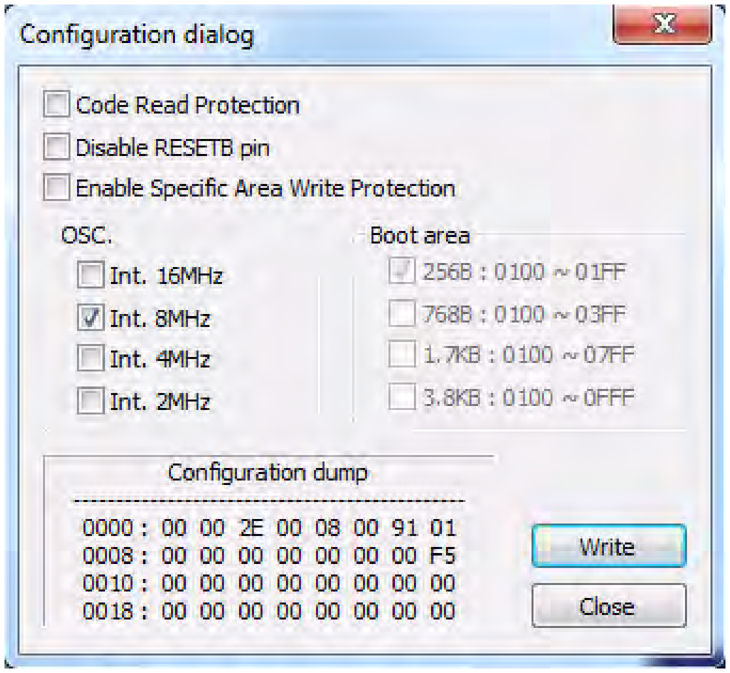
The purpose of the Peripheral Control function is to tell the OCDII whether the peripherals should be stopped or continuously run during Break (Debug) Mode. All peripherals, including the PLL and ADC functions, will be stopped when selecting **Stopped concurrently**. The Peripheral Control menu selection is disabled (greyed out) in the Break/Configure menu during emulation.

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

Chip Configuration

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

The Configuration dialog box shown in Figure 19 offers an example configuration for the Z51F0420 device.



☐ Code Read Protection
☐ Disable RESETB pin
☐ Enable Specific Area Write Protection

OSC.
☐ Int. 16MHz
☑ Int. 8MHz
☐ Int. 4MHz
☐ Int. 2MHz

Boot area
☑ 256B : 0100 ~ 01FF
☐ 768B : 0100 ~ 03FF
☐ 1.7KB : 0100 ~ 07FF
☐ 3.8KB : 0100 ~ 0FFF

```
Configuration dump
------------------------------------------------
0000 :  00  00  2E  00  08  00  91  01
0008 :  00  00  00  00  00  00  00  F5
0010 :  00  00  00  00  00  00  00  00
0018 :  00  00  00  00  00  00  00  00
```

Figure 19. Z51F0420 MCU Configuration Example

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

Clock Frequency (FAKE)

The OCDI interface does not support measuring emulation time. OCDII does support it, but the RTIME pin must first be connected. This clock frequency is FAKE, in that it is not the real emulation clock. This value will be used to calculate the emulation time from the device's clock count. If you have connected with OCDII and RTIME, this value will be ignored because the debugger can get the real emulation time.

Change OCD Series

To change the target series between OCDI and OCDII, select this option from the Break/ Configure menu.

View Menu

This section describes the components of the OCD II View menu.

Dockable Windows

The View menu, shown in Figure 20, supports the opening of child windows. If Child View is checked, the selected child view will be shown. The Debugger offers the following nine types of child windows.

- Output
- Diasm View
- Code View
- Xdata View
- Iram View
- Sfr View
- Registers
- Global Variable
- Local Variable

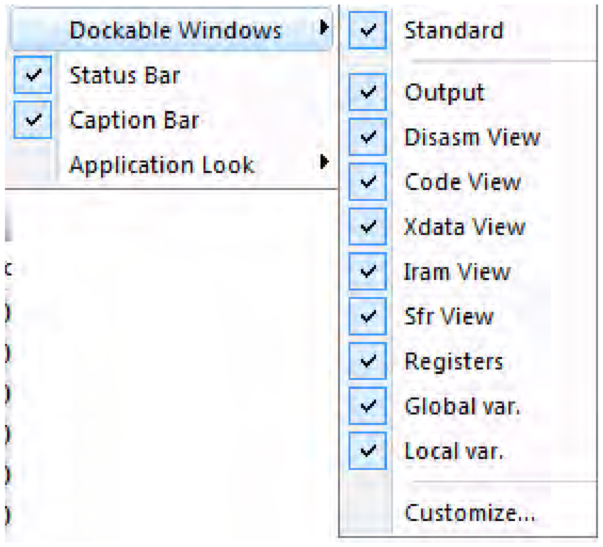


Figure 20. The OCDII View Menu

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

Customize

The **Customize...** selection offers users the flexibility to modify the Command, Toolbar, Keyboard, and Menu options in the debugger environment GUI.

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 information about the current state of the debugger.

Caption Bar

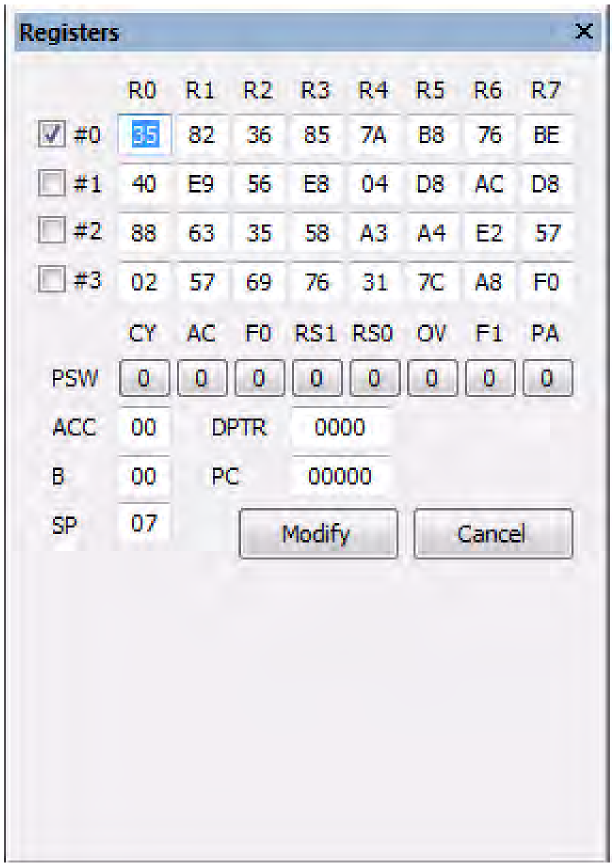
This menu selection displays or hides the caption bar, which is located at the bottom of the tool bars. The Caption bar displays the device name that is associated to either the OCD I or OCD II dongle hardware.

Application Look

This selection changes the visual style of the debugger GUI.

Registers

This menu selection opens a window that displays the Z8051 Series' basic registers. If this window is already open, selecting the **Registers** option will cause this window to disappear from the debugger frame. See Figure 21.



	R0	R1	R2	R3	R4	R5	R6	R7
✓ #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

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

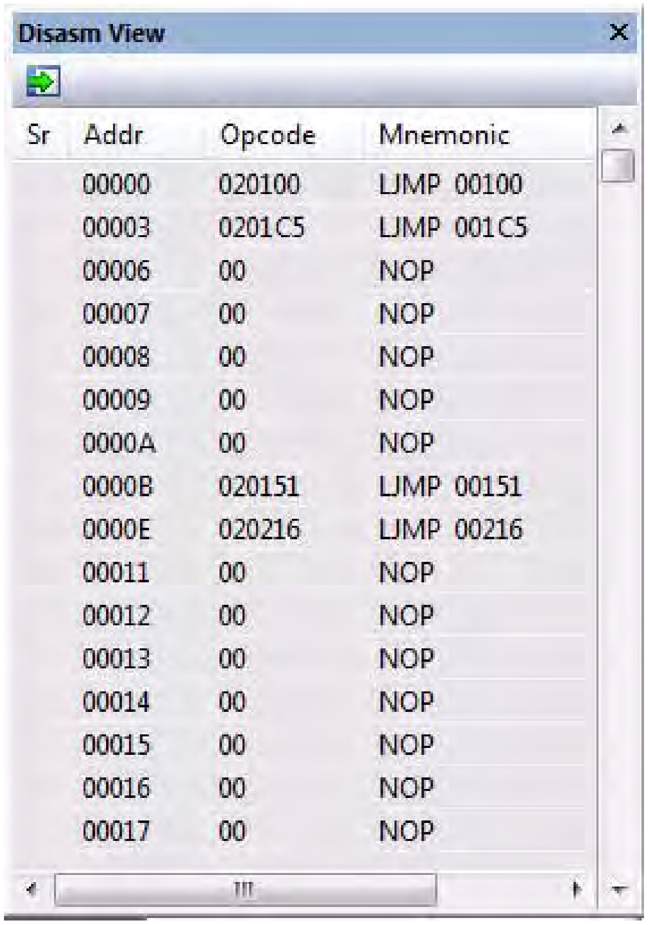
ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 21. The Registers Dialog

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

Disasm View

This menu selection opens a window which displays the memory spaces containing disassembled code. If this window is already open, selecting **Disasm View** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 22.



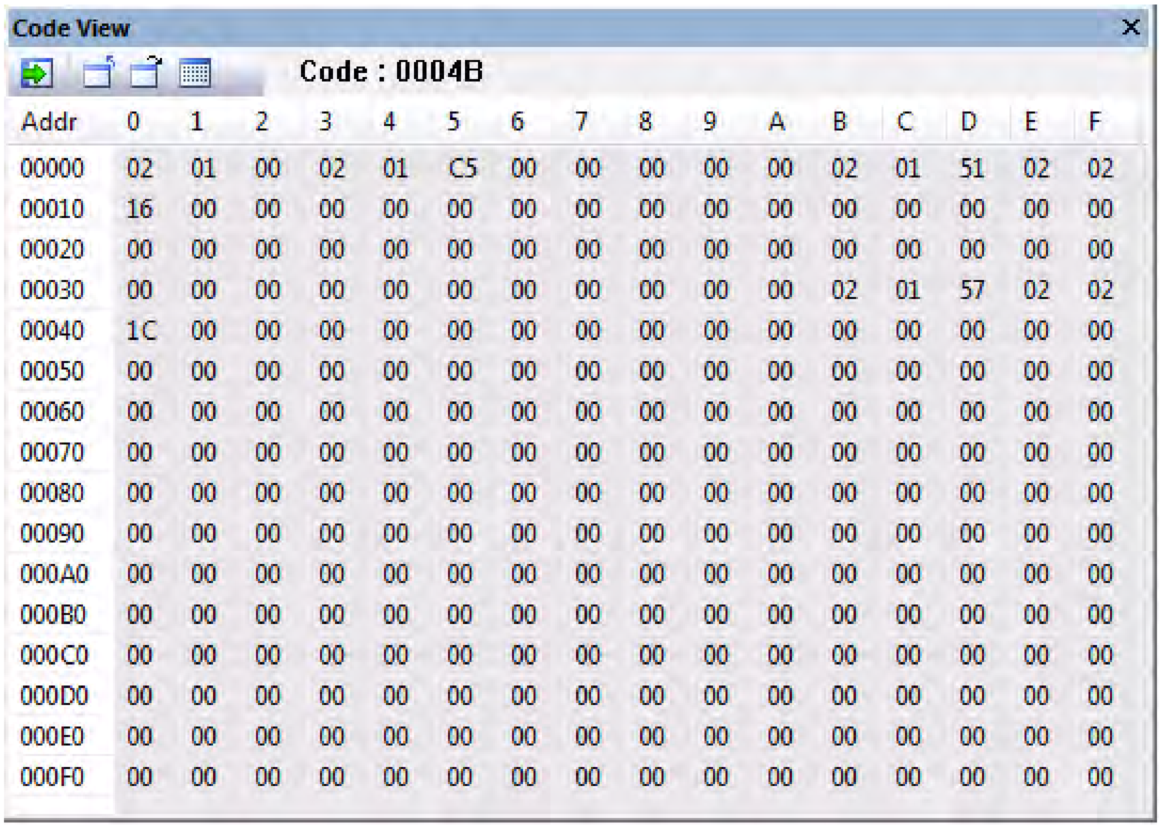
Sr	Addr	Opcode	Mnemonic
	00000	020100	LJMP 00100
	00003	0201C5	LJMP 001C5
	00006	00	NOP
	00007	00	NOP
	00008	00	NOP
	00009	00	NOP
	0000A	00	NOP
	0000B	020151	LJMP 00151
	0000E	020216	LJMP 00216
	00011	00	NOP
	00012	00	NOP
	00013	00	NOP
	00014	00	NOP
	00015	00	NOP
	00016	00	NOP
	00017	00	NOP

Figure 22. The Disasm View Dialog

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

Code View

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 View** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 23.



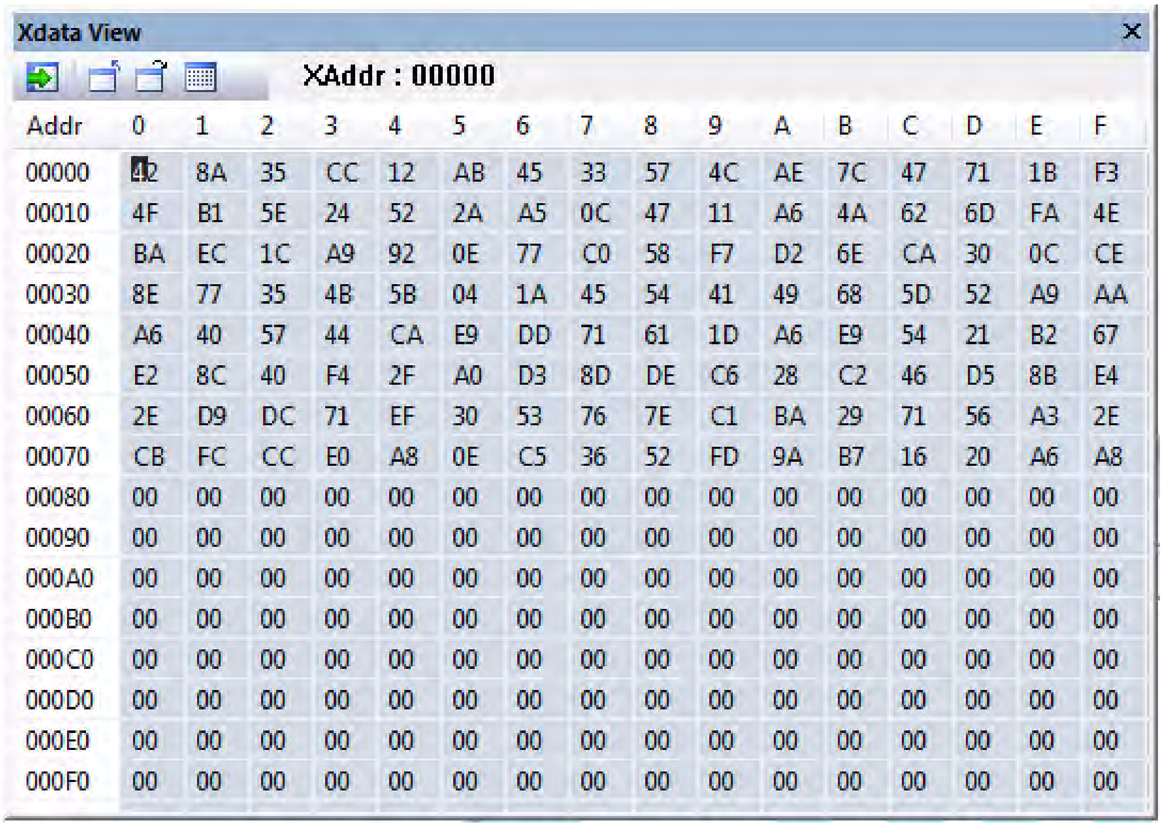
```
Code : 0004B

Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  02 01 00 02 01 C5 00 00 00 00 00 02 01 51 02 02
00010  16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030  00 00 00 00 00 00 00 00 00 00 00 02 01 57 02 02
00040  1C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 23. The Code View Dialog

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

Xdata View

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 View** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 24.



Xdata View

XAddr : 00000

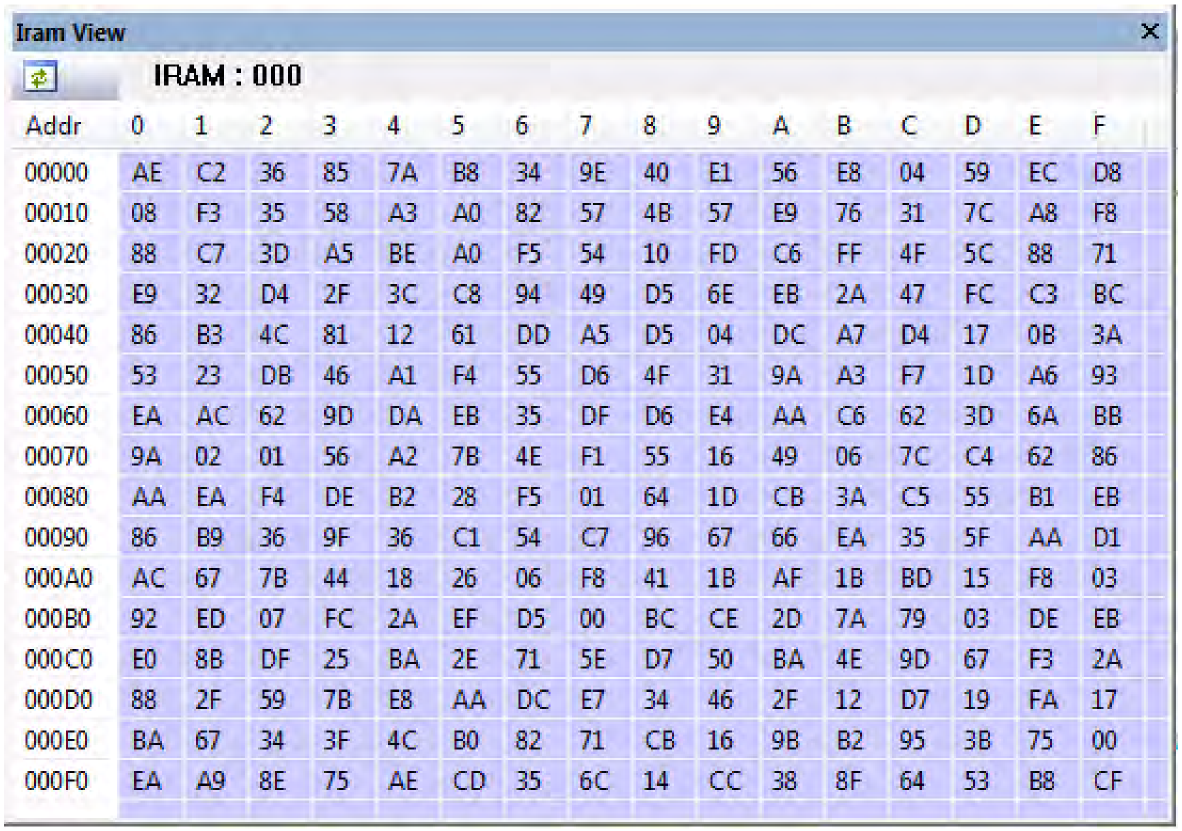
```
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  42 8A 35 CC 12 AB 45 33 57 4C AE 7C 47 71 1B F3
00010  4F B1 5E 24 52 2A A5 0C 47 11 A6 4A 62 6D FA 4E
00020  BA EC 1C A9 92 0E 77 C0 58 F7 D2 6E CA 30 0C CE
00030  8E 77 35 4B 5B 04 1A 45 54 41 49 68 5D 52 A9 AA
00040  A6 40 57 44 CA E9 DD 71 61 1D A6 E9 54 21 B2 67
00050  E2 8C 40 F4 2F A0 D3 8D DE C6 28 C2 46 D5 8B E4
00060  2E D9 DC 71 EF 30 53 76 7E C1 BA 29 71 56 A3 2E
00070  CB FC CC E0 A8 0E C5 36 52 FD 9A B7 16 20 A6 A8
00080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 24. The Xdata View Dialog

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

Iram View

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 View** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 25.


Iram View

IRAM : 000

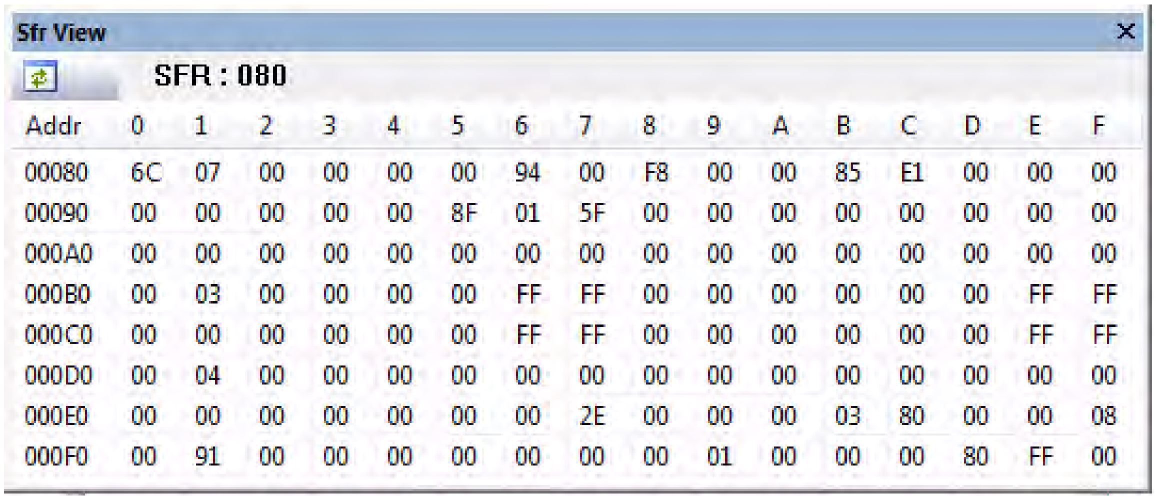
```
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  AE C2 36 85 7A B8 34 9E 40 E1 56 E8 04 59 EC D8
00010  08 F3 35 58 A3 A0 82 57 4B 57 E9 76 31 7C A8 F8
00020  88 C7 3D A5 BE A0 F5 54 10 FD C6 FF 4F 5C 88 71
00030  E9 32 D4 2F 3C C8 94 49 D5 6E EB 2A 47 FC C3 BC
00040  86 B3 4C 81 12 61 DD A5 D5 04 DC A7 D4 17 0B 3A
00050  53 23 DB 46 A1 F4 55 D6 4F 31 9A A3 F7 1D A6 93
00060  EA AC 62 9D DA EB 35 DF D6 E4 AA C6 62 3D 6A BB
00070  9A 02 01 56 A2 7B 4E F1 55 16 49 06 7C C4 62 86
00080  AA EA F4 DE B2 28 F5 01 64 1D CB 3A C5 55 B1 EB
00090  86 B9 36 9F 36 C1 54 C7 96 67 66 EA 35 5F AA D1
000A0  AC 67 7B 44 18 26 06 F8 41 1B AF 1B BD 15 F8 03
000B0  92 ED 07 FC 2A EF D5 00 BC CE 2D 7A 79 03 DE EB
000C0  E0 8B DF 25 BA 2E 71 5E D7 50 BA 4E 9D 67 F3 2A
000D0  88 2F 59 7B E8 AA DC E7 34 46 2F 12 D7 19 FA 17
000E0  BA 67 34 3F 4C B0 82 71 CB 16 9B B2 95 3B 75 00
000F0  EA A9 8E 75 AE CD 35 6C 14 CC 38 8F 64 53 B8 CF
```

Figure 25. The Iram View Dialog

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

Sfr View

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 View** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 26.



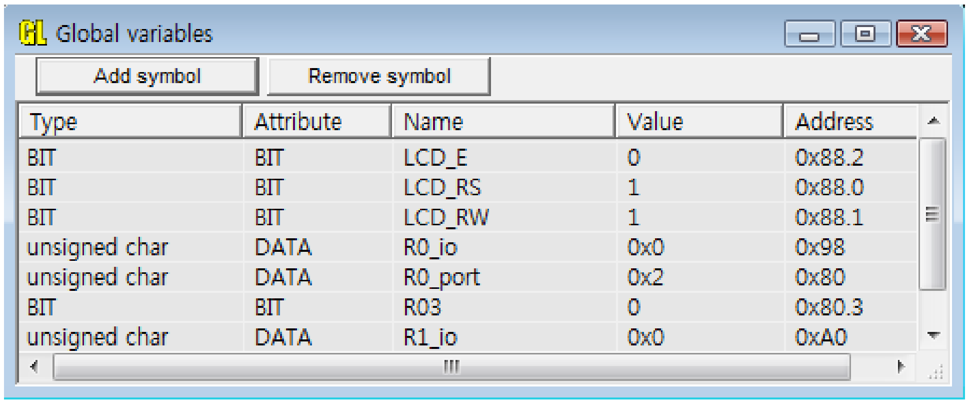
Addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00080	6C	07	00	00	00	00	94	00	F8	00	00	85	E1	00	00	00
00090	00	00	00	00	00	8F	01	5F	00	00	00	00	00	00	00	00
000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000B0	00	03	00	00	00	00	FF	FF	00	00	00	00	00	00	FF	FF
000C0	00	00	00	00	00	00	FF	FF	00	00	00	00	00	00	FF	FF
000D0	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000E0	00	00	00	00	00	00	00	2E	00	00	00	03	80	00	00	08
000F0	00	91	00	00	00	00	00	00	00	01	00	00	00	80	FF	00

Figure 26. The Sfr View Dialog

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

Global var.

This menu selection opens a window that displays global variables. If this window is already open, selecting **Global var.** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 27.



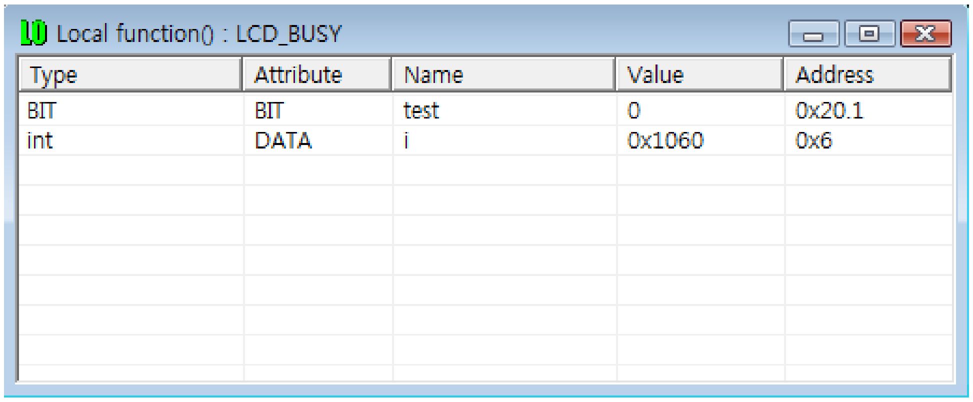
Type	Attribute	Name	Value	Address
BIT	BIT	LCD_E	0	0x88.2
BIT	BIT	LCD_RS	1	0x88.0
BIT	BIT	LCD_RW	1	0x88.1
unsigned char	DATA	R0_io	0x0	0x98
unsigned char	DATA	R0_port	0x2	0x80
BIT	BIT	R03	0	0x80.3
unsigned char	DATA	R1_io	0x0	0xA0

Figure 27. The Global Variables Dialog

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

Local var.

This menu selection opens a window that displays local variables. If this window is already open, selecting **Local var.** from the View menu will cause this window to disappear from the debugger frame. See the example in Figure 28.



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

Figure 28. The Local Function Dialog

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

Window Menu

The Window menu, shown in Figure 29, can be used to modify the arrangement of child windows or to directly select a child window.

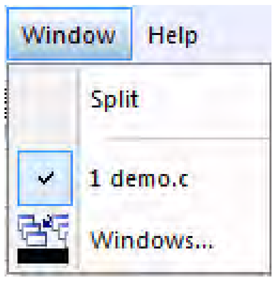
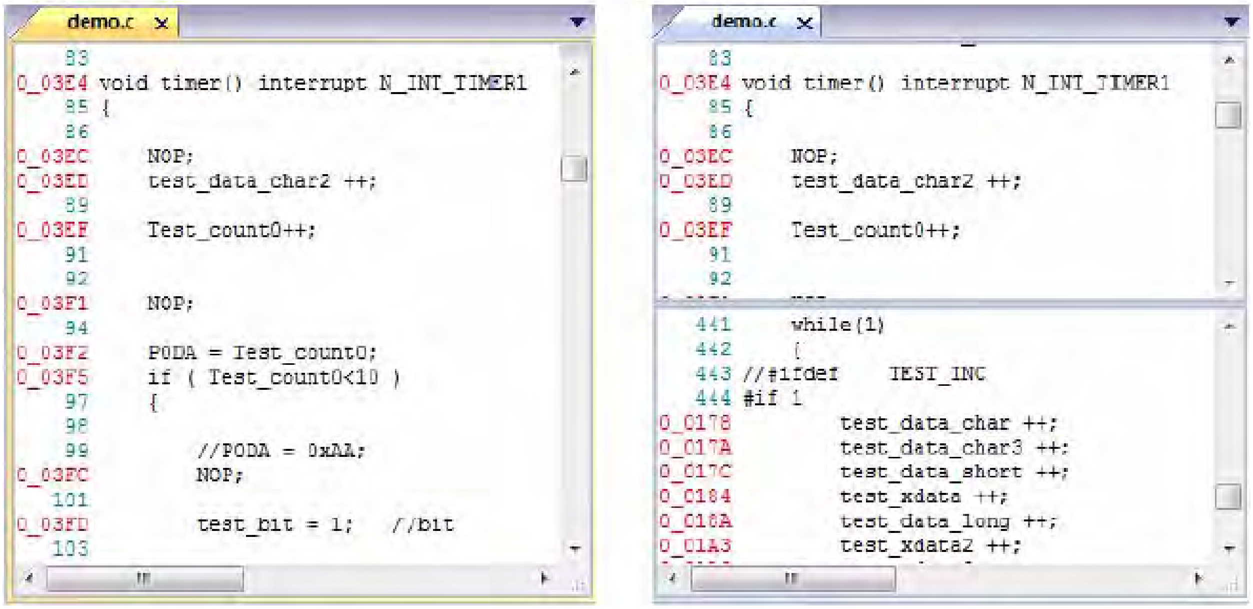


Figure 29. The OCD II Window Menu

Split

This menu selection splits the view of the text file windows, as shown in Figure 30. You can move or remove these split windows by dragging your mouse.



```
0_03E4 void timer() interrupt N_INT_TIMER1
       {

0_03EC     NOP;
0_03ED     test_data_char2 ++;

0_03EF     Test_count0++;


0_03F1     NOP;

0_03F2     P0DA = Test_count0;
0_03F5     if ( Test_count0<10 )
           {
               //P0DA = 0xAA;
0_03FC         NOP;

0_03FD         test_bit = 1;    //bit
```

```
0_03E4 void timer() interrupt N_INT_TIMER1
       {

0_03EC     NOP;
0_03ED     test_data_char2 ++;

0_03EF     Test_count0++;
```

```
           while(1)
           {
       //#ifdef    TEST_INC
       #if 1
0_0178         test_data_char ++;
0_017A         test_data_char3 ++;
0_017C         test_data_short ++;
0_0184         test_xdata ++;
0_018A         test_data_long ++;
0_01A3         test_xdata2 ++;
```

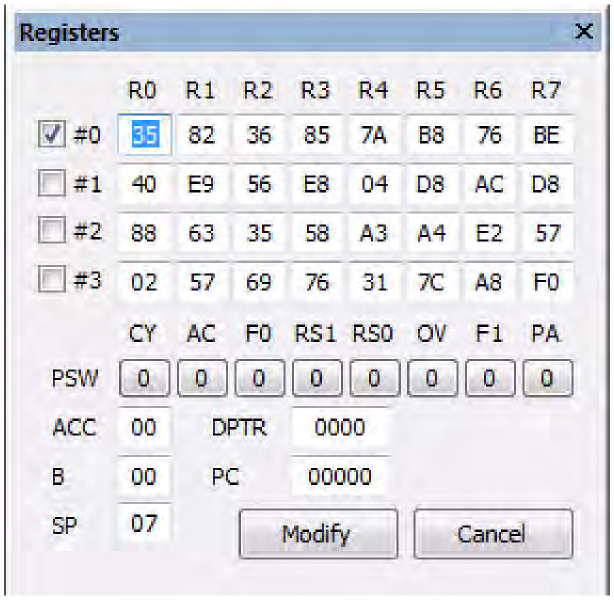
Figure 30. Split Windows

Child Windows

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

Registers Window

The Registers window allows users to edit the contents of the Z8051 Series registers. Figure 31 shows an example Registers window.



	R0	R1	R2	R3	R4	R5	R6	R7
✓ #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

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

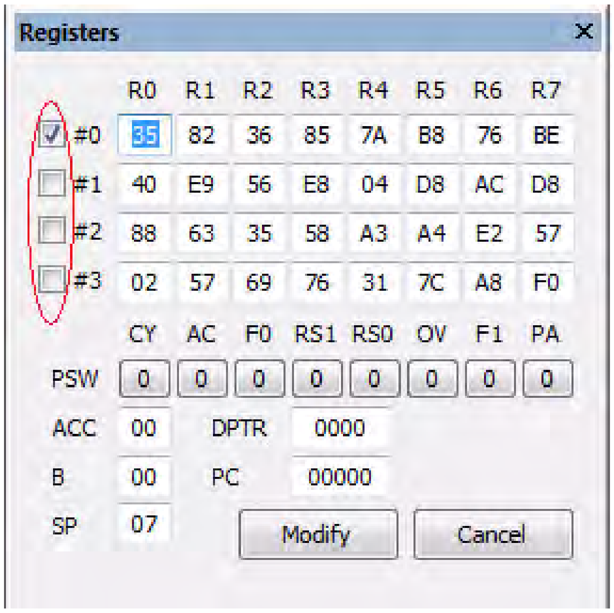
ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 31. Using the Registers Function, #1 of 6

Edit

Register values can be changed in the Edit window. New register values are downloaded to the target MCU upon clicking the **Modify** button.

In Figure 32, 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.


Registers

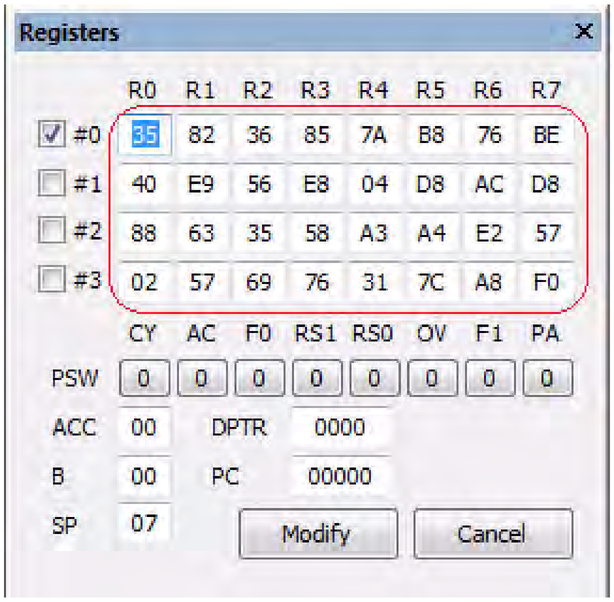
	R0	R1	R2	R3	R4	R5	R6	R7
[✓] #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

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

ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 32. Using the Registers Function, #2 of 6

In Figure 33, 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.



Registers

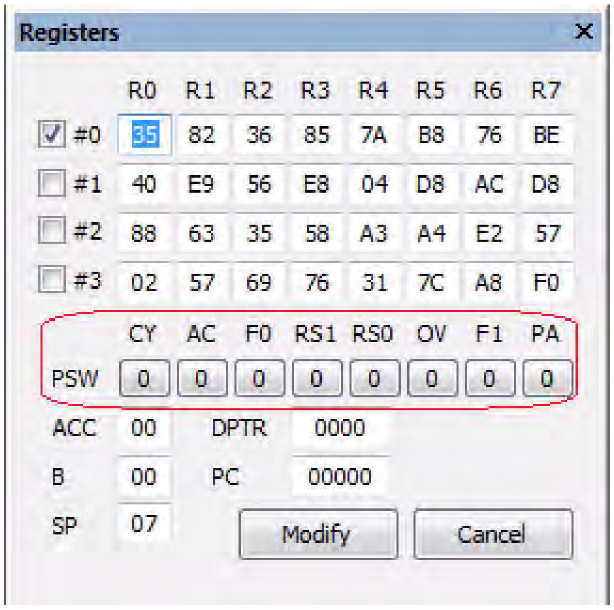
	R0	R1	R2	R3	R4	R5	R6	R7
✓ #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

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

ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 33. Using the Registers Function, #3 of 6

In Figure 34, the red area highlights the Program Status Word (PSW), in which bit units can be changed.



Registers

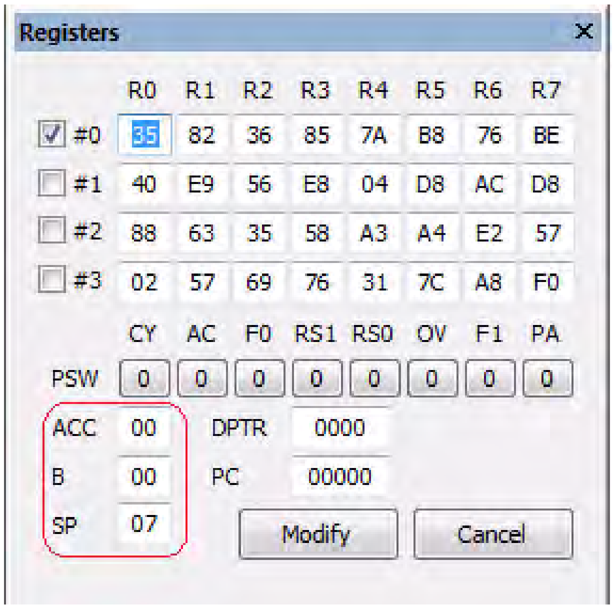
	R0	R1	R2	R3	R4	R5	R6	R7
✓ #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

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

ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 34. Using the Registers Function, #4 of 6

In Figure 35, 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.



Registers

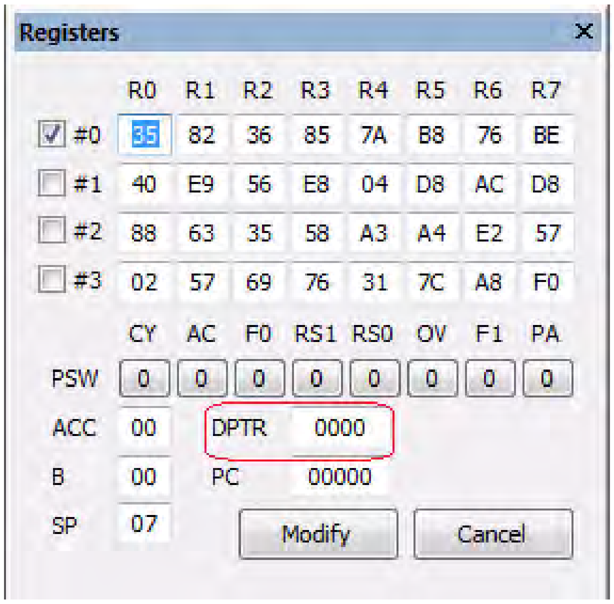
	R0	R1	R2	R3	R4	R5	R6	R7
✓ #0	35	82	36	85	7A	B8	76	BE
#1	40	E9	56	E8	04	D8	AC	D8
#2	88	63	35	58	A3	A4	E2	57
#3	02	57	69	76	31	7C	A8	F0

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

ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

Figure 35. Using the Registers Function, #5 of 6

In Figure 36, 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.



Registers

	R0	R1	R2	R3	R4	R5	R6	R7
☑ #0	35	82	36	85	7A	B8	76	BE
☐ #1	40	E9	56	E8	04	D8	AC	D8
☐ #2	88	63	35	58	A3	A4	E2	57
☐ #3	02	57	69	76	31	7C	A8	F0

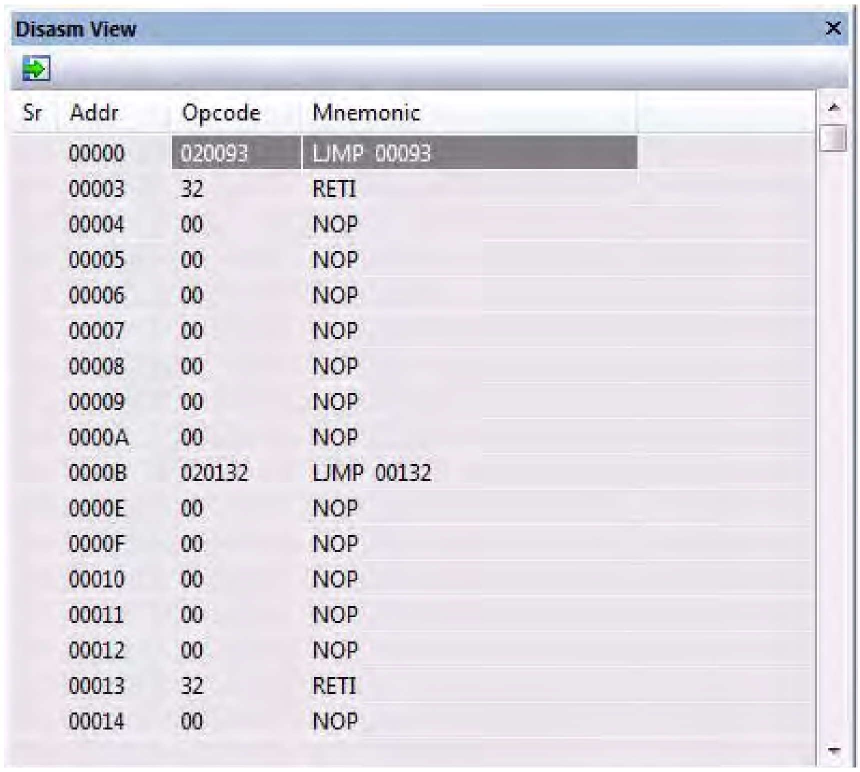
	CY	AC	F0	RS1	RS0	OV	F1	PA
PSW	0	0	0	0	0	0	0	0

ACC	00	DPTR	0000
B	00	PC	00000
SP	07		

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

Code Disassemble View

The Code Disassemble view displays the contents of code memory by using a disassemble format. All operand values must be entered in hexadecimal format. Figure 37 shows an example Code Disassembler view.

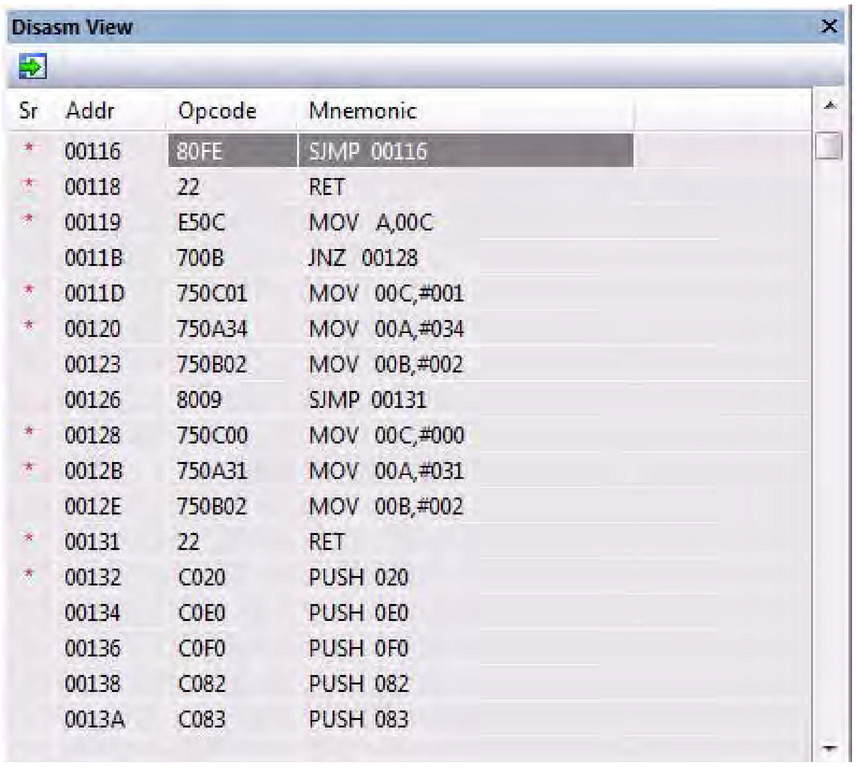


Disasm View

Sr	Addr	Opcode	Mnemonic
	00000	020093	LJMP 00093
	00003	32	RETI
	00004	00	NOP
	00005	00	NOP
	00006	00	NOP
	00007	00	NOP
	00008	00	NOP
	00009	00	NOP
	0000A	00	NOP
	0000B	020132	LJMP 00132
	0000E	00	NOP
	0000F	00	NOP
	00010	00	NOP
	00011	00	NOP
	00012	00	NOP
	00013	32	RETI
	00014	00	NOP

Figure 37. Using the Code Disassembler Function, #1 of 3

If map/symbol files are already loaded, the affected source lines are highlighted by red asterisks (*), as shown in Figure 38. Double-click any of these highlighted boxes to open its source file and move to the appropriate address line.



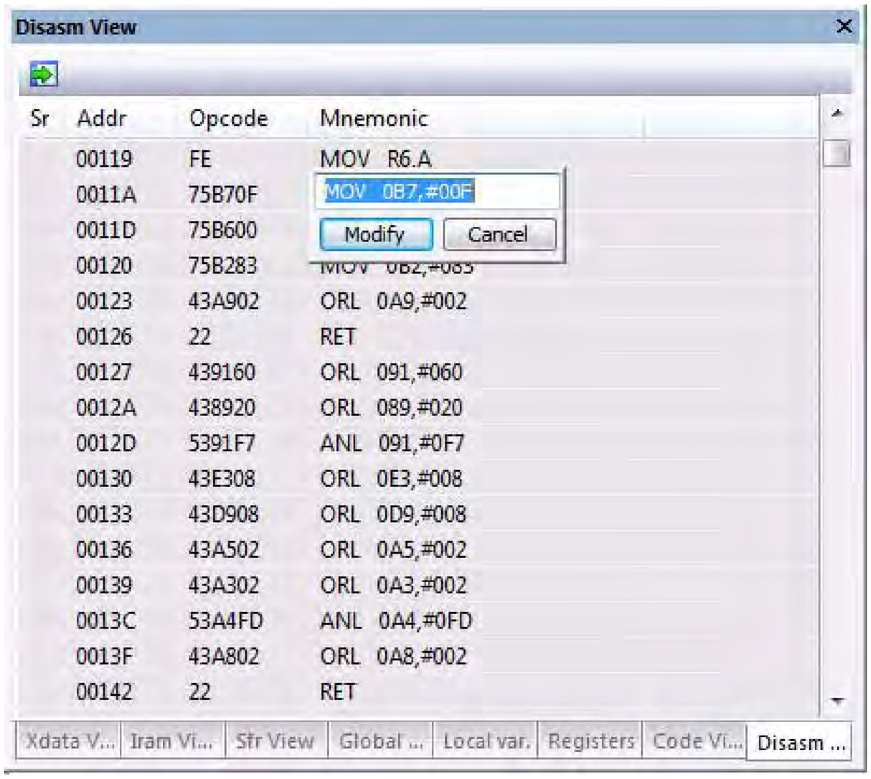
Sr	Addr	Opcode	Mnemonic
*	00116	80FE	SJMP 00116
*	00118	22	RET
*	00119	E50C	MOV A,00C
	0011B	700B	JNZ 00128
*	0011D	750C01	MOV 00C,#001
*	00120	750A34	MOV 00A,#034
	00123	750B02	MOV 00B,#002
	00126	8009	SJMP 00131
*	00128	750C00	MOV 00C,#000
*	0012B	750A31	MOV 00A,#031
	0012E	750B02	MOV 00B,#002
*	00131	22	RET
*	00132	C020	PUSH 020
	00134	C0E0	PUSH 0E0
	00136	C0F0	PUSH 0F0
	00138	C082	PUSH 082
	0013A	C083	PUSH 083

Figure 38. 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 double-click its Mnemonic field to open an edit field for the contents of that line, as shown in Figure 39. Change the contents of the line by entering an instruction, operand, etc., in hexadecimal format.



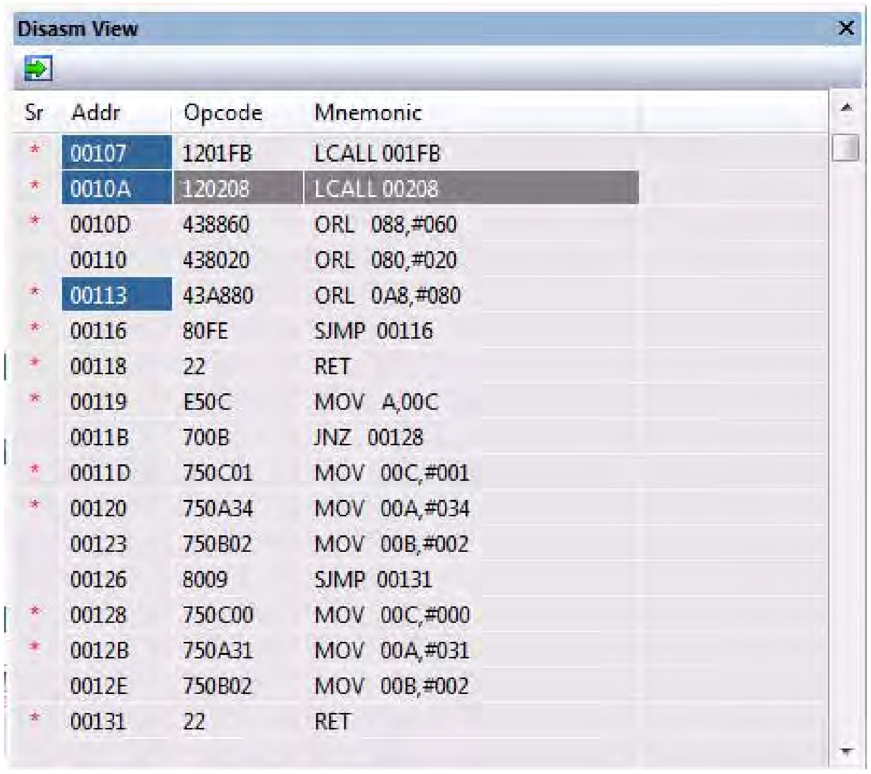
MOV 0B7,#00F

Sr	Addr	Opcode	Mnemonic
	00119	FE	MOV R6.A
	0011A	75B70F	
	0011D	75B600	
	00120	75B283	MOV 0B2,#083
	00123	43A902	ORL 0A9,#002
	00126	22	RET
	00127	439160	ORL 091,#060
	0012A	438920	ORL 089,#020
	0012D	5391F7	ANL 091,#0F7
	00130	43E308	ORL 0E3,#008
	00133	43D908	ORL 0D9,#008
	00136	43A502	ORL 0A5,#002
	00139	43A302	ORL 0A3,#002
	0013C	53A4FD	ANL 0A4,#0FD
	0013F	43A802	ORL 0A8,#002
	00142	22	RET

Figure 39. 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 40, the blue line represents a program counter breakpoint in the line, and the grey line represents the current program counter.



Sr	Addr	Opcode	Mnemonic
*	00107	1201FB	LCALL 001FB
*	0010A	120208	LCALL 00208
*	0010D	438860	ORL 088,#060
	00110	438020	ORL 080,#020
*	00113	43A880	ORL 0A8,#080
*	00116	80FE	SJMP 00116
*	00118	22	RET
*	00119	E50C	MOV A,00C
	0011B	700B	JNZ 00128
*	0011D	750C01	MOV 00C,#001
*	00120	750A34	MOV 00A,#034
	00123	750B02	MOV 00B,#002
	00126	8009	SJMP 00131
*	00128	750C00	MOV 00C,#000
*	0012B	750A31	MOV 00A,#031
	0012E	750B02	MOV 00B,#002
*	00131	22	RET

Figure 40. Using the PC Break Toggle Function

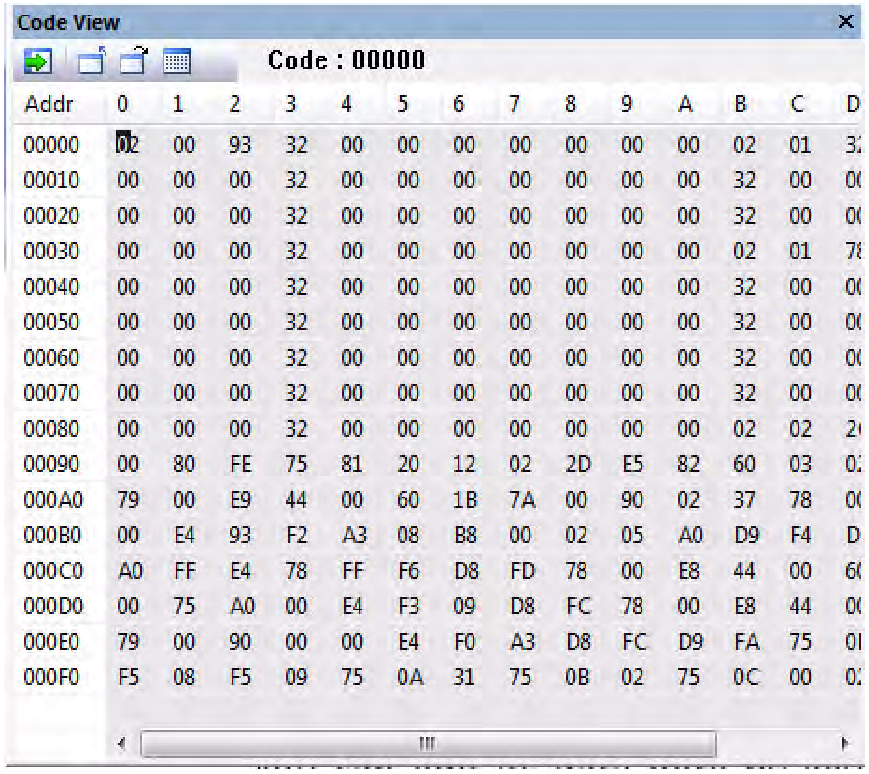
To set or clear a PC breakpoint, set your cursor on a selected line and double-click the mouse’s left button.

Note: Before you set a breakpoint on the Disasm View, a red asterisk (*) must be displayed adjacent to the **Addr** field that indicates the valid lines where you can set a breakpoint. Select **Load Hex** to download the hex file to the MCU. Select **Reset & Go** → **Break** to see the red asterisk (*) where you can set a breakpoint.

Code Dump View

Code dump views 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 41, for example, the page number is `00`.

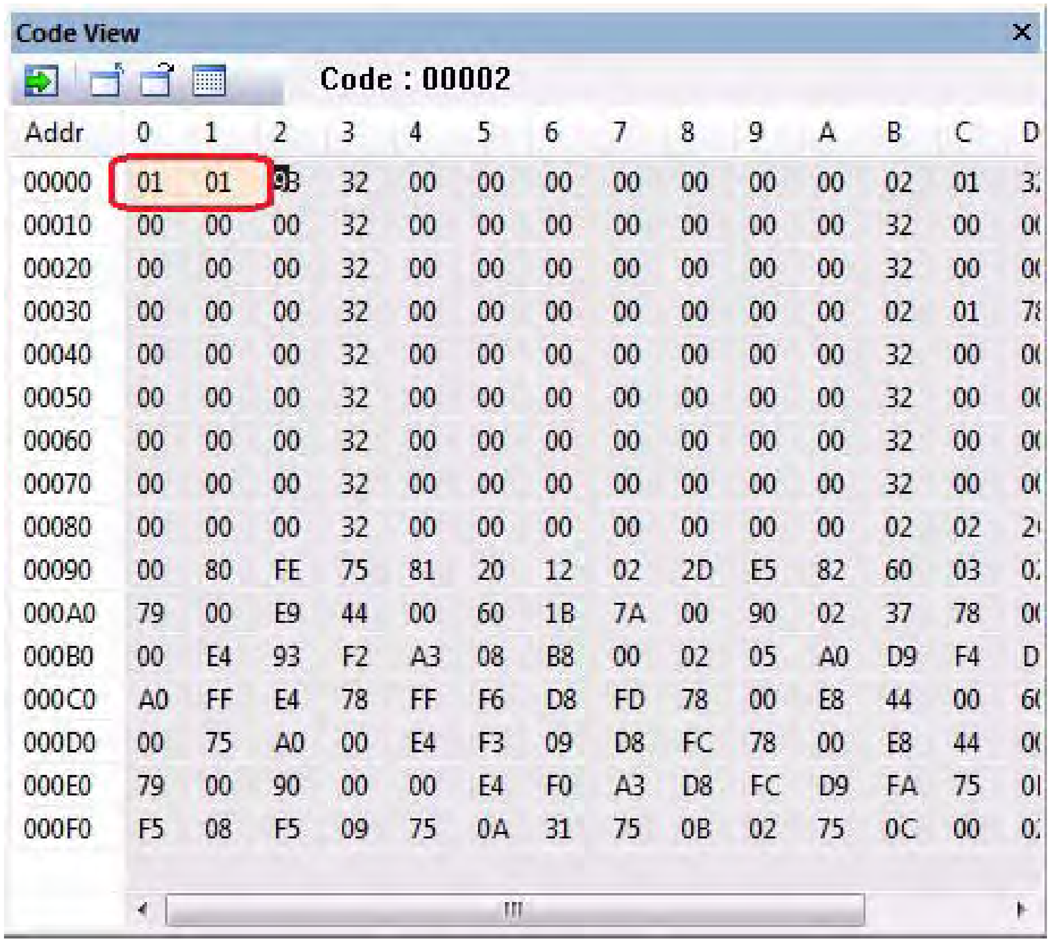

Code View

```
Code : 00000

Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D
00000  02 00 93 32 00 00 00 00 00 00 00 02 01 3
00010  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00020  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00030  00 00 00 32 00 00 00 00 00 00 00 02 01 7
00040  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00050  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00060  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00070  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00080  00 00 00 32 00 00 00 00 00 00 00 02 02 2
00090  00 80 FE 75 81 20 12 02 2D E5 82 60 03 0
000A0  79 00 E9 44 00 60 1B 7A 00 90 02 37 78 0
000B0  00 E4 93 F2 A3 08 B8 00 02 05 A0 D9 F4 D
000C0  A0 FF E4 78 FF F6 D8 FD 78 00 E8 44 00 6
000D0  00 75 A0 00 E4 F3 09 D8 FC 78 00 E8 44 0
000E0  79 00 90 00 00 E4 F0 A3 D8 FC D9 FA 75 0
000F0  F5 08 F5 09 75 0A 31 75 0B 02 75 0C 00 0
```

Figure 41. 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 background will change from grey to peach to indicate that the change was made, as highlighted in Figure 42.



```
Code View
Code : 00002
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D
00000  01 01 9B 32 00 00 00 00 00 00 00 02 01 3
00010  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00020  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00030  00 00 00 32 00 00 00 00 00 00 00 02 01 7
00040  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00050  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00060  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00070  00 00 00 32 00 00 00 00 00 00 00 32 00 0
00080  00 00 00 32 00 00 00 00 00 00 00 02 02 2
00090  00 80 FE 75 81 20 12 02 2D E5 82 60 03 0
000A0  79 00 E9 44 00 60 1B 7A 00 90 02 37 78 0
000B0  00 E4 93 F2 A3 08 B8 00 02 05 A0 D9 F4 D
000C0  A0 FF E4 78 FF F6 D8 FD 78 00 E8 44 00 6
000D0  00 75 A0 00 E4 F3 09 D8 FC 78 00 E8 44 0
000E0  79 00 90 00 00 E4 F0 A3 D8 FC D9 FA 75 0
000F0  F5 08 F5 09 75 0A 31 75 0B 02 75 0C 00 0
```

Figure 42. 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.

Move

Click the **Move** 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 43.

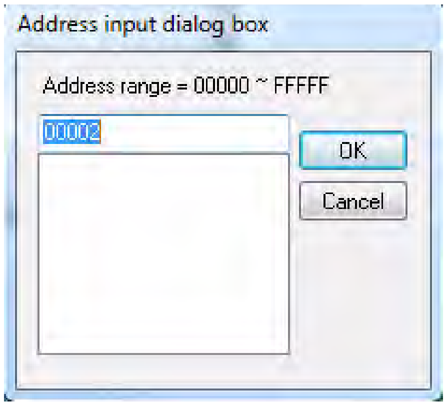


Figure 43. The Code Dump Function's Goto/Input Dialog

Pattern Load

Click the **Pattern 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 44.

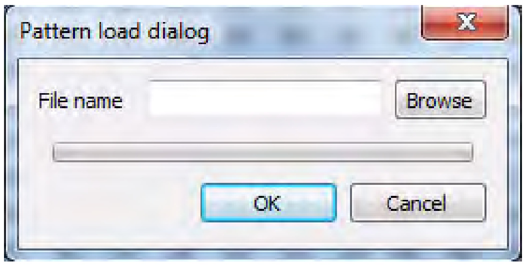
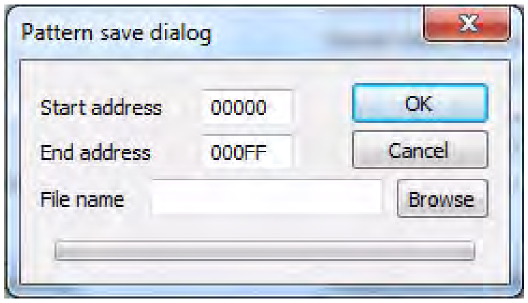


Figure 44. The Code Dump Function's Pattern Load Dialog

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.

Pattern Save

Click the **Pattern 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 45.

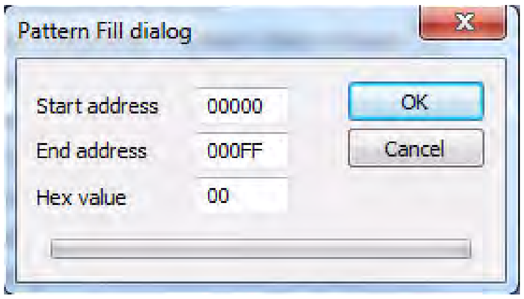


Start address 00000
End address 000FF
File name

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

Pattern Fill

Click the **Pattern 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 46.



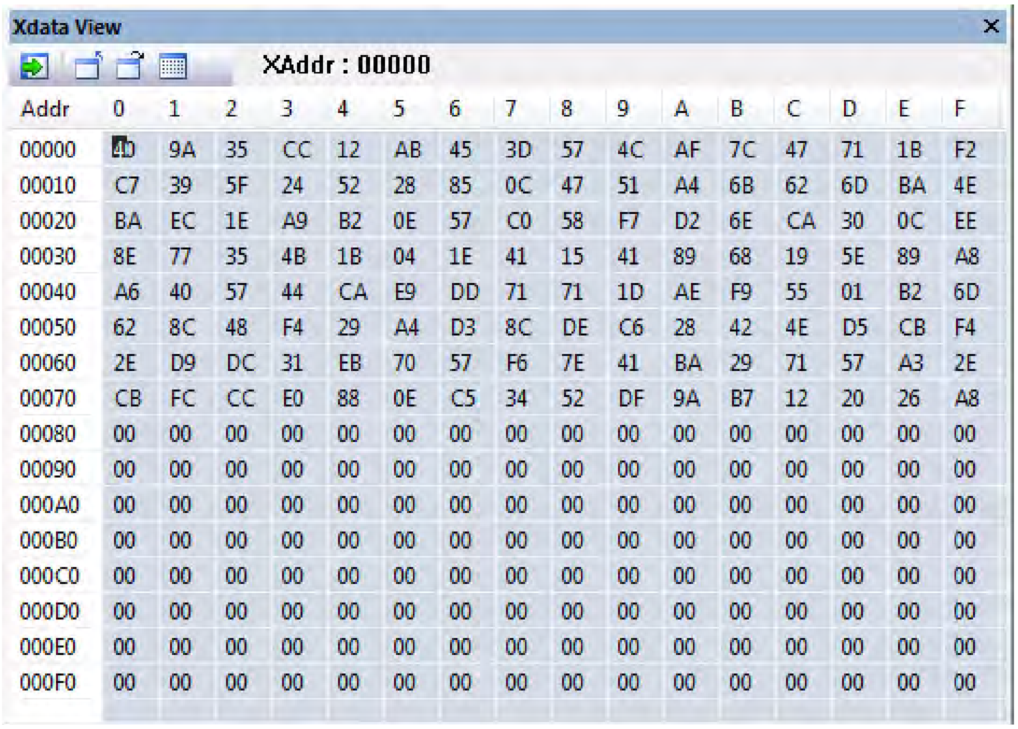
Start address 00000
End address 000FF
Hex value 00

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

Xdata Dump View

The Xdata Dump view 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 view 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 47, for example, the page number is `00`.

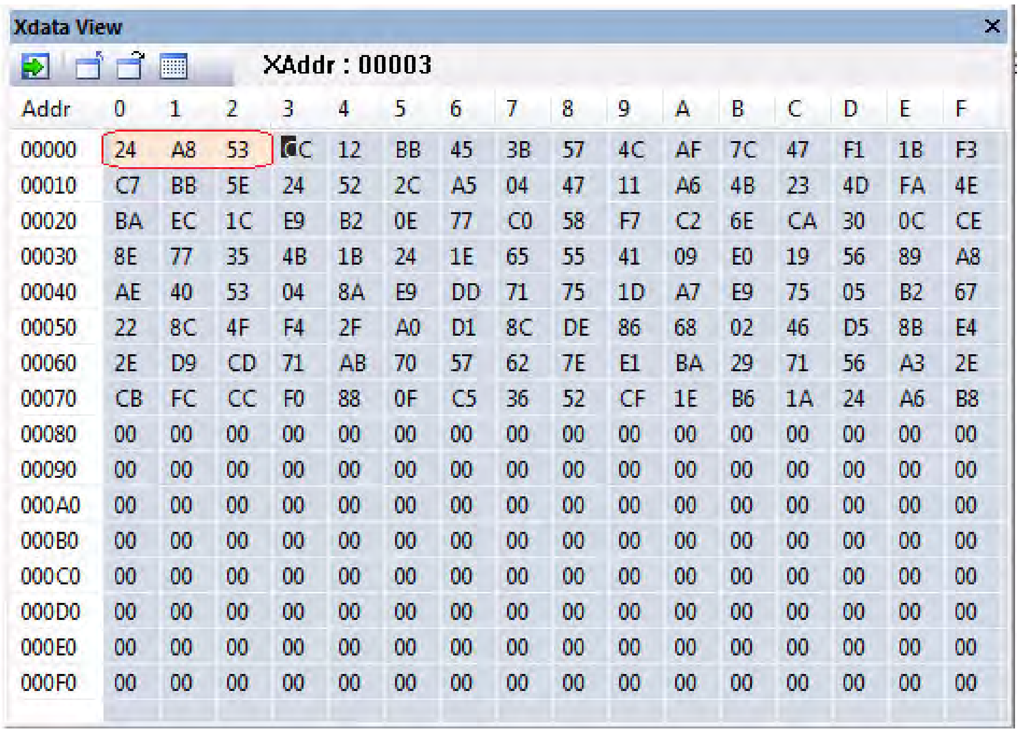


```
XAddr : 00000

Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  4D 9A 35 CC 12 AB 45 3D 57 4C AF 7C 47 71 1B F2
00010  C7 39 5F 24 52 28 85 0C 47 51 A4 6B 62 6D BA 4E
00020  BA EC 1E A9 B2 0E 57 C0 58 F7 D2 6E CA 30 0C EE
00030  8E 77 35 4B 1B 04 1E 41 15 41 89 68 19 5E 89 A8
00040  A6 40 57 44 CA E9 DD 71 71 1D AE F9 55 01 B2 6D
00050  62 8C 48 F4 29 A4 D3 8C DE C6 28 42 4E D5 CB F4
00060  2E D9 DC 31 EB 70 57 F6 7E 41 BA 29 71 57 A3 2E
00070  CB FC CC E0 88 0E C5 34 52 DF 9A B7 12 20 26 A8
00080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 47. Using the Xdata Dump Function, #1 of 2

Edit

Users can change data values in the XData Dump View 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 background will change from grey to peach to indicate that the change was made, as highlighted in Figure 48.



Xdata View

XAddr : 00003

Addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000	24	A8	53	CC	12	BB	45	3B	57	4C	AF	7C	47	F1	1B	F3
00010	C7	BB	5E	24	52	2C	A5	04	47	11	A6	4B	23	4D	FA	4E
00020	BA	EC	1C	E9	B2	0E	77	C0	58	F7	C2	6E	CA	30	0C	CE
00030	8E	77	35	4B	1B	24	1E	65	55	41	09	E0	19	56	89	A8
00040	AE	40	53	04	8A	E9	DD	71	75	1D	A7	E9	75	05	B2	67
00050	22	8C	4F	F4	2F	A0	D1	8C	DE	86	68	02	46	D5	8B	E4
00060	2E	D9	CD	71	AB	70	57	62	7E	E1	BA	29	71	56	A3	2E
00070	CB	FC	CC	F0	88	0F	C5	36	52	CF	1E	B6	1A	24	A6	B8
00080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figure 48. 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.

Move

Click the **Move** 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 49.

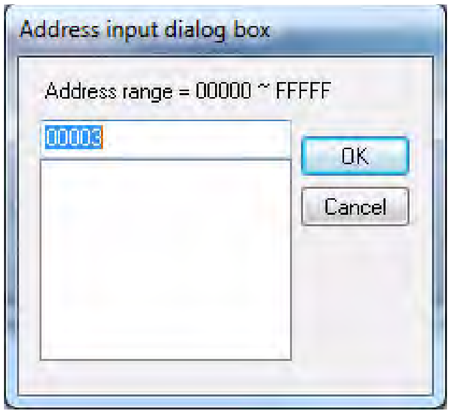


Figure 49. The Xdata Dump Function's Move/Input Dialog

Pattern Load

Click the **Pattern 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 50.

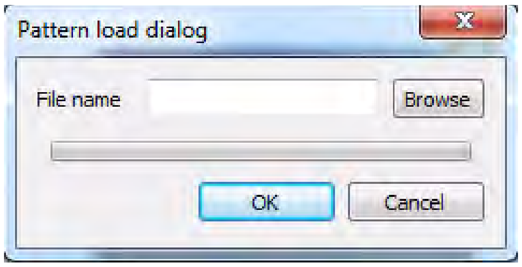


Figure 50. The Xdata Dump Function's Pattern Load Dialog

Pattern Save

Click the **Pattern Save** button to display the Pattern Save dialog, in which you can save the Xdata area as a pattern file; see Figure 51.

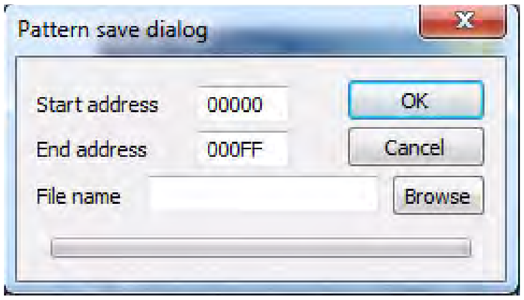


Figure 51. The Xdata Dump Function’s Pattern Save Dialog

Pattern Fill

Click the **Pattern 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 52.

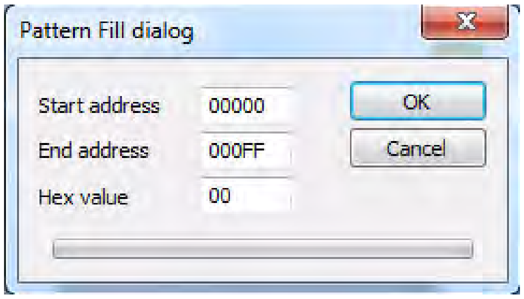


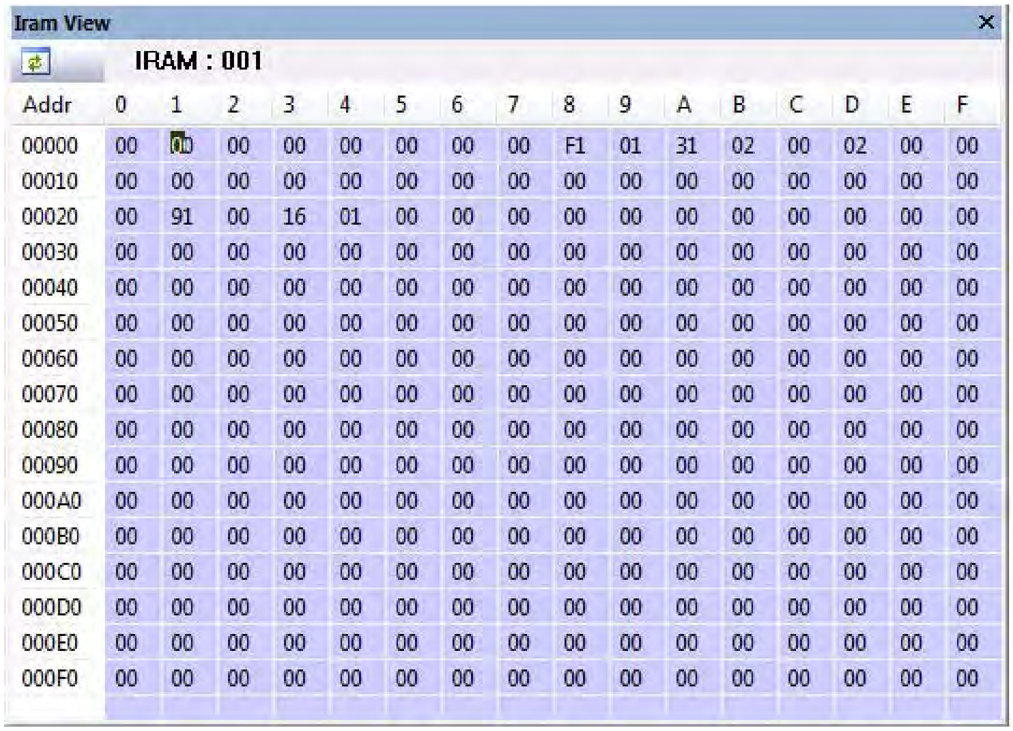
Figure 52. The Xdata Dump Function’s Pattern Fill Dialog

Iram Dump View

The Iram Dump view 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.

Figure 53 also shows Iram addresses in the range `00h–7Fh`, which represent the direct area; the characters representing these addresses are colored black.



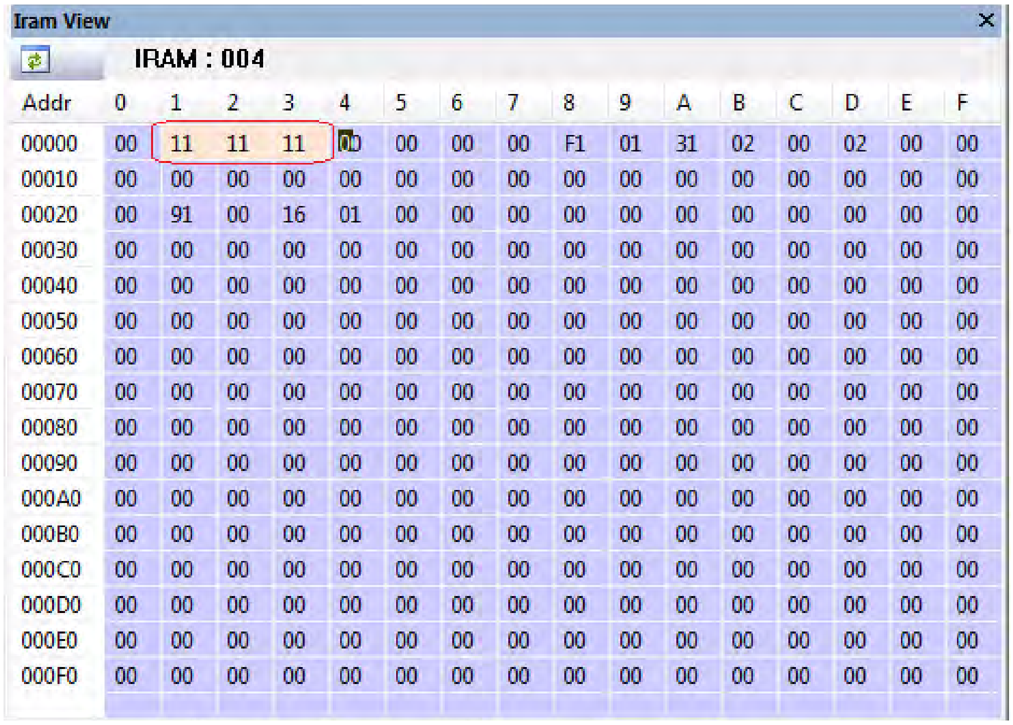
```
Iram View                                                          ×
      IRAM : 001
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  00 00 00 00 00 00 00 00 F1 01 31 02 00 02 00 00
00010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020  00 91 00 16 01 00 00 00 00 00 00 00 00 00 00 00
00030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 53. 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 view 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 background will change from grey to peach to indicate that the change was made, as highlighted in Figure 54.



```
IRAM : 004
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00000  00 11 11 11 00 00 00 00 F1 01 31 02 00 02 00 00
00010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00020  00 91 00 16 01 00 00 00 00 00 00 00 00 00 00 00
00030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

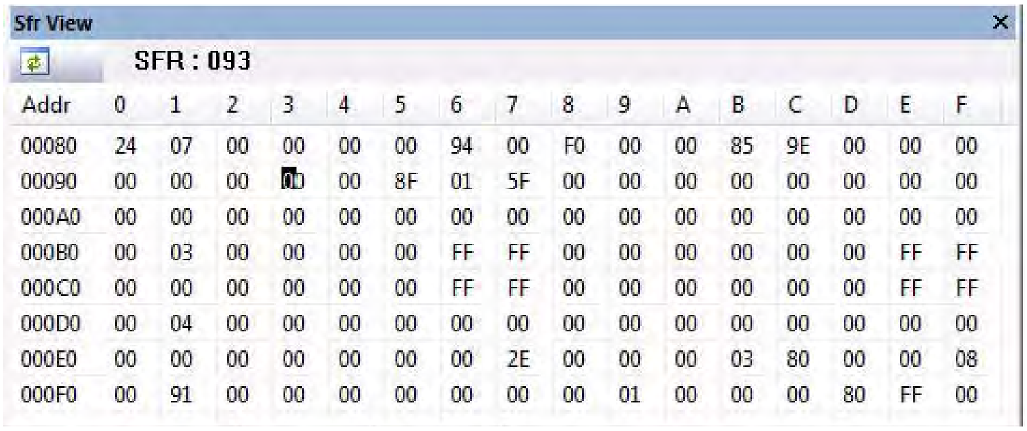
Figure 54. Using the Iram Dump Function, #2 of 2

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.

Sfr Dump View

The Special Function Register (Sfr) Dump view 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 view displays the address of the current cursor position and the checksum of the current page.

Figure 55 shows Sfr addresses in the range `80h–FFh`, which represent the direct area of Iram.



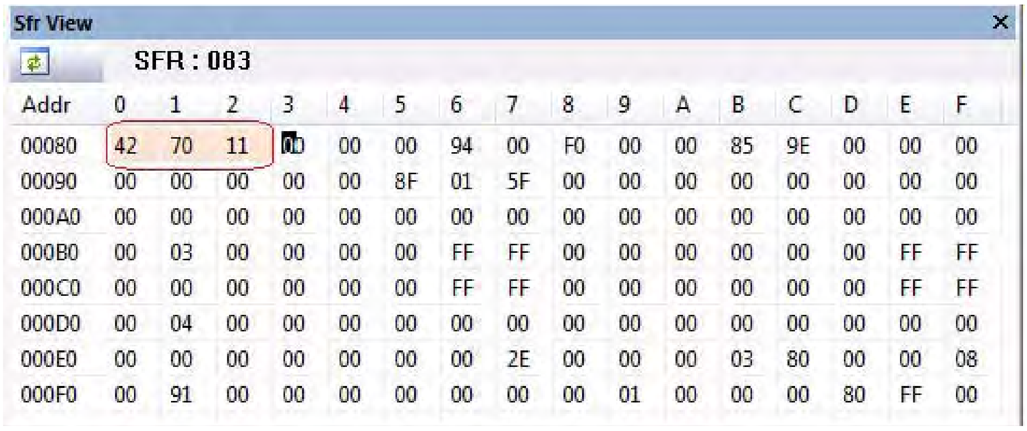
```
Sfr View
SFR : 093
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00080  24 07 00 00 00 00 94 00 F0 00 00 85 9E 00 00 00
00090  00 00 00 00 00 8F 01 5F 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 03 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
000C0  00 00 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
000D0  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 2E 00 00 00 03 80 00 00 08
000F0  00 91 00 00 00 00 00 00 00 01 00 00 00 80 FF 00
```

Figure 55. 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 view 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 background will change from white to peach to indicate that the change was made, as highlighted in Figure 56.



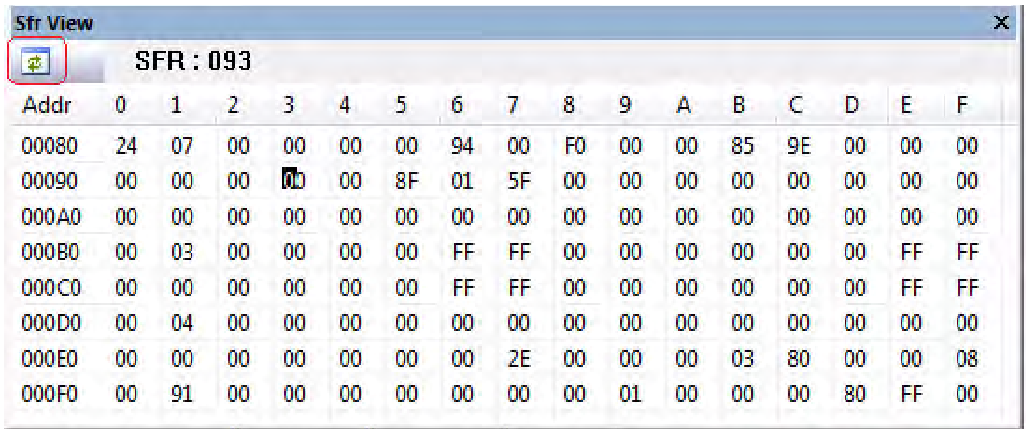
```
Sfr View
SFR : 083
Addr   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00080  42 70 11 00 00 00 94 00 F0 00 00 85 9E 00 00 00
00090  00 00 00 00 00 8F 01 5F 00 00 00 00 00 00 00 00
000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000B0  00 03 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
000C0  00 00 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
000D0  00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000E0  00 00 00 00 00 00 00 2E 00 00 00 03 80 00 00 08
000F0  00 91 00 00 00 00 00 00 00 01 00 00 00 80 FF 00
```

Figure 56. 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 peach color of your changed character pair background has changed back to white.

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 57) redisplays all current data.



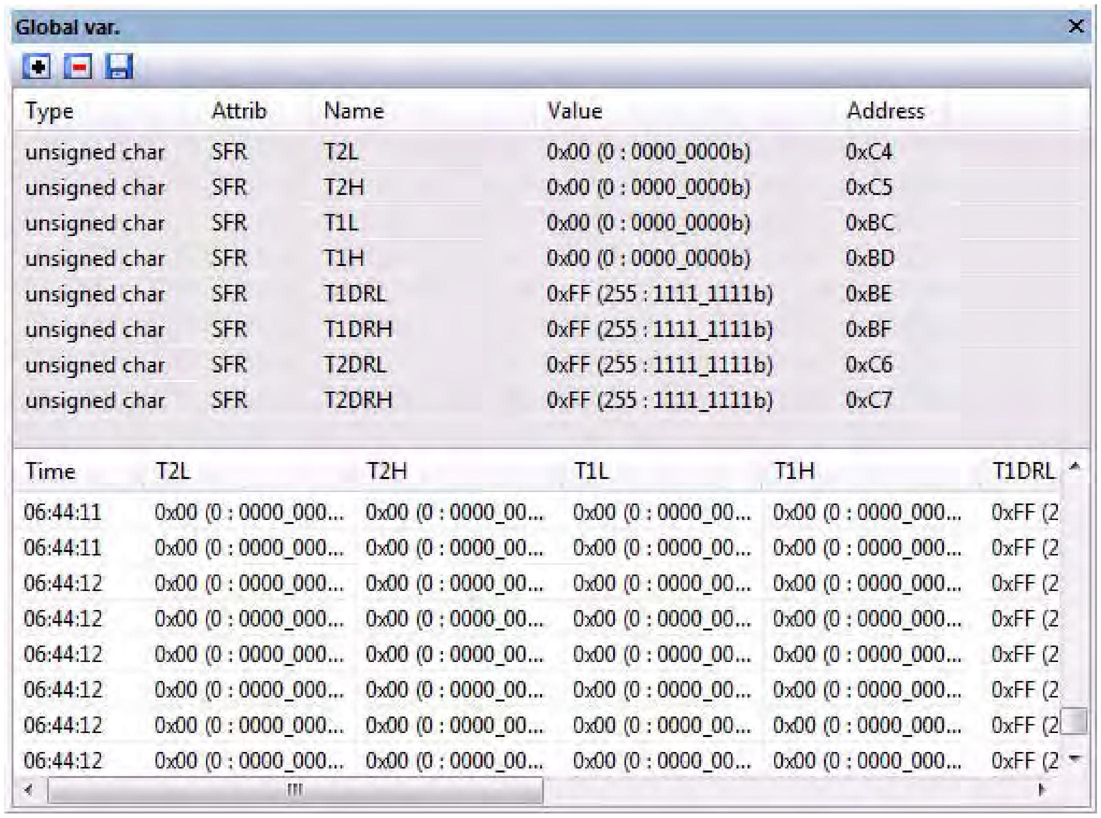
Addr	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00080	24	07	00	00	00	00	94	00	F0	00	00	85	9E	00	00	00
00090	00	00	00	00	00	8F	01	5F	00	00	00	00	00	00	00	00
000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000B0	00	03	00	00	00	00	FF	FF	00	00	00	00	00	00	FF	FF
000C0	00	00	00	00	00	00	FF	FF	00	00	00	00	00	00	FF	FF
000D0	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000E0	00	00	00	00	00	00	00	2E	00	00	00	03	80	00	00	08
000F0	00	91	00	00	00	00	00	00	00	01	00	00	00	80	FF	00

Figure 57. Using the SFRDump Function, #3 of 3

Global Variable

The Global var. 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 Global variable alleviates this problem by employing a map/symbol file; see Figure 58.



Type	Attrib	Name	Value	Address
unsigned char	SFR	T2L	0x00 (0 : 0000_0000b)	0xC4
unsigned char	SFR	T2H	0x00 (0 : 0000_0000b)	0xC5
unsigned char	SFR	T1L	0x00 (0 : 0000_0000b)	0xBC
unsigned char	SFR	T1H	0x00 (0 : 0000_0000b)	0xBD
unsigned char	SFR	T1DRL	0xFF (255 : 1111_1111b)	0xBE
unsigned char	SFR	T1DRH	0xFF (255 : 1111_1111b)	0xBF
unsigned char	SFR	T2DRL	0xFF (255 : 1111_1111b)	0xC6
unsigned char	SFR	T2DRH	0xFF (255 : 1111_1111b)	0xC7

Time	T2L	T2H	T1L	T1H	T1DRL
06:44:11	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:11	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2
06:44:12	0x00 (0 : 0000_000...	0x00 (0 : 0000_00...	0x00 (0 : 0000_00...	0x00 (0 : 0000_000...	0xFF (2

Figure 58. The Global Function's Global Variables Dialog

Add Variable

Clicking the **Add Variable** button displays the Global Variable Add/Remove dialog, in which you can add a global variable to the Global display list, shown in Figure 59.

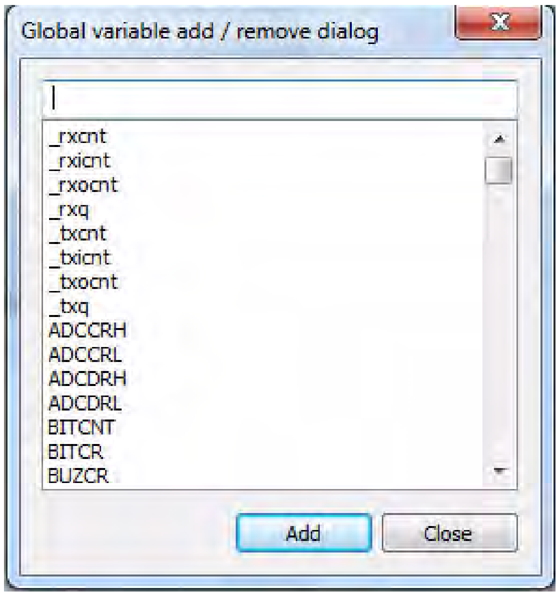


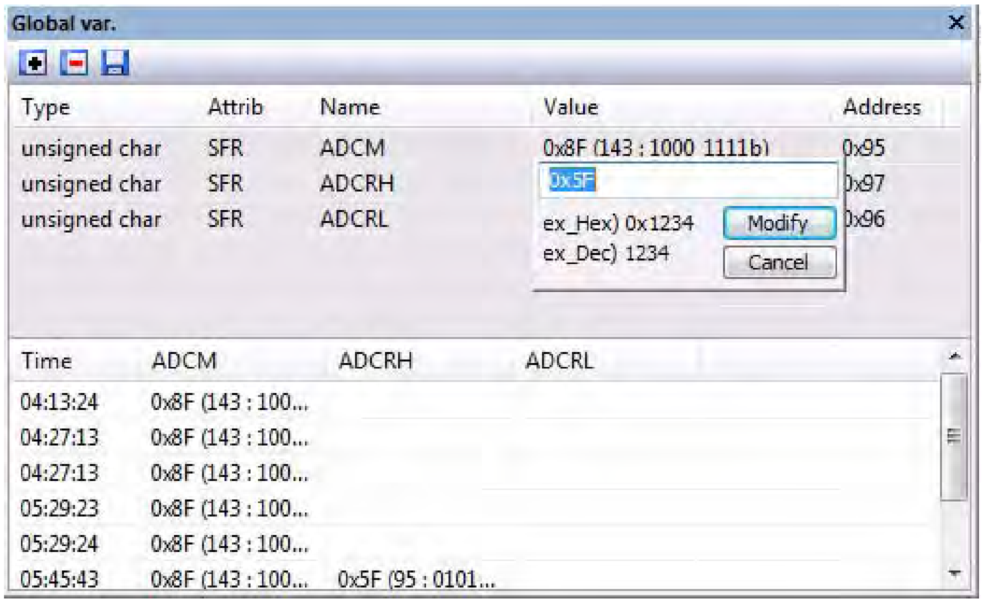
Figure 59. Adding A Global Variable

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 Global variable at any time, except during emulation. This editing method is quite simple; just place the cursor where you wish to make an edit, and double-click the left button on your mouse to display a pop-up dialog in which you can change the data and click the **Modify** pop-up button to incorporate the change, as shown in Figure 60.



Global var.

Type	Attrib	Name	Value	Address
unsigned char	SFR	ADCM	0x8F (143 : 1000 1111b)	0x95
unsigned char	SFR	ADCRH	0x5F	0x97
unsigned char	SFR	ADCRL		0x96

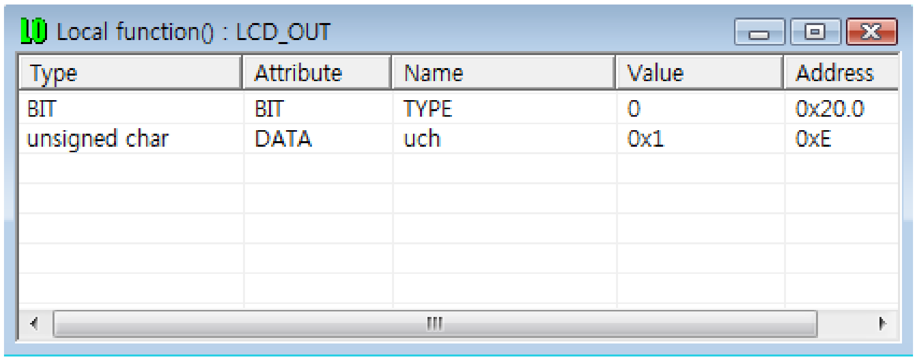
Time	ADCM	ADCRH	ADCRL
04:13:24	0x8F (143 : 100...		
04:27:13	0x8F (143 : 100...		
04:27:13	0x8F (143 : 100...		
05:29:23	0x8F (143 : 100...		
05:29:24	0x8F (143 : 100...		
05:45:43	0x8F (143 : 100...	0x5F (95 : 0101...	

Figure 60. Editing A Global Symbol

Local Variable

The Local variable 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 Local variable, shown in Figure 61, alleviates this problem by employing a map/symbol file, as described in the previous section.



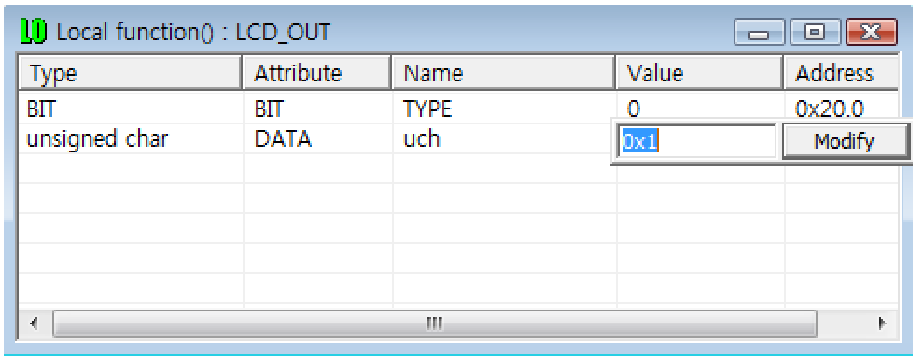
Local function() : LCD_OUT

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

Figure 61. The Local Function Dialog

Edit

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



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

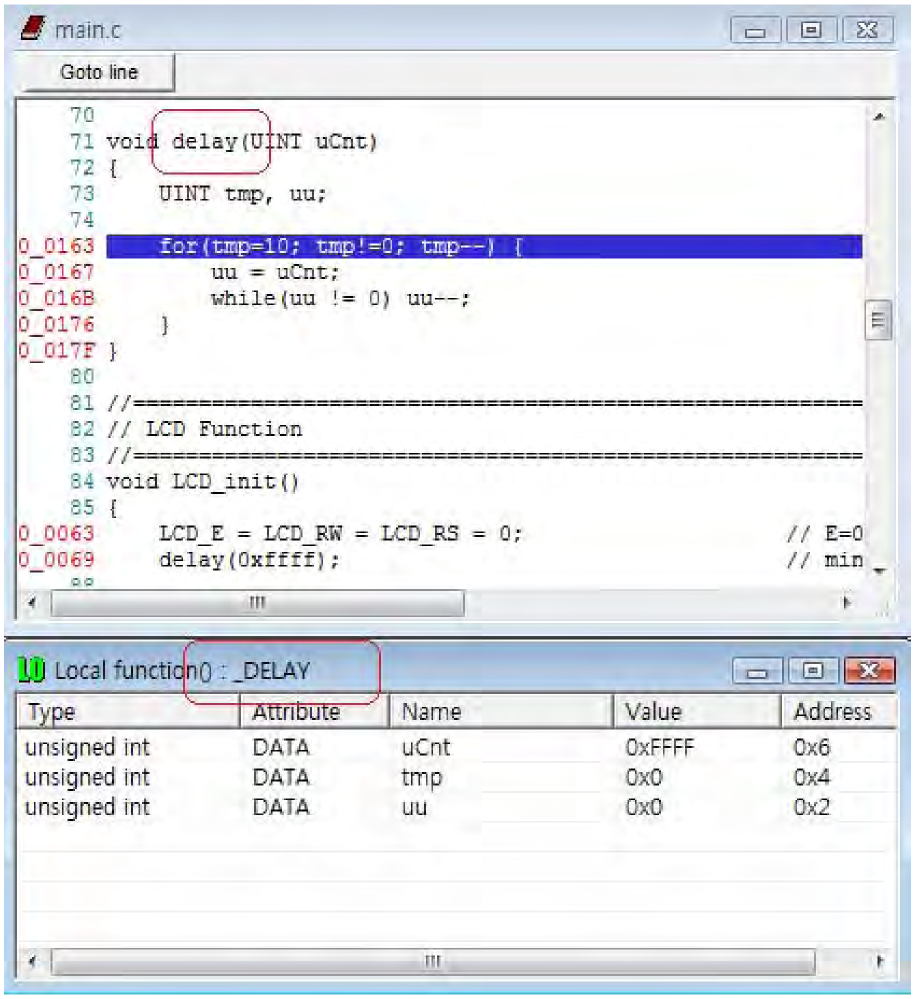
Figure 62. Editing A Local Symbol

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 63 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).



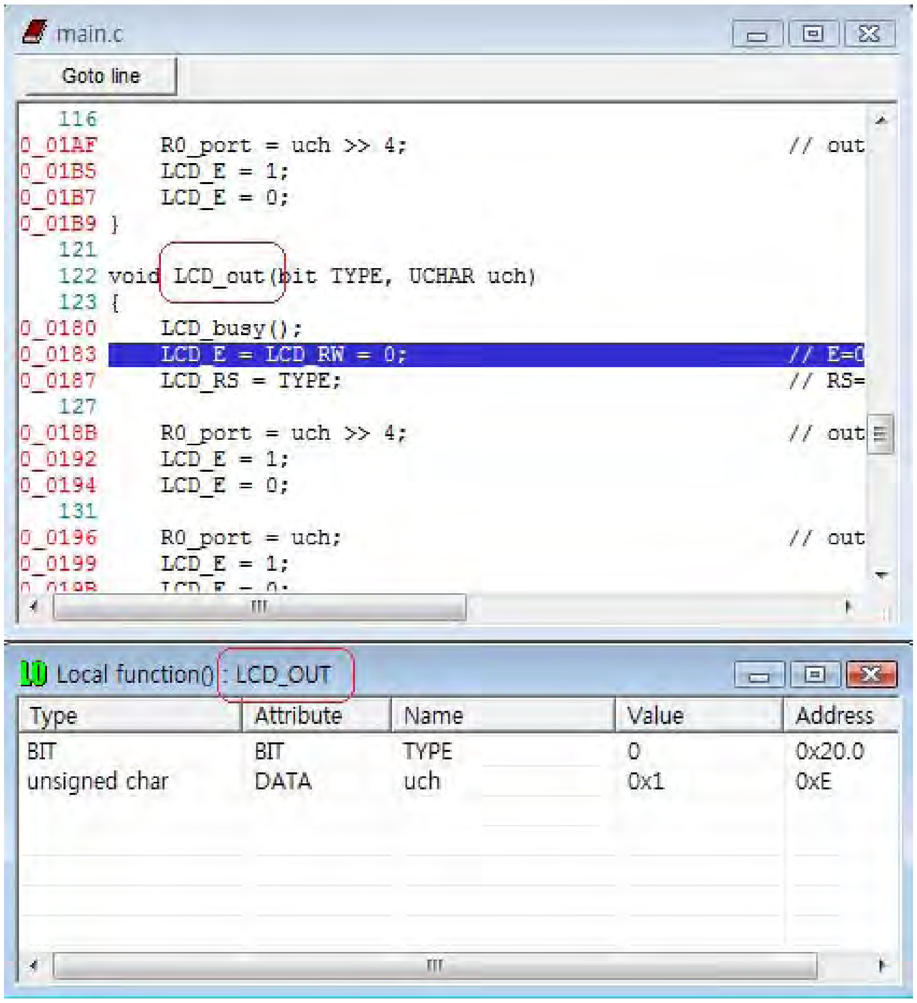
```
0_0163      for(tmp=10; tmp!=0; tmp--) {
0_0167          uu = uCnt;
0_016B          while(uu != 0) uu--;
0_0176      }
0_017F }
```

```
void delay(UINT uCnt)
{
    UINT tmp, uu;

    for(tmp=10; tmp!=0; tmp--) {
        uu = uCnt;
        while(uu != 0) uu--;
    }
}

//============================================================
// LCD Function
//============================================================
void LCD_init()
{
    LCD_E = LCD_RW = LCD_RS = 0;                    // E=0
    delay(0xffff);                                  // min
```

Type	Attribute	Name	Value	Address
unsigned int	DATA	uCnt	0xFFFF	0x6
unsigned int	DATA	tmp	0x0	0x4
unsigned int	DATA	uu	0x0	0x2

Figure 63. Example 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 64.



main.c

Goto line

```

0_01AF     R0_port = uch >> 4;                        // out
0_01B5     LCD_E = 1;
0_01B7     LCD_E = 0;
0_01B9 }

       void LCD_out(bit TYPE, UCHAR uch)
       {
0_0180     LCD_busy();
0_0183     LCD_E = LCD_RW = 0;                        // E=
0_0187     LCD_RS = TYPE;                             // RS=

0_018B     R0_port = uch >> 4;                        // out
0_0192     LCD_E = 1;
0_0194     LCD_E = 0;

0_0196     R0_port = uch;                             // out
0_0199     LCD_E = 1;
0_019B     LCD_E = 0;
```

Local function() : LCD_OUT

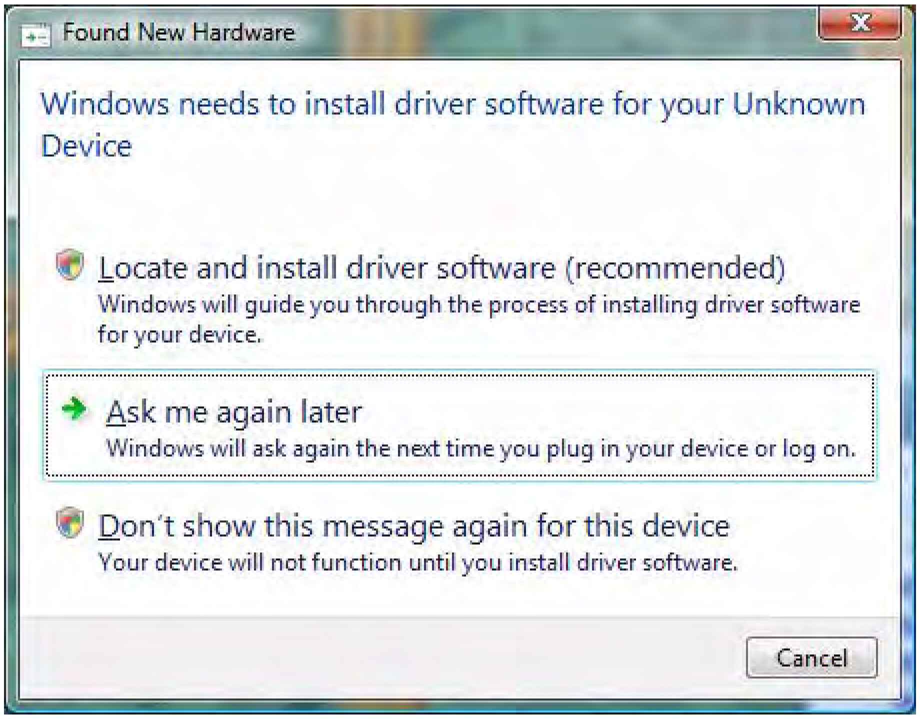
Type	Attribute	Name	Value	Address
BIT	BIT	TYPE	0	0x20.0
unsigned char	DATA	uch	0x1	0xE

Figure 64. Example Local Function, #2 of 2

Appendix A. OCD Driver Installation on Windows Vista Systems

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

1. Connect the OCDII 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 OCDII device.
2. After the PC detects the new hardware, it will display the Found New Hardware Wizard dialog box, shown in Figure 65. Click **Locate and install driver software (recommended)**.


Found New Hardware

Windows needs to install driver software for your Unknown Device

Locate and install driver software (recommended)
Windows will guide you through the process of installing driver software for your device.

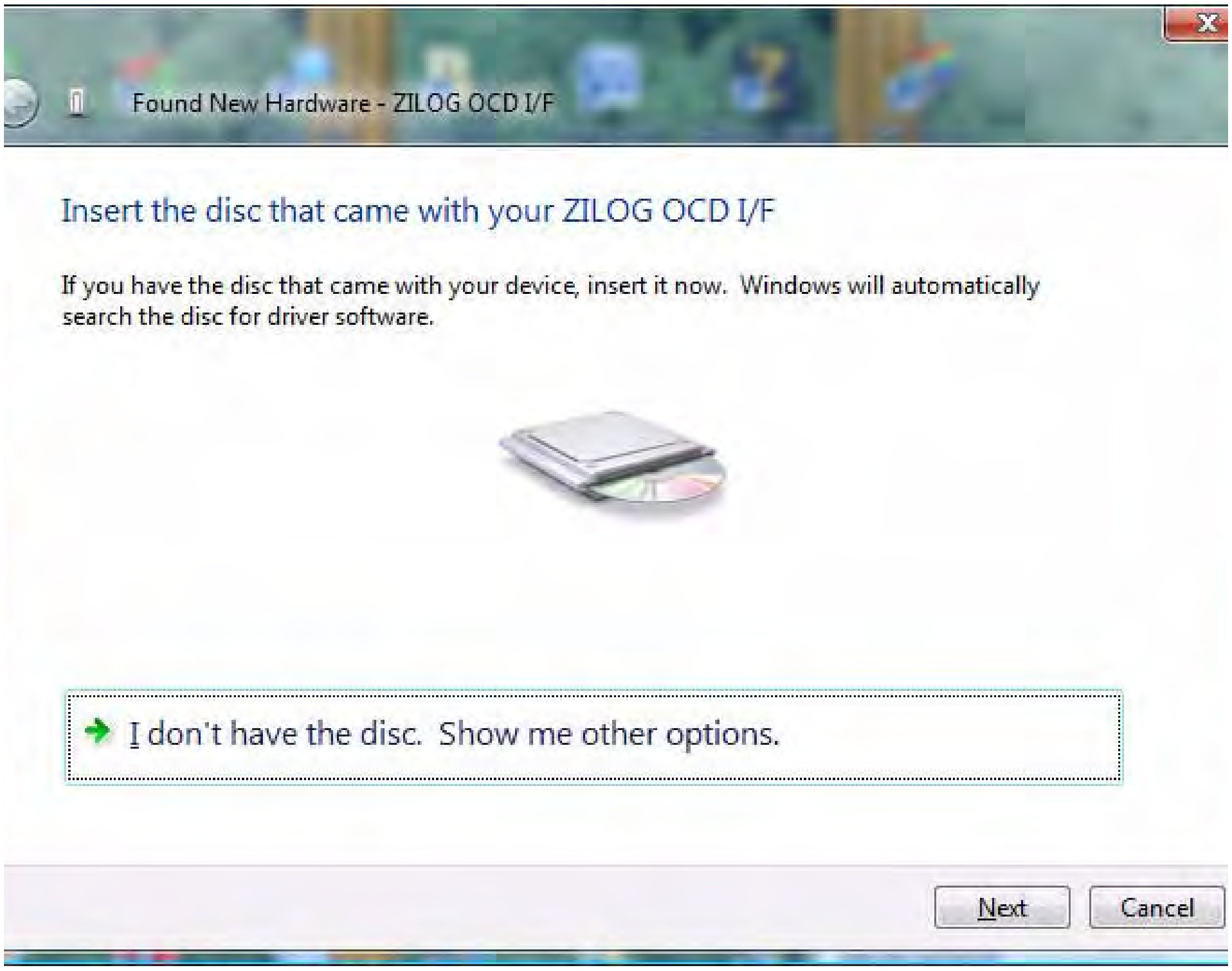
Ask me again later
Windows will ask again the next time you plug in your device or log on.

Don't show this message again for this device
Your device will not function until you install driver software.

Figure 65. 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 66, 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 67.

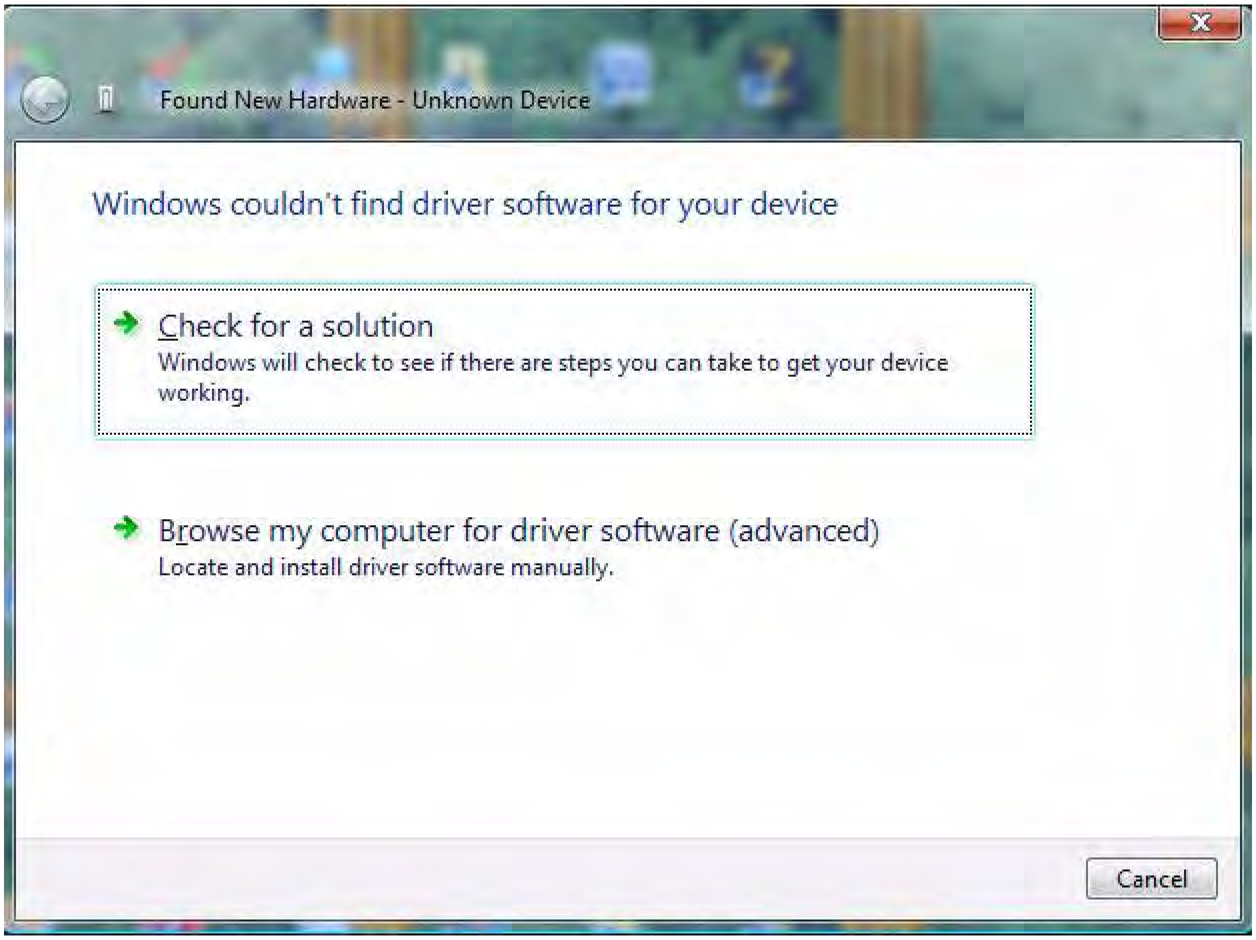

Found New Hardware - ZILOG OCD I/F

Insert the disc that came with your ZILOG OCD I/F

If you have the disc that came with your device, insert it now. Windows will automatically search the disc for driver software.

I don't have the disc. Show me other options.

Figure 66. Install Device Driver Dialog, Windows Vista


Found New Hardware - Unknown Device

Windows couldn't find driver software for your device

Check for a solution
Windows will check to see if there are steps you can take to get your device working.

Browse my computer for driver software (advanced)
Locate and install driver software manually.

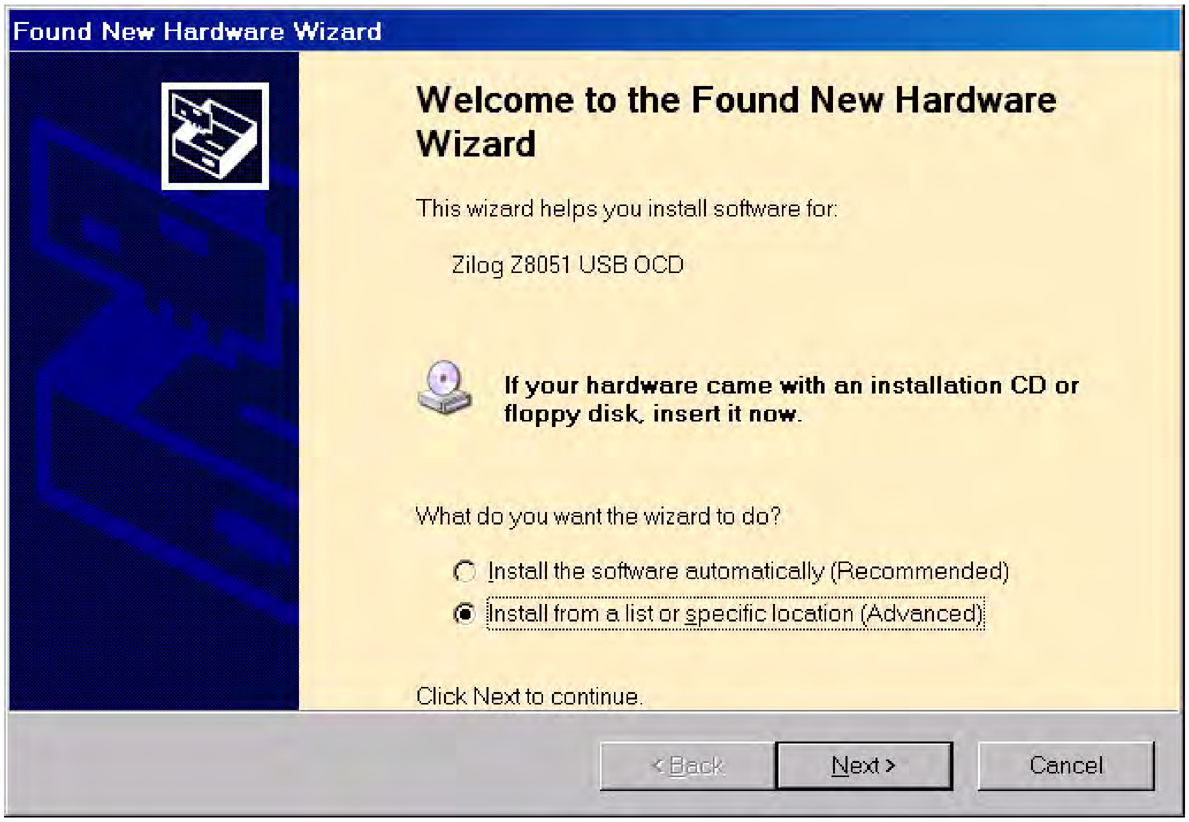
Figure 67. Couldn't Find Driver Dialog, Windows Vista

5. Return to page 4 and follow Steps 3 through 6.

Appendix B. OCD Driver Installation on Windows XP Systems

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

1. Connect the OCD hardware to the USB port of your PC by connecting the A-Male end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port, and connect the Mini-B end to the OCD device.
2. After the PC detects the new hardware, it will display the **Found New Hardware Wizard** dialog box, shown in Figure 68. Select **Install from a list or specific location (Advanced)**; then click **Next**.


Found New Hardware Wizard

Welcome to the Found New Hardware Wizard

This wizard helps you install software for:

Zilog Z8051 USB OCD

If your hardware came with an installation CD or floppy disk, insert it now.

What do you want the wizard to do?

○ Install the software automatically (Recommended)
● Install from a list or specific location (Advanced)

Click Next to continue.

Figure 68. The Found New Hardware Wizard Welcome Screen

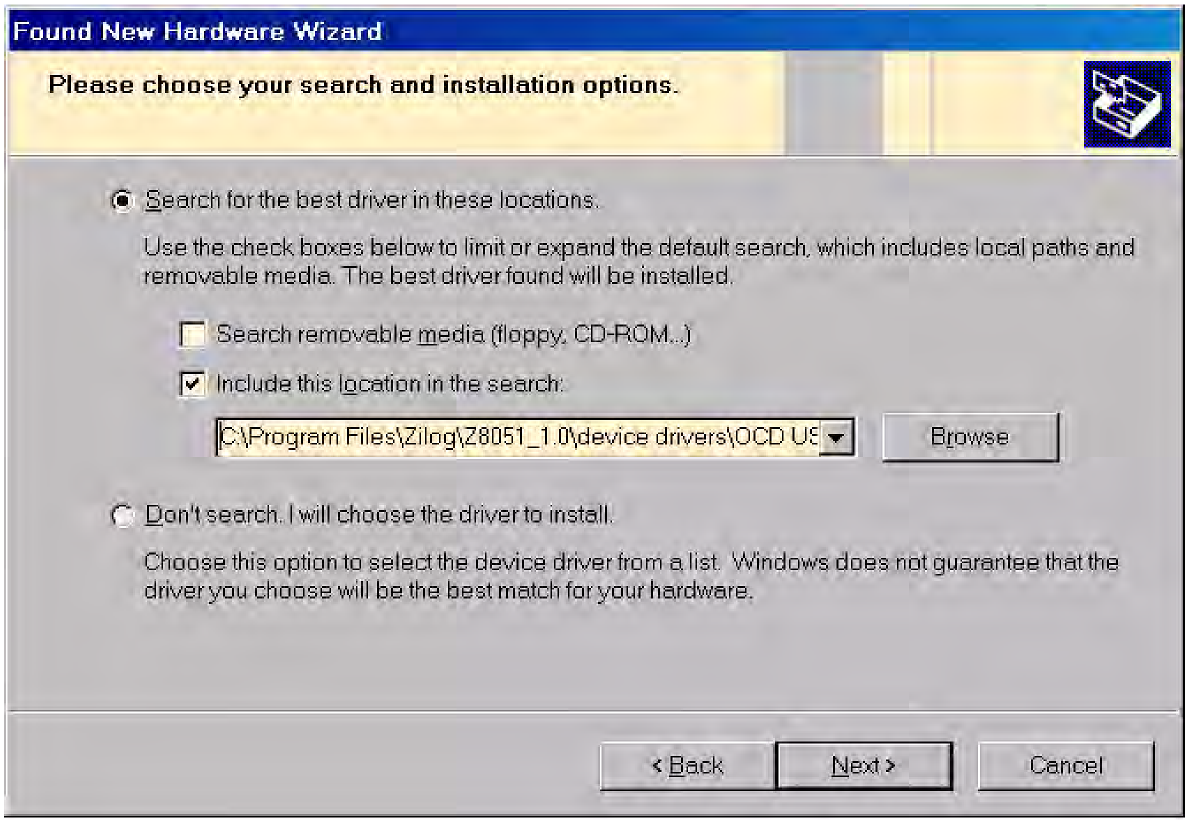
3. The next dialog box, shown in Figure 69, prompts you to enter a path or navigate to the directory in which the `.inf` file was installed. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse** button to navigate to one of the following paths and click the **Next** button, leaving all other selections at their default settings.
   - On OCDI dongle, use the following driver path for either 32-bit or 64-bit machines:

     <Z8051 Installation>\Z8051_<*version_number*>\device drivers\OCD USB\OCDI

   - On OCDII dongle, use the following driver path for either 32-bit or 64-bit machines:

     <Z8051 Installation>\Z8051_<*version_number*>\device drivers\OCD USB\OCDII

Note: On some installations, the Found New Hardware screen shown in Figure 68 may also display the text string, `Zilog Z8051 USB OCD - No Firmware`. This occurrence is normal and can be disregarded.



Please choose your search and installation options.

[● Search for the best driver in these locations.]

Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.

☐ Search removable media (floppy, CD-ROM...)

☑ Include this location in the search:

C:\Program Files\Zilog\Z8051_1.0\device drivers\OCD US

○ Don't search. I will choose the driver to install.

Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.

Figure 69. 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**.

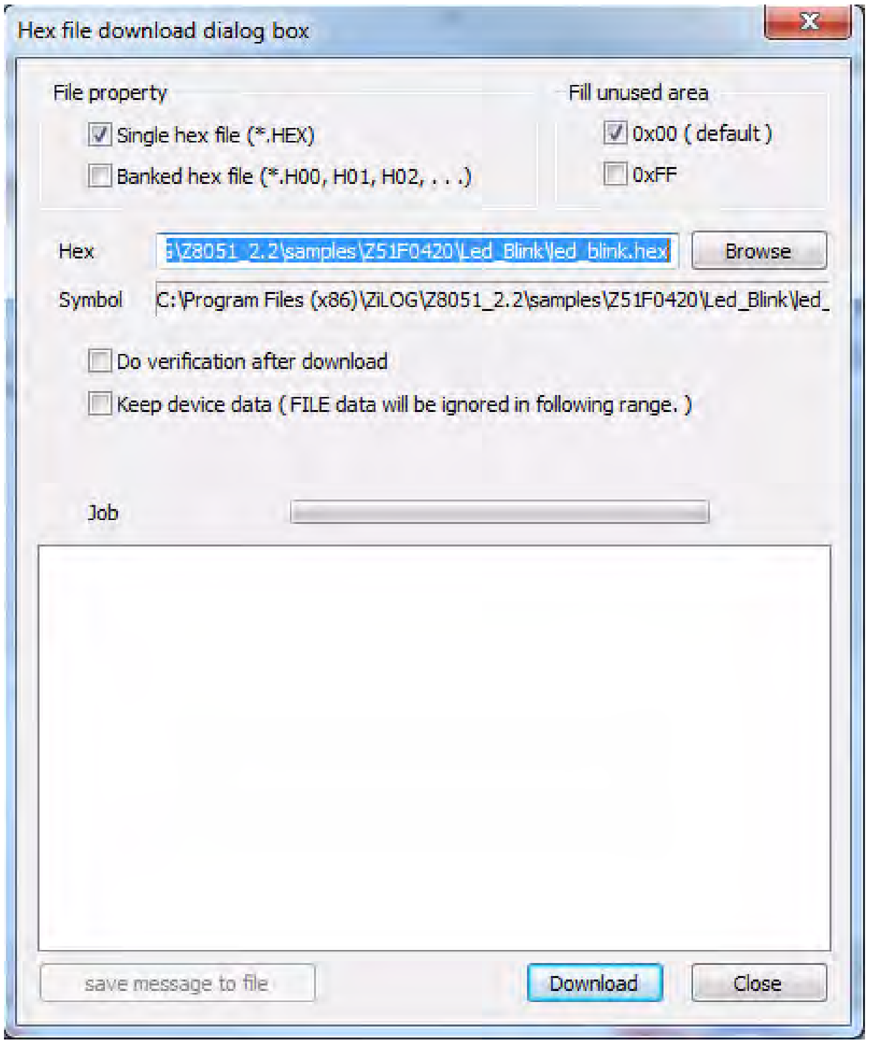
Appendix C. Load Hex Procedure

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

1. Run the Z8051 OCDII software. From the **Start** menu, navigate to **All Programs** → **Zilog Z8051 SW and Docs 2.2** → **Zilog Z8051 OCD I and II V1.0**.

Note: You can download the OCDII Software Z8051 Software and Documentation v2.2 for free from the Zilog Store.

2. From the Emulation menu, select **Load Hex**. The Hex File Download dialog box appears, as shown in Figure 70.


Hex file download dialog box

File property

[✓] Single hex file (*.HEX)

[ ] Banked hex file (*.H00, H01, H02, . . .)

Fill unused area

[✓] 0x00 (default)

[ ] 0xFF

Hex: G\Z8051_2.2\samples\Z51F0420\Led_Blink\led_blink.hex

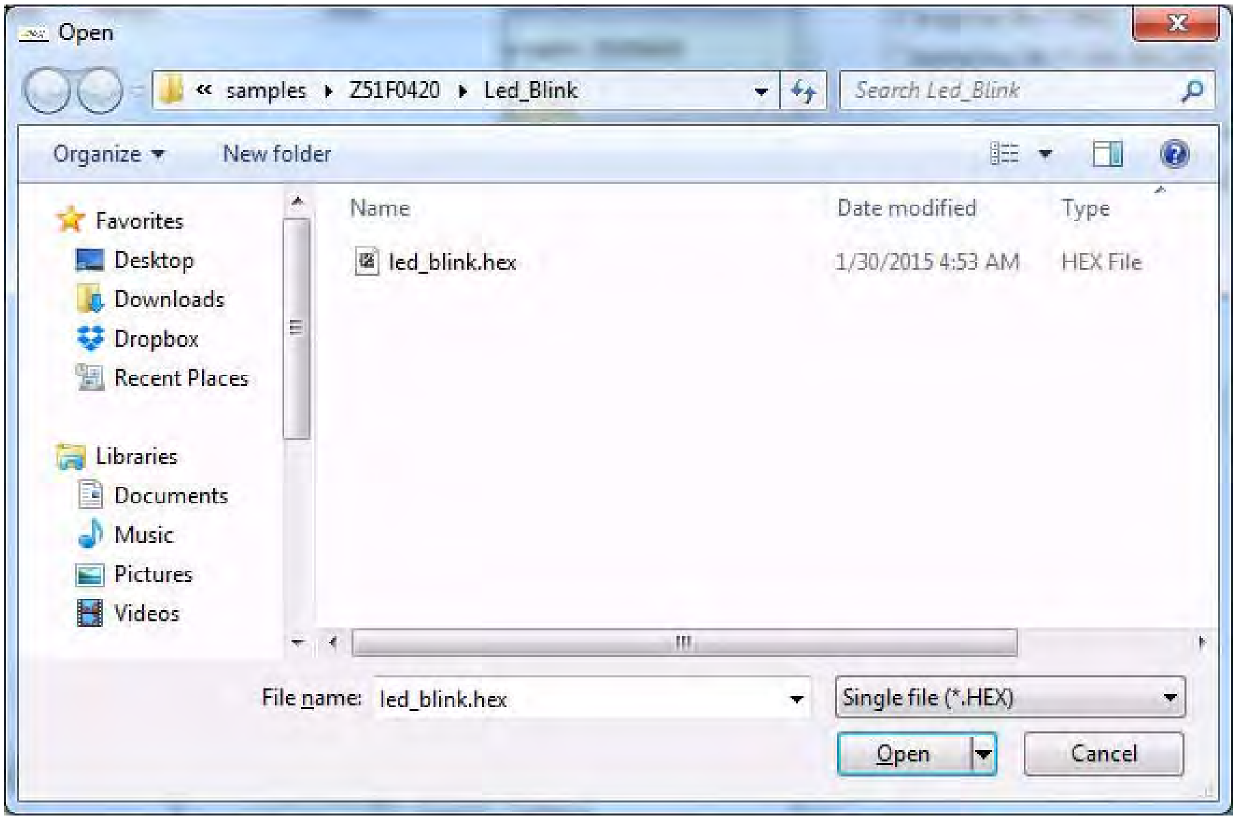
Symbol: C:\Program Files (x86)\ZiLOG\Z8051_2.2\samples\Z51F0420\Led_Blink\led_

[ ] Do verification after download

[ ] Keep device data (FILE data will be ignored in following range.)

Figure 70. The Hex File Download Dialog

3. In the Hex File Download dialog, click the **Browse** button to display the Open dialog shown in Figure 71.



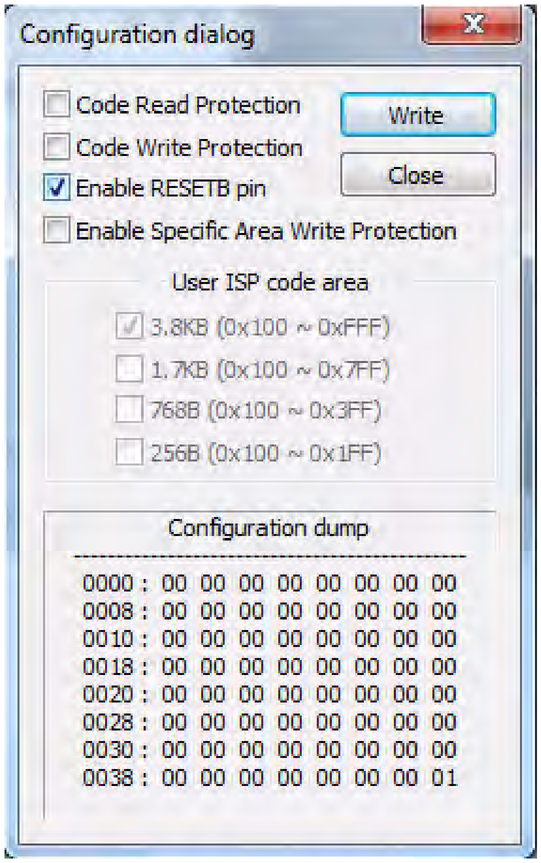
led_blink.hex 1/30/2015 4:53 AM HEX File

File name: led_blink.hex

Single file (*.HEX)

Figure 71. The Open Dialog

4. In the Open dialog, browse to and select the hex file that you wish to load into the memory space of the target MCU, then click the **Open** button to display this hex file in the Hex File Download dialog.
5. Click the **Download** button. The Configuration dialog will appear, as shown in Figure 72.



- [ ] Code Read Protection
- [ ] Code Write Protection
- [✓] Enable RESETB pin
- [ ] Enable Specific Area Write Protection

User ISP code area
- [✓] 3.8KB (0x100 ~ 0xFFF)
- [ ] 1.7KB (0x100 ~ 0x7FF)
- [ ] 768B (0x100 ~ 0x3FF)
- [ ] 256B (0x100 ~ 0x1FF)

```
Configuration dump
------------------------------------------------
0000 :  00  00  00  00  00  00  00  00
0008 :  00  00  00  00  00  00  00  00
0010 :  00  00  00  00  00  00  00  00
0018 :  00  00  00  00  00  00  00  00
0020 :  00  00  00  00  00  00  00  00
0028 :  00  00  00  00  00  00  00  00
0030 :  00  00  00  00  00  00  00  00
0038 :  00  00  00  00  00  00  00  01
```

Figure 72. The Configuration Dialog

6. In this Configuration dialog, click the **Write** button to complete the configuration of the target MCU’s Flash memory.

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.