Z8051™ Family of 8-Bit Microcontrollers

Z8051 On-Chip Debugger and In-System Programmer

User Manual

UM024001-0212

www.zilog.com

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

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 2012	01	Original issue.	All

Table of Contents

List of Figures

Introduction

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

The Z8051 On-Chip Debugger

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

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

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

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

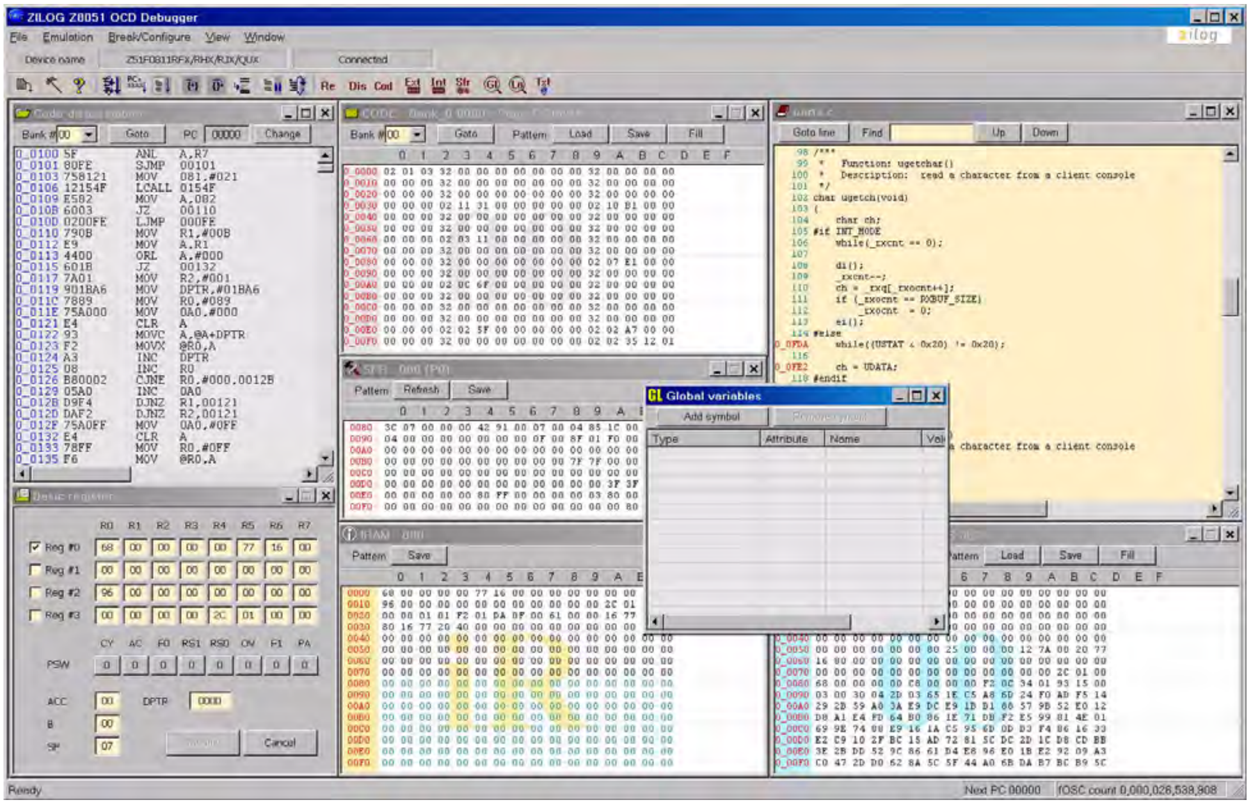


Figure 1. On-Chip Debugger Screen

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

Z8051 OCD Software and Documentation Installation

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

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

Note: In this filename, `<version_number>` refers to the version number of the OCD software. For example, this version number may be `1.0`.

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

Z8051 OCD Driver Installation

The driver programs for the Z8051 On-Chip Debugger are copied during the software and documentation installation. In the following procedure for Windows XP systems, ensure that the target side of the OCD will remain unconnected while you install these drivers. Refer to Figure 2 for guidance.

Figure 2. An Example Setup *[to be supplied]*

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

1. Connect the OCD hardware to the USB port of your PC by connecting the A-Male end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port, and connect the Mini-B end to the OCD device.
2. After the PC detects the new hardware, it will display the **Found New Hardware Wizard** dialog box, shown in Figure 3. Select **Install from a list or specific location (Advanced)**; then click **Next**.

Figure 3. The Found New Hardware Wizard Welcome Screen

3. The next dialog box, shown in Figure 4, prompts you to choose either the browse or entry options for where you will store the `.inf` file. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse** button to navigate to one of the following paths and click the **Next** button, leaving all other selections at their default settings. *[For correct and current branding, please change instances of "ZiLOG" to "Zilog".]*
   – On 32-bit machines, use the following path:

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

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

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

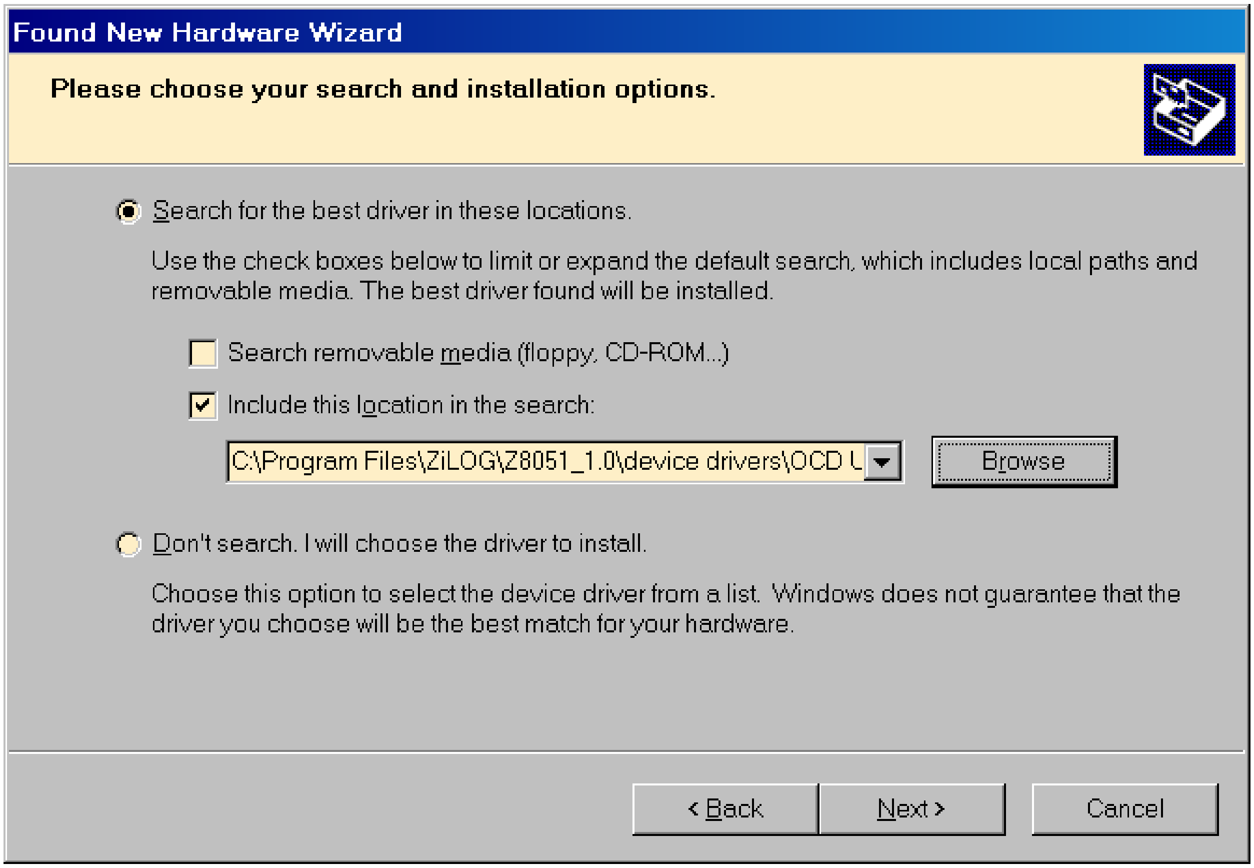


Figure 4. The Found New Hardware Wizard's Browse Screen

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

Understanding the OCD Menu Functions

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

File Menu

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

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

The OCD's File menu is shown in Figure 5.

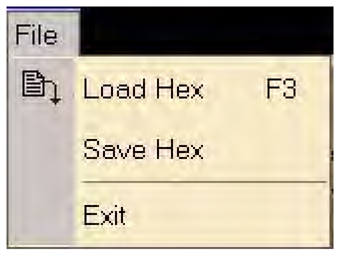


Figure 5. The OCD's File Menu

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

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 6.

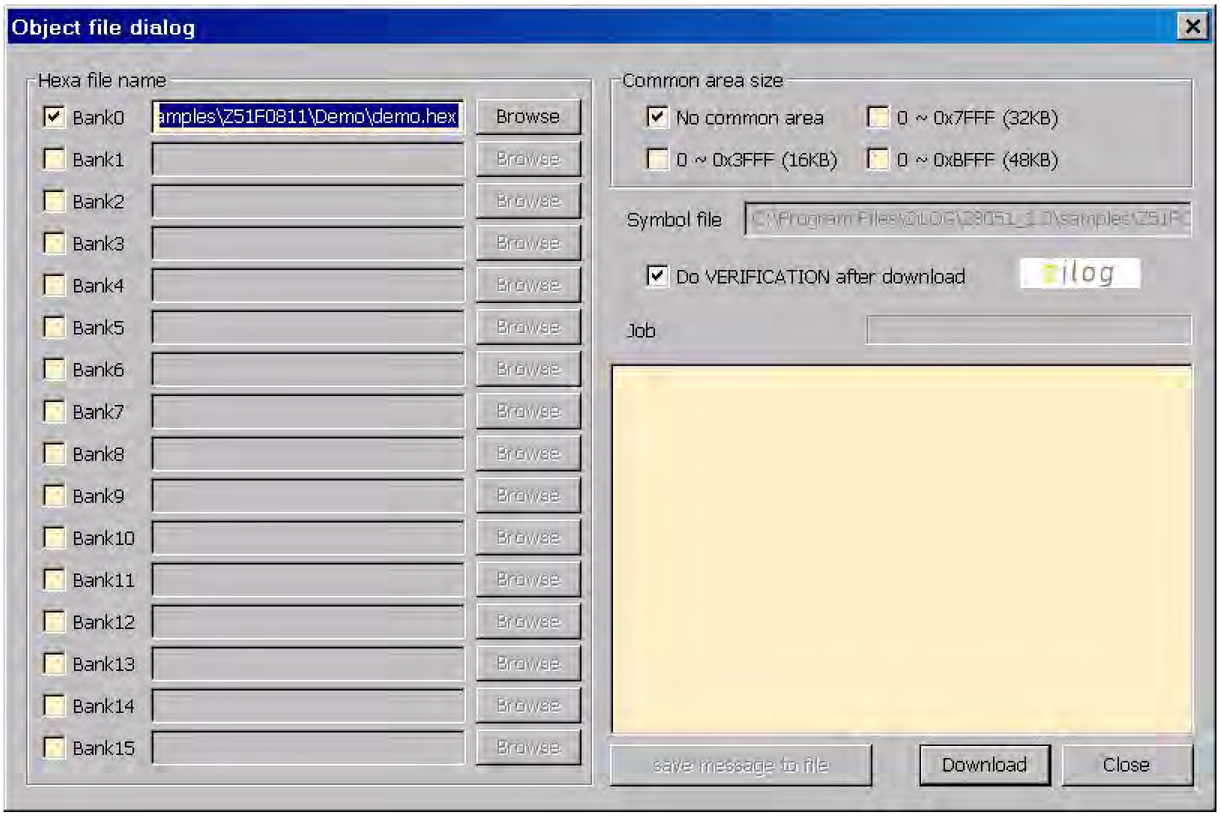


Figure 6. Object File Dialog

3. The **Hex file name** panel, located on the left side of the Object File dialog, displays 16 banks. If you are using the Z8051 MCU's LINEAR ADDRESS Mode, you are not required to select additional banks; LINEAR ADDRESS Mode uses only Bank 0. Click the **Browse** button for Bank 0 to display the Open File dialog shown in Figure 7.

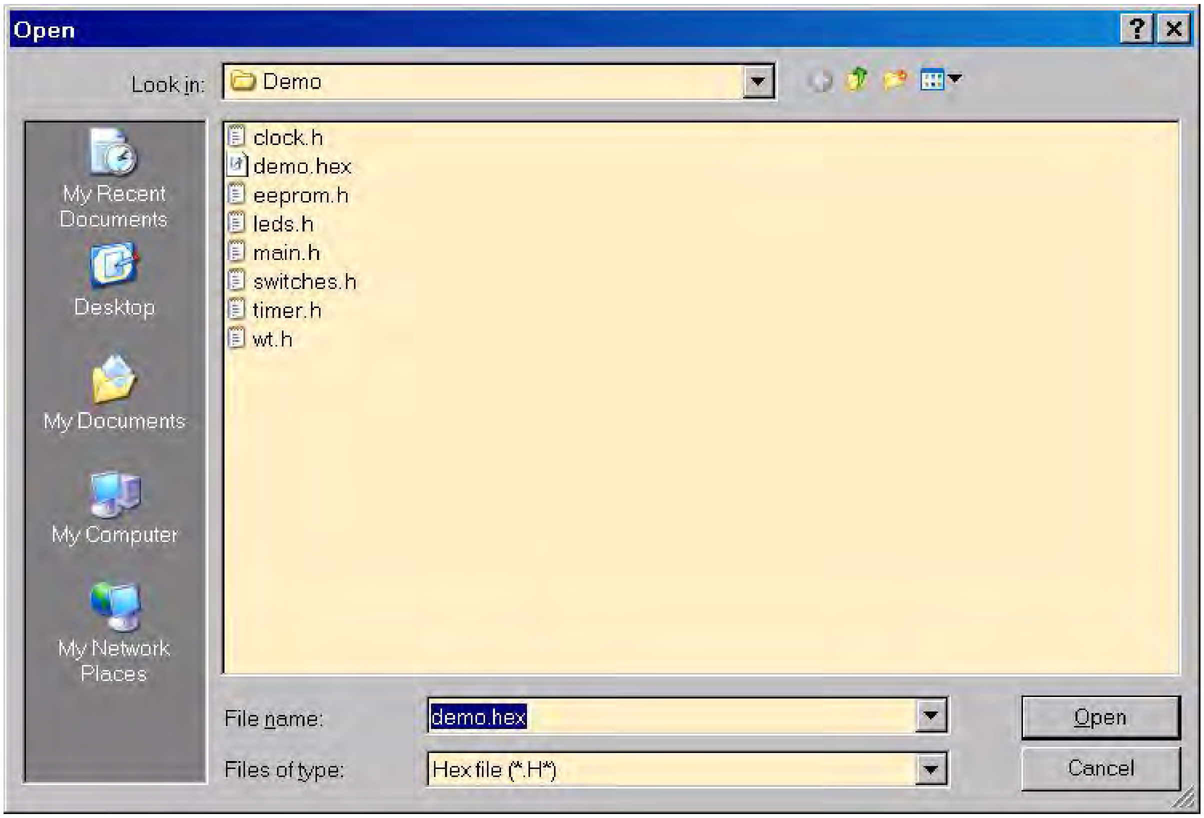


Figure 7. Open File Dialog

4. In the Open File dialog, select the hex file that you wish to load into the memory space of the target MCU, and click **OK**.

5. If previous PC breakpoints exist in the debugger environment, the Break Debug dialog box will appear, as shown in Figure 8. Click **Yes** if you wish to remove these breakpoints, or **No** if you prefer to retain them.

Figure 8. 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 9, lists the controls for starting or stopping an emulation routine. Use the Emulation menu to control the flow of code execution for debugging purposes.

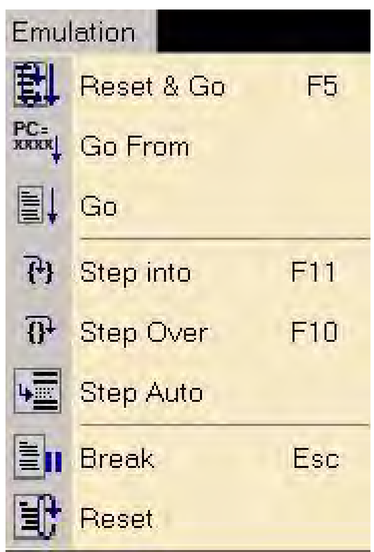


Figure 9. The OCD's Emulation Menu

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

Reset & Go

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

Go From

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

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

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

Go

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

- A stopped address *of [from the?]* previous emulation *[the last known address at the stopping of the last emulation that was run? Please clarify.]*.
- Where a break occurs, such that:
  - If a break occurs before the breakpoint, the current address is the PC breakpoint address
  - If a break occurs after the breakpoint, the current address is the next execution address of the PC breakpoint address
- If the target MCU was reset, the *[new?]* address is `0000h`.

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

Step

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

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

Step Over

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

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 100 ms; execution will continue unless the user halts it. This function is disabled (greyed out) in the Emulation menu during emulation.

Break

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

Reset

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

Break/Configure Menu

The Break/Configure menu, shown in Figure 10, lists PC breakpoint control, device configuration and hardware test functions.

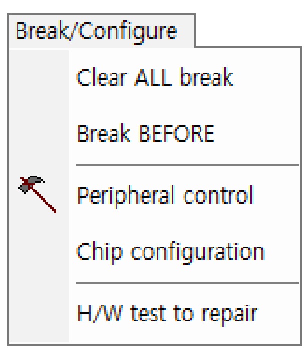


Figure 10. The OCD's Break/Configure Menu

Clear ALL Break

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

Break BEFORE (AFTER)

The Break BEFORE (AFTER) function prompts the user to select a PC breakpoint event either before or after execution. When selecting this menu option, the Break Control dialog box appears, prompting the user to choose one of these two conditions; see Figure 11.

Figure 11. Break Control Dialog

Selecting **Break before execution** causes a PC breakpoint when the PC reaches the PC breakpoint address; however, a PC breakpoint position will not be executed, as illustrated in the timing diagram shown in Figure 12.

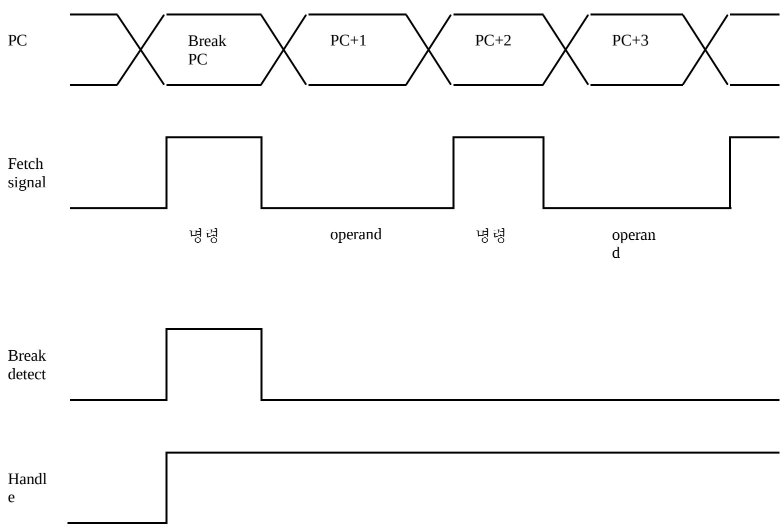


Figure 12. Break BEFORE Timing Diagram

Selecting **break after execution** causes a PC breakpoint to occur when the PC reaches the PC breakpoint address, and a PC breakpoint position is executed, as illustrated in the timing diagram shown in Figure 13.

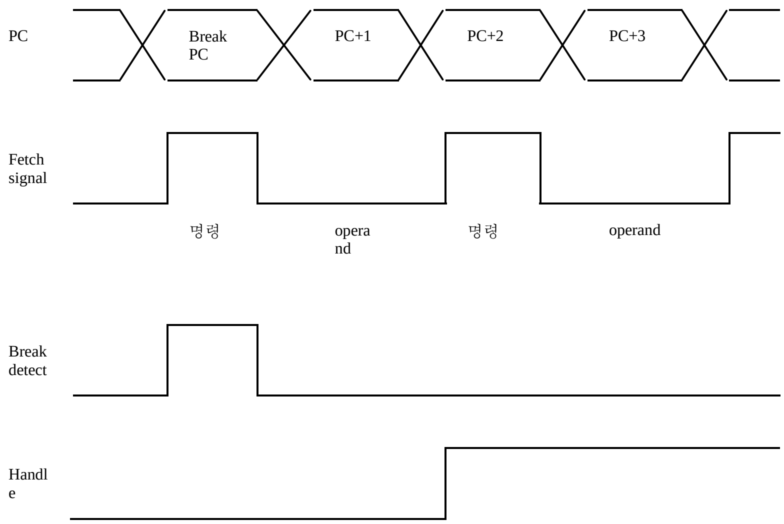


Figure 13. Break AFTER Timing Diagram

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

Peripheral Control

Selecting the Peripheral Control function from the Break/Configure menu prompts the user to determine whether the target MCU's internal peripheral functions should continue to operate or remain idle, as shown in Figure 14. These peripherals are always running during emulation by default.

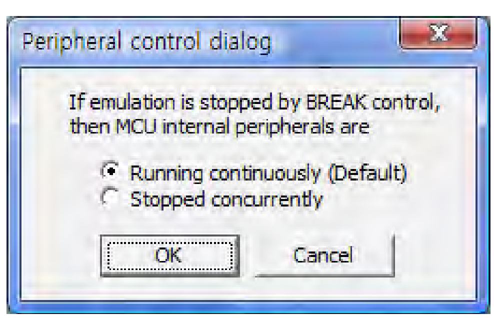


Figure 14. Peripheral Control Dialog

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

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

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

Chip Configuration

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

The Configuration dialog box shown in Figure 15 offers an example configuration for the Z51F0811 device.

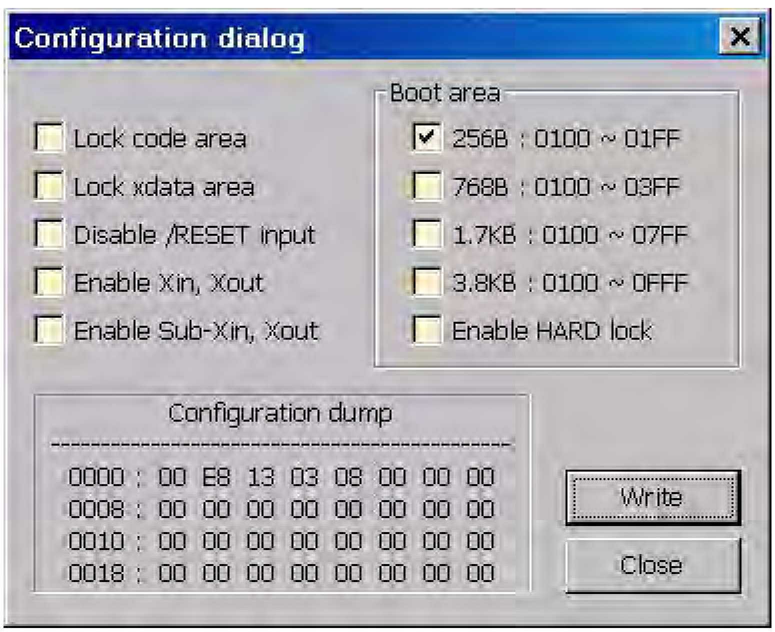


Figure 15. Z51F0811 MCU Configuration Example

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

Hardware Test to Repair

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

View Menu

The View menu, shown in Figure 16, supports the opening of child windows. The Debugger offers the following nine types of child windows:

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

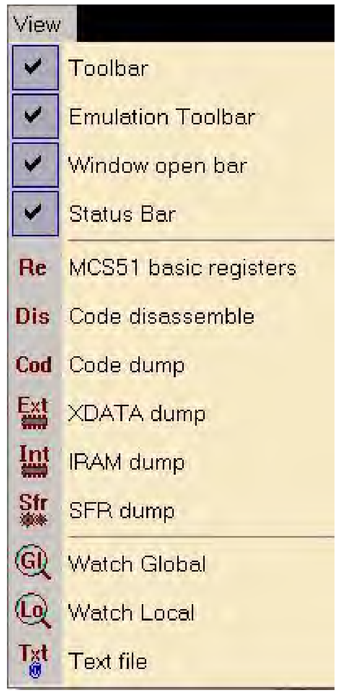


Figure 16. The OCD's View Menu

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

Toolbar

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

Emulation Toolbar

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

Window Open Bar

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

Status Bar

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

Z8051 Basic Registers

This menu selection opens a window that displays the Z8051 Series' basic registers. If this window is already open, selecting the **Z8051 Basic Registers** option will cause this window to appear at the top-most level of the debugger frame. See Figure 17.

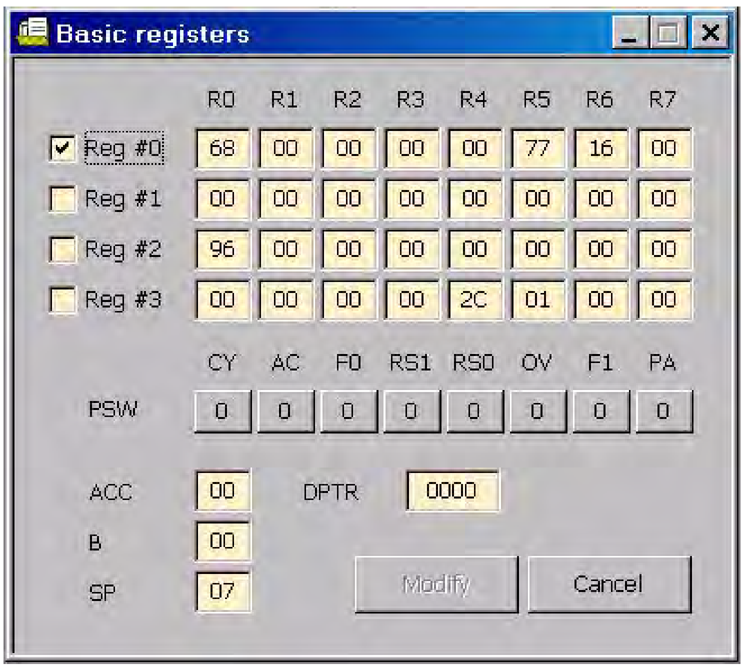


Figure 17. The Basic Registers Dialog

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

Code Disassembly

This menu selection opens a window which displays the memory spaces containing disassembled code. If this window is already open, selecting **Code Disassemble** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 18.

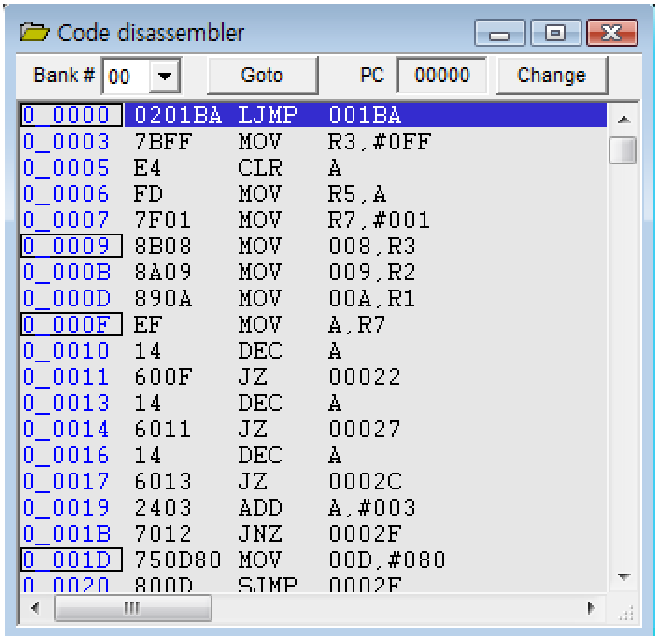


Figure 18. Code Disassembler Dialog

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

Code Dump

This menu selection opens a window which displays the contents of code memory in a *dumped* format. If this window is already open, selecting **Code Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 19.

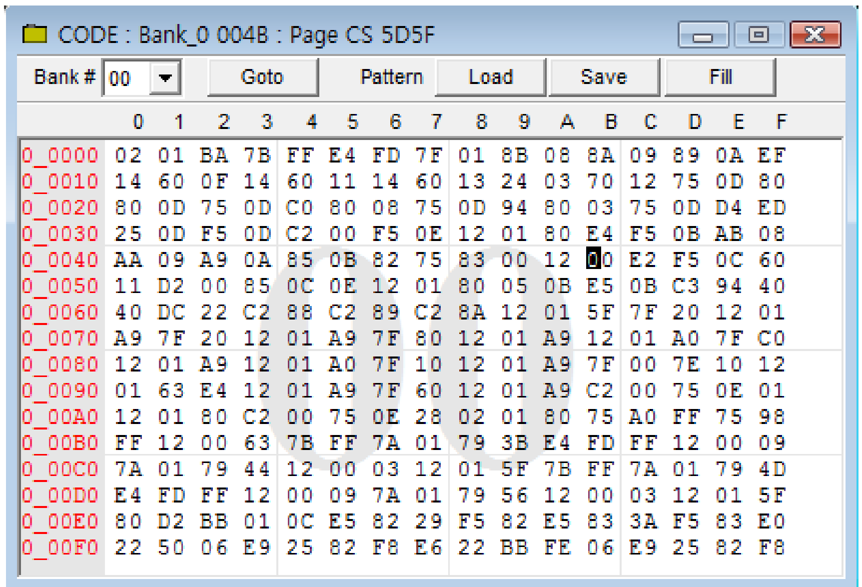


Figure 19. Code Dump Dialog

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

XDATA Dump

This menu selection opens a window which displays the contents of XDATA memory in a dumped format. The term *XDATA* refers to the external data memory contained in Z8051 Series devices. If this window is already open, selecting **XDATA Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 20.

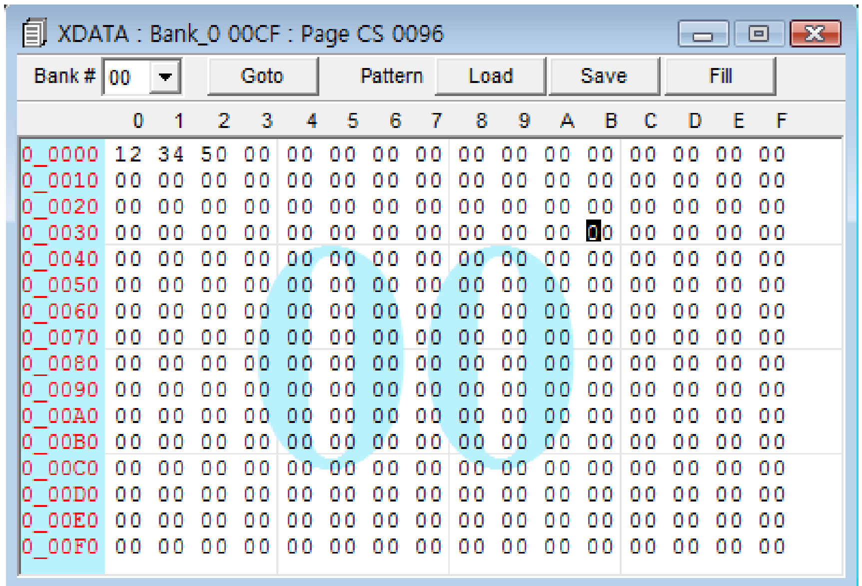


Figure 20. XDATA Dump Dialog

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

IRAM Dump

This menu selection opens a window which displays the contents of IRAM memory in a dumped format. The term *IRAM* refers to the internal data memory contained in Z8051 Series devices. If this window is already open, selecting **IRAM Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 21.

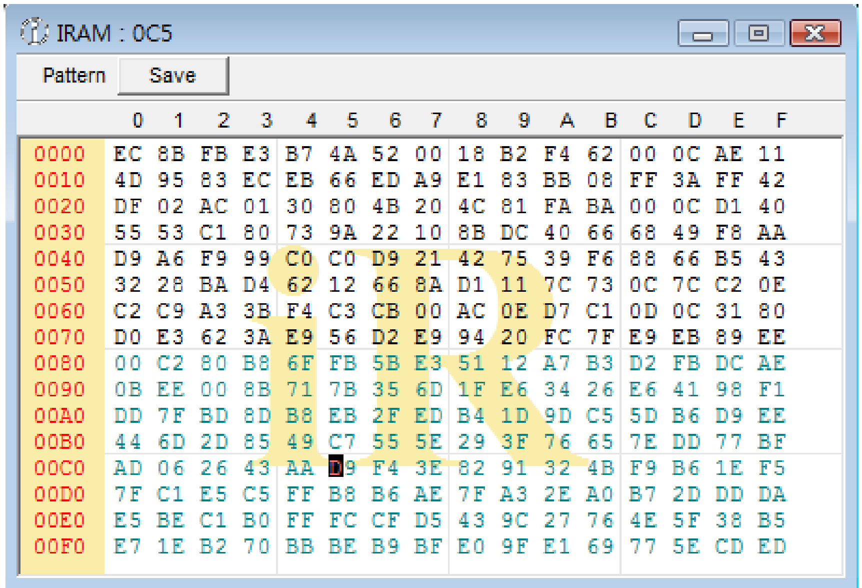


Figure 21. IRAM Dump Dialog

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

SFR Dump

This menu selection opens a window which displays the contents of the SFR peripherals in a dumped format. The term *SFR* refers to the special function registers contained in Z8051 Series devices. If this window is already open, selecting **SFR Dump** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 22.

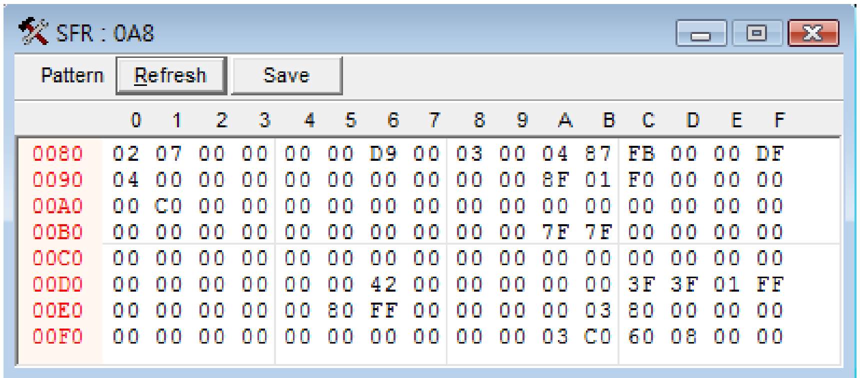


Figure 22. SFR Dump Dialog

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

Watch Global

This menu selection opens a window that displays global variables. If this window is already open, selecting **Watch Global** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 23.

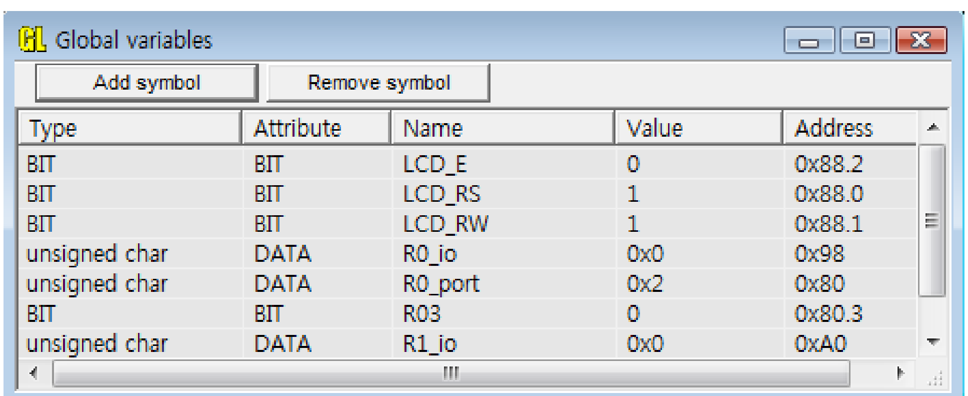


Figure 23. Global Variables Dialog

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

Watch Local

This menu selection opens a window that displays local variables. If this window is already open, selecting **Watch Local** from the View menu will cause this window to appear at the top-most level of the debugger frame. See the example in Figure 24.

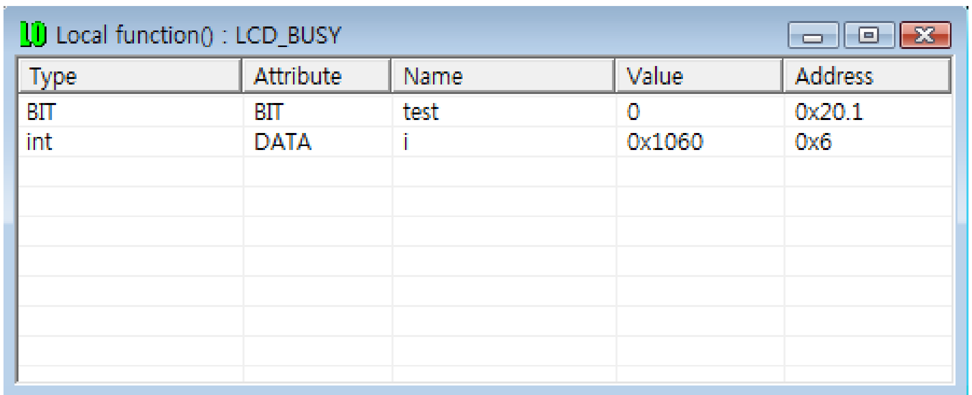


Figure 24. Local Function Dialog

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

Text File

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

If a selected text file is already open, selecting **Text File** from the View menu will cause the window containing the text file to appear at the top-most level of the debugger frame; otherwise, selecting **Text File** will open a new window. See the example text file in Figure 25.

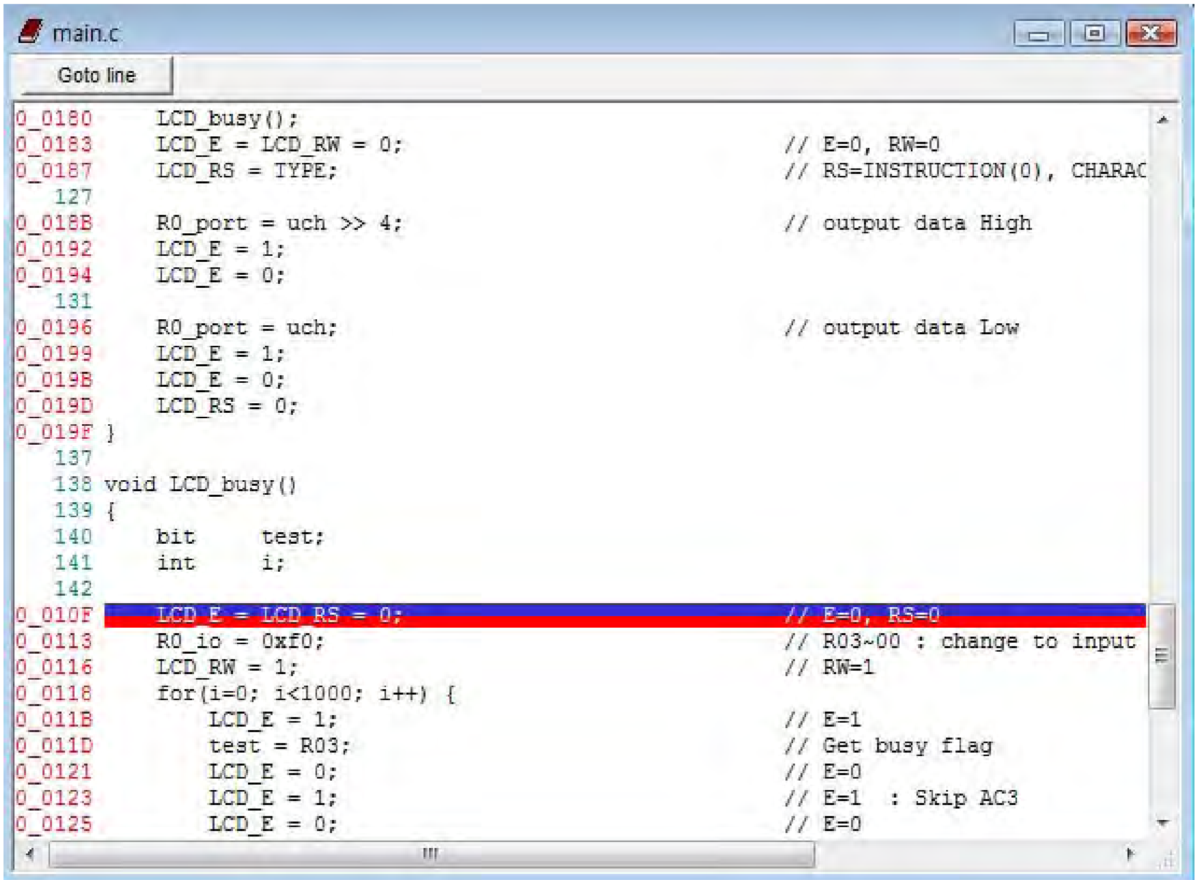


Figure 25. A Sample Text File

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

Window Menu

The Window menu, shown in Figure 26, can be used to modify the arrangement of child windows or to directly select a child window.

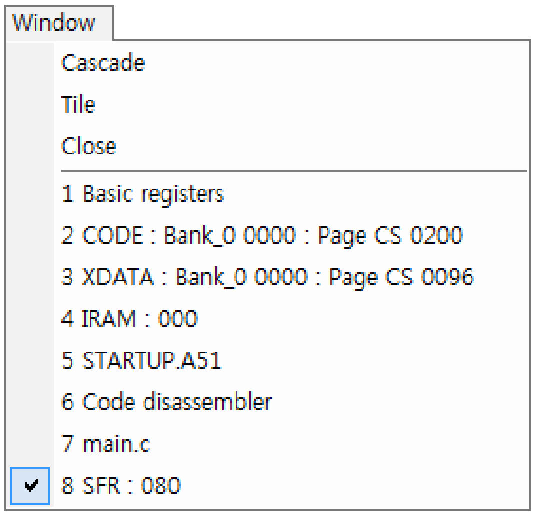


Figure 26. The OCD's Window Menu

Cascade

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

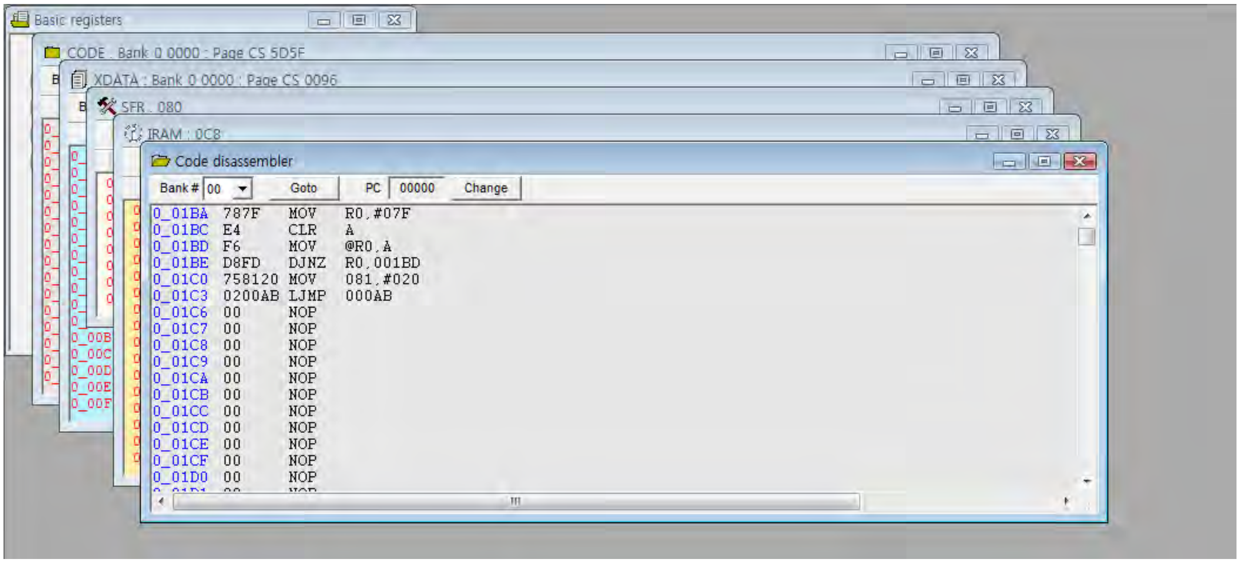


Figure 27. Cascaded Windows

Tile

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

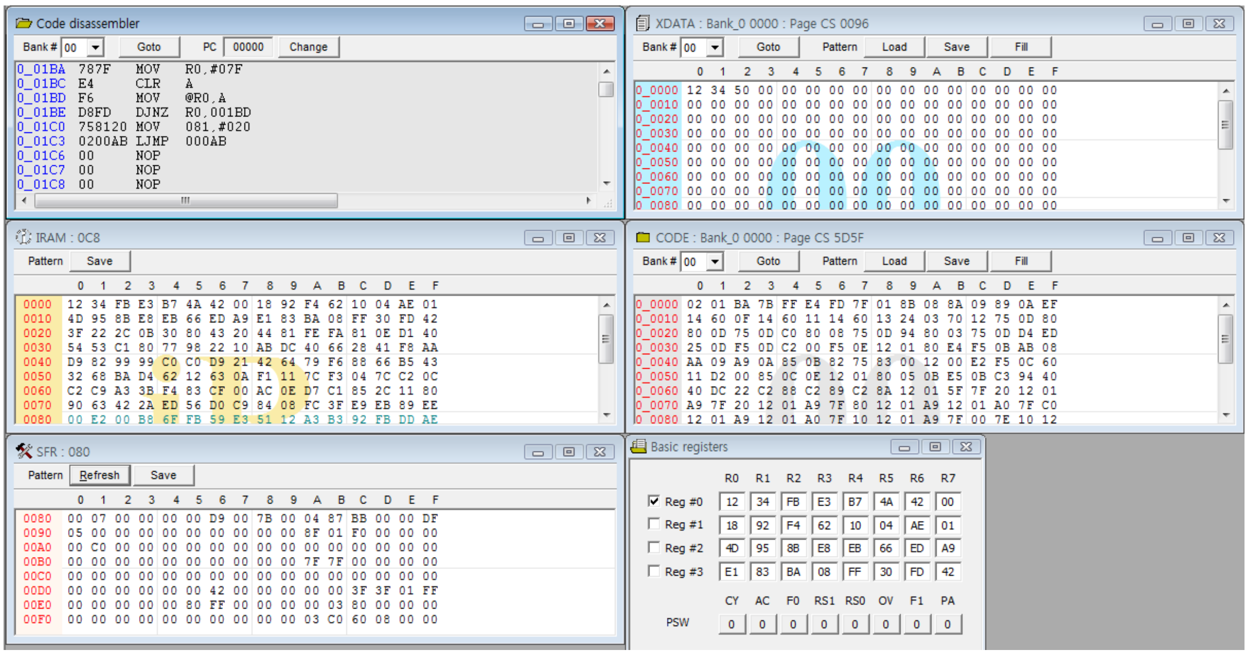


Figure 28. Tiled Windows

Close

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

Windows 1, 2, 3, Etc.

This menu selection assigns a sequential number (e.g., 1, 2, 3…) to each child window in the order in which it is opened. Users can directly select any open child window by its number. In Figure 26 on page 26, for example, selecting **6** from the **Window** menu will display the Code Disassembler window as the top-most window in the Debugger screen.

Child Windows

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

Z8051 Basic Registers Window

The Z8051 Basic Registers window allows users to edit the contents of the Z8051 Series registers. Figure 29 shows an example Z8051 Basic Registers window.

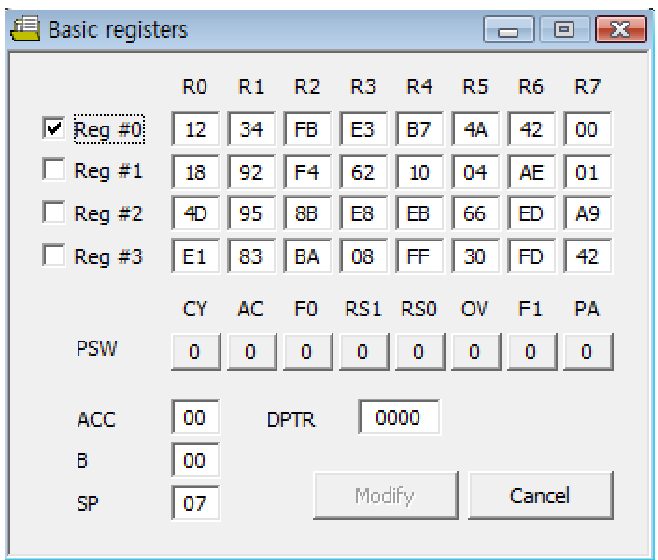


Figure 29. Using the Basic Registers Function, #1 of 6

Edit

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

In Figure 30, the current register bank is highlighted in the red area. Users can change register banks by selecting or deselecting any of the registers in this current register bank.

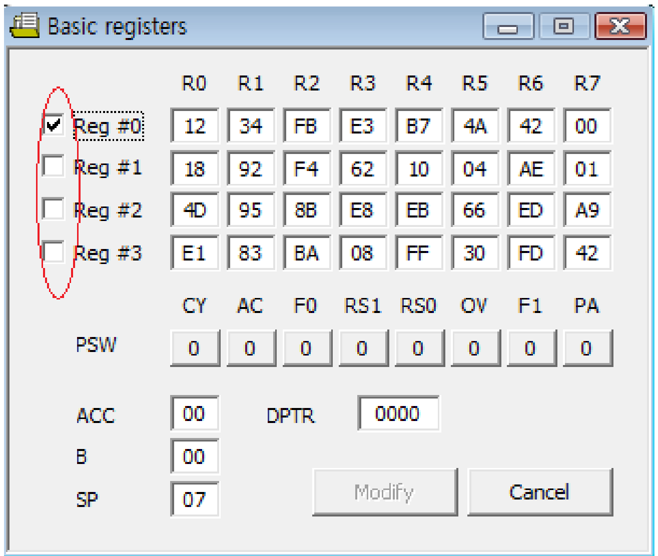


Figure 30. Using the Basic Registers Function, #2 of 6

In Figure 31, the R0–R7 registers are highlighted in the red area. These registers map to the same area as IRAM addresses in the range `00h–1Fh`. Users can change these values by entering 8-bit hexadecimal formats.

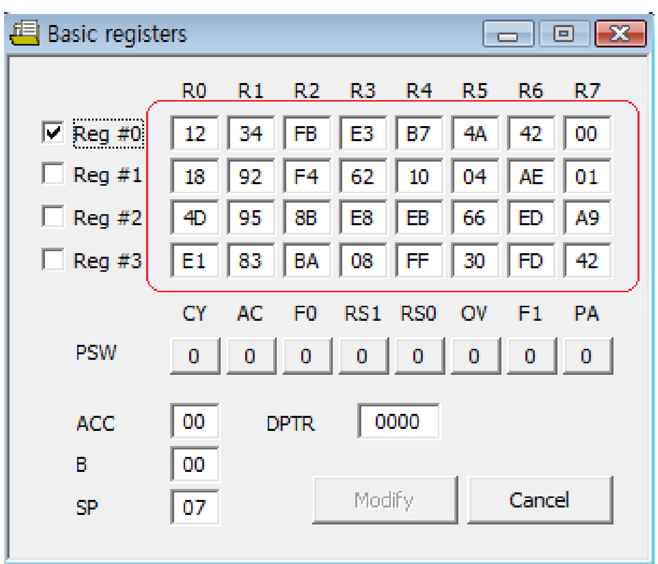


Figure 31. Using the Basic Registers Function, #3 of 6

In Figure 32, the red area highlights the Program Status Word (PSW), in which bit units can be changed.

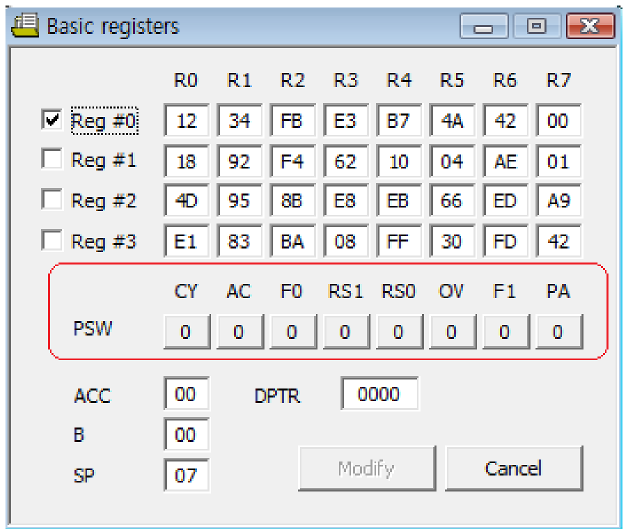


Figure 32. Using the Basic Registers Function, #4 of 6

In Figure 33, the red area highlights the Accumulator (ACC), the B Register (B) and the Stack Pointer (SP) registers. Enter a number in n 8-bit hexadecimal format to change any of these values.

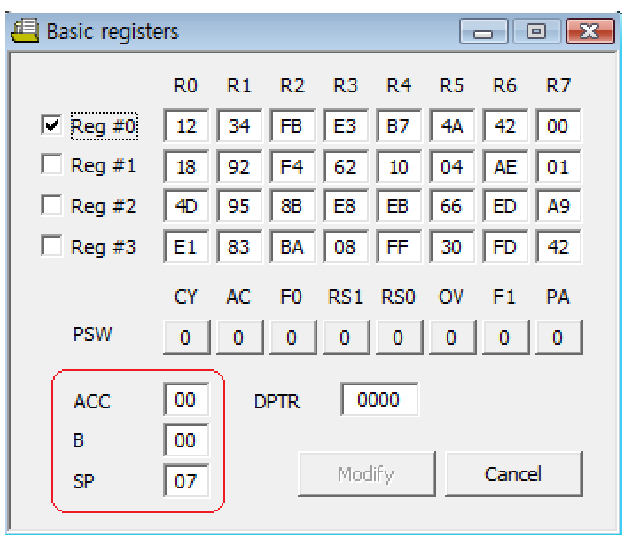


Figure 33. Using the Basic Registers Function, #5 of 6

In Figure 34, the red area highlights the DPTR Register which displays, and can be edited by, entering numbers in the 16-bit hexadecimal format. If the target MCU features more than two DPTRs, the DPTR field in this dialog shows the currently selected register. If each DPTR resides at a different address, Zilog recommends using the SFR window instead.

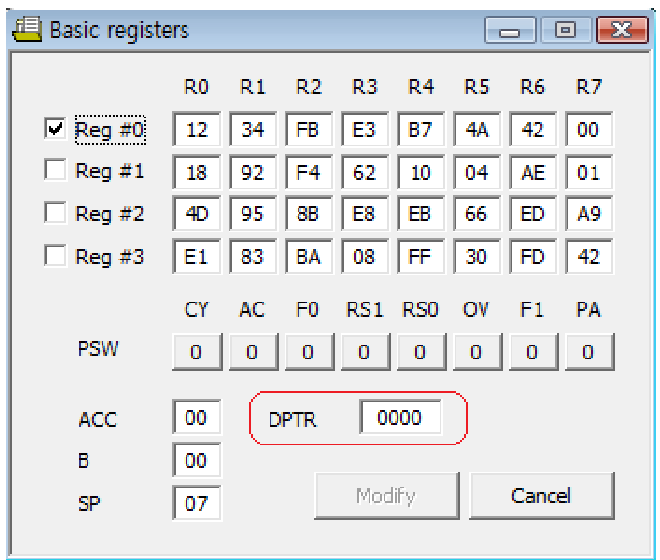


Figure 34. Using the Basic Registers Function, #6 of 6

Code Disassemble Window

The Code Disassemble window displays the contents of code memory by using a disassemble format. All operand values must be entered in hexadecimal format. Figure 35 shows an example Code Disassembler window.

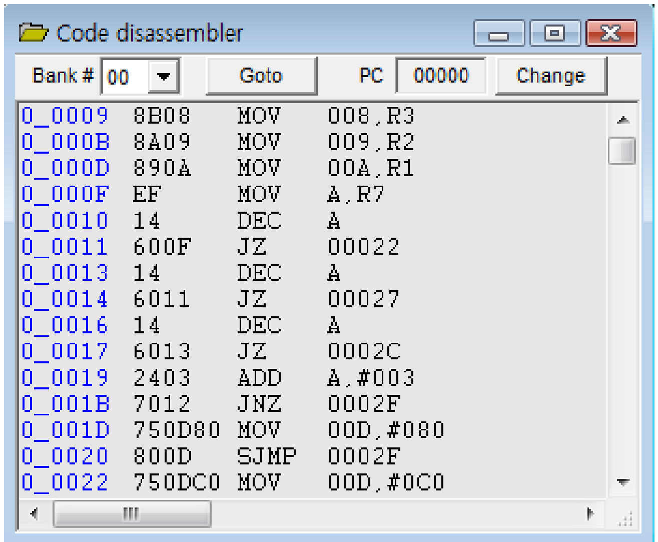


Figure 35. Using the Code Disassembler Function, #1 of 3

If map/symbol files are already loaded, the affected source lines are highlighted by boxes, as shown in Figure 36. Double-click any of these highlighted boxes to open its source file and move to the appropriate address line.

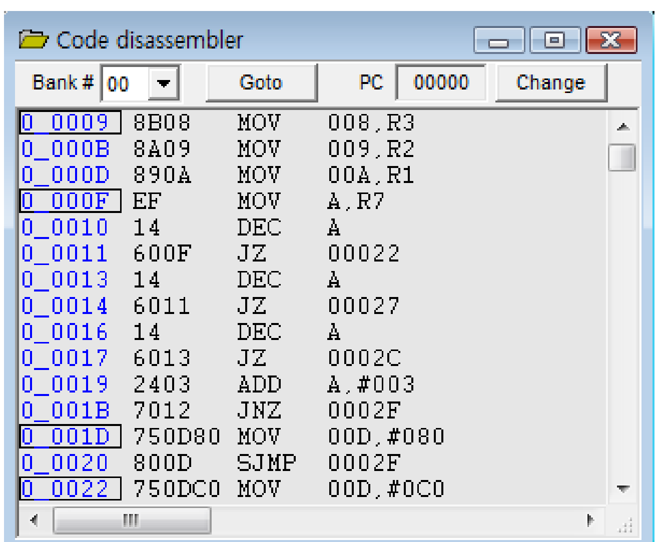


Figure 36. Using the Code Disassembler Function, #2 of 3

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

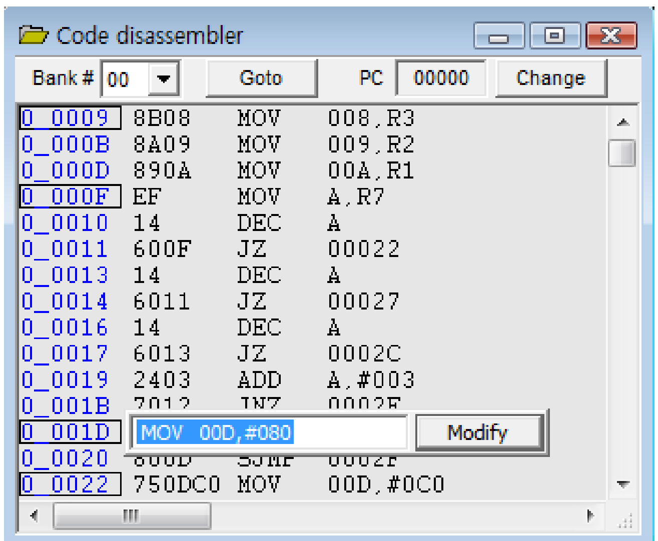


Figure 37. Using the Code Assembler Function, #3 of 3

PC Break Toggle

The target MCU's internal Program Counter (PC), sets or clears all PC breakpoint settings. The PC breakpoint count differs in each device in the Z8051 Series; normally, eight breakpoints can be set. In Figure 38, the red line represents a program counter breakpoint in the line, and the blue line represents the current program counter.

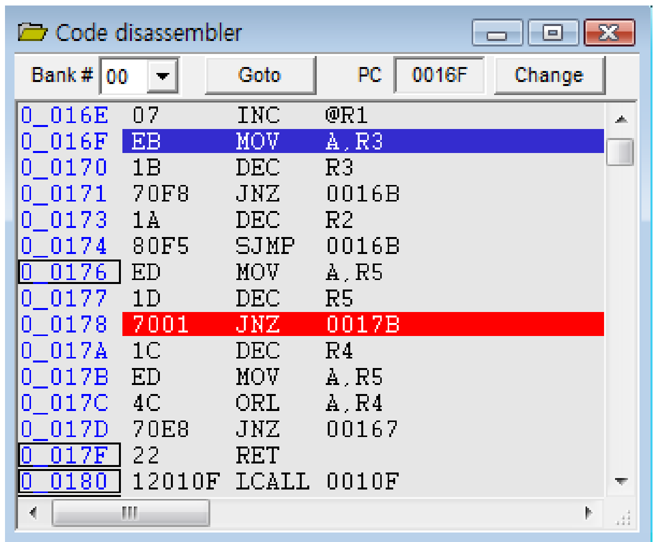


Figure 38. Using the PC Break Toggle Function

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

Code Dump Window

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

The upper side of the Code Dump window displays the address of the current cursor position and the checksum of the current page. The current page number is displayed as a watermark in the center of this window. In Figure 39, for example, the page number is `00`.

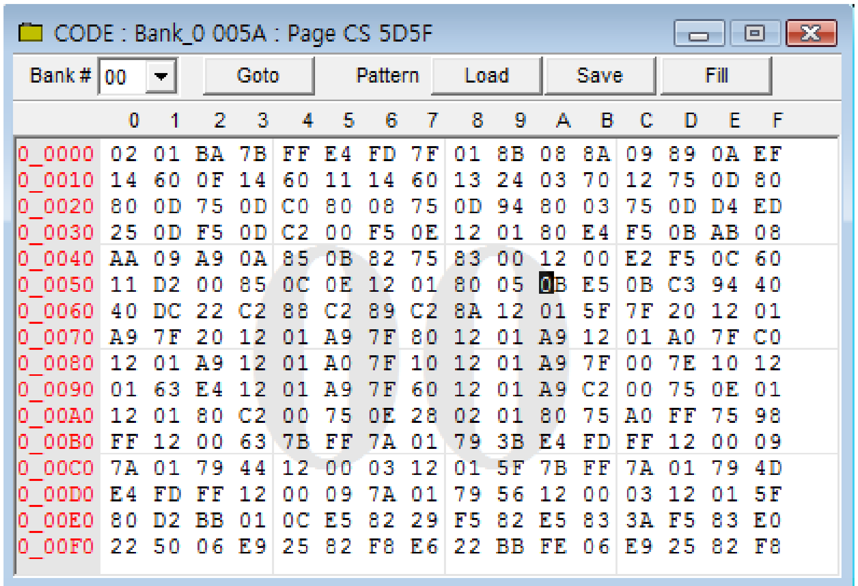


Figure 39. Using the Code Dump Function, #1 of 2

Edit

Users can change data values in the Code Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 40.

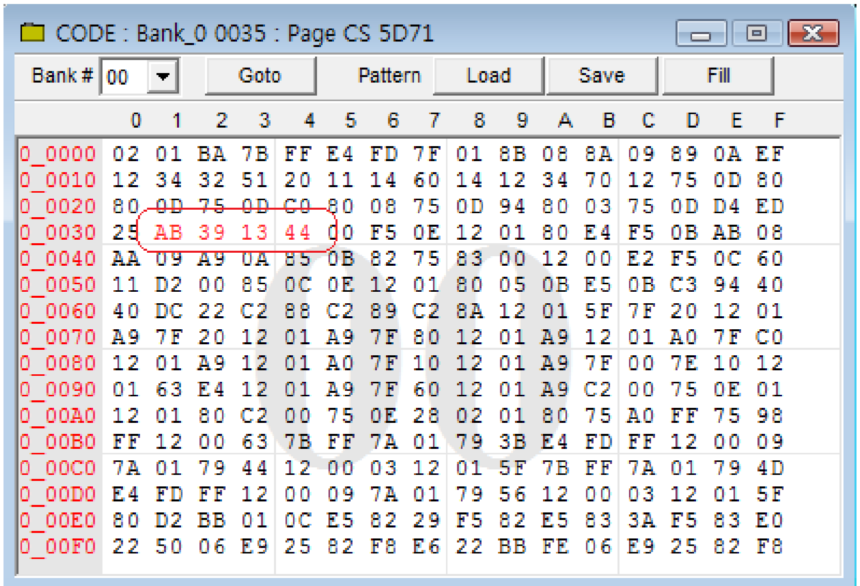


Figure 40. Using the Code Dump Function, #2 of 2

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

Bank

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

Goto

Click the **Goto** button to view memory locations in any 16-bit segments within the `0000h–FFFFh` address range in the Code Dump window or edit these memory locations by entering an address in hexadecimal format. See the example Input dialog in Figure 41.

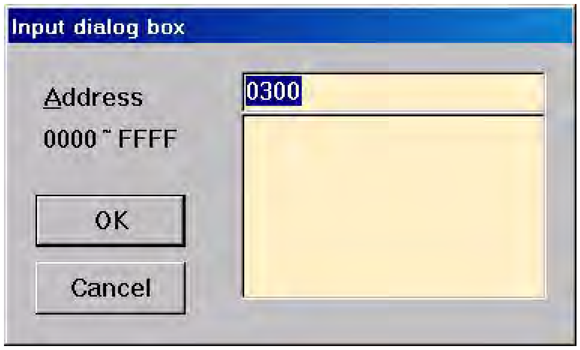


Figure 41. The Code Dump Function's Goto/Input Dialog

Load

Click the **Load** button to display the Pattern Load dialog, in which you can load a data pattern or code hex file to the code space; see Figure 42.

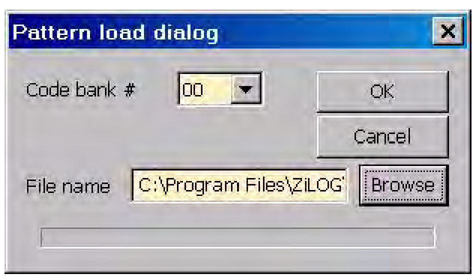


Figure 42. The Code Dump Function's Pattern Load Dialog

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

Save

Click the **Save** button to display the Pattern Save dialog, in which you can save a code space as a pattern file; see the example in Figure 43.

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

Fill

Click the **Fill** button to display the Pattern Fill dialog, in which you can write a common value in all code memory spaces in a specified address range; see the example in Figure 44.

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

XDATA Dump Window

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

The upper side of the XDATA Dump window displays the address of the current cursor position and the checksum of the current page. The current page number is displayed as a watermark in the center of this window. In Figure 45, for example, the page number is `00`.

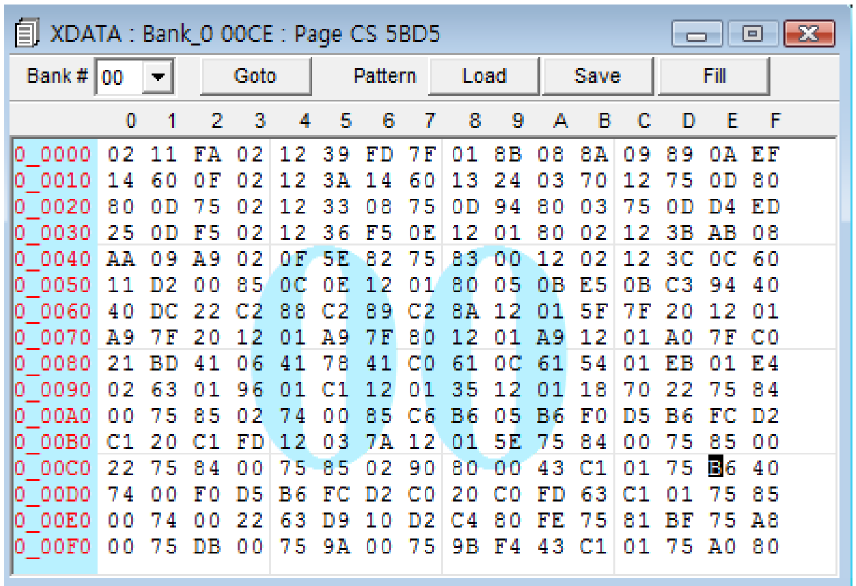


Figure 45. Using the XDATA Dump Function, #1 of 2

Edit

Users can change data values in the Code Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 46.

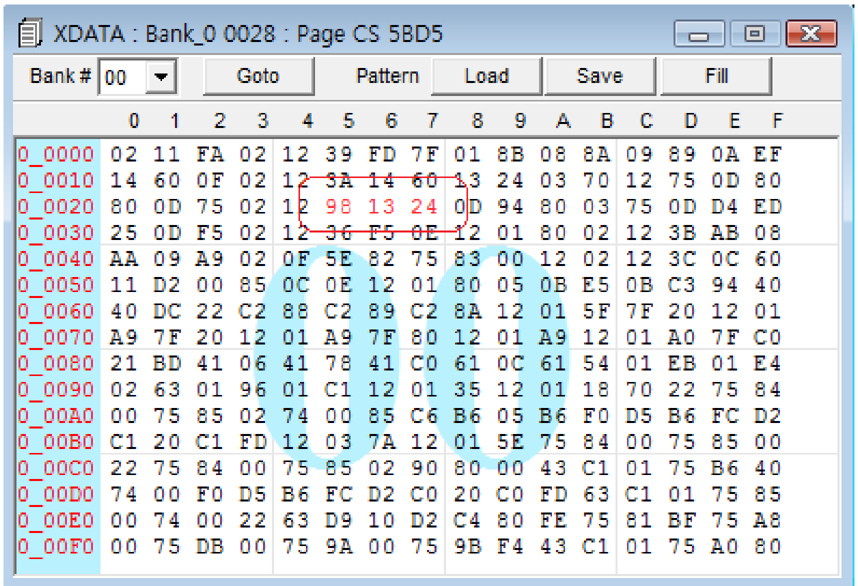


Figure 46. Using the XDATA Dump Function, #2 of 2

Bank

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

Goto

Click the **Goto** button to view memory locations in any 16-bit segments within the `0000h–FFFFh` address range in the XDATA Dump window or edit these memory locations by entering an address in hexadecimal format. See the example in Figure 47.

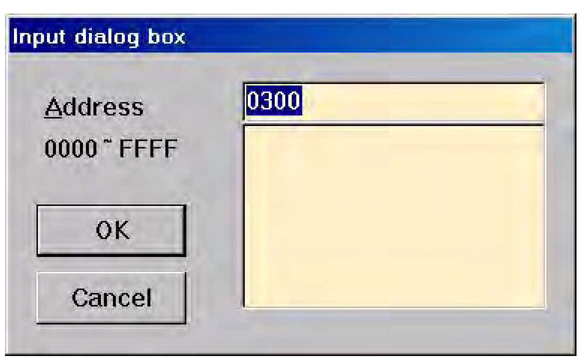


Figure 47. The XDATA Dump Function's Goto/Input Dialog

Load

Click the **Load** button to display the Pattern Load dialog, in which you can load a data pattern or code hex file to the XDATA area. However, this command does not clear the XDATA area; see Figure 48.

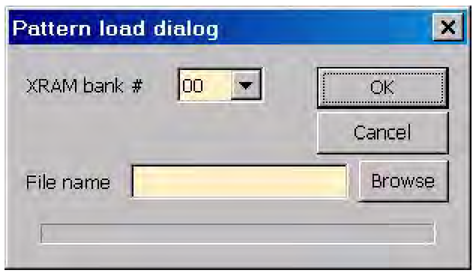


Figure 48. The XDATA Dump Function's Pattern Load Dialog

Save

Click the **Save** button to display the Pattern Save dialog, in which you can save the XDATA area as a pattern file; see Figure 49.

Figure 49. The XDATA Dump Function's Pattern Save Dialog

Fill

Click the **Fill** button to display the Pattern Fill dialog, in which you can write a common value in all XDATA memory spaces in a specified address range; see the example in Figure 50.

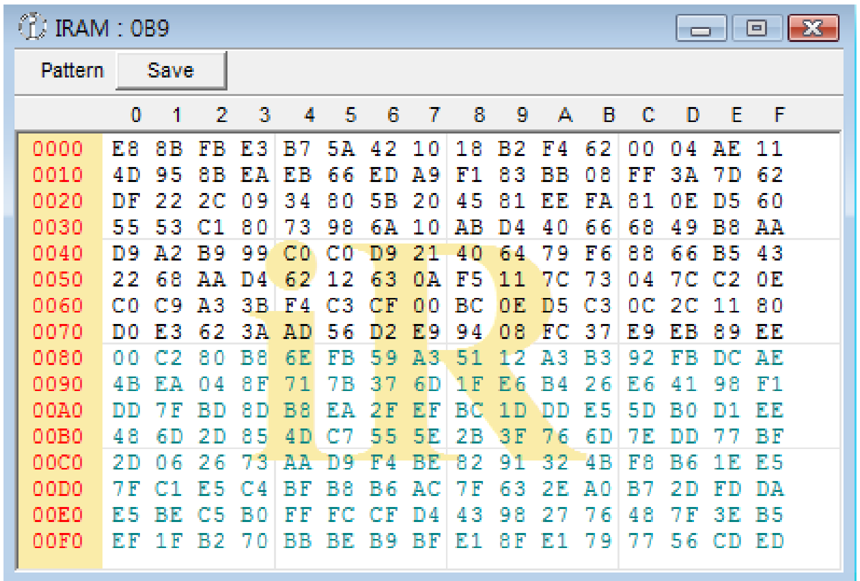
Figure 50. The XDATA Dump Function's Pattern Fill Dialog

IRAM Dump Window

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

The upper side of the IRAM Dump window displays the address of the current cursor position and the checksum of the current page. A watermark, displayed as `iR`, appears in the center of this window, as shown in Figure 51.

Figure 51 also shows IRAM addresses in the range `00h–7Fh`, which represent the direct area; the characters representing these addresses are colored black. IRAM addresses in the range `80h–FFh` represent the indirect area; these characters are colored cyan.



	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 51. Using the IRAM Dump Function, #1 of 2

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

Edit

Users can change data values in the IRAM Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 52.

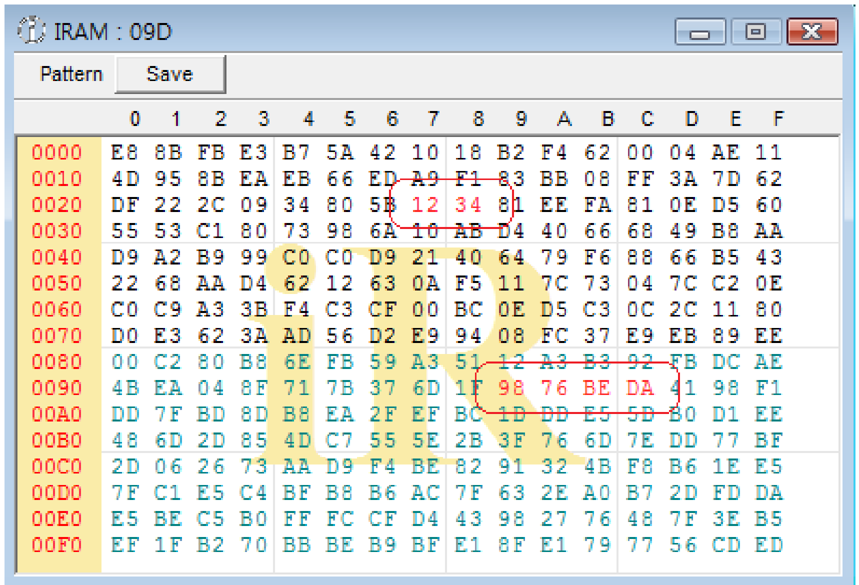


Figure 52. Using the IRAM Dump Function, #2 of 2

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

Save

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

SFR Dump Window

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

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

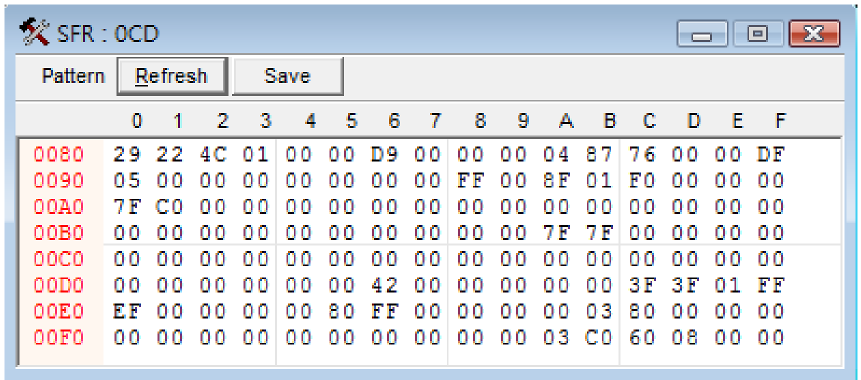


Figure 53. Using the SFR Dump Function, #1 of 3

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

Edit

Users can change data values in the SFR Dump window at any time, except during emulation. The editing method is quite simple; just place the cursor where you wish to make an edit, and write a new character pair in hexadecimal format. The color of the character pair will change from black to red to indicate that the change was made, as highlighted in Figure 54.

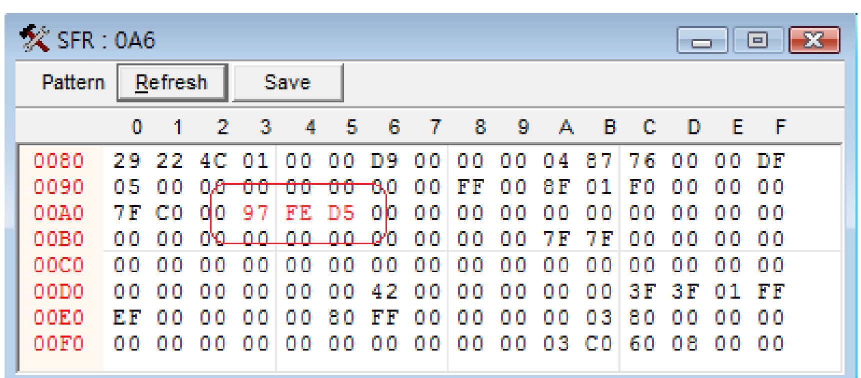


Figure 54. Using the SFR Dump Function, #2 of 3

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

Refresh

The SFR area includes static registers such as a stack pointer, an accumulator, etc. However, most SFRs are dynamic registers such as timers, I/Os, etc. Clicking the **Refresh** button (highlighted in Figure 55) redisplays all current data.

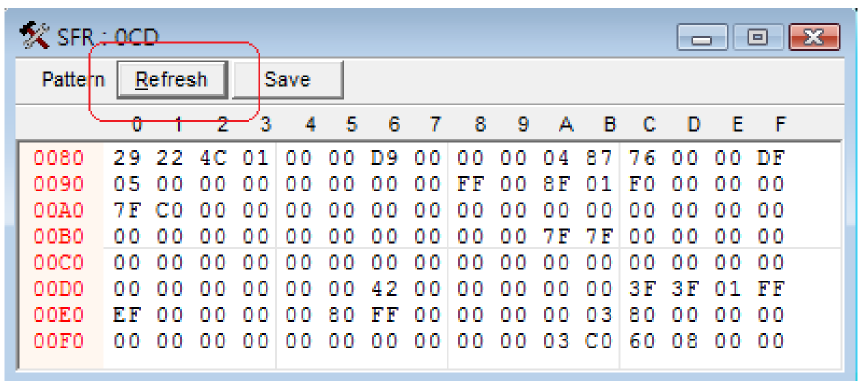


Figure 55. Using the SFRDump Function, #3 of 3

Save

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

Watch Global Window

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

The Watch Global window alleviates this problem by employing a map/symbol file; see Figure 56.

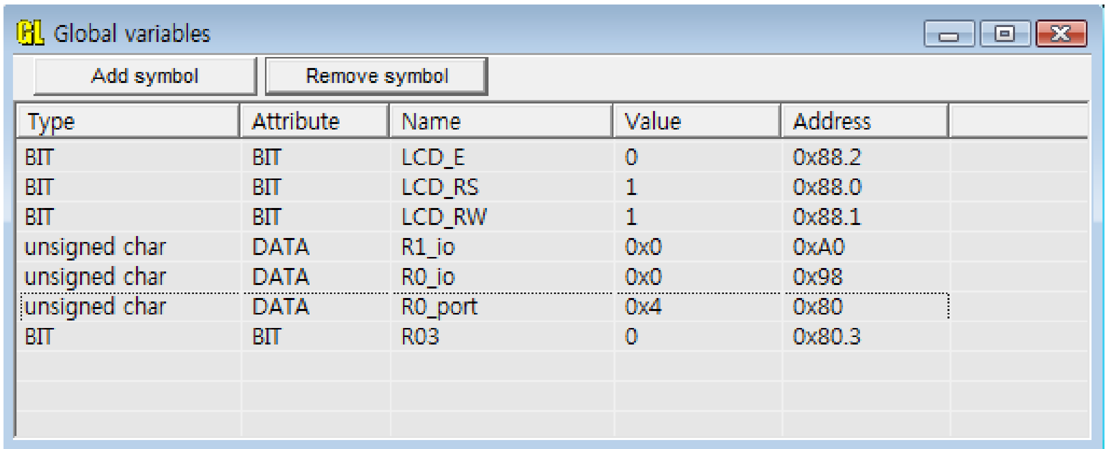


Figure 56. The Watch Global Function's Global Variables Dialog

Add Symbol

Clicking the **Add Symbol** button displays the Global Symbol Add/Remove dialog, in which you can add a global variable to the Watch Global display list, shown in Figure 57.

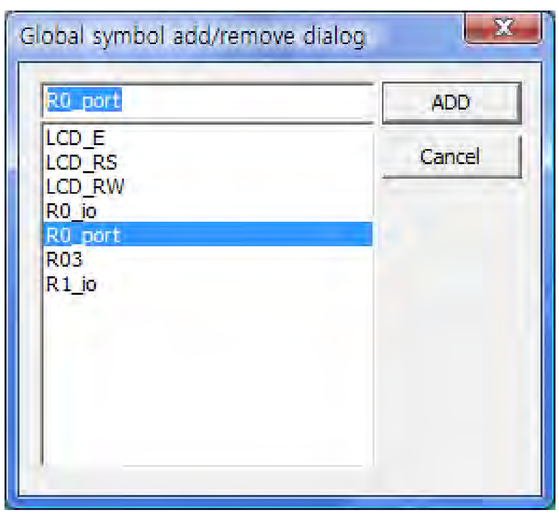


Figure 57. Adding A Global Symbol

Remove Symbol

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

Edit

Users can change data values in the Watch Global window at any time, except during emulation. This editing method is quite simple; just place the cursor where you wish to make an edit, and double-click the left button on your mouse to display a pop-up dialog in which you can change the data and click the **Modify** pop-up button to incorporate the change, as shown in Figure 58.

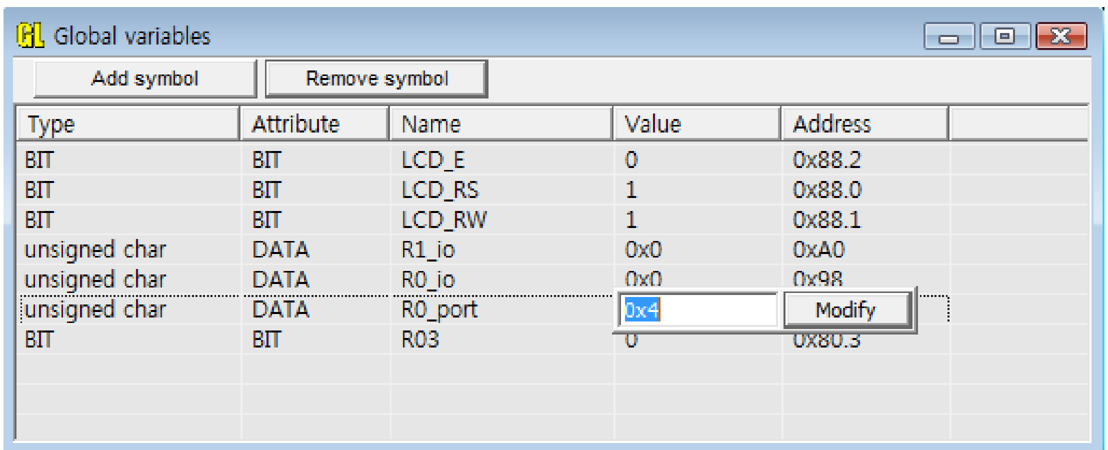


Figure 58. Editing A Global Symbol

Watch Local Window

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

Much like the issue with finding global variables, users could edit these local variables directly if finding them was not so tedious. The Watch Local window, shown in Figure 59, alleviates this problem by employing a map/symbol file, as described in the previous section.

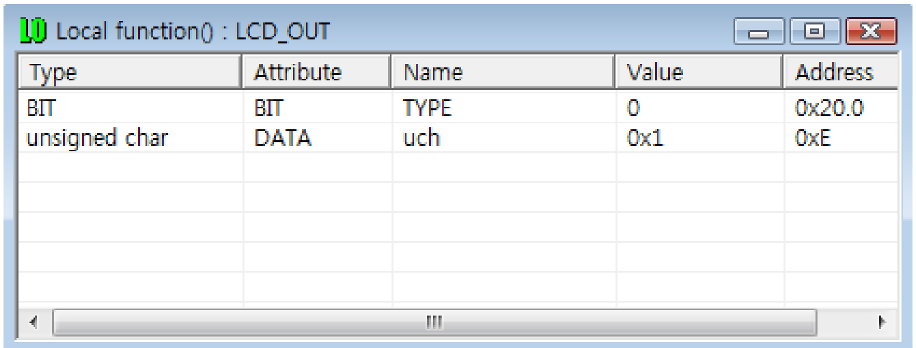


Figure 59. The Watch Local Function Dialog

Edit

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

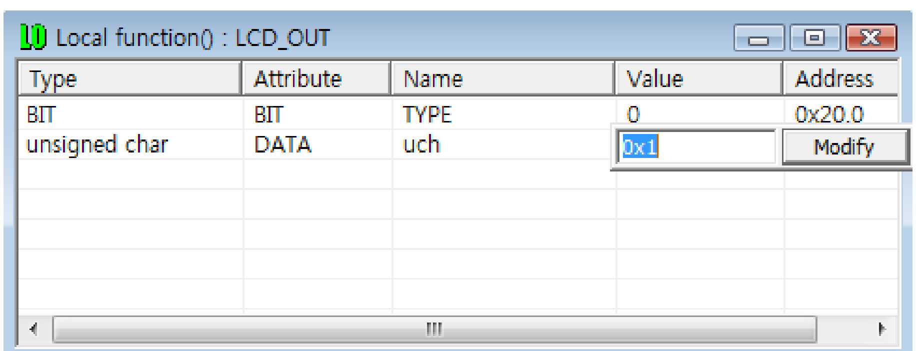


Figure 60. Editing A Local Symbol

Add or Remove Symbol

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

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

Figure 61 shows an example C source program module. The current program counter is located in the `delay(UINT uCnt)` function module (highlighted in the upper half of the figure), and the Local Variable window displays the name of the module and its variable (highlighted in the lower half of the figure).

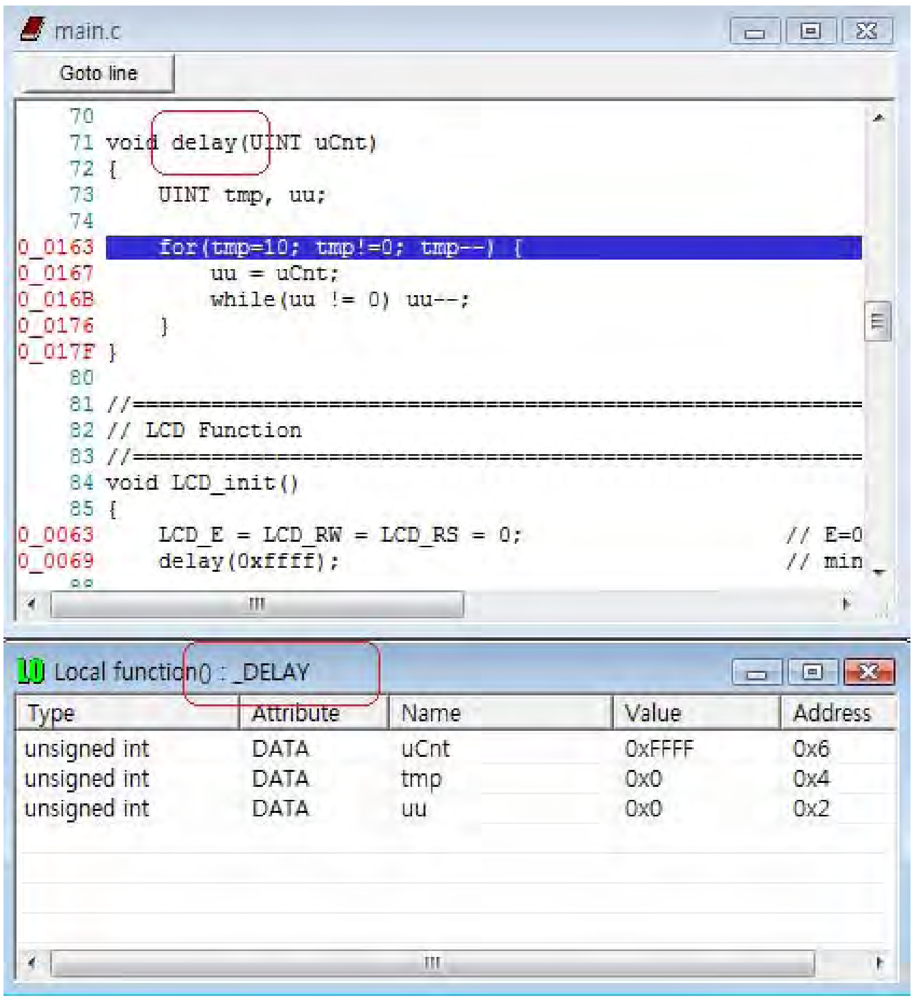


Figure 61. Example Watch Local Function, #1 of 2

If program flow is changed to another module, then the Local Variable list will be changed, as shown in Figure 62.

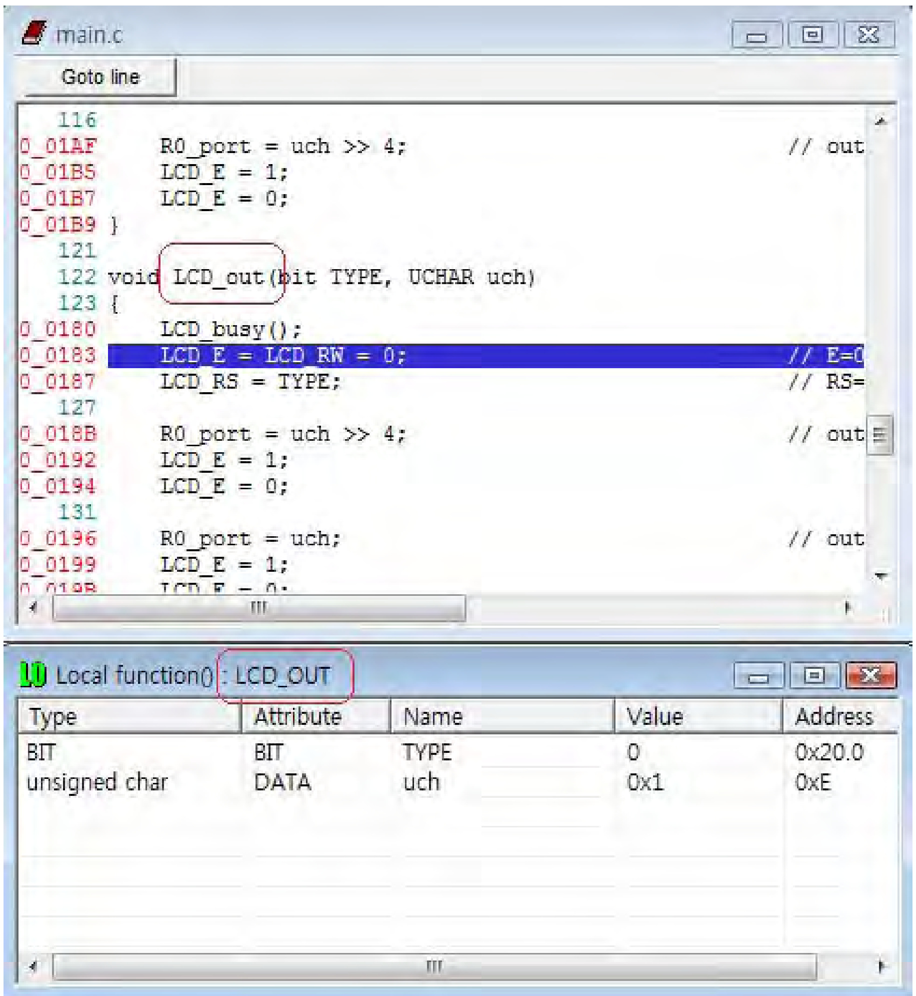


Figure 62. Example Watch Local Function, #2 of 2

Text File Window

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

Example 1. If a map/symbol file has not been loaded, or if the file does not include symbol information, only the line number is displayed, as highlighted in Figure 63.

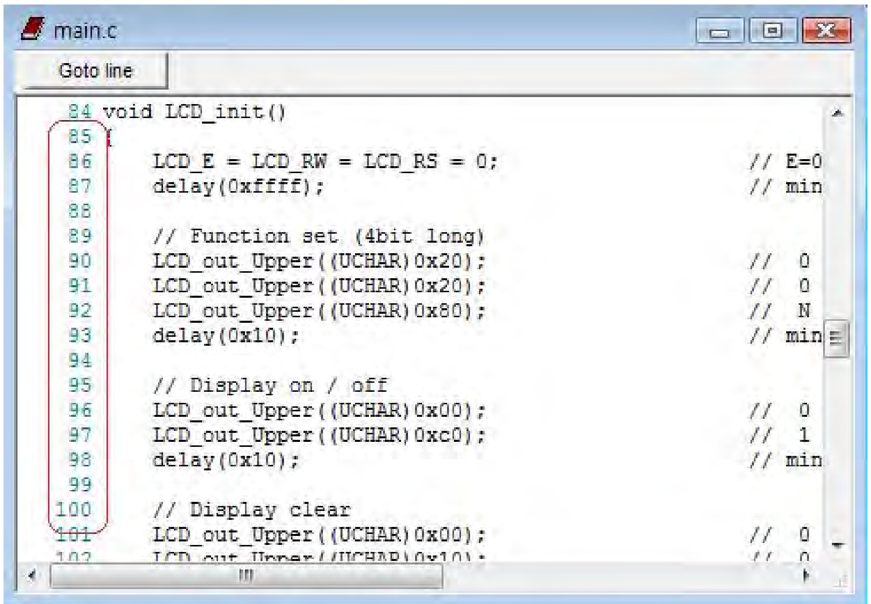


Figure 63. Using the Text File Function, #1 of 5

Example 2. If a map/symbol file has been loaded and the file includes symbol information, then the line number and address are displayed, as highlighted in Figure 64.

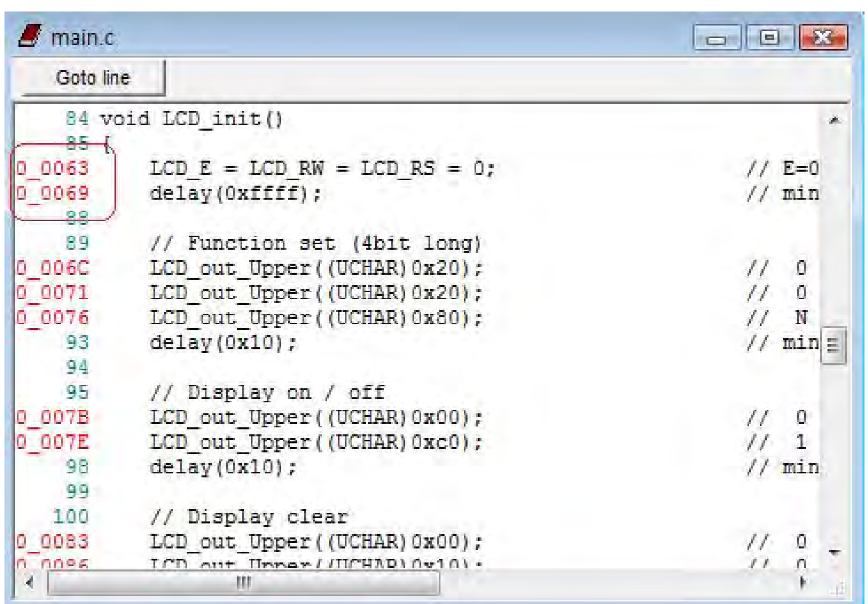


Figure 64. Using the Text File Function, #2 of 5

Goto Line

Clicking the **Goto Line** button displays the Get Decimal Number dialog box, which allows users to jump to another line in a text file; see Figure 65. Map/symbol information is not required.

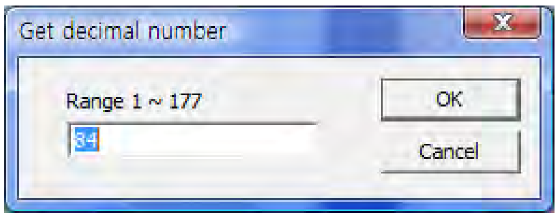


Figure 65. Using the Text File Function, #3 of 5

Disassemble Window Linkage

If a map/symbol file has been loaded and a text file is displayed, the text file will show addresses instead of line numbers. In the *[name]* dialog, and with your mouse, set your cursor in an address area (the left-most column) and double-click the left button to launch the Code Disassemble dialog, which will highlight the address; see Figure 66.

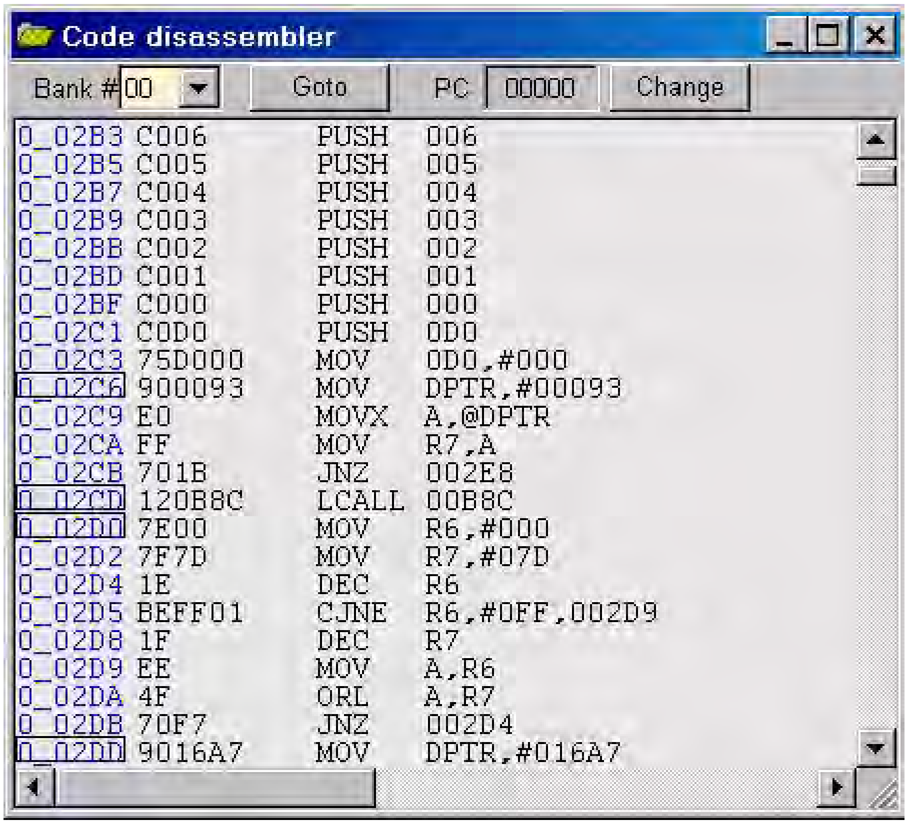


Figure 66. Code Disassemble Dialog

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

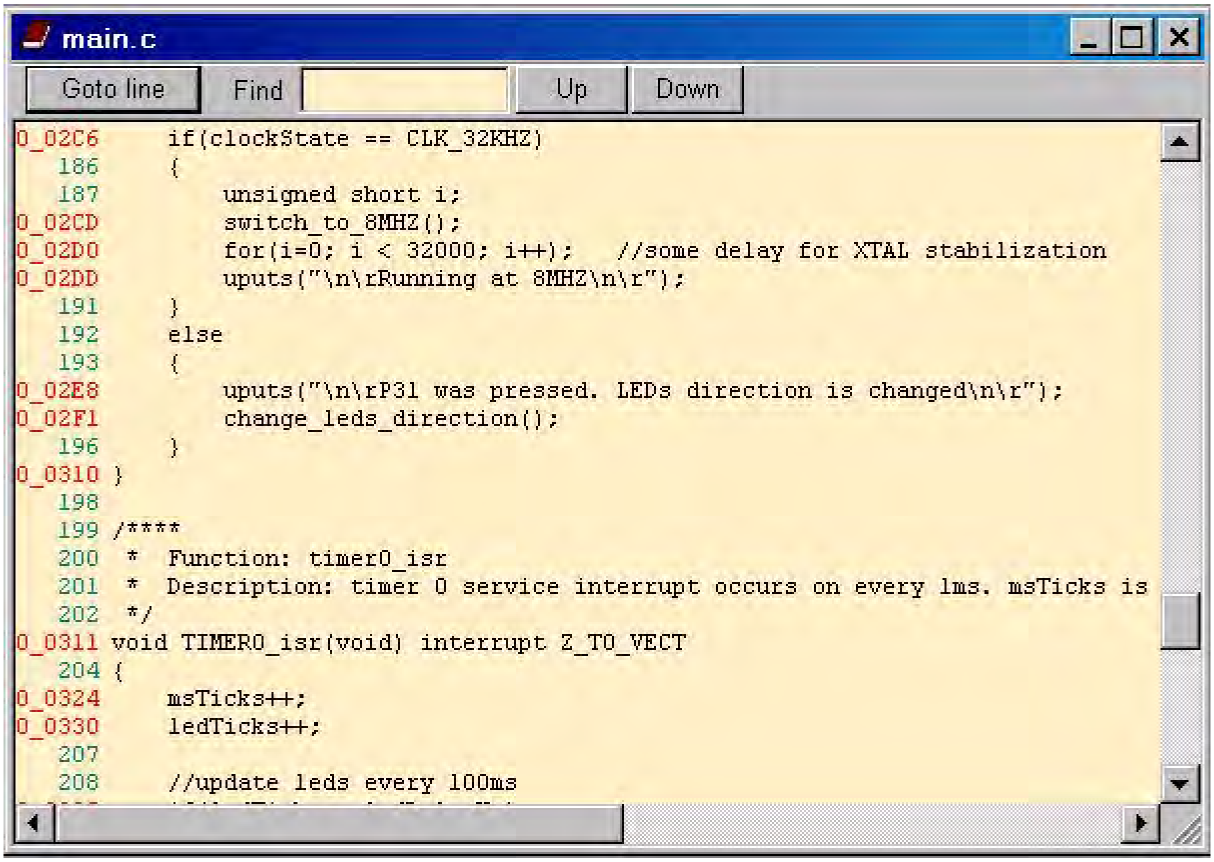


Figure 67. *[dialogX]*

Break Toggle

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

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

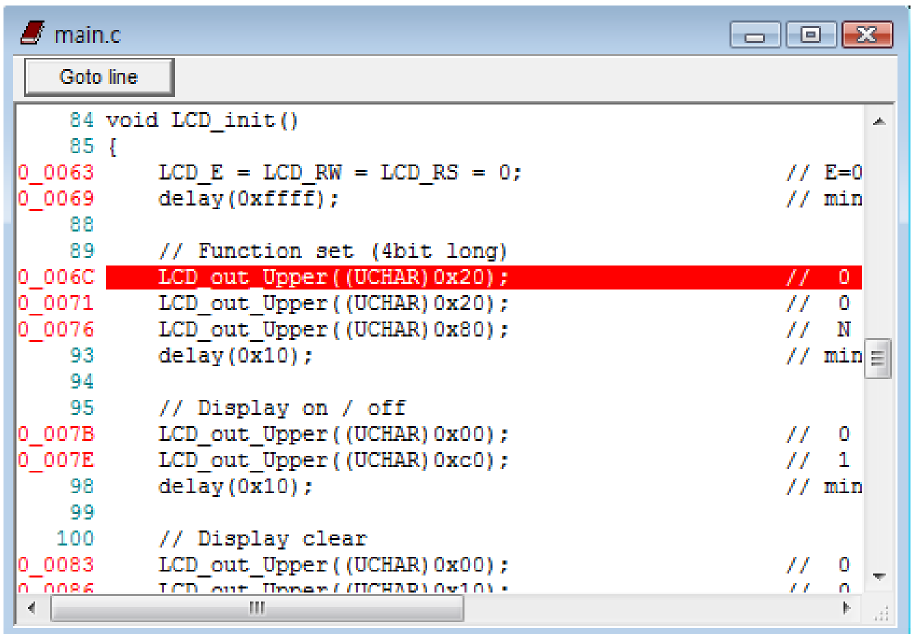


Figure 68. Using the Text File Function, #4 of 5

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

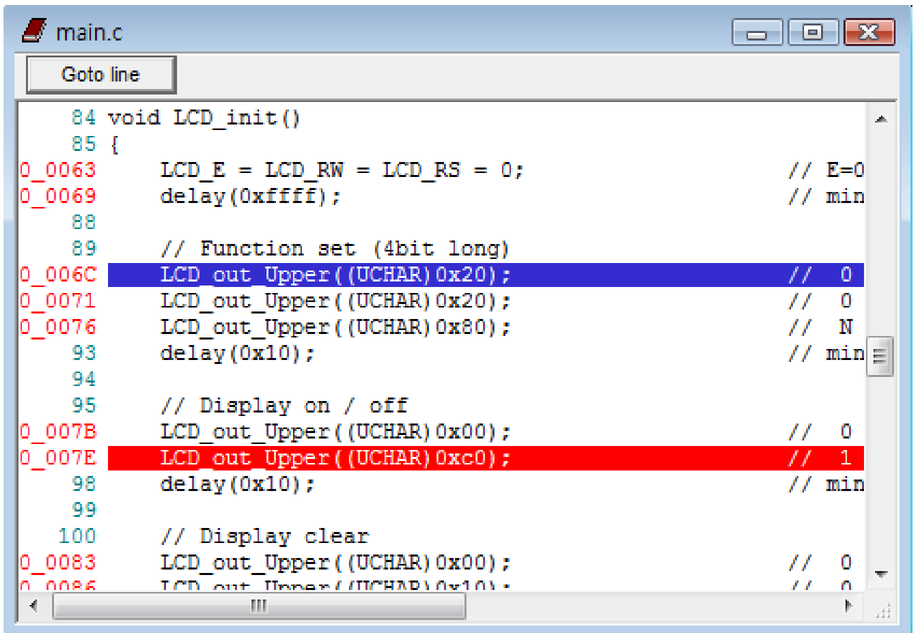


Figure 69. Using the Text File Function, #5 of 5

The Z8051 OCD In-System Programmer

The Z8051 On-Chip Debugger (OCD) In-System Programmer (ISP) has been developed to support Zilog's Z8051 8-bit MCUs. This document describes how to set up and use the Z8051 On-Chip Debugger ISP with your Z8051 Development Kit. The OCD ISP is used to program the Flash or EEPROM memory spaces of a target Z8051 MCU using Zilog's On-Chip Debugger. The OCD interface uses only two I/O lines[1] and does not require a socket adapter or specified power circuit.

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

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

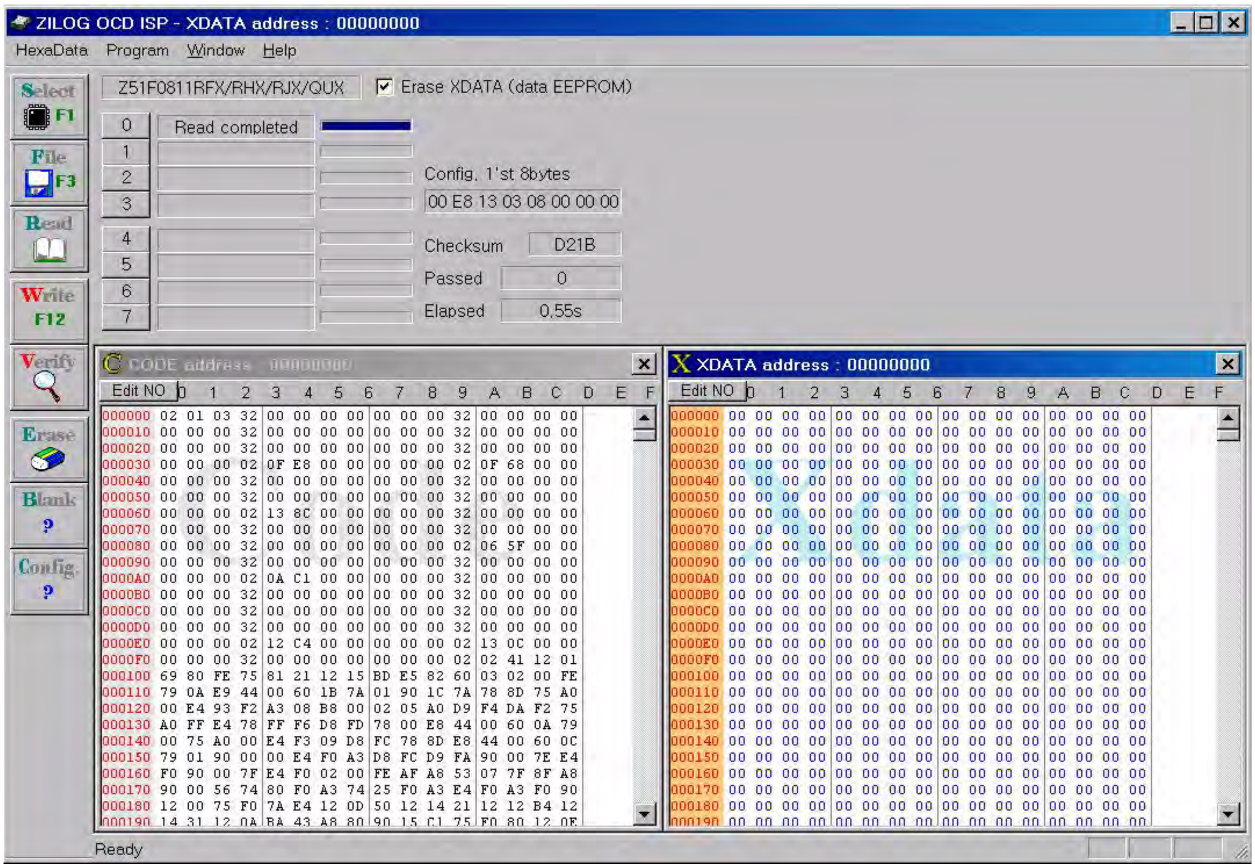


Figure 70. Example On-Chip Debugger ISP Screen

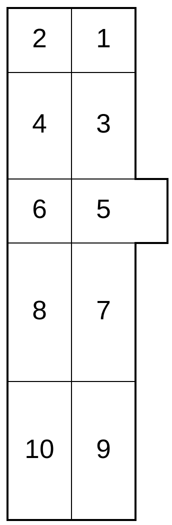
Features

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

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

Connect the Hardware

After installing the OCD software and drivers, hardware connections can be established. The pin positions of the Z8051 USB OCD interface are shown in Figure 71. Confirm the target device's pin positioning, and connect this interface to the USB port of your PC.



Pin No.	Function
2	V<Sub-scriptTa-ble>CC
4	GND
6	OCD S<Sub-scriptTa-ble>CLK
8	OCD S<Sub-scriptTa-ble>DATA

Figure 71. OCD Hardware ISP Pin Assignment (Bottom View)

Apply Power

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

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

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

2. Power on your PC.
3. Connect the OCD hardware to your PC.
4. Connect the OCD hardware to your target system.
5. Apply power to the target system.
6. Use the OCD In-System Programmer to perform your programming tasks.

7. When your programming work is complete, power off the target system.

Understanding the OCD ISP Menu Functions

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

HexData Menu

The HexData menu, shown in Figure 72, allows users to load their hexadecimal code to a target device for programming. Because each device operates on its own programming algorithm and features a different memory map, ISP functions are enabled only after a target device has been selected.

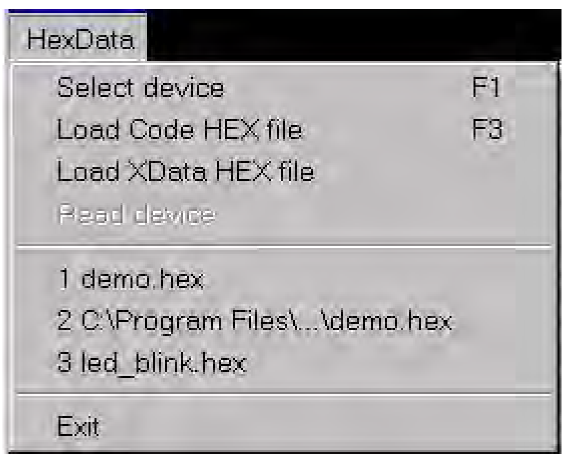


Figure 72. The OCD ISP's File Menu

Select Device

Observe the following procedure to select a target device.

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 73.

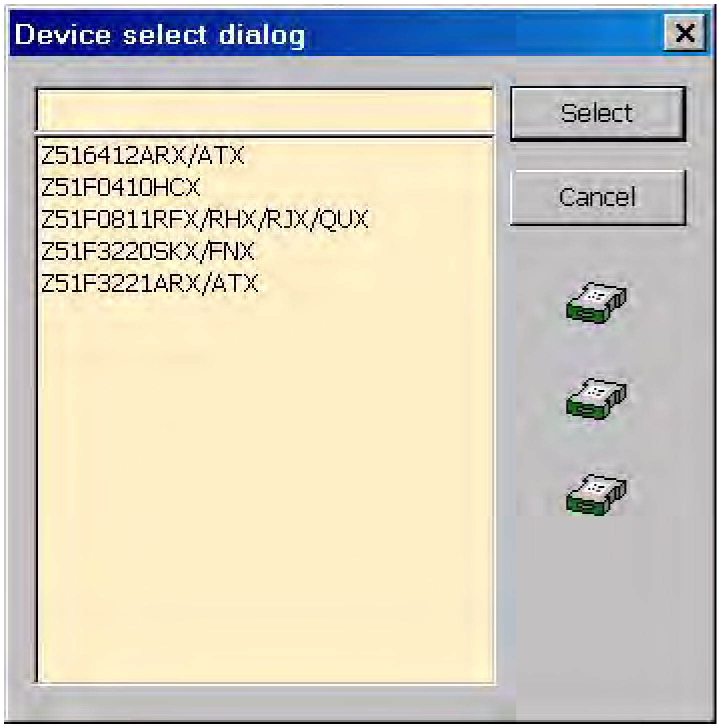


Figure 73. Device Select Dialog

Load Code HEX File

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

> **Note:** All hexadecimal files follow the Intel HEX format.

1. Select **Load Code HEX File** from the **HexData** menu to load a hexadecimal file from the host PC to a code buffer space generated by the In-System Programmer. The Fill Buffer dialog appears, as shown in Figure 74.

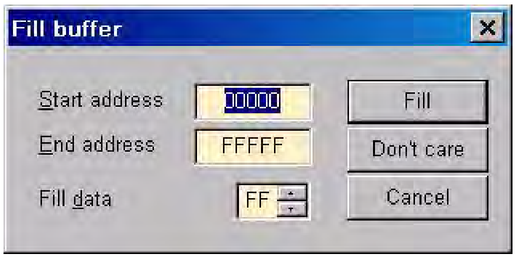


Figure 74. Fill Buffer Dialog

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

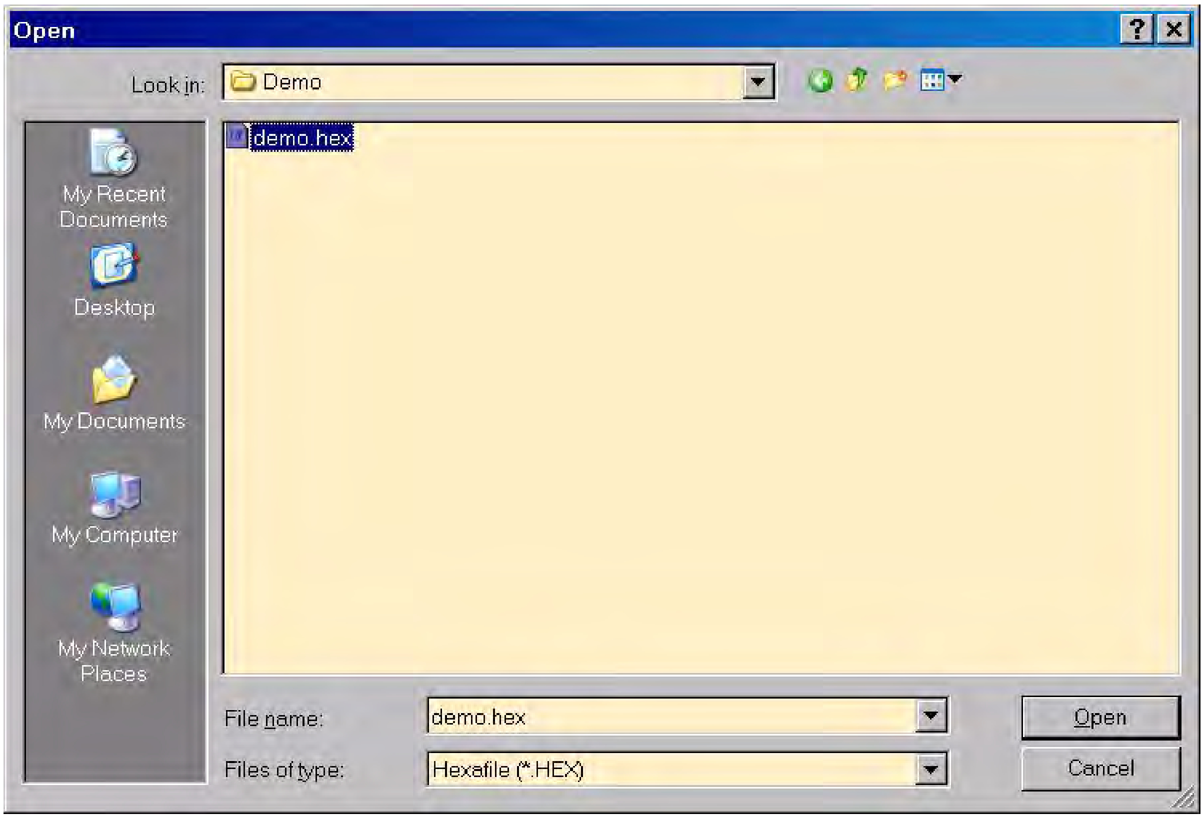


Figure 75. File Open Dialog

3. The OCD_ISP dialog box appears, as shown in Figure 76. After a hexadecimal file has been loaded, this dialog displays the name of the target device and a data checksum value, as highlighted in the figure.

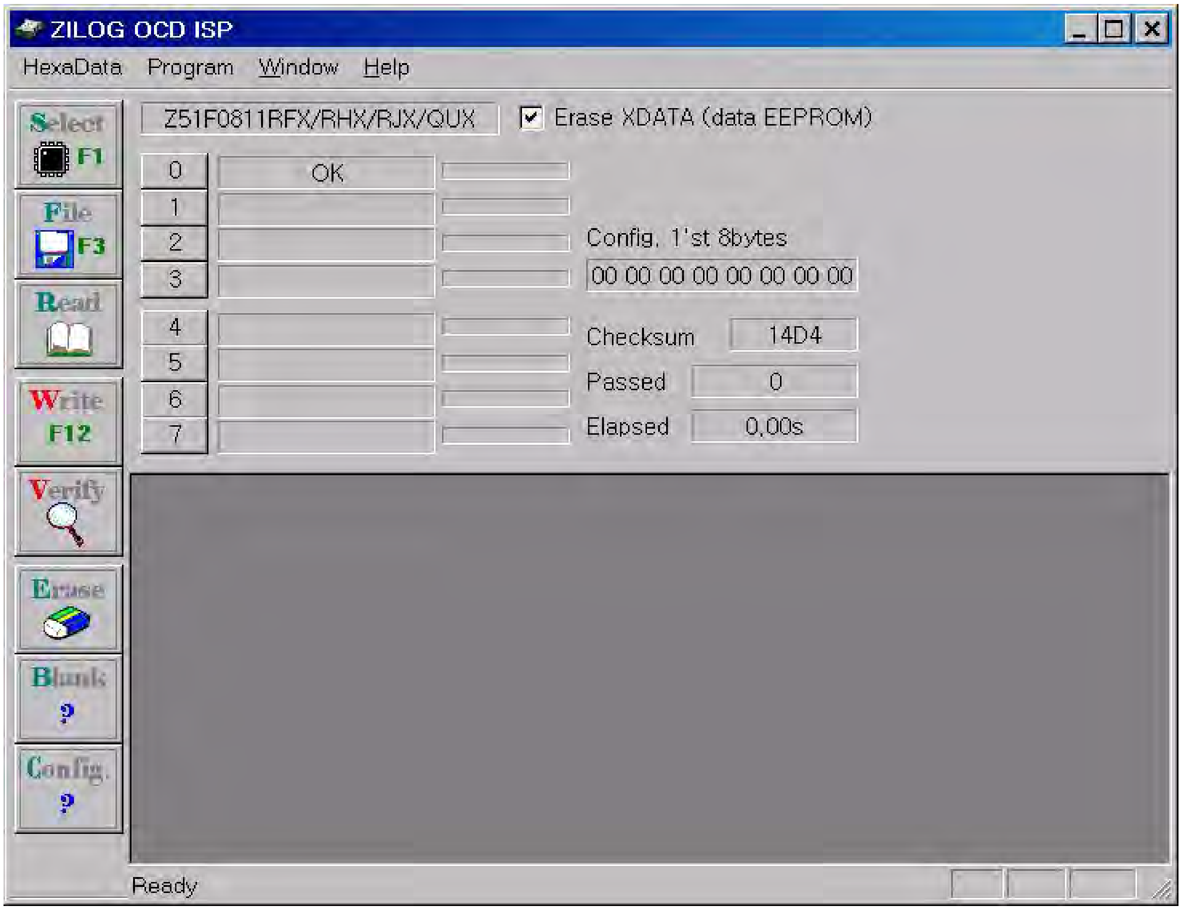


Figure 76. OCD ISP Dialog

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 77.

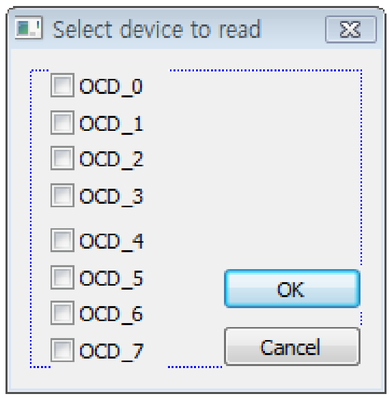


Figure 77. Select Device To Read Dialog

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

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

Most Recent Files

As the user opens and closes files, these files will appear in the Hex Data menu, and can be selected at any time in a current or future session. The maximum number of most recently-used files that will appear in the Hex Data menu is eight. See the example in Figure 78, which shows that the user has recently opened only three files; the third file (selected in the figure) is the demo.hex file.

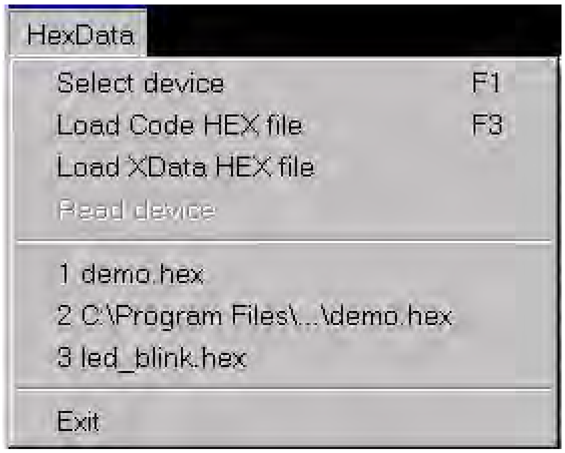


Figure 78. Most Recently Used Files

Exit

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

Program Menu

The Program menu, shown in Figure 79, lists all of the OCD ISP's programming main control functions, each of which is described in this section. This menu is enabled after the user selects a target device.

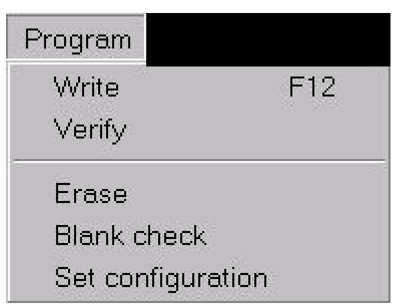


Figure 79. The OCD ISP's Program Menu

Write

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

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

Verify

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

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

Erase

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

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

Blank Check

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

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

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

Set Configuration

Because each device in the Z8051 Series is configured differently, use the Set Configuration function to configure the target device's I/O pin functions, oscillation method, code protection, etc.; see Figure 80 for an example.

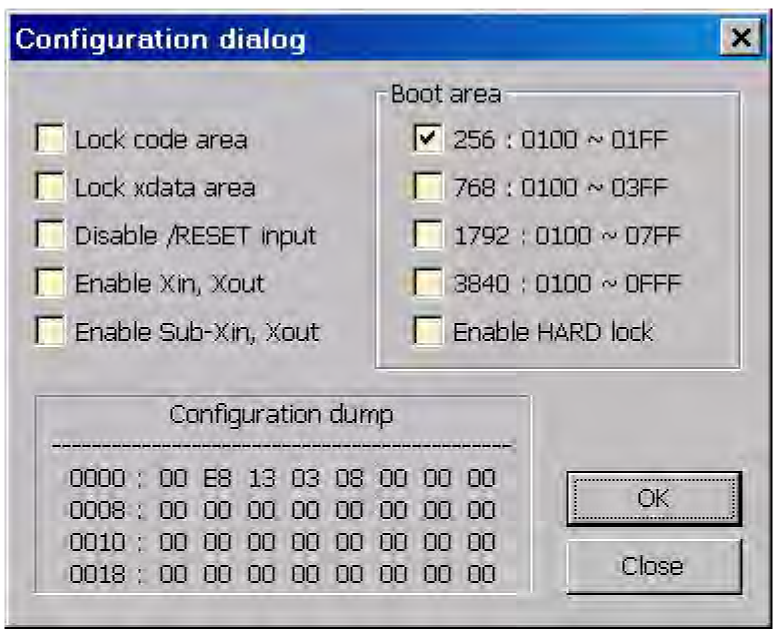


Figure 80. Configuration Dialog

Window Menu

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

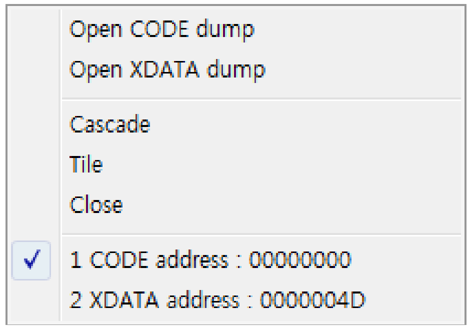


Figure 81. The OCD ISP's Window Menu

Open CODE Dump

Selecting **Open CODE Dump** from the **Window** menu opens a window which displays code memory in a dump format, as shown in Figure 82. If this window is already open, the window will move to the top-most level of the debugger frame.

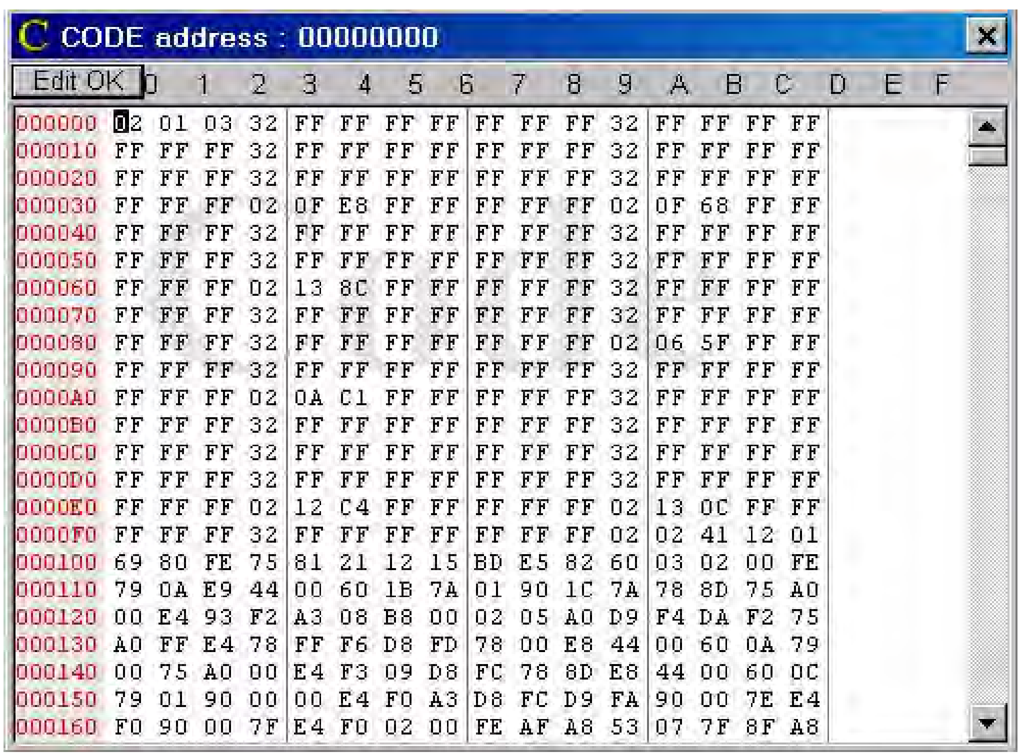


Figure 82. Open CODE Dump Child Window

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

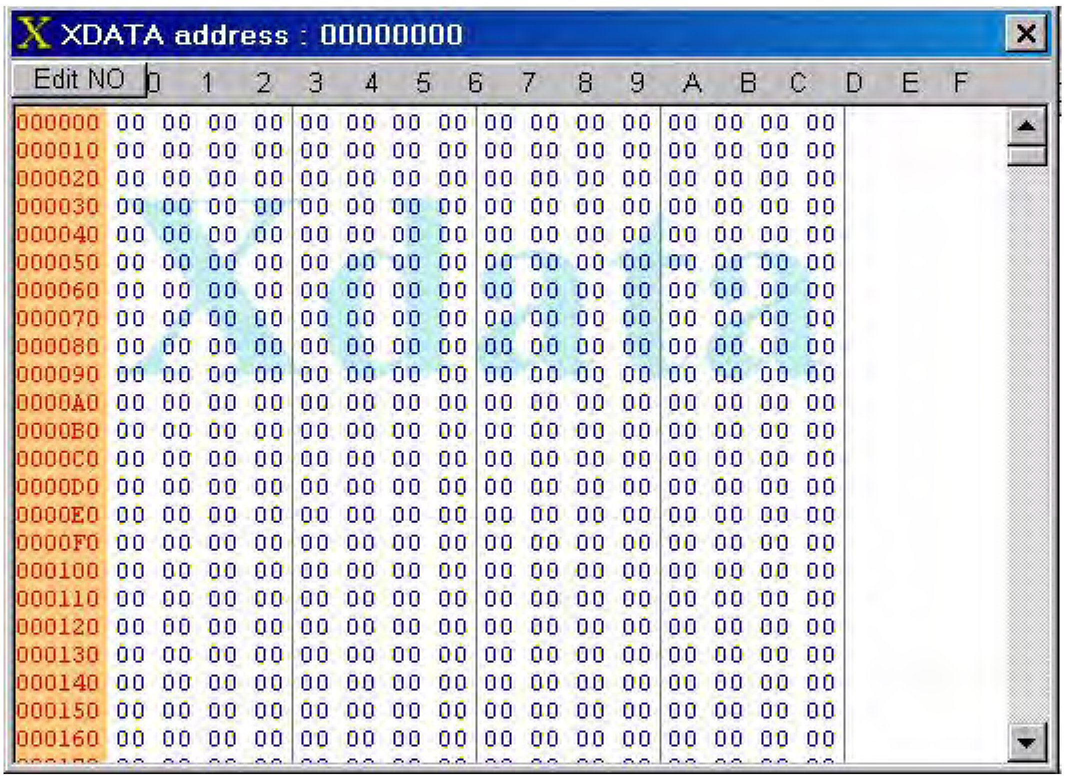


Figure 83. Open XData Dump Child Window

Cascade

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

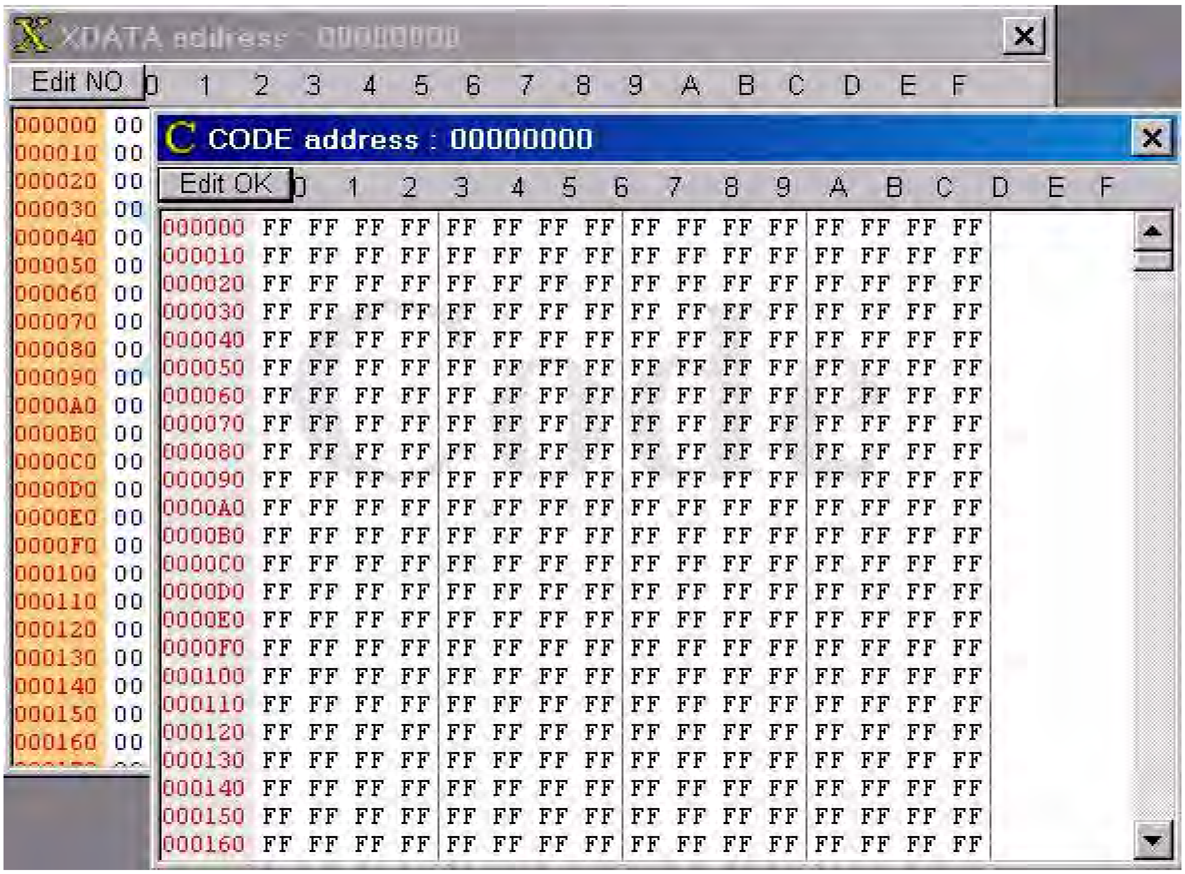


Figure 84. Cascading Child Windows

Tile

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

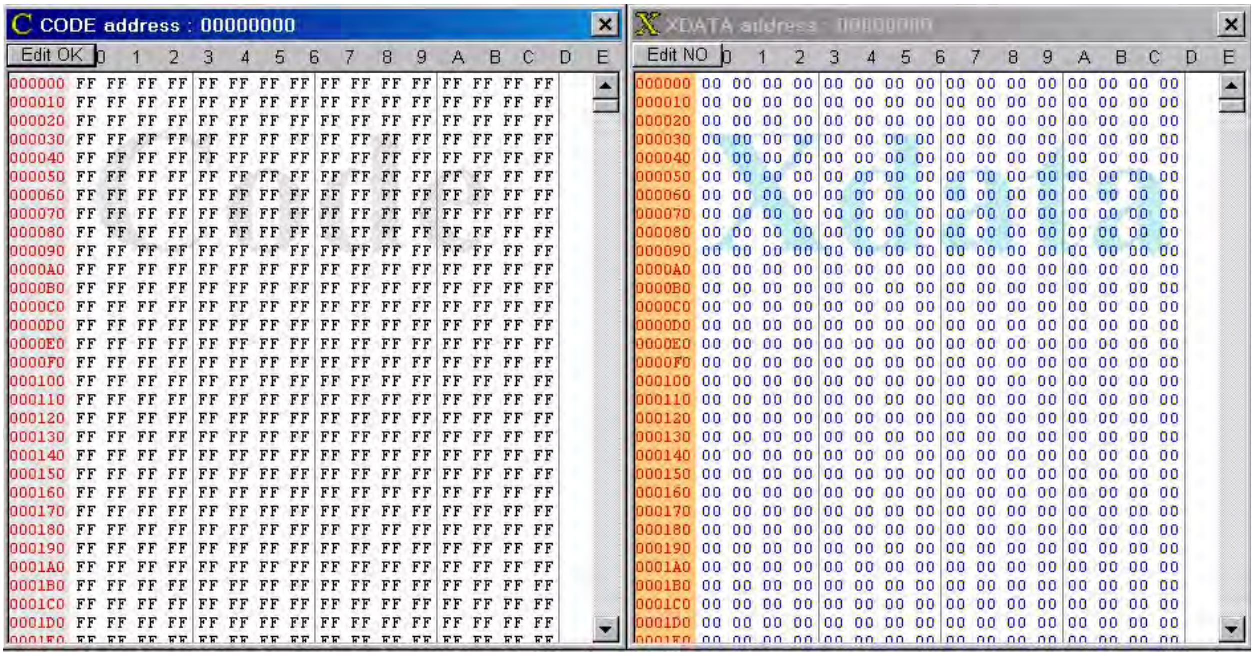


Figure 85. Tiled Child Windows

Close

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

Windows 1, 2, 3, Etc.

This menu selection assigns a sequential number (e.g., 1, 2, 3…) to each child window in the order in which it is opened. Users can directly select any open child window by its number. In Figure 81 on page 70, for example, selecting **2** from the **Window** menu will display the XData Dump window as the top-most window in the debugger screen.

Child Windows

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

Code Dump Window

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

The upper side of the Code Dump window displays the address of the current cursor position. The term Code is displayed as a watermark in the center of this window, as shown in Figure 86.

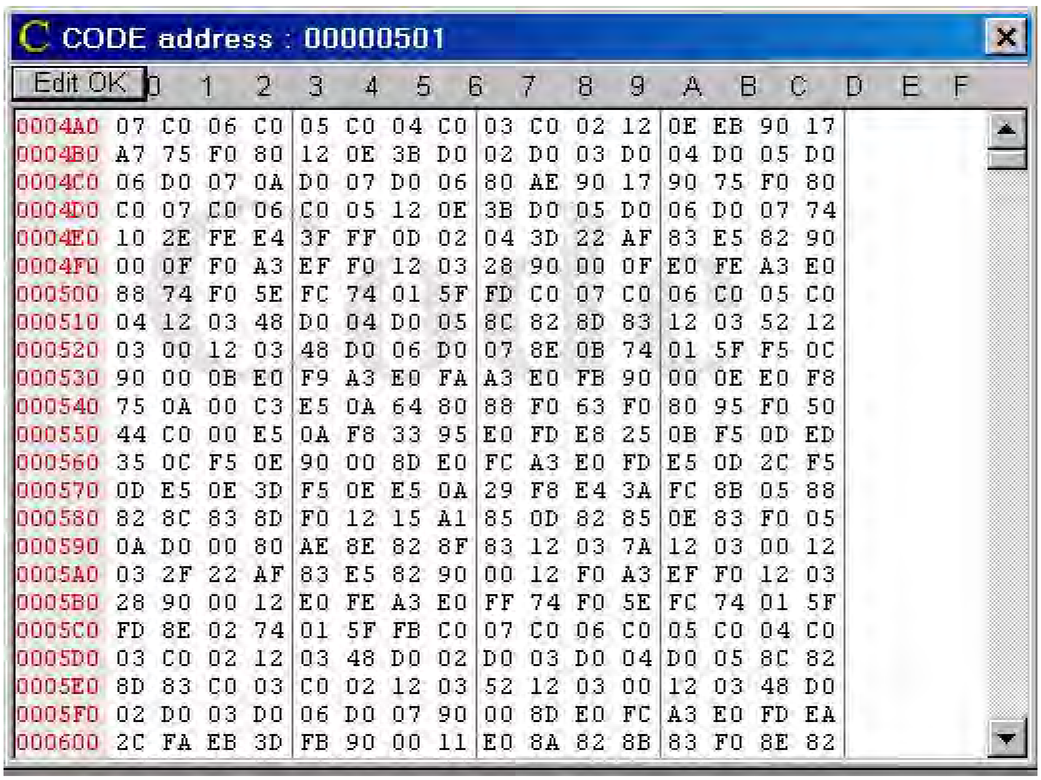


Figure 86. CODE Dump Child Window

Edit

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

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

Cursor Position

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

XData Dump Window

The XDATA Dump window displays each 8-bit segment of code memory in the hexadecimal format and supports the editing of this data. Each 256-byte page resides at an address in the range xx00–xxFFh, in which xx is the number of the page. The upper side of the XDATA Dump window displays the address of the current cursor position. The term XDATA is displayed as a watermark in the center of this window, as shown in Figure 87. Editing and cursor functions are the same for the XDATA Dump window as they are for the Code Dump window.

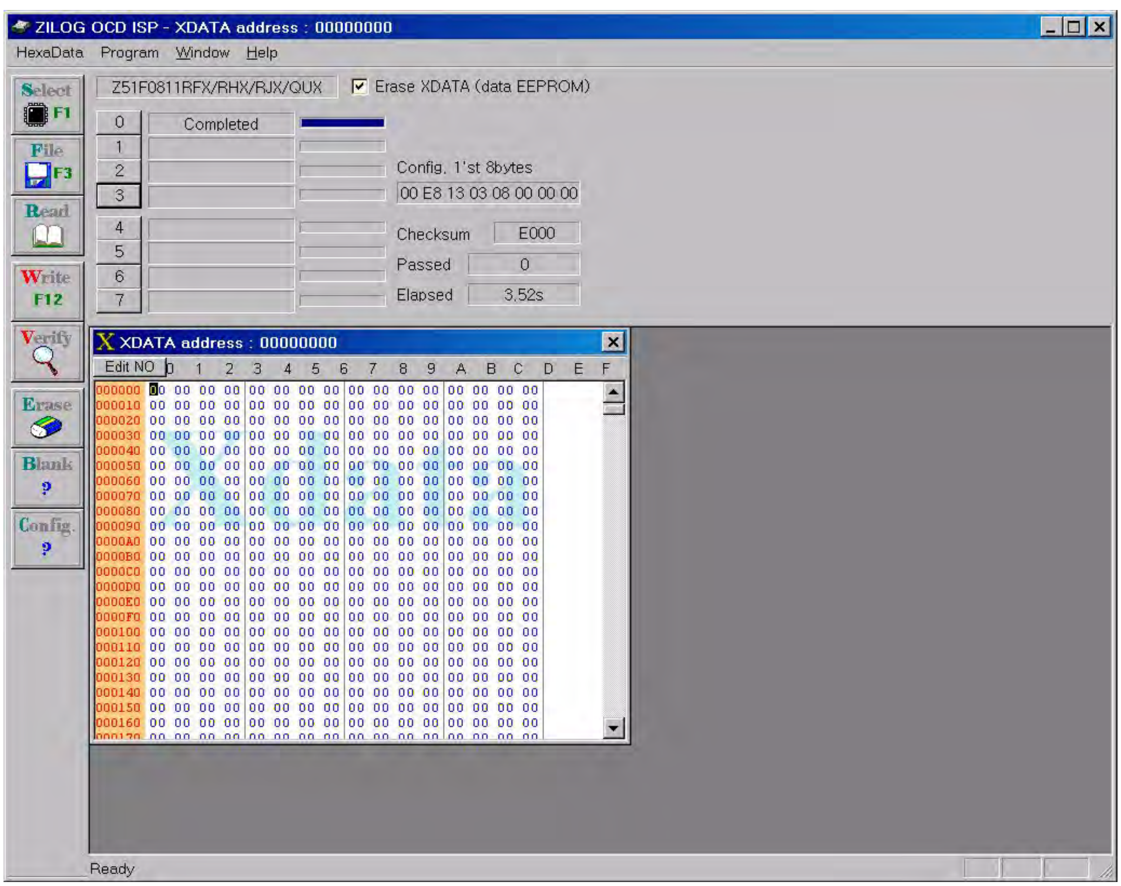


Figure 87. XData Dump Child Window

Appendix A. OCD Driver Installation on Windows Vista Systems

The driver programs for the Z8051 On-Chip Debugger are copied to the development PC during the software and documentation installation. In the following procedure for PCs running Windows Vista 32- and 64-bit operating systems, ensure that the target side of the OCD will remain unconnected while you install these drivers. Refer to Figure 2 on page 4 for guidance.

1. Connect the OCD hardware to the USB port of your PC by connecting the A (male) end of one of the two USB A (male)-to-Mini-B cables with the development PC's USB port. Connect the Mini-B end to the OCD device.
2. After the PC detects the new hardware, it will display the Found New Hardware Wizard dialog box, shown in Figure 88. Click **Locate and install driver software (recommended)**.

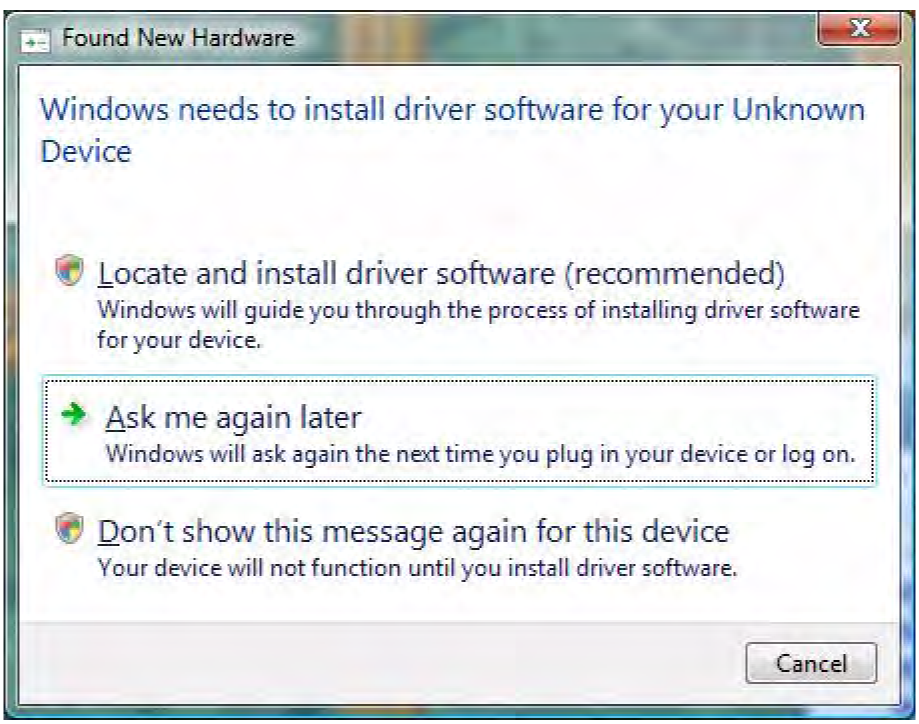


Figure 88. Found New Hardware Dialog, Windows Vista

3. Depending on your development PC's User Account Control settings, Windows may ask for permission to continue the installation. Click **Continue**.
4. When you see the Installing Device Driver dialog shown in Figure 89, do not click **Close**. Instead, wait until you see the dialog that follows, which is shown in Figure 90.

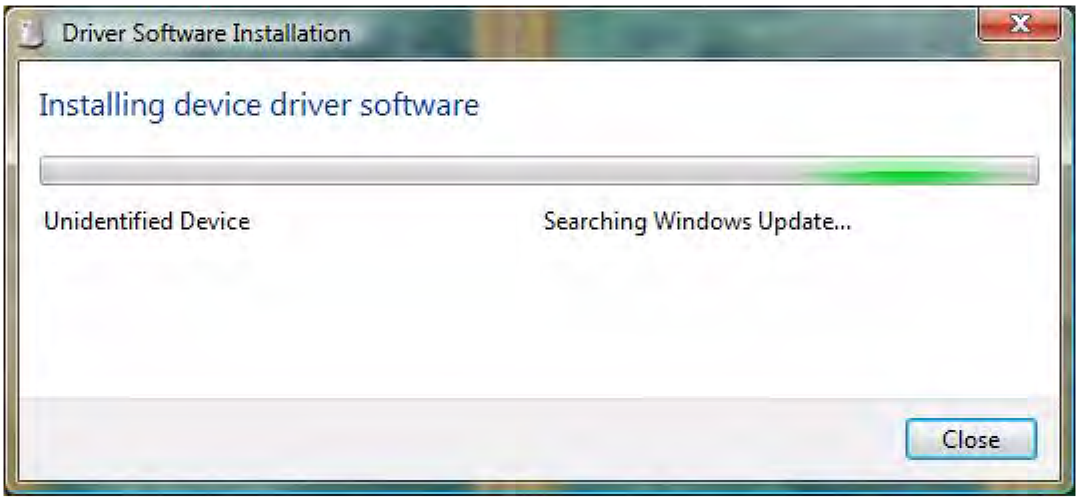


Figure 89. Install Device Driver Dialog, Windows Vista

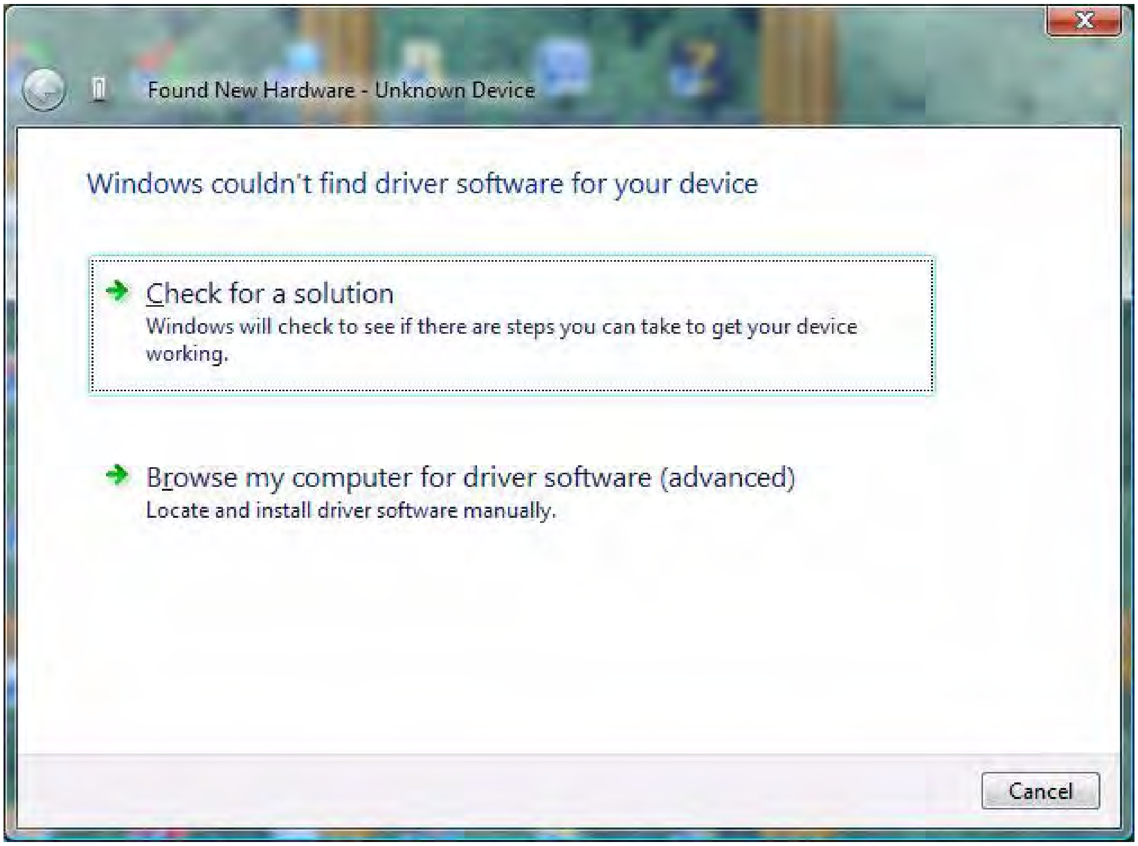


Figure 90. Couldn't Find Driver Dialog, Windows Vista

5. Select **Browse my computer for driver software (advanced)** to display the dialog shown in Figure 91, which prompts you to key in or browse for the location of the `.inf` file. Depending on the type of computer you use (32- bit or 64-bit), use the **Browse** button to navigate to one of the following paths, then click the **Next** button.
   - On 32-bit machines, use the following path:

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

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

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

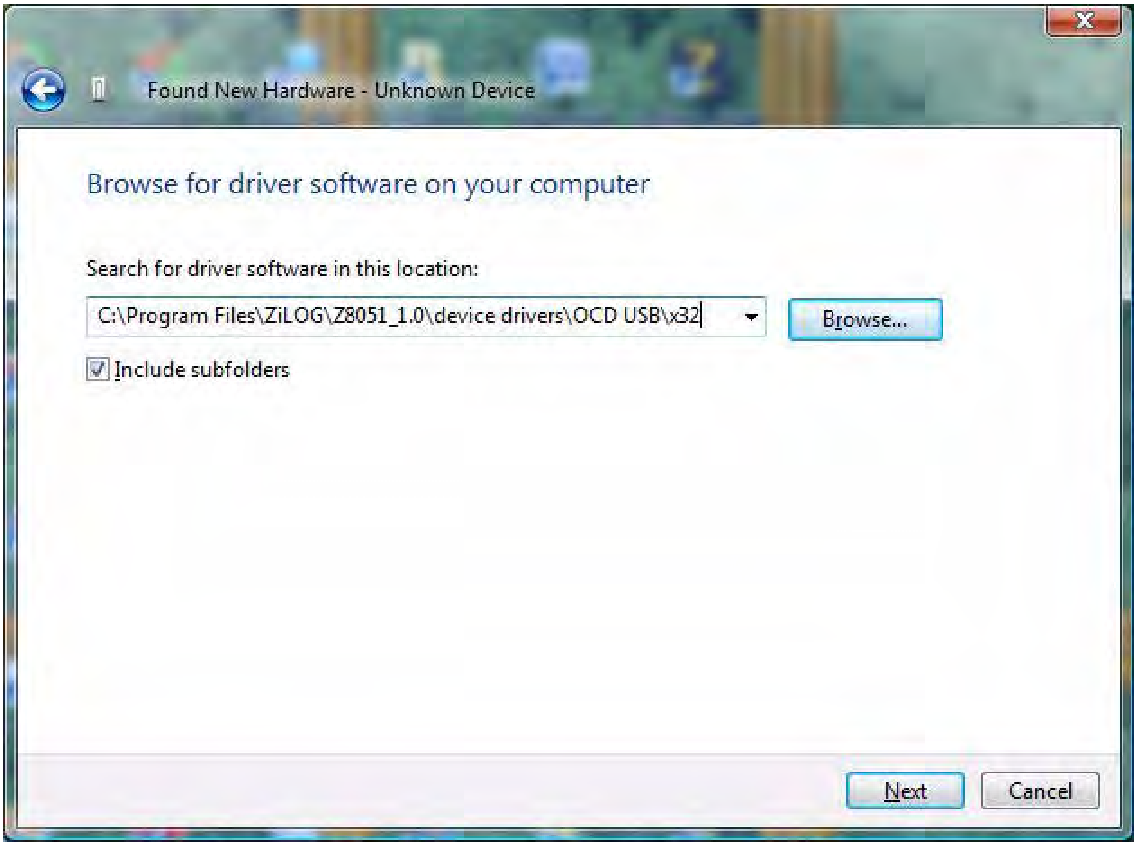


Figure 91. Browse For Driver Dialog, Windows Vista

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 92. Can't Verify Publisher Dialog, Windows Vista

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

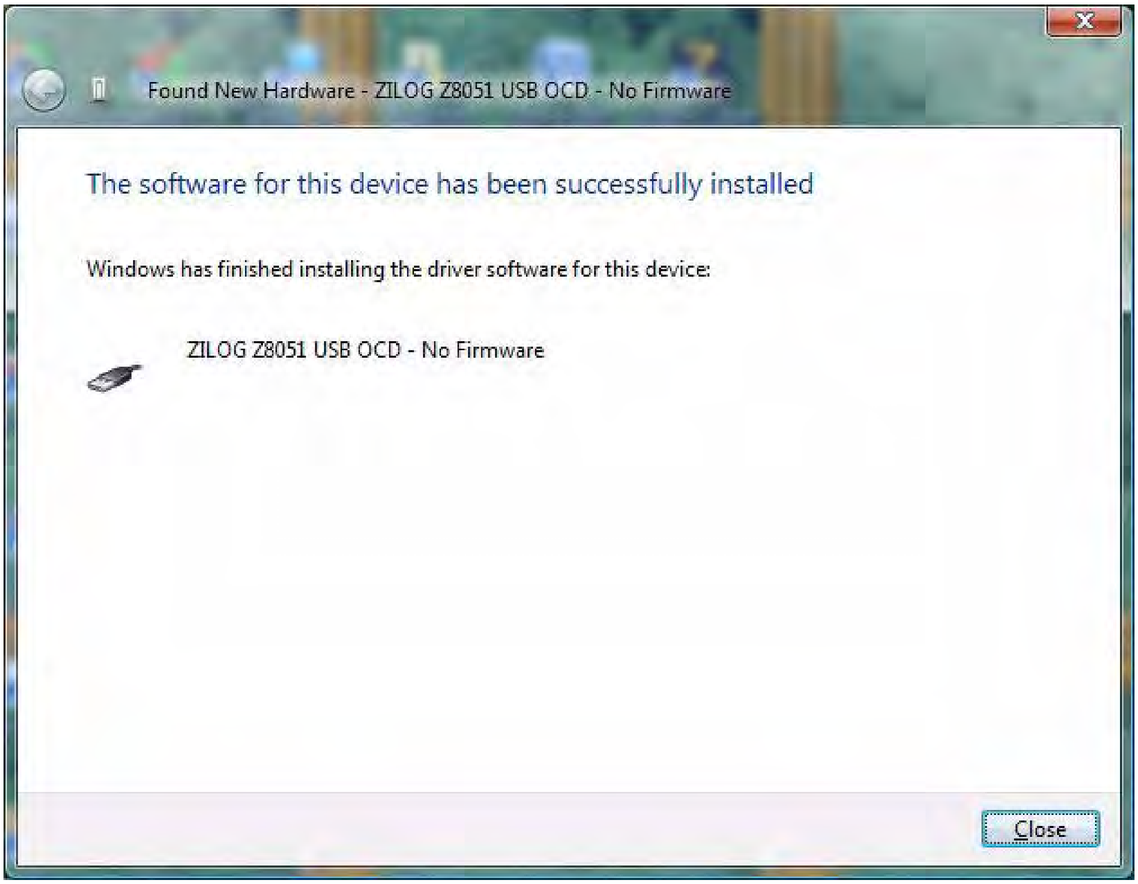


Figure 93. Successfully Installed Dialog, Windows Vista

Appendix B. OCD Driver Installation on Windows 7 Systems

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

1. Connect the OCD hardware to the USB port of your PC by connecting the A (male) end of one of the two USB A (male)-to-Mini-B cables with the host PC's USB port. Connect the Mini-B end to the OCD device.
2. After the PC detects the new hardware, it will display the *Installing device driver software* dialog shown in Figure 94. Click within this dialog to display the installation sequence of the driver software, which is diagrammed from top to bottom in Figure 95.

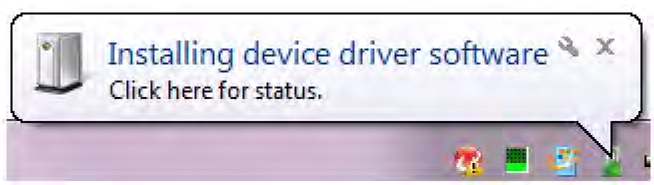


Figure 94. Install Device Driver Dialog, Windows 7

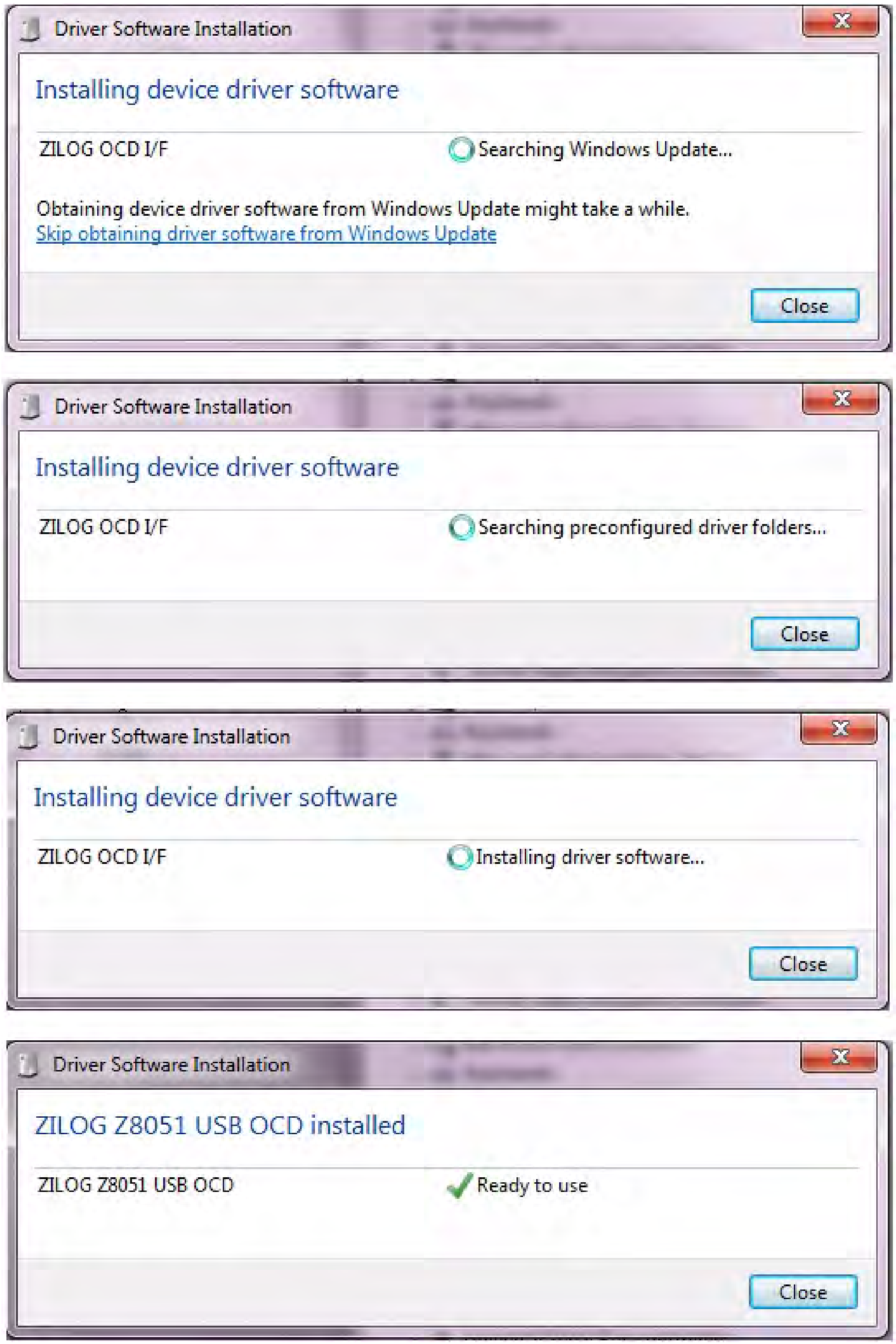


Figure 95. Driver Software Installation Dialog, Windows 7

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

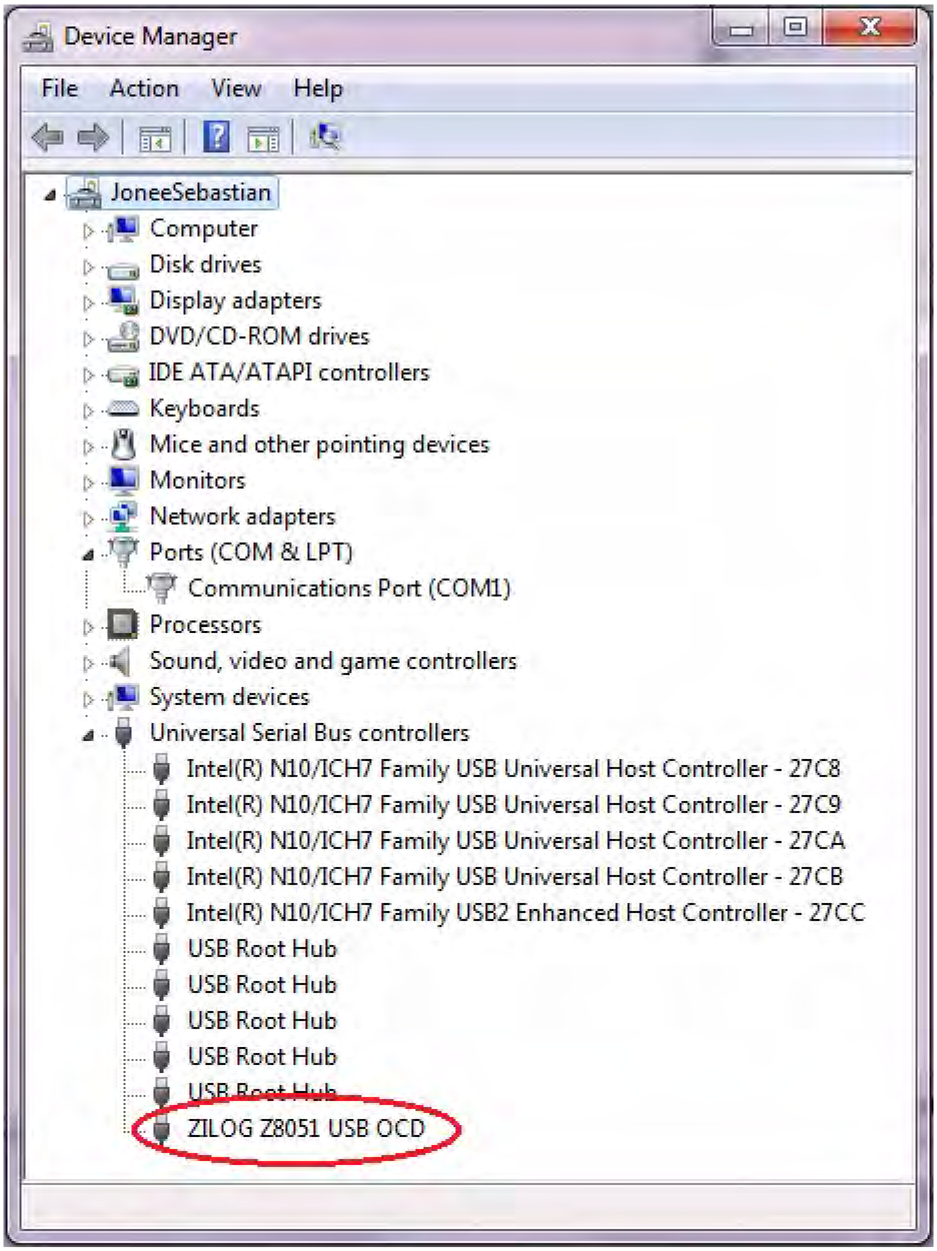


Figure 96. Device Manager Dialog, Windows 7

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at 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.