Megawin

8051 OCD ICE

User Manual

Contents

1 Introduction

Features

- Megawin proprietary OCD (On-Chip-Debug) technology
- On-chip & in-system real-time debugging
- Two-pin dedicated serial interface for OCD, no target resource occupied
- Directly linked to the debugger function of the *Keil 8051 IDE Software*
- USB connection between target and host (PC)
- Helpful debug actions: *Reset*, *Run*, *Stop*, *Step* and *Run to Cursor*
- Programmable breakpoints, up to 4 breakpoints can be inserted simultaneously
- Several debug-helpful windows: Register/Disassembly/Watch/Memory Windows
- Source-level (*Assembly* or *C-language*) debugging capability

Description

The all new "Megawin 8051 OCD ICE" is a powerful development tool for 8051 embedded system. By adopting the Megawin proprietary OCD (On-Chip-Debug) technology, this ICE provides on-chip and in-system real-time debugging. The user has no need to prepare any development board during developing, or the socket adapter used in the traditional ICE probe. All the thing the user needs to do is to reserve a 6-pin connector for the dedicated OCD interface: *VCC*, *OCD_SDA*, *OCD_SCL*, RST,CLK and *GND*.

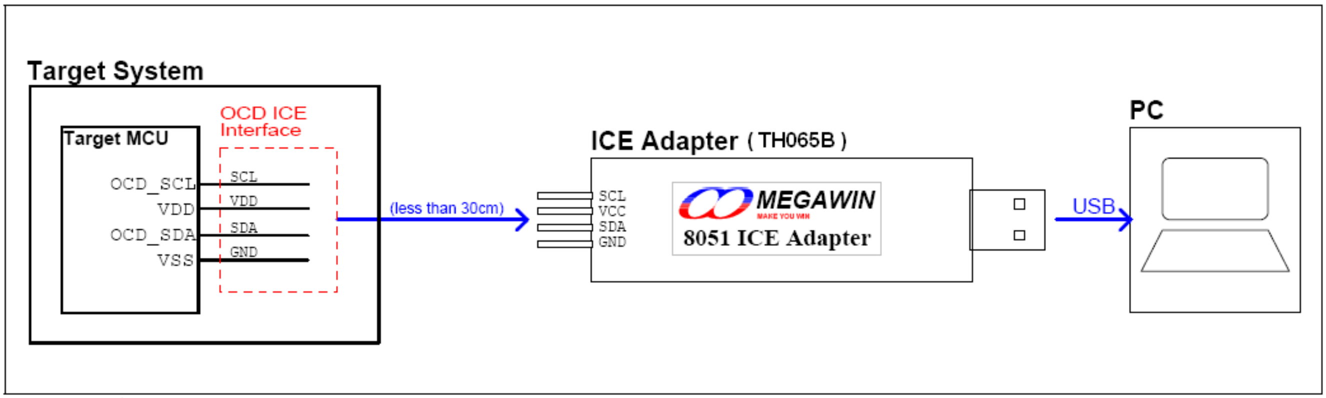
In addition, the most useful feature is that it can directly connect the user's target system to the *Keil 8051 IDE software* for debugging, which directly utilizes the Keil IDE's *dScope*-Debugger function. Of course, all the advantages are based on your using *Keil 8051 IDE software*.

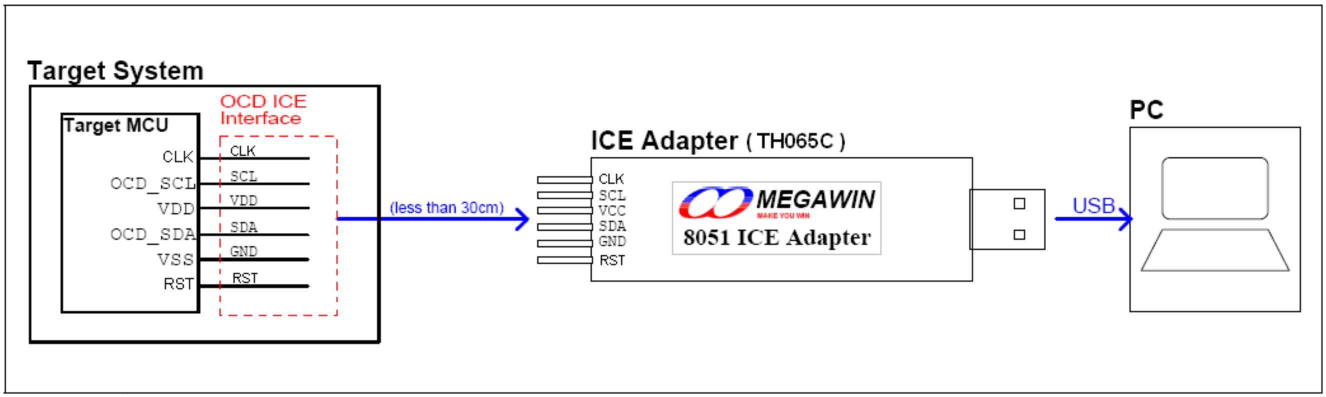
Note:
"Keil" is the trade mark of "Keil Elektronik GmbH and Keil Software, Inc.", and "Keil 8051 IDE software" is the most popular C51 compiler for 8051 embedded system development.

2 Hardware Setup

For debugging, the user should connect the target system to a PC via the ICE adapter, as shown below. The ICE adapter is a bus-powered USB device, and therefore there is no need of a power adapter for it.

Hardware Connection Diagram





Note: Refer to Section 6.5 for more information.

OCD ICE Interface Pin Number

Part No.	Package	OCD_SCL	OCD_SDA	RST	CLK
MPC82G516	40-pin DIP	29	30	N/A	N/A
	44-pin PLCC	32	33	N/A	N/A
	44-pin QFP	26	27	N/A	N/A
MG82FL(E)532/564	44-pin QFP	26	29	4	5
	48-pin LQFP	28	32	5	6
MG84FG516	48-pin LQFP	26	27	25	N/A
	64-pin LQFP	34	35	33	N/A
MG82FG5A32/5A64	48-pin LQFP	26	27	25	N/A
	64-pin LQFP	34	35	33	N/A
MG82FG5Bxx	28-pin SOP	27	28	26	N/A
	32-pin LQFP	18	19	17	N/A
MG82FG5Cxx	48-pin LQFP	26	27	25	N/A

	64-pin LQFP	**34**	**35**	**33**	**N/A**
MG82FG5Dxx	16-pin SOP	**15**	**16**	**14**	**N/A**
	20-pin SSOP	**19**	**20**	**18**	**N/A**
MG82G5E32	48-pin LQFP	**26**	**27**	**25**	**N/A**

****N/A : No need to connect**

3 Software Setup

This section tell the user how to do software setup before using the OCD ICE.

3.1 Install the USB Device Driver for the ICE Adapter

The user just needs to plug the ICE adapter into any USB port in a PC. There is no need to install any device driver for the ICE adapter.

3.2 Install the Megawin 8051 Database in the Keil 8051 IDE Software

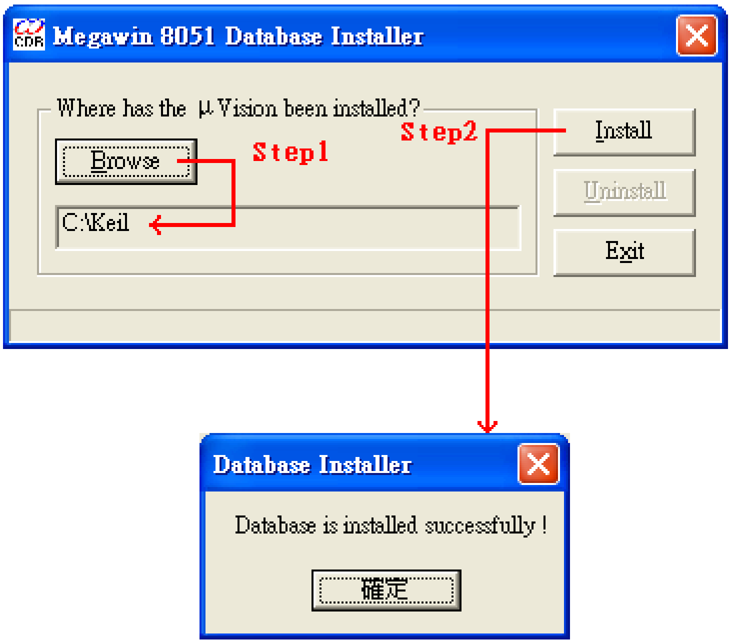
Activate the *"Setup.exe"* in the folder [Database Installer] to open the *Database Installer Application Program* to install the Megawin Database into the Keil 8051 IDE software. Of course, you should have installed the Keil 8051 IDE software, either µVision2 or µVision3, in your PC previously.

After opening the *Database Installer*, please follow the steps shown in the following GUI figure.

Step1) Click the **Browse** button to specify where the Keil software has been installed.
(Normally, when you install the Keil 8051 IDE software, the default install-path is "C:\KEIL".)

Step2) Click the **Install** button to start installing the Megawin Database into the Keil software.

GUI of the Database Installer



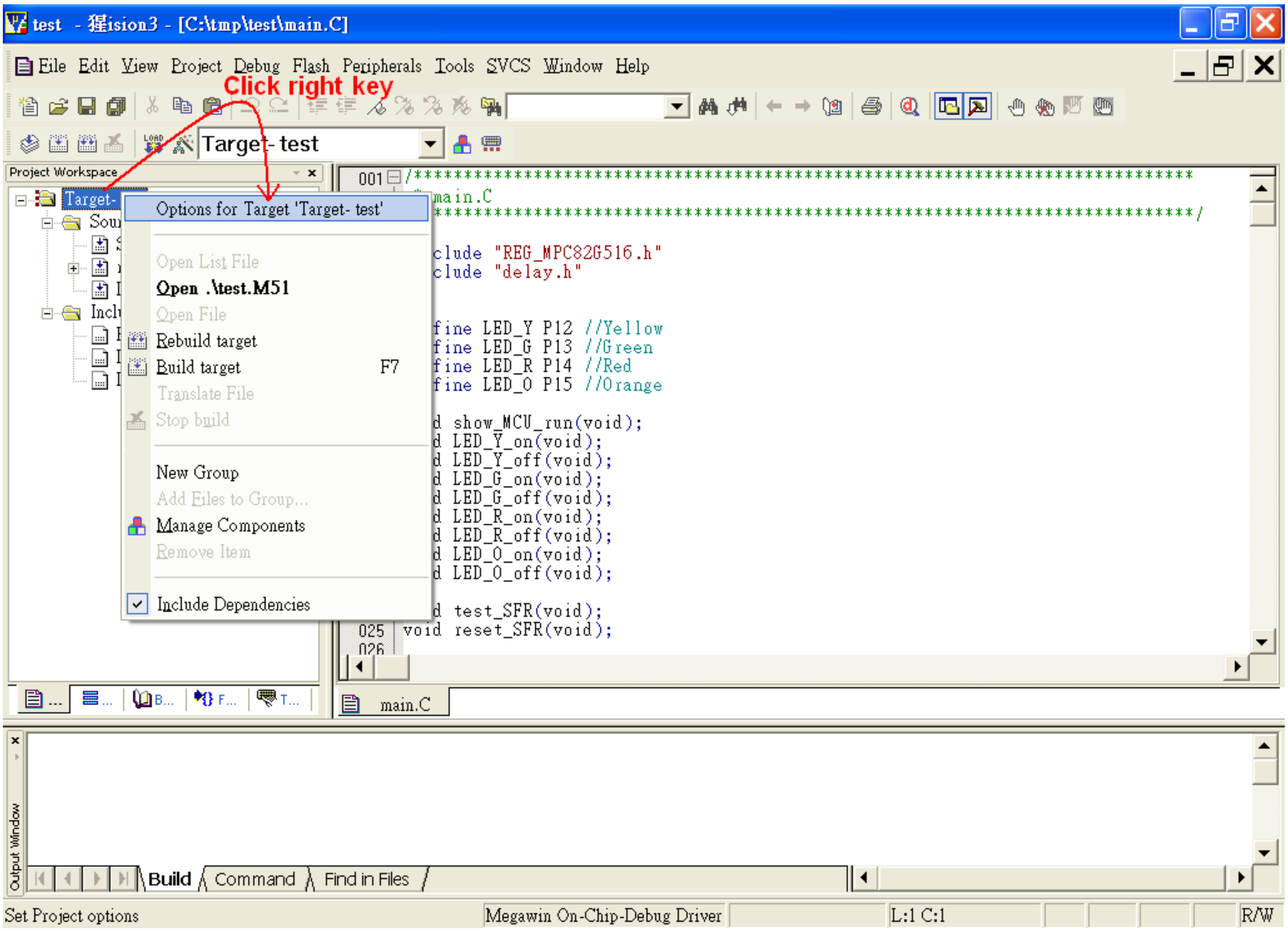
Where has the µ Vision been installed?

C:\Keil

Database is installed successfully !

4 Keil IDE Setup

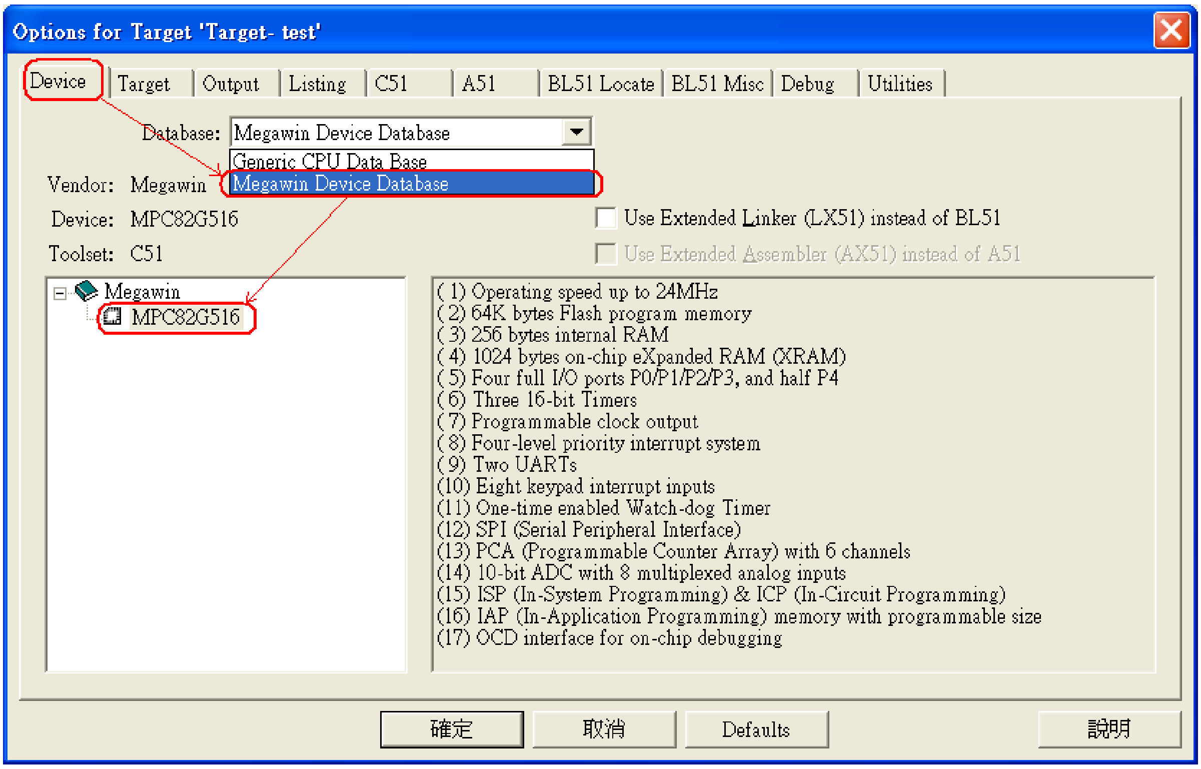
Before using the *dScope-D*ebugger function of the Keil IDE, the user should do some proper settings in the Keil IDE. First, open the µVision project you would like to debug. Then, move cursor to “Target-..” and click the mouse’s right button to invoke the “**Options for Target**”, as shown below.



```
/*****************************************************************************
 main.C
*****************************************************************************/
clude "REG_MPC82G516.h"
clude "delay.h"

fine LED_Y P12 //Yellow
fine LED_G P13 //Green
fine LED_R P14 //Red
fine LED_O P15 //Orange

d show_MCU_run(void);
d LED_Y_on(void);
d LED_Y_off(void);
d LED_G_on(void);
d LED_G_off(void);
d LED_R_on(void);
d LED_R_off(void);
d LED_O_on(void);
d LED_O_off(void);

d test_SFR(void);
void reset_SFR(void);
```

4.1 Options- Device

Select the “Megawin Device Database” and the target part number.



Options for Target 'Target- test'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Database: Megawin Device Database
Generic CPU Data Base
Megawin Device Database

Vendor: Megawin
Device: MPC82G516
Toolset: C51

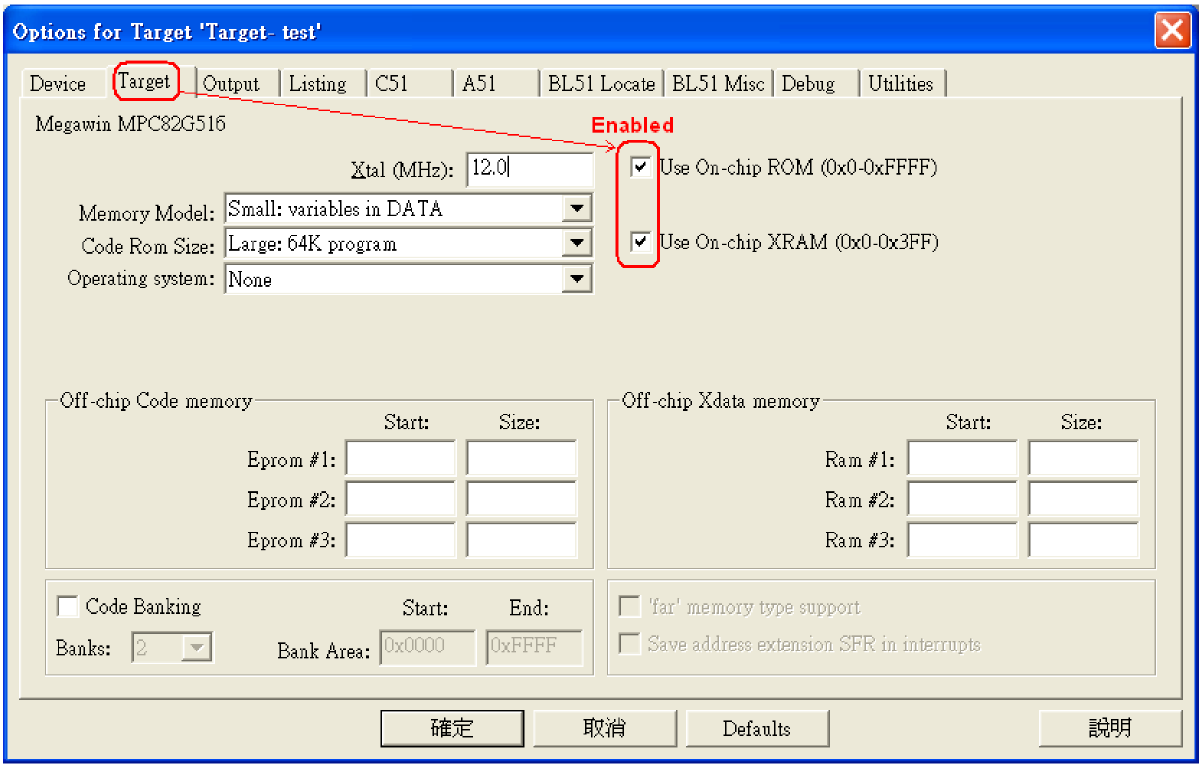
□ Use Extended Linker (LX51) instead of BL51
□ Use Extended Assembler (AX51) instead of A51

Megawin
MPC82G516

(1) Operating speed up to 24MHz
(2) 64K bytes Flash program memory
(3) 256 bytes internal RAM
(4) 1024 bytes on-chip eXpanded RAM (XRAM)
(5) Four full I/O ports P0/P1/P2/P3, and half P4
(6) Three 16-bit Timers
(7) Programmable clock output
(8) Four-level priority interrupt system
(9) Two UARTs
(10) Eight keypad interrupt inputs
(11) One-time enabled Watch-dog Timer
(12) SPI (Serial Peripheral Interface)
(13) PCA (Programmable Counter Array) with 6 channels
(14) 10-bit ADC with 8 multiplexed analog inputs
(15) ISP (In-System Programming) & ICP (In-Circuit Programming)
(16) IAP (In-Application Programming) memory with programmable size
(17) OCD interface for on-chip debugging

4.2 Options- Target

Enable the “Use on-chip ROM” and the “Use on-chip XRAM”.



Options for Target 'Target- test'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities
Enabled

Megawin MPC82G516

Xtal (MHz): 12.0
☑ Use On-chip ROM (0x0-0xFFFF)
Memory Model: Small: variables in DATA
Code Rom Size: Large: 64K program
☑ Use On-chip XRAM (0x0-0x3FF)
Operating system: None

Off-chip Code memory

	Start:	Size:
Eprom #1:		
Eprom #2:		
Eprom #3:		

Off-chip Xdata memory

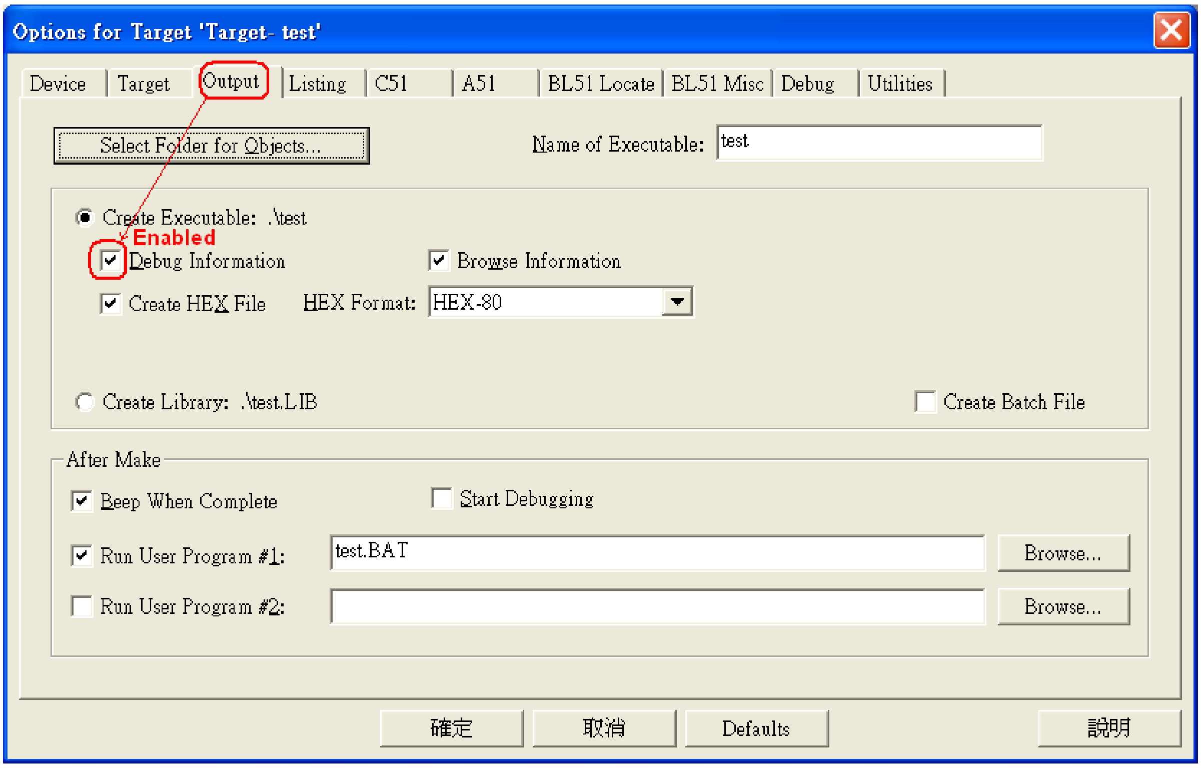
	Start:	Size:
Ram #1:		
Ram #2:		
Ram #3:		

□ Code Banking Start: End:
Banks: 2 Bank Area: 0x0000 0xFFFF

□ 'far' memory type support
□ Save address extension SFR in interrupts

4.3 Options- Output

Enable the “Debug Information”. It is necessary for creating an absolute OMF file for source-level debugging.



Options for Target 'Target- test'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Select Folder for Objects...   Name of Executable: test

● Create Executable: .\test

Enabled

☑ Debug Information   ☑ Browse Information

☑ Create HEX File   HEX Format: HEX-80

○ Create Library: .\test.LIB   □ Create Batch File

After Make

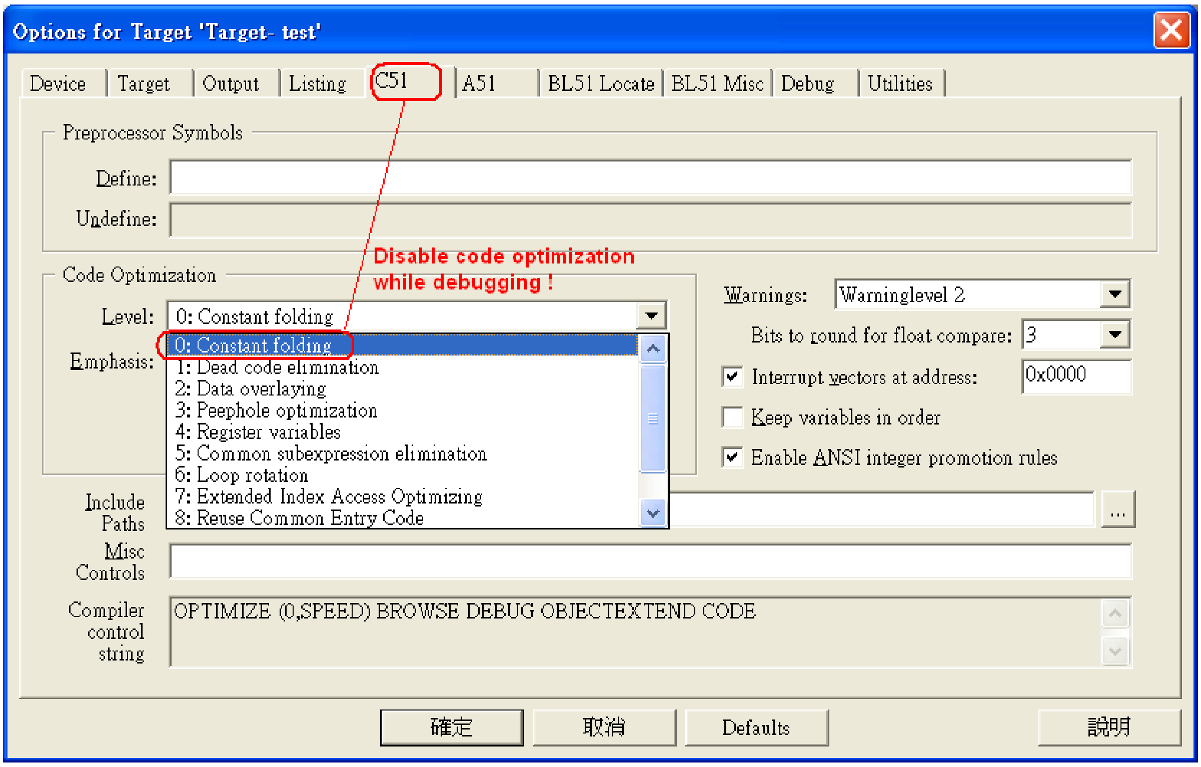
☑ Beep When Complete   □ Start Debugging

☑ Run User Program #1: test.BAT   Browse...

□ Run User Program #2:   Browse...

4.4 Options- C51

Disable the code optimization by selecting “Level 0: Constant folding”. Refer to *Section 6.3* for more information about this setting. *Note: This setting is optional.*



Options for Target 'Target- test'

Device | Target | Output | Listing | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Preprocessor Symbols

Define:

Undefine:

Code Optimization

Disable code optimization while debugging !

Level: 0: Constant folding

Emphasis:

0: Constant folding
1: Dead code elimination
2: Data overlaying
3: Peephole optimization
4: Register variables
5: Common subexpression elimination
6: Loop rotation
7: Extended Index Access Optimizing
8: Reuse Common Entry Code

Warnings: Warninglevel 2

Bits to round for float compare: 3

☑ Interrupt vectors at address: 0x0000

□ Keep variables in order

☑ Enable ANSI integer promotion rules

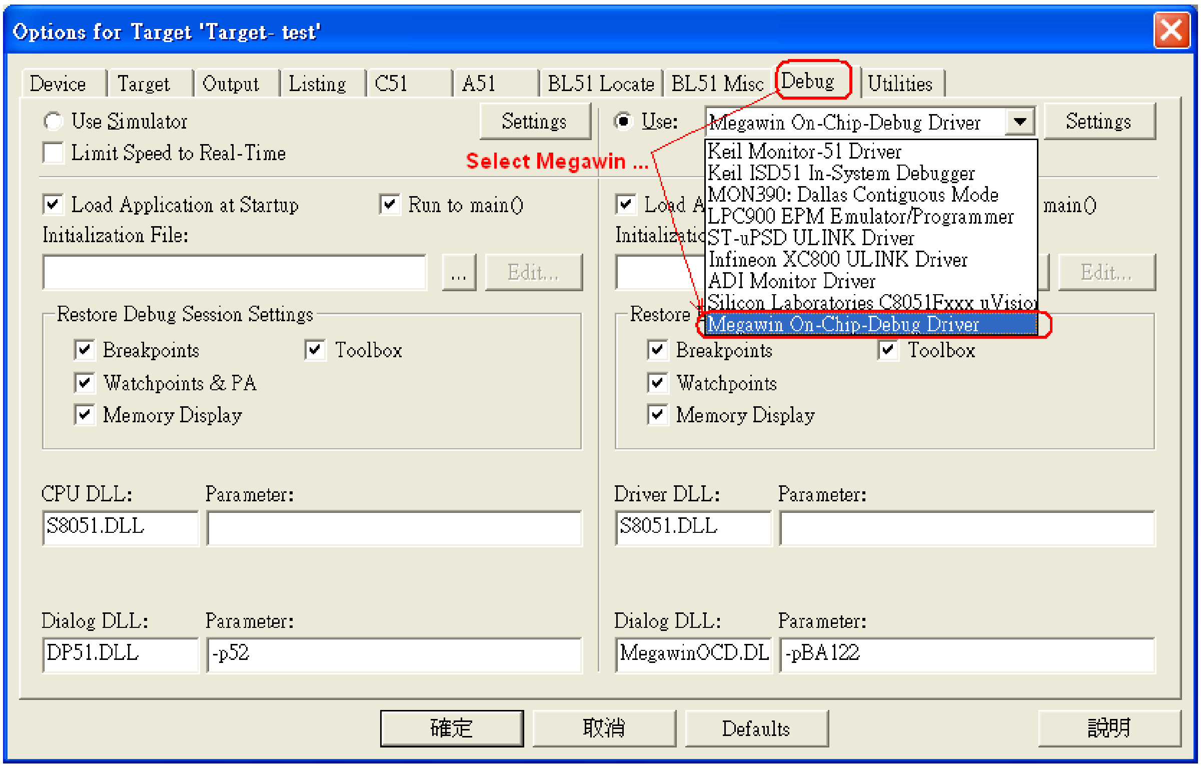
Include Paths

Misc Controls

Compiler control string: OPTIMIZE (0,SPEED) BROWSE DEBUG OBJECTEXTEND CODE

4.5 Options- Debug

Select the “Megawin On-Chip-Debug Driver”.

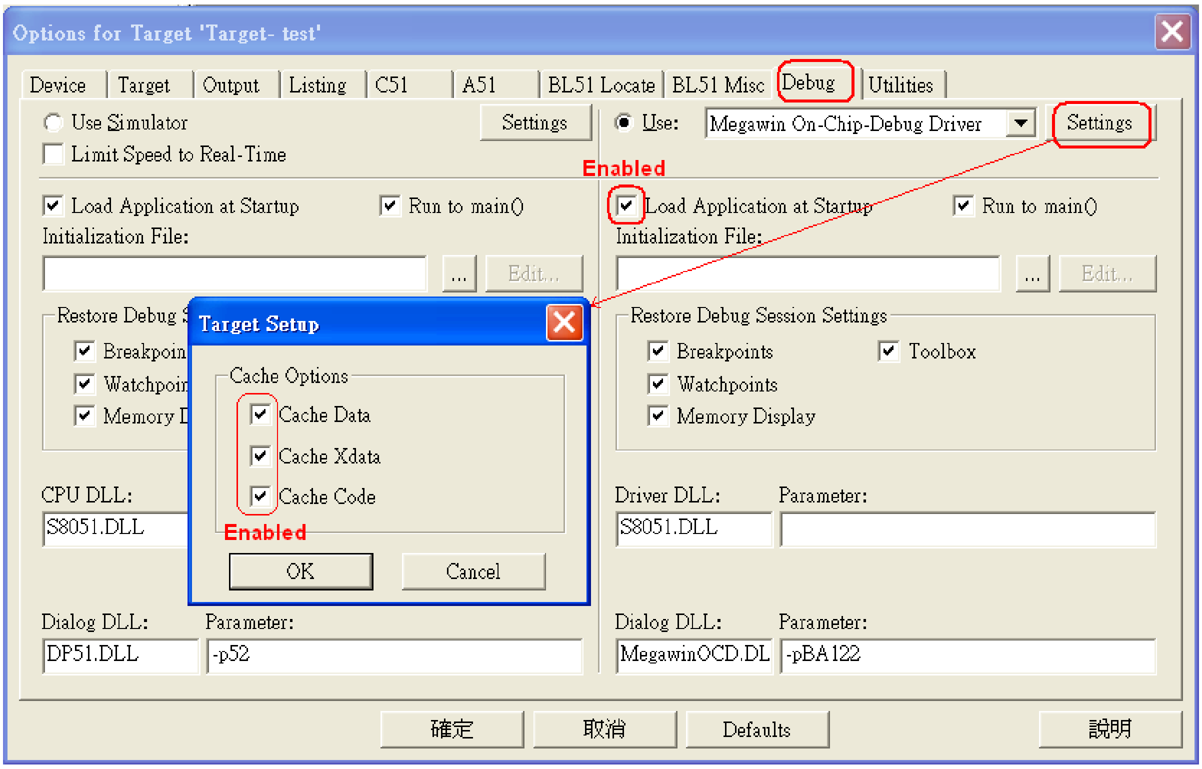


Options for Target 'Target- test'

○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watchpoints & PA
☑ Memory Display
CPU DLL: S8051.DLL Parameter:
Dialog DLL: DP51.DLL Parameter: -p52

● Use: Megawin On-Chip-Debug Driver
Keil Monitor-51 Driver
Keil ISD51 In-System Debugger
MON390: Dallas Contiguous Mode
LPC900 EPM Emulator/Programmer
ST-uPSD ULINK Driver
Infineon XC800 ULINK Driver
ADI Monitor Driver
Silicon Laboratories C8051Fxxx uVisio
Megawin On-Chip-Debug Driver
☑ Load A... main()
Initializatio...
Restore
☑ Breakpoints ☑ Toolbox
☑ Watchpoints
☑ Memory Display
Driver DLL: S8051.DLL Parameter:
Dialog DLL: MegawinOCD.DL Parameter: -pBA122

And, enable the “Load Application at Startup” and all the Cache Options.



Options for Target 'Target- test'

○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug S...
☑ Breakpoin...
☑ Watchpoin...
☑ Memory D...
CPU DLL: S8051.DLL
Dialog DLL: DP51.DLL Parameter: -p52

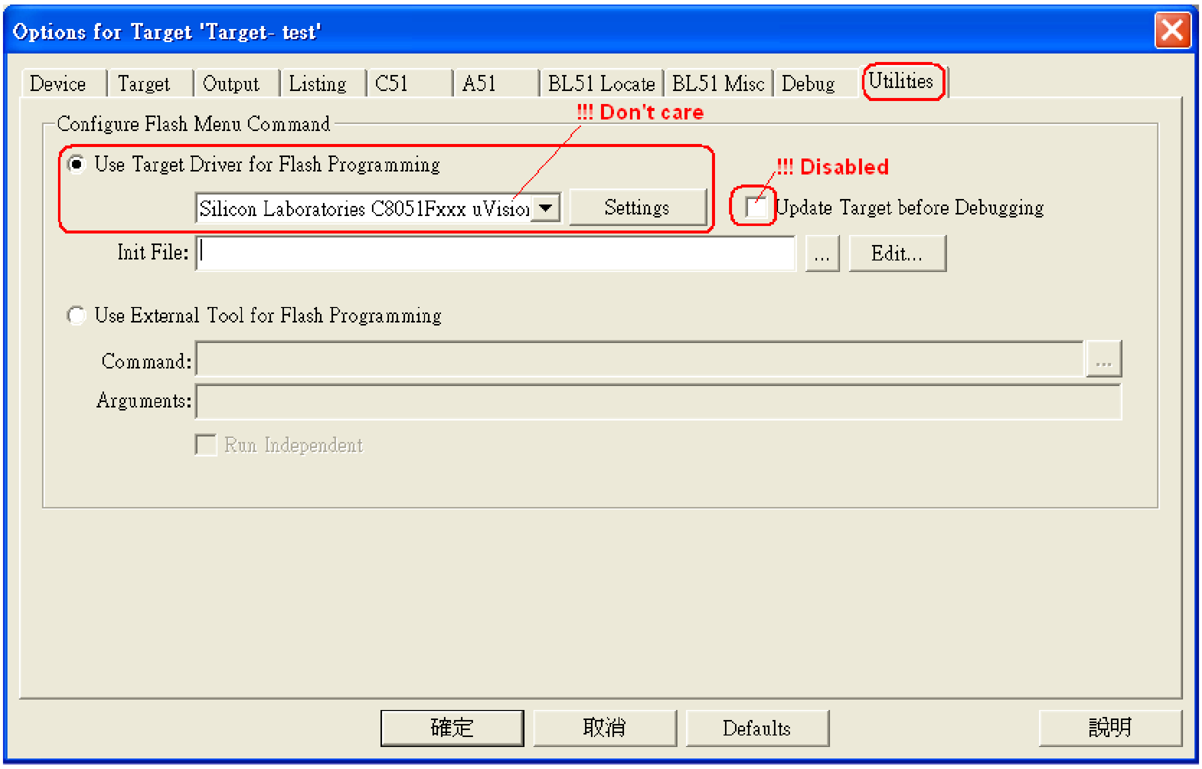
● Use: Megawin On-Chip-Debug Driver
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watchpoints
☑ Memory Display
Driver DLL: S8051.DLL Parameter:
Dialog DLL: MegawinOCD.DL Parameter: -pBA122

Target Setup
Cache Options
☑ Cache Data
☑ Cache Xdata
☑ Cache Code

4.6 Options- Utilities

Always disable the “Update Target before Debugging”. It is because we have enabled the “Load Application at Startup” shown in *Section 4.5*. And, leave the ”Use Target Driver for Flash Programming” *don't-cared*.

Note: µVision2 doesn't have this selection item.



Options for Target 'Target- test'

Configure Flash Menu Command

!!! Don't care

● Use Target Driver for Flash Programming

Silicon Laboratories C8051Fxxx uVision

!!! Disabled

□ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

Arguments:

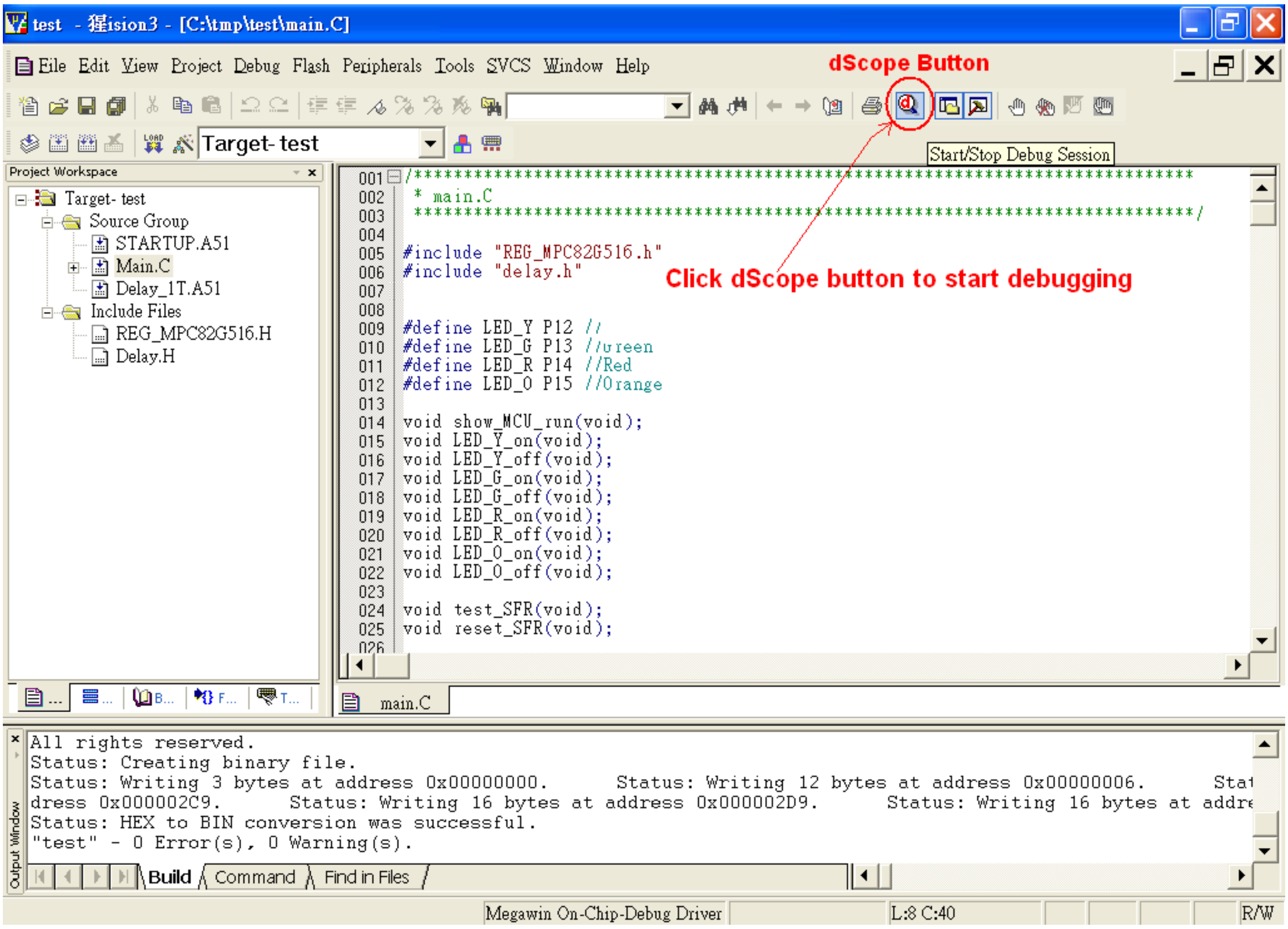
□ Run Independent

5 Start Debugging

After the tasks described in *Sections 2, 3 and 4* have been done, you can start debugging your µVision project.

5.1 Activate the *dScope*-Debugger Function

After building the project (suppose no error), you can enter the Keil IDE’s debugger mode by clicking the *dScope* button, as shown below. Now, the project code will be automatically downloaded into the target’s Flash. It will take some time.



dScope Button

Click dScope button to start debugging

```
/*****************************************************************************
 * main.C
 *****************************************************************************/

#include "REG_MPC82G516.h"
#include "delay.h"


#define LED_Y P12 //
#define LED_G P13 //ureen
#define LED_R P14 //Red
#define LED_O P15 //Orange

void show_MCU_run(void);
void LED_Y_on(void);
void LED_Y_off(void);
void LED_G_on(void);
void LED_G_off(void);
void LED_R_on(void);
void LED_R_off(void);
void LED_O_on(void);
void LED_O_off(void);

void test_SFR(void);
void reset_SFR(void);
```

```
All rights reserved.
Status: Creating binary file.
Status: Writing 3 bytes at address 0x00000000.      Status: Writing 12 bytes at address 0x00000006.      Stat
dress 0x000002C9.      Status: Writing 16 bytes at address 0x000002D9.      Status: Writing 16 bytes at addre
Status: HEX to BIN conversion was successful.
"test" - 0 Error(s), 0 Warning(s).
```

5.2 Introduction to the Debugger Environment

There are four basic windows regarding the debugging operation in the debugger environment. They are Register Window, Disassembly Window, Watch Window and Memory Window, as described below.

Register Window

This window shows the contents of the current register bank (R0~R7), the system registers (A, B, SP, DTPR and the Program Counter) and the Program Status Word (PSW). The register with blue background means its content is just changed due to the instruction just executed.

Disassembly Window

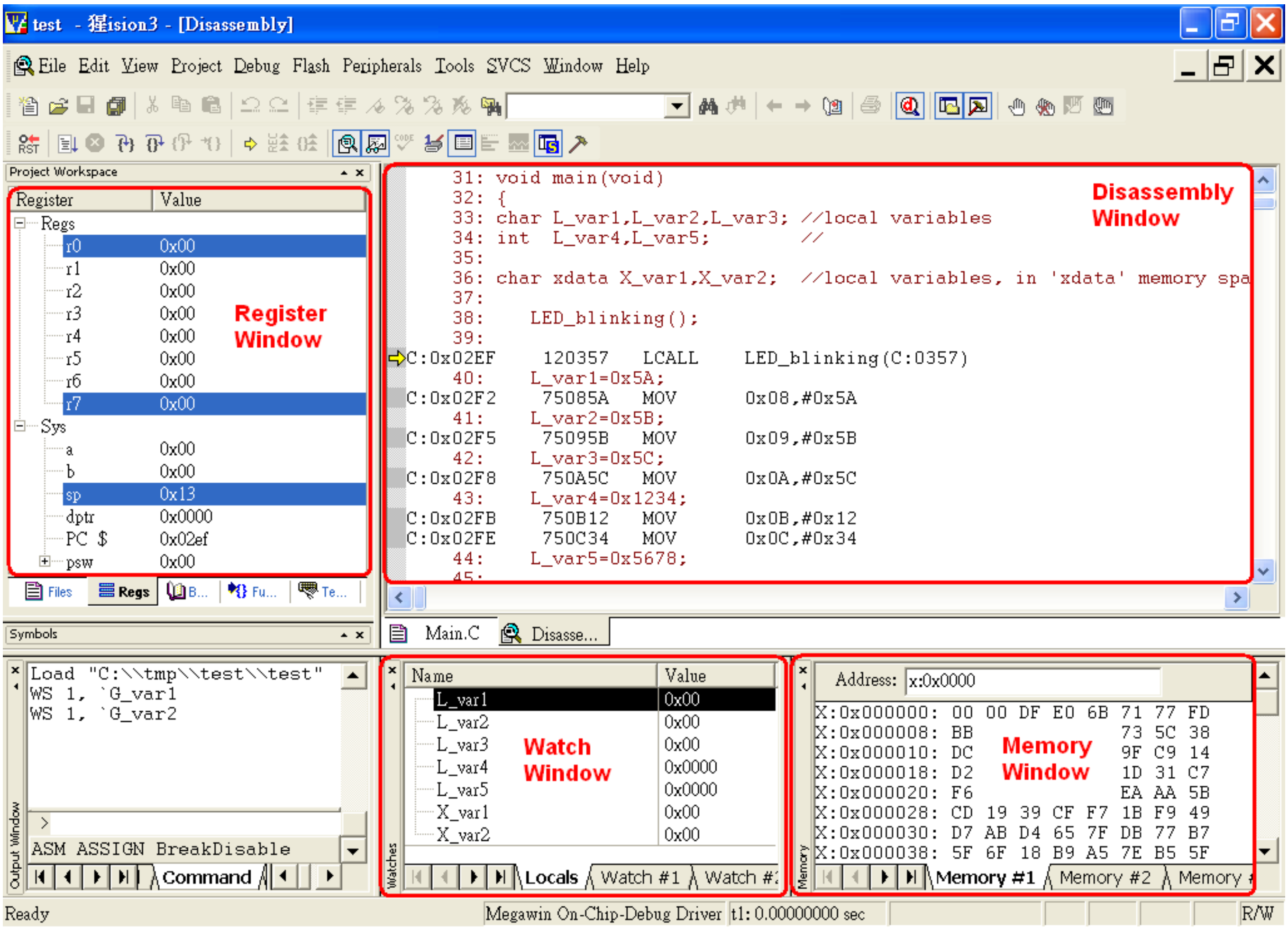
This window is the default window opened just when the debugger mode is entered. It shows the source-level code followed by its corresponding assembly code.

Watch Window

This window automatically shows the local variables when **Locals** is clicked. The local variables are the variables declared within a function including the main() function. To view the global variables, click **Watch #1** or **Watch #2** and type <**F2**> key to edit and enter the variable name. The variable with blue background means its content is just changed due to the instruction just executed.

Memory Window

This window shows the contents of the memory located at the **data**/**idata**/**xdata**/**code** memory space. The available commands are: d:0x00~d:0xFF, i:0x00~i:0xFF, x:0x0000~x:0xFFFF and c:0x0000~c:0xFFFF. The user can view any of the four memory by entering the corresponding command.


test - 礭ision3 - [Disassembly]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register Window
Files Regs B... Fu... Te...
Symbols
Disassembly Window
Main.C Disasse...
Output Window
Command
Watches
Watch Window
Locals Watch #1 Watch #2
Memory
Memory Window
Memory #1 Memory #2 Memory #3
Ready
Megawin On-Chip-Debug Driver
t1: 0.00000000 sec
R/W

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x13
dptr	0x0000
PC $	0x02ef
psw	0x00

```
    31: void main(void)
    32: {
    33: char L_var1,L_var2,L_var3; //local variables
    34: int  L_var4,L_var5;        //
    35:
    36: char xdata X_var1,X_var2;  //local variables, in 'xdata' memory spa
    37:
    38:     LED_blinking();
    39:
C:0x02EF    120357   LCALL    LED_blinking(C:0357)
    40:     L_var1=0x5A;
C:0x02F2    75085A   MOV      0x08,#0x5A
    41:     L_var2=0x5B;
C:0x02F5    75095B   MOV      0x09,#0x5B
    42:     L_var3=0x5C;
C:0x02F8    750A5C   MOV      0x0A,#0x5C
    43:     L_var4=0x1234;
C:0x02FB    750B12   MOV      0x0B,#0x12
C:0x02FE    750C34   MOV      0x0C,#0x34
    44:     L_var5=0x5678;
    45:
```

```
Load "C:\\tmp\\test\\test"
WS 1, `G_var1
WS 1, `G_var2
>
ASM ASSIGN BreakDisable
```

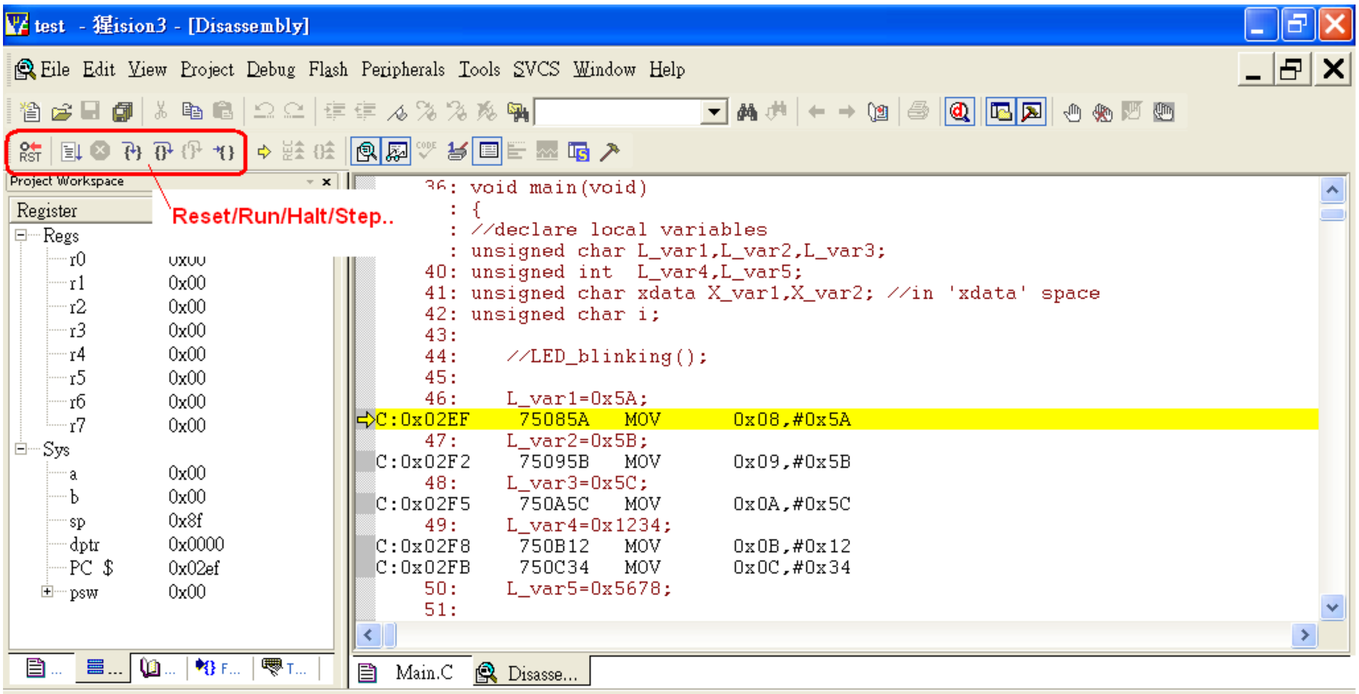
Name	Value
L_var1	0x00
L_var2	0x00
L_var3	0x00
L_var4	0x0000
L_var5	0x0000
X_var1	0x00
X_var2	0x00

Address: x:0x0000

```
X:0x000000: 00 00 DF E0 6B 71 77 FD
X:0x000008: BB                73 5C 38
X:0x000010: DC                9F C9 14
X:0x000018: D2                1D 31 C7
X:0x000020: F6                EA AA 5B
X:0x000028: CD 19 39 CF F7 1B F9 49
X:0x000030: D7 AB D4 65 7F DB 77 B7
X:0x000038: 5F 6F 18 B9 A5 7E B5 5F
```

5.2.1 Reset/Run/Halt/Step/Run-to-Cursor

Reset, Run, Halt (Stop), Step and Run-to-Cursor are the basic debug actions. The user can easily invoke any of these actions by clicking the short-cut buttons in the debugger GUI, as shown below.

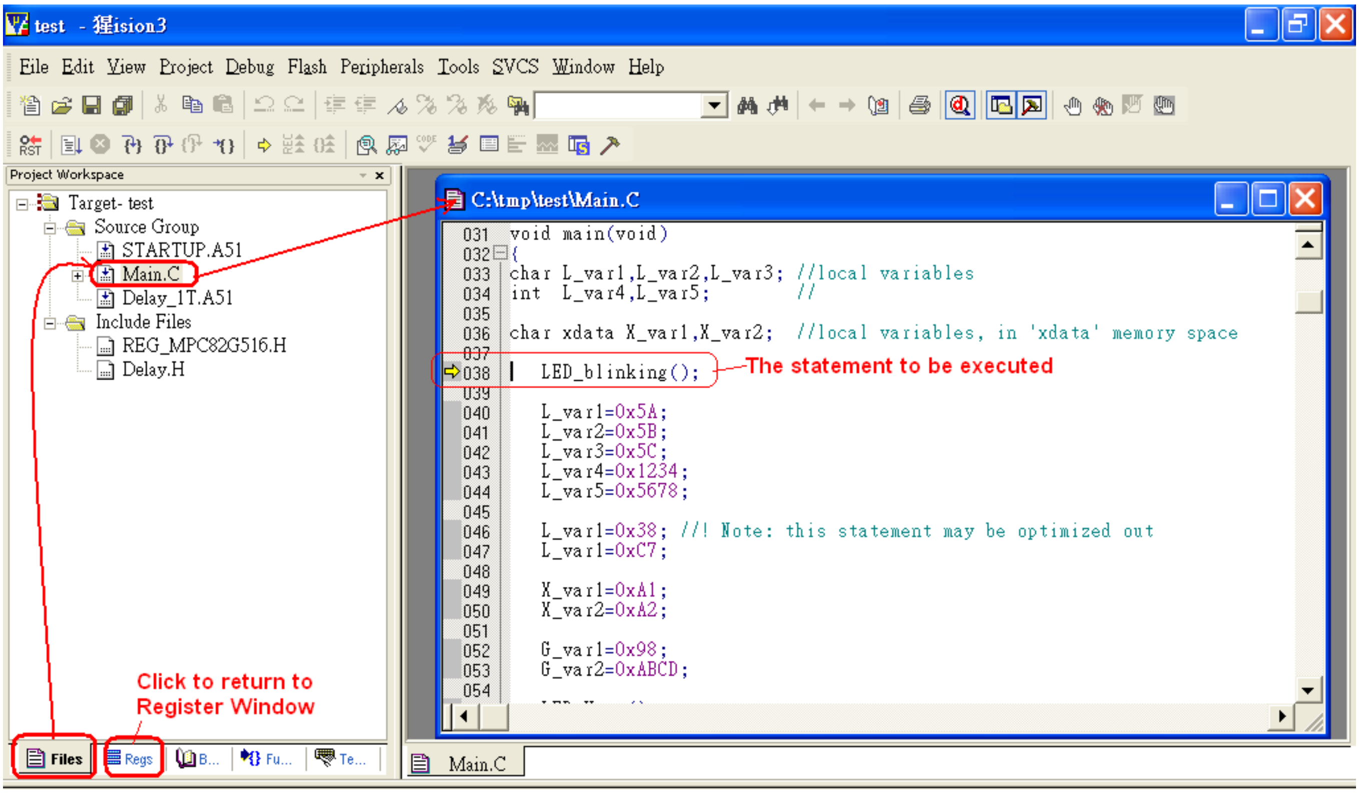

test - 猩ision3 - [Disassembly]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
RST
Project Workspace
Reset/Run/Halt/Step..
Main.C
Disasse...

Register	
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x8f
dptr	0x0000
PC $	0x02ef
psw	0x00

```
    36: void main(void)
      : {
      : //declare local variables
      : unsigned char L_var1,L_var2,L_var3;
    40: unsigned int  L_var4,L_var5;
    41: unsigned char xdata X_var1,X_var2; //in 'xdata' space
    42: unsigned char i;
    43:
    44:    //LED_blinking();
    45:
    46:    L_var1=0x5A;
C:0x02EF    75085A   MOV      0x08,#0x5A
    47:    L_var2=0x5B;
C:0x02F2    75095B   MOV      0x09,#0x5B
    48:    L_var3=0x5C;
C:0x02F5    750A5C   MOV      0x0A,#0x5C
    49:    L_var4=0x1234;
C:0x02F8    750B12   MOV      0x0B,#0x12
C:0x02FB    750C34   MOV      0x0C,#0x34
    50:    L_var5=0x5678;
    51:
```

5.2.2 Source-Level Debugging

To do the source-level debugging, open the source file by clicking **Files** to open the Project Workspace and select the source files you want. Click **Regs** again to return to Register Window, as shown below.



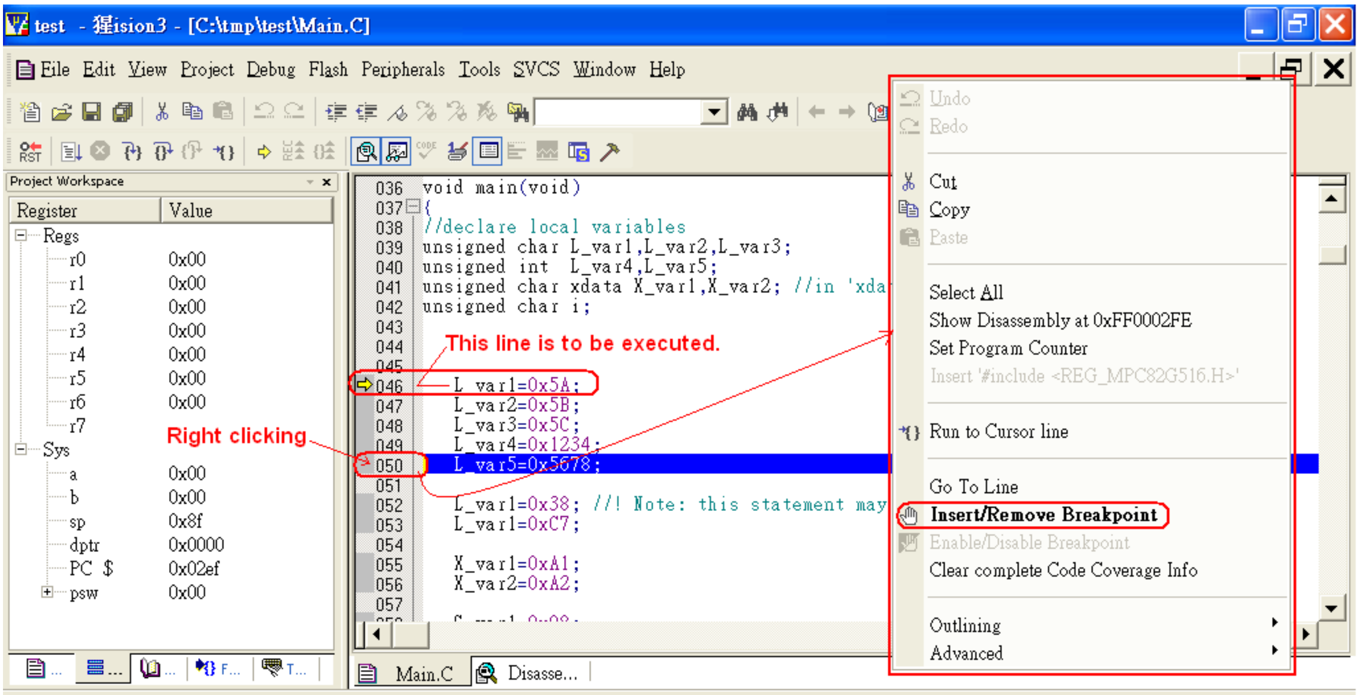
```
void main(void)
{
char L_var1,L_var2,L_var3; //local variables
int  L_var4,L_var5;        //

char xdata X_var1,X_var2;  //local variables, in 'xdata' memory space

    LED_blinking();

    L_var1=0x5A;
    L_var2=0x5B;
    L_var3=0x5C;
    L_var4=0x1234;
    L_var5=0x5678;

    L_var1=0x38; //! Note: this statement may be optimized out
    L_var1=0xC7;

    X_var1=0xA1;
    X_var2=0xA2;

    G_var1=0x98;
    G_var2=0xABCD;
```

5.2.3 Breakpoint Setting

There are total four breakpoints available for debugging. Up to four breakpoints can be inserted simultaneously.

Insert/Remove a Breakpoint

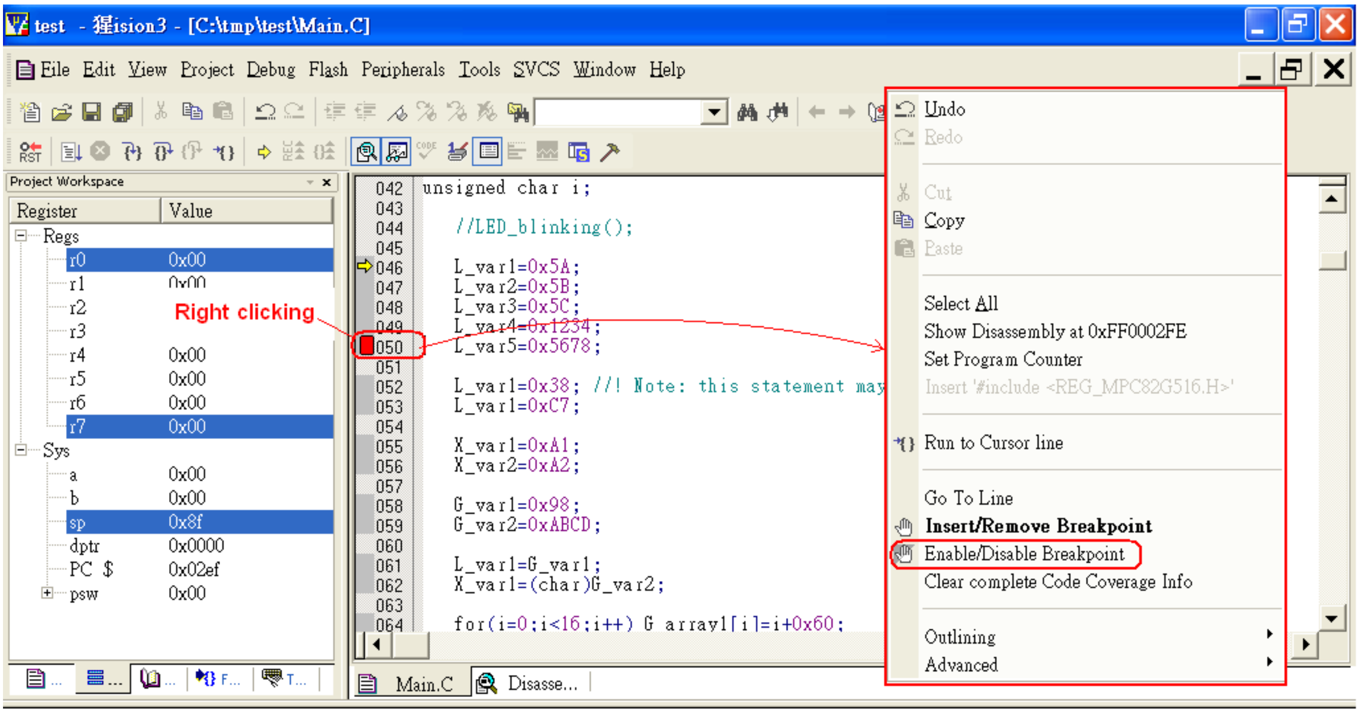
Move the cursor to the front of the line and click the right key, then click “**Insert/Remove Breakpoint**” for toggling between Insert and Remove, as shown below.



```
void main(void)
{
//declare local variables
unsigned char L_var1,L_var2,L_var3;
unsigned int  L_var4,L_var5;
unsigned char xdata X_var1,X_var2; //in 'xda
unsigned char i;


  L_var1=0x5A;
  L_var2=0x5B;
  L_var3=0x5C;
  L_var4=0x1234;
  L_var5=0x5678;

  L_var1=0x38; //! Note: this statement may
  L_var1=0xC7;

  X_var1=0xA1;
  X_var2=0xA2;
```

Enable/Disable a Breakpoint

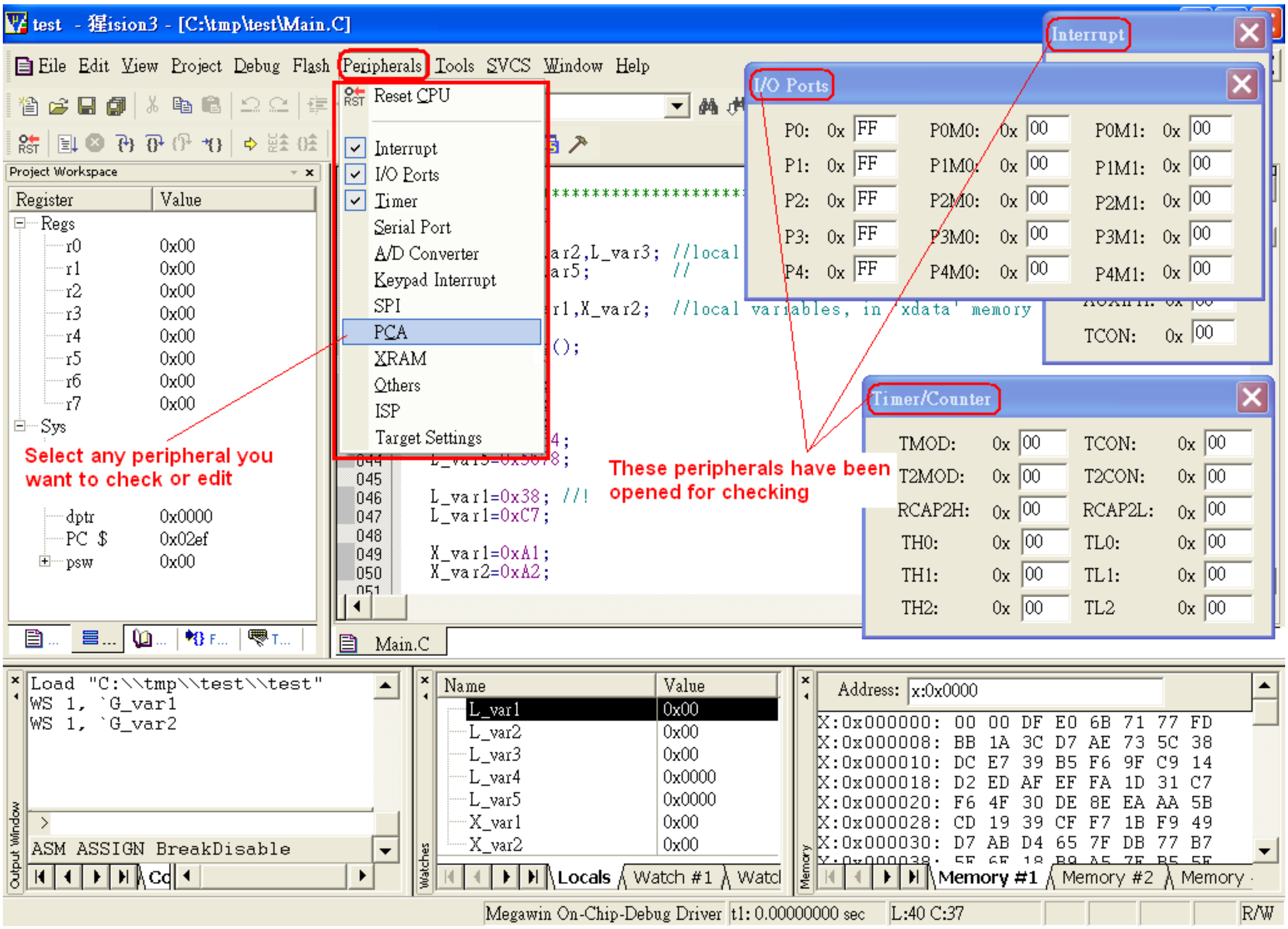
Move the cursor to the front of the line and click the right key, then click “**Enable/Disable Breakpoint**” for toggling between Enable and Disable. Of course, this line should have been inserted a breakpoint previously.



```
unsigned char i;

  //LED_blinking();

  L_var1=0x5A;
  L_var2=0x5B;
  L_var3=0x5C;
  L_var4=0x1234;
  L_var5=0x5678;

  L_var1=0x38; //! Note: this statement may
  L_var1=0xC7;

  X_var1=0xA1;
  X_var2=0xA2;

  G_var1=0x98;
  G_var2=0xABCD;

  L_var1=G_var1;
  X_var1=(char)G_var2;

  for(i=0;i<16;i++) G_array1[i]=i+0x60;
```

5.2.4 View/Edit the Contents of Peripherals' SFRs

There are many peripheral SFRs that don't belong to the registers shown in the Register Window. To view or edit these registers, select the **Peripherals** item on the main menu. A pulled-down sub-menu will be displayed, and the user can select a peripheral to view or edit its corresponding SFRs, as shown below.



Select any peripheral you want to check or edit

These peripherals have been opened for checking

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
dptr	0x0000
PC $	0x02ef
psw	0x00

I/O Ports:

P0: 0x FF		P0M0: 0x 00		P0M1: 0x 00	
P1: 0x FF		P1M0: 0x 00		P1M1: 0x 00	
P2: 0x FF		P2M0: 0x 00		P2M1: 0x 00	
P3: 0x FF		P3M0: 0x 00		P3M1: 0x 00	
P4: 0x FF		P4M0: 0x 00		P4M1: 0x 00	

Interrupt: TCON: 0x 00

Timer/Counter:

TMOD: 0x 00	TCON: 0x 00
T2MOD: 0x 00	T2CON: 0x 00
RCAP2H: 0x 00	RCAP2L: 0x 00
TH0: 0x 00	TL0: 0x 00
TH1: 0x 00	TL1: 0x 00
TH2: 0x 00	TL2 0x 00

```
ar2,L_var3;  //local
ar5;         //
r1,X_var2;   //local variables, in 'xdata' memory
();
4;
L_var5=0x5678;
L_var1=0x38; //!
L_var1=0xC7;
X_var1=0xA1;
X_var2=0xA2;
```

```
Load "C:\\tmp\\test\\test"
WS 1, `G_var1
WS 1, `G_var2
>
ASM ASSIGN BreakDisable
```

Name	Value
L_var1	0x00
L_var2	0x00
L_var3	0x00
L_var4	0x0000
L_var5	0x0000
X_var1	0x00
X_var2	0x00

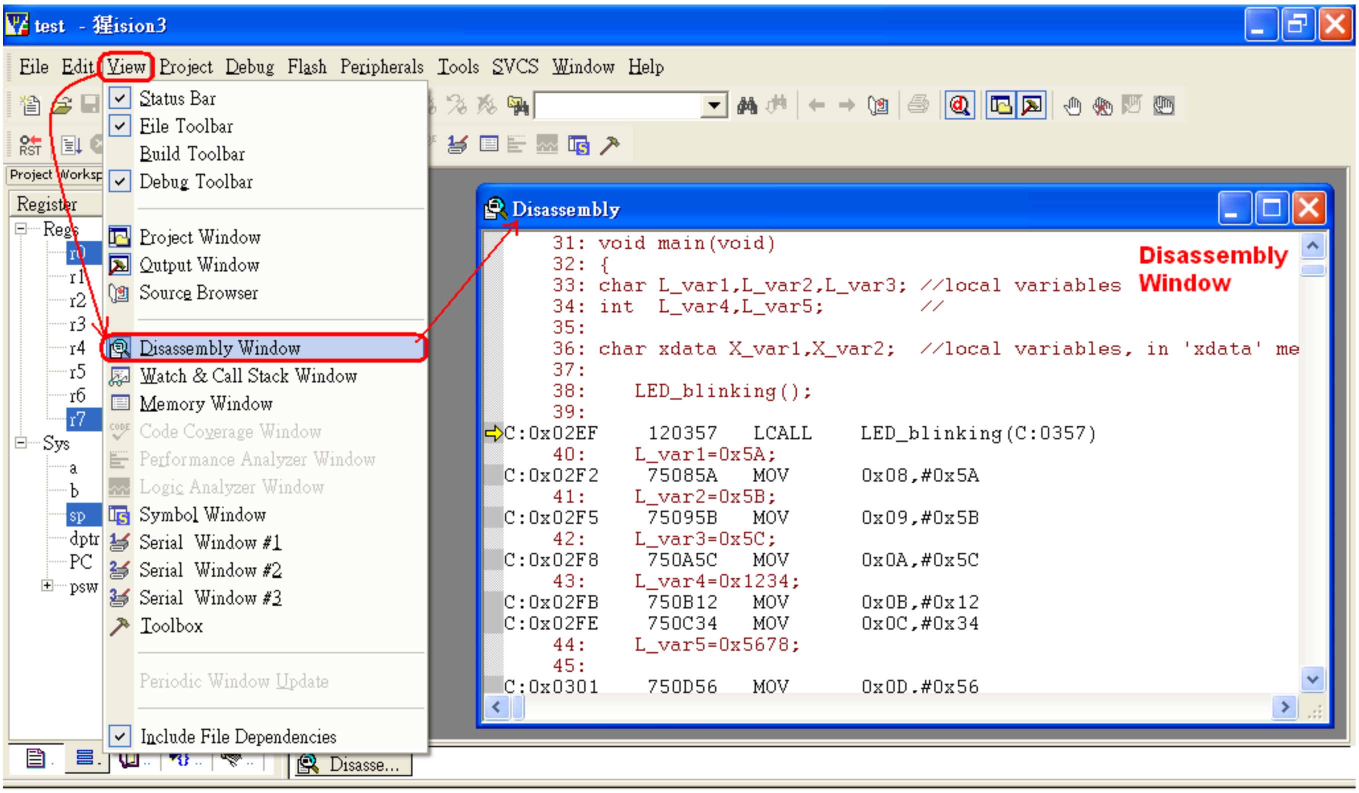
Address: x:0x0000

```
X:0x000000: 00 00 DF E0 6B 71 77 FD
X:0x000008: BB 1A 3C D7 AE 73 5C 38
X:0x000010: DC E7 39 B5 F6 9F C9 14
X:0x000018: D2 ED AF EF FA 1D 31 C7
X:0x000020: F6 4F 30 DE 8E EA AA 5B
X:0x000028: CD 19 39 CF F7 1B F9 49
X:0x000030: D7 AB D4 65 7F DB 77 B7
```

Megawin On-Chip-Debug Driver | t1: 0.00000000 sec | L:40 C:37 | R/W

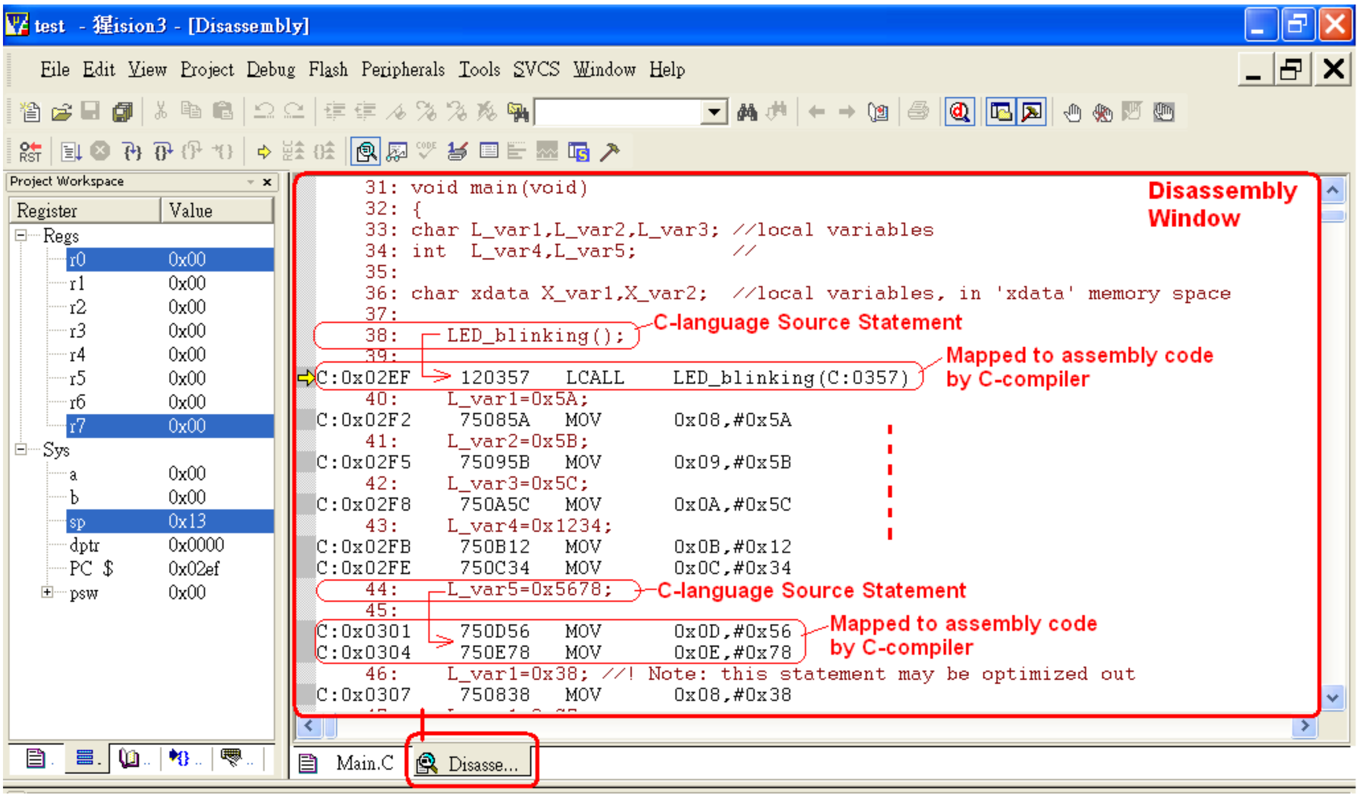
5.2.5 View- Disassembly Window

Disassembly Window displays source-level code followed by its corresponding assembly. To open this window, select the **View** item on the main menu. A pulled-down sub-menu will be displayed, and then select **Disassembly Window**, as shown below.



```
     31: void main(void)
     32: {
     33: char L_var1,L_var2,L_var3; //local variables
     34: int  L_var4,L_var5;        //
     35:
     36: char xdata X_var1,X_var2;  //local variables, in 'xdata' me
     37:
     38:    LED_blinking();
     39:
C:0x02EF    120357   LCALL    LED_blinking(C:0357)
     40:    L_var1=0x5A;
C:0x02F2    75085A   MOV      0x08,#0x5A
     41:    L_var2=0x5B;
C:0x02F5    75095B   MOV      0x09,#0x5B
     42:    L_var3=0x5C;
C:0x02F8    750A5C   MOV      0x0A,#0x5C
     43:    L_var4=0x1234;
C:0x02FB    750B12   MOV      0x0B,#0x12
C:0x02FE    750C34   MOV      0x0C,#0x34
     44:    L_var5=0x5678;
     45:
C:0x0301    750D56   MOV      0x0D,#0x56
```

Maximize the Disassembly Window for detailed description:

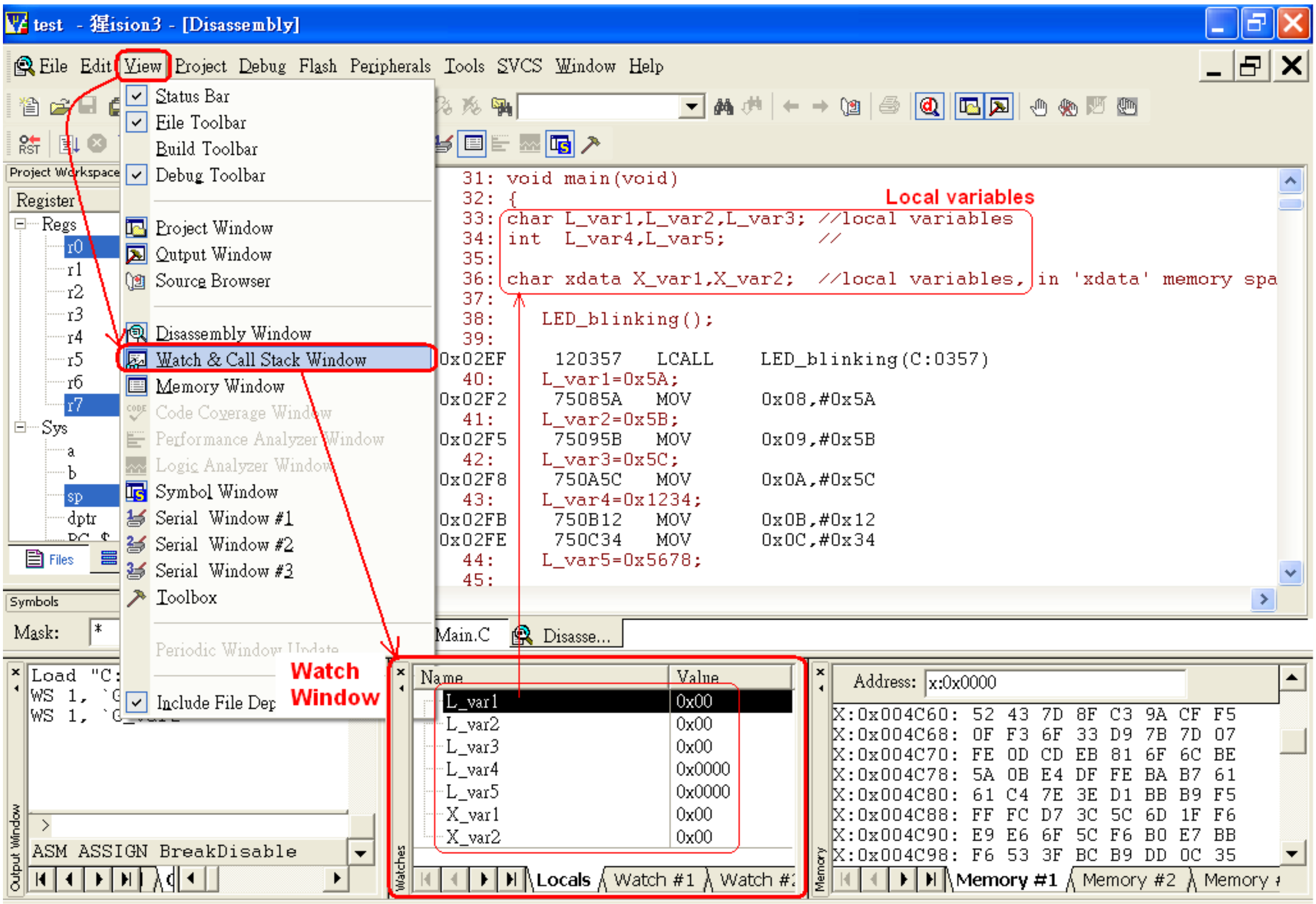

test - 𝕸ision3 - [Disassembly]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Disassembly Window
C-language Source Statement
Mapped to assembly code by C-compiler
C-language Source Statement
Mapped to assembly code by C-compiler
Main.C
Disasse...

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x13
dptr	0x0000
PC $	0x02ef
psw	0x00

```
     31: void main(void)
     32: {
     33: char L_var1,L_var2,L_var3; //local variables
     34: int  L_var4,L_var5;        //
     35:
     36: char xdata X_var1,X_var2;  //local variables, in 'xdata' memory space
     37:
     38:    LED_blinking();
     39:
C:0x02EF    120357   LCALL    LED_blinking(C:0357)
     40:    L_var1=0x5A;
C:0x02F2    75085A   MOV      0x08,#0x5A
     41:    L_var2=0x5B;
C:0x02F5    75095B   MOV      0x09,#0x5B
     42:    L_var3=0x5C;
C:0x02F8    750A5C   MOV      0x0A,#0x5C
     43:    L_var4=0x1234;
C:0x02FB    750B12   MOV      0x0B,#0x12
C:0x02FE    750C34   MOV      0x0C,#0x34
     44:    L_var5=0x5678;
     45:
C:0x0301    750D56   MOV      0x0D,#0x56
C:0x0304    750E78   MOV      0x0E,#0x78
     46:    L_var1=0x38; //! Note: this statement may be optimized out
C:0x0307    750838   MOV      0x08,#0x38
```

5.2.6 View- Watch Window

The Watch Window helps the user to check either local variables or global variables, as shown below.



```
  31: void main(void)
  32: {
  33: char L_var1,L_var2,L_var3; //local variables
  34: int  L_var4,L_var5;         //
  35:
  36: char xdata X_var1,X_var2;  //local variables, in 'xdata' memory spa
  37:
  38:    LED_blinking();
  39:
0x02EF    120357    LCALL    LED_blinking(C:0357)
  40:    L_var1=0x5A;
0x02F2    75085A    MOV      0x08,#0x5A
  41:    L_var2=0x5B;
0x02F5    75095B    MOV      0x09,#0x5B
  42:    L_var3=0x5C;
0x02F8    750A5C    MOV      0x0A,#0x5C
  43:    L_var4=0x1234;
0x02FB    750B12    MOV      0x0B,#0x12
0x02FE    750C34    MOV      0x0C,#0x34
  44:    L_var5=0x5678;
  45:
```

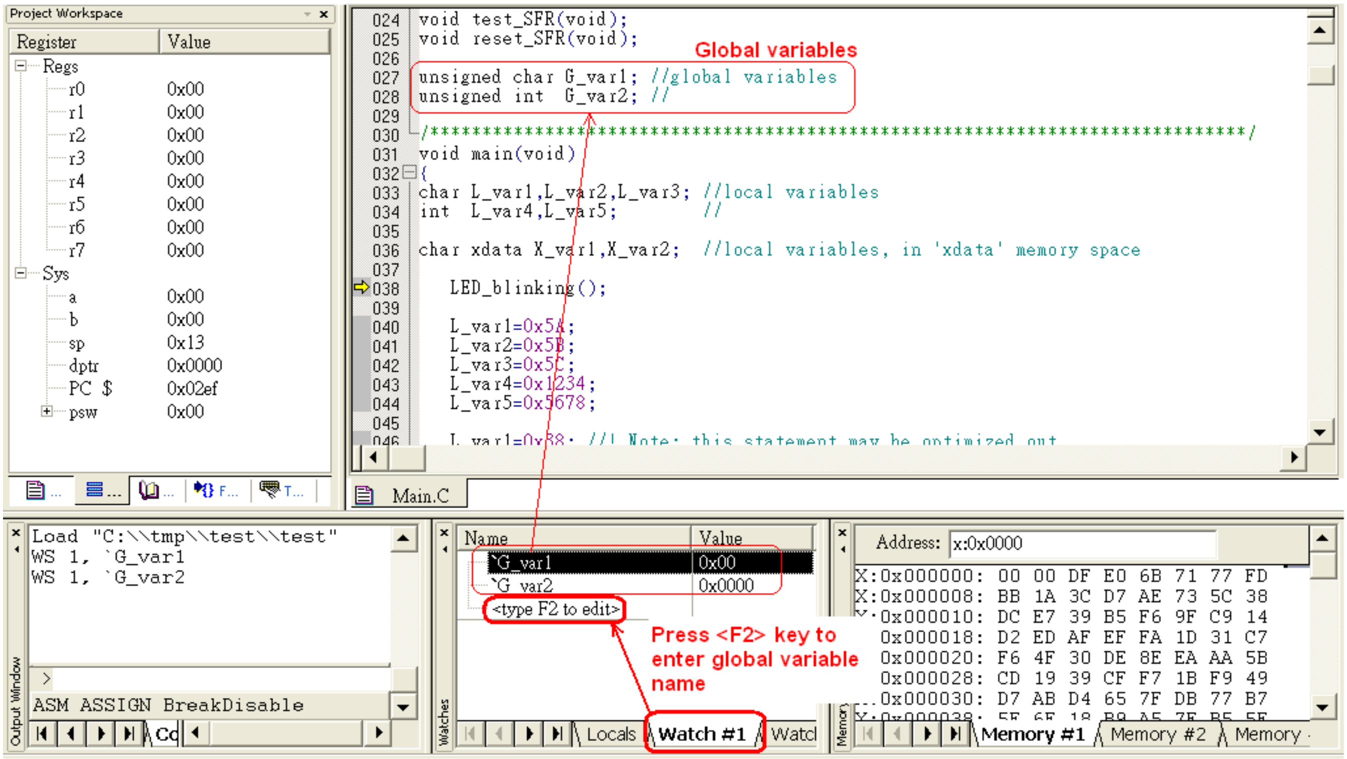
```
Load "C:
WS 1, `G
WS 1, `G_var2
ASM ASSIGN BreakDisable
```

Name	Value
L_var1	0x00
L_var2	0x00
L_var3	0x00
L_var4	0x0000
L_var5	0x0000
X_var1	0x00
X_var2	0x00

Address: x:0x0000

```
X:0x004C60:  52 43 7D 8F C3 9A CF F5
X:0x004C68:  0F F3 6F 33 D9 7B 7D 07
X:0x004C70:  FE 0D CD EB 81 6F 6C BE
X:0x004C78:  5A 0B E4 DF FE BA B7 61
X:0x004C80:  61 C4 7E 3E D1 BB B9 F5
X:0x004C88:  FF FC D7 3C 5C 6D 1F F6
X:0x004C90:  E9 E6 6F 5C F6 B0 E7 BB
X:0x004C98:  F6 53 3F BC B9 DD 0C 35
```

To check the global variables, click **Watch #1** or **#2**, then type <**F2**> key to enter the variable name.



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x13
dptr	0x0000
PC $	0x02ef
psw	0x00

```
void test_SFR(void);
void reset_SFR(void);

unsigned char G_var1; //global variables
unsigned int  G_var2; //
/************************************************************************/
void main(void)
{
char L_var1,L_var2,L_var3; //local variables
int  L_var4,L_var5;        //

char xdata X_var1,X_var2;  //local variables, in 'xdata' memory space

   LED_blinking();

   L_var1=0x5A;
   L_var2=0x5B;
   L_var3=0x5C;
   L_var4=0x1234;
   L_var5=0x5678;
   L_var1=0x88; //! Note: this statement may be optimized out
```

```
Load "C:\\tmp\\test\\test"
WS 1, `G_var1
WS 1, `G_var2
ASM ASSIGN BreakDisable
```

Name	Value
`G_var1	0x00
`G_var2	0x0000
<type F2 to edit>	

Address: x:0x0000

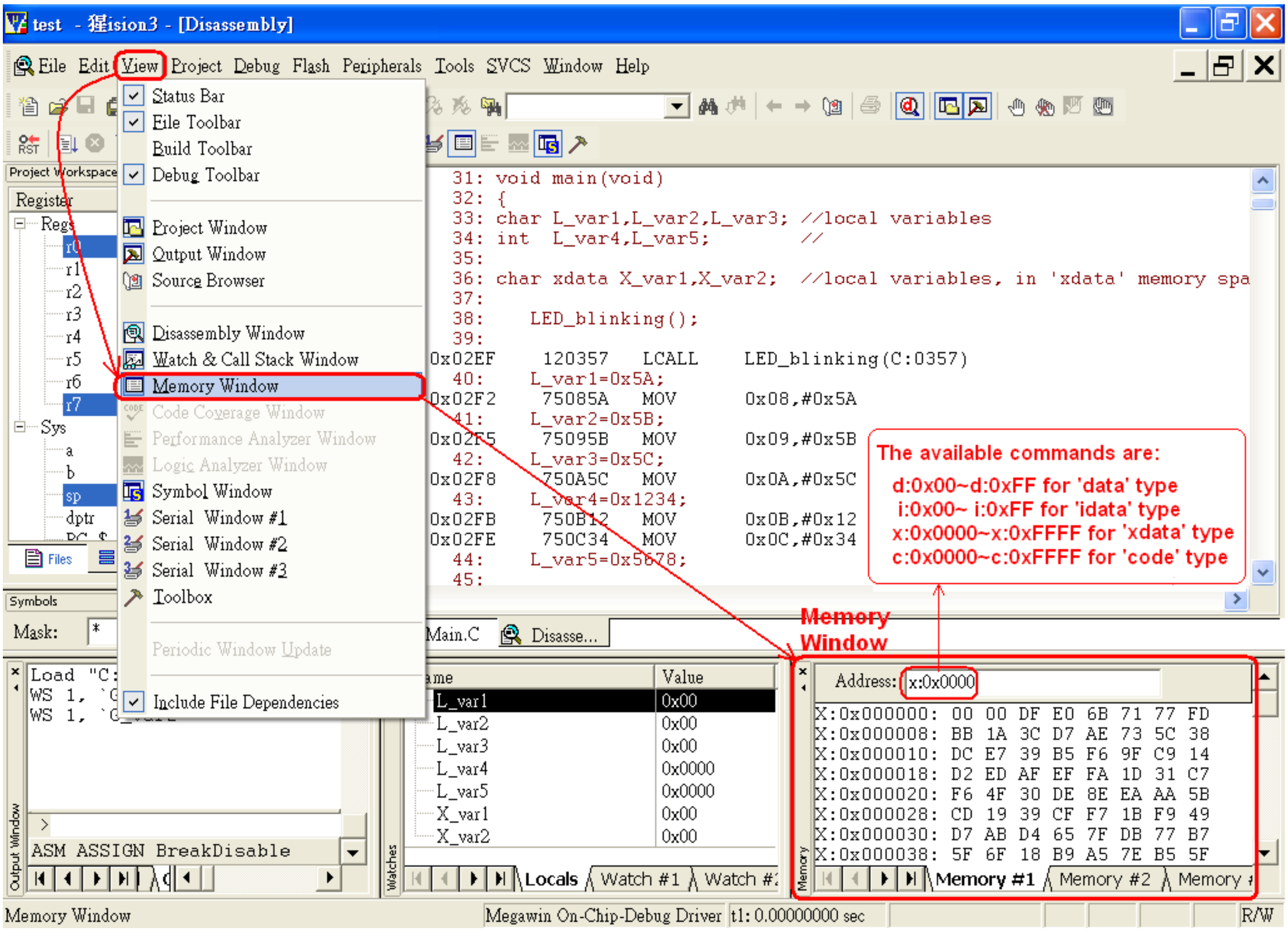
```
X:0x000000:  00 00 DF E0 6B 71 77 FD
X:0x000008:  BB 1A 3C D7 AE 73 5C 38
X:0x000010:  DC E7 39 B5 F6 9F C9 14
  0x000018:  D2 ED AF EF FA 1D 31 C7
  0x000020:  F6 4F 30 DE 8E EA AA 5B
  0x000028:  CD 19 39 CF F7 1B F9 49
  0x000030:  D7 AB D4 65 7F DB 77 B7
```

5.2.7 View- Memory Window

To open this window, select the **View** item on the main menu. A pulled-down sub-menu will be displayed, and then select **Memory Window**, as shown below. The available commands are:

(1) d:0x00~d:0xFF, for '**data**' type
(2) i:0x00~i:0xFF, for '**idata**' type
(3) x:0x0000~x:0xFFFF, for '**xdata**' type
(4) c:0x0000~c:0xFFFF., for '**code**' type

The user can view any of the four memory by entering the corresponding command. Refer to *Section 6.2* for how to display 'xdata' type variables.



```
  31: void main(void)
  32: {
  33: char L_var1,L_var2,L_var3; //local variables
  34: int  L_var4,L_var5;        //
  35:
  36: char xdata X_var1,X_var2;  //local variables, in 'xdata' memory spa
  37:
  38:     LED_blinking();
  39:
0x02EF    120357   LCALL    LED_blinking(C:0357)
  40:     L_var1=0x5A;
0x02F2    75085A   MOV      0x08,#0x5A
  41:     L_var2=0x5B;
0x02F5    75095B   MOV      0x09,#0x5B
  42:     L_var3=0x5C;
0x02F8    750A5C   MOV      0x0A,#0x5C
  43:     L_var4=0x1234;
0x02FB    750B12   MOV      0x0B,#0x12
0x02FE    750C34   MOV      0x0C,#0x34
  44:     L_var5=0x5678;
  45:
```

The available commands are:
d:0x00~d:0xFF for 'data' type
i:0x00~ i:0xFF for 'idata' type
x:0x0000~x:0xFFFF for 'xdata' type
c:0x0000~c:0xFFFF for 'code' type

Memory Window

```
Load "C:
WS 1, `G
WS 1, `G_var2
>
ASM ASSIGN BreakDisable
```

ame	Value
L_var1	0x00
L_var2	0x00
L_var3	0x00
L_var4	0x0000
L_var5	0x0000
X_var1	0x00
X_var2	0x00

Address: x:0x0000

```
X:0x000000: 00 00 DF E0 6B 71 77 FD
X:0x000008: BB 1A 3C D7 AE 73 5C 38
X:0x000010: DC E7 39 B5 F6 9F C9 14
X:0x000018: D2 ED AF EF FA 1D 31 C7
X:0x000020: F6 4F 30 DE 8E EA AA 5B
X:0x000028: CD 19 39 CF F7 1B F9 49
X:0x000030: D7 AB D4 65 7F DB 77 B7
X:0x000038: 5F 6F 18 B9 A5 7E B5 5F
```

6 Tools, Megawin ICP

6.1 About ICP

ICP is the acronym of In-Circuit Programming. Users can update the application code under the software control without removing the mounted MCU chip from the actual end product. In addition, because the programming data to be programmed to the target can be saved in the ICE adapter's non-volatile storage, this stand-alone programmer is able to work without host(PC) intervention. This feature is especially useful in the field without a PC.

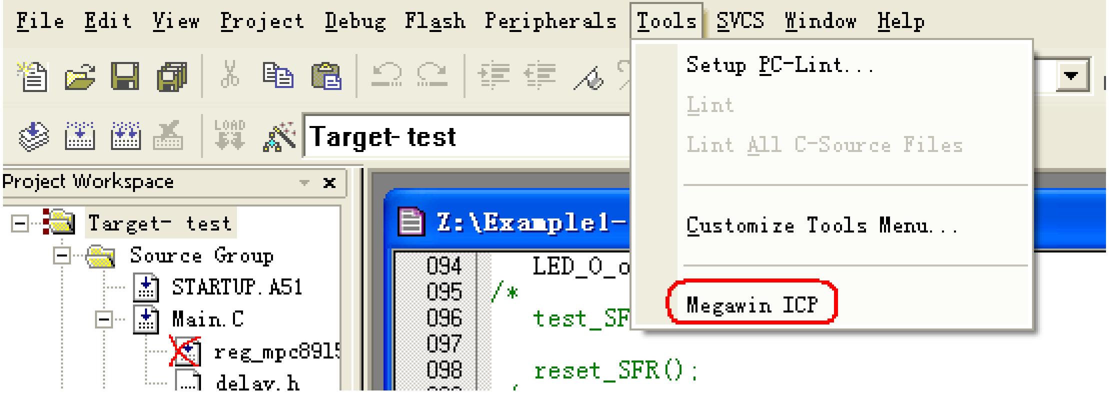
6.2 Use ICP

Here are the two ways of opening ICP application:

1. Execute "ICPProgrammer.exe" under "\C51\INC\Megawin\" of Keil's Install folder.

2. Click "Tools\Megawin ICP" from Keil's Menu bar.

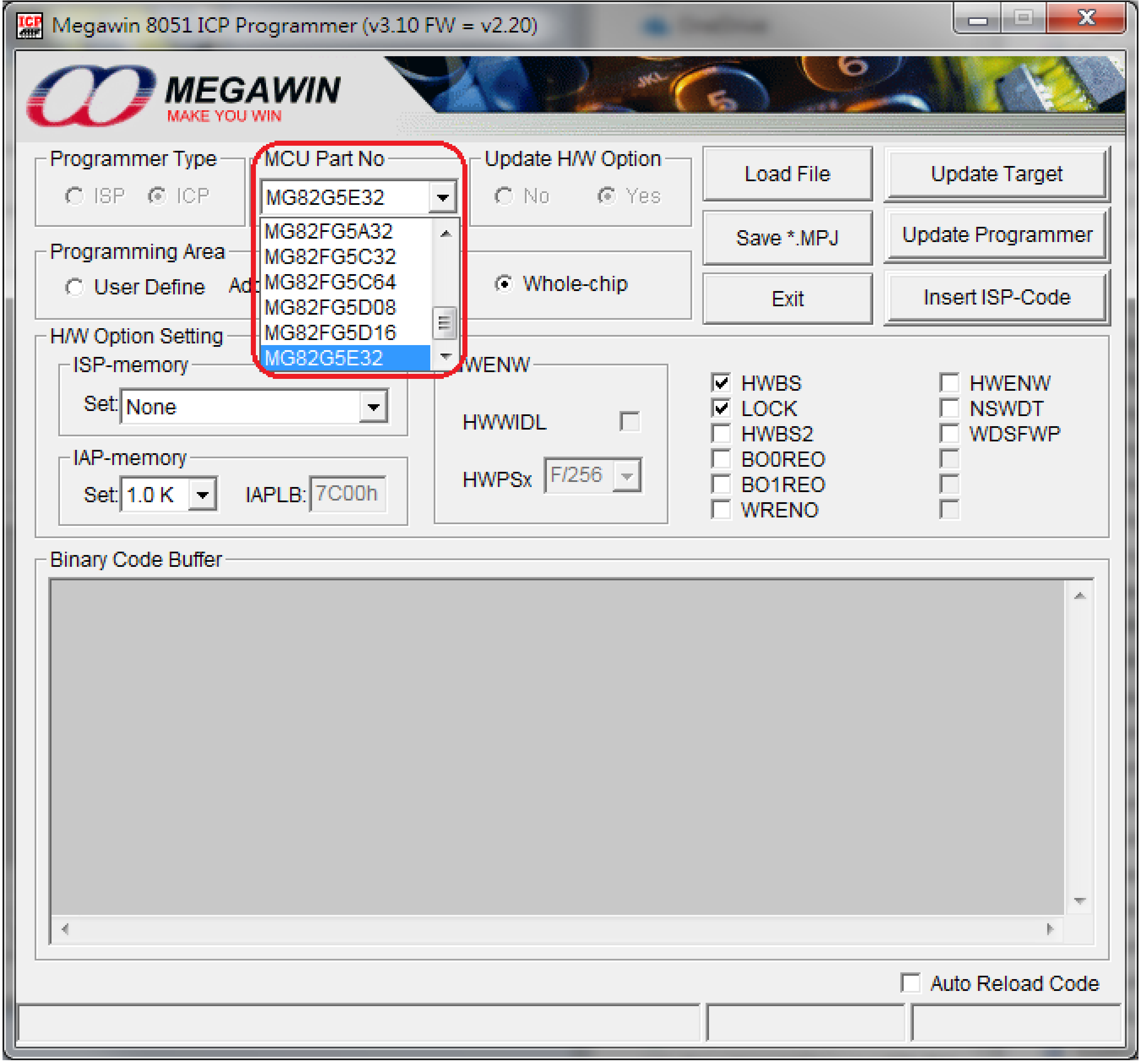
Attention: To program ICP application correctly by means 2, users have to open Project and Build first.



6.2.1 Update Programmer

Step 1: Choose a “MCU Part No.”

If users open ICP application by clicking Menu, Step 1 can be omitted. ICP application will choose MCU Part No. by Project automatically.



Megawin 8051 ICP Programmer (v3.10 FW = v2.20)

Programmer Type: ○ ISP ● ICP

MCU Part No: MG82G5E32

Update H/W Option: ○ No ● Yes

Programming Area: ○ User Define ● Whole-chip

H/W Option Setting

ISP-memory Set: None

IAP-memory Set: 1.0 K IAPLB: 7C00h

HWWIDL □

HWPSx F/256

☑ HWBS
☑ LOCK
□ HWBS2
□ BO0REO
□ BO1REO
□ WRENO

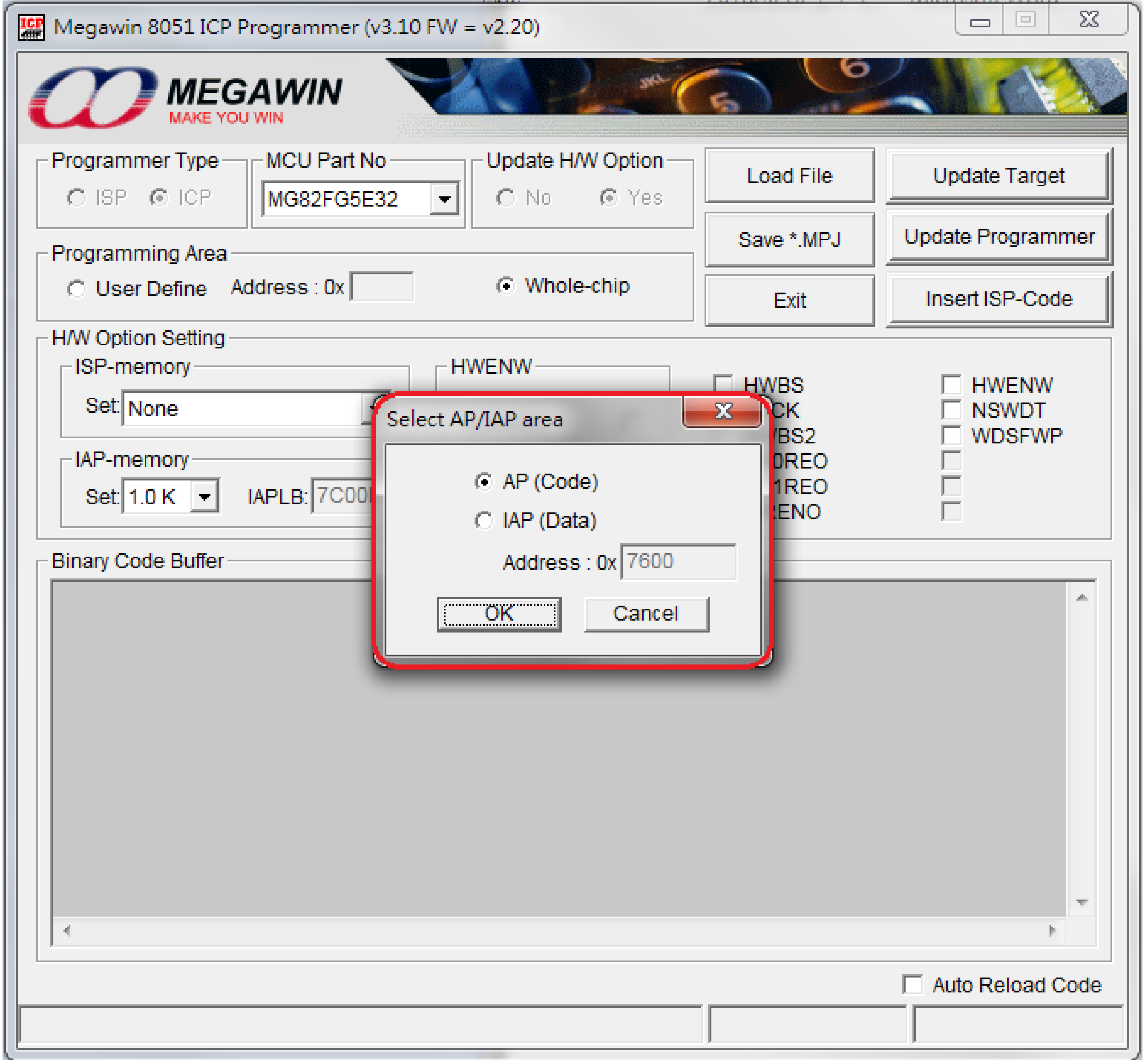
□ HWENW
□ NSWDT
□ WDSFWP

Binary Code Buffer

□ Auto Reload Code

Step 2: Click “Load File” and choose loading AP(Code) or IAP(Data). “Load File” can be clicked repeatedly to load different files. While loading IAP(Data), users have to key in Address. HEX and BIN data formats are supported for file loading.

If users open ICP application by clicking Menu, Step 2 can be omitted. ICP application will load Target file automatically.



Megawin 8051 ICP Programmer (v3.10 FW = v2.20)

Programmer Type: ○ ISP ● ICP

MCU Part No: MG82FG5E32

Update H/W Option: ○ No ● Yes

Programming Area: ○ User Define Address : 0x ● Whole-chip

H/W Option Setting

ISP-memory Set: None

IAP-memory Set: 1.0 K IAPLB: 7C00

Select AP/IAP area

● AP (Code)

○ IAP (Data)

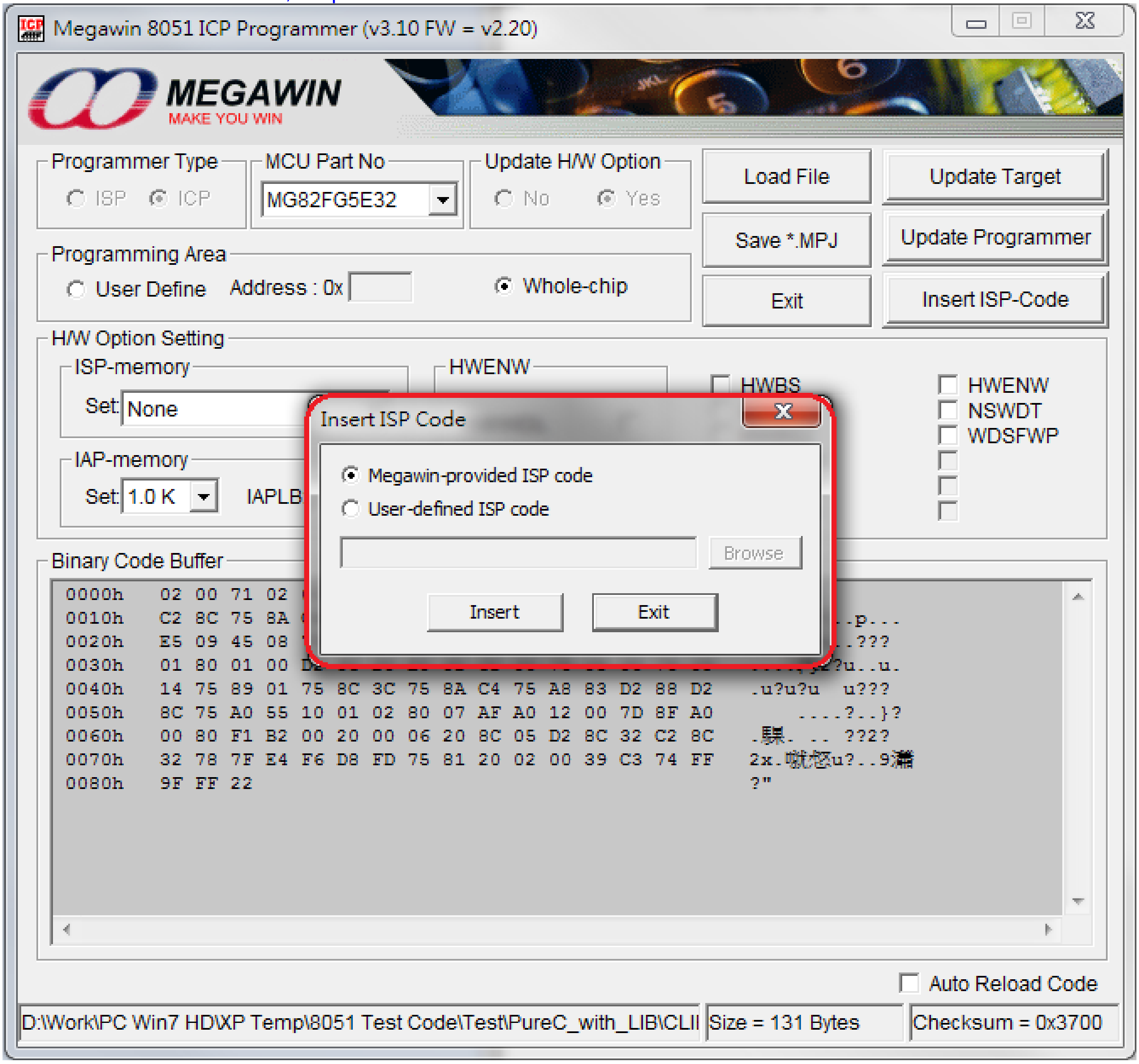
Address : 0x 7600

□ HWBS □ HWENW
□ ...CK □ NSWDT
□ ...BS2 □ WDSFWP
□ ...0REO □
□ ...1REO □
□ ...ENO □

□ Auto Reload Code

Step 3: Click “Insert ISP-Code” may choose to insert Megawin-provided ISP code or User-defined ISP code.

If ISP function is not needed, Step 3 can be omitted.



Megawin 8051 ICP Programmer (v3.10 FW = v2.20)

Programmer Type: ○ ISP ● ICP

MCU Part No: MG82FG5E32

Update H/W Option: ○ No ● Yes

Programming Area: ○ User Define Address : 0x ● Whole-chip

H/W Option Setting

ISP-memory Set: None

HWENW

☐ HWBS

☐ HWENW
☐ NSWDT
☐ WDSFWP

IAP-memory Set: 1.0 K

IAPLB

Insert ISP Code

● Megawin-provided ISP code
○ User-defined ISP code

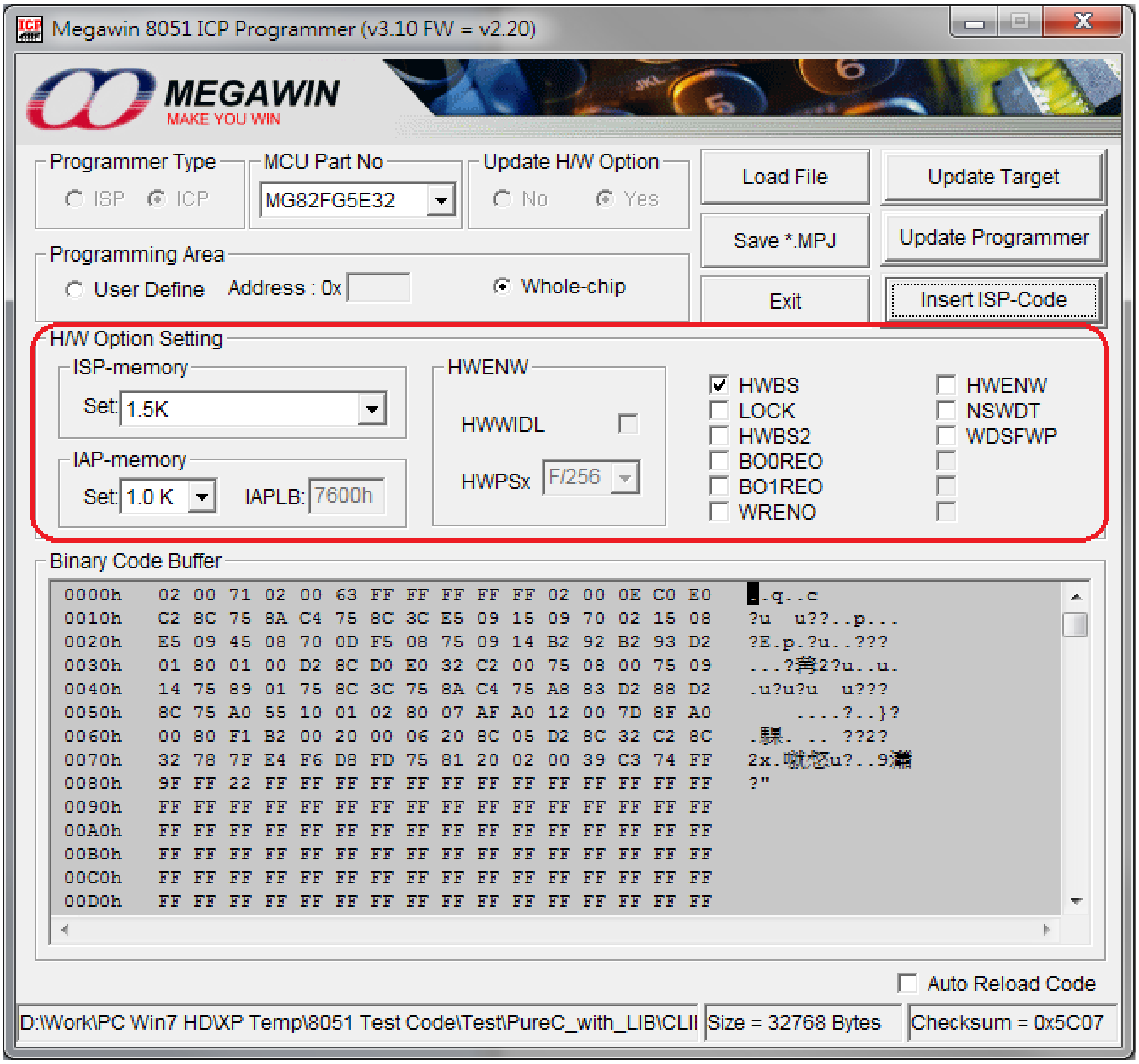
Binary Code Buffer

```
0000h   02 00 71 02
0010h   C2 8C 75 8A                                          ..p...
0020h   E5 09 45 08                                          ..???
0030h   01 80 01 00 D2                                       ?u..u.
0040h   14 75 89 01 75 8C 3C 75 8A C4 75 A8 83 D2 88 D2    .u?u?u  u???
0050h   8C 75 A0 55 10 01 02 80 07 AF A0 12 00 7D 8F A0       ....?..}?
0060h   00 80 F1 B2 00 20 00 06 20 8C 05 D2 8C 32 C2 8C    .騍. .. ????
0070h   32 78 7F E4 F6 D8 FD 75 81 20 02 00 39 C3 74 FF    2x.噈怼u?..9灩
0080h   9F FF 22                                           ?"
```

☐ Auto Reload Code

D:\Work\PC Win7 HD\XP Temp\8051 Test Code\Test\PureC_with_LIB\CLII | Size = 131 Bytes | Checksum = 0x3700

Step 4：H/W Option Setting



Megawin 8051 ICP Programmer (v3.10 FW = v2.20)

Programmer Type: ○ ISP ● ICP

MCU Part No: MG82FG5E32

Update H/W Option: ○ No ● Yes

Programming Area: ○ User Define Address : 0x ● Whole-chip

H/W Option Setting

ISP-memory Set: 1.5K

IAP-memory Set: 1.0 K IAPLB: 7600h

HWENW

HWWIDL □

HWPSx F/256

☑ HWBS
□ LOCK
□ HWBS2
□ BO0REO
□ BO1REO
□ WRENO

□ HWENW
□ NSWDT
□ WDSFWP
□
□
□

Binary Code Buffer

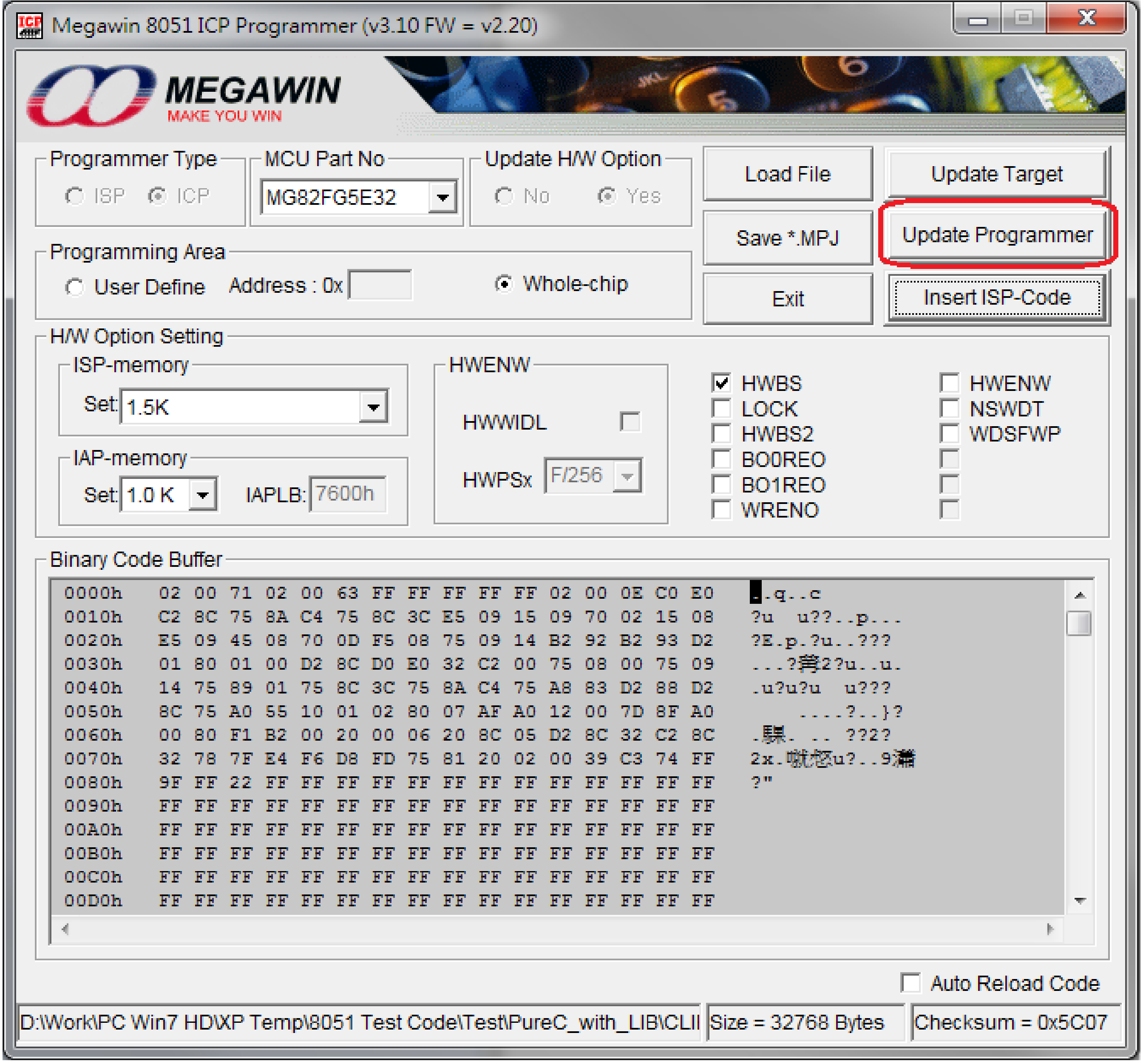
```
0000h   02 00 71 02 00 63 FF FF FF FF FF 02 00 0E C0 E0   ..q..c
0010h   C2 8C 75 8A C4 75 8C 3C E5 09 15 09 70 02 15 08   ?u  u??..p...
0020h   E5 09 45 08 70 0D F5 08 75 09 14 B2 92 B2 93 D2   ?E.p.?u..???
0030h   01 80 01 00 D2 8C D0 E0 32 C2 00 75 08 00 75 09   ...?霄2?u..u.
0040h   14 75 89 01 75 8C 3C 75 8A C4 75 A8 83 D2 88 D2   .u?u?u  u???
0050h   8C 75 A0 55 10 01 02 80 07 AF A0 12 00 7D 8F A0       ....?..}?
0060h   00 80 F1 B2 00 20 00 06 20 8C 05 D2 8C 32 C2 8C   .騏. .. ????
0070h   32 78 7F E4 F6 D8 FD 75 81 20 02 00 39 C3 74 FF   2x.啾怒u?..9瀟
0080h   9F FF 22 FF FF FF FF FF FF FF FF FF FF FF FF FF   ?"
0090h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00B0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00D0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

□ Auto Reload Code

D:\Work\PC Win7 HD\XP Temp\8051 Test Code\Test\PureC_with_LIB\CLII | Size = 32768 Bytes | Checksum = 0x5C07

Step 5: Click “Update Programmer” to download programming data to the ICE adapter.

“Update Programmer” function can be chosen only when connecting an ICE adapter (Only support TH065C or later versions).



Programmer Type: ○ ISP ● ICP

MCU Part No: MG82FG5E32

Update H/W Option: ○ No ● Yes

Programming Area: ○ User Define Address : 0x ● Whole-chip

H/W Option Setting

ISP-memory Set: 1.5K

IAP-memory Set: 1.0 K IAPLB: 7600h

HWENW: HWWIDL □ HWPSx F/256

☑ HWBS
□ LOCK
□ HWBS2
□ BO0REO
□ BO1REO
□ WRENO

□ HWENW
□ NSWDT
□ WDSFWP
□
□
□

Binary Code Buffer

```
0000h   02 00 71 02 00 63 FF FF FF FF FF 02 00 0E C0 E0   ..q..c
0010h   C2 8C 75 8A C4 75 8C 3C E5 09 15 09 70 02 15 08   ?u  u??..p...
0020h   E5 09 45 08 70 0D F5 08 75 09 14 B2 92 B2 93 D2   ?E.p.?u..???
0030h   01 80 01 00 D2 8C D0 E0 32 C2 00 75 08 00 75 09   ...?茾2?u..u.
0040h   14 75 89 01 75 8C 3C 75 8A C4 75 A8 83 D2 88 D2   .u?u?u  u???
0050h   8C 75 A0 55 10 01 02 80 07 AF A0 12 00 7D 8F A0     ....?..}?
0060h   00 80 F1 B2 00 20 00 06 20 8C 05 D2 8C 32 C2 8C   .騍. .. ???2?
0070h   32 78 7F E4 F6 D8 FD 75 81 20 02 00 39 C3 74 FF   2x.啨恖u?..9瀟
0080h   9F FF 22 FF FF FF FF FF FF FF FF FF FF FF FF FF   ?"
0090h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00A0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00B0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00C0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00D0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

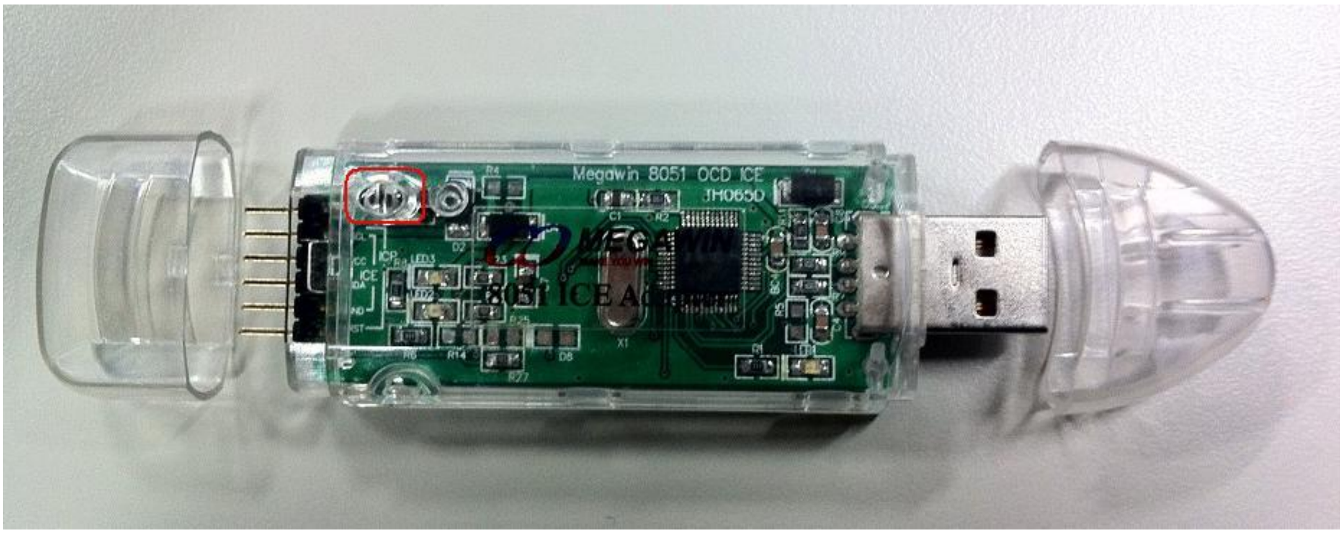
□ Auto Reload Code

D:\Work\PC Win7 HD\XP Temp\8051 Test Code\Test\PureC_with_LIB\CLII | Size = 32768 Bytes | Checksum = 0x5C07

6.2.2 Update Target

How to update the target? Users may

1. click “Update Target” to program on-line update, referring to steps 1 through 4 of 6.2.1 Update Programmer, or
2. click “Downloading” of ICE adapter to program off-line update, referring to 6.2.1 Update Programmer.



7 Special Notes

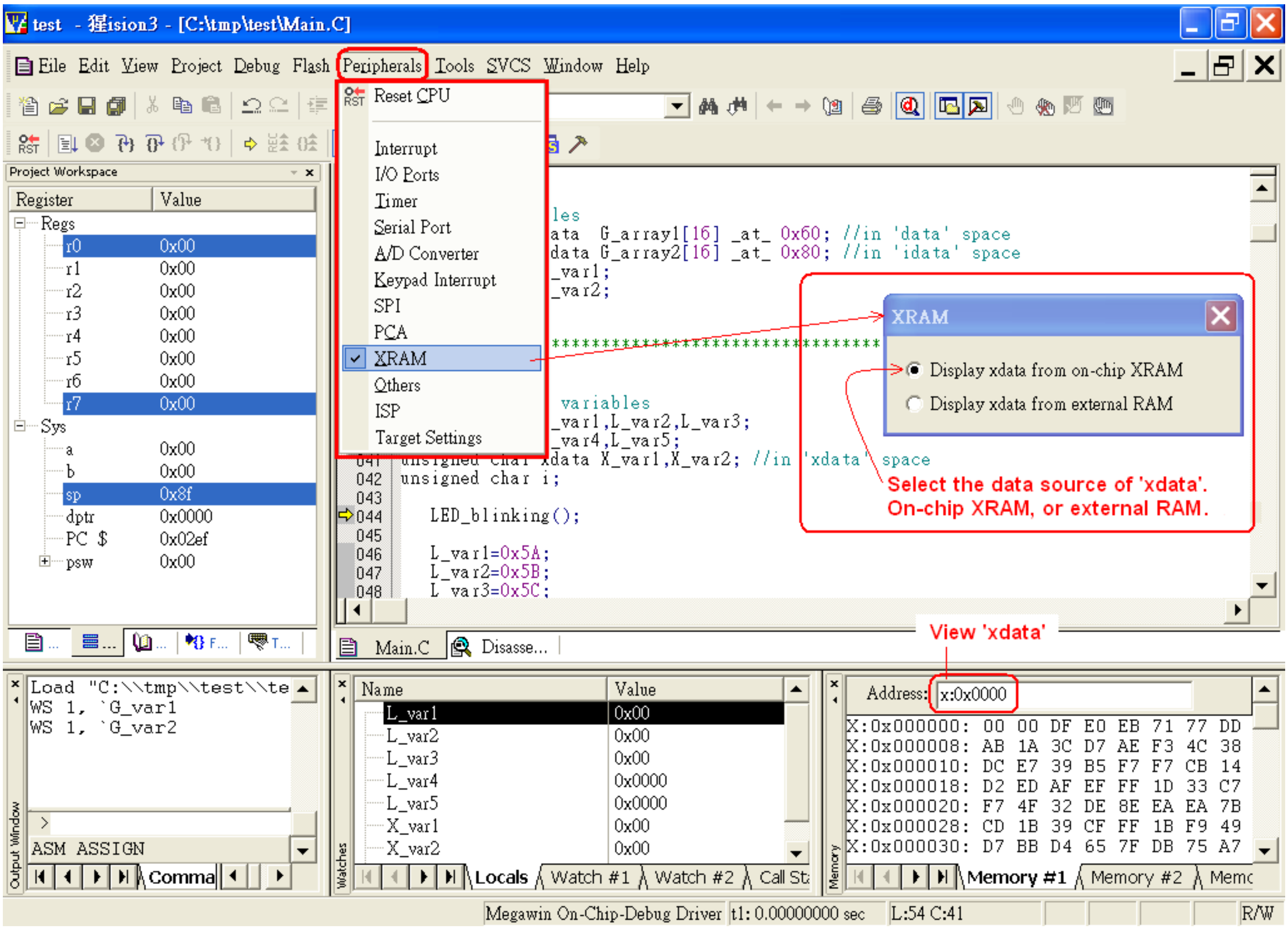
7.1 Register Definition Files

Register definition files *REG_MPC82G516.INC* and *REG_MPC82G516.H* define all Special Function Registers (SFRs) and bit-addressable control/status bits. They are installed into the default search path used by the Keil 8051 IDE software when you do the Software Setup (described in *Section 2*). Therefore, when using the Keil 8051 tools, you can include them by *$INCLUDE (REG_MPC82G516.INC)* and *#include <REG_MPC82G516.H>*. It is not necessary to copy a register definition file to each project's file directory.

7.2 On-chip XRAM and External Data Memory

Megawin 8051 devices provide on-chip XRAM (eXpanded RAM), which is accessed with the same instructions as the traditional external data memory. The size of on-chip XRAM in MPC82G516 is 1024 bytes with addresses 0x0000 to 0x03FF. That is, the address space of on-chip XRAM overlaps that of the external data memory. So, there must be a control bit used to distinguish these two physical memories during access. The ERAM bit (bit-1 in register AUXR) plays this role. *Because the C51 Compiler won't take care which physical memory the user wants to access, the user must manually clear this bit before accessing on-chip XRAM and set this bit before accessing external data memory*. By default, this control bit is '0' after powered on or chip reset for on-chip XRAM accessing.

The C51 Compiler offers two different memory types that access external data: *xdata* and *pdata*. (The *xdata* memory specifier refers to any location in the 64K-byte address space of external data memory. The *pdata* memory type specifier refers to only one page or 256 bytes of external data memory.) When the user want to view the variables declared by *xdata* or *pdata* directly in the Memory Window rather than in the Watch Window, he should select "**Display xdata from on-chip XRAM**" or "**Display xdata from external RAM**" under menu **Peripherals- XRAM**, as shown in the following figure.



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x8f
dptr	0x0000
PC $	0x02ef
psw	0x00

```
les
ata  G_array1[16] _at_ 0x60; //in 'data' space
data G_array2[16] _at_ 0x80; //in 'idata' space
_var1;
_var2;

***********************************

variables
_var1,L_var2,L_var3;
_var4,L_var5;
unsigned char xdata X_var1,X_var2; //in 'xdata' space
unsigned char i;

    LED_blinking();

    L_var1=0x5A;
    L_var2=0x5B;
    L_var3=0x5C;
```

XRAM

● Display xdata from on-chip XRAM
○ Display xdata from external RAM

Select the data source of 'xdata'.
On-chip XRAM, or external RAM.

View 'xdata'

```
Load "C:\\tmp\\test\\te
WS 1, `G_var1
WS 1, `G_var2
>
ASM ASSIGN
```

Name	Value
L_var1	0x00
L_var2	0x00
L_var3	0x00
L_var4	0x0000
L_var5	0x0000
X_var1	0x00
X_var2	0x00

Address: x:0x0000

```
X:0x000000: 00 00 DF E0 EB 71 77 DD
X:0x000008: AB 1A 3C D7 AE F3 4C 38
X:0x000010: DC E7 39 B5 F7 F7 CB 14
X:0x000018: D2 ED AF EF FF 1D 33 C7
X:0x000020: F7 4F 32 DE 8E EA EA 7B
X:0x000028: CD 1B 39 CF FF 1B F9 49
X:0x000030: D7 BB D4 65 7F DB 75 A7
```

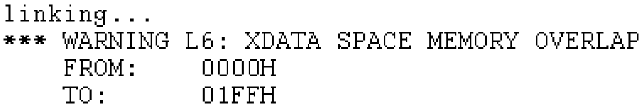
Megawin On-Chip-Debug Driver | t1: 0.00000000 sec | L:54 C:41 | R/W

The following example code shows how to use both on-chip XRAM and external RAM in an application. To view G_array1[ ], select “Display xdata from on-chip XRAM”; and to view G_array2[ ], select “Display xdata from external RAM”.

Example of using both on-chip XRAM and external RAM

```
unsigned char xdata G_array1[512] _at_ 0x0000;  // in 'xdata' space, will use on-chip XRAM
unsigned char xdata G_array2[512] _at_ 0x0000;  // in 'xdata' space, will use ext. RAM
unsigned int i;

   AUXR&=0xFD; //clear AUXR.1 for on-chip XRAM
   for (i=0; i<512; i++) G_array1[i]=0x5A;  // fill XRAM with 0x5A

   AUXR|=0x02; //set AUXR.1 for external RAM
   for (i=0; i<512; i++) G_array2[i]=0xA5;  // fill ext. RAM with 0xA5
```

Note that there will be a linking warning listed below. However, it doesn’t matter because we intentionally declare G_array1 and G_array2 in the same address space. In fact, we access to the different physical memory controlled by bit-1 of AUXR.



```
linking...
*** WARNING L6: XDATA SPACE MEMORY OVERLAP
    FROM:     0000H
    TO:       01FFH
```

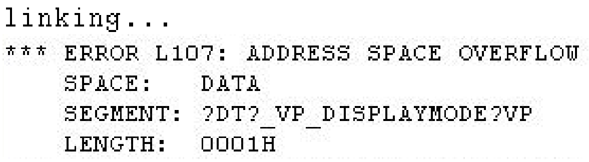
7.3 Code Optimization and Source-Level Debugging

As shown in the following source code, the C51 compiler won’t generate any machine code for “*L_var1=0x38;*” because this statement becomes meaningless due to its following statement “*L_var1=0xC7;*”. For code optimization, “*L_var1=0x38;*” will be optimized out unless the code optimization is disabled as described in *Section 4.4*.

```
unsigned char L_var1;

   L_var1=0x38;  // ! Note: this statement may be optimized out by the C51 compiler
   L_var1=0xC7;
```

So, during source-level debugging, *L_var1* will never show *0x38* but may show a random number when this statement is just executed. In fact, there in no machine code for this statement. The user should pay attention to it!

Sometimes, for debugging purpose, the user may disable the compiler’s code optimization. Note that once the compiler’s code optimization is disabled, there may be some linking errors which won’t occur when the code optimization is enabled. For example, refer to the following linking error message, it means the variables you use exceed the RAM an MCU has. To make this error disappear, the only way is to enable the compiler’s code optimization to let the compiler make more efficient use of the RAM.



```
linking...
*** ERROR L107: ADDRESS SPACE OVERFLOW
    SPACE:   DATA
    SEGMENT: ?DT?_VP_DISPLAYMODE?VP
    LENGTH:  0001H
```

7.4 "for-Loop" and Source-Level Debugging

The following two statements are fully the same for the 8051 CPU to execute them. During source-level debugging, there is no problem to apply *Step* action on Statement 1. However, it will take so much time if the user apply *Step* action on Statement 2. We think it is caused by unknown processing in the Keil debugger function. Before we getting the reply from Keil, we suggest using Statement 1 instead of Statement 2 in the source code if you want to do step-debugging in such statement. Another solution for Statement 2 is: move cursor to Line2 and click left key, then click *Run-to-Cursor* button to fly over Line 1.

Statement 1:

```
Line1:  for (i=0; i<16; i++) {
Line2:        G_array1[i]=i+0x60;
Line3:         }
```

Statement 2:

```
Line1:  for (i=0; i<16; i++) G_array1[i]=i+0x60;
Line2:  …
Line3:  …
```

7.5 Hardware Option Requirements During Debugging

There are two requirements regarding the hardware option in the *dScope-D*ebugger mode:

Requirement 1: The debugged chip must be in un-locked state

It is because if the debugged chip is locked, the downloading of the user's application code in the *dScope*-Debugger mode will cause the chip to be whole-chip erased, and therefore all the chip's hardware options will be disabled. Thus the debugged chip may not work well owing to losing its original hardware options. For example, for a locked chip with IAP-memory configured, after downloading the user's application code when entering the *dScope*-Debugger mode, its IAP-memory will disappear (i.e., disabled). So, the chip cannot work well.

Requirement 2: The ISP function of the debugged chip must be disabled

It is because if the ISP function is enabled, the debugged chip will always boot from the ISP-memory and run the ISP-code when the chip receives the *Reset* command in the *dScope-D*ebugger mode. It will cause a problem. That is, the code the MCU runs (i.e., the ISP-code) is different from the code of the opened Keil project (i.e., the user's application code). So, during debugging, the user needs to disable the ISP function by having the hardware option *HWBS* disabled temporarily.

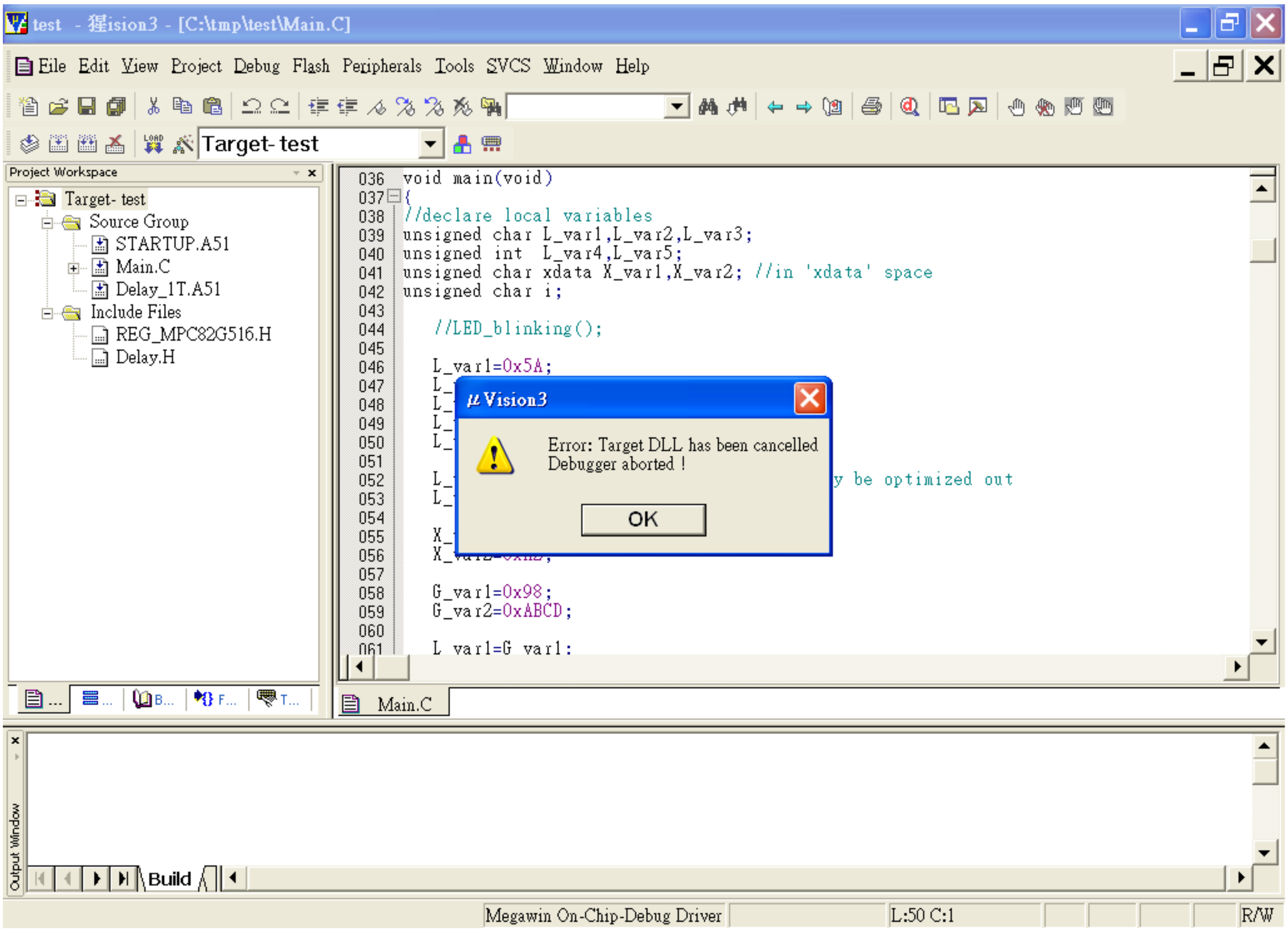
Note:
After the application code is debugged completely, the user may use the "Megawin 8051 ICP Programmer" to restore the original hardware option.

7.6 Error Message

There will be an error message “**Error: Target DLL has been cancelled. Debugger aborted !**” shown in following figure if:

(1) ICE adapter hardware fails, or
(2) Target MCU doesn’t work (for example, not powered on), or
(3) Cable error or improper connection between ICE adapter and the Target MCU.

Once the error message pops out, click “**OK**”. Then, check the above possible causes to solve the problem.



```
void main(void)
{
//declare local variables
unsigned char L_var1,L_var2,L_var3;
unsigned int  L_var4,L_var5;
unsigned char xdata X_var1,X_var2; //in 'xdata' space
unsigned char i;

    //LED_blinking();

    L_var1=0x5A;
    L_
    L_
    L_
    L_

    L_                                              y be optimized out
    L_

    X_
    X_

    G_var1=0x98;
    G_var2=0xABCD;

    L_var1=G_var1;
```

µVision3

Error: Target DLL has been cancelled
Debugger aborted !

OK

7.7 Properly Connect the ICE Adapter to a Host

The data transfer rate of the ICE adapter will be slowed down severely if it is connected to a host via a USB2.0 hub. So, to speed up the downloading when clicking *dScope* button to enter the debugger mode, the user had better directly plug the ICE adapter into the host's USB port, as shown in Figure 6.7.1. Don't plug into a hub and then to the host, as shown in Figure 6.7.2.

Figure 6.7.1 Directly plug into the host's USB port

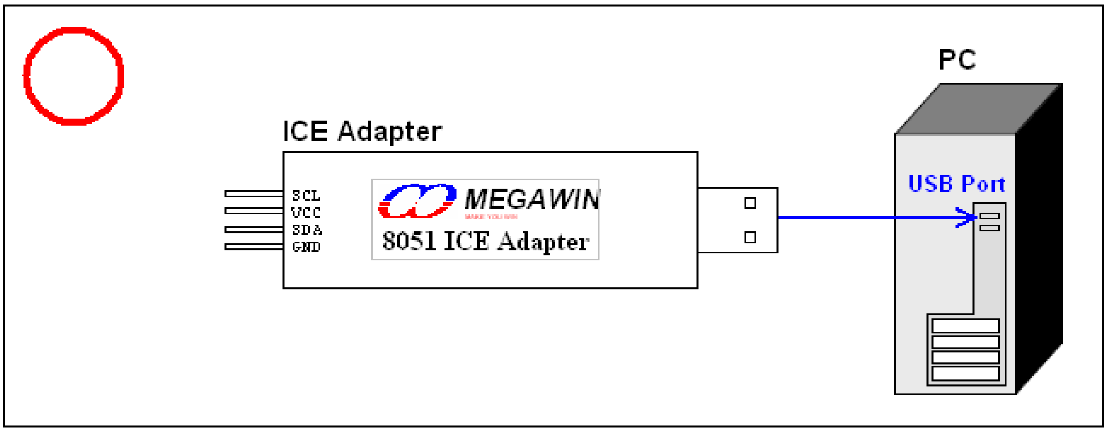
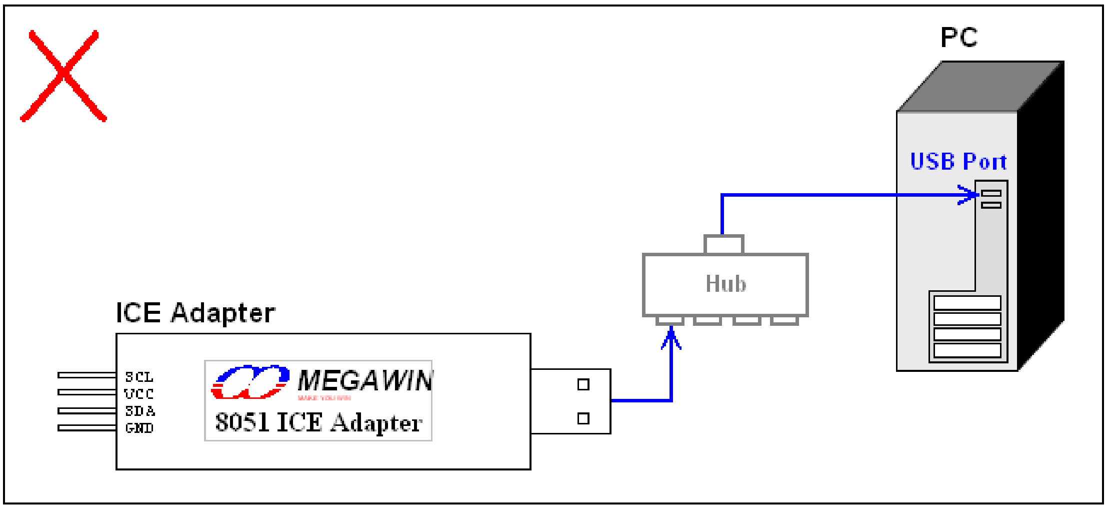


Figure 6.7.2 Don't plug into a hub and then to the host's USB port



Revision History

Revision	Description	Date
v1.00	The first release for beta-site test.	2007/08/15
v1.01	Add notes when installation fails. (Section 3.2)	2007/08/24
v1.02	Change to manually specify the installation path of the Keil software. (Section 3.2)	2007/08/27
v2.00	Add the notification of default installation path of Keil 8051 IDE software. (Section 3.2)	2007/08/29
	Update the Keil IDE Setup. (Section 4.4)	2007/10/08
	Update the Special Notes. (Section 6)	2007/10/08
	The formal released version.	2007/10/08
v2.10	(1) Improve the defect of breakpoint setting. (2) Fix the bug of wrong erasing range when downloading the application code.	2007/12/26
V2.20	(1) Update the data base for all series of MCU in Driver Installer. (2) Removed the function of detecting the ICE adapter when install Driver.	2009/02/27
V2.21	Change the folder name of Driver Install to Database install	2009/04/01
V2.30	(1) Supported MG82FL(E)532 and MG82FL(E)564 (2) Supported ICP function	2010/05/10
V2.31	Update “Database Installer “	2010/05/21
V2.32	Support uVision4	2010/06/02
V2.33	Update “IcpProgrammer.exe” in Database Installer	2010/08/25
V2.40	Supported MG84FG516	2011/05/02
V2.41	Update “IcpProgrammer.exe” in Database Installer	2011/06/01
V2.50	Support Off-Line Mode programming	2011/10/20
V2.51	Support H/W ver.TH065E to prevent to damage the MG84FG516	2012/04/01
V2.52	Fix the bug on ICP function for MG84FG516	2012/05/01
V2.53	Update “IcpProgrammer.exe” in Database Installer	2012/05/15
V2.54	(1) Supported “Maximum Counter” in Off-Line Mode programming (2) Supported “ Serial Number “ in Off-Line Mode programming (3) Improve the performance on Off-Line Mode programming.	2012/07/12
V2.55	Fix the bug on ICP function	2012/09/28
V2.56	Fix the bug on ICE function	2012/10/08
V2.60	(1) Supported “MPJ” file (2) Database support MG86FL(E)104 and MG86FL(E)508 (3) Supported MG82FG5A64 (4) Update “ warning message “ when OCD ICE in update processing	2012/12/10
V2.61	(1) Fix a bug for MG84FG516 at access P6M0 in debug mode (2) Update “ Megawin.dat “	2013/01/10
V2.62	Update “IcpProgrammer.exe” in Database Installer	2013/01/14
V2.63	Supported MG82FG5A32	2013/06/27
V2.64	Update the Hardware Setup. (Section 2)	2013/09/27
V2.70	(1) Supported MG82FG5B(32/16) (2) Supported MG20FL(E)809	2103/11/15
V2.71	Supported MG82FG5B(24/08)	2104/04/09
V2.72	Update H and INC files in H and INC folder	2014/05/15
V2.90	Supported MG82FG5C(64/32)	2015/04/15

V2.91	Update “IcpProgrammer.exe” in Database Installer	2015/05/21
V2.92	Update “IcpProgrammer.exe” in Database Installer	2015/05/22
V2.93	Update “MegawinOCD.dll” in Database Installer	2015/08/24
V2.94	Update “MegawinOCD.dll” in Database Installer	2015/12/14
V2.95	Support “ Auto Reload Code “ in IcpProgrammer	2016/09/01
V2.96	Update “IcpProgrammer.exe” in Database Installer	2015/09/26
V3.00	Supported MG82FG5D(08/16)	2017/06/09
V3.01	Update “IcpProgrammer.exe” in Database Installer	2017/11/16
V3.02	Update “MegawinOCD.dll” in Database Installer	2017/12/28
V3.03	Add HexEdit.dll in Database Installer and modify IcpProgrammer.exe	2018/01/23
V3.04	Modify the sample codes	2018/03/26
V3.05	Supported MG82G5E32	2018/05/10
V3.08	Fix Messages when reset to AP on ICP Programmer	2018/09/10
V3.09	Modify SFR’s “PDTCR” to “PDTCRA”	2018/11/07
V3.10	Update “MegawinOCD.dll”, Fix bug	2018/12/14
V3.11	Fix ICP Programmer bug, Off-line program error	2019/01/08