CMOSTEK

AN210

CMT2380F16 OCD ICE User Guide

Overview

This document provides the user guide for CMT2380F16 OCD debugger. Employed a 8051 core, the CMT2380F16 is a high performance transceiver integrated wireless MCU. The product is part of the CMOSTEK NextGenRF^{TM} product family which covers a complete product line consisting of transmitters, receivers, transceiver, etc. suitable for short-range wireless communication applications.

The product models covered in this document are shown in the below table.

Table1. Product Models Covered in This Document

Product Model	Frequency Range	Modulation Type	Tx Power	Sensitivity	Chip Properties	Packaging
CMT2380F16	127 - 1020 MHz	OOK/(G)FSK	+20 dBm	-120 dBm	Wireless MCU	QFN48

Table of Contents

1 Introduction

1.1 Features

- Support of OCD (On-Chip-Debug) function
- Built-in real-time debugging circuit in the 8051 core of the CMT2380F16
- Dedicated 2-pin serial interface for OCD, no system pin occupied
- Being compatible with Keil debug and simulation IDE interface for 8051
- Connecting to host PC via USB
- Supports of debug functions such as reset, full speed run, stop, step run, etc.
- Programmable breakpoints, up to 4 breakpoints can be inserted simultaneously
- Powerful debug windows such as register window, disassembly window, variable watch window and memory watch window.

1.2 Description

The OCD ICE for CMT2380F16 is a powerful development tool providing built-in real-time debug functions through using the on-chip-debug technique. Users perform development and debug directly in OCD ICE with no need for development boards or extra adapters as required in the traditional 8051 ICE. Users just need to reserve a 5-pin connector (VCC, OCD_SDA, OCD_SCL, RST and GND) used by the dedicated OCD interface:

Moreover, it offers direct access to Keil 8051 IDE software UI for user program debug with the utilization of the dScope-Debugger of Keil IDE, gaining full Keil IDE advantages.

Notes:

1. Keil is the trade mark of *Keil Elektronik Gmbh and Keil Software, Inc*., and Keil 8051 IDE software is popular software used in embedded system development.

2 Hardware Setup

When debugging, users need to connect the target system to a PC via the ICE adapter, as shown in the below figure. The ICE adapter is a bus-powered USB device with no need for power adapter.

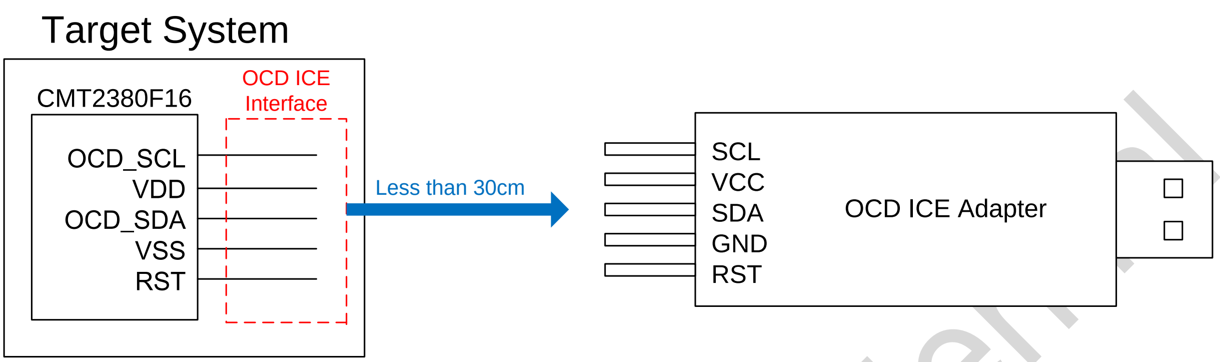

Figure 1. Hardware Setup

Pin numbers of the OCD ICE interface are as follows.

Table 2. Pin Numbers of OCD ICE Interface

Product Model	Package	OCD_SCL	OCD_SDA	RST
CMT2380F16	QFN-48	10	11	38

3 Software Setup

This chapter discusses the software settings before using the OCD ICE.

3.1 Driver Installation for ICE Adapter

Plug-in the ICE adapter into any USB port of a host PC directly with no need for driver installation.

3.2 Add CMT2380F16 Chip Information in Keil 8051 IDE

Plug-in the ICE adapter into the USB port of a host computer then execute *Setup.exe* in the folder *Database Installer* to add the chip information of CMT2380F16 into the Keil 8051 IDE μVision2, μVision3 or μVision4.

In the Database Installer window, perform *Add* operation following the steps as below.

Step 1, click *Browse* button to specify the Keil installation folder (default as *C:\KEIL*) in the target PC.

Step 2, click *Install* button to start installing the CMT2380F16 chip information into the Keil software.

The installation procedure is shown in the below figure.

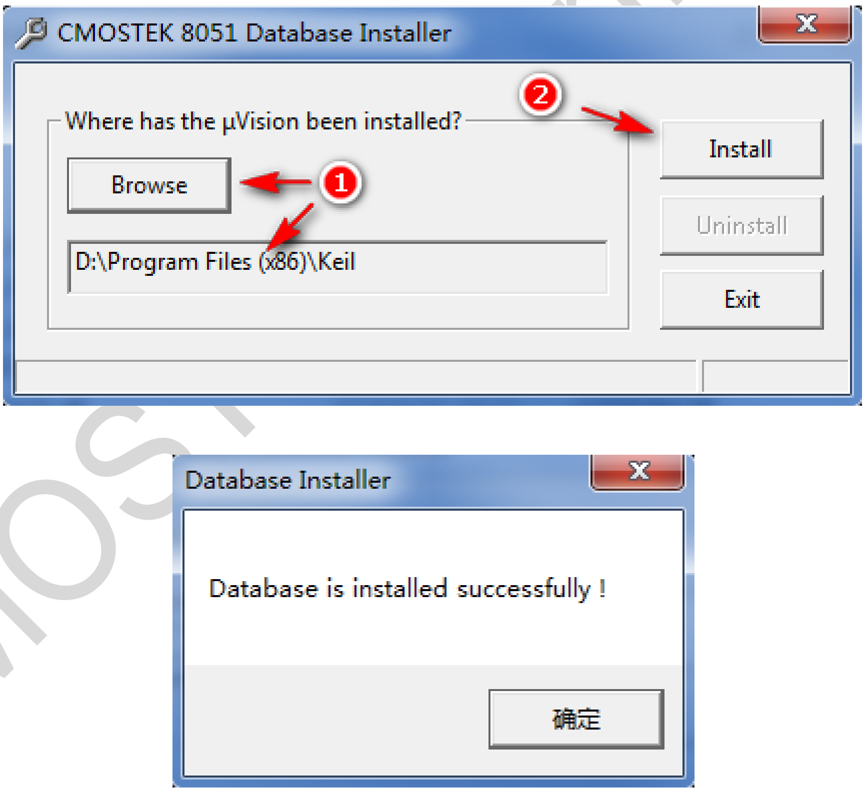

Figure 2. Installation Procedure

4 Keil IDE Setup

Before using the dScope-Debugger function of Keil IDE, users need to have some related settings. Open the µVision project to be debugged, right click the button inside the red circle in the below figure, then click the *Target Options* menu as shown in the below figure.

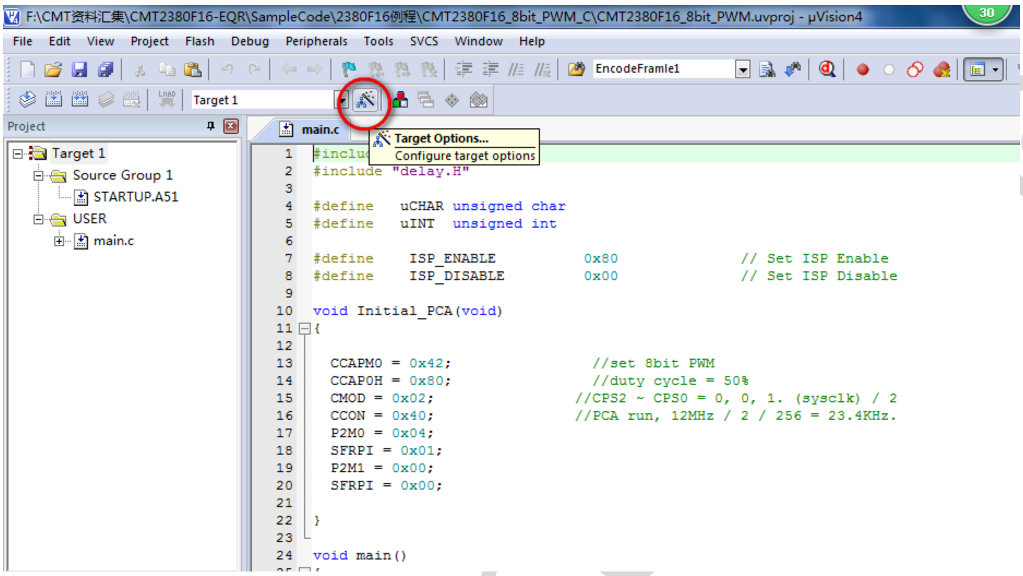

Figure 3. Keil IDE Setup

4.1 Device Settings

In the *Device* tab, select *CMOSTEK Device Database* and the target product model as shown in the below figure.

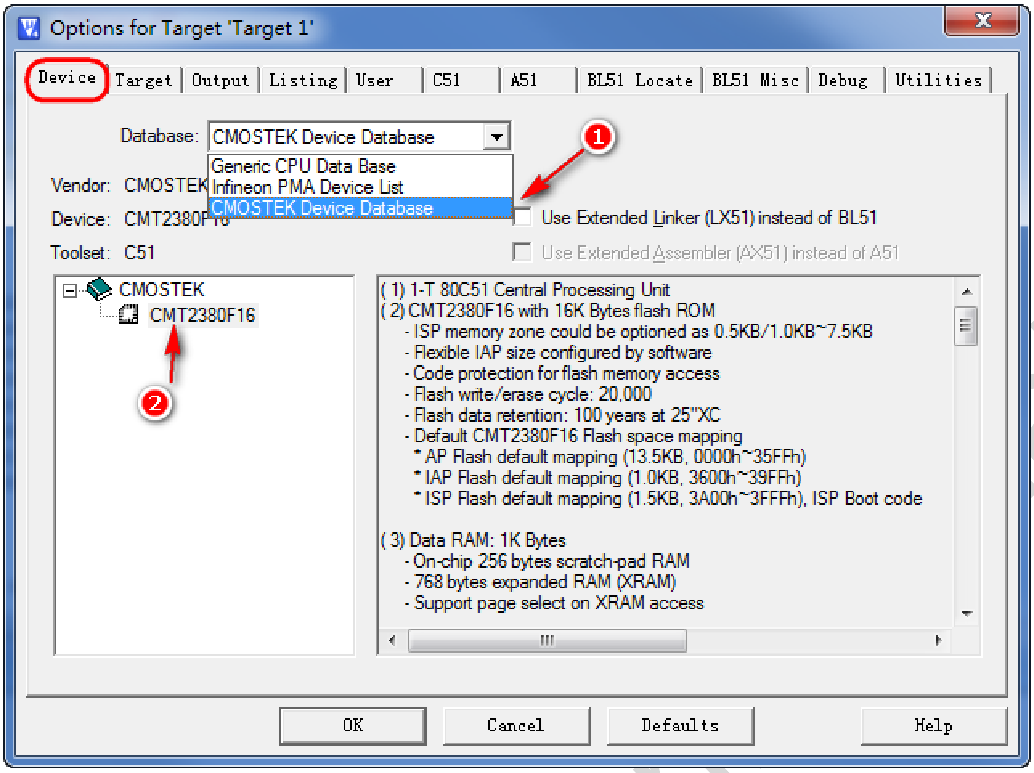

Figure 4. Device Settings

4.2 Target Settings

Tick on *Use on-chip ROM* and *Use on-chip XRAM* as shown in the below figure.

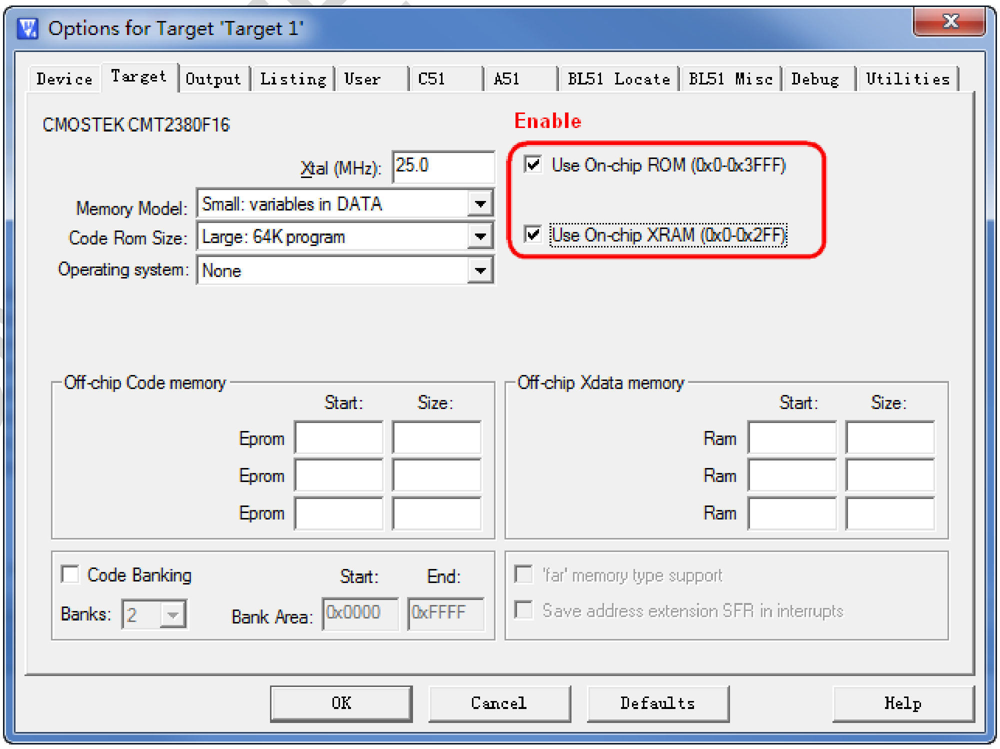

Figure 5. Target Settings

4.3 Output Settings

Tick on *Debug Information* as shown in the below figure to make sure an OMF file (Object Module Format) for source-level debugging being generated for ICE debug.

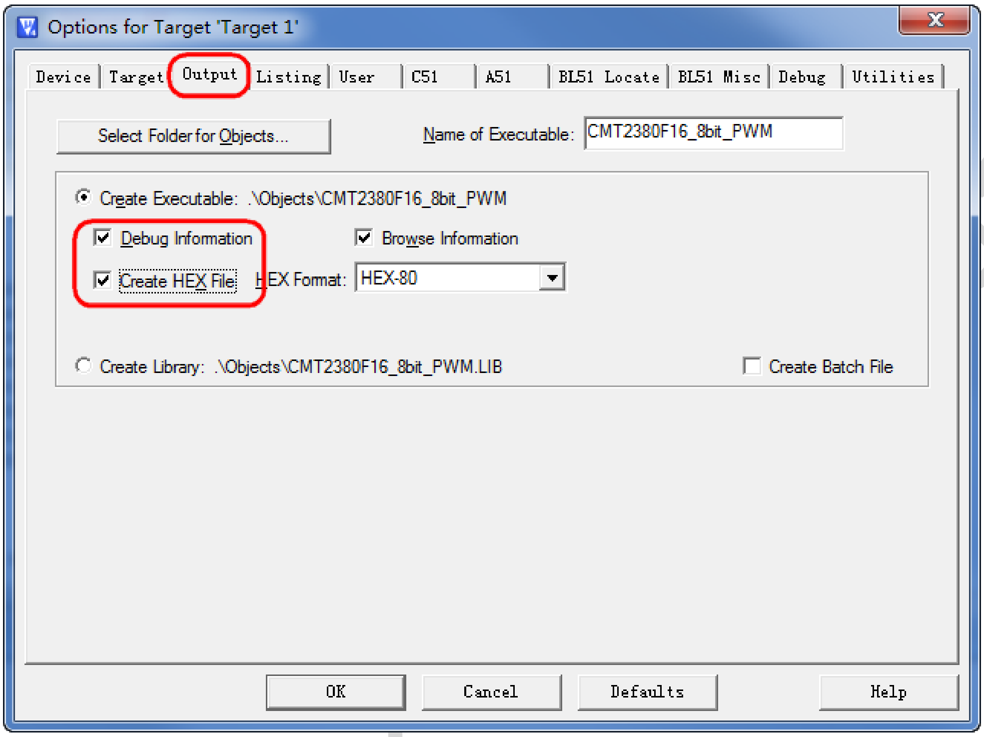

Figure 6. Output Settings

4.4 C51 Settings

Disable the code optimization by selecting *0: Constant folding* in textbox *Level*. Refer to Chapter 6.3 for more details.

Notes:

1. This setting is optional.

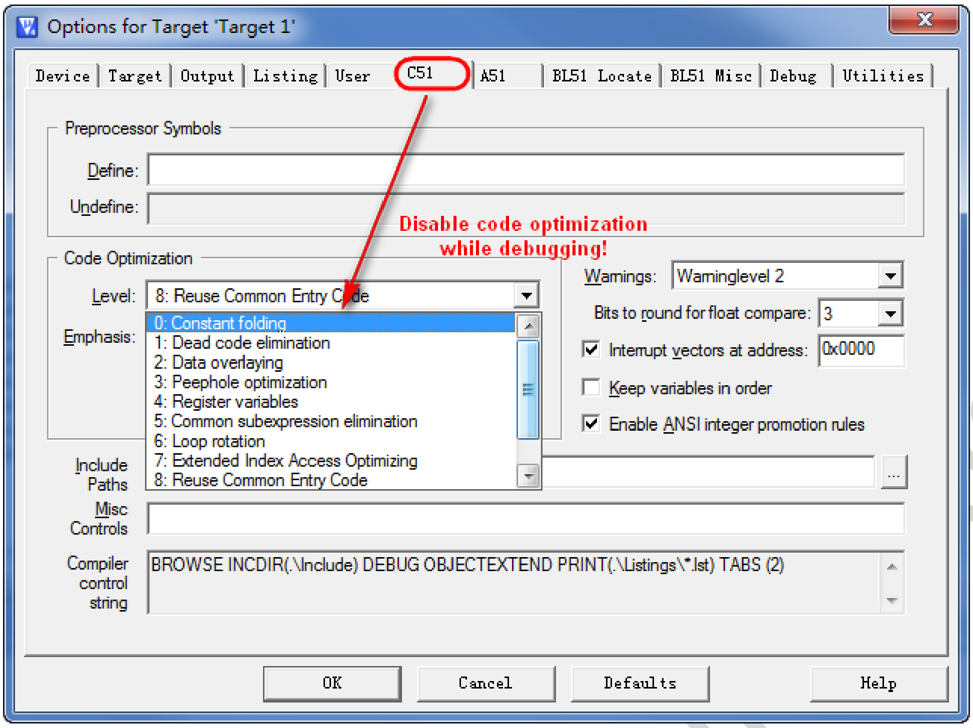

Figure 7. C51 Settings

4.5 Debug Settings

Select *On-Chip-Debug Drive* and tick on *Load Application at Startup* as well as all the cache option related parameters as shown in the red rectangle in the below figure.

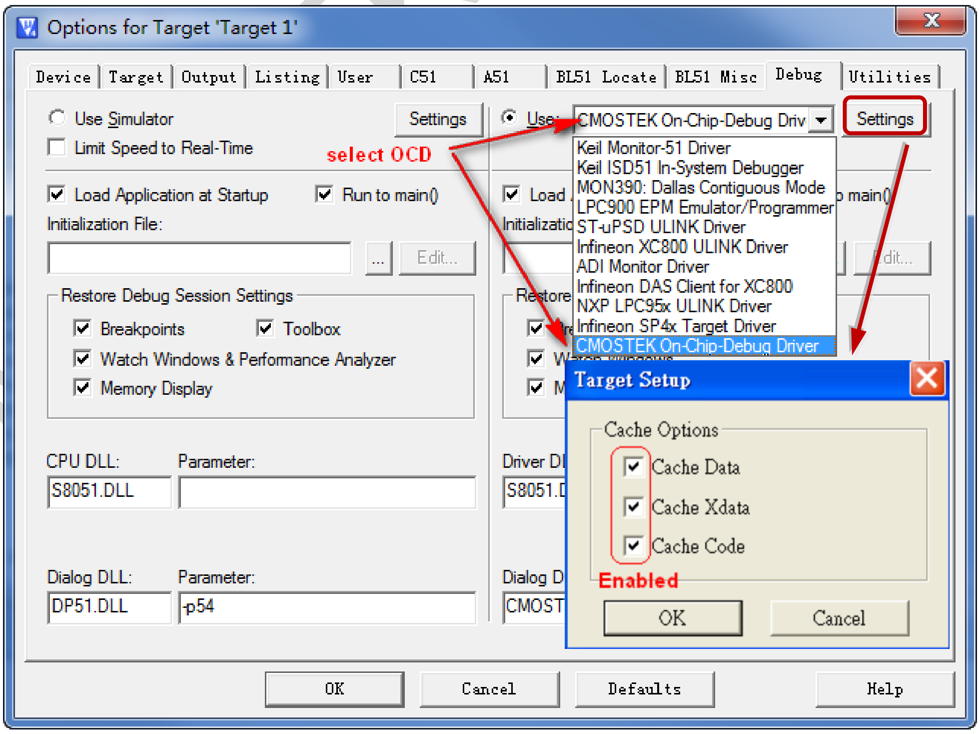

Figure 8. Debug Settings

4.6 Utilities Settings

Always tick off *Update Target before Debugging* as *Load Application at Startup* has been ticked on as shown in Figure 8 in Chapter 4.5. The driver displayed in *Use Target Driver for Flash Programming* may be different from *Silicon Laboratories C8051Fxxx uVision* due to users' installation of different drivers. Users can select any item in the drop-down list of *Use Target Driver for Flash Programming or* just leave it empty.

Notes:

1. These settings are not applicable to μVision2.

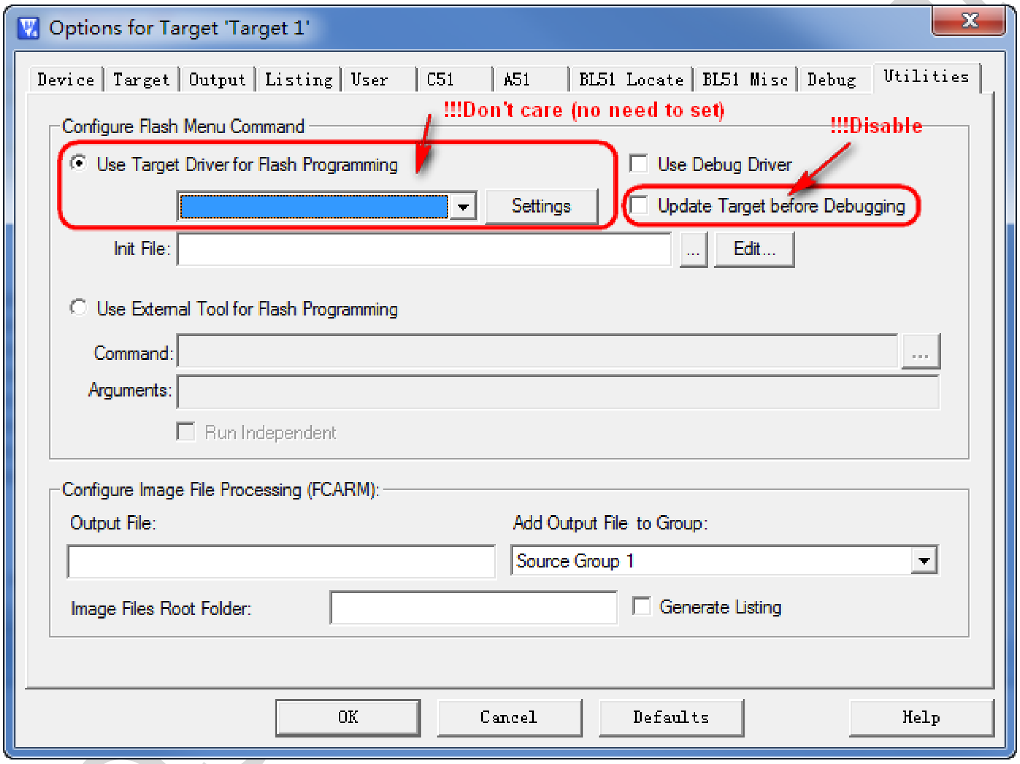

Figure 9. Utilities Settings

5 Start Debug

Users can start debug in μVision after the settings in chapter 2, 3 and 4 complete. .

5.1 Start dScope-Debugger Function

After completing project settings successfully, click *dScope* button to access the Keil IDE debug mode. When clicking *dScope* button, the user program will be downloaded into the CMT2380F16 as shown in the below figure, which will take some time.

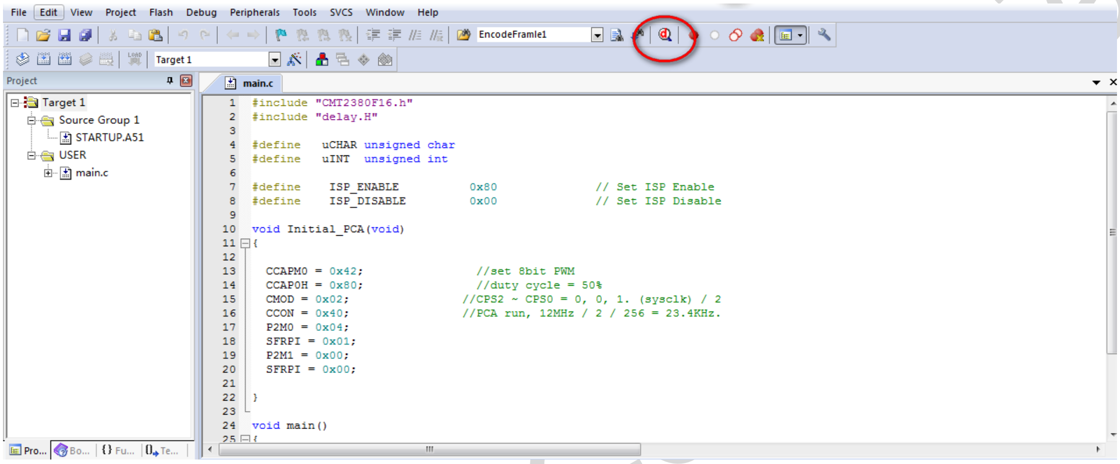

Figure 10. Start dScope-Debugger Function

5.2 Debug Environment Introduction

Generally 4 basic windows are used in the debug environment including register window, disassembly window, variable watch window and memory watch window, which are described as follows.

- Register window

This window shows the current register values (R0~R7), the system register values (A, B, SP, DTPR and the Program Counter) and the program status word (PSW). The blue background display of a register represents the register value now being changed by the instruction executed currently.

- Disassembly window

This window is opened by default when users accessing the debug mode. It shows the corresponding assembly code of the current program.

- Variable watch window

When *Locals* tab is selected, the window shows the local variables declared in the main() function. As for global variables, click Watch #1 or Watch #2, press <F2> key, then enter the variable name to view them. The blue background display of a variable represents the variable value now being changed by the instruction executed currently.

- Memory watch window

This window shows the contents in the memory located in the data/idata/xdata/code memory space. The available commands includes d:0x00~d:0xFF, i:0x00~i:0xFF, x:0x0000~x:0xFFFF and c:0x0000~c:0xFFFF. Users can view contents of any of these 4 types of memory spaces by entering the corresponding command.

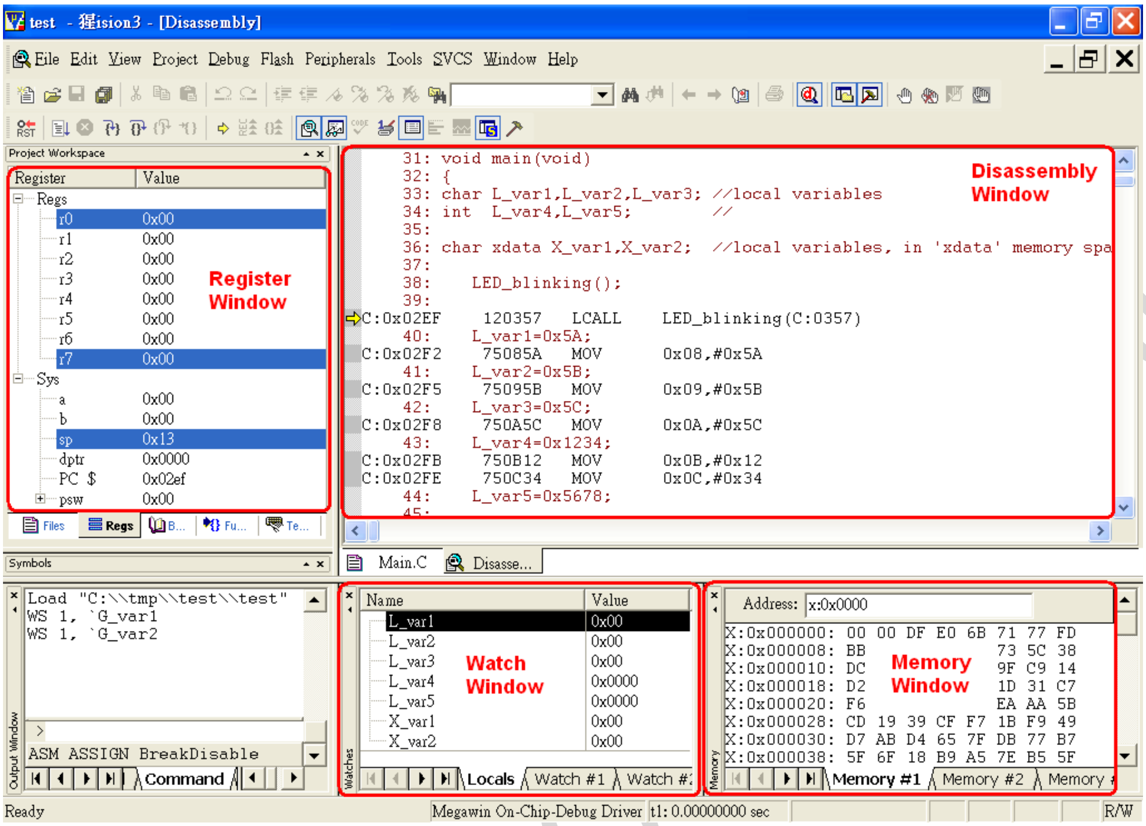

Figure 11. Debug Environment

5.2.1 Reset, Run, Stop, Step and Run-to-Cursor

Reset, Run, Stop, Step and Run-to-Cursor are the basic debug actions. Users can run these actions by clicking the short-cut buttons in the debugger GUI as shown in the below figure.

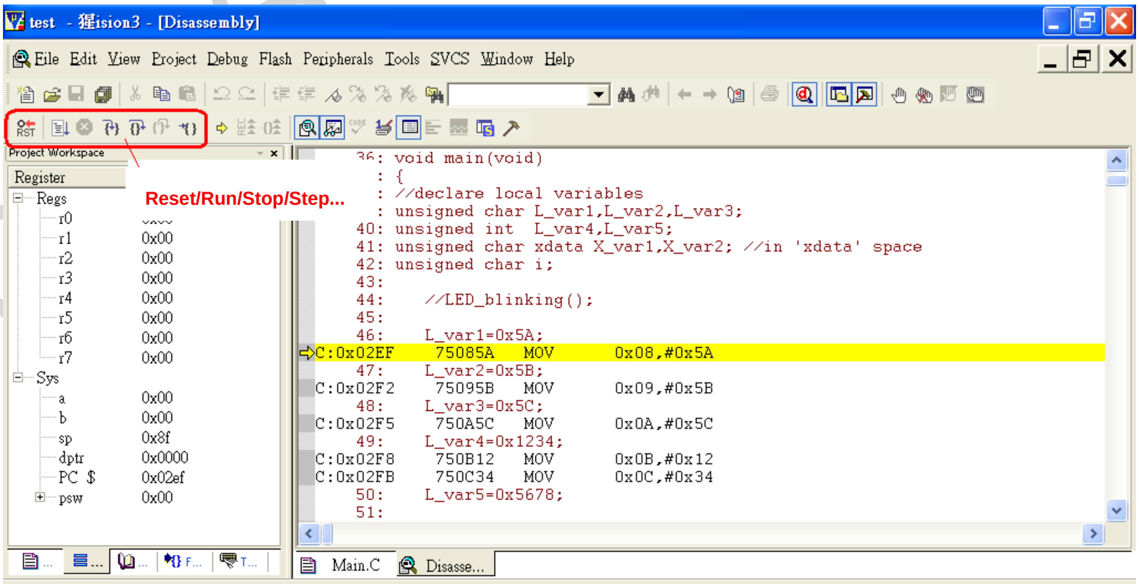

Figure 12. Debug Actions

5.2.2 Source-Level Debug

Open the source file in *Files* tab to perform the source-level debug and return to the register window by clicking *Regs* tab if needed, as shown in Figure 12.

5.2.3 Breakpoint Settings

It supports setting up to 4 breakpoints simultaneously during debugging.

- **Insert/remove breakpoint**

Move the cursor to an instruction line where users need have breakpoint operation, right click then select *Insert/Remove Breakpoint* to insert or remove the breakpoint as shown in the below figure.

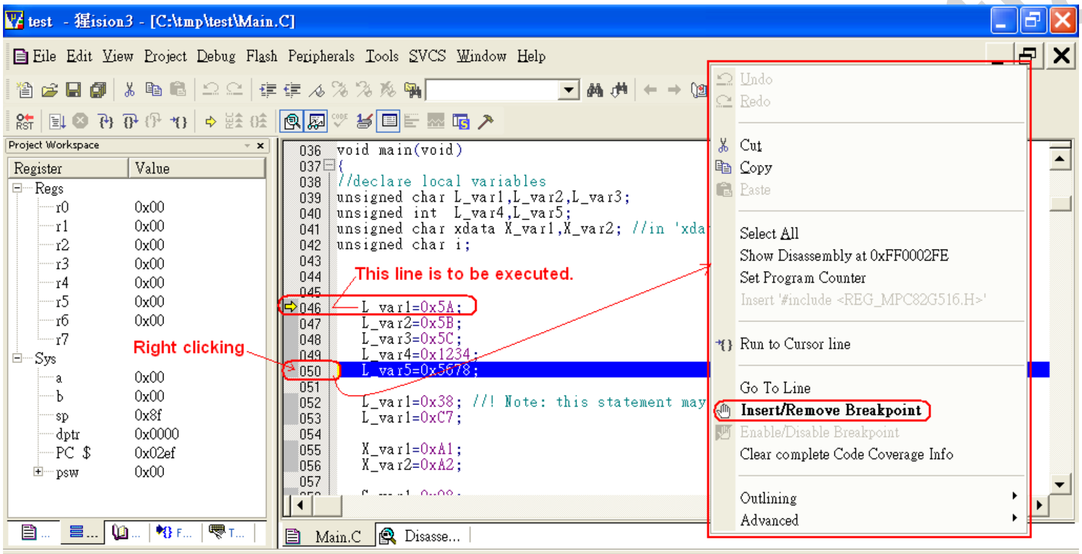

Figure 13. Insert/Remove Breakpoint

- **Enable/disable breakpoint**

Move the cursor to an instruction line, right click and then select *Enable/Disable Breakpoint* to enable or disable the breakpoint if this line already has a breakpoint inserted previously.

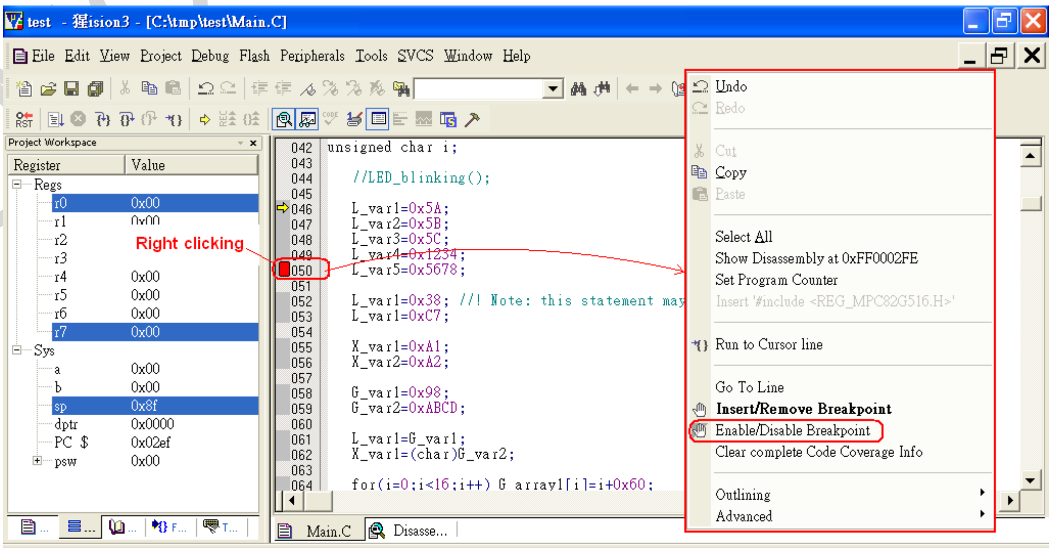

Figure 14. Enable/disable Breakpoint

5.2.4 View/Edit Contents of Peripheral Registers

Peripherals' registers cannot be viewed in the register window. They can be viewed by selecting a specific peripheral item and ticking on the registers for view in the sub-menu as shown in the below window.

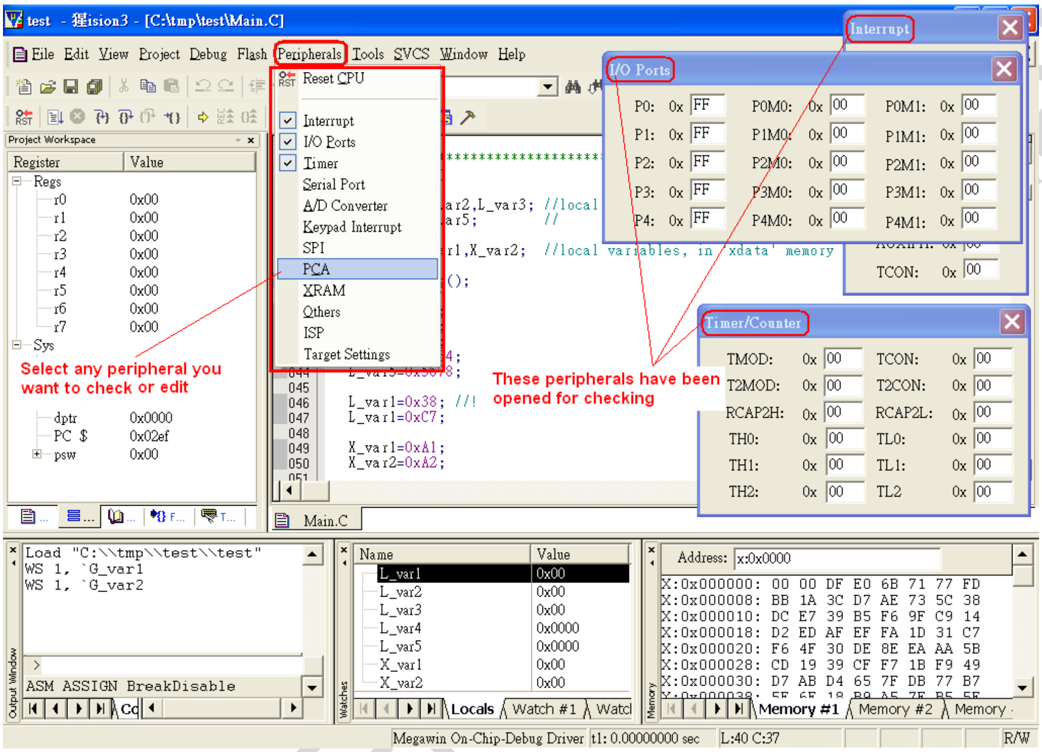

Figure 15. View/Edit the Contents of Peripheral Registers

5.2.5 View Disassembly Window

Disassembly window displays the corresponding assembly code of the source-level code. To open this window, select *View* in the main menu, then select *Disassembly Window* in its sub-menu as shown in the below figure.

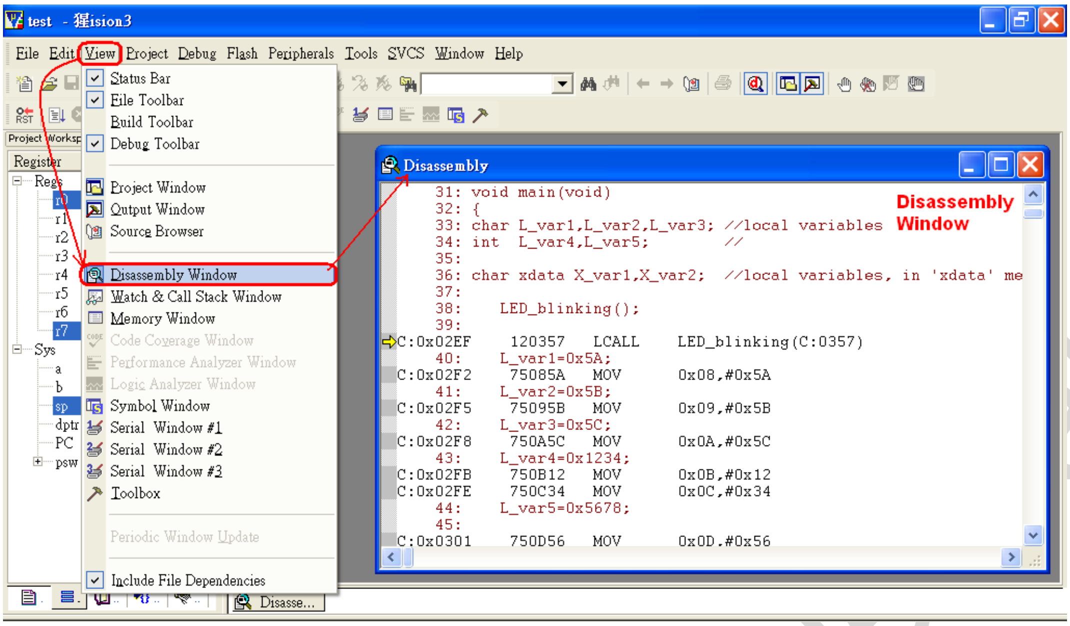

Figure 16. View Disassembly Window

The maximized *Disassembly* window is shown in the below figure.

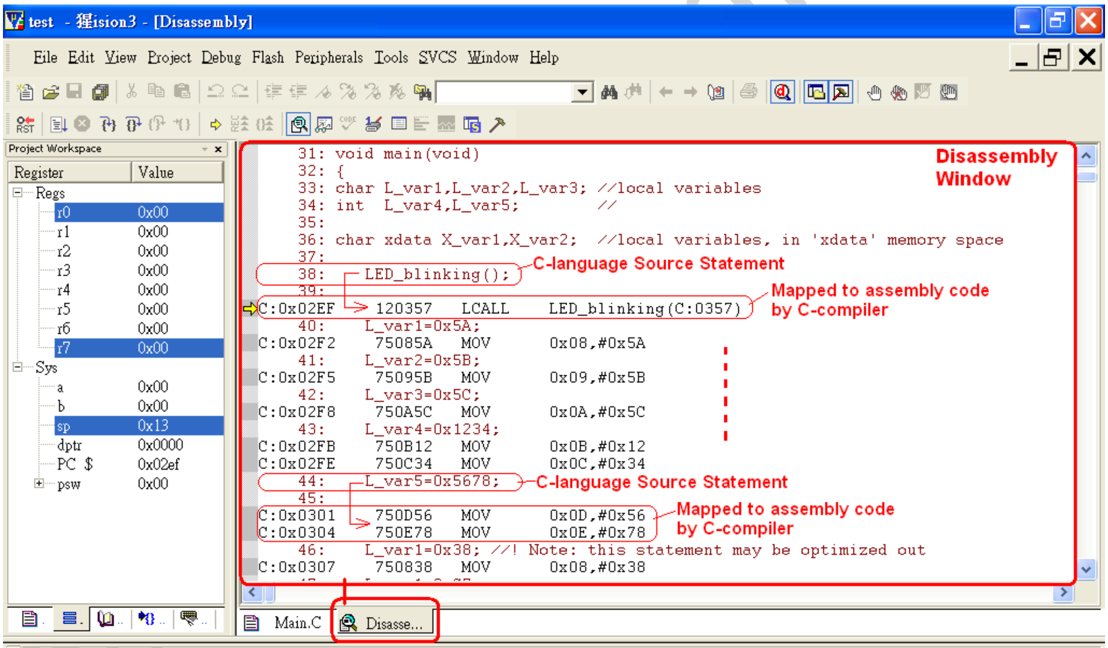

Figure 17. Maximized Disassembly Window

5.2.6 View Watch Window

The watch window helps users to check either local variables or global variables as shown in the below figure.

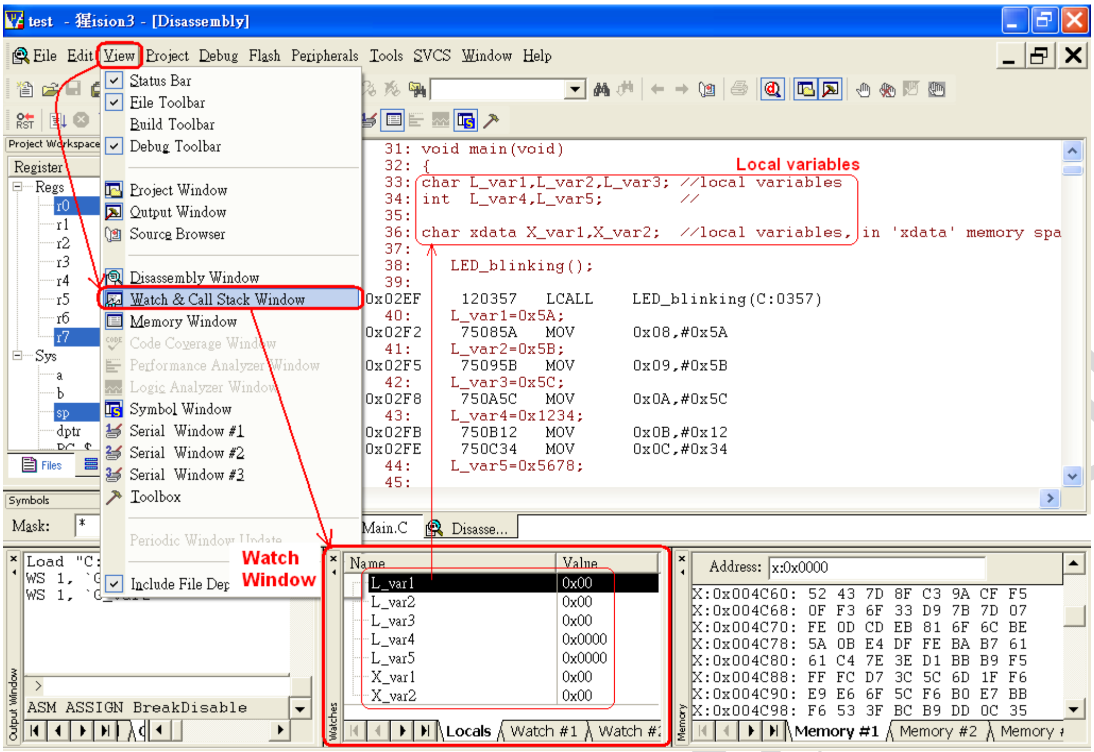

Figure 18. View Watch Window

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

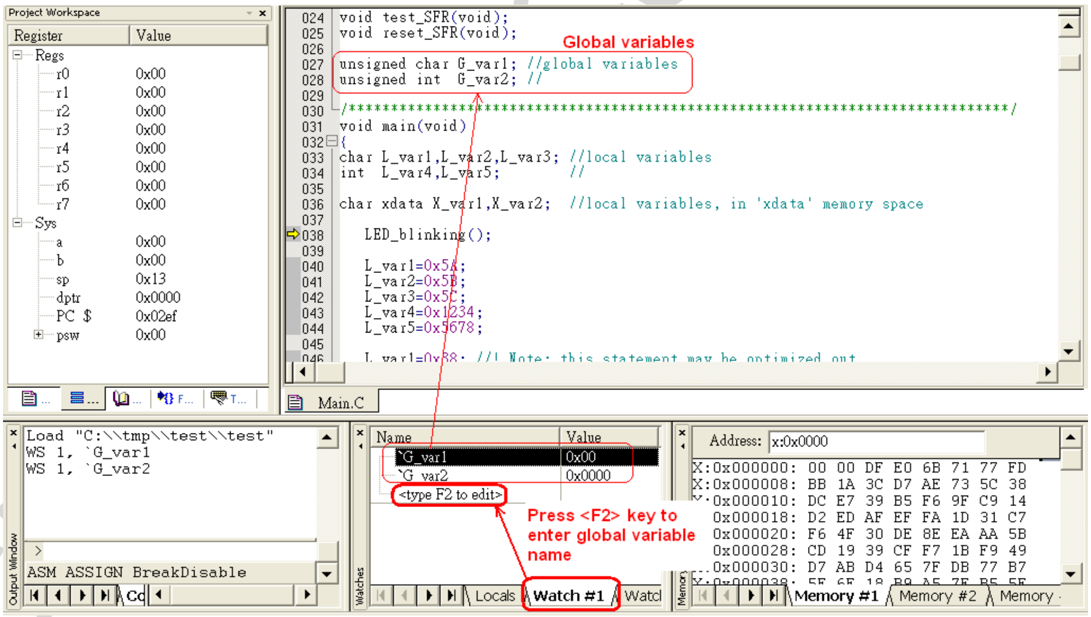

Figure 19. View Global Variables

5.2.7 View Memory Window

To open this window, select *View* item in the main menu then select *Memory Window* in its sub-menu. The available commands are as follows.

d:0x00~d:0xFF is for data type memory view.

i:0x00~i:0xFF is for idata type memory view

x:0x0000~x:0xFFFF is for xdata type memory view

c:0x0000~c:0xFFFF is for code type memory view

Users can view the memory content of any of these 4 types by entering the corresponding command. Refer to Chapter 7.2 for more details of xdata type memory view.

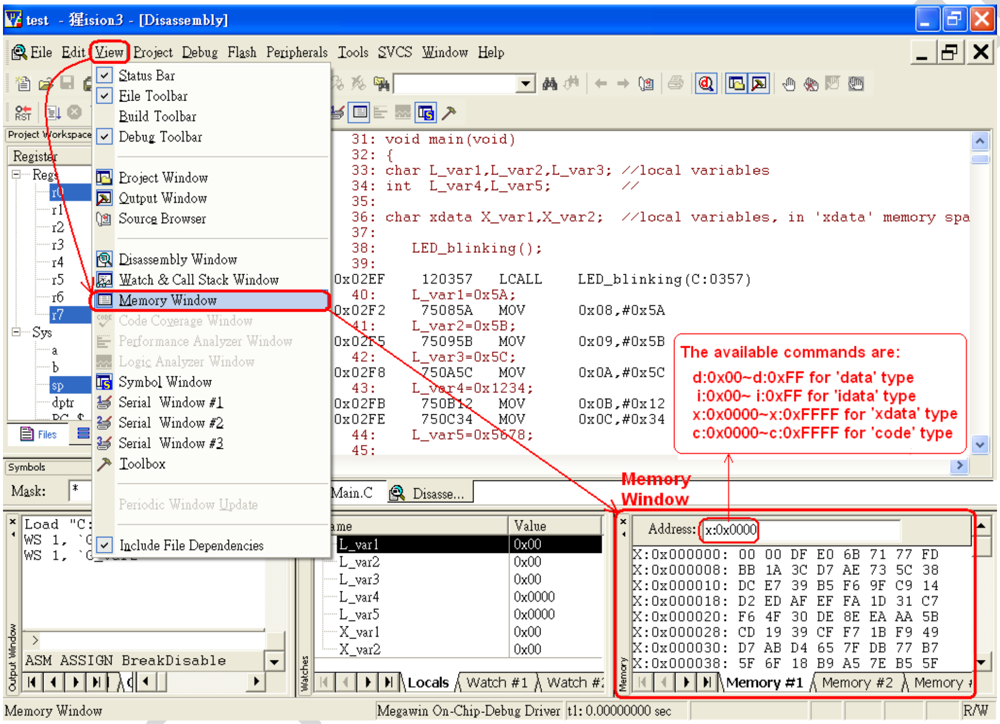

Figure 20. View Memory Window

6 ICP Tools

6.1 Introduction

ICP (In-Circuit Programming) is a tool allowing users to update user programs and modify hardware settings without removing a chip from the product, through using ICP software and ICE adapters. As user programs can be saved in non-volatile memory, it supports offline programming through the ICE adapter, with no need for PC connecting, suitable for cases without a computer.

6.2 ICP Usage

To open the ICP software, users need to go to the Keil installation folder \C51\INC\Cmostek\ and execute ICPProgrammer.exe.

Notes:

1. Please open the project and build it first, then ICP software can run correctly.

6.2.1 Download Program to ICE Adapter

Step 1, select MCU model

This step can be skipped if users open ICP software by clicking on the toolbar. The ICP software will apply the model used in the project automatically in this case.

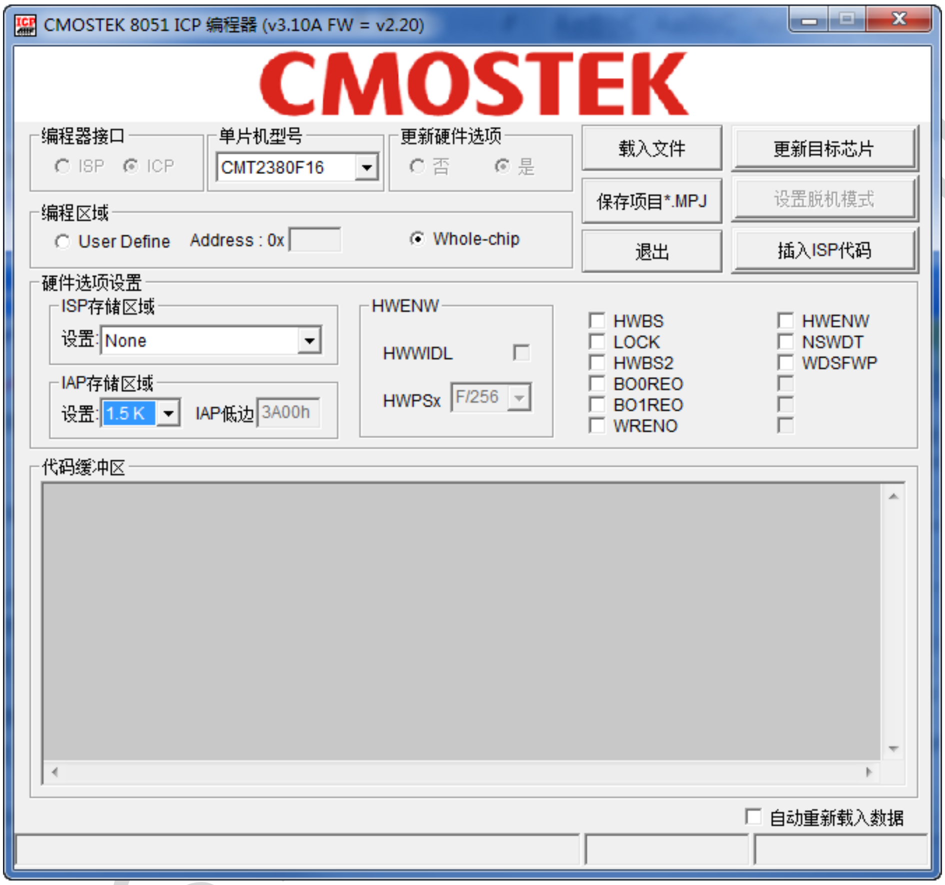

Figure 21. Select MCU Model

Step 2, click *load file* (as shown in the red rectangle in the below figure), select to load *AP file* or *IAP file*. Users can reload a file by repeating load file if needed. Please input file path when loading an IAP file. It supports HEX and BIN file formats.

If users click on the toolbar to open the ICP software, step 1 can be skipped as the ICP software will load the program used in the project automatically.

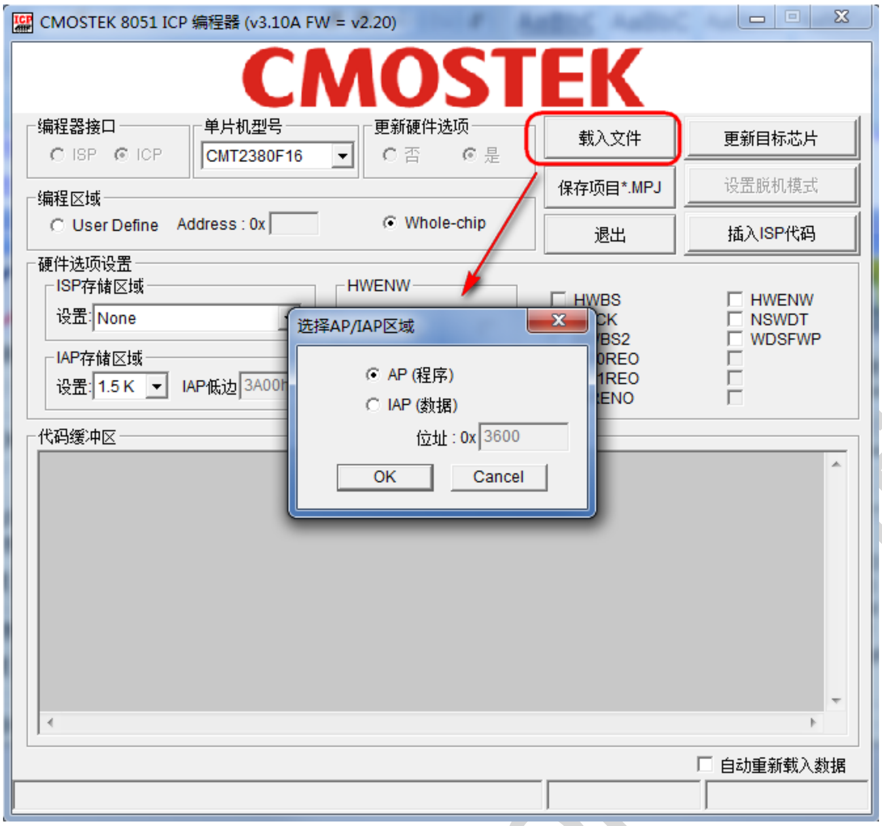

Figure 22. Load File

Step 3, click *Insert ISP Code* (as shown in the red rectangle in the below figure) to insert the ISP code provided by *Shengquan* or user-defined ISP code.

If users do not need to use the ISP function, skip step 3.

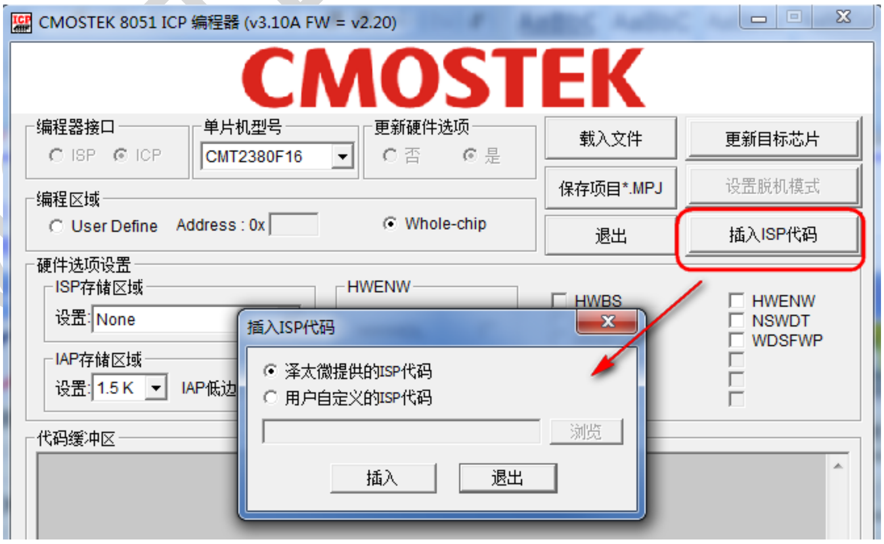

Figure 23. Insert ISP Code

Step 4, hardware settings.

Figure 24. Hardware Settings

Step 5, click *set offline mode* to download the data into the ICE adapter.

Please be noted that *set offline mode* (as shown in the red rectangle in the below figure) function can be used only when the ICE adapter is connected.

Figure 25. Set Offline Mode

6.2.2 Update Target Chip

The methods to update target chips are as follows.

Method 1, refer to step 1 to step 4 of *Download Program to ICE Adapter* in chapter 6.2.1, and click *update target chip* for online update.

Method 2, refer to *Download Program to ICE Adapter* in chapter 6.2.1, press *download* key on the ICE adapter for offline update.

7 Special Considerations

7.1 Register Definition Files

Register definition files *REG_CMT2380F16.INC* and *REG_CMT2380F16.H* define all Special Function Registers (SFRs) and bit-addressable control/status bits. They are installed in the default path of the Keil 8051 IDE software during the OCD ICE installation (see Chapter 2 for more details). Therefore, when using Keil for programming, users can include the register definition files by $INCLUDE (REG_ CMT2380F16.INC) or #include <REG_ CMT2380F16.H> with no need for copying the register definition files into users' project folders.

7.2 On-chip XRAM and External Data Memory

The CMT2380F16 provides on-chip XRAM (eXpanded RAM), which is accessed in the same way as the traditional external data memory. The size of on-chip XRAM in CMT2380F16 is 1024 bytes with an address range of 0x0000 to 0x03FF, which overlaps that of the external data memory. So, there must be a control bit applied to distinguish these two physical memories during access. The ERAM bit (bit-1 in register AUXR) plays this role. As the C51 compiler will not take care which physical memory a 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 power on or chip reset for on-chip XRAM accessing.

The C51 compiler offers two different memory types for accessing external data: xdata and pdata (the xdata memory can locate the 64 kbyte external data memory while the pdata can locate the 256 bits data only). To view xdata or pdata directly in the Memory Window rather than in the Watch Window, users need to click XRAM in *Peripherals* menu then tick on *Display xdata from on-chip XRAM* or *Display xdata from external RAM* in the sub-window popped up, as shown in the following figure.

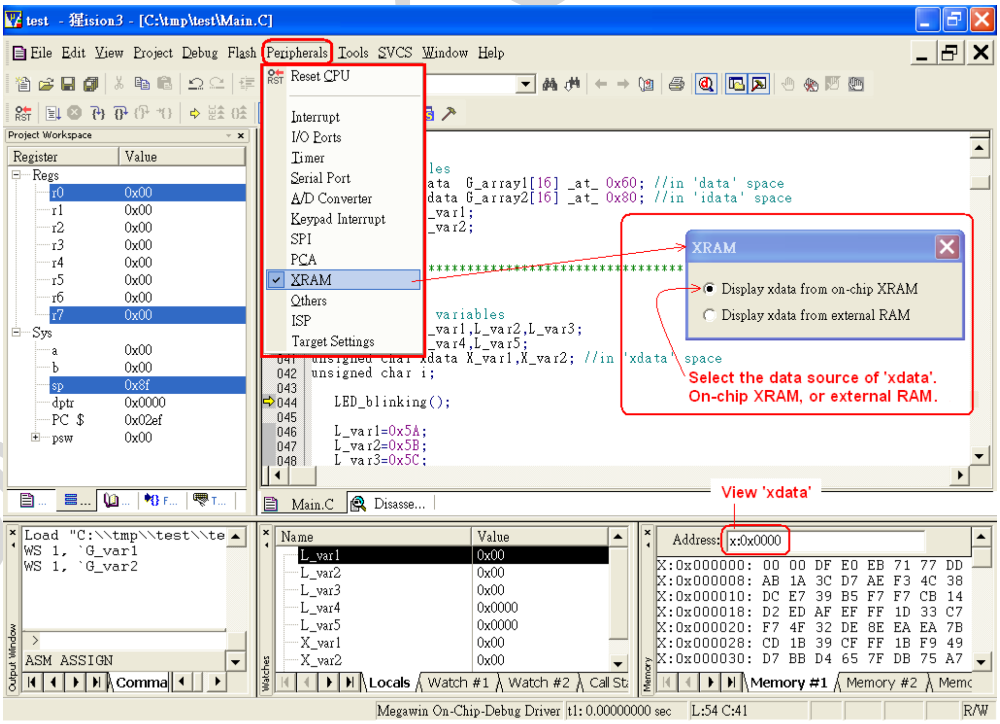

Figure 26. View Xdata or Pdata in Memory Window

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

An example of using both on-chip XRAM and external RAM are as follows.

```
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
```

Please be noted that the linking warning listed below can be ignored. Although we intentionally define G_array1 and G_array2 in the same address space, the ERAM bit is applied to control switching between the different physical memory.

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

Figure 27. Linking Warning

7.3 Code Optimization and Source-Level Debug

As shown in the following source code, the C51 compiler will not generate any machine code for L_var1=0x38 as it is followed by L_var1=0xC7, which makes it a meaningless instruction. Due to code optimization, L_var1=0x38 will be optimized out (ignored) unless the code optimization is disabled as described in Chapter 4.4.

```
unsigned char L_var1;

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

It should be noticed that, during source-level debug, when executing this instruction, L_var1 will not show 0x38 but a random number since there is no machine code for this instruction actually.

As users may disable the compiler’s code optimization for debug purpose sometimes, it should be noted 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, the below linking error message indicates the variables exceeding the MCU memory range. To make this error disappear, users need to enable the compiler’s code optimization to let the compiler make more efficient use of the memory.

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

Figure 28. Linking Error

7.4 Source-Level Debug per For-loop

The following two instruction sets are exactly the same for the 8051 CPU to execute them. When performing step run in source-level debug, the first instruction set will not encounter problem, however running the second instruction set will take much long time, which may be caused by uncertain processing in the Keil debugger on such instructions. So, for step run, it is recommended to use the first instruction set instead of the second one before we get the clarification from Keil. Another way for debugging the second instruction set is, moving the cursor to line 2 and clicking *Run-to-Cursor* button to skip line 1.

Instruction 1:

Line1: for (i=0; i<16; i++)

{

Line2:   G_array1[i]=i+0x60;

Line3:   }

Instruction 2:

Line1: for (i=0; i<16; i++)

G_array1[i]=i+0x60;

Line2: …

Line3: …

7.5 Hardware Requirements for Debug

There are two hardware requirements regarding to the dScope-Debugger mode.

- Requirement 1, the debugged chip must be in un-locked state

If a debugged chip is locked, the downloading of the user's application program in the dScope- Debugger mode will cause the chip to be erased, thus all the chip's hardware settings will be disabled, resulting in abnormal behaviors of the chip due to loss of original hardware settings. For example, for a locked chip with IAP configured, after accessing the dScope-Debugger and downloading the user's application program, its IAP setting will disappear, which may cause abnormal behaviors of the chip.

- Requirement 2, the ISP function of the debugged chip must be disabled

If the ISP function is enabled, the debugged chip will always boot from the ISP-memory and run the ISP program (e.g. ISP-code) instead of user program when receiving a reset command in the dScope-Debugger mode. Thus during debugging, the HWBS must be disabled to prevent ISP function execution.

Notes:

1. When application code debug completes, users can restore the original hardware settings using ICP Programmer.

7.6 Error Message

The error message, *Error - Target DLL has been cancelled. Debugger aborted!* ,as shown in the below figure will display under the following conditions.

1. ICE adapter hardware fails.
2. Target MCU doesn't work, e.g. chip is not powered on or damaged.
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.

Figure 29. Error Message

7.7 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 USB HUB. When debugging using dScope, users should plug the ICE adapter into the host's USB port directly to speed up the downloading, as shown in Figure 30. Please don't plug into a hub then connect to the host, as shown in Figure 31.

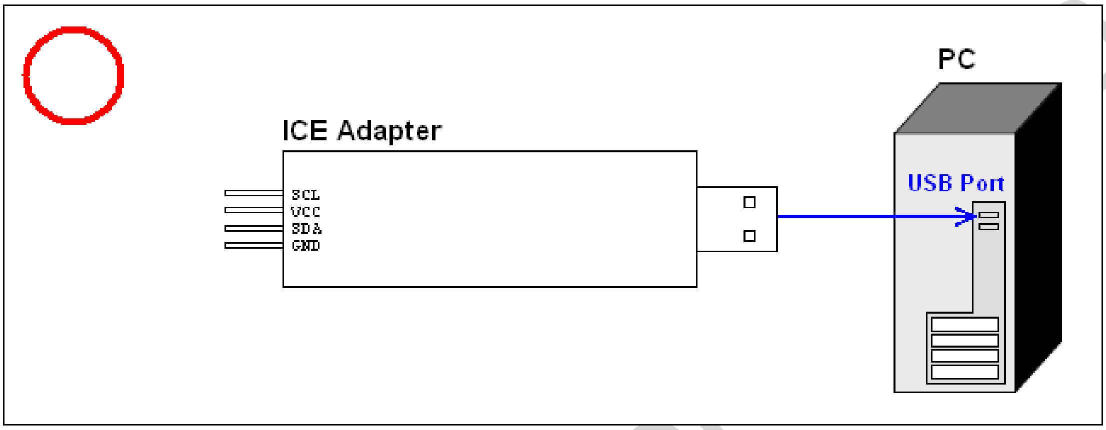

Figure 30. Plug into Host's USB Port Directly

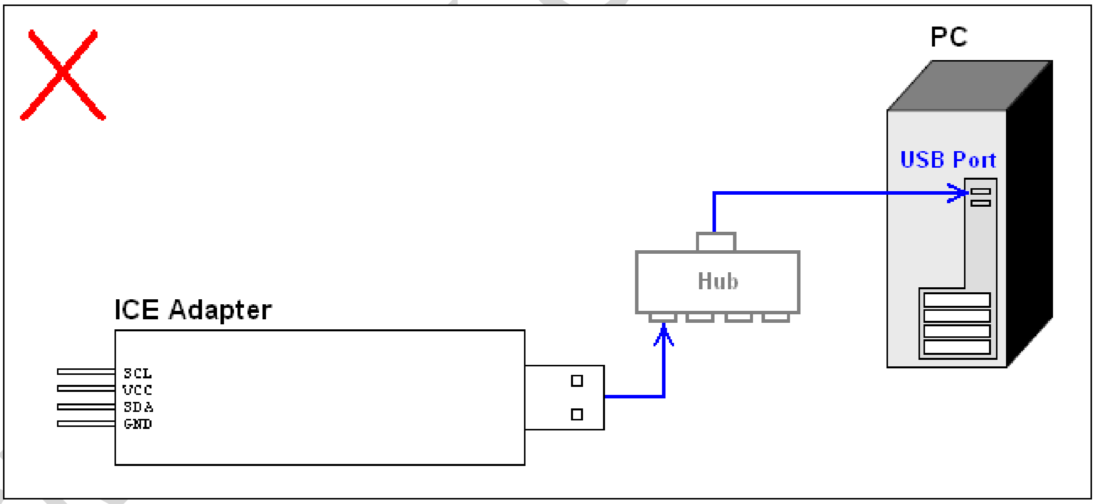

Figure 31. Don't Plug into a USC Hub

8 Revise History

Table 3. Revise History Records

Version No.	Chapter	Description	Date
0.8	All	Initial version	2018-5-6

9 Contacts

CMOSTEK Microelectronics Co., Ltd. Shenzhen Branch
Address: 2/F Building 3, Pingshan Private Enterprise S.T. Park, Xili, Nanshan District, Shenzhen, Guangdong, China

Tel:	+86-755-83231427
Post Code:	518057
Sales:	sales@cmostek.com
Supports:	support@cmostek.com
Website:	www.cmostek.com