SONiX 8-Bit MCU

M2IDE User Manual

Version 1.0

Modification Records

Version	Date	Description
V1.0	2013/7/10	Initial draft

Content

MODIFICATION RECORDS **2**

CONTENT **3**

PREFACE **4**

CHAPTER 1 SYSTEM OVERVIEW AND INSTALLATION **5**

1.1 M2IDE INTRODUCTION 5
1.2 INSTALLATION 5
1.2.1 System Configuration Requirements *5*
1.2.2 Hardware Installation *5*
1.2.3 Software Installation *7*

CHAPTER 2 WINDOWS INTERFACE **13**

2.1 QUICK START 13
2.2 MENU - FILE / EDIT / VIEW / DEBUG / AUXILIARY / WINDOWS / HELP OPTIONS 14
2.2.1 Start M2IDE System *14*
2.2.2 M2IDE Interface *16*
2.2.3 File menu（File） 19
2.2.4 Edit Menu（Edit） 22
2.2.5 View Menu（View） 26
2.2.6 Debug Menu（Debug） 27
2.2.7 Utility Menu（Utility） 31
2.2.8 Window Menu（Window） 33
2.2.9 Help Menu（Help） 34
2.2.10 Windows Management *35*
2.3 CREATE AND DEBUG APPLICATION PROGRAMS 38
2.3.1 Create a Project /New File *38*
2.3.2 Compiling and Linking of Program *44*
2.3.3 Running and Debugging of Program *46*
2.3.4 Compiling Option (Code Option) *51*
2.3.5 Types of Project File *52*
2.4 HOW TO EMULATE LCD 53

CHAPTER 3 DEVELOPMENT LANGUAGE **54**

3.1 INSTRUCTION SET 54
3.2 PSEUDO-INSTRUCTION 56
3.3 INCLUDE FILE 57
3.4 MACRO 58
3.5 CONDITIONAL COMPILATION CONTROL 62

APPENDIX **65**

APPENDIX I COMPILER ERROR INFORAMTION DESCRIPTION 65
APPENDIX II LIST OF MENU COMMANDS, TOOLS AND SHORTCUTS 69
APPENDIX III PSEUDO-INSTRUCTION LIST 72
APPENDIX IV FIGURE LIST 73
APPENDIX V. RELEVANT FAQ 75

Preface

For Sonix 8-bit micro-controller, the development system of SN8P2XXX series is adopted the on-line emulator (ICE), while the software operating in PC is M2IDE or SN8 C Studio.

This manual is to describe the M2IDE.

Chapter 1 System Overview and Installation

1.1 Introduction

M2IDE is composed by the editor, the assembler, the debugger, the programming AP of chips and other components, and can complete the establishment and management of engineering, the editing code, the compiling, the linking, and the generation of target code and the hardware emulation.

The user can login www.sonix.com.tw to download the latest installation file of M2IDE.

M2IDE Features:

- Real-time emulation for program and command
- Convenient application and installation
- Universal windows interface
- Support the operating platform of multi-source program file (a application item can contain more than one source program file)
- Support the software-simulating LCD emulation

1.2 Installation

1.2.1 System Configuration Requirements

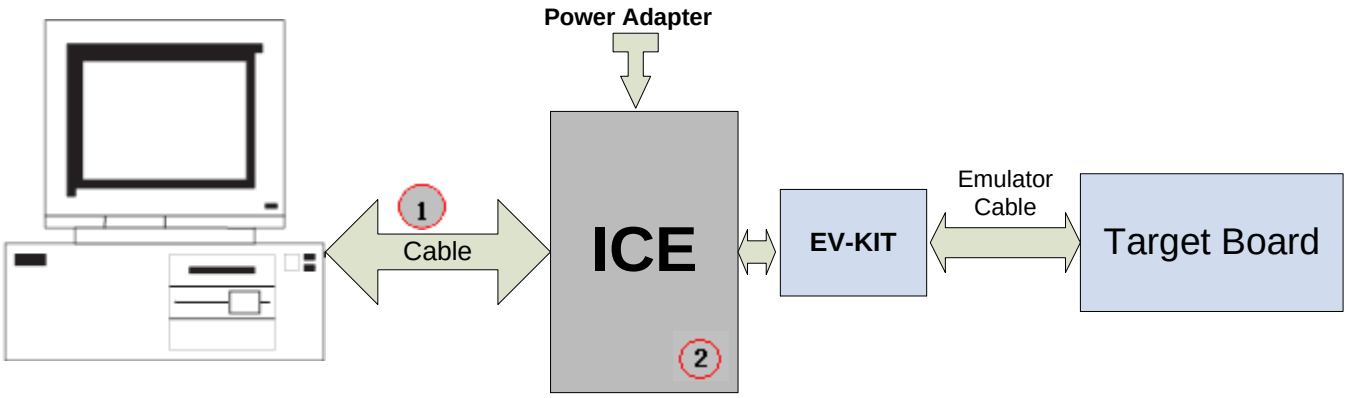

Figure 1-1 Emulation Connection Diagram

The descriptions of ① and ② in Figure 1-1 are as follows:

1. Cable of ① can be Printer Cable or USB Cable;
   SN8ICE 2K provides Printer port to connect with PC;
   SN8ICE 2K Plus II provides USB port to connect with PC;

   SONiX also provides UTP device, to convert Printer Port to USB Port;

In order to ensure the normal operation of M2IDE, the installation environment must meet the minimum configuration requirements, and the conditions are as follows:

- Windows 98/2000/ XP/ Vista operation system
- 32 MB available space of hard disk at least
- 32 MB space of memory at least

1.2.2 Hardware Installation

Before the hardware connection, please confirm whether the connection of crystal oscillator, capacitance of oscillating circuit, and the short-circuit cap is right. Please refer to the Installation of crystal oscillator in Notes for details.

Before the hardware connection, please confirm the power supply of ICE is Off!

SN8ICE 2K hardware is shown in Figure 1-2:

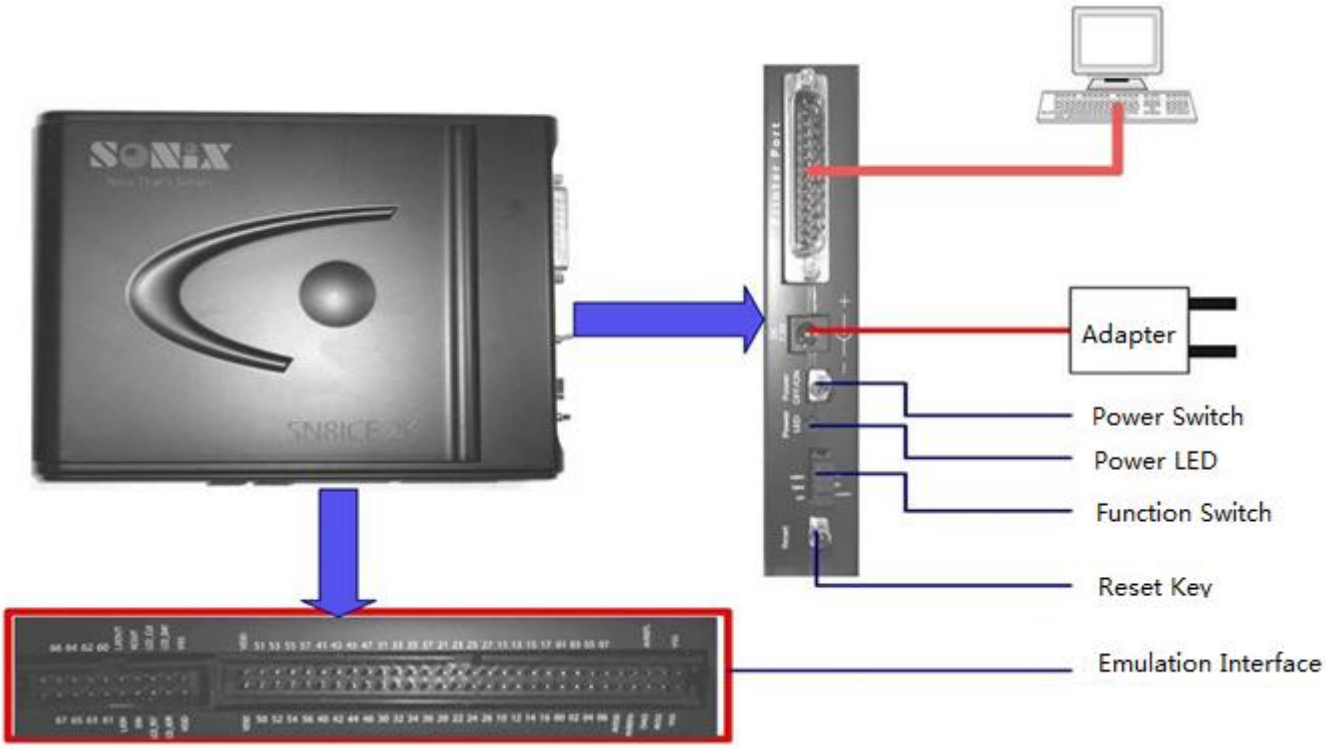

Figure 1-2 SN8ICE 2K Hardware Description

The hardware of emulator is installed in accordance with the following steps:

1. Connect SN8ICE 2K / SN8ICE_USB / SN8ICE 2K Plus on-line emulator with LPT port of PC via parallel port cable;
   The laptop user can use the tool of UTP provided by SONiX for connection;
   SN8ICE 2K Plus II on-line emulator is connected with USB interface of PC via USB cable.
2. Select suitable LPT port;
3. Emulator together with DC power supply（it must use the specialized DC power supply）;
4. After the development software M2IDE_Vxxx is installed, it can edit, compile or emulate program.

Post-connection diagram is shown in Firgure 1-3.

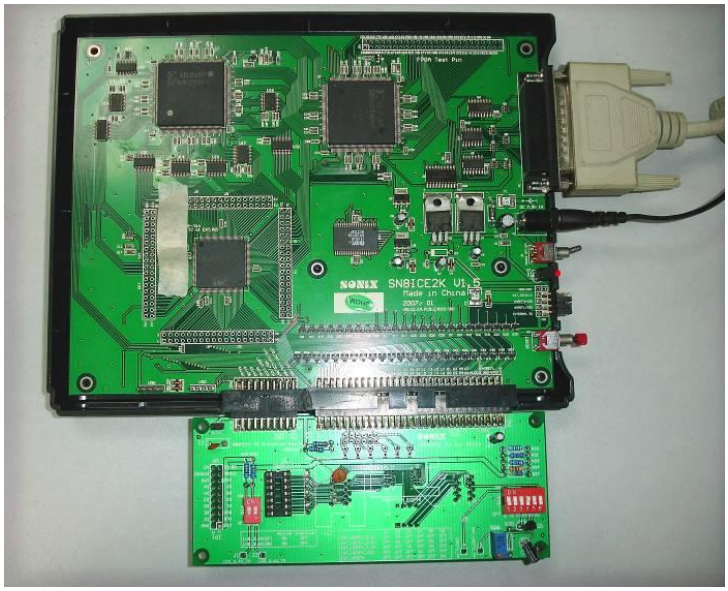

Figure 1-3. Post-connection Diagram

Please refer to SONiX's SN8ICE2K Easy GuideV1.0 for more information about emulator, the user can downloard from internet.

1.2.3 Software Installation

The format of installation file name is as follows: M2IDE_Vxxx.exe, of which M2IDE is the name of software package, Vxxx is the version of software, e.g. M2IDE_V119. The user can login www.sonix.com.tw to download the latest installation file of M2IDE.

M2IDE_V119 is taken for example to explain the installation steps of M2IDE compiler in the following.

Double click the installation file "M2IDE_V119.exe" to start the installation, at that time a guide dialog box will pop up (as shown in Figure 1-4), in this dialog box it will prompt the suitable emulator and chip type for such development environment to the user, and suggest the user to exit other running programs during the installation to ensure the successful installation.

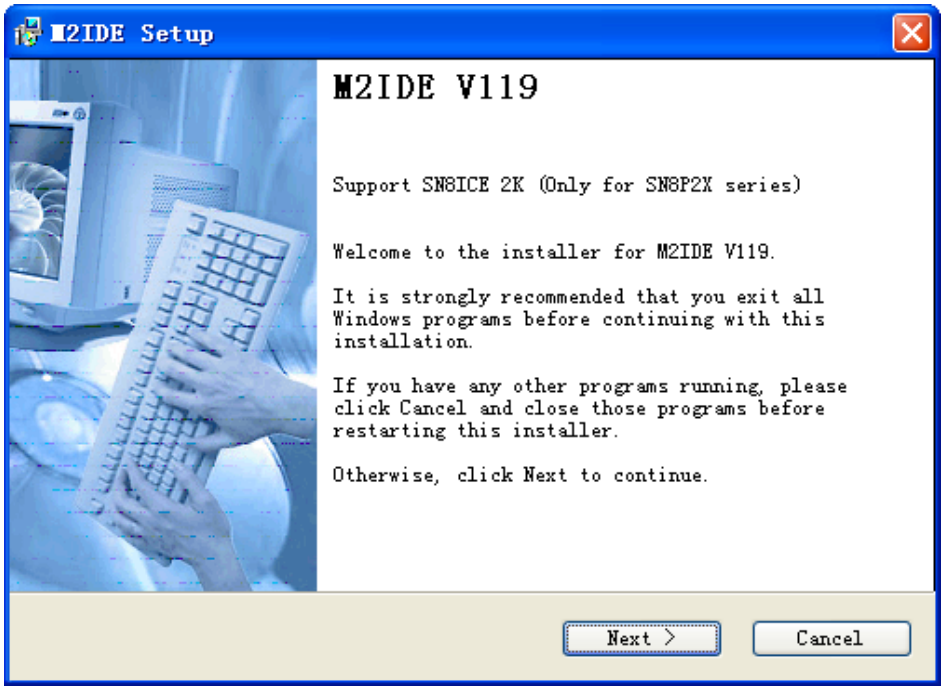

Figure 1-4. M2IDE Installation Guide Dialog Box

Click "Next", at that time an agreement dialog box will pop up (as shown in Figure 1-5), it asks the user to carefully read the License agreement on software application, if you want to continue, the item "I agree to the terms of this license agreement" must be selected.

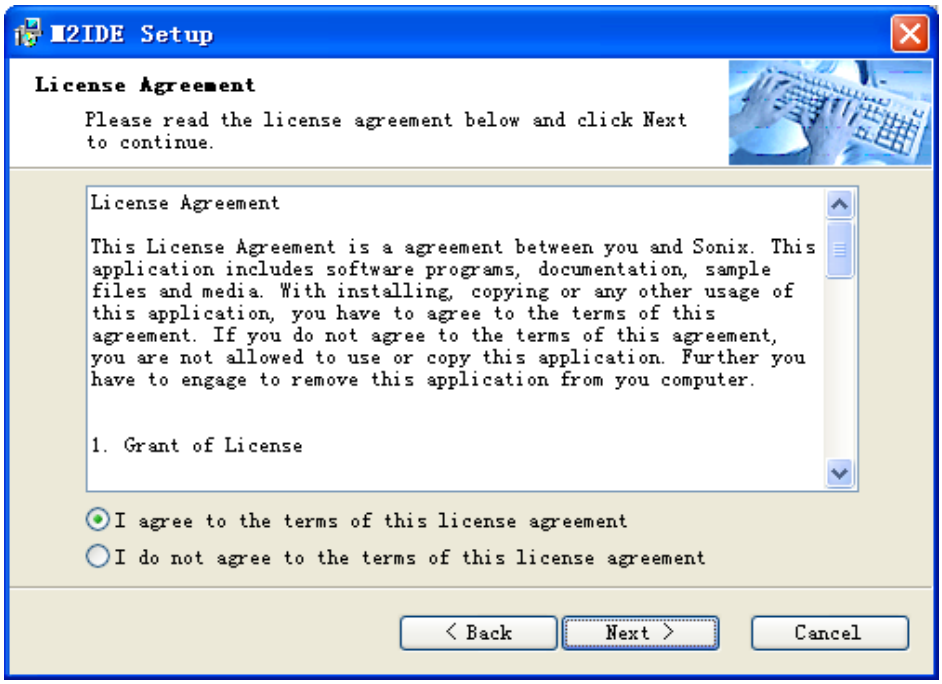

Figure 1-5. M2IDE Installation Agreement Dialog Box

Click “Next” again, the dialog box of installation path selection will pop up (as shown in Figure 1-6), the default installation path is C: \ Sonix \ M2IDE_V119; the user can also click “Change ...” to change the installation path, the window to view path after clicking “Change ...” is shown in Figure 1-7, when the appropriate path is specified, click “OK” to complete the selection of path.

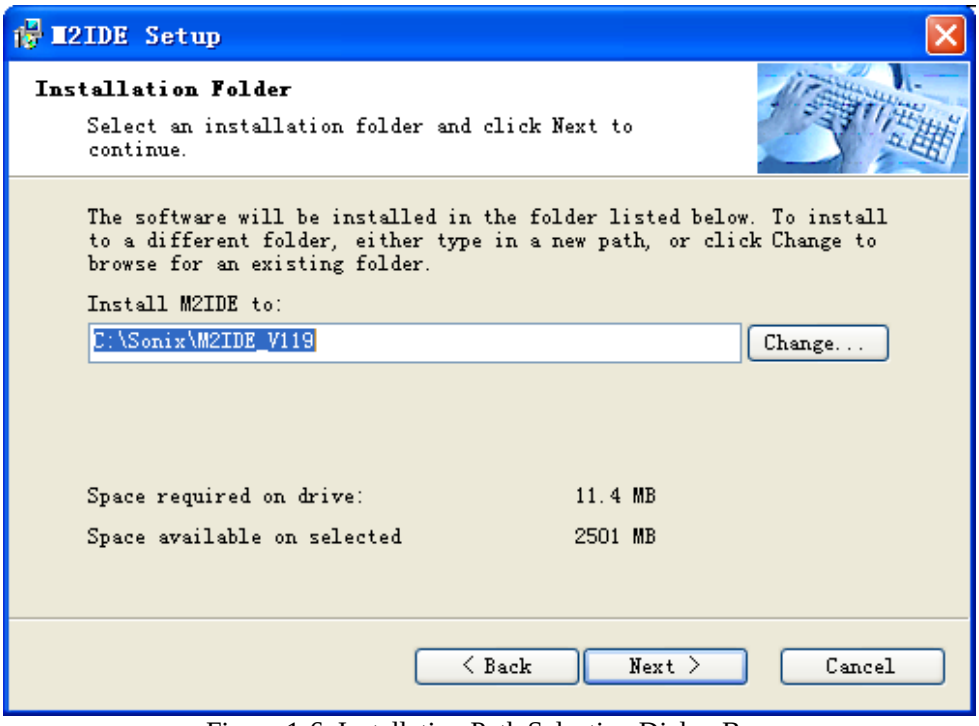

Figure 1-6. Installation Path Selection Dialog Box

Figure 1-7. Window of View Path

Continue to click “Next”, the folder setting window directed by shortcut will pop up, as shown in Figure 1-8, the installation file will create a shortcut, which can be used by the user to point to the default folder or a new folder, or to directly select in the list.

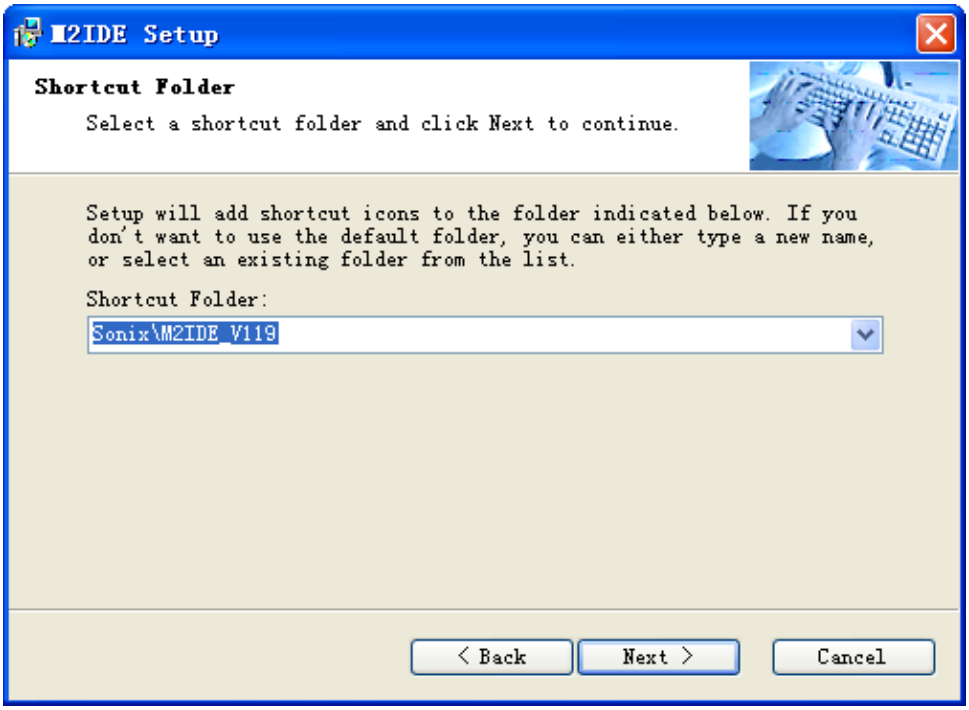

Figure 1-8 Pop-up Shortcut Setting Window

Click “Next” again, the dialog box of installation configuration information will pop up, as shown in Figure 1-9, which contains the relevant information about installation path and shortcut direction, if confirmed, you can click “Next”, at this time the installation progress window will pop up, as shown in Figure 1-10.

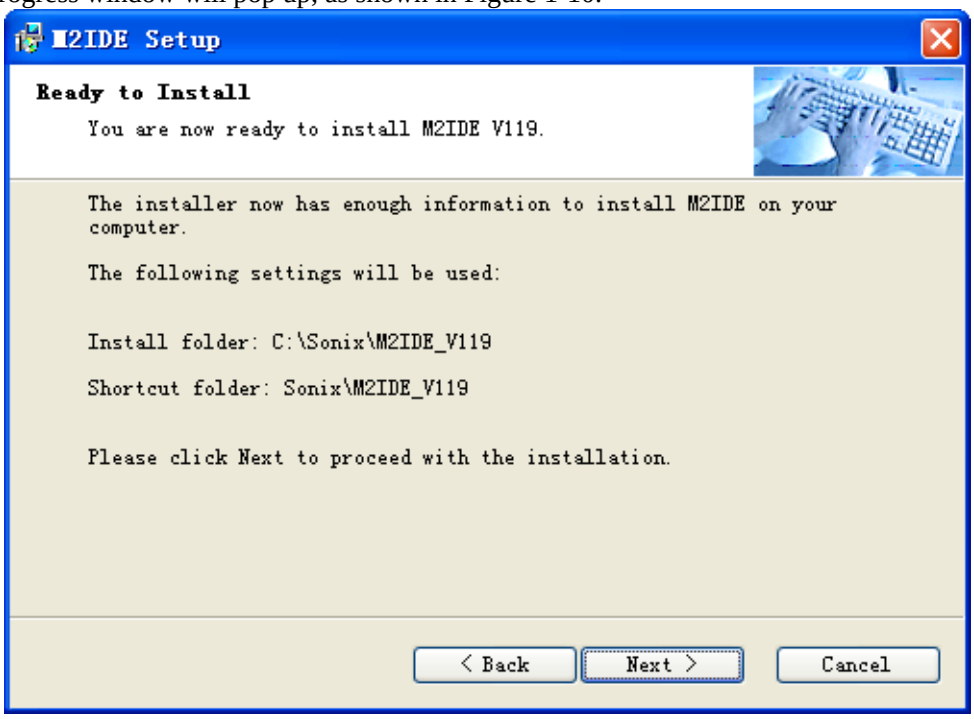

Figure 1-9. Installation Information

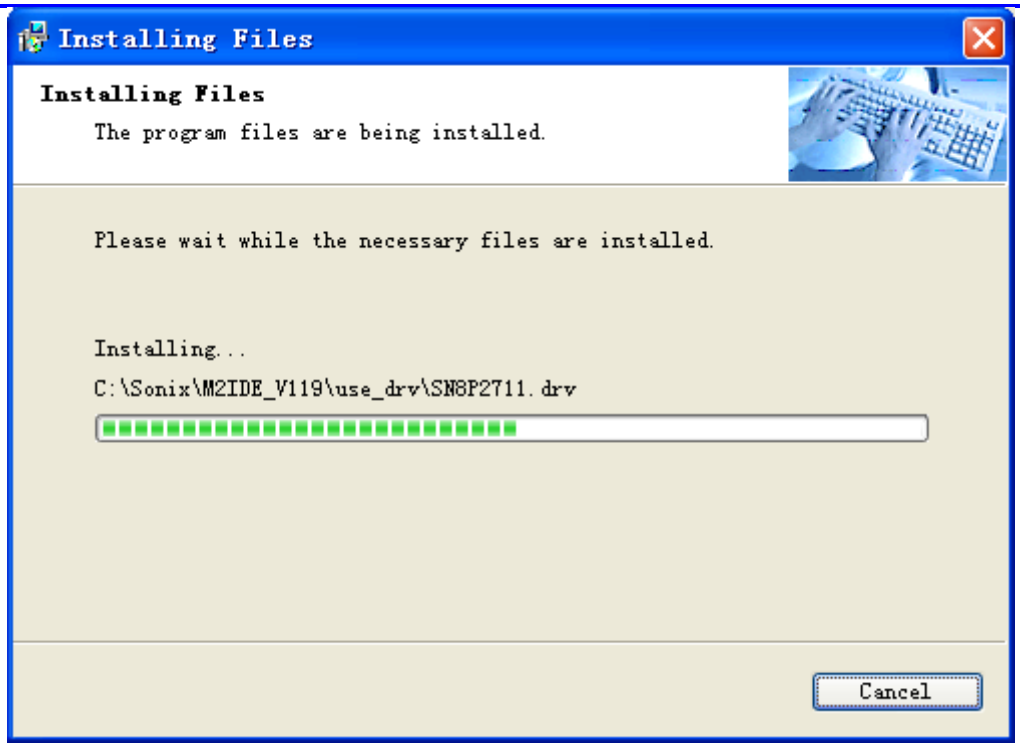

Figure 1-10. Installation Progress Window

Finally the window of installation completion will pop up, as shown in Figure 1-11, indicating that the program has been successfully installed, and then click "Finish" to end the installation. At this time, you can see the shortcut icon of M2IDE_V119 on the desktop, as shown in Figure 1-12, you can double click it to enter the integrated development environment.

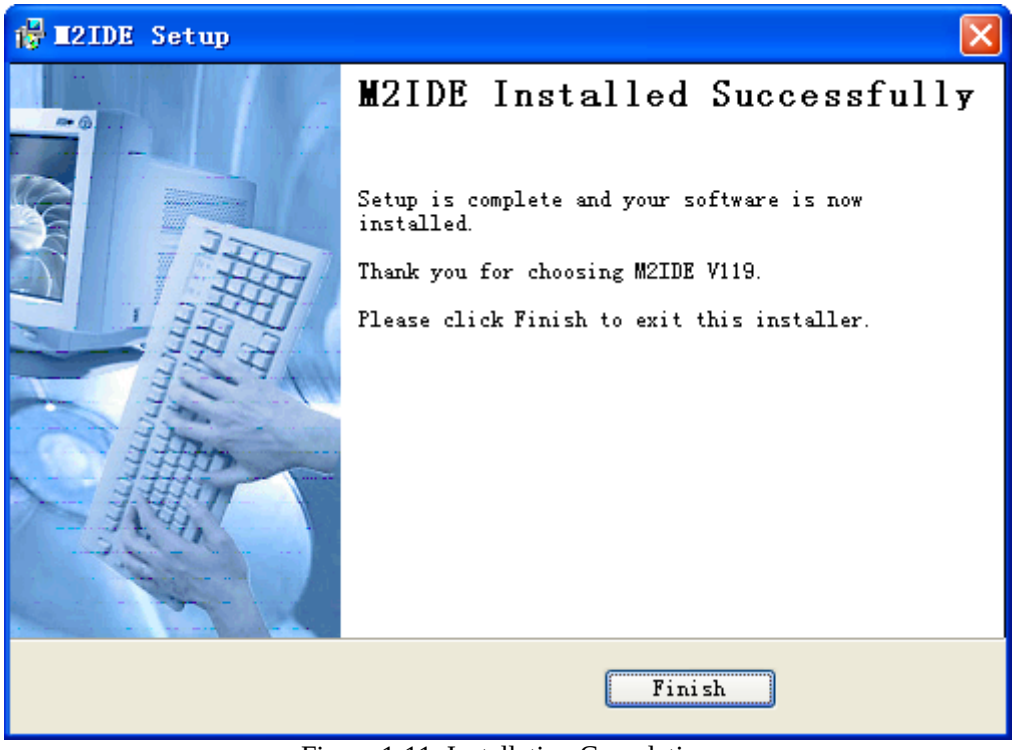

Figure 1-11. Installation Completion

Figure 1-12. Shortcut Icon

After the installation of M2IDE software is completed, double-click the shortcut icon or click "Start/Programs/Sonix/M2IDE V119/M2Asm119.exe" to enter the integrated development environment, as shown in Figure 1-13.

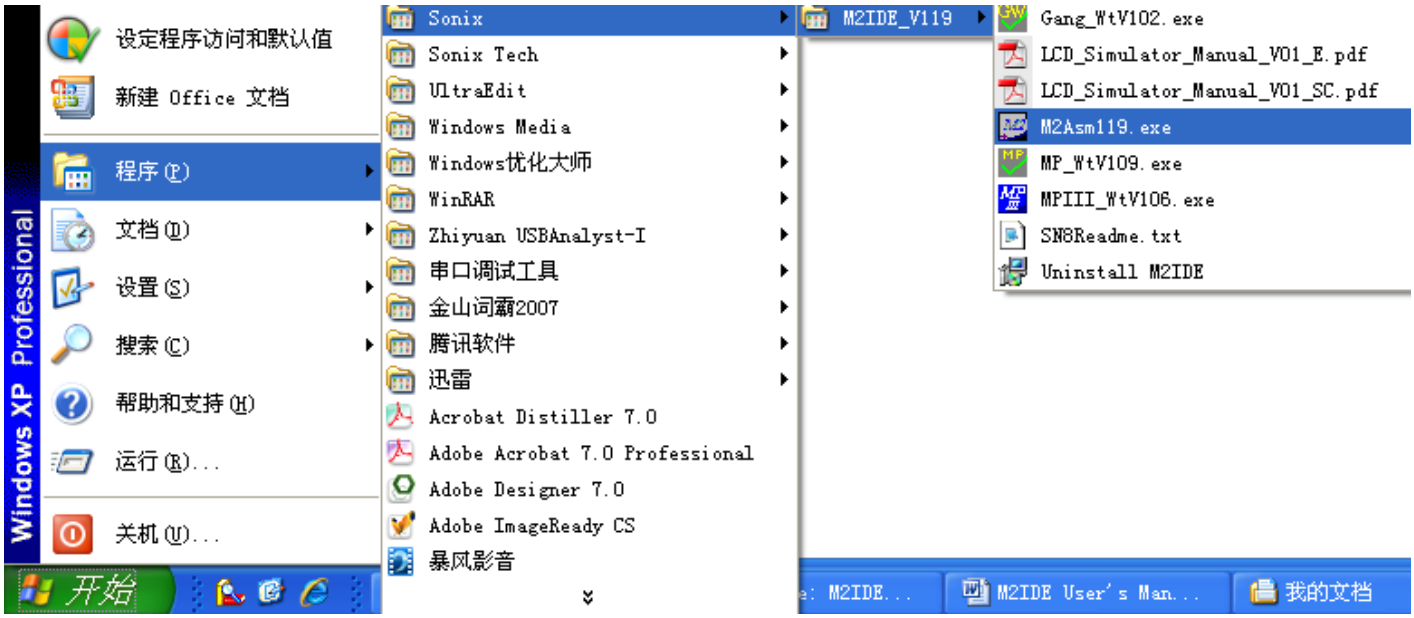

Figure 1-13. Start M2IDE System

The uninstalling method of M2IDE compiler is as follows:

To execute: Start /Programs/Sonix/M2IDE V119/Uninstall M2IDE.

Figure 1-14. Uninstall M2IDE

After it is uninstalled, the following dialog box will pop up.

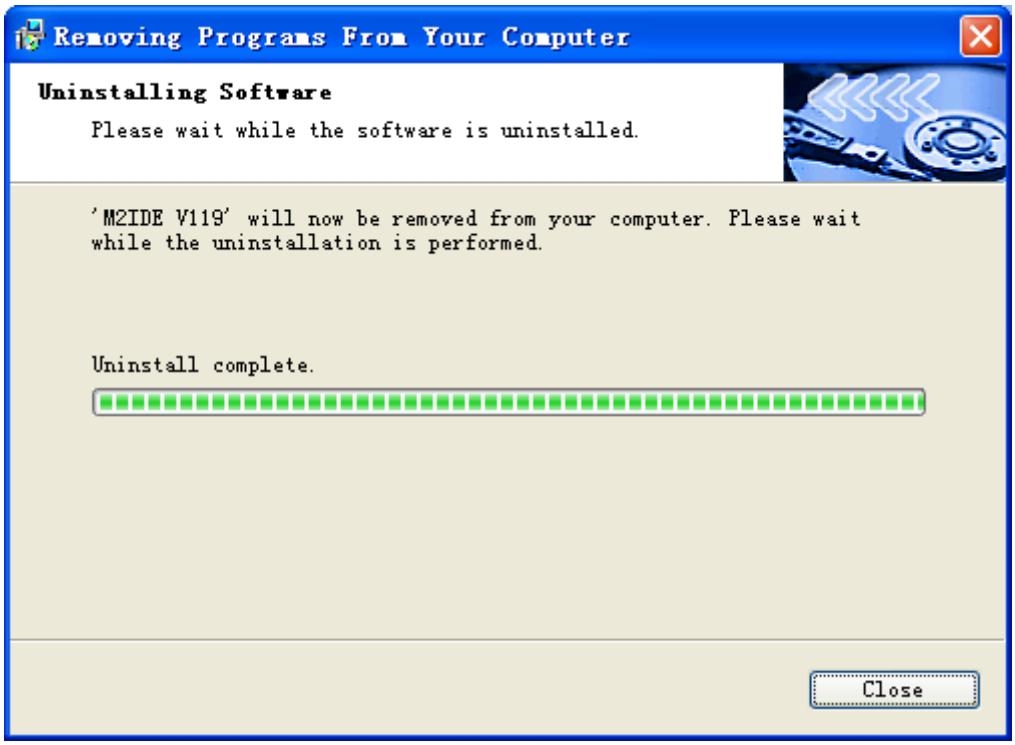

Figure 1-15. Message Box of Successful Uninstalling

Chapter 2 Windows Interface

2.1 Quickly Start

This section is to introduce how to quickly develop an application program.
The emulator is connected with PC via parallel port cable or USB cable, and the power source for on-line emulator shall be connected.

Step 1: Create a new project

- [Start] → [Programs] → [Sonix] → [M2IDE_V119] → M2Asm119.exe to open M2IDE;
- [File] menu → [New] command;
- [File] menu → [Save] command or press the shortcut key "Ctrl + S";
- Select the path to be saved, input the project name, e.g. Main.ASM;
- [File] menu → [New Project] command, open the saved source file;

Step 2: Create the source program file

- [File] menu → [New] command to create a source program file;
- Writing program, after finished, press [File] menu → [Save] command or press the shortcut key "Ctrl + S" to save, e.g. TEST.ASM file name;
- If multiple source program files are required, please repeat the above two steps;
- Open the project source file created in Step 1, use the pseudo-instruction of "Include" to include these files;

Step 3: Compile and debug the project

- [Debug] menu → [Build] command or "F7" to compile;
- Setting the content of each item in the compiling item "Compiler Option", if finished, click "OK" to confirm;
- System will compile all source program files in the project - if any errors in the program, the compiler will prompt the error in output message bar, the user can double click the error message line, and then the system will prompt the position of error and open the source file of such error, you can directly modify and save the file.
- If no errors in all the program files, the system will generate an executable file and load it into ICE for simulation and debugging;
- [Debug] menu → [Go] command or "F5" to run the program;

Repeat the above steps until there are no errors.

Step 4: Program chips

- [Start] → [Programs] → [Sonix] → [M2IDE_V119] → MPIII_WtV106.exe to open programming interface;
- Select the MCU type and programming file *. SN8, the file of .SN8 is generated automatically during the compiling of program, and saved in the same directory of source program;
- After .SN8 is successfully downloaded, the right message box of MPIII_WtV106.exe will give a prompt of successful download;
- Click "Auto-programming" to program MCU;

2.2 Menu - File / Edit / View / Debug / utility / Window / Help Options

2.2.1 Start M2IDE System

After the installation of M2IDE software is completed, double-click the shortcut icon on desktop or click "Start/Programs/Sonix/M2IDE V119/M2Asm119.exe" to enter the M2IDE, as shown in Figure 2-1:

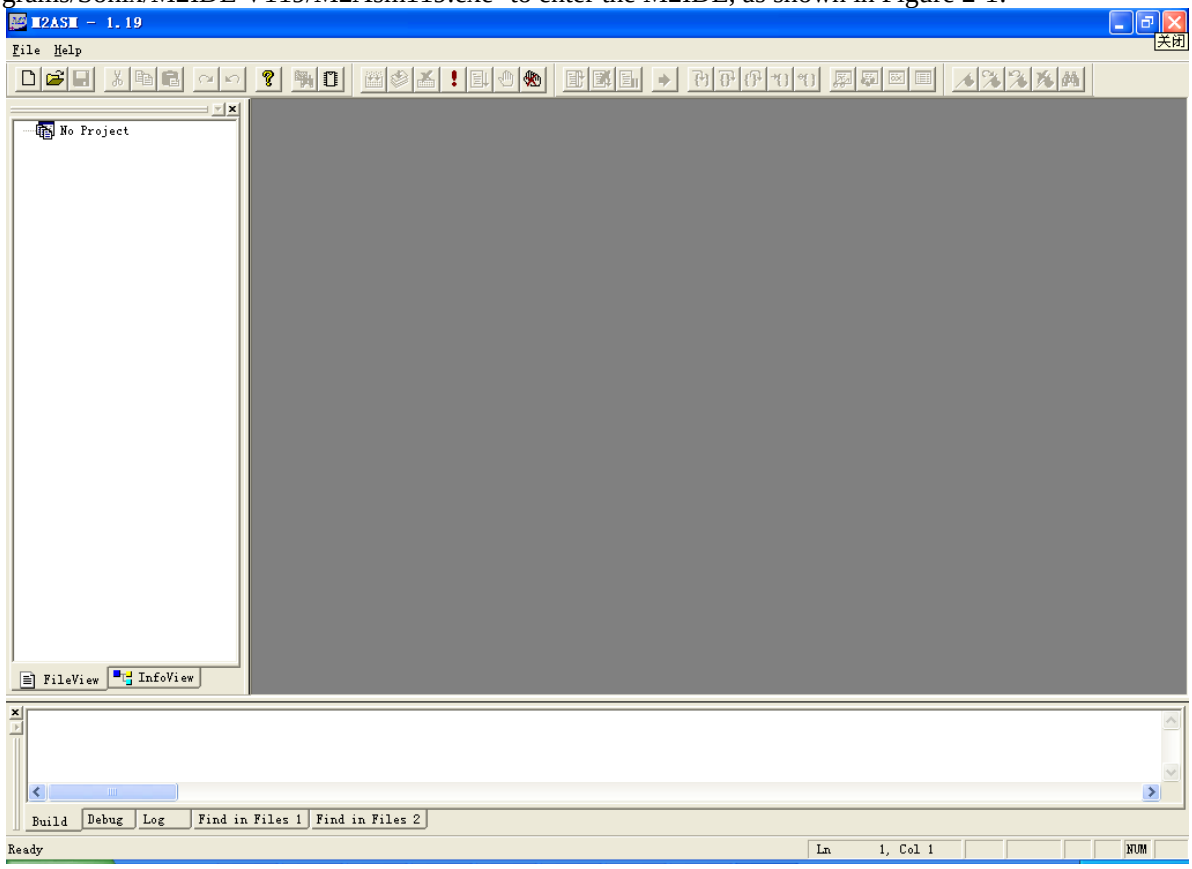

Figure 2-1. Integrated Development Environment

If it is the first time to use M2Asm, M2Asm will automatically open SN8Readme.txt and display the dialog box of Welcome.

At every upgrade of M2IDE compiler, the information of new compiler will be added in SN8Readme.txt, before using the compiler, you shall carefully read this specification.

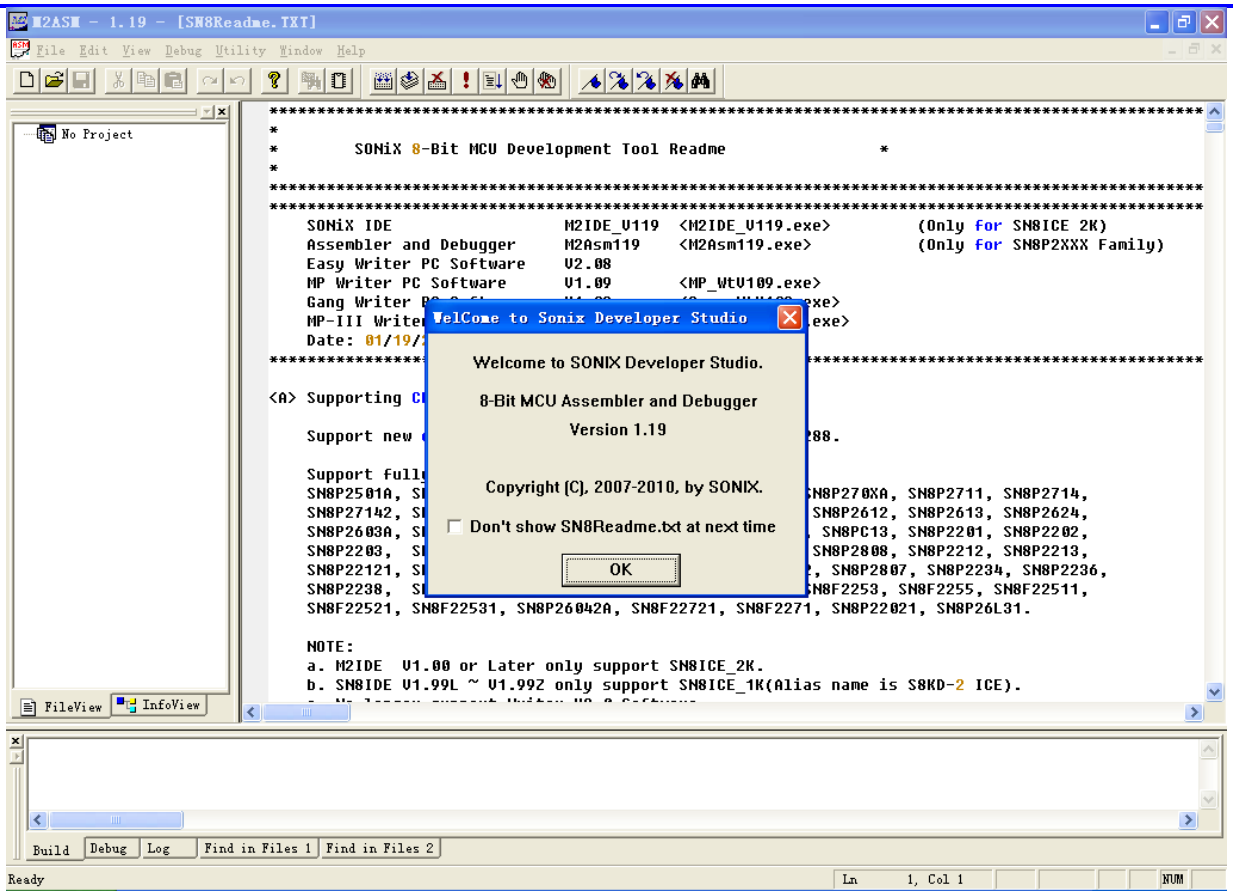

Figure 2-2. Open M2Asm Interface at First Time

If you do not want to show this image every time, you may select the small box in front of "Don't show SN8Readme.txt at next time" and then click OK.

In addition, you can also open SN8Readme.txt through the following methods:
1) To execute: Start/Programs /Sonix/M2IDE_V119/SN8Readme.txt;
2) View this file in the installation directory of compiler.

2.2.2 M2IDE Interface

As shown in the following figures, Figure 2-3 is the editing state, Figure 2-4 is the debugging state. A variety of debugging tools, commands, menus and running states are integrated into M2IDE.

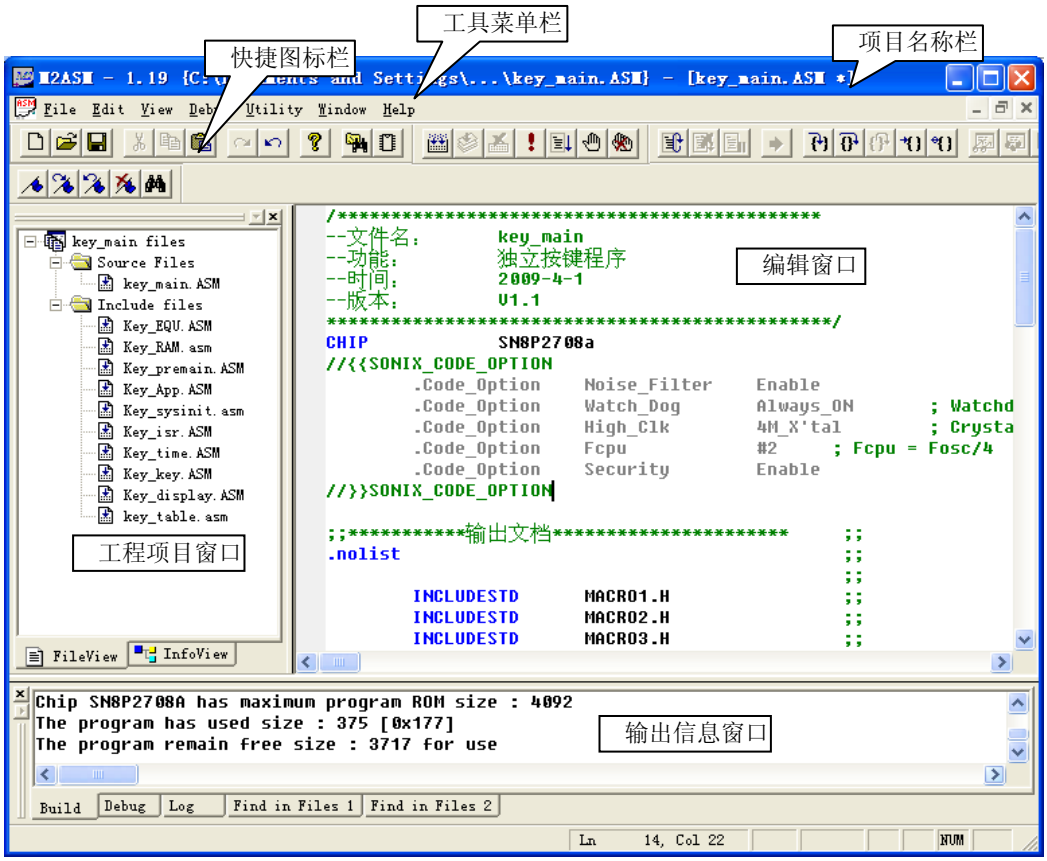

Figure 2-3. Editing Interface

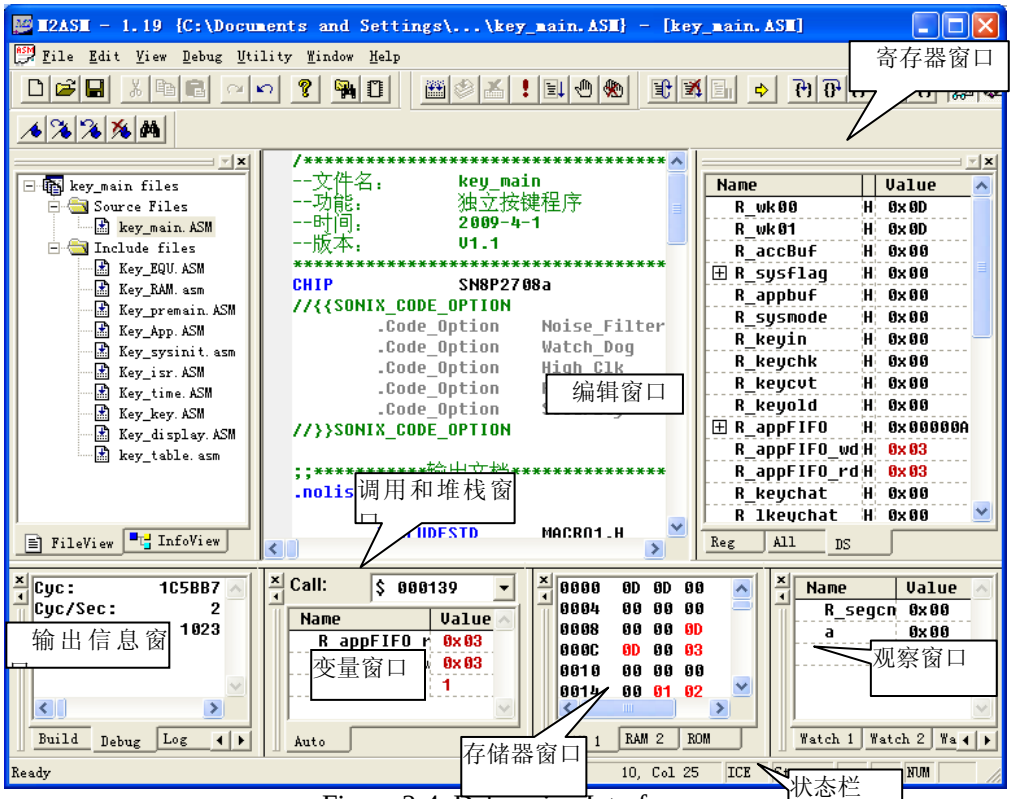

Figure 2-4. Debugging Interface

The brief descriptions of the role and function of various windows are in the following.

Menu bar

Menu bar is to provide a variety of operation menus, for example, creating project, editing, compiling program, linking and debugging and so on.

Shortcut icon bar

Shortcut icon bar is to provide a quick operation way for the frequently used tools, for example, creating a new file, compiling, running and so on.

Project name bar

Project name bar is to display the name and the storage path of the current project.

Editing window

Editing window is to input and modify the program.

Status Bar

It is to display the current debugging state, including running and stopped state, and the current row of program.

The information displayed in status bar is helpful in the compiling and debugging phase of program. For example, in the editing of program, to indicate the value of row and column, the data behind Ln means the current row of cursor, while Col means the current column of cursor, (the two fields on the far right of status bar) shows the current position of cursor.

Indicate "Simu ..." during compiling, and indicate "ICE Run ..." in runtime, and indicate "ICE Stop" when the operation is stopped.

The user can click right key on the blank part of toolbar or status bar, to select "Staus Bar" to display or close the status bar.

Project window

Project window is divided into two pages, Fileview page shows the information of current project group; Infoview page shows all the labels in the current project, and the program line can be marked, so it's convenient to the user to debug the program.

Output window

Output window is divided into five pages: Build, Debug, Log, Find in Files1 and Find in Files2.

Build page shows the information in the creation of project, including compiling, linking, checking, code size, warning, error, and other information.

Note:

Checksum: the user can perform Checksum via the Writer, to determine whether the information to be programmed into IC is correct.

IDE of SONiX is used to compile, after a successful compilation, some information about code will be given in the output information bar of IDE.

If the user starts the encryption option "Enable Security", after compilation it will generate two values of Checksum --- EPROM Checksum and Security Checksum; otherwise it will generate only one Checksum --- EPROM Checksum;

EPROM Checksum refers to the Checksum value of user's source code;

Security Checksum refers to the Checksum value of user's code under the encryption mechanism of SONiX;

Debug page shows the number of instructions executed by the program, the execution time and other information. Find in Files page is used to display all lists of a certain string in a specified file. Cyc: to indicate the instruction cycle to execute instruction;

Cyc/Sec: program is stopped after running, the rate of program execution can be roughly estimated according to this data;

Trace: to save the content of last executed 1024 instructions, press Ctrl + Shift + F11, to draw back the program and observe.

Variables window

Variables window is to show the current variable changed during operation in AUTO status. It is same as the observation window, to show the current value of variable, and the display formats of the two are the same, but it cannot be set in the variables window. This window will be shown up in debugging mode. The user can select Debug Windows via the View menu, after the drop-down menu is popped up, and then select Variables item to open or close the window (if the window is not opened, to execute this command can open the window; if the window has been opened, to execute this command can close the window). Generally, the window automatically displays the value of variable affected by the current instruction and the next two instructions, it's convenient to real-time track the variables in the current program.

Calls and stack window

Stack window is to display the use situation of stack and the stack-in functions, where it can determine whether the call of program is correct.

The address displayed in this window is the called entry address of running position of the current program, to select such address you can view which one is called.

The stack is increased hierarchy from 0 upwards, each stack-in operation make the number of stack layer to increase 1 (for example, CALL instruction or interrupt response, and each stack-out operation make the number of stack layer to decrease 1 (RET or RETI instruction). The entry address of current stack is displayed in Stack window, the outermost layer is shown in the lowest layer, and it will show $ 00000 if no stacks.

Calls and stack window is to display the use situation of stack and the address of stack-in subprogram or interrupted subprogram in the current running state. The user can determine the calling situation of program according to such information and further determine whether the current status is correct and there are any errors in calling.

Memory window

Memory window is to respectively display the current data and code in the data memory and program memory. The memory window of most MCUs has three pages: RAM1, RAM2 and ROM pages. However, there are some exceptions, for example, USB series MCUs have USB FIFO 0, USB FIFO 1, and SN8P2308 has LCD page, etc.

This window is only shown up in debugging. The user can click the button of “Memory” on the toolbar, or select Debug Windows via the View menu, after the drop-down menu is popped up and then select Memory to open or close the window.

Observation window

Observation window is to monitor the user-defined variables, namely to display the content of registers in data memory RAM. To facilitate the observation, the variables can be classified into four different viewing pages: Watch1, Watch2, Watch3 and Watch4.

The following three methods are used to add a variable to be monitored:

The first method is to use the mouse to click the blank part under Name of watch page in the observation bar, after the cursor is showing up and it is in the editing mode, you can add variables.

The second method is to double-click the variables to be monitored in the editing window, and then copy them to the Observation window, and you can see the current values of variables.

The third method is to select the variable, click right key of mouse, and then select “Add to Watch of xx” in the pop-up menu, and then the variables can be added into the watch.

Register window

Registers window is to display the registers and current value of variable defined by DS. The user can click the button of “Register” on the toolbar, or select Debug Windows via the View menu, after the drop-down menu is poped up, select Regirster to open or close the window. This window is divided into three pages, namely, Reg, All and DS, Reg is to display the most commonly used registers of system; All is to display all system registers; DS is to display the user-defined register variables, as shown in Figure 2-5.

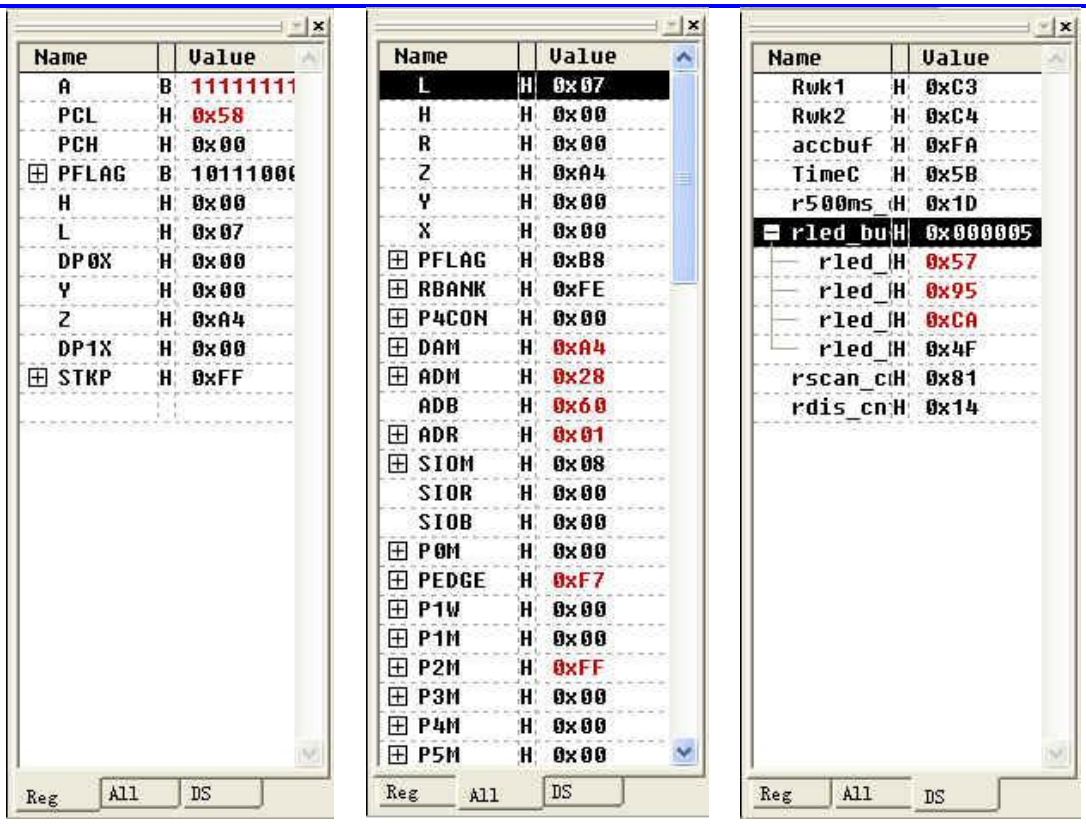

Figure 2-5 Register window

The functions of each menu are introduced in details in the following sections.

2.2.3 File menu（File）

File menu is mainly used to create a new file/project, or open a file/project, as shown in Figure 2-6.

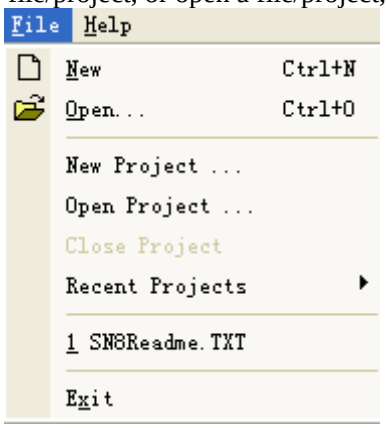

Figure 2-6. File Menu

- New: Create a new file;
- Open: Open an existing file;
- New Project: Create a new project file;
- Open Project: Open an existing project;
- Close Project: Close a project;
- Recent Project: Open the recently opened project file;
- 1 ~ 9: Open recently used source files or text files;
- Exit: Close/exit and prompt to save the file.
  When a source file is opened, there will be Close, Save, Save as ... and other options.
- Save: Save;

Note:

1.The underline of menu bar and each menu refers to the letter of shortcut of this menu, and the user press Alt + the corresponding letter to open the menu;
2.After this menu is opened, to directly press the corresponding letter can execute the corresponding function.

Examples:

If you want to open a . ASM file from the compiling interface, the steps are as follows:

Step 1: Open M2IDE (Figure 2-7);

Step 2: Press "Alt + F" in the compiler interface (at that time it will be not affected whether Caps Lock is enabled or not) (After execution, it is shown in Figure 2-8).

Step 3: If the file is recently opened, it will be displayed in 1-9, then directly press the corresponding letter; if not, press "O" key on the keyboard (after execution, it is shown in Figure 2-9);

Step 4: Select the file to be opened in computer.

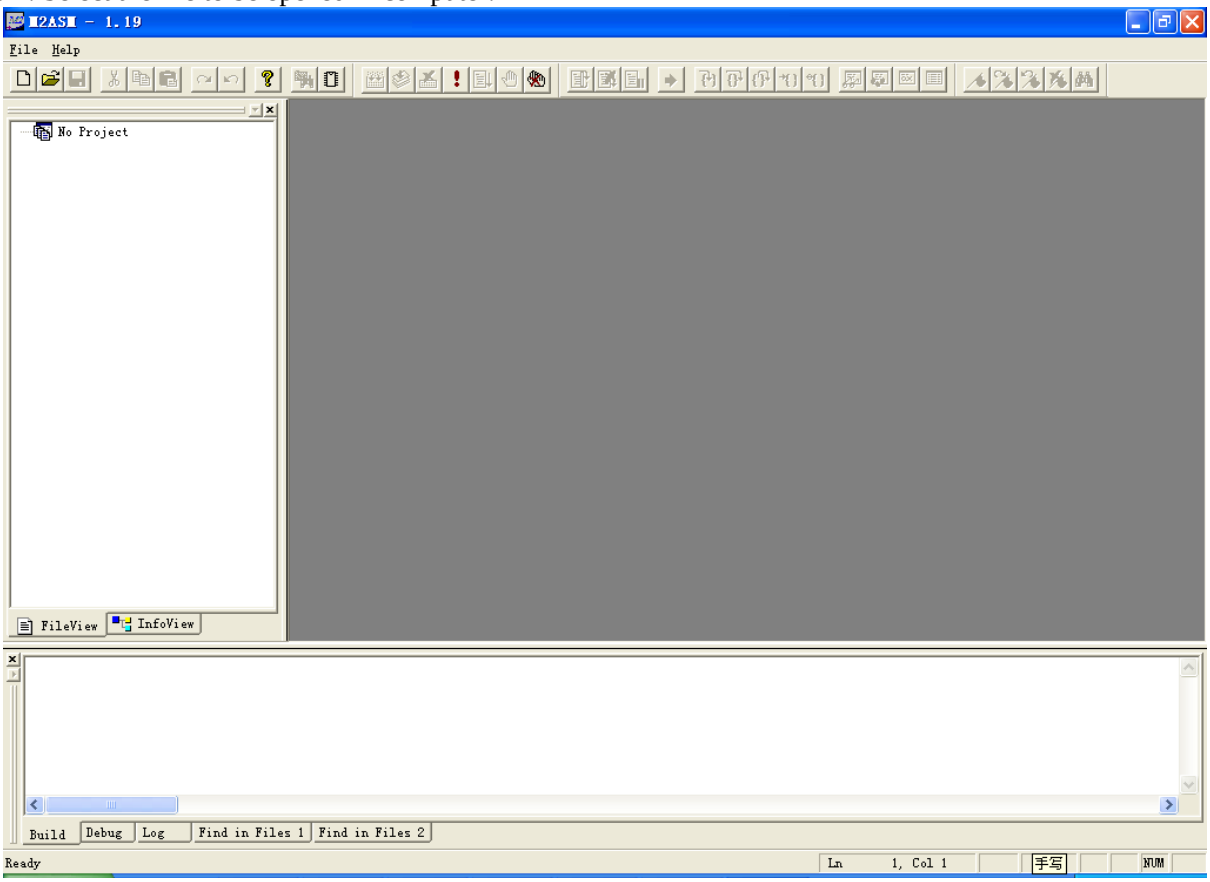

Figure 2-7 M2IDE Interface

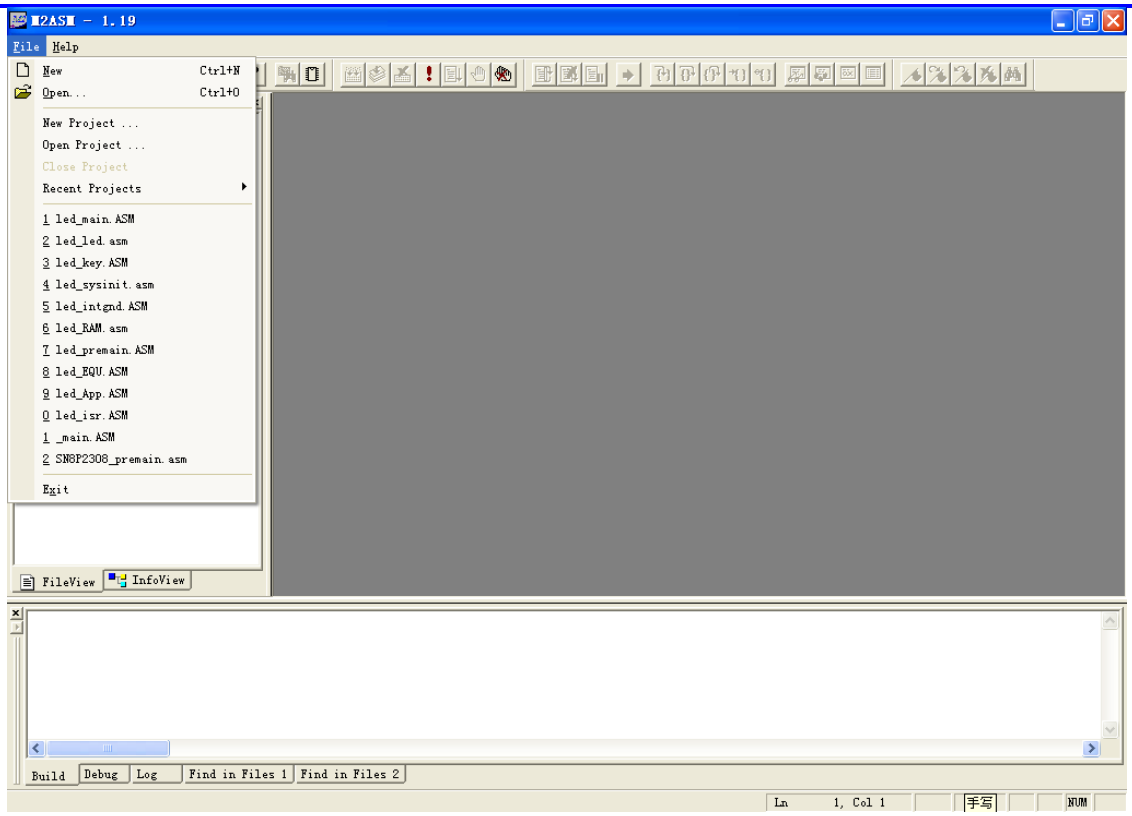

Figure 2-8 Open File Menu

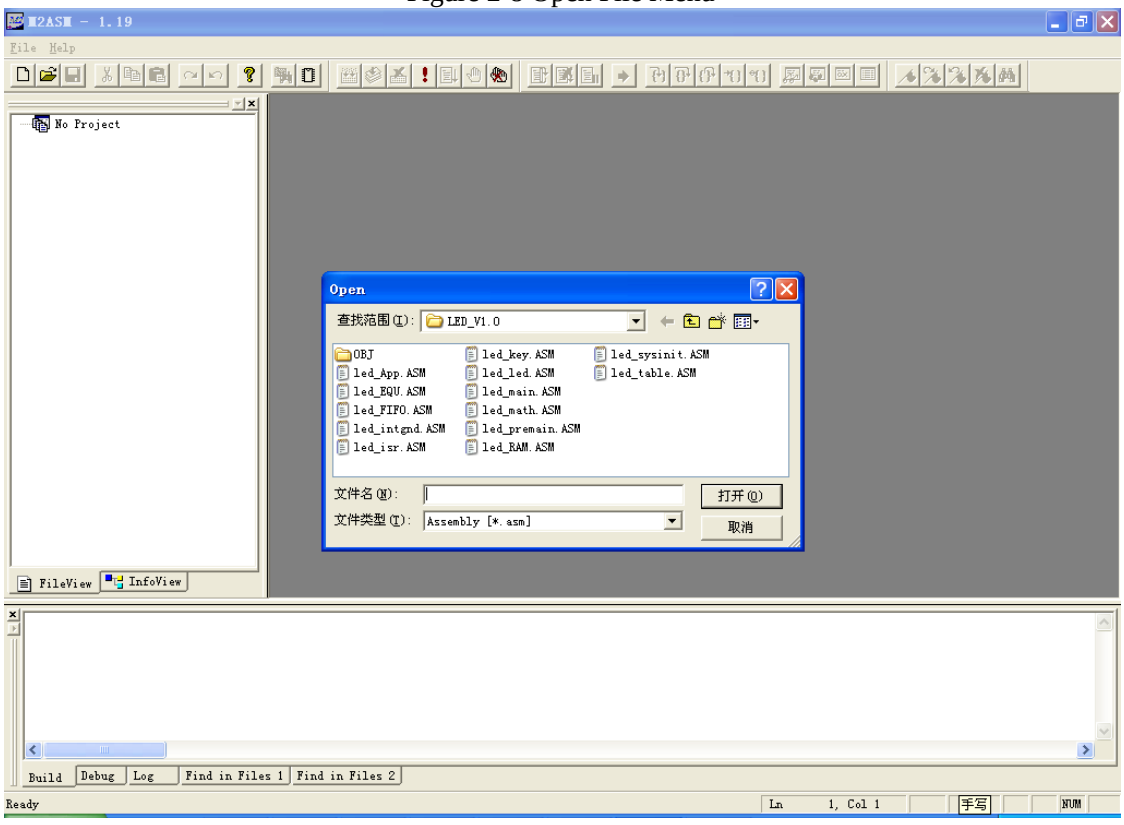

Figure 2-9 Select the File Name

So you can quickly and conveniently open a file.

In addition, after the File menu is opened, you can also use the arrow keys on the keyboard to select the function to be enabled, when you are using the arrow keys to select, the selected item will be highlighted, as shown in Figure 2-10:

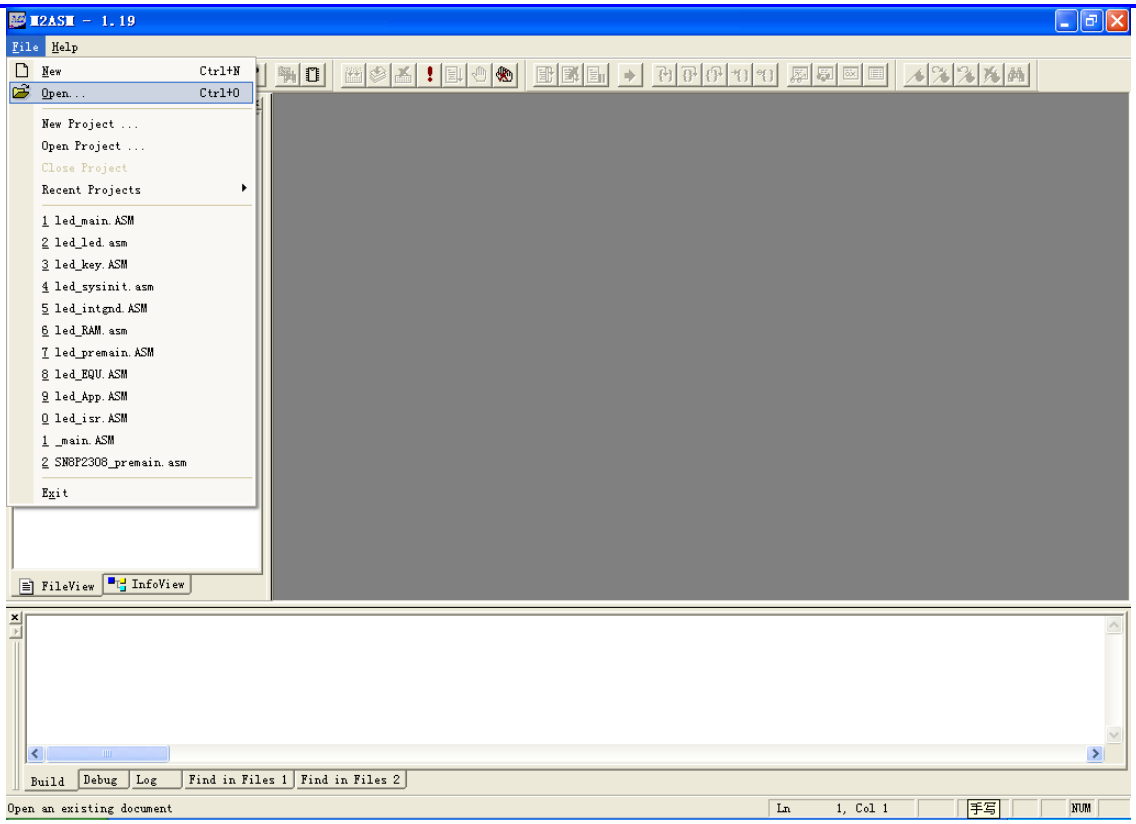

Figure 2-10 Use the Arrow Keys toSelect

Compared with using the mouse, to use the operation method of shortcut is more convenient and faster.

2.2.4 Edit Menu（Edit）

Edit menu is mainly to complete these operations to the program code, such as undo, redo, cut, copy, paste, select all, find and label operation and so on.

Figure 2-11 Edit Menu

- Undo ： Undo the last operation;

- Redo ： Restore the last operation;
- Cut：Cut;
- Copy：Copy;
- Paste：Paste;
- Select All：Select All;
- Find：to search in the current file;
- Find in Files：to search in all files;
- Replace：Replace;
- Prev BookMark：Previous mark;
- Next BookMark：Next mark;
- Toggle BookMark：Create a mark, to press again will cancel the current line mark of cursor;
- Clear BookMarks：Remove all marks.

The following examples are to introduce how to use the Find command in the menu.

To search in the current file:

If you want to find a register with the name of R_keyin given by the user in the current file, the steps are as follows:

Step 1: Execute Edit ---> Find command, or press the shortcut icon to find, and a dialog box will pop up, as shown in Figure 2-13;

Step 2: At that time you can find one by one in accordance with your needs (Find Next), or directly mark all lines with the line mark of R_keyin (MaskAll), together with Prev BookMark, Next BookMark, Toggle BookMark, Clear BookMarks and other operations to search.

After the execution of MaskAll, it is shown in Figure 2-14.

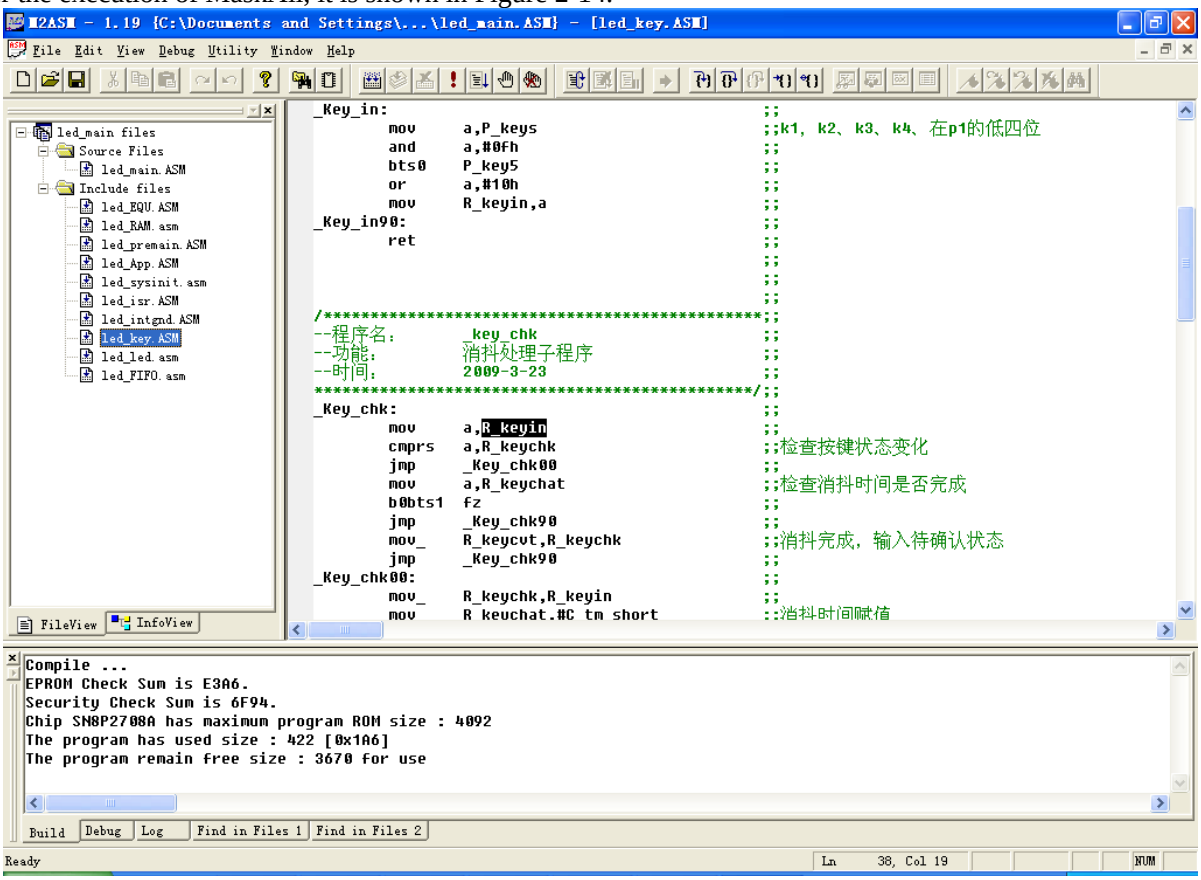

Figure 2-12 Find Register in the Current File

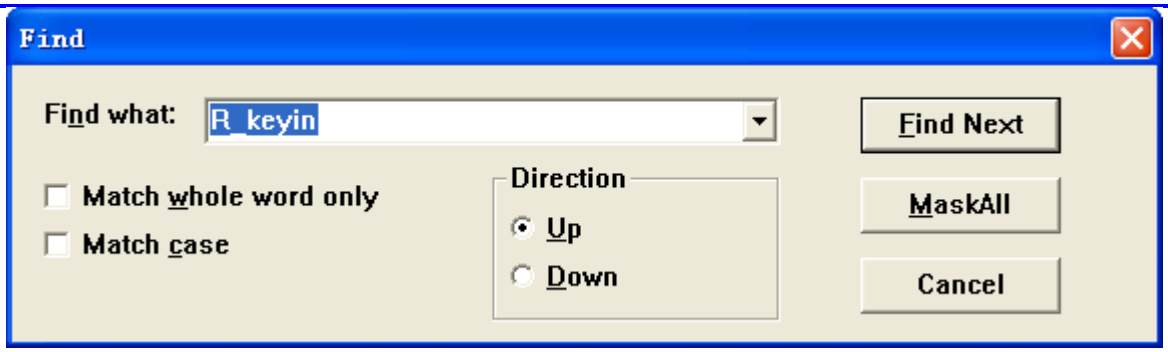

Figure 2-13 Information box to find register in the current file

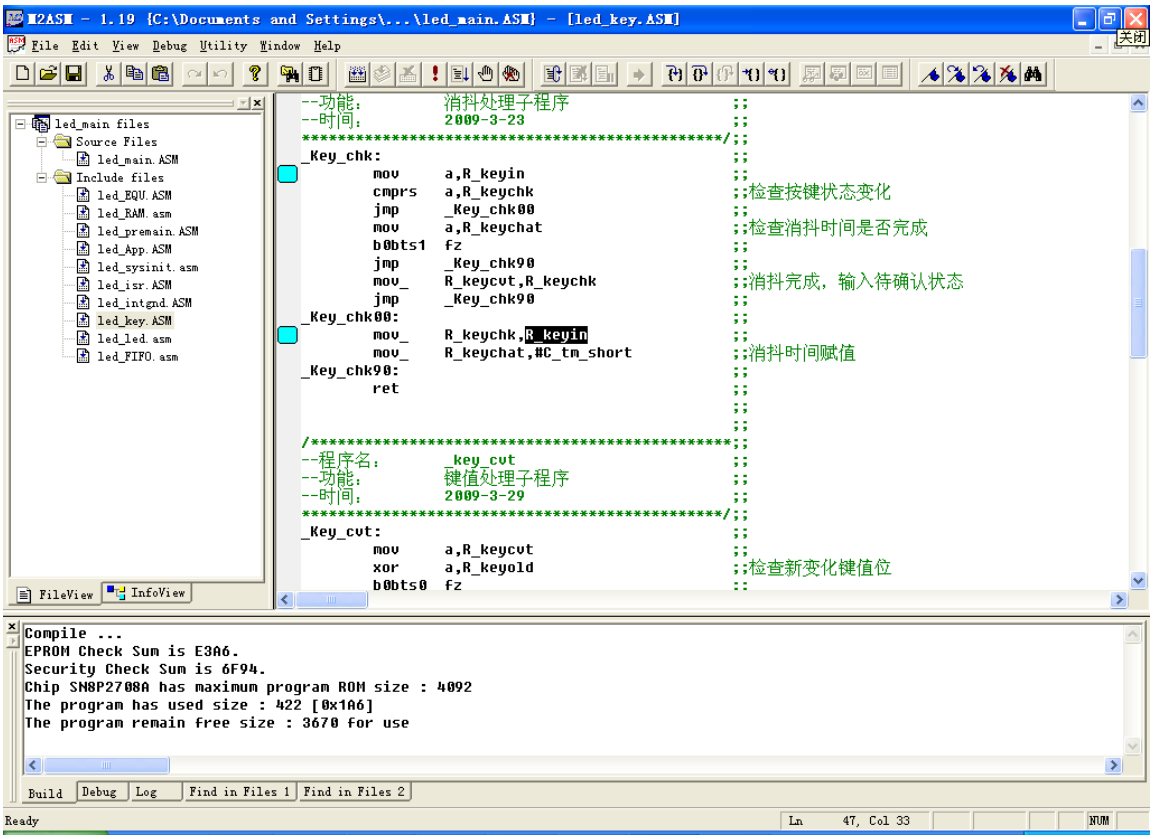

Figure 2-14 Compiler interface after the execution of MaskAll

Find in Files：

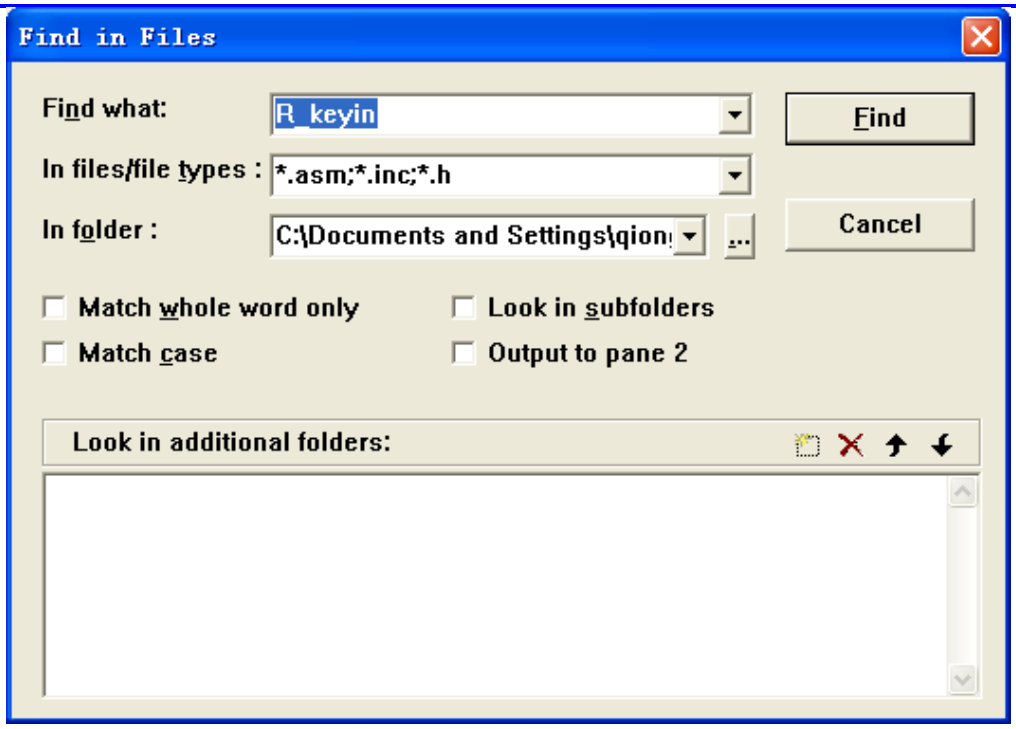

Figure 2-15 Information box to find in all files

Replace：

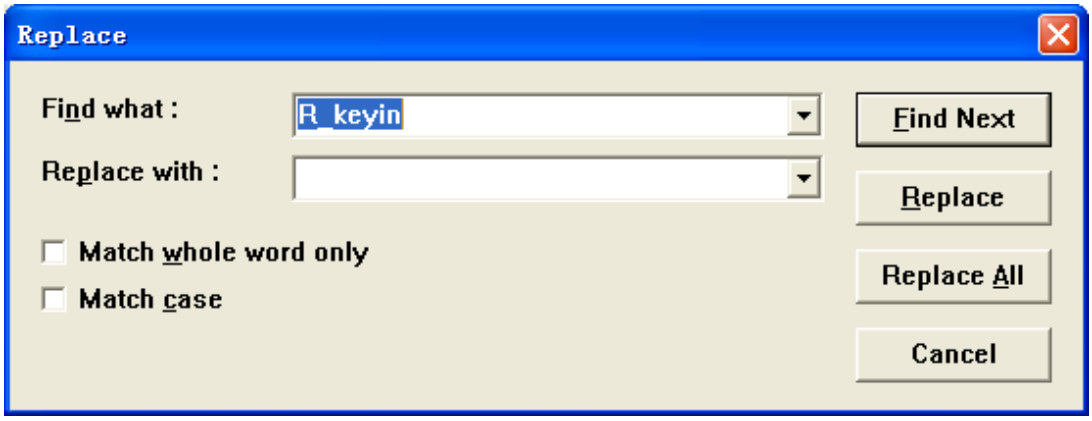

Figure 2-16 Replacement Information Box

Note:
There are two options in the pop-up dialog box of Find, Find in files and Replace:
Match whole word only and Match case
Of which,
Match whole word only means to completely match with the key letters and independently, otherwise the results are not displayed;
Match case means to be same with the combination of key letters.

2.2.5 View Menu（View）

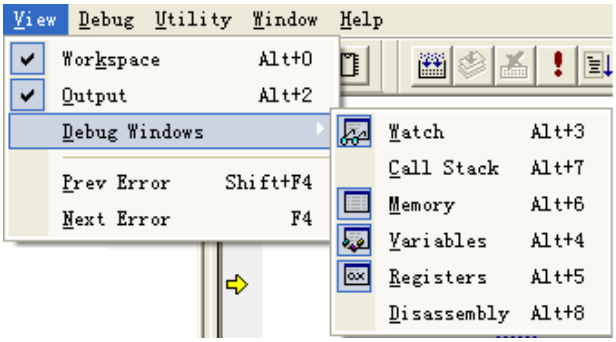

Figure 2-17 View Menu

- Workspace: If it is selected, it will show the project window, and it is selected by default;
- Output: If it is selected, it will show the output information window, and it is selected by default;
- Debug Windows: Debug window (its sub-menu will be normally displayed after the program is compiled and run for one time, and it is in gray in other cases);
- Watch: Observation window, and it is enabled by default after the program is run;
- Call Stack: Stack calling progression window, reserved;
- Memory: Memory window, and it is enabled by default after the program is run;
- Variables: Variables window, and it is enabled by default after the program is run;
- Registers: Registers window, and it is enabled by default after the program is run;
- Disassembly: Disassembly window, reserved;
- Prev Error: If the compiling is failed, the compiler will give error prompt, to execute this command is to look at the last error;
- Next Error: If the compiling is failed, the compiler will give error prompt, to execute this command is to look at the next error.

2.2.6 Debug Menu（Debug）

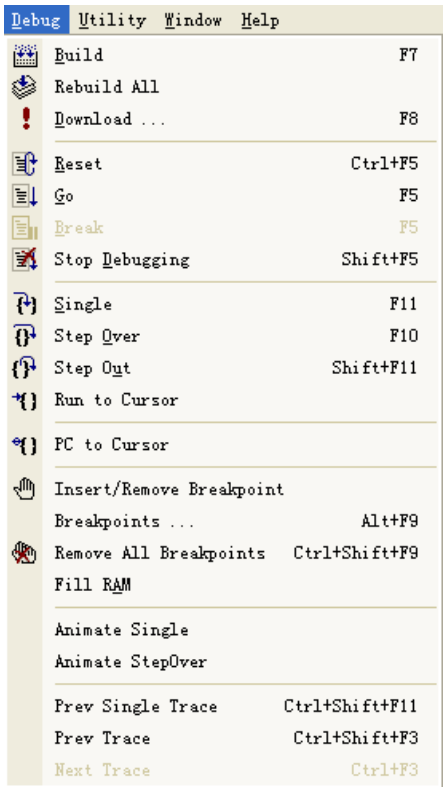

Figure 2-18 Debug Menu

- Build: Compiling program;
- Rebuild All: Recompile all (This option is effective during the building of project);
- Stop compiling, under normal circumstances the icon is gray (in the commonly used icon bar);
- Download ...: To execute this command the program will be automatically downloaded to the FPGA of emulator;
- Reset: Reset program;
- Go: Run program;
- Break: Pause the current running program;
- Stop Debugging: Stop debugging to return the post-compiling state;
- Cursor: The cursor is to display the next state and point to the current instruction (in the commonly used icon bar);
- Single: The shortcut icon bar is also known as Step Into command, which is run by single-step. It will only perform one instruction once and then it will stop, however, if the CALL program instruction is executed, it will enter the subprogram and stop at the first instruction of the subprogram;
- Step Over: Skip over the operation of functions. It will only perform one instruction once and then it will stop, however, if the CALL program instruction is executed, it will not enter the subprogram but stop at the next instruction after the CALL instruction; and all instructions in this subprogram will be executed, and the content of register and status register will be changed in accordance with the results of execution.
- Step Out: The end of operation. It can be used only when the simulation is stopped in the subprogram, it will execute all instructions between the current stopped program line and RET instruction (including RET instruction), and then it will stop at the next instruction after the CALL instruction;
- Run to Cursor: Run to the location of cursor (with memory, the user can press Ctrl + Shift + F11 to return to the previous instruction and view). To execute this command allows the program to run to the location of cursor in code window. This is equivalent to use the location of cursor as a temporary breakpoint.
- PC to Cursor: Point to the location of cursor (without memory). To execute this command allows the program counter to point to the current line of cursor, please note that it only modify the program counter PC, and no commands are executed.
- Insert / Remove Breakpoint: Insert/remove a breakpoint;

- Breakpoints ....: Breakpoint setting;
- Remove All Breakpoints: Remove all breakpoints;
- Fill RAM: Write data into a register;
- Animate Single: Automatically run at single-step. If a call instruction is encountered, it will enter the subprogram and then execute the first instruction in call program;
- Animate StepOver: Skip function automatically. If a call instruction is encountered, it will not enter the subprogram but execute the next instruction after the CALL instruction; and all instructions in this subprogram will be executed, and the content of register and status register will be changed in accordance with the results of execution.
- Prev Single Trace: Trace the last statement;
- Pre Trace: Trace to the previous 64 instructions;
- NextTrace: Trace to the next statement;

Note:
Step out command can only be used when the current stopping point is within the subprogram, otherwise it will cause unpredictable result.

Breakpoints….:

If you want to use this command, you must set a breakpoint at first; the dialog box for the execution of this command without breakpoint is as follows:

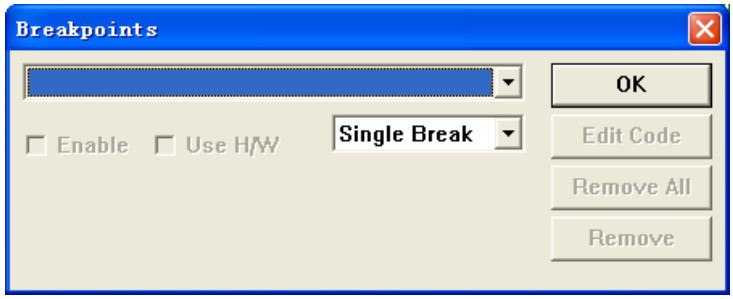

Figure 2-19 Breakpoints dialog box for the execution without breakpoint

After the breakpoint is set, the dialog box for the execution of this command is as follows:

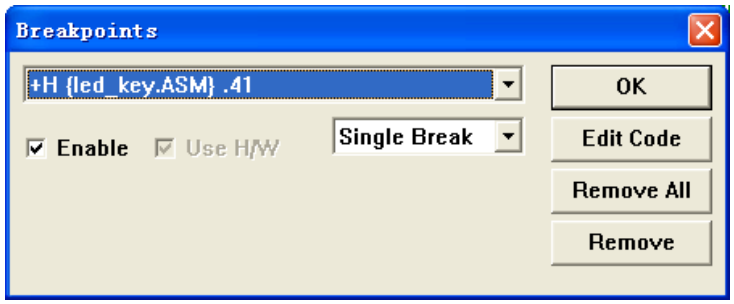

Figure 2-20 dialog box for the execution with breakpoint

If multiple breakpoints are set in file, you can select each breakpoint (Figure 2-21), and can choose the trigger mode (Figure 2-22).

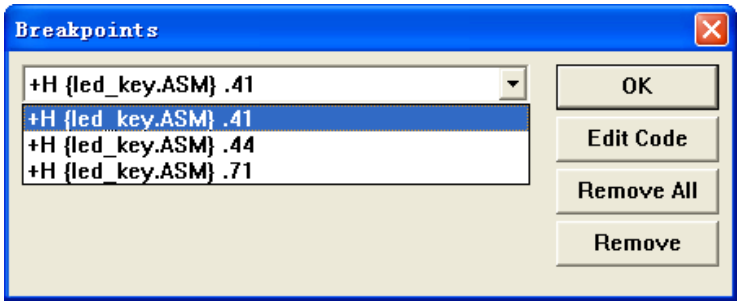

Figure 2-21 Dialog box for setting multiple breakpoints

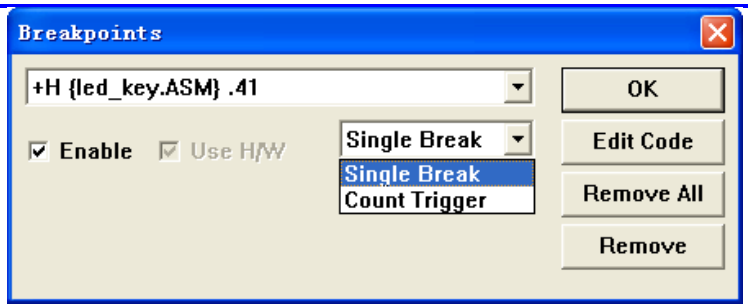

Figure 2-22 Dialog box for the selection of trigger mode

After the number of trigger is set, it will be shown as the following figure, taking 40 times as example:

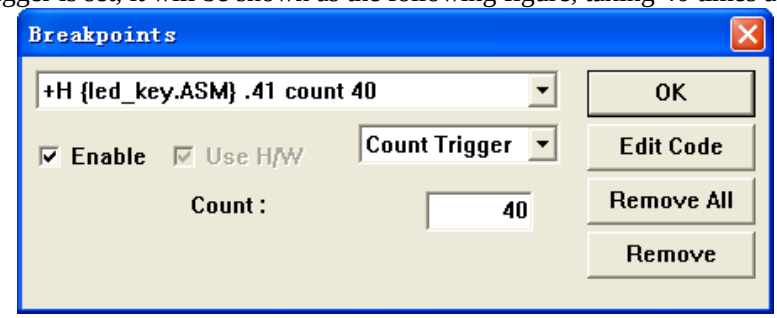

Figure 2-23 Dialog box after setting the number of trigger

Download……:

After downloading, the program can be run freely without the computer environment. After this command is used, it will pop up a dialog box for file selection, as shown in the following figure, to select. SN8 file to be downloaded, and click Open, and then it will pop up a success prompt box, as shown in the following figure, to prompt the user that the file is successfully downloaded and MCU is free running.

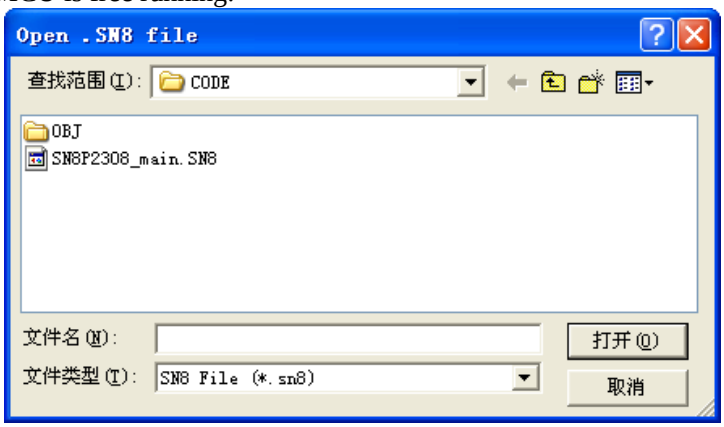

Figure 2-24 Dialog box when the program is being downloaded to ICE

Select. SN8 file to be downloaded

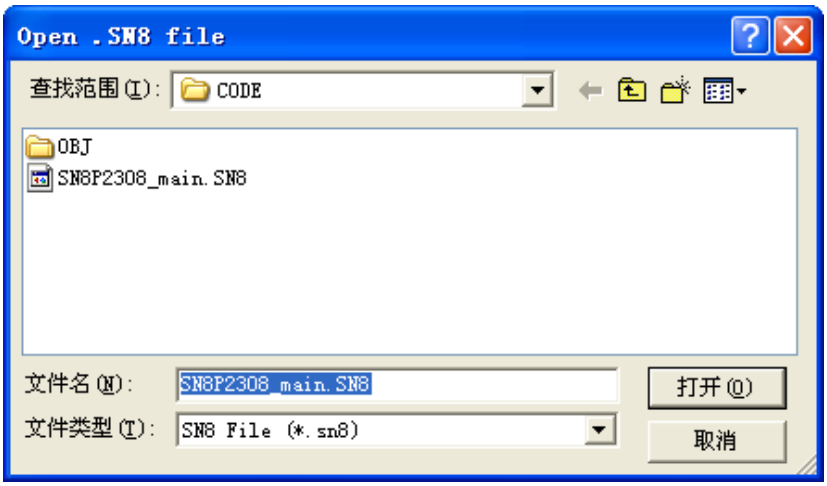

Figure 2-25 Select the program to be downloaded

After a successful download, the compiler will give a prompt for successful download, as shown in the following figure:

Figure 2-26 Successful download prompt box

2.2.7 Utility Menu（Utility）

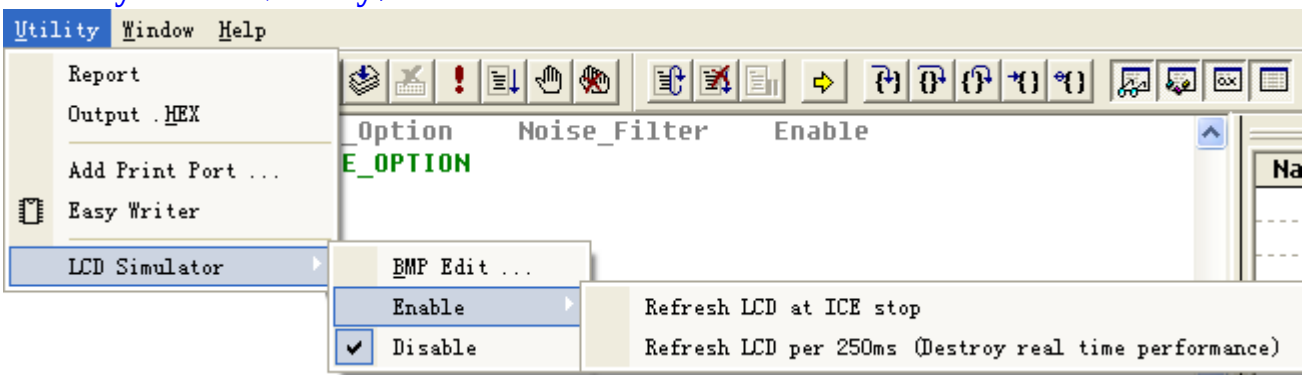

Figure 2-27 Utility Menu

- Report: Translate .SN8/.BIN file to. RPT file;
- Output HEX: Translate .SN8/.BIN file to. HEX file;
- Add Print Port: Add a printer port for the application of emulator;
- Easy Writer: Cooperate with Easy Writer to program chips;
- LCD Simulator: Simulation LCD function.

After the corresponding command is executed, it will pop-up the following dialog box:
Report：

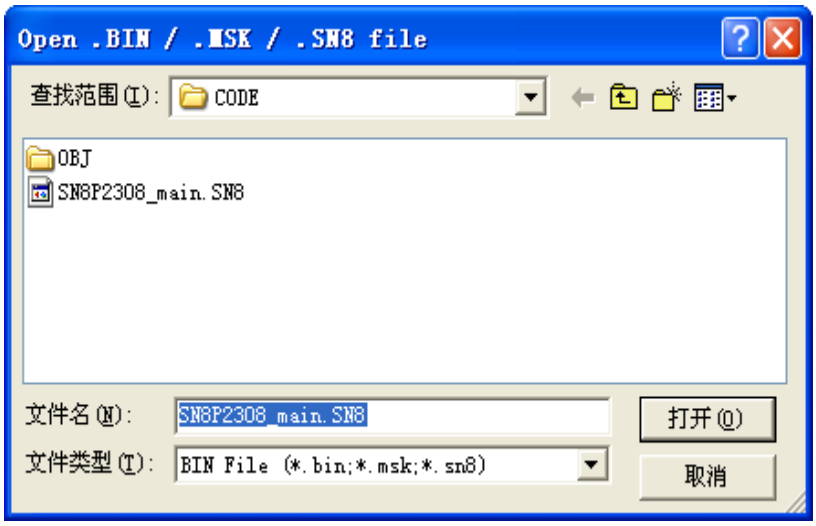

Figure 2-28 Dialog box after the execution of Report command

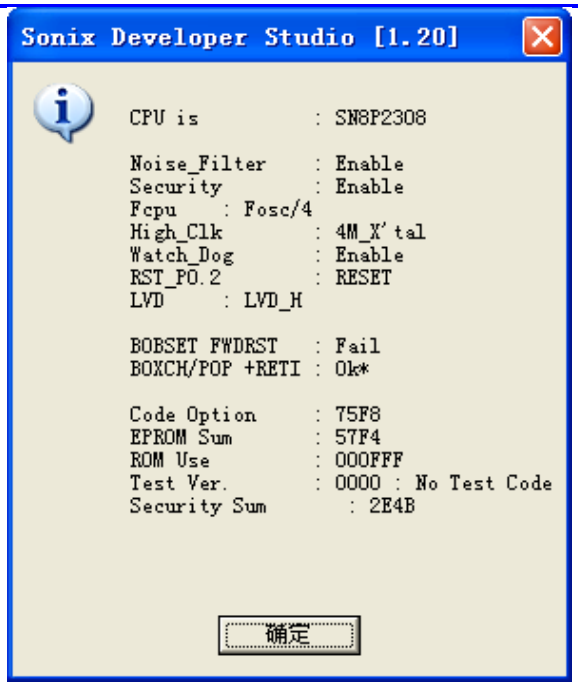

Figure 2-29 Information box after the execution of Report command

Output.Hex：

After the command is executed, in the source file directory it will automatically generate a SN8P2308_main.HEX file.

Easy Writer：

After the command is executed, it will pop up an information box, as shown in Figure 2-30, click OK to enter the programming interface of Easy Writer.

Figure 2-30 Prompt box after the execution of Easy Write command

LCD Simulator：

Please refer to **2.4** .

2.2.8 Window Menu（Window）

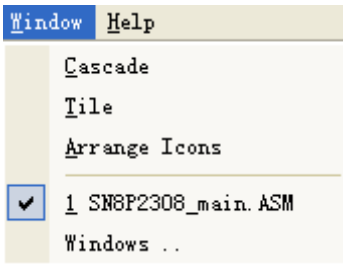

Figure 2-31 Window Menu

- Cascade：Set the cascading of windows;
- Tile：Set the tiling of windows;
- Arrange Icons：Arrange Icons on the bottom of the window.

After the execution of each command, the figures are shown as follows:
Cascade：Set the cascading of windows;

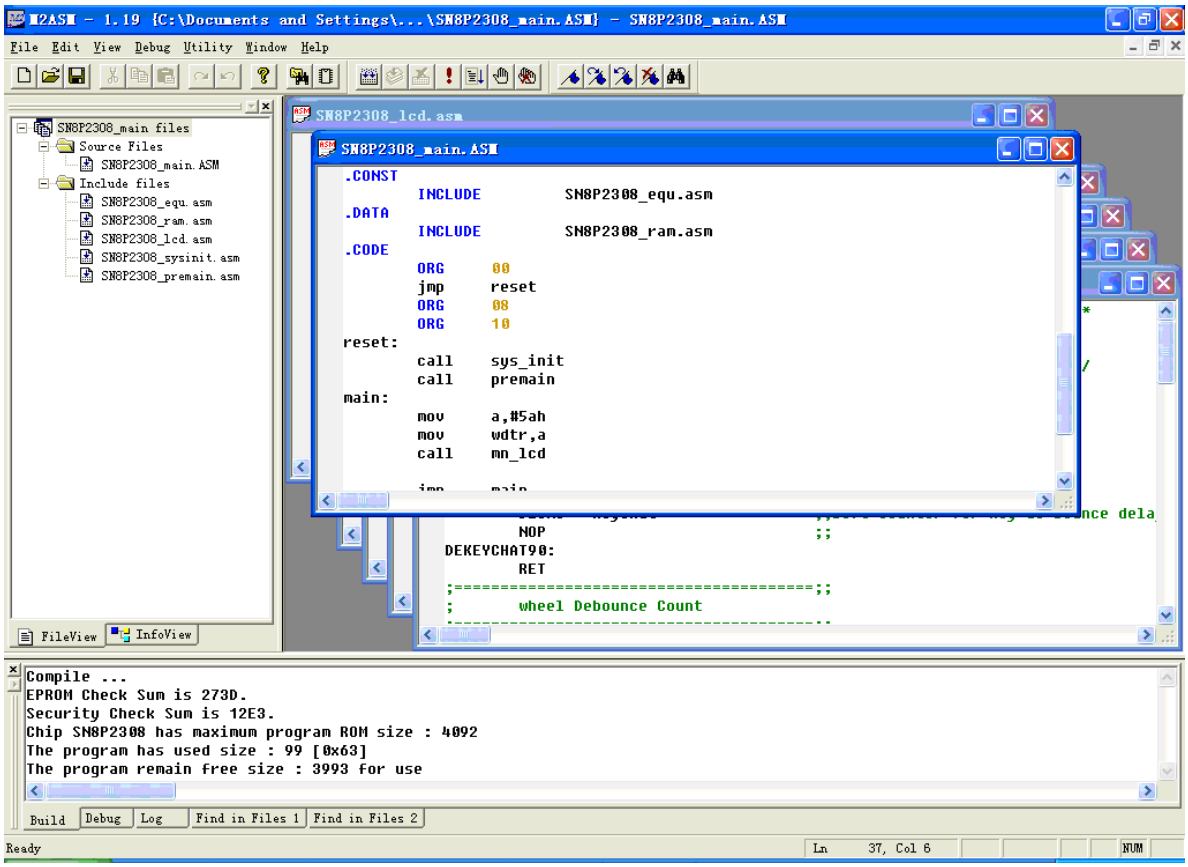

Figure 2-32 Windows cascading

Tile：Set the tiling of windows;

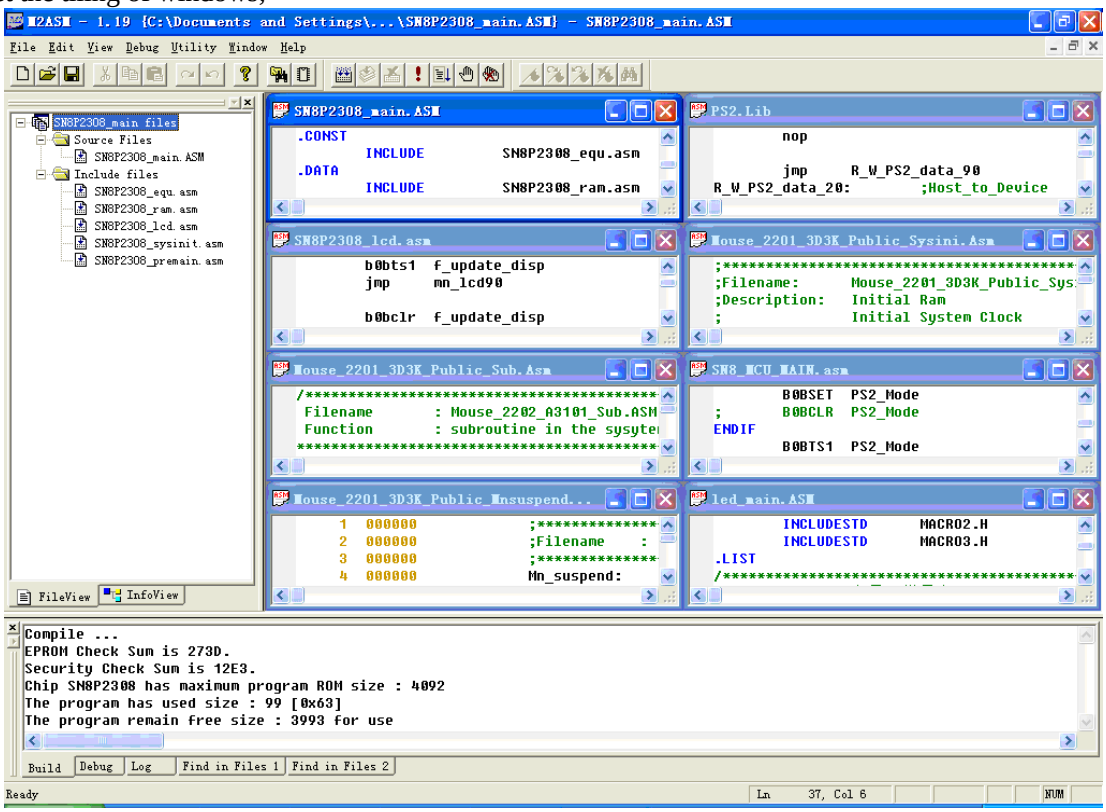

Figure 2-33 Windows tiling

Windows：

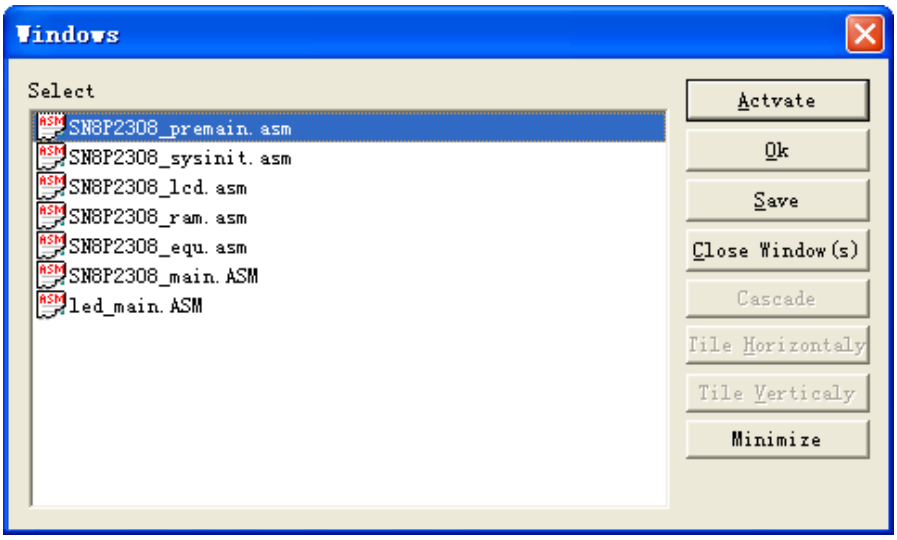

Figure 2-34 Window

2.2.9 Help Menu（Help）

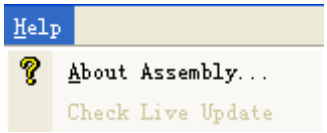

Figure 2-35 Help Menu

-  About Assembly： Help.

Please refer to Appendix II for the commands, tools and shortcuts of menu.

2.2.10 Windows Management

Note:

1.Under M2IDE interface, all windows can all be staying at anyplace and the user can move it around in the main window.

2.All windows can all be separately popped up for easy operation (e.g. turning off) and observation by double-clicking the edge of window, if double-click again, it will return to the compiler interface.

3.Click right key on the blank of toolbar or status bar, you can select whether the corresponding window shall be displayed or not in the pop-up menu.

The examples are respectively introduced in the following.

1. In the compiler interface, all windows have a mark of two white vertical or horizontal lines, which is called activity bar, as shown in the figure:

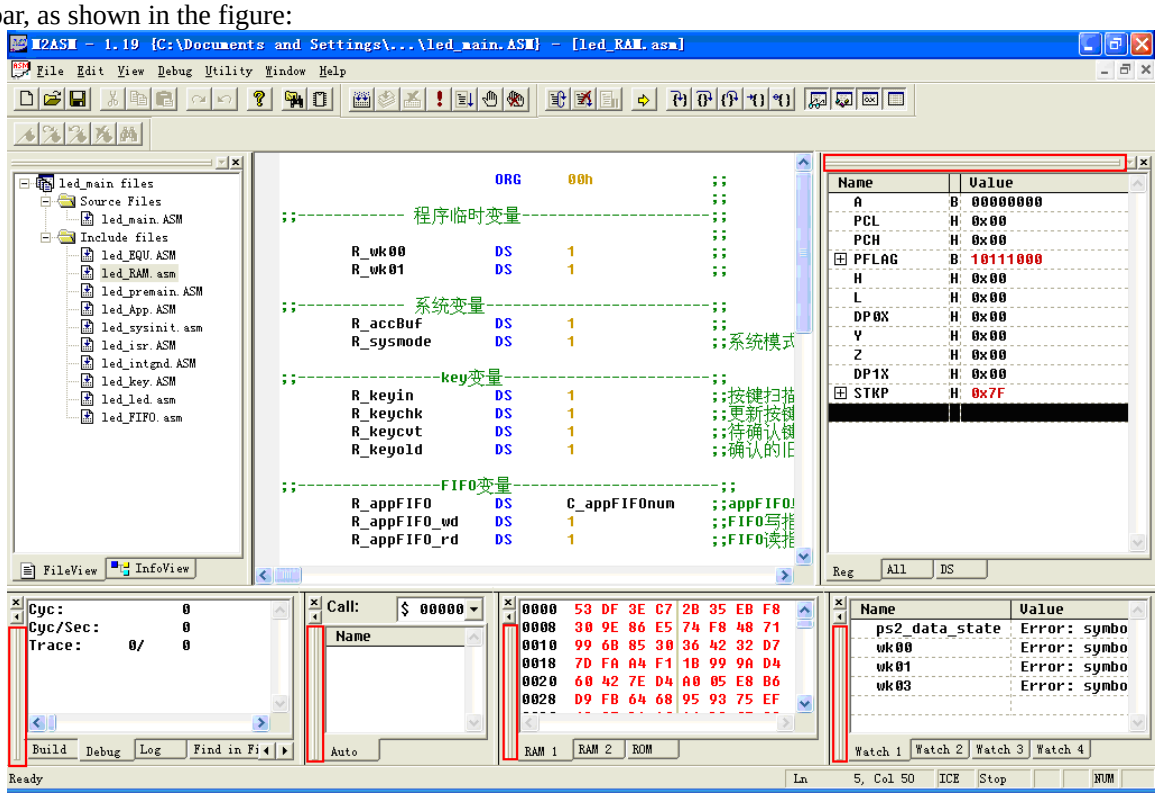

Figure 2-36 Activity bar of each window

Click these activity bars, and then hold to drag, when the window becomes a rectangle box with a black border, it can be moved to other parts of the interface (as shown in Figure 2-37).

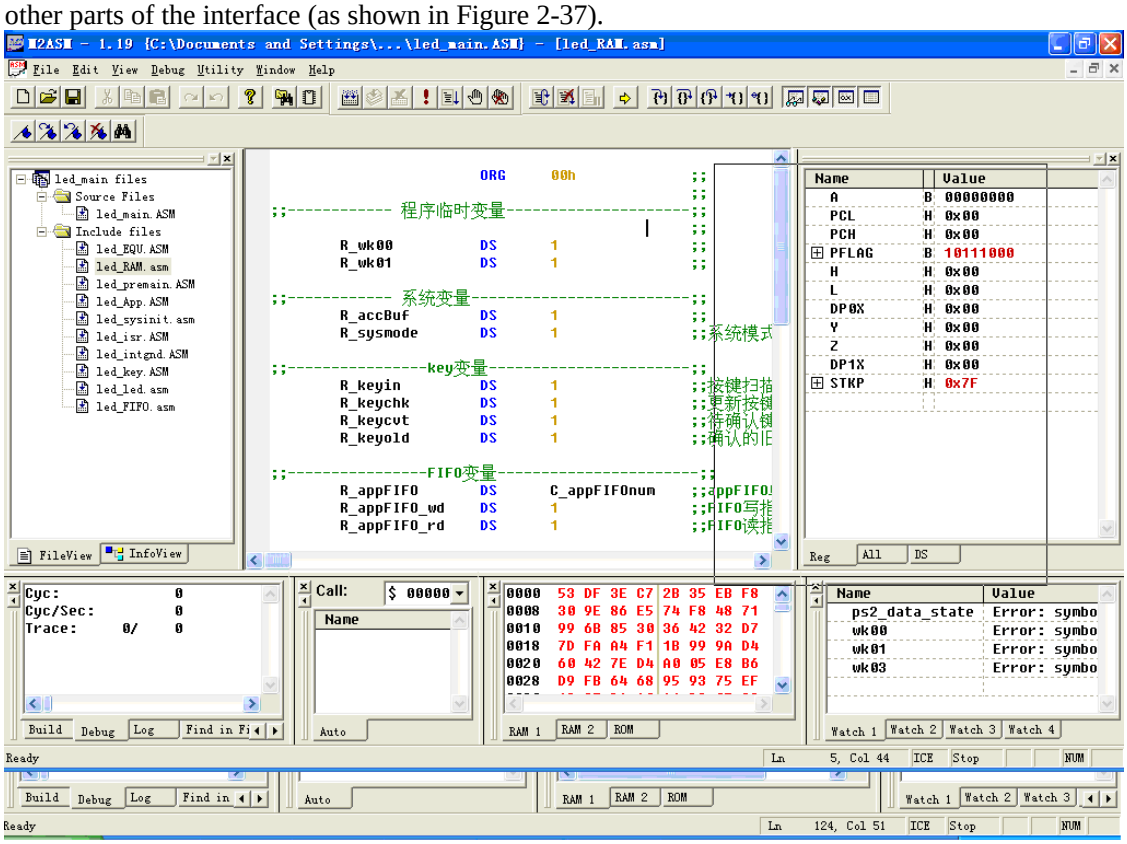

Figure 2-37 Window is moving

2. Similarly, double-click the activity bar, the window will pop up at once, as shown in the figure:

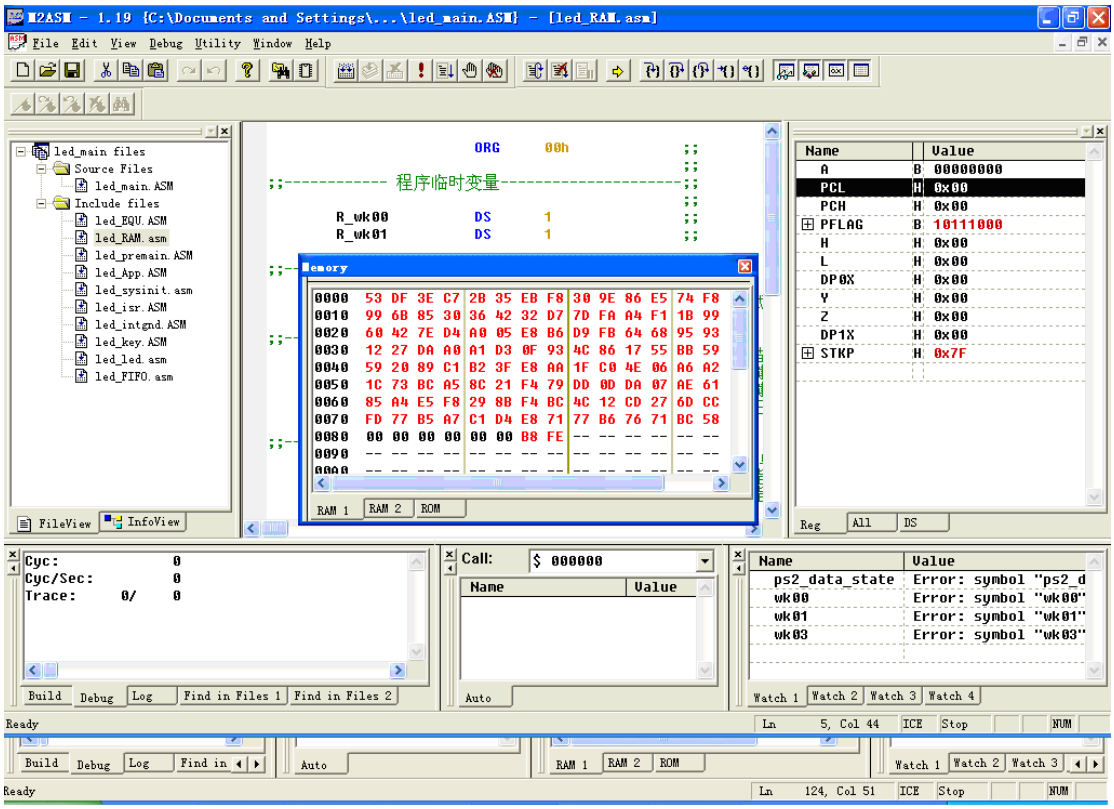

Figure 2-38 Double-click the activity bar to pop-up window

3. Click right key on the blank of toolbar or status bar, you can select whether the corresponding window will be displayed or not by the selection of Output, Watch, Variables, Registers, Memory, Workspace and so on in the pop-up menu, as shown in Figure 2-39:

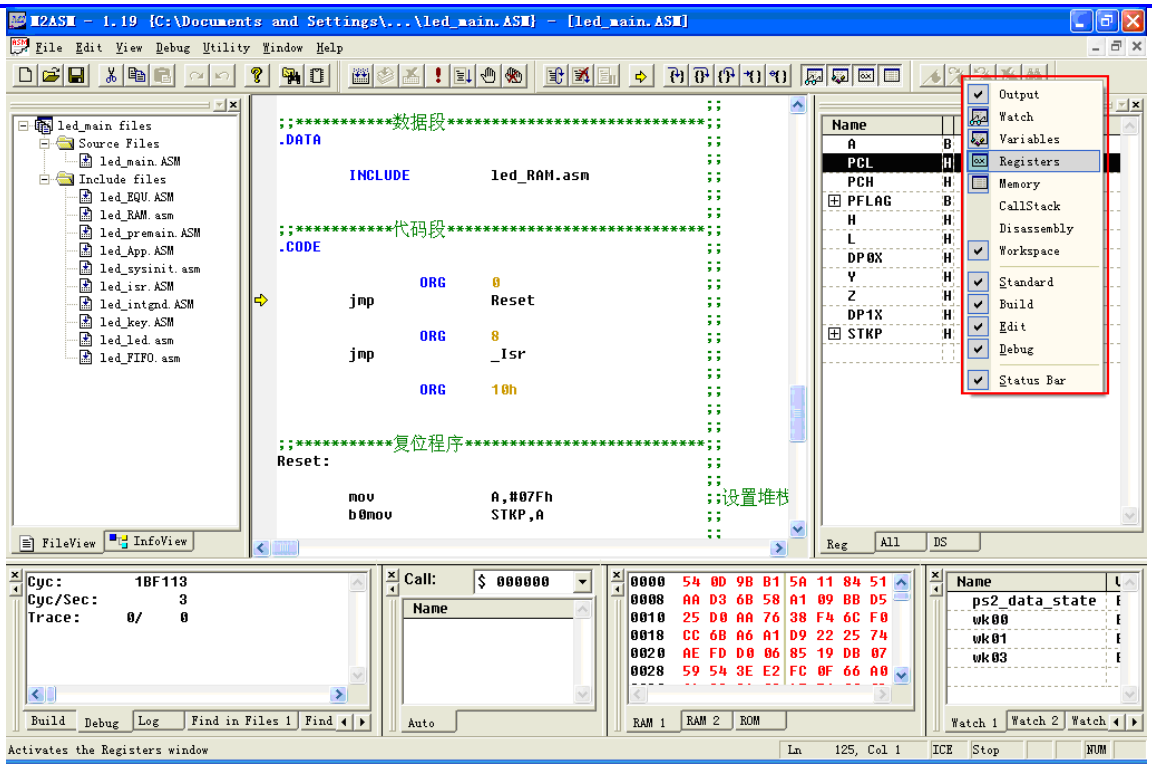

Figure 2-39 Right-click the blank of toolbar to choose to display or not display the window

As the above figure shows, the little black checkmark in front of name or to press the icon is to indicate that the window is already opened.

2.3 Create and Debug Application Programs

2.3.1 Create a Project /New File

The project mode is adopted to manage files in the M2IDE. All files including source code (including the main program and included files), header files, and descriptive documents, can be put in a same project file for unified management.

The general steps to create an application program are as follows:

- Create a new project file;
- Create a source file and enter the program code;
- Save the created source file into the project folder;
- Add the source code into the project.

The following is taken the creation of a new project file Test.PRJ for example, to introduce how to create an application program in details.

Double-click the shortcut icon of M2Asm119 on the desktop, open the development environment, if it is the first time to use M2Asm, M2Asm will automatically open SN8Readme.txt and show the Welcome dialog box, as shown in Figure 2-40. Please select “Don't show SN8Readme.txt at next time” on the dialog box and click the button of “OK”, so it will not show SN8Readme.txt at next time when you open M2Asm. If it is not the first to use the development environment, when M2Asm is started, it will open the last correctly processed project.

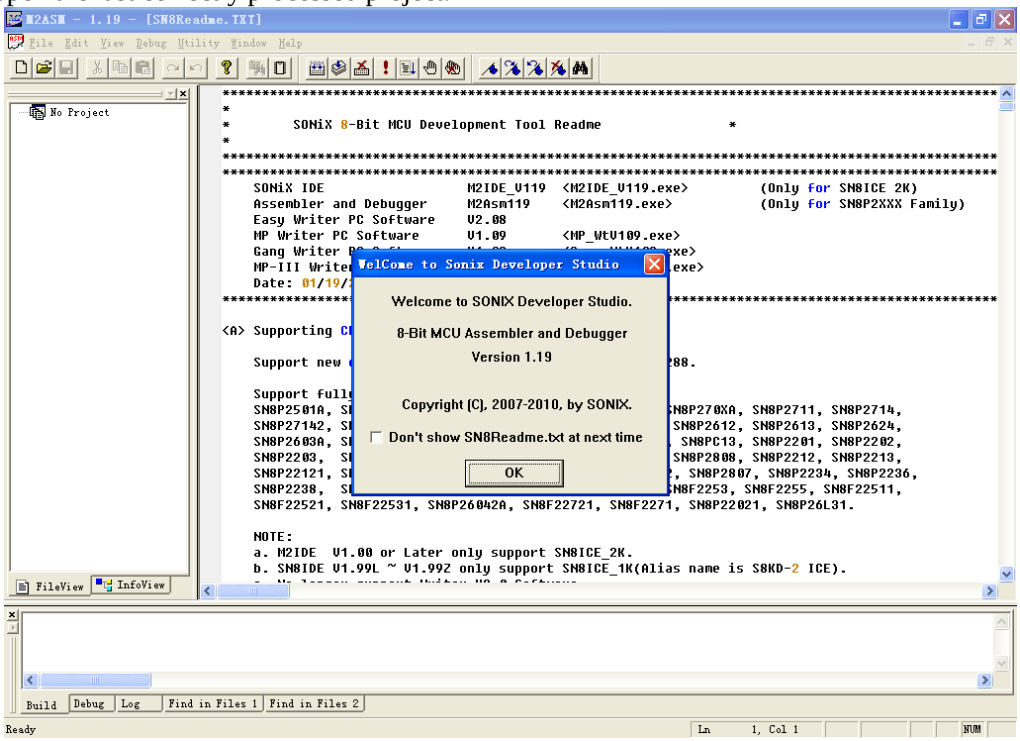

Figure 2-40 Open M2Asm interface at first time

Close all files and projects, M2Asm interface is shown in Figure 2-41, you can see that the project window, edit area and compiling information bar are empty, and a lot of buttons on the toolbar are in gray lock state.

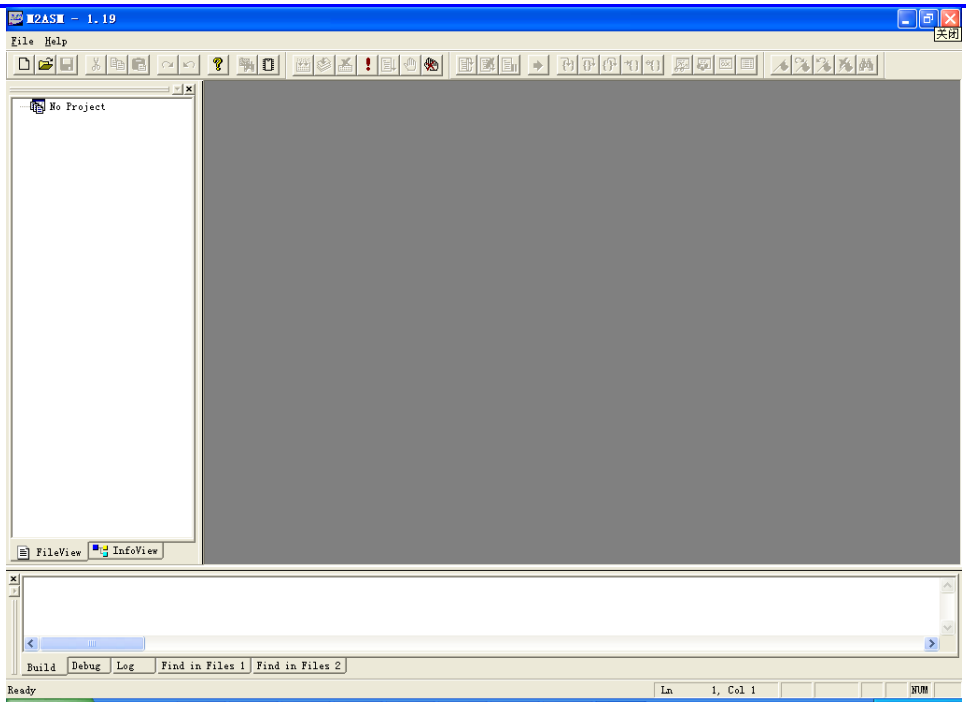

Figure 2-41 M2Asm interface without opened project or file

If you want to build a project, at first you shall create a file to add project, click the button of "New" on toolbar or "File" on menu bar to pop up a menu, as shown in Figure 2-42, select the option of "New", to successfully create a blank document, and the default file name is SONIX1, and it will create multiple files with multiple use of "New", and the file names are as follows: SONIX2, SONIX3 and so on.

Use the item of "File" on menu bar, the pop-up menu is shown in Figure 2-43, you can find that the menu at that time is different from the one at idle time, select the option of "Save", and a file saving dialog box will be popped up, as shown in Figure 2-44.

The following steps shall be completed:

- Rename a file name, the file name shall reflect the functions of file as far as possible;
- Select the save path for the file, it is recommended to save the file and project under the same directory, and the files of the project shall be saved in the same directory, to facilitate management; here the save path is as follows: D: \experimental procedure\Test, and the file name is Test, click "Save" to return. Then you can find that the file name on the title bar has been changed to Test.ASM.
- Use "File" menu in Figure 2-42 again, select the option of "New Project", the File Open dialog box will pop up, as shown in Figure 2-45, select the newly created Test file, click to open, and thus a project is successfully created. Project name is the same with the name of the just opened file, and both of the names are Test, meanwhile Test.ASM file is automatically included in the project file. The path to open current project and the name of current file are displayed in Title bar.

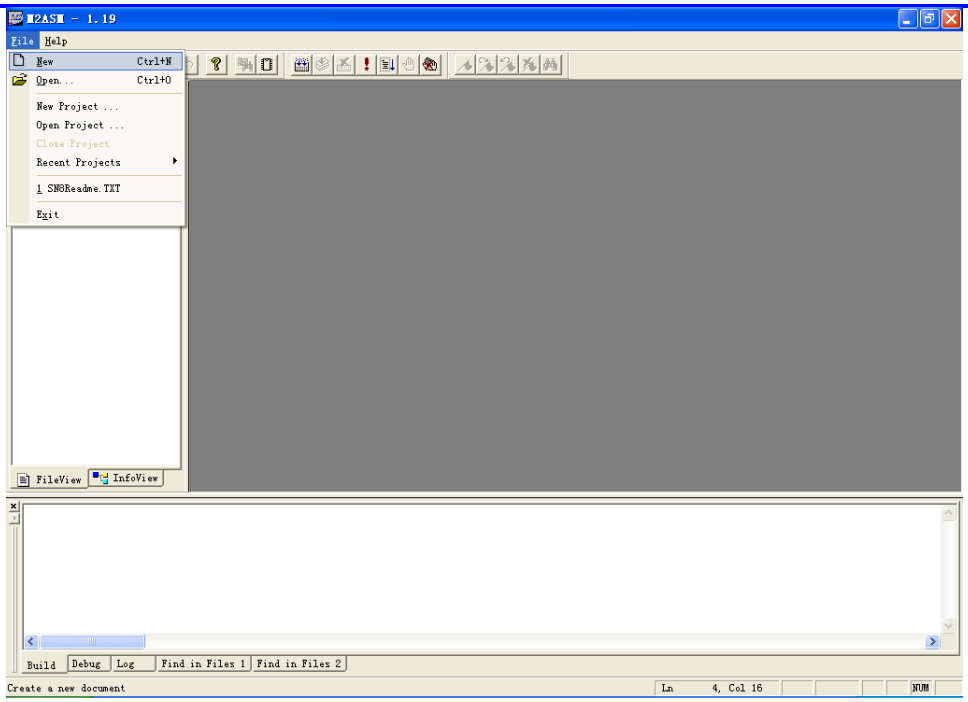

Figure 2-42 Drop-down menu of File in idle mode

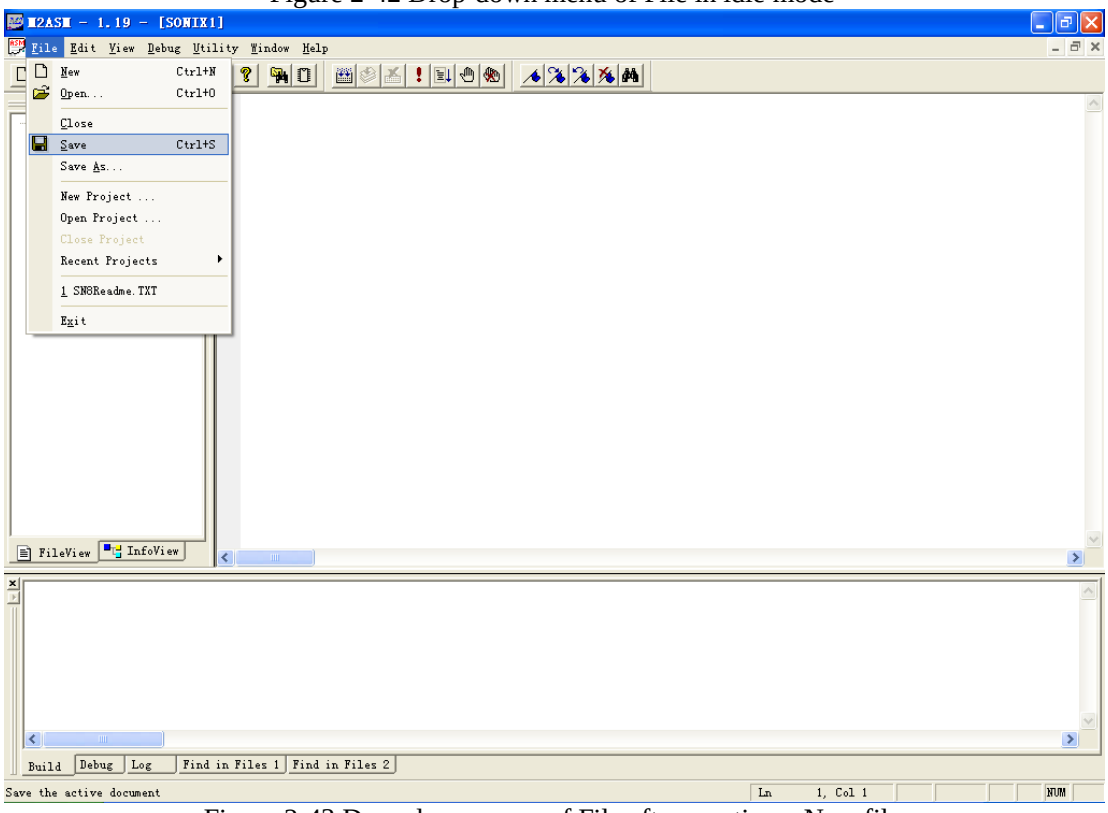

Figure 2-43 Drop-down menu of File after creating a New file

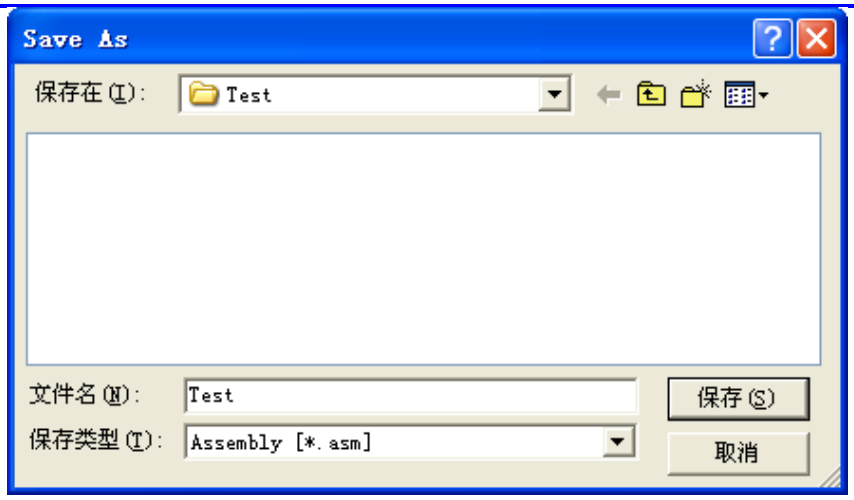

Figure 2-44 Dialog box to save file

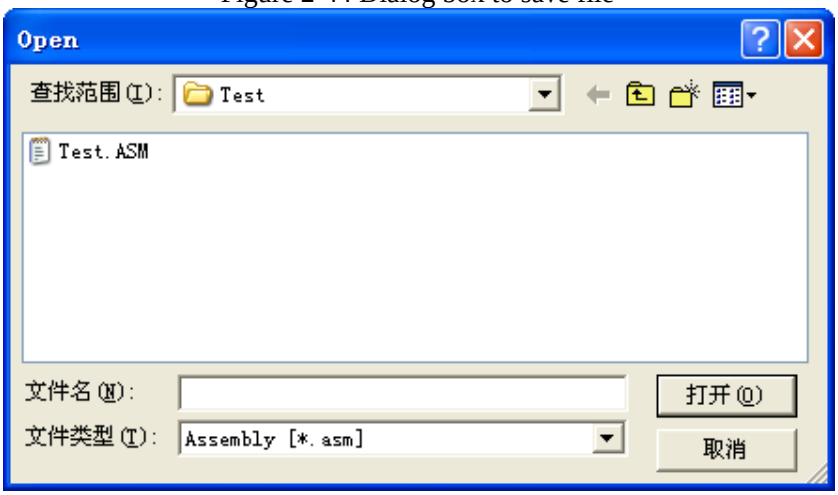

Figure 2-45 Dialog box to create a New Project

So far we have successfully creatted the project Test.PRJ, and included the file Test.ASM in the project, but there is no source file in this file. Next, please input the source code into this file. The text input method of M2Asm is almost same with that of other text editors, it can input, delete, select, copy, and paste, and execute other basic text processing commands, and the difference is that the different contents in edit box of assembly file are displayed in different colors (e.g. the pseudo-instruction is in blue, and the note is in green).

Please note that at the beginning of the program it requires to use “CHIP” pseudo-instruction to define the type of the used chip, so that in the process of compiling the compiler will pop up different configuration dialog box, which will be described in the section of Compiling, Linking and Debugging of Files in detail.

In the actual situation, a project contains multiple program files, the pseudo-instruction “INCLUDE” is used to include these files into the project, in the next compiling these included files will be automatically added into the project, please note that these included files shall be saved in the same directory of the project as far as possible, otherwise when you are using “INCLUDE”, you are required to specify the path of the included file.

If the project is required to add the header file, you can click right on the folder “Header Files” in the viewing window of file, and the menu of Figure 2-46 will pop up, after the selection of “Add Files to Folder”, the file selection dialog box will be shown up, as shown in Figure 2-47, select the header file to be added, click “Open” to add the header file.

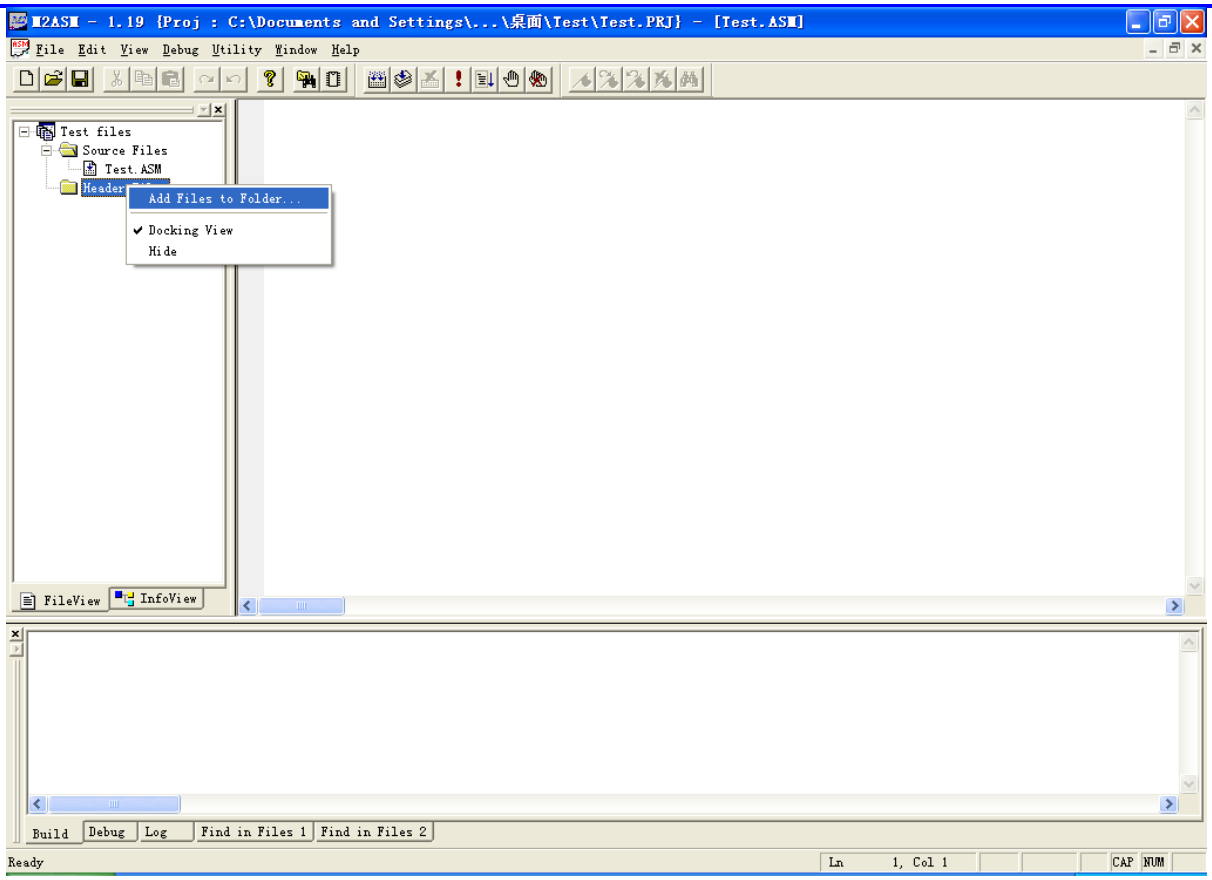

Figure 2-46 Header File Adding Menu

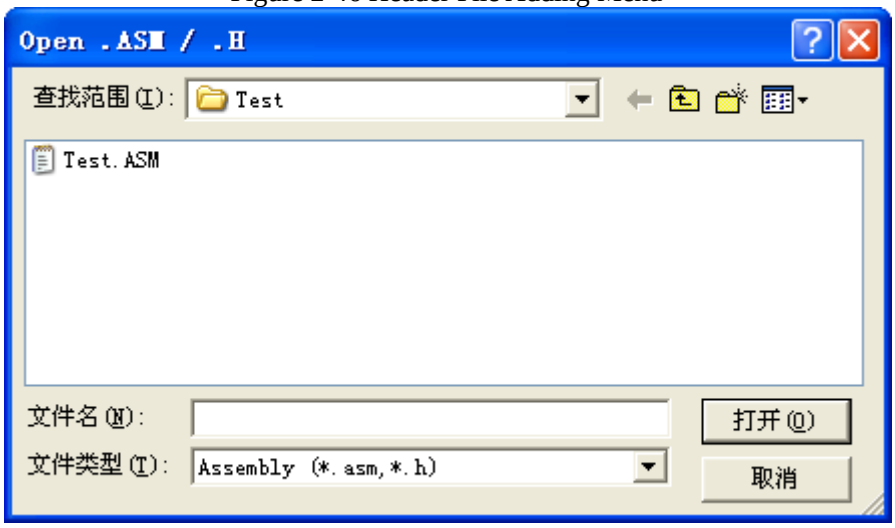

Figure 2-47 Header File Selection Dialog Box

The following is to describe how to use “INCLUDE” to specify the path of the included file.

When “INCLUDE” is used to include a file, if the file is saved in the same directory of the project, its format is as follows:

INCLUDE File Name

Examples:

INCLUDE led_premain.ASM
INCLUDE led_App.ASM
INCLUDE SN88X.H // Include the customized variable’s name

If the file is not saved in the same directory of the project, its format is as follows:

INCLUDE File path and File name

Examples:

INCLUDE USB_ISP\SN8_USB_ISP_EnC.lib
INCLUDE USB\SN8_USB_macro.h
INCLUDE sub \ Filename.ASM
INCLUDE C: \PROJECT.H // Include the customized macro

```
INCLUDE         ..\Parent\File2.ASM
```

Of which USB_ISP, USB and sub are the folder under the directory of the project, SN8_USB_ISP_EnC.lib, SN8_USB_macro.h and Filename.ASM file are included.

2.3.2 Compiling and Linking of Program

The project is created and code is added in the above section, now we can start to compile and link the program. The use of M2ASM can easily compile the program. Generally it is divided into the following steps.

Use the button of "Build" in toolbar or click the "Debug" to select "Build", and you can directly use the shortcut key F7 to compile and link the project, at that time the compiling configuration dialog box will pop up, as shown in the following figure.

Figure 2-48 Compiling Environment Setting Dialog Box

M2ASM is used the pseudo-instruction "CHIP" in the program to define the type of chip, and used the different pop-up configuration dialog boxes to configure the related attributes of chip. In Figure 2-48, it is the configuration dialog box for SN8P2708A, but the options are in gray lock state, which can be activated by clicking "Update Code Option", as shown in Figure 2-49, then we can modify the configuration options, please refer to **2.3.4 Compiling Option** (Code Option for the function introduction of each option.

Figure 2-49 Setting Dialog Box in Active State

After the configuration is finished, click "OK" to complete the compiling configuration, if the program is correct, the configuration information (as shown in Figure 2-50) will be displayed behind the defining statement of chip type in the code.

```
CHIP            SN8P2708a
//{{SONIX_CODE_OPTION
        .Code_Option    Noise_Filter    Enable
        .Code_Option    Watch_Dog       Always_ON       ; Watchdog still enable even in Green and Sleep mode
        .Code_Option    High_Clk        4M_X'tal        ; Crystal/Resonator: 2Mhz~10Mhz
        .Code_Option    Fcpu            #2      ; Fcpu = Fosc/4
        .Code_Option    Security        Enable

//}}SONIX_CODE_OPTION
```

Figure 2-50 Compiling Configuration Information

In the actual situations, it's hard to complete the compiling at one time. There are often some errors in the code, resulting in the failure of compiling. For example, after the execution of compiling instruction, M2ASM will pop up the error message prompt box as shown in Figure 2-51, in which it shows the wrong line, the number of warnings and errors.

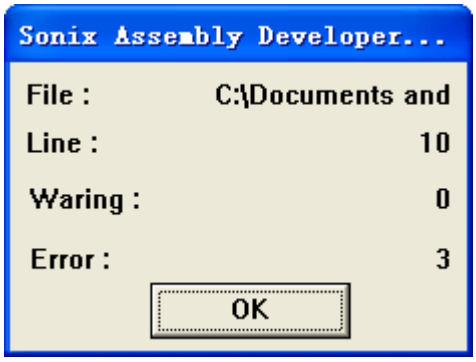

Figure 2-51 Compiling Error Information Prompt Box

Click "OK" to eliminate the error message box, then the interface is shown in Figure 2-52, and it can be found that a blue arrow is marked in front of the wrong statement. In the compiling information window, the error information of compiling are listed, including the type and location of error, the information highlighted in blue is the error information of the error indicated in the current editing box. The user can double-click the error information to display the current wrong line, for example, we double-click the first error information, and then the first error information will be highlighted in blue, and the blue arrow in the editing box will point to the first error statement.

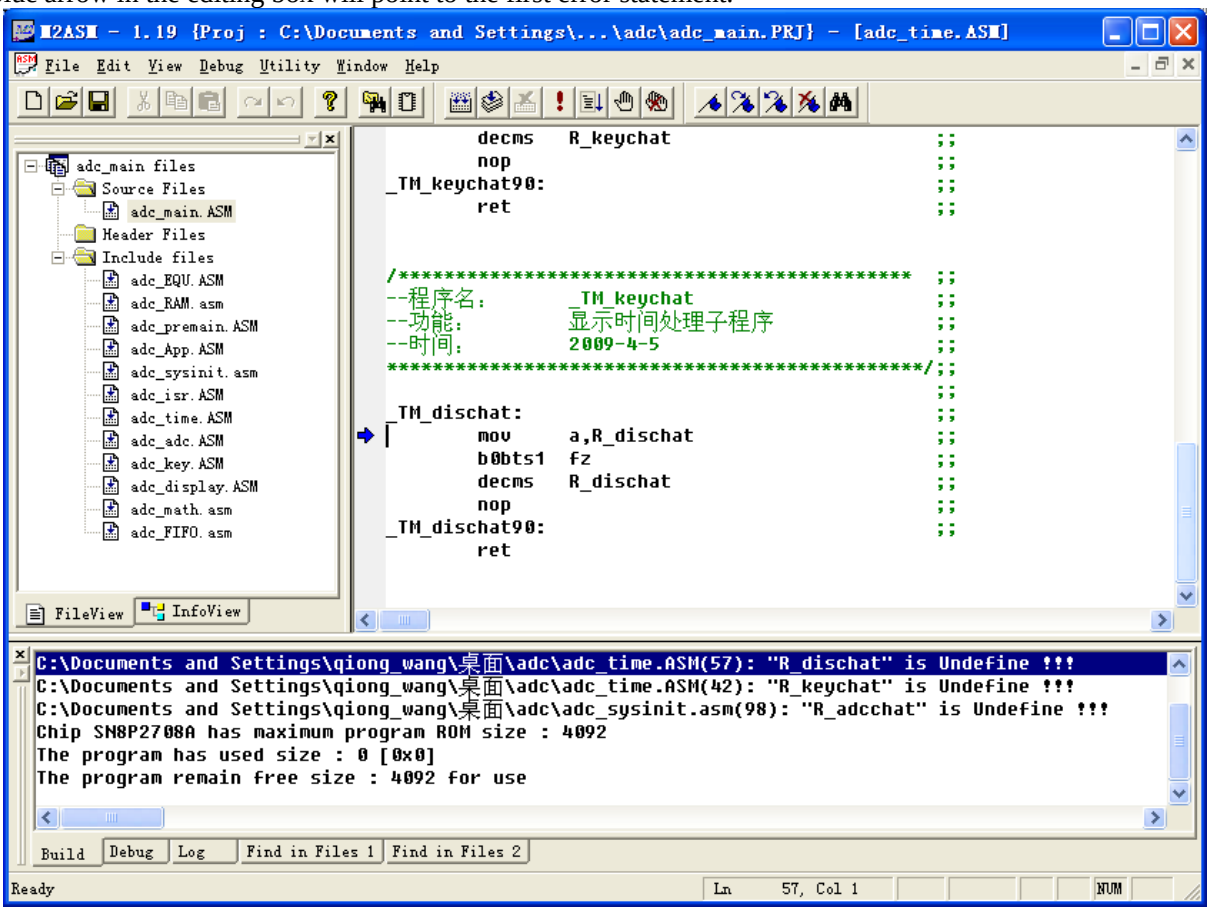

Figure 2-52 Compiling Error Interface

After the modification of error, it shall re-compile, if it is successfully compiled, the content of compiling information will be shown as Figure 2-53, in which it shows the checksum, the maximum ROMsize,the used size,the remain free size.

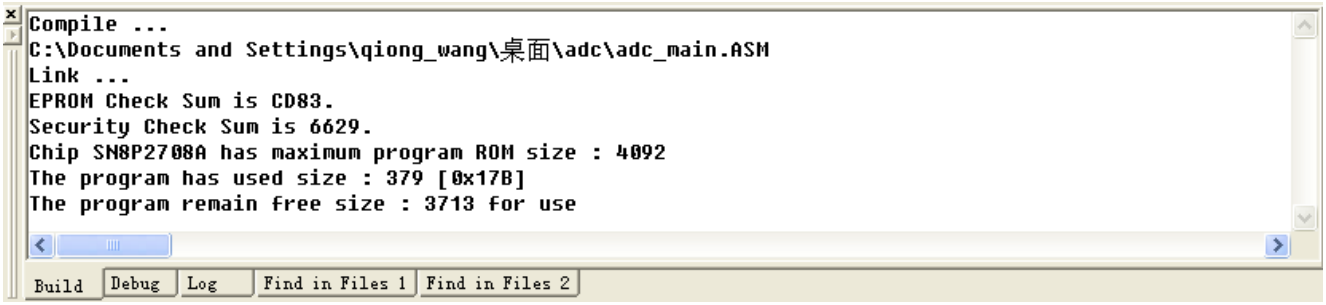

Figure 2-53 Compiling Information Box after Successful Compilation

2.3.3 Running and Debugging of Program

In order to help user to more easily analyze and debug programs, M2IDE provides many debugging commands and viewing window. In the process of debugging, if a variety of windows are well used to watch the running state of program, the change of variables and other situations, then twice as much can be accomplished with half effort. The commonly used debugging commands will be introduced at first in the following, and an example will be used to explain the process of debugging. It shall be noted that the debugging commands and debugging window are only effective in debugging mode.

Debugging Commands

There are three methods to execute the debugging commands provided by M2IDE.

Method 1: In debugging mode, use the debugging tools menu (Debug) in M2ASM menu bar, after the debugging commands menu is popped up, select the corresponding debugging command to execute the command;

Method 2: In debugging mode, click the icon of shortcut command in M2ASM toolbar, as shown in Figure 2-54, the corresponding command also can be executed.

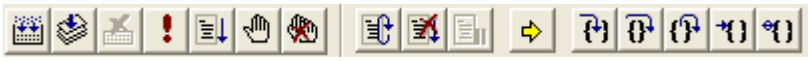

Figure 2-54 Shortcut Command Icons

Method 3: Use the shortcut keys, and the shortcut keys are shown in Table 2-2. To skillfully use shortcut keys can greatly improve the speed of debugging.

Table 2-2 Shortcut Keys of Debugging Commands

Shortcut Key	Command
F5	Start/stop operation
Ctrl+ F5	Reset
Shift+ F5	Exit debugging state
F7	Compiling program
F8	Download program to emulator
F9	Set breakpoints
Ctrl+ Shift+F9	Remove all breakpoints
F10	Skip over function
Ctrl+F10	Run to cursor
F11	Single-step operation
Shift+F11	Jump out of function
F12	Program counter is pointing to the cursor

In the M2IDE, the system provides a variety of ways to run the program for the user, and the user can use the corresponding command to achieve the full speed, single-step and other methods to run the program.

Commonly used operation methods are as follows:

Running at full speed/ Pause running (F5)

In the stop or pause mode, to execute this command will run the application program at full speed, in this way, you can view the achievement of the program's functionality. In practice, this command is used together with breakpoints, if it has already set breakpoint in key part of the program, after this command is executed, the program will execute to the breakpoint, and the pointer will point to the program line, and it will wait for executing other commands. During operation, to execute this command will stop the current command line.

Single-step operation (F11)

This command is to execute the current command statement pointed by the cursor, after execution, PC will point to the next instruction. If the subprogram is called during execution (or function), then it may jump into the subprogram. This command is to facilitate to exactly view the execution of each instruction, and combining with the observation window, the user can also view the influence of program execution on the register.

Skip over function (F10)

This command is also to execute the current command line, and the difference to single-step operation is that, if the statement to be executed is the function calling statement, this command will complete the execution of the function at one time without entering into the function. If the statement to be executed is the general assembly statement, the function is same with that of single-step operation.

Jump out of a function (Shift + F11)

This command is used to jump out of the current subprogram, if you want to quickly execute the current subprogram and return to the position where the function is called, you can use this command. Please note that if it is not in any subprogram at present, please try not to use this command, otherwise it may cause the failure of emulation.

Run to cursor (Ctrl + F10)

This command is executed to make the program to execute to the cursor position in the code window. It is equivalent to use the position of cursor as a temporary breakpoint.

Program counter is pointing to the cursor (F12)

This command is executed to make the program counter to point to the current line of cursor, please note that only the program counter PC is modified here, no commands are executed.

Reset (Ctrl + F5)

This command is executed to set 0 for the program counter, please note that this command cannot make external devices and system registers to be in the reset state, so the reset command is not equivalent to the hardware reset of CPU.

Download program (F8)

SN8ICE 2K supports to download programs, after the program is downloaded, it can freely run out of the computer environment. After this command is used, it will pop up a file selection dialog box, as shown in Figure 2-55; select . SN8 file to be downloaded, click Open, and a successful download prompt box will be popped up, as shown in Figure 2-56, to prompt the user that the file is successfully downloaded and MCU is freely running.

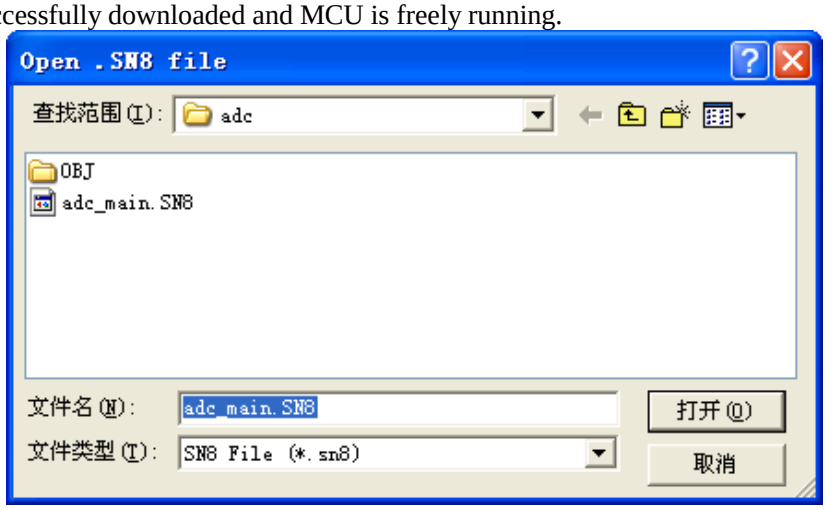

Figure 2-55 Download file selection dialog box

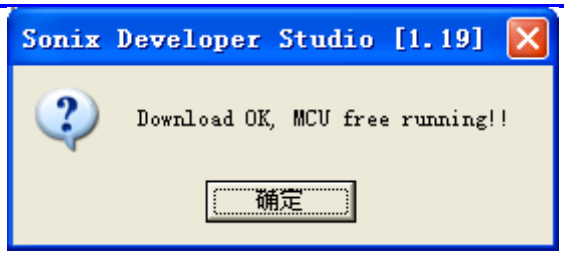

Figure 2-56 Successful download prompt box

The process of M2IDE to debug program is described in the following.

Under the condition of successful compiling, execute Go (Run) command to enter the debugging environment as shown in Figure 2-57, you can see some buttons of debugging commands which are not displayed before in the toolbar, and each observation window is also displayed. At that time the pointer of the program is pointing to the first jumping instruction "jmp reset", and the program will start to execute from here. There are two points to be noted:

M2IDE does not provide the function of software emulation, so it must connect the emulator with PC before entering the debugging environment, otherwise it will not enter into the emulation environment, and a prompt box will be shown up, as shown in Figure 2-58;

If the Go (Run) command is used before compiling, M2ASM will compile at first, and then directly enter the debugging state.

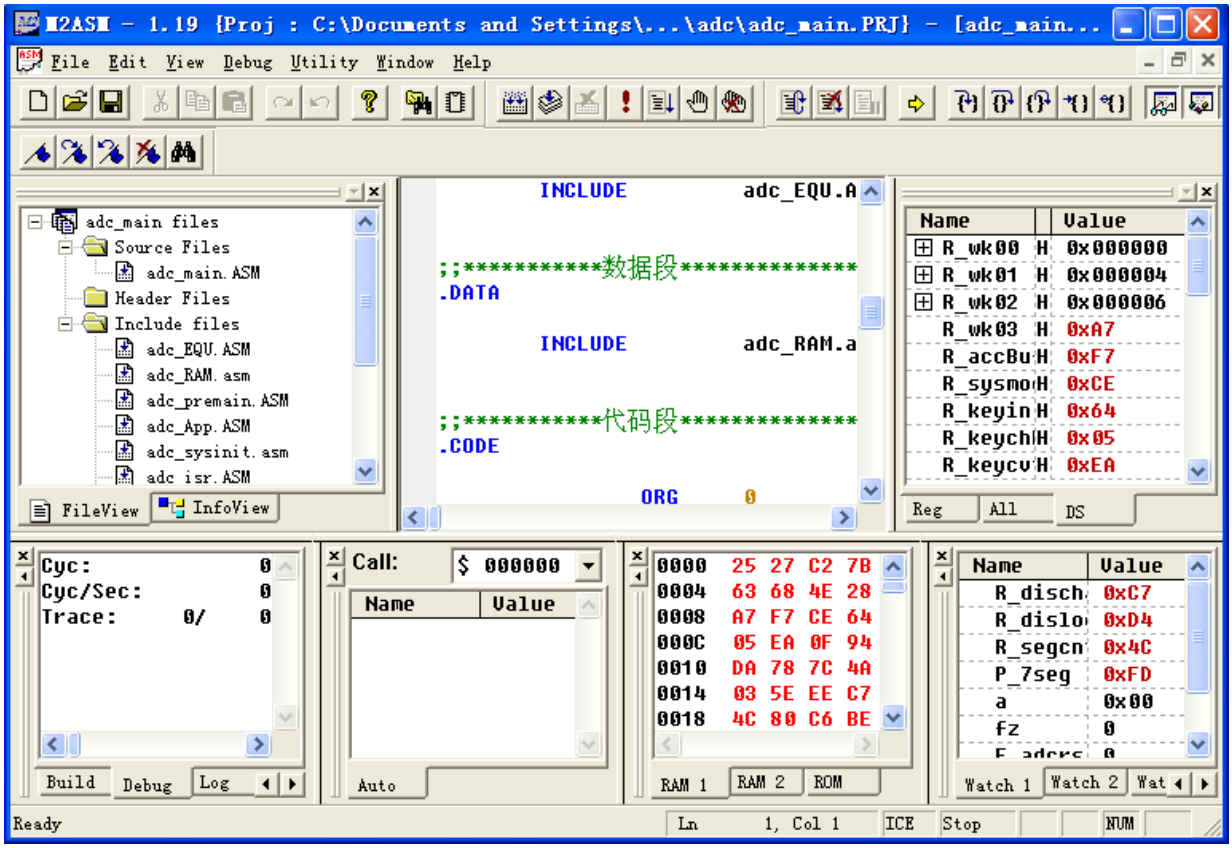

Figure 2-57 Debugging Interface

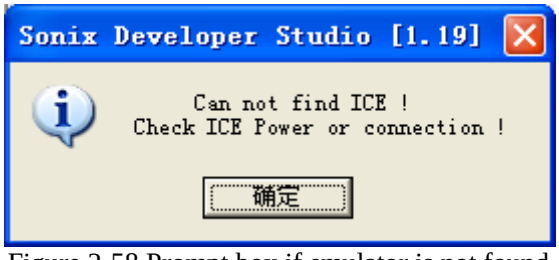

Figure 2-58 Prompt box if emulator is not found

To continuously execute Step Into (single-step) command, you can see the continuous changes of variables in the variable observation window, and the respective values are displayed. When the register or user-defined variable is changed, the register or the variable value in the register window will be displayed in red. The array in register window is to only display the address unit occupied by the array before the running of program, for the specific information of the array's member, you shall click sign "+" in the front of the array to expand the array, as shown in Figure 2-59, you can see that there

are 4 members in array rled_buf and their values are 0x17.

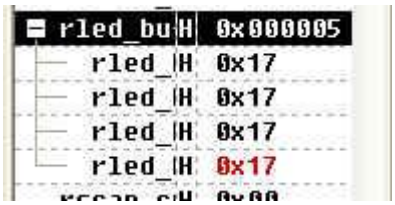

Figure 2-59 Array expanded in observation window

Move the cursor to the beginning of the subprogram mnled, click the icon, at that time a red solid dot will shown up in the front of the first instruction of mnled, indicating that a breakpoint is set at this instruction.

To execute Go (Run) command, the program will directly run to the breakpoint, as shown in Figure 2-60. In the observation window you can find that the values changed in the recent program segment are displayed in red, to input the names of two variables into the observation window, namely: "rled_buf" and "r500ms_cnt", you can be directly observe their current values , as shown in Figure 2-61.

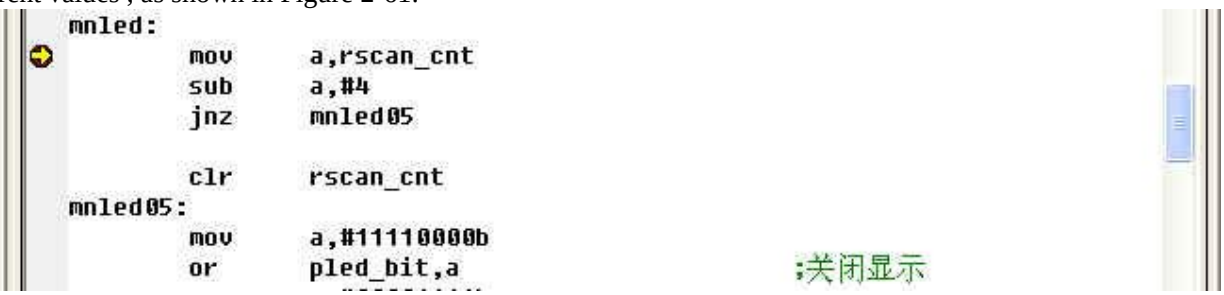

Figure 2-60 Program running to breakpoint

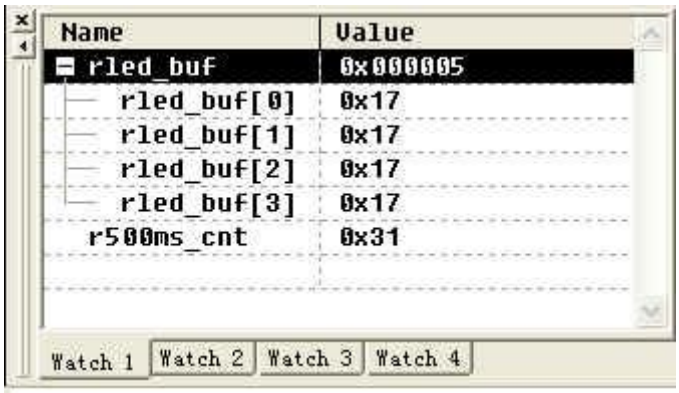

Figure 2-61 Directly observe the values of variables in the variable watch window

To execute Step Out (Jump out of a function) command, at that time you can find that the program is directly complete the execution of subprogram and stopped at next statement of call instruction, as shown in Figure 2-62.

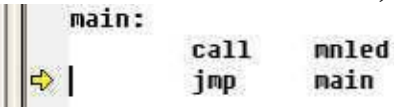

Figure 2-62 Jump out of a function

Click the icon to cancel the previously set breakpoints, and set breakpoint at the statement to call and display subprogram, and run to such instruction, as shown in Figure 2-63. To execute Step Over (Skip over function) command, you can find that the program does not enter mnled subprogram, but directly completes the execution of this subprogram, and stops at the next statement of call instruction, as shown in Figure 2-64.

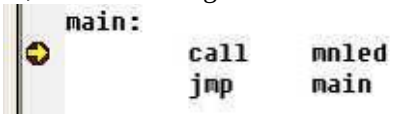

Figure 2-63 Run to the statement to call and display function

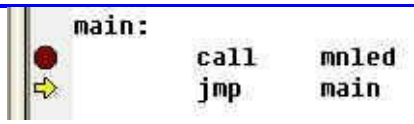

Figure 2-64 Skip over function

To execute Remove All Breakpoints (clear all breakpoints) command, you can see all breakpoints are removed.

To execute Go (Run) command again to make the program to run at full speed, you can see the interface as shown in Figure 2-65, to repeat Go (Run) command to pause the operation of program.

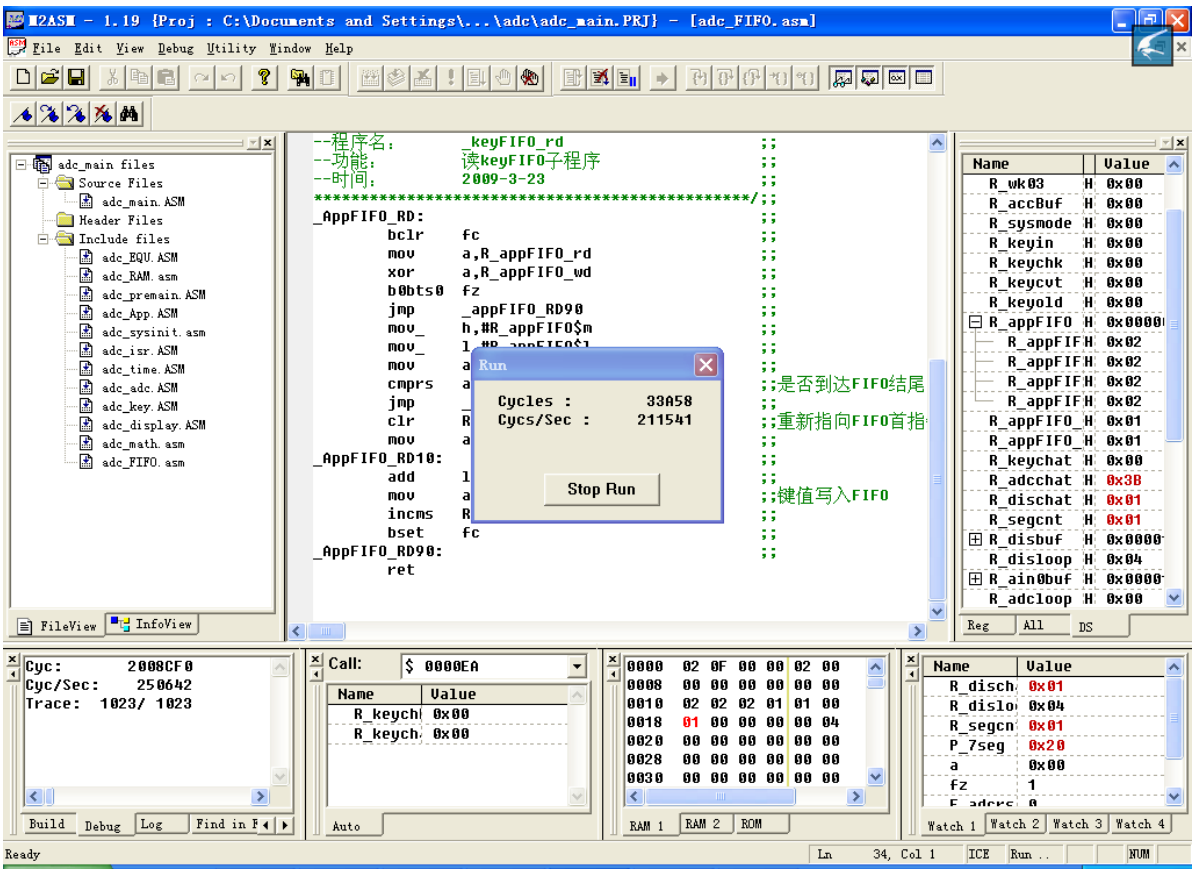

Figure 2-65 Interface of running at full speed

Stop running program, use the Download command to download the program to the emulator, then close M2IDE, and the program will be freely run in emulator.

In the above section, we have introduced the commonly used methods to debug program in M2IDE in details, the user can flexibly apply a variety of tools and debugging commands, it's the only way to faster and better write high-quality programs.

2.3.4 Compiling Option (Code Option)

In Section 2.3.2 Compiling and Linking of Program, it has been talked about the Code Option, as shown in Figure 2-66.

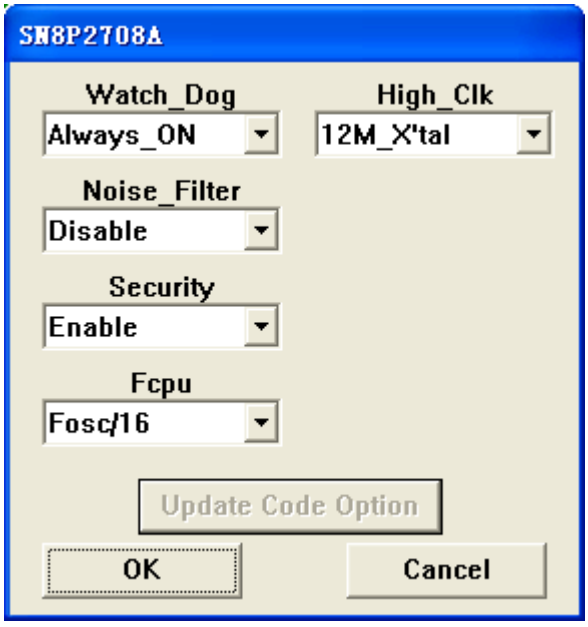

Figure 2-66 Code Options

Different chip declarations and different versions of the compiling software, Code Option has different options to choose. The meanings of some options are as follows:

◆ Watchdog:
Always on-Watchdog Timer is always turned on;
Enable-Watchdog timer is enabled in normal and slow mode, and stopped in green and sleep mode;
Disable-Watchdog timer is turned off.
※ Note: When you select the option of “Always on”, the system will not enter sleep mode.

◆ Reset_Pin:
Pxx-Select the internal reset, meanwhile this pin is used as unidirectional input port Pxx;
Reset-select the external reset;
※ Note: When you select the internal reset, Pxx port is the unidirectional input port, and there is no internal pull-up resistor.

◆ High_Clk:
IHRC_16M- The chip oscillating source is adopted the internal 16M high-speed RC oscillation circuit;
Ext_RC- The chip oscillating source is adopted the external RC oscillator circuit;
32K_X'tal- The chip oscillating source is adopted the external low frequency crystal oscillator (e.g. 32.768KHz);
4M_X'tal- The chip oscillating source is adopted the external standard quartz oscillator or ceramic oscillator (Generally, 2M ~ 10MHz);
12M_X'tal- The chip oscillating source is adopted the external high-speed quartz oscillator or ceramic oscillator (Generally, 10MHz ~ 16MHz).
※ Note: IHRC_16M option is only available in such IC that internally integrated the high-speed RC oscillation circuit, when this option is selected, the two pins (XIN/XOUT) will be used as general I/O.

◆ Fcpu:
Fosc/1- instruction cycle = 1 clock cycle;
Fosc/2- instruction cycle = 2 clock cycles;
Fosc/4- instruction cycle = 4 clock cycles;
Fosc/8- instruction cycle = 8 clock cycles;
Fosc/16- instruction cycle = 16 clock cycles;
※ Note: When Noise_Filter Enable or IHRC_16M is selected in Code Option, Fosc/1 and Fosc/2 in Fcpu option will be automatically blocked.

◆ Security:
enable-The program code is encrypted;
disable- The program code is not encrypted.

◆ Noise_Filter:
enable-Enable the function of noise filtering.
disable- Disable the function of noise filtering.
※ Note: When the noise filtering function is turned on, it will enhance the anti-jamming capability of chip, meanwhile Fosc/1 and Fosc/2 in Fcpu option will be automatically blocked.

◆ LVD:
LVD_L- When VDD is below 2.0V, LVD will reset the system;
LVD_M- When VDD is below 2.0V, LVD will reset the system and the 24-bit PFLAG register of LVD will be used as 2.4V low voltage detector;
LVD_H- When VDD is below 2.4V, LVD will reset the system and the 36-bit PFLAG register if LVD will be used as 3.6V low voltage detector;
LVD_MAX- When VDD is less than 3.6V, LVD will reset the system. (Individual models have such function, such as SN8P2522)

◆ Rst_Length
No-No external reset debounce time;
128 * ILRC- The external reset debounce time is 128 * ILRC.

◆ Ext_OSC
6MHz- The chip oscillating source is selected 6MHz crystal oscillator/ceramic oscillator;
12MHz- The chip oscillating source is selected 12MHzcrystal oscillator/ceramic oscillator;
16MHz- The chip oscillating source is selected 16MHzcrystal oscillator/ceramic oscillator.

◆ Fslow
Flosc/2- In slow mode the clock is Flosc/2;
Flosc/4- In slow mode the clock is Flosc/4.

◆ Ext_Reset_Length
No- No external reset debounce delay time;
128 * ILRC- The external reset debounce delay time is 128 * ILRC.

◆ IHRC_Detect
Enable- Enable the function of IHRC detection;
Disable- Disable the function of IHRC detection.

2.3.5 Types of Project File

.ASM	Assembly language program file
.HEX	The hex files for programming
.LST	Lists file
.PRJ	Project,. PRJ is the suffix of the name
.SN8	Programming files after compiled by system
.LIB	Library files
.BBB	Files generated after the online mode Read OTP is used
.INI	Files to record RollingCode

2.4 How to Emulate LCD

LCD Simulator Function is used the function of software interface simulation LCD display to facilitate the developers to debug program and simplify the hardware circuit in MCU (embedded LCD driver) emulation. No LCD screen is needed, and it only needs to edit the corresponding BMP in accordance with the open model mode of LCD screen, to load BMP graphics to edit and generate . LCD files in software simulation will be OK.

LCD simulation panel is shown in Figure 2-67.

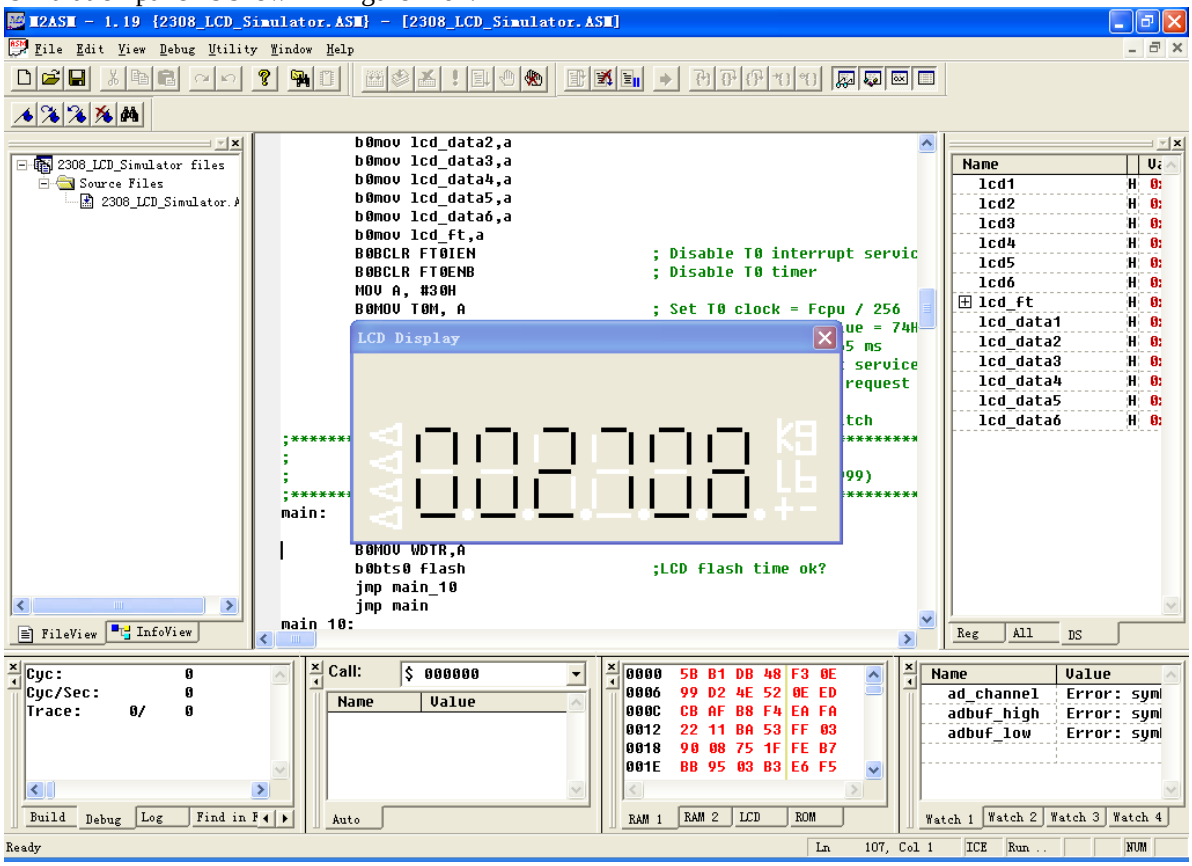

Figure 2-67 LCD software simulation panel

Please refer to LCD_Simulator_Manual_V01_SC.pdf document under the installation directory of compiler for the specific methods of LCD simulation;

Path: C: \ Sonix \ M2IDE_Vxxx \ LCD_Simulator_Manual_V01_EN.pdf

Chapter 3 Development Language

3.1 Instruction Set

Field	Formate	Description	C	DC	Z	Cycle
	MOV A,M	A ← M	-	-	√	1
M	MOV M,A	M ← A	-	-	-	1
O	B0MOV A,M	A ← M (bnak 0)	-	-	√	1
V	B0MOV M,A	M (bank 0) ← A	-	-	-	1
E	MOV A,I	A ← I	-	-	-	1
	B0MOV M,I	M← I (M is only suitable for system register R, Y, Z , RBANK, PFLAG)	-	-	-	1
	XCH A,M	A ← →M	-	-	-	1+N
	B0XCH A,M	A← →M (bank 0)	-	-	-	1+N
	MOVC	R, A← ROM [Y,Z]	-	-	-	2
	ADC A,M	A ← A + M + C，if carry generated, then C=1，Or C=0	√	√	√	1
A	ADC M,A	M ← A + M + C，if carry generated, then C=1，Or C=0	√	√	√	1+N
R	ADD A,M	A ← A + M，if carry generated, then C=1，Or C=0	√	√	√	1
I	ADD M,A	M ← A + M ，if carry generated, then C=1，Or C=0	√	√	√	1+N
T	B0ADD M,A	M (bank 0) ← A + M (bank 0)，if carry generated, then C=1，Or C=0	√	√	√	1+N
H	ADD A,I	A← A + I，if carry generated, then C=1，Or C=0	√	√	√	1
M	SBC A,M	A ← A - M - /C，if borrow generated, then C=0，Or C=1	√	√	√	1
E	SBC M,A	M ← A - M -/C，if borrow generated, then C=0，Or C=1	√	√	√	1+N
T	SUB A,M	A ← A – M，if borrow generated, then C=0，Or C=1	√	√	√	1
I	SUB M,A	M ← A – M ，if borrow generated, then C=0，Or C=1	√	√	√	1+N
C	SUB A,I	A← A – I，if borrow generated, then C=0，Or C=1	√	√	√	1
	DAA	Change the ACC data from the hex to decimal format	√	-	-	1
	MUL A,M	R, A ← A * M，the low byte of product is stored in ACC, and the high byte is stored in the system register R, and ZF flag bit is subject to the content of ACC	-	-	√	2
	AND A,M	A ← A and M	-	-	√	1
L	AND M,A	M ← A and M	-	-	√	1+N
O	AND A,I	A ← A and I	-	-	√	1
G	OR A,M	A ← A or M	-	-	√	1
I	OR M,A	M← A or M	-	-	√	1+N
C	OR A,I	A← A or I	-	-	√	1
	XOR A,M	A ← A xor M	-	-	√	1
	XOR M,A	M ← A xor M	-	-	√	1+N
	XOR A,I	A← A xor I	-	-	√	1
	SWAP M	A (b3~b0, b7~b4) ← → M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM M	M(b3~b0, b7~b4) ← → M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC M	A← M right shift with carry	√	-	-	1
O	RRCM M	M← M right shift with carry	√	-	-	1+N
C	RLC M	A ←M left shift with carry	√	-	-	1
E	RLCM M	M← M left shift with carry	√	-	-	1+N
S	CLR M	M← 0	-	-	-	1
S	BCLR M.b	M.b ← 0	-	-	-	1+N
	BSET M.b	M.b← 1	-	-	-	1+N
	B0BCLR M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET M.b	M(bank 0).b ← 1	-	-	-	1+N
B	CMPRS A,I	Compare, if equation, then skip over the next instruction, C and ZF flag bit may be affected	√	-	√	1+S
	CMPRS A,M	Compare, if equation, then skip over the next instruction, C and ZF flag bit may be affected	√	-	√	1+S
R	INCS M	A ← M + 1 ，if A = 0，then skip over the next instruction	-	-	-	1+S
A	INCMS M	M ← M + 1 ，if M = 0，then skip over the next instruction	-	-	-	1+N+S
N	DECS M	A ← M – 1，if A = 0，then skip over the next instruction	-	-	-	1+S
C	DECMS M	M ← M – 1，if M = 0，then skip over the next instruction	-	-	-	1+N+S
H	BTS0 M.b	if M.b = 0，then skip over the next instruction	-	-	-	1+S
	BTS1 M.b	if M.b = 1，then skip over the next instruction	-	-	-	1+S
	B0BTS0 M.b	if M(bank 0).b = 0 ，then skip over the next instruction	-	-	-	1+S
	B0BTS1 M.b	if M(bank 0).b = 1 ，then skip over the next instruction	-	-	-	1+S
	JMP d	Jumpping instruction，PC15/14 ← RomPages1/0，PC13~PC0← d	-	-	-	2
	CALL d	Call instruction for subprogram，Stack ← PC15~PC0，PC15/14← RomPages1/0，PC13~PC0 ← d	-	-	-	2
M	RET	Jumpping instruction of subprogram，PC← Stack	-	-	-	2
I	RETI	Jumpping instruction of interrupt treatment program，PC ← Stack ，enable the global interrupt control bit	-	-	-	2
S	RETLW	Jumpping instruction of subprogram， PC← Stack ，to save the obtained value in ACC	-	-	-	2
C	NOP	Dummy instruction, no special meanings	-	-	-	1

1. Explanation of symbols in instruction system

It can be seen from the above chapters and the instruction table some symbols are used in the operand and the destination operand. The meanings of these symbols are summarized as follows:

- A ：Accumulator ACC
- M ：A address unit in data memery RAM
- I ：Constant, it must be begun with symbol "#"，E.g：#0x1A, #V1
- . b ：Bit address, e.g.：the fifth bit of M unit of M.5 table data memery
- d ：Address of program memery ROM
- (X：Y)：Byte X and Y consists of a word (word)，of which Y is the lower byte
- (X：Y：Z)：a 24 bits data composed by three 8 bits arrays: X、Y and Z array

In the execution cycle
N: The system register is 1, and the user-defined register is 0;
S: If the condition is true, it is 1; if not, it is 0;

2. Influence of instruction on flag bit

The influence of SONiXMCU instruction on flag bit is divided into two categories: one is to affect the state of some flat bits of program state register PFLAG after execution, namely no matter the state of flag bit before execution of instruction, it will form new flag state according to the definition of flag bit during the execution of instruction; another one is not to affect the state of flag bit after execution, the state will be kept as the original state after the execution of instruction.

The influence on flag bit is varied from instructions, and the influence of each instruction on flag bit is shown in the Instruction Table.

Note: The instruction cycles are different in MCU of SONiX SN8P1000 Series and SN8P2000 series, it may be different with the above table. However, the actual instructions of some MCU may be only a part of the above but not all, so the MCU specification shall be prevail, this table is only to list and introduce the usage of each instruction.

3.2 Pseudo-instruction

Pseudo-instruction is the instruction to tell the assembler how to compile, it cannot control the operation of machine or compiled into machine code, and it only can be recognized by the assembler and it is to guide the implementation of compiling. The pseudo-instruction statement is relative to the instruction statement.

Instruction statement

Each instruction statement will generate the instruction code (i.e., object code) for MCU to execute when the source code is compiling, so such statement is also called the executable statement. Each instruction statement indicates a basic capability of MCU, for example, data transfer, add or subtract between two numbers, shift, etc., and this function will be fulfilled during the running of target program (an ordered collection of instruction code), which is relied one CPU, memory, I/O interface and other hardware to fulfill.

Pseudo-instruction statement

Pseudo-instruction statement is to guide the assembler how to compile program, so such statement is also called command statement. For example, the pseudo-instruction statement in the source code is to tell the assembler how to segment the program, which logic segments are the current segments in the program segment, and which segment register is pointed to them, which data is defined, how to distribute the storage unit, and so on. For pseudo-instruction statement, except the defined specific data will generate object code, the rest have no corresponding object code. Such commands functions of pseudo-instruction statement are fulfilled by executing a segment of program when the assembler is compiling, but not fulfilled when the target program is running.

Please refer to Appendix Ⅲ for pseudo-instruction table.

3.3 Include File

INCLUDE:

Syntax: INCLUDE FILE

Description:

A program file is required after this instruction, if this program file is not in the directory of currently being compiled program, the user shall specify the path for the file. The syntax of such program file is similar with that of common files. Generally, the file with the extension name .H always includes the constants and macros, and the file with the extension name .asm includes the instructions. The included files also can use the INCLUDE instruction, and the nesting level can be 255 levels at most.

Example:

```
INCLUDE     SN88X.H              // Include the name of customized variable
INCLUDE     C: \PROJECT.H        // Include the customized macros
INCLUDE     sub \ Filename.ASM
INCLUDE     ..\Parent\File2.ASM
```

INCLUDESTD:

Syntax: INCLUDESTD FILE

Description:

A program file is required after this command, and the path is fixed to the path of SN8ASM.EXE, the others can refer to INCLUDE instruction. This command allows the user to easily access the system-provided macro files.

Example:

```
INCLUDESTD    MACRO1.H    // Include the commonly used macros of system
INCLUDESTD    MACRO2.H    // Include the commonly used macro files of system
```

INCLUDEBIN:

Syntax: INCLUDEBIN FILE

Description:

A BIN file is required after this command, if this program file is not in the directory of currently being compiled program, the user shall specify the path for the file. The data unit of this file is WORD. If it is less than one WORD, it shall be counted as one WORD.

Example:

```
INCLUDE     SPEECH.BIN
INCLUDE     C: \SOUND.SND
```

3.4 Macro

SONiX supports the following default macros, with which it's more easily to write programs, the following macros are defined in three files in the compiler directory: MACRO1.H, MACRO2.H and MACRO3.H.

Category	Assembling mnemonic code	Extended format	Qty of instruction after extended	Function	Flag CF	DC	ZF	Cycle
COMMAND	CLC	B0BCLR FC	1	Clear C flag	0	–	–	1
	STC	B0BSET FC	1	Set C flag	1	–	–	1
	RSTWDT	B0BSET FWDRST	1	Reset the watchdog counter	–	–	–	1
	EINT	B0BSET fgie	1	Enable the global interrupt bit	–	–	–	1
	DINT	B0BCLR fgie	1	Disable the global interrupt bit	–	–	–	1
	NOT A	XOR A, #0FFh	1	A ← ~ A	–	–	–	1
	NEG A	XOR A, #0FFh ADD A, #1	2	A ← – A	–	–	–	2
ROTATE/SHIFT	SHL memory	B0BCLR FC RLCM memory	2	memory ←memory * 2	–	–	–	2
	SHR memory	B0BCLR FC RRCM memory	2	memory ←memory / 2	–	–	–	2
	B2B bit1, bit2	B0BCLR bit2 B0BTS0 bit1 B0BSET bit2	3	Exhange the state between bit2 and bit1	–	–	–	3
	ROL mem	RLCM mem B2B FC, mem.0	4	FC ← mem.7 mem←mem * 2 + FC	–	–	–	4
	ROR mem	RRCM mem B2B FC, mem.7	4	FC ← mem.0 Mem←mem / 2 +FC * 80h	–	–	–	4
	RCR mem	RRCM mem	1	another way to write	–	–	–	1
	RCL mem	RLCM mem	1	another way to write	–	–	–	1
BRANCH	JZ address	B0BTS0 FZ JMP address	2	If ZF == 1，then jump to address	–	–	–	2
	JNZ address	B0BTS1 FZ JMP address	2	If ZF == 0，then jump to address	–	–	–	2
	JC address	B0BTS0 FC JMP address	2	If CF == 1，then jump to address	–	–	–	2
	JNC address	B0BTS1 FC JMP address	2	If CF == 0，then jump to address	–	–	–	2
	JDC address	B0BTS0 FDC JMP address	2	If DCF == 1，then jump to address	–	–	–	2
	JNDC address	B0BTS1 FDC JMP address	2	If DCF == 0，then jump to address	–	–	–	2
BRANCH	JB1 bit, addr	BTS0 bit JMP addr	2	If bit == 1，then jump to addr	–	–	–	2
	JB0 bit, addr	BTS1 bit JMP addr	2	If bit == 0，then jump to addr	–	–	–	2
	DJNZ mem, adr	DECMS mem JMP adr	2	mem ← mem –1 If mem != 0，then jump to adr	–	–	–	2-3
	IJNZ mem, adr	INCMS mem JMP adr	2	mem ← mem +1 If mem != 0，then jump to adr	–	–	–	2-3
	CJNE A, m, adr	CMPRS A, m JMP adr	2	If ACC != m，then jump to adr	–	–	–	2-3
	CJE A, m, adr	CMPRS A, m JMP $+2 JMP adr	3	If ACC == m，then jump to adr	–	–	–	3-4
	CJAE A, m, adr	CMPRS A, m B0BTS0 FC JMP adr	3	If ACC >= m (Above Equal)，then jump to adr	–	–	–	3-4
	CJAE m, A, adr	CMPRS A, m B0BTS1 FC JMP adr	3	If m >= ACC (Above Equal)，then jump to adr	–	–	–	3-4
	CJBE A, m, adr	CJAE m, A, adr	3	If ACC <= m (Below Equal)，then jump to adr	–	–	–	3-4
	CJBE m, A, adr	CJAE A, m, adr	3	If m <= ACC (Below Equal)，then jump to adr	–	–	–	3-4
	CJA A,m, adr	CMPRS A, m	4	If ACC > m (Above)，then jump to	–	–	–	4-5

		B0BTS1 FC JMP $+2 JMP adr		adr				
	CJA m,A adr	CMPRS A, m B0BTS0 FC JMP $+2 JMP adr	4	If m > ACC (Above)，then jump to adr	–	–	–	4-5
	CJB A,m, adr	CJA m, A, adr	4	If ACC < m (Below)，then jump to adr	–	–	–	4-5
	CJB m,A adr	CJA A, m, adr	4	If m < ACC (Below)，then jump to adr	–	–	–	4-5

MACRO:

Syntax: NAME MACRO [PARA1[，PARE2，…]]
…
ENDM

Description:

Using Macros can simplify the writing of programs and make the subprogram to shorten the length of program code. If the macros and subprograms are well used, it may make the program to achieve the best state. The quantity of parameters of macro cannot be exceeded 255, but the length is not limited, and it can be composed by A ~ Z, a ~ z, 0 ~ 9 and @ # _. $ and so on, if the special characters are required, the symbols <> shall be used to separate the special characters. In addition, other macros can also be nested in a macro, and the number of levels is not limited.

Example:

```
MOV_    MACRO   ADDRESS,   VALUE
MOV     A, VALUE                    // Macro content
MOV     ADDRESS, A
ENDM

MOV_    1, #1 // Use macro
MOV_    2, <#6*8+1>
```

EXPAND:

Syntax: Same as the above description: when there is error in compiling or it is in simulation, MACRO shall be used to make the cursor to stop at the position of the called macro, to use EXPAND will make the cursor to stop inside the macro instruction, which is conducive to check errors.

REPEAT:

Syntax:
```
REPEAT      COUNT
…           ; REPEATED      COMMAND
ENDM
```

Description:

REPEAT is to repeatedly execute a command. The number of repetitions is decided by COUNT. ENDM is to indicate that this command segment is ended. REPEAT is to indicate the beginning of this command segment.

Example:

```
RRCM_1  MACRO       MEMORY,   VALUE
REPEAT  VALUE
RLCM    MEMORY
ENDM
ENDM
…
RRCM_     0, 4          ; Shit the content of RAM[0] to right for 4 times
```

FOR:

Syntax: FOR PARAMETER， < PARAMETER 1, PARAMETER 2 …>
…
ENDM

Description:

FOR is used to repeatedly execute a segment of command. The number of repetitions is decided by the parameter in the angle brackets, following the parameter sequence separated by commas, one parameter will replace the value of PARAMETER at one time.

Example:

```
EMP  =  0     (temp)
FOR          I, <M0, M1, M2, M3, M4, M5>
I EQU        TEMP
EMP  =       TEMP+1
ENDM
                                    ;  M0/1/2/3/4/5 EQU          0/1/2/3/4/5
```

FORC:

Syntax: FORC PARAMETER, <WORD SERIAL>
… ; Command to be repeated
ENDM

Description:

FORC is used to repeatedly execute a segment of command. The number of repetitions is decided by the length of character string in the angle brackets, following the character sequence of the string, one character will replace the value of PARAMETER at one time.

Example:

```
FORC        I, <012345>
M&I         EQU             I
ENDM
```

EXITM:

Syntax: EXITM

Description:

EXITM command is to make the macro expansion to end in advance, and it is often used when there is an error in the macro.

Example:

```
MEM         MACRO       value
.IF         value>=10
            ERROR       value must < 10
            EXITM
.ENDIF
            M&value     EQU    value
            END
```

Operation symbols in macro instruction

&

Syntax: …&PARAM

Description: In the macro, make the corresponding string to convert to parameters.

Example:

```
FORC        I, <012345>
M&I           EQU         I                    ; M0/1/2/… EQU     0/1/2/…
ENDM
```

%

Syntax: %PARAM Description：In the macro, make the corresponding string to convert to numerical values. Example:

```
ERROR_      MACRO           E1    E2    E3
ERROE    E1 E2    E3
ENDM
TEMP  =      10
ERROR_ TEMP IS %TEMP
```

After compiling, it will display the error information of TEMP IS 0XA.

!

Syntax: ! Character

Description: In the macro, make the next character to be without any special significance.
Example:

```
ENUM    MACRO    CH
FORC    I, <012345>
CH!&I   EQU      I          ; CH&0/1/… EQU    0/1/…
ENDM
ENDM

ENUM    M                   ; M0/1/2/… EQU 0/1/2/…
```

The above example uses two levels of macro, and "ENUM M" is expanded to:

```
FORC    I, <012345>
M&I     EQU   I             ; M0/1/2…    EQU   0/1/2/…
ENDM
```

If ！ is not used, and & is used at first expansion.

```
FORC    I, <012345>
M&I     EQU   I             ; M0/1/2…    EQU   0/1/2/…
ENDM
```

While in the second expansion, it is already the variable MI.

LOCAL

Symbol in macro to support the local. Example:

```
XXX      MACRO
LOCAL    XX1, XX2
JMP      XX1
XX1:
JMP      XX2
XX2:
NOP
ENDM
```

XXX can be called for multiple times, but XX1, XX2 cannot be repeatedly defined.

Default and variable parameters

Look at the following macros, when the first parameter is missing, it will not generate any errors.

```
ADD2    MACRO    A, B
        DW       A+B
        ENDM
ADD2    4                   ; DW+4
```

If you want to not omit the first parameter, you can rewrite as follows:

```
ADD2    MACRO    A: REQ, B
        DW       A+B
        ENDM
 ADD    2 , 4               ; Error generating
```

And if you want to omit the second parameter and use the default value, you can rewrite as follows:

```
ADD2    MACRO    A: REQ, B: =0;  或  B: =< expression >
        DW   A+B
        ENDM
ADD2    4              ;DW    4+0
```

Finally, if you want a variable, you can rewrite as follows:

```
ADDN    MACRO    params: VARARG
        TEMP = 0
        FOR      I, <params>
        TEMP = TEMP+I
        ENDM
        DW  TEMP
        ENDM
ADDN    1, 2, 3, 4          ; DW      10
```

3.5 Conditional Compilation Control

Syntax:

```
IF          expression          ;  if it is not 0, it will be true
IFE         expression          ;  if it is 0, it will be true
IFB         <argue>             ;  if it is blank, it will be true
IFNB        <argue>             ;  if it is not blank, it will be true
IFDEF       symbol              ;  if symbol is defined, it will be true
IFNDEF      symbol              ;  if symbol is not defined, it will be true
IFIDN       <str1>, <str2>      ;  if str1 == str2, then it will be true
IFDIF       <str1>, <str2>      ;  if str1 <> str2, then it will be true
IFIDNI      <str1>, <str2>      ;  if str1 == str2 (case insensitive), then it will be true
IFDIFI      <str1>, <str2>      ;  if str1 <> str2 (case insensitive), then it will be true
ELSEIF      expression
ELSEIFE     expression
ELSEIFB     <argue>
ELSEIFNB    <argue>
ELSEIFDEF   symbol
ELSEIFNDEF  symbol
ELSEIFIDN   <str1>, <str2>
ELSEIFIDNI  <str1>, <str2>
ELSEIFDIF   <str1>, <str2>
ELSEIFDIFI  <str1>, <str2>
ELSE        ENDIF
```

Description:

IF command is used to check the specific conditions and the compiling of control program, and ended with ENDIF command. ELSEIF/ELSE command is selectable. And nesting is available.

Example:

To avoid parameter blank, the following syntax can be used:

```
 TEMP = 0
   FOR         I <ZERO ONE THREE>
   IFNB         <I>
   I            EQU      TEMP
   ENDIF
   TEMP =   TEMP+1
   ENDM
```

To avoid. H file to be repeatedly loaded, the following syntax shall be used:

```
IFNDEF          —TESTFILE—          ;  the beginning of loading program
—TESTFILE—   EQU       1            ;  program content
…
ENDIF                               ;  end of program
```

Nesting is also available:

```
IF   VAR== 1
…
     ELSEIF   VAR      <= 10
…
IF   VAR >5
…
     ELSE
…
     ENDIF
…
     ELSEIF VAR     <= 100
…
     ELSE
…
     ENDIF
```

Error display under conditional compilation

Syntax:

```
.ERR                                ;  force to generate error
.ERRNZ      expression              ;  if it is not 0, the error will be generated
.ERRE       expression              ;  if it is 0, the error will be generated
.ERRB       <argue>                 ;  if it is blank, the error will be generated
.ERRNB      <argue>                 ;  if it is not blank, the error will be generated
.ERRDEF     symbol                  ;  if symbol is defined, the error will be generated
.ERRNDEF    symbol                  ;  if symbol is not defined, the error will be generated
.ERRIDN     <str1>, <str2>          ;  if tr1==str2, then the error will be generated
.ERRDIF     <str1>, <str2>          ;  if str1< >str2 , then the error will be generated
.ERRIDNI    <str1>, <str2>          ;  if str1==str2 (case insensitive), then the error will be generated
.ERRDIFI    <str1>, <str2>          ;  if str1< >str2 (case insensitive), then the error will be generated
.ERROR      <string>                ;  force to generate error, and use string as error information
.ECHO       <string>                ;  generate string as error information
```

Description: These conditional compilation commands are used to detect some special conditions, and can output an error information to alert the user.

Example:

```
ECHO        Here be compiled        // The information will be displayed when it is compiling to this line
```

For the following macro definitions, we can change the way of expression.

```
ADD2        MACRO                   A: REQ,     B: = <0>
            DW                      A+B
            ENDM
```

New expression:

```
ADD2        MACRO                   A, B
            .ERRB                   <A>
            IFB                     <B>
            DW                      A
            ELSE
            DW                      A+B
            ENDIF
            ENDM
```

Obviously, this is not a good way, just a demonstration.

.LIST:

Syntax: **.LIST**

Description:

. LIST is used to include the source code under the LIST command into the list file, which is the default setting.

Example:

```
.LIST
…                                   ;  It will be shown in the list file
.NOLIST
```

.NOLIST:

Syntax: **.NOLIST**

Description:

. NOLIST is to disable the list function till another. LIST to open it.Example:

```
.NOLIST
…                                   ;  It will not be shown in the list file
.LIST
```

.LISTIF:

Syntax: **.LISTIF**

Description:

. LISTIF list is used to compile the commands, even if the command has not been compiled, these commands will be listed

in the list file.
Example:

```
.LISTIF IF 0
 …                                   ;  It will be shown in the list file
ENDIF
```

.NOLISTIF:

Syntax: **.NOLISTIF**
Description:
. NOLISTIF is used to hide the instructions which have not been compiled, which is the default setting.
Example:

```
.NOLISTIF IF 0
 …                                   ;  It will not be shown in the list file
ENDIF
```

.LISTMACRO:

Syntax: **.LISTMACRO**
Description:
. LISTMACRO is to list the command that can generate program code or data in a macro, which is the default setting.
Example:

```
.LIST     MACRO     TEMP =0
          REPEAT 8
          DW    TEMP                ;  It will be shown in the list file
          DEMP = TEMP+1             ;  It will not be shown in the list file
          ENDM
```

.NOLISTMACRO:

Syntax: **.NOLISTMACRO**
Description:
. NOLISTMACRO is to keep the code in macro away from expansion.
Example:

```
.NOLIST   MACRO     TEMP = 0
          REPEAT    8
```

.LISTMACROALL:

Syntax: **.LISTMACROALL**
Description:
.LISTMACROALL is to list all instructions in a macro.
Example:

```
.LIST     MACROALL     TEMP = 0
          REPEAT 8
          DW    TEMP                ;  It will be shown in the list file
          DENP = TEMP+1             ;  It will be shown in the list file
          ENDM
```

Appendix

Appendix I Compiler Error Inforamtion Description

Table I-1 Error Information List

No.	Name	Prompt	Meanings
1	Syntax_Error	Syntax error happen !!!	Syntax error
2	Detrop_SEnd	Not Need) but find !!!	Existence of)
3	Miss_SStart	Need (but not find !!!	Absence of (
4	Miss_SEnd	Need) but not find !!!	Absence of)
5	Detrop_LEnd	Not Need } but find !!!	Existence of }
6	Miss_LEnd	Need } but not find !!!	Absence of }
7	Detrop_MarkEnd	Not Need */ but find !!!	Existence of '*/' symbol
8	Miss_MarkEnd	Need */ but not find !!!	Absence of '*/' symbol
9	Need_Space	Need space to next symbol !!!	Need space between symbols
10	Detrop_Space	Not need space here !!!	Not need space here
11	Detrop_Char	Not need char here !!!	Not need char here
12	Need_Value	Here need a value !!!	Absence of operational element
13	Bad_Value	Invalide value !!!	Invalide value
14	Bad_Char	Need format 'X' here !!!	Need char format' here
15	No_Bit	Invalid bit operation !!!	Invalid bit operation
16	Need_Bit	Here need format : XX.bit !!!	'XX.bit' format is needed here
17	Need_Imm	Here need format : #xx !!!	'#XX' format is needed here
18	No_Operator	Here +-*/!~() HIGH/LOW/MID can't access !!!	Operator +-*/!~() cannot access HIGH/MID/LOW
19	Bad_HMLJ	only Label$H/M/L/J can access !!!	Only Label$H/M/L/J can access
20	Bad_HML	only Label$H/M/L can access !!!	Only Label$H/M/L can access
21	Attrib_Order_Error	HIGH/MID/LOW must before -,!,~ !!!	Operator -,!,~ must be in the front of HIGH/MID/LOW
22	Attrib_Repeat_Error	HIGH/MID/LOW repeat too much !!!	HIGH/MID/LOW repeat too much
23	Bad_String	Expect \"string..\" here !!!	Absence of sting here
24	UnAccess_Symbol	Symbol not access here !!!	Symbol cannot access here
25	Undef_Symbol	Undefined Symbol !!!	Undefined symbol
26	Unknow_Command	Unknown Command !!!	Unknown command
27	Over_0x100	The value must limit 0 - 0xFF !!!	The value must be limited among 0x00 - 0xFF
28	Over_0x10000	The value must limit 0 - 0xFFFF !!!	The value must be limited among 0x0000 - 0xFFFF
29	Over_Range	The value over range !!!	The value is over range
30	Over_Imm_Range	The #nn value over range !!!	#nn(Immediate value) is over range
31	Over_Bit1_Range	The value (MM.b), MM value over range !!!	RAM value is over range
32	Over_Bit2_Range	The value (mm.B), B value over range !!!	The Bit value of RAM is over range
33	Code_OverWrite	The Code has overwrite from here, please check it !!!	The Code will be overwritten from here, please check it
34	Data_OverWrite	The Data has overwrite from here, please check it !!!	The Data will be overwritten from here, please check it
35	Over_Code	Over Program-Code Range !!!	Over the range of User ROM Size
36	Over_Voice	Over ROM Code Range !!!	Over the range of ROM Size
37	Over_RAM	Over RAM Range !!!	Over the range of RAM Size
38	Jmp_Over	JMP, CALL, ... command over range !!!	JMP, CALL, ... instruction is over range
39	Over_OJump	Over the JUMP range !!!	Over the range of JUMP instruction
40	Play_Over	PLAY command over range !!!	PLAYinstruction is over range
41	Over_List	The List buffer has over size, force disable list !!!	List buffer has been over the range, and force to disable list
42	Over_Stack	Over Stack, Please check your program for loop-include !!!	Over Stack, please check program
43	Over_TextEqu	Over TextEqu, Please check your program for loop-textequ !!!	TextEqu Over, please check the definition of textequ in the program
44	Over_Buf2	Over Stack, Please check your program for loop-macro !!!	Over Stack, Please check the loop-macro in the program
45	Buffer_Over	Buffer over range to record data !!!	Buffer is over the range of record data
46	Inter_Error	Happen internal unknow error !!!	Unknown internal error is happened
47	INI_Error	.INI happen error, stop read !!!	.INI file has error, stop reading
48	Source_Error	Source error, stop compile !!!	Source error, stop compiling
49	File_Null	File not exist, stop compile !!!	File is not existed, stop compiling
50	No_Chip	missing CHIP SNxxx at head of program !!!	Absence of chip name declaration
51	Chip_Fail	Wrong device selection !!!	Chip name declaration error

52	Chip_Redefined	Chip select fail, can't redefined chip !!!	Chip selection is failure, chip cannot be redefined
53	Not_In_System	The command only be supported at system !!!	The instruction only support this system
54	Not_The_uC	The command not be supported at the serial uC !!!	The instruction doesn't support this serial of MCU
55	ICE_Break_Format	Here need format : .Command reg1, reg2 ... !!!	.Command reg1, reg2 …format is needed
56	Align_Format	Only access .ALIGN number, number = 2, 4, 8,.., 65536 !!!	Only access .ALIGN) value , which is equal to 2,4,8..,65536
57	Text_Format	Must use the format .TEXT { .. } !!	.TEXT { .. } format must be used
58	Instr_Format	At INSTR & SUBSTR, the start pos must >= 1 !!!	INSTR & SUBSTR starting position must be greater than or equal to 1
59	Miss_Colon	Need : but not found !!!	Absence of : symbol
60	Miss_Dot	Need , but not found !!!	Absence of , symbol
61	Miss_SDot	Need . but not found !!!	Absence of . symbol
62	Miss_equ	Need = but not found !!!	Absence of = symbol
63	Miss_dequ	Need := but not found !!!	Absence of := symbol
64	Miss_ALU	Need A but not found !!!	Absence of ACC Register
65	Bad_AA	Only @@: be accessed !!!	Only @@: can be access
66	Loss_AA	Need @@: but lost !!!	Absence of @@: symbol
67	Label_Redef	The Label has exist !!!	The Label has existed (Label repeat)
68	Symbol_Redef	The Symbol has been defined !!!	The Symbol has been defined (Symbol repeat)
69	Symbol_BadType	The Symbol has other type !!!	The Symbol has other types
70	Symbol_Label	The Symbol must be label or set by EQU !!!	The Symbol must be label or set by EQU
71	Resource_Redef	The Resource has exist !!!	The Resource file has existed
72	Resource_NoUse	The Resource file not be used !!!	The Resource file cannot be used
73	Resource_Error	Loss the Resource or translate fail !!!	Lose the Resource file or transfer failures
74	Bad_Resource	Unexpect filename, please define before PROGRAM !!!	Unexpect filename, please define before PROGRAM
75	Resource_NoFind	Resource file not be find !!!	Resource file cannot be found
76	Resource_NoTran	Resource file translate fail !!!	Tansfer error of Resource file
77	Macro_Error	Macro error, stop compile !!!	Macro error, stop compiling
78	MacroName_Error	Macro_Name error, stop compile !!!	Macro name error, stop compiling
79	Macro_Redef	MACRO redefined !!!	MACRO is repeatedly defined
80	Loss_Param	Macro request parament loss !!!	Required paraments to lose Macro
81	Tran_Param_Val	Check fail for translate parament to value !!!	Check failure of parament to transfer value
82	EXITM_Fail	EXITM must be placed in MACRO/REPEAT/FOR/FORC .. ENDM !!!	EXITM must be placed in MACRO/REPEAT/FOR/FORC .. ENDM
83	TextEqu_Error	Define TEXT must be used as <...> !!!	<...> must be used to define TEXT
84	TextEqu_BeUsed	TEXTEQU be used before defined !!!	TEXTEQU shall be used before defined
85	TextEqu_Assign	TEXT must be assigned by TEXTEQU, CATSTR, ... !!!	TEXT must be assigned by TEXTEQU, CATSTR, ...
86	Need_TextStr	Need TEXT or <..> after TEXTEQU, CATSTR, ... !!!	TEXT or <..> must be followed TEXTEQU, CATSTR, …
87	Condiction_Error	Condiction error, stop compile !!!	Condiction error, stop compiling
88	Detrop_ELSE	Not need ELSE here !!!	Existence of "ELSE" instruction
89	Detrop_ELSEIF	Not need ELSEIF here !!!	Existence of "ELSEIF" instruction
90	Detrop_ENDIF	Not need ENDIF here !!!	Existence of "ENDIF" instruction
91	Loss_ENDIF	Need ENDIF but not found !!!	Absence of "ENDIF" instruction
92	ERR_Force	Forced error	Forced error
93	ERR_True	Forced error – expression false(not 0)	Forced error – expression is false(not 0)
94	ERR_False	Forced error - expression true(0)	Forced error - expression is true(0)
95	ERR_Blank	Forced error - string blank	Forced error – string is blank
96	ERR_NBlank	Forced error - string not blank	Forced error - string is not blank
97	ERR_Define	Forced error - symbol defined	Forced error - symbol is defined
98	ERR_NDefine	Forced error - symbol not defined	Forced error - symbol is not defined
99	ERR_Dif	Forced error - string different	Forced error – string is different
100	ERR_Idn	Forced error - string identical	Forced error – string is identical
101	LOG_ANDOR	Only && || can be use for logic operator	Only && and || can be used in logic operation
102	LOG_L00	missing '(' before ')'	Absence of "("before ")"
103	LOG_L01	missing ')' before EOL	Absence of ")" before EOL
104	LOG_L02	missing ')' before THEN/GOTO	Absence of ")" before THEN/GOTO
105	LOG_AELSE	missing .IF before .ELSExx / .ENDIF	Absence of IF before .ELSExx / .ENDIF
106	LOG_BENDIF	missing .ENDIF before EOF	Absence of .ENDIF before EOF

107	LOG_AENDW	missing .WHILE before .ENDW	Absence of .WHILE before .ENDW
108	LOG_AUNTIL	missing .REPEAT before .UNTIL / .UNTILDZ / .UNTILIZ	Absence of .REPEAT before .UNTIL / .UNTILDZ / .UNTILIZ
109	LOG_ASWITCH	missing .SWITCH before .CASE .DEFAULT .ENDS	Absence of .SWITCH before .CASE .DEFAULT .ENDS
110	LOG_ABREAK	missing .WHILE .REPEAT .SWITCH before .BREAK	Absence of .WHILE .REPEAT .SWITCH before .BREAK
111	LOG_ACONTINUE	missing .WHILE .REPEAT before .CONTINUE	Absence of .WHILE .REPEAT before .CONTINUE
112	LOG_GTLT	Use 'mem/acc >,>=,<,<=,==,!= mem/imm' for logic operator	'>,>=,<,<=,==,!= ' logic operator is requried
113	LOG_PATTERN	Syntax Error, Ex : .CMD 'bit && mem > imm || !bit'	Syntax Error, Ex : .CMD 'bit && mem > imm || !bit'
114	LOG_NOTBIT	only '! bit' can be access for logic operator	Only '! bit' can be access in logic operators
115	LOG_IFGO	only '.IF log THEN/GOTO xxx' can be access for logic operator	Only '.IF log THEN/GOTO xxx' can be access in logic operators
116	LOG_IFTHEN	for logic '.IF log THEN cmd', the 'cmd' must 1 word	For logic judgement, 'cmd' of '.IF log THEN cmd' must be 1 word
117	LOG_GT255	Use 'mem/acc > #255' or 'mem/acc <= #255' exist logic question	There is logic problem to use 'mem/acc > #255' or 'mem/acc <= #255' operation
118	LOG_LT0	Use 'mem/acc < #0 ' or 'mem/acc >= #0' exist logic question	There is logic problem to use 'mem/acc < #0 ' or 'mem/acc >= #0' operation
119	OP_NEED_INT	the expression can't access label or reg for operator	The operator of operation cannot access label or register
120	Public_Syntax	Need format : Public name1 [,name2 ..]	Public name1 [,name2 ..] format is needed
121	Public_Keyword	Can not use keyword for public name !!!	Public name cannot use keyword
122	Public_Redefined	The public name has been declared !!!	The public name has been declared
123	Extern_Syntax	Need format : Extern name1 [,name2 ..]	Extern name1 [,name2 ..] format is needed
124	Extern_Redef	Extern name has other type !!!	Extern name has other types
125	Bug_B0MOV_86	The command B0MOV M, #I, but #I can't be #0E6, #0E7h	B0MOV M, #Iinstruction,#I cannot be #0E6h,#0E7h
126	Bug_S8387P	SN8P1602, SN8P1603 and SN8P1604 has following DW limitation:	SN8P1602, SN8P1603 and SN8P1604 has following DW(Define Word) limitation
		If high byte = 0 then the bit 0 and bit 1 of low byte must be zero	If high byte = 0, then the bit 0 and bit 1 of low byte must be zero
127	Bug_ICE_AF	Instant addressing mode can not use #0AFh, please call SONIX FAE	Instant addressing mode can not use #0AFh, please call SONIX FAE
128	Need_AluImm	Here need parament : ALU or #xx !!!	ALU or #xx type parameters are needed by operator
129	Msg_AsmRETI	Before RETI, please use <POP> <B0XCH A, m> to protect FZ	Before use RETIinstruction, please use <POP> <B0XCH A, m>instruction to protect FZ
130	Msg_AddPCL	The JMP command over 256 boundary	JMPinstruction is over the range of 256
131	Chk_Code_0_3	At 0.. 3, no find JMP (>=8) command !!!	JMP(>=8)instruction is not found at 0..3 bit address
132	Chk_Code_4_7	The Code Area 4..7 be reserved for Test !!!	4..7 bit address in code area is reserved for test
133	Chk_Option_Loss	The code_option loss following items !!!	Code Option lose the following itens
134	Chk_Option_Pos1	.Code_Option must be declared after CHIP defined !!	.Code_Option must be declared after CHIP is defined
135	Chk_Option_Pos2	The code_option can't set at here !!	code option can't be set at here
136	Chk_Option_Redef	The code_option has redefined !!	code option is repeatedly defined
137	Chk_Option_NoDef1	The code_option not be defined !!	code option is not defined
		Please use the following code to defined .Code_Option	Please use the following code to defined code option
138	Chk_Option_NoDef2	The code_option not be defined !!	code option is not defined
139	Opt_HighClk_2	At RC mode, HighClk div 2 must Enable !!	At RC mode, HighClk div 2 must be enabled
140	Opt_OSG_Enable	At slow crystal, OSG must Enable !!	At slow crystal, OSG must be enabled
141	Over_IP_Bound	The code may be over 16K boundary, please check it !!!	Please check whether the code is over 16K
142	Over_0x8087	The value must limit 0x80 - 0x87 !!!	The value must be limited among 0x80 - 0x870x80 - 0x87
143	Over_0x808F	The value must limit 0x80 - 0x8F !!!	The value must be limited among 0x80 - 0x8F
144	RAM_ReadOnly	This is read-only RAM ..	This is read-only RAM .
145	RAM_WriteOnly	This is write-only RAM ..	This is write-only RAM .
146	RAM_NoUse	The RAM can't use ..	The RAM is not available
147	BIT_ReadOnly	This is read-only BIT ..	This is read-only BIT .
148	BIT_WriteOnly	This is write-only BIT ..	This is write-only BIT .
149	BIT_NoUse	The BIT can't use ..	The BIT is not available
150	BIT_MoveOnly	Only can use MOV command to change the bit ..	Only MOVinstruction can be used to change the state of bit
151	Roll_Pos	.Rolling_Code command only be placed	Rolling Code command only can be declared in .code

		at .CODE segment !!!	
152	Roll_Cnt	The maximum size of Rolling Code is 4 words !!!	The maximum size of Rolling Code is 4 words
153	Chip_Format	only <CHIP xx, PIC> can be access	Only <CHIP xx, PIC> can be access
154	AsmOpt_Format	.Assembly Option num title, str0, str1 !!	In inc file, code option format has errors
155	PicAsm1_Format	Only [Command Mem, W/F] can access !!	Only[Command Mem, W/F] can be access
156	PicAsm2_Format	Only [Command Mem, Bit (< 8)] can access !!	Only[Command Mem, Bit (< 8)] can be access
157	PicAsm3_Format	Only 0xD1 .. 0xD7 can access !!	Only0xD1..0xD7 can be access

Appendix II List of Menu Commands, Tools and Shortcuts

The following is listed each commands in menu, icons in toolbar, default shortcuts and the relevant descriptions.

File menu and File commands

Table Ⅱ-1 File mune and File naming

File menu	Toolbar Icon	Shortcut key	Description
New		Ctrl+N	Create a new source file or text file
Open		Ctrl+O	Open an existing file
Close			Close the current file
Save		Ctrl+S	Save the current file
Save As…			Save and rename the current file
New Project			Create a new project
Open Project			Open an existed project
Close Project			Close the current project
Recent Project			Indicate the current project
1~9			Open recently used source files or text files
Exit			Exit and prompt to save file

Edit menu and Edit commands

Table Ⅱ-2 Edit Menu and Commands of Editor

Edit menu	Toolbar Icon	Shortcut key	Description
Undo		Ctrl+Z	Undo the last operation
Redo		Ctrl+Y	Restore the last operation
Cut		Ctrl+X	Cut the selected text and paste to clipboard
Copy		Ctrl+C	Copy the selected text to clipboard
Paste		Ctrl+V	Paste the text in clipboard
Select All		Ctrl+A	Select All
Find		Ctrl+F	to search in the current file
Find in Files			to search in several files
Replace		Ctrl+H	Replace specific text
Prev BookMark		F2	Move the cursor to the previous mark
Next BookMark		Shift+F2	Move the cursor to the next mark
Toggle BookMark		Ctrl+F2	Place a mark in current line
Clear BookMarks		Ctrl+Shift+F2	Clear all marks in current file

View Menu

Table Ⅱ-3 View Menu (View)

View menu		Toolbar Icon	Shortcut key	Description
Workspace			Alt+0	Open workspace window
Output			Alt+2	Open output window
Debug Windows	Watch		Alt+3	Open Watch window
	Call Stack		Alt+7	Open Call Stack window
	Memory		Alt+6	Open Memory window
	Variables		Alt+4	Open Variables window
	Registers		Alt+5	Open Registers window
	Disassembly		Alt+8	Open Disassembly window
Prev Error			Shift+F4	Jump to the previous error
Next Error			F4	Jump to the next error

Debug Menu and Debugging Commands

Table Ⅱ-4 Debug Menu and Debugging Commands（Debug）

Debug menu	Toolbar Icon	Shortcut key	Description
Build		F7	Build
Rebuild All			Rebuild whole project and ignore independencies
Download		F8	Download .bin into rom-emulate
Reset		Ctrl+F5	Restart program
Go		F5	Start or continue to run program
Break		F5	Stop running program and insert debugging
Stop Debugging		Shift+F5	Stop debugging
Single		F11	Run to next statement at single step
Step Over		F10	Skip over the functions at single step
Step Out		Shift+F11	Jump out the current function at single step
Run to Cursor		Ctrl+F10	Run program to the line of pointer
PC to Cursor		F12	Run to the line of mouse
Breakpoint		F9	Insert or delete breakpoints
Breakpoints		Alt+F9	Edit the breakpoints in program
Remove all Breakpoints		Ctrl+Shift+F9	Delete all breakpoints
Fill RAM			Fill RAM via data
Animate Single			Automatically run at single step
Animate StepOver			Automatically skip over the function and run at single step
Prov Single Trace		Ctrl+Shift+F11	Trace the previous statement
Prov Trace		Ctrl+Shift+F3	Track to the previous 64 instructions
Next Trace		Ctrl+F3	Trace to the next statement

Utility Menu

Table Ⅱ-5 Utility Menu and Commands（Utility）

Utility menu	Toolbar Icon	Shortcut key	Description
Report			Translate .SN8/.BIN file to .RPT file
Output.HEX			Translate .SN8/.BIN file to .HEX file
Add Print Port…			Add print port to ICE application

Easy Writer			Work with Easy Writer to program IC
LCD Simulator			LCD simulation

Window Menu

Table Ⅱ-6 Window Menu (Windows)

WindowsMenu	Toolbar Icon	Shortcut key	Description
Cascade			Set the cascading of windows
Tile			Set the tiling of windows
Arrange Icons			Arrange Icons on the bottom of the window
1~9			Activate the selected target of window
Windows…			Current window operation

Help Menu

Table Ⅱ-7 Help Menu（Help）

HelpMenu	Toolbar Icon	Shortcut key	Description
About Assembly…			Display the program information, version number and copyright.

Appendix III Pseudo-instruction List

Pseudo-instruction	Description
CHIP, ENDP	Start and end of program
EQU, =, TEXTEQU CATSTR, SUBSTR, SIZESTR, INSTR	Variable expressing method
. CODE, . DATA, . CONST	Definition segment
ORG, . ALIGN	Set the address of program and data
DS	Data definition
DW	Define a WORD
INCLUDE, INCLUDEBIN, INCLUDESTD	Include another file
TITLE	Title definition of user
MACRO, EXPAND, ENDM, REPEAT, FOR, FORC, EXITM	Macro
. LIST, . NOLIST, . LISTIF, .NOLISTIF, . LISTMACRO, . NOLISTMACRO, . LISTMACROALL	List file control
IF, IFE, IFB, IFNB, IFDEF, IFNDEF, IFIDN, IFDIF, IFIDNI, IFDIFI, ELSEIF, ELSEIFE, ELSEIFB, ELSEIFNB, ELSEIFDEF, ELSEIFNDEF, ELSE, ENDIF	Conditional compilation control
. ERR, . ERRE, . ERRNZ, . ERRB, . ERRNB, . ERRDEF, .ERRNDEF ERROR, ECHO	Conditional compilation error display
. EXEC	Execute the external program
@BIT, @INT, @FIELD	Bit operation

Appendix IV Figure List

(Click the linkage to turn to corresponding figure)

Chapter 1

No.	Linkage	No.	Linkage
1	Figure 1-1 Emulation Connection Diagram	9	Figure 1-9. Installation Information
2	Figure 1-2 SN8ICE 2K Hardware Description	10	Figure 1-10. Installation Progress Window
3	Figure 1-3. Post-connection Diagram	11	Figure 1-11. Installation Completion
4	Figure 1-4. M2IDE Installation Guide Dialog Box	12	Figure 1-12. Shortcut Icon
5	Figure 1-5. M2IDE Installation Agreement Dialog Box	13	Figure 1-13. Start M2IDE System
6	Figure 1-6. Installation Path Selection Dialog Box	14	Figure 1-14. Uninstall M2IDE
7	Figure 1-7. Window of View Path	15	Figure 1-15. Message Box of Successful Uninstalling
8	Figure 1-8 Pop-up Shortcut Setting Window		

Chapter 2

No.	Linkage	No.	Linkage
1	Figure 2-1. Integrated Development Environment	35	Figure 2-35 Help Menu
2	Figure 2-2. Open M2Asm Interface at First Time	36	Figure 2-36 Activity bar of each window
3	Figure 2-3. Editing Interface	37	Figure 2-37 Window is moving
4	Figure 2-4. Debugging Interface	38	Figure 2-38 Double-click the activity bar to pop-up window
5	Figure 2-5 Register window	39	Figure 2-39 Right-click the blank of toolbar to choose to display or not display the window
6	Figure 2-6. File Menu	40	Figure 2-40 Open M2Asm interface at first time
7	Figure 2-7 M2IDE Interface	41	Figure 2-41 M2Asm interface without opened project or file
8	Figure 2-8 Open File Menu	42	Figure 2-42 Drop-down menu of File in idle mode
9	Figure 2-9 Select the File Name	43	Figure 2-43 Drop-down menu of File after creating a New file
10	Figure 2-10 Use the Arrow Keys to Select	44	Figure 2-44 Dialog box to save file
11	Figure 2-11 Edit Menu	45	Figure 2-45 Dialog box to create a New Project
12	Figure 2-12 Find Register in the Current File	46	Figure 2-46 Header File Adding Menu
13	Figure 2-13 Information box to find register in the current file	47	Figure 2-47 Header File Selection Dialog Box
14	Figure 2-14 Compiler interface after the execution of MaskAll	48	Figure 2-48 Compiling Environment Setting Dialog Box
15	Figure 2-15 Information box to find in all files	49	Figure 2-49 Setting Dialog Box in Active State
16	Figure 2-16 Replacement Information Box	50	Figure 2-50 Compiling Configuration Information
17	Figure 2-17 View Menu	51	Figure 2-51 Compiling Error Information Prompt Box
18	Figure 2-18 Debug Menu	52	Figure 2-52 Compiling Error Interface
19	Figure 2-19 Breakpoints dialog box for the execution without breakpoint	53	Figure 2-53 Compiling Information Box after Successful Compilation
20	Figure 2-20 Dialog box for the execution with breakpoint	54	Figure 2-54 Shortcut Command Icons

21	Figure 2-21 Dialog box for setting multiple breakpoints	55	Figure 2-55 Download file selection dialog box
22	Figure 2-22 Dialog box for the selection of trigger mode	56	Figure 2-56 Successful download prompt box
23	Figure 2-23 Dialog box after setting the number of trigger	57	Figure 2-57 Debugging Interface
24	Figure 2-24 Dialog box when the program is being downloaded to ICE	58	Figure 2-58 Prompt box if emulator is not foun
25	Figure 2-25 Select the program to be downloaded	59	Figure 2-59 Array expanded in observation window
26	Figure 2-26 Successful download prompt box	60	Figure 2-60 Program running to breakpoint
27	Figure 2-27 Utility Menu	61	Figure 2-61 Directly observe the values of variables in the variable watch window
28	Figure 2-28 Dialog box after the execution of Report command	62	Figure 2-62 Jump out of a function
29	Figure 2-29 Information box after the execution of Report command	63	Figure 2-63 Run to the statement to call and display function
30	Figure 2-30 Prompt box after the execution of Easy Write command	64	Figure 2-64 Skip over function
31	Figure 2-31 Window Menu	65	Figure 2-65 Interface of running at full speed
32	Figure 2-32 Windows cascading	66	Figure 2-66 Code Options
33	Figure 2-33 Windows tiling	67	Figure 2-67 LCD software simulation panel
34	Figure 2-34 Window		

Appendix V. Relevant FAQ

Please refer to sonix FAQ：
http://www.sonix.com.tw/sonix/MCU_FAQ.jsp

SONiX reserves all of rights for further explanation on the improvements of reliability, functionality and design of the following products. SONiX assumes no responsibility caused by the application and use of products or circuit covered in this manual. SONiX's products are not designed for the application in surgical implants, life sustaining, and other fields that the failure of SONiX's product may cause injury or death to individual. If SONiX's products are used in the above mentioned fields, even if these are resulted from the negligence in the product design and manufacturing by SONiX, the user shall indemnify all costs, losses, and the attorneys' fees directly or indirectly generated from reasonable personal injury or death, and ensure that SONiX and its employees, subsidiaries, affiliates and distributors have nothing to do with the above matters.

Corporate Headquarters
10F-1, No.36, Taiyuan Street, Chupei City, Hsinchu, Taiwan
TEL: (886)3-5600-888
FAX: (886)3-5600-889

Sonix Technology (Shenzhen) Co., Ltd
High Tech Industrial Park, Shenzhen, China
TEL: (86)755-2671-9666
FAX: (86)755-2671-9786

Taipei Office
15F-2, No.171 Song Ted Road, Taipei, Taiwan
TEL: (886)2-2759-1980
FAX: (886)2-2759-8180

Hong Kong Sales Office
Unit No.705,Level 7 Tower 1,Grand Central Plaza 138 Shatin Rural Committee Road, Shatin, New Territories, Hong Kong
TEL: (852)2723-8086
FAX: (852)2723-9179

Technical Supports
Sn8fae@SONiX.com.tw