FTM32ForgeIDE

User Manual

Fortior Technology (Shenzhen) Co., Ltd.

Preface

FTM32ForgeIDE, or an integrated development environment, is a software application that provides a comprehensive set of tools for developing and managing applications effectively. As a development platform, it has most of the features that R&D engineers desire. FTM32ForgeIDE is suitable for most 8-bit microprocessors and microcontrollers, so that the R&D engineers can develop new projects in a familiar development environment, Moreover, it is green software, fast and easy to be installed, extremely intelligent and user-friendly.

FTM32ForgeIDE adopts a simple and flexible interface style, and displays memory windows and register windows in categories. It supports intelligent code editing, syntax highlighting, and code navigation. With a rich set of library functions and a powerful integrated development and debugging tools, it supports the entire development lifecycle, making it user-friendly, efficient, and concise. Additionally, the software provides a robust solution framework for unified management of multiple projects. It includes compilation features with multiple optimization levels, enabling fully automated compilation and linking. It supports multiple breakpoint management methods, including conditional and address breakpoints, at both source and assembly levels. After code migration, it allows source code to be managed alongside Keil projects and directly converts Keil projects to FTM32ForgeIDE projects, offering enhanced compatibility. The platform also supports comprehensive device features, including peripheral simulation, and allows flexible debugging of drivers on target hardware. Integrated with the Fortior C51, FTM32ForgeIDE provides seamless control over the compiler, assembler, linker, and debugger within a single intelligent environment. FTM32ForgeIDE significantly enhances work productivity and saves considerable time.

Contents

Preface **2**
Contents **3**
1 FTM32ForgeIDE Introduction **5**
1.1 Menu Bar 6
1.1.1 File Menu 6
1.1.2 Edit Menu 6
1.1.3 View Menu 10
1.1.3.1 Customize User-defined Toolbar 11
1.1.4 Project Menu 13
1.1.5 Build Menu 13
1.1.6 Debug Menu 13
1.1.7 Remark Menu 15
1.1.8 Tools Menu 23
1.1.9 Window Menu 23
1.1.10 Help Menu 24
1.2 Project Settings 25
1.2.1 Settings 25
1.3 Window Introduction 30
1.3.1 Solution Browser 30
1.3.2 Class View 30
1.3.3 Properties View 30
1.3.4 Breakpoints Window 32
1.3.5 Disassembler Window 34
1.3.6 Registers Window 35
1.3.7 Memory Window 36
1.3.8 Watch Window 41
1.3.9 Command Window 43
1.3.10 Call Hierarchy Window 43
2 Create New Application **44**
2.1 Create A New Solution 44
2.2 Create A New Project 45
2.3 Create A New File 48
3 Code Editor **50**

4 Compiler 56

5 Download and Emulation 58

5.1 Download 58

5.2 Emulation 60

6 Revision History 62

1 FTM32ForgeIDE Introduction

FTM32ForgeIDE, or an integrated development environment, is a computer program that includes tools for a variety of programming and software tasks. This chapter introduces the FTM32ForgeIDE tools and windows, which aims to help the programmers to develop embedded applications quickly.

Run the FTM32ForgeIDE as administrator. The default interface is shown as below:

Figure 1-1 FTM32ForgeIDE Startup Interface

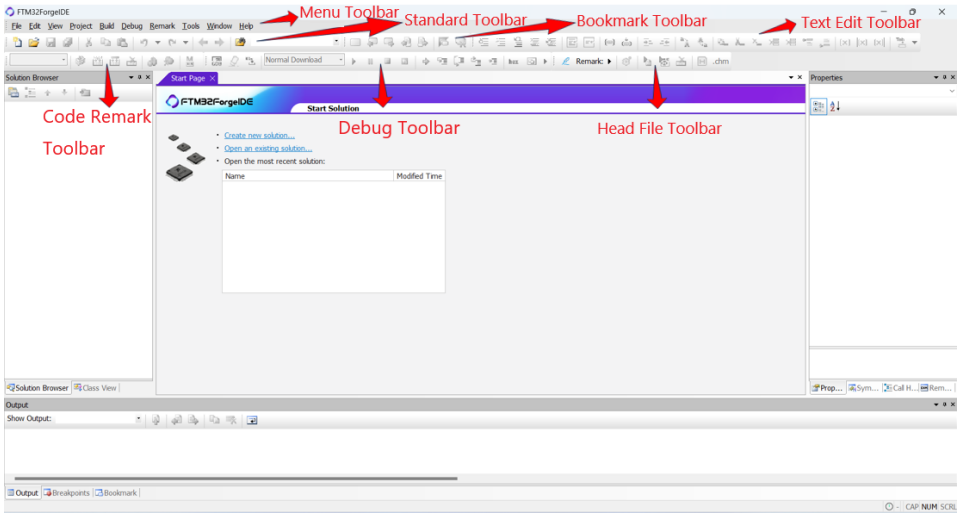


Figure 1-2 FTM32ForgeIDE Main Interface

1. Meau Bar: It integrates all tools and settings of the Fortior Riscv IDE.
2. Standard toolbar: It mainly has the following functions:
   - New, open, save and save as
   - Cut, copy, paste, undo and redo
   - Navigate backwards, navigate forwards, bookmark
3. Build toolbar: It includes compile, build, build all and stop build buttons.
4. Debug toolbar: It includes download code, go, break, stop debugging, restart, and step through.

5. Text Edit toolbar: It includes insert line comment, remove line comment, zoom in, zoom out and so on.
6. Bookmark toolbar: Use bookmarks for navigating between marked sections of the code.
7. Code Remark toolbar: View and generate help document for header files in the current project.
8. Solution Browser Window: Show all source and files in the current solutions.
9. Start Page: List the most recently used solutions.

1.1 Menu Bar

1.1.1 File Menu

File Menu	Icon	Shortcut	Description
New			Create a new project, file, and solution.
Open		Ctrl+O	Open a file.
Close			Close the file.
Open Solution		Ctrl+Shift+O	Open a project or solution.
Close Solution			Close a project or solution.
Save		Ctrl+S	Save the file.
Save As			Save and rename the file.
Save All		Ctrl+Shift+S	Save all open source and text files, including the project configuration settings.
Print		Ctrl+P	Print the selected content or area.
Print Preview			Preview the print result before printing.
Print Setup			Set printing options like the paper size, the page orientation etc.
Recent Files			List all recently used source or text files.
Recent Solution			List all recently used solutions.
Exit			Exit FTM32ForgeIDE and prompt to save files.

1.1.2 Edit Menu

Edit Menu	Icon	Shortcut	Description
Undo		Ctrl+Z	Cancel the last edit operation.
Redo		Ctrl+Y	Restore the last undone operation.
Cut		Ctrl+X	Cut the selected text to the clipboard.
Copy		Ctrl+C	Copy the selected text to the clipboard.
Paste		Ctrl+V	Paste the text from the clipboard.

Edit Menu	Icon	Shortcut	Description
Delete		Delete	Delete the selected text from the file.
Navigate Backwards		Ctrl+ -	Move the cursor back to the position occupied before a 'find' or 'go to line' command was executed.
Navigate Forwards		Ctrl+Shift+-	Move the cursor to the position before a 'Navigate Backwards' command was executed.
Find		Ctrl+F	Search for text patterns in the file.
Find/replace in Files		Ctrl+Shift+H	Searches for text patterns in several files. Or replace the specified text with another text in several files.
Go to line		Ctrl+G	Move the cursor to the specified line number in the current source file.
Replace		Ctrl+H	Replace the specified text with another text.
Format Code		Ctrl+K Ctrl+D	Format source code in the file.
Format Selected Code		Ctrl+K Ctrl+F	Format the selected source code.
Toggle Outlining		Ctrl+M Ctrl+L	Offer commands for source code outlining in the active file.
Toggle Outlining Current		Ctrl+M	Offer commands for code outlining on the current line.
View			See the following View Option table.
Indent			See the following Indent Option table.
Convert Case			See the following Convert Case Option table.
Line Operation			See the following Line Operation Option table.
Comment/Uncomment			See the following Comment / Uncomment Option table.
Space Character Operation			See the following Space Character Operation Option table.
Character Encoding			Include ANSI, UTF8, UFT8-BOM, UCS2 Big-Endian and UCS2 Little-Endian.
Breakpoint		Ctrl+Alt+B	Set the breakpoint.

View Option	Icon	Shortcut	Description
View Indent Guides		Ctrl+R, Ctrl+W	Show or hide the indent guides.
View Space			Show or hide.
Auto Wrap Line		Ctrl+E, Ctrl+W	Automatically wrap lines.
View Newline Character			Show or hide line breaks.
Zoom In			Zoom in on the text.
Zoom Out			Zoom out on the text.
Zoom Default			Set the font to 1:1 to the window size.

Indent Option	Icon	Shortcut	Description
Increase Indent			Increase the line indentation of each selected line with one tab.
Decrease Indent			Decrease the line indentation of each selected line with one tab.

Convert Case Option	Icon	Shortcut	Description
To Uppercase		Ctrl+Shift+U	Convert the selected text to uppercase letters.
To Lowercase		Ctrl+U	Convert the selected text to lowercase letters.

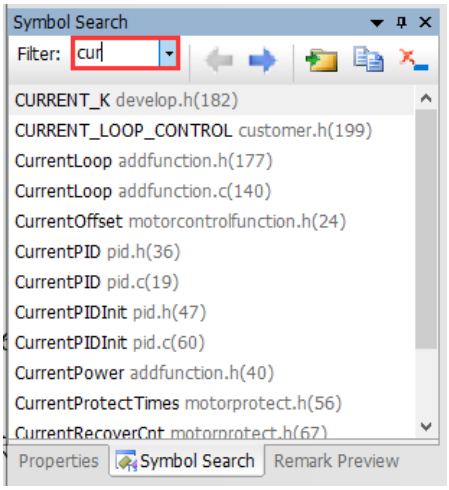
Line Operation Option	Icon	Shortcut	Description
Duplicate Current Line		Ctrl+D	Copy the current line and paste the next line.
Move Up Selected Lines		Ctrl+Alt+D	Move the selected line up.
Move Down Selected Lines		Alt+Down	Move the selected line down.
Copy Selected (Current) Line		Ctrl+Shift+C	Copy the selected line to the clipboard.
Cut Selected (Current) Line		Ctrl+Shift+X	Cut the selected line to the clipboard.
Remove Selected (Current) Line		Ctrl+Shift+Delete	Delete the text of the current line.
Remove All Blank Line			Delete all blank lines from the file.

Line Operation Option	Icon	Shortcut	Description
Remove All Blank Line(With Space Characters)			Replace blank lines with spaces in the file.
Insert Blank Line Above			Insert a blank line above the line.
Insert Blank Line Below			Insert a blank line below the line.

Comment/Uncomment Option	Icon	Shortcut	Description
Add/Remove Line Comment		Ctrl+K Ctrl+/	Convert the selected lines to comments. Or convert commented lines to code text.
Add Line Comment			Convert the selected lines to comments.
Remove Line Comment			Convert commented lines to code text.
Add Block Comment		Ctrl+?	Convert the selected code block to comments.
Remove Block Comment			Convert commented code block to code text.

Space Character Operation Option	Icon	Description
Remove Line-Head and Tail Space		Delete each leading tab, leading space, trailing tab or trailing space in the selected text.
Remove Line-Head Space		Delete each leading tab or leading space in the selected text.
Remove Line-Tail Space		Delete each trailing tab or trailing space in the selected text.
EOL to Space		Converts the newline characters into spaces.
Tab to Space(All)		Replace line-head tabs with spaces in the file.
Tab to Space(Line-Head)		Replace line-head tabs with spaces in the selected text.
Space to Tab(All)		Replace line-head spaces with tabs in the file.
Space to Tab(Line-Head)		Replace line-head spaces with tabs in the selected text.

1.1.3 View Menu

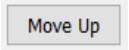
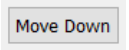
View Menu	Icon	Shortcut	Description
Solution Browser		Ctrl+Alt+L	Show the Solution Browser window. See Solution Browser for details.
Class View		Ctrl+Shift+C	Show the Class View window. See Class View for details.
Resource View		Ctrl+ Shift +E	Show the Resource View window.
Properties View		Alt+Enter	Show the Properties View window. See Properties View for details.
Bookmark Window		Ctrl+K Ctrl+W	Show the Bookmark window. You can use bookmarks for navigating between marked sections of the code. Bookmarks are displayed as magenta squares in the editor margin and are set through the toolbar buttons.
Command Window		Ctrl+Alt+A	Show the Command window.
Symbol Window		Ctrl+Alt+S	Show the Symbol Search window. Enter what you want to search. The window displays the search result and the location in the file. Click to forward or backward, click to open the directory where the file is located, click to copy file name to clipboard, click to clear the list.  Figure 1-3

View Menu	Icon	Shortcut	Description
Remark Preview		Remark Preview	Show the Remark window.
Find Result 1/ Find Result 2			Show the Result 1 / Result 2 window.
Output		Alt+2	Show the Output window.
Breakpoints		Ctrl+B	Show the Breakpoints window. See Breakpoints Window for details.
Debug Windows			Show the Debug windows.
Toolbars			Show or hide the toolbars.
Customize User-defined Toolbar…			See section Customize User-defined Toolbar for details.
Full Screen		Shift + Alt +Enter	Maximize the window to full screen.

Debug Window Option	Shortcut	Description
Watch		Show the Watch window. See Watch Window for details.
Auto Variables	Ctrl+Alt+V，A	Show the Auto Variables window.
Local Variables	Alt+4	Show the Local Variables window.
Memory		Show the Memory window. See Memory Window for details.
Disassembly	Alt+8	Show the Disassembly window. See Disassembler Window for details.
Register	Alt+5	Show the Register window. See Registers Window for details.

1.1.3.1 Customize User-defined Toolbar

Click “View” → “Customize User-defined Toolbar…” to show the User-Defined Toolbar window. As shown in Figure 1-4, the Toolbar Buttons list displays all tools that you can use. The User-defined Toolbar list displays the tools that you selected.

Click  to add the selected tool to User-defined Toolbar list, click  to remove the selected tool from User-defined Toolbar list, click  to move the selected tool up and click  to move the selected tool down.

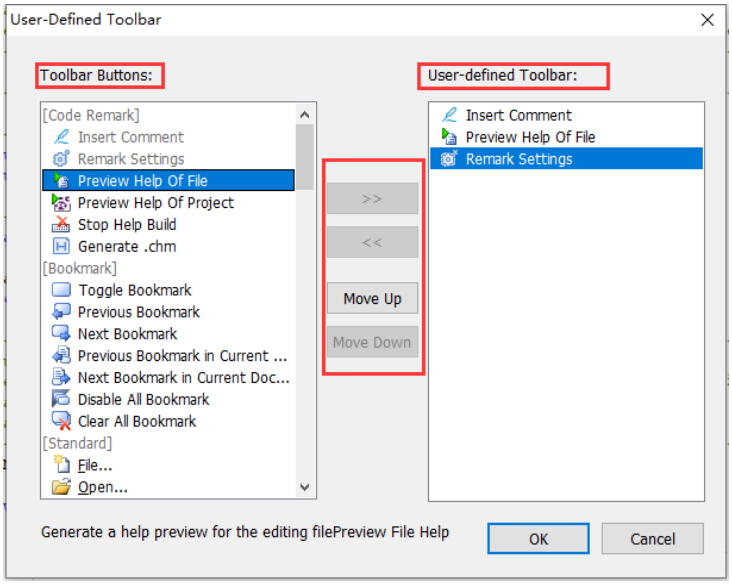


Figure 1-4

As shown in the following figure, you can move the User-Defined Toolbar as you want.

Figure 1-5

You can right click the mouse at any position of the toolbar to select to show or hide some toolbars, as shown in Figure 1-6. If the toolbar is selected, it is displayed. If not, it is hidden. You can select them according to your needs.

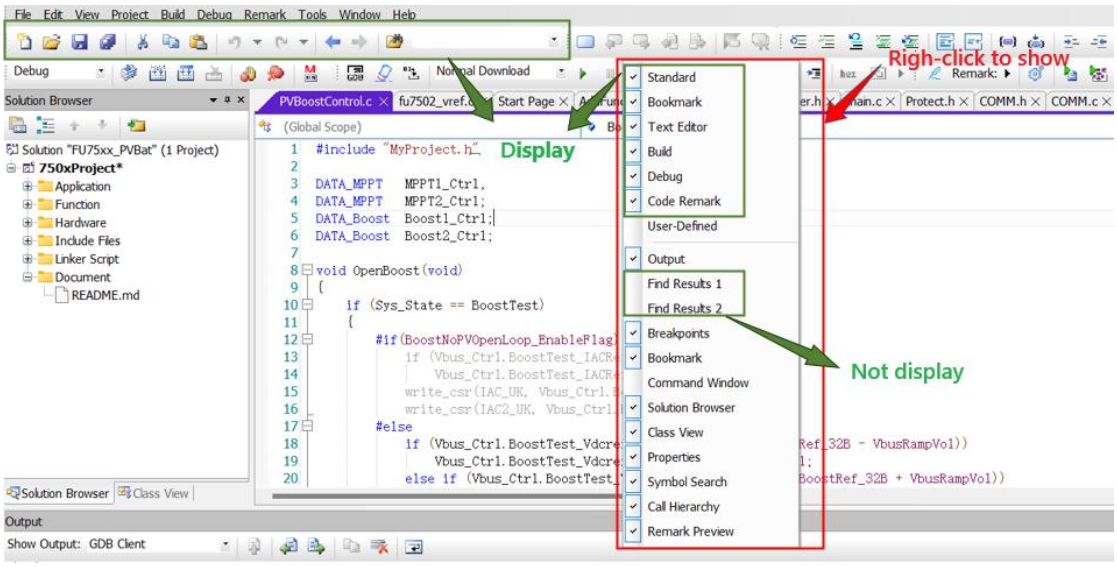


Figure 1-6

1.1.4 Project Menu

Project Menu	Description
Set Active Project	Set the current project as the active project.
Add to Project	Add a new file, folder or some existing files to the project.
Dependencies…	Show the Project Dependencies window.
Build Sequence…	Show the Compilation Generation Order window.
Settings…	Show the Project Settings window. See 1.2 Project Setting for details.
Exports Makefile…	Show the Exports Makefile(s) window.
Change Body	Show the Select Body window to change the chip type.
Insert Project into Solution…	Add an existing project to the solution.

1.1.5 Build Menu

Build Menu	Icon	Shortcut	Description
Compile		Ctrl+F7	Compile the file. Syntax check is performed during compilation and the compilation information is displayed in the Output window.
Build		F7	Build the target project, compile the modified file, and generate hex file.
Rebuild All		Ctrl+Alt+F7	Rebuild the target project, compile all sources files, and generate hex file.
Batch Build			Execute build-commands on the selected project.
Clean			Clean the target temporary file(s) created by the project
Set Active Configuration			Show the Set Active Project Configuration window. There are two configurations: Debug and Release. You can debug program built under debug version. Program built under release version is more convenient for programmers without debugging function.

1.1.6 Debug Menu

Debug Menu	Icon	Shortcut	Description
Erase Flash		Alt+F8	Erase the Flash.
Download Code		F8	Download the hex file to the Flash ROM device.
Download Mode			Program code download modes: Smart Download, Normal Download, Without

Debug Menu	Icon	Shortcut	Description
			Download
Go		F5	Start a debugging session.
Break		Ctrl+Alt +Break	Pause a debugging session.
Stop Debugging		Shift +F5	Stop a debugging session.
Restart		Ctrl+ Shift +F5	Restart a debugging session.
Step Into		F11	Execute a single-step debugging into a function. It run the its internal code step-by-step.
Step Over		F10	Execute a single-step debugging over a function. It executes all called functions without stepping into them.
Step Out		Shift+F11	Step out of the function.
Run to Cursor		Ctrl+F	Execute debugging to the current cursor position.
Show Next Statement			Show the next statement.
Quick Watch			Display the Quick Watch window.
Breakpoints		Ctrl+B	Open the Breakpoints Window. See section 1.3.4 Breakpoints Window for details.
Toggle Breakpoint		F9	Set the breakpoint on the current line.
Remove All Breakpoints			Remove all breakpoints in the program.
Disable All Breakpoints			Disable all breakpoints in the program.

1.1.7 Remark Menu

Remark: Create declaration samples for functions, data interfaces, etc. in header files. A newly created header.h file is shown in Figure 1-17 to make you understand easily (Detailed steps is shown in 2.3 Create A New File).

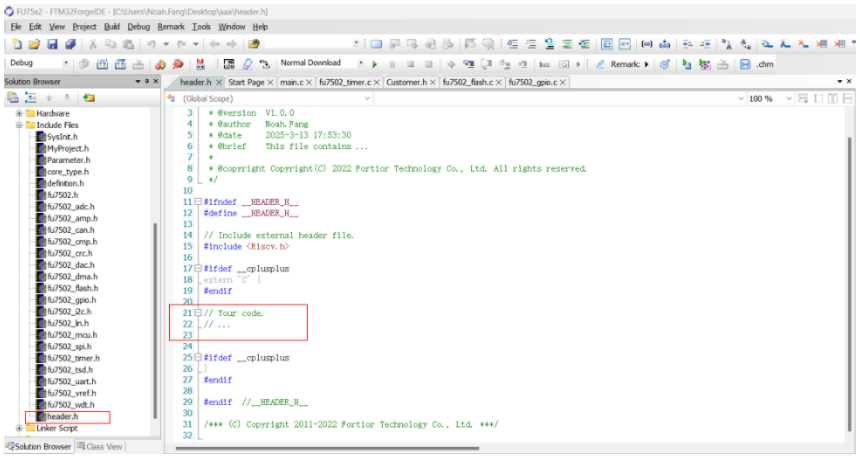


Figure 1-7

- Demo:
  - Project: Click “Remark”→ “Demo” → “Project” to add a project declaration sample in the header.h file, as the Figure 1-8

Figure 1-8

  - File: Click “Remark” → “Demo” → “File” to add a file declaration sample in the header.h file, such as function declaration, variable declaration, structure declaration, etc.

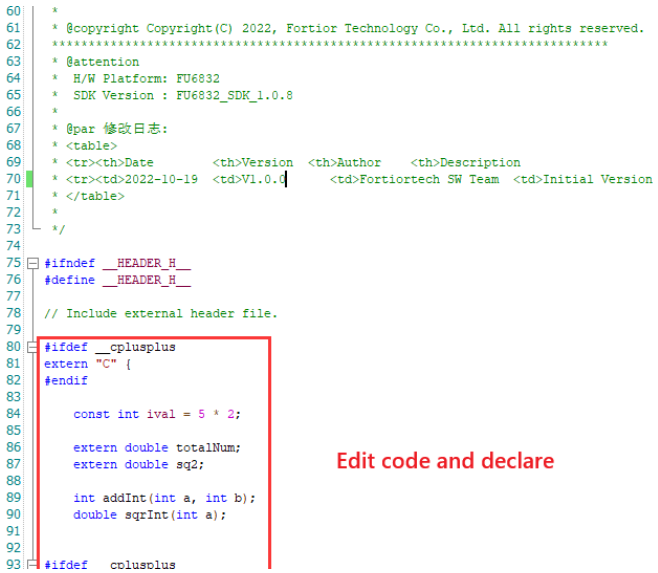


Figure 1-9

You can modify a file declaration based on the demo and program codes to declare the file as below.

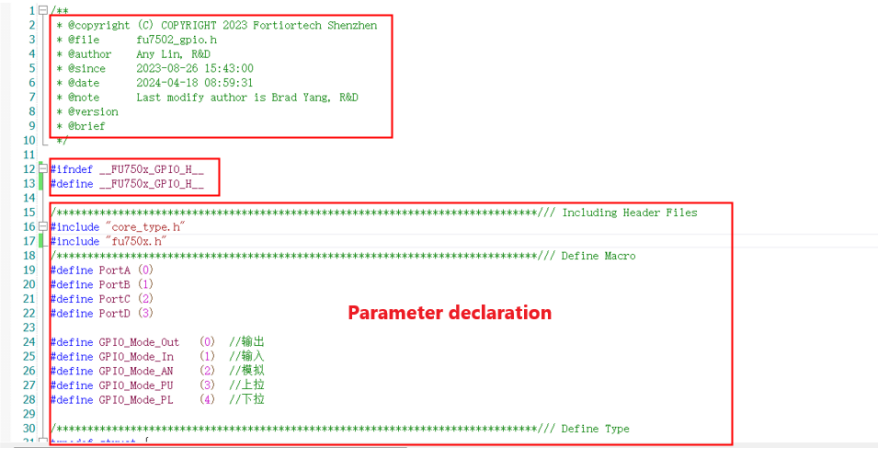


Figure 1-10

- Function: Click “Remark” → “Demo” → “Function” to add a function declaration sample in the header.h file.

```
/**@brief Reads data from a stream.
 *
 * @param[in]  buffer     Storage location for data.
 * @param[in]  size       Item size in bytes.
 * @param[out] count      number of items to be readed.
 *
 * @return  函数注册成功返回0，其它可能的值下表所示：
 * | Result Code   | Descriptions
 * | :----:        | :----:   |
 * | E_ABORT       | The draw operation was aborted.
 * | E_OUTOFMEMORY | Insufficient memory for the operation.
 * | E_INVALIDARG  | One or more parameters are invalid.
 *
 * @par 头文件
 *       csinc.h
 *
 * @par 示例
 * @code
 * void test( void )
 * {
 *    char list[30];
 *    int32_t  ret, numread;
 *
 *    // Attempt to read in 25 characters
 *    ret = fread( list, 30, &numread );
 *    printf( "Number of items read = %d\n", numread );
 *    printf( "Contents of buffer = %.25s\n", list );
 * }
 * @endcode
 *
 * @see         member add
 * @remarks     备注
 * @warning     函数使用警告
 * @pre         函数前置条件
 * @deprecated  函数过时说明                Function declaration
 */
int32_t fread(char * buffer, int32_t size, int32_t * count);
```

Figure 1-11

You can modify the function declaration based on the demo and program codes as below.

```
#define SPI_Interrupt_Disable          (0)
#define SPI_Interrupt_Enable           (1)

#define SPI_TMod_Normal                (0)
#define SPI_TMod_HighResistance        (1)

/*****************************************************************************/// Define Type
typedef struct {
    uint8 SPI_Mode;
    uint8 SPI_CPOL;
    uint8 SPI_CPHA;
    uint8 SPI_NSS;
    uint8 SPI_WireMod;
    uint8 SPI_Interrupt;
    uint8 SPI_TMod;
    uint32 SPI_Baud;
} SPI_InitTypeDef;
/*****************************************************************************/// External Symbols
/*****************************************************************************/// External Function
extern void SPI_Init(uint8 SPIX , SPI_InitTypeDef* Configuration);
extern void SPI_DoubleWire_ReadQuest(uint8 SPIX , uint8 const Data);
extern void SPI_SingleWire_ReadQuest(uint8 SPIX);
extern uint8 SPI_ReceiveData(uint8 SPIX);
extern void SPI_SendData(uint8 SPIX , uint8 const Data);      Function declaration
extern void SPI_Config(uint8 SPIX);
extern void SPI1_Test(void);

#endif
```

Figure 1-12

- Enum: Click “Remark” → “Demo” → “Enum” to add a enumeration type declaration sample in the header.h file.

```
/**
  * @brief  Interrupt Number Definition. The maximum of 32 Specific Interrupts are possible.
  */
typedef enum
{
    /******  Mini51 specific Interrupt Numbers *********************************************/

    BOD_IRQn    = 0, /*!< Brownout low voltage detected interrupt           */
    WDT_IRQn    = 1, /*!< Watch Dog Timer interrupt   */

} enum_name_t;
```

Figure 1-13

You can modify the enumeration type declaration based on the demo and program codes as below.

```
#define PortA (0)
#define PortB (1)
#define PortC (2)
#define PortD (3)

#define GPIO_Mode_Out    (0)  //输出
#define GPIO_Mode_In     (1)  //输入
#define GPIO_Mode_AN     (2)  //模拟
#define GPIO_Mode_PU     (3)  //上拉
#define GPIO_Mode_PL     (4)  //下拉

/*****************************************************************************/// Define Type
typedef struct {
    uint8 ModSel;
    uint8 PortSel;
    uint16 PINSel;
} GPIO_InitTypeDef;

typedef enum
{
    UART_DARM    = 0,
    DRAM_UART    = DMACFG0,
    I2C_DRAM     = DMACFG1,
    SPI_DRAM     = DMACFG2,
} eType_DMA_PIPE;

/*****************************************************************************/// External Symbols
/*****************************************************************************/// External Function
extern void  GPIO_Init(GPIO_InitTypeDef* Configuration);
```

Enumeration declaration

Figure 1-14

- Struct: Click “Remark” → “Demo” → “Struct” to add a structure declaration sample in the header.h file.

```
/* Data Type Definitions */
typedef volatile unsigned char  vu8;        ///< Define 8-bit unsigned volatile data type
typedef volatile unsigned short vu16;       ///< Define 16-bit unsigned volatile data type
typedef volatile unsigned long  vu32;       ///< Define 32-bit unsigned volatile data type

/**
   * @brief Get a 8-bit unsigned value from specified address
   * @param[in] addr Address to get 8-bit data from
   * @return  8-bit unsigned value stored in specified address
   */
#define M8(addr)  (*((vu8  *) (addr)))

/* Peripheral and SRAM base address */
#define FLASH_BASE            ((uint32_t)0x00000000)    ///< Flash base address
#define SRAM_BASE             ((uint32_t)0x20000000)    ///< SRAM base address
#define APB1PERIPH_BASE       ((uint32_t)0x40000000)    ///< APB1 base address
#define APB2PERIPH_BASE       ((uint32_t)0x40100000)    ///< APB2 base address
#define AHBPERIPH_BASE        ((uint32_t)0x50000000)    ///< AHB base address

/**
 * @brief Memory Mapped Structure for WDT Controller
 */
typedef struct
{
    /**
     * @var your_struct_name_t::WTCR
     * Offset: 0x00  Watchdog Timer Control Register
     * ------------------------------------------------------------------------------------------------
     * |Bits    |Field     |Descriptions
     * | :----: | :----:   | :---- |
     * |[0]     |WTR       |Reset Watchdog Timer Up Counter (Write Protect)
     * |        |          |0 = No effect.
     * |        |          |1 = Reset the internal 18-bit WDT up counter value.
     * |        |          |Note: This bit will be automatically cleared by hardware.
     * |[1]     |WTRE      |Watchdog Timer Time-out Reset Enable Control (Write Protect)
     * |        |          |Setting this bit will enable the WDT time-out reset function if the WDT up counter value has nc
     * |        |          |0 = WDT time-out reset function Disabled.
     * |        |          |1 = WDT time-out reset function Enabled.
     * |[10:8]  |WTIS      |Watchdog Timer Interval Selection
     * |        |          |These three bits select the time-out interval for the Watchdog Timer.
     * |        |          |000 = 24 * TWDT.
     * |        |          |001 = 26 * TWDT.
     * |        |          |010 = 28 * TWDT.
     * |        |          |011 = 210 * TWDT.
     * |        |          |100 = 212 * TWDT.
     * |        |          |101 = 214 * TWDT.
     * |[31]    |DBGACK_WDT|ICE Debug Mode Acknowledge Disable Control (Write Protect)
     * |        |          |0 = ICE debug mode acknowledgement effects WDT counting.
     * |        |          |WDT up counter will be kept while CPU is hanging by ICE.
     * |        |          |1 = ICE debug mode acknowledgement Disabled.
     * |        |          |WDT up counter will keep going no matter CPU is hanging by ICE or not.
     */
    /*! @brief 0x00  Watchdog Timer Control Register */
    uint32_t            WTCR;

    /*! @brief Configuration which is used to execute the demo */
    system_cfg_t        sys_cfg;

    /*! @brief CLI related configuration */
    cli_cfg_t           cli_cfg;

} your_struct_name_t;
```

Structure declaration

Figure 1-15

You can modify the Structure type declaration based on the demo and program codes as below.

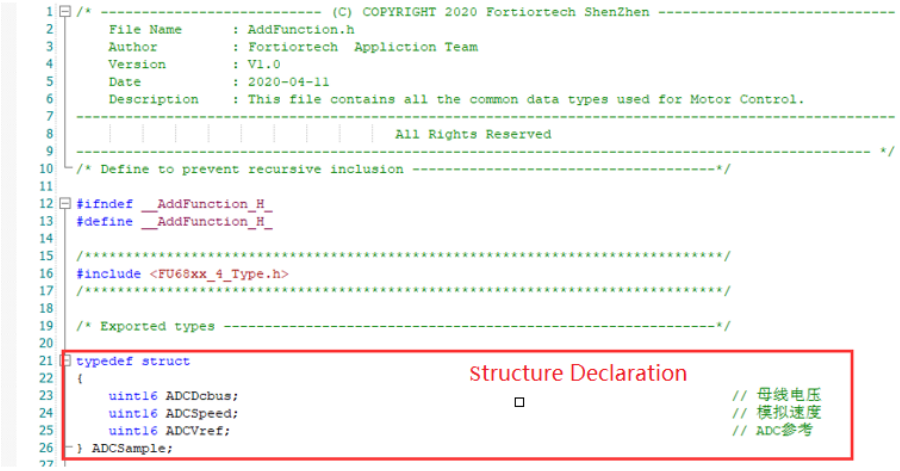


Figure 1-16

- Define: Click “Remark” → “Demo” → “Define” to add a define declaration sample in the header.h file..

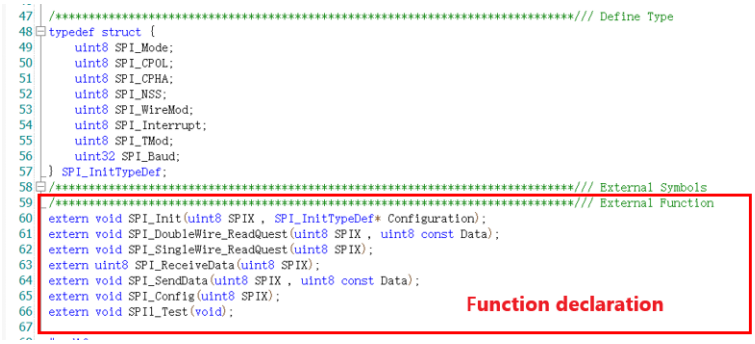


Figure 1-17

You can modify the define declaration based on the demo and program codes as below.

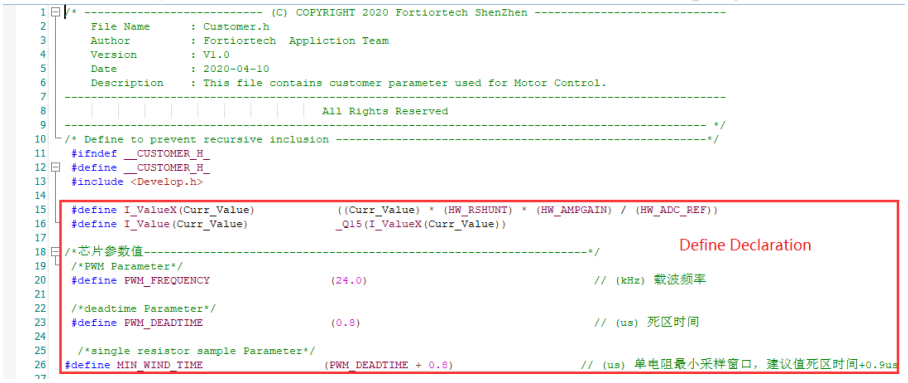


Figure 1-18

- Group: Click “Remark” → “Demo” → “Group” to add a group declaration sample in the header.h file..

```
143
144 /** @addtogroup YOUR_GROUP_NAME your group title
145   Configuration of the M0 Processor and Core Peripherals:
146     - interrupt numbers
147     - registers and bit fields
148     - peripheral base address
149     - peripheral ID
150     - Peripheral definitions
151   @{
152 */
153
154 // Your code
155 // ...
156
157
158 /*@}*/ /* end of group YOUR_GROUP_NAME */
159
```

Figure 1-19

- File Head Comment: Click “Remark” → “File Head Comment” to add a file header comment sample in the header.h file. The shortcut is Ctrl+Alt+K;
- Symbol Comment: Click “Remark” → “Symbol Comment” to add a symbol comment sample in the header.h file. The shortcut is Ctrl+Shift+K;
- brief: Click “Remark” → “brief” to add a brief declaration sample in the header.h file. The shortcut is Ctrl+L;
- brief(After Line): Click “Remark” → “brief(After Line)” to add a brief declaration sample behind the code line in the header.h file. The shortcut is Ctrl+Shift +L;
- Image(Non Chinese name): Click “Remark”→ “Image(Non Chinese name)” to insert a picture and add a picture declaration sample in the header.h file;
- Attention: Click “Remark” → “Attention” to add an attention declaration sample in the header.h file;
- Copyright: Click “Remark” → “Copyright” to add a copyright information declaration sample in the header.h file;
- Example: Click “Remark” → “Example” to add an example codes declaration in the header.h file;
- Include: Click “Remark” → “Include” to add a header file declaration sample in the header.h file;
- See: Click “Remark” → “See” to add a view member declaration sample in the header.h file;
- List: Click “Remark” → “List” to add a list declaration sample in the header.h file;
- Params: Click “Remark” → “Params” to add a parameters declaration sample in the header.h file;
- Ref: Click “Remark” → “Ref” to add a reference declaration sample in the header.h file;
- Mult Ref: Click “Remark” → “Mult Ref” to add a multiple references declaration sample in the header.h file;
- Table: Click “Remark” → “Table” to add an author information declaration sample in the header.h file;

- Table2: Click “Remark” → “Table2” to add an author information comment sample;
- Else: Click “Remark” → “Else” to add a declaration sample about remarks, warning, preconditions and deprecated information comment samples in the header.h file;
- User Define: Generate user-defined comments;
- Remark Settings: Set the path for adding pictures;
- Preview Help Of File : Click or “Remark” → “Preview Help Of File” to show the Remark Preview window;

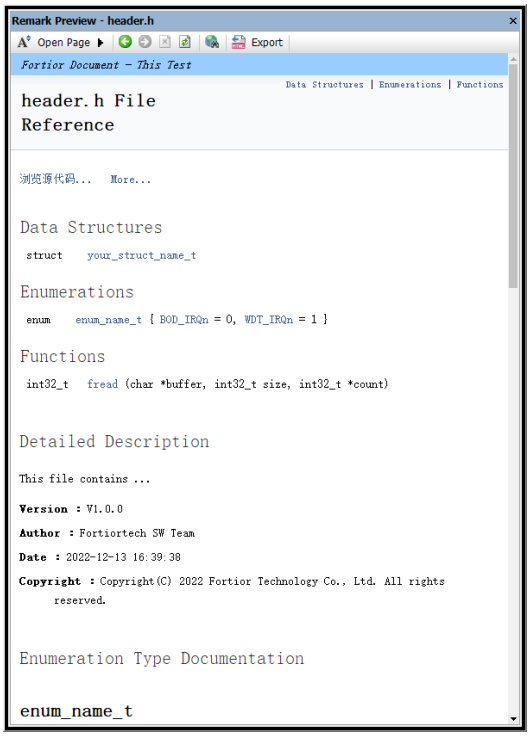


Figure 1-20

- Preview Help Of Project : Click or “Remark” → “Preview Help Of Project” to show Remark Preview window. All header.h files are listed in the window with brief descriptions, as shown in Figure 1-22;

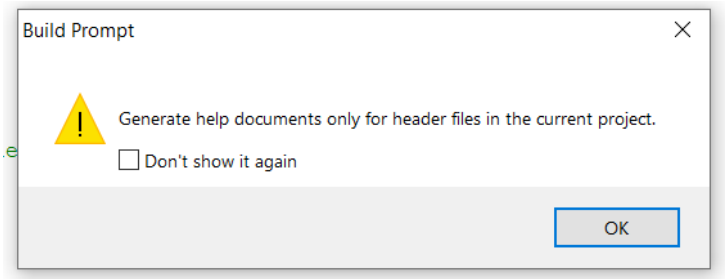


Figure 1-21

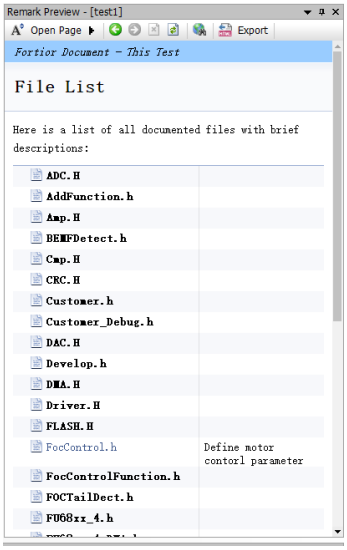


Figure 1-22

- Stop Help Build: Stop generate the Help file.
- Generate.chm  : Click  or “Remark” → “Generate.chm” to generate .chm help files. The Output window displays the compilation process and results of the generated help files, as shown in Figure 1-23.

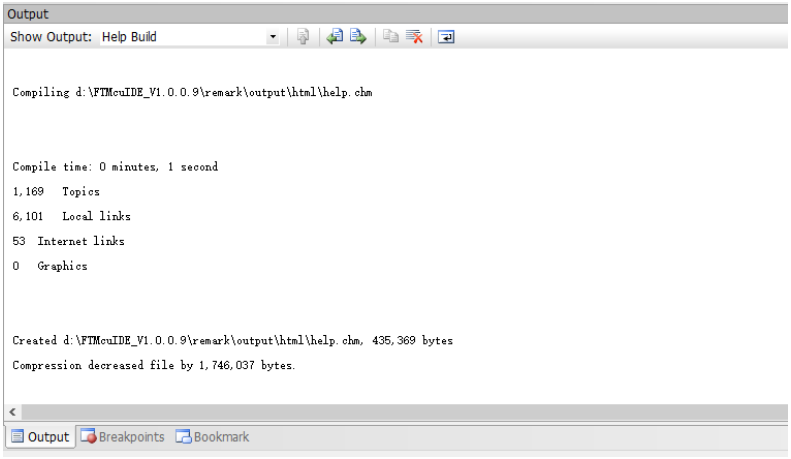


Figure 1-23

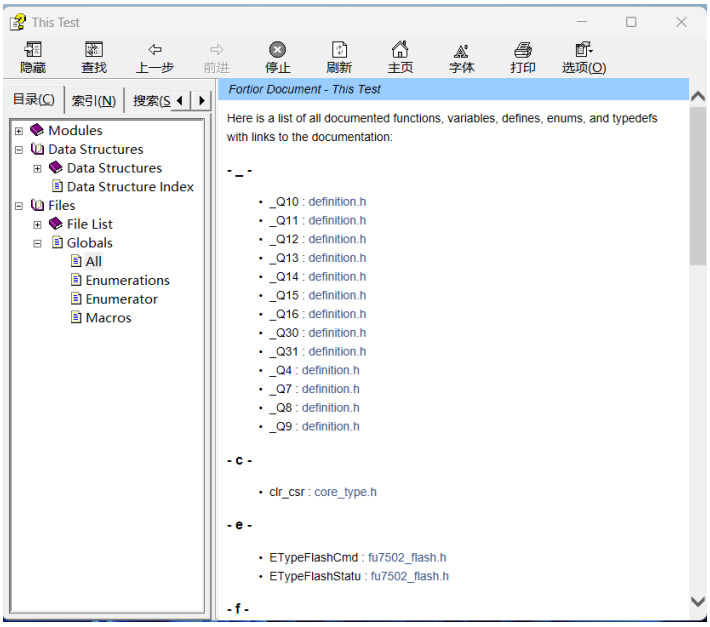


Figure 1-24

1.1.8 Tools Menu

Tools: Click to show the Options window. You can set the interface style, text font size, colors and so on.

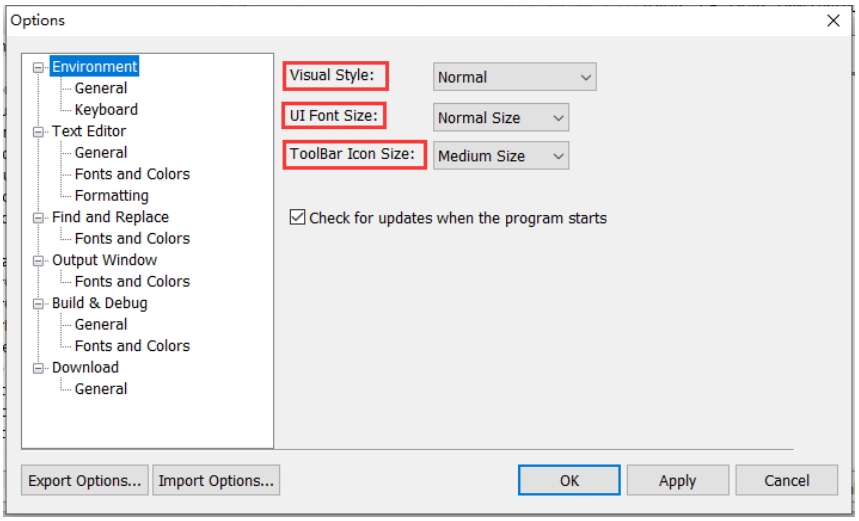
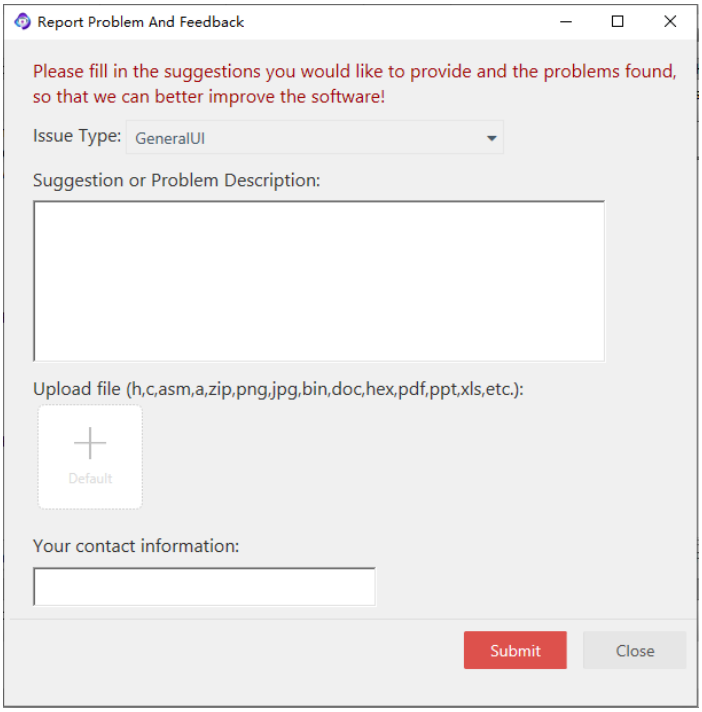


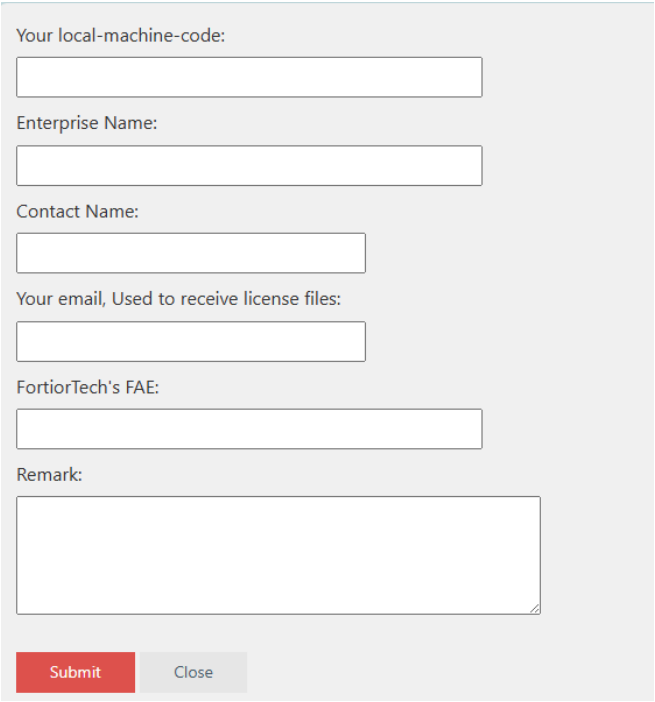
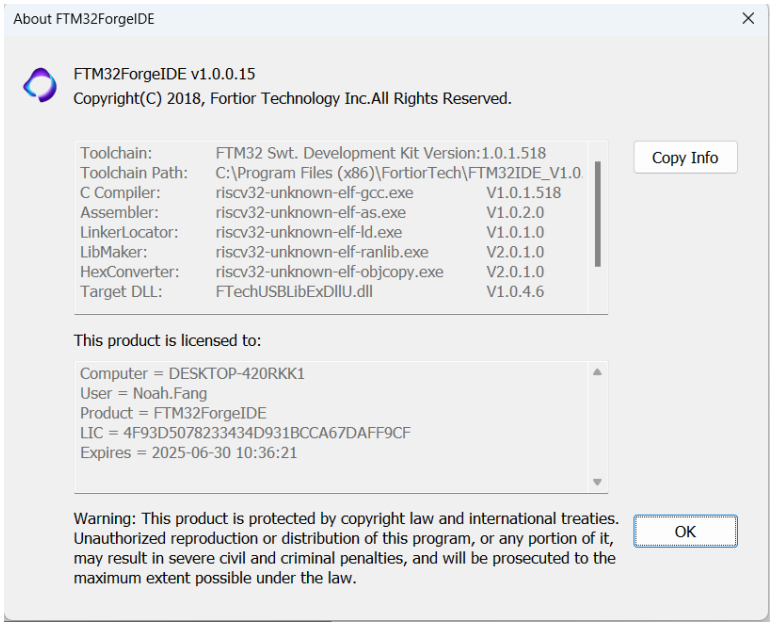
Figure 1-25

1.1.9 Window Menu

Window Menu	Description
New Window	Open a window to display the file.
Close All	Close all opening files.
Rearrange Windows	Rearrange the windows and reset them to the default settings.
Windows	Switch among the open files.

1.1.10 Help Menu

Help Menu	Description
Show Start Page	You create a new solution, open an existing solution and quickly open a solution in the recently used solution list.
Fortiortech on The Web	Visit Fortior Technology website.
User Manual (English)	Open the user manual in English
User Manual (Simplified Chinese)	Open the user manual in Chinese
Programming Guide	There're some manual: Porting Guide(Simplified Chinese), FTM32 C Programming Guide (Simplified Chinese) and FAQ.
Report Problem& Feedback Message…	Display the feedback information interface, where you can fill in the suggestions or the problems found during use, and attach a document for explanation.  Figure 1-26
Check for Updates	Check the latest patch or service pack for IDE update.
Apply for Licence	Apply for a software licence. The following window pops up after you click this button. Fill in the information as required and click Submit to submit the application. If the application is approved, the Licence document is sent to your email address.

Help Menu	Description
	 Figure 1-27
Import Licence	Import a software license
About IDE	The information about FTM32ForgeIDE.  Figure 1-28

1.2 Project Settings

1.2.1 Settings

Click “Project” → “Settings” to set the project. The shortcut key is Alt+F. All common settings can be reset to default values via the Resent button.

1. General: contains the directory and name of the output file and the selection of the makefile file. Among them, “Generate and overwrite makefile” automatically generates the makefile for the selected software; “Use external makefile” selects the makefile set by the programmer, as shown in Figure 1-29.
2. Target: As shown in Figure 1-30, the choice of chip type, please refer to the “FTM32 C Programming Guide (Simplified Chinese)” for specific descriptions.

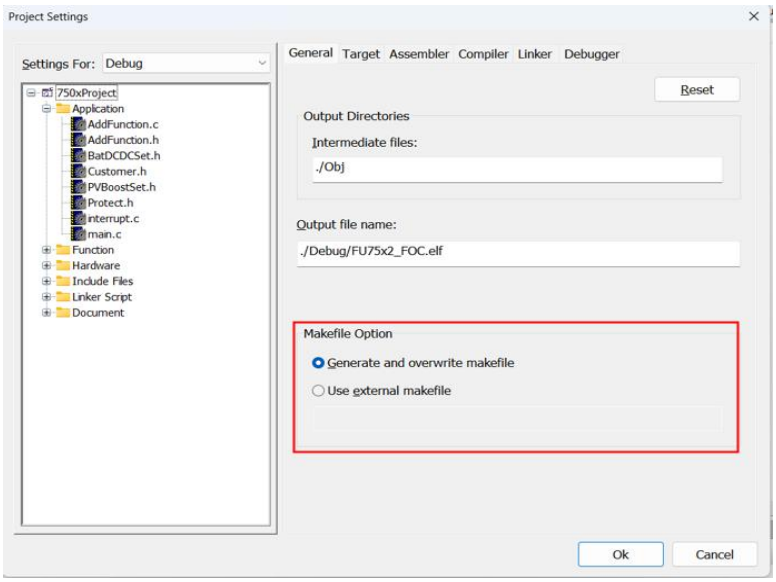


Figure 1-29

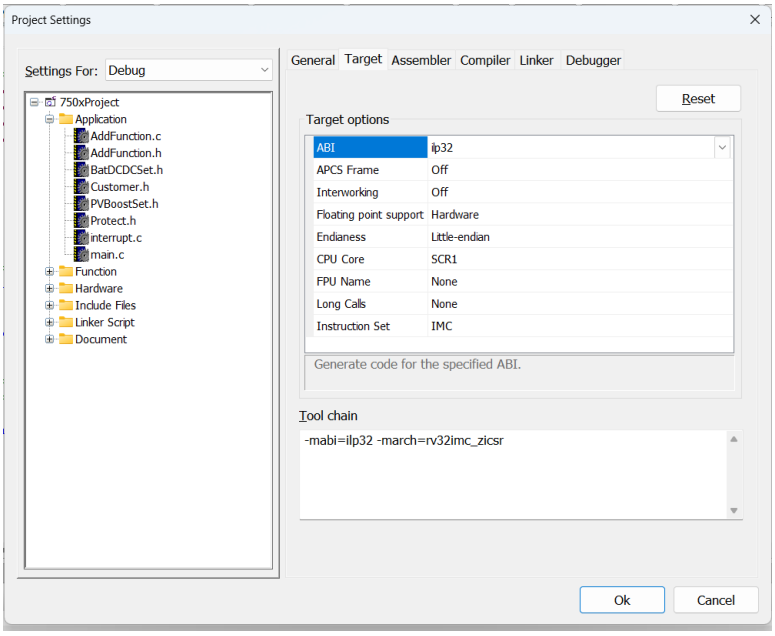


Figure 1-30

3. Assembler:
   - Define Symbol: Define macro in assembly code;
   - Undefine Symbol: Cancel macro in assembly code;

- Include path: Add the address of the header file, as shown in Figure 1-31 below;
- Warning as error: The default is Off, which turns off the operation that treats warnings as errors;
- Disable Warning: Generally not set;
- Disable All Warning: The default is Off;
- Auto convert to path relative to project path: It is generally recommended to check this option.

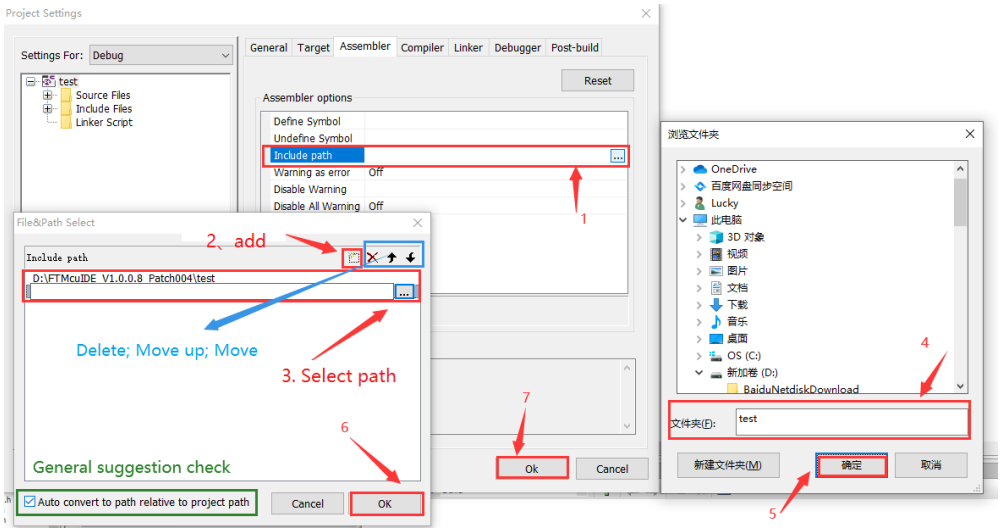


Figure 1-31

4. Compiler: Category has two categories: General and Debug, as shown in Figure 1-32 below;

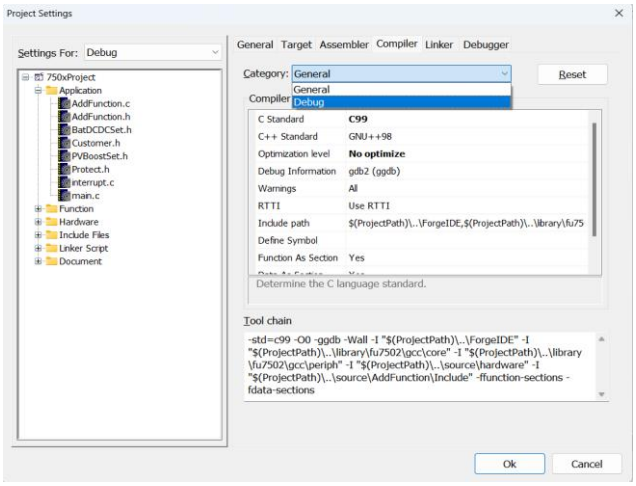


Figure 1-32

- C Standard: C language standard, select C99 here, or other standards;
- Define Symbol: Define macros in C language;
- Undefine Symbol: Cancel macros in C language;
- Include path: Add the address of the header file;
- Warning as error: The default is Off;
- Disable Warning: Generally not set;

- Disable All Warning: The default is Off;
- Signed Char: The default value is On, indicating that the defined variable is a signed variable. For example, char char8 is a signed character;
- Reentrant intlong: The default is On;
- Reentrant float: The default is On;
- Optimize for code: code optimization level, with the following five levels available. Default is O0, as shown in Figure 1-33;

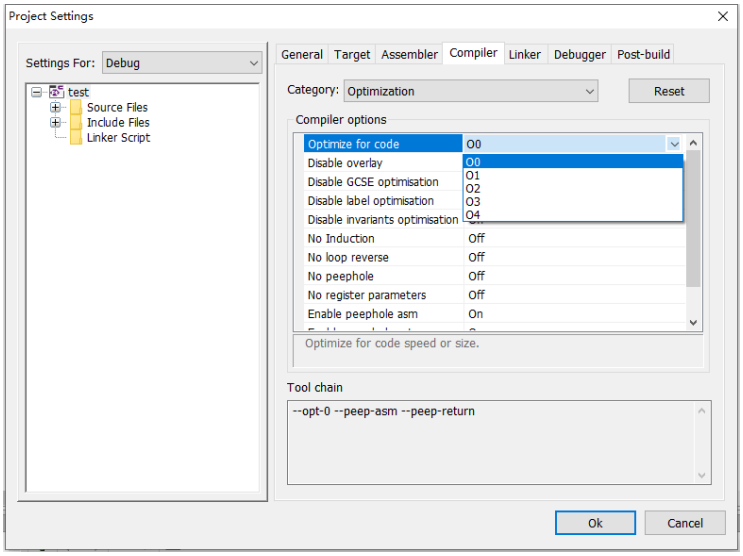


Figure 1-33

5. Linker:
   - Linker Script: Linker files;
   - Library Path: Defined library path;
   - Additional Libs: Extra needed library.

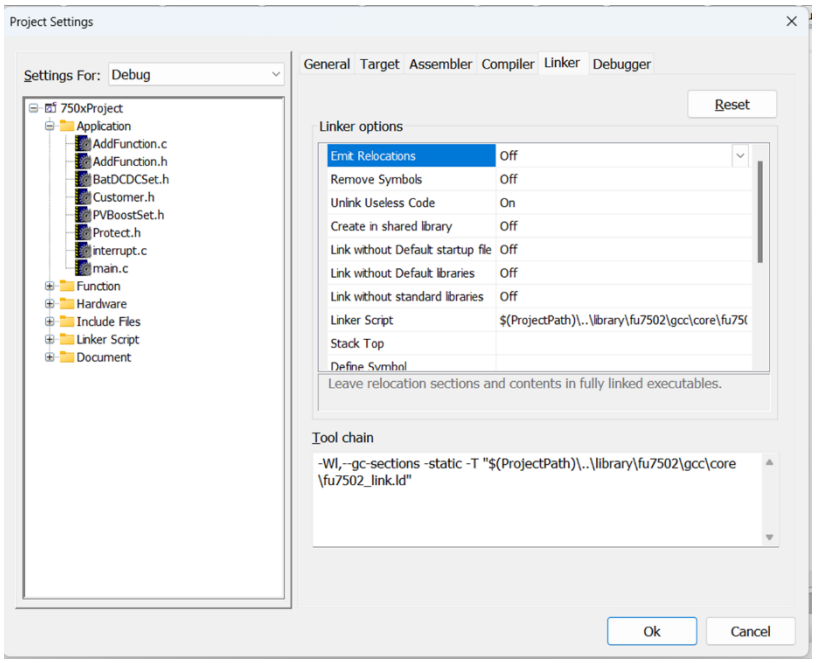


Figure 1-34

6. Debugger:

- Run to main function: Indicates that the register initialization operation is completed from the main function during debugging;
- Run to startup code: Indicates that the debugging starts from the startup code. At this time, the register is not initialized and the last operation value is retained;

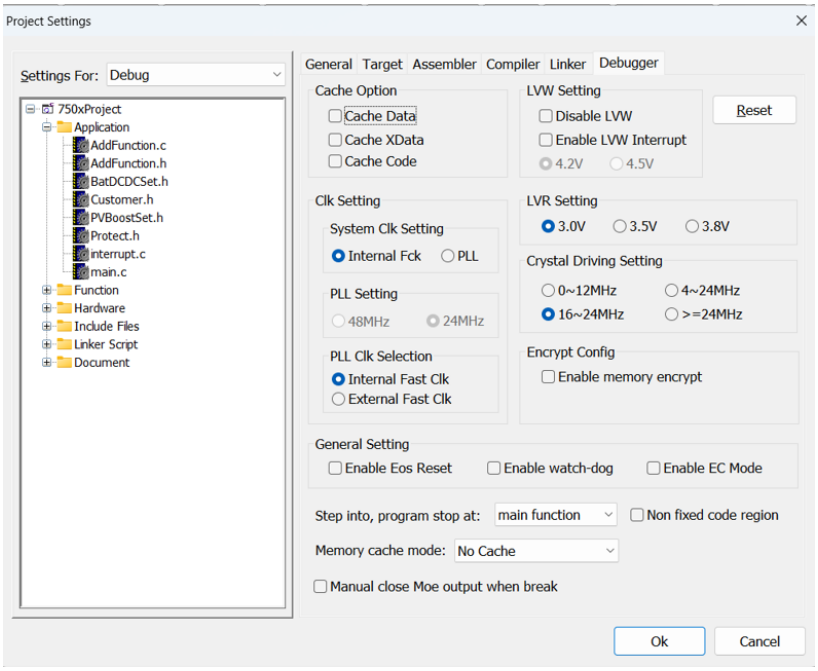


Figure 1-35

1.3 Window Introduction

In the FTM32Forge IDE, you can dock windows at specific locations and manage them by using tag groups. You can make a window in the “suspended” state, that is, make it always in the upper layer of other windows. Changing the size and position of the “floating window”, does not affect other windows.

1.3.1 Solution Browser

Solution Browser: Displays all project files in a tree structure to help you manage project files, as shown in Figure 1-36.

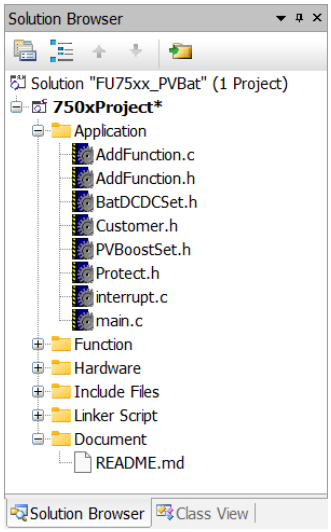


Figure 1-36

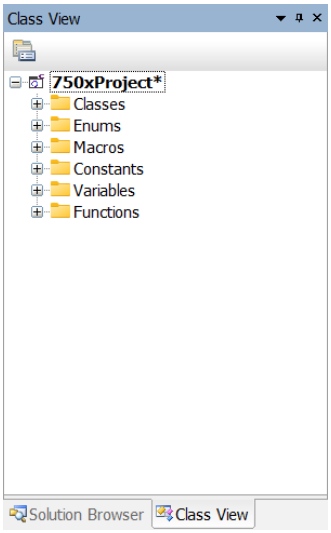


Figure 1-37

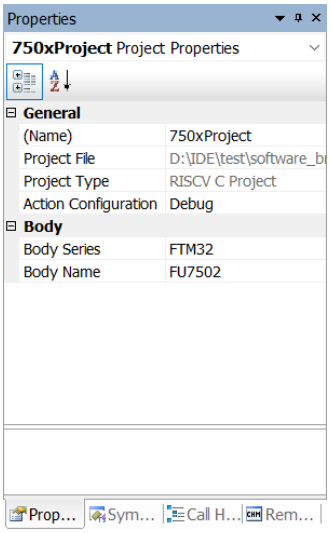


Figure 1-38

In the Solution Browser, you can change the order of folders or files. Operations: Click the folder or the file to activate the up and down arrow keys in the Solution Browser, and change the folder or file order by clicking the arrow key.

1.3.2 Class View

Class View: The tree structure shows the program-defined structure, macro definitions, global variables, etc., as shown in Figure 1-37.

1.3.3 Properties View

Properties View: Display the General, Body and Content of the selected project or file.,

- General: as shown in Figure 1-38
  - Name: Project Name, File Name;
  - Type: Project Type, File Type;
  - Absolute Path: Project File, Full Path
  - Relative Path;
  - Action Configuration: Debug;
- Content: as shown in Figure 1-38

- File Size;
- Modify Time;

■ Body: as shown in Figure 1-39

- Body Series: Riscv;
- Body Name: If you want to change the Body of the created project, you can click on the Body Name to change it, and the "SupportBody" displays the supported chip sub models, as shown in Figure 1-40.

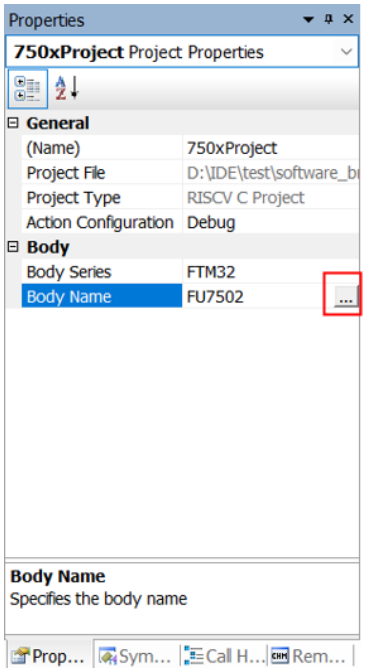


Figure 1-39

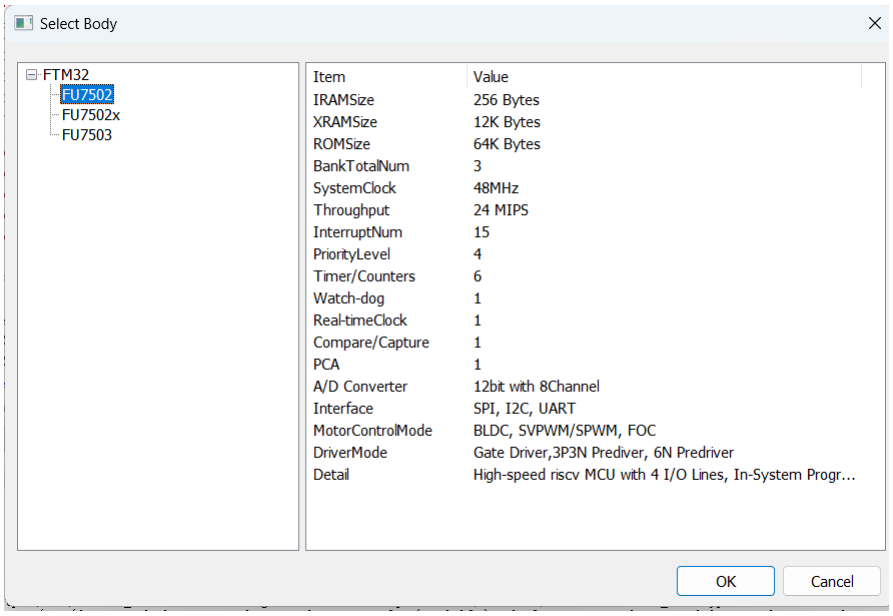


Figure 1-40

1.3.4 Breakpoints Window

Breakpoints Window: Displays the location of all breakpoints set by programmers in the program, including the file name, address, and number of lines, as shown in Figure 1-41 below. The shortcut key is Alt+F9.

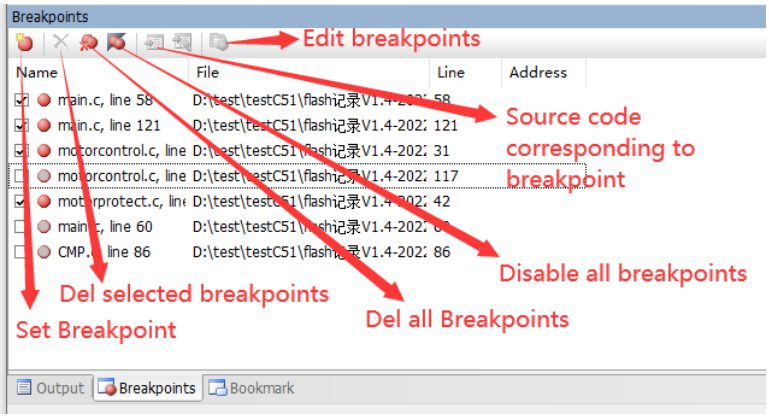


Figure 1-41

In static mode, You can more than 4 breakpoints; During dynamic debugging, the maximum number of valid breakpoints is 4, as shown in Figure 1-42 and Figure 1-43.

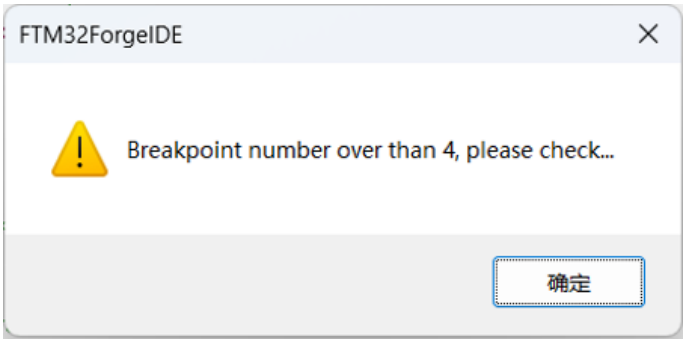


Figure 1-42

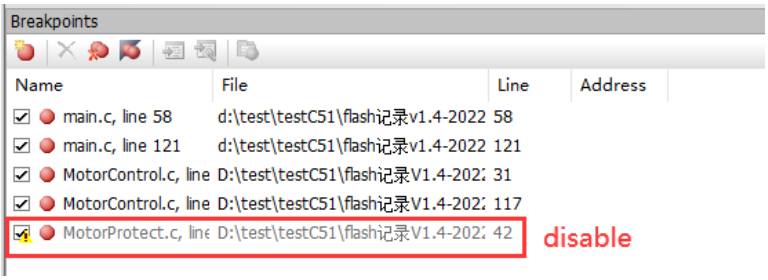


Figure 1-43

 : To set a breakpoint, enter the file name and number of lines to set the breakpoint, as shown in Figure 1-44. The setting result is shown in Figure 1-45, and the breakpoints window shows the new breakpoint just set.

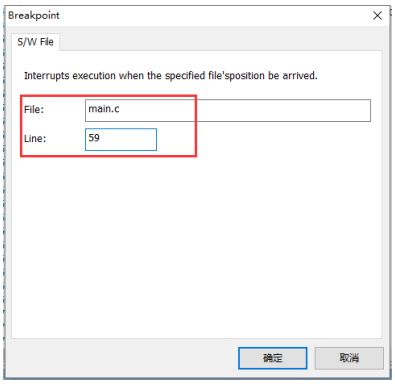


Figure 1-44

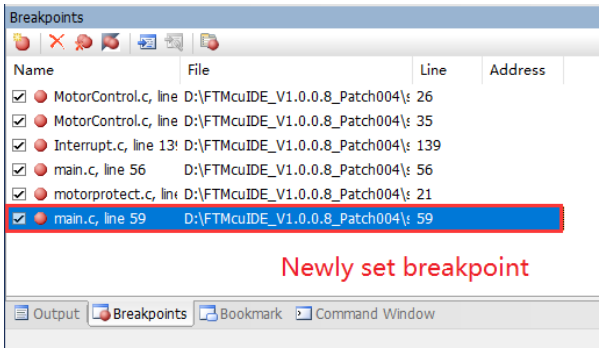


Figure 1-45

: Deletes the selected breakpoint.

: Delete all breakpoints.

: Make all the breakpoints set disable, that is, do not tick all breakpoints, as shown in Figure 1-46. If a breakpoint is checked, the breakpoint is enabled.

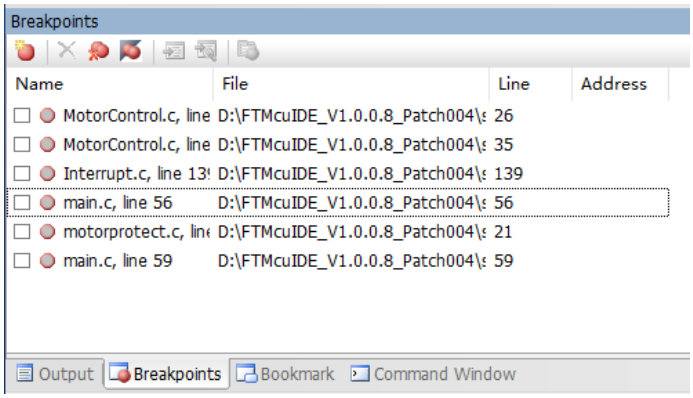


Figure 1-46

: The source code corresponding to the selected breakpoint, as shown in Figure 1-47.

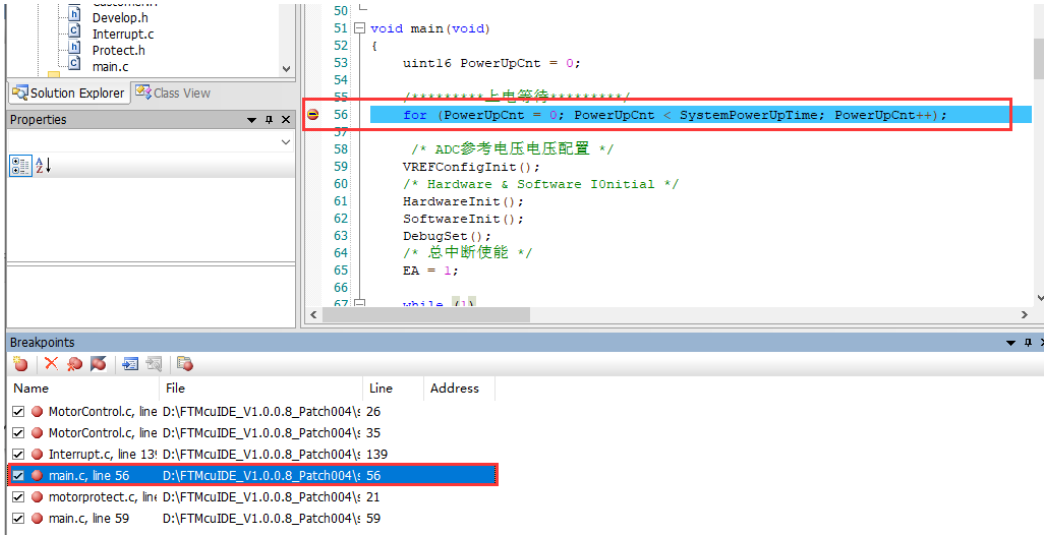


Figure 1-47

: Edit the selected breakpoint, for example, select the breakpoint main.c line56, click this option to

display the settings page shown in Figure 1-48, you can modify or delete the breakpoint.

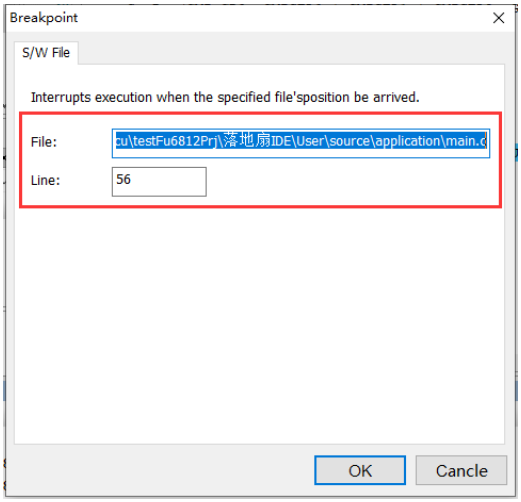


Figure 1-48

1.3.5 Disassembler Window

Disassembler Window: Displays the address, opcode, operand, assembly instructions, and corresponding source code, as shown in Figure 1-49. An assembly instruction is composed of an instruction mnemonic (abbreviation of the name of the instruction) and symbols representing variables, registers, and constants, that is, each assembly instruction is represented by an assembly instruction mnemonic followed by one or more symbols.

The address is the memory address where each assembly instruction is located, the opcode is the hexadecimal code corresponding to the mnemonic of the assembly instruction, such as 7F, and the operand is the operand of the assembly instruction, such as 3A corresponding to #0x3a.

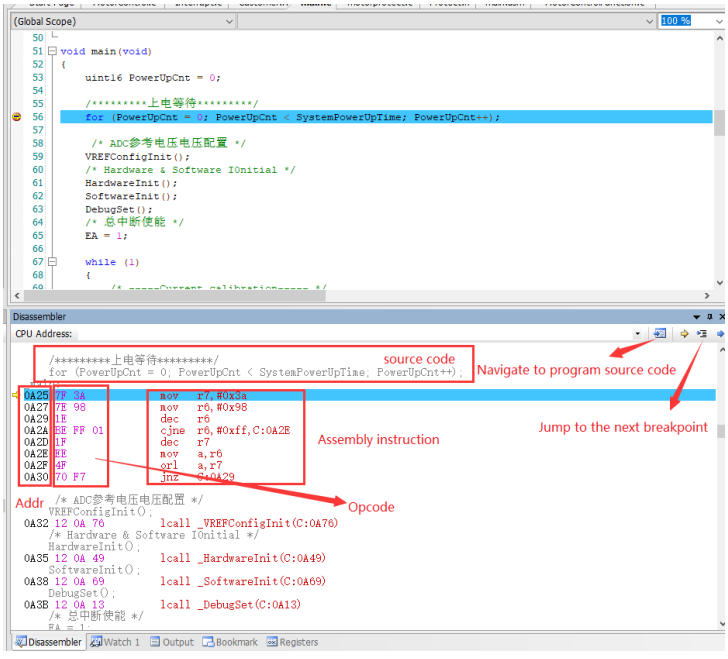


Figure 1-49

When running the program, click debug, and the assembly instructions run in sequence with the source code running line by line. The breakpoint set in the source code window also exists at the corresponding assembly instruction; On the contrary, if the breakpoint set in the source code window or disassembly window is canceled, the breakpoint corresponding to the disassembly window or source code window is also canceled.

 : Click this button to locate the source code corresponding to the running assembly instruction.

 : Click this button to run the program directly to the next breakpoint.

1.3.6 Registers Window

Registers Window: In the register window, three registers “All”, “General Register” and “GPIO” are displayed. “All” contains all register types, and you can select according to your own needs, as shown in Figure 1-50.

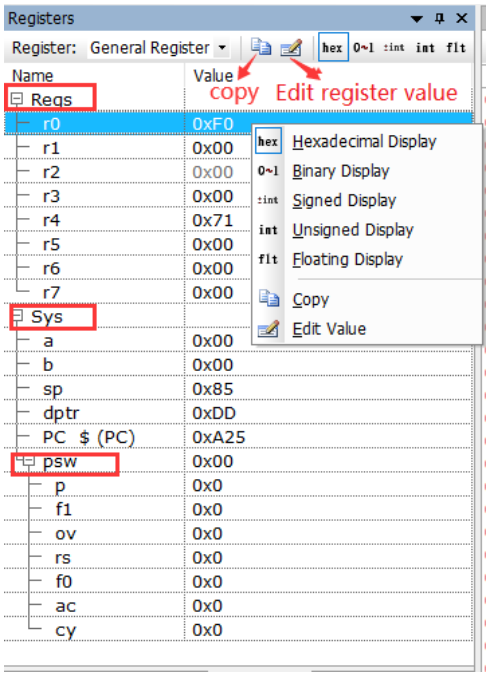


Figure 1-50

Regs: There is a set of current working registers r0~r7 that display the values in r0~r7 when the current assembly code runs.

Sys register: Includes: “a” accumulator; “b”; “sp” stack pointer, which is used to point to the top of the current stack; “dptr” data pointer, which is used as the address register of off-chip RAM addressing; “PC” program pointer, which indicates where the current program is running, pointing to the code area.

Psw program status word: including “p”, “f1”, “ov”, “rs”, “f0”, “ac”, “cy”.

“p” parity flag bit: every time an assembly instruction is executed, the chip sets or resets “p” according to the parity number of 1 in “a”. Odd is 1 and even is 0.

“Ov” is the overflow flag. When adding and subtracting signed numbers, the hardware sets or resets it; When OV=1, it means that a number has exceeded the range that the accumulator represents a signed number

in the form of complement, that is, it exceeds the range of - 128~+127.

"rs" selects bits for the working register group, which is set or cleared during programming.

"f0" is a software flag bit, a user-defined status flag, which is set or reset by software.

"ac" is the auxiliary carry flag bit. When performing an 8-bit addition operation, if the highest bit (D3) of the lower half byte has a carry, then "ac"=1, otherwise "ac"=0; When performing 8-bit subtraction operation, if D3 has borrow, "ac"=1, otherwise "ac"=0.

"cy" is the carry flag bit. When using operations such as addition, subtraction, multiplication and division, left shift, and right shift, "cy" is affected. When the highest bit (D7) of the data is added to generate carry, "cy"=1, otherwise "cy"=0; When performing such a subtraction operation, if the operation result has a debit, "cy"=1, otherwise "cy"=0.

As shown in Figure 1-59, click  to copy the selected register and register value; Click the  "Edit Value" option to edit the selected register value; Click  to display the register value in hexadecimal format; Click  to display the register value in binary form; Click to display the register value in the form of signed integer; Click  to display the register value in the form of unsigned integer; Click  to display the register value as a single-precision floating-point number.

1.3.7 Memory Window

Memory Window: In the memory window, you can view the data in different memory spaces such as Code, IRam, XRam, CCFG, and enter the queried address in the CPU Address to reach the address, as shown in Figure 1-51.

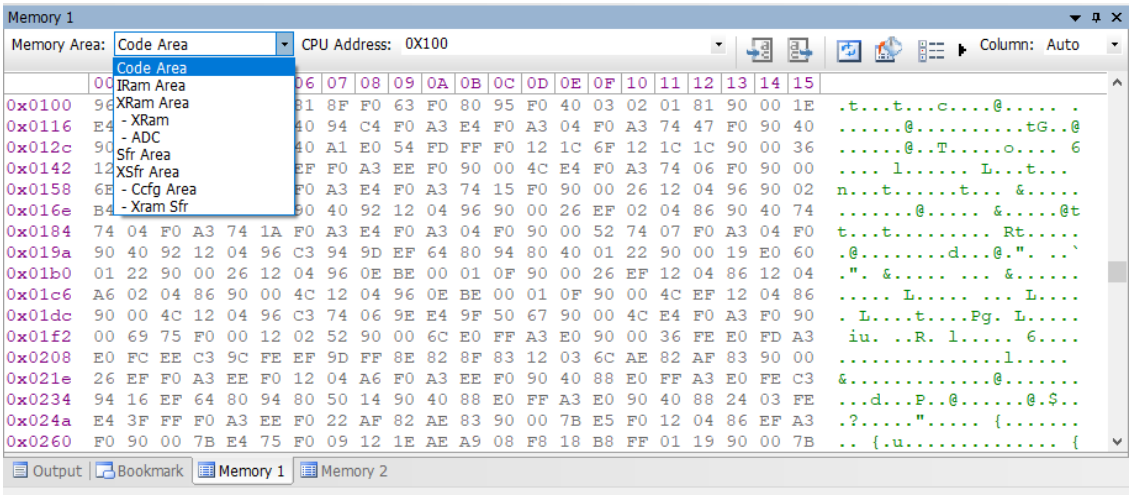


Figure 1-51

As shown in Figure 1-52, enter 0x100 in the CPU Address and press Enter. The memory window reaches the address of 0x100 to facilitate the programmer to view the address data.

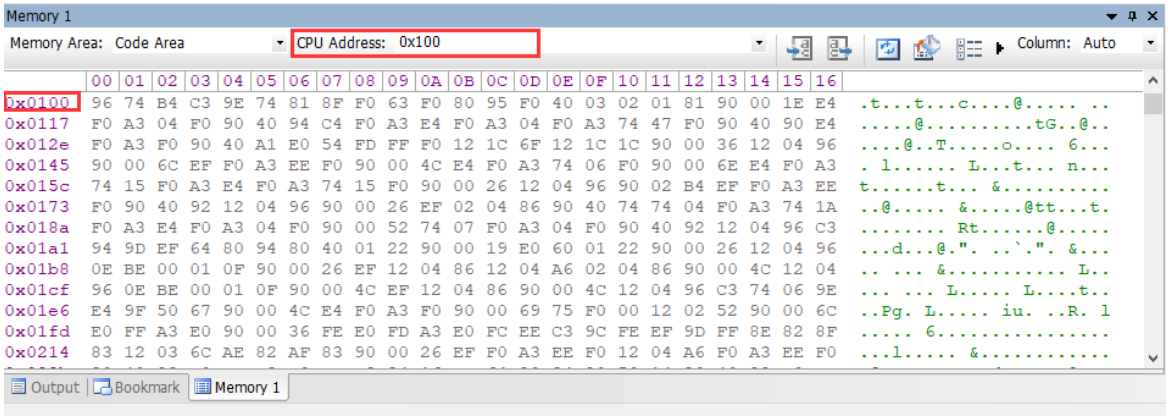


Figure 1-52

Similarly, entering the decimal number 100 (0x64) in CPU Address reaches its corresponding hexadecimal address, as shown in Figure 1-53.

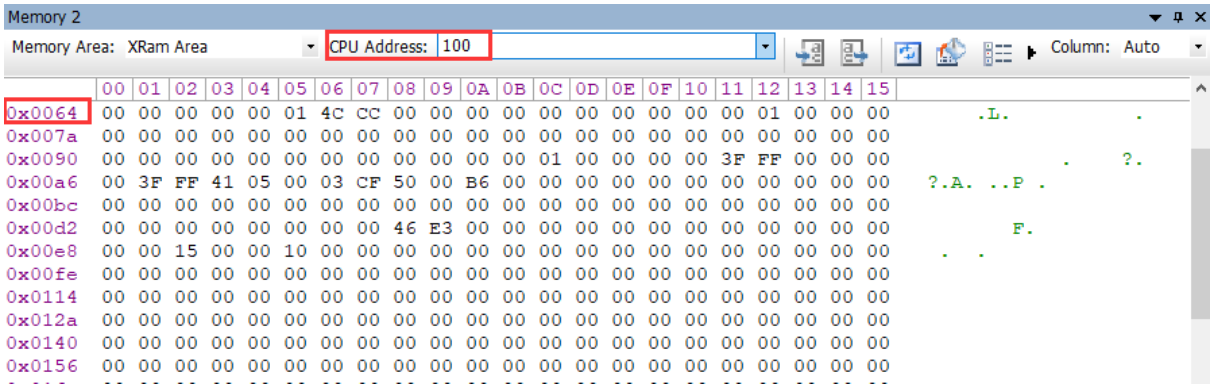


Figure 1-53

Export Memory Data: Export the data of the currently selected memory space, as shown in Figure 1-54First, enter the start and end address of the exported memory space, then select the address of the exported file, and then select whether to open the file after export. If selected, open the file after the end of the exported file, and finally click OK to finish exporting the memory data.

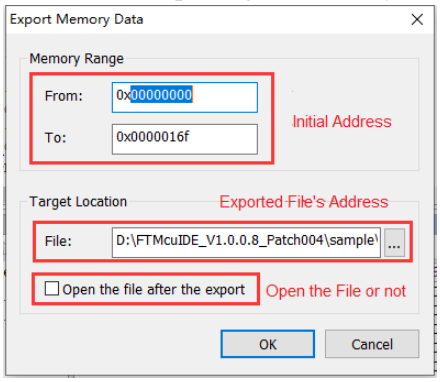


Figure 1-54

Import Memory Data: Import the memory data file, as shown in Figure 1-55. First, enter the start and end address of the imported memory space, then select the imported memory file, and click OK to import the memory file data into the set start and end address range.

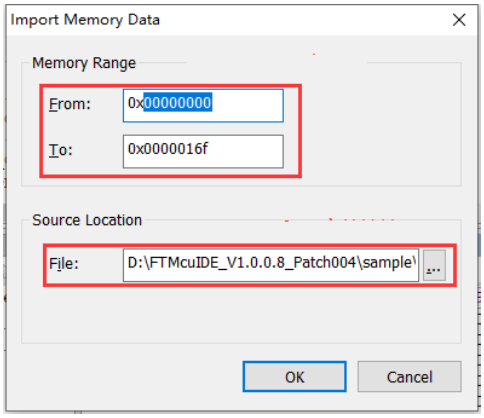


Figure 1-55

: Refresh Memory.

: Fill Pattern, as shown in Figure 1-56.

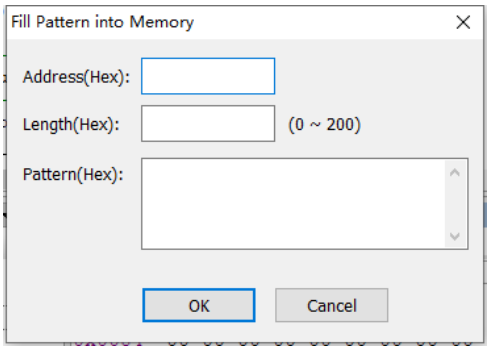


Figure 1-56

: Set the style of memory window, as shown in Figure 1-57, including a column composed of several bytes, data format, description and column layout format. The data format is displayed in five ways: hexadecimal, binary, signed integer, unsigned integer and single-precision floating-point type. For details, please see Registers Window.

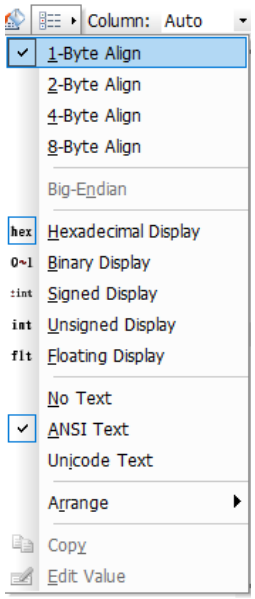


Figure 1-57

When 1-Byte Align is selected, the memory window is shown in Figure 1-58; When 2-Byte Align is selected, the memory window is shown in Figure 1-59. Similarly, you can know the style of the window when other selections are made.

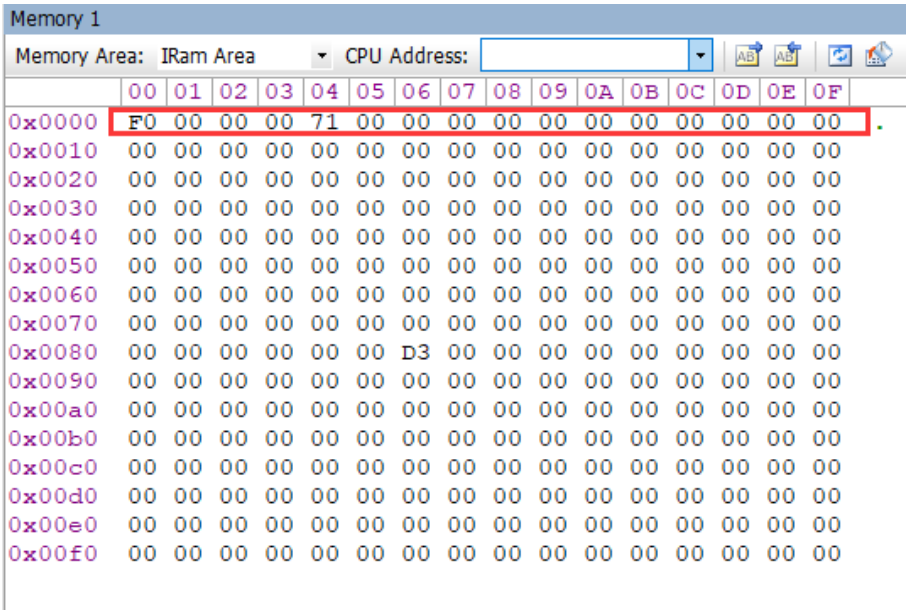


Figure 1-58

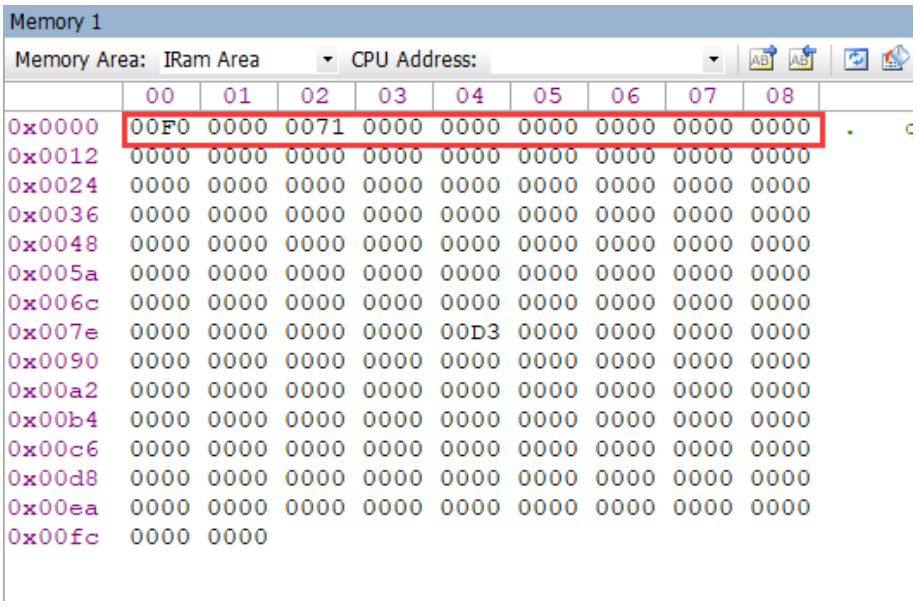


Figure 1-59

When No Text is selected, the memory window is displayed in hexadecimal 16-bit data format without description, as shown in Figure 1-60.

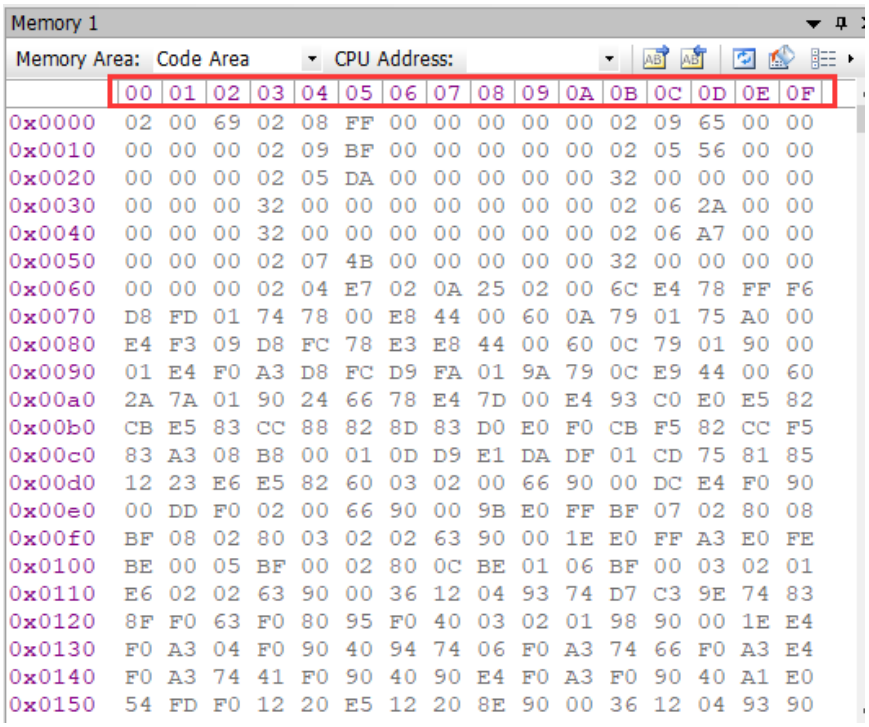


Figure 1-60

When ANSI Text is selected, the memory window is displayed in hexadecimal 16-bit data format with corresponding English description beside it, as shown in Figure 1-61.

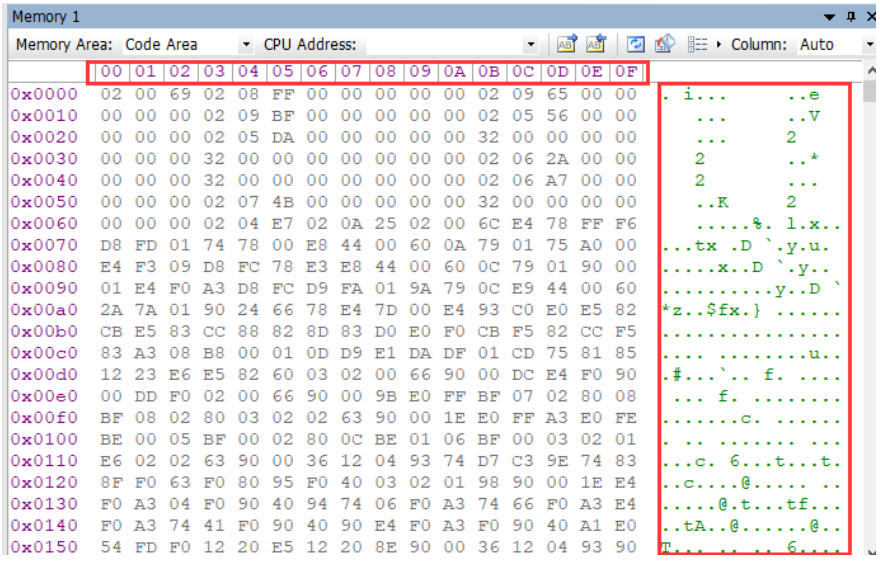


Figure 1-61

When Unicode Text is selected, the memory window is displayed in hexadecimal 16-bit data format with corresponding text description beside it, as shown in Figure 1-62.

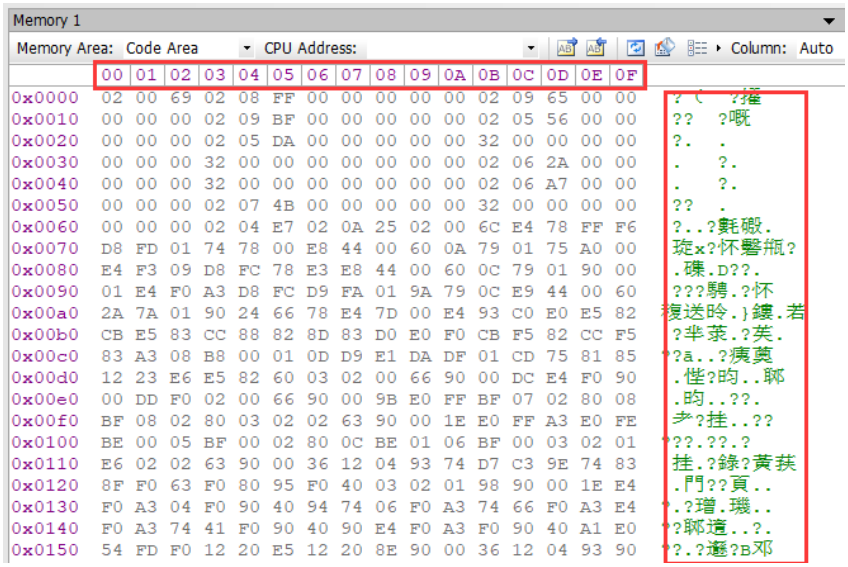


Figure 1-62

Column: Auto : Set the number of rows displayed in the memory window data. The options are Auto, 1, 2, 4, 8, 16, 32 and 64. When Auto is selected, the memory window adjusts the displayed data according to the size of the window; When 1 is selected, it indicates that the memory window data is displayed in 1 column, as shown in Figure 1-63; Similarly, other selections indicate that the memory window data is displayed in the selected number of columns.

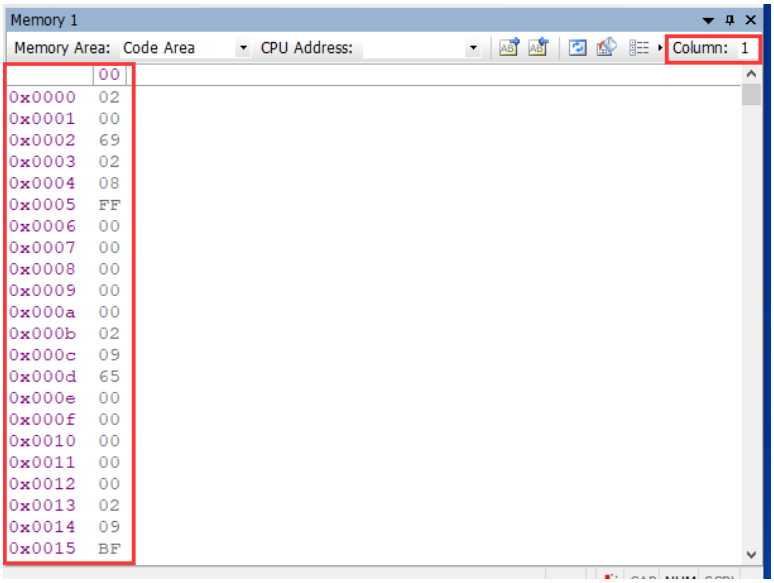


Figure 1-63

1.3.8 Watch Window

The FTM32ForgeIDE allows programmers to view the values of variables or expressions in source code, enabling you to track the changes of their values in real time during operation. You can view variables in many ways. For example, in the source code window, place the mouse over the variable you want to view, or view it by opening the Watch window.

Observe the change of variable value when the program is running, and you can enter the variable and query the variable value by yourself. The Watch window has the option of displaying variable address and variable type. When the variable address is selected, it is shown in Figure 1-64. When the display variable type is selected, as shown in Figure 1-65.

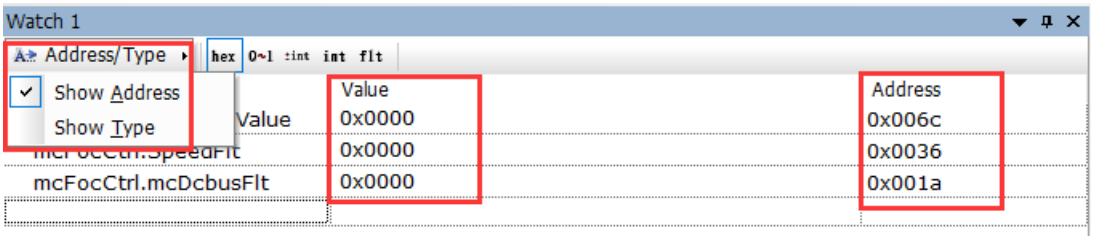


Figure 1-64

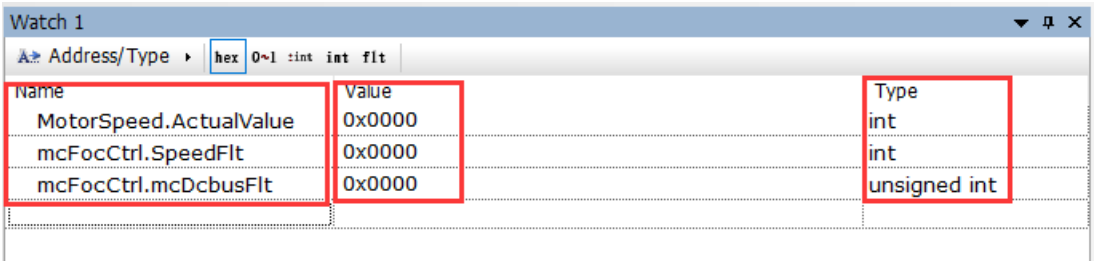


Figure 1-65

When running the program, set a monitoring point on the variable EA: click the dotted rectangle in the Watch window. When the input area appears, type EA and press Enter. The window immediately displays the current EA value 0x0 and type sbit, as shown in Figure 1-66. Continue to run the program. When the variable value monitored in the Watch window changes, the color of the variable value changes to red to indicate the change. As shown in Figure 1-67, the value of EA changes to 0x1. If you want to delete a variable in the Watch window, select it, and then right-click to select Delete.

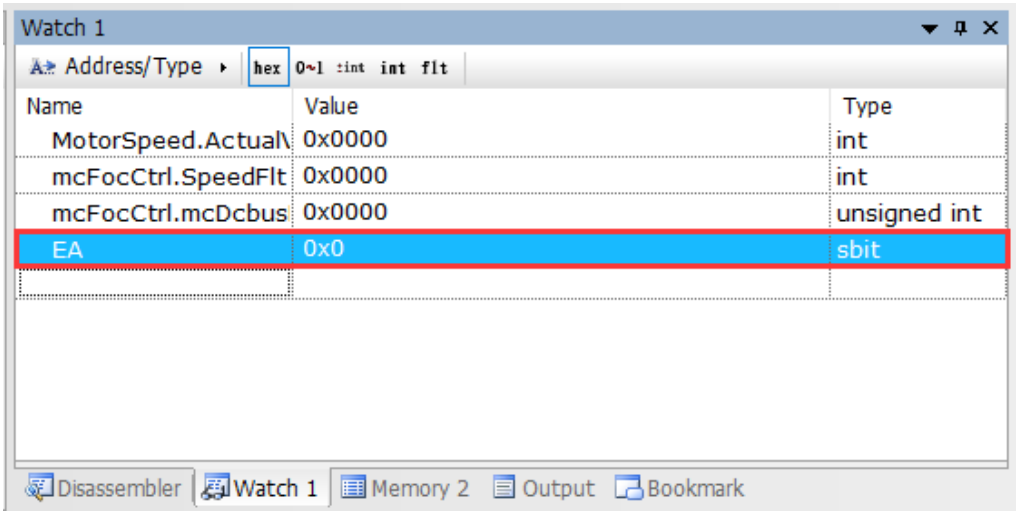


Figure 1-66

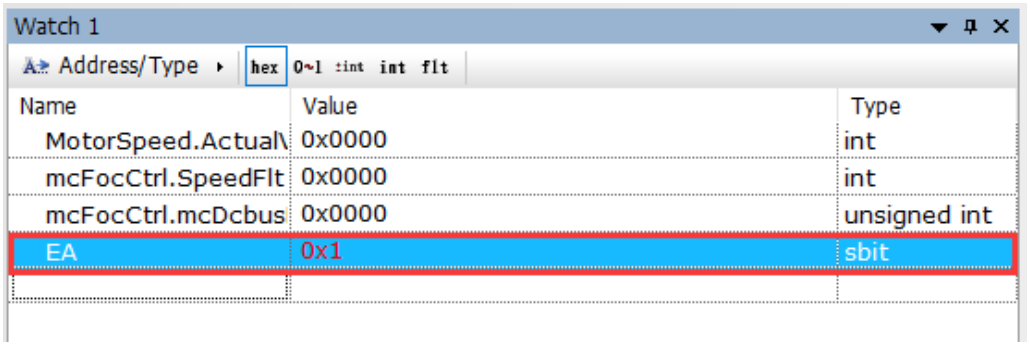


Figure 1-67

Variable values are displayed in five ways: hexadecimal, binary, signed integer, unsigned integer and single-precision floating-point type. For details, see 1.3.6 Registers Window.

1.3.9 Command Window

Command Window: Support command line input.

1.3.10 Call Hierarchy Window

Call Hierarchy window displays member names in the tree view pane, and lists call hierarchy of a selected function or method, as shown in Figure 1-68. You can select the function name to view its tree structure and understand call hierarchy.

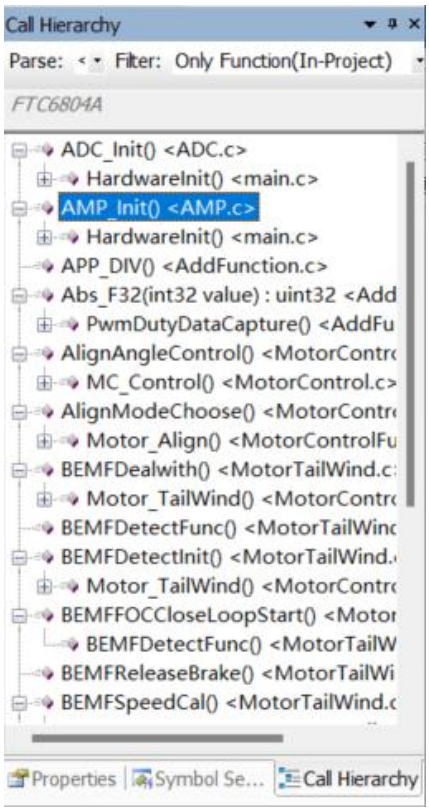


Figure 1-68

2 Create New Application

2.1 Create A New Solution

Select "File" →"New" →"Blank Solution" in the upper left corner, as shown in Figure 2-1.

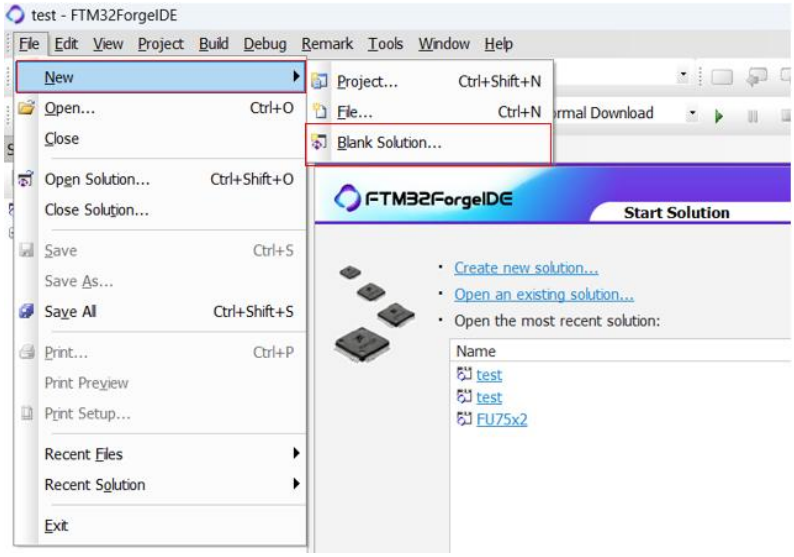


Figure 2-1

Select "Blank Solution", enter solution name, and select storage path of the solution from Location, as shown in Figure 2-2.

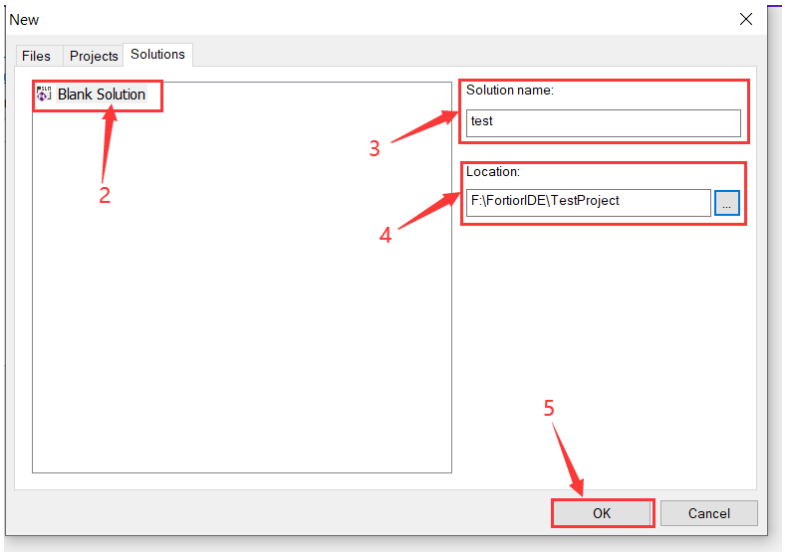


Figure 2-2

Click "OK". The creation result is shown in Figure 2-3. You can create or add projects and files in Solution "test".

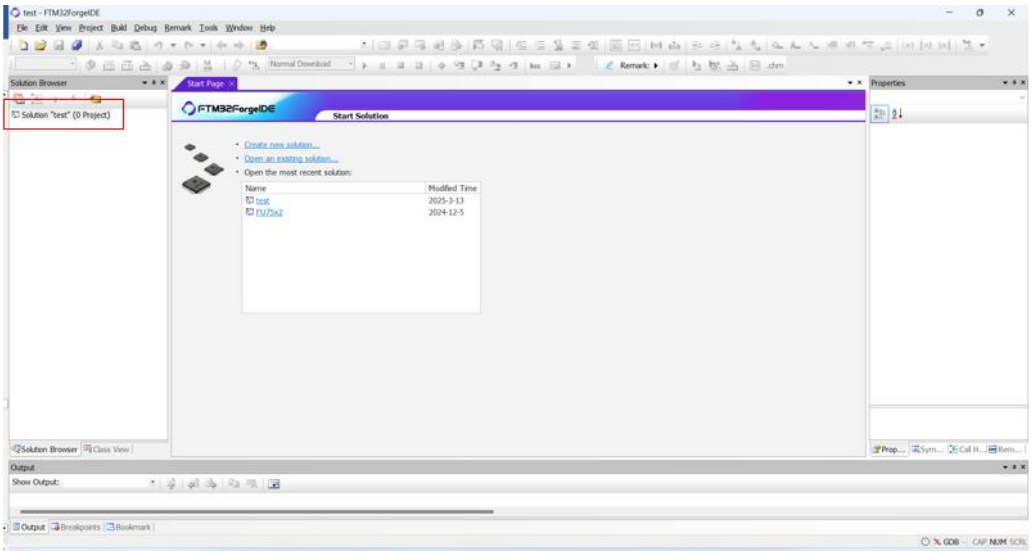


Figure 2-3

2.2 Create A New Project

Step 1: Select "File" →"New" →"Project" in the upper left corner, as shown in Figure 2-4.

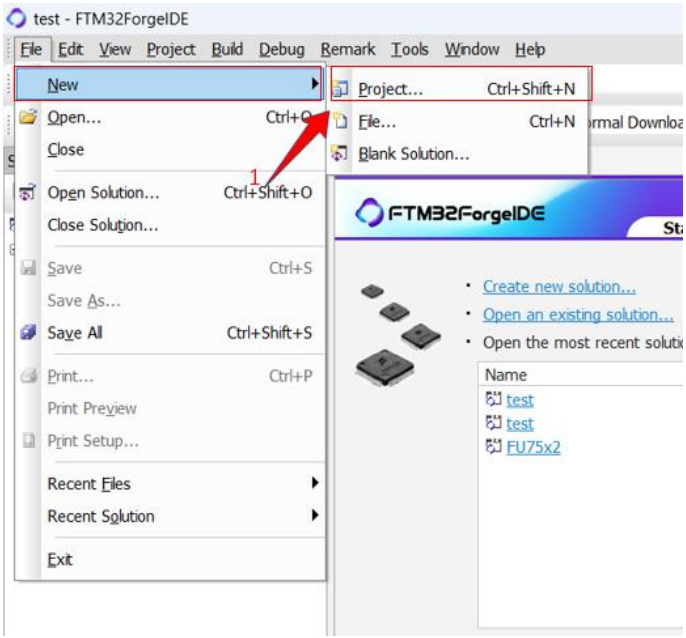


Figure 2-4

Step 2: Select the project type, including Riscv Project, Library Project, Assembly Project and Riscv Simple Project, as shown in Figure 2-5.

Library Project generates a ". lib" library file. Assembly Project generates a ". hex" file. Riscv Simple Project is a complete C project. Simple Project is a simplified version of C project.

Step 3: Enter the project name (for example, "test" or "test.c"). If you do not enter a suffix, the system

defaults the new project to be a C source project with the suffix ".c";

Step 4: Select storage path of the new project;

Step 5: Select "Create a new solution" or "Add to the current solution". The former runs the project in a newly created solution, and the latter adds the project to an existing solution;

Step 6: Click "OK" to select the chip type.

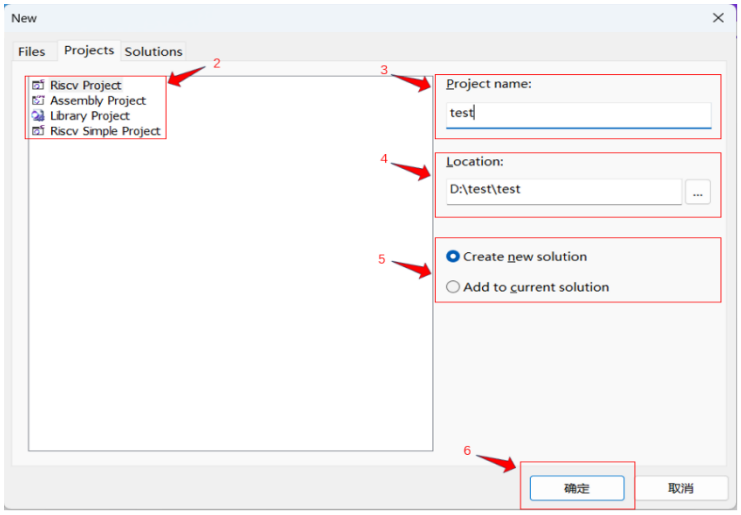


Figure 2-5

Select the chip model. As shown in Figure 2-6, the chip information is displayed on the right.

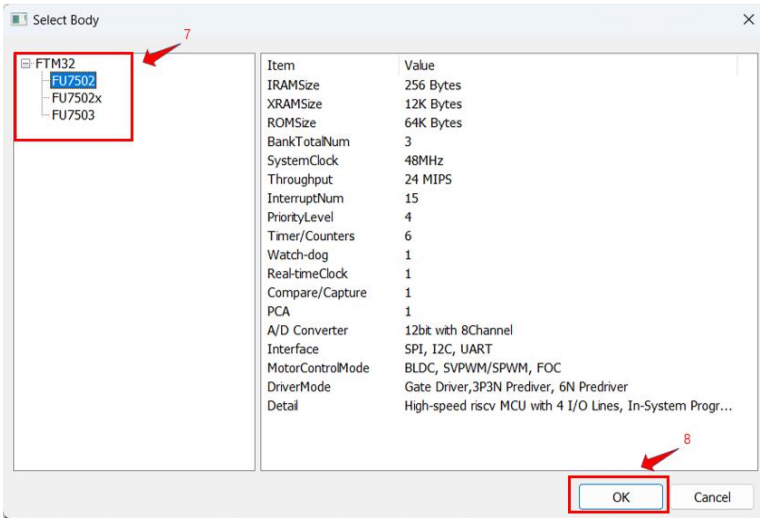


Figure 2-6

Step 7: Click "Finish". The creation result is shown in following figure. It contains the source file "main.c" and header file of the selected chip. You can write program code into "main.c" file.

See Figure 2-7 if Riscv Project and Library Project are created.

See Figure 2-8 if Assembly Project and Assembly Project are created.

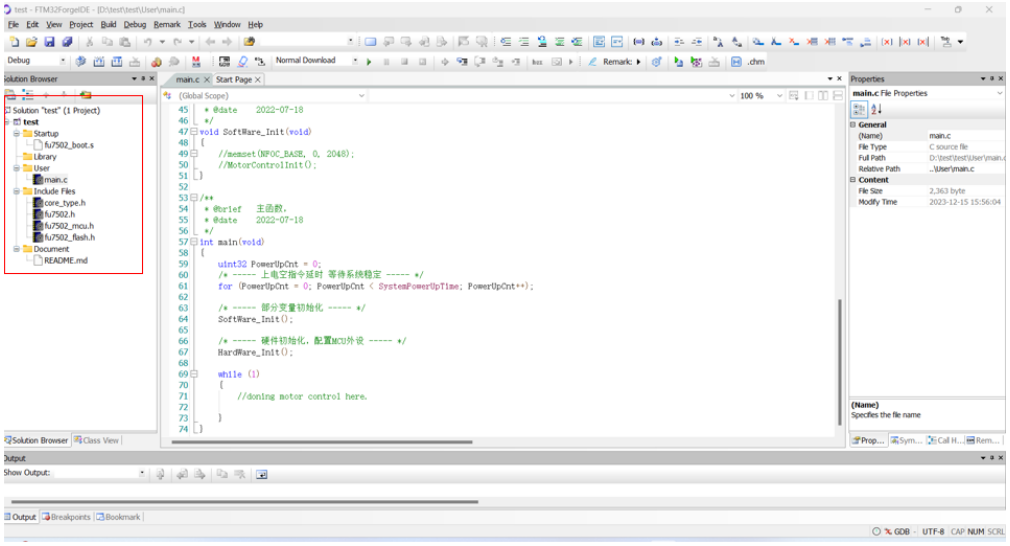


Figure 2-7

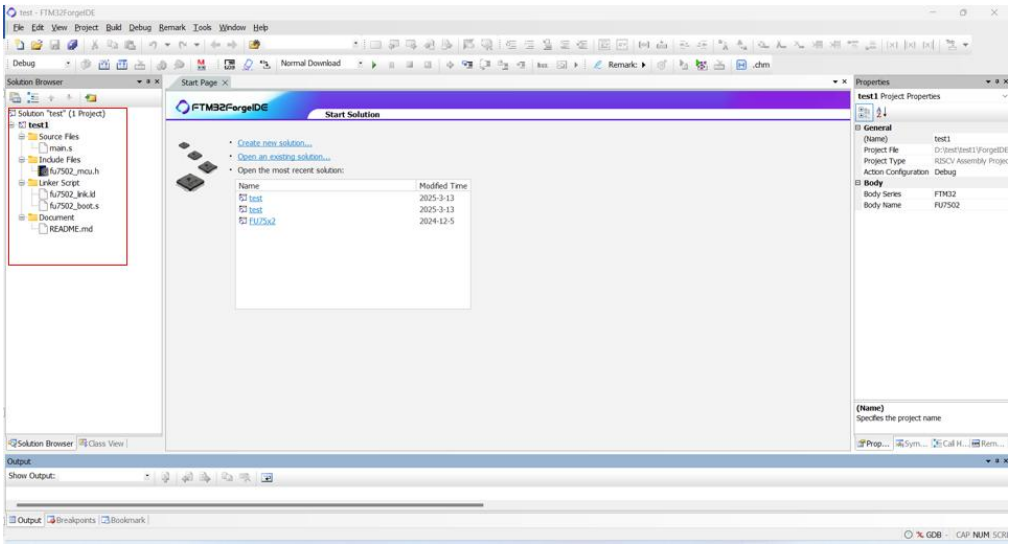


Figure 2-8

2.3 Create A New File

Step 1: Select "File" →"New" →"File" in the upper left corner, as shown in Figure 2-9.

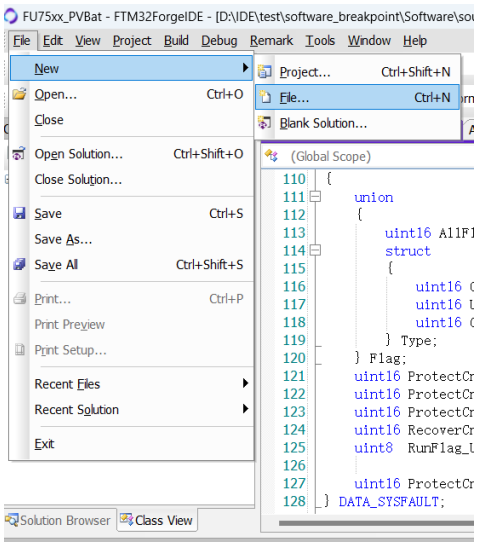


Figure 2-9

Step 2: Select the file type, including "C Header File", "C Source File", "Assembly Header File", "Assembly Source File" and "Text File", as shown in Figure 2-10.

- C Header File: xxx.h;
- C Source File: xxx.c;
- Assembly Header File: xxx.h;
- Assembly Source File: xxx.c;

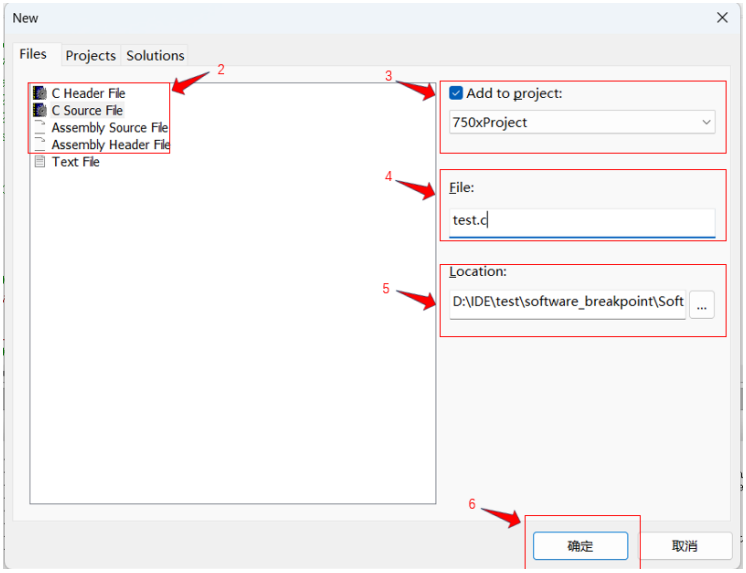


Figure 2-10

Step 3: Check or uncheck "Add to project". If it is selected, the created file is added to the selected project. If not, a separate file is created;

Step 4: Enter the file name, as shown in Figure 2-10. When "C Source File" is selected, you can enter

the file name as "test.c" or "test". If you enter "test", the system automatically completes the file suffix according to the selected file type;

Step 5: Select storage path of the new file;

Step 6: Click "OK". The creation result is shown in Figure 2-11. The newly created "test. c" file appears in "Source Files".

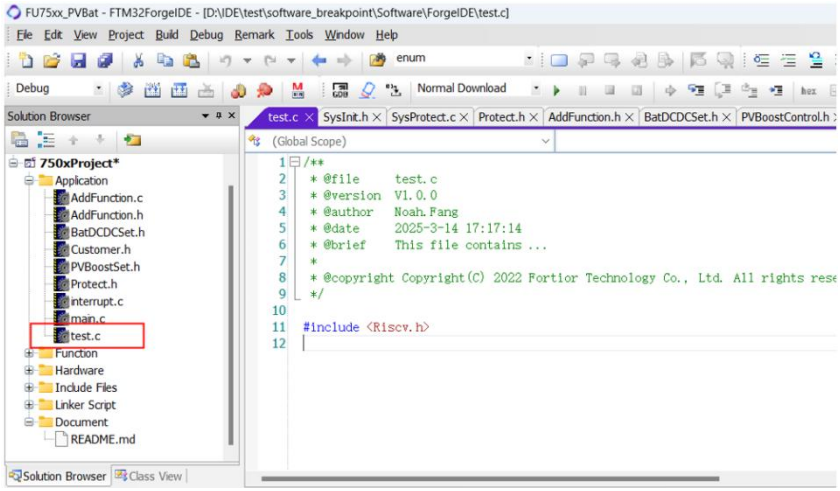


Figure 2-11

3 Code Editor

1. Support the association function. After defining a structure and structure members, and entering the structure name and English "." in the program, you can select the required structure members for the following operations;
2. Display the function at the mouse position, without sliding up and down to find the function entry;
3. Present the syntax of C and assembler by text style and color bar;
4. Directly reach the program line from the error list;
5. Support multi-byte characters and multiple encoding methods  ;
6. Show/Hide editor edge  ;
7. Parentheses match;
8. Auto Indent;
9. Indefinitely undo or redo an action in each window;
10. Click "Remark" to generate ".chm" help file to view the header file information. See 1.1.7 Remark Menu;
11. Click  to generate a “help.chm” help document to facilitate the user to view and understand the header file information;
12. Support the differential display of predefined failure codes, that is, graying out failure codes;
13. Support variable preview function. The variable definition is displayed when the mouse moves onto the variable. It is not necessary to right-click and select "Go to Definition Of 'XXX'" to view the variable;
14. Jump to a function definition. You can move the mouse onto a function, right click and select "Go to Definition Of 'XXX'" to jump to the definition of the function. As shown in Figure 3-1, right click and select "Go to Definition Of 'CMP3_Interrupt_Init'" to jump to its corresponding function definition, as shown in Figure 3-2.

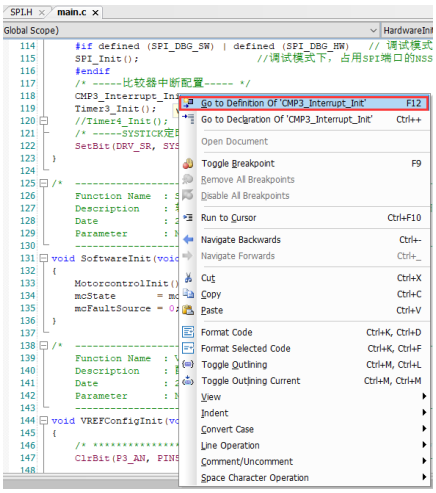
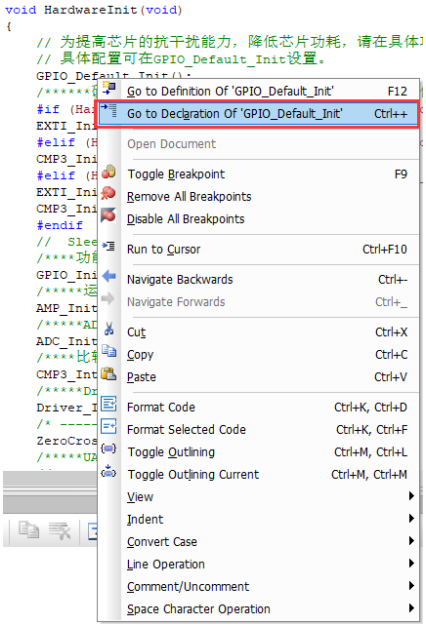


Figure 3-1

Figure 3-2

15. Jump to the declaration of a function in the header file. You can move the mouse onto a function, right click and select "Go to Declaration Of 'XXX'" to jump to the declaration of a function in the header file. As shown in Figure 3-3, right click and select "Go to Declaration Of 'GPIO_Default_Init'" to jump to the function declaration in the head file, as shown in Figure 3-4.



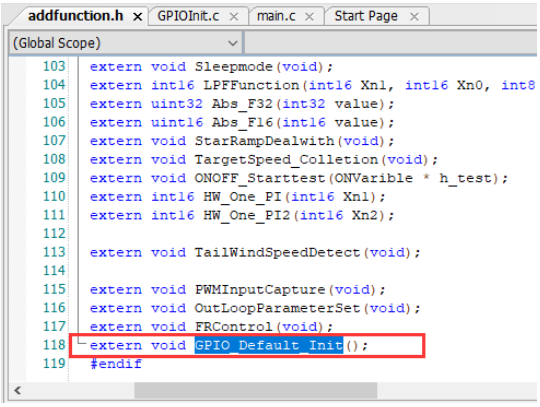
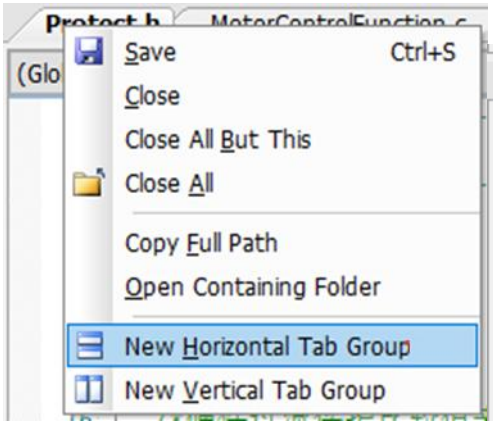


Figure 3-3 Figure 3-4

16. Support double-screen file display function  . If you use this feature to view two files at the same time, such as C file and H file. Right click to select the horizontal screen or click  ; right click to select the vertical screen or click  , as shown in Figure 3-5; If you want to return the single-screen mode, click the option shown in Figure 3-6 or click  again.



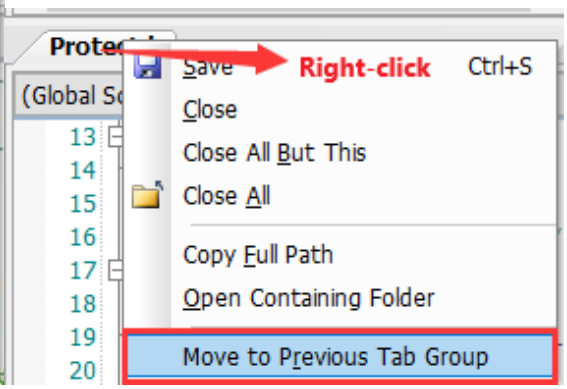


Figure 3-5 Figure 3-6

If horizontal screen is selected, the effect is shown in Figure 3-7 below;

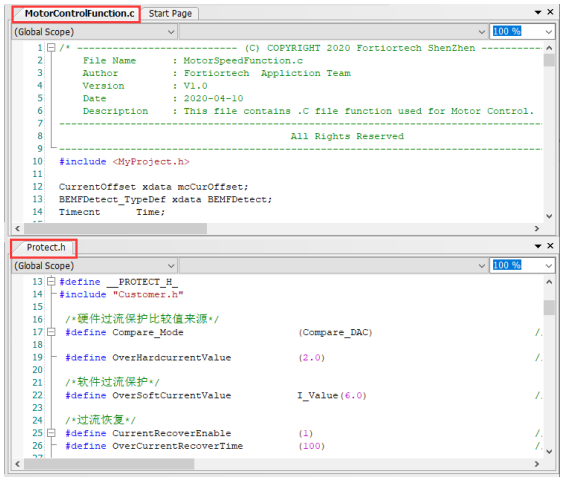


Figure 3-7

If vertical screen is selected, the effect is shown in Figure 3-8 below.

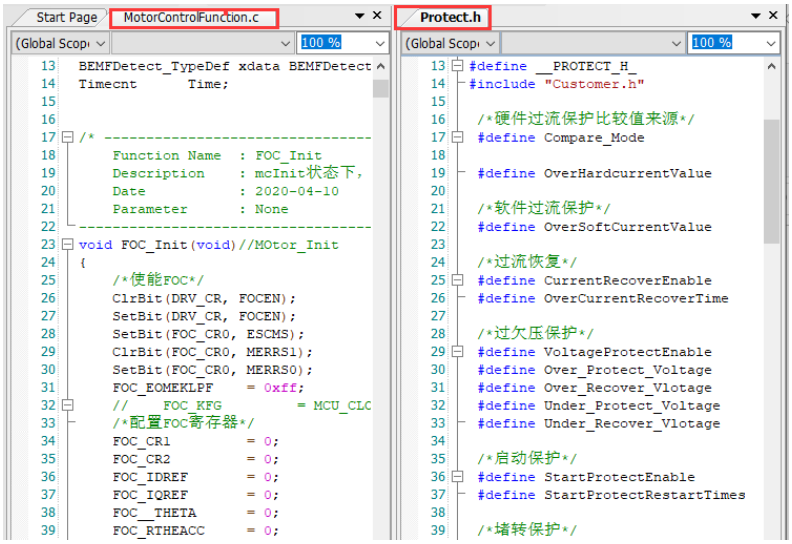


Figure 3-8

17. Quickly open the header file, such as the header file contained in the C file. You can quickly open the header file, as shown in Figure 3-9.

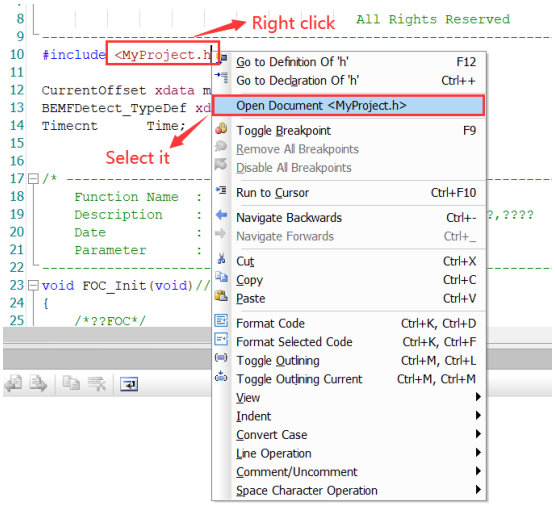


Figure 3-9

18. If there are too many open files and you don't want to close them, you can select them in the way shown in Figure 3-10.

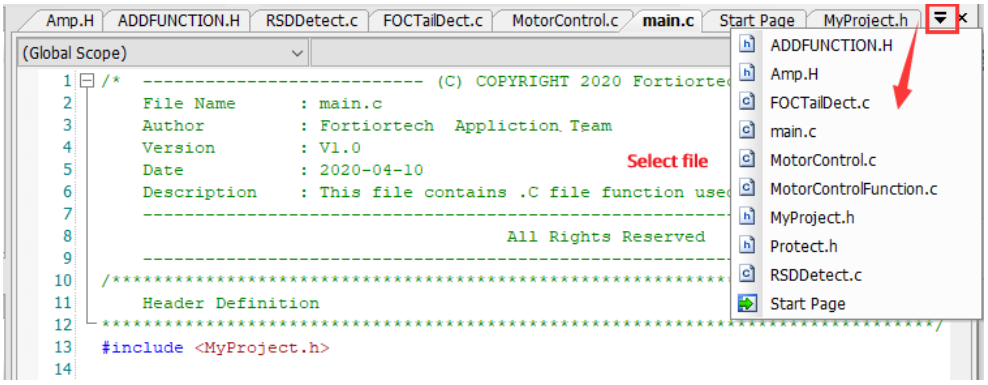


Figure 3-10

You can close the files in the way shown in Figure 3-11. You can also open the corresponding folder or copy the file path. It is very convenient to open the library file, and you do not need to find the file one by one in the installation directory.

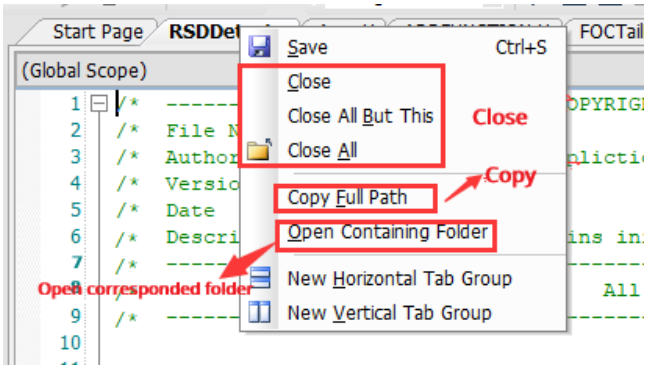


Figure 3-11

19. Find and replace: click  or press Ctrl+F or Ctrl+H to implement the feature, as shown in Figure 3-12. The search results are shown in Figure 3-13.

- In folder: Select the folder to search;
- Look in sub-folders: Search the subfolders under the folder;
- Stop: Stop searching;

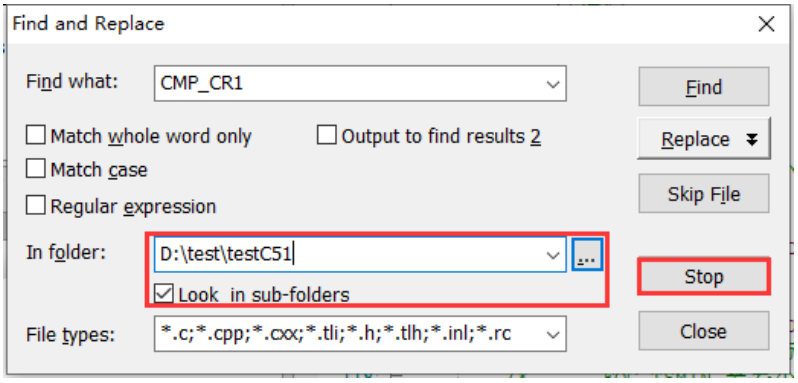


Figure 3-12

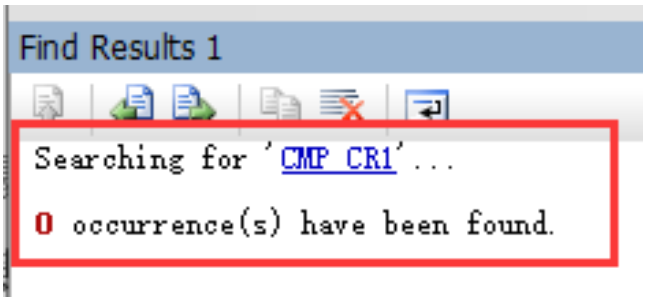


Figure 3-13

Separate search is operated as shown in Figure 3-14. Enter the characters you want to find, then select the search method and direction, and click "Find Next". There are two search directions: Up or Down, and four search methods: "Match whole word only", "Match case", "Regular expression" and "Search all open documents".

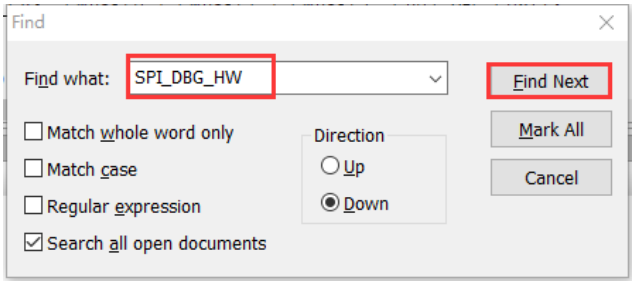


Figure 3-14

Individual replacement is operated as shown in Figure 3-15. Enter the characters that you want you find, then enter the replaced characters. Click "Find Next" to search the characters, and click "Replace" to replace them one by one. You can also click "Replace All" to replace all the characters

at a time. There are three search methods: "Match whole word only", "Match case" and "Regular expression". The replacement range includes the current selected file and the entire file.

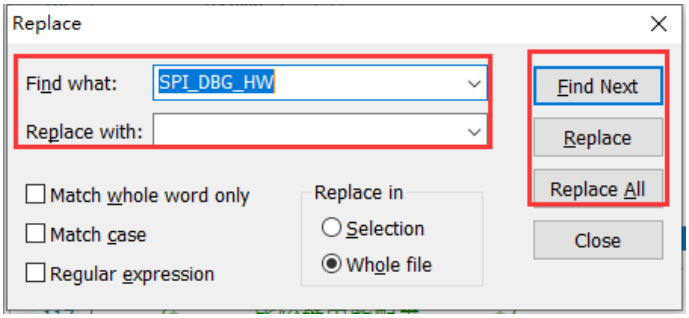


Figure 3-15

4 Compiler

Click  or  to compile the code and generate the ".hex" file. If the code is compiled successfully, as shown in Figure 4-1 below, Output window displays the blue font " Build Project FU75x2_FOC successfully ", and generates "FU75x2_FOC.hex" file with 0 error. If there is a warning, the number of warning is displayed as well.

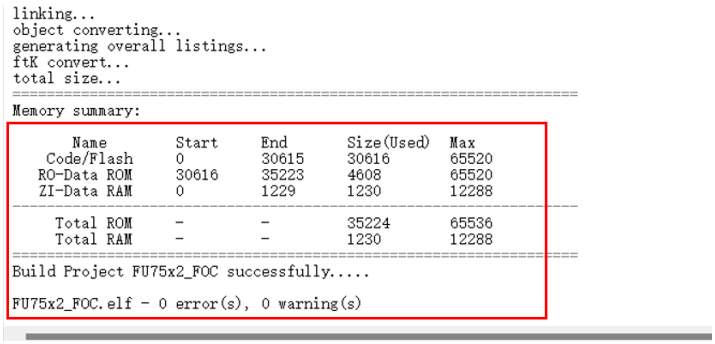


Figure 4-1

As shown in Figure 4-1, after the code is compiled successfully, the start and end addresses, space size and maximum memory space of each RAM are displayed.

The start address of Code/Flash is 0, the end address is 30615, the space size is 30616 bytes, and the maximum memory is 65520 bytes;

The start address of R0-Data ROM is 30616, the end address is 35223, the space size is 4608 bytes, and the maximum memory is 65520 bytes;

The start address of ZI-Data RAM is 0, the end address is 1229, the space size is 1230 bytes, and the maximum memory is 12288 bytes;

The start address of ROM/EPROM/FLASH is 0x0000, the end address is 0x6a8c, the space size is 27277 bytes, and the maximum memory is 32768 bytes.

If the compilation fails, the number of errors and warnings are displayed in the Output window, and the location of errors and warnings are indicated in red font. As shown in Figure 4-2, one error occurs, and it is generated at line 33 of the "AddFunction.c" file. The error type is syntax error, and the token ";" is missing.

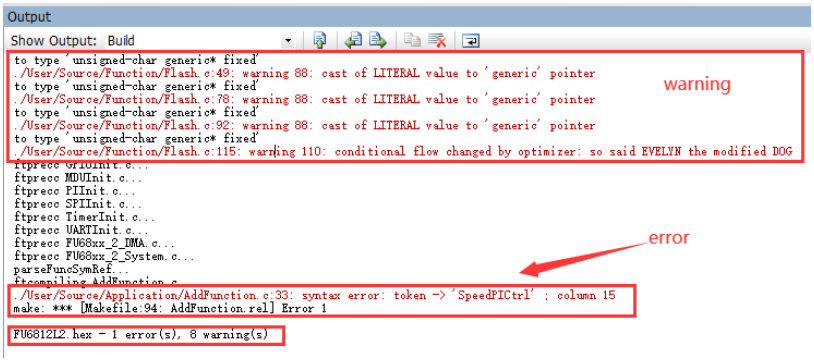


Figure 4-2

Double click the error to reach the location where the error occurs (similarly, double lick the warning to reach the location where the warning is generated), as shown in Figure 4-3, that is, the location where the blue cursor hits.

In particular, the compiler alerts that there is a syntax error in line 33, but in fact the error is often not in line 33. It is likely that the error is caused by a syntax error in front of line 33. In this case, line 33 and its preceding lines shall be checked as well. Through inspection, it is found that the error is in line 32. The token ";" is missing in the assignment statement. After the token is added, the compilation succeeds.

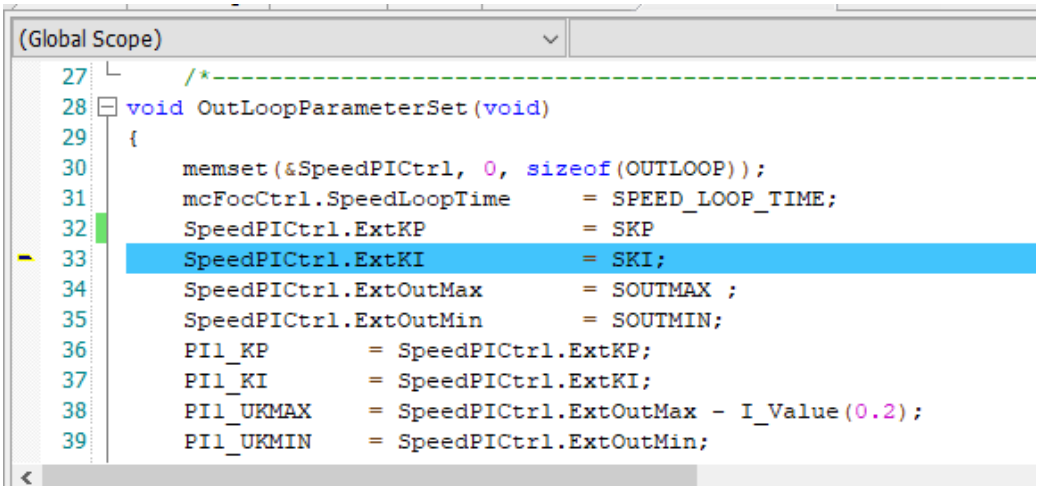


Figure 4-3

5 Download and Emulation

5.1 Download

FTM32ForgeIDE supports methods to download the program code: Smart Download, Normal Download and Without Download.

- Smart Download: After the chip is successfully programmed for the first time, if the program code is changed, programming operation is not performed during the next download so as to save time;
- Normal Download: The program code is downloaded each time when the chip is programmed;
- Without Download: No download operation is performed. IDE directly enters the simulation environment and considers the chip has been programmed.

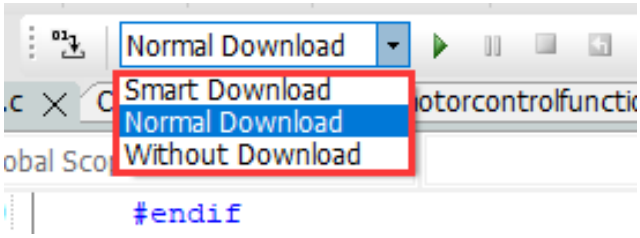


Figure 5-1

Connect the computer with the programming tool or development board, and click  to download the compiled hex file to the Flash of the chip. The download process is shown in Figure 5-2.

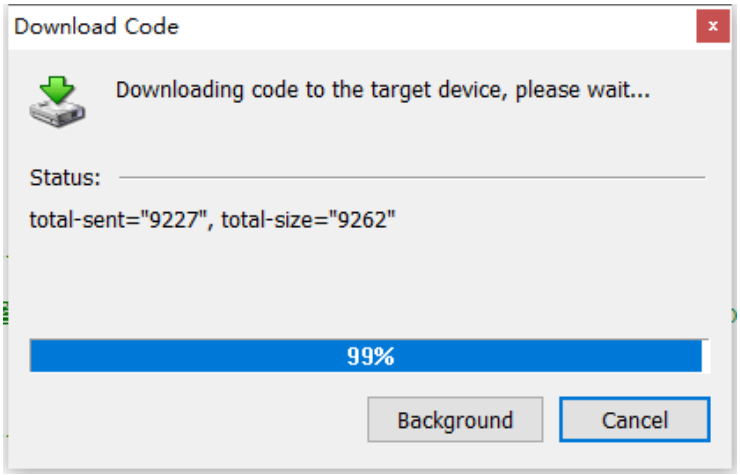


Figure 5-2

The program code has been successfully downloaded to the chip when the download progress bar reaches 100%. The download results are as shown in Figure 5-3: the device has been successfully connected, the Flash erase has been completed, 9262 bytes of program code has been written to Flash, and the Flash verification is successful.

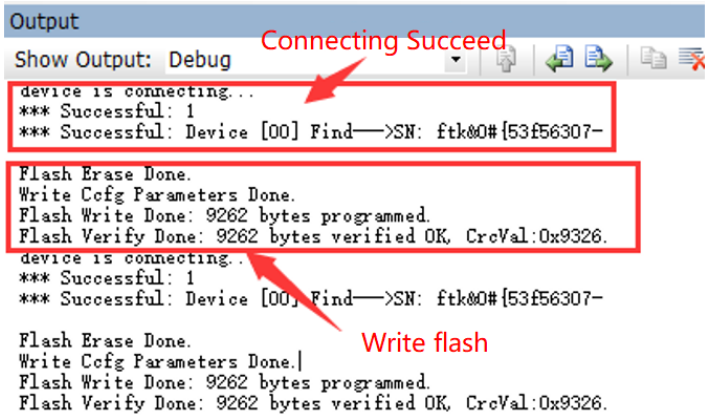


Figure 5-3

When you click the download button to download the program code and the computer fails to connect to the programming tool (such as the emulator) , a pop-up window appears to notify an error, as shown in Figure 5-4.

Similarly, you click the download button to download the program code and the computer fails to connect to development board, a pop-up window appears to notify an error, as shown in Figure 5-4. Also, Output window displays error message to alert the connection failure, as shown in Figure 5-5.

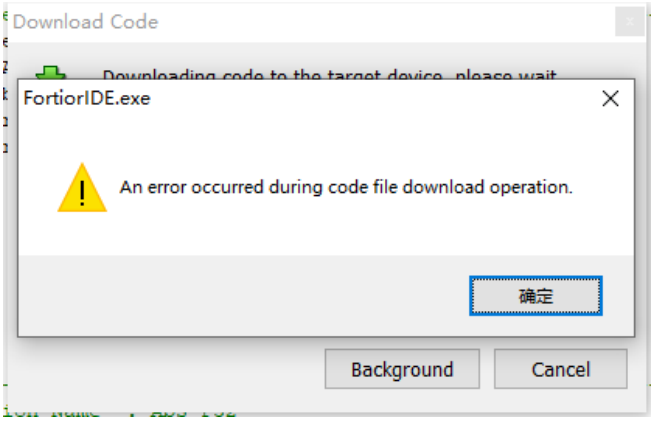


Figure 5-4

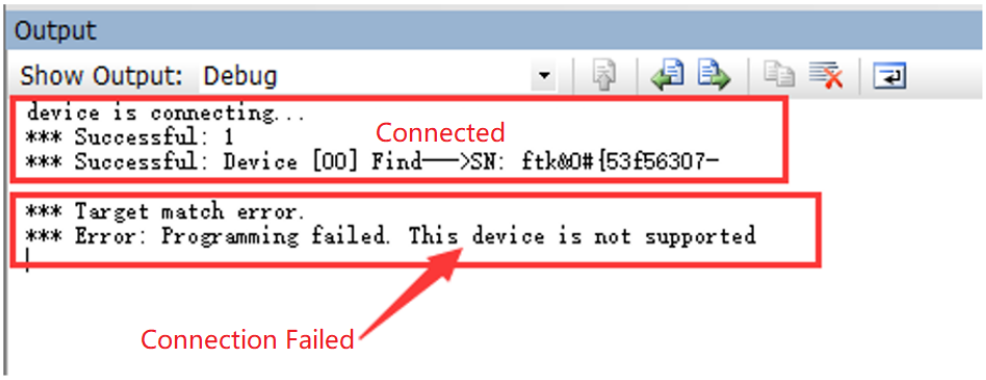


Figure 5-5

5.2 Emulation

Connect the computer with the programming tool or development board, click  to conduct emulation debugging and download the compiled hex file to the flash of the chip. The download process is shown in Figure 5-2. When the download progress bar reaches 100%, the program code has been successfully downloaded to the chip, and can be used for emulation debugging. The debug interface is shown in Figure 5-6.

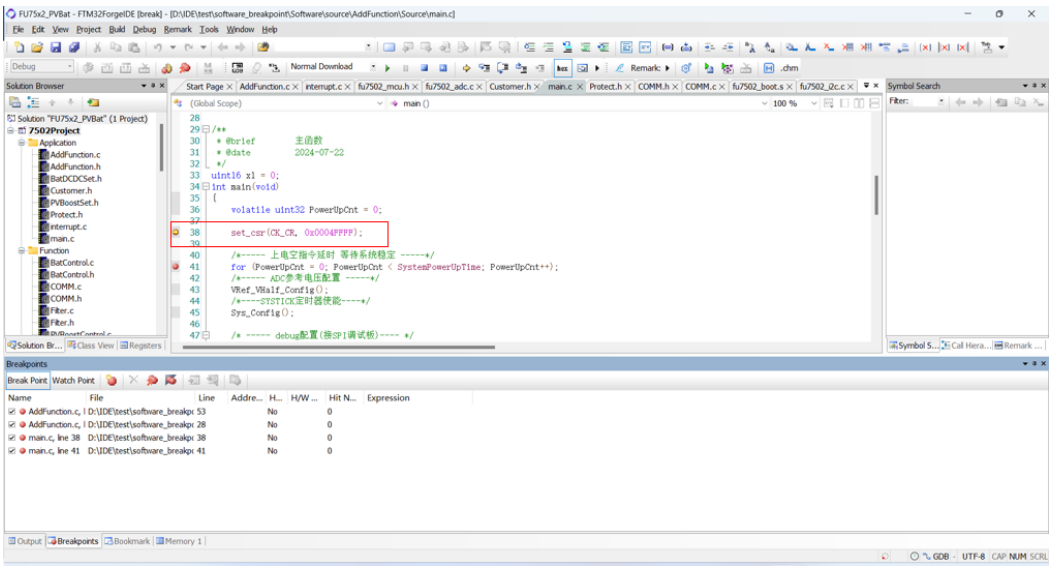


Figure 5-6

Click  to perform disassembly instruction debugging, and click  or F10 to conduct single-step debugging, as shown in Figure 5-7.

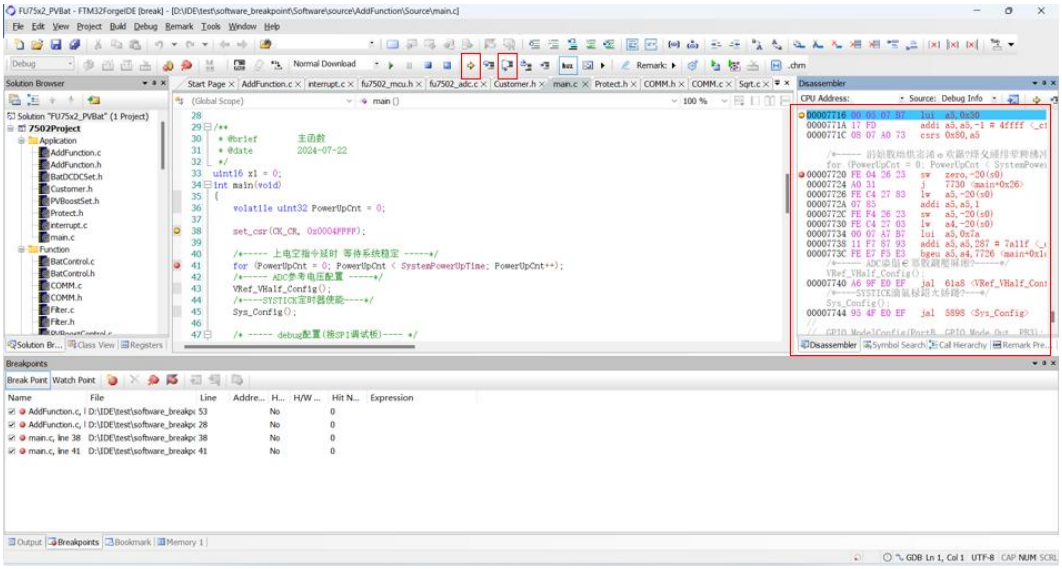


Figure 5-7

Click  to set the breakpoint. Click  or F5 to reach the next breakpoint from current code line, as shown in Figure 5-8.

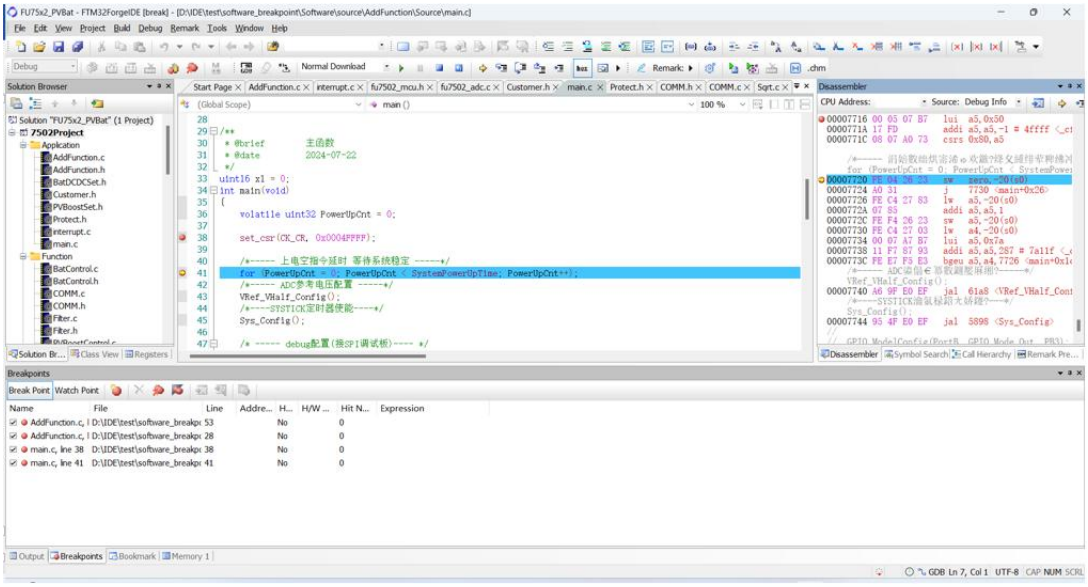


Figure 5-8

Click  to perform debugging in a function, and click  to exit the function, as shown in Figure 5-9.

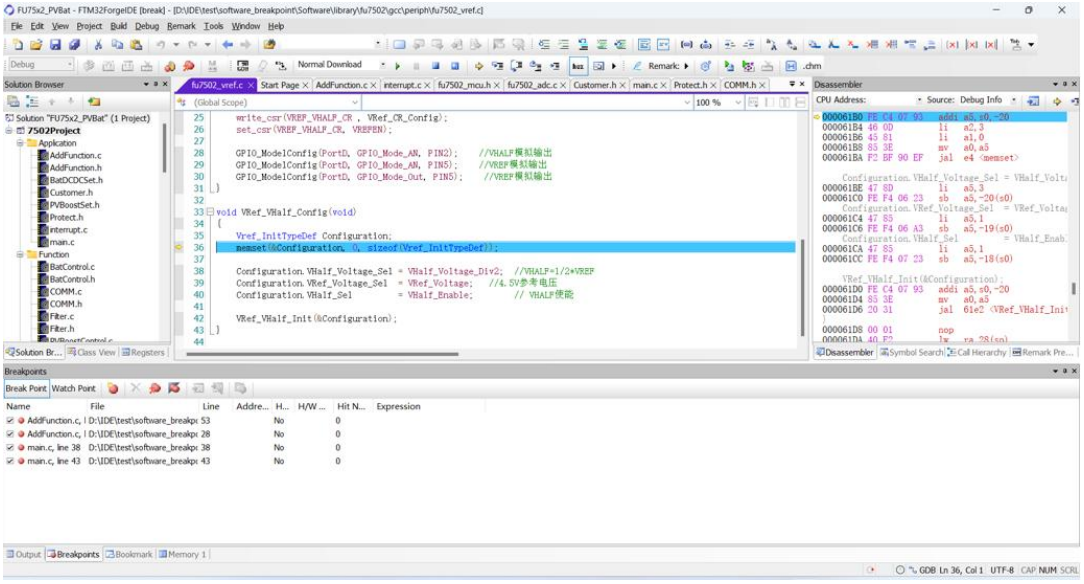


Figure 5-9

Output window displays disassembly window, register window, watch window and memory window. See section 1.3 Window Introduction for details.

6 Revision History

Version	Description	Date	Prepared By
V1.0	First release.	2025/03/19	Freya Fu

Copyright Notice

Right to make changes — Fortior Technology (Shenzhen) Co., Ltd. reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. The information contained in this manual is provided for the general use by our customers. Our customers shall ensure that they take appropriate action so that their use of our products does not infringe upon any patents. It is the policy of Fortior Technology (Shenzhen) Co., Ltd. to respect the valid patent rights of third parties and not to infringe upon or assist others to infringe upon such rights.

If there are any differences between the Chinese and the English contents, please take the Chinese version as the standard.

Fortior Technology (Shenzhen) Co., Ltd.

Room 203, 2/F, Building No.11, Keji Central Road 2,

Software Park, High-Tech Industrial Park, Shenzhen, P.R. China 518057

Tel: 0755-26867710

Fax: 0755-26867715

Web: www.fortiortech.com

Contained herein