

2.5.Simulation and Debugging

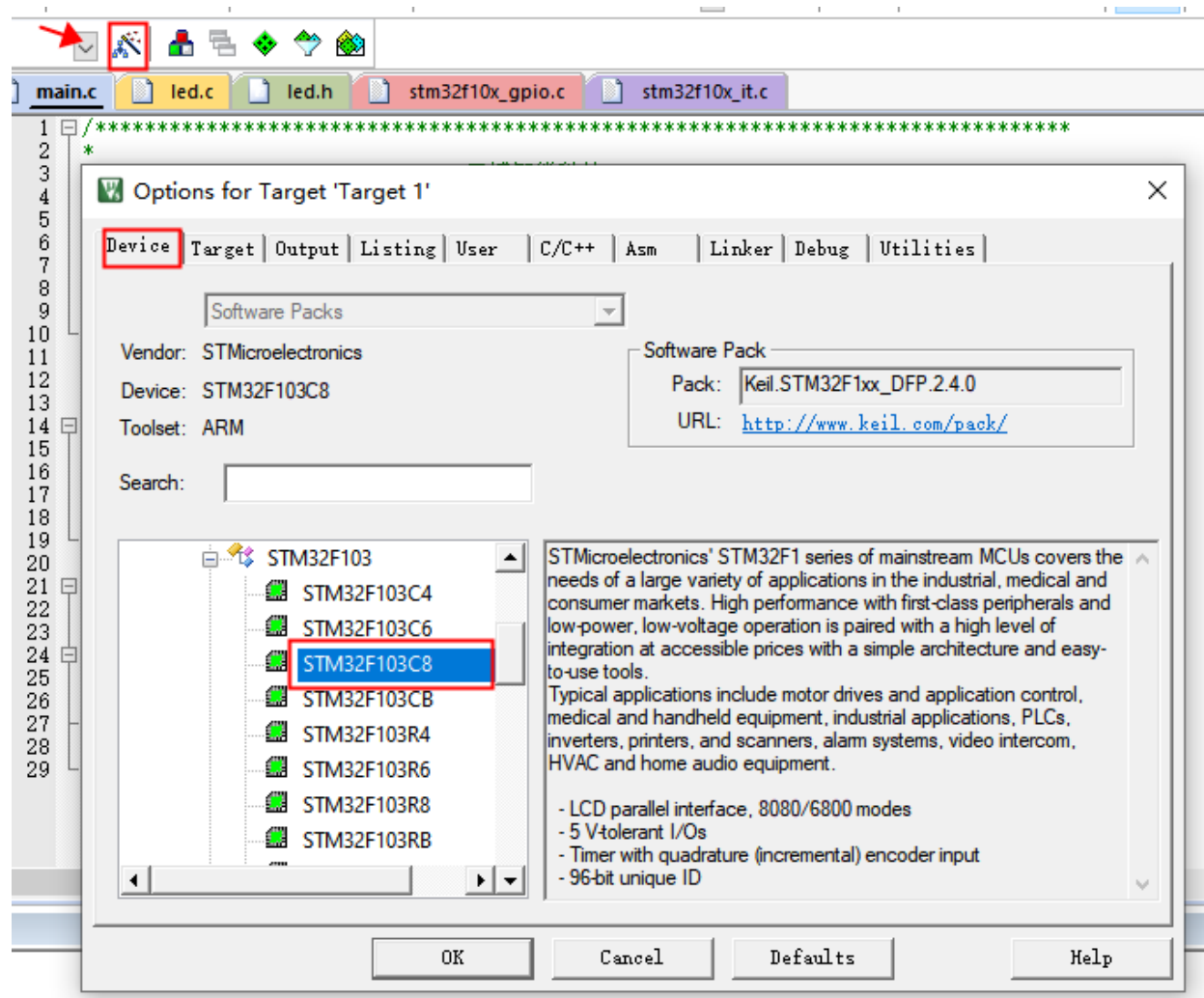
As a simulation software function with great potential, Keil can quickly solve the problems of code running through debug, and determine the location of the problem, check many hardware-related registers, and observe the changes of these register values. Does the code work properly. This can avoid frequent program downloads and prolong the life of the microcontroller Flash.

This tutorial uses STLINK for simulation. If you do not have hardware equipment, you need to purchase it yourself.

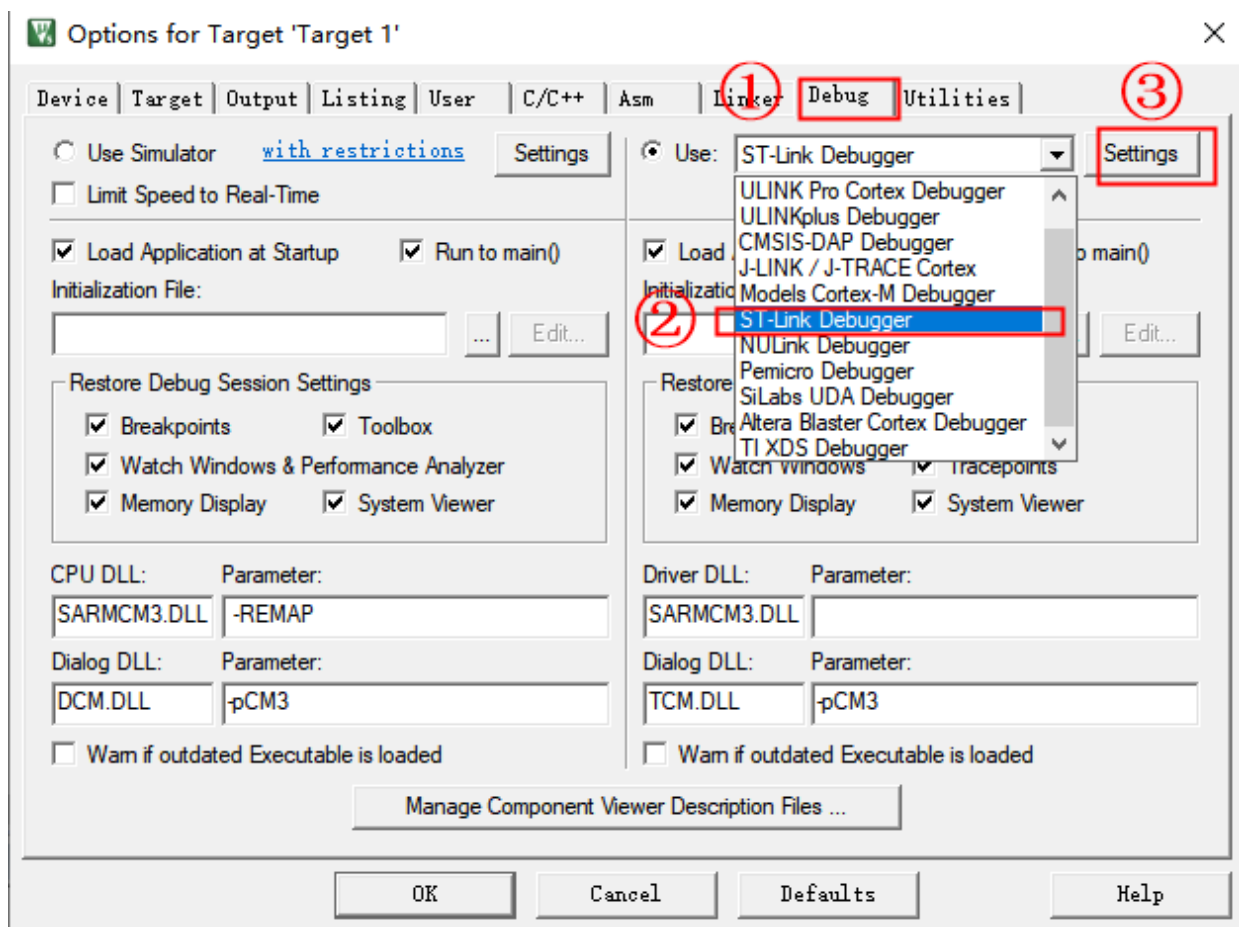
The basic simulation and debugging of keil5 are divided into two steps:

1. Software Configuration

Click the magic wand and select the STM32F103C8 chip in "Device"



Select the emulator in "Debug", I use STLINK, if you use JLINK or other emulators, you can replace it yourself.



After clicking “Setting”, go to the setting interface and select “SW” mode. If the hardware is normal and the wiring is ok, a device number will be automatically recognized.

Debug | Trace | Flash Download | Pack

Debug Adapter

Unit: ST-LINK/V2 ①

☐ Shareable ST-Link

Serial

57FF6E064984525035171587

Version: HW: V2 FW: V2J38S7

☒ Check version on start

Target Com

Port: SW ②

Clock

Req: 10 MHz Selected: 1.800 MHz

SW Device

IDCODE	Device Name	Move
0x1BA01477	ARM CoreSight SW-DP (ARM Core	Up Down

☒ Automatic Detection ID CODE:

☐ Manual Configuration Device Name:

Add Delete Update IR len: AP: 0

Debug

Connect & Reset Options

Connect: Normal Reset: Autodetect

☒ Reset after Connect ☐ Stop after Reset

Cache Options

☒ Cache Code ☒ Cache Memory


Download Options

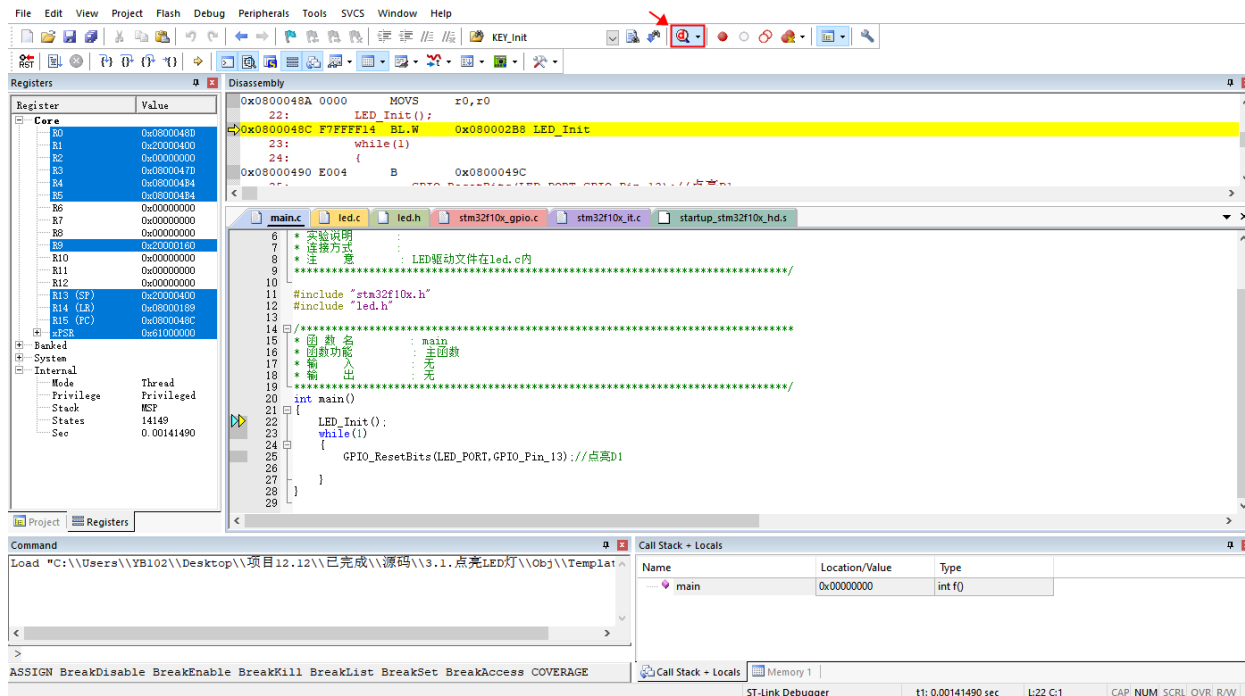
☐ Verify Code Download ☐ Download to Flash

确定 取消 应用(A)

After configuring the above, you can start debugging.

2.debug operation

Click on the debug icon  , Enter the debugging interface



Debug toolbar introduction, the order of introduction is from left to right::



1: reset

As the name suggests, reset means to start from scratch again, and its function is equivalent to pressing the reset button on the hardware, which is equivalent to realizing a hard reset. When the button is pressed, the code executes from the beginning again.

2: Execute to breakpoint

Execute to breakpoint Press this button to quickly execute to the breakpoint. Sometimes you don't need to watch how each step is executed. You just want to quickly execute to a certain place in the program to see the result. Press this button to satisfy your needs. Desire, of course, the premise is that a breakpoint has been set in the viewing place.

3: stop running code

This button is only available when the program has been executed, and it can stop the program and enter the single-step debugging state. Instantly puts the program in a stop state.

4: Stepping through (into a function)

Simply put, the button is to enter a certain function, if there is no function, it is equivalent to executing the last function

5: step-by-step debugging (stepping over functions)

When encountering a place with a function, the button can be used to step through the function without stepping into the function. In short, the function is skipped and not executed.

6: out of debug (Step out of the executed function)

This button is to enter the step-by-step debugging of the function. Sometimes it may not be necessary to execute the remaining part of the function. Through this button, the remaining part of the function can be directly executed in one step, and the function will jump out and return to the place where the function was called.

7: Execute to the cursor position

This button can quickly run the program to the cursor, which is similar to the button function of executing to the breakpoint.

8: Show the line of code being executed

The following adds a menu for the debug mode

9: command display window

10: reverse assembly window

This button lets you view assembly code, which is useful for analyzing programs.

11: module window

12: register window

13: Called function stack window

Through this button, the Call Stack+Locals window can be displayed, displaying the local variables and their values of the current function for easy viewing.

14: View (variables) window

MDK5 provides 2 observation windows (drop-down selection), and when this button is pressed, a window displaying variables will pop up. Enter the variable or expression you want to observe, and you can view its value. It is a very common debugging window.

15: memory window

MDK5 provides 4 memory viewing windows. Press the button, and a memory viewing window will pop up. You can enter the memory address to be viewed in it, and then observe the changes of this piece of memory.

16: Serial UART window

MDK5 provides 4 serial printing windows. Press this button, and a window similar to the serial port debugging assistant interface will pop up, which is used to display the content printed from the serial port.

17: Logic Analyzer window

There are 3 options below the icon, the first one is generally used, which is the logic analysis window. Create some IO ports through the SETUP button, so you can observe the level changes of these IOs and display them in various forms, which is more intuitive.

18: tracking window

19: System (peripheral IO, USART, TIM, etc.) window

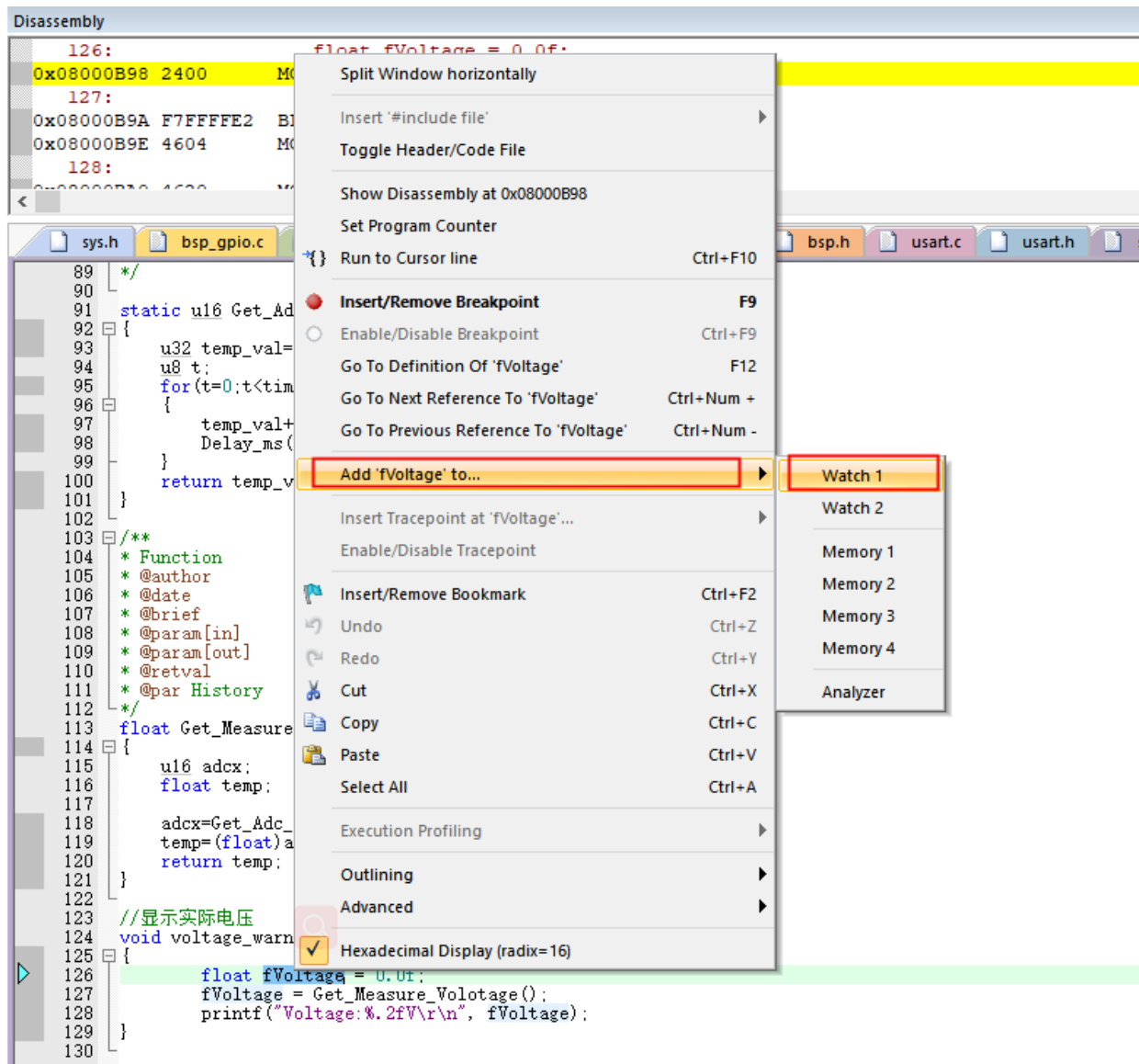
This button can provide a viewing window for various peripheral registers (selected by pull-down).
Select the corresponding peripheral to call up the relevant register table of the peripheral and display the values of these registers, which is convenient for checking whether the settings are correct.

20: toolbox window

Common debugging operations:

1.Add watch window

Add a watch window for the observed variable (position the cursor on the variable, right-click and select Add "start_flag to..."----->Watch1)



2.set breakpoint

After setting the breakpoint, you can debug in a single step and observe the running results of the program

The screenshot shows an IDE interface with the following components:

- Registers Panel:** Lists registers R0 through R15 and FSR. Register R9 is highlighted with a value of 0x20000160.
- Disassembly View:** Shows assembly code for the 'bsp_adc.c' file. Line 125 is highlighted in yellow. A red arrow points to the 'Run' button in the disassembly toolbar.
- Command Window:** Displays the command 'Load "C:\Users\YB102\Desktop\项目12.12\已完成\源码\3.4.ADC采集\ADC采集\output\BS\TB6612FNG\Source\APP\main.c\32, 1' and an error message: '*** error 35: undefined line number'.
- Watch Window:** Shows a watch entry for 'fVoltage' with a value of '<cannot evaluate>' and type 'uchar'. A red circle '3' is around this entry.

Generally, the program does not report an error, but the corresponding effect is not observed after downloading. You can set a breakpoint from the beginning of the main function to see if the corresponding function is executed according to the expected effect, whether the change of the variable is consistent with the expectation, more The debugging method can be learned from Baidu.