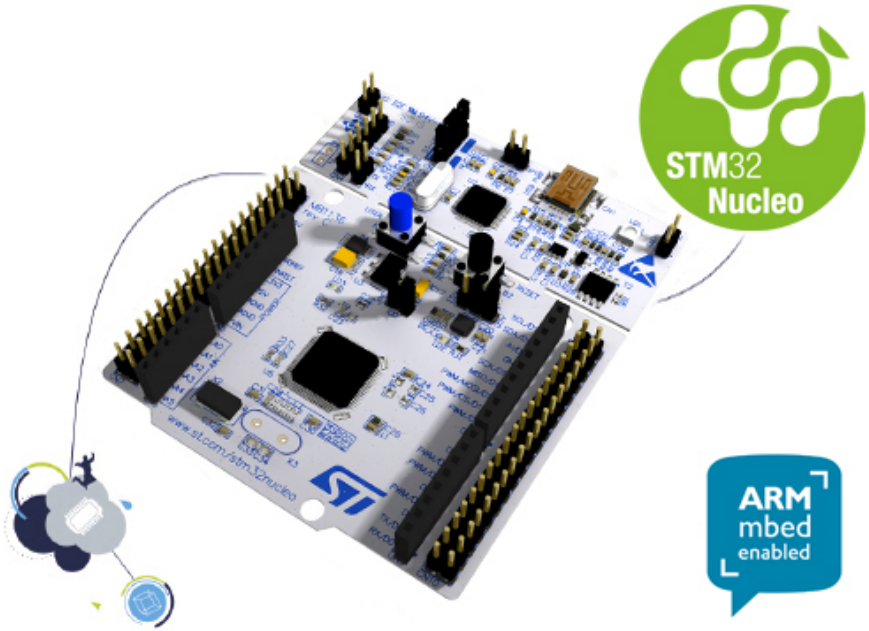
Overview of STM32 Board/Development Environment

STM32 Nucleo Board

- An ARM Cortex-M4 development board
- Build in a ST-LINK as debugger
- Arduino pin compatible



Hardware Block

Figure 3. Top layout

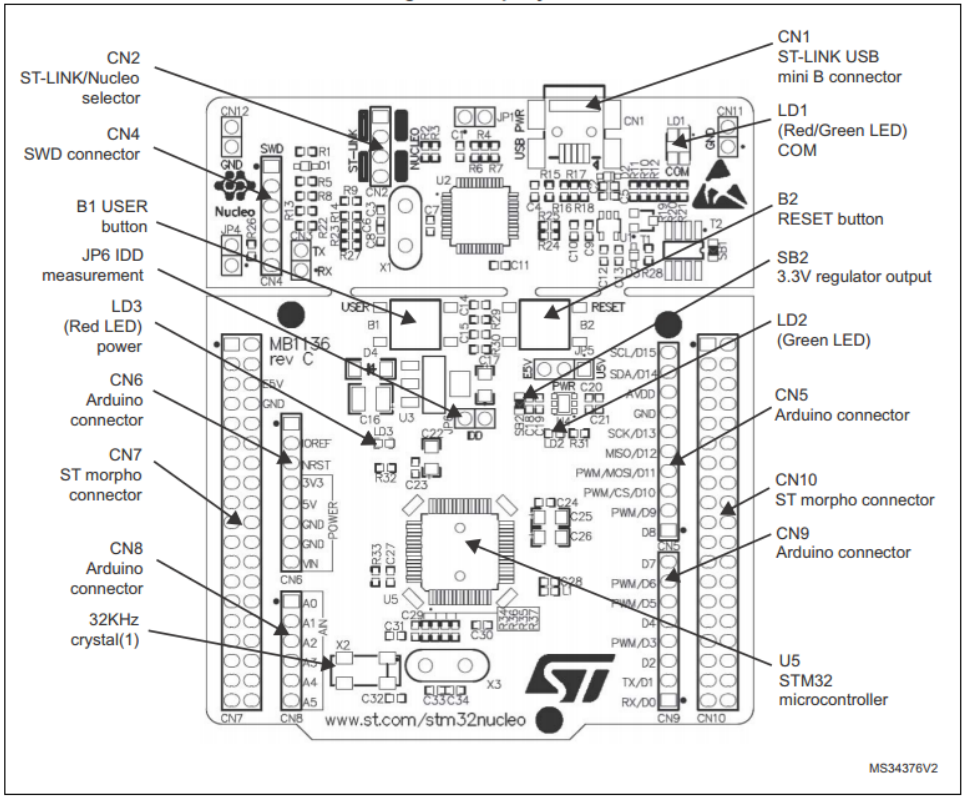
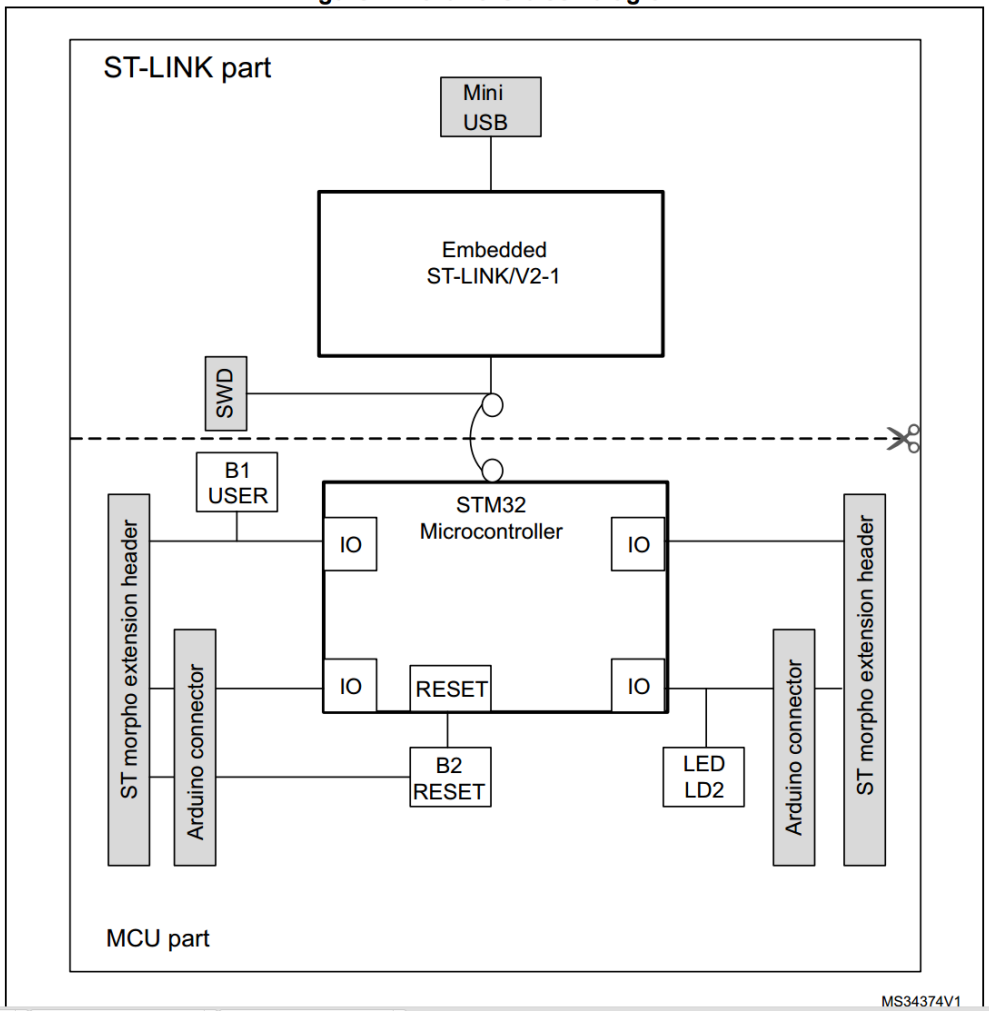
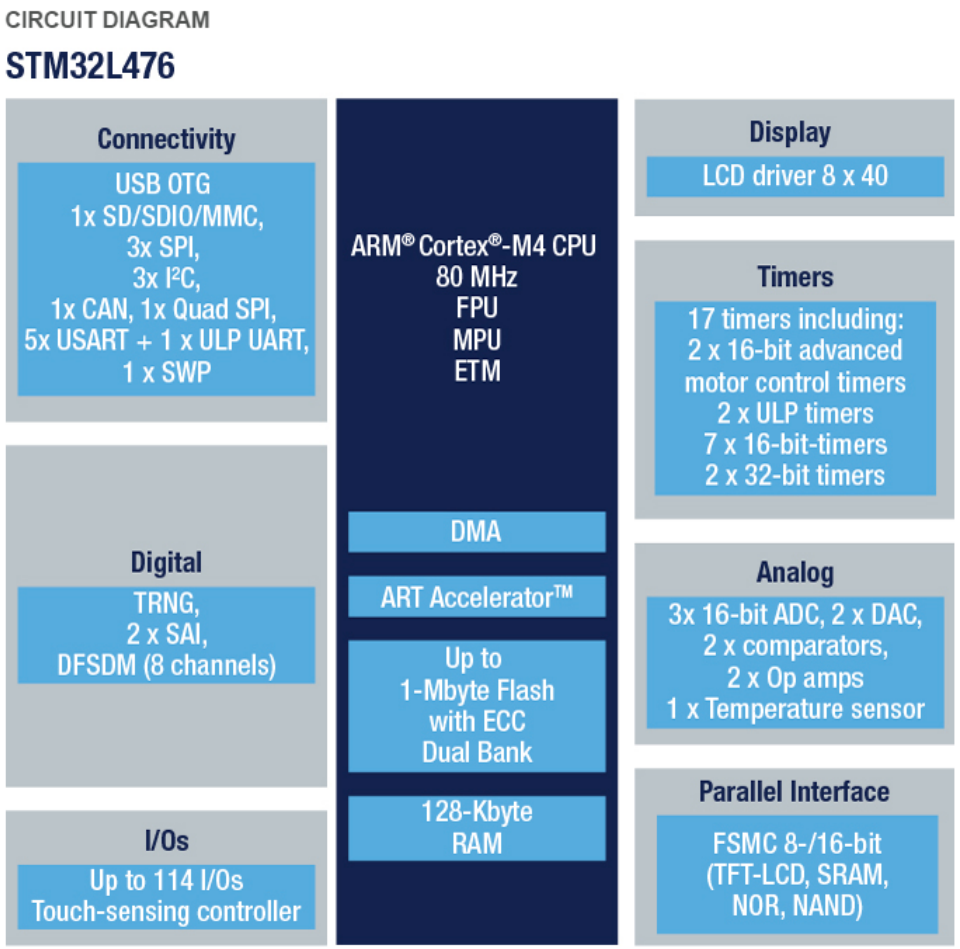


Figure 2. Hardware block diagram



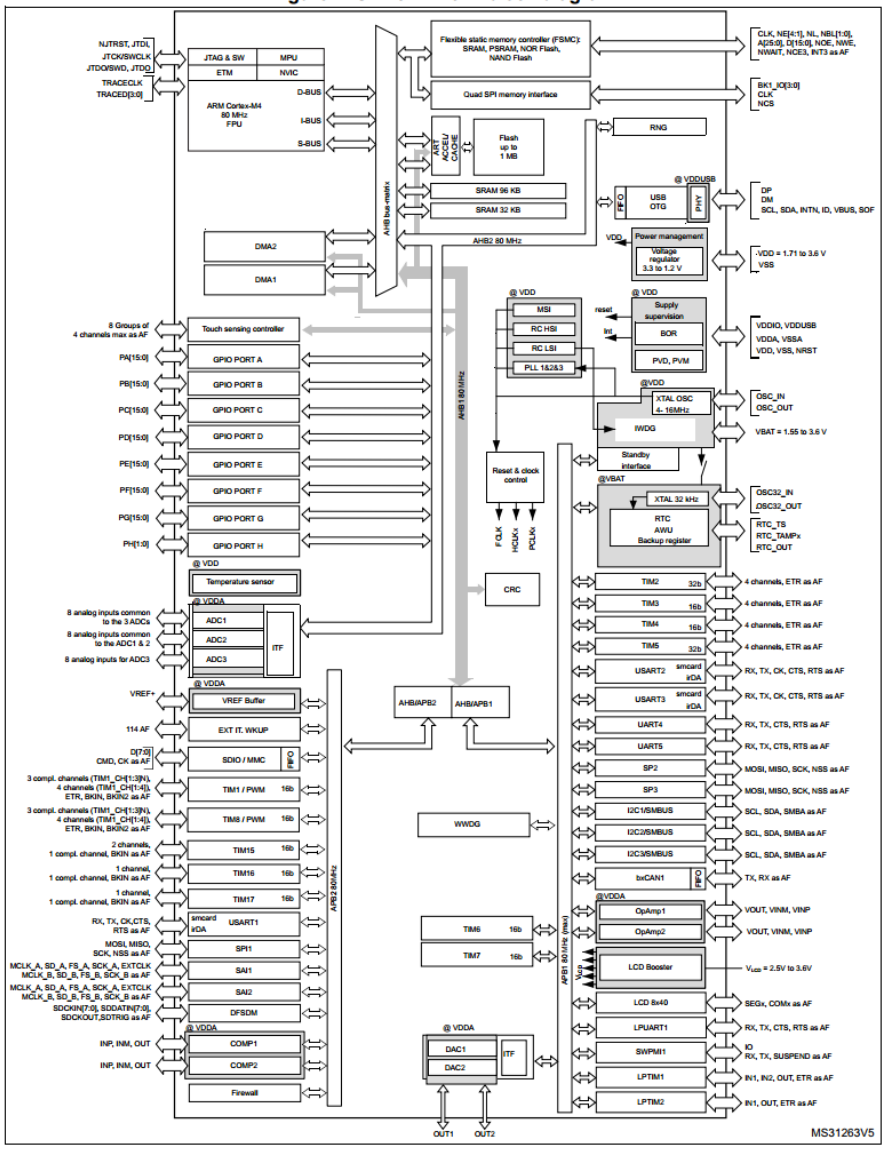
Features

- ARM Cortex-M4
  - frequency up to 80 MHz
  - MPU
  - 100DMIPS/1.25DMIPS/MHz (Dhrystone 2.1)
  - DSP instructions
- Memories
  - 1MB Flash
  - 128KB SRAM
- 51 GPIO pins
- Timers
- Communication interfaces
  - I2C, SPI, CAN, USART, USB,…
- 12-bit ADC and DAC



Block diagram

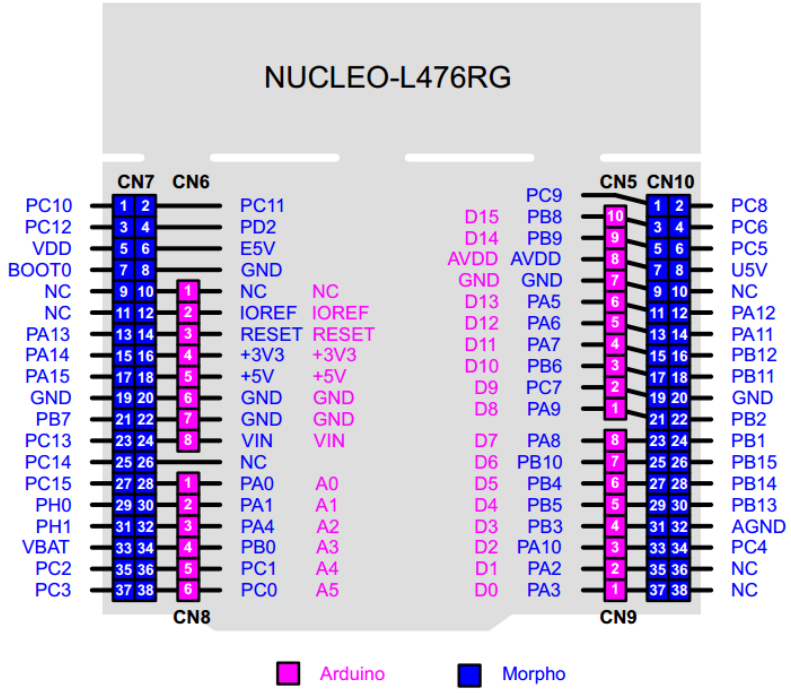
Figure 1. STM32L476xx block diagram



Note: AF: alternate function on I/O pins.

Pin Map

Figure 22. NUCLEO-L476RG



Development Environment

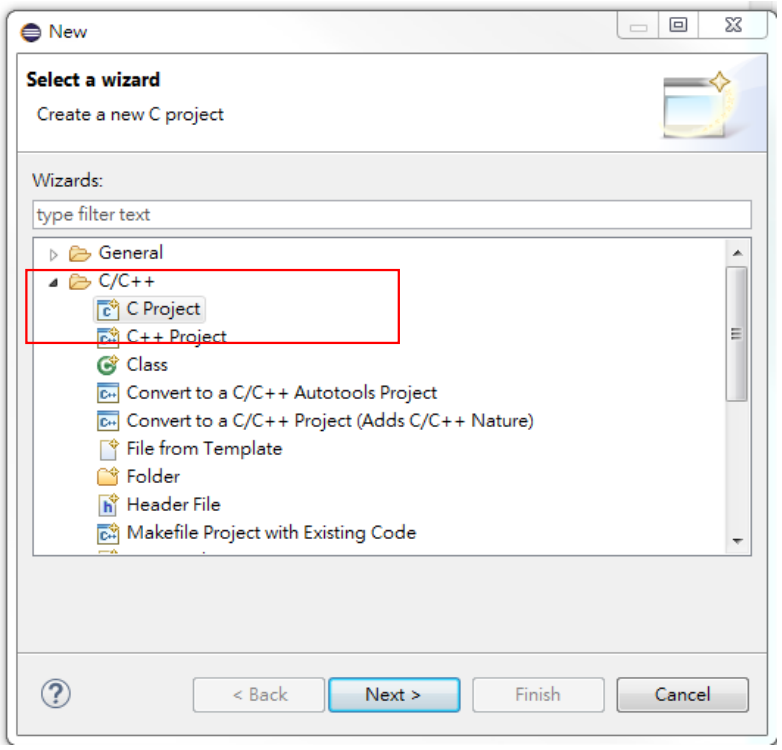
- We use SW4STM32 which is a eclipse based STM32 IDE tool
  - STM32 Devices database and libraries
  - Source code editor
  - Linker script generator
  - Building tools (GCC-based cross compiler, assembler, linker)
  - Debugging tools (OpenOCD, GDB)
  - Flash programing tools
  - http://www.openstm32.org/HomePage

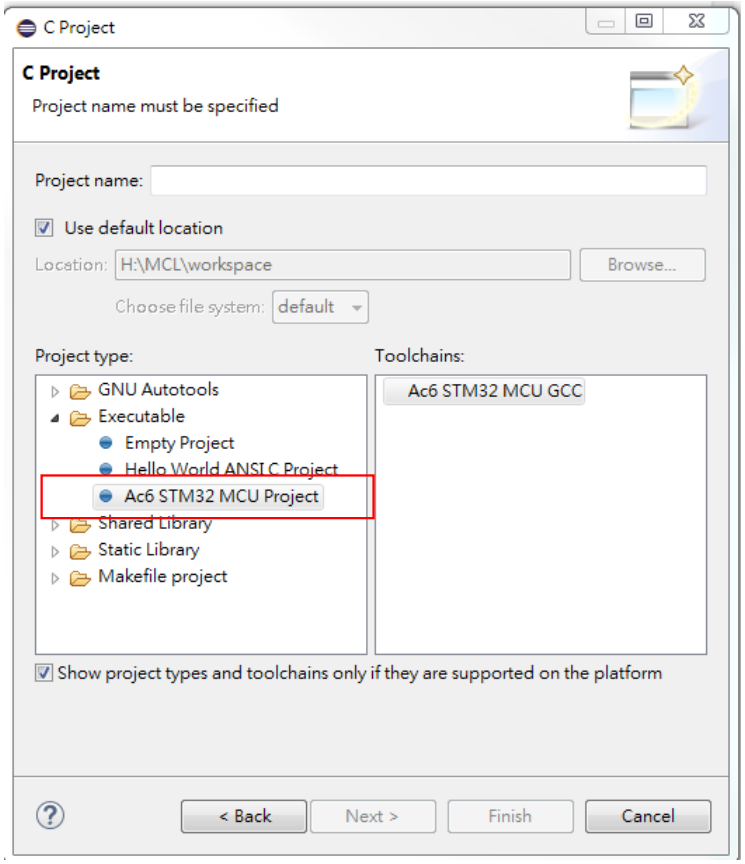
SW4STM32

- Download from http://www.openstm32.org/
- Windows 7
  - http://www.ac6-tools.com/downloads//SW4STM32/install_sw4stm32_win_64bits-v1.8.zip
- Linux
  - http://www.ac6-tools.com/downloads/SW4STM32/install_sw4stm32_linux_64bits-latest.run
  - Dependence
    - JRE7
    - sudo apt-get install libc6:i386 lib32ncurses5

Create Project

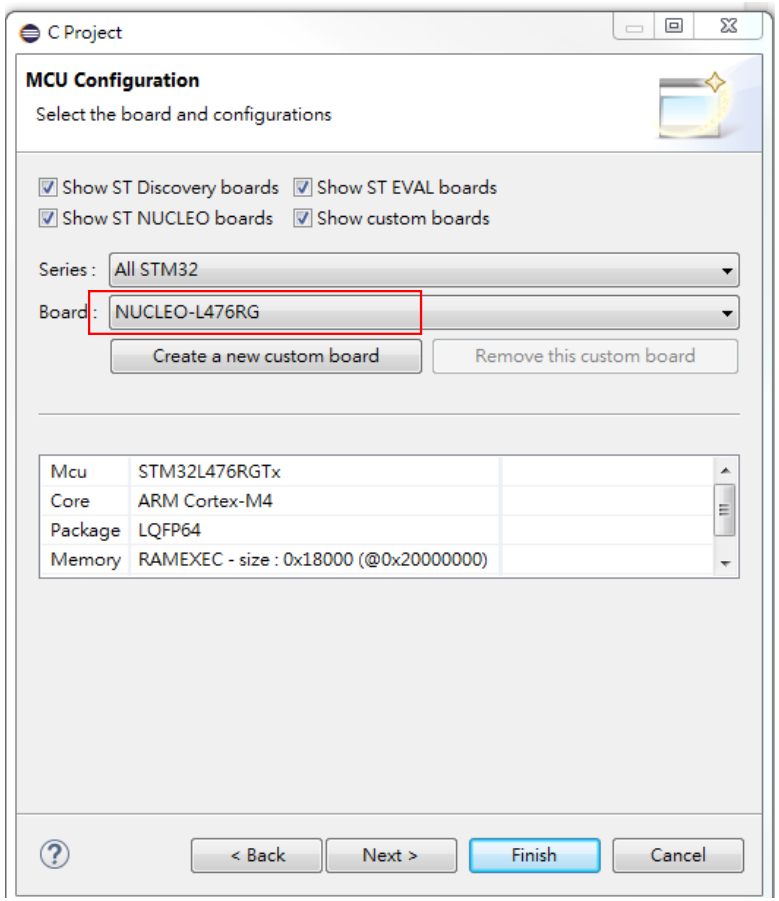
- Create a ‘lab1’ project



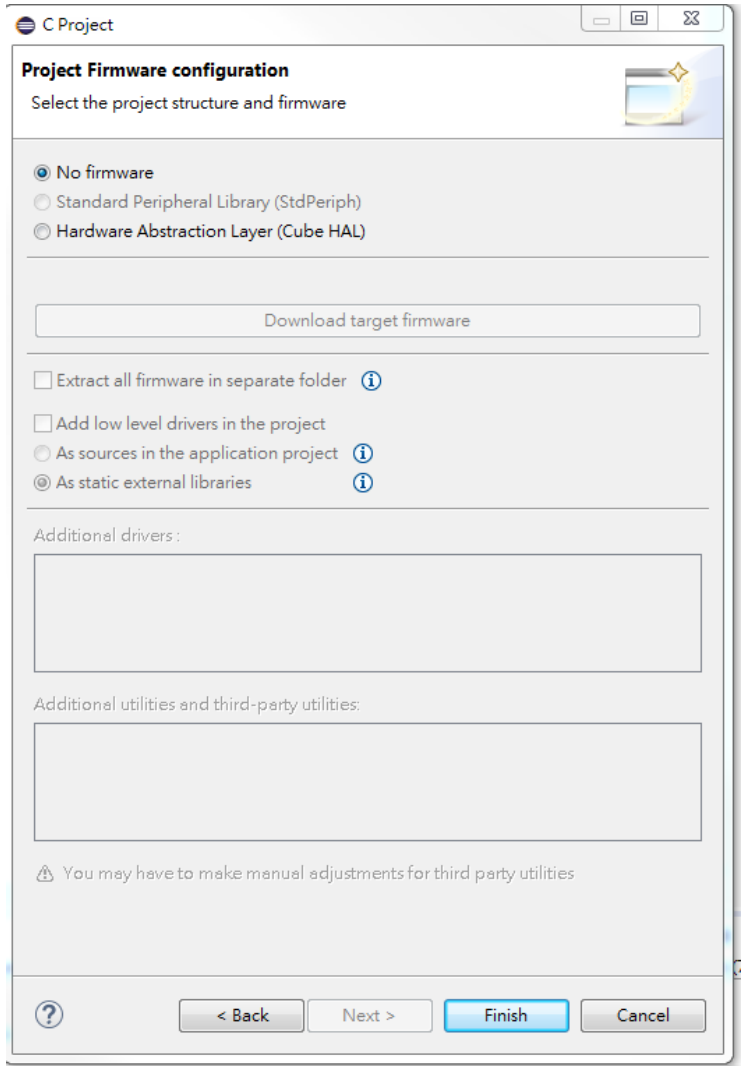


MCU Configuration

- Select NUCLEO-L476RG board

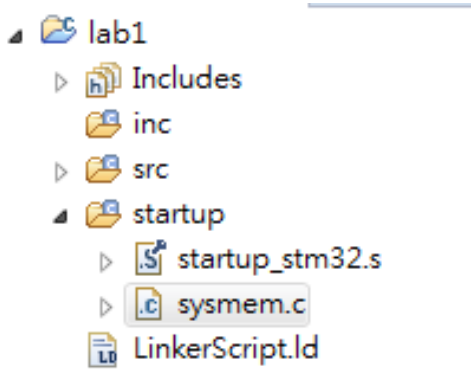


- Choose ‘No firmware’
- Then press ‘Finish’



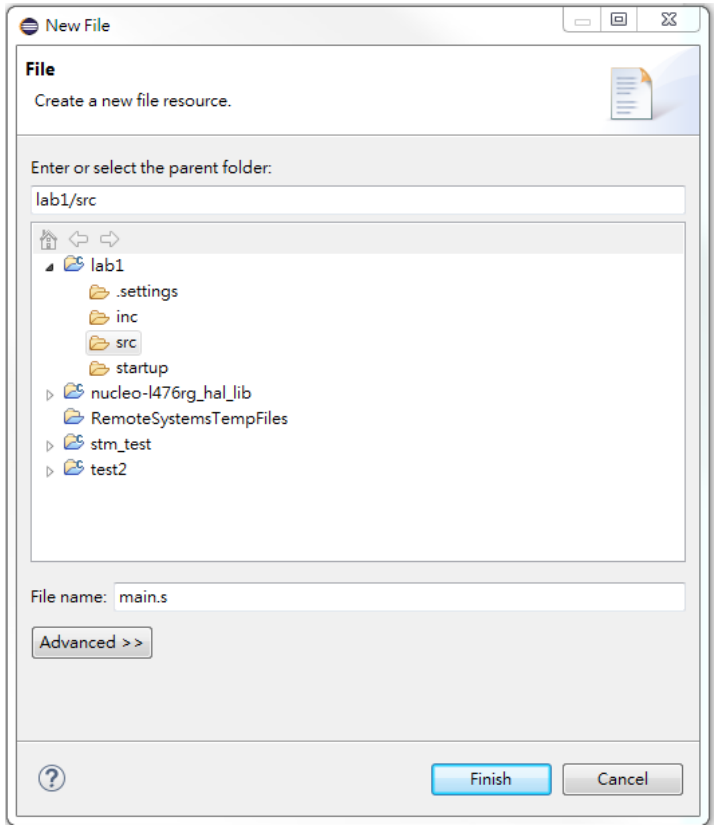
Project Files

- Then you can see the project files in the ‘Project Explorer’ list
- It contain the board startup code ‘**startup_stm32.s**’ and linker script ‘**LinderScript.ld**’



Create File

- Right click the lab1/src folder and create a file call **‘main.s’**



Write Your First Code

Use UAL syntax

Text section start point

Define global symbol

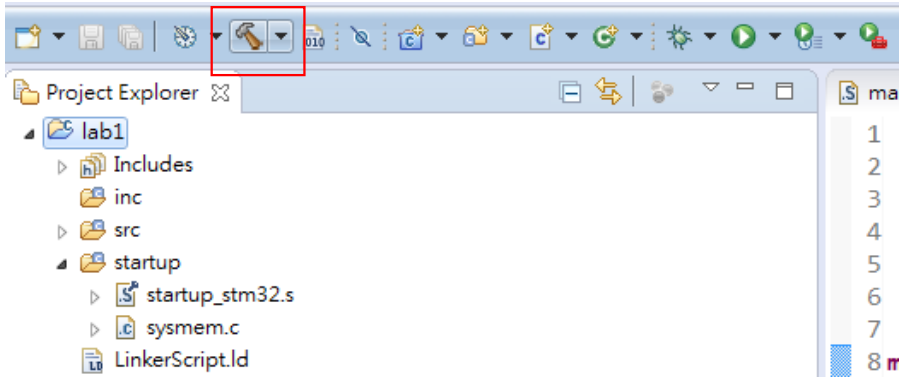
Define a constant variable ‘AA’

```
.syntax unified
.cpu cortex-m4
.thumb

.text
.global main
.equ AA, 0x5566

main:
    movs r0, #AA
    movs r1, #20
    adds r2, r0, r1
    B main
```

main.s

Build Code

- Write your first code
- Project->Build all




```
    .syntax unified
    .cpu cortex-m4
    .thumb

    .text
    .global main
    .equ AA, 0x5566

main:
    movs r0, #AA
    movs r1, #20
    adds r2, r0, r1
    B main
```

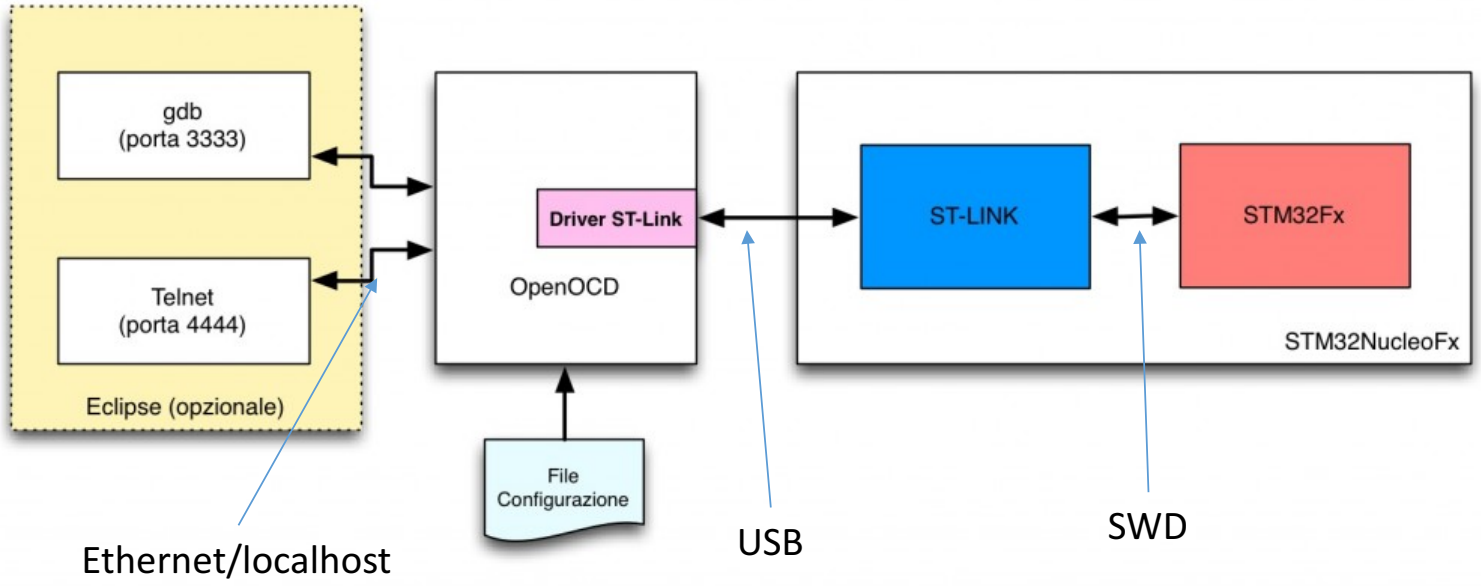
Is it your program entry point?

```
'Building target: lab1.elf'
'Invoking: MCU GCC Linker'
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16
'Finished building target: lab1.elf'
' '
make --no-print-directory post-build
'Generating binary and Printing size information:'
arm-none-eabi-objcopy -O binary "lab1.elf" "lab1.bin"
arm-none-eabi-size "lab1.elf"
   text    data     bss     dec     hex filename
    992    1080    1056    3128     c38 lab1.elf
' '
```

Build result

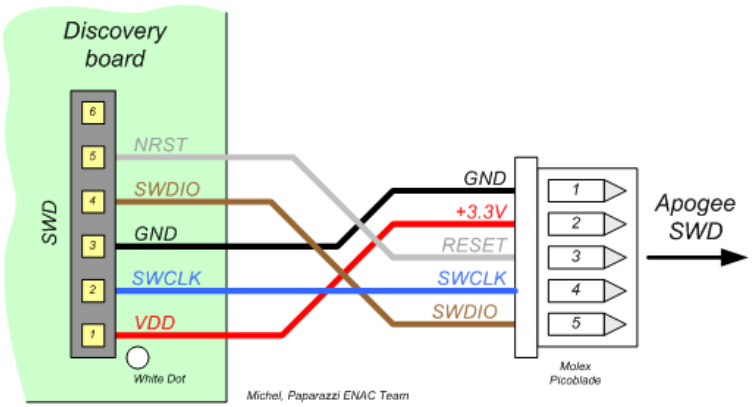
Debug

- ST-Link: A STM32 hardware flasher and debugger
- OpenOCD: An open source GDB server

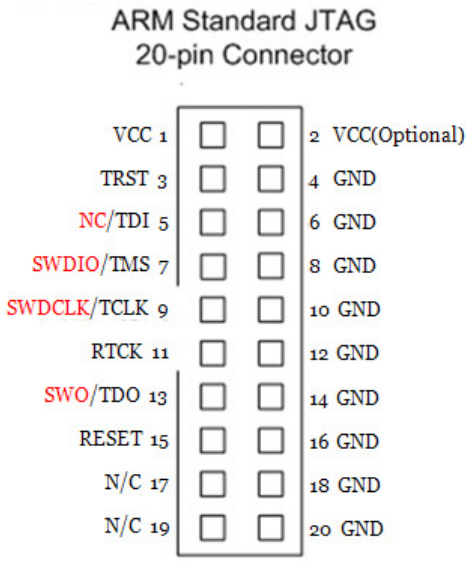


Debug Interface

- JTAG(Joint Test Action Group)
  - A standard ASICs hardware debug interface
- SWD(Serial Wire Debug)
  - Only use 5 wires from part of JTAG interface

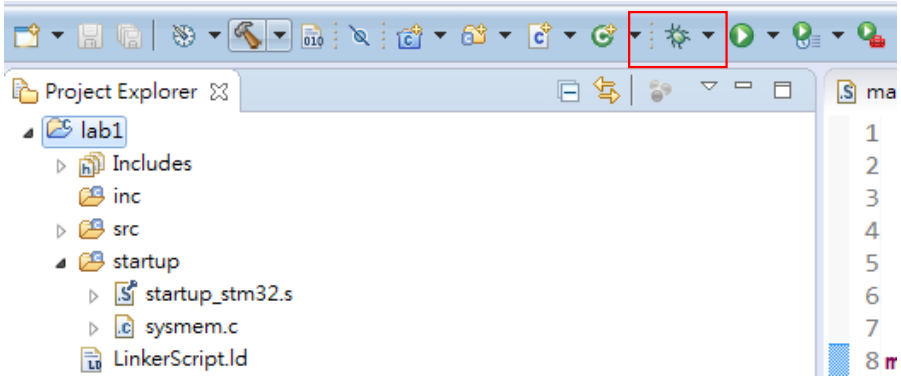


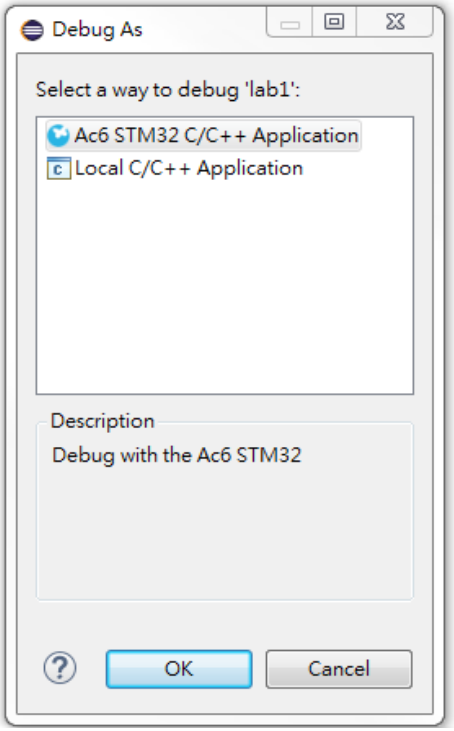
ARM Standard JTAG Connector(20-pins)



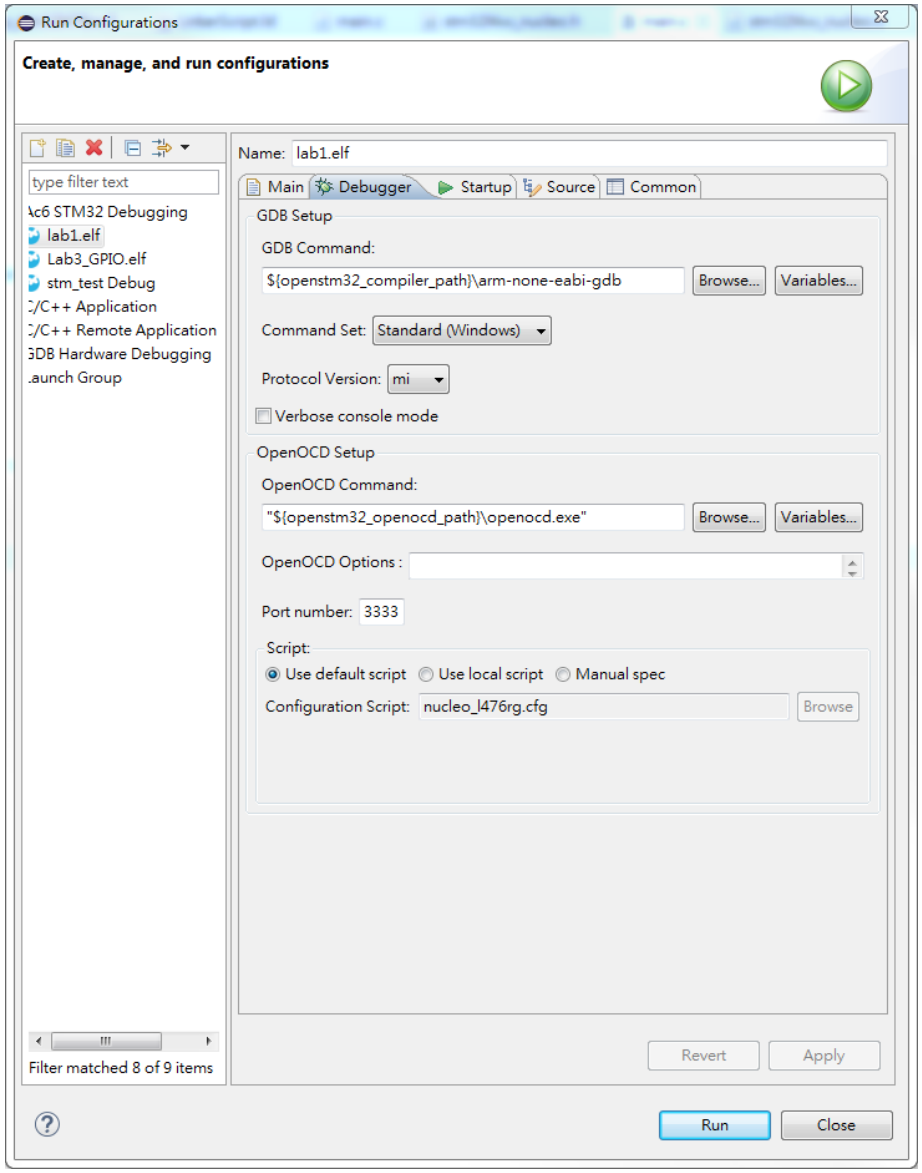
Create a debug configure

- Run->Debug
- Debug as ‘AC6 STM32 C/C++ Application’

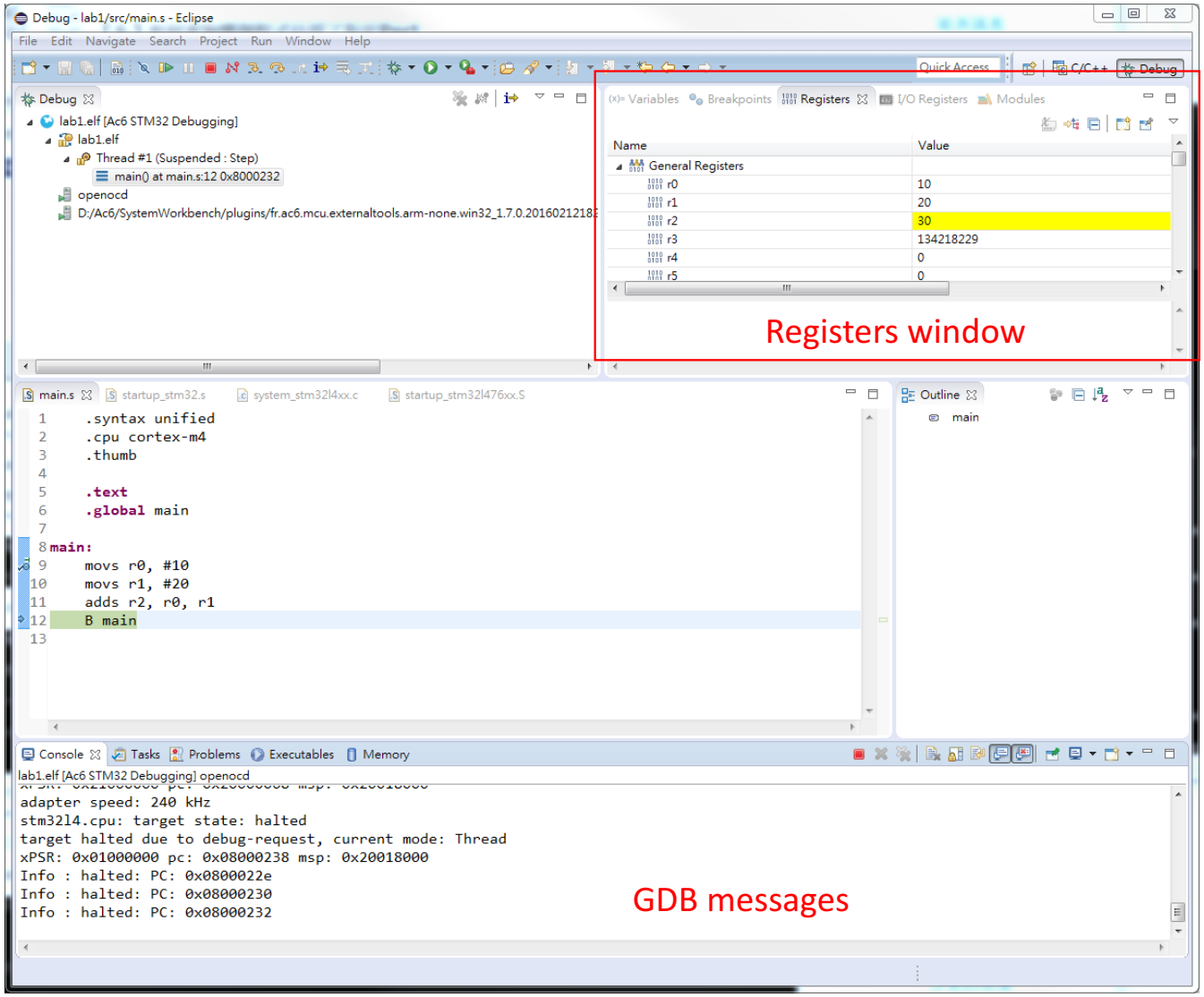




- Check your debugger configuration
- Run -> Debug Configuration

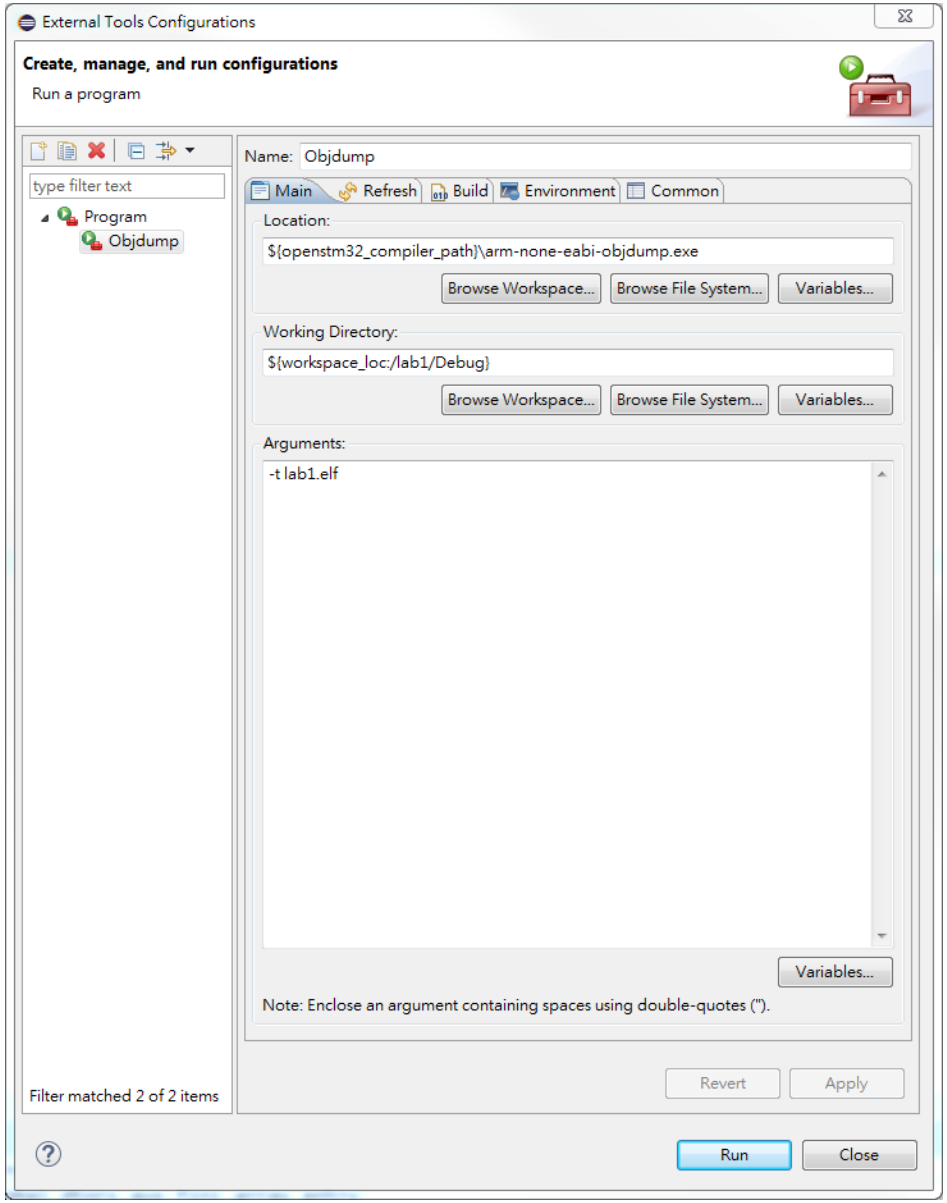


- By default the GDB will set the first breakpoint at ‘main’
- Press ‘Step into’ button or ‘F5’ will debug your code step by step.



Object Dump

- This tool can help you show the program's symbol table
- Run->External Tool-> External Tool Configurations
- Objdump usage guide
  - https://sourceware.org/binutils/docs/binutils/objdump.html



Symbol Table

Problems Tasks Console Properties Progress

```
<terminated> Objdump [Program] D:\Ac6\SystemWorkbench\plugins\fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829\tools\compiler\
080001a8 l     F .text   00000000 register_tm_clones
080001cc l     F .text   00000000 __do_global_dtors_aux
20000440 l       .bss    00000000 completed.6516
080003f8 l     O .fini_array    00000000 __do_global_dtors_aux_fini_array_entry
080001f4 l     F .text   00000000 frame_dummy
20000444 l       .bss    00000000 object.6521
080003f4 l     O .init_array    00000000 __frame_dummy_init_array_entry
00000000 l    df *ABS*   00000000 src/main.o
20000000 l       .data   00000000 X
20000004 l       .data   00000000 str
00000055 l       *ABS*   00000000 AA
0800023a l       .text   00000000 L
00000000 l    df *ABS*   00000000 init.c
00000000 l    df *ABS*   00000000 __call_atexit.c
080002e0 l     F .text   00000014 register_fini
00000000 l    df *ABS*   00000000 atexit.c
00000000 l    df *ABS*   00000000 fini.c
00000000 l    df *ABS*   00000000 __atexit.c
```

Symbol address

Section locate

Symbol name

Memory Access

- Define data variable
- Direct access
- Indirect read access

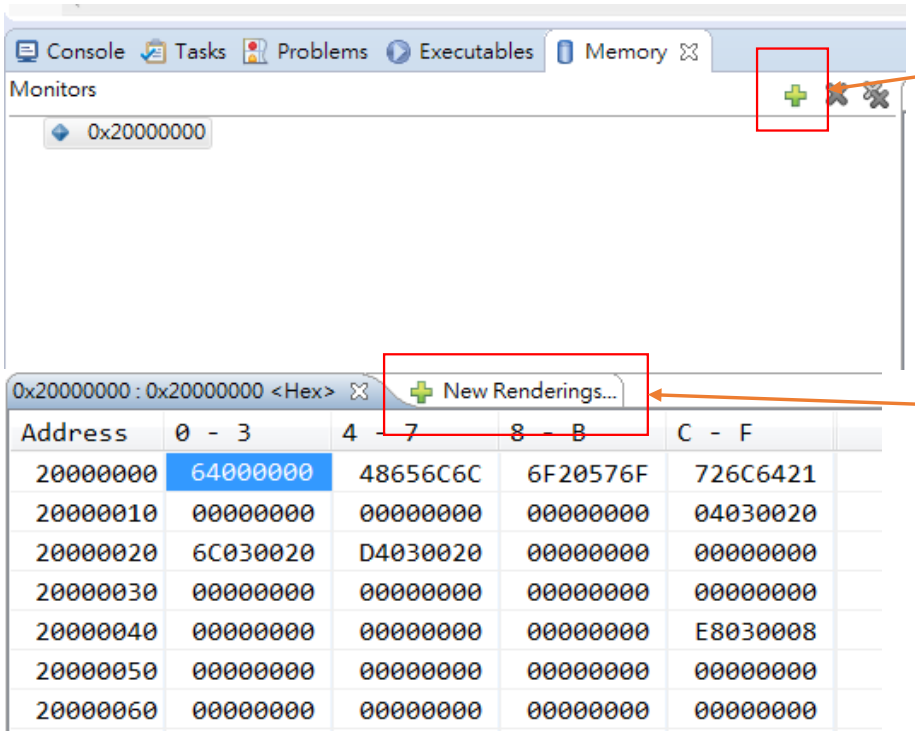
```
1      .syntax unified
2      .cpu cortex-m4
3      .thumb
4
5  .data
6      X: .word 100
7      str: .asciz "Hello World!"
8  .text
9      .global main
10     .equ AA, 0x55
11
12 main:
13     ldr r1, =X
14     ldr r0, [r1]
15     movs r2, #AA
16     adds r2, r2, r0
17     str  r2, [r1]
18
19     ldr  r1, =str
20     ldr  r2, [r1]
21 L:  B L
22
```

Data section start point

Write the data register into memory

Memory Monitors

- That can help you watch the memory content



Press it to add a memory monitor

Press “New Renderings” can change the display format

Reference

- Getting started with STM32 Nucleo board software development tools
  - http://www.st.com/content/ccc/resource/technical/document/user_manual/1b/03/1b/b4/88/20/4e/cd/DM00105928.pdf/files/DM00105928.pdf/jcr:content/translations/en.DM00105928.pdf
- STM32 Nucleo-64 boards user manual
  - http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

Linker Script

- https://www.math.utah.edu/docs/info/ld_toc.html#SEC4