MCSL2016

Lab1

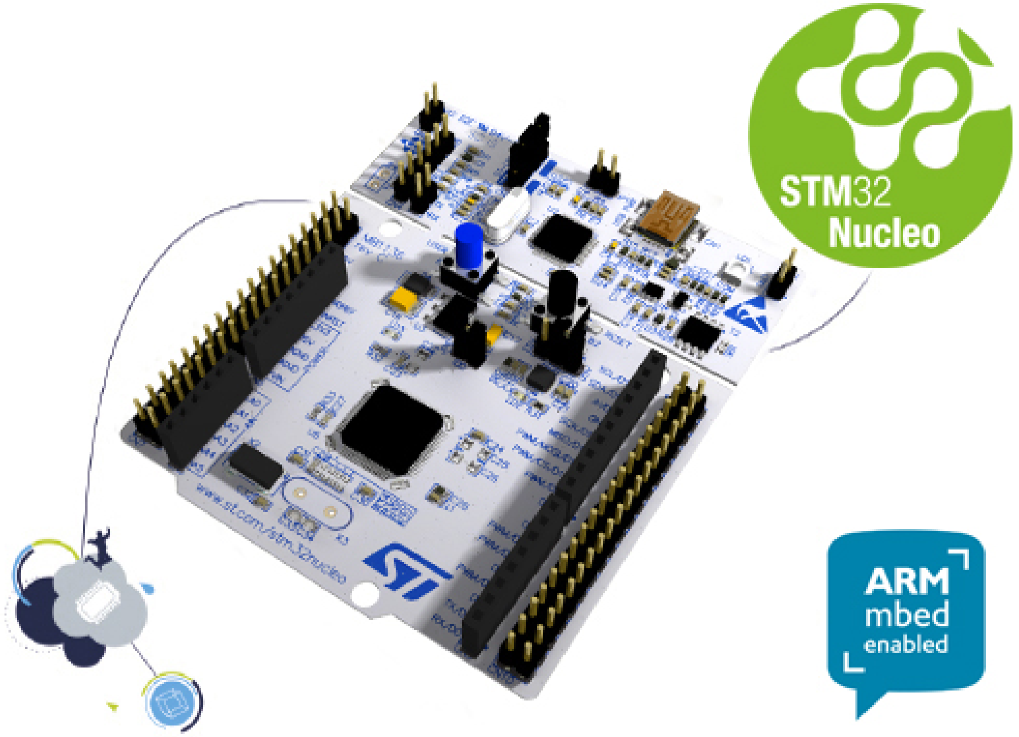
By Prof. Shiao-Li Tsao
NCTU CS 2016
Modified by Yun-Chien Cheng
NCTU ME 2017

一些想法

- 工程師要自己閱讀user manual，自己上網找答案解bug。所以每個bug都是學習的機會，自己解掉一個內力就強一分。
- ARM比8051難上手，從IDE到reg到port都是，但這是因為他功能更加強大，泛用性更高。
- 位址，ptr 一定要搞懂。

STM32 Nucleo Board

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



Hardware Block

Figure 3. Top layout

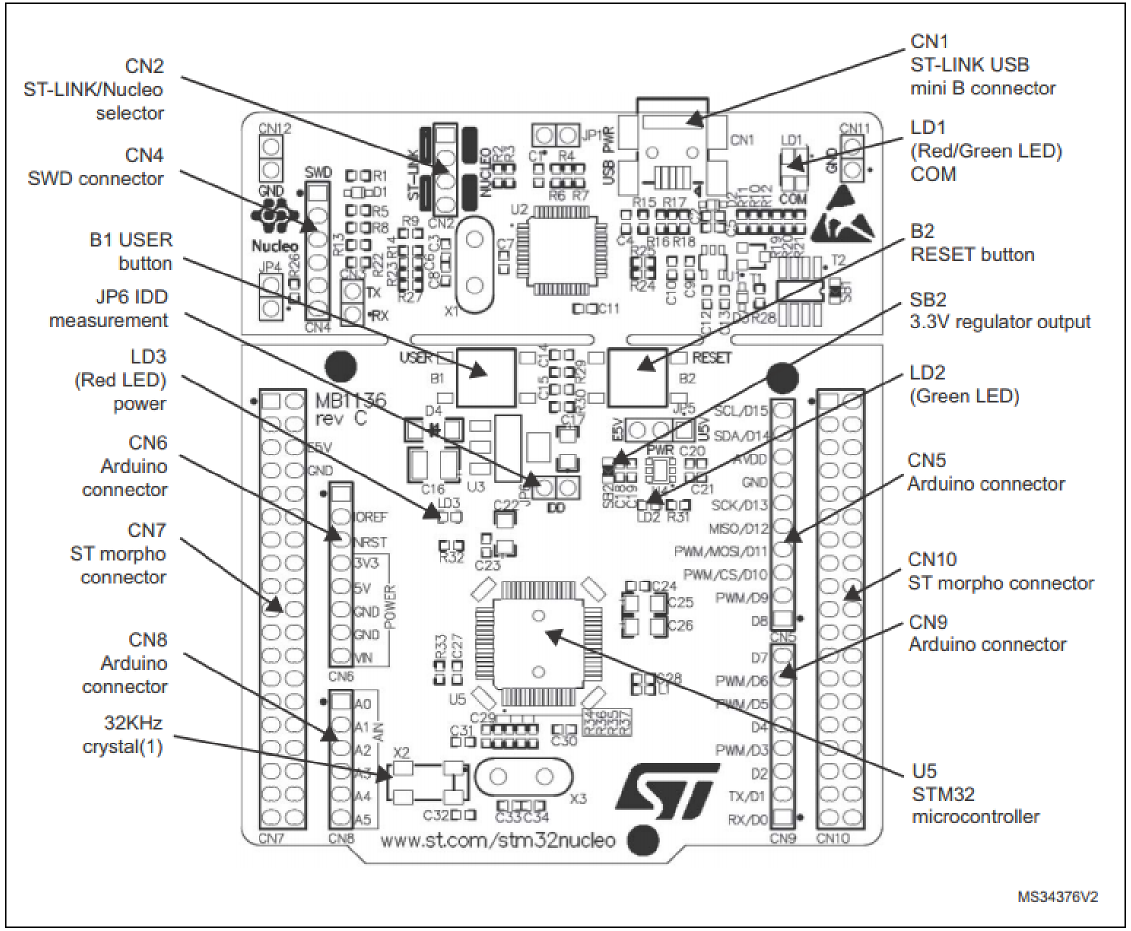
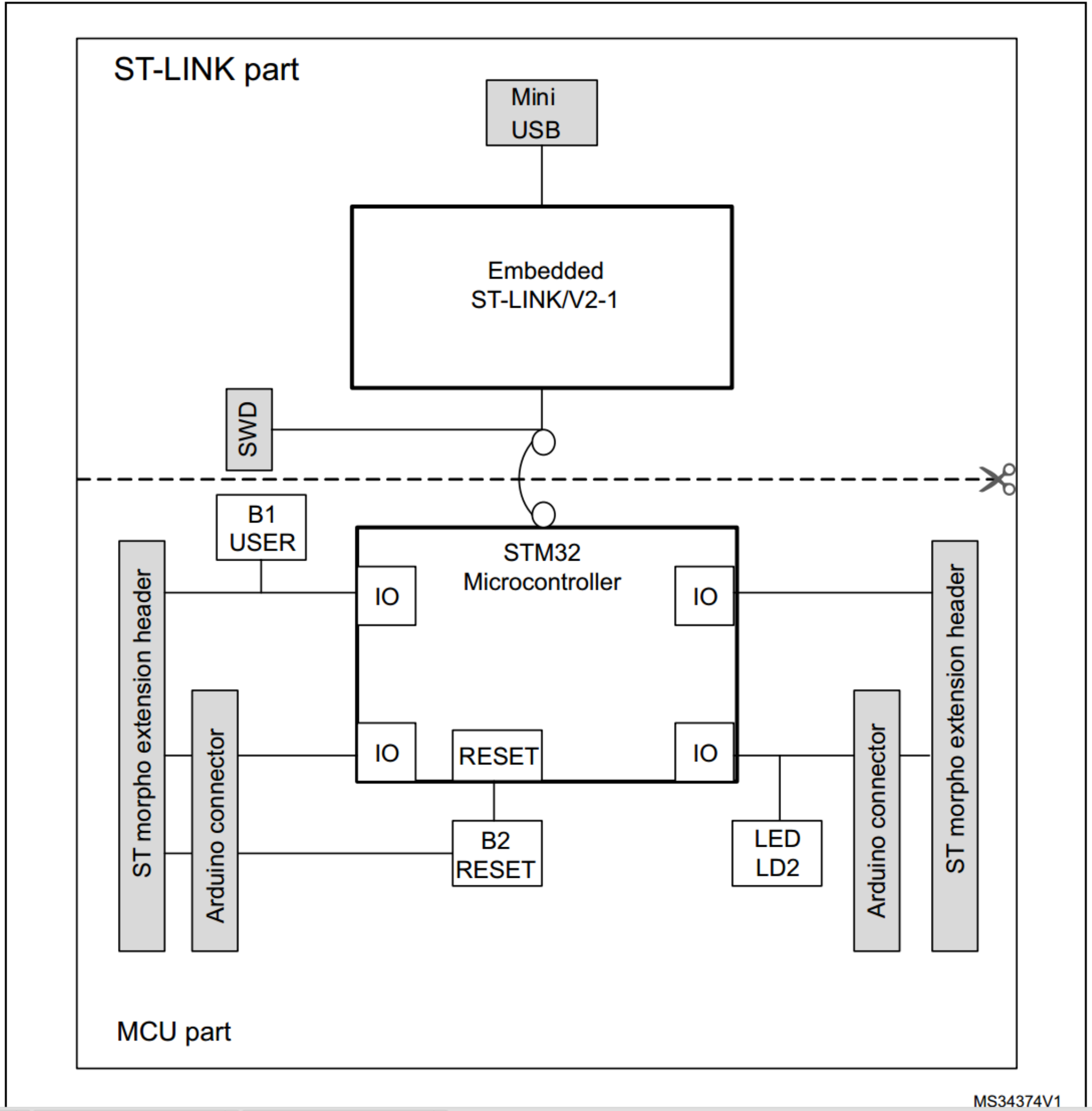
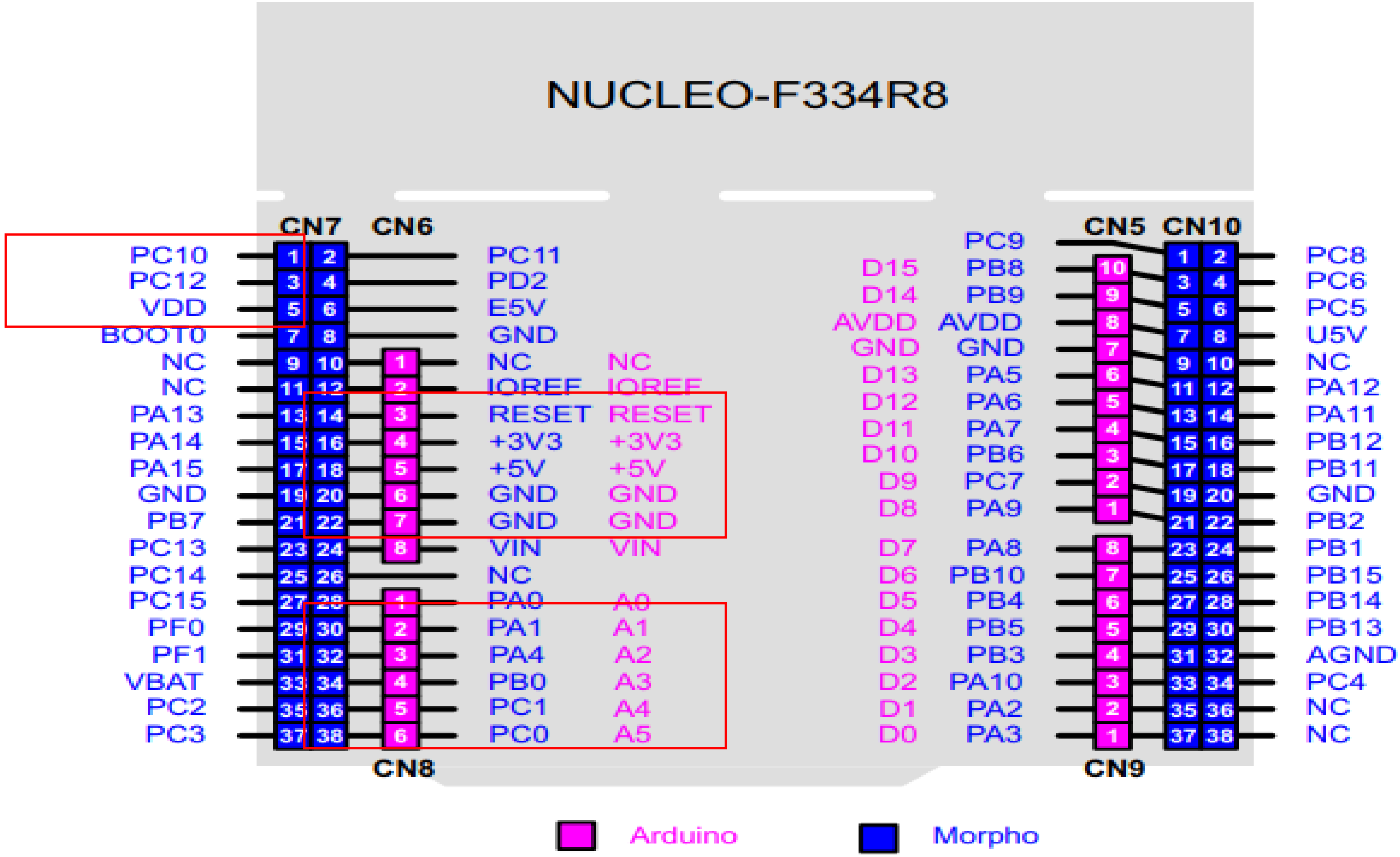


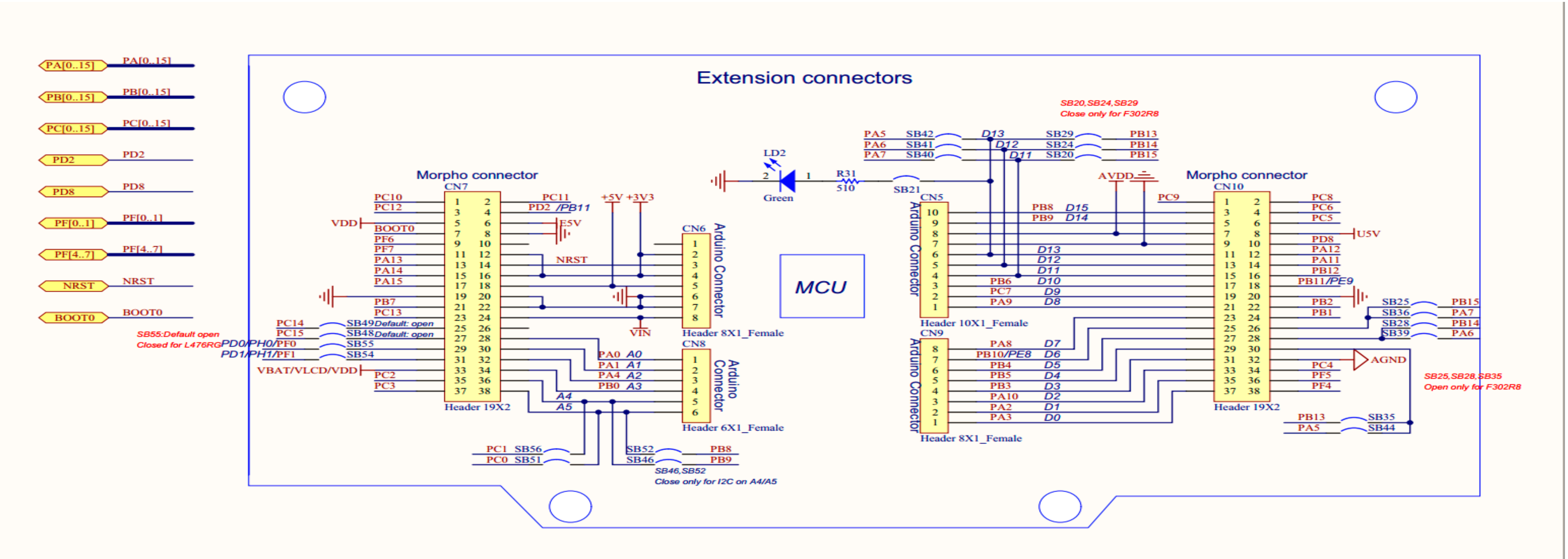
Figure 2. Hardware block diagram




NUCLEO-F334R8
CN7
CN6
CN5
CN10
PC10 1 2 PC11
PC12 3 4 PD2
VDD 5 6 E5V
BOOT0 7 8 GND
NC 9 10 1 NC NC
NC 11 12 2 IOREF IOREF
PA13 13 14 3 RESET RESET
PA14 15 16 4 +3V3 +3V3
PA15 17 18 5 +5V +5V
GND 19 20 6 GND GND
PB7 21 22 7 GND GND
PC13 23 24 8 VIN VIN
PC14 25 26 NC
PC15 27 28 1 PA0 A0
PF0 29 30 2 PA1 A1
PF1 31 32 3 PA4 A2
VBAT 33 34 4 PB0 A3
PC2 35 36 5 PC1 A4
PC3 37 38 6 PC0 A5
CN8
PC9 1 2 PC8
D15 PB8 10 3 4 PC6
D14 PB9 9 5 6 PC5
AVDD AVDD 8 7 8 U5V
GND GND 7 9 10 NC
D13 PA5 6 11 12 PA12
D12 PA6 5 13 14 PA11
D11 PA7 4 15 16 PB12
D10 PB6 3 17 18 PB11
D9 PC7 2 19 20 GND
D8 PA9 1 21 22 PB2
D7 PA8 8 23 24 PB1
D6 PB10 7 25 26 PB15
D5 PB4 6 27 28 PB14
D4 PB5 5 29 30 PB13
D3 PB3 4 31 32 AGND
D2 PA10 3 33 34 PC4
D1 PA2 2 35 36 NC
D0 PA3 1 37 38 NC
CN9
Arduino
Morpho

Nucleo Board Extension Connector

- 用於連接GPIO與外部電路
- 同學可參考Reference manual了解內部連接方式

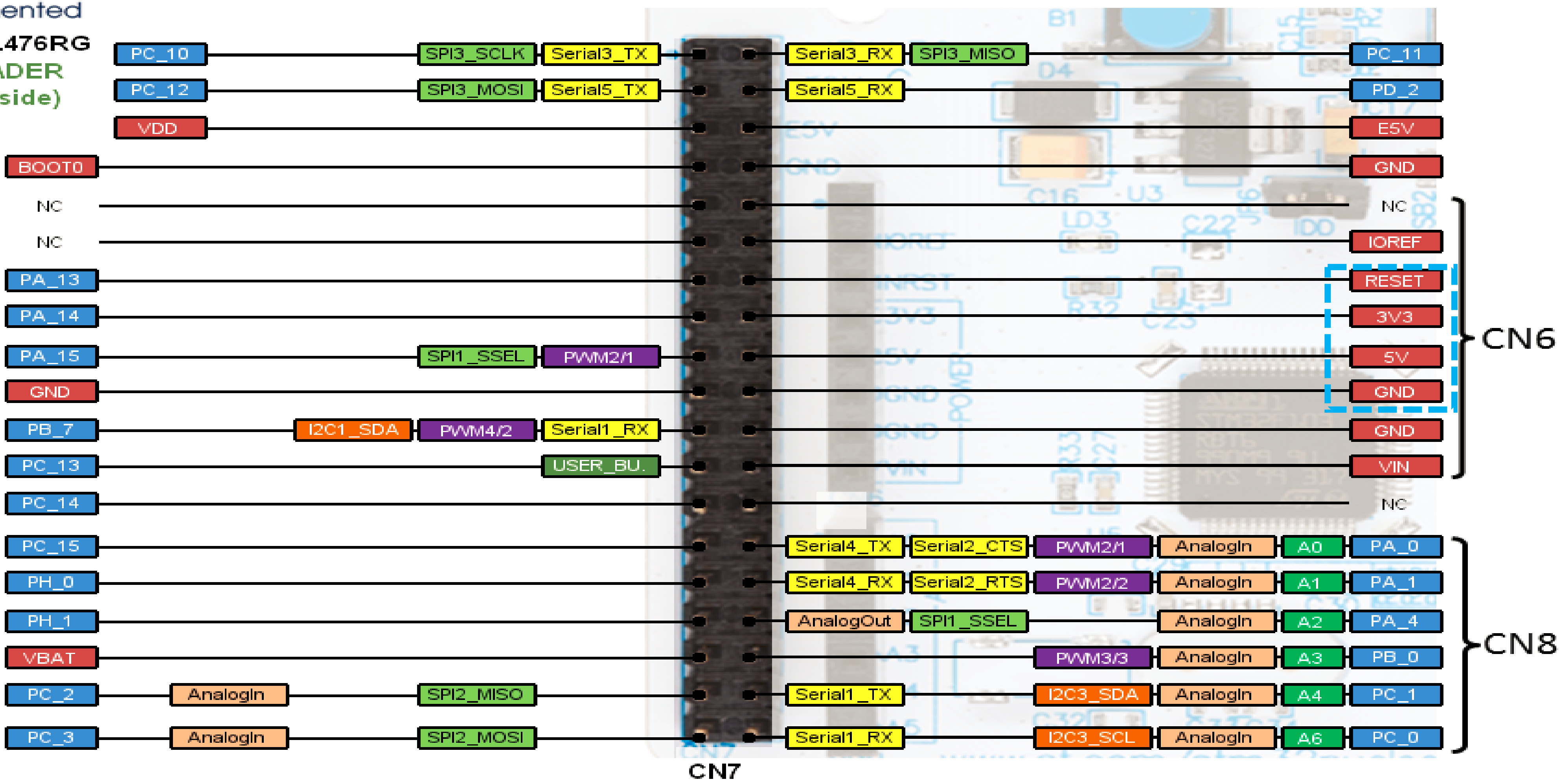


Arduino和Morpho connector有些是相通(short)的

life.augmented

NUCLEO-L476RG

CN7 HEADER (top left side)



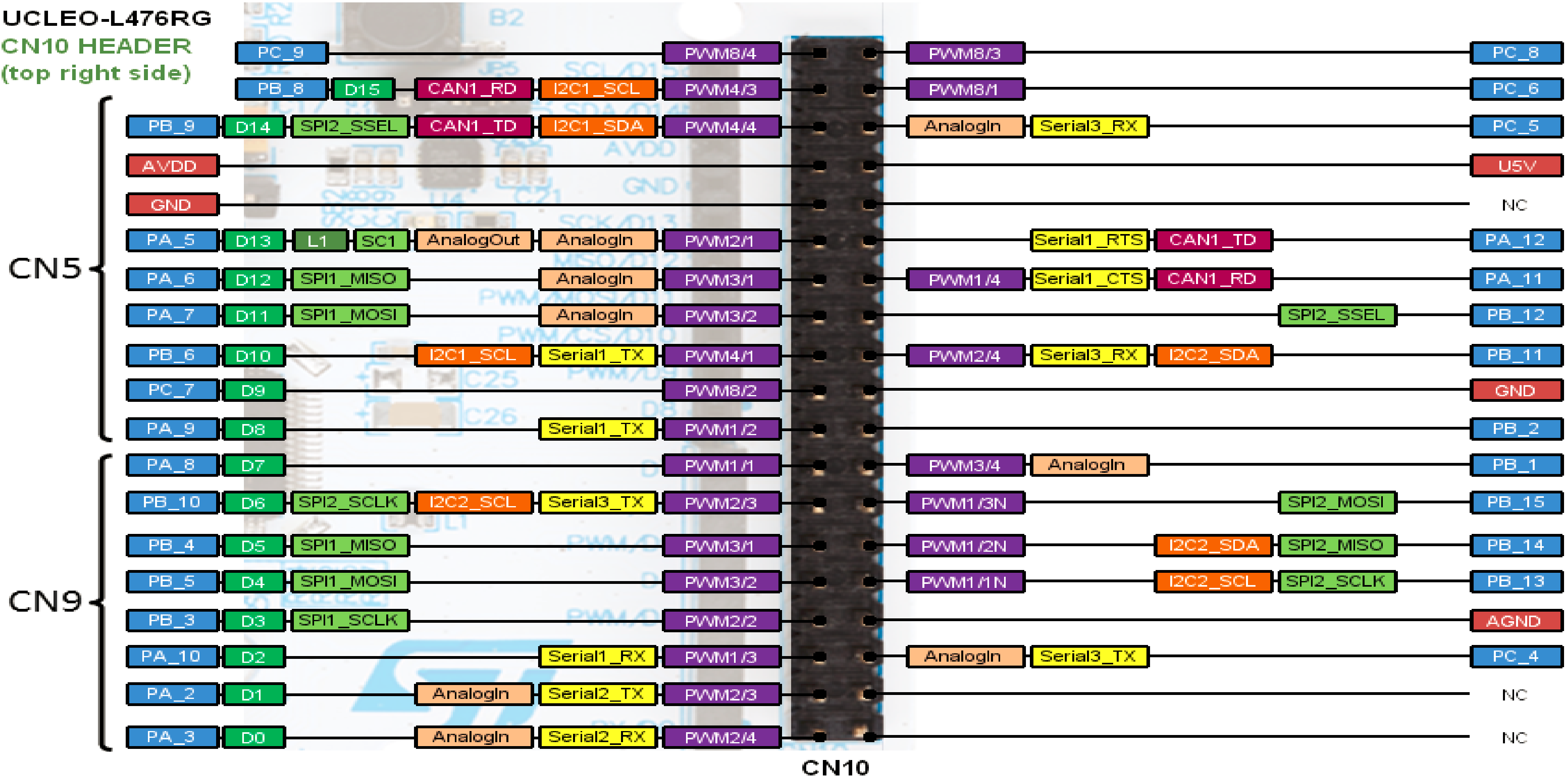
PC_10		SPI3_SCLK	Serial3_TX	Serial3_RX	SPI3_MISO				PC_11
PC_12		SPI3_MOSI	Serial5_TX	Serial5_RX					PD_2
VDD									E5V
BOOT0									GND
NC									NC
NC									IOREF
PA_13									RESET
PA_14									3V3
PA_15		SPI1_SSEL	PWM2/1						5V
GND									GND
PB_7	I2C1_SDA	PWM4/2	Serial1_RX						GND
PC_13			USER_BU.						VIN
PC_14									NC
PC_15				Serial4_TX	Serial2_CTS	PWM2/1	AnalogIn	A0	PA_0
PH_0				Serial4_RX	Serial2_RTS	PWM2/2	AnalogIn	A1	PA_1
PH_1				AnalogOut	SPI1_SSEL		AnalogIn	A2	PA_4
VBAT						PWM3/3	AnalogIn	A3	PB_0
PC_2	AnalogIn	SPI2_MISO		Serial1_TX		I2C3_SDA	AnalogIn	A4	PC_1
PC_3	AnalogIn	SPI2_MOSI		Serial1_RX		I2C3_SCL	AnalogIn	A6	PC_0

life.augmented

NUCLEO-L476RG

CN10 HEADER

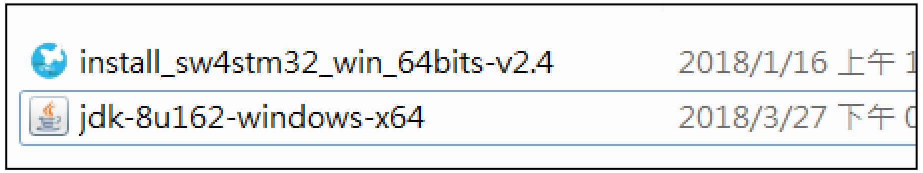
(top right side)



PC_9					PWM8/4	PWM8/3				PC_8
PB_8	D15		CAN1_RD	I2C1_SCL	PWM4/3	PWM8/1				PC_6
PB_9	D14	SPI2_SSEL	CAN1_TD	I2C1_SDA	PWM4/4	AnalogIn	Serial3_RX			PC_5
AVDD										U5V
GND										NC
PA_5	D13	L1 SC1	AnalogOut	AnalogIn	PWM2/1		Serial1_RTS	CAN1_TD		PA_12
PA_6	D12	SPI1_MISO		AnalogIn	PWM3/1	PWM1/4	Serial1_CTS	CAN1_RD		PA_11
PA_7	D11	SPI1_MOSI		AnalogIn	PWM3/2				SPI2_SSEL	PB_12
PB_6	D10		I2C1_SCL	Serial1_TX	PWM4/1	PWM2/4	Serial3_RX	I2C2_SDA		PB_11
PC_7	D9				PWM8/2					GND
PA_9	D8			Serial1_TX	PWM1/2					PB_2
PA_8	D7				PWM1/1	PWM3/4	AnalogIn			PB_1
PB_10	D6	SPI2_SCLK	I2C2_SCL	Serial3_TX	PWM2/3	PWM1/3N			SPI2_MOSI	PB_15
PB_4	D5	SPI1_MISO			PWM3/1	PWM1/2N		I2C2_SDA	SPI2_MISO	PB_14
PB_5	D4	SPI1_MOSI			PWM3/2	PWM1/1N		I2C2_SCL	SPI2_SCLK	PB_13
PB_3	D3	SPI1_SCLK			PWM2/2					AGND
PA_10	D2			Serial1_RX	PWM1/3	AnalogIn	Serial3_TX			PC_4
PA_2	D1		AnalogIn	Serial2_TX	PWM2/3					NC
PA_3	D0		AnalogIn	Serial2_RX	PWM2/4					NC

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



install_sw4stm32_win_64bits-v2.4 2018/1/16 上午 1

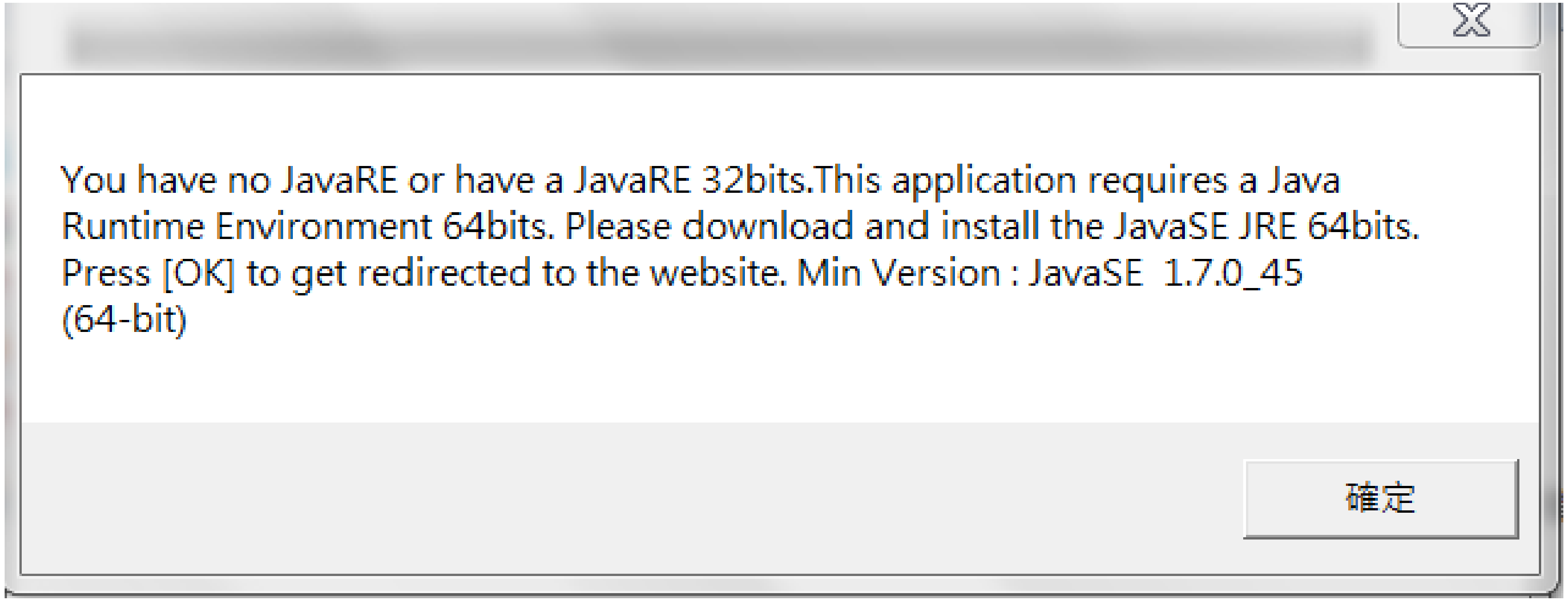
jdk-8u162-windows-x64 2018/3/27 下午

SW4STM32

- Download from http://www.openstm32.org/
- Windows 7 (此為舊版，請到網址中下載符合你win版本的最新版，不然跑出來會有bug)
    - 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

jdk-8u111-windows-x64

- 若安裝時出現以下對話框

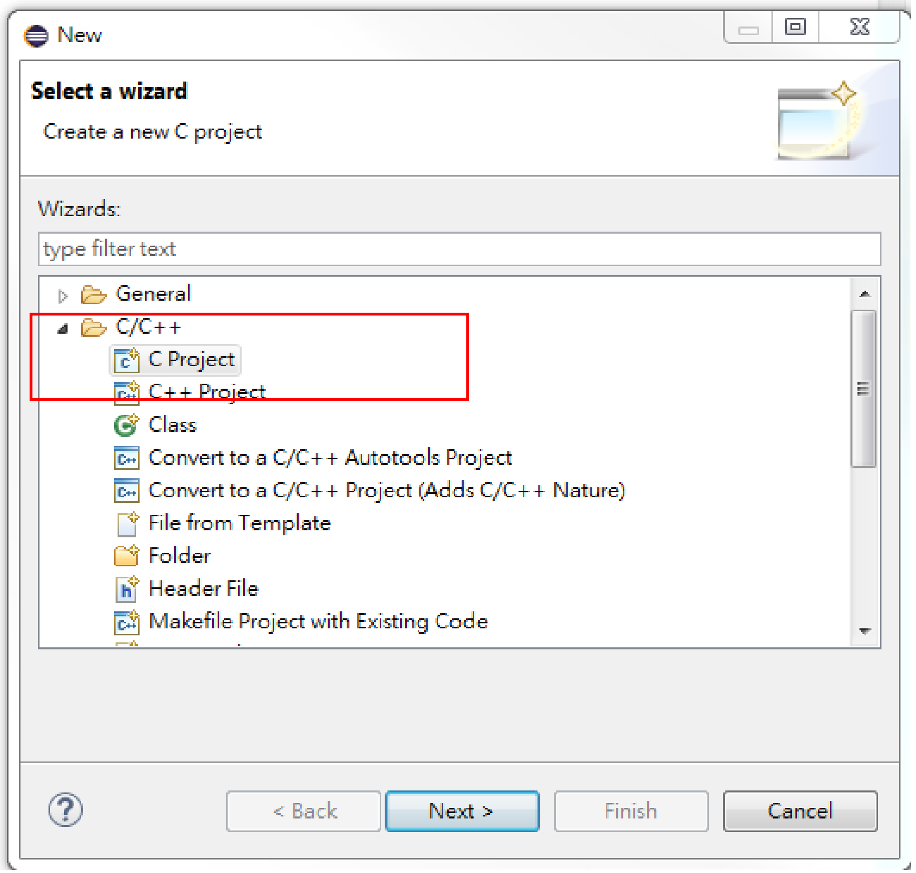


You have no JavaRE or have a JavaRE 32bits.This application requires a Java Runtime Environment 64bits. Please download and install the JavaSE JRE 64bits. Press [OK] to get redirected to the website. Min Version : JavaSE 1.7.0_45 (64-bit)

- 請到http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html 下載適合你電腦的檔案安裝

Create Project

- Create a ‘lab1’ project(檔名不要有空白, 不然可能會有問題)



New

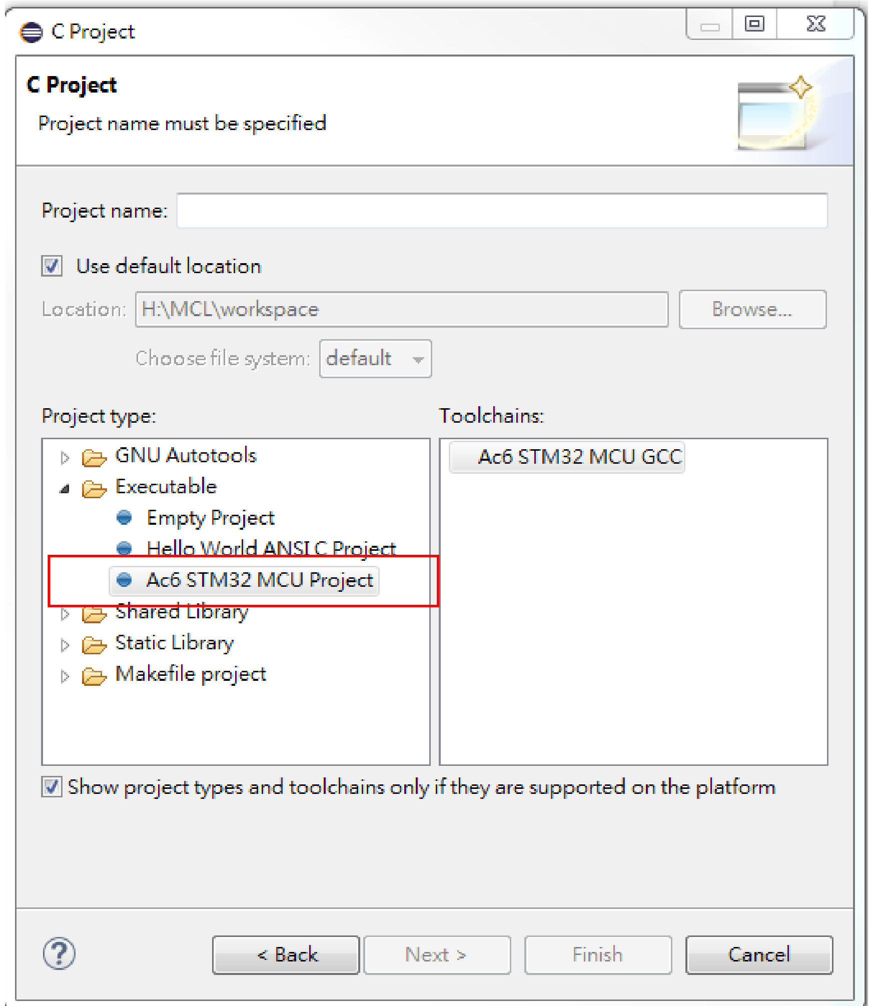
Select a wizard

Create a new C project

Wizards:

type filter text

General
C/C++
C Project
C++ Project
Class
Convert to a C/C++ Autotools Project
Convert to a C/C++ Project (Adds C/C++ Nature)
File from Template
Folder
Header File
Makefile Project with Existing Code
?
< Back
Next >
Finish
Cancel



C Project

C Project

Project name must be specified

Project name:

☑ Use default location

Location: H:\MCL\workspace

Choose file system: default

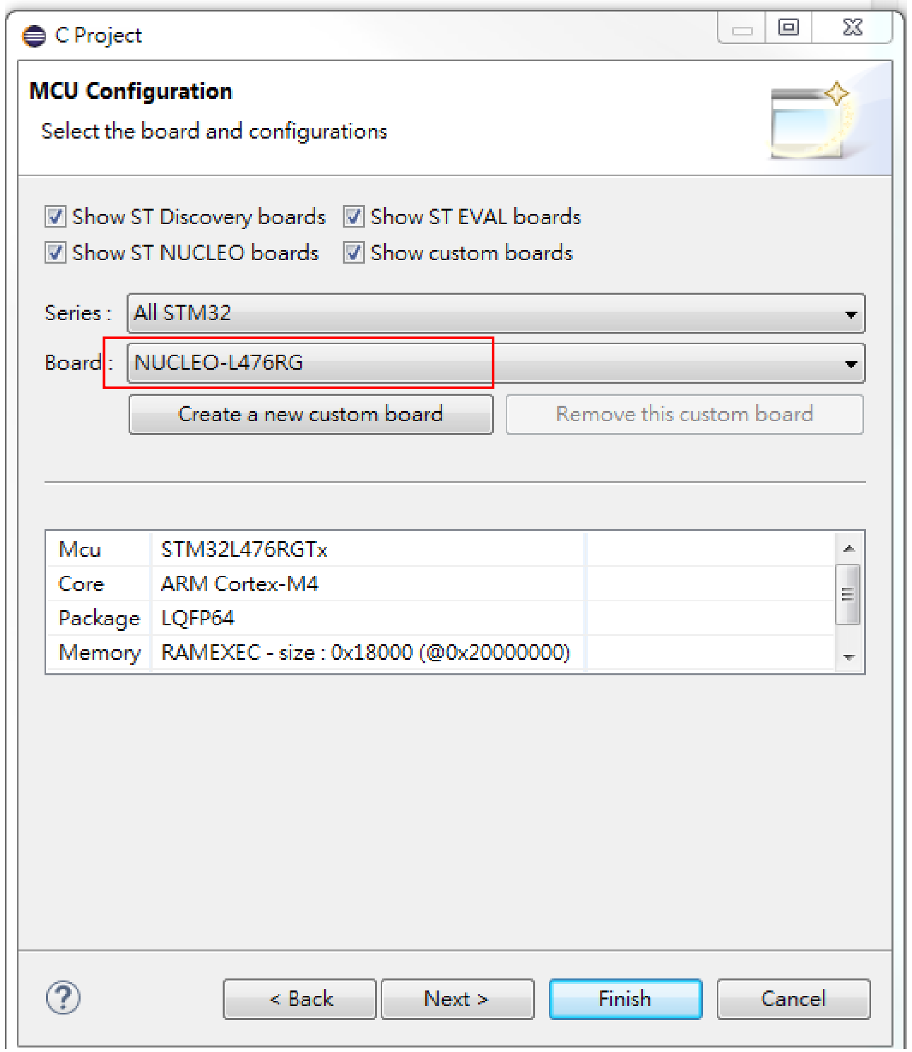
Project type:

Toolchains: Ac6 STM32 MCU GCC

☑ Show project types and toolchains only if they are supported on the platform

MCU Configuration

- Select NUCLEO-L476RG board



MCU Configuration

Select the board and configurations

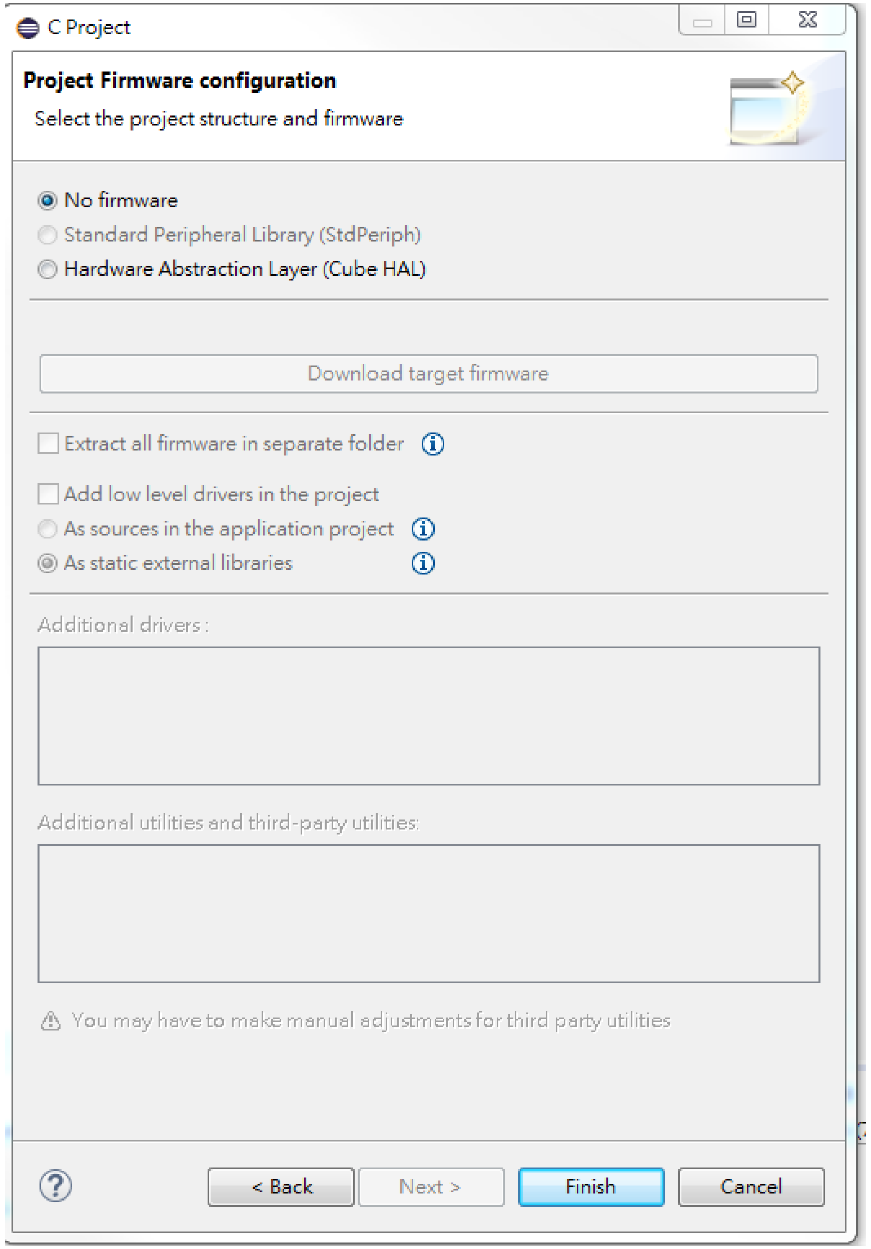
☑ Show ST Discovery boards ☑ Show ST EVAL boards
☑ Show ST NUCLEO boards ☑ Show custom boards

Series : All STM32

Board : NUCLEO-L476RG

Mcu	STM32L476RGTx
Core	ARM Cortex-M4
Package	LQFP64
Memory	RAMEXEC - size : 0x18000 (@0x20000000)

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



C Project

Project Firmware configuration

Select the project structure and firmware

● No firmware
○ Standard Peripheral Library (StdPeriph)
○ Hardware Abstraction Layer (Cube HAL)

□ Extract all firmware in separate folder

□ Add low level drivers in the project
○ As sources in the application project
● As static external libraries

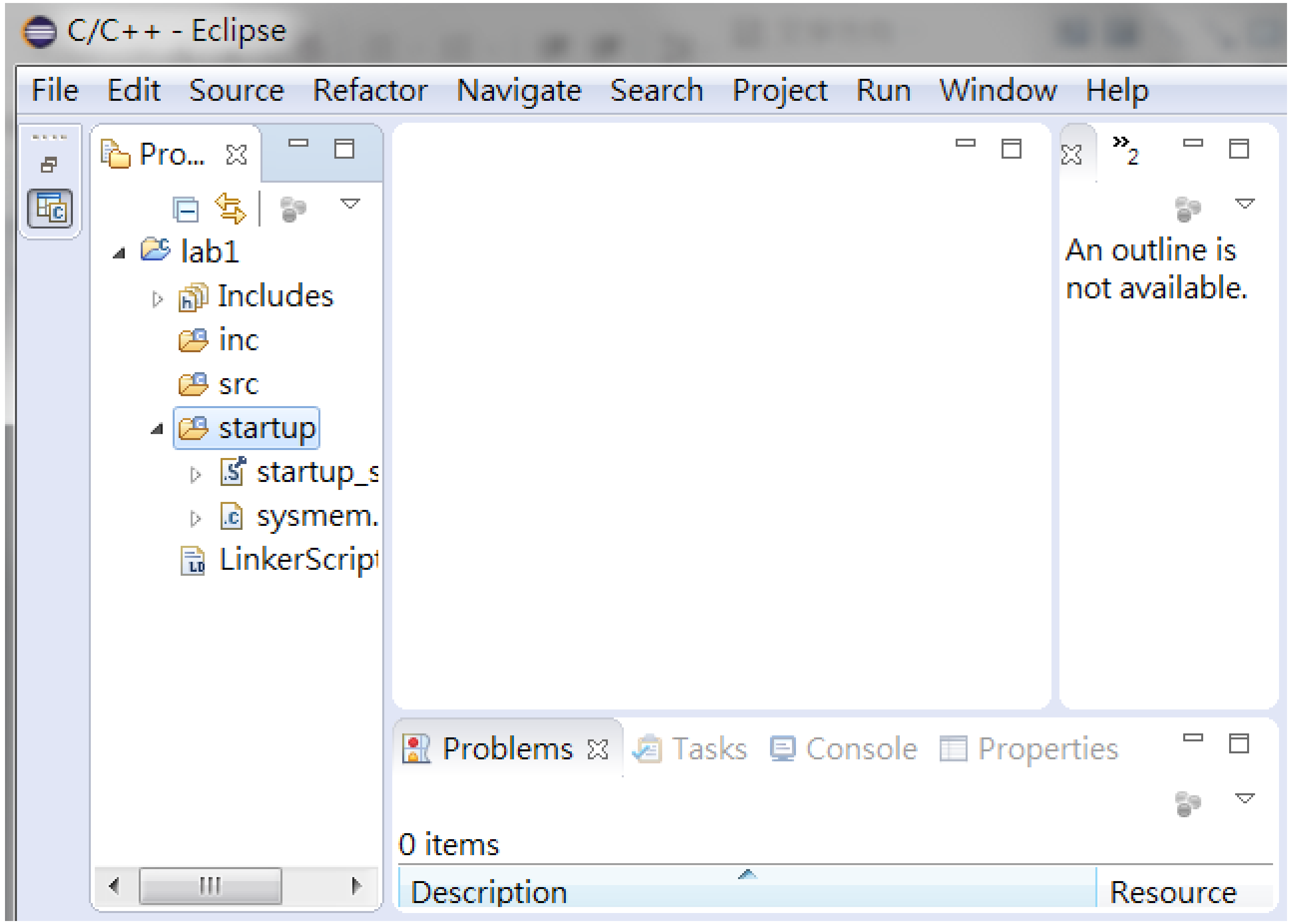
Additional drivers :

Additional utilities and third-party utilities:

You may have to make manual adjustments for third party utilities

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**’

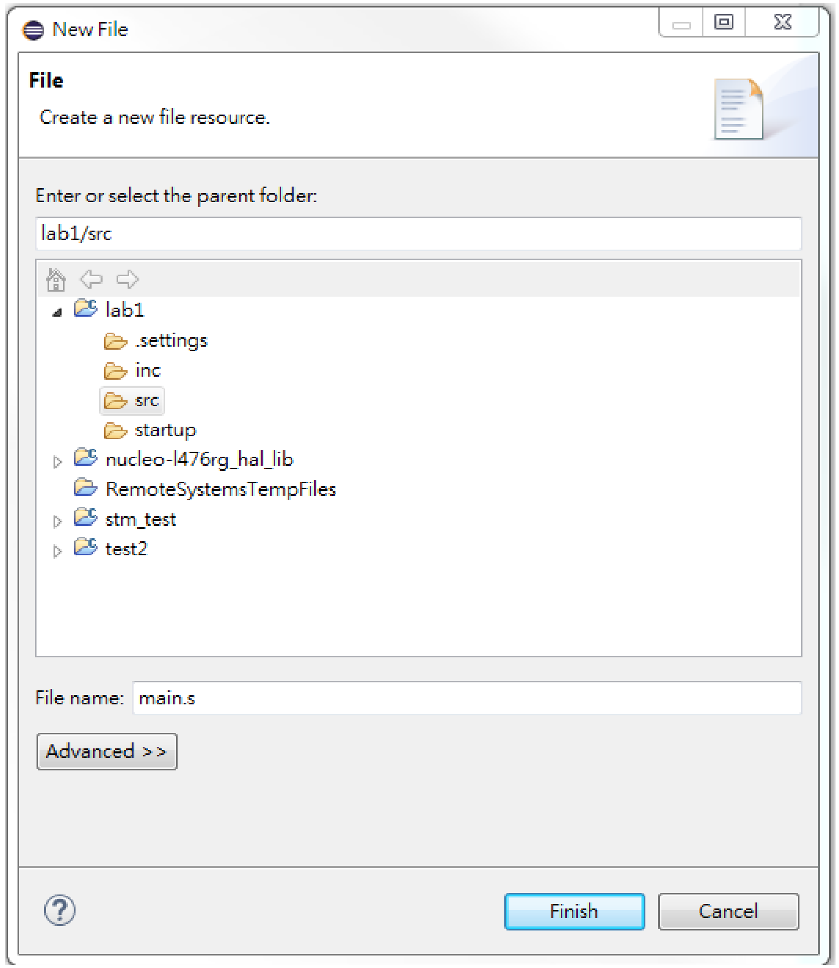


An outline is not available.

0 items

Create File

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



File

Create a new file resource.

Enter or select the parent folder: lab1/src

File name: main.s

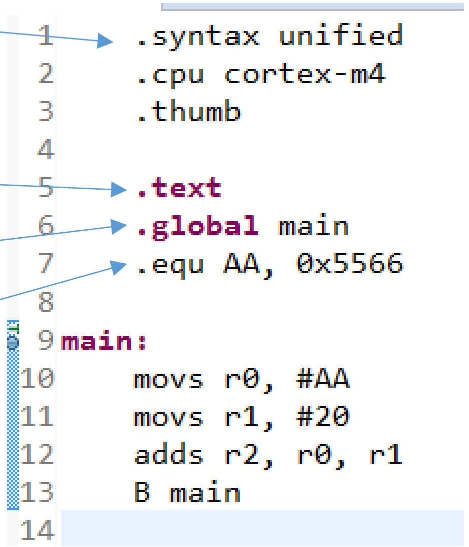
Write Your First Code

Use UAL syntax

Text section start point

Define global symbol

Define a constant symbol ‘AA’



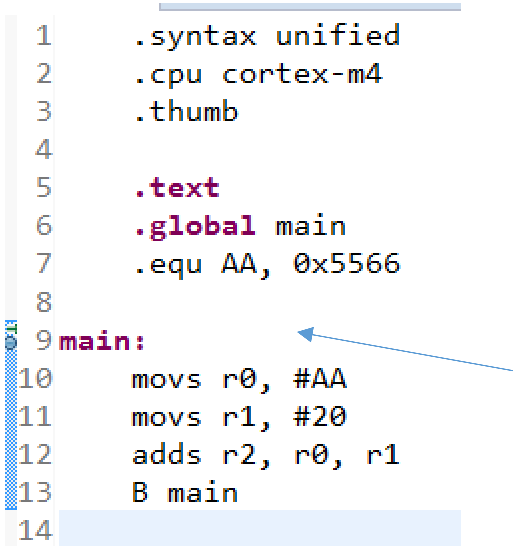
```
    .syntax unified
    .cpu cortex-m4
    .thumb

.text
.global main
.equ AA, 0x5566

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

```

main.s

0x5566 數值太大，改成0x55

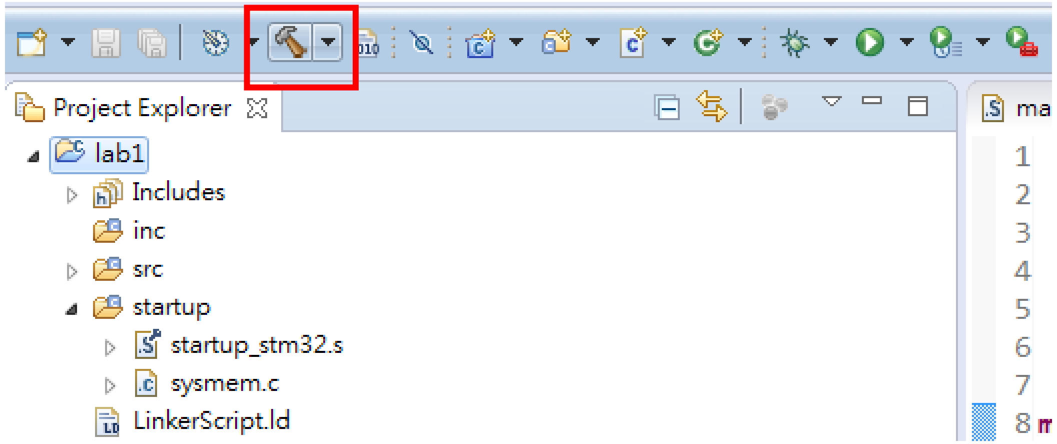
Build Code

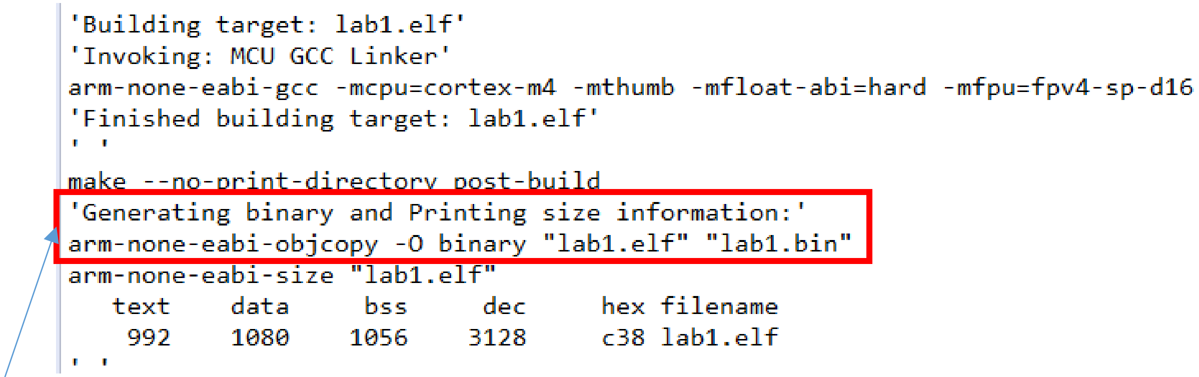
- Write your first code
- Project->Build all



```
	.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

```

Main entry point.





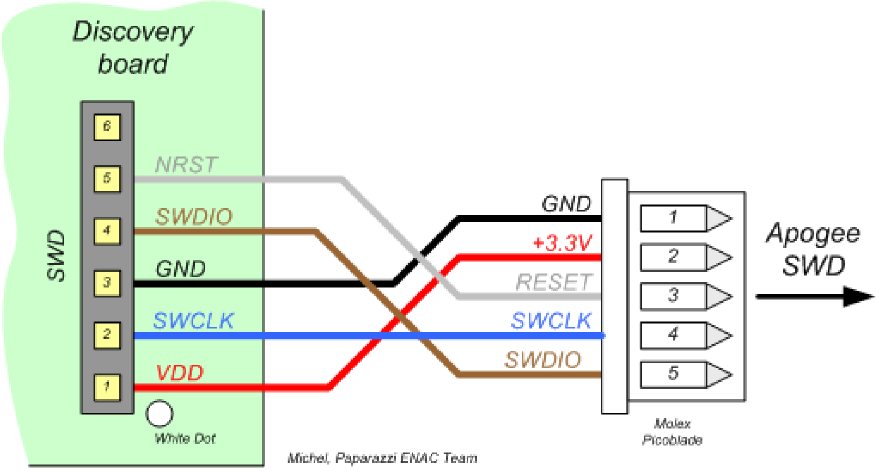
```
'Building target: lab1.elf'
'Invoking: MCU GCC Linker'
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16
'Finished building target: lab1.elf'
' '
make --no-print-directory post-build
'Generating binary and Printing size information:'
arm-none-eabi-objcopy -O binary "lab1.elf"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

Create the target image file

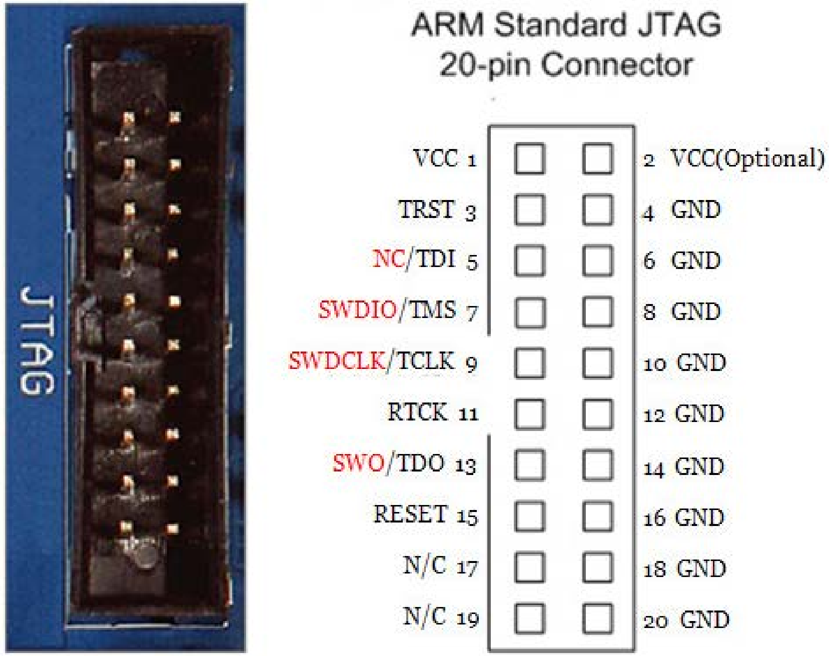
Build result

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



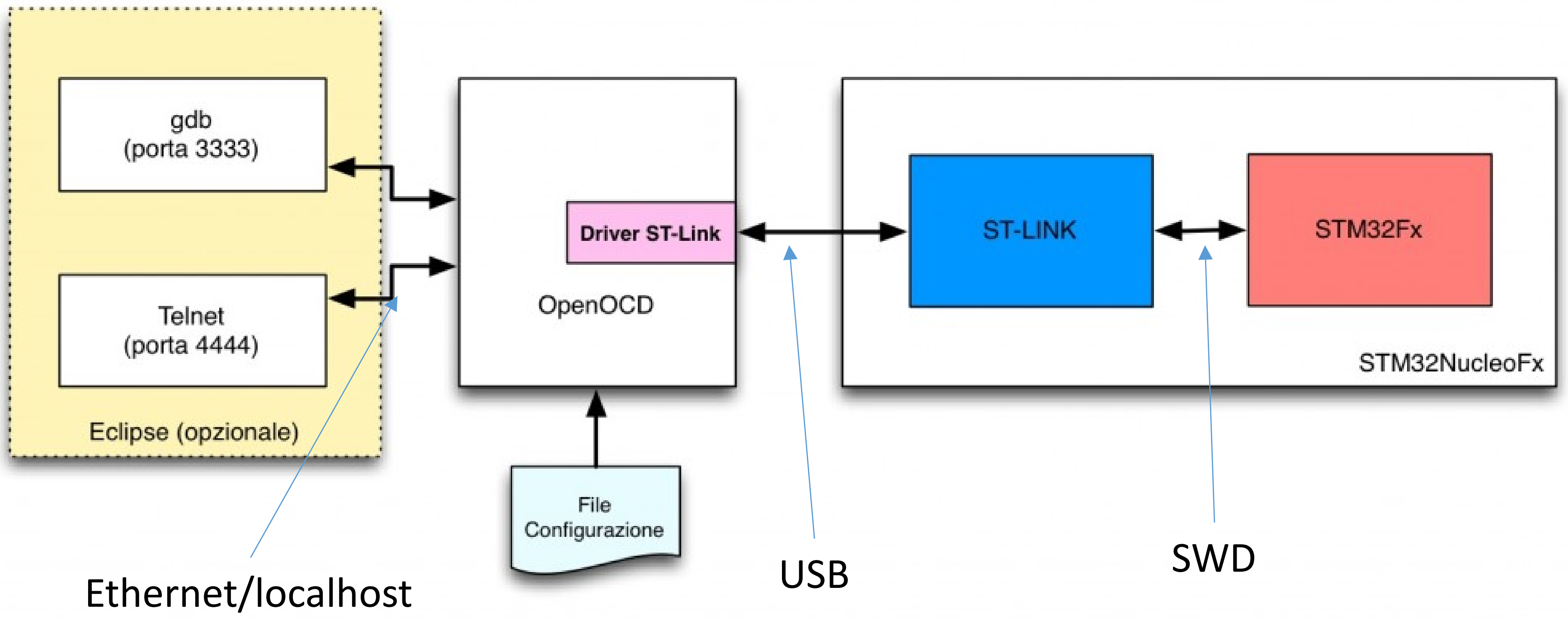
ARM Standard JTAG Connector(20-pins)



VCC	1	2	VCC(Optional)
TRST	3	4	GND
NC/TDI	5	6	GND
SWDIO/TMS	7	8	GND
SWDCLK/TCLK	9	10	GND
RTCK	11	12	GND
SWO/TDO	13	14	GND
RESET	15	16	GND
N/C	17	18	GND
N/C	19	20	GND

Debug

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



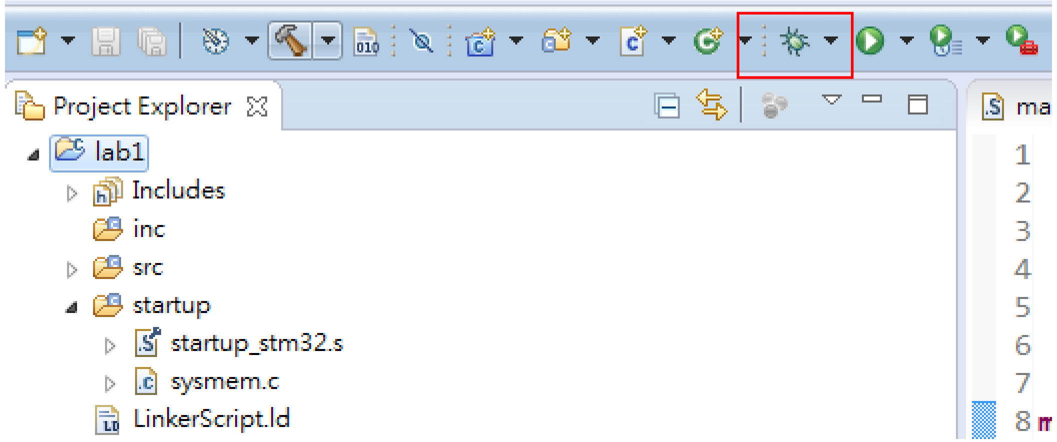
Ethernet/localhost

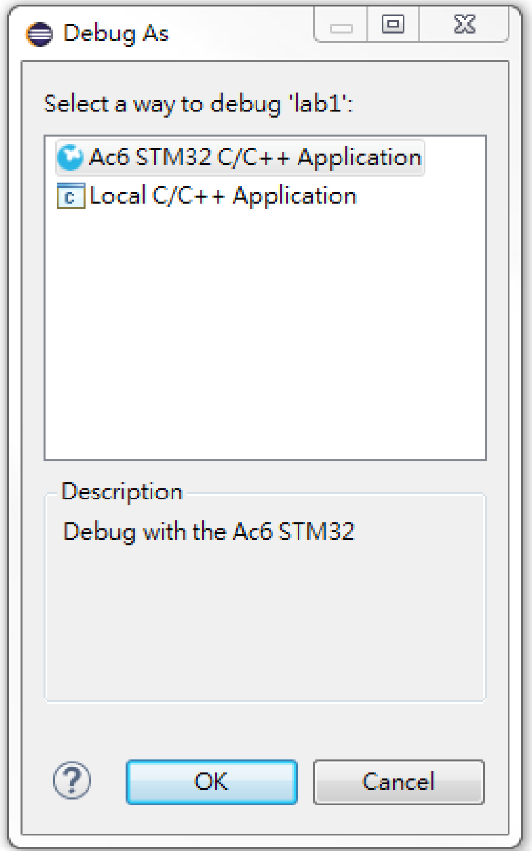
USB

SWD

Create a debug configure

- Run->Debug
- Debug as 'AC6 STM32 C/C++ Application'




Debug As

Select a way to debug 'lab1':

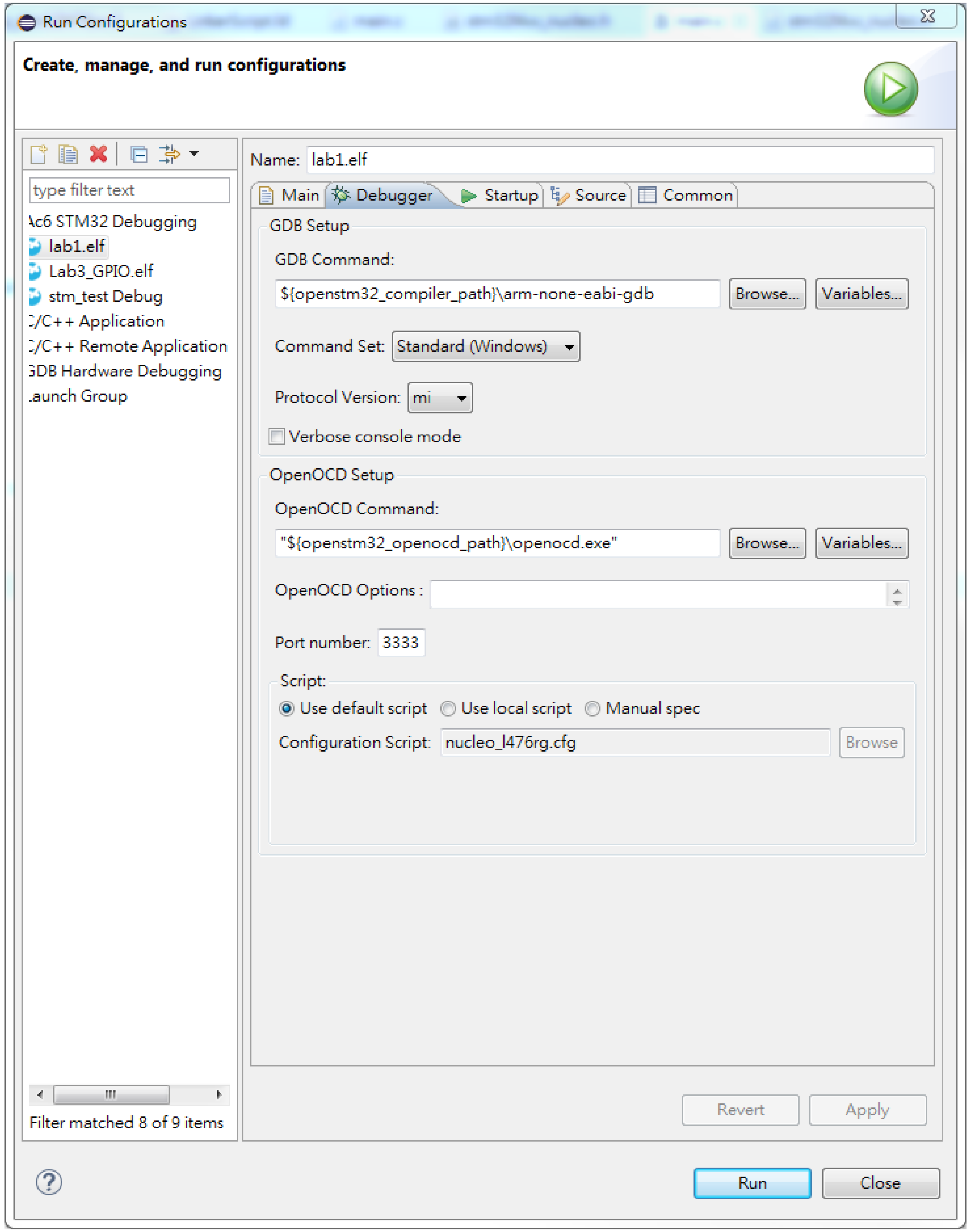
Ac6 STM32 C/C++ Application
Local C/C++ Application

Description

Debug with the Ac6 STM32

- Check your debugger configuration
- Run -> Debug Configuration
- LD1會紅綠閃爍

Note: Make sure your port 3333 no bind any network service!



Name: lab1.elf

GDB Command: ${openstm32_compiler_path}\arm-none-eabi-gdb

Command Set: Standard (Windows)

Protocol Version: mi

□ Verbose console mode

OpenOCD Command: "${openstm32_openocd_path}\openocd.exe"

OpenOCD Options :

Port number: 3333

Script: ● Use default script ○ Use local script ○ Manual spec

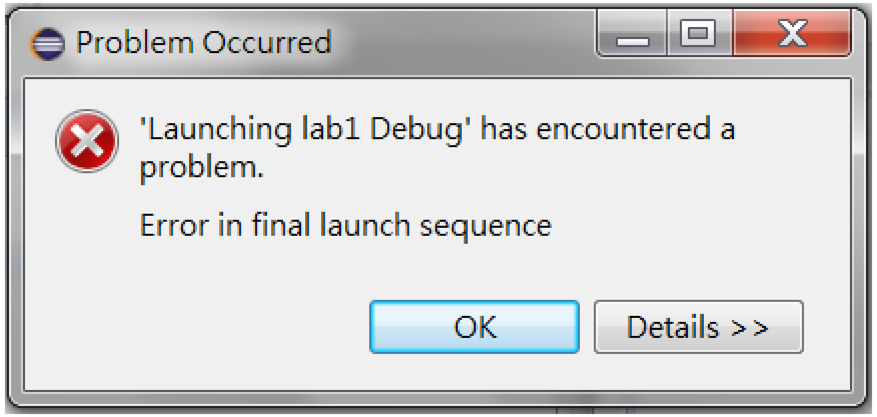
Configuration Script: nucleo_l476rg.cfg

Lab這樣就可以建立

下面介紹一些常見的Bug解決方法

Error in final launch sequence

- Terminate the debugging



Problem Occurred

'Launching lab1 Debug' has encountered a problem.

Error in final launch sequence

Error in final launch sequence

- 有很多同學在Debug時出現以下錯誤訊息：
- Error in final launch sequence
- Failed to execute MI command:
- -target-select remote localhost:3333

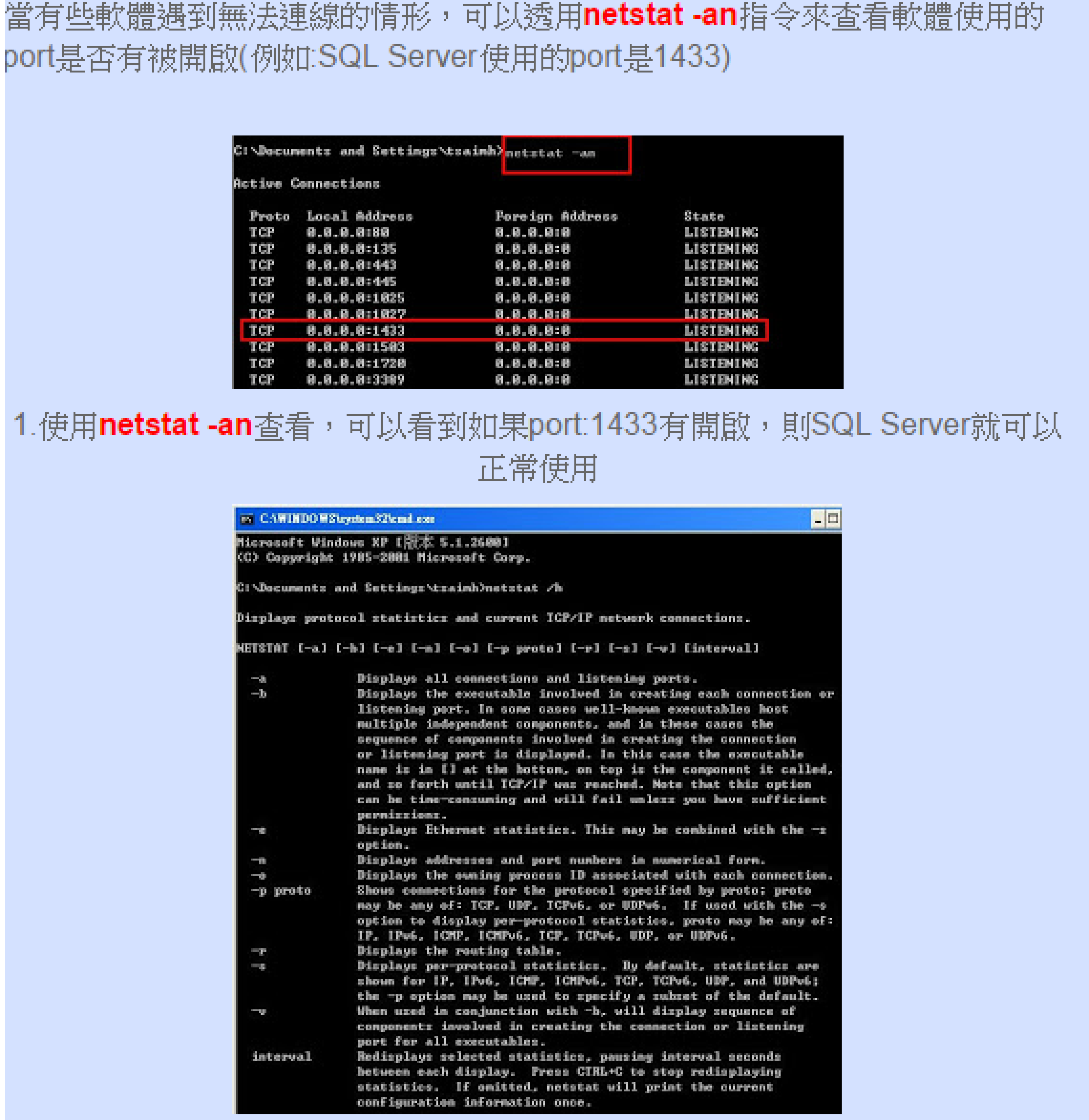
- Error message from debugger back end:
- localhost:3333: ¨t²Î¸Õ¹Ï±NºÏºÐ¾÷¥[¤J¨ì¤w³Q¥[¤JªººÏºÐ¾÷¥Ø¿ý¡C\r\n.
- localhost:3333: ¨t²Î¸Õ¹Ï±NºÏºÐ¾÷¥[¤J¨ì¤w³Q¥[¤JªººÏºÐ¾÷¥Ø¿ý¡C\r\n.

- 遇到相同問題的同學請先確認：
- 1.STM32(開發板)已連接電腦(第一次連接會自動安裝驅動，需要等段時間，安裝完成後會出現名為NODE_L476RG的1MB磁碟裝置)
- 2.若開發板已正確連接電腦仍出現相同問題，請檢查port3333是否有被電腦中其他程式佔用
- 3.若是之前有正常開啟debug，但第二次開啟debug時發生錯誤，請檢查原先的debug是否已經停止(不能同時開兩個debug)

Error in final launch sequence

- C:\>netstat -an

- Listening表示被占用



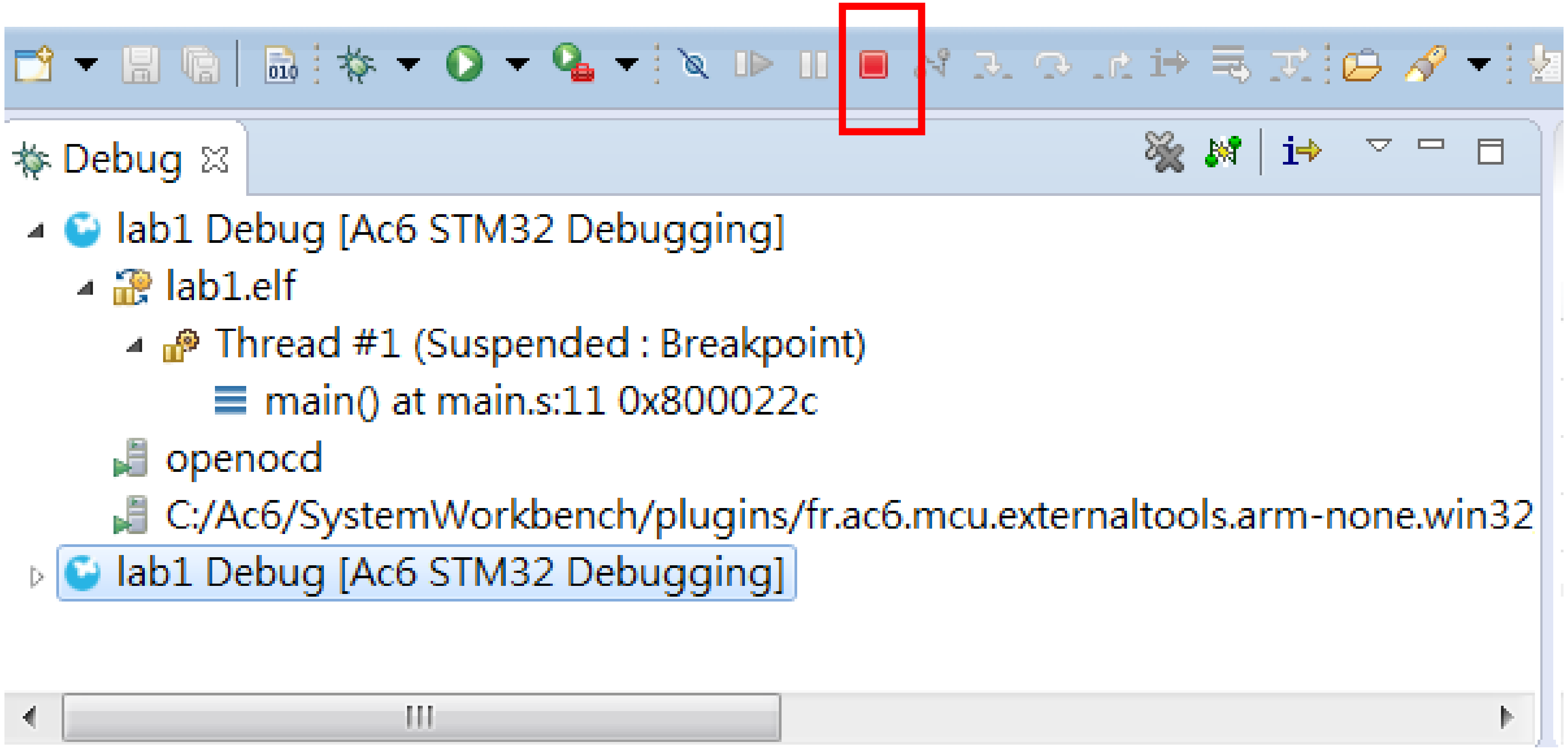
當有些軟體遇到無法連線的情形，可以透用**netstat -an**指令來查看軟體使用的port是否有被開啟(例如:SQL Server使用的port是1433)

```
C:\Documents and Settings\tsaimh>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1027           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1433           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1503           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1720           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
```

1.使用**netstat -an**查看，可以看到如果port:1433有開啟，則SQL Server就可以正常使用

```
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\tsaimh>netstat /h

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-b] [-e] [-n] [-o] [-p proto] [-r] [-s] [-v] [interval]

  -a            Displays all connections and listening ports.
  -b            Displays the executable involved in creating each connection or
                listening port. In some cases well-known executables host
                multiple independent components, and in these cases the
                sequence of components involved in creating the connection
                or listening port is displayed. In this case the executable
                name is in [] at the bottom, on top is the component it called,
                and so forth until TCP/IP was reached. Note that this option
                can be time-consuming and will fail unless you have sufficient
                permissions.
  -e            Displays Ethernet statistics. This may be combined with the -s
                option.
  -n            Displays addresses and port numbers in numerical form.
  -o            Displays the owning process ID associated with each connection.
  -p proto      Shows connections for the protocol specified by proto; proto
                may be any of: TCP, UDP, TCPv6, or UDPv6.  If used with the -s
                option to display per-protocol statistics, proto may be any of:
                IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, or UDPv6.
  -r            Displays the routing table.
  -s            Displays per-protocol statistics.  By default, statistics are
                shown for IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6;
                the -p option may be used to specify a subset of the default.
  -v            When used in conjunction with -b, will display sequence of
                components involved in creating the connection or listening
                port for all executables.
  interval      Redisplays selected statistics, pausing interval seconds
                between each display.  Press CTRL+C to stop redisplaying
                statistics.  If omitted, netstat will print the current
                configuration information once.
```

Error in final launch sequence



Debug

lab1 Debug [Ac6 STM32 Debugging]

lab1.elf

Thread #1 (Suspended : Breakpoint)

main() at main.s:11 0x800022c

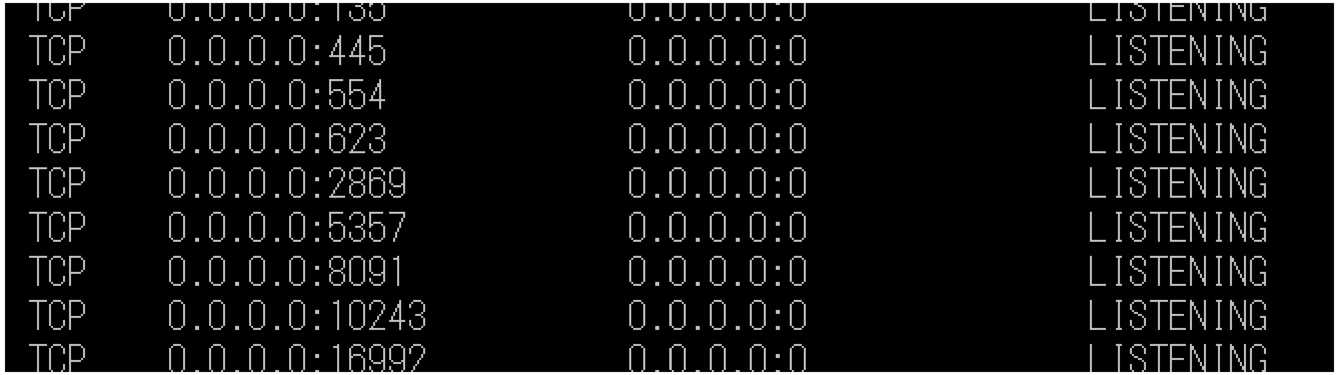
openocd

C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32

lab1 Debug [Ac6 STM32 Debugging]

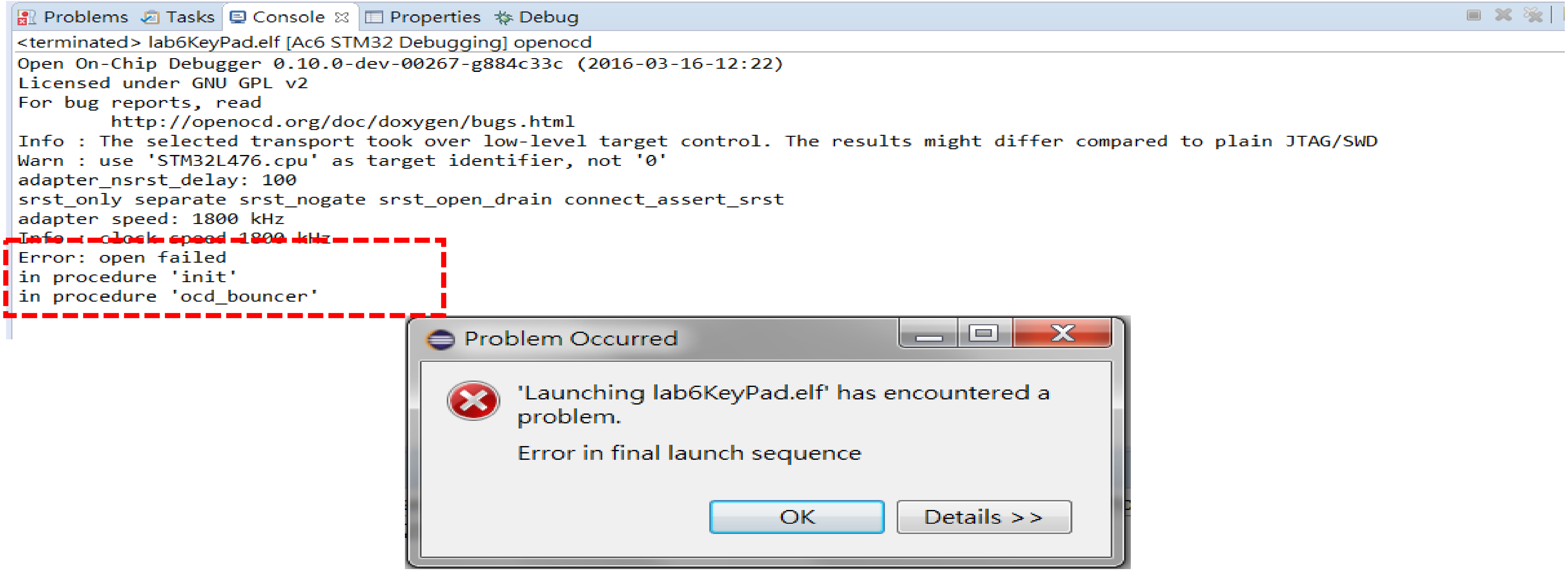
- 把之前開的debugging terminate掉

- Terminate debugging後 port3333 就沒有listening了



```
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
TCP    0.0.0.0:554            0.0.0.0:0              LISTENING
TCP    0.0.0.0:623            0.0.0.0:0              LISTENING
TCP    0.0.0.0:2869           0.0.0.0:0              LISTENING
TCP    0.0.0.0:5357           0.0.0.0:0              LISTENING
TCP    0.0.0.0:8091           0.0.0.0:0              LISTENING
TCP    0.0.0.0:10243          0.0.0.0:0              LISTENING
TCP    0.0.0.0:16992          0.0.0.0:0              LISTENING
```

- USB沒有插好，重插

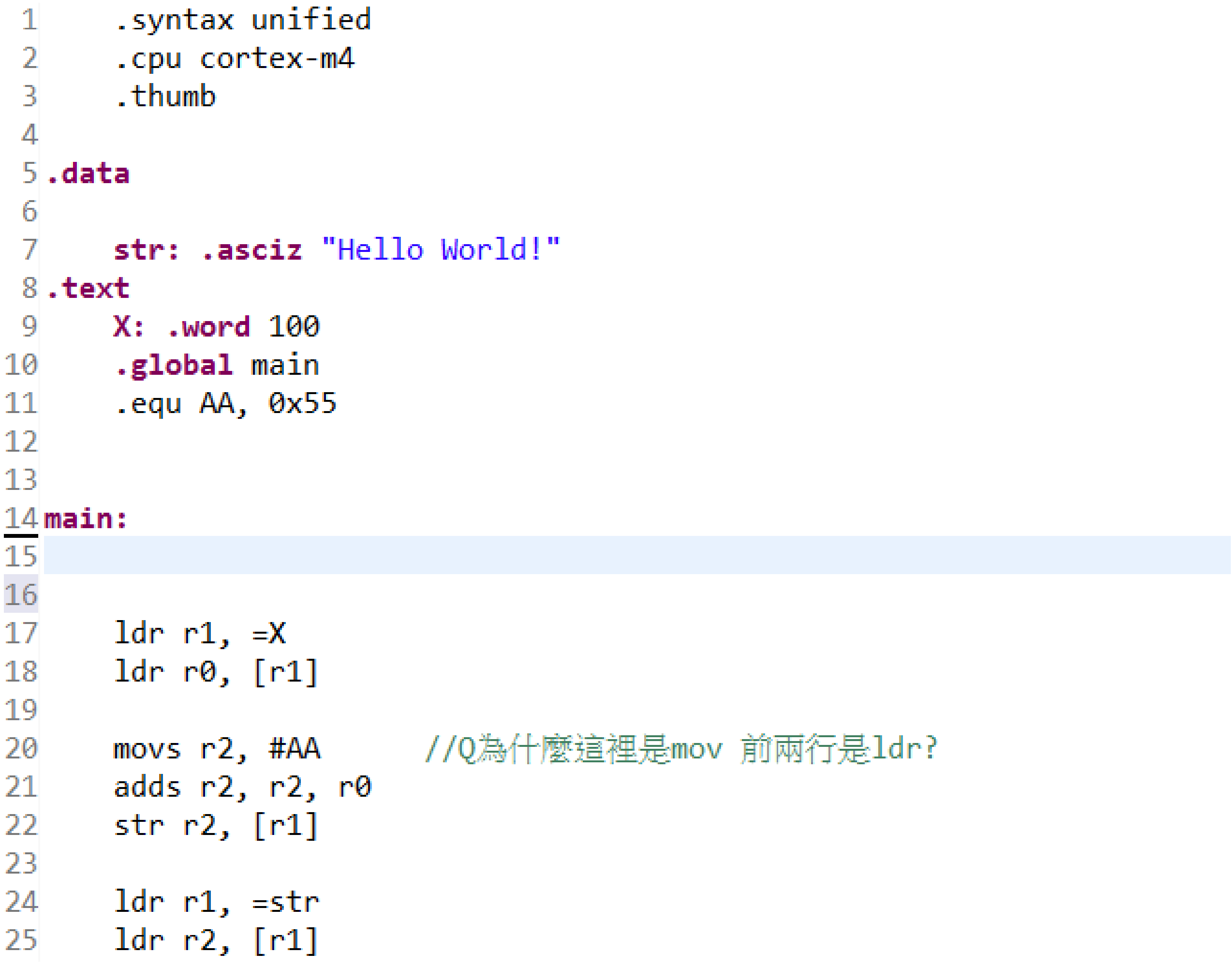


```
<terminated> lab6KeyPad.elf [Ac6 STM32 Debugging] openocd
Open On-Chip Debugger 0.10.0-dev-00267-g884c33c (2016-03-16-12:22)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : The selected transport took over low-level target control. The results might differ compared to plain JTAG/SWD
Warn : use 'STM32L476.cpu' as target identifier, not '0'
adapter_nsrst_delay: 100
srst_only separate srst_nogate srst_open_drain connect_assert_srst
adapter speed: 1800 kHz
Info : clock speed 1800 kHz
Error: open failed
in procedure 'init'
in procedure 'ocd_bouncer'
```

Problem Occurred

'Launching lab6KeyPad.elf' has encountered a problem.

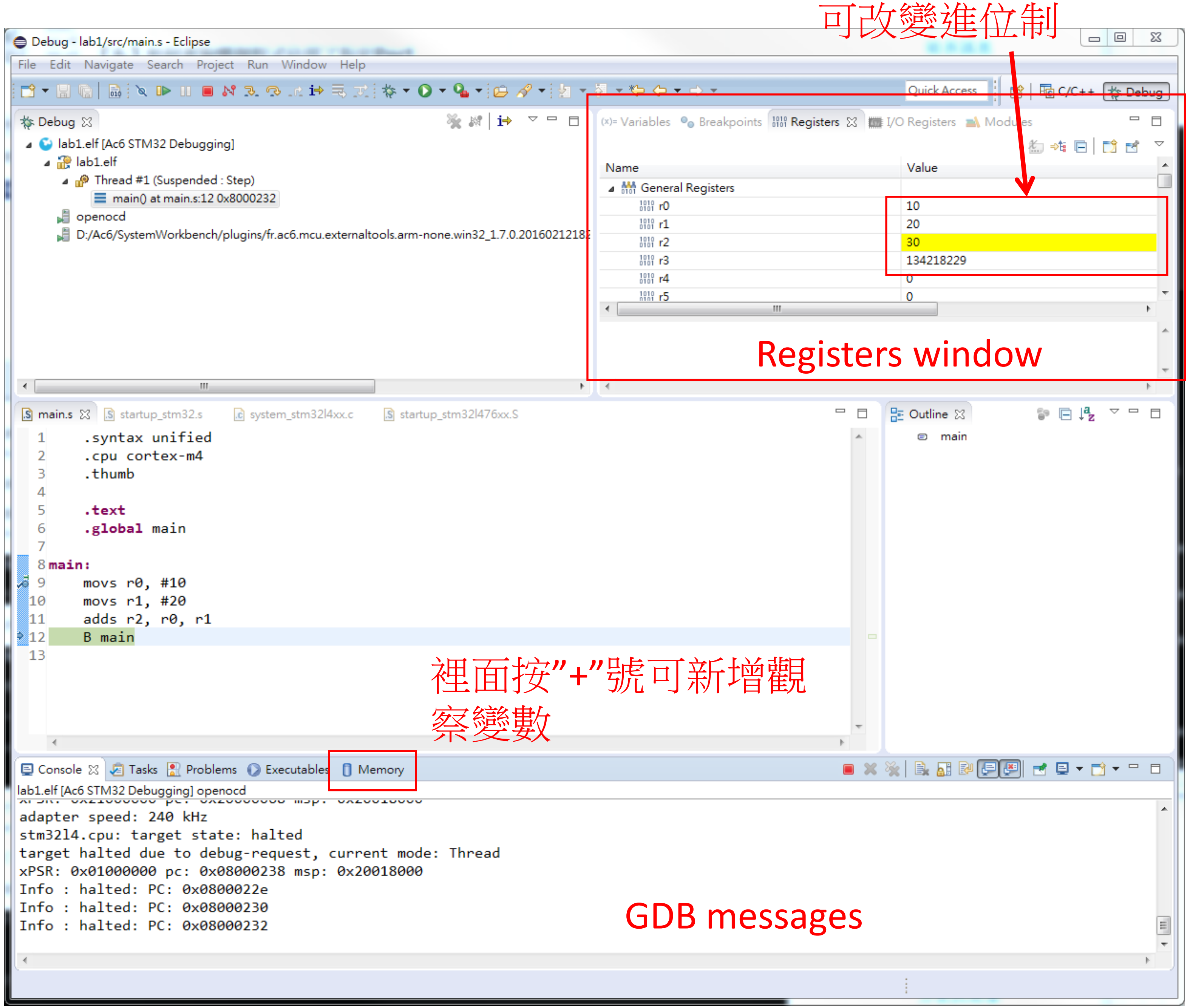
Error in final launch sequence



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

.data

    str: .asciz "Hello World!"
.text
    X: .word 100
    .global main
    .equ AA, 0x55


main:


    ldr r1, =X
    ldr r0, [r1]

    movs r2, #AA      //Q為什麼這裡是mov 前兩行是ldr?
    adds r2, r2, r0
    str r2, [r1]

    ldr r1, =str
    ldr r2, [r1]
```

Register

- By default the GDB will set the first breakpoint at ‘main’
  連點行號
- run Debug
- Press ‘Step into’ button or ‘F5’ will debug your code step by step.
- PSR: program state register



右鍵->number format
可改變進位制

lab1.elf [Ac6 STM32 Debugging]
lab1.elf
Thread #1 (Suspended : Step)
main() at main.s:12 0x8000232
openocd
D:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.2016021218

Name	Value
General Registers	
r0	10
r1	20
r2	30
r3	134218229
r4	0
r5	0

Registers window

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

    .text
    .global main

main:
    movs r0, #10
    movs r1, #20
    adds r2, r0, r1
    B main
```

裡面按”+”號可新增觀察變數

lab1.elf [Ac6 STM32 Debugging] openocd

```
adapter speed: 240 kHz
stm32l4.cpu: target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08000238 msp: 0x20018000
Info : halted: PC: 0x0800022e
Info : halted: PC: 0x08000230
Info : halted: PC: 0x08000232
```

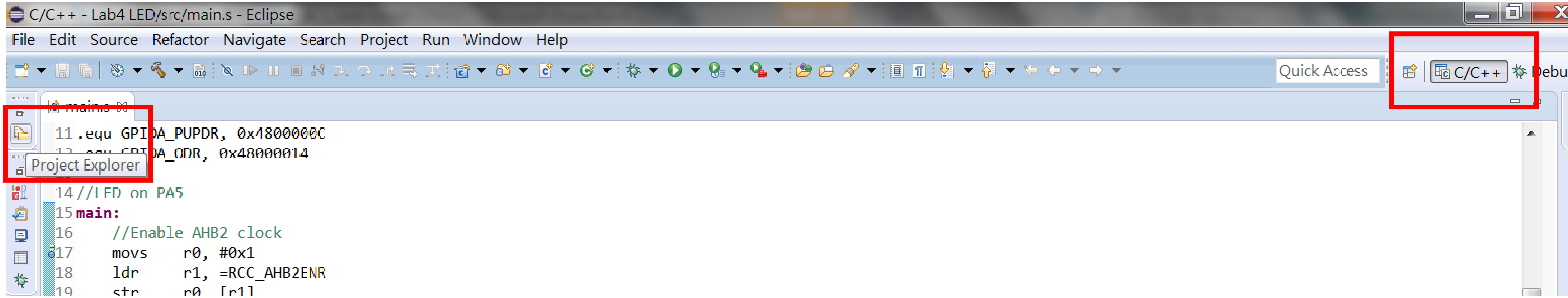
GDB messages

Run program

- Debug過程中，若在console中有顯示下列字樣表示程式ok

```
Info : Device id = 0x10076415
** Programming Finished **
** Verified OK **
```

開啟電腦中已存在之Project



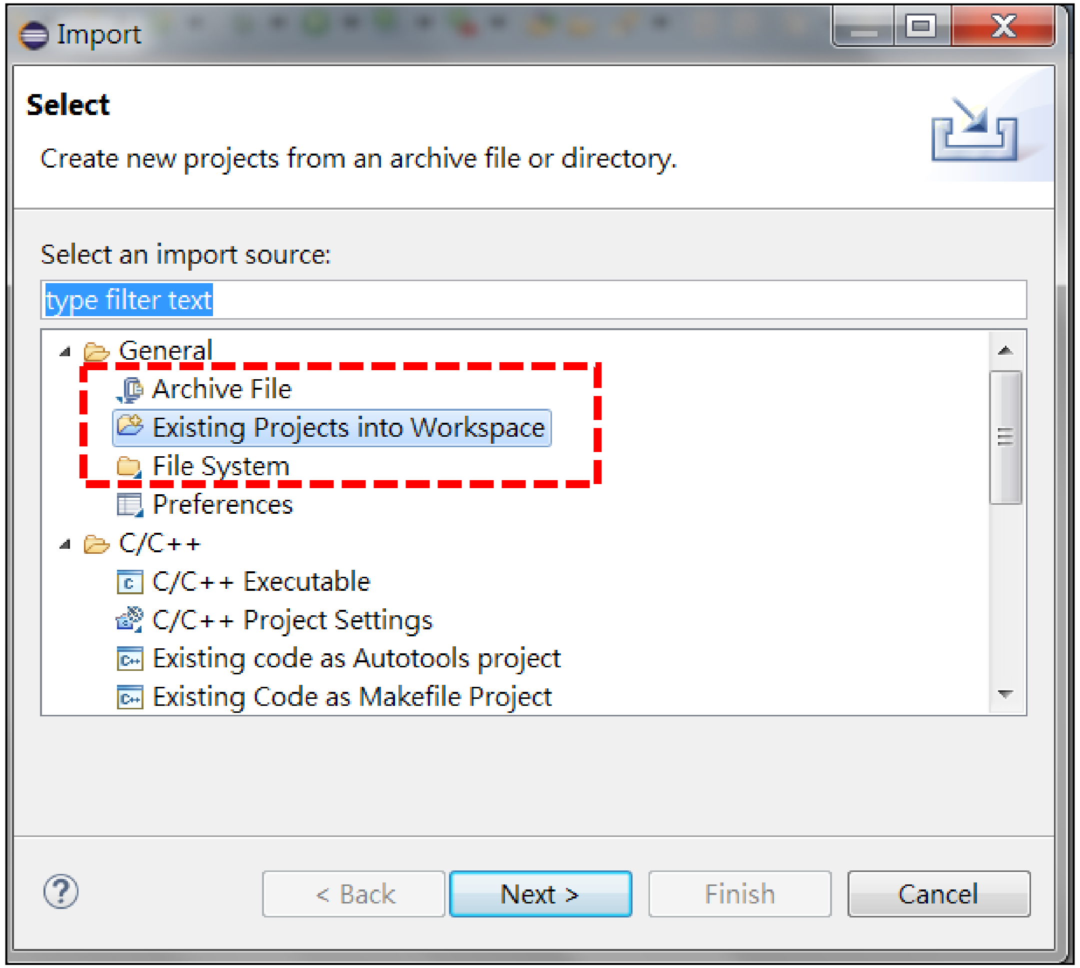
```
.equ GPIOA_PUPDR, 0x4800000C
.equ GPIOA_ODR, 0x48000014

//LED on PA5
main:
    //Enable AHB2 clock
    movs    r0, #0x1
    ldr     r1, =RCC_AHB2ENR
    str     r0, [r1]
```

1. 點Project Explorer
2. 點開所需的.s檔
3. 鐵鎚build
4. Debug

開啟別人給你之Project - 1

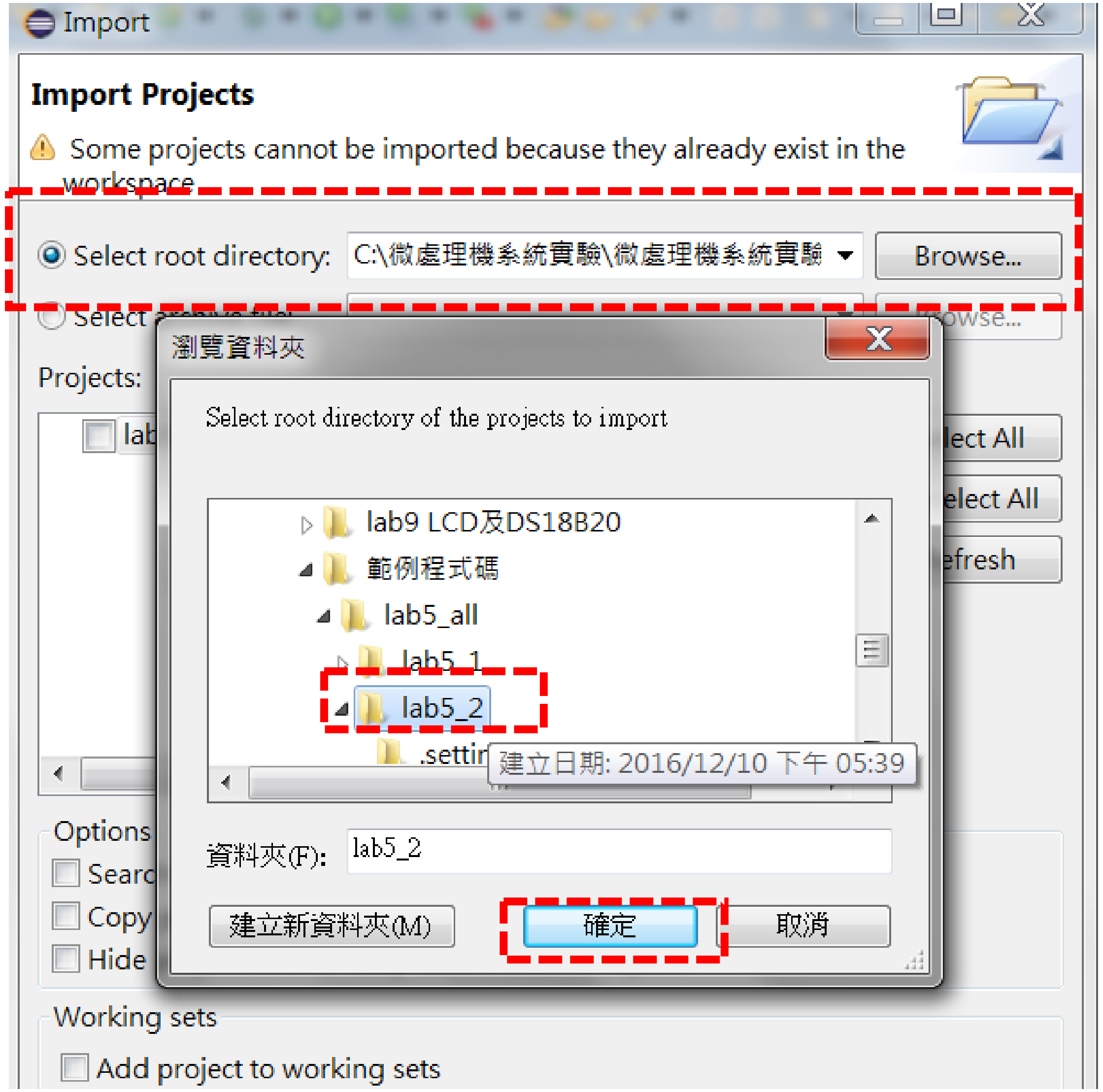
1. File → Import
2. 選General →

   Existing Projects into workspace



開啟別人給你之Project -2

- 找到該Project所在的整個資料夾
- 選取後，按確定把整個資料夾import
- 如果找不到檔案可能要設定include的資料夾路徑



Import Projects

Some projects cannot be imported because they already exist in the workspace

● Select root directory: C:\微處理機系統實驗\微處理機系統實驗

瀏覽資料夾

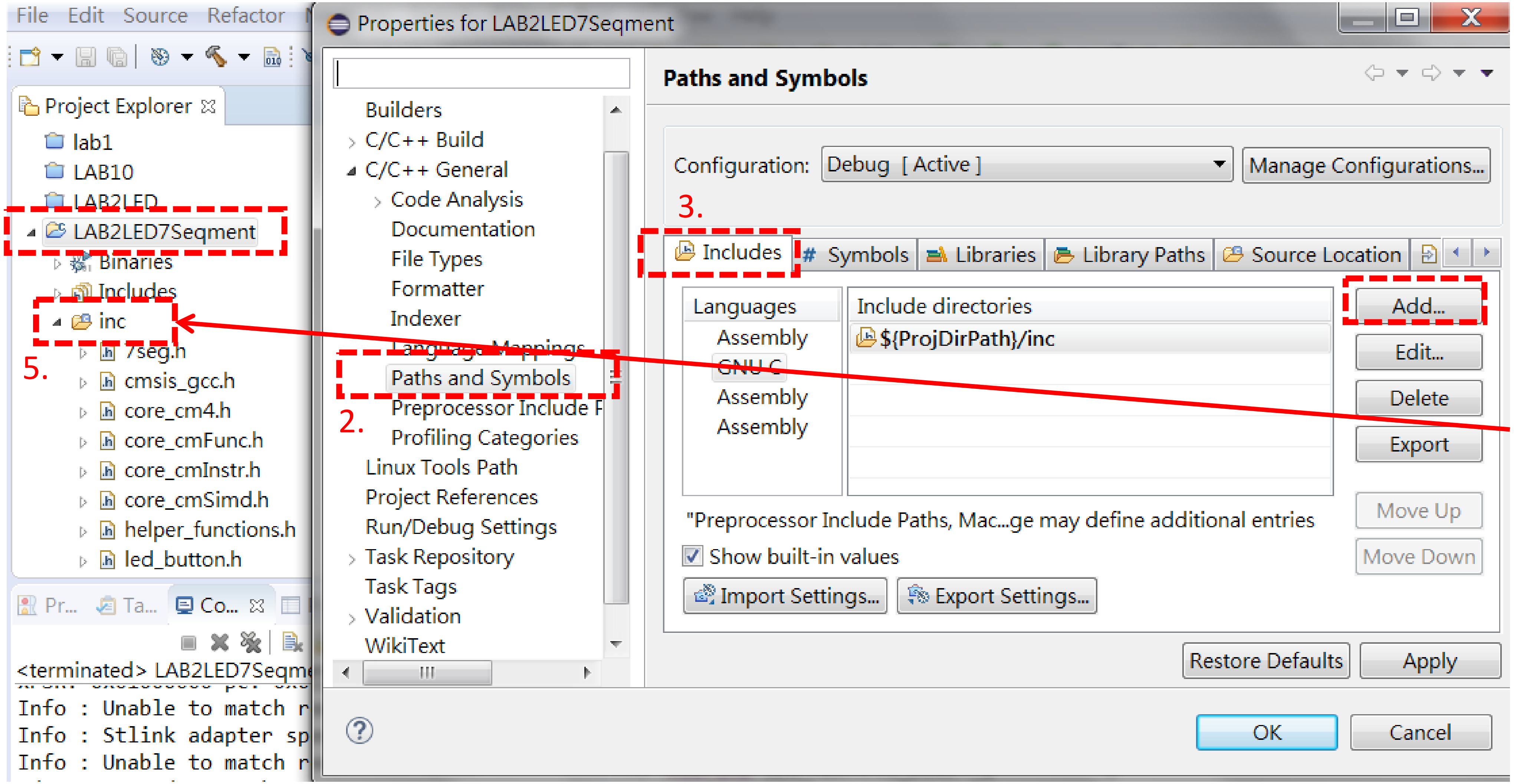
Select root directory of the projects to import

建立日期: 2016/12/10 下午 05:39

資料夾(F): lab5_2

如果找不到檔案可能要設定include的資料夾路徑，或者

1.右鍵選 property



Configuration: Debug [Active]

Include directories: ${ProjDirPath}/inc

"Preprocessor Include Paths, Mac...ge may define additional entries

☑ Show built-in values

```
<terminated> LAB2LED7Seqme
Info : Unable to match r
Info : Stlink adapter sp
Info : Unable to match r
```

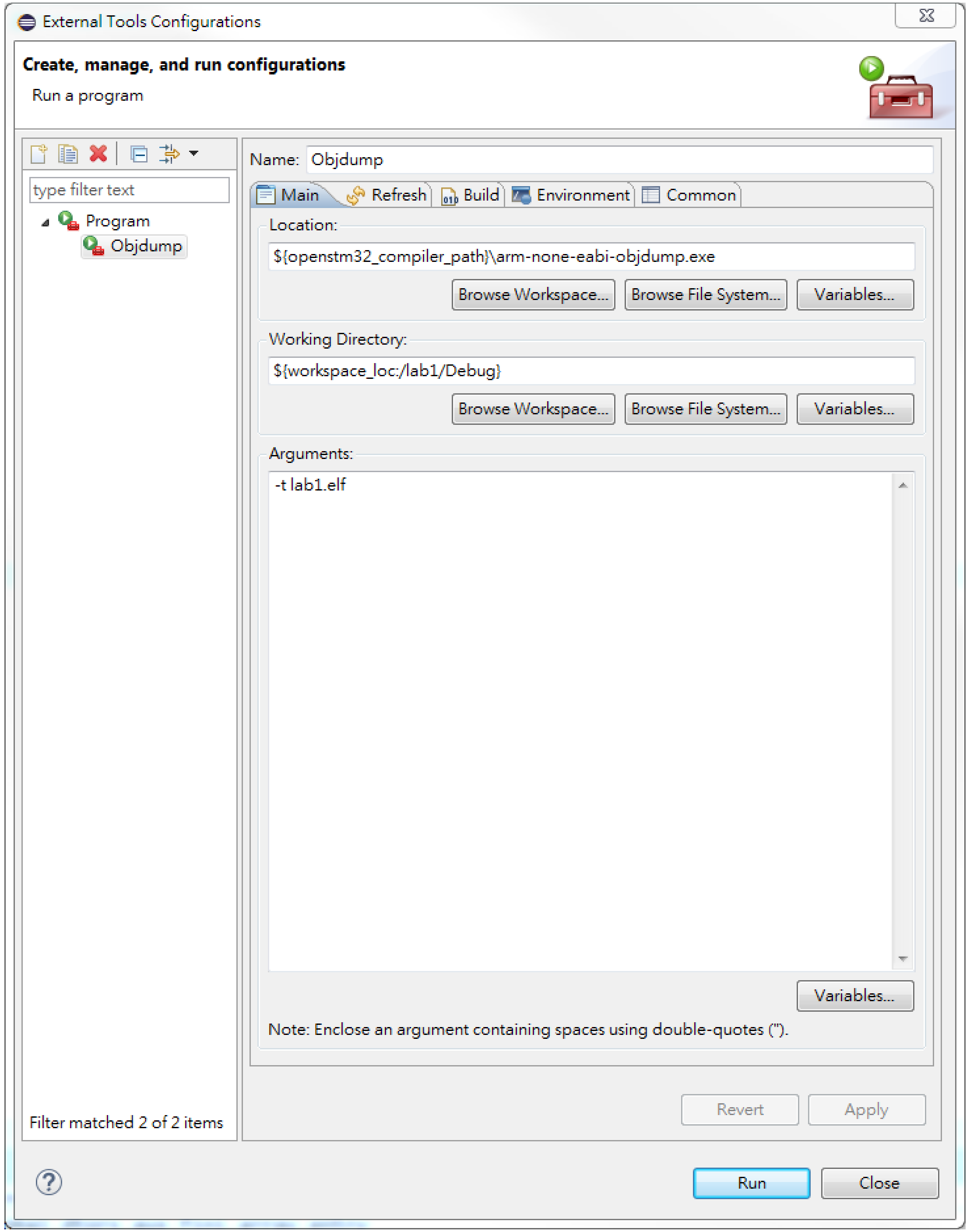
4. add後按 workspace，加入該 project的 inc就找的到

開啟別人給你的.s和.h程式碼

- 先建一個new Project
- 把.h和.s或.c拉入即可

Object Dump

- 可返組譯C到組合語言
- 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



External Tools Configurations

Create, manage, and run configurations

Run a program

Name: Objdump

Location:
${openstm32_compiler_path}\arm-none-eabi-objdump.exe

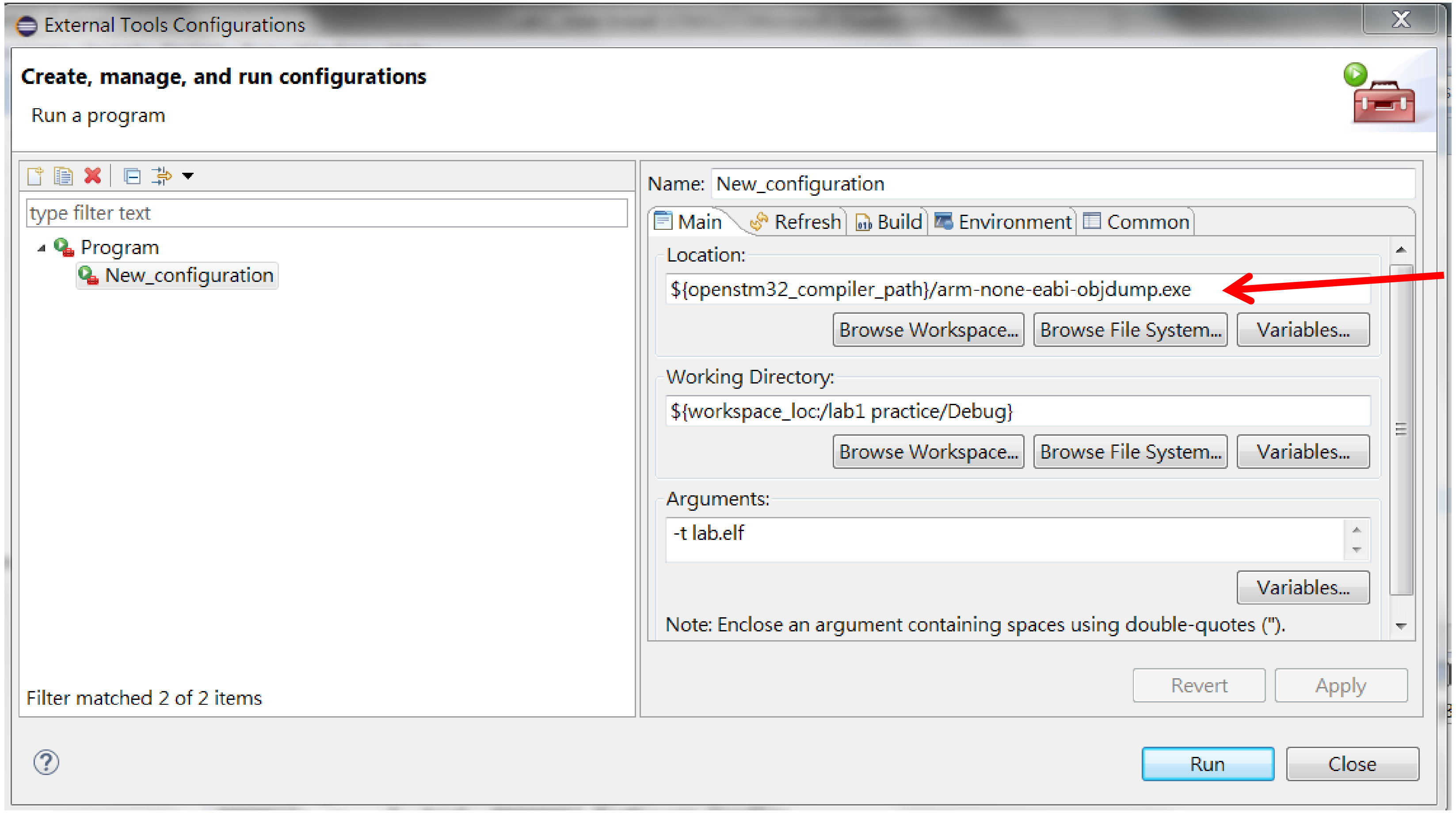
Working Directory:
${workspace_loc:/lab1/Debug}

Arguments:
-t lab1.elf

Note: Enclose an argument containing spaces using double-quotes (").

Filter matched 2 of 2 items

Object Dump　同上



External Tools Configurations

Create, manage, and run configurations

Run a program

type filter text

- Program
  - New_configuration

Filter matched 2 of 2 items

Name: New_configuration

Main | Refresh | Build | Environment | Common

Location:
${openstm32_compiler_path}/arm-none-eabi-objdump.exe

Working Directory:
${workspace_loc:/lab1 practice/Debug}

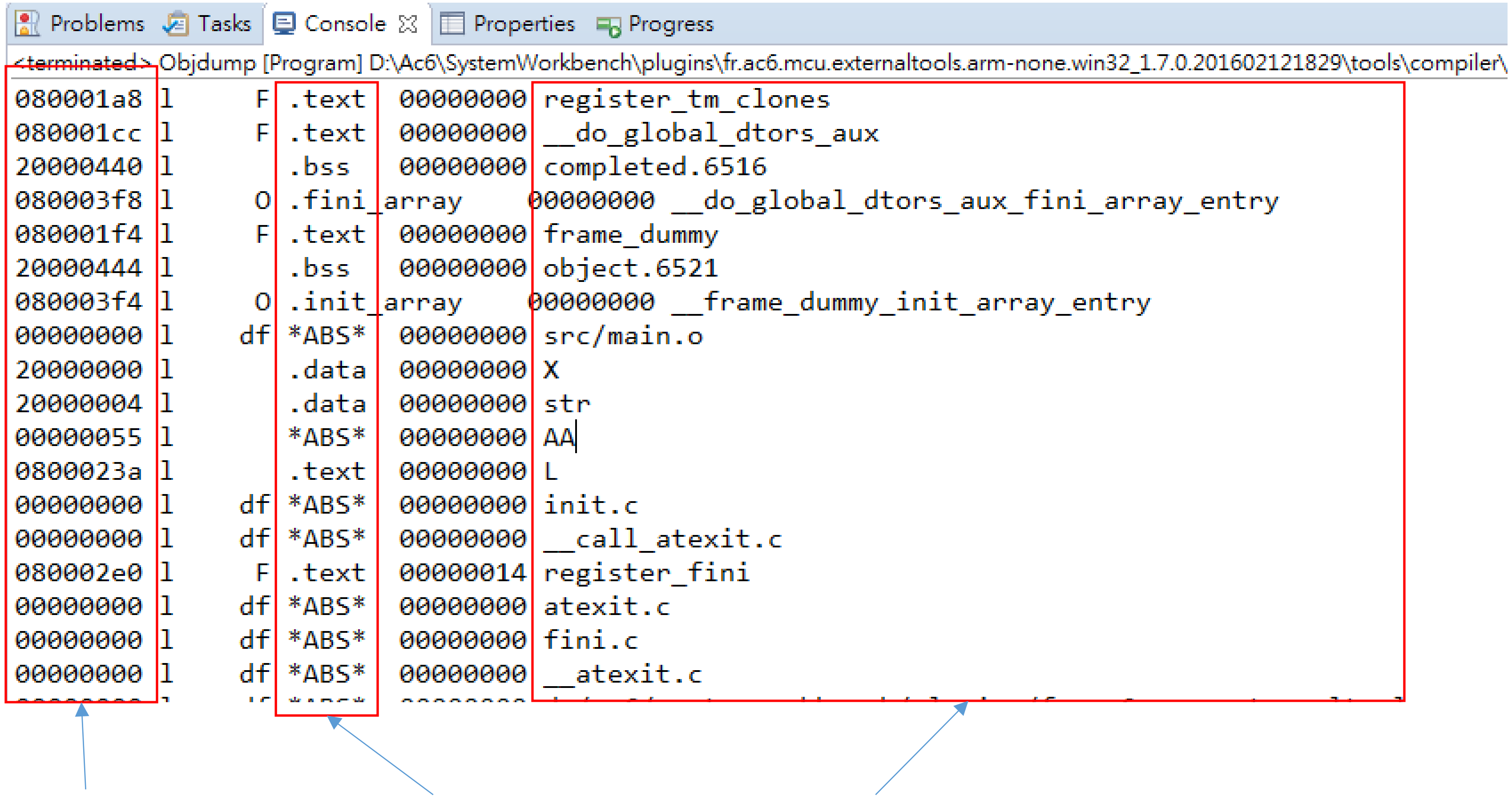
Arguments:
-t lab.elf

Note: Enclose an argument containing spaces using double-quotes (").

C:\Ac6\SystemWorkbench\plugins\fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829\tools\compiler\bin

objdump.exe檔存在這裡

Object Dump: Symbol Table



<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
```

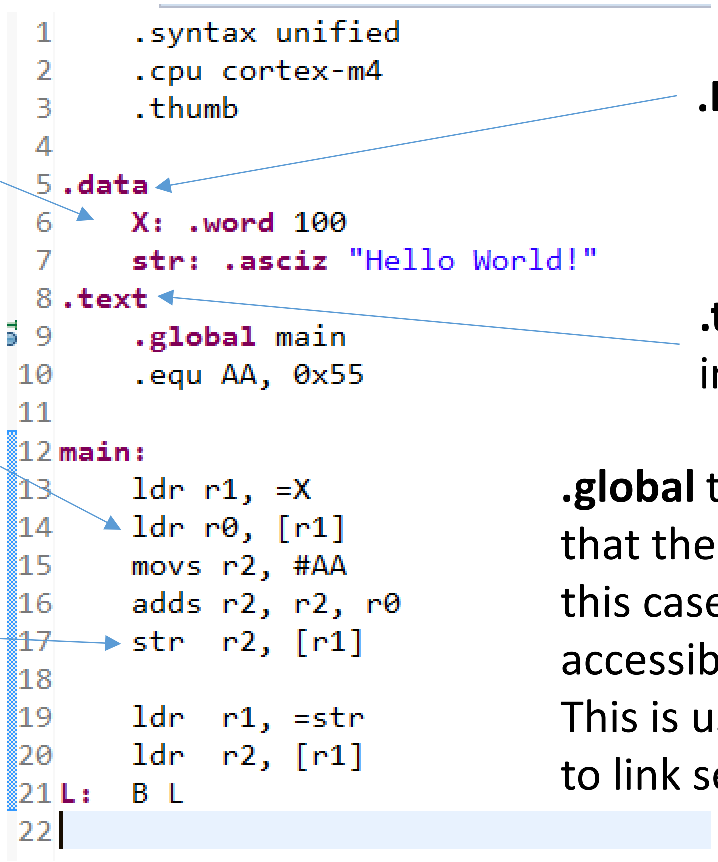
Symbol address

Section locate

Symbol name

Memory Access

- Define data variable
- Direct access
- Indirect read access



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

.data
    X: .word 100
    str: .asciz "Hello World!"
.text
    .global main
    .equ AA, 0x55

main:
    ldr r1, =X
    ldr r0, [r1]
    movs r2, #AA
    adds r2, r2, r0
    str  r2, [r1]

    ldr  r1, =str
    ldr  r2, [r1]
L:  B L

```

.Data section start point

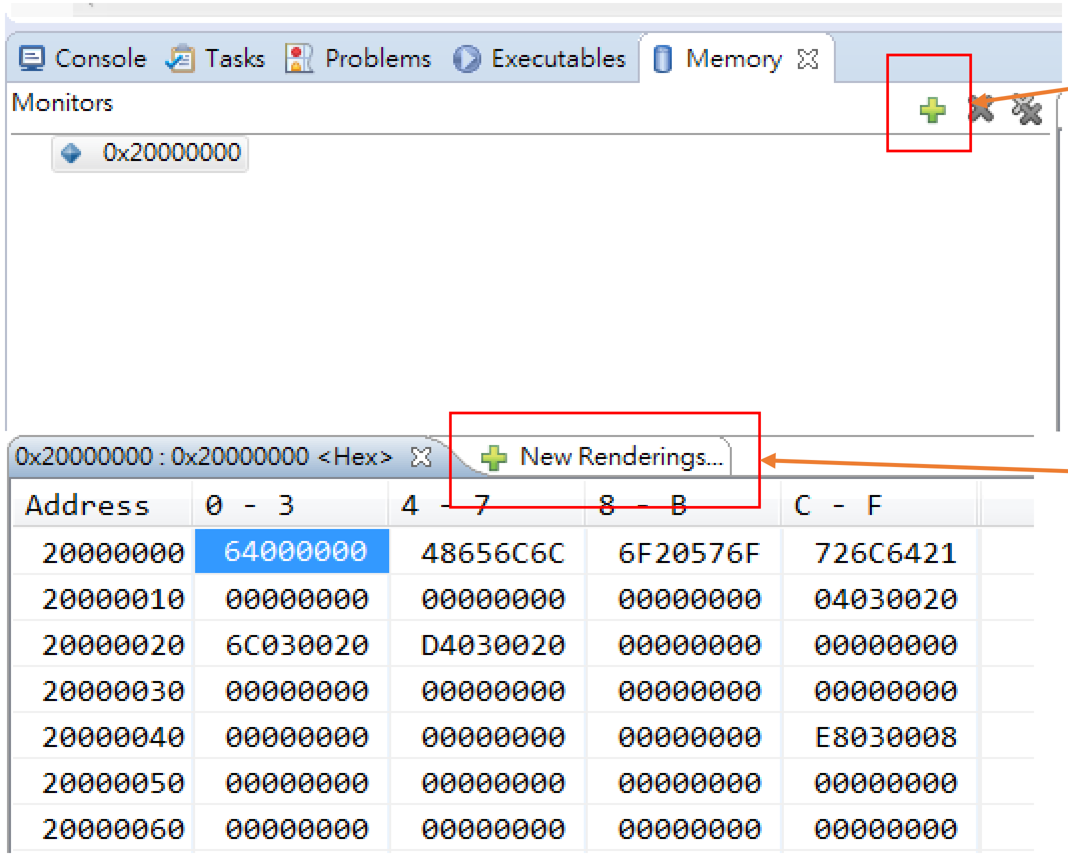
.text assembly instruction start point

.global tells the assembler that the label following it (in this case, "main") is accessible outside the file. This is useful when you want to link several files together.

Write the data register into memory

Memory Monitors

- That can help you watch the memory content



```
Address   0 - 3     4 - 7     8 - B     C - F
20000000  64000000  48656C6C  6F20576F  726C6421
20000010  00000000  00000000  00000000  04030020
20000020  6C030020  D4030020  00000000  00000000
20000030  00000000  00000000  00000000  00000000
20000040  00000000  00000000  00000000  E8030008
20000050  00000000  00000000  00000000  00000000
20000060  00000000  00000000  00000000  00000000
```

Press it to add a memory monitor

Press “New Renderings” can change the display format

Register 和 Monitor是你trace code的好朋友

Reference

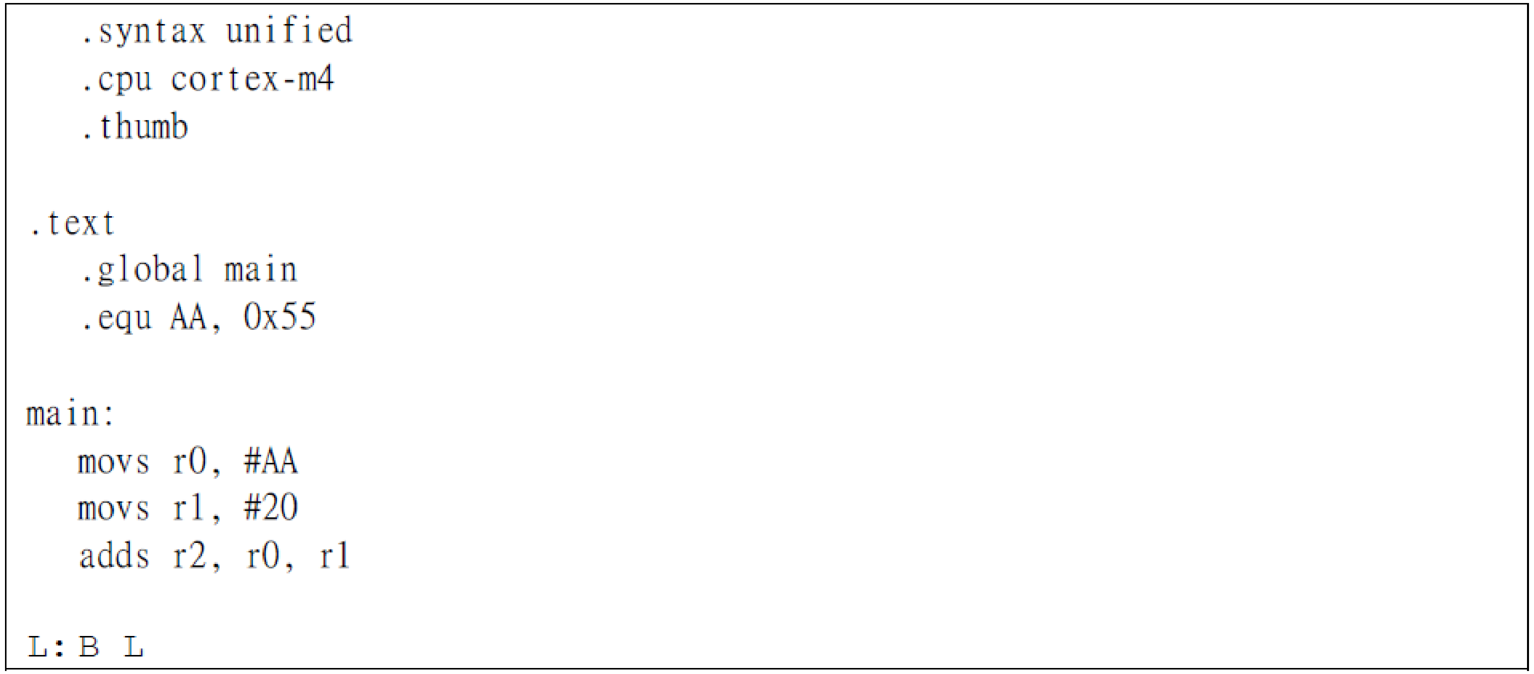
- Getting started with STM32 Nucleo board software development tools
  - http://www.st.com/content/ccc/resource/technical/document/user_manual/1b/03/1b/b4/88/20/4e/cd/DM00105928.pdf/files/DM00105928.pdf/jcr:content/translations/en.DM00105928.pdf
- Assembly 基本語法
  - http://www.w3ii.com/zh-TW/assembly_programming/assembly_basic_syntax.html
- 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

- 給linker看的，把.obj組成可執行檔
- https://www.math.utah.edu/docs/info/ld_toc.html#SEC4

Lab1

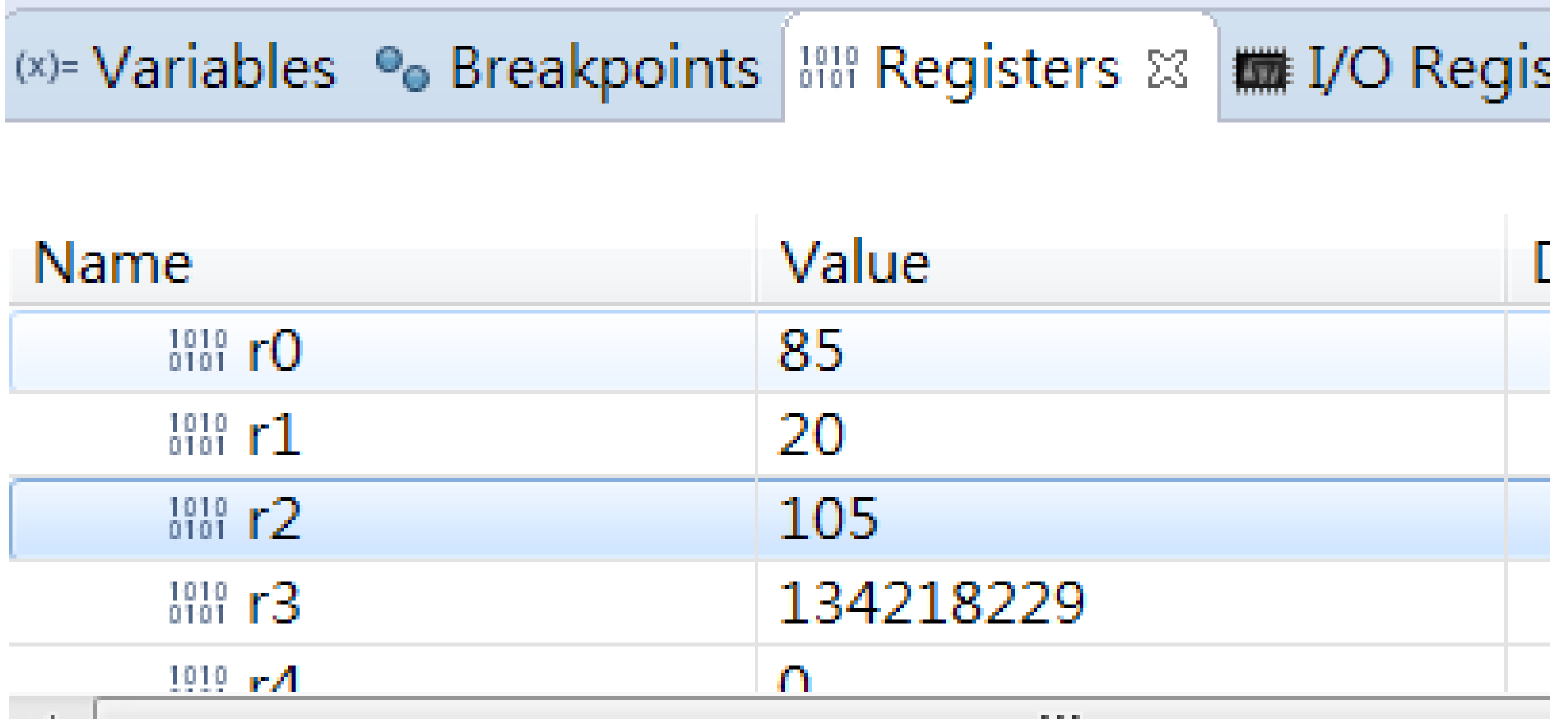
Lab1.1



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

.text
    .global main
    .equ AA, 0x55

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

L: B L
```

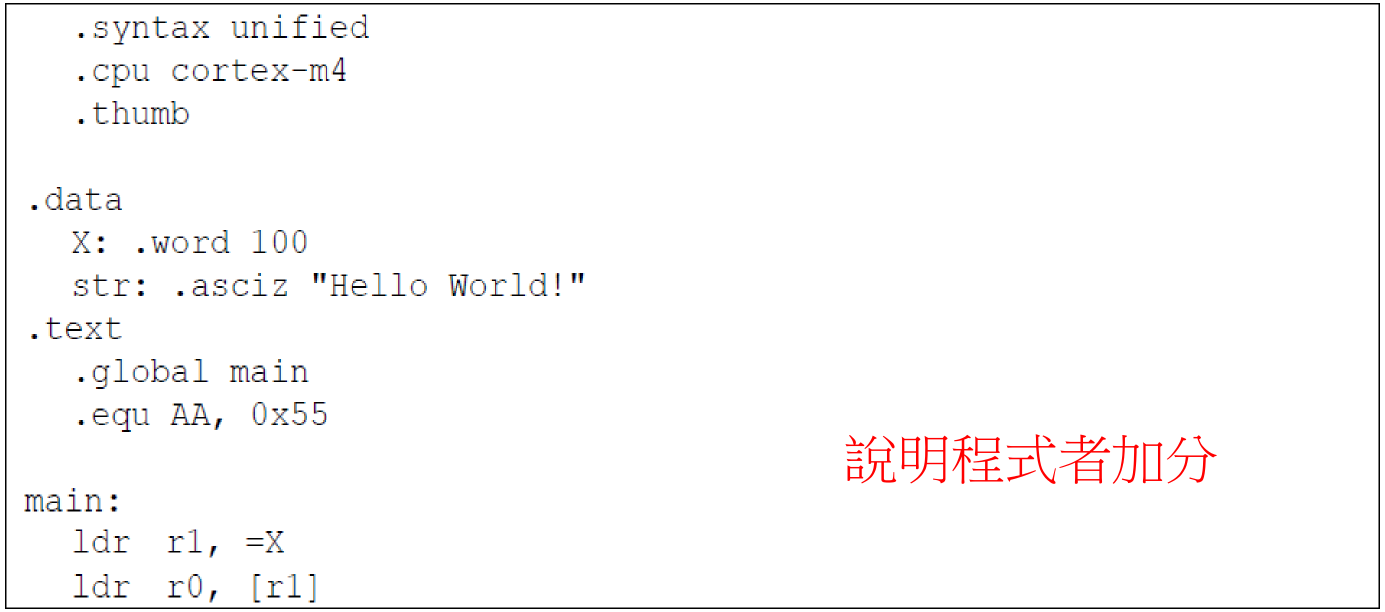
先看一下程式在幹嘛

Lab1.1



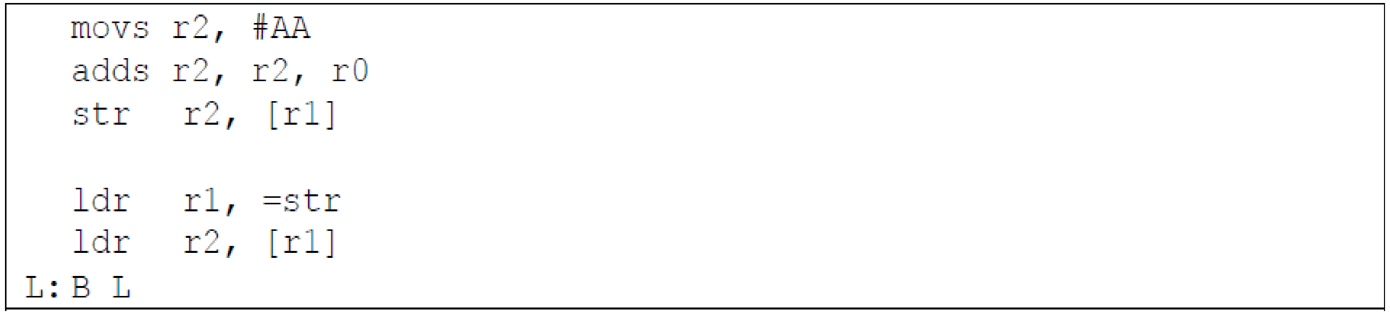
Name	Value
r0	85
r1	20
r2	105
r3	134218229
r4	0

Lab1.2

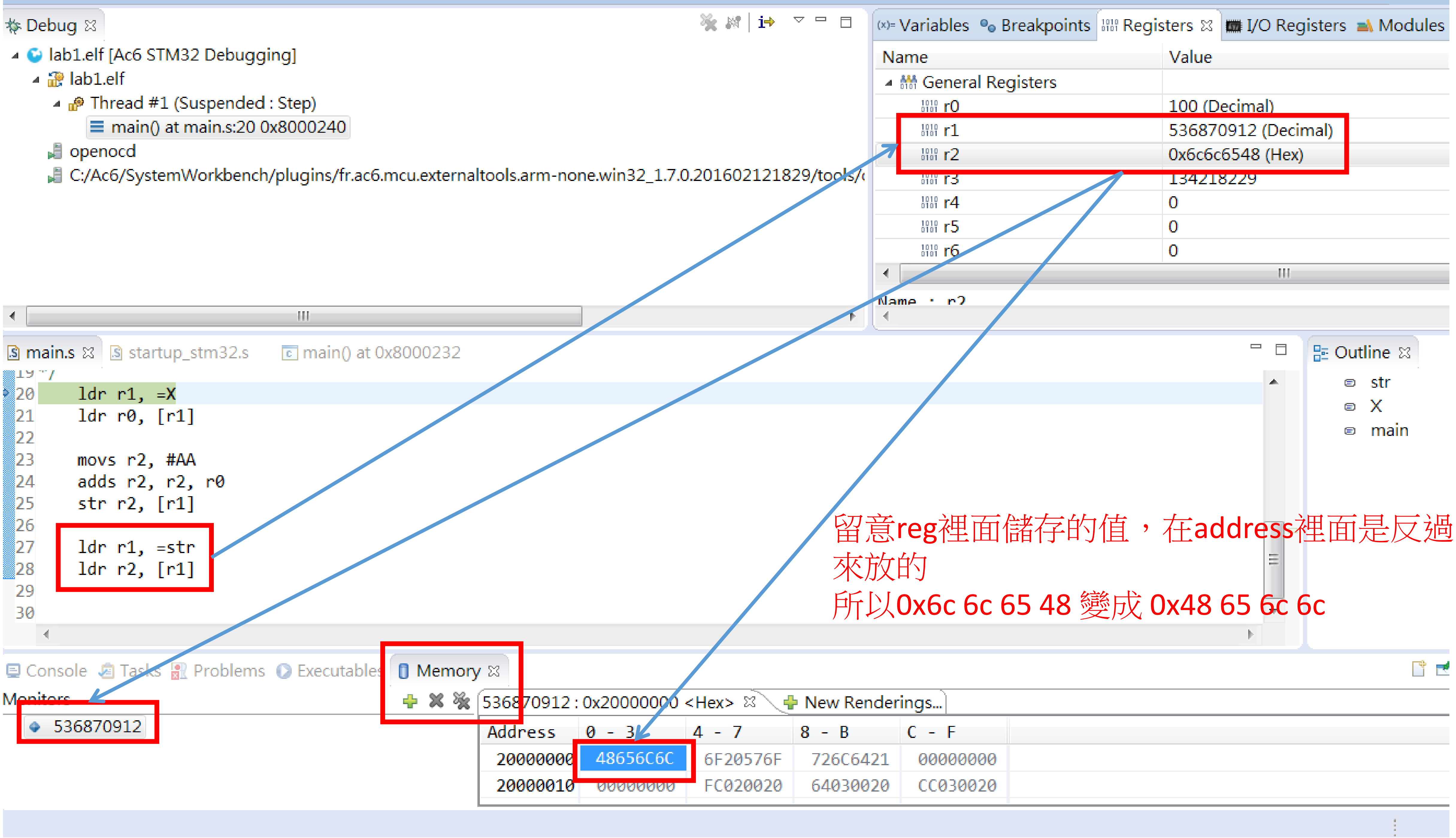


```
    .syntax unified
    .cpu cortex-m4
    .thumb

.data
    X: .word 100
    str: .asciz "Hello World!"
.text
    .global main
    .equ AA, 0x55

main:
    ldr  r1, =X
    ldr  r0, [r1]
```

說明程式者加分



```
    movs r2, #AA
    adds r2, r2, r0
    str   r2, [r1]

    ldr   r1, =str
    ldr   r2, [r1]
L: B L
```



lab1.elf [Ac6 STM32 Debugging]
lab1.elf
Thread #1 (Suspended : Step)
main() at main.s:20 0x8000240
openocd
C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829/tools/

Name	Value
General Registers	
r0	100 (Decimal)
r1	536870912 (Decimal)
r2	0x6c6c6548 (Hex)
r3	134218229
r4	0
r5	0
r6	0

Name : r2

```
*/
    ldr r1, =X
    ldr r0, [r1]

    movs r2, #AA
    adds r2, r2, r0
    str r2, [r1]

    ldr r1, =str
    ldr r2, [r1]


```

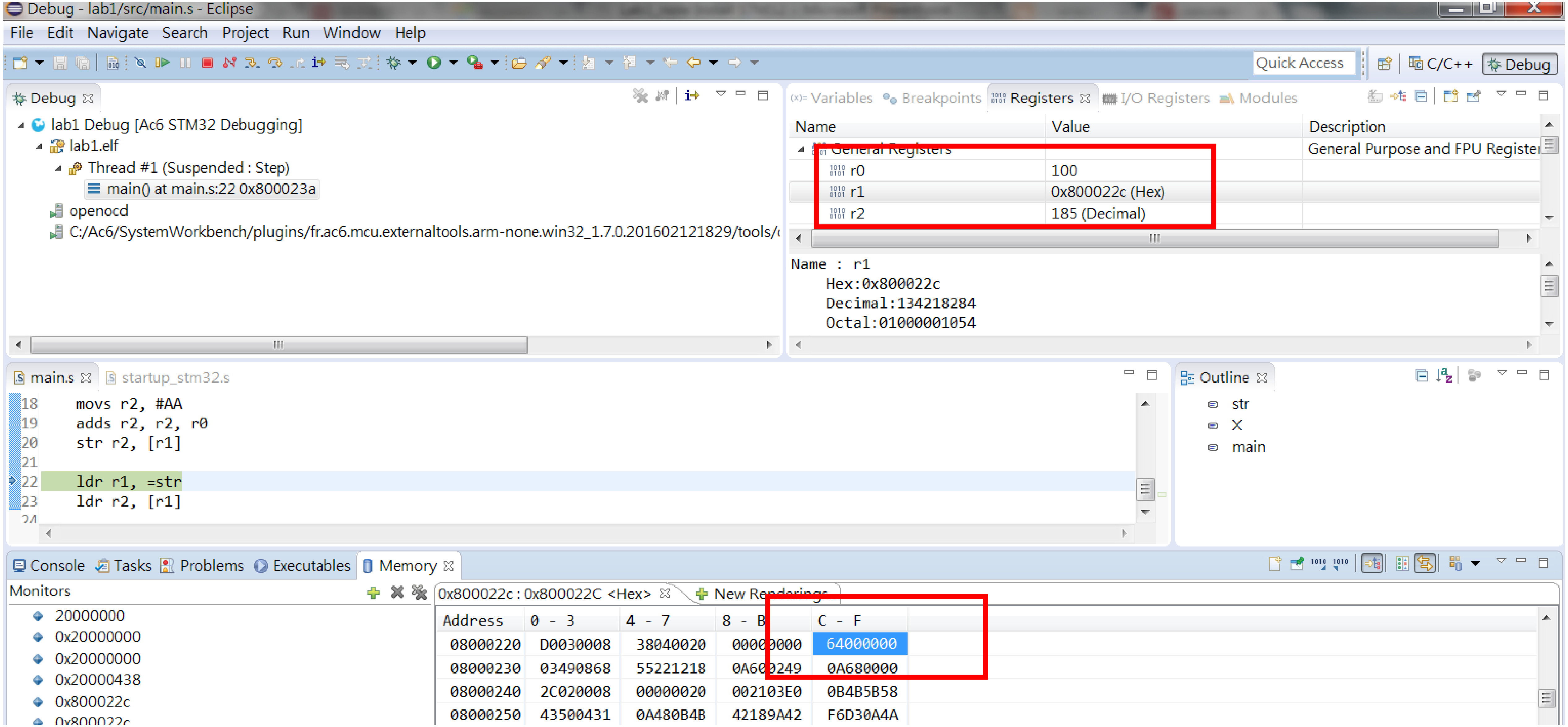
Outline: str, X, main

536870912

536870912 : 0x20000000 <Hex>

Address	0 - 3	4 - 7	8 - B	C - F
20000000	48656C6C	6F20576F	726C6421	00000000
20000010	00000000	FC020020	64030020	CC030020

留意reg裡面儲存的值，在address裡面是反過來放的
所以0x6c 6c 65 48 變成 0x48 65 6c 6c



```
lab1 Debug [Ac6 STM32 Debugging]
  lab1.elf
    Thread #1 (Suspended : Step)
      main() at main.s:22 0x800023a
  openocd
  C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829/tools/
```

Name	Value	Description
General Registers		General Purpose and FPU Registe
r0	100	
r1	0x800022c (Hex)	
r2	185 (Decimal)	

```
Name : r1
    Hex:0x800022c
    Decimal:134218284
    Octal:01000001054
```

```
    movs r2, #AA
    adds r2, r2, r0
    str r2, [r1]

    ldr r1, =str
    ldr r2, [r1]
```

```
Monitors
  20000000
  0x20000000
  0x20000000
  0x20000438
  0x800022c
  0x800022c
```

```
0x800022c : 0x800022C <Hex>

Address   0 - 3     4 - 7     8 - B     C - F
08000220  D0030008  38040020  00000000  64000000
08000230  03490868  55221218  0A600249  0A680000
08000240  2C020008  00000020  002103E0  0B4B5B58
08000250  43500431  0A480B4B  42189A42  F6D30A4A
```

2020 Spring 微處理機 LAB 1

PART 1. (50%)

1. 查閱programming manual，寫出MOV，STR，LDR用法與差異。(30%)

2. 舉一個暫存器間接定址法的程式碼並說明其運作過程。(20%)

PART 2. (50%) 實作題 請完成實驗 截圖紀錄實驗結果並附上程式碼

1. 組內組員，一人一題 (50%)
   a. 用組合語言寫出20H - 10H 並在register中追蹤其數值相加變化
   b. 用組合語言寫出5H x 9H 並在register中追蹤其數值相加變化

(請分別擷取計算前register中的值及計算後之值的變化)

PART 3. 加分練習，不計入平常成績

Fibonacci serial: 宣告一數值N (1≤N≤100)，計算Fib(N)並將回傳值存放至R4暫存器

- Tips: Fib(0) = 0 ; Fib(1) = 1 ; Fib(N) = Fib(N-1) + Fib(N-2) for N>1

各位同學,
LAB內容如Word 請依照格式填寫
繳交時請將作業pdf檔與main.s檔案壓縮後上傳
pdf及壓縮檔 檔名為 "學號+LAB1"
繳交截止為 2020/04/01 (三)
請各位準時上傳
助教