

# 嵌入式微處理器系統設計 – **ARM Developer Suite**

課程編號：EE5019701

授課教師：王乃堅 教授

課程助教：陳盈佑

# [ ADS 安裝 ]

- 安裝軟體：
  - 安裝ads1.2-demo資料夾中的Setup.exe

# ADS 簡介

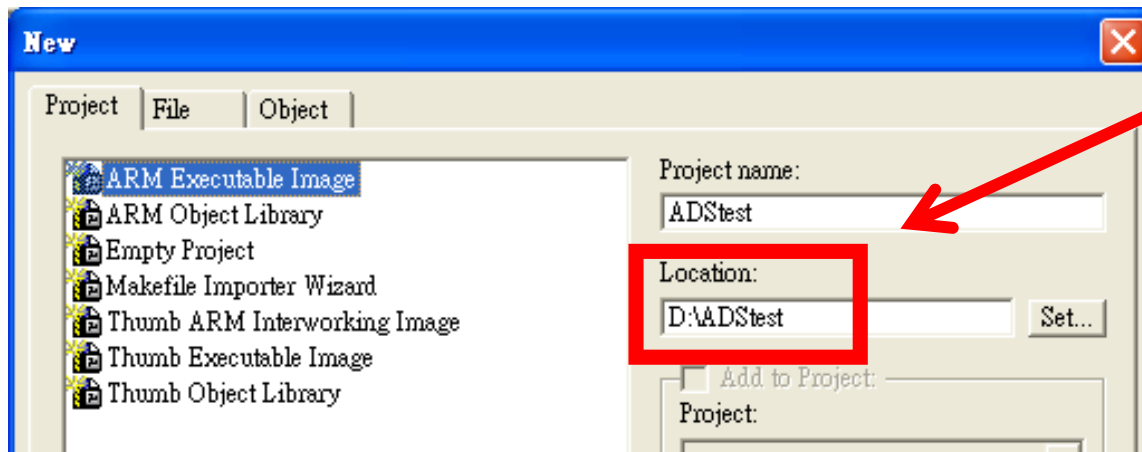
## ■ ARM處理器的開發工具ADS (ARM Developer Suite)

- ARM編譯器和ARM連接器組成的CodeWarrior IDE，這是一套用於程式編譯和程式連結的整合開發環境(IDE)。
  - 可以用於編寫ARM assembly code、標準C/C++程式碼，內建編譯器可以編/組譯這些檔案。
  - 內建連結器(linker)，可以將多個程式做連結成一個用於debug的image file(\*.axd檔)。
- ARM調試器AXD (ARM eXtended Debugger)，是一個功能強大的調試系統，不需要硬體支援的ARMulator的debug方式，可以單步除錯，看暫存器、記憶體的值、程式效能測試等輔助功能。

# [CodeWarrior-建立專案]

## ■ 在CodeWarrior新建project：

- 開啟CodeWarrior for ARM Developer Suite
- “File” > “New”
- 選”ARM Executable Image”
- “Project name”：輸入專案名稱
- “Set...”：保存的路徑

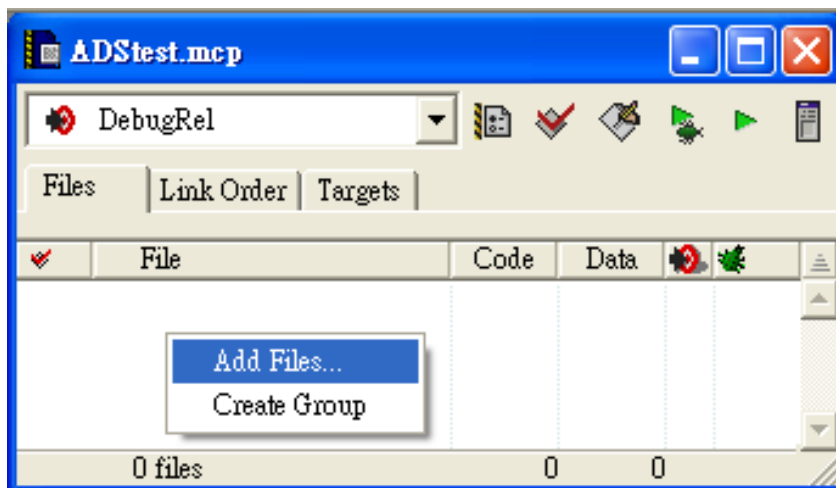


存檔位置

# CodeWarrior-加入已存在檔案

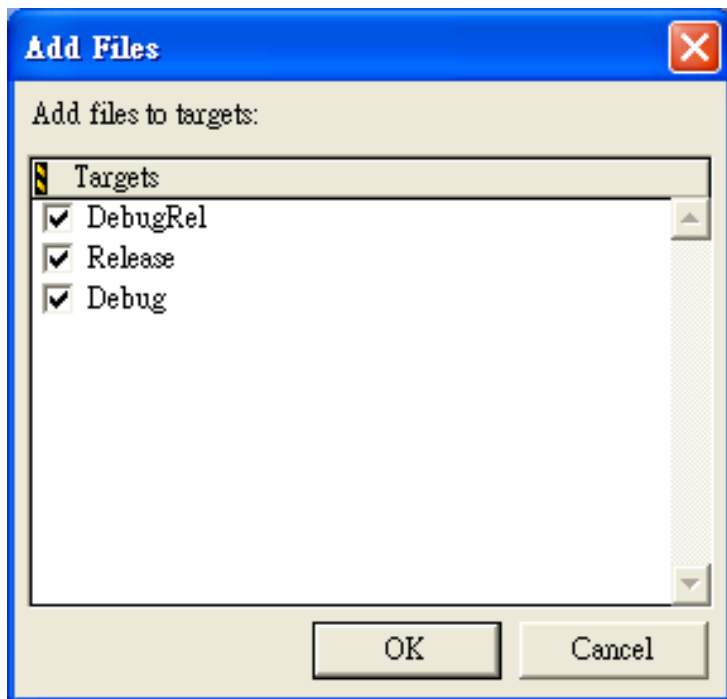
## ■ 加入.s檔：

- File標籤頁中 > 點選滑鼠右鍵
- “Project “ > “Add Files...”
- C:\Program Files\ARM\ADSV1\_2\Examples\asm\armex.s



# CodeWarrior-加入已存在檔案


- 3個target都要勾選：

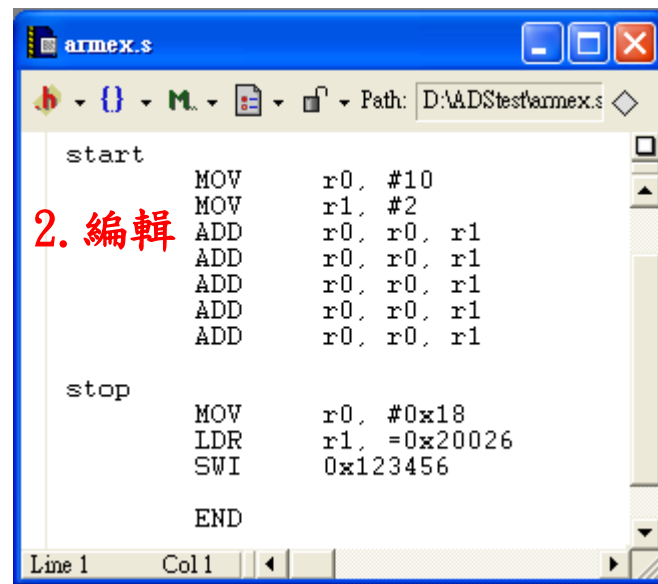
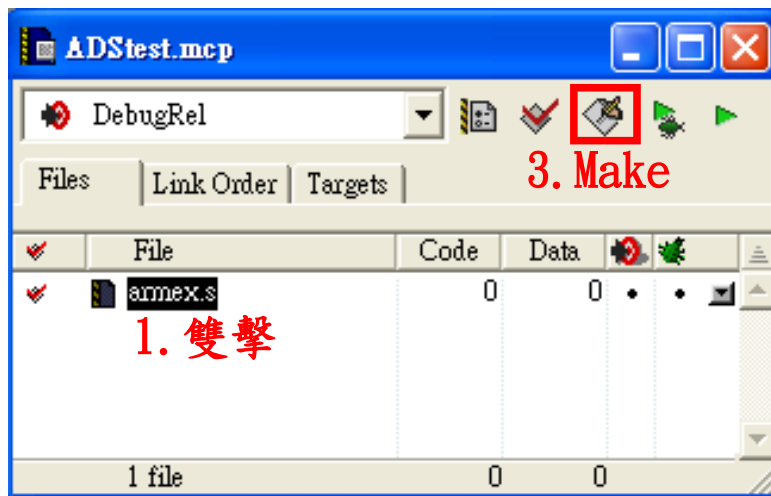


- **DebugRel :**  
使用該目標，在產生目標的時候，會為原始檔生成除錯資訊。
- **Release :**  
使用該目標不會生成任何除錯資訊。
- **Debug :**  
使用該目標為每一個原始檔生成最完全的除錯資訊。

# CodeWarrior-加入已存在檔案

## ■ 編輯與編譯:

- 雙擊檔案可開啟編輯視窗
- 編輯完成後可按F7或圖示進行”Make”
- Make完成會出現”Errors & Warnings”頁面



# CodeWarrior-新建原始碼檔案

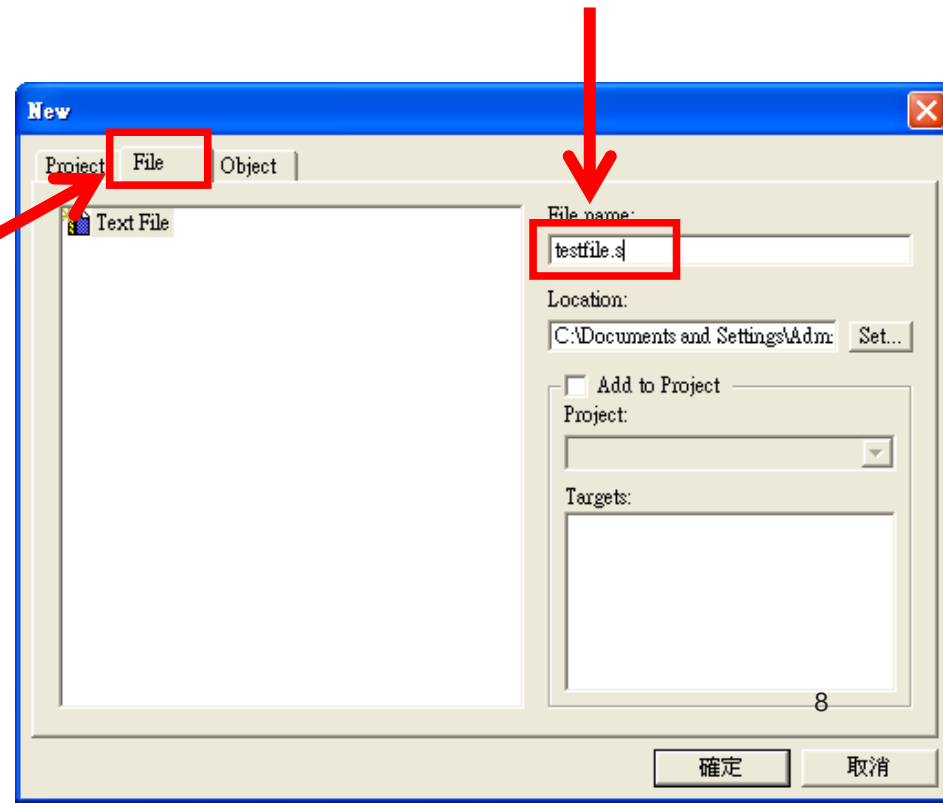
## ■ 在CodeWarrior新建原始碼檔案：

- “File” > “New”
- 選擇“File”頁面中的“Text File”
- “File name”：輸入檔案名稱
- “Location”：保存的路徑

檔案名稱須加上副檔名  
(.s)

例如:filename.s

File頁面

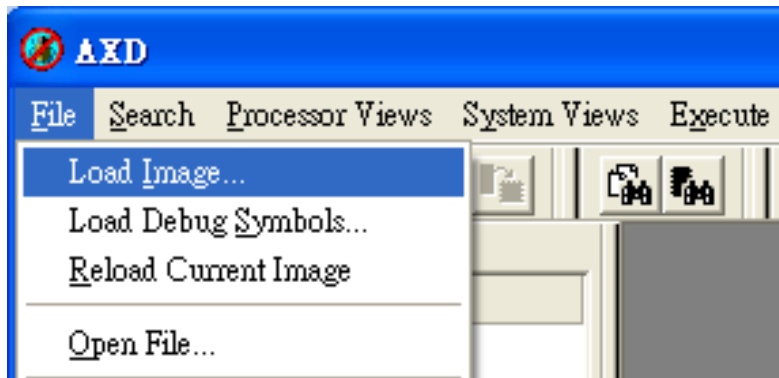




# [ AXD Debugger ]

## ■ 使用AXD Debugger除錯

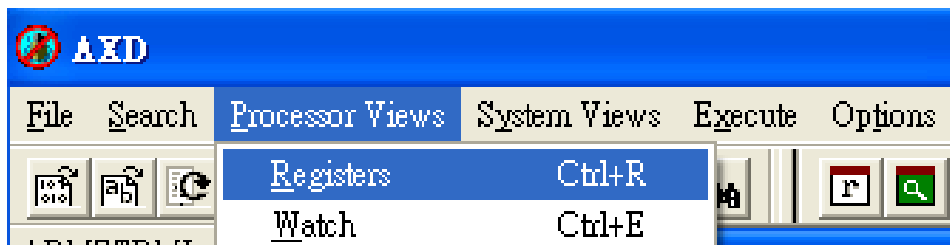
- 在CodeWarrior點選  來開啟AXD Debugger
- 或直接開啟AXD，點選”File” > “Load Image”
- 從D:\ADStest\ADStest\_Data\DebugRel\，載入ADStest.axf檔



# AXD Debugger

## ■ 暫存器觀察視窗：


- 點選“Processor Views” > “Registers”
- 或點選圖式 

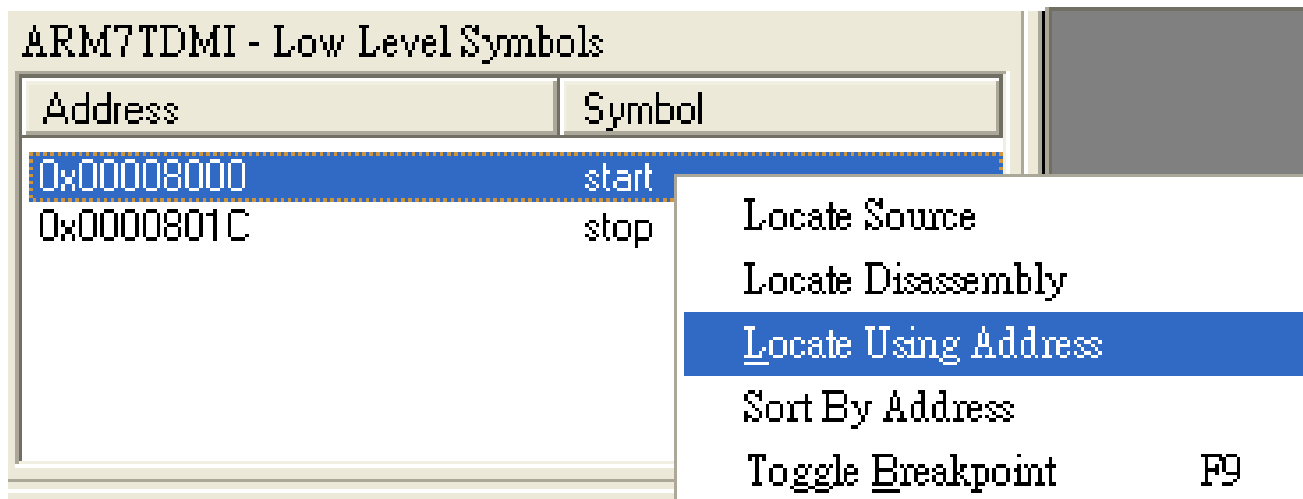


ARM7TDMI - Registers	
Register	Value
[-] Current	{...}
r0	0x00000000
r1	0x00000000
r2	0x00000000
r3	0x00000000
r4	0x00000000
r5	0x00000000
r6	0x00000000
r7	0x00000000
r8	0x00000000
r9	0x00000000
r10	0x00000000
r11	0x00000000
r12	0x00000000
r13	0x00000000
r14	0x00000000
pc	0x00008000
cpsr	nzcvqIFt_SVC
spsr	nzcvqift_Res

# AXD Debugger


## ■ Symbol觀察視窗

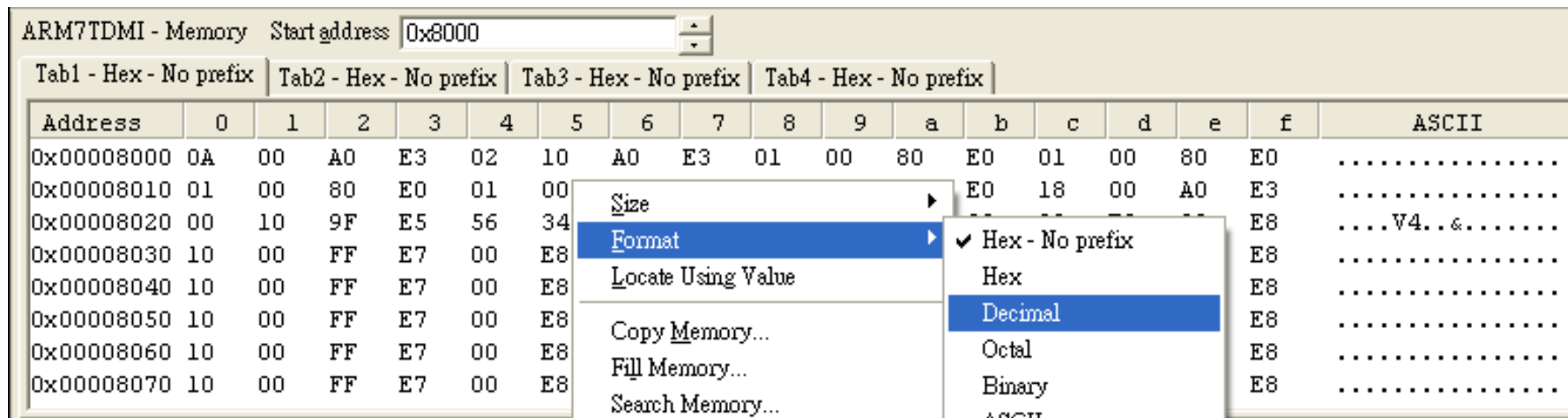
- 點選“Processor Views” > “Low Level Symbols”
- 或點選 圖示 
- 於Symbol上點”右鍵: > “Locate Using Address”，可以直接開啟該位置的Memory



# [ AXD Debugger ]

## ■ 記憶體觀察視窗

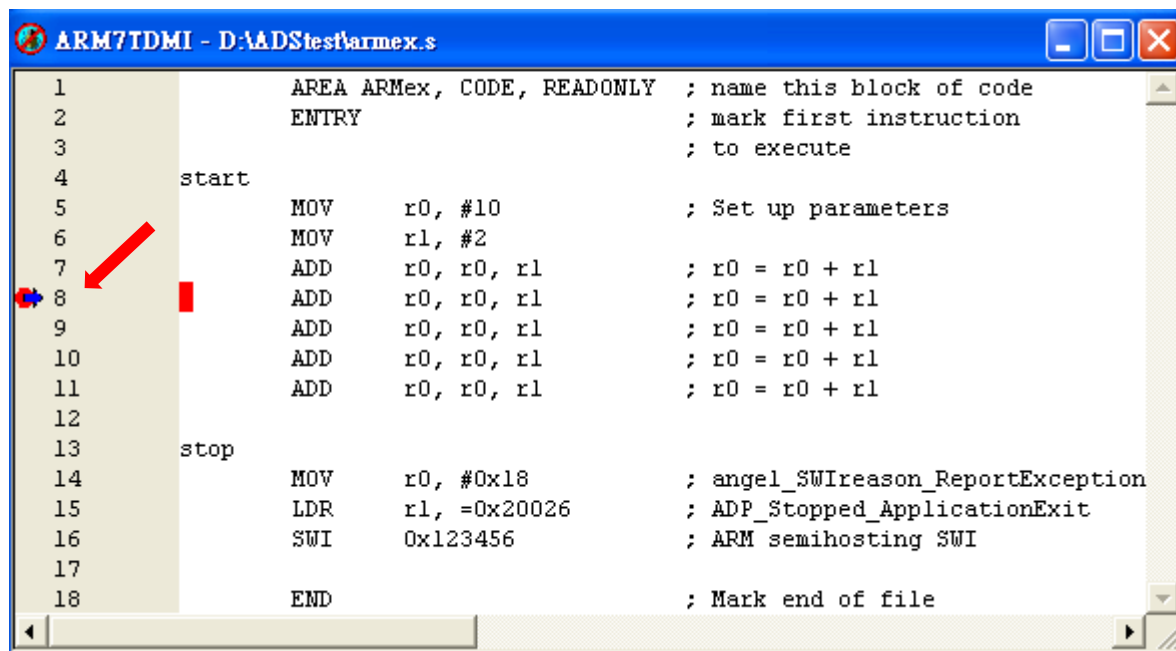
- 點選“Processor Views” > “Memory”
- 或點選 圖示 
- 可以點選”右鍵” > “Format” 來變更顯示格式



# [ AXD Debugger ]

## ■ 執行:

- 單步執行:F8
- 全部執行:F5
- 也可以在想要停下來的地方，點兩下設定中斷點

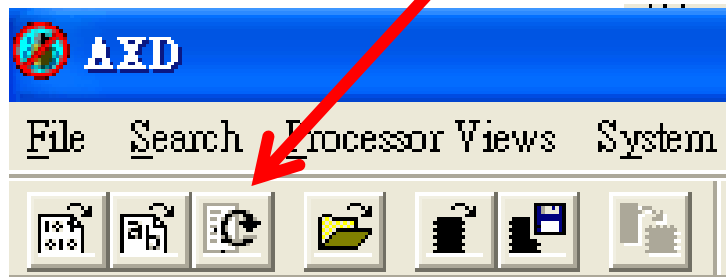


```
1      AREA ARMex, CODE, READONLY ; name this block of code
2      ENTRY                      ; mark first instruction
3                                  ; to execute
4      start
5      MOV     r0, #10             ; Set up parameters
6      MOV     r1, #2
7      ADD     r0, r0, r1          ; r0 = r0 + r1
8      ADD     r0, r0, r1          ; r0 = r0 + r1
9      ADD     r0, r0, r1          ; r0 = r0 + r1
10     ADD     r0, r0, r1          ; r0 = r0 + r1
11     ADD     r0, r0, r1          ; r0 = r0 + r1
12
13     stop
14     MOV     r0, #0x18            ; angel_SWIreason_ReportException
15     LDR     r1, =0x20026         ; ADP_Stopped_ApplicationExit
16     SWI     0x123456            ; ARM semihosting SWI
17
18     END                          ; Mark end of file
```

# [ AXD Debugger ]

## ■ 重置:

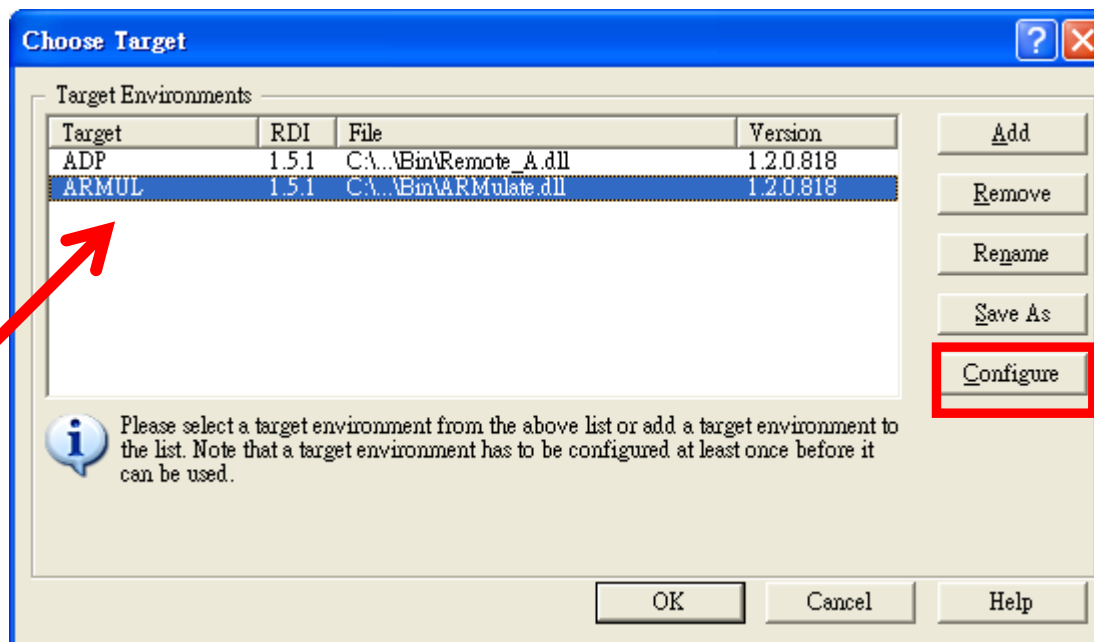
- 點選File -> Reload Current Image 或  圖示。



# [ AXD Debugger ]

## ■ 效率分析:

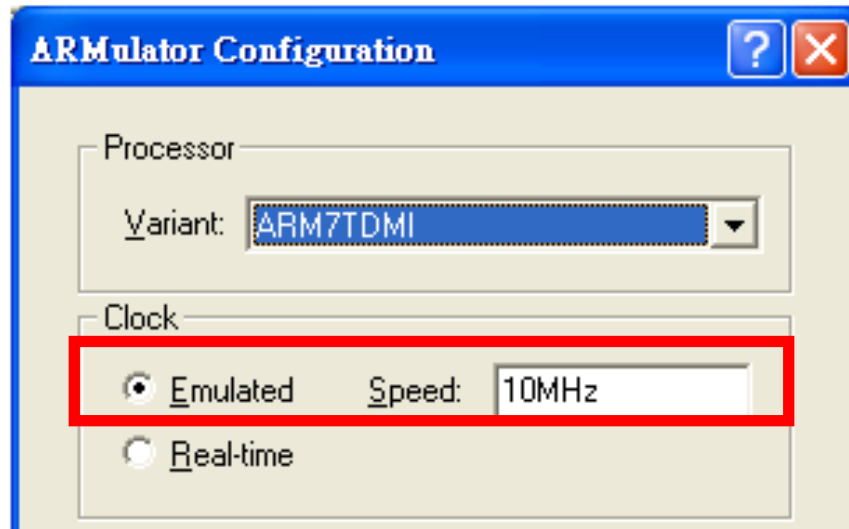
- 點選Options -> Configure Target ◦
- 選擇"ARMUL"再點選Configure ◦



# [ AXD Debugger ]

## ■ 效率分析:

- 在ARMulator Configuration視窗中。
- Clock選擇”Emulated”，Speed設定10MHz。

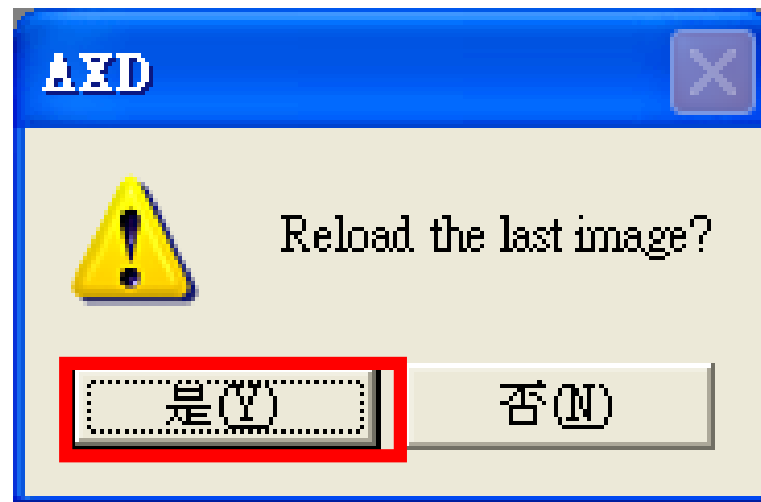




# [ AXD Debugger ]

## ■ 效率分析:

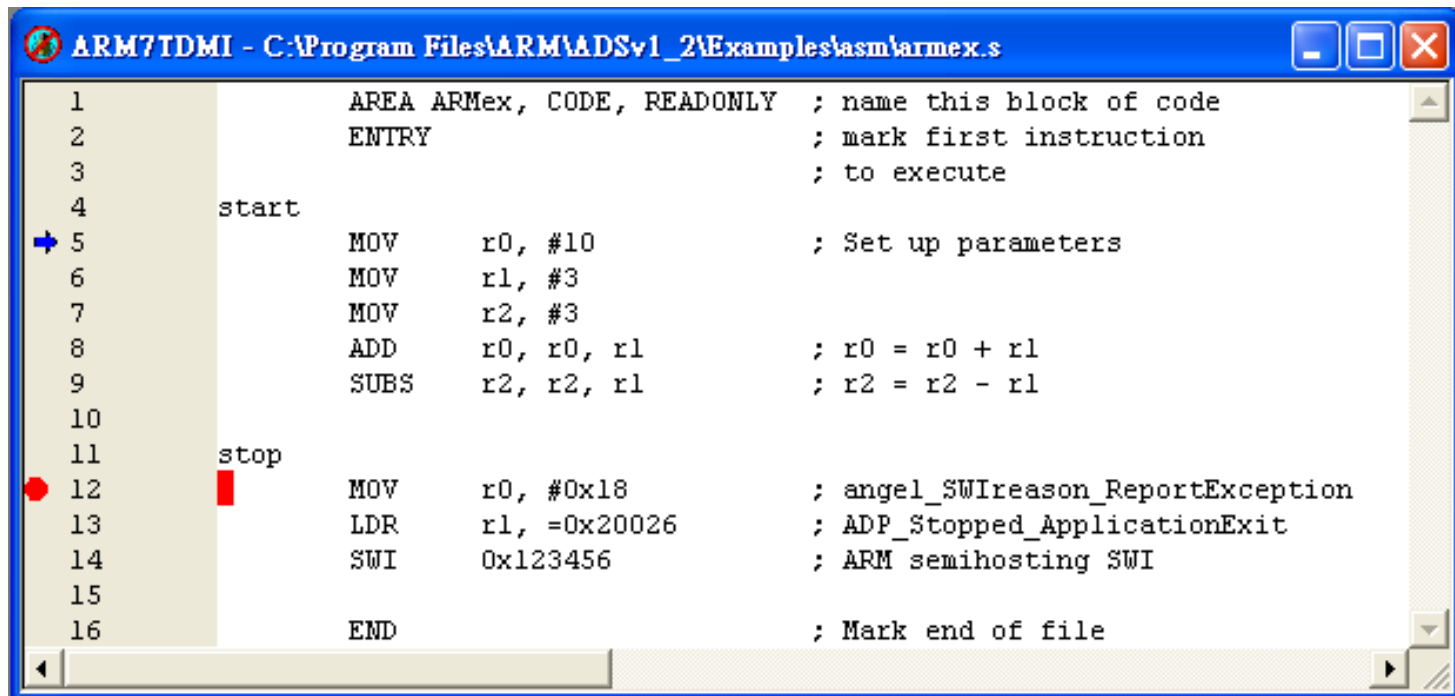
- 設定完成後按下確定。
- 彈出視窗詢問是否重新讀取image，選擇是。



# AXD Debugger

## ■ 效率分析:

- 設定程式執行斷點(量測結束點)。



The screenshot shows the ARM7TDMI AXD Debugger window. The title bar reads "ARM7TDMI - C:\Program Files\ARM\ADSv1\_2\Examples\asm\armex.s". The main window displays assembly code with line numbers 1 through 16. A blue arrow points to line 5, and a red dot marks line 12, indicating a breakpoint. The code includes instructions for setting up parameters and a semihosting SWI call.

```
1      AREA ARMex, CODE, READONLY ; name this block of code
2      ENTRY                      ; mark first instruction
3                                  ; to execute
4      start
5      MOV     r0, #10              ; Set up parameters
6      MOV     r1, #3
7      MOV     r2, #3
8      ADD     r0, r0, r1           ; r0 = r0 + r1
9      SUBS    r2, r2, r1           ; r2 = r2 - r1
10
11     stop
12     MOV     r0, #0x18             ; angel_SWIreason_ReportException
13     LDR     r1, =0x20026          ; ADP_Stopped_ApplicationExit
14     SWI     0x123456             ; ARM semihosting SWI
15
16     END                          ; Mark end of file
```

# [ AXD Debugger ]

## ■ 效率分析:

- System Views -> Debugger Internals 。
- 視窗點選標籤”Statistics” 。
- 在Statistics中空白處點右鍵-> Add New Reference Point 。
- 輸入名稱後按下確定 。



# AXD Debugger

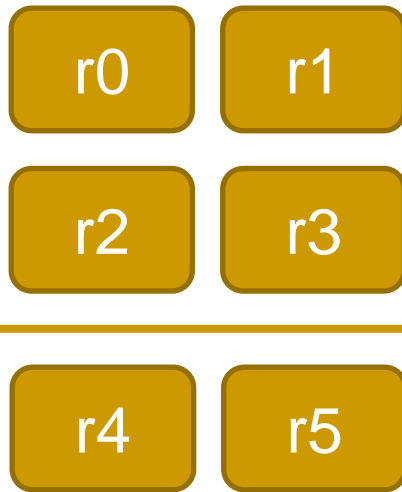
## ■ 效率分析:

- 完成後直接點選Execute ->Go。
- 程式執行至斷點停止。
- 觀看Statistics內所設定的Reference可得到時間資訊。

Debugger Internals							
Internal Variables				Statistics			
Ref...	Ins...	Cor...	S_...	N_...	I_...	C_...	Total
Statistics:5		8	7	1	0	0	8
time	5	8	7	1	0	0	8

# [ LAB1-1 ]

+



```
MOV    r0, #0
MOV    r1, #0xf7000000
MOV    r2, #0
MOV    r3, #0x09000000
```

Find r4, r5 ?

# [ LAB1-2 ]

## ■ 給定6筆數字， -10,11,20,50,-20,-3

1.做有號數的排序，

2.做有號數相加，查看是否有overflow

排序結果放ArrayB，overflow放R5，Sum放R7

3.做無號數的排序

4.做無號數相加，查看是否有overflow

排序結果放ArrayC，overflow放R6，Sum放R8

```
        AREA Matrix, CODE, READONLY ; mark first instruction
        ENTRY
start
        LDR r0,=ArrayA
stop

        AREA Data, DATA, READWRITE
ArrayA dcd -10,11,20,50,-20,-3
        END
```