

ESW聯盟「嵌入式系統與軟體工程」

取像Imaging 數位相機系統設計

課程：嵌入式系統與軟體工程

開發學校：中央大學資工系

陳慶瀚



實驗目的

- ▶ 於ARM平台進行影像操作(image manipulation)

實驗步驟

- ▶ (1) 影像的格式轉換
- ▶ (2) 演算法程式porting
- ▶ (3) 測試及debug

影像的格式轉換

- ▶ 先在Keil ARM中做模擬，因此影像要事先擺放到模擬的RAM當中，但是在這之前要先把現有的BMP圖檔轉換成Intel HEX FILE的格式，這部份需要查詢相關文件並且寫出格式轉換的程式，因此將這個工作獨立出來。

演算法程式porting

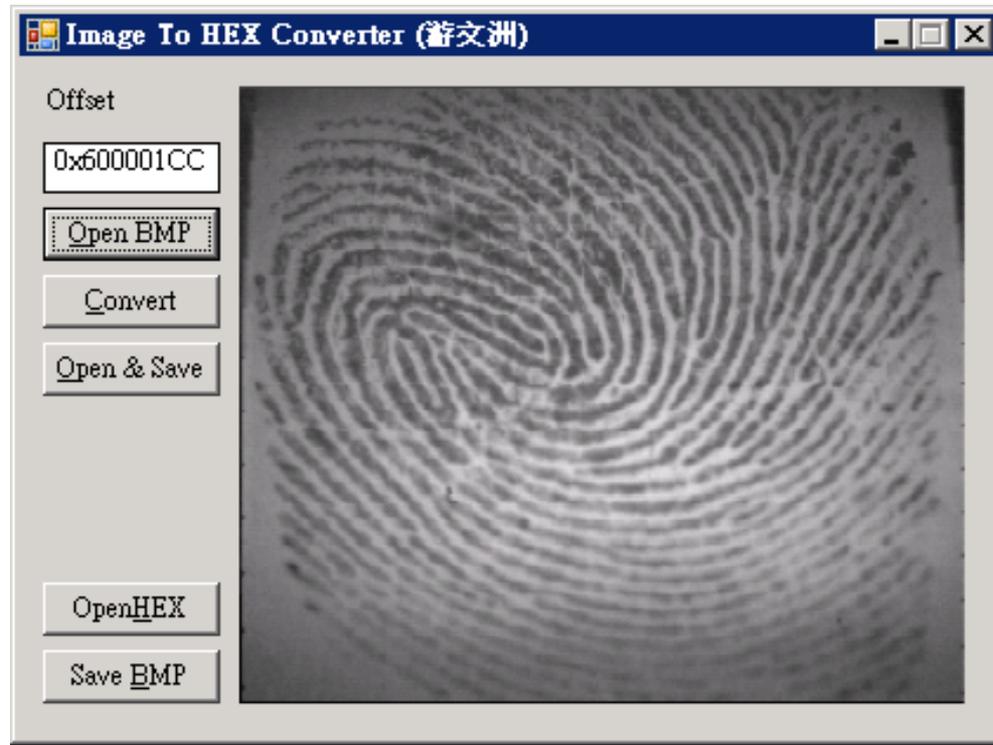
- ▶ 要將整個程式的演算法瀏覽過，清楚每個header檔案提供function的運作方式，並且在不改變處理結果的情況下，將程式碼修改成純C的格式，並且把程式碼移植到ARM上編譯。

測試及debug

- ▶ 剛移植好的程式碼處理完的結果可能會跟原先的不一樣，因此需要在程式碼porting以後反覆的做測驗，將錯誤的function找出，並且想辦法改正執行的結果，這部份需要跟程式碼porting的人一起合作。

步驟1、影像轉檔

- 將程式的輸入從圖檔輸入改為陣列輸入，利用自製程式把BMP檔案轉成Intel HEX檔格式。



步驟2、了解Intel hex file格式

- Intel hex file有多種模式。但此處只需要用到Data record (00)、Extended Linear Address Records (HEX386)(04)、End-of-File (EOF) Records(01)

以下為一個Intel Hex File範例：

:10001300AC12AD13AE10AF1112002F8E0E8F0F2244

:10000300E50B250DF509E50A350CF5081200132259

:03000000020023D8

:0C002300787FE4F6D8FD7581130200031D

:10002F00EFF88DF0A4FFEDC5F0CEA42EFEEEC88F016

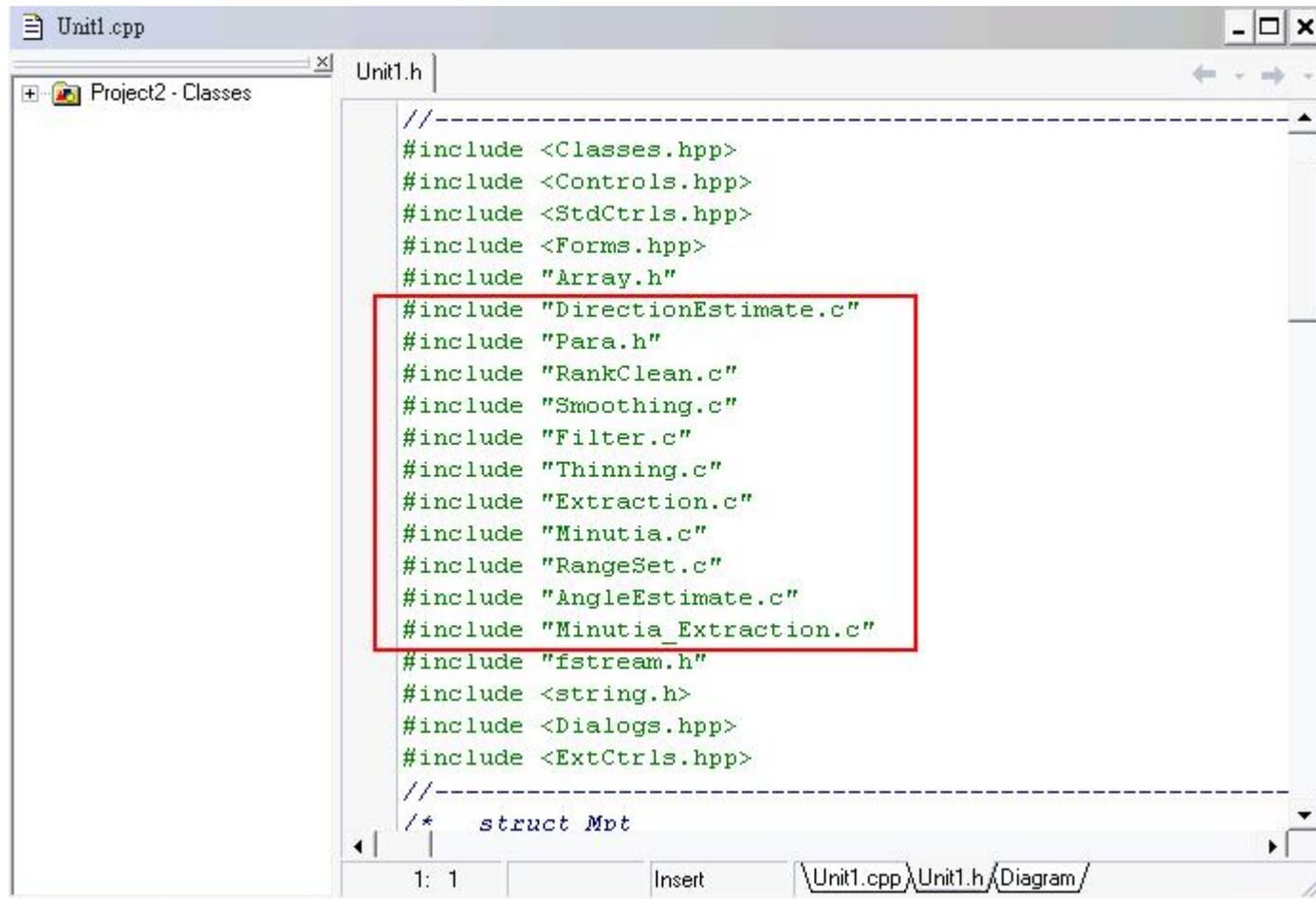
:04003F00A42EFE22CB

:00000001FF

步驟3、轉換實務

- 先以指令save儲存Intel Hex File內容陣列的.hex檔，並取出記憶體的位置。實際上，一個data record最多可以儲存FFh個bytes，我們便以此格式轉換bmp至Intel Hex File。特別注意，Data record中的address有4 bits，當累加超過FFFF時，亦即資料超過64K，則需要再寫入Extended linear address records模式的record，使Data record的address能夠在4 bits中表現出來。
- 轉換後的Intel Hex File，則利用指令load並給定記憶體位置直接放入記憶體中，即可完成。

步驟4、Porting(一)



```
//-----  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include "Array.h"  
#include "DirectionEstimate.c"  
#include "Para.h"  
#include "RankClean.c"  
#include "Smoothing.c"  
#include "Filter.c"  
#include "Thinning.c"  
#include "Extraction.c"  
#include "Minutia.c"  
#include "RangeSet.c"  
#include "AngleEstimate.c"  
#include "Minutia_Extraction.c"  
#include "fstream.h"  
#include <string.h>  
#include <Dialogs.hpp>  
#include <ExtCtrls.hpp>  
//-----  
/* struct Mpt
```

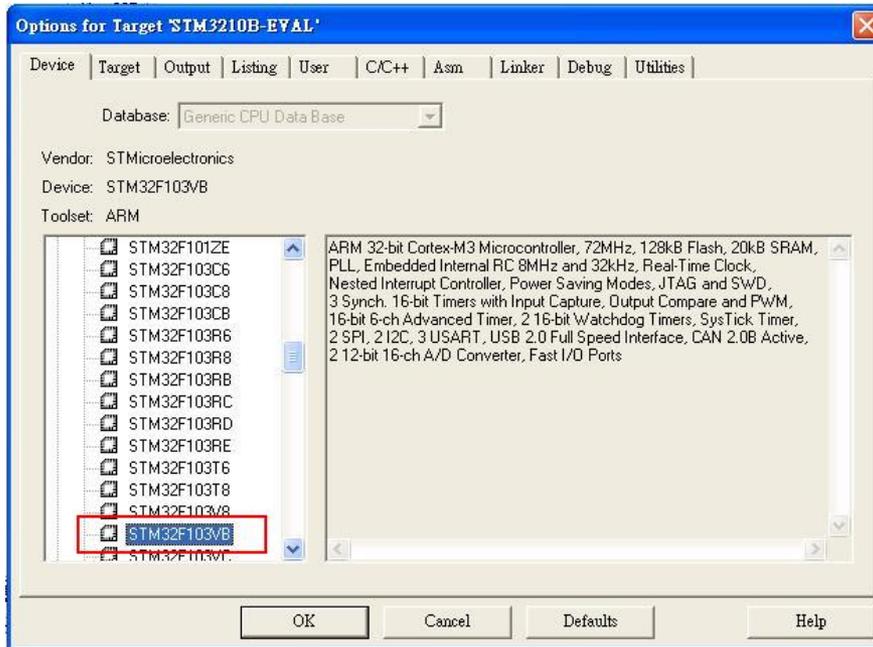
步驟5、Porting (二)

- 把修改過的code在Keil ARM上面編譯，一開始先只要在main中include就好，因為還不知道剛剛修改完的header檔案能不能夠編譯過，等編譯過以後再進行查核：
 1. 若只有.h檔案，則就在main中宣告” include “xxx.h” ” 如果有.h又有.c檔案，則在左邊的Project Workspace中要加入.c
 2. 檔，這時候Keil就會把所以這個.c檔所引用的檔案都拉近下拉式檔案中。而.h檔案則在main中引用。
 3. 若只有.c檔，則在左邊的Project Workspace中加入，但是因為沒有.h，所以上層在引用的時候，要寫extern type function(type)。

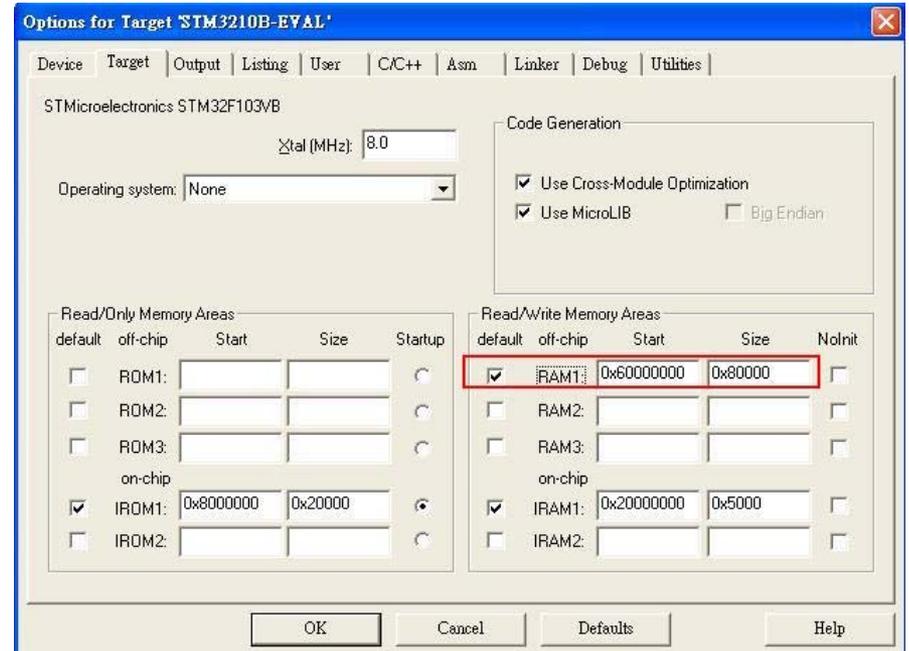
步驟5、Porting (二)

- 把修改過的code在Keil ARM上面編譯，一開始先只要在main中include就好，因為還不知道剛剛修改完的header檔案能不能夠編譯過，等編譯過以後再進行查核：
 1. 若只有.h檔案，則就在main中宣告” include “xxx.h” ” 如果有.h又有.c檔案，則在左邊的Project Workspace中要加入.c
 2. 檔，這時候Keil就會把所以這個.c檔所引用的檔案都拉近下拉式檔案中。而.h檔案則在main中引用。
 3. 若只有.c檔，則在左邊的Project Workspace中加入，但是因為沒有.h，所以上層在引用的時候，要寫extern type function(type)。

步驟6、配置記憶體



選擇Processor的型號



選擇記憶體區塊

步驟7、驗證

```
compiling stm32f10x_it.c...
compiling main.c...
compiling AngleEstimate.c...
Lib\AngleEstimate.c(136): warning: C3017W: Ang may be used before
compiling DirectionEstimate.c...
compiling Extraction.c...
compiling Filter.c...
compiling LocalM.c...
compiling Minutia.c...
compiling Para_Init.c...
compiling RangeSet.c...
compiling RankClean.c...
compiling Smoothing.c...
compiling Thinning.c...
assembling stm32f10x_vector.s...
linking...
Program Size: Code=14096 RO-data=760 RW-data=360 ZI-data=417504
```