



MICROCHIP

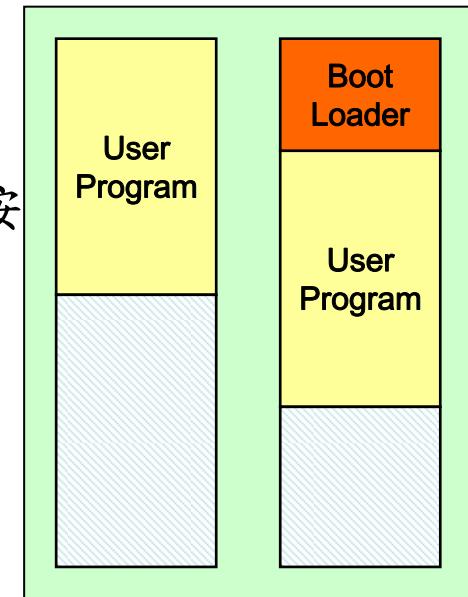
Regional Training Centers

使用 Microchip USB HID Boot Loader

Authors: Adam Syu
Microchip Technology Inc.

Boot-Loader 的運作原理

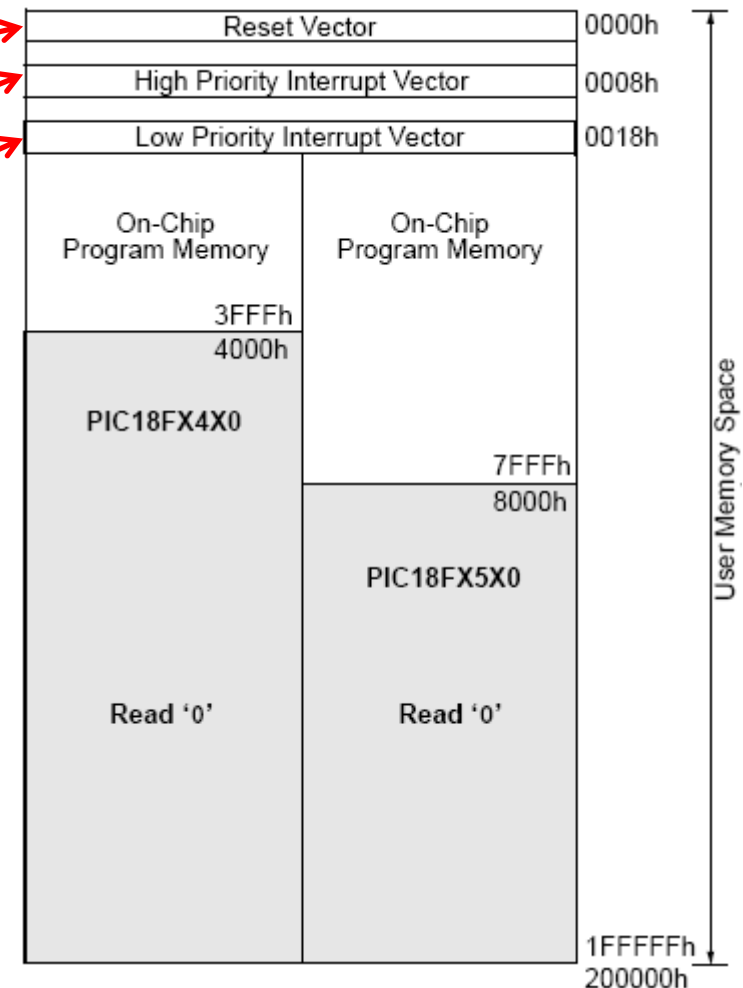
- 一般的程式記憶體的安排方式應如下圖所示, 未使用Boot-Loader時安排狀況如左圖所示; 具有Boot-Loader時之安排狀況應如右圖所示。
- 未使用Boot-Loader時, 程式記憶體中, 僅有User Program, MCU一旦完成Reset後, 就會立即執行User Program。
- 而使用Boot-Loader時, 程式記憶體中會同時存在有User Program 與Boot-Loader。
- MCU Reset後, Boot-Loader會透過某些機制, 如按鈕或特殊記憶體的 Pattern, 判斷要進入 Boot-Loader 來載入新的 User Program 或者直接將執行權交給使用者程式執行。
- 由於Boot-Loader佔據程式記憶體前端部分, 因此使用者程式必須告知程式連結器 (Linker) 將自己移位至較高的位址, 避開 Boot-Loader 空間。



程式記憶體配置狀況

PIC18F 的程式記憶體配置

- PIC18 系列 MCU 在 Reset 後的 Program Counter 指向 0x0000
- MCU 有兩個中斷向量
 - 0x0008 - 高優先權中斷
 - 0x0018 - 低優先權中斷
- 中斷後 MCU 的 Program Counter 直接被改變為 0x0008 or 0x0018
- 因為 0x0008 ~ 0x0018 的區間只有 16 Bytes，故一般的中斷向量中只有一個簡單的“goto”指令來令 MCU 跳至指定的中斷服務程式

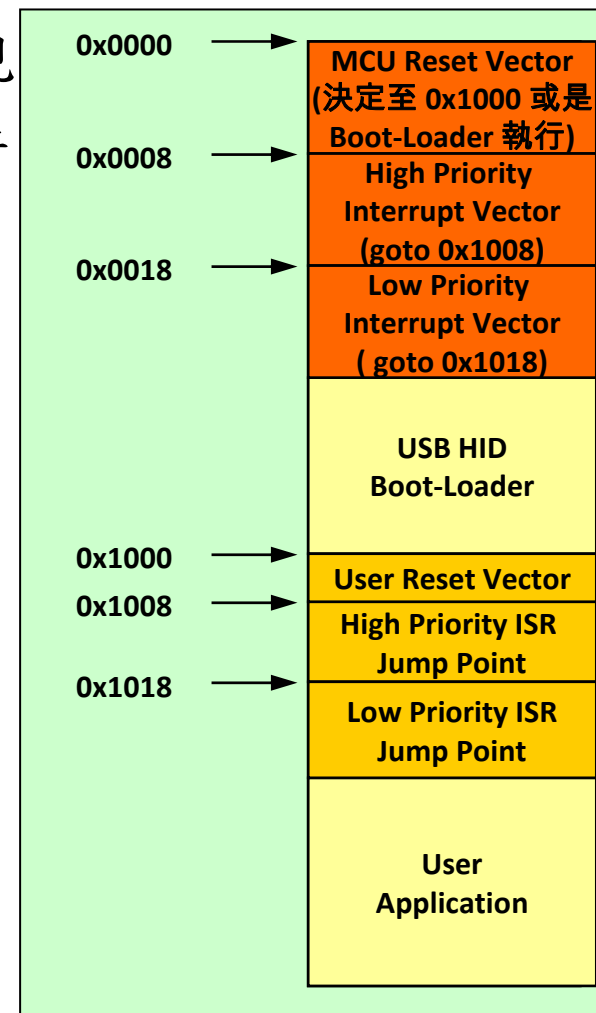


PIC18F Boot-Loader 與 User Program 並存時的記憶體規劃

- 加入 HID BootLoader 的 PIC18 系列記憶體規劃如圖所示。HID Boot Loader 的放置位置在最前面的 4K Bytes，User Application則在 0x1000 之後。
- CPU Reset 後，在 0x0000 開始執行並判斷要將控制權交由 User Program (0x1000) 或是 Boot-Loader

```
void main(void)
{
    TRISAbits.TRISA4 = 1;    // RA4 為 sw3 按鍵
    if(sw3 == 1)
    {
        _asm      goto 0x1000 _endasm
    }

    // 以下為 Boot-Loader 程式的剩餘部分！
    .....
}
```



Boot-Loader 使用時注意事項

- Boot-Loader :
是一個預先載入在MCU內的程式,該程式可透過通訊介面,如:UART, SPI, USB等,與PC或其他控制器溝通,進行自我韌體更新。
- Microchip對所有具有USB功能的MCU都有提供USB HID Boot-Loader的功能,因此可以輕易的進行自我韌體更新。
- USB HID透過USB介面與PC連接,搭配 Microchip 所提供之PC端軟體將欲更新的 F/W 燒錄至 MCU。如此將無須再使用 PICkit3/ICD3/Real ICE等開發工具進行 F/W 的燒錄。
- MCU 再出廠時是空白的,所以 Boot-Loader本身是透過PICkit 3、ICD 3 或是 Real ICE 等工具燒錄至MCU 的。
- 一旦 MCU 已經燒錄了 Boot-Loader,使用者的應用程式則需透過PC端的軟體經由Boot-Loader具備的燒錄功能,更新至 MCU中。
- 請不要在具有Boot Loader程式的MCU上使用PICkit3/ICD3/Real ICE等工具燒錄,此舉可能會覆蓋掉原來Boot-Loader的程式。

Reset, Interrupt 的重新導向

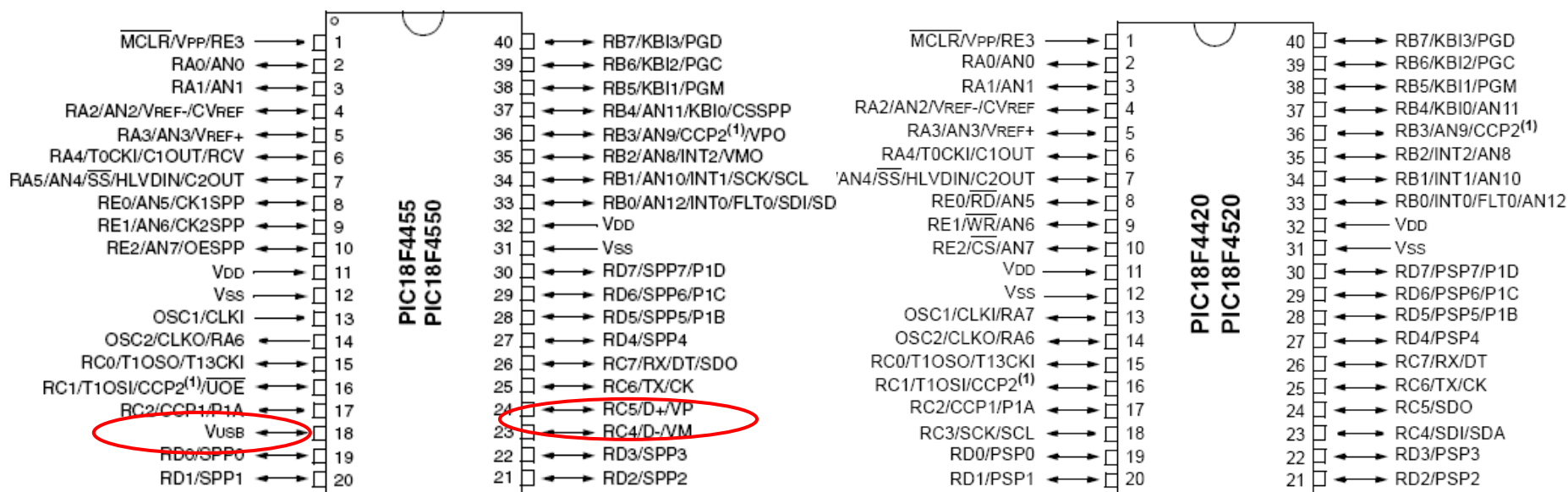
- 通常MCU中的Reset Vector, Interrupt Vector, Trap Vector等等是固定在特定的程式記憶體位置。這些位置是無法移動的。因此無法直接移位, 必須透過特定手段來處理。最常見的方式是使用多次跳躍的方式來導向(不同系列MCU處理方式不同)。
- 為了避免占用 User Program 執行時的 CPU 資源, Microchip 提供的 Boot-Loader 程式皆不使用中斷。
 - 直接以既定的結構安排將中斷導向至 User Program 安排好的中斷進入點
- 所以, 除了將 User Program 正確位移外, 安排 User Program 的中斷“進入點”至指定的新位址, 並且模擬正常程式執行時較用 C 語言啟動模組的動作是讓 User Program 能運作無誤的要件。
- 在不同系列的MCU中, 轉向概念是一樣的, 但手法不同。例如: MPLAB C18需使用linker script 檔案搭配程式碼來完成。MPLAB C30, C32則單純使用linker script 檔案即可完成。

USB HID Boot-Loader 取得方式

- USB HID Boot-Loader的相關資料, 包含在Microchip Application Libraries中, 可至專屬網頁 (<http://www.microchip.com/mal>)下載。
 - 本課程使用版本為Microchip Application Libraries v2011-06-02。
- Microchip Application Libraries包含所有Microchip的應用範例, 如USB, Graphics, TCP/IP, mTouch等。
- 檔案安裝後, 會在C磁碟機下產生Microchip Solutions xxxx資料夾。USB HID Boot-Loader的原始程式碼位於:
<C:\Microchip Solutions v2011-06-02\USB\Device - Bootloaders\HID\>
- 資料夾下有適用於不同MCU系列的USB HID Boot-loader原始碼。
- \TLS2118\HEX_Bootloader\ 中有 PIC18F4550 Bootloader 的 .hex

PICF4550 vs. PIC18F4520

- PIC18F4550 的 USB 信號腳位為 RC4 & RC5
- PIC18F4550 的 Pin-18 為 VUSB – 須加一個 0.47uF 電容

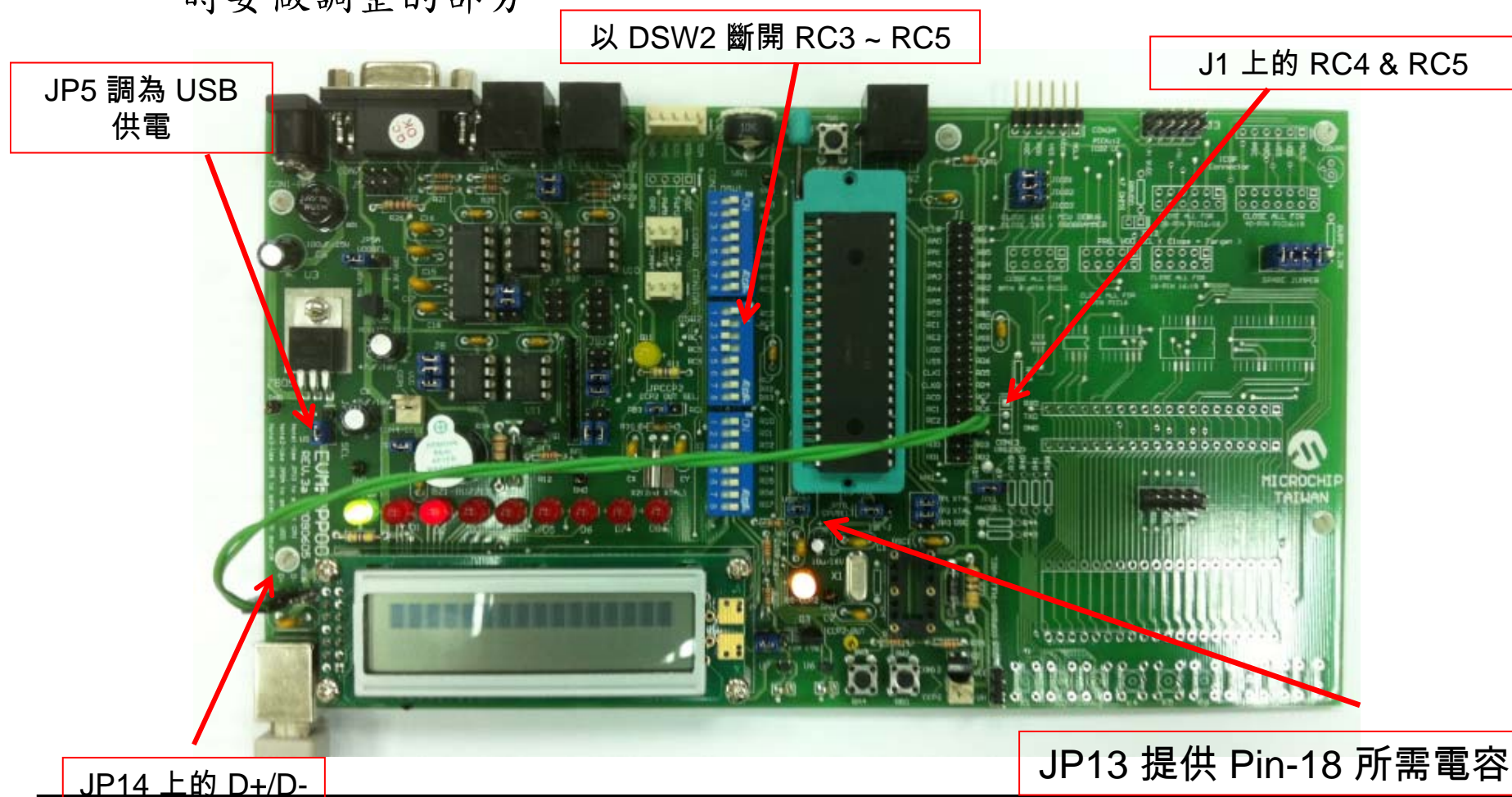


USB HID Boot-Loader & APP001

- 因為 Microchip MCU 出廠時的 Program 皆為空白，所以須先使用燒錄器將 USB HID Boot-Loader 燒錄至 MCU
- 本實驗使用以下的工具來載入 USB HID Boot-Loader
 - APP001T 實驗板 + PIC18F4550 MCU
 - PICkit 3 Debugger/Programmer
 - TLS2118\HEX_Bootloader\ 目錄中所附的 .hex 檔
- APP001T 出廠所裝置的 MCU 為 PIC18F4520
 - 須更換為 **PIC18F4550**
 - **CON5** 為 USB Connector，提供與 PC 的 USB 連結
 - **JP14** 中有將 USB 的接腳信號引出，可被連接至 MCU 的 D+/D-
 - **J1** 可以由 User 自由地將 MCU 信號自行跳接
 - **DSW1 ~ DSW3** 可以隔離 MCU 至板上周邊的連接
 - **JP13** 可以提供 PIC18F4550 Pin-18 所需連接電容的需求

PIC18F4550 使用於 APP001 時實驗板上需要的調整

- 以下為將 PIC18F4550 換裝上 APP001 後，與使用 USB HID Bootloader 時要做調整的部分



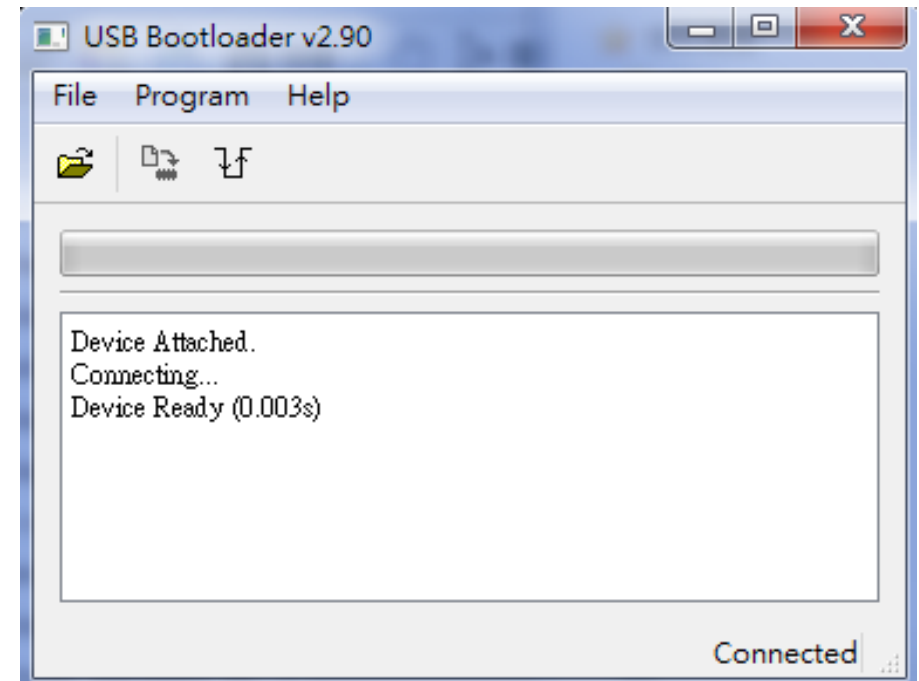
載入 USB HID Boot-Loader 程式至 PIC18F4550

- 在 MPLAB IDE 下選擇 PIC18F4550 為 Target MCU
- 使用 File -> Import 功能將以下檔案匯入
 - TLS2118\PIC18F4550_BootLoader\HID Bootloader
PIC18F4550.hex
- 選擇 PICkit 3 為 Programmer
- 將匯入的 .hex 檔案燒錄至 PIC18F4550
- 因為 USB HID Boot-Loader 使用的是 HID 類別的裝置，所以無須驅動程式的安裝
- 但是 ... 正確的程式才能進入 Boot-Loader !!
 - Sw3 (RA4) 在 MCU Power On 時要被按下 ...

USB HID Boot-Loader

程式進入方式及 GUI 程式的使用

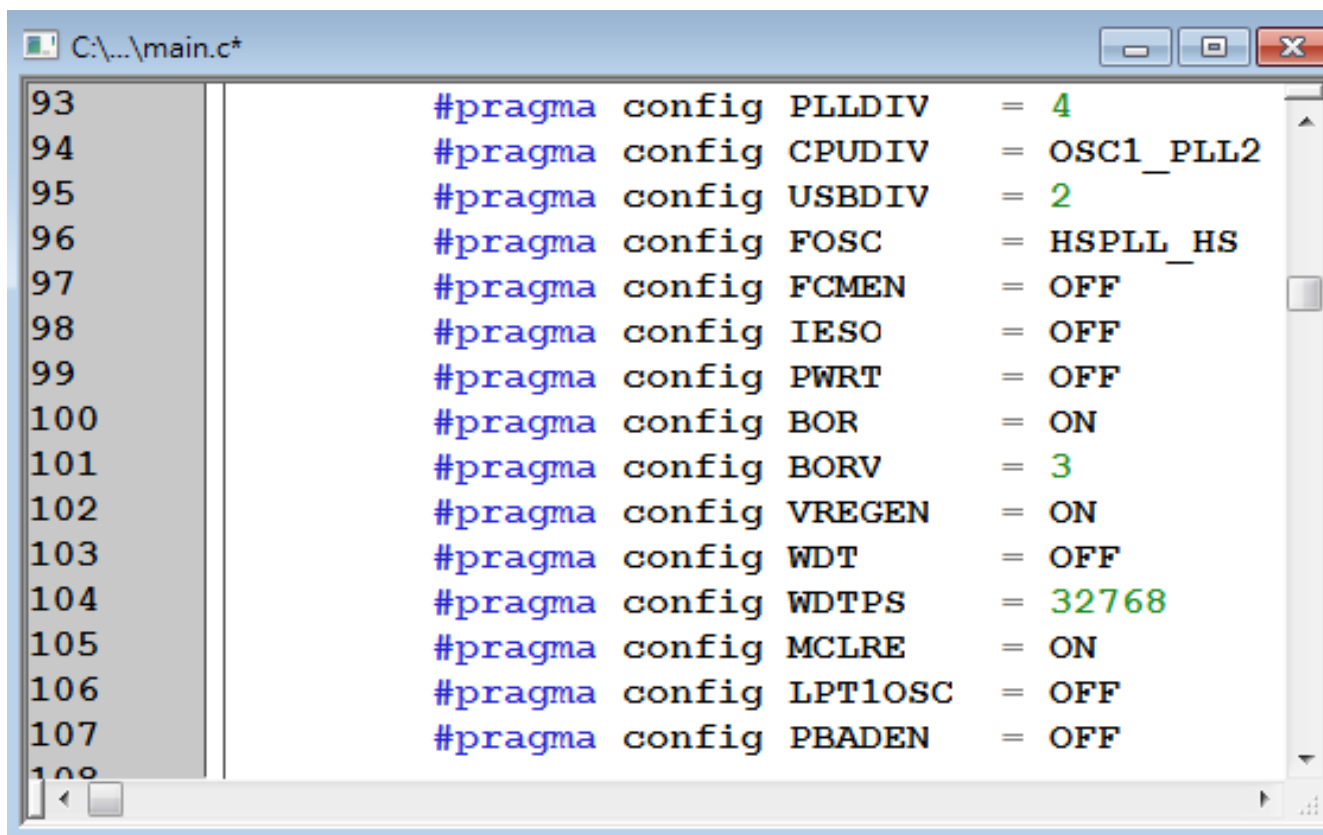
- 先執行以下的應用程式
 - **TLS2118\GUI_HIDBootLoader\HIDBootLoader.exe**
- 進入 Boot-Loader 的程序如下
 - 同時按下 SW1(RESET) & SW3 (RA4)
 - 先放開 SW1 候再放開 SW3
- Device Attached 的訊息將顯示於應用程式視窗
- 此時將可使用此 PC GUI 將 User Program 的 .hex 檔案載入並燒錄至 MCU



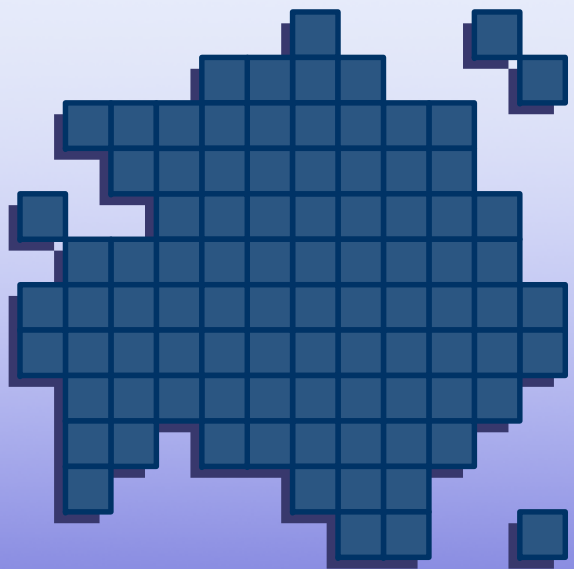
USB HID Boot-Loader

Configuration Bits 的設定值

- Boot-Loader 的 Configuration 設定如下
- User Program 不需要另外設定 Configuration Bits !!



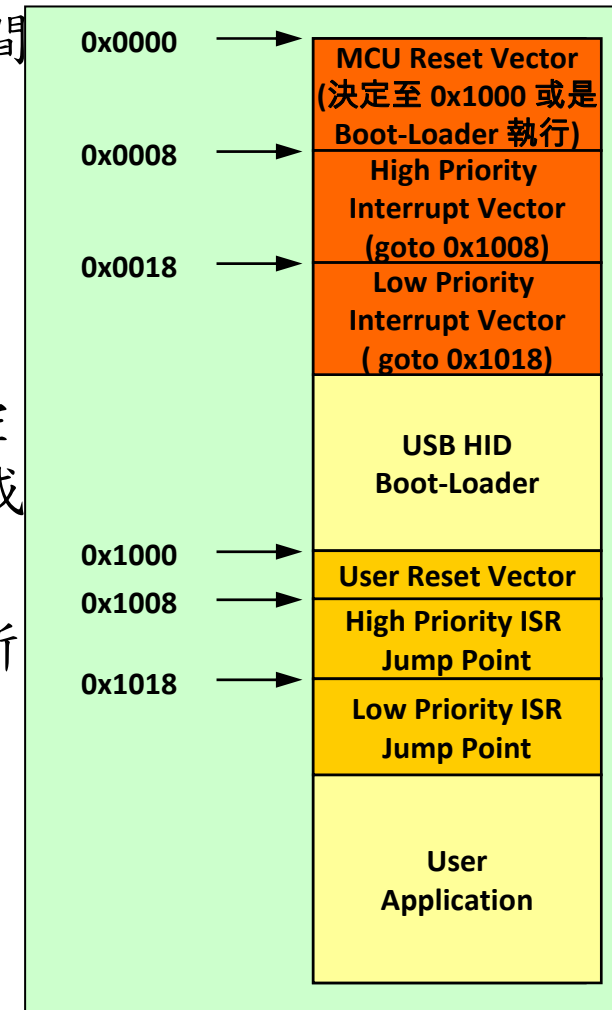
```
C:\...\main.c*
93      #pragma config PLLDIV      = 4
94      #pragma config CPUDIV      = OSC1_PLL2
95      #pragma config USBDIV      = 2
96      #pragma config FOSC        = HSPLL_HS
97      #pragma config FCMEN       = OFF
98      #pragma config IESO        = OFF
99      #pragma config PWRT        = OFF
100     #pragma config BOR          = ON
101     #pragma config BORV         = 3
102     #pragma config VREGEN        = ON
103     #pragma config WDT          = OFF
104     #pragma config WDTPS         = 32768
105     #pragma config MCLRE         = ON
106     #pragma config LPT1OSC       = OFF
107     #pragma config PBADEN        = OFF
108
```



適用於 USB HID Boot Loader 的 User Program 程式注意事項

PIC18F User Program 注意事項

- USB HID Boot-Loader 所佔的程式記憶體區間為 MCU 的最前面 4K Bytes (0x0000 ~ 0x0FFF)
- USB HID Boot-Loader 程式若要進入 User Program，會將控制權交給位址 0x1000
- 因為 Interrupt Vector 的程式進入是由硬體控制的，所以當中斷發生一定會跳到 0x0008 或是 0x0018 來執行
- Boot-Loader 沒有用到任何中斷資源，在中斷向量上直接將程式導向至已知位址
 - 0x0008 -> 0x1008
 - 0x0018 -> 0x1018
- 所以 User Program 要依據以上的規則來放置必要的程式碼來完成必要的作業！



PIC18F User Program

對位址 0x1000 的處置方式

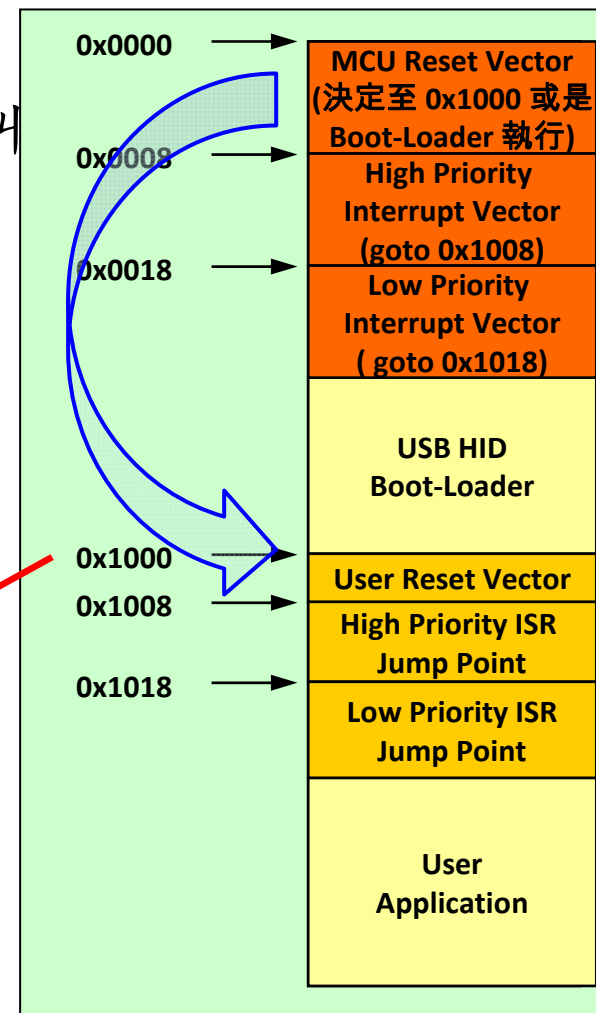
- `_startup()` 為 MPLAB C18 啟動模組的進入點，所以在 User Program 的開頭 0x1000 要叫用 `_startup()` 以便執行必要的初始工作
- `_startup()` 在完成初始作業後會將程式控制權交給 `main()`

```
extern void _startup (void);

#pragma code REMAPPED_RESET_VECTOR = 0x1000

    void _reset (void)
    {
        _asm goto _startup _endasm
    }

#pragma code
```

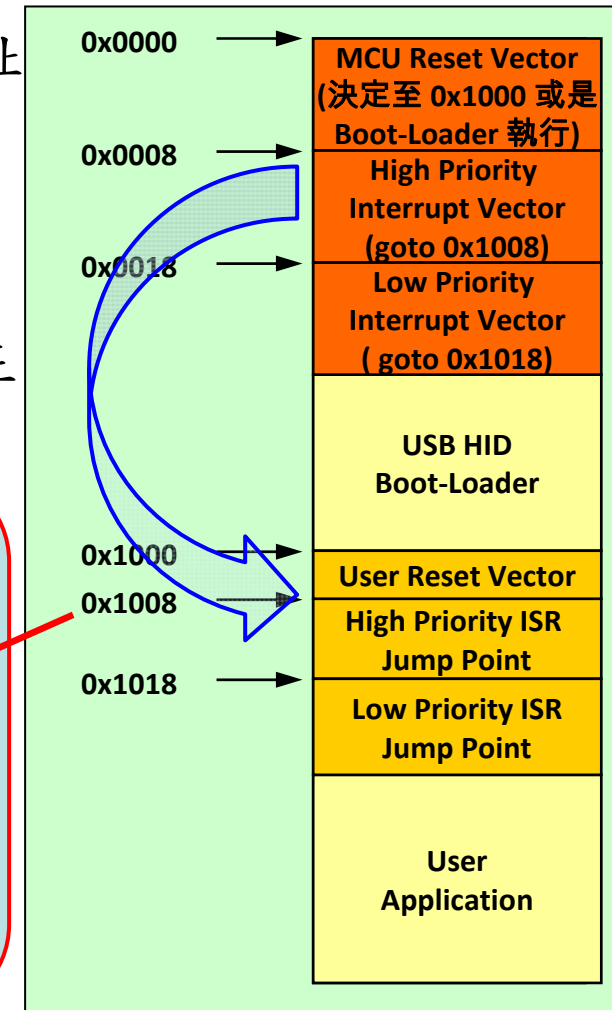


PIC18F User Program

對位址 0x1008 的處置方式

- 若 CPU 回應高優先全中斷請求時會跳至位址 0x0008 嘗試執行中斷服務程式
- Boot-Loader 會使用 “goto 0x1008” 指令跳至 0x1008 執行
- User Program 必須在 0x1008 將程式交由真正的 ISR -> HighISR()

```
#pragma code  
  
REMAPPED_HIGH_INTERRUPT_VECTOR =0x1008  
  
    void Remapped_High_ISR (void)  
    {  
        _asm goto HighISR _endasm  
    }  
  
#pragma code
```

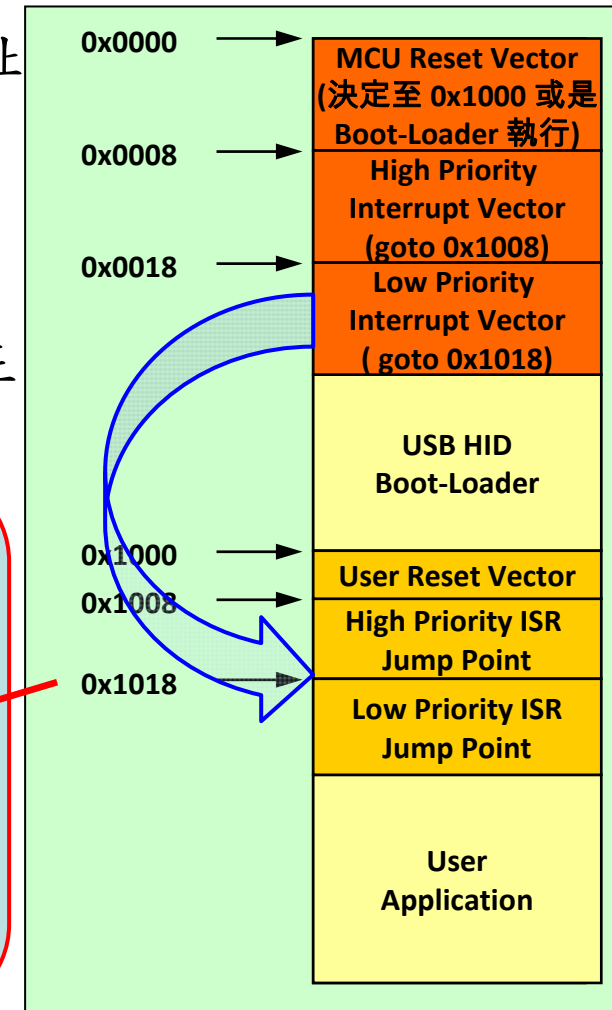


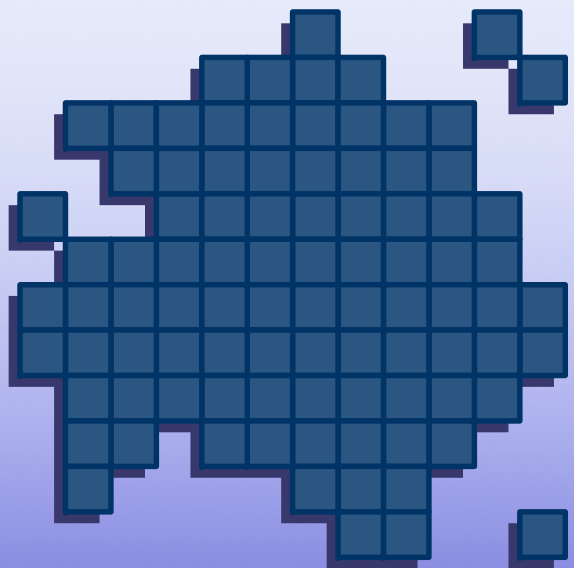
PIC18F User Program

對位址 0x1018 的處置方式

- 若 CPU 回應低優先全中斷請求時會跳至位址 0x0018 嘗試執行中斷服務程式
- Boot-Loader 會使用 “goto 0x1018” 指令跳至 0x1018 執行
- User Program 必須在 0x1008 將程式交由真正的 ISR -> LowISR()

```
#pragma code  
  
REMAPPED_LOW_INTERRUPT_VECTOR = 0x1008  
  
void Remapped_Low_ISR (void)  
{  
    _asm goto LowISR _endasm  
}  
  
#pragma code
```





適用於 USB HID Boot Loader
的定位程式範本及 .lkr 檔

定位程式範本 – BOOT_Relocate_ISR.c

```
extern void startup (void);
```

```
#pragma code REMAPPED_RESET_VECTOR = 0x1000
```

```
void _reset (void)
```

```
{  
    _asm goto _startup _endasm  
}
```

1

```
#pragma code REMAPPED_HIGH_INTERRUPT_VECTOR = 0x1008
```

```
void Remapped_High_ISR (void)
```

```
{  
    _asm goto HighISR _endasm  
}
```

2

```
#pragma code REMAPPED_LOW_INTERRUPT_VECTOR = 0x1018
```

```
void Remapped_Low_ISR (void)
```

```
{  
    _asm goto LowISR _endasm  
}
```

3

```
#pragma code
```

```
#pragma interrupt HighISR
```

```
void HighISR()
```

```
{  
}
```

4

```
#pragma code
```

```
#pragma interruptlow LowISR
```

```
void LowISR()
```

```
{  
}
```

5

```
#pragma code
```

HighISR() & LowISR 也
可以放在其他程式中 !

PIC18F User Program

定位程式範本的使用法

- 定位程式名稱為 BOOT_Relocate_ISR.c
- 定位程式的位置為：
 - TLS2118\PIC18F4550_BootLoader\
- BOOT_Relocate_ISR.c 可以為單獨的檔案加至各個 project 中
 - 最好先複製一份至各個專案所屬的目錄中
- 中斷服務程式可以直接加在 HighISR() 或是 LowISR()
- 也可將此定位程式中的 HighISR() or LowISR() 片段做註解處理，而在其他程式放置 HighISR() or LowISR()
 - Note1：處理中斷程式的宣告要一樣，名稱也必須相同
 - Note2：要修改中斷程式名稱的話要連定位程式的名稱一起改

PIC18F User Program 須使用的 .lkr 檔案

- User Program 不可以使用到 MCU 最前面的 4K Bytes program Memory
 - Boot-Loader 程式位於此區間
- User Program 需要修改過的連結器描述檔 (.lkr file)
- 修改過的連結器描述檔為：**rm18f4550 - HID Bootload.lkr**
 - 位置：TLS2118\PIC18F4550_BootLoader\
- 此連結器描述檔必須被加入 MPLAB IDE 的專案中
 - 若無此檔案則程式會與 Boot-Loader 的區間重疊
 - Boot-Loader 不會被自己破壞，但 User Program 會無法正常運作

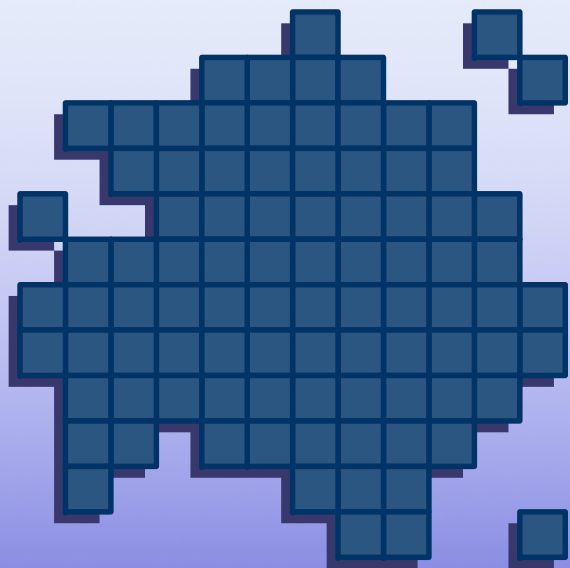
PIC18F User Program

須使用的 .lkr 檔案內容

- 0x0000 ~ 0x0FFF 的區間因為 Boot-Loader 而設為 Protected
- 0x1000 ~ 0x1029 的區間為了要安置 user Program 的起始區塊而設為 Protected (0x1000 , 0x1008 , 0x1018)
- 只有 0x102A ~ 0x7FFF 的區間可以自由放置程式

FILES c018i.o
FILES clib.lib
FILES p18f4550.lib

CODEPAGE	NAME=bootloader	START=0x0	END=0xFFF	PROTECTED
CODEPAGE	NAME=vectors	START=0x1000	END=0x1029	PROTECTED
CODEPAGE	NAME=page	START=0x102A	END=0x7FFF	
CODEPAGE	NAME=idlocs	START=0x200000	END=0x200007	PROTECTED
CODEPAGE	NAME=config	START=0x300000	END=0x30000D	PROTECTED
CODEPAGE	NAME=devid	START=0x3FFFFE	END=0x3FFFFF	PROTECTED
CODEPAGE	NAME=eedata	START=0xF00000	END=0xF000FF	PROTECTED

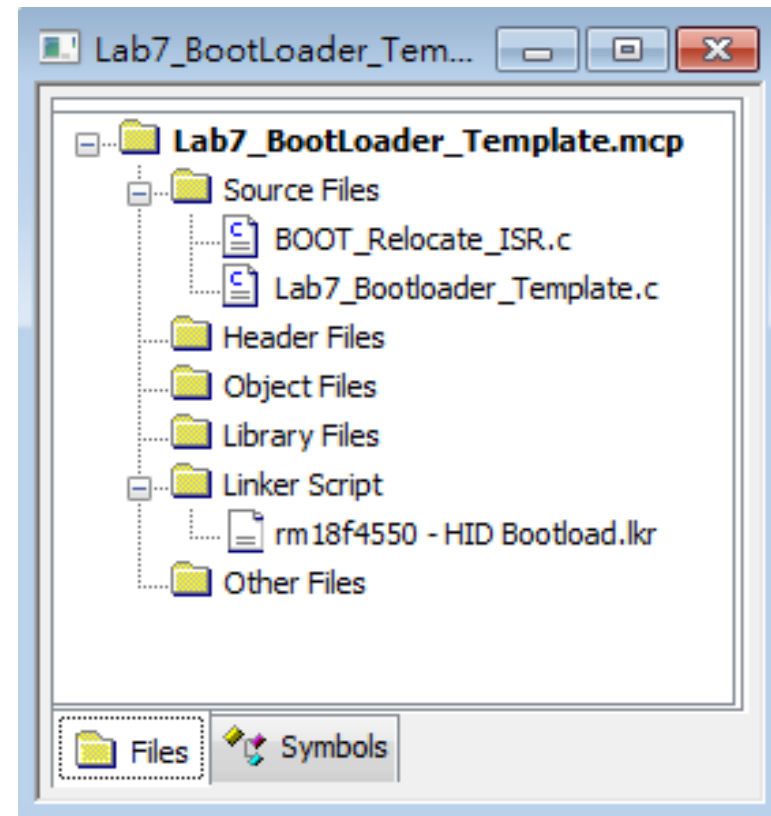


Lab7

觀察 PIC18F User Program 範本程式
並以 PC GUI 進行燒錄作業

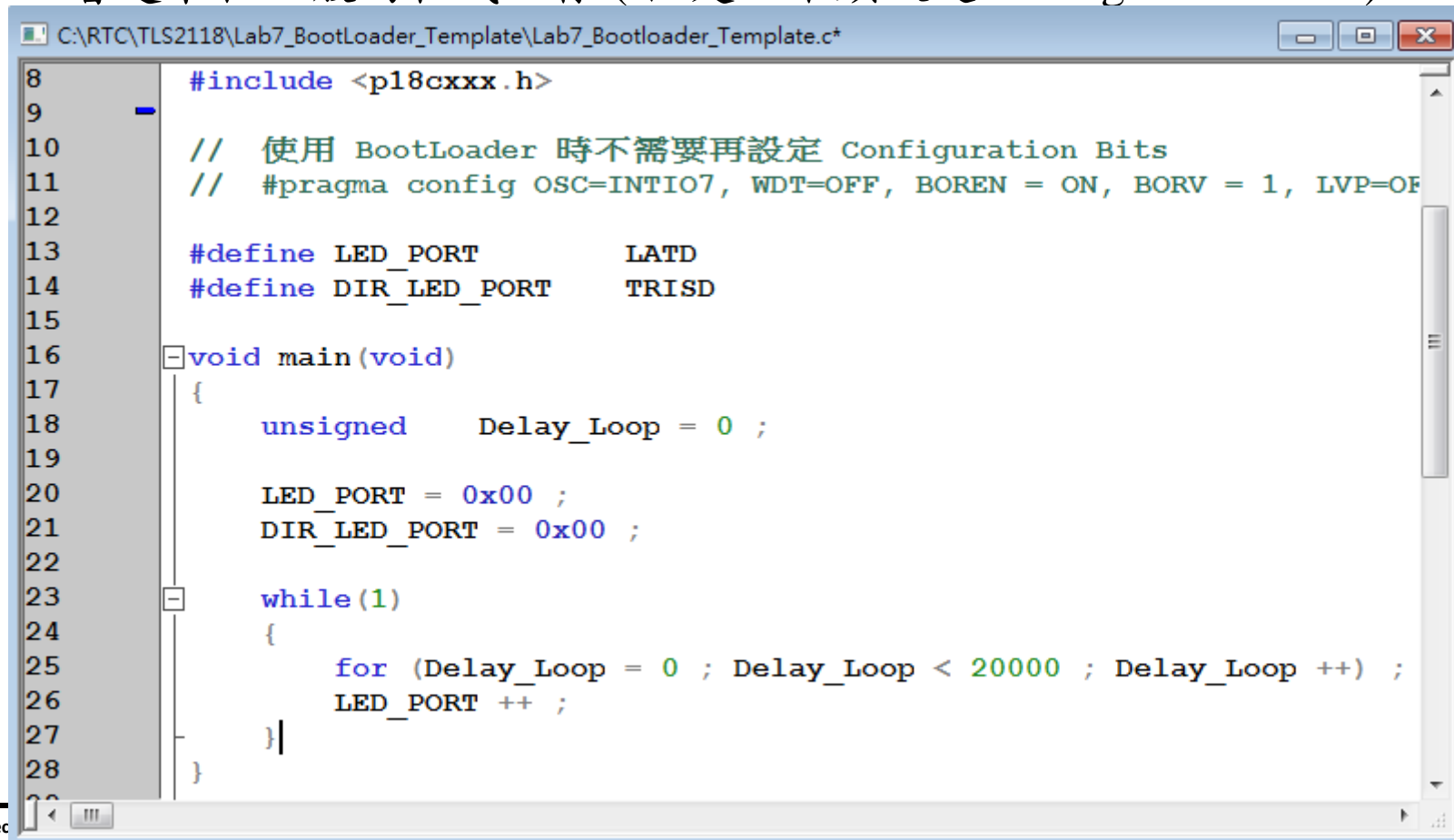
Lab7 觀察重點(1)

- 直接使用 Project -> Open 打開 Lab7
 - C:\RTC\TLS2118\Lab7_BootLoader_Template\Lab7_BootLoader_Template.mcp
- Lab7 專案中必須包含的檔案
 - Source Code
 - Lab7_Bootloader_Template.c
 - 負責調用啟動模組及定位中斷服務程式的程式段落
 - BOOT_Relocate_ISR.c
 - 修改過的連結器描述檔
 - Rm18f4550 – HID Bootloader.lkr



Lab7 觀察重點(2)

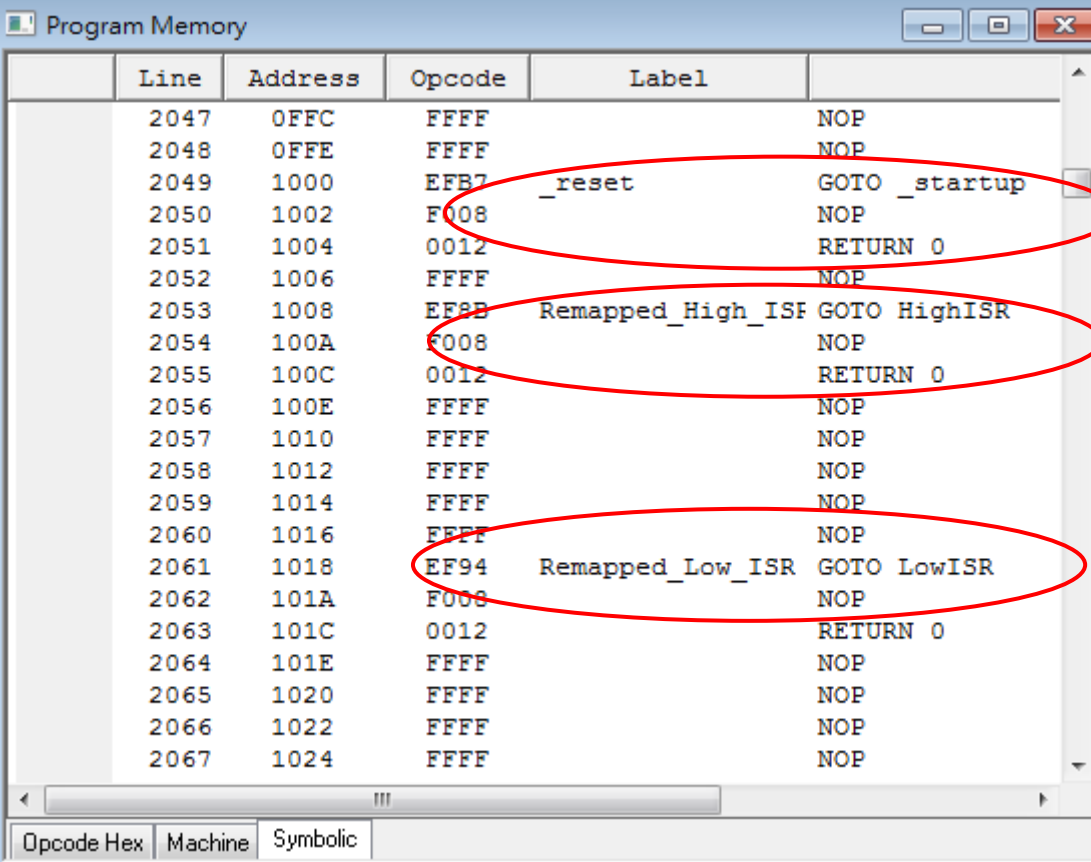
- Lab7_Bootloader_Template.c 的內容
 - 將負責調用啟動模組及定位中斷服務程式的程式段落獨立後，主程式看起來和一般的程式一樣（但是..不須設定 Configuration Bits）



```
8      #include <p18cxxx.h>
9
10     // 使用 BootLoader 時不需要再設定 Configuration Bits
11     // #pragma config OSC=INTIO7, WDT=OFF, BOREN = ON, BORV = 1, LVP=OF
12
13     #define LED_PORT      LATD
14     #define DIR_LED_PORT  TRISD
15
16     void main(void)
17     {
18         unsigned    Delay_Loop = 0 ;
19
20         LED_PORT = 0x00 ;
21         DIR_LED_PORT = 0x00 ;
22
23         while(1)
24         {
25             for (Delay_Loop = 0 ; Delay_Loop < 20000 ; Delay_Loop ++ ) ;
26             LED_PORT ++ ;
27         }
28     }
```

Lab7 觀察重點(3)

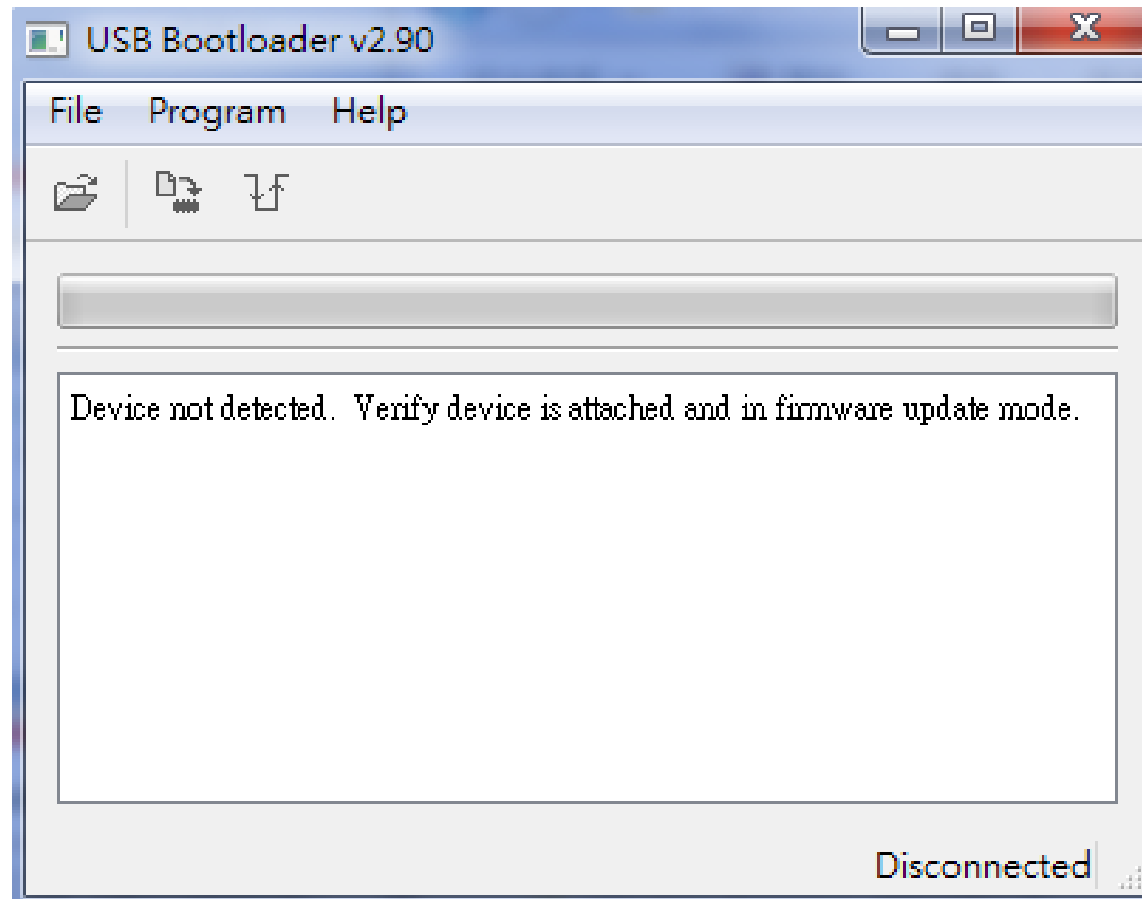
- 使用 View -> Program Memory 觀察 Program Memory 的內容
 - 程式碼的安排由位址 0x1000 開始
 - 0x1008 & 0x1018 被安插 GOTO 至中斷服務程式的程式碼



	Line	Address	Opcode	Label	
	2047	0FFC	FFFF		NOP
	2048	0FFE	FFFF		NOP
	2049	1000	EFB7	_reset	GOTO _startup
	2050	1002	F008		NOP
	2051	1004	0012		RETURN 0
	2052	1006	FFFF		NOP
	2053	1008	EF9B	Remapped_High_ISR	GOTO HighISR
	2054	100A	F008		NOP
	2055	100C	0012		RETURN 0
	2056	100E	FFFF		NOP
	2057	1010	FFFF		NOP
	2058	1012	FFFF		NOP
	2059	1014	FFFF		NOP
	2060	1016	FFFF		NOP
	2061	1018	EF94	Remapped_Low_ISR	GOTO LowISR
	2062	101A	F008		NOP
	2063	101C	0012		RETURN 0
	2064	101E	FFFF		NOP
	2065	1020	FFFF		NOP
	2066	1022	FFFF		NOP
	2067	1024	FFFF		NOP

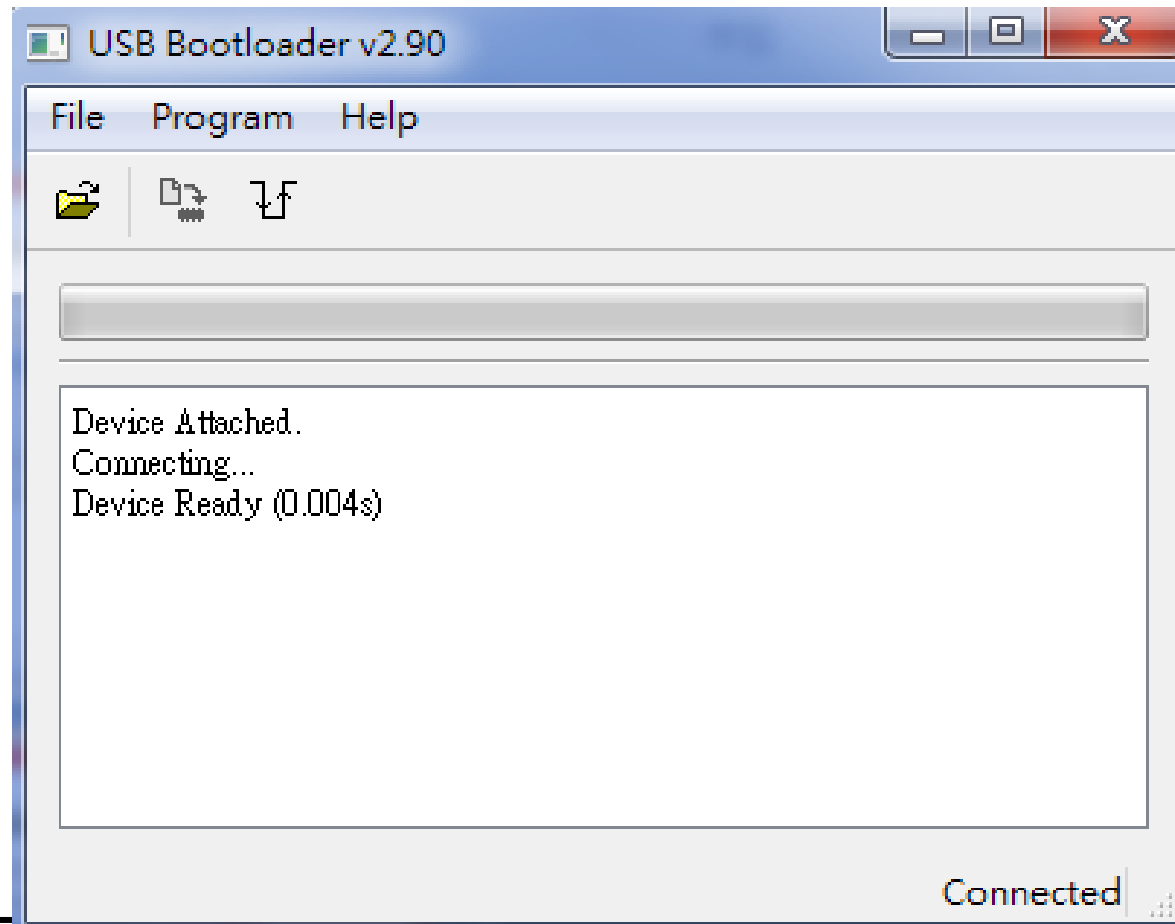
HIDBootloader.exe 的執行(1)

- 程式位置：C:\RTC\TLS2118\GUI_HIDBootLoader
 - 如果未偵測到 Device，輸出視窗會顯示 Device not detected ..



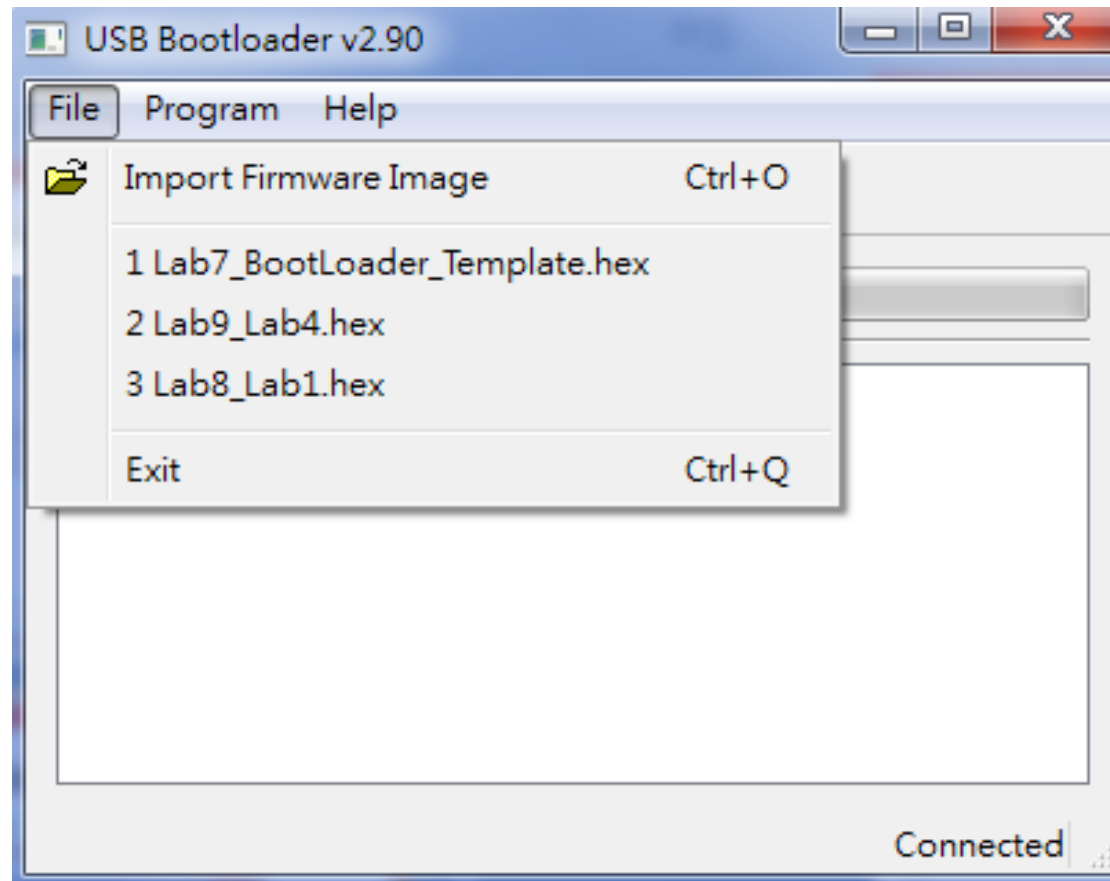
HIDBootloader.exe 的執行(2)

- 同時按下 SW1 & SW3 後，先放掉 SW1 再放掉 SW3
 - HIDBootloader 將偵測到 HID Device 並顯示 “Device Ready”



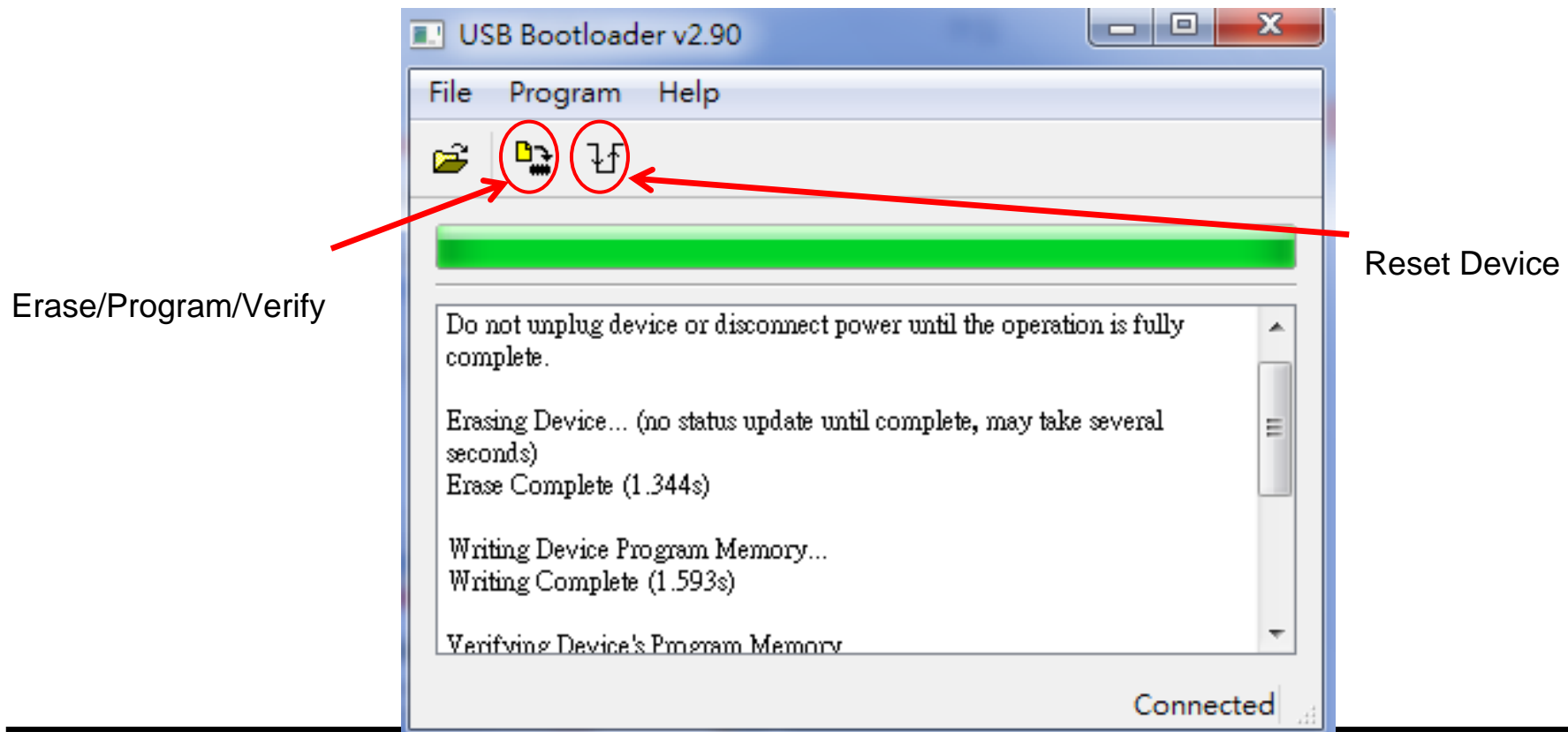
HIDBootloader.exe 的執行(3)

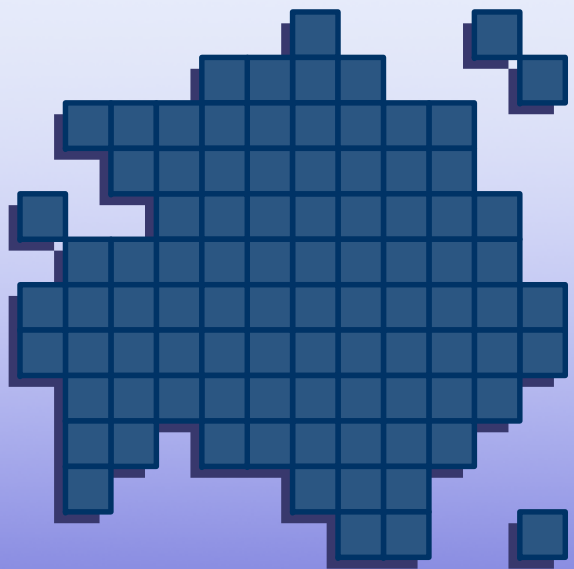
- 使用 File->Import Firmware Image 載入 Lab7 的 .hex 檔
 - Lab7_BootLoader_Template.hex



HIDBootloader.exe 的執行(4)

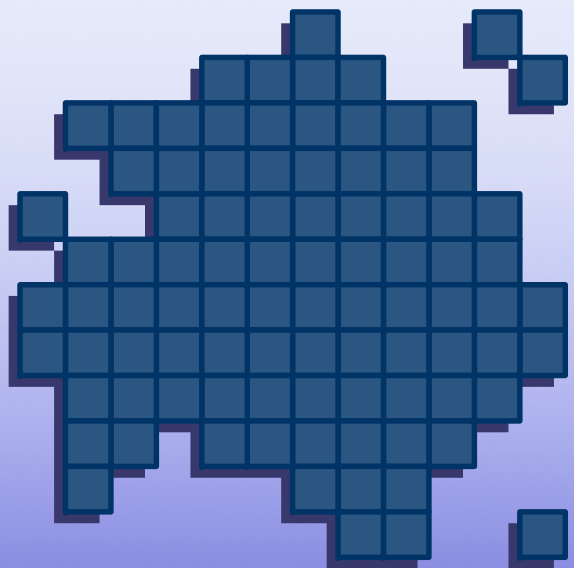
- 使用 Erase/Program/Verify 功能鈕來燒錄程式至 MCU
- 使用 Reset Device 功能鈕來 Reset MCU
- 因為 RESET 時沒有按下 SW3 所以執行 User Program





Lab8

修改 Lab1 的程式並製作一個能使用 USB HID BootLoader 載入的程式
(功能要和 Lab1 相同)



Lab9

修改 Lab4 的程式並製作一個能使用 USB HID BootLoader 載入的程式
(功能要和 Lab4 相同)