Marlin Overview

3D Printer 韌體原始碼閱讀心得 (I)

Marlin

是什麼東西

?

電腦, 電腦請問 Marlin 是什麼 ？

Marlin

可能是...

Marlin

- 馬林魚又稱旗魚，是多種海中大型掠食性魚類的總稱
  - 大型的種類可超過4公尺長。吻部細長。分佈於所有主要大洋。以其他魚類為食。旗魚是受歡迎的食物和遊釣魚種

Marlin到底是什麼東西?!

Marlin

可能是...

Marlin

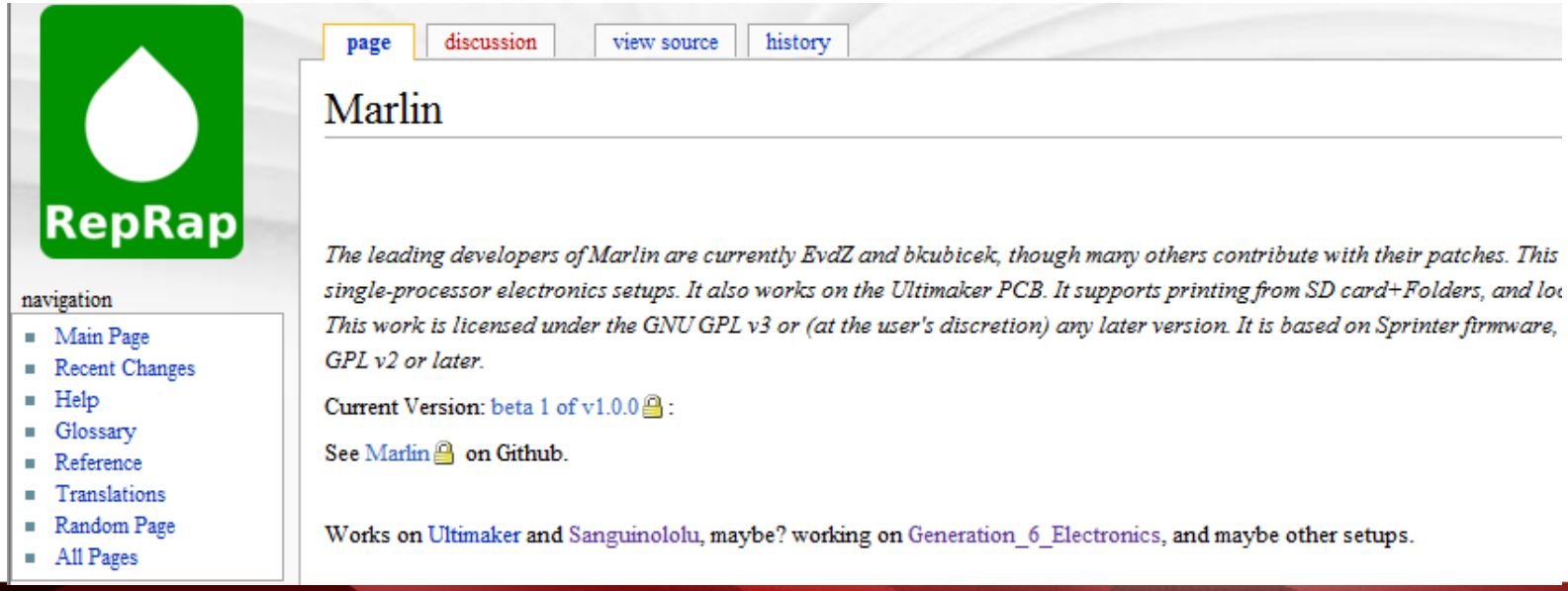
- 邁阿密馬林魚（**Miami Marlins**）是一支位於佛羅里達州邁阿密的美國職棒大聯盟球隊
- 又名美國兄弟象隊 **(BJ4)**

各位觀眾

Marlin 是...

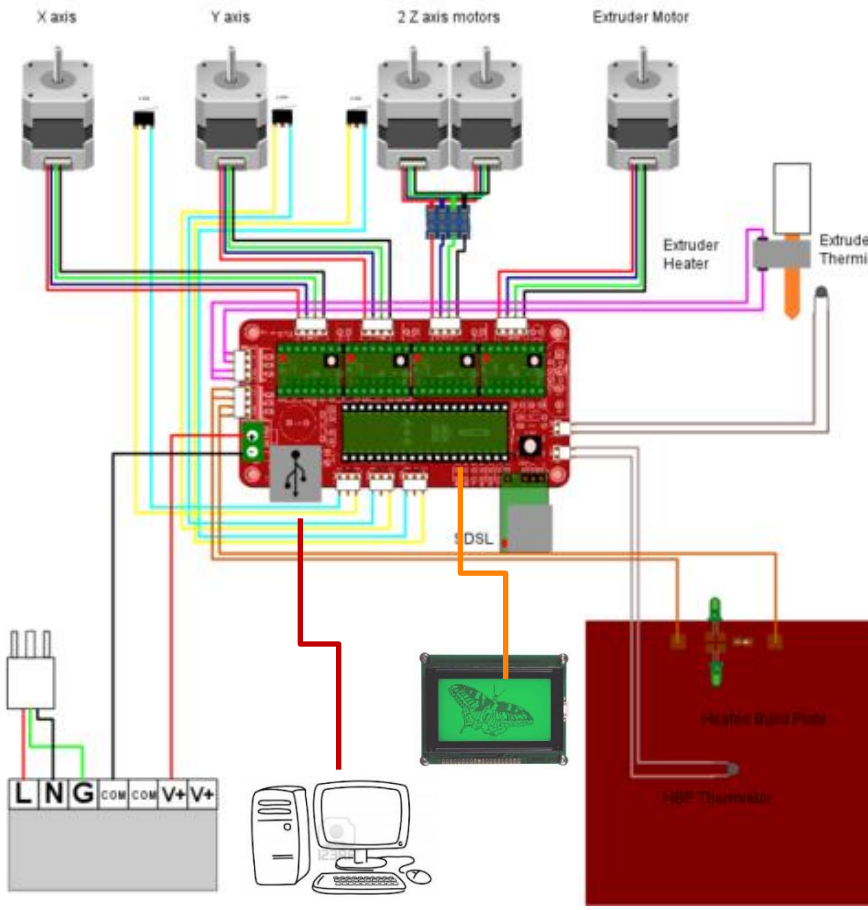
Marlin 是什麼？

- **Marlin 是一套 3D printer 的韌體**
  - 結合 **sprinter** 與 **grbl** 兩套韌體的功能
  - 是 **3D printer** 主流的韌體之一
  - 支援大多數 **3D printer** 的硬體平台



Marlin 是什麼？

- **Marlin 在 3D printer 扮演的角色**
  - 驅動控制板
  - 讀取 g code 檔執行列印的工作
  - 控制步進馬達列印出實體
  - 控制擠出頭及加熱板的溫度
  - 偵測擠出頭及加熱板的溫度
    作為控制溫度的回饋
  - 有讀寫 SD card 的功能
  - 支援 LCD 顯示列印的訊息



Marlin 是什麼？

- **Marlin 的 特色**
  - 在中斷中，處理步進馬達的梯形加減速
  - 在中斷中，處理擠出頭及加熱板的溫度控制
  - 有軌跡規劃預覽的功能 **(look ahead)**
  - 支援圓弧軌跡的規劃
  - 利用 **PID** 控制加熱的溫度
  - 溫度的偵測精度較高、誤差較少
  - **Endstop trigger** 時，有中止運作的功能
  - 整個 **Marlin** 的執行檔約 **50 KB**
  - 族繁不及備載…

講了這麼多是不是

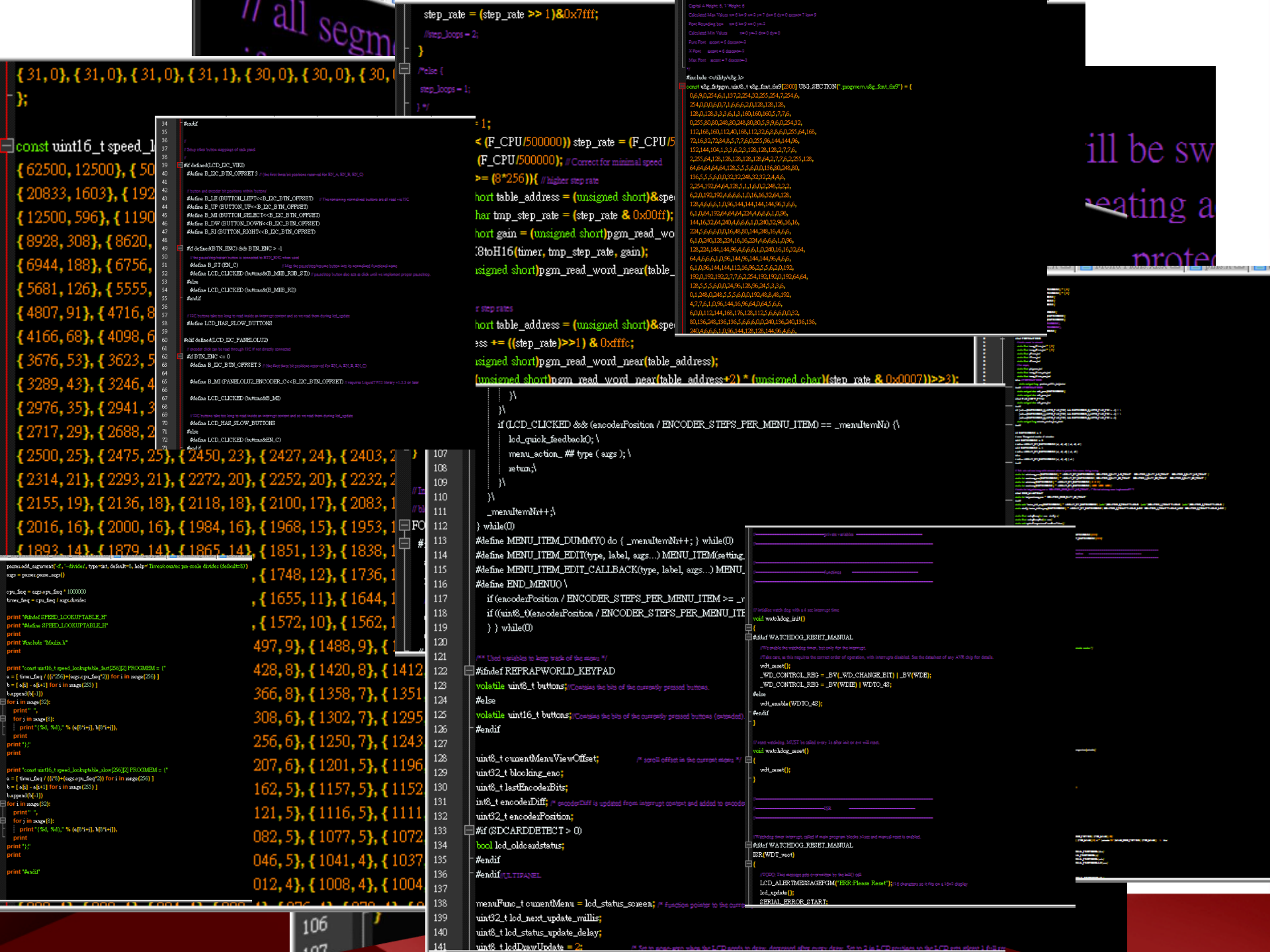
迫不及待

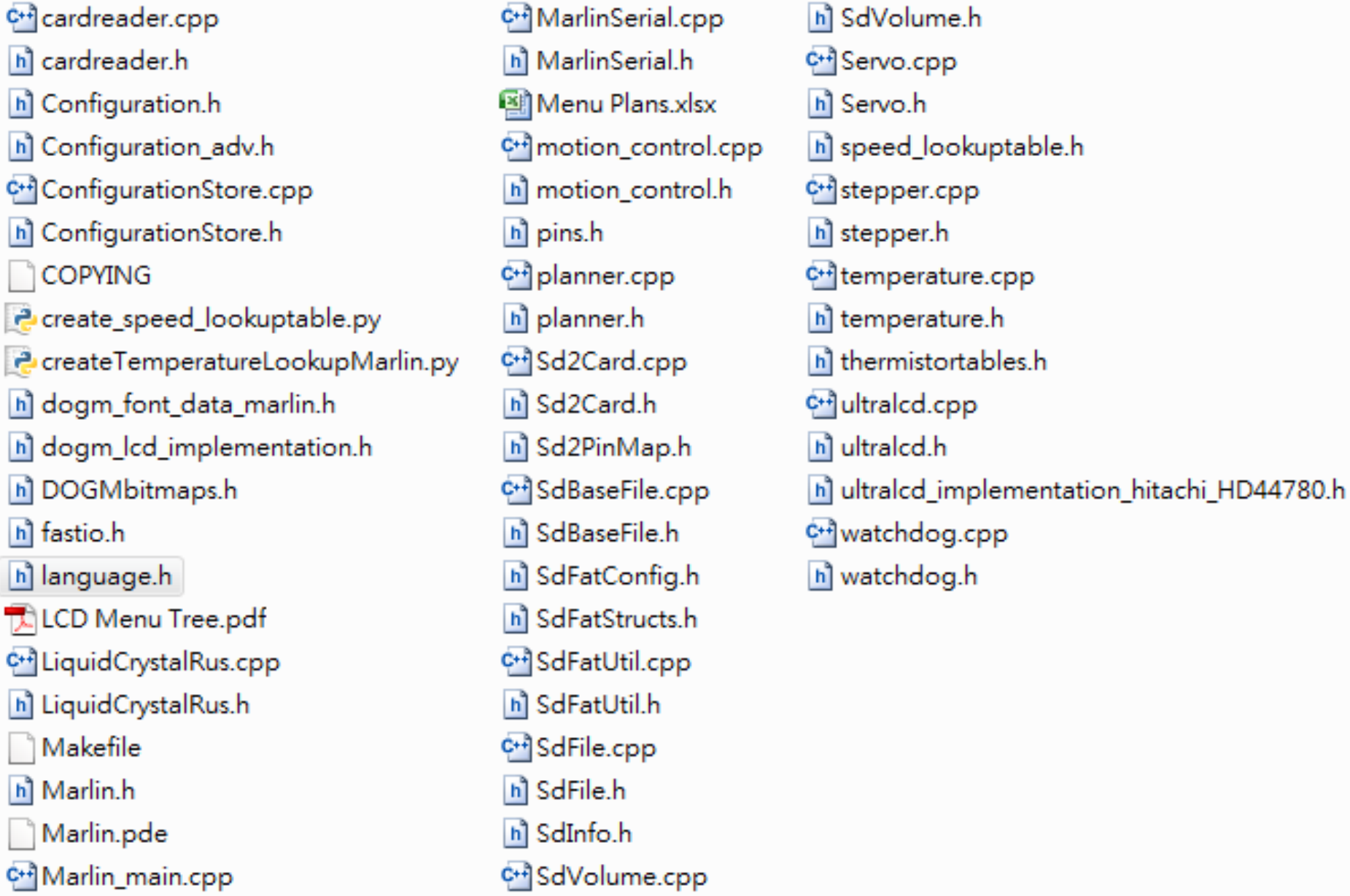
想看

Marlin 的

真相

了呢!




cardreader.cpp
cardreader.h
Configuration.h
Configuration_adv.h
ConfigurationStore.cpp
ConfigurationStore.h
COPYING
create_speed_lookuptable.py
createTemperatureLookupMarlin.py
dogm_font_data_marlin.h
dogm_lcd_implementation.h
DOGMbitmaps.h
fastio.h
language.h
LCD Menu Tree.pdf
LiquidCrystalRus.cpp
LiquidCrystalRus.h
Makefile
Marlin.h
Marlin.pde
Marlin_main.cpp
MarlinSerial.cpp
MarlinSerial.h
Menu Plans.xlsx
motion_control.cpp
motion_control.h
pins.h
planner.cpp
planner.h
Sd2Card.cpp
Sd2Card.h
Sd2PinMap.h
SdBaseFile.cpp
SdBaseFile.h
SdFatConfig.h
SdFatStructs.h
SdFatUtil.cpp
SdFatUtil.h
SdFile.cpp
SdFile.h
SdInfo.h
SdVolume.cpp
SdVolume.h
Servo.cpp
Servo.h
speed_lookuptable.h
stepper.cpp
stepper.h
temperature.cpp
temperature.h
thermistortables.h
ultralcd.cpp
ultralcd.h
ultralcd_implementation_hitachi_HD44780.h
watchdog.cpp
watchdog.h

真相

往往跟

正妹的素顏一樣

看過都會有

不堪回首

的傷痛

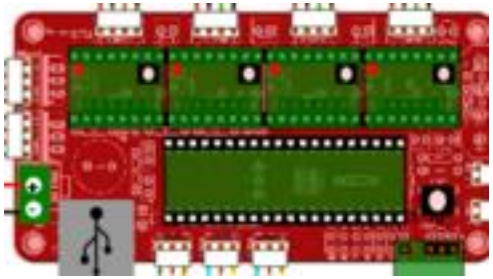
補妝中

請稍候 . . .

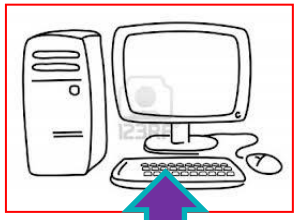
Marlin 燒錄的架構

Configuration lib

安裝組態的規畫



Marlin 運作時的架構



訊息接收端

Serial lib

SD lib

訊息顯示端

主體端

main lib

LCD lib

plan motion lib

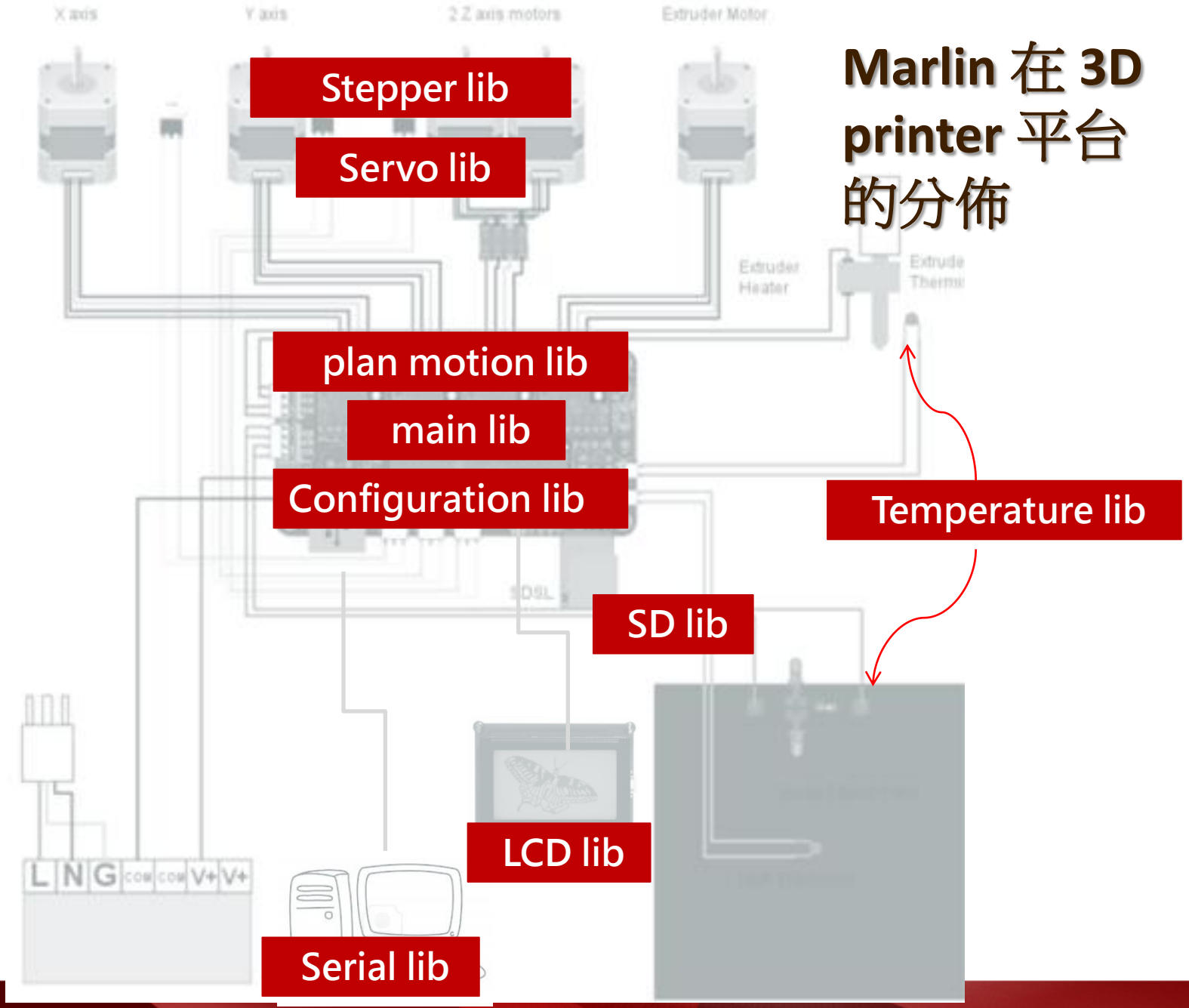
驅動端

Stepper lib

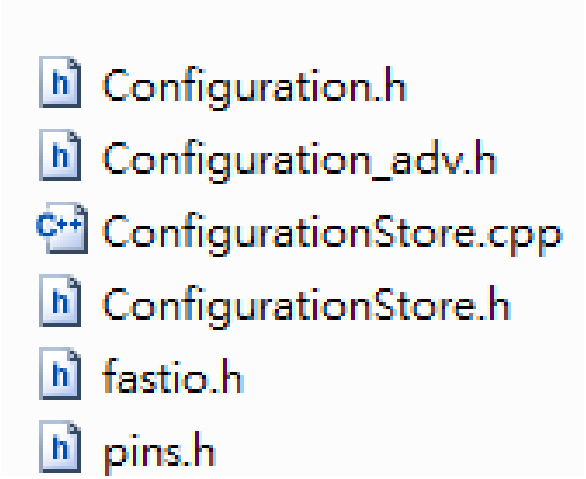
Servo lib

Temperature lib

Marlin 在 3D printer 平台的分佈



Configuration lib



Configuration lib

- **Marlin 支援的控制板**

Gen7 Alfons3 RAMPS Duemilanove

Sanguinololu Gen6 Ultimaker

RUMBA Teensylu 0.7 Gen3

Alpha OMCA Rambo MegaTronics

Configuration lib

- **user 使用 Marlin 時，首先要注意**
  - 用的控制板是否 **Marlin** 有支援
  - 在 Configuration.h 裡，要依控制板的型號，自行設定對應的 MOTHERBOARD 的參數
  - 同時會檢查控制板 **ATMega** 的型號是否符合
  - Marlin 預設 MOTHERBOARD 的參數是 99

```
#ifndef MOTHERBOARD
#define MOTHERBOARD 99
#endif
```

Configuration lib

- MOTHERBOARD 的參數列表

```
10 = Gen7 custom (Alfons3 Version) "https://github.com/Alfons3/G
11 = Gen7 v1.1, v1.2 = 11
12 = Gen7 v1.3
13 = Gen7 v1.4
3  = MEGA/RAMPS up to 1.2 = 3
33 = RAMPS 1.3 / 1.4 (Power outputs: Extruder, Bed, Fan)
34 = RAMPS 1.3 / 1.4 (Power outputs: Extruder0, Extruder1, Bed)
4  = Duemilanove w/ ATMega328P pin assignment
5  = Gen6
51 = Gen6 deluxe
6  = Sanguinololu < 1.2
62 = Sanguinololu 1.2 and above
```

Configuration lib

- MOTHERBOARD 的參數列表(續)

```
63 = Melzi
64 = STB V1.1
7  = Ultimaker
71 = Ultimaker (Older electronics. Pre 1.5.4. This is rare)
8  = Teensylu
80 = Rumba
81 = Printrboard (AT90USB1286)
82 = Brainwave (AT90USB646)
9  = Gen3+
70 = Megatronics
701= Megatronics v2.0
702= Minitronics v1.0
90 = Alpha OMCA board
91 = Final OMCA board
301 = Rambo
```

Configuration lib

- 在 pins.h 裡， marlin 會依 MOTHERBOARD 的參數 mapping 相對應的 **pin** 腳

```
//x axis pins
#define X_STEP_PIN 19
#define X_DIR_PIN 18
#define X_ENABLE_PIN 24
#define X_STOP_PIN 7

//y axis pins
#define Y_STEP_PIN 23
#define Y_DIR_PIN 22
#define Y_ENABLE_PIN 24
#define Y_STOP_PIN 5
```

Configuration lib

- 在 fastio.h 裡，會依對應的 **pin** 腳，設定 **ATMega** 暫存器做讀寫的動作

```
#define   DIO19_PIN      PIND2
#define   DIO19_RPORT PIND
#define   DIO19_WPORT    PORTD
#define   DIO19_DDR     DDRD
#define DIO19_PWM       NULL

#define   DIO20_PIN      PIND1
#define   DIO20_RPORT PIND
#define   DIO20_WPORT    PORTD
#define   DIO20_DDR     DDRD
```

Gen 系列

- **Gen7 v1.1, v1.2, v1.3 Electronics**
  - 採用的晶片型號 **ATMega644P, ATMega644, ATMega1284p**
  - 設定 **MOTHERBOARD** 為 11, 12

```
/**************************************************************************************************
* Gen7 v1.1, v1.2, v1.3 pin assignment
*
**************************************************************************************************/


#if MOTHERBOARD == 12
#define MOTHERBOARD 11
#define GEN7_VERSION 13 // v1.3
#endif

#if MOTHERBOARD == 11
#define KNOWN_BOARD

#if !defined(__AVR_ATmega644P__) && !defined(__AVR_ATmega644__) && !defined(__AVR_ATmega1284P__)
#error Oops! Make sure you have 'Gen7' selected from the 'Tools -> Boards' menu.

#endif
```

Gen 系列

- **Gen7 v1.4 Electronics**
  - 採用的蕊片型號 **ATMega644P, ATMega644, ATMega1284p**
  - 設定 **MOTHERBOARD** 為 13

```
/**************************************************************************
 * Gen7 v1.4 pin assignment
 *
 **************************************************************************/

#if MOTHERBOARD == 13
#define GEN7_VERSION 14 // v1.4
#endif

#if MOTHERBOARD == 13
#define KNOWN_BOARD

#if !defined(__AVR_ATmega644P__) && !defined(__AVR_ATmega644__) && !defined(__AVR_ATmega1284P__)
#error Oops! Make sure you have 'Gen7' selected from the 'Tools -> Boards' menu.

#endif
```

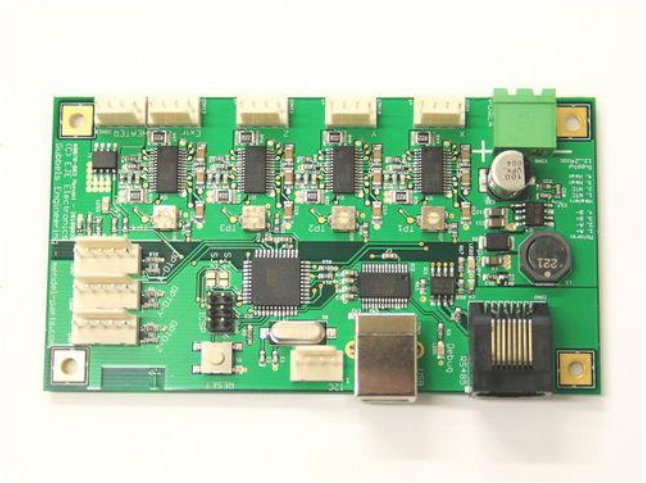
Gen 系列

- **Gen7 Alfons3 Electronics**
  - 採用的蕊片型號 **ATMega644P, ATMega644, ATMega1284p**
  - 設定 **MOTHERBOARD** 為 10

```
/*********************************************************************
*********
* Gen7 Alfons3  pin assignment
*
**************************************************************************
********/
/* These Pins are assigned for the modified GEN7 Board from Alfons

#if MOTHERBOARD == 10
#define KNOWN_BOARD

#if !defined(__AVR_ATmega644P__) && !defined(__AVR_ATmega644__) && !defined(__AVR_ATmega1284P__)
    #error Oops!  Make sure you have 'Gen7' selected from the 'Tools -> Boards' menu.

#endif
```

Gen 系列

- **Gen6 Electronics**
  - 採用的蕊片型號 **ATMega644P, ATMega1284p**
  - 設定 **MOTHERBOARD** 為 5 或是 51

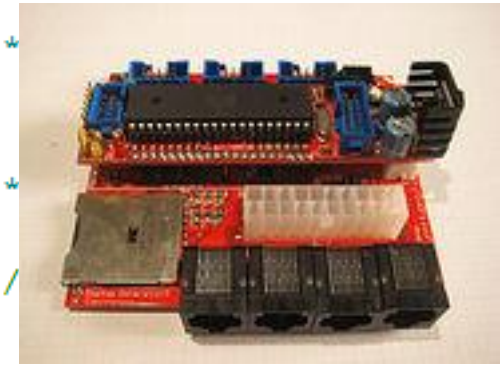


```
/**************************************************************
 * Gen6 pin assignment
 *
 **************************************************************/
#if MOTHERBOARD == 5 || MOTHERBOARD == 51
 #define KNOWN_BOARD 1

#ifndef __AVR_ATmega644P__
#ifndef __AVR_ATmega1284P__
 #error Oops!  Make sure you have 'Sanguino' selected from the 'Tools -> Boards' menu.
 #endif
 #endif
```

Gen 系列

- **Gen3 Electronics**
  - 採用的蕊片型號 **ATmega644P, ATmega1284p**
  - 設定 **MOTHERBOARD** 為 9

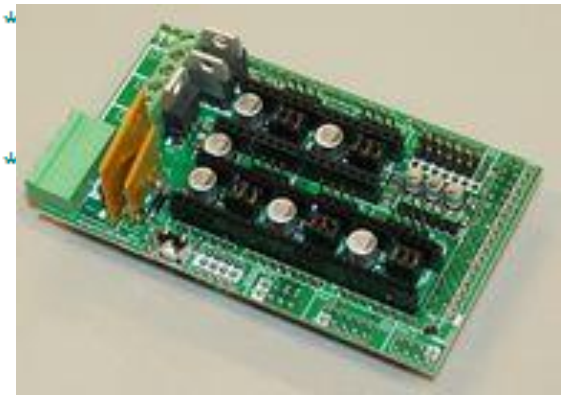
```
/**********************************************************************
 * Gen3+ pin assignment
 *
 **********************************************************************/
#if MOTHERBOARD == 9
#define MOTHERBOARD 6   /*TODO: Figure out, Why is this done?*/
#define KNOWN_BOARD 1
#ifndef __AVR_ATmega644P__
#ifndef __AVR_ATmega1284P__
#error Oops!  Make sure you have 'Sanguino' selected from the 'Tools -> Boards' menu.
#endif
#endif
```



RAMPS(MEGA)

- **RAMPS 1.2, 1.3, 1.4**
  - 採用的蕊片型號 **ATmega1280, ATmega2560**
  - 設定 **MOTHERBOARD** 為 3 (RAMPS 1.2) , 33 (RAMPS 1.3), 或是 34 (RAMPS 1.4)

```
/****************************************************************************
* Arduino Mega pin assignment
*
****************************************************************************/
#if MOTHERBOARD == 3 || MOTHERBOARD == 33 || MOTHERBOARD == 34
#define KNOWN_BOARD 1

//////////////////FIX THIS//////////////
#ifndef __AVR_ATmega1280__
 #ifndef __AVR_ATmega2560__
 #error Oops!  Make sure you have 'Arduino Mega' selected from the 'Tools -> Boards' menu.
 #endif
#endif
```



Duemilanove

- **Duemilanove w/ ATMega328p**
  - 採用的蕊片型號 **ATmega328p**
  - 設定 **MOTHERBOARD** 為 4

```
/*************************************************************
* Duemilanove w/ ATMega328P pin assignment
*
*************************************************************/
#if MOTHERBOARD == 4
#define KNOWN_BOARD 1

#ifndef __AVR_ATmega328P__
#error Oops!  Make sure you have 'Arduino Duemilanove w/ ATMega328' selected from the 'To
#endif
```

Sanguinololu 系列

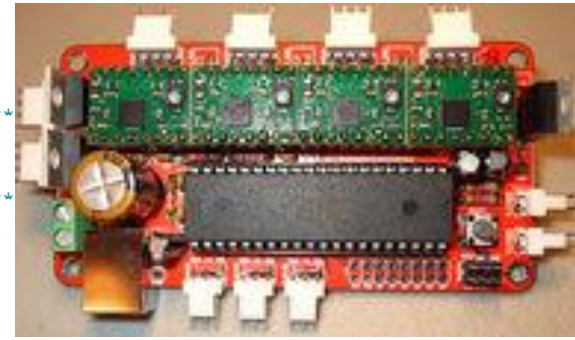
- **Sanguinololu, MELZI, STB V1.1**
  - 採用的蕊片型號 **ATmega644P, ATmega1284p**
  - 設定 **MOTHERBOARD** 為

    6 (Sanguinololu 1.2)

    62 (Sanguinololu 1.2 and above)

    63 (MELZI)

    64 (STB V1.1)



```
/*************************************************************
 * Sanguinololu pin assignment
 *
 *************************************************************/
#if MOTHERBOARD == 64
#define STB
#endif
#if MOTHERBOARD == 63
#define MELZI
#endif
#if MOTHERBOARD == 62 || MOTHERBOARD == 63 || MOTHERBOARD == 64
#undef MOTHERBOARD
#define MOTHERBOARD 6
#define SANGUINOLOLU_V_1_2
#endif
#if MOTHERBOARD == 6
#define KNOWN_BOARD 1
#ifndef __AVR_ATmega644P__
#ifndef __AVR_ATmega1284P__
#error Oops!  Make sure you have 'Sanguino' selected from the 'Tools -> Boards' menu.
#endif
#endif
```

當我拿到

新的

Sanguinololu

就可以

控制

3D printer

了嗎

？


你到底在胡说什么啊

Marlin & Sanguinololu

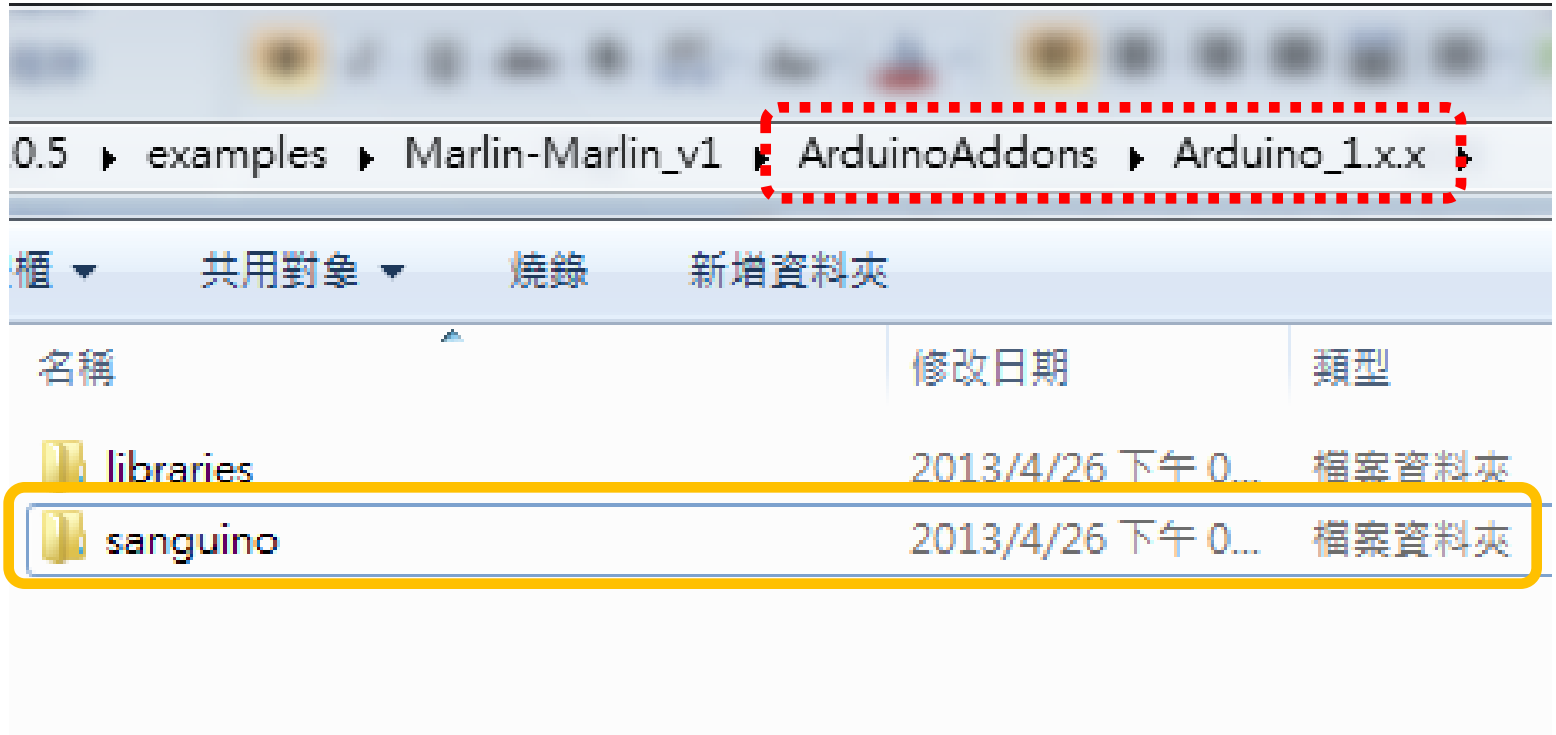
- 在用 **Sanguinololu** 之前需要安裝 **Marlin**
- 安裝 **Marlin** 需要下面幾個步驟
  1. **Arduino IDE** 要加進 **Sanguinololu** 的版本(board)
  2. **Sanguinololu** 控制板要燒錄 **Sanguino bootloader**
  3. **Sanguinololu** 利用 **Arduino IDE upload** Marlin

1. Sanguino IDE

- 從 GitHub 下載 Marlin 的 source code

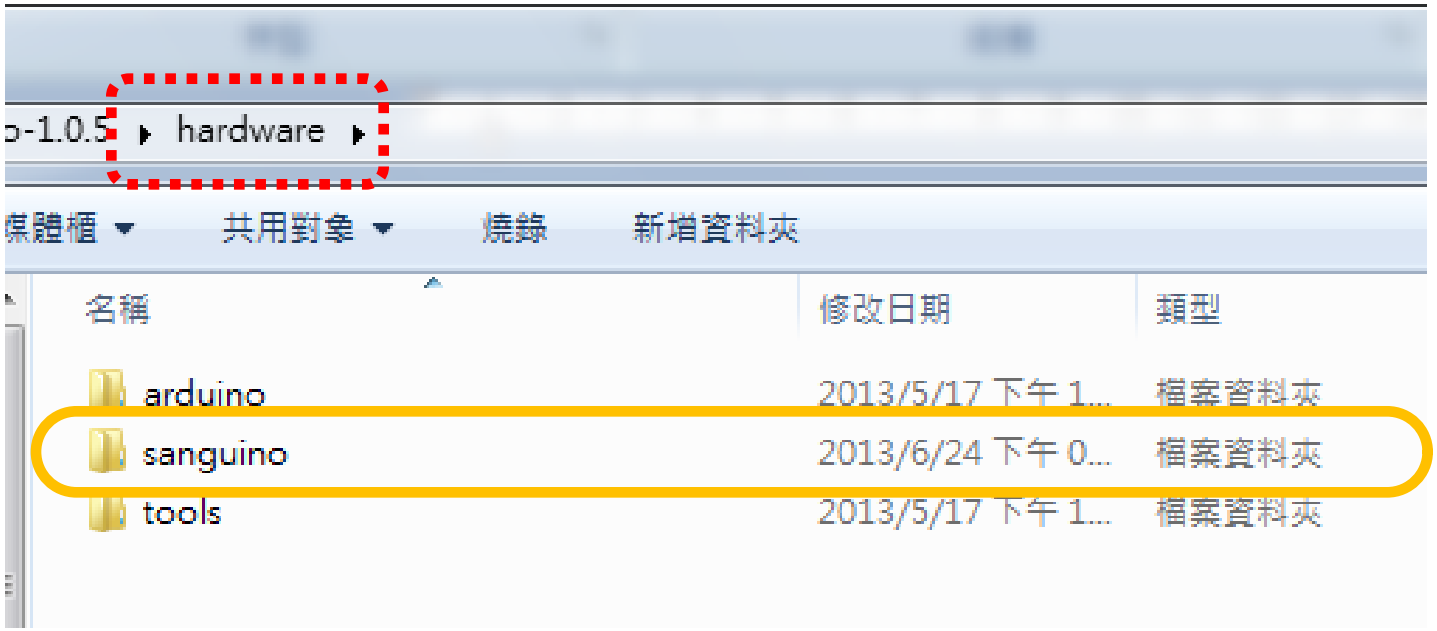
1. Sanguino IDE

- **Sanguino IDE 所在地**



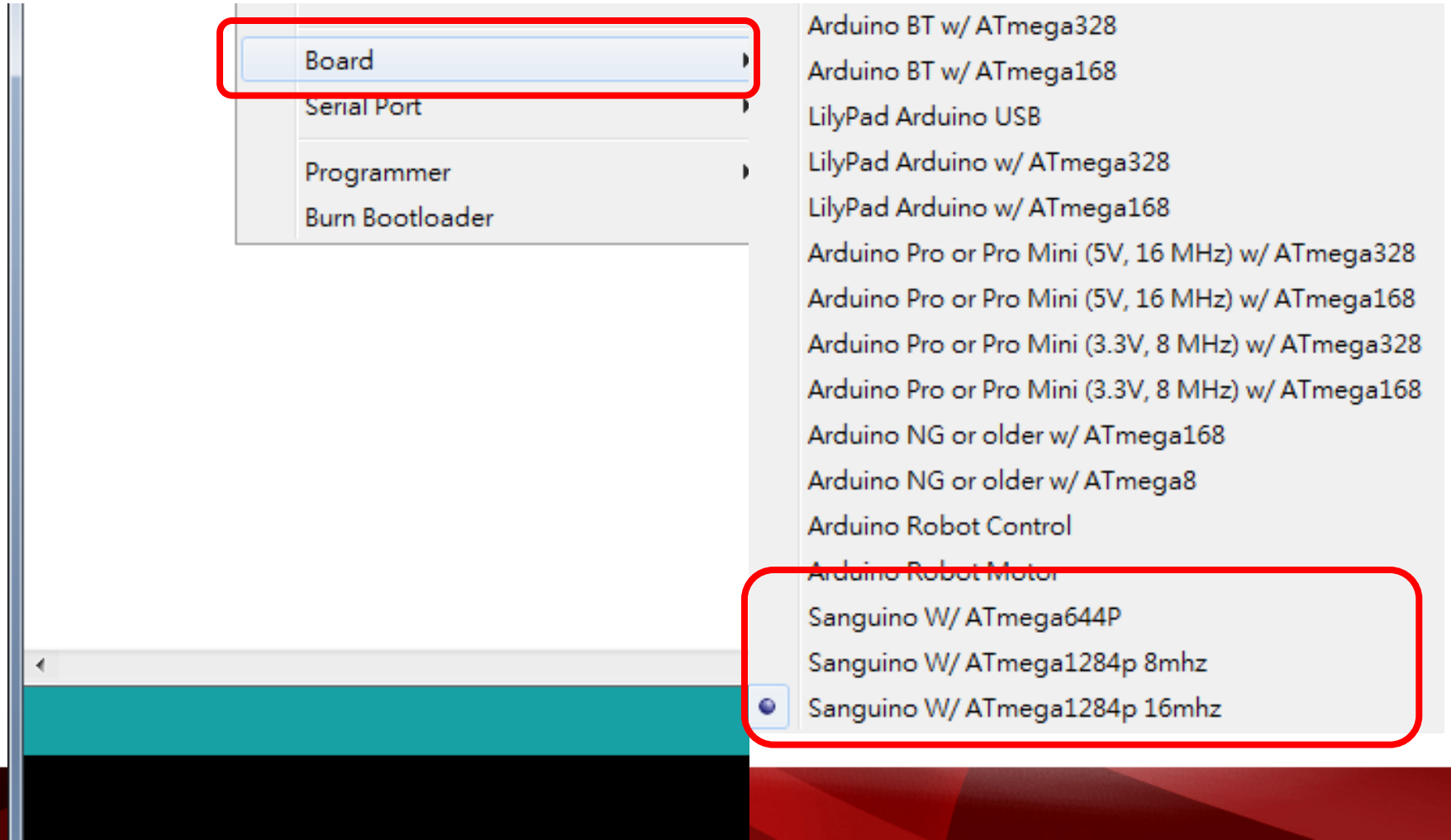
1. Sanguino IDE

- 將 Sanguino IDE 複製到…



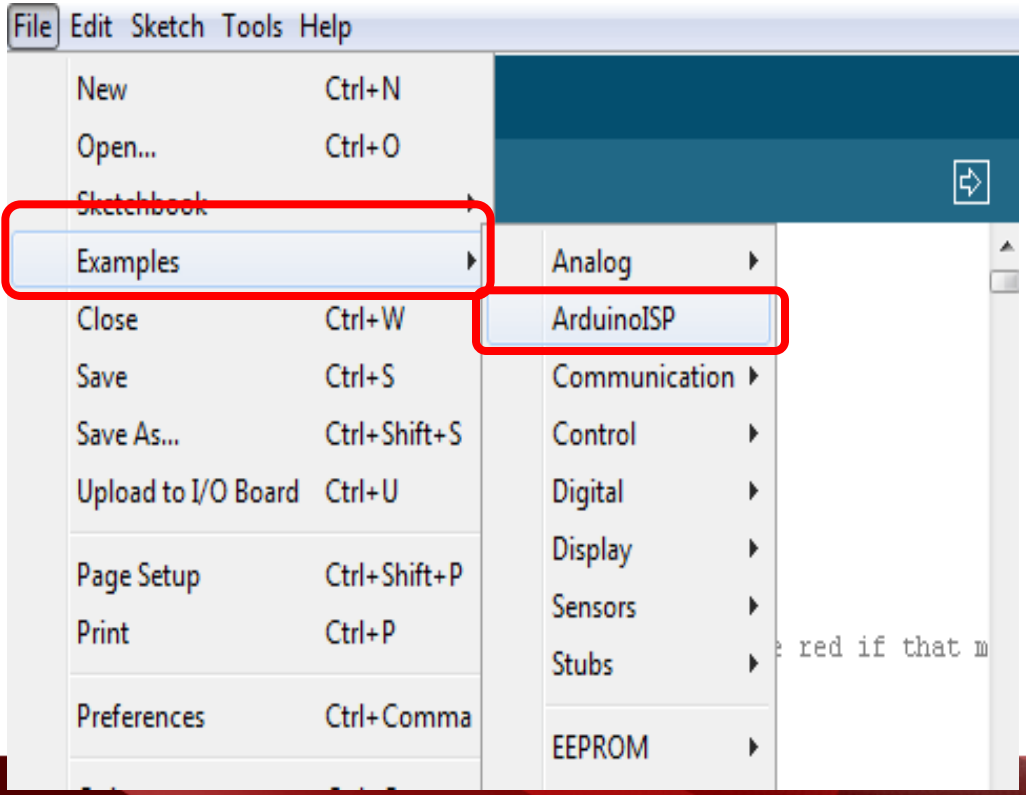
1. Sanguino IDE

- 開啟 **Arduino IDE** 的 **Board** 會看到…



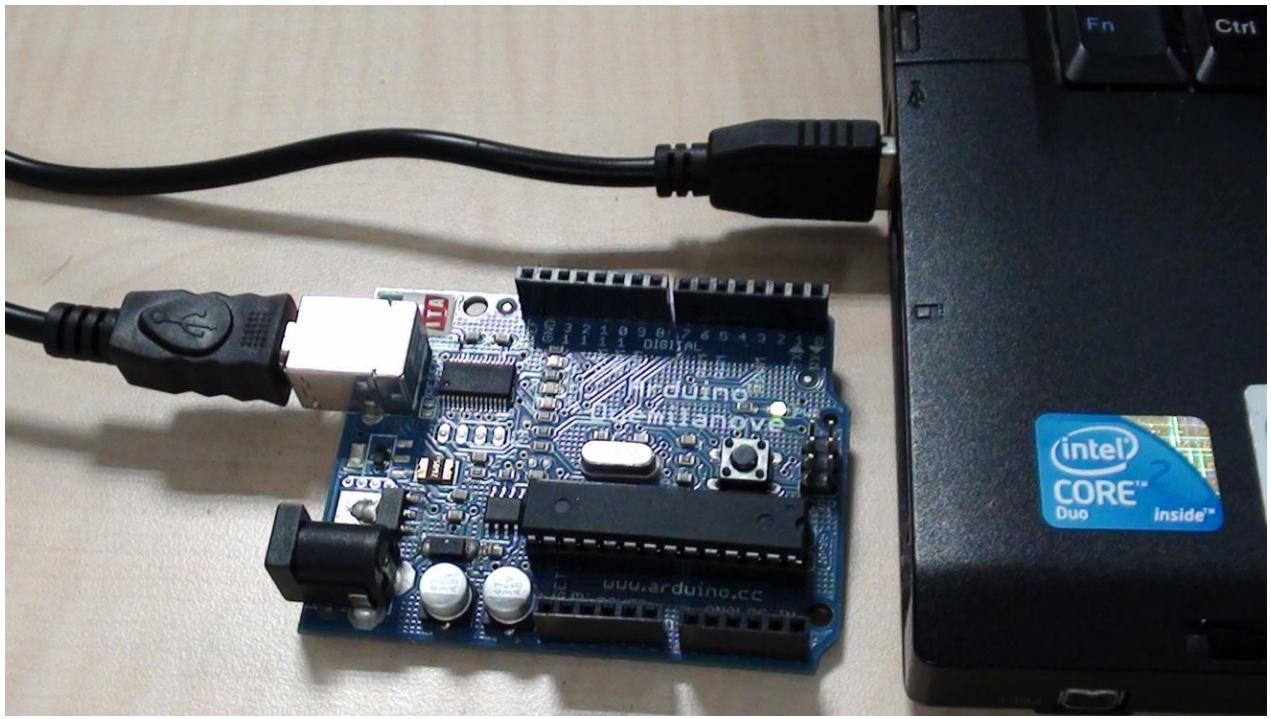
2. Sanguino bootloader

- 我們的方法是透過 **Arduino 燒錄 Sanguino bootloader**
  - 將 **Arduino 當作 ISP (In System Programmer)**
  - 使用 **Arduino IDE 的範例程式 ArduinoISP**



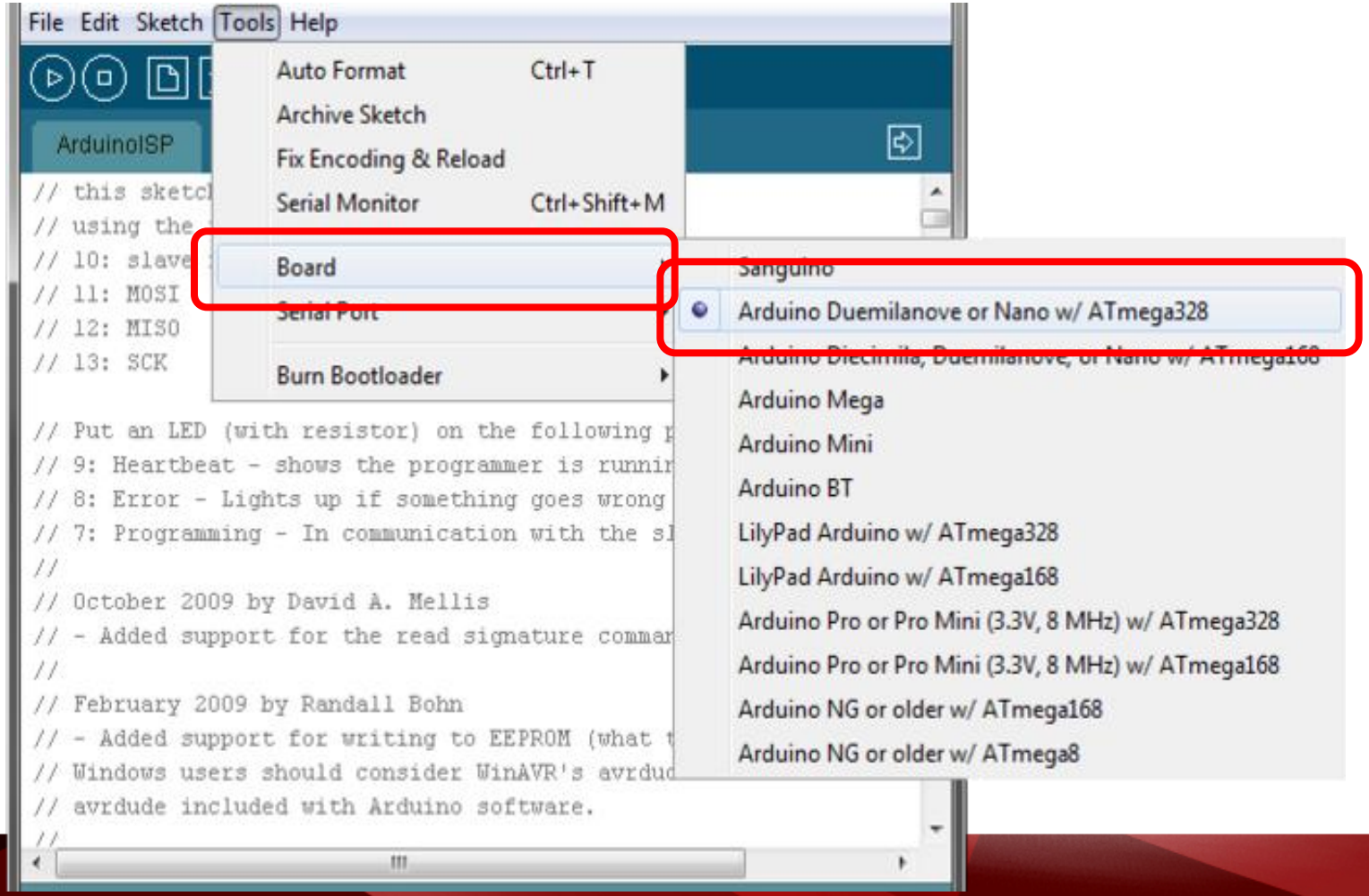
2. Sanguino bootloader

- **Arduino 與 PC 透過 USB 連線**



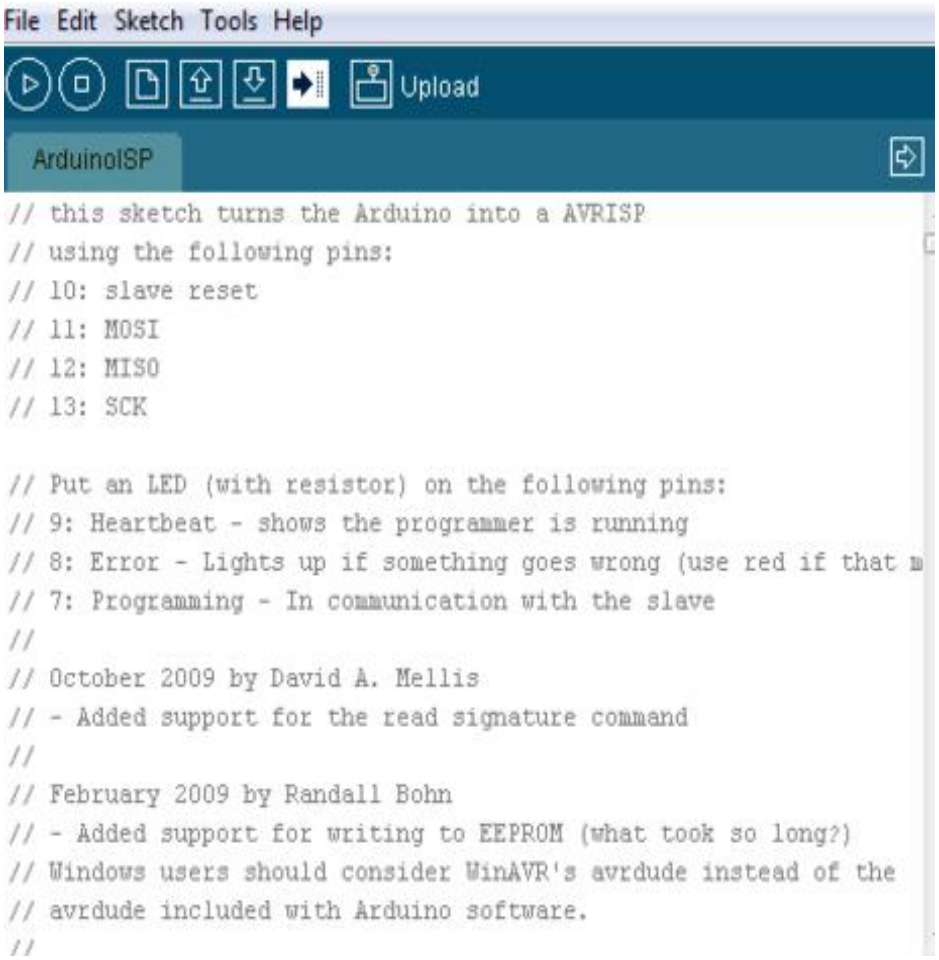
2. Sanguino bootloader

- 選擇 **Arduino** 的版本



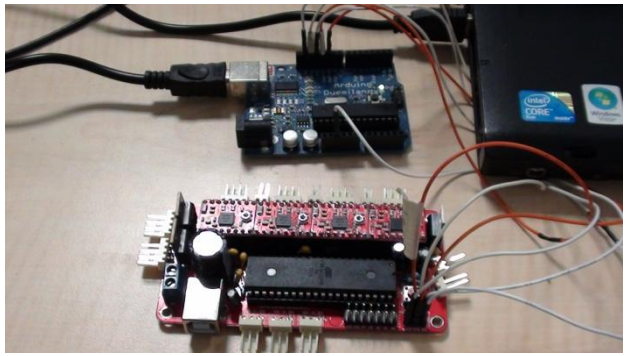
2. Sanguino bootloader

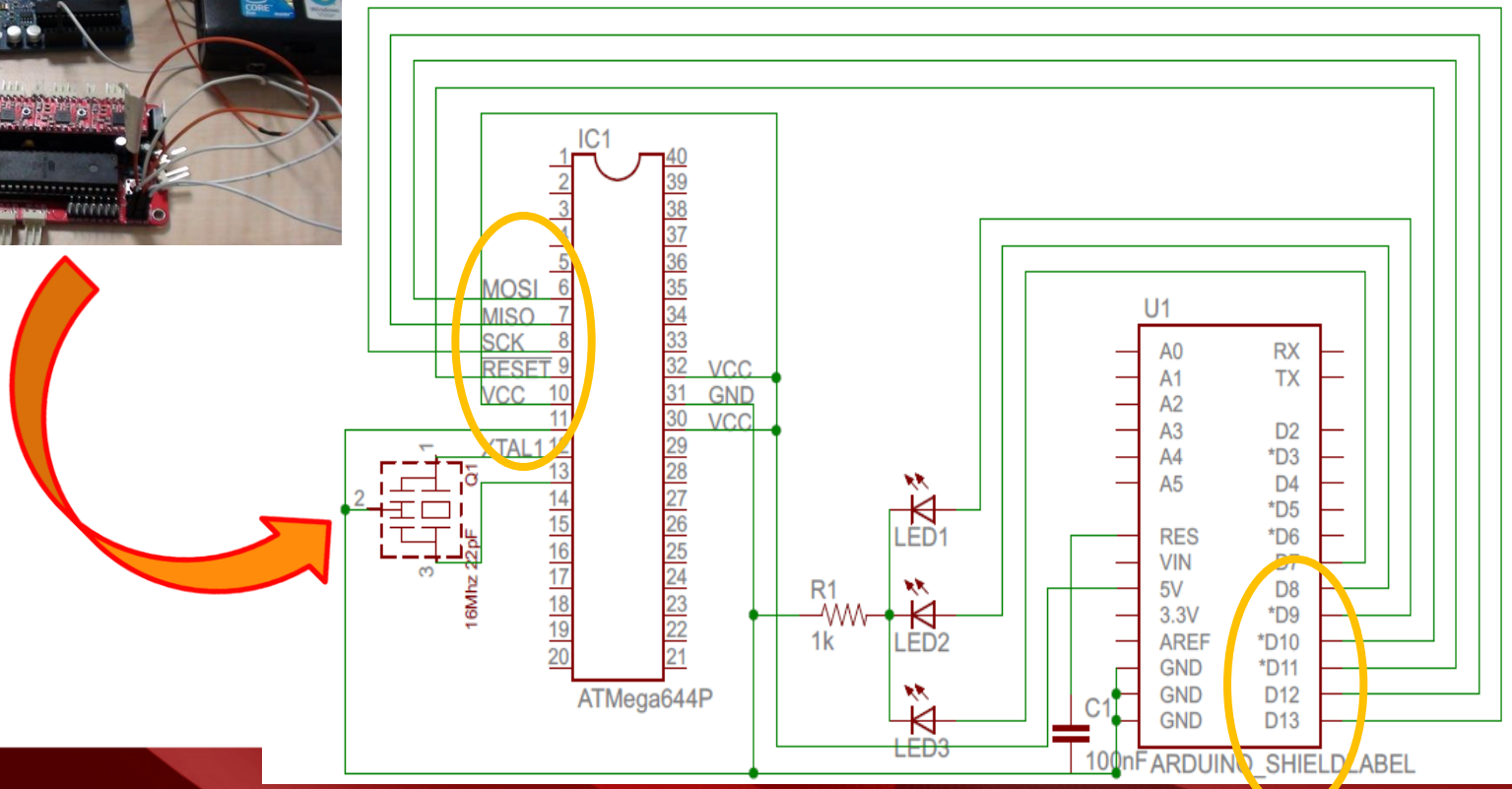
- 開啟 ArduinoISP 程式
- Compile & upload
  - Arduino 即可作為燒錄器用



2. Sanguino bootloader

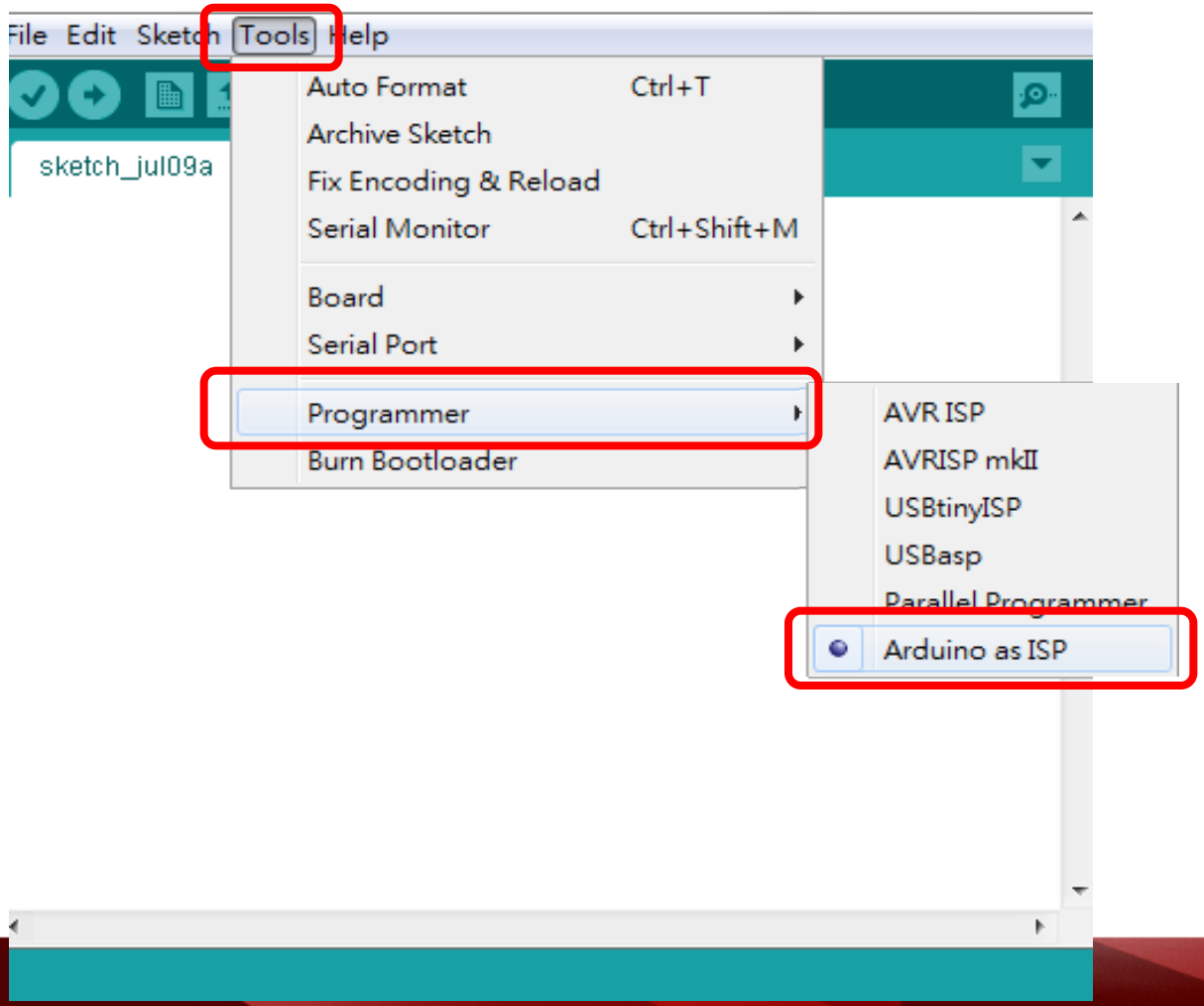
- **Arduino 的 SPI 與 Sanguinololu 的 SPI 連接**
  - 需要供電 5V 給 Sanguinololu





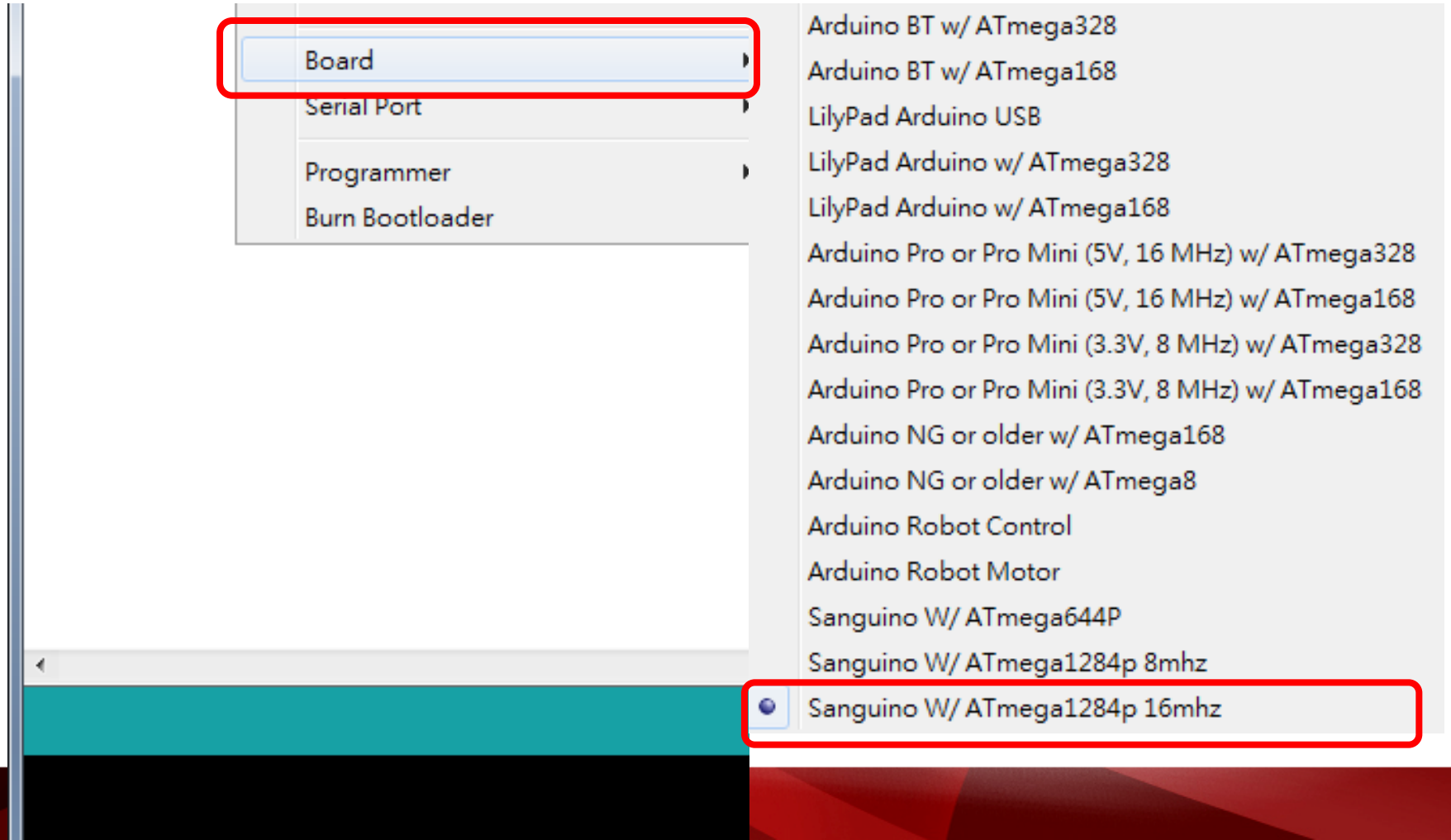
2. Sanguino bootloader

- **Arduino IDE 選取 “Arduino as ISP”**



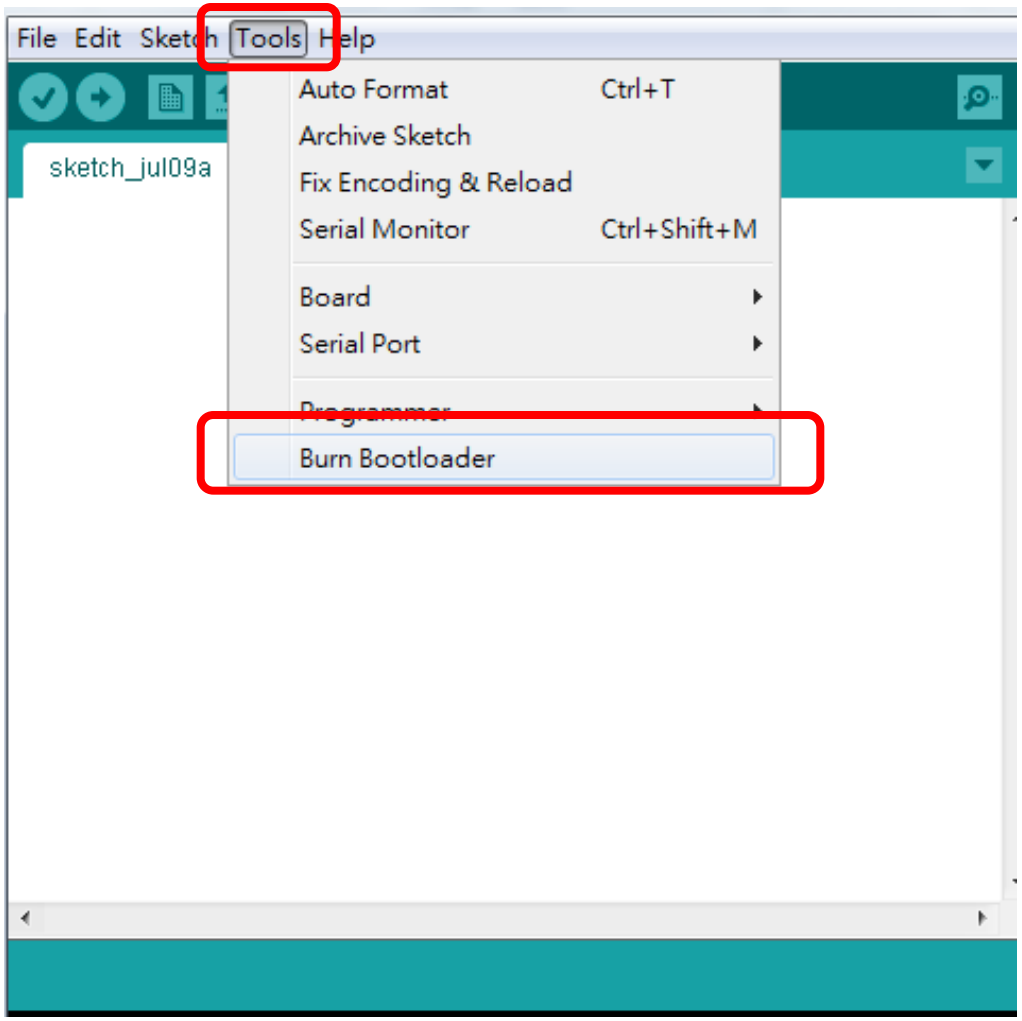
2. Sanguino bootloader

- 選擇 **Sanguinololu** 的版本



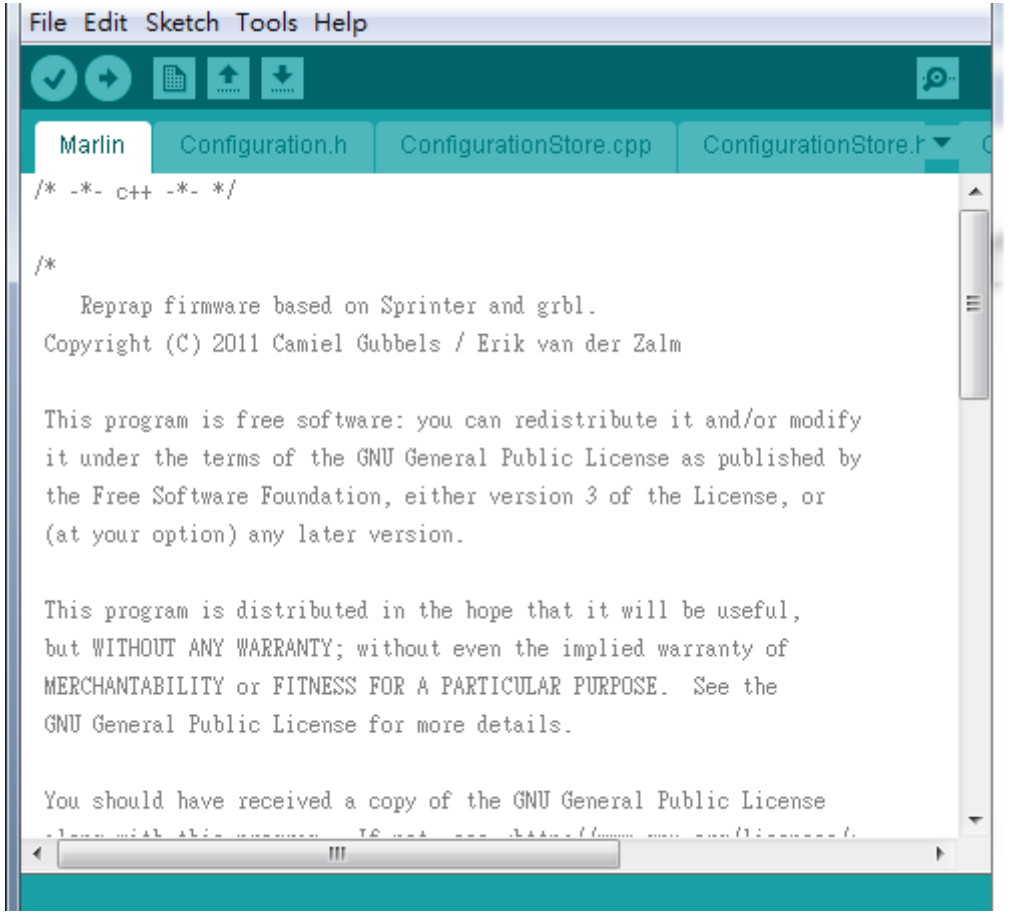
2. Sanguino bootloader

- 點選 **Burn Bootloader**
- 等候燒錄完成即可

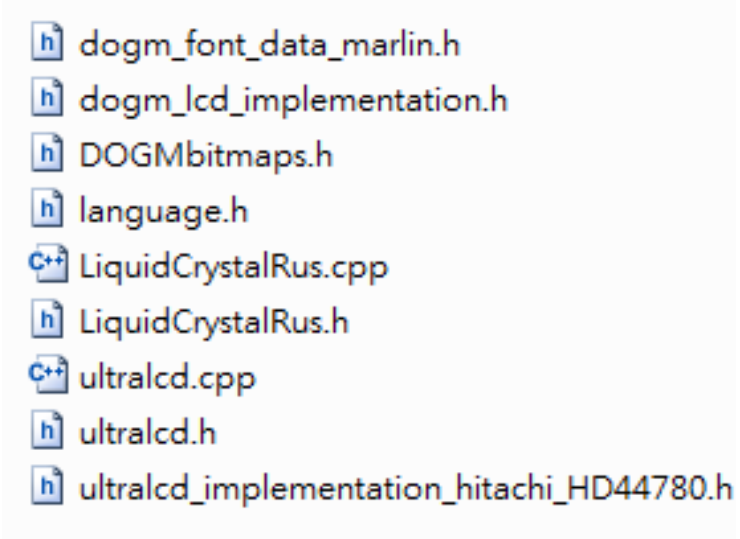


3. Upload Marlin

- 用 Arduino IDE 開啟 Marlin 的 project
- Compile & upload 即可



LCD lib


dogm_font_data_marlin.h
dogm_lcd_implementation.h
DOGMbitmaps.h
language.h
LiquidCrystalRus.cpp
LiquidCrystalRus.h
ultralcd.cpp
ultralcd.h
ultralcd_implementation_hitachi_HD44780.h

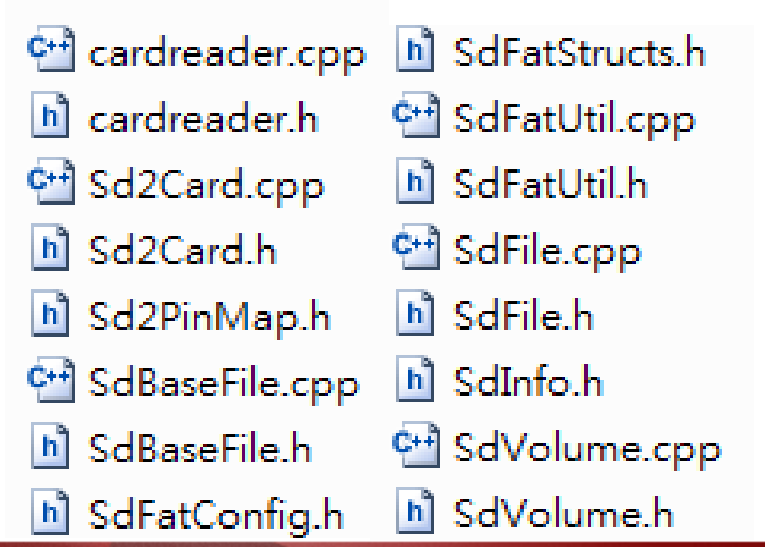
LCD lib

- **Marlin 支援的液晶顯示模組**
  - **ULTRA LCD**
  - **DOG LCD (Controller ST7565R graphic Display Family)**
  - **Hitachi HD44780 (與 Hitachi HD44780 相容)**

- **一共支援 8 種語言**

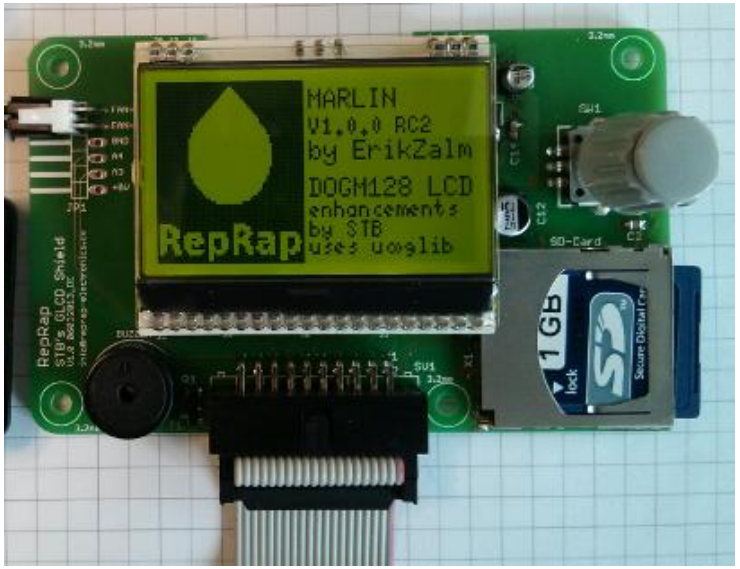
English	Polish	French
German	Spanish	Russian
Italian	Portuguese	Finnish

SD lib

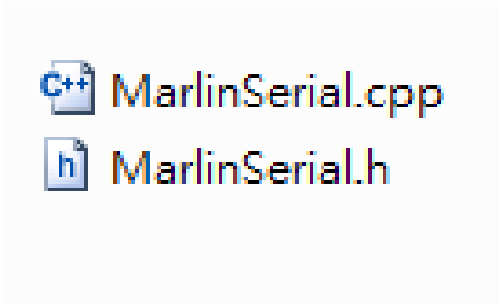

cardreader.cpp
cardreader.h
Sd2Card.cpp
Sd2Card.h
Sd2PinMap.h
SdBaseFile.cpp
SdBaseFile.h
SdFatConfig.h
SdFatStructs.h
SdFatUtil.cpp
SdFatUtil.h
SdFile.cpp
SdFile.h
SdInfo.h
SdVolume.cpp
SdVolume.h

SD lib

- **Marlin 支援讀寫 SD card**
  - **M20 - List SD card**
  - **M21 - Init SD card**
  - **M22 - Release SD card**
  - **M23 - Select SD file**
  - **M24 - Start/resume SD print**
  - **M25 - Pause SD print**
  - **M26 - Set SD position in bytes**
  - **M27 - Report SD print status**
  - **M28 - Start SD write**
  - **M29 - Stop SD write**
  - **M30 - Delete file from SD**



Serial lib

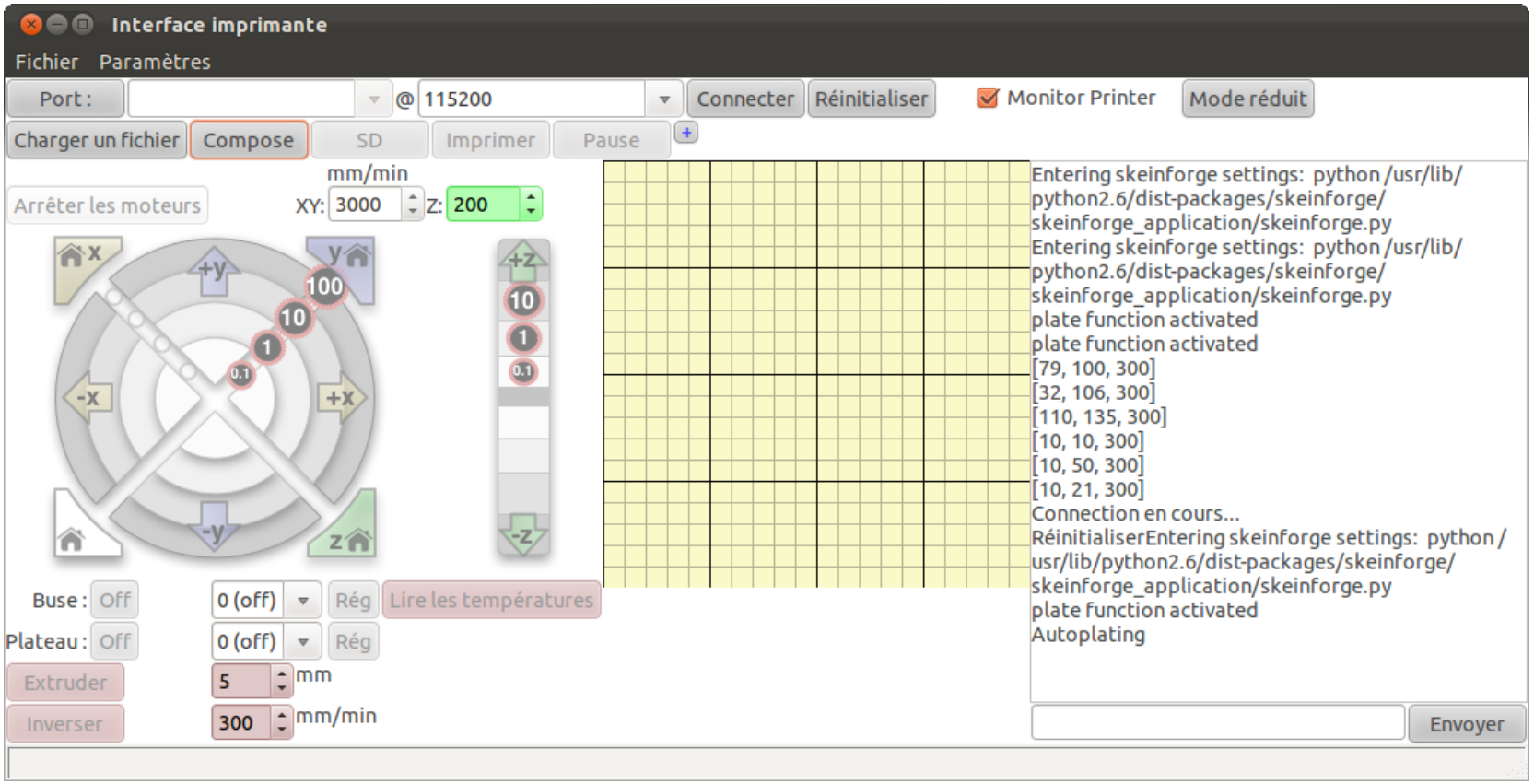


Serial lib

- **Marlin 可以透過 USB-to-serial adaptor 與 PC 作通訊**
  - **PC 端可以利用 3D printer 軟體，將 STL 檔轉成 gode 檔後，直接下 g code 指令列印出實體**
  - **最快的傳輸 baud rate 為 250000 bps**

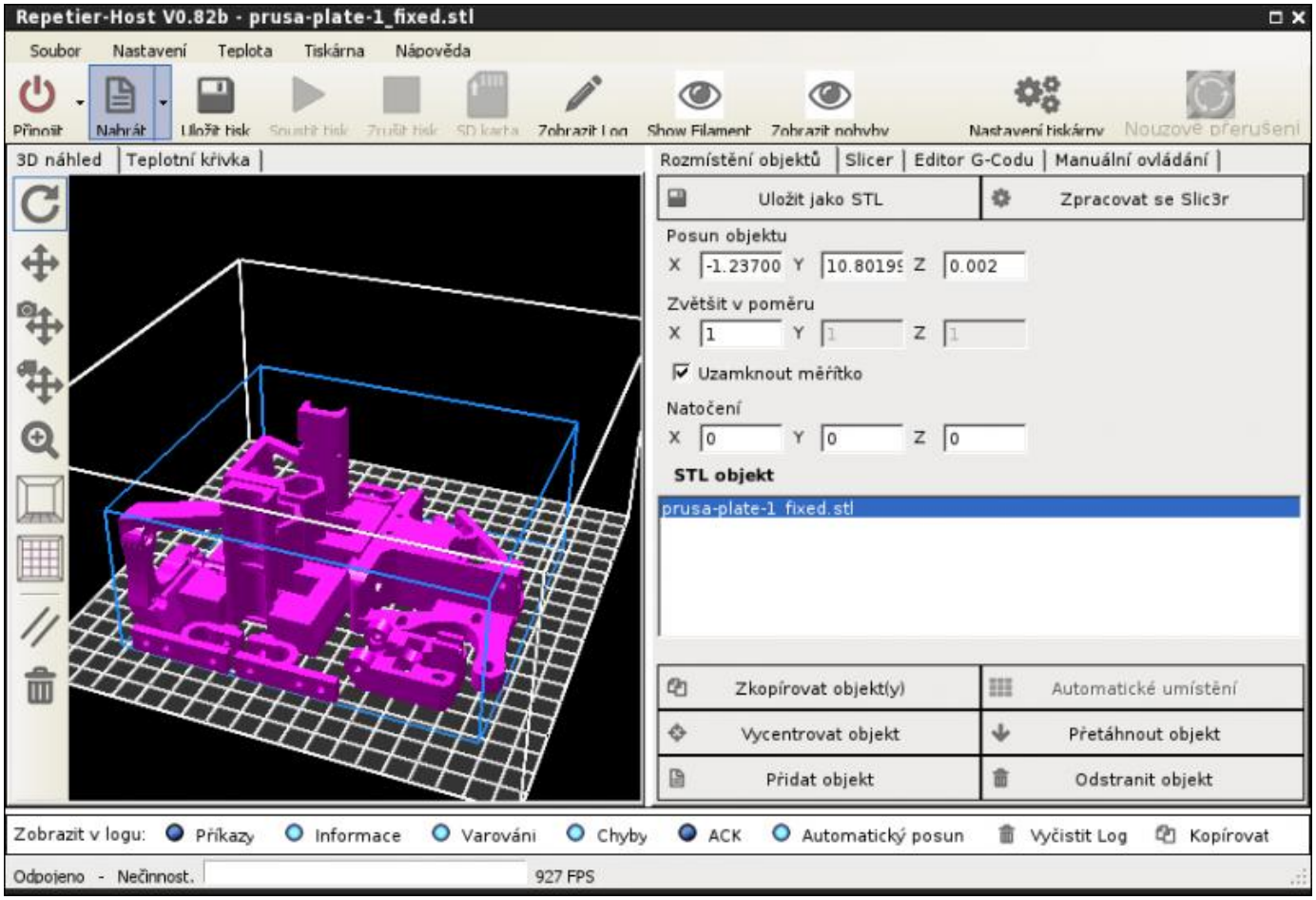
常見的3D printer 軟體

- **Printrun**



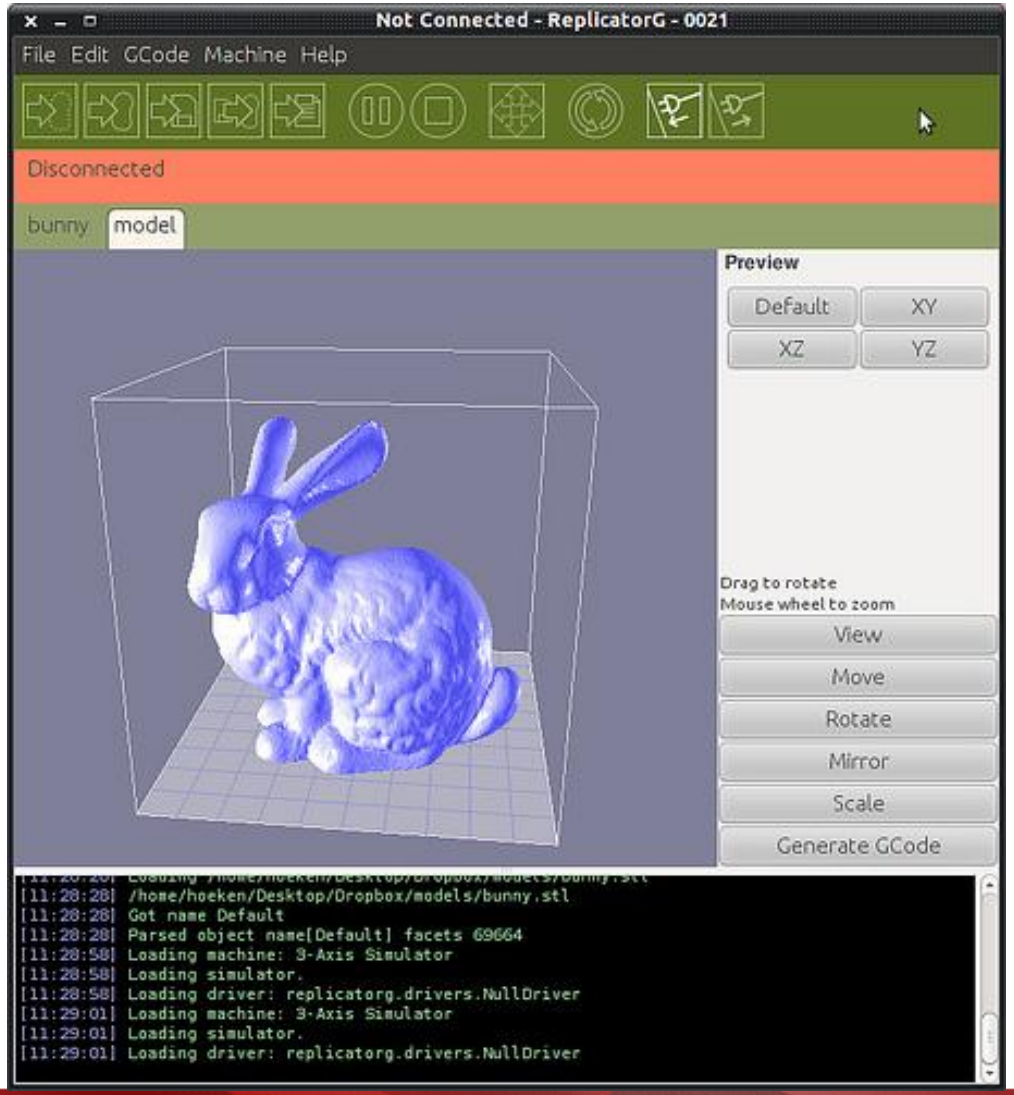
常見的3D printer 軟體

- **Repetier-Host**



常見的3D printer 軟體

- **ReplicatorG**

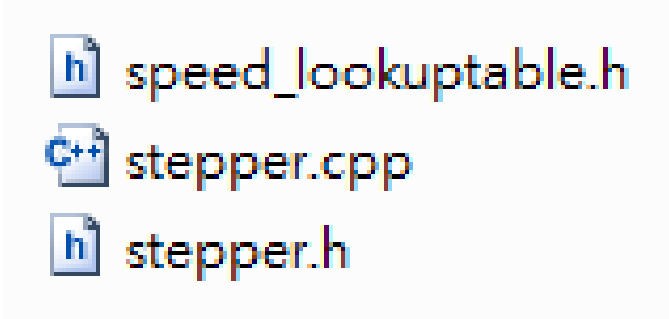


Servo lib

Servo lib

- **Marlin 有控制 RC (Radio Control) servo 的功能**
  - **Marlin_v1 新加的功能，Sprinter, Teacup 尚未支援**
  - **一般 3D printer 是用步進馬達為主**
  - **Marlin 預設時是 disable 的狀態**

Stepper lib

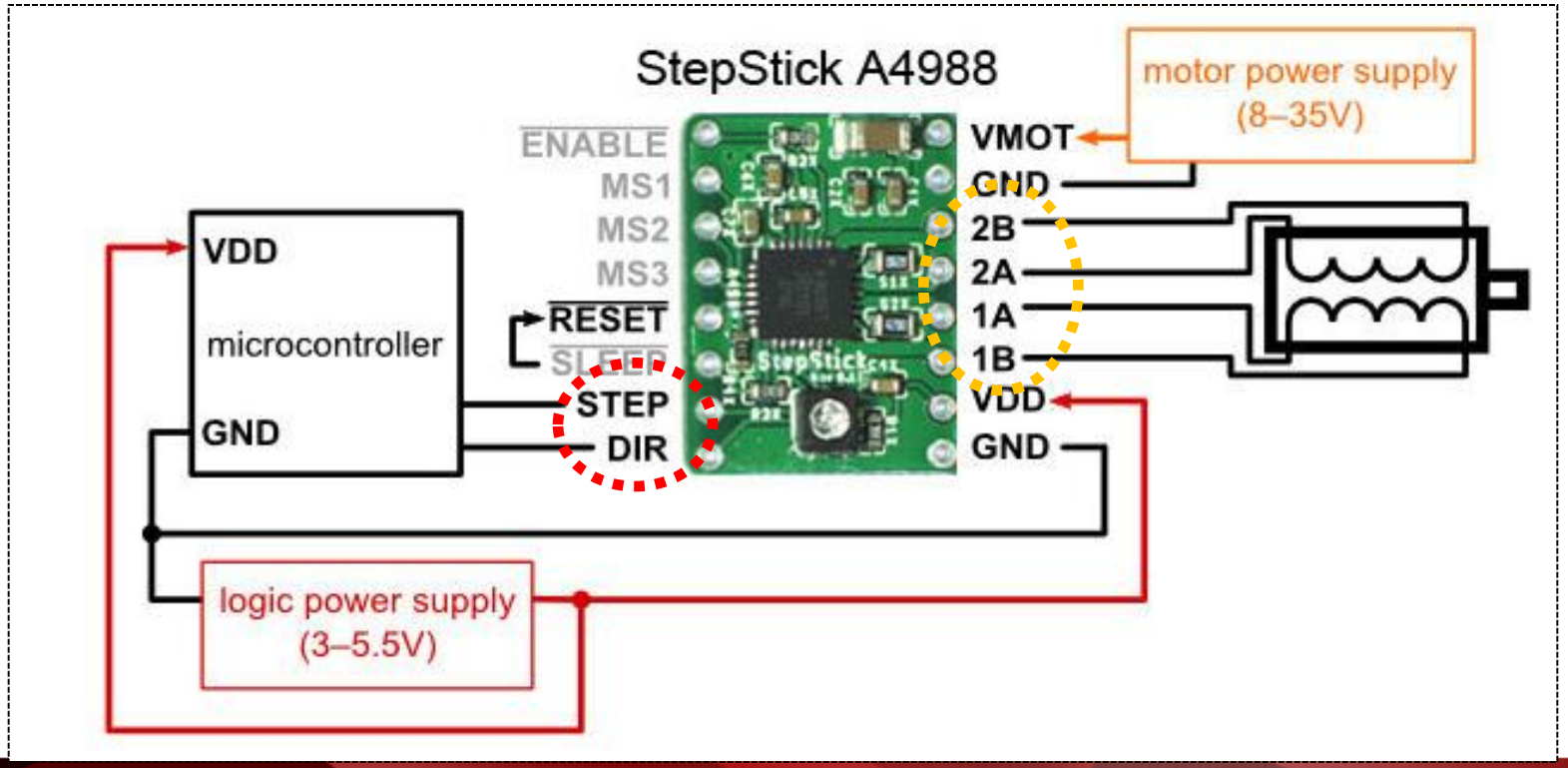


Stepper lib

- 實作驅動步進馬達的功能
  - Marlin 透過驅動板 (stepper motor driver) 控制步進馬達
  - 主要的步進驅動 IC 為 A4988
    - 由於 Marlin 的 stepper pulse 太快 (約 4us)，如果使用其他的步進驅動 IC (如 TB6560) 可能會有跟不上 stepper pulse 的問題，這部分 user 必須更改 Marlin 的程式， 使 stepper pulse 減慢速度。
- 檢查 endstop 否是被觸動
  - 被觸動時，會限制馬達持續往同一方向移動

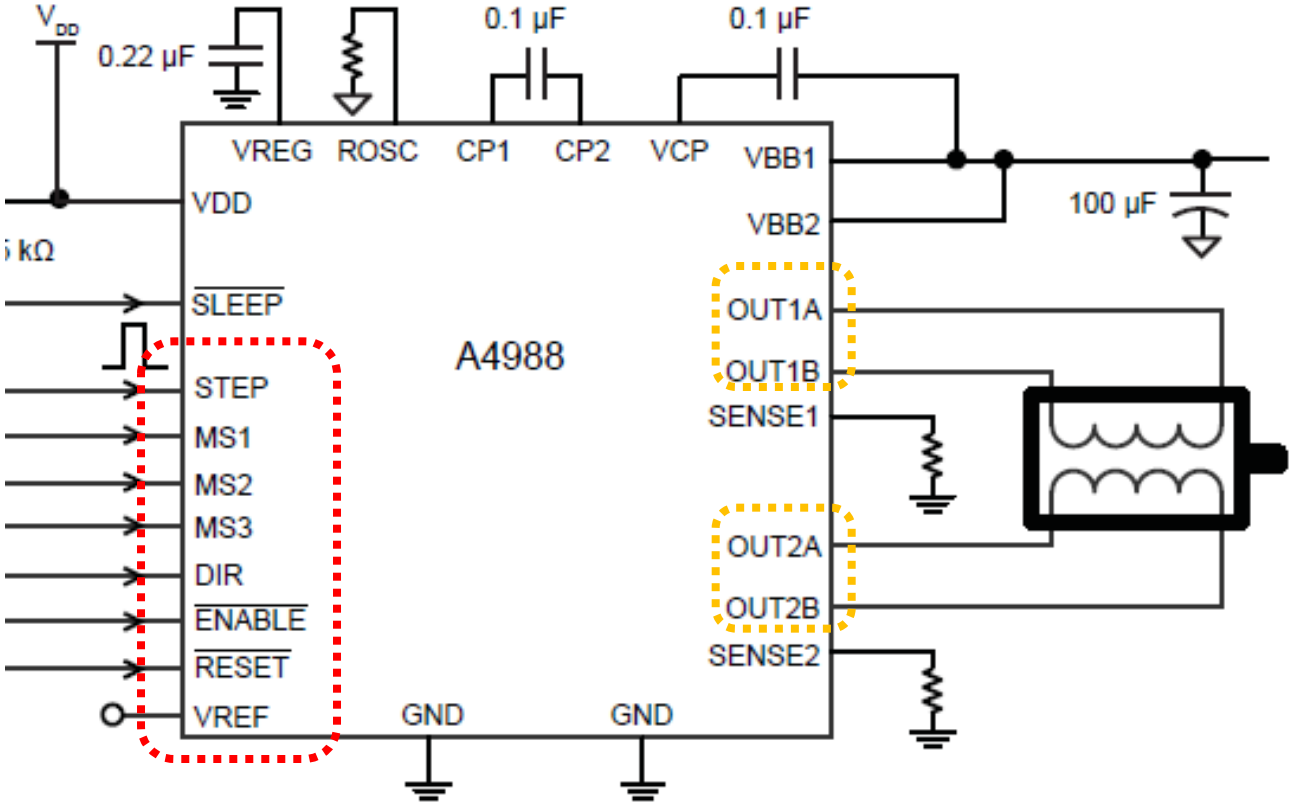
A4988

- **A4988 的電路圖**
  - 最大馬達線圈電壓: **35V**
  - 最大馬達線圈電流: **2A**



A4988

- **A4988 方塊圖**



A4988

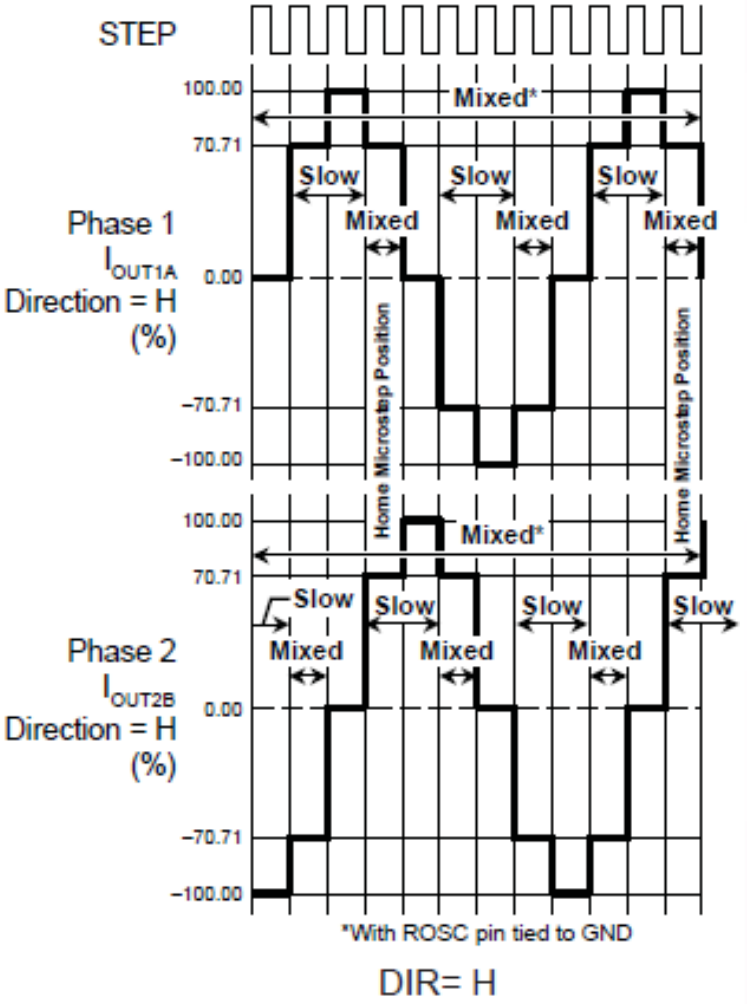
- **Microstep 的設定**
  - 可設定 **1/1, 1/2, 1/8, 1/16 mircostep**

Table 1. Microstepping Resolution Truth Table

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

A4988

- 可藉由 **RESET** 腳位重置馬達線圈電流相位
  - **RESET** 設 **high** 後，馬達線圈電流相位會回到 **home state**
- **SLEEP** 模式
  - **SLEEP** 設 **low**，會 關閉所有 **MOSFET**，降低晶片全部耗電。設 **high** 則是 **normal operation**。



A4988

- **Mixed Decay Operation**
  - 驅動步進馬達時，可以減少發生失步的情況

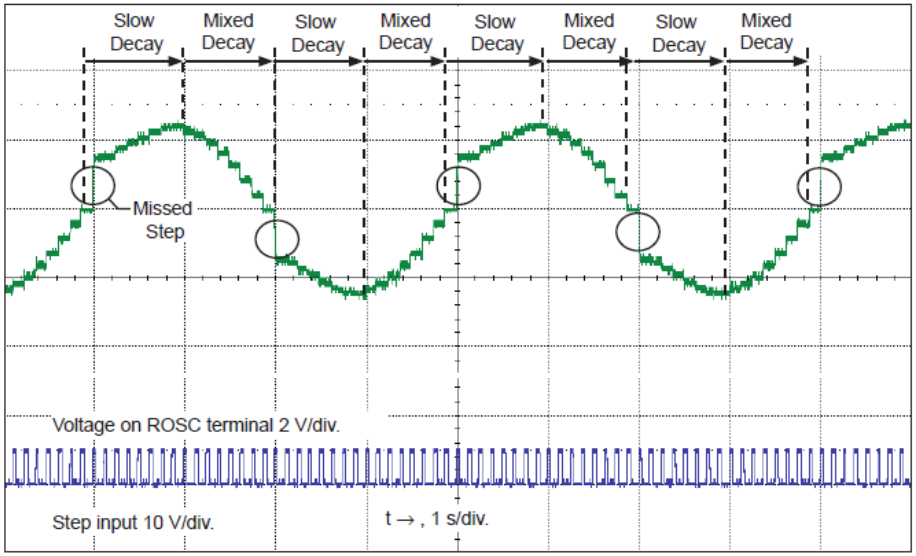


Figure 2. Missed steps in low-speed microstepping

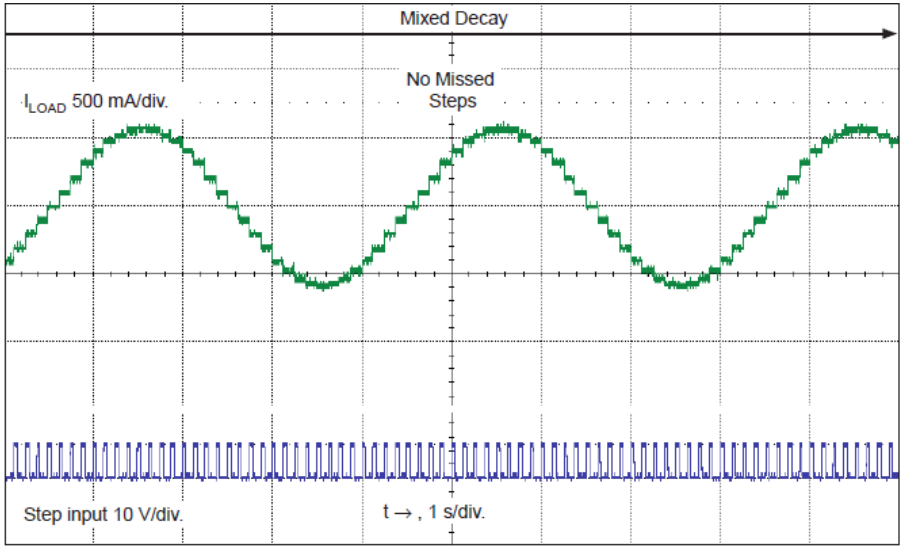
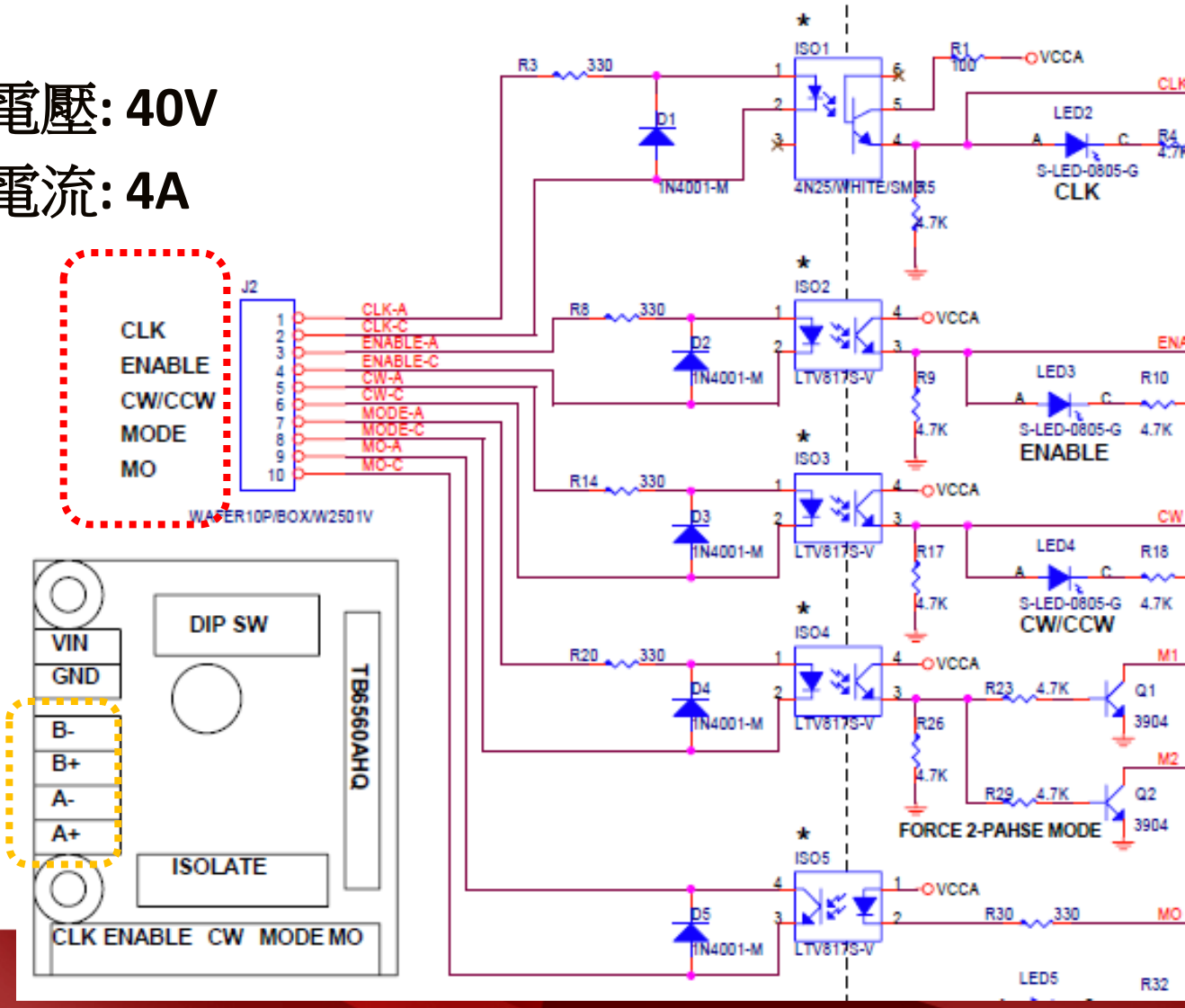


Figure 3. Continuous stepping using automatically-selected mixed stepping (ROSC pin grounded)

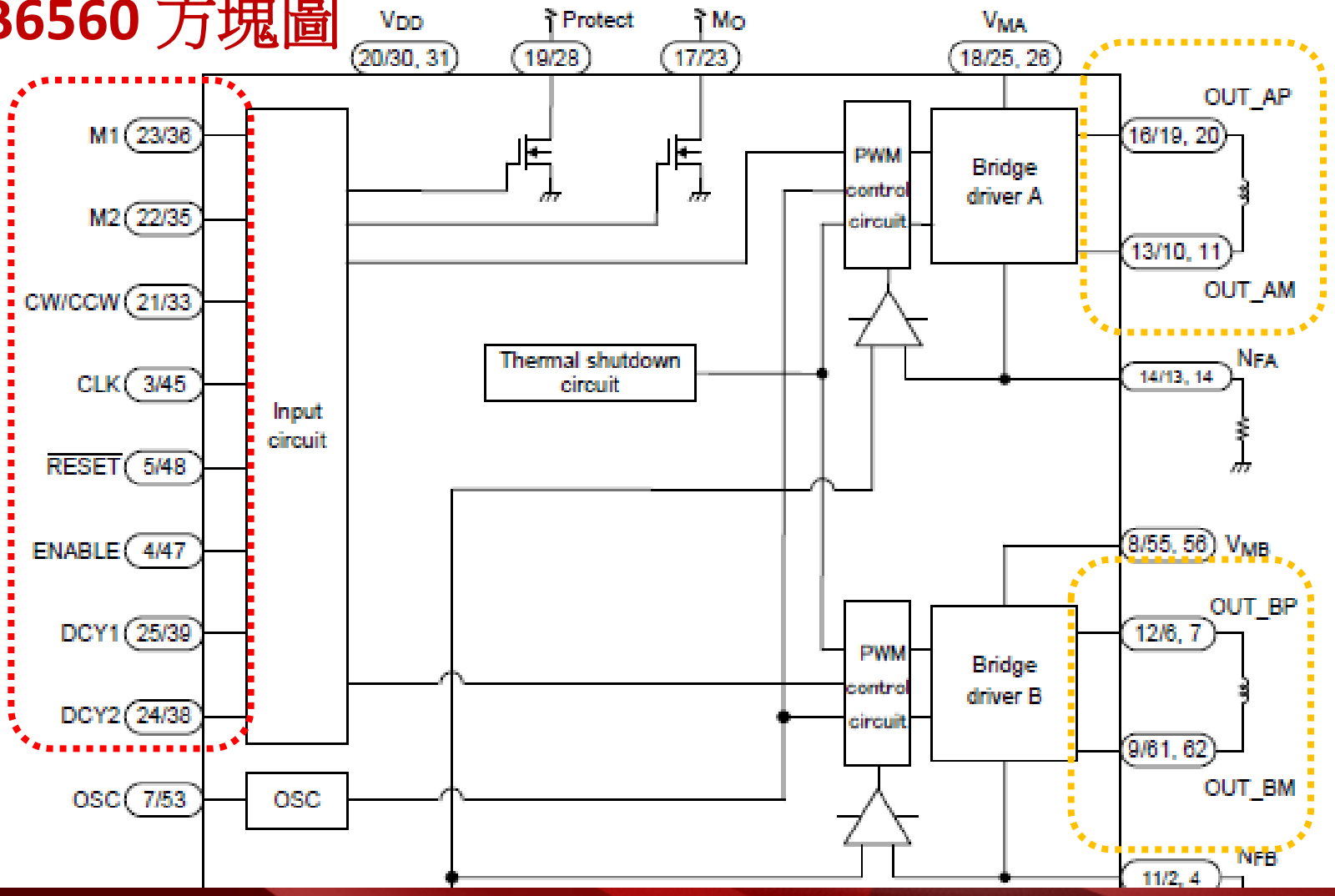
TB6560

- **TB6560 電路圖**
  - 最大馬達線圈電壓: 40V
  - 最大馬達線圈電流: 4A



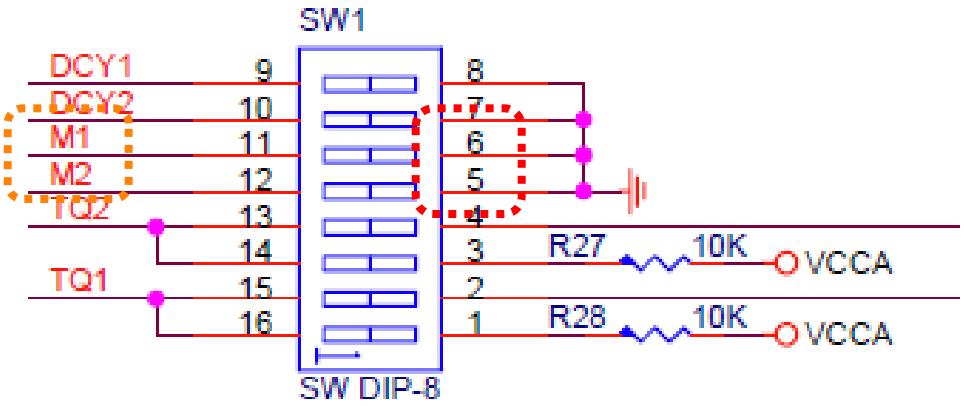
TB6560

- **TB6560 方塊圖**



TB6560

- **Microstep 的設定**
  - 可設定 **1/1, 1/2, 1/8, 1/16 mircostep**



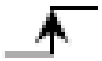
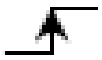
Inputs		Mode (Excitation)
M2	M1	
L	L	2-phase
L	H	1-2-phase
H	L	4W1-2-phase
H	H	2W1-2-phase

MODE

S5	S6	MODE
1	1	1/1
1	0	1/2
0	0	1/8
0	1	1/16

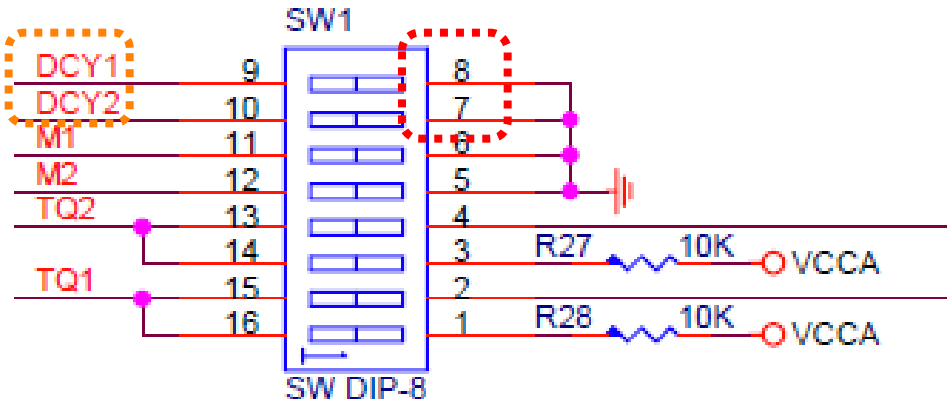
TB6560

- **CW/CCW 的設定**

Inputs				Output Mode
CLK	CW/CCW	$\overline{\text{RESET}}$	ENABLE	
	L	H	H	CW
	H	H	H	CCW
X	X	L	H	Initial mode
X	X	X	L	Z

TB6560

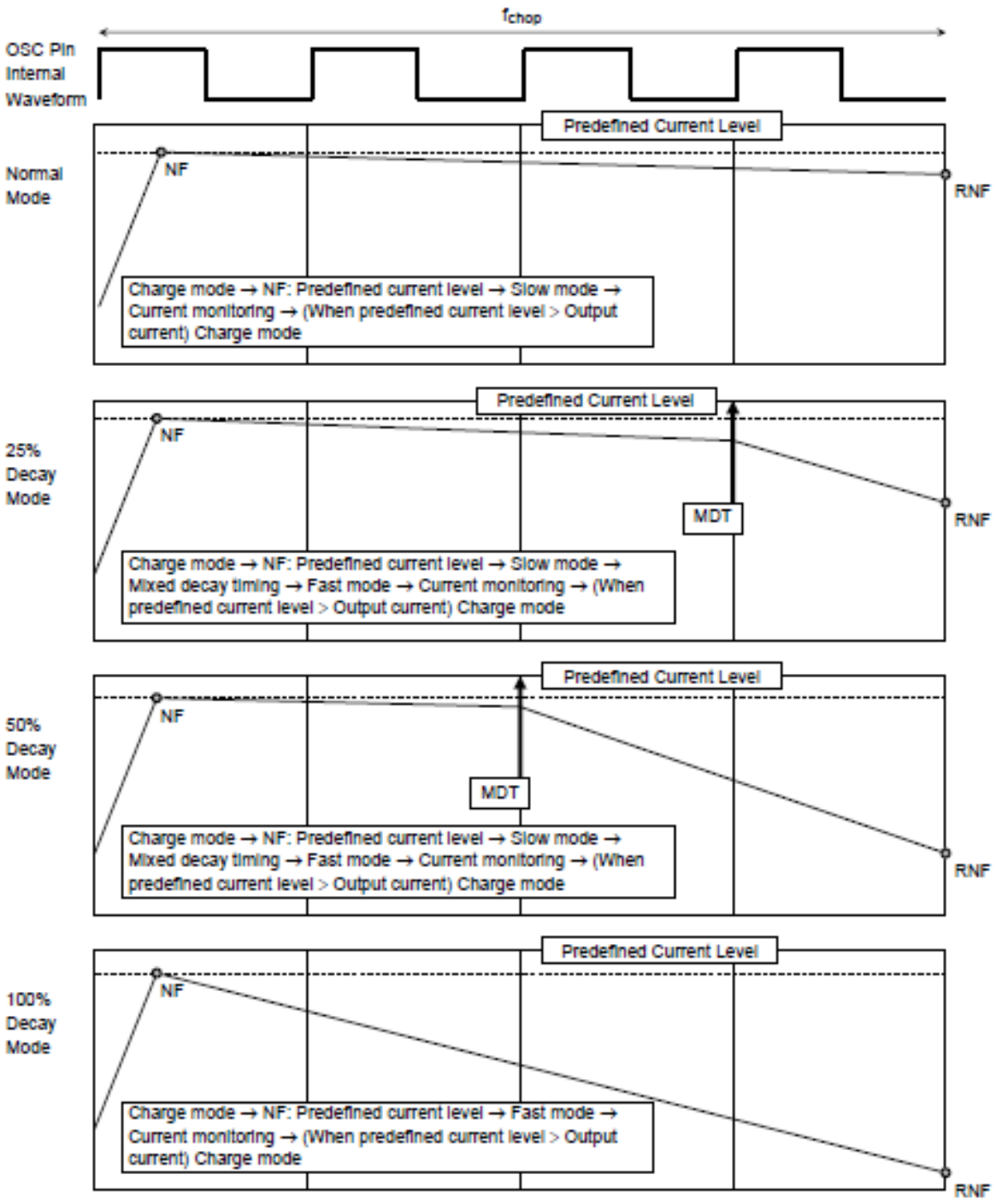
- **Decay 的設定**



DECAY

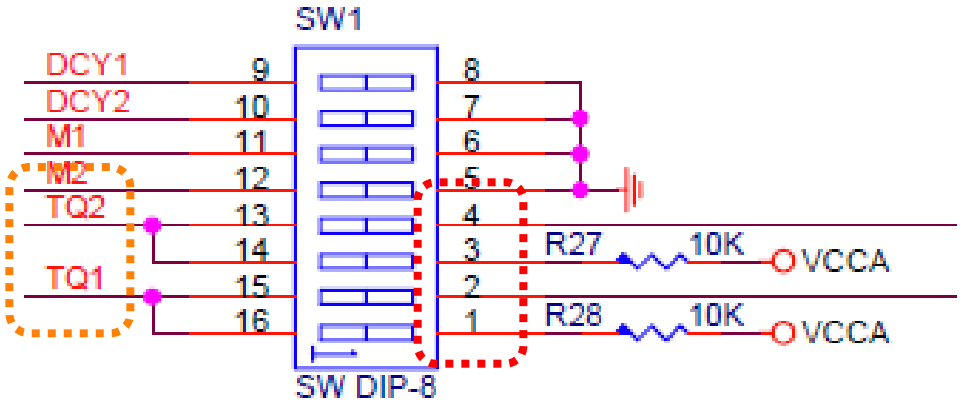
S7	S8	DECAY
1	1	0%
1	0	25%
0	1	50%
0	0	100%

DCY2	DCY1	Current Decay Setting
L	L	Normal 0%
L	H	25% Decay
H	L	50% Decay
H	H	100% Decay


f_{chop}
OSC Pin Internal Waveform
Normal Mode
Predefined Current Level
NF
RNF
Charge mode → NF: Predefined current level → Slow mode → Current monitoring → (When predefined current level > Output current) Charge mode
25% Decay Mode
Predefined Current Level
NF
MDT
RNF
Charge mode → NF: Predefined current level → Slow mode → Mixed decay timing → Fast mode → Current monitoring → (When predefined current level > Output current) Charge mode
50% Decay Mode
Predefined Current Level
NF
MDT
RNF
Charge mode → NF: Predefined current level → Slow mode → Mixed decay timing → Fast mode → Current monitoring → (When predefined current level > Output current) Charge mode
100% Decay Mode
Predefined Current Level
NF
RNF
Charge mode → NF: Predefined current level → Fast mode → Current monitoring → (When predefined current level > Output current) Charge mode

TB6560

- **Torque 的設定**



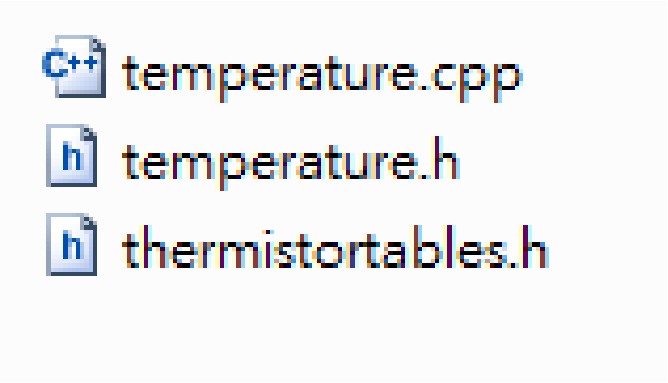
TQ2	TQ1	Current Ratio
L	L	100%
L	H	75%
H	L	50%
H	H	20% (Weak excitation)

CURRENT SEL.

S4	S3	S2	S1	CURRENT
0	0	0	0	100% -> 100%
0	0	1	0	100% -> 75%
1	0	0	0	100% -> 50%
1	0	1	0	100% -> 20%
0	0	0	1	75% -> 75%
1	0	0	1	75% -> 20%
0	1	0	0	50% -> 50%
0	1	1	0	50% -> 20%
0	1	0	1	20% -> 20%

0 = OFF

Temperature lib

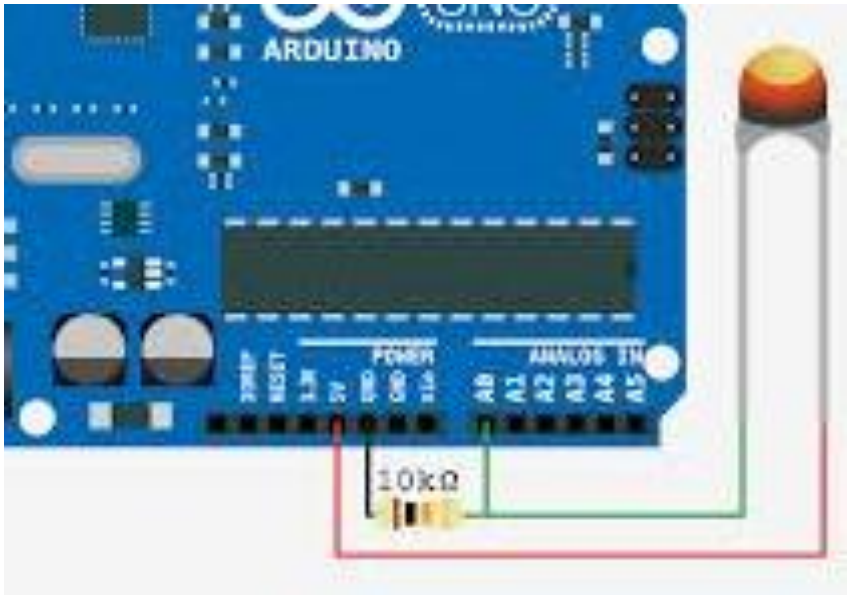


Temperature lib

- **Marlin 可以選用熱敏電阻 (thermistor) 或是熱電偶 (thermocouple)偵測溫度**
  - 加熱板只支援熱敏電阻
  - 擠出頭只支援一組熱電偶

Thermistor

- 熱敏電阻的接法
  - 一端接VCC (通常是 5V)
  - 一端接 GND 跟 Analog



- 熱敏電阻的用法
  - 由 Analog 讀取電壓的伏特數，得知溫度的變化

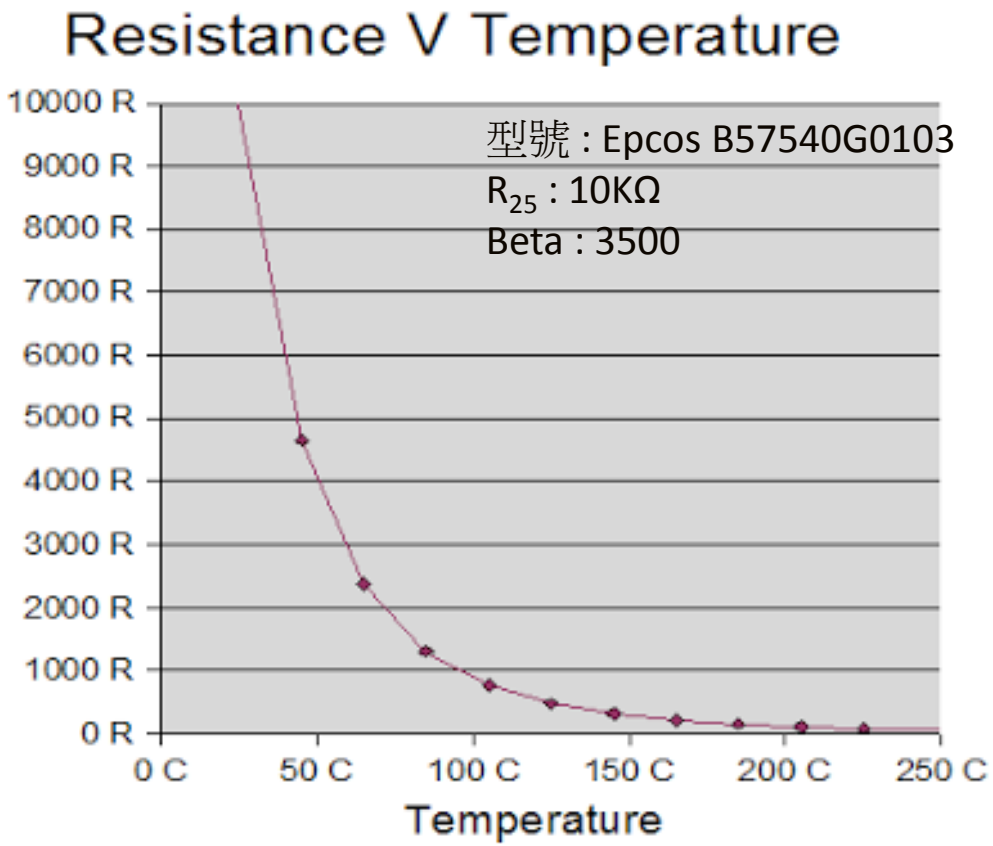
Thermistor

- 熱敏電阻的溫度與電阻值的關係式

$$R = R_0 e^{B\cdot(1/T - 1/T_0)}$$

Ro 是常溫 To 的電阻值

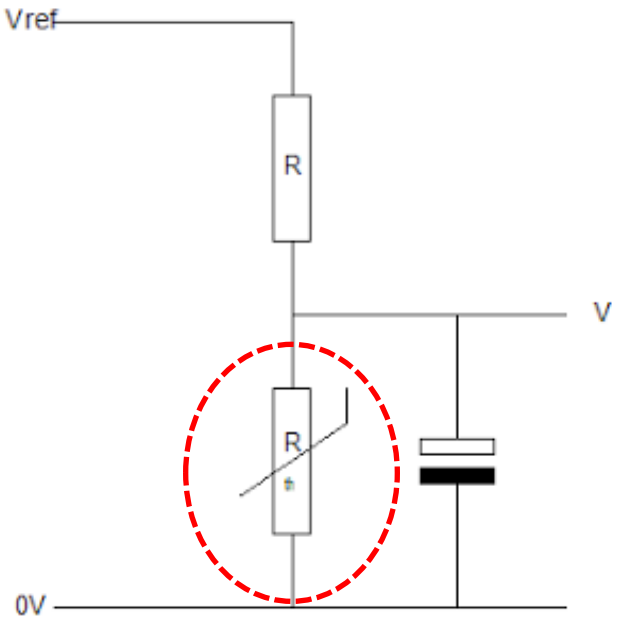
B 是 beta 值

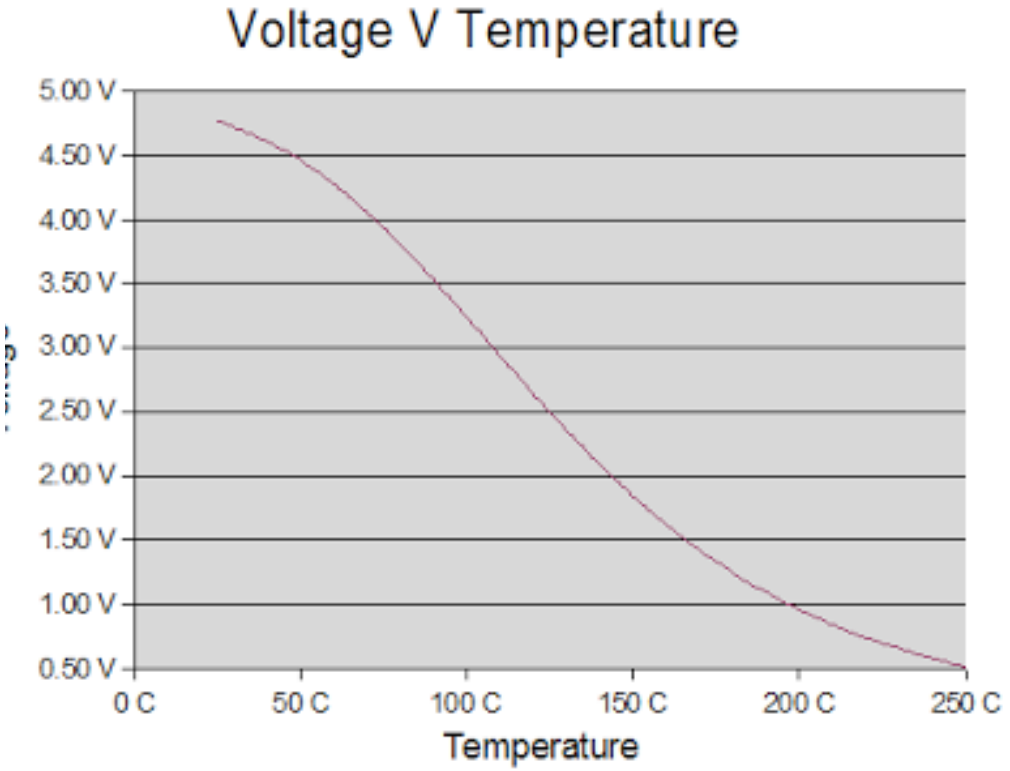


Thermistor

- 熱敏電阻的溫度與電壓的關係式

$$V = V_{ref} \cdot R_{th} / (R + R_{th})$$





Thermistor

- 可以利用 createTemperatureLookup.py
  - 建立熱敏電阻溫度與電壓的查表

```
C:\Python27>python createTemperatureLookup.py
// Thermistor lookup table for RepRap Temperature Sensor Boards (http://make.rrr
f.org/ts)
// Made with createTemperatureLookup.py (http://svn.reprap.org/trunk/reprap/firm
ware/Arduino/utilities/createTemperatureLookup.py)
// ./createTemperatureLookup.py --r0=100000 --t0=25 --r1=0 --r2=4000 --beta=4092
 --max-adc=2047
// r0: 100000
// t0: 25
// r1: 0
// r2: 4000
// beta: 4092
// max adc: 2047
#define NUMTEMPS 45
short temptable[NUMTEMPS][2] = {
   {1, 1146},
   {47, 332},
   {93, 275},
   {139, 245},
   {185, 226},
   {231, 211},
```

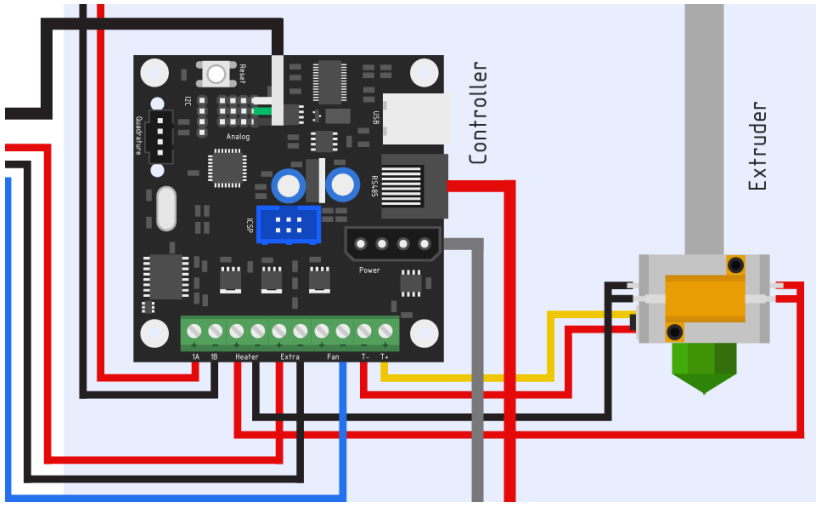
Thermistor

- 常用的熱敏電阻
  - EPCOS 100K Thermistor (B57540G0104F000)
  - EPCOS 100K Thermistor (B57560G1104F)
  - EPCOS 100K Thermistor (B57560G104F)
  - RRRF 100K Thermistor
  - RRRF 10K Thermistor
  - RS 10K Thermistor
  - Honeywell 100K Thermistor (135-104LAG-J01)
  - ATC Semitec 104GT-2
  - KTY82-210 (Philips) (2kOhm SMD)

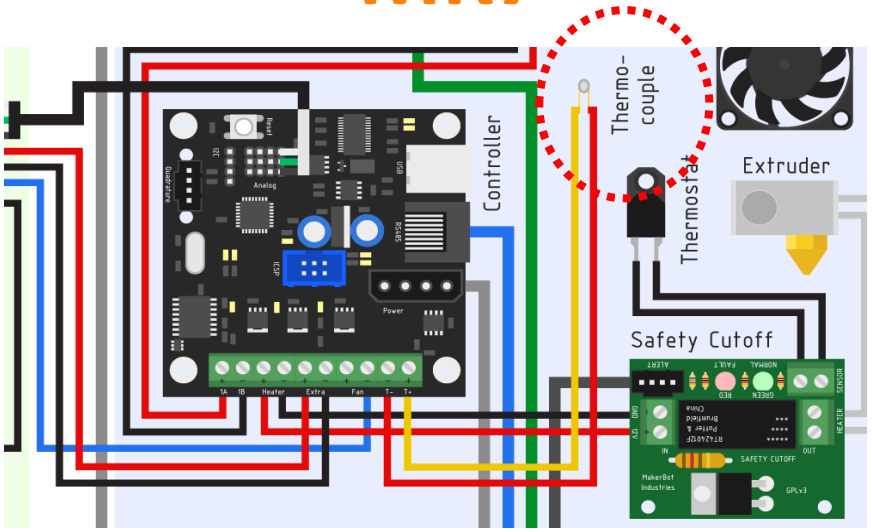
Thermocouple

- **MakerBot 的擠出頭從熱敏電阻變革為熱電偶**
  - **MakerBot Replicator MK5, MK6 用熱敏電阻**
  - **MakerBot Replicator MK7 之後用熱電偶**

MK5

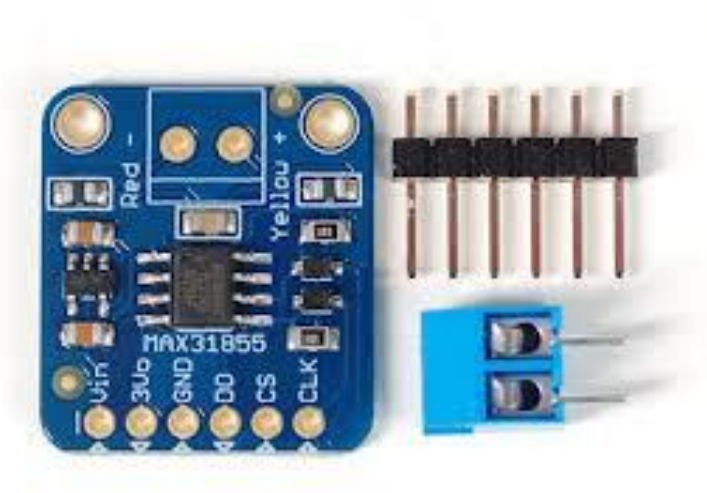


MK7



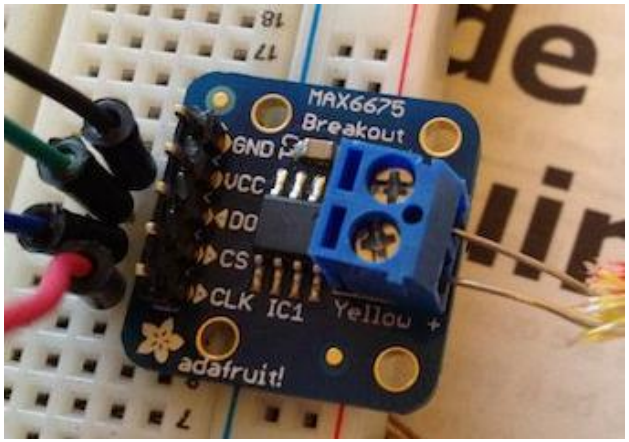
Thermocouple

- 熱電偶的接法
  - Marlin 支援兩種轉接板 Max6675 及 AD595



Max6675

- **Max6675 簡介**
  - **12 bit 熱電偶式的 ADC**
  - **SPI 通訊介面**
  - **可偵測溫度為 0˚C ~ 1023.75˚C**
- **Max6675 的接法**
  - **以 Arduino SPI 為例**




```
' using the following arduino pins:
'
' pin name:    not-mega:         mega(1280 and 2560)
' slave reset: 10:               53
' MOSI:        11:               51
' MISO:        12:               50
' SCK:         13:               52
'
```

Max6675

- **Max6675 溫度與電壓的關係**

$$V_{OUT} = (41\mu V / {}^{\circ}C) \times (T_R - T_{AMB})$$

V_{OUT} is the thermocouple output voltage (µV).

T_R is the temperature of the remote thermocouple junction (°C).

T_{AMB} is the ambient temperature (°C).

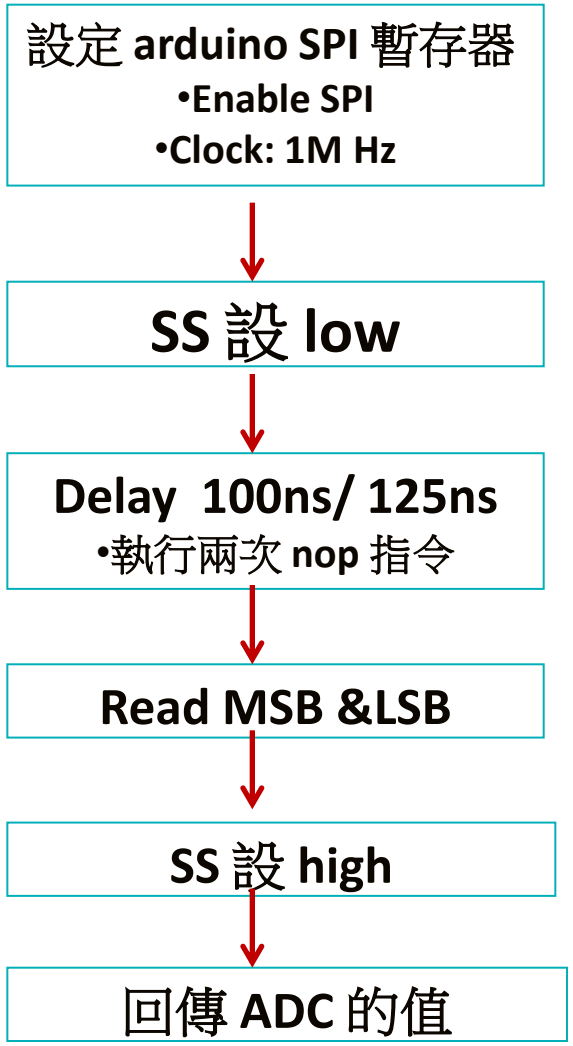
Max6675

- **Marlin 利用 Max6675 感測溫度**
  - 擠出頭溫度 =

    Max6675_val *(1024/4096)

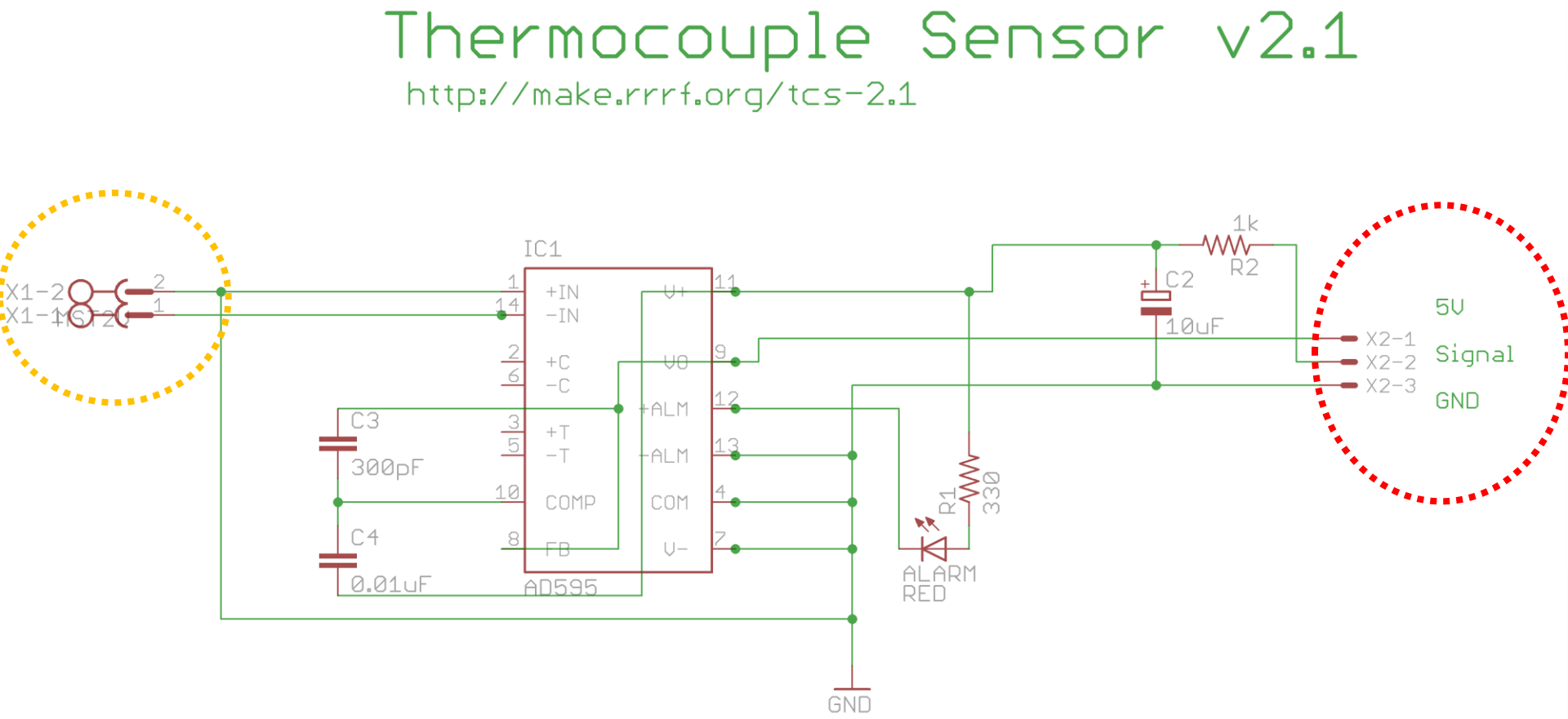
Max6675 可測溫度範圍 0˚C ~ 1023.75˚C

Max6675 的解晰度是 12 bit (4096)



AD595

- **AD595 的接法**



AD595

- **AD595 溫度與電壓的關係**
  - 每上升 1 ˚C 電壓上升 10 mV (即 1V 表示 100 ˚C)

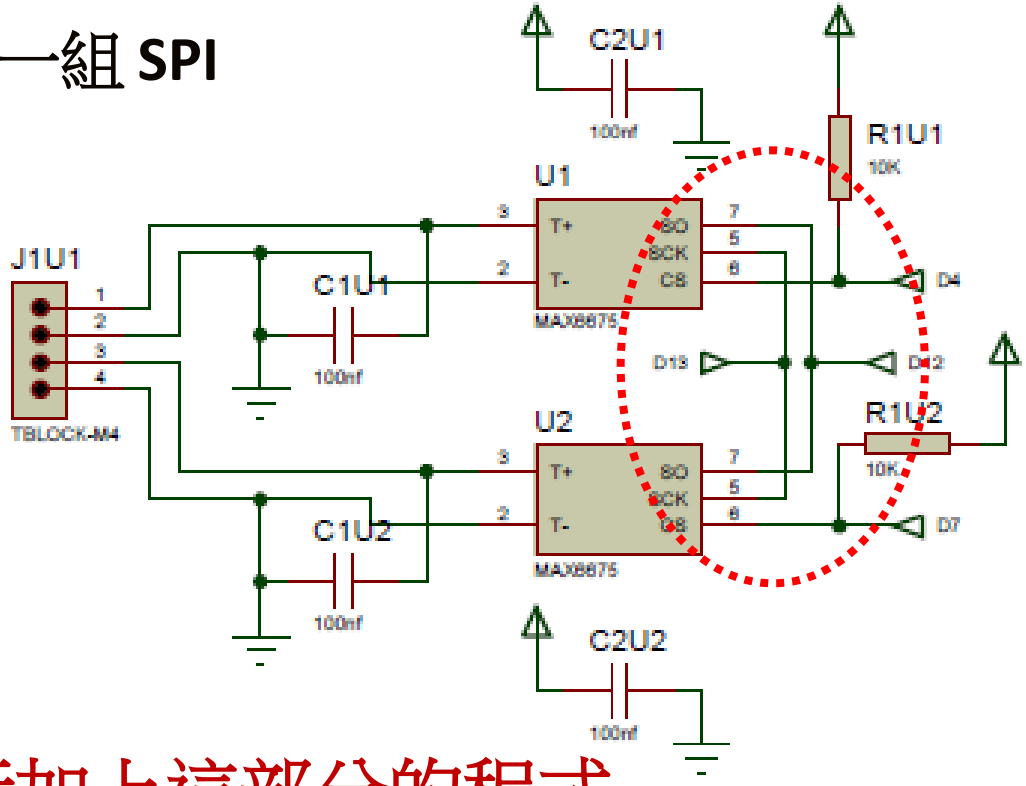
- **Marlin 利用 AD595 感測溫度**
  - 擠出頭溫度 = ADC_val / 1024.0 * (5.0 * 100.0)
  - Arduino analog 解晰度是 10 bit (1024)
  - 參考電壓是 5 V
  - AD595 的 V/˚C 比值是 100

怎麼辦?

如果兩個擠出頭都剛好是用熱電偶的話

Thermocouple

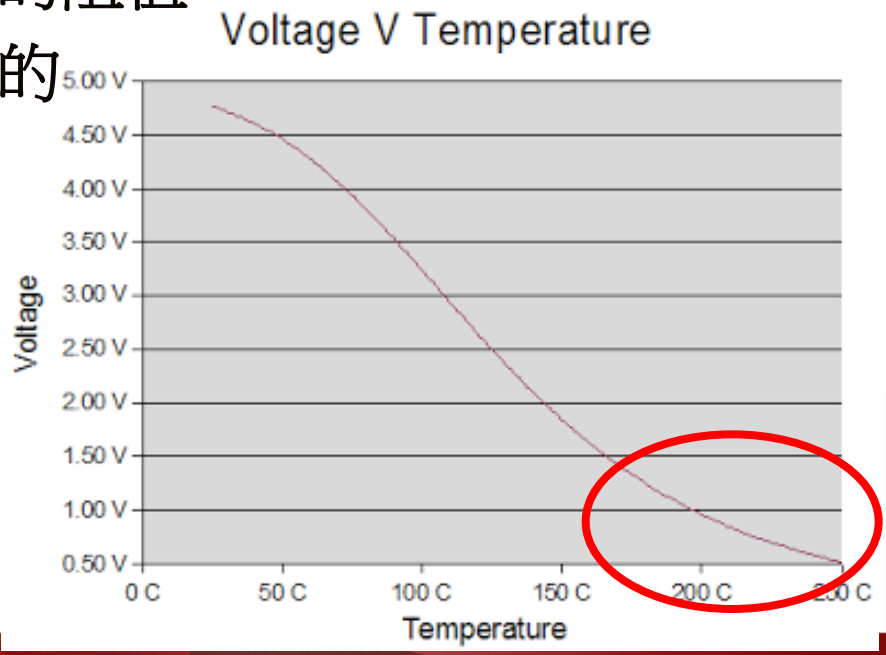
- **MakerBot 雙擠出頭的做法 (MakerBot Replicator MK8)**
  - **兩個 Max6675 轉接板接一組 SPI**
  - **SCK, SO 共用**



- **使用 Marlin 的話要自行加上這部分的程式**

Thermocouple vs. Thermistor

- **Thermister 的特性**
  - 可測溫度範圍較小
    - 依型號而定，大約在 **20 ~ 300˚C** 之間
  - 使用上，不需要轉接板或是放大電路
  - 依據感測的溫度調整熱敏電阻的阻值
  - 溫度與電阻值的關係是非線性的
  - 溫度與電壓的關係是非線性的
    - 高溫時，**V/T** 值的變動較大
    - 在 **50 ˚C** 內精確度很高



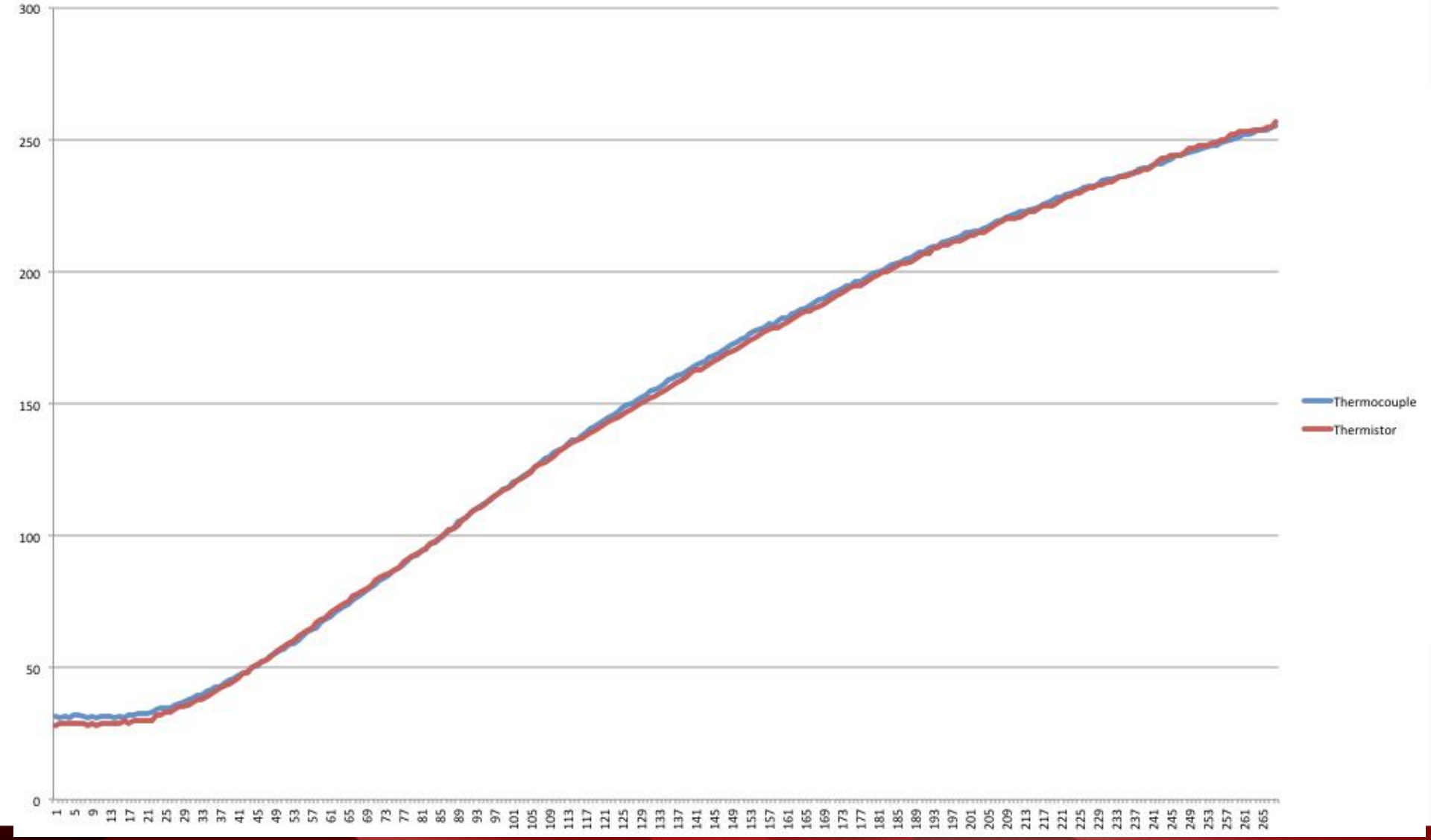
Thermocouple vs. Thermistor

- **Thermocouple 的特性**
  - 可測溫度範圍較大
    - 依材質的組成有不同的型號
    - **Type T: -200-350˚C**
    - **Type J: –40 to +750 ˚C**
    - **Type E: –50 to 740 ˚C / –110 to 140 ˚C.**
    - **Type K: –200 ˚C to +1350 ˚C**
  - 使用上，需要轉接板或是放大電路
  - 依據感測的溫度產生相對應的電壓
  - 透過轉接板，將溫度與電壓調整成線性關係

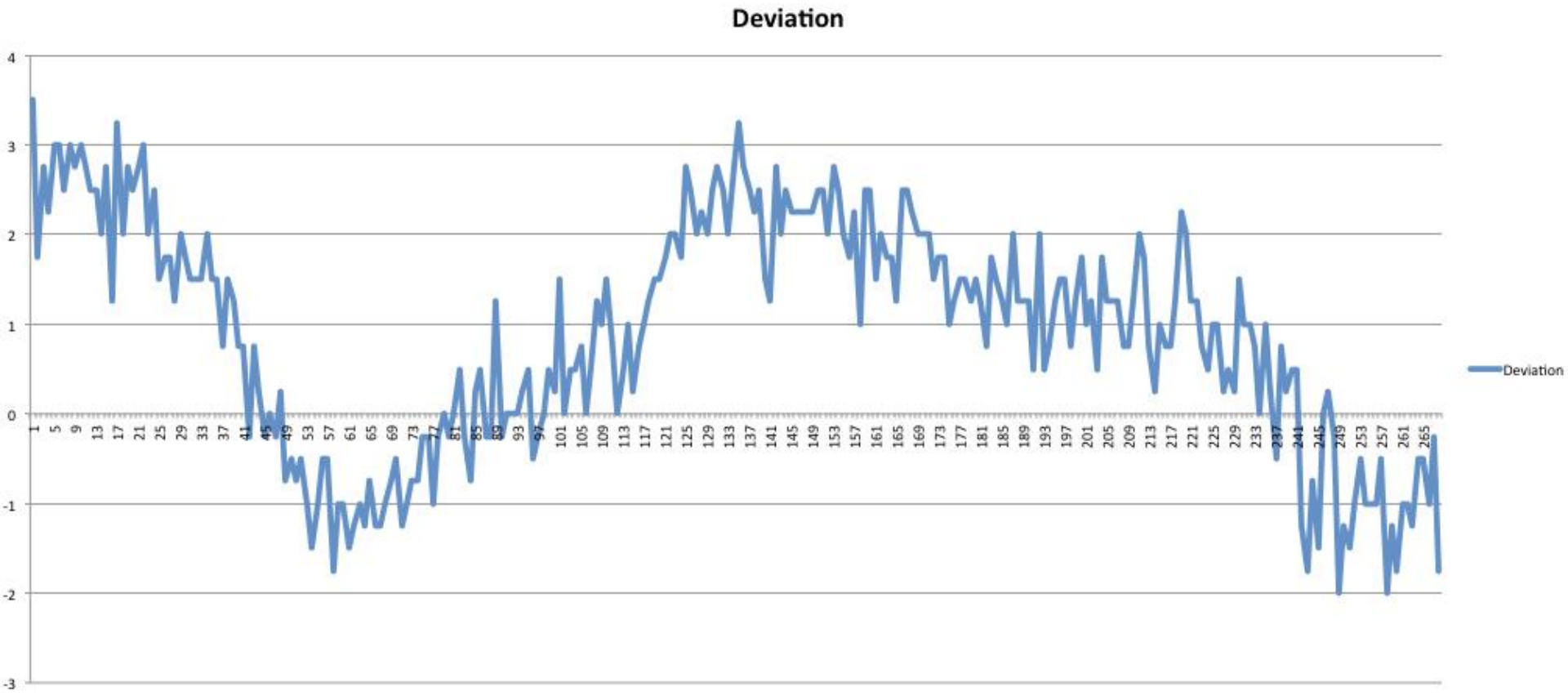
Thermocouple vs. Thermistor

- **Reprap blog 做的實驗**
- 比較熱電偶與熱敏電阻精準度
  - 從常溫下同時開始加熱到 **250˚C**，比較兩者溫度的變化
  - 實驗結果為兩者最大誤差為 **3˚C**

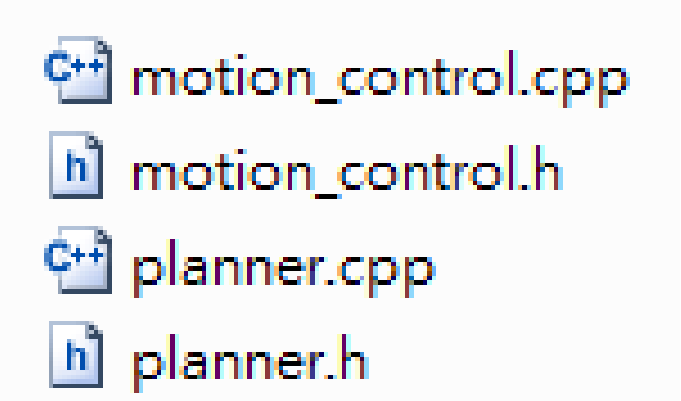
Thermocouple vs. Thermistor



Thermocouple vs. Thermistor

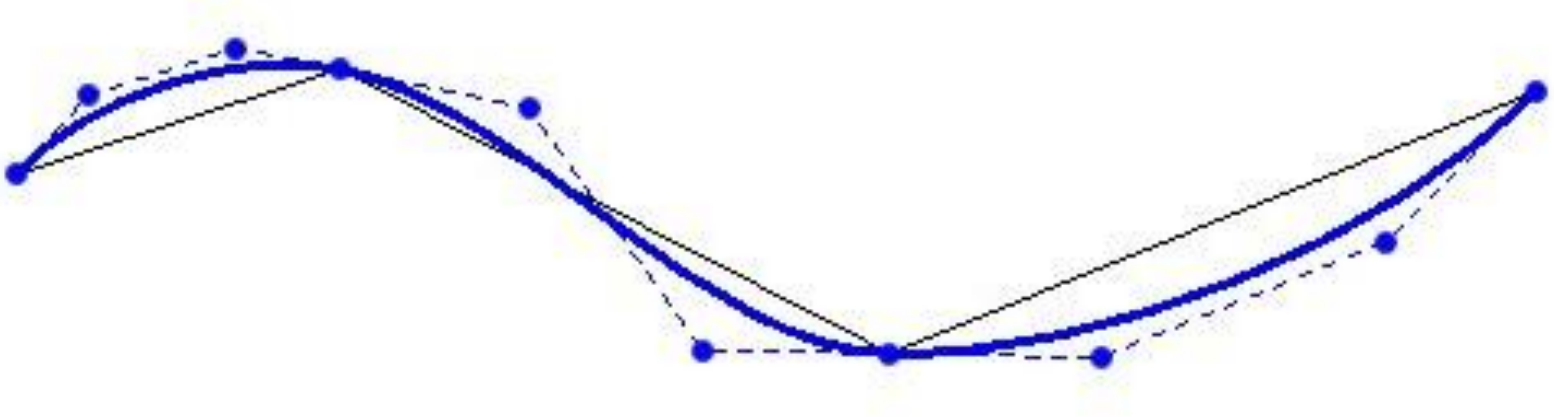


Plan motion lib

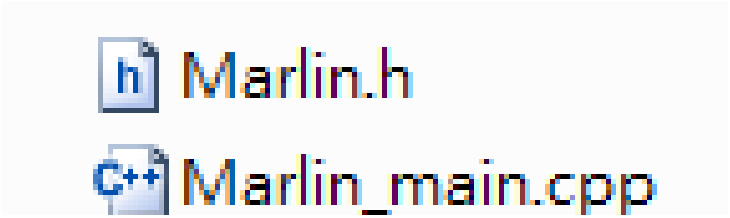


Plan motion lib

- **Marlin 軌跡規劃的實作內容**
  - 直線軌跡的規劃
  - 圓弧軌跡的規劃
  - 軌跡的梯形加減速規劃
  - 軌跡規劃的預覽 **(look ahead)**
  - 對兩個軌跡在連接處的速度規劃

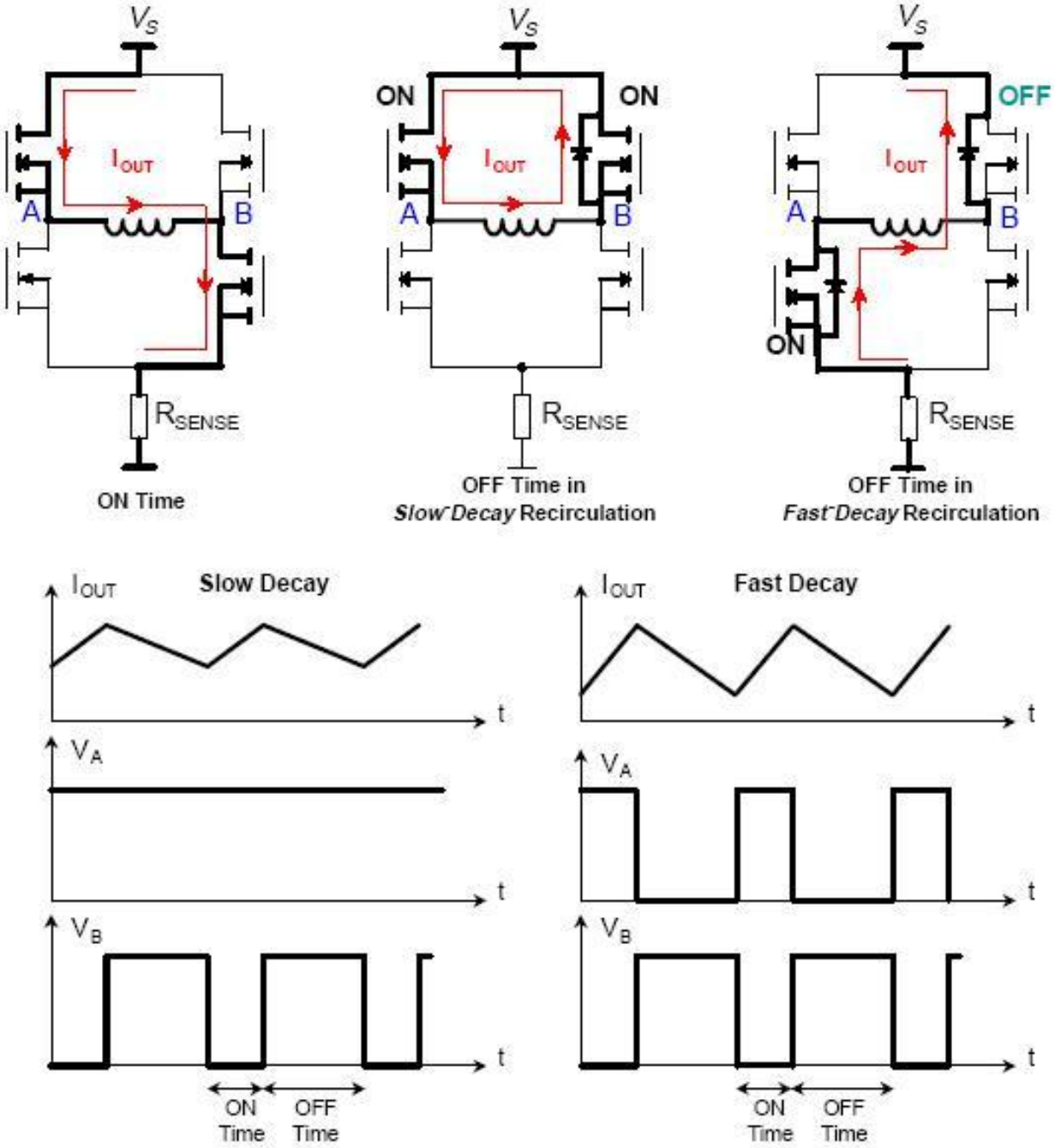


Main lib



Main lib

- **Main Lib 實作的功能**
  - **G code的解譯**
  - 系統參數的設定
  - 即時檢測是否有錯誤發生，並能立即停止
  - 顯示錯誤訊息


V_S
I_{OUT}
A
B
R_{SENSE}
ON Time
ON
ON
OFF Time in
Slow Decay Recirculation
OFF
ON
OFF Time in
Fast Decay Recirculation
Slow Decay
Fast Decay
V_A
V_B
t
ON Time
OFF Time

Marlin

main loop

3D Printer 韌體原始碼閱讀心得 (Ⅱ)

Outline

- 硬體相關參數 (#define) 的設計
- Marlin 的內部參數介紹 (一)
- Marlin 路徑規畫器的參數與資料結構介紹
- 主程式 (main loop) 解說
- G-code 解譯解說

硬體相關參數的設計

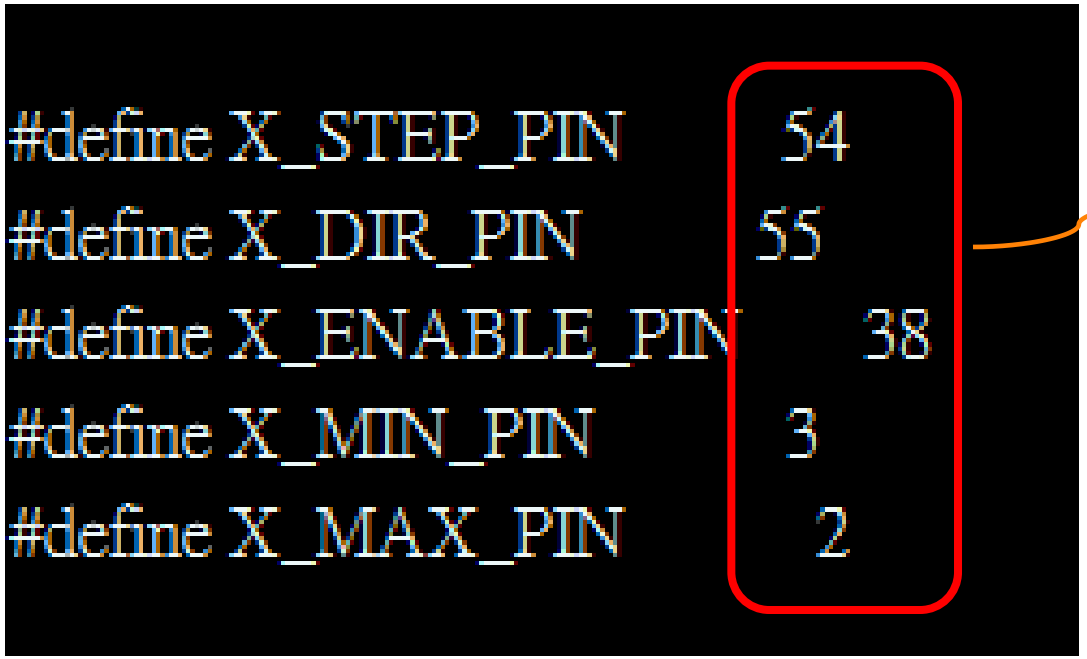
pin.h 硬體相關參數設定

- 在 pin.h 裡，marlin 定義了多種 ATmega 型號 3D printer 控制器的腳位對應
  - Marlin 在 Configuration.h 以 MOTHERBOARD 的參數決定控制器的型號
  - Marlin 支援的型號可以參考 Marlin overview 的投影片

```
#ifndef MOTHERBOARD
#define MOTHERBOARD 99
#endif
```

pin.h 硬體相關參數設定

- **XYZ 軸相關的腳位**
  - 值為 **-1** 表示該腳 **disable** 的狀態
  - 一個軸定義的腳位有五個



依 **3D printer** 控制器的型號設定的參數會不同

pin.h 硬體相關設定

- 擠出頭步進馬達相關的腳位
  - 最多支援 3 個擠出頭，預設是 1 個擠出頭
  - 值為 -1 表示該腳 disable 的狀態

```
// This defines the number of extruders
#define EXTRUDERS 1
```

擠出頭的個數
可在此更改

```
#define E0_STEP_PIN         28
#define E0_DIR_PIN          27
#define E0_ENABLE_PIN       24

#define E1_STEP_PIN         -1 // 19
#define E1_DIR_PIN          -1 // 18
#define E1_ENABLE_PIN       24

#define E2_STEP_PIN         -1 // 17
#define E2_DIR_PIN          -1 // 16
#define E2_ENABLE_PIN       24
```

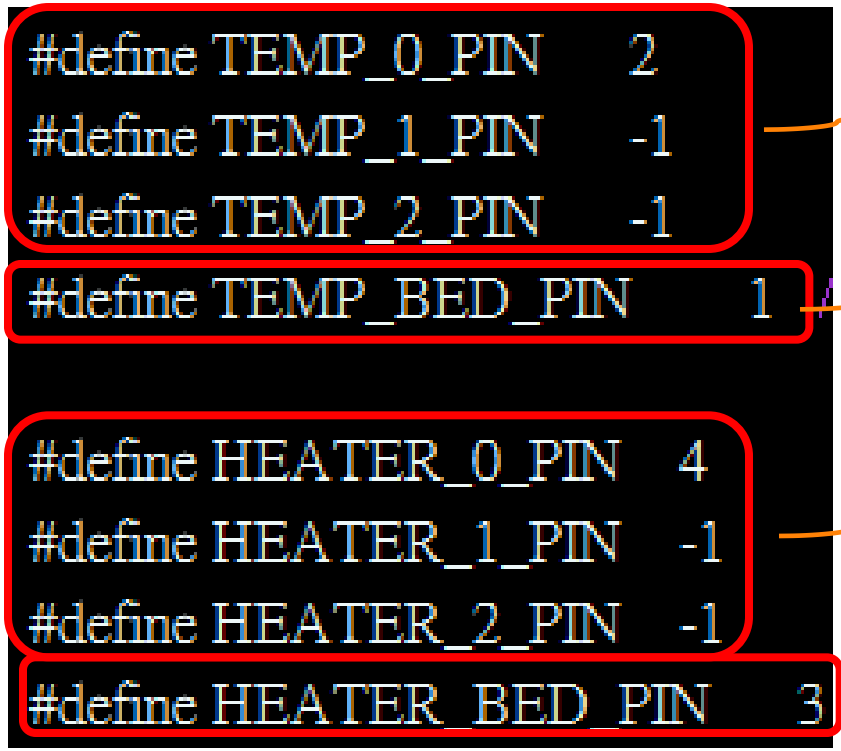
#1.

#2.

#3.

pin.h 硬體相關設定

- 擠出頭溫度控制相關的腳位
  - 依控制器的型號，決定加熱器的個數以及 analog 的個數
  - 定義的腳位的值為 -1 表示 disable 的狀態



pin.h 硬體相關設定

- 擠出頭 analog 腳位是定義在 Arduino 的 analog 腳位上，當與其他 digital 腳位定義相同時，並不會 mapping 到 Arduino 同一根腳

```
/****************************************************************************************
* Arduino Mega pin assignment
*
****************************************************************************************/
#if MOTHERBOARD == 3 || MOTHERBOARD == 33 || MOTHERBOARD == 34 || MOTHERBOARD == 77
```

```
#define SDPOWER            -1
#define SDSS               25//53
#define LED_PIN            13
```

```
#define TEMP_0_PIN         13  // ANALOG NUMBERING
#define TEMP_1_PIN         15  // ANALOG NUMBERING
#define TEMP_2_PIN         -1  // ANALOG NUMBERING
```

digital pin

analog pin

pin.h 硬體相關設定

- 電源供應器的設定
  - Marlin 支援 ATX power 或是 X-Box 360 203W
  - 預設是 ATX power

```
#ifndef POWER_SUPPLY
  #define POWER_SUPPLY 1
#endif
// 1 = ATX
#if (POWER_SUPPLY == 1)
  #define PS_ON_AWAKE  LOW
  #define PS_ON_ASLEEP HIGH
#endif
// 2 = X-Box 360 203W
#if (POWER_SUPPLY == 2)
  #define PS_ON_AWAKE  HIGH
  #define PS_ON_ASLEEP LOW
#endif
```

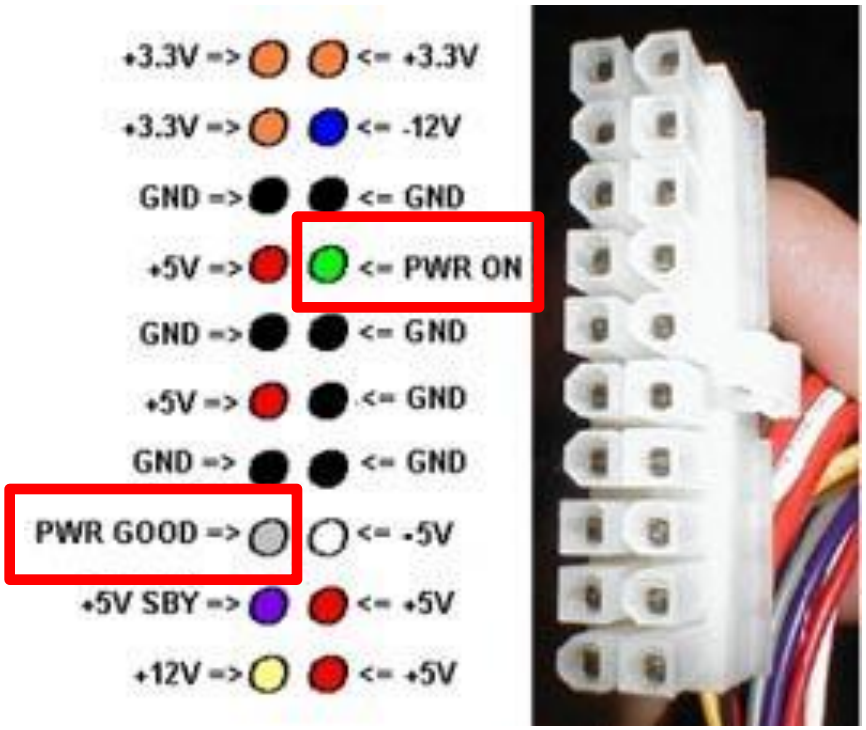
pin.h 硬體相關設定

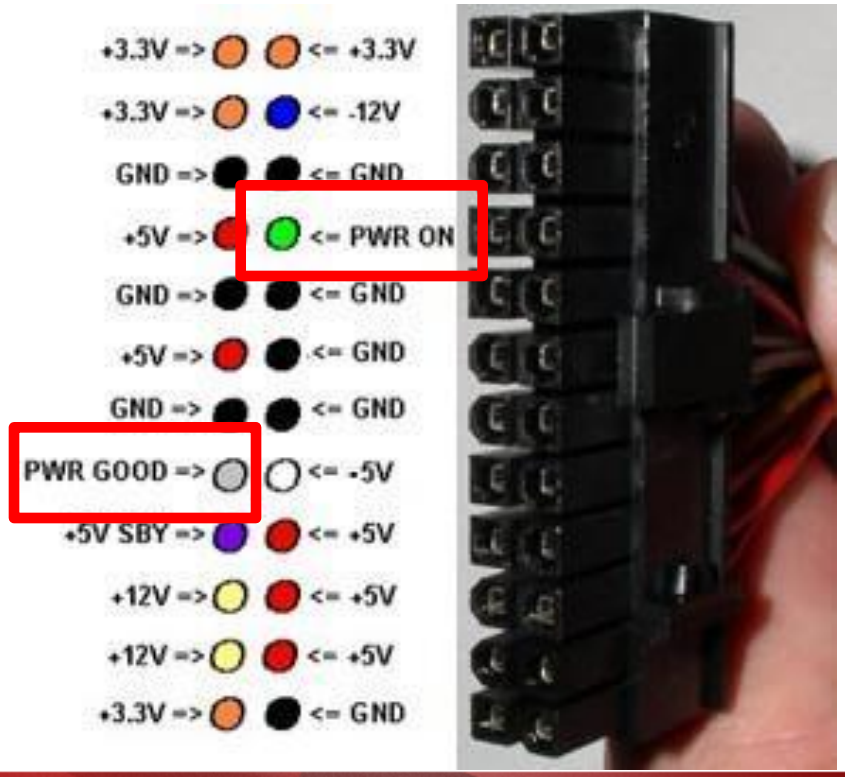
- 電源供應器 **PS_ON** 腳位
  - 用於 **power on ATX power** 或是 **X-Box 360 203W**
  - 定義的腳位的值為 **-1** 表示 **disable** 的狀態
  - 不是每個控制器都有定義 **PS_ON**

```
#define LED_PIN          -1

#define FAN_PIN          -1

#define PS_ON_PIN        14
#define KILL_PIN         -1
```

Marlin setup function

- 沒有定義 **PS_ON** 的做法
  - **ATX power 有 20 pin 與 24 pin 兩種 connecter**
  - **PWR ON 與 PWR GOOD (PWR OK) 要短路在一起**





pin.h 硬體相關設定

- 連接 Ultra LCD 相關的腳位

RAMPS pin	LCD PIN	Purpose
GND	1	
5V	2	
	3 (10K trimpot)	Contrast
D16	4	RS
GND	5	R/W
D17	6	Enable
	7	Data 1
	8	Data 2
	9	Data 3
23	10	Data 4
25	11	Data 5
27	12	Data 6
29	13	Data 7
	14	Data 8
	15	Backlight +
	16	Backlight -

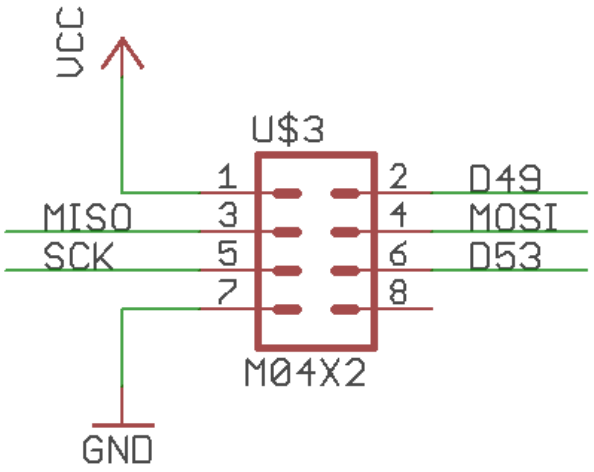
RS (Register Select)可以選擇 instruction mode or character mode

```
#define LCD_PINS_RS 16
#define LCD_PINS_ENABLE 17
#define LCD_PINS_D4 23
#define LCD_PINS_D5 25
#define LCD_PINS_D6 27
#define LCD_PINS_D7 29
```

對 LCD 讀寫資料

pin.h 硬體相關設定

- **SD card 腳位的設定**
  - **SD card 卡為 SPI 介面，當沒有使用 SD card 時，SPI 可以做為其他用途**
  - **依控制器的型號， SPI 的腳位會有不同**




```
#ifndef SDSUPPORT
// these pins are defined in the SD library if building with
  #define SCK_PIN         9
  #define MISO_PIN        11
  #define MOSI_PIN        10
#endif

#define SDPOWER           -1
#define SDSS              53
```

pin.h 硬體相關設定

- 其他硬體相關的腳位
  - 不是每個控制器都會有定義這些腳位

```
#define LED_PIN          -1
#define FAN_PIN          -1
#define KILL_PIN         -1
```

風扇的腳位

LED 的腳位

Kill pin 可以 disable 所有的步進馬達與加熱器

fastio.h 腳位mapping

- 在 pin.h 裡腳位定義的參數，是透過 fastio.h 的 macro，將參數 mapping 到控制器的暫存器

```
/// Read a pin wrapper
#define READ(IO)  _READ(IO)
/// Write to a pin wrapper
#define WRITE(IO, v)  _WRITE(IO, v)
/// set pin as input wrapper
#define SET_INPUT(IO)  _SET_INPUT(IO)
/// set pin as output wrapper
#define SET_OUTPUT(IO)  _SET_OUTPUT(IO)
/// check if pin is an input wrapper
#define GET_INPUT(IO)  _GET_INPUT(IO)
/// check if pin is an output wrapper
#define GET_OUTPUT(IO)  _GET_OUTPUT(IO)
/// check if pin is an timer wrapper
#define GET_TIMER(IO)  _GET_TIMER(IO)
```

pin.h 定義的參數

讀取該腳位的值

對該腳位做寫值

設定該腳位為 input 腳位

設定該腳位為 output 腳位

檢查該腳位是否為 input 腳位

檢查該腳位是否為 output 腳位

檢查該腳位是否為 timer 腳位

fastio.h 腳位 mapping

- **_SET_OUTPUT() 與 _SET_INTPUT() 實作**
  - 將 pin.h 的參數與控制器的暫存器連結起來做 I/O 的控制

控制器暫存器位址

設定該腳位為 output 狀態

設定該腳位為 input 狀態

```
/// set pin as input
#define    _SET_INPUT(IO) do {DIO ## IO ## _DDR &= ~MASK(DIO ## IO ## _PIN); } while (0)
/// set pin as output
#define    _SET_OUTPUT(IO) do {DIO ## IO ## _DDR |= MASK(DIO ## IO ## _PIN); } while (0)
```

fastio.h 腳位 mapping

- 在 fastio.h 依控制器型號定義各個暫存器位址

```
#define DIO10_PIN      PINB2
#define DIO10_DDR      DDRB
```

暫存器腳位的位址， **SET_OUTPUT() / SET_INTPUT()** 設定的地方

```
#define DIO10_RPORT PINB
#define DIO10_WPORT    PORTB
```

該腳位做 I/O 讀寫的位址
PORTB- The Port B Data Register - read/write
PINB- The Port B Input Pins Register - read only

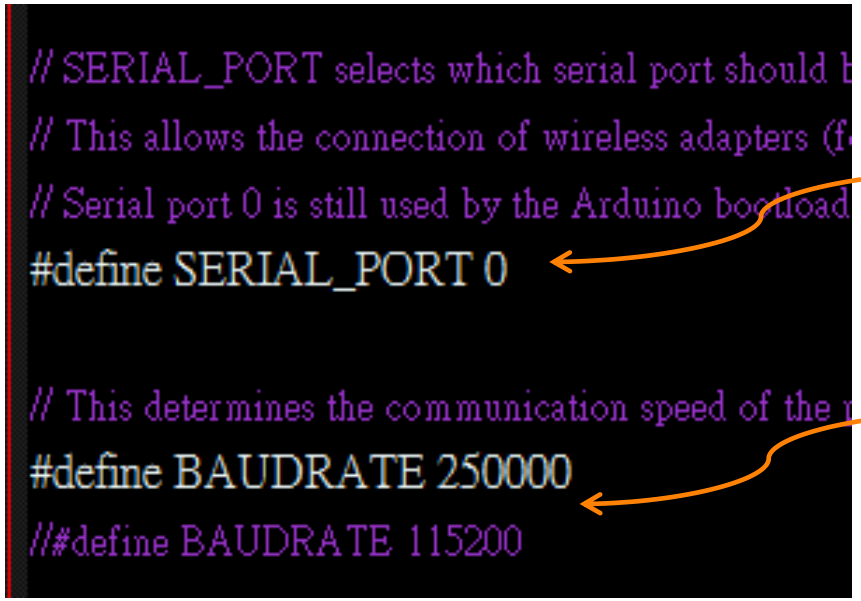
```
#define DIO10_PWM      NULL
```

該腳位設定 **PWM** 的位址，**NULL** 表示該腳位沒有 **PWM** 功能

Marlin 內部參數介紹

Marlin 內部參數

- **Serial port 參數 (Configurature.h)**



Ardiuno bootloader 使用的 serial port

設定 serial port baudrate，最高速度為 250K bps

Marlin 內部參數

- 設定擠出頭及加熱板的溫度 sensor 型號 (Configurature.h)
  - #define TEMP_SENSOR_0 -2
    - -2 表示該擠出頭使用熱電偶與 MAX6675 轉接板
  - #define TEMP_SENSOR_1 -1
    - -1 表示該擠出頭使用熱電偶與 AD595 轉接板
  - #define TEMP_SENSOR_2 0
    - 0 表示沒有使用該擠出頭
  - #define TEMP_SENSOR_BED 1
    - 大於 1 表示該擠出頭使用熱敏電阻，不同的參數表示不同型號的熱敏電阻

```
//// Temperature sensor settings:
//  -2 is thermocouple with MAX6675 (only for sensor 0)
//  -1 is thermocouple with AD595
//  0 is not used
//  1 is 100k thermistor - best choice for EPCOS 100k (4.7k pullup)
//  2 is 200k thermistor - ATC Semitec 204GT-2 (4.7k pullup)
//  3 is mendel-parts thermistor (4.7k pullup)
//  4 is 10k thermistor !! do not use it for a hotend.
//    It gives bad resolution at high temp. !!
//  5 is 100K thermistor - ATC Semitec 104GT-2 (Used in ParCan) (4.7k pullup)
//  6 is 100k EPCOS - Not as accurate as table 1
//    (created using a fluke thermocouple) (4.7k pullup)
//  7 is 100k Honeywell thermistor 135-104LAG-J01 (4.7k pullup)
//  8 is 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup)
//  9 is 100k GE Sensing AL03006-58.2K-97-G1 (4.7k pullup)
//  10 is 100k RS thermistor 198-961 (4.7k pullup)
//
//    1k ohm pullup tables - This is not normal,
//    you would have to have changed out your 4.7k for 1k
//    (but gives greater accuracy and more stable PID)
//  51 is 100k thermistor - EPCOS (1k pullup)
//  52 is 200k thermistor - ATC Semitec 204GT-2 (1k pullup)
//  55 is 100k thermistor - ATC Semitec 104GT-2 (Used in ParCan) (1k pullup)
```

Marlin 內部參數

- 加熱溫度範圍的限制
  - 最低溫限制，預設為 5 度
  - 溫度低於限制，marlin 會中斷所有的工作並傳送錯誤訊息

```
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
```

各個擠出頭的低溫限制

```
#define BED_MINTEMP 5
```

加熱板的低溫限制

Marlin 內部參數

- 加熱溫度範圍的限制
  - 最高溫限制，擠出頭預設為 **275** 度, 加熱板為 **150** 度
  - 溫度高於限制，**marlin** 會中斷所有的工作並傳送錯誤訊息

```
#define HEATER_0_MAXTEMP 275
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
```

各個擠出頭的高溫限制

```
#define BED_MAXTEMP 150
```

加熱板的高溫限制

Marlin 內部參數

- 定義 3D printer 工作空間的參數

```
#define X_MAX_POS 205
#define X_MIN_POS 0
#define Y_MAX_POS 205
#define Y_MIN_POS 0
#define Z_MAX_POS 200
#define Z_MIN_POS 0
```

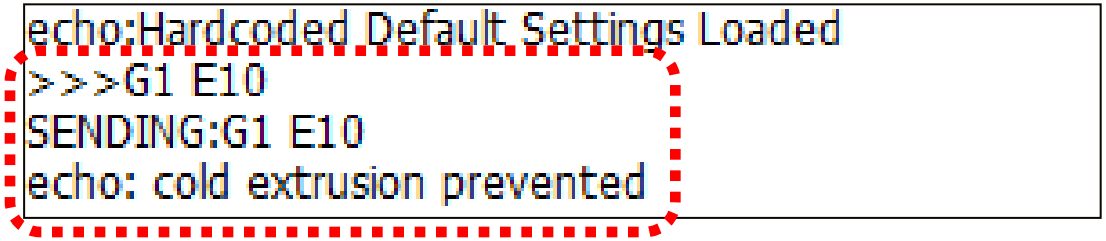
X, Y, Z 工作位置的設定，單位為 millimeter

X, Y, Z 最大的位移

```
#define X_MAX_LENGTH (X_MAX_POS - X_MIN_POS)
#define Y_MAX_LENGTH (Y_MAX_POS - Y_MIN_POS)
#define Z_MAX_LENGTH (Z_MAX_POS - Z_MIN_POS)
```

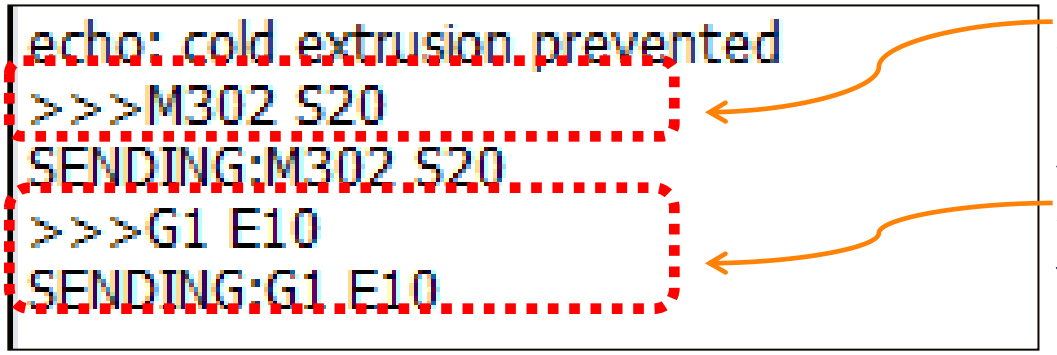
Marlin 內部參數

- 擠出頭的步進馬達最低可工作溫度的參數
  - **#define EXTRUDE_MINTEMP 170**
  - 當擠出頭的溫度低於 **EXTRUDE_MINTEMP** 時，**marlin** 會禁止擠出頭的步進馬達運作，同時會得到如下列的錯誤訊息



Marlin 內部參數

- 動態更改擠出頭步進馬達的最低可工作溫度
  - **Marlin 修改的 g code 為 M302 S(新的最低可工作溫度)**

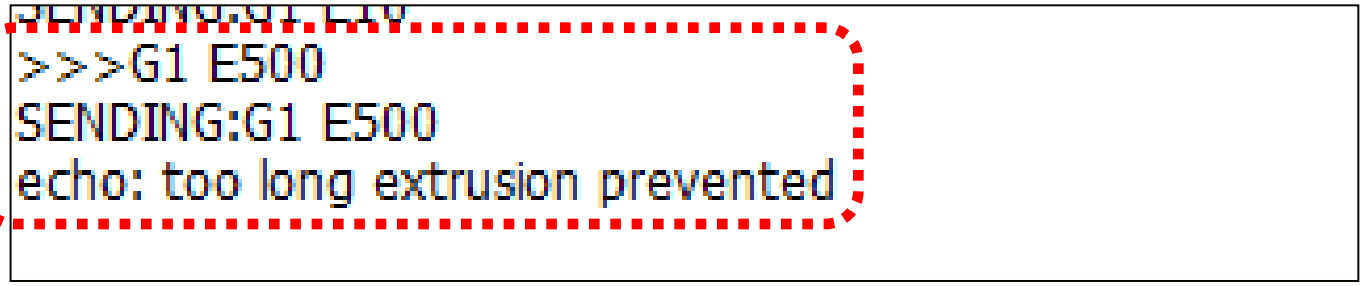


更改最低可工作溫度

更改後，再次下 **G1** 指令不會有錯誤訊息

Marlin 內部參數

- 擠出頭單一指令擠料的長度限制
  - **#define EXTRUDE_MAXLENGTH**
    (X_MAX_LENGTH+Y_MAX_LENGTH)
  - 此限制沒有提供 **G code** 做更改



Marlin 內部參數

- 步進馬達 DIR 設定
  - 設 true 表示， DIR 為 high 時該軸會向 MIN 的方向移動
  - 設 false 表示，DIR 為 high 時該軸會向 MAX 的方向移動
  - 如果步進馬達移動的方向跟你要的不一樣，可以改這裡的參數，或者直接調整馬達的訊號線

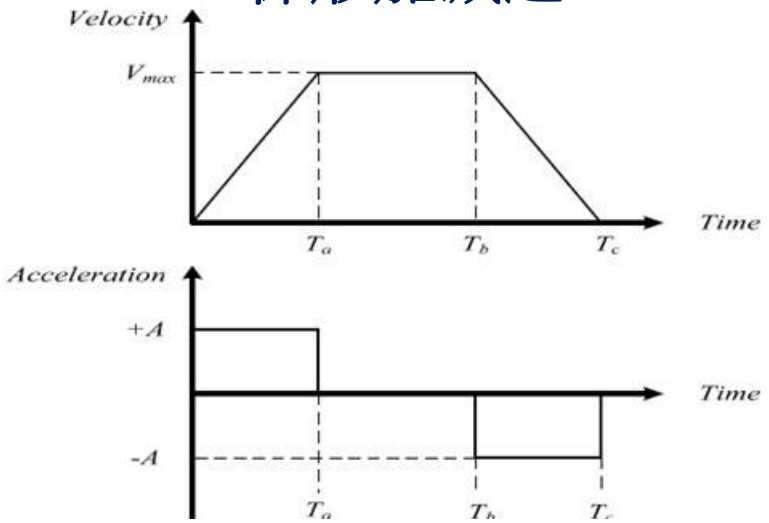
```
#define INVERT_X_DIR true    // for Mendel set to false, for Orca set to tru
#define INVERT_Y_DIR false   // for Mendel set to true, for Orca set to fal
#define INVERT_Z_DIR true    // for Mendel set to false, for Orca set to tru
#define INVERT_E0_DIR false  // for direct drive extruder v9 set to true, fo
#define INVERT_E1_DIR false  // for direct drive extruder v9 set to true, f
#define INVERT_E2_DIR false  // for direct drive extruder v9 set to true, fo
```

Marlin 路徑規畫器的參數與資料結構介紹

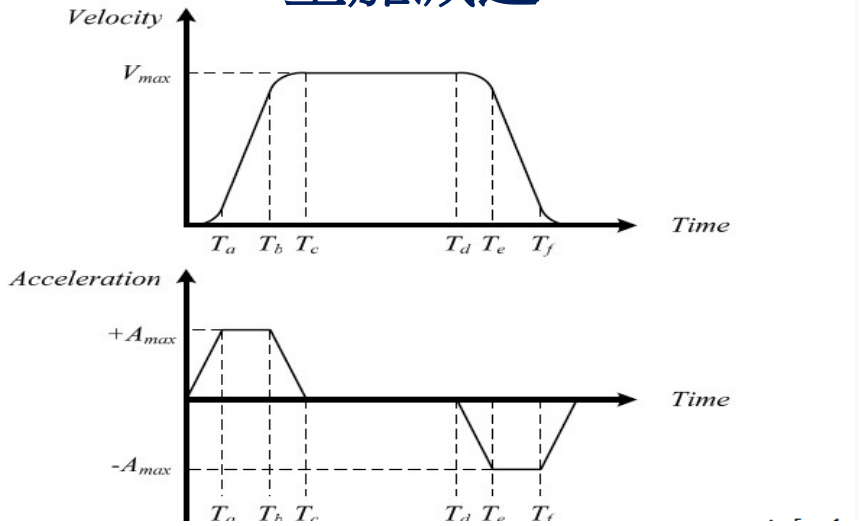
Marlin 路徑規畫器的參數

- 使用步進馬達時，通常會以梯形加減速或是 S 型加減速做速度的規畫
- **Marlin** 是使用梯形加減速

梯形加減速

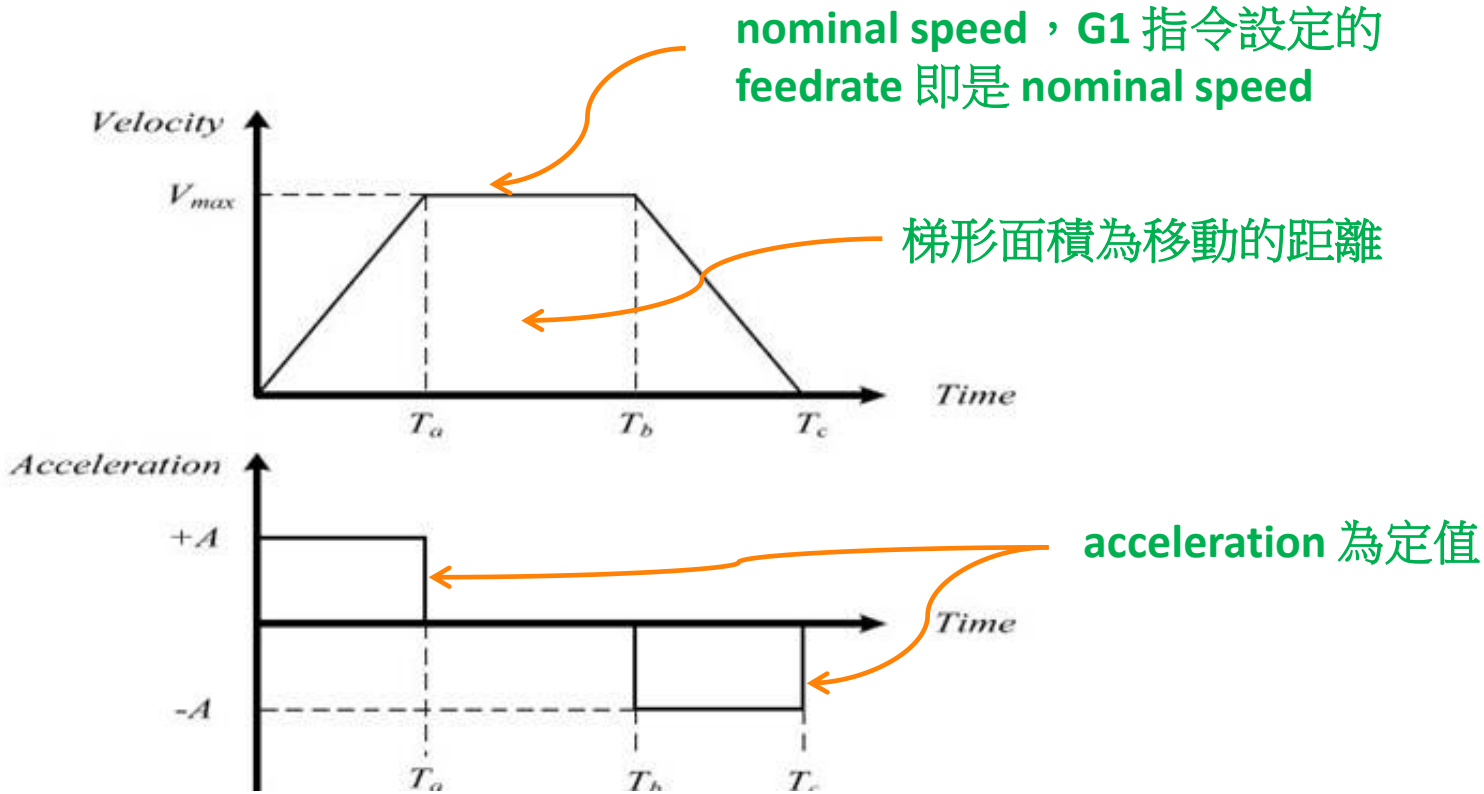


S 型加減速



Marlin 路徑規畫器的參數

- 梯形加減速的參數



Marlin 路徑規畫器的參數

- 路徑規畫的參數設定
  - **DEFAULT_AXIS_STEPS_PER_UNIT** 移動 **1 mm** 的 **step** 數
  - **DEFAULT_MAX_FEEDRATE** 可容許的最大 **nominal speed**
  - **DEFAULT_MAX_ACCELERATION** 可容許的最大加速度
  - **DEFAULT_ACCELERATION** 預設的加速度值
  - **DEFAULT_RETRACT_ACCELERATION** 擠出頭回捲料時用的加速度值

```
#define DEFAULT_AXIS_STEPS_PER_UNIT  {78.7402,78.7402,200.0*8/3,760*1.1}
#define DEFAULT_MAX_FEEDRATE         {500, 500, 5, 25}    // (mm/sec)
#define DEFAULT_MAX_ACCELERATION     {9000,9000,100,10000}    //X, Y, Z, E ma

#define DEFAULT_ACCELERATION         3000    //X, Y, Z and E max acceleration in mm/s^2 fo
#define DEFAULT_RETRACT_ACCELERATION 3000    //X, Y, Z and E max acceleration in mm
```

Marlin 路徑規畫器的參數

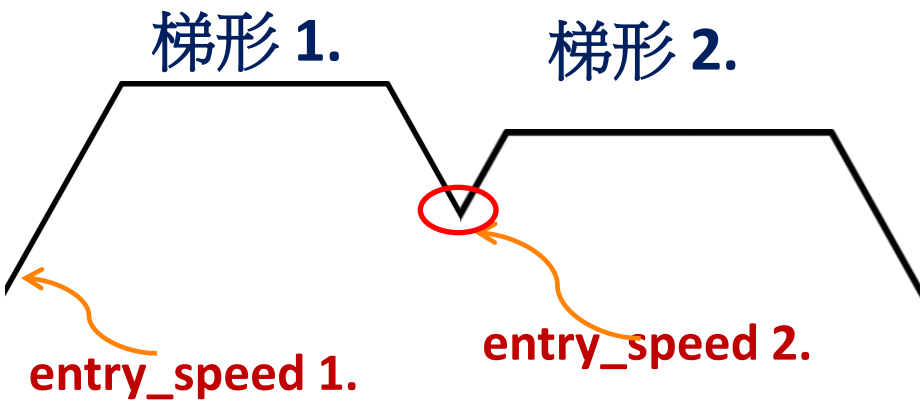
- 路徑規畫初速度的設定
  - X, Y, Z, E 最大的初速度，單位為 mm/sec
  - 規畫兩條路徑的連接速度時，會對初速度做計算

```
#define DEFAULT_XYJERK          20.0    // (mm/sec)
#define DEFAULT_ZJERK           0.4     // (mm/sec)
#define DEFAULT_EJERK           5.0     // (mm/sec)
```

X,Y 軸最大初速度

Z 軸最大初速度

E 軸最大初速度



Marlin 路徑規畫器的資料結構

- 直線軌跡規畫 器的資料結構 **block_t**

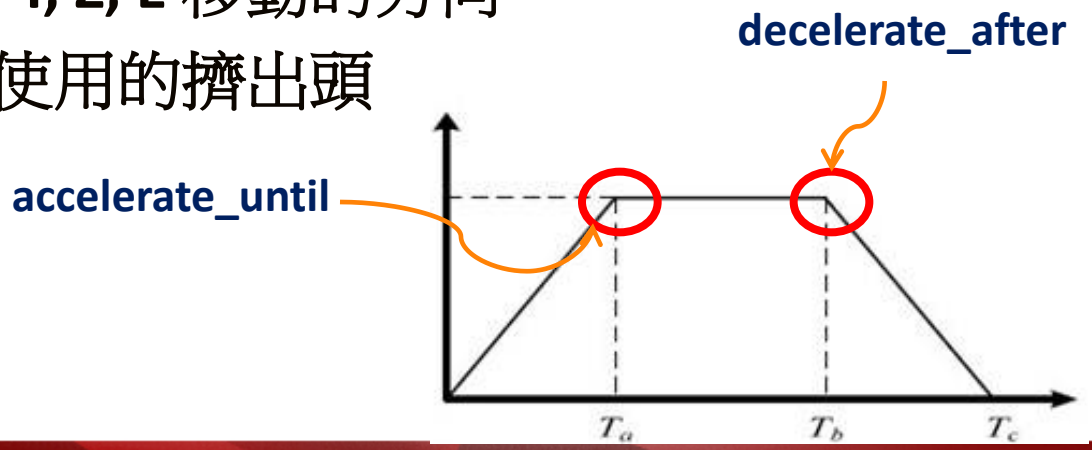
Marlin 路徑規畫器的資料結構

1. 直線軌跡設定的參數

```
typedef struct {
  // Fields used by the bresenham algorithm for tracing the line
  long steps_x, steps_y, steps_z, steps_e;  // Step count along each axis
  unsigned long step_event_count;           // The number of step events required to co
  long accelerate_until;                    // The index of the step event on which to stop a
  long decelerate_after;                    // The index of the step event on which to start d
  long acceleration_rate;                   // The acceleration rate used for acceleration cal
  unsigned char direction_bits;             // The direction bit set for this block (refers to
  unsigned char active_extruder;            // Selects the active extruder
```

Marlin 路徑規畫器的資料結構

1. 直線軌跡設定的參數
   - **steps_x, steps_y, steps_z, steps_e** 表示各軸步進馬達的 **step** 數
   - **step_event_count** 此 **block_t** 最大的 **step** 數
   - **accelerate_until** 加速的 **step** 數
   - **decelerate_after** 開始減速的 **step**
   - **acceleration_rate** 記錄目前的加速度
   - **direction_bits** 記錄 **X, Y, Z, E** 移動的方向
   - **active_extruder** 目前使用的擠出頭



Marlin 路徑規畫器的資料結構

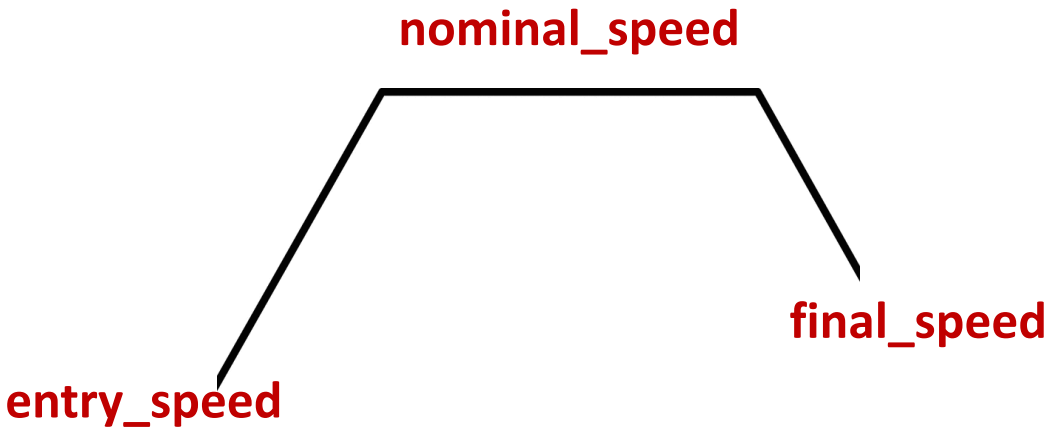
2. 軌跡規畫初始設定的參數

```
float nominal_speed;                // The nominal speed for this block in mm/sec
float entry_speed;                  // Entry speed at previous-current junction in mm
float max_entry_speed;              // Maximum allowable junction entry speed in
float millimeters;                  // The total travel of this block in mm
float acceleration;                 // acceleration mm/sec^2
unsigned char recalculate_flag;     // Planner flag to recalculate trapezoids on en
unsigned char nominal_length_flag;  // Planner flag for nominal speed always
```

Marlin 路徑規畫器的資料結構

2. 軌跡規畫初始設定的參數

- nominal_speed, entry_speed , max_entry_speed



- millimeters 此 block 移動的距離
- acceleration 此 block 的加速度 (mm/sec^2)
- recalculate_flag 記錄是否重新規畫連接速度的 flag
- nominal_length_flag 記錄 nominal_speed 是否能達到的 flag

Marlin 路徑規畫器的資料結構

3. 梯形加減速的參數

```
// Settings for the trapezoid generator
unsigned long nominal_rate;                   // The nominal step rate for this blo
unsigned long initial_rate;                  // The jerk-adjusted step rate at start of
unsigned long final_rate;                    // The minimal rate at exit
unsigned long acceleration_st;                // acceleration steps/sec^2
```

3. 其他參數

```
  unsigned long fan_speed;
  volatile char busy;
} block_t;
```

Marlin 資料結構

3. 規畫梯形加減速的參數
   - nominal_rate (step/sec)
   - initial_rate (step/sec)
   - final_rate (step/sec)
   - acceleration_st (step/sec^2)
4. 其他參數
   - fan_speed 設定風扇的轉速
   - busy 記錄 block 是否被執行

Marlin

路徑規畫

的實作

我們將在

下(ㄒㄨˋ)次(ㄐㄧˋ)

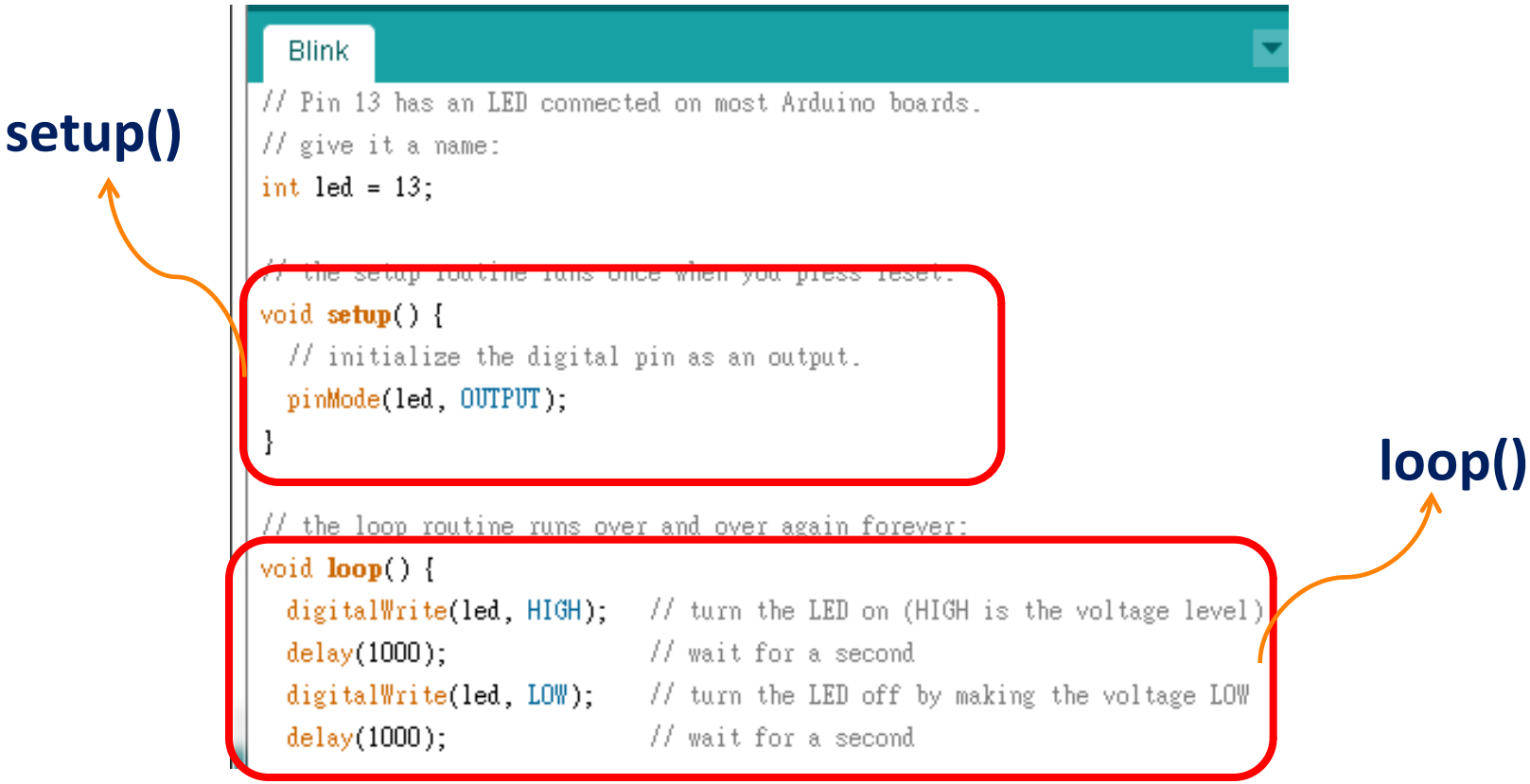
讀書會

講解

Marlin 主程式架構
Main loop 解說

Marlin main loop

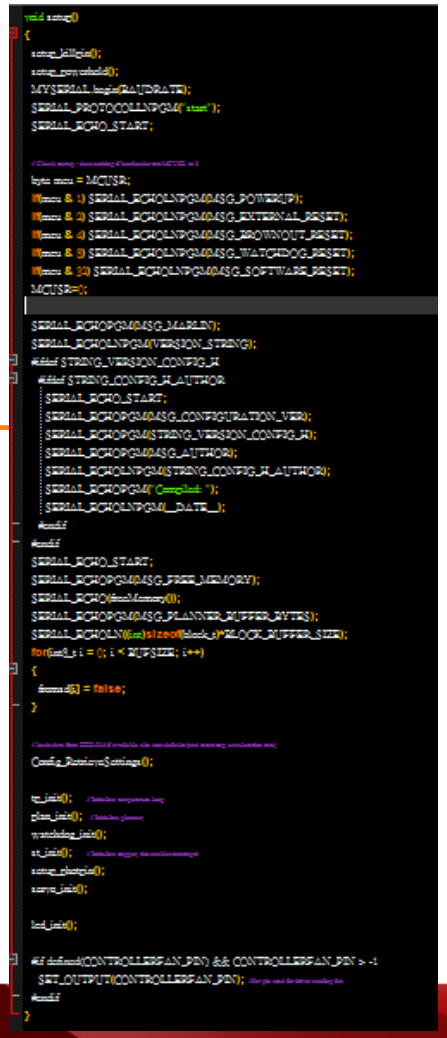
- **Arduino 的主程式一般分成兩個 function**
  - **Setup() 為初始化的動作，Loop() 為執行的主程式**



Marlin main loop

- **Marlin 的主程式亦不例外**

setup()



loop()

Marlin main loop

- 開始講解 **setup ()**

```
void setup()
{
  setup_killpin();
  setup_powerhold();
  MYSERIAL.begin(BAUDRATE);
  SERIAL_PROTOCOLLNPGM("start");
  SERIAL_ECHO_START;

  // Check startup - does nothing if bootloader sets MCUSR to 0
  byte mcu = MCUSR;
  if(mcu & 1) SERIAL_ECHOLNPGM(MSG_POWERUP);
  if(mcu & 2) SERIAL_ECHOLNPGM(MSG_EXTERNAL_RESET);
  if(mcu & 4) SERIAL_ECHOLNPGM(MSG_BROWNOUT_RESET);
  if(mcu & 8) SERIAL_ECHOLNPGM(MSG_WATCHDOG_RESET);
  if(mcu & 32) SERIAL_ECHOLNPGM(MSG_SOFTWARE_RESET);
```

Marlin setup function

- **setup function 的講解**

```
void setup()
{
  setup_killpin();
  setup_powerhold();
  MYSERIAL.begin(BAUDRATE);
  SERIAL_PROTOCOLLNPGM("start");
  SERIAL_ECHO_START;

  // Check startup - does nothing if bootloader sets MCUS
  byte mcu = MCUSR;
  if(mcu & 1) SERIAL_ECHOLNPGM(MSG_PO
  if(mcu & 2) SERIAL_ECHOLNPGM(MSG_EX
```

setup_killpin()

Kill pin 設 high 時會 disable marlin 各個功能，包括步進馬達、加熱器

setup_powerhold()

在有定義 PS_ON 情況下，會依 ATX power 或是 X-Box 360 203W 設定 PS_ON

Marlin setup function

- **setup function 的講解**

```
SERIAL_PROTOCOLLNPGM("start");
SERIAL_ECHO_START;

// Check startup - does nothing if bootloader sets MCUSR to 0
byte mcu = MCUSR;
if(mcu & 1) SERIAL_ECHOLNPGM(MSG_POWERUP);
if(mcu & 2) SERIAL_ECHOLNPGM(MSG_EXTERNAL_RESET);
if(mcu & 4) SERIAL_ECHOLNPGM(MSG_BROWNOUT_RESET);
if(mcu & 8) SERIAL_ECHOLNPGM(MSG_WATCHDOG_RESET);
if(mcu & 32) SERIAL_ECHOLNPGM(MSG_SOFTWARE_RESET);
MCUSR=0;
```

檢查 **startup** 是否成功
檢查 **reset flag MCUSR** 的值

MCUSR = 0 正常
MCUSR = 1 Power-on Reset
MCUSR = 2 External Reset
MCUSR = 4 Brown-Out Reset
MCUSR = 8 Watchdog Reset
MCUSR = 32 JTAG Reset

Marlin setup function

- **setup function 的講解**

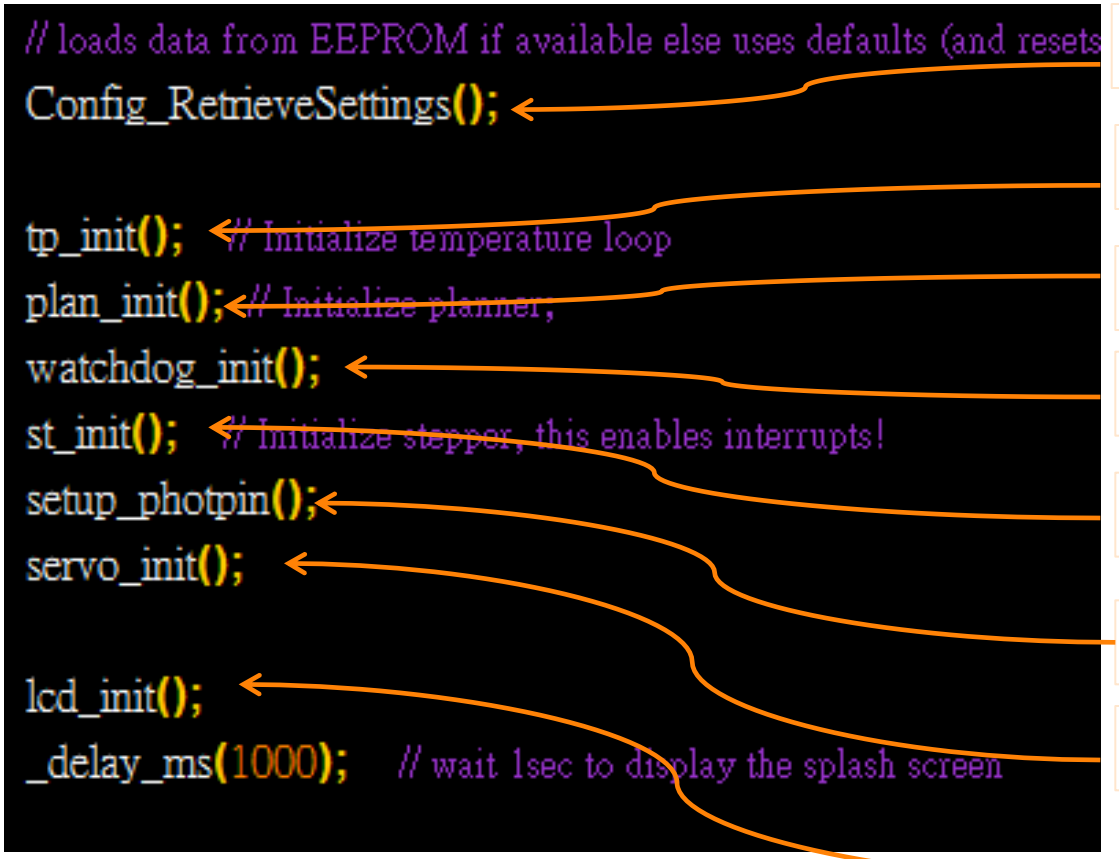
```
#ifdef STRING_VERSION_CONFIG_H
  #ifdef STRING_CONFIG_H_AUTHOR
    SERIAL_ECHO_START;
    SERIAL_ECHOPGM(MSG_CONFIGURATION_VER);
    SERIAL_ECHOPGM(STRING_VERSION_CONFIG_H);
    SERIAL_ECHOPGM(MSG_AUTHOR);
    SERIAL_ECHOLNPGM(STRING_CONFIG_H_AUTHOR);
    SERIAL_ECHOPGM("Compiled: ");
    SERIAL_ECHOLNPGM(__DATE__);
  #endif
#endif
SERIAL_ECHO_START;
SERIAL_ECHOPGM(MSG_FREE_MEMORY);
SERIAL_ECHO(freeMemory());
SERIAL_ECHOPGM(MSG_PLANNER_BUFFER_BYTES);
SERIAL_ECHOLN((int)sizeof(block_t)*BLOCK_BUFFER_SIZE);
```

Marlin 的版本訊息

系統剩下多少 **memory**，及 **block** 的 **size**

Marlin setup function

- **setup function 的講解**



從 **EEPROM** 載入系統參數

溫度設定的初始化

路徑規畫的初始化

watchdog 的初始化

步進馬達的初始化

下一頁 相机触发功能初始化

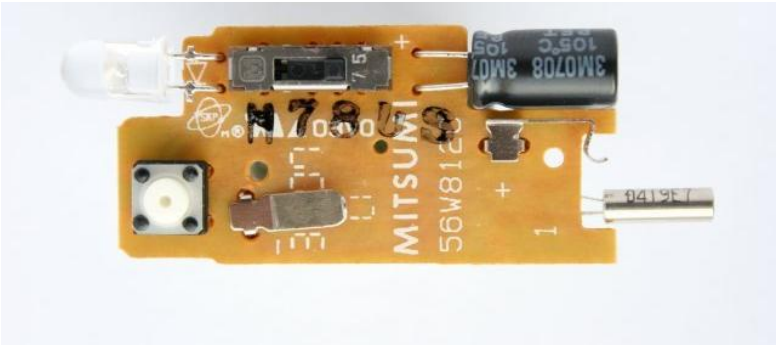
RC servo 的初始化

LCD 的初始化

Marlin setup function

6. setup_photpin()

– **Triggers a camera by emulating a Canon RC-1 Remote**



Marlin setup function

- **setup function 的講解**

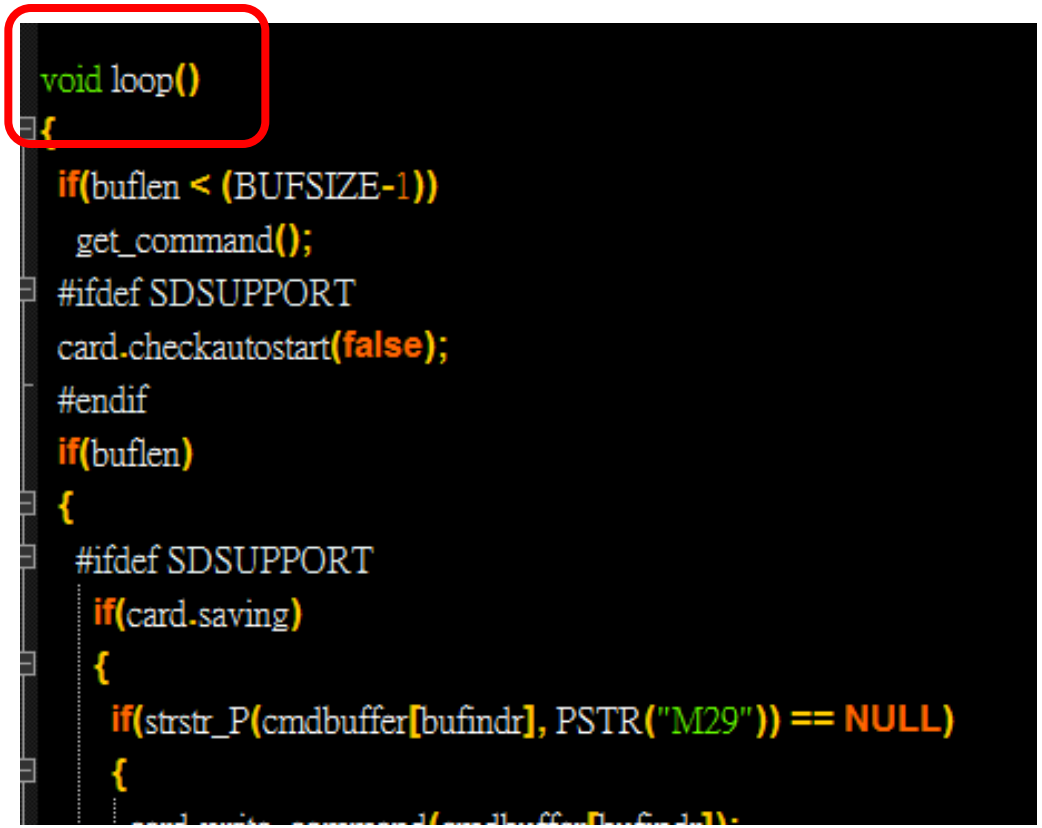
```
servo_init();

lcd_init();
_delay_ms(1000);    // wait 1sec to display the splash screen

#if defined(CONTROLLERFAN_PIN) && CONTROLLERFAN_PIN > -1
  SET_OUTPUT(CONTROLLERFAN_PIN); //Set pin used for driver cooling fan
#endif
}
```

設定 **CONTROLLERFAN_PIN**

- **setup function 結束**

Marlin loop function

- 開始 loop function 的講解



```
void loop()
{
 if(buflen < (BUFSIZE-1))
  get_command();
 #ifdef SDSUPPORT
 card.checkautostart(false);
 #endif
 if(buflen)
 {
  #ifdef SDSUPPORT
   if(card.saving)
   {
    if(strstr_P(cmdbuffer[bufindr], PSTR("M29")) == NULL)
    {
```

Marlin loop function

- **loop function 的講解**

```
void loop()
{
  if(buflen < (BUFSIZE-1))
    get_command();
  #ifdef SDSUPPORT
  card.checkautostart(false);
  #endif
  if(buflen)
```

讀取 G code 的 buffer 為 cmdbuffer[BUFSIZE][MAX_CMD_SIZE]，BUFSIZE 為 4, 最多存 4 條指令。MAX_CMD_SIZE 為 96, 一條指令最多存 96 個字元 。

如果 cmdbuffer 有空間，則讀取 G code 指令

Marlin loop function

- **loop function 的講解**

```
#ifdef SDSUPPORT
  if(card.saving)
  {
    if(strstr_P(cmdbuffer[bufindr], PSTR("M29")) == NULL)
    {
      card.write_command(cmdbuffer[bufindr]);
      if(card.logging)
      {
        process_commands();
      }
      else
      {
        SERIAL_PROTOCOLLNPGM(MSG_OK);
      }
    }
    else
    {
      card.closefile();
      SERIAL_PROTOCOLLNPGM(MSG_FILE_SAVED);
    }
  }
```

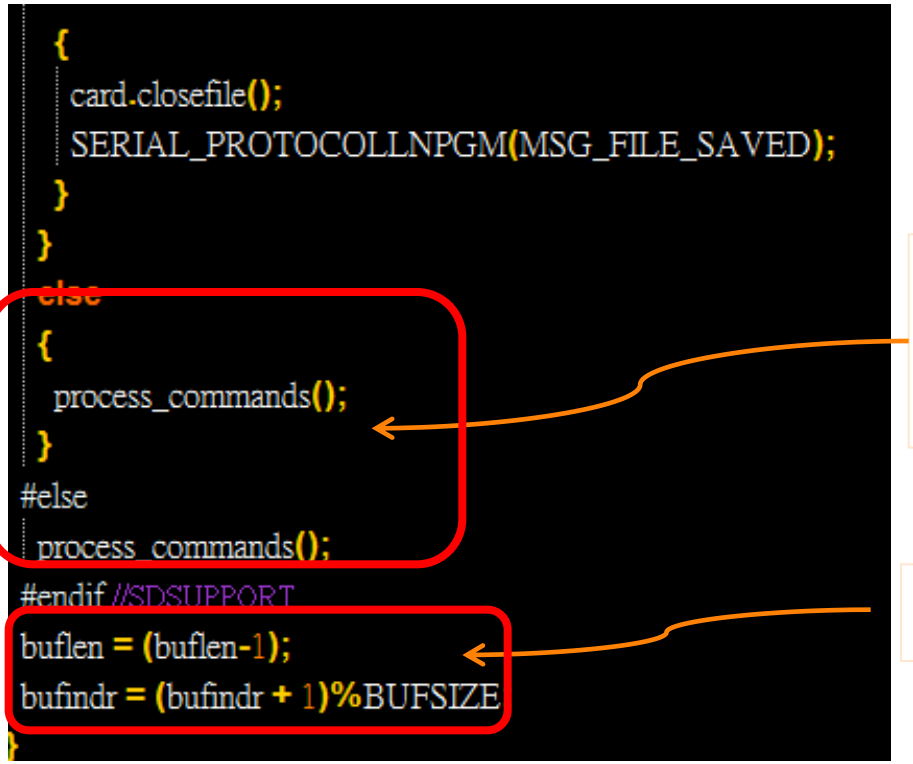
當 SD 卡開啟檔案作寫入 (M 28) 時，card.saving 會設 true，會將 cmdbuffer 的內容複 製到 SD 卡

如果 cmdbuffer 讀取到 M29 的指令 ，則結束複 製動作 。

解譯並執行 g code 的動作

Marlin loop function

- **loop function 的講解**

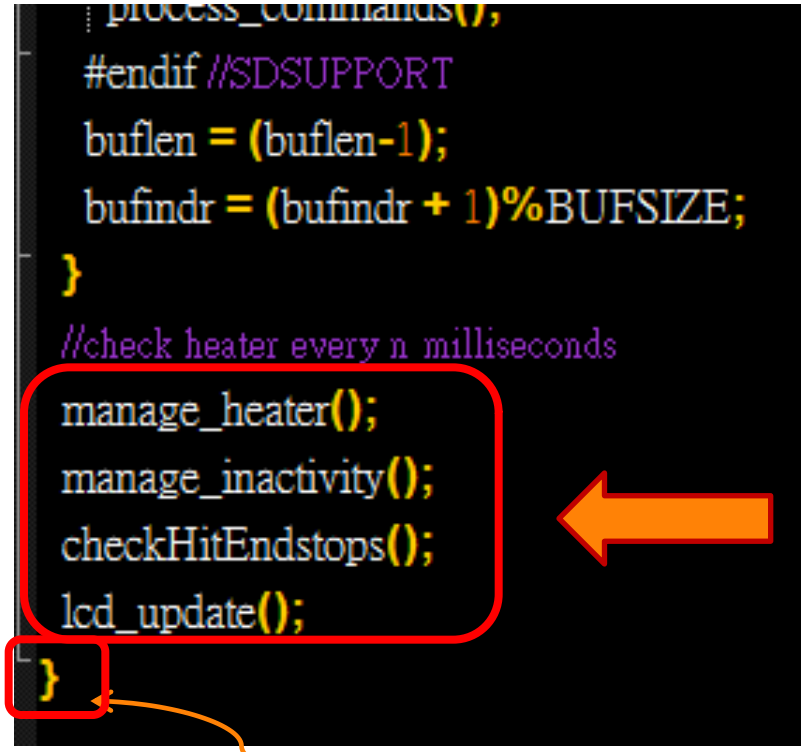


card.saving 為 false 時或是沒有定義 SDSUPPORT時，則做解譯並執行 g code 的動作

載入 **cmdbuffer** 下一個 的指令

Marlin loop function

- **loop function 的講解**



1. manage_heater() 為加熱溫度的控制(*)
2. manage_inactivity() 檢查系統否有異常的狀況(*)
3. checkHitEndstops() 檢查 endstop 的狀態
4. 更新 LCD 的訊息

- **loop function 結束**

Marlin loop function

1. 開始講解 manage_heater()

```
void manage_heater()
{
  float pid_input;
  float pid_output;

  if(temp_meas_ready != true)   //better readability
    return;

  updateTemperaturesFromRawValues();

  for(int e = 0; e < EXTRUDERS; e++)
  {
```

Marlin loop function

1. manage_heater() 的講解

```
void manage_heater()
{
  float pid_input;
  float pid_output;

  if(temp_meas_ready != true)   //better readability
    return;

  updateTemperaturesFromRawValues();

  for(int e = 0; e < EXTRUDERS; e++)
  {
```

檢查是否需要做更新動作，否則結束

- 依據進中斷的累積數次是否達128 次 (128ms)，決定是否要更新目前的溫度。
- 在中斷中，累積數次達128 次 (128ms)，會更新 soft_pwm

更新每個擠出頭及加熱板目前的溫度，由 analog 值轉成溫度

Marlin loop function

1. manage_heater() 的講解

```
for(int e = 0; e < EXTRUDERS; e++)
{

#ifdef PIDTEMP
  pid_input = current_temperature[e];

  #ifndef PID_OPENLOOP
    pid_error[e] = target_temperature[e] - pid_input;
    if(pid_error[e] > PID_FUNCTIONAL_RANGE) {
      pid_output = BANG_MAX;
      pid_reset[e] = true;
    }
    else if(pid_error[e] < -PID_FUNCTIONAL_RANGE || target_temperature[e] == 0) {
      pid_output = 0;
      pid_reset[e] = true;
    }
    else {
      if(pid_reset[e] == true) {
        temp_iState[e] = 0.0;
        pid_reset[e] = false;
      }
      pTerm[e] = Kp * pid_error[e];
      temp_iState[e] += pid_error[e];
      temp_iState[e] = constrain(temp_iState[e], temp_iState_min[e], temp_iState_max[e]);
```

對每個擠出頭及加熱板做 **soft_pwm** 的計算

- 計算每個擠出頭及加熱板需要加熱的時間
- 此計算採用 **PID** 控制 **(*)**

Marlin loop function

1. manage_heater() 的講解

- soft_pwm 的計算方法

```
Kp = 22.2,  Ki = 0.14155776,  Kd = 869.7509765625,  K1 = 0.95,  K2 = 0.05
pid_input = current_temperature[e]
pid_error[e] = target_temperature[e] - pid_input
pTerm[e] = Kp * pid_error[e]
temp_iState[e] += pid_error[e]
temp_iState[e] = constrain(temp_iState[e], temp_iState_min[e],
    temp_iState_max[e])
iTerm[e] = Ki * temp_iState[e]
dTerm[e] = (Kd * (pid_input - temp_dState[e]))*K2 + (K1 * dTerm[e])
temp_dState[e] = pid_input
pid_output = constrain(pTerm[e] + iTerm[e] - dTerm[e], 0, PID_MAX)
```

取得目前溫度

計算 P 項

計算 I 項

計算 D 項

更新 soft_pwm

Marlin loop function

2. 開始講解 manage_inactivity()

```
void manage_inactivity()
{
  if( (millis() - previous_millis_cmd) >  max_inactive_time )
    if(max_inactive_time)
      kill();
  if(stepper_inactive_time)  {
    if( (millis() - previous_millis_cmd) >  stepper_inactive_time )
    {
      if(blocks_queued() == false) {
        disable_x();
        disable_y();
        disable_z();
        disable_e0();
        disable_e1();
        disable_e2();
      }
    }
```

Marlin loop function

2. manage_inactivity() 的講解

```
void manage_inactivity()
{
 if( (millis() - previous_millis_cmd) >  max_inactive_time )
  if(max_inactive_time)
   kill();
 if(stepper_inactive_time)  {
  if( (millis() - previous_millis_cmd) >  stepper_inactive_time )
  {
   if(blocks_queued() == false) {
    disable_x();
    disable_y();
    disable_z();
    disable_e0();
    disable_e1();
    disable_e2();
   }
  }
```

如果兩個 g code 執行間隔的時間> max_inactive_time 會中止所有的動作 (call kill())

- 預設的情況下，不會執行這個檢查， **max_inactive_time = 0**
- 當下 **M85 S(time)** 指令設定 **max_inactive_time** 時，才會做檢查

Marlin loop function

2. manage_inactivity() 的講解

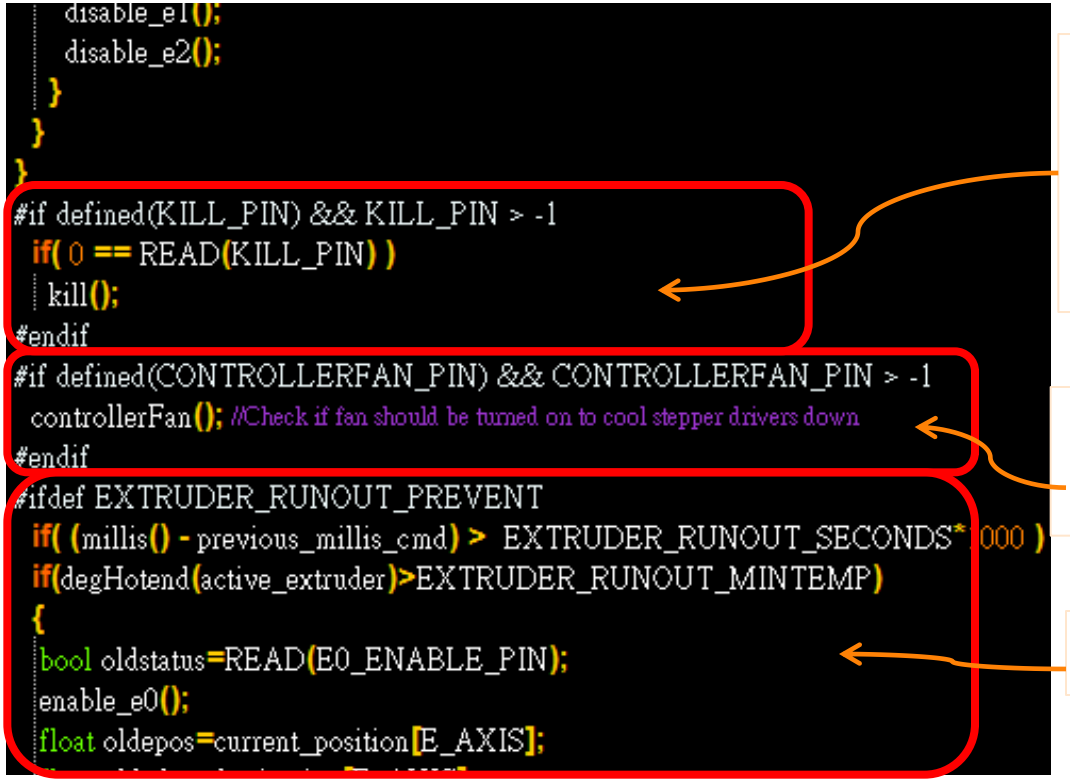
```
void manage_inactivity()
{
  if( (millis() - previous_millis_cmd) >  max_inactive_time )
    if(max_inactive_time)
      kill();
  if(stepper_inactive_time) {
    if( (millis() - previous_millis_cmd) >  stepper_inactive_time )
    {
      if(blocks_queued() == false) {
        disable_x();
        disable_y();
        disable_z();
        disable_e0();
        disable_e1();
        disable_e2();
      }
    }
  }
```

如果兩個 g code 驅動步進馬達的間隔時間 大於 **stepper_inactive_time (1 min)** 同時沒有新的 **block** ，則會 **disable** 所有的步進馬達

- 做路徑規畫 **call plan_buffer_line()** 時，會做 **enable** 的動作
- 或是下 **M17** 指令也會做 **enable** 的動作

Marlin loop function

2. manage_inactivity() 的講解



如果有定義 **KILL_PIN** 時，會檢查 **KILL_PIN** 是否被觸發，如果被觸發，則會 **disable** 所有的步進馬達與加熱器

如果有定義 **CONTROLLERFAN_PIN**，則會對風扇做控制

擠出頭 **RUNOUT** 的保護(*)

manage_inactivity function

2. manage_inactivity() 的講解

- EXTRUDER_RUNOUT_PREVENT 的講解
- 在 Configuration_adv.h 有定義 EXTRUDER_RUNOUT_PREVENT，預設是 remark 的

```
// extruder run-out prevention.
//if the machine is idle, and the temperature over MINTEMP, every
//#define EXTRUDER_RUNOUT_PREVENT
#define EXTRUDER_RUNOUT_MINTEMP 190
#define EXTRUDER_RUNOUT_SECONDS 30.
#define EXTRUDER_RUNOUT_ESTEPS 14. //mm fila
#define EXTRUDER_RUNOUT_SPEED 1500.  //extrus
#define EXTRUDER_RUNOUT_EXTRUDE 100
```

執行 runout prevent 的最低溫度

允許系統閒置的時間，超過會執行 EXTRUDER_RUNOUT_PREVENT

擠出頭擠出料的長度為 14 mm

擠出頭步進馬達的轉速 (mm/min)

擠出料長度的系數值

manage_inactivity function

2. manage_inactivity() 的講解

– EXTRUDER_RUNOUT_PREVENT 的講解

```
#ifdef EXTRUDER_RUNOUT_PREVENT
  if( (millis() - previous_millis_cmd) > EXTRUDER_RUNOUT_SECONDS*1000 )
  if(degHotend(active_extruder)>EXTRUDER_RUNOUT_MINTEMP)
  {
   bool oldstatus=READ(E0_ENABLE_PIN);
   enable_e0();
   float oldepos=current_position[E_AXIS];
   float oldedes=destination[E_AXIS];
   plan_buffer_line(current_position[X_AXIS], current_position[Y_AXIS], current_position[Z
                    current_position[E_AXIS]+EXTRUDER_RUNOUT_EXTRUDE*EXTRUDER
                    EXTRUDER_RUNOUT_SPEED/60.*EXTRUDER_RUNOUT_ESTEPS/axis_s
   current_position[E_AXIS]=oldepos;
   destination[E_AXIS]=oldedes;
   plan_set_e_position(oldepos);
   previous_millis_cmd=millis();
   st_synchronize();
   WRITE(E0_ENABLE_PIN,oldstatus);
  }
#endif
```

系統超過閒置時間，同時溫度大於**EXTRUDER_RUNOUT_MINTEMP**

備份擠出頭的狀態

擠料的路徑規畫

還原擠出頭的狀態

call st_synchronize() (*)

manage_inactivity function

2. manage_inactivity() 的講解

- st_synchronize() 的講解

```
// Block until all buffered steps are executed
void st_synchronize()
{
  while( blocks_queued()) {
    manage_heater();
    manage_inactivity();
    lcd_update();
  }
}
```

等待每個 block 都被執行，在等待的同時會執行以下動作

call manage_heater()

call manage_inactivity()

call lcd_update()

Marlin loop function

2. manage_inactivity() 的講解

檢查每個軸的步馬達是否要 disable

- 檢查目前每個 block 各個軸要輸出的 step數是 否都是 0，如果單一軸的 step 數都是 0 時，該軸的馬達會被 disable
- 做路徑規畫 call plan_buffer_line() 時，會做 enable 的動作
- 或是下 M17 指令也會做 enable 的動作

- manage_inactivity() 結束

G-code 解譯解說

G code parse

G code 相關的實作分成兩部分

- **get_command() 讀取 G code**
  - 讀取 serial port 或是 SD 卡 的 g code，存進 cmdbuffer
- **process_commands() 解譯並執行 G code**
  - 讀取cmdbuffer 進行 g code 的解譯，將解譯的內容做處理，包括軌跡規畫(寫入block_buffer)、加熱溫度的設定、系統參數設定…等。

get_command function

- 開始講解 **get_command()**

```
void get_command()
{
  while( MYSERIAL.available() > 0  && buflen < BUFSIZE) {
    serial_char = MYSERIAL.read();
    if(serial_char == '\n' ||
       serial_char == '\r' ||
       (serial_char == ':' && comment_mode == false) ||
       serial_count >= (MAX_CMD_SIZE - 1) )
    {
      if(!serial_count) { //if empty line
        comment_mode = false; //for new command
        return;
      }
      cmdbuffer[bufindw][serial_count] = 0; //terminate string
      if(!comment_mode){
        comment_mode = false; //for new command
        fromsd[bufindw] = false;
        if(strchr(cmdbuffer[bufindw], 'N') != NULL)
        {
          strchr_pointer = strchr(cmdbuffer[bufindw], 'N');
          gcode_N = (strtol(&cmdbuffer[bufindw][strchr_pointer - cmdb
```

get_command function

- **get_command() 的講解**

```
void get_command()
{
  while( MYSERIAL.available() > 0  && buflen < BUFSIZE) {
    serial_char = MYSERIAL.read();
    if(serial_char == '\n' ||
      serial_char == '\r' ||
      (serial_char == ':' && comment_mode == false) ||
      serial_count >= (MAX_CMD_SIZE - 1) )
    {
      if(!serial_count) { //if empty line
        comment_mode = false; //for new command
        return;
      }
      cmdbuffer[bufindw][serial_count] = 0; //terminate string
      if(!comment_mode){
        comment_mode = false; //for new command
        fromsd[bufindw] = false;
        if(strchr(cmdbuffer[bufindw], 'N') != NULL)
        {
          strchr_pointer = strchr(cmdbuffer[bufindw], 'N');
          gcode_N = (strtol(&cmdbuffer[bufindw][strchr_pointer - cmdb
```

當 **serial port** 有收到值且 **cmdbuffer** 有空間時， **cmdbuffer** 會讀取 **G code**

當讀到 **'\n'**、**'\r'** 、**':'**且不是註解或是滿 **95** 個字元時，即表示一條 **G code** 指令結束

get_command function

- **get_command() 的講解**

```
  comment_mode = false; //for new command
  return;
}
cmdbuffer[bufindw][serial_count] = 0; //terminate string
if(!comment_mode){
  comment_mode = false; //for new command
  fromsd[bufindw] = false;
  if(strchr(cmdbuffer[bufindw], 'N') != NULL)
  {
    strchr_pointer = strchr(cmdbuffer[bufindw], 'N');
    gcode_N = (strtol(&cmdbuffer[bufindw][strchr_pointer - cmdbuffer[bufindw] + 1], NULL, 10));
    if(gcode_N != gcode_LastN+1 && (strstr_P(cmdbuffer[bufindw], PSTR("M110")) == NULL) ) {
      SERIAL_ERROR_START;
      SERIAL_ERRORPGM(MSG_ERR_LINE_NO);
      SERIAL_ERRORLN(gcode_LastN);
      //Serial.println(gcode_N);
      FlushSerialRequestResend();
      serial_count = 0;
      return;
    }
```

當 **cmdbuffer** 有讀到 **'N'** 字元時，以十進位計算 **'N'** 字元之後的數字和 **gcode_N**

當同一個 **cmdbuffer** 有讀到 **M110** 的指令，即表示要以 **gcode_N** 設定目前指令的行數

當沒有讀到 **M110** 的指令，則 **gcode_N** 的值必須等於 **gcode_LastN + 1**，否則會傳送錯誤訊息

get_command function

- **get_command() 的講解**

```
}

if(strchr(cmdbuffer[bufindw], '*') != NULL)
{
  byte checksum = 0;
  byte count = 0;
  while(cmdbuffer[bufindw][count] != '*') checksum = checksum^cmdbuffer[bufindw][count++];
  strchr_pointer = strchr(cmdbuffer[bufindw], '*');

  if( (int)(strtod(&cmdbuffer[bufindw][strchr_pointer - cmdbuffer[bufindw] + 1], NULL)) != checksum) {
    SERIAL_ERROR_START;
    SERIAL_ERRORPGM(MSG_ERR_CHECKSUM_MISMATCH);
    SERIAL_ERRORLN(gcode_LastN);
    FlushSerialRequestResend();
    serial_count = 0;
    return;
  }
  //if no errors, continue parsing
}
else
{
  SERIAL_ERROR_START;
  SERIAL_ERRORPGM(MSG_ERR_NO_CHECKSUM);
  SERIAL_ERRORLN(gcode_LastN);
  FlushSerialRequestResend();
  serial_count = 0;
  return;
}
```

接續上一頁同一個指令

當 **cmdbuffer** 有讀到 **‘*’** 字元時，計算從字串起始位址到 **‘*’** 的 **checksum**

擷取 **’*’** 後的數字是否等於 **checksum**，如果不等於則傳送錯誤息

如果 **cmdbuffer** 沒有讀到 **‘*’** 字元，則傳送錯誤息

get_command function

- **get_command() 的講解**

```
if((strchr(cmdbuffer[bufindw], 'G') != NULL)){
  strchr_pointer = strchr(cmdbuffer[bufindw], 'G');
  switch((int)((strtod(&cmdbuffer[bufindw][strchr_pointer - cmdbuffer[bufindw] + 1], NULL)))
  case 0:
  case 1:
  case 2:
  case 3:
    if(Stopped == false) { // If printer is stopped by an error the G[0-3] codes are ignored.
  #ifdef SDSUPPORT
      if(card.saving)
        break;
  #endif //SDSUPPORT
      SERIAL_PROTOCOLLNPGM(MSG_OK);
    }
    else {
      SERIAL_ERRORLNPGM(MSG_ERR_STOPPED);
      LCD_MESSAGEPGM(MSG_STOPPED);
    }
    break;
  default:
    break;
  }

}
```

當 **cmdbuffer** 讀到 **G0, G1, G2, G3** 的指令時，如果系統是停止運作的狀態 ，則傳送錯誤訊息。如果是正常運作的狀態則回傳 **OK**。

get_command function

- **get_command() 的講解**

```
    }
    bufindw = (bufindw + 1)%BUFSIZE;
    buflen += 1;
  }
  serial_count = 0; //clear buffer
}
else
{
  if(serial_char == ';') comment_mode = true;
  if(!comment_mode) cmdbuffer[bufindw][serial_count++] = serial_char;
}
}
#ifdef SDSUPPORT
```

讀取一個指令結束，載入下一個 **cmdbuffer** 空間

當指令不是讀'\n'、'\r'、':'且不是註解或是沒滿 **95** 個字元時

- 將讀到的字元存進 **cmdbuffer**
- 如果是註解則略過

get_command function

- **get_command() 的講解**

```
#ifdef SDSUPPORT
if(!card.sdprinting || serial_count!=0){
  return;
}
while( !card.eof()  && buflen < BUFSIZE) {
  int16_t n=card.get();
  serial_char = (char)n;
  if(serial_char == '\n' ||
     serial_char == '\r' ||
     (serial_char == ':' && comment_mode == false) ||
     serial_count >= (MAX_CMD_SIZE - 1)||n==-1)
  {
    if(card.eof()){
      SERIAL_PROTOCOLLNPGM(MSG_FILE_PRINTED);
      stoptime=millis();
      char time[30];
      unsigned long t=(stoptime-starttime)/1000;
      int hours, minutes;
      minutes=(t/60)%60;
      hours=t/60/60;
      sprintf_P(time, PSTR("%i hours %i minutes"),hours, minutes);
      SERIAL_ECHO_START;
      SERIAL_ECHOLN(time);
      lcd_setstatus(time);
      card.printingHasFinished();
      card.checkautostart(true);

    }
```

從 **SD card** 讀 **g code** 指令，做法與 **serial port** 相同，所以 **BJ4**

get_command function

- **get_command() 的結束**

```
      lcd_setstatus(time);
      card.printingHasFinished();
      card.checkautostart(true);

    }
    if(!serial_count)
    {
      comment_mode = false; //for new command
      return; //if empty line
    }
    cmdbuffer[bufindw][serial_count] = 0; //terminate string
//    if(!comment_mode){
      fromsd[bufindw] = true;
      buflen += 1;
      bufindw = (bufindw + 1)%BUFSIZE;
//    }
    comment_mode = false; //for new command
    serial_count = 0; //clear buffer
  }
  else
  {
    if(serial_char == ';') comment_mode = true;
    if(!comment_mode) cmdbuffer[bufindw][serial_count++] = serial_char;
  }
 }

 #endif //SDSUPPORT

}
```

接續上一頁，從 SD card 讀 g code 指令

process_commands function

- 開始講解 **process_commands()**

```
void process_commands()
{
  unsigned long codenum; //throw away variable
  char *starpos = NULL;

  if(code_seen('G'))
  {
    switch((int)code_value())
    {
    case 0: // G0 -> G1
    case 1: // G1
      if(Stopped == false) {
        get_coordinates(); // For X Y Z E F
        prepare_move();
        //ClearToSend();
        return;
      }
```

process_commands function

- 講解 process_commands()

```
void process_commands()
{
  unsigned long codenum; //throw away variable
  char *starpos = NULL;

  if(code_seen('G'))
  {
    switch((int)code_value())
    {
    case 0: // G0 -> G1
    case 1: // G1
      if(Stopped == false) {
        get_coordinates(); // For X Y Z E F
        prepare_move();
        //ClearToSend();
        return;
      }
```

從 cmdbuffer 讀到 G code

G0, G1 指令，為讀取目標位置，準備做路徑規畫

process_commands function

- 講解 process_commands()

```
  //break;
case 2: //G2 - CW ARC
  if(Stopped == false) {
    get_arc_coordinates();
    prepare_arc_move(true);
    return;
  }
case 3: //G3 - CCW ARC
  if(Stopped == false) {
    get_arc_coordinates();
    prepare_arc_move(false);
    return;
  }
```

G2 指令，為讀取目標位置，準備做順時針的圓弧徑規畫

ex: G2 X1.75 Y-0.125 I0 J-0.375

G3 指令，為讀取目標位置，準備做逆時針的圓弧徑規畫

process_commands function

- 講解 process_commands()

```
case 4: // G4 dwell
  LCD_MESSAGEPGM(MSG_DWELL);
  codenum = 0;
  if(code_seen('P')) codenum = code_value(); // milliseconds to wait
  if(code_seen('S')) codenum = code_value() * 1000; // seconds to wait

  st_synchronize();
  codenum += millis();  // keep track of when we started waiting
  previous_millis_cmd = millis();
  while(millis()  < codenum ){
    manage_heater();
    manage_inactivity();
    lcd_update();
  }
  break;
```

G4 指令，為設定等待的時間後，並開始做等待的動作

在等待的時間內會執行
manage_heater()
manage_inactivity()
lcd_update()

process_commands function

- 講解 process_commands()

```
#ifdef FWRETRACT
case 10: // G10 retract
if(!retracted)
{
  destination[X_AXIS]=current_position[X_AXIS];
  destination[Y_AXIS]=current_position[Y_AXIS];
  destination[Z_AXIS]=current_position[Z_AXIS];
  current_position[Z_AXIS]+=-retract_zlift;
  destination[E_AXIS]=current_position[E_AXIS]-retract_length;
  feedrate=retract_feedrate;
  retracted=true;
  prepare_move();
}

break;
```

預設是沒有 **FWRETRACT**，
且目前只做過部分的測試

G10，為回捲塑料的 指令
如果 **retracted** 為 **true** ，則結束。
否則，

- **XYZ** 目標位置設為上一個 **block** 的位置
- 同時 **Z** 目前位置多減 **0.08 mm**，
- **E** 目標位置回捲 **3mm**
- 設定 **feedrate = retract_feedrate (17*60)**
- 設定 **retracted = true**

•準備做路徑規畫

process_commands function

- 講解 process_commands()

```
#ifdef FWRETRACT
case 10: // G10 retract
if(!retracted)
{
  destination[X_AXIS]=current_position[X_AXIS];
  destination[Y_AXIS]=current_position[Y_AXIS];
  destination[Z_AXIS]=current_position[Z_AXIS];
  current_position[Z_AXIS]+=-retract_zlift;
  destination[E_AXIS]=current_position[E_AXIS]-retract_length;
  feedrate=retract_feedrate;
  retracted=true;
  prepare_move();
}

break;
```

預設是沒有 **FWRETRACT**，
且目前只做過部分的測試

G10，為回捲塑料的 指令
如果 **retracted** 為 **true** ，則結束。
否則，

- **XYZ** 目標位置設為上一個 **block** 的位置
- 同時 **Z** 目前位置多減 **0.08 mm**，
- **E** 目標位置回捲 **3mm**
- 設定 **feedrate = retract_feedrate (17*60)**
- 設定 **retracted = true**

•準備做路徑規畫

process_commands function

- 講解 process_commands()

```
break;
case 11: // G10 retract_recover
if(!retracted)
{
  destination[X_AXIS]=current_position[X_AXIS];
  destination[Y_AXIS]=current_position[Y_AXIS];
  destination[Z_AXIS]=current_position[Z_AXIS];

  current_position[Z_AXIS]+=retract_zlift;
  current_position[E_AXIS]+=-retract_recover_length;
  feedrate=retract_recover_feedrate;
  retracted=false;
  prepare_move();
}
break;
#endif //FWRETRACT
```

預設是沒有 **FWRETRACT**，且目前只做過部分的測試

G11，為回補塑料的 指令，還原執行 **G10** 前的狀態，即做與 **G10** 相反的動作

process_commands function

- 講解 process_commands()

```
break;
#endif //FWRETRACT
case 28: //G28 Home all Axis one at a time
saved_feedrate = feedrate;
saved_feedmultiply = feedmultiply;
feedmultiply = 100;
previous_millis_cmd = millis();

enable_endstops(true);

for(int8_t i=0; i < NUM_AXIS; i++) {
 destination[i] = current_position[i];
}
  feedrate = 0.0;
```

G28，為歸回零點的指令

備份原先的 **feedrate**

更新執行指令的 **timer**

開啟檢查 **endstop**的功能

process_commands function

- 講解 process_commands()

```
home_all_axis = !((code_seen(axis_codes[0])) || (code_seen(axis_codes[1])) || (code_seen(axis_codes[2])));

#if Z_HOME_DIR > 0                      // If homing away from BED do Z first
if((home_all_axis) || (code_seen(axis_codes[Z_AXIS]))) {
  HOMEAXIS(Z);
}
#endif

#ifdef QUICK_HOME
if((home_all_axis)||( code_seen(axis_codes[X_AXIS]) && code_seen(axis_codes[Y_AXIS])) )  //first diagonal mo
{
  current_position[X_AXIS] = 0;current_position[Y_AXIS] = 0;

  #ifndef DUAL_X_CARRIAGE
  int x_axis_home_dir = home_dir(X_AXIS);
```

記錄 **cmdbuffer** 有要歸零的軸

如果回 **home** 的方向是在 **MAX** 方向且指令要求 **Z** 軸歸 **home** 時，**call HOMEAXIS(Z) (*)**

process_commands function

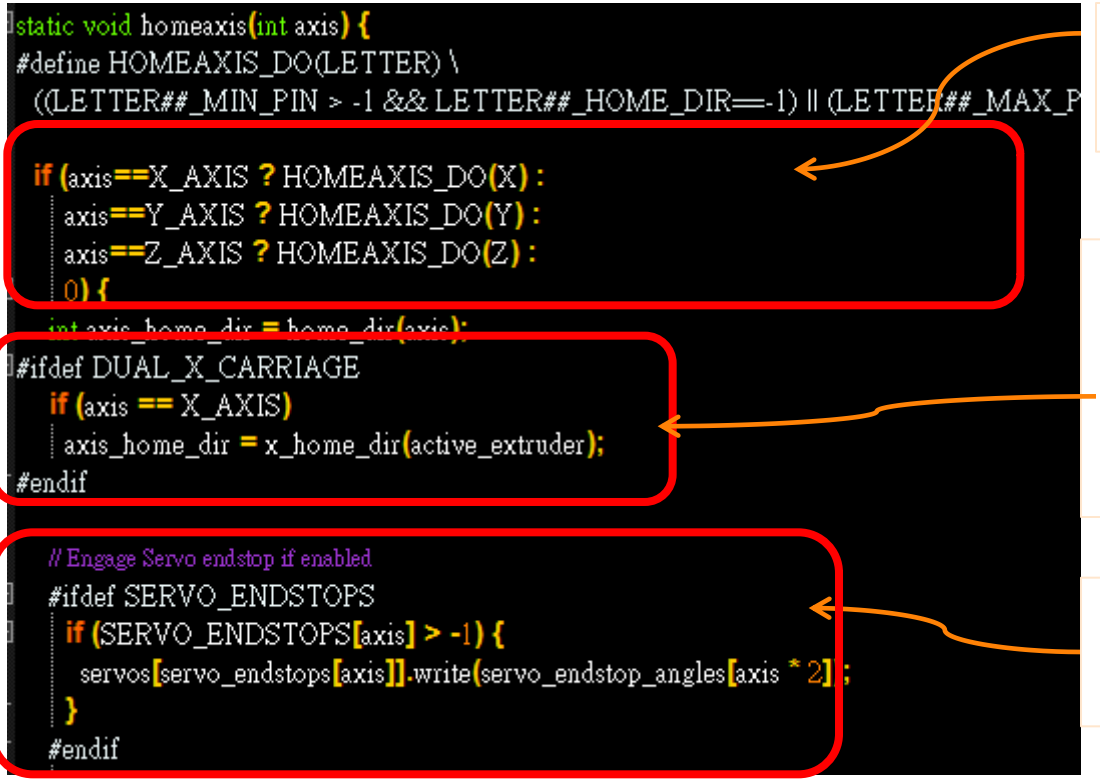
- 講解 process_commands()
  - 講解 HOMEAXIS()

```
  }
}
#define HOMEAXIS(LETTER) homeaxis(LETTER##_AXIS)

void process_commands()
{
  unsigned long codenum; //throw away variable
  char *starpos = NULL;

  if(code_seen('G'))
  {
    switch((int)code_value())
    {
```

call homeaxis ()(*)

process_commands function

- 講解 process_commands()
  - 講解 homeaxis()



如果 **axis** 為 **X, Y, Z** 其中一軸，則執行以下歸 **home** 的動作

如果 **3D printer** 是採用雙 **X** 軸，即各別的 **X** 軸分別用一個步進馬達來帶動一個擠出頭時，則會判斷 **X** 軸是哪一顆要歸 **home**

如果 **axis** 是用 **RC servo** 時，才做的動作

process_commands function

- 講解 process_commands()
  - 講解 homeaxis()

```
  if (SERVO_ENDSTOPS[axis] > -1) {
    servos[servo_endstops[axis]].write(servo_endstop_angles[axis * 2]);
  }
#endif

current_position[axis] = 0;
plan_set_position(current_position[X_AXIS], current_position[Y_AXIS], current_posit
destination[axis] = 1.5 * max_length(axis) * axis_home_dir;
feedrate = homing_feedrate[axis];
plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS], de
st_synchronize();

current_position[axis] = 0;
plan_set_position(current_position[X_AXIS], current_position[Y_AXIS], current_posit
destination[axis] = -home_retract_mm(axis) * axis_home_dir;
plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS], de
st_synchronize();

 destination[axis] = 2*home_retract_mm(axis) * axis_home_dir;
plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS], de
st_synchronize();
```

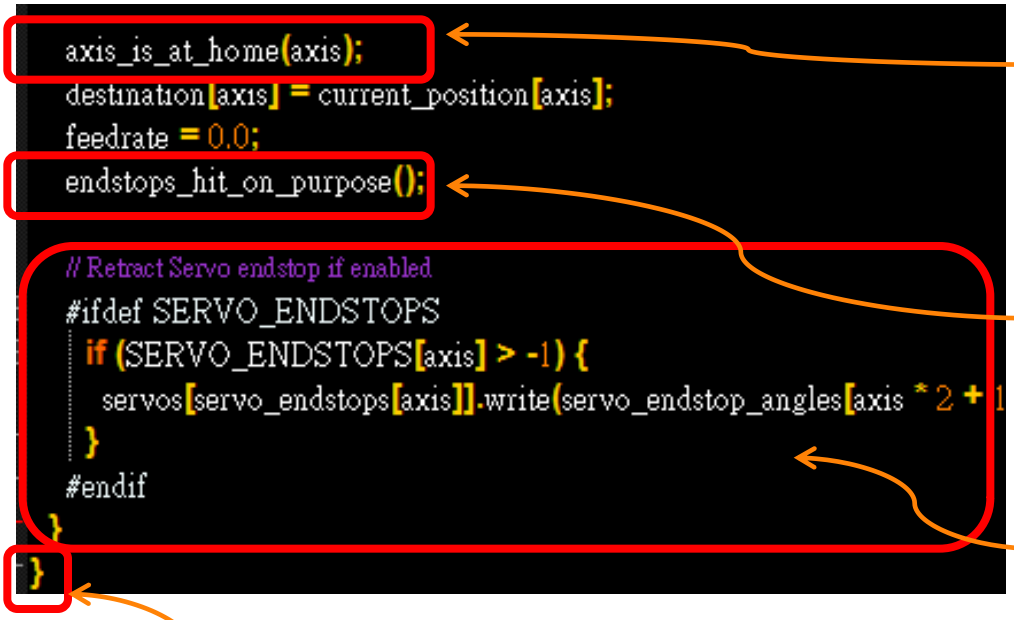
- 設定 **axis** 目前位置為 **0**，
- 設定 **axis** 目標位置為 **1.5 * MAX_LENGTH*HOME_DIR**
- 設定歸 **home** 的 **feedrate**
- 做路徑規畫並等待執行完畢

- 設定 **axis** 目前位置為 **0**，
- 設定 **axis** 目標位置，**X, Y** 軸為 **5 mm, Z**軸為 **1 mm**，方向與 **HOME_DIR** 反向
- 做路徑規畫並等待執行完畢

再次做歸 **home** 的動作

process_commands function

- 講解 process_commands()
  - 講解 homeaxis()



設定 **axis** 目前位置為 **home** 的位置，以及該軸的工作空間的 **min_pos** 與 **max_pos**

清除 **endstop** 的狀態

如果 **axis** 是用 **RC servo** 時，才做的動作

  - homeaxis() 結束

process_commands function

- 講解 process_commands()

```
#ifdef QUICK_HOME
if((home_all_axis)||( code_seen(axis_codes[X_AXIS]) && code_seen(axis_codes[Y
{
 current_position[X_AXIS] = 0;current_position[Y_AXIS] = 0;

#ifndef DUAL_X_CARRIAGE
 int x_axis_home_dir = home_dir(X_AXIS);
#else
 int x_axis_home_dir = x_home_dir(active_extruder);
#endif

 plan_set_position(current_position[X_AXIS], current_position[Y_AXIS], current_
 destination[X_AXIS] = 1.5 * max_length(X_AXIS) * x_axis_home_dir;destination
 feedrate = homing_feedrate[X_AXIS];
 if(homing_feedrate[Y_AXIS]<feedrate)
  feedrate =homing_feedrate[Y_AXIS];
 plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS
 st_synchronize();

 axis_is_at_home(X_AXIS);
 axis_is_at_home(Y_AXIS);
 plan_set_position(current_position[X_AXIS], current_position[Y_AXIS], current_
 destination[X_AXIS] = current_position[X_AXIS];
 destination[Y_AXIS] = current_position[Y_AXIS];
 plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS
 feedrate = 0.0;
 st_synchronize();
 endstops_hit_on_purpose();
```

- **QUICK_HOME 預設為 remark**
- **QUICK_HOME 的做法是 X, Y軸為同時歸 HOME，除此之外與 homeaxis() 的做法相似。**

process_commands function

- 講解 process_commands()

```
  current_position[Y_AXIS] = destination[Y_AXIS];
  current_position[Z_AXIS] = destination[Z_AXIS];
}
#endif

if((home_all_axis) || (code_seen(axis_codes[X_AXIS])))
{
#ifdef DUAL_X_CARRIAGE
  int tmp_extruder = active_extruder;
  active_extruder = !active_extruder;
  HOMEAXIS(X);
  inactive_x_carriage_pos = current_position[X_AXIS];
  active_extruder = tmp_extruder;
#endif
  HOMEAXIS(X);
}

if((home_all_axis) || (code_seen(axis_codes[Y_AXIS]))) {
  HOMEAXIS(Y);
}

#if Z_HOME_DIR < 0                      // If homing towards BED do Z last
if((home_all_axis) || (code_seen(axis_codes[Z_AXIS]))) {
  HOMEAXIS(Z);
}
```

執行 X 軸歸 home 的動作

執行 Y 軸歸 home 的動作

如果 Z 軸回 home 的方向是在 MIN 方向且指令要求 Z 軸歸 home 時，作歸 home 的動作，確保加熱板及擠出頭不會受損

process_commands function

- 講解 process_commands()

```
if(code_seen(axis_codes[X_AXIS]))
{
  if(code_value_long() != 0) {
    current_position[X_AXIS]=code_value()+add_homeing[0];
  }
}

if(code_seen(axis_codes[Y_AXIS])) {
  if(code_value_long() != 0) {
    current_position[Y_AXIS]=code_value()+add_homeing[1];
  }
}

if(code_seen(axis_codes[Z_AXIS])) {
  if(code_value_long() != 0) {
    current_position[Z_AXIS]=code_value()+add_homeing[2];
  }
}
plan_set_position(current_position[X_AXIS], current_position[Y_
```

如果歸 home 指令裡，附加的X, Y, Z 軸移動位置時，會讀取各個軸的目標位置做路徑規畫

process_commands function

- 講解 process_commands()

```
#ifdef ENDSTOPS_ONLY_FOR_HOMING
  enable_endstops(false);
#endif

feedrate = saved_feedrate;
feedmultiply = saved_feedmultiply;
previous_millis_cmd = millis();
endstops_hit_on_purpose();
break;
```

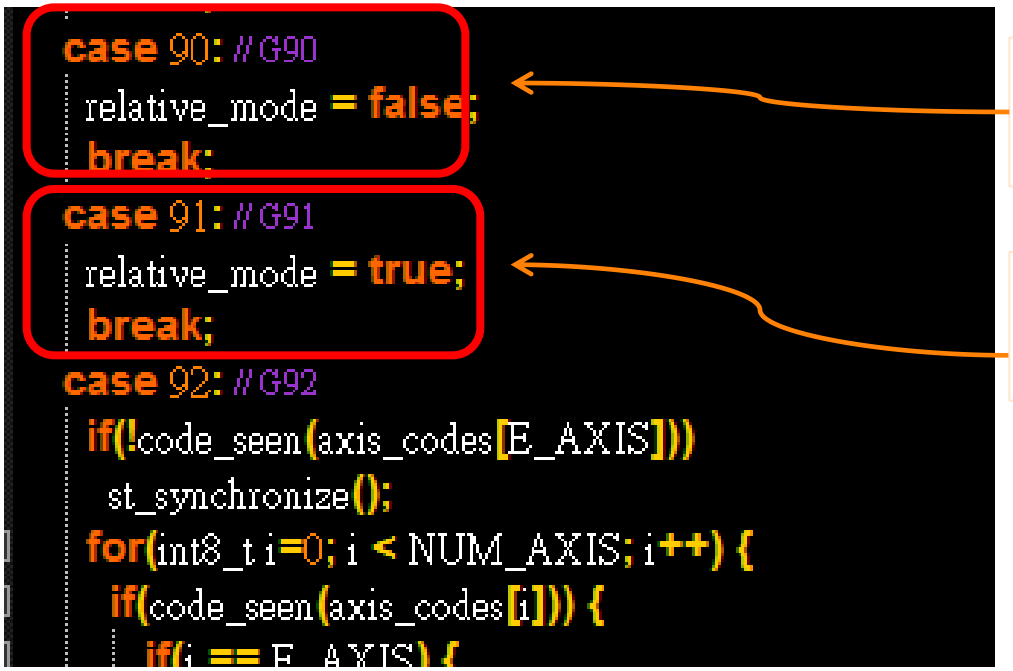
如果 **endstop** 只用在歸 **home** 時會 **diable** 檢查 **endstop** 的功能,

還原先前的 **feedrate**

– **G28** 結束

process_commands function

- 講解 process_commands()



G90 為設定位置的模式為絕對(absolute)位置

G91 設定位置的模式為相對(relative)位置

process_commands function

- 講解 process_commands()

```
case 92: // G92
  if(!code_seen(axis_codes[E_AXIS]))
    st_synchronize();
  for(int8_t i=0; i < NUM_AXIS; i++) {
    if(i == E_AXIS) {
      current_position[i] = code_value();
      plan_set_e_position(current_position[E_AXIS]);
    }
    else {
      current_position[i] = code_value()+add_homeing[i];
      plan_set_position(current_position[X_AXIS], current_position[Y
    }
  }
  break;
}
```

G92 為重新設定目前位置

如果指令沒有設定 E 軸位置，則會等待 block 執行完畢

設定 E 軸的位置時，會先 disable 中斷，設定完再 restore 中斷的狀態

```
CRITICAL_SECTION_START;
count_position[E_AXIS] = e;
CRITICAL_SECTION_END;
```

process_commands function

- 講解 process_commands()

```
else if(code_seen('M'))
{
  switch( (int)code_value() )
  {
#ifdef ULTIPANEL
  case 0: // M0 - Unconditional stop - Wait for user button press on LCD
  case 1: // M1 - Conditional stop - Wait for user button press on LCD
  {
    LCD_MESSAGEPGM(MSG_USERWAIT);
    codenum = 0;
    if(code_seen('P')) codenum = code_value(); // milliseconds to w
    if(code_seen('S')) codenum = code_value() * 1000; // seconds

    st_synchronize();
```

講解 M code 部分，只講解重要的地方

process_commands function

- 講解 process_commands()

```
#ifdef FWRETRACT
case 207: //M207 - set retract length S[positive mm] F[feedrate mm/sec] Z[
{
  if(code_seen('S'))
  {
    retract_length = code_value() ;
  }
  if(code_seen('F'))
  {
    retract_feedrate = code_value() ;
  }
  if(code_seen('Z'))
  {
    retract_zlift = code_value() ;
  }
}break;
```

預設是沒有 **FWRETRACT**，
且目前只做過部分的測試

M207為設定 G10 的參數

- 設定擠出頭回捲的長度(mm)
- 設定 **feedrate (mm/sec)**
- 設定 **Z** 軸提升的高度 **(mm)**

process_commands function

- 講解 process_commands()

```
case 208: // M208 - set retract recover length S[positive mm surplus to
{
  if(code_seen('S'))
  {
    retract_recover_length = code_value() ;
  }
  if(code_seen('F'))
  {
    retract_recover_feedrate = code_value() ;
  }
}break;
```

預設是沒有 **FWRETRACT**，
且目前只做過部分的測試

M208為設定 G11 的參數

- 設定擠出頭進料的長度(mm)
- 設定 **feedrate (mm/sec)**

process_commands function

- 講解 process_commands()

```
#ifdef FILAMENTCHANGEENABLE
case 600: //Pause for filament change X[pos] Y[pos] Z[relative lift] E[initial retract] L[later retract
{
  float target[4];
  float lastpos[4];
  target[X_AXIS]=current_position[X_AXIS];
  target[Y_AXIS]=current_position[Y_AXIS];
  target[Z_AXIS]=current_position[Z_AXIS];
  target[E_AXIS]=current_position[E_AXIS];
  lastpos[X_AXIS]=current_position[X_AXIS];
  lastpos[Y_AXIS]=current_position[Y_AXIS];
  lastpos[Z_AXIS]=current_position[Z_AXIS];
  lastpos[E_AXIS]=current_position[E_AXIS];
  //retract by E
  if(code_seen('E'))
  {
    target[E_AXIS]+= code_value();
  }
  else
  {
```

M600 為換塑料的指令

- 備份目前位置
- 設定擠出頭回捲的長度，長度是 **initial retract** 或 **default** 值 **(-2)**
- 設定 **Z** 軸提升 的高度，長度是 **relative lift** 或 **default** 值 **(3)**
- 設定 **X, Y** 軸的位置，位置為目前的位置加上 **offset** ， **offset** 為 **pos** 或**default** 值 **(10)**
- 擠出頭退料
  - 設定擠出頭回捲的長度，長度是 **later retract distance** 或 **default** 值 **(-100)**
- 下一頁

process_commands function

- 講解 process_commands()

```
//return to normal
if(code_seen('L'))
{
  target[E_AXIS]+= -code_value();
}
else
{
  #ifdef FILAMENTCHANGE_FINALRETRACT
    target[E_AXIS]+=(-1)*FILAMENTCHANGE_FINALRETRACT ;
  #endif
}
current_position[E_AXIS]=target[E_AXIS]; //the long retract of L is compensate
plan_set_e_position(current_position[E_AXIS]);
plan_buffer_line(target[X_AXIS], target[Y_AXIS], target[Z_AXIS], targe
plan_buffer_line(lastpos[X_AXIS], lastpos[Y_AXIS], target[Z_AXIS], tar
plan_buffer_line(lastpos[X_AXIS], lastpos[Y_AXIS], lastpos[Z_AXIS], ta
plan_buffer_line(lastpos[X_AXIS], lastpos[Y_AXIS], lastpos[Z_AXIS], la
}
break;
```

M600 為換塑料的指令

- 接上一頁
- 規畫上敘的路徑並執行
- **Disable** 所有的擠出頭
- 等待 **EN_C button (LCD)clicked**
- 擠出頭進料
  - 設定擠出頭給進的長度，長度是 **later retract distance** 或 **default** 值 **(100)**
- 規畫上列的路徑並執行
- 還原目前位置
- 結束

process_commands function

- 講解 process_commands()

```
else if(code_seen('T'))
{
  tmp_extruder = code_value();
  if(tmp_extruder >= EXTRUDERS) {
    SERIAL_ECHO_START;
    SERIAL_ECHO("T");
    SERIAL_ECHO(tmp_extruder);
    SERIAL_ECHOLN(MSG_INVALID_EXTRUDER);
  }
  else {
    boolean make_move = false;
    if(code_seen('F')) {
      make_move = true;
      next_feedrate = code_value();
      if(next_feedrate > 0.0) {
        feedrate = next_feedrate;
      }
    }
    #if EXTRUDERS > 1
    if(tmp_extruder != active_extruder) {
```

T [extruder] F[feedrate] 為換擠出頭做列印工作的指令

讀取要更換擠出頭的編號

讀取更換後的 **feedrate**

process_commands function

- 講解 process_commands()

```
#if EXTRUDERS > 1
if(tmp_extruder != active_extruder) {
  // Save current position to return to after applying extruder offset
  memcpy(destination, current_position, sizeof(destination));
#ifdef DUAL_X_CARRIAGE
  // only apply Y extruder offset in dual x carriage mode (x offset is already used in de
  current_position[Y_AXIS] = current_position[Y_AXIS] -
                             extruder_offset[Y_AXIS][active_extruder] +
                             extruder_offset[Y_AXIS][tmp_extruder];

  float tmp_x_pos = current_position[X_AXIS];

  // Set the new active extruder and position
  active_extruder = tmp_extruder;
  axis_is_at_home(X_AXIS); //this function updates X min/max values.
  current_position[X_AXIS] = inactive_x_carriage_pos;
  inactive_x_carriage_pos = tmp_x_pos;
#else
```

擠出頭數要大於 1，並且檢查指定的擠出頭與目前的擠出頭是否相同，相同則結束

備份舊的擠出頭最後的位置

如果 DUAL_X_CARRIAGE 結構時，會是重新設定的擠出頭 X, Y 軸的位置

process_commands function

- 講解 process_commands()

```
  current_position[X_AXIS] = inactive_x_carriage_pos;
  inactive_x_carriage_pos = tmp_x_pos;
#else
  // Offset extruder (only by XY)
  int i;
  for(i = 0; i < 2; i++) {
    current_position[i] = current_position[i] -
                          extruder_offset[i][active_extruder] +
                          extruder_offset[i][tmp_extruder];
  }
  // Set the new active extruder and position
  active_extruder = tmp_extruder;
#endif //else DUAL_X_CARRIAGE
  plan_set_position(current_position[X_AXIS], current_position[Y_AXIS],
  // Move to the old position if 'F' was in the parameters
  if(make_move && Stopped == false) {
    prepare_move();
  }
}
```

如果是其他結構時，則載入該擠出頭的 **offset** 作定位的動作

設定新的擠出頭的編號並執行上敘的路徑規畫

process_commands function

- 講解 process_commands()

```
    // Move to the old position if 'F' was in the parameters
    if(make_move && Stopped == false) {
      prepare_move();
    }
  }
  #endif
  SERIAL_ECHO_START;
  SERIAL_ECHO(MSG_ACTIVE_EXTRUDER);
  SERIAL_PROTOCOLLN((int)active_extruder);
}
```

更換擠出頭後，會傳送的訊息

其他 M code 的列表

- M0 Unconditional stop - Wait for user button press on LCD
- M1 Conditional stop - Wait for user button press on LCD
- M20 List SD card
- M21 Initialize SD card
- M22 Release SD card
- M23 Select SD file
- M24 Start/resume SD print
- M25 Pause SD print
- M26 Set SD position
- M27 Report SD print status

其他 M code 的列表

- M28 Begin write to SD card
- M29 Stop writing to SD card
- M30 Delete a file on the SD card
- M42 Change pin status
- M104 Set Extruder Temperature
- M140 Set Hotbed Temperature
- M105 Get Extruder Temperature
- M109 Wait for extruder heater to reach target
- M190 Wait for bed heater to reach target

其他 M code 的列表

- M106 Fan On
- M107 Fan Off
- M80 ATX Power On
- M81 ATX Power Off
- M82 set extruder to absolute mode
- M83 set extruder to relative mode
- M17 Enable/Power all stepper motors
- M18/ M84 Disable all stepper motors
- M85 set max_inactive_time
- M92 Set axis_steps_per_unit

其他 M code 的列表

- M114 Get Current Position
- M115 Get Firmware Version and Capabilities
- M119 Get Endstop Status
- M120 enable_endstops(false)
- M121 enable_endstops(true)
- M201 Set max printing acceleration
- M203 Set max feedrate mm/sec
- M204 Set default acceleration

其他 M code 的列表

- **M205 advanced settings**
  - **minimumfeedrate, mintravelfeedrate, minsegmenttime, max_xy_jerk, max_z_jerk, max_e_jerk**
- **M206 set additional homeing offset**
- **M218 set hotend offset (mm)**
  - **T<extruder_number> X<offset_on_X> Y<offset_on_Y>**
- **M220 S<factor in percent> set speed factor override percentage (feedmultiply)**
- **M221 S<factor in percent> set extrude factor override percentage (extrudemultiply)**

其他 M code 的列表

- M280 set servo position absolute
  - P: servo index, S: angle or microseconds
- M300 play beep sound
- M301 Set PID parameters (Hot End)
- M304 Set PID parameters (Hot Bed)
- M240 Triggers a camera by emulating a Canon RC-1
- M302 allow cold extrudes or set the minimum extrude temperature
- M303 PID autotune
- M400 finish all moves

其他 M code 的列表

- M500 Store settings in EEPROM
- M501 Read settings from EEPROM
- M502 Revert to default settings
- M503 print settings currently in memory
- M540 Use S[0/1] to enable or disable the stop SD card print on endstop hit
- M907 Set digital trimpot motor current using axis codes
- M908 Control digital trimpot directly
- M350 Set microstepping mode
- M351 Toggle MS1 MS2 pins directly

其他 M code 的列表

- **M999 Restart after being stopped**
  - 在發生錯誤使系統停止的情況下，如果排除錯誤後，可以下 **M999** 指令，使系統重新開始

Marlin Planner

3D Printer 韌體原始碼閱讀心得 (Ⅲ)

Outline

- **Marlin 的內部參數介紹 (二)**
- **Marlin 路徑規畫的實作**
  - 直線軌跡的實作
  - 圓弧軌跡的實作

Marlin 內部參數介紹(二)

Marlin 內部參數

- 歸 home 的 feedrate 參數 (Configurature.h)

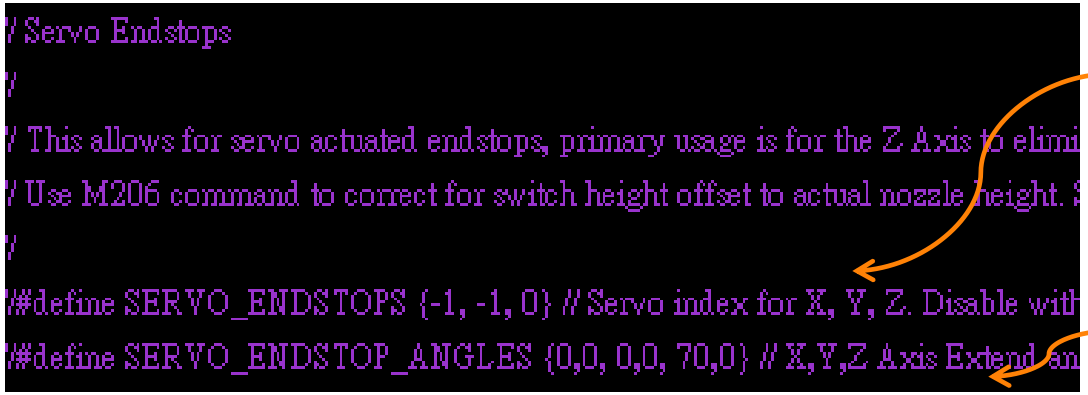
```
/// MOVEMENT SETTINGS
#define NUM_AXIS 4 // The axis order in all axis related arrays is X, Y, Z, E
#define HOMING_FEEDRATE {50*60, 50*60, 4*60, 0}  // set the homing
```

- 各軸預設的 jerk 參數 (Configurature.h)

```
// The speed change that does not require acceleration (i.e. the software might assume it can be done instantaneou
#define DEFAULT_XYJERK                20.0    // (mm/sec)
#define DEFAULT_ZJERK                 0.4     // (mm/sec)
#define DEFAULT_EJERK                 5.0     // (mm/sec)
```

Marlin 內部參數

- 設定 RC servo 的參數 (Configurature.h)
  - 預設是關掉的



各軸 RC servo index 的設定
- -1 表示 disable

各軸 RC servo 歸 home 位置

Marlin 內部參數

- **Marlin_main.cpp 的參數**

```
float homing_feedrate[] = HOMING_FEEDRATE;
bool axis_relative_modes[] = AXIS_RELATIVE_MODES;
int feedmultiply=100; //100->1 200->2
int saved_feedmultiply;
int extrudemultiply=100; //100->1 200->2
float current_position[NUM_AXIS] = { 0.0, 0.0, 0.0, 0.0 };
```

各軸位置模式 的狀態
- false 表示絕對位置
- true 表示相對位置

feedrate 的乘數 ，預設是 100

記錄目前的位置

```
float add_homeing[3]={0,0,0};
float min_pos[3] = { X_MIN_POS, Y_MIN_POS, Z_MIN_POS };
float max_pos[3] = { X_MAX_POS, Y_MAX_POS, Z_MAX_POS };
```

home offset

最小與最大的工作空間

Marlin 內部參數

- **Marlin_main.cpp 的參數**

```
#if EXTRUDERS > 1
float extruder_offset[2][EXTRUDERS] = {
#if defined(EXTRUDER_OFFSET_X) && defined(EXTRUDER_OFFSET_Y)
  EXTRUDER_OFFSET_X, EXTRUDER_OFFSET_Y
#endif
};
#endif
uint8_t active_extruder = 0;
int fanSpeed=0;
```

擠出頭 **offset** 設定

目前擠出頭的編號

風扇的轉速

```
#ifdef SERVO_ENDSTOPS
  int servo_endstops[] = SERVO_ENDSTOPS;
  int servo_endstop_angles[] = SERVO_ENDSTOP_ANGLES;
#endif
```

各軸 **RC servo** 的設定

Marlin 內部參數

- **Marlin_main.cpp 的參數**

```
#ifdef FWRETRACT
  bool autoretract_enabled=true;
  bool retracted=false;
  float retract_length=3, retract_feedrate=17*60, retract_zlift=0.8;
  float retract_recover_length=0, retract_recover_feedrate=8*60;
#endif
```

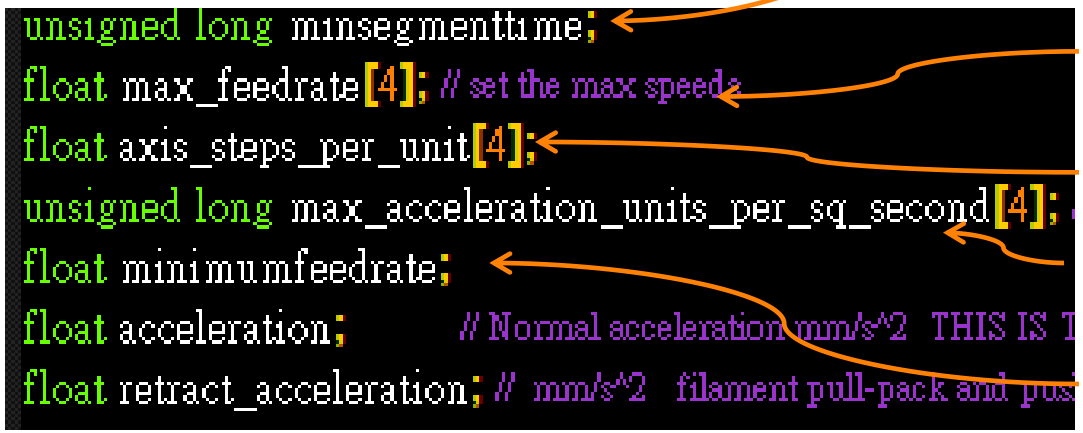
預設是沒有 **FWRETRACT**，且目前只做過部分的測試

FWRETRACT 的設定

- 可參考 **marlin main loop** 的講解指令 **G10** 的投影片

Marlin 內部參數

- **planner.cpp 的參數**



執行路徑的最小時間
- 預設是 2ms

各軸最大的 feedrate

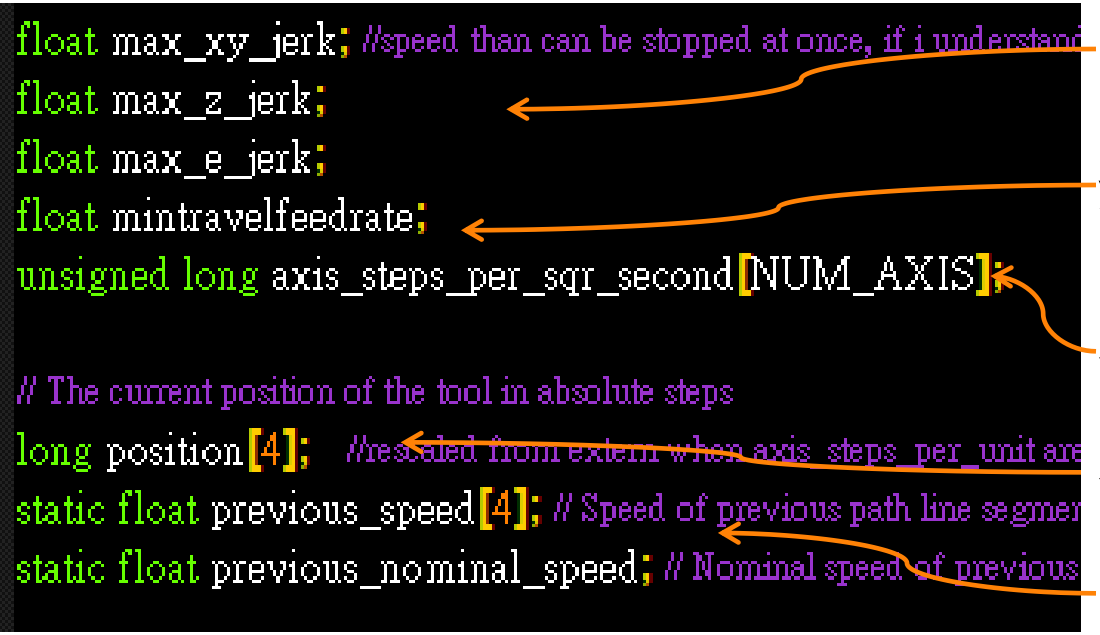
各軸移動 1 mm 的 step 數

各軸最大的加速度

最小的 feedrate，預設是 0

Marlin 內部參數

- **planner.cpp 的參數**



各軸的最大 **jerk**

最小的 **travel feedrate**

- 預 設是 **0**

各軸的加速度**(step/sec_sqr)**

各軸的位置用於計算路徑

前一路徑的各軸 **speed** 及 **nominal speed**

Marlin 內部參數

- **planner.cpp 的參數**

在執行g code 指令時， autotemp 會控制加熱

```
#ifdef AUTOTEMP
float autotemp_max=250;
float autotemp_min=210;
float autotemp_factor=0.1;
bool autotemp_enabled=false;
#endif
```

autotemp 的最高溫及最低溫

autotemp 的 gain 值

autotemp enable 的狀態

Marlin 內部參數

- **planner.cpp 的參數**

```
block_t block_buffer[BLOCK_BUFFER_SIZE];
volatile unsigned char block_buffer_head;
volatile unsigned char block_buffer_tail;
```

路徑規畫的資料結構

- **BLOCK_BUFFER_SIZE = 18**
- 詳細的內容可參考 **marlin main loop** 的投影片

記錄 **block** 的頭跟尾

Marlin 內部參數

- **temperature.cpp 的參數**

```
int target_temperature[EXTRUDERS] = { 0 };
int target_temperature_bed = 0;
int current_temperature_raw[EXTRUDERS] = { 0 };
float current_temperature[EXTRUDERS] = { 0.0 };
int current_temperature_bed_raw = 0;
float current_temperature_bed = 0.0;
```

目標溫度

目前溫度

目前溫度的 **ADC** 值

Marlin 內部參數

- **temperature.cpp 的參數**

```
#ifdef PIDTEMP
 float Kp=DEFAULT_Kp;
 float Ki=(DEFAULT_Ki*PID_dT);
 float Kd=(DEFAULT_Kd/PID_dT);
 #ifdef PID_ADD_EXTRUSION_RATE
  float Kc=DEFAULT_Kc;
 #endif
#endif //PIDTEMP

#ifdef PIDTEMPBED
 float bedKp=DEFAULT_bedKp;
 float bedKi=(DEFAULT_bedKi*PID_dT);
 float bedKd=(DEFAULT_bedKd/PID_dT);
#endif //PIDTEMPBED

#ifdef FAN_SOFT_PWM
 unsigned char fanSpeedSoftPwm;
#endif
```

擠出頭 **Kp, Ki, Kd** 的值

加熱板 **Kp, Ki, Kd** 的值

計算風扇轉速 **PWM** 的值

Marlin 內部參數

- **temperature.cpp 的參數**

```
#ifdef PIDTEMP
  //static cannot be external:
  static float temp_iState[EXTRUDERS] = { 0 };
  static float temp_dState[EXTRUDERS] = { 0 };
  static float pTerm[EXTRUDERS];
  static float iTerm[EXTRUDERS];
  static float dTerm[EXTRUDERS];
  //int output;
  static float pid_error[EXTRUDERS];
  static float temp_iState_min[EXTRUDERS];
  static float temp_iState_max[EXTRUDERS];
  // static float pid_input[EXTRUDERS];
  // static float pid_output[EXTRUDERS];
  static bool pid_reset[EXTRUDERS];
#endif //PIDTEMP
```

擠出頭用 **PID** 做加熱控制時，計算 **PWM** 會用到的參數

- 可參考 **marlin main loop** 的投影片

Marlin 內部參數

- **temperature.cpp 的參數**

```
#ifdef PIDTEMPBED
  //static cannot be external:
  static float temp_iState_bed = { 0 };
  static float temp_dState_bed = { 0 };
  static float pTerm_bed;
  static float iTerm_bed;
  static float dTerm_bed;
  //int output;
  static float pid_error_bed;
  static float temp_iState_min_bed;
  static float temp_iState_max_bed;
#else //PIDTEMPBED
  static unsigned long  previous_millis_bed_heater;
#endif //PIDTEMPBED
  static unsigned char soft_pwm[EXTRUDERS];
  static unsigned char soft_pwm_bed;
#ifdef FAN_SOFT_PWM
  static unsigned char soft_pwm_fan;
#endif
```

加熱板用 **PID** 做加熱控制時，計算 **PWM** 會用到的參數

- 可參考 **marlin main loop** 的投影片

擠出頭及加熱板加熱的 **PWM** 值

風扇的 **PWM** 值

Marlin 內部參數

- **temperature.cpp 的參數**

```
// Init min and max temp with extreme values to prevent false errors during startup
static int minttemp_raw[EXTRUDERS] = ARRAY_BY_EXTRUDERS( HEATER_0_RAW_LO_TEMP , HEATER_1_RAW_LO_TEMP , HEATER_2_RA
static int maxttemp_raw[EXTRUDERS] = ARRAY_BY_EXTRUDERS( HEATER_0_RAW_HI_TEMP , HEATER_1_RAW_HI_TEMP , HEATER_2_RAW
static int minttemp[EXTRUDERS] = ARRAY_BY_EXTRUDERS( 0, 0, 0 );
static int maxttemp[EXTRUDERS] = ARRAY_BY_EXTRUDERS( 16383, 16383, 16383 );
static int bed_minttemp_raw = HEATER_BED_RAW_LO_TEMP; /* No bed mintemp error implemented?!? */
#ifdef BED_MAXTEMP
static int bed_maxttemp_raw = HEATER_BED_RAW_HI_TEMP;
#endif
```

擠出頭及加熱板的高低溫的限制

Marlin 內部參數

- **temperature.cpp 的參數**

```
#ifdef TEMP_SENSOR_1_AS_REDUNDANT
  static void *heater_ttbl_map[2] = {(void *)HEATER_0_TEMPTABLE, (void *)HEATER_1_TEMPTABLE };
  static uint8_t heater_ttbllen_map[2] = { HEATER_0_TEMPTABLE_LEN, HEATER_1_TEMPTABLE_LEN };
#else
  static void *heater_ttbl_map[EXTRUDERS] = ARRAY_BY_EXTRUDERS( (void *)HEATER_0_TEMPTABLE, (void *)HEATER_1_TEMPTABLE, (void *)HE
  static uint8_t heater_ttbllen_map[EXTRUDERS] = ARRAY_BY_EXTRUDERS( HEATER_0_TEMPTABLE_LEN, HEATER_1_TEMPTABLE_LEN, HEATER_
#endif
```

擠出頭及加熱板的 **ADC** 與溫度的對照表

Marlin 內部參數

- **thermistortables.h 的參數**

```
#define _TT_NAME(_N) temptable_ ## _N
#define TT_NAME(_N) _TT_NAME(_N)

#ifdef THERMISTORHEATER_0
# define HEATER_0_TEMPTABLE TT_NAME(THERMISTORHEATER_0)
# define HEATER_0_TEMPTABLE_LEN (sizeof(HEATER_0_TEMPTABLE)/sizeof(*HEATER_0_TEMPTABLE))
#else
# ifdef HEATER_0_USES_THERMISTOR
#  error No heater 0 thermistor table specified
# else //HEATER_0_USES_THERMISTOR
#  define HEATER_0_TEMPTABLE NULL
#  define HEATER_0_TEMPTABLE_LEN 0
# endif //HEATER_0_USES_THERMISTOR
#endif
```

擠出頭的 **ADC** 與溫度的對照表設定

- 擠出頭依編號有各別的設定

Marlin 內部參數

- **thermistortables.h 的參數**

```
#ifdef THERMISTORBED
 # define BEDTEMPTABLE TT_NAME(THERMISTORBED)
 # define BEDTEMPTABLE_LEN (sizeof(BEDTEMPTABLE)/sizeof(*BEDTEMPTABLE))
 #else
# ifdef BED_USES_THERMISTOR
 #  error No bed thermistor table specified
 # endif // BED_USES_THERMISTOR
 #endif

 //Set the high and low raw values for the heater, this indicates which raw value is a high or low temperature
#ifndef HEATER_BED_RAW_HI_TEMP
# ifdef BED_USES_THERMISTOR   //In case of a thermistor the highest temperature results in the lowest ADC
 #  define HEATER_BED_RAW_HI_TEMP 0
 #  define HEATER_BED_RAW_LO_TEMP 16383
 # else                        //In case of an thermocouple the highest temperature results in the highest ADC value
 #  define HEATER_BED_RAW_HI_TEMP 16383
 #  define HEATER_BED_RAW_LO_TEMP 0
 # endif
 #endif
```

加熱板的 **ADC** 與溫度的對照表設定

Marlin 內部參數

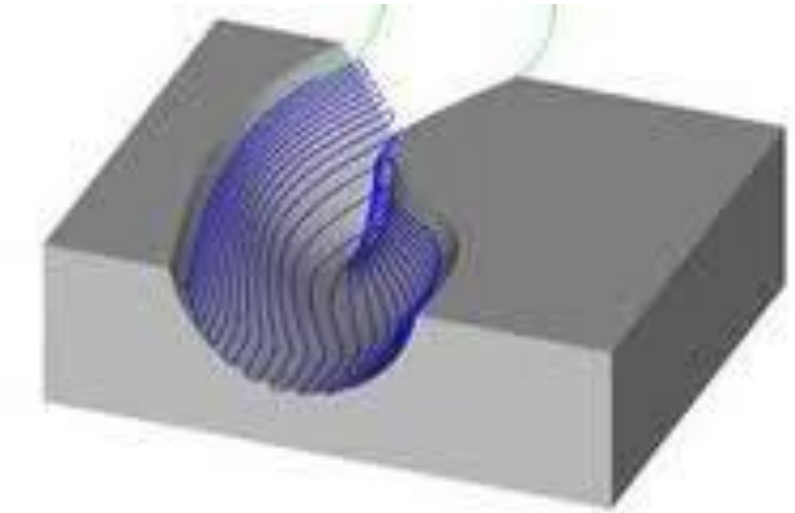
- **thermistortables.h 的參數**
  - 各型號的熱敏電阻 ADC 值與溫度的對照表
  - 可以參考 marlin overview 的投影片

```
#if (THERMISTORHEATER_0 == 1) || (THERMISTORHEATER_1 == 1) || (THERMISTORHEATER_2 == 1) || (THERMISTORBED == 1

const short temptable_1[][2] PROGMEM = {
{    23*OVERSAMPLENR ,    300   },
{    25*OVERSAMPLENR ,    295   },
{    27*OVERSAMPLENR ,    290   },
{    28*OVERSAMPLENR ,    285   },
{    31*OVERSAMPLENR ,    280   },
{    33*OVERSAMPLENR ,    275   },
{    35*OVERSAMPLENR ,    270   },
{    38*OVERSAMPLENR ,    265   },
{    41*OVERSAMPLENR ,    260   },
{    44*OVERSAMPLENR ,    255   },
{    48*OVERSAMPLENR ,    250   },
{    52*OVERSAMPLENR ,    245   },
{    56*OVERSAMPLENR ,    240   },
{    61*OVERSAMPLENR ,    235   },
{    66*OVERSAMPLENR ,    230   },
{    71*OVERSAMPLENR ,    225   },
```

Marlin 路徑規畫的實作

Marlin 路徑規畫的實作

- **Marlin 的路徑規畫有**
  - 直線軌跡的規畫
  - 圓弧軌跡的規畫



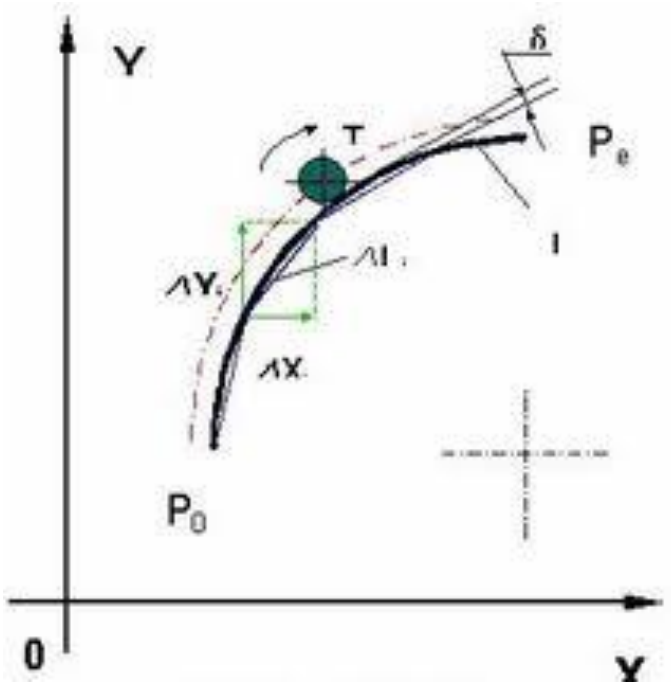
Marlin 路徑規畫的實作

- 直線軌跡是由多個 **G01** 指令規畫出來

G1 X91.020 Y90.230 F900.000 E2.05962
G1 X91.350 Y90.030 E2.09963
G1 X91.690 Y89.860 E2.13905
G1 X92.050 Y89.740 E2.17840
G1 X92.430 Y89.650 E2.21890
G1 X93.000 Y89.600 E2.27823

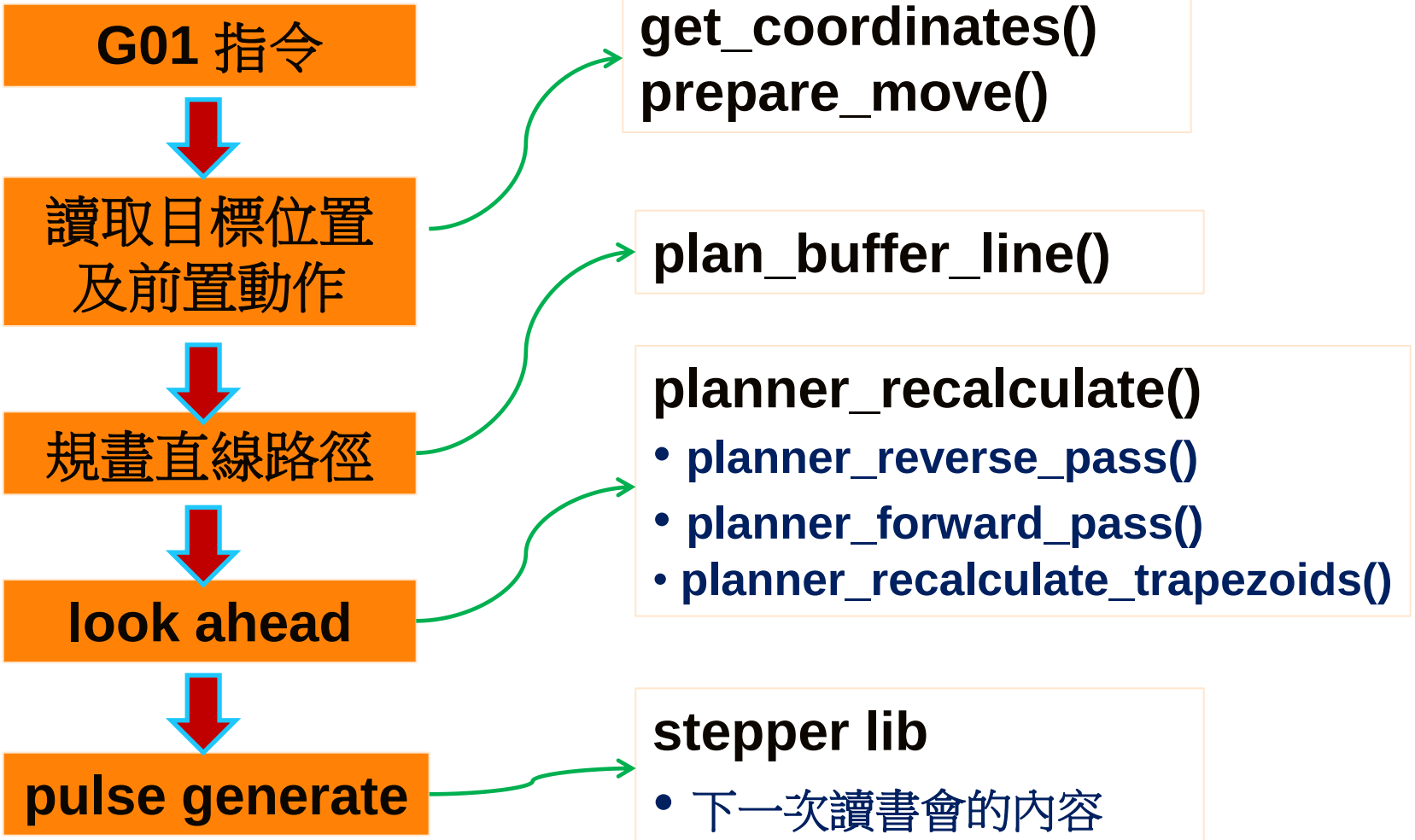
Marlin 路徑規畫的實作

- 圓弧軌跡的指令格式 **G02, G03**
  - 給半徑方式 : **G02(G03) X_Y_R_F_**
  - 給圓心方式 : **G02(G03) X_Y_I_J_F_**
  - 給角度方式 : **G02(G03) I_J_A_F_**



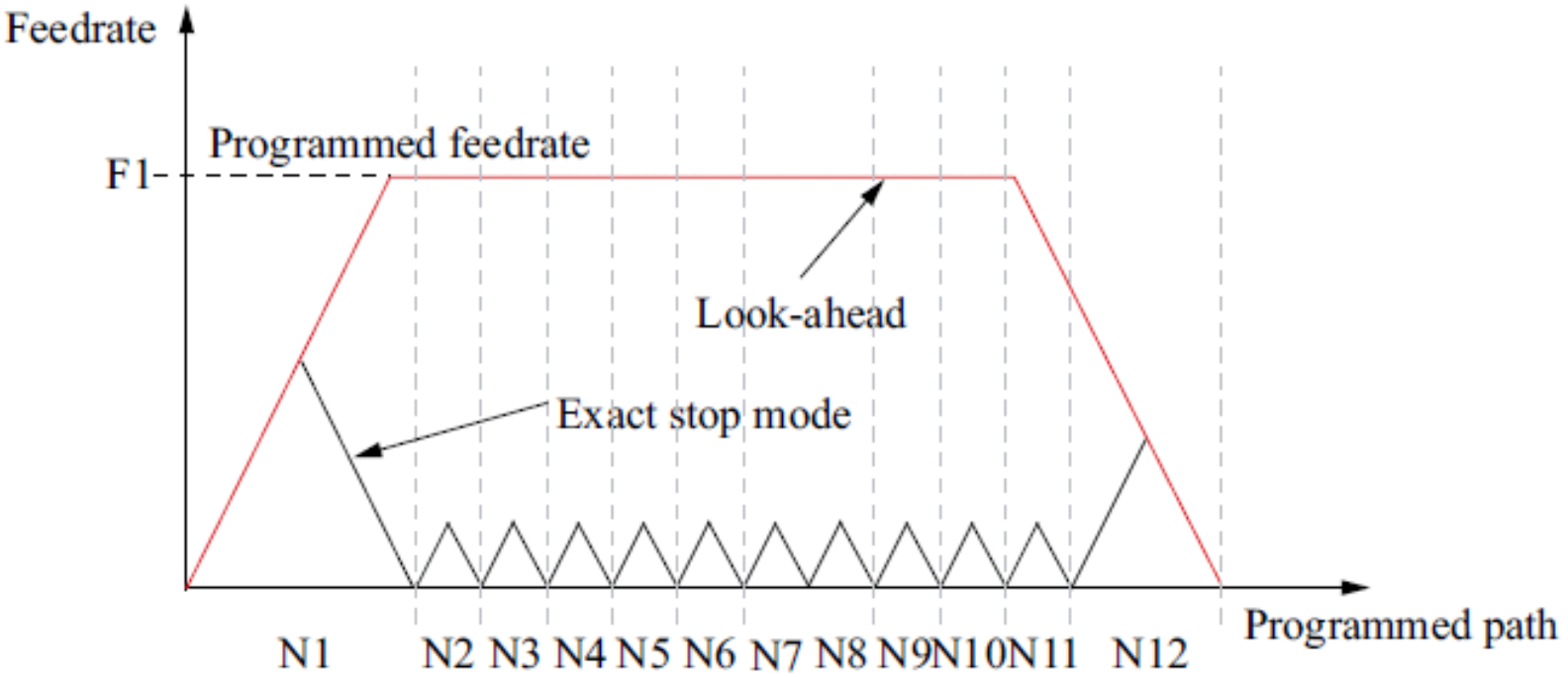
Marlin 直線軌跡的實作

Marlin 直線軌跡規畫流程



Marlin 直線軌跡規畫流程

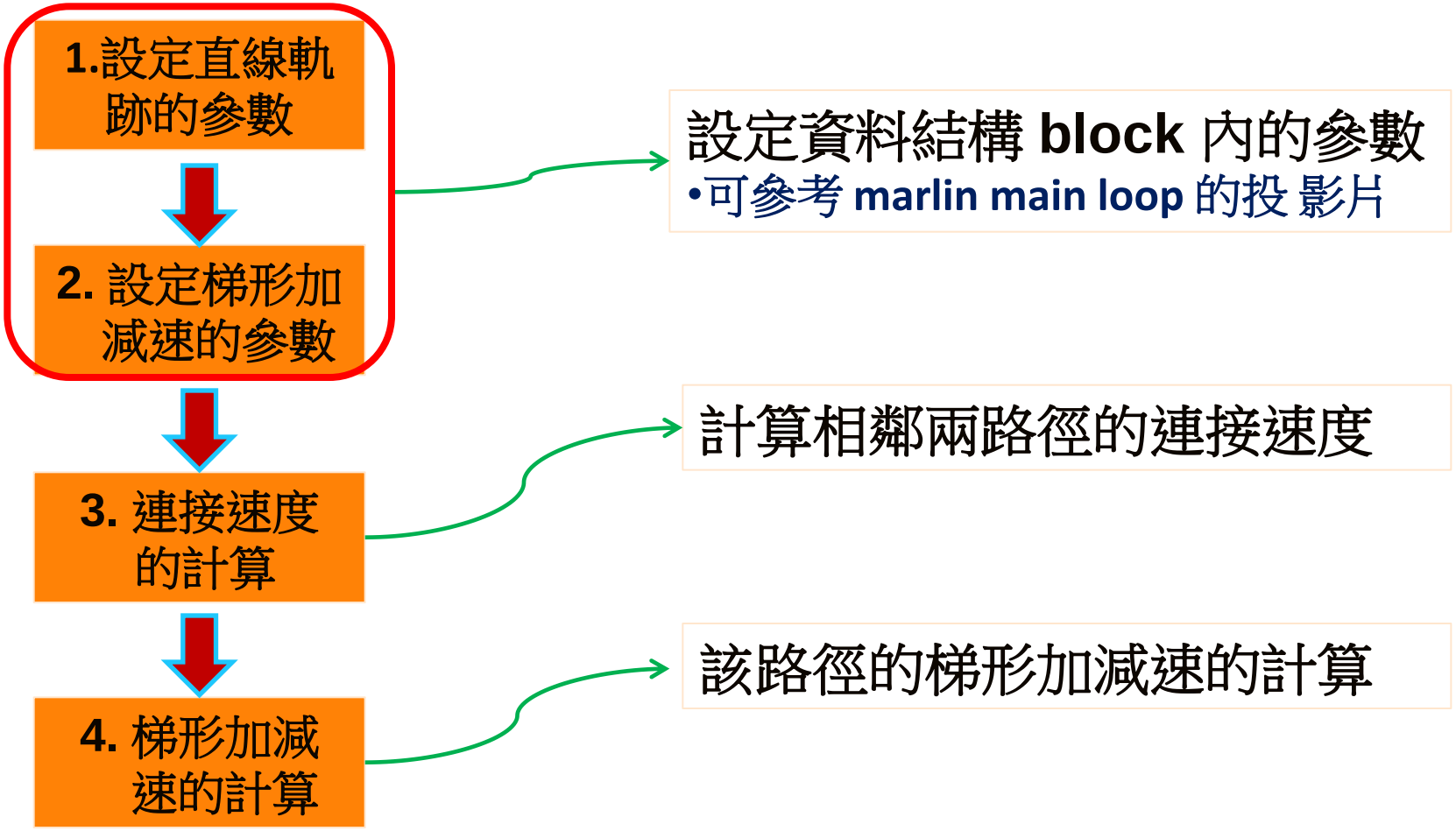
- **Look ahead 的作用**



Marlin 規畫直線軌跡

Marlin 規畫直線軌跡

- plan_buffer_line() 的流程



Marlin 規畫直線軌跡

1. 開始 plan_buffer_line() 的講解

```
float junction_deviation = 0.1;
// Add a new linear movement to the buffer. steps_x, _y and _z is the absolute position in
// mm. Microseconds specify how many microseconds the move should take to perform. To aid acceleration
// calculation the caller must also provide the physical length of the line in millimeters.
void plan_buffer_line(const float &x, const float &y, const float &z, const float &e, float feed_rate, const uint8_t &extruder)
{
  // Calculate the buffer head after we push this byte
  int next_buffer_head = next_block_index(block_buffer_head);

  // If the buffer is full: good! That means we are well ahead of the robot.
  // Rest here until there is room in the buffer.
  while(block_buffer_tail == next_buffer_head)
  {
    manage_heater();
    manage_inactivity();
    lcd_update();
  }
```

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
void plan_buffer_line(const float &x, const float &y, const float &z
{
  // Calculate the buffer head after we push this byte
  int next_buffer_head = next_block_index(block_buffer_head);

  // If the buffer is full: good! That means we are well ahead of the robot.
  // Rest here until there is room in the buffer.
  while(block_buffer_tail == next_buffer_head)
  {
    manage_heater();
    manage_inactivity();
    lcd_update();
  }
```

檢查 **block** 是否有空間，如果沒有空間，則等待並執行

- **manage_heater()**
- **manage_inactivity()**
- **lcd_update()**

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
long target[4];
target[X_AXIS] = lround(x*axis_steps_per_unit[X_AXIS]);
target[Y_AXIS] = lround(y*axis_steps_per_unit[Y_AXIS]);
target[Z_AXIS] = lround(z*axis_steps_per_unit[Z_AXIS]);
target[E_AXIS] = lround(e*axis_steps_per_unit[E_AXIS]);
```

計算各軸目標位置的 **step** 數

```
#ifdef PREVENT_DANGEROUS_EXTRUDE
if(target[E_AXIS]!=position[E_AXIS])
{
  if(degHotend(active_extruder)<extrude_min_temp)
  {
    position[E_AXIS]=target[E_AXIS]; //behave as if the move really took pl
    SERIAL_ECHO_START;
    SERIAL_ECHOLNPGM(MSG_ERR_COLD_EXTRUDE_STOP);
  }
```

當有擠料的動作，且擠出頭的溫度低 **extrude_min_temp** 時，不會執行擠料作為保護

- 可參考 **marlin main loop** 的投影片

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
{
  position[E_AXIS]=target[E_AXIS]; //behave as if the move really took place, but ignore E
  SERIAL_ECHO_START;
  SERIAL_ECHOLNPGM(MSG_ERR_COLD_EXTRUDE_STOP);
}

#ifdef PREVENT_LENGTHY_EXTRUDE
if(labs(target[E_AXIS]-position[E_AXIS])>axis_steps_per_unit[E_AXIS]*EXTR
{
  position[E_AXIS]=target[E_AXIS]; //behave as if the move really took place, but ignore E
  SERIAL_ECHO_START;
  SERIAL_ECHOLNPGM(MSG_ERR_LONG_EXTRUDE_STOP);
}
#endif
}
#endif
```

如果擠料的長度超過 **EXTRUDE_MAXLENGTH** 時，不會執行擠料的動作

- 可參考 **marlin main loop** 的投影片

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
// Prepare to set up new block
block_t *block = &block_buffer[block_buffer_head];

// Mark block as not busy (Not executed by the stepper interrupt)
block->busy = false;
if (block->step_event_count <= dropsegments)
{
  return;
}

block->fan_speed = fanSpeed;

// Compute direction bits for this block
block->direction_bits = 0;
```

設定新的 **block**

- **dropsegment = 5**
- 如果 **step_evemt_count** 低於 **5** 個 **step**，則乎略這個指令

設定風扇的速度

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
if ((target[X_AXIS]-position[X_AXIS]) + (target[Y_AXIS]-position[Y_AXIS]) < 0)
{
  block->direction_bits |= (1<<X_AXIS);
}
if ((target[X_AXIS]-position[X_AXIS]) - (target[Y_AXIS]-position[Y_AXIS]) < 0)
{
  block->direction_bits |= (1<<Y_AXIS);
}
if (target[Z_AXIS] < position[Z_AXIS])
{
  block->direction_bits |= (1<<Z_AXIS);
}
if (target[E_AXIS] < position[E_AXIS])
{
  block->direction_bits |= (1<<E_AXIS);
}
```

設定各軸的移動方向

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
block->active_extruder = extruder;

if(block->steps_x != 0) enable_x();
if(block->steps_y != 0) enable_y();
#ifndef Z_LATE_ENABLE
if(block->steps_z != 0) enable_z();
#endif

// Enable all
if(block->steps_e != 0)
{
  enable_e0();
  enable_e1();
  enable_e2();
}
```

設定擠出頭編號

enable 各軸步進馬達

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
if (block->steps_e == 0)
{
  if(feed_rate<mintravelfeedrate) feed_rate=mintravelfeedrate;
}
else
{
  if(feed_rate<minimumfeedrate) feed_rate=minimumfeedrate;
}
```

設定 **feedrate**

- 如果 **step_e = 0**，則 **feedrate** 最低速限制為 **mintravelfeedrate**
- 如果 **step_e ≠ 0**，則 **feedrate** 最低速限制為 **minimumfeedrate**

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

計算各軸步進馬達要移動的 **step** 數

```
float delta_mm[4];
delta_mm[X_AXIS] = (target[X_AXIS]-position[X_AXIS])/axis_steps_per_unit[X_AXIS];
delta_mm[Y_AXIS] = (target[Y_AXIS]-position[Y_AXIS])/axis_steps_per_unit[Y_AXIS];
delta_mm[Z_AXIS] = (target[Z_AXIS]-position[Z_AXIS])/axis_steps_per_unit[Z_AXIS];
delta_mm[E_AXIS] = ((target[E_AXIS]-position[E_AXIS])/axis_steps_per_unit[E_AXIS])*extrudemultiply/100.0;
```

計算移動的距離

```
if ( block->steps_x <=dropsegments && block->steps_y <=dropsegments && block->steps_z <=dropsegments )
{
  block->millimeters = fabs(delta_mm[E_AXIS]);
}
else
{
  block->millimeters = sqrt(square(delta_mm[X_AXIS]) + square(delta_mm[Y_AXIS]) + square(delta_mm[Z_AXIS]));
}
float inverse_millimeters = 1.0/block->millimeters;  // Inverse millimeters to remove multiple divides
```

Marlin 規畫直線軌跡

1. plan_buffer_line() 的講解

```
float inverse_second = feed_rate * inverse_millimeters;

int moves_queued=(block_buffer_head-block_buffer_tail + BLOCK_BUFF

ifdef SLOWDOWN
// segment time im micro seconds
unsigned long segment_time = lround(1000000.0/inverse_second);
if ((moves_queued > 1) && (moves_queued < (BLOCK_BUFFER_SIZE
{
  if (segment_time < minsegmenttime)
  { // buffer is draining, add extra time.  The amount of time added increases if the buffer is
    inverse_second=1000000.0/(segment_time+lround(2*(minsegmenttime-
    #ifdef XY_FREQUENCY_LIMIT
      segment_time = lround(1000000.0/inverse_second);
    #endif
  }
}
endif
```

計算排入行程的 **block** 數

如果 **block** 數大於 **1** 且小於 **BLOCK_BUFFER_SIZE** 的一半時

- 如果該 **block** 運作的時間小於 **minsegmenttime** 時，會對 **inverse_second** 重新作計算，**inverse_second** 的值會比先前小
- **minsegmenttime** 預設是 **2ms**

Marlin 規畫直線軌跡

2. plan_buffer_line() 的講解

```
block->nominal_speed = block->millimeters * inverse_second; // (mm/sec) Alway
block->nominal_rate = ceil(block->step_event_count * inverse_second); // (step
```

計算 block 的 nominal speed (mm/sec) 及 nominal rate (step/sec)

```
// Calculate and limit speed in mm/sec for each axis
float current_speed[4];
float speed_factor = 1.0; //factor <=1 do decrease speed
for(int i=0; i < 4; i++)
{
  current_speed[i] = delta_mm[i] * inverse_second;
  if(fabs(current_speed[i]) > max_feedrate[i])
    speed_factor = min(speed_factor, max_feedrate[i] / fabs(current_speed[i]));
}
```

計算各軸的 current_speed 及 speed_factor

```
// Correct the speed
if( speed_factor < 1.0)
{
  for(unsigned char i=0; i < 4; i++)
  {
    current_speed[i] *= speed_factor;
  }
  block->nominal_speed *= speed_factor;
  block->nominal_rate *= speed_factor;
}
```

speed_factor < 1 時，表示指令給的 feedrate 超過系統的限制，normal speed 及 normal rate 會重新作計算

Marlin 規畫直線軌跡

2. plan_buffer_line() 的講解

計算 **block** 的 **acceleration_st(step/sqr_sec)**

```
// Compute and limit the acceleration rate for the trapezoid generator.
float steps_per_mm = block->step_event_count/block->millimeters;
if(block->steps_x == 0 && block->steps_y == 0 && block->steps_z == 0)
{
  block->acceleration_st = ceil(retract_acceleration * steps_per_mm); // convert to: acceleration steps/sec^2
}
else
{
  block->acceleration_st = ceil(acceleration * steps_per_mm); // convert to: acceleration steps/sec^2
  // Limit acceleration per axis
  if(((float)block->acceleration_st * (float)block->steps_x / (float)block->step_event_count) > axis_steps_per_sqr_second[X_AXIS])
    block->acceleration_st = axis_steps_per_sqr_second[X_AXIS];
  if(((float)block->acceleration_st * (float)block->steps_y / (float)block->step_event_count) > axis_steps_per_sqr_second[Y_AXIS])
    block->acceleration_st = axis_steps_per_sqr_second[Y_AXIS];
  if(((float)block->acceleration_st * (float)block->steps_e / (float)block->step_event_count) > axis_steps_per_sqr_second[E_AXIS])
    block->acceleration_st = axis_steps_per_sqr_second[E_AXIS];
  if(((float)block->acceleration_st * (float)block->steps_z / (float)block->step_event_count ) > axis_steps_per_sqr_second[Z_AXIS])
    block->acceleration_st = axis_steps_per_sqr_second[Z_AXIS];
}
```

Marlin 規畫直線軌跡

2. plan_buffer_line() 的講解

計算 **block** 的**acceleration (mm/sqr_sec)**

```
    block->acceleration_st = axis_steps_per_sqr_second[Z_AXIS];
  }
  block->acceleration = block->acceleration_st / steps_per_mm;
  block->acceleration_rate = (long)((float)block->acceleration_st * (16777216.0 / (F_CPU / 8.0)));
```

計算**block** 的 **acceleration_rate**

- 作為控制器計算 **step** 生成的加速度值

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

```
#if 0  // Use old jerk for now
 double unit_vec[3];

 unit_vec[X_AXIS] = delta_mm[X_AXIS]*inverse_millimeters;
 unit_vec[Y_AXIS] = delta_mm[Y_AXIS]*inverse_millimeters;
 unit_vec[Z_AXIS] = delta_mm[Z_AXIS]*inverse_millimeters;

 double vmax_junction = MINIMUM_PLANNER_SPEED; // Set default max junction speed

 // Skip first block or when previous_nominal_speed is used as a flag for homing and offset cycles.
 if ((block_buffer_head != block_buffer_tail) && (previous_nominal_speed > 0.0)) {
   // Compute cosine of angle between previous and current path. (prev_unit_vec is negative)
   double cos_theta = - previous_unit_vec[X_AXIS] * unit_vec[X_AXIS]
     - previous_unit_vec[Y_AXIS] * unit_vec[Y_AXIS]
     - previous_unit_vec[Z_AXIS] * unit_vec[Z_AXIS] ;
```

預設不會做連接速度的計算

- **grbl** 會做此計算

計算各軸的移動的單位向量

計算前一 路徑與該路徑的夾角，以 **cos** 表示

- **marlin** 沒有對 **previous_unit_vec[3]** 宣告

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

```
  // Skip and use default max junction speed for 0 degree acute junction.
  if (cos_theta < 0.95) {
    vmax_junction = min(previous_nominal_speed,block->nominal_speed);
    // Skip and avoid divide by zero for straight junctions at 180 degrees. Limit to min() of nominal speeds.
    if (cos_theta > -0.95) {
      // Compute maximum junction velocity based on maximum acceleration and junction deviation
      double sin_theta_d2 = sqrt(0.5*(1.0-cos_theta)); // Trig half angle identity. Always positive.
      vmax_junction = min(vmax_junction,
      sqrt(block->acceleration * junction_deviation * sin_theta_d2/(1.0-sin_theta_d2)) );
    }
  }
#endif
```

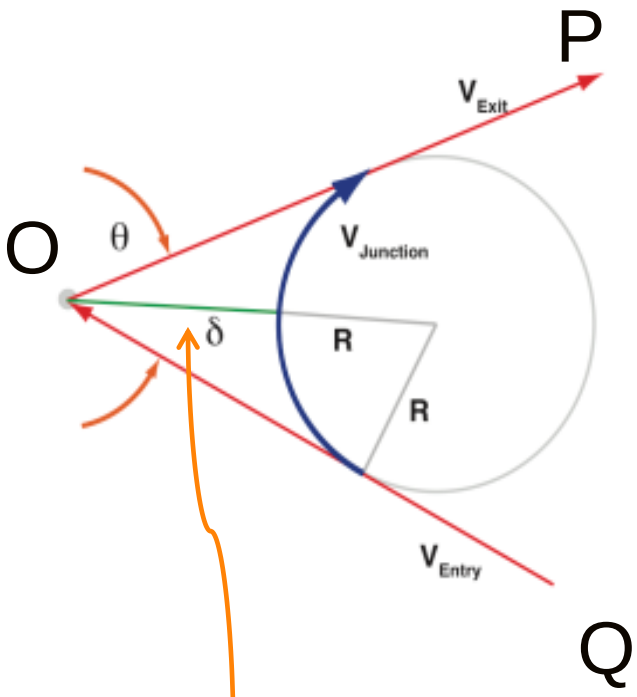
cos 值小於0.95，連接速度為前一路徑與該路徑的nominal speed 較小值

cos 值同時大於 -0.95，會對連接速度作計算(*)

Marlin 規畫直線軌跡

3. 連接速度的計算

將連接速度當作圓周運動的速度，作圓周運動時，角速度為 **ω**, 速度 **v = Rω**, 加速度 **a = Rω^2**。所以 **v =** $\sqrt{aR}$ **(v_juction)**



δ : **junction_deviation**
(Marlin 預設為 0.1mm)

所以只要知道圓周半徑 **R**，即可以得到連接速度

$$\sin\frac{\theta}{2}=\frac{R}{\delta+R} \quad \Rightarrow \quad R=\delta\frac{\sin\frac{\theta}{2}}{1-\sin\frac{\theta}{2}}$$

另外 $\cos\theta=\dfrac{\overrightarrow{OP}\circ\overrightarrow{OQ}}{\left\|\overrightarrow{OP}\right\|\left\|\overrightarrow{OQ}\right\|}$，$\sin\dfrac{\theta}{2}=\pm\sqrt{\dfrac{1-\cos\theta}{2}}$

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

```
float vmax_junction = max_xy_jerk/2;
float vmax_junction_factor = 1.0;
if(fabs(current_speed[Z_AXIS]) > max_z_jerk/2)
  vmax_junction = min(vmax_junction, max_z_jerk/2);
if(fabs(current_speed[E_AXIS]) > max_e_jerk/2)
  vmax_junction = min(vmax_junction, max_e_jerk/2);
vmax_junction = min(vmax_junction, block->nominal_speed);
float safe_speed = vmax_junction;
```

設定 **vmax_junction**

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

– 計算 vmax_juction , vmax_juction_factor，設定 block 的 max_entry_speed

```
if ((moves_queued > 1) && (previous_nominal_speed > 0.0001)) {
  float jerk = sqrt(pow((current_speed[X_AXIS]-previous_speed[X_AXIS]), 2)+pow((current_speed[Y_AXIS]-previous_speed[Y_AXIS]), 2));
  //  if((fabs(previous_speed[X_AXIS]) > 0.0001) || (fabs(previous_speed[Y_AXIS]) > 0.0001)) {
  vmax_junction = block->nominal_speed;
  //  }
  if (jerk > max_xy_jerk) {
    vmax_junction_factor = (max_xy_jerk/jerk);
  }
  if(fabs(current_speed[Z_AXIS] - previous_speed[Z_AXIS]) > max_z_jerk) {
    vmax_junction_factor= min(vmax_junction_factor, (max_z_jerk/fabs(current_speed[Z_AXIS] - previous_speed[Z_AXIS])));
  }
  if(fabs(current_speed[E_AXIS] - previous_speed[E_AXIS]) > max_e_jerk) {
    vmax_junction_factor = min(vmax_junction_factor, (max_e_jerk/fabs(current_speed[E_AXIS] - previous_speed[E_AXIS])));
  }
  vmax_junction = min(previous_nominal_speed, vmax_junction * vmax_junction_factor); // Limit speed to max previous speed
}
block->max_entry_speed = vmax_junction;
```

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

– 計算 **block** 的 **entry_speed**

```
// Initialize block entry speed. Compute based on deceleration to user-defined MINIMUM_PLANNER_SPEED.
double v_allowable = max_allowable_speed(-block->acceleration,MINIMUM_PLANNER_SPEED,block->millimeters);
block->entry_speed = min(vmax_junction, v_allowable);
```

max_allowable_spped function (*) 為

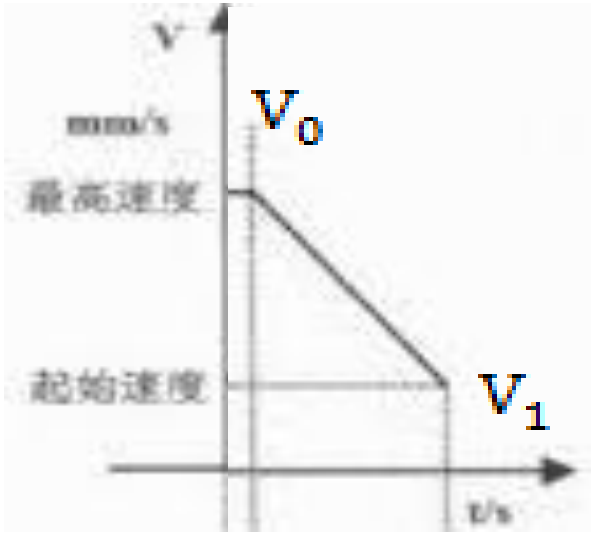
$$\mathbf{v\_allowable} = \sqrt{\mathbf{MINIMUM\_PLANNER\_SPEED}^2 - \ 2 \times (-\mathbf{a}) \times \mathbf{millimeters}}$$

Marlin 規畫直線軌跡

3. plan_buffer_line() 的講解

– 計算 **block** 的 **entry_speed**

在加速度及移動的距離確定時，
entry_speed 所容許最高速的情況是，
做等減速的運動，末速度為
MINIMUM_PLANNER_SPEED



速度的公式： $V_0^2 = V_1^2 - 2 \times (-a) \times d$

所以，可容許最高速為

$$\mathbf{v\_allowable} = \sqrt{\mathbf{MINIMUM\_PLANNER\_SPEED}^2 - 2 \times (-a) \times \mathbf{millimeters}}$$

Marlin 規畫直線軌跡

4. plan_buffer_line() 的講解

```
if (block->nominal_speed <= v_allowable) {
  block->nominal_length_flag = true;
}
else {
  block->nominal_length_flag = false;
}
block->recalculate_flag = true; // Always calculate trapezoid for new block

// Update previous path unit_vector and nominal speed
memcpy(previous_speed, current_speed, sizeof(previous_speed));
previous_nominal_speed = block->nominal_speed;
```

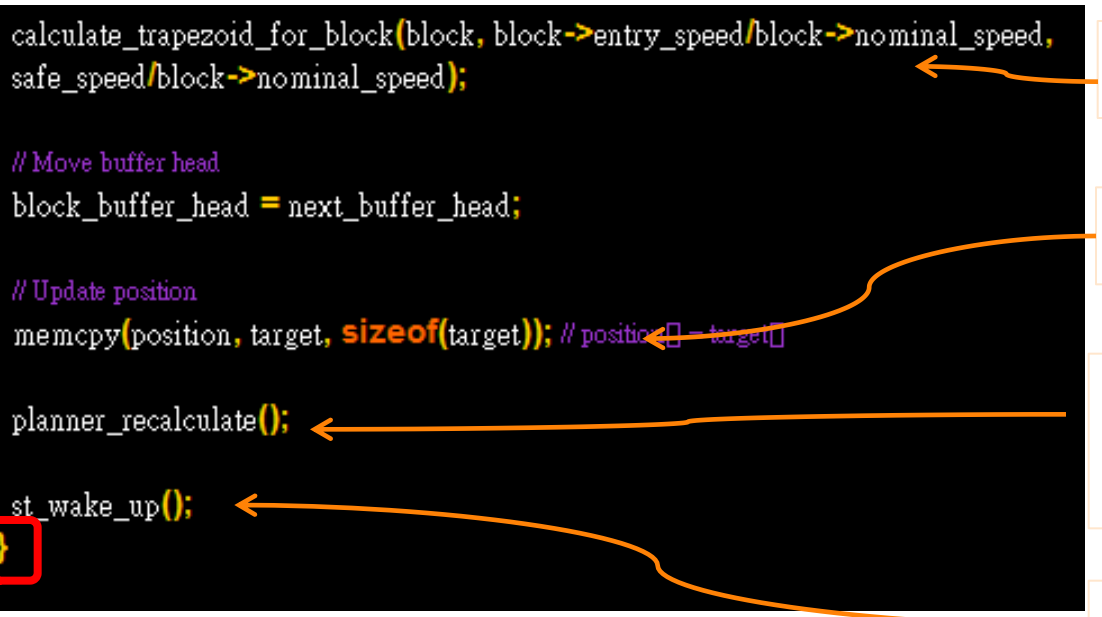
如果 **nominal_speed** 小於 **v_allowable**，在路徑規畫時，**block** 的連接速度會重新計算

設定 **block** 在路徑規畫時，是否需重新計算梯形加減速，**true** 表示需要重新計算

記錄 **block** 速度

Marlin規畫直線軌跡

4. plan_buffer_line() 的講解



計算梯形加減速(*)

記錄目前的位置

重新對所有的 **block** 做軌跡規畫(*)

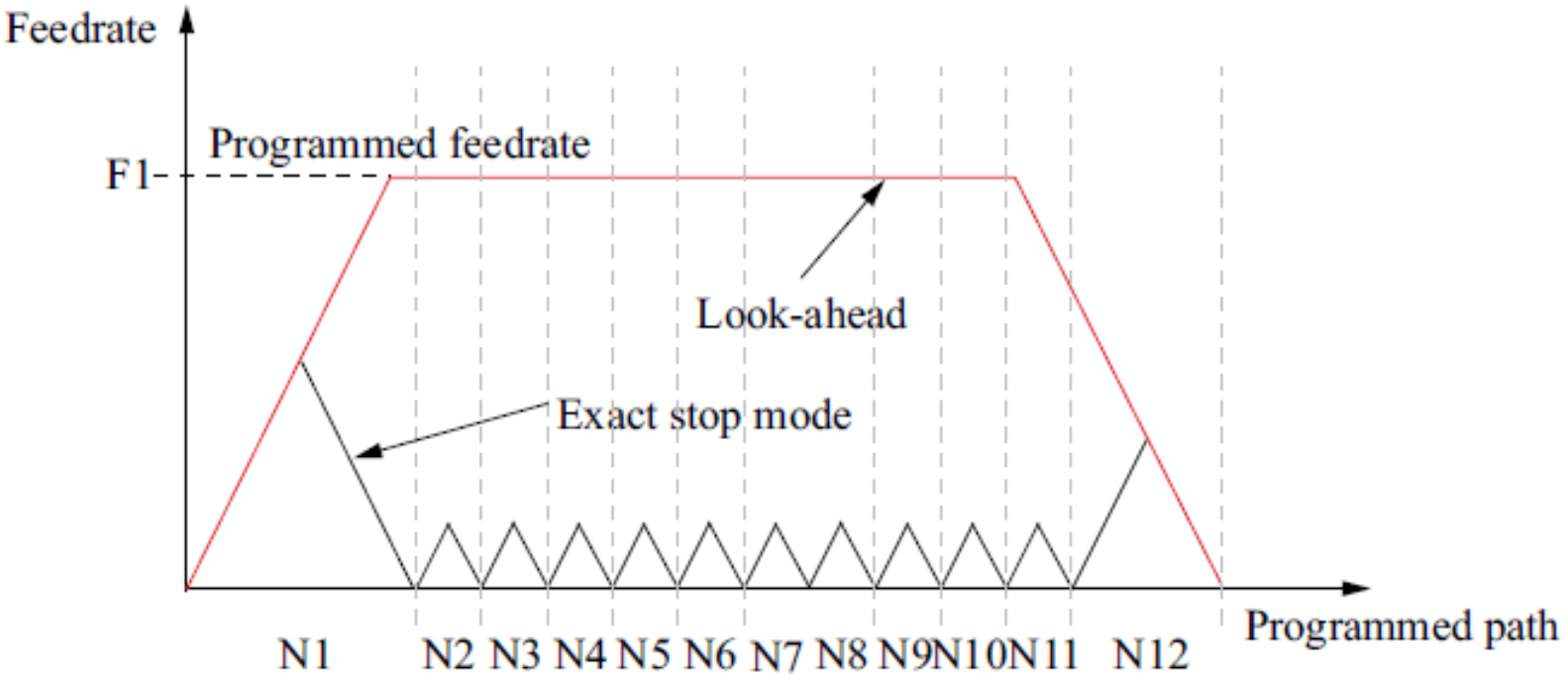
enable stepper interrupt

- **plan_buffer_line function 結束**

Marlin look ahead

Marlin look ahead

- **Look ahead 的作用**



Marlin look ahead

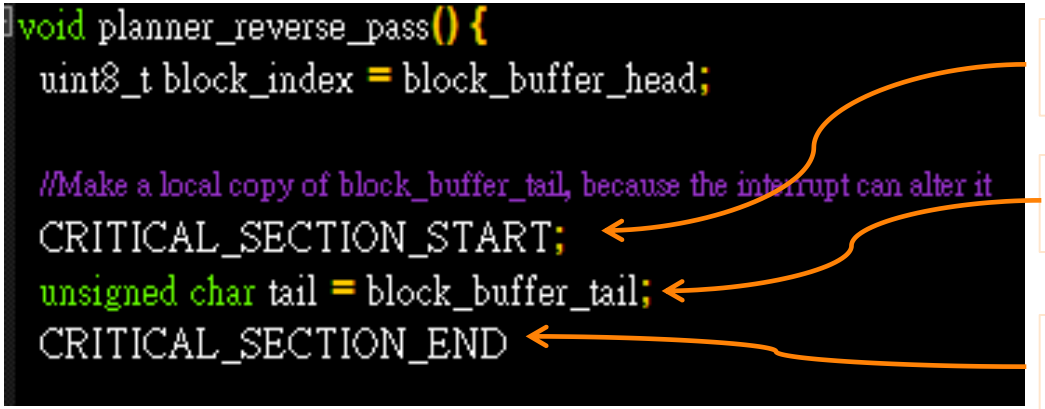
- **planner_recalculate() 的講解**

```
void planner_recalculate() {
  planner_reverse_pass();
  planner_forward_pass();
  planner_recalculate_trapezoids();
}
```

1. 對 block 做反向的的檢查(*)
2. 對 block 做順向的的檢查(*)
3. 重新計算梯 形加減速(*)

Marlin look ahead

1. planner_reverse_pass() 的講解



關掉中斷

讀取 **block** 的結尾

還原中斷設定

Marlin look ahead

1. planner_reverse_pass() 的講解

- 將三三相鄰的 **block call planner_reverse_pass_kernel(*) function** 做連接速度的 檢查

```
if(((block_buffer_head-tail + BLOCK_BUFFER_SIZE) & (BLOCK_BUFFER_SIZE - 1)) > 3) {
  block_index = (block_buffer_head - 3) & (BLOCK_BUFFER_SIZE - 1);
  block_t *block[3] = {
    NULL, NULL, NULL        };
  while(block_index != tail) {
    block_index = prev_block_index(block_index);
    block[2]= block[1];
    block[1]= block[0];
    block[0] = &block_buffer[block_index];
    planner_reverse_pass_kernel(block[0], block[1], block[2]);
  }
 }
}
```

Marlin look ahead

1. planner_reverse_pass() 的講解

– 開始 planner_reverse_pass_kernel function 的講解

```
// The kernel called by planner_recalculate() when scanning the plan from last to first entry.
void planner_reverse_pass_kernel(block_t *previous, block_t *current, block_t *next) {
  if(!current) {
    return;
  }

  if (next) {
    // If entry speed is already at the maximum entry speed, no need to recheck. Block is cruising.
    // If not, block in state of acceleration or deceleration. Reset entry speed to maximum and
    // check for maximum allowable speed reductions to ensure maximum possible planned speed.
```

Block 的檢查

Marlin look ahead

1. planner_reverse_pass() 的講解

– planner_reverse_pass_kernel function 的講解

```
if (current->entry_speed != current->max_entry_speed) {

  // If nominal length true, max junction speed is guaranteed to be reached. Only compute
  // for max allowable speed if block is decelerating and nominal length is false.
  if ((!current->nominal_length_flag) && (current->max_entry_speed > next->entry_speed)) {
    current->entry_speed = min( current->max_entry_speed,
    max_allowable_speed(-current->acceleration,next->entry_speed,current->millimeters));
  }
  else {
    current->entry_speed = current->max_entry_speed;
  }
  current->recalculate_flag = true;

}
// Skip last block. Already initialized and set for recalculation.
```

當 entry speed 未到最高速時

current block 的 entry speed 重新計算

- nominal_length_flag 為 true
- next block 的 entry speed 未到上一個 block 的最高速
- 以next block 的 entry speed 為末速度，計算容許 的最大速度

Marlin look ahead

1. planner_reverse_pass() 的講解

- planner_reverse_pass_kernel function 的講解

```
    else {
      current->entry_speed = current->max_entry_speed;
    }
    current->recalculate_flag = true;

  }
} // Skip last block. Already initialized and set for recalculation.
}

// planner_recalculate() needs to go over the current plan twice. Once in reverse and once forward. This
```

entry speed 更新後，梯形加減速需要重新計算

planner_reverse_pass_kernel function 結束

Marlin look ahead

2. planner_forward_pass() 的講解

– 將三三相鄰的 **block call planner_forward_pass_kernel(*) function** 做連接速度的 檢查

```
void planner_forward_pass() {
  uint8_t block_index = block_buffer_tail;
  block_t *block[3] = {
    NULL, NULL, NULL   };

  while(block_index != block_buffer_head) {
    block[0] = block[1];
    block[1] = block[2];
    block[2] = &block_buffer[block_index];
    planner_forward_pass_kernel(block[0],block[1],block[2]);
    block_index = next_block_index(block_index);
  }
  planner_forward_pass_kernel(block[1], block[2], NULL);
}
```

Marlin look ahead

2. planner_forward_pass() 的講解

– **Planner_forward_pass_kernel function 的講解**

```
// The kernel called by planner_recalculate() when scanning the plan from first to last entry.
void planner_forward_pass_kernel(block_t *previous, block_t *current, block_t *next) {
  if(!previous) {
    return;
  }

  // If the previous block is an acceleration block, but it is not long enough to complete the
  // full speed change within the block, we need to adjust the entry speed accordingly. Entry
  // speeds have already been reset, maximized, and reverse planned by reverse planner.
  // If nominal length is true, max junction speed is guaranteed to be reached. No need to recheck.
  if (!previous->nominal_length_flag) {
    if (previous->entry_speed < current->entry_speed) {
      double entry_speed = min( current->entry_speed
```

Block 的檢查

previous block 的 nominal_length_flag 為 false 時，會重新計算目前的 current block 的 entry spped

Marlin look ahead

2. planner_forward_pass() 的講解

- planner_forward_pass_kernel function 的講解

```
if (!previous->nominal_length_flag) {
  if (previous->entry_speed < current->entry_speed) {
    double entry_speed = min( current->entry_speed,
    max_allowable_speed(-previous->acceleration,previous->entry_speed,previous->millim

    // Check for junction speed change
    if (current->entry_speed != entry_speed) {
      current->entry_speed = entry_speed;
      current->recalculate_flag = true;
    }
  }
}
}
```

重新計算 current block 的 entry speed

1. previous block 的 nominal_length_flag 為 false
2. previous block 的 entry speed 小於 current block 的 entry speed 時
3. 以previous block 的 entry speed 為初速度，計算容許的最大速度
4. 檢查 entry speed
5. 設定 recalculate flag

planner_forward_pass_kernel function 結束

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

```
void planner_recalculate_trapezoids() {
  int8_t block_index = block_buffer_tail;
  block_t *current;
  block_t *next = NULL;

  while(block_index != block_buffer_head) {
    current = next;
    next = &block_buffer[block_index];
    if (current) {
      // Recalculate if current block entry or exit junction speed has changed.
      if (current->recalculate_flag || next->recalculate_flag) {
        // NOTE: Entry and exit factors always > 0 by all previous logic operations.
```

對每個 **block** 作梯形加減速的計算

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

```
if (current) {
  // Recalculate if current block entry or exit junction speed has changed.
  if (current->recalculate_flag || next->recalculate_flag) {
    // NOTE: Entry and exit factors always > 0 by all previous logic operations.
    calculate_trapezoid_for_block(current, current->entry_speed/current->nominal_speed,
    next->entry_speed/current->nominal_speed);
    current->recalculate_flag = false; // Reset current only to ensure next trapezoid is computed
  }
}
block_index = next_block_index( block_index );
}
```

current block 與 next block 的 recalculate flag為 true 時

- 做梯形加減速的計算(*)
- **recalculate flag 設 false**

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

- calculate_trapezoid_for_block function 的講解

```
void calculate_trapezoid_for_block(block_t *block, float entry_factor, float exit_factor) {
  unsigned long initial_rate = ceil(block->nominal_rate*entry_factor); // (step/min)
  unsigned long final_rate = ceil(block->nominal_rate*exit_factor); // (step/min)

  // Limit minimal step rate (Otherwise the timer will overflow.)
  if(initial_rate <120) {
    initial_rate=120;
  }
  if(final_rate < 120) {
    final_rate=120;
  }
```

計算初速度及末速度

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

- calculate_trapezoid_for_block function 的講解

```
long acceleration = block->acceleration_st;
int32_t accelerate_steps =
  ceil(estimate_acceleration_distance(block->initial_rate, block->nominal_rate, acceleration));
int32_t decelerate_steps =
  floor(estimate_acceleration_distance(block->nominal_rate, block->final_rate, -acceleration));

// Calculate the size of Plateau of Nominal Rate.
int32_t plateau_steps = block->step_event_count-accelerate_steps-decelerate_steps;
```

計算加速段的 step 數，減速段的 step 數及 nominal段 的 step 數(*)

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

- calculate_trapezoid_for_block function 的講解
  - estimate_acceleration_distance function 的講解

```
// given acceleration:
FORCE_INLINE float estimate_acceleration_distance(float initial_rate, float target_rate, float acceleration)
{
  if (acceleration!=0) {
    return((target_rate*target_rate-initial_rate*initial_rate)/
      (2.0*acceleration));
  }
  else {
    return 0.0;  // acceleration was 0, set acceleration distance to 0
  }
}
```

$$\text{accelerate\_steps} = \frac{\text{final\_rate}^2 - \text{initial\_rate}^2}{2 \times \text{acceleration\_per\_minute}}$$

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

– calculate_trapezoid_for_block function 的講解

```
// Is the Plateau of Nominal Rate smaller than nothing? That means no cruising, and we will
// have to use intersection_distance() to calculate when to abort acceleration and start braking
// in order to reach the final_rate exactly at the end of this block.
if (plateau_steps < 0) {
  accelerate_steps = ceil(intersection_distance(block->initial_rate, block->final_rate, acceleration, block->step_event_count));
  accelerate_steps = max(accelerate_steps,0); // Check limits due to numerical round-off
  accelerate_steps = min((uint32_t)accelerate_steps,block->step_event_count);//(We can cast here to unsigned, because the above line ensu
  plateau_steps = 0;
}
```

當 **nominal** 段的 **step** 數 **< 0** 時，重新計算加速段的 **step (*)**

- 梯形加減速的規畫退化為三角形加減速
- 即 **nominal** 段的 **step** 數等於 **0**

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

- calculate_trapezoid_for_block function 的講解
  - intersection_distance function 的講解(*)

```
FORCE_INLINE float intersection_distance(float initial_rate, float final_rate, float acceleration, float distance)
{
 if (acceleration!=0) {
  return((2.0*acceleration*distance-initial_rate*initial_rate+final_rate*final_rate)/
    (4.0*acceleration) );
 }
 else {
   return 0.0;  // acceleration was 0, set intersection distance to 0
 }
}
```

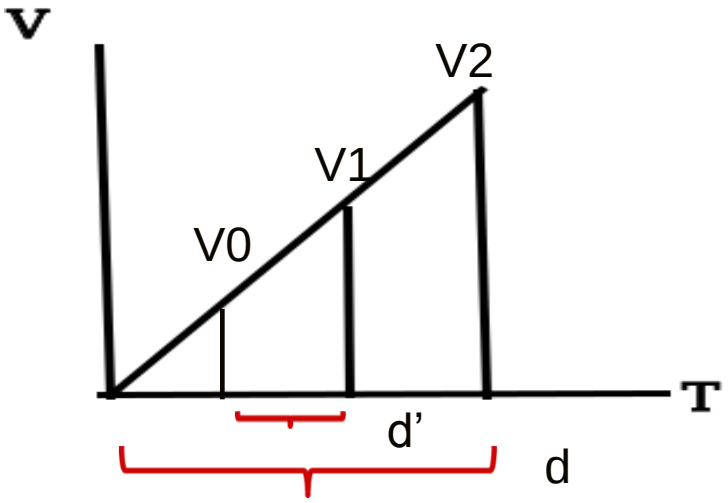
$$\text{accelerate\_steps} = \frac{2 \times \text{acceleration\_per\_minute} \times \text{step\_event\_count} - \text{initial\_rate}^2 + \text{final\_rate}^2}{4 * \text{acceleration\_per\_minute}}$$

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

- calculate_trapezoid_for_block function 的講解
  - intersection_distance function 的講解

原來的規畫是加速到**V2** 的速度，現在改為加速到 **V1** 的位置。在確定 **V0, V2**，以及 **d** 的情況下，同時不知道 **V1** 時，要計算 **d'**，即 **accelerate_steps** 數



利用速度公式：

$$V_1^2 = V_0^2 + 2 \times a \times d'$$

$$V_1^2 = V_2^2 - 2 \times a \times (d - d')$$

$$\Rightarrow 4ad' = V_0^2 + V_2^2 + 2ad$$

Marlin look ahead

3. planner_recalculate_trapezoids() 的講解

– calculate_trapezoid_for_block function 的講解

```
// block->decelerate_after = accelerate_steps+plateau_steps;
CRITICAL_SECTION_START;  // Fill variables used by the stepper in a critical section
if(block->busy == false) { // Don't update variables if block is busy.
  block->accelerate_until = accelerate_steps;
  block->decelerate_after = accelerate_steps+plateau_steps;
  block->initial_rate = initial_rate;
  block->final_rate = final_rate;
}
CRITICAL_SECTION_END;
}
```

關掉中斷

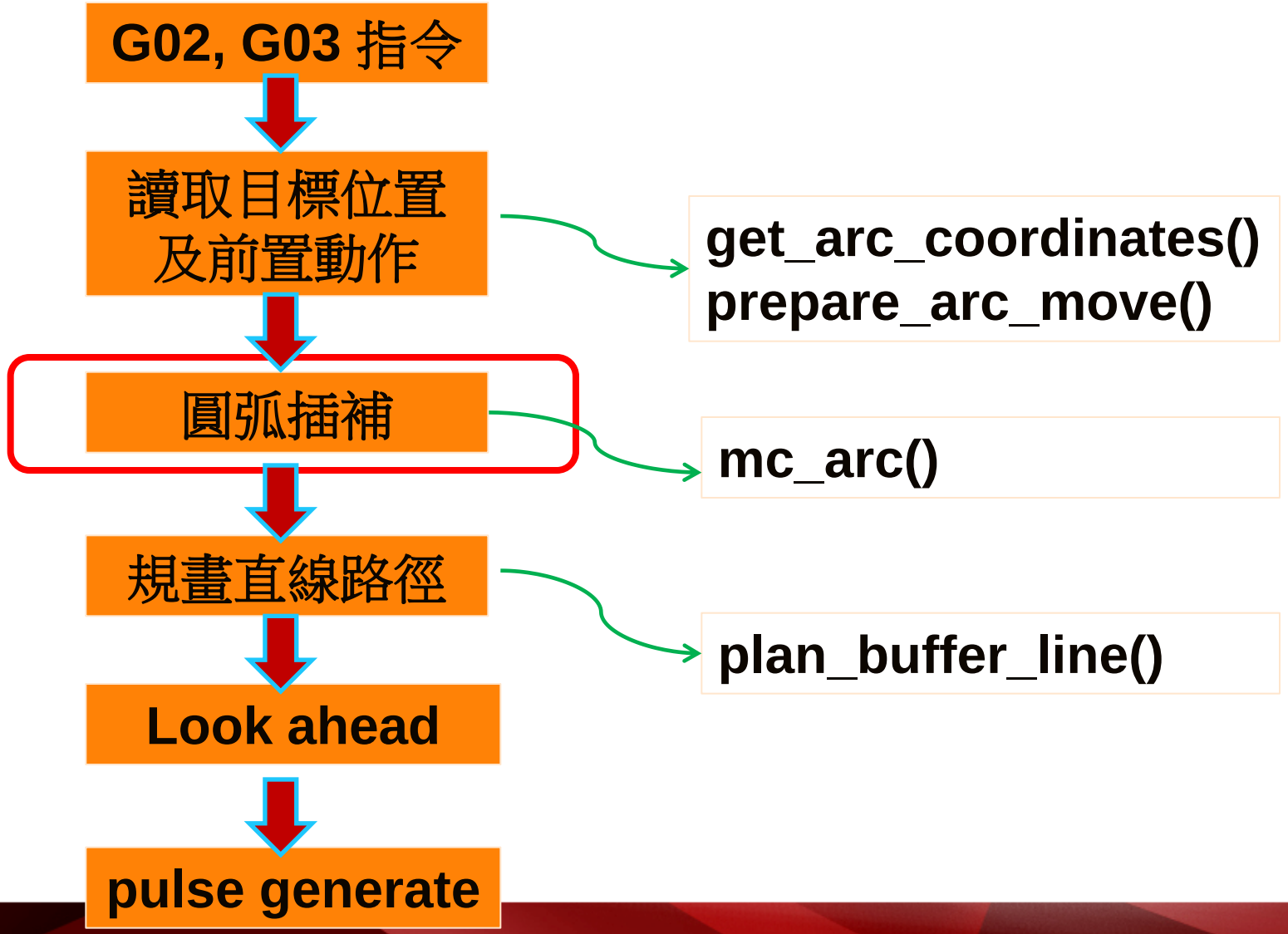
設定加速段的 step 數、減速段的 step 數、nominal 段的 step 數、 初速度及末速度

還原中斷設定

calculate_trapezoid_for_block function 結束

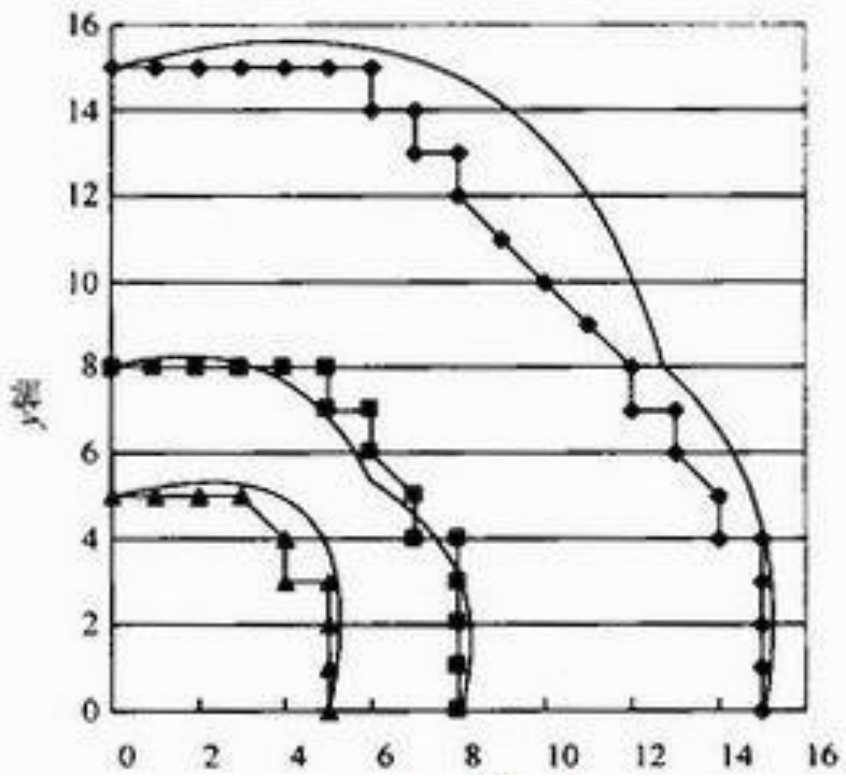
Marlin 圓弧軌跡的實作

Marlin 圓弧軌跡規畫流程



Marlin 圓弧軌跡規畫流程

- 圓弧插補
  - 將圓弧細分為多個小線段(segment)
  - Marlin 預設是 1mm 為一段



Marlin 圓弧規畫的前置動作

Marlin 圓弧軌跡

- G2 指令的解譯 (marlin_main.cpp)

```
case 2: // G2 - CW ARC
  if(Stopped == false) {
    get_arc_coordinates();
    prepare_arc_move(true);
    return;
  }
case 3: // G3 - CCW ARC
  if(Stopped == false) {
    get_arc_coordinates();
    prepare_arc_move(false);
    return;
  }
```

開始做圓弧軌跡的規畫

Marlin 圓弧軌跡

- **prepare_arc_move() 的講解 (marlin_main.cpp)**

```
void prepare_arc_move(char isclockwise) {
  float r = hypot(offset[X_AXIS], offset[Y_AXIS]); // Compute arc radius for mc_arc

  // Trace the arc
  mc_arc(current_position, destination, offset, X_AXIS, Y_AXIS, Z_AXIS, feedr

  // As far as the parser is concerned, the position is now == target. In reality the
  // motion control system might still be processing the action and the real tool position
  // in any intermediate location.
  for(int8_t i=0; i < NUM_AXIS; i++) {
    current_position[i] = destination[i];
  }
  previous_millis_cmd = millis();
}
```

計算圓弧半徑

規畫圓弧軸跡

Marlin 圓弧插補

Marlin 圓弧插補

- 開始 mc_arc() 的講解 (motion_control.cpp)
  - G2 X1.75 Y-0.125 I0 J-0.375
  - 圓弧軸跡的第三軸

```
// The arc is approximated by generating a huge number of tiny, linear segments. The length of each
// segment is configured in settings.mm_per_arc_segment
void mc_arc(float *position, float *target, float *offset, uint8_t axis_0, uint8_t axis_1,
  uint8_t axis_linear, float feed_rate, float radius, uint8_t isclockwise, uint8_t extruder)
{
  //  int acceleration_manager_was_enabled = plan_is_acceleration_manager_enabled();
  //  plan_set_acceleration_manager_enabled(false); // disable acceleration management for the duration of the arc
  float center_axis0 = position[axis_0] + offset[axis_0];
  float center_axis1 = position[axis_1] + offset[axis_1];
  float linear_travel = target[axis_linear] - position[axis_linear];
  float extruder_travel = target[E_AXIS] - position[E_AXIS];
  float r_axis0 = -offset[axis_0];  // Radius vector from center to current location
```

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**

```
//  plan_set_acceleration_manager_enabled(false); // disable acceleration management fo
float center_axis0 = position[axis_0] + offset[axis_0];
float center_axis1 = position[axis_1] + offset[axis_1];
float linear_travel = target[axis_linear] - position[axis_linear];
float extruder_travel = target[E_AXIS] - position[E_AXIS];
float r_axis0 = -offset[axis_0];  // Radius vector from center to current location
float r_axis1 = -offset[axis_1];
float rt_axis0 = target[axis_0] - center_axis0;
float rt_axis1 = target[axis_1] - center_axis1;
```

設定圓心的位置

第三個軸移動的距離及擠出頭擠料的長度

圓心到起始點的向量

圓心到終點的向量

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**

```
// CCW angle between position and target from circle center. Only one atan2() trig computation required.
float angular_travel = atan2(r_axis0*rt_axis1-r_axis1*rt_axis0, r_axis0*rt_axis0+r_axis1*rt_axis1);
if (angular_travel < 0) { angular_travel += 2*M_PI; }
if (isclockwise) { angular_travel -= 2*M_PI; }

float millimeters_of_travel = hypot(angular_travel*radius, fabs(linear_travel));
if (millimeters_of_travel < 0.001) { return; }
uint16_t segments = floor(millimeters_of_travel/MM_PER_ARC_SEGMENT);
if(segments == 0) segments = 1;
```

計算圓弧的夾角

計算圓弧軸跡移動的距離

計算圓弧細分的段數

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**

```
float theta_per_segment = angular_travel/segments;
float linear_per_segment = linear_travel/segments;
float extruder_per_segment = extruder_travel/segments;

/* Vector rotation by transformation matrix: r is the original vector, r_T is
  and phi is the angle of rotation. Based on the solution approach by Jens
    r_T = [cos(phi) -sin(phi);
           sin(phi)  cos(phi] * r ;

  For arc generation, the center of the circle is the axis of rotation and the r
```

計算圓弧每段的角度

計算第三軸每段移動的距離

計算擠出頭每段擠出的長度

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**

```
// Vector rotation matrix values
float cos_T = 1-0.5*theta_per_segment*theta_per_segment; //
float sin_T = theta_per_segment;

float arc_target[4];
float sin_Ti;
float cos_Ti;
float r_axisi;
uint16_t i;
int8_t count = 0;

// Initialize the linear axis
arc_target[axis_linear] = position[axis_linear];

// Initialize the extruder axis
arc_target[E_AXIS] = position[E_AXIS];
```

用 **Taylor approximation**，
計算圓弧每段的 **cos** 及 **sin** 值，

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**

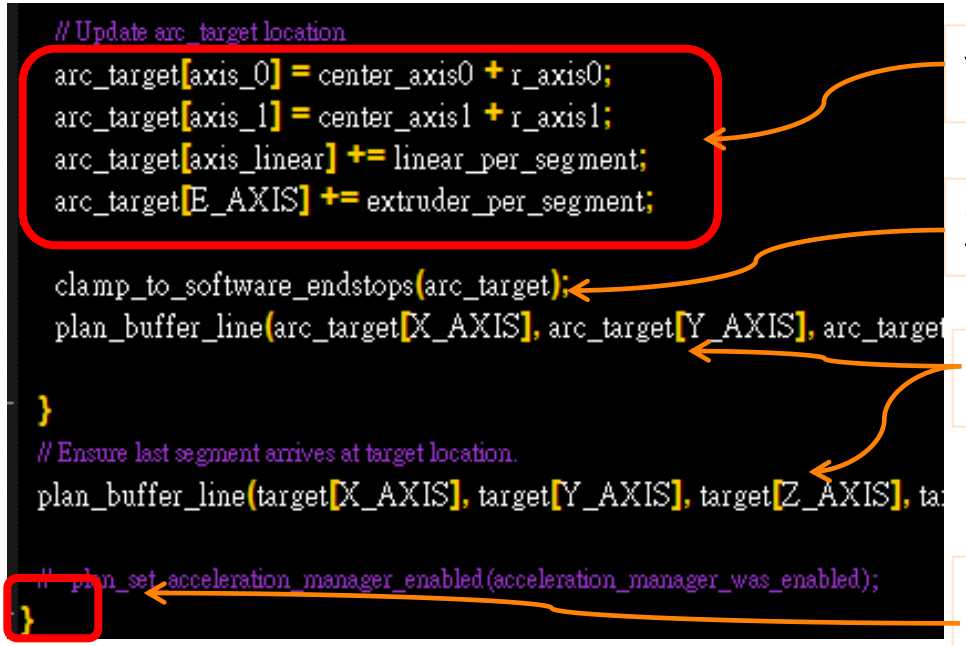
```
for (i = 1; i<segments; i++) { // Increment (segments-1)

  if (count < N_ARC_CORRECTION) {
    // Apply vector rotation matrix
    r_axisi = r_axis0*sin_T + r_axis1*cos_T;
    r_axis0 = r_axis0*cos_T - r_axis1*sin_T;
    r_axis1 = r_axisi;
    count++;
  } else {
    // Arc correction to radius vector. Computed only every N_ARC_COR
    // Compute exact location by applying transformation matrix from initi
    cos_Ti = cos(i*theta_per_segment);
    sin_Ti = sin(i*theta_per_segment);
    r_axis0 = -offset[axis_0]*cos_Ti + offset[axis_1]*sin_Ti;
    r_axis1 = -offset[axis_0]*sin_Ti - offset[axis_1]*cos_Ti;
    count = 0;
  }
```

在容許的誤差範圍下 **(N_ARC_CORRECTION)**，利用旋轉矩陣，估計圓弧該段的向量

超過容許的誤差，會重新計算該段的 **cos** 及**sin** 值，再計算該段的向量

Marlin 圓弧插補

- **mc_arc() 的講解 (motion_control.cpp)**



設定該段的目標位置

檢查目標位置是否在工作空間內

做直線軌跡的規畫

mc_arc function 結束

Marlin Stepper

3D Printer 韌體原始碼閱讀心得 (IV)

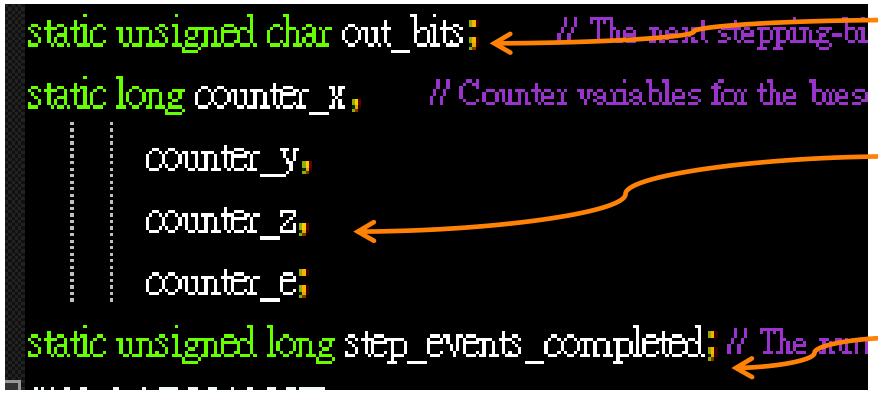
Outline

- **Marlin 的內部參數介紹 (三)**
- **Marlin step generate 的實作**
- **Marlin 溫度控制的實作**
- **Marlin 溫度感測**

Marlin 內部參數介紹(三)

Marlin 內部參數

- **Stepper.h 的參數**



XYZE 四軸的方向參數

XYZE 四軸的計數 step 的參數

該block 的 step 數 (即 XYZE 四軸最大 step 數)

Marlin 內部參數

- **Stepper.h 的參數**

```
static long acceleration_time, deceleration_time;
//static unsigned long accelerate_until, decelerate_after, acceleration_rate, initi
static unsigned short acc_step_rate; //needed for deccelaration start point
static char step_loops;
static unsigned short OCR1A_nominal;
static unsigned short step_loops_nominal;

volatile long endstops_trigsteps[3]={0,0,0};
volatile long endstops_stepsTotal,endstops_stepsDone;
static volatile bool endstop_x_hit=false;
static volatile bool endstop_y_hit=false;
static volatile bool endstop_z_hit=false;
```

記錄步進馬達加速度段的速度

該 **block** 在 **nominal** 段的速度

XYZ 軸 endstop 的狀態

Marlin 內部參數

- **Stepper.h 的參數**

```
volatile long count_position[NUM_AXIS] = {0, 0, 0, 0};
volatile signed char count_direction[NUM_AXIS] = {1, 1, 1, 1};
```

XYZ 軸目前的位置及方向

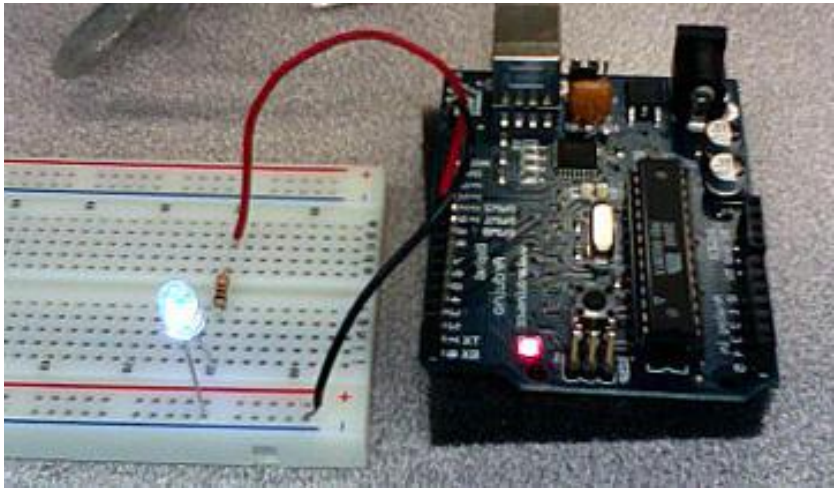
```
#define ENABLE_STEPPER_DRIVER_INTERRUPT()  TIMSK1 |= (1<<OCIE1A)
#define DISABLE_STEPPER_DRIVER_INTERRUPT() TIMSK1 &= ~(1<<OCIE1A)
```

Stepper 中斷的 enable 及 diable

Marlin step generate 的實作

Marlin step generate 的實作

- **Marlin 是利用中斷的方式來實作**
  - 使用 **Arduino (Atmega) 的 16-bit timer1 中斷**
  - 在 **ISR (Interrupt Service Routine) 內執行 step generate**



Marlin step generate 的實作

- **Arduino (Atmega) timer1 的設定**
  - **Clock select bit (timer divider)**

TCCR1B

Bit	7	6	5	4	3	2	1	0	TCCR1B
0×81	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/w	
Initial Value	0	0	0	0	0	0	0	0	

The most important settings are the last three bits in TCCR1B, CS12, CS11, and CS10. These determine the timer clock setting. By setting these bits in various combinations, you can tell the timer to run at different speeds. Here's the relevant table from the datasheet:

Clock Select bit description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From Prescaler)
0	1	1	$clk_{I/O}$/64 (From Prescaler)
1	0	0	$clk_{I/O}$/256 (From Prescaler)
1	0	1	$clk_{I/O}$/1024 (From Prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

Marlin step generate 的實作

- **Arduino (Atmega) timer1 的設定**
  - **timer1 單位時間的設定與計算**

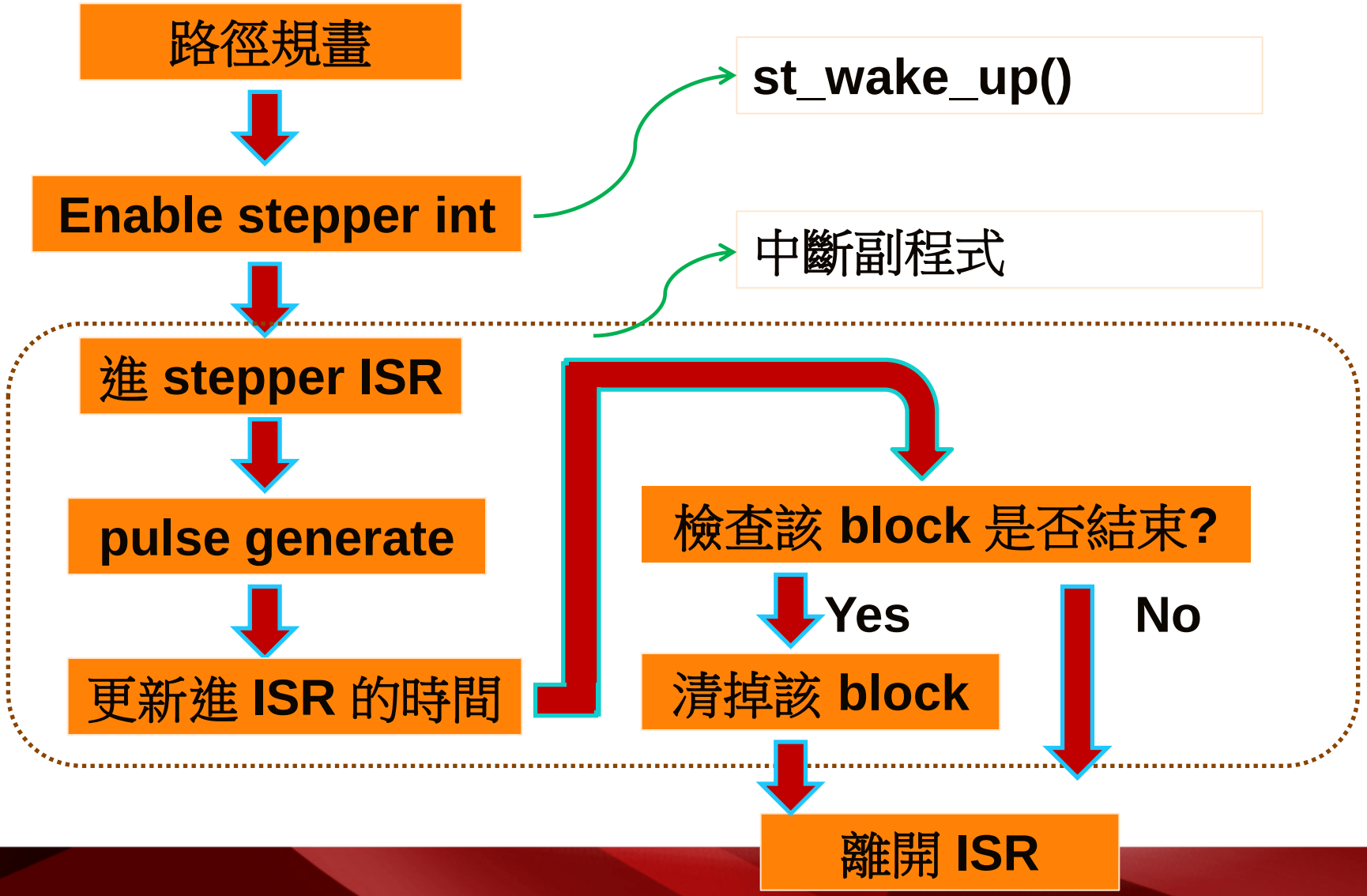
`TCCR1B |= (1 << CS10);` and `TCCR1B |= (1 << CS12);`, the clock source is divided by 1024. This gives a timer resolution of 1/(16*10^6 / 1024), or 6.4e-5 seconds. Now the timer will overflow every (65535 * 6.4e-5s), or 4.194s. That however is still too long.

```
(target time) = (timer resolution) * (# timer counts + 1)
```

and rearrange to get

```
(# timer counts + 1) = (target time) / (timer resolution)
(# timer counts + 1) = (1 s) / (6.4e-5 s)
(# timer counts + 1) = 15625
(# timer counts) = 15625 - 1 = 15624
```

Marlin step generate 的流程



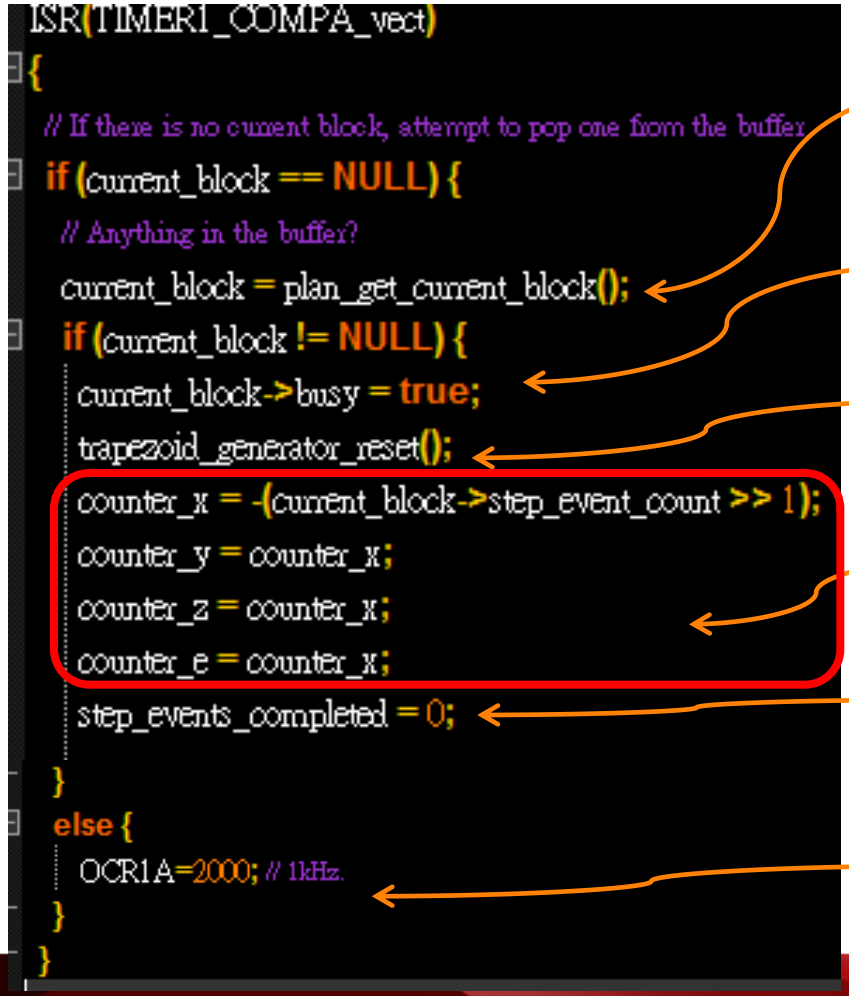
Marlin step generate 的實作

- 開始 ISR() 的講解 (stepper.cpp)

```
// "The Stepper Driver Interrupt" - This timer interrupt is the workhorse.
// It pops blocks from the block_buffer and executes them by pulsing the stepper pins appropiately.
ISR(TIMER1_COMPA_vect)
{
 // If there is no current block, attempt to pop one from the buffer
 if (current_block == NULL) {
  // Anything in the buffer?
  current_block = plan_get_current_block();
  if (current_block != NULL) {
    current_block->busy = true;
```

Marlin step generate 的實作

- **ISR() 的講解**



```
ISR(TIMER1_COMPA_vect)
{
  // If there is no current block, attempt to pop one from the buffer
  if (current_block == NULL) {
    // Anything in the buffer?
    current_block = plan_get_current_block();
    if (current_block != NULL) {
      current_block->busy = true;
      trapezoid_generator_reset();
      counter_x = -(current_block->step_event_count >> 1);
      counter_y = counter_x;
      counter_z = counter_x;
      counter_e = counter_x;
      step_events_completed = 0;
    }
    else {
      OCR1A=2000; // 1kHz.
    }
  }
```

載入新的 **block**

設定 **block** 的狀態

trapezoid_generator_reset()

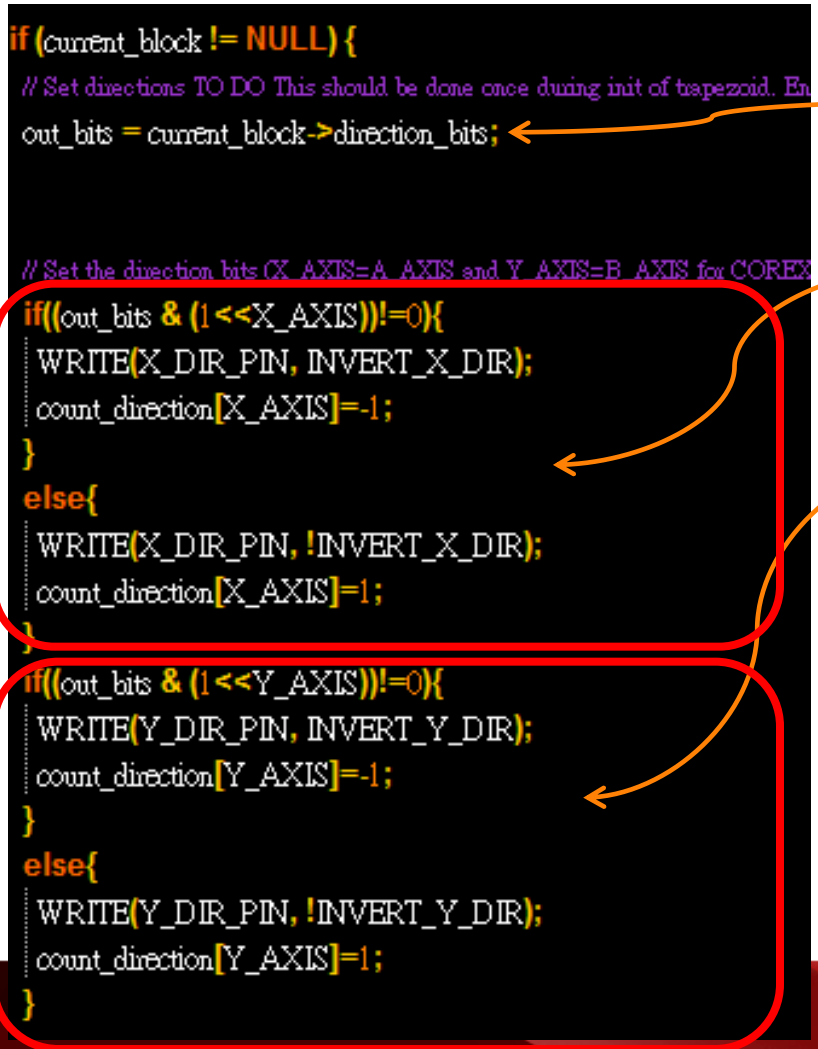
設定 **XYZE** 軸 **step** 的計數

該 **block** 的完成度

如果 **block** 為空時，設定下次中斷的時間

Marlin step generate 的實作

- **ISR() 的講解**



讀取 **XYZE** 各軸步進馬達的方向

設定 **X** 軸的方向

設定 **Y** 軸的方向

Marlin step generate 的實作

- **ISR() 的講解**

```
if ((out_bits & (1<<X_AXIS)) != 0) {  // stepping along -X axis
  CHECK_ENDSTOPS
  {
    #if defined(X_MIN_PIN) && X_MIN_PIN > -1
      bool x_min_endstop=(READ(X_MIN_PIN) != X_ENDSTOPS_INVERTING);
      if(x_min_endstop && old_x_min_endstop && (current_block->steps_x > 0)) {
        endstops_trigsteps[X_AXIS] = count_position[X_AXIS];
        endstop_x_hit=true;
        step_events_completed = current_block->step_event_count;
      }
      old_x_min_endstop = x_min_endstop;
    #endif
  }
}
```

步進馬達 往 **X** 軸 **min** 方向移動

當 **X** 軸 **endstop** 被觸動時

- 記錄當時 **X** 軸的位置
- 設定 **X** 軸 **endstop** 的狀態
- 設定該 **block** 已完成

Marlin step generate 的實作

- **ISR() 的講解**

```
else { // +direction
  CHECK_ENDSTOPS
  {
    #if defined(X_MAX_PIN) && X_MAX_PIN > -1
      bool x_max_endstop=(READ(X_MAX_PIN) != X_ENDSTOPS_INVERTING);
      if(x_max_endstop && old_x_max_endstop && (current_block->steps_x > 0)){
        endstops_trigsteps[X_AXIS] = count_position[X_AXIS];
        endstop_x_hit=true;
        step_events_completed = current_block->step_event_count;
      }
      old_x_max_endstop = x_max_endstop;
    #endif
  }
}
```

步進馬達 往 X 軸 max 方向移動

當 X 軸 endstop 被觸動時
- 記錄當時 X 軸的位置
- 設定 X 軸 endstop 的狀態
- 設定該 block 已完成

檢查Y 軸的 endstop 做法與 X 軸相同

Marlin step generate 的實作

- **ISR() 的講解**

```
if ((out_bits & (1<<Z_AXIS)) != 0) {  // -direction
  WRITE(Z_DIR_PIN,INVERT_Z_DIR);

  #ifdef Z_DUAL_STEPPER_DRIVERS
   WRITE(Z2_DIR_PIN,INVERT_Z_DIR);
  #endif

  count_direction[Z_AXIS]=-1;
  CHECK_ENDSTOPS
  {
    #if defined(Z_MIN_PIN) && Z_MIN_PIN > -1
      bool z_min_endstop=(READ(Z_MIN_PIN) != Z_ENDSTOPS_INVERTING);
      if(z_min_endstop && old_z_min_endstop && (current_block->steps_z > 0)) {
        endstops_trigsteps[Z_AXIS] = count_position[Z_AXIS];
        endstop_z_hit=true;
        step_events_completed = current_block->step_event_count;
      }
      old_z_min_endstop = z_min_endstop;
    #endif
  }
```

設定 Z 軸的往 min 的方向

- Z_DUAL_STEPPER_DRIVERS 表示 Z 軸是用兩個 stepper driver

當 Z 軸 endstop 被觸動時

- 記錄當時 Z 軸的位置
- 設定 Z 軸 endstop 的狀態
- 設定該 block 已完成

Marlin step generate 的實作

- **ISR() 的講解**

```
else {
  WRITE(Z_DIR_PIN,!INVERT_Z_DIR);

  #ifdef Z_DUAL_STEPPER_DRIVERS
    WRITE(Z2_DIR_PIN,!INVERT_Z_DIR);
  #endif

  count_direction[Z_AXIS]=1;
  CHECK_ENDSTOPS
  {
    #if defined(Z_MAX_PIN) && Z_MAX_PIN > -1
      bool z_max_endstop=(READ(Z_MAX_PIN) != Z_ENDSTOPS_INVERTING);
      if(z_max_endstop && old_z_max_endstop && (current_block->steps_z > 0)) {
        endstops_trigsteps[Z_AXIS] = count_position[Z_AXIS];
        endstop_z_hit=true;
        step_events_completed = current_block->step_event_count;
      }
      old_z_max_endstop = z_max_endstop;
    #endif
```

設定 Z 軸的往 max 的方向

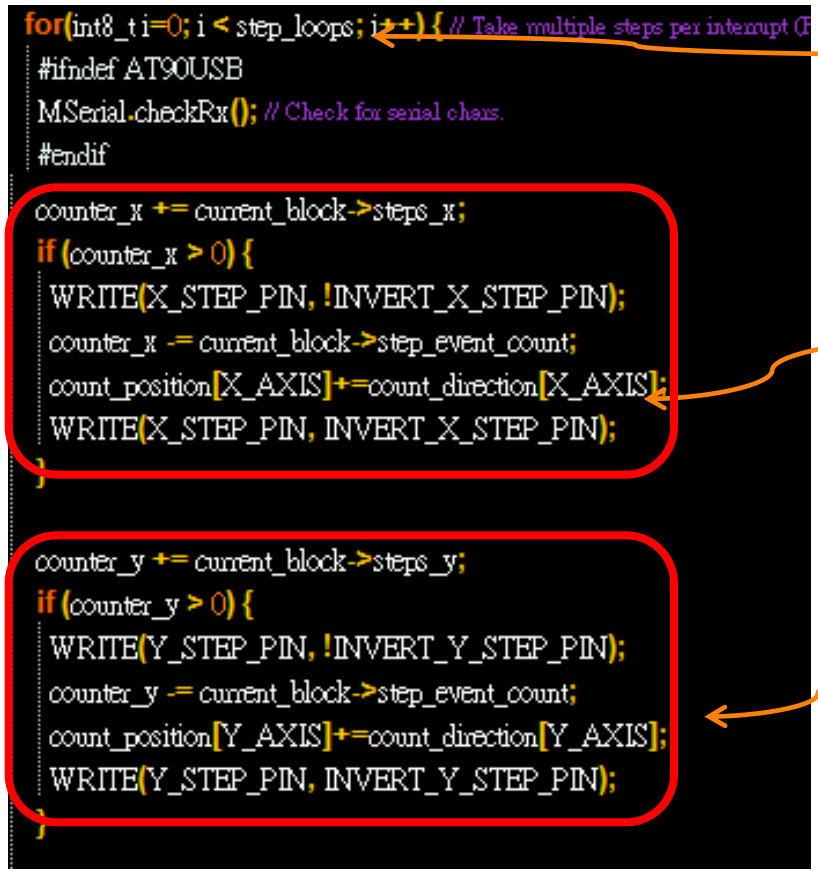
- Z_DUAL_STEPPER_DRIVERS 表示 Z 軸是用兩個 stepper driver

當 Z 軸 endstop 被觸動時

- 記錄當時 Z 軸的位置
- 設定 Z 軸 endstop 的狀態
- 設定該 block 已完成

Marlin step generate 的實作

- **ISR() 的講解**



step_loop 會依 calc_timer() 設定

X 軸生成 step

Y 軸生成 step

Marlin step generate 的實作

- **ISR() 的講解**

```
counter_z += current_block->steps_z;
if (counter_z > 0) {
  WRITE(Z_STEP_PIN, !INVERT_Z_STEP_PIN);

  #ifdef Z_DUAL_STEPPER_DRIVERS
    WRITE(Z2_STEP_PIN, !INVERT_Z_STEP_PIN);
  #endif

  counter_z -= current_block->step_event_count;
  count_position[Z_AXIS]+=count_direction[Z_AXIS];
  WRITE(Z_STEP_PIN, INVERT_Z_STEP_PIN);

  #ifdef Z_DUAL_STEPPER_DRIVERS
    WRITE(Z2_STEP_PIN, INVERT_Z_STEP_PIN);
  #endif
}
step_events_completed += 1;
if(step_events_completed >= current_block->step_event_count) break;
```

Z 軸生成 step

更新 block 的進度

Marlin step generate 的實作

- **ISR() 的講解**

```
// Calculate new timer value
unsigned short timer;
unsigned short step_rate;
if (step_events_completed <= (unsigned long int)current_block->accelerate_until) {

  MultiU24X24toH16(acc_step_rate, acceleration_time, current_block->acceleration_rate);
  acc_step_rate += current_block->initial_rate;

  // upper limit
  if(acc_step_rate > current_block->nominal_rate)
    acc_step_rate = current_block->nominal_rate;

  // step_rate to timer interval
  timer = calc_timer(acc_step_rate);
  OCR1A = timer;
  acceleration_time += timer;
```

梯形加速階段

val = long1 * long2 >> 24

- 用組語填 avr 的暫存器實作
- **acceleration_time 的單位 0.5 us**
- **acceleration_rate 的單位 1/8 (step / s^2)**
- **acc_step_rate 的單位 (step / s)**

更新目前的速度

更新下次中斷的時間

Marlin step generate 的實作

- **ISR() 的講解**

```
else if (step_events_completed > (unsigned long int)current_block->decelerate_after) {
  MultiU24X24toH16(step_rate, deceleration_time, current_block->acceleration_rate);

  if(step_rate > acc_step_rate) { // Check step_rate stays positive
    step_rate = current_block->final_rate;
  }
  else {
    step_rate = acc_step_rate - step_rate; // Decelerate from aceleration end point.
  }

  // lower limit
  if(step_rate < current_block->final_rate)
    step_rate = current_block->final_rate;

  // step_rate to timer interval
  timer = calc_timer(step_rate);
  OCR1A = timer;
  deceleration_time += timer;
```

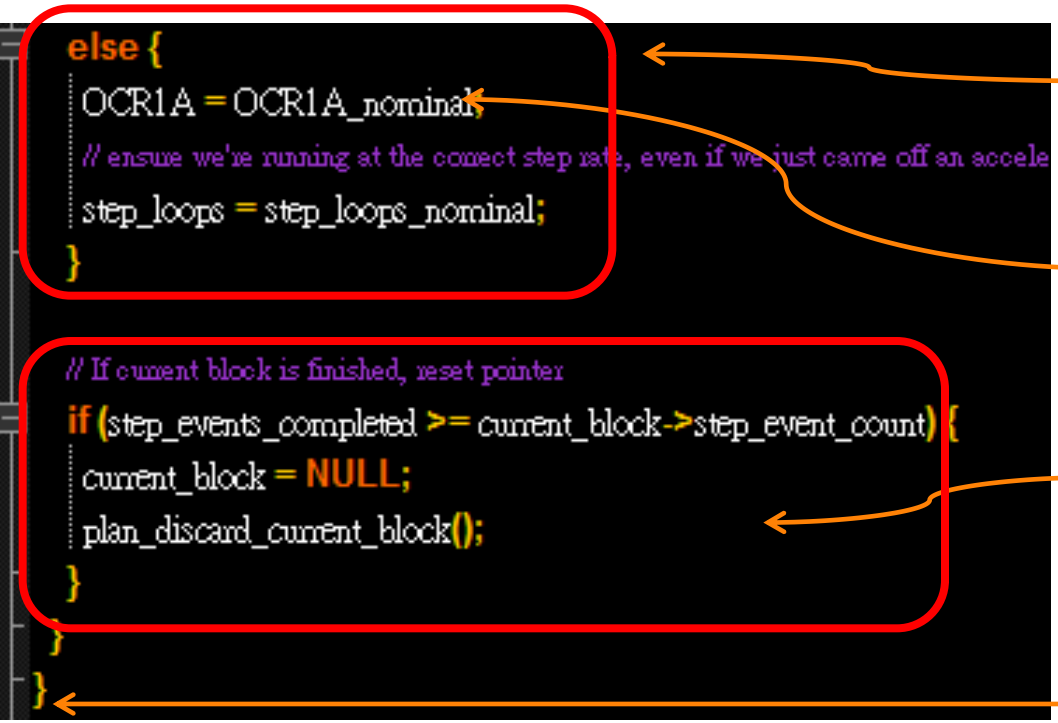
梯形減速階段

更新目前的速度

更新下次中斷的時間

Marlin step generate 的實作

- **ISR() 的講解**



梯形 **nominal** 階段

更新下次中斷的時間

該 **block** 被完成後清除

ISR() 結束

Marlin step generate 的實作

- 開始 **trapezoid_generator_reset() 的講解 (stepper.cpp)**

```
// Initializes the trapezoid generator from the current block. Called whenever a new
// block begins.
FORCE_INLINE void trapezoid_generator_reset() {
  deceleration_time = 0;
  // step_rate to timer interval
  OCR1A_nominal = calc_timer(current_block->nominal_rate);
  // make a note of the number of step loops required at nominal speed
  step_loops_nominal = step_loops;
```

Marlin step generate 的實作

- **trapezoid_generator_reset() 的講解**

```
// Initializes the trapezoid generator from the current block. Called whe
// block begins.
FORCE_INLINE void trapezoid_generator_reset() {
  deceleration_time = 0;
  // step_rate to timer interval
  OCR1A_nominal = calc_timer(current_block->nominal_rate);
  // make a note of the number of step loops required at nominal speed
  step_loops_nominal = step_loops;
  acc_step_rate = current_block->initial_rate;
  acceleration_time = calc_timer(acc_step_rate);
  OCR1A = acceleration_time;

}
```

設定減速段的時間

計算 **nominal** 段進中斷的時間

設定加速段的速度

計算加速段進中斷的時間

trapezoid_generator_reset() 結束

Marlin step generate 的實作

- 開始 calc_timer() 的講解 (stepper.cpp)

```
FORCE_INLINE unsigned short calc_timer(unsigned short step_rate) {
unsigned short timer;
if(step_rate > MAX_STEP_FREQUENCY) step_rate = MAX_STEP_FREQUENCY;

if(step_rate > 20000) { // If steprate > 20kHz >> step 4 times
 step_rate = (step_rate >> 2)&0x3fff;
 step_loops = 4;
}
else if(step_rate > 10000) { // If steprate > 10kHz >> step 2 times
 step_rate = (step_rate >> 1)&0x7fff;
 step_loops = 2;
}
else {
 step_loops = 1;
}
```

Marlin step generate 的實作

- **calc_timer() 的講解**

```
ORCE_INLINE unsigned short calc_timer(unsigned short step_rate) {
unsigned short timer;
if(step_rate > MAX_STEP_FREQUENCY) step_rate = MAX_STEP_FREQUENCY;

if(step_rate > 20000) { // If steprate > 20kHz >> step 4 times
 step_rate = (step_rate >> 2)&0x3fff;
 step_loops = 4;
}
else if(step_rate > 10000) { // If steprate > 10kHz >> step 2 times
 step_rate = (step_rate >> 1)&0x7fff;
 step_loops = 2;
}
else {
 step_loops = 1;
}
```

設定 **step_rate** 及 **step_loop**

- **stepper** 中斷一次約 **5KHz**
- 超過 **5KHz** 用 **for loop** 的方式代替
- 超過 **5KHz** 打出的 **pulse** 不會很均勻

Marlin step generate 的實作

- **calc_timer() 的講解**

```
if(step_rate < (F_CPU/500000)) step_rate = (F_CPU/500000);
step_rate -= (F_CPU/500000); // Correct for minimal speed
if(step_rate >= (8*256)){ // higher step rate
  unsigned short table_address = (unsigned short)&speed_lookuptable_fast[(unsigned char)(step_rate>>8)][0];
  unsigned char tmp_step_rate = (step_rate & 0x00ff);
  unsigned short gain = (unsigned short)pgm_read_word_near(table_address+2);
  MultiU16X8toH16(timer, tmp_step_rate, gain);
  timer = (unsigned short)pgm_read_word_near(table_address) - timer;
}
```

在高速的情形下

timer 的基數 F_CPU/500000

- **stepper 中斷最低速的限制**

利用查表的方式計算下次進中斷的時間

- **用 tmp_step_rate 及 gain 做內插估計 timer**
- **val = intval1 * intval2 >> 16**

Marlin step generate 的實作

- **calc_timer() 的講解**

```
else { // lower step rates
  unsigned short table_address = (unsigned short)&speed_lookuptable_slow[0][0];
  table_address += ((step_rate)>>1) & 0xfffc;
  timer = (unsigned short)pgm_read_word_near(table_address);
  timer -= (((unsigned short)pgm_read_word_near(table_address+2) * (unsigned char)(step_rate & 0x0007))>>3);
}
if(timer < 100) { timer = 100; MYSERIAL.print(MSG_STEPPER_TOO_HIGH); MYSERIAL.println(step_rate); }
return timer;
}
```

在低速的情形下

利用查表的方式計算下次進中斷的時間

- 做法類似上一頁

stepper 中斷最快限制約 20 KHz

calc_timer() 結束

Marlin step generate 的實作

- **speed_lookuptable.h 的列表**
  - **speed_lookuptable_fast**

```
#if F_CPU == 16000000

const uint16_t speed_lookuptable_fast[256][2] PROGMEM = {\
{62500, 55556}, {6944, 3268}, {3676, 1176}, {2500, 607}, {1893, 369}, {1524, 249}, {1275, 179}, {1096, 135},
{961, 105}, {856, 85}, {771, 69}, {702, 58}, {644, 49}, {595, 42}, {553, 37}, {516, 32},
{484, 28}, {456, 25}, {431, 23}, {408, 20}, {388, 19}, {369, 16}, {353, 16}, {337, 14},
{323, 13}, {310, 11}, {299, 11}, {288, 11}, {277, 9}, {268, 9}, {259, 8}, {251, 8},
{243, 8}, {235, 7}, {228, 6}, {222, 6}, {216, 6}, {210, 6}, {204, 5}, {199, 5},
{194, 5}, {189, 4}, {185, 4}, {181, 4}, {177, 4}, {173, 4}, {169, 4}, {165, 3},
{162, 3}, {159, 4}, {155, 3}, {152, 3}, {149, 2}, {147, 3}, {144, 3}, {141, 2},
{139, 3}, {136, 2}, {134, 2}, {132, 3}, {129, 2}, {127, 2}, {125, 2}, {123, 2},
{121, 2}, {119, 1}, {118, 2}, {116, 2}, {114, 1}, {113, 2}, {111, 2}, {109, 1},
{108, 2}, {106, 1}, {105, 2}, {103, 1}, {102, 1}, {101, 1}, {100, 2}, {98, 1},
{97, 1}, {96, 1}, {95, 2}, {93, 1}, {92, 1}, {91, 1}, {90, 1}, {89, 1},
{88, 1}, {87, 1}, {86, 1}, {85, 1}, {84, 1}, {83, 0}, {83, 1}, {82, 1},
{81, 1}, {80, 1}, {79, 1}, {78, 0}, {78, 1}, {77, 1}, {76, 1}, {75, 0},
{75, 1}, {74, 1}, {73, 1}, {72, 0}, {72, 1}, {71, 1}, {70, 0}, {70, 1},
{69, 0}, {69, 1}, {68, 1}, {67, 0}, {67, 1}, {66, 0}, {66, 1}, {65, 0},
{65, 1}, {64, 1}, {63, 0}, {63, 1}, {62, 0}, {62, 1}, {61, 0}, {61, 1},
{60, 0}, {60, 0}, {60, 1}, {59, 0}, {59, 1}, {58, 0}, {58, 1}, {57, 0},
```

Marlin step generate 的實作

- **speed_lookuptable.h 的列表**
  - **speed_lookuptable_slow**

```
const uint16_t speed_lookuptable_slow[256][2] PROGMEM = {\
{62500, 12500}, {50000, 8334}, {41666, 5952}, {35714, 4464}, {31250, 3473}, {27777, 2777}, {25000, 2273}, {22727, 1894},
{20833, 1603}, {19230, 1373}, {17857, 1191}, {16666, 1041}, {15625, 920}, {14705, 817}, {13888, 731}, {13157, 657},
{12500, 596}, {11904, 541}, {11363, 494}, {10869, 453}, {10416, 416}, {10000, 385}, {9615, 356}, {9259, 331},
{8928, 308}, {8620, 287}, {8333, 269}, {8064, 252}, {7812, 237}, {7575, 223}, {7352, 210}, {7142, 198},
{6944, 188}, {6756, 178}, {6578, 168}, {6410, 160}, {6250, 153}, {6097, 145}, {5952, 139}, {5813, 132},
{5681, 126}, {5555, 121}, {5434, 115}, {5319, 111}, {5208, 106}, {5102, 102}, {5000, 99}, {4901, 94},
{4807, 91}, {4716, 87}, {4629, 84}, {4545, 81}, {4464, 79}, {4385, 75}, {4310, 73}, {4237, 71},
{4166, 68}, {4098, 66}, {4032, 64}, {3968, 62}, {3906, 60}, {3846, 59}, {3787, 56}, {3731, 55},
{3676, 53}, {3623, 52}, {3571, 50}, {3521, 49}, {3472, 48}, {3424, 46}, {3378, 45}, {3333, 44},
{3289, 43}, {3246, 41}, {3205, 41}, {3164, 39}, {3125, 39}, {3086, 38}, {3048, 36}, {3012, 36},
{2976, 35}, {2941, 35}, {2906, 33}, {2873, 33}, {2840, 32}, {2808, 31}, {2777, 30}, {2747, 30},
{2717, 29}, {2688, 29}, {2659, 28}, {2631, 27}, {2604, 27}, {2577, 26}, {2551, 26}, {2525, 25},
{2500, 25}, {2475, 25}, {2450, 23}, {2427, 24}, {2403, 23}, {2380, 22}, {2358, 22}, {2336, 22},
{2314, 21}, {2293, 21}, {2272, 20}, {2252, 20}, {2232, 20}, {2212, 20}, {2192, 19}, {2173, 18},
{2155, 19}, {2136, 18}, {2118, 18}, {2100, 17}, {2083, 17}, {2066, 17}, {2049, 17}, {2032, 16},
{2016, 16}, {2000, 16}, {1984, 16}, {1968, 15}, {1953, 16}, {1937, 14}, {1923, 15}, {1908, 15},
{1893, 14}, {1879, 14}, {1865, 14}, {1851, 13}, {1838, 14}, {1824, 13}, {1811, 13}, {1798, 13},
{1785, 12}, {1773, 13}, {1760, 12}, {1748, 12}, {1736, 12}, {1724, 12}, {1712, 12}, {1700, 11},
{1689, 12}, {1677, 11}, {1666, 11}, {1655, 11}, {1644, 11}, {1633, 10}, {1623, 11}, {1612, 10},
```

Marlin step generate 的實作

- 利用 **create_speed_lookuptable.py** 建立 **speed_lookuptable.h** 列表
  - **cpu_freq** 為 **16 MHz**
  - **Timer_freq** 為 **2 MHz, 8** 為 **timer divider**

```
parser = argparse.ArgumentParser(description=__doc__)
parser.add_argument('-f', '--cpu-freq', type=int, default=16, help='CPU clockrate in MHz (default=16)')
parser.add_argument('-d', '--divider', type=int, default=8, help='Timer/counter pre-scale divider (default=8)')
args = parser.parse_args()

cpu_freq = args.cpu_freq * 1000000
timer_freq = cpu_freq / args.divider
```

Marlin step generate 的實作

- 利用 create_speed_lookuptable.py 建立 speed_lookuptable.h 列表
  - 計算 speed_lookuptable_fast 列表
  - a[i] 表示 timer counter
  - b[i] 表示 a[i] 與 a[i+1] 的差

```
print "const uint16_t speed_lookuptable_fast[256][2] PROGMEM = {"
a = [ timer_freq / ((i*256)+(args.cpu_freq*2)) for i in range(256) ]
b = [ a[i] - a[i+1] for i in range(255) ]
b.append(b[-1])
for i in range(32):
    print "  ",
    for j in range(8):
        print "{%d, %d}," % (a[8*i+j], b[8*i+j]),
    print
print "};"
print
```

Marlin step generate 的實作

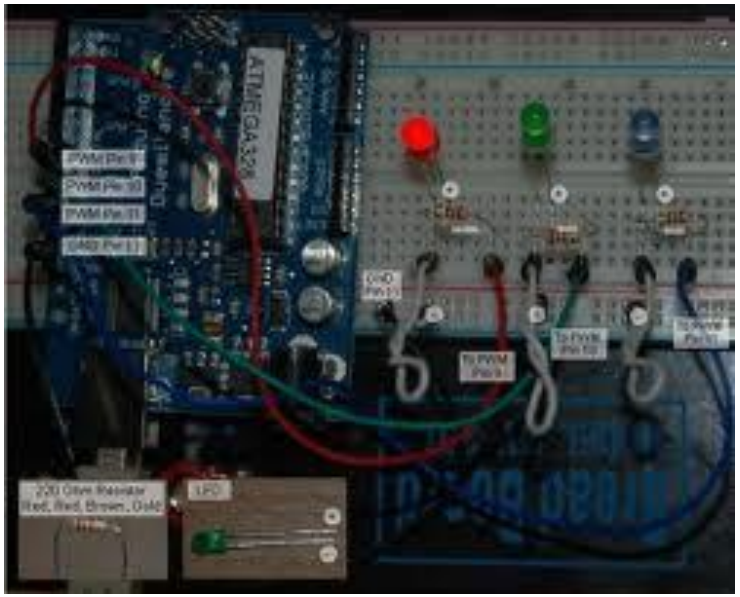
- 利用 **create_speed_lookuptable.py** 建立 **speed_lookuptable.h** 列表
  - 計算 **speed_lookuptable_slow** 列表
  - 與 **speed_lookuptable_fast** 做法相同

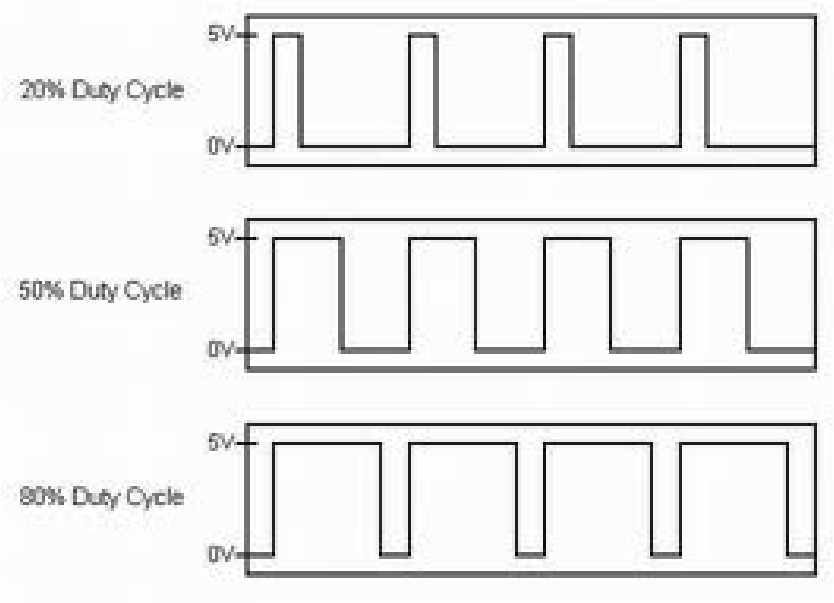
```
print "const uint16_t speed_lookuptable_slow[256][2] PROGMEM = {"
a = [ timer_freq / ((i*8)+(args.cpu_freq*2)) for i in range(256) ]
b = [ a[i] - a[i+1] for i in range(255) ]
b.append(b[-1])
for i in range(32):
    print "  ",
    for j in range(8):
        print "{%d, %d}," % (a[8*i+j], b[8*i+j]),
    print
print "};"
print
```

Marlin 溫度控制的實作

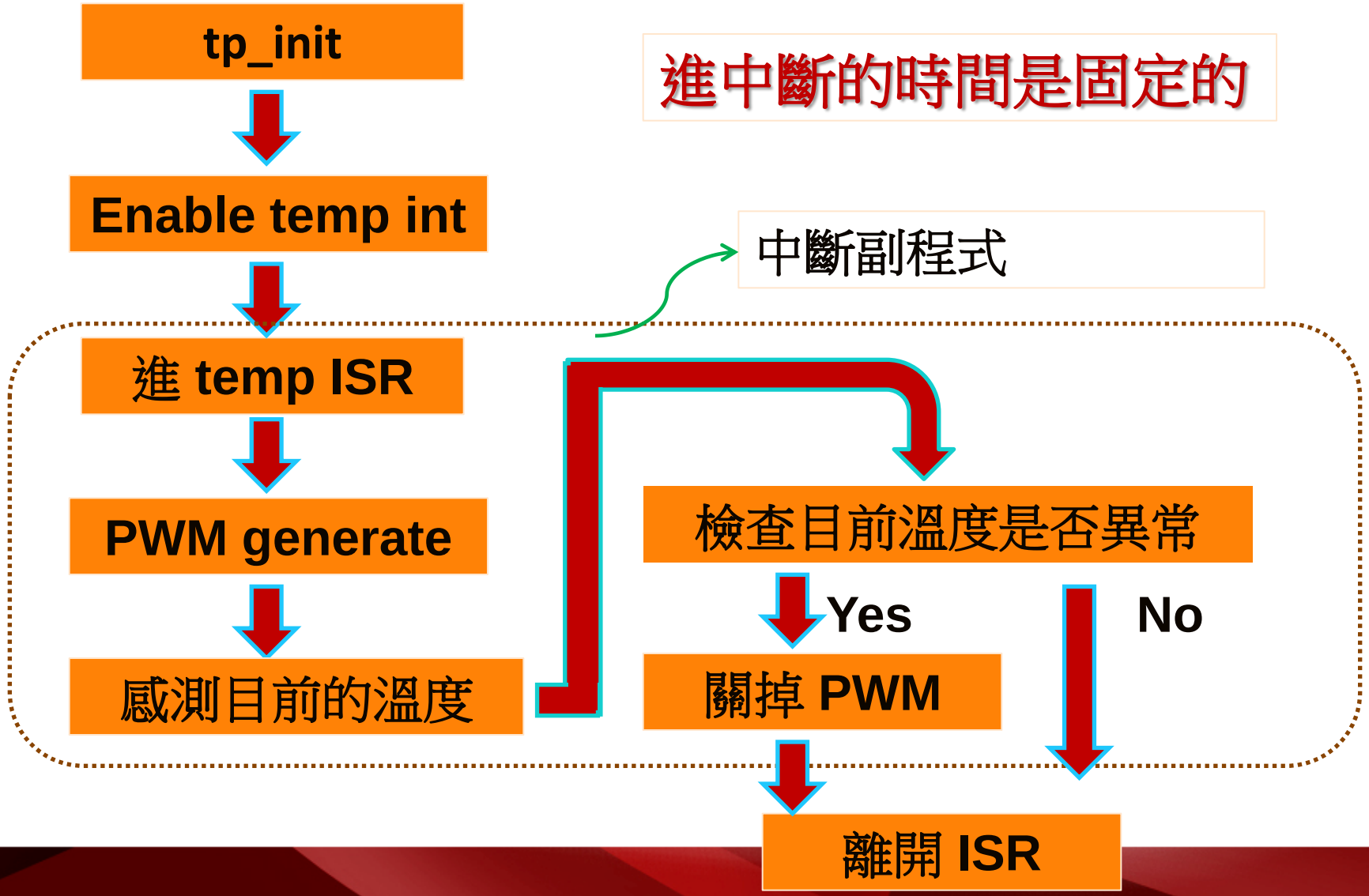
Marlin 溫度控制的實作

- **Marlin 是利用中斷的方式來實作**
  - 使用 **Arduino (Atmega)** 的 **8-bit timer0** 中斷
  - 在 **ISR (Interrupt Service Routine)** 內用 **PWM** 控制溫度





Marlin 溫度控制的流程



Marlin 溫度控制的實作

- 開始 ISR() 的講解 (temperature.cpp)

```
// Timer 0 is shared with millies
ISR(TIMER0_COMPB_vect)
{
  //these variables are only accesible from the ISR, but static, so they don't loose their value
  static unsigned char temp_count = 0;
  static unsigned long raw_temp_0_value = 0;
  static unsigned long raw_temp_1_value = 0;
  static unsigned long raw_temp_2_value = 0;
  static unsigned long raw_temp_bed_value = 0;
  static unsigned char temp_state = 0;
  static unsigned char pwm_count = 1;
  static unsigned char soft_pwm_0;
  #if EXTRUDERS > 1
  static unsigned char soft_pwm_1;
  #endif
```

Marlin 溫度控制的實作

- **ISR() 的講解**

```
//these variables are only accesible from the ISR, but stat
static unsigned char temp_count = 0;
static unsigned long raw_temp_0_value = 0;
static unsigned long raw_temp_1_value = 0;
static unsigned long raw_temp_2_value = 0;
static unsigned long raw_temp_bed_value = 0;
static unsigned char temp_state = 0;
static unsigned char pwm_count = 1;
static unsigned char soft_pwm_0;
#if EXTRUDERS > 1
static unsigned char soft_pwm_1;
#endif
#if EXTRUDERS > 2
static unsigned char soft_pwm_2;
#endif
#if HEATER_BED_PIN > -1
static unsigned char soft_pwm_b;
#endif
```

記錄擠出頭及加熱板 ADC 的數值

PWM 週期的計數器

依擠出頭個數決定

Marlin 溫度控制的實作

- **ISR() 的講解**

```
if(pwm_count == 0){
  soft_pwm_0 = soft_pwm[0];
  if(soft_pwm_0 > 0) WRITE(HEATER_0_PIN,1);
  #if EXTRUDERS > 1
  soft_pwm_1 = soft_pwm[1];
  if(soft_pwm_1 > 0) WRITE(HEATER_1_PIN,1);
  #endif
  #if EXTRUDERS > 2
  soft_pwm_2 = soft_pwm[2];
  if(soft_pwm_2 > 0) WRITE(HEATER_2_PIN,1);
  #endif
  #if defined(HEATER_BED_PIN) && HEATER_BED_PIN > -1
  soft_pwm_b = soft_pwm_bed;
  if(soft_pwm_b > 0) WRITE(HEATER_BED_PIN,1);
  #endif
  #ifdef FAN_SOFT_PWM
  soft_pwm_fan =(unsigned char) fanSpeed;
  if(soft_pwm_fan > 0) WRITE(FAN_PIN,1);
  #endif
}
```

前一個 **PWM** 週期結束

讀取各個擠出頭的 **PWM duty** 及做加熱的動作

讀取加熱板的 **PWM duty** 及做加熱的動作

讀取風扇的 **PWM duty** 及做風扇的轉動

Marlin 溫度控制的實作

- **ISR() 的講解**

```
}
if(soft_pwm_0 <= pwm_count) WRITE(HEATER_0_PIN,0);
#if EXTRUDERS > 1
if(soft_pwm_1 <= pwm_count) WRITE(HEATER_1_PIN,0);
#endif
#if EXTRUDERS > 2
if(soft_pwm_2 <= pwm_count) WRITE(HEATER_2_PIN,0);
#endif
#if defined(HEATER_BED_PIN) && HEATER_BED_PIN > -1
if(soft_pwm_b <= pwm_count) WRITE(HEATER_BED_PIN,0);
#endif
#ifdef FAN_SOFT_PWM
if(soft_pwm_fan <= pwm_count) WRITE(FAN_PIN,0);
#endif

pwm_count++;
pwm_count &= 0x7f;
```

檢查各個擠出頭、加熱板及風扇的 **duty**，及做關掉的動作

計數 **PWM** 的週期

- **PWM 的週期為 128 ms**

Marlin 溫度控制的實作

- **ISR() 的講解**

```
switch(temp_state) {
  case 0: // Prepare TEMP_0
    #if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
      #if TEMP_0_PIN > 7
        ADCSRB = 1<<MUX5;
      #else
        ADCSRB = 0;
      #endif
      ADMUX = ((1 << REFS0) | (TEMP_0_PIN & 0x07));
      ADCSRA |= 1<<ADSC; // Start conversion
    #endif
    lcd_buttons_update();
    temp_state = 1;
    break;
  case 1: // Measure TEMP_0
    #if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
      raw_temp_0_value += ADC;
    #endif
    #ifdef HEATER_0_USES_MAX6675 // TODO remove the blocking
      raw_temp_0_value = read_max6675();
    #endif
    temp_state = 2;
    break;
```

設定 **TEMP_0_PIN** 的 **analog** 暫存器

讀取 **TEMP_0_PIN** 的 **ADC** 數值

- 以讀取 **16** 次 **ADC** 的平均值做為參考溫度
- 可以用 **max6675** 感測熱電偶

Marlin 溫度控制的實作

- **ISR() 的講解**

```
case 2: // Prepare TEMP_BED
 #if defined(TEMP_BED_PIN) && (TEMP_BED_PIN > -1)
  #if TEMP_BED_PIN > 7
   ADCSRB = 1<<MUX5;
  #else
   ADCSRB = 0;
  #endif
  ADMUX = ((1 << REFS0) | (TEMP_BED_PIN & 0x07));
  ADCSRA |= 1<<ADSC; // Start conversion
 #endif
 lcd_buttons_update();
 temp_state = 3;
 break;
case 3: // Measure TEMP_BED
 #if defined(TEMP_BED_PIN) && (TEMP_BED_PIN > -1)
  raw_temp_bed_value += ADC;
 #endif
 temp_state = 4;
 break;
```

設定 **TEMP_BED_PIN** 的 **analog** 暫存器

讀取 **TEMP_BED_PIN** 的 **ADC** 數值

- 以讀取 **16** 次 **ADC** 的平均值做為參考溫度

Marlin 溫度控制的實作

- **ISR() 的講解**

```
case 4: // Prepare TEMP_1
 #if defined(TEMP_1_PIN) && (TEMP_1_PIN > -1)
  #if TEMP_1_PIN > 7
   ADCSRB = 1<<MUX5;
  #else
   ADCSRB = 0;
  #endif
  ADMUX = ((1 << REFS0) | (TEMP_1_PIN & 0x07));
  ADCSRA |= 1<<ADSC; // Start conversion
 #endif
 lcd_buttons_update();
 temp_state = 5;
 break;
case 5: // Measure TEMP_1
 #if defined(TEMP_1_PIN) && (TEMP_1_PIN > -1)
  raw_temp_1_value += ADC;
 #endif
 temp_state = 6;
 break;
```

設定 **TEMP_1_PIN** 的 **analog** 暫存器

讀取 **TEMP_1_PIN** 的 **ADC** 數值

- 以讀取 **16** 次 **ADC** 的平均值做為參考溫度

Marlin 溫度控制的實作

- **ISR() 的講解**

```
case 6: // Prepare TEMP_2
#if defined(TEMP_2_PIN) && (TEMP_2_PIN > -1)
  #if TEMP_2_PIN > 7
    ADCSRB = 1<<MUX5;
  #else
    ADCSRB = 0;
  #endif
  ADMUX = ((1 << REFS0) | (TEMP_2_PIN & 0x07));
  ADCSRA |= 1<<ADSC; // Start conversion
#endif
lcd_buttons_update();
temp_state = 7;
break;
case 7: // Measure TEMP_2
#if defined(TEMP_2_PIN) && (TEMP_2_PIN > -1)
  raw_temp_2_value += ADC;
#endif
temp_state = 0;
temp_count++;
break;
```

設定 **TEMP_1_PIN** 的 **analog** 暫存器

讀取 **TEMP_1_PIN** 的 **ADC** 數值

- 以讀取 **16** 次 **ADC** 的平均值做為參考溫度

計數感溫的週期

Marlin 溫度控制的實作

- **ISR() 的講解**

```
if(temp_count >= 16) // 8 ms * 16 = 128ms.
{
  if (!temp_meas_ready) //Only update the raw values if they have b
  {
    current_temperature_raw[0] = raw_temp_0_value;
#if EXTRUDERS > 1
    current_temperature_raw[1] = raw_temp_1_value;
#endif
#if EXTRUDERS > 2
    current_temperature_raw[2] = raw_temp_2_value;
#endif
    current_temperature_bed_raw = raw_temp_bed_value;
  }

  temp_meas_ready = true;
  temp_count = 0;
  raw_temp_0_value = 0;
  raw_temp_1_value = 0;
  raw_temp_2_value = 0;
  raw_temp_bed_value = 0;
```

感溫週期結束，更新目前溫度

將 **ADC** 值轉換為溫度的 **flag**

更新各個擠出頭及加熱板的 **ADC** 值

Marlin 溫度控制的實作

- **ISR() 的講解**

```
#if HEATER_0_RAW_LO_TEMP > HEATER_0_RAW_HI_TEMP
  if(current_temperature_raw[0] <= maxttemp_raw[0]) {
#else
  if(current_temperature_raw[0] >= maxttemp_raw[0]) {
#endif
    max_temp_error(0);
  }
#if HEATER_0_RAW_LO_TEMP > HEATER_0_RAW_HI_TEMP
  if(current_temperature_raw[0] >= minttemp_raw[0]) {
#else
  if(current_temperature_raw[0] <= minttemp_raw[0]) {
#endif
    min_temp_error(0);
  }
#if EXTRUDERS > 1
#if HEATER_1_RAW_LO_TEMP > HEATER_1_RAW_HI_TEMP
  if(current_temperature_raw[1] <= maxttemp_raw[1]) {
#else
  if(current_temperature_raw[1] >= maxttemp_raw[1]) {
#endif
```

檢查擠出頭 0 的溫度是否異常

- 如果有，則關掉 PWM 並發出錯誤訊息

Marlin 溫度控制的實作

- **ISR() 的講解**

```
#if EXTRUDERS > 1
#if HEATER_1_RAW_LO_TEMP > HEATER_1_RAW_HI_TEMP
  if(current_temperature_raw[1] <= maxttemp_raw[1]) {
#else
  if(current_temperature_raw[1] >= maxttemp_raw[1]) {
#endif
    max_temp_error(1);
  }
#if HEATER_1_RAW_LO_TEMP > HEATER_1_RAW_HI_TEMP
  if(current_temperature_raw[1] >= minttemp_raw[1]) {
#else
  if(current_temperature_raw[1] <= minttemp_raw[1]) {
#endif
    min_temp_error(1);
  }
#endif
#if EXTRUDERS > 2
#if HEATER_2_RAW_LO_TEMP > HEATER_2_RAW_HI_TEMP
  if(current_temperature_raw[2] <= maxttemp_raw[2]) {
#else
  if(current_temperature_raw[2] >= maxttemp_raw[2]) {
```

檢查擠出頭 1 的溫度是否異常

- 如果有，則關掉 PWM 並發出錯誤訊息

Marlin 溫度控制的實作

- **ISR() 的講解**

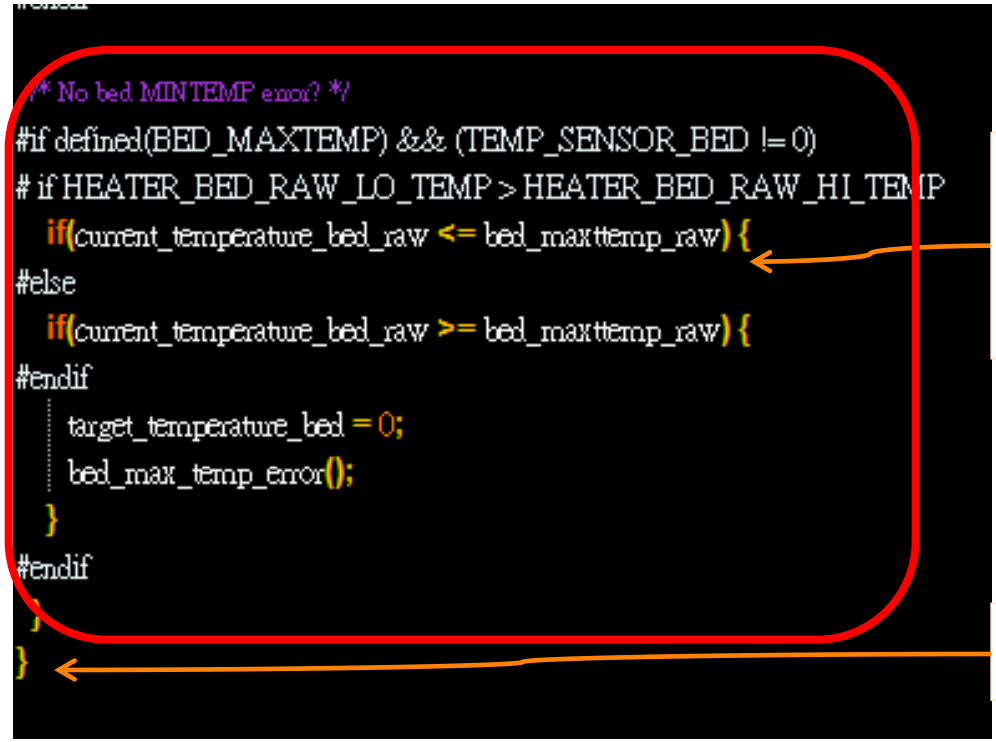
```
  }
#endif
#if EXTRUDERS > 2
#if HEATER_2_RAW_LO_TEMP > HEATER_2_RAW_HI_TEMP
  if(current_temperature_raw[2] <= maxttemp_raw[2]) {
#else
  if(current_temperature_raw[2] >= maxttemp_raw[2]) {
#endif
    max_temp_error(2);
  }
#if HEATER_2_RAW_LO_TEMP > HEATER_2_RAW_HI_TEMP
  if(current_temperature_raw[2] >= minttemp_raw[2]) {
#else
  if(current_temperature_raw[2] <= minttemp_raw[2]) {
#endif
    min_temp_error(2);
  }
#endif

 /* No bed MINTEMP error? */
```

檢查擠出頭 2 的溫度是否異常

- 如果有，則關掉 PWM 並發出錯誤訊息

Marlin 溫度控制的實作

- **ISR() 的講解**



檢查加熱板的溫度是否異常

- 如果有，則關掉 **PWM** 並發出錯誤訊息

ISR 結束

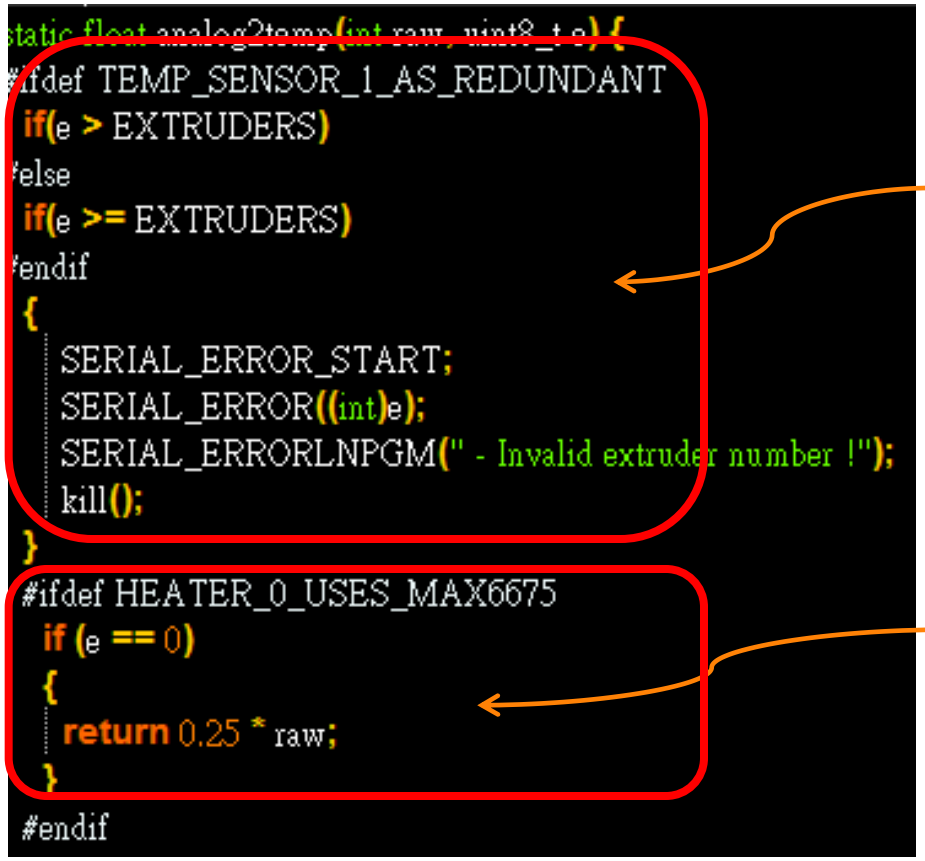
Marlin 溫度感測

Marlin 溫度感測

- 溫度感測區分為熱敏電阻與熱電偶
  - Marlin 依熱敏電阻型號，建立相對應的溫度查表
  - Marlin 支援的熱電偶的轉接板有 max6675 及 AD595
  - 可參考 Marlin overview 的投影片

Marlin 溫度感測

- **analog2temp() 的講解 (temperature.cpp)**



檢查擠出頭的編號

使用 **max6675** 感測溫度

Marlin 溫度感測

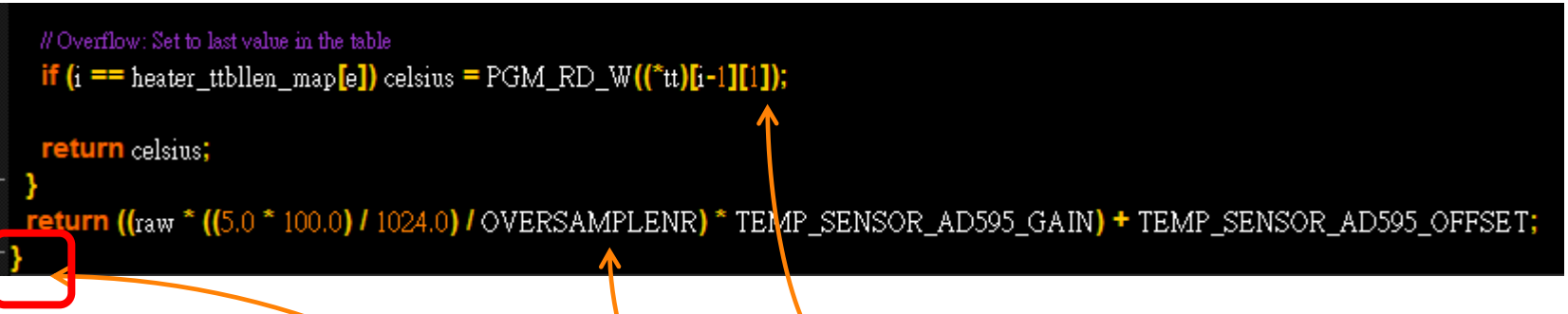
- **analog2temp() 的講解**

```
if(heater_ttbl_map[e] != NULL)
{
  float celsius = 0;
  uint8_t i;
  short (*tt)[][2] = (short (*)[][2])(heater_ttbl_map[e]);

  for (i=1; i<heater_ttbllen_map[e]; i++)
  {
    if (PGM_RD_W((*tt)[i][0]) > raw)
    {
      celsius = PGM_RD_W((*tt)[i-1][1]) +
        (raw - PGM_RD_W((*tt)[i-1][0])) *
        (float)(PGM_RD_W((*tt)[i][1]) - PGM_RD_W((*tt)[i-1][1])) /
        (float)(PGM_RD_W((*tt)[i][0]) - PGM_RD_W((*tt)[i-1][0]));
      break;
    }
  }
```

透過查表，利用內差的方式估計擠出頭目前的溫度

Marlin 溫度感測

- **analog2temp() 的講解**



當擠出頭的溫度超出範圍的估算

使用 **AD595** 感測溫度

analog2temp funtion 結束

Marlin 溫度感測

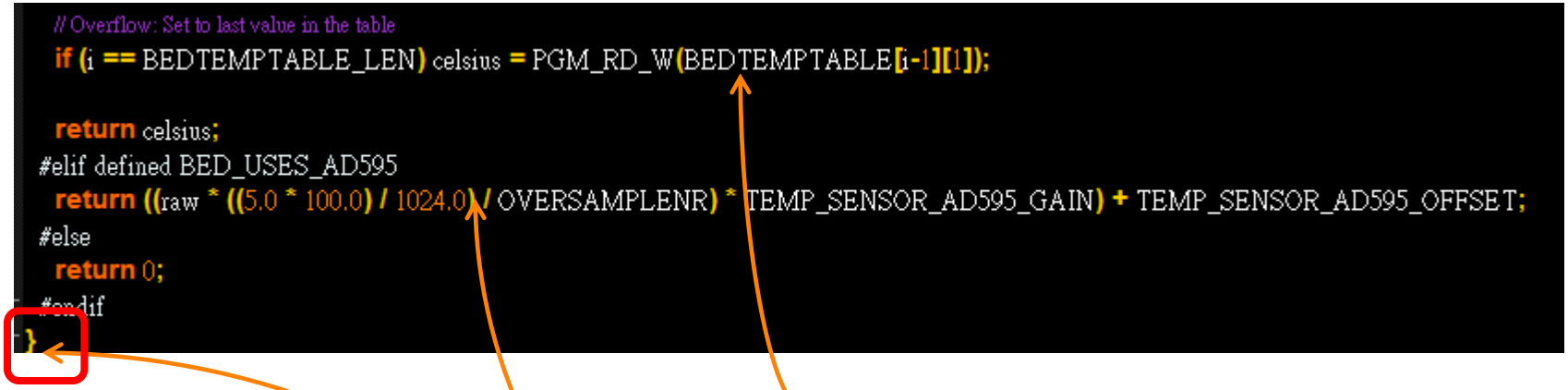
- **analog2tempBed() 的講解 (temperature.cpp)**

```
static float analog2tempBed(int raw) {
  #ifdef BED_USES_THERMISTOR
    float celsius = 0;
    byte i;

    for (i=1; i<BEDTEMPTABLE_LEN; i++)
    {
      if (PGM_RD_W(BEDTEMPTABLE[i][0]) > raw)
      {
        celsius  = PGM_RD_W(BEDTEMPTABLE[i-1][1]) +
          (raw - PGM_RD_W(BEDTEMPTABLE[i-1][0])) *
          (float)(PGM_RD_W(BEDTEMPTABLE[i][1]) - PGM_RD_W(BEDTEMPTABLE[i-1][1])) /
          (float)(PGM_RD_W(BEDTEMPTABLE[i][0]) - PGM_RD_W(BEDTEMPTABLE[i-1][0]));
        break;
      }
    }
```

透過查表，利用內差的方式估計加熱板目前的溫度

Marlin 溫度感測

- **analog2tempBed() 的講解 (temperature.cpp)**



```
  // Overflow: Set to last value in the table
  if (i == BEDTEMPTABLE_LEN) celsius = PGM_RD_W(BEDTEMPTABLE[i-1][1]);

  return celsius;
#elif defined BED_USES_AD595
  return ((raw * ((5.0 * 100.0) / 1024.0) / OVERSAMPLENR) * TEMP_SENSOR_AD595_GAIN) + TEMP_SENSOR_AD595_OFFSET;
#else
  return 0;
#endif
}
```

當加熱板的溫度超出範圍的估算

使用 **AD595** 感測溫度

analog2tempBed funtion 結束

Marlin 溫度感測

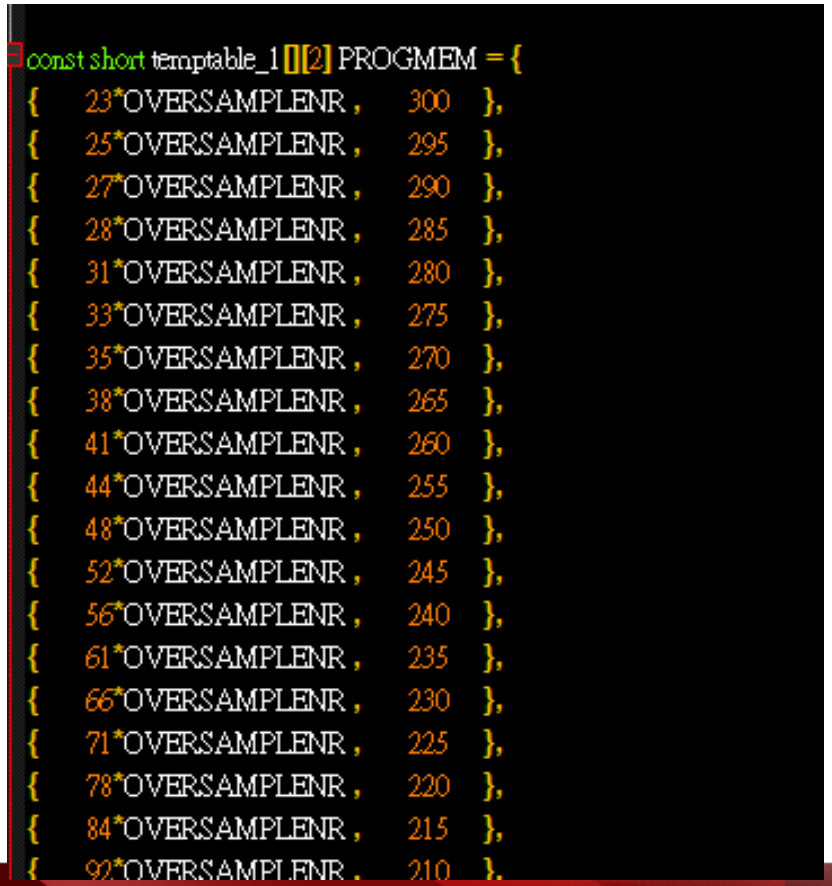
- **updateTemperaturesFromRawValues() 的講解 (temperature.cpp)**

```
static void updateTemperaturesFromRawValues()
{
  for(uint8_t e=0;e<EXTRUDERS;e++)
  {
    current_temperature[e] = analog2temp(current_temperature_raw[e], e);
  }
  current_temperature_bed = analog2tempBed(current_temperature_bed_raw);
  #ifdef TEMP_SENSOR_1_AS_REDUNDANT
    redundant_temperature = analog2temp(redundant_temperature_raw, 1);
  #endif
  //Reset the watchdog after we know we have a temperature measurement.
  watchdog_reset();

  CRITICAL_SECTION_START;
  temp_meas_ready = false;
  CRITICAL_SECTION_END;
}
```

讀取擠出頭及加熱板目前的溫度

Marlin 溫度感測

- **thermistortables.h 的列表**
  - 可參考 **Marlin overview** 的投影片



```
const short temptable_1[][2] PROGMEM = {
{   23*OVERSAMPLENR,   300  },
{   25*OVERSAMPLENR,   295  },
{   27*OVERSAMPLENR,   290  },
{   28*OVERSAMPLENR,   285  },
{   31*OVERSAMPLENR,   280  },
{   33*OVERSAMPLENR,   275  },
{   35*OVERSAMPLENR,   270  },
{   38*OVERSAMPLENR,   265  },
{   41*OVERSAMPLENR,   260  },
{   44*OVERSAMPLENR,   255  },
{   48*OVERSAMPLENR,   250  },
{   52*OVERSAMPLENR,   245  },
{   56*OVERSAMPLENR,   240  },
{   61*OVERSAMPLENR,   235  },
{   66*OVERSAMPLENR,   230  },
{   71*OVERSAMPLENR,   225  },
{   78*OVERSAMPLENR,   220  },
{   84*OVERSAMPLENR,   215  },
{   92*OVERSAMPLENR,   210  },
```

Marlin Porting Notes

3D Printer 韌體原始碼閱讀心得 (V)

Outline

- 移植 Marlin 需注意的事項
  - 目標的硬體平台規格是否符合需求
  - 需改寫原來 Marlin 有關硬體部分的程式
- Marlin 在 86duino 上的移植
  - GPIO 的移植
  - ADC, SPI, serial port 的移植
  - Marlin timer 中斷的移植

移植 Malrin 需注意的事項

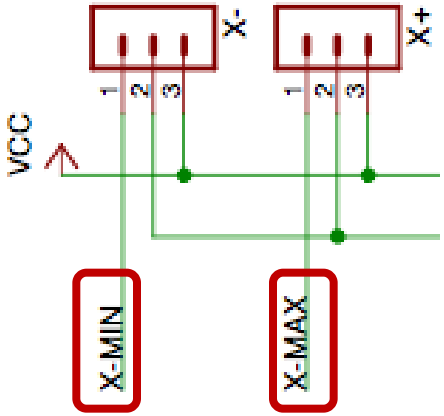
移植 Marlin 需注意的事項

- 將移植 **Marlin** 到目標的硬體平台要考慮的問題
  - 目標的硬體平台 **CPU** 的性能是否符合 **Marlin**的需求
  - 目標的硬體平台是否能滿足 **Marlin** 對硬體的 **I/O** 裝置
  - 需要改寫 **Marlin low-level hardware access** 部分的程式

移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

1. **Marlin 對 GPIO 腳位的需求**
   - 控制 **XYZ** 三軸步進馬達的腳位
     - 一軸至少要 **3** 個腳位**(step, dir, enable)**
   - 至少能控制一個 **E** 軸步進馬達的腳位
   - **XYZ** 三軸近接開關的腳位
     - 一軸最多 **2** 個，最少 **0** 個



移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

1. Marlin 對 GPIO 腳位的需求
   - 最少需要 12 個腳位
     - 僅控制 XYZE 四軸的步進馬達，沒有近接開關
   - 全功能需要 24 個腳位
     - 控制 XYZ 三軸的步進馬達, 3 個擠出頭的步進馬達, 6 個近接開關

移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

2. Marlin 對 GPIO 規格的要求
   - I/O 輸出與輸入的電壓是否需要做轉換
     - ATMega 的 I/O 輸出與輸入的電壓範圍是 0V ～ 5V
     - 以 A4988 為例，訊號高電位的電壓要在 3 V ～ 5.5 V 之間
   - Marlin 對 I/O 存取時間有限制
     - 原始的 Marlin 存取 I/O 的時間為 3us 左右
     - Marlin 中斷處理的必須小於 50us，所以 I/O 存取時間不能太長
     - Intel Galileo 可能不太合適

Q: What is the maximum rate at which GPIO output pins can be updated?

The GPIO output pins on Intel® Galileo are provided by an I2C Port Expander that is running at standard mode (100 kHz). Each I2C request to update a GPIO requires approximately 2ms. In addition to software overhead, this restricts the frequency achievable on the GPIO outputs to approximately 230 Hz.

移植 Marlin 需注意的事項

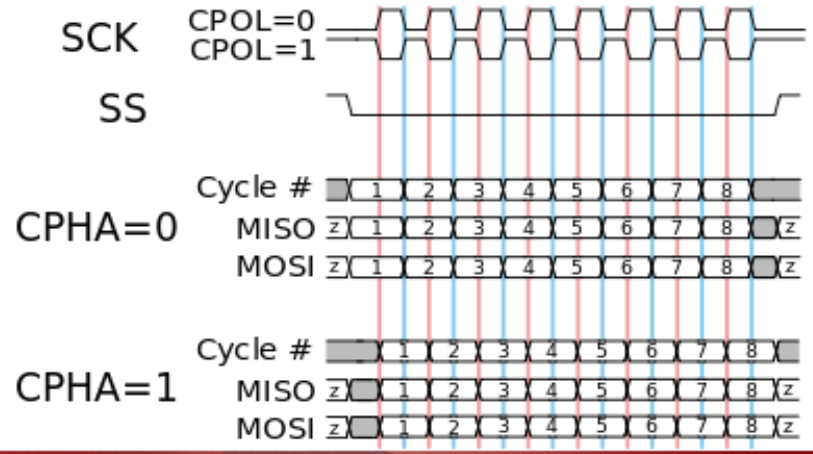
Marlin 的移植需考慮硬體平台是否有符合需求

3. Marlin 對硬體週邊裝置的需求
   - 支援 ADC 讀取熱敏電阻感測溫度
     - 僅感測 1 個擠出頭的溫度，最少需要 1 組 ADC
     - 一般是 2 組 ADC，可以感測 1 個擠出頭及露熱板的溫度
     - ADC 存取的時間不能太長 ，與存取 I/O 的原因相同
     - 如果移植的硬體平台 ADC 的解析度太低，或是偵測電壓的範圍太小，都會造成溫度的誤差變大
       - 原始 Marlin 使用 的 ADC 解析度為 10 bit , 偵測電壓的範圍是 0 V ～ 5V

移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

3. Marlin 對硬體週邊裝置的需求
   - 提供 SPI 讀取熱電偶，感測溫度(非必須)
     - 原始的 Marlin 最多提供一組 SPI
     - 原始Marlin 的 SPI 傳輸速度是 1 MHz
     - SPI 支援的 mode 愈多愈好
       - Max6675 的 mode 是 CPOL = 0, CPHP = 0



移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

3. 是否支援 Marlin 對硬體週邊裝置的需求
   - 與 PC 做通訊的機制
     - Marlin 是透過 USB serial 的介面與 PC 做通訊
   - 如果要以網路的方式與 PC 做通訊
     - 移植的硬體平台本身要有網路通訊的裝置，如 ethernet, ethercat
     - 加入網路通訊相關的程式

移植 Marlin 需注意的事項

Marlin 的移植需考慮硬體平台是否有符合需求

4. Marlin 對中斷的規格需求
   - 需要 2 組可發中斷的 timer
     - 一組是 stepper 的 timer, 一組是控制加熱的 timer
   - 中斷的 latency 不能太長
     - 中斷 latency 加上中斷副程式的時間必須小於 50 us
   - timer 解析度的需求
     - 原始 Marlin 的 stepper timer 解析度為 0.5 us
     - 原始 Marlin 的控制加熱timer 解析度為 8 us (1000/128 us)

移植 Marlin 需注意的事項

移植Marlin 到非 ATMega 平台，需改寫如下部分的程式

- **Marlin 對 GPIO 的讀寫**
- **Marlin 對 ADC 的存取**
- **Marlin 中斷的處理**
- **Marlin 對 serial port 的存取**
- **Marlin 對 SPI 的存取 (非必須)**
- **Marlin 對 LCD 顯示模組的存取 (非必須)**
- **Marlin 對 SD card 的存取 (非必須)**

移植 Marlin 需注意的事項

- **Marlin 對 GPIO 讀寫的地方**
  - 在 fastio.h 設定 ATMega 暫存器對 I/O 做存取

```
/// toggle a pin
#define _TOGGLE(IO)  do {DIO ## IO ## _RPORT = MASK(DIO ## IO ## _PIN); } while (0)

/// set pin as input
#define  _SET_INPUT(IO) do {DIO ## IO ## _DDR &= ~MASK(DIO ## IO ## _PIN); } while (0)
/// set pin as output
#define  _SET_OUTPUT(IO) do {DIO ## IO ## _DDR |= MASK(DIO ## IO ## _PIN); } while (0)

/// check if pin is an input
#define  _GET_INPUT(IO)  ((DIO ## IO ## _DDR & MASK(DIO ## IO ## _PIN)) == 0)
/// check if pin is an output
#define  _GET_OUTPUT(IO)  ((DIO ## IO ## _DDR & MASK(DIO ## IO ## _PIN)) != 0)

/// check if pin is an timer
#define  _GET_TIMER(IO)  (DIO ## IO ## _PWM)
```

移植 Marlin 需注意的事項

- **Marlin 對 ADC 存取的地方 (temperature.cpp)**
  - 在 **tp_init()** 設定 **ADC** 的暫存器
  - 在 **isr()** 讀取 **ADC** 的數值

```
ADCSRA = 1<<ADEN | 1<<ADSC | 1<<ADIF | 0x07;
DIDR0 = 0;
#ifdef DIDR2
 DIDR2 = 0;
#endif
#if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
 #if TEMP_0_PIN < 8
  DIDR0 |= 1 << TEMP_0_PIN;
 #else
  DIDR2 |= 1<<(TEMP_0_PIN - 8);
 #endif
#endif
#if defined(TEMP_1_PIN) && (TEMP_1_PIN > -1)
 #if TEMP_1_PIN < 8
  DIDR0 |= 1<<TEMP_1_PIN;
 #else
  DIDR2 |= 1<<(TEMP_1_PIN - 8);
 #endif
#endif
```

```
switch(temp_state) {
 case 0: // Prepare TEMP_0
 #if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  #if TEMP_0_PIN > 7
   ADCSRB = 1<<MUX5;
  #else
   ADCSRB = 0;
  #endif
  ADMUX = ((1 << REFS0) | (TEMP_0_PIN & 0x07));
  ADCSRA |= 1<<ADSC; // Start conversion
 #endif
 lcd_buttons_update();
 temp_state = 1;
 break;
 case 1: // Measure TEMP_0
 #if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  raw_temp_0_value += ADC;
 #endif
 #ifdef HEATER_0_USES_MAX6675 // TODO remove the blocking
  raw_temp_0_value = read_max6675();
 #endif
 temp_state = 2;
 break;
```

移植 Marlin 需注意的事項

- **Marlin 中斷處理的地方**
  - 在 **stepper.cpp** 處理 **stepper** 的中斷
  - 在 **temperature.cpp** 處理 **PWM** 的中斷

```
ISR(TIMER1_COMPA_vect)
{
  // If there is no current block, attempt to pop one from the buffer
  if (current_block == NULL) {
    // Anything in the buffer?
    current_block = plan_get_current_block();
    if (current_block != NULL) {
      current_block->busy = true;
      trapezoid_generator_reset();
      counter_x = -(current_block->step_event_count >> 1);
      counter_y = counter_x;
      counter_z = counter_x;
      counter_e = counter_x;
      step_events_completed = 0;

      #ifdef Z_LATE_ENABLE
        if(current_block->steps_z > 0) {
          enable_z();
          OCR1A = 2000; //1ms wait
          return;
        }
      #endif
```

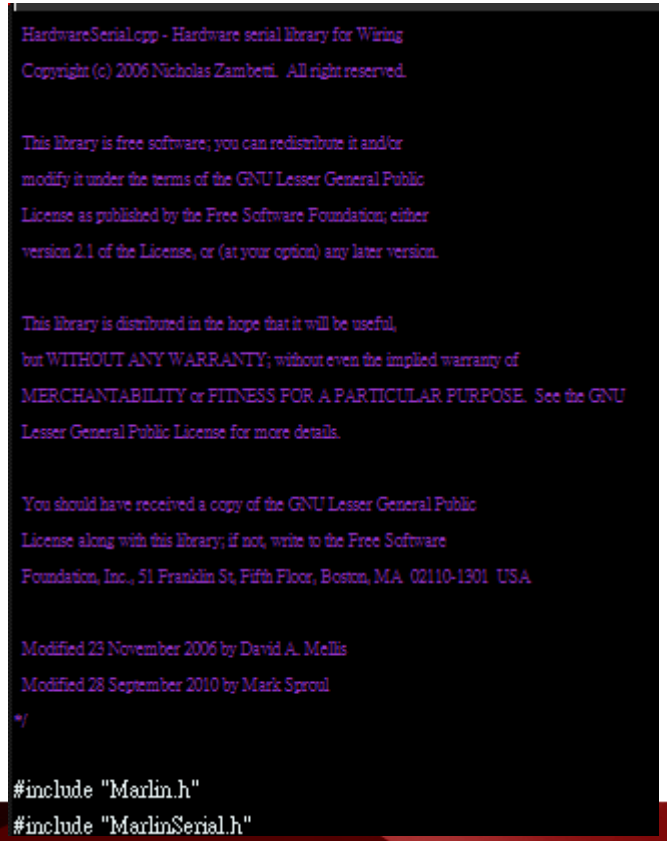
```
// Timer 0 is shared with millies
ISR(TIMER0_COMPB_vect)
{
  //these variables are only accesible from the ISR, but static, so they don't lose their value
  static unsigned char temp_count = 0;
  static unsigned long raw_temp_0_value = 0;
  static unsigned long raw_temp_1_value = 0;
  static unsigned long raw_temp_2_value = 0;
  static unsigned long raw_temp_bed_value = 0;
  static unsigned char temp_state = 0;
  static unsigned char pwm_count = (1 << SOFT_PWM_SCALE);
  static unsigned char soft_pwm_0;
  #if EXTRUDERS > 1
  static unsigned char soft_pwm_1;
  #endif
  #if EXTRUDERS > 2
  static unsigned char soft_pwm_2;
```

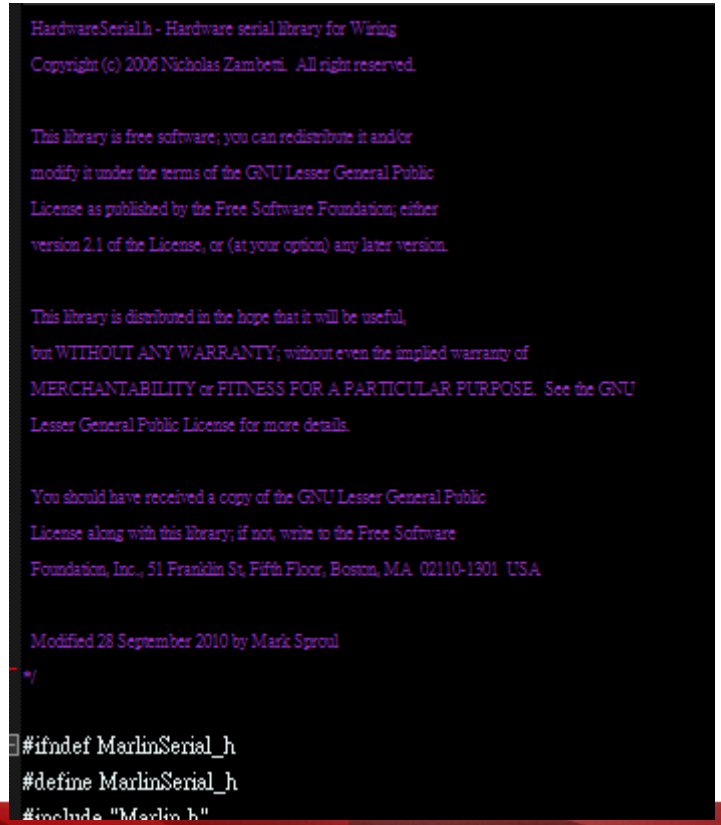
移植 Marlin 需注意的事項

- **Marlin serial lib 的做法**
  - **Marlin 為了兼容 Atmel AT90USB 型號的 MCU ，沒有直接 call Arduino 的 HardwareSerial lib**
  - **Marlin 用 MarlinSerial lib 取代 HardwareSerial lib**
    - **MarlinSerial lib 的底層的 fuction 與 HardwareSerial lib 相同**
    - **MarlinSerial lib 多了判斷 AT90USB 型號的機制**
    - **MarlinSerial lib 增加一些 讀寫 的 marco 方便使用**
    - **在 call MarlinSerial lib 時，會 disable HardwareSerial lib 的功能，所以用 Arduino IIED 編譯時不會發生衝突**

移植 Marlin 需注意的事項

- **Marlin 對 serial port 存取的地方**
  - **Marlin 透過 MarlinSerial.h 及 MarlinSerial.cpp 對 serial port 存取**





移植 Marlin 需注意的事項

- **Marlin 對 SPI 存取的地方**
  - 在 **tp_init()** 設定 **SPI**
  - 在 **readMax6675()** 讀取 **SPI** 的數值

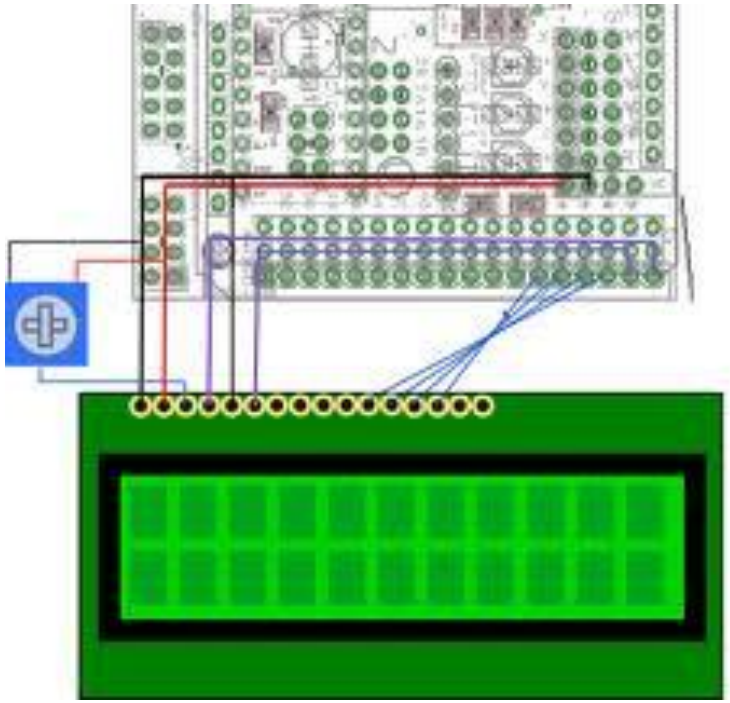
```
#ifdef HEATER_0_USES_MAX6675
  #ifndef SDSUPPORT
    SET_OUTPUT(MAX_SCK_PIN);
    WRITE(MAX_SCK_PIN,0);

    SET_OUTPUT(MAX_MOSI_PIN);
    WRITE(MAX_MOSI_PIN,1);

    SET_INPUT(MAX_MISO_PIN);
    WRITE(MAX_MISO_PIN,1);
  #endif

  SET_OUTPUT(MAX6675_SS);
  WRITE(MAX6675_SS,1);
#endif
```

```
#ifdef  PRR
  PRR &= ~(1<<PRSPI);
#elif defined PRR0
  PRR0 &= ~(1<<PRSPI);
#endif

SPCR = (1<<MSTR) | (1<<SPE) | (1<<SPR0);

// enable TT_MAX6675
WRITE(MAX6675_SS, 0);

// ensure 100ns delay - a bit extra is fine
asm("nop");//50ns on 20Mhz, 62.5ns on 16Mhz
asm("nop");//50ns on 20Mhz, 62.5ns on 16Mhz

// read MSB
SPDR = 0;
for (;(SPSR & (1<<SPIF)) == 0;);
max6675_temp = SPDR;
max6675_temp <<= 8;

// read LSB
SPDR = 0;
```

```
for (;(SPSR & (1<<SPIF)) == 0;);
max6675_temp |= SPDR;

// disable TT_MAX6675
WRITE(MAX6675_SS, 1);

if (max6675_temp & 4)
{
  // thermocouple open
  max6675_temp = 2000;
}
else
{
  max6675_temp = max6675_temp >> 3;
}

return max6675_temp;
```

移植 Marlin 需注意的事項

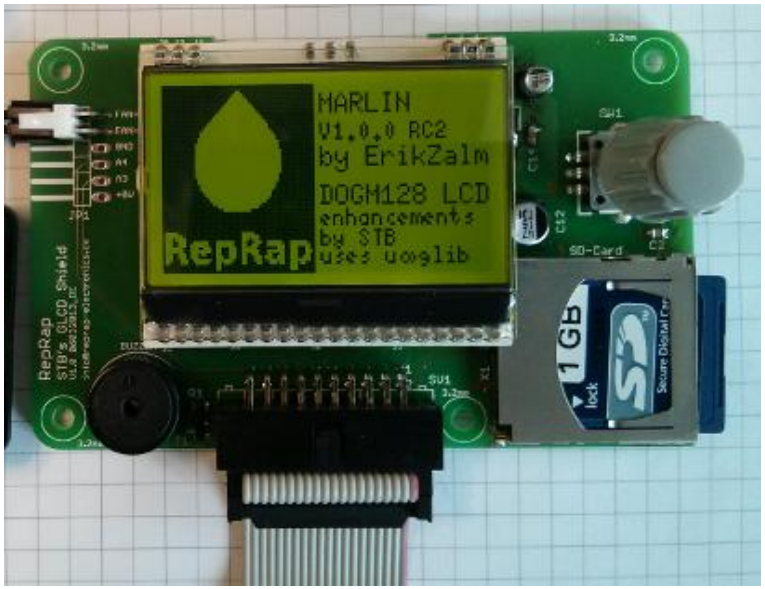
- **Marlin 對 LCD 顯示模組的存取**
  - **Marlin 是利用 GPIO 對 LCD 顯示模組作設定，所以只需改寫 GPIO 即可**



RAMPS pin	LCD PIN	Purpose
GND	1	
5V	2	
	3 (10K trimpot)	Contrast
D16	4	RS
GND	5	R/W
D17	6	Enable
	7	Data 1
	8	Data 2
	9	Data 3
	10	Data 4
23	11	Data 5
25	12	Data 6
27	13	Data 7
29	14	Data 8
	15	Backlight +
	16	Backlight -

移植 Marlin 需注意的事項

- **Marlin 對 SD card 的存取**
  - **Marlin 是利用 SPI 對 SD card 做讀寫，所以只需改寫 SPI 即可**
  - **熱電偶與 SD card 用不同的 SS (CS) 腳位，共用一組 SPI**



Marlin 的移植以 86Duino 為例

Marlin 的移植以 86Duino 為例

- 針對 86Duino 移植 Marlin 的想法
  - 只改寫 low-level 硬體部分的程式，與硬體無關的程式都沿用，盡量以最少的方式做改寫
  - 由於 86Duino IDE 與 Arduino IDE 相容，盡量使用 Ardiuno 原有的 function 做改寫
  - 原始 Marlin 中斷的機制與 86Duino 的中斷機制不同，這部分會改寫，中斷副程式的內容則保留

Marlin 的移植以 86duino 為例

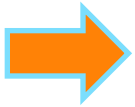
- 針對 86Duino 移植 Marlin 的做法
  - 對 86Duino GPIO 的移植
  - 對 86Duino 硬體裝置的移植
  - 86Duino 對 timer 中斷的移植

86Duino GPIO 的移植

86Duino GPIO 的移植

1. 刪除 pins.h 原有控制板型號的定義

```
#ifndef PINS_H
#define PINS_H

#define X_MS1_PIN -1
#define X_MS2_PIN -1
#define Y_MS1_PIN -1
#define Y_MS2_PIN -1
#define Z_MS1_PIN -1
#define Z_MS2_PIN -1
#define E0_MS1_PIN -1
#define E0_MS2_PIN -1
#define E1_MS1_PIN -1
#define E1_MS2_PIN -1
#define DIGIPOTSS_PIN -1

#if MOTHERBOARD == 99
#define KNOWN_BOARD 1

#define X_STEP_PIN       2
#define X_DIR_PIN        3
#define X_ENABLE_PIN     -1
#define X_STOP_PIN       16
```



```
#ifndef PINS_H
#define PINS_H

#define X_MS1_PIN -1
#define X_MS2_PIN -1
#define Y_MS1_PIN -1
#define Y_MS2_PIN -1
#define Z_MS1_PIN -1
#define Z_MS2_PIN -1
#define E0_MS1_PIN -1
#define E0_MS2_PIN -1
#define E1_MS1_PIN -1
#define E1_MS2_PIN -1
#define DIGIPOTSS_PIN -1

#define SENSITIVE_PINS {0, 1, X_STEP_PIN, X_DIR_PIN, X_
                        HEATER_BED_PIN, FAN_PIN,          \
                        _E0_PINS _E1_PINS _E2_PINS        \
                        /*analogInputToDigitalPin(TEMP_0_PIN), analogInputToDigitalPin(
#endif
```

86Duino GPIO 的移植

2. 新增 86Duino 控制板型號

– 在 pins.h 新增 86Duino Zero 及 86Duino One 的 型號

```
#define E0_MS1_PIN -1
#define E0_MS2_PIN -1
#define E1_MS1_PIN -1
#define E1_MS2_PIN -1
#define DIGIPOTSS_PIN -1

#if MOTHERBOARD == 860
#define KNOWN_BOARD 1

#define X_STEP_PIN
#define X_DIR_PIN       2
#define X_ENABLE_PIN    3
#define X_MIN_PIN       4
#define X_MAX_PIN       -1   //2 //Max endstops defau
```

```
#if MOTHERBOARD == 861
#define KNOWN_BOARD 1

#define X_STEP_PIN
#define X_DIR_PIN       2
#define X_ENABLE_PIN    3
#define X_MIN_PIN       4
#define X_MAX_PIN       -1   //2 //Max endstops default

#define Y_STEP_PIN      5
#define Y_DIR_PIN       6
#define Y_ENABLE_PIN    7
#define Y_MIN_PIN       21
#define Y_MAX_PIN       -1   //15
```

86Duino GPIO 的移植

3. 在 Configuration.h 設定對應的 86Duino 參數
   - 下圖以 86Duino One 為例

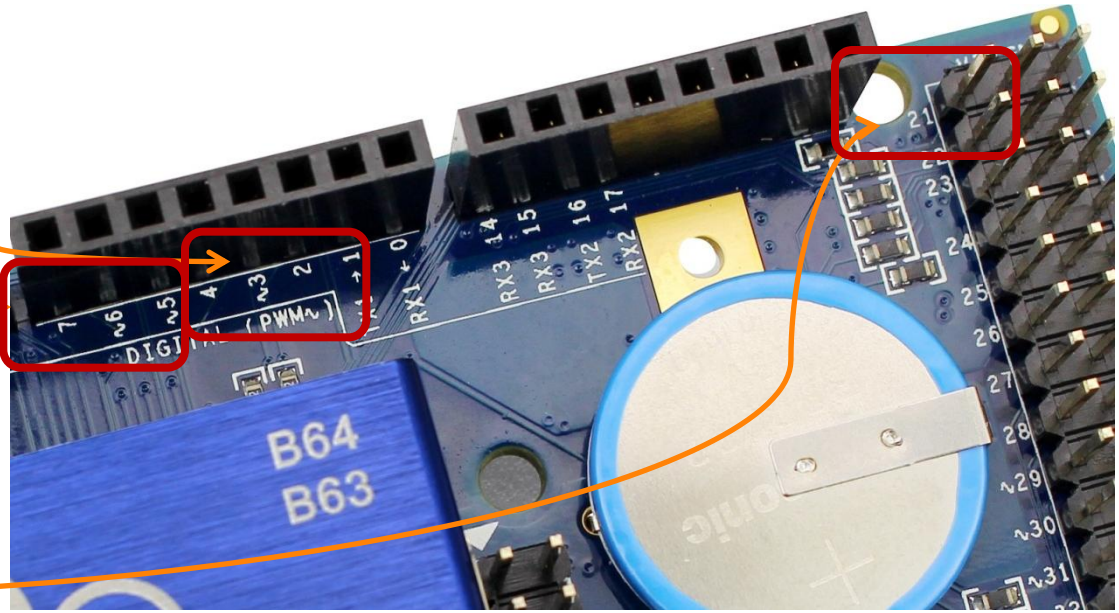
```
// 860 = 86Duino Zero
// 861 = 86Duino One

#ifndef MOTHERBOARD
#define MOTHERBOARD 861
#endif
```

86duino GPIO 的移植

4. 86Duino I/O 腳位的對應

– 下圖以 **86Duino One** 為例

```
#if MOTHERBOARD == 861
#define KNOWN_BOARD 1

#define X_STEP_PIN
#define X_DIR_PIN       2
#define X_ENABLE_PIN    3
#define X_MIN_PIN       4
#define X_MAX_PIN       -1   //2 //Max endstop default

#define Y_STEP_PIN      5
#define Y_DIR_PIN       6
#define Y_ENABLE_PIN    7
#define Y_MIN_PIN       21
#define Y_MAX_PIN       -1   //15
```

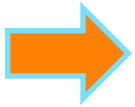


86duino GPIO 的移植

5. 在 fastio.h 改寫對 I/O 存取的 function

– 採用與 Arduino 相容的 function 對 86Duino 的 GPIO 做存取

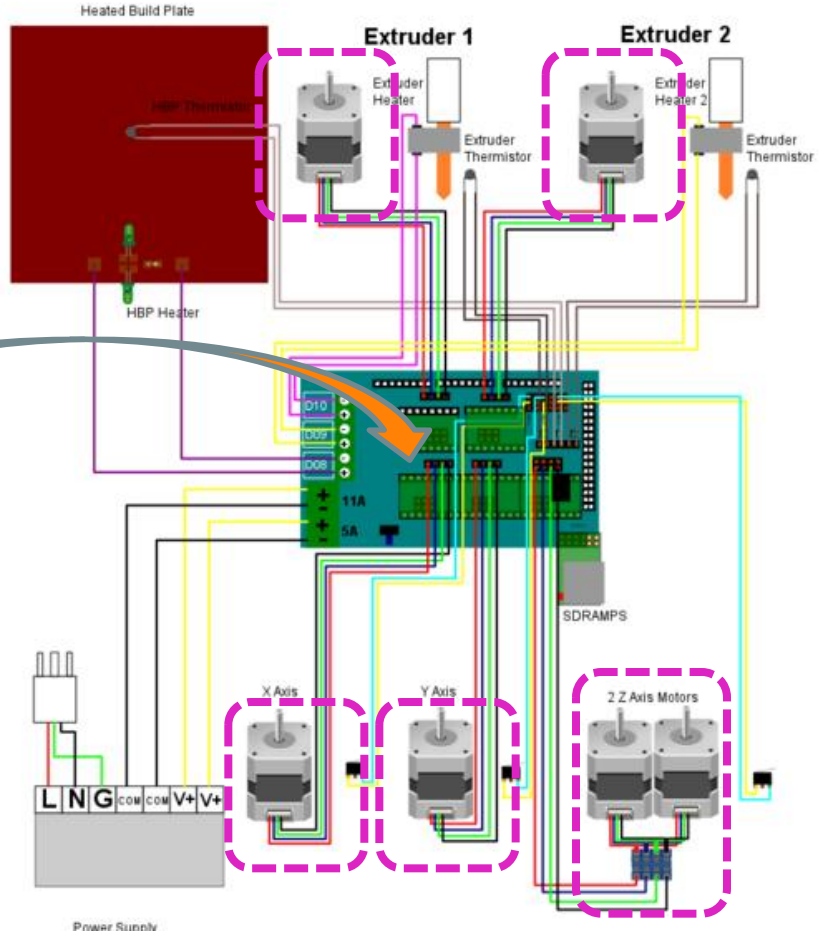
```
/// Read a pin wrapper
#define READ(IO)  _READ(IO)
/// Write to a pin wrapper
#define WRITE(IO, v)  _WRITE(IO, v)
/// set pin as input wrapper
#define SET_INPUT(IO)  _SET_INPUT(IO)
/// set pin as output wrapper
#define SET_OUTPUT(IO)  _SET_OUTPUT(IO)
/// check if pin is an input wrapper
#define GET_INPUT(IO)  _GET_INPUT(IO)
/// check if pin is an output wrapper
#define GET_OUTPUT(IO)  _GET_OUTPUT(IO)
/// check if pin is an timer wrapper
#define GET_TIMER(IO)  _GET_TIMER(IO)
```



```
/// Read a pin wrapper
#define READ(IO)  digitalRead(IO) //_READ(IO)
/// Write to a pin wrapper
#define WRITE(IO, v)  digitalWrite(IO,v) //_WRITE(IO, v)

/// toggle a pin wrapper
//#define TOGGLE(IO)  _TOGGLE(IO)

/// set pin as input wrapper
#define SET_INPUT(IO)  pinMode(IO,INPUT) //_SET_INPUT
/// set pin as output wrapper
#define SET_OUTPUT(IO)  pinMode(IO, OUTPUT) //_SET_
```

86duino GPIO 的移植

- 按照上敘步驟 的做法，便完成 **Marlin 對 86Duino GPIO 的移植**



86Duino 硬體裝置的移植

86duino 硬體裝置的移植

- 針對 86Duino 與 Marlin 相關硬體裝置的移植
  - 用來讀取熱敏電阻，感測溫度的 ADC
  - 用來讀取熱熱電偶，感測溫度的 SPI
  - 與 PC 做傳輸的serial port

86Duino ADC 的移植

86Duino ADC 的移植

1. 86Duino 的 ADC 與 Arduino 的 ADC 差異

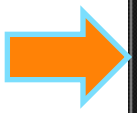
- 86Duino ADC 的偵測電壓為 0 ~ 3.3 V，解析度是 11 bit
- ATMega ADC 的偵測電壓為 0 ~ 5 V，解析度是 10 bit
- 由於兩者偵測電壓的範圍不同，所以 thermistortables.h 的查表要重新產生

```
#ifndef THERMISTORTABLES_H_
#define THERMISTORTABLES_H_

#include "Marlin.h"

#define OVERSAMPLENR 16

#if (THERMISTORHEATER_0 == 1) || (THERMISTORHEATER_1 == 1) || (THER

const short temptable_1[][2] PROGMEM = {
{   23*OVERSAMPLENR,    300   },
{   25*OVERSAMPLENR,    295   },
{   27*OVERSAMPLENR,    290   },
{   28*OVERSAMPLENR,    285   },
{   31*OVERSAMPLENR,    280   },
{   33*OVERSAMPLENR,    275   },
{   35*OVERSAMPLENR,    270   },
{   38*OVERSAMPLENR,    265   },
{   41*OVERSAMPLENR,    260   },
{   44*OVERSAMPLENR,    255   },
{   48*OVERSAMPLENR,    250   },
```

86Duino ADC 的移植

2. thermistortables.h 原有的查表可以刪除

```
#ifndef THERMISTORTABLES_H_
#define THERMISTORTABLES_H_

#include "Marlin.h"

#define OVERSAMPLENR 16

#if (THERMISTORHEATER_0 == 1) || (THERMISTORHEATER_1 == 1) || (THERI

const short temptable_1[][2] PROGMEM = {
{   23*OVERSAMPLENR,   300   },
{   25*OVERSAMPLENR,   295   },
{   27*OVERSAMPLENR,   290   },
{   28*OVERSAMPLENR,   285   },
{   31*OVERSAMPLENR,   280   },
{   33*OVERSAMPLENR,   275   },
{   35*OVERSAMPLENR,   270   },
{   38*OVERSAMPLENR,   265   },
{   41*OVERSAMPLENR,   260   },
{   44*OVERSAMPLENR,   255   },
{   48*OVERSAMPLENR,   250   },
{   52*OVERSAMPLENR,   245   },
{   56*OVERSAMPLENR,   240   },
{   61*OVERSAMPLENR,   235   },
{   66*OVERSAMPLENR,   230   },
{   71*OVERSAMPLENR,   225   },
{   78*OVERSAMPLENR,   220   },
```



```
#ifndef THERMISTORTABLES_H_
#define THERMISTORTABLES_H_

#include "Marlin.h"

#define OVERSAMPLENR 16

#define _TT_NAME(_N) temptable_ ## _N
#define TT_NAME(_N) _TT_NAME(_N)

#ifdef THERMISTORHEATER_0
# define HEATER_0_TEMPTABLE TT_NAME(THERMISTORHEATER_0)
# define HEATER_0_TEMPTABLE_LEN (sizeof(HEATER_0_TEMPTABLE)/sizeof(*HE
#else
# ifdef HEATER_0_USES_THERMISTOR
#  error No heater 0 thermistor table specified
# else //HEATER_0_USES_THERMISTOR
#  define HEATER_0_TEMPTABLE NULL
#  define HEATER_0_TEMPTABLE_LEN 0
# endif //HEATER_0_USES_THERMISTOR
#endif
```

86Duino ADC 的移植

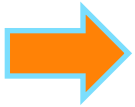
3. **Marlin 是利用 createTemperatureLookupMarlin .py 產生 thermistortables**
   - **createTemperatureLookupMarlin .py 預設 ADC 的偵測電壓為 0 ~ 5 V，所以要先修改為 3.3 V，再產生 thermistortables**

```
r2 = r2                 # resistance at middle temperature
t3 = t3 + 273.15        # high temperature (250C)
r3 = r3                 # resistance at high temperature
self.rp = rp            # pull-up resistance
self.vadc = 3.3         # ADC reference
self.vcc = 3.3          # supply voltage to potential divider
a1 = log(r1)
a2 = log(r2)
a3 = log(r3)
```

86Duino ADC 的移植

4. 由 createTemperatureLookupMarlin .py 產生的查表複製到 thermistortables.h

```
C:\DJGPP\EX_marlin>createTemperatureLoo
-10
// Thermistor lookup table for Marlin
// ./createTemperatureLookup.py --rp=47
.6 --num-temps=36
#define NUMTEMPS 36
short temptable[NUMTEMPS][2] = {
   {145, 350},
   {163, 340},
   {185, 330},
   {211, 320},
   {240, 310},
   {276, 300},
   {318, 290},
   {367, 280},
   {427, 270},
   {498, 260},
   {585, 250},
   {690, 240},
   {817, 230},
   {973, 220},
   {1165, 210},
```

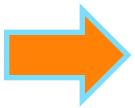


```
#if (THERMISTORHEATER_0 == 1) || (THERMISTORHEATER_1 == 1)
const short temptable_1[][2] PROGMEM = {
/*
  {92, 350},
  {104, 340},
  {118, 330},
  {133, 320},
  {152, 310},
  {173, 300},
  {199, 290},
  {229, 280},
  {266, 270},
  {309, 260},
  {362, 250},
  {425, 240},
  {503, 230},
  {598, 220},
  {714, 210},
  {858, 200},
  {1036, 190},
  {1258, 180},
  {1535, 170},
  {1880, 160},
```

86Duino ADC 的移植

5. 刪除 tp_init() 初始化 ATMega ADC 暫存器的設定 (temperature.cpp)

```
ADCSRA = 1<<ADEN | 1<<ADSC | 1<<ADIF | 0x07;
DIDR0 = 0;
#ifdef DIDR2
  DIDR2 = 0;
#endif
#if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  #if TEMP_0_PIN < 8
    DIDR0 |= 1 << TEMP_0_PIN;
  #else
    DIDR2 |= 1<<(TEMP_0_PIN - 8);
  #endif
#endif
#if defined(TEMP_1_PIN) && (TEMP_1_PIN > -1)
  #if TEMP_1_PIN < 8
    DIDR0 |= 1<<TEMP_1_PIN;
  #else
    DIDR2 |= 1<<(TEMP_1_PIN - 8);
  #endif
#endif
```



```
    SET_OUTPUT(MAX_MOSI_PIN);
    WRITE(MAX_MOSI_PIN,1);

    SET_INPUT(MAX_MISO_PIN);
    WRITE(MAX_MISO_PIN,1);
  #endif

  SET_OUTPUT(MAX6675_SS);
  WRITE(MAX6675_SS,1);
#endif

// Use timer0 for temperature measurement
// Interleave temperature interrupt with millies interrupt
OCR0B = 128;
TIMSK0 |= (1<<OCIE0B);

// Wait for temperature measurement to settle
delay(250);
```

86Duino ADC 的移植

6. 改寫 ISR() 對 ADC 讀取(temperature.cpp)

原來對 **ATMega** 暫存器作操作

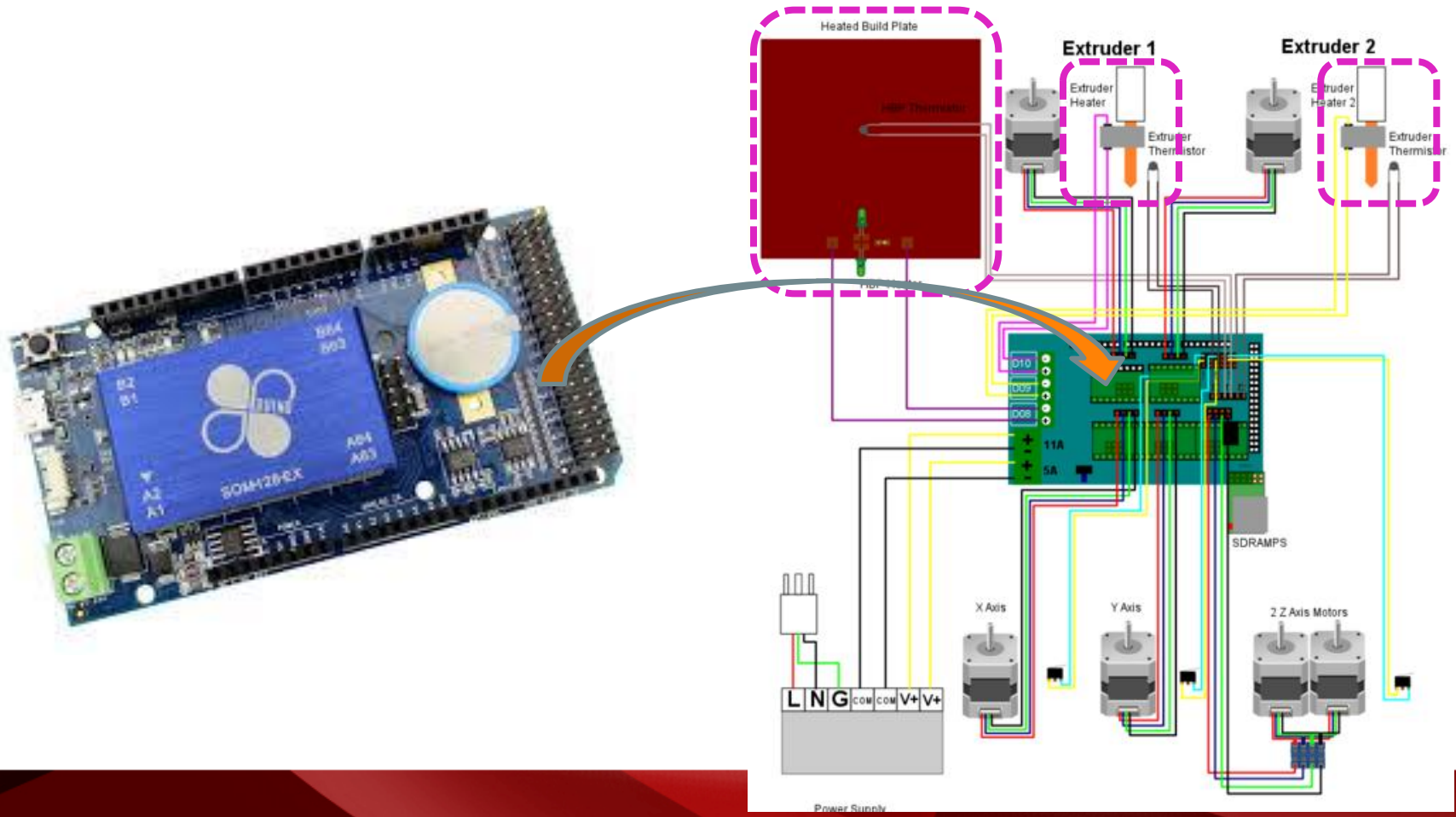
```
case 0: // Prepare TEMP_0
#if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  #if TEMP_0_PIN > 7
    ADCSRB = 1<<MUX5;
  #else
    ADCSRB = 0;
  #endif
  ADMUX = ((1 << REFS0) | (TEMP_0_PIN & 0x07));
  ADCSRA |= 1<<ADSC; // Start conversion
#endif
lcd_buttons_update();
temp_state = 1;
break;
case 1: // Measure TEMP_0
#if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  raw_temp_0_value += ADC;
#endif
#ifdef HEATER_0_USES_MAX6675 // TODO remove the blocking
  raw_temp_0_value = read_max6675();
#endif
temp_state = 2;
break;
```

analogRead () 與 Arduino 的 function 相容

```
#if defined(TEMP_0_PIN) && (TEMP_0_PIN > -1)
  raw_temp_0_value += analogRead(TEMP_0_PIN); //ADC
#endif
#ifdef HEATER_0_USES_MAX6675 // TODO remove the blockin
  raw_temp_0_value = read_max6675();
#endif
temp_state = 2;
```

86Duino ADC 的移植

- 按照上敘的步驟 ，便完成 **Marlin 對 86Duino ADC 的移植**



86Duino SPI 的移植

86Duino SPI 的移植

- 86Duino 的 SPI 與 Arduino 的 SPI 差異
  - 86duino SPI 最高速為 50 MHz, Arduino SPI 最高速為 4 MHz
    - Marlin 設定 SPI 的速度是 1MHz
  - 86duino SPI 輸出的高電位為 3.3 V, Arduino SPI 輸出的高電位為 5V

86Duino SPI 的移植

1. 改寫 tp_init() 對 SPI 的設定(temperature.cpp)

原來對 ATMega 暫存器作操作

```
#ifdef HEATER_0_USES_MAX6675
  #ifndef SDSUPPORT
    SET_OUTPUT(MAX_SCK_PIN);
    WRITE(MAX_SCK_PIN,0);

    SET_OUTPUT(MAX_MOSI_PIN);
    WRITE(MAX_MOSI_PIN,1);

    SET_INPUT(MAX_MISO_PIN);
    WRITE(MAX_MISO_PIN,1);
  #endif

  SET_OUTPUT(MAX6675_SS);
  WRITE(MAX6675_SS,1);
#endif
```

SPI.begin() 對 SPI 初始化
SPI.setDataMode() 設定 SPI 的 mode
SPI.setClockDivider() 設定 SPI 速度
3 個 funcion 都與 Arduino 的 function 相容

```
#ifdef HEATER_0_USES_MAX6675
  SPI.begin();
  SPI.setDataMode(SPI_MODE0);
  SPI.setClockDivider(SPI_CLOCK_DIV100); // 1M Hz
#endif
```

86Duino SPI 的移植

2. 改寫 readMax6675() 對 Max6675 做讀值 (temperature.cpp)

原來對 **ATMega** 暫存器作操作

```
#ifdef  PRR
 PRR &= ~(1<<PRSPI);
#elif defined PRR0
 PRR0 &= ~(1<<PRSPI);
#endif

SPCR = (1<<MSTR) | (1<<SPE) | (1<<SPR
// enable TT_MAX6675
WRITE(MAX6675_SS, 0);

// ensure 100ns delay - a bit extra is fine
asm("nop");//50ns on 20Mhz, 62.5ns on 16Mhz
asm("nop");//50ns on 20Mhz, 62.5ns on 16Mhz

// read MSB
SPDR = 0;
for (;(SPSR & (1<<SPIF)) == 0;);
max6675_temp = SPDR;
max6675_temp <<= 8;

// read LSB
SPDR = 0;
```

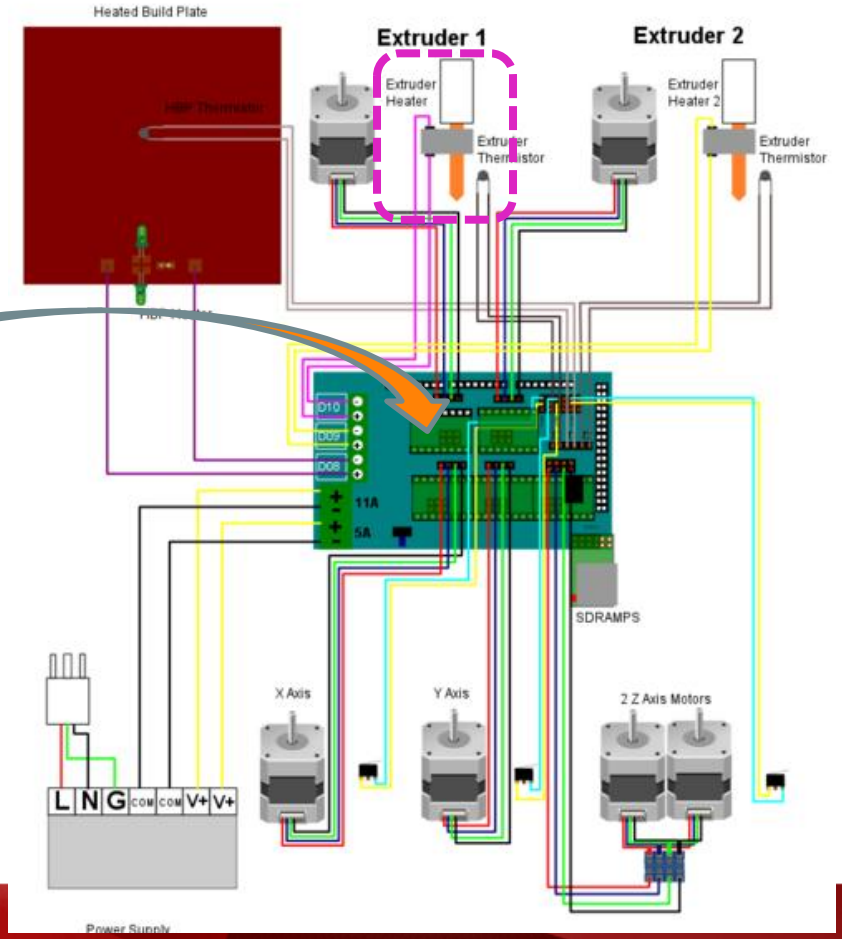
```
for (;(SPSR & (1<<SPIF)) == 0;);
max6675_temp |= SPDR;

// disable TT_MAX6675
WRITE(MAX6675_SS, 1);

if (max6675_temp & 4)
{
 // thermocouple open
 max6675_temp = 2000;
}
else
{
 max6675_temp = max6675_temp >> 3;
}

return max6675_temp;
```

相對應原來讀取 **max6675** 的行為

```
SPI.SPICS(LOW);

max6675_temp = SPI.transfer(0x0);
max6675_temp <<= 8;
max6675_temp |= SPI.transfer(0x0);
SPI.SPICS(HIGH);
if (max6675_temp & 4) {
 // thermocouple open
 max6675_temp = 2000;
}
else {
 max6675_temp = max6675_temp >> 3;
}
return max6675_temp;
```

86Duino SPI 的移植

- 按照上敘步驟 ，便完成 **Marlin 對 86Duino SPI 的移植**



86Duino serial port 的移植

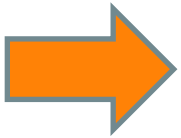
86Duino serial port 的移植

- 移植 86Duino serial port 的做法
  - 雖然 86Duino serial lib 與 Arduino serial lib 相容，但是 Marlin serial lib 只有底層的部分與 Arduino serial lib 相同(如 P.16 所敘)
  - 所以做法是，將 Marlin serial lib 底層部分的程式改為 86Duino serial lib 的程式

86Duino serial port 的移植

1. 移植 begin() 的設定 (MarlinSerial.cpp)

```
void MarlinSerial::begin(long baud)
{
  uint16_t baud_setting;
  bool useU2X = true;

#if F_CPU == 16000000UL && SERIAL_PORT == 0
  // hardcoded exception for compatibility with the bootloader shipped
  // with the Duemilanove and previous boards and the firmware on the 8U2
  // on the Uno and Mega 2560.
  if (baud == 57600) {
    useU2X = false;
  }
#endif

  if (useU2X) {
    M_UCSRxA = 1 << M_U2Xx;
    baud_setting = (F_CPU / 4 / baud - 1) / 2;
  } else {
    M_UCSRxA = 0;
    baud_setting = (F_CPU / 8 / baud - 1) / 2;
  }

  // assign the baud_setting, a.k.a. ubbr (USART Baud Rate Register)
  M_UBRRxH = baud_setting >> 8;
  M_UBRRxL = baud_setting;

  sbi(M_UCSRxB, M_RXENx);
  sbi(M_UCSRxB, M_TXENx);
  sbi(M_UCSRxB, M_RXCIEx);
}
```

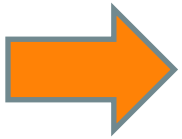


```
void MarlinSerial::begin(long baud)
{
  com_SetUSBPins(7,0,7,1);
  if ((handle = com_Init(USB_COM)) == NULL)
  {
      printf("COM init fail!!\n");
  }
  com_SetTimeOut(handle,2000L);
  //com_SetBPS(handle, baud);
  //com_SetFormat(handle,BYTESIZE8|NOPARITY|STOPBIT1);
  com_FlushTxQueue(handle);
  com_FlushRxQueue(handle);
  while(com_Ready(handle) == false);
  delay_ms(100);
}
```

86Duino serial port 的移植

2. 移植 end() 的設定 (MarlinSerial.cpp)

```
void MarlinSerial::end()
{
  cbi(M_UCSRxB, M_RXENx);
  cbi(M_UCSRxB, M_TXENx);
  cbi(M_UCSRxB, M_RXCIEx);
}
```

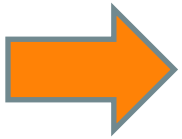


```
void MarlinSerial::end()
{
  com_FlushWFIFO(handle);
  com_Close(handle);
}
```

86Duino serial port 的移植

3. 移植 flush() 的設定 (MarlinSerial.cpp)

```
void MarlinSerial::flush()
{
  // don't reverse this or there may be problems if the RX interrupt
  // occurs after reading the value of rx_buffer_head but before writing
  // the value to rx_buffer_tail; the previous value of rx_buffer_head
  // may be written to rx_buffer_tail, making it appear as if the buffer
  // don't reverse this or there may be problems if the RX interrupt
  // occurs after reading the value of rx_buffer_head but before writing
  // the value to rx_buffer_tail; the previous value of rx_buffer_head
  // may be written to rx_buffer_tail, making it appear as if the buffer
  // were full, not empty.
  rx_buffer.head = rx_buffer.tail;
}
```



```
void MarlinSerial::flush()
{
  com_FlushTxQueue(handle);
}
```

86Duino serial port 的移植

- 改寫上敘 **function** 的內容，便完成 **Marlin** 對 **86Duino serial** 的移植

86Duino 移植 Marlin 的 timer 中斷

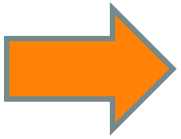
86Duino 移植 Marlin 的 timer 中斷

- **86Duino 移植 timer 中斷的做法**
  - 利用 **86Duino** 的 **PWM timer** 中斷取代 **Marlin** 的 **timer** 中斷
  - **86Duino PWM** 的精度為 **10 ns**，精度足夠用來當作 **Marlin** 的 **timer**
  - 設定 **86Duino** 每個 **PWM** 的週期結束發一次中斷，做為 **Marlin** 的 **timer** 中斷

86Duino 移植 Marlin 的 timer 中斷

1. st_init() 初始化 timer1 的改寫 (stepper.cpp)

```
// waveform generation = 0100 = CTC
TCCR1B &= ~(1<<WGM13);
TCCR1B |=  (1<<WGM12);
TCCR1A &= ~(1<<WGM11);
TCCR1A &= ~(1<<WGM10);

// output mode = 00 (disconnected)
TCCR1A &= ~(3<<COM1A0);
TCCR1A &= ~(3<<COM1B0);

// Set the timer pre-scaler
// Generally we use a divider of 8, resulting in a 2MHz timer
// frequency on a 16MHz MCU. If you are going to change this, be
// sure to regenerate speed_lookuptable.h with
// create_speed_lookuptable.py
TCCR1B = (TCCR1B & ~(0x07<<CS10)) | (2<<CS10);

OCR1A = 0x4000;
TCNT1 = 0;
ENABLE_STEPPER_DRIVER_INTERRUPT();
```



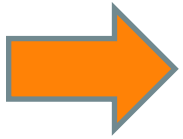
```
StepperMCM_Init();
InstallStepperISR();
SetInt_STEP();
ENABLE_STEPPER_DRIVER_INTERRUPT();
```

86Duino 移植 Marlin 的 timer 中斷

2. ISR() 設定 timer1 下次進中斷的時間 (stepper.cpp)

– block 為空時，下次進中斷的時間 是固定的

```
if (current_block == NULL) {
  // Anything in the buffer?
  current_block = plan_get_current_block();
  if (current_block != NULL) {
    current_block->busy = true;
    trapezoid_generator_reset();
    counter_x = -(current_block->step_event_count >>
    counter_y = counter_x;
    counter_z = counter_x;
    counter_e = counter_x;
    step_events_completed = 0;

    #ifdef Z_LATE_ENABLE
      if(current_block->steps_z > 0) {
        enable_z();
        OCR1A = 2000; //1ms wait
        return;
      }
    #endif
  }
  else {
    OCR1A=2000; // 1kHz.
  }
}
```



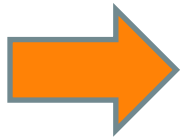
```
    current_block->busy = true;
    trapezoid_generator_reset();
    counter_x = -(current_block->step_event_count >> 1);
    counter_y = counter_x;
    counter_z = counter_x;
    counter_e = counter_x;
    step_events_completed = 0;

    #ifdef Z_LATE_ENABLE
      if(current_block->steps_z > 0) {
        enable_z();
        SetTimer_STEP(2000L); //OCR1A = 2000; //1ms wait
        return;
      }
    #endif

  }
  else {

    SetTimer_STEP(2000L); //OCR1A=2000; // 1kHz.
  }
```

86Duino 移植 Marlin 的 timer 中斷

3. ISR() 設定 timer1 下次進中斷的時間 (stepper.cpp)

– block 為加速段時，設定下次進中斷的時間

```
// Calculare new timer value
unsigned short timer;
unsigned short step_rate;
if (step_events_completed <= (unsigned long int)current_block->accel

MultiU24X24toH16(acc_step_rate, acceleration_time, current_block->
acc_step_rate += current_block->initial_rate;

// upper limit
if(acc_step_rate > current_block->nominal_rate)
  acc_step_rate = current_block->nominal_rate;

// step_rate to timer interval
timer = calc_timer(acc_step_rate);
OCR1A = timer;
acceleration_time += timer;
```



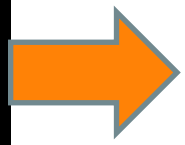
```
unsigned short timer;
unsigned short step_rate;
if (step_events_completed <= (unsigned long int)current_block->acceler

MultiU24X24toH16(acc_step_rate, acceleration_time, current_block->a
acc_step_rate += (unsigned short) current_block->initial_rate;

// upper limit
if(acc_step_rate > (unsigned short) current_block->nominal_rate)
  acc_step_rate = (unsigned short) current_block->nominal_rate;

// step_rate to timer interval
timer = calc_timer(acc_step_rate);
SetTimer_STEP((unsigned long)timer); //OCR1A = timer;
acceleration_time += (unsigned long)timer;
```

86Duino 移植 Marlin 的 timer 中斷

4. ISR() 設定 timer1 下次進中斷的時間 (stepper.cpp)

– block 為減速段時，設定下次進中斷的時間

```
}
else if (step_events_completed > (unsigned long int)current_block->decel
  MultiU24X24toH16(step_rate, deceleration_time, current_block->acceler

  if(step_rate > acc_step_rate) { // Check step_rate stays positive
    step_rate = current_block->final_rate;
  }
  else {
    step_rate = acc_step_rate - step_rate; // Decelerate from aceleration end point.
  }

  // lower limit
  if(step_rate < current_block->final_rate)
    step_rate = current_block->final_rate;

  // step_rate to timer interval
  timer = calc_timer(step_rate);
  OCR1A = timer;
  deceleration_time += timer;
```



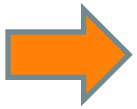
```
if(step_rate > acc_step_rate) { // Check step_rate stays positive
  step_rate = (unsigned short) current_block->final_rate;
}
else {
  step_rate = acc_step_rate - step_rate; // Decelerate from aceleration
}

// lower limit
if(step_rate < (unsigned short) current_block->final_rate)
  step_rate = (unsigned short) current_block->final_rate;

// step_rate to timer interval
timer = calc_timer(step_rate);
SetTimer_STEP((unsigned long)timer); //OCR1A = timer;
deceleration_time += (unsigned long) timer;
```

86Duino 移植 Marlin 的 timer 中斷

5. ISR() 設定 timer1 下次進中斷的時間 (stepper.cpp)

– block 為nominal 段時，設定下次進中斷的時間

```
else {
  OCR1A = OCR1A_nominal;
  // ensure we're running at the correct step rate, even if we just cam
  step_loops = step_loops_nominal;
}

// If current block is finished, reset pointer
```



```
else {
  SetTimer_STEP((unsigned long)OCR1A_nominal); //OCR1A = C
  // ensure we're running at the correct step rate, even if we just came off an acceleration
  step_loops = step_loops_nominal;
}
```

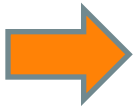
- 按照上敘步驟 ，完成 timer1 的移植

86Duino 移植 Marlin 的 timer 中斷

5. ISR() 設定 timer1 下次進中斷的時間 (stepper.cpp)

- block 為nominal 段時，設定下次進中斷的時間

```
else {
  OCR1A = OCR1A_nominal;
  // ensure we're running at the correct step rate, even if we just cam
  step_loops = step_loops_nominal;
}

// If current block is finished, reset pointer
```



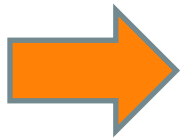
```
else {
  SetTimer_STEP((unsigned long)OCR1A_nominal); //OCR1A = C
  // ensure we're running at the correct step rate, even if we just came off an acceleration
  step_loops = step_loops_nominal;
}
```

86Duino 移植 Marlin 的 timer 中斷

1. tp_init() 初始化 timer0 的改寫 (temperature.cpp)

– 設定進中斷的時間 是固定的，約 1ms

```
  #endif
#endif

// Use timer0 for temperature measurement
// Interleave temperature interrupt with millies interrupt
OCR0B = 128;
TIMSK0 |= (1<<OCIE0B);

// Wait for temperature measurement to settle
delay(250);

#ifdef HEATER_0_MINTEMP
  minttemp[0] = HEATER_0_MINTEMP;
  while(analog2temp(minttemp_raw[0], 0) < HEATER_0_MINTEM
#if HEATER_0_RAW_LO_TEMP < HEATER_0_RAW_HI_TEMP
```



```
#ifdef HEATER_0_USES_MAX6675
  SPI.begin();
  SPI.setDataMode(SPI_MODE0);
  SPI.setClockDivider(SPI_CLOCK_DIV100); // 1M Hz
#endif

HeatMCM_Init();
InstallHeatISR();
SetInt_Heat0();
EnableInt_Heat0();
// Use timer0 for temperature measurement
// Interleave temperature interrupt with millies interrupt

// Wait for temperature measurement to settle
delay(250);
```

86Duino 移植 Marlin 的 timer 中斷

2. **86Duino 在 ISR() 裡 reload 進中斷的時間 (temperature.cpp)**

```
  if(current_temperature_bed_raw <= bed_maxttemp_raw) {
#else
  if(current_temperature_bed_raw >= bed_maxttemp_raw) {
#endif
    target_temperature_bed = 0;
    bed_max_temp_error();
  }
#endif
 }
 ReloadTimer_Heat0();
}
```

- 按照上敘步驟 ，完成 **timer0** 的移植

86Duino 移植 Marlin 的 timer 中斷

- 以上是初步移植 **Marlin timer** 中斷的改寫
  - 之後可能為了縮短中斷的 **latency** 時間及加快存取 **I/O** 的速度，做一些程式上的改寫，增加執行 **Marlin** 的效能

T H A N K Y O U