SSD1306 128*64 OLED Module

GND： 電源地

VCC： 2.2V~5.5V

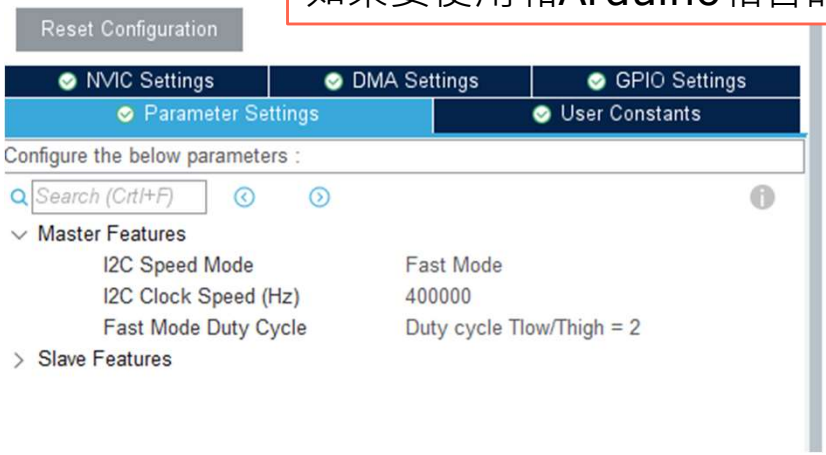
SCL： CLK 時鐘（高電平2.2V~5.5V）

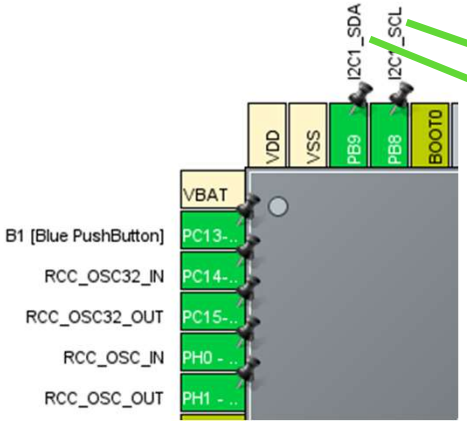
SDA： MOSI 資料（高電平2.2V~5.5V）

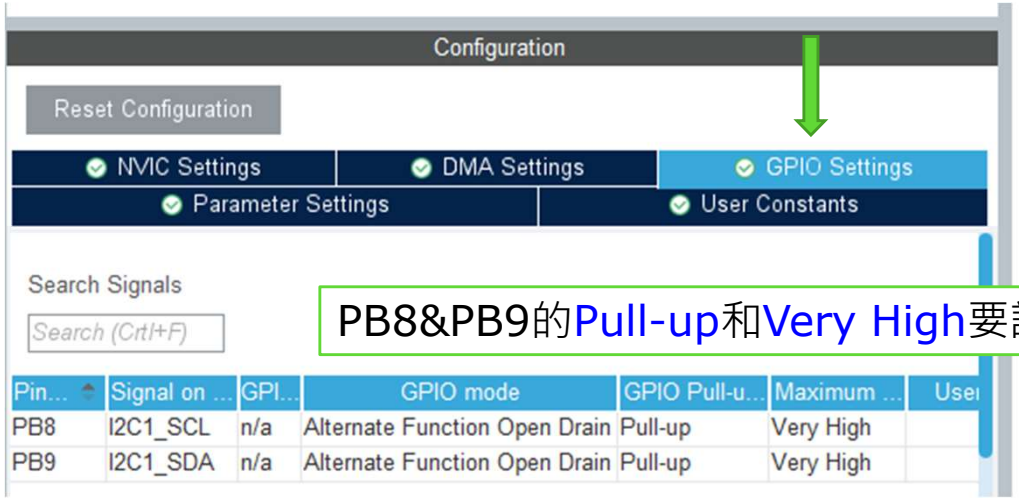
IIC 位置:0x78

*.ioc檔設定，容易出錯的地方

如果要使用和Arduino相容的腳位，I²C1在PB8 & PB9







PB8&PB9的Pull-up和Very High要設起來

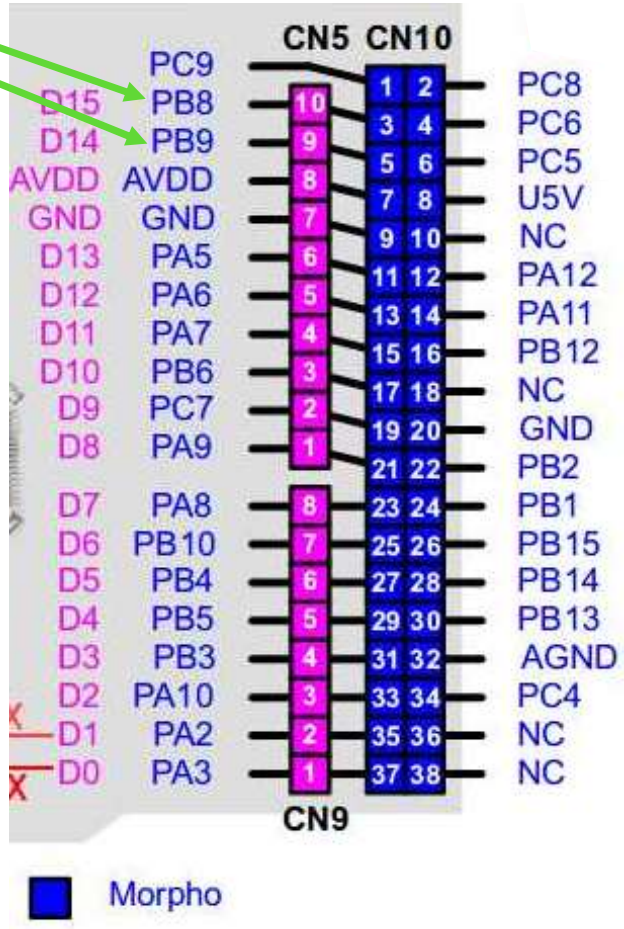
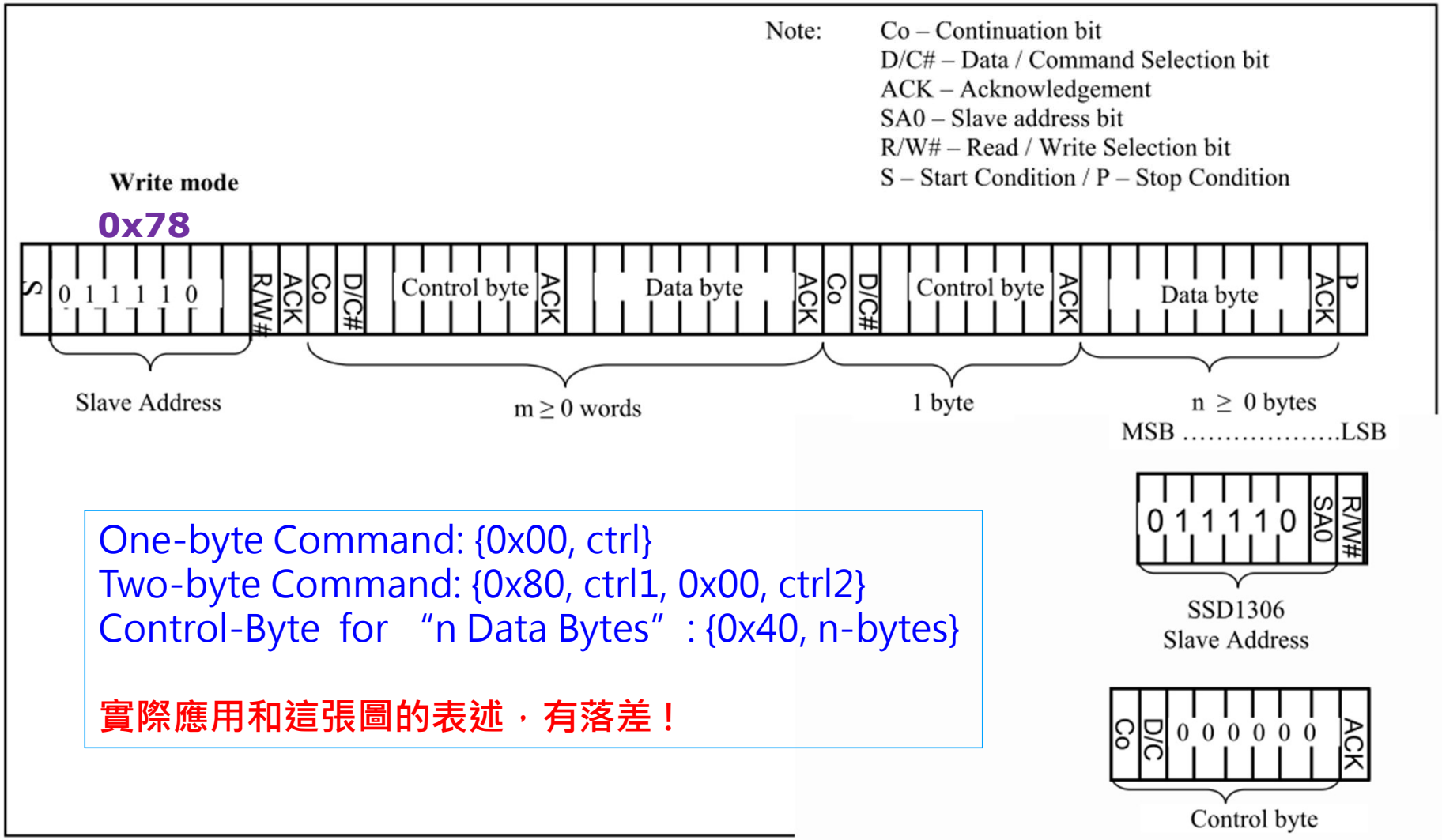


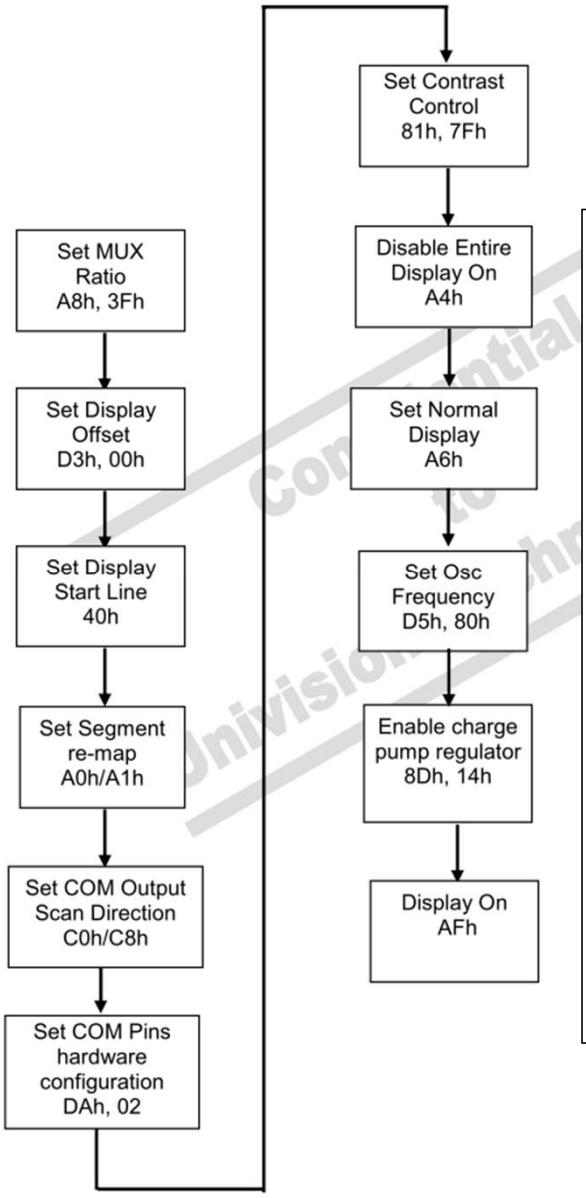
Figure 8-7 : I²C-bus data format



One-byte Command: {0x00, ctrl}
Two-byte Command: {0x80, ctrl1, 0x00, ctrl2}
Control-Byte for "n Data Bytes" : {0x40, n-bytes}

實際應用和這張圖的表述，有落差！

Figure 2 : Software Initialization Flow Chart

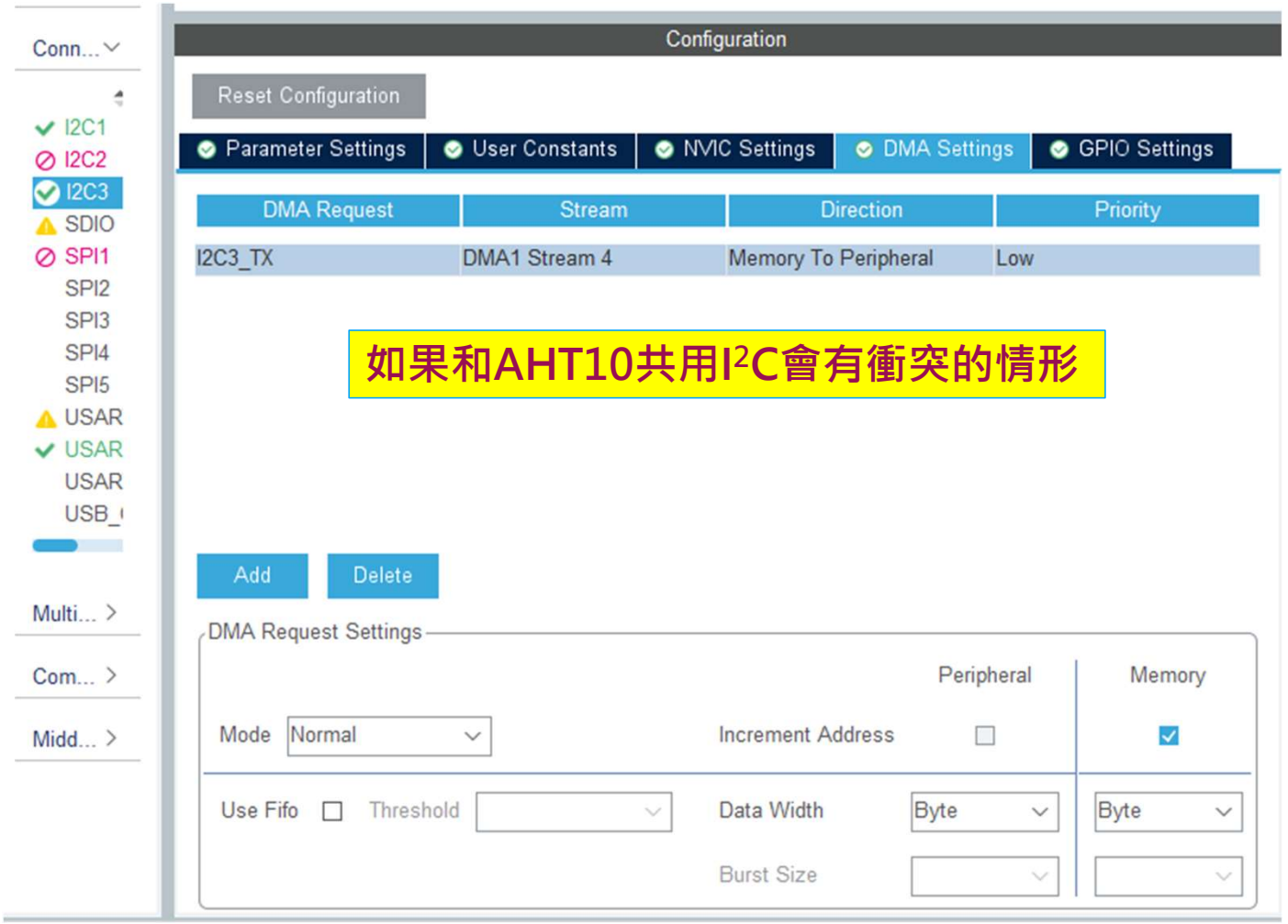


SSD1306 Datasheet's Command Sequence

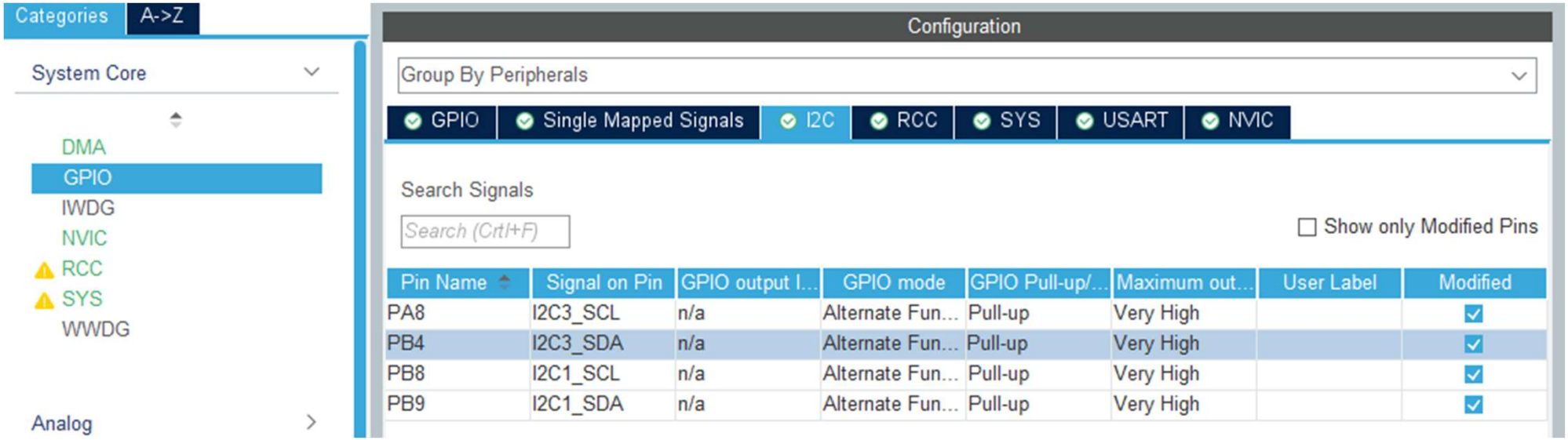
```
Set MUX Ratio                              A8h, 3Fh
Set Display Offset                         D3h, 00h
Set Display Start Line                     40h
Set Segment re-map                         A0h/A1h
Set COM Output Scan Direction              C0h/C8h
Set COM Pins hardware configuration        DAh, 02h
Set Contrast Control                       81h, 7Fh
Disable Entire Display On                  A4h
Set Normal Display                         A6h
Set Osc Frequency                          D5h, 80h
Enable charge pump regulator               8Dh, 14h
Display On                                 AFh
```

DMA有助於大量且連續的資料移轉

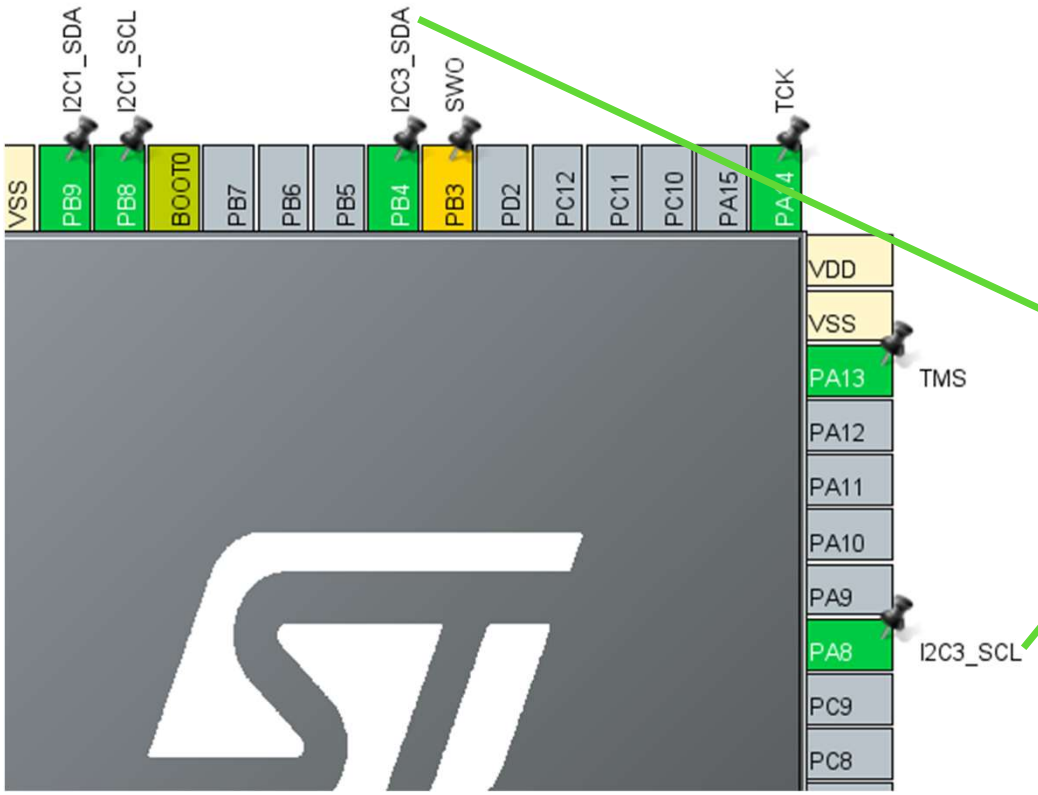
宣告一個內部的Frame Buffer，修正完資料後，再一次性更新SSD1306的GDDRAM

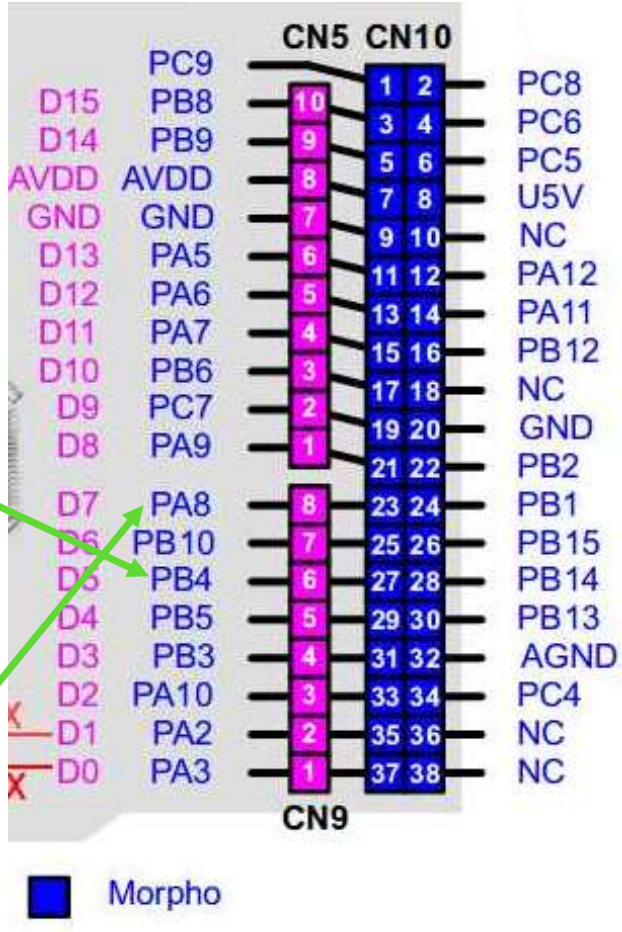


記得在System Core的GPIO設定中將I²C腳位Pull-up

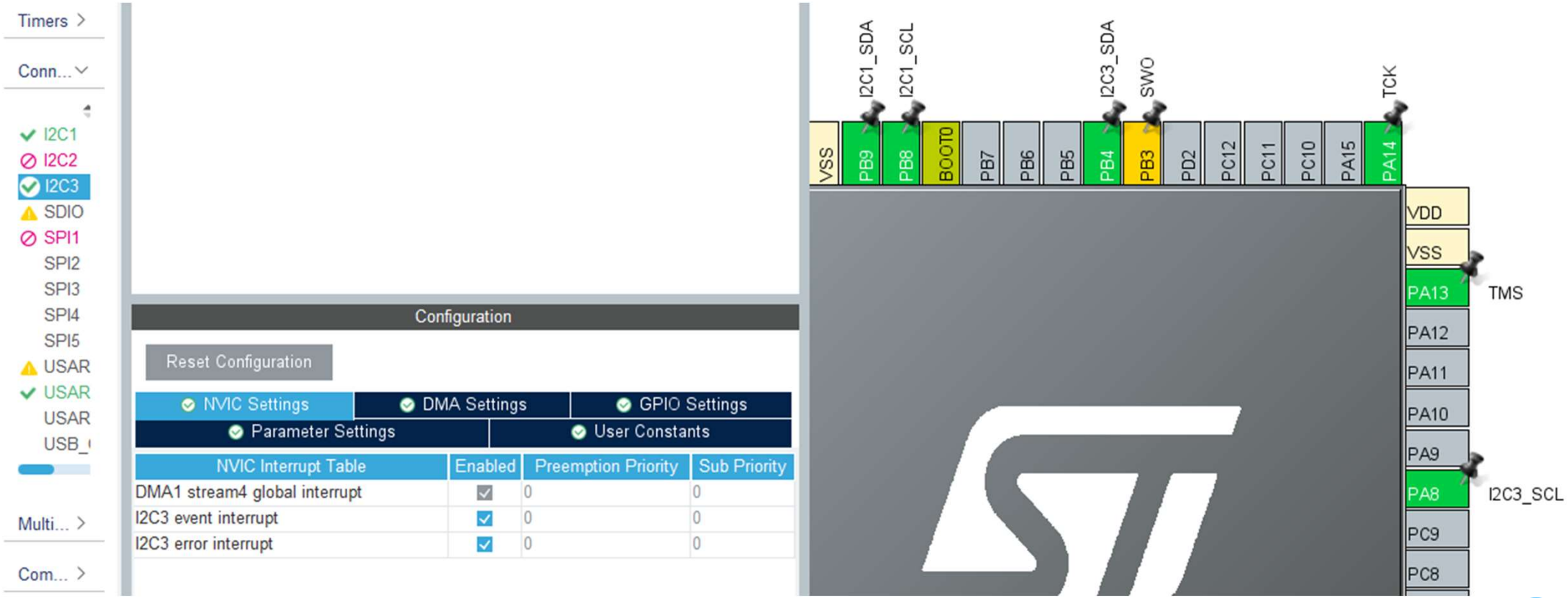


I²C3的腳位設定


I2C1_SDA
I2C1_SCL
I2C3_SDA
SWO
TCK
VSS
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PD2
PC12
PC11
PC10
PA15
PA14
VDD
VSS
PA13
TMS
PA12
PA11
PA10
PA9
PA8
I2C3_SCL
PC9
PC8

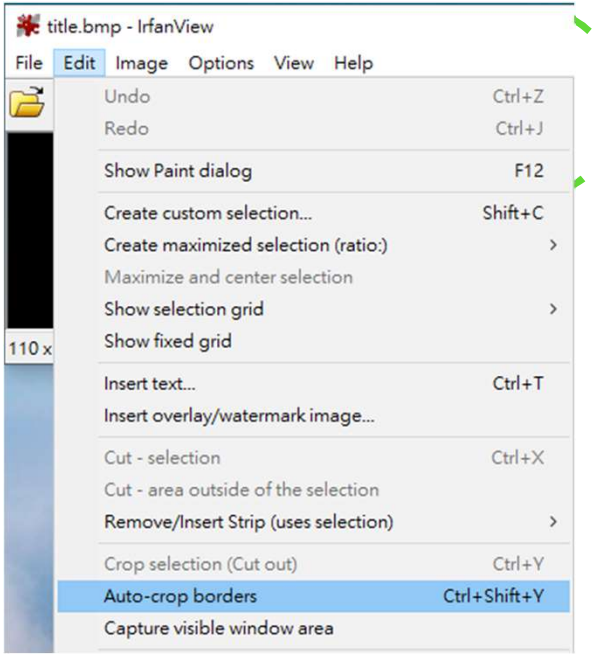

CN5 CN10
PC9
D15 PB8
D14 PB9
AVDD AVDD
GND GND
D13 PA5
D12 PA6
D11 PA7
D10 PB6
D9 PC7
D8 PA9
D7 PA8
PB10
PB4
D4 PB5
D3 PB3
D2 PA10
D1 PA2
D0 PA3
CN9
1 2 PC8
3 4 PC6
5 6 PC5
7 8 U5V
9 10 NC
11 12 PA12
13 14 PA11
15 16 PB12
17 18 NC
19 20 GND
21 22 PB2
23 24 PB1
25 26 PB15
27 28 PB14
29 30 PB13
31 32 AGND
33 34 PC4
35 36 NC
37 38 NC
Morpho

NVIC Settings; Interrupt一定要打開！



℃ %RH

- 使用Times New Roman的18號字型，在放大倍率100%的狀態下；“shift+win-key+s” 從螢幕上剪下來
- 貼到IrfanView中，用Auto-crop borders去邊，存成*.bmp
- 在小畫家中加上垂直分隔線，然後儲存成單色點陣圖



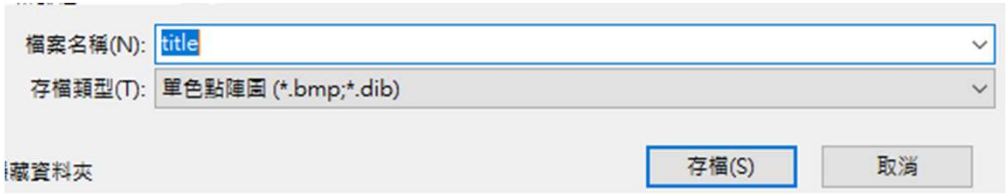


image2cpp http://javl.github.io/image2cpp/

image2cpp is a simple tool to change images into byte arrays (or your array back into an image) for use with Arduino and (monochrome) displays such as OLEDs. It was originally made to work with the Adafruit OLED library. An example sketch for Arduino and this library can be found here.

More info (and credits) can be found in the Github repository. This is also where you can report any issues you might come across.

This tool also works offline. Simply save this page to your computer and open the file in your browser.

1. Select image

選擇檔案 title.bmp

title.bmp remove

or

1. Paste byte array

128 x 64 px

Read as horizontal Read as vertical

2. Image Settings

Canvas size(s): title.bmp (file resolution: 110 x 16)
128 x 16 glyph remove

Background color: ○ White ○ Black ◉ Transparent

Invert image colors ☐

Brightness / alpha threshold: 128
0 - 255; if the brightness of a pixel is above the given level the pixel becomes white, otherwise they become black. When using alpha, opaque and transparent are used instead.

Scaling original size

Center: ☑ horizontally ☐ vertically

Rotate image: ☐ rotate 180 degrees

Flip: ☑ horizontally ☐ vertically

Note: centering the image only works when using a canvas larger than the original image.

---

3. Preview

因為SSD1306原點被設定在右上角

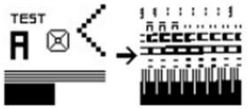
---

4. Output

Code output format plain bytes

Draw mode: Vertical - 1 bit per pixel

If your image looks all messed up on your display, like the image below, try using a different mode.



Generate code

https://oleddisplay.squix.ch/#/home

字型檔是在此產生，根據Library Version >=3.0.0的選項，以Byte為單位填入GDDSRAM中

SSD1306 Tools | Home

Font Converter

Select the font family and other settings to generate a new font file

Preview Display

OLED 0.96" (128x64)

Font Family

Dialog

Style

Plain

Size

24

Library Version

>=3.0.0

Create | Buy Display (Affiliate) | Teleport a Beer!

Do you like this font generator? If you do consider **spending me a beer** for my hardwork or use this link for your next hardware order at Banggood: Go to my blog to find more of my projects... Scroll down to see the created header file content...

C Font File

Copy this text into you font header file (e.g. font.h) in the Arduino IDE

```
// Created by http://oleddisplay.squix.ch/ Consider a donation
// In case of problems make sure that you are using the font file with the correct version!
const char Dialog_plain_24[] PROGMEM = {
    0x18, // Width: 24
    0x1D, // Height: 29
    0x20, // First Char: 32
    0xE0, // Numbers of Chars: 224

    // Jump Table:
    0xFF, 0xFF, 0x00, 0x08,  // 32:65535
```

設定**右上角**為 原點；[X,Y]：[128_{Bytes}*8_{Bytes}]

	xByte=127+Row*128			...		Row*128
0	Byte-127	Byte-126	Byte-125	~	Byte-2	Byte-1	Byte-0
	Byte-255	Byte-254	Byte-253	~	Byte-130	Byte-129	Byte-128
	Byte-383	Byte-382	Byte-381	~	Byte-258	Byte-257	Byte-256
yByte...	Byte-511	Byte-510	Byte-509	~	Byte-386	Byte-385	Byte-384
	Byte-639	Byte-638	Byte-637	~	Byte-514	Byte-513	Byte-512
	Byte-767	Byte-766	Byte-765	~	Byte-642	Byte-641	Byte-640
	Byte-895	Byte-894	Byte-893	~	Byte-770	Byte-769	Byte-768
7	Byte-1023	Byte-1022	Byte-1021	~	Byte-898	Byte-897	Byte-896

Byte
Bit-0
Bit-1
Bit-2
Bit-3
Bit-4
Bit-5
Bit-6
Bit-7