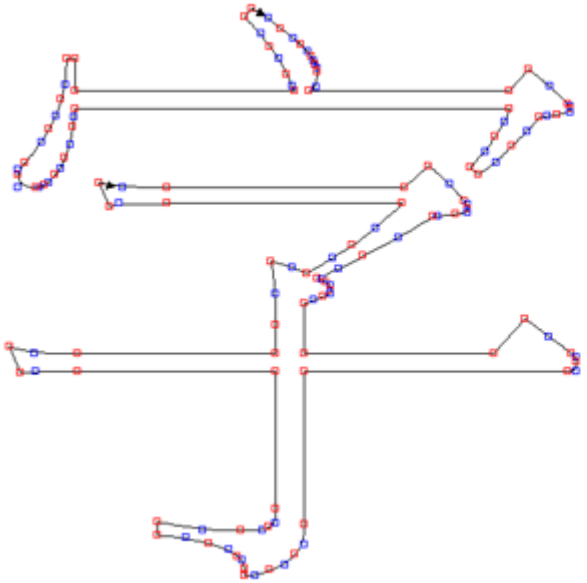
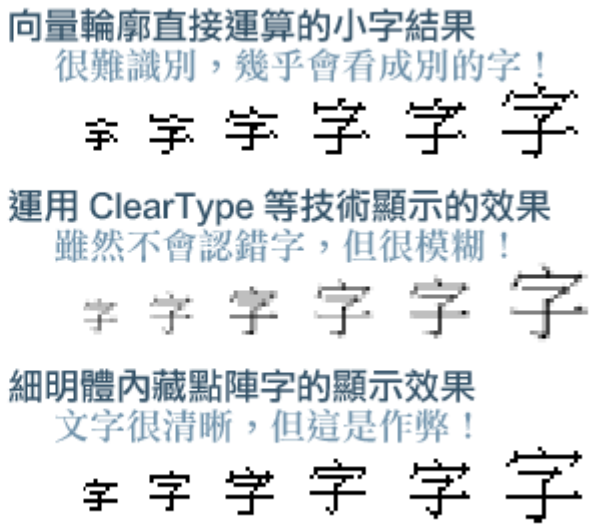
ARM GPIO 7-Segment Control

Part of this document is by
Prof. Shiao-Li Tsao
CS Dept., NCTU

A story about GPIO

- 螢幕3個顏色，24byte表示一個pixel。
- 中文字的表示
  - 最早是用Big5大五碼，二個碼(double byte)16bit+16bit組出一個中文字。
  - 後來是24x24點陣，清楚，但放大就會有鋸齒
  - 現在是用向量字，放大也漂亮。





Single 7-Segment to IO Ports

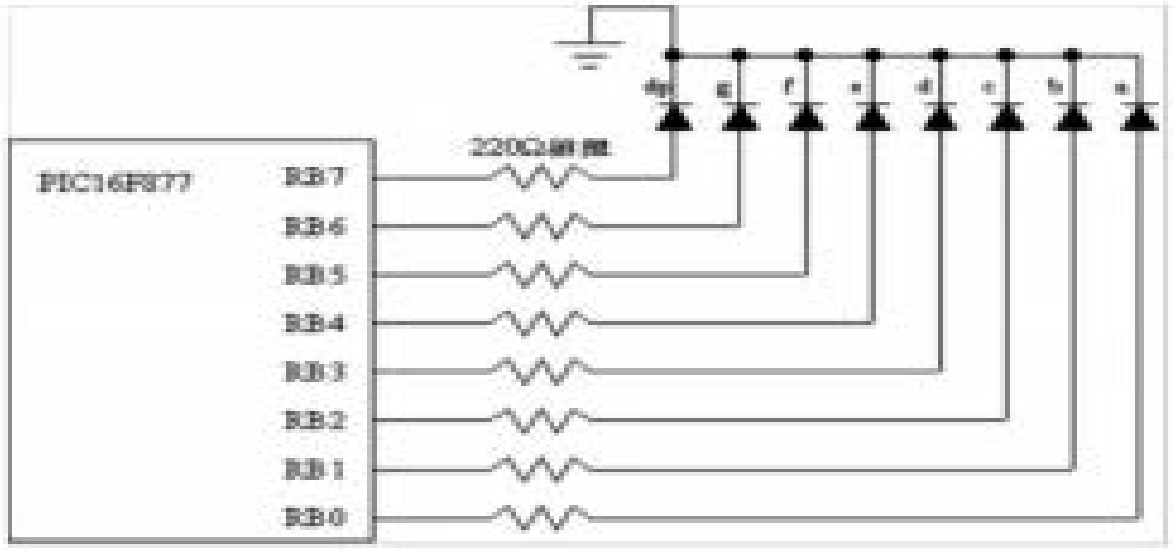
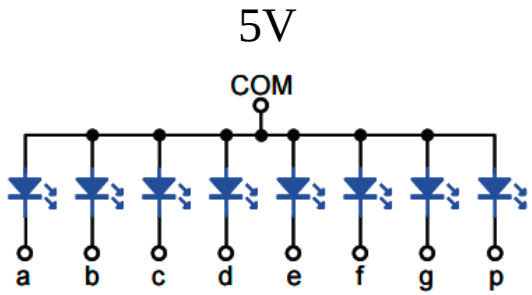
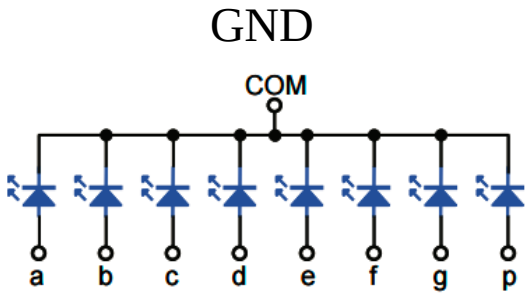


圖 7 共陰極電路圖

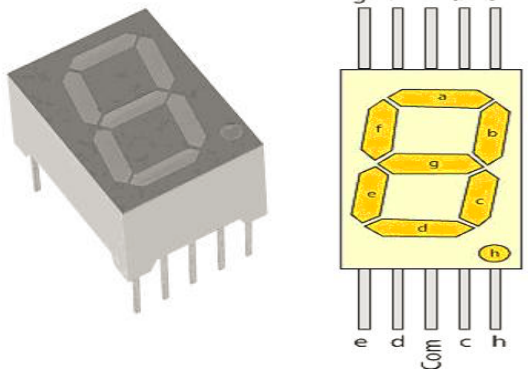


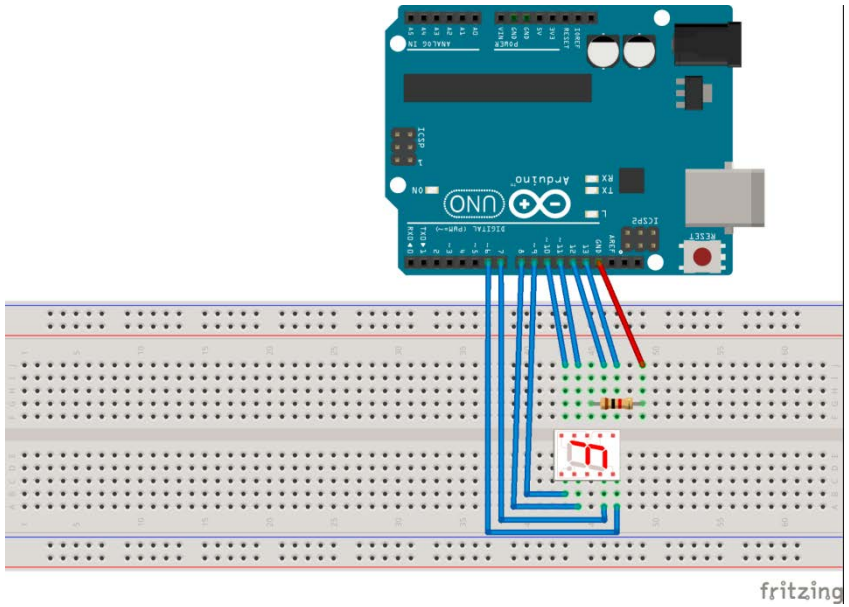
(a)共陽極結構

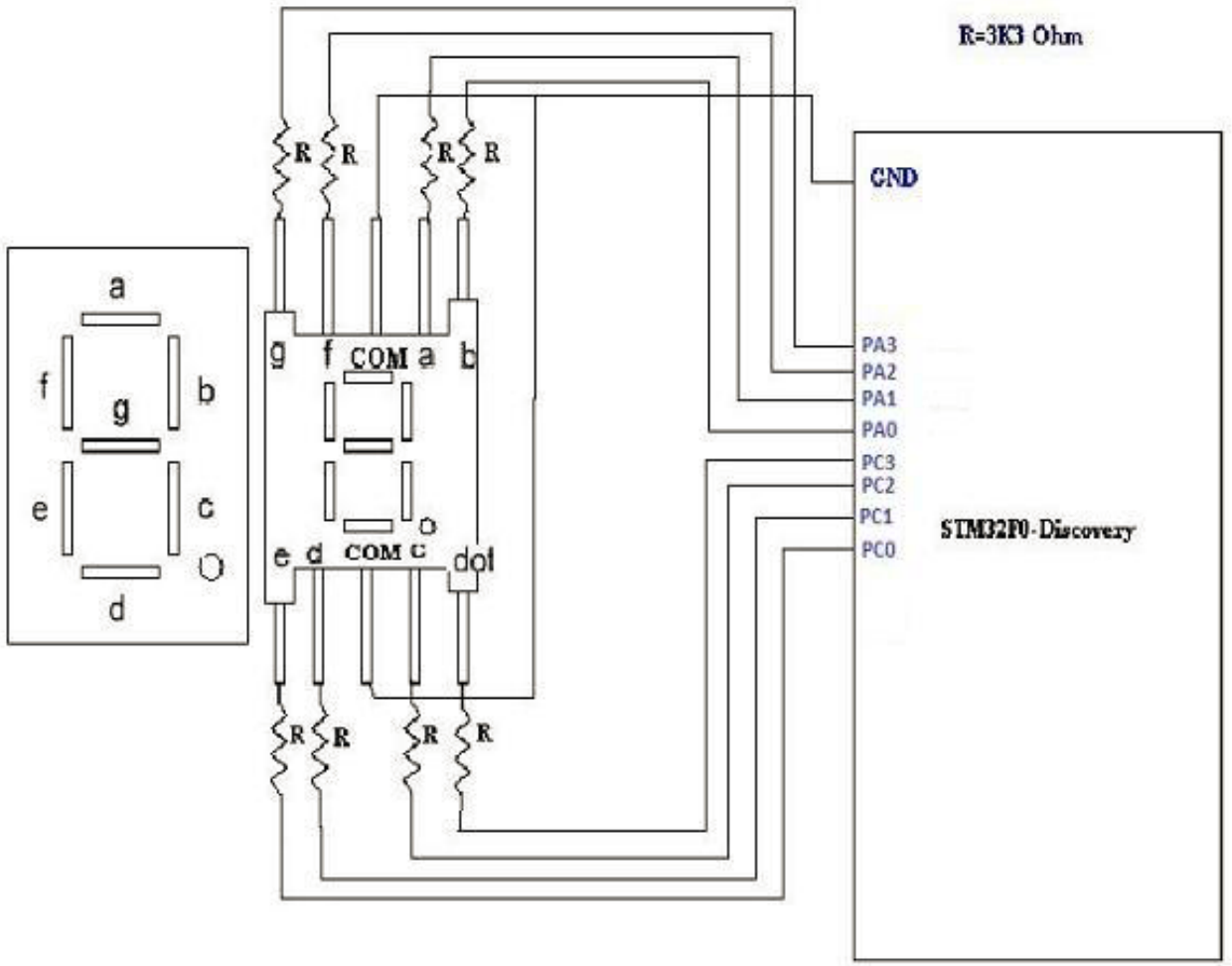


(b)共陰極結構

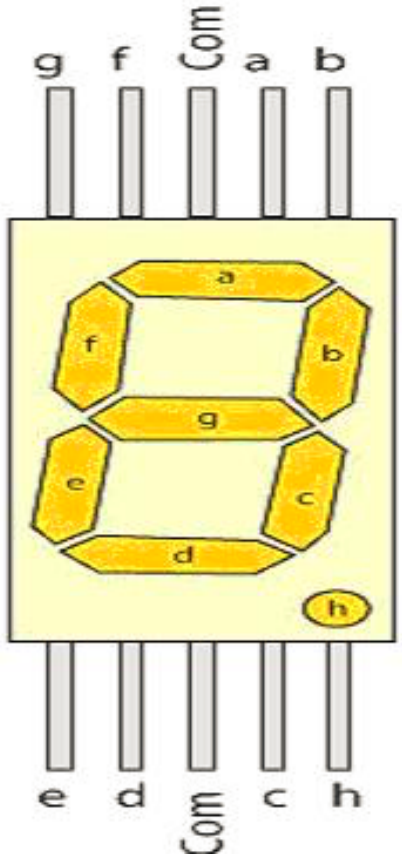
Single 7-Segment







Single 7-Segment Display Coding



g	f	e	d	dp	c	b	a	number	hex code PORT A		PORT C
0	1	1	1	0	1	1	1	0	0X0007	&	0X0007
0	0	0	0	0	1	1	0	1	0X0001	&	0X0004
1	0	1	1	0	0	1	1	2	0X000B	&	0X0003
1	0	0	1	0	1	1	1	3	0X000B	&	0X0006
0	1	1	0	0	1	1	0	4	0X000D	&	0X0004
1	1	0	1	0	1	0	1	5	0X000E	&	0X0006
1	1	1	1	0	1	0	1	6	0X000E	&	0X0007
0	0	0	0	0	1	1	1	7	0X0003	&	0X0004
1	1	1	1	0	1	1	1	8	0X000F	&	0X0007
1	1	0	1	0	1	1	1	9	0X000F	&	0X0006

GPIO port output data register (GPIOx_ODR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data bit (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A..F).

Reference manual STM32L4x6 IO周邊register操作相關參考文件 =>

P267 GPIO port output data register (GPIOx_ODR) (x = A..H) 注意ODR只有後面八個bit有用,所以一定是00XX

Single 7-segment C Code to Control ODR

```
int main(void)
{
Init_GPIO();
while(1)
{
GPIOA->ODR = 0x0007;
GPIOC->ODR = 0x0007;
delay_ms(1000);
GPIOA->ODR = 0x0001;
GPIOC->ODR = 0x0004;
delay_ms(1000);
GPIOA->ODR = 0x000B;
GPIOC->ODR = 0x0003;
delay_ms(1000);
…
}
}
```

0111
0111
Display 0

This one?

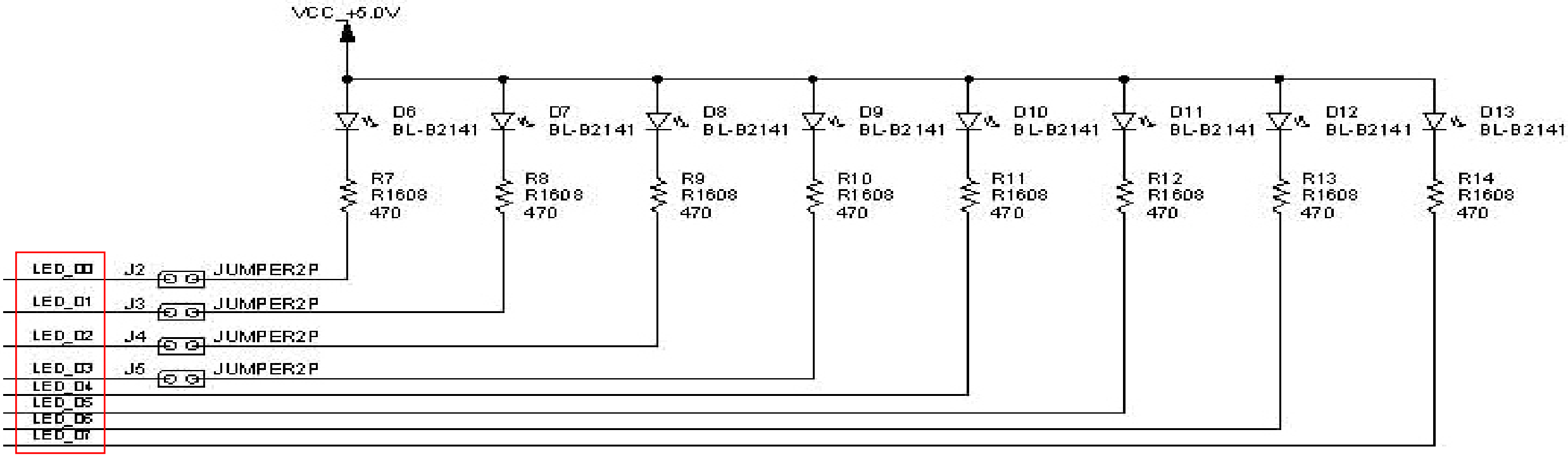
This one?

Display code

```
void Init_GPIO(void)
{ GPIO_InitTypeDef GPIO_InitStructure;
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA |
RCC_AHBPeriph_GPIOC, ENABLE);
GPIO_InitStructure.GPIO_Pin =GPIO_Pin_0 |
GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 ;
GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_10MHz;
GPIO_InitStructure.GPIO_Mode =
GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 |
GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3 ;
GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_10MHz;
GPIO_InitStructure.GPIO_Mode =
GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
GPIO_Init(GPIOA, &GPIO_InitStructure);}
```

Initial code

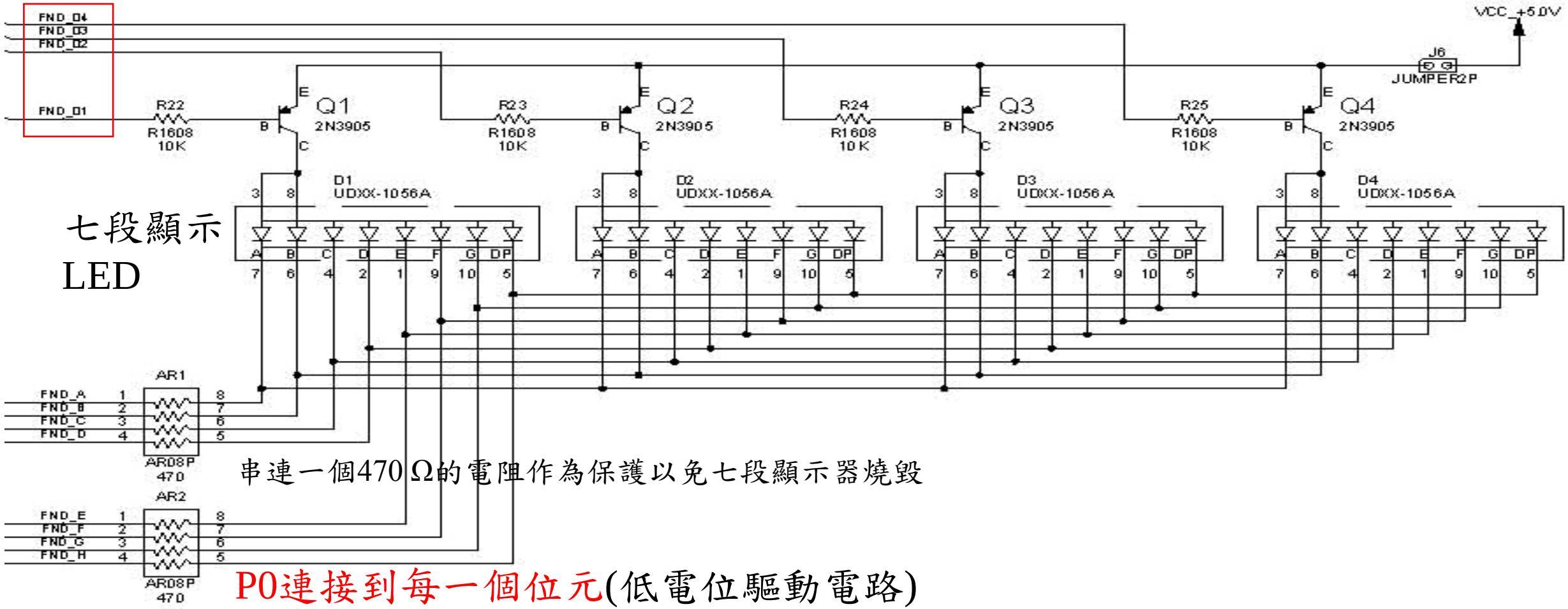
Single 7-segment 8-digit-LED Output Circuit
單一七段顯示器驅動線路



P1輸出連接到每一個位元(低電位驅動電路)

Multiple-7 Segment多個七段顯示器驅動

P4連接到每一個數字(review較大負載驅動電路)



七段顯示LED

串連一個470Ω的電阻作為保護以免七段顯示器燒毀

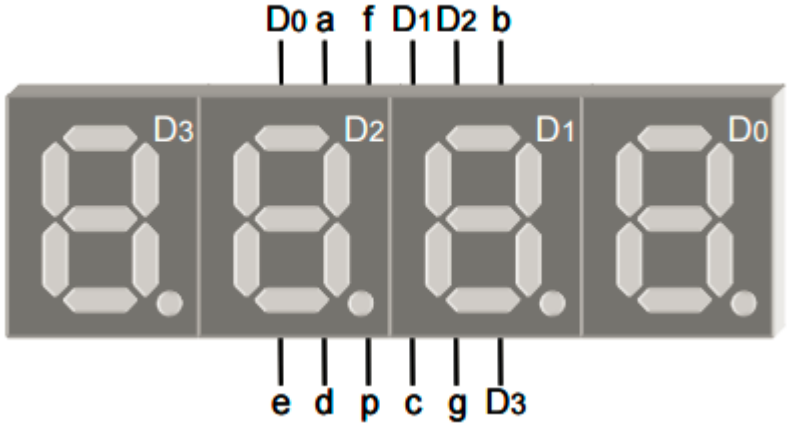
P0連接到每一個位元(低電位驅動電路)

有沒有什麼方法可進一步減少控制7-segment的pin腳?

From Parallel Data to Serial Data

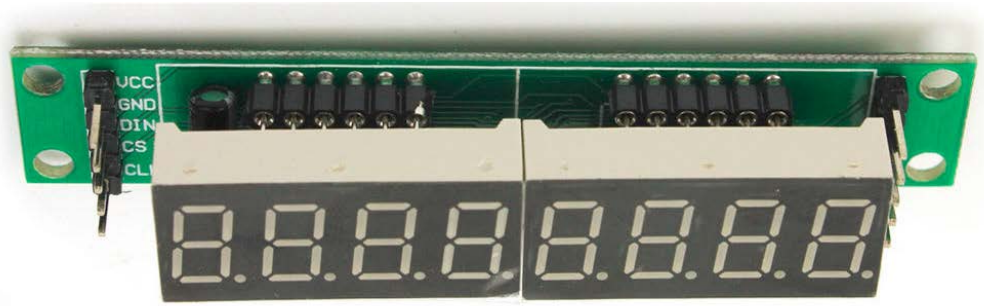
- If we connect stm32 I/O pin on 7-Seg LED directly, we use eight 7-Seg LED →We will need 16 GPIO pin!
- We have to scan eight 7-Seg LED to show different number on it

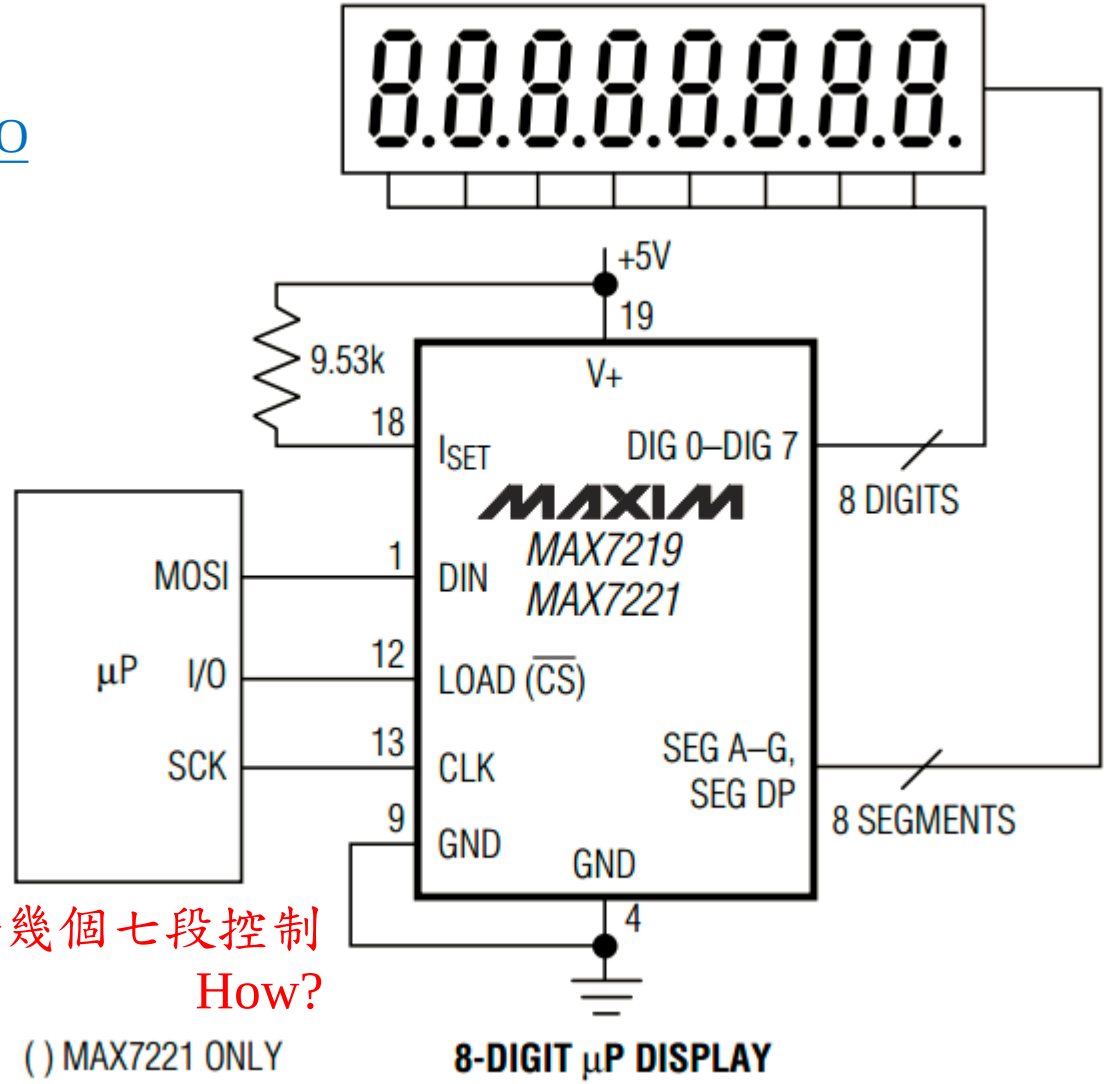
→ We use Max7219 to simplify our work



Max7219 Display Driving IC

•https://www.sparkfun.com/datasheets/Components/General/COM-09622-MAX7219-MAX7221.pdf





只要三根pin就可以做好幾個七段控制

How?

時脈(Clock Rate)

- **時脈**（又譯：**時鐘頻率**，英語：clock rate）是指同步電路中時鐘的基礎頻率，它以“每秒時鐘週期”（clock cycles per second）來度量，量度單位採用SI單位赫茲（Hz）。例如，來自晶振的基準頻率通常等於一個固定的正弦波形，則時鐘頻率就是這個基準頻率，電子電路會為數位電子設備將它轉化成對應的脈衝方波。
- 在單個時鐘週期內（現代非嵌入式微處理器的這個時間一般都短於1 nS）邏輯零狀態與邏輯一狀態來回切換。 由於發熱和電氣規格的限制，週期裡邏輯零狀態的持續時間歷來要長於邏輯一狀態。
- 中央處理器（CPU）製造商常為時脈較高的CPU定額外的高價。

https://zh.wikipedia.org/wiki/%E6%97%B6%E9%92%9F%E9%A2%91%E7%8E%87

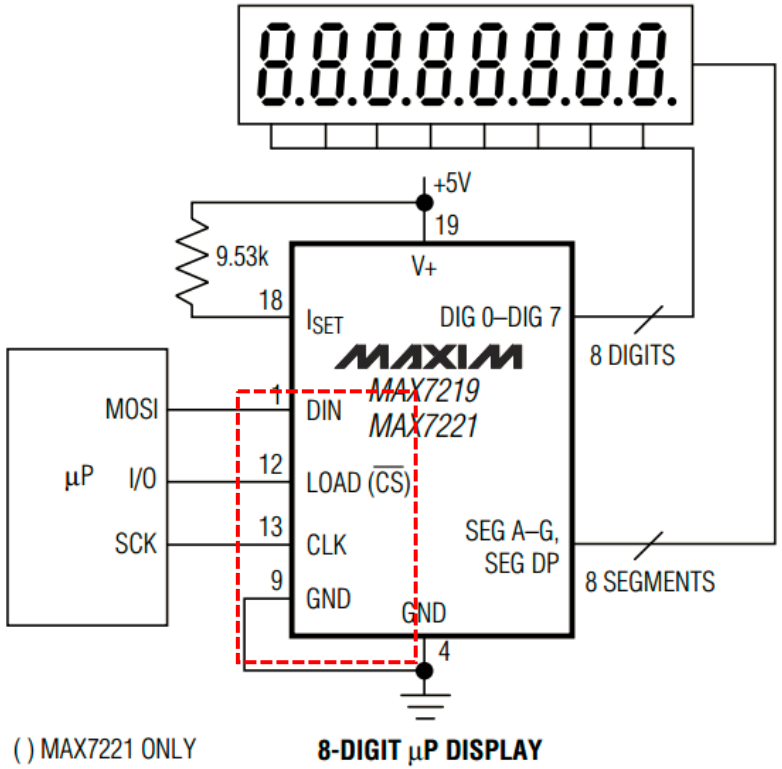
Max7219 – DIN, CS, CLK

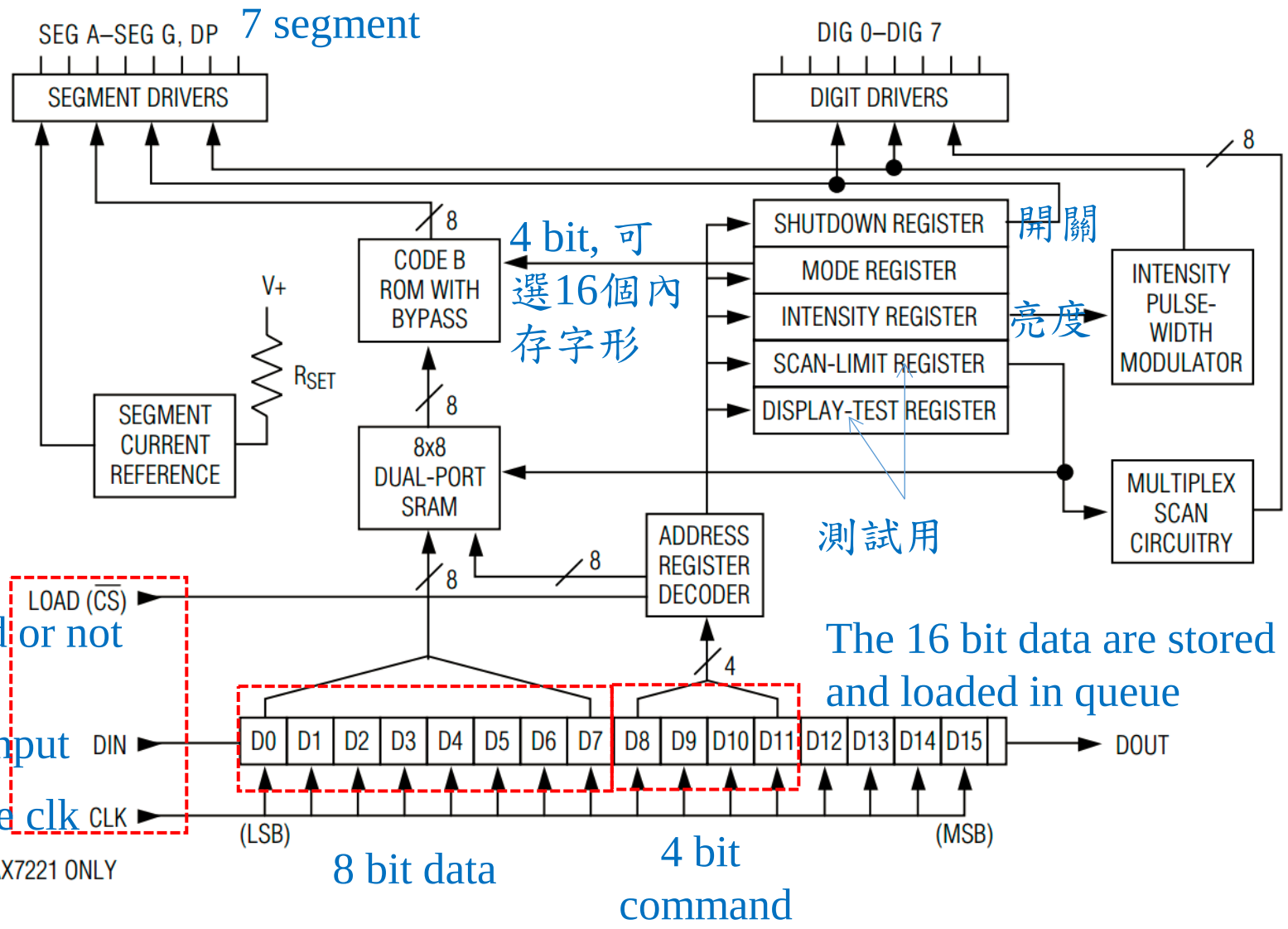
- DIN: Serial-Data Input. **Data is loaded into the internal 16-bit shift register on CLK's rising edge.**
- CS: Load-Data Input. The last 16 bits of serial data are latched on LOAD(CS)'s rising edge.
- CLK: Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register.

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Max7219


+5V
19
9.53k
18
V+
ISET
DIG 0–DIG 7
MAXIM
MAX7219
MAX7221
8 DIGITS
MOSI
µP
I/O
SCK
1
DIN
12
LOAD (CS)
13
CLK
9
GND
SEG A–G,
SEG DP
8 SEGMENTS
GND
4
( ) MAX7221 ONLY
8-DIGIT µP DISPLAY


7 segment
SEG A–SEG G, DP
SEGMENT DRIVERS
DIG 0–DIG 7
DIGIT DRIVERS
8
CODE B
ROM WITH
BYPASS
4 bit, 可
選16個內
存字形
SHUTDOWN REGISTER
開關
MODE REGISTER
INTENSITY REGISTER
亮度
SCAN-LIMIT REGISTER
DISPLAY-TEST REGISTER
INTENSITY
PULSE-
WIDTH
MODULATOR
V+
RSET
SEGMENT
CURRENT
REFERENCE
8x8
DUAL-PORT
SRAM
ADDRESS
REGISTER
DECODER
MULTIPLEX
SCAN
CIRCUITRY
測試用
LOAD (CS)
load or not
The 16 bit data are stored
and loaded in queue
Data input
DIN
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
DOUT
Data sample clk
CLK
(LSB)
(MSB)
( ) MAX7221 ONLY
8 bit data
4 bit
command

Max7219 – Register Address Map

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

選擇讓八個7seg中的哪一個7seg digit亮
(decode/no decode mode需先設定好)

在7seg開始使用前需要先把以下設定好
設定哪幾個7seg digit是decode哪幾個no decode
設定LED亮度
設定要亮幾個digit的LED
設定要正常運作還是關上
測試用

印出供同學參考

7seg.h

```
// Define a lot of Non-Decode Mode Constants
#define SEG_DATA_NON_DECODE_0          0b1111110
#define SEG_DATA_NON_DECODE_1          0b0110000
…

// Define ADDRESS Constants for 7seg commands
#define SEG_ADDRESS_NOP                0
#define SEG_ADDRESS_DIGIT_0            1
#define SEG_ADDRESS_DIGIT_1            2
#define SEG_ADDRESS_DIGIT_2            3
#define SEG_ADDRESS_DIGIT_3            4
#define SEG_ADDRESS_DIGIT_4            5
#define SEG_ADDRESS_DIGIT_5            6
#define SEG_ADDRESS_DIGIT_6            7
#define SEG_ADDRESS_DIGIT_7            8
#define SEG_ADDRESS_DECODE_MODE        9
#define SEG_ADDRESS_ITENSITY           10
#define SEG_ADDRESS_SCAN_LIMIT         11
#define SEG_ADDRESS_SHUTDOWN           12
#define SEG_ADDRESS_DISPLAY_TEST       15
```

```
// Define DATA Constants for 7seg commands
#define SEG_DATA_DECODE_0              0
#define SEG_DATA_DECODE_1              1
#define SEG_DATA_DECODE_2              2
#define SEG_DATA_DECODE_3              3
#define SEG_DATA_DECODE_4              4
#define SEG_DATA_DECODE_5              5
#define SEG_DATA_DECODE_6              6
#define SEG_DATA_DECODE_7              7
#define SEG_DATA_DECODE_8              8
#define SEG_DATA_DECODE_9              9
#define SEG_DATA_DECODE_DASH           10
#define SEG_DATA_DECODE_E              11
#define SEG_DATA_DECODE_H              12
#define SEG_DATA_DECODE_L              13
#define SEG_DATA_DECODE_P              14
#define SEG_DATA_DECODE_BLANK          15
```

印出供同學參考

Max7219—Registers and Their Functions

- 被設定為Decode Mode的7 seg會透過解碼器將D0~D3的值解碼成0~9,-,E,H,L,P,(空白)後顯示在7seg上(請參考Table 5的對應)。
- 被設定為No Decode Mode的7 seg則會將D0~D7直接顯示在7-Seg LED 的對應pin腳上(請參考Table6)。
- Intensity: 用來設定7-Seg LED的亮度，0到15越來越亮。
- Scan Limit: 用來設定7-Seg LED的顯示位數0表示顯示一位，1表示顯示兩位，以此類推。
- Shutdown: 設為normal operation時LED會運作，設為shutdown mode時7-Seg LED會關掉，是一種省電模式。
- Display Test:測試用。會讓所有LED都亮起來。

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Max7219—Decode-Mode Register

Step1: 先送一組16 bit DIN設定哪幾個7seg是decode，哪幾個是no decode。
D8~D11 = 1001。D0~D7對應7seg的digit(見Table4)

Step2: 再依顯示需求送16 bit DIN決定要亮哪個7seg digit (D8~D11)，顯示什麼(D0~D7)。
此時D0~D7要依據Step1對該digit的設定，來決定送decode還是nodecode的data

Address:D8~D11 = 0b1001

Table 4. Decode-Mode Register Examples (Address (Hex) = 0xX9)

DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	0x00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	0x01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0x0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	0xFF

可以選擇八個7seg中，哪幾個要用
decode做，哪幾個要用no decode做

Address: D8~D11

0b0001 ~0b1000

Max7219—Decode-Mode Register

- When the code B decode mode is used
  - the decoder looks only at the lower nibble of the data in the digit registers (D3–D0), disregarding bits D4–D6.
  - D7, which sets the decimal point (SEG DP), is independent of the decoder and is positive logic (D7 = 1 turns the decimal point on).

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

- D8~D11代表要亮哪一個digit的7seg
- Decode mode只需要給D0~D3值，MAX自己會去找對應數字，然後選定7seg中哪幾個led要亮。
- No decode mode則是直接D0~D7和7seg中led去對應。

Table 5. Code B Font

7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

*The decimal point is set by bit D7 = 1

Address: D8~D11
0b0001
~0b1000

Max7219— No Decode-Mode

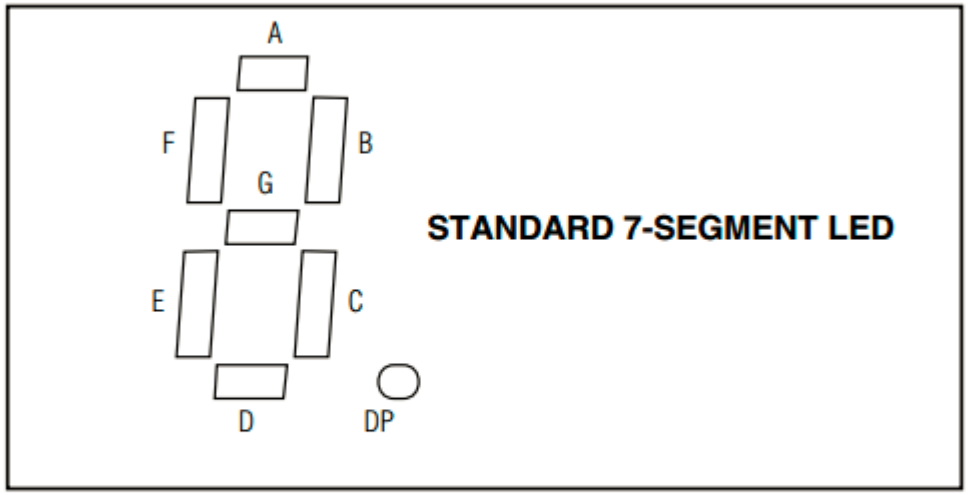
- When no-decode is selected, data bits D7– D0 correspond to the segment lines of the MAX7219/MAX7221

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

在No decode mode設定下
D8~D11代表要亮哪一個7seg
D0~D7 則代表要亮該7seg的哪幾個LED

Table 6. No-Decode Mode Data Bits and Corresponding Segment Lines



	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Corresponding Segment Line	DP	A	B	C	D	E	F	G

No Decode，直接做對應

Max7219—Shutdown Register

- When the MAX7219 is in shutdown mode, the scan oscillator is halted, all segment current sources are pulled to ground, and all digit drivers are pulled to V+(共陽極下就會關上), thereby blanking the display. Data in the digit and control registers remains unaltered.
- Normal operation wakes up the 7seg.

Address: 0b1100

Table 3. Shutdown Register Format (Address (Hex) = 0xXC)

MODE	ADDRESS CODE (HEX)	REGISTER DATA							
		D7	D6	D5	D4	D3	D2	D1	D0
Shutdown Mode	0xXC	X	X	X	X	X	X	X	0
Normal Operation	0xXC	X	X	X	X	X	X	X	1

設0關上，設1正常運作

Max7219—Intensity Register

Address: 0b1010

Table 7. Intensity Register Format (Address (Hex) = 0xXA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	0xX0
3/32	2/16	X	X	X	X	0	0	0	1	0xX1
5/32	3/16	X	X	X	X	0	0	1	0	0xX2
7/32	4/16	X	X	X	X	0	0	1	1	0xX3
9/32	5/16	X	X	X	X	0	1	0	0	0xX4
11/32	6/16	X	X	X	X	0	1	0	1	0xX5
13/32	7/16	X	X	X	X	0	1	1	0	0xX6
15/32	8/16	X	X	X	X	0	1	1	1	0xX7
17/32	9/16	X	X	X	X	1	0	0	0	0xX8
19/32	10/16	X	X	X	X	1	0	0	1	0xX9
21/32	11/16	X	X	X	X	1	0	1	0	0xXA
23/32	12/16	X	X	X	X	1	0	1	1	0xXB
25/32	13/16	X	X	X	X	1	1	0	0	0xXC
27/32	14/16	X	X	X	X	1	1	0	1	0xXD
29/32	15/16	X	X	X	X	1	1	1	0	0xXE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	0xXF

Max7219—Scan-Limit Register

- The scan-limit register sets how many digits are displayed, from 1 to 8. The number of scanned digits affects the display brightness,

Address: 0b1011

Table 8. Scan-Limit Register Format (Address (Hex) = 0xXB)

SCAN LIMIT	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
Display digit 0 only*	X	X	X	X	X	0	0	0	0xX0
Display digits 0 & 1*	X	X	X	X	X	0	0	1	0xX1
Display digits 0 1 2*	X	X	X	X	X	0	1	0	0xX2
Display digits 0 1 2 3	X	X	X	X	X	0	1	1	0xX3
Display digits 0 1 2 3 4	X	X	X	X	X	1	0	0	0xX4
Display digits 0 1 2 3 4 5	X	X	X	X	X	1	0	1	0xX5
Display digits 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	0xX6
Display digits 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	0xX7

*See *Scan-Limit Register* section for application.

選擇要亮哪幾個七段

Max7219—Display Test Register

- The display-test register operates in two modes: normal and display test. Display-test mode turns all LEDs on by overriding, but not altering, all controls and digit registers (including the shutdown register).

Address: 0b1111

Table 10. Display-Test Register Format (Address (Hex) = 0xXF)

MODE	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Normal Operation	X	X	X	X	X	X	X	0
Display Test Mode	X	X	X	X	X	X	X	1

Note: The MAX7219/MAX7221 remain in display-test mode (all LEDs on) until the display-test register is reconfigured for normal operation.

Goto
000-5-MCSL-ARMGPIO7Segment ycc
P2~18 for the figures of signal

範例指令-練習

	D12~D15	D8~D11 (Command/address)	D0~D7 (Data)
Set 0-3 decode, 4-7 no-decode	0xX	0b1001 (9)	0b0000 1111
Set all no decode mode			
Scan limit digit 0 only			
Shot down			
Wake up			
Display 0 on nodecode digit 0			
Display 0 on decode digit 1			

範例指令-解答

	D12~D15	D8~D11 (Command/address)	D0~D7 (Data)
Set 0-3 decode, 4-7 no-decode	0xX	0b1001 (9)	0b0000 1111
Set all no decode mode	0xX	0b1001 (9)	0x00
Scan limit digit 0 only	0xX	0b1011(11)	0xX0
Shot down	0xX	0b1100(12)	0xX0
Wake up	0xX	0b1100(12)	0xX1
Display 0 on nodecode digit 0	0xX	0b0001(1)	0b0111110
Display 0 on decode digit 1	0xX	0b0010(2)	0b0000 (or 0)

Max7219 Initialization and Operation Steps(重要)

Step1: 關上Display test mode (如果有被打開)

- Initialization需至少設定以下控制

  Step2: 設定哪幾個是decode，哪幾個是non decode

  Step3: 設定亮哪幾個7seg digit

  Step4: 把Max7219控制7seq digit的電壓打開，設定要顯示的digit

```
// Set Decode Mode to non-decode mode
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0x00);
// Set Scan Limit to digit 0 only
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0x00);
// Wakeup 7seg
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);
```

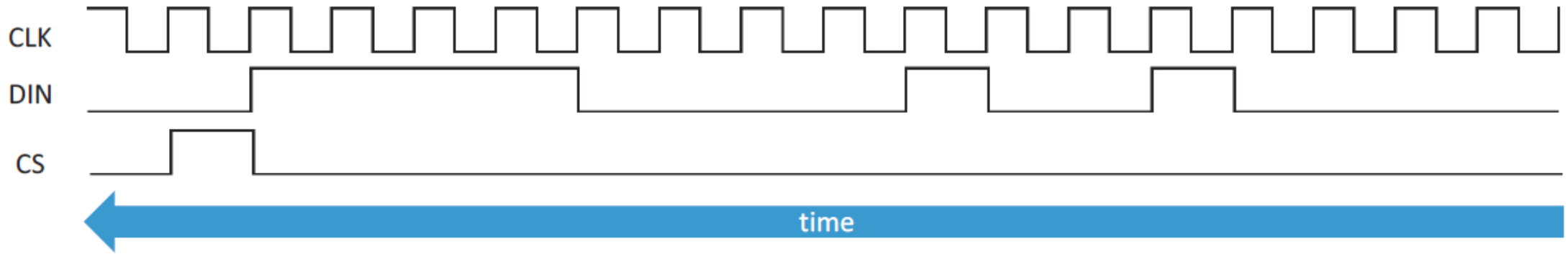
- Step5: 開始送顯示資料

```
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_0, SEG_DATA_NON_DECODE_LOOP[current]);
```

至pdf講解訊號序列

I want to set decode mode(Code B decode for digit 0-3, no decode for digits 4-7), thus I have to set Serial-Data as below!
And then send a rising edge on CS pin to latch the Serial-Data!

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				DATA							
X	X	X	X	1	0	0	1	0	0	0	0	1	1	1	1



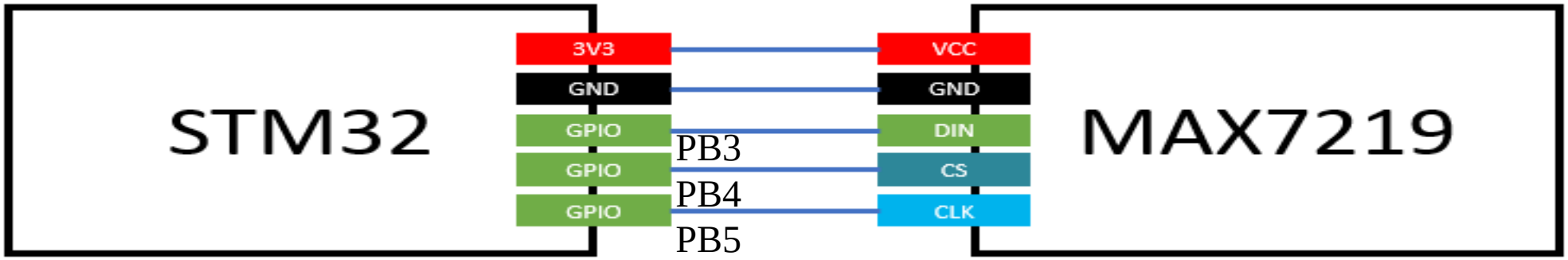
注意這裡的時脈是程式給的，不是外接的石英oscillator。又因為CLK是rising edge觸發。
所以執行
CLK=0;
然後DIN=X; CS=X;
再執行CLK=1;
讓DIN和CS的值出現後，再把CLK由0變成1，出現CLK rising edge觸發，
即可完成一次值的設定或讀取

Lab3 7-Segment
範例程式

Code provided by NCTU CS 謝明恩 吳赫倫
Slide made by NCTU ME 助教 林穎毅
20170326

Lab 3.1 Max7219與7-Seg LED練習—without code-B decode mode

- 將stm32的3.3V接到7-Seg LED板的VCC，GND接到GND，並選擇三個GPIO接腳分別接到DIN、CS和CLK。



- 完成以下程式碼，並利用GPIO控制Max7219並在7-Seg LED上的第一位依序顯示0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F (時間間隔1秒)，範例影片如下： https://goo.gl/ZDZcdl

Assembly

```
arr: .byte 0x7e, 0x30,
0x6d, …

main:
    BL  GPIO_init
    BL  max7219_init

    ldr r9, =arr
    ldr r2, =#0
.for_loop:
    mov r0, #1
    ldrb r1, [r9, r2]
    BL MAX7219Send
    BL Delay
    add r2, r2, #1
    cmp r2, #16
    bne .for_loop
    mov r2, #0
    b .for_loop
```

```
.equ DECODE_MODE,                    0x09

max7219_init:
       push {r0, r1, r2, lr}
       ldr r0, =#DECODE_MODE
       ldr r1, =#0x0
       BL MAX7219Send
       ldr r0, =#DISPLAY_TEST
       ldr r1, =#0x0
       BL MAX7219Send
       ldr r0, =#SCAN_LIMIT
       ldr r1, =0x0
       BL MAX7219Send
       ldr r0, =#INTENSITY
       ldr r1, =#0xA
       BL MAX7219Send
       ldr r0, =#SHUTDOWN
       ldr r1, =#0x1
       BL MAX7219Send
       pop {r0, r1, r2, pc}
```

Assembly

```
.equ DATA,                                  0x20 //PA5
.equ LOAD,                                  0x40 //PA6
.equ CLOCK,                                 0x80 //PA7

MAX7219Send://input parameter: r0 is address , r1 is data
...
lsl r0, r0, #8
add r0, r0, r1
ldr r1, =#GPIOA_BASE
ldr r2, =#LOAD
ldr r3, =#DATA
ldr r4, =#CLOCK
ldr r5, =#GPIO_BSRR_OFFSET
ldr r6, =#GPIO_BRR_OFFSET
mov r7, #16//r7 = i
```

```
.max7219send_loop:
      mov r8, #1
      sub r9, r7, #1
      lsl r8, r8, r9 // r8 = mask
      str r4, [r1,r6]//HAL_GPIO_WritePin(GPIOA, CLOCK, 0);
      tst r0, r8
      beq .bit_not_set//bit not set
      str r3, [r1,r5]
      b .if_done
.bit_not_set:
      str r3, [r1,r6]
.if_done:
      str r4, [r1,r5]
      subs r7, r7, #1
      bgt .max7219send_loop
      str r2, [r1,r6]
      str r2, [r1,r5]
...
```

程式語言撰寫架構

- main.c裡面
  - PIN腳設定
  - main function

- 7seg.c裡面
  - init_7seg
  - send_7seg

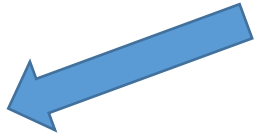
程式語言撰寫架構

- main.c裡面
  - PIN腳設定
  - main function

- 7seg.c裡面
  - init_7seg
  - send_7seg

init_7seg function

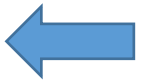
```
int init_7seg(GPIO_TypeDef* gpio, int DIN, int CS, int CLK){
    // Enable AHB2 Clock
    if(gpio==GPIOA){
        RCC->AHB2ENR |= RCC_AHB2ENR_GPIOAEN;
    }
    else if(gpio==GPIOB){
        RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;
    }
    else{
        // Error! Add other cases to suit other GPIO pins
        return -1;
    }
```



Enable clock

init_7seg function (code接續上一張slide)

```
    // Set GPIO pins to output mode (01)
    // First Clear bits(&) then set bits(|)
    gpio->MODER &= ~(0b11 << (2*DIN));
    gpio->MODER |= (0b01 << (2*DIN));
    gpio->MODER &= ~(0b11 << (2*CS));
    gpio->MODER |= (0b01 << (2*CS));
    gpio->MODER &= ~(0b11 << (2*CLK));
    gpio->MODER |= (0b01 << (2*CLK));

    // Close display test
    send_7seg(gpio, DIN, CS, CLK, SEG_ADDRESS_DISPLAY_TEST, 0x00);

    return 0;
}
```



設定Moder register, 把指定的Pin腳設為output

見下頁

Display test全部設0 (關上)

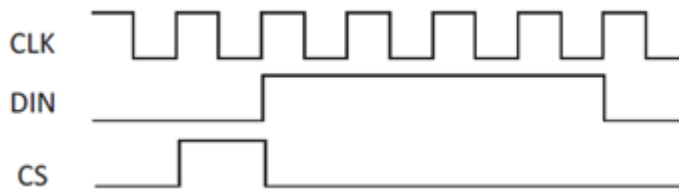
send_7seg function 重要，搭配輸出波形圖詳細說明

```
void send_7seg(GPIO_TypeDef* gpio, int DIN, int CS, int CLK, int address, int data){
    // The payload to send
    int payload = ((address&0xFF)<<8)|(data&0xFF);

    // Start the sending cycles
    // 16 data-bits + 1 CS signal
    int total_cycles = 16+1;

    for(int a=1;a<=total_cycles;a++){
        // Reset CLK when enter
        reset_gpio(gpio, CLK);

        // Set DIN according to data except for last cycle(CS)
        if(((payload>>(total_cycles-1-a))&0x1) && a!=total_cycles){
            set_gpio(gpio, DIN);
        }
        else{
            reset_gpio(gpio, DIN);
        }
```

把command存在前8個bit (D8~D15)，資料存在後8個bit(D0~D7)

把前面兩個hexadecimal清零，保留後兩個

前16碼為輸入資料(XXXX+4bit address+8 bit data)，最後一碼為告訴顯示器要接收這筆資料的訊號。



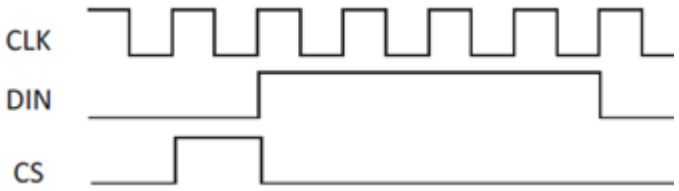
CLK 設為0

留意送入MAX7219的CLK是我們用程式製造出來的

Payload要輸出1且16bit還沒完

一個值一個值的讀取payload的資料，然後輸出到DIN pin腳上

send_7seg function (code接續上一張slide)




```
        // Set CS at last cycle
        if(a==total_cycles){
            set_gpio(gpio, CS);
        }
        else{
            reset_gpio(gpio, CS);
        }

        // Set CLK when leaving (7seg set data at rising edge)
        set_gpio(gpio, CLK);
    }

    return;
}
```

CS 在rising edge 時，開始讀入之前存在register D0~D15的資料們，在這裡設為1，就是產生rising edge

CLK 設為1, 此時CLK的pin生成一rising edge，MAX7219讀入DIN上設定好的值

Set/reset function

```
6  void set_gpio(GPIO_TypeDef* gpio, int pin){
7      gpio->BSRR |= (1 << pin);
8  }
9  void reset_gpio(GPIO_TypeDef* gpio, int pin){
10     gpio->BRR |= (1 << pin);
11 }
```

直接把GPIOB 的某一pin值改為1，和ODR不同

直接把GPIOB 的某一pin值改為0，和ODR不同

程式語言撰寫架構

- main.c裡面
  - PIN腳設定
  - main function

- 7seg.c裡面
  - init_7seg
  - send_7seg

基本設定跟PIN角設定

```
1 #include "stm32l476xx.h"
2 #include "helper_functions.h"
3 #include "led_button.h"
4 #include "7seg.h"
// Define pins for 7seg
#define SEG_gpio GPIOB
#define DIN_pin 3
#define CS_pin 4
#define CLK_pin 5
```

main function

```
int main(){

    if(init_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin) != 0){
        // Fail to init 7seg
        return -1;
    }
```

初始化GPIO

SEG_ADDRESS_XXX的定義寫在7seg.h裡面

修改程式碼帶著看

```
// Set Decode Mode to non-decode mode
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0x00);
// Set Scan Limit to digit 0 only
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SCAN_LIMIT, 0x00);
// Wakeup 7seg
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_SHUTDOWN, 0x01);
```

0x00: 把每個7seg都設定成NO decode mode

只打開第0個的7segment (0x00)

把7segment 設定為normal operation (0x01)

main function (code接續上一張slide)

```
int SEG_DATA_NON_DECODE_LOOP[17] = {
    SEG_DATA_NON_DECODE_0,
    SEG_DATA_NON_DECODE_1,
    SEG_DATA_NON_DECODE_2,
    SEG_DATA_NON_DECODE_3,
    SEG_DATA_NON_DECODE_4,
    SEG_DATA_NON_DECODE_5,
    SEG_DATA_NON_DECODE_6,
    SEG_DATA_NON_DECODE_7,
    SEG_DATA_NON_DECODE_8,
    SEG_DATA_NON_DECODE_9,
    SEG_DATA_NON_DECODE_0,
    SEG_DATA_NON_DECODE_A,
    SEG_DATA_NON_DECODE_B,
    SEG_DATA_NON_DECODE_C,
    SEG_DATA_NON_DECODE_D,
    SEG_DATA_NON_DECODE_E,
    SEG_DATA_NON_DECODE_F
};
```

定義在7seg.h內

led_button.c | 7seg.h | helper_functions.c | startu

```
4  #include "stm32l476xx.h"
5
6  // Define a lot of Non-Decode Mode Constants
7  #define SEG_DATA_NON_DECODE_0       0b1111110
8  #define SEG_DATA_NON_DECODE_1       0b0110000
9  #define SEG_DATA_NON_DECODE_2       0b1101101
10 #define SEG_DATA_NON_DECODE_3       0b1111001
11 #define SEG_DATA_NON_DECODE_4       0b0110011
12 #define SEG_DATA_NON_DECODE_5       0b1011011
13 #define SEG_DATA_NON_DECODE_6       0b1011111
14 #define SEG_DATA_NON_DECODE_7       0b1110000
15 #define SEG_DATA_NON_DECODE_8       0b1111111
16 #define SEG_DATA_NON_DECODE_9       0b1111011
17 #define SEG_DATA_NON_DECODE_A       0b1110111
18 #define SEG_DATA_NON_DECODE_B       0b0011111
19 #define SEG_DATA_NON_DECODE_C       0b1001110
20 #define SEG_DATA_NON_DECODE_D       0b0111101
21 #define SEG_DATA_NON_DECODE_E       0b1001111
```

main function (code接續上一張slide)

```
int current=0;
while(1){
    // Write to digit 0
    send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_0, SEG_DATA_NON_DECODE_LOOP[current]);
    current = (current+1)%17;
    delay_without_interrupt(1000);
}


    return 0;
}
```

SEG_ADDRESS_XXX的定義寫在7seg.h裡面

No decode:亮第0個7seg　　第0個7seg要亮哪幾個LED

SEG_DATA_NON_DECODE_LOOP 內有0~H共17個數

led_button.c | 7seg.h | helper_functions.c | start

```
26 // Define ADDRESS Constants for 7seg commands
27 #define SEG_ADDRESS_NOP             0
28 #define SEG_ADDRESS_DIGIT_0         1
29 #define SEG_ADDRESS_DIGIT_1         2
30 #define SEG_ADDRESS_DIGIT_2         3
31 #define SEG_ADDRESS_DIGIT_3         4
32 #define SEG_ADDRESS_DIGIT_4         5
33 #define SEG_ADDRESS_DIGIT_5         6
34 #define SEG_ADDRESS_DIGIT_6         7
35 #define SEG_ADDRESS_DIGIT_7         8
36 #define SEG_ADDRESS_DECODE_MODE     9
37 #define SEG_ADDRESS_ITENSITY        10
38 #define SEG_ADDRESS_SCAN_LIMIT      11
39 #define SEG_ADDRESS_SHUTDOWN        12
40 #define SEG_ADDRESS_DISPLAY_TEST    15
```

Lab 3.2 Max7219與7-Seg LED練習—use code-B decode mode

- 利用GPIO控制Max7219並在7-Seg LED上顯示自己的學號，例如學號為1234567則顯示下圖,請使用decode mode ：
- 完成以下程式碼，將放在student_id array 裡的學號顯示到7-seg LED上。

```
main:
    BL  GPIO_init
    BL  max7219_init


        ldr r9, =arr
        ldr r2, =#0
        ldr r3, =#8
        ldr r4, =#9
.for_loop:
        ldrb r1, [r9, r2]
        add r0, r2, #1
        sub r0, r4, r0
        BL MAX7219Send
        add r2, r2, #1
        cmp r2, #8
        bne .for_loop
loop:
        b loop
```

Part of main function

- 請記得先設定decode mode

```
// Set Decode Mode to Code B decode mode
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DECODE_MODE, 0xFF);
```

0xFF: 把每個7seg都設定成decode mode

```
// student id starts with zero but don't write zero, otherwise it will consider it as 0x
int student_id = 123456;
// Write to digits
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_0, student_id%10);
student_id = student_id/10;
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_1, student_id%10);
student_id = student_id/10;
send_7seg(SEG_gpio, DIN_pin, CS_pin, CLK_pin, SEG_ADDRESS_DIGIT_2, student_id%10);
```

decode:亮第0個7seg　　第0個7seg要的數字代號

decode:亮第1個7seg　　第1個7seg要的數字代號

display_number function -- an alternative function for 學號 display

```
int display_number(GPIO_TypeDef* gpio, int DIN, int CS, int CLK, int num, int num_digs){
    for(int i=1;i<=num_digs;i++){ 顯示要顯示的數字序列
        send_7seg(gpio, DIN, CS, CLK, i, num % 10);
        num /= 10;                    address  data
    }
    for(int i=num_digs+1;i<=8;i++){剩餘的七段關上，15 在decode中表示關上七段
        num /= 10;
        send_7seg(gpio, DIN, CS, CLK, i, 15);
    }
    if(num != 0)
        return -1;
    return 0;
}
```

Lab 3.2 輔助說明

- LAB 3.2的學號若是0開頭的同學宣告時請不要把開頭的0寫進去(會被當成8進位)
  EX:
  學號為0123456的同學請宣告如下
  student_id: .word 123456
  但7-SEG LED上請將學號的0顯示出來!

Lab 3.3 (加分)顯示Fibonacci數

- 請設計一組語程式偵測實驗板上的User button，當User button按N次時7-Seg LED上會顯示fib(N)的值。
- fib(0) = 0、fib(1) = 1、fib(2) = 1 、…
- 若fib(N) ≥ 100000000則顯示-1。

浮點數運算問題

- 有同學遇到在做小數點除法運算時會直接跳到Infinite_Loop問題，遇到相同問題的同學，請參考E3參考資料STM32L4 Programming manual，瞭解Coprocessor access control register (CPACR)的功能，並參考Enabling the FPU(page 256)將FPU權限打開。