

Upboard Lab02

- Upboard Lab02
- A. Introduction to this Lab Practicing Unit
 - I. Task
 - II. Overview
 - III. Hardware
 - IV. IC
 - 1. 擴充板背面
 - 2. 74HC165長相
 - 3. 74HC595長相
 - 4. MAX7219長相
 - 5. MCP3201長相
 - V. 實驗目的
 - 問題
 - 解決方式
 - VI. 74HC165
 - 1. 腳位說明
 - 2. Operating mode
 - 3. UpBoard pin diagram
 - VII. 74HC595
 - 1. 腳位說明
 - 2. Operating mode
 - 3. UpBoard pin diagram
 - VIII. MAX7219
 - 1. 腳位說明
 - 2. Data Format
 - 3. UpBoard pin diagram
 - XI. ADC
 - 實驗目的
 - XII. MCP3201
 - 1. 腳位說明
 - 2. SPI communication
 - 3. 比例轉換公式
 - 4. UpBoard pin diagram
 - XIII. PWM

- 1. 實驗目的
- 2. 運作原理
- 3. PWM in mraa
- B. Demonstration
 - I. 按鍵輸入偵測
 - II. 基礎跑馬燈應用：多顆LED輸出
 - III. 基礎點矩陣應用：矩陣跑馬燈
 - IV ADC
 - SPI 傳送資料
 - PWM

A. Introduction to this Lab Practicing Unit

I. Task

1. 透過控制 74HC165、74HC595、MAX7219 IC 了解控制周邊元件的過程，並了解序列與平行訊號的原理與轉換。
2. 透過控制 MCP3201 IC 了解類比數位轉換器的使用。
3. 透過控制UpBoard上的 PWM 針腳，了解 PWM 針腳如何應用。

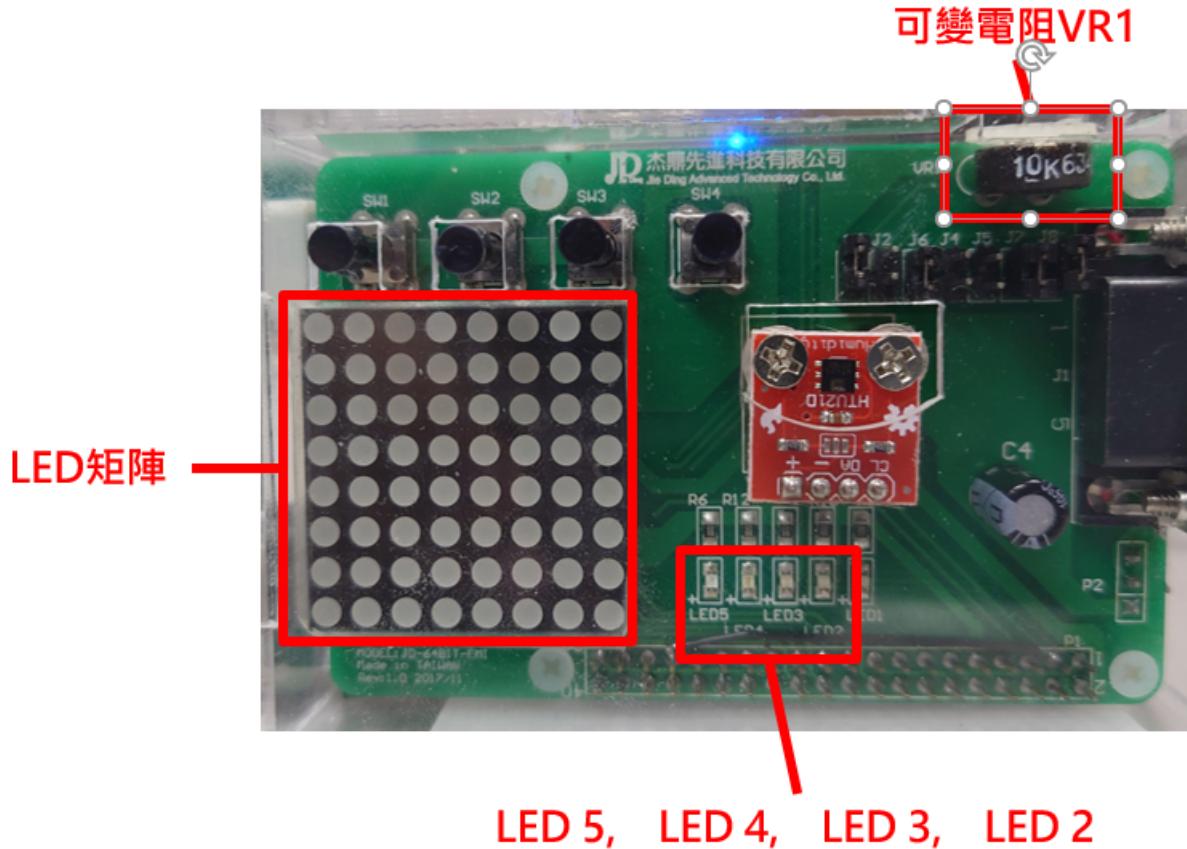
II. Overview

1. IC
 - 74HC165 (平行訊號 → 序列訊號) 可以搭配Lab01
 - 74HC595 (序列訊號 → 平行訊號)
 - MAX7219 (LED矩陣控制)
2. ADC
 - MCP3201 (類比數位轉換器)
3. PWM (脈波寬度調變)

Device (ubilinux)	Function	Linux GPIO	UP pinout	Pin ▼	Pin	UP pinout	Linux GPIO	Function	Device (ubilinux)
			3.3V	1	2	5V			
i2c-5	I2C1_SDA	2	GPIO0	3	4	5V			
i2c-5	I2C1_SCL	3	GPIO1	5	6	Ground			
lio:device0	ADC0	4	GPIO2	7	8	GPIO15	14	UART1_TX	ttyS4
			Ground	9	10	GPIO16	15	UART1_RX	ttyS4
ttyS4	UART1_RTS	17	GPIO3	11	12	GPIO17	18	PCM_CLK	
		27	GPIO4	13	14	Ground			
		22	GPIO5	15	16	GPIO18	23		
			3.3V	17	18	GPIO19	24		
	SPI_MOSI	10	GPIO6	19	20	Ground			
	SPI_MISO	9	GPIO7	21	22	GPIO20	25		
	SPI_CLK	11	GPIO8	23	24	GPIO21	8	SPI_CS0	spidev2.0
			Ground	25	26	GPIO22	7	SPI_CS1	spidev2.1
i2c-0	ID_SD	0	GPIO9	27	28	GPIO23	1	ID_SC	i2c-0
		5	GPIO10	29	30	Ground			
		6	GPIO11	31	32	GPIO24	12	PWM0	pwmchip0/pwm0
pwmchip1/pwm0	PWM1	13	GPIO12	33	34	Ground			
	PCM_FS	19	GPIO13	35	36	GPIO25	16	UART1_CTS	ttyS4
		26	GPIO14	37	38	GPIO26	20	PCM_DIN	
			Ground	39	40	GPIO27	21	PCM_DOUT	

III. Hardware

本次實驗會使用到的硬體



IV. IC

積體電路 (Integrated Circuit)，或稱晶片 (chip)

將電路元件集中製造於半導體晶圓的表面，並經由封裝處理後，透過其保留之對外針腳輸出 / 入電磁訊號，能實現特定應用。

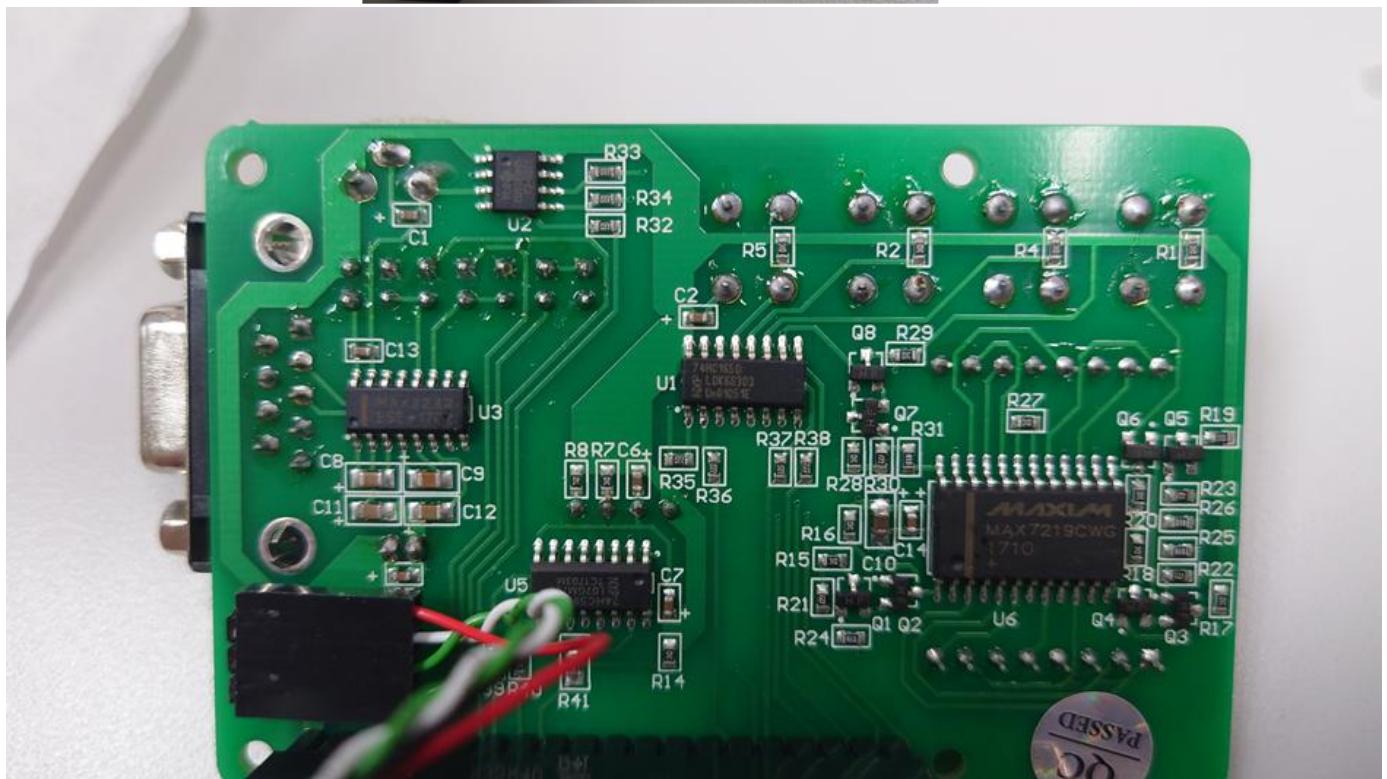
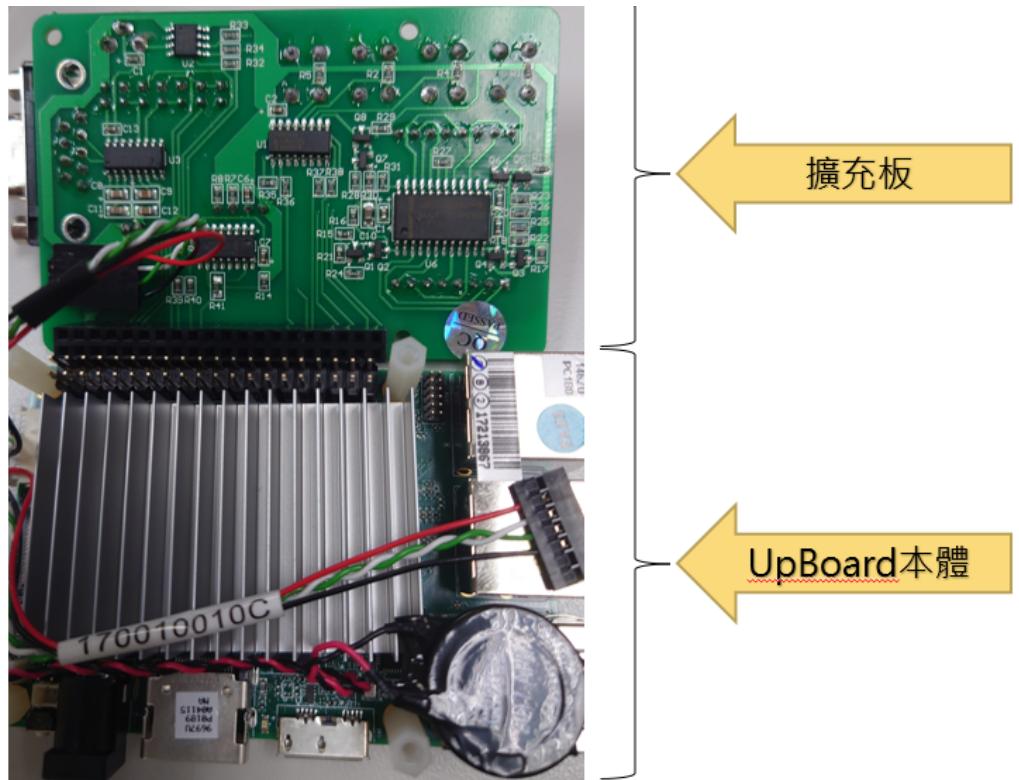
以下介紹之 74HC165、74HC595、MAX7219、MCP3201 均為 IC 的特定種類之型號代稱，實際型號可能有些許差異，但應用特性並無不同。

例如 74HC165 與 74HCT165 在本實驗中的應用中功能可視為相同

備註: 本實驗使用到的 IC 均在擴充版背面，看不到

1. 擴充板背面

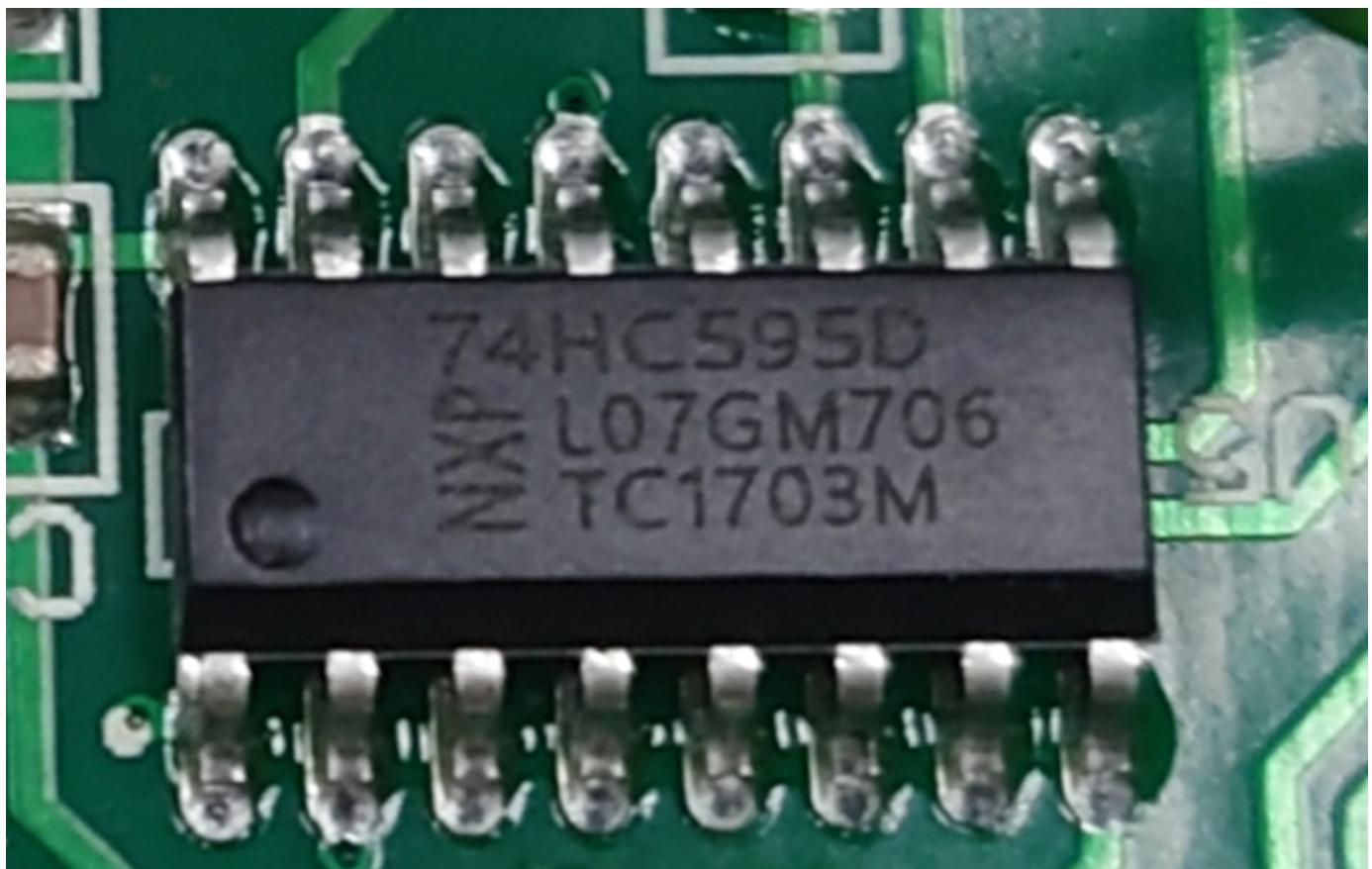
▶ 擴充板背面:



2. 74HC165長相



3. 74HC595長相



4. MAX7219長相



5. MCP3201長相



V. 實驗目的

問題

直覺來說，一般會以一個針腳控制一個裝置，如 LED 等裝置。但如果周邊裝置過多時會導致針腳數量不夠用，為了因應此問題，我們可以使用平行訊號 (parallel) 與序列訊號 (serial) 的轉換，實現少量針腳控制大量裝置之應用。

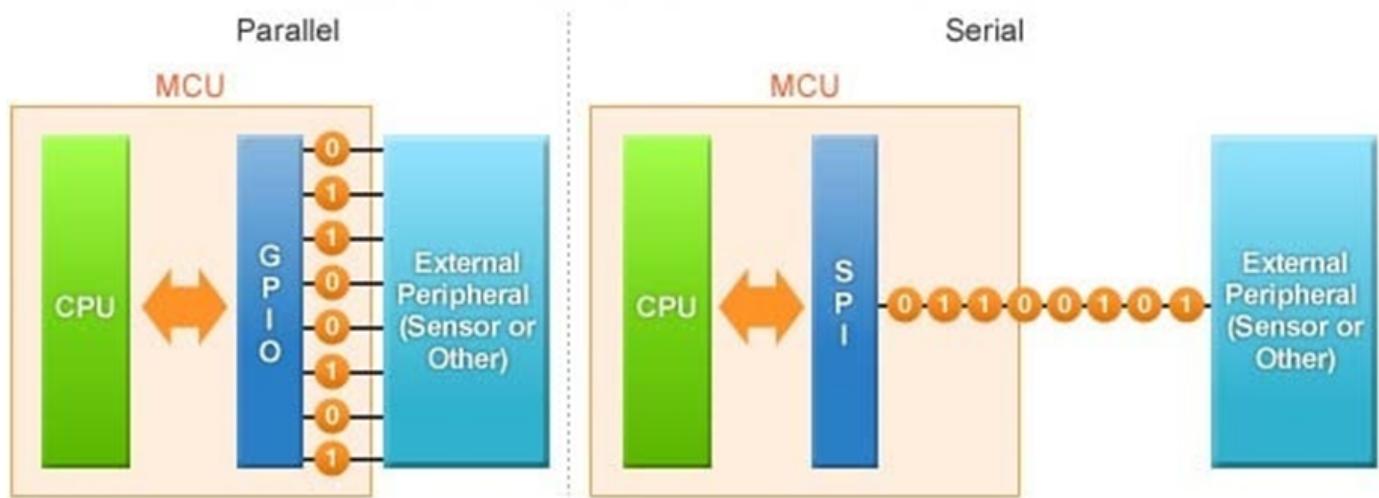
因此我們將學習符合平行 - 序列訊號轉換需求之特定 IC 型號的使用方法。

解決方式

1. 使用 74HC165 平行轉序列晶片
2. 使用 74HC595 序列轉平行晶片
3. 使用 MAX7219 控制 LED 矩陣

VI. 74HC165

74HC165 主要功能是 將平行輸入 (parallel) 轉為序列輸出 (serial)。其硬體規格也支援序列輸入，但本次實驗之應用不會使用到此功能。



1. 腳位說明

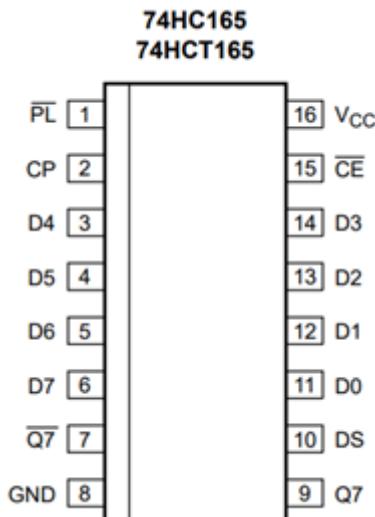


Table 2. Pin description

Symbol	Pin	Description
\overline{PL}	1	asynchronous parallel load input (active LOW)
CP	2	clock input (LOW-to-HIGH edge-triggered)
$\overline{Q7}$	7	complementary output from the last stage
GND	8	ground (0 V)
Q7	9	serial output from the last stage
DS	10	serial data input
D0 to D7	11, 12, 13, 14, 3, 4, 5, 6	parallel data inputs (also referred to as Dn)
\overline{CE}	15	clock enable input (active LOW)
V _{cc}	16	positive supply voltage

2. Operating mode

Operating modes	Inputs					Qn registers		Outputs	
	\overline{PL}	\overline{CE}	CP	DS	D0 to D7	Q0	Q1 to Q6	Q7	$\overline{Q7}$
parallel load	L	X	X	X	L	L	L to L	L	H
	L	X	X	X	H	H	H to H	H	L
serial shift	H	L	↑	I	X	L	q0 to q5	q6	$\overline{q6}$
	H	L	↑	h	X	H	q0 to q5	q6	$\overline{q6}$
	H	↑	L	I	X	L	q0 to q5	q6	$\overline{q6}$
	H	↑	L	h	X	H	q0 to q5	q6	$\overline{q6}$
hold "do nothing"	H	H	X	X	X	q0	q1 to q6	q7	$\overline{q7}$
	H	X	H	X	X	q0	q1 to q6	q7	$\overline{q7}$

PL:

- 當 PL 為低時將 D0~D7 之電位訊號平行載入至 74HC165 晶片的暫存器
- 當 PL 為高時停止並行載入資料

CE:

- CE 腳位為低時致能 CLK 訊號
- CE 腳位為高時 CLK

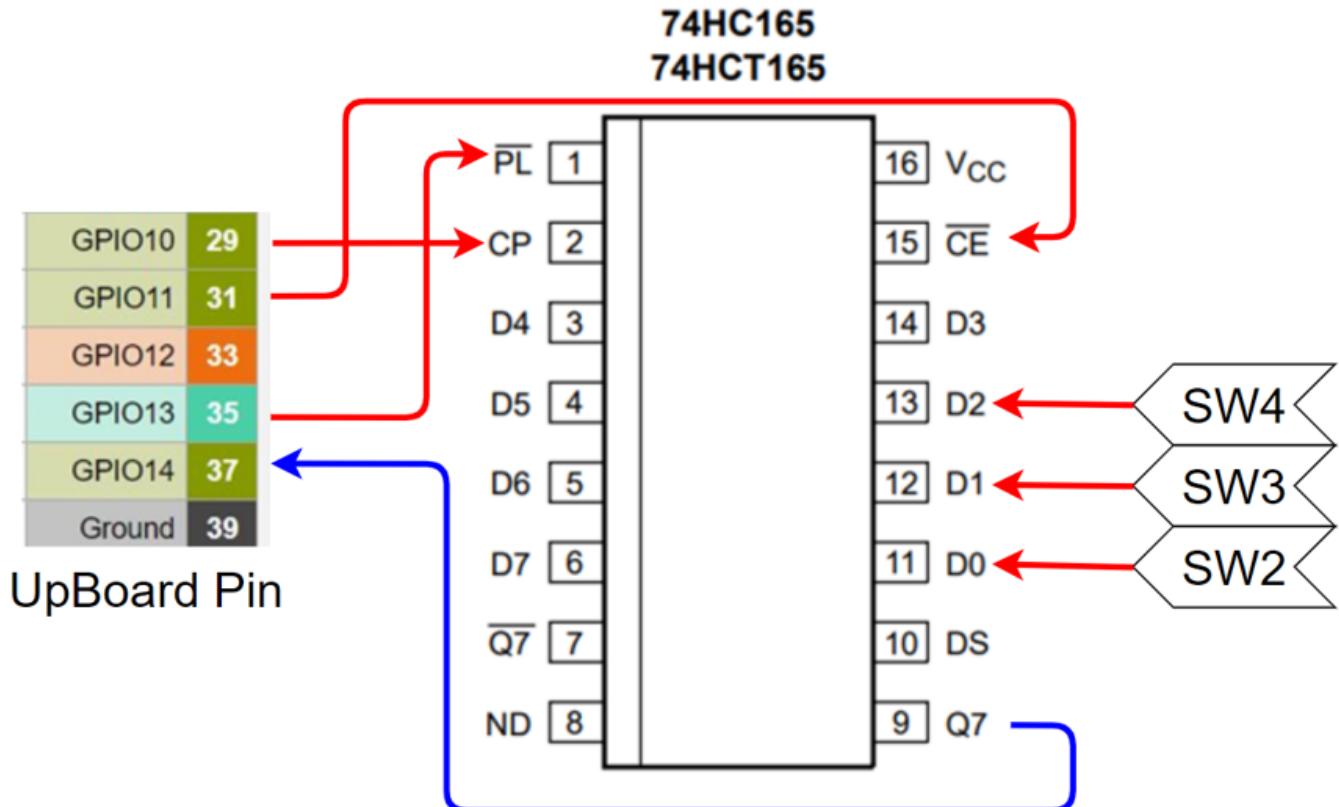
CP (CLK):

- Clock 每次產生正緣觸發時，D0~D7 進行右移
- 同時會將 DS 之電位訊號讀取至 D0 暫存器，本次實驗不會使用 DS 針腳

Q7 (DATA):

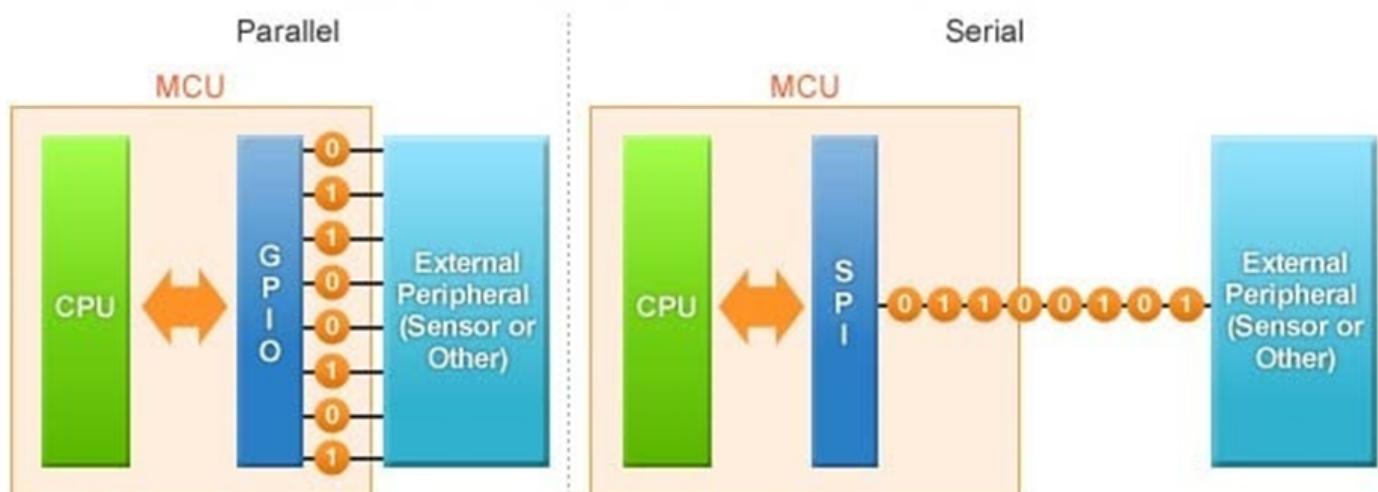
- 其電位訊號與 D7 之暫存器電位訊號相同，用於序列輸出

3. UpBoard pin diagram



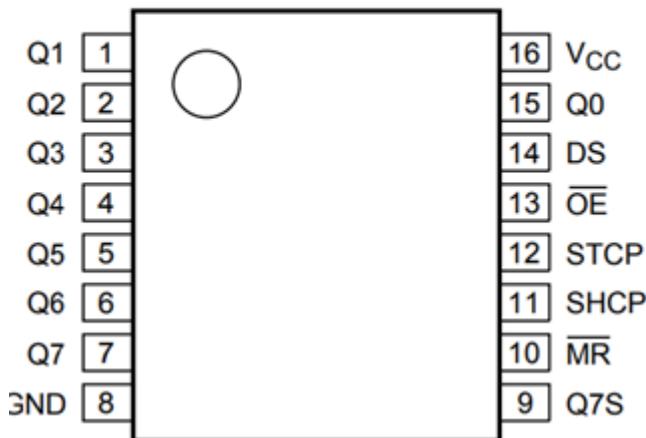
VII. 74HC595

74HC595 主要功能是將序列輸入 (serial) 轉為平行輸出 (parallel)。其硬體規格也支援序列輸出，本次實驗之應用只會使用到序列輸入轉為平行輸出的功能。



1. 腳位說明

74HC595 74HCT595

**Table 2. Pin description**

Symbol	Pin	Description
Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7	15, 1, 2, 3, 4, 5, 6, 7	parallel data output
GND	8	ground (0 V)
Q7S	9	serial data output
MR	10	master reset (active LOW)
SHCP	11	shift register clock input
STCP	12	storage register clock input
OE	13	output enable input (active LOW)
DS	14	serial data input
Q0	15	parallel data output 0
V _{cc}	16	supply voltage

2. Operating mode

Table 3. Function table [1]

Control				Input	Output	Function	
SHCP	STCP	OE	MR	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on MR only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

DS:

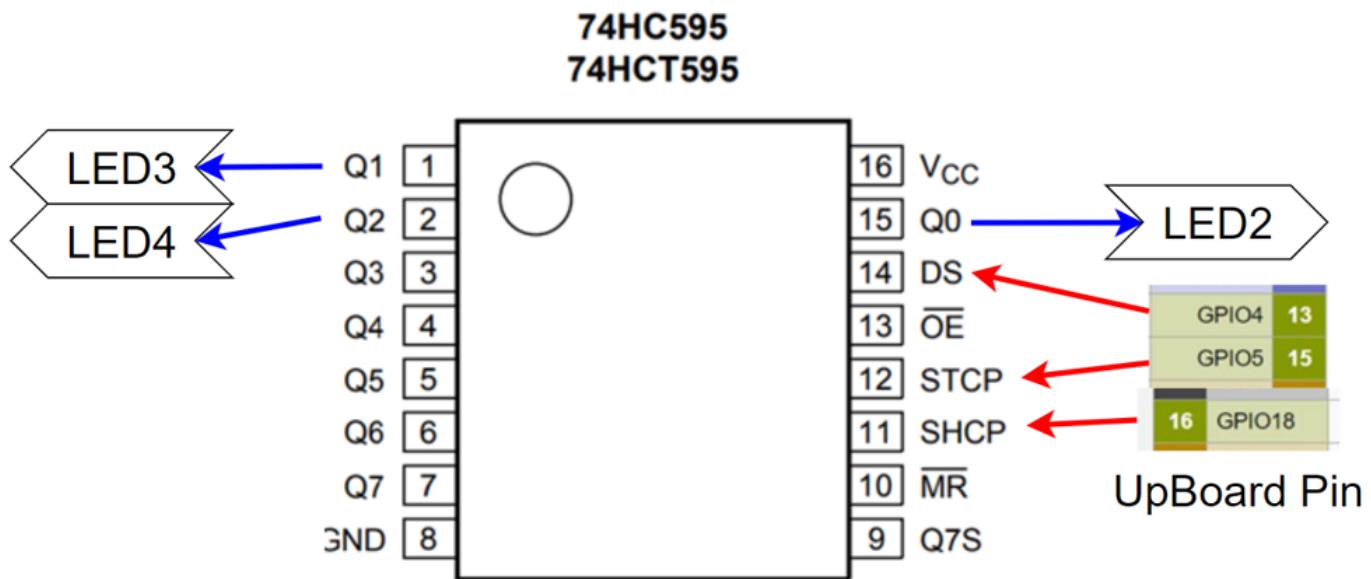
- 用於輸入序列訊號，被動等待 SHCP 發生正緣觸發時的讀取。

STCP:

- 當 STCP 發生正緣觸發時，將 Q0~Q7 暫存器之訊號輸出至其 Q0~Q7 對應之腳位。

SHCP:

- 當 SHCP 發生正緣觸發時，將 Q0~Q6 暫存器之訊號移動 (shift) 至 Q1~Q7 暫存器，例如 Q0 → Q1、Q6 → Q7，並讀取 DS 腳位之電位訊號至 Q0 暫存器。
- 當 SHCP 發生正緣觸發時，原本在 Q6 的值會移到 Q7，並透過 Q7S 進行序列輸出，本次實驗不會使用到 Q7S 針腳。

3. UpBoard pin diagram**VIII. MAX7219**

MAX7219 是一個 24 接腳的 DIP(Dual In-line Package) 封裝晶片，透過序列輸入 / 輸出驅動 8*8 LED矩陣、或 7 段顯示器

1. 腳位說明

TOP VIEW

PIN	NAME	FUNCTION
1	DIN	Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge.
2, 3, 5-8, 10, 11	DIG 0-DIG 7	Eight-Digit Drive Lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off.
4, 9	GND	Ground (both GND pins must be connected)
12	LOAD (MAX7219)	Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge.
	\overline{CS} (MAX7221)	Chip-Select Input. Serial data is loaded into the shift register while \overline{CS} is low. The last 16 bits of serial data are latched on \overline{CS} 's rising edge.
13	CLK	Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while \overline{CS} is low.
14-17, 20-23	SEG A-SEG G, DP	Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off.
18	ISET	Connect to VDD through a resistor (RSET) to set the peak segment current (Refer to <i>Selecting RSET Resistor and Using External Drivers</i> section).
19	V+	Positive Supply Voltage. Connect to +5V.
24	DOUT	Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance.

2. Data Format

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	0XA
Scan Limit	X	1	0	1	1	0XB
Shutdown	X	1	1	0	0	0XC
Display Test	X	1	1	1	1	0XF

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			

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	0XA
Scan Limit	X	1	0	1	1	0XB
Shutdown	X	1	1	0	0	0XC
Display Test	X	1	1	1	1	0XF

LOAD:

- 當 LOAD 為低時，允許序列資料（包括CLK與DIN）進入暫存器
- 當 LOAD 為高時，不允許序列資料進入暫存器
- 當 LOAD 發生正緣觸發時，將暫存器資料鎖住 (latched)

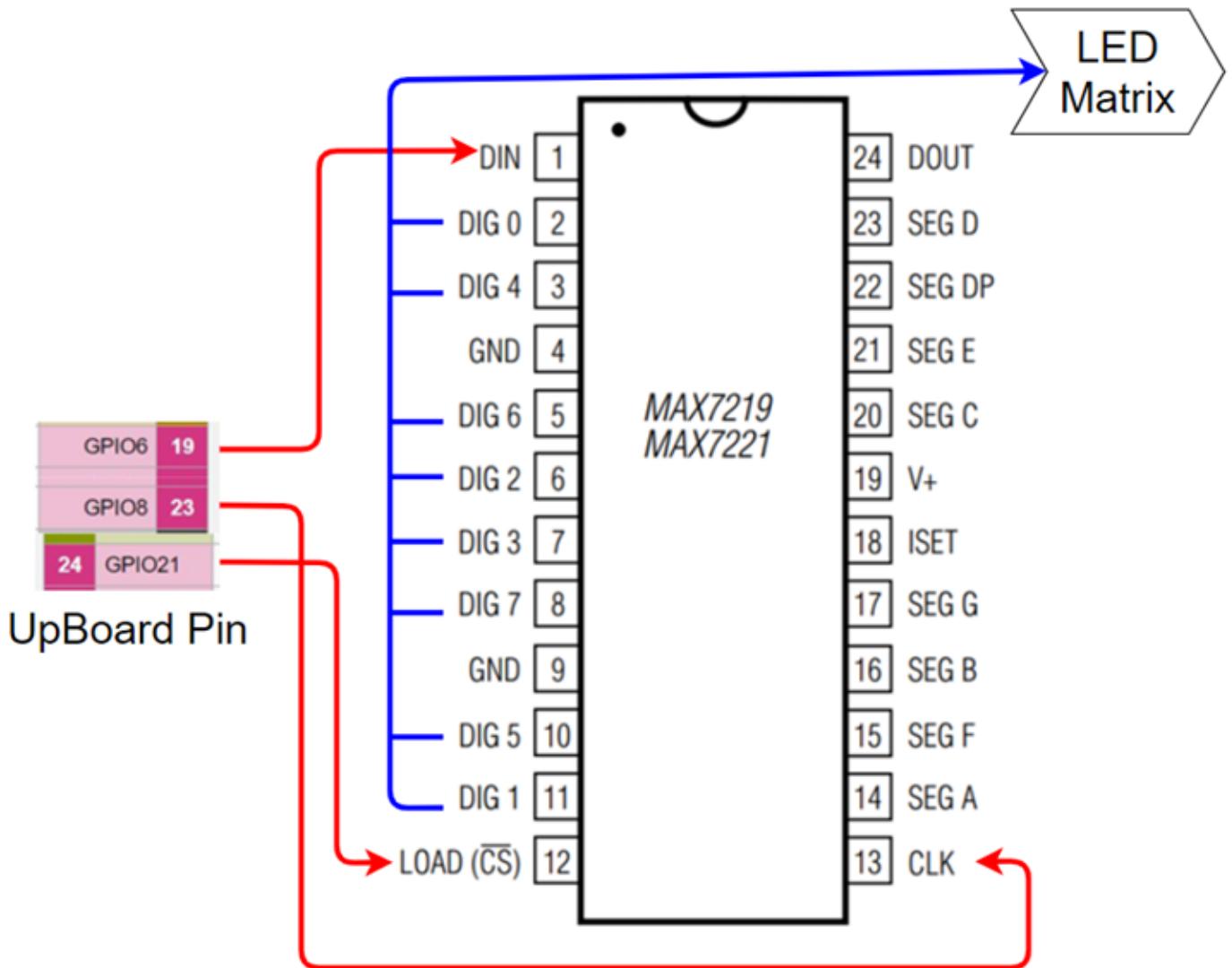
CLK:

- 當 CLK 發生正緣觸發時，從 DIN 讀取序列資料至暫存器

DIN:

- 用於輸入序列訊號，被動等待 CLK 發生正緣觸發時的讀取

3. UpBoard pin diagram



從 DIN 輸入的 16 bits 資料，格式應符合以下規範：

- D0~D7 為資料內容（值）
- D8~D11 為位址，例如：
第一行 LED 位址為0x1
0xa位址用來調整亮度

- D12~D15 不管 (硬體設計考量)

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使用前須初始化，方式為透過 DIN 以序列訊號的方式溝通，對特定位址寫入特定值步驟：

- 對 0x0c 寫入 0x01 : leave shutdown mode
- 對 0x0f 寫入 0x00 : turn off display test
- 對 0x09 寫入 0x00 : set decode mode: no decode
- 對 0x0b 寫入 0x07 : set scan limit to full
- 對 0xa 寫入 0x01 : set brightness (duty cycle = 3/32)

XI. ADC

實驗目的

本實驗將透過數位類比轉換器，量測外部電路中的可變電阻之分壓，從過程學到如何與數位類比轉換器等 SPI 裝置交換資訊。

XII. MCP3201

1. 腳位說明

Package Types

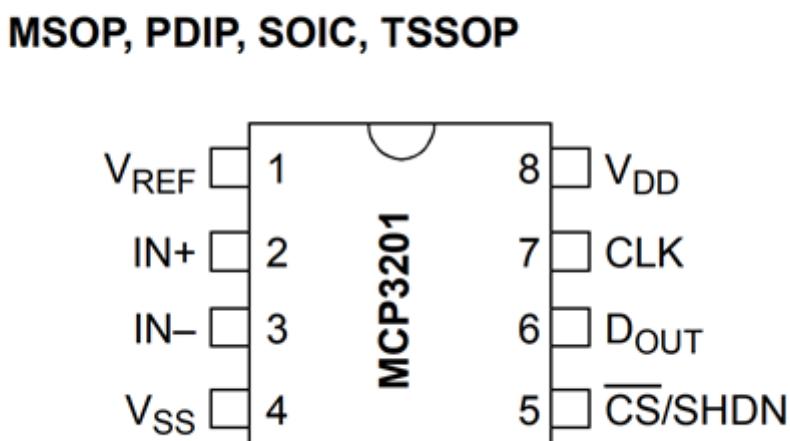


TABLE 3-1: PIN FUNCTION TABLE

MCP3201	Symbol	Description
MSOP, PDIP, SOIC, TSSOP		
1	V_{REF}	Reference Voltage Input
2	IN^+	Positive Analog Input
3	IN^-	Negative Analog Input
4	V_{SS}	Ground
5	$\overline{CS}/SHDN$	Chip Select/Shutdown Input
6	D_{OUT}	Serial Data Out
7	CLK	Serial Clock
8	V_{DD}	+2.7V to 5.5V Power Supply

CS:

- 當 CS 為高電位時，停止 SPI 通訊。
- 當 CS 為低電位時，開始 SPI 通訊。
-

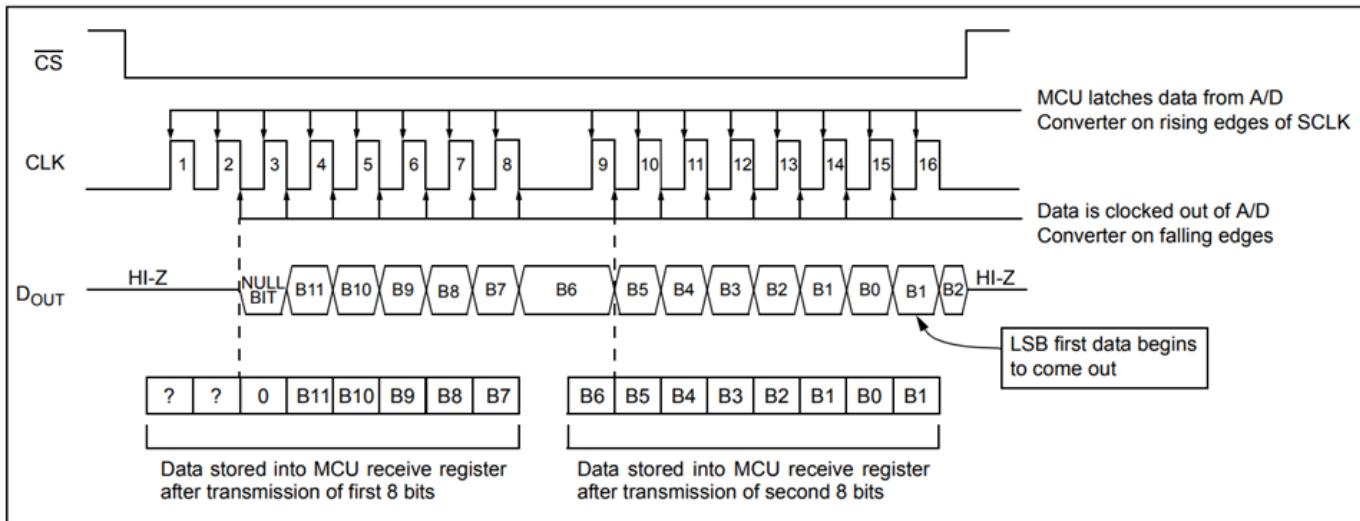
CLK:

- 當 CLK 發生負緣觸發時，從 IN^+ 讀取電壓值，與 V_{REF} 做比較得到電壓差的數位訊號，並存放於暫存器，但須注意讀取後約 200 ns 後之資料才準確 (tDO 硬體規格規定)，因此有兩種取值方式：
 - 直接在最穩定的正緣觸發時取值，此方法的有效位元（包括 null bit）為第 3~15 次取得的值，本實驗範例使用此方式取值。
 - 在負緣觸發後至少延遲 200 ns 後取值，此方法的有效位元為（包括 null bit）為第 2~14 次取得的值。

DOUT:

- 用於輸出序列訊號，等待 CLK 發生負緣觸發時的讀取後將暫存器內的值序列輸出。

2. SPI communication



3. 比例轉換公式

$$\frac{\text{測量值}}{\text{最大可測得範圍值}} = \frac{\text{實際輸入電壓}}{\text{參考最大電壓}}$$

$$\frac{\text{Digital Output Code}}{1 \ll 12(\text{left shift})} = \frac{V_{in}}{V_{REF}}, \text{ MCP3201 resolution} = 12\text{bits}$$

移項後可得 Digital Output Code = $\frac{V_{in} * 4096}{V_{REF}}$

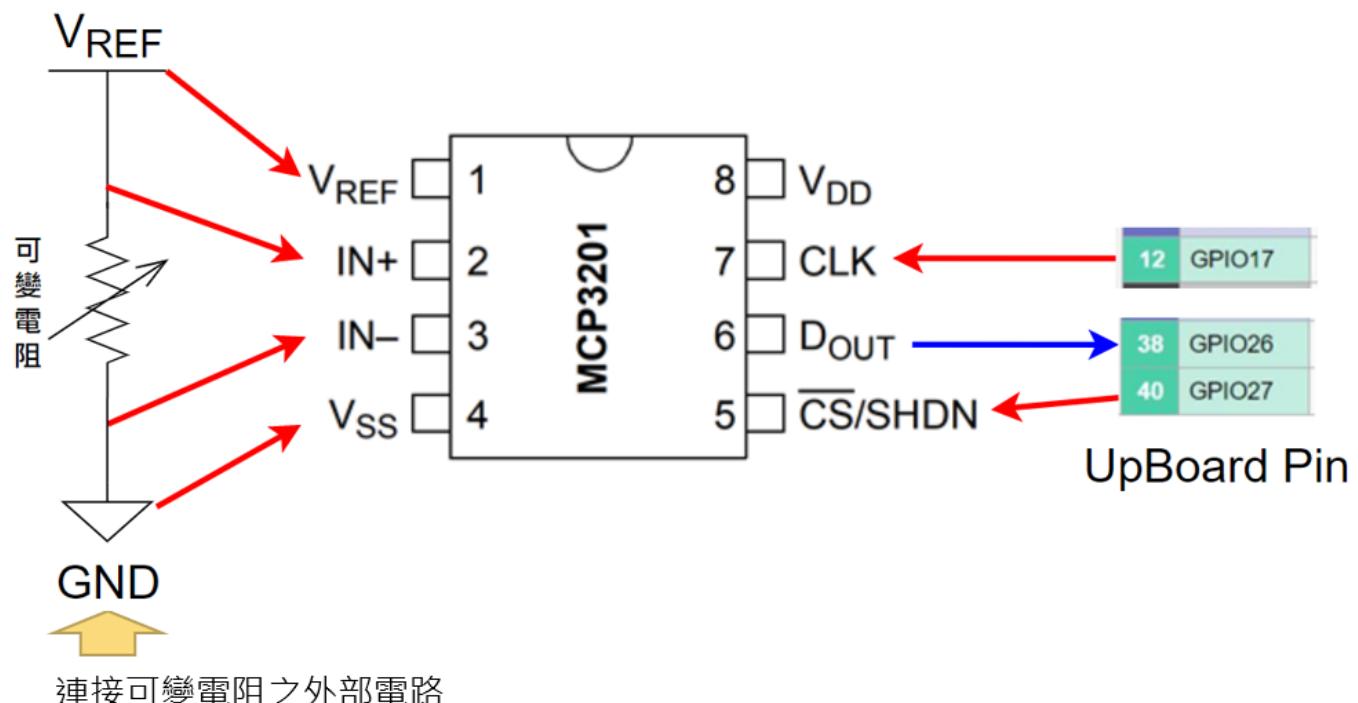
EQUATION 4-2:

$$\text{Digital Output Code} = \frac{4096 * V_{IN}}{V_{REF}}$$

Where:

$$\begin{aligned} V_{IN} &= \text{Analog Input Voltage} = V_{(IN^+)} - V_{(IN^-)} \\ V_{REF} &= \text{Reference Voltage} \end{aligned}$$

4. UpBoard pin diagram



XIII. PWM

1. 實驗目的

- 問題

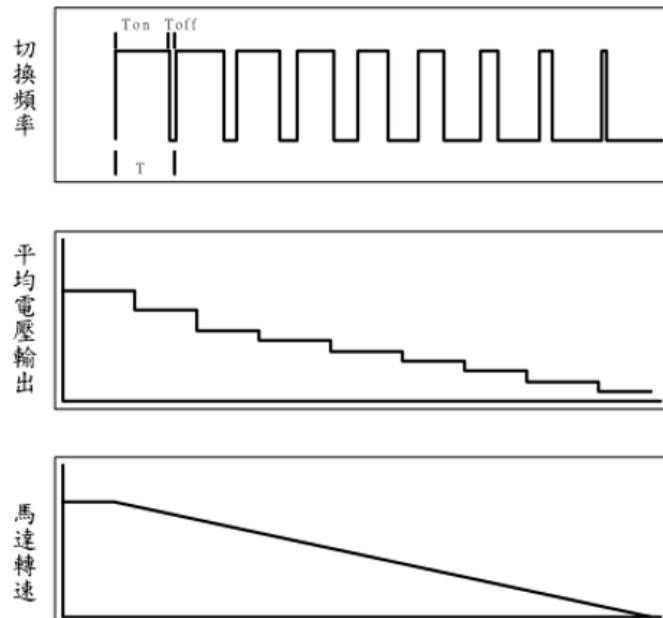
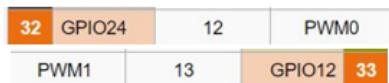
- 類比訊號可利用電壓大小決定輸出的功率，在額定工作範圍內，通常越高的電壓差能產生較高的工作效率，例如提升 LED 亮度、改變馬達轉速等。但數位訊號僅有高電位與低電位兩種電壓階級，若要達到與類比相同的電壓調整功能，則需要特殊的轉換電路。

- 解決方式

- 脈波寬度調變 (PWM) 可以藉由 duty cycle 的調整，在使用數位訊號的條件下模擬出類比訊號的電壓調整功能。

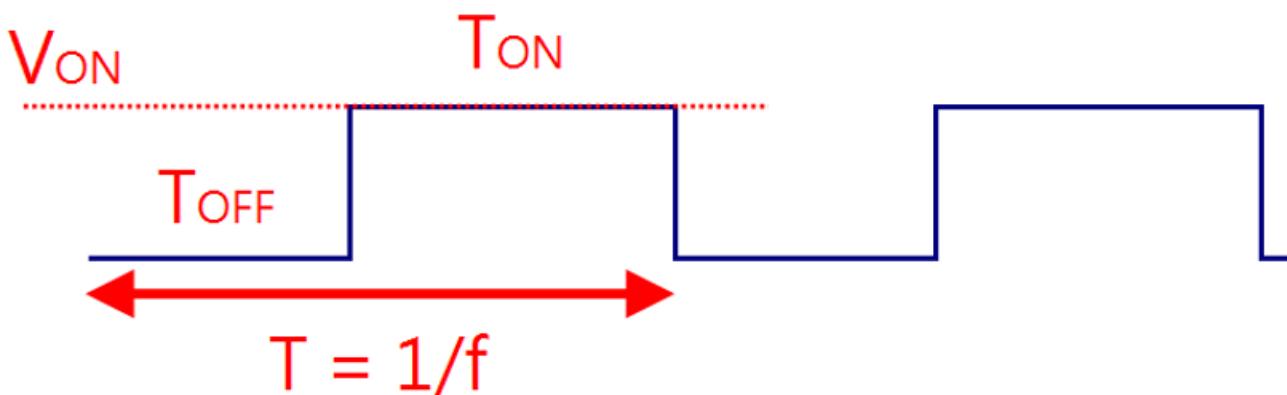
▶ UP board PWM 頻率：

- 293 Hz ~ 6.4 MHz
- PIN:
 - PWM0: 32
 - PWM1: 33



2. 運作原理

- 切換週期 T ，通常我們會用頻率 f (Hz) 來表達
- V_{on} 為最大可輸出的電壓



- $duty\ cycle = Ton/Toff\ (%)$ ，為電壓處於高電位 (ON) 的時間長與電壓處於低電位 (OFF) 時間長相除的百分比值

- 模擬出的電壓 $V = V_{on} \times \text{duty cycle} (\%)$

可以知道 duty cycle 越高，模擬電壓 V 就越高，當完全不切至低電位的時候，duty cycle = 100%，輸出電壓 V 將維持在最大可輸出的電壓 V_{on} 。

Duty cycle = 50%



Duty cycle = 20%

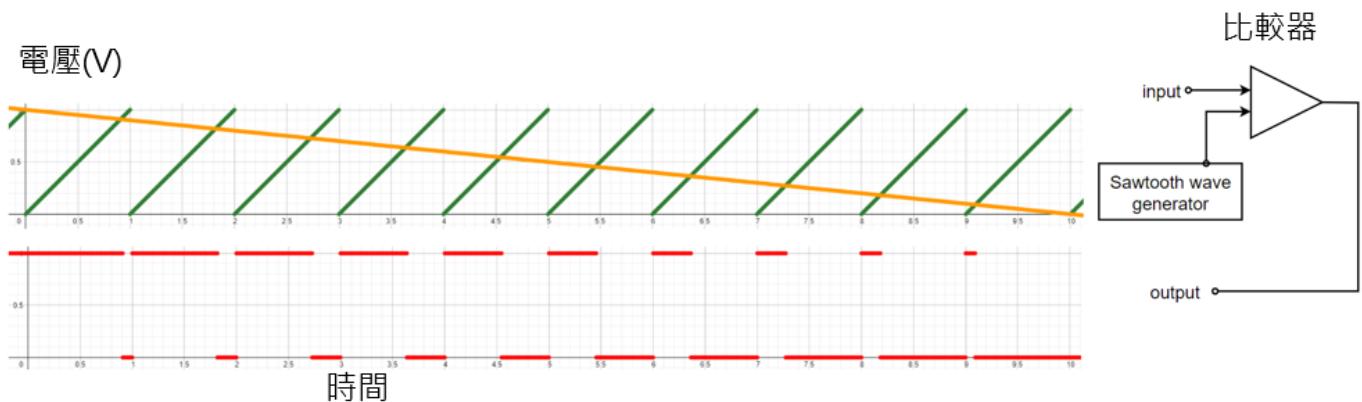


Duty cycle = 80%



將輸入的電壓值與頻率固定之鋸齒波透過比較器進行電位高低的比較，結果將決定工作頻率比 (duty cycle)。

- 黃線：輸入電壓
- 綠線：鋸齒波之電壓
- 紅線：比較器之輸出，同義於利用 duty cycle 產生的模擬電壓



3. PWM in mraa

- dev = mraa_pwm_context mraa_pwm_init (int pin)
初始化 pwm 裝置連接針腳 (pin) 取得裝置資料 mraa_pwm_context dev
- mraa_result_t mraa_pwm_period_us (mraa_pwm_context dev, int us)
設定 pwm 裝置 dev 的工作週期 (us 表示週期，單位: 微秒)
- mraa_result_t mraa_pwm_enable (mraa_pwm_context dev, int enable)
開啟 pwm 裝置 dev (enable: 0 = 關閉、1 = 開啟)
- mraa_result_t mraa_pwm_write (mraa_pwm_context dev, float percentage)
依據輸入的百分比值 (percentage 範圍: 0.0~1.0) 決定該裝置 (dev) 的 duty cycle
- mraa_result_t mraa_pwm_close (mraa_pwm_context dev)
關閉 pwm 裝置 dev

B. Demonstration

I. 按鍵輸入偵測

- 利用 74HC165 IC 驅動來達到多顆按鍵偵測功能。按下 SW2、SW3、SW4 做按鍵偵測，結果顯示於螢幕上。
 - 單個 74HC165 可以檢測 8 個平行輸入訊號，所以會有 0~7 共八個輸入的暫存值
 - 此實驗只用到 SW2,SW3,SW4 共三個輸入訊號，分別對應的腳位為 D0、D1、D2，觀察 SW2, SW3, SW4 按鈕按壓後，D0、D1、D2 三個輸出的高低電位變化，有按壓時輸出 0，沒按壓時輸出 1。

```

26 int main() {
27     int switch_status[8] = {0};
28     // PL: asynchronous parallel load input (active LOW)
29     mraa_gpio_context pin_pl = mraa_gpio_init(UP_HAT_74HC165_PL);
30     // Q7: serial output from the last stage
31     mraa_gpio_context pin_data = mraa_gpio_init(UP_HAT_74HC165_Q7);
32     // CE: clock enable input (active LOW)
33     mraa_gpio_context pin_ce = mraa_gpio_init(UP_HAT_74HC165_CE);
34     // CP: clock input (LOW-to-HIGH edge-triggered)
35     mraa_gpio_context pin_clk = mraa_gpio_init(UP_HAT_74HC165_CP);
36
37     if (!(pin_pl && pin_data && pin_ce && pin_clk)) {
38         fprintf(stderr, "Failed to initialize GPIO. Did you run with sudo?\n");
39         return EXIT_FAILURE;
40     }
41
42     mraa_gpio_dir(pin_pl, MRAA_GPIO_OUT);
43     mraa_gpio_dir(pin_data, MRAA_GPIO_IN);
44     mraa_gpio_dir(pin_ce, MRAA_GPIO_OUT);
45     mraa_gpio_dir(pin_clk, MRAA_GPIO_OUT);
46
47     while (!stopped) {
48         mraa_gpio_write(pin_ce, 1); // disable clk input
49         delay_ns(1000);
50         mraa_gpio_write(pin_pl, 0); // enable parallel data input
51         delay_ns(1000);
52         mraa_gpio_write(pin_pl, █); // disable & hold parallel data input
53         delay_ns(1000);
54         mraa_gpio_write(pin_ce, █); // enable clk input
55         delay_ns(1000);
56
57         for (int i = 7; i >= 0; i--) {
58             mraa_gpio_write(pin_clk, 0); // unset clk signal
59             delay_ns(1000);
60             switch_status[i] =
61                 mraa_gpio_read(pin_data); // read one bit data from pin_data
62             mraa_gpio_write(pin_clk, █); // set clk signal
63             delay_ns(1000);
64         }
65         mraa_gpio_write(pin_clk, 0); // unset clk signal
66         delay_ns(1000);
67         for (int i = █; i < █; i++) {
68             printf("SW%d = %d ", █, switch_status█); // print switch status
69         }
70     }
71     printf("\n");
72     delay_ms(100);
73 }

```

初始化

載入一次平行訊號

依序讀取序列訊號
注意順序: D7的暫存器值會先出來
所以先放在陣列中的第 7 個位置
因此陣列中每個元素的index即代表來自Di暫存器

印出按鈕狀態
注意: 序列輸出是從 D7 開始輸出
接收時注意順序，SW2~4 接在 D0~2

Reminder: 74HC165

PL:

- 當 PL 為低時將 D0~D7 之電位訊號平行載入至 74HC165 晶片的暫存器
- 當 PL 為高時停止並行載入資料

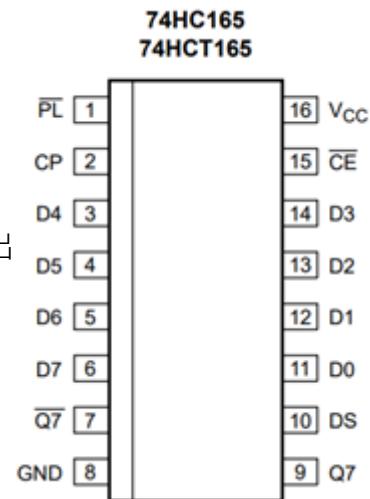
CE:

- CE 腳位為低時致能 CLK 訊號
- CE 腳位為高時 CLK

CP (CLK):

- Clock 每次產生正緣觸發時，D0~D7 進行右移
- 同時會將 DS 之電位訊號讀取至 D0 暫存器，本次實驗不會使用 DS 針腳

Q7 (DATA):



- 其電位訊號與 D7 之暫存器電位訊號相同，用於序列輸出

II. 基礎跑馬燈應用：多顆LED輸出

使用 74HC595 IC 驅動多顆 LED 輸出功能。板子上的 LED2、LED3、LED4 執行跑馬燈功能。

寫入序列資料進入 74HC595

- STCP (en) 為了待會的正緣觸發先拉低
- SHCP (sl) 為了待會的正緣觸發先拉低
- 將序列值寫入 DS (data)
- 對 SHCP 發出正緣觸發，先將所有暫存器位移一次，並將 DS 的值讀入 Q0
 - Q0~Q6 → Q1~Q7
 - Q0 = DS
- 2.~4.重複8次，相當於讀取 8 bits 長度的序列資料進入暫存器

從 74HC595 平行輸出

- 對 STCP 發出正緣觸發，將 Q0~Q7 的暫存器之值輸出至對應之 Q0~Q7 腳位

▶ 前置

方便稍後使用之
LED 亮滅順序

```

26 int main() {
27     int led_template[3][8] = {
28         {1,0,0,0,0,0,0,0},
29         {0,1,0,0,0,0,0,0},
30         {0,0,1,0,0,0,0,0},
31     };
32     int led_status[8] = {0};
33     // pin 13(DS): serial data pin
34     mraa_gpio_context pin_data = mraa_gpio_init(UP_HAT_74HC595_DS);
35     // pin 15(STCP): when positive edge trigger, output shift register state
36     mraa_gpio_context pin_en = mraa_gpio_init(UP_HAT_74HC595_STCP);
37     // pin 16(SHCP): when positive edge trigger, shift parallel register & load from pin_DS
38     mraa_gpio_context pin_sl = mraa_gpio_init(UP_HAT_74HC595_SHCP);

39
40     if (!(pin_data && pin_en && pin_sl)) {
41         fprintf(stderr, "Failed to initialize GPIO. Did you run with sudo?\n");
42         return EXIT_FAILURE;
43     }
44     mraa_gpio_dir(pin_data, MRAA_GPIO_OUT);
45     mraa_gpio_dir(pin_en, MRAA_GPIO_OUT);
46     mraa_gpio_dir(pin_sl, MRAA_GPIO_OUT);
47
48     int t = 0; // time variable

```

▶ While loop 內容:

取出先前設定好
的亮滅順序

寫入 74HC595
之暫存器

將暫存器內容寫
至針腳 Q0~Q7

```

50     while (!stopped) {
51         for (int i = 0; i < LED_COUNT; ++i) { // for each LED
52             led_status[i] = led_template[t][i]; // (t == i);
53             /* t=0: {1,0,0,0,0,0,0,0}
54             * t=1: {0,1,0,0,0,0,0,0}
55             * t=2: {0,0,1,0,0,0,0,0}
56             */
57         }
58         mraa_gpio_write(pin_en, 0); // unset pin_en to simulate positive edge later
59         for (int i = 7; i >= 0; i--) { // for each bit from pin_DS (notice: order is reverse)
60             // unset pin_clk to simulate positive edge later
61             mraa_gpio_write(pin_sl, 0);
62             // send led_status to data_pin.
63             mraa_gpio_write(pin_data, led_status[i]);
64             // simulate positive edge to load & shift 1 bit data
65             mraa_gpio_write(pin_sl, 1);
66         }
67         mraa_gpio_write(pin_en, 1); // set pin_en: output register state to light up LED.
68         delay_seconds(1);
69         // t = {0, 1, 2} loop
70         t++;
71         if (t >= LED_COUNT)
72             t = 0;
73     }

```

Reminder: 74HC595

DS:

- 用於輸入序列訊號，被動等待 SHCP 發生正緣觸發時的讀取。

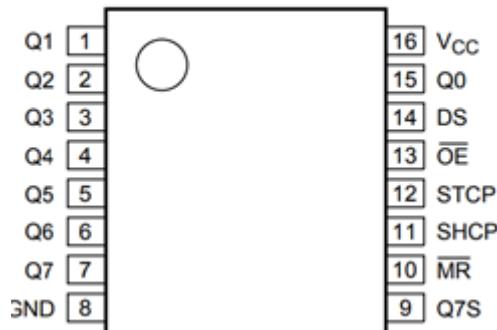
STCP:

- 當 STCP 發生正緣觸發時，將 Q0~Q7 暫存器之訊號輸出至其 Q0~Q7 對應之腳位。

SHCP:

- 當 SHCP 發生正緣觸發時，將 Q0~Q6 暫存器之訊號移動 (shift) 至 Q1~Q7 暫存器，例如 Q0 → Q1、Q6 → Q7，並讀取 DS 腳位之電位訊號至 Q0 暫存器。
- 當 SHCP 發生正緣觸發時，原本在 Q6 的值會移到 Q7，並透過 Q7S 進行序列輸出。

74HC595
74HCT595



本次實驗不會使用到 Q7S 針腳。

III. 基礎點矩陣應用：矩陣跑馬燈

使用 MAX7219 IC 驅動 8×8 LED 點矩陣。板子上點矩陣執行跑馬燈功能

- 將符合 MAX7219 資料格式的內容傳入 IC

- sent_byte

- 傳入參數 : 8 bits 序列訊息
 - 作用 : 傳送序列訊息至MAX7219

- CLK為了待會的正緣觸發先拉低
- 將參數從右邊數來第 i 個 bit 輸出至 DIN，因為須從最左側的 bit (MSB) 開始傳，因此 i = 7~0
- 對 CLK 發出正緣觸發，在 LOAD 為低電位的條件下可讀取 DIN 進入暫存器
- 重複1.~3.，直到 8 bits 都送達 MAX7219

```

26  /**
27   * Send a byte over the DIN pin
28   */
29 void send_byte(uint8_t d) {
30     for (int i = 7; i >= 0; --i) {
31       mraa_gpio_write(pin_clk, 0);
32       delay_ns(1000);
33       mraa_gpio_write(pin_din, (d >> i) & 1u);
34       delay_ns(1000);
35       mraa_gpio_write(pin_clk, 1);
36       delay_ns(1000);
37     }
38 }
```

- write_reg

- 傳入參數：16 bits 序列長訊息，此處為了理解拆成前半 8 bits 之 addr 與後半 8 bits 之 data
- 作用：傳送序列訊息至 MAX7219

1. CLK 設為高電位，使訊息間不互相影響
2. LOAD 拉低，允許序列資料（包括CLK與DIN）進入暫存器
3. 利用sent_byte 函式傳送前 8 bits
4. 利用sent_byte 函式傳送後 8 bits
5. CLK 設為低電位，使訊息間不互相影響
6. LOAD 拉低，不允許序列資料進入暫存器，並將暫存器資料鎖住

```

40  /**
41   * Write data to the register at addr
42   */
43 void write_reg(uint8_t addr, uint8_t data) {
44     mraa_gpio_write(pin_clk, 1);
45     mraa_gpio_write(pin_load, ■);
46     send_byte(addr);
47     send_byte(data);
48     mraa_gpio_write(pin_clk, 0);
49     mraa_gpio_write(pin_load, 1);
50     delay_ns(1000);
51 }

```

- 依照使用手冊，初始化 LED 矩陣

- init_matrix
- 作用：利用 write_reg (addr, data) 函式調整 LED 矩陣的設定

1. 對 0x0c 寫入 0x01 : leave shutdown mode
2. 對 0x0f 寫入 0x00 : turn off display test
3. 對 0x09 寫入 0x00 : set decode mode: no decode
4. 對 0x0b 寫入 0x07 : set scan limit to full
5. 對 0x0a 寫入 0x01 : set brightness (duty cycle = 3/32)

```

56 void init_matrix() {
57     // Refer to the datasheet for the meaning of these numbers
58
59     write_reg(0x0c, 0x01); // leave shutdown mode
60     write_reg(0x0f, 0x00); // turn off display test
61     write_reg(0x09, 0x00); // set decode mode: no decode
62     write_reg(0x0b, 0x07); // set scan limit to full
63     write_reg(0x0a, ■); // set brightness (duty cycle = 3/32)
64 }

```

▶ 前置

```

66 int main() {
67     pin_load = mraa_gpio_init(UP_HAT_MAX7219_LOAD);
68     pin_din = mraa_gpio_init(UP_HAT_MAX7219_DIN);
69     pin_clk = mraa_gpio_init(UP_HAT_MAX7219_CLK);
70
71     if (!(pin_load & pin_din & pin_clk)) {
72         fprintf(stderr, "Failed to initialize GPIO. Did you run with sudo?\n");
73         return EXIT_FAILURE;
74     }
75
76     mraa_gpio_dir(pin_load, MRAA_GPIO_OUT);
77     mraa_gpio_dir(pin_din, MRAA_GPIO_OUT);
78     mraa_gpio_dir(pin_clk, MRAA_GPIO_OUT);
79
80     init_matrix();

```

初始化 調整 LED 矩陣設定

▶ while loop

將第一~八排的亮滅樣式 傳
進MAX7219

注意: LED 矩陣位於 MAX7219 暫存器之
addr = 1~8 的位址

亮滅樣式變換

```

86     while (!stopped) {
87         // To set the pattern of the nth row (1-indexed), write to register n the
88         // 8-bit pattern
89         for (int i = 1; i <= 8; ++i) {
90             write_reg(i, row_pattern);
91         }
92         // rotate pattern left
93         row_pattern = row_pattern < (1 << 7) ? row_pattern << 1 : 1;
94         delay_ms(100);
95     }

```

Reminder: MAX7219

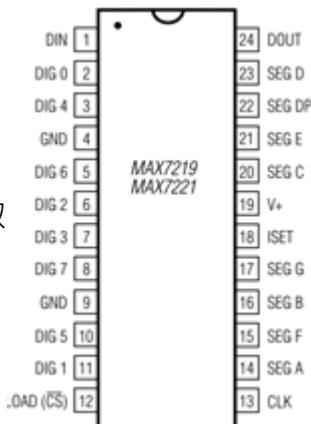
LOAD:

- 當 LOAD 為低時，允許序列資料（包括CLK與DIN）進入暫存器
- 當 LOAD 為高時，不允許序列資料進入暫存器
- 當 LOAD 發生正緣觸發時，將暫存器資料鎖住 (latched)

CLK:

- 當 CLK 發生正緣觸發時，從 DIN 讀取序列資料至暫存器

DIN:



- 用於輸入序列訊號，被動等待 CLK 發生正緣觸發時的讀取

IV ADC

利用 MCP3201 IC。撥動開發版上的可變電阻 VR1，測量可變電阻 VR1 之分壓，並將結果顯示於螢幕上

SPI 傳送資料

- 把 CS 拉低以開始 SPI 溝通
- 對 CLK 發出正緣觸發，使 ADC 進行類比數位轉換，並將當前的數值放在暫存器
- 將該位元之數值經由 DOUT 讀出並記錄
- 當下累加值 = 上次累加值 << 1 + 此次讀得資料
前三位元不累加
- 重複 2.~ 4. 直到 12 位元讀完
- 把 CS 拉高結束 SPI 溝通

▶ 前置

MCP3201 硬體最高
解析度 = 12 bits →

參考電壓為 3.3 伏特 (V) →

初始化

```

18 // ADC output resolution in bits
19 #define ADC_RESOLUTION 12
20 // Reference Voltage
21 #define VREF 3.3f
22 int main() {
23     // This line is used for resource collection. Ignore it.
24     signal(SIGINT, int_handler);
25
26     // pin 12(CLK): clock pin
27     pin_clk = mraa_gpio_init(UP_HAT_MCP3201_CLK);
28     // pin 38(DOUT): SPI data pin
29     pin_data = mraa_gpio_init(UP_HAT_MCP3201_DOUT);
30     // pin 40(CS/SHDN): pin_cs used to initiate communication with the device
31     pin_cs = mraa_gpio_init(UP_HAT_MCP3201_CS);
32
33     mraa_gpio_dir(pin_clk, MRAA_GPIO_OUT);
34     mraa_gpio_dir(pin_data, MRAA_GPIO_IN);
35     mraa_gpio_dir(pin_cs, MRAA_GPIO_OUT);
36
37 }
```

► while loop

SPI 溝通開始 →

讀值
注意: 因為讀得的值是二進位，
因此要利用 shift 轉換，
且前 3 bits 不管(第0~2位)，
只取第 3~15 位

SPI 溝通結束 →

代入轉換公式得到實際電壓
(數位形式)

```

47     while (!stopped) {
48         mraa_gpio_write(pin_cs, 0); // SPI communication start
49         delay_ns(1000);
50         uint16_t data_ADC = 0;
51         for (int SPICount = 0; SPICount < 15; SPICount++) {
52             mraa_gpio_write(pin_clk, 0);
53             delay_ns(1000);
54             mraa_gpio_write(pin_clk, 1);
55             if (SPICount >= 3) { // first 3 cycles doesn't matter
56                 data_ADC = (data_ADC << 1) + mraa_gpio_read(pin_data);
57             }
58             delay_ns(1000);
59         }
60         mraa_gpio_write(pin_cs, 1); //SPI communication end
61         float voltage = VREF * data_ADC / (1 << ADC_RESOLUTION);
62         printf("%fv\n", voltage);
63         delay_ms(10);
64     }

```

Reminder: MCP3201

CS

- 當 CS 為高電位時，停止 SPI 通訊。
- 當 CS 為低電位時，開始 SPI 通訊。

CLK

- 當 CLK 發生負緣觸發時，從 IN+ 讀取電壓值，與 VREF 做比較得到電壓差的數位訊號，並存放於暫存器，但須注意讀取後約 200 ns 後之資料才準確 (tDO 硬體規格規定)，因此有兩種取值方式：
 - (1) 直接在最穩定的正緣觸發時取值，此方法的有效位元（包括 null bit）為第 3~15 次取得的值，本實驗範例使用此方式取值。
 - (2) 在負緣觸發後至少延遲 200 ns 後取值，此方法的有效位元為（包括 null bit）為第 2 ~14 次取得的值。

DOUT

- 用於輸出序列訊號，等待 CLK 發生負緣觸發時的讀取後將暫存器內的值序列輸出。

PWM

使用系統內建 PWM 輸出至 LED 5 上。當調變脈波寬度時，LED5 有亮暗變化。

- 使用步驟：
 - 初始化
 - (1) 初始化 PWM 針腳

- (2) 設定 PWM 週期
- (3) 開啟 PWM 針腳
- 使用
 - 對 PWM 針腳寫入 duty cycle 的百分比值

```
23 int main() {  
24     // pin 32: pwm pin with LEDs  
25     mraa_pwm_context pin_pwm = mraa_pwm_init(UP_HAT_LED5);  
26     mraa_pwm_period_us(pin_pwm, 200); // set pwm frequency to 5000hz  
27     mraa_pwm_enable(pin_pwm, 1);      // enable pwm pin  
28  
29     float duty_cycle = 0.0f;  
30  
31     signal(SIGINT, int_handler);  
32     while (!stopped) {  
33         printf("DUTY CYCLE: %f\n", duty_cycle);  
34         mraa_pwm_write(pin_pwm, [REDACTED]); // change LED duty cycle  
35  
36         duty_cycle = duty_cycle + 0.01f; // increment the duty cycle  
37         if (duty_cycle >= 1.0f) {  
38             duty_cycle = 0.0f;  
39         }  
40         delay_ms(10);  
41     }
```