

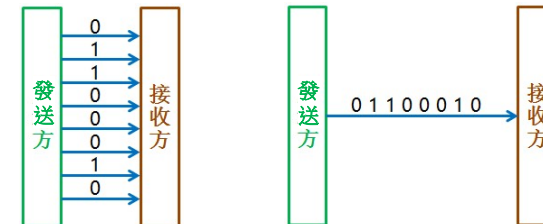
ESP32 DAY3

Serial Communication, SPI
8*8 Dot Matrix via Max7219
240*280 TFT 1.69" screen via ST7789V3

資料通訊

• 通訊傳輸依同一時間傳輸的資料線數多寡，可分為兩種：

1. 並列傳輸 (Parallel): 處理器與周邊之間有多條資料線連結，能一口氣輸出或接收多個位元資料。
2. 串列傳輸 (Serial, 又稱序列傳輸): 用一條信號線，每單位時間傳送一個位元資料。



串列傳輸

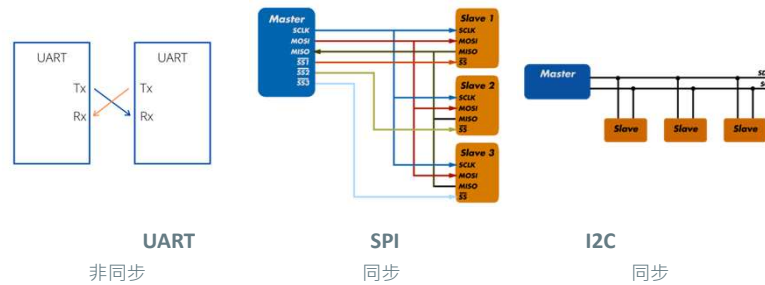
Serial Communication

名詞解釋

- 通訊方式依傳輸方向可分為單工、半雙工、全雙工
 1. 單工(Simplex): 線路上的訊號只能做單向傳送
 2. 半雙工(Half-duplex): 線路上的訊號可以雙向傳送，但是不能同時傳送
 3. 全雙工(Full-duplex): 線路上的訊號可以同時雙向傳送
- 通訊協定(Communication Protocols)定義電腦間互相通訊的協議標準
- 串列傳輸依資料是否同步，可區分：
 1. Asynchronous transmission (非同步傳輸)
 2. Synchronous Transmission (同步傳輸)

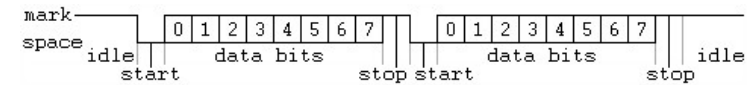
<https://wiki.csie.ncku.edu.tw/embedded/USART>

串列傳輸們



Asynchronized Transmission

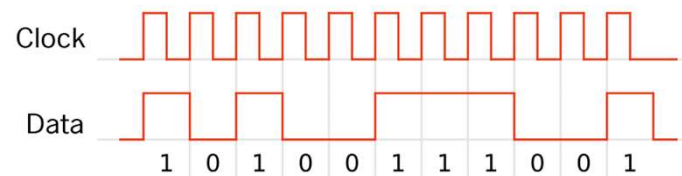
- Only one line/pin is needed to transfer data, and two sides need to use the same transfer rate / bps(bits per second) / baud rate.



- Special changes in signal is needed to “mark” the beginning and ending or “frame” one unit of data.

Synchronized Transmission

At least two pins/lines are needed, one line for clock signal, and then other one for data.



UART

UART

- 通用非同步收發傳輸器 (Universal Asynchronous Receiver/Transmitter · 簡稱為 UART)
- 將資料以串列通信的方式傳輸。
- 把接收到的資料轉換成bit · 並依照事先設定的速度 · 透過數位電壓的高低傳輸。

開始位元	資料位元							停止位元
0	B1	B2	B3	B4	B5	B6	B7	1

- TX 傳遞資料 · RX 接收資料。

Serial Functions

- `Serial.begin(speed)`
- `Serial.begin(speed, config=SERIAL_8N1)`
 - speed: in bits per second (baud) *[type: long]*
 - 鮑(baud)=鮑率(baud rate): 資料傳輸的速率(單位: bps)。
 - For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 28800, 38400, 57600, or **115200**
 - config: sets data, parity, and stop bits. *[default: SERIAL_8N1]*
 - Valid values are: SERIAL_αβγ
 - α: 5~8 (資料位元)
 - β: N, E, O (同位元檢查)
 - γ: 1, 2 (停止位元)

<https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/>

<https://www.analog.com/cn/design-center/landing-pages/002/tech-articles-taiwan/uart-a-hardware-communication-protocol.html>

UART的格式

每一筆傳輸的資料由四部份所組成 · 共分為:

1. 起始位元(Start Bit)
2. 資料內容(Data) · 可以是從 MSB to LSB 或是從 LSB to MSB
 1. MSB: Most Significant Bit
 2. LSB: Least Significant Bit
3. 奇偶同位元檢查碼(Parity Bit)
4. 停止位元(Stop Bit)



小練習

- 傳送資料數字 1 (ASCII:49/ 0b00110001)
- 在 TX 上會看到什麼呢？
- SERIAL_7O1 → 0 0110001 0 1
- SERIAL_7N1 → 0 0110001 1
- SERIAL_7O2 → 0 0110001 0 11
- SERIAL_8E1 → 0 00110001 1 1
- SERIAL_8N2 → 0 00110001 11

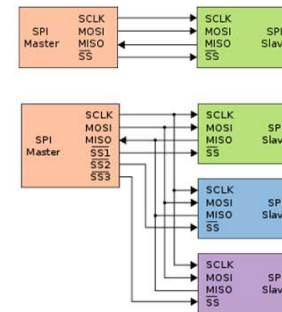
ODD(奇數檢查) 跟 EVEN(偶數檢查)

Lab 11. Reading from Serial port

Use serial port to enter 1-9 from PC to ESP32 to control the brightness of the LED_BUILTIN.

```
Serial.begin(...);
ledcAttach(...);
Serial.print(...);
Serial.println(...);
Serial.available();
Serial.read();
ledcWrite(...);
```

SPI (Serial Peripheral Interface)



- 最早由Motorola公司提出
- SPI是一種4線同步序列資料協定
- SPI 為一主從式架構，通常有一個 Master (主設備) 和一個 (或多個) Slave (從設備)。
- SPI在一對一連接時需要使用四個接點，一對二時要五個，一對三時要六個，即N+3的概念。

<https://magicjacking.pixnet.net/blog/post/164725144>

SPI

接腳名稱及意義

接腳名稱	中文	說明
MOSI	主出從入	Master Output, Slave Input
MISO	主入從出	Master Input, Slave Output
SCLK	時脈訊號	Serial Clock, 由 master 產生並控制
/SS	晶片致能	Slave Select, 由 master 控制。 slave 只有在 /SS 信號為低電位時，才會對 master 的操作指令有反應。

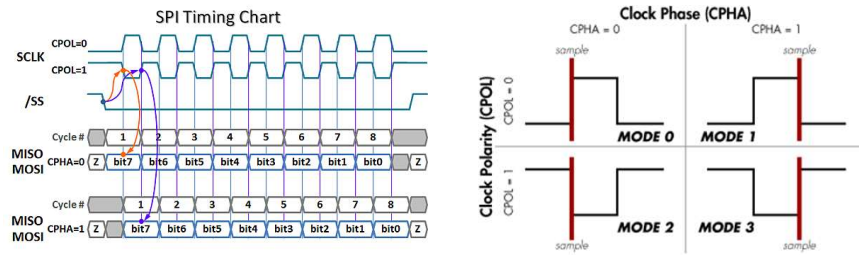
SPI 的接腳有另一套常用的名稱:

- | | |
|--------------------------------|---------------------------------------|
| • SDO : Serial Data Out · 資料輸出 | — MOSI (Master Output, Slave Input) |
| • SDI : Serial Data In · 資料輸入 | — MISO (Master Input, Slave Output) |
| • SCK : 對應 SCLK | — SCLK (Serial Clock) |
| • /CS : 對應 /SS, Chip Select | — SS (Slave Select) |

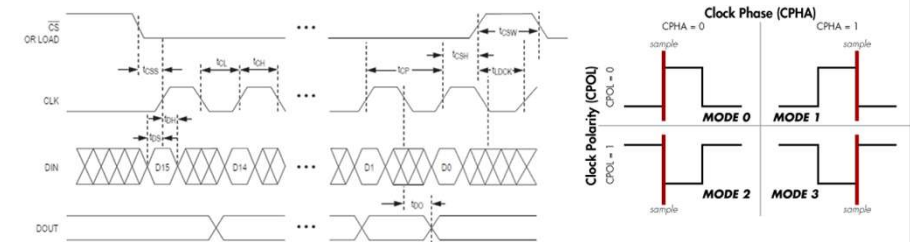
SPI Modes

SPI modes are determined by two parameters: CPOL and CPHA

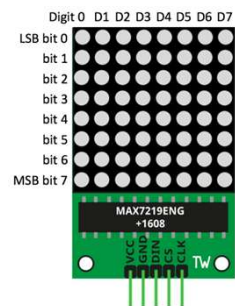
- CPOL (Clock Polarity): 0: Idle Low; 1: Idle High
- CPHA (Clock Phase): 0: odd phase (1,3,5,7,9, ...): 1: even phase (2,4,6,8,10,...)



According to this timing diagram, what is the right SPI mode?



MAX7219 8*8 LED點矩陣顯示模組



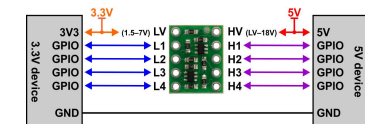
- Slave端
- 8 x 8 規格的矩陣型LED顯示器
- 共陰
- 列掃描 / 行掃描 + 視覺暫留

MAX7219 LED點陣顯示模組 模組參數：

1. 單個模組可以驅動一個8*8共陰點陣
2. 模組尺寸：長5釐米X寬3.2釐米X高1.5釐米
3. 帶4個固定螺絲孔，孔徑3mm
4. 模組帶輸入輸出介面，支援多個模組串聯

Read the data sheet, or?

- Google: [max7219 data sheet](https://www.analog.com/media/en/technical-documentation/data-sheets/MAX7219-MAX7221.pdf)
- <https://www.analog.com/media/en/technical-documentation/data-sheets/MAX7219-MAX7221.pdf>
- Take note on electrical characteristics
 - We often have 3.3V devices (ESP32, RaspberryPi, STM32, Arduino Nano, ...) and 5V devices (e.g. Arduino Uno)
 - Max7219 is actually a 5V device, but fortunately it is very tolerant and can surprisingly work with ESP32, a 3.3V device.
 - Often we need something called "level shifter" with communicate between devices of difference Vcc.
- Pay attention on how to communicate and work with it
 - SPI speed, SPI mode, data format, important registers



Serial Data Format (16 bits)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0

address

data

Register Address Map

Register	D15-D12	A3	A2	A1	A0	Hex code
No operation	X	0	0	0	0	X0
Column 1	X	0	0	0	1	X1
Column 2	X	0	0	1	0	X2
Column 3	X	0	0	1	1	X3
Column 4	X	0	1	0	0	X4
Column 5	X	0	1	0	1	X5
Column 6	X	0	1	1	0	X6
Column 7	X	0	1	1	1	X7
Column 8	X	1	0	0	0	X8
Decode mode	X	1	0	0	1	X9
Intensity	X	1	0	1	0	XA
Scan limit	X	1	0	1	1	XB
Shutdown	X	1	1	0	0	XC
Display test	X	1	1	1	1	XF

<https://www.farnell.com/datasheets/29075.pdf>
Intensity Register Format

Duty Cycle	D7-D4	D3	D2	D1	D0	Hex code
1/32 (min)	Don't care	0	0	0	0	X0h
3/32	Don't care	0	0	0	1	X1h
5/32	Don't care	0	0	1	0	X2h
7/32	Don't care	0	0	1	1	X3h
9/32	Don't care	0	1	0	0	X4h
11/32	Don't care	0	1	0	1	X5h
13/32	Don't care	0	1	1	0	X6h
15/32	Don't care	0	1	1	1	X7h
17/32	Don't care	1	0	0	0	X8h
19/32	Don't care	1	0	0	1	X9h
21/32	Don't care	1	0	1	0	XAh
23/32	Don't care	1	0	1	1	XBh
25/32	Don't care	1	1	0	0	XCh
27/32	Don't care	1	1	0	1	XDh
29/32	Don't care	1	1	1	0	XEh
31/32	Don't care	1	1	1	1	XFh

- 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+, thereby blanking the display.

Shutdown register Format (address XCh)

Mode	Address	D7	D6	D5	D4	D3	D2	D1	D0
Shutdown Mode	XCh	X	X	X	X	X	X	X	0
Normal Operation	XCh	X	X	X	X	X	X	X	1

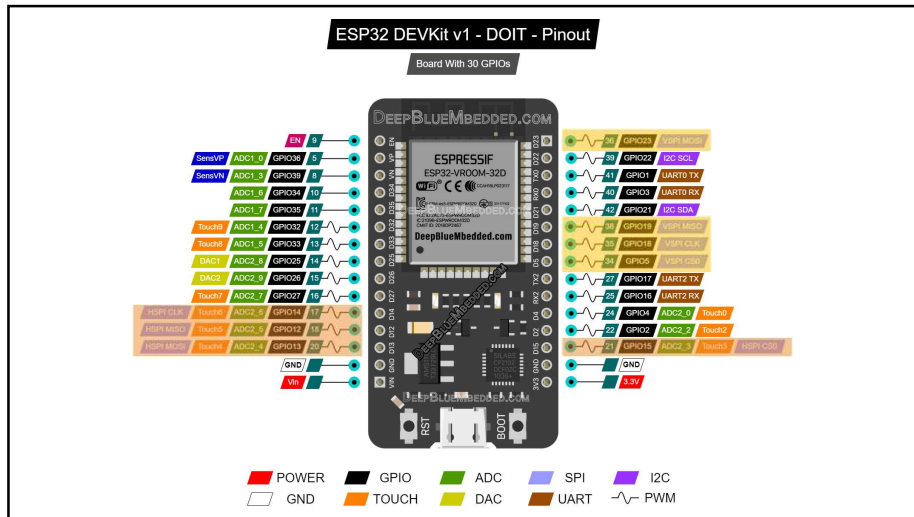
- Decode Mode Register: Setting this register sets no decode operation for each column. Each bit in the register corresponds to one column. A logic 0 enables non decode mode. In non decode mode data bits 0 to 7 correspond to the row lines as outlines in the table below.

Decode Mode register (address 0xX9)

Decode Mode	D7	D6	D5	D4	D3	D2	D1	D0	Hex code
No decode for digits 7-0	0	0	0	0	0	0	0	0	00

Scan Limit Register format (address 0xB)

Scan Limit	D7-D3	D2	D1	D0	Hex code
Column 0	Don't care	0	0	0	X0h
Column 0,1	Don't care	0	0	1	X1h
Column 0,1,2	Don't care	0	1	0	X2h
Column 0,1,2,3	Don't care	0	1	1	X3h
Column 0,1,2,3,4	Don't care	1	0	0	X4h
Column 0,1,2,3,4,5	Don't care	1	0	1	X5h
Column 0,1,2,3,4,5,6	Don't care	1	1	0	X6h
Column 0,1,2,3,4,5,6,7	Don't care	1	1	1	X7h



SPI Settings class

```
class SPISettings {
public:
    SPISettings():_clock(1000000),_bitOrder(SPI_MSBFIRST),_dataMode(SPI_MODE0) {}
    SPISettings(uint32_t clock, uint8_t bitOrder, uint8_t dataMode) :
        _clock(clock), _bitOrder(bitOrder), _dataMode(dataMode) {}

    uint32_t _clock;
    uint8_t _bitOrder;
    uint8_t _dataMode;
};
```

```
#define SPI_MODE0 0      #define SPI_LSBFIRST 0
#define SPI_MODE1 1      #define SPI_MSBFIRST 1
#define SPI_MODE2 2
#define SPI_MODE3 3
```

<https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-spi.h>

SPI class

```
class SPIClass {
    SPIClass(uint8_t spi_bus = HSPI);
    ~SPIClass();
    void begin(int8_t sck = -1, int8_t miso = -1, int8_t mosi = -1, int8_t ss = -1);
    void end();

    void beginTransaction(SPISettings settings);
    void endTransaction(void);
    void transfer(void *data, uint32_t size);
    uint8_t transfer(uint8_t data);
    uint16_t transfer16(uint16_t data);
    uint32_t transfer32(uint32_t data);
    void transferBytes(const uint8_t *data, uint8_t *out, uint32_t size);
};
```

<https://github.com/espressif/arduino-esp32/blob/master/libraries/SPI/src/SPI.h>
<https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-spi.h>

Lab11 – SPI with MAX7219

- Use the “Display Test” Register (0x0f) to a) enter the display test mode, b) delay 250ms, c) leave the display test mode in the loop() function.
- Use the “Shutdown Register” (0x0c) to leave the shutdown mode and enter the normal operation mode.
- Use Digit0 – Digit7 registers (0x01 – 0x08) to display a zebra pattern.
- Need to set the “Scan Limit Register” (0x0b) to 7 to scan all eight columns/digits.
- Use the “Intensity” Register (0x0a) to vary the brightness of the display.
- Create a function plot(x, y, onoff) to turn on/off the specific LED.
- Draw two diagonal lines using the plot function.

Homework

- Draw arrow according to the pose on your joystick (left arrow, right arrow, up arrow, down arrow)
- Use button/switch to control normal/inverse display

```

      *
    ***
  * * *
    *
    *
    *
  
```

```

    *** **
    * * **
    * * *
    * * *
    * * *
    * * *
  
```

- Make the arrow “flies” or animates in its direction

```

      *      *      *      *      *      *      *
    ***      *      *      *      *      *      *
  * * *      *      *      *      *      *      *
    *      *      *      *      *      *      *
    *      *      *      *      *      *      *
    *      *      *      *      *      *      *
  
```