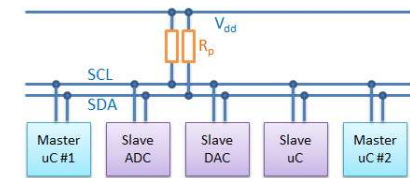


ESP32 DAY4

I2C with DS3231

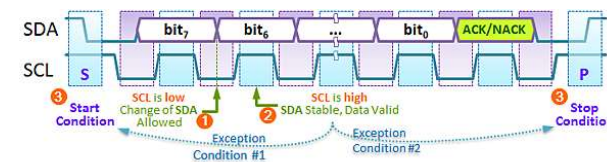
Inter-Integrated Circuit Bus(I2C)

- 積體電路之間的匯流排，可簡寫成I2C、I²C、IIC (Bus)
- 荷蘭飛利浦Philips半導體公司(現為NXP)在1982年所開發，主要是為了讓微控制器或 CPU 以較少的接腳數連接眾多的**低速週邊裝置**之用。
- 有兩條信號線: SDA(傳遞資料)、SCL(傳遞時脈clock)
- 半雙工、同步串列通訊



i²c

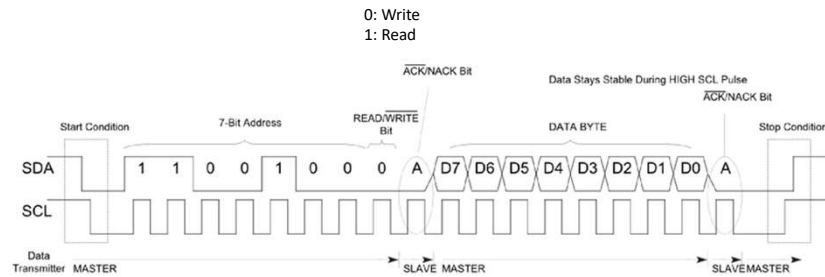
I²C訊號傳遞



1. SCL 為 Low 時，SDA 可以改變資料。如上圖標示 ① 的位置。
2. SCL 為 High 時，SDA 必需保持訊號穩定，不可以改變，以方便對方讀取(栓鎖)資料。如上圖標示 ② 的位置。
3. SCL 為 High 時，如果 SDA 有變動則視為特殊狀況：Start (啟始，SDA 由 High 轉為 Low) 或 Stop (結束，SDA 由 Low 轉為 High)。如上圖二個標示 ③ 的位置。上面三條定義剛好把 SCL 和 SDA 二條訊號線會發生的訊號全部都包括了。

<https://www.ctimes.com.tw/DispArt-tw.asp?k=體感互動/VR/盛群&o=190123182541>

I²C 的 Timing Diagram



<https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html>

DS3231 RTC 高精度時鐘模組 + AT24C32 EEPROM

I2C Message



<https://docs.arduino.cc/learn/communication/wire/>

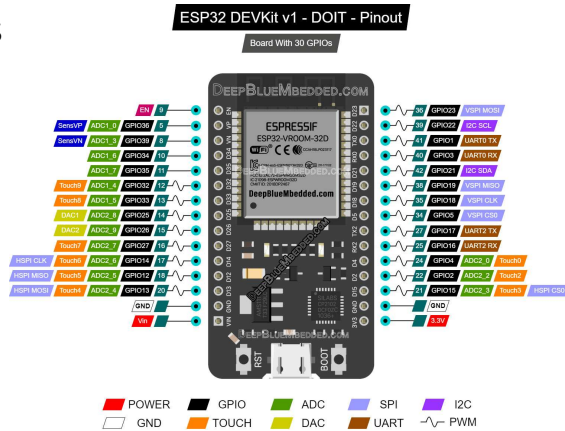
DS3231

- The DS3231 is a low-cost, extremely accurate I2C real-time clock (RTC) with an integrated temperature compensated crystal oscillator (TCXO) and crystal.
- The device incorporates a battery input, and maintains accurate timekeeping when main power to the device is interrupted. The integration of the crystal resonator enhances the long-term accuracy of the device as well as reduces the piece-part count in a manufacturing line.
- The RTC maintains seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year.
- The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Two programmable time-of-day alarms and a programmable square-wave output are provided. Address and data are transferred serially through an I2C bidirectional bus.

I2C addresses

- 0x57: AT24C32
- 0x68: DS3231

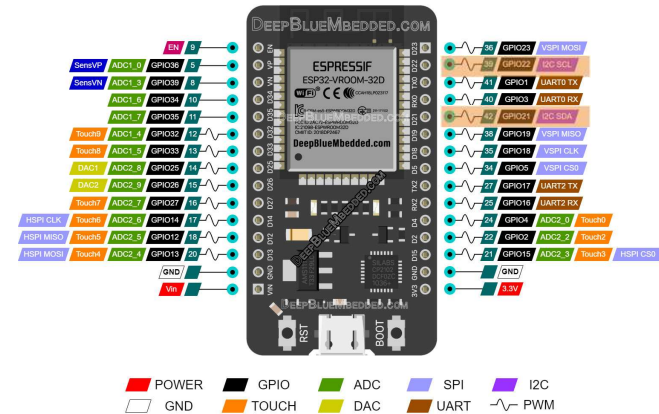
32K
SQW
SCL
SDA
VCC
GND



<https://randomnerdtutorials.com/esp32-i2c-scanner-arduino/>

ESP32 DEVKit v1 - DOIT - Pinout

Board With 30 GPIOs



I2C master (#include <Wire.h>)

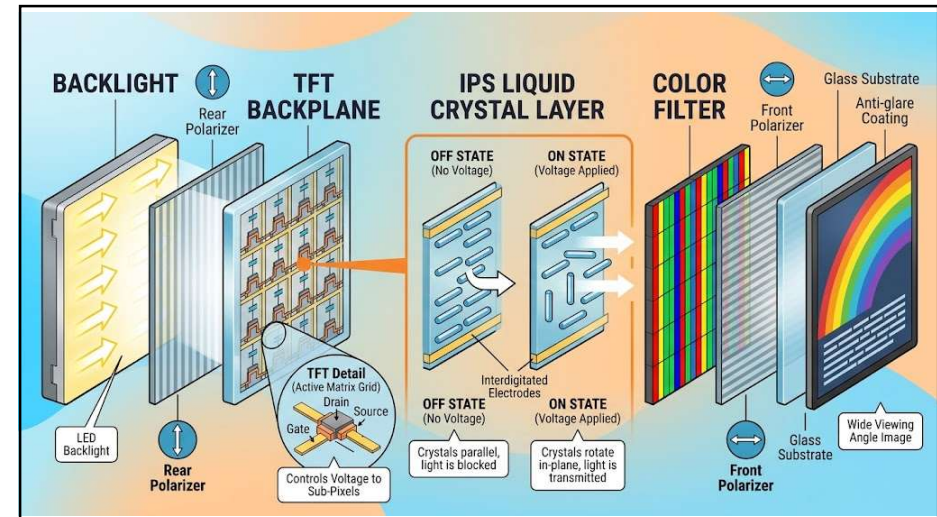
- `bool begin(int sdaPin, int sclPin, uint32_t frequency);`
- `void beginTransmission(uint16_t address);`
- `size_t write(uint8_t);`
- `size_t write(const uint8_t *, size_t);`
- `uint8_t requestFrom(uint16_t address, uint8_t size, bool sendStop);`
- `uint8_t endTransmission(bool sendStop=true);`
- `bool setPins(int sdaPin, int sclPin);`
- `bool setClock(uint32_t frequency);`
- `uint32_t getClock();`
- `void setTimeout(uint16_t timeOutMillis=50);`
- `uint16_t getTimeout();`
- `bool end();`

<https://docs.espressif.com/projects/arduino-esp32/en/latest/api/i2c.html>

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds			Seconds	00-59	
01h	0	10 Minutes			Minutes			Minutes	00-59	
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour			Hours	1-12 + AM/PM 00-23	
03h	0	0	0	0	0	Day		Day	1-7	
04h	0	0	10 Date			Date		Date	01-31	
05h	Century	0	0	10 Month	Month			Month/ Century	01-12 + Century	
06h	10 Year				Year			Year	00-99	
07h	A1M1	10 Seconds			Seconds			Alarm 1 Seconds	00-59	
08h	A1M2	10 Minutes			Minutes			Alarm 1 Minutes	00-59	
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour			Alarm 1 Hours	1-12 + AM/PM 00-23	
0Ah	A1M4	DY/DT	10 Date		Day			Alarm 1 Day	1-7	
					Date			Alarm 1 Date	1-31	
0Bh	A2M2	10 Minutes			Minutes			Alarm 2 Minutes	00-59	
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour			Alarm 2 Hours	1-12 + AM/PM 00-23	
0Dh	A2M4	DY/DT	10 Date		Day			Alarm 2 Day	1-7	
					Date			Alarm 2 Date	1-31	
0Eh	EO5C	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

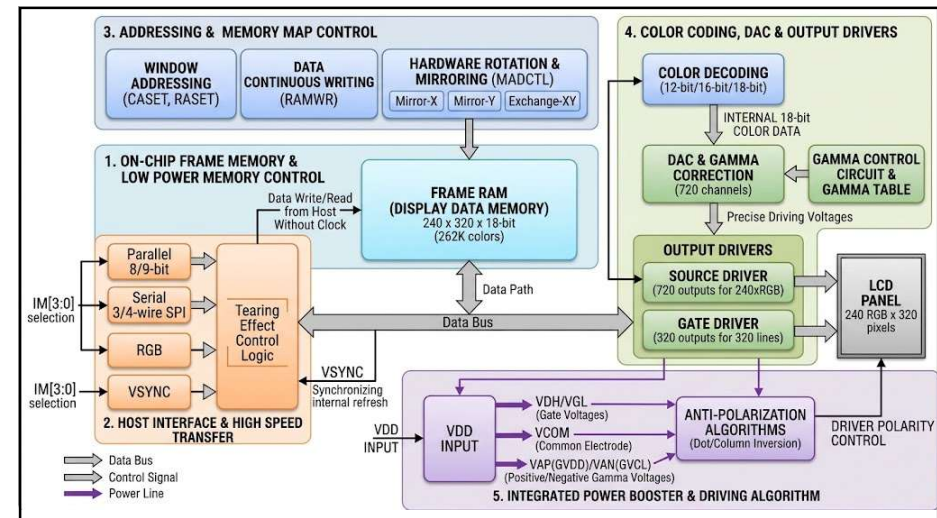
Homework

- Read date, time, and temperature from DS3231 and print clock information repeatedly in the Serial Monitor.
- In the setup function, read the current time, set alarm1 to be 10 seconds into the future, and fire when second matches. The alarm will trigger SWQ to flash LED @ 1Hz interval (yes, please connect a LED between GND & SQW).
- Modify your loop body to read the status register to check if alarm has fired or not. If it is fired, clear it after 3 seconds.



ST7789V3

240x280 1.69" TFT Display



- The write mode of the interface means the micro controller writes commands and data to the LCD driver. In 4-lines serial interface, data packet contains just transmission byte and control bit D/CX is transferred by the D/CX pin. If D/CX is "low", the transmission byte is interpreted as a command byte. If D/CX is "high", the transmission byte is stored in the display data RAM (memory write command), or command register as parameter.
- Any instruction can be sent in any order to the driver. The **MSB is transmitted first**. The serial interface is initialized when CSX is high. In this state, SCLK clock pulse or SDA data have no effect. A **falling edge on CSX enables the serial interface** and indicates the start of data transmission.



7.4.3 Serial Interface Characteristics (4-line serial):

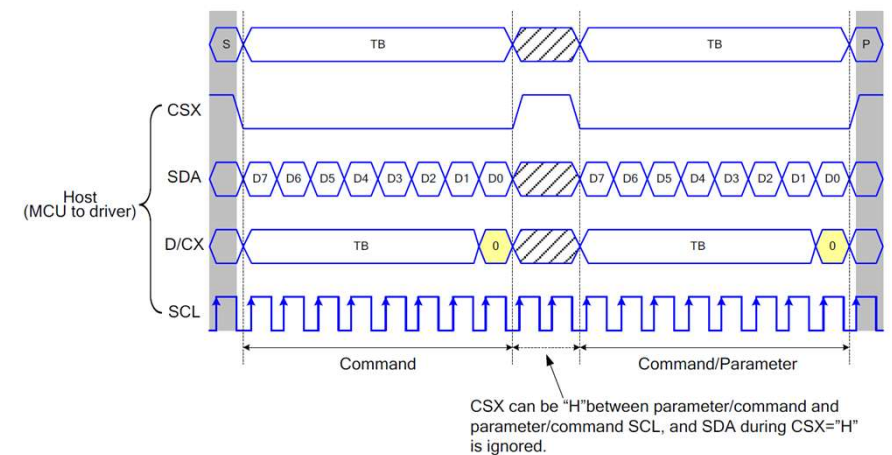
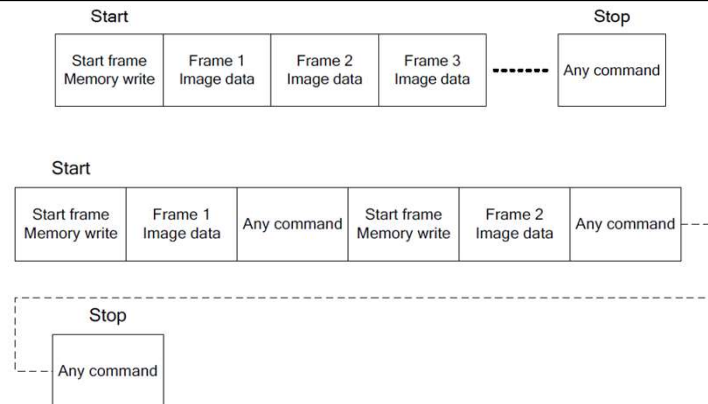
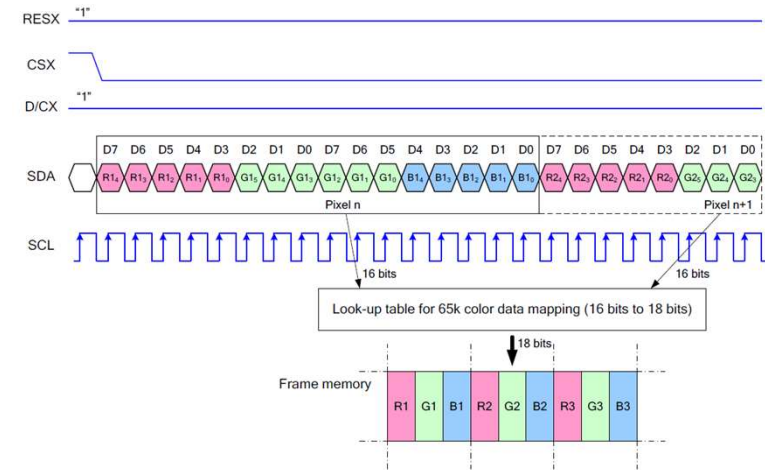


Figure 14 4-line serial interface write protocol (write to register with control bit in transmission)

8.6 Data Transfer Mode

The module has three kinds color modes for transferring data to the display RAM. These are 12-bit color per pixel, **16-bit color per pixel** and 18-bit color per pixel. The data format is described for each interface. Data can be downloaded to the frame memory by 2 methods.

8.7.15 Write data for 16-bit/pixel (RGB-5-6-5-bit input), 65K-Colors, 3Ah="05h"



Note 1: These apply to all data transfer Color modes on both serial and parallel interfaces.

Note 2: The frame memory can contain both odd and even number of pixels for both methods. Only complete pixel data will be stored in the frame memory.

指令 Hex	指令名稱	作用與開發關鍵說明	規格書行為與限制
01h	SWRESET	軟體重設：將所有暫存器重設為出廠預設值。通常在硬體 Reset 釋放後發送。	執行後需要等待 5ms 才能發送新指令；若是在 Sleep In 模式下發送，則必須等待 120ms 才能發送 Sleep Out。
11h	SLPOUT	退出睡眠模式：晶片上電時預設會維持在 Sleep In 狀態（以省電且防止開機瞬間產生雜訊），必須發送此指令開啟內部 DC/DC 電壓與振盪器。	喚醒動作需要時間穩定，發送後至少需要延遲 120ms 才能開始發送後續的顯示控制指令。
3Ah	COLMOD	設定像素格式：設定 MCU 介面與 RGB 介面的像素色彩深度（例如我們在 240x280 螢幕中設定 0x55，代表採用 16-bit/pixel 的 RGB565 色彩格式）。	如果要使用 18-bit 顏色（RGB666），則在此設定 0x66。
36h	MADCTL	資料存取控制：配置畫面記憶體（Frame Memory）的資料寫入方向與顯示控制，包括 X/Y 定址方向、RGB/BGR 色彩順序、以及行列交換（旋轉 90°/180°/270°）。	若在測試時發現紅藍顏色顛倒，通常需要在此微調 BGR 選擇位元。
2Ah	CASET	設定行地址：定義畫面要寫入或讀取的 X 軸起訖邊界（XS 至 XE）。	與 RASET 配合，用來限制每一次繪圖或更新的範圍。
2Bh	RASET	設定列地址：定義畫面要寫入或讀取的 Y 軸起訖邊界（YS 至 YE）。	針對非標準 320 行的螢幕（如 240x280），可能必須在此處加上垂直位移量（Row Offset）以進行對齊。
2Ch	RAMWR	記憶體寫入：通知晶片開始將 MCU 送過來的像素資料填入由 CASET 與 RASET 圈定好的 Frame Memory 視窗中。	發送此指令後，即可在 SPI 匯流排上不間斷地快速灌入像素資料。
29h	DISPON	點亮螢幕：從 Frame Memory 讀取像素資料並正式輸出到 LCD 面板上顯示。	它是整個初始化序列的最後一個指令。

可以請 AI 將所有命令整理成常用、偶爾用、和鮮少使用的命令

指令 Hex	指令名稱	作用與開發關鍵說明
28h	DISPOFF	關閉顯示輸出：關閉螢幕輸出並插入空白畫面（注意：此時 Frame Memory 裡的影像資料並不會被清除）。
10h	SLPIN	進入睡眠模式：關閉內部振盪器與 DC/DC booster 電路，使晶片進入極低功耗狀態（此時 Frame Memory 資料仍會保留）。
21h / 20h	INVON / INVOFF	開啟/關閉顯示反相：控制色彩是否進行負片（Inversion）轉換。
13h	NORON	一般顯示模式開啟：關閉局部（Partial）或閒置（Idle）顯示模式，恢復全螢幕正常顯示。
12h / 30h	PTLON / PTLAR	局部顯示模式與區域設定：僅點亮由 PTLAR 定義的特定幾行（Rows）像素。
35h / 34h	TEON / TEOFF	開啟/關閉 Tearing Effect 訊號：啟用/停用硬體 TE 腳位（垂直同步）的輸出（STE (44h) 用於指定輸出特定 Scanline 的 TE 訊號）。
37h	VSCRSADD	設定垂直捲動 RAM 起始地址：設定 Frame Memory 讀取並顯示在螢幕頂部那一行的硬體偏移值。

Homework – use you ST7789 display module to:

- Write a function dot(int x, int y, uint16_t color) to draw a pixel.
- Write a function fill(int x1, int y1, int x2, int y2, uint16_t color) to draw a solid block.
- Write a function line(int x1, int y1, int x2, int y2, uint16_t color) to draw a line
- Write a function circle(int x, int y, uint16_t color) to draw a circle.
- Write a function inverse(bool onoff) to display normal or inverse display.
- Write functions flipHorizontally(onoff) and flipVertically(onoff) to flip the display horizontally and vertically (using hardware commands, not rearrange pixels in your code!).
- // Write a function text(int x, int y, string text, uint16_t color); to draw a string

指令 Hex	指令名稱	為什麼一般開發者「極少使用」？	備註
00h	NOP	無動作。	僅在開發特殊硬體測試、同步除錯或需要短暫時序延遲時才會用到。
04h, 09h, 0Ah, 08h, 0Ch	RDDID / RDDST / RDDPM / RDDMADCTL / RDDCOLMOD	讀取晶片 ID、基本狀態、電源狀態、MADCTL 與 COLMOD 設定。	市面上 99% 的 4 線 SPI 彩屏模組為了簡化接線， 根本沒有接出 MISO 讀取腳位 （只有單向寫入），因此在實務上根本無法執行讀取動作。
2Eh / 3Eh	RAMRD / RDMEMC	讀取顯示記憶體（Frame Memory）中的像素色彩資料。	原因同上。在一般的單向寫入彩屏專案中，幾乎不會有人將像素資料從螢幕重新讀回微控制器。
38h / 39h	IDMOFF / IDMON	關閉/開啟閒置模式。在 IDMON 開啟時色彩會降為 8 色（RGB 各 1-bit）以降低靜態功耗。	除非是電池要求極度苛刻的電子紙替代方案，否則一般彩色 UI 專案絕對不會開啟此模式。
80h 到 DFh	Command Table 2 (Extended)	包含 RAMCTRL、RGBCTRL、PORCTRL、GCTRL（控制 Gate 開啟電平 VGH 達 +14.97V 等）等。	這些暫存器用於控制晶片底層的極性反相、時脈頻率、升壓振盪頻率與硬體電壓。這在螢幕玻璃出廠時已被原廠直接燒定在 NVM 中， 一般開發者直接帶入原廠給的初始化 Magic Numbers 即可，絕不需自行調整。
E0h / E1h	PVGAMCTRL / NVGAMCTRL	正極性與負極性 Gamma 矯正曲線設定（各高達 16 個參數）。	這是校正液晶透光度與偏色極為專業的曲線表，一般直接沿用原廠提供的預設 Gamma 參數即可。
C7h	CABCCTRL	自適應背光亮度控制（CABC）參數配置（如 PWM 輸出極性與除頻）。	一般背光控制在硬體上多是由 MCU 直控（如使用 LEDC），而非經由 ST7789 晶片輸出 PWM 控制背光，因此一般不常調用。
FAh, FCh, FEh	PROMEN / NVMMSET / PROMACT	內建非揮發性記憶體（NVM）的燒錄模式啟用、編程地址與動作執行指令。	此指令屬於生產線專用。 一旦燒錄錯誤可能導致晶片鎖死或損壞，一般開發者在撰寫韌體時必須嚴禁調用。