

# 外部中斷與計時控制實習

## 本章單元

- 外部中斷控制實習
- 睡眠省電模式
- 外部中斷應用實習(人體紅外線感測器、矩陣式按鍵、旋轉編碼器)
- 計時器控制實習
- 計時器應用實習(音樂、PWM、馬達控制、WDT、RTC)

微處理機輸入信號的處理方式有三種：

◎程式 I/O：在程式迴圈中必須不斷的用指令去偵測輸入腳的狀態，在第三章中就是這種方式來處理。

優點：程式簡單、容易維護。

缺點：會浪費 CPU 的時間，且程式必須執行到輸入指令時，才會偵測輸入端，若輸入端的信號來的太過快速，CPU 將無法即時偵測到，致使輸入信號漏失。

◎中斷(Interrupt)I/O：致能中斷後，即可不用理會輸入端。有中斷輸入信號時，會立即通知 CPU 停止目前的工作，去執行中斷服務函數(ISR)。執行完後，回原主程式繼續執行。中斷應用於需要最優先的動作處理，本章將詳細介紹。

優點：不用去偵測輸入端，不會浪費 CPU 時間，且能偵測到快速的輸入信號。

缺點：若應用於大量的資料傳輸時會疲於奔命，較不適合。

◎直接記憶存取(DMA：Direct Memory Access)：在開發板內 MCU 的周邊設備如 UART、TIME、SPI、ADC、I<sup>2</sup>C 及 USB 等工作時，可透過 DMA 控制器來通知 CPU 持住(hold)，此時 CPU 會切除對外的 BUS 線，停止對周邊設備與記憶體溝通。而由 DMA 控制器取代 CPU 來控制 BUS 線，令周邊設備直接和 RAM 存取資料。它適合用於整批大量資料的存取，MG32x02z 內含 DMA 周邊設備，但在 Arduino IDE 無法提供 DMA 功能。

ARM®Cortex®-M0 內含嵌套向量中斷控制器(NVIC: Nested Vectored Interrupt Controller)，提供多個中斷向量及多層中斷優先等級，具有快速中斷響應，可提高處理器的反應能力與效能，大量減少中斷延時時間。其中 MG32x02z 系列如下圖及下表所示：

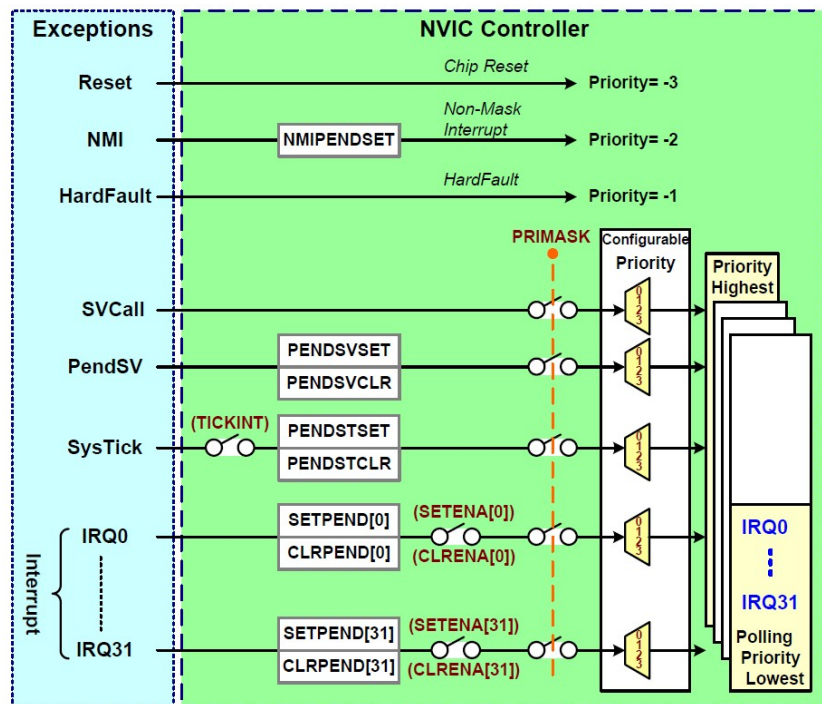


圖 5-1 MG32x02z 系列的 NVIC 控制

表 5-1 NVIC 異常向量

| NVIC          |          |                |              |              |                    | Comment                         |
|---------------|----------|----------------|--------------|--------------|--------------------|---------------------------------|
| Exception No. | IRQ No.  | Interrupt Name | Priority     | Activation   | Exception handlers |                                 |
| (1) 0         | -        | Initial        | -            |              |                    |                                 |
| (2) 1         | -        | Reset          | -3           | Asynchronous |                    | Reset exception                 |
| (3) 2         | -14      | NMI            | -2           | Asynchronous | System handlers    | Non Maskable Interrupt          |
| 3             | -13      | HardFault      | -1           | Synchronous  | Fault handler      | Cortex-M0 Hard Fault Interrupt  |
| 4~10          | -        | Reserved       | -            |              |                    |                                 |
| 11            | -5       | SVC            | Configurable | Synchronous  | System handlers    | Cortex-M0 SV Call Interrupt     |
| 12~13         | -        | Reserved       | -            |              |                    |                                 |
| 14            | -2       | PendSV         | Configurable | Asynchronous | System handlers    | Cortex-M0 Pend SV Interrupt     |
| (4)15         | -1       | SysTick        | Configurable | Asynchronous | System handlers    | Cortex-M0 System Tick Interrupt |
| 16~47         | 0~31 (5) | -              | Configurable | Asynchronous | ISRs               | Peripheral Interrupts           |

Configurable : Programmable priority level 0~3

表 5-2 異常中斷向量

| 異常編號  | IRQ 編號 | 特殊異常狀況                         | 向量位址          |
|-------|--------|--------------------------------|---------------|
| 0     |        | initial_sp(初始化堆疊)              | 0x0000_0000   |
| 1     |        | Reset(重置)或 IEC60730_MANAGER    | 0x0000_0004   |
| 2     | -14    | NMI(不可遮蔽中斷)                    | 0x0000_0008   |
| 3     | -13    | HardFault(處理器用)                | 0x0000_000C   |
| 4~10  | -5~-12 | 保留                             | 0x0010~0x0028 |
| 11    | -5     | SVCall(supervisor call)        | 0x0000_002C   |
| 12~13 | -3~-4  | 保留                             | 0x0030~0x0034 |
| 14    | -2     | PendSV                         | 0x0000_0038   |
| 15    | -1     | SysTick(系統節拍計時器)               | 0x0000_003C   |
| 16    | 0      | WWDT(視窗看門狗計時)                  | 0x0000_0040   |
| 17    | 1      | SYS(系統整體中斷)                    | 0x0000_0044   |
| 18    | 2      | 保留                             | 0x0000_0048   |
| 19    | 3      | EXIC EXINT0-PA 外部中斷            | 0x0000_004C   |
| 20    | 4      | EXIC EXINT1-PB 外部中斷            | 0x0000_0050   |
| 21    | 5      | EXIC EXINT2-PC 外部中斷            | 0x0000_0054   |
| 22    | 6      | EXIC EXINT3-PD 及 PE(限 128)外部中斷 | 0x0000_0058   |
| 23~46 | 7~30   | (中間省略)                         | 0x005C~0x00B8 |
| 47    | 31     | APX                            | 0x0000_00BC   |



## 5-1 外部中斷控制實習

Cortex®-M0 的嵌套式向量中斷控制器(NVIC)除了系統中斷外，還提供 32 個周邊設備中斷向量(IRQ0~31)、四層中斷優先等級及快速中斷響應，可提高處理器的反應速度與效能。。

### 5-1.1 外部中斷控制實習

MG32F02U128AD64 可將每支 GPIO 接腳作為外部中斷，特性如下：

◎ 所有的 GPIO 接腳(PA~PD)均可作為外部中斷輸入。

◎ 外部中斷控制函數式：

```
attachInterrupt(pin, ISR, mode) //設定中斷接腳、中斷服務函數及觸發輸入模式
noInterrupts();    //停止外部中斷
interrupts();      //恢復外部中斷
```

◎ 可選擇四種外部中斷觸發輸入模式(mode)如下：

LOW：當引腳為低電位時觸發中斷服務程式。

CHANGE：當引腳電位發生變化時觸發中斷服務程式。

RISING：正邊緣當引腳電平由低電位變為高電位時觸發中斷服務程式。

FALLING：負邊緣當引腳電平由高電位變為低電位時觸發中斷服務程式。

◎ 內含喚醒中斷控制(WIC: Wakeup Interrupt Controller)，可在 MCU 進入省電模式後，藉由中斷來進行喚醒事件控制。

1. 範例 EINT1：致能外部中斷後，按鍵(USER)產生中斷，令序列監控窗顯示中斷次數。
2. 範例 EINT2：致能外部中斷後，按 KEY1 產生中斷計數遞加，按 KEY2 產生

中斷計數遞減。

3. 範例 EINT3：令 LED 輸出紅黃綠燈變化，當按 USER 鍵產生中斷令黃燈互閃。

## 5-1.2 省電模式控制實習

MG32x02z 內含系統電源控制器(PW：Power controller)，操作電源電壓為 1.8V~5.5V，特性如下所示：

- ◎ 內含電源管理控制來進行電源省電及喚醒(wakeup)控制。
  - ◎ 提供三組電源工作模式：正常(Normal)、睡眠(SLEEP)及停止(STOP)。
  - ◎ 提供由睡眠(SLEEP)及停止(STOP)省電模式的喚醒(wakeup)功能。
1. 電源工作模式：有 ON、SLEEP 及 STOP 電源工作模式，如下表(a)(b)所示：

表 5-3(a) 電源工作模式

| Power Operation Mode   | Normal              | Power-Down  |   |
|------------------------|---------------------|---|---|
| Internal Device        | ON                  | SLEEP   | STOP  |
| Current Consumption    | full                | low   | very low  |
| ARM32 Cortex-M0        | Normal mode         | Sleep mode  | Deep Sleep mode   |
| Core LDO               | full/low power (*1) | full/low power (*1)                               | full/low power (*2)   |
| CPU Clock              | run                 | stop  | stop  |
| System Clock/PLL/IHRCO | run                 | run   | stop  |
| XOSC/ILRCO             | normal              | normal  | Hardware Configure  |
| IO Pins                | normal              | normal  | normal  |
| Peripheral Modules     | normal              | configurable                                      | disabled except configurable  |
| Enter Power Mode       |                     | Execute WFI/WFE instruction command + SLEEPDEEP=0 | Execute WFI/WFE instruction command + SLEEPDEEP=1   |
| Wake up Events         |                     | All Interrupts or wakeup events                   | LVR, BOD0/BOD1, CMP,RTC(from CK_LS,CK_UT), EXINT External interrupt pins (only level), IWDG(CK_ILRCO), I2C(slave address detection) |

表 5-3(b) 電源工作模式-內部時脈控制

| Internal Device              | ON              | SLEEP                    | STOP  |
|------------------------------|-----------------|--------------------------|---|
| ARM32 Cortex-M0              | on              | off                      | off   |
| System Core Clock (AHB, APB) | on              | on                       | off   |
| SWD/Debug                    | CSC Reg setting | CSC Reg presetting       | CSC Reg presetting                                  |
| Ext INT pins                 | CSC Reg setting | interrupt enable setting | interrupt enable setting                            |
| Analog Comparator            | CSC Reg setting | CSC Reg presetting       | CSC Reg presetting                                  |
| RTC                          | CSC Reg setting | CSC Reg presetting       | CSC Reg presetting                                  |
| IWDT                         | CSC Reg setting | CSC Reg presetting       | CSC Reg presetting                                  |
| I2Cx                         | CSC Reg setting | CSC Reg presetting       | off (slave address detect using external SCL clock) |
| Other Modules                | CSC Reg setting | CSC Reg presetting       | off   |

- (1) 正常(ON)模式：CPU 可以全速執行，所有周邊設備模組都可以正常工作，這些模組可以各自啓用或禁用以節省功耗。
- (2) 睡眠(SLEEP)模式：只有 CPU 停止工作並進入睡眠模式，所有周邊設備模組都可以可設定繼續執行或休眠。在這種模式下，晶片可以被中斷或事件來喚醒。
- (3) 停止(STOP)模式：可使用較低的  $V_{DD}$  來維持 STOP 模式運作，提供最低的功耗。此時 CPU 是進入 CPU 深度睡眠模式，只有少數的周邊設備模組可以配置在 STOP 模式下操作，包括 IWDT、RTC、CMP 模組以及穩壓電路(LDO)、LVR、BOD0、BOD1、BOD2 等。只有 GPIO 外部輸入和一些事件可以喚醒，恢復正常 ON 模式工作。

2. 電源工作模式的電氣特性，如下表(a)(b)所示：

表 5-4(a) 電源工作電氣特性(1)

| Symbol            | Parameter  | Conditions  | Limits |      |     | Unit |
|-------------------|--|---|--------|------|-----|------|
|                   |  |   | Min    | Typ  | Max |      |
|                   | Current Consumption                                  |   |        |      |     |      |
| I <sub>OP1</sub>  | ON(normal) mode operating current                    | TL1 (APB=AHB=32KHz) dhrystone                           |        | 0.10 |     | mA   |
| I <sub>OP2</sub>  |  | TL2 (APB=AHB=12MHz) dhrystone                           |        | 3.5  |     | mA   |
| I <sub>OP3</sub>  |  | TL3 (APB=AHB=24MHz) dhrystone + IP                      |        | 7.9  |     | mA   |
| I <sub>OP4</sub>  |  | TL4 (APB=AHB=24MHz - XTAL) dhrystone + IP               |        | 8.7  |     | mA   |
| I <sub>OP5</sub>  |  | TL5 (APB=AHB=24MHz - EXTCK) dhrystone + IP              |        | 7.7  |     | mA   |
| I <sub>OP6</sub>  |  | TL6 (APB=AHB=48MHz) dhrystone + all IP                  |        | 22.0 |     | mA   |
| I <sub>SLP0</sub> | SLEEP mode operating current (IWDT Enable)           | SL0 (ILRCO on: IWDT Disable, APB=AHB=32KHz)             |        | 108  |     | uA   |
| I <sub>SLP1</sub> |  | SL1 (IHRCO on: APB=6MHz,AHB=3MHz)                       |        | 872  |     | uA   |
| I <sub>SLP2</sub> |  | SL2 (IHRCO on: APB=AHB=12MHz)                           |        | 1198 |     | uA   |
| I <sub>SLP3</sub> |  | SL3 (XTAL=12MHz: APB/AHB=6MHz/3MHz)                     |        | 1690 |     | uA   |
| I <sub>SLP4</sub> |  | SL4 (ILRCO on: APB=AHB=32KHz) with Low Power SLEEP mode |        | 39   |     | uA   |
| I <sub>STP0</sub> | STOP mode operating current (LVR/BOD0/BOD1 Disabled) | ST0 (ILRCO off)   |        | 1.67 |     | uA   |
| I <sub>STP1</sub> |  | ST1 (IWDT Enable, ILRCO=32KHz)                          |        | 4.10 |     | uA   |
| I <sub>STP2</sub> |  | ST2 (RTC Enable, ILRCO=32KHz)                           |        | 4.00 |     | uA   |

表 5-4(b) 電源工作電氣特性(2)

| Symbol                | Parameter                        | Conditions         | Limits   |      |          | Unit |
|-----------------------|----------------------------------|--------------------|----------|------|----------|------|
|                       |                                  |                    | Min      | Typ  | Max      |      |
| BOD Characteristics   |                                  |                    |          |      |          |      |
| V <sub>LVR</sub>      | LVR detection level (VR0)        | TA = -40℃ to +105℃ |          | 1.28 |          | Volt |
| V <sub>BOD0</sub>     | BOD0 detection level (VR0)       | TA = -40℃ to +105℃ | 1.40     |      | 1.45     | Volt |
| I <sub>BOD0+LVR</sub> | BOD0 and LVR Current Consumption | TA = 25℃           |          |      | 3.5      | uA   |
| V <sub>BOD10</sub>    | BOD1 detection level for 2.0V    | TA = -40℃ to +105℃ | 1.8(*1)  | 2.0  | 2.2(*1)  | Volt |
| V <sub>BOD11</sub>    | BOD1 detection level for 2.4V    | TA = -40℃ to +105℃ | 2.22(*1) | 2.4  | 2.62(*1) | Volt |
| V <sub>BOD12</sub>    | BOD1 detection level for 3.7V    | TA = -40℃ to +105℃ | 3.50     | 3.7  | 3.90     | Volt |
| V <sub>BOD13</sub>    | BOD1 detection level for 4.2V    | TA = -40℃ to +105℃ | 3.89(*1) | 4.2  | 4.59(*1) | Volt |
| I <sub>BOD1</sub>     | BOD1 Current Consumption         | TA = 25℃           |          |      | 9.0      | uA   |
| V <sub>BOD2</sub>     | BOD2 detection level for 1.7V    | TA = -40℃ to +105℃ | 1.65     | 1.70 | 1.75     | Volt |
| I <sub>BOD2</sub>     | BOD2 Current Consumption         | TA = 25℃           |          |      | 9.0      | uA   |
| Operating Condition   |                                  |                    |          |      |          |      |
| V <sub>PSR</sub>      | Power-on Slop Rate               | TA = -40℃ to +105℃ | 0.05     |      |          | V/ms |
| V <sub>OP1</sub>      | CPU Operating Speed 0~48MHz      | TA = -40℃ to +105℃ | 2.7      |      | 5.5      | Volt |
| V <sub>OP2</sub>      | CPU Operating Speed 0~12MHz      | TA = -40℃ to +105℃ | 1.8      |      | 5.5      | Volt |

3. 電源管理控制進入睡眠模式後，可藉由 GPIO 腳及其它中斷源來喚醒，可使用的函數及範例如下：

```
attachInterrupt(USER,isr, FALLING) 致能外部中斷腳、中斷函數及負緣觸發
LowPower.sleep(SLEEP_FOREVER) 進入睡眠省電模式
LowPower.idle(SLEEP_8S) 進入低電源閒置省電模式 8 秒
LowPower.standby(SLEEP_8S) 進低電源入待機省電模式 8 秒
LowPower.powerStandby(SLEEP_FOREVER) 進入電源待機省電模式
LowPower.powerStandby(SLEEP_FOREVER, ADC_OFF, USB_OFF)
LowPower.powerDown(SLEEP_FOREVER) 進入電源下降省電模式
LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF, USB_OFF)
LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF, USB_OFF, RTC_OFF)
LowPower.longPowerDown(SLEEP_FOREVER) 進入長電源下降省電模式
```

- (1) 範例 SLEEP1：LED 閃 5 次後進入睡眠模式停止動作，定時喚醒或永遠

不喚醒。

- (2) 範例 SLEEP2：進入省電模式停止動作，按鍵(USER)產生中斷喚醒令  
LED 閃 5 次後從開始

### 5-1.3 人體紅外線感測器實習

人體紅外線感測器可以用來感測人體接近的距離，可分兩種，如下：

- ◎ 主動式：感測器會發射紅外線，如小便斗上的感測器，人體接近時會自動沖水。
- ◎ 被動式：感測器不會發射紅外線，只感應人體紅外線的接近來產生反應，即本章所介紹的人體紅外線感測器。

人體紅外線感測器的感測距離最遠大約可達 6 公尺，角度可達到 110 度左右，它的背面有兩個可調電阻  $S_x$ (調靈敏度)及  $T_x$ (調延遲時間)，如下圖所示：

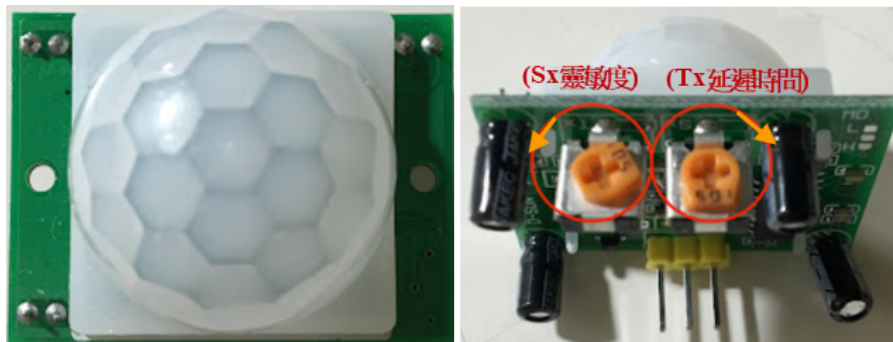


圖 7- 人體紅外線感測器模組外型圖

實驗時，因為比較靠近人體紅外線感測器，建議將  $S_x$ (調靈敏度)順時鐘方向調到最低及  $T_x$ (調延遲時間)反時鐘方向調到最高；再微調到適當的距離。

人體紅外線感測器模組有三支腳(VCC、OUT、GND)，其中 VCC 使用 5V 工作，而人體靠近時會令 OUT 腳會輸出高電位。

1. 範例 IR1：使用人體紅外線感測器，在序列監控視窗顯示人體是否靠近。
2. 範例 IR2：使用人體紅外線感測器，以外部中斷喚醒方式，在 LED 顯示人體是否靠近。

### 5-1.4 矩陣式按鍵控制實習

矩陣式按鍵分成列(COL1~4)及行(ROW1~4)，實習電路如下圖所示：

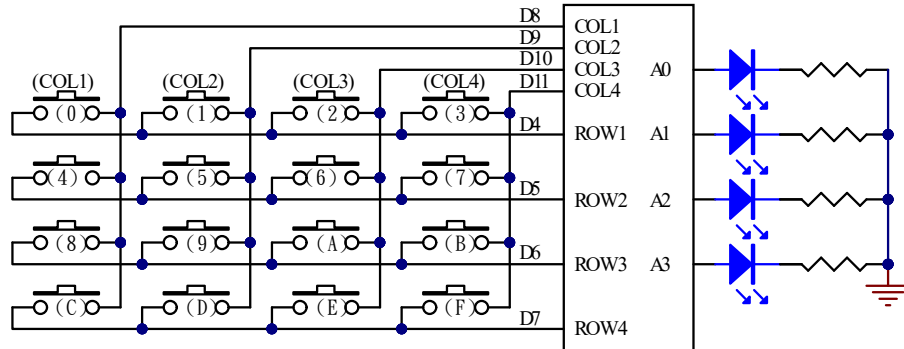


圖 5-2 矩陣式按鍵實習電路

1. 鍵盤掃描控制，如下：

- (1) 偵測鍵盤是否按鍵：用“0”重覆掃描 ROW1→2→3→4，平時 COL1~4(PB0~3)內含有提升電阻接  $V_{DD}$ ，使無按鍵時 COL1~4 均為“1”，如下圖所示：

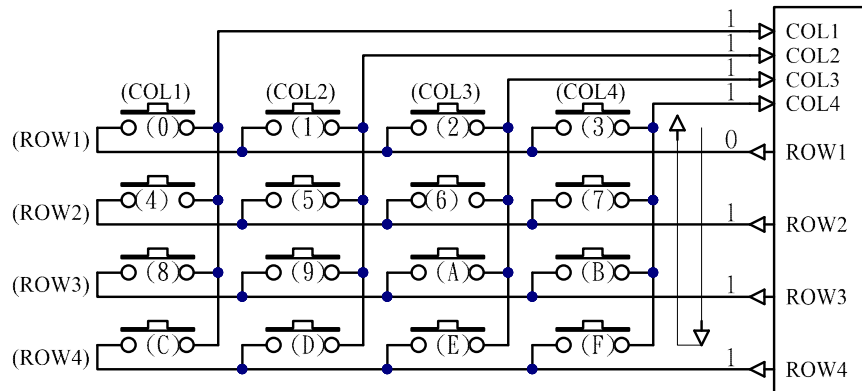


圖 5-3 4\*4 的鍵盤電路圖未按鍵時的動作



- (2) 當有按鍵時會令 COL1~4 其中一個為“0”。如按下 5 鍵，則當掃描輸出到 ROW2=0 時，同時也令輸入 COL2=0。如下圖所示：

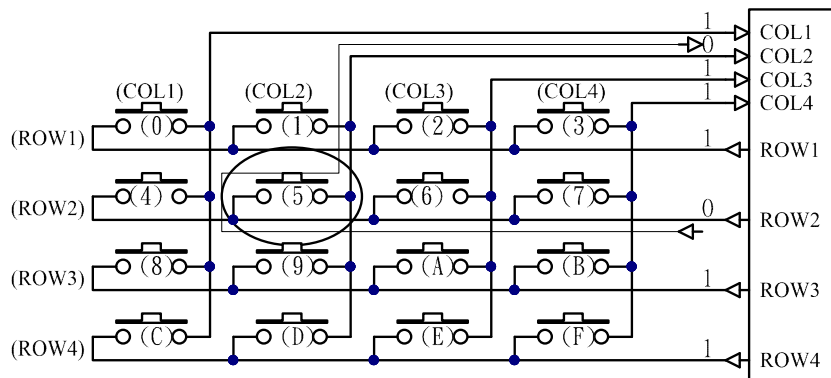


圖 5-4 4\*4 的鍵盤電路的動作

- (1) 由 ROW1~4 掃描輸出及 COL1~4 按鍵輸入的位置，可產生相對應按鍵的數碼，此時以電路加以掃描，即可讀取按鍵的資料。如下表所示：

表 5-5 相對應按鍵的數碼

| 按<br>鍵 | 掃描輸出        | 按鍵輸入        | 按<br>鍵 | 掃描輸出        | 按鍵輸入        |
|--------|-------------|-------------|--------|-------------|-------------|
|        | ROW 4 3 2 1 | COL 4 3 2 1 |        | ROW 4 3 2 1 | COL 4 3 2 1 |
| 0      | 1 1 1 0     | 1 1 1 0     | 8      | 1 0 1 1     | 1 1 1 0     |
| 1      | 1 1 1 0     | 1 1 0 1     | 9      | 1 0 1 1     | 1 1 0 1     |
| 2      | 1 1 1 0     | 1 0 1 1     | A      | 1 0 1 1     | 1 0 1 1     |
| 3      | 1 1 1 0     | 0 1 1 1     | B      | 1 0 1 1     | 0 1 1 1     |
| 4      | 1 1 0 1     | 1 1 1 0     | C      | 0 1 1 1     | 1 1 1 0     |
| 5      | 1 1 0 1     | 1 1 0 1     | D      | 0 1 1 1     | 1 1 0 1     |
| 6      | 1 1 0 1     | 1 0 1 1     | E      | 0 1 1 1     | 1 0 1 1     |
| 7      | 1 1 0 1     | 0 1 1 1     | F      | 0 1 1 1     | 0 1 1 1     |

- (4) 去除按鍵機械彈跳：偵測到鍵盤有按鍵後，必須使用軟體來延時，以避開這一段機械彈跳的時間。
- (5) 讀取鍵盤電路方法有二種：

- (a) 輪詢法：ROW1~4 用 “0” 去掃描，未按鍵時已知 COL1~4=1111。在每一固定時間去讀取 COL1~4，來判斷是否有按鍵。當 COL1~4 其中一個 bit=0 時，表示有按鍵，再去讀取 COL1~4 及 ROW1~4 資料判斷是按那一個鍵。
- (b) 中斷法：將 COL1~4 一起送到 GPIO 外部中斷腳。令 ROW1~4=0000，當其中一個有按鍵時會令=0，而產生負緣觸發輸入外部中斷腳。同時立即執行中斷函數式掃描是按那一個鍵，來輸出按鍵資料。
- (6) 偵測鍵盤是否未放開鍵：檢查 COL1~4 輸入，若其中有一個為 0 時，表示鍵盤未放開鍵必須再等待。當 COL1~4=1111 時，表示鍵盤均已放開，可以往下執行，避免重覆產生數碼。

2. 鍵盤掃描實習範例：實習電路如下圖(a)(b)所示：

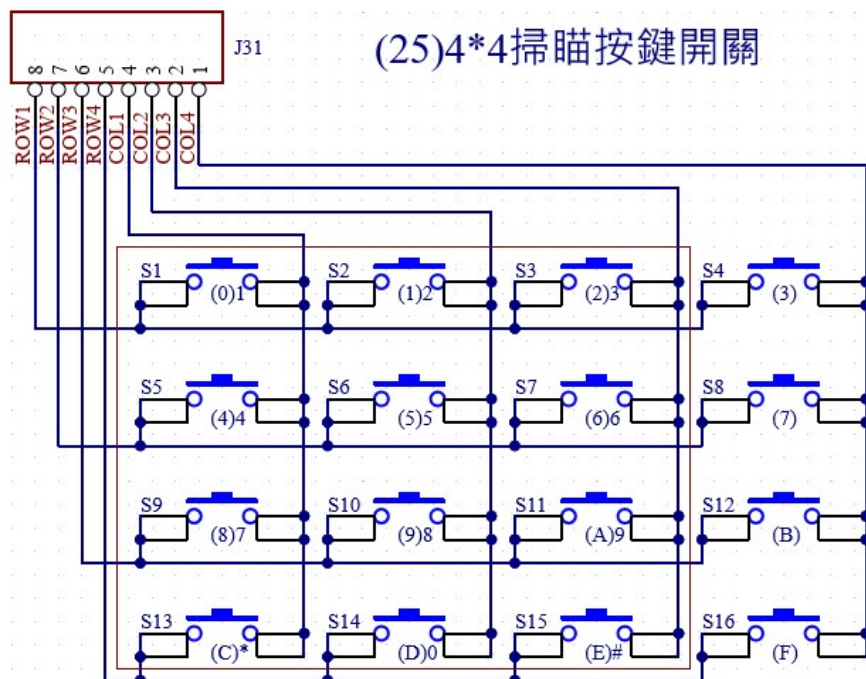


圖 5-5(a) 4\*4 的鍵盤電路的電路

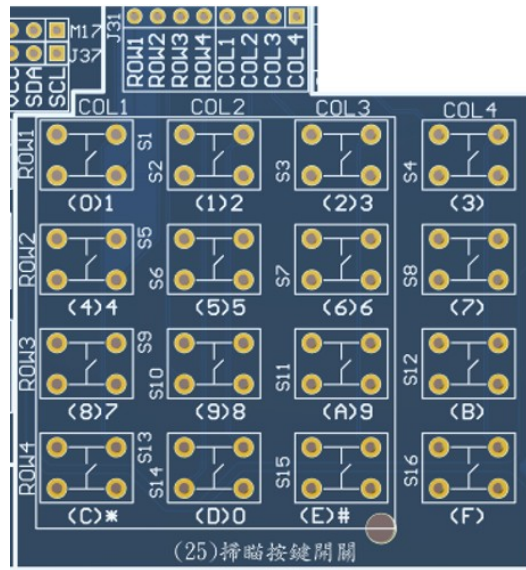


圖 5-5(b) 4\*4 的鍵盤電路的外型

- (1) 範例 KEY4x4\_1：按鍵掃描輸入 0~F，在 LED 顯示按鍵數值。
- (2) 範例 KEY4x4\_2：按鍵掃描輸入 0~F，在 LED 顯示按鍵數值。
- (3) 範例 KEY4x4\_3：系統進入睡眠省電模式，等待按鍵產生外部中斷喚醒，，在 LED 顯示按鍵數值。
- (4) 範例 keypad\_1：ROW0~3 掃描輸出，COL0~3 按鍵輸入，在串列監控視窗顯示按鍵資料。

### 5-1.5 旋轉編碼器實習

旋轉編碼器(rotary)普遍應用於音響及其他控制場合，當用手旋轉時會產生兩個相差 90 度的 A、B 相脈波，且在正轉及反轉會有差別。如下圖(a)(b)所示：

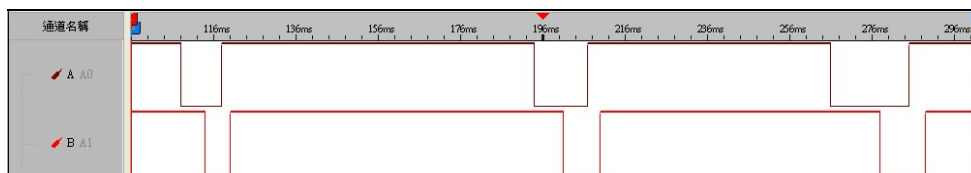


圖 5-6(a) 旋轉編碼器-正轉輸出波形

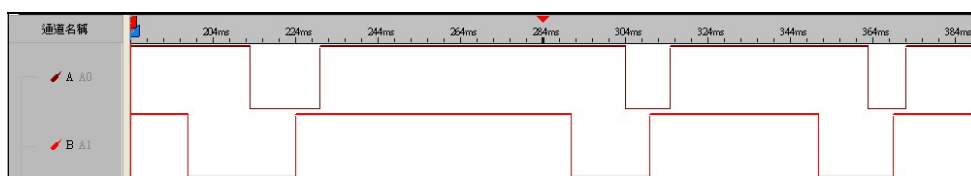


圖 5-6(b) 旋轉編碼器-反轉輸出波形

但旋轉編碼器(rotary)本身為機械式開關，旋轉不順時會有機械彈跳現象。如下圖(c)所示：

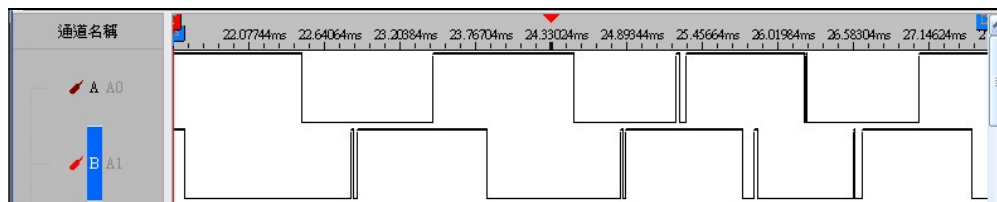


圖 5-6(c) 旋轉編碼器-機械彈跳波形

可用 GPIO 偵測 A、B 相脈波的電位來控制計數器的遞加或遞減，由 MCU 在 J13(PHA、PHB)輸入 A、B 相脈波，來讀取旋轉編碼器(RO1)的旋轉資料，如下圖所示：

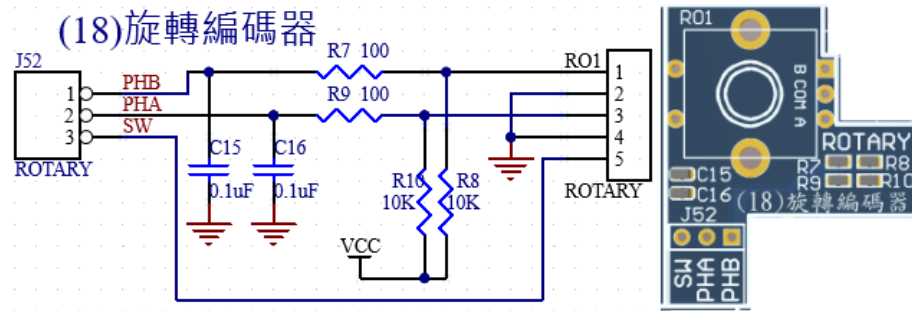


圖 5-7 旋轉編碼器實習電路及外型

A 相有負緣觸發輸入時，再檢查 B 相的信號電位，即可知道旋轉的方向，再控制計數器的上下數計數。如下圖所示：

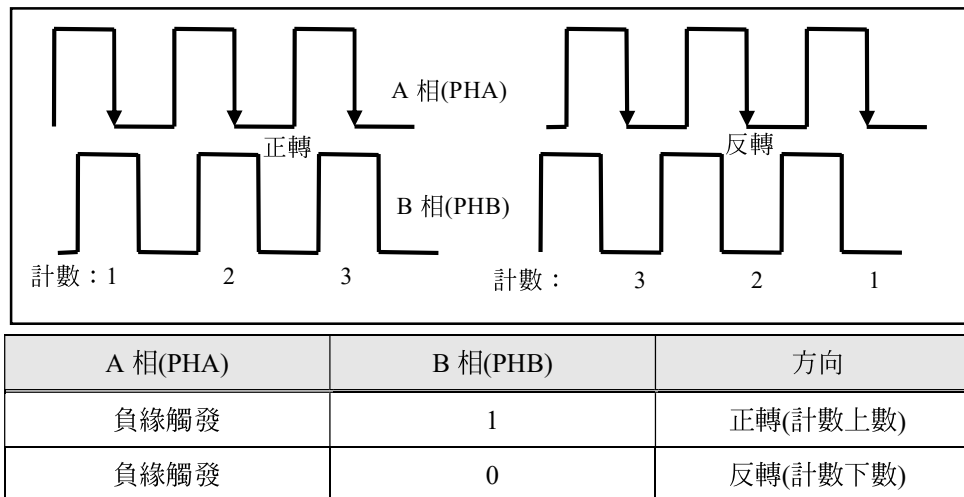


圖 5-8 旋轉編碼器動作

- 範例 ROTARY1：令旋轉編碼器(ROTARY)動作，在串列監控視顯示正/反轉計數值。執行結果，以邏輯分析儀量測，如下圖(a)(b)所示：

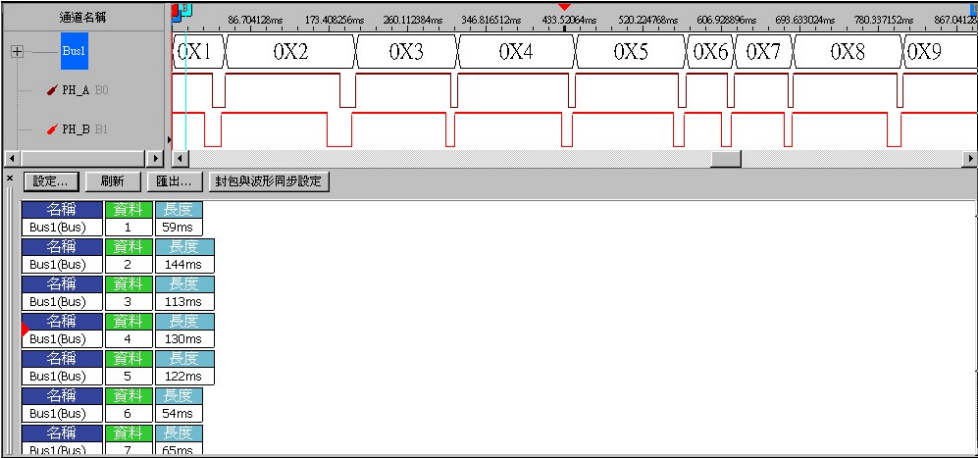


圖 5-9(a) 旋轉編碼器-順時針正轉動作測試

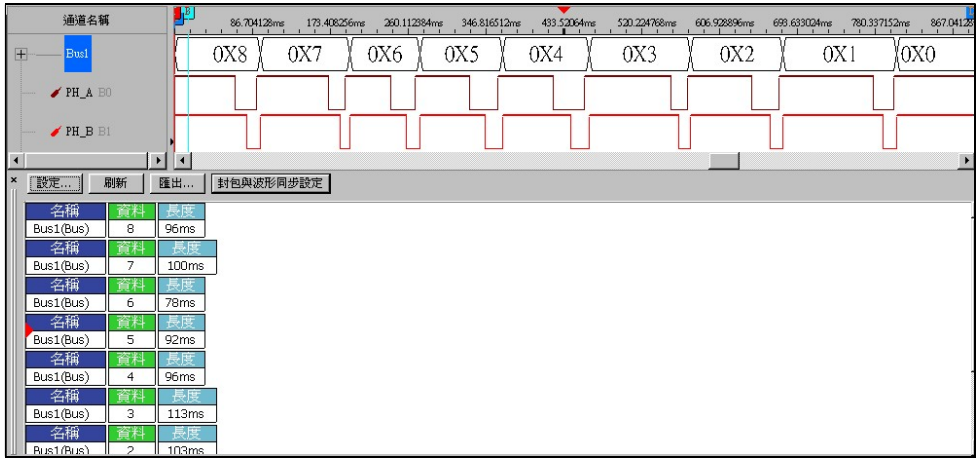


圖 5-9(b) 旋轉編碼器-反時針反轉動作測試

## 5-2 計時控制與應用實習

MG32x02z 內含有時脈來源控制(CSC: Clock Source Controller)、系統節拍(SysTick)計時器、通用計時模組(TM)、看門狗計時器(WDT)及即時時脈(RTC)。其中在 Arduino 由系統節拍(SysTick)計時器提供延時功能及通用計時模組提供計時中斷及 PWM 輸出功能。

### 5-2.1 時間函數 TIMER 實習

Arduino 官方提供的時間函數如下：

1. 阻塞函數：在延時時間內，MCU 只能計時等待，不能執行其它程式。如下：

(1) delay(ms)：讓設備暫停執行毫秒(ms)

例如 delay(1000); //原地空等延時 1 秒

(2) delayMicroseconds(us)：讓設備暫停執行微秒(us)

例如 delayMicroseconds (100); //原地空等延時 100uS

2. 非阻塞形式：MCU 進行 systick 計時期間，同時可以執行其它程式。如下：

(1) millis()：讀取現在時間的毫秒(ms)數。

(2) micros()：讀取現在時間的微秒(us)數。

(3) 範例 Timer1：以非阻塞形式 systick 計時間隔 1 秒，在序列埠監控窗顯示目前時間。

3. 計時中斷：MCU 內含 7 個通用計時模組(TM00/01/10/16/20/26/36)，致能計時中斷後，同時可以執行其它程式，當計時時間到達會產生計時中斷，並且暫

停目前執行程式，而去執行計時中斷服務函數。範例如下：

- (1) 範例 Timer2：使用程式庫(MsTimer2.h)，致能計時中斷，控制 LED 閃爍時間。
- (2) 範例 Timer3：使用程式庫(TimerOne.h)，控制 LED 閃爍時間，同時在序列埠監控窗顯示 LED 閃爍次數遞加。



## 5-2.2 音樂控制實習

可使用通用計時模組(TM)發出音頻，但在開發板 TH244A 僅限 PWM 輸出接腳 D3、D5、D6、D8、D9、D10、D11 有“~”符號者可輸出音頻，如下圖所示：

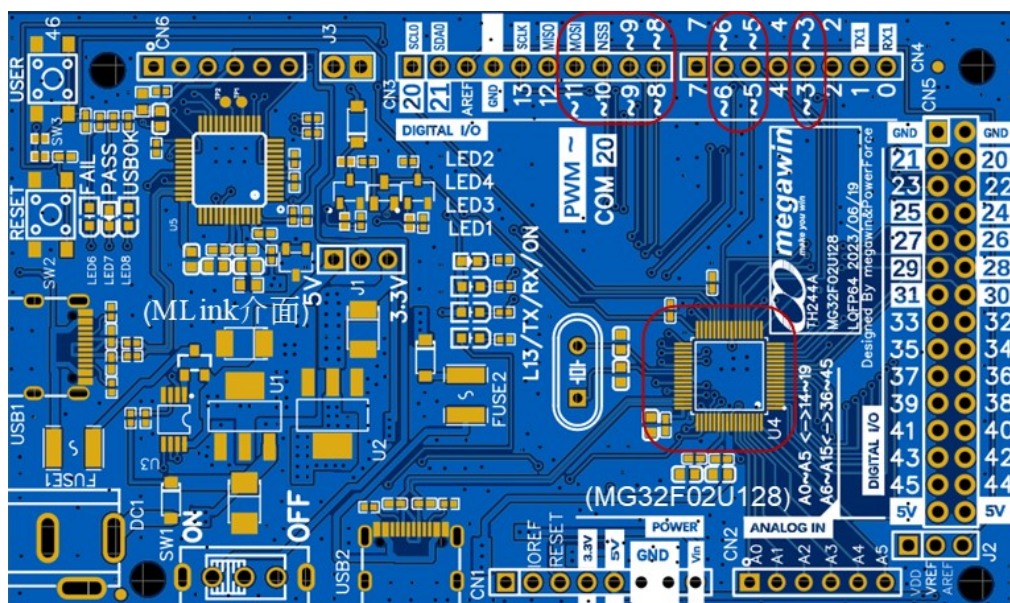


圖 5-10 開發板 TH244A 的 PWM 接腳位置

可使用的音頻(Tone)函數式如下表所示：

表 5-6 音頻(Tone)函數式

| 音頻函數          | 語法                             | 常數                    |
|---------------|--------------------------------|-----------------------|
| tone()        | tone(pin, frequency, duration) | pin : 3、5、6、8、9、10、11 |
| noteDurations | noteDurations(duration)        | frequency=1~65535Hz   |
| noTone()      | noTone(pin)                    | duration=持續 ms        |

1. 可藉由通用計時模組來產生音樂，首先必須知道各種音階的頻率。如下表所示：

表 5-7 音階的頻率

| 八度音   | DO   | DO#  | RE   | RE#  | MI   | FA   | FA#  | SO   | SO#  | LA   | LA#  | SI   |
|-------|------|------|------|------|------|------|------|------|------|------|------|------|
| 第 0 度 | 65   | 69   | 73   | 78   | 82   | 87   | 93   | 98   | 104  | 110  | 116  | 123  |
| 第 1 度 | 131  | 139  | 147  | 156  | 165  | 175  | 185  | 196  | 208  | 220  | 233  | 247  |
| 第 2 度 | 262  | 277  | 294  | 311  | 330  | 349  | 370  | 392  | 415  | 440  | 466  | 494  |
| 第 3 度 | 523  | 554  | 587  | 622  | 659  | 698  | 740  | 784  | 831  | 880  | 932  | 988  |
| 第 4 度 | 1046 | 1109 | 1175 | 1245 | 1318 | 1397 | 1480 | 1568 | 1661 | 1760 | 1865 | 1976 |
| 第 5 度 | 2093 | 2217 | 2349 | 2489 | 2637 | 2794 | 2960 | 3136 | 3322 | 3520 | 3729 | 3951 |
| 第 6 度 | 4186 | 4435 | 4699 | 4978 | 5274 | 5587 | 5919 | 6271 | 6645 | 7040 | 7459 | 7902 |

將這些音階定義在 music.h 內，再依順序讀取資料，將音頻輸出即可演奏音樂。music.h 內容如下：

```

/*****
*檔名：music.h
*功能：節拍和簡譜頻率
*****/

//---第 0 八度音階---
#define DO0    65
#define DO_0   69 //DO0#
#define RE0    73
#define RE_0   78 //RE0#
#define MI0    82
#define FA0    87
#define FA_0   93 //FA0#
#define SO0    98
#define SO_0   104 //SO0#
#define LA0    110
#define LA_0   116 //LA0#
#define SI0    123
//---第 1 八度音階---
#define DO1    131

```

```
#define DO_1 139 //DO1#  
#define RE1 147  
#define RE_1 156 //RE1#  
#define MI1 165  
#define FA1 175  
(後面省略)
```

2. 範例 Music1：使用計時器令蜂鳴器輸出音階。
3. 範例 Music2：使用計時器蜂鳴器輸出音樂。
4. 範例 Music3：使用 IO 埠令蜂鳴器輸出可變的音頻。
5. 範例 Music4：使用可變電阻調整蜂鳴器輸出音頻，同時在序列埠監控窗顯示音頻數值。

### 5-2.3 PWM 輸出實習

脈波寬度調變(PWM : Pulse Width Modulation)可調整電功率，它應用電路的全開(ON)和全關(OFF)來控制電路，工作時損耗極低，有很高的能源效率。可調整喇叭音量、LED 亮度及馬達速度。如下圖所示：

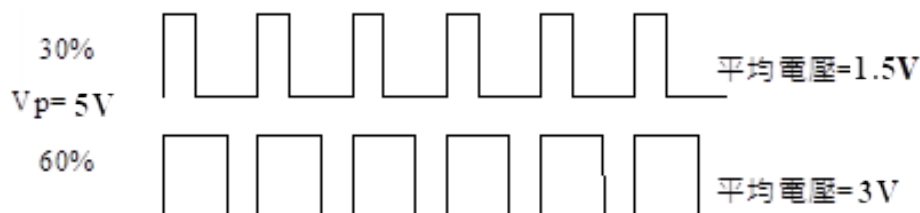


圖 5-11 波寬調變(PWM)輸出波形

PWM 波形以輸出高電位(HI)的脈波佔空比(duty)時間與工作週期(period)時間(HI+LO)的比率，來決定平均電壓。平均電壓(Va)的定義如下：

$$\text{平均電壓} = \text{工作週期} * \text{HI} / (\text{HI} + \text{LO}) * \text{峰值電壓}(V_p)$$

TH244A 開發板使用通用計時模組(TM20/26/36)提供 7 個 PWM 輸出，分別在接腳 D3、D5、D6、D8、D9、D10、D11 有“~”符號者可輸出 PWM 波形。預設頻率為 1KHz，且可以各自獨立控制佔空比(duty)範圍 0~100%，設定函數如下：

```
analogWrite( pin , Value ) , pin =3、6、5、8、9、10、11 , Value=0~255
```

TH244A 開發板的特色是支援使用者可設定頻率(Fre)範圍為 300Hz~5000Hz。頻率分為三個組可獨立設定：3/6(TM20)、5/8/9(TM36)和 10/11(TM26)。如下表所示：

表 5-8 PWM 輸出接腳

| 計時器  | 計時器接腳名稱   | MCU 接腳名稱 | 開發板接腳名稱 |
|------|-----------|----------|---------|
| TM20 | TM20_OC11 | PB10     | 3       |
|      | TM20_OC00 | PC0      | 6       |
| TM36 | TM36_OC12 | PB11     | 5       |
|      | TM36_OC3  | PC12     | 8       |
|      | TM36_OC2  | PD0      | 9       |
| TM26 | TM26_OC00 | PD2      | 11      |
|      | TM26_OC11 | PD9      | 10      |

PWM 設定頻率的函數，如下：

`analogWriteFrequency(PWM_FRQ_xx, Fre)`；設定特定接腳輸出 PWM 的頻率

PWM 設定頻率的範例，如下：

`analogWriteFrequency( PWM_FRQ_D3_D6, Frq )` 設定 3/6 輸出頻率 Fre  
`analogWriteFrequency(PWM_FRQ_D5_D8_D9, Frq )` 設定 5/8/9 輸出頻率  
`analogWriteFrequency( PWM_FRQ_D10_D11, Frq )` 設定 10/11 輸出頻率  
`analogWriteFrequency( PWM_FRQ_ALL, Frq )` 設定全部 PWM 輸出頻率

其中 Frq 為設定頻率，可設定範圍為 300Hz~5000Hz。超出這個範圍的數值都會限制在此範圍內，如 Frq 小於 300 限制為 300Hz 或者大於 5000 限制為 5000Hz。範例如下：

1. 範例 LED1：由序列監控窗或 KEY1~KEY2 控制 LED 亮度。
2. 範例 LED2：由序列監控窗或 KEY1~KEY2 外部中斷控制 LED 亮度。
3. 範例 LED3：令 LED 產生呼吸(漸暗及漸亮)的功能，若接喇叭可控制音量，若接直流馬達可控制轉速。

4. 範例 RGB1：令三色 LED 輸出不同彩色。
5. 範例 RGB2：RGB LED 自動顏色循環變化。
6. 範例 RGB3：RGB LED 自動顏色循環變化。

### 5-2.4 直流馬達控制實習

通用計時模組的 PWM 輸出波形經驅動晶片(L9110)來控制直流馬達，如下圖(a)(b)所示：

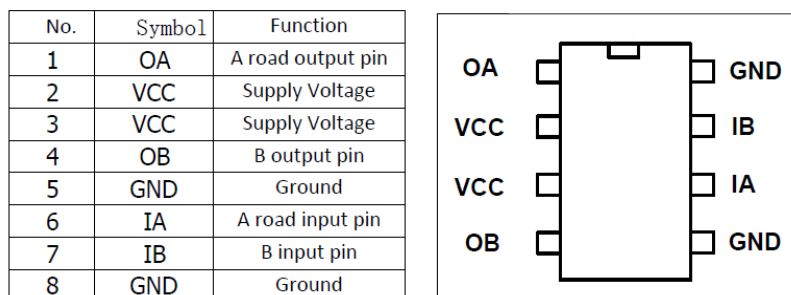


圖 5-12(a) 直流馬達驅動晶片接腳

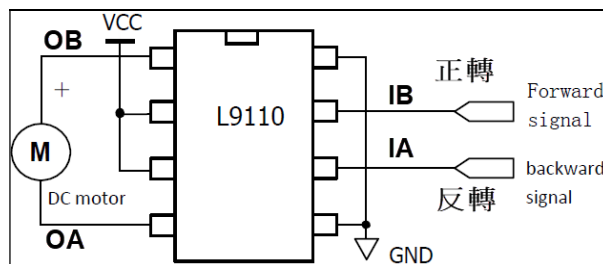


圖 5-12(b) 直流馬達控制電路

馬達驅動晶片 L9110 的電氣特性，如下表所示：

表 5-9 馬達驅動晶片電氣特性

| Symbol           | Parameters        | Range |         |      | Units |
|------------------|-------------------|-------|---------|------|-------|
|                  |                   | Min   | Typical | Max  |       |
| VCC              | Supply Voltage    | 2.5   | 6       | 12   | V     |
| I <sub>dd</sub>  | Quiescent Current | —     | 0       | 2    | uA    |
| I <sub>in</sub>  | Operating current | 200   | 350     | 500  | uA    |
| I <sub>C</sub>   | Continuous        | 750   | 800     | 850  | mA    |
| I <sub>Max</sub> | Current peak      | —     | 1500    | 2000 | mA    |

|   | IA | IB | OA | OB |
|---|----|----|----|----|
| H | L  | H  | L  | L  |
| L | H  | L  | H  | H  |
| L | L  | L  | L  | L  |
| H | H  | H  | L  | L  |

其控制信號，如下圖所示：

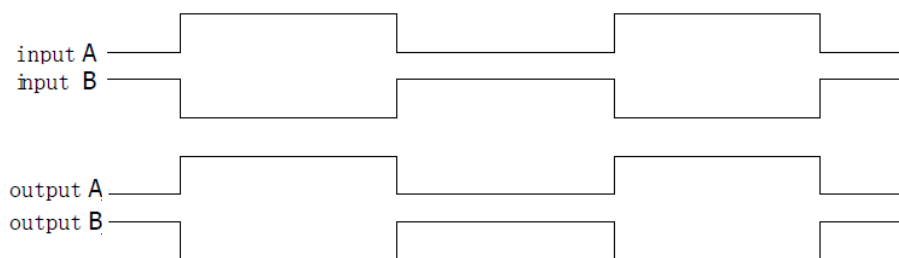


圖 5-13 L9110 馬達驅動晶片控制信號

由 J19 輸入信號(IA1、IB1 及 IA2、IB2)，經 U1 及 U2(L9110)可控制兩軸直流馬達 MOTOR1(J2)及 MOTOR2(J1、M1)，同時可在 J13 切換馬達電源(VM)為內部+5V 或外部電源 J12(VEXT)。如下圖(a)(b)所示：

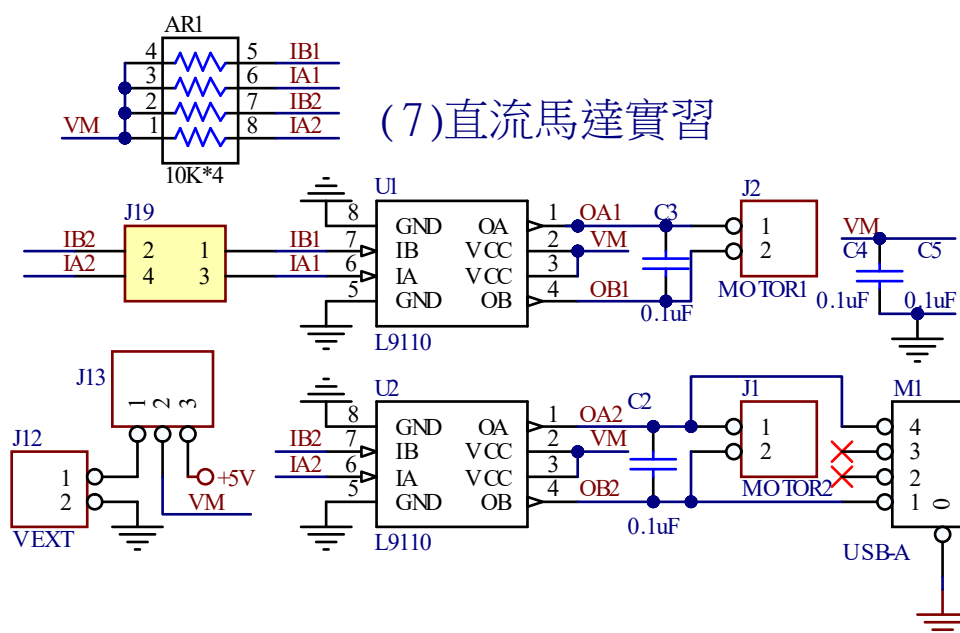


圖 5-14(a) DC 馬達實習電路



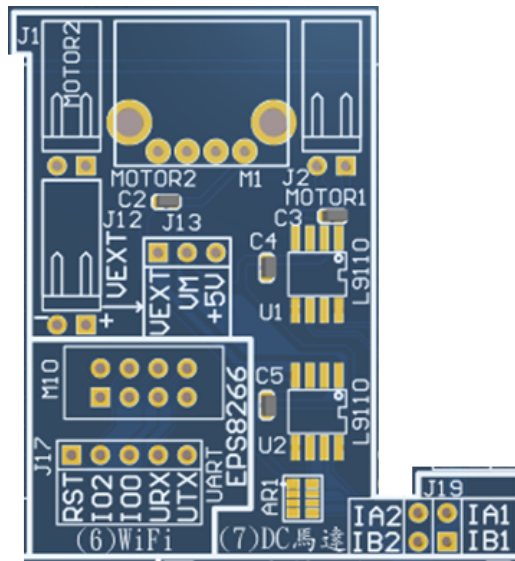


圖 5-14(b) DC 馬達實習外型

1. 範例 DC\_MOTOR1：直流馬達正反轉控制。
2. 範例 DC\_MOTOR2：直流馬達正反轉與轉速控制。

### 5-2.5 RC 伺服馬達控制實習

本章使用角度型的 RC 伺服(Servo)馬達，藉由 PWM 波形的佔空比(duty)來控制角度，其中 PWM 波形的週期(period)時間必須大於 20mS 以上，然後由佔空比(duty)時間 500uS~2500uS 來控制馬達旋轉 0 度~180 度，如下圖)所示：

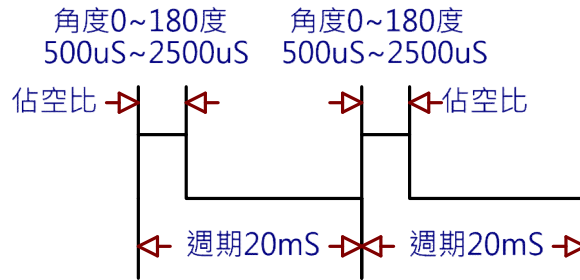


圖 5-15 RC 伺服馬達角度控制(換圖)

RC 伺服馬達實習時，可在 J42(Server)插入兩個 RC 伺服馬達(1~2)，其中棕色(GND)、紅色(+5V)、橙色控制(Ser)。再由 J43(Ser1~2)輸入控制信號，如下圖所示：

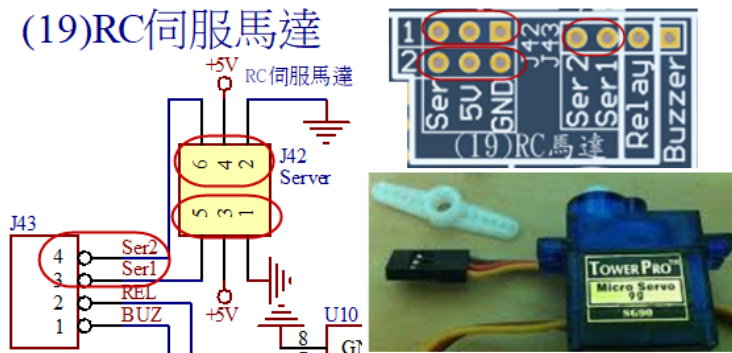


圖 5-16 RC 伺服馬達實習

1. 範例 RC\_MOTOR1：由串列監控視窗輸入 500~2500，RC 伺服馬達轉為 0~90 角度。操作如下圖所示：



圖 5-17 範例 RC\_MOTOR1 操作

2. 範例 RC\_MOTOR2: 由串列監控視窗輸入 0~9, RC 伺服馬達轉為 0~80 角度。

## 5-2.6 看門狗計時器(WDT)控制實習

MG32x02z 內含獨立看門狗計時器(IWDT: Independent Watch Dog Timer)及視窗看門狗計時器(WWDT: Window Watch Dog Timer)。

IWDT 看門狗計時器包括 12-bit 預除頻器、8-bit 計時器。看門狗計時器啟動後 WDT 計時器會由 0xFF 不停的下數，必須在 WDT 計時器下數到 0 之前重新載入計時器為 0xFF，然後再重新開始下數計數。

如果由於受到干擾而令 MCU 當機，無法重新載入 WDT 計時器，在 WDT 計時器下數到 0 時，會自動令系統重置(Reset)或產生中斷，避免系統當機時間過長而產生嚴重後果。

可使用的看門狗時間(WDT)指令，如下所示：

```
/*看門狗時間(WDT)指令*/
wdt_disableRST();    // 禁能 WDT 中斷重置 MCU 功能
wdt_enableRST();      // 致能 WDT 中斷重置 MCU 功能
wdt_enable(timeout period); // 致能及設定 WDT 超時時間(時間較不精準備)
wdt_disable();        // 禁能 WDT
wdt_reset();          // WDT 時間重置
```

設定看門狗時間(WDT)的超時時間，如下：

```
SLEEP_15MS = WDTO_15MS,    // watchdog timer 15ms timeout
SLEEP_30MS = WDTO_30MS,    // watchdog timer 30ms timeout
SLEEP_60MS = WDTO_60MS,    // watchdog timer 60ms timeout
SLEEP_120MS = WDTO_120MS,   // watchdog timer 120ms timeout
```

```
SLEEP_250MS = WDTO_250MS,    // watchdog timer 250ms timeout
SLEEP_500MS = WDTO_500MS,    // watchdog timer 500ms timeout
SLEEP_1S    = WDTO_1S,        // watchdog timer 1s timeout
SLEEP_2S    = WDTO_2S,        // watchdog timer 2s timeout
SLEEP_4S    = WDTO_4S,        // watchdog timer 4s timeout
SLEEP_8S    = WDTO_8S,        // watchdog timer 8s timeout
```

1. 範例 IWDT1\_RST：LED 閃爍，若延時超過 IWDT 時間，則不斷重置，LED 不閃爍。
2. 範例 IWDT2\_RST：設定看門狗時間(WDT)中斷，若 WDT 時間溢位會產生中斷令 LED 閃爍。