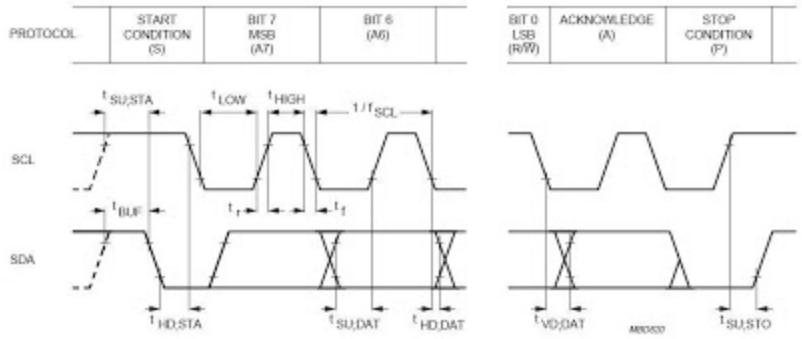
I²C (Inter-Integrated Circuit) 是內部整合電路的稱呼，是一種串列通訊匯流排，使用多主從架構，由飛利浦公司在 1980 年代為了讓主機板、嵌入式系統或手機用以連接低速週邊裝置而發展。I²C 的正確讀法為 "I-squared-C" ，而 "I-two-C" 則是另一種錯誤但被廣泛使用的讀法，在大陸地區則多以 "I方C" 稱之。截至 2006 年 11 月 1 日為止，使用 I²C 協定不需要為其專利付費，但製造商仍然需要付費以獲得 I²C Slave (從屬裝置位址)。
I²C的參考設計使用一個7位元長度的位址空間但保留了16個位址，所以在一組匯流排最多可和112個節點通訊。常見的I²C匯流排依傳輸速率的不同而有不同的模式：標準模式（100 Kbit/s）、低速模式（10 Kbit/s），但時脈頻率可被允許下降至零，這代表可以暫停通訊。而新一代的I²C匯流排可以和更多的節點（支援10位元長度的位址空間）以更快的速率通訊：快速模式（400 Kbit/s）、高速模式（3.4 Mbit/s）。

I2C 的啟動條件及停止條件



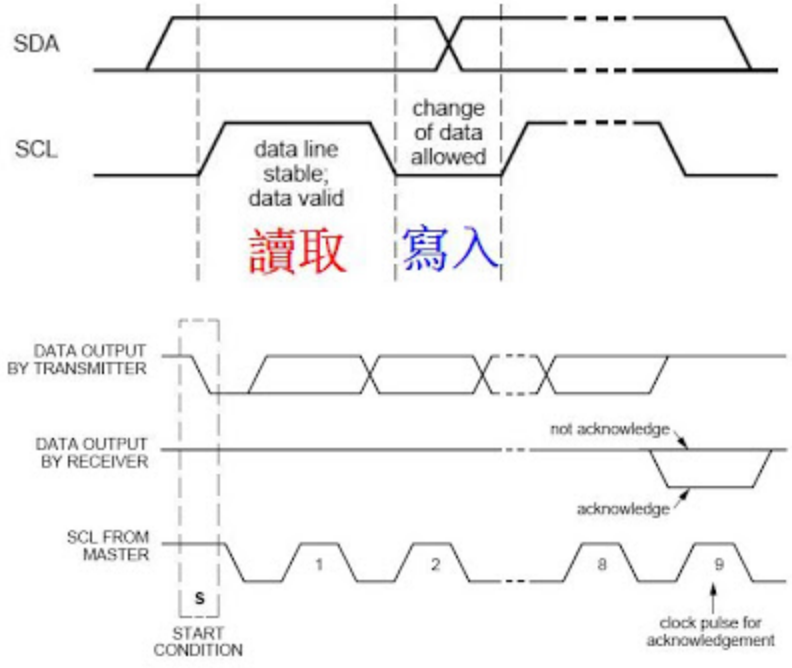
1. I2C start condition 有二種情況，如上圖所示，虛線表示 read 動作時的第二次 start condition，實線表示 r/w 時的第一次 start condition。
2. I2C stop condition 只有一種情況，如上圖所示。

```
void i2c_start(void)
{
   // for second start signal on i2c_read
   I2C_SDA = HIGH;
   I2C_SCL = HIGH;
   i2c_wait();

   // send start signal
   I2C_SDA = LOW;
   i2c_wait2();
   I2C_SCL = LOW;
}

void i2c_stop(void)
{
   i2c_wait2();
   I2C_SDA = LOW;
   i2c_wait2();
   I2C_SCL = HIGH;
   i2c_wait2();
   I2C_SDA = HIGH;
}
```

I2C 的讀寫動作

1. 當 SCL=HIGH 時，表示 SDA 穩定，可以做讀取動作。

2. 當 SCL=LOW 時，表示 SDA 混亂，不可以讀取；因為此時可以設定 SDA 的值，也就是做寫入動作。
3. master 每一次傳送八個 bit，最後 slave 會回傳一個 ack bit，表示接受是否完成。
4. master 每一次接受八個 bit，最後 master 要傳送一個 ack bit，表示接受已經完成。
5. 在傳送完第八個 bit 之後，再等待 slave 接受完成後，需將 SDA 設成 HIGH，此時 slave 會將 SDA 拉回 LOW，表示接受動作完成。如果 acknowledge=HIGH，也就是 slave 沒有拉成 LOW 則表示傳送失敗。



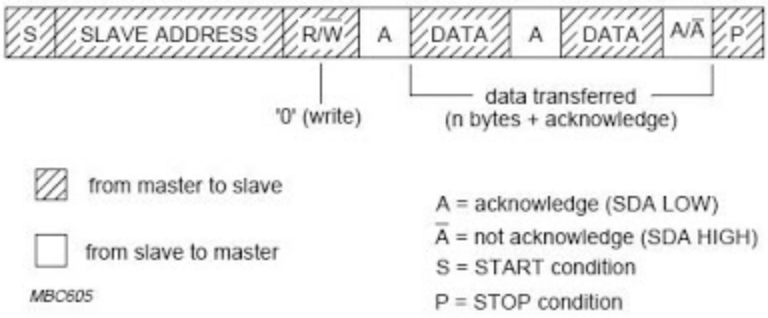
bit i2c_write(unsigned char value)

```
{
    char i=9;
    bit  ack;
    // upload data
    while(--i)
    {
        i2c_wait2();
        I2C_SDA = (value &amp;amp; 0x80) ? HIGH : LOW;
        i2c_wait2();
        // send data
        I2C_SCL = HIGH;
        i2c_wait();
        value &amp;lt;&amp;lt;= 1;
        I2C_SCL = LOW;
    }
    // get acknowledgement
    i2c_wait2();
    I2C_SDA = HIGH;
    i2c_wait2();
    I2C_SCL = HIGH;
    i2c_wait2();
    ack = I2C_SDA;
    i2c_wait2();
```

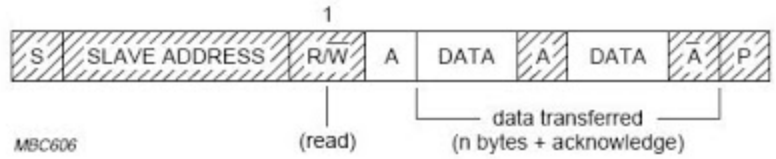
```
    I2C_SCL = LOW;
    // return acknowledge
    return ack;
}
unsigned char i2c_read(bit acknowledge)
{
    char i=9;
    unsigned char value=0;
    // read data
     while(--i)
    {
        value &amp;lt;&amp;lt;= 1;
        i2c_wait();
        I2C_SCL = HIGH;
        i2c_wait2();
        value |= I2C_SDA;
        i2c_wait2();
        I2C_SCL = LOW;
    }
    // send acknowledge
    i2c_wait2();
    I2C_SDA = acknowledge;
    i2c_wait2();
    I2C_SCL = HIGH;
    i2c_wait();
    I2C_SCL = LOW;
    // return data
    return value;
}
```

標準 I2C 讀寫流程 by Philips

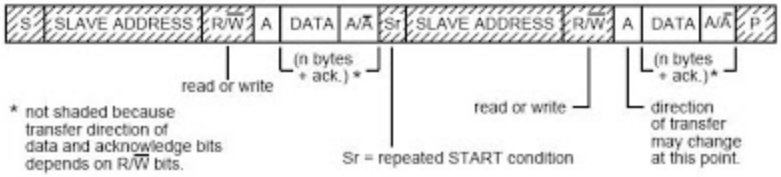
1. 讀取完最後一個 byte 時，記得要回傳 no acknowledge(SDA=HIGH)，表示已經沒有要繼續讀取資料。
2. 讀取完最後一個 byte 時，回傳 acknowledge(SDA=LOW)，再傳送 stop signal，則會造成後續的讀寫動作失敗。(這是在讀取 PCF8593 的經驗)



完整寫入流程



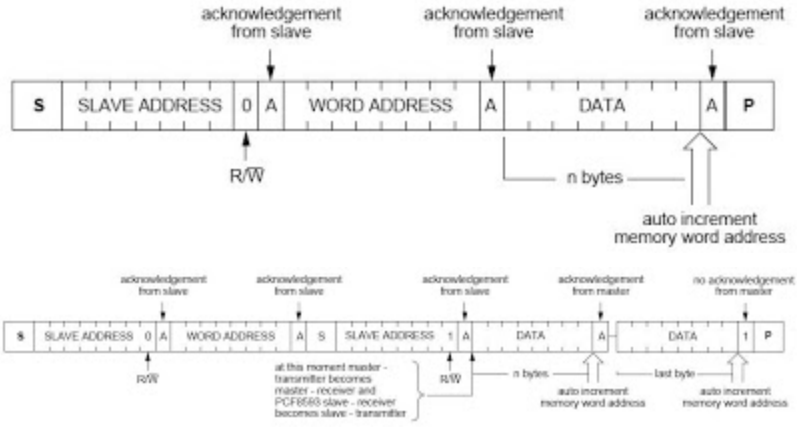
完整讀取流程



複合式讀寫流程

常用 I2C NVRAM 讀寫流程 by Philips PCF8953

1. 一般 RTC 都有帶一些 NVRAM 或是讀寫 EEPROM，需要先指定讀寫的 register address 才可以。所以在寫時，先寫入 slave address 後，需再寫入 register address，讓 chip 知道你要寫入的起始位址，接下來才能寫入 data。
2. 由於讀取前也要先寫入 register address，所以一般 NVRAM 讀取動作都是使用標準複合式流程，也就是先寫入 register address，再下一次 start conditon，再做讀取動作。




```
void i2c_write_byte(unsigned char slave_addr, unsigned char reg_addr, unsigned char value)
{
   char i=10;

   EA = 0;
   while (--i)
   {
      i2c_start();
      if(i2c_write(slave_addr)) continue;
      if(i2c_write(reg_addr)) continue;
      if(i2c_write(value)) continue;
      i2c_stop();
      break;
   }
```

```
    EA = 1;
}

unsigned char i2c_read_byte(unsigned char slave_addr, unsigned char reg_addr)
{
    char i=10;
    unsigned char value=0;

    EA = 0;
    while (--i)
    {
        // send register address
        i2c_start();
        if(i2c_write(slave_addr&amp;amp;0xfe)) continue;
        if(i2c_write(reg_addr)) continue;
        // read data
        i2c_start();
        if(i2c_write(slave_addr | 1)) continue;
        value = i2c_read(1);
        i2c_stop();
        break;
    }
    EA = 1;
    return value;
}
```