SIOC嵌入式軟體實驗
實驗九：I2C

WU-YANG
Technology Co., Ltd.

support.wuyang@gmail.com

Outline

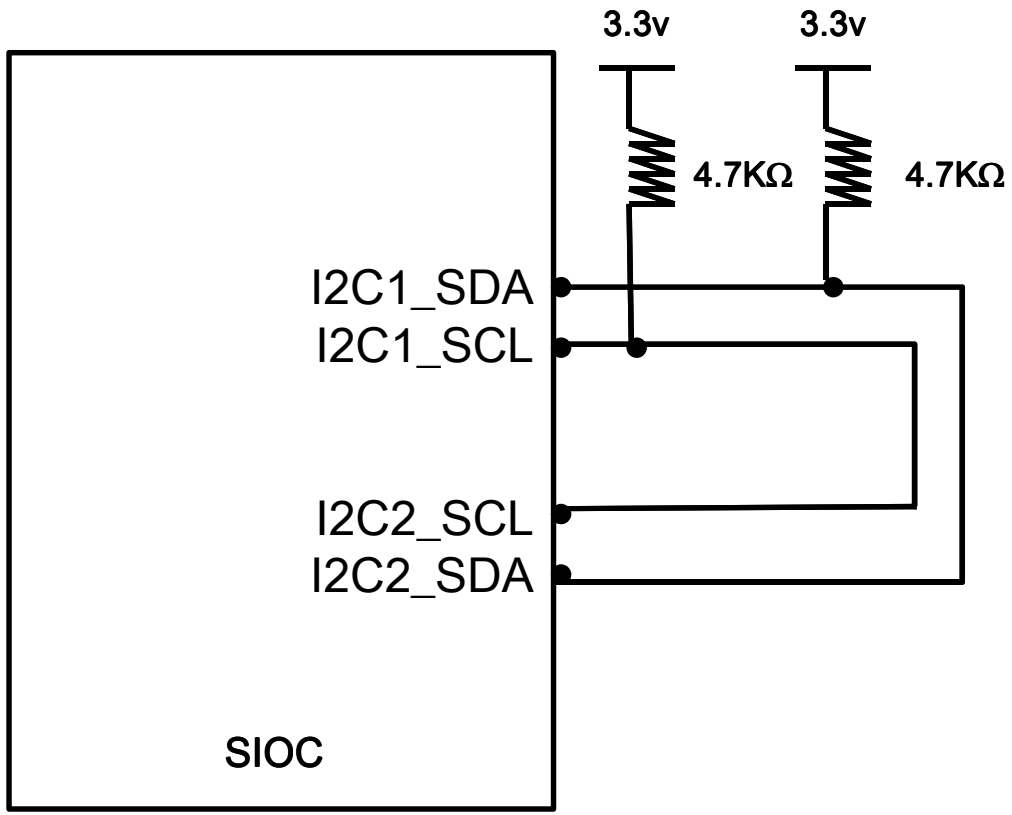
- I2C Introduction
- I2C Standard Driver Library
- 實驗

I2C Introduction

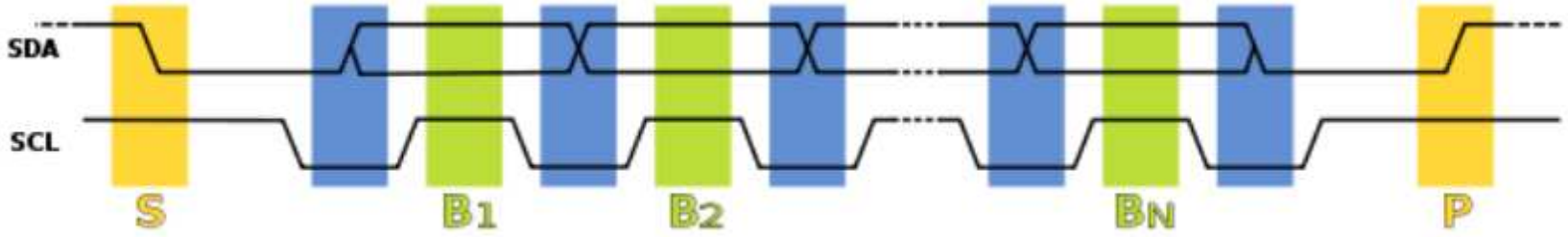
I2C Introduction

- I²C使用兩條雙向開放集極(Open Drain)，串列資料(SDA)及串列時脈(SCL)並利用電阻將電位上拉。
- I²C允許相當大的工作電壓範圍，但典型的電壓準位為+3.3v或+5v。
- 由SDA和SCL構成的串列匯流排，可發送和接收資料。在CPU與被控IC之間、IC與IC之間進行雙向傳送。
- I2C匯流排在傳送資料過程中共有三種類型信號， 它們分別是：START信號、FINISH信號和ACK信號。

Hardware Architecture



State Signal



START：SCL為high時，SDA由high變成low，開始傳送資料。
FINISH：SCL為high時，SDA由low變成high，結束傳送資料。
ACK：接收資料的IC在接收到8bit資料後，向發送資料的IC發出特定的低電平脈衝，表示已收到資料。

I2C Standard Driver Library

I2C_InitTypeDef structure

```
typedef struct
{
    u16 I2C_Mode;
    u16 I2C_DutyCycle;
    u16 I2C_OwnAddress1;
    u16 I2C_Ack;
    u16 I2C_AcknowledgedAddress;
    u32 I2C_ClockSpeed;
} I2C_InitTypeDef;
```

I2C_InitTypeDef -I2C_Mode

I2C_Mode	Description
I2C_Mode_I2C	I2C is configured in I2C mode
I2C_Mode_SMBusDevice	I2C is configured in SMBus device mode
I2C_Mode_SMBusHost	I2C is configured in SMBus host mode

I2C_InitTypeDef - I2C_DutyCycle

I2C_DutyCycle	Description
I2C_DutyCycle_16_9	I2C fast mode Tlow/Thigh=16/9
I2C_DutyCycle_2	I2C fast mode Tlow/Thigh=2

I2C_InitTypeDef - I2C_OwnAddress1

I2C_OwnAddress1	Description
I2C1_SLAVE_ADDRESS7	This member is used to configure the first device own address.
I2C2_SLAVE_ADDRESS7	

I2C_InitTypeDef - I2C_Ack

I2C_Ack	Description
I2C_Ack_Enable	Enables the acknowledgement
I2C_Ack_Disable	Disables the acknowledgement

I2C_ITConfig function

Function prototype	void I2C_ITConfig(I2C_TypeDef* I2Cx, u16 I2C_IT, FunctionalState NewState)
Behavior description	Enables or disables the specified I2C interrupts.
Input parameter1	I2Cx: where x can be 1 or 2 to select the I2C peripheral.
Input parameter2	I2C_IT: I2C interrupts sources to be enabled or disabled. Refer to I2C_IT for more details on the allowed values for this parameter.
Input parameter3	NewState: new state of the specified I2C interrupts. This parameter can be set to ENABLE or DISABLE.

- Example
  I2C_ITConfig(I2C2, I2C_IT_BUF | I2C_IT_EVT, ENABLE);

I2C_Cmd function

Function prototype	void I2C_Cmd(I2C_TypeDef* I2Cx, FunctionalState NewState)
Behavior description	Enables or disables the specified I2C peripheral.
Input parameter1	I2Cx: where x can be 1or 2 to select the I2C peripheral.
Input parameter2	NewState: new state of the I2Cx peripheral. This parameter can be set to ENABLE or DISABLE.

- Example
  I2C_Cmd(I2C1, ENABLE);

I2C_GenerateSTART function

Function prototype	void I2C_GenerateSTART(I2C_TypeDef* I2Cx, FunctionalState NewState)
Behavior description	Generates I2Cx communication Start condition.
Input parameter1	I2Cx: where x can be 1or 2 to select the I2C peripheral.
Input parameter2	NewState: new state of the I2C Start condition generation. This parameter can be: ENABLE or DISABLE.

- Example
  I2C_GenerateSTART(I2C1, ENABLE);

I2C_GenerateSTOP function

Function prototype	void I2C_GenerateSTOP(I2C_TypeDef* I2Cx, FunctionalState NewState)
Behavior description	Generates I2Cx communication Stop condition.
Input parameter1	I2Cx: where x can be 1or 2 to select the I2C peripheral.
Input parameter2	NewState: new state of the I2C Stop condition generation. This parameter can be: ENABLE or DISABLE.

- Example
  I2C_GenerateSTOP(I2C2, ENABLE);

I2C_SendData function

Function prototype	void I2C_SendData(I2C_TypeDef* I2Cx, u8 Data)
Behavior description	Sends a data byte through the I2Cx peripheral.
Input parameter1	I2Cx: where x can be 1 or 2 to select the I2C peripheral.
Input parameter2	Data: byte to be transmitted.

- Example
  I2C_SendData(I2C2, 0x5D);

I2C_ReceiveData function

Function prototype	u8 I2C_ReceiveData(I2C_TypeDef* I2Cx)
Behavior description	Returns the most recent received data by the I2Cx peripheral.
Input parameter	I2Cx: where x can be 1 or 2 to select the I2C peripheral.

- Example

```
u8 ReceivedData;
ReceivedData = I2C_ReceiveData(I2C1);
```

實驗

1、兩組I2C互相傳輸固定筆數資料

2、兩組I2C互相傳輸任意筆數資料

實驗1

- 說明：

  如何使用interrupt讓I2C_1(master)能傳送data給I2C_2(slave)，其中使用7 bits address模式，clock rate是200KHz。

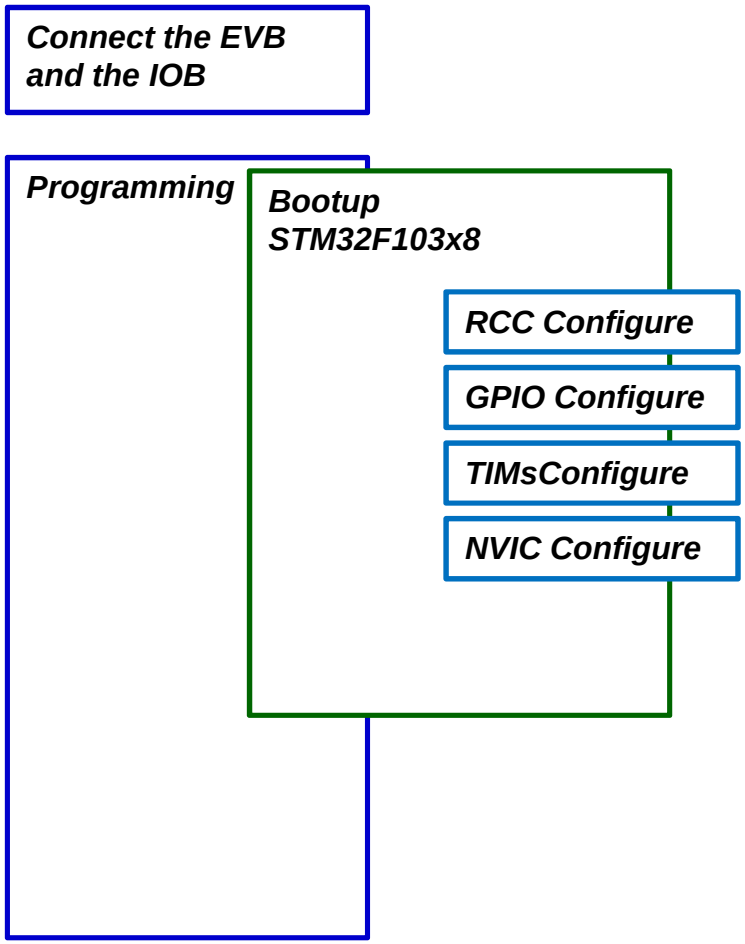
Step 1程式架構

- 程式架構

	<..\I2C>
<project>	單元實驗**Project**目錄
<source>	程式碼目錄
<include>	引入檔目錄
<library>	函式庫目錄
<image>	燒錄配置檔目錄
	<..\I2C\image>
Lab.dfu	燒錄配置檔
	<..\I2C\source>
main.c	硬體配置程式
stm32f10x_it.c	中斷服務程式
hw_config.c	Enable I2C

Development Flow

Embedded Software Side




```
int main(void)
{
 I2C Configure();
  //I2C1's & I2C1's Buffer compare and
save return state;
  Buffercmp();


  if(TransferStatus)
    SUCESS
  else
    FAILED
}
```

Configure RCC

RCC FwLib Functions List

Function name	Description
RCC_DeInit	Resets the RCC clock configuration to the default reset state.
RCC_HSEConfig	Configures the External High Speed oscillator (HSE).
RCC_WaitForHSEStartUp	Waits for HSE start-up.
RCC_HCLKConfig	Configures the AHB clock (HCLK).
RCC_PCLK1Config	Configures the Low Speed APB clock (PCLK1).
RCC_PCLK2Config	Configures the High Speed APB clock (PCLK2).
RCC_PLLConfig	Configures the PLL clock source and multiplication factor.
RCC_PLLCmd	Enables or disables the PLL.
RCC_SYSCLKConfig	Configures the system clock (SYSCLK).
RCC_APB2PeriphClockCmd	Enables or disables the High Speed APB (APB2) peripheral clock.

```
void Set_System(void)
{
  .
  .
  .
#ifndef USE_STM3210C_EVAL
  /* Enable USB_DISCONNECT GPIO clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIO_DISCONNECT, ENABLE);

  /* Configure USB pull-up pin */
  GPIO_InitStructure.GPIO_Pin = USB_DISCONNECT_PIN;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD;
  GPIO_Init(USB_DISCONNECT, &GPIO_InitStructure);
#endif /* USE_STM3210C_EVAL */
  Set_USBClock();
  USB_Interrupts_Config();
  USB_Init();
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  /* Enable I2C1 and I2C2 clock */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1 | RCC_APB1Periph_I2C2, ENABLE);
  /* Enable GPIOB clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
}
```

Configure I2C (1)

```
int main(void) {
  …
  /* I2C1 configuration ------------------------------------------------------*/
  //modify your code

  I2C_InitStructure.I2C_Mode =                //I2C模式
  I2C_InitStructure.I2C_DutyCycle =           //快速模式下的選項，100KHZ以上才有用
  I2C_InitStructure.I2C_OwnAddress1 =         //slave 7 address長度
  I2C_InitStructure.I2C_Ack =                 //每次都會回送ACK
  I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
  I2C_InitStructure.I2C_ClockSpeed = ClockSpeed;
        //I2C 速度配置，這個範例是200KHz，一般是40KHZ，400KHZ是極限，一般到不了那麼高
  I2C_Init(I2C1, &I2C_InitStructure);
  printf("I2C1 configuration\r\n");

  /* I2C2 configuration ------------------------------------------------------*/
  //modify your code
  I2C_InitStructure.I2C_OwnAddress1 = //slave 7 address長度
  I2C_Init(I2C2, &I2C_InitStructure);
  printf("I2C2 configuration\r\n");


}
```

Configure I2C (2)

```
int main(void) {
…
  /*----- Transmission Phase -----*/
  /* Send I2C1 START condition */

  /* Enable I2C1 and I2C2 event and buffer interrupt */
  I2C_ITConfig(I2C1, I2C_IT_EVT | I2C_IT_BUF, ENABLE);
  I2C_ITConfig(I2C2, I2C_IT_EVT | I2C_IT_BUF, ENABLE);

  //modify your code
  /* Enable I2C1 and I2C2  by I2C_Cmd()-------------------------------*/

  //modify your code
  /*I2C1 send START signal-----------------------------------------------*/
  printf("I2C1 Send START condition\r\n");

  /* Send data */
  while(Rx_Idx < (BufferSize+1))
  {
  }

  /* Check the corectness of written data */
  printf("Check the corectness of written data\r\n");
  //modify your code
  /*call Buffercmp Function for comparing Tx & Rx, then return state to TransferStatus*/
  printf("Buffedrcmp Finish\r\n");
}
```

Configure I2C (3)

```
int main(void) {
   …
  //modify your code
  /* TransferStatus = PASSED, if the transmitted and received data are equal */
  /* TransferStatus = FAILED, if the transmitted and received data are different
*/


}
```

IRQ Service(1)

```
void I2C1_EV_IRQHandler(void)
{
  switch (I2C_GetLastEvent(I2C1)){
    /* Test on I2C1 EV5 and clear it */
    case I2C_EVENT_MASTER_MODE_SELECT:
      …
      break;
    /* Test on I2C1 EV6 and first EV8 and clear them */
    case I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED:
      …
      //modify your code
      /* I2C_SendData Send the first data */
      break;
    /* Test on I2C1 EV8 and clear it */
    case I2C_EVENT_MASTER_BYTE_TRANSMITTED:
      if(Tx_Idx < BufferSize){
        //modify your code
        /* I2C_SendData Send buffer data */
      }
      else{
        …
      }
      break;
    default:
      break;
  }
}
```

IRQ Service(2)

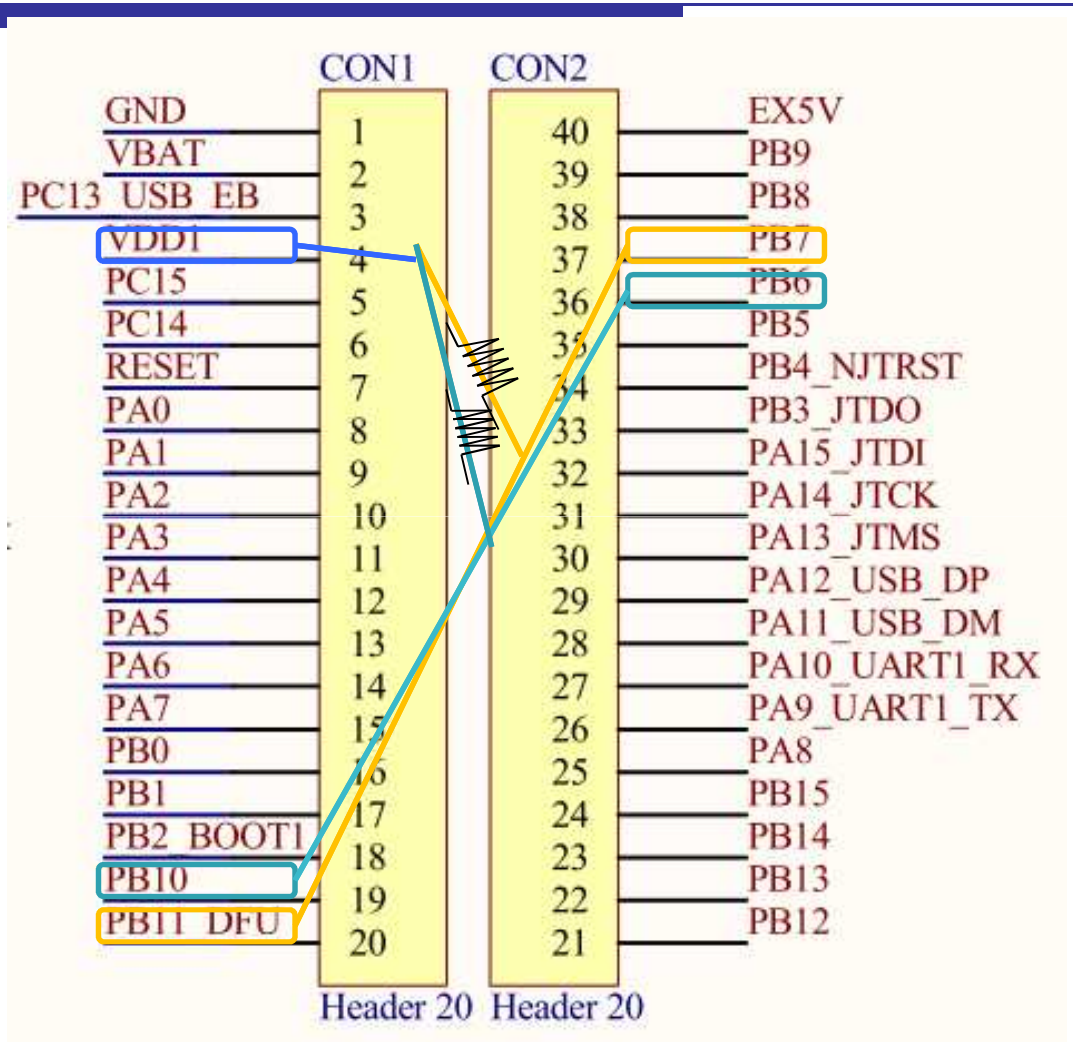
```
void I2C2_EV_IRQHandler(void)
{
  switch (I2C_GetLastEvent(I2C2)){
    /* Test on I2C2 EV1 and clear it */
    case I2C_EVENT_SLAVE_RECEIVER_ADDRESS_MATCHED:
      break;

    /* Test on I2C2 EV2 and clear it */
    case I2C_EVENT_SLAVE_BYTE_RECEIVED:
      if (Rx_Idx < BufferSize){
        //modify your code
        /* I2C2_Buffer_Rx Store received data buffer */
      }
      else{
        …
      }
      break;
    /* Test on I2C2 EV4 and clear it */
    case I2C_EVENT_SLAVE_STOP_DETECTED:
      …
      break;
    default:
      break;
  }
}
```

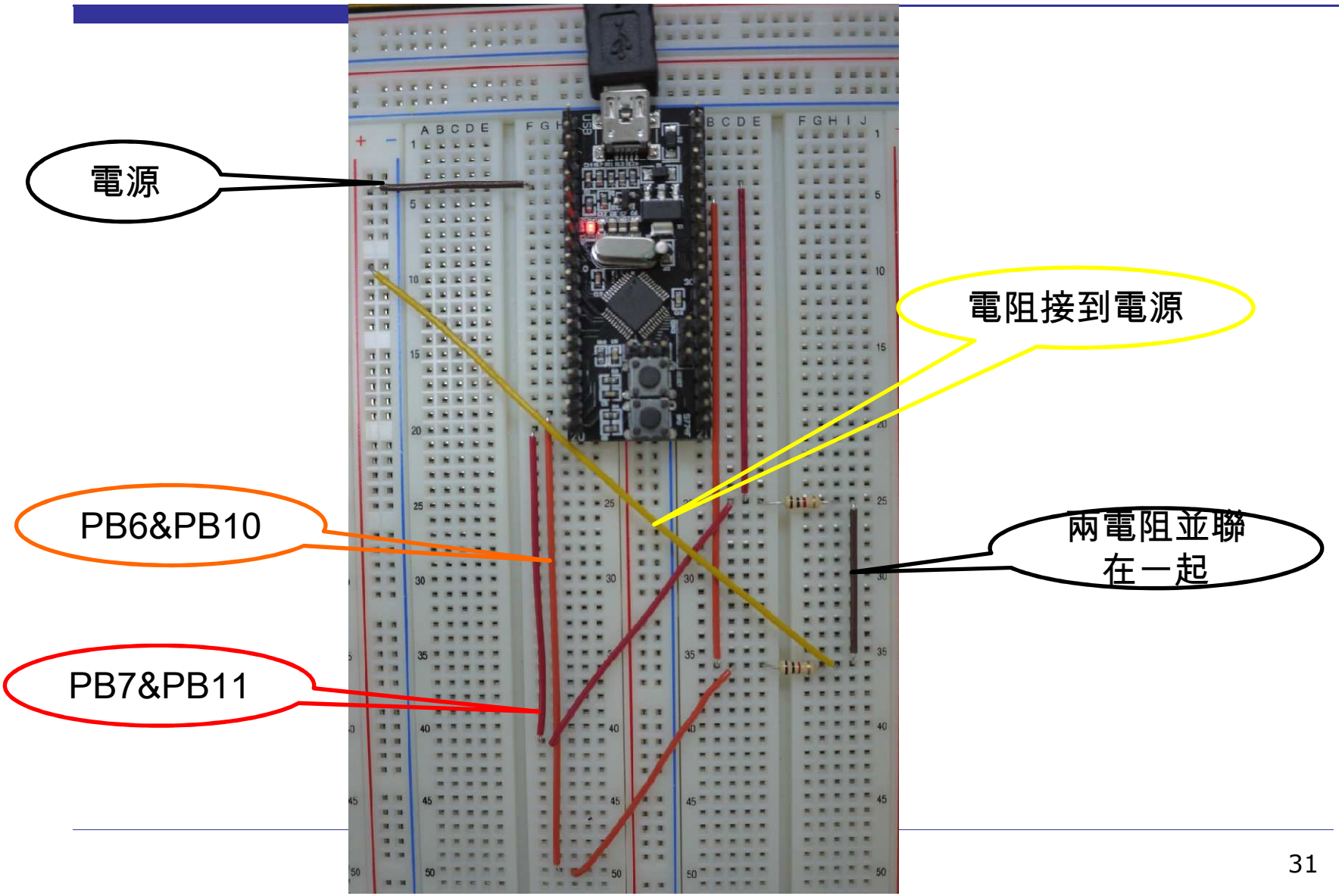
Step2 硬體電路配置(1)

SIOC腳位名稱	SIOC腳位編號
VCC3.3V	CON1.4
I2C1_SCL	CON1.10
I2C1_SDA	CON1.11
I2C2_SCL	CON2.36
I2C2_SDA	CON2.37

Step2 硬體電路配置(2)



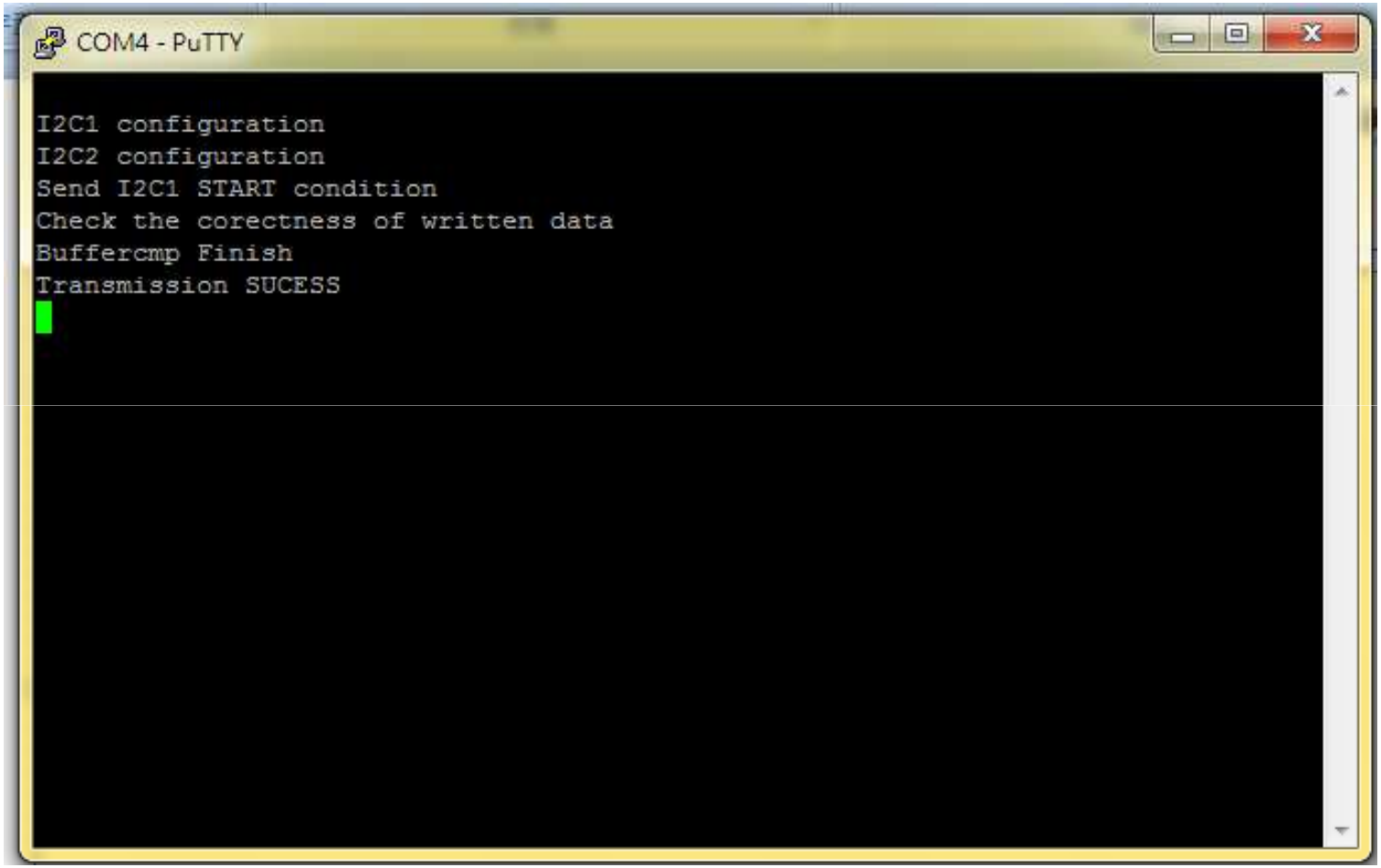
Step2 硬體電路配置(3)



Step 3 編譯燒錄程式並觀察結果

- 編譯
- 將編譯後的hex檔轉換為dfu
- 透過USB 燒錄dfu檔
- 注意燒入時不可以接電源，燒入完成後再接起電源線

DEMO


COM4 - PuTTY
I2C1 configuration
I2C2 configuration
Send I2C1 START condition
Check the corectness of written data
Buffercmp Finish
Transmission SUCESS

實驗2

- 說明：

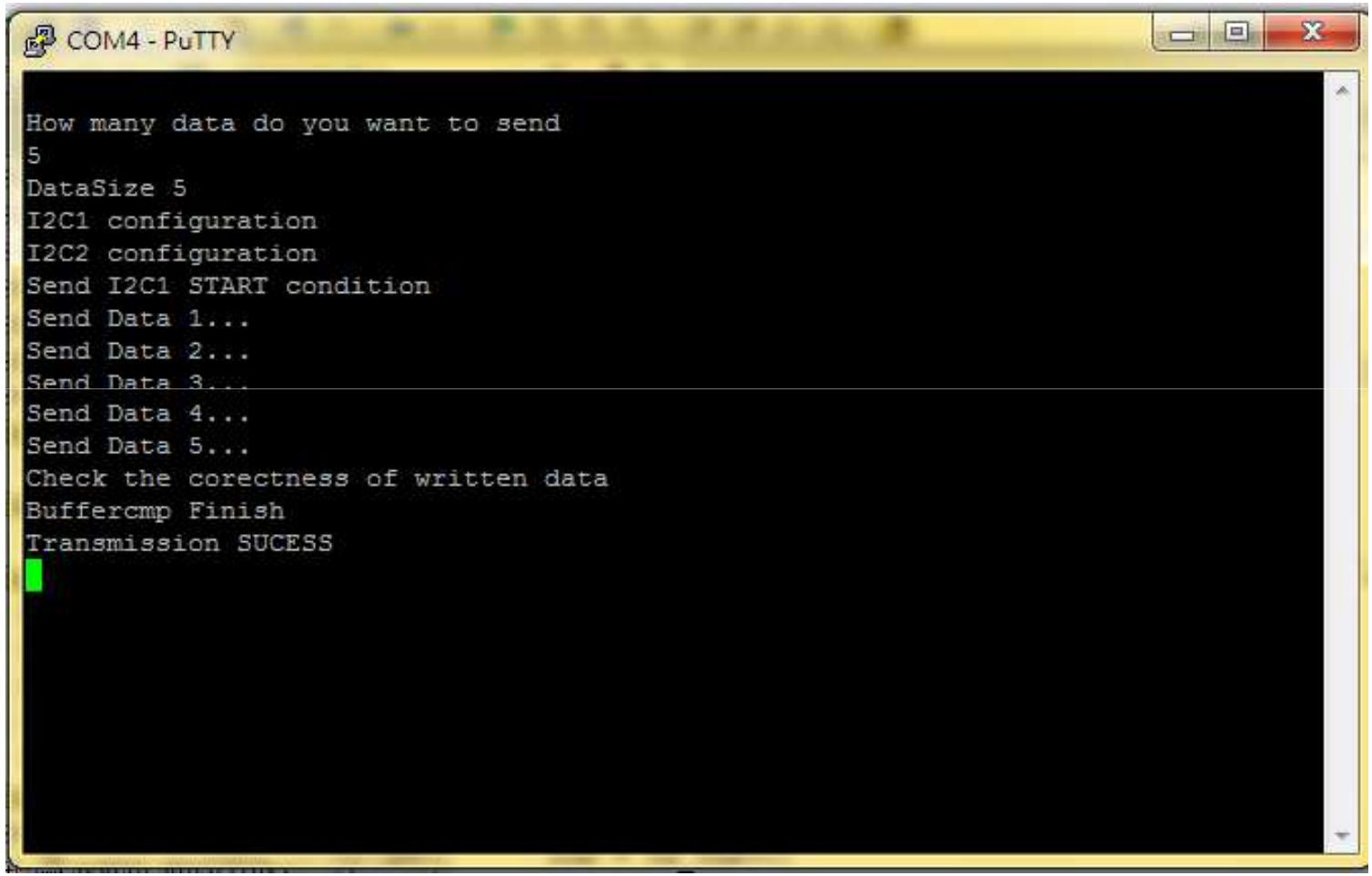
  如何使用interrupt讓能傳送使用者指定多少筆數目的data給I2C_2，其中使用7 bits address模式，clock rate是200KHz。

- 要求：

  BufferSize 設定為10

  I2C1_Buffer_Tx[BufferSize] = {1,2,3,4,5,6,7,8,9,10};

DEMO


COM4 - PuTTY
How many data do you want to send
5
DataSize 5
I2C1 configuration
I2C2 configuration
Send I2C1 START condition
Send Data 1...
Send Data 2...
Send Data 3...
Send Data 4...
Send Data 5...
Check the corectness of written data
Buffercmp Finish
Transmission SUCESS

Q & A