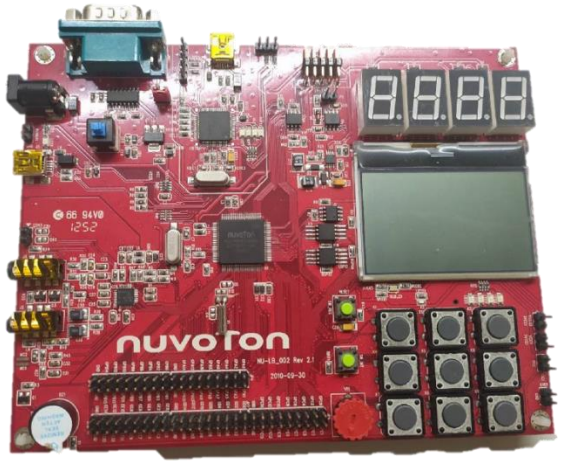
微處理機系統與介面技術 LAB 5

系所：通訊三　　學號 :409430030　　姓名:翁佳煌

<實驗器材>

NUC 140 開發板



PL2303 USB to UART 線

<實驗過程與方法>

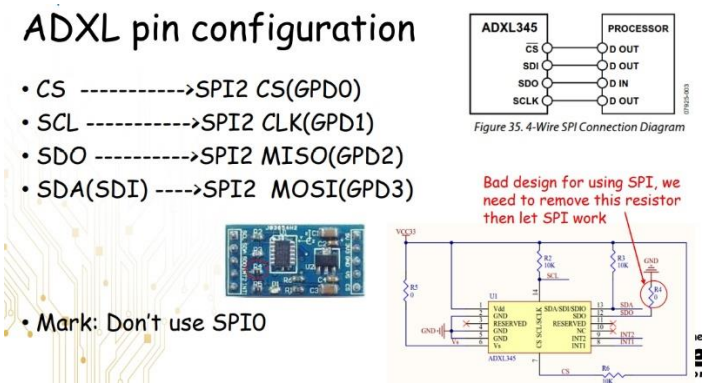
首先，先按照下**圖 1.1** 完成接線，

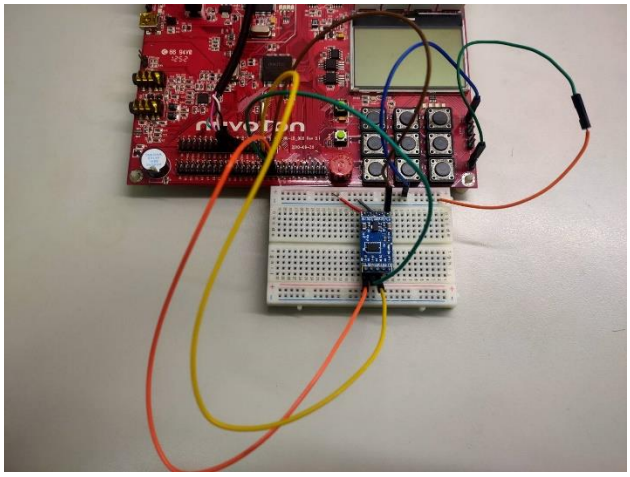
CS 為選擇從機設備（由主設備控制），一般預設為低電位選中。

SCLK 為系統時鐘訊號（由主設備產生）。

MISO (Master In Slave Out)主設備接收 從設備傳送。

MOSI (Master Out Slave In)主設備傳送 從設備接收。





▲圖 1.1

Main 函示中，第 85~91 行為負責印出 ADXL 三軸的資料，SPI_read 的部分後面會說明。

```
68  int main(void)
69  {
70      /* Unlock protected registers */
71      SYS_UnlockReg();
72
73      /* Init system, peripheral clock and multi-function I/O */
74      SYS_Init();
75
76      /* Lock protected registers */
77      SYS_LockReg();
78
79      /* Configure UART0: 115200, 8-bit word, no parity bit, 1 stop bit. */
80      UART_Open(UART0, 115200);
81
82      /* Init SPI */
83      SPI_Init();
84
85      while(1){
86        axisdata[0]=((float)((SPI_read(DATAX1) << 8) | SPI_read(DATAX0))/256);
87        axisdata[1]=((float)((SPI_read(DATAY1) << 8) | SPI_read(DATAY0))/256);
88        axisdata[2]=((float)((SPI_read(DATAZ1) << 8) | SPI_read(DATAZ0))/256);
89        for( i=0;i<2000000;i++){}
90        printf("x=%.2f  y=%.2f  z=%.2f\n",axisdata[0],axisdata[1],axisdata[2]);
91      }
92      while(1);
93  }
```

▲圖 1.2

在上**圖 1.2** 第 74 行中的 SYS_Init ()裡面，因為 sample_code 是採用 SPI0，本實驗需要使用 SPI2，所以第 134~135 行的部分需改成 SPI2 的 multi-function pin，如下**圖 1.3** 所示。

```
133     /* Setup SPI2 multi-function pins */
134     SYS->GPD_MFP = SYS_GPD_MFP_PD0_SPI2_SS0 | SYS_GPD_MFP_PD1_SPI2_CLK | SYS_GPD_MFP_PD2_SPI2_MISO0 | SYS_GPD_MFP_PD3_SPI2_MOSI0;
135     SYS->ALT_MFP = SYS_ALT_MFP_PD0_SPI2_SS0 | SYS_ALT_MFP_PD1_SPI2_CLK | SYS_ALT_MFP_PD2_SPI2_MISO0 | SYS_ALT_MFP_PD3_SPI2_MOSI0;
136
```

▲圖 1.3

在上**圖 1.2** 第 83 行呼叫的 SPI_Init()中(下**圖 1.4**)，首先第 152 行對 SPI2 配置成 master 模式，選擇為 SPI_MODE_2(CLKP=1; RX_NEG=1; TX_NEG=0)，並設定 DataWidth 為 0x08。

第 154 行則是設定相對應的 SPI2 control register。

第 155 行則是關閉 AutoSS 的功能。

第 156 行把 ss 訊號線拉 HIGH，代表目前閒置。

第 159~177 行進行 ADXL 的初始化，SPI_write 的部分後面會說明。

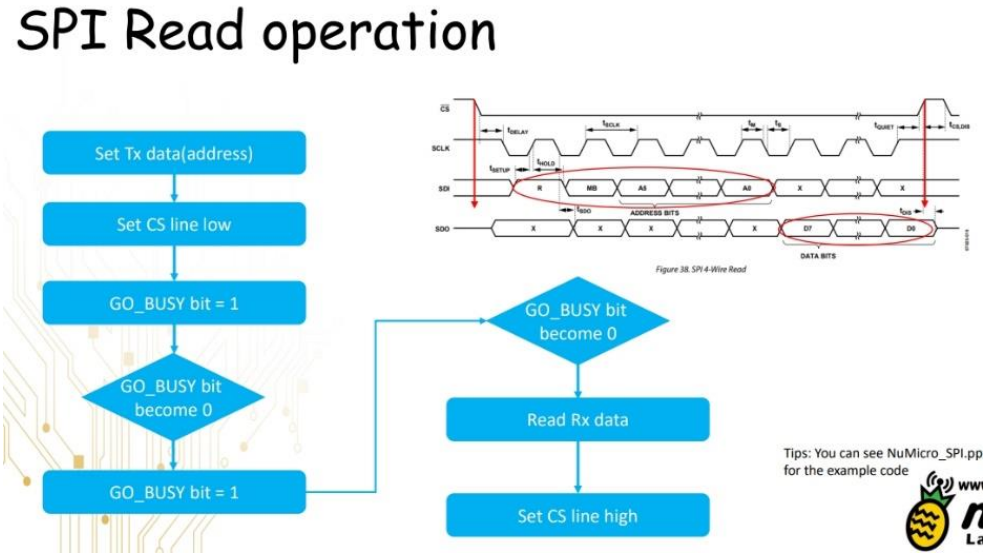
```
142  void SPI_Init(void)
143  {
144      int offset[3],loop=20;
145      /*---------------------------------------------------------------------------------------------------------*/
146      /* Init SPI                                                                                                */
147      /*---------------------------------------------------------------------------------------------------------*/
148      /* Configure as a master, clock idle low, 32-bit transaction, drive output on falling clock edge and latch input on rising edge. */
149      /* Set IP clock divider. SPI clock rate = 2 MHz */
150      printf("\nStart...\n");
151
152      SPI_Open(SPI2, SPI_MASTER, SPI_MODE_2, 0x08, 2000000);
153
154      SPI2->CNTRL |=0x846;
155      SPI_DisableAutoSS(SPI2);
156      SPI_SET_SS0_HIGH(SPI2);
157
158      //write
159      SPI_write(DATA_FORMAT,0x0b);
160      SPI_write(POWER_CTL,0x08);
161      SPI_write(FIFO_CTL,0x80);
162      for(i=0;i<3;i++)
163          SPI_write(0x1e+i,0);
164
165      //identify
166      printf("device id: %x\n",(uint8_t)SPI_read(0x00));
167      for(j=0;j<3000000;j++){}
168
169      for(j=0;j<loop;j++)
170      {
171          for(i=0;i<3;i++){
172              offset[i] += ((SPI_read(0x33+(2*i))<<8) | SPI_read(0x32+(2*i)));
173          }
174      }
175      SPI_write(0x1e,-1*offset[0]/(4*loop));
176      SPI_write(0x1f,-1*offset[1]/(4*loop));
177      SPI_write(0x20,-1*(offset[2]-256)/(4*loop));
178  }
```

▲圖 1.4

在 SPI_Read 函示中，首先在第 56 行把訊號線拉 low 代表要開始傳輸，第 59 行寫入 ADXL 的 register address，做 addr|0x80 的運算是因為最高位加上去代表是要讀取，接著就很簡地依照流程圖完成資料傳輸，最後並在 70 行 ss 拉 high，回傳讀到的資料。

R/W	MB	A5	A4	A3	A2	A1	A0

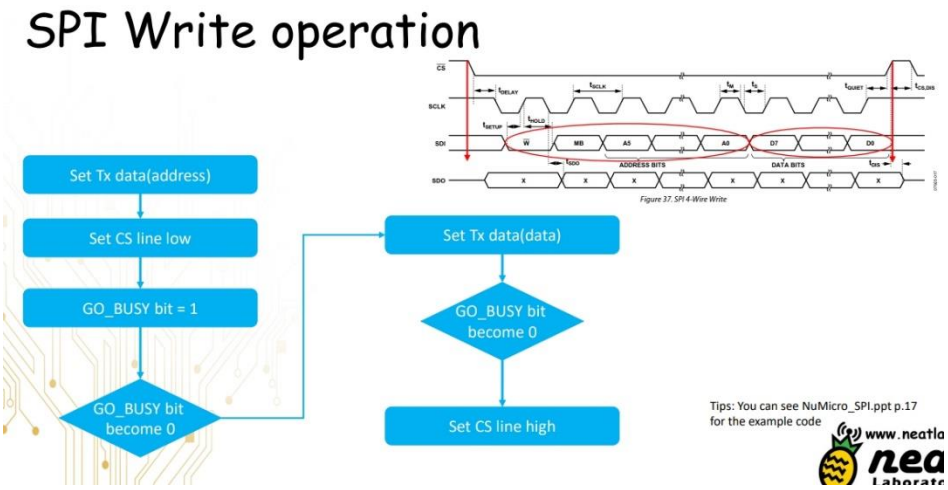
```
53  uint8_t SPI_Read(uint8_t addr)
54  {
55      uint8_t ReadData;
56      SPI_SET_SS0_LOW(SPI2);
57
58      //SPI2->SSR |= 0x1;
59      SPI_WRITE_TX0(SPI2,addr|0x80);
60      //SPI2->TX[0]=(addr|0x80);
61
62      SPI_TRIGGER(SPI2);
63      while(SPI_IS_BUSY(SPI2));
64
65      SPI_TRIGGER(SPI2); //set Go_Busy bit = 1
66      while(SPI_IS_BUSY(SPI2));
67
68      ReadData=SPI_READ_RX0(SPI2);
69      SPI_ClearRxFIFO(SPI2);
70      SPI_SET_SS0_HIGH(SPI2);
71
72      return ReadData;
```



▲圖 1.5

最後為 SPI_Write 的部分，基本上概念相同，在 33 行先拉 low 代表可以進行傳輸，接下來也是依照 SPI Write 的 operation 就可以完成寫入，並在 49 行 ss 拉 high 代表結束。

```
31  void SPI_Write(uint8_t addr,uint8_t data)
32  {
33    SPI_SET_SS0_LOW(SPI2);
34    //SPI2->SSR |= 0x1;
35
36     /* Write to TX register */
37    SPI_WRITE_TX0(SPI2,addr);
38    //SPI2->TX[0]=addr;
39
40    SPI_TRIGGER(SPI2);
41
42    while(SPI_IS_BUSY(SPI2));
43
44    SPI_WRITE_TX0(SPI2,addr);
45    SPI_TRIGGER(SPI2);
46    while(SPI_IS_BUSY(SPI2));
47
48    SPI_ClearTxFIFO(SPI2); //Clear TX FIFO buffer.
49    SPI_SET_SS0_HIGH(SPI2);
50
51  }
52
```



▲圖 1.6

<心得與收穫>

最後一次的 LAB7 主題 SPI 相比上次的 LAB6 的 I2C 簡單很多，眼看這學起的 LAB 也已經走到尾聲，不敢置信自己撐到了最後，還剩下最後的 Final_Project 就結束這堂微處理機的課程了，感謝老師這學期的用心與付出，也感謝助教樂意回答我們許多笨問題，我想我之後有機會還會繼續修老師後續開的課程。總之，這堂課真的收穫滿滿，我也將給予極高的評價並推薦給學弟妹們。