微處理機系統與介面技術 LAB 6

系所: 電機系　　　　　　　學號: 609415194　　　　　　姓名: 劉家豪

<實驗器材>

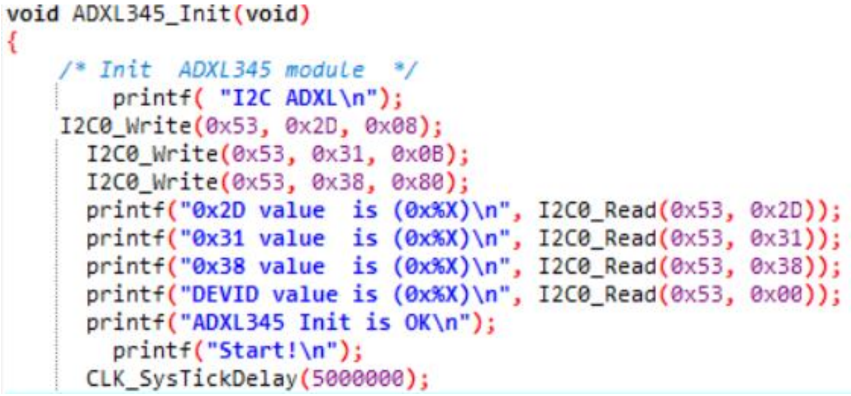
NUC 140 V2.0 開發板、ADXL345

<實驗過程與方法>

這次的 LAB 主要是了解 I2C protocal 的部分。

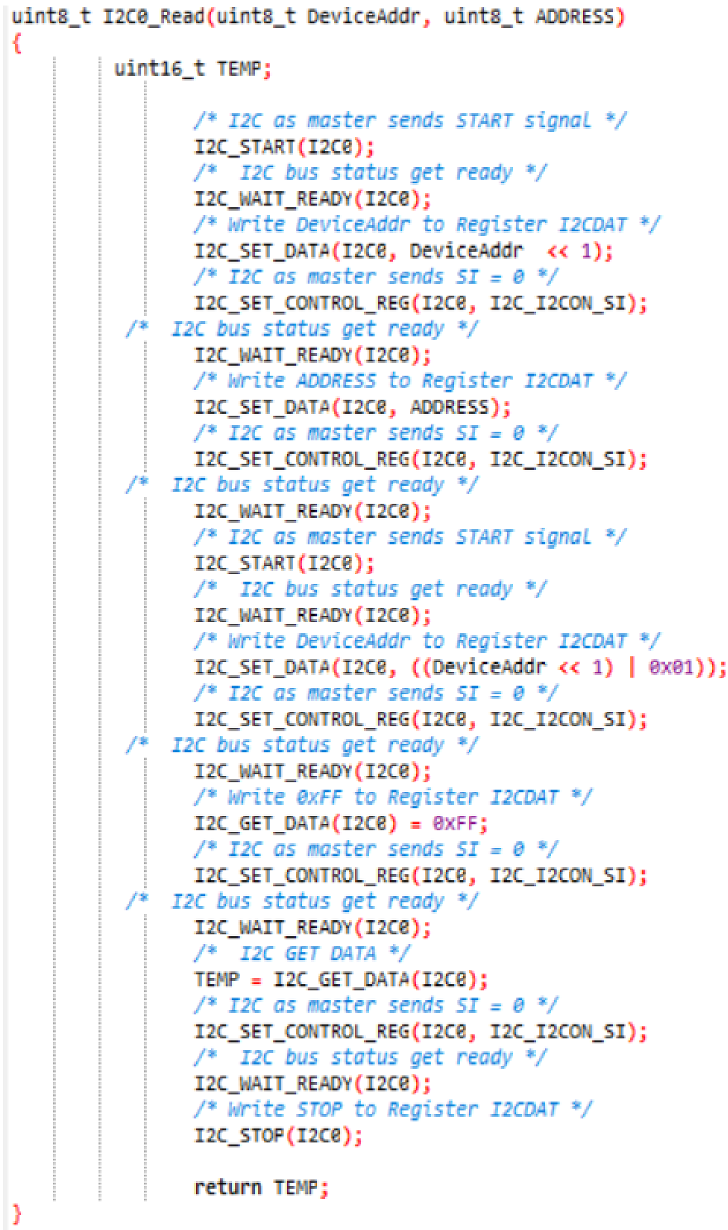
以下這段 code 主要是將 Adxl 做 initial 將助教給的參數寫入特定記憶體。

但寫入的部分需要自己再包一個 function



```
void ADXL345_Init(void)
{
    /* Init  ADXL345 module  */
        printf( "I2C ADXL\n");
    I2C0_Write(0x53, 0x2D, 0x08);
      I2C0_Write(0x53, 0x31, 0x0B);
      I2C0_Write(0x53, 0x38, 0x80);
      printf("0x2D value  is (0x%X)\n", I2C0_Read(0x53, 0x2D));
      printf("0x31 value  is (0x%X)\n", I2C0_Read(0x53, 0x31));
      printf("0x38 value  is (0x%X)\n", I2C0_Read(0x53, 0x38));
      printf("DEVID value is (0x%X)\n", I2C0_Read(0x53, 0x00));
      printf("ADXL345 Init is OK\n");
        printf("Start!\n");
      CLK_SysTickDelay(5000000);
```

adxl 初始化

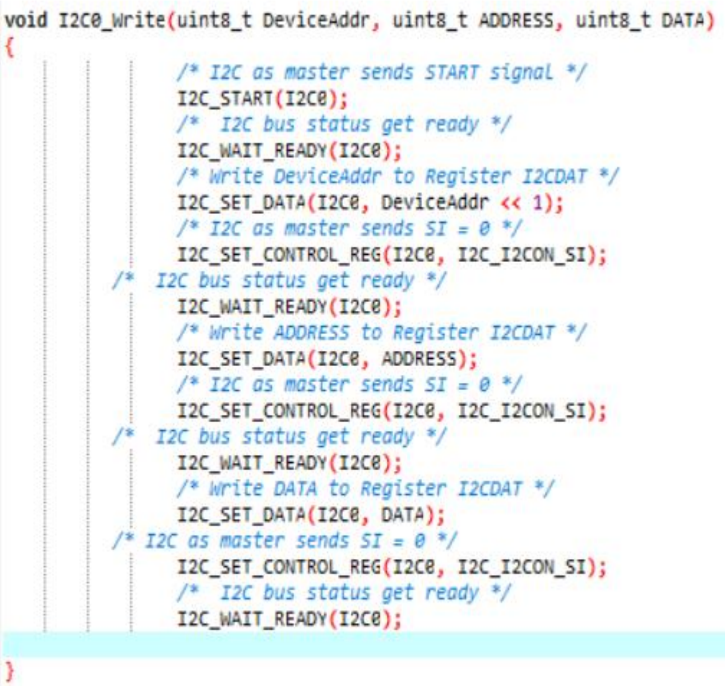


```
uint8_t I2C0_Read(uint8_t DeviceAddr, uint8_t ADDRESS)
{
         uint16_t TEMP;

                /* I2C as master sends START signal */
                I2C_START(I2C0);
                /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write DeviceAddr to Register I2CDAT */
                I2C_SET_DATA(I2C0, DeviceAddr  << 1);
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write ADDRESS to Register I2CDAT */
                I2C_SET_DATA(I2C0, ADDRESS);
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* I2C as master sends START signal */
                I2C_START(I2C0);
                /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write DeviceAddr to Register I2CDAT */
                I2C_SET_DATA(I2C0, ((DeviceAddr << 1) | 0x01));
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write 0xFF to Register I2CDAT */
                I2C_GET_DATA(I2C0) = 0xFF;
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /*  I2C GET DATA */
                TEMP = I2C_GET_DATA(I2C0);
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
                /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write STOP to Register I2CDAT */
                I2C_STOP(I2C0);

                return TEMP;
}
```

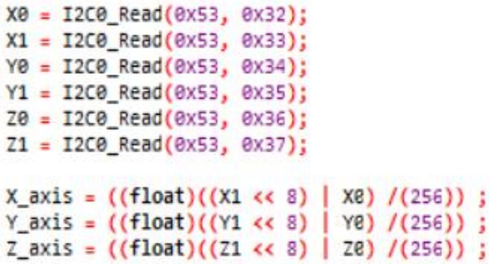
記憶體讀取的 function



```
void I2C0_Write(uint8_t DeviceAddr, uint8_t ADDRESS, uint8_t DATA)
{
                /* I2C as master sends START signal */
                I2C_START(I2C0);
                /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write DeviceAddr to Register I2CDAT */
                I2C_SET_DATA(I2C0, DeviceAddr << 1);
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write ADDRESS to Register I2CDAT */
                I2C_SET_DATA(I2C0, ADDRESS);
                /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
          /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);
                /* Write DATA to Register I2CDAT */
                I2C_SET_DATA(I2C0, DATA);
          /* I2C as master sends SI = 0 */
                I2C_SET_CONTROL_REG(I2C0, I2C_I2CON_SI);
                /*  I2C bus status get ready */
                I2C_WAIT_READY(I2C0);

}
```

記憶體寫入的 function



```
X0 = I2C0_Read(0x53, 0x32);
X1 = I2C0_Read(0x53, 0x33);
Y0 = I2C0_Read(0x53, 0x34);
Y1 = I2C0_Read(0x53, 0x35);
Z0 = I2C0_Read(0x53, 0x36);
Z1 = I2C0_Read(0x53, 0x37);

X_axis = ((float)((X1 << 8) | X0) /(256)) ;
Y_axis = ((float)((Y1 << 8) | Y0) /(256)) ;
Z_axis = ((float)((Z1 << 8) | Z0) /(256)) ;
```

再 main function 讀出 Data 並做 G 值轉換除以 256。

<心得與收穫>

了解如何讀懂 Adxl 的 Datasheet，並且了解 I2C 的傳輸，收穫最多的是了解宣告的記憶體的取捨，但是在讀取資料時有時會遇到 bug，這些 bug 與參數的 bit 數息息相關，與 SPI 不同的是沒有了 SS 的選擇，而是直接寫入我們的 slave advice。