微處理機系統與介面技術 LAB 6

系所：通訊三 學號：409430022 姓名：李宜叡

一、實驗器材

NUC 140 V2.0 開發板



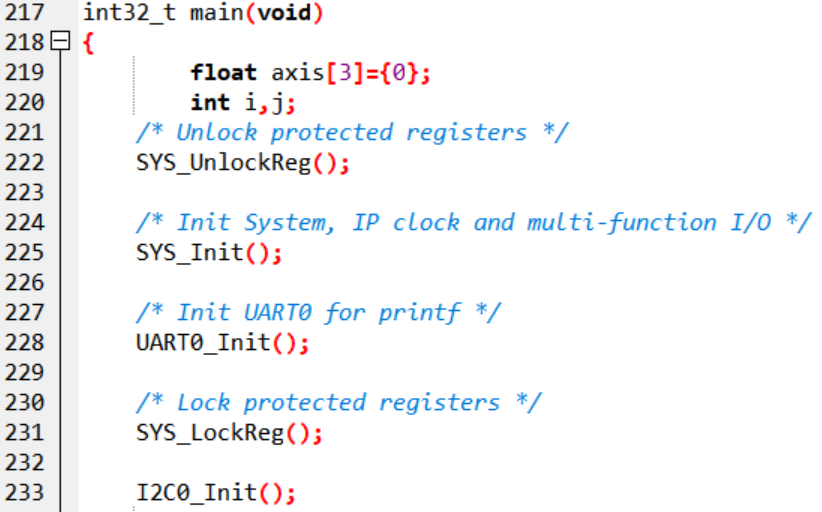
PL2303 USB to UART線

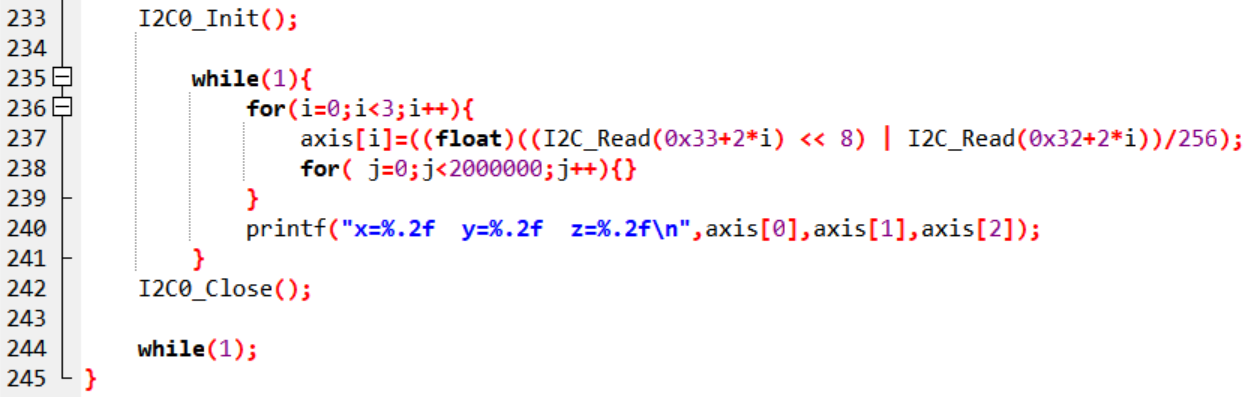


ADXL



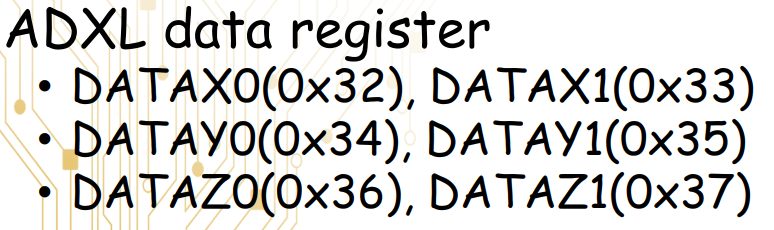
二、實驗過程與方法

這次實驗牽涉到的內容很多，首先從main開始：



Line217~234：前面幾行一樣是初始化的部分。

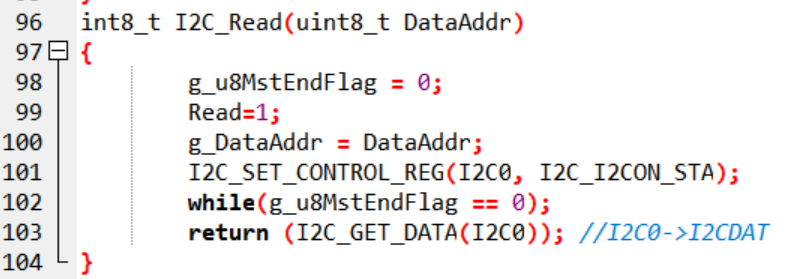
Line235~241：進入一個無限迴圈，不斷重複讀取x、y、z三軸檢測到的的數值，程式中可以看到讀取的行為會先從代表x軸的腳位(0x32、0x33)開始，第二次迴圈(i=1)是讀取y軸、最後一次讀取z軸的數值。



讀取後，做一次延遲，讀完三軸並校正之後印出讀取數值。

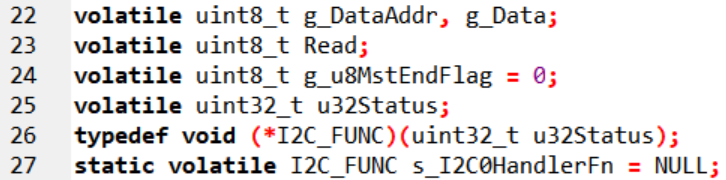
看到這裡，我們順著往前翻I2C\_Read的程式碼：

(1) I2C\_Read



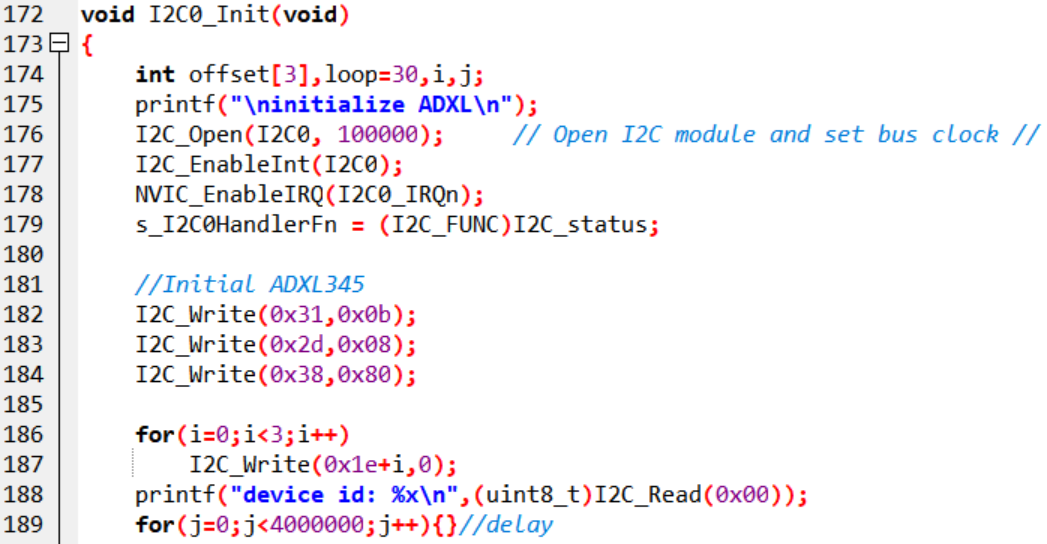
Line96~103：在讀取I2C的程式中，首先可以看到函式接收的數值型別為 unit8\_t，這個就是前面lab常常出現的8 bits資料型態，而在這個函式中所代表的是資料位址。在98行可以見到g\_u8MstEndFlag被設置為0，這個變數代表的是必須終止的旗標，也就是在意料之外的突發狀況發生時，這個旗標就會被設置為1從而讓 bus 直接終止階段。在目前這個函式中只是一般的讀取資料，因此被設置為0。

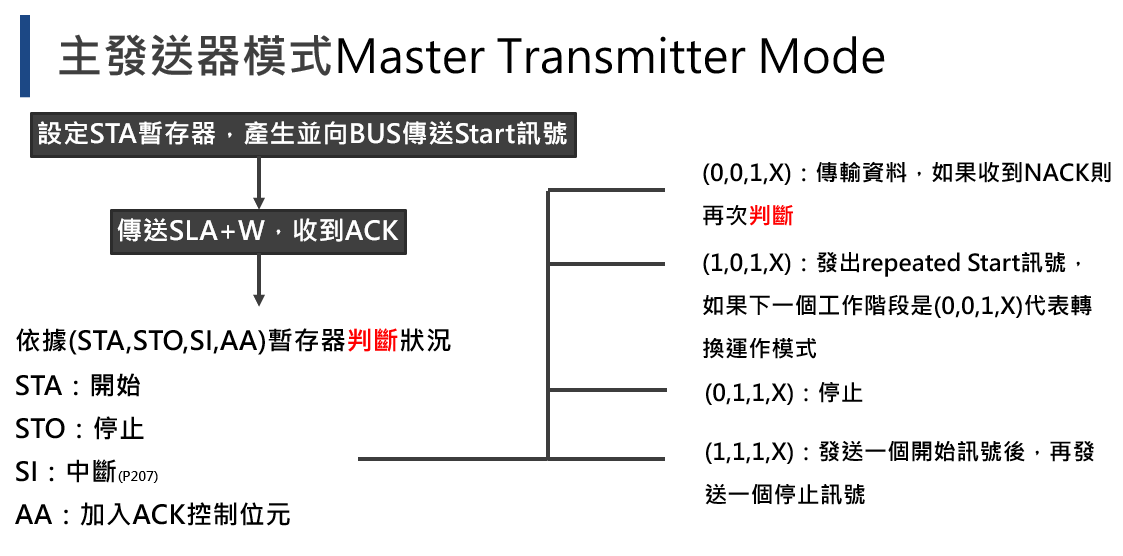
在99行把名為Read的變數設值為1，代表目前的工作情況是讀取，相對地，在寫入函式 I2C\_Write中他就會被設值為0；100行將資料提取出來，這邊可以看到顯然g\_DataAddr變數是全域變數，並且宣告時是設定為volatile的：

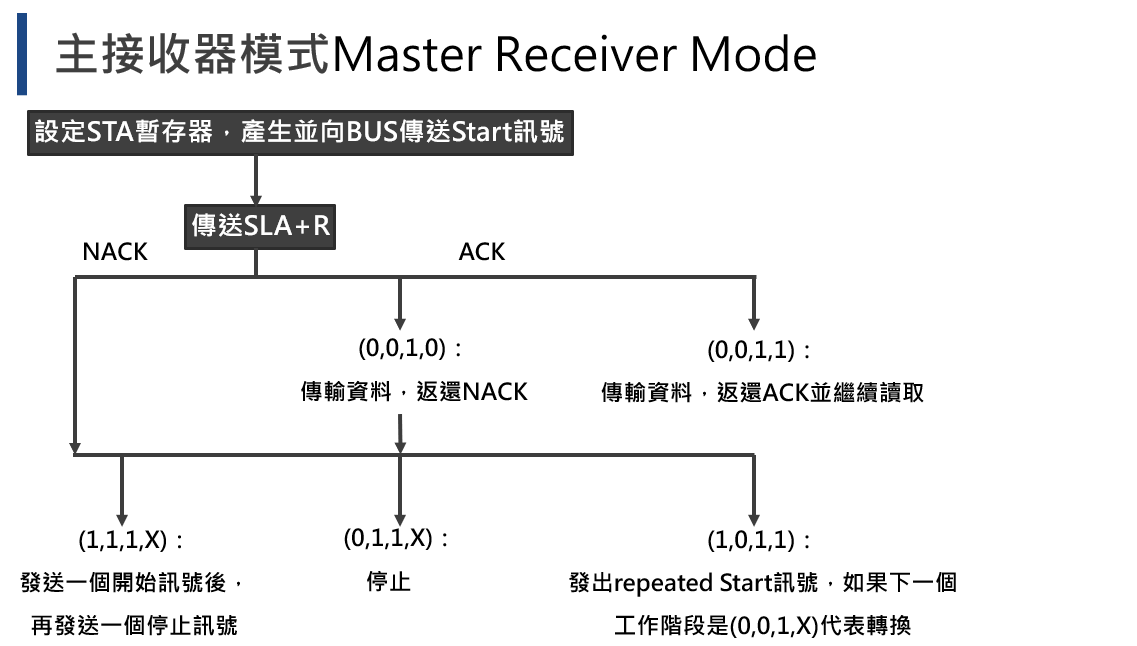


至於101行，在後面會頻繁使用到，這行的意義就是**設定四個暫存器的數值**以表現當前工作狀態，四個分別為(STA,STO,SI,AA)暫存器，對四個的說明如下：STA：開始傳輸通訊資料階段、STO：停止傳輸通訊工作階段、SI：中斷、AA：加入ACK控制位元，換句話說就是以這四個的數值來控制匯流排中積體電路之間的交互行為，這就需要有開始、停止、中斷以及答覆的位元。而後面mustend旗標的數值改變為1的時候會跳出無限迴圈，從而傳回資料。

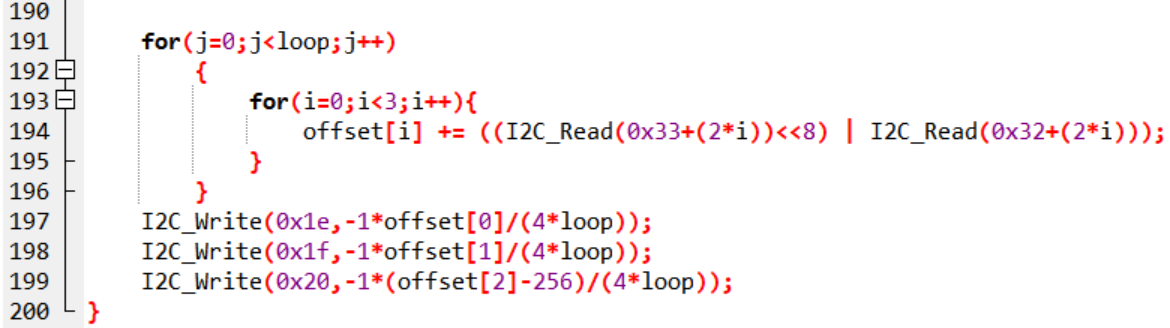
在看完前面main函式與讀取函式之後，接下來檢視i2c的初始化程式碼：

(2) I2C0\_Init

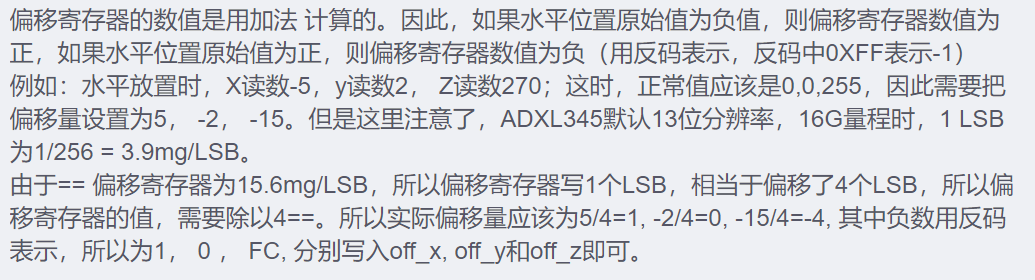
Line174~189：這邊開始了對ADXL的初始化，179行呼叫了下面會討論的I2C\_status函式，這個就是對工作階段採取相應行動的函式，而s\_I2C0HandlerFn在IRQ\_Handler會被呼叫，他被呼叫的時候就會接著呼叫I2C\_status函式，然後就是對當前的工作階段按表操課。至於採取行動的流程圖，就是在手冊中的那幾張很複雜的圖，我在之後上課要上台報告的簡報之中有簡化並大致呈現，如下圖：



礙於上課報告的時長還有必要性，我只有做主機模式，且在現實中，似乎大部分在操控sensor MCU都是以master模式去做操控，部分EEPROM(電子抹除式可複寫唯讀記憶體)才會需要搭配MCU作為slave操作，所以後面的簡報都是簡短說明概念。

回到程式碼，在182行以及之後兩行都呼叫寫入函式，在後面會再詳細審視。186、187行就是在對資料的數值初始化設為0，188印出設備的ID，然後189行延遲一段時間。

Line191~200： loop在前面設值為30，用意是取30次平均，也可以只取10時平均就輸出；在194行讀取了x軸(後面的迴圈中會依序讀取y軸與z軸)的數值，取or運算變成16位元的資料之後存進offset裡面；到了197~199行，執行了寫入程式，這邊要注意的是前面是資料位址(0x1e、0x1f、0x20)，後面首先有個 -1 乘以offset的平均值，至於乘以-1的原因在[某網站](https://blog.csdn.net/acktomas/article/details/101372918)有言：



轉成繁體：*偏移暫存器的數值是用加法計算的。因此，如果水準位置原始值為負值，則偏移暫存器數值為正，如果水準位置原始值為正，則偏移寄存器數值為負（用反碼表示，反碼中0XFF表示-1），****簡單來說就是必須取反才正確。***

*例如：水準放置時，X讀數-5，y讀數2， Z讀數270;這時，正常值應該是0，0，255，因此需要把偏移量設置為5， -2， -15。但是這裡注意了，ADXL345預設13位解析度，16G量程時，1 LSB為1/256 = 3.9mg/LSB。*

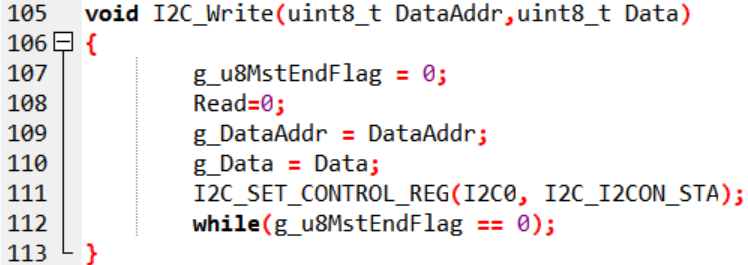
至於為什麼平均要除以四倍loop，那個四倍的原因也有說明：

*由於偏移暫存器為15.6mg/LSB，所以偏移暫存器寫1個LSB，相當於偏移了4個LSB，所以偏移寄存器的值，需要除以4。所以實際偏移量應該為5/4=1，*

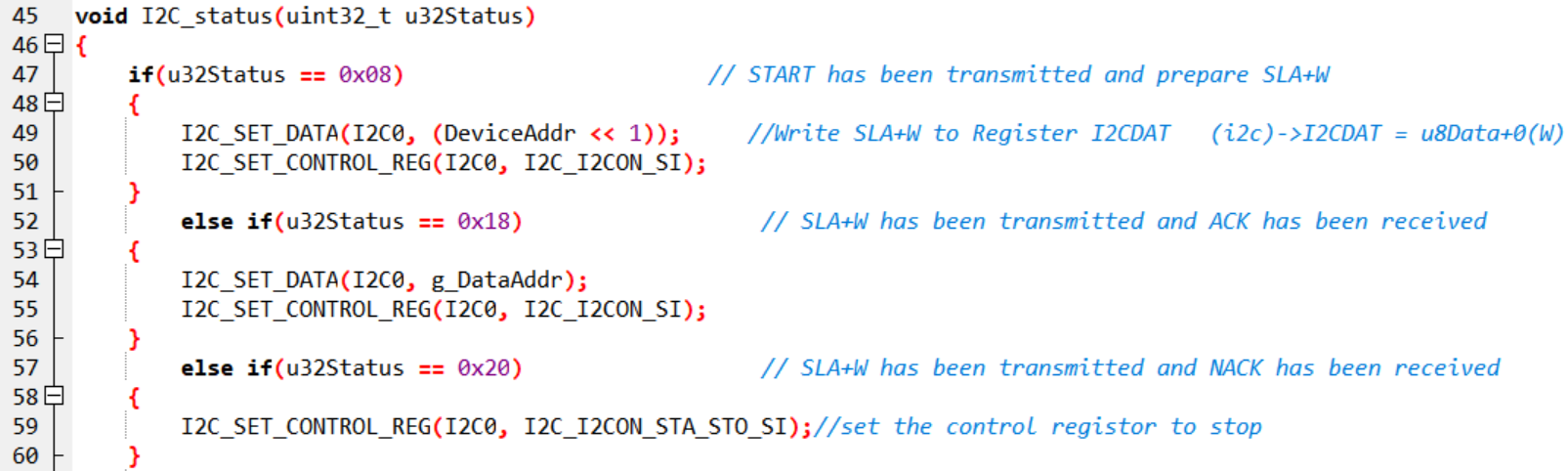
*-2/4=0， -15/4=-4， 其中負數用反碼表示，所以為1， 0 ， FC， 分別寫入off\_x， off\_y和off\_z即可。*

到這裡，看完了I2C0\_Init的內容，下一部分會檢視I2C\_Write函式，然後是I2C\_status函式。

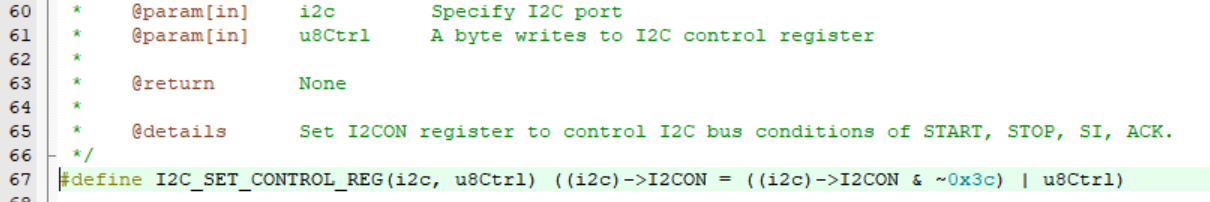
(3) I2C\_Write



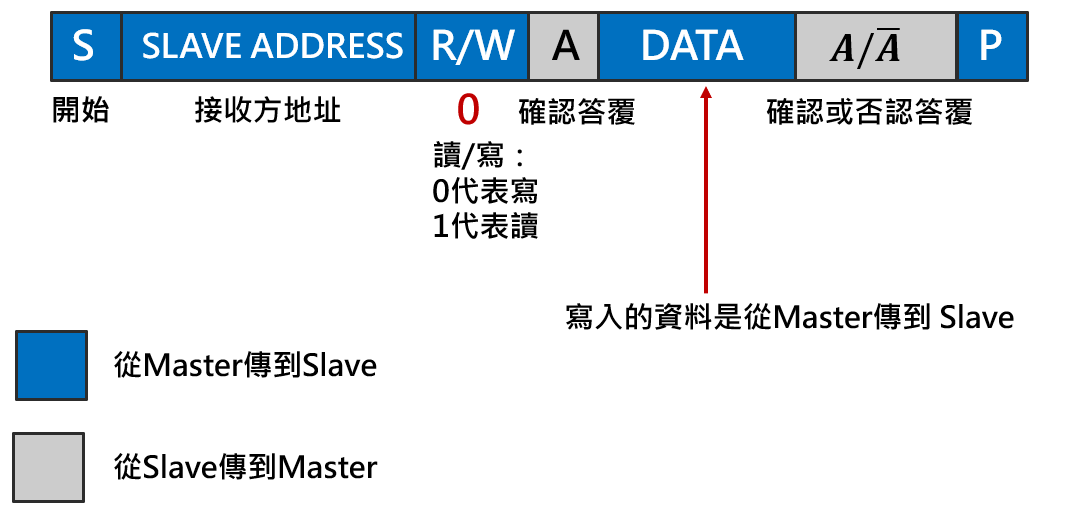
寫入函式與讀取函式有異曲同工之妙，差別在於108行將Read設置為0代表當前工作並非讀取而是寫入。

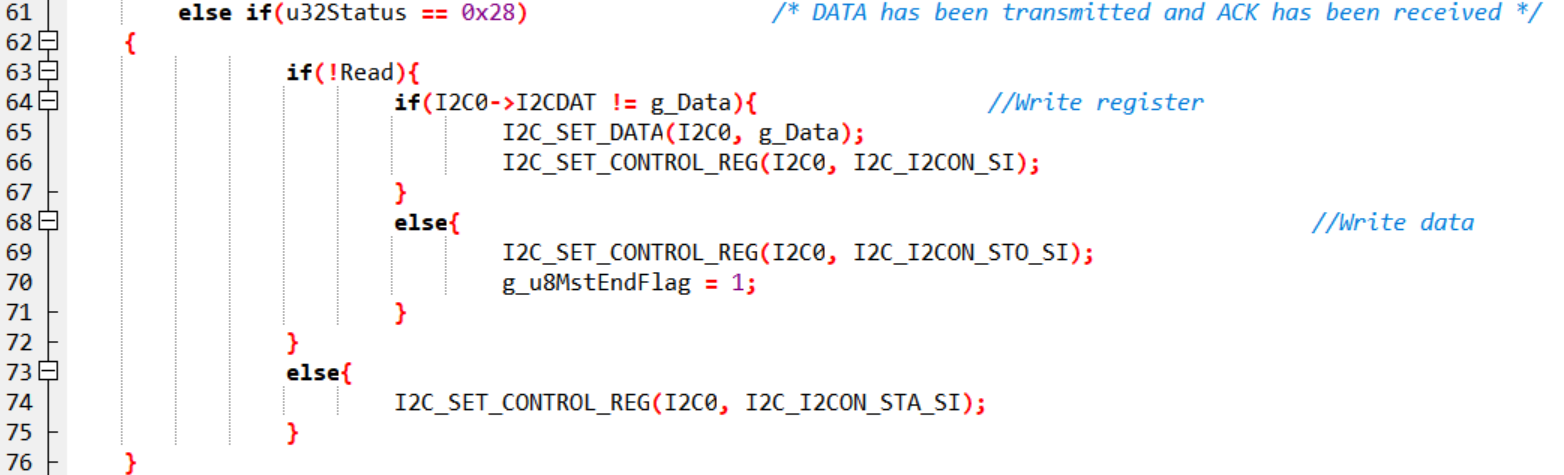
(4) I2C\_status

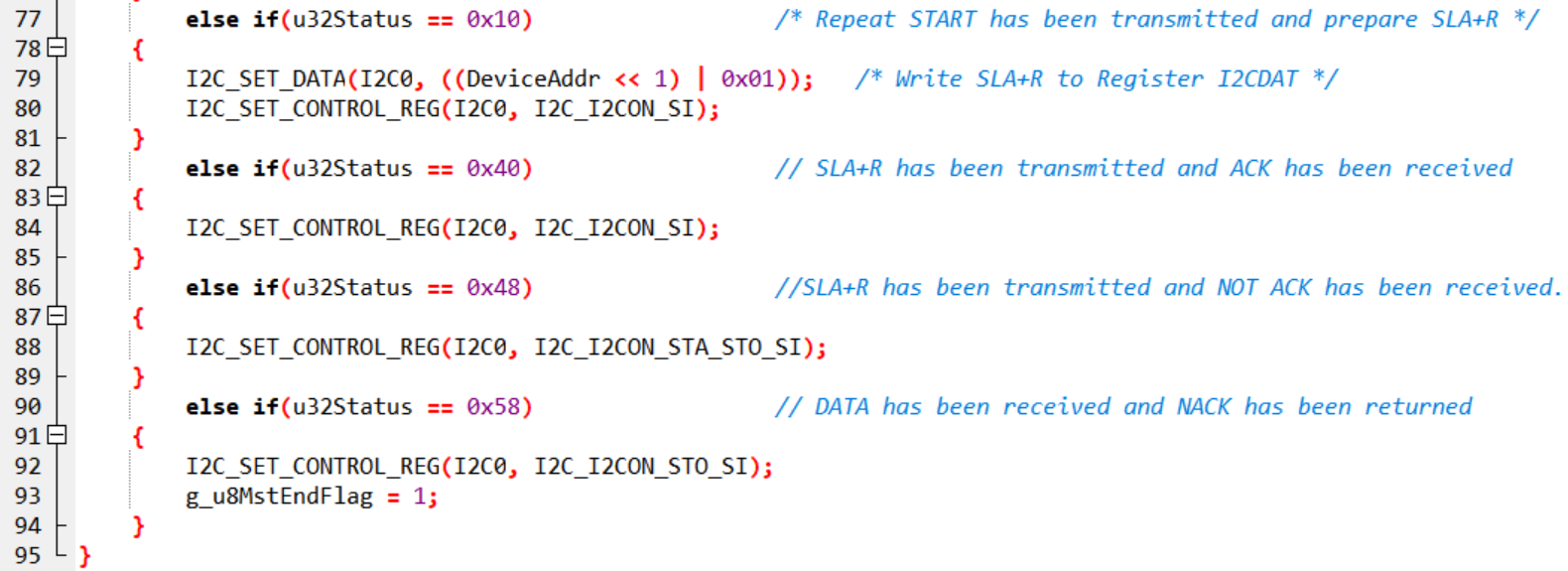
Line45~50：首先此函式考慮的第一個狀況是0x08，也就是開始訊號已經被傳送並且後面跟的8個位元是 slave 地址與W寫入指令，也就是從機寫入地址，此即前面圖片的主發送器模式，接著在第49行就把從機位址寫進資料暫存器，下一行(第50行)更新工作階段，該函式的原始宣告如下：



可以看到這個函式就是更新四個暫存器的數值，這在流程圖中扮演十分重要的角色。

Line51~55：此函式考慮的第二個狀況是0x18，也就是已經傳輸SLA+W且有收到ACK，按照I2C的運作邏輯，在傳輸完位址後，這一步就可以開始傳輸資料，也就是下圖時序中藍色開始傳輸DATA的步驟。

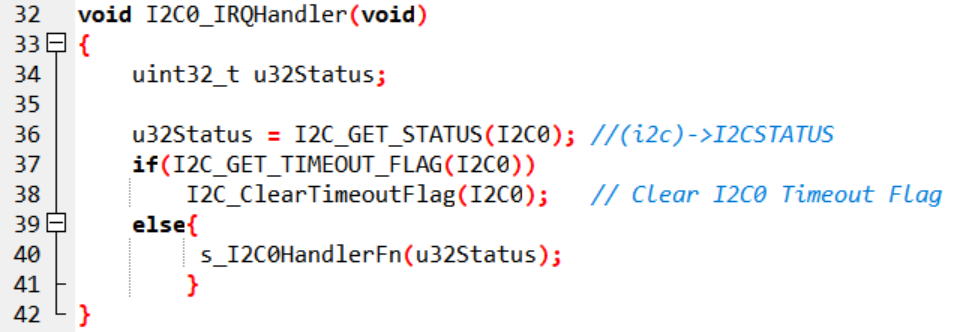
Line56~60：與上一個狀況相反，當收到NACK，代表設備不回應，此時就應該讓整個過程終止，也就是把控制暫存器停止。

Line61~76：這個段落考慮的是0x28的狀況，也就是傳完資料並且收到ACK，這時在第63行首先要藉由Read變數判斷當前是讀取還是寫入，首先如果並非讀取階段，在第64行判斷如果資料暫存器與讀取到要寫入的資料**不同**，則在65行再把資料灌到資料暫存器一次，然後在第66行設置狀態為中斷；否則如果資料暫存器與讀取到要寫入的資料**相同**，就等同已經執行完寫入了，在第69行設定狀態為終止(STO)，並把MustEnd旗標設為1；而在第73行考慮的就是Read=1的情形，代表要讀取，於是在第74行將狀態設定為開始，讓裝置可以開始進行讀取的下一步驟。

Line77~89：這邊是考慮repeat Start的情況，在接收到repeat Start訊號時一般代表的是模式的轉變，例如寫入轉變為讀取，有時即便讀取與寫入對象並沒有改變，但根據規範在開始訊號之後一定都要接8位元的地址。首先是0x10的情況，也就是repeat Start訊號被發出，這時就要把SLA+R寫入資料暫存器裡面，然後才是接後面第82行的0x40也就是有收到ACK的情況。跟前面寫入的步驟類似，如果ACK就中斷、如果NACK就終止。

Line90~95：這裡考慮的是如果收到資料但後面卻是收到NACK，代表見鬼了，需要把狀態暫存器設定為終止(第92行)並且在第93行把MustEnd旗標舉起來。

(5) I2C0\_IRQHandler



Line34~44：一如以往地有IRQ Handler，這邊是當中斷時要抓出當前狀態，並判斷如果是超時中斷就清除超時旗標；如果不是代表遇到了前面Status函式設置的中斷，這時就透過第40行再一次呼叫Status進入判斷流程。

三、個人心得

這次實驗剛好是我們小組要上台報告的內容，所以在打報告時挪用了我自己負責的部分輔助說明。雖然I2C極為節省腳位，但是整體卻是非常複雜，這也造成我的簡報數量很難去精簡，所以如果要上台報告可能就必須適度地快速帶過以平衡時間分配。

手冊中最讓我頭痛的就是那些流程圖，花很長時間看完卻不知道Master、Slave Mode跟GC Mode適用在甚麼情形，直到我上網查才比較清楚功能。在網上有一段生動的比喻：I2C主機和從機的本質區別是看誰能提供SCL信號，主機就像老大，控制時鐘線，從機就像小混混，小混混可以有1個或者多個，每個有固定的編號也就是位址，老大如果要叫小混混打一個人，先發送小混混的編號（位址），小混混確認後，老大再發打人的資料（數據），然後小混混得到數據 ，去執行，這就是主機模式。

小混混如果要叫老大救命（數據），就要先申請，掌控時鐘線，然後按上述類似步驟，這就是從機模式。