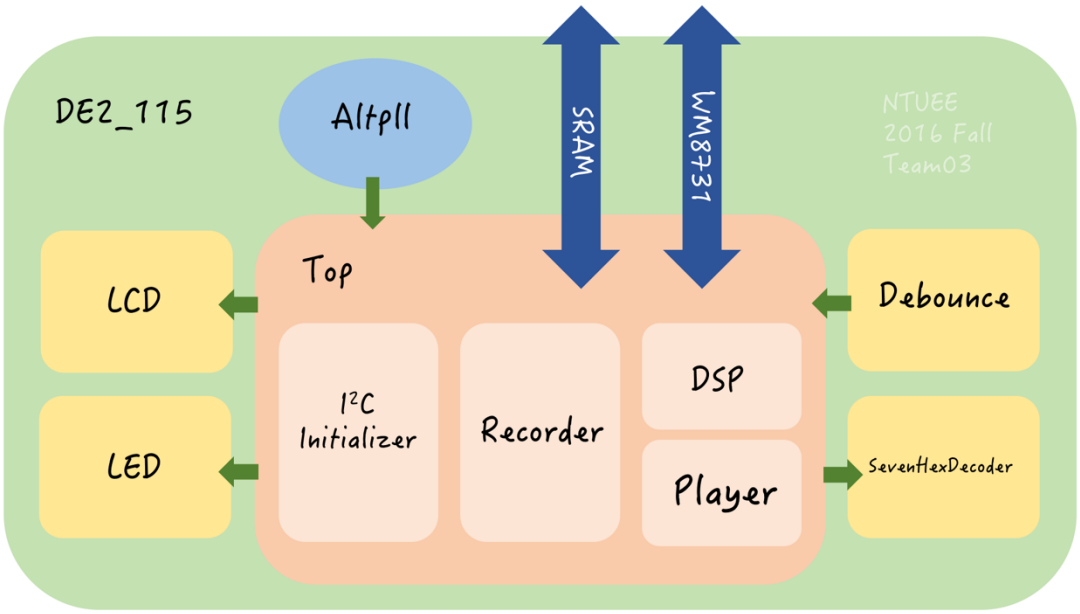
實驗三 教學手冊

team03
b02901121 李孟學　b02901095 鍾杰　b02901010 張捷勝

1 實作架構

本次實驗的程式可以分成若干不同功能的 submodule，詳細架構如下圖：

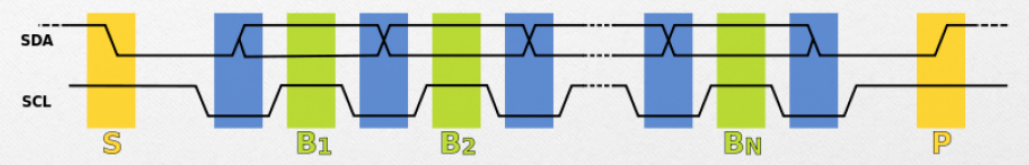


2 WM8731 Initialization

在使用 WM8731 之前，需要利用 I2C Protocol 來初始化，設定 registers。

2.1 I²C Protocol

Inter-Integrated Circuit 使用兩條訊號 Serial Data Line (SDA)及 Serial Clock (SCL)來運作。



Start (yellow)：SCL(1), SDA (1->0)

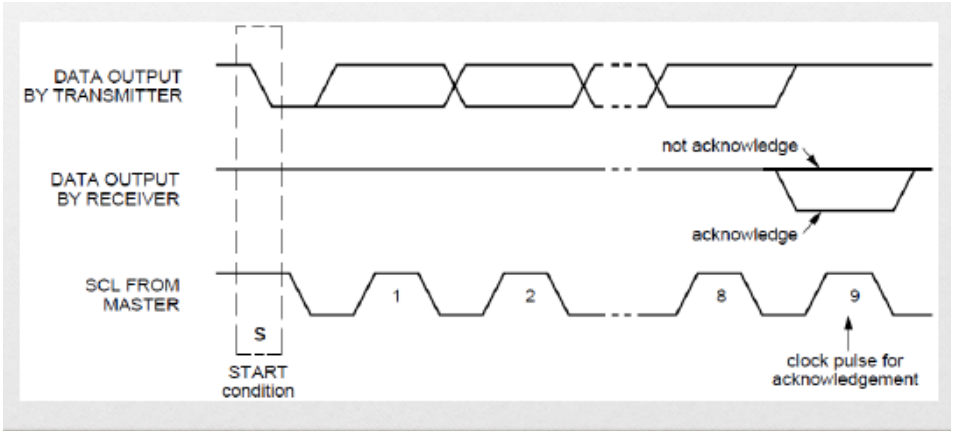
Set Data (blue)：SCL(0), SDA (data)

Transfer Data (green)：SCL(1), SDA (data)

End (yellow)：SCL(1), SDA (0->1)

2.2 Ack Bit

除了上述規則之外，每送出 8 個 bits 之後，還要把 SDA 設成 High Impedance (1'bz)，讓另一端送 Ack 進來。

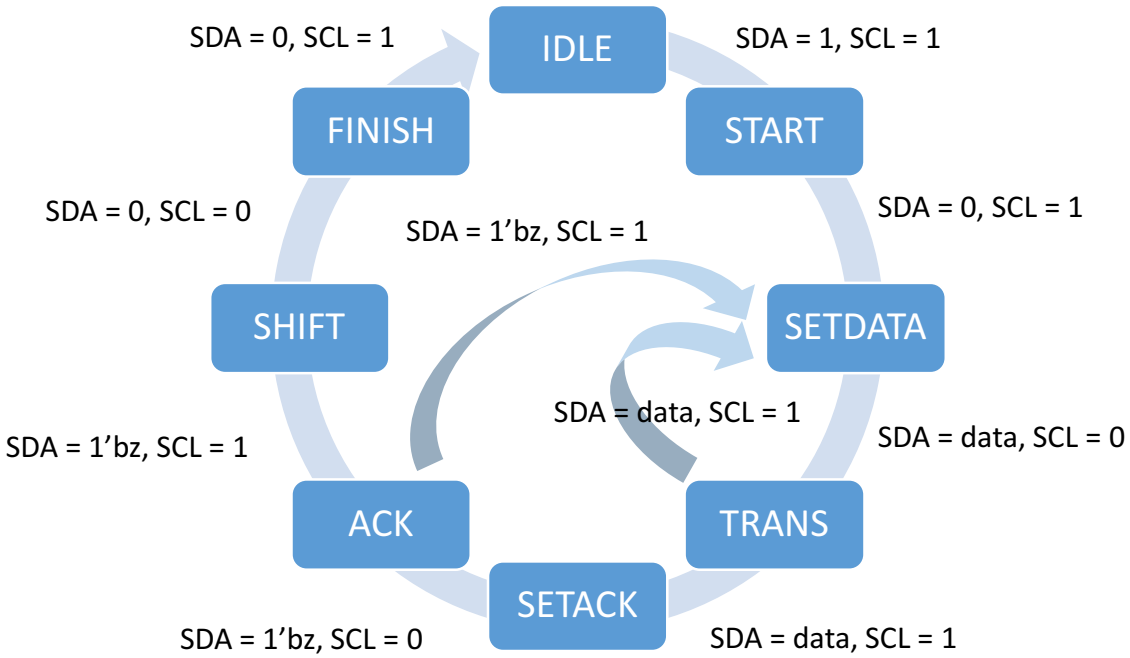


2.3 Recommended Setting

基本上依照助教給的這些設定一條一條傳進去，就能成功讓 WM8731 開始運作了。

Left Line In	000_0000_0_1001_0111
Right Line In	000_0001_0_1001_0111
Left Headphone Out	000_0010_0_0111_1001
Right Headphone Out	000_0011_0_0111_1001
Analogue Audio Path Control	000_0100_0_0001_0101
Digital Audio Path Control	000_0101_0_0000_0000
Power Down Control	000_0110_0_0000_0000
Digital Audio Interface Format	000_0111_0_0100_0010
Sampling Control	000_1000_0_0001_1001
Active Control	000_1001_0_0000_0001

2.4 Finite State Machine



3 LCD

3.1 Initialization

LCD module 開機之後需要一段時間來初始化，並且把需要的設定值(Instructions)傳入。詳細時程可參考助教投影片或使用說明書。

3.2 Module Pin

- ✓ LCD_DATA[0]~[7]：放 Instruction Codes 或 DDRAM/CGRAM Address。
- ✓ LCD_EN：有點類似 I^2C 裡面 SCL 的作用，不過是在 negative edge 時，才讓 LCD 讀資料。
- ✓ LCD_RW：控制讀/寫資料，如果只是要顯示可以直接設為 0(寫)
- ✓ LCD_RS：傳 Instructions 的時候設為 0，傳 Data(CGRAM)的時候設為 1
- ✓ LCD_ON：設為 1 是開機
- ✓ LCD_BLON：設為 0 關 Back Light

3.3 Instructions

詳細資訊可參考投影片中的 Instruction Table，比較需要的基本設定如下：

- ✓ FUNCTION_SET 8'h38

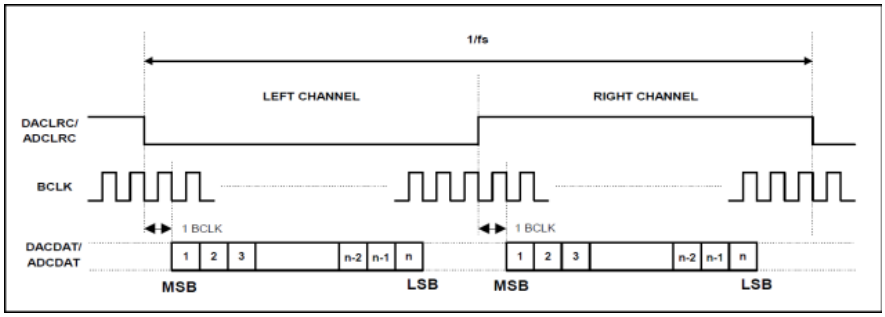
- ✓ DISPLAY_ON 8'h0C
- ✓ CLEAR_DISPLAY 8'h01
- ✓ ENTRY_MODE_SET 8'h06
- ✓ DDRAM_ADDR_SET 8'h80
- ✓ RETURN 8'hC0

4 Recorder

4.1 介紹

Recorder 負責處理錄音的部分，主要的運作就是將音效晶片讀取到的值寫入 SRAM 中。

在這邊要注意的是，音效晶片傳輸資料的方式採用如下圖所示的 I^2S 格式。



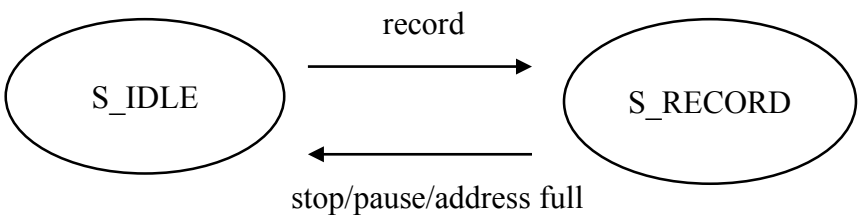
首先，I^2S 在傳輸的過程中會分別傳輸左聲道以及右聲道，並且用 LRC 這個訊號告知現在處於哪一個聲道。但是在錄音時，由於左右聲道是相同的，所以我們只要取一個聲道就可以了。

另外，在每一組傳輸訊號之前都會有一個空的 clock，所以需要經過這個 clock 才能開始記錄傳進來的值。

最後必須要特別注意的是，在每一組資料的 LSB 後面還會有未知數量的空的 clock，因此下一組資料開始的判斷必須藉由 LRC 訊號的幫助完成。

SRAM 的部分主要就是確認控制線沒有出問題就可以了。

4.2 Finite State Machine



這邊雖然寫成只有兩個 state，但是由於 LRC 以及空的 clock 的關係，建議可以用多一點的 state，避免 S_RECORD 容易出現 bug。

4.3 Implementation Details

為了處理資料一開始空的 clock，使用一個變數 pass_one_clock 去紀錄是否已經通過這個 clock。

操作 SRAM 時，並沒有等到全部 16 個 bit 收集完後才寫入 SRAM，而是每一個 clock 都一直讓他寫入，好處是可以不用處理 SRAM 的控制線，但是要小心 address 的位置。

5 Player

5.1 介紹

這部分負責兩件事情：第一件事從 SRAM 中取出一筆資料後根據播放模式與速率進行訊號處理，第二件事情是根據 Recorder 那邊提過的 I^2S 格式將處理完的資料送給音效晶片。實作時可以拆成兩個 submodule 會比較好處理，其中做訊號處理的部分可以使用 DACLRCK 來更新 flip flop，搭配使用 BCLK 的一個 sender 來一個 bit 一個 bit 丟出處理好的資料。

5.2 快速播放

實作方式有兩種：一種是根據速率決定間隔多少 address 來取資料播放，這種方式可能會使實際播放時隨速率增加使原本的聲音聽起來頻率變高；另一種方式是一次先連續播放一段 address 區間的資料，接著再根據速率等比例跳過一大段 address 區間繼續去下一個區間的資料，這種方式在速率高的情況下因為會一次跳過一大段資料不播放，可能聽起來會比較明顯有一點像是一小段一小段在跳的感覺。

5.3 慢速播放

慢速播放分成一次內差和線性內差。一次內差比較簡單，只要在兩筆資料間持續丟出第一筆資料就好；線性內差就會需要同時存有連續兩筆資料，才能計算兩筆資料間的內差數值，這部分要特別注意要把資料看作是 2's complement 來處理，也要記得處理運算中可能出現的 overflow 問題。

6 Altpll

這個部分可以用 Qsys 直接合成出來，設計成輸入 50MHz 的 clock 後同時輸出另外幾種頻率的 clock。可能會需要的是給 I^2S initialize 用的

100kHz、給 LCD 用的 800kHz 跟其他部分用的 12MHz。

7 其他建議

這個實驗要盡量把每個部分都分成一個 submodule 來處理，會比較容易 debug。每個 submodule 完成後最好也要自己寫個簡單的 test bench 測試一下 IO 的運作情況是否如預期，如此一來之後全部合起來一起跑的時候才比較不會出錯。