微處理機系統實習　　Lab8

班級：資訊三丁 學號：D1009212 姓名：邱柏宇

一、【實驗目的】：

What was your design? What were the concepts you have used for your design?

8.1

建立一個 node_index 的變數，來記錄歌播放到哪裡，並且小於 500 時關閉 PWM 輸出，大於等於 500 且小於 4000 就把 PWM 的 Frequncy 設定為 PWM 輸出的數值，這裡統一把 DutyCycle 都設為 90，剛好聲音是好聽的，當大於等於 4000 時把 Frequncy 改設為歌曲的音符頻率，然後看是不是 0 來決定要不要關掉 PWM 輸出，最後加上 pitch 的 delay，這裡一樣 mod 72 來做到重複播放。

```
if (u32ADCvalue < 500) {
    PWM_DisableOutput(PWM1, PWM_CH_0_MASK);
    node_index = 0;
} else if (u32ADCvalue >= 500 && u32ADCvalue < 4000) {
    PWM_ConfigOutputChannel(PWM1, PWM_CH0, u32ADCvalue, 90);
    PWM_EnableOutput(PWM1, PWM_CH_0_MASK);
    node_index = 0;
} else if (u32ADCvalue >= 4000) {
    PWM_ConfigOutputChannel(PWM1, PWM_CH0, music[node_index], 90);  // 0=Buzzer ON
    if (music[node_index] != 0)
        PWM_EnableOutput(PWM1, PWM_CH_0_MASK);
    else
        PWM_DisableOutput(PWM1, PWM_CH_0_MASK);
    CLK_SysTickDelay(pitch[node_index]);
    node_index++;
    node_index %= 72;
}
```

二、【遭遇的問題】：

What problems you faced during design and implementation?

Sample 中提供的譜快慢有些不合理，老師不是很滿意。

三、【解決方法】：

How did you solve the problems?

將 pitch 調成統一長度即可

```
uint32_t pitch[72] = {
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms,
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms,
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms,
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms,
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms,
    P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms, P500ms};
```

四、【未能解決的問題】：

Was there any problem that you were unable to solve? Why was it unsolvable?

有嘗試使用 timer 來處理 pitch 的 delay，不過這樣需要多開一個 timer，感覺沒必要，所以在想能不能用 ADC 的中斷來當作 timer，因為 ADC 也是有 clock 的，不過最後結果不是很好，因為 ADC 轉換那邊是用確認轉換訊號，速度太快不太適合拿來當作 timer。

（還有下一頁）

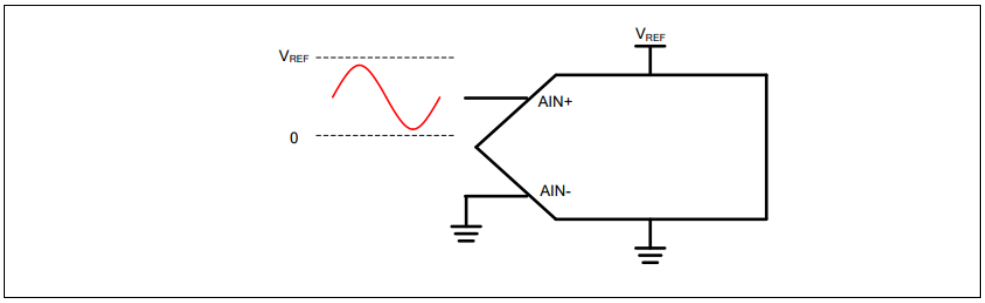
五、【投影片的問題】：

SINGLE_END, DIFFERENTIAL 以及 SINGLE, SINGLE_CYCLE, CONTINUOUS 是甚麼意思，並且有甚麼差別？

首先 ADC 中的 SINGLE_END 和 DIFFERENTIAL 是兩種不同的輸入模式。SINGLE_END 模式也可以稱作「單端輸入」，只需要一個輸入信號。DIFFERENTIAL 模式也可以稱作「全差分輸入」，需要兩個輸入信號。兩者的偵測方式可以參考下面技術手冊的資料。

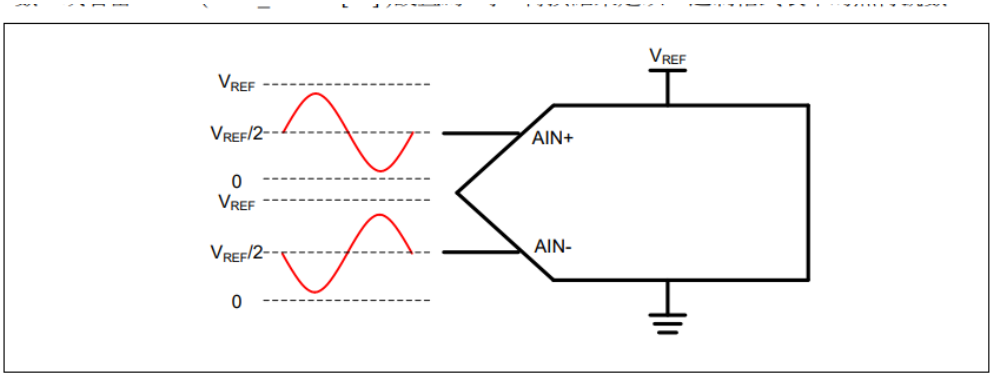
SINGLE_END 模式

ADC 將輸入信號直接輸入到 ADC 的模擬輸入端。ADC 輸入一個信號和一個固定的基準電壓，在 NUC140 中的基準電壓也就是 GND，然後進行比較後輸出一個數位信號，但是如果只要出現信號噪音，就會沒有辦法抵銷，所以不是很精準。



DIFFERENTIAL 模式

在 DIFFERENTIAL 模式下，ADC 將輸入信號分為兩個相等的部分，並且兩個訊號會是 180 度的反向信號，然後將兩個進行比較輸出一個數位信號，是兩者的差值，這樣的優點是如果有信號噪音，會剛好相互抵消掉，所以較為精準。



NUV140 因為只有 8 個通道，所以最多使用 8 個 SINGLE_END 或 4 個 DIFFERENTIAL。

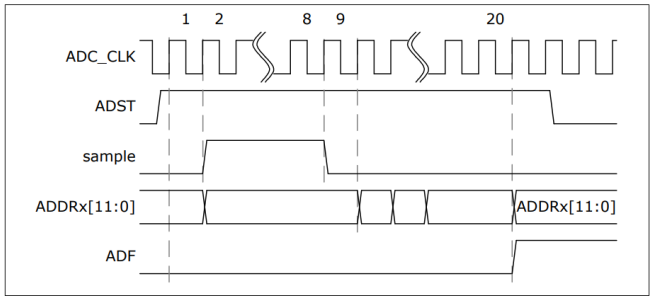
（還有下一頁）

SINGLE, SINGLE_CYCLE, CONTINUOUS 是 ADC 的三種操作模式。

SINGLE

只對指定通道進行一次 A/D 轉換。

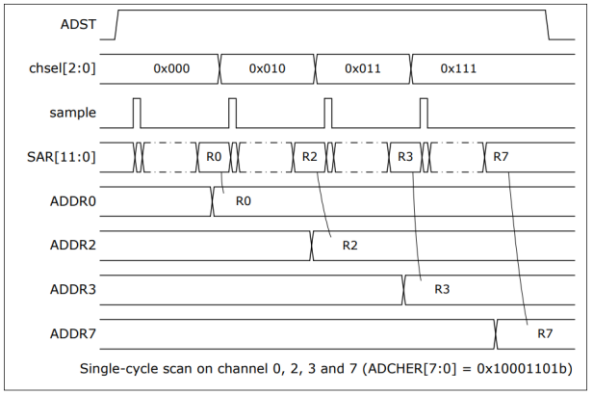
首先 ADCR 的 ADST 會被設定為 1，可以是軟體或是硬體中斷設定，A/D 轉換為會被存到 A/D 資料暫存器對應的頻道，ADSR 的 ADF 會被設定為 1，接下來如果 ADCR 的 ADIE 被設定為 1，ADC 的中斷就會被置換出來，技術手冊中提到比較重點的是 ADST 必須要在結束轉換的時候間隔 ADST 一個 ADC_CLK 週期才能設定為 0，不然可能會讓 ADC 運作不正常。



SINGLE_CYCLE

對所有指定通道進行一個 A/D 轉換週期

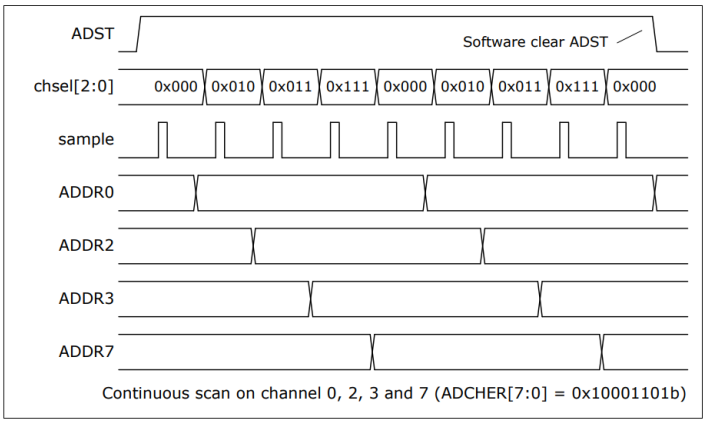
按照從編號最低的通道到編號最高的通道的順序，跟上面的很像，不過可以想成可以一次 ADST 中可以轉換多個通道，但 ADST 還是會在結束時設為 0。



CONTINUOUS

這是一個連續掃描的模式，從上面兩種模式可以得知，開始轉換都是依照 ADST 是否為 01，可以連續掃描就是因為這個模式中 ADST 不會自動變成 0，所以就可以保持著上面提到的轉換流程和放到每個通道的暫存器裡面。

（還有下一頁）



那這次 sample code 中為甚麼會是用 SINGLE_CYCLE 而不是 CONTINUOUS，8.1 題目要求一直讀取可變電阻，來決定現在播放的模式不是嗎？

其實兩者都可以做到不斷的掃瞄，重點就在能不能一直把 ADST 保持在 1，換句話說就是 SINGLE_SYCLE 只要把 ADST 固定都會設為 1 來進行下一次轉換，就可以做到也是連續轉換了。

可以來測試一下，首先 trace code 可以發現到，在 adc.h 中有提供 ADC_START_CONV，也就是把 ADST 設為 1 的 function。

```
/**
  * @brief Start the A/D conversion.
  * @param[in] adc The pointer of the specified ADC module.
  * @return None
  * @details ADST (ADCR[11]) can be set to 1 from three sources: software, PWM Center-aligned trigger and external pin STADC.
  */
#define ADC_START_CONV(adc) ((adc)->ADCR |= ADC_ADCR_ADST_Msk)
```

```
478   #define ADC_ADCR_ADST_Pos       11                          /*!< ADC_T::ADCR: ADST Position */
479   #define ADC_ADCR_ADST_Msk       (1ul << ADC_ADCR_ADST_Pos)  /*!< ADC_T::ADCR: ADST Mask */
480
```

測試 SINGLE_CYCLE 模式，分別在一直 ADST 設為 1 的情況能不能做到連續掃描。

```
while (TRUE) {
    ADC_START_CONV(ADC);
```

```
ADC_START_CONV(ADC);
while (TRUE) {
```

左邊可以做到連續掃描，而右邊就只有在程式一開始有掃描一次而已。

要讓右邊也可以做到連續掃描的話，就需要設定為 CONTINUOUS 模式。

```
#define ADC_INPUT_MODE ADC_INPUT_MODE_SINGLE_END
#define ADC_OPERATION_MODE ADC_OPERATION_MODE_CONTINUOUS
```