姓名: 徐華佑　　學號: R11921018　　嵌入式系統 HW6

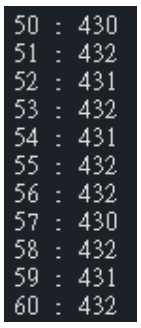
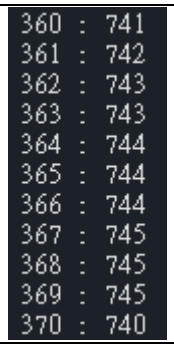
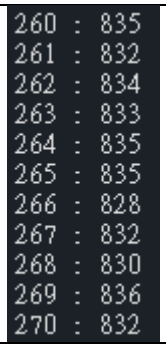
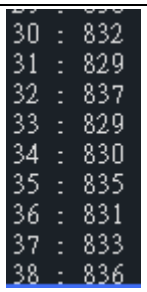
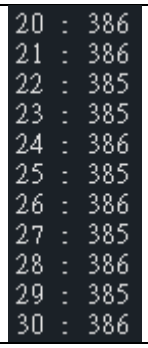
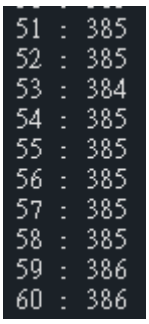
Data Acquisition, DMA, and Interrupt

GitHub：
https://github.com/HenryShyuHui/eb_proj6.git

1. We provide an example code in "timer_trigger_adc" folder to setup ADC triggered by timer, sampling the signal of STM32 internal temperature sensor at a fixed frequency and generating an interrupt when each conversion finishes. Please modify the (a) trigger frequency and (b) sampling time of ADC.

(a) Modify trigger frequency

由於 timer 選擇的時鐘源為 PLL，產生的震盪頻率為 80MHz，在這範例程式中選擇 TIM1 來輸出。prescaler 的用途在於，將原本 TIM1 的訊號進行分頻，先將頻濾下降，period 則是採樣的個數，如: prescalar = (4000-1)，period = (1000-1)，則可以想成，先將 TIM1 預分頻成 20kHz (80MHz/4000)，從 0 數至 999 共需花 50ms，所以採樣頻率為 80Hz。故可以利用透過調整 prescalar 和 period 的數值，達到調整 trigger frequency 的目的。

						
Prescaler	4000-1	400-1	40-1	400-1	40000-1	40000-1
Period	1000-1	1000-1	1000-1	100-1	1000-1	2000-1
Sampling rate	20Hz	200Hz	2000Hz	2000Hz	2Hz	1Hz

在最右邊的圖可以看到 sampling rate 為 1Hz，實際上在 terminal 上看到的數值更新速度也為一秒一次，所以可觀察到這 code 會在每一個 sampling point 印出資訊，印出的數值之平均值隨著頻率的 sampling rate 的上升而增加，並不會受到 perscalar 和 period 的個別數值有影響。

(b) Modify sampling time of ADC

調整 ADC sampling time2k7，表 ADC 在進行採樣時需要經過幾個 clock cycle。使用

```
#define ADC_SAMPLETIME_2CYCLES_5          (LL_ADC_SAMPLINGTIME_2CYCLES_5)    /*!< Sampling time 2.5 ADC clock cycles */
#define ADC_SAMPLETIME_6CYCLES_5          (LL_ADC_SAMPLINGTIME_6CYCLES_5)    /*!< Sampling time 6.5 ADC clock cycles */
#define ADC_SAMPLETIME_12CYCLES_5         (LL_ADC_SAMPLINGTIME_12CYCLES_5)   /*!< Sampling time 12.5 ADC clock cycles */
#define ADC_SAMPLETIME_24CYCLES_5         (LL_ADC_SAMPLINGTIME_24CYCLES_5)   /*!< Sampling time 24.5 ADC clock cycles */
#define ADC_SAMPLETIME_47CYCLES_5         (LL_ADC_SAMPLINGTIME_47CYCLES_5)   /*!< Sampling time 47.5 ADC clock cycles */
#define ADC_SAMPLETIME_92CYCLES_5         (LL_ADC_SAMPLINGTIME_92CYCLES_5)   /*!< Sampling time 92.5 ADC clock cycles */
#define ADC_SAMPLETIME_247CYCLES_5        (LL_ADC_SAMPLINGTIME_247CYCLES_5)  /*!< Sampling time 247.5 ADC clock cycles */
#define ADC_SAMPLETIME_640CYCLES_5        (LL_ADC_SAMPLINGTIME_640CYCLES_5)  /*!< Sampling time 640.5 ADC clock cycles */
```

可知道可以調整成 2.5, 6.5, 12.5, 24.5, 47.5, 92.5, 247.5, 640.5 個 clock cycle。

```
sConfig.Channel = ADC_CHANNEL_TEMPSENSOR;
sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;//ADC_SAMPLETIME_247CYCLES_5;
sConfig.SingleDiff = ADC_SINGLE_ENDED;
sConfig.OffsetNumber = ADC_OFFSET_NONE;
sConfig.Offset = 0;
```

使用 Prescaler = 4000-1，period = 1000-1 進行測試

Sampling time	2.5	12.5	24.5	92.5	640.5
	161 : 395 162 : 395 163 : 395 164 : 395 165 : 395 166 : 396 167 : 396 168 : 395 169 : 396 170 : 396	10 : 1043 11 : 1043 12 : 1044 13 : 1044 14 : 1044 15 : 1045 16 : 1043 17 : 1043 18 : 1045 19 : 1044 20 : 1044	70 : 975 71 : 976 72 : 976 73 : 975 74 : 976 75 : 975 76 : 975 77 : 975 78 : 975 79 : 975	70 : 939 71 : 939 72 : 939 73 : 938 74 : 938 75 : 940 76 : 939 77 : 938 78 : 938 79 : 938 80 : 939	50 : 938 51 : 940 52 : 939 53 : 939 54 : 940 55 : 939 56 : 939 57 : 941 58 : 939 59 : 938 60 : 939

Ref:

https://blog.csdn.net/Naisu_kun/article/details/121532288

https://blog.csdn.net/lwb450921/article/details/125708831

https://www.st.com/content/ccc/resource/training/technical/product_training/group1/b9/b6/ca/ca/9f/95/4e/62/stm32mp13-analog-adc/files/stm32mp13-analog-adc.pdf/jcr:content/translations/en.stm32mp13-analog-adc.pdf

2. **We provide an example code in “timer_trigger_adc_DMA_ver” folder to setup ADC triggered by timer, sampling the signal of STM32 internal temperature sensor at a fixed frequency. Further, the code setups DMA to transfer the data from ADC data register to a specific buffer when each conversion finishes. When the top half of buffer is filled, the interrupt will be generated and print all data in the top half of buffer. When the bottom half of buffer is filled, the interrupt makes all data in the bottom half of buffer printed. We left some “to do”s in the code, please finish them.**

根據題目要求要實作 DMA，將 buffer 分成兩半，當其中一半滿時將資料轉移並印出。使用兩個 buffer 來存取資料(data)，先宣告一些變數，如 flag、buffer 及 data (題目原本有給 sample_buffer，這個實作會使用 data 來做代替)，設定 buffer 為 100，有兩個 buffer，所以 data 的大小為 200 = length。

```
13   bool flag = 0;
14   volatile int counter = 0;
15   uint32_t length = 200;
16   uint16_t buffer[100] = {0};
17   uint16_t data[200] = {0};
```

在這兩個 todo 的函式中，buffer 會去存取現在的 data 值，當 buffer 被填滿時，event_queue 會 call timer_count_callback 的函式。而這兩個 todo 的函式是從 main 中的程式在執行時會被 called。

```
138  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
139  {
140      //to do ..........................
141      flag = 1;
142      for(int i=100; i<200;i++) {
143          buffer[i-100] = data[i];
144      }
145      event_queue.call(timer_count_callback);
146  }
147
148  void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef *hadc)
149  {
150      //to do ..........................
151      flag = 1;
152      for(int i=0; i<100;i++) {
153          buffer[i] = data[i];
154      }
155      event_queue.call(timer_count_callback);
156  }
```

當 timer_count_callback 被呼叫的時候，會產生 interrupt 並將 buffer 中的值印出。

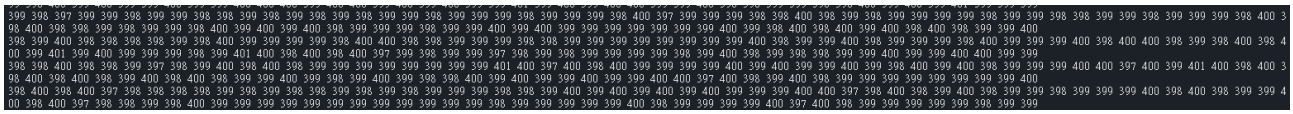
```
122  void timer_count_callback()
123  {
124      // uint16_t val = HAL_ADC_GetValue(&hadc1);
125      // printf("%d : %d\n", counter++, val);
126      if (flag==1) {
127          flag = 0;
128
129          for (int i=0; i<100; i++) {
130              printf("%d ", buffer[i]);
131              // printf("%d : %d ", i, buffer[i]);
132          }
133          printf("\n");
134      }
135  }
```

而在 main 函式中，要先將以下參數傳入 start 中。且在開始之後會由 start_IT 對上圖 HAL 等函式進行呼叫。

```
int main()
{
    //HAL_Init();
    //SystemClock_Config();

    TIM1_Init();
    ADC1_DMA1CH1_init();

    HAL_ADC_Start_DMA(&hadc1, (uint32_t*)data, length);
    HAL_TIM_Base_Start_IT(&htim1);
    event_queue.dispatch_forever();

    return 0;
}
```

Result:

由於 prescalar = 4000-1，period = 1000-1，所以 sampling rate 如同第一題為 20Hz，在輸出 100 筆資料所花的時間為 100/20 = 5s，每次的 print 會有 5s 的間隔，且實際上在觀察時也和理論相同。



3. **Here is an MbedOS microphone example which can record the sound in two second(https://github.com/janjongboom/b-l475e-iot01a-audio-mbed). Please modify the code such that the program executes continuously and periodically (You don't need to preserve audio data or print it out). This code setups DMA to transfer audio data to PCM_Buffer. When the top and bottom half of PCM_Buffer are filled, the corresponding interrupts will generate. Please choose two GPIO pins as output and connect them to logic analyzer. Once the PCM_Buffer top half event occurs, toggle pin1's output voltage. Once the PCM_Buffer bottom half event occurs, toggle pin2's output voltage. By this way, you can observe the frequency of audio sampling.**

1. 將 main.cpp 的第 39 行 wav_header 宣告方式改成 unsigned，使程式順利編譯。
2. 宣告兩個輸出 GPIO 腳位 out 及 out2，讓訊號線可以連接邏輯分析儀。

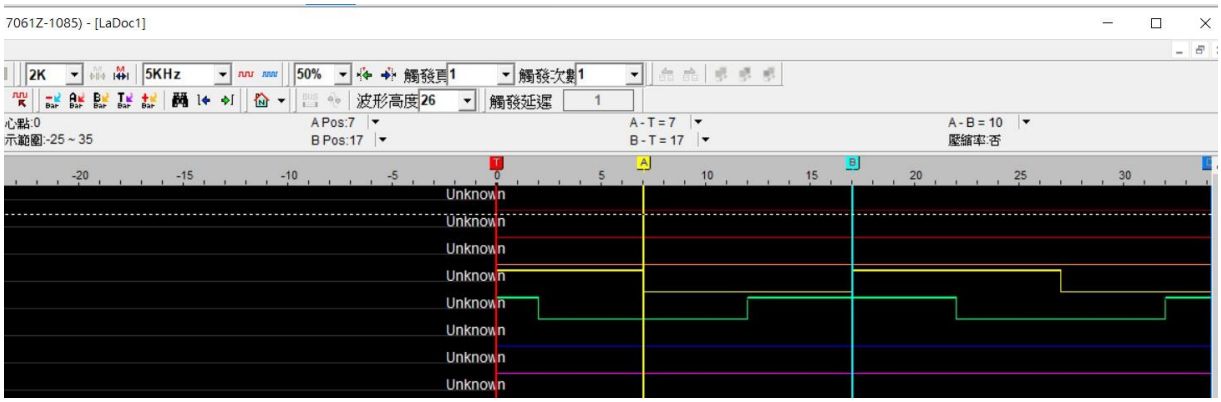
```
10  // DigitalIn toggle(D4);
11  static DigitalOut out(D4);
12  static DigitalOut out2(D3);
```

3. BSP_AUDIO_IN_HalfTransfer_CallBack 和 BSP_AUDIO_IN_TransferComplete_CallBack 這兩個函式表示讀取一半的數據及全部的數據。為了使整個 program 可以持續循環進行，因此將後段把 audio data 存入 PCM_buffer 中的 code 註解，並在前面加上 toggle，使得方便在邏輯分析儀上觀察進行的進度，因此在 half callback 時設 定 toggle out 腳位 complete callback 時 toggle out2 腳位 。

```
79  void BSP_AUDIO_IN_HalfTransfer_CallBack(uint32_t Instance) {
80      half_transfer_events++;
81
82      out = !out;
83      // out2 = 0;
84      if (half_transfer_events < SKIP_FIRST_EVENTS) return;
85
86      uint32_t buffer_size = PCM_BUFFER_LEN / 2; /* Half Transfer */
87      uint32_t nb_samples = buffer_size / sizeof(int16_t); /* Bytes to Length */
88
89      // if ((TARGET_AUDIO_BUFFER_IX + nb_samples) > TARGET_AUDIO_BUFFER_NB_SAMPLES) {
90      //     return;
91      // }
92
93      // /* Copy first half of PCM_Buffer from Microphones onto Fill_Buffer */
94      // memcpy(((uint8_t*)TARGET_AUDIO_BUFFER) + (TARGET_AUDIO_BUFFER_IX * 2), PCM_Buffer, buffer_size);
95      // TARGET_AUDIO_BUFFER_IX += nb_samples;
96
97      // if (TARGET_AUDIO_BUFFER_IX >= TARGET_AUDIO_BUFFER_NB_SAMPLES) {
98      //     ev_queue.call(&target_audio_buffer_full);
99      //     return;
100     // }
101 }
102
```

```c
void BSP_AUDIO_IN_TransferComplete_CallBack(uint32_t Instance) {
    transfer_complete_events++;

    // out = 2;
    out2 = !out2;

    if (transfer_complete_events < SKIP_FIRST_EVENTS) return;

    uint32_t buffer_size = PCM_BUFFER_LEN / 2; /* Half Transfer */
    uint32_t nb_samples = buffer_size / sizeof(int16_t); /* Bytes to Length */

    // if ((TARGET_AUDIO_BUFFER_IX + nb_samples) > TARGET_AUDIO_BUFFER_NB_SAMPLES)
    //     return;
    // }

    // /* Copy second half of PCM_Buffer from Microphones onto Fill_Buffer */
    // memcpy(((uint8_t*)TARGET_AUDIO_BUFFER) + (TARGET_AUDIO_BUFFER_IX * 2),
    //     ((uint8_t*)PCM_Buffer) + (nb_samples * 2), buffer_size);
    // TARGET_AUDIO_BUFFER_IX += nb_samples;

    // if (TARGET_AUDIO_BUFFER_IX >= TARGET_AUDIO_BUFFER_NB_SAMPLES) {
    //     ev_queue.call(&target_audio_buffer_full);
    //     return;
    // }
}
```

4. 連接邏輯分析儀觀察，兩條訊號線輸出的波型相位相差 90 度，符合實驗要求，在半滿及全滿時進行切換。




```c
#define AUDIO_VOLUME_VALUE              32
#define AUDIO_SAMPLING_FREQUENCY        16000
#define AUDIO_DFSDM_DMAx_MIC1_IRQHandler    DMA1_Channel4_IRQHandler
```

可以觀察到，原先設定的 sampling freq 為 16000Hz，在 timer 為 80MHz 的情況下，sampling rate 為 80MHz/16000 = 5000Hz。而在視波器的 A-B=10，在 5kHz 的取樣下表示 A~B 點的間隔秒數為 2ms