Report
107061112 王昊文

I. Design a program that can play tone music using either PWM timers of the NUC140.
   1. (15%) When program starts and no key pressed:
      (1) Blink and change the color of RGB LED indefinitely.
      (2) Display a message of “Prepare to play music” on the LCD.
      (3) Display “0” on 7-segment LED.
   2. (40%) Press “1” of the keypad
      (1) Playthe NTHU University anthem.
      (2) Display a message of “Now playing the NTHU University anthem” as well as your student no. and name on the LCD.
      (3) Display “1” on 7-segment LED.
   3. (40%) When press “2” of the keypad
      (1) Play a single-tone sine wave sound with frequency of your choice (e.g., 1 kHz) using PWM. You can use DDS concept to synthesize the sine wave.
      (2) Display a message of “Now playing my choice of single tone” as well as your student no. and name on the LCD.
      (3) Display “2” on 7-segment LED.
   4. (5%) If any other key is pressed, or if no key pressed and the music is done, your program restarts (goto 1).

Design Concept:

I will divide the whole system into three states; the reset state, number == 1 and number == 2. I will use the return value of ScanKey() as our state number. Note that when ScanKey() == 0 indicates that no key pressed, but since we are required to play a whole song, this will keep us inside a state even though currently no key pressed. Using the ScanKey() == 0 condition can let us return to the reset state while the music or the single tone sine wave finished playing. Using this advantage, if we scan while playing music, we can easily satisfy what is required in question number 4.

The method to scan while playing music, is simply using a for loop and not use the internal system delay instruction, as follows:

```
for (i = 0; i < 50 && ScanKey(); i++);
```

Although the drawback is that we are not able to calculate the actual delay time, but we can sort of neglect the drawback since it is not that important.

In order to answer (1), we simply output any arbitrary CMR to PWM_Out 1, 2, 3 (the RGB ports), we can generate a color indefinitely, and if we generate different random numbers as the time goes on, it will look like the LED is changing color indefinitely.

And for question (2), we are required to output different frequencies, by changing the parameter “frequency” for CNR, simply construct an array, which will

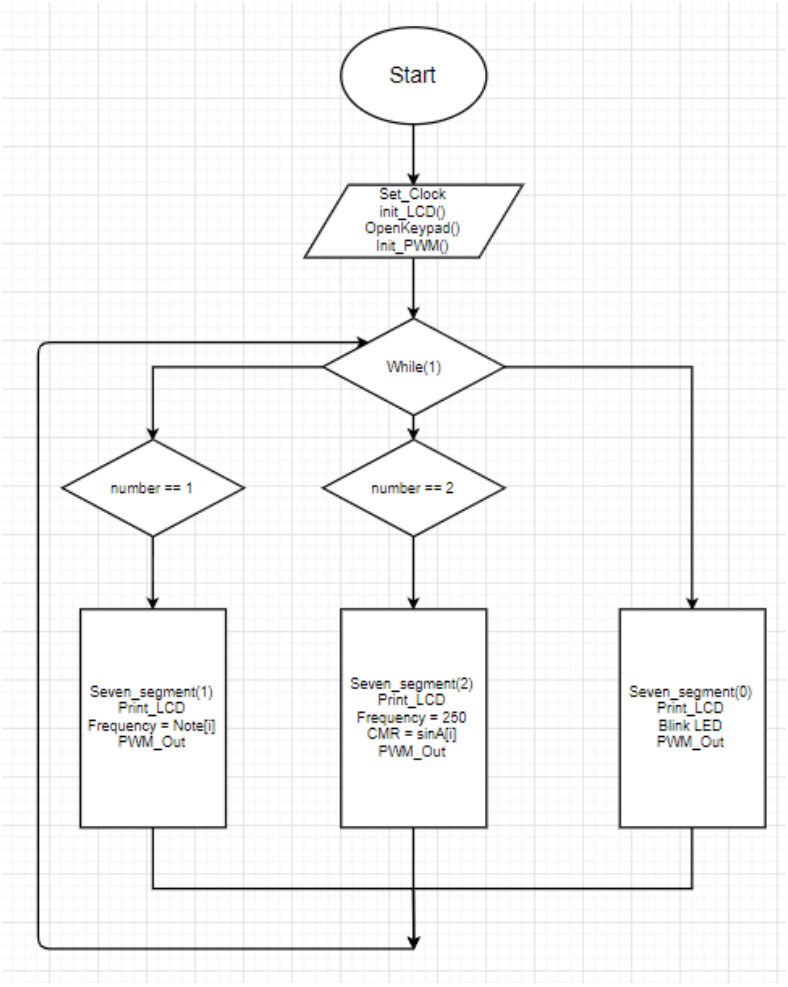
store the notes. While playing each note, use the delay method as I mentioned above, then we are able to jump out of the state if any other key pressed.

For question (2), the problem becomes more complicated. We must use the concept of comparing CMR and CNR. Whenever CNR is larger than CMR, the PWMout will output 1, and on the other hand, output 0. Using this concept, we can generate a sound similar to sine wave. I simply construct a table, which are the amplitude of sine wave (the value are constructed by Excel). My maximum amplitude is 100, since the reset value of CNR is 99. So if I let CNR = 99, then by the formula of CNR

CNR = Clock / Frequency / (PreScaler + 1) / ClockDivider – 1;
99 = 12000000 / Frequency / 120 / 4 – 1;
Frequency = 250(Hz).

So the frequency of my sine wave is 250Hz. The delay time between each point will affect the frequency a little bit, but since we want the sine wave to be smooth and continuous, so the delay time shouldn't be long. Hence we can the neglect the affect with the delay time.
This concept is similar to ADC, since the board only reads digital signal, we cut the sine wave into pieces and approximate it as a continuous sine wave.

Flow Chart:



Code:

```
//
// Smpl_PWM_Music : use PWM to generate musical notes
//
// Output : PWM0 (GPA12)

#include <stdio.h>
#include <math.h>

#include "NUC1xx.h"
#include "GPIO.h"
#include "SYS.h"
#include "PWM.h"
#include "LCD.h"
#include "Note_Freq.h"
#include "Seven_Segment.h"
#include "Scankey.h"

int32_t main (void)
{
    int delay_cnt;              // for delay count
    int i;
        int j;
    uint32_t Clock;
    uint32_t Frequency;
    uint8_t  PreScaler;
    uint8_t  ClockDivider;
    uint8_t  DutyCycle;
    uint16_t number;            // for scankey
    uint16_t CNR, CMR;

    uint16_t NTHU_DELAY[77] = {
        3, 1, 4, 2, 1, 1, 4,
        4, 1, 1, 2, 2, 1, 1, 2, 2,
        5, 1, 2, 2, 2, 1, 1, 2,
        2, 3, 1, 2, 4, 1, 1, 2,
        4, 2, 2, 3, 1, 4,
        4, 2, 2, 3, 1, 4,
        6, 2, 2, 1, 1, 4,
        4, 2, 2, 3, 1, 4,
        6, 2, 6, 2,
        4, 2, 2, 2, 2, 4,
        6, 2, 6, 2,
        4, 2, 2, 2, 1, 1, 4
    };
    uint16_t NTHU_ANTH[77] = {
        D5u, G5, A5u, C6, D6u, C6, A5u,
        G5, A5u, G5, D5u, C5, D5u, F5, A5u, 0,
        C6, D6u, A5u, G5, F5, G5, F5, D5u,
```

```
        F5, A5u, G5u, A5u, C6, D6, C6, A5u,
        D6u, C6, D6u, A5u, C6, A5u,
        C6, A5u, G5, F5, G5, A5u,
        D5u, G5, F5, G5, F5, D5u,
        C6, A5u, G5, F5, G5, D5u,
        D6u, 0, C6, 0,
        A5u, C6, A5u, F5, G5, A5u,
        D6u, 0, C7, 0,
        A5u, C6, A5u, F5, G5, F5, D5u
    };
    uint16_t sineA[100] ={
        53, 56, 59, 62, 65,
        68, 71, 74, 77, 79,
        82, 84, 86, 89, 90,
        92, 94, 95, 96, 98,
        98, 99, 100, 100, 100,
        100, 100, 99, 98, 98,
        96, 95, 94, 92, 90,
        89, 86, 84, 82, 79,
        77, 74, 71, 68, 65,
        62, 59, 56, 53, 50,
        47, 44, 41, 38, 35,
        32, 29, 26, 23, 21,
        18, 16, 14, 12, 10,
        8, 6, 5, 4, 2,
        2, 1, 0, 0, 0,
        0, 0, 1, 2, 2,
        4, 5, 6, 8, 10,
        12, 14, 16, 18, 21,
        23, 26, 29, 32, 35,
        38, 41, 44, 47, 50
    };
    //Enable 12Mhz and set HCLK->12Mhz

    UNLOCKREG();
    SYSCLK->PWRCON.XTL12M_EN = 1;
    SYSCLK->CLKSEL0.HCLK_S = 0;
    LOCKREG();

    init_LCD();
    clear_LCD();
    OpenKeyPad();

// PWM_No = PWM channel number
// PWM_CLKSRC_SEL   = 0: 12M, 1:32K, 2:HCLK, 3:22M
// PWM_PreScaler    : PWM clock is divided by (PreScaler + 1)
// PWM_ClockDivider = 0: 1/2, 1: 1/4, 2: 1/8, 3: 1/16, 4: 1
    init_PWM(0, 0, 119, 4);
```

```
    init_PWM(1, 0, 119, 4);
    init_PWM(2, 0, 119, 4);
    init_PWM(3, 0, 119, 4); // initialize PWM0(GPA12) to output 1MHz sq
uare wave
    Clock = 12000000;
    PreScaler = 119;
    ClockDivider = 1;
    DutyCycle = 50;
    Number = ScanKey()
    while(1)
    {
        number = ScanKey();
        if (number == 1) {
            clear_LCD();
            DutyCycle = 50;
            PWM_Stop(0);
            PWM_Stop(1);
            PWM_Stop(2);
            show_seven_segment(0, number);          // show ssd
            print_Line(0, "Now playing the");
            print_Line(1, "NTHU University");
            print_Line(2, "anthem");
            print_Line(3, "107061112 Howard");
            for (i = 0; i < 77 && ScanKey() == 0; i++) {
                Frequency = NTHU_ANTH[i];
                //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_C
lockDivider / (PWM_CNR + 1)
                CNR = Clock / Frequency / (PreScaler + 1) / ClockDivide
r - 1;
                // Duty Cycle = (CMR0+1) / (CNR0+1)
                CMR = (CNR + 1) * DutyCycle /100  - 1;

                PWM_Out(3, CNR, CMR);
                if (Frequency == 0)
                    PWM_Stop(3);

                for (delay_cnt = 250 * NTHU_DELAY[i]; (delay_cnt >= 0)
&& ScanKey() == 0; delay_cnt--) {    // for delaying
                    DrvSYS_Delay(1);
                }
            }
        }
        else if (number == 2) {
            clear_LCD();
            PWM_Stop(0);
            PWM_Stop(1);
            PWM_Stop(2);
            show_seven_segment(0, number);          // show ssd
```

```
            print_Line(0, "Now playing");
            print_Line(1, "My choice of");
            print_Line(2, "the single tone");
            print_Line(3, "107061112 Howard");
            Frequency = 250;
            CNR = Clock / Frequency / (PreScaler + 1) / ClockDivider -
 1;
            for (j = 0; j < 15 && ScanKey() == 0; j++) {
                for (i = 0; i < 100 && ScanKey() == 0; i++) {
                    PWM_Out(3, CNR, sineA[i]);
                    for (delay_cnt = 20; (delay_cnt >= 0) && ScanKey()
== 0; delay_cnt--) { // for delaying
                        DrvSYS_Delay(1);
                    }
                }
            }
        }
        else {
            clear_LCD();
            close_seven_segment();
            show_seven_segment(0, 0);           // show ssd
            print_Line(0, "Prepare to play");
            print_Line(1, "music");
            PWM_Stop(3);
            for (DutyCycle = 100; DutyCycle > 60; DutyCycle--) {
                //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_C
lockDivider / (PWM_CNR + 1)
                CNR = Clock / Frequency / (PreScaler + 1) / ClockDivide
r - 1;
                // Duty Cycle = (CMR0+1) / (CNR0+1)
                CMR = (CNR + 1) * DutyCycle /100  - 1;
                PWM_Out(0, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 30)/100  - 1;
                PWM_Out(1, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 60)/100  - 1;
                PWM_Out(2, CNR, CMR);
                for (delay_cnt = 40; (delay_cnt >= 0) && ScanKey() == 0
; delay_cnt--) { // for delaying
                    DrvSYS_Delay(1);
                }
            }
            for (DutyCycle = 100; DutyCycle > 60; DutyCycle--) {
                //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_C
lockDivider / (PWM_CNR + 1)
                CNR = Clock / Frequency / (PreScaler + 1) / ClockDivide
r - 1;
            // Duty Cycle = (CMR0+1) / (CNR0+1)
            CMR = (CNR + 1) * DutyCycle /100  - 1;
```

```
                PWM_Out(2, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 30)/100  - 1;
                PWM_Out(1, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 60)/100  - 1;
                PWM_Out(0, CNR, CMR);
                for (delay_cnt = 40; (delay_cnt >= 0) && ScanKey() == 0
; delay_cnt--) { // for delaying
                    DrvSYS_Delay(1);
                }
            }
        }
    }
}
```

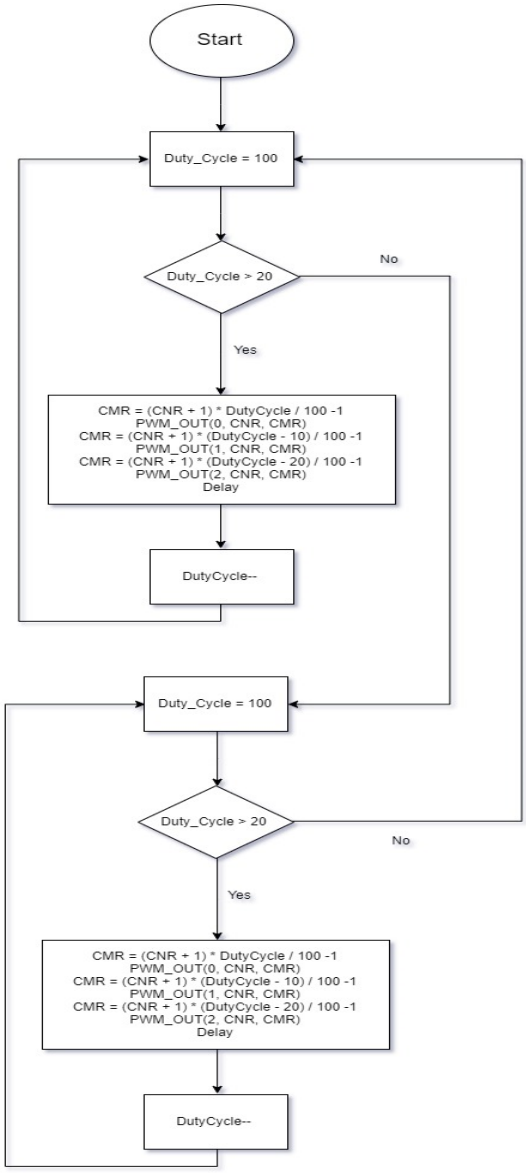
Question:

I. On the NUC140 experiment board, we have a RGB LED installed and driven with GPA12, 13, 14 which happen to be PWM0, 1, 2. Write a program that can change the RGB LED color **gradually** and **repeatedly** using PWM method. Explain your method, code, and result in report and **demo your result to TA**.

Design Concept:

We can change the “brightness” of R, G, B, by adjusting the duty cycle of the PWM signal. So if we adjust the duty cycle using for loop, we can get a result of the color is “gradually” changing its brightness (we call it breathing light). Moreover, if R, G, B, is changing the brightness at the same time, We can get a result which looks the color is changing gradually all the time.

Flow Chart:




```
//
// Smpl_PWM_Music : use PWM to generate musical notes
//
// Output : PWM0 (GPA12)

#include <stdio.h>
#include <math.h>

#include "NUC1xx.h"
#include "GPIO.h"
#include "SYS.h"
#include "PWM.h"
#include "LCD.h"
#include "Note_Freq.h"
#include "Seven_Segment.h"
#include "Scankey.h"
```

```c
int32_t main (void)
{
    int delay_cnt;              // for delay count
    int i;
        int j;
    uint32_t Clock;
    uint32_t Frequency;
    uint8_t  PreScaler;
    uint8_t  ClockDivider;
    uint8_t  DutyCycle;
    uint16_t number;            // for scankey
    uint16_t CNR, CMR;


    //Enable 12Mhz and set HCLK->12Mhz

    UNLOCKREG();
    SYSCLK->PWRCON.XTL12M_EN = 1;
    SYSCLK->CLKSEL0.HCLK_S = 0;
    LOCKREG();

    init_LCD();
    clear_LCD();
    OpenKeyPad();

// PWM_No = PWM channel number
// PWM_CLKSRC_SEL   = 0: 12M, 1:32K, 2:HCLK, 3:22M
// PWM_PreScaler    : PWM clock is divided by (PreScaler + 1)
// PWM_ClockDivider = 0: 1/2, 1: 1/4, 2: 1/8, 3: 1/16, 4: 1
    init_PWM(0, 0, 119, 4);
    init_PWM(1, 0, 119, 4);
    init_PWM(2, 0, 119, 4);
    init_PWM(3, 0, 119, 4); // initialize PWM0(GPA12) to output 1MHz sq
uare wave
    Clock = 12000000;
    PreScaler = 119;
    ClockDivider = 1;
    DutyCycle = 50;
    while(1) {
      clear_LCD();
      close_seven_segment();
      PWM_Stop(3);
      for (DutyCycle = 100; DutyCycle > 20; DutyCycle--) {
      //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_ClockDivide
r / (PWM_CNR + 1)
          CNR = Clock / Frequency / (PreScaler + 1) / ClockDivider - 1;
          // Duty Cycle = (CMR0+1) / (CNR0+1)
          CMR = (CNR + 1) * DutyCycle /100  - 1;
```

```
            PWM_Out(0, CNR, CMR);
            CMR = (CNR + 1) * (DutyCycle - 10)/100  - 1;
            PWM_Out(1, CNR, CMR);
            CMR = (CNR + 1) * (DutyCycle - 20)/100  - 1;
            PWM_Out(2, CNR, CMR);
            for (delay_cnt = 40; (delay_cnt >= 0) && ScanKey() == 0; d
elay_cnt--) { // for delaying
                  DrvSYS_Delay(1);
                }
            }
            for (DutyCycle = 100; DutyCycle > 20; DutyCycle--) {
                //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_C
lockDivider / (PWM_CNR + 1)
                CNR = Clock / Frequency / (PreScaler + 1) / ClockDivide
r - 1;
            // Duty Cycle = (CMR0+1) / (CNR0+1)
            CMR = (CNR + 1) * DutyCycle /100  - 1;
                PWM_Out(2, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 20)/100  - 1;
                PWM_Out(1, CNR, CMR);
                CMR = (CNR + 1) * (DutyCycle - 10)/100  - 1;
                PWM_Out(0, CNR, CMR);
                for (delay_cnt = 40; (delay_cnt >= 0) && ScanKey() == 0
; delay_cnt--) { // for delaying
                    DrvSYS_Delay(1);
                }
            }
      }
}
```