HW12 report

107061112 王昊文

1. Write an ARM Cortex-M0 NUC140 program that can read and display internal temperature sensor output by using ADC7 continuously.

    1. Read NUC140 technical reference manual about internal temperature sensor
    2. Set up ADC7 to convert temperature sensor output continuously
    3. Display the temperature result

Design Concept:

The most important part for this homework is that we will have to choose the correct input channel for the ADC input. So inside the InitADC() function, we first choose the clock source and the mode for ADC, by default, we will choose the single mode and the 12MHz clock. Then, since in this problem we are required to read the temperature from the internal temperature sensor, and the register storing the number of the temperature is in channel 7. There are four analog input for channel 7 storing inside the PRESEL register, as shown below

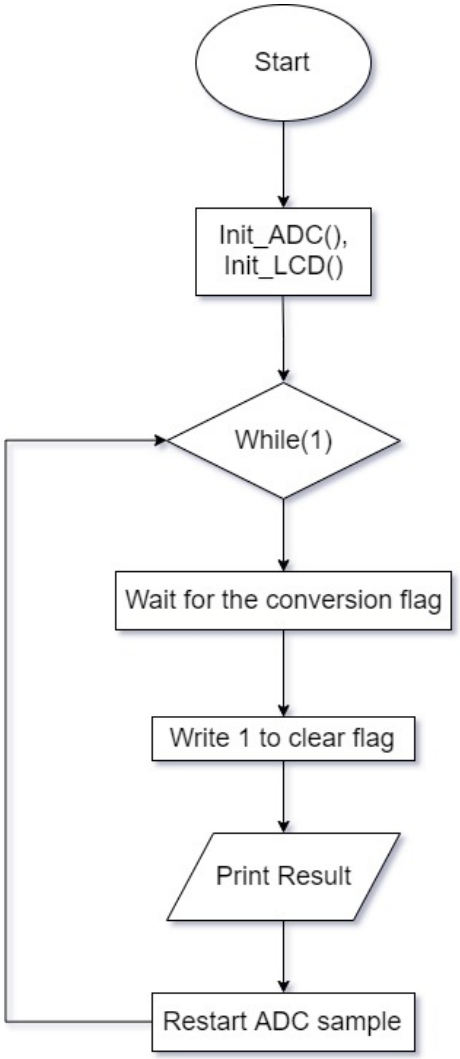
[9:8]	**PRESEL**	**Analog Input Channel 7 Selection** 00 = External analog input 01 = Internal band-gap voltage 10 = Internal temperature sensor 11 = Reserved **Note:** When software select the band-gap voltage as the analog input source of ADC channel 7, ADC clock rate needs to be limited to lower than 300 kHz.

I realize that inside the library ADC.c, the function DrvADC_ConfigADCChannel() can let us select the PRESEL register. So DrvADC_ConfigADCChannel(2) will let us select the internal temperature sensor.

The main program is pretty simple. If we want to read the value inside the register, look the flag ADF. If the analog signal is converted to digital signal successfully, we then read out the value from the ADDR register.

It will print out a value among 845~848.

Design Flow:



My Code:

```
//
// Smpl_ADC_VR1 : use ADC7 to read Variable Resistor (on-board)
//
#include <stdio.h>

#include "NUC1xx.h"
#include "SYS.h"
#include "LCD.h"
#include "ADC.h"
void InitADC(void)
{
    /* Step 1. GPIO initial */
    GPIOA->OFFD|=0x00800000;    //Disable digital input path
    SYS->GPAMFP.ADC7_SS21_AD6=1;        //Set ADC function

    /* Step 2. Enable and Select ADC clock source, and then enable ADC
module */
    SYSCLK->CLKSEL1.ADC_S = 2;  //Select 22Mhz for ADC
    SYSCLK->CLKDIV.ADC_N = 1;   //ADC clock source = 22Mhz/2 =11Mhz;
    SYSCLK->APBCLK.ADC_EN = 1;  //Enable clock source
```

```
    ADC->ADCR.ADEN = 1;         //Enable ADC module

    /* Step 3. Select Operation mode */
    ADC->ADCR.DIFFEN = 0;       //single end input
    ADC->ADCR.ADMD   = 0;       //single mode

    /* Step 4. Select ADC channel */
    ADC->ADCHER.CHEN = 0x80;
    DrvADC_ConfigADCChannel7(2);
    /* Step 5. Enable ADC interrupt */
    ADC-
>ADSR.ADF =1;           //clear the A/D interrupt flags for safe
    ADC->ADCR.ADIE = 1;
//  NVIC_EnableIRQ(ADC_IRQn);

    /* Step 6. Enable WDT module */
    ADC->ADCR.ADST=1;
}

int32_t main (void)
{
    char TEXT1[16]="ADC Value:      ";
    UNLOCKREG();
    SYSCLK->PWRCON.XTL12M_EN = 1; // enable external clock (12MHz)
    SYSCLK->CLKSEL0.HCLK_S = 0;   // select external clock (12MHz)
    LOCKREG();

    InitADC();          // initialize ADC

    init_LCD();  // initialize LCD pannel
    clear_LCD();  // clear LCD panel
    print_Line(0, "Smpl_ADC_VR1");

    while(1)
    {
        while(ADC-
>ADSR.ADF==0); // wait till conversion flag = 1, conversion is done
        ADC->ADSR.ADF=1;             // write 1 to clear the flag
        sprintf(TEXT1+10,"%4d",ADC-
>ADDR[7].RSLT); // convert ADC7 value into text
        print_Line(1, TEXT1);      // output TEXT to LCD
        DrvSYS_Delay(20000);       // delay
        ADC->ADCR.ADST=1;            // restart ADC sample
    }
}
```

Question:

I. On the NUC140 experiment board, we have a VR connected to ADC7 and 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 using PWM method according to the VR voltage inputted by ADC7. Explain your method, code, result in report and demo your result to TA.

Design Concept:

Since we are required to change the colors gradually, we are required to use PWM. We can change the brightness of RGB by modifying the duty cycle of the duty cycle. The maximum value of the variable resistor is 4095, so if we modify the voltage value by 100 and divide by 4095, we can change the duty cycle while changing the voltage inputted by ADC7.

So if continue to use the initial function by the previous problem, and we only change the PRESEL to "00" (external analog input), then make the CMR, CNR related to the value read, then we can change the color gradually.

I let the digital value read have the following relation with duty cycle

```
DutyCycle = ADC->ADDR[7].RSLT * 100 / 4095;
```

And let R's CMR have the following relationship with respect to duty cycle

```
CMR = (CNR + 1) * (10*sqrt((100 - DutyCycle)))/ 100  - 1;
```

G's CMR:

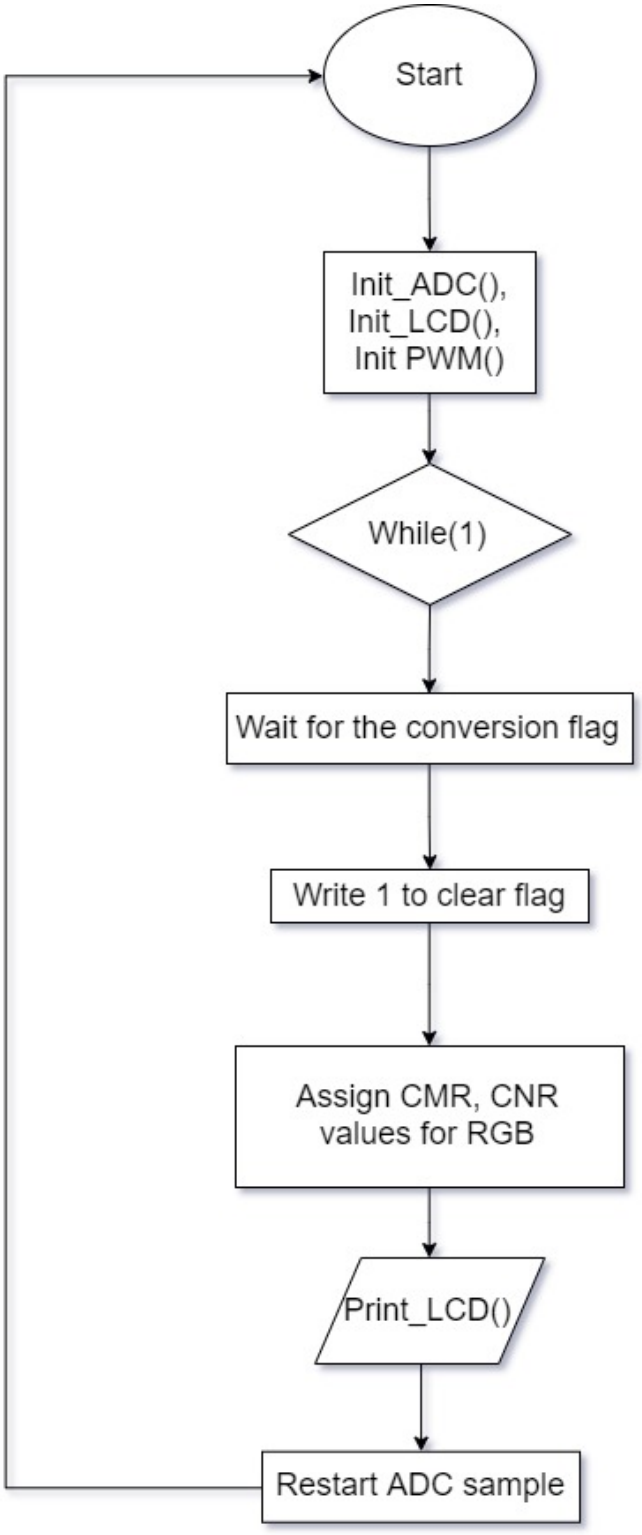
```
CMR = (CNR + 1) * (10*sqrt((DutyCycle / 2 + 50)))/ 100  - 1;
```

B's CMR:

```
CMR = (CNR + 1) * (10*sqrt(((100 - DutyCycle) / 2))) / 100  - 1;
```

Using these function, we can let the colors change gradually.

Design Flow:



My Code:

```
//
// Smpl_ADC_VR1 : use ADC7 to read Variable Resistor (on-board)
//
#include <stdio.h>

#include "NUC1xx.h"
#include "SYS.h"
```

```c
#include "LCD.h"
#include "PWM.h"
#include "GPIO.h"
#include "ADC.h"
#include "math.h"

void InitADC(void)
{
    /* Step 1. GPIO initial */
    GPIOA->OFFD|=0x00800000;    //Disable digital input path
    SYS->GPAMFP.ADC7_SS21_AD6=1;        //Set ADC function

    /* Step 2. Enable and Select ADC clock source, and then enable ADC
module */
    SYSCLK->CLKSEL1.ADC_S = 2;  //Select 22Mhz for ADC
    SYSCLK->CLKDIV.ADC_N = 1;   //ADC clock source = 22Mhz/2 =11Mhz;
    SYSCLK->APBCLK.ADC_EN = 1;  //Enable clock source
    ADC->ADCR.ADEN = 1;         //Enable ADC module

    /* Step 3. Select Operation mode */
    ADC->ADCR.DIFFEN = 0;       //single end input
    ADC->ADCR.ADMD   = 0;       //single mode

    /* Step 4. Select ADC channel */
    ADC->ADCHER.CHEN = 0x80;
    DrvADC_ConfigADCChannel7(0);
    /* Step 5. Enable ADC interrupt */
    ADC-
>ADSR.ADF =1;           //clear the A/D interrupt flags for safe
    ADC->ADCR.ADIE = 1;
//  NVIC_EnableIRQ(ADC_IRQn);

    /* Step 6. Enable WDT module */
    ADC->ADCR.ADST=1;
}

int32_t main (void)
{
    uint32_t Clock;
    uint32_t Frequency;
    uint8_t  PreScaler;
    uint8_t  ClockDivider;
    uint8_t  DutyCycle;
    uint16_t CNR, CMR;
    char TEXT1[16]="ADC Value:      ";
    char TEXT2[15] = "DC Value:      ";

    Clock = 12000000;
```

```
    PreScaler = 119;
    ClockDivider = 1;
    DutyCycle = 0;
    Frequency = 500;
    UNLOCKREG();
    SYSCLK->PWRCON.XTL12M_EN = 1; // enable external clock (12MHz)
    SYSCLK->CLKSEL0.HCLK_S = 0;   // select external clock (12MHz)
    LOCKREG();

    InitADC();          // initialize ADC
    init_PWM(0, 0, 119, 4);
    init_PWM(1, 0, 119, 4);
    init_PWM(2, 0, 119, 4);
    init_LCD();  // initialize LCD pannel
    clear_LCD();  // clear LCD panel
    print_Line(0, "Smpl_ADC_VR1");

    while(1)
    {
        while(ADC-
>ADSR.ADF==0); // wait till conversion flag = 1, conversion is done
        ADC->ADSR.ADF=1;             // write 1 to clear the flag
        //PWM_FreqOut = PWM_Clock / (PWM_PreScaler + 1) / PWM_ClockDivi
der / (PWM_CNR + 1)
        DutyCycle = ADC->ADDR[7].RSLT * 100 / 4095;
        sprintf(TEXT2+9, "%4d", DutyCycle);
        print_Line(2, TEXT2);
        CNR = Clock / Frequency / (PreScaler + 1) / ClockDivider - 1;
    // Duty Cycle = (CMR0+1) / (CNR0+1)
    CMR = (CNR + 1) * (10*sqrt((100 - DutyCycle)))/ 100  - 1;
        PWM_Out(0, CNR, CMR);
        CMR = (CNR + 1) * (10*sqrt((DutyCycle / 2 + 50)))/ 100  -
 1;
        PWM_Out(1, CNR, CMR);
        CMR = (CNR + 1) * (10*sqrt(((100 - DutyCycle) / 2))) / 100  -
 1;
        PWM_Out(2, CNR, CMR);
        sprintf(TEXT1+10,"%4d",ADC-
>ADDR[7].RSLT); // convert ADC7 value into text
        print_Line(1, TEXT1);      // output TEXT to LCD
        DrvSYS_Delay(20000);       // delay
        ADC->ADCR.ADST=1;            // restart ADC sample
    }
}
```

Discussion:

這次的 lab 一樣難度不高，基本上也是更改範例 code 即可。這次作業遇到的大問題主要是在選擇輸出正確的 register，只需詳細閱讀指南就可以輕鬆完成。我覺得美中不足的地方是這次 lab 並不需要瞭解到 ADC 讀取的方式，其實對於那些 scan mode，continuous mode 其實還是抱持著很大的疑問，或是有更多實際的例子可以讓我們參考。第二題的部分我花了很多時間讓我顏色的變化變的更多元更連續的感覺，助教看到的那些 square root function 只是為了不要讓顏色變化太線性看起來太單調而已。

總算是完成這學期所有作業啦～