HW11 Report
107061112 王昊文

I. Write an ARM Cortex-M0 NUC140 program for measuring and display on LCD the distance obtained by SR04:
   1. The displayed distance number should be an average of consecutive 5 SR04 echo measurements (每量得 5 次數據後算出平均)
   2. Try 2 distances: 10 cm, 20 cm

Design Concept:

By using the capture function of the timer, and the trigger of the ultrasonic, we can calculate the distance between the ultrasonic generator and the object. In this lab, I am going to use the one shot mode to accomplish the task. During the one shot mode, once the counter counts to the compare value, the counter will reset. Hence we must enable the counter every time the timer interrupts.

So, the design flow for the lab is pretty simple. We first initialize our counter, where we select the counting frequency we desire. I select the 12Mhz clock with a prescale value of 11, which gives a frequency of 1Mhz. We then set the counter to one-shot mode, and set up all the capture flags and enable the counter.

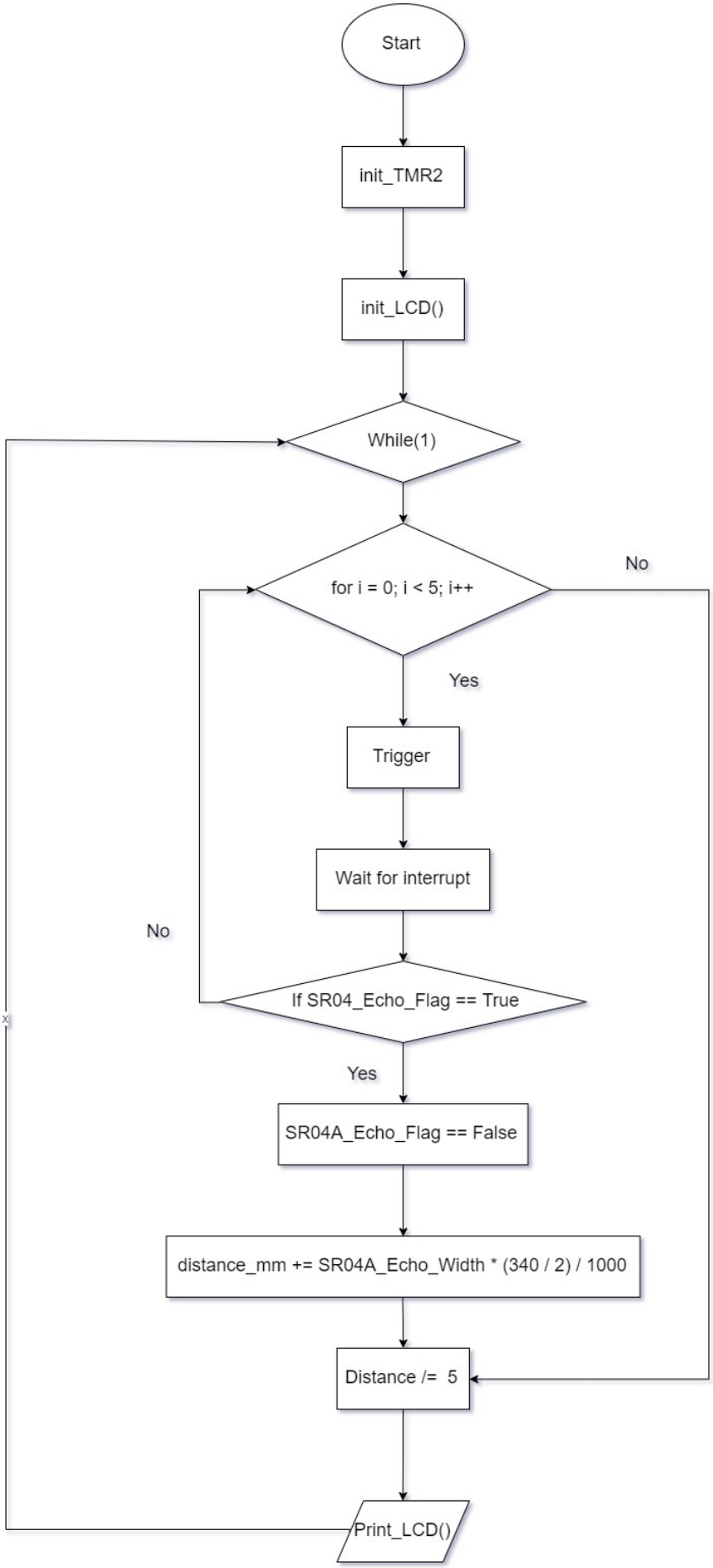
The most important part of the program is the IRQ handler function. The IRQ function will be triggered in two conditions. First is once the counter counts to the compare value. The second case is when the TEXIF is set to one, in other words, when the echo arrives the ultrasonic generator, it will interrupt. Inside the interrupt, we first capture the value of the counter, then we make the counter to continue. Then the most important part is to check if TEXIF is 1, we want to make sure that the ultrasonic is echoed back to the user circuit. If the system is interrupted because the counter counts to the compare value, we will ignore the captured value.

For the main function, we simply record five time the captured value and divide by five to get the average value.

Timer parameters:

Clock: 12MHz
Prescale: 11
TCMPR: 0xffffff
Mode: One-Shot
TEXEN: Enabled
TEXIEN: Enabled

Program Flow:


Start
init_TMR2
init_LCD()
While(1)
for i = 0; i < 5; i++
No
Yes
Trigger
Wait for interrupt
No
If SR04_Echo_Flag == True
Yes
SR04A_Echo_Flag == False
distance_mm += SR04A_Echo_Width * (340 / 2) / 1000
Distance /= 5
Print_LCD()

My Code:

```
//
// Smpl_Timer_SR04 : using one SR04 Ultrasound Sensors
//
// Timer Capture :
// GPB2 / RTS0 / T2EX (NUC140 pin34)
// GPB4 / RX1         (NUC140 pin19)

// SR04 Ultrasound Sensor
// pin1 Vcc : to Vcc
// pin2 Trig: to GPB4      (NUC140VE3xN pin19)
// pin3 ECHO: to GPB2/T2EX (NUC140VE3xN pin34)
// pin4 Gnd : to GND

#include <stdio.h>
#include "NUC1xx.h"
#include "GPIO.h"
#include "SYS.h"
#include "LCD.h"

// Global definition
#define _SR04A_ECHO        (GPB_2)          //NUC140VE3xN, Pin19
#define _SR04A_TRIG        (GPB_4)          //NUC140VE3xN, Pin34
#define _SR04A_TRIG_Low  (GPB_4=0)
#define _SR04A_TRIG_High (GPB_4=1)

// Global variables
volatile uint32_t SR04A_Echo_Width = 0;
volatile uint32_t SR04A_Echo_Flag  = FALSE;
//
// Initial TMR2
//
// Timer Clock: 12 MHz
// Prescale:        11
// Compare:     0xffffff
// Mode:            One-Shot mode
// Capture:     Enable, Capture with Falling Edge
void init_TMR2(void)
{
    //Step 1: T2EX pin Enable (PB.2, Pin34)
    SYS->GPBMFP.UART0_nRTS_nWRL = 1;
    SYS->ALTMFP.PB2_T2EX = 1;

  //Step 2: Timer Controller Reset and Setting Clock Source
    SYS->IPRSTC2.TMR2_RST = 1;          //Timer Controller: Reset
    SYS->IPRSTC2.TMR2_RST = 0;          //Timer Controller: Normal
    SYSCLK->CLKSEL1.TMR2_S = 0;         //Timer Clock = 12 MHz
    SYSCLK->APBCLK.TMR2_EN = 1;         //Timer C lock Enable
```

```
    //Step 3: Timer Controller Setting
    //  TMR0_T = (12 MHz / (11+1) / 1000000)^-1 = 1.000 Second
    TIMER2->TCMPR = 0xffffff;      //Timer Compare Value:  [0~16777215]
    TIMER2->TCSR.PRESCALE = 11;         //Timer Prescale:       [0~255]
    TIMER2->TCSR.MODE = 0;              //Timer Operation Mode: One-
Shot

    //Step 4: External Capture Mode Setting
    TIMER2->TEXCON.TEXEN = 1;        //External Capture Function Enable
    TIMER2->TEXCON.RSTCAPSEL = 0;  //Capture Mode Select: Capture Mode
    TIMER2->TEXCON.TEX_EDGE = 2;      //Capture Edge: Rising & Falling

    //Step 5: Timer Interrupt Setting
//  TIMER2->TCSR.IE = 1;                    //Timeout Interrupt Enable
//  TIMER2->u32TISR |= 0x01;               //Clear Timeout Flag (TIF)
    TIMER2->TEXCON.TEXIEN = 1;             //Capture Interrupt Enable
    TIMER2->u32TEXISR |= 0x01;             //Clear Capture Flag (TEXIF)
    NVIC_EnableIRQ(TMR2_IRQn);             //Timer NVIC IRQ Enable

    //Step 6: Start Timer Capture (Set by Ultrasonic_Trigger() Function)
//  TIMER2->TCSR.CRST = 1;                 //Timer Counter Reset
//  TIMER2->TCSR.CEN = 1;                      //Timer Start
}

// TMR2 Interrupt Handler
void TMR2_IRQHandler(void)
{
    TIMER2->TEXCON.RSTCAPSEL = 0;
// set back for falling edge to capture
    TIMER2->TCSR.CEN = 1;                    //Timer Start
    if(TIMER2->TEXISR.TEXIF == 1)          //Capture Flag (TEXIF)
    {
        TIMER2->u32TEXISR |= 0x01;         //Clear Capture Flag (TEXIF)
        SR04A_Echo_Width = TIMER2->TCAP; //Load Capture Value (Unit: us)
        SR04A_Echo_Flag  = TRUE;
    }
}

// Ultrasonic Trigger
void SR04_Trigger(void)
{
    //Trigger of Ultrasonic Sensor
    _SR04A_TRIG_High;
    DrvSYS_Delay(10);                          // 10us for TRIG width
    _SR04A_TRIG_Low;

  TIMER2->TEXCON.RSTCAPSEL = 1;
// set for rising edge trigger to reset counter
```

```
}

void init_GPIO_SR04(void)
{
    //Ultrasonic I/O Pins Initial
    GPIOB->PMD.PMD2 = 0;
 //_SR04_ECHO pin, Input
    GPIOB->PMD.PMD4 = 1;              //_SR04_TRIG pin, Output
    _SR04A_TRIG_Low;                  // set Trig output to Low
}

//------------------------------
// MAIN function
//------------------------------
int main (void)
{
    char     TEXT2[16] = "SR04a: ";
    int i;                  // loop index
    uint32_t distance_mm;

    //System Clock Initial
    UNLOCKREG();
    DrvSYS_SetOscCtrl(E_SYS_XTL12M, ENABLE);
    while(DrvSYS_GetChipClockSourceStatus(E_SYS_XTL12M) == 0);
    DrvSYS_Open(50000000);
    LOCKREG();

    init_LCD();                  // initialize LCD
    clear_LCD();                  // clear LCD display
    print_Line(0, "Smpl_Timer_SR04 "); // Line 0 display

    init_TMR2();                      // initialize Timer2 Capture
    init_GPIO_SR04();

    while(1) {
        distance_mm = 0;
        for (i = 0; i < 5; i++) {
            SR04_Trigger();       // Trigger Ultrasound Sensor for 10us
            DrvSYS_Delay(40000);   // Wait 40ms for Echo to trigger int
errupt
            if (SR04A_Echo_Flag == TRUE) {
                SR04A_Echo_Flag = FALSE;
                distance_mm += SR04A_Echo_Width * (340 / 2) / 1000;
            }
        }
        distance_mm /= 5;
        sprintf(TEXT2+6, " %4d mm  ", distance_mm);
        print_Line(1, TEXT2);               //Line 2: distance [mm]
```

```
        DrvSYS_Delay(10000);
    }
}
```

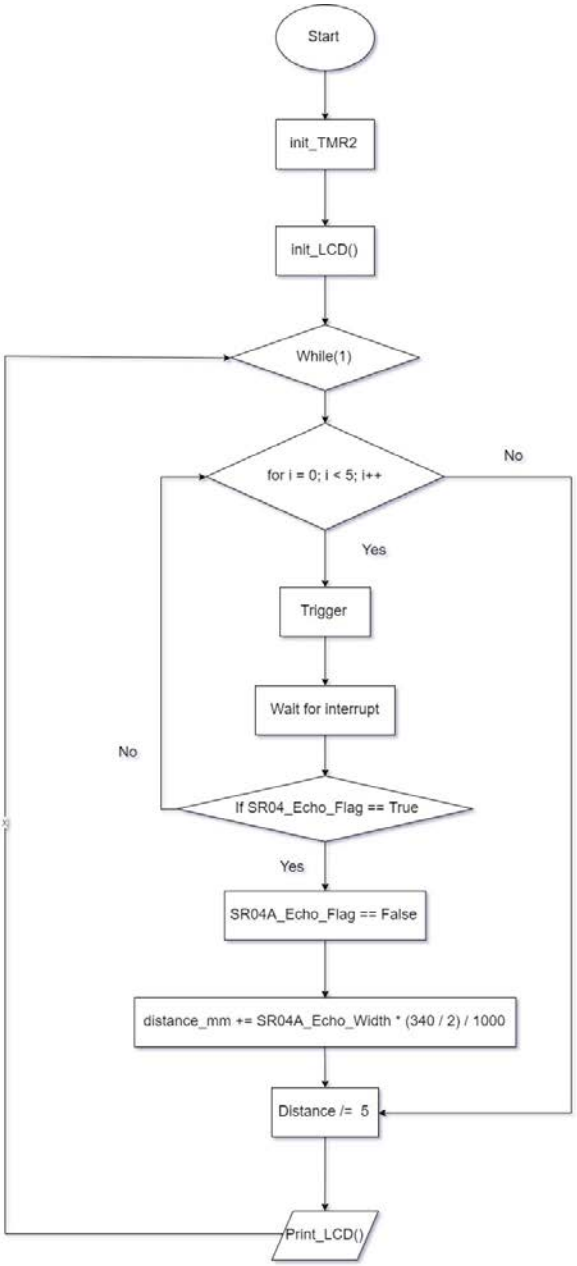
Question:

I. Design & implement a “different” method (the way of using the timer, e.g., using modes other than one-shot) from the one in the example code of measuring the echo pulse width of SR04. You should study the Technical Reference Manual and come up with a different idea of using the timer or whatever to measure the echo pulse width. Then implement your idea into code. Explain your method, code, and result in report and **demo your result to TA**.

Design Concept:

The main function is similar to what it is like in first problem. In this case, we choose to use the periodic method for counting. The only difference is that periodic will automatically start once the system is interrupted. Hence the only difference is that we don’t need to make CEN == 1 inside the IRQ. The other part of the program remains the same.

Program Flow (Similar to previous):



Code:

```
//
// Smpl_Timer_SR04 : using one SR04 Ultrasound Sensors
//
// Timer Capture :
// GPB2 / RTS0 / T2EX (NUC140 pin34)
// GPB4 / RX1         (NUC140 pin19)

// SR04 Ultrasound Sensor
// pin1 Vcc : to Vcc
// pin2 Trig: to GPB4      (NUC140VE3xN pin19)
// pin3 ECHO: to GPB2/T2EX (NUC140VE3xN pin34)
// pin4 Gnd : to GND

#include <stdio.h>
```

```c
#include "NUC1xx.h"
#include "GPIO.h"
#include "SYS.h"
#include "LCD.h"

// Global definition
#define _SR04A_ECHO        (GPB_2)          //NUC140VE3xN, Pin19
#define _SR04A_TRIG        (GPB_4)          //NUC140VE3xN, Pin34
#define _SR04A_TRIG_Low  (GPB_4=0)
#define _SR04A_TRIG_High (GPB_4=1)

// Global variables
volatile uint32_t SR04A_Echo_Width = 0;
volatile uint32_t SR04A_Echo_Flag  = FALSE;
//
// Initial TMR2
//
// Timer Clock: 12 MHz
// Prescale:        11
// Compare:     0xffffff
// Mode:            One-Shot mode
// Capture:     Enable, Capture with Falling Edge
void init_TMR2(void)
{
    //Step 1: T2EX pin Enable (PB.2, Pin34)
    SYS->GPBMFP.UART0_nRTS_nWRL = 1;
    SYS->ALTMFP.PB2_T2EX = 1;

  //Step 2: Timer Controller Reset and Setting Clock Source
    SYS->IPRSTC2.TMR2_RST = 1;          //Timer Controller: Reset
    SYS->IPRSTC2.TMR2_RST = 0;          //Timer Controller: Normal
    SYSCLK->CLKSEL1.TMR2_S = 0;         //Timer Clock = 12 MHz
    SYSCLK->APBCLK.TMR2_EN = 1;         //Timer C lock Enable

    //Step 3: Timer Controller Setting
    //  TMR0_T = (12 MHz / (11+1) / 1000000)^-1 = 1.000 Second
    TIMER2->TCMPR = 0x000f00;      //Timer Compare Value:  [0~16777215]
    TIMER2->TCSR.PRESCALE = 11;         //Timer Prescale:       [0~255]
    TIMER2->TCSR.MODE = 1;         //Timer Operation Mode: Periodic

    //Step 4: External Capture Mode Setting
    TIMER2->TEXCON.TEXEN = 1;        //External Capture Function Enable
    TIMER2->TEXCON.RSTCAPSEL = 0;   //Capture Mode Select: Capture Mode
    TIMER2->TEXCON.TEX_EDGE = 2;      //Capture Edge: Rising & Falling

    //Step 5: Timer Interrupt Setting
//  TIMER2->TCSR.IE = 1;                    //Timeout Interrupt Enable
//  TIMER2->u32TISR |= 0x01;              //Clear Timeout Flag (TIF)
```

```
    TIMER2->TEXCON.TEXIEN = 1;            //Capture Interrupt Enable
    TIMER2->u32TEXISR |= 0x01;            //Clear Capture Flag (TEXIF)
    NVIC_EnableIRQ(TMR2_IRQn);            //Timer NVIC IRQ Enable

    //Step 6: Start Timer Capture (Set by Ultrasonic_Trigger() Function)
//  TIMER2->TCSR.CRST = 1;                //Timer Counter Reset
        TIMER2->TCSR.CEN = 1;                     //Timer Start
}

// TMR2 Interrupt Handler
void TMR2_IRQHandler(void)
{
    TIMER2->TEXCON.RSTCAPSEL = 0;
// set back for falling edge to capture
//  TIMER2->TCSR.CEN = 1;                         //Timer Start
    if(TIMER2->TEXISR.TEXIF == 1)         //Capture Flag (TEXIF)
    {
        TIMER2->u32TEXISR |= 0x01;         //Clear Capture Flag (TEXIF)
        SR04A_Echo_Width = TIMER2->TCAP;
//Load Capture Value (Unit: us)
        SR04A_Echo_Flag  = TRUE;
    }
}

// Ultrasonic Trigger
void SR04_Trigger(void)
{
    //Trigger of Ultrasonic Sensor
    _SR04A_TRIG_High;
    DrvSYS_Delay(10);                              // 10us for TRIG width
    _SR04A_TRIG_Low;

  TIMER2->TEXCON.RSTCAPSEL = 1;
      // set for rising edge trigger to reset counter
}

void init_GPIO_SR04(void)
{
    //Ultrasonic I/O Pins Initial
    GPIOB->PMD.PMD2 = 0;  //_SR04_ECHO pin, Input
    GPIOB->PMD.PMD4 = 1;        //_SR04_TRIG pin, Output
    _SR04A_TRIG_Low;              // set Trig output to Low
}

//------------------------------
// MAIN function
//------------------------------
int main (void)
```

```
{
    char     TEXT2[16] = "SR04a: ";
    int i;                  // loop index
    uint32_t distance_mm;

    //System Clock Initial
    UNLOCKREG();
    DrvSYS_SetOscCtrl(E_SYS_XTL12M, ENABLE);
    while(DrvSYS_GetChipClockSourceStatus(E_SYS_XTL12M) == 0);
    DrvSYS_Open(50000000);
    LOCKREG();

    init_LCD();                 // initialize LCD
    clear_LCD();                 // clear LCD display
    print_Line(0, "Smpl_Timer_SR04 "); // Line 0 display

    init_TMR2();                    // initialize Timer2 Capture
    init_GPIO_SR04();

    while(1) {
        distance_mm = 0;
        for (i = 0; i < 5; i++) {
            SR04_Trigger();       // Trigger Ultrasound Sensor for 10us
            DrvSYS_Delay(40000);   // Wait 40ms for Echo to trigger int
errupt
            if (SR04A_Echo_Flag == TRUE) {
                SR04A_Echo_Flag = FALSE;
                distance_mm += SR04A_Echo_Width * (340 / 2) / 1000;
            }
        }
        distance_mm /= 5;
        sprintf(TEXT2+6, " %4d mm  ", distance_mm);
        print_Line(1, TEXT2);               //Line 2: distance [mm]
        DrvSYS_Delay(10000);
    }
}
```

Discussion:

這次題目並不難，題目要求取五次平均基本上只要稍微修改一下老師提供的範例 code 即可。不過在做 Question 時要求使用不同的 mode 來做，這時有些小小的發現。剛開始測試時，我想看 periodic mode 之下是不是真的不需要在 interrupt 中再次 enable counter. 當我把 interrupt 中 CEN == 1 的指令註解掉，我發現在 periodic mode 之下竟然完全不會動。後來才發現我從第一題裡面複製的 code，init()裡面根本沒有將 CEN enable 的指令。所以在 periodic 的模式之下我們必須在初始化時就將 timer 打開，他才會不斷的中斷，然後開始數。

但到這邊也讓我想到，在 One-Shot 模式下不需要透過手動 enable counter 便會自己開始數。理由是因為只要超音波收到回傳的訊號就會進入 interrupt 裡面，裡面就有 enable 的指令。