ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Điện Tử - Viễn Thông

BÁO CÁO
THỰC HÀNH VI ĐIỀU KHIỂN

Sinh viên thực hiện: Hồ Viết Đức Huy
Mã số sinh viên: 20200218
Lớp: 20DTV_hè
Mã học phần: ETC10010
Giảng viên hướng dẫn: ThS. Trần Tuấn Kiệt

Năm học: 2021 – 2022

BÀI 1 – LẬP TRÌNH VI ĐIỀU KHIỂN PIC VÀ I/O PORT

<u>Bài tập 1:</u> Tiến hành viết code để đọc giá trị từ nút nhấn S3 và S4, khi S3 được nhấn thì sẽ đếm lên, S4 được nhấn thì sẽ đếm xuống và giá trị đếm sẽ được xuất ra LED

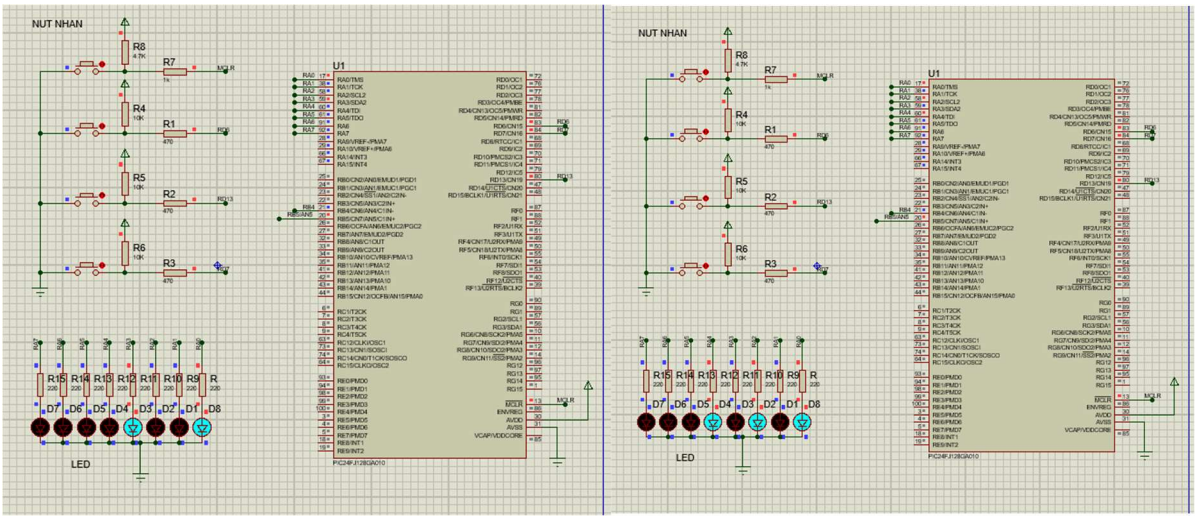
1. **<u>Phần 1 + 2:</u>** Code + giải thích code (ở phần comment của đoạn code):

```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = FRCDIV           // Oscillator Select (Fast RC Oscillator with Postscaler (FRCDIV))

#pragma config IESO = ON                // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768          // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128            // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON              // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON              // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2               // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF               // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF                // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF             // JTAG Port Enable (JTAG port is disabled)


// #pragma config statements should precede project file includes.
```

```
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <p24fj128ga010.h>                  //Khai báo thư viện của PIC24FJ128GA010

void MSdelay(unsigned int val)
{
    unsigned int i,j;
      for(i=0;i<val;i++)
         for(j=0;j<165;j++);
 }                                           // Hàm Delay với đơn vị ms (milisecond)
int main() {
   TRISD = 0xffff;                           //Cho các chân PORT D là INPUT (button)
   PORTD = 0xffff;                           // Cho các chân PORT D lên 1.(khi nhấn nút là
                                             mức 0).
   TRISA = 0;                                //Cho các chân PORT A là OUTPUT (LED).
   PORTA = 0;                                // Cho các chân PORT A là 0.
 while (1) {                                 //Đặt vòng lặp while với điều kiện là 1 để nó lặp vô
                                             hạn.
      if ((PORTD & 0x0040) == 0)             //AND PORTD với 0100 nếu nút nhấn thì 0000
                                             AND 0100 = 0 thì thực hiện trong if ngược lại thì
                                             0100 AND 0100 != 0 thì bỏ qua lệnh trong if.
      {
         PORTA = PORTA + 1;                  // PORTA (8 LED) thực hiện chức năng đếm lên.
         MSdelay(1000);                      // Chờ 1s để có thể quan sát LED.
      }
      if ((PORTD & 0x2000) == 0)             //AND PORTD với 0010 0000 0000 0000 nếu nút
                                             nhấn thì 0000 0000 0000 0000 AND 0010 0000
                                             0000 0000 = 0 thì thực hiện trong if ngược lại thì
                                             0010 0000 0000 0000 AND 0010 0000 0000 0000
                                             != 0 thì bỏ qua lệnh trong if.
```

```
    {
        PORTA = PORTA - 1;              //PORTA (8 LED) thực hiện đếm xuống.
        MSdelay(1000);                  // Chờ 1s để có thể quan sát LED.
    }
  }
  return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- LED thể hiện qua 8 bit với bit 0 là LED tắt còn 1 là LED sáng.
- Khi nhấn nút thứ 2 thì sẽ tăng lên 1. Ví dụ, lúc đầu LED mang giá trị 0000 0000 thì khi nhấn nút thì sẽ lên 0000 0001 tiếp khi nhấn 1 lần nữa thì sẽ cộng thêm 1 nữa 0000 0010 tương ứng như vậy khi nhấn thì cộng thêm 1 nữa 0000 0011, 0000 0100, 0000 1000, ….. ,1111 1111.
- Khi nhấn nút thứ 3 thì sẽ giảm xuống 1. Ví dụ, LED mang giá trị 0000 1111 thì khi nhấn nút S4 (RD13) thì sẽ giảm xuống 0000 1110 tiếp khi nhấn 1 lần nữa thì sẽ giảm thêm 1 nữa đó là 0000 1101 tương ứng như vậy sẽ giảm mãi khi về 0000 0000.

Bài tập 2: Tiến hành viết code để đọc giá trị từ nút nhấn S3 và S4, khi S3 được nhấn thì LED sẽ sáng dần, S4 được nhấn thì LED sẽ tối dần.

1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code):

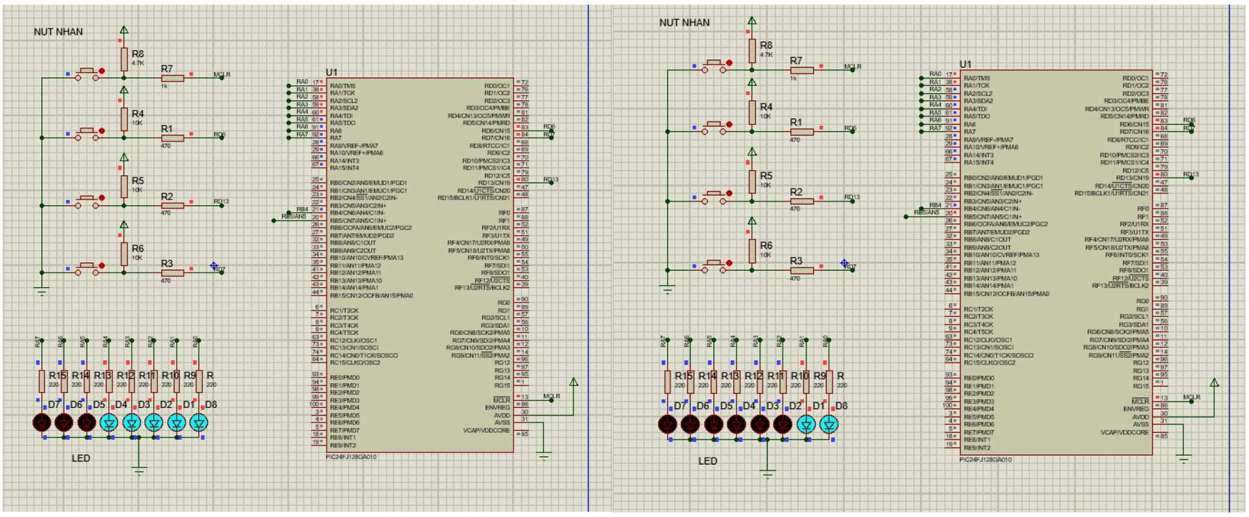
```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator
disabled)
```

```
#pragma config OSCIOFNC = OFF          // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD          // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = FRCDIV          // Oscillator Select (Fast RC Oscillator with Postscaler (FRCDIV))

#pragma config IESO = ON               // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768         // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128           // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON             // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON             // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2              // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF              // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF               // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF            // JTAG Port Enable (JTAG port is disabled)


// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.


#include <xc.h>

#include <p24fj128ga010.h>                  //Khai bao thu vien cua PIC24FJ128GA010


void MSdelay(unsigned int val)
```

```
{
    unsigned int i,j;
      for(i=0;i<val;i++)
         for(j=0;j<165;j++);
}                                          // Hàm Delay với đơn vị ms (milisecond).
int main() {
   TRISD = 0xffff;                         //Cho các chân PORT D là INPUT (button).
   PORTD = 0xffff;                         //Cho các chân PORT D lên 1.
   TRISA = 0;                              //Cho các chân PORT A là OUTPUT (LED).
   while (1) {
      if ((PORTD & 0x0040) == 0)           //AND PORTD với 0100 nếu nút nhấn thì 0000
                                           AND 0100 = 0 thì thực hiện trong if ngược lại thì
                                           0100 AND 0100 != 0 thì bỏ qua lệnh trong if.
      {
         PORTA = 0x01;                     //Cho các chân PORT A với RA0 = 1 còn lại là 0.
         int i;
         for (i = 0; i < 8; i++)           //Vòng lặp for để LED sáng dần từ RA0 -> RA7.
         {
            PORTA = (PORTA << 1) | 0x01;   //LED sẽ dịch trái và OR với 1 để thể hiện sáng
                                           dần.Ví dụ ban đầu là RA0 sáng sau đó dịch phải thì
                                           RA1 sáng và OR với 1 để RA0 cũng sáng tương tự
                                           vậy sau khi dịch trái lần nữa thì RA2 và RA1 sáng
                                           và OR 1 để RA0 cũng sáng theo.
            MSdelay(1000);                 // Chờ 1s để có thể quan sát LED
         }
      }
      if ((PORTD & 0x2000) == 0)           //AND PORTD với 0010 0000 0000 0000 nếu nút
                                           nhấn thì 0000 0000 0000 0000 AND 0010 0000
                                           0000 0000 = 0 thì thực hiện trong if ngược lại thì
                                           0010 0000 0000 0000 AND 0010 0000 0000 0000
                                           != 0 thì bỏ qua lệnh trong if.
      {
```

```
        int i;
        for (i = 0; i < 8; i++)          //Vòng lặp for để LED tắt dần từ RA7 -> RA0.
        {
          PORTA = PORTA >> 1;            //Dịch phải
          MSdelay(1000);                 //Chờ 1s để có thể quan sát LED
        }
      }
    }
    return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- Lúc đầu, cho 8 LED tương ứng với 8 chân RA0 -> RA7 của PORTA tắt hết.
- Khi nhấn nút thứ 2, cho bit RA0 = 1 để lần dịch trái bit tiếp theo cũng sáng đồng thời bit trước đó cũng là 1. Khi 8 LED đều sáng hết, giữ nguyên trạng thái sáng đó đợi khi được nhấn nút thứ 3.
- Khi nhấn nút thứ 3, khi 8 LED đang sáng đồng thời thực hiện dịch phải 1 bit thì LED sẽ tắt dần từ RA7 -> RA0 đến khi tắt hết. Lúc này, đợi nút nhấn thứ 2 được nhấn sẽ thực hiện sáng dần lại.

BÀI 2. TIMER

<u>Bài tập 1:</u> Lập trình để hệ thống đếm lên sau 500 ms sử dụng Timer 2 ở chế độ định thời, xuất kết quả ra led.

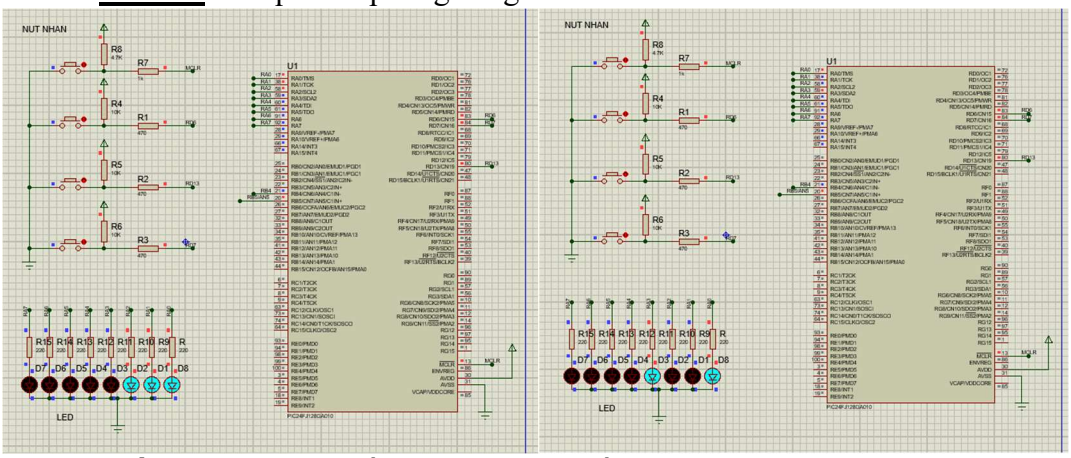
1. **<u>Phần 1 + 2:</u>** Code + giải thích code (ở phần comment của đoạn code):

```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = PRI              // Oscillator Select (Primary Oscillator (XT, HS, EC))

#pragma config IESO = ON                // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768          // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128            // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON              // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = OFF             // Watchdog Timer Enable (Watchdog Timer is disabled)

#pragma config ICS = PGx2               // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF               // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF                // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = ON              // JTAG Port Enable (JTAG port is enabled)


// #pragma config statements should precede project file includes.
```

```
// Use project enums instead of #define for ON and OFF.


#include <xc.h>
#include <p24fj128ga010.h>


void Timer2_Init ()                     //Hàm khai báo sử dụng TIMER2
{
   T2CON = 0x00;                        //Dừng Timer2 và reset lại thanh ghi điều khiển.
   TMR2 = 0x00;                         //Xoá nội dung của Timer2
                //F là tần số của bộ Timer đang dùng.
                //PRx là thanh ghi chu kì.
                //Thời gian delay cần dùng là 500ms = 0.5s với Prescale = 64 và F = 4Mhz
                //Công thức Tdelay = (PRx * Prescale) / F
                //Từ đó, ta có PR2 = 31250.
   PR2 = 31250;                         //Thiết lập thanh ghi chu kì với giá trị 31250
   T2CONbits.TCS = 0;                   //Cấu hình Clock trong (FOSC/2)
   T2CONbits.TCKPS = 0b10;              //Cấu hình 1:64 prescale value
   T2CONbits.TON = 1;                   // Bắt đầu hoạt động 16 bit Timer2
                //Vì Timer2 có thể sử dụng kết hợp với Tmer4 để tạo thành cặp timer 32 bit nên
                việc cấu hình thanh ghi TON = 1 để thực hiện cấu hình Timer2 hoạt động 16 bit.
}
int main(void) {
   TRISA = 0;                           //Cho các chân PORT A là OUTPUT (LED).
   PORTA = 0;                           //Cho cho các chân PORTA là 0 (LED).
   Timer2_Init ();                      //Gọi hàm Timer2.
   while (1)
   {
        if (TMR2 == PR2) {              //Thanh ghi TMR==PR(chu kì) tương ứng với 0,5s.
```

```
        PORTA++;                    //Đếm lên xuất ra LED.
        TMR2 = 0;                   //Xóa nội dung của Timer2 cho lần đếm tiếp theo.
    }
  }
  return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- Việc đếm lên rất đơn giản để thực hiện, chỉ cần cộng thêm 1 và thể hiện cái bit đó ra 8 LED ứng với RA0 -> RA7 của PORTA. Ví dụ, 0000 0000 đếm lên sẽ là 0000 0001 (LED0 = RA0 sáng còn lại tắt), 0000 0010,….., 1111 1111.
- Khi TMR==PR có nghĩ là khi chu kì máy chạy đúng số chu kì đã thiết lập trước đó thì nó sẽ chạy chu kì đó trong khoảng thời gian mà ta có thể tính toán được như công thức ở trên code.

Bài tập 2: Lập trình để hệ thống thỏa mãn yêu cầu sau: Khi nhấn S3 1 lần thì LED sáng dần 20%, khi sáng cực đại mà vẫn nhấn nữa thì LED sẽ tắt rồi sáng dần lên tương ứng với số lần nhấn của S3

1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code):

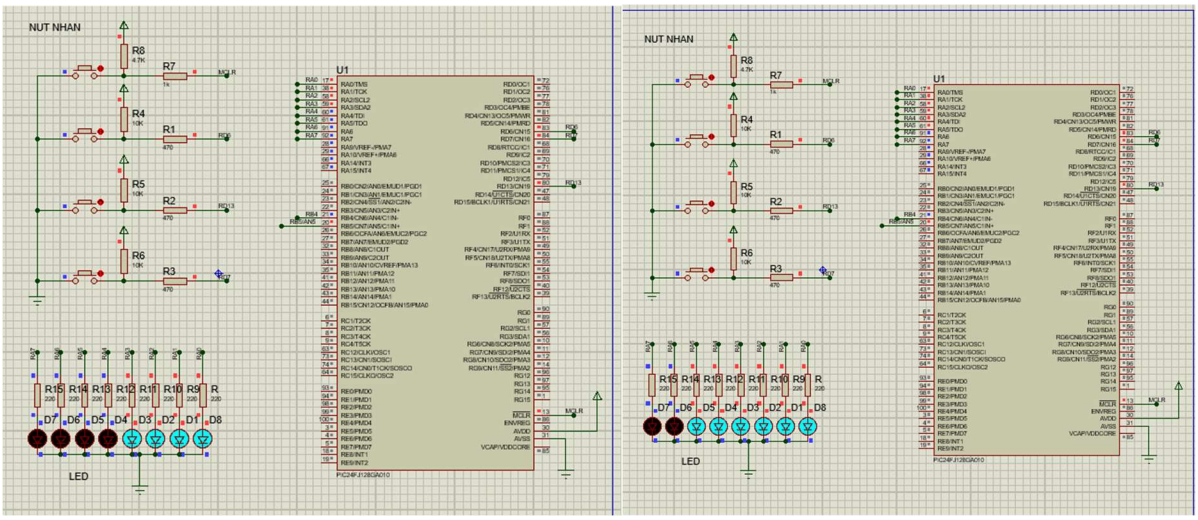
```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)
```

```
#pragma config FNOSC = PRI          // Oscillator Select (Primary Oscillator (XT, HS, EC))
#pragma config IESO = ON            // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)

// CONFIG1
#pragma config WDTPS = PS32768      // Watchdog Timer Postscaler (1:32,768)
#pragma config FWPSA = PR128        // WDT Prescaler (Prescaler ratio of 1:128)
#pragma config WINDIS = ON          // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))
#pragma config FWDTEN = OFF         // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config ICS = PGx2           // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)
#pragma config GWRP = OFF           // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF            // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = ON          // JTAG Port Enable (JTAG port is enabled)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.
#include <xc.h>
#include <p24fj128ga010.h>

void Timer1_Init(){                 //Hàm khai báo sử dụng TIMER1
   T1CON = 0x00;                    // Dừng Timer1 và reset lại thanh ghi điều khiển
   TMR1 = 0;                        //  Xoá nội dung của Timer1
            //Thời gian delay cần dùng là 1s với Prescale = 64 và F = 4Mhz
            //Công thức Tdelay = (PRx * Prescale) / F
            //Từ đó, ta có PR1 = 62500
```

```
    PR1 = 62500;                        // Thiết lập thanh ghi chu kì với giá trị 62500
    T1CONbits.TCS = 0;                  // Cấu hình Clock trong (FOSC/2)
    T1CONbits.TCKPS = 2;                // Cấu hình 1:64 prescale value
    T1CONbits.TON = 1;                  // Bắt đầu hoạt động 16 bit Timer1
}

int main() {
    TRISA = 0;                          // Cho các chân PORT A là OUTPUT (LED).
    TRISDbits.TRISD6 = 1;               // Cho chân RD6 (nút S3)  của PORTD là INPUT
                                           (button).
    Timer1_Init();                      // Gọi hàm Timer1.
    PORTA = 0;                          //Ban đầu cho các LED tắt hết.
    while(1){
        if ((PORTA == 0xff) && (PORTDbits.RD6 == 0))  //Nếu đèn sáng hết và nút S3 đang
                                                         nhấn thì thực hiện if.
      {
        if(TMR1 == PR1){                //Cho delay 1s
            PORTA = 0;                  // Tắt hết các LED
            TMR1 = 0;                   // Xoá nội dung của Timer1
        }
      }
      if (PORTDbits.RD6 == 0)           //Nếu chỉ nút S3 nhấn mà đèn chưa sáng hết thì thực
                                           hiện if.
      {
        if(TMR1 == PR1){                //Cho delay 1s
            PORTA = (PORTA << 2) | 0x03; // câu lệnh sáng dần LED 20%(dịch 2 LED)và OR
                                            để sáng dần(như bài 1 câu 2).
            TMR1 = 0;                   // Xoá nội dung của Timer1
        }
```

```
    }
}
  return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus

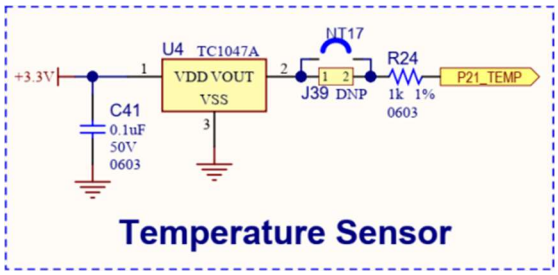


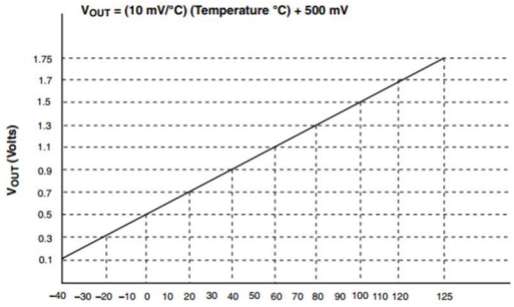
3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus

- Khi các LED đều sáng (tức là sáng cực đại) mà vẫn nhấn nút thì LED sẽ tắt hết
- Sau đó, khi nhấn nút thì LED sẽ sáng tương ứng với số lần nhấn nút thứ 2(S3) (với 1 lần nhấn nút S3 là LED sáng dần 20%)

BÀI 3. ADC

<u>Bài tập 1:</u> Trên board Explorer 16 có cảm biến nhiệt độ được kết nối như hình 2.13. Công thức tính nhiệt độ và đồ thị biểu diễn nhiệt độ theo điện thế ngõ ra được thể hiện trong hình 2.13. Tiến hành viết chương trình C để thực hiện đọc nhiệt độ từ cảm biến nhiệt độ sau mỗi 1 giây và ghi kết quả ra LED.





1. **<u>Phần 1 + 2:</u>** Code + giải thích code (ở phần comment của đoạn code):

// CONFIG2

#pragma config POSCMOD = NONE // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = FRCDIV // Oscillator Select (Fast RC Oscillator with Postscaler (FRCDIV))

#pragma config IESO = ON // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)

// CONFIG1

#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128 // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

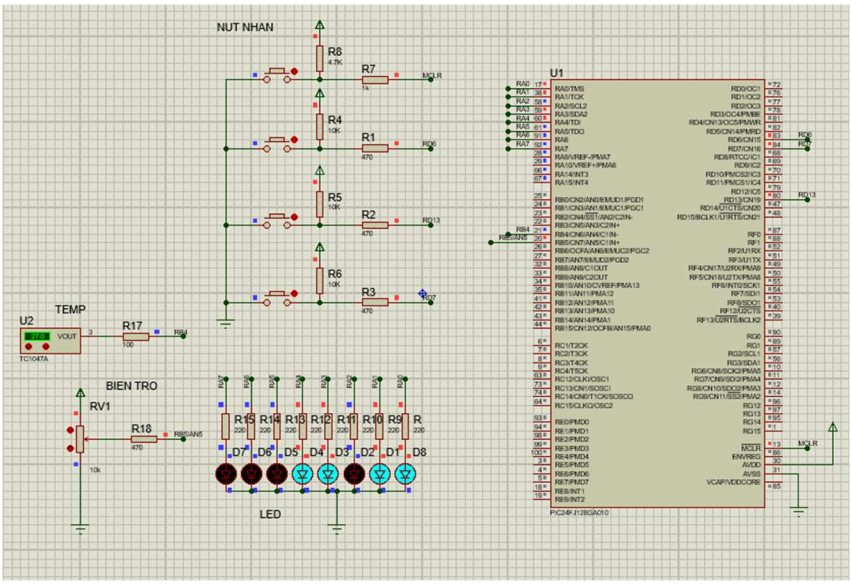
#pragma config FWDTEN = ON // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2 // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

```
#pragma config GWRP = OFF // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF // JTAG Port Enable (JTAG port is disabled)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.


#include <xc.h>

#include "p24fj128ga010.h"

#include <stdio.h>

#define POT 5                   // điện trở ở chân AN5 thay bằng chữ POT

#define TEMP 4                  //cảm biến nhiệt độ ở chân AN4 thay bằng chữ TEMP


void initADC(int pinNumber) {
   AD1PCFGbits.PCFG4 = pinNumber;   //Thanh ghi cấu hình ADC, đọc giá trị của I/O ports
   AD1CON1 = 0x00E0;                // Thanh ghi SSRC = 111 (bit 5 -> 7 của thanh ghi
                                    AD1CON1 là SSRC0 -> SSRC2).
   AD1CSSL = 0;                     // kênh analog là bỏ qua từ quét đầu vào (CSSL0).
   AD1CON2 = 0;                     // Thanh ghi ở bit 13 -> bit 15 của thanh ghi
                                    AD1CON2.
                                    // NVCFG0 = 0 -> AVSS
                                    // PVCFG = 00 -> AVDD
   AD1CON3 = 0x1F02;                // Cấu hình clock cho bộ ADC
                                    //thanh ghi ADCS của AD1CON3  <7:0> = 2 Tcy
                                    //thanh ghi SAMC<4:0> = 31 Tad
   AD1CON1bits.ADON = 1;            //Bộ ADC đang hoạt động
}
```

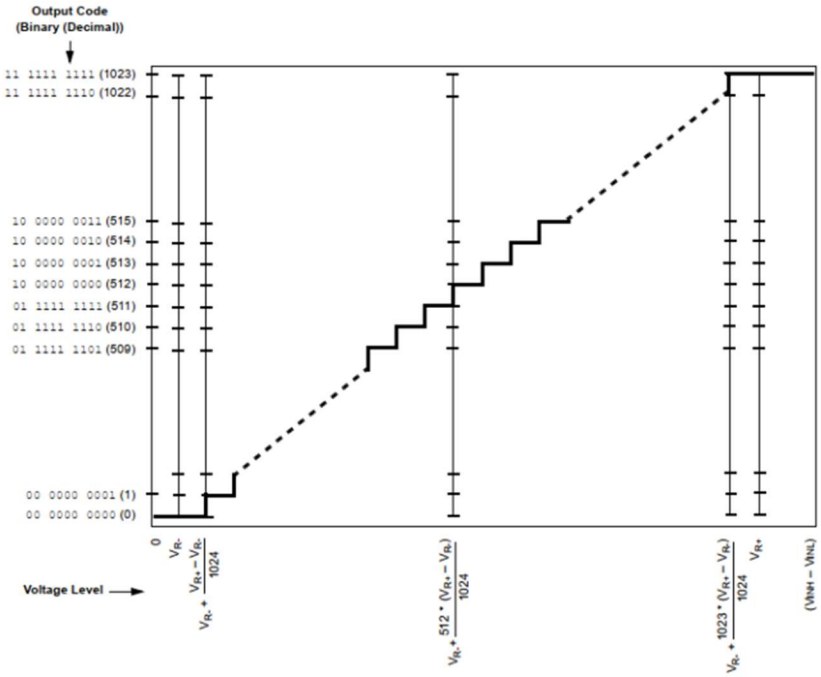
```
int readADC(int ch) {
   AD1CHS = ch;                          // 1. select analog input channel
   AD1CON1bits.SAMP = 1;                 // 2. start sampling
   while (!AD1CON1bits.DONE);            // 3. wait for the conversion to complete
   return ADC1BUF0;                      // 4. read the conversion result
}

int main(void) {
   initADC(TEMP);                        //Khai báo ADC đưa vào hàm main và gán biến
                                         TEMP ( cảm biến nhiệt độ)
   TRISA = 0;                            //Cho các chân PORT A là OUTPUT (LED).
   PORTA = 0;                            // Tắt hết LED
   int a = 0;                            // Khai báo biến a để đọc dữ liệu ADC
   float vout = 0.0;                     // Biến thể hiện giá trị của cảm biến nhiệt độ
   int temp;                             // Biến quy đổi thể hiện qua Volt kế
    while (1) {
      a = readADC(TEMP);                 //Đọc giá trị ADC của cảm biến nhiệt độ
                  //Công thức chuyển đổi: Vout = (Code × (VR+ − VR−)) /1024
                  //Ta có công thức: Vout = (10mV/oC) x (Temparature oC) + 500mV
      vout = (5 * (float)(a)) /1024;
      temp = (vout * 1000 - 500.0) /10.0;
      PORTA = temp;
   }
    return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- Lúc mô phỏng cảm biến nhiệt độ (TC1047A), ta phải thực hiện việc chuyển đổi thông qua hàm chuyển đổi: $Vout = (Code \times (VR+ - VR-)) / 1024$ đồng thời áp dụng công thức tính nhiệt độ đã đề cập ở trên để biến đổi thành công tức tính nhiệt độ
- Dựa vào sơ đồ chuyển đổi:



- Từ đó, với nhiệt độ cảm biến nhiệt độ tương ứng sẽ xuất ra giá trị LED (dạng binary tương ứng)

BÀI 4. NGẮT – INTERRUPT

Bài tập 1: Cấu hình PIC24/32 để thực hiện ngắt trên Timer1 để bật tắt led sau mỗi 500 ms

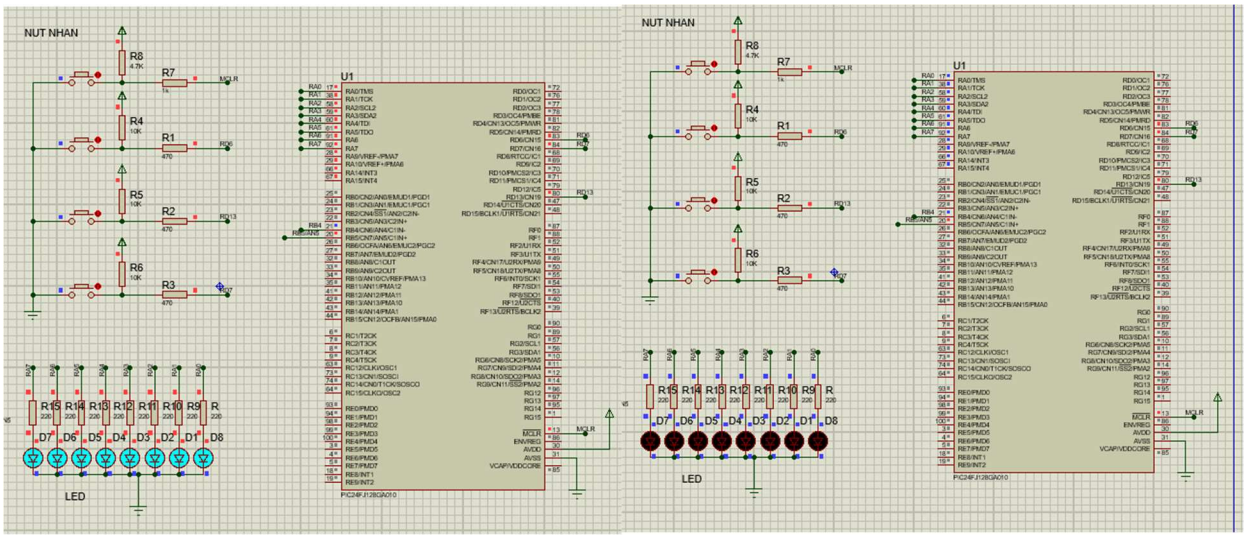
1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code):

```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = PRI              // Oscillator Select (Primary Oscillator (XT, HS, EC))

#pragma config IESO = ON                // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768          // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128            // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON              // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON              // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2               // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF               // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF                // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = ON              // JTAG Port Enable (JTAG port is enabled)


// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.
```

```
#include <xc.h>
#include <p24fj128ga010.h>

void Timer1_Init(void) {
    T1CON = 0x00;               //Dừng Timer1 và reset lại thanh ghi điều khiển.
    TMR1 = 0x00;                //Xoá nội dung của Timer1
                                //F là tần số của bộ Timer đang dùng
                                //PRx là thanh ghi chu kì
                                //Thời gian delay cần dùng là 500ms = 0.5s với Prescale = 64 và F = 4Mhz
                                //Công thức Tdelay = (PRx * Prescale) / F
                                //Từ đó, ta có PR1 = 31250
    PR1 = 31250;
    _T1IP = 0x05;               // giá trị ưu tiên ngắt khác 0 để có thể cho phép Timer sinh ra ngắt
    _T1IF = 0;                  // Bit được xóa bởi phần mềm (đưa xuống 0)
    _T1IE = 1;                  // bit Timer Interrupt Enable được đưa lên 1 để cho phép ngắt
                                timer1
    T1CONbits.TCS = 0;          // Cấu hình Clock trong (FOSC/2)
    T1CONbits.TCKPS = 2;        // Cấu hình 1:64 prescale value
    T1CONbits.TON = 1;          // Bắt đầu hoạt động Timer1
}

void _ISR _T1Interrupt(void) {      // Hàm ngắt Timer1(khi TMR1 tràn thì cờ bật(bằng 1) sau
                                    đó thực hiện ngắt.
    PORTA = ~PORTA;                 //Đảo trạng thái LED.
    _T1IF = 0;                      //Reset lại cờ ngắt.
}

int main(void) {
```

```
    TRISA = 0;                  //Cho các chân PORT A là OUTPUT (LED).
    PORTA = 0;                  //Cho PORTA tắt hết  ( tương ứng mức 0).
    Timer1_Init();              //Gọi hàm đã cấu hình Timer1
    while (1);                  //Treo chương trình để có thể thực hiện liên tục.
    return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- Sử dụng ngắt timer1 có nghĩa là khi giá trị thanh ghi TMR1 bị tràn lúc này cờ ngắt T1IF đưa lên 1 báo hiệu ngắt → đưa vào hàm ngắt thực hiện chức năng đảo trạng thái LED.
- Lúc đầu, các LED tắt hết. Cấu hình TIMER1 với thời gian delay là 500ms. Đợi sau 0.5s thì LED sáng (đảo trạng thái).
- Sau khoảng 0.5s, LED tiếp tục đảo trạng thái. Tương tự cho những khoảng 0.5s tiếp theo LED nhấp nháy liên tục do hàm while(1);

Bài tập 2: Cấu hình PIC24/32 để thực hiện ngắt trên ADC để đọc giá trị của cảm biến nhiệt độ và ghi ra LED

1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code):

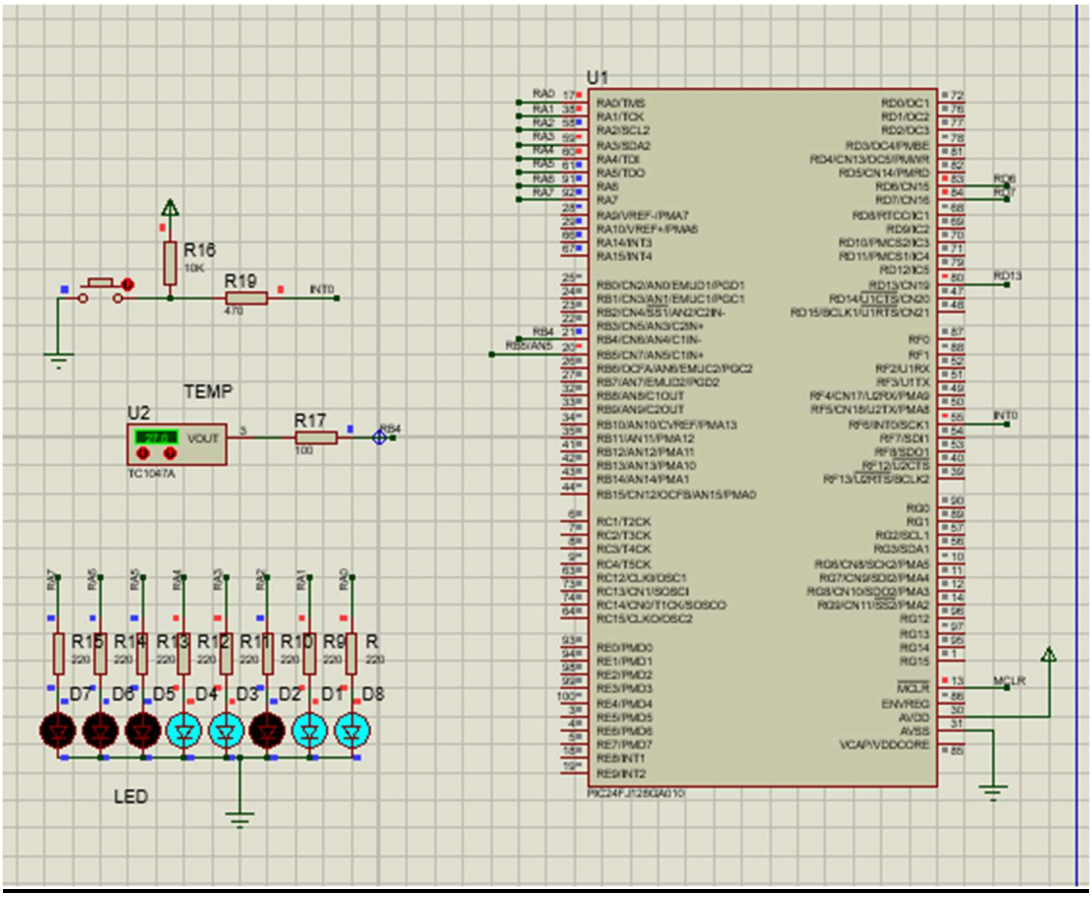
```
// CONFIG2

#pragma config POSCMOD = NONE           // Primary Oscillator Select (Primary oscillator
disabled)
```

```
#pragma config OSCIOFNC = OFF          // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD          // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = PRI             // Oscillator Select (Primary Oscillator (XT, HS, EC))

#pragma config IESO = ON               // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768         // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128           // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON             // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON             // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2              // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF              // General Code Segment Write Protect (Writes to program memory are allowed)

#pragma config GCP = OFF               // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF            // JTAG Port Enable (JTAG port is enabled)


// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.


#include <xc.h>

#include <p24fj128ga010.h>

#include <stdio.h>

#define TEMP 4                         // Temperature sensor connected to AN4 input
```

```
int code ;
float temp = 0.0;
float vout = 0.0;
void ADC_Init() {
   AD1PCFGbits.PCFG4 = 0;          //Thanh ghi cấu hình ADC, đọc giá trá của I/O ports.
   AD1CON1 = 0x00E0;               // Thanh ghi SSRC = 111 (bit 5 -> 7 của thanh ghi
                                   AD1CON1 là SSRC0 -> SSRC2).
   AD1CSSL = 0;                    // kênh analog là bỏ qua từ quét đầu vào (CSSL0)
   AD1CON2 = 0;                    // Thanh ghi ở bit 13 -> bit 15 của thanh ghi AD1CON2
                                   // NVCFG0 = 0 -> AVSS
                                   // PVCFG = 00 ->  AVDD
   AD1CON3 = 0x1F02;               // Cấu hình clock cho bộ ADC
                                   //thanh ghi ADCS c?a AD1CON3  <7:0> = 2 Tcy
                                   //thanh ghi SAMC<4:0> = 31 Tad
   AD1CON1bits.ADON = 1;           // Bộ ADC đang hoạt động
   _AD1IP = 3;                     //Cấu hình độ ưu tiên
   _AD1IF = 0;                     //Xóa cờ ngắt
   _AD1IE = 1;                     //Bật interrupt
}

void INT0_interrupt_Init()
{
   INTCON2bits.INT0EP = 1;         //Khi có kích cạnh xuống -> tạo ra ngắt
   _INT0IP = 5;                    //Cấu hình độ ưu tiên
   _INT0IF = 0;                    //Xóa cờ ngắt
   _INT0IE = 1;                    //Cho phép ngắt -> bật Interrupt
}
```

```
//Khi có sự kiện ngắt ( tức là có kích cạnh xuống - nút được nhấn)
//Thì INT) sẽ bật bit SAMP của ADC lên 1-> ADC có thể lấy mẫu
void _ISR _INT0Interrupt (void)     //hàm ngắt INT0 (khi nhấn nút thì SAMP của ADC bật và
                                    thực hiện chức năng của nó).
{
   _INT0IF = 0;                     // Xóa cờ ngắt.
   AD1CHS = TEMP;                   // Chọn kênh thứ 4 của ADC cho cảm biến nhiệt độ.
   AD1CON1bits.SAMP = 1;            // Khi ngắt xảy ra bit SAMPLE lên 1 (của ADC).
}

void _ISR _ADC1Interrupt(void) {
   _AD1IF = 0;                      //Xóa cờ ngắt
   code = ADC1BUF0;
                //Công thức chuyển đổi: Vout = (Code ×(VR+−VR−)) /1024
                //Ta có công thức: Vout = (10mV/oC)* (Temparature oC) + 500mV
   vout = (5 * (float)(code)) / 1024.0;
   temp = (vout * 1000 - 500.0) / (10.0) ;
   PORTA = temp;
}

int main(void) {
   TRISA = 0;                       //Cho các chân PORT A là OUTPUT (LED).
   INT0_interrupt_Init();           //Khai báo ngắt đưa vào hàm main kiểm tra BUTTON
   ADC_Init();                      //Khai báo ADC đưa vào hàm main và gán biến TEMP (
                                    cảm biến nhiệt độ)
   while (1);
   return 0;
}
```

2. **<u>Phần 3:</u>** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- Vì ngắt ADC khi được xảy ra thì phải đưa thanh ghi SAMP lên 1. Vì vậy, cần có thêm sự hỗ trợ của ngắt INT0 để kích hoạt SAMP lên 1 thông qua nút nhấn.
- Khi nhấn nút, thì ADC mới có thể bắt đầu lấy mẫu nên trước đó sẽ không thể quan sát được LED nếu chưa nhấn nút.
- Việc đọc giá trị cảm biến nhiệt độ đã trình bày ở bài ADC nên chỉ sử dụng lại những chi tiết đó. Sau khi nhấn nút, thì LED mới hiện lên giá trị đã đọc ở cảm biến TC1047A.
- Lưu ý khi điều chỉnh giá trị cảm biến nhiệt độ nếu không nhấn nút lần nữa thì giá trị sẽ không được xuất ra LED (hay thay đổi qua LED) vì ADC chưa lấy mẫu nên hàm ngắt ADC chưa xảy ra. Ví dụ, như hình trên cảm biến nhiệt độ mang giá trị 19 độ C thì LED thể hiện ở dạng 0001 0010, tuy nhiên khi điều chỉnh qua 25 độ C lúc chưa nhấn nút thì LED cũng sẽ không thay đổi giá trị
- Sau khi nhấn nút, thì LED mới thay đổi giá trị nhị phân của nó vì lúc này ngắt ADC đã được xảy ra.

BÀI 5. GIAO TIẾP UART

Bài tập 1 + 2: Thực hiện cấu hình ngắt mỗi khi nhận được giá trị cho UART1.

Thực hiện chương trình cấu hình UART để mỗi khi nhận được chuỗi "Hello" từ PC thì sẽ gửi lại chuỗi "My name is A", với A là tên sinh viên. Khi nhận được chuỗi khác chuỗi "Hello", thì gửi lại chuỗi "I do not know". Lưu ý có sử dụng ngắt UART.

Thực hiện chương trình cấu hình UART và ADC để mỗi khi nhận được chuỗi "temperature?" thì sẽ gửi lại giá trị nhiệt độ đọc được từ cảm biến nhiệt độ. Ví dụ nhiều độ là 28ºC thì sẽ gửi đến PC chuỗi kí tự là "28 doC"

1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code):

```
// CONFIG2

#pragma config POSCMOD = HS             // Primary Oscillator Select (HS Oscillator mode selected)

#pragma config OSCIOFNC = OFF           // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD           // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = PRI              // Oscillator Select (Primary Oscillator (XT, HS, EC))

#pragma config IESO = ON                // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)


// CONFIG1

#pragma config WDTPS = PS32768          // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128            // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON              // Watchdog Timer Window (Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON              // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2               // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF               // General Code Segment Write Protect (Writes to program memory are allowed)
```

```
#pragma config GCP = OFF            // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF         // JTAG Port Enable (JTAG port is disabled)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <p24fj128ga010.h>
#include <stdio.h>
#include <string.h>

#define CTS _RF12                   // Clear To Send, input, HW handshake
#define RTS _RF13                   // Request To Send, output, HW handshake
            //Ta có công thức tính: Baud rate = Fcy /(4 * (UxBRG  + 1)) với BRGH = 1
            //và công thức: UxBRG = (Fcy/ (4  * Baud rate)) - 1
            //Giả sử ta muốn Baud rate là 14400 -> U1BRG sẽ là 68
#define BRATE 68                    // 14400 Bd (BREGH=1)
#define U_ENABLE 0x8008             // enable UART, BREGH =1 , 1 stop, no parity
#define U_TX 0x0400                 // enable transmission, clear all flags
#define BACKSPACE 0x08              // ASCII backspace character code
#define BUF_SIZE 128
#define TEMP 4                      // Temperature sensor connected to AN4 input

char tr[BUF_SIZE];
int index = 0;
int flag = 0;
float temperature = 0.0;
```

```
void initU2(void) {
   U2BRG = BRATE;                  //Cấu hình tốc độ baud trên thanh ghi UBRG
   U2MODE = U_ENABLE;              //Cấu hình số lượng bit truyền đi là 8, cấu hình parity và
                                   stop bit
   U2MODEbits.BRGH = 1;            //Cấu hình tốc độ bằng BRGH
   U2STA = U_TX;                   //Cho phép UART2 hoạt động
   _TRISF13 = 0;
   _TRISF12 = 1;
   RTS = 1;
}

int putU2(int c) {                 // Hàm gửi từng kí tự
   while (CTS);                    //Kiểm tra chân CTS có ở mức 0 chưa, nếu CTS = 1 -> bên
                                   nhận đang bận.
   while (U2STAbits.UTXBF);        //Kiểm tra nếu bộ Tx đầy -> UTXBF = 0 -> While không
                                   thả điều kiện -> Thoát While
   U2TXREG = c;                    //Gửi dữ liệu
   return c;
}

void putsU2( char *s) {            //Hàm gửi dữ liệu gồm nhiều kí tự
   while ( *s)                     //loop until *s == ?\0?, end of string
      putU2 ( *s++);               //gửi ký tự và trỏ đến ký tự tiếp theo
}

char getU2(void) {                 //Hàm nhận dữ liệu 1 kí tự
   RTS = 0;                        //Xóa chân RTS về 0 để yêu cầu muốn nhận dữ liệu
```

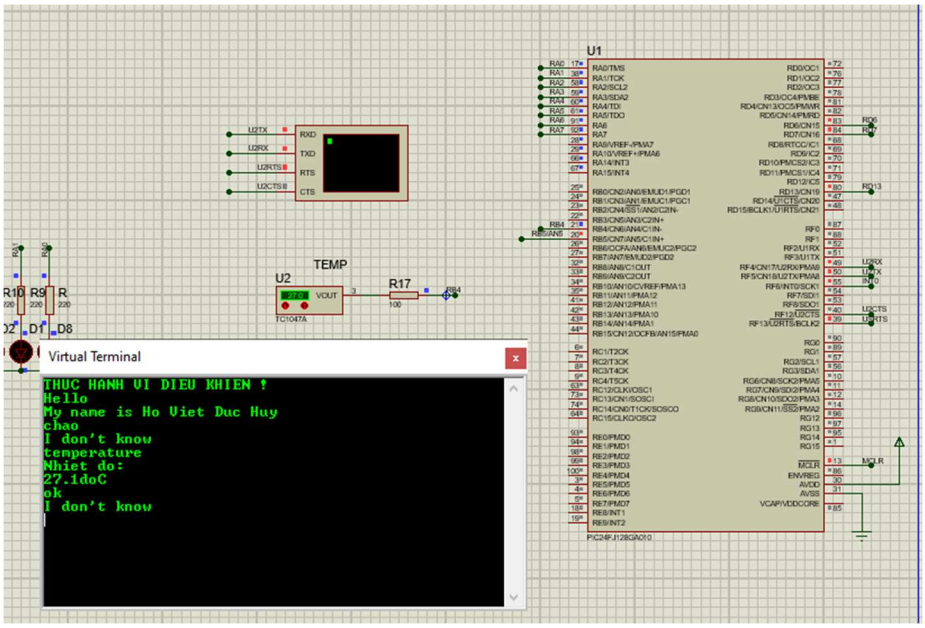
```
    while (!U2STAbits.URXDA);      //Khi URXDA = 0 -> FIFO đang rỗng -> Đợi đến khi nhận
                                   được dữ liệu -> URXDA = 1-> Thoát while
    RTS = 1;
    return U2RXREG;                //đọc dữ liệu từ bộ nhận
}

void U2_Interrupt_Init() {         //Hàm cấu hình ngắt UART2
    U2STAbits.URXISEL = 0;
    _U2RXIP = 4;                   //Cấu hình độ ưu tiên
    _U2RXIF = 0;                   //Xóa cờ ngắt
    _U2RXIE = 1;                   //Bật ngắt UART2
}

void _ISR _U2RXInterrupt(void) {   //Hàm ngắt của quá trình nhận trên UART2
    _U2RXIF = 0;                   //Xóa cờ ngắt
    char c = U2RXREG;
    tr[index] = c;
    putU2(tr[index]);              //gửi kí tự ngược lại
    index++;
    if (index > BUF_SIZE)
        index = 0;
    if (c == '\r') {
        if (strcmp(tr, "Hello\r") == 0) {
            putsU2("My name is Ho Viet Duc Huy\n\r");
        }
       else if (strcmp(tr, "temperature\r") == 0) {
            putsU2("Nhiet do: \r\n");
            int temp = (int)(temperature * 10);
```

```
            putU2(temp / 100 + 48);
            putU2((temp % 100) / 10 + 48);
            putU2('.');
            putU2((temp % 100) % 10 + 48);
            putsU2("doC\r\n");
        }
        else {
            putsU2("I don't know\n\r");
        }
        for (index = 0; index < BUF_SIZE; index++)
            tr[index] = '\0';
            index = 0;
        }
        PORTA = index;
}

char *getsnU2(char *s, int len) {       //Hàm nhận dữ liệu gồm nhiều kí tự
    char *p = s;                        // copy the buffer pointer
    int cc = 0;                         // character count
    do {
        *s = getU2();                   // wait for a new character
        if (( *s == BACKSPACE) && (s > p)) {
            putU2( ' ');                // overwrite the last character
            putU2( BACKSPACE);
            len++;
            s--;                        // back the pointer
            continue;
        }
```

```
        if ( *s == '\n')                // line feed, ignore it
            continue;
        if ( *s == '\r')                // end of line, end loop
            break;
        s++;                            // increment buffer pointer
        len--;
    } while (len > 1);                  // until buffer full
    *s = '\0';                          // null terminate the string
    return p;                           // return buffer pointer
}

void initADC() {
    AD1PCFGbits.PCFG4 = 0;
    AD1CON1 = 0x00E0;
    AD1CSSL = 0;
    AD1CON2 = 0;
    AD1CON3 = 0x1F02;
    AD1CON1bits.ADON = 1;
}

int readADC(int ch) {
    AD1CHS = ch;                        // 1. select analog input channel
    AD1CON1bits.SAMP = 1;               // 2. start sampling
    while (!AD1CON1bits.DONE);          // 3. wait for the conversion to complete
    return ADC1BUF0;                    // 4. read the conversion result
}

int main() {
```

```
    char s[BUF_SIZE];
    TRISA = 0;
    initU2();
    putsU2("THUC HANH VI DIEU KHIEN !\n\r");
    U2_Interrupt_Init();
    initADC();
    int code ;
    while (1)
    {
        code = readADC(TEMP);              //Đọc giá trị ADC của kênh thứ 4 -> biến TEMP
        temperature = (5 * (float)(code) * 1000 / 1024 - 500) / 10;      // hàm tính nhiệt độ từ cảm
                                                                          biến nhiệt độ
    }
    return 0;
}
```

2. **<u>Phần 3:</u>** Kết quả mô phỏng bằng Proteus:



3. **<u>Phần 4:</u>** Giải thích kết quả mô phỏng bằng Proteus

- Lúc đầu, cửa sổ Vitural Terminal sẽ hiện dòng chữ "THUC HANH VI DIEU KHIEN !" do ta đã mô tả trong hàm main. Nếu nhấn nút RESET BUTTON thì sẽ trả về dòng chữ "THUC HANH VI DIEU KHIEN !"
- Nếu nhập chữ "Hello" vào cửa sổ Terminal thì sẽ hiện ra dòng chữ "My name is Truong Minh Thanh" vì UART nhận dữ liệu và gửi lại dữ liệu hiện lên cửa sổ. Nếu nhập khác "Hello" thì sẽ hiện dòng chữ "I don't know"
- Nếu nhập chữ "temperature" thì sẽ nhận được giá trị của cảm biến nhiệt độ.
- Tất cả những kí tự nhập vào sẽ được xử lí trong ngắt, còn LED ở PORTA sẽ đưa ra số kí tự (index). Ví dụ, có 12 kí tự thì LED sẽ sáng 0000 1100,….

BÀI 6. GIAO TIẾP VỚI LCD

Bài tập: Lập trình để hệ thống hiển thị nhiệt độ môi trường trên LCD dạng thập phân có 1 chữ số ở phần thập phân sử dụng ngắt ADC. Kết quả hiển thị trên LCD: Nhiet do la: 27.5.

Thêm ký tự đặt biệt ºC vào kết quả trên. Kết quả hiển thị trên LCD: Nhiet do la: 27.5ºC.

1. **Phần 1 + 2:** Code + giải thích code (ở phần comment của đoạn code)

// CONFIG2

#pragma config POSCMOD = NONE // Primary Oscillator Select (Primary oscillator disabled)

#pragma config OSCIOFNC = OFF // Primary Oscillator Output Function (OSC2/CLKO/RC15 functions as CLKO (FOSC/2))

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Clock switching and Fail-Safe Clock Monitor are disabled)

#pragma config FNOSC = FRCDIV // Oscillator Select (Fast RC Oscillator with Postscaler (FRCDIV))

#pragma config IESO = ON // Internal External Switch Over Mode (IESO mode (Two-Speed Start-up) enabled)

// CONFIG1

#pragma config WDTPS = PS32768 // Watchdog Timer Postscaler (1:32,768)

#pragma config FWPSA = PR128 // WDT Prescaler (Prescaler ratio of 1:128)

#pragma config WINDIS = ON // Watchdog Timer Window Standard Watchdog Timer enabled,(Windowed-mode is disabled))

#pragma config FWDTEN = ON // Watchdog Timer Enable (Watchdog Timer is enabled)

#pragma config ICS = PGx2 // Comm Channel Select (Emulator/debugger uses EMUC2/EMUD2)

#pragma config GWRP = OFF // General Code Segment Write (Writes to program memory are allowed)

#pragma config GCP = OFF // General Code Segment Code Protect (Code protection is disabled)

#pragma config JTAGEN = OFF // JTAG Port Enable (JTAG port is disabled)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

```
#include <xc.h>
#include <p24FJ128GA010.h>
#define LCDDATA 1 // RS = 1 ; access data register
#define LCDCMD 0 // RS = 0 ; access command register
#define PMDATA PMDIN1 // PMP data buffer
//#define BusyLCD()  ReadLCD ( LCDCMD) & 0x80
//#define AddrLCD()  ReadLCD ( LCDCMD) & 0x7F
//#define getLCD()     ReadLCD ( LCDDATA)
#define putLCD( d) WriteLCD( LCDDATA, (d))
#define CmdLCD( c) WriteLCD( LCDCMD, (c))
#define HomeLCD()  WriteLCD( LCDCMD, 2)
#define ClrLCD()   WriteLCD( LCDCMD, 1)

int code = 0;
int check_lcd = 0;

void LCDinit(void) {
   // PMP initialization
   // PMP initialization
   PMCON = 0x83BF;                  //Cho giá trị nhị phân tương ứng với 16 bit của
                                    thanh ghi này là 1000 0011 1011 1111
   PMMODE = 0x3FF;                  //Cho thanh ghi chế độ với 0000 0011 1111 1111
   PMAEN = 0x0001;                  //Ở thanh ghi này chỉ quan tâm đến thanh ghi PMA0
                                    nên ta bật PMA0 = 1, còn lại là 0
                                    // init TMR1
   T1CON = 0x8030;                  // Fosc/2, 1:256 prescaler value, 16us/tick
                                    // wait for >30ms
   TMR1 = 0; while (TMR1 < 2000);   //2000 x 16us = 32ms
   PMADDR = LCDCMD;                 // command register
```

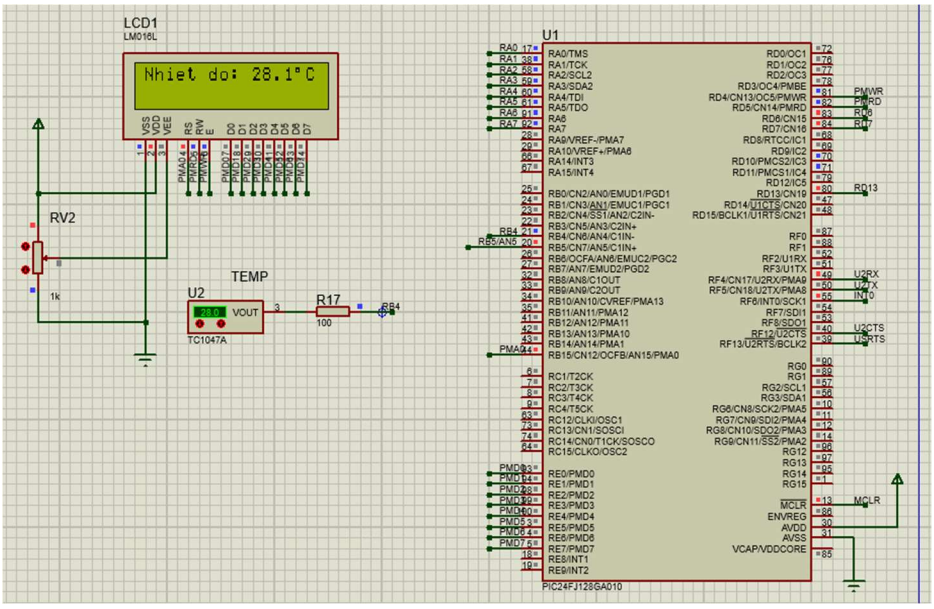
```
    PMDATA = 0b00111000;            //8 bits, 2 line, 5x7 ( chiều dài dữ liệu)
    TMR1 = 0; while (TMR1 < 1);     //3 x 16us = 32ms
    PMDATA = 0b00001100;            // ON< cursor off, blink off
    TMR1 = 0; while (TMR1 < 1);     //3 x 16us = 32ms
    PMDATA = 0b00000001;            //clear display
    TMR1 = 0; while (TMR1 < 26)
    PMDATA = 0b00000110;            // increment cursor, no shift
    TMR1 = 0; while (TMR1 < 26);
}

void WriteLCD (int addr, char c)
{
    PMADDR = addr;
    PMDATA = c;
    TMR1 = 0; while (TMR1 < 1);
}

void putsLCD(char* s) {
    while ( *s)
        putLCD( *s++);
} //putsLCD

void setDDRAM_ADDR (int address)
{
    PMADDR = LCDCMD;
    PMDATA = 0b10000000 | (0x7f & address);
}
```

```
void ADC_Init() {
   AD1PCFGbits.PCFG4 = 0;          //Thanh ghi cấu hình ADC, đọc giá trá của I/O ports
   AD1CON1 = 0x00E0;               // Thanh ghi SSRC = 111 (bit 5 -> 7 của thanh ghi
                                   AD1CON1 là SSRC0 -> SSRC2)
                                   // Bộ đếm nội kết thúc và bắt đầu chuyển đổi
                                   //tự động chuyển đổi
   AD1CSSL = 0;                    // kênh analog là bỏ qua từ quét đầu vào (CSSL0)
   AD1CON2 = 0;                    // Thanh ghi ở bit 13 -> bit 15 của thanh ghi AD1CON2
                                   // NVCFG0 = 0 -> AVSS
                                   // PVCFG = 00 ->  AVDD
   AD1CON3 = 0x1F02;               // Cấu hình clock cho bộ ADC
                                   //thanh ghi ADCS c?a AD1CON3  <7:0> = 2 Tcy
                                   //thanh ghi SAMC<4:0> = 31 Tad
   AD1CON1bits.ADON = 1;           // Bộ ADC đang hoạt động
   _AD1IP = 5;                     //Cấu hình độ ưu tiên
   _AD1IF = 0;                     //Xóa cờ ngắt
   _AD1IE = 1;                     //Bật interrupt
}

int readADC(int ch) {
   AD1CHS = ch;                    // 1. select analog input channel
   AD1CON1bits.SAMP = 1;           // 2. start sampling
   while (!AD1CON1bits.DONE);      // 3. wait for the conversion to complete
   return ADC1BUF0;                // 4. read the conversion result
}

void _ISR _ADC1Interrupt(void) {
   _AD1IF = 0;                     //Xóa cờ ngắt
```

```
    code = ADC1BUF0;
    AD1CON1bits.SAMP = 1;
}

int main() {
    LCDinit();
    ADC_Init();
    AD1CHS = 4;
    AD1CON1bits.SAMP = 1;
    float temperature = 0.0;
    int temp;
    PMADDR = LCDCMD ;
    PMDATA = 0b00000001;
    TMR1 = 0; while ( TMR1 > 26);
    while (1)
    {
        PMADDR = LCDCMD ;
        PMDATA = 0b00000010;
        TMR1 = 0; while ( TMR1 < 26);
                    //Công thức chuyển đổi: Vout = (Code ×(VR+−VR−)) /1024
                    //Ta có công thức: Vout = (10mV/oC)* (Temparature oC) + 500mV
        temperature = (5 * (float)(code) * 1000 / 1024 - 500) / 10;
        temp = (int)(temperature * 10);
        putsLCD ("Nhiet do: ");
        putLCD(temp / 100 + 48);
        putLCD((temp % 100) / 10 + 48);
        putLCD ('.');
        putLCD((temp % 100) % 10 + 48);
```

```
        putLCD(0xdf);
        putLCD('C');
        check_lcd = 0;
    }
    return 0;
}
```

2. **Phần 3:** Kết quả mô phỏng bằng Proteus:



3. **Phần 4:** Giải thích kết quả mô phỏng bằng Proteus:

- ✓ Để hiển thị nhiệt độ từ cảm biến nhiệt độ, ta sử dụng bộ ngắt ADC như đã làm ở bài tập ngắt Inerrupt
- ✓ LCD cũng khá giống với UART ở khía cạnh lập trình những chức năng như hiển thị kí tự, đọc, nhận,….
- ✓ Khi điều chỉnh nhiệt độ, vi điều khiển cũng tính toán và đưa ra giá trị giống như bài UART. Tùy vào mức điều chỉnh giá trị từ cảm biến (nút mũi tên ở cảm biến) sẽ cho ra giá trị của cảm biến TC1047A.