Lab04 - Interrupt

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1 Introduction

This lab provides hand-on activity and insight of interrupt usages.

2 Program Description

2.1 Program 1 - Measure Microcontrol Temperature every 100ms Using Timer 1

To use timer 1 to generate an interrupt every 100ms, waveform generation mode 4 is used to configured CTC Timer/Counter mode of operation with top value equal to OCR1A value. we can compute the appropriate OCR1A top value to generate 100ms by following calculation

$$\frac{OCR1A}{CLK} = 100ms$$

OCR1A = CLK * 100ms = 16Mhz/64 * 100ms

OCR1A = 25000

We enable the ADC peripheral to use ADC channel 8 to measure the micro controller temperature. ADC interupt enable bit is set to enable interrupt.

To be critically safe while printing out temperature result, interrupt is disable while printing. bellow is critical block of code

```
*pSREG = (*pSREG) & (~0x80);// clear bit 7
float adcVal = (float)runningAdcVal/(float)conversions;
float voltage = calcTemperature(adcVal);
Serial.print("ADC Value "); Serial.println(adcVal);
Serial.print("Temperature Value "); Serial.println(voltage);
conversions = 0; runningAdcVal = 0;
*pSREG = (*pSREG) | (0x80); // set bit 7
```

There are 2 way to calibrate this temperature sensors.

- set the offset value T_{OS} for measured ADC value and a gain k. From result in Figure 1 and 2 we can determine the offset and gain value as follow

  Datasheet : 320mv ~30 C
  Measured : 393mV ~30 C

  Datasheet : 325mv(estimated) ~36 C
  Measured : 400mV ~36 C

  Notice each celcius increase corresponding to 1mV increase. Since gain is a contants of one we can conclude the offset if $393mV - 320mV = 73$

- We can also determine the temperature by plotting voltage as a function of temperature similar to figure 3. Note that voltage and temperature value collected in Table 2 are estimated values e.g 16 *Celsius data point is collected by placing the micro controller in the freezer for a few minutes. After plotting, we can estimate the best fit line equation to find temperature based on voltage. As we can see in Figure 3, the 2 line is closely align with in +- of 10 Celcius. Before calibrate the error range is +- 70 Celcius (Difference between Voltage measured at 25 *Celcius and voltage in datasheet recorded at25* Celcius = 393 - 320 = 73 =¿ ~70)

Video demo https://youtu.be/inFhzNNPQE0

Voltage(mV)	Temperature (*C)
242	-45
314	25
380	85

Figure 1: Datasheet microcontroller voltage to temperature measurement

Voltage(mV)	Temperature (*C)
360	16
393.06	30
400.1	36

Figure 2: Measured voltage to temperature measurement

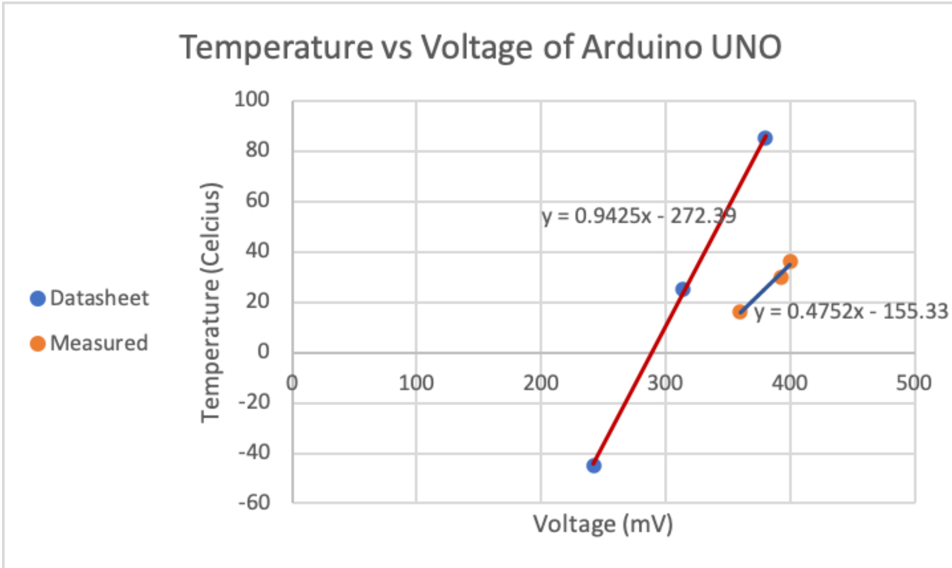

Figure 3: Voltage vs Temperature measurement using best fit line.

3 Summary

This lab introduced Timer/Counter 1 peripheral

4 Appendix

4.1 Main program

```
#include <stdint.h>;
#include "expriments.h";

int main(void) {
  init();
  Serial.begin(57600);
  experiment01();
  return 0;
}
```

4.2 Program 01

```

#include <Arduino.h>
#include "avr/interrupt.h"

/************************************************/
/***********EXPERIMENT 01**********************/
/************************************************/


volatile uint8_t
                *ioDDRB,

                *pADMUX,
                *pADCSRA,
                *pADCL,
```

```
                  *pADCH,

                  *pSREG, // AVR status register, to control global interrupt

                  *pTCCR1A, // timer counter control register A
                  *pTCCR1B, // timer counter control register B . A and B help group information and not nessary similar to I/O A and B.
                  *pOCR1AH, // output control register A high
                  *pOCR1AL, // output control register A low
                  *pTIMSK1 // enable timer/coutner 1 Interrupt Mask register(15.11.8);
;

uint16_t runningAdcVal = 0;
uint8_t conversions = 0;

void startADCConversion() {
  // start ADC conversion
  *pADCSRA = (*pADCSRA) | 0x40;
}


float calcTemperature(float adcVal) {
  float voltage = adcVal * 1.1 * 1000. / 1024.0 ; // 1.1 = Vref, *1000 to convert to mV
  return voltage * 0.4752 - 155.33; // 2 and 25.0 is offset value computed by looking at the best fit line between measured and datasheet value
}

// function get called every 100ms
ISR(TIMER1_COMPA_vect, ISR_BLOCK) {
  startADCConversion();
}

ISR(ADC_vect, ISR_BLOCK) {
  uint16_t result;
  result = *pADCL;// lower 8bits of ADC conversion (0x00??)
  result = result | (((uint16_t)*pADCH) << 8);
  result = result & 0x3FFF;// clear bits 15-10 (ADC res is only 10 bits)
  runningAdcVal += result;
  conversions += 1;
}

void myHardDelay(uint32_t ms) {
  volatile int16_t count;

  while (ms) {
    for (count = 0; count < 835; count++);
    ms -= 1;
  }
}


void experiment01() {

  ioDDRB  = (uint8_t *) 0x24;

  // ADC
  pADMUX = (uint8_t *) 0x7C;
  pADCSRA = (uint8_t *) 0x7A;
  pADCL = (uint8_t *) 0x78;
  pADCH = (uint8_t *) 0x79;

  // GLOBAL interupt
  pSREG = (uint8_t *) 0x5F;

  // Timer 1
  pTCCR1A = (uint8_t *) 0x80;
  pTCCR1B = (uint8_t *) 0x81;
  pOCR1AH = (uint8_t *) 0x89;
```

```
    pOCR1AL = (uint8_t *) 0x88;
    pTIMSK1 = (uint8_t *) 0x6F;

     //make PB1 an output
     *ioDDRB = 0x02; // DDRB[7:0] = 0000 0010

    // init ADC peripheral to measure temperature on ADC channel 8
    *pADMUX = 0xC8; // ADIE[3]: ADC enable  interupt, b[7:4] = 1100 (select ADC channel 8)
    *pADCSRA = 0x8F; // b[7] = ADEN: 1, b[3] = ADIE = 1, b[2:0] = ADPS: 111 (clock  = CLK/128 = 125kHz)

    // configure timmer with period of 100ms
    // configure timer counter 1 for CTC mode and generate a square wave @ X hz

    // Toggle OC1A on compare match; mode 4 = pTCCR1A[1:0] & pTCCR1B[4:3] = WGM[3:0] = 0b0100 = mode 4
    *pTCCR1A = (uint8_t *) 0x40;//0x01000000;
    *pTCCR1B = (uint8_t *) 0b00001011;// Mode 4 (CTC TOP = OCR1A, clock[2:0] = 0b110 =>  = 16Mhz/64=250Khz)

    // configure period = 1 / CLK / OCR1
    uint16_t OCR1A_val = 25000;
    *pOCR1AH = (OCR1A_val >> 8) & 0x00FF;
    *pOCR1AL = (OCR1A_val) & 0x00FF;

    // Enable Overflow interrupt
    *pTIMSK1 = 0b00000010; // bit 1-Enable Timer1 output compare A match interupt enable.

    while(1) {
      // no need to wait using while loop here and chcek for complere
      *pSREG = (*pSREG) & (~0x80);// clear bit 7
      float adcVal = (float)runningAdcVal/(float)conversions;
      float voltage = calcTemperature(adcVal);
      Serial.print("ADC Value ");Serial.println(adcVal);
      Serial.print("Temperature Value ");Serial.println(voltage);
      conversions = 0; runningAdcVal = 0;
      *pSREG = (*pSREG) | (0x80); // set bit 7
      delay(1000);
    }
}
```