**Microcomputers**

**EEL 4746**

**Assignment 5 – Daniel Taylor**

Submit the solution for the following problems

**Problem 1**

Answer the following questions:

1. How do we start the conversion in the ATmega 328? How do we recognize the end of conversion?

**I am aware that we were taught that the main register for ADC is ADCSA, but my AVR code in Microchip Studio doesn’t detect ADCSA, only ADCSRA. So, from now on, “ADCSRA” can be interchanged with “ADCSA” and means the register that you can use for ADC conversion with an ATMEGA328P.**

To start the ADC conversion with an ATMEGA 328, we first must assume the pin for the selected ADC channel is an input pin, the ADC module is turned on (by writing 1 to the ADEN bit), the conversion speed selected, and the voltage reference and ADC input channels are connected. Then, the conversion is started by **writing a one to the ADSC bit of the ADCSRA register**. **The end of this conversion is recognized by monitoring the ADIF bit in the ADCSRA register**. When the ADIF bit is HIGH, the conversion is complete and the ADCL/ADCH registers can be read, and 1 must be written into ADIF to clear the flag and reset it.

1. Which bits of which register of the ATMega328P are used to select conversion speed?

The **3 ADPS (ADPS0, ADPS1, ADPS2) bits from the ADCSRA register** are used to select conversion speed. This determines the division factor between the crystal frequency and the input clock to the ADC.

1. Write a program that gets analog data from channel 2 of ADC unit and display the results on Port B and Port D. This is done forever. (use the Polling method)

Assume, right justified and a pre scaler of 64.

Channel 2 of ADC

#include <avr/io.h>

#define *F\_CPU* 16000000UL

#include <util/delay.h>

int main(void) {

//data direction, PORB and PORTD is outputs.

DDRD = 0xFF;

DDRB = 0xFF;

//Write 1 to the ADEN ADCSA register to enable ADC conversion, have prescaler of 64. ADCSA = 1000 0110;

ADCSRA = 0x86;

// ADMUX = 0100 0010, meaning we’re using AVV for VREF and using Channel 2 with right-justified.

ADMUX = 0x42;

//after initialization, while loop to start conversion, poll, do instruction when conversion finished, and repeat

while(1) {

//write 1 to ADSC in ADSCRA to start conversion

ADCSRA |= ADSC;

//use polling to monitor ADIF

while(ADCSA & (1<<ADIF) == 0) {}

//now exited poll loop, ADIF is 1, meaning conversion is complete.

//reset flag

ADCSRA |= (1<<ADIF);

//display ACD low and ACD high on port D and B respectively

PORTD = ADCL;

PORTB = ADCH;

//100 ms after conversion complete so no corruption

*\_delay\_ms*(100);

}

return 0;

}

1. Repeat question 6 by using the interrupt method.

#include <avr/io.h>

#define *F\_CPU* 16000000UL

#include <util/delay.h>

#include <avr/interrupt.h>

int main(void) {

//data direction, PORB and PORTD is outputs.

DDRD = 0xFF;

DDRB = 0xFF;

//Write 1 to the ADEN ADCSA register to enable ADC conversion, have prescaler of 64. ADCSA = 1000 1110;

ADCSRA = 0x8E;

// ADMUX = 0100 0010, meaning we’re using AVV for VREF and using Channel 2 with right-justified.

ADMUX = 0x42;

//enable global interrupt

sei();

//after initialization, while loop to start conversion, poll, do instruction when conversion finished, and repeat

while(1) {

//write 1 to ADSC in ADSCRA to start conversion

ADCSRA |= ADSC;

}

return 0;

}

ISR (ADC\_vect) {

//don't need to reset flag since using interrupts

//ADCSA |= (1<<ADIF);

//display ADC low and ADC high on port D and B respectively

PORTD = ADCL;

PORTB = ADCH;

//100 ms after conversion complete so no corruption

*\_delay\_ms*(100);

}

**Problem 2**

Using Timer 0 and inverted Fast PWM mode, write a program that generates a wave with frequency of 46.875 KHZ and duty cycle of 70%. Assume XTAL = 12MHz

Using FastPWM mode

Find Prescaler value N

Foco = Fclk/(256N)  
N = 46875 \* (256/12000000) = 1. No prescalar needed..

Find OCR0A value.

DC = ((OCR0A + 1)/256) \* 100, OCR0A = 70 \* 256/100 -1 = 178.2. OCR0A is 178.

#include <avr/io.h>

int main(void) {

//Data direction for PORTD PIN 6 since we’re using OCR0A.

DDRD = 1<<6;

//Turn on Fast PWM mode for inverting

//TCCR0A = 1100 0011

TCCR0A = 0xC3;

//Set no prescalar

TCCR0B = 0x01;

//Set OCR0A value

OCR0A = 178;

return 0;

}

//Don’t need while loop since it does it all for you

**Problem 3**

Using Timer2 and non-inverted phase correct PWM mode, write a program that generates a wave with a frequency of 61.3 Hz and duty cycle of 19%. Assume XTAL = 8MHz

1. Find prescaler needed

61.3 = 8000000/(N\*510), so N = 61.3 \* (510 / 8000000) = 0.0039, so no prescaler.

1. Find OCR2A Value

DC = (OCRA2/255) \* 100, so

1. Duty Cycle = 19, so 19 = OCRA2/255 \* 100

OCRA2 = 48

#include <avr/io.h>

int main(void) {

//Data direction for PORTB pin 3 since we’re using OCR2A.

DDRB = 1<<3;

//Turn on phase correct PWM mode for non-inverting

//TCCR0A = 1000 0001

TCCR2A = 0x81;

//Set no prescalar

TCCR2B = 0x01;

//Set OCR0A value

OCR2A = 48;

return 0;

}