Patrick Austin

CPE 301 - 1104, Fall 2016

Homework 5

10/10/2016

//Patrick Austin

//CPE 301 HW 5 Problem 1

//Revision Number 1

//Revision date: 10/10/2016

//NOTE: adapted from class example code 10/10/16

//hardware pointer declarations

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCCR1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCCR1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCCR1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

void myDelay( unsigned long mSeconds );

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x80; //enable output on LED pin, pin 7

\*myTCCR1A = 0; //zero out timer controls to enable normal mode, timer off

\*myTCCR1B = 0;

\*myTCCR1C = 0;

\*myTIMSK1 = 0; //zero out interrupt register to disable interrupts

}

void loop()

{

//toggle LED to demonstrate working delay function

\*myPortB = \*myPortB | 0x80; // enable output pin, ie LED on

myDelay(500); //wait a while

\*myPortB = \*myPortB & 0x7F; //disable output pin, ie LED off

myDelay(500); //wait a while

}

void myDelay( unsigned long mSeconds )

{

\*myTCCR1B = \*myTCCR1B & 0xF8; //set timer to off

//calculate preload value for the timer. need preload value such that the time needed for the

//timer to go from the preload value to raising the overflow flag will take 'mSeconds' ms.

//this implementation uses the 1024 prescaler.

\*myTCNT1 = (unsigned int) (65536 - (long) (15.625 \* mSeconds)); //get preload value

\*myTCCR1B = \*myTCCR1B | 0x05; //enable timer with 1024 prescaler

while ( (\*myTIFR1 & 0x01) == 0 ) //until the overflow flag is raised, do nothing

{}

\*myTCCR1B = 0; //delay complete, turn off the timer

\*myTIFR1 = \*myTIFR1 | 0x01; //reset the oveflow flag by writing a 1, finished

}

//Patrick Austin

//CPE 301 HW 5 Problem 2

//Revision Number 1

//Revision date: 10/10/2016

//hardware pointer declarations

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCCR1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCCR1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCCR1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

void myDelayMicroSeconds( unsigned long microSeconds );

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x40; //enable output on specified pin, pin 6

\*myTCCR1A = 0; //zero out timer controls to enable normal mode, timer off

\*myTCCR1B = 0;

\*myTCCR1C = 0;

\*myTIMSK1 = 0; //zero out interrupt register to disable interrupts

}

void loop()

{

// create a square wave at 440 hZ on port B6. That is, go high for 220 hZ and low for 220 hZ.

// 1/440 = 2.272727 ms. Half that is 1.13636363 ms. So we want a 1.13636363 ms delay.

// 1.1363 ms = approx 1136 microseconds. By rule of thumb, use the highest clock speed that will allow

// the desired interval to be timed. So use a myDelay that runs for 1136 microseconds using prescaler 1.

\*myPortB = \*myPortB | 0x40; //enable output pin

myDelayMicroSeconds(1136); //wait 1/2 period

\*myPortB = \*myPortB & 0xBF; //disable output pin

myDelayMicroSeconds(1136); //wait 1/2 period

}

void myDelayMicroSeconds( unsigned long microSeconds ) //adapted from problem 1 and in-class code

{

\*myTCCR1B = \*myTCCR1B & 0xF8; //set timer to off

//calculate preload value for the timer. need preload value such that the time needed for the

//timer to go from the preload value to raising the overflow flag will take 'microSeconds' microseconds.

//this implementation uses the 1 prescaler.

\*myTCNT1 = (unsigned int) (65536 - (long) (microSeconds / .0625)); //get preload value

\*myTCCR1B = \*myTCCR1B | 0x01; //enable timer with 1 prescaler

while ( (\*myTIFR1 & 0x01) == 0 ) //until the overflow flag is raised, do nothing

{}

\*myTCCR1B = 0; //delay complete, turn off the timer

\*myTIFR1 = \*myTIFR1 | 0x01; //reset the oveflow flag by writing a 1, finished

}

//Patrick Austin

//CPE 301 HW 5 Problem 3

//Revision Number 1

//Revision date: 10/10/2016

//hardware pointer declarations

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCCR1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCCR1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCCR1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

void myDelayMicroSeconds( unsigned long microSeconds );

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x40; //enable output on specified pin, pin 6

\*myTCCR1A = 0; //zero out timer controls to enable normal mode, timer off

\*myTCCR1B = 0;

\*myTCCR1C = 0;

\*myTIMSK1 = 0; //zero out interrupt register to disable interrupts

}

void loop()

{

// create a square wave at 12 khZ on port B6. That is, go high for 6000 hZ and low for 6000 hZ.

// 1/12000 = .08333333333 ms. Half that is .041666666667 ms. So we want a .04166666667 ms delay.

// .04166667 ms = approx 42 microseconds. By rule of thumb, use the highest clock speed that will allow

// the desired interval to be timed. So use a myDelay that runs for 42 microseconds using prescaler 1.

\*myPortB = \*myPortB | 0x40; //enable output pin

myDelayMicroSeconds(42); //wait 1/2 period

\*myPortB = \*myPortB & 0xBF; //disable output pin

myDelayMicroSeconds(42); //wait 1/2 period

}

void myDelayMicroSeconds( unsigned long microSeconds ) //adapted from problem 1 and in-class code

{

\*myTCCR1B = \*myTCCR1B & 0xF8; //set timer to off

//calculate preload value for the timer. need preload value such that the time needed for the

//timer to go from the preload value to raising the overflow flag will take 'microSeconds' microseconds.

//this implementation uses the 1 prescaler.

\*myTCNT1 = (unsigned int) (65536 - (long) (microSeconds / .0625)); //get preload value

\*myTCCR1B = \*myTCCR1B | 0x01; //enable timer with 1 prescaler

while ( (\*myTIFR1 & 0x01) == 0 ) //until the overflow flag is raised, do nothing

{}

\*myTCCR1B = 0; //delay complete, turn off the timer

\*myTIFR1 = \*myTIFR1 | 0x01; //reset the oveflow flag by writing a 1, finished

}

//Patrick Austin

//CPE 301 HW 5 Problem 4

//Revision Number 1

//Revision date: 10/10/2016

//hardware pointer declarations

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCCR1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCCR1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCCR1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

void myDelayMicroSeconds( unsigned long microSeconds );

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x40; //enable output on specified pin, pin 6

\*myTCCR1A = 0; //zero out timer controls to enable normal mode, timer off

\*myTCCR1B = 0;

\*myTCCR1C = 0;

\*myTIMSK1 = 0; //zero out interrupt register to disable interrupts

}

void loop()

{

// create a rectangular wave at 500 hZ on port B6. 30% duty cycle, so high 30% per wave and low 70%.

// 1/500 = 2ms. 30% of 2ms = .6ms and 70% of 2ms = 1.4ms. So high .6 ms (600 usec), low 1.4ms.

// 1.4ms = 1400 microseconds. By rule of thumb, use the highest clock speed that will allow

// the desired interval to be timed. So use myDelays for each interval using prescaler 1.

\*myPortB = \*myPortB | 0x40; //enable output pin

myDelayMicroSeconds(600); //wait 30% of period

\*myPortB = \*myPortB & 0xBF; //disable output pin

myDelayMicroSeconds(1400); //wait 70% of period

}

void myDelayMicroSeconds( unsigned long microSeconds ) //adapted from problem 1 and in-class code

{

\*myTCCR1B = \*myTCCR1B & 0xF8; //set timer to off

//calculate preload value for the timer. need preload value such that the time needed for the

//timer to go from the preload value to raising the overflow flag will take 'microSeconds' microseconds.

//this implementation uses the 1 prescaler.

\*myTCNT1 = (unsigned int) (65536 - (long) (microSeconds / .0625)); //get preload value

\*myTCCR1B = \*myTCCR1B | 0x01; //enable timer with 1 prescaler

while ( (\*myTIFR1 & 0x01) == 0 ) //until the overflow flag is raised, do nothing

{}

\*myTCCR1B = 0; //delay complete, turn off the timer

\*myTIFR1 = \*myTIFR1 | 0x01; //reset the oveflow flag by writing a 1, finished

}

//Patrick Austin

//CPE 301 HW 5 Problem 5

//Revision Number 1

//Revision date: 10/10/2016

//hardware pointer declarations

volatile unsigned char\* myPortDDRB = (unsigned char\*) 0x24;

volatile unsigned char\* myPortB = (unsigned char\*) 0x25;

volatile unsigned char\* myTIFR1 = (unsigned char\*) 0x36;

volatile unsigned char\* myTCCR1A = (unsigned char\*) 0x80;

volatile unsigned char\* myTCCR1B = (unsigned char\*) 0x81;

volatile unsigned char\* myTCCR1C = (unsigned char\*) 0x82;

volatile unsigned int\* myTCNT1 = (unsigned int\*) 0x84;

volatile unsigned char\* myTIMSK1 = (unsigned char\*) 0x6F;

//lookup table for shutterSelect function. Values are corresponding periods in ms

//for the fractions of a second specified in the prompt.

//EG, values[1] = 500 (ms) for the shutter to open 1/2 second.

static const unsigned int values[] = { 1000, 500, 250, 125, 67, 33, 17, 8, 4, 2, 1 };

//prototypes

void shutterSelect ( unsigned int shutterSpeed );

void myDelayMilliSeconds( unsigned long mSeconds ); //reused from problem 1

void setup()

{

\*myPortDDRB = \*myPortDDRB | 0x80; //enable output on specified pin, pin 7

\*myTCCR1A = 0; //zero out timer controls to enable normal mode, timer off

\*myTCCR1B = 0;

\*myTCCR1C = 0;

\*myTIMSK1 = 0; //zero out interrupt register to disable interrupts

}

void loop()

{

// User will pass via some input a value 1-10 to select shutter speed in a final implementation.

// In this program, we merely cycle through the different options for test purposes.

for ( unsigned int i = 0; i != 11; i++ ) //for all the different shutter speed options...

{

shutterSelect(i); //test a shutter speed

myDelayMilliSeconds(1000); //wait a little while

}

}

void shutterSelect ( unsigned int shutterSpeed )

{

//Based on the selected speed, open the shutter for timeToWait ms and then close it.

//Simply grab the time to wait from the lookup table, open the shutter, wait specified

//length of time, close shutter.

unsigned int timeToWait = values[shutterSpeed];

\*myPortB = \*myPortB | 0x80; //open shutter

myDelayMilliSeconds(timeToWait); //delay specified number of ms

\*myPortB = \*myPortB & 0x7F; //close shutter

}

void myDelayMilliSeconds( unsigned long mSeconds )

{

\*myTCCR1B = \*myTCCR1B & 0xF8; //set timer to off

//calculate preload value for the timer. need preload value such that the time needed for the

//timer to go from the preload value to raising the overflow flag will take 'mSeconds' ms.

//this implementation uses the 1024 prescaler.

\*myTCNT1 = (unsigned int) (65536 - (long) (15.625 \* mSeconds)); //get preload value

\*myTCCR1B = \*myTCCR1B | 0x05; //enable timer with 1024 prescaler

while ( (\*myTIFR1 & 0x01) == 0 ) //until the overflow flag is raised, do nothing

{}

\*myTCCR1B = 0; //delay complete, turn off the timer

\*myTIFR1 = \*myTIFR1 | 0x01; //reset the oveflow flag by writing a 1, finished

}







