Deliverables:

1)

2)

|  |
| --- |
| Time to Sample In uS: 129 |
| Time to Print in uS: 4841 |

3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position | Analog input | ADC sample | Correct Fixed-point | Measured Fixed-point Output |
| 0.10 cm | 0.2 | 390 | 10 | 20 |
| 0.40 cm | 0.4 | 800 | 40 | 40 |
| 0.80 cm | .79 | 1560 | 80 | 790 |
| 1.20 cm | 1.28 | 2500 | 120 | 127 |
| 1.40 cm | 1.51 | 2940 | 140 | 150 |

4)

// Lab8.cpp

// Runs on LM4F120 or TM4C123

// Student names: change this to your names or look very silly

// Last modification date: change this to the last modification date or look very silly

// Last Modified: 1/17/2020

// Specifications:

// Measure distance using slide pot, sample at 60 Hz

// maximum distance can be any value from 1.5 to 2cm

// minimum distance is 0 cm

// Calculate distance in fixed point, 0.01cm

// Analog Input connected to PD2=ADC0 (OG 5)

// displays distance on Sitronox ST7735

// PF3, PF2, PF1 are heartbeats (use them in creative ways)

// must include at least one class used in an appropriate way

#define Main //Replace with Main, MainOne, MainTwo, or MainThree depending on current test

#define CONVERSION\_NUMERATOR 210

#define CONVERSION\_DENOMINATOR 4096

#define CONVERSION\_OFFSET 10

#include <stdint.h>

#include "../inc/tm4c123gh6pm.h"

#include "PLL.h"

#include "ST7735.h"

#include "TExaS.h"

#include "PLL.h"

#include "SlidePot.h"

#include "print.h"

SlidePot Sensor(CONVERSION\_NUMERATOR,CONVERSION\_DENOMINATOR, CONVERSION\_OFFSET);

extern "C" void DisableInterrupts(void);

extern "C" void EnableInterrupts(void);

extern "C" void SysTick\_Handler(void);

#define PF1 (\*((volatile uint32\_t \*)0x40025008))

#define PF2 (\*((volatile uint32\_t \*)0x40025010))

#define PF3 (\*((volatile uint32\_t \*)0x40025020))

#define PF4 (\*((volatile uint32\_t \*)0x40025040))

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*SysTick\_Init\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Initialize Systick periodic interrupts

// Input: interrupt period

// Units of period are 12.5ns

// Maximum is 2^24-1

// Minimum is determined by length of ISR

// Output: none

void SysTick\_Init(unsigned long period){

//\*\*\* students write this \*\*\*\*\*\*

NVIC\_ST\_CTRL\_R = 0;

NVIC\_ST\_RELOAD\_R = period;

NVIC\_ST\_CURRENT\_R = 0;

NVIC\_ST\_CTRL\_R = 7;

}

// Initialize Port F so PF1, PF2 and PF3 are heartbeats

void PortF\_Init(void){

//\*\*\* students write this \*\*\*\*\*\*

SYSCTL\_RCGCGPIO\_R |= 0x20;

\_\_asm\_\_ {

NOP

NOP

}

GPIO\_PORTF\_DIR\_R |= 0x04;

GPIO\_PORTF\_DEN\_R |= 0x04;

}

void toggleHearBeat(void) {

GPIO\_PORTF\_DATA\_R ^= 0x04;

}

uint32\_t Data; // 12-bit ADC

uint32\_t Position; // 32-bit fixed-point 0.01 cm

#ifdef MainOne

int main(void){ // single step this program and look at Data

DisableInterrupts();

TExaS\_Init(); // start scope set system clock to 80 MHz

ADC\_Init(); // turn on ADC, PD2, set channel to 5

EnableInterrupts();

while(1){

Data = ADC\_In(); // sample 12-bit channel 5, PD2

}

}

#endif

#ifdef MainTwo

uint32\_t time0,time1,time2,time3;

uint32\_t ADCtime,OutDectime; // in usec

int main(void){

TExaS\_Init(); // Bus clock is 80 MHz

NVIC\_ST\_RELOAD\_R = 0x00FFFFFF; // maximum reload value

NVIC\_ST\_CURRENT\_R = 0; // any write to current clears it

NVIC\_ST\_CTRL\_R = 5;

ADC\_Init(); // turn on ADC, set channel to 5

ADC0\_SAC\_R = 4; // 16-point averaging, move this line into your ADC\_Init()

ST7735\_InitR(INITR\_GREENTAB);

while(1){ // use SysTick

time0 = NVIC\_ST\_CURRENT\_R;

Data = ADC\_In(); // sample 12-bit channel 5

time1 = NVIC\_ST\_CURRENT\_R;

ST7735\_SetCursor(0,0);

time2 = NVIC\_ST\_CURRENT\_R;

LCD\_OutDec(Data);

ST7735\_OutString(" "); // spaces cover up characters from last output

time3 = NVIC\_ST\_CURRENT\_R;

ADCtime = ((time0-time1)&0x0FFFFFF)/80; // usec

OutDectime = ((time2-time3)&0x0FFFFFF)/80; // usec

}

}

#endif

#ifdef MainThree

int main(void){

DisableInterrupts();

TExaS\_Init(); // Bus clock is 80 MHz

ST7735\_InitR(INITR\_GREENTAB);

PortF\_Init();

ADC\_Init(); // turn on ADC, PD2, set channel to 5

EnableInterrupts();

while(1){

PF2 ^= 0x04; // Heartbeat

Data = ADC\_In(); // sample 12-bit channel 5, PD2

PF3 = 0x08; // Profile Convert

Position = Sensor.Convert(Data);

PF3 = 0; // end of Convert Profile

PF1 = 0x02; // Profile LCD

ST7735\_SetCursor(0,0);

LCD\_OutDec(Data); ST7735\_OutString(" ");

ST7735\_SetCursor(6,0);

LCD\_OutFix(Position);

PF1 = 0; // end of LCD Profile

}

}

#endif

#ifdef Main

// final main program to create distance meter

int main(void){

//\*\*\* students write this \*\*\*\*\*\*

DisableInterrupts();

TExaS\_Init(); // bus clock at 80 MHz

ST7735\_InitR(INITR\_GREENTAB);

ADC\_Init(); // turn on ADC, PD2, set channel to 5

SysTick\_Init(8000000);

PortF\_Init();

// more initializations

EnableInterrupts();

while(1){

Sensor.Sync(); // wait for semaphore

// can call Sensor.ADCsample, Sensor.Distance, Sensor.Convert as needed

uint32\_t distance = Sensor.Distance();

ST7735\_SetCursor(0,0);

LCD\_OutFix(distance);

ST7735\_OutString(" cm");

}

}

#endif

void SysTick\_Handler(void){ // every sample

//\*\*\* students write this \*\*\*\*\*\*

// should call ADC\_In() and Sensor.Save

toggleHearBeat();

Sensor.Save(ADC\_In());

}

// SlidePot.cpp

// Runs on LM4F120/TM4C123

// Provide functions that initialize ADC0 and use a slide pot to measure distance

// Created: 3/28/2018

// Student names: change this to your names or look very silly

// Last modification date: change this to the last modification date or look very silly

#define Simulator //change to Hardware or Simulator depending on environment

#include <stdint.h>

#include "SlidePot.h"

#include "../inc/tm4c123gh6pm.h"

// ADC initialization function

// Input: none

// Output: none

// measures from PD2, analog channel 5

void ADC\_Init(void){

//\*\*\* students write this \*\*\*\*\*\*

SYSCTL\_RCGCADC\_R |= 0x01; //enable ADC0

SYSCTL\_RCGCGPIO\_R |= 1<<3; //enable Port D

\_\_nop();

\_\_nop(); //wait for clock

GPIO\_PORTD\_DIR\_R |= 1<<2;

GPIO\_PORTD\_AFSEL\_R |= 1<<2;

GPIO\_PORTD\_DEN\_R &= ~(1<<2);

GPIO\_PORTD\_AMSEL\_R |= 1<<2;

ADC0\_PC\_R &= ~(0x0F);

ADC0\_PC\_R |= 0x01;

ADC0\_SSPRI\_R = 0x0123;

ADC0\_ACTSS\_R &= ~0x0008;

ADC0\_EMUX\_R &= ~0xF000;

ADC0\_SSMUX3\_R &= ~0x000F;

ADC0\_SSMUX3\_R += 5;

ADC0\_SSCTL3\_R = 0x0006;

ADC0\_IM\_R &= ~0x0008;

ADC0\_ACTSS\_R |= 0x0008;

#ifdef Hardware

ADC0\_SAC\_R = 0x06;

#endif

//copied from textbook

}

//------------ADCIn------------

// Busy-wait Analog to digital conversion

// Input: none

// Output: 12-bit result of ADC conversion

// measures from PD2, analog channel 5

uint32\_t ADC\_In(void){

//\*\*\* students write this \*\*\*\*\*\*

uint32\_t result;

ADC0\_PSSI\_R = 0x0008;

while((ADC0\_RIS\_R&0x08)==0){};

result = ADC0\_SSFIFO3\_R&0xFFF;

ADC0\_ISC\_R = 0x0008;

return result;

}

// constructor, invoked on creation of class

// m and b are linear calibration coeffients

SlidePot::SlidePot(uint32\_t mT, uint32\_t mD, uint32\_t b){

//\*\*\* students write this \*\*\*\*\*\*

// initialize all private variables

// make slope equal to m and offset equal to b

slopeNumerator = mT;

slopeDenominator = mD;

offset = b;

}

void SlidePot::Save(uint32\_t n){

//\*\*\* students write this \*\*\*\*\*\*

// 1) save ADC sample into private variable

// 2) calculate distance from ADC, save into private variable

// 3) set semaphore flag = 1

data = n;

distance = Convert(data);

flag = 1;

}

uint32\_t SlidePot::Convert(uint32\_t n){

//\*\*\* students write this \*\*\*\*\*\*

// use calibration data to convert ADC sample to distance

return (slopeNumerator\*n - offset )/slopeDenominator;

//TODO: Verify overflow? Idk why but the lab manual says to watch out for it here

}

void SlidePot::Sync(void){

// 1) wait for semaphore flag to be nonzero

// 2) set semaphore flag to 0

while(flag==0);

flag = 0;

}

uint32\_t SlidePot::ADCsample(void){ // return ADC sample value (0 to 4095)

//\*\*\* students write this \*\*\*\*\*\*

// return last calculated ADC sample

return data; // replace this with solution

}

uint32\_t SlidePot::Distance(void){ // return distance value (0 to 2000), 0.001cm

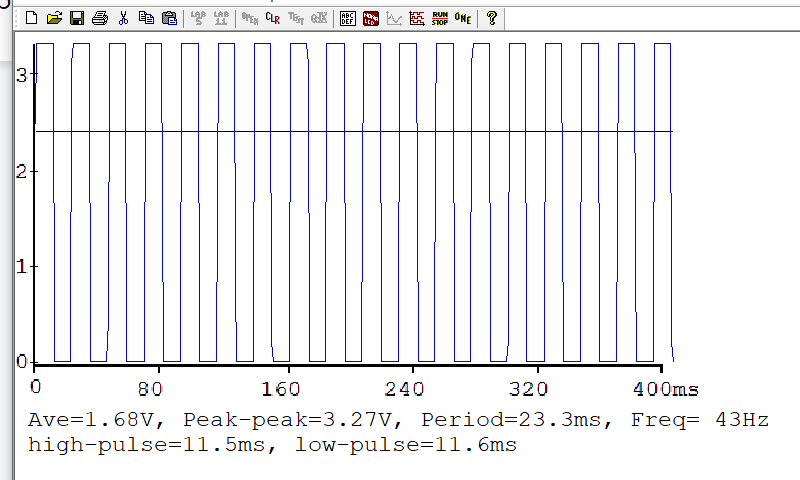
//\*\*\* students write this \*\*\*\*\*\*

// return last calculated distance in 0.001cm

return distance; // replace this with solution

}

5)



6)

Average accuracy  (with units in cm)  =

|  |  |  |
| --- | --- | --- |
| True position  xti | Measured Position  xmi | Error  xti - xmi |
| .1 | .2 |  |
| .4 | .4 |  |
| .8 | .79 |  |
| 1.2 | 1.51 |  |
|  |  |  |

*Table 8.2. Accuracy results of the position measurement system.*