LAB 8 Deliverables

Shreyas Kudari (sak3392)

Rakesh Johny (rj

Calibration Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position | Analog input | ADC sample | Correct Fixed-Point | Measured Fixed-Point |
| 0.40 cm | 0.18 V | 232 | 400 | 412 |
| 0.60 cm | 0.62 V | 780 | 600 | 617 |
| 0.80 cm | 1.03 V | 1269 | 800 | 802 |
| 1.20 cm | 1.93 V | 2367 | 1200 | 1208 |
| 1.40 cm | 2.31 V | 2843 | 1400 | 1412 |
| 1.60 cm | 2.73 V | 3388 | 1600 | 1595 |
| 1.80 cm | 3.16 V | 3935 | 1800 | 1802 |

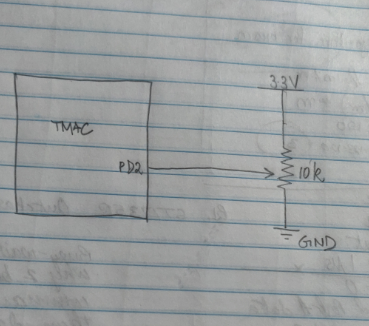
Accuracy Table

|  |  |  |
| --- | --- | --- |
| True Position  Xti | Measured Position  Xmi | Error  Xti-Xmi |
| 0.50cm | 0.498cm | 0.002 |
| 0.70cm | 0.704cm | -0.004 |
| 1.30cm | 1.305cm | -0.005 |
| 1.50cm | 1.506cm | -0.006 |
| 1.70cm | 1.705cm | -0.005 |

Average Error: 0.004

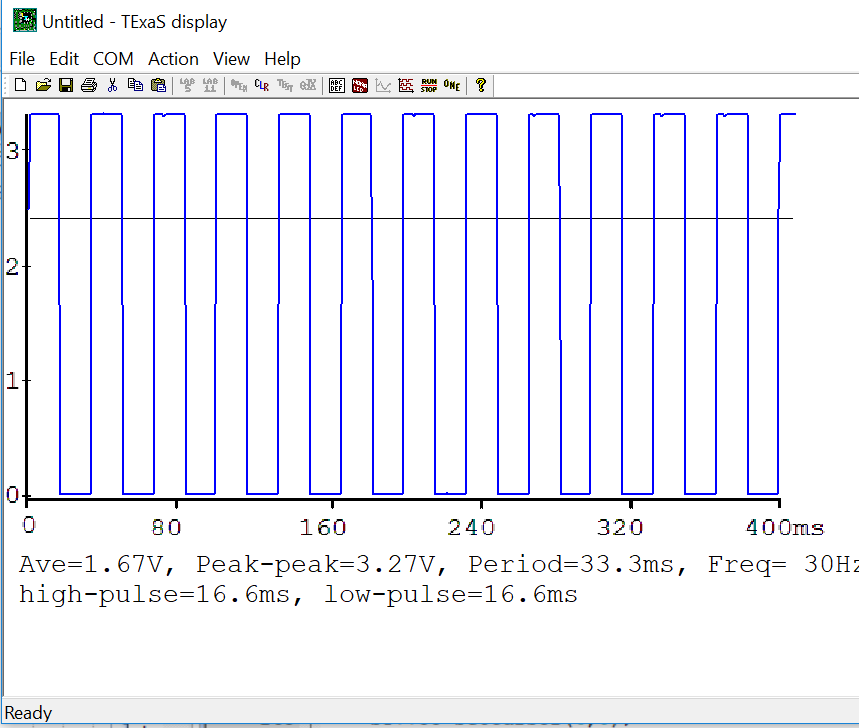
Accuracy: 0.4%

Circuit Diagram



Texas Display

60Hz Verification



**Each oscillation has 2 samples. Therefore, Real frequency = 2x30Hz = 60Hz**

Time Profiles for ADC and LCD

Code:

Main.cpp

// Lab8.cpp

// Runs on LM4F120 or TM4C123

// Student names: Shreyas Kudari and Rakesh Johny

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

// Last Modified: 3/28/2018

// 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, 0.001cm

// Analog Input connected to PD2=ADC5

// 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

#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(2000,0);

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){

NVIC\_ST\_CTRL\_R=0;

NVIC\_ST\_CURRENT\_R=0;

NVIC\_ST\_RELOAD\_R=period;

NVIC\_SYS\_PRI3\_R=(NVIC\_SYS\_PRI3\_R&0x00FFFFFF)|0x40000000;

NVIC\_ST\_CTRL\_R=0x07;

}

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

void PortF\_Init(void){

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

volatile uint32\_t delay;

SYSCTL\_RCGCGPIO\_R|=0x20;

delay++;

delay++;

GPIO\_PORTF\_LOCK\_R=GPIO\_LOCK\_KEY;

GPIO\_PORTF\_CR\_R |=0x0E; //enable commit for PF1, PF2 and PF3

GPIO\_PORTF\_AMSEL\_R &= ~0x0E; //disable analog functionality on PF1, PF2 and PF

//GPIO\_PORTF\_PCTL\_R = (GPIO\_PORTF\_PCTL\_R&0xFFFF000F)+0x00000000;//configure PF0 and PF4 as GPIO

GPIO\_PORTF\_AFSEL\_R &= ~0x0E; //disable alt funct on PF0 and PF4

GPIO\_PORTF\_DIR\_R|=0x0E;

GPIO\_PORTF\_DEN\_R|=0x0E;

//GPIO\_PORTF\_PUR\_R=0x0E;

}

uint32\_t Data; // 12-bit ADC

uint32\_t Position; // 32-bit fixed-point 0.001 cm

int main1(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

}

}

int main2(void){

DisableInterrupts();

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

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

ST7735\_InitR(INITR\_REDTAB);

PortF\_Init();

EnableInterrupts();

while(1){ // use scope to measure execution time for ADC\_In and LCD\_OutDec

PF2 = 0x04; // Profile ADC

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

PF2 = 0x00; // end of ADC Profile

ST7735\_SetCursor(0,0);

PF1 = 0x02; // Profile LCD

LCD\_OutDec(Data);

ST7735\_OutString(" "); // these spaces are used to coverup characters from last output

PF1 = 0; // end of LCD Profile

}

}

int main3(void){

DisableInterrupts();

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

ST7735\_InitR(INITR\_REDTAB);

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

}

}

// final main program to create distance meter

int main(void){

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

DisableInterrupts();

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

ST7735\_InitR(INITR\_REDTAB);

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

PortF\_Init();

// more initializations

EnableInterrupts();

SysTick\_Init(1333333); //80M/60

while(1){

Sensor.Sync(); // wait for semaphore

ST7735\_SetCursor(0,0);

Position=Sensor.Distance();

LCD\_OutFix(Position);

ST7735\_SetCursor(6,0);

ST7735\_OutChar(67);

ST7735\_OutChar(77);

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

}

}

void SysTick\_Handler(void){ // every 16.67 ms

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

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

PF1^=0x02;

PF1^=0x02;

uint32\_t data=ADC\_In();

Sensor.Save(data);

PF1^=0x02;

}

Slidepot.cpp

// 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

#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){

volatile int delay=0;

SYSCTL\_RCGCGPIO\_R |= 0x08; //activate port D clock

delay++;

GPIO\_PORTD\_DIR\_R|= ~0x04;

GPIO\_PORTD\_AFSEL\_R |= 0x04;

GPIO\_PORTD\_DEN\_R &= ~0x04;

GPIO\_PORTD\_AMSEL\_R |= 0x04;

SYSCTL\_RCGCADC\_R|=0x01; //enable ADC clock

delay++;

delay++;

delay++;

delay++; //extra time to stabilize

ADC0\_PC\_R=0x01; //sampling speed 125ksamples/second

ADC0\_SSPRI\_R=0x0123; //set priorities for the sequencers. Sequencer 3 is the highest priority

ADC0\_ACTSS\_R &= ~0x0008; //disable sample sequencer 3

ADC0\_EMUX\_R &= ~0xF000; //seq 3 is software trigger

ADC0\_SSMUX3\_R = (ADC0\_SSMUX3\_R&0xFFFFFFF0)+5; //Ain5

ADC0\_SSCTL3\_R = 0x0006;

ADC0\_IM\_R &= ~0x0008; //disable SS3 interrupts

ADC0\_ACTSS\_R |= 0x0008; //enable sample sequencer 3

}

//------------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){

uint32\_t data;

ADC0\_PSSI\_R=0x0008; //start ADC

while((ADC0\_RIS\_R&0x08)==0){} //busy wait

data=ADC0\_SSFIFO3\_R&0xFFF; //read bits 11-0

ADC0\_ISC\_R=0x0008; //clear flag

return data; // remove this, replace with real code

}

// constructor, invoked on creation of class

// m and b are linear calibration coeffients

SlidePot::SlidePot(uint32\_t m, uint32\_t b){

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

// initialize all private variables

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

data=0;

flag=0;

distance=0;

slope=m;

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(n);

flag=1;

}

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

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

// use calibration data to convert ADC sample to distance

return (1556\*n)/4096+311;; // replace this with solution

}

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

}

Print.s

; Print.s

; 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

; Runs on LM4F120 or TM4C123

; EE319K lab 7 device driver for any LCD

;

; As part of Lab 7, students need to implement these LCD\_OutDec and LCD\_OutFix

; This driver assumes two low-level LCD functions

; ST7735\_OutChar outputs a single 8-bit ASCII character

; ST7735\_OutString outputs a null-terminated string

IMPORT ST7735\_OutChar

IMPORT ST7735\_OutString

AREA |.text|, CODE, READONLY, ALIGN=2

THUMB

;-----------------------LCD\_OutDec-----------------------

; Output a 32-bit number in unsigned decimal format

; Input: R0 (call by value) 32-bit unsigned number

; Output: none

; Invariables: This function must not permanently modify registers R4 to R11

LCD\_OutDec PROC

EXPORT LCD\_OutDec

;put your Lab 7 solution here

n EQU 0

CMP R0, #10

BLO Base

PUSH {R0, LR}

MOV R2, #10

UDIV R0, R2

BL LCD\_OutDec

LDR R0, [SP,#n]

MOV R2, #10

UDIV R1, R0, R2

MUL R1, R2

SUB R0, R1

STR R0, [SP,#n]

ADD R0, #0x30

BL ST7735\_OutChar

POP {R0, PC}

Base PUSH {R0, LR}

ADD R0, #0x30

BL ST7735\_OutChar

POP {R0, PC}

BX LR

ENDP

;\* \* \* \* \* \* \* \* End of LCD\_OutDec \* \* \* \* \* \* \* \*

; -----------------------LCD \_OutFix----------------------

; Output characters to LCD display in fixed-point format

; unsigned decimal, resolution 0.001, range 0.000 to 9.999

; Inputs: R0 is an unsigned 32-bit number

; Outputs: none

; E.g., R0=0, then output "0.000 "

; R0=3, then output "0.003 "

; R0=89, then output "0.089 "

; R0=123, then output "0.123 "

; R0=9999, then output "9.999 "

; R0>9999, then output "\*.\*\*\* "

; Invariables: This function must not permanently modify registers R4 to R11

LCD\_OutFix PROC

EXPORT LCD\_OutFix

;put your Lab 7 solution here

VAL EQU 0

PUSH {R0, LR}

SUB SP, #8

MOV R1, #9999

CMP R0, R1

BHI stars

STR R0, [SP,#VAL]

MOV R2, #1000

LDR R3, [SP, #VAL]

UDIV R1,R3,R2

ADD R0, R1, #0x30

MUL R1, R2

SUB R3, R1

STR R3, [SP, #VAL]

BL ST7735\_OutChar

MOV R0, #0x2E

BL ST7735\_OutChar

MOV R2, #100

LDR R3, [SP, #VAL]

UDIV R1,R3,R2

ADD R0, R1, #0x30

MUL R1, R2

SUB R3, R1

STR R3, [SP, #VAL]

BL ST7735\_OutChar

MOV R2, #10

LDR R3, [SP, #VAL]

UDIV R1,R3,R2

ADD R0, R1, #0x30

MUL R1, R2

SUB R3, R1

STR R3, [SP, #VAL]

BL ST7735\_OutChar

LDR R0, [SP, #VAL]

ADD R0,#0x30

BL ST7735\_OutChar

B done

stars MOV R0,#0x2A

BL ST7735\_OutChar

MOV R0, #0x2E

BL ST7735\_OutChar

MOV R0,#0x2A

BL ST7735\_OutChar

MOV R0,#0x2A

BL ST7735\_OutChar

MOV R0,#0x2A

BL ST7735\_OutChar

done ADD SP, #8

POP {R0, LR}

BX LR

ENDP

ALIGN

;\* \* \* \* \* \* \* \* End of LCD\_OutFix \* \* \* \* \* \* \* \*

ALIGN ; make sure the end of this section is aligned

END ; end of file