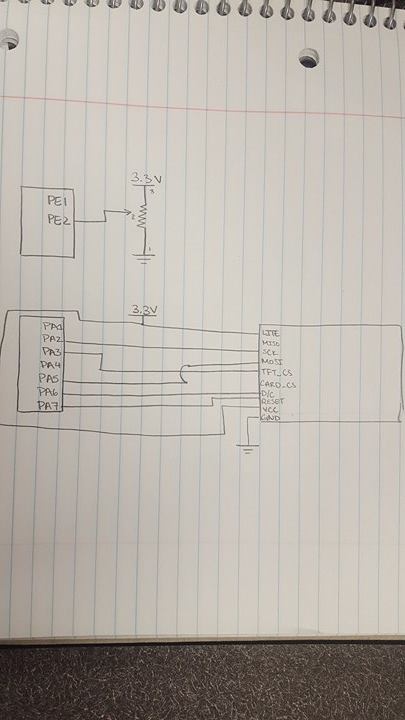
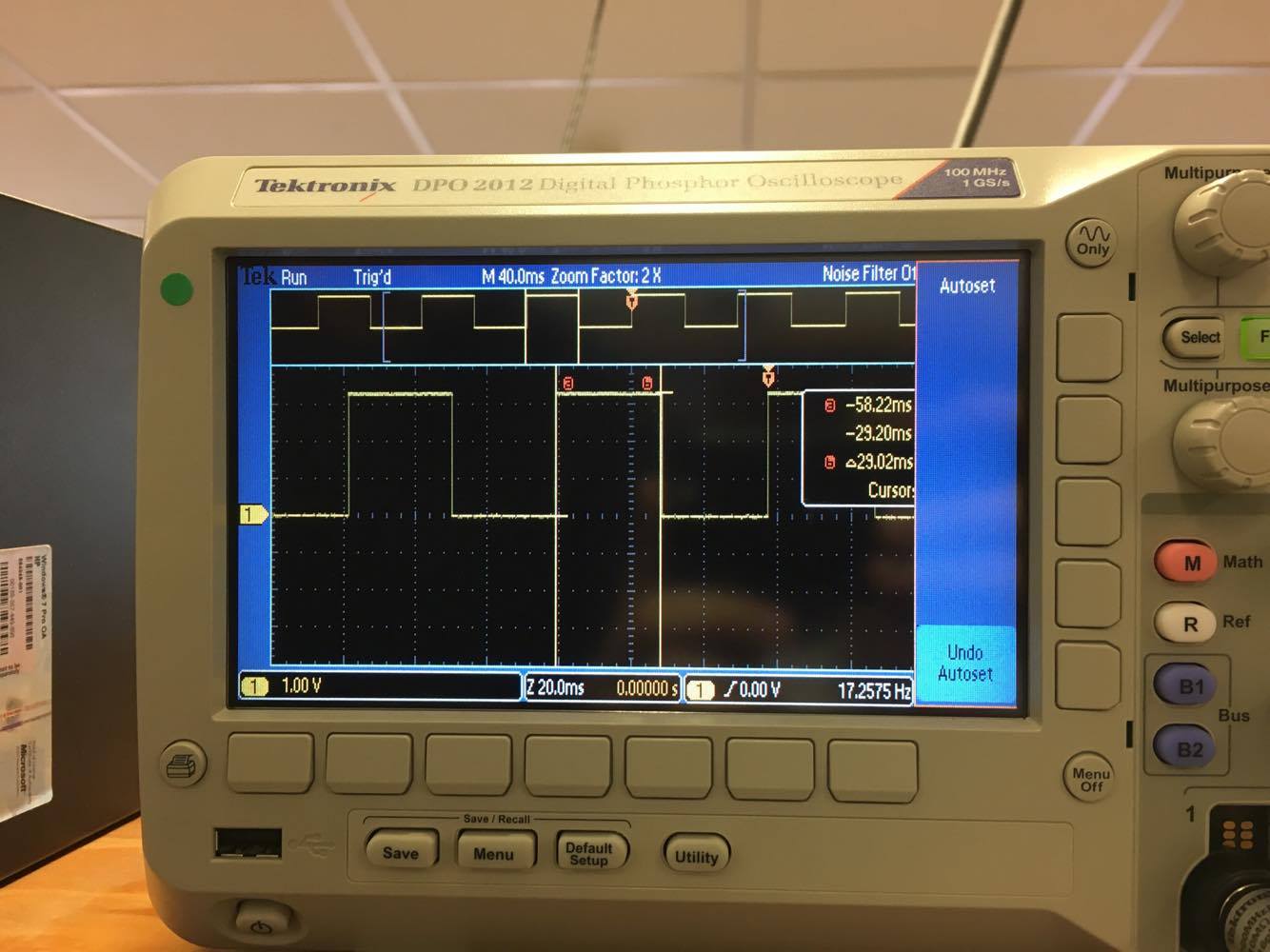
Lab 8 Deliverables

Daniel Canterino djc3323

Pranav Padmanabha pp9638





|  |  |  |
| --- | --- | --- |
| Analog Input | reading | point in .001cm |
| 0.24 | 0 | 0 |
| 1.12 | 322 | 500 |
| 2.05 | 1616 | 1000 |
| 2.92 | 3192 | 1500 |
| 3.28 | 4092 | 2000 |

// Lab8.c

// Runs on LM4F120 or TM4C123

// Student names: Pranav Padmanabha and Daniel Canterino

// Last modification date: 4/8/2017

// Last Modified: 4/5/2016

// Analog Input connected to PE2=ADC1

// displays on Sitronox ST7735

// PF3, PF2, PF1 are heartbeats

#include <stdint.h>

#include "ST7735.h"

#include "TExaS.h"

#include "ADC.h"

#include "print.h"

#include "tm4c123gh6pm.h"

//\*\*\*\*\*the first three main programs are for debugging \*\*\*\*\*

// main1 tests just the ADC and slide pot, use debugger to see data

// main2 adds the LCD to the ADC and slide pot, ADC data is on Nokia

// main3 adds your convert function, position data is no Nokia

void DisableInterrupts(void); // Disable interrupts

void EnableInterrupts(void); // Enable interrupts

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

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

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

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

#define PD0 (\*((volatile uint32\_t \*)0x40007004))

volatile uint32\_t Counts;

int ADCmail;

int ADCstatus;

void PortF\_Init(void){

SYSCTL\_RCGCGPIO\_R|=0x20;

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

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

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

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

GPIO\_PORTF\_AFSEL\_R&=~(0x0E);//turn on pf123 for heartbeats, no inputs, only outputs, disable af and am

GPIO\_PORTF\_AMSEL\_R&=~(0x0E);

GPIO\_PORTF\_PCTL\_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

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

ADC\_Init(); // turn on ADC, set channel to 1

while(1){

Data = ADC\_In(); // sample 12-bit channel 1

}

}

int main2(void){

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

ADC\_Init(); // turn on ADC, set channel to 1

ST7735\_InitR(INITR\_REDTAB);

PortF\_Init();

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 1

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

}

}

uint32\_t Convert(uint32\_t input){

input=((4373\*input)+1984300)/10000; ///NEED TO STABILIZE DATA AND GET DATA FOR LINE, RIGHT NOW IT IS NOT LINEAR

return input;

}

int main3(void){

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

ST7735\_InitR(INITR\_REDTAB);

PortF\_Init();

ADC\_Init(); // turn on ADC, set channel to 1

while(1){

PF2 ^= 0x04; // Heartbeat

Data = ADC\_In(); // sample 12-bit channel 1

PF3 = 0x08; // Profile Convert

Position = 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

}

}

void SysTick\_Init(uint32\_t period){

NVIC\_ST\_CTRL\_R = 0; //disable the interrupt during setup

NVIC\_ST\_RELOAD\_R = period - 1; //reload value (2\*10^6)

NVIC\_ST\_CURRENT\_R = 0; //clears data in current

NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R&0x00FFFFFF)|0x40000000; //priority 2

NVIC\_ST\_CTRL\_R = 0x00000007; //Enable the interrupts

}

void SysTick\_Handler(void){

GPIO\_PORTF\_DATA\_R^=0x02;//toggles pf1

GPIO\_PORTF\_DATA\_R^=0x04;//toggles pf2

Data= ADC\_In();//sets data equal to the adc input

ADCmail=Data;//sets global variable adcmail to adc input

ADCstatus=1;//sets global variable adcstatus to 1 to indicate there is new data available

GPIO\_PORTF\_DATA\_R^=0x08;//toggles pf3

}

int main(void){

TExaS\_Init();

// your Lab 8

ST7735\_InitR(INITR\_REDTAB);

PortF\_Init();

ADC\_Init(); // turn on ADC, set channel to 1

EnableInterrupts(); //enables interrupts after all initialization is complete

int output;

while(1){

SysTick\_Init(1160000); //calls systick init with 2\*10^6 ;;; 80MHz \* 40Hz (1/40)

while(ADCstatus==0){//waits for systick handler to set adc status to 1 indicating fresh data

}

output=Convert(ADCmail);// calls convert on the adc data stored at the global variable --> converts to cm stores to output

ST7735\_SetCursor(0,0);

LCD\_OutDec(ADCmail); //assembly function out dec

ST7735\_OutString(" ");

ST7735\_SetCursor(6,0);

LCD\_OutFix(output); //assembly function outfix

ST7735\_OutString(" cm");

ADCstatus=0; //resets adc status to 0

}

}

// ADC.c

// Runs on LM4F120/TM4C123

// Provide functions that initialize ADC0

// Last Modified: 3/6/2015

// 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 "tm4c123gh6pm.h"

// ADC initialization function

// Input: none

// Output: none

void ADC\_Init(void){

SYSCTL\_RCGCGPIO\_R|=0x10;//port e initialiation, pe2 is input, no outputs, enable alternate function and analong function on pe2, disable i/o function

while ((SYSCTL\_PRGPIO\_R&0x10) ==0){};//2 cycles

GPIO\_PORTE\_DIR\_R&=~(0x04);

GPIO\_PORTE\_AFSEL\_R|=(0x04);

GPIO\_PORTE\_DEN\_R&=~(0x04);

GPIO\_PORTE\_AMSEL\_R|=(0x04);

SYSCTL\_RCGCADC\_R|=0x01;//activate adc0

while ((SYSCTL\_RCGCADC\_R&0x01)==0){}; //4 cycles

ADC0\_PC\_R=0x01;//configure for 125K

ADC0\_SSPRI\_R=0x0123;//sequencer 3 is highest priority

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

ADC0\_EMUX\_R&=~(0xF000);//seq3 is software trigger

ADC0\_SSMUX3\_R=(ADC0\_SSMUX3\_R & 0xFFFFFFF0) +1;//clear ss3 field and set channel ain0 (pe2)

ADC0\_SSCTL3\_R= (0x0006);//no ts0 d0, yes ie0 end0

ADC0\_IM\_R &= ~(0x0008);//disable ss3 interrupts

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

ADC0\_SAC\_R=0x04;//AVERAGES 16 SAMPLES NOW

}

//------------ADC\_In------------

// Busy-wait Analog to digital conversion

// Input: none

// Output: 12-bit result of ADC conversion

uint32\_t ADC\_In(void){

uint32\_t result;

ADC0\_PSSI\_R = 0x0008;//initiate ss3

while ((ADC0\_RIS\_R & 0x08) == 0){//wait for conversion done

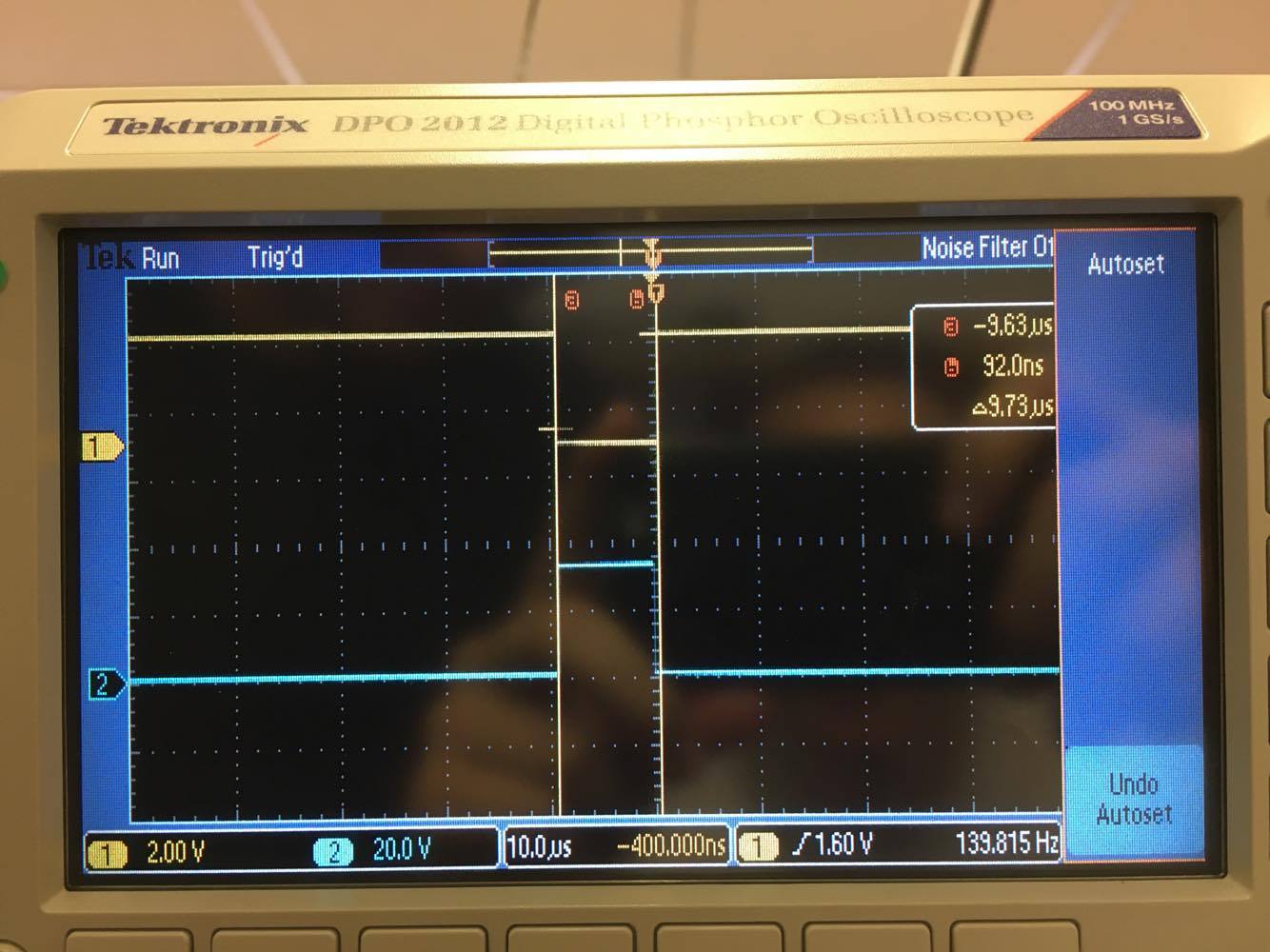
}

result=(ADC0\_SSFIFO3\_R & 0xFFF);//read 12-bit result

ADC0\_ISC\_R = 0x0008;//acknowledge completion

return (result);

}



|  |  |  |
| --- | --- | --- |
| True Position xti | Measured Position xmi | Error xti-xmi |
| 1.17cm | 1.155cm | .015cm |
| .52cm | .489cm | .031cm |
| 1.68 | 1.709 | .011cm |
| .41cm | .319 | .091cm |
| .73cm | .738cm | .008cm |