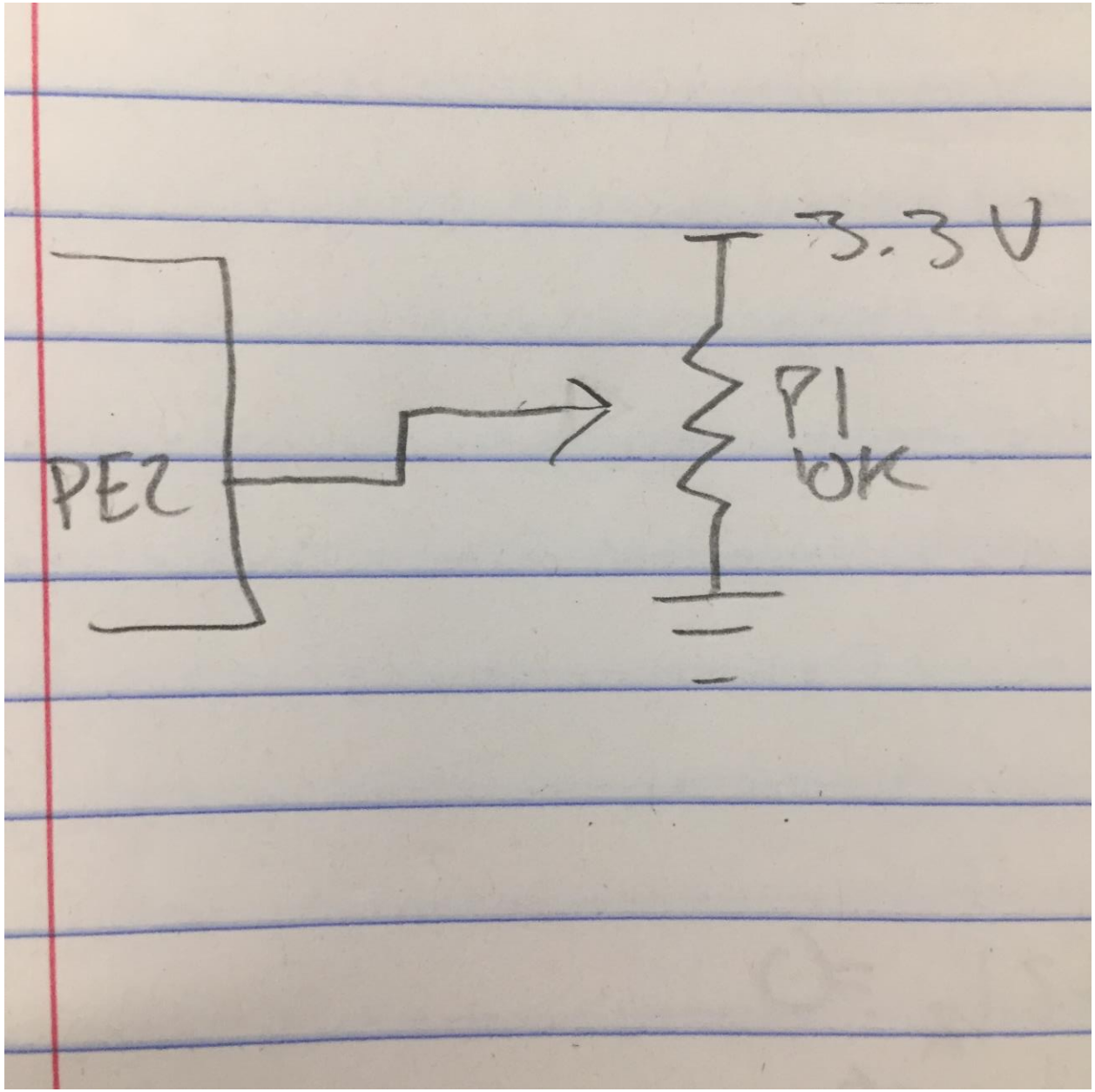
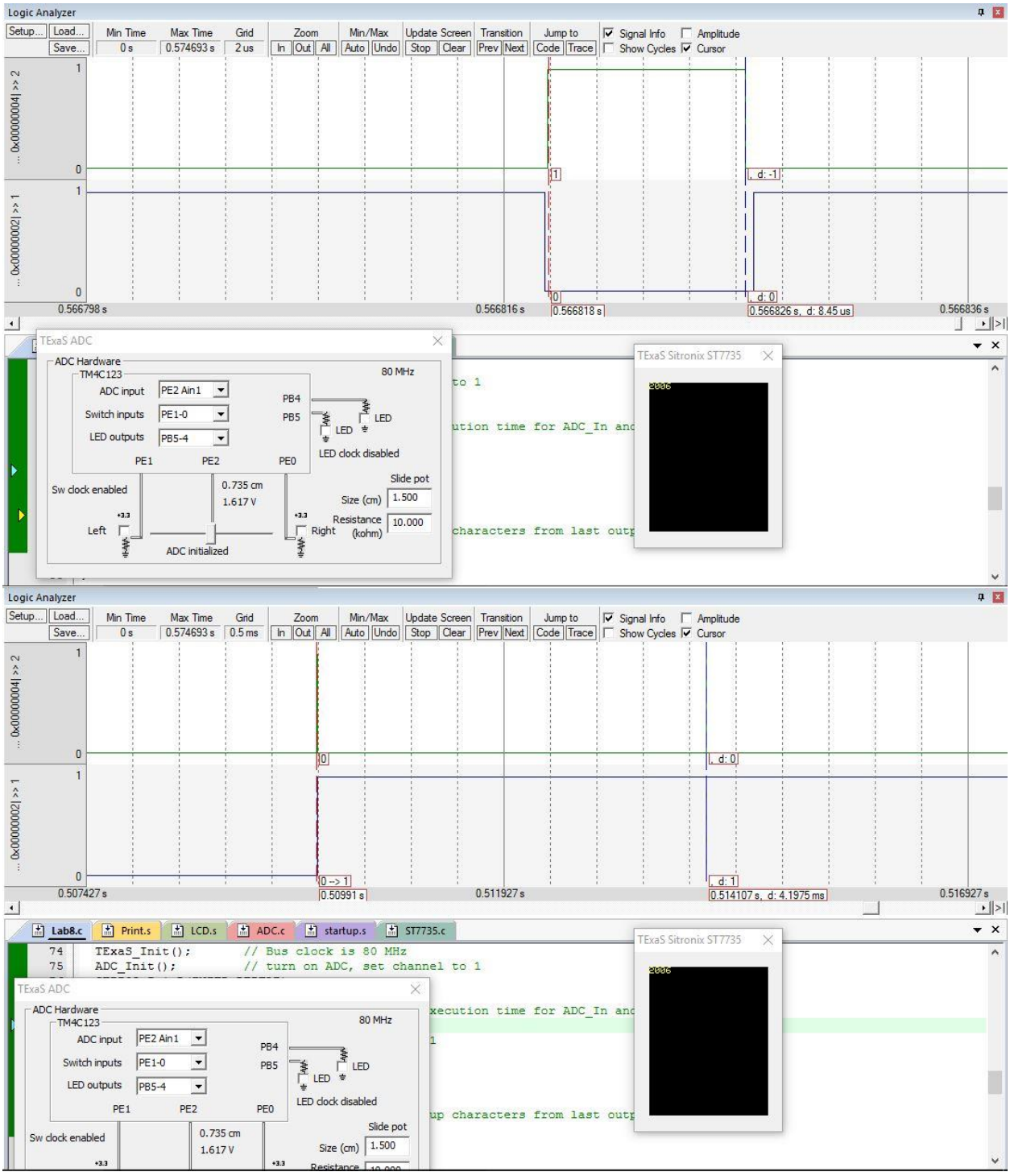
```
// ADC.c
// Runs on LM4F120/TM4C123
// Provide functions that initialize ADC0
// Last Modified: 11/8/2017
// Student names: Fawadul Haq and Rafael Herrejon
// Last modification date: change this to the last modification date or look very silly

#include <stdint.h>
#include "tm4c123gh6pm.h"

// ADC initialization function
// Input: none
// Output: none
void ADC_Init(void){ volatile uint32_t delay;
        SYSCTL_RCGCGPIO_R |= 0x30;      // 1) activate clock for Port E and F
  while((SYSCTL_PRGPIO_R&0x30) == 0){};
  GPIO_PORTE_DIR_R &= ~0x04;      // 2) make PE2 input
  GPIO_PORTE_AFSEL_R |= 0x04;     // 3) enable alternate fun on PE2
  GPIO_PORTE_DEN_R &= ~0x04;      // 4) disable digital I/O on PE2
  GPIO_PORTE_AMSEL_R |= 0x04;     // 5) enable analog fun on PE2

        SYSCTL_RCGCADC_R |= 0x01;   // Enable ADC clock
        delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;
        ADC0_PC_R      = 0x01;      // Set 125kHz ADC conversion speed
        ADC0_SSPRI_R   = 0x0123;    // Set sequencer priority: sequence 3 is highest priority
        ADC0_ACTSS_R  &= ~(0x08);   // Disable selected sequence 3
        ADC0_EMUX_R   &= ~(0xF000); // Set software start trigger event
        ADC0_SSMUX3_R &= ~(0x0F);   // Set input source (channel 1: PE2)
        ADC0_SSMUX3_R |= 0x01;
        ADC0_SSCTL3_R  = 0x06;      // Set sample control bits
        ADC0_IM_R     &= ~(0x08);   // Disable interrupts
        ADC0_ACTSS_R  |= 0x08;      // Enable selected sequencer 3
}

//------------ADC_In------------
// Busy-wait Analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
uint32_t ADC_In(void){
  uint32_t output;

        ADC0_PSSI_R |= 0x08;            // set software trigger
        while((ADC0_RIS_R&0x08) == 0){}    // busy-wait for Raw Interrupt Status
        output = ADC0_SSFIFO3_R&0x0FFF;    // Read the ADC Data
        ADC0_ISC_R = 0x08;              // Clear sample complete flag
```

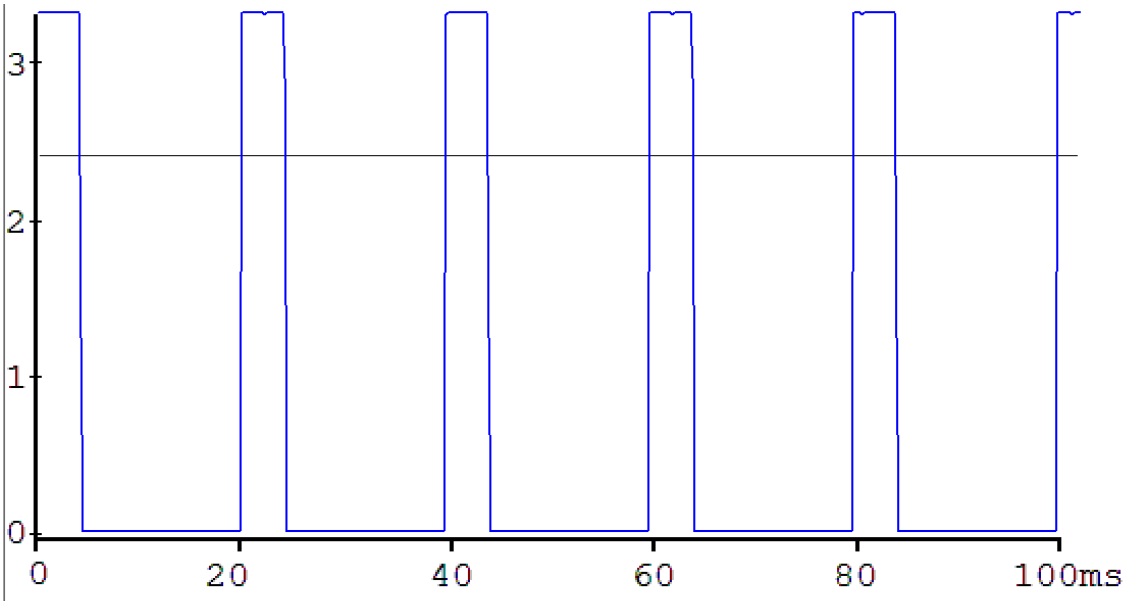
```
	return output;
}

// Lab8.c
// Runs on LM4F120 or TM4C123
// Student names: Fawadul Haq and Rafael Herrejon
// Last modification date: change this to the last modification date or look very silly
// Last Modified: 11/8/2017

// 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 on Nokia

void DisableInterrupts(void);      // Disable interrupts
void EnableInterrupts(void);       // Enable interrupts
void SysTick_Init(uint32_t period); // Initialize systick interrupts
void SysTick_Handler(void);        // Systick Interrupt, samples ADC

#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
void PortF_Init(void){
	//F
	GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x1F;          // allow changes to PF4-0
  // only PF0 needs to be unlocked, other bits can't be locked
  GPIO_PORTF_AMSEL_R = 0x00;       // 3) disable analog on PF
  GPIO_PORTF_PCTL_R = 0x00000000;   // 4) PCTL GPIO on PF4-0
  GPIO_PORTF_DIR_R = 0x0E;         // 5) PF4,PF0 in, PF3-1 out
  GPIO_PORTF_AFSEL_R = 0x00;       // 6) disable alt funct on PF7-0
  GPIO_PORTF_PDR_R = 0x11;         // enable pull-down on PF0 and PF4
  GPIO_PORTF_DEN_R = 0x1F;         // 7) enable digital I/O on PF4-0
}
```

```
void SysTick_Init(uint32_t period){
  NVIC_ST_CTRL_R = 0;
          NVIC_ST_RELOAD_R = period;
          NVIC_ST_CURRENT_R = 0;
          NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; // priority 2
          NVIC_ST_CTRL_R = 0x00000007; //0111
}


uint32_t Data;       // 12-bit ADC
uint32_t ADCMail;    // input from Systick Handler sampling ADC
uint32_t Position;   // 32-bit fixed-point 0.001 cm
uint8_t flag = 0;        // ADC flag

void SysTick_Handler(void){
          PF3 ^= 0x08;
          PF3 ^= 0x08;
          ADCMail = ADC_In();
          flag = 1;
          PF3 ^= 0x08;
}


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("    ");  // spaces cover up characters from last output
    PF1 = 0;         // end of LCD Profile
  }
}
```

```
uint32_t Convert(uint32_t input){
  return ((511*input)/1250)+263;    // write this your self
}
int main3(void){
  TExaS_Init();        // Bus clock is 80 MHz
  ST7735_InitR(INITR_REDTAB);
  ADC_Init();        // turn on ADC, set channel to 1
            PortF_Init();
  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
  }
}
int main(void){
  TExaS_Init();
            ST7735_InitR(INITR_REDTAB);
            DisableInterrupts();
            SysTick_Init(1600000); // change 0 to whatever the RELOAD needs to be 50 Hz
            ADC_Init();   // inits port E (clock) and ADC
            PortF_Init(); // inits Port F
            EnableInterrupts();
  // your Lab 8
  while(1){
                        while(flag == 0){} // busy wait for systick to sample ADC
                        Data = ADCMail;          // Getting mail
                        flag = 0;

                        PF2 ^= 0x04;
                        Position = Convert(Data); // COnverting
                        PF2 ^= 0x04;

                        PF1 ^= 0x02;
                        ST7735_SetCursor(6,0);    // Outputting
                        LCD_OutFix(Position);
    PF1 ^= 0x02;
            }
}
```


3.3V
PE2
PI
10K


Logic Analyzer
Setup... Load... Save...
Min Time 0 s
Max Time 0.574693 s
Grid 2 us
Zoom In Out All
Min/Max Auto Undo
Update Screen Stop Clear
Transition Prev Next
Jump to Code Trace
Signal Info
Amplitude
Show Cycles
Cursor
0x00000004| >> 2
0x00000002| >> 1
0.566798 s
0.566816 s
0.566818 s
0.566826 s, d: 8.45 us
0.566836 s
TExaS ADC
ADC Hardware
TM4C123
ADC input PE2 Ain1
Switch inputs PE1-0
LED outputs PB5-4
80 MHz
PB4
PB5
LED
LED clock disabled
PE1 PE2 PE0
Sw clock enabled
0.735 cm
1.617 V
Slide pot
Size (cm) 1.500
Resistance (kohm) 10.000
Left
Right
ADC initialized
TExaS Sitronix ST7735
Logic Analyzer
0.507427 s
0.50991 s
0.511927 s
0.514107 s, d: 4.1975 ms
0.516927 s
Lab8.c
Print.s
LCD.s
ADC.c
startup.s
ST7735.c
TExaS_Init(); // Bus clock is 80 MHz
ADC_Init(); // turn on ADC, set channel to 1



Ave=0.77V, Peak-peak=3.27V, Period=20.0ms, Freq= 50Hz
high-pulse=4.4ms, low-pulse=15.6ms

Position	Analog input	ADC sample	Correct fixed point	Measured fixed point
0	0.0002	17	0	
0.3	0.0015	34	300	277
0.6	0.591	782	600	583
0.9	1.383	1695	900	956
1.2	1.886	2310	1200	1207
1.5	2.504	3083	1500	1523
1.8	3.099	3874	1800	1847
2	3.241	4058	2000	1922

True Position	Measured Position	Difference	
0.3	0.292	0.008	0.008
0.6	0.606	-0.006	0.006
0.9	0.906	-0.006	0.006
1.2	1.249	-0.049	0.049
1.5	1.557	-0.057	0.057
1.8	1.818	-0.018	0.018
		Average ->	0.024