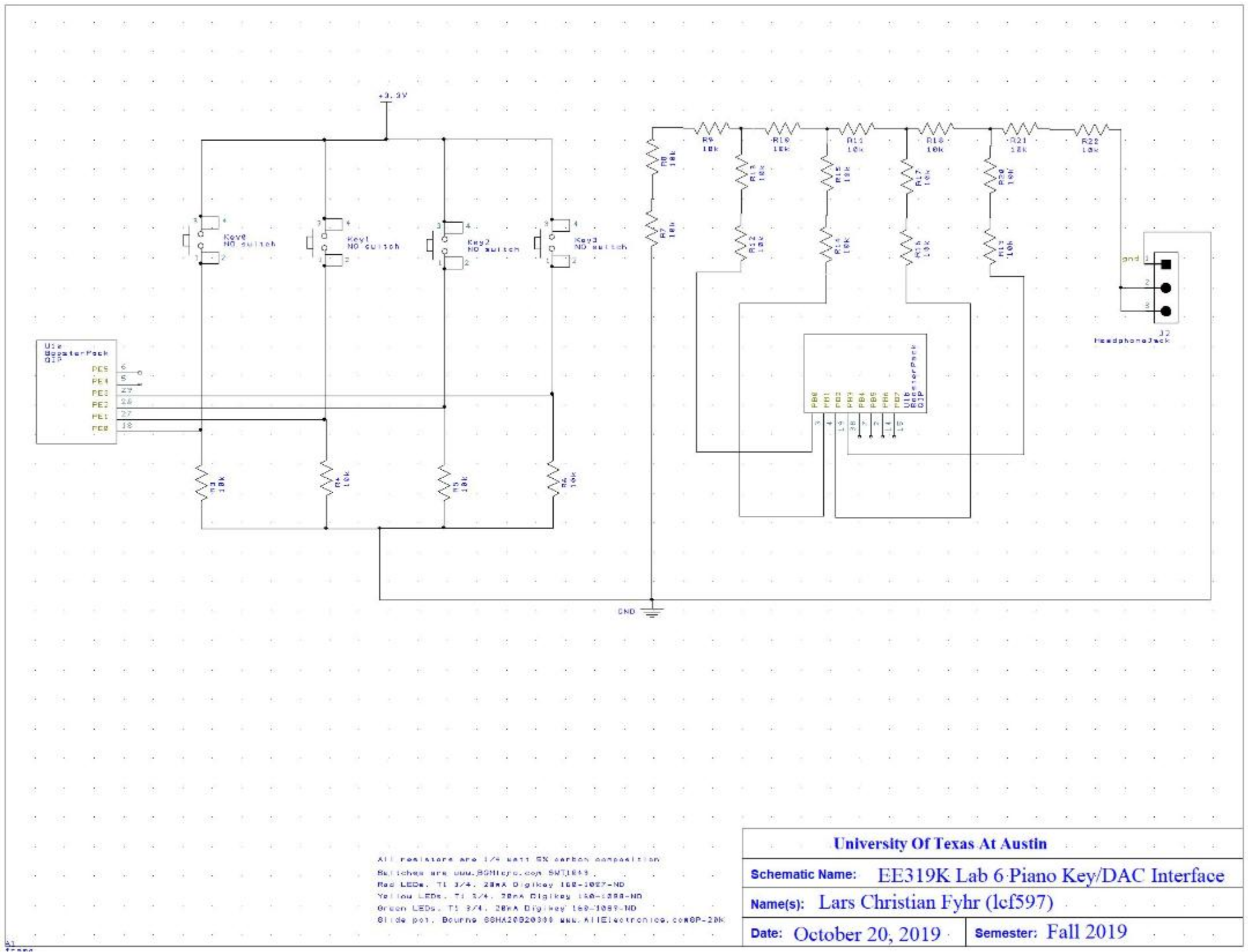
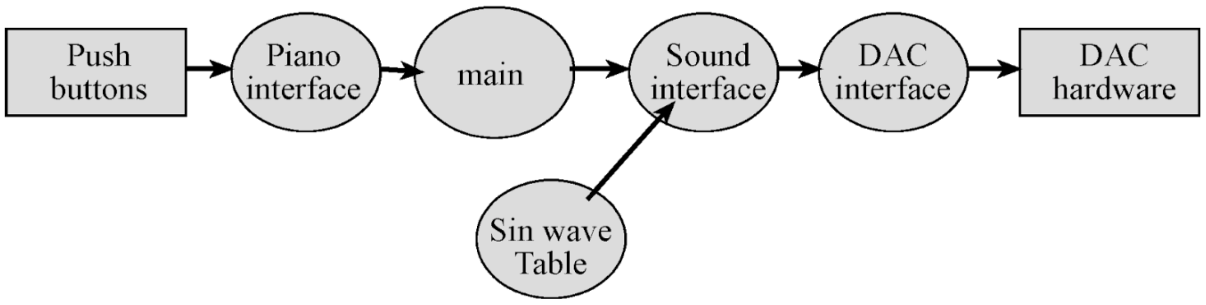
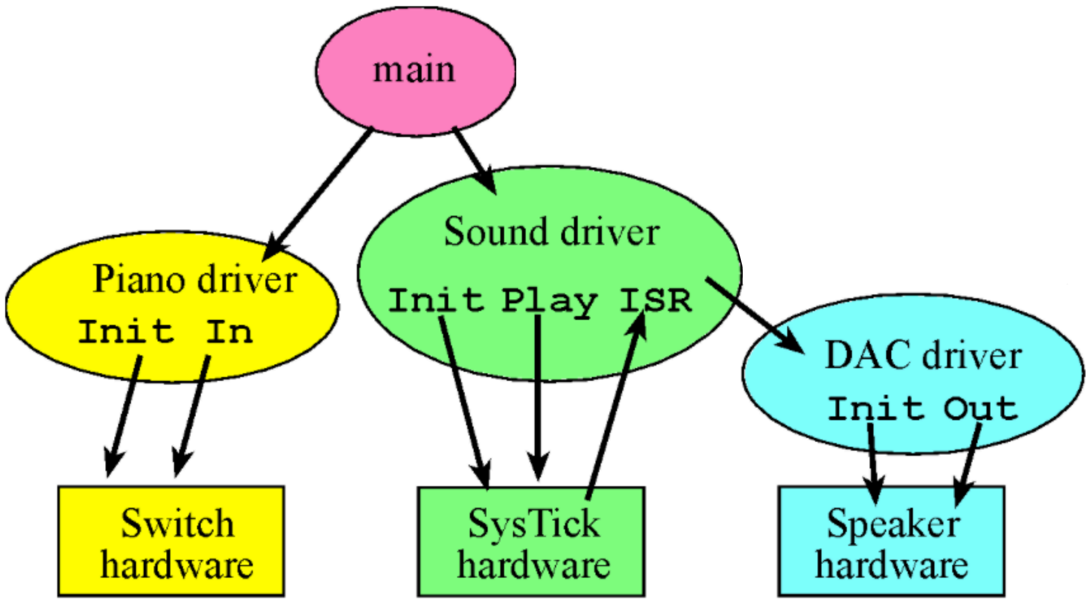
Lars Christian Fyhr (lcf597)

Lab 6 Deliverables

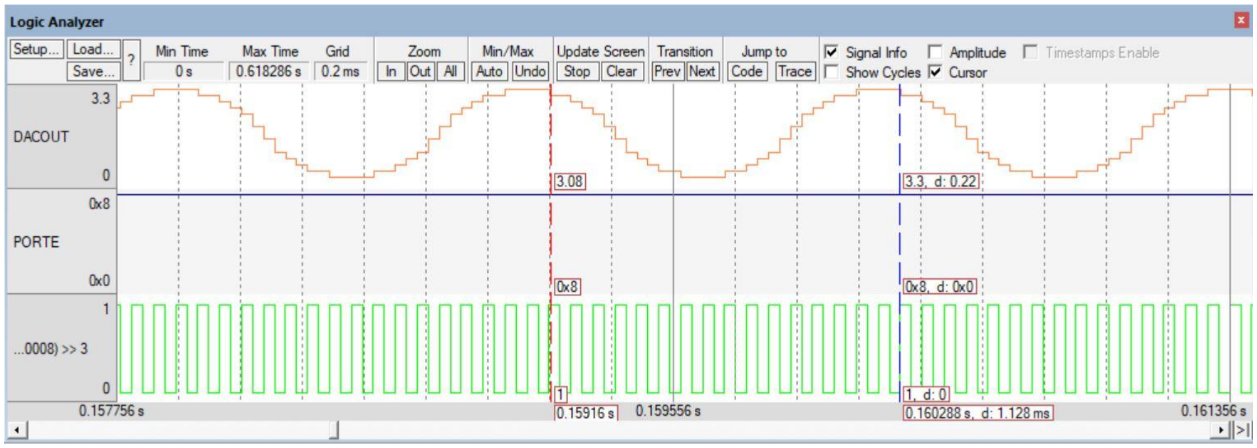

+3.3V
Key0 NO switch
Key1 NO switch
Key2 NO switch
Key3 NO switch
U1a BoosterPack DIP
PE5
PE4
PE3
PE2
PE1
PE0
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R9 10k
R10 10k
R11 10k
R18 10k
R21 10k
R22 10k
R12 10k
R13 10k
R14 10k
R15 10k
R16 10k
R17 10k
R19 10k
R20 10k
U1b BoosterPack DIP
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
gnd
J2 HeadphoneJack
GND
All resistors are 1/4 watt 5% carbon composition
Switches are www.BGMicro.com SWT1049.
Red LEDs. T1 3/4. 20mA Digikey 160-1027-ND
Yellow LEDs. T1 3/4. 20mA Digikey 160-1088-ND
Green LEDs. T1 3/4. 20mA Digikey 160-1089-ND
Slide pot. Bourns 08HA20B20300 www.AllElectronics.com8P-20K
University Of Texas At Austin
Schematic Name: EE319K Lab 6 Piano Key/DAC Interface
Name(s): Lars Christian Fyhr (lcf597)
Date: October 20, 2019
Semester: Fall 2019

DATA STRUCTURES/DATA FLOW/CALL GRAPH

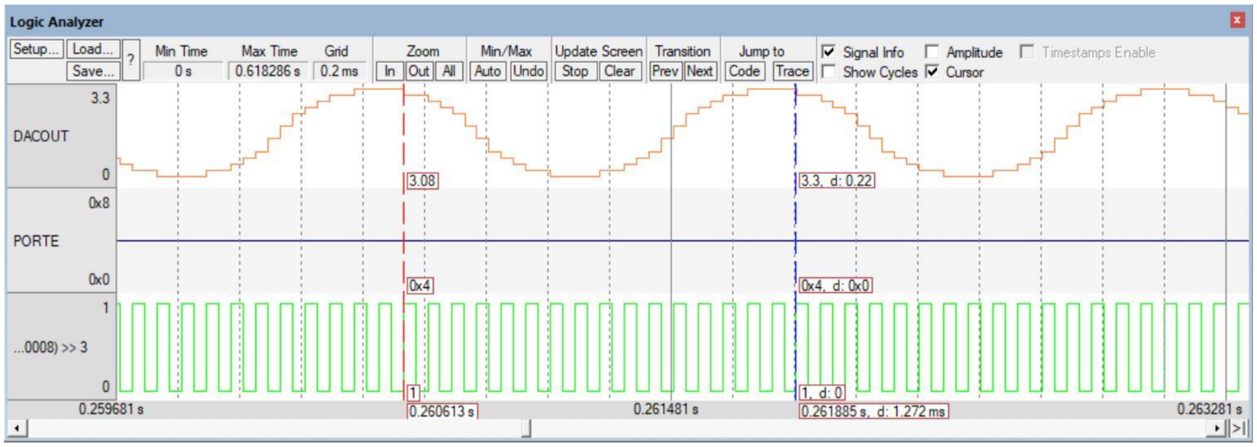




DUAL SCOPE (TexasDisplay)



PORTE = x08, Period = 1.128 ms



PORTE = x04, Period = 1.272 ms

Measurement Data

Bit3 bit2 bit1 bit0	Theoretical DAC voltage	Measured DAC voltage
0	0 V	0.001 V
1	0.22 V	0.216 V
2	0.44 V	0.425 V
3	0.66 V	0.653 V
4	0.88 V	0.867 V
5	1.10 V	1.077 V
6	1.32 V	1.311 V
7	1.54 V	1.509 V
8	1.76 V	1.742 V
9	1.98 V	1.956 V
10	2.2 V	2.188 V
11	2.42 V	2.400 V
12	2.64 V	2.604 V
13	2.86 V	2.809 V
14	3.08 V	2.982 V
15	3.3 V	3.194 V

Resolution: 0.0998 V
Range: 3.193 V
Precision: 4 bit for entries
Accuracy: ~98.3%

ANSWERS:

1. The interrupt trigger occurs once the count flag of SysTick is triggered, thus triggering the interrupt. Essentially it triggers after the Sound_Play(value) that is entered into the reload register is done counting.
2. The SysTick_Handler vector, or the interrupt vector we used for this lab, is located in the startup file.
3. First it finishes executing the current instruction, next it will push the eight necessary registers to the stack, LR is set to the address of xFFFFFFF9, IPSR is set to the interrupt number, and PC is loaded with interrupt vector. (Context Switch)
4. Since the LR holds the “special value” above, it will know to pop the eight registers off the stack, and return the PC to the instruction that was last executed + 1, which returns control to main program.

MAIN CODE

```
// Lab6.c
// Runs on LM4F120 or TM4C123
// Use SysTick interrupts to implement a 4-key digital piano
// MOOC lab 13 or EE319K lab6 starter
// Program written by: Lars Christian Fyhr (lcf597)
// Date Created: 3/6/17
// Last Modified: 10/21/19
// Lab number: 6
// Hardware connections
// PE0-3: Input Switches for Piano Keys
// PB0-3: Output Ports to DAC (R= xxx Kohms)

#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
#include "Sound.h"
#include "Piano.h"
#include "TExaS.h"

#define A 5681/2;
#define G 6378/2;
#define E 7585/2;
#define C 9557/2;


// basic functions defined at end of startup.s
void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts

int main(void){
			DisableInterrupts();
  TExaS_Init(SW_PIN_PE3210,DAC_PIN_PB3210,ScopeOn);    // bus clock at 80 MHz
  Piano_Init();
  Sound_Init();
			Heartbeat_Init();
			int Key;
			int frq;
										// initialize the piano key and its output frequency
  while(1){
				Key = Piano_In();
				if (Key == 0x08) {
					frq = A;
				}
				if (Key == 0x04) {
					frq = G;
				}
										// set output frequency given inputted key
				if (Key == 0x02) {
					frq = E;
				}
				if (Key == 0x01) {
					frq = C;
				}
				if (Key != 0x00) {
					EnableInterrupts();			// when keys pressed, play given sound at frequency
					Sound_Play(frq);
				}
				if (Key == 0x00) {
					DisableInterrupts();			// when keys not pressed, stop sounds
				}
  }
}
// Piano.c
// This software configures the off-board piano keys
// Lab 6 requires a minimum of 4 keys, but you could have more
// Runs on LM4F120 or TM4C123
// Program written by: Lars Christian Fyhr (lcf597)
// Date Created: 3/6/17
// Last Modified: 10/21/19
// Lab number: 6
// Hardware connections
// PE0-3: Input Switches for Piano Keys

// Code files contain the actual implemenation for public functions
// this file also contains an private functions and private data
#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
```

```
// **************Piano_Init*********************
// Initialize four piano key inputs, called once to initialize the digital ports
// Input: none
// Output: none
void Piano_Init(void){
			volatile int delay;
  SYSCTL_RCGCGPIO_R |= 0x10;   // enable Port E Clock
  delay = 0;
  GPIO_PORTE_DIR_R &= ~(0xF0);     // input on PE0-3
  GPIO_PORTE_DEN_R |= 0x0F;     // enable digital on PE0-3
}

// **************Piano_In*********************
// Input from piano key inputs
// Input: none
// Output: 0 to 15 depending on keys
//   0x01 is just Key0, 0x02 is just Key1, 0x04 is just Key2, 0x08 is just Key3
//   bit n is set if key n is pressed
uint32_t Piano_In(void){
			int note;
			note = GPIO_PORTE_DATA_R;
  return note; // returns the note to be played
}




// Sound.c
// This module contains the SysTick ISR that plays sound
// Runs on LM4F120 or TM4C123
// Program written by: Lars Christian Fyhr (lcf597)
// Date Created: 3/6/17
// Last Modified: 10/21/19
// Lab number: 6
// Hardware connections
// PE0-3: Input Switches for Piano Keys
// PB0-3: Output Ports to DAC (R-2R DAC: R = 5 Kohms)

// Code files contain the actual implementation for public functions
// this file also contains an private functions and private data
#include <stdint.h>
#include "dac.h"
#include "../inc/tm4c123gh6pm.h"

// initialize SineWave approximation array and the index

uint32_t SineWave[32] = {7,9,11,12,13,14,14,15,15,15,15,15,14,14,13,12,11,9,7,5,4,3,2,2,1,1,1,1,2,2,3,4,5};
int indx;
//int heart;										// only to see heartbeat on board


// **************Sound_Init*********************
// Initialize digital outputs and SysTick timer
// Called once, with sound/interrupts initially off
// Input: none
// Output: none
void Sound_Init() {
			DAC_Init();
			indx = 0;
			NVIC_ST_CTRL_R = 0;
			NVIC_ST_RELOAD_R = 0x000000FF;
			NVIC_ST_CURRENT_R = 0;
			NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R & 0x00FFFFFF) | 0x20000000;
			NVIC_ST_CTRL_R = 0x0007;
}

// **************Heartbeat_Init*********************
// Initialize digital on-board output for flashing heartbeat
// Called once, with interrupts off
// Input: none
// Output: none
void Heartbeat_Init(void) {
			volatile int delay;
  SYSCTL_RCGCGPIO_R |= 0x20;   // enable Port F Clock
  delay = 0;
  GPIO_PORTF_DIR_R |= 0x08;     // output on PF3
  GPIO_PORTF_DEN_R |= 0x08;     // enable digital on PF3
}
```

```
// **************SysTick_Handler*********************
// Toggles heartbeat, sends information to DAC_Out
// Use crossed out lines to visually see heartbeat on board
// Called everytime Count flag runs down, when interrupts on
// Input: none
// Output: none
void SysTick_Handler(void) {
//                                      if (heart%1000 == 1) {
                                        GPIO_PORTF_DATA_R ^= 0x08;
//                                      }
                                        indx = (indx + 1)%32;
                                        DAC_Out(SineWave[indx]);                    // outputs one DAC output per handle for a whole cycle (then repeats)
//                                      heart++;
}

// **************Sound_Play*********************
// Start sound output, and set Systick interrupt period
// Sound continues until Sound_Play called again
// This function returns right away and sound is produced using a periodic interrupt
// Input: interrupt period
//          Units of period to be determined by YOU
//          Maximum period to be determined by YOU
//          Minimum period to be determined by YOU
//        if period equals zero, disable sound output
// Output: none


void Sound_Play(uint32_t period){
                                        NVIC_ST_RELOAD_R = period - 1;
}
```

```
// dac.c
// This software configures DAC output
// Lab 6 requires a minimum of 4 bits for the DAC, but you could have 5 or 6 bits
// Runs on LM4F120 or TM4C123
// Program written by: Lars Christian Fyhr (lcf597)
// Date Created: 3/6/17
// Last Modified: 10/21/18
// Lab number: 6
// Hardware connections
// PB0-3: Output Ports to DAC (R2-R DAC: R = 5 Kohms)

#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
// Code files contain the actual implemenation for public functions
// this file also contains an private functions and private data

// **************DAC_Init*********************
// Initialize 4-bit DAC, called once
// Input: none
// Output: none
void DAC_Init(void){
                    volatile int delay;
  SYSCTL_RCGCGPIO_R |= 0x02;   // enable Port B clock
  delay = 0;
  GPIO_PORTB_DIR_R |= 0x0F;     // output on PB0-3
  GPIO_PORTB_DEN_R |= 0x0F;     // enable digital on PB0-3
}

// **************DAC_Out*********************
// output to DAC
// Input: 4-bit data, 0 to 15
// Input=n is converted to n*3.3V/15
// Output: none
void DAC_Out(uint32_t data){
                    GPIO_PORTB_DATA_R = data;                                          // outputs DAC value given from SineWave array
}
```