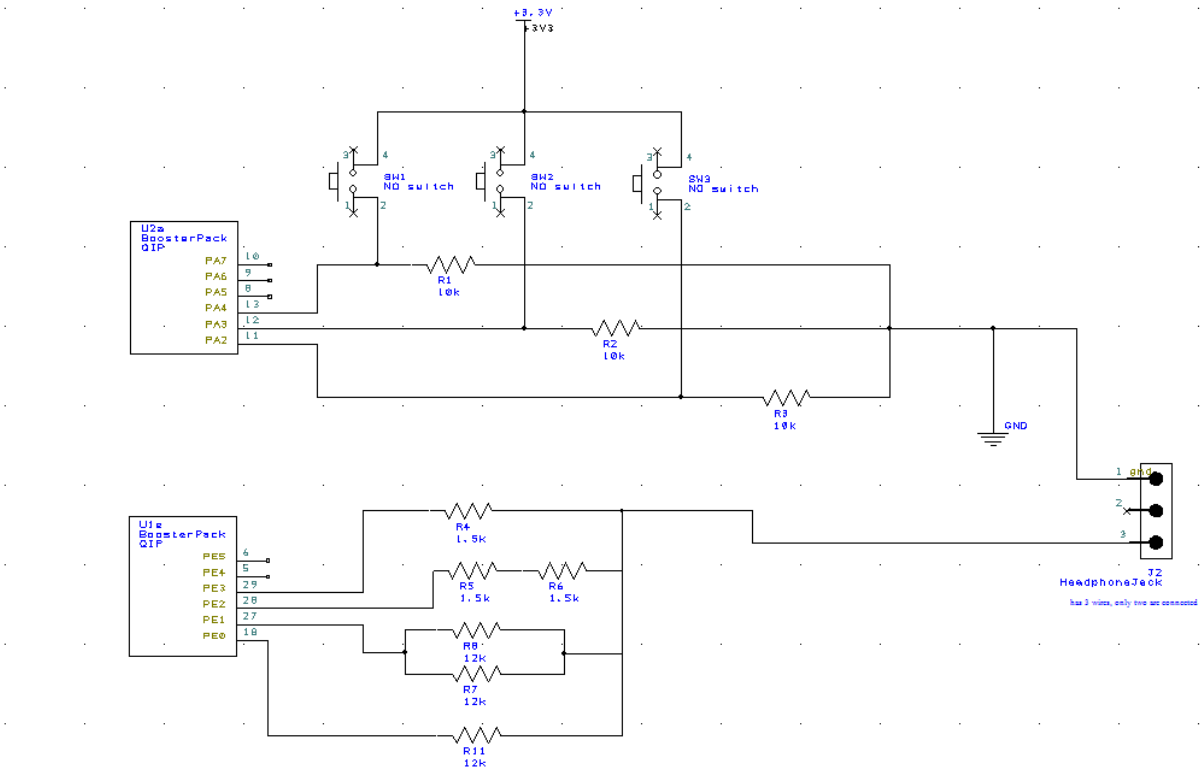
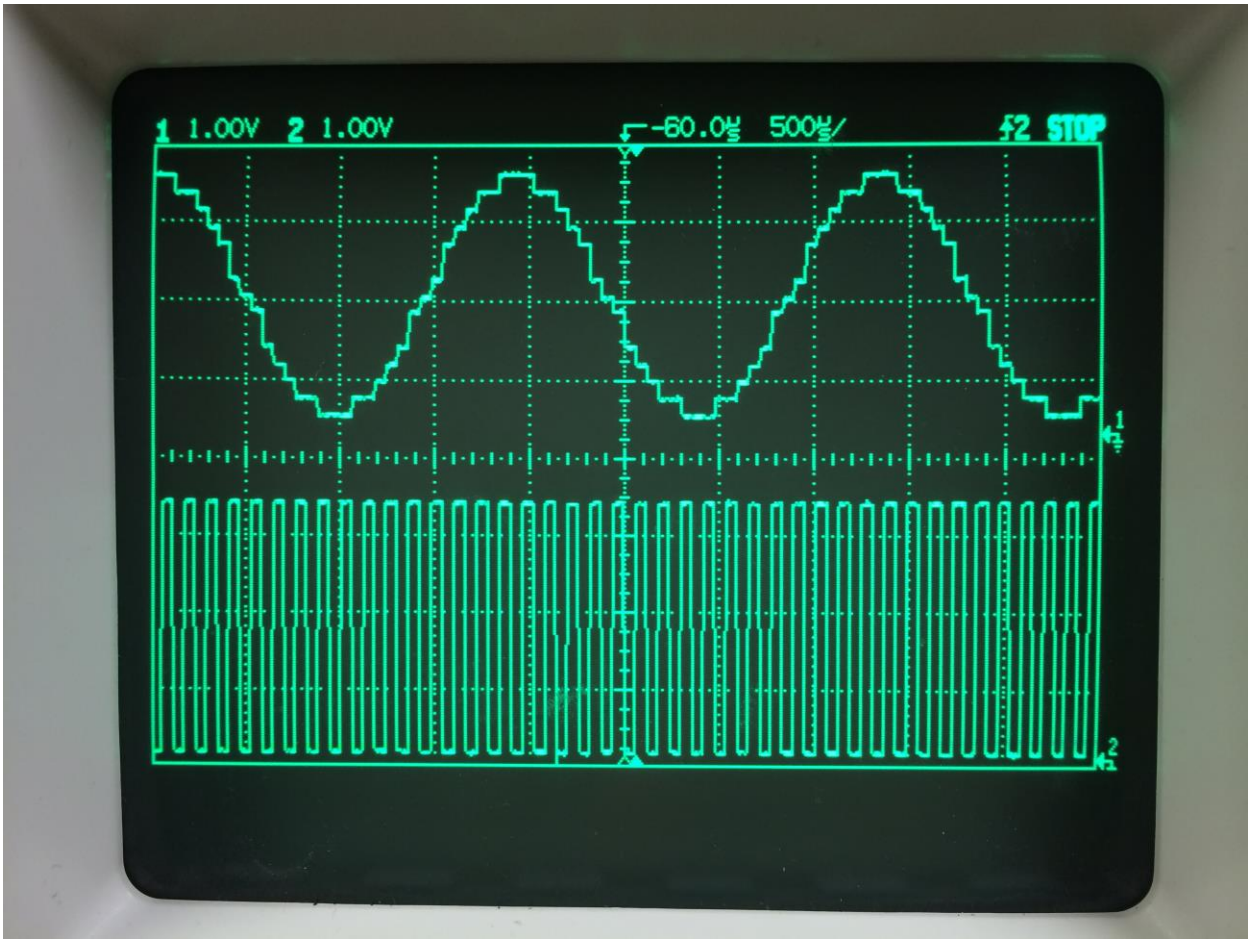
Ben Fu and Rochelle Roberts

Lab 6 Report

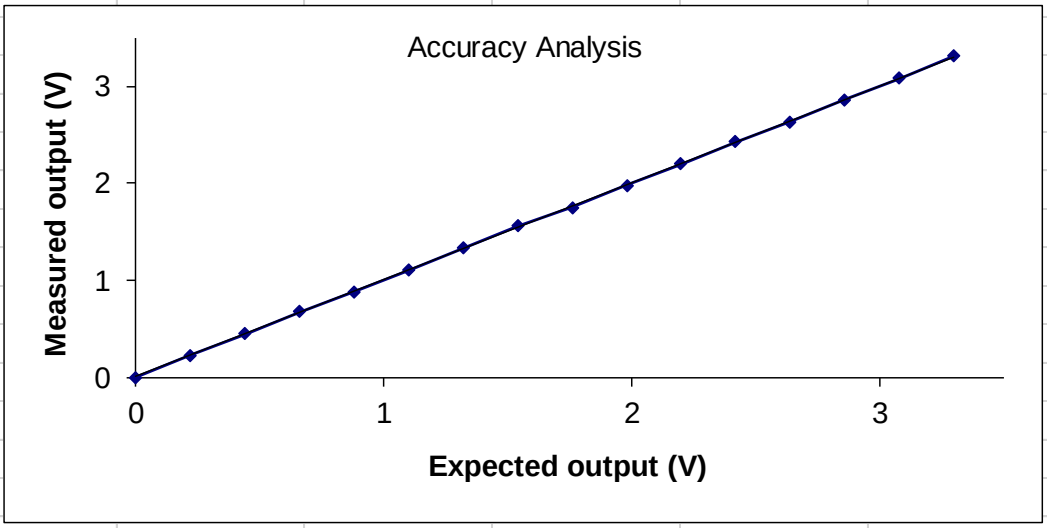


Array for Sine wave located in RAM

8
9
11
12
13
14
14
15
15
15
14
14
13
12
11
9
8
7
5
4
3
2
2
1
1
1
2
2
3
4
5
7


1 1.00V 2 1.00V
-60.0µs 500µs/
f2 STOP

	Actual (V)	Theory (V)	|Error| (V)	|Error|/3.3V	Resolution (V)	Accuracy
0	0.000	0.000	0.000	0.00%		0
1	0.224	0.220	0.004	0.12%	0.224	0.01818
2	0.450	0.440	0.010	0.30%	0.226	0.02273
3	0.674	0.660	0.014	0.42%	0.224	0.02121
4	0.881	0.880	0.001	0.03%	0.207	0.00114
5	1.105	1.100	0.005	0.15%	0.224	0.00455
6	1.331	1.320	0.011	0.33%	0.226	0.00833
7	1.555	1.540	0.015	0.45%	0.224	0.00974
8	1.750	1.760	0.010	0.30%	0.195	0.00568
9	1.975	1.980	0.005	0.15%	0.225	0.00253
10	2.201	2.200	0.001	0.03%	0.226	0.00045
11	2.426	2.420	0.006	0.18%	0.225	0.00248
12	2.634	2.640	0.006	0.18%	0.208	0.00227
13	2.858	2.860	0.002	0.06%	0.224	0.0007
14	3.084	3.080	0.004	0.12%	0.226	0.0013
15	3.309	3.300	0.009	0.27%	0.225	0.00273
Average accuracy of full scale(V)=			0.006	0.20%		
Average resolution (V)=			0.221			



Resolution: range/precision=	3.3/15
Range: (max V - min V)=	(3.3V-0V)=3.3V
Precision:	4bits (16 alternatives)
Accuracy:	(actual-ideal)/ideal

Resolution = 3.3/15 = 0.22

Range = 3.3 V

Precision = 4*16 = 64

Accuracy = shown in table above

1.) The interrupt trigger occurs whenever systick reaches 0 (NVIC_ST_CURRENT_R = 0x00)
2.) The interrupt vector is in the startup.s file
3.) Steps
    a. 8 registers are pushed onto the stack (R0 on top)
    b. Vector address is loaded into the PC
    c. LR is set to 0xFFFFFFF9
    d. IPSR is set to interrupt number
    e. PC is set to ISR address
    f. ISR clears the interrupt flag
    g. ISR executes BX LR
4.) Since the top 24 bits of LR are 0xFFFFFFFF, the ISR pops the registers off the stack and uses the bottom 8 bits of LR to return from interrupt.

```
// Lab6.c
// Runs on LM4F120 or TM4C123
// Program written by: put your names here
// Date Created: 8/25/2013
// Last Modified: 10/9/2013
// Section 1-2pm     TA: Saugata Bhattacharyya
// Lab number: 6
// Brief description of the program
//   A digital piano with 4 keys and a 4-bit DAC
// Hardware connections


#include "tm4c123gh6pm.h"
#include "PLL.h"
#include "Sound.h"
#include "Piano.h"

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

void PortF_Init(void){unsigned long volatile delay;
     SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF; // activate port F
  delay = SYSCTL_RCGC2_R;                  // 2 NOP
  GPIO_PORTF_DIR_R |= 0x04;      // PF2 is output
  GPIO_PORTF_AFSEL_R &= ~0x04;   // disable alternate select function
on PF2
  GPIO_PORTF_DEN_R |= 0x04;      // dig enable for PF2
}

void delay_funct(void){unsigned long i; unsigned char x;
     for(i=0;i<800000;i++){
          x=1;
     }
          GPIO_PORTF_DATA_R ^= 0x04; //toggle PF2 for heartbeat
}

int main(void){
  PLL_Init();          // bus clock at 80 MHz
  // all initializations go here
     PortF_Init();                    //initialize Port F
     Piano_Init();       // Port A contains value from switches
     Sound_Init(4000);   // initialize SysTick timer
  EnableInterrupts();
  while(1){
          delay_funct();
          Sound_Play(Piano_In());

// main loop, read from switchs change sounds
  }
}
```

```
// dac.c
// This software configures DAC output
// Runs on LM4F120 or TM4C123
// Program written by: put your names here
// Date Created: 8/25/2013
// Last Modified: 10/9/2013
// Section 1-2pm     TA: Saugata Bhattacharyya
// Lab number: 6
// Hardware connections

#include "tm4c123gh6pm.h"
// put code definitions for the software (actual C code)
// this file explains how the module works

// **************DAC_Init*********************
// Initialize 4-bit DAC, called once
// Input: none
// Output: none
void DAC_Init(void){unsigned long volatile delay;
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOE; // activate port E
  delay = SYSCTL_RCGC2_R;                    // 2 NOP
     GPIO_PORTE_AMSEL_R &= ~0x0F;            // no analog
  GPIO_PORTE_PCTL_R &= ~0x0000FFFF;   // regular GPIO function
  GPIO_PORTE_DIR_R |= 0x0F;       // PE3-0 are outputs
  GPIO_PORTE_AFSEL_R &= ~0x0F;   // disable alternate select function
on PE3-0
  GPIO_PORTE_DEN_R |= 0x0F;       // dig enable for PE3-0
     GPIO_PORTE_DR8R_R |= 0x0F;             //drive speakers
}

// **************DAC_Out*********************
// output to DAC
// Input: 4-bit data, 0 to 15
// Output: none
void DAC_Out(unsigned long data){
  GPIO_PORTE_DATA_R = data;
}
```

```
// Piano.c
// This software configures the off-board piano keys
// Runs on LM4F120 or TM4C123
// Program written by: put your names here
// Date Created: 8/25/2013
// Last Modified: 10/9/2013
// Section 1-2pm     TA: Saugata Bhattacharyya
// Lab number: 6
// Hardware connections

#include "tm4c123gh6pm.h"
// put code definitions for the software (actual C code)
// this file explains how the module works

// **************Piano_Init*********************
// Initialize piano key inputs, called once
// Input: none
// Output: none
void Piano_Init(void){unsigned long volatile delay;
  SYSCTL_RCGC2_R |= 0x01; // activate port A
  delay = SYSCTL_RCGC2_R;                   // 2 NOP
     GPIO_PORTA_AMSEL_R &= ~0x1C;           // no analog
  GPIO_PORTA_PCTL_R &= ~0x000FFF00;   // regular GPIO function
  GPIO_PORTA_DIR_R |= ~0x1C;      // PA4-2 are inputs
  GPIO_PORTA_AFSEL_R &= ~0x1C;   // disable alternate select function
on PA4-2
  GPIO_PORTA_DEN_R |= 0x1C;      // dig enable for PA4-2
}
// **************Piano_In*********************
// Input from piano key inputs
// Input: none
// Output: 0 to 7 depending on keys
// 0x01 is just Key0, 0x02 is just Key1, 0x04 is just Key2
unsigned long Piano_In(void){
      unsigned long volatile key;                              //define
variable 'key'
           key = GPIO_PORTA_DATA_R>>2;//read the input from switches

   return key;  // replace this line with actual code
}
```

```
// Sound.c,
// This module contains the SysTick ISR that plays sound
// Runs on LM4F120 or TM4C123
// Program written by: put your names here
// Date Created: 8/25/2013
// Last Modified: 10/9/2013
// Section 1-2pm     TA: Saugata Bhattacharyya
// Lab number: 6
// Hardware connections
#include "tm4c123gh6pm.h"
#include "dac.h"
#include "piano.h"
// put code definitions for the software (actual C code)
// this file explains how the module works

// **************Sound_Init*********************
// Initialize Systick periodic interrupts
// Input: Initial interrupt period
//           Units to be determined by YOU
//           Maximum to be determined by YOU
//           Minimum to be determined by YOU
// Output: none
const unsigned char SineWave[32] =
{8,9,11,12,13,14,14,15,15,15,14,14,13,12,11,9,8,7,5,4,3,2,2,1,1,1,2,2,
3,4,5,7};
static int Index=0;           // Index varies from 0 to 31
void Sound_Init(unsigned long period){unsigned volatile long delay;
  DAC_Init();          // Port E is DAC
  Index = 0;                                  // initialize the Index
pointer
     SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB; // activate port B
  delay = SYSCTL_RCGC2_R;                     // 2 NOP
  GPIO_PORTB_DIR_R |= 0x04;      // PB2 is output (heartbeat)
  GPIO_PORTB_AFSEL_R &= ~0x04;   // disable alternate select function
on PB2
  GPIO_PORTB_DEN_R |= 0x04;      // dig enable for PB2
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = period-1;// reload value
  NVIC_ST_CURRENT_R = 0;      // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF); // priority 0
  NVIC_ST_CTRL_R = 0x00000007; // enable SysTick with core clock and
interrupts
}

// **************Sound_Play*********************
// Start sound output, and set Systick interrupt period
// Input: interrupt period
//           Units to be determined by YOU
//           Maximum to be determined by YOU
//           Minimum to be determined by YOU
//         input of zero disables sound output
// Output: none
```

```
void Sound_Play (unsigned long volatile note){
      switch (note){
            case 0x01: NVIC_ST_RELOAD_R = 4778;
                  break;
            case 0x02: NVIC_ST_RELOAD_R = 4257;
                  break;
            case 0x04: NVIC_ST_RELOAD_R = 3792;
                  break;
            case 0x05: Song();
                  break;
            default: NVIC_ST_RELOAD_R = 0;
      }
}
void delay_note(void){unsigned long i; unsigned char x;
      for(i=0;i<2000000;i++){
            x=1;
      }
            GPIO_PORTF_DATA_R ^= 0x04; //toggle PF2 for heartbeat
}
void delay_rest(void){unsigned long i; unsigned char x;
      for(i=0;i<50000;i++){
            x=1;
      }
            GPIO_PORTF_DATA_R ^= 0x04; //toggle PF2 for heartbeat
}
int Song(void){unsigned long i; //play the Hobbit
      const unsigned long Song[98] =
{4778,4257,3792,3792,3792,3189,3189,3189,3792,3792,3792,4257,4257,4257
,4778,4778,

      4778,4778,4778,4778,4778,4778,4778,3792,3792,3189,3189,2841,2841,
2841,2841,2841,2389,2389,2531,2531,2531,3189,

      3189,3189,3792,3792,3792,3792,3792,3579,3792,4257,4257,4257,4257,
4257,4257,

      4778,4257,3792,3792,3792,3189,3189,3189,3792,3792,3792,4257,4257,
4257,4778,4257,4778,4778,4778,4778,4778,4778,

      4778,3792,3792,3189,3189,2841,2841,2841,2841,2841,2841,3189,3189,
3792,3792,3792,4257,4257,4257,4257,4257,4257,4257,};
      for(i=0;i<98;i++){
            NVIC_ST_RELOAD_R = Song[i];
            delay_note();
            if(
                  Piano_In()==1){                  //if switch one is
pressed, exit
                        return 0;}

      }
            return 0;
```

```
}


// Interrupt service routine
// Executed periodically, the actual period
// determined by the current Reload.
void SysTick_Handler(void){
      GPIO_PORTB_DATA_R ^= 0x04;       //toggle PB2 for heartbeat
  Index = (Index+1)&0x1F;               // increment the pointer to
the index
  DAC_Out(SineWave[Index]);    // output one value each interrupt
}
```