Hacettepe University
Computer Engineering Department
BBM434: Embedded Systems Laboratory
8th Lab Assignment Report

(2nd Section)

Şerafettin Berk SEVGİ
21328414

Cemal ÜNAL
21328538

07.05.2017

OBJECTIVE

In this assignment, it's requested to implement a basic piano interface with using Digital to Analog Conversion (DAC) principles on Tiva LaunchPad. The main idea is to program switches on interface as a piano octave (or less number of note).

THEORETICAL BACKGROUND FOR THE LAB

In pyhsics, **sound** is a vibration that propagates as a typically audible mechanical wave of pressure and displacement, through a transmission medium such as air or water. Sound exists as varying pressure waves that are created when a physical object moves, vibrating the air next to it. There is a need for materiality such as air, water, etc. to conduct a sound. The sound generated by the vibration of the sound source is also provided by the vibrations of the particles in the material environment.

Digital to Analog Converters (DAC) are required for the production of sound waves in electronic environment. With DAC, the sound waves produced in the electronic environment are propagated in the material environment through a speaker and the sound is produced. The electronic representation of the sound wave with harmonic characteristic (sinusoidal) with 4 bit DAC to be used in this paper is shown below.

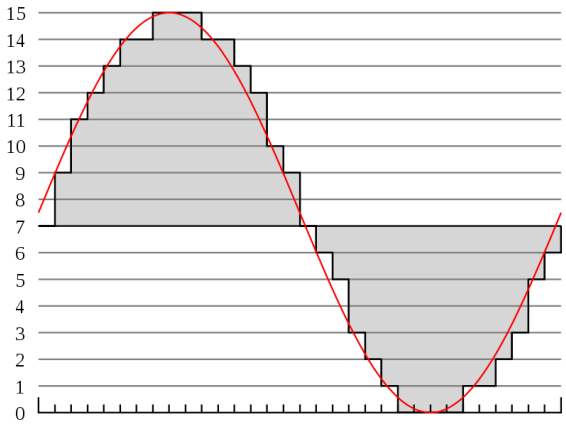


Figure 1. 4-bit representation of sine wave

According to the figure above, a sinusoidal wave can be obtained by taking the values 0 to 15 at different time intervals in the electronic environment.

PROCEDURE AND RESULTS

In a digital environment, an audio loop occurs with a sequence of 32 numbers ranging from 0 to 15 and a transmission speed to the output port. This cycle is performed using the SysTick Timer and allows the SysTick Timer to operate at different frequencies, which can be obtained in multiple note values.

In this assignment, 4 switch buttons are used and these 4 buttons are programmed to output Do, Re, Mi and Fa notes respectively. With an Android platform application named **gStrings**, the audio frequency corresponding to each button and the SysTick value that will generate this frequency are measured.

Note	SysTick Reload Value
Do	1900
Re	1693
Mi	1508
Fa	1426

Accordingly, to create a voice in a particular note, the 32 numeric values that make up the sound wave must change as often as the SysTick value corresponding to that note.

In implementation, the DAC circuit on the board modulates the voltage from port B using resistors 1.5, 3, 6 and 12 KΩ, giving the appropriate output current a speaker. The schematic of the DAC circuit used is shown below.

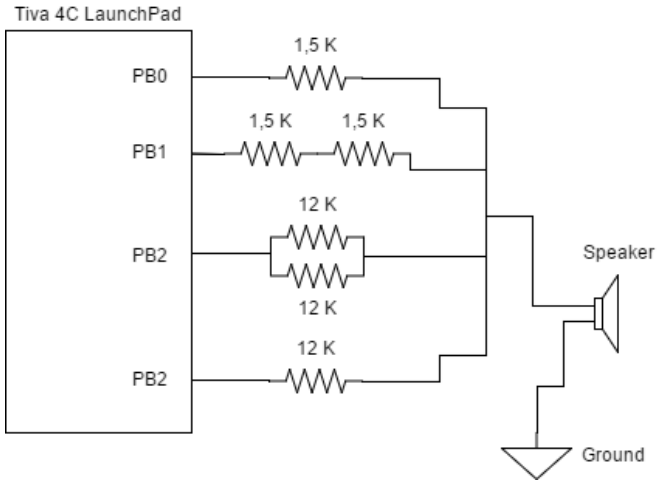
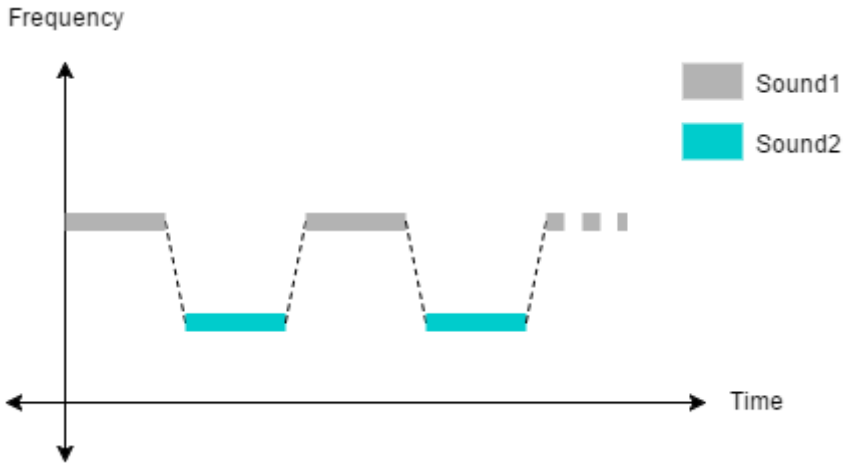


Figure 2. DAC circuit diagram

The reload value of the SysTick timer is manipulated with the switch buttons connected to the GPIO Port E. First button(PE0) is giving a sound that has the same frequency with Do. Other buttons also generates sounds with the same frequency Re, Mi, Fa respectively. At the beginning, there will be no sound until a switch button is pressed. After that if a switch button is pressed, then appropriate sound will be generated until the switch button is released.

Interaction of two sound waves

If two different notes are pressed in the Piano interface, the LaunchPad will play two different sounds in different time periods to produce a new sound. At the specified time intervals, the SysTick timer goes from the current note to the other note in a very short time, and the new note plays again in a very short time. Thus, a new smooth sound will be generated.



VIDEO OF LABORATORY ASSIGNMENT DEMO

- BBM432 - Lab.8

CODE

```
// ***** 1. Pre-processor Directives Section *****
#include "TExaS.h"
#include "tm4c123gh6pm.h"

void EnableInterrupts(void);
void DisableInterrupts(void);
void WaitForInterrupt(void);
```

```
/* SysTick variables with respect to specific note values*/
unsigned int Do = 1900;
unsigned int Re = 1693;
unsigned int Mi = 1508;
unsigned int Fa = 1426;

/*Sine wave represantation on digital environment*/
const unsigned char SineWave[32] =
{7,9,11,12,13,14,14,15,15,15,14,14,13,12,10,9,7,6,5,3,2,1,0,0,0,0,1,1,2,3,5,6};

unsigned char Index = 0;                    // Index varies from 0 to 15
// **************DAC_Init*********************
// Initialize 3-bit DAC
// Input: none
// Output: none
void DAC_Init(void){
    unsigned long volatile delay;
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOB;   // activate port B
    delay = SYSCTL_RCGC2_R;                 // allow time to finish activating
    GPIO_PORTB_AMSEL_R &= ~0x0F;            // no analog
    GPIO_PORTB_PCTL_R &= ~0x00000FFF;       // regular GPIO function
    GPIO_PORTB_DIR_R |= 0x0F;               // make PB3-0 out
    GPIO_PORTB_AFSEL_R &= ~0x0F;            // disable alt funct on PB3-0
    GPIO_PORTB_DEN_R |= 0x0F;               // enable digital I/O on PB3-0
}


// **************Sound_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 Sound_Init(){
    DAC_Init();                     // Port B is DAC
    Index = 0;
    NVIC_ST_CTRL_R = 0;             // disable SysTick during setup
    NVIC_ST_CURRENT_R = 0;          // any write to current clears it
    NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; // priority 2
}

void Switch_Init(){
    unsigned volatile long delay;
    SYSCTL_RCGC2_R |= 0x00000010;         // activate clock for port E
      delay = SYSCTL_RCGC2_R;             // initialize count and wait for clock
    GPIO_PORTE_LOCK_R = 0x4C4F434B;       // 2) unlock GPIO Port E
    GPIO_PORTE_CR_R = 0x0F;               // allow changes to PE0-3
    GPIO_PORTE_DIR_R &= ~0x0F;            // 5) PE0-3 in
    GPIO_PORTE_AFSEL_R &= ~0x0F;          //     disable alt funct on PE0-3
    GPIO_PORTE_DEN_R |= 0x0F;             //     enable digital I/O on PE0-3
    GPIO_PORTE_PCTL_R &= ~0x0000000F;     //  configure PE0 as GPIO
    GPIO_PORTE_AMSEL_R &= ~0x0F;          //disable analog functionality PE0-3
    GPIO_PORTE_PUR_R |= 0x00;             //     enable weak pull-up
    GPIO_PORTE_IS_R &= ~0x0F;             // (d) PE0-3 is edge-sensitive
    GPIO_PORTE_IBE_R |= 0x0F;             //     PE0-3 is both edges
    GPIO_PORTE_ICR_R = 0x0F;              // (e) clear flag0-3
    GPIO_PORTE_IM_R |= 0x0F;              // (f) arm interrupt on PE0-3
    NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF00)|0x00000020; // (g) priority 1
    NVIC_EN0_R = 0x00000010;              // (h) enable interrupt 4 in NVIC
```

```
}
/* Each counter indicates condition of when a specified button is pressed*/
int doc=0;      //Do counter
int rec=0;      //Re counter
int mic=0;      //Mi counter
int fac=0;      //Fa counter

/* Time interval variable*/
int interval=0;
// *************SuperPos********************
// Mixes 2 sound with different notes
// Based on context switching mechanism
// Input: SysTick Reload values of two notes
// Output: none
void SuperPos(int Note1, int Note2){

    if(interval == 0)       //Allows to play first note on first time interval
    {
        NVIC_ST_RELOAD_R = Note1 - 1;
    }
    else if(interval == 1) //Allows to play second note on second time interval
    {
        NVIC_ST_RELOAD_R = Note2 - 1;
    }
    interval = (interval + 1) % 2;                  //Turns back
}

void GPIOPortE_Handler(){

    if(GPIO_PORTE_DATA_R & 0x01)          //Do
    {
        NVIC_ST_RELOAD_R = Do-1;
        NVIC_ST_CTRL_R = 0x0007;
        doc++;
        if(GPIO_PORTE_DATA_R & 0x02)      //If Re is pressed when Do was pressed
            rec++;
        else if(GPIO_PORTE_DATA_R & 0x04) //If Mi is pressed when Do was pressed
            mic++;
        else if(GPIO_PORTE_DATA_R & 0x08) //If Fa is pressed when Do was pressed
            fac++;
        else{
            rec = 0;
            mic = 0;
            fac = 0;
        }
    }
    else if(GPIO_PORTE_DATA_R & 0x02){    //Re
        NVIC_ST_RELOAD_R = Re-1;
        NVIC_ST_CTRL_R = 0x0007;
        rec++;
        if(GPIO_PORTE_DATA_R & 0x01)      //If Do is pressed when Re was pressed
            doc++;
        else if(GPIO_PORTE_DATA_R & 0x04) //If Mi is pressed when Re was pressed
            mic++;
        else if(GPIO_PORTE_DATA_R & 0x08) //If Fa is pressed when Re was pressed
            fac++;
        else{
            doc = 0;
            mic = 0;
            fac = 0;
        }
```

```
    }
    else if(GPIO_PORTE_DATA_R & 0x04){    //Mi
        NVIC_ST_RELOAD_R = Mi-1;
        NVIC_ST_CTRL_R = 0x0007;
        mic++;
        if(GPIO_PORTE_DATA_R & 0x01)      //If Do is pressed when Mi was pressed
            doc++;
        else if(GPIO_PORTE_DATA_R & 0x02) //If Re is pressed when Mi was pressed
            rec++;
        else if(GPIO_PORTE_DATA_R & 0x08) //If Fa is pressed when Mi was pressed
            fac++;
        else{
            doc = 0;
            rec = 0;
            fac = 0;
        }
    }
    else if(GPIO_PORTE_DATA_R & 0x08){    //Fa
        NVIC_ST_RELOAD_R = Fa-1;
        NVIC_ST_CTRL_R = 0x0007;
        fac++;
        if(GPIO_PORTE_DATA_R & 0x01)      //If Do is pressed when Fa was pressed
            doc++;
        else if(GPIO_PORTE_DATA_R & 0x02) //If Re is pressed when Fa was pressed
            rec++;
        else if(GPIO_PORTE_DATA_R & 0x04) //If Mi is pressed when Fa was pressed
            mic++;
        else{
            doc = 0;
            rec = 0;
            mic = 0;
        }
    }
    else                                  //When no button is pressed
    {
        NVIC_ST_CTRL_R = 0x00;
        doc =0;
        rec =0;
        mic =0;
        fac =0;
    }

    GPIO_PORTE_ICR_R = 0x0F;              // ack, clear interrupt flag0-3
}

// the sound frequency will be (interrupt frequency)/(size of the table)
void SysTick_Handler(void){
    Index = (Index+1)&0x1F;  // 7,9,11,12,13,14,14,15,15,...
    GPIO_PORTB_DATA_R = SineWave[Index];
    if(doc > 0 && rec > 0 )
        SuperPos(Do,Re);
    if(rec > 0 && mic > 0)
        SuperPos(Re,Mi);
    if(mic > 0 && fac > 0)
        SuperPos(Mi,Fa);
    if(doc > 0 && mic > 0)
        SuperPos(Do,Mi);
    if(doc > 0 && fac > 0)
        SuperPos(Do,Fa);
    if(rec > 0 && fac > 0)
        SuperPos(Re,Fa);
```

```
}

int main(void){                                 // running at 16 MHz

    Sound_Init();                                // initialize SysTick timer
    Switch_Init();

    while(1){
        WaitForInterrupt();
    }

}
```