Xuecong Fan A01972388
Robert McKay A02080393
November 14, 2017

Lab 6

Introduction:

For this lab mainly we are used 'tiva C series development board, 'analog test board module' and 'MCP4725 12-bit DAC' for our lab equipment. The purposed of this lab is use DAC to show different frequency and waveform. Ues spin button to control different wave on the analog test board module to test and listen different sound for different frequency. (Figure 1 and Figure 2 shows the 'analog test board module' and 'MCP4725 12-bit DAC' we used in this lab.)

Figure 1:"analog test board module"

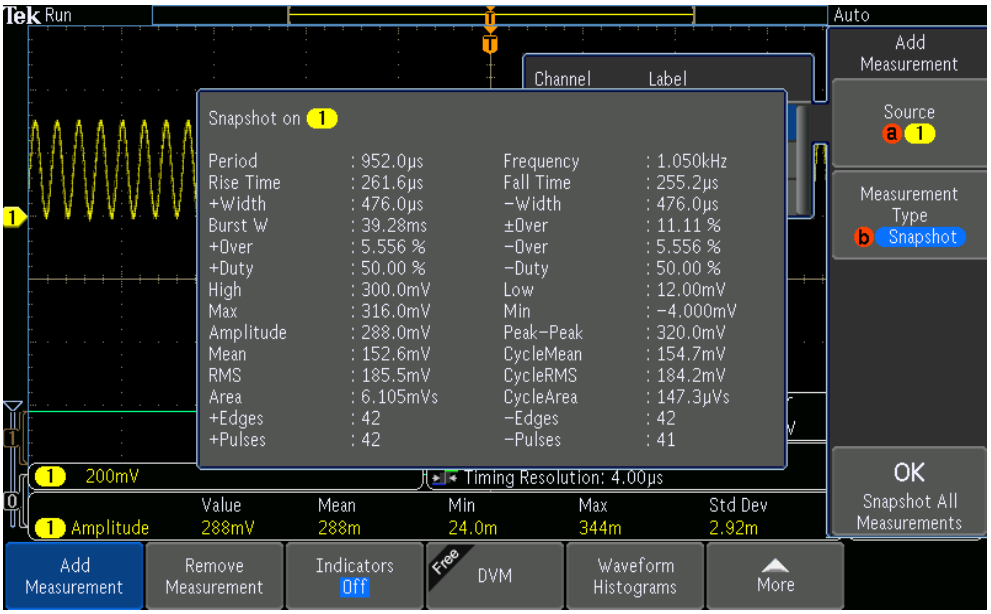
Figure 2:"MCP4725 12-bit DAC'"

Lab Detail:

First in this lab use DAC to output the external 16 MHz crystal with the PLL function to find most convenient frequency for the system clock. Then connect DAC to analog test board module to test the sound we code. Now we need configure the ADC in every 2 ms with an interrupt. It would produce a sine wave now. Next, used the interrupt and wave we found above to compute and let the spin button work for the range from 100Hz to 1000Hz.

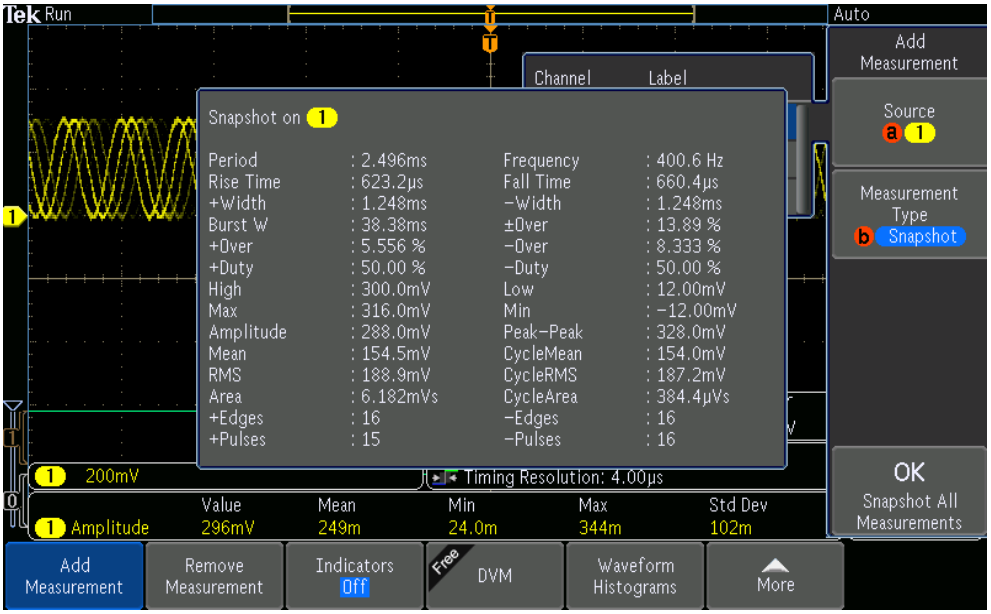
Lab data:
In this lab we used 3.3 voltage for the DAC from the 'tiva Cseries development board'.For A0 pin we used 0 and for I^2C address of the DAC we use '1100010'.
Correct frequencies are produced for the following codes:

For 0:



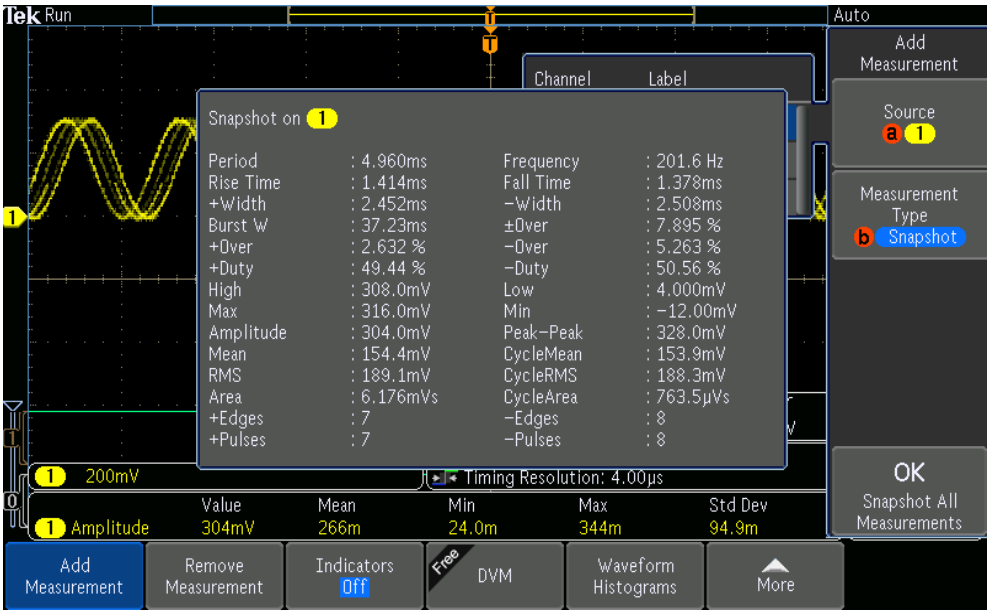
Picture:1

For 1023:



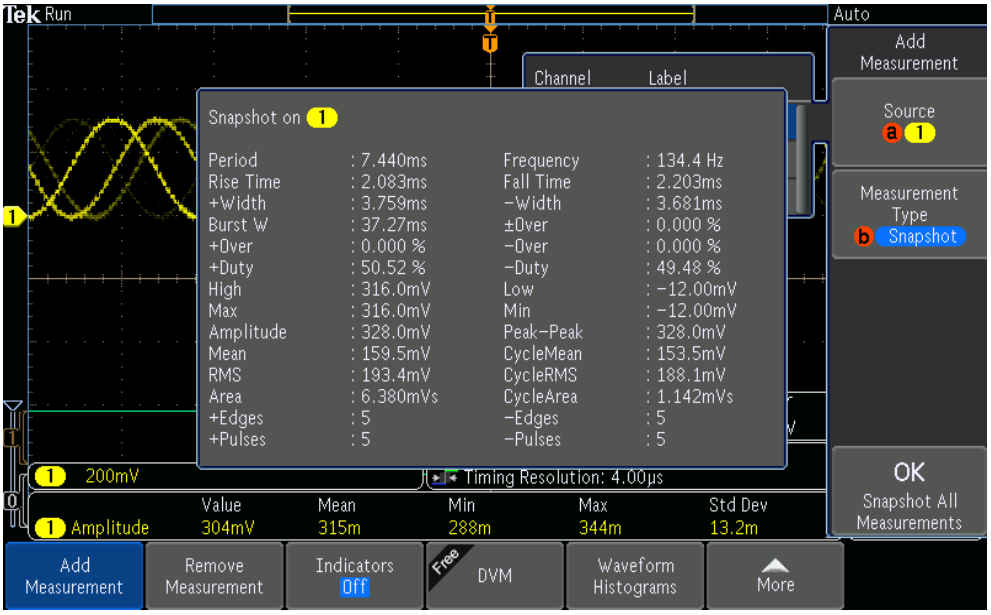
Picture:2

For 2047:



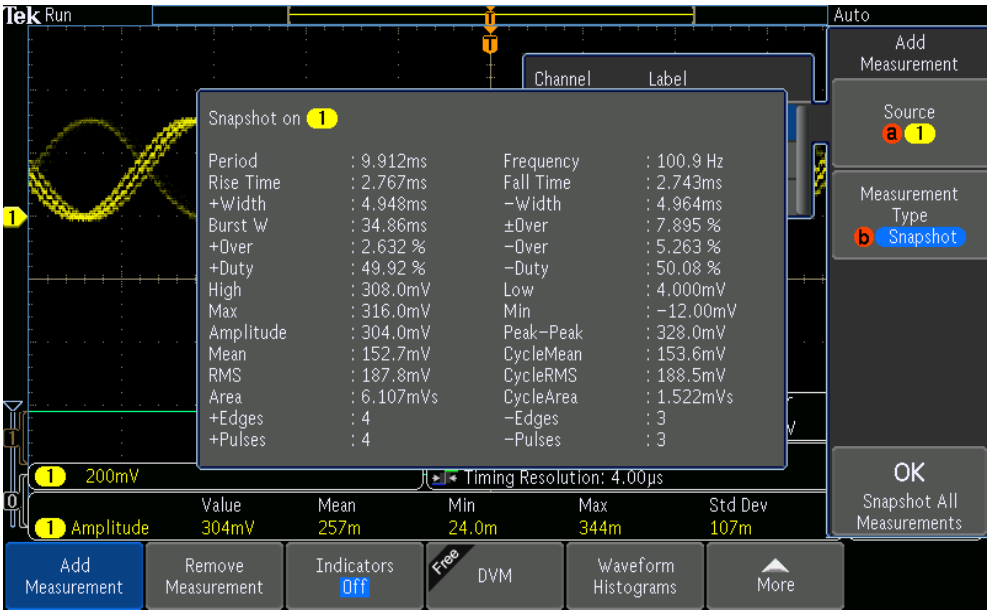
Picture:3

For 3017:



Picture:4

For 4095:



Picture:5

Software Design:

```c
#include <stdio.h>
#include "TM4C123GH6PM.h"
#define SYSDIV2 4
#include <math.h>
#define PI 3.14159265358979323846
unsigned int* systemClock = (unsigned int*) 0x400FE000;
unsigned int* portA = (unsigned int*) 0x40004000;
unsigned int* portB = (unsigned int*) 0x40005000;
unsigned int* portE = (unsigned int*) 0x40024000;
unsigned int* i2c0 = (unsigned int*) 0x40020000;
unsigned int* adc0 = (unsigned int*) 0x40038000;
unsigned int* timer0 = (unsigned int*) 0x40030000;
unsigned int* timer1 = (unsigned int*) 0x40031000;
unsigned int* timer2 = (unsigned int*) 0x40032000;
unsigned int* corePeripheral = (unsigned int*) 0xE000E000;
unsigned int i = 0;
double doublevalue;
unsigned int value;
unsigned int adc_value;
unsigned int adc_count;
unsigned int table_of_values[40] = {
    2047,
    2367,
    2680,
    2977,
    3250,
    3495,
    3703,
    3871,
    3994,
    4069,
    4095,
    4069,
    3994,
    3871,
    3703,
    3495,
    3250,
    2977,
    2680,
    2367,
    2047,
    1727,
    1414,
    1117,
    844,
    599,
    391,
    223,
    100,
    25,
    0,
    25,
    100,
    223,
    391,
    599,
    844,
    1117,
    1414,
    1727};

 // configure the system to get its clock from the PLL
void getSin(void) {
//  doublevalue = (1.65 + 1.65*sin(9*i*PI/180))/3.3;
 //  value = doublevalue;
 //  value = value << 12;
    value = table_of_values[i];
    i++;
    if (i == 40) {
        i = 0;
    }
}
```

Figure 3:"Code1"

```
void PLL_Init(void){
  // 0) use RCC2
  SYSCTL_RCC2_R |= SYSCTL_RCC2_USERCC2;
  // 1) bypass PLL while initializing
  SYSCTL_RCC2_R |= SYSCTL_RCC2_BYPASS2;
  // 2) select the crystal value and oscillator source
  SYSCTL_RCC_R = (SYSCTL_RCC_R & ~0x000007C0) //CLEAR BITS 10-6
                                 + 0x00000540; // 10101, configure for 16 MHz crystal
    SYSCTL_RCC2_R &= ~0x00000070; // configure for main oscillator source
  // 3) activate PLL by clearing PWRDN
  SYSCTL_RCC2_R &= ~SYSCTL_RCC2_PWRDN2;
  // 4) set the desired system divider and the system divider least significant bit
  SYSCTL_RCC2_R |= SYSCTL_RCC2_DIV400;  // use 400 MHz PLL
  SYSCTL_RCC2_R = (SYSCTL_RCC2_R&~0x1FC00000) // clear system clock divider field
                + (SYSDIV2<<22);      // configure for 80 MHz clock
  // 5) wait for the PLL to lock by polling PLLRIS
  while((SYSCTL_RIS_R&SYSCTL_RIS_PLLLRIS)==0){};
  // 6) enable use of PLL by clearing BYPASS
  SYSCTL_RCC2_R &= ~SYSCTL_RCC2_BYPASS2;
}
void transmit(void) {
    i2c0[0x0/4] = 0xC4; // writes 1100 0100, which is 1100 010 with an added 0 trailing.
    getSin();
    i2c0[0x8/4] = value >> 8; // a byte to be transmitted
    i2c0[0x4/4] = 0x3; // stop, start, run stuff
    while ((i2c0[0x4/4] & 0x1) == 0x1);
    while ((i2c0[0x4/4] & 0x2) == 0x2);
    i2c0[0x0/4] = 0xC4; // writes 1100 0100, which is 1100 010 with an added 0 trailing.
    i2c0[0x8/4] = value; // a byte to be transmitted
    i2c0[0x4/4] = 0x5; // stop, start, run stuff
    while ((i2c0[0x4/4] & 0x1) == 0x1);
    while ((i2c0[0x4/4] & 0x2) == 0x2);
}
void ADC0_InitSWTriggerSeq3_Ch9(void){
    SYSCTL_RCGCGPIO_R |= 0x10;  // activate clock for port E
    while((SYSCTL_PRGPIO_R&0x10) == 0){

    };
    GPIO_PORTE_DIR_R &= ~0x10;   // make PE4 input
    GPIO_PORTE_AFSEL_R |=0x10;  //enable alternate function on PE4
    GPIO_PORTA_DEN_R &= ~0x10; //disable digital I/O on PE4
    GPIO_PORTA_AMSEL_R |= 0x10; //enable analog function on PE4
    SYSCTL_RCGCADC_R |= 0x01;
    ADC0_PC_R = 0x01;
    ADC0_SSPRI_R = 0x0123;
    ADC0_ACTSS_R &= ~0x0008;
    ADC0_EMUX_R &= ~0xF000;
    ADC0_SSMUX3_R = (ADC0_SSMUX3_R&0xFFFFFFF0) + 9;
    ADC0_SSCTL3_R = 0x0006;
    ADC0_IM_R &= ~0x0008;
    ADC0_ACTSS_R |= 0x0008;
}
int inseq3(void){
    int 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_RIS_R  = 0x0008; // acknowledge completion
    return result;
}
void TIMER0A_Handler(void) {
    timer0[0x24/4] = 0x1; // clear interrupt
    adc_value += inseq3();
    adc_count++;
}
void TIMER1A_Handler(void) {
    timer1[0x24/4] = 0x1; // clear interrupt
    adc_value /= adc_count;
    timer2[0x28/4] = (adc_value*4+adc_value/2+adc_value/3 + 1);
    adc_count = 0;
    adc_value = 0;
}
```

Figure 4:"Code2"

```
146  void TIMER2A_Handler(void) {
147      transmit();
148      timer2[0x24/4] = 0x1;
149  }
150  void timer0init(void) {
151      timer0[0xC/4] = 0x0;
152      timer0[0x0/4] = 0x0;
153      timer0[0x4/4] = 0x2;
154      timer0[0x28/4] = 0x270FF;
155      timer0[0x38/4] = 0x0;
156      timer0[0x24/4] = 0x1;
157      timer0[0x18/4] = 0x1;
158      corePeripheral[0x100/4] |= 0x800000;
159  }
160  void timer1init(void) {
161      timer1[0xC/4] = 0x0;
162      timer1[0x0/4] = 0x0;
163      timer1[0x4/4] = 0x2;
164      timer1[0x28/4] = 0x26259FF;
165      timer1[0x38/4] = 0x0;
166      timer1[0x24/4] = 0x1;
167      timer1[0x18/4] = 0x1;
168      corePeripheral[0x100/4] |= 0x200000;
169  }
170  void timer2init(void) {
171      timer2[0xC/4] = 0x0;
172      timer2[0x0/4] = 0x0;
173      timer2[0x4/4] = 0x2;
174      timer2[0x38/4] = 0x0;
175      timer2[0x24/4] = 0x1;
176      timer2[0x18/4] = 0x1;
177      corePeripheral[0x100/4] |= 0x80000;
178  }
179  int main(void)
180  {
181      PLL_Init();
182      systemClock[0x608/4] = 0x13; // Enable GPIOA, B, and E in clock
183      systemClock[0x620/4] = 0x1; // Enable I2C0
184      systemClock[0x604/4] = 0x7; // Enable GPTM 0-2
185      timer0init();
186      timer1init();
187      timer2init();
188      //corePeripheral[0x100/4] |= 0xA8000;
189      corePeripheral[0x410/4] |= 0x60000000;
190      corePeripheral[0x414/4] |= 0x40002000;
191      ADC0_InitSWTriggerSeq3_Ch9();
192      portB[0x400/4] = 0xFF; // Direction register for port B
193      portB[0x420/4] |= 0xC; // 1100, enable alternate function I2C0  on PB2 and PB3
194      portB[0x50C/4] |= 0x8; // PB3 open drain
195      portB[0x52C/4] = 0x3300; //control register configuration
196      i2c0[0x20/4] |= 0x00000010; //master/slave config
197      i2c0[0xC/4] |= 0x00000001; // TPR = 1
198
199      portB[0x51C/4] = 0xFF;
200      timer2[0xC/4] = 0x1;
201      timer1[0xC/4] = 0x1;
202      timer0[0xC/4] = 0x1;
203      while(1) {
204
205      }
206  }
207
```

Figure 5:"Code2"

Issue:
In this lab the biggest issue is to found the different timer for the code. For our code we always find the first timer but can't found second and third timer.

Conclusions:
From this lab we studied how to set up DAC and control different frequency used spin button on analog test board. We also learned how to find different frequency based on our code.