SPRING 2017 PROJECT1_ DAC

MICROCONTROLLER III

By Julie Kim [redacted]

CECS 447 Monday, Wednesday 10:00am – 12:15 pm

Due Date: February 26, 2018

Instructor: John Vu

FEBRUARY 26, 2018

California State University Long Beach

Table of Contents

Introduction:

DAC project is to generate the digital value from TM4C123 microcontroller to control the output of different voltage in the purpose of generating a sound wave in a sine wave form. The component used are microcontroller by using the GPIO pin, Systick timer, and interrupts to convert the digital output from TM4C123 to analog output as voltage.

Operation:

A network of resistors in the form of binary weighted or R2R ladder schematic is built (DAC) to output the voltage according to the digital output value of the microcontroller. The component needed are: 4x4 Matrix keypad (8 pin output type), LM386 amplifier (IC only, not a kit), 0.25W - 1W speaker without built-in amplifier, 50 x 1.5K ohms 1/4W 1% tolerance resistors.

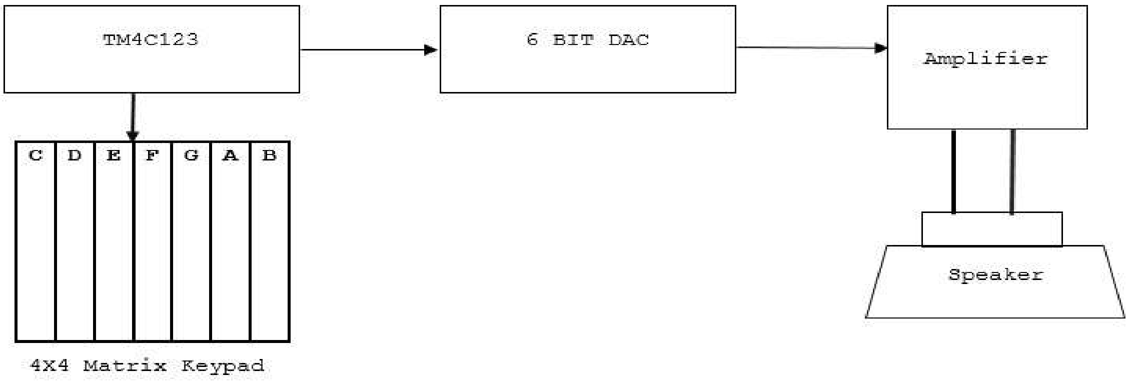
6 bit DAC is built by connecting to the six pins of PA2 to PA7. This project uses the R2R resister network of 1.5Khom resistor with 1% tolerance. The R2R ladder hardware is used because it can be constructed with only one value of resistor, while the binary weighted need different value of resistors of 2 to the power of 2. The LSB bit connects to the DAC output next to the ground pin, and the next pin PA3 connect to the adjacent pin output, in the order of LSB to MSB. The output from the DAC is connected to a potentiometer to adjust the voltage to the amplifier to control the volume to the speaker. There are two 1 micro ferrate capacitors, 1 50 micro ferrate capacitors and two 100 micro ferrate capacitors connected between the output of the voltage from the amplifier (LM386) and the speaker to generate smoother sound to the speaker. 4x4 matrix Keypad is used to interface between the control pin (PD0 to PD3, and PC4 to PC7) to the output of each sinewave frequency. There are 16 possible control keys and it uses only 8 pins of the microcontroller to interface it. 1W speaker is connected to the output from the amplifier. This way the project is building a speaker note from ready made kit but as an IC circuit of an amplifier, capacitors and speaker to generate sounds.

To generate triangle, square, and sine wave, SW1(push button from PF4) is pushed and release to change the mode of the waveform. Each mode is incremented each time SW1 is pushed. Mode 1, 2, and three control the above output. Mode 4 generate the 4x4 Matric Keypad interface. It generate sinewave with different frequency for each key, ‘1’ for Note ‘C’ with frequency of 262Hz, ‘2’ Note ‘d’ with frequency of 294Hz, ‘3’ Note ‘e’ with frequency of 330Hz, ‘4’ Note ‘f’ with frequency of 349Hz, ‘5’ Note ‘g’ with frequency of 392Hz, ‘6’ Note ‘a’ with frequency of 440Hz, and ‘7’ Note ‘b’ with frequency of 494Hz. Mode generate a short Music.

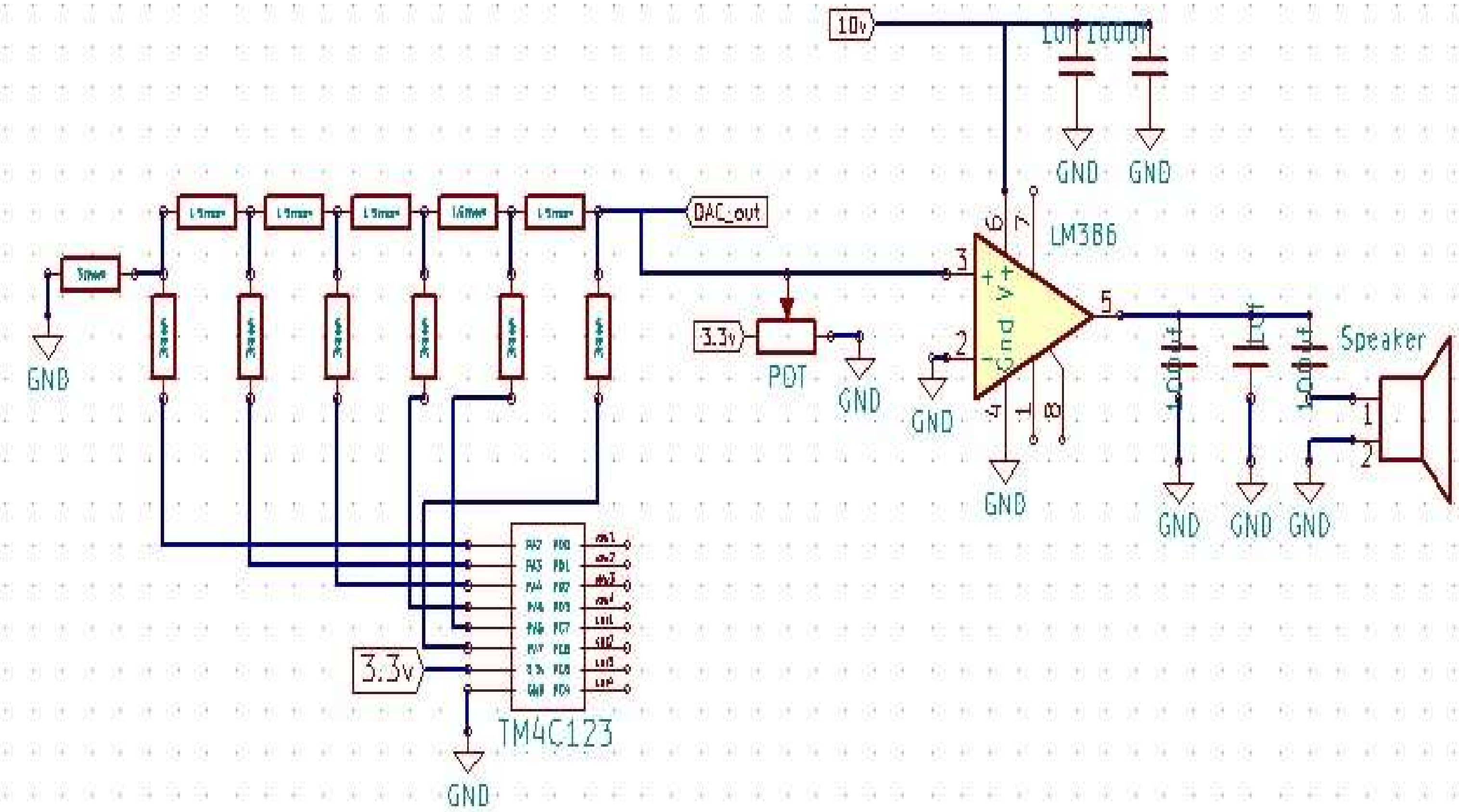
Hardware:

Hardware:

- ❖ Hardware Block Diagrams



- ❖ **Schematics**



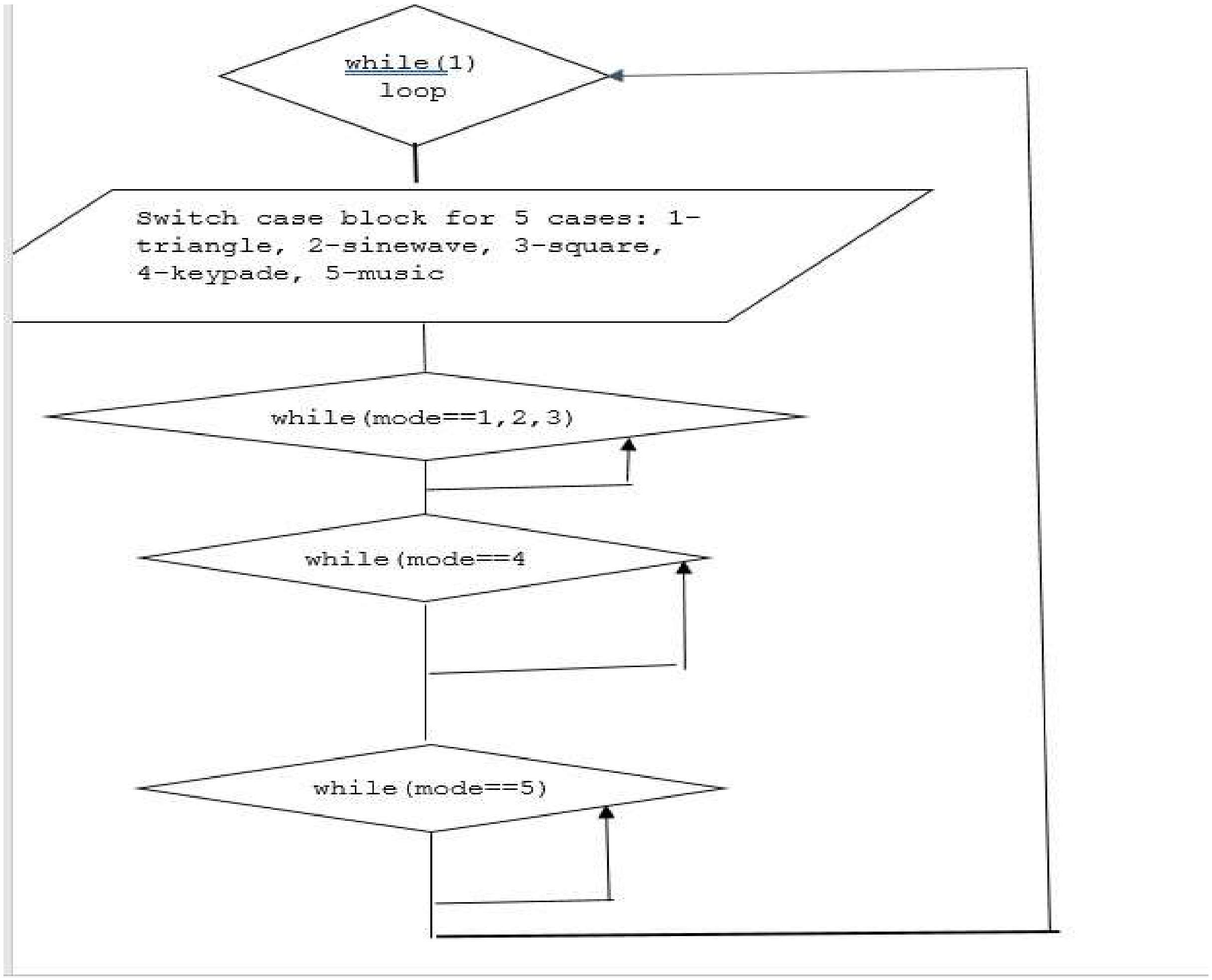
Software:

- **Software Approach:**
  There are two software device drivers to interface between the TM4C123 and the output from the network of the DAC schematic network. The first software device driver is to initialize the 6-bit port A (2 to 7-pin) of the microcontroller. Those pins are connected to the output from the DAC (LSB to MSB). Logic 1 of the pins send out 3.3v and logic 0 send out 0v. The varying of the pins output generate different voltage as each digital value generated by the software. The second software driver is to generate the digital output from the each pin of the microcontroller. As logic '1' gives 3.3v, and logic '0' gives 0v.

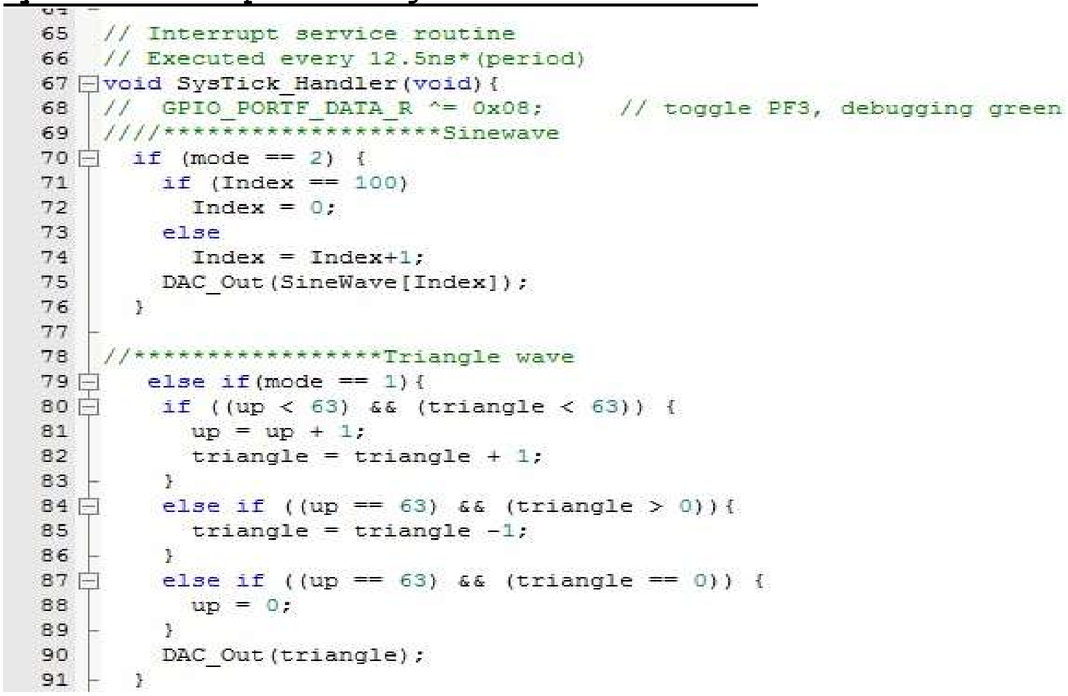
  Systick Timer is used to controller the period of how long each digital value to be output for before outputting the next digital value from the TM4C123 to generate the next voltage. The combination of digital value can generate triangle wave, square wave or sinewave. For this project, sinewave is generated to output the sound to the speaker to generate music based on the frequency generated. The formula to generate each digital value is: 1.65*SIN(((2*PI())/100)*F87) + 1.65. The Systick timer generate each different frequency based on the period of each digital value generated for. If the counting of the Systick is more, longer period, more counts of Systick timer, of each digital value and the frequency is slow. For example, the 262Hz note require the counting of Systick of 3023 while the Note 'B' of 494 has higher frequency require smaller period of each digital value and the counting value is only 1800. Therefore, each Note has different frequency of sinewave to generate the music.
  Interrupt and polling are used to controller the switch mode of each sinevwave frequency. The concept of negative logic is applied to controlled the switches (PF4) and the 4x4 Matric Keypad (PD0 to PD3 for row pins and PC4 to PC7 for column pins). When pressing on the PF4 and Keypad, the nine pins generate logic '0' to control the change of sinewave. Edge trigger interrupt changes the mode between triangle wave, sinewave and square wave. To output different frequency or Notes of 'A', 'B', 'C', 'D', 'F', 'G', polling the negative logic of the Keypad.

  To generate a short music, a short delay, about 1.5 second between each Note, is generated. A short music of "GEGGEGGEAGGE FFDDFFDDGFEDECC" is generated by having each Note plays for 1.5s.

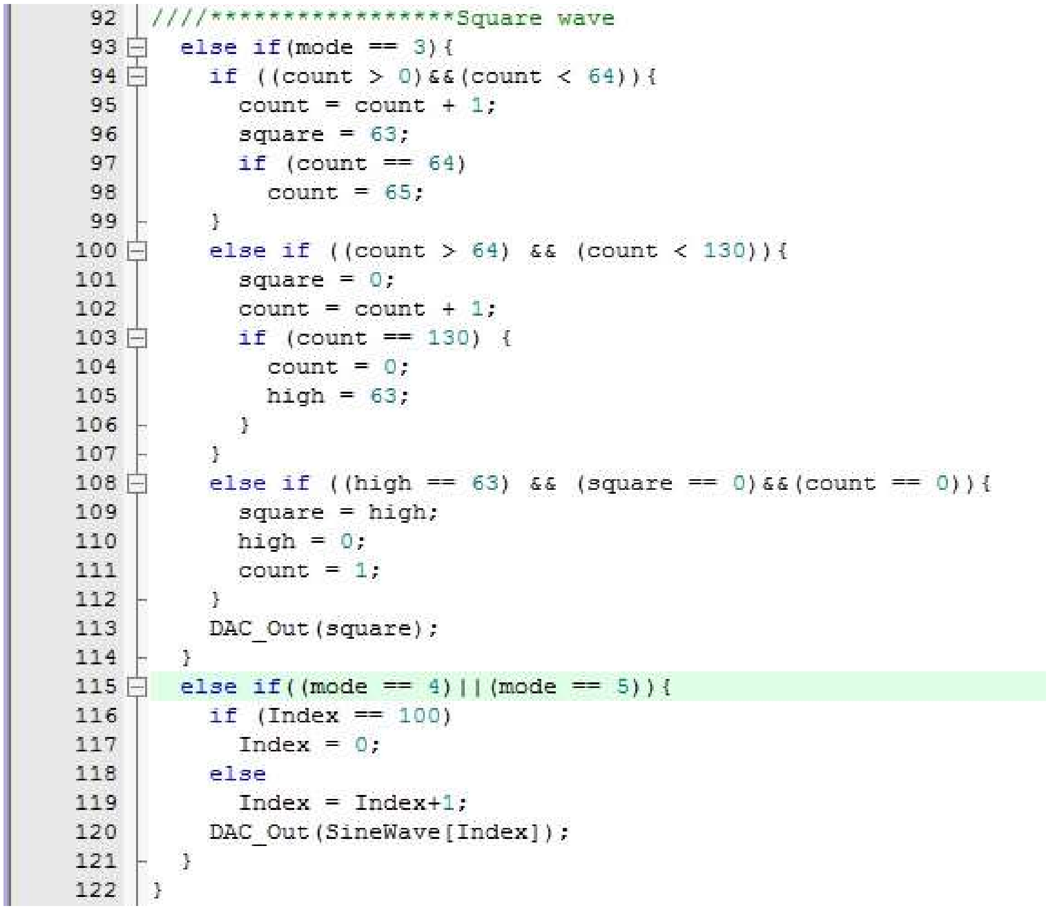
- **Software Diagram**


while(1)
loop
Switch case block for 5 cases: 1-triangle, 2-sinewave, 3-square, 4-keypade, 5-music
while(mode==1,2,3)
while(mode==4
while(mode==5)

Systick interrupt Handling of Sound Generation:

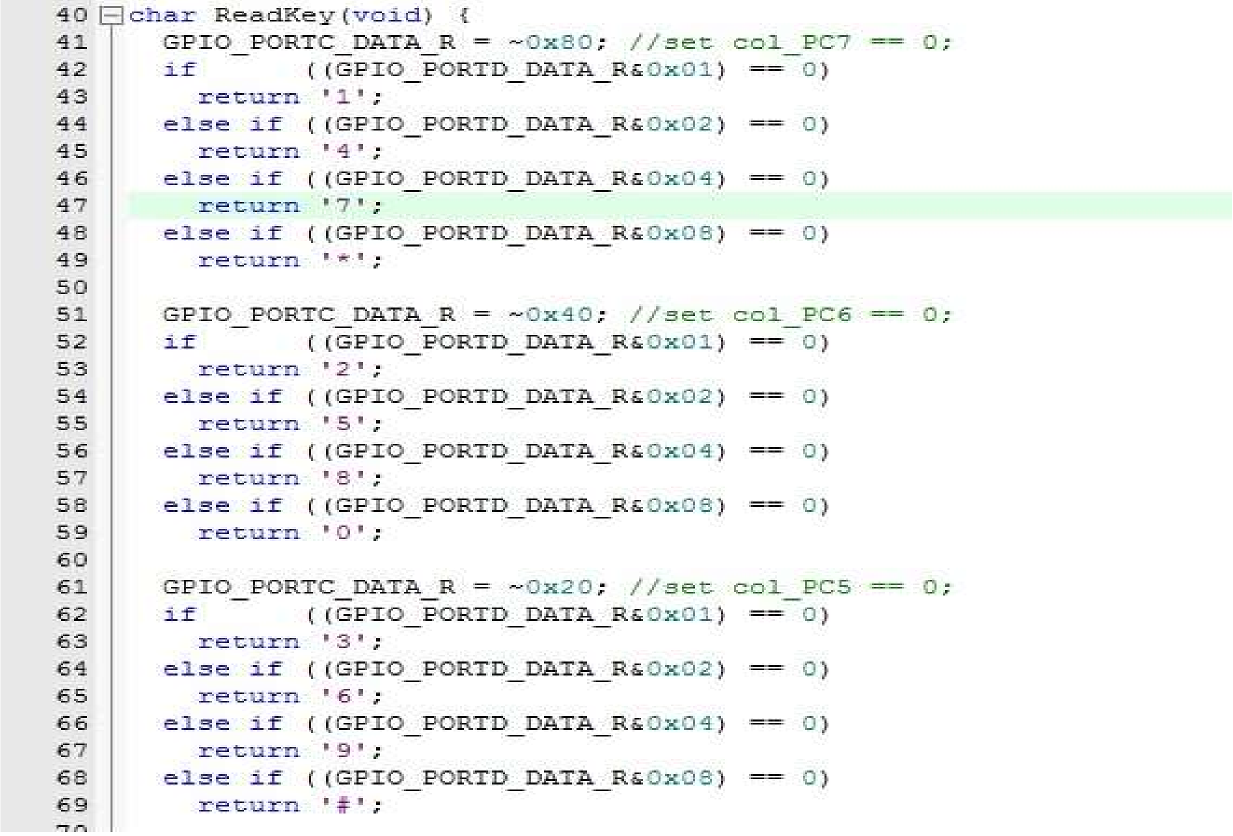


```
// Interrupt service routine
// Executed every 12.5ns*(period)
void SysTick_Handler(void){
//  GPIO_PORTF_DATA_R ^= 0x08;     // toggle PF3, debugging green
////******************Sinewave
  if (mode == 2) {
    if (Index == 100)
      Index = 0;
    else
      Index = Index+1;
    DAC_Out(SineWave[Index]);
  }

//*****************Triangle wave
   else if(mode == 1){
    if ((up < 63) && (triangle < 63)) {
      up = up + 1;
      triangle = triangle + 1;
    }
    else if ((up == 63) && (triangle > 0)){
      triangle = triangle -1;
    }
    else if ((up == 63) && (triangle == 0)) {
      up = 0;
    }
    DAC_Out(triangle);
  }
```

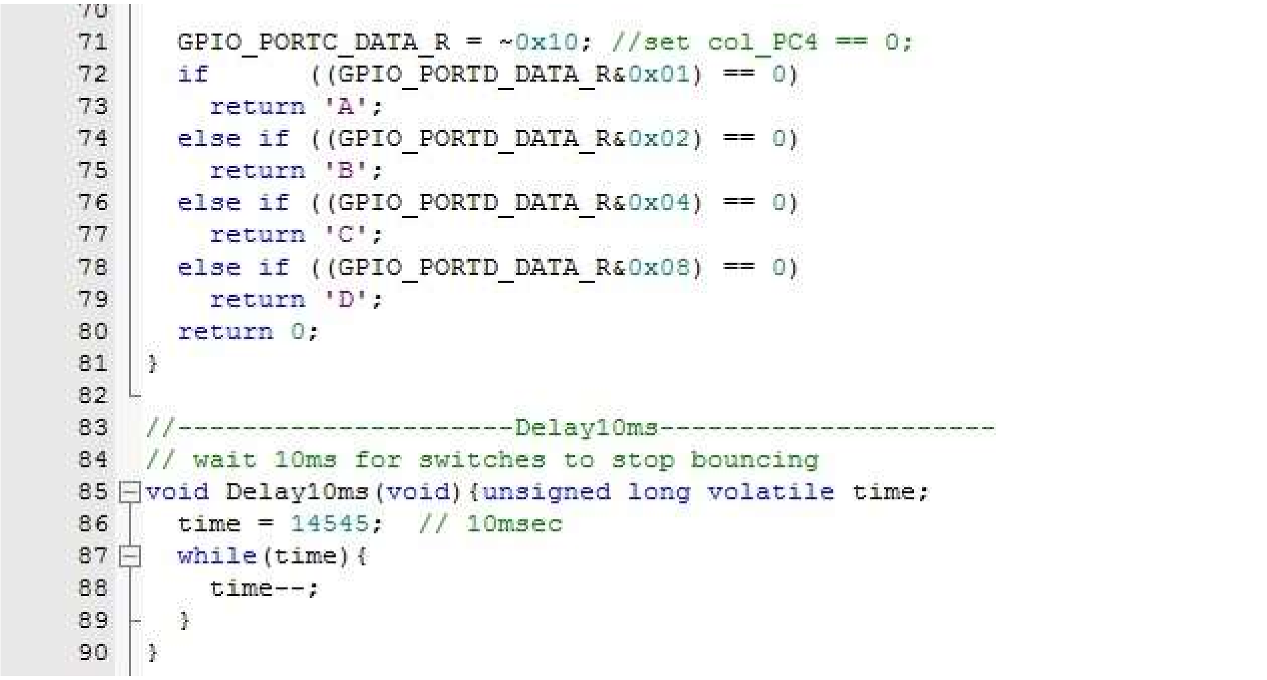


```
////*****************Square wave
  else if(mode == 3){
    if ((count > 0)&&(count < 64)){
      count = count + 1;
      square = 63;
      if (count == 64)
        count = 65;
    }
    else if ((count > 64) && (count < 130)){
      square = 0;
      count = count + 1;
      if (count == 130) {
        count = 0;
        high = 63;
      }
    }
    else if ((high == 63) && (square == 0)&&(count == 0)){
      square = high;
      high = 0;
      count = 1;
    }
    DAC_Out(square);
  }
  else if((mode == 4)||(mode == 5)){
    if (Index == 100)
      Index = 0;
    else
      Index = Index+1;
    DAC_Out(SineWave[Index]);
  }
}
```

Interfacing with 4x4 Matrix Keypad:

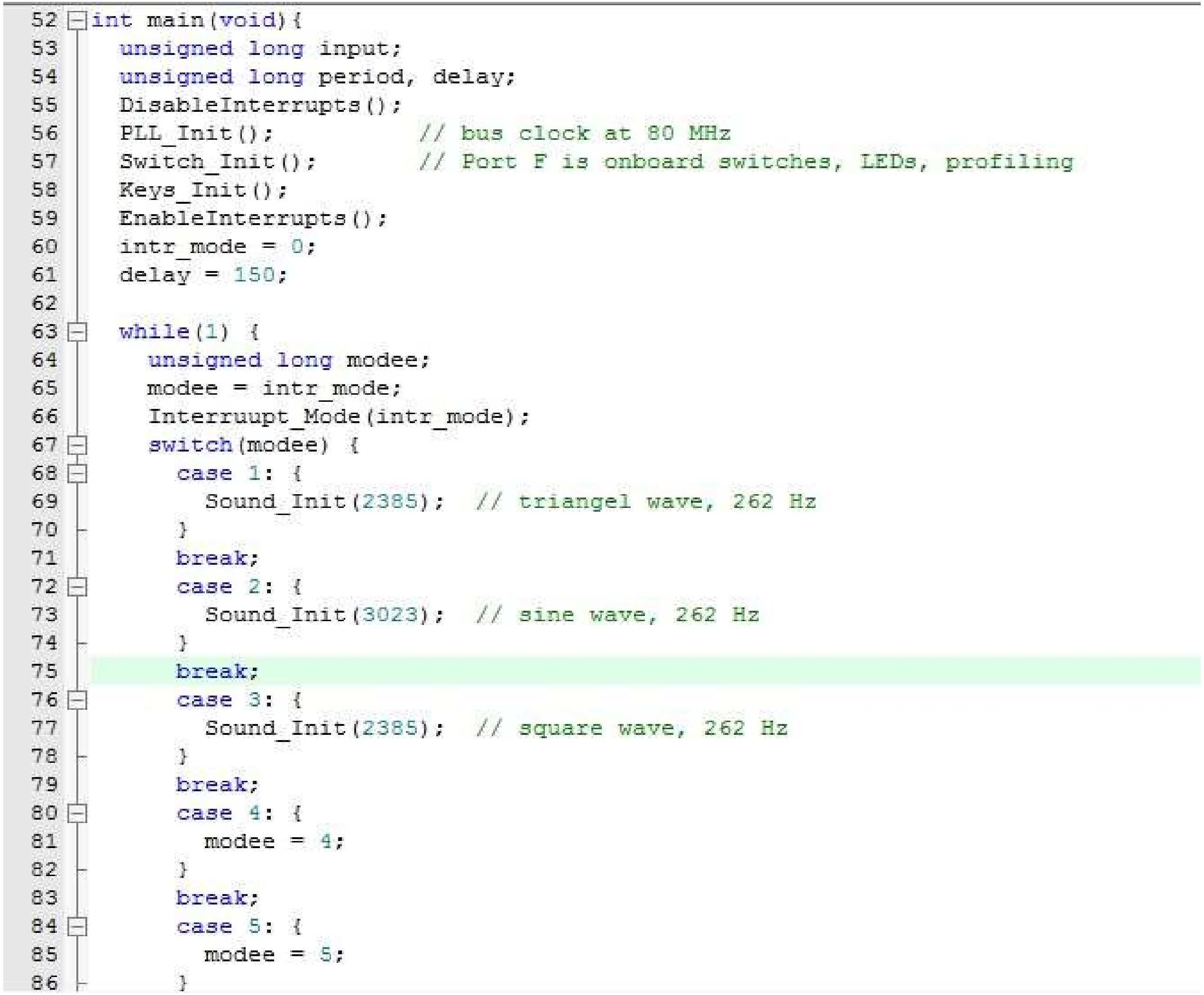


```
char ReadKey(void) {
  GPIO_PORTC_DATA_R = ~0x80; //set col_PC7 == 0;
  if      ((GPIO_PORTD_DATA_R&0x01) == 0)
    return '1';
  else if ((GPIO_PORTD_DATA_R&0x02) == 0)
    return '4';
  else if ((GPIO_PORTD_DATA_R&0x04) == 0)
    return '7';
  else if ((GPIO_PORTD_DATA_R&0x08) == 0)
    return '*';

  GPIO_PORTC_DATA_R = ~0x40; //set col_PC6 == 0;
  if      ((GPIO_PORTD_DATA_R&0x01) == 0)
    return '2';
  else if ((GPIO_PORTD_DATA_R&0x02) == 0)
    return '5';
  else if ((GPIO_PORTD_DATA_R&0x04) == 0)
    return '8';
  else if ((GPIO_PORTD_DATA_R&0x08) == 0)
    return '0';

  GPIO_PORTC_DATA_R = ~0x20; //set col_PC5 == 0;
  if      ((GPIO_PORTD_DATA_R&0x01) == 0)
    return '3';
  else if ((GPIO_PORTD_DATA_R&0x02) == 0)
    return '6';
  else if ((GPIO_PORTD_DATA_R&0x04) == 0)
    return '9';
  else if ((GPIO_PORTD_DATA_R&0x08) == 0)
    return '#';
```

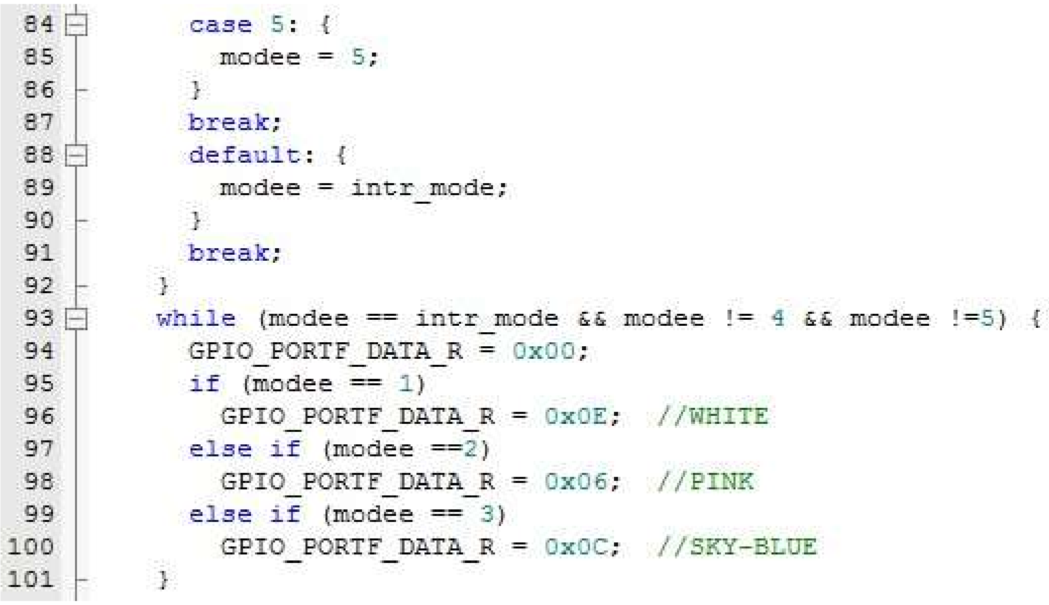


```
  GPIO_PORTC_DATA_R = ~0x10; //set col_PC4 == 0;
  if      ((GPIO_PORTD_DATA_R&0x01) == 0)
    return 'A';
  else if ((GPIO_PORTD_DATA_R&0x02) == 0)
    return 'B';
  else if ((GPIO_PORTD_DATA_R&0x04) == 0)
    return 'C';
  else if ((GPIO_PORTD_DATA_R&0x08) == 0)
    return 'D';
  return 0;
}

//----------------------Delay10ms----------------------
// wait 10ms for switches to stop bouncing
void Delay10ms(void){unsigned long volatile time;
  time = 14545;  // 10msec
  while(time){
    time--;
  }
}
```

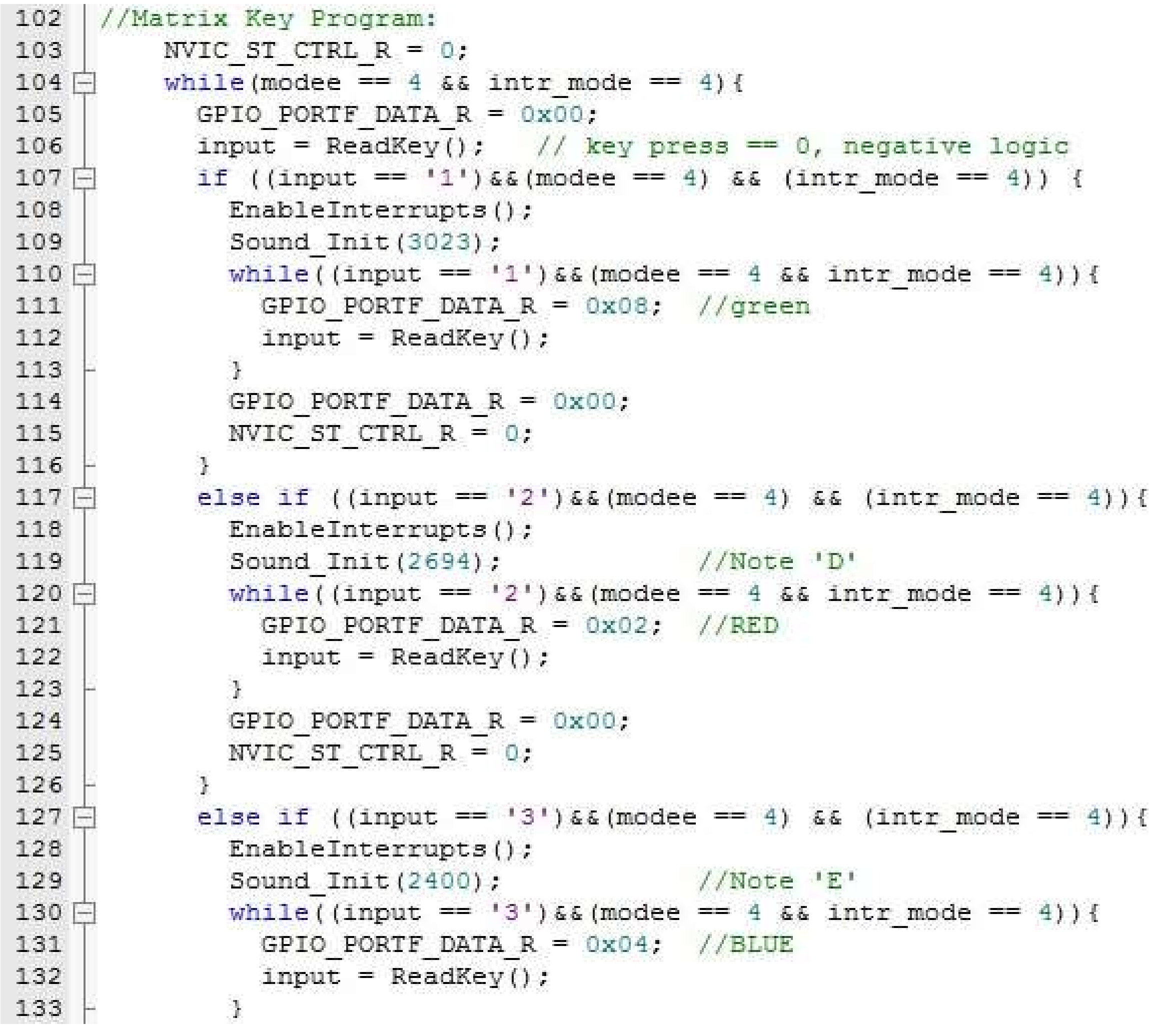
Main Program:



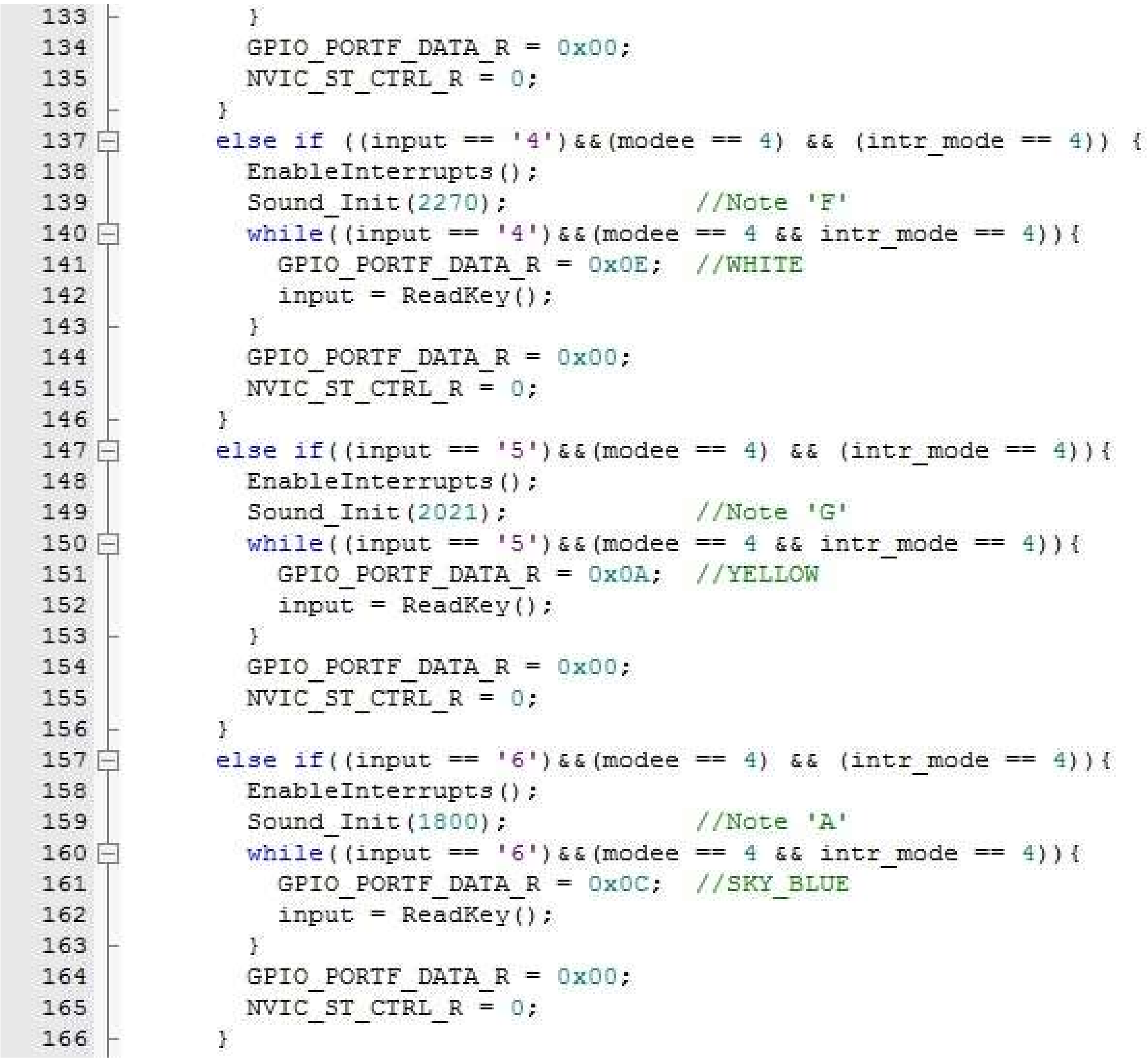
```
int main(void){
  unsigned long input;
  unsigned long period, delay;
  DisableInterrupts();
  PLL_Init();          // bus clock at 80 MHz
  Switch_Init();       // Port F is onboard switches, LEDs, profiling
  Keys_Init();
  EnableInterrupts();
  intr_mode = 0;
  delay = 150;

  while(1) {
    unsigned long modee;
    modee = intr_mode;
    Interruupt_Mode(intr_mode);
    switch(modee) {
      case 1: {
        Sound_Init(2385);  // triangel wave, 262 Hz
      }
      break;
      case 2: {
        Sound_Init(3023);  // sine wave, 262 Hz
      }
      break;
      case 3: {
        Sound_Init(2385);  // square wave, 262 Hz
      }
      break;
      case 4: {
        modee = 4;
      }
      break;
      case 5: {
        modee = 5;
      }
```



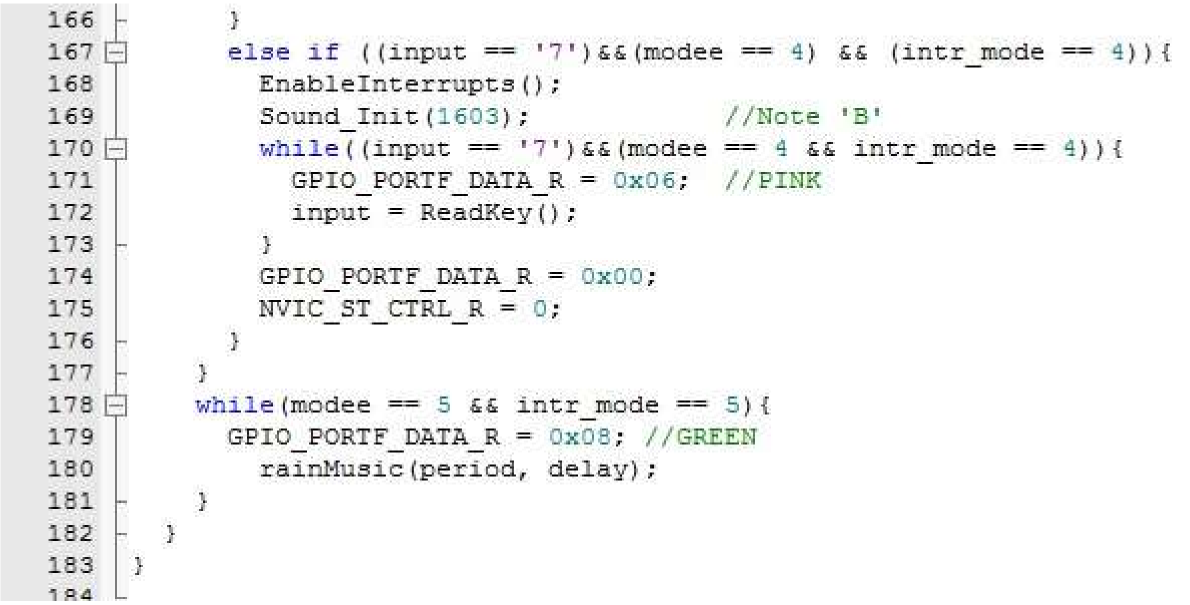
```
      case 5: {
        modee = 5;
      }
      break;
      default: {
        modee = intr_mode;
      }
      break;
    }
    while (modee == intr_mode && modee != 4 && modee !=5) {
      GPIO_PORTF_DATA_R = 0x00;
      if (modee == 1)
        GPIO_PORTF_DATA_R = 0x0E;  //WHITE
      else if (modee ==2)
        GPIO_PORTF_DATA_R = 0x06;  //PINK
      else if (modee == 3)
        GPIO_PORTF_DATA_R = 0x0C;  //SKY-BLUE
    }
```



```
//Matrix Key Program:
    NVIC_ST_CTRL_R = 0;
    while(modee == 4 && intr_mode == 4){
      GPIO_PORTF_DATA_R = 0x00;
      input = ReadKey();   // key press == 0, negative logic
      if ((input == '1')&&(modee == 4) && (intr_mode == 4)) {
        EnableInterrupts();
        Sound_Init(3023);
        while((input == '1')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x08;  //green
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
      else if ((input == '2')&&(modee == 4) && (intr_mode == 4)){
        EnableInterrupts();
        Sound_Init(2694);            //Note 'D'
        while((input == '2')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x02;  //RED
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
      else if ((input == '3')&&(modee == 4) && (intr_mode == 4)){
        EnableInterrupts();
        Sound_Init(2400);            //Note 'E'
        while((input == '3')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x04;  //BLUE
          input = ReadKey();
        }
```



```
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
      else if ((input == '4')&&(modee == 4) && (intr_mode == 4)) {
        EnableInterrupts();
        Sound_Init(2270);            //Note 'F'
        while((input == '4')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x0E;  //WHITE
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
      else if((input == '5')&&(modee == 4) && (intr_mode == 4)){
        EnableInterrupts();
        Sound_Init(2021);            //Note 'G'
        while((input == '5')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x0A;  //YELLOW
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
      else if((input == '6')&&(modee == 4) && (intr_mode == 4)){
        EnableInterrupts();
        Sound_Init(1800);            //Note 'A'
        while((input == '6')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x0C;  //SKY_BLUE
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
```



```
      }
      else if ((input == '7')&&(modee == 4) && (intr_mode == 4)){
        EnableInterrupts();
        Sound_Init(1603);            //Note 'B'
        while((input == '7')&&(modee == 4 && intr_mode == 4)){
          GPIO_PORTF_DATA_R = 0x06;  //PINK
          input = ReadKey();
        }
        GPIO_PORTF_DATA_R = 0x00;
        NVIC_ST_CTRL_R = 0;
      }
    }
    while(modee == 5 && intr_mode == 5){
      GPIO_PORTF_DATA_R = 0x08; //GREEN
        rainMusic(period, delay);
    }
  }
}
```

Conclusion:

While working on the project, there are many failures encountered in trying to generate the correct frequency of sinewave. The output of the wave frequency 3 digit different of square wave and triangle wave. From the failure of this project and be able to fix it late while working on it, it makes clearer of how to interface between software and hardware in manipulating the bit correctly to generate the waveform. To control the mode of each waveform on the Keypad requires logic of Systick clock enable and interrupt enable at the proper condition while executing the program, so that the interrupt will response at the right time and the waveform stop generating when the Keypad is release by stopping the counting of the Systick clock. Since there is bounce of the switch of PF4, the counting of the interrupt mode is not exact. The count number could count up more than one. The reason is because of not using the bouncing delay to of about 20ms before setting the count of the interrupt. This failure does not affect much on generating the sound as the main object of this project. The benefit of working on this project not only give hands on experience but open the deeper understand of how edged trigger and Systick interrupt work and how to control them properly.

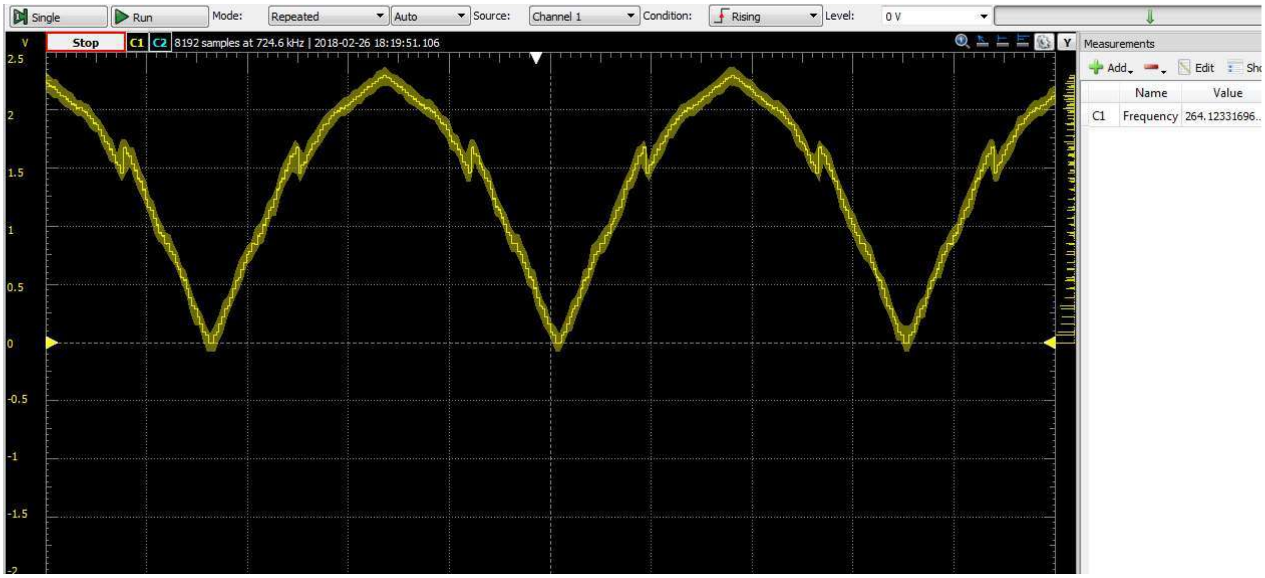
Truth Table:

Calibrating Voltage Value to Binary value

	Decimal	Binary Value						V_out
		D5	D4	D3	D2	D1	D0	Voultage
3.3	0	0	0	0	0	0	0	0
3.3	1	0	0	0	0	0	1	0.0515625
3.3	2	0	0	0	0	1	0	0.103125
3.3	3	0	0	0	0	1	1	0.1546875
3.3	4	0	0	0	1	0	0	0.20625
3.3	5	0	0	0	1	0	1	0.2578125
3.3	6	0	0	0	1	1	0	0.309375
3.3	7	0	0	0	1	1	1	0.3609375
3.3	8	0	0	1	0	0	0	0.4125
3.3	9	0	0	1	0	0	1	0.4640625
3.3	10	0	0	1	0	1	0	0.515625
3.3	11	0	0	1	0	1	1	0.5671875
3.3	12	0	0	1	1	0	0	0.61875
3.3	13	0	0	1	1	0	1	0.6703125
3.3	14	0	0	1	1	1	0	0.721875
3.3	15	0	0	1	1	1	1	0.7734375
3.3	16	0	1	0	0	0	0	0.825
3.3	17	0	1	0	0	0	1	0.8765625
3.3	18	0	1	0	0	1	0	0.928125
3.3	19	0	1	0	0	1	1	0.9796875
3.3	20	0	1	0	1	0	0	1.03125
3.3	21	0	1	0	1	0	1	1.0828125
3.3	22	0	1	0	1	1	0	1.134375
3.3	23	0	1	0	1	1	1	1.1859375
3.3	24	0	1	1	0	0	0	1.2375

3.3	25	0	1	1	0	0	1	1.2890625
3.3	26	0	1	1	0	1	0	1.340625
3.3	27	0	1	1	0	1	1	1.3921875
3.3	28	0	1	1	1	0	0	1.44375
3.3	29	0	1	1	1	0	1	1.4953125
3.3	30	0	1	1	1	1	0	1.546875
3.3	31	0	1	1	1	1	1	1.5984375
3.3	32	1	0	0	0	0	0	1.65
3.3	33	1	0	0	0	0	1	1.7015625
3.3	34	1	0	0	0	1	0	1.753125
3.3	35	1	0	0	0	1	1	1.8046875
3.3	36	1	0	0	1	0	0	1.85625
3.3	37	1	0	0	1	0	1	1.9078125
3.3	38	1	0	0	1	1	0	1.959375
3.3	39	1	0	0	1	1	1	2.0109375
3.3	40	1	0	1	0	0	0	2.0625
3.3	41	1	0	1	0	0	1	2.1140625
3.3	42	1	0	1	0	1	0	2.165625
3.3	43	1	0	1	0	1	1	2.2171875
3.3	44	1	0	1	1	0	0	2.26875
3.3	45	1	0	1	1	0	1	2.3203125
3.3	46	1	0	1	1	1	0	2.371875
3.3	47	1	0	1	1	1	1	2.4234375
3.3	48	1	1	0	0	0	0	2.475
3.3	49	1	1	0	0	0	1	2.5265625
3.3	50	1	1	0	0	1	0	2.578125
3.3	51	1	1	0	0	1	1	2.6296875
3.3	52	1	1	0	1	0	0	2.68125
3.3	53	1	1	0	1	0	1	2.7328125
3.3	54	1	1	0	1	1	0	2.784375
3.3	55	1	1	0	1	1	1	2.8359375
3.3	56	1	1	1	0	0	0	2.8875
3.3	57	1	1	1	0	0	1	2.9390625
3.3	58	1	1	1	0	1	0	2.990625
3.3	59	1	1	1	0	1	1	3.0421875
3.3	60	1	1	1	1	0	0	3.09375
3.3	61	1	1	1	1	0	1	3.1453125
3.3	62	1	1	1	1	1	0	3.196875
3.3	63	1	1	1	1	1	1	3.2484375

Figures:

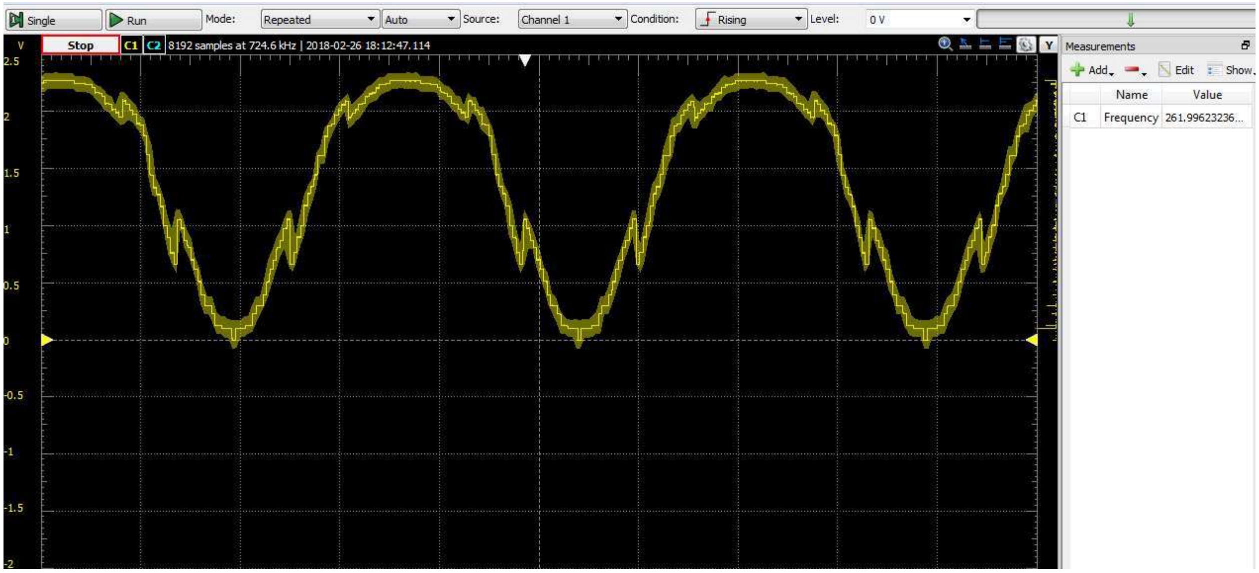


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:19:51.106

Measurements

	Name	Value
C1	Frequency	264.12331696...

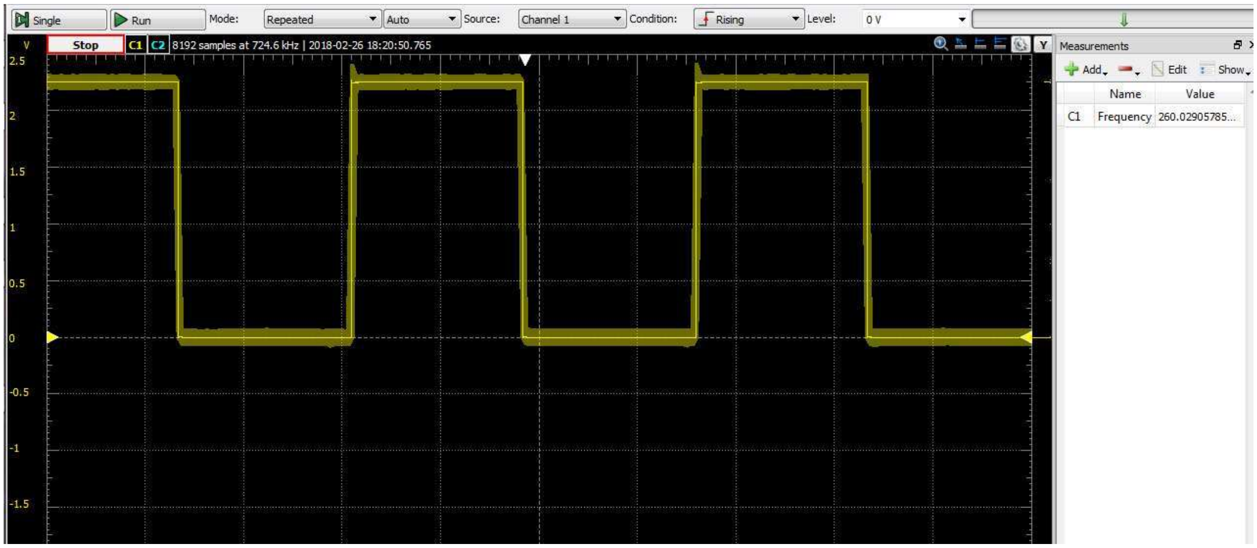


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:12:47.114

Measurements

	Name	Value
C1	Frequency	261.99623236...

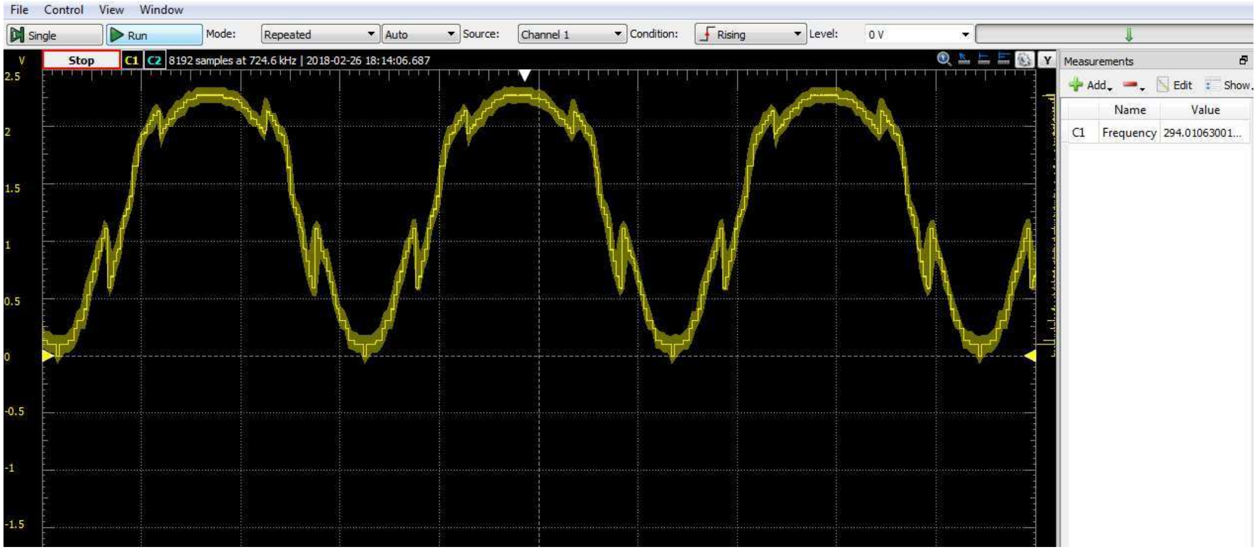


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:20:50.765

Measurements

	Name	Value
C1	Frequency	260.02905785...

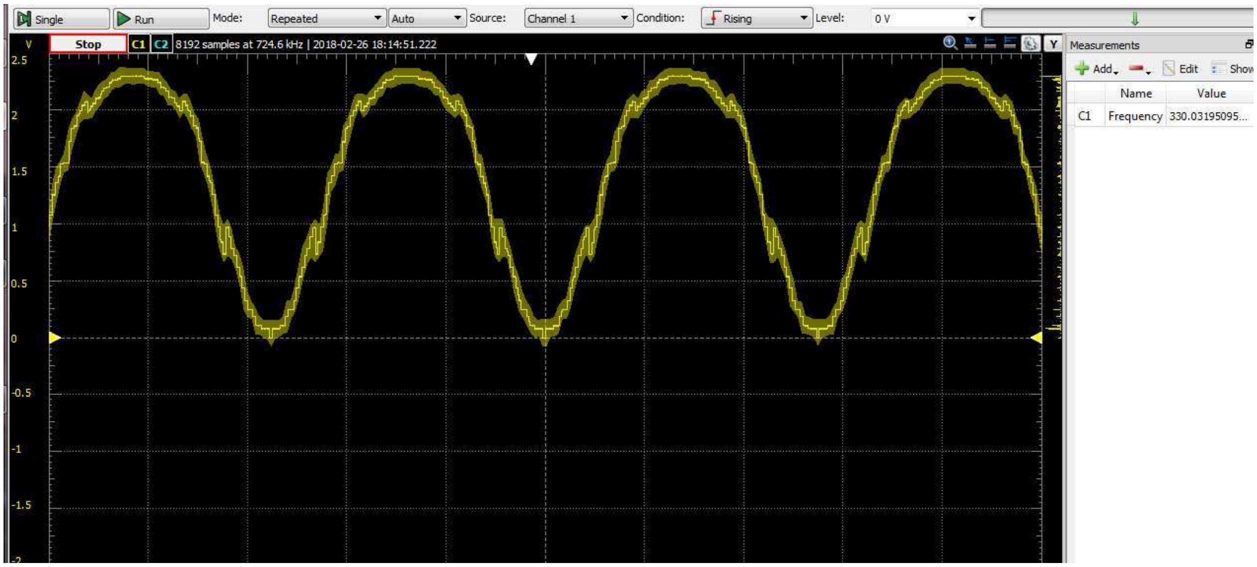


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

Stop C1 C2 8192 samples at 724.6 kHz | 2018-02-26 18:14:06.687

Measurements

	Name	Value
C1	Frequency	294.01063001...

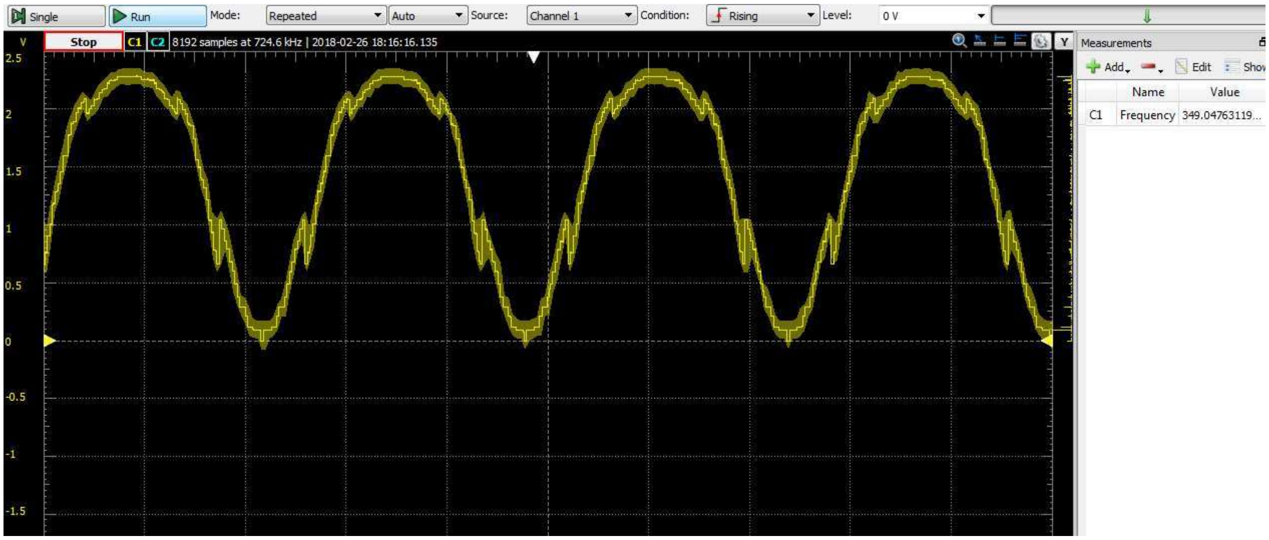


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

Stop C1 C2 8192 samples at 724.6 kHz | 2018-02-26 18:14:51.222

Measurements

	Name	Value
C1	Frequency	330.03195095...

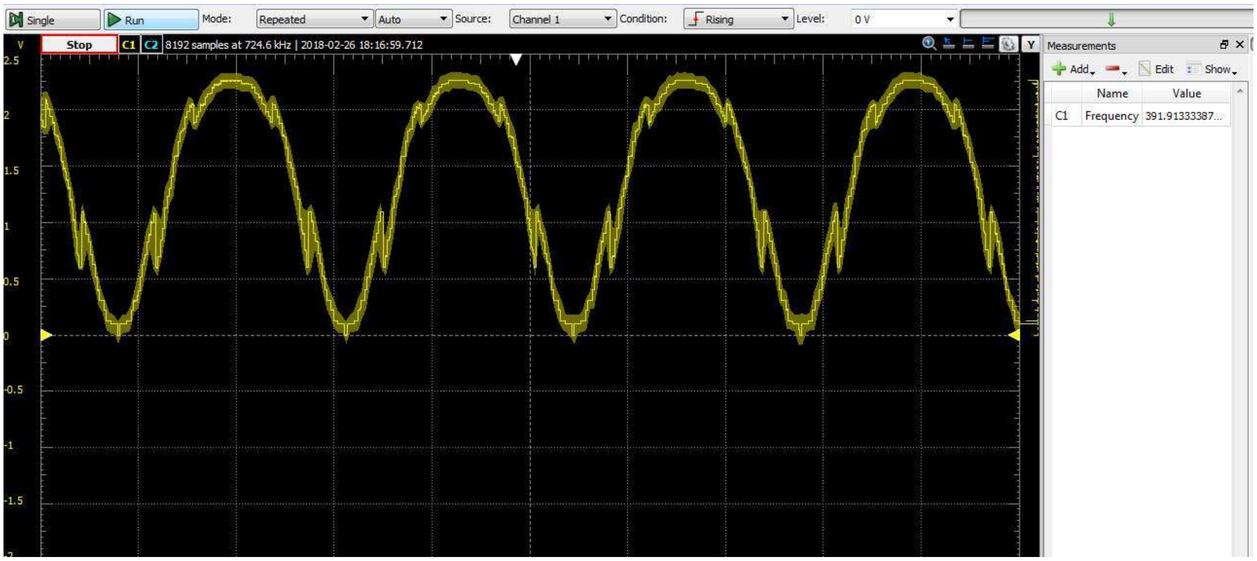


Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

Stop C1 C2 8192 samples at 724.6 kHz | 2018-02-26 18:16:16.135

Measurements

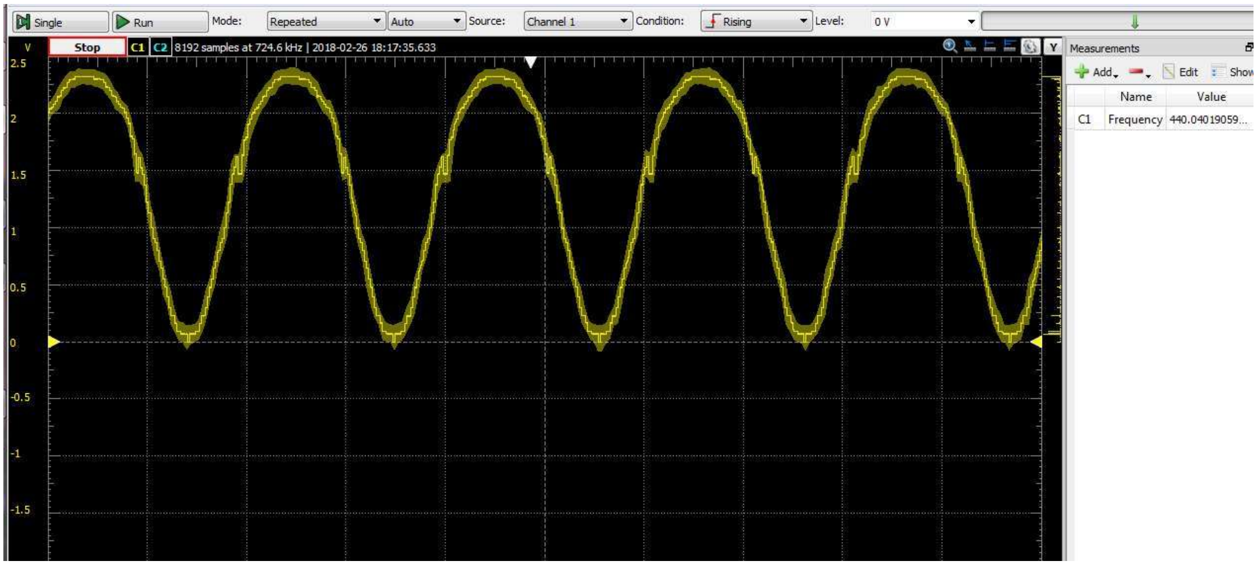
	Name	Value
C1	Frequency	349.04763119...



Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:16:59.712

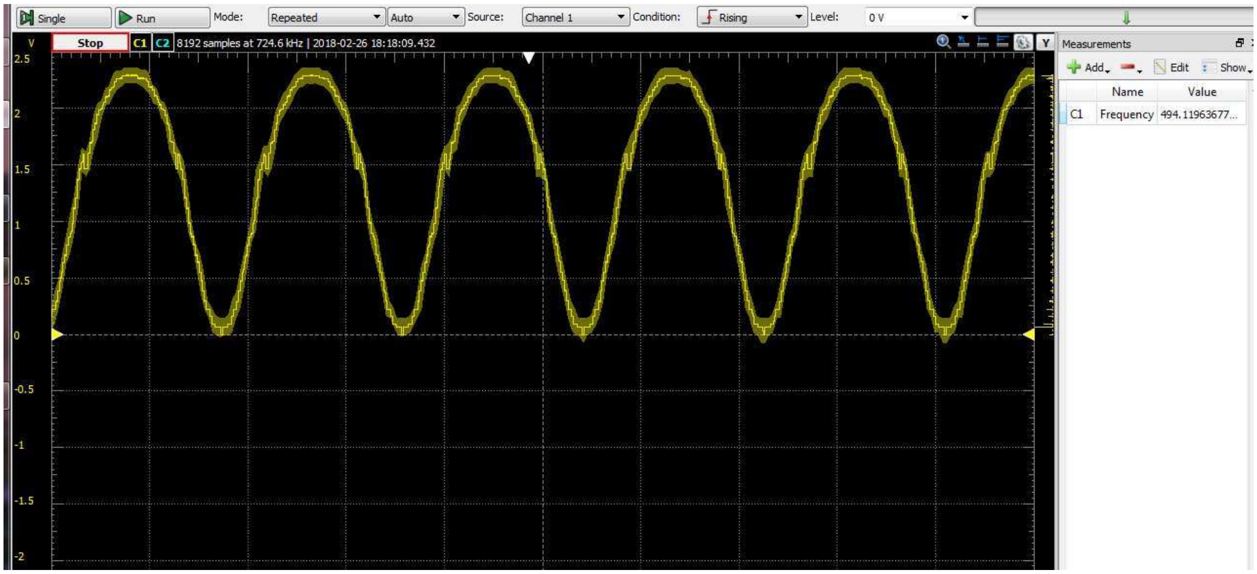
	Name	Value
C1	Frequency	391.91333387...



Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:17:35.633

	Name	Value
C1	Frequency	440.04019059...



Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

8192 samples at 724.6 kHz | 2018-02-26 18:18:09.432

	Name	Value
C1	Frequency	494.11963677...

References:

1. Lecture power point
2. Textbook Chapter 10