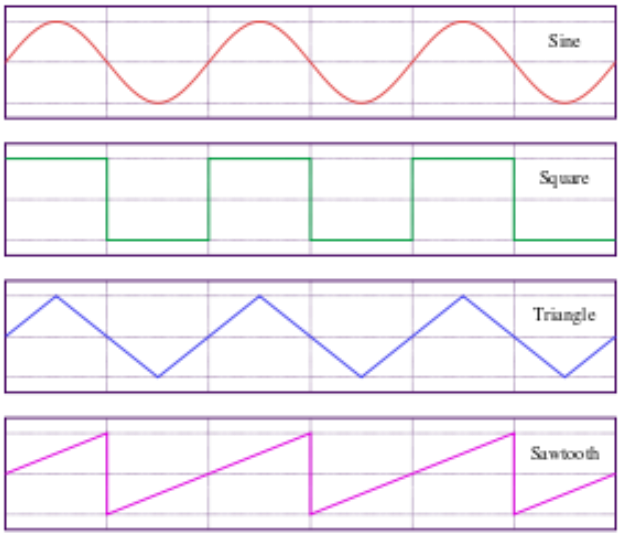
Function Generator

A **function generator** is usually a piece of electronic test equipment or software used to generate different types of electrical waveforms over a wide range of frequencies. Some of the most common waveforms produced by the function generator are the sine, square, triangular and saw-tooth shapes. These waveforms can be either repetitive or single-shot (which requires an internal or external trigger source)



Specifications:

Controller: 89v51rd2 (8051 variant)

User Interface: Interactive form

Misc: DAC 0808 (all in one card kit of RVCE)

Microcontroller program

```
ORG 0000
LJMP AAA
DLY: DB 255,175,125,75,50,25,10,2
SINETAB: DB
0,6,13,19,26,32,39,45,51,57,63,69,74,79,84,89,94,98,102,106,109,113,116,118,120,122,124,125,126,
126,127
TRITAB: DB
0,4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76,80,84,88,92,96,100,104,108,112,116,120
AAA:
        MOV SCON,#50H   ;configuring the serial port
        MOV TMOD,#20H
        MOV TH1,#-3H   ; timer 1 required for the above purpose
        SETB TR1
        MOV P0,#00H
        MOV P1,#0FFH
    AA:
        ACALL GETCH  ;read a byte to P1
        MOV A,P1
        CLR ACC.7
```

```
        CLR ACC.6
        CLR ACC.5
        MOV DPTR,#DLY
        MOVC A,@A+DPTR
        MOV R2,A
        MOV TMOD,#01H
        JB P1.5,SINTRN
        JB P1.6,SWTTH
        JMP DIGI
        SJMP $

SWTTH:                  ;To generate saw-tooth wave
        MOV R3,#118
        MOV A,#7FH
        LP:
                MOV P0,A
                ACALL DELAY
                INC A
                CJNE A,#118,LP
                JB RI,AA
                MOV A,#7FH
                JMP LP

DIGI:                   ;To generate square wave
         MOV A,#0FFH

         MOV R3,#59
  UPPER:                ;upper half
         MOV P0,A
         ACALL DELAY
         DJNZ R3,UPPER

         MOV A,#7FH
         MOV R3,#59
        LOWER:          ;lower half
         MOV P0,A
         ACALL DELAY
         DJNZ R3,LOWER

         JB RI,AA
         JMP DIGI


SINTRN:                 ;common code sine and triangular wave generation
    JB P1.6,TRN
    MOV DPTR,#SINETAB
          JMP START
        TRN: MOV DPTR,#TRITAB
        START:
         MOV R3, #30
         LP1:           ;first quarter of one cycle of the wave
```

```
            CLR A
            MOVC A,@A+DPTR
            ADD A,#127
            MOV P0,A
            ACALL DELAY
            INC DPTR
            DJNZ R3,LP1



    MOV R3,#29
    MOV A,#29
    JB P1.6,TRN1
MOV DPTR,#SINETAB
      JMP START1
    TRN1: MOV DPTR,#TRITAB
    START1:
    LP2:        ;second quarter of one cycle of the wave
      PUSH 0E0H
            MOVC A,@A+DPTR
            ADD A,#127
            MOV P0,A
            ACALL DELAY
            POP 0E0H
            DEC A
            DJNZ R3,LP2



      MOV R3,#30
     LP3:        ; third quarter of one cycle of the wave
            CLR A
            MOVC A,@A+DPTR
            MOV R4,A
            MOV A ,#127
            SUBB A,R4
            MOV P0,A
            ACALL DELAY
            INC DPTR
            DJNZ R3,LP3

            JB P1.6,TRN2

   MOV DPTR,#SINETAB
     JMP START2
    TRN2: MOV DPTR,#TRITAB
    START2:
            MOV A,#29
            MOV R3,#29
    LP4:        ; fourth quarter of one cycle of the wave
            PUSH 0E0H
```

```
                MOVC A,@A+DPTR
                MOV R4,A
                MOV A,#127
                SUBB A, R4
                MOV P0,A
                ACALL DELAY
                POP 0E0H
                DEC A
                DJNZ R3,LP4

          JB RI,TEMP
                JMP START
        TEMP:JMP AA

  DELAY:
                PUSH 0E0H                    ;DELAY procedure uses Timer 0
                CLR A
                SUBB A,R2
                MOV TL0,A
                MOV TH0,#0FFH
                SETB TR0
        REP1:JNB TF0,REP1
                 CLR TF0
            CLR A
                SUBB A,#0
                POP 0E0H
                RET

GETCH:          ;serial communication- read from the UI
GAGAIN:         JNB RI,GAGAIN
                 MOV P1,SBUF
                CLR RI
                RET

END
```

Building the User Application Interface using Visual Studio 2010

The function generator project also requires an user interface to be built, so that the user can select the waveform and the required frequency in the application provided and that data can be sent to the microcontroller which when received can be decoded appropriately. Hence, the user need not have the knowledge of bit patterns as described above and need not toggle switches of the All-In-One Card Kit.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Windows.Forms;


namespace project
{
   public partial class Form1 : Form
   {
      byte[] b =new byte[1];
      byte[] temp = new byte[1];
      int count = 0;
      public Form1()
      {
         InitializeComponent();

      }

      private void Form1_Load(object sender, EventArgs e)
      {
         serialPort1.Open(); //open the serial port for communication


      }

      private void button1_Click_1(object sender, EventArgs e)
      {
         count++;
                  //a second “click on Generate without reselect”
         if (count > 1)
            serialPort1.Write(temp, 0, 1);  //send data
         else
            serialPort1.Write(b, 0, 1);
          //make every radio button uncheched
         Square.Checked =false;
         Sine.Checked =false;
         Sawtooth.Checked = false;
         Triangular.Checked =false;
      //reinitialise b to 10000000B
         b[0] = (byte)0x80;
      }


      private void Square_CheckedChanged(object sender, EventArgs e)
      {

         if (Square.Checked == true)
         {

            b[0] = (byte)0x80;
            temp[0] = b[0];
                  //store b in temp (in case a second Generate button Click)

         }
```

```
}

private void Sine_CheckedChanged(object sender, EventArgs e)
{

    if (Sine.Checked == true)
    {

        b[0] = (byte)0xA0;
        temp[0] = b[0];
        count = 0;
    }


}

private void Sawtooth_CheckedChanged(object sender, EventArgs e)
{
    if (Sawtooth.Checked== true)
    {

        b[0] = (byte)0xC0;
        temp[0] = b[0];
        count = 0;

    }

}

private void Triangular_CheckedChanged(object sender, EventArgs e)
{

    if (Triangular.Checked == true)
    {

        b[0] = (byte)0xE0;
        temp[0] = b[0];
        count = 0;
    }

}

private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    int i = 0;
  for (i = 0; i <= 7; i++)
  {
    if (listBox1.GetSelected(i) == true)
    {

        count = 0;

        switch (i)
        {
            case 0: b[0] = (byte)((int)b[0] | 0x00); break;
```

```
                case 1: b[0] = (byte)((int)b[0] | 0x01); break;
                case 2: b[0] = (byte)((int)b[0] | 0x02); break;
                case 3: b[0] = (byte)((int)b[0] | 0x03); break;
                case 4: b[0] = (byte)((int)b[0] | 0x04); break;
                case 5: b[0] = (byte)((int)b[0] | 0x05); break;
                case 6: b[0] = (byte)((int)b[0] | 0x06); break;
                case 7: b[0] = (byte)((int)b[0] | 0x07); break;

            }
            temp[0] = b[0];
        } //end of if
    }   //end of for
  } //end of listBox1_SelectedIndexChanged()
 } //end of form class
} //end of namespace
```

Snapshot of the user interface