"Let your gentleness be evident to all."

23 Sound Programming with PC Speaker

Sound programming can be classified as with PC speaker and with sound blaster card. In this chapter, let's see sound programming with PC speaker.

23.1 Introduction

Almost all systems have PC speaker. People who like to have digitized sound go for MIDI card or sound blaster card. But for normal operations, it is enough to have PC speaker.

23.2 Programming PIT

For sound programming with PC speakers, we must be aware of PIT (Programmable Interval Timer) that is present on our microcomputer system. PIT or 8253 chip is an LSI peripheral designed to permit easy implementation of timer. People from Electronics background may be aware that Timer is the one which produces clock signals. And so PIT can be setup to work as a one shot pulse generator, square wave generator or as rate generator. We can set the PIT to supply the required frequency by supplying values 'N' to the port 43h.

$$\text{Formula to calculate } N = \frac{1.9 \text{ MHz}}{f}$$

where f is the required frequency

The sequence of operations be:

i. Initialize PIT to accept divisor by OUTing B6h at 43h.
ii. OUT LSB of 'N' at 42h
iii. OUT MSB of 'N' at 42h

Now the PIT will produce clock signals with the frequency 'f'.

23.3 Producing Sound

If we connect a timer with PC speaker, it will produce sound. We can connect PIT with PC speakers to get the required sound. The output port of speaker is 61h. bit0 of port 61h is used to enable timer to supply clock signal to speaker i.e. connects PIT with speaker.

Now let's write our own `sound( )` and `nosound( )` function to produce sound.

```
#define ON        (1)
#define OFF       (0)

/*-----------------------------------------------
      ChangeSpeaker - Turn speaker on or off.    */

void ChangeSpeaker( int status )
{
   int portval;
   portval = inportb( 0x61 );
   if ( status==ON )
            portval |= 0x03;
        else
            portval &=~ 0x03;
   outportb( 0x61, portval );
} /*--ChangeSpeaker( )----------*/

void Sound( int hertz )
{
   unsigned divisor = 1193180L / hertz;

   ChangeSpeaker( ON );

   outportb( 0x43, 0xB6 );
   outportb( 0x42, divisor & 0xFF ) ;
   outportb( 0x42, divisor >> 8 ) ;
} /*--Sound( )-----*/

void NoSound( void )
{
   ChangeSpeaker( OFF );
} /*--NoSound( )------*/

int main( void )
{
   Sound( 355 );
   delay( 1000 );
   Sound( 733 );
   delay( 1000 );
   NoSound( );
   return(0);
} /*--main( )-------*/
```

TC also has `sound( )` and `nosound( )` functions. If you don't want to write your own code, you can use those built-in functions.

23.4 Notes & Frequencies

You may want to know the frequencies of each *note* to produce the right sound. In general, an octave is a doubling in frequency. There are twelve distinct tones in an octave. The frequencies of higher octaves are just a multiple of frequencies for lower octaves. The note 'A' below “middle C” is exactly 440Hz. Other notes may be calculated from this by using a simple formula:

$$Frequency = 440 * 2^{(Offset / 12)}$$

where Offset is the “distance” between note 'A' and the note in semitones.

Using the above formula, any part of the frequency table can be calculated. The following program demonstrates this.

```
#include <math.h>
char  *Note_Names[] =
                  {
                     "A",
                     "B Flat",
                     "B",
                     "C",
                     "C Sharp",
                     "D",
                     "E Flat",
                     "E",
                     "F",
                     "F Sharp",
                     "G",
                     "G Sharp"
                  };
int main( void )
{
   double frequency;
   int offset;
   for( offset=0; offset<13; ++offset )
    {
      frequency = 440.0 * pow( 2.0, offset / 12.0 );
      printf( "The Frequency of %s is %f Hz\n",
                        Note_Names[offset%12], frequency );
    }
   return(0);
} /*--main( )--------*/
```

23.5 Piano Keys and Frequencies

The following diagram shows the frequencies for a typical Piano.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
27.500	29.135	30.868	32.703	24.648	36.708	30.891	41.203	43.654	46.249	48.999	51.913	55.000	58.270	61.735	65.406	69.296	73.416	77.782	82.407	87.307	92.499	97.999	103.83	110.00	116.54	123.47	130.81	138.59	146.83	155.56	164.81

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
174.61	185.00	196.00	207.65	220.00	233.08	246.94	261.63	277.18	293.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	880.00	932.33	987.77

64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88
1046.5	1108.7	1174.7	1244.5	1318.5	1396.9	1460.0	1568.0	1661.2	1760.0	1864.7	1975.5	2093.0	2217.5	2349.3	2489.0	2637.0	2793.8	2960.0	3136.0	3322.4	3520.0	3729.3	3951.1	4186.0

23.6 Piano Program

The following is the code for a Piano program. The main idea here is you have to use port 60h to get a key, you should not use `getch( )`. Since we are using port 60h, the keyboard buffer won't get cleared automatically. So we should clear the keyboard buffer very often to avoid unnecessary beep sound that signals the keyboard buffer's full status.

This program will provide you the opportunity to try 8 octaves. As the frequencies of higher octaves are just a multiple of frequencies of lower octaves, I could have used a single dimensional array `notes[12]`. But I have used a two dimensional array `notes[7][12]` to avoid calculations and to increase the speed.

```
#define  ESC      (129)

#include <stdio.h>
#include <conio.h>
#include <dos.h>

int main( void )
{
   void ClrKeyBrdBuffer( );
   float notes[7][12] =
      {
         { 130.81, 138.59, 146.83, 155.56, 164.81, 174.61, 185.0,
                  196.0, 207.65, 220.0, 227.31, 246.96 },
         { 261.63, 277.18, 293.66, 311.13, 329.63, 349.23, 369.63,
                  392.0, 415.3, 440.0, 454.62, 493.92 },
         { 523.25, 554.37, 587.33, 622.25, 659.26, 698.46, 739.99,
                  783.99, 830.61, 880.0, 909.24, 987.84 },
         { 1046.5, 1108.73, 1174.66, 1244.51, 1328.51, 1396.91, 1479.98,
                  1567.98, 1661.22, 1760.0, 1818.48, 1975.68 },
         { 2093.0, 2217.46, 2349.32, 2489.02, 2637.02, 2793.83, 2959.96,
                  3135.96, 3322.44, 3520.0, 3636.96, 3951.36 },
         { 4186.0, 4434.92, 4698.64, 4978.04, 5274.04, 5587.86, 5919.92,
                  6271.92, 6644.88, 7040.0, 7273.92, 7902.72 },
         { 8372.0, 8869.89, 9397.28,9956.08,10548.08,11175.32, 11839.84,
                  12543.84, 13289.76, 14080.0, 14547.84, 15805.44 }
      };
   int n, i, p, q, octave = 2,
      note[ ] = { 1, 3, 99, 6, 8, 10, 99, 13, 15, 99, 18, 0, 2, 4, 5, 7,
                  9, 11, 12, 14, 16, 17 };
      /* keys[]="awsedftgyhujkolp;']"  <- for note[]  */
   clrscr( );
   printf( "Piano for A to Z of C \n\n"
         "Note-> C Df D Ef E F Fs G Af A Bf B C Df D Ef E F Fs \n"
         "Keys-> a w  s e  d f t  g y  h u  j k o  l p  ; ' ] \n\n"
         "Octave-> 1 2 3 4 5 6 7 8 \n\n"
```

```
         "Quit-> ESC \n" );
   while( (n=inportb(0x60)) != ESC )
     {
      ClrKeyBrdBuffer( );
      p = 2;      /*dummy*/
      if ( n>=2&&n<=8 )
           octave = n-2;
       else
           switch( n )
            {
               case 79:
               case 80:
               case 81: octave = n-79;
                        break;
               case 75:
               case 76:
               case 77: octave = n-72;
                        break;
               case 71: octave = 6;
            }
       if ( n>=17&&n<=27 )
            p = n-17;
          else if ( n>=30&&n<=40 )
            p = n-19;
       p = note[p];
       if ( p>=0&&p<=21 )
            sound( (int)notes[octave][p] );
       if ( n>136 )
            nosound( );
     }
   printf( "Quiting..." );
   getch( );
   return(0);
} /*--main( )----------*/

void ClrKeyBrdBuffer(void)
{
   outportb( 0x20, 0x20 );    /* reset PIC */
   while( bioskey(1) )        /* read all chars until it empty */
            bioskey( 0 );
} /*--ClrKeyBrd( )------*/
```

Exercise

1. Using program find out the frequency and delay used for ordinary beep sound that is produced by `printf("\a");`. Do not use any gadgets or Trial and Error Techniques.

Suggested Projects

1. Write software that plays MIDI files through PC speaker.