

# PC INTERFACED AUDIO PLAYBACK DEVICE: M-PLAYER



N.V. VENKATARAYALU AND M. SOMASUNDARAM

Sounds of various kinds have always fascinated human beings. Many devices have been invented for recording and playing back the sounds—from magnetic tapes to DVD (digital versatile disc), from Adlib cards to high-performance sound cards with 'surround sound' capability. For personal computers (PCs), there is a wide variety of such devices. A modern PC, generally, has a 'Sound Blaster' card installed in it. If your PC does not have a sound card, here is a low-cost audio playback circuit with bass, treble, and volume controls to create your own music player.

The playback device 'M-player' (i.e. media player) described here uses minimal hardware to achieve a moderately good-quality audio playback device. The software that accompanies the hardware is meant for a PC running under MS-DOS or a compatible operating system. This device can play a simple 8-bit PCM (pulse code modulation) wave file with some special effects. The PC is connected to the device through the PC parallel port.

## Hardware

The circuit functions as an 8-bit mono player, i.e. the sound files (with .WAV extension) with sound quantised to eight bits or 256 levels can be played. In case of files with 16-bit quantisation, these are re-quantised as discussed under 'Software' subheading. Thus, only eight bits are sent to the card through the printer port.

Since there is no duplex communication necessary between the player card and the PC, it is sufficient to use the eight output data lines of the port 378H (pins 2 through 9 of 25-pin D-connector). This 8-bit digital output is converted into an analogue signal using DAC 0808 (IC1) from National Semiconductor.

The output current from the DAC varies with the input digital level (represented by bits D0 through D7), the reference voltage ( $V_{ref}$ ), and the value of series resistor R1 connected to  $V_{ref}$  pin 14 of DAC0808 IC. The output current  $I_o$  (in mA) is given by the relationship:

$$I_o = \frac{V_{ref}}{R1} \left[ \frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256} \right]$$

where  $V_{ref}$  is the reference voltage in volts and R1 is the resistance in kilo-ohms.

The output current from the DAC is converted into its corresponding voltage using a simple current-to-voltage converter wired around one part of the dual wideband JFET op-amp LF353. The output from IC2(a) is the required audio signal that has to be processed and amplified to feed the speaker. The part following the I-V converter is the bass- and treble-control circuit employing RC-type variable low-pass and high-pass filters connected to the input of audio amplifier built around the second op-amp inside LF353 [IC2(b)].

The frequency response of the filters can be varied using potentiometers VR1 and VR2. The low frequencies or bass can be cut or boosted with the help of potentiometer VR1. Similarly, high frequencies or treble can be cut or boosted with the help of potentiometer VR2. At low frequencies, capacitors C2, C3, and C4 act as open circuits and the effective feedback is through 10k resistors (R4, R5, and R6) and potentiometer VR1.

The audio amplifier IC2(b) acts as an inverting amplifier and the amplification (or attenuation) of the low-frequency bass signals depends on the value of potentiometer VR1. The frequency  $f_1$  at which  $C = C2 = C3$  becomes effective is given by the equation:

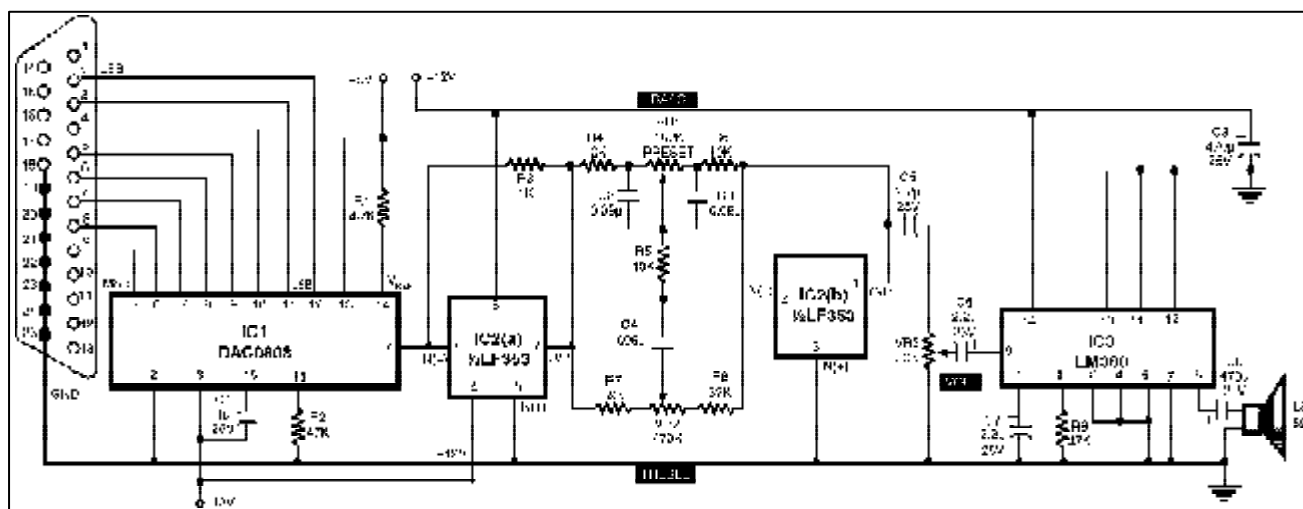


Fig. 1: Circuit of M-player audio playback device

$$f_2 = \frac{1}{2\pi \times \sqrt{R_1} \times C_1} \parallel x$$

At frequencies higher than  $f_2$  ( $f > f_2$ , high end of audio range), capacitors C2 and C3 overcome the effect of potentiometer VR1. As C2 and C3 behave as short, potentiometer VR1 has no effect on the output response. Now, the gain is controlled by treble potentiometer VR2. The frequency  $f_2$ , below which treble potentiometer VR2 has no effect on the response, is given by the equation:

$$f_2 = \frac{1}{2\pi \times \sqrt{R_2} \times C_2} \parallel F_2$$

The output of this module is sent to the final 2-watt audio power amplifier (LM380) stage through potentiometer VR3 which is used as the volume control. The power output of this module is fed to an 8-

ohm speaker. The output-end audio power amplifier is designed to give a gain of around 50.

One can also use LM380 in various other configurations as per one's requirements. Another popular configuration is the 'bridge configuration'—in which two LM380s can be used to obtain larger power output with a gain of 300.

Fig. 2: Actual-size PCB layout for M-player

### Parallel port

The output of the parallel port is TTL compatible. So, logic level 1 is indicated by +5V and logic level 0 by 0V. The current that one can sink and source varies from port to port. Most parallel ports can sink and source around 12 mA.

Fig. 3: Component layout for the PCB

The software assumes 0x378 (378H) to be the base address of the parallel port to which the device is connected. Another possible base address is 0x278 (278H). It is advised to modify this address of the parallel port in the software program, after checking the device profile.

Actual-size PCB layout for audio playback circuit of Fig. 1 is given in Fig. 2 and its component layout in Fig. 3.

### Software

The software accompanying this construction project is written in Turbo C/C++ for DOS. It can be used to play simple 8-bit PCM wave files. 16-bit wave files are converted into 8-bit PCM data before proceeding.

Even stereo wave files can be played; but not the stereo way. Only one channel is chosen. Up to six-channel PCM data can be read and con-

verted into mono 8-bit PCM data. This software is accompanied with a C<sub>T</sub>UI-based interface.

The wave file format is probably the least undocumented sound format since there are different schemes with different number of chunks of related information in the file. Even the chunks can be of variable size. Therefore it is difficult to get documentation on all available chunks.

This software can be used only on PCM data with data chunk. Every wave file has some minimum chunks (see Table II). These chunks will be present in every wave file. Then there are other chunks which are actually non-standard. In PCM itself, the above chunk may be followed either by DATA chunk or by LIST chunk which, in turn, has lots of sub-chunks. (Any information obtained on these chunks by the readers may please be shared with the authors.)

During playback, the speed with which the processor in the PC can execute the main loop is first studied using a dummy loop and thus the delay is adaptively varied with respect to the speed

### PARTS LIST

<i>Semiconductors:</i>	
IC1	- DAC0808 8-bit D/A converter
IC2	- LF353 JFET input wide-band op-amp
IC3	- LM380, 2-watt audio amplifier
<i>Resistors (all 1/4 watt, ±5% carbon film, unless stated otherwise)</i>	
R1	- 4.7-kilo-ohm
R2, R9	- 47-kilo-ohm
R3	- 1-kilo-ohm
R4-R6	- 10-kilo-ohm
R7, R8	- 39-kilo-ohm
VR1	- 100-kilo-ohm potmeter
VR2	- 470-kilo-ohm potmeter
VR3	- 50-kilo-ohm potmeter
<i>Capacitors:</i>	
C1	- 1µF, 25V electrolytic
C2, C3	- 0.05µF ceramic disk
C4	- 0.005µF ceramic disk
C5-C7	- 2.2µF, 25V electrolytic
C8, C9	- 470µF, 25V electrolytic
<i>Miscellaneous:</i>	
	- 25-pin D connector (male)
	- Loudspeaker 8-ohm, 2W
	- Power supply: (a) +12V, 500mA
	- (b) -12V, 100mA
	- (c) +5V, 100mA

TABLE I

### Relevant Details of Parallel Port

Pin No. (D-type 25)	Pin No. (centronics)	SPP signal	Direction in/out	Register
2	2	Data 0	Out	Data
3	3	Data 1	Out	Data
4	4	Data 2	Out	Data
5	5	Data 3	Out	Data
6	6	Data 4	Out	Data
7	7	Data 5	Out	Data
8	8	Data 6	Out	Data
9	9	Data 7	Out	Data
18 - 25	19-30	Ground	Gnd	

**TABLE II**  
Wave File Format

From byte	Number of bytes	Information
<b>RIFF chunk:</b>		
0	4	Contains the characters 'RIFF'
4	4	Size of the RIFF chunk
<b>WAVE chunk:</b>		
0	4	Contains the characters 'WAVE'
4	Variable	The FORMAT chunk
<b>The normal FORMAT chunk:</b>		
0	4	Contains the characters 'fmt'
4	4	Size of the FORMAT chunk
8	2	Value specifying the scheme 1-PCM, 85-MPEG layer III
10	2	Number of channels 1-mono, 2-stereo, etc.
12	4	Number of samples per second. This gives us the playback rate.
16	4	Average number of bytes per second. This field is used to allocate buffers, etc.
20	2	Contains block alignment information.
22	Variable	This field contains format-specific data. For PCM files, this field is 2 bytes long

of target processor. This is one of the methods to achieve invariance of the playback speed over a wide range of processor speeds available.

bass, treble, and volume for the wave file that is played. Thus, the hardware and software complement each other to provide a good music player. The software

The software can be used to play with the following effects:

- Play normally
- Play with a different playback rate, i.e. play it fast or slow
- Fade-in or fade-out the volume levels either linearly or exponentially
- Reverse the wave file and then play

The menu items can be selected using keyboard keys Alt+F for file, Alt+E for effects, and Alt+O for operation. Apart from the software, the hardware can be used to vary

does not include mouse support.

## Conclusion

We have presented a simple sound card to playback .wav files with bass and treble controls. Though the current design plays only mono files (stereo files are converted to mono), a stereo file player can be designed in a similar manner. The software can be modified to play audio files other than .wav files without any change in the hardware circuit. The encoding format of the other audio file types (like .ra, .mp3) is only to be known. With that, those files can be decoded and raw digital 8-bit data can finally be sent to the hardware device. The hardware device can even be permanently mounted inside the PC with all the power supplies (+12V, +5V, and -12V) tapped from the system's SMPS.

*Note: The complete source code consisting of Mplayer.cpp, Sounds.h, Globals.h, the executable file Mplayer.exe, and a sample wave file are likely to be included in next month's CD (optional) accompanying EFY.*

## Program Listing

```

MPLAYER.CPP
#include "Sounds.h"
void DisplayTip(char *string)
{
    text_info tinf;
    if(strlen(string)<75)
    {
        gettextinfo(&tinf);
        textbackground(LIGHTGRAY);textcolor(RED);
        gotoxy(2,25);
        for(int i=0;i<75;i++) printf(" ");
        gotoxy(2,25);
        printf(string);
        textattr(tinf.attribute);
        gotoxy(tinf.curx,tinf.cury);
    }
    return;
}
void Window(int x1,int y1,int x2,int y2,char
            *caption,int BackCol,int TextCol)
{
    text_info tinfo;
    int ij;
    gettextinfo(&tinfo);
    textbackground(BackCol);textcolor(TextCol);
    for(j=y1;j<=y2;j++){
        gotoxy(x1,j);
        for(i=x1;i<=x2;i++)
            printf(" ");
        gotoxy(x1+1,y1);
        for(i=x1+1;i<=x2-1;i++)
            printf("%c",205);
        gotoxy(x1+1,y2);
        for(i=x1+1;i<=x2-1;i++)

```

```

            printf("%c",205);
            for(j=y1+1;j<=y2-1;j++){
                gotoxy(x1,j);
                printf("%c",186);
                gotoxy(x2,j);
                printf("%c",186);
            }
            gotoxy(x1,y1);printf("%c",201);
            gotoxy(x2,y1);printf("%c",187);
            gotoxy(x1,y2);printf("%c",200);
            gotoxy(x2,y2);printf("%c",188);
            if(caption!=NULL){
                textcolor(WHITE);
                gotoxy(x1+2,y1);
                printf("%s",caption);
            }
            textattr(tinfo.attribute);
            return;
        }
        void DrawScreen(void)
        {
            textbackground(LIGHTGRAY);textcolor(BLACK);
            clrscr();
            Window(1,2,80,24,NULL,BLUE,WHITE);
            gotoxy(1,1);printf(" File Effects Operation");
            textcolor(RED);
            gotoxy(3,1);printf("F");
            gotoxy(12,1);printf("E");
            gotoxy(24,1);printf("O");
            textbackground(BLUE);textcolor(LIGHTBLUE);
            gotoxy(3,10);printf(" ");
            delay(75);
            gotoxy(3,11);printf(" ");
            delay(75);
            gotoxy(3,12);printf(" ");

```

```

            delay(75);
            gotoxy(3,13);printf(" ");
            delay(75);
            gotoxy(3,14);printf(" ");
            delay(75);
            gotoxy(3,15);printf(" ");
            delay(75);
            gotoxy(3,16);printf(" ");
            delay(75);
            gotoxy(3,17);printf(" ");
            return;
        }
        void MenuInitialise(void)
        {
            int i;
            // The FILE menu option
            Menu[MNU_FILE].nextMenu=MNU_EFFECT;
            Menu[MNU_FILE].prevMenu=MNU_OPERATION;
            Menu[MNU_FILE].Child=FALSE;
            Menu[MNU_FILE].num_items=4;
            for(i=0;i<4;i++)
            {
                Menu[MNU_FILE].Enabled[i]=TRUE;
                Menu[MNU_FILE].subMenu[i]=NONE;
                Menu[MNU_FILE].String[i]=(char *)malloc(15);
                Menu[MNU_FILE].Tip[i]=(char *)malloc(50);
                Menu[MNU_FILE].OptionID[i]=1+i;
            }
            Menu[MNU_FILE].Enabled[1]=FALSE;
            strcpy(&(Menu[MNU_FILE].String[0][0]),"Open");
            strcpy(&(Menu[MNU_FILE].String[1][0]),"Save");
            strcpy(&(Menu[MNU_FILE].String[2][0]),".");
            strcpy(&(Menu[MNU_FILE].String[3][0]),"Exit");
            strcpy(&(Menu[MNU_FILE].Tip[0][0]),"Open the
                *.wav file");

```

```

strcpy(&(Menu[MNU_FILE].Tip[1][0]),"Save as a
      *wav file");
strcpy(&(Menu[MNU_FILE].Tip[2][0])," ");
strcpy(&(Menu[MNU_FILE].Tip[3][0]),"Quit the
      program");
Menu[MNU_FILE].AtX=2;Menu[MNU_FILE].AtY=2;
// The EFFECT menu option
Menu[MNU_EFFECT].nextMenu=MNU_OPERATION;
Menu[MNU_EFFECT].prevMenu=MNU_FILE;
Menu[MNU_EFFECT].Child=FALSE;
Menu[MNU_EFFECT].num_items=5;
for(i=0;i<5;i++)
{
Menu[MNU_EFFECT].Enabled[i]=FALSE;
Menu[MNU_EFFECT].subMenu[i]=NONE;
Menu[MNU_EFFECT].String[i].String[i]=
      (char*)malloc(15);
Menu[MNU_EFFECT].Tip[i]=(char *)malloc(50);
Menu[MNU_EFFECT].OptionID[i]=11+i;
}
strcpy(&(Menu[MNU_EFFECT].String[0][0]),"Fade
      In");
strcpy(&(Menu[MNU_EFFECT].String[1][0]),"Fade
      Out");
strcpy(&(Menu[MNU_EFFECT].String[2][0]),"-");
strcpy(&(Menu[MNU_EFFECT].String[3][0]),"Reverse");
strcpy(&(Menu[MNU_EFFECT].String[4][0]),"Playback
      Rate");
strcpy(&(Menu[MNU_EFFECT].Tip[0][0]),"Reduce
      volume with increasing time");
strcpy(&(Menu[MNU_EFFECT].Tip[1][0]),"Increase
      volume with increasing time");
strcpy(&(Menu[MNU_EFFECT].Tip[2][0])," ");
strcpy(&(Menu[MNU_EFFECT].Tip[3][0]),"Reverse
      the wave file");
strcpy(&(Menu[MNU_EFFECT].Tip[4][0]),"Vary
      the Playack Rate");
Menu[MNU_EFFECT].subMenu[0]=MNU_FADEIN;
Menu[MNU_EFFECT].subMenu[1]=MNU_FADEOUT;
Menu[MNU_EFFECT].AtX=11;Menu[MNU_EFFECT].
      AtY=2;
// The OPERATION menu option
Menu[MNU_OPERATION].nextMenu=MNU_FILE;
Menu[MNU_OPERATION].prevMenu=MNU_EFFECT;
Menu[MNU_OPERATION].Child=FALSE;
Menu[MNU_OPERATION].num_items=3;
for(i=0;i<3;i++)
{
Menu[MNU_OPERATION].Enabled[i]=FALSE;
Menu[MNU_OPERATION].subMenu[i]=NONE;
Menu[MNU_OPERATION].String[i]=
      (char*)malloc(15);
Menu[MNU_OPERATION].Tip[i]=
      (char*)malloc(50);
Menu[MNU_OPERATION].OptionID[i]=21+i;
}
strcpy(&(Menu[MNU_OPERATION].String[0][0]),"Play");
strcpy(&(Menu[MNU_OPERATION].String[1]
      [0]),"-");
strcpy(&(Menu[MNU_OPERATION].String[2]
      [0]),"Record");
strcpy(&(Menu[MNU_OPERATION].Tip[0][0]),"Play
      the file that was opened");
strcpy(&(Menu[MNU_OPERATION].Tip[1]
      [0]),"-");
strcpy(&(Menu[MNU_OPERATION].Tip[2][0]),
      "Record sound through the microphone");
Menu[MNU_OPERATION].AtX=23;Menu
      [MNU_OPERATION].AtY=2;
// The FADE-IN menu option
Menu[MNU_FADEIN].nextMenu=Menu
      [MNU_FADEIN].prevMenu=NONE;
Menu[MNU_FADEIN].Child=TRUE;
Menu[MNU_FADEIN].num_items=2;
for(i=0;i<2;i++)
{
Menu[MNU_FADEIN].Enabled[i]=FALSE;
Menu[MNU_FADEIN].subMenu[i]=NONE;
Menu[MNU_FADEIN].String[i]=
      (char *)malloc(15);
Menu[MNU_FADEIN].Tip[i]=(char *)malloc(50);
Menu[MNU_FADEIN].OptionID[i]=31+i;
}
strcpy(&(Menu[MNU_FADEIN].String[0][0]),"Linear");
strcpy(&(Menu[MNU_FADEIN].String[1][0]),"
      Exponential");
strcpy(&(Menu[MNU_FADEIN].Tip[0][0]),"Apply
      Linear attenuation or amplification");
strcpy(&(Menu[MNU_FADEIN].Tip[1][0]),"Apply
      Exponential attenuation or amplification");
Menu[MNU_FADEIN].AtX=33;Menu
      [MNU_FADEIN].AtY=2;
// The FADE-OUT menu option
Menu[MNU_FADEOUT].nextMenu=Menu
      [MNU_FADEOUT].prevMenu=NONE;
Menu[MNU_FADEOUT].Child=TRUE;
Menu[MNU_FADEOUT].num_items=2;
for(i=0;i<2;i++)
{
Menu[MNU_FADEOUT].Enabled[i]=FALSE;
Menu[MNU_FADEOUT].subMenu[i]=NONE;
Menu[MNU_FADEOUT].String[i]=
      (char *)malloc(15);
Menu[MNU_FADEOUT].Tip[i]=
      (char*)malloc(50);
Menu[MNU_FADEOUT].OptionID[i]=41+i;
}
strcpy(&(Menu[MNU_FADEOUT].String[0][0]),
      "Linear");
strcpy(&(Menu[MNU_FADEOUT].String[1][0]),
      "Exponential");
strcpy(&(Menu[MNU_FADEOUT].Tip[0][0]),"Apply
      Linear attenuation or amplification");
strcpy(&(Menu[MNU_FADEOUT].Tip[1][0]),"Apply
      Exponential attenuation or amplification");
Menu[MNU_FADEOUT].AtX=33;Menu
      [MNU_FADEOUT].AtY=2;
void RemoveMenu(int MenuID)
{
int i,j;
textbackground(BLUE);textcolor(WHITE);
gotoxy(Menu[MenuID].AtX,Menu[MenuID].AtY);
for(i=0;i<30;i++) printf("%c",205);
for(i=1;i<=Menu[MenuID].num_items+2;i++)
{
gotoxy(Menu[MenuID].AtX,Menu[MenuID].AtY+i);
for(j=0;j<30;j++) printf(" ");
}
return;
}
int ShowMenu(int MenuID)
{
MENU *menu;
int *subMenu;
int nextMenu, prevMenu;
char **String, **Tip;
int *OptionID;
BOOL *Enabled;
char IsChild;
int num_items;
int longLength,length;
int StartX,StartY;
int i,j;
int CurSelect=0,ch,RetVal;
menu=&(Menu[MenuID]);
num_items=menu->num_items;
String=menu->String;
nextMenu=menu->nextMenu;
prevMenu=menu->prevMenu;
subMenu=menu->subMenu;
IsChild=menu->Child;
OptionID=menu->OptionID;
Tip=menu->Tip;
Enabled=menu->Enabled;
StartX=menu->AtX;
StartY=menu->AtY;
longLength=strlen(String[0]);
if(subMenu[0]!=NULL) longLength+=3;
for(i=1;i<num_items;i++)
{
length=strlen(String[i]);
if(subMenu[i]!=NULL) length+=3;
if(length>longLength) longLength=length;
}
textbackground(LIGHTGRAY);textcolor(WHITE);
for(i=StartY;i<StartY+num_items+2;i++)
{
gotoxy(StartX,i);printf(" ");
gotoxy(StartX+longLength+5,i);printf(" ");
}
StartX++;
gotoxy(StartX,StartY);printf("%c",218);
for(i=0;i<longLength+2;i++) printf("%c",196);
printf("%c",191);
gotoxy(StartX,num_items+StartY+1);printf("%c",192);
for(i=0;i<longLength+2;i++) printf("%c",196);
printf("%c",217);
for(i=0;i<num_items;i++)
{
if(String[i][0]!='-')
{
textcolor(WHITE);
gotoxy(StartX,StartY+i+1);printf("%c ",179);
if(Enabled[i])
textcolor(BLACK);
else
textcolor(BROWN);
gotoxy(StartX+2,StartY+i+1);
for(j=0;j<longLength+1;j++)
if(j<strlen(String[i]))
printf("%c",String[i][j]);
else
printf(" ");
textcolor(WHITE);
printf("%c",179);
}
else
{
textcolor(WHITE);
gotoxy(StartX,StartY+i+1);printf("%c",195);
for(j=0;j<longLength+2;j++) printf("%c",196);
printf("%c",180);
}
}
for(;;)
{
DisplayTip(Tip[CurSelect]);
textbackground(GREEN);
if(Enabled[CurSelect])
textcolor(BLACK);
else
textcolor(BROWN);
gotoxy(StartX+1,StartY+CurSelect+1);
printf(" ");
for(j=0;j<longLength+1;j++)
if(j<strlen(String[CurSelect]))
printf("%c",String[CurSelect][j]);
else
printf(" ");
ch=getch();
if(ch==0) ch=getch();
ch+=300;
switch(ch)
{
case ESCAPE:
RemoveMenu(MenuID);
return(-1);
case ENTER:
RemoveMenu(MenuID);
if(Enabled[CurSelect]==TRUE)
return(OptionID[CurSelect]);
else
return(-1);
case LEFT_ARROW:
if(IsChild==TRUE)
{

```

```

RemoveMenu(MenuID);
return(0);
}
else
{
if(prevMenu!=NONE)
{
RemoveMenu(MenuID);
return(ShowMenu(prevMenu));
}
}
break;
case RIGHT_ARROW:
if(subMenu[CurSelect]!=NONE)
{
RetVal=ShowMenu(subMenu[CurSelect]);
if(RetVal!=0)
{
RemoveMenu(MenuID);
return(RetVal);
}
}
else
{
if(nextMenu!=NONE)
{
RemoveMenu(MenuID);
return(ShowMenu(nextMenu));
}
}
break;
case DOWN_ARROW:
textbackground(LIGHTGRAY);
if(Enabled[CurSelect])
textcolor(BLACK);
else
textcolor(BROWN);
gotoxy(StartX+1,StartY+CurSelect+1);
printf(" ");
for(j=0;j<longLength+1;j++)
if(j<strlen(String[CurSelect]))
printf("%c",String[CurSelect][j]);
else
printf(" ");
CurSelect++;
if(CurSelect==num_items) CurSelect=0;
while(String[CurSelect][0]==' ')
{
if(CurSelect==num_items)
CurSelect=0;
else
CurSelect++;
}
break;
case UP_ARROW:
textbackground(LIGHTGRAY);
if(Enabled[CurSelect])
textcolor(BLACK);
else
textcolor(BROWN);
gotoxy(StartX+1,StartY+CurSelect+1);
printf(" ");
for(j=0;j<longLength+1;j++)
if(j<strlen(String[CurSelect]))
printf("%c",String[CurSelect][j]);
else
printf(" ");
CurSelect--;
if(CurSelect<0) CurSelect=num_items-1;
while(String[CurSelect][0]==' ')
{
if(CurSelect<0)
CurSelect=num_items-1;
else
CurSelect--;
}
break;
}
}

}
void ButtonDisplay(int x1,int y1,char state
char *caption)
{
text_info tinfo;
gettextinfo(&tinfo);
int i;
if(state==ENABLE_NOTACTIVE) textcolor
(YELLOW);
if(state==ENABLE_ACTIVE) textcolor(WHITE);
if(state==DISABLE) textcolor(LIGHTGRAY);
textbackground(CYAN);
gotoxy(x1,y1);printf(" %s ",caption);
textbackground(LIGHTGRAY);textcolor(YELLOW);
printf("%c",220);
gotoxy(x1+1,y1+1);for(i=0;i<8;i++)printf("%c",223);
textattr(tinfo.attribute);
}
void ButtonPushed(int x1,int y1,char *caption)
{
text_info tinfo;
gettextinfo(&tinfo);
int i;
textbackground(LIGHTGRAY);textcolor(WHITE);
gotoxy(x1,y1);printf(" ");
gotoxy(x1,y1+1);printf(" ");
textbackground(CYAN);
gotoxy(x1+1,y1);printf(" %s ",caption);
delay(250);
gotoxy(x1,y1);printf(" %s ",caption);
textbackground(LIGHTGRAY);textcolor(YELLOW);
printf("%c",220);
gotoxy(x1,y1+1);printf(" ");for(i=0;i<8;i++)
printf("%c",223);
textattr(tinfo.attribute);
}
BOOL DisplayDialog(char mode)
{
int Control=0,ch;
int x=29,y=5,i=0,N=0;
char TempStr[40];TempStr[0]=0;
switch(mode)
{
case FILE_OPEN: Window(10,3,70,9,"Open
File",LIGHTGRAY,YELLOW);break;
case FILE_SAVE: Window(10,3,70,9,"Save
File",LIGHTGRAY,YELLOW);break;
case PLAYBACK_RATE: Window(10,3,70,9,"
Playback Rate",LIGHTGRAY,YELLOW);break;
}
ButtonDisplay(25,7,ENABLE_NOTACTIVE,"
Ok ");
ButtonDisplay(45,7,ENABLE_NOTACTIVE,"Cancel");
textbackground(LIGHTGRAY);textcolor(YELLOW);
gotoxy(13,5);
if(mode==FILE_OPEN || mode==FILE_SAVE)
{
printf("Enter Filename: ");
strcpy(TempStr,sFileName);
N=39;
}
else
{
printf("Playback Rate : ");
strcpy(TempStr,sPlayBackRate);
N=5;
}
textbackground(BLUE);textcolor(WHITE);
printf(" ");
gotoxy(29,5);printf("%s",TempStr);
i=strlen(TempStr);
x+=i;
for(;;)
{
switch(Control)
{
case 0:
_setcursortype(_NORMALCURSOR);
textbackground(BLUE);textcolor(WHITE);
ButtonDisplay(45,7,ENABLE_NOTACTIVE,"Cancel");
gotoxy(x,y);
break;
case 1:
_setcursortype(_NOCURSOR);
ButtonDisplay(25,7,ENABLE_ACTIVE," Ok ");
break;
case 2:
ButtonDisplay(45,7,ENABLE_ACTIVE,"Cancel");
ButtonDisplay(25,7,ENABLE_NOTACTIVE,"
Ok ");
break;
}
ch=getch();
if(ch==0) ch=getch()+300;
ch+=300;
switch(ch)
{
case TAB:
Control=(++Control)%3;
break;
case ESCAPE:
_setcursortype(_NOCURSOR);
ButtonPushed(45,7,"Cancel");
ch=1; Control=2;
break;
case ENTER:
_setcursortype(_NOCURSOR);
ButtonPushed(25,7," Ok ");
ch=1;Control=1;
break;
case SPACE:
if(Control==2){_setcursortype(_NOCURSOR);
ButtonPushed(45,7,"Cancel");ch=1;}
if(Control==1){_setcursortype(_NOCURSOR);
ButtonPushed(25,7," Ok ");ch=1;}
break;
case BACK_SPACE:
if(Control==0 && i>0)
{
gotoxy(--x,y);
printf(" ");
i--;
TempStr[i]=0;
gotoxy(29,5);
printf("%s",TempStr);
}
break;
default:
ch-=300;
if(ch<300 && i<N)
{
TempStr[i++]=(char)ch;
TempStr[i]=0;
gotoxy(29,5);
printf("%s",TempStr);
x++;
}
break;
}
if(ch==1) break;
}
textbackground(BLUE);textcolor(WHITE);
for(ch=3;ch<=9;ch++)
}
gotoxy(10,ch);
for(i=10;i<=70;i++)
printf(" ");
}
if(Control==1)
{
if(mode==FILE_SAVE || mode==FILE_OPEN)
strcpy(sFileName,TempStr);
if(mode==PLAYBACK_RATE)strcpy(sPlayBackRate,
TempStr);
return(TRUE);
}
return(FALSE);
}

```



```

        OUTPUT;dlen=NoSamples;
    BOOL bits16=FALSE;
    if(wave.fmt.FormatSpecific==BITS16) bits16=
        TRUE;
    if(bits16==FALSE)
    while(dlen>0)
    {
        fread(array,1,nChannels,fsource);
        switch(nChannels)
        {
        case 1: fputc((int)array[0],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        fputc((int)array[0],fdest);
        break;
        case 2: fputc((int)array[0],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        fputc((int)array[1],fdest);
        break;
        case 3: fputc((int)array[0],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        fputc((int)array[1],fdest);
        break;
        case 4: fputc((int)array[0],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        fputc((int)array[2],fdest);
        break;
        case 6: fputc((int)array[1],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        fputc((int)array[4],fdest);
        break;
        }
        dlen--;
    }
    else
    {
        NoSamples/=2;
        dlen=NoSamples;
        while(dlen>0)
        {
            fread(arrayi,2,nChannels,fsource);
            switch(nChannels)
            {
            case 1: array[0]=(char)((long)(arrayi[0]+
                32768)*255/65535);
                fputc((int)array[0],fdest);
                if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
                fputc((int)array[0],fdest);
                break;
            case 2: array[0]=(char)((long)(arrayi[0]+
                32768)*255/65535);
                fputc((int)array[0],fdest);
                if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
                {
                    array[1]=(char)((long)(arrayi[1]+32768)*255/
                        65535);
                    fputc((int)array[1],fdest);
                }
                break;
            case 3: array[0]=(char)((long)(arrayi[0]+
                32768)*255/65535);
                fputc((int)array[0],fdest);
                if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
                {
                    array[1]=(char)((long)(arrayi[1]+32768)*255/
                        65535);
                    fputc((int)array[1],fdest);
                }
            }
        }
    }

```

```

        break;
    case 4: array[0]=(char)((long)(arrayi[0]+
        32768)*255/65535);
        fputc((int)array[0],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        {
            array[2]=(char)((long)(arrayi[2]+32768)*255/
                65535);
            fputc((int)array[2],fdest);
        }
        break;
    case 6: array[1]=(char)((long)(arrayi[1]+
        32768)*255/65535);
        fputc((int)array[1],fdest);
        if(TYPE_OF_OUTPUT==STEREO_OUTPUT)
        {
            array[4]=(char)((long)(arrayi[4]+32768)*255/
                65535);
            fputc((int)array[4],fdest);
        }
        break;
    }
    dlen--;
}
fclose(fsouce);
fclose(fdest);
return TRUE;
}
else
{
    printf("\a");
    DisplayTip("The file is not available!");
    return FALSE;
}
}

```

**GLOBALS.H**

```

#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <conio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#define FALSE 0
#define TRUE 1
#define ENABLE_ACTIVE 1
#define ENABLE_NOTACTIVE 2
#define DISABLE 0
#define NONE -1
#define MNU_FILE 0
#define MNU_EFFECT 1
#define MNU_OPERATION 2
#define MNU_FADEIN 3
#define MNU_FADEOUT 4
#define FILE_OPEN 1
#define FILE_SAVE 2
#define FILE_EXIT 4
#define FADEIN_LINEAR 31
#define FADEIN_EXP 32
#define FADEOUT_LINEAR 41
#define FADEOUT_EXP 42
#define REVERSE 14
#define PLAYBACK_RATE 15
#define PLAY 21
#define RECORD 22

```

```

#define AltE 318
#define AltF 333
#define AltO 324
#define AltX 345
#define LEFT_ARROW 375
#define RIGHT_ARROW 377
#define UP_ARROW 372
#define DOWN_ARROW 380
#define ESCAPE 327
#define ENTER 313
#define SPACE 332
#define BACK_SPACE 308
#define TAB 309
#define PCM 1
#define IN 0
#define OUT 1
#define LINEAR 0
#define EXPONENTIAL 1
#define FADEIN 0
#define FADEOUT 1
#define DATA_OUT 0x378
#define BITS16 16
#define BITS8 8
#define STEREO_OUTPUT 2
#define MONO_OUTPUT 1
typedef char BOOL;
typedef struct{
    char rID[4];
    unsigned long rLen;
}RIFF;
typedef struct{
    char fID[4];
    unsigned long fLen;
    unsigned int wFormatTag;
    unsigned int nChannels;
    unsigned long nSamplesPerSec;
    unsigned long nAvgBytesPerSec;
    unsigned int nBlockAlign;
    unsigned int FormatSpecific;
}FORMATCHUNK;
typedef struct{
    char wID[4];
    FORMATCHUNK fmt;
}WAVE;
typedef struct{
    char dID[4];
    unsigned long dLen;
}DATA;
struct MENU
{
    int subMenu[10];
    char *Tip[10];
    char *String[10];
    int OptionID[10];
    BOOL Enabled[10];
    int num_items;
    char Child;
    int AtX,AtY;
    int nextMenu;
    int prevMenu;
} Menu[5];
long RateOfPlayBack=15000,PBR;
long double NoSamples=76455;
double SamplingFrequency=44000;
char sFileName[40];
char sPlayBackRate[6];

```