

```

***** D S P . P A S *****
{
  -----*
  Task      : Demo Program for demonstration of DSP
              Programming
  -----*
  Author     : Michael Tischer / Bruno Jennrich
  Developed on : 03/20/1994
  Last update : 04/5/1995
  *****
}

{$m 16000,16000,16000}

{$A-}                { no word alignment of structures }
{$X+}                { Function results optional }

Uses ARGs,DSPUTIL,SBUTIL,MIXUTIL,DMAUTIL,DOS;

Const
  DSP_NAME = 'DSP';
  BUFSIZE  = 8192;
  ERROR    = $ffff;
  NO_ERROR = 0;

var
  SBB      : SBBASE;
  lFrequency : Longint;
  iADC,iST  : Boolean;
  iDuration,
  iSource,iOR : Integer;
  CurFile    : Integer; { File access through DOS }
  lpRecordSource : string;
  FileNames  : NArgStrings;
  NumFiles   : integer;
  lpBuffer   : pointer;
  uMemSize, uFrg : Word;
  dspHeader  : DSPRECPLAY;

{ ***** }
{
  -----*
  M A I N   P R O G R A M
  -----*
}
{ ***** }

Begin
  if sb_GetEnviron( SBB, GetEnv( 'BLASTER' ) ) <> NO_ERROR then
  Begin
    Writeln( 'BLASTER environment variable not available' );
    Halt( 0 );
  End;

  if dsp_SetBase( SBB, TRUE ) = NO_ERROR then
  Begin
    NumFiles := GetNArg( '-', FileNames );

    if NumFiles = 0 then
    Begin
      Writeln('Call:',DSP_NAME,' [-bX][-dX][-fX][-r][-s][-w] Filename');
      Writeln;
      Writeln( '-bX   : DMA buffer size ' );
      Writeln( '-dX   : Length of recording in seconds' );
      Writeln( '-fX   : Frequency in Hz' );
      Writeln( '-r    : Recording (else playback)' );
      Writeln( '-s    : Stereo (else mono)' );
      Writeln( '-w    : 16 bit transfer (Word) ' );
      Writeln( '-iMIC/CD/LINE : Recording source (Input)' );
      Halt( 0 );
    End;

    {-- filenames were entered in the command line -----}

    iADC := GetArg( '-r', _none, NIL, 0 ) <> 0; { Recording (Record) }

    dspHeader.iStereo := FALSE;                { Set defaults }
    dspHeader.uFrequency := 10000;
    dspHeader.iBits := 8;

    iOR := dsp_FileOpen( FileNames[0], CurFile );
    if not iADC then
    if iOR=0 then
      dsp_ReadHeader( CurFile , dspHeader )
    else
    Begin
      Writeln( 'File ', FileNames[0], ' not found!' );
      Halt( 0 );
    End;

    if GetArg( '-s', _none, NIL, 0 ) <> 0 then
      dspHeader.iStereo := TRUE;
  End;

```

```

lFrequency := dspHeader.uFrequency;
if GetArg( '-f', _long, @lFrequency, 1 ) <> 0 then { Frequency }
    dspHeader.uFrequency := Word ( lFrequency );

if SBB.uDspVersion >= DSP_4XX then
    if GetArg( '-w', _none, Nil, 0 ) <> 0 then
        dspHeader.iBits := 16;

if iADC then dsp_WriteHeader( CurFile, dspHeader );

iDuration := 10; { Duration of recording }
GetArg( '-d', _int, @iDuration, 1 );

uMemSize := BUFSIZE; { Memory available for DMA }
GetArg( '-b', _int, @uMemSize, 1 );

lpRecordSource := '';
GetArg( '-i', _string, @lpRecordSource, 1 );
iSource := -1;
if( Pos( Up( lpRecordSource ), 'MIC' ) = 1 ) then iSource := MIC
else
if( Pos( Up( lpRecordSource ), 'CD' ) = 1 ) then iSource := CD
else
if( Pos( Up( lpRecordSource ), 'LINE' ) = 1 ) then iSource := LINE;

sb_Print( SBB );
End
else
Begin
    Writeln( 'DSP control not possible!' );
    Halt( 0 );
End;

if ( ( dspHeader.iStereo ) and not dsp_CanStereo ) then
    Writeln( ' WARNING! Stereo mode NOT supported.' );
if dspHeader.iBits > dsp_MaxBits then
    Writeln( ' WARNING! 16 bit mode NOT supported!' );

uFrq := dspHeader.uFrequency;
iSt := dspHeader.iStereo;
dsp_AdjustFrq( uFrq, iADC, iSt );
if uFrq < dspHeader.uFrequency then
    Writeln( 'WARNING! Frequency too high! Adjusting!' );

uMemSize := uMemSize and not $0003;
{ uMemSize is now divisible by 4 }
lpBuffer := dma_AllocMem( uMemSize ); { Allocate DMA memory }
if lpBuffer <> NIL then
    Begin
        dsp_DoRecPlay( CurFile , iADC, iSource, dspHeader, iDuration,
            lpBuffer, uMemSize );
        dma_Free( lpBuffer ); { Free up memory }
    End
else
    Writeln( 'No more free memory!' );
dsp_FileClose( CurFile ); { Close file }
End.

```