

DSP.C

```

/*****
/*          D S P . C          */
/**-----**/
/* Task      : Demo programm for demonstration of DSP      */
/*          programming                                         */
/**-----**/
/* Author    : Michael Tischer / Bruno Jennrich             */
/* Developed on : 03/20/1994                                   */
/* Last update : 04/05/1995                                   */
/* Compiler Options: 1 byte struct alignment, no optimizations */
/* COMPILER: Borland C++ 3.1, Microsoft Visual C++ 1.5       */
/*****

/*- Add include files -----*/
#include <fcntl.h>
#include <dos.h>
#include <process.h>

#include "sbutil.h"
#include "dmautil.h"
#include "dsputil.h"
#include "mixutil.h"
#include "irqutil.h"
#include "args.h"

/*- Link "libraries" -----*/
#include "sbutil.c"
#include "dmautil.c"
#include "dsputil.c"
#include "mixutil.c"
#include "irqutil.c"

```

```

#include "args.c"

#define DSP_NAME "DSP"

SBBASE SBB;

#define BUFSIZE 8192U

/*****
/*          M A I N   P R O G R A M          */
*****/
VOID main( INT argc, PCHAR argv[] )
{
    PCHAR      pFile;
    LONG       lFrequency;
    INT        iADC, iDuration, iHandle;
    PCHAR      lpRecordSource;
    INT        iSource, iSt;
    LPBYTE     lpBuffer;
    WORD       uMemSize, uFrq;
    DSPRECPLAY dspHeader;

    if( sb_GetEnviron( &SBB, getenv( "BLASTER" ) ) != NO_ERROR )
    {
        printf("BLASTER environment variable not available/present\n");
        exit(0);
    }

    if( dsp_SetBase( &SBB, TRUE ) == NO_ERROR )
    {
        pFile = NULL;          /* Filename of file containing the data */
        GetNArg( argc, argv, "-", ( PCHAR *)&pFile, 1 );
    }
}

```

```

if( pFile == NULL )
{
    printf("Call: %s [-bX][-dX][-fX][-r][-s][-w] Filename\n",
           DSP_NAME );
    printf("\n-bX    : DMA buffer size \n");
    printf("-dX    : Length of recording in seconds\n");
    printf("-fX    : Frequency in Hz\n");
    printf("-r    : Recording (else playback)\n");
    printf("-s    : Stereo (else mono)\n");
    printf("-w    : 16 bit transfer (Word) \n");
    printf("-iMIC/CD/LINE : Recording source (Input)\n");
    exit(0);
}

iADC = 0;                                     /* Recording (Record) */
iADC = GetArg( argc, argv, "-r", _none, NULL, 0 );

dspHeader.iStereo = FALSE;                    /* Set defaults */
dspHeader.uFrequency = 10000;
dspHeader.iBits = 8;

dsp_FileOpen( pFile, iADC, &iHandle );
if( !iADC ) dsp_ReadHeader( iHandle, &dspHeader );

if( GetArg( argc, argv, "-s", _none, NULL, 0 ) )
    dspHeader.iStereo = TRUE;

lFrequency = dspHeader.uFrequency;           /* Frequency */
if( GetArg( argc, argv, "-f", _long, &lFrequency, 1 ) )
    dspHeader.uFrequency = ( WORD )lFrequency;

```

```

if( SBB.uDspVersion >= DSP_4XX )
    if( GetArg( argc, argv, "-w", _none, NULL, 0 ) )
        dspHeader.iBits = 16;

if( iADC )    dsp_WriteHeader( iHandle, &dspHeader );

iDuration = 10;                                /* Duration of recording */
GetArg( argc, argv, "-d", _int, &iDuration, 1 );

uMemSize = BUFSIZE;    /* Memory available for DMA */
GetArg( argc, argv, "-b", _int, &uMemSize, 1 );

lpRecordSource = NULL;
GetArg( argc, argv, "-i", _string, &lpRecordSource, 1 );
iSource = -1;
if( _fstrcmp( lpRecordSource, "MIC" ) == 0 )    iSource = MIC;
else
if( _fstrcmp( lpRecordSource, "CD" ) == 0 )    iSource = CD;
else
if( _fstrcmp( lpRecordSource, "LINE" ) == 0 )    iSource = LINE;

    sb_Print( &SBB );
}
else
{
    printf("DSP control not possible!");
    exit( 0 );
}

if( ( dspHeader.iStereo ) && ! dsp_CanStereo() )
    printf("WARNING! Stereo mode NOT supported.\n");
if( dspHeader.iBits > dsp_MaxBits() )

```

```

printf("WARNING! 16 bit mode NOT supported!\n");

uFrq = dspHeader.uFrequency;
iSt = dspHeader.iStereo;
dsp_AdjustFrq( &uFrq, iADC, &iSt );
if( uFrq < dspHeader.uFrequency )
    printf("WARNING! Frequency too high! Adjusting!\n");

uMemSize &= ~0x0003U;          /* uMemSize is now divisible by 4 */
lpBuffer = dma_AllocMem( uMemSize );    /* Allocate DMA memory */
if( lpBuffer )
{
    dsp_DoREPLAY( iHandle, iADC, iSource, &dspHeader, iDuration,
                  lpBuffer, uMemSize );
    dma_Free( lpBuffer );          /* Free memory */
}
else printf("No more free memory!\n");
dsp_FileClose( iHandle );          /* Close file */
}

```