

```

/*****
/*          M I X . C          */
**/
/* Task      : Tool for changing mixer settings. Current */
/*            settings are preserved.                    */
**/
/* Author     : Michael Tischer / Bruno Jennrich      */
/* Developed on : 03/20/1994                          */
/* Last update  : 04/06/1995                          */
**/
/* COMPILER    : Borland C++ 3.1, Microsoft Visual C++ 1.5 */
*****/

/*-- Add include files -----*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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

/*- Document the following lines within a project: -----*/
/*#include "sbutil.c"
#include "mixutil.c"
#define DSP_VERSIONONLY
#include "dsputil.c"
#include "irqutil.c"
#include "args.c"*/

#define MIX_NAME "MIX"                                /* For output in Help screen */

/*- Help variables -----*/
PCHAR onoff[2] = {"OFF", "ON"};                      /* For switch settings */
PCHAR pString;                                       /* For evaluation of parameter strings */
PCHAR pValue[ 7 ];
INT iNumStrings, iIdx;

/*****
/* Print_Mix3Settings : Output current status of CT1345. */
*****/
VOID Print_Mix3Settings( VOID )
{
    printf(" ADC Filter %s\n", mix3_GetADCFilter() ? "on" : "off" );
    printf(" DAC Filter %s\n", mix3_GetDACFilter() ? "on" : "off" );
    printf(" LowPass Filter: %s\n",
        mix3_GetADDACLowPass() ? "8.8 kHz" : "3.2 kHz" );
    printf(" DAC Stereo %s\n", mix3_GetDACStereo() ? "on" : "off" );
    printf(" SOURCE (DSP-input): " );
    switch( mix3_GetADCSource() )
    {
        case CD:    printf("CD\n");    break;
        case LINE:  printf("LINE\n");  break;
        case MIC:   printf("MIC\n");   break;
    }

    printf("\nVolumes:\n");
    printf(" MASTER: %d %d\n", mix3_GetVolume( MASTER_L ),
        mix3_GetVolume( MASTER_R ) );
    printf(" MIC      : %d\n", mix3_GetVolume( MIC ) );
    printf(" CD       : %d %d\n", mix3_GetVolume( CD_L ),
        mix3_GetVolume( CD_R ) );
    printf(" LINE     : %d %d\n", mix3_GetVolume( LINE_L ),
        mix3_GetVolume( LINE_R ) );
    printf(" VOICE    : %d %d\n", mix3_GetVolume( VOICE_L ),
        mix3_GetVolume( VOICE_R ) );
    printf(" MIDI     : %d %d\n", mix3_GetVolume( MIDI_L ),
        mix3_GetVolume( MIDI_R ) );
}

/*****
/* Print_Mix4Settings : Output current status of CT1745. */
*****/
VOID Print_Mix4Settings( VOID )

```

```

{
printf("Left SOURCE/ADC settings:\n");
printf(" MIC    CD    LINE  MIDI\n");
printf(" %s      %s %s %s %s %s %s\n",
    mix4_GetADCSourceL( MIC )    ? "X" : "-",
    mix4_GetADCSourceL( CD_L )   ? "X" : "-",
    mix4_GetADCSourceL( CD_R )   ? "X" : "-",
    mix4_GetADCSourceL( LINE_L ) ? "X" : "-",
    mix4_GetADCSourceL( LINE_R ) ? "X" : "-",
    mix4_GetADCSourceL( MIDI_L ) ? "X" : "-",
    mix4_GetADCSourceL( MIDI_R ) ? "X" : "-" );

printf("Right SOURCE/ADC settings:\n");
printf(" MIC    CD    LINE  MIDI\n");
printf(" %s      %s %s %s %s %s %s\n",
    mix4_GetADCSourceR( MIC )    ? "X" : "-",
    mix4_GetADCSourceR( CD_L )   ? "X" : "-",
    mix4_GetADCSourceR( CD_R )   ? "X" : "-",
    mix4_GetADCSourceR( LINE_L ) ? "X" : "-",
    mix4_GetADCSourceR( LINE_R ) ? "X" : "-",
    mix4_GetADCSourceR( MIDI_L ) ? "X" : "-",
    mix4_GetADCSourceR( MIDI_R ) ? "X" : "-" );

printf(" OUT settings:\n");
printf(" MIC    CD    LINE\n");
printf(" %s      %s %s %s %s\n",
    mix4_GetOUTSource( MIC )    ? "X" : "-",
    mix4_GetOUTSource( CD_L )   ? "X" : "-",
    mix4_GetOUTSource( CD_R )   ? "X" : "-",
    mix4_GetOUTSource( LINE_L ) ? "X" : "-",
    mix4_GetOUTSource( LINE_R ) ? "X" : "-" );

printf(" ADC gain %d, %d\n", mix4_GetADCGain( CH_LEFT ),
    mix4_GetADCGain( CH_RIGHT ) );
printf(" OUT gain %d, %d\n", mix4_GetOUTGain( CH_LEFT ),
    mix4_GetOUTGain( CH_RIGHT ) );
printf(" Automatic Gain Control %s\n", mix4_GetAGC() ? "on" : "off" );
printf(" Treble: %d, %d\n", mix4_GetTreble( CH_LEFT ),
    mix4_GetTreble( CH_RIGHT ) );
printf(" Bass: %d, %d\n", mix4_GetBass( CH_LEFT ),
    mix4_GetBass( CH_RIGHT ) );

printf("Volumes:\n");
printf(" MASTER : %d dB %d dB\n", mix4_GetVolume( MASTER_L ),
    mix4_GetVolume( MASTER_R ) );
printf(" MIC : %d\n", mix4_GetVolume( MIC ) );
printf(" CD : %d %d \n", mix4_GetVolume( CD_L ),
    mix4_GetVolume( CD_R ) );
printf(" LINE : %d dB %d dB\n", mix4_GetVolume( LINE_L ),
    mix4_GetVolume( LINE_R ) );
printf(" VOICE : %d dB %d dB\n", mix4_GetVolume( VOICE_L ),
    mix4_GetVolume( VOICE_R ) );
printf(" MIDI : %d dB %d dB\n", mix4_GetVolume( MIDI_L ),
    mix4_GetVolume( MIDI_R ) );
printf(" PCSPEAKER : %d\n", mix4_GetVolume( PCSPEAKER ) );

}

/*****
/* Process_Mix3 : Processing command line parameters to set */
/* mixer (DSP3.xx). */
/*-----*/
/* Input : argc : Argument Counter (command line) */
/* argv : Argument Values (Command line) */
*****/
VOID Process_Mix3( INT argc, PCHAR argv[] )
{
    INT iADCFilter, iDACFilter, iLowPass, iVol[2], iADCSource;

    if( argc == 1 )
    {
        printf("Call: %s [ -ADCFilter:ON/OFF]\n", MIX_NAME );
        printf("[ -DACFilter:ON/OFF][ -LowPass:3.2kHz/8.8kHz]\n");
        printf("[ -MIC:0-255][ -CD:0-255[,0-255]][ -LINE:0-255[,0-255]]\n");
        printf("[ -VOICE:0-255[,0-255]][ -MIDI:0-255[,0-255]]\n");
        printf("[ -MASTER:0-255[,0-255]][ -SOURCE:CD/LINE/MIC]\n");
        printf("[ /r ] = Reset mixer\n");
    }
}

```

```

    printf("[ /q ] = no output (Quiet)\n");
    return;
}

/* Change ADC filter setting: */
iADCFilter = mix3_GetADCFilter();
if( GetArg( argc, argv, "-ADCFilter:", _string, &pString, 1 ) == 1 )
{
    iIdx = FindString( onoff, pString, 2 ) - 1;
    iADCFilter = iIdx < 0 ? iADCFilter : iIdx;
    if( iIdx < 0 )
        printf("Invalid ADC filter setting [ON or OFF]\n");
}
mix3_SetADCFilter( iADCFilter );

/* Change DAC filter setting: */
iDACFilter = mix3_GetDACFilter();
if( GetArg( argc, argv, "-DAC filter:", _string, &pString, 1 ) == 1 )
{
    iIdx = FindString( onoff, pString, 2 ) - 1;
    iDACFilter = iIdx < 0 ? iDACFilter : iIdx;
    if( iIdx < 0 )
        printf("Invalid ADC filter settting [ON or OFF]\n");
}
mix3_SetDACFilter( iDACFilter );

/* Change low pass filter setting: */
iLowPass = mix3_GetADDACLowPass();
if( GetArg( argc, argv, "-LOWPASS:", _string, &pString, 1 ) == 1 )
{
    if( _fstricmp( "3.2kHz", pString ) == 0 ) iLowPass = FALSE;
    else
        if( _fstricmp( "8.8kHz", pString ) == 0 ) iLowPass = TRUE;
    else printf("Invalid low pass setting [3.2kHz or 8.8kHz]\n");
}
mix3_SetADDACLowPass( iLowPass );

/* Change microphone volume: */
iVol[L] = mix3_GetVolume( MIC );
GetArg( argc, argv, "-MIC:", _int, &iVol[L], 1 );
mix3_SetVolume( MIC, iVol[L], 0 );

/* Change CD volume: */
iVol[L] = mix3_GetVolume( CD_L );
iVol[R] = mix3_GetVolume( CD_R );
if( GetArg( argc, argv, "-CD:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix3_SetVolume( CD, iVol[L], iVol[R] );

/* Change LINE volume: */
iVol[L] = mix3_GetVolume( LINE_L );
iVol[R] = mix3_GetVolume( LINE_R );
if( GetArg( argc, argv, "-LINE:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix3_SetVolume( LINE, iVol[L], iVol[R] );

/* Change VOICE volume: */
iVol[L] = mix3_GetVolume( VOICE_L );
iVol[R] = mix3_GetVolume( VOICE_R );
if( GetArg( argc, argv, "-VOICE:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix3_SetVolume( VOICE, iVol[L], iVol[R] );

/* Change MIDI volume: */
iVol[L] = mix3_GetVolume( MIDI_L );
iVol[R] = mix3_GetVolume( MIDI_R );
if( GetArg( argc, argv, "-MIDI:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix3_SetVolume( MIDI, iVol[L], iVol[R] );

/* Change MASTER volume: */
iVol[L] = mix3_GetVolume( MASTER_L );
iVol[R] = mix3_GetVolume( MASTER_R );
if( GetArg( argc, argv, "-MASTER:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix3_SetVolume( MASTER, iVol[L], iVol[R] );

/* Set ADC input: */
iADCSource = mix3_GetADCSource();
if( GetArg( argc, argv, "-SOURCE:", _string, &pString, 1 ) == 1 )
{
    if( _fstricmp( "CD", pString ) == 0 ) iADCSource = CD;
    else

```

```

    if( _fstricmp( "LINE", pString ) == 0 ) iADCSrc = LINE;
    else
    if( _fstricmp( "MIC", pString ) == 0 ) iADCSrc = MIC;
    else printf("Invalid recording source [CD, LINE or MIC]\n");
}
mix3_SetADCSrc( iADCSrc );

/* Output current mixer setting? */
if( !GetArg( argc, argv, "/q", _none, NULL, 0 ) )
    Print_Mix3Settings();
}

/*****
/* Process_Mix4 : Processing command line parameters to set
/* mixer (DSP4.xx).
/*
/*-----*/
/* Input : argc : Argument Counter (Command line)
/* argv : Argument Values (Command line)
/*
*****/
VOID Process_Mix4( INT argc, PCHAR argv[] )
{
    INT iVol[2], iADC[2], iDAC[2],
        iTreble[ 2 ], iBass[ 2 ], iAGC,
        iADCGain[ 2 ], iOUTGain[ 2 ];

    if( argc == 1 )
    {
        printf("Guide %s\n", MIX_NAME );
        printf("[ -MIC:0-255][ -SPEAKER:0-255][ -CD:0-255[,0-255]]\n");
        printf("[ -LINE:0-255[,0-255]][ -VOICE:0-255[,0-255]]\n");
        printf("[ -MIDI:0-255[,0-255]][ -MASTER:0-255[,0-255]]\n");
        printf("[ -ADC_L:[ MIDI_L:0/1]      3  [-ADC_R:[ MIDI_L:0/1]\n");
        printf("          [,MIDI_R:0/1]      3  [,MIDI_R:0/1]\n");
        printf("          [,MIDI:0/1]      3  [,MIDI:0/1]\n");
        printf("          [,LINE_L:0/1]      3  [,LINE_L:0/1]\n");
        printf("          [,LINE_R:0/1]      3  [,LINE_R:0/1]\n");
        printf("          [,LINE:0/1]      3  [,LINE:0/1]\n");
        printf("          [,CD_L:0/1]      3  [,CD_L:0/1]\n");
        printf("          [,CD_R:0/1]      3  [,CD_R:0/1]\n");
        printf("          [,CD:0/1]      3  [,CD:0/1]\n");
        printf("          [,MIC:0/1]      3  [,MIC:0/1]\n");
        printf("[ -OUT_L:[LINE_L:0/1][,LINE_R:0/1][,LINE:0/1]\n");
        printf("          [,CD_L:0/1][,CD_R:0/1][,CD:0/1][,MIC:0/1]\n");
        printf("[ -TREBLE:0-15[,0-15]][ -BASS:0-15[,0-15]]\n");
        printf("[ -ADCGAIN:0-3[,0-3]][ -OUTGAIN:0-3[,0-3]][-AGC:ON/OFF]\n");
        printf(" [/r] = Reset mixer\n");
        printf(" [/q] = No output (Quiet)\n\n");
        printf("Sample call: %s -MIC:255 -CD:128,255 -LINE:255\n",
            MIX_NAME);
        printf("-ADC_L:CD_L:1,CD_R:0 -ADC_R:CD_L:0,CD_R:1\n");
        return;
    }

    /* Set microphone volume : */
    iVol[L] = mix4_GetVolume( MIC );
    GetArg( argc, argv, "-MIC:", _int, &iVol[L], 1 );
    mix4_SetVolume( MIC, iVol[L], 0 );

    /* Set PC speaker Volume : */
    iVol[L] = mix4_GetVolume( PCSPEAKER );
    GetArg( argc, argv, "-SPEAKER:", _int, &iVol[L], 1 );
    mix4_SetVolume( PCSPEAKER, iVol[L], 0 );

    /* Set CD volume : */
    iVol[L] = mix4_GetVolume( CD_L );
    iVol[R] = mix4_GetVolume( CD_R );
    if( GetArg( argc, argv, "-CD:", _int, &iVol, 2 ) == 1 )
        iVol[R] = iVol[L];
    mix4_SetVolume( CD, iVol[L], iVol[R] );

    /* Set LINE volume : */
    iVol[L] = mix4_GetVolume( LINE_L );
    iVol[R] = mix4_GetVolume( LINE_R );
    if( GetArg( argc, argv, "-LINE:", _int, &iVol, 2 ) == 1 )
        iVol[R] = iVol[L];
    mix4_SetVolume( LINE, iVol[L], iVol[R] );

    /* Set VOICE volume : */
    iVol[L] = mix4_GetVolume( VOICE_L );
    iVol[R] = mix4_GetVolume( VOICE_R );
    if( GetArg( argc, argv, "-VOICE:", _int, &iVol, 2 ) == 1 )
        iVol[R] = iVol[L];

```

```

mix4_SetVolume( VOICE, iVol[L], iVol[R] );

/* Set MIDI volume : */
iVol[L] = mix4_GetVolume( MIDI_L );
iVol[R] = mix4_GetVolume( MIDI_R );
if( GetArg( argc, argv, "-MIDI:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix4_SetVolume( MIDI, iVol[L], iVol[R] );

/* Set MASTER volume : */
iVol[L] = mix4_GetVolume( MASTER_L );
iVol[R] = mix4_GetVolume( MASTER_R );
if( GetArg( argc, argv, "-MASTER:", _int, &iVol, 2 ) == 1 )
    iVol[R] = iVol[L];
mix4_SetVolume( MASTER, iVol[L], iVol[R] );

/* Set input sources for left ADC : */
iNumStrings = GetArg( argc, argv, "-ADC_L:",
    _string, &pValue, 7 );

iADC[L] = mix4_GetADCSourceL( MIDI_L );
iADC[R] = mix4_GetADCSourceL( MIDI_R );
if( FindString( pValue, "MIDI_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "MIDI_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "MIDI_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "MIDI_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "MIDI:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "MIDI:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceL( MIDI_L, iADC[L] );
mix4_SetADCSourceL( MIDI_R, iADC[R] );

iADC[L] = mix4_GetADCSourceL( LINE_L );
iADC[R] = mix4_GetADCSourceL( LINE_R );
if( FindString( pValue, "LINE_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "LINE_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "LINE_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "LINE_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "LINE:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "LINE:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceL( LINE_L, iADC[L] );
mix4_SetADCSourceL( LINE_R, iADC[R] );

iADC[L] = mix4_GetADCSourceL( CD_L );
iADC[R] = mix4_GetADCSourceL( CD_R );
if( FindString( pValue, "CD_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "CD_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "CD_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "CD_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "CD:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "CD:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceL( CD_L, iADC[L] );
mix4_SetADCSourceL( CD_R, iADC[R] );

iADC[L] = mix4_GetADCSourceL( MIC );
if( FindString( pValue, "MIC:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "MIC:1", iNumStrings ) ) iADC[L]=TRUE;
mix4_SetADCSourceL( MIC, iADC[L] );

/* Set input sources for right ADC : */
iNumStrings = GetArg( argc, argv, "-ADC_R:",
    _string, &pValue, 7 );

iADC[L] = mix4_GetADCSourceR( MIDI_L );
iADC[R] = mix4_GetADCSourceR( MIDI_R );
if( FindString( pValue, "MIDI_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "MIDI_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "MIDI_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "MIDI_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "MIDI:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "MIDI:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceR( MIDI_L, iADC[L] );
mix4_SetADCSourceR( MIDI_R, iADC[R] );

```

```

iADC[L] = mix4_GetADCSourceR( LINE_L );
iADC[R] = mix4_GetADCSourceR( LINE_R );
if( FindString( pValue, "LINE_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "LINE_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "LINE_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "LINE_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "LINE:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "LINE:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceR( LINE_L, iADC[L] );
mix4_SetADCSourceR( LINE_R, iADC[R] );

iADC[L] = mix4_GetADCSourceR( CD_L );
iADC[R] = mix4_GetADCSourceR( CD_R );
if( FindString( pValue, "CD_L:0", iNumStrings ) ) iADC[L]=FALSE;
if( FindString( pValue, "CD_R:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "CD_L:1", iNumStrings ) ) iADC[L]=TRUE;
if( FindString( pValue, "CD_R:1", iNumStrings ) ) iADC[R]=TRUE;
if( FindString( pValue, "CD:0", iNumStrings ) )
    iADC[L]=iADC[R]=FALSE;
if( FindString( pValue, "CD:1", iNumStrings ) )
    iADC[L]=iADC[R]=TRUE;
mix4_SetADCSourceR( CD_L, iADC[L] );
mix4_SetADCSourceR( CD_R, iADC[R] );

iADC[R] = mix4_GetADCSourceR( MIC );
if( FindString( pValue, "MIC:0", iNumStrings ) ) iADC[R]=FALSE;
if( FindString( pValue, "MIC:1", iNumStrings ) ) iADC[R]=TRUE;
mix4_SetADCSourceR( MIC, iADC[R] );

/* Set output sources : */
iNumStrings = GetArg( argc, argv, "-OUT:",
    _string, &pValue, 7 );

iDAC[L] = mix4_GetOUTSource( LINE_L );
iDAC[R] = mix4_GetOUTSource( LINE_R );
if( FindString( pValue, "LINE_L:0", iNumStrings ) )
    iDAC[L] = FALSE;
if( FindString( pValue, "LINE_R:0", iNumStrings ) )
    iDAC[R] = FALSE;
if( FindString( pValue, "LINE_L:1", iNumStrings ) )
    iDAC[L] = TRUE;
if( FindString( pValue, "LINE_R:1", iNumStrings ) )
    iDAC[R] = TRUE;
if( FindString( pValue, "LINE:0", iNumStrings ) )
    iDAC[L] = iDAC[R] = FALSE;
if( FindString( pValue, "LINE:1", iNumStrings ) )
    iDAC[L] = iDAC[R] = TRUE;
mix4_SetOUTSource( LINE_L, iDAC[L] );
mix4_SetOUTSource( LINE_R, iDAC[R] );

iDAC[L] = mix4_GetOUTSource( CD_L );
iDAC[R] = mix4_GetOUTSource( CD_R );
if( FindString( pValue, "CD_L:0", iNumStrings ) )
    iDAC[L] = FALSE;
if( FindString( pValue, "CD_R:0", iNumStrings ) )
    iDAC[R] = FALSE;
if( FindString( pValue, "CD_L:1", iNumStrings ) )
    iDAC[L] = TRUE;
if( FindString( pValue, "CD_R:1", iNumStrings ) )
    iDAC[R] = TRUE;
if( FindString( pValue, "CD:0", iNumStrings ) )
    iDAC[L] = iDAC[R] = FALSE;
if( FindString( pValue, "CD:1", iNumStrings ) )
    iDAC[L] = iDAC[R] = TRUE;
mix4_SetOUTSource( CD_L, iDAC[L] );
mix4_SetOUTSource( CD_R, iDAC[R] );

iDAC[L] = mix4_GetOUTSource( MIC );
if( FindString( pValue, "MIC:0", iNumStrings ) )
    iDAC[L] = FALSE;
if( FindString( pValue, "MIC:1", iNumStrings ) )
    iDAC[L] = TRUE;
mix4_SetOUTSource( MIC, iDAC[L] );

```

