```
//------------------------------------------------------------------------//
//                                                                        //
//   Name:  Talkthrough for the ADSP-BF533 EZ-KIT Lite                    //
//                                                                        //
//------------------------------------------------------------------------//
//                                                                        //
//  (C) Copyright 2006 - Analog Devices, Inc.  All rights reserved.       //
//                                                                        //
//  Project Name:   BF533 C Talkthrough TDM                               //
//                                                                        //
//  Date Modified:  04/03/03        Rev 1.0                         //
//                                                                        //
//  Software:       VisualDSP++4.5                                        //
//                                                                        //
//  Hardware:       ADSP-BF533 EZ-KIT Board                               //
//                                                                        //
//  Connections:    Disconnect RSCLK0 and TSCLK0 (Turn SW9 pin 6 OFF)     //
//                  Disconnect RFS0 and TFS0 (Turn SW9 pin 5 OFF)         //
//                  Connect an input source (such as a radio) to the Audio //
//                  input jack and an output source (such as headphones) to //
//                  the Audio output jack                                 //
//                                                                        //
//  Purpose:        This program sets up the SPI port on the ADSP-BF533 to //
//                  configure the AD1836 codec.  The SPI port is disabled //
//                  after initialization.  The data to/from the codec are //
//                  transfered over SPORT0 in TDM mode                    //
//                                                                        //
//------------------------------------------------------------------------//

#include "Talkthrough.h"
#include "sysreg.h"
#include "ccblkfn.h"
#include "stats.h"
#include "stdio.h"


//------------------------------------------------------------------------//
// Variables                                                              //
//                                                                        //
// Description: The variables iChannelxLeftIn and iChannelxRightIn contain //
//              the data coming from the codec AD1836.  The (processed)   //
//              playback data are written into the variables              //
//              iChannelxLeftOut and iChannelxRightOut respectively, which //
//              are then sent back to the codec in the SPORT0 ISR.        //
//              The values in the array iCodec1836TxRegs can be modified to //
//              set up the codec in different configurations according to //
//              the AD1885 data sheet.                                    //
//------------------------------------------------------------------------//
// left input data from ad1836
int iChannel0LeftIn, iChannel1LeftIn;
// right input data from ad1836
int iChannel0RightIn, iChannel1RightIn;
// left ouput data for ad1836
int iChannel0LeftOut, iChannel1LeftOut;
// right ouput data for ad1836
int iChannel0RightOut, iChannel1RightOut;
// array for registers to configure the ad1836
// names are defined in "Talkthrough.h"
volatile short sCodec1836TxRegs[CODEC_1836_REGS_LENGTH] =
{
                    DAC_CONTROL_1   | 0x000,
                    DAC_CONTROL_2   | 0x000,
                    DAC_VOLUME_0    | 0x3ff,
                    DAC_VOLUME_1    | 0x3ff,
                    DAC_VOLUME_2    | 0x3ff,
                    DAC_VOLUME_3    | 0x3ff,
                    DAC_VOLUME_4    | 0x3ff,
                    DAC_VOLUME_5    | 0x3ff,
                    ADC_CONTROL_1   | 0x000,
                    ADC_CONTROL_2   | 0x180,
                    ADC_CONTROL_3   | 0x000

};
```

```
// SPORT0 DMA transmit buffer
volatile int iTxBuffer1[8];
// SPORT0 DMA receive buffer
volatile int iRxBuffer1[8];


int yn = 0;
int xn = 0;
int num1 = 0;
int num2 = 0;
int playFlag = 1;
int recFlag = 1;
int doX = 0;

short sound[INPUTLEN] = {
#include "Sig_chirpfinal.hex"
};

short sound2[INPUTLEN] = {
#include "Sig_ChirpD.hex"
};

short sinus[INPUTLEN] = {
#include "sinus.hex"
};

short soundIn[INPUTLEN];

short corr[LAGS] = {0};
int lags = LAGS;
int input_l = SAMPLES;
int n_measurements = 0;
int NumberOfMeasurements = 50;
int distance[50] = {0};



//------------------------------------------------------------------------//
// Function:    main                                                      //
//                                                                        //
// Description: After calling a few initalization routines, main() just   //
//              waits in a loop forever.  The code to process the incoming //
//              data can be placed in the function Process_Data() in the  //
//              file "Process_Data.c".                                    //
//------------------------------------------------------------------------//
void main(void)
{   sysreg_write(reg_SYSCFG, 0x32);     //Initialize System Configuration Register

    Init_EBIU();
    Init_Flash();
    Init1836();
    Init_Sport0();
    Init_DMA();
    Init_Interrupts();

    Enable_DMA_Sport0();

    while(1)
    {
        if(doX){
        crosscorr_fr16(sound,
                       soundIn,
                       input_l,
                       lags,
                       corr);

        distance[n_measurements] = calc_dist(corr);
        //printf("%d \n",distance[n_measurements]/10);
        n_measurements++;
        if(n_measurements >= NumberOfMeasurements){
            n_measurements = 0;
        }
```

```
        num1 = 0;           //used to count when we record and play audio
        playFlag = 1;       //used to enable output audio
        recFlag = 1;        //used to enable input recording
        }
    }
}
```