```
/*
 *  Copyright 2010 by Spectrum Digital Incorporated.
 *  All rights reserved. Property of Spectrum Digital Incorporated.
 */

/*
 *  AIC3204 Tone
 *
 */
#include "stdio.h"
#include "usbstk5515.h"
extern Int16 AIC3204_rset( Uint16 regnum, Uint16 regval);

#define Rcv 0x08
#define Xmit 0x20

/* ---------------------------------------------------------------------- *
 *                                                                        *
 *  AIC3204 Loop                                                          *
 *      Output input from STEREO IN 1 through the HEADPHONE jack          *
 *                                                                        *
 * ---------------------------------------------------------------------- */
Int16 aic3204_loop_stereo_in1( )
{
    /* Pre-generated sine wave data, 16-bit signed samples */
    Int16 j, i = 0;
    Int16 sample, data1, data2, data3, data4;

    /* Configure AIC3204 */
    AIC3204_rset( 0, 0);         // Select page 0
    AIC3204_rset( 1, 1);         // Reset codec
    AIC3204_rset( 0, 1);         // Point to page 1
    AIC3204_rset( 1, 8);         // Disable crude AVDD generation from DVDD
    AIC3204_rset( 2, 1);         // Enable Analog Blocks, use LDO power
    AIC3204_rset( 0, 0);         // Select page 0

    /* PLL and Clocks config and Power Up  */
    AIC3204_rset( 27, 13);       // I2S, 16bits, BCLK e WCLK output, DOUT alta
impedancia
    AIC3204_rset( 28, 0);       // Data ofset = 0
    AIC3204_rset( 4, 3);         // PLL setting: PLLCLK <- MCLK, CODEC_CLKIN <-PLLCLK
    AIC3204_rset( 6, 8);         // PLL setting: J=8
    AIC3204_rset( 7, 18);        // PLL setting: MSB D=4672
    AIC3204_rset( 8, 64);        // PLL setting: LSB D=4672
    AIC3204_rset( 29, 0);       // BDIV_CLKIN = DAC_CLK
    AIC3204_rset( 30, 132);     // BCLK / 4
    AIC3204_rset( 5, 145);       // PLL setting: Power up PLL, P=1 e R=1
    AIC3204_rset( 13, 0 );       // MSB for DOSR = 128
    AIC3204_rset( 14, 128);      // LSB for DOSR = 128
    AIC3204_rset( 20, 128);      // AOSR for AOSR = 128
    AIC3204_rset( 11, 134);    // Power up NDAC and set NDAC value to 6
    AIC3204_rset( 12, 134);    // Power up MDAC and set MDAC value to 6
    AIC3204_rset( 18, 134);    // Power up NADC and set NADC value to 6
    AIC3204_rset( 19, 134);    // Power up MADC and set MADC value to 6

    /* DAC ROUTING and Power Up */
    AIC3204_rset(  0, 1);      // Select page 1
    AIC3204_rset( 12, 8);      // LDAC AFIR routed to HPL
    AIC3204_rset( 13, 8);      // RDAC AFIR routed to HPR
    AIC3204_rset(  0, 0);      // Select page 0
    AIC3204_rset( 64, 2);      // Left vol=right vol
    AIC3204_rset( 65, 16);     // Left DAC gain to xdB VOL; Right tracks Left
    AIC3204_rset( 63, 0xd4);   // Power up left, right data paths and set channel
    AIC3204_rset(  0, 1);      // Select page 1
    AIC3204_rset( 16, 0x06);   // Unmute HPL , 6dB gain
    AIC3204_rset( 17, 0x06);   // Unmute HPR , 6dB gain
    AIC3204_rset(  9, 0x30);   // Power up HPL, HPR
    AIC3204_rset(  0, 0);      // Select page 0
```

```
    USBSTK5515_wait( 500 );     // Wait

    /* ADC ROUTING and Power Up */
    AIC3204_rset(  0, 1 );        // Select page 1
    AIC3204_rset( 52, 0x30 );     // IN2_L to LADC_P through 40 kohm
    AIC3204_rset( 55, 0x30 );     // IN2_R to RADC_P through 40 kohmm
    AIC3204_rset( 54, 0x03 );     // CM_1 (common mode) to LADC_M through 40 kohm
    AIC3204_rset( 57, 0x03 );     // CM_1 (common mode) to RADC_M through 40 kohm
    AIC3204_rset( 59, 0x00 );     // MIC_PGA_L unmute
    AIC3204_rset( 60, 0x00 );     // MIC_PGA_R unmute
    AIC3204_rset(  0, 0 );        // Select page 0
    AIC3204_rset( 81, 0xc0 );     // Power up Left and Right ADC
    AIC3204_rset( 82, 0x00 );     // Fine gain
    AIC3204_rset( 83, 0x20 );     // Unmute Left ADC
    AIC3204_rset( 84, 0x20 );     // Unmute Right ADC
    AIC3204_rset( 0, 0 );
    USBSTK5515_wait( 200 );  // Wait

    /* I2S settings */
    I2S0_SRGR = 0x0;
    I2S0_CR = 0x8010;     // 16-bit word, slave, enable I2C
    I2S0_ICMR = 0x3f;     // Enable interrupts

    Config_DMA_I2S();

    /* Play Tone */
    //for ( i = 0 ; i < 5 ; i++ )
    //{
         while (1)
         {
            //for ( j = 0 ; j < 1000 ; j++ )
            //{
                //for ( sample = 0 ; sample < 48 ; sample++ )
                //{
                    /* Read Digital audio input */
                    data3 = I2S0_W0_MSW_R;  // 16 bit left channel received audio data
                    data1 = I2S0_W0_LSW_R;
                    data4 = I2S0_W1_MSW_R;  // 16 bit right channel received audio data
                    data2 = I2S0_W1_LSW_R;

                    while((Rcv & I2S0_IR) == 0);  // Wait for interrupt pending flag
                    /* Write Digital audio input */
                    I2S0_W0_MSW_W = data3;  // 16 bit left channel transmit audio data
                    I2S0_W0_LSW_W = 0;
                    I2S0_W1_MSW_W = data4;  // 16 bit right channel transmit audio data
                    I2S0_W1_LSW_W = 0;
                    while((Xmit & I2S0_IR) == 0);  // Wait for interrupt pending flag
                //}
            //}
         }
    //}
    /* Disble I2S */
    //I2S0_CR = 0x00;

    return 0;
}
```