/\*

\* This file is written using examples provided by Spectrum Digital

\*/

#include "stdio.h"

#include "usbstk5515.h"

#include "usbstk5515\_gpio.h"

#include "usbstk5515\_i2c.h"

#define AIC3204\_I2C\_ADDR 0x18

#define Rcv 0x08

#define Xmit 0x20

#ifndef TRUE

#define TRUE 1

#endif

#define freq\_6\_857 0xF1

#define freq\_8 0xE1

#define freq\_9\_6 0xD1

#define freq\_12 0xC1

#define freq\_16 0xB1

#define freq\_24 0xA1

#define freq\_48 0x91

/\* ------------------------------------------------------------------------ \*

\* \*

\* Testing Function \*

\* \*

\* ------------------------------------------------------------------------ \*/

Int16 AIC3204\_rget( Uint16 regnum, Uint16\* regval )

{

Int16 retcode = 0;

Uint8 cmd[2];

cmd[0] = regnum & 0x007F; // 7-bit Register Address

cmd[1] = 0;

retcode |= USBSTK5515\_I2C\_write( AIC3204\_I2C\_ADDR, cmd, 1 );

retcode |= USBSTK5515\_I2C\_read( AIC3204\_I2C\_ADDR, cmd, 1 );

\*regval = cmd[0];

USBSTK5515\_wait( 10 );

return retcode;

}

Int16 AIC3204\_rset( Uint16 regnum, Uint16 regval )

{

Uint8 cmd[2];

cmd[0] = regnum & 0x007F; // 7-bit Register Address

cmd[1] = regval; // 8-bit Register Data

return USBSTK5515\_I2C\_write( AIC3204\_I2C\_ADDR, cmd, 2 );

}

void AIC3204\_config(Uint8 sampling\_freq)

{

/\* Configure AIC3204 \*/

AIC3204\_rset( 0, 0 ); // Select page 0

AIC3204\_rset( 1, 1 ); // Reset codec

AIC3204\_rset( 0, 1 ); // Select 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, 0x0d ); // BCLK and WCLK is set as o/p to AIC3204(Master)

AIC3204\_rset( 28, 0x00 ); // Data ofset = 0

AIC3204\_rset( 4, 3 ); // PLL setting: PLLCLK <- MCLK, CODEC\_CLKIN <-PLL CLK

AIC3204\_rset( 6, 7 ); // PLL setting: J=7

AIC3204\_rset( 7, 0x06 ); // PLL setting: HI\_BYTE(D=1680)

AIC3204\_rset( 8, 0x90 ); // PLL setting: LO\_BYTE(D=1680)

AIC3204\_rset( 30, 0x88 ); // For 32 bit clocks per frame in Master mode ONLY

// BCLK=DAC\_CLK/N =(12288000/8) = 1.536MHz = 32\*fs

AIC3204\_rset( 5, sampling\_freq); // PLL setting: Power up PLL, P=1 and R=1

AIC3204\_rset( 13, 0 ); // Hi\_Byte(DOSR) for DOSR = 128 decimal or 0x0080 DAC oversamppling

AIC3204\_rset( 14, 0x80 ); // Lo\_Byte(DOSR) for DOSR = 128 decimal or 0x0080

AIC3204\_rset( 20, 0x80 ); // AOSR for AOSR = 128 decimal or 0x0080 for decimation filters 1 to 6

AIC3204\_rset( 11, 0x82 ); // Power up NDAC and set NDAC value to 2

AIC3204\_rset( 12, 0x87 ); // Power up MDAC and set MDAC value to 7

AIC3204\_rset( 18, 0x87 ); // Power up NADC and set NADC value to 7

AIC3204\_rset( 19, 0x82 ); // Power up MADC and set MADC value to 2

/\* DAC ROUTING and Power Up \*/

AIC3204\_rset( 0, 0x01 ); // Select page 1

AIC3204\_rset( 12, 0x08 ); // LDAC AFIR routed to HPL

AIC3204\_rset( 13, 0x08 ); // RDAC AFIR routed to HPR

AIC3204\_rset( 0, 0x00 ); // Select page 0

AIC3204\_rset( 64, 0x02 ); // Left vol=right vol

AIC3204\_rset( 65, 0x00 ); // Left DAC gain to 0dB VOL; Right tracks Left

AIC3204\_rset( 63, 0xd4 ); // Power up left,right data paths and set channel

AIC3204\_rset( 0, 0x01 ); // Select page 1

AIC3204\_rset( 16, 0x00 ); // Unmute HPL , 0dB gain

AIC3204\_rset( 17, 0x00 ); // Unmute HPR , 0dB gain

AIC3204\_rset( 9, 0x30 ); // Power up HPL,HPR

AIC3204\_rset( 0, 0x00 ); // Select page 0

USBSTK5515\_wait( 500 ); // Wait

/\* ADC ROUTING and Power Up \*/

AIC3204\_rset( 0, 1 ); // Select page 1

AIC3204\_rset( 0x34, 0x30 ); // STEREO 1 Jack

// IN2\_L to LADC\_P through 40 kohm

AIC3204\_rset( 0x37, 0x30 ); // IN2\_R to RADC\_P through 40 kohmm

AIC3204\_rset( 0x36, 3 ); // CM\_1 (common mode) to LADC\_M through 40 kohm

AIC3204\_rset( 0x39, 0xc0 ); // CM\_1 (common mode) to RADC\_M through 40 kohm

AIC3204\_rset( 0x3b, 0 ); // MIC\_PGA\_L unmute

AIC3204\_rset( 0x3c, 0 ); // MIC\_PGA\_R unmute

AIC3204\_rset( 0, 0 ); // Select page 0

AIC3204\_rset( 0x51, 0xc0 ); // Powerup Left and Right ADC

AIC3204\_rset( 0x52, 0 ); // Unmute Left and 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

}

/\* ------------------------------------------------------------------------ \*

\* \*

\* main( ) \*

\* \*

\* ------------------------------------------------------------------------ \*/

main( void )

{

Int16 LEFT\_DATA,RIGHT\_DATA;

/\* Initialize BSL \*/

SYS\_EXBUSSEL = 0x6100;

USBSTK5515\_init( );

AIC3204\_config(freq\_48);

//while(TRUE)

while(1)

{

while((Rcv & I2S0\_IR) == 0); // Wait for interrupt pending flag

LEFT\_DATA = I2S0\_W0\_MSW\_R; // 16 bit left channel received audio data

RIGHT\_DATA = I2S0\_W1\_MSW\_R; // 16 bit right channel received audio data

while((Xmit & I2S0\_IR) == 0);

I2S0\_W0\_MSW\_W = LEFT\_DATA; // 16 bit left channel transmit audio data

I2S0\_W1\_MSW\_W = RIGHT\_DATA; // 16 bit right channel transmit audio data

}

/\* Disble I2S \*/

I2S0\_CR = 0x00;

return 0;

}

Extra code:

Extra Code:(Convolution part)

#define IR\_length 11 //Length of filter coeffs

#define In\_length 51//length of input signal

#define tmp\_l IR\_length+In\_length-1

Int16 x[In\_length]={};

Int16 coefs[IR\_length]={};

Int16 coefs\_rev[IR\_length]={};

Int16 array[tmp\_l]={0};

Int16 buffer[IR\_length]={0};

Int16 buffer\_temp[IR\_length]={0};

int i=0,j=0,k=0,n=0;

Int16 temp=0;

Int16 output=0;

Int16 m=0;

/\*Include in main

for(k=0;k<IR\_length;k++)

{

buffer[k]=0;

}

\*/

Int16 convolution(Int16 input)

{

/\*Reversal function- include in main

for(i=0;i<IR\_length;i++)

{

coefs\_rev[i]=coefs[IR\_length-1-i];

}

\*/

buffer\_temp[0]=0;

for(n=1;n<IR\_length1;n++)

{

buffer\_temp[n]=buffer[n-1];

}

for(n=0;n<IR\_length;n++)

{

buffer[n]=buffer\_temp[n];

}

buffer[0]=input;

for(n=0;n<IR\_length;n++)

{

temp+= buffer[n]\*coefs[n];

}

m=temp;

temp=0;

return m;

}

/\*loop to give inputs

for(i=0;i<tmp\_l;i++)

{

if(i<In\_length)

{

output=convolution(x[i]);

}

else

{

output=convolution(0);

}

array[i]=output;

}

Simplified C Code:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Online C Compiler.

Code, Compile, Run and Debug C program online.

Write your code in this editor and press "Run" button to compile and execute it.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#define IR\_length 4 //Length of filter coeffs

#define In\_length 4//length of input signal

#define tmp\_l IR\_length+In\_length-1

int x[In\_length]={1,2,2,1};

int coefs[IR\_length]={1,2,3,4};

//Int16 coefs\_rev[IR\_length]={0};

int array[tmp\_l]={0};

int buffer[IR\_length]={0};

int buffer\_temp[IR\_length]={0};

int i=0,j=0,k=0,n=0;

int temp=0;

int output=0;

int m=0;

int convolution(int input)

{

/\*Reversal function- include in main

for(i=0;i<IR\_length;i++)

{

coefs\_rev[i]=coefs[IR\_length-1-i];

}

\*/

//for(j=0; j<tmp\_l ;j++)

//{

buffer\_temp[0]=0;

for(n=1;n<IR\_length;n++)

{

buffer\_temp[n]=buffer[n-1];

}

for(n=0;n<IR\_length;n++)

{

buffer[n]=buffer\_temp[n];

}

for(n=0;n<IR\_length;n++)

{

printf("buffer array=%d\t",buffer[n]);

}

printf("\n");

{

buffer[0]=input;

}

printf("buffer[0]= %d\n",buffer[0]);

for(n=0;n<IR\_length;n++)

{

temp+= buffer[n]\*coefs[n];

}

m=temp;

temp=0;

return m;

//printf("array[j]= %d\n",array[j]);

//}

}

int main()

{

for(k=0;k<IR\_length;k++)

{

buffer[k]=0;

}

/\*Include in main

for(k=0;k<IR\_length;k++)

{

buffer[k]=0;

}

\*/

//loop to give inputs

for(i=0;i<tmp\_l;i++)

{

if(i<In\_length)

{

output=convolution(x[i]);

}

else

{

output=convolution(0);

}

array[i]=output;

printf("output= %d\n",output);

}

return 0;

}