#include "dsk6713\_aic23.h" //codec support

Uint32 fs = DSK6713\_AIC23\_FREQ\_8KHZ; //set sampling rate

#define DSK6713\_AIC23\_INPUT\_MIC 0x0015

#define DSK6713\_AIC23\_INPUT\_LINE 0x0011

Uint16 inputsource=DSK6713\_AIC23\_INPUT\_MIC; //select input

#define LOOPLENGTH 8

short loopindex = 0; //table index

short gain = 10; //gain factor

short sine\_table[LOOPLENGTH]={0,707,1000,707,0,-707,-1000,-707}; //sine values

void main()

{

 comm\_poll(); //init DSK,codec,McBSP

 DSK6713\_LED\_init(); //init LED from BSL

 DSK6713\_DIP\_init(); //init DIP from BSL

 while(1) //infinite loop

 {if(DSK6713\_DIP\_get(0)==0) //=0 if DIP switch #0 pressed

 {

 DSK6713\_LED\_on(); //turn LED #0 ON

 output\_left\_sample(sine\_table[loopindex++]\*gain); //output sample

 if (loopindex >= LOOPLENGTH) loopindex = 0; //reset table index

 }

 else DSK6713\_LED\_off(0); //turn LED off if not pressed

 } //end of while(1) infinite loop

} //end of main

//C6713dskinit.c Includes functions from TI in the C6713 CSL and C6713DSK BSL

#include "C6713dskinit.h"

#define using\_bios //if BIOS don't use top of vector table

extern Uint32 fs; //for sampling frequency

extern Uint16 inputsource;

void c6713\_dsk\_init() //dsp-peripheral initialization

{

DSK6713\_init(); //call BSL to init DSK-EMIF,PLL)

hAIC23\_handle=DSK6713\_AIC23\_openCodec(0, &config); //handle(pointer) to codec

DSK6713\_AIC23\_setFreq(hAIC23\_handle, fs); //set sample rate

DSK6713\_AIC23\_rset(hAIC23\_handle, 0x0004, inputsource); // choose mic or line in

MCBSP\_config(DSK6713\_AIC23\_DATAHANDLE,&AIC23CfgData); //interface 32 bits toAIC23

MCBSP\_start(DSK6713\_AIC23\_DATAHANDLE, MCBSP\_XMIT\_START | MCBSP\_RCV\_START |

 MCBSP\_SRGR\_START | MCBSP\_SRGR\_FRAMESYNC, 220); //start data channel again

}

void comm\_poll() //added for communication/init using polling

{

 poll=1; //1 if using polling

 c6713\_dsk\_init(); //init DSP and codec

}

void comm\_intr() //for communication/init using interrupt

{

 poll=0; //0 since not polling

 IRQ\_globalDisable(); //disable interrupts

 c6713\_dsk\_init(); //init DSP and codec

 CODECEventId=MCBSP\_getXmtEventId(DSK6713\_AIC23\_codecdatahandle);//McBSP1 Xmit

#ifndef using\_bios //do not need to point to vector table

 IRQ\_setVecs(vectors); //point to the IRQ vector table

#endif //since interrupt vector handles this

 IRQ\_map(CODECEventId, 11); //map McBSP1 Xmit to INT11

 IRQ\_reset(CODECEventId); //reset codec INT 11

 IRQ\_globalEnable(); //globally enable interrupts

 IRQ\_nmiEnable(); //enable NMI interrupt

 IRQ\_enable(CODECEventId); //enable CODEC eventXmit INT11

 output\_sample(0); //start McBSP interrupt outputting a sample

}

void output\_sample(int out\_data) //for out to Left and Right channels

{

 short CHANNEL\_data;

 AIC\_data.uint=0; //clear data structure

 AIC\_data.uint=out\_data; //32-bit data -->data structure

//The existing interface defaults to right channel. To default instead to the

//left channel and use output\_sample(short), left and right channels are swapped

//In main source program use LEFT 0 and RIGHT 1 (opposite of what is used here)

 CHANNEL\_data=AIC\_data.channel[RIGHT]; //swap left and right channels

 AIC\_data.channel[RIGHT]=AIC\_data.channel[LEFT];

 AIC\_data.channel[LEFT]=CHANNEL\_data;

 if (poll) while(!MCBSP\_xrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to transmit

 MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE,AIC\_data.uint); //write/output data

}

void output\_left\_sample(short out\_data) //for output from left channel

{

 AIC\_data.uint=0; //clear data structure

 AIC\_data.channel[LEFT]=out\_data; //data from Left channel -->data structure

 if (poll) while(!MCBSP\_xrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to transmit

 MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE,AIC\_data.uint); //output left channel

}

void output\_right\_sample(short out\_data) //for output from right channel

{

 AIC\_data.uint=0; //clear data structure

 AIC\_data.channel[RIGHT]=out\_data; //data from Right channel -->data structure

 if (poll) while(!MCBSP\_xrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to transmit

 MCBSP\_write(DSK6713\_AIC23\_DATAHANDLE,AIC\_data.uint); //output right channel

}

Uint32 input\_sample() //for 32-bit input

{

 short CHANNEL\_data;

 if (poll) while(!MCBSP\_rrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to receive

 AIC\_data.uint=MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //read data

//Swapping left and right channels (see comments in output\_sample())

 CHANNEL\_data=AIC\_data.channel[RIGHT]; //swap left and right channel

 AIC\_data.channel[RIGHT]=AIC\_data.channel[LEFT];

 AIC\_data.channel[LEFT]=CHANNEL\_data;

 return(AIC\_data.uint);

}

short input\_left\_sample() //input to left channel

{

 if (poll) while(!MCBSP\_rrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to receive

 AIC\_data.uint=MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //read into left channel

 return(AIC\_data.channel[LEFT]); //return left channel data

}

short input\_right\_sample() //input to right channel

{

 if (poll) while(!MCBSP\_rrdy(DSK6713\_AIC23\_DATAHANDLE)); //if ready to receive

 AIC\_data.uint=MCBSP\_read(DSK6713\_AIC23\_DATAHANDLE); //read into right channel

 return(AIC\_data.channel[RIGHT]); //return right channel data

}