```
/*
Program block_audio.c uses double-buffered block transfer to implement audio in-out for
the ADSP-BF706 EZ-KIT Mini evaluation system. It is an extensively modified and minimised
version of the program TalkThrough_BF706Mini.c supplied by ADI. This contained errors in
the SPORT definitions of the word lengths. Some of the routines have also been replaced
or modified. Although the program uses block transfer and DMA, it is still bound by an
eternal loop for audio I/O, and is not truly interrupt driven. For this, users are
referred to c_inout_int.c (C code) or interrupt_inout.asm (assembly version), published
earlier by the author.

Note: it requires inclusion of the codec (ADAU1761) driver in the system.svc.
The program follows 2-space indentation and no error checking. Use with caution.

Author: Patrick Gaydecki
Date  : 23.02.2021
*/

#include <drivers/codec/adau1761/adi_adau1761.h>
#include <string.h>
#include <cdefBF706.h>
#define SAMPLES 256
#define BUFFER 1024
#define CMEM 1024
#define SAMPLE_RATE (ADI_ADAU1761_SAMPLE_RATE_48KHZ)

uint8_t sportRxMem[ADI_SPORT_DMA_MEMORY_SIZE];
uint8_t sportTxMem[ADI_SPORT_DMA_MEMORY_SIZE];
ADI_ADAU1761_SPORT_INFO sportRxInfo;
ADI_ADAU1761_SPORT_INFO sportTxInfo;
fract32 RxBuf1[SAMPLES];
fract32 RxBuf2[SAMPLES];
fract32 TxBuf1[SAMPLES];
fract32 TxBuf2[SAMPLES];
fract32 *pRxAudio;
fract32 *pTxAudio;
ADI_ADAU1761_SPORT_INFO sportRxInfo;
ADI_ADAU1761_SPORT_INFO sportTxInfo;
uint8_t codecm[CMEM];
ADI_ADAU1761_HANDLE  hADAU1761;

void TWI_write(uint16_t, uint8_t);
void codec_configure(void);
void sport_configure(void);

// Function sport_configure initialises the SPORT0. Refer to pages 31-59, 31-67,
// 31-75 and 31-76 of the ADSP-BF70x Blackfin+ Processor Hardware Reference manual.

void sport_configure()
{
   *pREG_PORTC_FER=0x0003F0;             // Set up Port C in peripheral mode
   *pREG_PORTC_FER_SET=0x0003F0;         // Set up Port C in peripheral mode
   *pREG_SPORT0_CTL_A=0x2001973;         // Set up SPORT0 (A) as TX to codec, 24 bits
   *pREG_SPORT0_DIV_A=0x400001;          // 64 bits per frame, clock divisor of 1
   *pREG_SPORT0_CTL_B=0x0001973;         // Set up SPORT0 (B) as RX from codec, 24 bits
   *pREG_SPORT0_DIV_B=0x400001;          // 64 bits per frame, clock divisor of 1
}

// Function TWI_write is a simple driver for the TWI. Refer to page 24-15 onwards
// of the ADSP-BF70x Blackfin+ Processor Hardware Reference manual.

void TWI_write(uint16_t reg_add, uint8_t reg_data)
{
   int n;
   reg_add=(reg_add<<8)|(reg_add>>8);    // Reverse low order and high order bytes
   *pREG_TWI0_CLKDIV=0x3232;             // Set duty cycle
   *pREG_TWI0_CTL=0x8c;                  // Set prescale and enable TWI
   *pREG_TWI0_MSTRADDR=0x38;             // Address of codec
   *pREG_TWI0_TXDATA16=reg_add;          // Address of register to set, LSB then MSB
   *pREG_TWI0_MSTRCTL=0xc1;              // Command to send three bytes and enable tx
   for(n=0;n<8000;n++){}                 // Delay since codec must respond
   *pREG_TWI0_TXDATA8=reg_data;          // Data to write
```

```
   for(n=0;n<10000;n++){}                // Delay
   *pREG_TWI0_ISTAT=0x050;               // Clear TXERV interrupt
   for(n=0;n<10000;n++){}                // Delay
   *pREG_TWI0_ISTAT=0x010;               // Clear MCOMP interrupt
}

// Function codec_configure initialises the ADAU1761 codec. Refer to the control register
// descriptions, page 51 onwards of the ADAU1761 data sheet.

void codec_configure()
{
   TWI_write(0x4000, 0x01);              // Enable master clock, disable PLL
   TWI_write(0x40F9, 0x7f);              // Enable all clocks
   TWI_write(0x40Fa, 0x03);              // Enable all clocks
   TWI_write(0x4015, 0x01);              // Set serial port master mode
   TWI_write(0x4019, 0x13);              // Set ADC to on, both channels
   TWI_write(0x401c, 0x21);              // Enable left channel mixer
   TWI_write(0x401e, 0x41);              // Enable right channel mixer
   TWI_write(0x4029, 0x03);              // Turn on power, both channels
   TWI_write(0x402A, 0x03);              // Set both DACs on
   TWI_write(0x40f2, 0x01);              // DAC gets L, R input from serial port
   TWI_write(0x40f3, 0x01);              // ADC sends L, R input to serial port
   TWI_write(0x400a, 0x0b);              // Set left line-in gain to 0 dB
   TWI_write(0x400c, 0x0b);              // Set right line-in gain to 0 dB
   TWI_write(0x4023, 0xe7);              // Set left headphone volume to 0 dB
   TWI_write(0x4024, 0xe7);              // Set right headphone volume to 0 dB
   TWI_write(0x4017, 0x00);              // Set codec default sample rate, 48 kHz
}

int main(void)
{
   bool aa=true;
   *pREG_SEC0_CCTL0=0X10001;             //  Replaces adi_initComponents();
   codec_configure();                    //  Configure codec for standard in-out
   *pREG_PORTC_DIR_SET=1;                //  Debug only. For pulsing GPIO when sampled
   *pREG_PORTC_DATA=0;

// Open ADAU1761 driver. Not necessary if code written from low-level
   adi_adau1761_Open(0, codecm, CMEM,   ADI_ADAU1761_COMM_DEV_TWI, &hADAU1761);

// configure SPORT for Rx data transfer
   sportRxInfo.eChannel = ADI_HALF_SPORT_B;
   sportRxInfo.pMemory = sportRxMem;
   sportRxInfo.eDataLen = ADI_ADAU1761_SPORT_DATA_32;
   adi_adau1761_ConfigSPORT(hADAU1761, ADI_ADAU1761_SPORT_INPUT, &sportRxInfo);

// configure SPORT for Tx data transfer
   sportTxInfo.eChannel = ADI_HALF_SPORT_A;
   sportTxInfo.pMemory = sportTxMem;
   sportTxInfo.eDataLen = ADI_ADAU1761_SPORT_DATA_32;
   adi_adau1761_ConfigSPORT(hADAU1761, ADI_ADAU1761_SPORT_OUTPUT, &sportTxInfo);
// Configure SPORT with correct word length settings
   sport_configure();

// Enable data buffers, 2 for Rx and 2 for Tx
   adi_adau1761_SubmitRxBuffer(hADAU1761, RxBuf1, BUFFER);
   adi_adau1761_SubmitRxBuffer(hADAU1761, RxBuf2, BUFFER);
   adi_adau1761_SubmitTxBuffer(hADAU1761, TxBuf1, BUFFER);
   adi_adau1761_SubmitTxBuffer(hADAU1761, TxBuf2, BUFFER);

// Enable input and output
   adi_adau1761_EnableInput(hADAU1761, true);
   adi_adau1761_EnableOutput(hADAU1761, true);

// Go round this loop forever, receiving and transmitting buffers.
// Program swaps between buffers 1 and 2.
   while (true)
   {
// Toggles pin 4 on connector P2. This will be 187.5 Hz, i.e. 48000/256
      aa=!aa;
      *pREG_PORTC_DATA=aa;
```

```
        adi_adau1761_GetRxBuffer(hADAU1761, (void**)&pRxAudio);
        if (pRxAudio!= NULL) adi_adau1761_SubmitRxBuffer(hADAU1761, pRxAudio, BUFFER);
        adi_adau1761_GetTxBuffer(hADAU1761, (void**)&pTxAudio);
        if (pTxAudio!= NULL)
        {
           memcpy(&pTxAudio[0], &pRxAudio[0], BUFFER);
           adi_adau1761_SubmitTxBuffer(hADAU1761, pTxAudio, BUFFER);
        }
     }
}
```