```
/*
Program inout demonstrates a simple audio in-out capability of the ADSP-BF706 EZ-KIT Mini
evaluation system using a polled, sample-by-sample algorithm. The program is completely
self-contained, using only the header cdefBF706.h for the addresses of the BF706 registers.

Author: Patrick Gaydecki
Date  : 15.12.2016
*/

#include <stdio.h>
#include <cdefBF706.h>
#include "system/adi_initialize.h"

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

// Function sport_configure initialises the SPORT0. Refer to pages 26-59, 26-67,
// 26-75 and 26-76 of the ADSP-BF70x Blackfin+ Processor Hardware Reference manual.
void sport_configure()
{
  *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 transmit
  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)
{
  adi_initComponents();
  bool my_audio=true;
  codec_configure();
  sport_configure();
  while(my_audio)
  {
    while((*pREG_SPORT0_CTL_B & 0xc0000000)==0x0){}  // Wait for input buffer flag, right ch
    *pREG_SPORT0_TXPRI_A=*pREG_SPORT0_RXPRI_B;       // Read right channel in, send it out
    while((*pREG_SPORT0_CTL_B & 0xc0000000)==0x0){}  // Wait for input buffer flag, left ch
    *pREG_SPORT0_TXPRI_A=*pREG_SPORT0_RXPRI_B;       // Read left channel in, send it out
  }
  return 0;
}
```