6 VÝVOJOVÝ MODUL ANALOG DEVICES ADSP-BF533 EZ-KIT LITE (II ČASŤ)

Ciele cvičenia:

- demonštračný program pre prístup k audio AD a DA prevodníkom,
- aplikačné príručky (EE – Engineer to Engineer Notes) a ich využitie pri tvorbe programov,
- knižničná funkcia pre výpočet reálnej FFT,
- žrebovanie zadaní.

6.1 AUDIO ROZHRANIE

Typickým príkladom, ktorý je možné demonštrovať pomocou vývojových modulov prakticky všetkých výrobcov DSP je vstup a výstup analógových dát s využitím AD a DA prevodníkov. Výnimkou nie je ani vývojový modul **ADSP BF533 EZ-KIT Lite**, ktorý je osadený multikanálovým 96-kHz kodekom AD1836 [1]. AD1836 je jednočípový kodek, ktorý podporuje 3 stereo DA kanály a 2 stereo AD kanály, pričom využíva multibitovú sigma-delta[1] technológiu prevodu. Kodek využíva SPI rozhranie na konfiguráciu interných registrov AD1836 (napr. nastavenie frekvencie vzorkovaia, módu činnosti, zisk, a pod.).

Kodek je pripojený k DSP pomocou rozhrania SPORT0 procesora BF533. Procesor môže komunikovať s audio kodekom v časovom multiplexe (**TDM** – Time Division Multiplex) alebo v dvoj-vodičovom móde (**TWI** – Two Wire Interface, tiež označovaný ako **IS mód**).

TWI mód umožňuje kodeku pracovať so vzorkovacou frekvenciou až 96 kHz, umožňuje však využitie len dvoch DA kanálov. TDM mód umožňuje činnosť kodeku len do 48 kHz, umožňuje však využitie všetkých troch DA výstupov.

6.1.1 DEMONŠTRAČNÝ PRÍKLAD PRE PRÍSTUP K AD A DA KODEKOM

Inicializácia kodeku AD1836 vyžaduje veľmi podrobnú znalosť registrov samotného kodeku ako aj procesora BFť33. Tieto informácie presahujú rámec programov preberaných na cvičeniach a je ich možné nájsť v príslušných manuáloch.

V distribúcii prostredia VisualDSP++ je možné nájsť pripravené projekty pre prácu s periférnymi obvodmi modulu ADSP BF533 EZ-KIT Lite. Príklady pre prácu s audio kodekom AD1836 sú v adresári[2]

..\Blackfin\Examples\ADSP-BF533 EZ-Kit Lite\Audio Codec Talkthrough...

[1] Princíp sigma-delta prevodníkov bude vysvetlený na jednej z nasledujúcich prednášok.

[2] Platné pre verziu VisualDSP++ v.4.5.

ktorý obsahuje projekty pre TDM a IS módy v ASM aj jazyku C.

Uvedené projekty v jazyku C najskôr inicializujú použité periférie:
EEBIU - pripojenie externých pamätí
FLASH - konfigurácia IO vývodov (pre všeobecné použitie) FLASH pamäte
SPORT0 - sériový port procesora
DMA - použité DMA kanály

a následne skonfigurujú prerušovací systém procesora a povolia prenos z/do rozhrania SPORT0 pomocou DMA kanálov, čo je realizované v tele funkcie main.

```
//-------------------------------------------------------------------------------------------------//
// Function:     main                                                                              //
//                                                                                                 //
// Description:  After calling a few initalization routines, main() just                          //
//               waits in a loop forever.  The code to process the incoming                        //
//               data can be placed in the function Process_Data() in the                          //
//               file "Process_Data.c".                                                            //
//-------------------------------------------------------------------------------------------------//
void main(void)
{

        sysreg_write(reg_SYSCFG, 0x32);       //Initialize System Configuration Register
        Init_EBIU();
        Init_Flash();
        Init1836();
        Init_Sport0();
        Init_DMA();
        Init_Sport_Interrupts();
        Enable_DMA_Sport0();

        while(1);
}
```

Následne hlavný program čaká v nekonečnej slučke a systém prerušení realizuje spracovanie vstupných a výstupných dát pomocou funkcie (Process_data.c)

```
#include "Talkthrough.h"

//-------------------------------------------------------------------------------------------------------------------------------//
// Function:     Process_Data()                                                                                                  //
//                                                                                                                               //
// Description: This function is called from inside the SPORT0 ISR every                                                        //
//                                    time a complete audio frame has been received. The new                                     //
//                                    input samples can be found in the variables iChannel0LeftIn,                               //
//                                    iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn                                     //
//                                    respectively. The processed data should be stored in                                       //
//                                    iChannel0LeftOut, iChannel0RightOut, iChannel1LeftOut,                                     //
//                                    iChannel1RightOut, iChannel2LeftOut and iChannel2RightOut                                  //
//                                    respectively.                                                                              //
//-------------------------------------------------------------------------------------------------------------------------------//
void Process_Data(void)
{
        iChannel0LeftOut = iChannel0LeftIn;
        iChannel0RightOut = iChannel0RightIn;
        iChannel1LeftOut = iChannel1LeftIn;
        iChannel1RightOut = iChannel1RightIn;
}
```

Najjednoduchšie projekty[3] môžu využiť uvedenú štruktúru projektu a jednoduchou modifikáciou slučky funkcie *Process_Data()* realizovať spracovanie reálnych audio dát (napr. pomocou FIT filtra z minulých cvičení).

Príklad
Pomocou osciloskopu, generátora a vývojového modulu ADSP BF533 EZ-Kit Lite overte funkčnosť funkcie Process_Data().

6.1.2 APLIKAČNÉ PRÍRUČKY A ICH VYUŽITIE PRI TVORBE APLIKÁCIÍ

Už pri zbežnom pohľade na zvyšné zdrojové kódy projektu je vidno, že obsahujú nízkoúrovňové nastavovanie preiférií BF533 a AD1836. Typickým postupom, ktorý je možné využiť pri efektívnom zvládnutí uvedenej problematiky je preštudovanie vhodných aplikačných príručiek (**AN** - Application Notes). Firma Analog Devices pre AN z oblasti DSP používa označenie **Engineer to Engineer Note EE-xxx**, kde xxx je príslušné číslo. Prehľad dostupných EE je možné nájsť v [2].

EE reprezentujú koncentrovaný zdroj informácií z danej oblasti a umožňujú pri vývojovej práci výrazné zrýchlenie tvorby aplikácie. Napr. pre pochopenie konfigurácie prerušovacieho systému procesora BF533 v jazyku C, ktoré je použité v demonštračnom príklade AD a DA kodeku je zaujímavá EE-192 [3].

Príklad
Identifikujte konštrukcie na inicializáciu prerušovacieho systému opísané v EE-192 v projekte pre prácu s AD a DA kodekom.

Príklad
Pozrite si zoznam EE dostupných pre procesory ADSP firmy Analog Devices.

6.2 KNIŽNIČNÁ FUNKCIA PRE VÝPOČET FFT

Algoritmus výpočtu rýchlej Fourierovej transformácie (**FFT** - Fast Fourier Transformation) patrí medzi základné algoritmy ČSS. Knižnice pre ČSS v prostredí VisualDSP++ [4] poskytujú pre programátora viacero verzií knižničných C funkcií pre výpočet FFT (napr. funkcií pre výpočet pre priame a spätné transformácie, FFT pre rýdzo reálne vstupné postupnosti, FFT pre komplexné postupnosti, FFT využívajúce rôzne základy – Radix 2, Radix 4, a pod.).

Napr. aj jeden z príkladov pre demonštráciu spätného telemetrického kanálu [5] využíval výpočet reálnej FFT v nekonečnej slučke funkcie main():

[3] Uvedená koncepcia spracovania vzoriek však má viaceré praktické obmedzenia. Spracovanie aktuálnej vzorky musí byť ukončené do príchodu nasledujúcej vzorky. Pri zložitejších algoritmoch tento prístup nie je výhodný a je potrebné realizovať napr. blokové spracovanie. Toto je možné realizovať napr. tak, že v prerušení sú dáta zapisované/čítané do/z vyrovnávacích pamätí. Po ich naplnení/vyprázdnení je ich naplnenie/vyprázdnenie signalizované hlavnému programu. V hlavnom programe je potom možné realizovať príslušnú blokovú operáciu. Počas jej realizácie je možné zachytávať dáta do sekundárnej vyrovnávacej pamäte (tzv. **ping-pong technika**). Využitie tejto techniky bude demonštrované v nasledujúcom cvičení pri blokovom spracovaní vzoriek AD a DA prevodníkov pomocou IIR filtra.

```
while (1) {

        //generate input
        create_samples(freq);

        //fft
        rfft_fr16(input_arr, t, out, w, wst, n, block_exponent, scale_method);

        for (i=0; i<NUMPOINTS; i++) {
                mag[i] = sqrt(out[i].re*out[i].re+out[i].im*out[i].im);
        }

        //write to BTC channel
        btc_write_array(0, (unsigned int*)input_arr, sizeof(input_arr));
        btc_write_array(1, (unsigned int*)mag, sizeof(mag));


        freq += BTC_CHAN2;
        if (freq > MAXFREQ) freq = MINFREQ;
}
```

Kompletný projekt je možné nájsť v adresári[4]
..\Blackfin\Examples\ ADSP-BF533 EZ-Kit Lite\Background_Telemetry\ fftDemo

Príklad[5]
*Preštudujte dokumentáciu ku knižničnej funkcii **rfft_fr16()** a na upravte projekt BTC_fft tak, aby demonštroval správnu funkčnosť knižničnej funkcie v simulátore prostredia ViusalDSP++.*

LITERATÚRA

[1] AD1836A – Multichannel 96 kHz Codec. Datasheet, Analog Devices, Inc., 2003.

[2] http://www.analog.com/en/embedded-processing-dsp/blackfin/processors/application-notes/resources/index.html

[3] Joe, B.: Using C to Create Interrupt Driven Systems on Blackfin Processors. Engineer to Engineer Note EE-192, May 2003, pp.1-9.

[4] VisualDSP++ 4.5 C/C++ Compiler and Library Manual for Blackfin Processors. Analog Devices, Inc., April 2006.

[5] VisualDSP++ 4.5 Getting Starter Guide. Analog Devices, Inc., January 2005, pp.3.22-3.46.

[4] Platné pre VisualDSP++ v.4.5.

[5] Podobný typ úloh bude realizovaný v zadaniach, ktorých úlohou je demonštrovanie funkčnosti vybraných knižničných funkcií pre DSP.