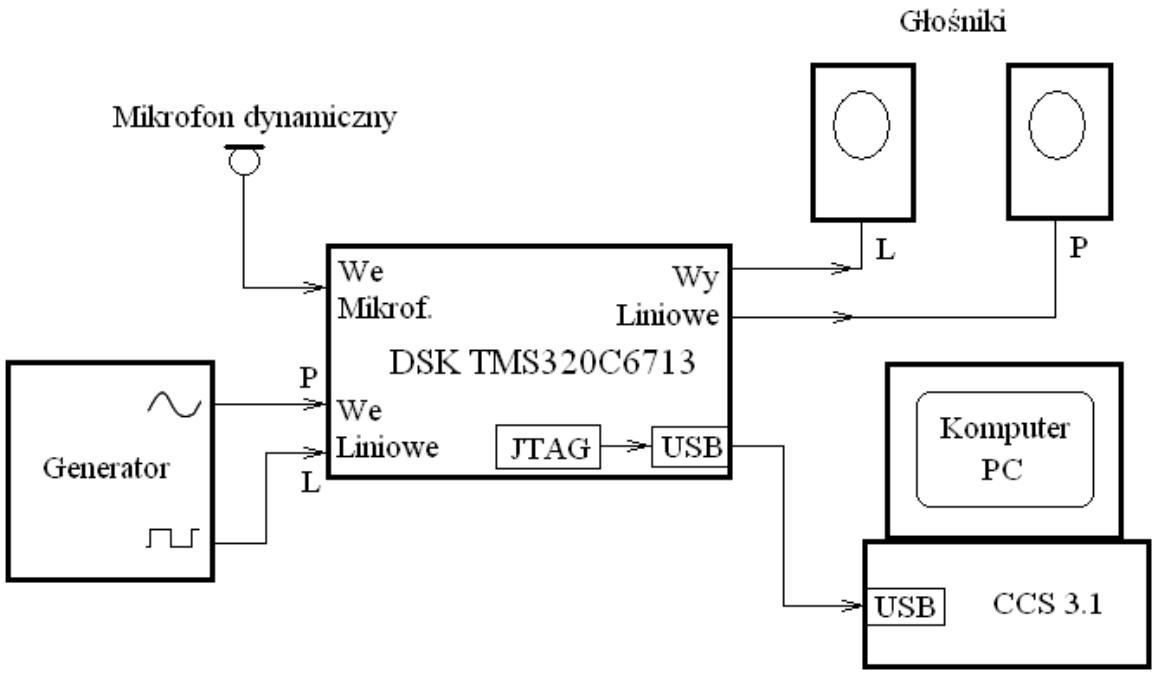
Procesory Sygnałowe i Logika Programowalna
Laboratorium
Ćw. 1

Zajęcia wprowadzające I

1. Wstęp

Celem ćwiczenia jest praktyczne zapoznanie studentów z własnościami i metodami programowania zmiennoprzecinkowych procesorów sygnałowych Texas Instruments TMS320C6713 (C6713). Głównym elementem zestawu laboratoryjnego jest Texas Instruments Starter Kit (DSK) zawierający kartę z procesorem C6713 oraz zestawem interfejsów analogowych i cyfrowych, jak również oprogramowanie narzędziowe Code Composer Studio (CCS) wersji 3.1.

Schemat zestawu laboratoryjnego

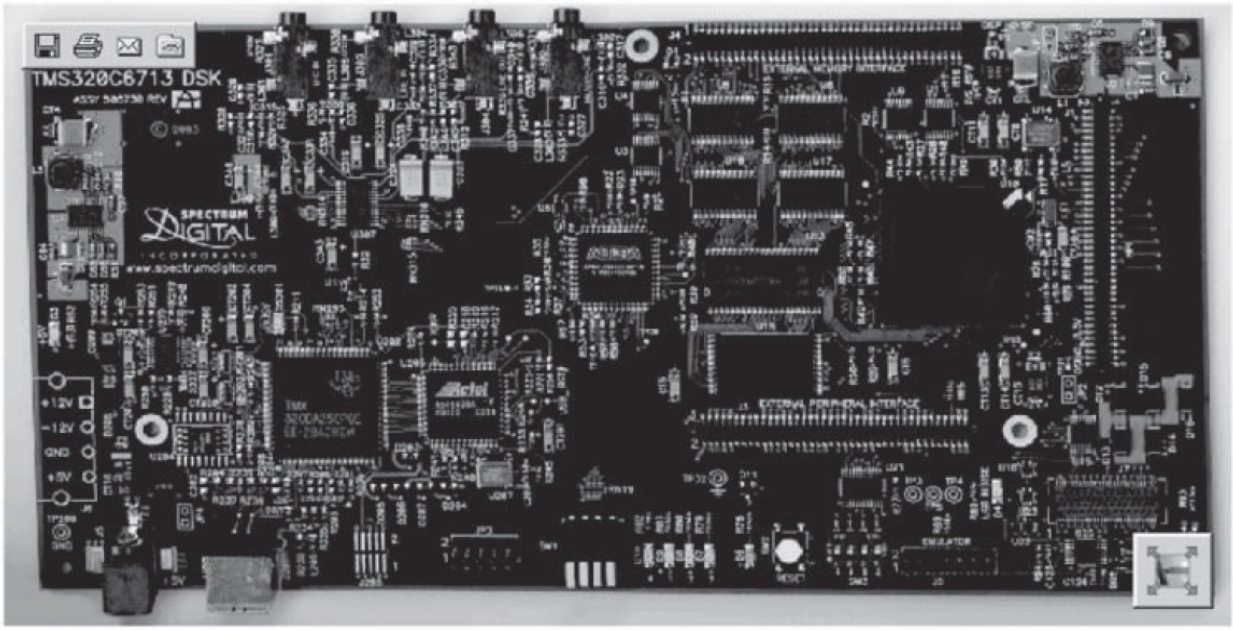


Opis zestawu laboratoryjnego

1. TI DSP Starter Kit (DSK) procesora TMS320C6713 zawierający:
   a) Kartę z procesorem TMS320C6713, urządzeniami peryferyjnymi (m. in. 32-bitowy kodek stereo dla wejścia i wyjścia analogowego) oraz emulatorem JTAG (USB)
   b) Code Composer Studio (CCS) - zintegrowane środowisko programistyczne (IDE), zawierające kompilatory języka C oraz asemblera, linker, debugger, itp. CCS jest instalowany na komputerach klasy PC.
   c) przewód USB do połączenia DSK z komputerem sterującym klasy PC oraz zasilacz.
2. Generatory sygnału sinusoidalnego (1 Vp-p) oraz prostokątnego (ampl. 0.5 V) o regulowanej częstotliwości w zakresie 200 Hz – 20000 Hz.
3. Mikrofon dynamiczny o impedancji 600 Ω
4. Dwa głośniki aktywne

2. Karta DSK

Karta DSK jest kompletnym systemem DSP czasu rzeczywistego, zawierającym zmiennoprzecinkowy procesor sygnałowy TMS320C6713 oraz 32-bitowy codec stereo TLV320AIC23 (AIC23) – techologia sigma-delta, przetwornik AC i DC, sterowany zegarem 12-MHz, częstotliwość próbkowania 8-96 kHz, wejście linowe stereo, wejście mikrofonowe mono, wyjście liniowe stereo, wyjście słuchawkowe.
Dwa złącza 80-pinowe pozwalają na podłączenie kart rozszerzeń (daughter cards), urządzeń peryferyjnych (m. in. wyświetlacza ciekłokrystalicznego LCD) oraz interfejsów pamięci zewnętrznej. Karta DSK zawiera cztery możliwe do wykorzystania przez programistę diody świecące oraz cztery micro przełączniki dwustanowe. Na płycie jest 16 MB synchronicznej dynamicznej pamięci RAM – SDRAM oraz 256 kB pamięci Flash, znajdują sie tam równiez przetwornice napięcia 5V-1.26V (zasilanie rdzeńa procesora C6713) oraz 5V-3.3V (zasilanie pamięci oraz urządzeń peryferyjnych). Maksymalna częstotliwość pracy procesora to 225 Mhz.



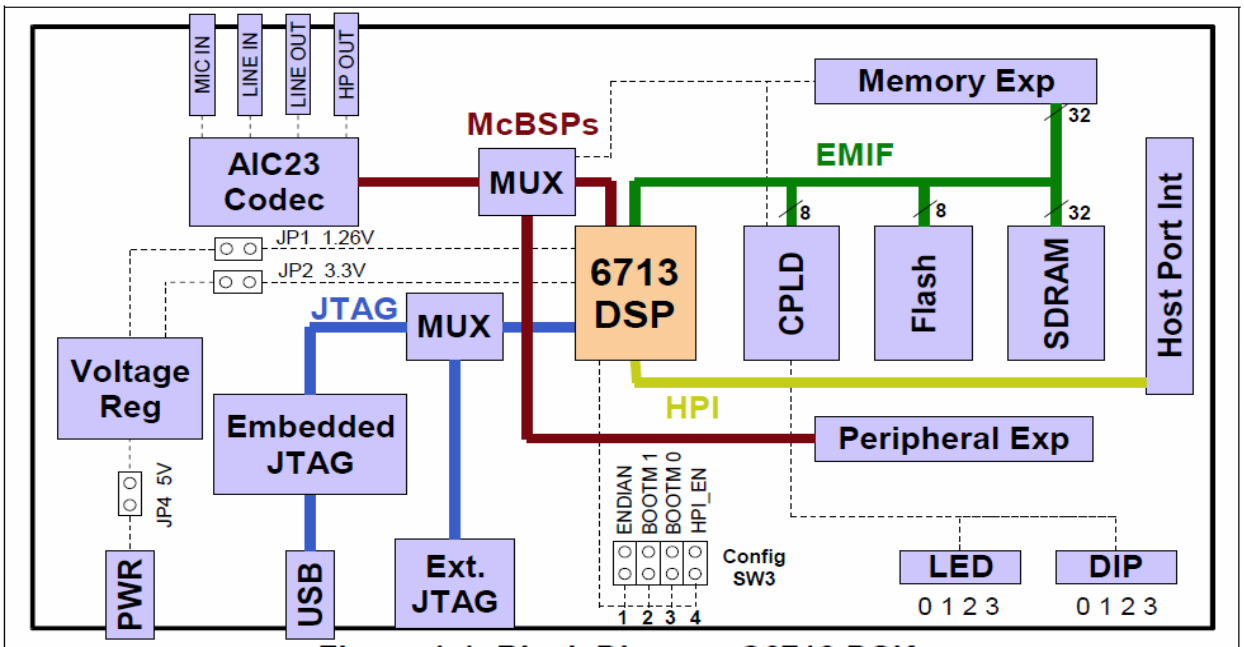
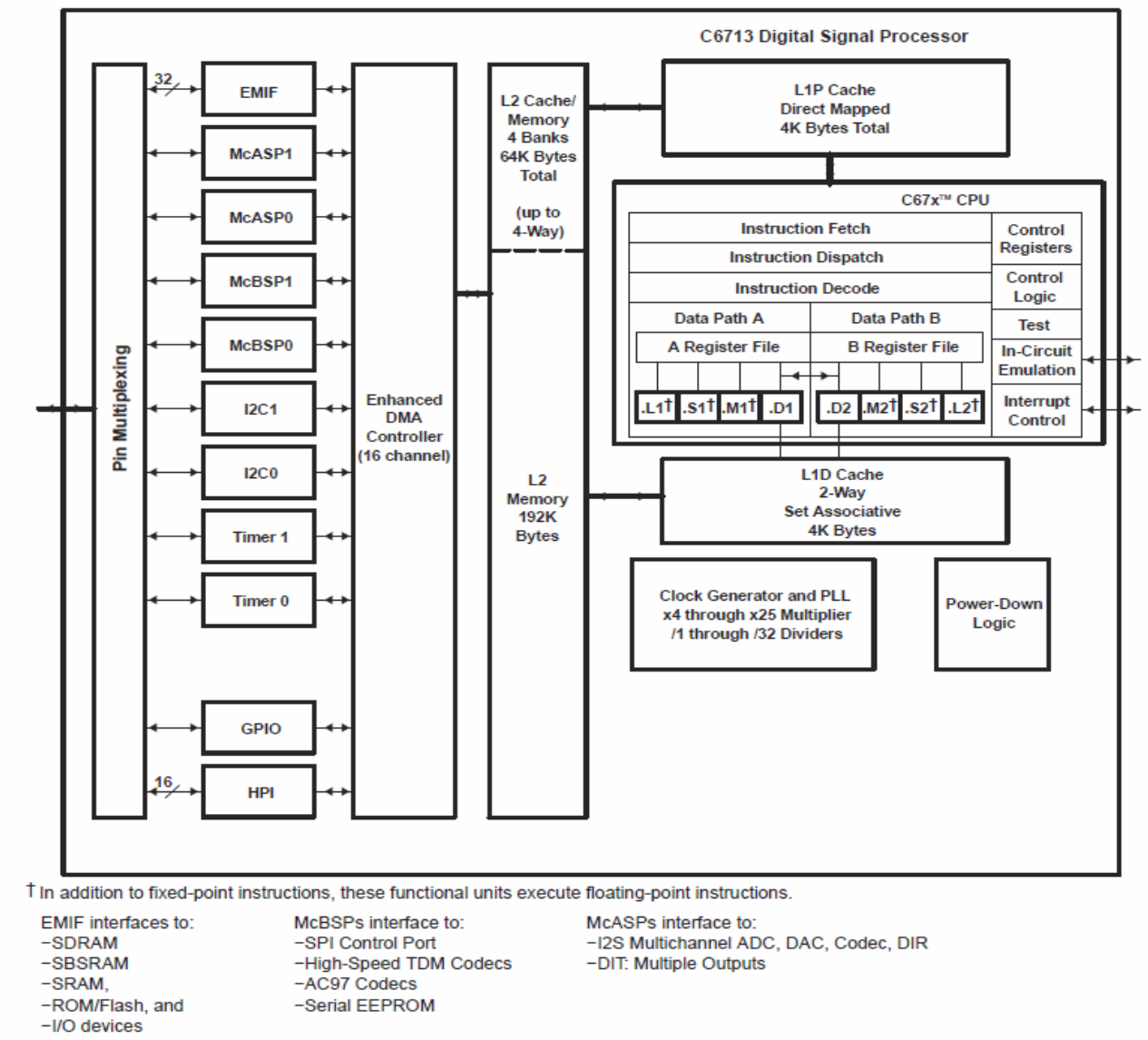


Figure 1-1, Block Diagram C6713 DSK

3. Procesor sygnałowy TMS320C6713

Procesor sygnałowy TMS320C6713 jest oparty na architekturze VLIW (Very Long Instruction Word – bardzo długie słowo instrukcji). Przeznaczony jest do wykonywania intensywnych obliczeń numerycznych – przy maksymalnej częstotliwości zegara 225 MHz procesor może pobierać osiem 32 bitowych instrukcji co 1/225 MHz czyli co 4,44 ns.



C6713 posiada:
- 264 kB wewnętrznej pamięci; 4 kB L1P i 4 kB L2P pamięci podręcznej (Cache) oraz 256 kB pamięci L2 dzielonej pomiędzy programy oraz dane.
- 8 niezależnych jednostek funkcjonalnych (.L1, .L2, .S1, .S2, .M1, .M2, .D1, oraz .D2)
- dwie grupy po 16 rejestrów ogólnego przeznaczenia (A i B),

Procesor TMS320C6713 należy do grupy procesorów zmiennoprzecinkowych (C67xx - C6701, C6711, C6713) ale może też wykonywać operacje stałoprzecinkowe jak procesory z rodziny stałoprzecinkowych C6xxx - C62xx, C64xx.

4. Code Composer Studio

Oprogramowanie instalowane na współpracującym z DSK komputerem PC nosi nazwe Code Composer Studio ver. 3.1. jest zintegrowanym środowiskiem programistycznym pozwalającym na generowanie kodu programu, przy wykorzystaniu kompilatorów języka C, asemblera oraz linkera. Dzięki protokołowi RTDX (real-time data exchange) oraz emulatorowi JTAG (joint team action group) wykorzystującego łacze USB, środowisko to pozwala również na debagowanie programów oraz sterowanie ich pracą w czasie rzeczywistym.

Projekty tworzonych programów należy umieszczać w folderze „C:\CCStudio_v3.1\MyProjects\”

Typy plików generowanych przez CCS:
1. file.pjt: plik konfiguracyjny projektu o nazwie „file”,
2. file.c: plik zawierający kod źródłowy programu w języku C,
3. file.asm: plik zawierający kod źródłowy programu w asemblerze,
4. file.sa: plik zawierający kod źródłowy programu w asemblerze liniowym (linear assembler),
5. file.h: plik nagłówkowy programu w języku C,
6. file.lib: plik zawierający bibliotekę funkcji np. rts6700.lib,
7. file.cmd: plik zawierający komendy linkera,
8. file.obj: plik typu „object" wygenerowany przez asembler,
9. file.out: program wykonywalny utworzony przez linkera przeznaczony do załadowania i wykonania przez procesor C6713,
10. file.cdb: plik konfiguracyjny tworzony gdy wykorzystywane są funkcje DSP/BIOS.

Pliki wymagane przy tworzeniu projektu
1. C6713dskinit.c: Zawiera funkcje do inicjowania DSK: codec'a, portów szeregowych i innych urzadzeń I/O. Nie jest on dostarczany z CCS.
2. C6713dskinit.h: Plik nagówkowy. W nim ustawia się różne opcje np.: wzmocnienie sygnały analogowego na wejściu przetwornika AC, źródło sygnału – czy wejście liniowe czy mikrofonowe itp.
3. C6713dsk.cmd: Plik komend linkera.
4. Vectors_intr.asm: Zmodyfikowana w stosunku do wesji CCS wersja pliku z wektorami przerwań. Przerwania INT4 do INT15, są dostępne dla uzytkownika. Wykorzystuje się zwykle INT11.
5. Vectors_poll.asm: wersja pliku z wektorami przerwań dla programów używających techniki "polling".
6. *rts6700.lib*,*dsk6713bsl.lib*,*csl6713.lib:* biblioteki funkcji odpowiednio: wykonywania programu (run-time support), zarządzania DSK (board support) oraz zarządzania procesorem (chip support). Te pliki sa dostarczane z CCS.

4.1. Szybki test DSK

a) W pamięci flash DSK umieszczony jest program „post.c” (Power on Sef Test).
W chwili włączenia zasilania lub resetu DSK program ten jest ładowany automatycznie i wykonywany jako pierwsze zadanie procesora. Program ten wykorzystuje bibloteke „board support librarry” (BSL) i wykonuje test DSK.
Sprawdzane są:
- pamięci: wewnętrzna, zewnętrzna i flash,
- dwa buforowane wielokanalowe porty szeregowe (multichannel buffered serial ports – McBSP),
- mechanizmy bezposredniego dostępu do pamięci (direct memory acces – DMA),
- audio codec,
- oraz diody świecące.

Jeśli test przeszedł pomyślnie wszystkie cztery diody migają trzykrotnie i pozostają

zaświecone. W trakcie testowania kodeka, przez sekundę generowany jest ton o częstotliwości 1 kHz.

b) Należy uruchomić CCS. W trakcie uruchamiania programu wszystkie diody świecące zgasną. Należy z meny programu uruchomić uruchomić sekwencję poleceń:
GEL -> Check DSK -> Quick Test
Quick Test jest wykonywany dla potwierdzenia poprawności działania DSK.
Gdy test przejdzie pomyślnie otrzymuje się następujące komunikaty
Swiches: 15 - co oznacza, ze DIP switches (0, 1, 2, 3) są w górnym położeniu – $(1,1,1,1)_2$, gdyby w trakcie testu DIP Switch 3 był wciśnięty uzyskano by wartość $(1,1,1,0)_2 = (14)_{10}$
Board revision: 1
CPLD Revision: 2

5. Przykładowy program

Program „loop_stereo.c" wczytuje dane z wejścia analogowego – kanał lewy i prawy i wysyła ten sygnał w postaci niezmienionej na wyjście stereofoniczne.

```
//Loop_stereo.c  Stereo input/output to/from both channels

#include "dsk6713_aic23.h"                                          //codec-DSK
support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;                                  //set sampling rate

#define  LEFT    0
#define  RIGHT  1
union {Uint32 combo; short channel[2];} AIC23_data;

interrupt void c_int11()                                            //interrupt service routine
{
        AIC23_data.combo = input_sample();                          //input 32-bit sample

//      output_left_sample(AIC23_data.channel[LEFT]);     //I/O left channels
        output_sample(AIC23_data.combo);          //I/O left an rigrt channels

        return;
}

void main()                                                         //main function
{
        comm_intr();                          //init DSK, codec, McBSP
        while(1);                     //infinite loop
}
```

##

*Vectors_intr.asm Vector file for interrupt INT11

```
  .global _vectors                    ;global symbols
  .global _c_int00
  .global _vector1
  .global _vector2
  .global _vector3
  .global _vector4
  .global _vector5
  .global _vector6
  .global _vector7
```

```
  .global _vector8
  .global _vector9
  .global _vector10
  .global _c_int11                    ;for INT11
  .global _vector12
  .global _vector13
  .global _vector14
  .global _vector15

  .ref _c_int00                       ;entry address

VEC_ENTRY .macro addr                 ;macro for ISR
   STW   B0,*--B15
   MVKL  addr,B0
   MVKH  addr,B0
   B     B0
   LDW   *B15++,B0
   NOP   2
   NOP
   NOP
  .endm

_vec_dummy:
 B    B3
 NOP  5

 .sect ".vecs"                        ;aligned IST section
 .align 1024
_vectors:
_vector0:   VEC_ENTRY _c_int00   ;RESET
_vector1:   VEC_ENTRY _vec_dummy      ;NMI
_vector2:   VEC_ENTRY _vec_dummy      ;RSVD
_vector3:   VEC_ENTRY _vec_dummy
_vector4:   VEC_ENTRY _vec_dummy
_vector5:   VEC_ENTRY _vec_dummy
_vector6:   VEC_ENTRY _vec_dummy
_vector7:   VEC_ENTRY _vec_dummy
_vector8:   VEC_ENTRY _vec_dummy
_vector9:   VEC_ENTRY _vec_dummy
_vector10:  VEC_ENTRY _vec_dummy
_vector11:  VEC_ENTRY _c_int11        ;ISR address
_vector12:  VEC_ENTRY _vec_dummy
_vector13:  VEC_ENTRY _vec_dummy
_vector14:  VEC_ENTRY _vec_dummy
_vector15:  VEC_ENTRY _vec_dummy

################################################################################

/*C6713dsk.cmd  Linker command file*/

MEMORY
{
```

```
  IVECS:    org=0h,              len=0x220
  IRAM:                  org=0x00000220,      len=0x0002FDE0 /*internal memory*/
  SDRAM:      org=0x80000000, len=0x00100000 /*external memory*/
  FLASH:      org=0x90000000, len=0x00020000 /*flash memory*/
}

SECTIONS
{
  .EXT_RAM :> SDRAM
  .vectors :> IVECS    /*in vector file*/
  .text    :> IRAM     /*Created by C Compiler*/
  .bss     :> IRAM
  .cinit   :> IRAM
  .stack   :> IRAM
  .sysmem  :> IRAM
  .const   :> IRAM
  .switch  :> IRAM
  .far     :> IRAM
  .cio     :> IRAM
  .csldata :> IRAM
}
```

##

```
/*C6713dskinit.h Include file for C6713DSK.C */

#include "dsk6713.h"
#include "dsk6713_aic23.h"

#define LEFT  1              //data structure for union of 32-bit data
#define RIGHT 0               //into two 16-bit data
union {
          Uint32 uint;
          short channel[2];
          } AIC_data;

extern far void vectors();        //external function

static Uint32 CODECEventId, poll;

// This is needed to modify the BSL's data channel McBSP configuration
// See the changes below
MCBSP_Config AIC23CfgData = {
      MCBSP_FMKS(SPCR, FREE, NO)            |
      MCBSP_FMKS(SPCR, SOFT, NO)            |
      MCBSP_FMKS(SPCR, FRST, YES)           |
      MCBSP_FMKS(SPCR, GRST, YES)           |
      MCBSP_FMKS(SPCR, XINTM, XRDY)         |
      MCBSP_FMKS(SPCR, XSYNCERR, NO)        |
      MCBSP_FMKS(SPCR, XRST, YES)           |
      MCBSP_FMKS(SPCR, DLB, OFF)            |
      MCBSP_FMKS(SPCR, RJUST, RZF)          |
      MCBSP_FMKS(SPCR, CLKSTP, DISABLE)     |
      MCBSP_FMKS(SPCR, DXENA, OFF)          |
      MCBSP_FMKS(SPCR, RINTM, RRDY)         |
      MCBSP_FMKS(SPCR, RSYNCERR, NO)        |
      MCBSP_FMKS(SPCR, RRST, YES),
```

```
    MCBSP_FMKS(RCR, RPHASE, SINGLE)        |
    MCBSP_FMKS(RCR, RFRLEN2, DEFAULT)      |
    MCBSP_FMKS(RCR, RWDLEN2, DEFAULT)      |
    MCBSP_FMKS(RCR, RCOMPAND, MSB)         |
    MCBSP_FMKS(RCR, RFIG, NO)              |
    MCBSP_FMKS(RCR, RDATDLY, 0BIT)         |
    MCBSP_FMKS(RCR, RFRLEN1, OF(0))        | // This changes to 1 FRAME
    MCBSP_FMKS(RCR, RWDLEN1, 32BIT)        | // This changes to 32 bits per frame
    MCBSP_FMKS(RCR, RWDREVRS, DISABLE),

    MCBSP_FMKS(XCR, XPHASE, SINGLE)        |
    MCBSP_FMKS(XCR, XFRLEN2, DEFAULT)      |
    MCBSP_FMKS(XCR, XWDLEN2, DEFAULT)      |
    MCBSP_FMKS(XCR, XCOMPAND, MSB)         |
    MCBSP_FMKS(XCR, XFIG, NO)              |
    MCBSP_FMKS(XCR, XDATDLY, 0BIT)         |
    MCBSP_FMKS(XCR, XFRLEN1, OF(0))        | // This changes to 1 FRAME
    MCBSP_FMKS(XCR, XWDLEN1, 32BIT)        | // This changes to 32 bits per frame
    MCBSP_FMKS(XCR, XWDREVRS, DISABLE),

    MCBSP_FMKS(SRGR, GSYNC, DEFAULT)       |
    MCBSP_FMKS(SRGR, CLKSP, DEFAULT)       |
    MCBSP_FMKS(SRGR, CLKSM, DEFAULT)       |
    MCBSP_FMKS(SRGR, FSGM, DEFAULT)        |
    MCBSP_FMKS(SRGR, FPER, DEFAULT)        |
    MCBSP_FMKS(SRGR, FWID, DEFAULT)        |
    MCBSP_FMKS(SRGR, CLKGDV, DEFAULT),

    MCBSP_MCR_DEFAULT,
    MCBSP_RCER_DEFAULT,
    MCBSP_XCER_DEFAULT,

    MCBSP_FMKS(PCR, XIOEN, SP)             |
    MCBSP_FMKS(PCR, RIOEN, SP)             |
    MCBSP_FMKS(PCR, FSXM, EXTERNAL)        |
    MCBSP_FMKS(PCR, FSRM, EXTERNAL)        |
    MCBSP_FMKS(PCR, CLKXM, INPUT)          |
    MCBSP_FMKS(PCR, CLKRM, INPUT)          |
    MCBSP_FMKS(PCR, CLKSSTAT, DEFAULT)     |
    MCBSP_FMKS(PCR, DXSTAT, DEFAULT)       |
    MCBSP_FMKS(PCR, FSXP, ACTIVEHIGH)      |
    MCBSP_FMKS(PCR, FSRP, ACTIVEHIGH)      |
    MCBSP_FMKS(PCR, CLKXP, FALLING)        |
    MCBSP_FMKS(PCR, CLKRP, RISING)
  };


DSK6713_AIC23_Config config = { \
  0x0017, /* Set-Up Reg 0      Left line input channel volume control */  \
        /* LRS     0         simultaneous left/right volume: disabled */\
        /* LIM     0         left line input mute: disabled */         \
        /* XX      00        reserved */                               \
        /* LIV     10111     left line input volume: 0 dB */           \
                                                                       \
  0x0017, /* Set-Up Reg 1      Right line input channel volume control */ \
        /* RLS     0         simultaneous right/left volume: disabled */\
        /* RIM     0         right line input mute: disabled */        \
        /* XX      00        reserved */                               \
        /* RIV     10111     right line input volume: 0 dB */          \
                                                                       \
  0x01f9, /* Set-Up Reg 2      Left channel headphone volume control */   \
        /* LRS     1         simultaneous left/right volume: enabled */ \
        /* LZC     1         left channel zero-cross detect: enabled */ \
```

```
        /* LHV     1111001    left headphone volume: 0 dB */            \
                                                                        \
   0x01f9, /* Set-Up Reg 3       Right channel headphone volume control */  \
        /* RLS     1          simultaneous right/left volume: enabled */ \
        /* RZC     1          right channel zero-cross detect: enabled */\
        /* RHV     1111001    right headphone volume: 0 dB */           \
                                                                        \
   0x0011, /* Set-Up Reg 4       Analog audio path control */              \
        /* X       0          reserved */                               \
        /* STA     00         sidetone attenuation: -6 dB */            \
        /* STE     0          sidetone: disabled */                     \
        /* DAC     1          DAC: selected */                          \
        /* BYP     0          bypass: off */                            \
        /* INSEL   0          input select for ADC: line */             \
        /* MICM    0          microphone mute: disabled */              \
        /* MICB    1          microphone boost: enabled */              \
                                                                        \
   0x0000, /* Set-Up Reg 5       Digital audio path control */            \
        /* XXXXX   00000      reserved */                               \
        /* DACM    0          DAC soft mute: disabled */                \
        /* DEEMP   00         deemphasis control: disabled */           \
        /* ADCHP   0          ADC high-pass filter: disabled */         \
                                                                        \
   0x0000, /* Set-Up Reg 6       Power down control */                    \
        /* X       0          reserved */                               \
        /* OFF     0          device power: on (i.e. not off) */        \
        /* CLK     0          clock: on */                              \
        /* OSC     0          oscillator: on */                         \
        /* OUT     0          outputs: on */                            \
        /* DAC     0          DAC: on */                                \
        /* ADC     0          ADC: on */                                \
        /* MIC     0          microphone: on */                         \
        /* LINE    0          line input: on */                         \
                                                                        \
   0x0043, /* Set-Up Reg 7       Digital audio interface format */        \
        /* XX      00         reserved */                               \
        /* MS      1          master/slave mode: master */              \
        /* LRSWAP  0          DAC left/right swap: disabled */          \
        /* LRP     0          DAC lrp: MSB on 1st BCLK */               \
        /* IWL     00         input bit length: 16 bit */              \
        /* FOR     11         data format: DSP format */                \
                                                                        \
   0x0081, /* Set-Up Reg 8       Sample rate control */                   \
        /* X       0          reserved */                               \
        /* CLKOUT  1          clock output divider: 2 (MCLK/2) */       \
        /* CLKIN   0          clock input divider: 2 (MCLK/2) */        \
        /* SR,BOSR 00000      sampling rate: ADC  48 kHz DAC  48 kHz */  \
        /* USB/N   1          clock mode select (USB/normal): USB */    \
                                                                        \
   0x0001  /* Set-Up Reg 9       Digital interface activation */          \
        /* XX..X   00000000   reserved */                               \
        /* ACT     1          active */                                 \
};

DSK6713_AIC23_CodecHandle hAIC23_handle;

void c6713_dsk_init();
void comm_poll();
void comm_intr();
void output_sample(int);
void output_left_sample(short);
void output_right_sample(short);
Uint32 input_sample();
```

```
short input_left_sample();
short input_right_sample();

##################################################################################

//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

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
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
}
```

##

BIBLIOGRAFIA

R. Chassaing, Digital Signal Processing and Applications with the C6713 and C6416 DSK, Wiley & Sons, Inc., 2005,