Procesory DSP

Laboratorium nr 1

Obsługa strumieni danych

1. Informacje ogólne

Cele ćwiczenia:

- Nauczyć studentów obsługi środowiska programistycznego Code Composer Studio w zakresie umiejętności budowania, konfigurowania i kompilacji projektu dla płyty uruchomieniowej DSK6713
- Zapoznać studentów z zasadami przyjmowania do i wyprowadzania z aplikacji sygnału w postaci strumienia danych portu komunikacyjnego.
- Nauczyć studentów sposobów dostępu do danych w aplikacji obliczeniowej.

Zadania do wykonania w ramach przygotowania do ćwiczenia laboratoryjnego:

a) Zapoznać się z podstawowymi informacjami technicznymi n/t procesora TMS320C6713, w szczególności z mapą pamięci procesora. **Należy wydrukować i przynieść na ćwiczenie:**
- tabelę zawierająca adresy pamięci procesora oraz jego urządzeń peryferyjnych,
- pierwszą stronę niniejszej instrukcji.

b) Na podstawie dokumentacji fabrycznej (www.spectrumdigital.com) zapoznać się ze schematem blokowym oraz ideowym karty uruchomieniowej TMDS320C6713DSK.

c) Wyszukać informacje niezbędne do obsłużenia diod LED oraz przełącznika DIP-SWITCH (adresy urządzeń)

d) Odszukać kodek audio, zapoznać się z dokumentacją tego kodeka (www.ti.com). Zwrócić uwagę na maksymalne dopuszczalne wartości napięć interfejsu analogowego. Skonfrontować to ze schematem ideowym płyty. Jaka jest maksymalna dopuszczalna wartość napięcia wyjściowego,które można ustawić na generatorze aby nie zniszczyć płyty?

e) Odszukać port procesora DSP , do którego jest dołączony kodek audio. Zapoznać się z dokumentacja tego portu (www.ti.com).

f) Zapoznać się z obsługą środowiska Code Composer Studio v5 na podstawie tutoriali dostępnych na stronie www.ti.com

g) Dokładnie przeanalizować aplikacje LabPDSP1_AP1 i LabPDSP1_AP2, w szczególności pliki źródłowe. Przemyśleć niezbędne do wykonania ćwiczenia modyfikacje tych aplikacji .

Zasady oceniania:

Każde z zadań opisanych w dalszej części jest oceniane w skali 0-10 pkt.
Oceny z zadań :

Nr zadania	*Nazwa zadania*	*Liczba punktów*
1.	Wejściówka	
2.	Aplikacja $y_L(n)=x_L(n)$, $y_R(n)=x_L(n)$ wykorzystująca system przerwań	
3.	Linia opóźniająca $y_R(n)=y_L(n)=x_L(n-K)$	
4.	Aplikacja $y_L(n)=x_L(n)+x_R(n)$, $y_R(n)=x_L(n)-x_R(n)$ z wykorzystaniem bufora ping-pong o długości N=128	
5.	Aplikacja $y_L(n)=x_L(n)$, $y_R(n)=x_L(n)$ wykorzystująca kontroler dma i system przerwań	
	Suma punktów	

Wykonanie ćwiczenia jest oceniane wg wzoru:

$$Liczbapunktów = 10\frac{Sumapunktówuzyskanychzzadań}{Liczbapunktówmożliwychdozdobycia}$$

Sposoby obsługi strumienia danych w ćwiczeniu laboratoryjnym
Strumień danych portu komunikacyjnego może zostać przekazany do aplikacji na kilka sposobów

1. Próbka po próbce
   a) poprzez przepatrywanie portu
   b) z użyciem systemu przerwań
2. Blokowo z użyciem techniki bezpośredniego dostępu do pamięci
   a) poprzez przepatrywanie kontrolera DMA
   b) z użyciem systemu przerwań

Wykorzystanie danego sposobu zależy od stopnia złożoności numerycznej aplikacji rozumianej jako czas zużyty na obliczenia podzielony przez dostępny czas procesora. Generalnie obsługa próbka po próbce zużywa znacznie więcej czasu niż obsługa blokowa z użyciem kanałów DMA. W każdym z przypadków użycie systemu przerwań zmniejsza czas niezbędny na obsługę kanału komunikacyjnego do niezbędnego minimum.

1.1. Program obsługujący dane w systemie próbka po próbce z przepatrywaniem

Struktura programu jest pokazana poniżej:

```
1. void main(void){
2.       Int16 słowoDanych;
3.       niezbędneInicjalizacje();
4.       for(;;){
5.             if( portMaNoweDane()){
6.                   słowoDanych = pobierzDaneZPortu();
7.                   słowoDanych = obliczenia(słowoDanych);
8.                   wyślijDaneDoPortu( słowoDanych );
9.             }
10.            }
11.      }
```

Klasyczna struktura programu wykonującego obliczenia w czasie rzeczywistym opiera się na wykorzystaniu nieskończonej pętli, w której program oczekuje na pojawienie się nowych danych a następnie pobiera je i wykonuje na nich obliczenia. Dotyczy to zarówno systemów mikroprocesorowych jak i systemów przetwarzania sygnałów. Reżim czasowy w systemie przetwarzania sygnałów jest wyznaczony przez rytm próbkowania. Np. dla systemu z kodekiem audio z próbkowaniem o częstotliwości f=48kHz pobranie danych obliczenia i wysłanie wyników musi trwać krócej niż 1/f a więc maksymalnie 20,8μs. Należy zwrócić uwagę iż ciągłe odwoływanie się do portu stanowi stratę czasu, który potencjalnie może zostać wykorzystany na wykonywanie obliczeń. Należy pamiętać, że system MUSI pobrać próbkę wejściową i wypracować wyjściową przed nadejściem kolejnej. Jeżeli to nie nastąpi system gubi próbkę wejściową a próbka wyjściowa przyjmuje wartość ostatnio wypracowanej. Wskutek tego w sygnale pojawiają się nieakceptowalne zniekształcenia.

W systemie z przepatrywaniem nie ma możliwości buforowania danych w większe bloki i wykonywania obliczeń na tych blokach w czasie rzeczywistym ponieważ w czasie wykonywania

obliczeń nie jest obsługiwany port wejścia - wyjścia. Tej wady nie ma system wykorzystujący przerwania.

1.2. Program obsługujący dane w systemie próbka po próbce z wykorzystaniem przerwań

Program wykorzystujacy przerwania musi mieć w swej budowie procedurę obsługi przerwania, tzw handler, tu nazywany `portIO_ISR()`

Struktura programu jest pokazana poniżej:

```
1. Uint16 dataAvailableFlag=NULL;
2. Int16 dataIn, dataOut;
3. void interrupt portIO_ISR(void){
4.       dataIn = pobierzDaneZPortu();
5.       wyślijDaneDoPortu( dataOut );
6.       dataAvailableFlag = NEW_RX_DATA;
7. }
8.
9. void main(void){
10.            niezbędne inicjalizacje();
11.            for(;;){
12.                  if( dataAvailableFlag == NEW_RX_DATA){
13.                              dataOut  = obliczenia(dataIn);
14.                  }
15.            }
16.      }
```

Program wykorzystuje zmienne globalne dataIn oraz dataOut które służą do buforowania próbek. Program korzysta z założenia, że nadajnik i odbiornik portu pracuja z tymi samymi zegarami. Wówczas możliwe jest pominiecie przerwania od nadajnika a dane do niego wpisywać w przerwaniu od odbiornika portu. Zmienna globalna `dataAvailableFlag` służy do przekazania informacji z handlera przerwania do aplikacji o nadejściu nowych danych. Ustawienie tych danych inicjuje proces obróbki danych.

W linii nr 3 kodu wykorzystano słowo kluczowe `interrupt`. Informuje ono kompilator o tym, że funkcja musi zachowywać na stosie pełen kontekst programu.

1.3. Program obsługujący dane blokowo z wykorzystaniem techniki bezpośredniego dostępu do pamięci z przepatrywaniem kontrolera DMA

Kontroler DMA po wystąpieniu określonego zdarzenia *(eventu)* automatycznie kopiuje dane z jednego obszaru pamięci do drugiego. W ćwiczeniu laboratoryjnym zdarzeniem tym jest odbiór znaku przez port komunikacyjny. Po odpowiednim zaprogramowaniu kontroler kolejno wypełnia bufor danych, który zachowuje się jak bufor kołowy. Kontroler DMA sygnalizuje poprzez wewnętrzne flagi zapełnienie ½ bufora i całego bufora. Po zapełnieniu całego bufora dane są wpisywane od początku. Flagi mogą zostać odczytane przez aplikację lub mogą być źródłem przerwania.

Inicjalizacje kontrolera DMA powinny znaleźć się w funkcji `niezbędneInicjalizacje()` wywoływanej w linii 4.

Struktura programu jest pokazana poniżej:

```
1. Int16 dataIn[BUFFER_LEN], dataOut[BUFFER_LEN];
2.
3. void main(void){
4.       niezbędneInicjalizacje();
5.       for(;;){
6.             if( czytajFlaga05edma3()==1){
7.                         obliczenia(&dataIn[0], &dataOut[0]);
8.             }
9.             else if( czytajFlaga1edma3()==1){
10.                         obliczenia(&dataIn[BUFFER_LEN/2],
         &dataOut[BUFFER_LEN/2]);
11.                }
12.            }
13.      }
```

Podobnie jak w przypadku obsługi próbka po próbce system MUSI nadążyć z obliczeniami zanim wypełni się kolejna porcja danych wejściowych.

Należy zwrócić uwagę na zmieniony interfejs funkcji `obliczenia`, który przekazuje do funkcji adresy buforów wejściowego i wyjściowego, na których funkcja będzie wykonywać obliczenia. Dzięki temu funkcja może operować na danych zawartych w całym buforze.

1.4. Program obsługujący dane blokowo z wykorzystaniem techniki bezpośredniego dostępu do pamięci i systemu przerwań

Struktura programu jest pokazana poniżej:

```
1. Uint16 bufferAvailableFlag=NULL;
2. Int16 dataIn[BUFFER_LEN], dataOut[BUFFER_LEN];
3. void interrupt edma3_ISR(void){
4.       if(flaga05==1){
5.              bufferAvailableFlag = BUFFER_05;
6.       }
7.       else {
8.             bufferAvailableFlag = BUFFER_1;
9.             }
10.      }
11.
12.      void main(void){
13.            niezbędneInicjalizacje();
14.            for(;;){
15.                  if( bufferAvailableFlag == BUFFER_05){
16.                        obliczenia(&dataIn[0], &dataOut[0]);
17.                }
```

```
18.                 else if( bufferAvailableFlag == BUFFER_1){
19.                             obliczenia(&dataIn[BUFFER_LEN/2],
                                &dataOut[BUFFER_LEN/2]);
20.                 }
21.             }
22.     }
```

Program wykorzystuje bufory globalne dataIn oraz dataOut które służą do buforowania próbek. Program korzysta z założenia, że nadajnik i odbiornik portu pracuja z tymi samymi zegarami. Zmienna globalna `bufferAvailableFlag` służy do przekazania informacji z handlera przerwania do aplikacji o dostępności nowej porcji danych. Ustawienie tej zmiennej inicjuje proces obróbki danych.

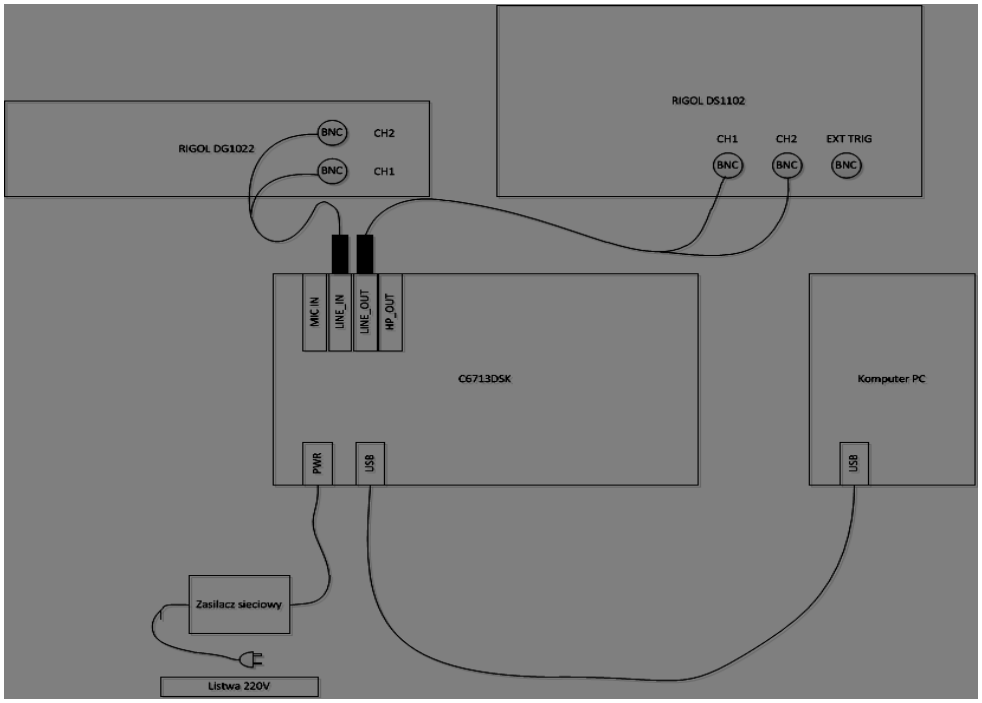
2. Wykonanie ćwiczenia

Wykonanie ćwiczenia polega na modyfikowaniu aplikacji przykładowych LabPDSP1_AP1 i LabPDSP1_AP2.

Do wykonania ćwiczenia będą wykorzystane następujące elementy:

- Karta uruchomieniowa DSK6713
- Generator przebiegów arbitralnych Rigol typ: DG1022
- Oscyloskop Rigol typ: DS1102
- Zestaw kabli.

Należy zmontować układ zgodnie z rysunkiem.



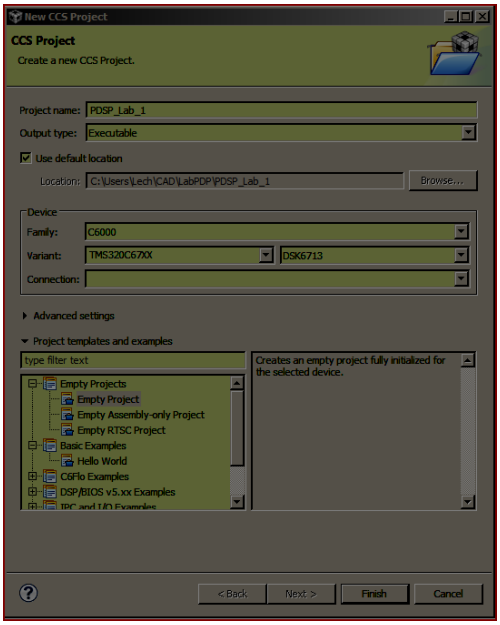
Uwaga: PRZED WŁĄCZENIEM ZASILANIA DO ELEMENTÓW STANOWISKA UZYSKAĆ ZGODĘ PROWADZĄCEGO.

2.1. Uruchomienie środowiska programistycznego

Ćwiczenie laboratoryjne należy rozpocząć od uruchomienia aplikacji Code Composer Studio. Skrót

do aplikacji  znajduje się na pulpicie. Środowisko jest oparte o IDE „Eclipse” na licencni GPL. Po otwarciu aplikacji należy w zakładce File wybrać opcję „Switch Workspace” i wpisać w katalogu C:\LABPDSP\ nazwę własną w formacie NRGRUPY_INICJAŁY. Środowisko utworzy katalog o nazwie NRGRUPY_INICJAŁY, który będzie dostępny na dysku komputera do zakończenia semestru. Po zakończeniu pracy zawsze należy skopiować zawartość katalogu na własny dysk przenośny.

Wybrać zakładkę File > New > CCS Project oraz wybrać następujące opcje utworzenia projektu;



- Project name = „PDSP_LAB1”
- Output type = Executable
- Device Family = C6000
- Variant = TMS320C67XX, DSK6713
- Connection = Spectrum Digital DSK-EVM-eZdsp onboard Emulator
- Project tremplates and Examples = Empty Project

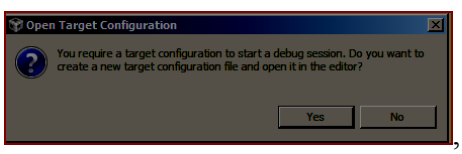
Środowisko w katalogu NRGRUPY_INICJAŁY utworzy podkatalog „PDSP_LAB1” , a w nim plik main.c otwarty również w oknie edytora.

W pliku main.c należy wpisać poniższy kod

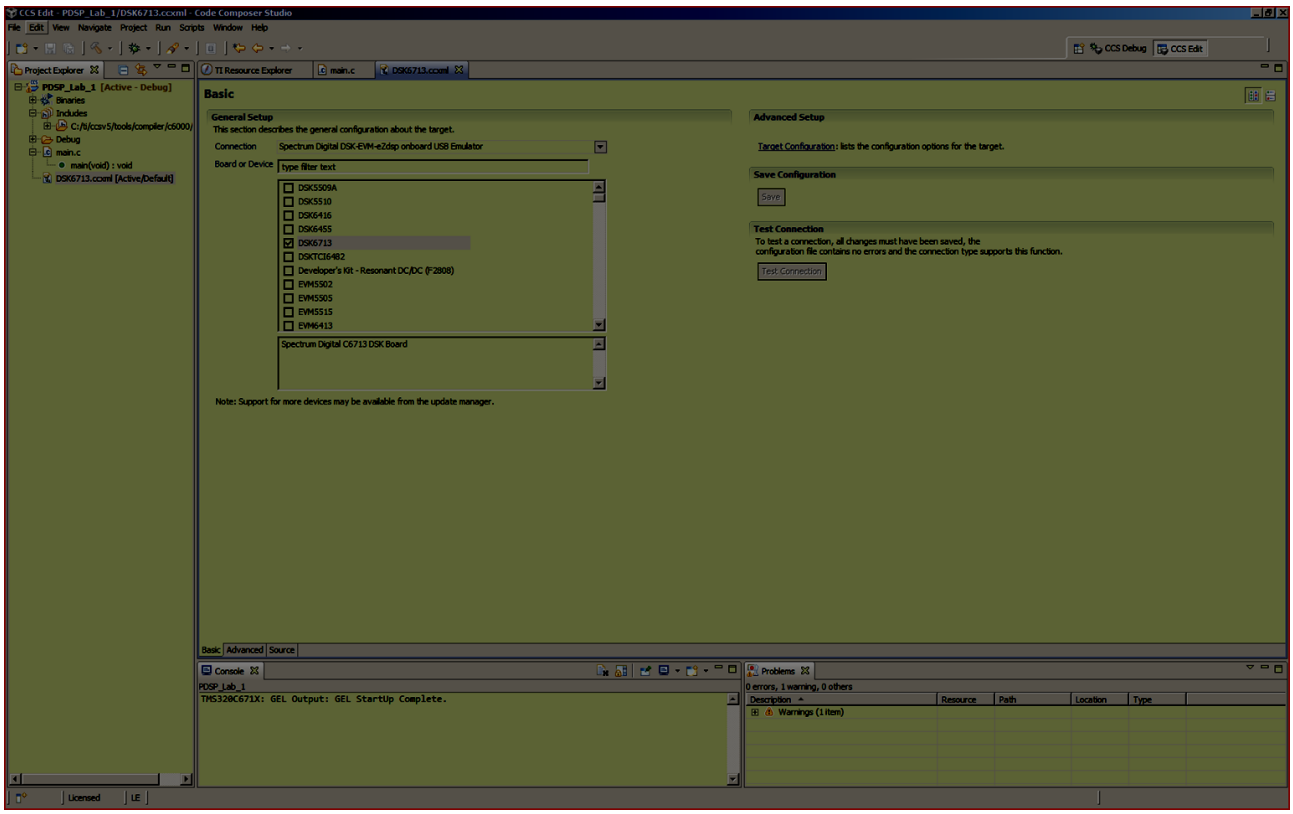
```
#include <stdio.h>

void main(void) {
	printf("Hello world\n");
}
```

Następnie należy wcisnąć ikonę „Debug”. Środowisko IDE wyprowadzi poniższy komunikat:



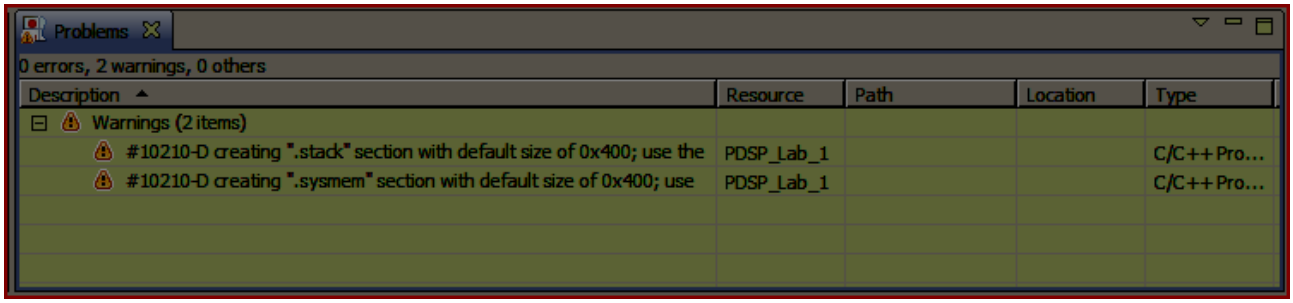
, który należy potwierdzić. Utworzony zostanie plik konfiguracyjny środowiska debugowania. Należy nadać mu nazwę DSK6713.ccxml. Nastąpi otwarcie okna konfiguracji



Należy wybrać następujące opcje:

- Connection = Spectrum Digital DSK-EVM-eZdsp onboard USB Emulator
- Board or Device = DSK6713

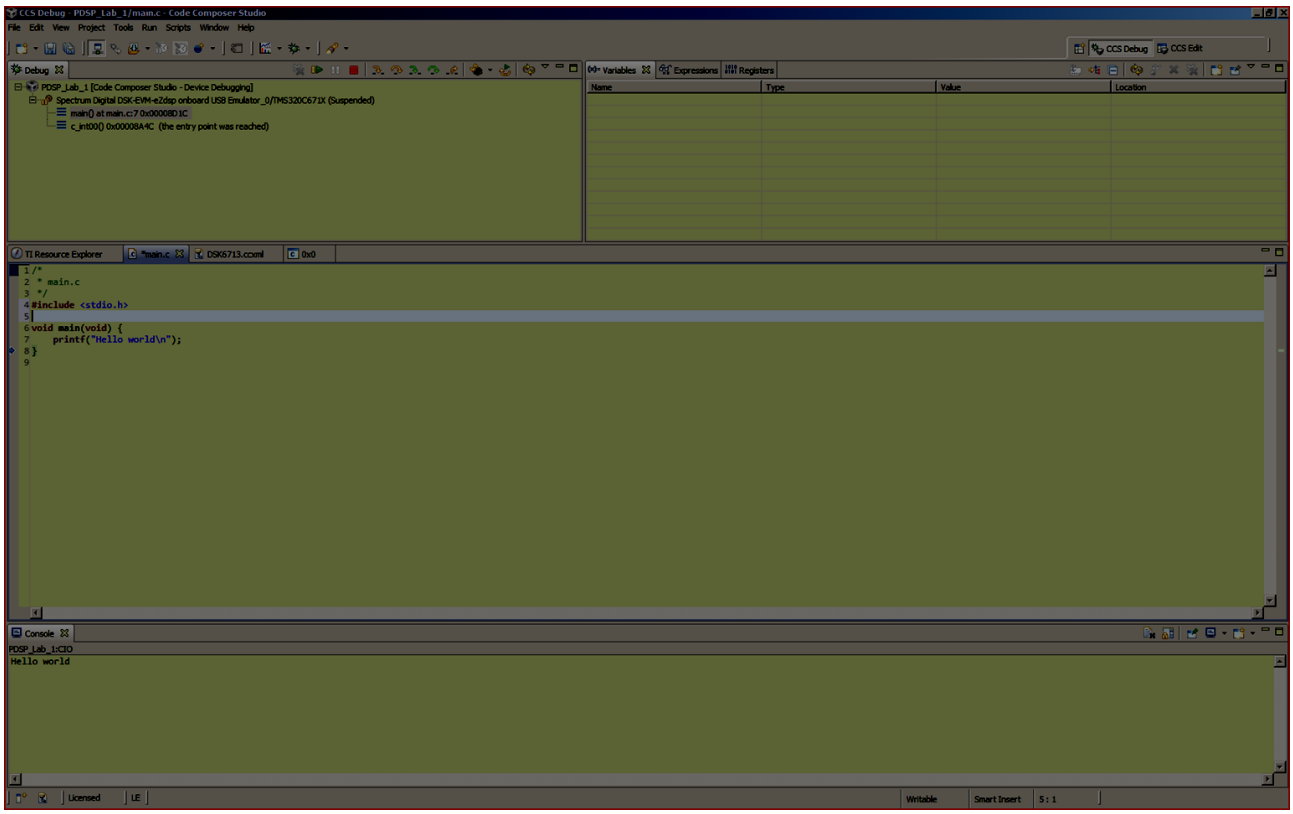
a następnie zapisać konfigurację. Z kolei należy przejść do okna `main.c` i skompilować plik. Ewentualne błędy należy poprawić. Środowisko informuje o błędach i ostrzeżeniach w oknie Problems. Na obecnym etapie można zignorować poniższe ostrzeżenia:



Usunięcie tych ostrzeżeń wymaga umieszczenia w projekcie skryptu konfiguracji pamięci, który jest częścią aplikacji przykładowych.

Kolejną czynnością jest uruchomienie debuggera. Środowisko IDE zmienia perspektywę na CCS Debug. Zmiana perspektywy jest możliwa po wybraniu odpowiedniej ikony w prawym górnym rogu aplikacji. Uruchomienie aplikacji jest możliwe w trybie „Step into", „Step over", „Step return", „Run to line", lub „Run". Czynności te są dostępne po wciśnięciu odpowiednich kontrolek lub z poziomu menu Run>.

Uruchom aplikację. Program powinien wypisać na konsoli tekst umieszczony jako wywołanie funkcji `printf`.



2.2. Testowanie aplikacji LabPDSP1_AP1(zadanie nie podlega ocenie)

Zadanie polega na uruchomieniu środowiska CCSv5 oraz importowaniu projektu LabPDSP1_AP1 do środowiska. Następnie należy skompilować projekt oraz uruchomić debugger. Po wydaniu komendy „run" należy ocenić funkcjonowanie aplikacji Wszelkie trudności podczas uruchamiania należy zgłaszać prowadzącemu.

2.3. Aplikacja $y_L(n)=x_L(n)$, $y_R(n)=x_L(n)$ wykorzystująca system przerwań

Wykonanie zadania polega na:

- uzupełnieniu tablicy wektorów przerwań w pliku `vecs_int.asm` odwołaniem do handlera przerwania
- Zmodyfikowaniu funkcji `niezbędneInicjalizacje()`
- napisaniu handlera przerwania. Funkcje niezbędne do pracy z systemem przerwań znajdują się w bloku komentarza „informacje dla studentów"
- zmodyfikowaniu funkcji `main()` tak, by zamiast przepatrywać port badała flagę ustawiana przez przerwanie.
- uruchomieniu i sprawdzeniu działania aplikacji

2.4. Linia opóźniająca $y_R(n)=y_L(n)=x_L(n-K)$

Należy tak zmodyfikować funkcję `obliczenia` aby realizowała funkcję $y_R(n)=x_R(n)$; $y_L(n)=x_L(n-K)$ dla K zadawanego

- w fazie kompilacji programu
- jako wielokrotność liczby odczytanej z przełącznika DIP-SWITCH

Działanie aplikacji ocenić przy użyciu oscyloskopu. Przebieg wejściowy z generatora powinien mieć kształt sinusoidalny lub prostokątny. Ocenić opóźnienie wnoszone w kanale lewym w stosunku do prawego.

2.5. Aplikacja $y_L(n)=x_L(n)+x_R(n)$, $y_R(n)=x_L(n)-x_R(n)$ z wykorzystaniem bufora ping-pong o długości N=128

Należy tak zmodyfikować funkcję bufory `dataIn, dataOut` oraz handler przerwania `portIO_ISR` aby kolekcjonowały dane w buforach o długości BUFFER_LEN zadanej przez prowadzącego. Obliczenia na blokach danych wykonać na zmiennych typu Int16 w funkcji `obliczenia.`

Zaobserwować z użyciem oscyloskopu działanie aplikacji. Ocenić działanie w przypadku gdy wynik obliczeń przekracza zakres arytmometru.

2.6. Aplikacja $y_L(n)=x_L(n)$, $y_R(n)=x_L(n)$ wykorzystująca kontroler dma i system przerwań

Wykonanie zadania polega na:

- uzupełnieniu tablicy wektorów przerwań w pliku `vecs_int.asm` odwołaniem do handlera przerwania
- Zmodyfikowaniu funkcji `niezbędneInicjalizacje()`
- napisaniu handlera przerwania. Funkcje niezbędne do pracy z systemem przerwań znajdują się w bloku komentarza „informacje dla studentów”
- zmodyfikowaniu funkcji `main()` tak, by zamiast przepatrywać port badała flagi ustawiane przez przerwanie.
- uruchomieniu i sprawdzeniu działania aplikacji

3. Aplikacja przykładowa

3.1. Plik „PDSP_lab1.c”

```
#include "dsk6713_aic23.h"                    //support file for codec,DSK
Uint32 fs = DSK6713_AIC23_FREQ_8KHZ;//set sampling rate
short  loop = 0;                              //table index
short gain = 10;                                        //gain factor
short sine_table[8]={0,707,1000,707,0,-707,-1000,-707};//sine values

/*
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()
{
 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(0);                       //turn LED #0 ON
    output_sample(sine_table[loop]*gain);//output for on-time sec
    if (loop < 7) ++loop;                    //check for end of table
    else loop = 0;                           //reinit loop index
  }
  else DSK6713_LED_off(0);                   //turn LED off if not pressed
 }                                  //end of while (1) infinite loop
}
```

3.2. Plik c6713dskinit.c

```
//C6713dskinit.c Plik wykorzystuje biblioteki C6713 CSL (Chip Support Library) i
C6713DSK BSL (Board Support Library) (TI)

#include "c6713dskinit.h"
#define using_bios
extern Uint32 fs;                       //częstotliwość próbkowania

/*
 * Inicjalizacja peryferiów DSP
 */

void c6713_dsk_init()
{
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
}
```

3.3. Plik c6713dskinit.h

```
/*
 * c6713dskinit.h
 *
 */

#ifndef C6713DSKINIT_H_
#define C6713DSKINIT_H_

/*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();



#endif /* C6713DSKINIT_H_ */
```

3.4. Plik dsk6713_aic23.h

```
/*
 * dsk6713_aic23.h
 *
```

```
 */

#ifndef DSK6713_AIC23_H_
#define DSK6713_AIC23_H_
/*
 *  Copyright 2002 by Spectrum Digital Incorporated.
 *  All rights reserved. Property of Spectrum Digital Incorporated.
 */

/*
 *  ======== dsk6713_aic23.h ========
 *
 *  Codec interface for AIC23 on the DSK6713 board
 */
#ifndef DSK6713_AIC23_
#define DSK6713_AIC23_

#ifdef __cplusplus
extern "C" {
#endif

#include <csl.h>
#include <csl_mcbsp.h>

/* AIC23 McBSP defines */
#define DSK6713_AIC23_CONTROLHANDLE  DSK6713_AIC23_codeccontrolhandle
#define DSK6713_AIC23_DATAHANDLE     DSK6713_AIC23_codecdatahandle

/* McBSP handles */
extern MCBSP_Handle DSK6713_AIC23_DATAHANDLE;
extern MCBSP_Handle DSK6713_AIC23_CONTROLHANDLE;

/* Codec module definitions */
#define DSK6713_AIC23_NUMREGS         10
#define DSK6713_AIC23_LEFTINVOL       0
#define DSK6713_AIC23_RIGHTINVOL      1
#define DSK6713_AIC23_LEFTHPVOL       2
#define DSK6713_AIC23_RIGHTHPVOL      3
#define DSK6713_AIC23_ANAPATH         4
#define DSK6713_AIC23_DIGPATH         5
#define DSK6713_AIC23_POWERDOWN       6
#define DSK6713_AIC23_DIGIF           7
#define DSK6713_AIC23_SAMPLERATE      8
#define DSK6713_AIC23_DIGACT          9
#define DSK6713_AIC23_RESET           15

/* Frequency Definitions */
#define DSK6713_AIC23_FREQ_8KHZ       1
#define DSK6713_AIC23_FREQ_16KHZ      2
#define DSK6713_AIC23_FREQ_24KHZ      3
#define DSK6713_AIC23_FREQ_32KHZ      4
#define DSK6713_AIC23_FREQ_44KHZ      5
#define DSK6713_AIC23_FREQ_48KHZ      6
#define DSK6713_AIC23_FREQ_96KHZ      7

/* Codec Handle */
typedef int DSK6713_AIC23_CodecHandle;

/* Parameter Structure for the DSK6713 AIC23 Codec */
typedef struct DSK6713_AIC23_Config {
    int regs[DSK6713_AIC23_NUMREGS];
} DSK6713_AIC23_Config;
```

```
#define DSK6713_AIC23_DEFAULTCONFIG { \
    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 */           \
                                                                             \
    0x00d8, /* Set-Up Reg 2       Left channel headphone volume control */   \
            /* LRS     0          simultaneous left/right volume: enabled */ \
            /* LZC     1          left channel zero-cross detect: enabled */ \
            /* LHV     1011000    left headphone volume: 0 dB */             \
                                                                             \
    0x00d8, /* Set-Up Reg 3       Right channel headphone volume control */  \
            /* RLS     0          simultaneous right/left volume: enabled */ \
            /* RZC     1          right channel zero-cross detect: enabled */\
            /* RHV     1011000    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 */                                 \
}

/* Set codec register regnum to value regval */
void DSK6713_AIC23_rset(DSK6713_AIC23_CodecHandle hCodec, Uint16 regnum, Uint16
regval);

/* Return value of codec register regnum */
Uint16 DSK6713_AIC23_rget(DSK6713_AIC23_CodecHandle hCodec, Uint16 regnum);

/* Open the codec with id and return handle */
DSK6713_AIC23_CodecHandle DSK6713_AIC23_openCodec(int id, DSK6713_AIC23_Config
*Config);

/* Close the codec */
void DSK6713_AIC23_closeCodec(DSK6713_AIC23_CodecHandle hCodec);

/* Configure the codec register values */
void DSK6713_AIC23_config(DSK6713_AIC23_CodecHandle hCodec, DSK6713_AIC23_Config
*Config);

/* Write a 32-bit value to the codec */
Int16 DSK6713_AIC23_write(DSK6713_AIC23_CodecHandle hCodec, Uint32 val);

/* Read a 32-bit value from the codec */
Int16 DSK6713_AIC23_read(DSK6713_AIC23_CodecHandle hCodec, Uint32 *val);

/* Set the codec output gain */
void DSK6713_AIC23_outGain(DSK6713_AIC23_CodecHandle hCodec, Uint16 outGain);

/* Set the codec loopback mode */
void DSK6713_AIC23_loopback(DSK6713_AIC23_CodecHandle hCodec, Int16 mode);

/* Enable/disable codec mute mode */
void DSK6713_AIC23_mute(DSK6713_AIC23_CodecHandle hCodec, Int16 mode);

/* Enable/disable codec powerdown modes for DAC, ADC */
void DSK6713_AIC23_powerDown(DSK6713_AIC23_CodecHandle hCodec, Uint16 sect);

/* Set the codec sample rate frequency */
void DSK6713_AIC23_setFreq(DSK6713_AIC23_CodecHandle hCodec, Uint32 freq);

#ifdef __cplusplus
}
#endif

#endif
```

3.5. *Plik dsk613.h*

```
/*
 *  Copyright 2002 by Spectrum Digital Incorporated.
 *  All rights reserved. Property of Spectrum Digital Incorporated.
 */

/*
```

```
 *  ======== dsk6713.h ========
 *
 *  This files contains DSK6713 board specific I/O registers
 *  define for the CPLD.
 */

#ifndef DSK6713_
#define DSK6713_

#ifdef __cplusplus
extern "C" {
#endif

#include <csl.h>

/*
 *  Note:  Bit definitions for each register field
 *         needs to be supplied here for the CPLD
 *         and other board peripherals.
 */

/* Compatability definitions */
#define NULL                0

/* CPLD address definitions */
#define DSK6713_CPLD_BASE    0x90080000

/* CPLD Register Indices */
#define DSK6713_USER_REG     0
#define DSK6713_DC_REG       1
#define DSK6713_VERSION      4
#define DSK6713_MISC         6

/* CPLD Register Bits */
#define DC_DET               0x80
#define DC_STAT1             0x20
#define DC_STAT0             0x10
#define DC_CNTL1             0x02
#define DC_CNTL0             0x01

#define TIN1SEL              0x08
#define TIN0SEL              0x04
#define MCBSP2SEL            0x02
#define MCBSP1SEL            0x01

/* Initialize all board APIs */
void DSK6713_init();

/* Read an 8-bit value from a CPLD register */
Uint8 DSK6713_rget(Int16 regnum);

/* Write an 8-bit value to a CPLD register */
void DSK6713_rset(Int16 regnum, Uint8 regval);

/* Spin in a delay loop for delay iterations */
void DSK6713_wait(Uint32 delay);

/* Spin in a delay loop for delay microseconds */
void DSK6713_waitusec(Uint32 delay);

/* Get the DSK version */
Int16 DSK6713_getVersion();
```

```
#ifdef __cplusplus
}
#endif

#endif
```

3.6. Plik C6713dsk.cmd

```
/*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
}
```