ANEXA 1

```
/*********************************************************************/
/*     goertzel.c     */

/*   Aplicatia foloseste algoritmul Goertzel pentru a identifica tonurile de
apelare ale unui telefon; rata de esantionare este  8000 Hz (8 kHz) */

#include <stdio.h>                          // Folosit pentru printf() si puts()
#include "goertzelcfg.h"

#include "dsk5416.h"
#include "dsk5416_pcm3002.h"

#include "bargraph.h"                       //folosit pt afisajul cu LED-uri
#include "switches.h"                       //intrerupatoarele de pe placa –
                                        permit selectarea modului de operare (1-9)
#include "goertzel.h"                       //definire butoane

#define GOERTZEL_THRESHOLD 0x100            //Suma puterilor componentelor
                        spectrale ale frecventelor de pe linii si de pe coloana
 (generate la apasarea unui buton) trebuie sa fie mai mare decat aceasta limita

/*********************************************************************/
/* Configurarea registrilor PCM3002 */
/*********************************************************************/

DSK5416_PCM3002_Config setup = {             //configurarea implicita

    0x1FF,      // programare Registru 0 - atenuarea CDA pt canalul stang
                // LDL=1 -> nivelul iesirii este controlat de campul AL
                                                              (bitii 7:0)
                // AL=FFh -> atenuare canal stang 0dB

    0x3FF,      // programare Registru 1 - atenuarea CDA pt canalul drept
                // LDL=1 -> nivelul iesirii este controlat de campul AL
                                                              (bitii 7:0)
                // AL=FFh -> atenuare canal stang 0dB

    0x0,        // programare Registru 2
                  // Filtrul trece sus al CAD este activat
                  // Modul Power-Down e dezactivat
                  // Modul detectie zero infinit dezactivat
                  // Iesirile CDA active (operatie normala)
                  // Modul soft Mute dezactivat (implicit)

    0x0           // programare Registru 3
                  // DAC pe 16 biti, MSB primul
                  // ADC pe 16 biti, MSB primul
                  // Modul Loop-Back dezactivat
                  // canalul stanga este “H”, canalul dreapta este “L”.
};

/*********************************************************************/
/* Pentru compatibilitate cu functiile read / write ale pcm3002  */
/* aceste variabile trebuie declarate Int16 sau short int si nu int */
/*********************************************************************/

Int16 left_input;
Int16 left_output;
```

```
Int16 right_input;
Int16 right_output;
Int16 mono_input;

//variabile pt puterea de iesire calculata la aceste frecvente
short int goertzel_output_697_Hz;
short int goertzel_output_770_Hz;
short int goertzel_output_852_Hz;
short int goertzel_output_941_Hz;

short int goertzel_output_1209_Hz;
short int goertzel_output_1336_Hz;
short int goertzel_output_1477_Hz;
short int goertzel_output_1633_Hz;

//variabile folosite pt calcularea recursiva a coeficientilor "v"
short int delay_697_Hz[3] = { 0, 0, 0 };
short int delay_770_Hz[3] = { 0, 0, 0 };
short int delay_852_Hz[3] = { 0, 0, 0 };
short int delay_941_Hz[3] = { 0, 0, 0 };

short int delay_1209_Hz[3] = { 0, 0, 0 };
short int delay_1336_Hz[3] = { 0, 0, 0 };
short int delay_1477_Hz[3] = { 0, 0, 0 };
short int delay_1633_Hz[3] = { 0, 0, 0 };

short int goertzel_bit_mask = 0; //bitii acestei variabile sunt indicatori ai
frecventelor valide receptionate

/************************************************************************
Aceasta procedura nu este apelata de main(). Ea este apelata de DSP/BIOS
************************************************************************/

void UserTask()
{
DSK5416_PCM3002_CodecHandle hCodec;
unsigned long i;
unsigned int switch_value;
unsigned int counter = 0;
short int goertzel_power;
unsigned int N = 0;

/* initializeaza codecul */
hCodec = DSK5416_PCM3002_openCodec(0, &setup);

/* Afisaza detalile la Stdout */
puts("TMS320C5416 DSK: Goertzel algorithm to identify touch tones of
telephone.\n");

for ( i = 0 ; i < 12000000 ; i++ )
 {                                        // Citeste canalul de intrare stanga
  while (!DSK5416_PCM3002_read16(hCodec, &left_input));
                                           // Reda la canalul de iesire stanga
  while (!DSK5416_PCM3002_write16(hCodec, left_output));
                                          // Citeste canalul de intrare dreapta
  while (!DSK5416_PCM3002_read16(hCodec, &right_input));
                                           // Reda la canalul de iesire dreapta
  while (!DSK5416_PCM3002_write16(hCodec, right_output));
                              /* Citeste valorile intrerupatoarelor de pe
                                        placa si afisaza valoarea la Stdout */
```

```
 switch_value = switch_status_display();
                                    /* genereaza intrare mono de la intrarile
                                                              stanga si dreapta */
 mono_input = stereo_to_mono (left_input, right_input);
                                             /* proceseaza unul din 6 esantioane
                                             e.g. luate la frecventa de 8 kHz */
                                  //valoarea lui counter asigura ca numai unul
                                           din cele 6 esantioane este prelucrat
if (0 == counter)
   {if (0 == switch_value)          //Valoare intrerupatoare = 0.
        {left_output = mono_input;  //redare semnale mono in difuzoare
         right_output = mono_input;
        }
   else if (1 == switch_value)                  /* Valoare intrerupatoare = 1.
                                                     Filtru Goertzel la 693 Hz.
                                        Primul rand al tastaturii telefonului */
       {left_output = goertzel_value (mono_input, COEFFICIENT_697_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
        right_output = left_output;          /* folosit pt testarea acuratetei
                                                        frecventei Goertzel */
       }
   else if (2 == switch_value)                  /* Valoare intrerupatoare = 2.
                                                     Filtru Goertzel la 770 Hz.
                                     Al doilea rand al tastaturii telefonului */
      {left_output = goertzel_value( mono_input,COEFFICIENT_770_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
       right_output = left_output;
      }
   else if (3 == switch_value)                  /* Valoare intrerupatoare = 3.
                                                     Filtru Goertzel la 852 Hz.
                                   Al treilea rand al tastaturii telefonului */
      {left_output = goertzel_value( mono_input, COEFFICIENT_852_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
       right_output = left_output;
      }
  else if (4 == switch_value)                   /* Valoare intrerupatoare = 4.
                                                     Filtru Goertzel la 941 Hz.
                                  Al patrulea rand al tastaturii telefonului */
     {left_output = goertzel_value( mono_input, COEFFICIENT_941_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
      right_output = left_output;
     }
 else if (5 == switch_value)                    /* Valoare intrerupatoare = 5.
                                                     Filtru Goertzel la 1209 Hz.
                                       Prima coloana a tastaturii telefonului */
     {left_output = goertzel_value( mono_input, COEFFICIENT_1209_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
      right_output = left_output;
     }
 else if (6 == switch_value)                    /* Valoare intrerupatoare = 6.
                                                     Filtru Goertzel la 1336 Hz.
                                        A doua coloana a tastaturii telefonului
     {left_output = goertzel_value( mono_input, COEFFICIENT_1336_Hz);
                             //Calculeaza valoare Goertzel la aceasta frecventa
      right_output = left_output;
     }
else if (7 == switch_value)                     /* Valoare intrerupatoare = 7.
                                                     Filtru Goertzel la 1477 Hz.
```

```
                                    A treia coloana a tastaturii telefonului*/
     {left_output = goertzel_value( mono_input, COEFFICIENT_1477_Hz);
                              //Calculeaza valoare Goertzel la aceasta frecventa
      right_output = left_output;
     }
else if (8 == switch_value)                    /* Valoare intrerupatoare = 8.
                                                    Filtru Goertzel la 1633 Hz.
                         A patra coloana a tastaturii telefonului (nefolosit)*/
     {left_output = goertzel_value( mono_input, COEFFICIENT_1633_Hz);
                              //Calculeaza valoare Goertzel la aceasta frecventa
      right_output = left_output;
     }
else if (9 == switch_value)                    /* Valoare intrerupatoare = 9.
                                  Filtre Goertzel de la 697 Hz la 1633 Hz.
                                                          Butoanele 1 la #
     {             /* Filtru Goertzel pentru fiecare din frecvente */
      goertzel_filter(&delay_697_Hz[0], mono_input, COEFFICIENT_697_Hz);
      goertzel_filter(&delay_770_Hz[0], mono_input, COEFFICIENT_770_Hz);
      goertzel_filter(&delay_852_Hz[0], mono_input, COEFFICIENT_852_Hz);
      goertzel_filter(&delay_941_Hz[0], mono_input, COEFFICIENT_941_Hz);
      goertzel_filter(&delay_1209_Hz[0], mono_input, COEFFICIENT_1209_Hz);
      goertzel_filter(&delay_1336_Hz[0], mono_input, COEFFICIENT_1336_Hz);
      goertzel_filter(&delay_1477_Hz[0], mono_input, COEFFICIENT_1477_Hz);
      goertzel_filter(&delay_1633_Hz[0], mono_input, COEFFICIENT_1633_Hz);

      N++;                               //incrementeaza N

      if (199 == N)                  //primul rand al tastaturii telefonului?
        {goertzel_output_697_Hz = calculate_goertzel_output(&delay_697_Hz[0],
COEFFICIENT_697_Hz);
                  //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
         if (goertzel_output_697_Hz > GOERTZEL_THRESHOLD)
                                       /* daca rezultatul este mai mare decat
                               limita specificata seteaza bitul 0 in bit_mask */
            {goertzel_bit_mask |= 0x0001; }     // Seteaza bitul 0 in bit mask
        else {goertzel_bit_mask &= 0xFFFE;}
                                              // Reseteaza bitul 0 in bit mask

       }
     else if ( 200 == N)           //Al doilea rand al tastaturii telefonului?
           {goertzel_output_770_Hz = calculate_goertzel_output(
&delay_770_Hz[0], COEFFICIENT_770_Hz);
                  //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
          if ( goertzel_output_770_Hz > GOERTZEL_THRESHOLD)
             {goertzel_bit_mask |= 0x0002; }   // Seteaza bitul 1 in bit mask
          else {goertzel_bit_mask &= 0xFFFD; }
                                              // Reseteaza bitul 1 in bit mask
         }
   else if ( 201 == N)            //Al treilea rand al tastaturii telefonului?
          {goertzel_output_852_Hz = calculate_goertzel_output(
&delay_852_Hz[0], COEFFICIENT_852_Hz);
                  //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
          if ( goertzel_output_852_Hz > GOERTZEL_THRESHOLD)
             {goertzel_bit_mask |= 0x0004; } // Seteaza bitul 2 in bit mask
          else {goertzel_bit_mask &= 0xFFFB; }
```

```
                                                // Reseteaza bitul 2 in bit mask

         }
  else if ( 202 == N)            //Al patrulea rand al tastaturii telefonului?
        {goertzel_output_941_Hz = calculate_goertzel_output( &delay_941_Hz[0],
COEFFICIENT_941_Hz);
                 //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
         if ( goertzel_output_941_Hz > GOERTZEL_THRESHOLD)
            {goertzel_bit_mask |= 0x0008; }     // Seteaza bitul 3 in bit mask
         else {goertzel_bit_mask &= 0xFFF7; }
                                                // Reseteaza bitul 3 in bit mask

        }
 else if ( 203 == N)                  //Prima coloana a tastaturii telefonului
       {goertzel_output_1209_Hz = calculate_goertzel_output( delay_1209_Hz[0],
COEFFICIENT_1209_Hz);
                 //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
        if ( goertzel_output_1209_Hz > GOERTZEL_THRESHOLD)
           {goertzel_bit_mask |= 0x0010; }     // Seteaza bitul 4 in bit mask
        else {goertzel_bit_mask &= 0xFFEF; }
                                                // Reseteaza bitul 4 in bit mask

       }
else if ( 204 == N)                   //A doua coloana a tastaturii telefonului
      {goertzel_output_1336_Hz = calculate_goertzel_output( &delay_1336_Hz[0],
COEFFICIENT_1336_Hz);
                 //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
       if ( goertzel_output_1336_Hz > GOERTZEL_THRESHOLD)
          {goertzel_bit_mask |= 0x0020; }       // Seteaza bitul 5 in bit mask
       else {goertzel_bit_mask &= 0xFFDF; }
                                                // Reseteaza bitul 5 in bit mask
                                  }
else if ( 205 == N)                  //A treia coloana a tastaturii telefonului
      {goertzel_output_1477_Hz = calculate_goertzel_output( &delay_1477_Hz[0],
COEFFICIENT_1477_Hz);
                 //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
       if ( goertzel_output_1477_Hz > GOERTZEL_THRESHOLD)
          {goertzel_bit_mask |= 0x0040; }       // Seteaza bitul 6 in bit mask
       else {goertzel_bit_mask &= 0xFFBF; }
                                                // Reseteaza bitul 6 in bit mask
      }
else if (206 == N)                   //A patra coloana a tastaturii telefonului
     {goertzel_output_1633_Hz = calculate_goertzel_output( &delay_1633_Hz[0],
COEFFICIENT_1633_Hz);
                 //Calculeaza puterea de iesire Goertzel pt aceasta frecventa.
      if ( goertzel_output_1633_Hz > GOERTZEL_THRESHOLD)
         {goertzel_bit_mask |= 0x0080; } // Seteaza bitul 7 in bit mask
      else {goertzel_bit_mask &= 0xFF7F; } // Reseteaza bitul  7 in bit mask
     }
 else if ( 207 == N)            //Afisaza butonul apasat si reseteaza bit_mask
       {if ( BUTTON_0 == goertzel_bit_mask)
                       //compara bit_mask cu "butoanele declarate in goertzel.h
           {puts("Button 0 pressed.\n"); }
        else if ( BUTTON_1 == goertzel_bit_mask)
           {puts ("Button 1 pressed.\n"); }
        else if ( BUTTON_2 == goertzel_bit_mask)
           {puts ("Button 2 pressed.\n"); }
        else if ( BUTTON_3 == goertzel_bit_mask)
```

```
           {puts("Button 3 pressed.\n");  }
         else if ( BUTTON_4 == goertzel_bit_mask)
           {puts ("Button 4 pressed.\n"); }
         else if ( BUTTON_5 == goertzel_bit_mask)
           {puts ("Button 5 pressed.\n"); }
         else if ( BUTTON_6 == goertzel_bit_mask)
           {puts("Button 6 pressed.\n");  }
         else if ( BUTTON_7 == goertzel_bit_mask)
           {puts ("Button 7 pressed.\n"); }
         else if ( BUTTON_8 == goertzel_bit_mask)
           {puts ("Button 8 pressed.\n"); }
         else if ( BUTTON_9 == goertzel_bit_mask)
            {puts("Button 9 pressed.\n");  }
         else if ( BUTTON_STAR == goertzel_bit_mask)
            {puts ("Button * pressed.\n"); }
         else if ( BUTTON_HASH == goertzel_bit_mask)
            {puts ("Button # pressed.\n"); }
              /* Incepe din nou */
         N = 0;
         goertzel_bit_mask = 0;
       }
                                /* Aduna puterile de iesire a fiecarui filtru */
goertzel_power = goertzel_output_697_Hz + goertzel_output_770_Hz +
goertzel_output_852_Hz + goertzel_output_941_Hz +
goertzel_output_1209_Hz + goertzel_output_1336_Hz +
goertzel_output_1477_Hz + goertzel_output_1633_Hz ;
                                                        /* Calculeaza iesirea */

left_output = (short int) ( (long)(mono_input * goertzel_power) >> (15 - 2)) ;
right_output = left_output;
}
}
if ( counter < 5 )                //numai unul din 6 esantioane este procesat,
                                            celelelte esantioane sunt ignorate
   { counter++; }
else
   { counter = 0;}

      /* Afisaza iesirea pe display-ul cu LED-uri */
   bargraph_6dB ( left_output, right_output);
}
    /* Termina precesarea. Inchide codecul*/
    DSK5416_PCM3002_closeCodec(hCodec);

    puts("TMS320C5416 DSK has terminated.\n");
}

/****************************************************************************
/* main()
/****************************************************************************

void main()
{   /* Initializeaza librariile de suport ale placii    */
    DSK5416_init();
    /* Toate celelalte functii sunt apelate de DSP/BIOS */
}
/****************************************************************************/
```

ANEXA 2

```
/*************************************************************************
/
/*     alien_voices.c                                                    */
/*   TMS320C5416 DSK foloseste o rata de esantionare 48000 Hz (48 kHz).
*/
/*   Preia semnalul de intrare de la un microfon sau de la un CD-player, il
multiplica cu un semnal sinusoidal si genereaza suma sau diferenta de
frecvente, iar in final trimite iesirea spre difuzor.
/*************************************************************************
/
#include <stdio.h>    /* Required for functions printf() and puts()
*/
#include "alien_voicescfg.h"

#include "dsk5416.h"
#include "dsk5416_pcm3002.h"

#include "bargraph.h"
#include "IIR_high_pass_filters.h"
#include "IIR_low_pass_filters.h"
#include "sinewaves.h"
#include "switches.h"

/*************************************************************************
/
/* Configurarea registrilor codecului PCM3002                           */
/*************************************************************************
/

DSK5416_PCM3002_Config setup = {
    0x1FF,      // programare Registru 0 – atenuarea pe canalul stang CDA
    0x1FF,      // programare Registru 1 – atenuarea pe canalul drept CDA
    0x0,        // programare Registru 2 – diferite moduri de control
    0x0         // programare Registru 3 – pt controlul codecului
};

/*************************************************************************
/
/* functia ring_modulation() -  produce modulatie in inel prin multiplicarea
intrarii cu un semnal sinusoidal. Parametrii : intrarea audio si frecventa de
modulatie */
/*************************************************************************
/
static short int ring_modulation ( short int input1, short int input2)
{ signed long result;
  result = (long) ( input1 * input2) >> 15;
  return ( (short int) result);
}

/*********************************************************************/
/* Pentru compatibilitate cu functiile read / write ale pcm3002  */
/* aceste variabile trebuie declarate Int16 sau short int si nu int */
/*********************************************************************/
Int16 left_input;
Int16 left_output;
Int16 right_input;
Int16 right_output;
Int16 mono_input;
```

```
/************************************************************************
Aceasta procedura nu este apelata de main, ci este apelata de DSP/BIOS
************************************************************************/

void UserTask()
{   DSK5416_PCM3002_CodecHandle hCodec;
    unsigned long i;
    unsigned int switch_value;
    short int sinewave;
    short int temp;

    /* initializeaza codecul */
    hCodec = DSK5416_PCM3002_openCodec(0, &setup);

   /* Afisaza detalile la Stdout */
    puts("TMS320C5416 DSK: Generating alien voices using ring modulation.
Bargraph in 6dB intervals.\n");

    for ( i = 0 ; i < 12000000 ; i++ )
    {                                          // Citeste canalul de intrare
stanga
      while (!DSK5416_PCM3002_read16(hCodec, &left_input));
                                              // Reda la canalul de iesire stanga
      while (!DSK5416_PCM3002_write16(hCodec, left_output));

                                               /* Procesarea canalului stang */
      if ( 0 == switch_value)
                               /* intrarea este redata la iesire plus un
procent */
         { left_output = left_input + left_input/8;  }

      else if ( 1 == switch_value)
        {         /* Genereaza semnal sinusoidal modulator, cu frecventa de 2
Hz si amplitudine 32767 */
         sinewave = generate_sinewave_1 ( 2, 32767);
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 2 == switch_value)
        {         /* genereaza semnal sinusoidal de 20 Hz si amplitudine 32767
*/
         sinewave =  generate_sinewave_1 ( 20, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 3 == switch_value)
        {         /* genereaza semnal sinusoidal de 100 Hz si amplitudine
32767 */
         sinewave =  generate_sinewave_1 ( 100, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 4 == switch_value)
        {   /* genereaza semnal sinusoidal de 200 Hz si amplitudine 32767 */
         sinewave =  generate_sinewave_1 ( 200, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
```

```
      else if ( 5 == switch_value)
        {     /* genereaza semnal sinusoidal de 500 Hz si amplitudine 32767 */
         sinewave =  generate_sinewave_1 ( 500, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 6 == switch_value)
        {    /* genereaza semnal sinusoidal de 10000 Hz si amplitudine 32767*/
         sinewave =  generate_sinewave_1 ( 1000, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 7 == switch_value)
        {    /* genereaza semnal sinusoidal de 2000 Hz si amplitudine 32767 */
         sinewave =  generate_sinewave_1 ( 2000, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }
      else if ( 8 == switch_value)
        {     /* genereaza semnal sinusoidal de 600 Hz si amplitudine 32767 si
apoi se filtreaza */
         sinewave =  generate_sinewave_1 ( 600, 32767 );
         temp = ring_modulation ( left_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_low_pass_600Hz[0], temp);
        }
      else if ( 9 == switch_value)
        {         /* genereaza semnal sinusoidal de 1000 Hz si amplitudine
32767 si apoi se filtreaza */
         sinewave =  generate_sinewave_1 ( 1000, 32767 );
         temp = ring_modulation ( left_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_low_pass_1000Hz[0], temp);
        }
      else if ( 10 == switch_value)
        {
         /* genereaza semnal sinusoidal de 2000 Hz si amplitudine 32767 si
apoi se filtreaza */
         sinewave =  generate_sinewave_1 ( 2000, 32767 );
         temp = ring_modulation ( left_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_low_pass_2000Hz[0], temp);
        }
      else if  ( 11 == switch_value)
        {         /* se moduleaza de 2 ori: prima data trece sus la 2000 Hz,
si a doua oara trece jos la 2400 Hz. */
         sinewave =  generate_sinewave_1 ( 2000, 32767 );
         temp = ring_modulation ( mono_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_high_pass_2000Hz[0], temp);
         sinewave = generate_sinewave_2 ( 2400, 32767);
         temp = ring_modulation ( temp, sinewave);
         left_output = fourth_order_IIR_direct_form_II (
&IIR_low_pass_2400Hz[0], temp);
         left_output += (mono_input/8);
        }
      else if  ( 12 == switch_value)
```

```
        {         /* se moduleaza de 2 ori: prima data trece jos la 2000 Hz,
si a doua oara trece sus la 2400 Hz. */
         sinewave =  generate_sinewave_1 ( 2000, 32767 );
         temp = ring_modulation ( mono_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_low_pass_2000Hz[0], temp);
         sinewave = generate_sinewave_2 ( 2400, 32767);
         temp = ring_modulation ( temp, sinewave);
         left_output = fourth_order_IIR_direct_form_II (
&IIR_high_pass_2400Hz[0], temp);
         left_output += (mono_input/8);
        }
      else if  ( 13 == switch_value)
        {         /* se moduleaza de 2 ori: prima data trece sus la 2400 Hz,
si a doua oara trece jos la 2000 Hz. */
         sinewave =  generate_sinewave_1 ( 2400, 32767 );
         temp = ring_modulation ( mono_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_high_pass_2400Hz[0], temp);
         sinewave = generate_sinewave_2 ( 2000, 32767);
         temp = ring_modulation ( temp, sinewave);
         left_output = fourth_order_IIR_direct_form_II (
&IIR_low_pass_2000Hz[0], temp);
         left_output += (mono_input/8);
        }
      else if  ( 14 == switch_value)
        {         /* se moduleaza de 2 ori: prima data trece jos la 2400 Hz,
si a doua oara trece sus la 2000 Hz. */
         sinewave =  generate_sinewave_1 ( 2400, 32767 );
         temp = ring_modulation ( mono_input, sinewave);
         left_output = fourth_order_IIR_direct_form_I (
&IIR_low_pass_2400Hz[0], temp);
         sinewave = generate_sinewave_2 ( 2000, 32767);
         temp = ring_modulation ( temp, sinewave);
         left_output = fourth_order_IIR_direct_form_II (
&IIR_high_pass_2000Hz[0], temp);
         left_output += (mono_input/8);
        }
      else if ( 15 == switch_value)
        {         /* genereaza semnal sinusoidal de 500 Hz si amplitudine
32767 si aduna reverberatie */
         sinewave =  generate_sinewave_1 (500, 32767 );
         left_output = ring_modulation ( left_input, sinewave);
         left_output += (left_input/8);
        }

                                          /* citeste canalul de intrare drept */
      while (!DSK5416_PCM3002_read16(hCodec, &right_input));

                                                    /* reda canalul drept */
      while (!DSK5416_PCM3002_write16(hCodec, right_output));

                          /* citeste starea placii DSK si afiseaza la Stdout */
      switch_value = switch_status_display();

           /* unde este necesar, genreaza intrare mono din intrarile stanga si
                                                                   dreapta*/
      mono_input = stereo_to_mono ( left_input, right_input);
```

```
      /* procesarea de semnal */

      if ( 0 == switch_value)
        { right_output = right_input + (right_input/8); }

      else if ( 1 == switch_value)
        {                  /* unda sinusoidala de 2 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 2 == switch_value)
        {                  /* unda sinusoidala de 20 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 3 == switch_value)
        {                /* unda sinusoidala de 100 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 4 == switch_value)
        {                /* unda sinusoidala de 200 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 5 == switch_value)
        {                /* unda sinusoidala de 500 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 6 == switch_value)
        {               /* unda sinusoidala de 1000 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 7 == switch_value)
        {               /* unda sinusoidala de 2000 Hz si amplitudine 32767 */
         right_output = ring_modulation ( right_input, sinewave);
         right_output += (right_input/8);
        }
      else if ( 8 == switch_value)
        {                /* unda sinusoidala de 600 Hz si amplitudine 32767 */
         temp = ring_modulation ( right_input, sinewave);
         right_output = fourth_order_IIR_direct_form_II (
&IIR_high_pass_600Hz[0], temp);
        }
      else if ( 9 == switch_value)
        {               /* unda sinusoidala de 1000 Hz si amplitudine 32767 */
         temp = ring_modulation ( right_input, sinewave);
         right_output = fourth_order_IIR_direct_form_II (
&IIR_high_pass_1000Hz[0], temp);
        }
      else if ( 10 == switch_value)
        {               /* unda sinusoidala de 2000 Hz si amplitudine 32767 */
         temp = ring_modulation ( right_input, sinewave);
         right_output = fourth_order_IIR_direct_form_II (
&IIR_high_pass_2000Hz[0], temp);
```

```
        }
      else if  ( 11 == switch_value)
        {
         /* modulatie trece sus la 2000 Hz si apoi modulatie trece jos la 2400
                                                                          Hz*/
         right_output = left_output;
         right_output += (mono_input/8);
        }
      else if  ( 12 == switch_value)
        {
   /* modulatie trece jos la 2000 Hz si apoi modulatie trece sus la 2400 Hz */
         right_output = left_output;
         right_output += (mono_input/8);
        }
      else if  ( 13 == switch_value)
        {
   /* modulatie trece sus la 2400 Hz si apoi modulatie trece jos la 2000 Hz */
         right_output = left_output;
         right_output += (mono_input/8);
        }
      else if  ( 14 == switch_value)
        {
   /* modulatie trece jos la 2400 Hz si apoi modulatie trece sus la 2000 Hz */
         right_output = left_output;
         right_output += (mono_input/8);
        }
      else if ( 15 == switch_value)
        {                /* unda sinusoidala de 200 Hz si amplitudine 32767,
                                              si apoi se aduna reverberatii */
         temp = ring_modulation ( right_input, sinewave);
         right_output = reverberation ( temp );
         right_output += (mono_input/8);
        }

      /* se afiseaza iesirea */
      bargraph_6dB ( left_output, right_output);
     }

    /* Termina precesarea. Inchide codecul */
    DSK5416_PCM3002_closeCodec(hCodec);

    puts("TMS320C5416 DSK has terminated\n");
}
/*************************************************************************
/
/* main()
*/
/*************************************************************************
/

void main()
{   /* Initialize the board support library                        */
    /* There is no need to initialize the DIP switches and the LEDs */

    DSK5416_init();

    /* All other functions are scheduled by DSP/BIOS */
}

/************************************************
```