

## MODUL 6

## 4-Preset Equalizer menggunakan filter IIR

---

### 1. Pendahuluan

Ketika anda menjalankan aplikasi pemutar lagu seperti WinAmp, anda akan menemukan tombol untuk *equalizer*. Bila anda klik maka akan ditampilkan *slider-slider* untuk mengatur respon frekuensi. Anda ingin suara bass terdengar lebih berdebam? Geser saja slider untuk frekuensi bass ke atas. Anda ingin suara treble terdengar lebih bergemerincing? Geser saja slider untuk frekuensi treble ke atas. Anda juga bisa memilih preset yang sudah disediakan, tinggal pilih saja, ingin Full Bass, Full Treble, Rock, Pop, Jazz, Classic, dan sebagainya. Pada praktikum ini anda akan mencoba membuat 4 buah preset didalam DSK TMS320C5402 menggunakan bahasa-C. Empat preset tersebut adalah Normal (tanpa filter), Bass, Bass-Voice, dan Voice- Treble.

### 2. Tujuan

Setelah menyelesaikan praktikum ini, yang anda peroleh adalah :

- dapat menjelaskan cara mendisain filter IIR
- dapat menjelaskan cara mendapatkan nilai koefesien filter IIR menggunakan Matlab
- dapat mengimplementasikan disain filter IIR pada DSK TMS320C5402 sebagai 4 preset equalizer menggunakan bahasa-C
- dapat memanfaatkan dua user-switch pada DSK TMS320C5402 sebagai input untuk memilih jenis preset.

### 3. Gambaran Disain

DSK TMS320C5402 dilengkapi dengan sebuah codec. Mainkan sebuah lagu mp3 pada komputer, suara yang dihasilkan dimasukkan pada ADC, lalu diproses didalam DSP, hasil proses dikeluarkan melalui DAC untuk anda dengarkan kembali melalui speaker. Didalam DSP, suara akan melewati proses filter. Pada percobaan sebelumnya anda telah mengetahui bahwa pada filter digital, respon frekuensi bergantung pada koefesien filter. Bila nilai koefesien filter berubah maka respon frekuensinya juga berubah. Program filter yang akan anda buat hanya satu, tetapi koefesien yang disediakan ada empat macam masing-masing untuk empat respon frekuensi yang berbeda. Koefesien mana yang dipakai akan dipilih menggunakan dua user-switch pada board DSK. Empat kalimat terakhir tadi harus anda pahami, bila tidak bisa gawat nantinya. Mari kita coba membuat equalizer tersebut.

## 4. Dasar Teori

Filter IIR adalah salah satu tipe dari filter digital yang dipakai pada aplikasi Digital Signal Processing (DSP). IIR kepanjangan dari Infinite Impulse Response. Mengapa disebut respons impulsnya tak terbatas (infinite)? Karena *adanya feedback* didalam filter, jika anda memasukkan sebuah impulse (yaitu sebuah sinyal '1' diikuti dengan banyak sinyal '0'), maka pada outputnya akan terus menerus beresilasi karena adanya umpan balik, walaupun pada prakteknya akan hilang pada suatu saat.

Keuntungan filter IIR antara lain adalah membutuhkan koefisien yang lebih sedikit untuk respon frekuensi yang curam sehingga dapat mengurangi jumlah waktu komputasi.

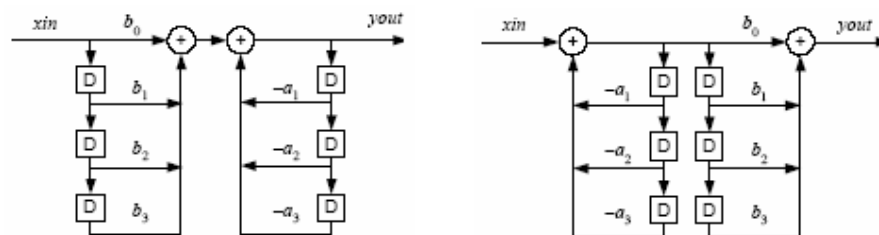
$$x[n] \longrightarrow \boxed{h[n]} \longrightarrow y[n] \quad y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m]$$

Fungsi transfer filter IIR adalah:

$$H(z) = \frac{b_0 + b_1z^{-1} + \dots + b_Nz^{-N}}{1 + a_1z^{-1} + \dots + a_Nz^{-N}}$$

$$y[n] = \sum_{m=0}^q b_m x[n-m] - \sum_{m=0}^p a_m y[n-m]$$

Flow graph dari filter IIR sebagai berikut :

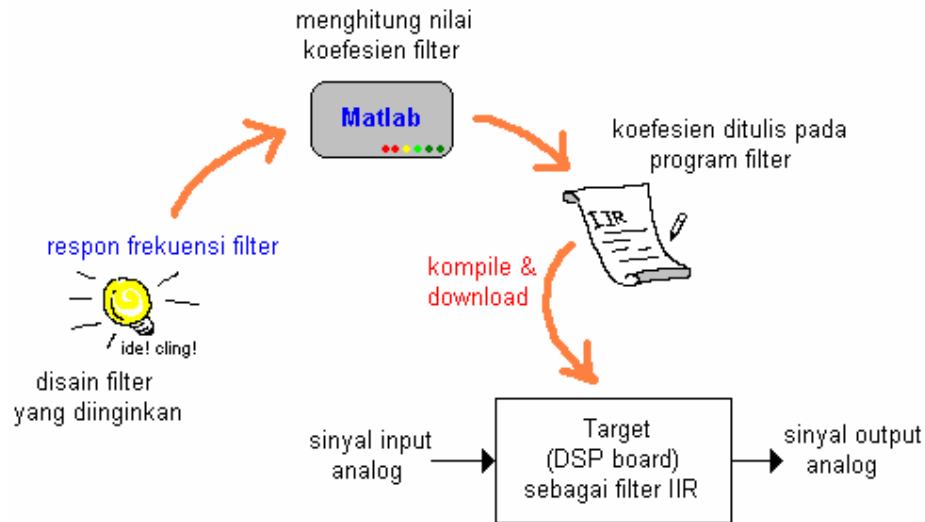


Gambar 1. Flowgraph filter IIR Direct Form I (kiri) dan Direct Form II (kanan)

### Bagaimana mengimplementasikan filter IIR pada DSP TMS320C5402 ?

Secara singkat, tahapan-tahapan untuk membuat filter digital IIR pada praktikum ini sama dengan percobaan sebelumnya, yaitu:

1. Menentukan respon frekuensi filter yang diinginkan
2. Menghitung nilai koefisien filter dengan Matlab
3. Menuliskan koefisien filter kedalam program filter
4. Kompilasi program dan download kode mesin ke DSP
5. Menguji sistem dengan memberikan sinyal input dari audio-out komputer dan mendengarkan hasilnya melalui speaker



Gambar 2. Ilustrasi alur implementasi filter IIR

## Menghitung koefisien filter IIR menggunakan fungsi **BUTTER** pada Matlab

Pada praktikum ini akan digunakan fungsi *butter* yang disediakan oleh Matlab untuk mendapatkan koefisien filter dengan metode butterworth.

Perintah  $[b,a] = \text{BUTTER}(N,W_n)$ ; pada Matlab digunakan untuk mendisain filter IIR low-pass menggunakan metode butterworth **orde N** dengan **frekuensi cut-off pada  $W_n$** , dan menghasilkan koefisien filter pada vektor B (numerator) dan vektor A (denominator) sebanyak  $N+1$ . Nilai dari frekuensi cut-off  $W_n$  haruslah bernilai antara  $0.0 < W_n < 1.0$ , dimana *1.0 menunjukkan setengah dari frekuensi sampling*.

$[b,a] = \text{BUTTER}(N,W_n, 'high')$ ;  
digunakan untuk mendisain filter high-pass

$[b,a] = \text{BUTTER}(N,W_n)$ ;  
dengan  $W_n=[W_1 W_2]$  digunakan untuk mendisain filter band-pass

$[b,a] = \text{BUTTER}(N,W_n, 'stop')$ ;  
dengan  $W_n=[W_1 W_2]$  digunakan untuk mendisain filter band-stop

Lebih detail tentang *syntax* BUTTER anda dapat memanfaatkan fungsi *help* pada Matlab.

## Langkah-langkah mendisain filter IIR low-pass menggunakan BUTTER

1. Pilih orde filter, misal  $N=10$
2. Menentukan frekuensi cut-off
3. Koefisien filter dapat dihitung menggunakan perintah BUTTER pada Matlab.

4. Simpan nilai koefisien pada file, yang nantinya digunakan untuk mengimplementasikan filter IIR dengan konvolusi pada pemrograman DSP.

### Program Matlab untuk menghasilkan koefisien filter menggunakan BUTTER

Berikut contoh disain program Matlab untuk menghasilkan koefisien filter IIR low-pass dengan frekuensi cut-off 2KHz pada frekuensi sampling sebesar 16KHz. Nilai koefisien yang dihasilkan disimpan dalam file bertipe teks.

Pertama, tentukan berapa frekuensi sampling yang akan digunakan

Frekuensi sampling filter = 16KHz

Setengah frekuensi sampling filter = 8KHz

Kedua, tentukan berapa frekuensi cut-off filter

Frekuensi cut-off filter = 2KHz

nilai  $\omega_n$  sebagai cut-off bernilai

$$\frac{\text{frek.cuoff}}{\frac{1}{2}\text{frek.sampling}} = \frac{2\text{KHz}}{8\text{KHz}} = 0.25$$

Sehingga program Matlab untuk mendapatkan koefisien filter low-pass IIR orde 4 adalah:

```
[b,a] = butter(4, 0.25);           % [b,a] = butter(n,wn);
```

Program lengkap dalam Matlab untuk mendapatkan koefisien filter low-pass IIR dengan frekuensi cut-off 2KHz pada frekuensi sampling 16KHz sebagai berikut:

```
f=8;           % ½ frekuensi sampling
n=3;           % orde filter
w1=0.25;       % frek. cut-off, dinormalisasi terhadap ½ fs
[b,a]=butter(n,w1); % hitung koefisien IIR

% tulis kedalam file iir.txt
fid=fopen('iir.txt','w');
fprintf(fid,'Koefisien filter IIR orde %d \n',n);
fprintf(fid, strcat('Generated on-',datestr(now),'\n\n'));
fprintf(fid,'Nilai koefisien b\n');
for i=1:n+1
    fprintf(fid,'%1.6f \n', b(i));
end
fprintf(fid,'Nilai koefisien a\n');
for i=1:n+1
    fprintf(fid,'%1.6f \n', a(i));
end
fclose(fid);

[h,w]=freqz(b,a,128);
```

```

%plot dalam desibel
%plot((w*f)/3.14, 20*log(abs(h)));
plot((w*f)/3.14, (abs(h)));

xlabel ('f (KHz)');ylabel ('|H|');
title(strcat('Respon Frekuensi Filter IIR orde-',int2str(n)));grid;

```

Anda dapat melihat nilai koefisien yang telah disimpan dalam file iir.txt menggunakan windows explorer atau pada Matlab command window menggunakan perintah type iir.txt.

```

MATLAB Command Window
File Edit View Window Help
>> type iir.txt

Koeffisien Filter IIR orde 3
Generated on-23-Nov-2005 07:38:47

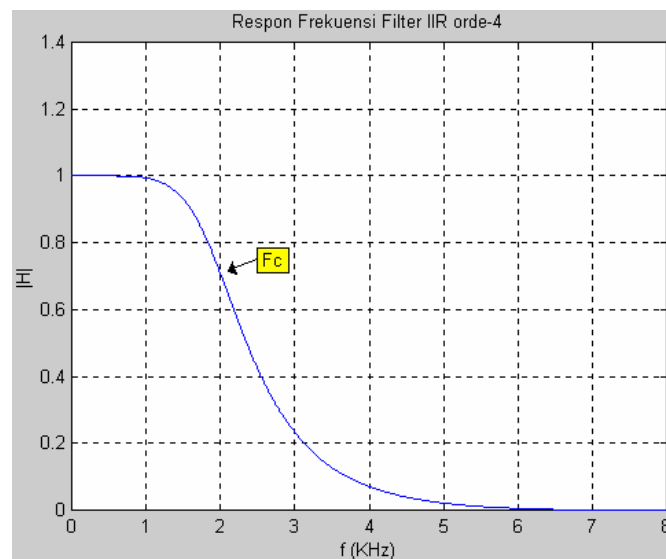
Nilai koefisien b
0.031689
0.095068
0.095068
0.031689
Nilai koefisien a
1.000000
-1.459029
0.910369
-0.197825

>> |

```

Gambar 3. Melihat isi dari file iir.txt yang telah dibuat

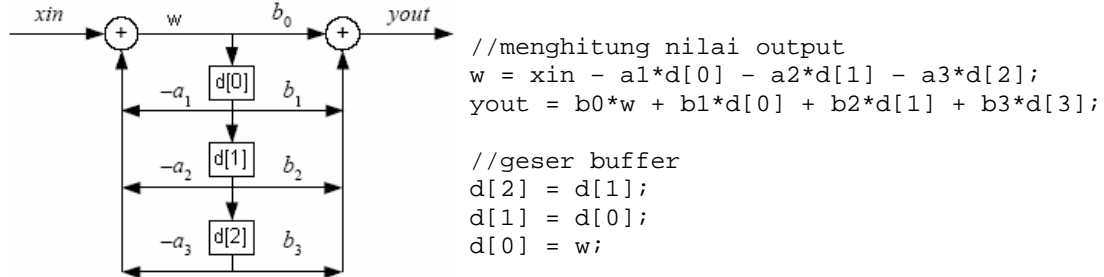
Respon frekuensi filter IIR orde 3 hasil disain tersebut ditunjukkan oleh gambar 4.



Gambar 4. Plot respon frekuensi filter low-pass IIR orde 3

## Membuat program filter IIR pada DSP

Seperti diagram flow Direct Form II yang ditunjukkan oleh gambar 1, maka langkah yang harus dilakukan ditunjukkan seperti pada gambar 5 berikut.



Gambar 5. Ilustrasi langkah program filter IIR orde 3

## Implementasi filter IIR pada DSP menggunakan bahasa-C

```

/*****
/* IIR.c
/* Modified from codec.c by Texas Instruments
/* Author: Hary Oktavianto. PENS-ITS. 30.11.04
/*
*****/

#include <type.h>
#include <board.h>
#include <codec.h>
#include <mcbsp54.h>

#define orde 3

int iir(int data);
int getDipSwStatus(void);
void delay(s16 period);

/*****
/* Global Variables
*****/

HANDLE hHandset;
int data;
int i,dipsw=0;
float w,output;
float d[orde+1]={0};

/*****
/* Koefesien filter
*****/
float b[orde+1]={0.166667, 0.500000, 0.500000, 0.166667};
float a[orde+1]={1.000000, -0.000000, 0.333333, -0.000000};

/*****
/* MAIN PROGRAM
*****/

void main()
{
    s16 cnt=2;

    if (brd_init(100))
        return;

```

```

    /* blink the leds a couple times */
    while ( cnt-- )
    {
        brd_led_toggle(BRD_LED0);
        delay(1000);
        brd_led_toggle(BRD_LED1);
        delay(1000);
        brd_led_toggle(BRD_LED2);
        delay(1000);
    }

    /* Open Handset Codec */
    hHandset = codec_open(HANDSET_CODEC);      /* Acquire handle to codec */

    /* Set codec parameters */

    codec_dac_mode(hHandset, CODEC_DAC_15BIT);      /* DAC in 15-bit mode */
    codec_adc_mode(hHandset, CODEC_ADC_15BIT);      /* ADC in 15-bit mode */
    codec_ain_gain(hHandset, CODEC_AIN_0dB);        /* 0dB gain on analog input to ADC */
    codec_aout_gain(hHandset, CODEC_AOUT_MINUS_0dB); /* 0dB gain on analog output from DAC */
    codec_sample_rate(hHandset, SR_16000);         /* 16KHz sampling rate */

    /* Polling and digital loopback */
    while (1)
    {
        /* Wait for sample from handset */
        while (!MCBSP_RRDY(HANDSET_CODEC)) {};

        /* Read sample from and write back to handset codec */
        data = *(volatile u16*) DRR1_ADDR(HANDSET_CODEC);
        *(volatile u16*)DXR1_ADDR(HANDSET_CODEC) = iir(data);
    }
}

int iir(int data)
{
    /*mengambil posisi user-switch
    dipsw = getDipSwStatus();

    /*menghitung nilai output
    w = data - a[1]*d[0] - a[2]*d[1] - a[3]*d[2];
    output = b[0]*w + b[1]*d[0] + b[2]*d[1] + b[3]*d[2];

    /*geser buffer
    d[2] = d[1];
    d[1] = d[0];
    d[0] = w;

    return( (int) output );
}

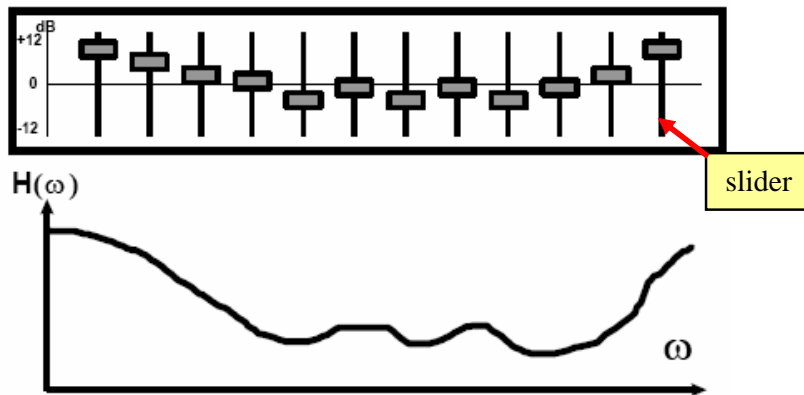
int getDipSwStatus(void)
{
    /* DIP SWITCH I/O Port
    static ioport unsigned int port0001;
    /* Dip Switch 7,8 Status
    return( (port0001 & 0x60) >> 5);
}

void delay(s16 period)
{
    int i, j;
    for(i=0; i<period; i++) {for(j=0; j<period>>1; j++);}
}

```

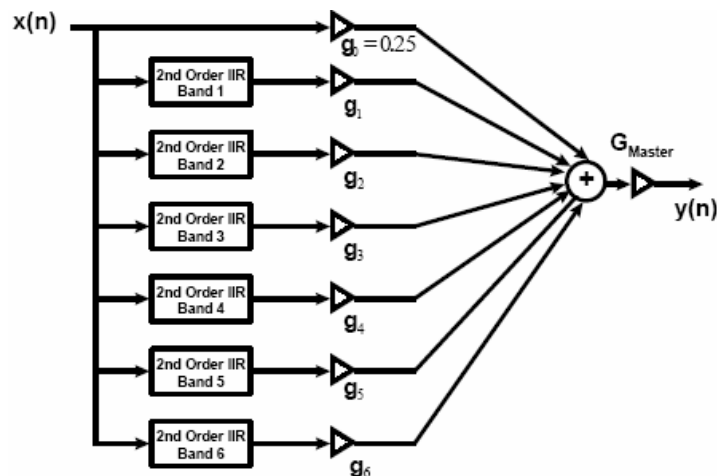
## EQUALIZER

Equalizer digunakan untuk mengatur amplitudo sinyal pada daerah frekuensi yang diinginkan. Pada *graphic equalizer*, spektrum frekuensi dibagi menjadi beberapa daerah (*band*) frekuensi menggunakan filter band-pass. Kita dapat mengatur *slider* untuk penguatan (gain) yang berbeda memberikan kita “gambaran visual” (gambar 6) untuk semua respon frekuensi dari peralatan equalizer.



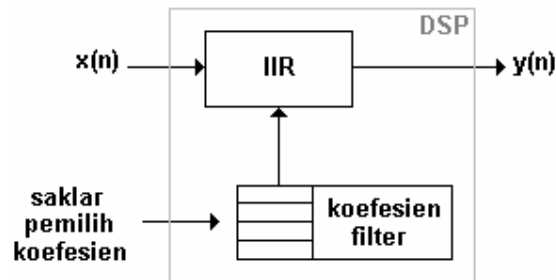
Gambar 6. Equalizer graphic analog 12 band yang umum dijumpai

Equalizer analog umumnya menggunakan komponen pasif (resistor, kapasitor, dan induktor) dan aktif (transistor atau op-amp). Menambahkan jumlah band menghasilkan ukuran PCB (Printed Circuit Board) yang semakin besar. Bila sistem yang sama diimplementasikan pada DSP, bagaimanapun juga, jumlah band hanya dibatasi oleh kecepatan DSP (MIPS) walaupun ukuran board tidak berubah. Tetapi hal ini dapat diatasi bila sistem diimplementasikan menggunakan FPGA (Field Programmable Gate Array). Resistor dan kapasitor digantikan oleh koefisien filter waktu-diskrit, yang disimpan dalam memori dan dapat dimodifikasi dengan mudah.



Gambar 7. Equalizer digital 6 band yang diimplementasikan pada DSP

Gambar 7 menunjukkan contoh equalizer yang diimplementasikan pada DSP. Dan rancangan disain sistem equalizer yang akan anda coba pada praktikum ini ditunjukkan oleh gambar 8.



Gambar 8. Rancangan sistem equalizer yang akan anda coba

## Referensi

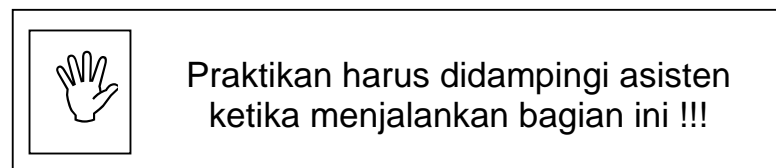
- Dan Ledger and John Tomarakos, *Using The Low Cost, High Performance ADSP-21065L Digital Signal Processor For Digital Audio Applications*, DSP Applications Group, Analog Devices, Norwood, MA 02062, USA, Revision 1.0 - 12/4/98
- Hyeokho Choi, *IIR Filter*, Digital Signal Processing Laboratory ELEC434, Rice University, 2000

## 5. Peralatan

- 1 set PC yang dilengkapi dengan software Matlab dan Code Composer Studio v2
- 1 set DSK TMS320C5402
- 1 set function generator
- 1 set oscilloscope
- 1 set speaker
- sumber suara (PC) + kabel mono

## 6. Prosedur Praktikum

Praktikan diharapkan mengikuti langkah-langkah prosedur praktikum dan apabila ada kesulitan harap bertanya kepada asisten.



Tujuan:

Mendisain filter IIR **low-pass orde 3** dengan frekuensi **cut-off 3 KHz** pada frekuensi **sampling 16000 Hz**.


Langkah-langkah:

1. Persiapan peralatan :
  - a. PC dalam keadaan mati.
  - b. Hubungkan DSK ke PC menggunakan kabel paralel port yang tersedia.
  - c. Hubungkan output adaptor ke input power DSK.
  - d. Hubungkan kabel power adaptor, nyalakan adaptor.
  - e. Nyalakan PC
  - f. Jalankan aplikasi Code Composer Studio dan pastikan dapat terhubung dengan board DSK
2. Bukalah software Matlab, dan ketikkan program berikut pada matlab editor dan jalankan.

```
f=8;                % ½ frekuensi sampling = 8kHz
n=3;                % orde filter
wl=0.375;          % fc = 3kHz
[b,a]=butter(n,wl); % hitung koefesien IIR

% tulis kedalam file iir.txt
fid=fopen('iir.txt','w');
fprintf(fid,'Koefesien filter IIR orde %d \n',n);
fprintf(fid, strcat('Generated on-',datestr(now),'\n\n'));
fprintf(fid,'Nilai koefesien b\n');
for i=1:n+1
    fprintf(fid,'%1.6f \n', b(i));
end
fprintf(fid,'Nilai koefesien a\n');
for i=1:n+1
    fprintf(fid,'%1.6f \n', a(i));
end
fclose(fid);
[H,w]=freqz(b,a,128);           % respon frekuensi
%plot((w*f)/3.14, 20*log(abs(h))); %plot dalam desibel
plot((w*f)/3.14, (abs(H)));
xlabel('f (KHz)');ylabel('|H|');
title(strcat('Respon Frekuensi Filter IIR orde-',int2str(n)));grid;
```

3. Amati dan catat hasil plot respon frekuensi filter pada Matlab, anda akan membandingkan hasil yang dihitung oleh Matlab dengan hasil percobaan.
4. Pada Matlab command window ketikkan perintah `type iir.txt` untuk melihat isi file iir.txt.
5. Dengan menggunakan Windows Explorer, buatlah folder baru pada direktori **D:\prak\_pengolahansinyal** dengan kelas Anda diikuti dengan subfolder nama Anda. Perhatikan penulisan folder yang Anda buat. Kemudian salinlah direktori iir pada **C:\ti\examples\dsks402\dsp\iir** kedalam direktori **D:\prak\_pengolahansinyal\kelas\nama**. Hal ini dimaksudkan untuk mempermudah mengembalikan isi project seperti dalam keadaan semula apabila terjadi kesalahan.

6. Pada CCS, dengan menggunakan **Project → Open**, bukalah file project *iir.pjt* pada direktori **D:\prak\_pengolahansinyal\kelas\nama\iir**. Apabila file library pada project tersebut ada yang tidak ditemukan maka carilah library tersebut pada direktori c:\ti yang sesuai. Hal ini terjadi karena lokasi project berpindah tempat. File library yang digunakan ada tiga yaitu:
  - a. rts.lib pada direktori c:\ti\c5400\cgtools\lib
  - b. dsk5402.lib pada direktori c:\ti\c5400\dsk5402\lib
  - c. drv5402.lib pada direktori c:\ti\c5400\dsk5402\lib
7. Bukalah file **iir.c** didalam source pada jendela project view. Lakukan copy-paste nilai koefesien yang tersimpan pada file iir.txt.
8. Pilih **Project → Rebuild All** (atau dengan menekan ikon ). Maka CCS akan merekompilasi, mengassembler dan melakukan relink semua file pada project. Pesan pada proses ini akan ditampilkan pada bagian bawah window
9. Setelah kompilasi selesai dan tidak ada error, pilih **File → Load Program**, browse dan pilih file iir.out. Maka CCS akan meload program pada target DSP dan membuka window dis-assembly yang memperlihatkan instruksi program dalam bahasa assembler.
10. Pilih **Debug → Go Main**
11. Siapkan oscilloscope yang telah dikalibrasi. Atur volt/div pada 0.5 volt dan atur time/div pada 1ms atau disesuaikan agar sinyal terlihat dengan jelas.
12. Siapkan function generator untuk menghasilkan sinyal sinusoida dengan frekuensi 1KHz dan amplitudo 50mVpp.
13. Hubungkan output dari function generator pada input dari board DSK (jack audio untuk microphone, bersebelahan dengan konektor RJ-11 untuk line telephone) dan hubungkan output dari board DSK (jack audio untuk speaker, bersebelahan dengan konektor DB9 untuk serial port) pada input dari oscilloscope.
14. Pilih **Debug → Run**, kemudian dengan amplitudo input dari function generator tetap sebesar 50mVpp dengan frekuensi 1KHz, ubahlah frekuensinya semakin besar perlahan-lahan. Amati dan catat hasil pengamatan seperti pada tabel 1.
15. Pilih **Debug → Halt**, untuk menghentikan eksekusi pada board.
16. Kerjakan tugas.
17. Tutuplah program CCS dengan memilih **File → Exit**.
18. Matikan adaptor power suplai. Lakukan prosedur *shutdown* PC dengan benar. Rapikan kembali kabel dan peralatan.

Tabel 1. Hasil Pengukuran Filter IIR low-pass 3 KHz

No.	Vin = 50 mVpp	
	Frekuensi input (KHz)	Vout (mVpp)
1	1.0	
2	1.5	
3	2.0	
4	2.5	
5	3.0	
6	3.5	
7	4.0	
8	4.5	
9	5.0	
10	5.5	

## 7. Tugas

Disainlah sistem 4-preset equalizer seperti yang ditunjukkan oleh gambar 7. Sistem 4-preset equalizer tersebut mempunyai sebuah filter IIR dengan empat koefisien. Hanya satu koefisien yang digunakan pada satu waktu. Pemilihan koefisien dilakukan dengan cara mengubah posisi DipSwitch pada board DSK. **Saklar yang boleh anda ubah adalah saklar nomor 7 dan nomor 8. Jangan mengubah posisi saklar yang lain karena akan mengakibatkan board DSK tidak dapat bekerja.** Macam filter yang akan didisain ditunjukkan oleh tabel 2.

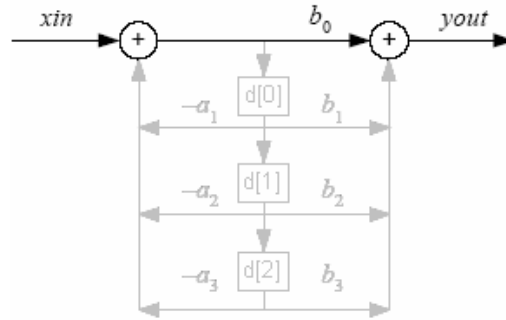
Tabel 2. Saklar 4-preset equalizer

Preset #	DipSW-7	DipSW-8	Filter
1 (Normal)	OFF	OFF	Tidak ada
2 (Bass)	OFF	ON	LPF 1 kHz
3 (Bass-Voice)	ON	OFF	LPF 3 kHz
4 (Voice-Treble)	ON	ON	HPF 1 kHz

\*) posisi ON = saklar kebawah = logika 0

Koefisien filter tabel 2 diperoleh menggunakan bantuan Matlab. Simpan hasilnya pada program iir.c dengan melakukan modifikasi program yang diperlukan. Gunakan subrutin `getDipSwStatus()` untuk mengambil nilai posisi DipSwitch, nilai yang dikembalikan adalah 0,1,2 dan 3.

Preset nomor 1 tidak menggunakan filter, artinya bila anda memilih preset ini maka sinyal input akan langsung menuju output. Perhatikan gambar 9, anda dapat menuliskan langsung koefisien  $a[ ]$  dan  $b[ ]$  sehingga flowgraph yang terjadi hanya melewati input menuju output.



Gambar 9. Berapakah nilai  $a[ ]$  dan  $b[ ]$  agar  $xin$  langsung menuju  $yout$  ?

Apabila anda telah selesai memodifikasi program, lakukan kompilasi program dan download kode mesin kedalam DSP. Matikan function generator dan oscilloscope karena tidak digunakan lagi.

Mainkan sebuah lagu mp3 pada komputer. Dengan kabel yang tersedia, hubungkan audio-out pada komputer ke jack input pada board DSK (jack untuk microphone, bersebelahan dengan konektor RJ-11 untuk line telephone). Kemudian hubungkan output dari board DSK ke speaker aktif.

Jalankan program pada DSP dengan memilih **Debug** → **Run**. Cobalah mengganti-ganti posisi dip-switch ke-7 dan ke-8. Apakah program yang anda buat telah berhasil?

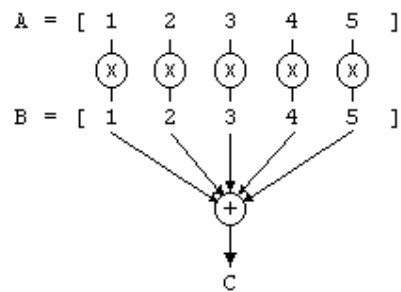
## 8. Analisa

1. Bandingkan plot respon frekuensi hasil perhitungan Matlab pada percobaan langkah ke-4 dengan hasil percobaan pada tabel 1.
2. Bandingkan plot respon frekuensi filter low-pass dengan frekuensi cut-off 3KHz jenis IIR yang anda coba pada praktikum ini dengan jenis FIR yang anda dapat pada percobaan sebelumnya. Jelaskan perbedaannya!
3. Tulislah program-C yang anda buat untuk mengimplementasikan sistem 4-preset equalizer tersebut (hanya program yang anda tambahkan/modifikasi saja).

## 9. Pertanyaan pendahuluan

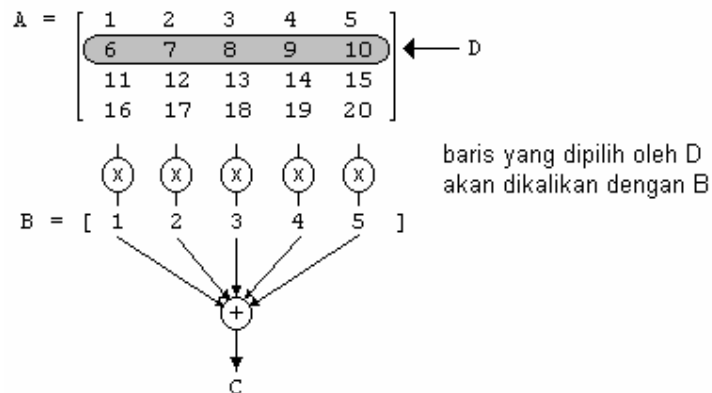
Buatlah program dalam bahasa-C untuk merealisasikan tugas berikut. Anda tidak perlu meminta *user* memasukkan nilai kedalam array, anggap saja nilai array sudah ditentukan seperti pada gambar.

1. Variabel A dan variabel B masing-masing mempunyai elemen berupa matriks ordo 1 x 5. Jumlah dari perkalian tiap-tiap elemen yang seletak disimpan pada variabel C.



Gambar 10. Ilustrasi pertanyaan pertama

2. Variabel A mempunyai elemen berupa matriks ordo  $4 \times 5$ . Variabel B mempunyai elemen berupa matriks ordo  $1 \times 5$ . Variabel D akan memilih salah satu baris dari variabel A untuk dikalikan dengan variabel B. Variabel C menyimpan jumlah dari perkalian tiap-tiap elemen yang seletak untuk deret tertentu pada variabel A yang telah dipilih dengan variabel B.



Gambar 11. Ilustrasi pertanyaan kedua

## 10. Tambahan

Berikan saran atau komentar guna pengembangan lebih lanjut praktikum ini.