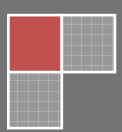
2010

Praktikum Pengolahan Sinyal Digital

(Matlab & TMS320C6713)

Tri Budi Santoso
Hary Octavianto
Miftahul Huda

Laboratorium Sinyal
EEPIS - ITS



KATA PENGANTAR

Alhamdulillah, dengan mengucap puji syukur ke hadirat Alloh SWT, karena atas rahmad dan hidayahNya *Buku Petunjuk Praktikum Pengolahan Sinyal Digital* ini selesai dibuat.

Buku ini disusun untuk memenuhi kebutuhan mahasiswa Politeknik Elektronika Negeri Surabaya dan bisa dipergunakan untuk semua Program Studi yang ada, baik untuk jenjang D3 maupun D4. Buku ini merupakan pengganti dari Modul Pengolahan Sinyal Digital sebelumnya yang telah disusun berbasis On-Board Simulator TMS320C5402. Dengan metode penyampaian yang sederhana diharapkan siswa dapat memanfaatkan modul ini dalam pembelajaran. Di dalam buku ini juga dilengkapi contoh-contoh yang lebih mengarah ke bentuk yang lebih aplikatif dengan memanfaatkan perangkat lunak Matlab dan sebuah On-Board Simulator TMS320C6713.

Dengan selesainya buku ini dengan tulus ikhlas dan kerendahan hati, kami mengucapkan terima kasih yang mendalam kepada:

1. Para Pimpinan Politeknik Elektronika Negeri Surabaya - Institut Teknologi Sepuluh Nopember yang telah memberikan segala fasilitas, dorongan semangat, dan suasana kerja yang memberikan semangat kami untuk menyelesaikan buku ini.
2. Rekan-rekan di Group *Signal Processing* kampus Politeknik Elektronika Negeri Surabaya yang selalu menyediakan waktunya untuk berdiskusi dalam proses pembuatan buku ini.
3. Keluarga di rumah yang dengan ikhlas meluangkan waktu bagi kami untuk menyelesaikan buku ini.

Mudah-mudahan sumbangan pemikiran yang kecil ini bisa memberikan kontribusi dalam mencerdaskan mahasiswa dilingkungan kampus PENS.

Surabaya, 22 Mei 2010

Penulis

DAFTAR ISI

Daftar Pustaka

Bio Data Penulis

MODUL 0

PENGENALAN TMS320C6713 DSK

I. TUJUAN

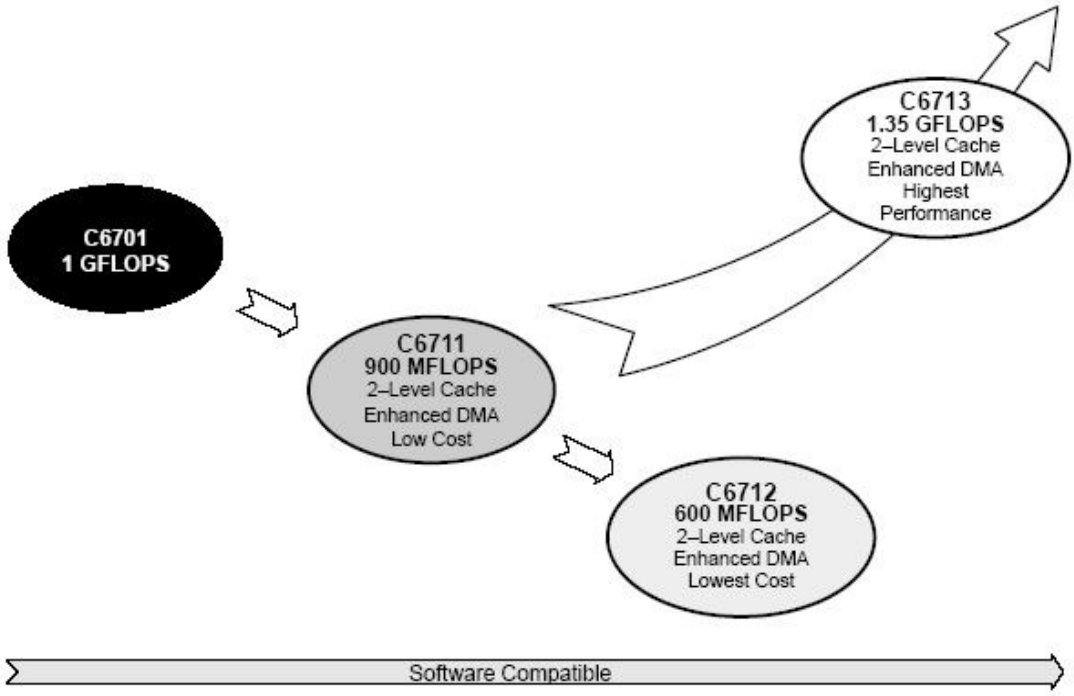
- Siswa memahami bentuk fisik TMS320C6713

- Siswa mampu mengoperasikan modul TMS320C6713

II. PENGANTAR

Salah satu produk dari Texas Instruments, yaitu platform TMS320C6000 DSP pada high-performance digital signal processors (DSPs) telah meluncurkan TMS320C6713. Tipe DSK C6713 mememiliki kinerja tertinggi diantara keluarga C6000 DSP platform pada tipe floating-point DSPs. Dengan dibekali clock rate sebesar 225 MHz, C6713 dapat memproses informasi pada rate 1.35 giga-floating-point operations per second (GFLOPS).

Produk ini diperkenalkan pada February 1997, C6000 DSP platform didasarkan pada arsitektur TI's VelociTI™, yang merupakan pengembangan dari very-long-instruction-word (VLIW) architecture untuk DSPs. Pengembangan fitur pada arsitektur VelociTI melibatkan instruction packing, conditional branching, dan pre-fetched branching, yang mana in imenimbulkan permasalahan berkaitan dengan implementasi implementasi VLIW yang muncul sebelumnya.



Gambar 1. Roadmap floating point DSP produk TI

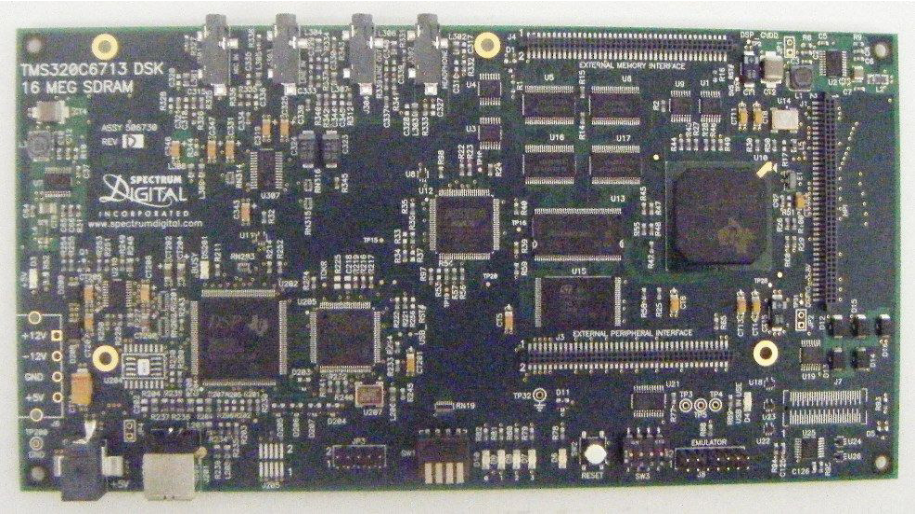
Architecture sifatnya highly deterministic, yang mana beberapa hambatan bagaimana ketika instruksi di-fetched, executed, atau stored. Flexibilitas arsitektur ini merupakan kunci untuk terobosan level efficiency pada compiler C6000.

Roadmap untuk platform DSP floating-point C6000 diberikan dalam Gambar 1, yang mana menunjukkan komitmen dari produk TI's untuk tampil sebagai highest-performance DSPs.

Dalam kesempatan ini kami mencoba untuk mengajak anda mengenali modul TMS320C6713DSK dari dekat. Kita akan mulai mengenali secara fisik tentang komponen penyusun modul ini, mempelajari spesifikasinya, mengetahui cara kerja dan hubungan antar bagian, dan memahami bagaimana board ini berhubungan dengan dunia luar. Pada bagian pertama pembahasan akan dibicarakan tentang pengenalan secara fisik dari DSP board TMS320C6713, dan pada bagian kedua akan dibahas bagaimana cara kita mengoperasikan perangkat ini.

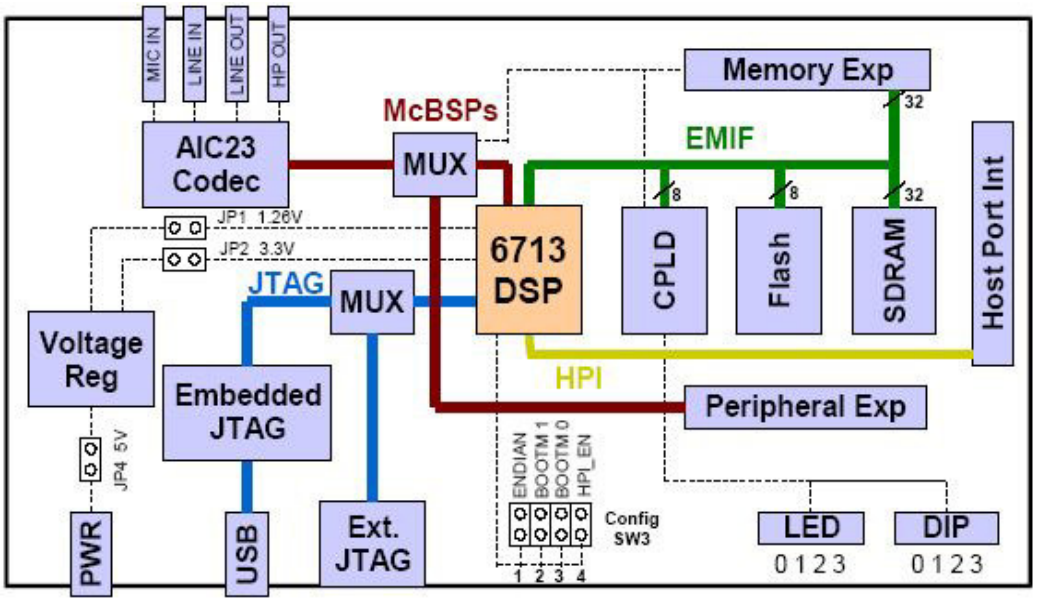
III. PENGENALAN TMS320C6713 SECARA FISIK

C6713 DSK merupakan suatu ***low-cost stand alone development platform*** yang memberi kesempatan kepada user untuk melakukan evaluasi dan membangun aplikasi sendiri berbasis keluarga DSP C67xx. DSK juga membantu sebagai sebuah hardware reference design untuk TMS320C6713 DSP. Secara fisik C6713 DSK bisa dilihat seperti gambar berikut ini.



Gambar 2. DSP Starter Kit TMS320C6713

DSK disusun dalam kemasan yang mana semua komponen pendukungnya disediakan pada suatu on-board devices yang memungkinkan digunakan dalam berbagai aplikasi dengan beragam keperluan yang berkaitan dengan pengolahan sinyal digital.



Gambar 3. Blok Diagram C6713 DSK

Untuk memudahkan dalam pemahaman, sebuah diagram skelmatik fungsional nya diberikan pada Gambar diatas. Disini ada beberapa fitur kunci seperti:

- Processor TMS320C6713 DSP dari Texas Instrument yang beroperasi pada 225 MHz
- Sebuah AIC23 stereo codec
- Memory synchronous DRAM 16 M byte
- Flash Memory non-volatile sebesar 512 Kbytes (256 Kbytes secara default digunakan)
- 4 user accessible LED dan DIP Switch

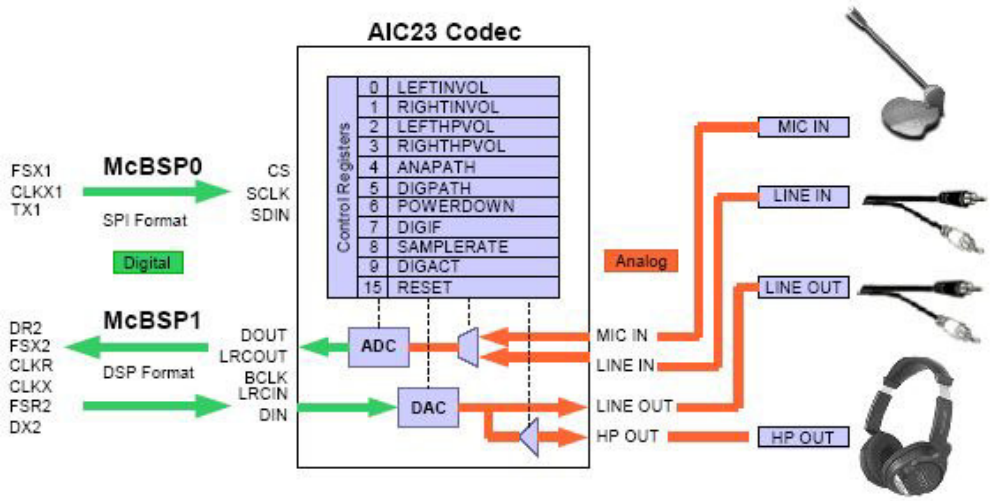
- Software board yang diimplementasikan pada CPLD
- Standar Expansion kokector
- JTAG emulation pada on-board JTAG emulator dengan USB Host atau external emulator
- Single power supply (+5 V)

DSK menggunakan suatu stereo codec dari Texas Instrument bertipe AIC23 untuk interface sinyal audio input dan output. Codec menyampel sinyal-sinyal analog pada microphone atau line input dan mengkonversikannya ke dalam bentuk data digital sehingga dapat diproses dengan DSP. Ketika DSP telah menyelesaikan pengolahan data (pemfilteran, dsb), maka codec ini juga bisa berfungsi dalam mengkonversi sinyal digital menjadi sinyal output analog melalui line out atau headphone.

Codec berkomunikasi menggunakan dua channel serial, satu untuk mengontrol register konfigurasi line codec, dan satunya untuk mengirim dan menerima sampel-sampel audio digital. McBSP0 digunakan sebagai unidirectional control channel. Ini seharusnya deprogram untuk mengirim suatu 16-bit control word ke AIC23 dalam format SPI. Ada 7 bit atas pada control word yang seharusnya menspesisikasi register untuk dimodifikasi dan ada 9 bit bawah yang berisi nilai-nilai register. Control channel hanya digunakan ketika mengkonfigurasi codec, pada umumnya dalam kondisi idle ketika data audio sedang ditransmisi.

McBSP1 digunakan sebagai bi-directional data channel. Semua audio data mengalir melalui data channel. Beberapa format data telah bisa disupport didsarkan pada tiga variable pada sample width, clock signal source, dan serial data format. Contoh-contoh DSK secara umum menggunakan sebuah lebar sampel 16-bit dengan codec dalam master mode sehinga mampu membangkitkan frame sync dan bil clocks pada sampel rate yang tepat tampa ada usaha exktra pada sisi DSP. Pilihan format serial adalah mode DSP yang dirancang secara khusus untuk beroperasi dengan McBSP port pada TI DSPs.

Codec memiliki suatu 12 MHz system clock. Sistem ini berkaitan dengan mode sampel rate pada USB, diberi nama ini karena banyak sistem USB yang menggunakan suatu clock 12 MHz dan dapat membangkitkan clock 12 MHz yang terpisah-pisah dalam bentuk frekuensi yang sudah umum seperti 48 kHz, 44.1 KHz, dan 8 KHz. Sampel rate di-set dengagn codec's SAMPLERATE register. Gambar berikut ini menunjukkan sebuah codec interface pada C6713 DSK.



Gambar 4. Codec Interface TMS320C6713 DSK

IV. CARA PENGOPERASIAN CODE COMPOSER STUDIO

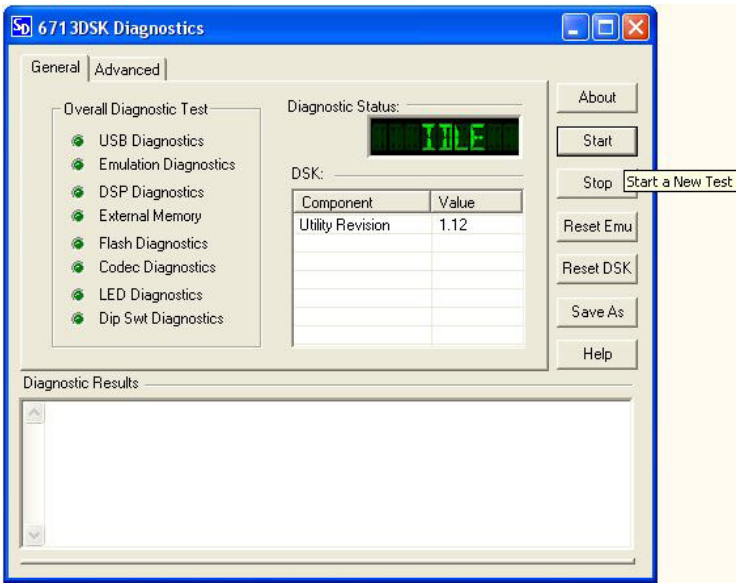
4.1. Mendiagnostik Koneksi CCS dengan komputer Anda

Sebelum anda mulai menggunakan Code Composer Studio untuk membut sebuah program yang berkaitan dengan TMS320C6713, anda harus memastikan bahwa perangkat anda sudah terhubung dengan baik dan siap untuk digunakan. Untuk mengetahui apakah DSK anda siap bekerja, anda ikuti beberapa instruksi berikut ini.

1. Perhatikan display monitor anda, anda cari dan double click ikon berikut ini

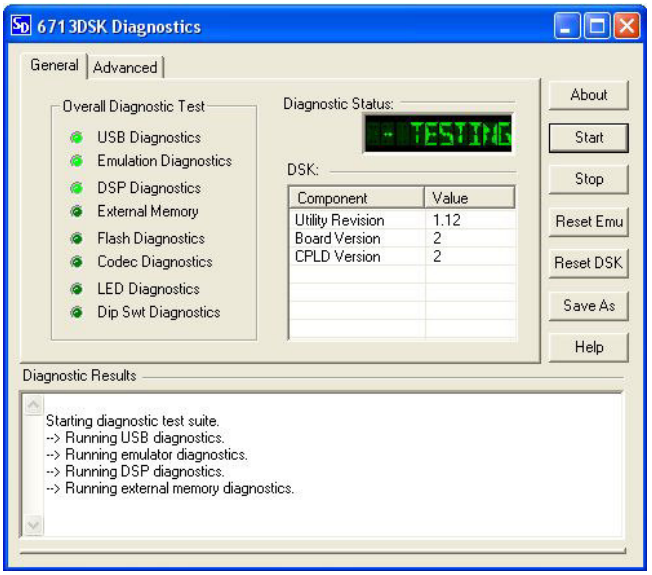
Gambar 5. Ikon Diagnostik DSK

Tampilan berikutnya yang muncul adalah sebuah Diaknostik untuk 6713DSK seperti berikut.



Gambar 6. Tampilan awal diagnostik DSK

2. Anda pasang sebuah headset pada lineout atau spk out pada DSK board.
3. Click pada Tombol Start, dan perhatikan proses diaknostik yang sedang berjalan. Pada display Overall Diagnostic Test seperti USB Diagnostics, Emulation Diagnostic, dst menunjukkan warna hijau dan pada DSK: Component menunjukkan informasi dan Pada Value menunjukkan angka-angka, maka proses diagnostic akan berjalan lancar dan tidak ada masalah. Jika pada salah satu komponen Diagnostic Test menunjukkan warna merah, maka ada satu masalah pada DSP Board anda. Pada umumnya, kesalahan terjadi karena pengguna DSP board terlupa memasang konektor USB.



Gambar 7. Proses Diagnostik DSK

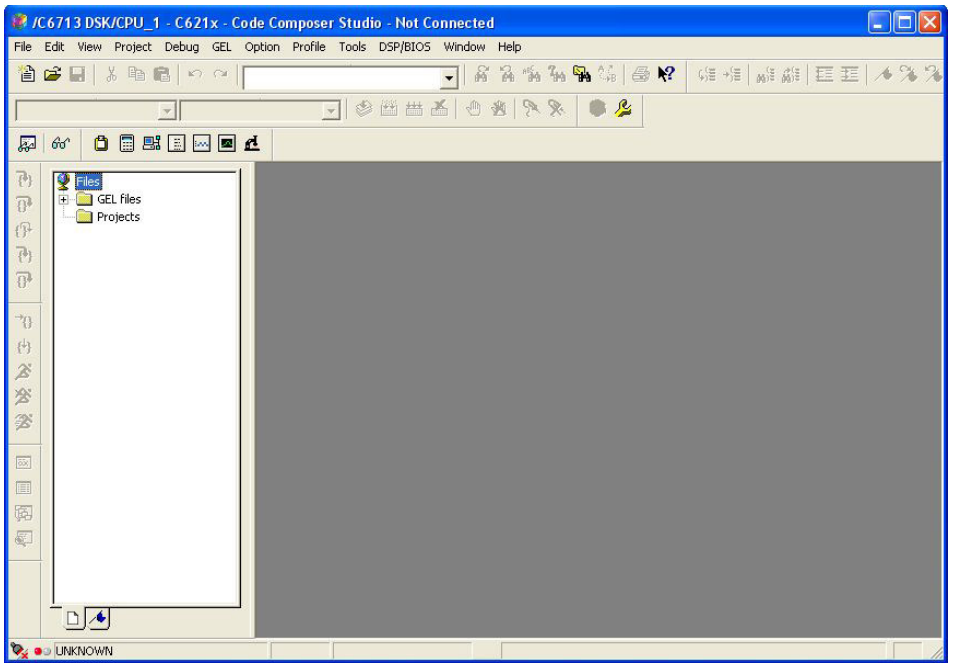
3.2. Membuka CCS

Ketika semua koneksi PC dengan DSP Board sudah, anda besa melanjutkan untuk memulai menggunakan DSP Board anda.

1. Perhatikan pada layar momonitor anda, jika pada display aca icon seperti gambar berikut ini, anda langsung click dua kali.

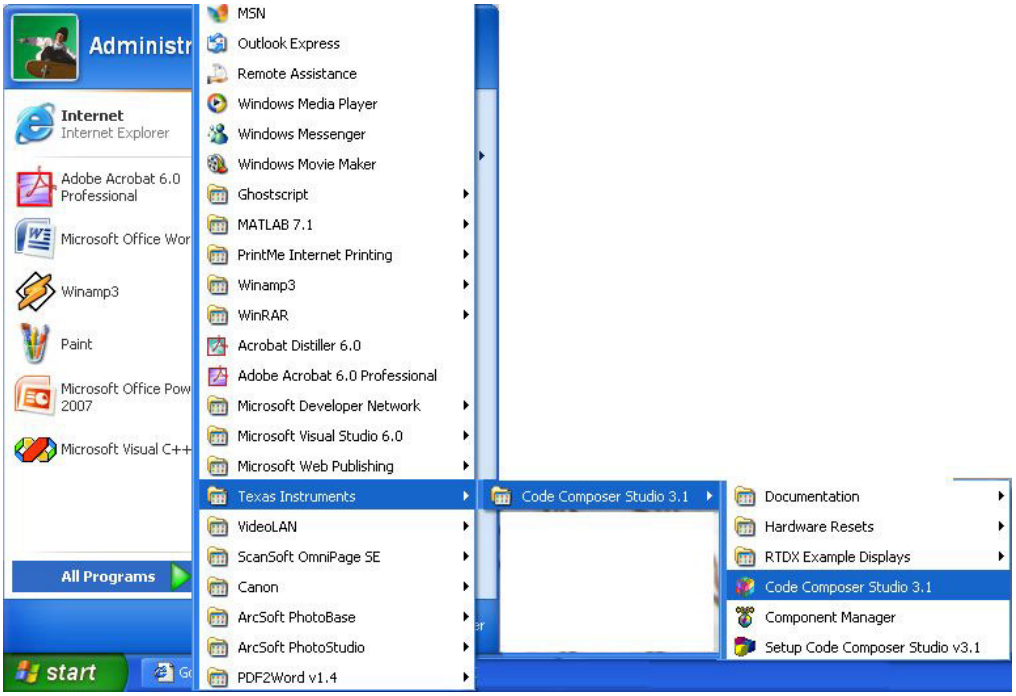
Gambar 8. Ikon CCStudio 3.1

2. Langkah tersebut akan membuka CCS anda dan muncul tampilan seperti berikut.



Gambar 9. Tampilan awal CCStudio 3.1 untuk TMS320C6713

3. Jika pada display monitor anda tidak terdapat icon seperti tersebut diatas, anda bisa melakukannya seperi berikut. Click pada **Start Windows →All Programs →Texas Instrument → Code Composer Studio 3.1→Code Composer Studio 3.1.** Langkah ini menghasilkan efek yang sama dengan langkah sebelumnya.



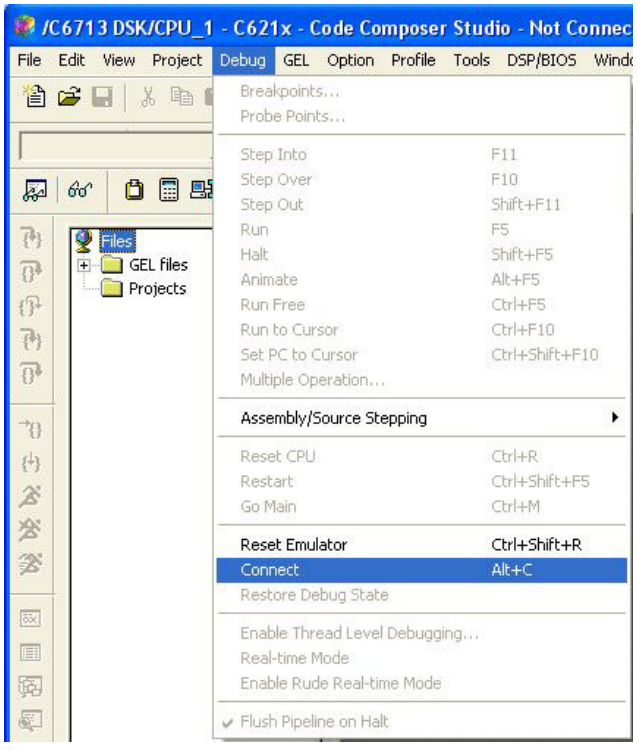
Gambar 10. Langkah alternative membuka CCStudio 3.1

3.3. Mengaktifkan CCS

1. Ketika pertama kali CCS telah terbuka, posisinya masih dalam kondisi belum siap untuk bekerja. Hal ini dikerenakan koneksi dengan PC Host belum ada. Untuk menghubungkan CCS anda dengan PC, anda perhatikan bagianujungkiri pada CCS di monitor yang sedang aktif. Ada ikon seperti kwas, ini menunjukkan bahwa CCS pada PC anda belum terhubung dengan DSP Board.

Gambar 11. Tanda TMS320C6713 dan CCS3.1 belum terhubung

2. Anda click pada Toolbar Debug→Connect. Perhatikan tampilan pada ujung kiri bawah CCS anda, jika kwas telah berwarna hijau dan tanda silang telah hilang diganti dengan komentar seperti dibawah ini, maka hubungan CCS pada PC Host dan DSP Board telah benar.



Gambar 12. Langkah menghubungkan TTMS320C6713 dengan CCS3.1

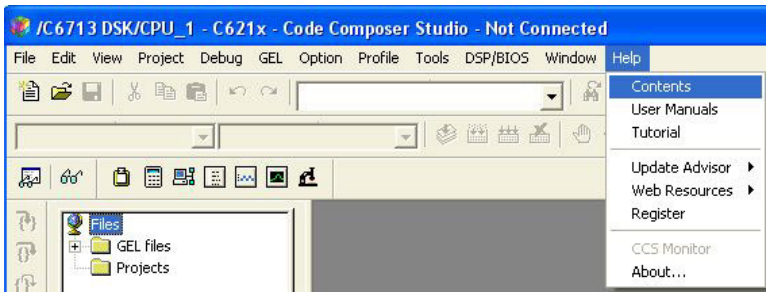
Gambar 13. Tanda TMS320C6713 dan CCS3.1 sudah terhubung

Setelah kondisi ini anda penuhi, anda siap untuk melangkah lebih lanjut yaitu membuat program sederhana, dst….

3.4. Memanfaatkan Help

Salah satu kelemahan siswa di PENS secara umum kurang memanfaatkan petunjuk yang diberikan sehinga cenderung bertanya atau menyatakan tidak bisa menjalankan. Alangkah baiknya jika anda mulai dengan membaca semua pentunjuk yang diberikan oleh pembuat barang. Denagn cara itu anda akan mudah sekali menjalankan suatu perangkat di depan anda.

1.

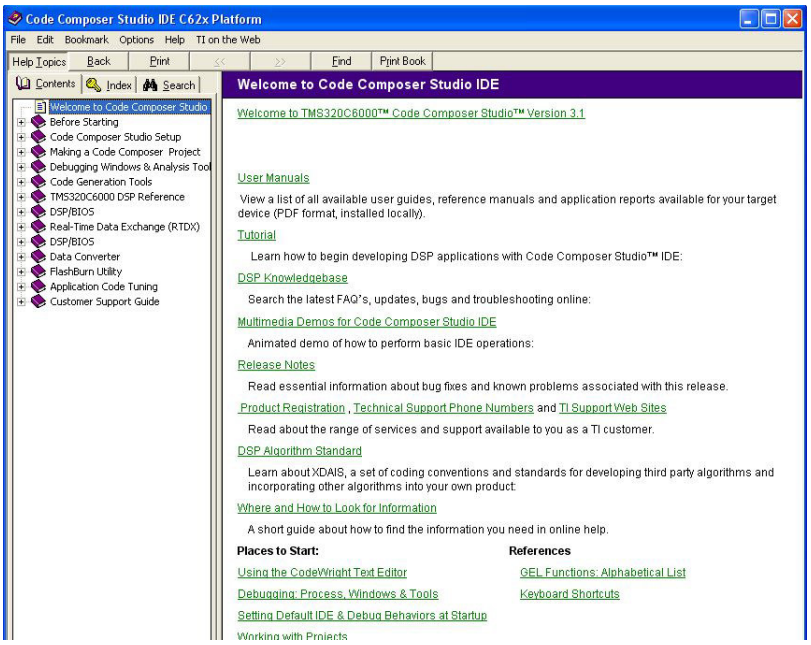


2. Click pada Help→Content, maka anda akan mendapatkan berbagai petunjuk terkait dengan cara penggunaan CCS anda. Bahkan pembuat perangkat ini akan memberi salam kepada anda…(smile)

Gambar 14. Penggunaan Fasilitas Help pada CCS CCS3.1

Contents

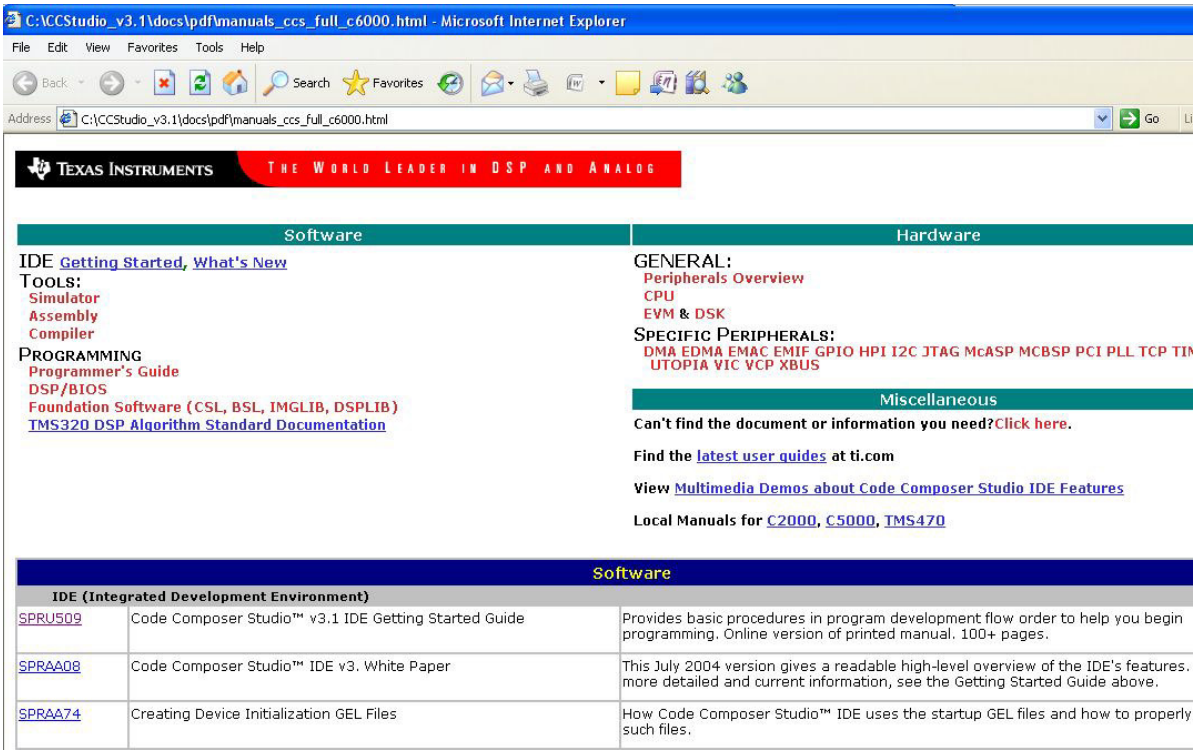
Fasilitas Help Content cukup banyak, anda dapat mengetahui berbagai informasi tentang CCS31 secar global.



Gambar 15. Fasilitas Help Contents pada CCS CCS3.1

User Manuals

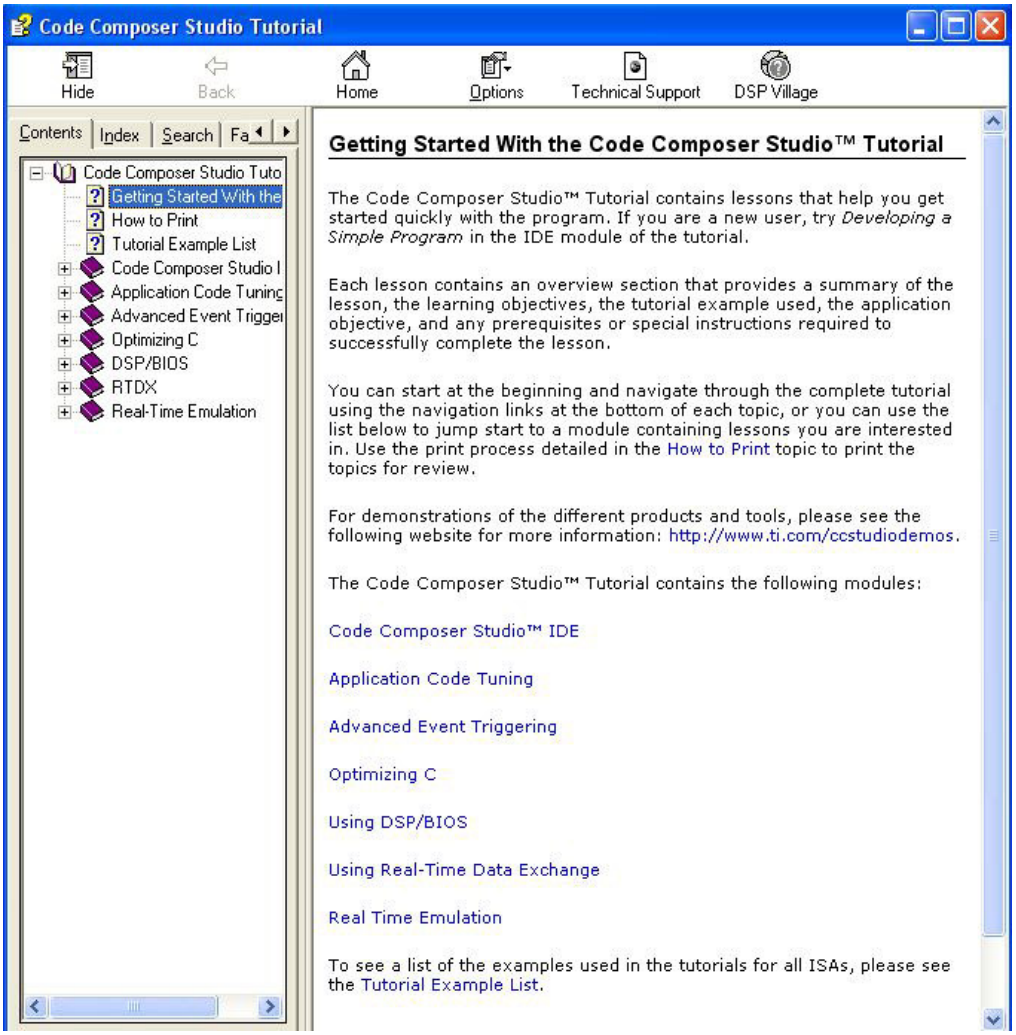
Fasilitas lain yang ada pada Help adalah User Manual. Langkahnya adalah **Help →User Manuals**. Disini berisi informasi tentang bagaimana arsitektur Processor TMS320C6713, bagaimana cara menggunakan CCS3.1, dsb yang sebagian besar, mungkin semau file tersimpan dalam *. Pdf.



Gambar 16. Fasilitas Help User Manuals pada CCS CCS3.1

Tutorial

Fasilitas lainnya yang disediakan oleh Help CCS adalah tentang Tutorial. Langkahnya adalah **Help→Tutorial.** Disini anda ditununtun sau persatu tentang bagaimana cara mengoperasikan CCS3.1 dan membangun sebuah progam dengan memanfaatkan TMS320C6713. Tutorial ini lebih bersifat interaktif, tetapi cenderung menampilkan perintah-perintah dasar. Jika anda mendalami lebih lanjut tentang TMS320C6713 sebaiknya anda membaca seluruh User Manual dari CCS atau membaca tutorial dari buku lain.



Gambar 17. Fasilitas Help Tutorial pada CCS CCS3.1

MODUL II

PENGENALAN MATLAB UNTUK PENGOLAHAN SINYAL DIGITAL

I. TUJUAN INSTRUKSIONAL

- Siswa memahami perintah dasar pada Matlab
- Siswa mampu menyusun program sederhana yang berhubungan dengan dasar pengolahan sinyal

II. MATERI PEMBELAJARAN MATLAB

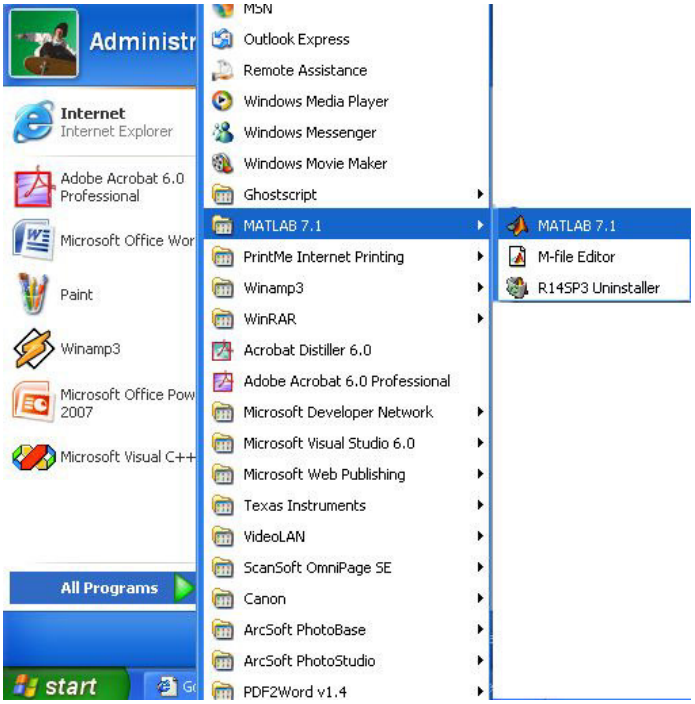
Matlab merupakan paket komersial ***'Matrix Laboratory'*** yang beroperasi sebagai suatu *environment* pemrograman yang interaktif. Matlab melebihi kelebihan di dalam kemampuan ***environment komputasional*** dan sebagai bahasa pemrograman yang mudah untuk menangani matrix dan aritmatik yang komplek. Perangkat ini bisa dikembangkan oleh pembuat programmnya, dan bisa juga digunakan sebagai suatu tool satandar untuk berbagai pekerjaan keilmuan dan bidang rekayasa. Dibanding perangkat lunak lainnya, perangkat ini memiliki kelebihan dalam penggamaran dengan mudah untuk bentuk dua dimensi dan tiga dimensi.

Matlab memiliki dua metode berbeda untuk melakukan eksekusi *command* (perintah): *interactive mode* dan *batch mode*. Di dalam ***interactive mode***, command diketikkan (atau cut-and-pasted) ke dalam *'command window'*. Di dalam ***batch mode***, sederetan commands disimpan dalam bentuk text file (menggunakan Matlab's built-in editor, atau text editor lainnya seperti Notepad) dengan suatu eksetensi **'.m'**. ***Batch command*** dalam suatu file selanjutnya dieksekusi dengan mengetikkan nama file pada prompt ***Matlab command***. Pada ***Matlab's built-in editor*** versi yang baru anda bisa langsung melakukan ekseskusi program yang anda buat. Keuntungan mengunakan suatu file '.m' adalah memberi keleluasaan bagi anda untuk melakukan sedikit perubahan pada kode anda.

2.1. Membuka Matlab

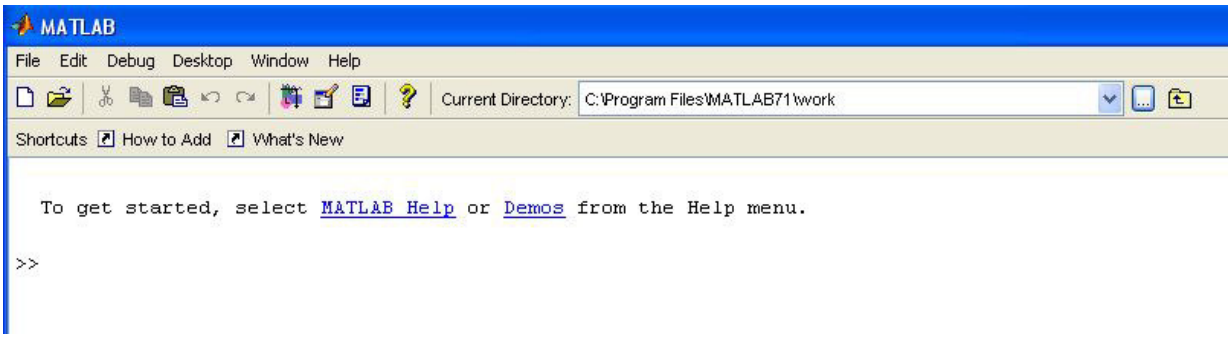
Anda dapat memulai Matlab dengan cara membukanya dengan click duakali mouse anda pada ikon Matlab yang ada di display monitor anda. Bisa juga anda mencari melalui Windows Start→All Program→Matlab→ dst…

Gambar 1. Ikon Matlab pada Display monitor



Gambar 2. Langkah membuka Matlab melaluiWindows Start

Langkah ini akan menyebabkan munculnya **Matlab Command** anda yang pertama. Tunggu beberapa ssaat sampai tampilan seperti pada Gambar 3 muncul. Dalam hal ini anda harus bersabar, karena setiap computer memiliki kemampuan berbeda dalam menjalankan aplikasi. Hal ini dipengaruhi oleh kemampuan processor, motherboard, dan memory yang dimilikinya. Bagi yang menggunakan computer dengan spesifikasi hardware rendah, dimohon bersabar….



Gambar 3. Tampilan Matlab Command

2.2. Perintah Help

Pepatah mengatakan, "***malu bertanya sesat di jalan***", hal ini juga berlaku bagi anda yang ingin mempelajari perangkat lunak seperti Matlab. Akan lebih mulia jika anda anda memanfaatkan fasilitas help yang tersedia di Matlab. Hal ini akan sangat membantu anda untuk menyelesaikan persoalan. Sebab jika anda terlalu sering bertanya kepada teman sebelah

anda, belum tentu dia mau menjawab dengan senang hati.

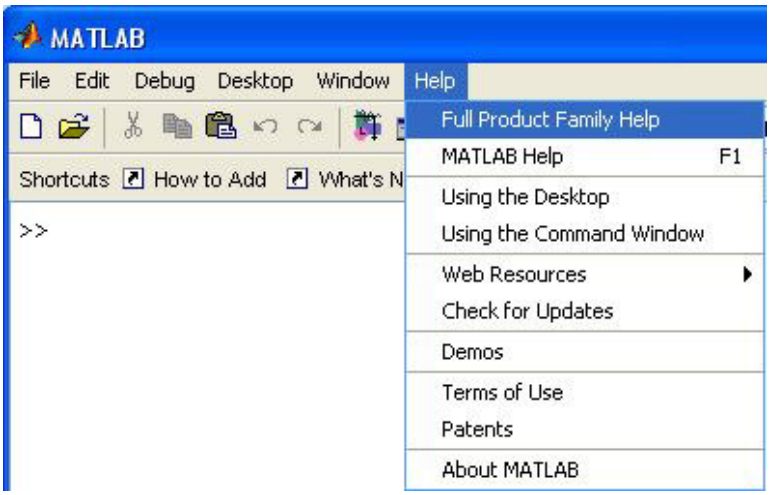
Jika anda sudah mengetahui persoalan dengan jelas, anda bisa memanfatkan Matlab Command dengan cara mengetik help dilanjut dengan fungsi yang anda ingin ketahui. Misalnya anda ingin mengetahui cara menggunakan persamaan sinusoida, maka anda lakukan langkah seperti dibawah ini.

```
>> help sin
```

Maka akan tampil seperti berikut.

```
SIN Sine.
SIN(X) is the sine of the elements of X.
Overloaded methods
help sym/sin.m
```

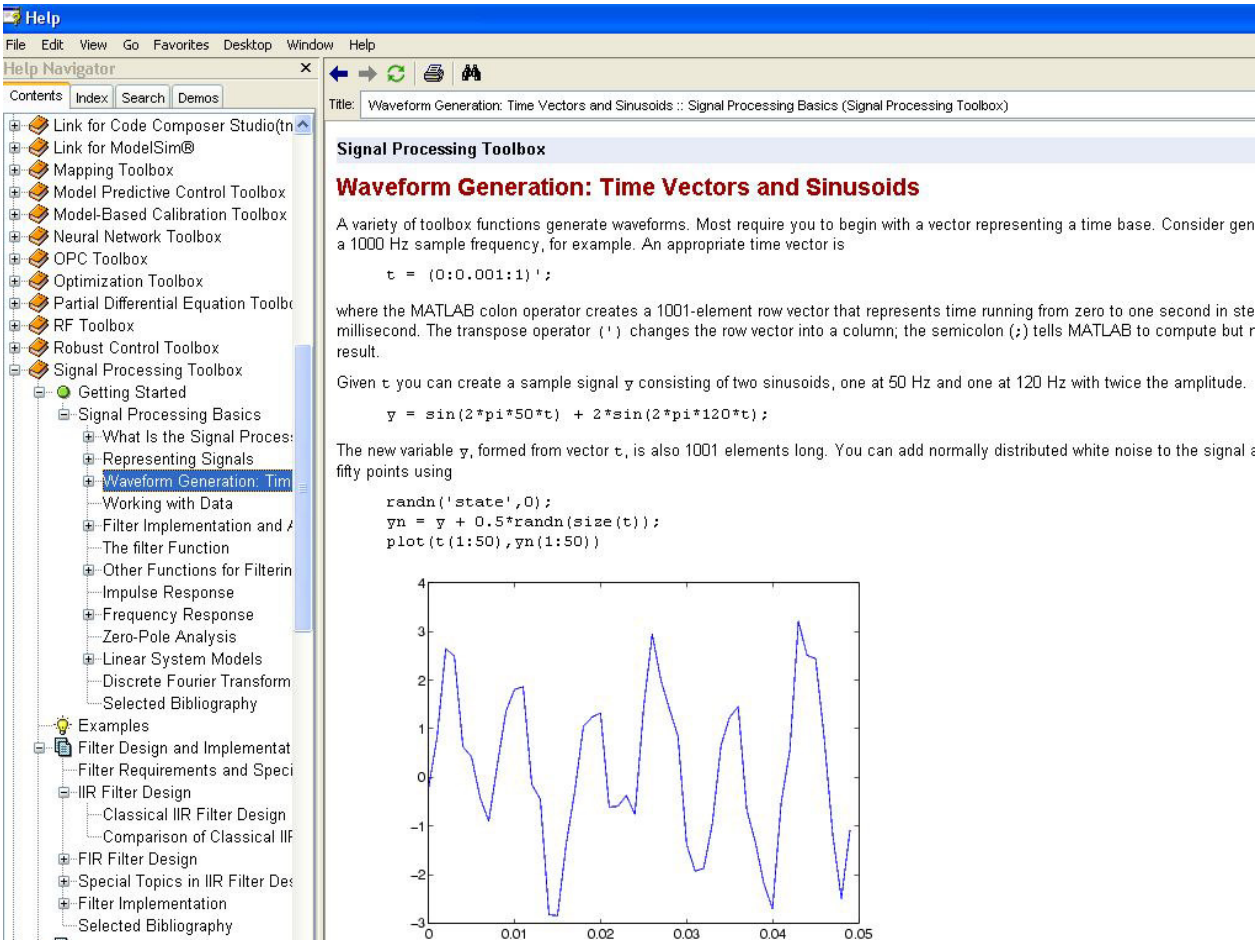
Jika anda hanya mengetahui kerangka besar persoalan, tetapi tidak tahu fungsi apa yang anda tanyakan, maka anda bisa memanfaatkan fasilitas help dengan cara melihat demo yang sudah disediakan oleh Matlab. Caranya adalah anda click toolbar ***Help→Full Product Family Help.***



Gambar 4. Pemanfaatan Help Matlab

Dengan cara ini Matlab akan menampilkan seluruh fasilitas help yang dimiliki. Dengan demikian and tinggal pilih jenis permasalahan yang ingin anda selesaikan. PIlih salah satu, dan disitu akan ditampilkan contoh program dan pembahasannya.

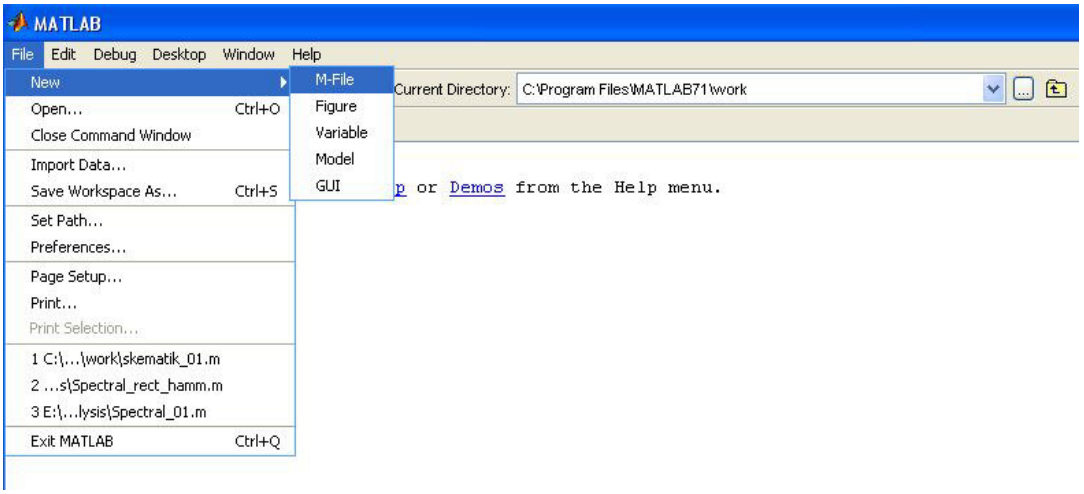
Pada contoh berikut ini kita ingin melihat cara membuat program untuk Signal Processing, yang secara lebih spesifik adalah pembangkitan sinyal. Kita click pada tanda '+' di depan **Signal Processing Toolbox→Getting Started→Waveform Generation**. Langkah ini akan memunculkan tampilan seperti dibawah ini.



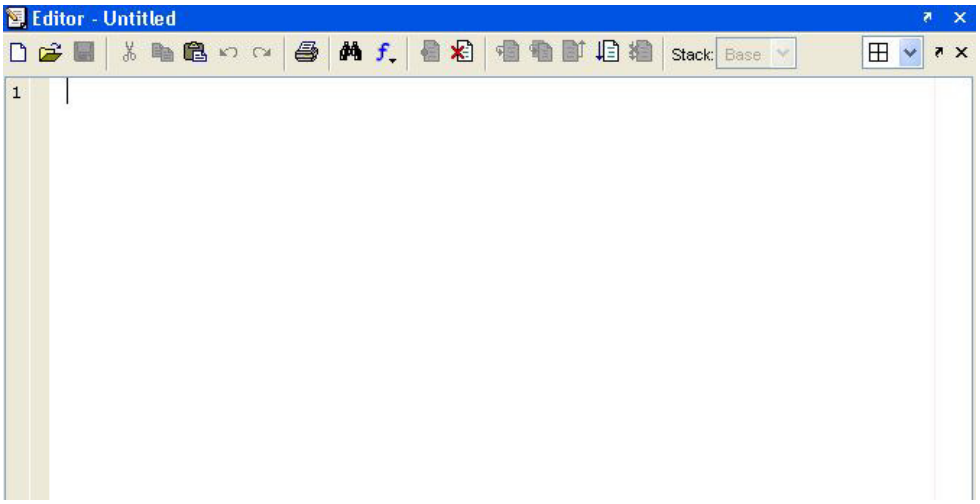
Gambar 6. Pemanfaatan Help Signal Processing

2.3. Membuat File *.m

Untuk membuat sebuah program Matlab berbasis Batch command, maka akan berurusan dengan pembuatan sebuah m-file atau file ***.m**. Pembuatan program dengan cara ini sangat seerhana, anda mulai dari membuat file dengan click pada File→New→M-File seperti tampak pada Gambar7. Maka akan muncul sebuah Matlab Editor seperti pada Gambar 8, disini anda diberi kebebasan untuk menuliskan program sesuai dengan persoalan yang ingian anda selesaikan. Posisi Matlab Editor biasanya terpisah dari Matlab Command, tetapi ada juga yang posisinya menjadi satu dengan Matlab Command. Jika anda menggunakan seri Matlab 7, bisanya Matlab Editor terintegrasi dengan Matlab Command.



Gambar 7. Memulai Membuat File.m

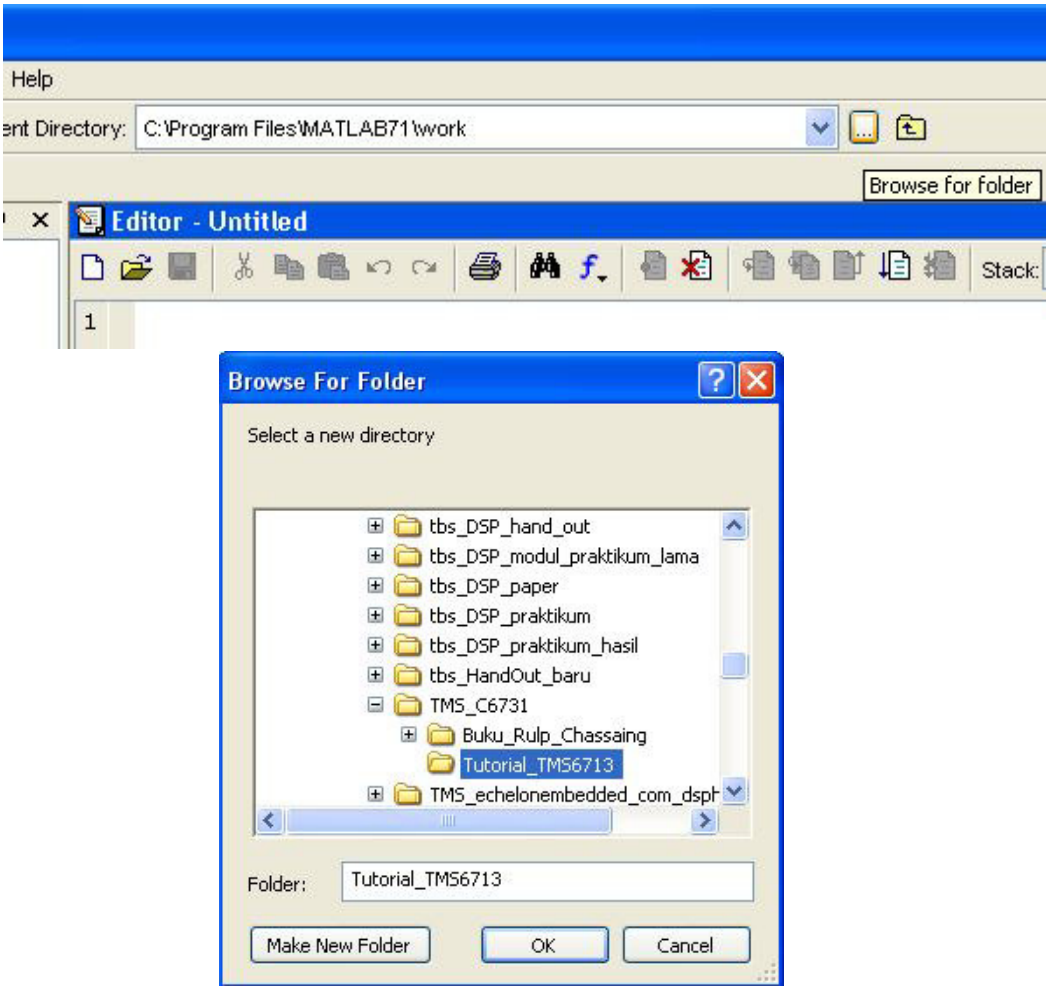


Gambar 8. Tampilan Matlab Editor

2.4. Menentukan Direktori Tempat Bekerja:

Anggap anda sudah membuat program dengan menggunakan Matlab Editor, pada waktu anda ingin melakukan eksekusi yang pertama kali, biasanya anda diminta untuk menyimpan file *.m tersebut. Pada waktu melakukan penyimpanan bisanya ditanyakan apakah menggunakan folder default atau anda akan memindah ke folder yang lain.

And bisa saja menentukan folder tempat anda menyimpan progam anda sebelum pembuatan program dimulai, caranya adalah dengan click pada toolbar Browse for folder yang berupa kotak dengan tanda titik-titik seperti pada Gambar 9. Ketika anda click tanda itu, otomatis akan muncul tampilan pilihan folder dimana yang anda inginkan. Jika anda memilik salah satu cukup click di folder terebut, tetpi jika anda ingin membuat folder yang baru juga diberi keleluasaan melalui **Make New Folder**.



Gambar 9. Menentukan Folder Kerja

III. PERALATAN PERCOBAAN

Untuk pelaksanaan kegiatan praktikum ini diperlukanperalatan percobaan sebagai berikut:

- Komputer dengan Spesifikasi Minimal Memory 512 KB, dilengkapi dengan perangkat Audio pendukung
- Operating System Windows
- Perangkat Lunak Matlab

IV. PERCOBAAN

Pada awal percobaan ini, anda bisa memanfaatkan Matlab Command atau juga dikenal

sebagai Matlab Interactive Mode untuk membuat program-program sederhana. Kita mulai dari percobaan melakukan perhitungan sederha aritmatika dan dilanjutkan dengan bentuk perulangan dengan memanfaatkan looping.

4.1. Memulai Suatu Operasi Aritmatika

Bentuk Jumlahan:

Misalnya anda ingin melakukan operasi penjumlahan antara variable a dan variable b, dengan nilai-nilai yang sudah anda tetapkan. Untuk itu anda bisa melakukannya dnegan cara mengetikkan seperti berikut ini

```
a=2; b=3;
c=a+b

c =
5
```

Bagian terakhir yang muncul ‘c = 5’ merupakan hasil eksekusi dari Matlab Command. Proses ini akan berjalan cukup cepat, sebab hanya melibatkan persamaan sederhana. Coba anda lanjutkan dengan melakukan operasi penjumlahan y = x1+x2+x3+x4+x5, dimana x1=10, x2=1, x3=8, x4=9 ,dan x5=2.

Bentuk Perkalian:

Dalam kasus ini anda diminta untuk menghitung volume pada suatu bola dengan jari-jari r= 2. Anda ingat bahwa persamaan matematik untuk volume suatu bola dengan jari-jari r adalah:

$$Volume = (4/3)\pi r^3$$

Nilai ‘π’ di dalam Matlab bisa diwakili oleh notasi ‘pi’. Anda dapat menyelesaikan dengan mengetik seperti berikut:

```
r=2;
vol=(4/3)*pi*r^3;

vol
```

Hasilnya adalah:

```
vol =
33.5103
```

Perhatikan perbedaan contoh pertama dengan contoh kedua. Pada contoh pertama setelah c=a+b tidak dilengkapi dengan tanda *semicolon* ‘;’, hal ini akan menyebabkan Matlab secara langsung akan menampilkan hasil eksekusi. Sedangkan pada contoh kedua, vol=(4/3)*pi*r^3; , pada contoh ini tanda ‘;’ menyebabkan Matlab tidak mengeksekusi dan menunjukkan

hasilnya secara langsung. Jika anda masih penasaran anda bisa mencoba untuk tidak menggunakan ';'. Cobalah untuk mencoba sekali lagi pada permasalahan penghitungan luasan sebuah persegi panjang yang sisi-sisinya adalah a = 10, dan b = 12.

Bentuk Pembagian:

Operasi pembagian di dalam Matlab memiliki hirarki sama dengan operasi perkalian, untuk melakukan pembagian c=a/b dengan masing-masing varbiabel bernilai a=34, dan b = 3, anda dapat melakukannya dengan mengetikkan perintah seperti berikut.

```
a=34;
b=3;
c=a/b
```

Hasilnya adalah

```
c =
11.3333
```

Coba anda lakukan sebuah bentuk kombinasi antara perkalian/pembagian dan penjumlahan/pengurangan untuk kasus yang dituliskan dalam persamaan seperti dibawah ini: y = ax + b/c. Dalam hal ini nilai a =2, x = 1.7, b=10, dan c=2.5.

4. 2. Bentuk Perulangan

Ada beberapa cara untuk melakukan perulangan dalam Matlab, bisa menggunakan for atau anda cukup menentukan suatu variable sebagai matrik yang memiliki rentang tertentu. Pada contoh pertama ini anda akan melakukan sebuah operasi perulangan dengan memanfaatkan for… end. Disini anda menentukan nilai x dari 1 sampai 9, dan anda akan memproses sebuah bentuk operasi yang melibatkan variable x tersebut. Anda dapat melakukannya dengan cara seperti berikut.

```
for x=1:9,
   y=x.^2-5*x-3
end
```

Maka di dalam Matlab Command akan muncul hasil seperti berikut.

```
y = -7
y = -9
y = -9
y = -7
y = -3
y = 3
y = 11
y = 21
y = 33
```

Anda juga bisa membuat sebuah bentuk perulangan dengan cara yang lebih sederhana seperti

berikut

```
x=1:9;
y=x.^2-5*x-3
```

Hasilnya adalah

```
y=
    -7    -9    -9    -7    -3     3    11    21    33
```

Pada perintah pertama, disebutkan x=1:9, hal ini berarti anda menetapkan bahwa x bernilai 1 sampai 9 dengan step kenaikan sebesar 1. Jika anda merubah perintah diatas dengan x=1:.5:9, maka proses kenaikan nilai x adalah 1, 1.5, 2.0, 2.5 ….9. Hal ini terjadi karena step kenaikannya anda tetapkan sebesar 0.5(dalam bahasa Indonesia 0,5).

4.3. Bentuk Input

Dalam percobaan ini anda bisa saja melakukannya dalam mode Matlab Command, tetapi akan lebih bermanfaat jika anda menyusun percobaan dalam Matlab Batch Mode atau penulisan program menggunakan Matlab Editor.

Pada contoh berikut ini anda akan melakukan penghitungan nilai volume sebuah bola yang memiliki jari-jari (radius) ydng ditetapkand alam variabel r. Proses pemasukan nilai variable r dilakukan dengan memanfaatkan perintah fungsi ‘input’. Langkah yang harus anda lakukan adalah dengan cara membuat sebuah program dengan Matlab Editor sbb.

```
%File Name: hitung_01.m
r=0;
while r<10
r=input('Masukkan nilai radius: ');
if r<0,break,end
vol=(4/3)*pi*r^3;
fprintf('Volume= %7.3f\n',vol)
end
```

Simpan program anda dengan nama hitung_01.m, untuk sementara anda biarkan saja Matlab menentukan lokasi folder anda bekerja secara default, dalam hal ini berarti dia berada di folder ***work*** pada Matlab. Lanjutkan dengan menjalankan program anda dengan memanfaatkan toolbar run yang ada di Matlab Editor. Pada Matlab Command akan muncul tampilan seperti berikut.

Masukkan nilai radius:

Anda ketikkan angka ‘3’, maka akan keluar hasilnya `Volume= 113.097`

Program seperti diatas akan berguna jika anda menginginkan sebuah program yang mana nilai variabelnya ingin anda masukkan dan anda rubah secara bebas tanpa berurusan dengan **Matlab Editor**. Anda bisa juga melakukan eksekusi program anda diatas dengan cara mengetikkan ‘hitung_01’ pada **Matlab Command**. Selanjutnya akan muncul tampilan yang sama seperti ketika anda menjalankan program melalui Matlab Editor.

4.4. Membuat Fungsi

Sebuah fungsi sangat bermanfaat pemrograman yang panjang dan melibatkan banyak perhitungan-perhitungan yang cukup rumit. Dengan membuat sub program –sub program yang terpisah dalam setiap fungsi berbeda, anda akan dengan mudah mengetahui dan mengoreksi kesalahan yang terjadi. Dengan cara ini anda bisa berfikir secara terstruktur dan menghemat memori otak anda agar terbebas dari deretan syntak yang panjang dari sebuah program.

Sebagai pengenalan disini kita coba untuk membuat sebuah fungsi penghitungan nilai x dengan persamaan seperti berikut

$$y = \frac{(2x^3 + 7x^2 + 3x - 1)}{(x^2 - 3x + 3e^{(-x)})} \qquad (1)$$

$$y = \frac{(2x^3 + 7x^2 + 3x - 1)}{(x^2 - 3x + 3e^{(-x)})} \qquad (1)$$

Untuk merealisasikannya kita bisa membuat programnya dalam **Matlab Editor** seperti berikut ini.

```
function y=demof_(x)
y=(2*x.^3 + 7*x.^2 + 3*x-1)./(x.^2-3*x + 5*exp(-x));
```

Anda simpan program diatas dengan nama demof_.m, menyesuaikan dengan bentuk fungsi yang anda buat. Program diatas tidak menghasilkan apa-apa jika anda mengeksekusinya, sebab variable x dalam hal ini belum diberi nilai. Agar program diatas bisa bekerja anda bisa memanggilnya melalui program Matlab yang lain. Anda bisa membuatnya dalam **Matlab Editor** atau secara langsung anda mengetikkan di **Matlab Command**.

```
x=0:1:10;
y=demof_(x);
y
```

Coba anda amati hasilnya seperti apa?...

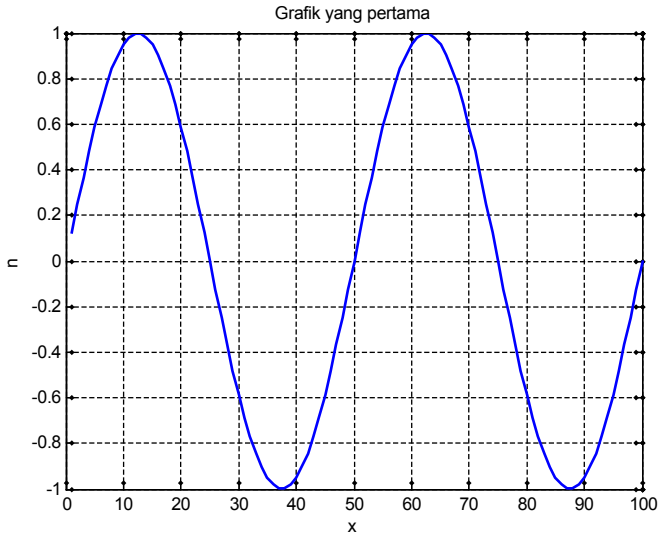
4.5. Membuat Grafik

Satu kelebihan dari Matlab adalah dalam hal pembuatan grafik dari sebuah persamaan matematika. Jika anda menginginkan untuk melihat hasil secara cepat dan berorientasi pada

analisa, maka cara berikut ini merupakan langkah yang tepat. Anda mulai membuat grafik untuk pembangkitan sinyal sinusoida yang memiliki frekuensi 2 Hz, dan amplitude sebesar 1. Caranya adalah dengan membuat sebuah program pada Matlab Editor seperti berikut ini.

```
%File Name:graph_1.m
T=100;
f=2;
n=1/T:1/T:1;
x=sin(2*pi*f*n);
plot(x,'linewidth',2)
title('Grafik yang pertama')
xlabel('x');ylabel('n');
grid
```

Jika anda selesai dan telah menyimpan program tersebut, anda tekan toolbar Run, maka akan muncul tampilan sebagai berikut.



Gambar 10. Tampilan Sinyal Sinus

Tampilan diatas menunjukkan bahwa sinyal sinus terbangkit sebagai fungsi dari deretan nilai-nilai x, yang dalam hal ini adalah urutan sampelnya. Tentu saja grafik diatas belum sepenuhnya mewakisi sebuah sinyal sinusoida yang benar. Coba anda lakukan modifikasi program anda diatas. Anda sisipkan beberapa perintah berikut ini setelah baris ke-5.

```
nn=1:length(n);
plot(nn/T,x,'linewidth',2)
```

Anda amati perbedaan hasilnya, dan jangan lupa mencatatnya untuk analisa.

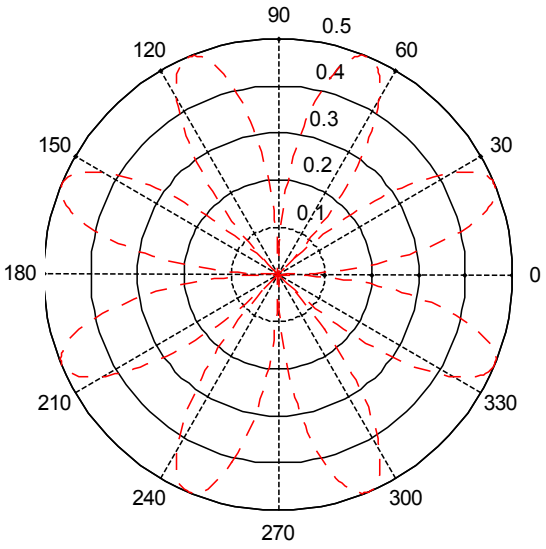
Membuat Grafik Polar

Fungsi polar menerima koordinat polar, kemudian menggambarkannya dalam bentuk koordinat polardi dalam sebuah bidang Cartesian, dan menampilkan sebuah grid pada bidang tersebut. Sebuah sintak berbentuk dasar ***polar(theta,rho)*** akan menghasilkan sebuah koordinat polar dengan sudut **theta** sebagai fungsi radius **rho**. Untuk selanjutnya anda bisa mencoba sebuah program sederhana seperti dibawah ini. Jika anda masih penasaran dengan, ana bisa

mendapatkan bantuan dari Matlab melalui help dari library yang ada.

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t),'--r')
```

Jika anda menjalankan program ini, akan muncul tampilan seperti berikut.



Gambar 11. Tampilan Polar Plot

Akan lebih baik jika anda mencoa untuk merubah tampilan dengan cara memodifikasi baris ke-2 pada program diatas menjadi sebagai berikut.

```
polar(t,sin(2*t).*cos(2*t))
```

Perhatikan apa yang terjadi, dan cobalah dengan berbagai variasi pada baris tersebut. Untuk ini anda harus membuat catatan dan melakukan analisa dari langkah anda.

```
polar(t,sin(2*t).*cos(2*t),'--b')
polar(t,sin(2*t).*cos(2*t),'*')
```

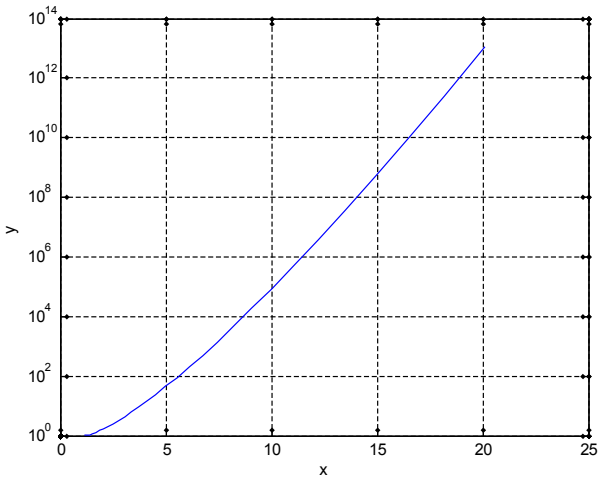
Grafik Semilog

Sintak ‘semilogx’ dan ‘semilogy’ akan menggambarkan plot data dalam bentuk terskala logarithmic pada sumbu x dan sumbu y. Semilogx(Y) menghasilkan sebuah gambaran menggunakan suatu nilai terskala logaritma basis 10 pada sumbu-xdan suatu tampilan terskala linear pada sumbu y. Untuk lebih jelasnya, anda bisa memulai dengan sebuah program sederhana berikut ini.

```
%File Name:graph_4.m
t=.1:.1:3;
x=exp(t);
```

```
y=exp(t.*sinh(t));
semilogy(x,y)
grid
xlabel('x');ylabel('y')
```

Ketika anda menjalankan program tersebut, maka akan muncul tampilan seperti berikut



Gambar 12. Tampilan Grafik Semilogy

Coba anda lakukan modifikasi bagian program `semilogy(x,y)` menjadi seperti berikut `loglog(x,y)`, lanjutkan dengan merubah bagian beriktu ini

```
x=exp(t);
y=exp(t.*sinh(t));
semilogy(x,y)
```

menjadi

```
y=exp(t);
x=exp(t.*sinh(t));
semilogx(x,y)
```

Anda amati hasilnya, catat dan lakukan analisa dari langkah yang sudah anda lakukan.

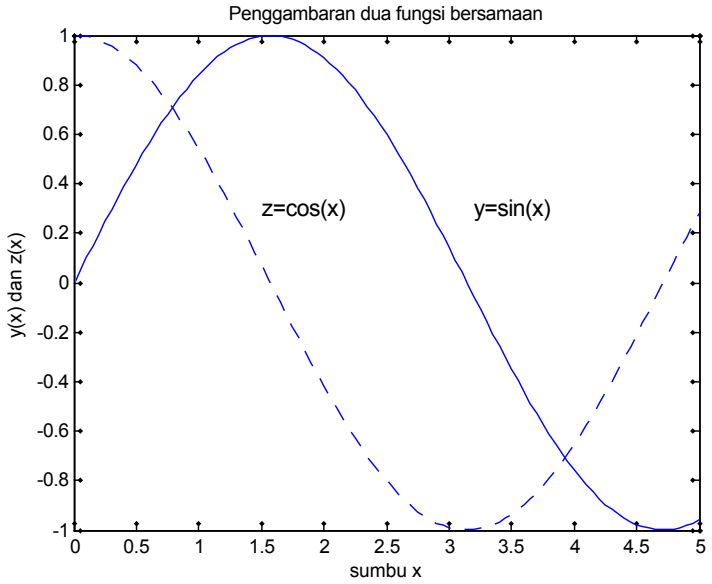
Grafik Dua Fungsi

Untuk menampilkan dua buah fungsi menjadi sebuah grafik merupakan hal yang menarik. Anda bisa memulainya dengan menuliskan program untuk membangkitkan dua buah sinyal sinuoida, yang pertama dalah sinyal sinus, sedang yang kedua adalah sinyal cosinus. Jangan lupa anda menampilkan text sebagai penjelasan bagian mana yang menggambarkan sinyal sinus, dan bagian mana yang menggambarkan sinya cosinus. Jika anda merasa mengalami kesulitan, tidak ada salahnya anda buat program seperti dibawah ini.

```
%File Name:graph_5.m
x=0:0.05:5;
y=sin(x);
plot(x,y)
```

```
hold on
z=cos(x);
plot(x,z,'--')
title('Penggambaran dua fungsi bersamaan')
xlabel('sumbu x');ylabel('y(x) dan z(x)')
text(3.2,0.3,'y=sin(x)','fontsize',12)
text(1.5,0.3,'z=cos(x)','fontsize',12)
```

Jika program diatas dijalankan, maka akan muncul tampilan seperti berikut.



Gambar 13. Tampilan Dua Grafik dan Text Penjelasan

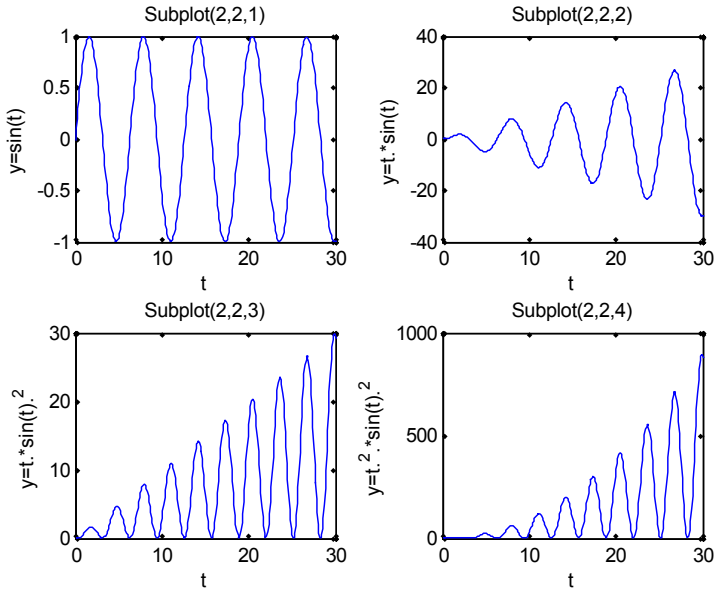
Tidak salah jika anda penasaran apakah kemampuan Matlab hanya membuat dua sinyal secara bersamaan. Coba anda modifikasi program diatas untuk menampilkan 3,4 atau bahkan 10 sinyal secara bersamaan. Anda tidak perlu ragu dahal hal ini, tohresiko terburuk dari langkah anda adalah Matlab akan mengalami hang, alias kecapekan dalam mengeksekusi program anda….(smile)

Grafik dengan Banyak Tampilan (Subplot)

Jika anda ingin menampilkan beberapa grafik dalam frame terpisah-pisah, anda bisa memanfaatkan subplot. Sintak **subplot(m,n,p)** atau **subplot(mnp)** akan memecah gambar menjadi m x n matrik dengan axis kecil. Hal ini akan menghasilkan sebuah gambar yang tersusun dari m baris dan n kolom. Sementara p menyatakan proses penggambaran pada urusat ke-p yangakan diproses. Misalnya, anda menyebutkan sintak **subplot (2,2,1)**, maka akan muncul sebanyak 2 x 2 gambar, dan perintah **plot (y)** setelah sintak tersebut akan menempatkan grafik y pada gambar nomor 1 yang terletak di pojok kiri atas. Urutan untuk gambar adalah kiri atas, turun ke bagian bawahnya sesuai jumlah maksimum baris yang dibuat, dalam contoh tersebut akan muncul 2 baris. Kemudian dilanjutkan dengan penggambaran pada kolom berikutnya, mulai dari atar, menurun dst. Belum juga paham?...Coba anda buat program seperti berikut ini.

```
%File Name: graph_6.m
clear;clf
t=0:.1:30;
subplot(2,2,1)
y=sin(t);
plot(t,y),title('Subplot(2,2,1)'),ylabel('y=sin(t)'),xlabel('t')
subplot(2,2,2)
y1=t.*sin(t);
plot(t,y1),title('Subplot(2,2,2)'),ylabel('y=t.*sin(t)'),xlabel('t')
subplot(2,2,3)
y=t.*sin(t).^2;
plot(t,y),title('Subplot(2,2,3)'),ylabel('y=t.*sin(t).^2'),xlabel('t')
subplot(2,2,4)
y=t.^2.*sin(t).^2;
plot(t,y),title('Subplot(2,2,4)'),ylabel('y=t.^2.*sin(t).^2'),xlabel('t'
)
```

Program diatas akan menghasilkan tampilan seperti berikut ini.



Gambar 14. Tampilan Subplot 4 Frame Gambar

Grafik 3 Dimensi

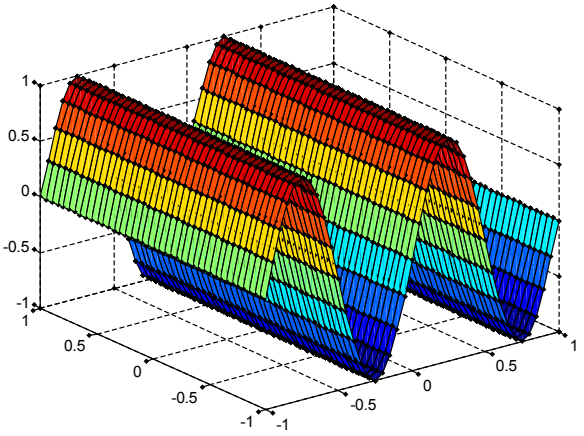
Penggambaran grafik 3 dimensi merupakan hal yang sangat menarik, tetapi seringkali kita terbentur dengan kesulitand alam merealisasikannya. Dengan bantuan Matlab library, anda akan dengan mudah membuat sebuah grafik 3 dimensi yang anda inginkan.

Library **mesh(X,Y,Z)** merupakan sebuah gambar wireframe mesh dengan wara yang ditentukan oleh z, sedemikian hingga akan proporsional dengan level ketinggian permukaan yang dihasilkan dari nilai fungsinya. Jika X dan Y adalah vector, panjang (X) = n dan panjang (Y) = m, dimana [m,n] = ukuran dari (Z). Dalam hal ini, (X(j), Y(i), Z(i,j)) merupakan bentuk interseksi pada wireframe grid lines; X dan Y berkaitan dengan kolom-kolom dan baris-baris pada Z.

Jika penjelasan ini membuat kepala anda menjadi pusing, tidak ada salahnya anda langsung saja mencoba dengan membuat sebuah program seperti dibawah. Pada program ini, anda akan membuat sebuah kontur puncak suatu permukaan yang dibangkitkan oleh fungsi x bernilai dari -1 sampai+1 dengan step 0.05, dan fungsi y juga memiliki nilai yang sama.

Kemudian anda tetapkan bahwa nilai z merupakan sebuah grafik sinus sebagai fungsi nilai x. Kurang lebihnya program anda adalah seperti berikut.

```
%tiga_dim_01.m
[X,Y] = meshgrid(-1:.05:1, -1:.05:1);
Z=sin(2*pi*X);
surf(X,Y,Z)
```



Gambar 15. Tampilan Grafik 3 Dimensi

Untuk mengetahui bagaimana perintah ini bekerja coba anda lakukan perubahan pada `Z=sin(2*pi*X);` menjadi `Z=sin(2*pi*Y);` dan perhatikan apa yang terjadi. Jika anda masih penasaran, lanjutkan dengan menetapkan z sebagai fungsi x dan y sekaligus, `Z=sin(2*pi*X) + sin(2*pi*Y);`
Amati bentuk yang dihasilkan, seharusnya anda menjadi lebih paham setelah langkah ini.

4.6. Operasi File

Save

Sintak **save('namafile')** akan menyimpan semua variable workspace di dalam suatu format biner di dalam directory kerja anda sekarang dengan format file **namafile.mat**. MAT-file memiliki double-precision, dan binary. Untuk lebih mudah dalam memahami cara menyimpan file data dengan Matlab, mari kita coba program berikut ini. Disini kita akan melakukan penyimpanan sebuah matrik yang tersusun dari 4x4, dan kita simpan dalam format ASCII.

```
a = magic(4);
b = ones(2, 4) * -5.7;
c = [8 6 4 2];
save -ascii mydata.dat
```

Load

Setelah menyimpan file data yang anda buat dengan MAtlab, tentu anda akan mencoba memeriksa apakah data tersebut sudah tersimpan dengan baik. Anda mungkin saj bias membuka data secara langsung dengan menggunakan text editor atau windows explorer. Agar langkah anda lebih mantab, akan lebih baik jika anda melakukannya dengan Matlab. Sintak **load** akan memanggil semua variable dari MAT-file matlab.mat, jika file yang dipanggil ada, makan akan ditampilkan, sedangkan jika filenya tidak ada akan muncul pesan kesalahan. Sintak **load filename** akan memanggil semua variable dari fiel yang namanya sesuai dengan **filename** terebut. Akan lebih mudah jika dibelakang filename diberi tipe ekstension filenya, misalnya **filename.dat.** Berikut ini adalah contoh pemanggilan file dengan Matlab.

```
clear
load mydata.dat
```

Coba anda jalankan program ini, maka MATLAB akan memanggil semua data dari file ASCII, yang bernama **mydata.dat**. Selanjutnya data ditampilkan seperti berikut ini

```
mydata =

   16.0000    2.0000    3.0000   13.0000
    5.0000   11.0000   10.0000    8.0000
    9.0000    7.0000    6.0000   12.0000
    4.0000   14.0000   15.0000    1.0000
   -5.7000   -5.7000   -5.7000   -5.7000
   -5.7000   -5.7000   -5.7000   -5.7000
    8.0000    6.0000    4.0000    2.0000
```

Apakah anda penasaran dengan operasi file ini? Cobalah untuk melakukan operasi file yang bertipe *.txt. Anda bisa melakukannya untuk pengolahan data, dsb.

4.7. Matlab untuk File Audio

Operasi file audio sebetulnya merupakan kelanjutan dari operasi file, tetapi karena sifatnya lebih khusus (format, pengkodean, sampling rate, dsb) maka penanganan file audio kita bicarakan secara terpisah disini.

Pembangkitan Sinyal Suara

Kita mulai sebuah pembangkitan sinyal audio dengan cara menggunakan sinyal sinusoida yang memiliki frekuensi beragam. Sinyal selanjutnya kita amati melalui perangkat audio. Anda tidak usah kawatir dalam urusan interfacing, sebab system audio pada computer anda tidak bermasalah, secara otomatis Matlab akan mengaktifkan file audio sesuai program anda.

```
%File Name:lamp_02.m
fs=8000;
dt=1/fs;
dur=2.8;
t=0:dt:dur;
psi=2*pi*(100 + 200*t + 500*t.*t);
xx= 7.7*sin(psi);
sound(xx,fs);
```

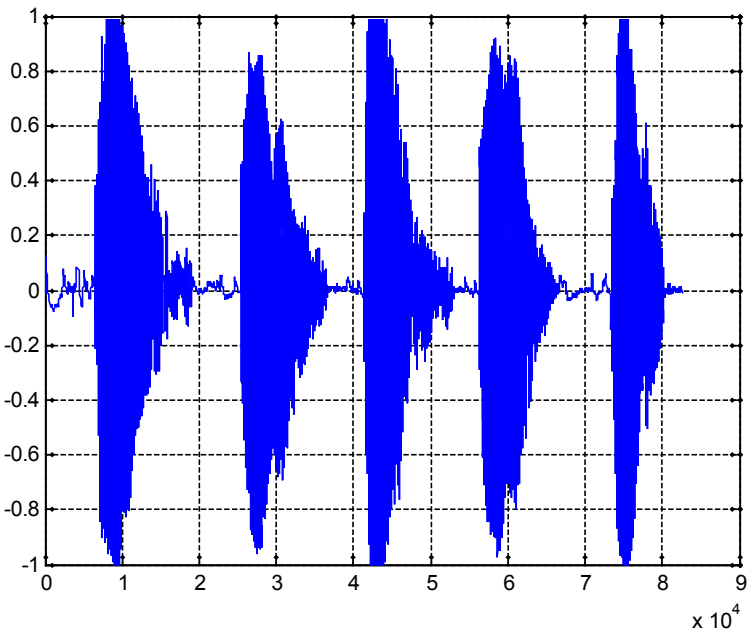
Jalankan program ini, anda akan mendapatkan sebuah suara sirine…….

Memanggil File *.wav

Langkah yang kedua disini adalah bagaimana cara anda memanggil sebuah file *.wav dan mencobanya dengan Program Matlab. Untuk itu anda harus memahami cara menentukan frekuensi sampling. Ketika anda memberikan perintah `[y, fs] = wavread('aiueo.wav');`, maka Matlab akan memanggil file 'aiueo.wav' dengan frekuensi sampling sebesar 22050, dan hasilnya disimpan di dalam variable y. Anda bisa merubah frekuensi samplignya dengan perintah `Y=resample(y,10000,fs);`, langkah ini akan merubah frekuensi sampling dari fs(22050) menjadi 10000. Untuk lebih mudahnya, anda coba program dibawah ini.

```
%File Name: lamp_05.m
clear all;
[y, fs] = wavread('file_aiueo.wav'); %read in the wav file
Y=resample(y,10000,fs);
sound(y,fs) %play back the wav file
tt=length(y);
t=1:tt;
plot(t,y) %plot the original waveform
grid
```

Ketika anda menjalankan program diatas, akan muncul ucapan a-i-u-e-o, dan diikuti dengan tampilan grafik seperti pada Gambar 16. Akan lebih baik jika anda mencoba dengan file audio yang lain, dan catat setiap kejadian yang anda anggap menarik.

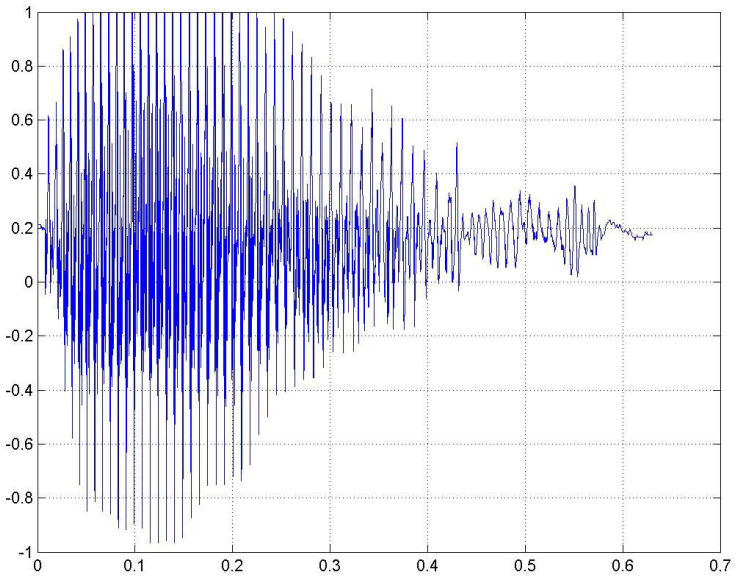


Gambar 16. Tampilan File aiueo.wav

Merekam dan Membuat File *.wav

Pada bagian ini kita akan melakukan proses perekaman dan penyimpanan file audio yang sudah diperoleh. Untuk langkah ini kita bisa memanfaatkan **wavrecord** dan **wavwrite** yang sudah disediakan oleh Library Matlab. Kita coba dengan program berikut ini.

```
%File Name: lamp_06.m
%Oleh: Tri Budi Santoso
%Lab Pengolah Sinyal, EEPIS-ITS
%WAVRECORD(N,FS,CH) me-record N sampel audio pada frekuensi FS Hertz
%dari CH channel input yang disediakan Windows WAVE audio device.
%Standar audio rate adalah 8000, 11025, 22050, dan 44100 Hz.
%Sample-sampel dikembalikan dalam suatu matrik dengan ukuran N x CH.
%Jika tidak ditetapkan maka, secara default FS=11025 Hz, dan CH=1.

clear all;
fs = 8000;
y  = wavrecord(0.8*fs, fs, 'double');
wavplay(y,fs);
wavwrite(y,fs,'aaa.wav');
t=1:length(y);
plot(t/fs,y)
grid on
title('Hasil Perekaman Suara')
ylabel('Nilai')
xlabel('waktu (detik)')
```



Jalankan program ini, anda akan mendapatkan suara hasil rekaman suara ‘aaa’, dan sebuah tampilan gambar berikut ini.

Gambar 17. Hasil Perekaman Ditampilkan Dalam Bentuk Gambar

4.8. Discrete Fourier Transform

Kebiasaan kita dalam pengamatan domain frekuensi adalah memanfaatkan penggunaan transformasi fourier diskrit (discrete fourier transform) yang didasari dengan pola pendekatan sinyal sinusoida. Secara umum persamaan untuk discrete fourier transform (DFT) dan inverse discrete fourier transform dinyatakan sebagai berikut.

$$X(k) = \sum_{n=0}^{N} x(n) e^{-jk\omega_0 n} \qquad 0 \le k \le N-1 \qquad (2)$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{jk\omega_0 n} \qquad 0 \le n \le N-1 \qquad (3)$$

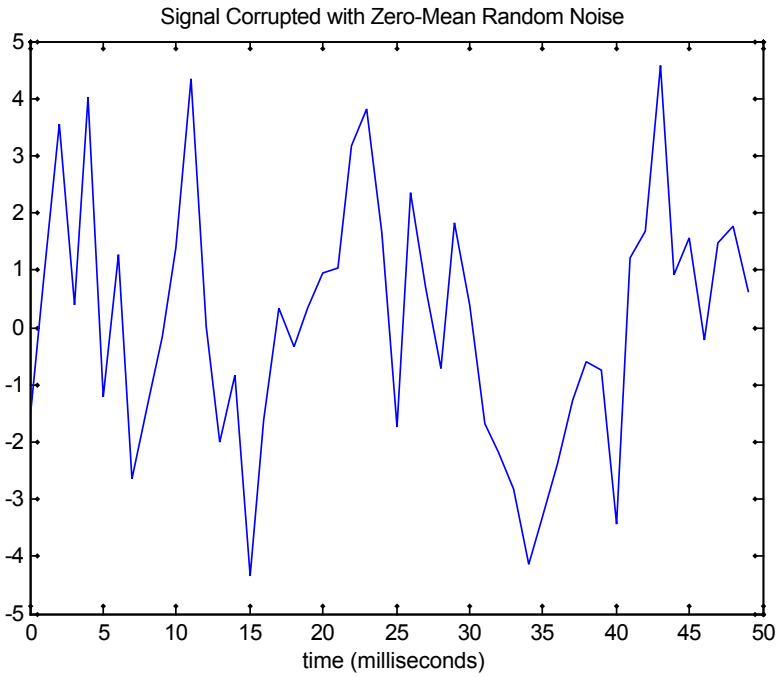
dimana

$$\omega_0 = 2\pi/N$$

Di dalam library sintaknya adalah sbb `X = fft(x)` dan `x = ifft(X)`, walaupun disini dituliskan sebagai fft, tetapi sebetulnya operasinya mengacu pada persamaan diatas. Seringkali sintak diatas dituliskan sebagai `X = fft(x,N)` dimana nilai N merupakan 2^n terdekat diatas nilai panjangnya vector x. Untuk lebih mudah memahami bagaimana operasiDFT dalam Matlab dapat digunakan untuk melakukan transformasi dari domain waktu menjadi domain frekuensi, anda bisa mencoba program berikut ini. Pada program ini anda membuat sebuah data sinyal sinus yang disampel dengan frekuensi 1000 Hz , yang mana dituliskan sebagai `t = 0:0.001:0.6;`. Selanjutnya dua sinyal disusun dengan frekuensi 50 Hz dan 120 Hz, dan keduanya digabungkan dengan opersi pernjumlahan. Sinyal diberi perlakuan dengan noise Gaussian yang memilini nilai zero-mean.

```
%File Name: dft_01.m
t = 0:0.001:0.6;
x = sin(2*pi*50*t)+sin(2*pi*120*t);
y = x + 2*randn(size(t));
plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')
```

Jalankan program diatas, dan akan diperoleh tampilan seperti pada Gambar berikut.



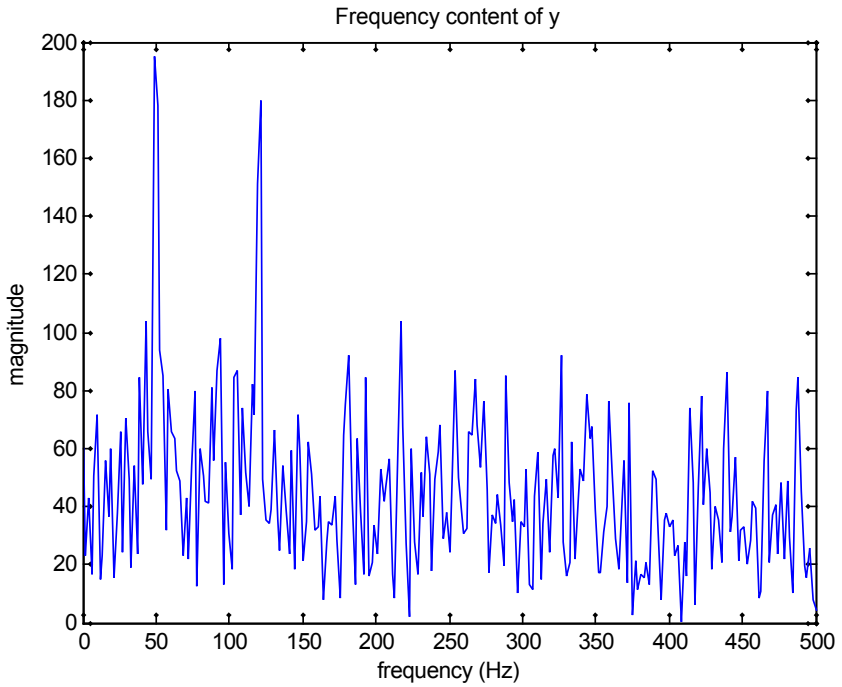
Gambar 18. Dua Sinyal Sinus plus Noise Terbangkit

Langkah anda belum selesai, untuk itu lanjutkan dengan melakukan konversi ke domain frekuensi dengan library fft. Hasil yang berupa variabel real dan imajiner dikonversi ke bentuk absolute. Untuk menampilkan dalam domain frekuensi perlau dilakukan penskalaan dengan frekuensi sampling.

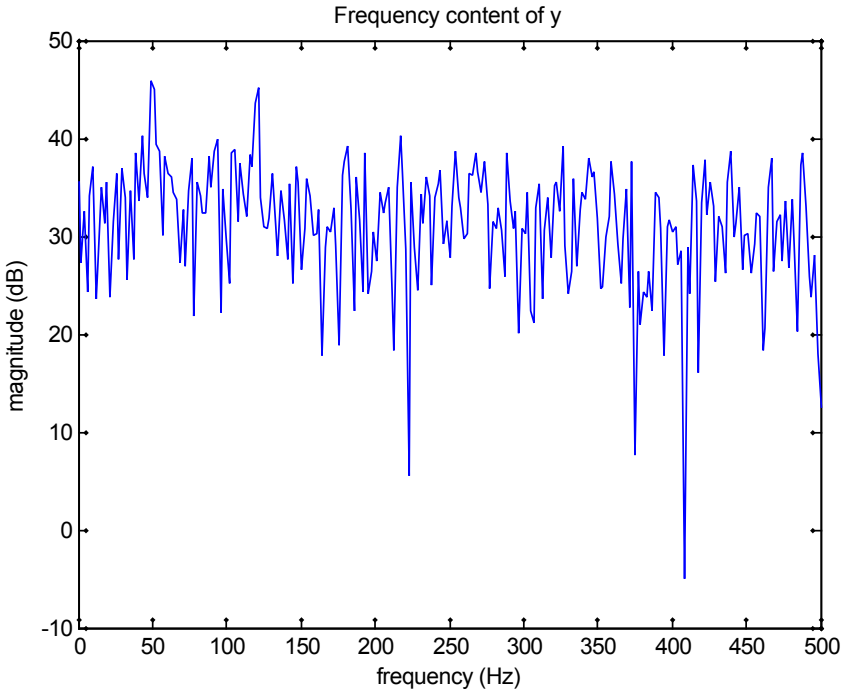
```
Y = fft(y,512);
Y_abs=abs(Y);
f = 1000*(0:256)/512;
figure(1)
plot(f,Y_abs(1:257))
title('Frequency content of y')
xlabel('frequency (Hz)')
ylabel('magnitude')
```

Tambahan untuk menyajikan dalam besaran yanglebih umum adalah sebagai berikut.

```
figure(2)
plot(f,20*log10(Y_abs(1:257)))
%plot(Y_abs)
%Pyy = Y.* conj(Y) / 512;
%f = 1000*(0:256)/512;
%plot(f,Pyy(1:257))
title('Frequency content of y')
xlabel('frequency (Hz)')
ylabel('magnitude (dB)')
```



Gambar 19. Domain Frekuensi dalam Skala Linear



Gambar 20. Domain Frekuensi dalam Skala deci Bell

4.9. Freqz

Cara lain untuk menyajikan transformasi dari domain waktu ke domain frekuensi adalah dengan menggunakan library freqz. Perintah H = FREQZ(B,A,W) akan menghasilkan sebuah respon frekuensi yang ditandai dalam vector W, denagn besaran dalam (yang ternormalisasi antara 0 dan π radiant). Library ini banyak digunakan dalam analisa kinerjua sebuah filter digital, karena fungsi freqz didasari oleh bentuk perbandingan polynomial pada filter digital.

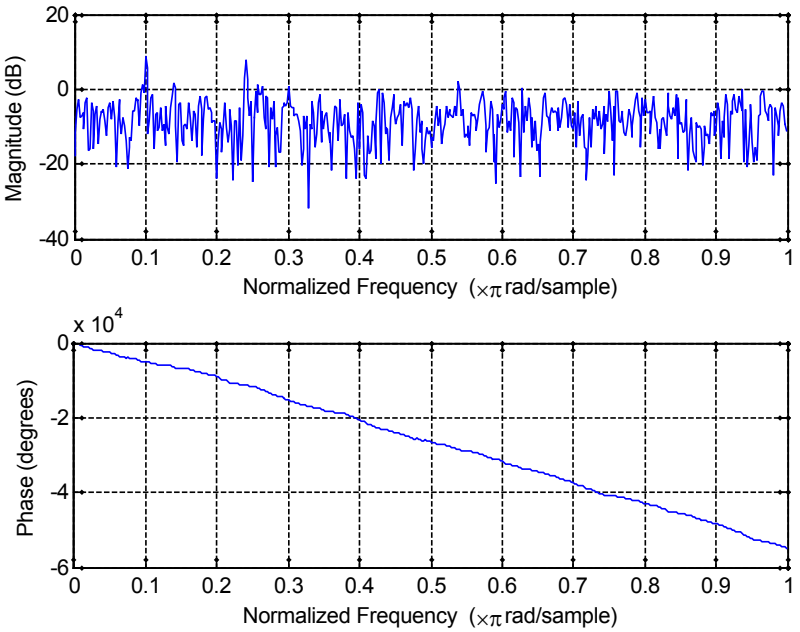
$$H(e)^{j\omega} = \frac{B(e)^{j\omega}}{A(e)^{j\omega}} = \frac{b(1) + b(2)e^{-j\omega} + \cdots + b(m+1)e^{-jm\omega}}{a(1) + a(2)e^{-j\omega} + \cdots + b(n+1)e^{-jn\omega}} \quad (4)$$

$$H(e)^{j\omega} = \frac{B(e)^{j\omega}}{A(e)^{j\omega}} = \frac{b(1) + b(2)e^{-j\omega} + \cdots + b(m+1)e^{-jm\omega}}{a(1) + a(2)e^{-j\omega} + \cdots + b(n+1)e^{-jn\omega}} \quad (4)$$

Jika anda belum juga paham maksud penjelasan diatas, sebaiknya kita coba program berikut ini, yang mana dengan contoh persoalan yang sama dengan kasus sebelumnya tetapi kita menggunakan fungsi freqz untuk mengamati bentuk sinyal dalam domain frekuensi.

```
%file name: freqz_01.m
t = 0:0.001:0.6;
x = sin(2*pi*50*t)+sin(2*pi*120*t);
y = x + 2*randn(size(t));
plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')
freqz(y,100)
```

Jalankan program ini dan andadapatkan tampilan seperti gambar dibawah ini.



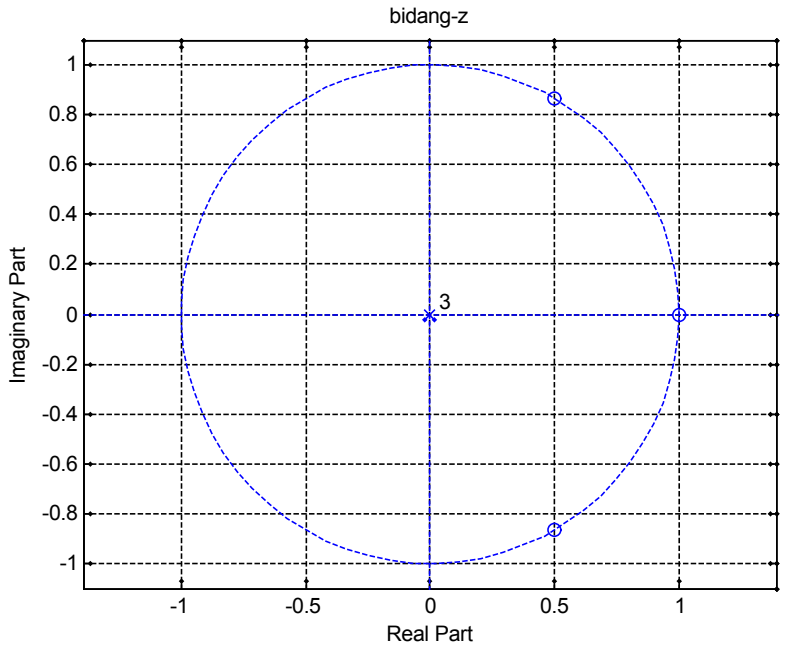
Gambar 21. Domain Frekuensi hasil transformasi dengan freqz

4.10. Penggambaran Bidang -Z

Satu hal yang cukup penting dalam permasalahan pengolahan sinyal digital adalah transformasi-z, dan didalam kasus ini anda harus berurusan dengan penggambaran dalam domain z atau yang lebihdikenal sebagai **unit cirle z-plane.** Fungsi yang disediakan oleh Library Matlab adalah digunakan untuk menampilkan nilai **zero** dan **pole** pada suatu filter digital. Sintak zplane(Hq) akan menggambarkan **zero** dan **pole** dari filter Hq terkuantisasi. Simbol 'o' merepresentasikan suatu **zero** pada filter, dan simbol 'x' merepresentasikan suatu **pole** pada filter tersebut. Untuk lebih mudahnya anda coba sebuah program sederhana berikut ini.

```
B=[1;exp(j*pi/3);exp(-j*pi/3)];
A=[0;0;0];
zplane(B,A)
grid
title('bidang-z')
```

Pada program tersebuta, kita memiliki nilai nilai zero pada yang dituliskan pada variable B, dan nilai pole yang dituliskan pada variable A. Jika anda menjalankan program diatas, akan diperoleh tampilan seperti pada Gambar 22 berikut ini.



Gambar 21. Penggambaran pole dan zero pada Bidang z

V. **TUGAS**

Anda telah melakukan percobaan dengan memanfaatkan perangkat lunak Matlab yang berkaitan dengan Pengolahan Sinyal Digital. Dari apa yang sudah anda lakukan buat catatan-catatan yang anda anggap penting dan anda harus menemukan persoalan baru dalam penggunaan Matlab yang belum disinggung dalam percobaan ini.

MODUL III

SIMPLE IO UNTUK TMS320C6713

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini, yang anda peroleh adalah:

- Siswa mampu memahami karakterisktik dan cara kerja codec input-output AIC23 pada TMS320C6713
- Siswa mampu membangun suatu program sederhana menggunakan Code Composer untuk proses pengolahan IO sederhana

II. INPUT DAN OUPUT PADA DSK

Umumnya aplikasi menggunakanteknik-teknik DSP memerlukan paling tidak sebuah sistem dasar yang ditunjukkan pada Gambar 1, yang tersusun dari analog input, dan analog output. Pada bagian jalur input ada sebuah anti aliasing filter untuk mengeliminasi frekuensi diatas Nyquist Frequency, yang didefinisikan senilai ½ dari frekuensi smpling F_s. Jika ini tidak terpenuhi, akan tejadi aliaasing, yang mana suatu sinyal dengan frekuensi lebih tinggi dari ½ F_s akan disamarkan sebagai sebuah frekuesi lebih rendah dari ½ F_s. Teorema sampling menjelaskan bahwa frekuensi sampling harus lebih besar atau sama dengan dua kali frekuensi tertinggi penyusun sinyal input (f_{max}), sehingga:

$$F_s \geq f_{max} \rightarrow F_s \geq 2f \qquad (1)$$

$$F_s \geq f_{max} \rightarrow F_s \geq 2f \qquad (1)$$

yang mana dikatikan dengan periode,

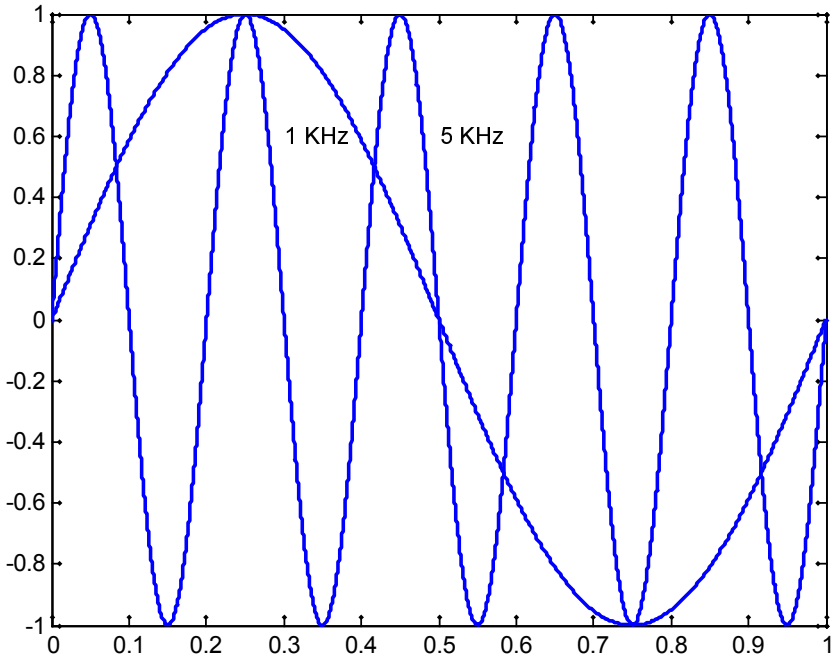
$$(1/T_s) > 2(T_s) \leftrightarrow T > 2T_s$$
$$\leftrightarrow T_s < T/2$$

dimana Ts adalah periode sampling.

Gambar 1. Diagram blok system input/output pada DSP

Periode samplig Ts, harus lbih kecil dari ½ period sinyal input. Sebagai contoh, jika kita

anggap bahwa telinga tidak mampu mendeteksi frekuensi lebih tinggi dari 20 Khz, kita dapat menggunkn low-pass-filter dengan bandwidth atau frekuensi cut-off pada 20KHz untuk menghindari aliasing. Kita dapat melakukan pengambilan sample musik pada Fs>40KHz (biasanya 44,1 atau 48 KHz) dan menghilankan komponen-komponen frekuensi lebih tinggi dari 20 KHz. Gambar 2. mengilustrasikan sebuah sinyal ter-alias.



Gambar 2. Sinyal 5 KHz muncul sebagai aliasing sinyal 1 KHz.

Kita tetapkan frekuensi sampling Fs = 4 KHz, atau periode sampling senilai Ts = 0.25 ms. Suatu sinyal dengan frekuensi 5 KHz jika disample dengan Fs = 4KHz akan muncul sebagai sinyal 1Khz, sehingga sinyal 1 KHz dalam kondisi ter-alias. Juga jika kita paksakan sinyal dengan frekuensi 3 KHz dan 9 KHz keduanya akan muncul sebagai sinyal 1 Khz.

2.1. TLV320AIC23 (AIC23) Onboard Codec untuk Input dan Output

Pada DSK board terdapat TLV320AIC23 (AIC23) codec untuk input dan output. Rangkaian ADC pada codec mengkonversi sinyal input analog menjadi representasi sinyal digital untuk bisa diproses oleh DSP. Level maksimum sinyal input analog untuk bisa dikonversi ditentukan oleh spesifikasi rangkaian ADC pada codec, yang dalam hal ini adalah 6 Volt p-p. Setelah sinyal yang di-capture (ditangkap) diproses, hasilnya akan dikirimkan kembali ke dunia luar. Pada lintasan output Gambar 2.1. ada sebuah ADC, yang beroperasi kebalikan dari ADC. Suatu filter menghaluskan output dan melakukan rekonstruksi sinyal output. ADC, DAC dan pemfilterannya adalah fungsi-fungsi yang dibentuk oleh single-chip codec AIC23 pada board DSK.

AIC23 adalah suatu codec audio stereo yang bekerja di dasarkan pada teknologi sigma-delta. Diagram blok fungsionalnya ada pada Gambar 2.3. Codec ini membentuk semua fungsional diagram blok AIC23 seperti ADC, DAC, low-pass filter, oversampling, dsb. AIC23 memiliki spesifikasi transfer data word dengan panjang 16, 20, 24, dan 32 bit.

Konverter sigma-delta dapat mencapai resolusi tinggi dengan teknik oversampling ratio tetapi dengan resiko data rate menjadi lebih rendah. Sampling rate senilai 8, 16, 24, 32, 44.1, 48 dan 96 kHz sudah disupport dan siap untuk digunakan dalam bentuk program.

Filter digital interpolasi menghasilkan oversampling. Daya noise quantization pada sejumlah device bersifat independent terhadap sampling rate. Suatu modulator dilibatkan untuk membentuk noise sehingga membentangkan ***range of interest***. Spectrum noise didistribusikan antara 0 s/d (Fs/2), sedemikian hingga hanya suatu noise kecil yang muncul pada sinyal dalam frekuensi band tertentu. Sehingga noise actual pada suatu ***band of interest*** powernya akan menjadi lebih rendah. Suatu filter digital juga dilibatkan untuk menghilangkan ***out-of-band noise***.

Suatu crystal 12-MHz mensupplay clock untuk codec AIC23 (sebagaimana untuk DSP dan USB interface). Penggunaan master clock 12 MHz, dengan oversampling rate 250Fs, dan 272Fs, suatu sample rate audio tepatnya pada 48 KHz (12 MHz / 250) dan suatu CD rate pada 44.1 kHz (12 MHz / 272) dapat ditetapkan. Sampling rate di-set dengan register codec SAMPLE RATE.

ADC mengkonversi suatu sinyal input menjadi output discrete digital word dalam suatu format complement-2 yang berkaitan dengan nilai sinyal analog yang direpresentasikannnya. DAC melibatkan suatu filter interpolasi dan suatu modulator digital. Suatu filter desimasi meredukasi data rate digital menjadi senilai sampling rate. Output DAC pertama kali dilewatkan melalui sebuah internal ***low-pass reconstruction filter*** untuk menghasilkan suatu output analog. Kinerja ***low-noise*** untuk ADC dan DAC dicapai menggunakan teknik oversampling dengan ***noise shaping*** yang disediakan oleh modulator ***sigma-delta***.

Komunikasi dengan AIC23 code untuk input dan output menggunakan dua ***multichannel buffered*** serial port MCBSPs pada C6713. McBSP0 digunakan sebagai suatu ***unidirectional channel*** untuk mengirim dan menerima data audio.

Sebagai alternative ***I/O daughter card*** dapat digunakan sebagai input dan output. Sejumlah card dapat ditancapkan (plug) ke DSK melalui eksternal peripheral interface konektor 80-pin J3 pada DSK board.

III.PERALATAN PERCOBAAN

- 1 set PC yang dilengkapi dengan software Code Composer Studio.
- 1 set DSK TMS320C5402
- 1 set Speaker Active
- Function Generator
- Oscilloscope

IV.PERCOBAAN

4.1. Penataan Peralatan Percobaan

Lakukan penataan peralatan percobaan seperti pada Gambar 3 berikut ini. Dalam hal ini posisi DSP board dihubungkan dengan PC menggunakan kabel USB. Line In DSP Board dihubungkan dengan sebuah Function Generator, dan ***Line Out*** DSp Board dihubungkan dengan speaker active atau oscilloscope. Anda bisa juga melakukannya secara parallel untuk oscilloscope dan speaker active, asal tidak terjadi gangguan pada sinyal outputnya.

Gambar 3. Penataan peralatan percobaan

4.2. Loop Menggunakan Interupt (loop_intr)

1. Buat sebuah project baru dengan CCStudio dengan nama **loop_intr.pjt.**
2. Buat sebuah source program baru dengan cara click pada File→New→Source File, beri nama **loop_intr.c**. Program ini merupakan langkah dasar untuk mengakses AIC23 dengan interrupt driven menggunakan INT11.

```
//Loop_intr.c Loop program using interrupt. Output = delayed input
#include "dsk6713_aic23.h"            //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;    //set sampling rate
interrupt void c_int11()        //interrupt service routine
{
   short sample_data;
   sample_data = input_sample(); //input data
   output_sample(sample_data);   //output data
   return;
}
void main()
{
  comm_intr();                 //init DSK, codec, McBSP
  while(1);                  //infinite loop
}
```

Setelah inisialisasi dan selection/enabling pada suatu interrupt, eksekusi menunggu sampai tak hingga sampai while loop terjadi suatu interrupt. Ketika interrupt terjadi, proses eksekusi ke ISR c_intr11, seperti yang telah dispesifikasikan di dalam vectors_intr.asm. Sebuah interrupt terjadi pada setiap periode sampel Ts = 1/Fs =1/(8 KHz) = 0.125 ms, pada suatu waktu yang mana nilai sample input dibaca dari codec's ADC dan selanjutnya mengirimkan outputnya ke codec's DAC.

3. Tambahkan file-file berikut ini ke project yang baru anda susun:
   - C6713dskinit.c
   - dsk6713_aic23.h
   - Vectors_intr.asm
   - C6713dsk.cmd
   - rts6700.lib, dsk6713bsl.lib, csl6713.lib

4. Lakukan scan all file dependencies, untuk mengetahui file-file apa saja yang berkaitan dengan program yang sedang anda susun.
5. Lakukan pengesetan build option, dengan langkah seperti pada percobaan sebelumnya.
   Pada **Compiler** → Category Basic→Target Version:C670x
   Category Files→Obj Directory: C:\CCStudio\MyProjetcs\loop_intr\Debug, tentu saja dalam hal ini akan melakukan perubahan lokasi Obj Directory sesuai dengan folder kerja yang sudah anda buat sendiri.
   Category Preprocessors→Include Search Path: C:\CCStudio\c6000\dsk6713\include
   Category Preprocessors→Predefine Symbol:CHIP_6713
   Pada **Linker**→Caregory Basic→ beri tanda tentang pada check box di depan Suppress Banner dan Exhaustively Read Libraries.
   Caregory Basic→Output Filename: .\Debug\loop_intr.out

Caregory Basic→Map Filename: .\Debug\loop_intr.map

6. Lakukan kompilasi program, build dan Rebuild All
7. Load output program, dan lanjutkan dengan running program anda
8. Perhatikan apa yang terjadi pada speaker active anda. Lakukan perubahan nilai pada function generator (bisa frekuensinya anda naik turunkan atau amplitude sinyalnya yang anda rubah,bisa juga jenis sinyal yang terbangkit dari function generator yang anda rubah), perhatikan suara yang dihasilkan pada speaker active.
9. Coba gantikan speaker active dengan sebuah probe oscilloscope, dan perhatikan sinyal yang muncul pada oscilloscope anda. Dalam hal ini anda bisa melakukan secara parallel antara oscilloscope dengan speaker active.

4.3. Loop Menggunakan Polling

1. Buat sebuah project baru dengan CCStudio dengan nama **loop_pool.pjt.**
2. Buat sebuah source program baru dengan cara click pada File→New→ Source File, beri nama **loop_pool.c**. Program ini merupakan langkah dasar untuk penggambaran input dan output pada suatu nilai sample pada setiap periode Ts. Teknik pooling menggunakan sebuah prosedur kontinyu pada pengujian ketika data dalam kondisi ready.

```
//loop_poll.c Loop program using polling. Output=delayed input
#include "DSK6713_AIC23.h"          //codec-DSK file support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
void main()
{
  short sample_data;
  comm_poll();                      //init DSK, codec, McBSP
  while(1)                                 //infinite loop
  {
    sample_data = input_sample(); //input sample
    output_sample(sample_data);   //output sample
  }
}
```

3. Tambahkan file-file berikut ini ke project yang baru anda susun:
   - C6713dskinit.c
   - dsk6713_aic23.h
   - cectors_poll.asm
   - C6713dsk.cmd
   - rts6700.lib, dsk6713bsl.lib, csl6713.lib

4. Lakukan Scan All Files Dependencies, untuk mengetahui file-file apa saja yang berkaitan dengan program yang sedang anda susun.
5. Lakukan pengesetan build option, dengan langkah seperti pada percobaan sebelumnya.

Pada **Compiler** → Category Basic→Target Version:C670x
Category Files→Obj Directory: C:\CCStudio\MyProjetcs\loop_poll\Debug, tentu saja dalam hal ini akan melakukan perubahan lokasi Obj Directory sesuai dengan folder kerja yang sudah anda buat sendiri.
Category Preprocessors→Include Search Path: C:\CCStudio\c6000\dsk6713\include
Category Preprocessors→Predefine Symbol:CHIP_6713
Pada **Linker**→Caregory Basic→ beri tanda tentang pada check box di depan Suppress Banner dan Exhaustively Read Libraries.
Caregory Basic→Output Filename: .\Debug\loop_poll.out
Caregory Basic→Map Filename: .\Debug\loop_poll.map

6. Lakukan kompilasi program, increment build dan Rebuild All
7. Load output program, dan lanjutkan dengan running program anda
8. Perhatikan apa yang terjadi pada speaker active anda. Lakukan perubahan nilai pada function generator (bisa frekuensinya anda naik turunkan atau amplitude sinyalnya yang anda rubah, bisa juga jenis sinyal yang terbangkit dari function generator yang anda rubah), perhatikan suara yang dihasilkan pada speaker active.

4.4. Pembangkitan Sinyal Sinus dengan Slider Amplitudo dan Frekuensi

1. Buat sebuah project baru dengan CCStudio dengan nama **sine2sliders.pjt.**
2. Buat sebuah source program baru dengan cara click pada File→New→ Source File, beri nama **sine2sliders.c**.

```
//Sine2sliders.c Sine generation with different # of points
#include "DSK6713_AIC23.h"          //codec-DSK interface support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;    //set sampling rate

short loop = 0;
short sine_table[32]={0,195,383,556,707,831,924,981,1000,
                                          981,924,831,707,556,383,195,
                                          0,-195,-383,-556,-707,-831,-924,-981,-1000,
                -981,-924,-831,-707,-556,-383,-195};//sine data
short gain = 1;                            //for slider
short frequency = 2;                        //for slider

void main()
{
 comm_poll();                        //init DSK,codec,McBSP
 while(1)                          //infinite loop
  {
   output_sample(sine_table[loop]*gain);//output scaled value
   loop += frequency;                 //incr frequency index
   loop = loop % 32;                  //modulo 32 to reinit index
  }
}
```

Program berbasis polling sine2sliders yang disajikan pada listing program di atas menghasilkan pembangkitkan sebuah gelombang sinus. Dua slider digunakan untuk memberi variasi nilai amplitude (gain) dan frekuensi. Pada gelombang sinusoida yang dibangkitkannya. Penggunaan lookup table 32 point, variable frekuensi ditentukan dengan pemilihan suatu angka berbeda pada point-point per cycle. Gain slider menskalakan volume/amplitude pada gelombang.

Nilai-nilai data sinus sebanyak 32 titik di dalam table atau buffer berkaitan dengan sin(t),

dimana t = 0, 11.25, 22.5,…348.75 derajat(terskala 1000). Slider frekuensi menempatkan nilai dari 2 s/d 8 dengan kenaikan 2. Operator modulo-2 digunakan untuk menguji ketika akhir daru buffer yang berisi data nilai-nilai sinus dicapai. Ketika loop index mencapai nilai 32, akan diinisialisasikan ulang ke nol. Sebagai contoh, dengan frekuensi slider pada posisi 2, loop atau frekuensi index melangkah melalui nilai-nilai lain di dalam table. Hal ini berkaitan dengan nilai-nilai data di dalam 1 siklus.

3. Tambahkan file-file berikut ini ke project yang baru anda susun:
   - C6713dskinit.c
   - dsk6713_aic23.h
   - cectors_poll.asm
   - C6713dsk.cmd
   - rts6700.lib, dsk6713bsl.lib, csl6713.lib

4. Lakukan Scan All Files Dependencies, untuk mengetahui file-file apa saja yang berkaitan dengan program yang sedang anda susun.
5. Lakukan pengesetan build option, dengan langkah seperti pada percobaan sebelumnya.
   Pada **Compiler** → Category Basic→Target Version:C670x
   Category Files→Obj Directory: C:\CCStudio\MyProjetcs\sine2sliders\Debug, tentu saja dalam hal ini akan melakukan perubahan lokasi Obj Directory sesuai dengan folder kerja yang sudah anda buat sendiri.
   Category Preprocessors→Include Search Path: C:\CCStudio\c6000\dsk6713\include
   Category Preprocessors→Predefine Symbol:CHIP_6713
   Pada **Linker**→Caregory Basic→ beri tanda tentang pada check box di depan Suppress Banner dan Exhaustively Read Libraries.
   Caregory Basic→Output Filename: .\Debug\ sine2sliders.out
   Caregory Basic→Map Filename: .\Debug\ sine2sliders.map

6. Lakukan kompilasi program, increment build dan Rebuild All.
7. Load output program, dan lanjutkan dengan running program anda.
8. Cari file “sine2sliders.gel” dengan menggunakan file search Windows, dan catat folder dimana posisi file ini berada.
9. Pilih File→ Load GEL, dapatkan file “sine2sliders.gel”
10. Pilih GEL→ Sine Parameter → Gain. Ulangi untuk Sine Parameter→ Frequency
11. Running program anda, dan ubah-ubah posisi anak panah pada Slider Gain, atau Slider Frequency, catat pengaruhnya pada sinyal output yang dihasilkan.
12. Ferifikasi bahwa frekuensi sinyal output yang dihasilkan adalah senilai f=Fs/16 = 500 Hz. Untuk ini anda harus melakukan pengukuran dengan Oscilloscope. Jika anda belum paham caranya, anda tanyakan kepada asisten praktikum atau dosen pengampu. Penggunaan oscilloscope untuk mengamati nilai frekuensi memang tidak terlalu bagus, sebab ada perangkat lain yang lebih teliti untuk tugas ini, yaitu Frequency Counter. Tetapi dalam hal ini anda harus bisa memprediksi nilai frekuensi dengan keterbatasan peralatan yang ada, yaitu Oscilloscope.
13. Naikkan posisi slider frekuensi ke 4, 6, 8, dan ferifikasi apakah sinyal frekuensi terbangkit juga mengalami perubahan. Tetap anda gunakan Oscilloscope untuk pengamatan nilai frekuensi.

V. TUGAS

1. Dari gambar yang anda peroleh melalui pengukuran menggunakan Oscilloscope, jelaskan bagaimana anda bisa memperhitungkan nilai frekuensi dari sinyal yang muncul dari praktikum anda. Jika pada Oscilloscope yang anda gunakan kebetulan telah disertai dengan perangkat frekuensi counter, anda harus tetap menjelaskan perhitungan secara manual dari gambar yang anda peroleh.

2. Lakukan perubahan pada nilai amplitude sinyal yang anda bangkitkan, amati dan catat apakah sinyal terbangkit sudah linear dengan kenaikan yang anda tetapkan pada gain slider.

3. Anda harus melakukan perubahan pada nilai frekuensi sampling yang anda gunakan pada praktikum ini, untuk itu syntak bagian mana yang harus anda rubah?

4. Jika frkeuensi sampling yang anda gunakan dirubah menjadi 16 KHz, sementara pembangkitan sinyal sinyal dilakukan dengan cara yang sama, berapa frekuensi output dari sinyal yang dihasilkan? Jelaskan perhitungannya.

MODUL IV

PEMBANGKITAN SINYAL DENGAN TMS320C6713

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini, yang anda peroleh adalah:

- Siswa mampu membangun sebuah program untuk pembangkitan berbagai bentuk sinyal dengan memanfaatkan TMS320C6713 DSK

II. INPUT DAN OUPUT PADA DSK

Didalam laboratorium, anda pernah bekerja menggunakan *function generator*, alat yang dapat menghasilkan beberapa bentuk sinyal seperti sinyal sinusoida, sinyal persegi dan sinyal segitiga. Secara analog, anda dapat membuat rangkaiannya menggunakan rangkaian op-amp.

Pada praktikum ini anda akan mencoba membuat function generator secara digital. Yang anda butuhkan adalah sebuah sistem prosesor dan sebuah DAC. Sistem prosesor yang digunakan bisa berupa mikroprosesor atau mikrokontroler. Tetapi untuk praktikum, tentu saja anda akan menggunakan DSP. Lalu bagaimana cara membuat sinyal-sinyal yang diinginkan? Cukup dengan memprogram prosesor tersebut untuk menghasilkan nilai-nilai tertentu yang nantinya akan diterjemahkan oleh DAC menjadi sebuah sinyal analog.

Untuk menghasilkan bentuk sinyal yang bermacam-macam, anda tidak perlu merombak atau menambahkan komponen (*hardware)*. Cukup melakukan modifikasi pada program yang anda buat. Bila program diubah maka bentuk sinyal berubah. *Programmable*, itulah salah satu kelebihan bila anda bekerja didunia digital.

2.1. Sinyal-Sinyal Periodik

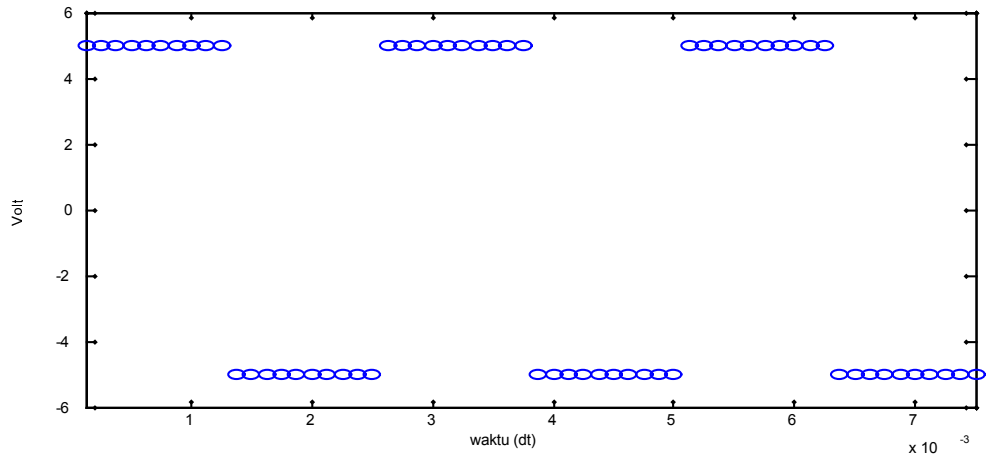
Sinyal persegi

Sinyal persegi mempunyai duty-cycle 50% dan terdiri dari dua level tegangan. Level tegangan bawah (V_L) dan level tegangan atas (V_H). Perhatikan gambar 1. Lingkaran-lingkaran biru menunjukkan nilai-nilai level tegangan yang diberikan pada kecepatan sampling yang telah

ditentukan. Algoritma untuk menghasilkan sinyal persegi adalah:

1. Isi variabel dengan nilai V_L
2. Keluarkan isi variabel ke DAC sebanyak N kali
3. Isi variabel dengan nilai V_H
4. Keluarkan isi variabel ke DAC sebanyak N kali
5. Ulangi langkah 1-4

Level tegangan V_L dan V_H yang diberikan menentukan amplitudo sinyal persegi. Frekuensi sampling yang digunakan oleh DAC dan pengulangan sebanyak N kali akan menentukan frekuensi dari sinyal persegi yang dibuat.



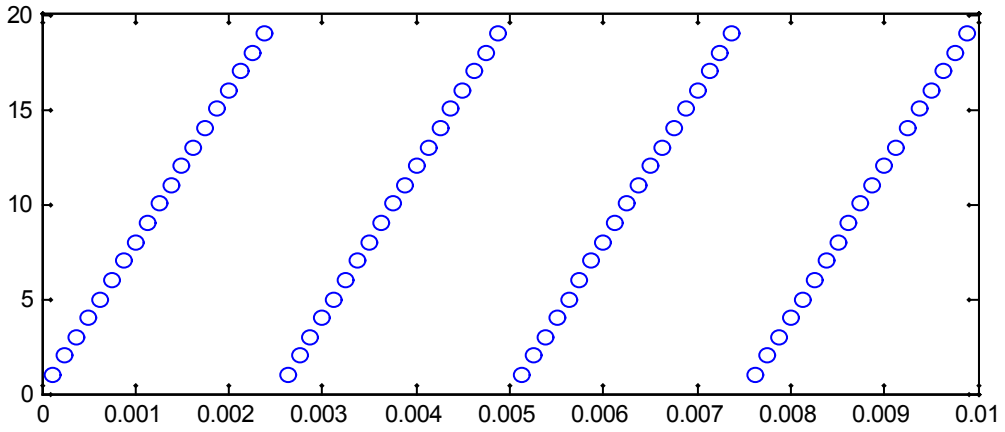
Gambar 1. Sinyal persegi dengan duty-cycle 50%

Dari algorithma diatas didapatkan bahwa untuk membentuk satu siklus sinyal persegi diperlukan jumlah sampel sebanyak 2N, atau periode sinyal T ekuivalen dengan 2N. Jika kita menggunakan sampling rate 8000 sample per detik, dan N bernilai 10, maka dalam satu detik akan terjadi jumlah siklus sinyal sebanyak SR/T = 8000/20 = 400. Nilai ini yang menentukan frekuensi sinyal terbangkit. Dengan cara lain disebutkan bahwa satu periode sinyal tersusun dari 20 sampel, atau ekuivalent dengan 20/8000 detik = 0.0025 detik. Maka frekuensi f = 1/T = 1/0.0025 = 400 Hz. Nilai frekuensi ini adalah nilai frekuensi fundamental, sebab sinyal persegi merupakan sinyal yang tersusun dari jumlahan sinyal dengan frekuensi fundamental dan sinyal-sinyal lain yang merupakan harmonisanya sampai dalam jumlah yang tak berhingga.

Sinyal Gigi Gergaji

Sinyal gigi-gergaji terdiri dari dua bagian yaitu, bagian linier dan titik diskontinyu. Perhatikan gambar 2. Bagian linier dapat anda hasilkan dengan cara menambahkan sebuah bilangan konstan (V_K) pada bilangan awal (V_I) yang selalu tetap. Penambahan ini berlangsung terus sampai pada batas yang anda tentukan. Bila batas ini tercapai maka hasil penjumlahan dikembalikan ke V_I. Algoritma untuk menghasilkan sinyal gigi-gergaji adalah :

1. Isi variabel dengan nilai awal V_I
2. Keluarkan isi variabel ke DAC
3. Tambahkan variabel dengan nilai konstan V_K
4. Bila nilai variabel melebihi batas yang ditentukan, kembali ke langkah nomor-1
5. Bila nilai variabel belum melebihi batas, kembali ke langkah nomor-2

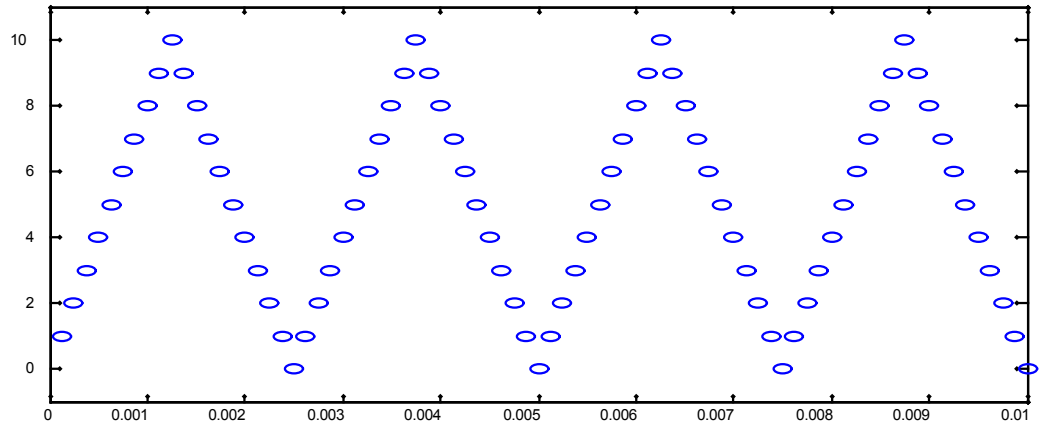


Gambar 2. Sinyal gigi-gergaji

Level tegangan awal V_I dan batas atas yang diberikan menentukan amplitudo sinyal gigi gergaji. Frekuensi sampling yang digunakan oleh DAC dan kecuraman yang ditentukan oleh penambahan dengan V_K akan menentukan frekuensi dari sinyal persegi yang dibuat.

Sinyal Segitiga

Sinyal segitiga dapat anda buat dengan cara menambahkan sebuah nilai konstan (V_K) terhadap nilai awal (V_I). Penambahan ini terus dilakukan sampai hasil penjumlahan mencapai batas (V_{MAX}) yang anda tentukan. Kemudian hasil penjumlahan dikurangi dengan V_K sampai mencapai batas V_I. Karena nilai penjumlahan dan nilai pengurangan adalah sama maka akan dihasilkan gelombang segitiga seperti yang tampak pada Gambar 3.



Gambar 3. Sinyal segitiga

Algoritma untuk menghasilkan sinyal segitiga adalah:

1. Isi variabel dengan nilai awal VI
2. Keluarkan isi variabel ke DAC
3. Tambahkan variabel dengan nilai konstan VC
4. Ulangi ke langkah nomor-2 bila nilai variabel masih dibawah VMAX
5. Keluarkan isi variabel ke DAC
6. Kurangkan variabel dengan nilai konstan VC
7. Ulangi ke langkah nomor-5 bila nilai variabel belum mencapai VI
8. Ulangi ke langkah nomor-1

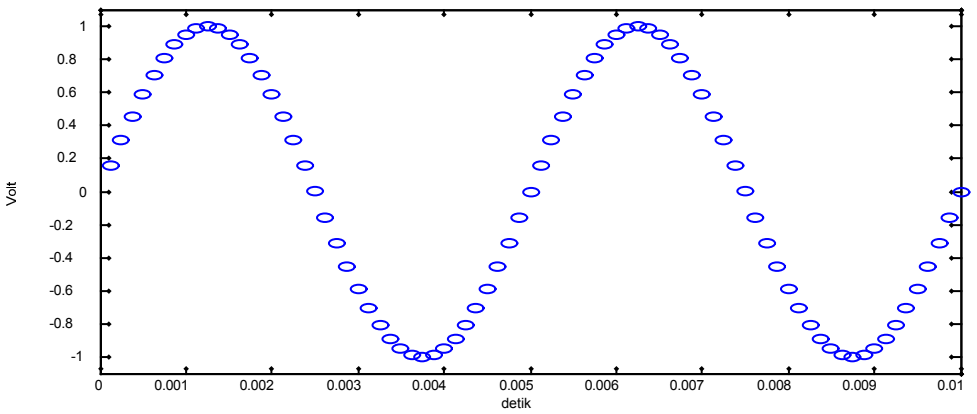
Level tegangan awal VI dan batas atas VMAX yang diberikan menentukan amplitudo sinyal segitiga. Frekuensi sampling yang digunakan oleh DAC dan kecepatan naik-turun yang ditentukan oleh penambahan dan pengurangan dengan VK akan menentukan frekuensi dari sinyal persegi yang dibuat.

Sinyal Sinus

Anda dapat menghasilkan sinyal sinusoida seperti yang ditunjukkan oleh Gambar 4 menggunakan persamaan identitas trigonometri yang menyebutkan bahwa

$$\sin(n\theta)=2\cos(\theta)*\sin\{(n-1)\theta\}-\sin\{(n-2)\theta\}.$$

Persamaan tersebut menggunakan dua langkah untuk menghasilkan sinyal sinuoida. Pertama, hitung nilai cos(θ) diatas kertas. Kedua, menghasilkan sinyal itu sendiri, menggunakan satu perkalian dan satu pengurangan berdasarkan *counter* n. Sinyal sinus yang akan anda hasilkan diasumsikan bahwa nilai dari sin{(n-1)θ} dan sin{(n-2)θ} sudah dihitung sebelumnya dan disimpan pada variabel didalam program. Untuk menghasilkan sinyal sinusoida dengan frekuensi tertentu bergantung pada nilai awal dari cos(θ).



Gambar 4. Sinyal sinusoida

Algoritma untuk menghasilkan sinyal sinusoida adalah:

1. Inisialisasi nilai n dengan nol
2. Tentukan nilai θ dengan persamaan (2*π*f)/fs, dimana nilai f menentukan frekuensi sinyal sinusoida yang akan dihasilkan dan fs adalah frekuensi sampling
3. Hitung dan simpan nilai dari 2cos(θ) pada variabel C,
4. Berdasarkan dari nilai n dan θ, simpan nilai awal dari sin{(n-1)θ} pada variabel A dan sin{(n-2)θ} pada variabel B
5. Hitunglah nilai dari persamaan sin(nθ) menggunakan nilai dari variabel A, B, dan
   C dan kirim hasilnya ke DAC
6. Simpan nilai sin{(n-1)θ} ke variabel B
7. Simpan nilai persamaan sin(nθ) yang didapat pada variabel A
8. Tambahkan n dengan 1
9. Ulangi langkah 4-7

Sinyal Pseudo Random

Pseudo random bisa direalisasikan dalam bentuk Simple Shift Register Generator (SSGR) yang menfeedback sinyal-sinyal ke suatu input pada shift register (delay line). SSGR adalah linear jika fungsi-fungsi feedbacknya dapat diekspresikan sebagai suatu operasi modulo-2 (XOR). Fungsi ini menghasilkan sebuah sekuen bilangan binary random atau yang juga dikenal sebagai PN (pseudo noise) sekueen.

Gambar 5. Model dasar Pseudo Random-Generator

Fungsi feedback f(x0,x1,…) merupakan suatu operasi jumlahan modulo-2 pada nilai x_i dan sel-sel shift register dengan c_i sebagai nilai koefisien koneksi ($c_i = 0$ = open; $c_i = 1$ = connect). Suatu SSGR dengan flip-flop sebanyak L menghasilkan sederetan nilai yang tergantung pada panjang register L, feedback tap connection dan initial conditions. Ketika periode (panjang) sekuen senilai $Nc = 2^L - 1$ terjadi, maka PN sekuen yang dihasilkan dinyatakan memiliki nilai maximum length sequence atau m-sequence.

Contoh pembentukan pseudo random atau pseudo noise random dapat dilihat pada gambar berikut ini. Dengan menyusun feedback pada b_0 dan b_1 dan outputnya sebagai feedback ke input PRG (dalam hal ini b_0), maka akan diperoleh pola acak pada output (b_3). Sebagai inisialisasi semua nilai pada shift register bernilai 1. Pola random yang dihasilkan bisa dilihat

pada Tabel 1.

Gambar 6. Pseudo random generator 4-bit

Table 1. Pola random output PRG

Waktu	Isi bit ke-i			
ke-n	b0	b1	b2	b3
0	1	1	1	1
1	0	1	1	**1**
2	1	0	1	**1**
3	0	1	0	**1**
4	0	0	1	**0**
5	1	0	0	**1**
6	1	1	0	**0**
7	1	1	1	**0**
8	0	1	1	**1**
9	1	0	1	**1**
10	0	1	0	**1**
11	0	0	1	**0**
12	1	0	0	**1**
13	1	1	0	**0**
14	1	1	1	**0**
15	0	1	1	**1**
16	1	0	1	**1**
17	0	1	0	**1**
18	0	0	1	**0**
19	1	0	0	**1**
20	1	1	0	**0**
21	1	1	1	**0**

III. PERALATAN PERCOBAAN

- 1 set PC yang dilengkapi dengan software Code Composer Studio.
- 1 set DSK TMS320C5402
- 1 set Speaker Active
- Function Generator
- Oscilloscope

IV.PERCOBAAN

Untuk melakukan praktikum ini anda harus melakukan pengesetan peralatan PC Host sebagai tempat menyusun project, TMS320C6713 DSK sebagai pengolah sinyal, Function Generator sebagai pembangkit sinyal input dan Oscilloscope atau Speaker Active sebagai display sinyal output yang dihasilkan. Jangan melangkah lebih jauh sebelum penataan perangkat anda disetujui oleh dosen atau asisten praktikum.

4.1. Pembangkitan Sinyal Persegi

1. Buat project baru dan beri nama **squarewave.pjt**
2. Buat source dan beri nama **squarewave.c.** Untuk source program anda bisa memanfatkan listing berikut ini.

```
//Squarewave.c Generates a squarewave using a look-up table
#include "dsk6713_aic23.h"                    //codec-DSK interface support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;            //set sampling rate
#define table_size (int)0x40                  //size of table=64
short data_table[table_size];                 //data table array
int i;
interrupt void c_int11()                      //interrupt service routine
{
 output_sample(data_table[i]);                //output value each Ts
 if (i < table_size) ++i;                     //if table size is reached
 else i = 0;                                  //reinitialize counter
 return;                                      //return from interrupt
}

main()
{
 for(i=0; i<table_size/2; i++)                //set 1st half of buffer
   data_table[i] = 0x7FFF;                    //with max value (2^15)-1
 for(i=table_size/2; i<table_size; i++) //set 2nd half of buffer
   data_table[i] = -0x8000;                   //with -(2^15)

 i = 0;                                                    //reinit counter
 comm_intr();                                 //init DSK, codec, McBSP
 while (1);                                   //infinite loop
}
```

3. Tambahkan source program ke dalam project yang sudah anda buat.
4. Tambahkan file-file pendukung untuk berkomunikasi dengan IO sesuai yang dibutuhkan project ini, seperti file **vectors_intr.asm**, dsb.
5. Tambahkan file-file library pendukung project anda, seperti **rts6700.lib**, dsb.
6. Lakukan Scan All Files Dependencies untuk mengetahui apakah file-file pendukung sudah terhubung dengan project anda.
7. Lakukan pengesetan pada Build Option, untuk Compiler dan Linker.
8. Compile program anda, increment build dan rebuild all. Dan lanjutkan Load outputnya ke memory DSK.
9. Running program anda, perhatikan apa yang tampil pada Oscilloscope atau Speaker Active anda.
10. Lakukan penghentian program anda dengan menekan Halt, dan lakukan perubahan nilai hexa decimal pada variable size_tabel, misalnya anda ganti sehingga ekuivalent dengan

32. Dan lakukan rebuild all, dst untuk mengetahui pengaruh pada bentuk sinyal yang dihasilkan.

11. Lanjutkan modifikasi pembangkitan sinyal persegi dengan melakukan perubahan sesuai table berikut. Untuk melakukan perubahan ini anda harus memperhatikan nilai sampling rate yang digunakan

Tabel pengesetan sinyal persegi

No	size_table	Dec_equivalent	Estimasi Frequensi	Frek. hasil pengukuran	Bentuk sinyal satu periode
1		512			
2		256			
3		128			
4	0x40	64	125 Hz	…..	
5		32			
6		16			
7		8			

4.2. Pembangkitan Sinyal Gigi Gergaji

1. Buat project baru dan beri nama **ramptable.pjt**
2. Buat source dan beri nama **ramptable.c.** Untuk source program anda bisa memanfatkan listing berikut ini.

```
//Ramptable.c Generates a ramp using a look-up table

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

#define table_size (int)0x400  //size of table=1024
short data_table[table_size];    //data table array
int i;

interrupt void c_int11()       //interrupt service routine
{
 output_sample(data_table[i]); //ramp value for each Ts
 if (i < table_size-1) i++;    //if table size is reached
 else i = 0;                   //reinitialize counter
 return;                                  //return from interrupt
}
main()
{
  for(i=0; i < table_size; i++)
   {
    data_table[i] = 0x0;       //clear each buffer location
    data_table[i] = i * 0x20;  //set to 0,32,64,96,...,32736
   }
```

```
    i = 0;                              //reinit counter
    comm_intr();              //init DSK, codec, McBSP
    while (1);                //infinite loop
}
```

3. Tambahkan source program ke dalam project yang sudah anda buat.
4. Tambahkan file-file pendukung untuk berkomunikasi dengan IO sesuai yang dibutuhkan project ini, seperti file **vectors_intr.asm**, dsb.
5. Tambahkan file-file library pendukung project anda, seperti **rts6700.lib**, dsb.
6. Lakukan Scan All Files Dependencies untuk mengetahui apakah file-file pendukung sudah terhubung dengan project anda.
7. Lakukan pengesetan pada Build Option, untuk Compiler dan Linker.
8. Compile program anda, increment build dan rebuild all. Dan lanjutkan Load outputnya ke memory DSK.
9. Running program anda, perhatikan apa yang tampil pada Oscilloscope atau Speaker Active anda.
10. Lakukan penghentian program anda dengan menekan Halt, dan lakukan perubahan nilai hexa decimal pada variable size_tabel, misalnya anda ganti sehingga ekuivalent dengan 512. Dan lakukan **rebuild all**, dst untuk mengetahui pengaruh pada bentuk sinyal yang dihasilkan.
11. Lanjutkan modifikasi pembangkitan sinyal persegi dengan melakukan perubahan sesuai table berikut. Untuk melakukan perubahan ini anda harus memperhatikan nilai sampling rate yang digunakan

    Tabel pengesetan sinyal gigi gergaji (ramp)

No	size_table	Dec_equivalent	Estimasi Frequensi	Frek. hasil pengukuran	Bentuk sinyal satu periode
1	0x400	1024	7.8 Hz	…..	
2		512			
3		256			
4		128			
5		64			
6		32			
7		16			

4.3. Pembangkitan Sinyal Segitiga

1. Buat project baru dan beri nama **segitiga.pjt**
2. Buat source dan beri nama **segitiga.c.** Untuk source program anda bisa memodifikasi listing program pada ramptable.c, sedemikian hingga dua siklus pada sinyal ramp akan ekuivalent dengan satu siklus sinyal segitiga. Atau sebagai alternative, anda bisa memaksakan satu siklus sinyal ramp akan ekuivalen dengan satu siklus sinyal segitiga,

tetapi amplitude sinyal segitiga senilai setengah amplitude sinyal ramp.
3. Tambahkan source program ke dalam project yang sudah anda buat.
4. Tambahkan file-file pendukung untuk berkomunikasi dengan IO sesuai yang dibutuhkan project ini, seperti file **vectors_intr.asm**, dsb.
5. Tambahkan file-file library pendukung project anda, seperti **rts6700.lib**, dsb.
6. Lakukan Scan All Files Dependencies untuk mengetahui apakah file-file pendukung sudah terhubung dengan project anda.
7. Lakukan pengesetan pada Build Option, untuk Compiler dan Linker.
8. Compile program anda, increment build dan rebuild all. Dan lanjutkan Load outputnya ke memory DSK.
9. Running program anda, perhatikan apa yang tampil pada Oscilloscope atau Speaker Active anda.
10. Lakukan penghentian program anda dengan menekan Halt, dan lakukan perubahan nilai hexa decimal pada variable size_tabel, misalnya anda ganti sehingga ekuivalent dengan 512. Dan lakukan **rebuild all**, dst untuk mengetahui pengaruh pada bentuk sinyal yang dihasilkan.
11. Lanjutkan modifikasi pembangkitan sinyal persegi dengan melakukan perubahan sesuai table berikut. Untuk melakukan perubahan ini anda harus memperhatikan nilai sampling rate yang digunakan

Tabel pengesetan sinyal segitiga

No	size_table	Dec_equivalent	Estimasi Frequensi	Frek. hasil pengukuran	Bentuk sinyal satu periode
1	0x400	1024	7.8 Hz	…..	
2		512			
3		256			
4		128			
5		64			
6		32			
7		16			

4.4.Pembangkitan Sinyal Sinus

1. Buat project baru dan beri nama **sinegen_table.pjt**
2. Buat source dan beri nama **sinegen_table.c.** Untuk source program anda bisa memanfatkan listing berikut ini.

```
//Sinegen_table.c Generates a sinusoid for a look-up table

#include "DSK6713_AIC23.h"        //codec-DSK support file
```

```
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
#include <math.h>
#define table_size (short)10          //set table size
short sine_table[table_size];         //sine table array
int i;

interrupt void c_int11()      //interrupt service routine
{
 output_sample(sine_table[i]); //output each sine value
 if (i < table_size - 1) ++i;  //incr index until end of table
    else i = 0;                        //reinit index if end of table
 return;                               //return from interrupt
}

void main()
{
float pi=3.14159;

for(i = 0; i < table_size; i++)
  sine_table[i]=10000*sin(2.0*pi*i/table_size); //scaled values

i = 0;
comm_intr();                //init DSK, codec, McBSP
while(1);                   //infinite loop
}
```

3. Tambahkan source program ke dalam project yang sudah anda buat.
4. Tambahkan file-file pendukung untuk berkomunikasi dengan IO sesuai yang dibutuhkan project ini, seperti file **vectors_intr.asm**, dsb.
5. Tambahkan file-file library pendukung project anda, seperti **rts6700.lib**, dsb.
6. Lakukan Scan All Files Dependencies untuk mengetahui apakah file-file pendukung sudah terhubung dengan project anda.
7. Lakukan pengesetan pada Build Option, untuk Compiler dan Linker.
8. Compile program anda, increment build dan rebuild all. Dan lanjutkan Load outputnya ke memory DSK.
9. Running program anda, perhatikan apa yang tampil pada Oscilloscope atau Speaker Active anda.
10. Lakukan penghentian program anda dengan menekan Halt, dan lakukan perubahan nilai hexa decimal pada variable size_tabel, misalnya anda ganti sehingga ekuivalent dengan 512. Dan lakukan **rebuild all**, dst untuk mengetahui pengaruh pada bentuk sinyal yang dihasilkan.
11. Lanjutkan modifikasi pembangkitan sinyal persegi dengan melakukan perubahan sesuai table berikut. Untuk melakukan perubahan ini anda harus memperhatikan nilai sampling rate yang digunakan

Tabel pengesetan sinyal sinus

No	size_table (decimal)	Estimasi Frequensi	Frek. hasil pengukuran	Bentuk sinyal satu periode
1	10	800 Hz	…..	
2		400 Hz		
3		300 Hz		

4		200 Hz		
5		150 Hz		
6		100 Hz		
7		80 Hz		

4.5.Pembangkitan Sinyal Pseudo Noise

1. Buat project baru dan beri nama **noise_gen.pjt**
2. Buat source dan beri nama **noise_gen.c.** Untuk source program anda bisa memanfatkan listing berikut ini.

```
//Noise_gen.c Pseudo-random sequence generation

#include "DSK6713_AIC23.h"        //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
#include "noise_gen.h"                    //header file for noise sequence
short fb;
shift_reg sreg;                           //shift reg structure

interrupt void c_int11()          //interrupt service routine
{
 short prnseq;                    //for pseudo-random sequence

 if(sreg.bt.b0)                   //sequence{1,-1}based on bit b0
   prnseq = -8000;                        //scaled negative noise level
 else
   prnseq = 8000;             //scaled positive noise level
 fb  =(sreg.bt.b0)^(sreg.bt.b1);   //XOR bits 0,1
 fb ^=(sreg.bt.b11)^(sreg.bt.b13); //with bits 11,13 ->fb
 sreg.regval<<=1;                 //shift register 1 bit to left
 sreg.bt.b0 = fb;                       //close feedback path

 output_sample(prnseq);             //output scaled sequence
 return;                                //return from interrupt
}


void main()
{
 sreg.regval = 0xFFFF;             //set shift register
 fb = 1;                        //initial feedback value
 comm_intr();                     //init DSK, codec, McBSP
 while (1);                      //infinite loop
}
```

Pada listing program diatas, pengesetan pada bit register adalah b0, b1, b11 dan b13 yang di-EXOR, atau digunakan sebagai feedback pada pseudo noise generator. Untuk menghasilkan pola random yang lain, anda bisa melakukan perubahan dengan memiliki bit register yang akan anda gunakan sebagai feedback.

3. Tambahkan source program ke dalam project yang sudah anda buat.
4. Tambahkan file-file pendukung untuk berkomunikasi dengan IO sesuai yang dibutuhkan project ini, seperti file **vectors_intr.asm**, dsb.
5. Tambahkan file-file library pendukung project anda, seperti **rts6700.lib**, dsb.
6. Lakukan Scan All Files Dependencies untuk mengetahui apakah file-file pendukung sudah terhubung dengan project anda.
7. Lakukan pengesetan pada Build Option, untuk Compiler dan Linker.
8. Compile program anda, increment build dan rebuild all. Dan lanjutkan Load outputnya ke memory DSK.
9. Running program anda, perhatikan apa yang tampil pada Oscilloscope atau Speaker Active anda.
10. Lakukan penghentian program anda dengan menekan Halt, dan lakukan perubahan nilai hexa decimal pada variable size_tabel, misalnya anda ganti sehingga ekuivalent dengan 512. Dan lakukan **rebuild all**, dst untuk mengetahui pengaruh pada bentuk sinyal yang dihasilkan.
11. Lanjutkan modifikasi pembangkitan sinyal persegi dengan melakukan perubahan sesuai table berikut. Untuk melakukan perubahan ini anda harus memperhatikan nilai sampling rate yang digunakan

Tabel pengesetan feedback pseudo noise

No	Bit register feedback	Bentuk sinyal satu periode
1	b0, b1, b11 dan b13	
2	b0, b1, b2 dan b3	
3	b1, b3, b5 dan b7	
4	b0, b1, b11, b12 dan b13	
5		
6		

V. TUGAS

1. Buatlah diagram alir (flowchart) untuk tiap pembangkitan sinyal yang telah anda lakukan.
2. Dari hasil percobaan, menurut anda berapakah frekuensi tertinggi yang mampu dihasilkan apabila kita membangkitkan sinyal menggunakan board DSKC6713?
3. Pada pembangkitan sinyal persegi dan sinyal gigi-gergaji apakah menghasilkan bentuk sinyal yang tidak sesuai harapan? Mengapa hal ini dapat terjadi?

MODUL V

INTEGRASI MATLAB SIMULINK DENGAN TMS320C6713 DSK

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:

- Siswa memahami konsep integrasi perangkat Lunak Matlab dengan DSP Board
- Siswa mampu membangun sebuah integrasi untuk pengolahan sinyal sederhana

II. INTEGRASI MATLAB TOOLS DENGAN DSP BOARD TMS320C6713.

Matlab telah menyediakan sebuah fungsi untuk berkomunikasi dengan Embedded Target. Salah satun fungsinya adalah untuk Texas Instruments TMS320C6000 (tm) DSP Platform, yang mengintegrasikan Simulink and MATLAB dengan Texas Instruments eXpressDSP(tm) tools. Software ini memungkinkan bagi anda untuk membangun dan memvalidasi hasil perancangan pengolahan sinyal digital mulai dari koncep sampai dengan kode dan secara otomatis melakukan pembuatan prototipe pada Texas Instruments DSP Starter Kit atau Evaluation Module yang anda miliki. Proses ***build*** pada program akan menghasilkan suatu Code Composer Studio project dari C code yang dibangkitkan melalui Real-Time Workshop. Semua fitur disediakan oleh Code Composer Studio(tm), misalnya untuk **editing, building, debugging, code profiling**, dan **project management**. Di sini juga ada pilihan, dimana Code Composer Studio(tm) project yang sudah terbentuk secara otomatis di-***compiled and linked***, dan hasil executable di-loaded kepada board, dan dijalankan pada C6xxx DSP.

Dengan membangun sebuah software user interface menggunakan tool Matlab yang mana akan memungkinkan bagi seorang user untuk mengeksekusi secara mudah menjadi kode bahasa C, C++ dan assembly pada sebuah DSP board Texas Instrument (TMS320C6713).

2.1. Blok Diagram Sistem

Software interface yang dibangun untuk integrasi kode akan beroperasi menyesuaikan block diagram yang ada pada Gambar 1.

2.2. Host PC with MATLAB Development Tools

Untuk awal proses pengolahan sinyal yang bisa diimplementasikan, diawali dengan penulisan kode menggunakan Matlab development tools. Jika kode pengolahan sinyal diadalam bahasa C++, C, hal ini bisa dikonversi ke dalam suatu MEX file, yang mana bisa diinterface dengan mengunakan Matlab. Kode-kode Matlab yang dibangkitakan disimpan dalam bentuk input file atau suatu workspace yang akan digunakan di dalam Simulink.

Kita juga bisa melakukan langsung pemodelan suatu system dengan memanfatkan perangkat Simulink yang sudah disediakan di dalam Matlab. Cara kedua ini relatif lebih sederhana, tetapi akan lebih baik jika anda lebih dulu menguasai pemrograman dalam Matlab Command atau Bahasa C.

Gambar 1. Blok Diagram integrasi Matlab dengan DSP board

2.3. Simulink

Simulink merupakan sebuah tool Box yang dikembangkan dari Matlab yang digunakan untuk berbagai pemodelansistem,salah satunya adalah untuk menyelesaikan Pengolagan Sinyal. Dengan tool box kita bisa membentuk sebuah model system secar lebih mudah, bisa mengamati karakteristik sinyal input dan outputnya dengan cara menampilkannya secara real time, dan sistemnya disajikan secara graphic user interface (GUI).

Kita bisa membuat nmodel baru di dalam Simulink dengan menggunakan **Matlab Command, Matlab FDATool** atau melalui penulisan menggunakan bahasa C.

Setelah suatu model dibuat dan mampu merepresentasikan karaktersistik system yang kita inginkan, **Simulink** bisa mengirimkan hasil kompilasi kepada **Code Composer Studio (CCS).**

2.4. Code Composer Studio (CCS)

CCS mengintegrasikan software yang sudah dibentuk dari Simulink dan mengkonversikannya ke dalam bahasa C dan assembly, yang mana outputnya di-download sebagai sebuah output file pada TMS3206713. Proses running pdapat diaksesa dari CCS debugging tools atau melintasi suatu link untuk CCS atau Real-time data Exchange. Kalau kita masih tidak mau meninggalkan **Matlab**, kita juga bisa mengakses proses running dari **Matlab Simulink.**

2.5. DSP Board (TMS3206713)

TMS3206713 menggantikan kerja dari Matlab Simulink. Board TMS3206713 menghasilkan output proses pengolahan data atau sinyal secara real time dan bisa ditampilkan melalui oscilloscope. Ouput tampilan pada oscilloscope merepresentasikan hasil pengolahan sinyal yang kita bangun.

III.PERALATAN PERCOBAAN

- Sebuah PC dilengkapi dengan OS Windows Xp, Sound Card dan Speaker
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box Simulink
- CCS dan DSP Board TMS320C6713
- Microphone
- Oscilloscope

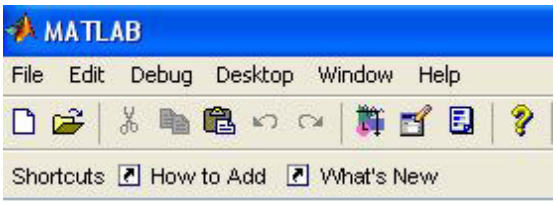
IV.PERCOBAAN

4.1. Memulai Matlab Simulink

1. Untuk memulai sebuah sesi Simulink, anda perlu membuka Matlab terlebih dahulu, setelah Matlab Command dalam kondisi aktif, anda ketikkan

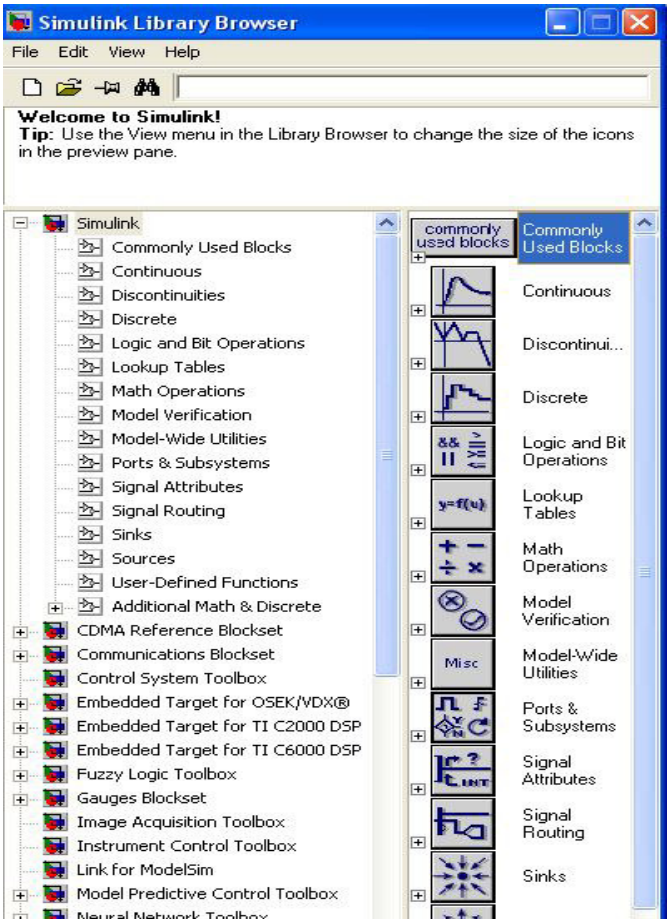
   ```
   >> simulink
   ```

   Sebagai alternative dari langkah ini, anda bisa juga membuka simulink dengan click mouse anda pada ikon Simulink yang terletak di toolbar berikut



Gambar 2. Langkah alternative membuka simulink

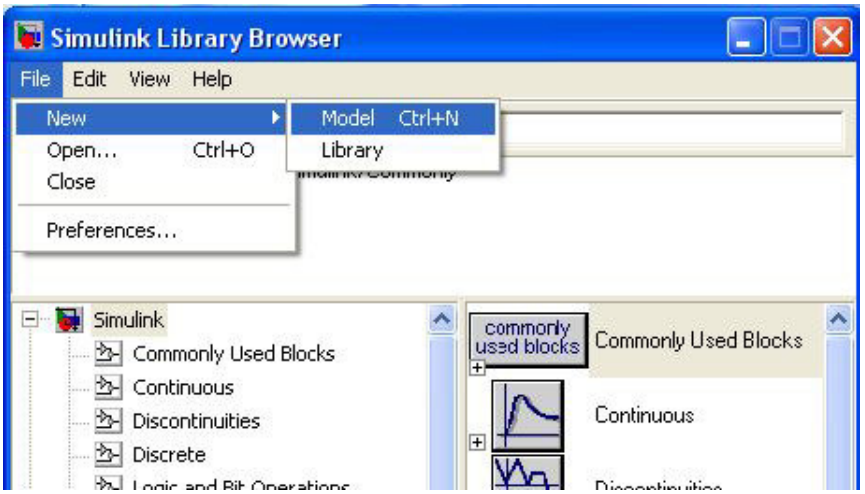
Simulink's library akan menampilkan sebuah window yang menunjukkan bagian-bagian dari simulink yang bisa digunakan untuk menyusun block set untuk sebuah konstruksi model.



Gambar 3. Window Simulink Library

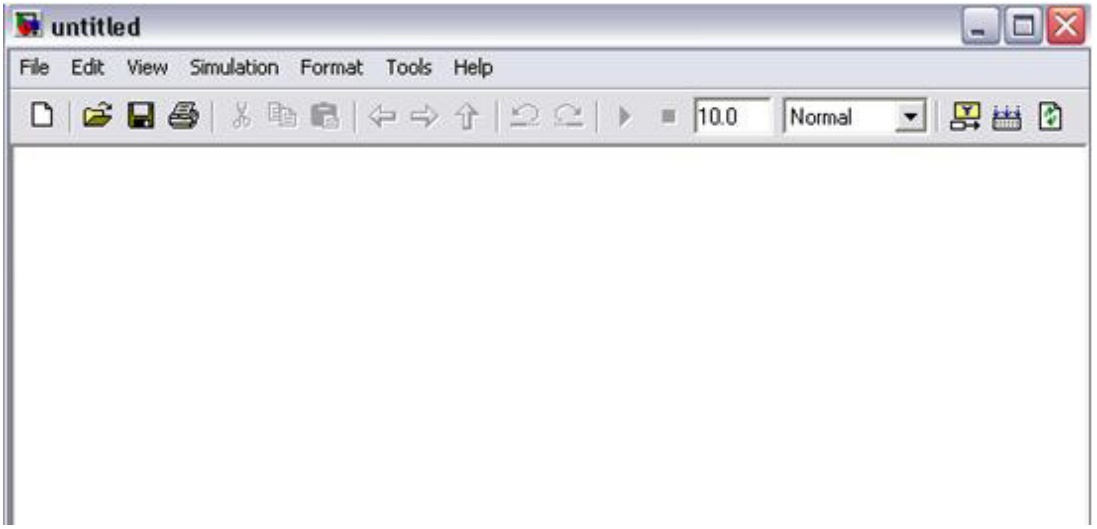
2. Untuk melihat isi dari blockset, click pada tanda "+"di depan masing-masing kelompok toolbox.
3. Untuk memulai membuat sebuah model click pada NEW FILE ICON seperti yang ditunjukkan pada tampilan diatas. Sebagai alternatifnya, anda bisa juga menggunakan keyboard **CTRL+N**.

Sebuah window yang baru akan muncul pada layar monitor anda. Anda akan memulai membuat model di dalam window ini, dan juga akan melakukan simulasi dari model yang anda buat nantinya.



Gambar 4. Langkah membuat New Model

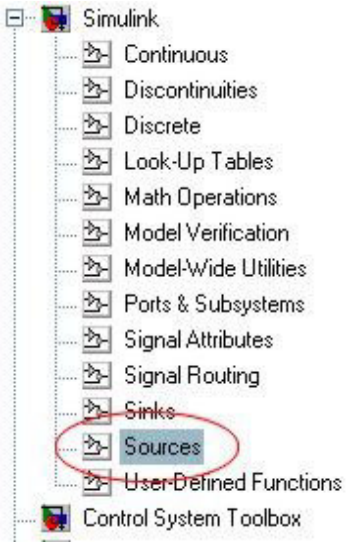
4. Akan muncul sebuah model simulink seperti berikut



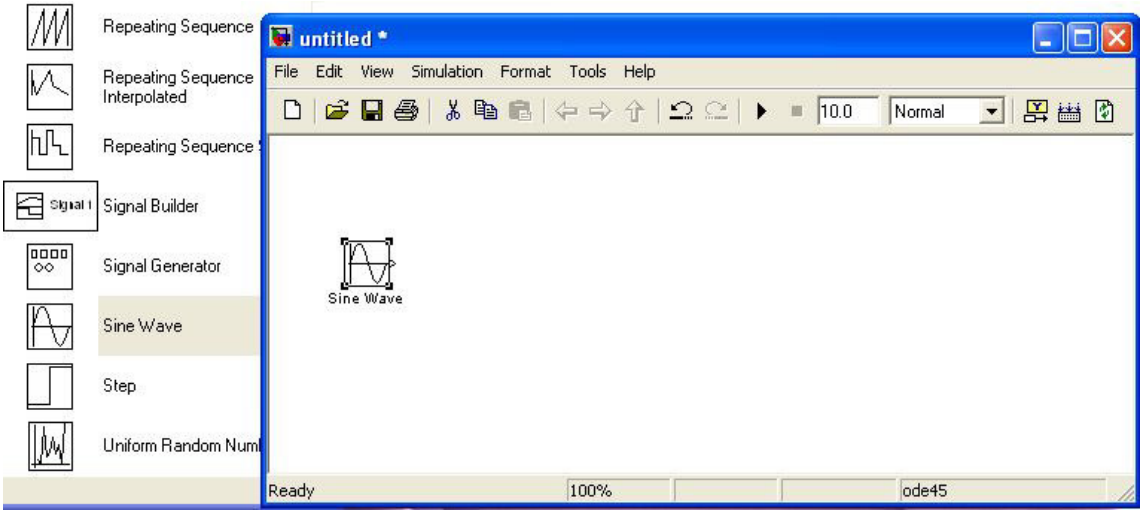
Gambar 5. Window Simulink yang baru

5. Dari sesi BLOCK SET CATEGORIES pada SIMULINK LIBRARY BROWSER window, click pada tanda "+" di depan group **Simulink** untuk mengeskpansi tree dan pilih (click pada) **Sources**.

Sebuah set block-blok akan muncul di dalam group BLOCKSET. Click pada blok **Sine Wave** dan drag (seret) blok ini window workspace (yang juga kita kenel sebagai model window).



Gambar 6. Langkah ekspansi pada sebuah blok library



Gambar 7. Langkah menggeser sebuah blok Sine wave ke Workspace

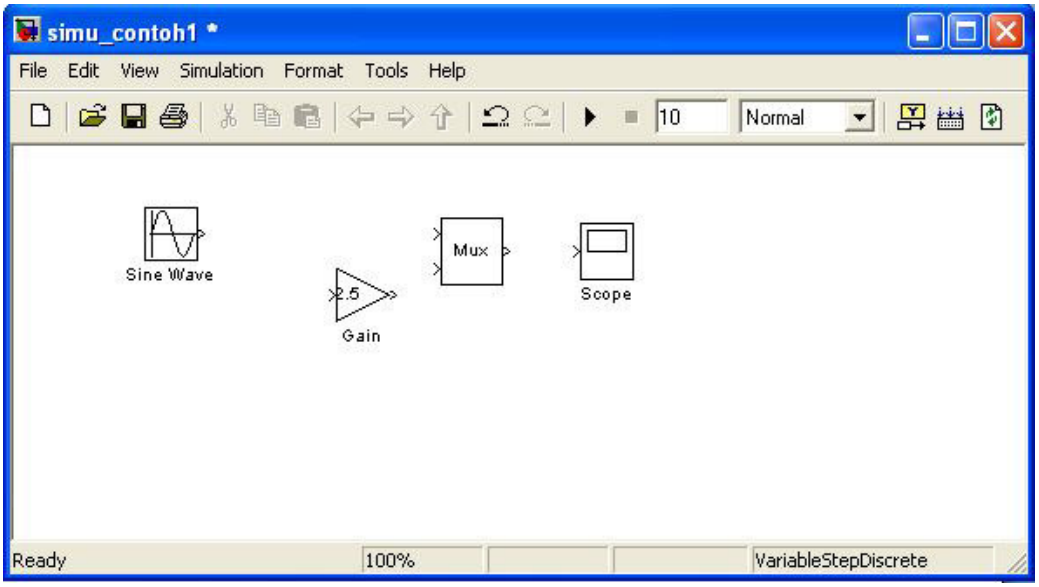
6. Sebaiknya anda segera menyimpan model ini, misalnya beri nama**"simu_contoh1"**. Untuk menyimpannya sebagai model, anda bisa click gambar ikon flopy disk. Atau dari menu File, pilih **Save** atau anda bisa juga dengan keyboard CTRL+S. Semua file model Simulink akan memiliki extension "**.mdl**".

7. Lanjutkan proses anda untuk menambah lebih banyak lagi blok yang anda perlukan untuk membangun model. Tambahkan sebuah **Scope** dari library **Sinks**, sebuah blok **Gain** dari library **Common Used Bloks**, dan sebuah blok **Mux** dari library **Signal Routing**.

Catatan: Jika anda hanya mengetahui nama sebuah block, tetapi tidak mengetahui lokasinya, maka anda bisa memasukkan nama blok tersebut ke SEARCH WINDOW

(pada **Find** prompt) dan Simulink akan membawanya ke blok tertentu.

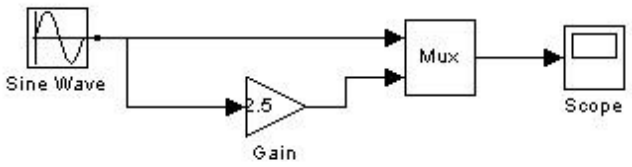
8. Setelah anda menyelesaikan langkah diatas, seharusnya anda mendapatkan bentuk tampilan model yang terdiri dari berbagai komponen (blok) seperti berikut:



Gambar 7. Hasil pengambilan komponen untuk model

Anda bisa menghapus atau menggeser untuk memindahkan posisi setiap blok yang anda inginkan.

9. Untuk membentuk sebuah hubungan antar blok, gerakkan kursor ke salah satu output dari salah satu blok yang dalam hal ini ditunjukkan dengan arah panah ke luar atau tanda ">" disebelah kanan (biasanya) pada blok. Ketika salah satu port disentuh dengan kursor dan anda buat garis hubungan (tanda "+") ke salah satu input dari blok yang lain (tanda “>”) di depan blok (biasanya sebelah kiri).

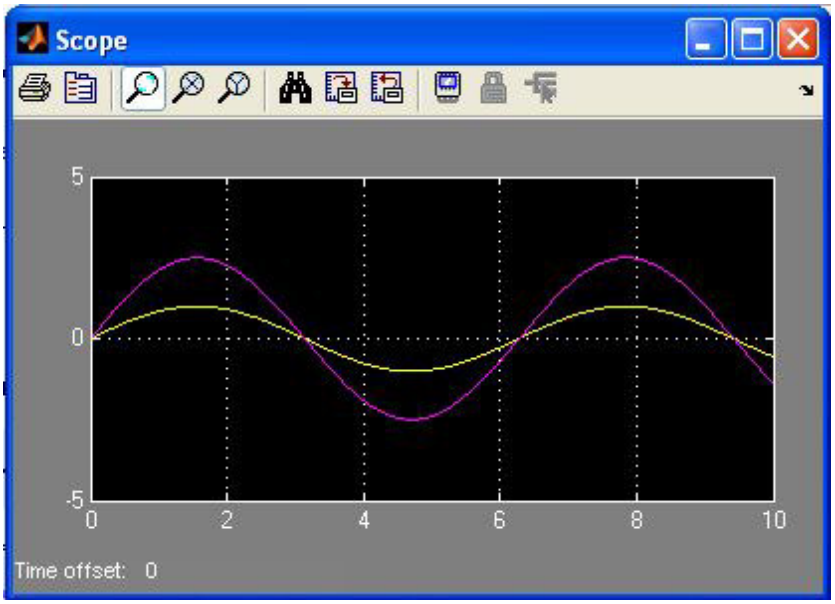


Gambar 8. Hubungan antar blok untuk model

Sebuah sinyal sinus dibangkitkan dari Sine Wave block (sebuah sumber) dan ditampilkan dengan menggunakan oscilloscope. Sinyal sinus yang sudah mengalami proses di dalam terintegrator dikirimkan ke oscilloscope untuk ditampilkan bersama dengan sinyal sinus asli dengan memanfaatkan sebuah **Mux**, yang mana dalam hal ni berfungsi untuk melakukan multiplexing sinyal dalam bentuk scalar, vector, atau matrix di dalam sebuah bus.

10. Sekarang anda bisa me-**run** simulasi pada system sederhana diatas dengan click pada ikon play atau tanda ( ). Sebagai alternative lain, anda bisa menggunakan keyboard CTRL+T, atau memilih sub menu **Start** (dibawah menu **Simulation**).

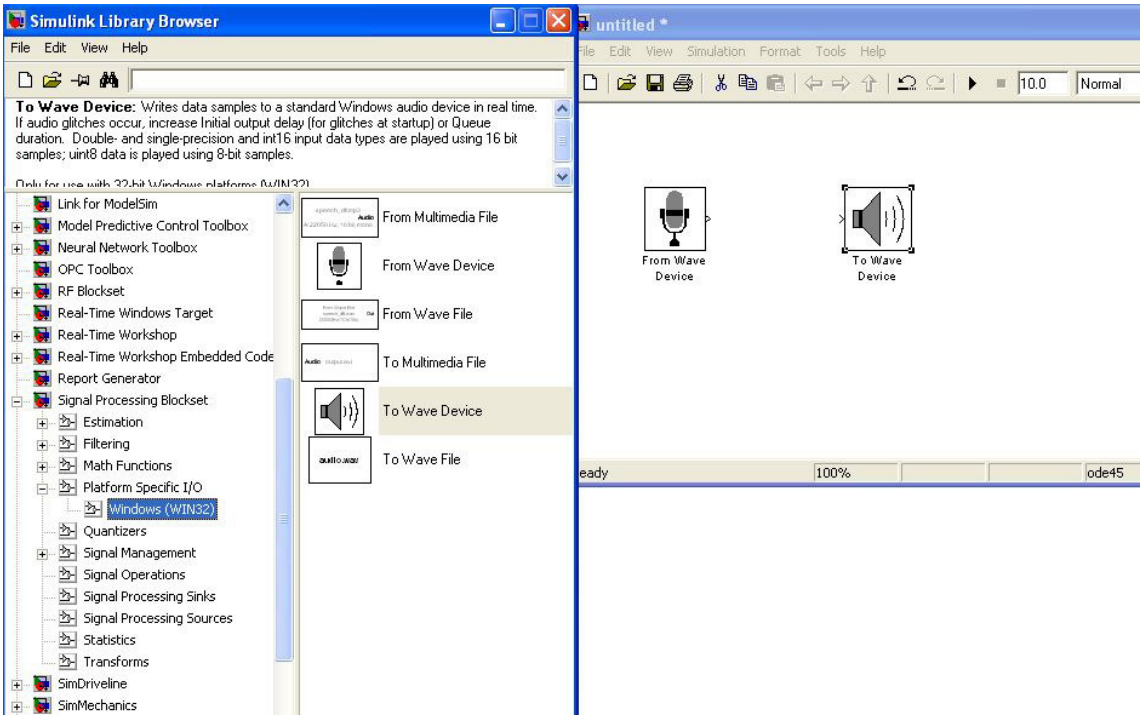
11. Jika anda ingin mengamati bentuk sinyal yang dihasilkan, double click pada blok Scope.



Gambar 9. Tampilan Scoope dua sinyal yang bercampur

4.2. Membangun Sebah Model Audio Sederhana

1. Untuk memulai membuat sebuah model click pada NEW FILE ICON seperti yang ditunjukkan pada tampilan diatas. Sebagai alternatifnya, anda bisa juga menggunakan keyboard **CTRL+N**.
2. Dari sesi BLOCK SET CATEGORIES pada SIMULINK LIBRARY BROWSER window, click pada tanda "+" di depan group Signal Processing Blockset untuk mengeskpansi, dan lanjutkan dengan mengekspansi Platform Specific I/O→ Windows WIN32.



Gambar 10. Langkah membangun model audio sederhana

Sekumpulan blok berkaitan dengan masalah I/O akan muncul di dalam group tersebut.

3. Click pada blok **From Wave Device** (gambar microphone) click dan seret blok ini window workspace (yang juga kita kenel sebagai model window).

4. Lakukan hal yang sama pada **To Wave Device** (gambar speaker)

Gambar 11. Gambaran system audio sederhana

5. Sebaiknya anda segera menyimpan model ini, beri nama**"sistem_Audio01"**. Untuk menyimpannya sebagai model, anda bisa click gambar ikon flopy disk. Atau dari menu File, pilih **Save** atau anda bisa juga dengan keyboard CTRL+S. Semua file model Simulink akan memiliki extension ".**mdl**".

6. Sekarang anda bisa me-**run** simulasi pada system sederhana diatas dengan click pada ikon play atau tanda (▶). Ucapkan kata-kata pada microphone anda, dan dengarkan apa yang keluar dari speaker.

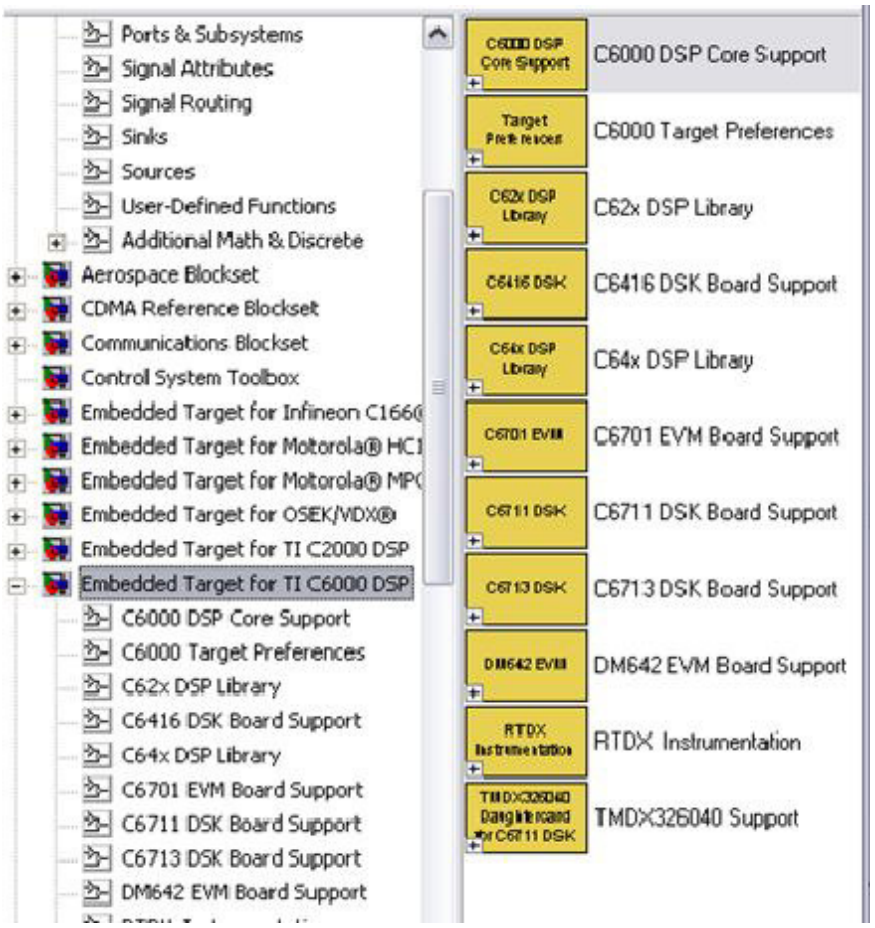
7. Jika anda ingin tahu lebih jauh, anda click pada blok From Wave Device dan anda amati parameternya, jika anda fikir ada yang perlu dirubah coba lakukan beberapa perubahan dan amati efeknya pada system anda. Lakukan hal yang sama pada To Wave Device.

4.3. Membangun Sebuah Koneksi dengan DSP Board

Pada sesi ini kita coba untuk mengkonfigurasikan sebuah koneksi Simulink dengan Real-Time board yang dalam hal ini kita gunakan C6713DSK. Konfigurasi ini akan dibentuk selangkah demi selangkah dalam rangkaian percobaan berikut.

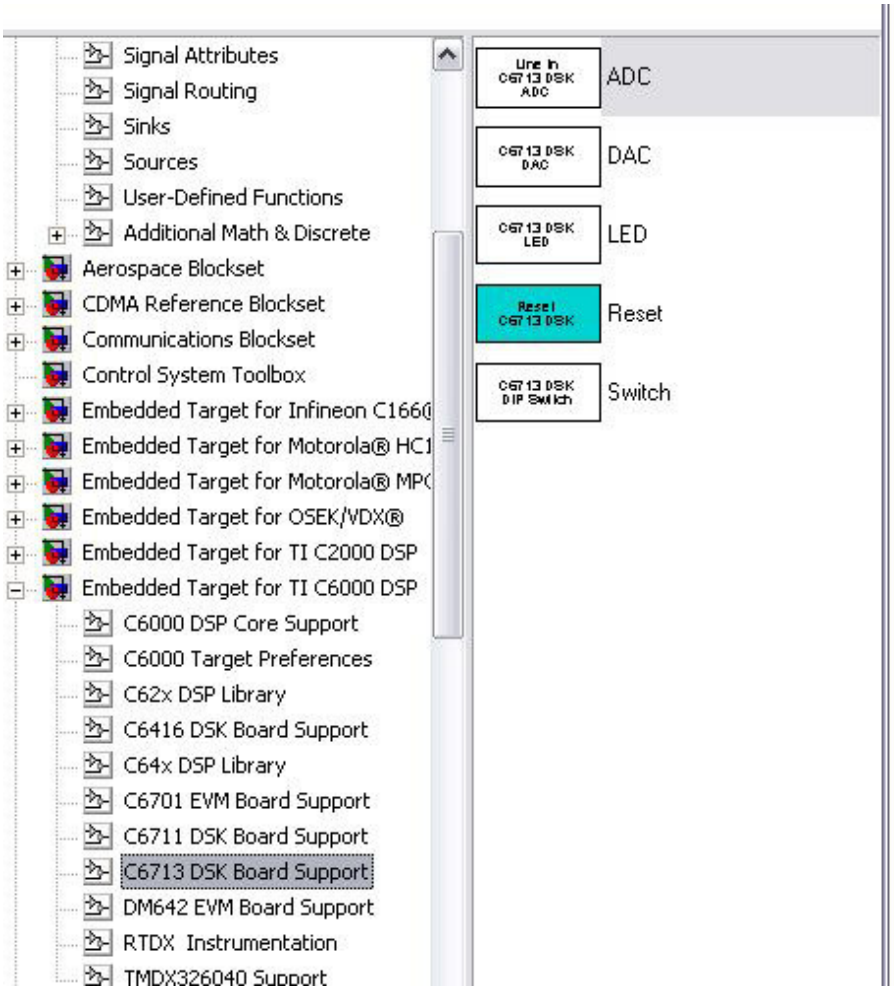
1. Aktifkan Code Composer Studio (CCS) anda. Jangan lupa semua koneksu USB, dan kondisikan TMS320C6713 DSK dalam keadaan benar-benar siap.
2. Aktifkan Matlab, dan lanjutkan untuk membuka Simulink.
3. Buat sebuah workspace baru pada Simulink.

4. Dari sesi BLOCK SET CATEGORIES pada SIMULINK LIBRARY BROWSER window, click pada tanda "+" di depan group **Ebedded Target for TIC6000** DSP



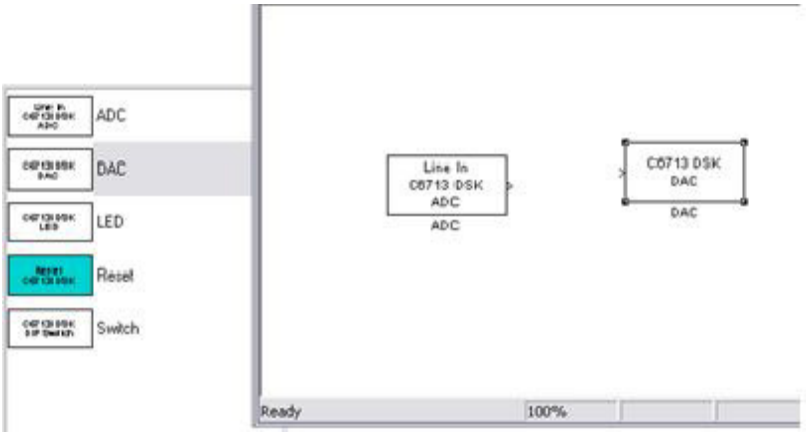
Gambar 12. Ekspansi pada Embedded Target for TI C6000 DSP

5. Anda click "tanda +" di depan group **C6713 DSK Board Support** untuk mengekpansi isinya. Disini akan muncul beragam pilihan blok bagian-bagian dari blok diagram DSK yang mewakili fungsi masing-masing. Misalnya ADC bisa digunakan untuk membuat koneksi sinyal dari luar (dari microphone, fungction generator, atau sumber sinyal yang lain) untuk diproses dengan model DSP. Blok DAC digunakan untuk mengolah hasil proses DSP dan dkirimkan ke perangkat analog di dunia luar, misalnya speaker atau oscilloscope.



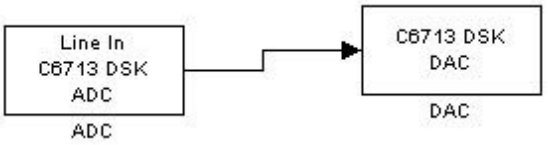
Gambar 13. Hasil ekspansi C6713 DSK Board Support

6. Lakukan Drag and drop dari **Simulink Library** menuju **Workspace** (Simulink Model) yang sedang anda buat. Proses ini persis sama dengan teknik yang anda gunakan untuk membangun hubungan pada model sinyal audio yang telah anda lakukan pada percobaan sebelumnya.



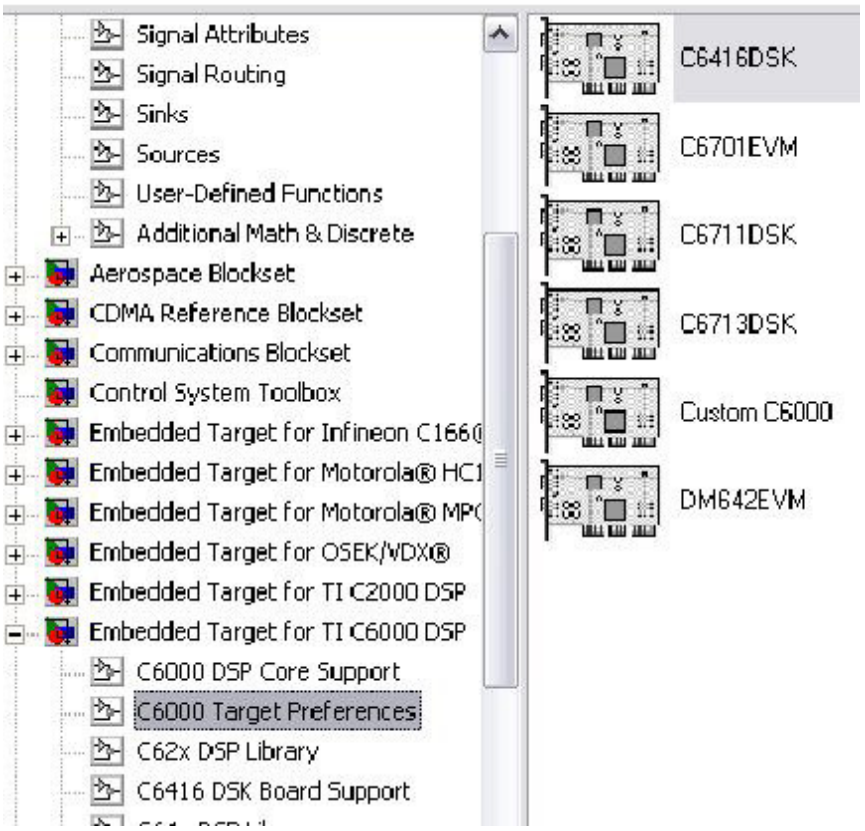
Gambar 14. Menempatkan ADC dan DAC pada Workspace

7. Untuk membentuk sebuah hubungan antar blok, gerakkan kursor ke salah satu output dari salah satu blok yang dalam hal ini ditunjukkan dengan arah panah ke luar atau tanda ">" disebelah kanan (biasanya) pada blok Line in ADC. Ketika salah satu port disentuh dengan kursor dan anda buat garis hubungan (tanda "+") ke salah satu input dari blok yang lain (tanda ">") di depan blok DAC (biasanya sebelah kiri).

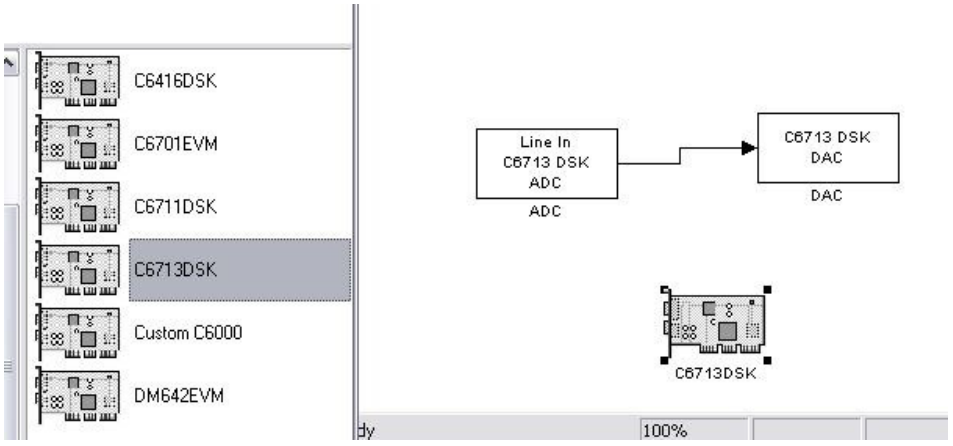


Gambar 15. Model hubungan ADC dan DAC pada Workspace

8. Anda click "tanda +" di depan group **C6000 DSP Preference** untuk mengekpansi isinya. Disini akan muncul beragam pilihan blok bagian-bagian dari blok diagram platform DSK yang mewakili tipe masing-masing.
9. Lanjutkan pilihan pada C6713, jika blok ini tidak ada sebaiknya anda konsultasi dengan asisten atau dosen pengampu praktikum.
10. Lakukan Drag and drop dari **C6000 DSP Preference** menuju **Workspace** (Simulink Model) yang sedang anda buat. Proses ini persis sama dengan teknik yang anda gunakan untuk membangun hubungan pada model sinyal audio yang telah anda lakukan pada percobaan sebelumnya.
11. Simpan model yang sudah anda bangun, kali ini anda bisa memberi nama sesuka anda, asalkan jangan sampai lupa jika sewaktu-waktu anda ingin membukanya kembali.



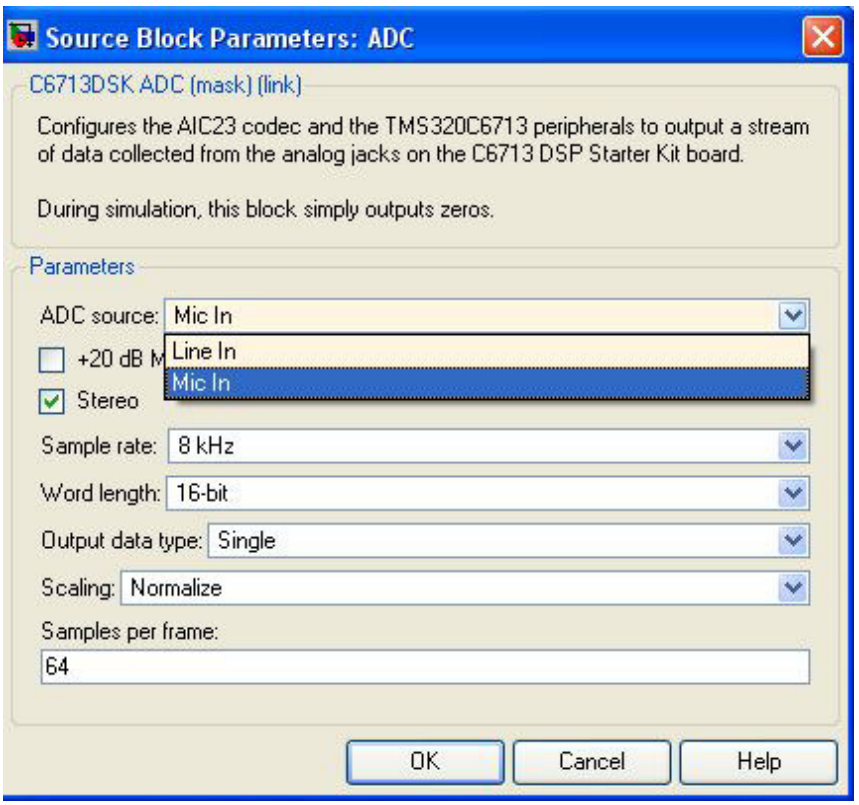
Gambar 16. Hasil ekspansi C6000 Target Preference



Gambar 17. Menempatkan C6713DSK pada Workspace

12. Untuk memulai melakukan simulasi, anda harus memodifikasi

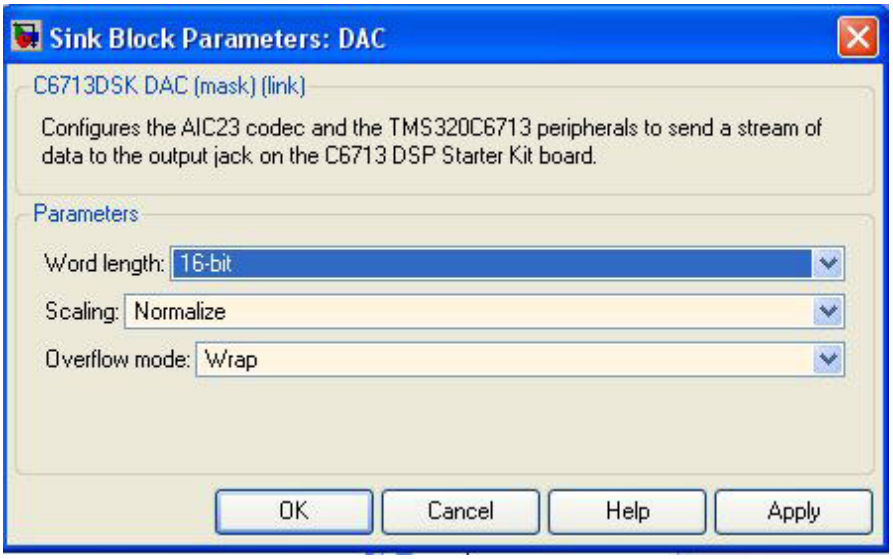
parameter-parameter dari semua blok diagram. Anda mulai dengan Double Click pada Line In C6713DSK ADC.



Gambar 18. Penentuan parameter pada blok ADC

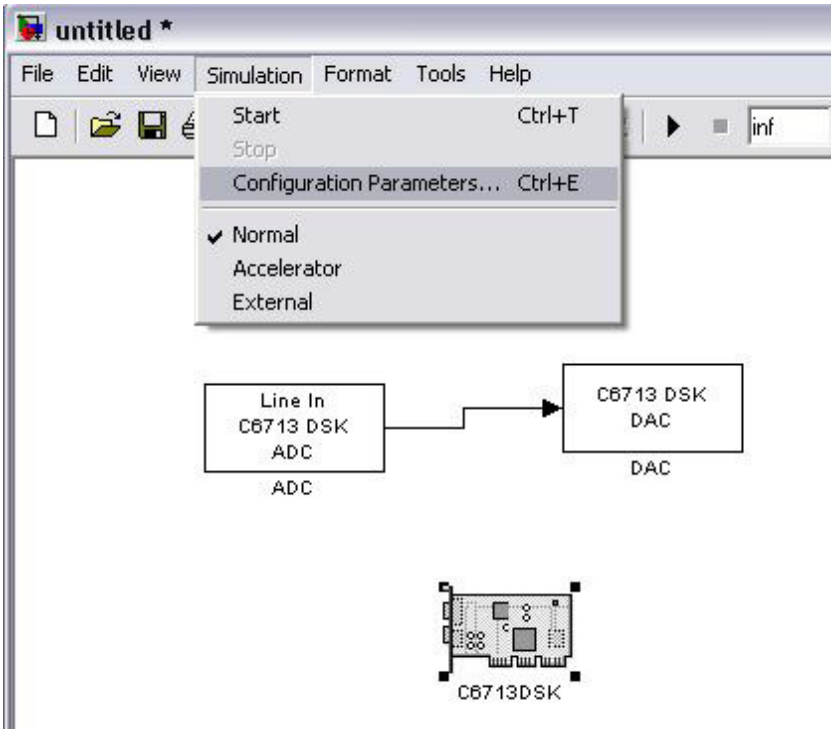
Pada scroll bar ADC Source, pilih Mic In, dalam hal ini anda bisa saja memilih Line In. Untuk sementara biaskan saja nilai-nilai seperti sample rate, World length, Ouput data type, Sacling dan sample per frame. Click **Apply→ OK.**

13. Anda lakukan hal yang sama pada C6713DSK DAC. Dalam hal ini tetapkan nilai Word Length 16-bit. Sedangkan parameter yang lain sebaiknya anda mengikuti persis seperti pada gambar berikut. Click **Apply→OK**.



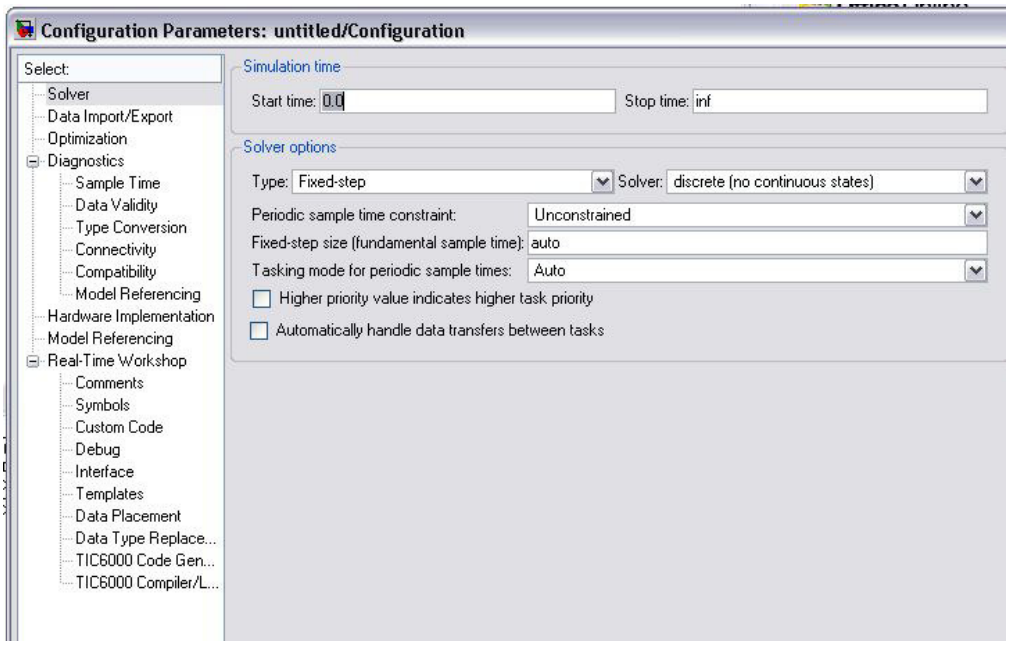
Gambar 19. Penentuan parameter pada blok DAC

14. Pada model simulink yang anda buat, anda modifikasi Configuration Parameters dnegn cara pada Toolbar anda pilih **Simulation → Configuration Parameter**



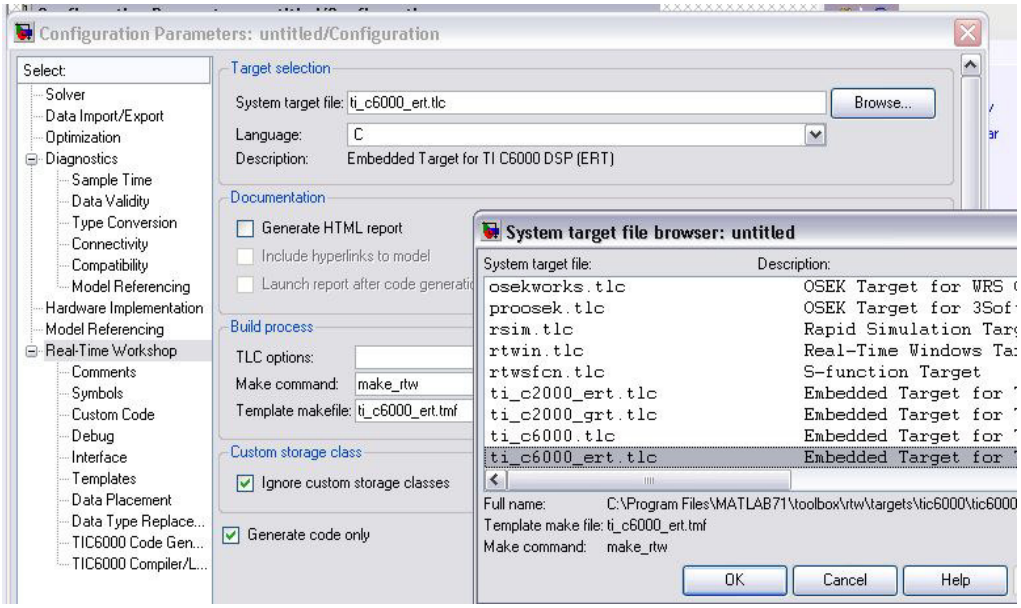
Gambar 20. Langkah konfigurasi parameter

Langkah diatas akan menampilkan gambar seperti berikut ini



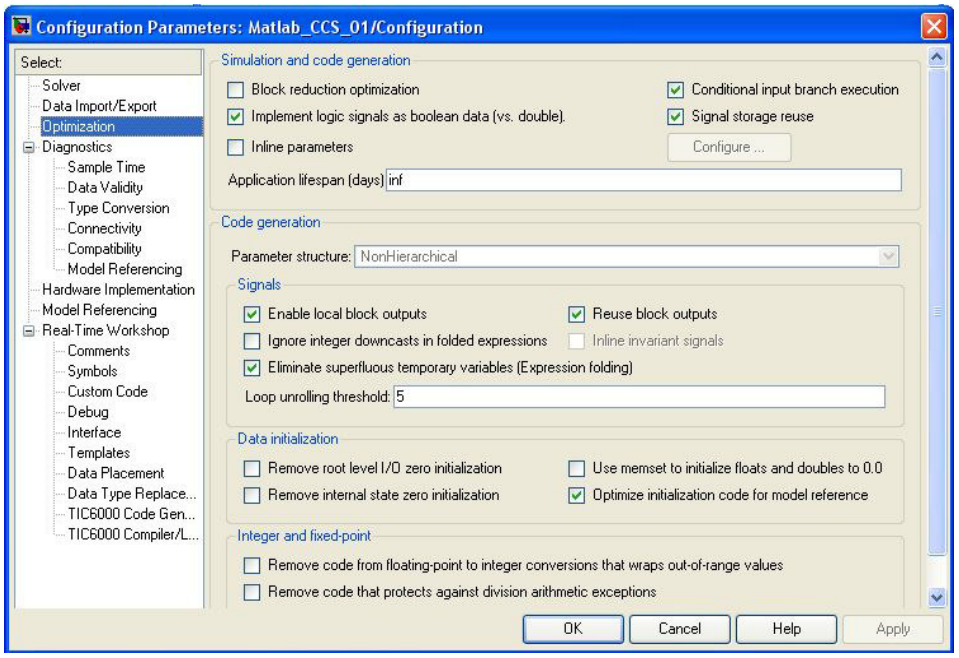
Gambar 21. Configuration Paramter

15. Perhatikan bagian kiri tampilan diatas, pada katagori Select anda click **"tanda +"** Real Time Workshop. Jika **"tanda +"** sudah berubah menjadi **"tanda -"**, maka anda tidak perlu menekan click. Selanjutnya anda click tepat pada Real Time Workshop.



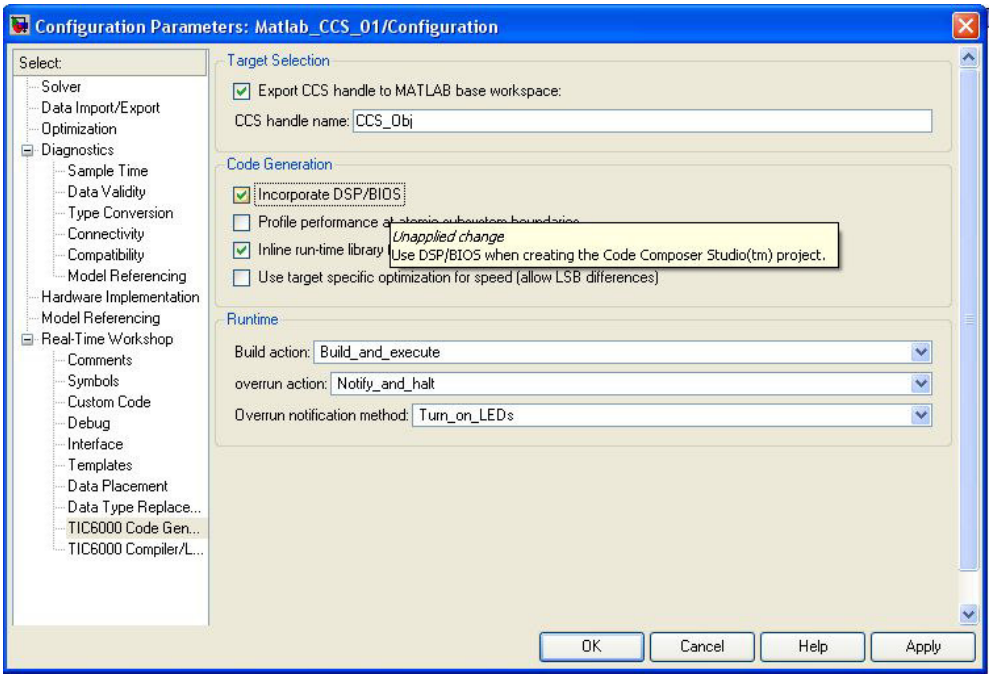
Gambar 22. Seting System target file

16. Pada System target file anda click **Browse**, dan pilih pada **ti_c6000ert.tlc**
17. Click katagori **Optimization**, hilangkan centang **Block reduction optimization**



Gambar 23. Seting katagori Optimization

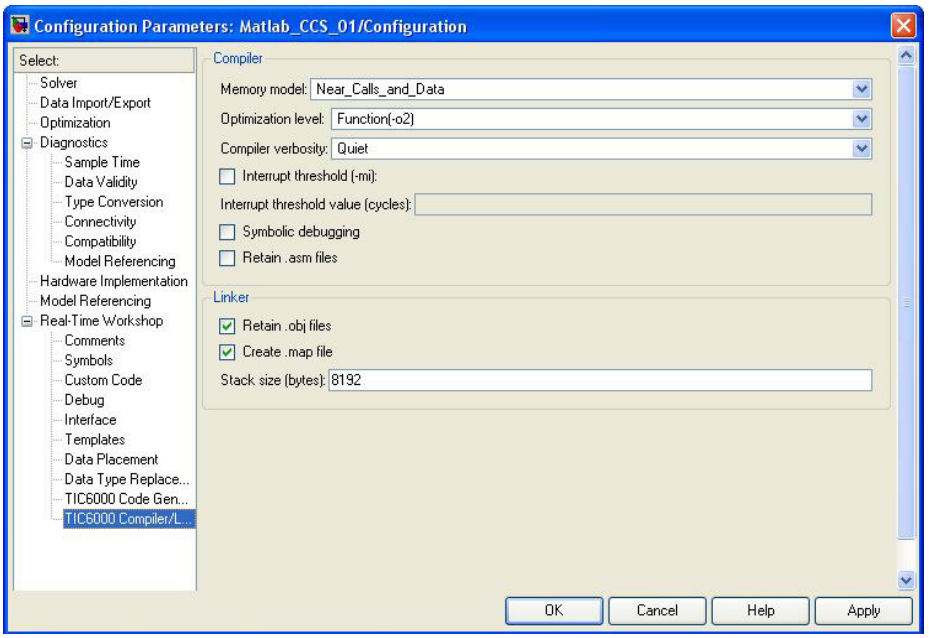
18. Anda lihat kembali katagori yang ada di dalam kelompok Real-Time Workshop, click pada **TIC6000 Code Gen…,** hilangkan centang pada **Incorporate DSP/BIOS**



Gambar 23. Seting katagori Optimization

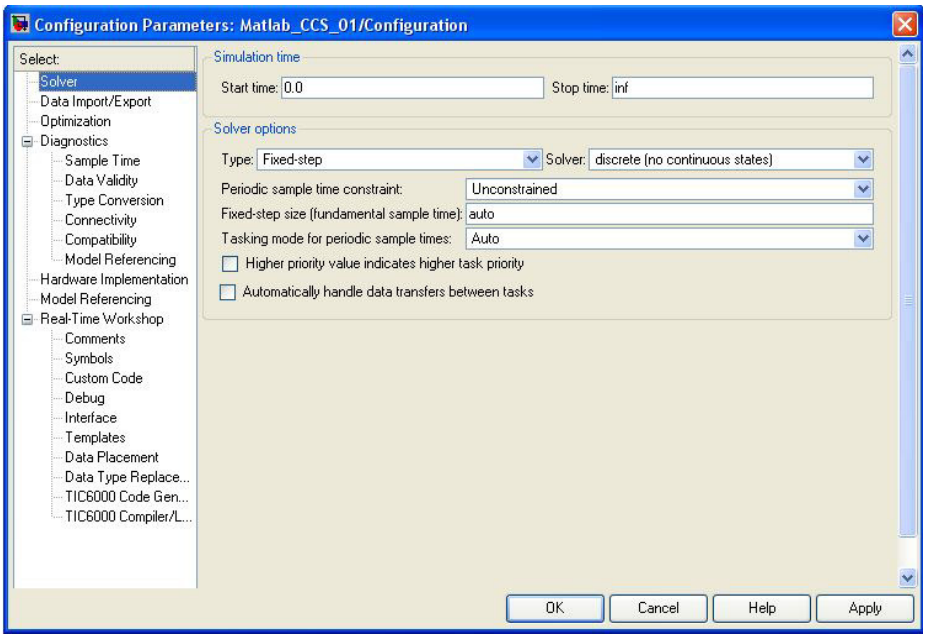
19. Click pada katagori TIC6000/L…, lakukan pemeriksaan apakah kondisinya sudah

seperti gambar berikut. Jika sudah sama, anda tidak perlu melakukan perubahan.



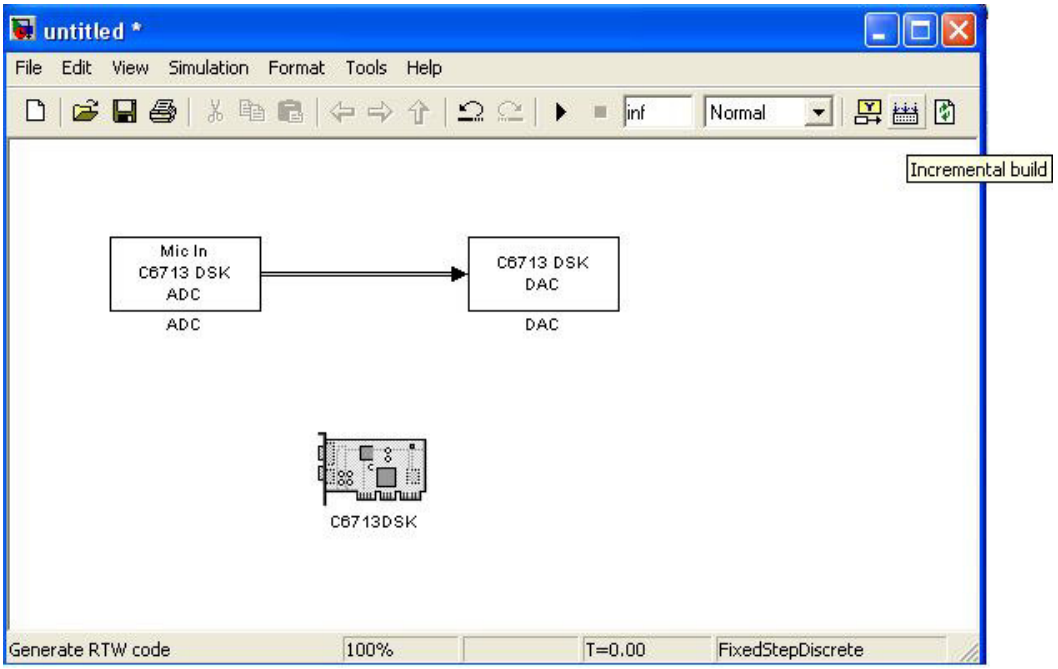
Gambar 24. Pemeriksaan TIC6000 Compiler/L…

20. Click pada katagori **Solver**, jika kondisinya sudah seperti gambar berikut, untuk sementara jangan lakukan perubahan.



Gambar 25. Pemeriksaan katagori Solver

21. Anda kembali aktif pada Simulink Model yang sedang anda buat, arahkan mouse anda ke toolbar Incremental Build, lakukan proses building.



Gambar 26. Persiapan proses build program

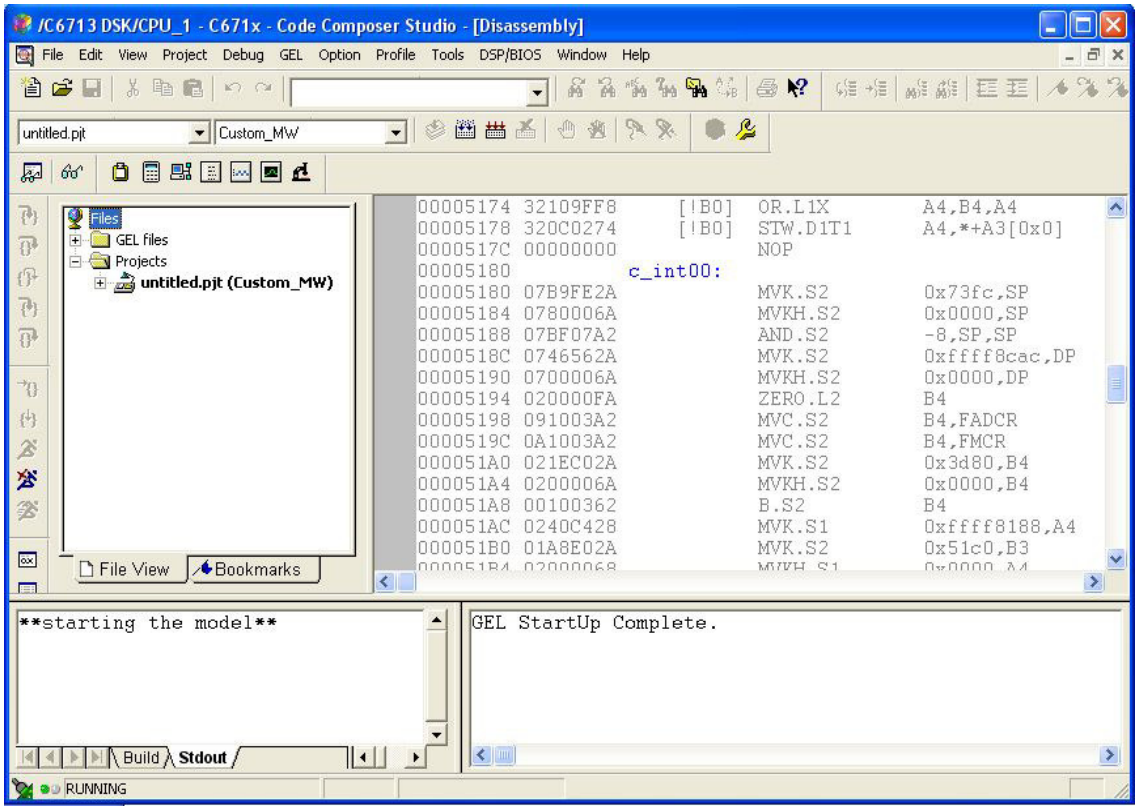
Perhatikan apa yang terjadi, jika tidak ada pesan error, maka pada **Matlab Command** akan muncul beberapa indicator berupa tampilan text seperti berikut…

```
### Generating code into build directory: C:\Program Files\MATLAB71\work\untitled_c6000_rtw
### Connecting to Code Composer Studio(tm)...
Warning: This version of the Link for Code Composer Studio(tm) has not been tested
with your version of Code Composer Studio(tm) IDE. Refer to the Link for Code
Composer Studio(tm) data sheet for supported versions of Code Composer Studio(tm).
Warning: CloseText: No project is loaded, nothing to close
### Invoking Target Language Compiler on untitled.rtw
    tlc
    -r
    C:\Program Files\MATLAB71\work\untitled_c6000_rtw\untitled.rtw
    C:\Program Files\MATLAB71\toolbox\rtw\targets\tic6000\tic6000\ti_c6000_ert.tlc
    -OC:\Program Files\MATLAB71\work\untitled_c6000_rtw
    -IC:\Program Files\MATLAB71\toolbox\rtw\targets\tic6000\tic6000
    -IC:\Program Files\MATLAB71\toolbox\rtw\targets\tic6000\tic6000blks\tlc_c
    -IC:\Program Files\MATLAB71\work\untitled_c6000_rtw\tlc
    -IC:\Program Files\MATLAB71\rtw\c\tlc\mw
    -IC:\Program Files\MATLAB71\rtw\c\tlc\lib
    -IC:\Program Files\MATLAB71\rtw\c\tlc\blocks
    -IC:\Program Files\MATLAB71\rtw\c\tlc\fixpt
    -IC:\Program Files\MATLAB71\stateflow\c\tlc
    -aEnforceIntegerDowncast=1
    -aFoldNonRolledExpr=1
    -aInlineInvariantSignals=0
    -aInlineParameters=0
    -aLocalBlockOutputs=1
    -aRollThreshold=5
```

```
        -aGenerateReport=0
        -aGenCodeOnly=1
        -aRTWVerbose=1
        -aIncludeHyperlinkInReport=0
        -aLaunchReport=0
        -aForceParamTrailComments=0
        -aGenerateComments=1
        -aIgnoreCustomStorageClasses=1
        -aIncHierarchyInIds=0
        -aMaxRTWIdLen=31
        -aShowEliminatedStatements=0
        -aPrefixModelToSubsysFcnNames=1
        -aIncDataTypeInIds=0
        -aInsertBlockDesc=0
        -aSimulinkBlockComments=1
        -aInlinedPrmAccess="Literals"
        -aTargetFcnLib="ansi_tfl_tmw.mat"
        -aGenFloatMathFcnCalls="ANSI_C"
        -aIsPILTarget=0
        -aIncludeMdlTerminateFcn=1
        -aCombineOutputUpdateFcns=1
        -aSuppressErrorStatus=0
        -aERTCustomFileBanners=1
        -aLogVarNameModifier="rt_"
        -aGenerateFullHeader=1
        -aGenerateSampleERTMain=0
        -aMatFileLogging=0
        -aMultiInstanceERTCode=0
        -aPurelyIntegerCode=0
        -aexportCCSObj=1
        -accsObjName="CCS_Obj"
        -auseDSPBIOS=0
        -aProfileGenCode=0
        -aInlineDSPBlks=1
        -aFPopt=0
        -amemModel="Near_Calls_and_Data"
        -aoptLevel="Function(-o2)"
        -aCompilerVerbosity="Quiet"
        -aInterruptThreshOnC6x=0
        -aInterruptThreshValue=""
        -aSymbolicDebugOnC6x=0
        -aRetainAsmFiles=0
        -aRetainObjFiles=1
        -aCreateMapFile=1
        -aUserStackSize=8192
        -ac6000BuildAction="Build_and_execute"
        -aOverrunAction="Notify_and_halt"
        -aOverrunNotificationMethod="Turn_on_LEDs"
        -aUserStackSize=8192
        -p10000
    ### Loading TLC function libraries
    ....
    ### Initial pass through model to cache user defined code
    .
    ### Caching model source code
    ..........
```

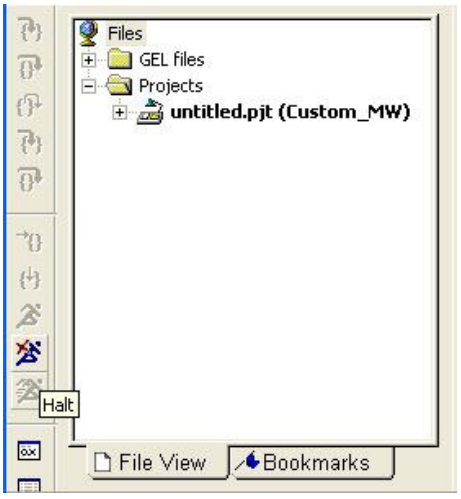
```
### Writing header file untitled_types.h
### Writing header file untitled.h
.
### Writing source file untitled.c
### Writing header file untitled_private.h
.
### Writing source file untitled_main.c
### TLC code generation complete.
### Creating project marker file: rtw_proj.tmw
### Wrapping unrecognized make command (angle brackets added)
###    <dummy>
### in default batch file
### Creating untitled.mk from C:\Program
Files\MATLAB71\toolbox\rtw\targets\tic6000\tic6000\ti_c6000_ert.tmf
Warning: uget_param is an obsolete function, use get_param instead
This warning can be turned off by issuing the following command at the matlab prompt:
warning('off','Simulink:uget_param_obsolete')
Warning: uset_param is an obsolete function, use set_param instead
This warning can be turned off by issuing the following command at the matlab prompt:
warning('off','Simulink:uset_param_obsolete')
Warning: The specified project 'C:\Program Files\MATLAB71\work\untitled_c6000_rtw\untitled.pjt' cannot be
closed because it does not exist.
### Creating project in Code Composer Studio(tm)
### Building Code Composer Studio(tm) project...
### Build complete
### Downloading COFF file
### Downloaded: untitled.out
### Build procedure complete.
```

Sedangkan pada pada Code Composer Studio akan mucul seperti berikut



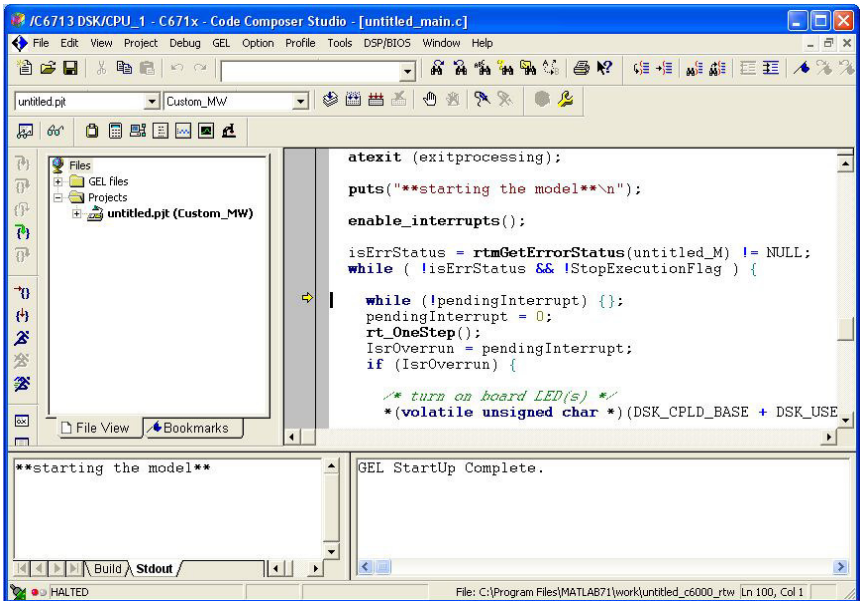
Gambar 27. Persiapan proses build program

22. Coba anda ambil microphone dan ucapkan kata-kata seperti check sound-3x, dan dengarkan apa yang muncul di Speaker Active.
23. Hentikan program yang sedang berjalan pada CCS, dengan cara tekan Halt.



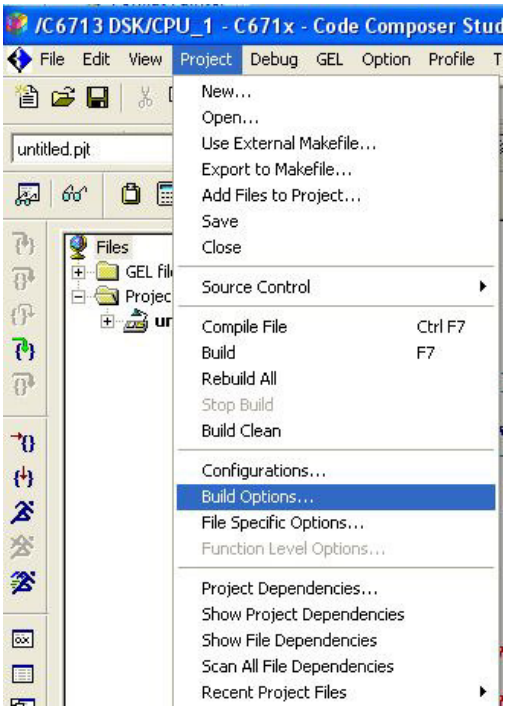
Gambar 28. Menghentikan jalannya program dari CCS

Perhatikan tampilan yang baru pada CCS, tampak disitu muncul listing program dari source code program anda.



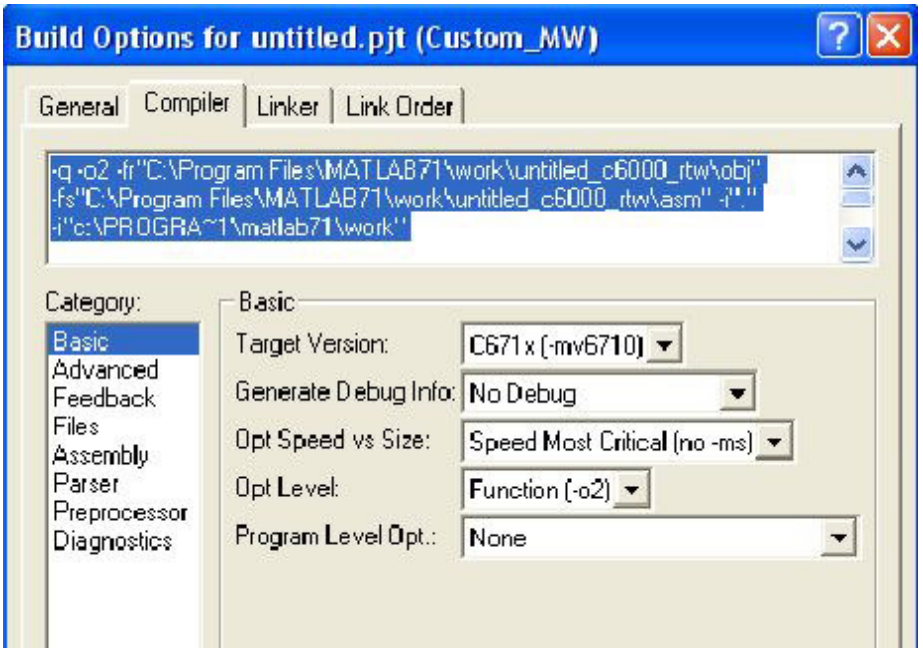
Gambar 29. Tampilan CCS setelah proses cross compilasi

24. Pada CCS anda Click pada Project, dan lanjutkan pada Build Option



Gambar 30. Langkah mengamati build option

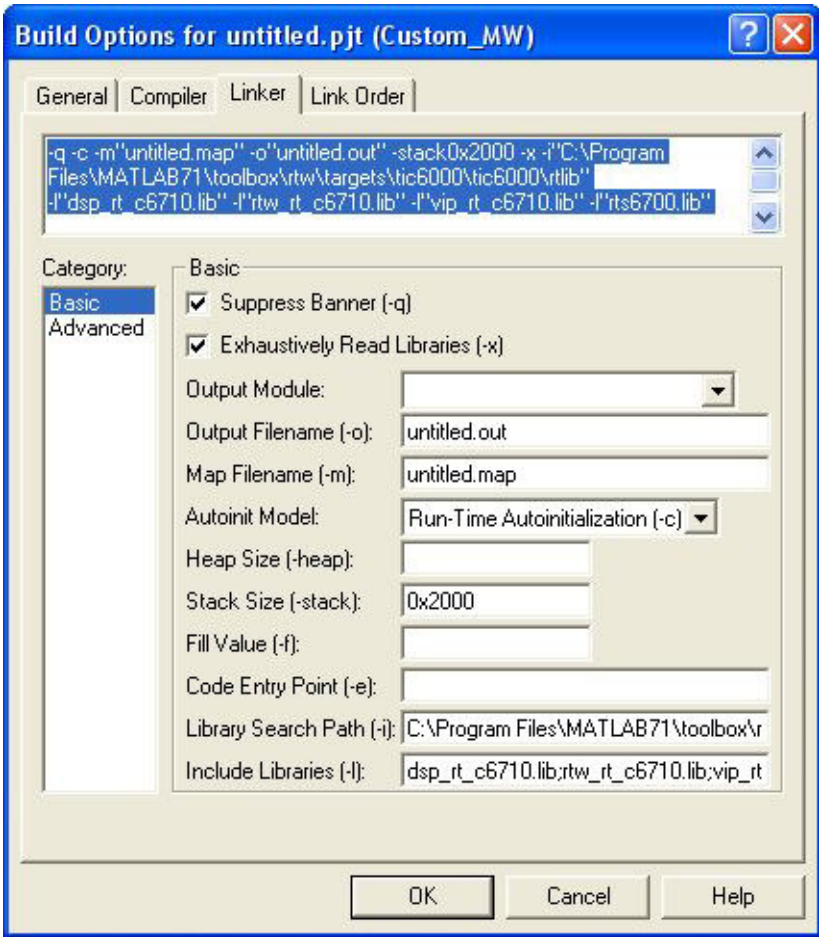
25. Perhatikan tampilan yang muncul anda click pada **Compiler**, lanjutkan click pada Category: **Basic**



Gambar 31. Tampilan build Option pada Category Basic

Lihat bagian teks yang ada background nya (high light) diatas.
C:\Program Files\MATLAB71\work\untitled_c6000_rtw\obj" -fs"C:\Program Files\MATLAB71\work\untitled_c6000_rtw\asm" -i"."
-i"c:\PROGRA~1\matlab71\work"
Ini menunjukkan bahwa proses ini dihasilkan dair kompilasi yang dilakukan melalui Matlab71, dan hasilnya disimpan dalam sub folder work pada Matlab.

26. Untuk sementara biarkan saja, dan anda coba click pada Linker, dan lanjutkan pada Category Basic. Anda cukup memperhatikan saja dan coba anda telaah, apa kaitannya keterangan yang tertulis pada seting kalimat yang ada highlight nya dengan seting yang anda lakukan.



Gambar 32. Seting lokasi Library search path dan Include Library pada Category Basic

-q -c -m"untitled.map" -o"untitled.out" -stack0x2000 -x -i"C:\Program Files\MATLAB71\toolbox\rtw\targets\tic6000\tic6000\rtlib" -l"dsp_rt_c6710.lib" -l"rtw_rt_c6710.lib" -l"vip_rt_c6710.lib" -l"rts6700.lib"

V. TUGAS

1. Anda cari referensi, dan berikan penjelasan apa arti seting dari build option yang ada pada CCS dari program yang dihasilkan oleh proses cross kompilasi model yang telah anda susun pada Matlab Simulink tersebut.

MODUL VI

PERANCANGAN FILTER DIGITAL DENGAN FDA TOOL

I. TUJUAN INSTRUKSIONAL

- Siswa mamahami penggunaan FDA Tool Matlab
- Siswa mampu merancang filter digital (FIR dan IIR) dengan FDA Tool
- Siswa mampu menganalisa filter hasil perancangan, dengan FDA Tool
- Siswa mampu mengeksport design hasil perancangan untuk implementasi filter

II. PENGENALAN PADA FILTER DESIGN AND ANALYSIS TOOL (FDATOOL)

Suatu *Filter Design and Analysis Tool* (FDATool) adalah suatu graphical user interface (GUI) yang sangat bermanfaat di dalam Signal Processing Toolbox untuk melakukan perancangan dan analisa filter.

FDATool memungkinkan bagai anda untuk melakukan perancangan filter FIR atau IIR dengan cepat melalui setting spesifikasi kinerja filter, dengan melakukan proses import filter dari MATLAB workspace anda atau melalui penambahan, pemindahan atau penghapusan pole dan zero. FDATool juga menyediakan perangkat untuk analisa filter, seperti penggambaran respon magnitude dan respon fase dan penggambaran pole-zero.

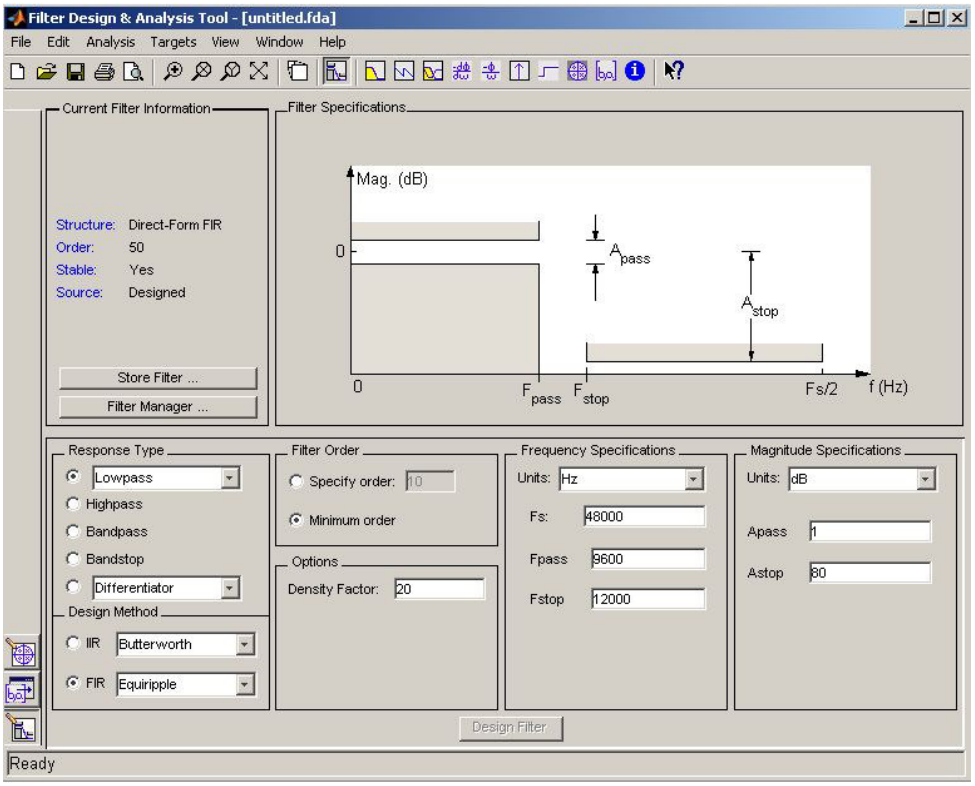
Anda dapat memanfaatkan FDATool sebagai suatu pilihan alternative untuk merancang filter menggunakan Matlab Command.

2.1. Cara Memulai

Pada Matlab command anda ketikkan ***fdatool*** seperti berikut:

>>***fdatool***

Suatu Tip pada Day dialog muncul dengan saran-saran untuk menggunakan FDATool. Kemudian, tampilan GUI muncul dengan suatu default filter.



Gambar 1. Tampilan pertama FDA Tools

GUI memiliki 3 bagian utama:

- Area Current Filter Information
- Area Filter Display region
- Design panel

Perhatikan baris pembatas tampilan diatas, bagian atas pada GUI menyajikan informasi spesifikasi filter dan respon filter yang sekarang digunakan. ***Area The Current Filter Information***, pada setengah tampilan bagian atas tersebut menyajikan sifat-sifat filter, penamaan struktur filter, orde, jumlah bagian yang digunakan dan keadaan kestabilan filter (stabil atau tidak). Bagian ini juga menyediakan akses pada ***Filter manager*** untuk bekerja dengan multiple filters.

Area Filter Display, pada bagian kanan atas, menyajikan beragam respon filter seperti respon magnitudo response, group delay dan koefisien-koefisien filter.

Setengah dari tampilan bagian bawah pada GUI adalah bagian interaktif pada ***FDATool***. Area ini adalah bagian **Design Panel**, disini anda bisa mendefinisikan spesifikasi filter anda. Ini akan mengontrol apda yang akan ditampilkan pada tampilan bagian atas tyang sudah dijelaskan diatas. Panel yang lain dapat disajikan di bagian yang lebih rendah dengan menggunakan ***sidebar buttons***.

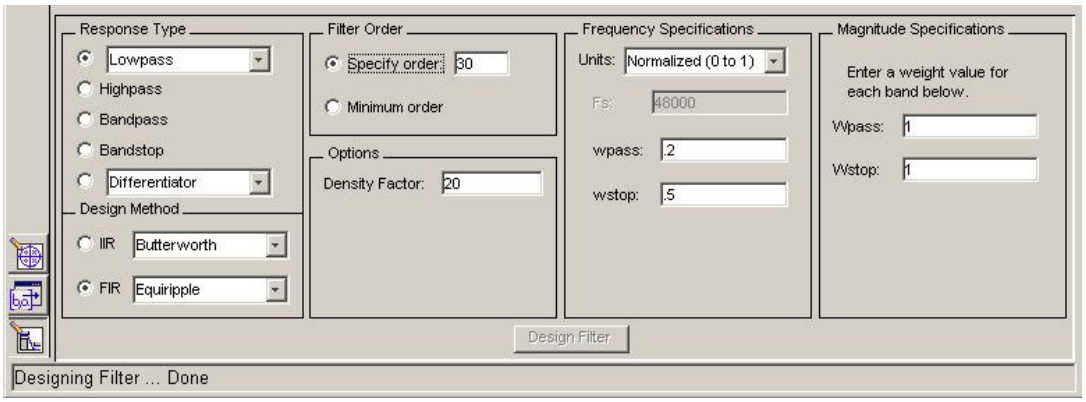
Perangkat ini jugamenyediakan suatu ***Context-sensitive help***. Anda dapat melakukan click-kanan atau click pada tombol ***What's This?*** Untuk mendapatkan informasi pada bagian berbeda pada perangkat ini.

2.2. Perancangan Suatu Filter

Kita akan merancang suatu low pass filter yang meloloskan semua frekuensi yang lebih kecil atau sama dengan dari dibawah 20% dari frekuensi Nyquist (setengah dari sampling frequency) dan akan meredam frekuensi-frekuensi yang lebih tinggi atau sama dengan 50% dari frekuensi Nyquist. Kita akan menggunakan sebuah FIR Equiripple filter dengan spesifikasi seperti berikut:

- Passband attenuation 1 dB
- Stopband attenuation 80 dB
- Passband frequency 0.2 [Normalized (0 to 1)]
- Stopband frequency 0.5 [Normalized (0 to 1)]

Untuk mengimplementasikan rancangan ini, kita akan menggunakan sepesifikasi seperti pada Gambar 2 berikut:

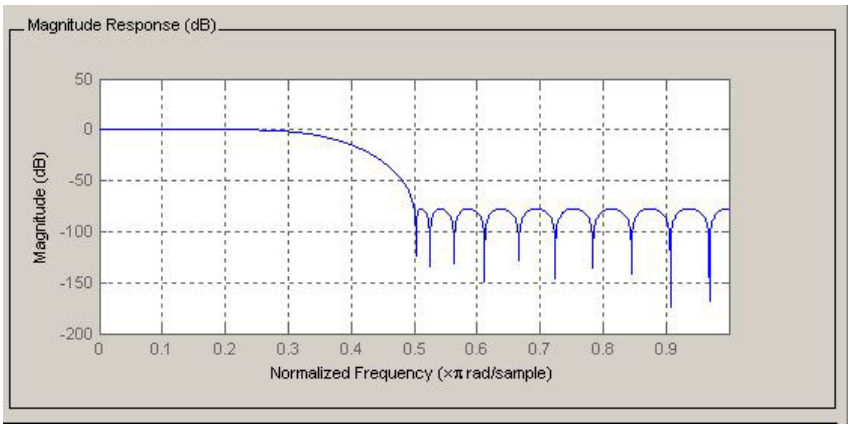


Gambar 2. Contoh Spesifikasi FIR filter

Langkah-langkahnya adalah sbb:

1. Pilih **Lowpass** dari menu dropdown dibawah **Response Type**, dan **Equiripple** dibawah **FIR Design Method**. Secara umum, ketika anda merubah **Response Type** atau **Design Method**, parameter-parameter filter dan area **Filter Display** akan ter-update secara otomatis.
2. Pilih **Specify order** pada area **Filter Order** dan masukkan 30.
3. **FIR Equiripple filter** memiliki suatu **Density Factor Option** yang mana mengontrol density (kerapatan) pada grid frekuensi. Peningkatan nilai yang dihasilkan suatu filter akan lebih mendekatkan pada suatu sifat ideal equiripple filter, tetapi akan menyebabkan proses komputasi yang diperlukan untuk eksekusi juga menjadi lebih panjang. Tetapkan nilai ni pada 20.
4. Pilih **Normalized** (0 to 1) yang ada di dalam menu pull down **Units** dalam area **Frequency Specifications**.
5. Masukkan 0.2 untuk wpass dan 0.5 untuk nilai wstop di dalam area **Frequency Specifications**.
6. Wpass dan Wstop, di dalam area Magnitude Specifications adalah bobot positif, digunakan sepanjang optimisasi FIR Equiripple filter.
7. Setelah setting spesifikasi perancangan, click pada tombol Design Filter pada bagian bawah GUI untuk perancangan filter.

Respon magnitudo pada filter ditampilkan di dalam area Filter Analysis setelah koefisien-koefisien ini dikomputasi.



Gambar 3. Respon Magnitudo FIR hasil perancangan

2.3. Pengamatan untuk Analisa Lain

Pertama kali anda merancang sebuah filter, anda dapat mengamati beberapa tampilan filter berikut ini yang ditampilkan pada suatu window, dengan cara melakukan click pada salah satu button yang ada di toolbar:

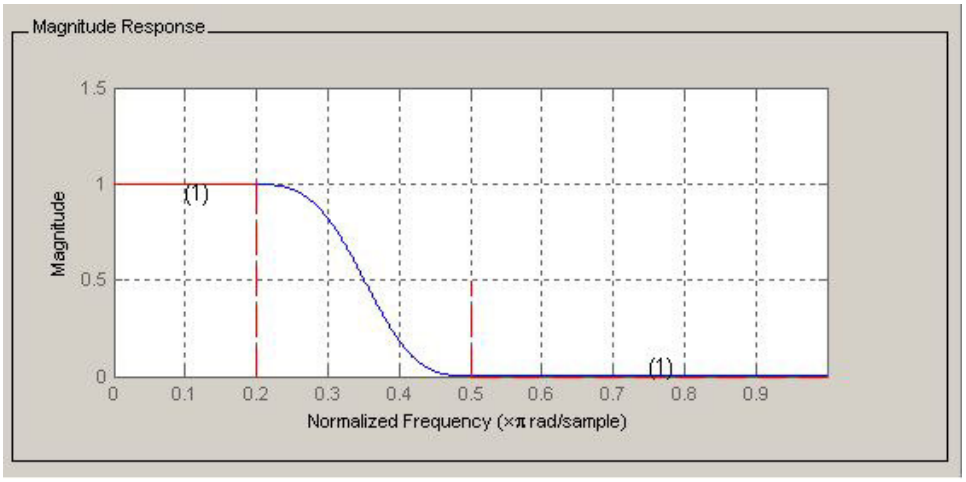
Gambar 4. Button yang ada di Toolbar Analisa.

Dari kiri ke kanan, tombol-tombol tersebut adalah:

- Magnitude response
- Phase response Magnitude and Phase responses
- Group delay response
- Phase delay response
- Impulse response
- Step response
- Pole-zero plot
- Filter Coefficients
- Filter Information.

2.4. Pembandingan Perancangan untuk Spesifikasi Filter

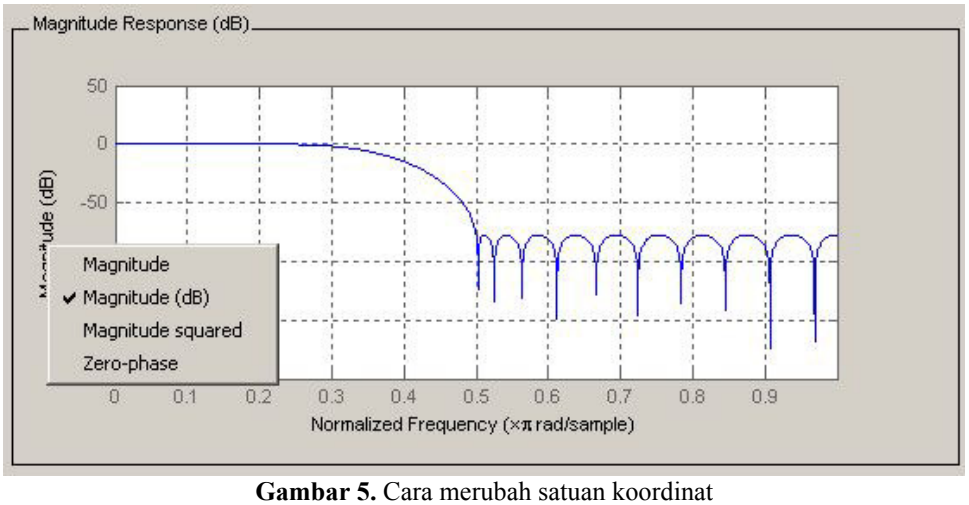
FDATool memungkinkan bagi anda untuk mengukur seberapa dekat hasil perancnagan anda dengan spesifikasi filter dnegna menggunakan Specification yang mana akan menyajikan spesifikasi filter dalam bentuk plot (gambaran) responnya. Di dalam Display Region, ketika gambaran Magnitude ditampilkan, pilih Specification Mask dari menu View untuk menampilkan spesifikasi filter pada gambaran responsenya. Respon magnitudo pada filter dengan Specification mask ditunjukkan berikut ini.



Gambar 4. Respon magnitudo pada filter

2.4.1. Perubahan Axes Units (Satuan Koordinat)

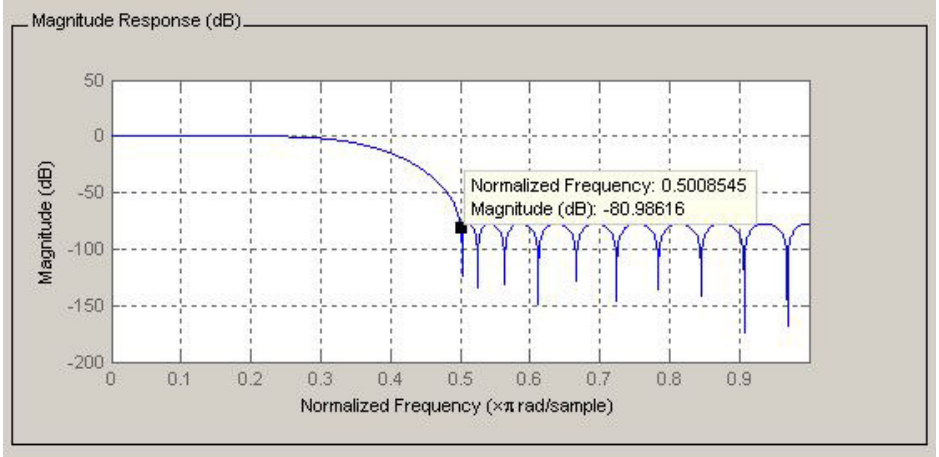
Anda dapat merubah satuan pada sumbu x atau y dengan cara menekan click-kanan pada mouse anda pada suatu axis label dan memilih satuan tertentu. Satuan berikut ini bisa dimiliki dengan checkmark (pemberian tanda centang).



Gambar 5. Cara merubah satuan koordinat

2.4.2.Marking Data Points

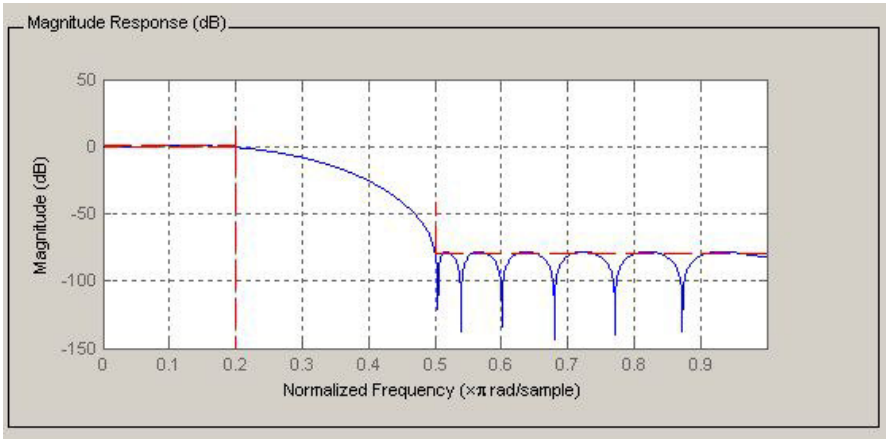
Di dalam area Display, anda click pada suatu titik di dalam gambaran (plot) untuk menambah suatu **data marker**, yang mana akan menampilkan nilai pada titik tersebut. Click-kanan pada tampilan **data marker**, suatu menu akan tampil dimana anda menggerakkan mouse, anda bisa berpindah, men- delete atau mengatur kemunculan data markers tersebut.



Gambar 6. Marking Data Point

2.5. Optimisasi Perancangan

Untuk meminimisasi biaa dalam implementasi filter, kita akan mencoba untuk mereduksi jumlah koefisien dengan menggunakan opsi **Minimum Order** di dalam design panel. Anda rubah pemilihan pada **Filter Order** dan pindahkan **Minimum Order** di dalam **Design Region** dan biarkan parameter yang lain. Click pada **Design Filter** button untuk merancang satu filter baru.



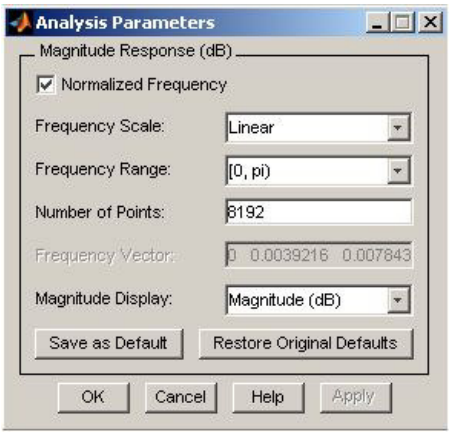
Gambar 7. Hasil perancangan optimisasi filter

Seperti yang anda lihat di dalam are **Current Filter Information**, order filter berkurang

dari 30 menjadi 16, jumlah pada **ripples** menurun dan **transition width** menjadi lebih lebar. Spesifikasi passband dan stopband masih bisa diperoleh dari kriteria perancangan ini.

2.6. Perubahan Parameter Analisis

Dengan click-kanan pada **plot and selecting Analysis Parameters**, anda dapat menampilkan suatu **dialog box** untuk suatu perubahan **analysis-specific parameters.** (Anda dapat juga memilih Analysis Parameters dari menu Analysis).



Gambar 8. Menu Analisys Paramter

Untuk menyipan parameter-parameter seperti pada nilai default, click **Save as Default**. Untuk menampilkan kembali nilai default **MATLAB-defined**, click **Restore Original Defaults**.

2.7. Meng-Export Filter

Jika anda telah benar-benar puas dengan filter hasil perancangan anda, anda dapat mengeksport filter anda dengan tujuan berikut ini:

- MATLAB workspace
- MAT-file
- Text-file

Untuk melakukan export file dari FDT Tool langkahnya adalah pilih **Export** dari menu File.

1. Jika export ke MATLAB workspace, anda dapat melakukan export sebagai **coefficients** atau **as an object** dengan memilih dari menu **pulldown** yang ada di dibawah **Export As**.
2. Jika anda ingin meng- export **as an object**, property object mengontrol beberapa aspek pada tampilan dan perilakunya.

Gambar 9. Menu Export
a) Export As Coefficient
b) Export As Object

Anda dapat menggunakan perintah GET and SET dari MATLAB command prompt untuk mengakses dan memanipulasi nilai properti pada object.

2.7.1. Pembangkitan suatu M-file

FDATool memungkinkan bagi anda untuk membangkitkan M-code untuk meng-create ulang filter anda. Hal ini member kesempatan bagi anda untuk membenamkan (embed) hasil rancangan anda ke dalam existing code atau secara otomatis untuk membuat filter anda dalam suatu script.
Pilih Generate M-file dari menu File dan spesifikasikan nama file dalam kotak dialog Generate M-file.

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:

- PC yang memiliki spesifikasi memory minimal 1 Gb
- Operating System minimal Windows Xp
- Perangkat Lunak Matlab
- Sistem Audio untuk PC

IV. PERCOBAAN

4.1. Design FIR Windowing

1. Rancang sebuah FIR filter, dengan orde 16, frekuensi sampling 10000 Hz, dan frekuensi cut off 2000 Hz. Metode perancangan menggunakan window hamming. Karakteristik respon frekuensi adalah low pass filter.

2. Dapatkan gambaran bentuk Magnitude Response, Phase Response, Magnitude and Phase Response, Impulse Response, Step Response, Pole/zero Plot, dan Filter Coefficients.

3. Catat daerah -3 dB pada Magnitude Reponse, dan perhatikan nilai ini tepat pada frekuensi berapa pada sumbu mendatar. Jika waktu anda menekan click posisinya tidak pada -3 dB, anda bisa menahan dan menggesernya, sehingga anda semakin dekat dengan posisi -3 dB.

4. Lakukan langkah 1 dan 2 untuk HPF dan BPF (2000 - 4000).

4.2. Design FIR Least Square

1. Rancang sebuah FIR filter, dengan orde 16, frekuensi sampling 10000 Hz, dan frekuensi pass (Fpass) 3000 Hz, dan frekuensi stop (Fstop) = 3500 Hz. Metode perancangan menggunakan equiripple. Karakteristik respon frekuensi adalah low pass filter.
2. Dapatkan gambaran bentuk Magnitude Response, Phase Response, Magnitude and Phase Response, Impulse Response, Step Response, Pole/zero Plot, dan Filter Coefficients.
3. Catat daerah -3 dB pada Magnitude Reponse, dan perhatikan nilai ini tepat pada frekuensi berapa pada sumbu mendatar
4. Lakukan langkah 1 dan 2 untuk HPF dan BPF (Fstop=1500, Fpass=2000 pada bagian low frequency, dan Fstop=4000, Fpass=4500 pada bagian high frequency)

4.3. Design IIR

1. Rancang sebuah IIR filter, dengan orde 16, frekuensi sampling 10000 Hz, dan frekuensi cut off 2000 Hz. Metode perancangan menggunakan analog prototype Butterworth. Karakteristik respon frekuensi adalah low pass filter.
2. Dapatkan gambaran bentuk Magnitude Response, Phase Response, Magnitude and Phase Response, Impulse Response, Step Response, Pole/zero Plot, dan Filter Coefficients.
3. Usahakan posisi pengamatan pada respon frekuensi (Magnitude Response), pada bagian kiri bawah anda cari toolbar Pole/zero Editor dan click. Anda perhatikan tampilan layar monitor FDA Tool, selanjutnya click pada suatu pole dan gerakkan posisinya, perhatikan perubahan yang terjadi pada Magnitude Response. Dengan cara yang sama anda click pada zero, dan gerakkan posisinya, amati perubahan pada Magnitude Response.
4. Lakukan langkah 1 dan 2 untuk Analog Prototipe Chebychev, dan Elliptical.

V. TUGAS DAN ANALISA

1. Dari semua percobaan yang anda lakukan dengan FDA Tool, buat analisa respon frekuensi pada filter FIR dan IIR.
2. Untuk filter IIR yang anda rancang, jelaskan pengaruh perubahan possisi pole/zero pada respon frkeuensi (Magnitude Reponse) yang dihasilkannya.

MODUL VII

PERANCANGAN DAN IMPLEMENTASI FILTER FIR DENGAN MATLAB

I. TUJUAN INSTRUKSIONAL

- Siswa mamamahi konsep perancangan sebuah FIR filter
- Siswa malakukan analisa kinerja filter dalam domain frekuensi
- Siswa dapat melakukan proses pemfilteran sinyal audio

II. KONSEP PERANCANGAN FIR FILTER

Tujuan perancangan filter adalah untuk menghasilkan sebuah system yang mampu memberikan perlakukan khusus pada data (sinyal input) sesuai dengan spesifikasi frekuensi yang ditetapkan. Contohnya dalah bagaimana menghilangkan komponen frekuensi diatas 30 Hz pada suatu sinyal yang disampel dengan frekuensi sampling sebesar 100Hz.

2.1. Low Pass Filter Ideal

Sebuah low pass filter ideal adalah satu filter yang meloloskan semua komponen frekuensi dibawah frekuensi cut-off (ω_c)dari sinyal input, dan menghentikan semua komponen frekuensi diatas frekuensi cut-off (ω_c). Dalam domain frekuensi filter ideal ini dinyatakan sebagai:

$$H_{LP}(e^{j\omega}) = \begin{cases} 1, & 0 \le \omega \le \omega_c \\ 0, & \omega_c \le \omega \le \pi \end{cases} \quad H_{LP}(e^{j\omega}) = \begin{cases} 1, & 0 \le \omega \le \omega_c \\ 0, & \omega_c \le \omega \le \pi \end{cases} \tag{1}$$

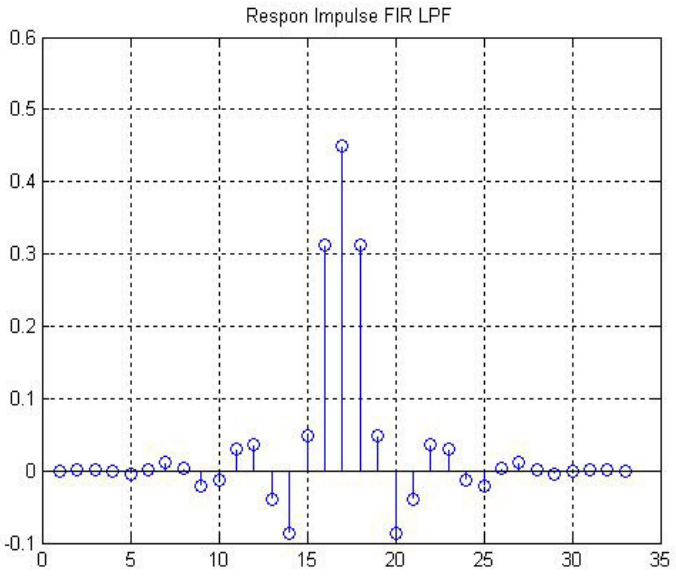
Di mana:
- $H_{LP}(e^{j\omega})$ adalah low pass filter
- ω adalah frekuensi kerja dalam satuan radiant
- ω_c adalah frekuensi cut off

Di dalam domain waktu, bentuk filter diatas dinyatakan dalam bentuk impulse response ideal lowpass filter, dan dapat dinyatakan sebagai

$$h_{LP}[n] = \frac{sin(\omega_c n)}{\pi n} \qquad -\infty < n < \infty$$

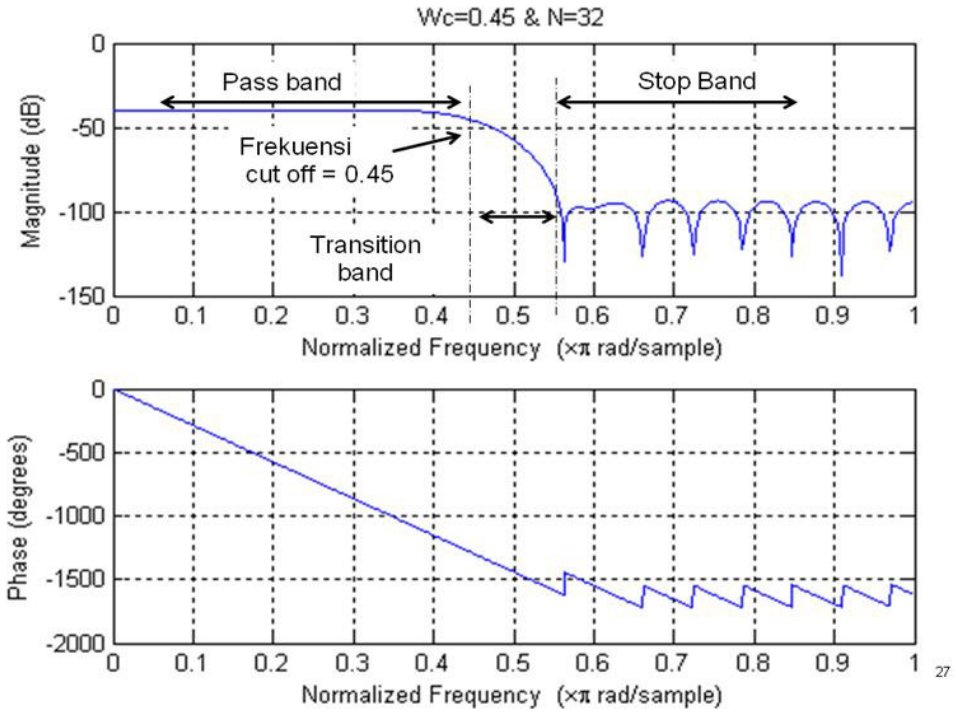
$$h_{LP}[n] = \frac{sin(\omega_c n)}{\pi n} \qquad -\infty < n < \infty \tag{2}$$

Persamaan diatas juga dikenal sebagai ***raised cosine***.



Gambar 1. Respon Impulse FIR Filter

Respon Frekuensi sebuah Low Pass Filter yang disusun dengan mengacu bentuk filter digital FIR memiliki bentuk seperti Gambar 2 di bawah ini. Bagian Pass Band adalah daerah dimana komponen frekuensi diloloskan, pada daerah transition band komponen frekuensi sinyal input mulai mengalami redaman, dan pada daerah Stop Band komponen frekuensi sinyal input mengalami redaman yang sangat besar, sehingga dengan kata lain sinyal dihentikan.



Gambar 2. Respon Frekuensi LPF FIR Filter

2.2. Filter FIR

Suatu *finite impulse response* **(FIR)** filter merupakan suatu tipe pada digital filter. Pengertian impulse response, bagaimana respon liter terhadap input, terminologi 'finite' dipahami sebagai penetapan output sistem diperhitungkan pada sample intervals yang diawali dari nol sampai dengan angka tertentu.

Berbeda dengan terminologi IIR filter, yang mana nilai ouput di waktu sebelum mulainya sitem (<0) sudah diperhitungkan dengan adanya parameter feedback internal dari sistem tersebut.

FIR Filter bisa dinyatakan dalam sebuah persamaan beda yang merepresentasikan hubungan output/input sistem LTI sebagai berikut:

(3)

Di mana:
M = orde filter
k = filter indek
b_k = filter coefficients

Di dalam realisasinya sebuah FIR filter bisa didekati dengan bentuk blok diagram standar seperti berikut.

Gambar 3. Diagram Blok FIR Filter

2.3. Perancangan FIR dengan metode Windowing

Matlab telah menyediakan sebuah fungsi **`fir1()`** yang dapat digunakan untuk merancang linear-phase FIR filter konvensional seperti lowpass, highpass, bandpass, dan bandstop didasarkan pada metode windowing. Dengan menggunakan perintah **`b = fir1(N,Wn)`** akan menghasilkan sebuah vector **b** yang merupakan respon impulse response pada suatu ***lowpass filter*** dengan order senilai **N**. Nilai frekuensi ***cut-off*** (**`Wn`**) harus bernilai atara 0 sampai 1 yang mana mengacu pada suatu nilai berkaitan berkaitan dengan nilai setengah dari frekuensi sampling (*f*s), yang dalam beberapa literature juga dinyatakan sebagai *sampling rate*. NIlai frekuensi cut off ini merupakan bentuk ternormalisasi dari *f*s/2.

Untuk menghasilkan sebuah bentuk filter yang lain, misalnya ***high pass filter***, kita harus memodifikasi dengan satu perintah tambahan seperti berikut **`b = fir1(N,Wn,'high').`** Perintah ini akan menghhasilkan sebuah respon impulse ***high pass filter*** dengan order N, dan sebuah frekuensi ***cut-off*** ternormalisasi **`Wn`**. Dengan cara yang hampir sama, kita bisa menyusun perintah, `b = fir1(N,Wn,'pass')`, untuk menghasilkan sebuah ***band pass filter***, yang mana dalam hal ini **`Wn`** memiliki dua-element vektor untuk menentukan nilai ***passband*** pada ***band pass*** **filter**. Dan untuk ***band stop filter*** kita bisa melakukannya dengan **`b = fir1(N,Wn,'stop').`**

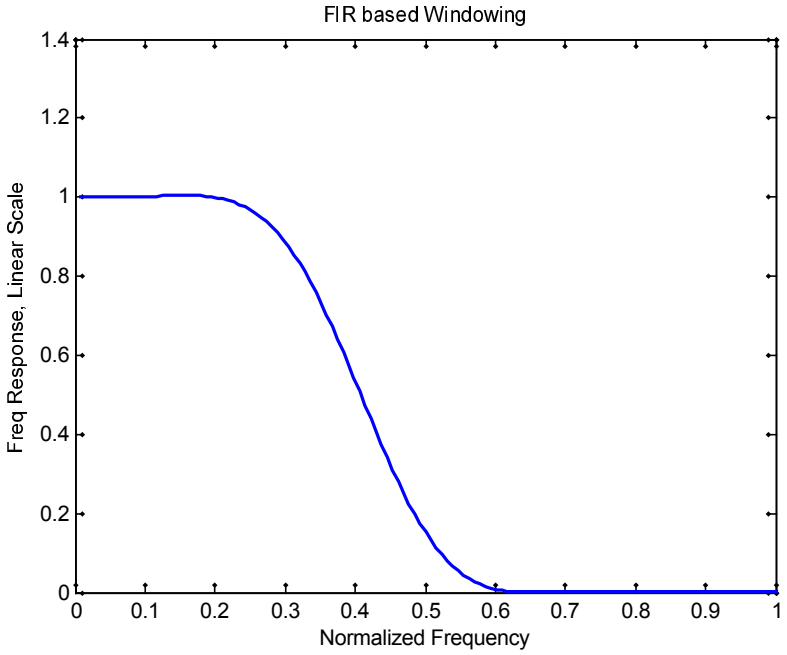
Jika kita tidak menyebutkan jenis window yang digunakan, maka secara default, Matlab akan menentukan semua perintah diatas bahwa window yang digunakan adalah time ***window Hamming***. Jenis fungsi window yang lain dapat digunakan dengan cara menyebutkan secara spesifik jenis window yang digunakan dalam merancang FIR filter. Misalnya untuk window Blackman, dapat dilakukan dengan cara sebagai berikut **`b = fir1(N, Wn, blackman(N))`**.

Pada contoh program berikut ini disajikan sebuah cara merancang FIR low pass filter dengan menggunakan jenis window hamming. Dalam hal ini nilai orde filternya adalah 16, dan frekuensi cut off sebesar 0.4.

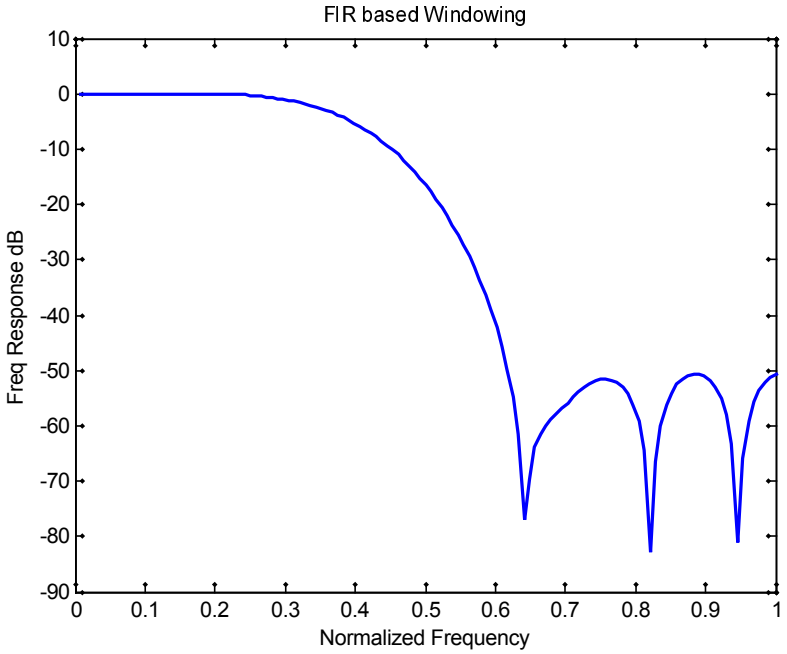
```
%file name: FIR_design_02.m
%menggunakan metode windowing dg memanfaatkan standar Matlab fir1
clear all;
N=16;
w_c=0.4;
b = FIR1(N,w_c);
bb=fft(b,256);
ff=1:length(bb)/2;
plot(2*ff/length(bb),abs(bb(1:length(bb)/2)),'linewidth',2)
xlabel('Normalized Frequency');
ylabel('Magnitude Linear Scale');
title('FIR based Windowing');
```

Prosedur perancangan diatas akan menghasilkan sebuah low pass filter yang memiliki respon frekuensi seperti pada Gambar 4. Bentuk yang disajikan dalam respon frekuensi ini adalah dalam skala linear, belum dalam bentuk terskala decibel. Jika anda ingin melakukan perubahan untuk menghasilkan bentuk skala decibel, anda harus memodifikasi pada nilai

respon frekuensinya dalam skala logaritmik.



Gambar 4. Respon Frekuensi FIR Filter dengan Windowing, dalam skala linear



Gambar 5. Respon Frekuensi FIR Filter dengan Windowing, dalam skala dB

Respon frekuensi FIR yang dihasailkan menunjukkan bahwa pada nilai frekuensi cut off, dalam hal ini ditunjukkan dengan nilai w = 0.4, mulai menunjukkan efek redaman terhadap sinyal input, sehingga outputnya mengalami pelemahan nilai. Pada nilai w = 0.6, sinyal

benar-benar mengalami peredaman sampai nol dan demikian seterusnya nilai redaman menghasilkan sinyal output yang dilemahkan sampai mendekati nilai nol. Sedangkan dalam skala logaritmik, hasilnya menunjukkan adanya ripple pada frekuensi diatas w = 0,6.

2.4. Perancangan FIR dengan Parks-McClellan

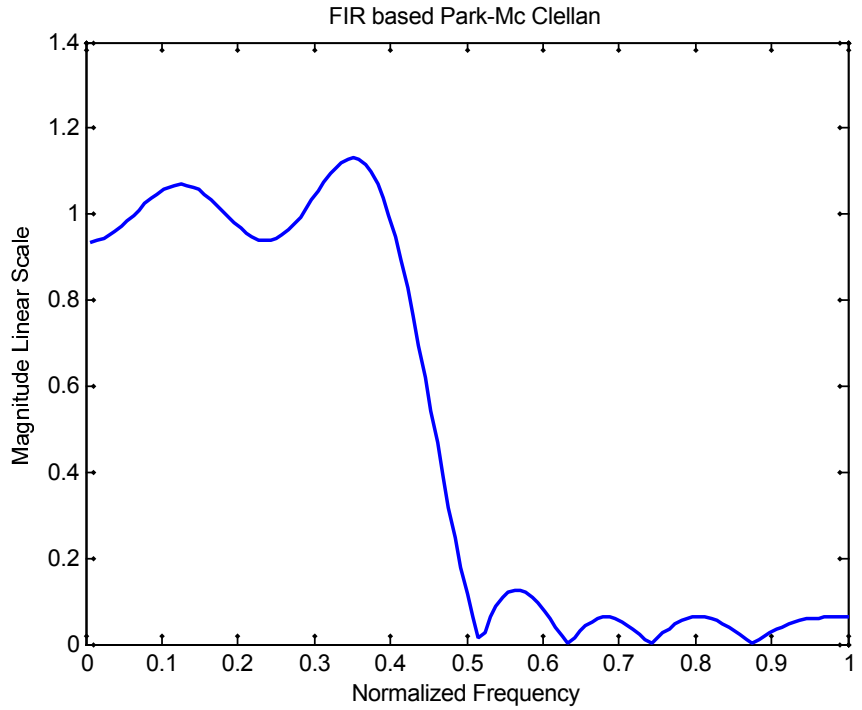
Pada perangkat lunak Matlab juga telah terdapat berbagai contoh fungsi funtuk perancangan FIR filter yang merupakan hasil pengembangan dari berbagai algorimta. Salah satu diantara fungsi tersebut adalah yang dibentuk didasarkan pada algorimta Park-McClelland. Bentuk sintaknya cukup sederhana seperti berikut: **`b = remez(N,F,A)`**. Dalam hal ini, **b** merupakan sebuah respon impulse sepanjang **`N+1`** dari sebuah filter FIR dengan order N, yang memiliki karakteristik respon phase linear. Variabel **F** adalah suatu vector yang merepresentasikan nilai-nilai band frekuensi terskala terhadap frekuensi sampling (***f*s/2**), bernilai antara 0 sampai 1. Sehingga dalam hal ini **F = 0** akan merepresentasikan nilai **0** Hz, dan **F = 1** berkaitan dengan suatu nilai frekueni linear sebesar ***f*s/2**. Variable **A** adalah suatu vector dengan nilai real vector yang merepresentasikan nilai magnitudo sebagai fungsi dari nilai frekuensi pada vector **F**. Antara F Dan A, merupakan sebuah pasangan bentuk respon frekuensi yang disederhanakan sehingga bisa juga dituliskan sebagai **`(F(k),A(k))`** untuk nilai k genap dan **`(F(k+1),A(k+1))`** untuk nilai k ganjil. Untuk nilai **`k`** ganjil, bandwidth diantara **`F(k+1)`** dan **`F(k+2)`** ditandai sebagai suatu daerah ***transition bands.***

Untuk lebih memudahkan pemahaman, berikut ini kita akan merancang sebuah filter FIR yang memiliki karakteristik low pass filter, dengan nilai-nilai yang sama dengan pada kasus perancangan menggunakan teknik ***windowing***. Disini nilai frekuensi cut off ditetapkan wc=0,4, dan orde filter ditetapkan N =16.

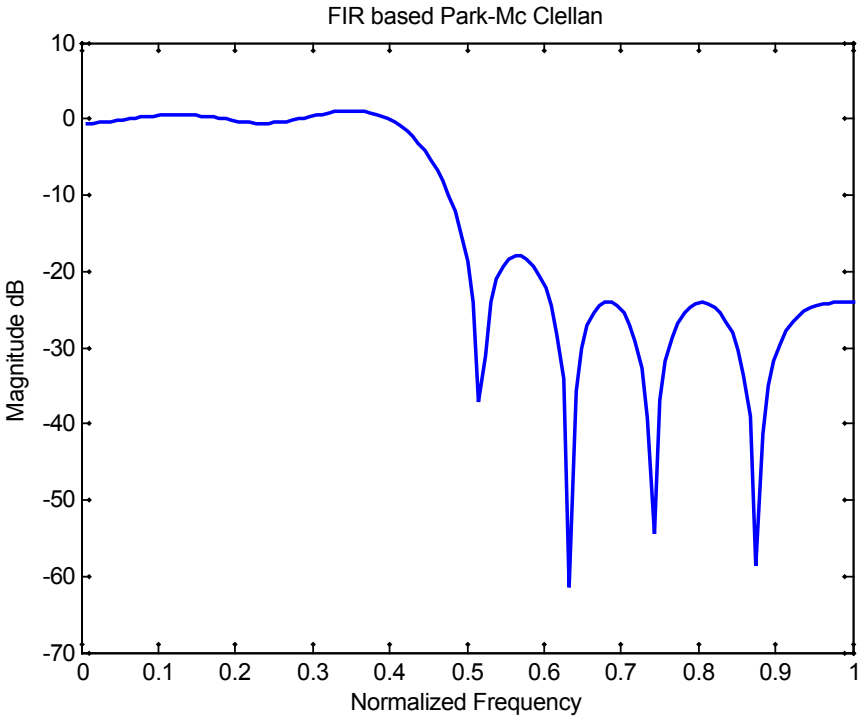
```
clear all;
N=16;
F=[0 .1 .2 .3 .4 .5 .6 .7 .8 1.0];
A=[1 1 1 1 1 0 0 0 0 0];
b=remez(N,F,A);
bb=fft(b,256);
figure(1);
ff=1:length(bb)/2;
plot(2*ff/length(bb),abs(bb(1:length(bb)/2)),'linewidth',2)
xlabel('Normalized Frequency');
ylabel('Magnitude Linear Scale');
title('FIR based Park-Mc Clellan');
```

Program diatas menghasilkan sebuah FIR filter dengan respon frekuensi terskala linear. Untuk menghasilkan sebuah FIR filter terskala logarithmic dalam satuan dB diperlukan modifikasi perintah sebagai berikut:

```
figure(2);
plot(2*ff/length(bb),20*log10(abs(bb(1:length(bb)/2))),'linewidth',2)
xlabel('Normalized Frequency');
ylabel('Magnitude dB');
title('FIR based Park-Mc Clelland');
```



Gambar 6. Respon Frekuensi FIR Filter dengan design Park-Mc Clelland



Gambar 7. Respon Frekuensi FIR Filter dengan design Park-Mc Clelland terskala dB

III.PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:
- PC yang memiliki spesifikasi memory minimal 1 Gb
- Operating System minimal Windows Xp
- Perangkat Lunak Matlab
- Sistem Audio untuk PC

IV. PERCOBAAN

4.1. Merancang FIR filter dengan Teknik Windowing

1. Pada langkah ini anda diharuskan merancang sebuah filter yang meloloskan semua frekuensi ternormalisasi kurang dari atau sama dengan 0.45, dan meredam semua komponen frekuensi yang lebih besar dari 0.45 (frekuensi ***cut-off***, wc=0,45). Dalam hal ini anda tentukan order filter sebesar 32, dan perancangan yang anda lakukan berbasis pada teknik windowing.

2. Amati bentuk respon impulse dan respon magnitude dan fase sebagai fungsi frekuensi (respon frekuensi). Anda harus mampu menggambarkan bentuk respon frekuensi di dalam skala dB.
3. Cari titik dimana nilai magnitudonya mengalami peredaman sampai -3 dB, dapatkan nilai frekuensinya. Dan beri tanda daerah ***pass band***, ***transition band***, dan ***stop band*** dari LPF yang telah anda rancang tersebut.
4. Ulangi langkah 1 sampai 3 untuk kasus HPF dimana, frekuensi ***cut-off*** dalam hal ini bernilai 0.25.
5. Untuk langkah ini anda harus merancang sebuah band pass filter yang akan digunakan untuk meloloskan komponen frekuensi (ternormalisasi) antara 0,25 sampai dengan 0,55. Amati bentuk respon frekuensinya, dan dapatkan nilai frekuensi pada saat redaman magnitudonya senilai -3dB. Anda tarik garis dari posisi ***cut off*** rendah (-3 dB pertama) sampai dengan cut-off tinggi (-3dB kedua) pada BPF yang telah anda rancang.

Di sini hal ini anda harus mengambil gambar respon frekuensi dari filter tersebut, dan menandai lokasi-lokasi yang diminta dalam petunjuk. Anda sebaiknya melakukannya secara manual untuk lebih mudah memahami perintah-perintah diatas.

4.2. Merancang FIR filter dengan Park Mc Clelland

1. Anda tetapkan pemetaan frekuensi untuk filter anda, misalnya F=[0 .15 .25 .35 .45 .55 .65 .75 .85 1.0]; Usahakan komponen penyusun vector F selalu genap, misalnya 6, 8, 10, dsb. Untuk mendapatkan sebuah LPF, tetapkan vector A yang merepresentasikan pemetaan magnitude pada filter yang anda rancang, dalam hal ini tetapkan A bernilai 1 untuk 6 komponen pertama, dan bernilai 0 untuk komponen sisanya. Dalam hal ini jumlah

komponen A harus sama dengan komonen pada F.
2. Dapatkan koefisien-koefisien filter LPF anda, dengan fungsi '***remez***' yang mendukung algorithma Park-Mc Clelland dengan order filter N=16.
3. Berikan gambaran respon frekuensinya (magnitude dan fase sebagai fungsi perubahan frekuensi).
4. Ulangi langkah 1 sampai 3 untuk mendapatkan sebuah ***high pass filter*** (HPF) dengan frekuensi cut off wc=0.25.
5. Ulangi langkah 2 sampai 3 untuk mendapatkan sebuah ***band pass filter*** (BPF) yang memiliki daerah ***pass band*** dari 0.25 sampai 0.55.

Untuk langkah 1 sampai 5 pada bagian ini anda harus membandingkannya dengan prosedur perancangan yang telah anda susun dengan metode windowing. Anda amati perbedaan respon frekuensi kedua metode tersebut untuk satu jenis filter yang sama, misalnya untuk LPF pada teknik window anda bandingkan dengan LPF pada Park-Mc Clelland.

4.3. Pemfilteran Sinyal Sederhana

Untuk langkah berikutnya, anda bisa memiliki dengan Windowing atau Park-Mc Clelland sesuai dengan selera anda.
1. Bangkitkan sebuah sinyal sinus yang memiliki amplitude A=1, frekuensi f=200, dan sampling rate yang digunakan adalah 2000 Hz.
2. Bangkitkan sinyal sinus kedua, dengan amplitude A2= 0.25, frekuensi f2=500 Hz, dengan sampling rate yang sama dengan langkah 1. Lakukan penjumlahan sinyal pertama dengan sinyal kedua, sinyal_out = sinyal1 + sinyal2.

Gambar 8. Proses pemfilteran pada sinyal sinus

3. Lakukan proses pemfilteran dengan menggunakan LPF yang telah dibuat pada langkah percobaan 4.1.
4. Amati bentuk sinyal input pertama, sinyal input kedua, sinyal hasil jumlahan dalam domain waktu dan domain frekuensi.
5. Amati bentuk sinyal setelah proses pemfilteran dalam domain waktu dan domain frekuensi.
6. Anda bangkitkan sinyal nois Gaussian yang dalam hal ini panjangnya sama dengan vector yang anda gunakan untuk menyusun sinyal pada langkah 1 pada percobaan 4.3 ini.
7. Anda kalikan nilai nois terbangkit dengan 0.2 (WGN = mean + var*random_Gauss, dalam hal ini mean = 0,0).

8. Jumlahkan nois ini dengan sinyal sinus (sinyal pertama) yang anda miliki, sinyal_ nois = sinyal + nois.
9. Lakukan pemfilteran dengan LPF seperti yang telah anda gunakan pada langkah 3, pada percobaan 4.3 ini.
10. Amati gambaran sinyal sinus, sinyal_nois, dan sinyal output hasil proses pemfilteran dalam domain waktu dan domain frekuensi.

Gambar 9. Proses pemfilteran pada sinyal sinus

4.4. Pemfilteran Sinyal Audio I

Pada percobaan ini anda akan menggunakan sinyal Audio untuk menguji kemampuan anda dalam menyusun sebuah filter digital dan memahami cara kerjanya.

1. Lakukan pengambilan sinyal audio dengan memanfaatkan fungsi ***wavread*** yang ada pada Matlab, misalnya file ***a.wav*** atau file lain yang memiliki karakteristik mono (bukan stereo).
2. Lakukan proses resample sehingga sampling rate yang baru bernilai 10000.
3. Ambil satu frame, kurang lebih 20 ms (ekuivalent dengan 0,02*sampling rate), dan sajikan gambaran sinyal tersebut dalam domain waktu dan domain frkeuensi.
4. Bangkitkan sebuah noise Gaussian dengan vector sepanjang sinyal audio, tentukan nilai varians noise sebesar 0.2 , → nois=var*nois_terbangkit.
5. Lakukan proses additive noise, sinyal_nois = sinyal_audio + nois.
6. Ambil satu frame sinyal_nois (20 ms), dan lakukan pengamatan bentuk sinyal plus nois dalam domain waktu dan domain frekuensi.
7. Anda manfaatkan ***low pass filter*** (LPF) yang sudah anda rancang pada langkah percobaan sebelumnya (anda bisa memiliki 4.1 atau 4.2) untuk memfilter sinyal_nois.
8. Ambil satu frame sinyal output dari filter (20 ms), dapatkan gambaran sinyal output dari filter dalam domain waktu dan domain frekuensi.
9. Berikan catatan untuk melakukan analisa anda, dengan membandingkan karakteristik sinyal audio asli, sinyal audio plus nois dan sinyal output dari LPF.
10. Untuk lebih memantapkan pengamatan anda, coba bandingkan karakteristik suara sinyal asli, sinyal bercampur nois, dan sinyal output dari filter. Untuk pengamatan ini anda harus melakukan sepanjang sinyal (bukan 1 frame).

4.5. Pemfilteran Sinyal Audio II

Pada percobaan ini anda diminta untuk menyusun sebuah equalizer sederhana yang terdiri dari 3 buah filter digital (LPF, BPF dan HPF), dan melakukan pemfilteran pada sinyal audio

untuk mengamati kinerjanya.

1. Rancang 3 buah filter (LPF, BPF, dan HPF), bisa anda gunakan dari langkah percobaan 4.1 atau 4.2 sesuai selera anda. Lakukan modifikasi pada LPF (wc=0.35), BPF (wc1=0.35 & wc2=0.55), dan HPF (wc=0.55). Amati bentuk respon frekuensi ketiga filter tersebut.
2. Ambil sebuah sinyal audio (file *.wav) dalam hal ini anda bisa memanfaatkan file audio yang ada di PC anda.
3. Lakukan proses resample sehingga frekuensi sampling yang baru menjadi 10000 (untuk sinyal suara) atau 20000 Hz (untuk sinyal musik).

Gambar 9. Model Pemfilteran paralel

4. Lakukan proses pemfilteran dengan ketiga filter tersebut secara parallel.
5. Ambil satu frame sinyal output (20 ms) dari salah satu output filter, misalnya output dari LPF dan lakukan proses pengamatan domain frekuensi.
6. Lakukan hal yang sama pada output dari BPF dan HPF.
7. Catat perbedaan output dari ketiga filter tersebut, dan landa gunakan untuk melakukan analisa.

V. TUGAS DAN ANALISA

1. Bandingkan karakteristik filter hasil perancangan dengan Windowing dan Park Mc Clelland, jelaskan perbedaannya.

2. Bandingkan hasil pemfilteran sinyal audio bernoise dengan menggunakan teknik windowing dengan hasil pemfilteran pada sinyal yang sama jika anda menggunakan Park Mc Clelland.

MODUL VIII

PERANCANGAN DAN IMPLEMENTASI FIR FILTER

DENGAN TMS320C6713

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Siswa mampu menjelaskan cara mendisain FIR filter dengan menggunakan Matlab Command.
- Siswa mampu mengimplementasikan disain filter FIR pada DSK TMS320C6713 menggunakan
  bahasa-C

II. PERANCANGAN FIR FILTER

2.1. Perancangan Filter FIR Memanfaatkan Library Matlab.

Dalam hal ini kita memanfaatkan salah satu library berupa fungsi yang bernama fir1. Fungsi ini merupakan sebuah design filter FIR yang didasarkan pada metode window. Bentuk sintaknya sederhana seperti berikut ini:

```
B = FIR1(N,Wn)
```

Langkah ini akan menghasilkan sebuah design filter FIR dengan order N yang memiliki respon frekuensi dengan krakteristik Low Pass Filter. Outputnya berupa sebuah deretan koefisien filter sepanjang N+1 dan ditandai sebagai vector B.

Nilai cut-off frequency Wn harus berana diantara nilai 0 < Wn < 1.0, yang mana nilai 1.0 dalam hal ini akan ekuivalen dengan nilai ½ dari nilai sampling rate. Koefisien-koefisien yang disimpan dalam variable B merupakan nilai real dan memiliki kecenderungan karakteristik perubahan fase yang linear terhadap perubahan nilai frekuensi. Gain ternormalisasi filter pada frekuensi cut off Wn adalah sebesar -6 dB.

Untuk memberikan perubahan karakteristik respon frekuensi pada filter FIR yang dihasilkan, kita bisa merubah sintak diatas sebagai berikut:

```
B = FIR1(N,Wn,'high')
```

Jika anda ingin mendapatkan sebuah filter band pass, anda harus melakukan pengesetan nilai Wn sebagai dua element vektor nilai frekuensi rendah dan nilai frekuensi tinggi, Wn = [W1 W2], bentuk sintaknya adalah sebagai berikut:

```
B = FIR1(N,Wn,'bandpass')
```

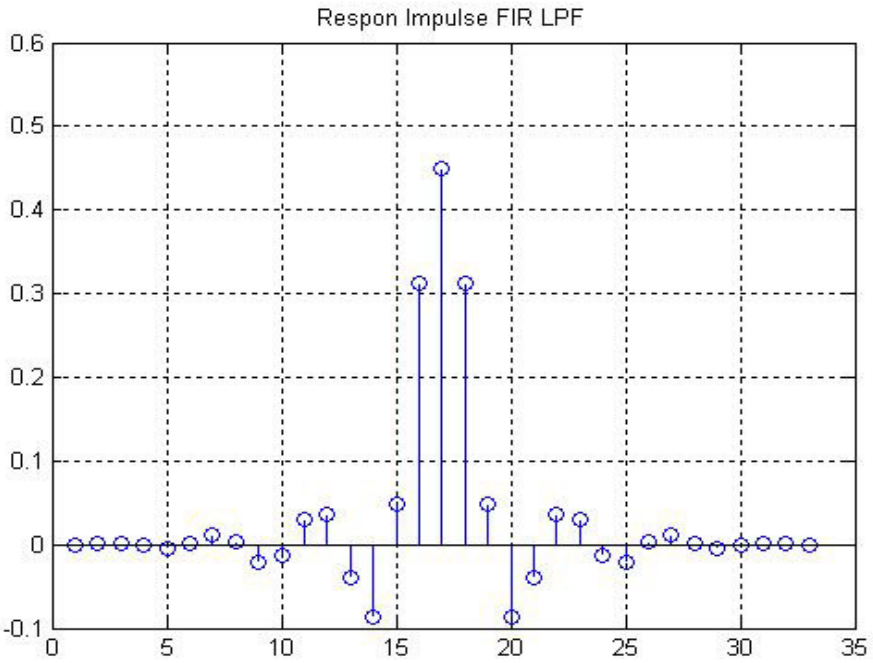
Sedangkan untuk menghasilkan sebuah band stop filter syntaknya adalah sebagai berikut:

```
B = FIR1(N,Wn,'stop')
```

Berikut ini adalah langkah merancang sebbuah losa pass filter dengan nilai orde filter N = 32, dan frkeuensi cut off Wn = 0.45.

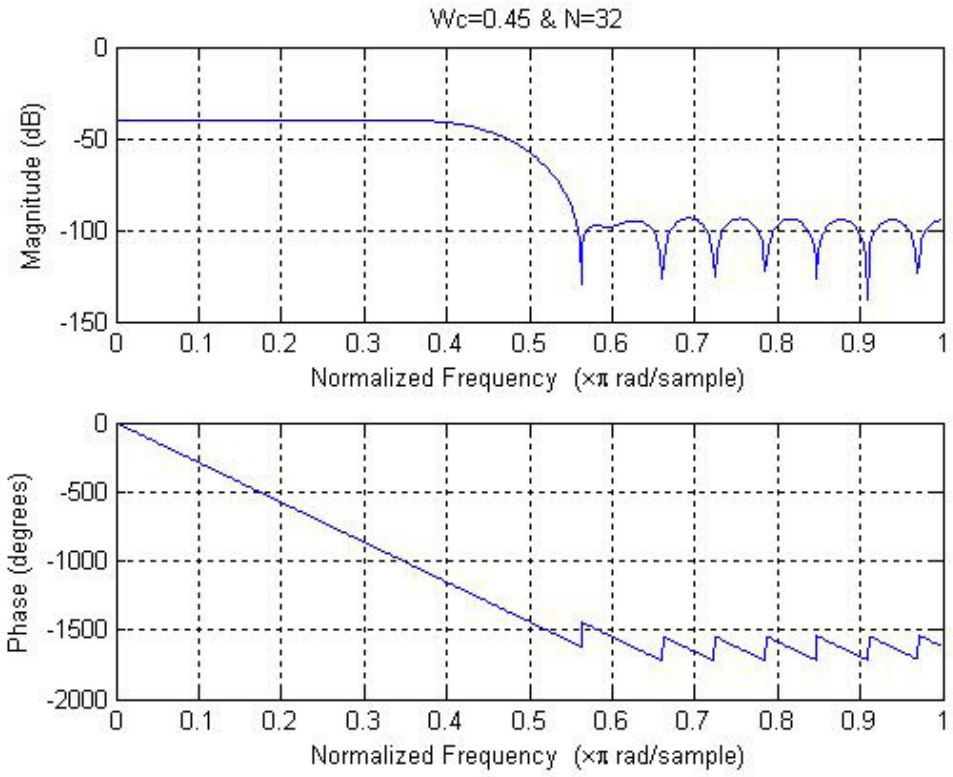
```
clear all;
N=32;
Wn=0.45;
B = FIR1(N,Wn,'low');
figure(1);stem(B);
figure(2);freqz(B,100);
```

Program tersebut akan menghasilkan sebuah vector B yang berisi koefisien-koefisien low pas filter, dan memilikibentuk respon impulse seperti berikut.



Gambar 1. Respon Impuse Low Pass Filter

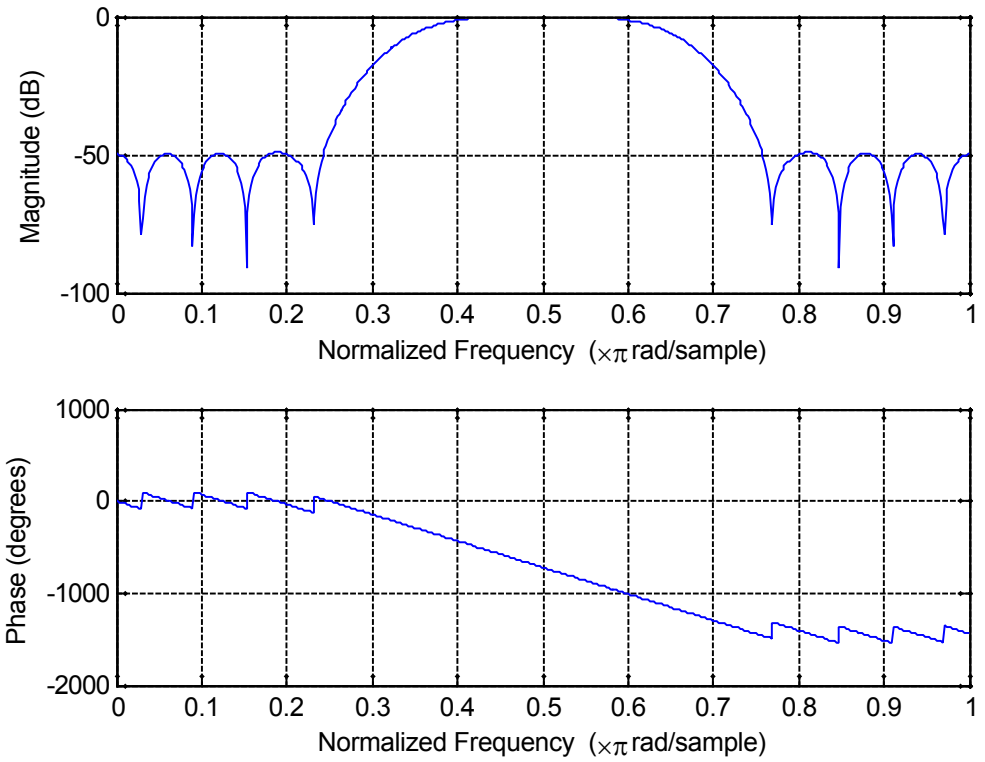
Sedangkan respon frekuensinya dapat dilihat seperti pada gambar berikut ini.



Gambar 2. Respon Frekuensi Low Pass Filter

Contoh perancangan untuk sebuah band pass filter yang akan meloloskan frekuensi ternormalisasi antara 0.35 dan 0.65, dengan nilai order filter sebesar N=32 adalah seperti program dibawah ini.

```
b = fir1(32,[0.35 0.65]);
freqz(b,1,512)
```



Gambar 3. Respon Frekuensi Sebuah band Pass Filter

2.2 Proses Loading Koefisien Filter ke Project CCS

Dari nilai koefisien-koefisien tersebut selanjutnya perlu dilakukan perubahan untuk menyesuaikan dengan standar data yang digunakan pada program CCS. Untuk itu perlu dilakukan perubahan menjadi data integer 16 bit dengan syntak seperti berikut:

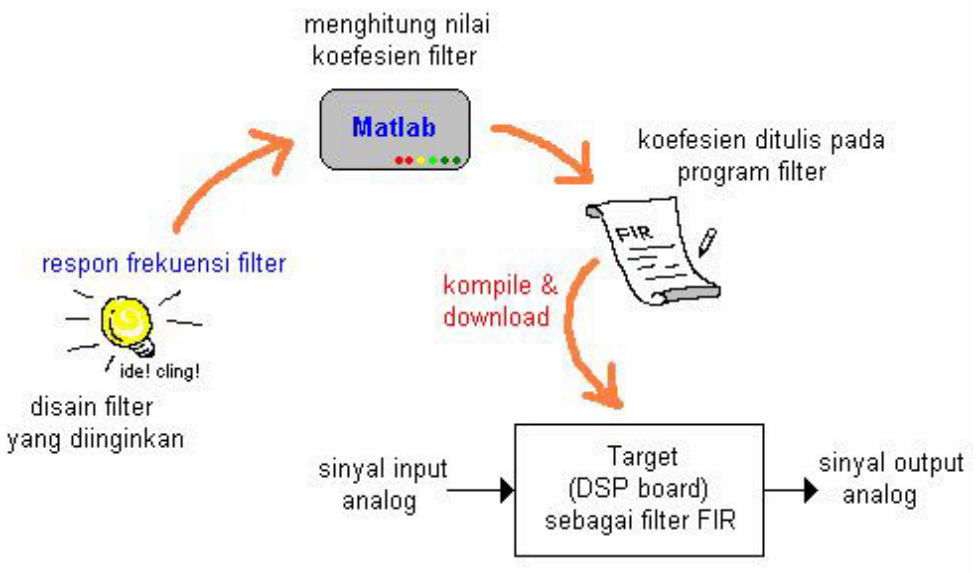
```
c = int16(B*2^15);
```

Langkah ini akan merubah nilai-nilai koefisin yang awalnya dalam bentuk pecahan menjadi deretan nilai-nilai integer 16 bit. Langkah ini dilakukan untuk mempermudah pembuatan dan eksekusi program CCS. Kita pahami bahwa DSP board yang kita gunakan adalah tipe floating point, tetepi DSP board ini memiliki kemampuan untuk melakukan eksekusi program dan data yang disusun dalam format fix point.

Nilai koefisien yang ditampilkan di dalam tampilan Matlab sebagai output dari variable c, selanjutnya bisa dimodifikasi dengan menggunakan program Notepad untuk menghasilkan sebuah file berkstensi **cof**. Pemilihan program editor Notepad ini dilakukan karena sifatnya yang universal dan bisa melakukan editing berbagai macam format data dan menghasilkan penyimpanan yang bagus, walaupun bentuk tampilannya sederhana.

File koefisien filter *.coef selanjutnya diinclude-kan ke project yang anda susun menggunakan program CCS melalui langkah Add Files to project. Sehingga secara sederhana proses loading koefisien filter hasil perancangan dengan matlab ke peoject yang anda susun

adalah seperti pada gambar berikut ini.



Gambar 4. Ilustrasi proses downloading koefisien filter keDSP board

III.PERALATAN PERCOBAAN

- 1 set PC yang dilengkapi dengan software Code Composer Studio.
- 1 set DSK TMS320C5402
- 1 set Speaker Active
- Function Generator
-

- Oscilloscope

Gambar 5. Penataan peralatan percobaan

IV.LANGKAH PERCOBAAN

4.1. Perancangan dengan Menggunakan Matlab

1. Tentukan spesifikasi filter yang akan anda buat, misalnya tetapkan nilai orde filter 32, frekuensi cut of ternormalisasi terhadap ½ frekuensi sampling sebesar 0.25. Anda gunakan perancangan FIR filter secara klasik menggunakan teknik windowing seperti berikut ini.

```
%LPF_01.m
N=32;Wn=0.25;
B=fir1(N,Wn);
freqz(B,1,256);
c=int16(B*2^15)'
```

Hasilnya adalah

```
c =
    0
  -44
  -86
  -90
```

```
.....
-44
 0
```

2. Jangan lupa anda mengimpan gambar respon frekuensi dan dari program yang telah anda buat tadi anda juga harus memodifikasi sehingga bisa menampilkan respon impulse FIR filter, atau yang manggambarkan nilai-nilai koefisiend alam bentuk tampilan grafik.

4.2. Mempersiapkan File Koefisien Filter

1. Anda aktifkan Notepad editor, selanjutnya anda blok dan kopikan nilai-nilai c yang dihasilkan oleh program Matlab yang sudah anda buat.

2. Susun dan modifikasi editor pada Notepad anda, sehingga memenuhi standar sebuah file koefisien yang bisa diambil dengan bahasa peomrograman C seperti berikut.

```
#define  N  33  //mendefinisikan nilai N = 33

short  h[N]=

{

 0,-44,-86,-90,

 0,191,381,370,

 0,-665,-1248,-1176,

 0,2273,5044,7327,8211,7327,5044,2273,

 0,-1176,-1248,-665,

 0,370,381,191,

 0,-90,-86,-44,

 0

};
```

3. Anda simpan file ini dengan langah sbb, **File Name:** LP1000.cof, **Save as type:** All Files, dan **Encoding:** ASCII. Selanjutnya anda tekan Ok Jangan lupa anda menyimpan file tersebut pada folder dimana anda akan membangun sebuah project FIR filter dengan CCS.

4.3. Menyusun Project FIR Filter dengan DSK Board

1. Buat sebuah project baru dengan nama FIR_LP.pjt

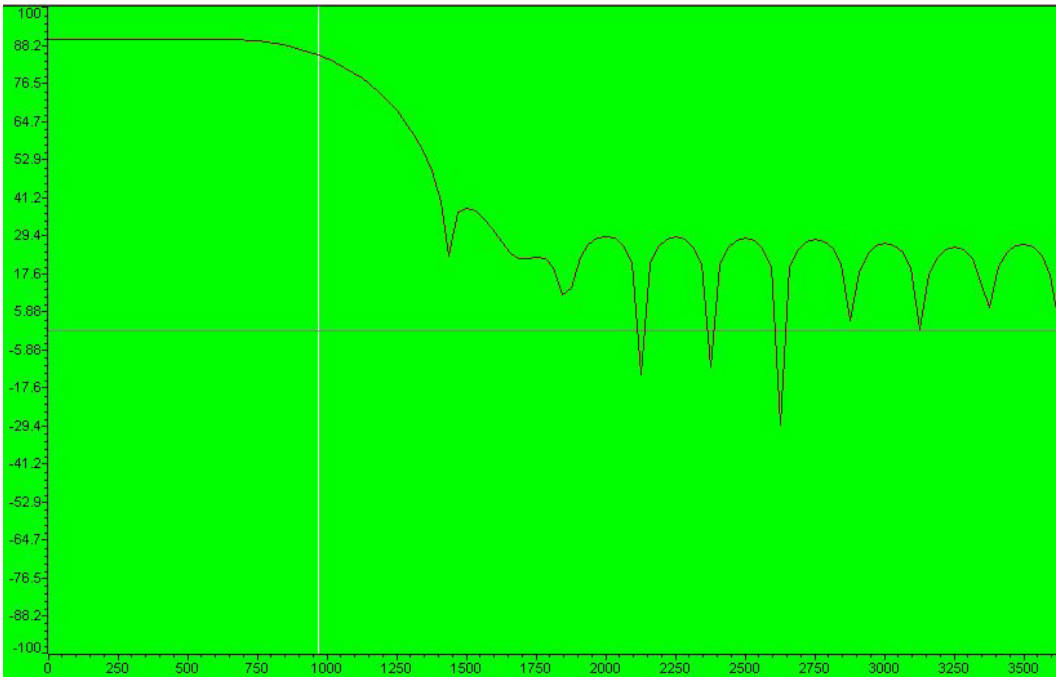
2. Buat sebuah program dalam bahasa FIR_LP.c , dengan menggunakan editor CCS yang sudah anda aktifkan dan simpan pada folder FIR_LP.pjt.

```
//FIR_LP.c  Koefisien FIR filter akan di-include with length N
#include "LP1000.cof"                              //file koefieisn filter
#include "dsk6713_aic23.h"                         //codec-dsk support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  //set sampling rate
int yn = 0;                                        //initialize filter's output
short dly[N];              //delay samples

interrupt void c_int11()     //ISR
{
 short i;

          dly[0]=input_sample();            //input newest sample
          yn = 0;                 //initialize filter's output
 for (i = 0; i< N; i++)
   yn += (h[i] * dly[i]);  //y(n) += h(i)* x(n-i)-->h(i): impulse respon of filter
 for (i = N-1; i > 0; i--)  //starting @ end of buffer
            dly[i] = dly[i-1];     //update delays with data move
          output_sample(yn >> 15);   //scale output filter sample
          return;
}

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

3. Anda tambahkan file **FIR_LP.c** ke dalam **FIR_LP.pjt** melalui prosedur yang benar, yaitu **Add File to Project**, dst.
4. Lakukan hal yang sama untuk file koefisien filter **LP1000.cof**.
5. Ulangi langkah 3 untuk file-file seperti : **dsk6713_aic23.h, c6713dskinit.c, Vectors_intr.asm, csl6713.lib, rts6700.lib**, dan **dsk6713bsl.lib**.
6. Lakukan **Scan All file Dependencies** untuk mengetahui file-file terkait lainnya yang diperlukan.
7. Lakukan pengesetan build option untuk menentukan lokasi Debug, dsb.
8. Load output program ke DSK board.
9. Buat tampilan grafik dengan memanfaatkan langkah View, dst. Dalam hal ini anda bisa menetapkan **Display Type:** FFT Magnitude, **Signal Type:** Real, **Start Address:** h, **Acquisition Buffer Size:**128, **FFT Frame Size:** 32, **FFT Order:** 8, **DSP Data Type:** 16-bit signed integer, **Sampling Rate:** 8000, **Magnitude Display Scale:** Logarithmic.

Parameter-parameter lain yang tidak disebutkan disini bisa anda tentukan sendiri .

10. Jalankan program anda untuks ejenak dan hentikan. Selanjutnya dari Toolbar Window pada CCS anda lihat tampilan. Anda amati tampilan respon frekuensi pada CCS yang telah anda buat. Bandingkan dengan gambar respon frekuensi yang dihasilkan pada Matlab pada waktu anda merancang FIR filter tersebut. Jika keduanya sudah menunjukkan pola gambar y ang sama, maka prose perancangan FIR filter yang anda lakukan sudah benar.



Gambar 6. Respon frekuensi pada display CCS

4.4. Melakukan pengujian dengan sinyal Sinus

1. Lakukan penataan peralatan seperti pada Gambar 5 yang ditunjukkan sebelumnya. Jangan

lupa anda melakukan pengecekan sinyal yang dihasilkan oleh function generator dengan menentukan nilai Vpp sebesar 100 mVolt.

2. Lakukan perubahan nilai frekuensi pada function generator untuk mengetahui apakah sinyal outputnya sudah benar. Dalam hal ini tampilan pada oscilloscope akan berubah menyesuaikan perubahan yang dilakukan pada nilai frekuensi pada function generator.

3. Jalankan program yang ada pada CCS, dan masukkan sinyal output dari function generator ke DSK board melalui line-in. Anda lihat output yang dihasilkan oleh DSP board dengan menghubungkan oscilloscope dengan Line Out pada DSP Board.

4. Lakukan perubahan nilai frekuensi pada function generator dan catat nilai-nilai seperti yang dibutuhkan pada Table 1.

5. Buat gambar pada beberapa sampel bentuk sinyal output yang dihasilkan pada oscilloscope untuk beberapa nilai frekuensi.

Tabel 1. Hasil pengukuran LPF

No	V in 100 m Vpp		Vout/Vin Dalam dB
	Frek. Input (Hz)	Vout (mVpp)	
1	500		
2	600		
3	700		
4	800		
5	900		
6	1000		
7	1100		
8	1200		
9	1300		
10	1400		
11	1500		
12	1750		
13	2000		
14	2500		
15	3000		
16	3500		
17	4000		

6. Anda lakukan perhitungan menggunakan calculator untuk mendapatkan nilai perbandingan Vout/Vin dari data yang telah anda dapatkan.

7. Buat grafik dari nilai-nilai Vout/Vin sebagai fungsi dari nilai frekuensi yangbervariasi dari 500 Hz sampai 4000 Hz.

4.5. Perancangan & Implementasi Filter FIR (High Pass, Band Pass dan Band Stop)

Dalam hal ini anda diharuskan mengulang prosedur percobaan dari langkah 4.1 sampai 4.4. untuk spesifikasi filter yang baru dengan karakteristik berbeda seperti berikut ini:

High Pass Filter:
- Orde filter: 32
- Frekuensi Cut Off: 0.5

Band Pass Filter:
- Orde filter: 32
- Frekuensi Cut Off: 0.2 dan 0.5

Bans Stop Filter
- Orde filter: 32
- Frekuensi Cut Off: 0.2 dan 0.5

V. TUGAS

1. Pada tugas mendisain filter low-pass 1 KHz, bandingkan plot grafik respon frekuensi filter pada Matlab dengan tabel 1. Lakukan hal yang sama untuk High Pass Filter, Band Pass Filter dan Band Stop Filter.

2. Filter low-pass 1KHz dirancang hanya untuk melewatkan sinyal input pada range frekuensi 0 sampai dengan 1 KHz masih ada sinyal input dengan frekuensi lebih tinggi dari 1 KHz yang masih dilewatkan? Berikan penjelasannya.

MODUL IX
PERANCANGAN DAN IMPLEMENTASI FILTER IIR DENGAN MATLAB

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Siswa memahami cara menghitung nilai koefisien filter IIR dengan teknik direct form I dan direct form II.
- Siswa mampu mengimplemantasikan hasil rancangan ke sebuah file sinyal audio sebagai sinyal uji.

II. KONSEP PERANCANGAN IIR FILTER

Tujuan perancangan filter adalah untuk menghasilkan sebuah system yang mampu memberikan perlakukan khusus pada data (sinyal input) sesuai dengan spesifikasi frekuensi yang ditetapkan. Contohnya dalah bagaimana menghilangkan komponen frekuensi diatas 400 Hz pada suatu sinyal yang disampel dengan frekuensi sampling sebesar 2000 Hz.

2.1. Filter IIR

Dalam terminologi IIR filter, yang mana nilai ouput di waktu sebelum mulainya sistem (< 0) sudah diperhitungkan dengan adanya parameter ***feedback internal*** dari sistem tersebut. IIR Filter bisa dinyatakan dalam sebuah persamaan beda yang merepresentasikan hubungan ***output/input*** sistem LTI sebagai berikut:

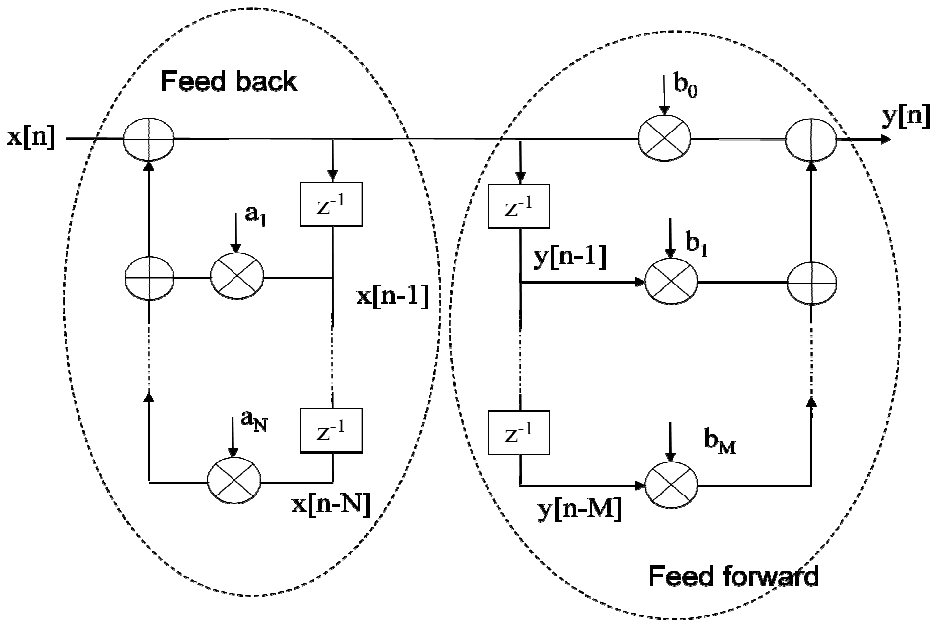
$$y[n] = \sum_{l=1}^{N} a_l y[n-l] + \sum_{k=0}^{M} b_k x[n-k] \qquad (1)$$

di mana:
M = orde filter
i, j = filter indek
b_i = koefisien *feed forward* IIR filter
a_j = koefisien *feed back* IIR filter
M+N+1, menentukan banyaknya (total koefisien), sedangkan untuk order filter biasanya ditetapkan sebagai nilai N

Filter IIR memiliki kecenderungan bentuk ***roll off*** yang tajam. Berbeda dengan design pada suatu filter FIR, yang seringkali didasarkan pada pendekatan sebuah deretan input secara numerik, IIR filter design seringkali dimulai dengan sebuah analog filter yang mana kita rasakan

memiliki sifat mendekati. ***Fungsi transfer*** (system function) dapat direpresentasikan sebagai jumlahan tak hingga (infinite) pada sederetan ***ratio of polynomials***. Dengan adanya penggunaan ***feed back***, mungkin sekali system ini tidak stabil. Sistem ini juga tidak memiliki respon fase linear terhadap perubahan frekuensi sinyal input.

Di dalam realisasinya sebuah IIR filter bisa didekati dengan bentuk blok diagram standar seperti berikut.



Gambar 1. Diagram Blok IIR Filter

2.2. Design IIR dengan Prototype Analog Filter

Salah satu metode yang sederhana dalam merancang sebuah IIR Filter adalah melalui pembuatan prototipe Filter Analog (Butterworth, Chebyshev, dsb). Dari hasil perancangan ini kemudian kita amati nilai ***pole & zero***, bentuk ***respon frekuensinya***, dan ***respon impulse***. Di dalam aktifitas praktikum ini yang akan kita lakukan adalah merancang sebuah IIR filter, melihat respon frekuensinya dan mencobanya untuk melakukan pemfilteran pada sbuah sinyal.

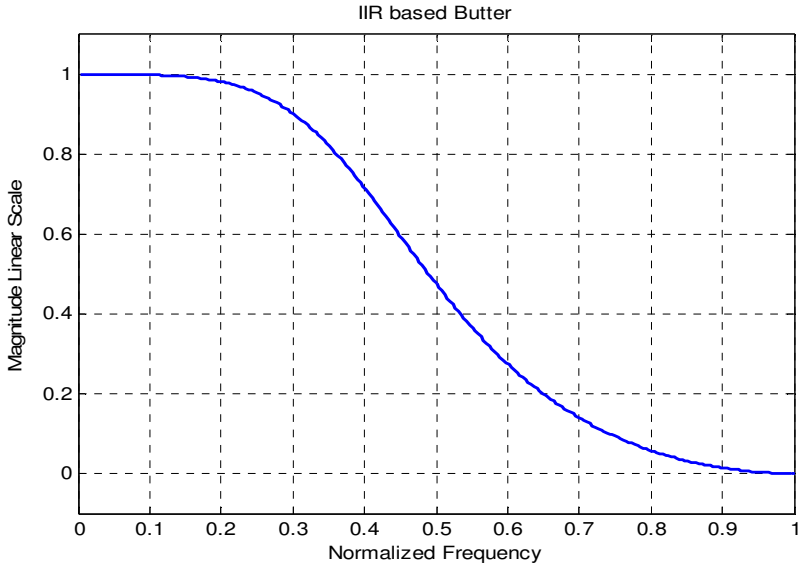
Matlab telah menyediakan banyak fungsi yang berkaitan dengan perancangan IIR filter menggunakan ***prototype*** analog filter, salah satunya adalah ***butterworth filter analog***. Untuk mendesing analog ***butterworth filter***, anda lakukan dengan perintah `[B,A] = butter(N,Wn)`. Langkah ini akan menghasilkan sebuah ***low pass filter*** digital yang berbasis pada ***Analog Butterwoth filter*** dengan order senilai N, dan frekuensi cut off sebesar Wn. Variabel B merupakan koefisien ***feed forward*** IIR filter, dan variable A merupakan koefisien ***feedback filter*** IIR. Koefisien-koefieisn ini akan menentukan bentuk perbandingan polynomial-z. Nilai frekuensi ***cut-off*** harus berada diantara 0,0 < Wn < 1,0. Dimana 0 dalam hal ini akan ekuivalen

dengan 0 Hz, sedangkan nilai 1,0 akan ekuivalen dengan nilai *f*s/2 (setengah sampling rate atau setengan frekuensi sampling).

Jika dalam proses perancangan ditetapkan bahwa Wn merupakan dua-element vektor, misalnya `Wn = [W1 W2]`, maka perintah **`butter`** diatas akan menghasilkan suatu ***band pass filter*** dengan orde 2N, dengan daerah ***pass band*** yang berada diantara `W1 < W < W2`. Sedangkan untuk menghasilkan filter high pass, dan band stop, kita harus memodifikasi perintah diatas seperti halnya pada waktu melakukkannya pada perancangan FIR filter.

Berikut ini kita akan melakukan perancangan sebuah filter IIR dengan orde N=2, frekuensi cut off wc=0,4, dan dalam hal ini kita menginginkan respon frekuensinya merupakan sebuah ***low pass filter***. Langkah pembuatan programnya adalah seperti berikut.

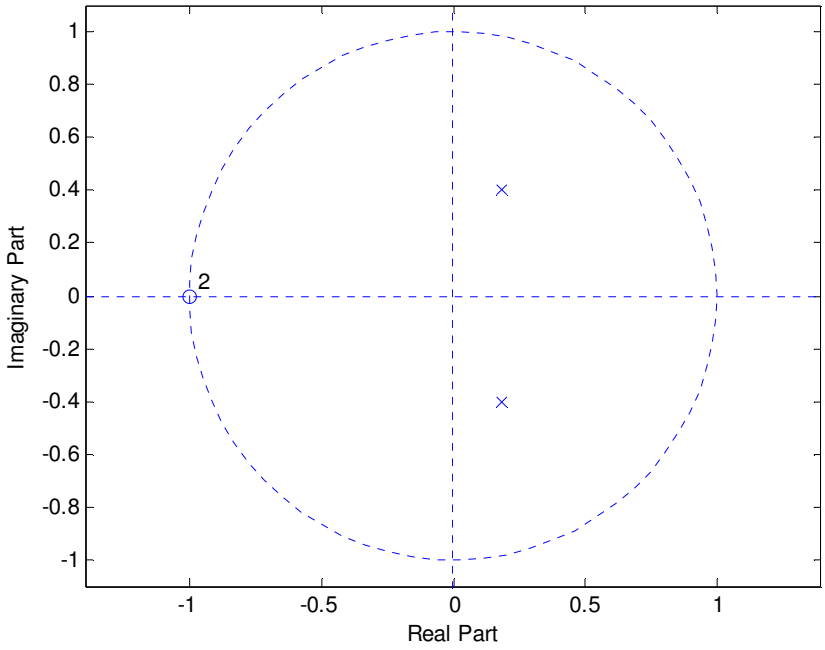
```
clear all;
N = 2; % Order of filter
wc = 0.4; % Cut Off frequency
[b,a] = butter(N,wc);
[H,W]=freqz(b,a,256);
hh=length(H);
f=1:hh;
figure(1);
plot(f/hh,abs(H),'linewidth',2);
axis([0 1 -.1 1.1]);
zplane(b,a)
```



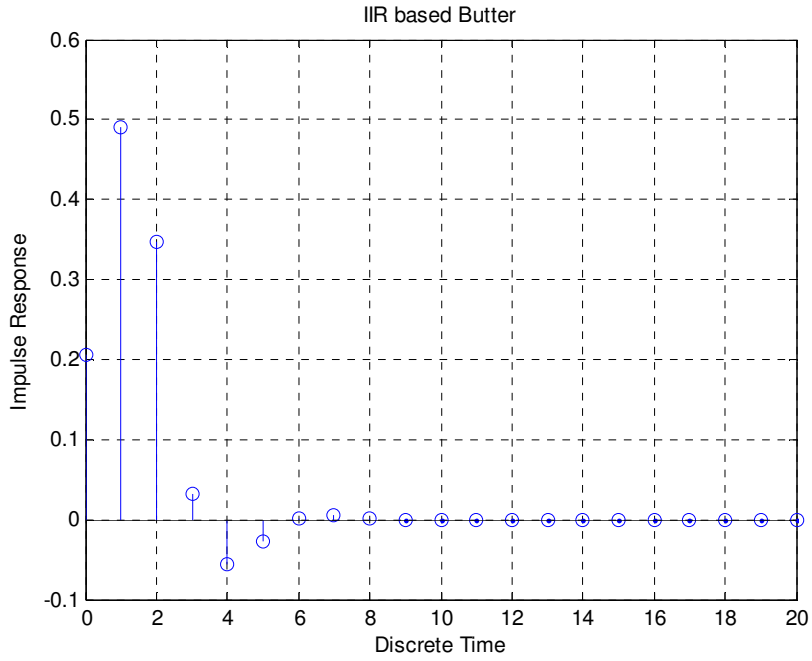
Gambar 2. Respon Frekuensi IIR Filter dengan prototipe Butterworth, skala linear

Dalam penskalaan dalam deci Bell (dB) perlu tambahan langkah berikut ini.

```
plot(f/hh,20*log10(abs(H)),'linewidth',2);
[y,t] = impz(b,a,21);
stem(t,y);
```



Gambar 3. Posisi zero dan pole pada bidang z



Gambar 4. Respon Impulse IIR Filter dengan prototipe Butterworth

2.3. Perancangan IIR Filter dengan Metode Direct Design

Tidak seperti metode analog prototyping, metode direct design tidak dibatasi oleh konfigurasi repon frekuensi standar klasik seperti lowpass, highpass, bandpass, atau bandstop. Dalam hal ini proses perancangan dilakukan secar bebas, bahkan bisa dilakukan secara multiband.

Algorithm membentuk suatu ***least-squares fit*** di dalam domain waktu. Langkah ini akan digunakan untuk menghitung koefisien ***denominator*** (penyebut) menggunakan persamaan ***Yule-Walker*** pada nilai koefisien korelasi yang diperoleh dari invers transformasi Fourier pada respon frekuensi yang sudah dipilih. Untuk mendapatkan bagian koefisien polinomial ***numerator*** (pembilang), ***yulewalk*** melakukan beberapa langkah berikut:

1. Hitung suatu polinomial numerator berkaitan dengan suatu dekomposisi additive pada ***power frequency response***.
2. Evaluasi respon frekuensi secara lengkap berkaitan dengan polynomial numerator dan denominator.
3. Gunakan suatu teknik faktorisasi spektral untuk menentukan respon impulse pada filter.
4. Dapatkan polynomial numerator dengan suatu least-squares fit untuk respons impulse ini.

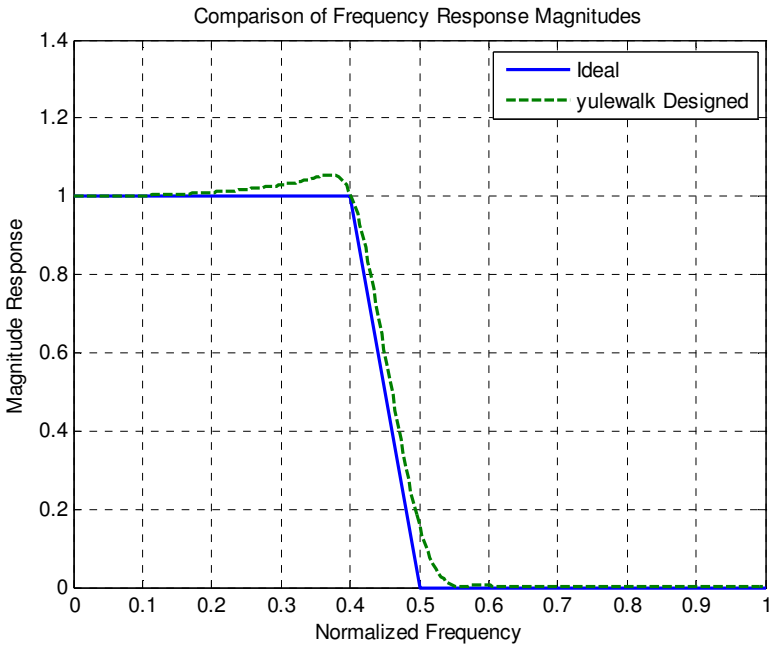
Outputnya adalah sebuah dereten koefisien filter IIR dengan pangkat z seperti berikut ini.

$$\frac{B(z)}{A(z)} = \frac{b(1)+b(2)z^{-1}+\cdots b(n+1)z^{-n}}{a(1)+a(2)z^{-1}+\cdots a(n+1)z^{-n}} \qquad (2)$$

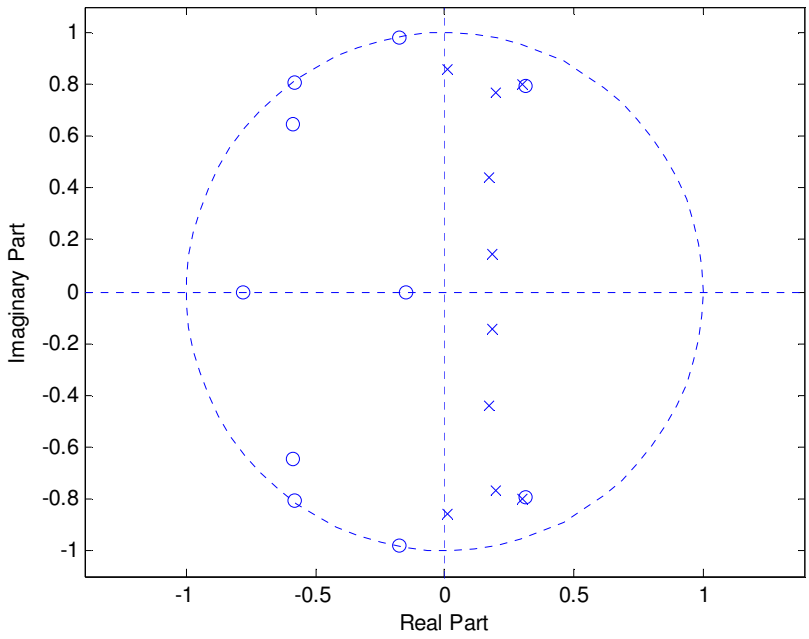
Matlab menyediakan sebuah fungsi yulewalk yang bisa digunakan untuk merancang sebuah filter digital recursive IIR melalui penentuan pada suatu respon frekuensi secara spesifik. Berikut ini kita akan melakukan perancangan sebuah filter IIR dengan orde N=10, frekuensi cut off wc=0,4, dan dalam hal ini kita menginginkan respon frekuensinya merupakan sebuah ***low pass filter***. Langkah pembuatan programnya adalah seperti berikut.

```
m=[1 1 1 1 1 0 0 0 0 0];
f=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1];
[b,a]=yulewalk(10,f,m);
[h,w]=freqz(b,a,256);
plot(f,m,w/pi,abs(h),'--','linewidth',2)
legend('Ideal','yulewalk Designed')
title('Comparison of Frequency Response Magnitudes')
xlabel('Normalized Frequency')
ylabel('Magnitude Response')
grid on
zplane(b,a)
```

Hasilnya diperoleh seperti pada Gambar 5 dan Gambar 6 berikut.



Gambar 5. Respon frekuensi IIR Filter dengan direct design method

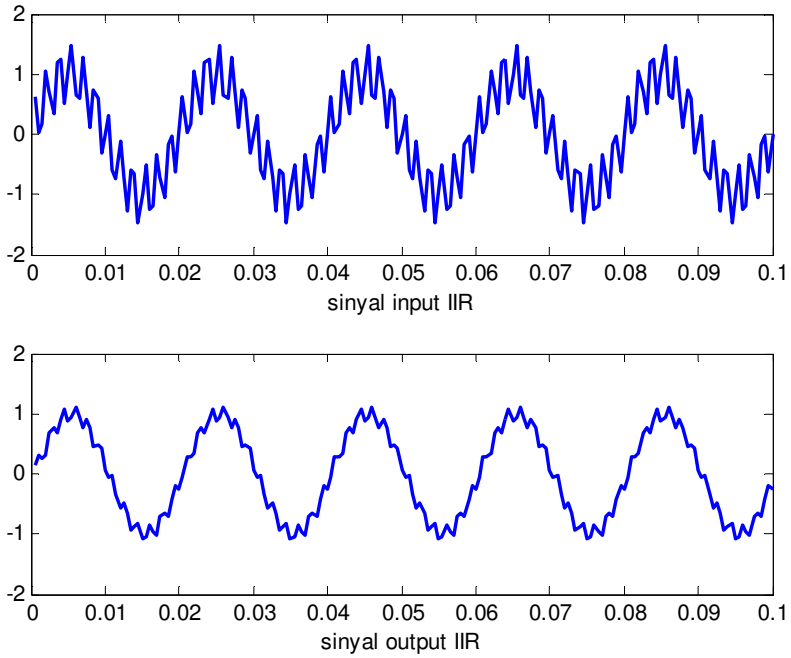


Gambar 6. Lokasi zero dan pole Filter IIR direct design method

2.4. Pemfilteran dengan IIR

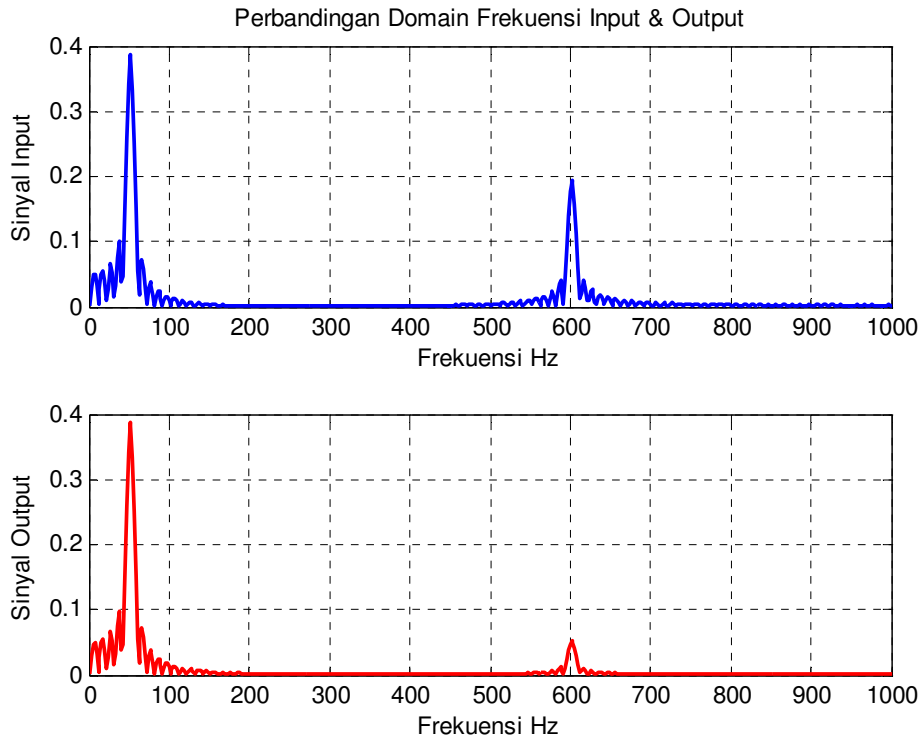
Proses pemfilteran dengan IIR pada dasarnya sama dengan proses pemfilteran menggunakan FIR. Dengan didasari operasi konvolusi, proses pemfilteran bisa dilakukan dengan memanfaatkan fungsi yang sudah disediakan dalam library Matlab. Untuk melakukan pemfilteran bisa menggunakan perintah berikut `y = filter(B,A,x)`. Perintah ini merupakan proses pemfilteran IIR pada sinyal input x yang dalam ini memiliki bentuk satu dimensi. Proses pemfilterannya dilakukan dengan mengacu pada bentuk diagram blok IIR filter ***direct form II***.

Untuk lebih mudahnya, kita akan mencoba menyusun sebuah proses pemfilteran dengan IIR. Dalam hal ini sinyal input x(n) berupa sebuah sinyal sinus yang tersusun dari dua frekuensi, yaitu 50 Hz dan 600 Hz. Sinyal difilter dengan ***low pass filter*** IIR yang memiliki frekuensi ***cut off*** 400 Hz.



Gambar 7. Perbandingan sinyal input dan sinyal ouput pada domain waktu

Dari Gambar 7 dapat dilihat bahwa bentuk sinyal input dengan sinyal output cukup memiliki perbedaan. Pengaruh frekuensi tinggi yang muncul pada sinyal input, bisa diredam sehingga bentuk sinyal menjadi lebih halus. Untuk lebih jelasnya, efek penggunaan IIR filter bisa dilihat dalam perbandingan domain frkeuensi seperti pada Gambar 6. Proses peredaman pada komponen frekuensi 600 Hz mampu mereduksi dari level 0,2 sampai menjadi < 0,05.



Gambar 8. Perbandingan sinyal input dan sinyal ouput pada domain frekuensi

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:

- PC yang memiliki spesifikasi memory minimal 1 Gb
- Operating System minimal Windows Xp
- Perangkat Lunak Matlab
- Sistem Audio untuk PC

IV. PERCOBAAN

4.1. Perancangan Filter IIR dengan Analog Prototipe

1. Pada langkah ini anda diharuskan merancang sebuah filter yang meloloskan semua frekuensi ternormalisasi kurang dari atau sama dengan 0.45, dan meredam semua komponen frekuensi yang lebih besar dari 0.45 (frekuensi ***cut-off***, wc=0,45). Dalam hal ini anda tentukan order filter sebesar 4, dan perancangan yang anda lakukan berbasis pada teknik analog prototyping Butterworth.
2. Amati bentuk respon impulse dan respon magnitude dan fase sebagai fungsi frekuensi (respon frekuensi). Anda harus mampu menggambarkan respon frekuensi di dalam skala dB.
3. Cari titik dimana nilai magnitudonya mengalami peredaman sampai -3 dB, dapatkan nilai frekuensinya. Dan beri tanda daerah ***pass band***, ***transition band***, dan ***stop band*** dari LPF yang telah anda rancang tersebut.
4. Ulangi langkah 1 sampai 3 untuk kasus HPF, dimana frekuensi ***cut-off*** bernilai 0.25.
5. Untuk langkah ini anda harus merancang sebuah band pass filter yang akan digunakan untuk meloloskan komponen frekuensi (ternormalisasi) antara 0,25 sampai dengan 0,55.
6. Amati bentuk respon frekuensinya, dan dapatkan nilai frekuensi pada saat redaman magnitudonya senilai -3dB. Anda tarik garis dari posisi ***cut off*** rendah (-3 dB pertama) sampai dengan cut-off tinggi (-3dB kedua) pada BPF yang telah anda rancang.

Di sini hal ini anda harus mengambil gambar respon frekuensi dari filter tersebut, dan menandai lokasi-lokasi yang diminta dalam petunjuk. Anda sebaiknya melakukannya secara manual untuk lebih mudah memahami perintah-perintah diatas.

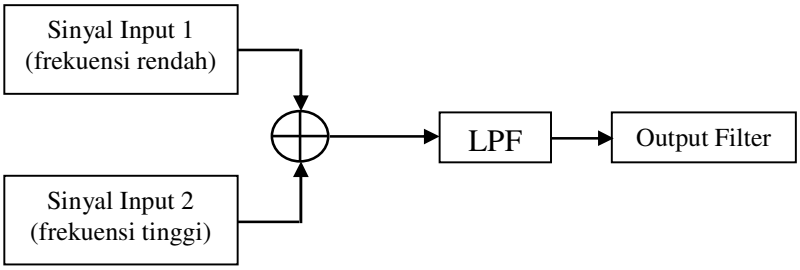
4.2. Perancangan Filter IIR dengan Direct Design Method

1. Anda tetapkan pemetaan frekuensi untuk filter anda, misalnya F=[0 .15 .25 .35 .45 .55 .65 .75 .85 1.0]; Usahakan komponen penyusun vector F selalu genap, misalnya 6, 8, 10, dsb. Untuk mendapatkan sebuah LPF, tetapkan vector A yang merepresentasikan pemetaan magnitude pada filter yang anda rancang, dalam hal ini tetapkan A bernilai 1 untuk 6 komponen pertama, dan bernilai 0 untuk komponen sisanya. Dalam hal ini jumlah komponen A harus sama dengan komonen pada F.
2. Dapatkan koefisien-koefisien filter LPF anda, dengan fungsi '***remez***' yang mendukung algorithma Yule-Walker dengan order filter N=10.
3. Berikan gambaran respon frekuensinya (magnitude dan fase sebagai fungsi frekuensi).
4. Ulangi langkah 1 sampai 3 untuk mendapatkan sebuah ***high pass filter*** (HPF) dengan frekuensi cut off wc=0.25.
5. Ulangi langkah 2 sampai 3 untuk mendapatkan sebuah ***band pass filter*** (BPF) yang memiliki daerah ***pass band*** dari 0.25 sampai 0.55.
6. Untuk langkah 1 sampai 5 pada bagian ini anda harus membandingkannya dengan prosedur perancangan yang telah anda susun dengan metode windowing.
7. Anda amati perbedaan respon frekuensi kedua metode tersebut untuk satu jenis filter yang sama, misalnya untuk LPF pada teknik window anda bandingkan dengan LPF pada Yule-Walker.

4.3. Pemfilteran IIR Untuk Sinyal Sederhana

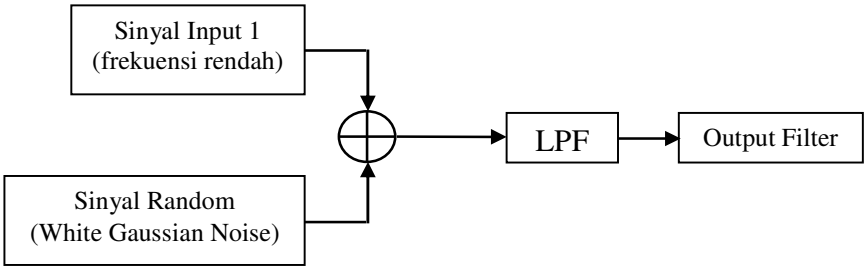
Untuk langkah berikutnya, anda bisa memiliki dengan Analog Prototype atau Direct Design algorithma Yule Walker sesuai dengan selera anda.

1. Bangkitkan sebuah sinyal sinus yang memiliki amplitude A=1, frekuensi f=200, dan sampling rate yang digunakan adalah 2000 Hz.
2. Bangkitkan sinyal sinus kedua, dengan amplitude A2= 0.25, frekuensi f2=600 Hz, dengan sampling rate yang sama dengan langkah 1. Lakukan penjumlahan sinyal pertama dengan sinyal kedua, sinyal_out = sinyal1 + sinyal2.



Gambar 9. Proses pemfilteran pada sinyal sinus

3. Lakukan proses pemfilteran dengan menggunakan LPF yang telah dibuat pada langkah percobaan 4.1.
4. Amati bentuk sinyal input pertama, sinyal input kedua, sinyal hasil jumlahan dalam domain waktu dan domain frekuensi.
5. Amati bentuk sinyal setelah proses pemfilteran dalam domain waktu dan domain frekuensi.
6. Anda bangkitkan sinyal nois Gaussian yang dalam hal ini panjangnya sama dengan vector yang anda gunakan untuk menyusun sinyal pada langkah 1 pada percobaan 4.3 ini.
7. Anda kalikan nilai nois terbangkit dengan 0.2 (WGN = mean + var*random_Gauss, dalam hal ini mean = 0,0).
8. Jumlahkan nois ini dengan sinyal sinus (sinyal pertama) yang anda miliki, sinyal_ nois = sinyal + nois.
9. Lakukan pemfilteran dengan LPF seperti yang telah anda gunakan pada langkah 3, pada percobaan 4.3 ini.
10. Amati gambaran sinyal sinus, sinyal_nois, dan sinyal output hasil proses pemfilteran dalam domain waktu dan domain frekuensi.



Gambar 10. Proses pemfilteran pada sinyal sinus

4.4. Pemfilteran IIR untuk Sinyal Audio Bernois

Pada percobaan ini anda akan menggunakan sinyal Audio untuk menguji kemampuan anda dalam menyusun sebuah filter digital dan memahami cara kerjanya.

1. Lakukan pengambilan sinyal audio dengan memanfaatkan fungsi ***wavread*** yang ada pada Matlab, misalnya file ***a.wav*** atau file lain yang memiliki karakteristik mono (bukan stereo).
2. Lakukan proses resample sehingga sampling rate yang baru bernilai 10000.
3. Ambil satu frame, kurang lebih 20 ms (ekuivalent dengan 0,02*sampling rate), dan sajikan gambaran sinyal tersebut dalam domain waktu dan domain frkeuensi.
4. Bangkitkan sebuah noise Gaussian dengan vector sepanjang sinyal audio, tentukan nilai varians noise sebesar 0.2 , → nois=var*nois_terbangkit.
5. Lakukan proses additive noise, sinyal_nois = sinyal_audio + nois.
6. Ambil satu frame sinyal_nois (20 ms), dan lakukan pengamatan bentuk sinyal plus nois dalam domain waktu dan domain frekuensi.
7. Anda manfaatkan ***low pass filter*** (LPF) yang sudah anda rancang pada langkah percobaan sebelumnya (anda bisa memiliki 4.1 atau 4.2) untuk memfilter sinyal_nois.
8. Ambil satu frame sinyal output dari filter (20 ms), dapatkan gambaran sinyal output dari filter dalam domain waktu dan domain frekuensi.
9. Berikan catatan untuk melakukan analisa anda, dengan membandingkan karakteristik sinyal audio asli, sinyal audio plus nois dan sinyal output dari LPF.
10. Untuk lebih memantapkan pengamatan anda, coba bandingkan karakteristik suara sinyal asli, sinyal bercampur nois, dan sinyal output dari filter. Untuk pengamatan ini anda harus melakukan sepanjang sinyal (bukan 1 frame).

V. TUGAS DAN ANALISA

1. Bandingkan karakteristik filter hasil perancangan dengan Analog Prototipe Butterworth dan Direct Design Yule Walker, jelaskan perbedaannya.
2. Bandingkan hasil pemfilteran sinyal audio bernoise dengan menggunakan teknik Analog Prototipe dengan hasil pemfilteran pada sinyal yang sama jika anda menggunakan Direct Design Yule Walker.
3. Bandingkan karakteristik respon frekuensi filter (LPF, BPF dan HPF) yang dirancang menggunakan IIR (Analog Prototipe Butterworth) dan yang dirancang menggunakan FIR (Windowing)

MODUL X

PERANCANGAN DAN IMPLEMENTASI FILTER IIR

DENGAN TMS320C6713

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Siswa mampu menjelaskan cara mendisain IIR filter dengan menggunakan Matlab Command.
- Siswa mampu mengimplementasikan disain filter IIR pada DSK TMS320C6713 menggunakan bahasa-C.

II. KONSEP PERANCANGAN IIR FILTER

Tujuan perancangan filter adalah untuk menghasilkan sebuah system yang mampu memberikan perlakukan khusus pada data (sinyal input) sesuai dengan spesifikasi frekuensi yang ditetapkan. Contohnya adalah bagaimana menghilangkan komponen frekuensi diatas 1000 Hz pada suatu sinyal yang disampel dengan frekuensi sampling sebesar 8000 Hz.

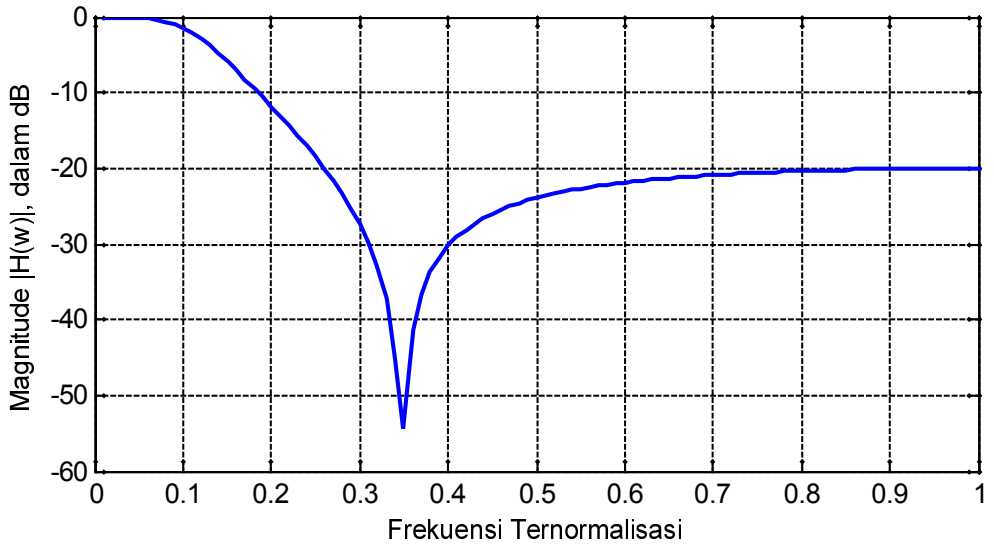
2.1. Filter IIR

Dalam terminologi IIR filter, yang mana nilai ouput di waktu sebelum mulainya sistem (< 0) sudah diperhitungkan dengan adanya parameter ***feedback internal*** dari sistem tersebut. IIR Filter bisa dinyatakan dalam sebuah persamaan beda yang merepresentasikan hubungan ***output/input*** sistem LTI sebagai berikut:

(1)

di mana:
M = orde filter
i, j = filter indek
b_i = koefisien *feed forward* IIR filter
a_j = koefisien *feed back* IIR filter
M+N+1, menentukan banyaknya (total koefisien), sedangkan untuk order filter biasanya ditetapkan sebagai nilai N

Satu contoh filter IIR yang disusun dengan metode chebysev dengan frekuensi cut off =0.25 dan orde 2 memiliki memiliki karakteristik respon frekuensi seperti Gambar 1 berikut ini.



Gambar 1. Respon frekuensi IIR Filter orde 2, dengan metode Chebysev II

Dari gambar tersebut menunjukkan bahwa prosesperedaman sinyal sudah mulai sebelum frekueni ternormalisasi sinyal input 0,25 dan peredaman terbesar terjadi disekitar daerah frekuensi 0,3 sampai 0,4. Perbandingan sinyal output dan input menunjukkan kenaikan level pada frekuensi diatas 0,4, hal ini menunjukkan bahwa peredaman sinyal berkurang. Karakteristik system IIR dengan order 2 (katagori orde rendah) yang disusun dengan metode Chebysev II secara umum memiliki karakteristik seperti diatas.

Jika anda ingin mengetahui frekuensi riil dalam besaran Hz, anda harus melakukan proses ekuivalensi terhadap frekuensi samping yang anda gunakan. Skala 0 sampai 1 pada gambar diatas akan akuivalen dengan senilai 0 Hz sampai Fs/2 Hz. Dalam hal ini untuk frekuensi sampling sebesar 8000 Hz, akan memberikan gambaran kinerja system IIR untuk daerah 0 Hz sampai 4000 Hz.

2.2. Implementasi Filter IIR

Ada beberapa struktur diagram blok untuk mempermudah dalam usaha mendekati ke arah implementasinya. Beberapa struktur yang cukup popular adalah seperti berikut ini:

Direct Form I Structure

Merupakan struktur yang disusun secara langsung dari persamaan beda yang merepresentasikan sebuah hubungan input dan output pada system IIR. Jika kita memiliki persamaan beda seperti pada persamaan (2),

$$y[n] = a_1 y[n-1] + b_0 x[n] + b_1 x[n-1] \quad (2)$$

Sistem diatas memiliki bentuk hubungan input dan output dalam domain-z seperti berikut ini.

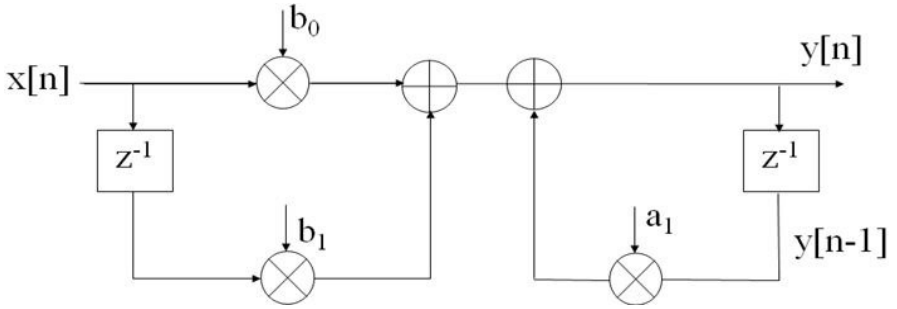
$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} = \left(\frac{1}{1 - a_1 z^{-1}}\right)(b_0 + b_1 z^{-1}) = \left(\frac{1}{A(z)}\right)B(z)$$

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} = \left(\frac{1}{1 - a_1 z^{-1}}\right)(b_0 + b_1 z^{-1}) = \left(\frac{1}{A(z)}\right)B(z) \quad (3)$$

Untuk implementasinya kita harus menjabarkan lebih dulu persamaan domain z tersebut menjadi persamaan beda seperti berikut.

$$v(n) = b_0 x(n) + b_1 x(n-1)$$
$$y(n) = a_1 y(n-1) + v(n-1) \quad (4)$$

Dengan mengacu pada persamaan beda (4) kita bisa merealisasikan dengan Direct Form I untuk IIR adalah seperti Gambar 2. Dari gambar tersebut jelas bahwa untuk suatu orde filter N=1 diperlukan delay sebanyak 2 elemen.



Gambar 2. Direct Form Structure I filter IIR orde 1

Direct Form II Structure

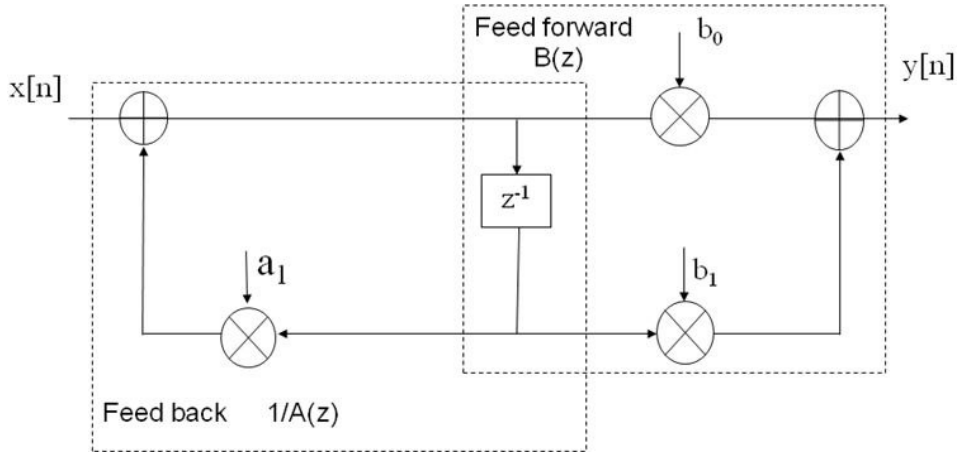
Merupakan satu struktur yang banyak digunakan untuk langkah implementasi system IIR. Struktur ini bisa didapatkan dengan penyederhanaan direct form structure I dengan cara memodifikasi persamaan (3).

$$H(z) = \left(\frac{1}{A(z)}\right)B(z) = B(z)\left(\frac{1}{A(z)}\right)$$

$$H(z) = \left(\frac{1}{A(z)}\right)B(z) = B(z)\left(\frac{1}{A(z)}\right) \quad (5)$$

Dari persamaan (4) kita bisa membentuk sebuah diagram blok untuk implemetasi Direct Form II Structure seperti berikut ini. Struktur ini memiliki kelebihan disbanding struktur

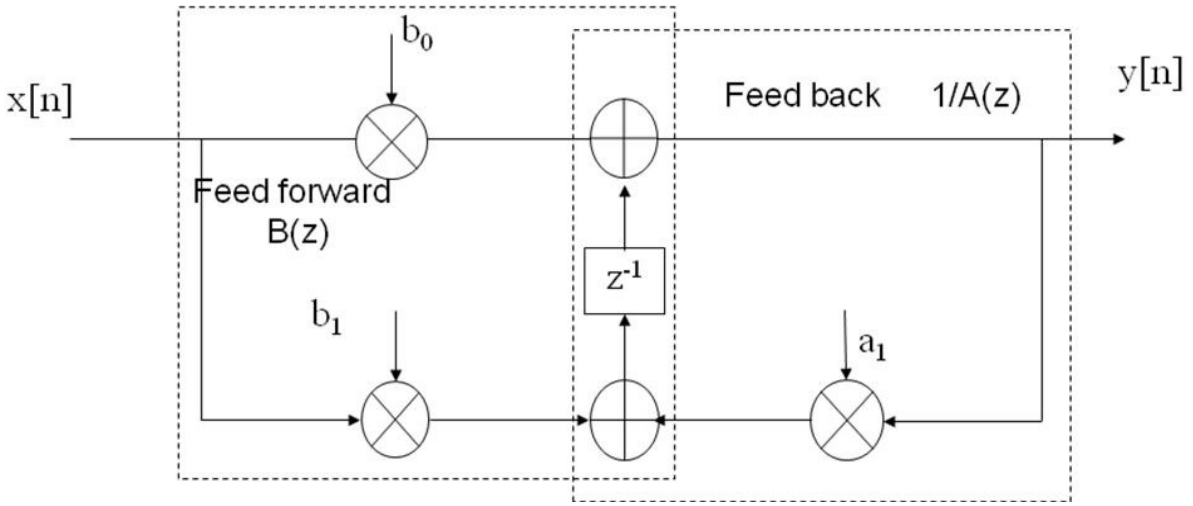
sebelumnya, dimana dalam hal ini diperlukan elemen delay yang lebih sedikit untuk filter IIR dengan orde yang sama.



Gambar 3. Direct Form II Structure system IIR orde 1

Direct Form II Transpose

Struktur ini merupakan model pembalikan arah dari struktur yang ada pada Direct Form II Structure. Dalam hal ini semua arah anak panah mengalami transoformasi arah. Untuk elemen pengali dipertukarkan posisinya dimana untuk koefisien feedforward didekatkan ke arah input, sedangkan untuk koefisien feedback didekatkan ke arah output. Posisi yang dihasilkan struktur ini memiliki efisiensi seperti pada Direct Form II Structure, tetapi bentuk dasarnya mirip dengan Direct Form I Structure, sehingga kelihatan lebih sederhana dan cukup efisien. Bentuk yang dihasilkan untuk filter IIR dengan orde 1 bisa dilihat pada Gambar 4



Gambar 4. Direct Form II Structure Transpose untuk sistem IIR orde 1

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:

- 1 set PC yang dilengkapi dengan software Code Composer Studio.

- 1 set DSK TMS320C5402
- 1 set Speaker Active
- Function Generator
- Oscilloscope

Gambar 5. Penataan peralatan percobaan

IV.PERCOBAAN

4.1.Perancangan dengan Menggunakan Matlab

1. Tentukan spesifikasi filter yang akan anda buat, misalnya tetapkan nilai orde filter 2, frekuensi cut of ternormalisasi terhadap ½ frekuensi sampling sebesar Wn=0.25. Anda gunakan perancangan IIR filter dengan metode Chebysev II seperti berikut ini.

```
%File Name: IIR_LPF.m
N=2;R=20;Wn=0.25;
[B,A] = cheby2(N,R,Wn);
freqz(B,A,100);
b_16=int16(B*2^15)
a_16=int16(A*2^15)
```

2. Jangan lupa anda mengimpan gambar respon frekuensi dan dari program yang telah anda buat tadi anda juga harus memodifikasi sehingga bisa menampilkan respon impulse IIR filter, atau yang manggambarkan nilai-nilai koefisien dalam bentuk tampilan grafik.

4.2. Mempersiapkan File Koefisien Filter

1. Anda aktifkan Notepad editor, selanjutnya anda blok dan kopikan nilai-nilai c yang dihasilkan oleh program Matlab yang sudah anda buat.

2. Susun dan modifikasi editor pada Notepad anda, sehingga memenuhi standar sebuah file koefisien yang bisa diambil dengan bahasa peomrograman C seperti berikut.

```
//lp1000.cof IIR lowpass coefficients file, cutoff frequency 1 kHz
int a[3] ={32767, -32768, 20072};      //numerator coefficients
int b[3] ={3431, -3356, 3431};
```

3. Anda simpan file ini dengan langah sbb, **File Name:** LP1000.cof, **Save as type:** All Files, dan **Encoding:** ASCII. Selanjutnya anda tekan Ok. Jangan lupa anda menyimpan file tersebut pada folder dimana anda akan membangun sebuah project IIR filter dengan CCS.

4.3.Menyusun Project IIR Filter dengan DSK Board

1. Buat sebuah project baru dengan nama IIR_LP.pjt

2. Buat sebuah program dalam bahasa IIR_LP.c , dengan menggunakan editor CCS yang sudah anda aktifkan dan simpan pada folder IIR_LP.pjt.

```
/IIR_LP.c IIR filter menggunakan cascaded Direct Form II
//Koefisien-2 a dan b berkaitan dengan nilai-nilai a dan b dari MATLAB

#include "DSK6713_AIC23.h"       //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;     //set sampling rate
#include "lp1000.cof"      //memanggil koefisien filter low pass
short dly[3] = {0};        //inisialisasi element delay

interrupt void c_int11()    //ISR
{
   short  input;
   int   un, yn;
   input = input_sample();
   un=input-((b[0]*dly[0])>>15)-((b[1]*dly[1])>>15)-((b[2]*dly[2])>>15);
   yn=((a[0]*un)>>15)+((a[1]*dly[0])>>15)+((a[2]*dly[1])>>15);

   dly[2] = dly[1];   //update delays
   dly[1] = dly[0];   //update delays
   dly[0] = un;       //update delays
   input = yn;

   output_sample((short)yn);     //output final result for time n
   return;                       //return from ISR
}

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

3. Anda tambahkan file **IIR_LP.c** ke dalam **IIR_LP.pjt** melalui prosedur yang benar, yaitu **Add File to Project**, dst.
4. Lakukan hal yang sama untuk file koefisien filter **LP1000.cof**.
5. Ulangi langkah 3 untuk file-file seperti : **dsk6713_aic23.h, c6713dskinit.c, Vectors_intr.asm, csl6713.lib, rts6700.lib**, dan **dsk6713bsl.lib**.
6. Lakukan **Scan All file Dependencies** untuk mengetahui file-file terkait lainnya yang diperlukan.
7. Lakukan pengesetan **build option** untuk menentukan lokasi Debug, dsb.
8. Load output program ke DSK board.

4.4.Pengujian dengan Sinyal Sinus

1. Lakukan penataan peralatan seperti pada Gambar xx yang ditunjukkan sebelumnya. Jangan lupa anda melakukan pengecekan sinyal yang dihasilkan oleh function generator dengan menentukan nilai Vpp sebesar 100 mVolt.

2. Lakukan perubahan nilai frekuensi pada function generator untuk mengetahui apakah sinyal outputnya sudah benar. Dalam hal ini tampilan pada oscilloscope akan berubah

menyesuaikan perubahan yang dilakukan pada nilai frekuensi pada function generator.

3. Jalankan program yang ada pada CCS, dan masukkan sinyal output dari function generator ke DSK board melalui line-in. Anda lihat output yang dihasilkan oleh DSP board dengan menghubungkan oscilloscope dengan Line Out pada DSP Board.
4. Lakukan perubahan nilai frekuensi pada function generator dan catat nilai-nilai seperti yang dibutuhkan pada table 1.
5. Buat gambar pada beberapa sampel bentuk sinyal output yang dihasilkan pada oscilloscope untuk beberapa nilai frekuensi.

Tabel 1. Hasil pengukuran LPF
Fs=8000 Hz

No	ω (freq normalized)	f (dalam Hz)	V_Out	V_In/V_Out	20 log (V_In/V_Out)
1	0.1				
2	0.2				
3	0.25				
4	0.26				
5	0.27				
6	0.28				
7	0.29				
8	0.30				
9	0.31				
10	0.32				
11	0.33				
12	0.34				
13	0.35				
14	0.26				
15	0.27				
16	0.28				
17	0.29				
18	0.30				
19	0.31				
20	0.32				

21	0.33				
22	0.34				
23	0.35				
24	0.36				
25	0.37				
26	0.38				
27	0.39				
28	0.40				
29	0.50				
30	0.60				
31	0.70				
32	0.70				
33	0.80				
34	0.90				

6. Anda lakukan perhitungan menggunakan calculator untuk konversi dari ω menjadi f (Hz).

7. Buat grafik dari hasil perbandingan V_Out/V_In dan 20log(V_Out/V_In) sebagai fungsi dari nilai frekuensi yang bervariasi dari 0.1 sampai 1.0. Dalam hal ini anda harus memisahkan kedua grafik tersebut, karena tidak mungkin mendapatkan gabungan dua grafik dalam skala yang berbeda.

4.5.Perancangan dan Implementasi High Pass Filter dengan Metode IIR

1. Tentukan spesifikasi filter yang akan anda buat, misalnya tetapkan nilai orde filter 2, frekuensi cut of ternormalisasi terhadap ½ frekuensi sampling sebesar Wn=0.5. Anda gunakan perancangan IIR filter dengan metode Chebysev II seperti berikut ini.

```
%File Name: IIR_HPF.m
N=2;R=20;Wn=0.5;
[B,A] = cheby2(N,R,Wn,'high');
freqz(B,A,100);
b_16=int16(B*2^15)
a_16=int16(A*2^15)
```

2. Anda aktifkan Notepad editor, selanjutnya anda blok dan kopikan nilai-nilai c yang

dihasilkan oleh program Matlab yang sudah anda buat.

3. Susun dan modifikasi editor pada Notepad anda, sehingga memenuhi standar sebuah file koefisien yang bisa diambil dengan bahasa peomrograman C seperti berikut.

```
//hp2000.cof IIR highpass coefficients file, cutoff frequency 2 kHz
int a[3] ={32767, 29127, 10923};  //numerator coefficients
int b[3] ={5461, -3641, 5461};
```

4. Buat sebuah project IIR_HP.pjt dan buat program dalam bahasa IIR_HP.c , dengan menggunakan editor CCS yang sudah anda aktifkan dan simpan pada folder IIR_HP.pjt. Lakukan proses aktivasi program CCS untuk menghasilkan sebuah system IIR HPF seperti pada langkah yang telah anda lakukan pada kasus low pass filter IIR. Dalam hal ini tentu saja anda harus melakukan sedikit modifikasi pada proses `#include "lp1000.cof"` menjadi `#include "hp2000.cof"`

5. Proses selanjutnya adalah seperti yang telah anda lakukan pada IIR low pass filter, anda seting pada build option, kompilasi, download program dan dilanjutkan dengan pengujian dan analisa proses pemfilteran terhadap sinyal input dari function generator.

V. TUGAS DAN ANALISA

1. Pada tugas mendisain filter low-pass 1 KHz, bandingkan plot grafik respon frekuensi filter pada Matlab dengan Tabel 1. Lakukan hal yang sama untuk High Pass Filter

2. Filter low-pass 1 KHz dirancang hanya untuk melewatkan sinyal input pada range frekuensi 0 sampai dengan 1 KHz masih ada sinyal input dengan frekuensi lebih tinggi dari 1 KHz yang masih dilewatkan? Berikan penjelasannya.

MODUL TAMBAHAN I

PEMBANGKITAN SINYAL DENGAN PERSAMAAN BEDA

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Mampu memanfaatkan persamaan beda untuk pembangkitan sinyal sederhana dengan TMS320C6713
- Mampu memanfaatkan persamaan beda untuk membangkitkan berbagai jenis sinyal yang lebih komplek

II. PERSAMAAN BEDA

Persamaan beda merupakan suatu formula untuk komputasi suatu sampel output pada waktu yang didasarkan pada sample yang diperoleh diwaktu sebelumnya dan sampel diwaktu sekarang, perhitungan ini dilakukan di dalam domain waktu. Kita bias menuliskan bentuk umum persamaan beda untuk sebuah system yang kausal, *linear time invariant* (LTI) seperti berikut ini.

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_M x(n-M) \\ -a_1 y(n-1) - a_2 y(n-2) - \square - a_N y(n-N) - a_1 y(n-1) - a_2 y(n-2) - \square - a_N y(n-N) \quad (1)$$

Persamaan (1) seringkali dituliskan menjadi bentuk yang lebih umum seperti berikut.

$$y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{j=1}^{N} b_j x(n-j) \quad (2)$$

Dimana x(n) merupakan sinyal input, a_i (I = 0,1,….M)dan b_j (j = 1,2,…N) disebut sebagai koefisien-koefisien dari persamaan beda.

2.1. Pembangkitan Dual Tone dengan Persamaan Beda Orde 2

Berikut ini adalah contoh pembangkitan dual tone dengan menggunakan persamaan beda. Outputnya disimpan didalam memory dan ditampilkan dengan fasilitas Code Composer Studio. Untuk menyajikan suara yang dihasilkan anda bisa menggunakan fasilitas line-out dan speaker aktif. Persamaan beda untuk menyusun pembangkita sinyal sinus adalah:

$$y(n) = Ay(n-1) – y(n-2) \quad (3)$$

dimana:

A = cos(ωT) dan dengan dua kondisi inisial adalah y(-1), y(-2) dan ω= 2πf

Dalam hal ini kita tetapkan bahwa frekuensi sampling sebesar 8kHz, sehingga nilai periode sampling adalah sebesar T=1/f = (1/8000) detik = 0.125 mdetik. Transformasi ke dalam domain-z memberikan bentuk baru sebagai berikut:

$$Y(z) = A\{z^{-1}Y(z) + y(-1)\} - \{ z^{-2}Y(z) + z^{-1}y(-1) + y(-2) \} \qquad (4)$$

Yang mana bisa dimodifikasi sebagai

$$Y(z) = Az^{-1}Y(z) + Ay(-1) - z^{-2}Y(z) - z^{-1}y(-1) - y(-2)$$

$$Y(z) = Az^{-1}Y(z) - z^{-2}Y(z) + Ay(-1) - z^{-1}y(-1) - y(-2)$$

$$Y(z) - Az^{-1}Y(z) + z^{-2}Y(z) = Ay(-1) - z^{-1}y(-1) - y(-2)$$

$$Y(z)\{1 - Az^{-1} + z^{-2}\} = Ay(-1) - z^{-1}y(-1) - y(-2)$$

$$= 2\cos(\omega T)(-\sin(\omega T)) - z^{-1}(-\sin(\omega T)) - (-\sin(2\omega T))$$

$$= -2\cos(\omega T)\sin(\omega T) + z^{-1}\sin(\omega T) - (-\sin(2\omega T))$$

$$= z^{-1}\sin(\omega T) \qquad (5)$$

Penyelesaian untuk Y(z) memberikan:

$$Y(z) = \frac{z^{-1}\sin(\omega T)}{1 - Az^{-1} + z^{-2}} = \frac{z^{1}\sin(\omega T)}{(z^{2} - Az^{1} + 1)} \qquad (6)$$

Invers untuk Y(z) memberikan hasil:

$$y(n) = Z^{-1}\{Y(z)\} = \sin(n\omega T) \qquad (7)$$

$$y(n) = Z^{-1}\{Y(z)\} = \sin(n\omega T) \qquad (7)$$

Untuk pembangkitan sinyal dengan frekuensi f1 = 1.5 kHz dan f2 =2kHz, bagaimana caranya?

Dalam hal ini kita harus menentukan nilai-nilai A, y(-1) dan y(-2). Hasil perhitungan dalam nilai pecahan (floating point) kemudian kita rubah menjadi nilai-nilai format fix point dengan konversi ke 2^{14}(2^14). Jika anda menginginkan untuk ke bentuk format 2^{15}(2^15) juga tidak menjadi persoalan.

f_1 = 1.5 kHz ➔ A= 2cos(2ωT) = 2cos(2*π*1500*0.00125) = 0.765

selanjutnya dikonversi ke bentuk penskalaan 2^{14}. ➔A*2^14 = 12,540

y(-1)= -sin(ωT) = - 0.924
selanjutnya dikonversi ke bentuk penskalaan 2^{14}. ➔y(-1)*2^14 =-15,137
y(-2)= -sin(2ωT) = - 0.707
selanjutnya dikonversi ke bentuk penskalaan 2^{14}. ➔y(-2)*2^14 =-11,585

f_2 = 2.0 kHz ➔ A= 2cos(2ωT) = 0
y(-1)= -sin(ωT) = -1
selanjutnya dikonversi ke bentuk penskalaan 2^{14}. ➔y(-1)*2^14 =-16,384

y(-2)= 0

Maka diperoleh nilai-nilai array sebagai y1[3] = {0,-15137,-11585}, A1 = 12540

dan y2[3] = {0,- 16384,0}, A2 = 0

Nilai-nilai ini digunakan untuk menyusun program, pada bagian akhir sinyal dari y1[3] dan y2[3] digabungkan untuk membentuk output dual tone pada persamaan beda.

2.2. Swept Sinusoida dengan Persamaan Beda

Untuk pembangkitan sinyal sinus dengan nilai frekuensi ter-sweep (naik dari nilai rendah sampai mencapai frekuensi tinggi tertentu) bisa kita gunakan persamaan beda berikut ini.

$$y(n) = Ay(n-1) – y(n-2) \quad (8)$$

Dimana A = 2cos(ωT), dan dua kondisi inisialisasi adlaah y(-1) =-sin(ωT) dan y(-2) =-sin(2ωT)

Suatu sinyal inisialisasi dengan frekuensi 500 Hz digunakan. Frekuensi dinaikkan 10 Hz sampai mencapai nilai 3500 Hz. DUsari masing-masing sinusoida terbangkit adalah 200 sampel, dan dapat direduksi untuksuatu proses yang lebih cepat. Dengan frekuensi inisialisasi pada 500 Hz, kita mendapatkan nilai konstanta sebagai:

A = 30274, y(0)=0, y(1)=-6270, dan y(-2)=-11585

Cara menghitungnya bagaimana?
f=500 Hz, ➔ A = 2cos(ωT) = 2cos(2*π*500*0.000125)=1.8478 ➔A*2^14 = 30274
y(-1) = -sin(ωT) = -sin(2*π*500*0.000125)=-0.3827 ➔y(-1)*2^14 = -6270
y(-2) = -sin(ωT) = -sin(2*π*2*500*0.000125)=-0.7071 ➔y(-2)*2^14 = -11585

Untuk setiap nilai frekuensi (500, 510,….dst) function coeff-gel dipanggil untuk menghitung suatu set konstanta pada A, y(n-1), dan y(n-2) dandiimplementasikan dalam persamaan beda untuk pembangkitan sinyal sesuai frekuensi yang terkait.

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:
- PC yang memiliki spesifikasi memory minimal 1 Gb
- Operating System minimal Windows Xp
- Perangkat Lunak Matlab
- Sistem Audio untuk PC

IV. PERCOBAAN

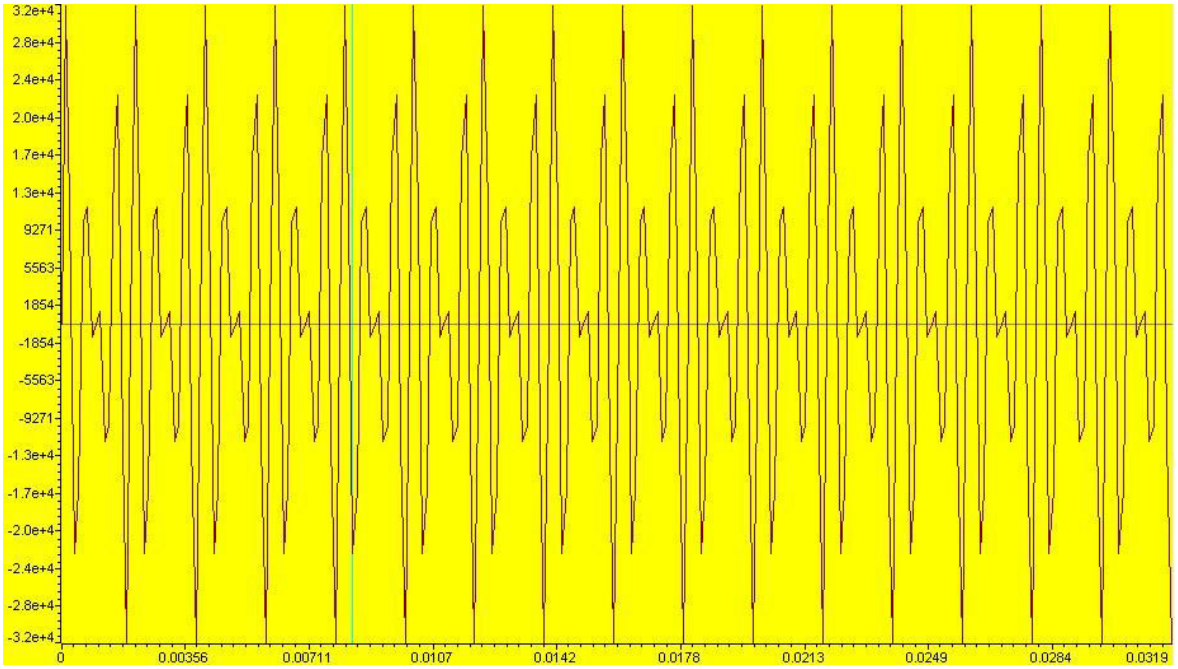
4.1. Pembangkitan Dual Tone dengan Persamaan Beda

1. Buat sebuah project baru dengan nama **tone_2.pjt** Anda buat sebuah program berbasis persamaan beda untuk pembangkitan sinyal dual tone dan beri nama **tone_2.c** dan simpat pada folder bernama **tone_2.** Listing program untuk pembangkitan dual tone adalah seperti berikut.

```
#include "DSK6713_AIC23.h" //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;     //set sampling rate
short sinegen(void);       //for generating tone
short output;                       //for output
short sinegen_buffer[256];      //buffer for output data
const short bufferlength = 256; //buffer size for plot with CCS
short i = 0;                        //buffer count index

short y1[3] = {0,-30278,-23167}; //y1(0),y1(-1),y1(-2) for 1.5kHz
const short A1 = 25068;          //A1 = 2coswT scaled by 2^14
short y2[3] = {0,-32768,0};      //y2(0),y2(-1),y2(-2) for 2kHz
const short A2 = 0;              //A2 = 2coswT scaled by 2^14

interrupt void c_int11()         //ISR
{
 output = sinegen();          //out from tone generation function
 sinegen_buffer[i] = output;        //output into buffer
 output_sample(output);           //output result
 i++;                              //increment buffer count
 if (i == bufferlength)     i = 0;  //if buffer count = size of buffer
 return;                           //return to main
}

short sinegen()                    //function to generate tone
{
 y1[0] =((y1[1]*A1)>>14)-y1[2]; //y1(n)= A1*y1(n-1)-y1(n-2)
 y1[2] = y1[1];                 //update y1(n-2)
 y1[1] = y1[0];                 //update y1(n-1)
```

```
 y2[0] =((y2[1]*A2)>>14)-y2[2]; //y2(n)= A2*y2(n-1)-y2(n-2)
 y2[2] = y2[1];                 //update y2(n-2)
 y2[1] = y2[0];                 //update y2(n-1)

 return (y1[0] + y2[0]);        //add the two tones
}

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

2. Anda tambahkan file **tone_2.c** ke dalam **tone_2.pjt** melalui prosedur yang benar, yaitu **Add File to Project**, dst.
3. Ulangi langkah 2 untuk file-file seperti : **dsk6713_aic23.h, c6713dskinit.c, Vectors_intr.asm, csl6713.lib, rts6700.lib**, dan **dsk6713bsl.lib**.
4. Lakukan **Scan All file Dependencies** untuk mengetahui file-file terkait lainnya yang diperlukan.
5. Lakukan pengesetan build option untuk menentukan lokasi Debug, dsb.
6. Load output program ke DSK board.Jalankan program anda, dan dengarkan suara yang dihasilkan dengan menggunakan speaker aktif. Untuk melihatbentuk gambar yang dihasilkan anda bisa memanfaatkan view graph pada CCS.



Gambar 1. Tampilan Domain Waktu dual-tone

Gambar 2. Tampilan Domain Frekuensi dual-tone

4.2. Pembangkitan Swept Sinusoid

1. Buat sebuah project baru dengan nama **SweepDE.pjt** Anda buat sebuah program berbasis persamaan beda untuk pembangkitan swept sinusoid dan beri nama **SweepDE.c** dan simpat pada folder bernama **SweepDE.** Listing program untuk pembangkitan swept sinusoid adalah seperti berikut.

```
//SweepDE.c  Generates a sweeping sinusoid using a difference equation

#include "DSK6713_AIC23.h"              //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;      //set sampling rate
#include <math.h>
#define  two_pi     (2*3.1415926)         //2*pi
#define  two_14           16384    //2^14
#define  T             0.000125    //sample period = 1/Fs
#define  MIN_FREQ              500       //initial frequency of sweep
#define  MAX_FREQ             3500       //max frequency of sweep
#define  STEP_FREQ            10    //step frequency
#define  SWEEP_PERIOD        200    //lasting time at one frequency
short       y0  = 0;                    //initial output
short    y_1 = -6270;                   //y(-1)=-sinwT(scaled)  f=500 Hz
short    y_2 = -11585;                  //y(-2_=-sin2wT(scaled) f=500 Hz
short    A = 30274;                 //A = 2*coswT scaled by 2^14
short       freq = MIN_FREQ;            //current frequency
short    sweep_count = 0;           //counter for lasting time
void     coeff_gen(short);          //function prototype generate coeff

interrupt void c_int11()            //ISR
{
```

```
 sweep_count++;                          //inc lasting time at one
frequency
 if(sweep_count >= SWEEP_PERIOD)     //lasting time reaches max duration
  {
  if(freq>=MAX_FREQ) freq=MIN_FREQ;//if current freq is max reinit
  else          freq=freq + STEP_FREQ; //incr to next higher frequency


  coeff_gen(freq);               //function for new set of coeff
  sweep_count = 0;               //reset counter for lasting time
  }

 y0=((A * y_1)>>14) - y_2; //y(n) = A*y(n-1) - y(n-2)
 y_2 = y_1;                          //update y(n-2)
 y_1 = y0;                                 //update y(n-1)
 output_sample(y0);             //output result
}

void coeff_gen(short freq)       //calculate new set of coeff
{
 float w;                                  //angular frequency

 w = two_pi*freq;                          //w = 2*pi*f
 A = 2*cos(w*T)*two_14;                    //A = 2*coswT * (2^14)
 y_1 = -sin(w*T)*two_14;         //y_1 = -sinwT *(2^14)
 y_2 = -sin(2*T*w)*two_14;       //y_2 = -sin2wT * (2^14)
 return;
}

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

2. Anda tambahkan file **SweepDE.c** ke dalam **SweepDE.pjt** melalui prosedur yang benar, yaitu **Add File to Project**, dst.
3. Ulangi langkah 2 untuk file-file seperti : **dsk6713_aic23.h, c6713dskinit.c, Vectors_intr.asm, csl6713.lib, rts6700.lib**, dan **dsk6713bsl.lib**.
4. Lakukan **Scan All file Dependencies** untuk mengetahui file-file terkait lainnya yang diperlukan.
5. Lakukan pengesetan build option untuk menentukan lokasi Debug, dsb. Load output program ke DSK board.Jalankan program anda, dan dengarkan suara yang dihasilkan dengan menggunakan speaker aktif.

V. TUGAS DAN ANALISA

1. Dapatkan nilai-nilai frekuensi penyusun nada DTMF, dan selanjutnya coba bangkitkan sinyal dual tone untuk angka 0,1, 2, dst.

2. Pada proses pembangkitan swept sinusoid, anda coba menghaluskan step kenaikan dari 20

Hz menjadi 10 Hz, dan setiap nilai frekuensi sinyal terbangkit diberikan waktu pembangkitan selama 100 sampel. Perhatikan hasil yang anda peroleh, dan bandingkan dengan proses pembangkitan swept sinusoid yang telah anda lakukan sebelumnya.

MODUL TAMBAHAN II

SISTEM FILTER KASKADE DENGAN TMS320C6713

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Siswa mampu menjelaskan cara membangun sebuah IIR filter dengan struktut kaskade

II. STRUKTUR KASCADE PADA SISTEM LTI

Di dalam suatu hubungan system kaskade, output dari system pertama merupakan input bagi system kedua, dan output dari keseluruhan system kaskade didapatkan dari output system kedua.

Gambar 1. Strukur kaskade LTI

Sifat-sifat kumutatif dan asosiatif pada operasi konvolusi bisa sangat bermanfaat untuk menyelesaikan persoalan kaskade. Ekspresi matematik untuk output adalah:

$$\begin{aligned} y(n) &= (x(n)*h_1(n))*h_2(n) \\ &= x(n)*(h_1(n)*h_2(n)) \\ &= x(n)*(h_2(n)*h_1(n)) \end{aligned} \quad (1)$$

2.1. Respon Frekuensi Sistem LTI Kaskade

Jika diketahui input system $x(n) = e^{j\hat{\omega}n}$ $x(n) = e^{j\hat{\omega}n}$, maka outputnya pada sistem kaskade bagian pertama adalah

$$w(n) = H_1(\hat{\omega}).e^{j\hat{\omega}n} \; w(n) = H_1(\hat{\omega}).e^{j\hat{\omega}n} \quad (2)$$

Output system kaskade bagian kedua adalah:

$$y_2(n) = H_2(\hat{\omega})\left(H_1(\hat{\omega}).e^{j\hat{\omega}n}\right) \quad y_2(n) = H_2(\hat{\omega})\left(H_1(\hat{\omega}).e^{j\hat{\omega}n}\right)$$
$$= H_2(\hat{\omega})H_1(\hat{\omega}).e^{j\hat{\omega}n} \quad = H_2(\hat{\omega})H_1(\hat{\omega}).e^{j\hat{\omega}n}$$

(3)

2.2. Realisasi Diagram Blok Struktur Kaskade Filter

Untuk sebuah system filter IIR order 2 dalam struktur kaskade bisa disusun dari struktur dasar direct form II. Fungsi transfer H(z) di dalam terminology struktur kaskade system berorde2 dapat dituliskan sebagai berikut:

$$H(z) = \prod_{i=1}^{\frac{N}{2}} \frac{a_{0i} + a_{1i}z^{-1} + a_{2i}z^{-2}}{1 + b_{1i}z^{-1} + b_{2i}z^{-2}} \qquad (4)$$

Dalam hal ini N menentukan order filter. Untuk kasus order filter N=4, satu contoh fungsi transfer sistek kaskade bisa dituliskan seperti berikut.

$$H(z) = \frac{(a_{01} + a_{11}z^{-1} + a_{21}z^{-2})(a_{02} + a_{12}z^{-1} + a_{22}z^{-2})}{(1 + b_{12}z^{-1} + b_{21}z^{-2})(1 + b_{12}z^{-1} + b_{22}z^{-2})} \qquad (5)$$

Untuk realisasi filternya dilakukan dengan sebuah diagram blok seperti berikut:

Gambar 2. Struktur Kaskade 2 tingkat untuk system order 4

Dari gambar tersebut bisa dilihat bahwa output dari bagian pertama (stage 1), yang dalam hal ini dinotasikan sebagai $y_1(n)$ akan berfungsi sebagai output pada bagian kedua (stage 2).

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:

- 1 set PC yang dilengkapi dengan software Code Composer Studio.

- 1 set DSK TMS320C5402
- 1 set Speaker Active
- Function Generator
- Oscilloscope

Gambar 3. Penataan peralatan percobaan

IV.PERCOBAAN

4.1. Menyusun Project IIR Filter Kaskade dengan DSK Board

1. Buat sebuah project baru dengan nama IIR_cas.pjt

2. Buat sebuah program dalam bahasa IIR_cas.c , dengan menggunakan editor CCS yang sudah anda aktifkan dan simpan pada folder IIR_cas.pjt.

```
/IIR_cas.c IIR filter menggunakan cascaded Direct Form II
//Koefisien-2 a dan b berkaitan dengan nilai-nilai a dan b dari MATLAB

#include "DSK6713_AIC23.h"             //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;     //set sampling rate
#include "lp2000.cof"            //memanggil koefisien filter low pass
short dly[stages][3] = {0};      //inisialisasi element delay

interrupt void c_int11()    //ISR
{
 short   i, input;
 int     un, yn;

 input = input_sample();         //input to 1st stage


for (i = 0; i < stages; i++)     //repeat for each stage
   {
         un=input-((b[i][0]*dly[i][0])>>15) - ((b[i][1]*dly[i][1])>>15);
         yn=((a[i][0]*un)>>15)+((a[i][1]*dly[i][0])>>15)+
         ((a[i][2]*dly[i][1])>>15);

         dly[i][1] = dly[i][0];   //update delays
         dly[i][0] = un;             //update delays
         input = yn;        //intermediate output->input to next stage
   }

   output_sample((short)yn);          //output final result for time n
   return;                        //return from ISR
}

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

3. Susun sebuah file untuk koefisien, jika anda belum tahu dengan yangakan anda rencanakan, tidak ada salahnya untuk menyusun seperti yang ada di bawah ini.
   //lp2000.cof IIR lowpass coefficients file, cutoff frequency 2 kHz

```
#define stages 4                //number of 2nd-order stages

int a[stages][3] =     {   //numerator coefficients
{304, 608, 304},           //a10, a11, a12 for 1st stage
{32768, 66530, 33778},     //a20, a21, a22 for 2nd stage
{32768, 65518, 32765},     //a30, a31, a32 for 3rd stage
{32768, 64542, 31788}  };  //a40, a41, a42 for 4th stage

int b[stages][2] =     {   //denominator coefficients
{0, 318},                  //b11, b12 for 1st stage
{0, 3015},                 //b21, b22 for 2nd stage
{0, 9362},                 //b31, b32 for 3rd stage
{0, 22070}             };  //b41, b42 for 4th stage
```

4. Anda tambahkan file **IIR_cas.c** ke dalam **IIR_cas.pjt** melalui prosedur yang benar, yaitu **Add File to Project**, dst.
5. Lakukan hal yang sama untuk file koefisien filter **LP2000.cof**.
6. Ulangi langkah 3 untuk file-file seperti : **dsk6713_aic23.h, c6713dskinit.c, Vectors_intr.asm, csl6713.lib, rts6700.lib**, dan **dsk6713bsl.lib**.
7. Lakukan **Scan All file Dependencies** untuk mengetahui file-file terkait lainnya yang diperlukan.
8. Lakukan pengesetan **build option** untuk menentukan lokasi Debug, dsb.
9. Load output program ke DSK board.

4.2. Pengujian dengan Sinyal Sinus

1. Lakukan penataan peralatan (seperti yang dijelaskan dosen pengampu/asisten). Jangan lupa anda melakukan pengecekan sinyal yang dihasilkan oleh function generator dengan menentukan nilai Vpp sebesar 100 mVolt.

2. Lakukan perubahan nilai frekuensi pada function generator untuk mengetahui apakah sinyal outputnya sudah benar. Dalam hal ini tampilan pada oscilloscope akan berubah menyesuaikan perubahan yang dilakukan pada nilai frekuensi pada function generator.

3. Jalankan program yang ada pada CCS, dan masukkan sinyal output dari function generator ke DSK board melalui line-in. Anda lihat output yang dihasilkan oleh DSP board dengan menghubungkan oscilloscope dengan Line Out pada DSP Board.

4. Lakukan perubahan nilai frekuensi pada function generator dan catat nilai-nilai seperti yang dibutuhkan pada table 1.

5. Buat gambar pada beberapa sampel bentuk sinyal output yang dihasilkan pada oscilloscope untuk beberapa nilai frekuensi.

Tabel 1. Hasil pengukuran LPF
Fs=8000 Hz

No	ω (freq normalized)	f (dalam Hz)	V_Out	V_In/V_Out	20 log (V_In/V_Out)
1	0.1				
2	0.2				
3	0.30				
4	0.40				
5	0.50				
6	0.60				
7	0.70				
8	0.70				
9	0.80				
10	0.90				

6. Anda lakukan perhitungan menggunakan calculator untuk konversi dari ω menjadi f (Hz).
7. Buat grafik dari hasil perbandingan V_Out/V_In dan 20log(V_Out/V_In) sebagai fungsi dari nilai frekuensi yang bervariasi dari 0.1 sampai 1.0. Dalam hal ini anda harus memisahkan kedua grafik tersebut, karena tidak mungkin mendapatkan gabungan dua grafik dalam skala yang berbeda.

V. TUGAS DAN ANALISA

1. Pada tugas mendisain filter low-pass 2 KHz, bandingkan plot grafik respon frekuensi filter pada Matlab dengan Tabel 1. Lakukan hal yang sama untuk High Pass Filter

MODUL TAMBAHAN III

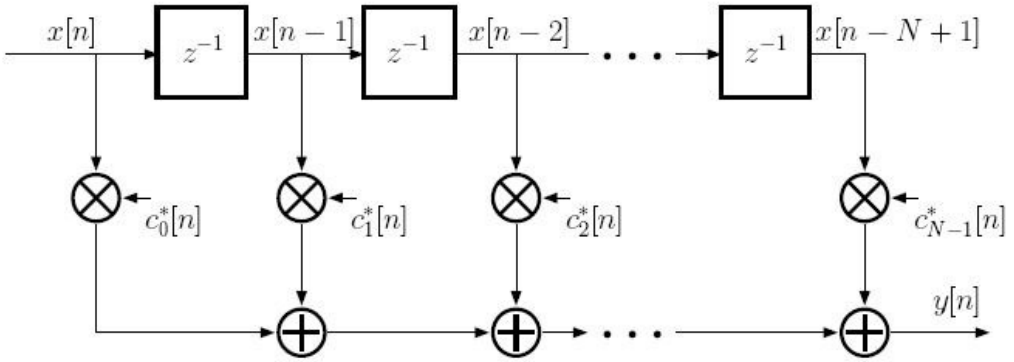
ADAPTIF FILTER FIR UNTUK NOISE CANCELLATION

I. TUJUAN INSTRUKSIONAL

Setelah menyelesaikan praktikum ini diharapkan:
- Siswa memahami memahami prinsip kerja filter adaptif
- Siswa mampu menyusun sebuah filter adaptif sederhana untuk proses Noise Cancellation pada DSK TMS320C6713

II. ADAPTIF FILTER

Di dalam digital filter konvensional seperti FIR dan IIR, diasumsikan bahwa parameter proses untuk menentukan karakteristik sudah diketahui. Mungkin saja mereka memiliki variasi terhadap waktu, tetapisecara alamiah variasinya diasumsikan dalam kondisi yang bisa diketahui. Di dalam beberapa masalah praktis, akan terjadi suatu ketidak pastian dalam beberapa parameter-parameter tersebut karena ketidak sesuaian data pengujian dengan proses yang terjadi. Beberapa parameter mungkin diharapkan untuk berubah sesuai dengan perubahan waktu, tetapi secara alamiah perubahan yang terjadi tidak dapat diprediksi. Di dalam sejumlah kasus sangat diinginkan untuk merancang suatu filter yang mampu melakukan pembelajaran sendiri (*self-learning*) sedemikian hingga filter tersebut dapat beradaptasi dengan sistuasi yang ditanganinya.



Gambar 1. Contoh FIR filter

Suatu catatan yang perlu kita cermati dalam konteks pembuatan filters adaptive, yang dahal hal ini adalah bagaimana mendapatkan nilai koefisien-koefisien filter yang mampu menyesuaikan dengan algorithma yang digunakan dan mengatasi masalah yang berkaitan dengan hal-hal yang tidak kita inginkan seperti sinyal gangguan dengan cirri yang kurang jelas

atau gangguan pada system lainnya. Adaptive filters dapat digunakan untuk mengatur koefisien filter pada waktu lingkungan pengganggu tidak diketahui secara pasti dan berfvariasi sebagai fungsi waktu.

Di dalam suatu transversal filter atau yang mungkin anda kenal sebagai FIR filter dengan jumlah element sepanjang N, seperti diberikan pada Gambar 1, pada setiap waktu n, sampel output y[n] dikomputasi sebagai suatu jumlahan terbobot pada sampel input yang sekarang dan sampel-sample input di waktu sebelumnya:

$$y[n] = \sum_{k=0}^{N-1} c_k^*[n]x[n-k] \qquad (1)$$

Dimana $c_k[n]$: adalah koefisien filter yang bersifat tergantung kepada waktu (kita gunakan complex conjugated coefficients $c^*_k[n]$ sehingga turunannya pada algorithma adaptifnya bisa valid untuk sinyal-sinyal komplek). Persamaan ini ditulis ulang sebagai bentuk vector, menggunakan $x[n] = [x[n], x[n-1], \ldots. x[n-N+1]\ ]^T$ yang merupakan tap-input vector pada waktu n, dan $c[n] = [c_0[n], c_1[n], \ldots. c_{N-1}[n]\ ]^T$ merupakan koefisien vector pada waktu n, sedemikian hingga didapatkan bentuk seperti berikut:

$$y[n] = c^H[n]x[n] \qquad (2)$$

Keduanya, baik x[n] maupun c[n] adalan vector kolom sepanjang N. $c^H[n] = (c^*)^T[n]$ adalah hermitian pada vektor c[n] (setiap elemen dikonjugatkan dengan *, dan kolom vector adalah transpose T menjadi suatu vector baris)

Koefisien-koefisien pda suatu adaptif filter ditata untuk melakukan kompensasi terhadap perubahan sinyal input, sinyal output, atau paratemer system. Dalam kondisi yang rigid, suatu system dapat melakukan pembelajaran pada karakteristik sinyal dan melakukan traking pada perubahan yang lambat. Suatu adaptif filter dapat sangat bermanfaat ketika terjadi ketidak tentuan pada karakteristik sinyal ketika karakteristik-karakteristik ini berubah.

Gambar 2. Struktur dasar Adaptive Filter

Secara konsep, skema adaptif ini sukup sederhana. Sebagian besar skema adaptif dapat digambarkan dengan suatu struktur seperti pada Gambar 2. Ini merupakan suatu struktur dasar adaptif filteryang mana output adaptif filter ***y*** dibandingkan dengan desired sinyal (yang diingikan) ***d*** untuk menghasilkan suatu sinyal error ***e***. Sinyal error merupakan input untuk

algorithma adaptif , yang mana akan meng-adjust variable filter untukm emenuhi beberapa criteria atau rule yang sudah ditentukan sebelumnya. Sinyal yangd iinginkan bisanya sangat sulit untuk ditentukan. Satu pertanyaan yang mungkin muncul di dalam benak kita adalah: Mengapa kita mencoba untuk membangkitkan sinyal yang diinginkan pada y jika sudahmengetahui sinyal tersebut? Secara mengejutkan kejadian di dalam alam menunjukkan bahwa sinyal yang diinginkan tidak diketahui di dalam adaptif filter.

Koefisien-koefisien pada adptif filter bisa di-adjust, atau dioptimalkan, menggunakan algorithma LMS (*least mean square*) yang didasarkan pada sinyal *error*. Disini kita hanya membicarakan algorithma pemcarian LMS dengan suatu *linear combiner* (FIR Filter), walaupun sebenarnya ada beberapa strategi untuk membentuk suatu pemfilteran secara adaptif. Output dari adaptive filter pada Gambar 1 diatas adalah.

$$y(n) = \sum_{k=0}^{N-1} w_k(n)x(n-k) \qquad (2)$$

Dimana $w_k(n)$ merepresentasikan N bobot atau koefisien untuk suatu waktu spesifik n. Persamaan konvolusi (1) telah diimplementasikan pada filter FIR yang standar. Praktik yang lebih umum untuk penentuan nilai bobot pada koefisien adaptif filter adalah berkaitan dengan masalah adaptif filtering dan neural network.

Suatu ukuran kinerja yang diperlukan untuk seberapa bagus suatu filter. Ukuran ini disarakan pada sinyal error seperti berikut:

$$e(n) = d(n) - y(n) \qquad (3)$$

yang merupakan perbedaan nilai antara sinyal yang diinginkan d(n) dengan output pada adaptif filter $y(n)$. Bobot pada koefisien $w_k(n)$ di-adjust sedemikian hingga suatu fungsi *mean squarer error* bisa diminimisasi. Fungsi *mean squarer error* ini adalah $E[e^2(n)]$, dimana E merepresentasikan nilai ekspektasi. Disini ada sebanyak k bobot atau koefisien, sehingga diperlukan suatu gradient pada fungsi *mean squared error*. Suatu estimasi dapat diperoleh di dalam penggunaan gradian $e^2(n)$, menghasilkan:

$$w_k(n+1) = w_k(n) + 2\beta e(n)x(n-k) \qquad k = 0,1,\ldots N-1 \qquad (4)$$

yang mana merepresentasikan algorithma LMS. Persamaan diatas, memberikan sesuatu yang sederhana tetapi sangat bermanfaat dan efisien dalam melakukan proses update bobot, atau koefisien, tanpa perlu untuk proses averaging atai diferensiasi, dan akan digunakan unuk implementasi adaptif filter. Input untuk adaptif filter adalah x(n), dan *rate of convergence* dan akurasi pada proses adaptasi (adaptive step size) adalah β.

Untuk setiap waktu spesifik n, setiap koefisien atau bobot $w_k(n)$ di update atau digantikan dengan suatu koefisien baru, didasarkan pada persamaan (3), jika tidak terjadi maka kondisinya $e(n)$ sudah bernilai nol. Setelah output filter $y(n)$, sinyal error $e(n)$, dan setiap koefisien $w_k(n)$ di-update untuk waktu tertentu n, suatu sampel baru (dari ADC) diperoleh dan proses adaptasi diulang lagi untuk waktu yang berikutnya. Pada suatu kondisi dimana $e(n)$ sudah bernilai nol, proses update tidak terjadi lagi.

Linear adaptive combiner merupakan satu struktur adaptif filter yang paling bermanfaat dan merupakan suatu FIR filter yang adjustable. Dalam hal ini koefisien-koefisien, atau bbot, pada adaptive filter FIR dapat di-adjust berdasarkan pada suatu perubahan lingkungan sebagai sinyal input. Adaptive filter IIR, juga bisa digunakan, tetapi dalam hal ini akan terjadi proses peng-update-an pada pole-pole yang mana akan perlu pertimbangan pada kestabilan filter IIR, sebab bisa saja terjadi suatu kondisi dimana posisi pole berada diluar unit cirle bidang z sehingga system menjadi tidak stabil.

2.1. Struktur Adaptive Filter untuk Noise Cancellation

Dengan memodifikasi bentuk dasar pada Gambar1 kita akan mampu menyusun sebuah adaptif filter untuk proses noise cancellation seperti pada Gambar 3.

Gambar 3. Noise Cancellatiion dengan Adaptive Filter

Dimana

d =desired signal
n = uncorrelated additive noise
n' = noise input untuk adaptive filter yang berkorelasi dengan noise n

Sinyal yang diinginkan (*desired signal*) *d* mengalami gannggguan oleh suatu *uncorrelated noise n*. Input pada adaptive filter berupa suatu noise n' yang berkorelasi dengan noise *n*. Noise *n'* bisa berasal dari sumber yang sama dengan *n* tetapi telah dimodifikasi oleh lingkungan (*environment*). Output adaptive filter *y* digunakan untuk pembanding noise. Ketika ini terjadi, sinyal error yang dihasilkan mendekati sinyal yang diinginkan (desired signal) *d*. Output keseluruhan system adalah sinyal error dan bukan output dari adaptive filter *y*. Jika *d* dalam kondisi un-correlated dengan n, strateginya adalah untuk meminimisasi $E(e^2)$, dimana E() adalah nilai ekspektasi pertama. Nilai ekspektasi pertama secar umum tidak diketahui, sehingga biasanya didekati denagn suatu running average atau fungsi sesaat itu sendiri. Komponen sinyal $E(d^2)$, tidak akan terpengaruh, dan hanya komponen noise $E[(n-y)^2]$ yang akan diminimisasi.

III. PERANGKAT PERCOBAAN

Dalam pelaksanaan praktikum ini diperlukan perangkat percobaan berupa:
- PC yang memiliki spesifikasi memory minimal 1 Gb
- Operating System minimal Windows Xp
- Code Composser Studio
- Sistem Audio untuk PC

IV. PERCOBAAN

Di dalam percobaan ini, kita akan menyusun sebuah adaptif filter yang cara kerjanya didasarkan pada Gambar 2. Dalam hal ini kita memanfaatkan algorithma LMS untuk meng-cancel noise (sinyal sinus yang tidak singinkan) yang dalam hal ini adalah *undesirable sinusoid*. Sinyal yang diinginkan berupa sinus dengan frekuensi 1500 Hz (*desired signal* d), dan sinyal yang tidak diinginkan berupa sinyal sinus dengan frekuensi 321 Hz yang dalam hal ini sebagai sinyal pengganggu (*undesired signal*, n), dan sinyal yang lain n' digunakan sebagai input pada *adaptive filter*. Suatu *reference* (*template*) sinyal cosine dengan frekuensi 312 ditetapkan sebagai n', merupakan input bagi adaptive filter yang tersusun dari 30-tap koefisien filter FIR. *Reference signal cosine* 312 Hz dikorelasikan dengan *adaptive noise* 312 Hz (sinus), tetapi bukan dengan desired signal d, 1500 Hz. Untuk setiap waktu n(sample) output pada adaptive filter FIR dihitung dan 30 bobot atau 30 koefisien di-*update* sepanjang *delay* sampel. Sinyal *error* E, adalah output keseluruhan sistem struktur adaptive.

4.1. Pembangkitan Sinyal-sinyal noise reference, sinus table, dan dplus noise dengan Matlab

1. Buat program Matlab untuk membangkitkan sinyal-sinyal yang dibutuhkan diatas. Dalam hal ini anda bias memanfaatkan listing program berikut ini.

```
%Adaptnoise.M Generates:
%dplusn.h (s312+s1500),
%refnoise.h cos(312),and
%sin1500.h

for i=1:128
  desired(i) = round(100*sin(2*pi*(i-1)*1500/8000)); %sin(1500)
  addnoise(i) = round(100*sin(2*pi*(i-1)*312/8000)); %sin(312)
  refnoise(i) = round(100*cos(2*pi*(i-1)*312/8000)); %cos(312)
end
dplusn = addnoise + desired;         %sin(312)+sin(1500)

fid=fopen('dplusn.h','w');     %desired + noise
fprintf(fid,'short dplusn[128]={');
fprintf(fid,'%d, ' ,dplusn(1:127));
fprintf(fid,'%d' ,dplusn(128));
fprintf(fid,'};\n');
fclose(fid);

fid=fopen('refnoise.h','w');           %reference noise
fprintf(fid,'short refnoise[128]={');
fprintf(fid,'%d, ' ,refnoise(1:127));
```

```
fprintf(fid,'%d' ,refnoise(128));
fprintf(fid,'};\n');
fclose(fid);

fid=fopen('sin1500.h','w');            %desired sin(1500)
fprintf(fid,'short sin1500[128]={');
fprintf(fid,'%d, ' ,desired(1:127));
fprintf(fid,'%d' ,desired(128));
fprintf(fid,'};\n');
fclose(fid);
```

2. Amati apakah di dalam folder tempat anda menyimpan program sudah terdapat file-file *.txt seperti berikut ini. File sin 1500Hz dengan nilai-nilai data yang merepresentasikan sinyal sinus 1500Hz yang diinginkan.

```
short sin1500[128]= {0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92, 0, 92, 71, -38, -100, -38, 71, 92,
0, -92, -71, 38, 100, 38, -71, -92};

short dplusn[128]={0, 116, 118, 29, -17, 56, 170, 191,
93, -11, -7, 81, 120, 34, -100, -143, -70, 7, -24, -138, -198, -129,
-7, 32, -39, -108, -62, 71, 155, 111, 17, 5, 100, 189, 160, 37, -43,
-3, 82, 79, -37, -150, -147, -52, 2, -62, -167, -179, -72, 39, 40, -45,
-82, 3, 133, 171, 92, 7, 29, 133, 184, 107, -22, -65, 3, 70, 26, -103,
-182, -131, -28, -7, -93, -174, -137, -7, 78, 40, -45, -43, 68, 176,
166, 62, -1, 54, 150, 154, 41, -74, -77, 8, 47, -34, -157, -188, -100,
-6, -19, -115, -159, -75, 57, 103, 34, -36, 4, 127, 197, 138, 26, -4,
74, 147, 104, -29, -115, -77, 11, 15, -90, -190, -171, -58, 14, -33,
-122, -121};


short refnoise[128]={100, 97, 88, 74, 56, 34, 10, -14, -38, -59, -77,
-90, -98, -100, -96, -86, -71, -52, -30, -6, 19, 42, 63, 80, 92, 99,
100, 95, 84, 68, 48, 25, 1, -23, -46, -66, -82, -94, -99, -99, -93, -81,
-65, -44, -21, 3, 27, 50, 69, 85, 95, 100, 98, 91, 79, 61, 40, 17, -8,
-31, -54, -72, -87, -96, -100, -98, -89, -76, -58, -36, -13, 12, 36, 57,
75, 89, 97, 100, 97, 87, 73, 54, 32, 8, -16, -40, -61, -78, -91, -98,
-100, -95, -85, -70, -50, -28, -4, 21, 44, 64, 81, 93, 99, 99, 94, 83,
67, 47, 24, -1, -25, -48, -68, -83, -94, -100, -99, -92, -80, -63, -43,
-19, 5, 29, 51, 71, 86, 96};
```

3. Jika anda penasaran dengan bentuk sinyal yang dihasilkan, anda dapat memodifikasi program diatas untuk melihat bentuk sinyal masing-masing.

4.2. **Membangun sebuah project Filter Adaptive dengan TMS**

1. Bangun sebuah project baru, anda bias member nama adaptive filter, atau yang lain

2. Susun sebuah program dan beri nama **adaptnoise.c** yang berbasis adaptive filter untuk

suatu tujuan meng-cancel noise. Jika belum punya gambaran program tersebut, anda bisa memanfaatkan listing program berikut ini.

```
//Adaptnoise.c  Adaptive FIR filter for noise cancellation

#include "DSK6713_AIC23.h"
Uint32 fs= DSK6713_AIC23_FREQ_8KHZ;

#include "refnoise.h"           //cosine 312 Hz
#include "dplusn.h"             //sin(1500) + sin(312)
#define beta 1E-10              //rate of convergence
#define N 30                    //# of weights (coefficients)
#define NS 128                  //# of output sample points
float w[N];                     //buffer weights of adapt filter
float delay[N];                 //input buffer to adapt filter
short output;                   //overall output
short out_type = 1;             //output type for slider

interrupt void c_int11()        //ISR
{
 short i;
 static short buffercount=0;    //init count of # out samples
 float yn, E;                   //output filter/"error" signal

 delay[0] = refnoise[buffercount];  //cos(312Hz) input to adapt FIR
 yn = 0;                            //init output of adapt filter
 for (i = 0; i < N; i++)            //to calculate out of adapt FIR
     yn += (w[i] * delay[i]);       //output of adaptive filter

 E = dplusn[buffercount] - yn;      //"error" signal=(d+n)-yn

 for (i = N-1; i >= 0; i--)         //to update weights and delays
   {
     w[i] = w[i] + beta*E*delay[i]; //update weights
     delay[i] = delay[i-1];         //update delay samples
   }
 buffercount++;                     //increment buffer count
 if (buffercount >= NS)            //if buffercount=# out samples


     buffercount = 0;               //reinit count



 if (out_type == 1)                //if slider in position 1

     output = ((short)E*10);        //"error" signal overall output
 else if (out_type == 2)           //if slider in position 2
     output=dplusn[buffercount]*10; //desired(1500)+noise(312)
 output_sample(output);                //overall output result
 return;                        //return from ISR
}

void main()
{
 short T=0;
 for (T = 0; T < 30; T++)
   {
     w[T] = 0;                  //init buffer for weights
     delay[T] = 0;              //init buffer for delay samples
   }
```

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

3. Lakukan pengesetan pada **Build Option**, dst.
4. Simpan program anda, lakukan build, dan langkah-langkah lain yang diperlukan.
5. Coba anda Run program anda,jika ada keselahan tentu saja nada harus membetulkan sampai kondisi program anda benar-benar bebas dari error.
6. Persiapkan peralatan untuk pengujian, dalam hal ini adalah Oscilloscope untuk mengamati bentuk sinyal yang dihasilkan. Anda bisa melihat hasilnya dari *line-out*.
7. Dari file sinus yang ada di lookup table, didapatkan bahwa frekuensi yang terbangkit dengan sin 1500 Hz adalah

   f = Fs(# of cycles)/(# of points) = 8000(24)/(128) = 1500 Hz

8. Jika anda m asih belum paham untuk apa file-file tersebut, maka pada waktu anda melakukan **build** dan **run project,** anda akan mendapatkan verifikasi output berikut ini:

   - Undesired noise 312 Hz secara gradual ter-reduksi (cancell) sementara sinusoida 1500 Hz akan tetap (dipertahankan).

   - Nilai beta yang terlalu besar menyebabkan *rate of convergence* cepat, sehingga proses adaptive tidak tampak, sebab outputnya langsung berupa 1500Hz.

DAFTAR PUSTAKA:

1. ___, "*TMS 32C6713 Reference Manual*", Texas Instrumen, USA 2006.
2. Rulph Chassaing, "*Digital Signal Processing and Applications with the C6713 and C6414 DSK*", John Willey and Sons, New Jersey, USA 2005.
3. Sen M Kuo, Woon-Seng Gan, "*Digital Signal Processors, Architectures, Implementations and Applications*", Person Prentice Hall, USA 2005.
4. James Mc Clelland, "*DSP First, a Multimedia Approach*", Prentice Hall, 1999.
5. Texas Instrument Application Reports SPRA809A, "*How to Begin Development Today With the TMS320C6713Floating Point DSP*", October 2002.
6. Abdullah A Wardak, "*Practical Guidelines and Examples for the Users of the TMS320C6713 DSK*", World Academy of Science, Engineering and Technology 45 2008.
7. D. Richard Brown III, "*Digital Signal Processing and Applications with the TMS320C6713 DSK*", Worcester Polytechnic Institute, 2-days Workshop, October 15-16, 2009.
8. Kwadwo Boateng and Charles Badu , "*Integration of Matlab Tools for DSP Code Generation*", Bradley University, ECE Department, December 6th, 2005
9. ___, "*Matlab Tutorial*", http://www.owlnet.rice.edu/~elec241/matlab.html
10. ___, "*Matlab Tutorial*", http://www.mathworks.com
11. Hary Octavianto, Tri Budi Santoso, Titon Dutono, "*Praktikum Pengolahan Sinyal Digital berbasis TMS320C5402*", Politeknik Elektronika Negeri Surabaya – ITS, 2007.

Bio Data Tim Penyusun

Tri Budi Santoso

Menyelesaikan pendidikan S1 Jurusan Teknik Fisika 1994. Bergabung dengan Politeknik Elektronika Negeri Surabaya, sejak tahun 1995. Tahun 1999 menyelesaikan pendidikan Pasca Sarjana Teknik Elektro, bidang keahlian Telekomunikasi. Bekerja sebagai staf pengajar di Jurusan Teknik Telekomunikasi Politeknik Elektronika Negeri Surabaya untuk mata kuliah Pengolah Sinyal Digital. Pada waktu penyusunan buku ini yang yang bersangkutan sedang mengikuti pendidikan Program S3 di Institut Teknologi Sepuluh Nopermber Surabaya (ITS) bidang Telekomunikasi Multimedia.

Hary Octavianto

Menyelesaikan pendidikan S1 Jurusan Teknik Elektro, program Studi Elektronika pada tahun 2001. Bergabung dengan Politeknik Elektronika Negeri Surabaya, sejak tahun 2001. Bekerja sebagai staf pengajar di Jurusan Elektronika untuk Praktikum Pengolah Sinyal dan Embedded System. Tahun 2008 mengikuti pendidikan S2 di National Technolofy University, Taiwan di bidang Elektronika. Pada tahun 2010 menyelesaikan pendidikan Program S2 di National Taiwan University.

Miftahul Huda

Menyelesaikan pendidikan S1 Jurusan Fisika 1990. Bergabung dengan Politeknik Elektronika Negeri Surabaya, sejak tahun 1991. Tahun 2001 menyelesaikan pendidikan Pasca Sarjana Teknik Elektro, bidang keahlian Teknik Telekomunikasi. Saat ini bekerja sebagai staf pengajar di Jurusan Teknik Telekomunikasi Politeknik Elektronika Negeri Surabaya untuk mata kuliah Pengolah Sinyal Digital.