PETUNJUK PRAKTIS

PENGGUNAAN MOTOROLA DSP56002EVM

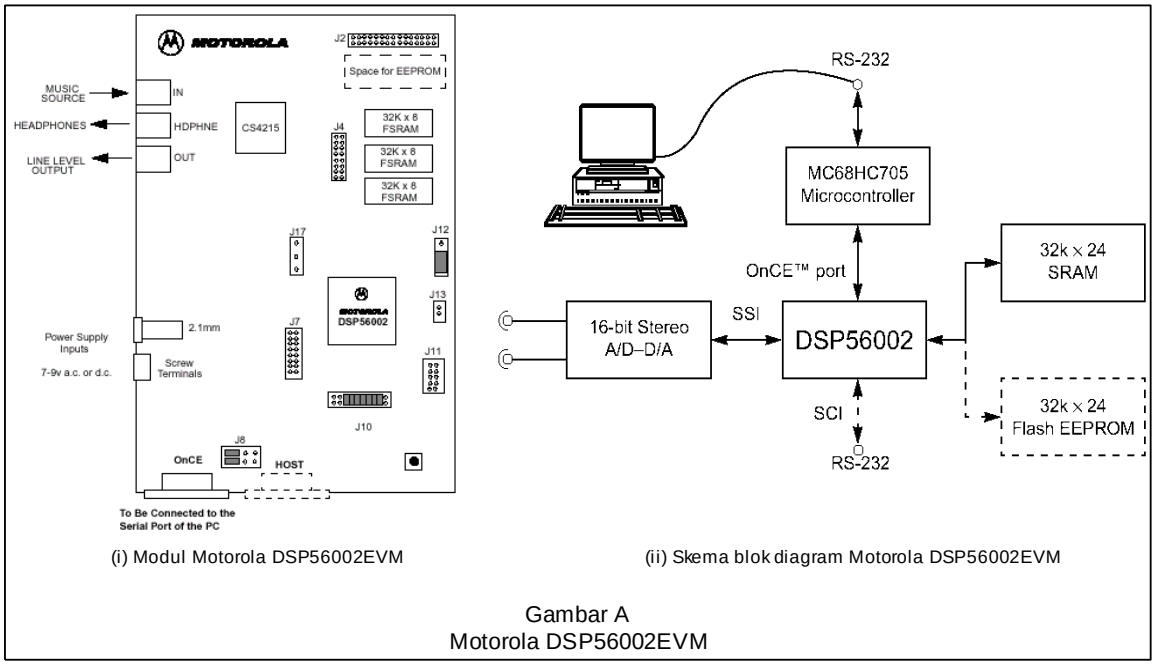
I. INSTALASI SOFTWARE

a. Copy-kan seluruh file yang ada dalam disket (2 disket) ke direktori tertentu dalam hardisk komputer.

b. Instal software untuk mengoperasikan Motorola DSP56002EVM dengan menjalankan file install.exe dan tempatkan file instalasi pada direktori tertentu (sebaiknya ditempatkan pada direktori yang sama seperti langkah a).

c. Ekstak file evm28.exe dengan menjalankan file tersebut dan tempatkan di direktori tertentu (sebaiknya ditempatkan pada direktori yang sama seperti langkah a).

d. Software Debugger Motorola DSP56002EVM siap digunakan.



(i) Modul Motorola DSP56002EVM (ii) Skema blok diagram Motorola DSP56002EVM

Gambar A
Motorola DSP56002EVM

II. INSTALASI HARDWARE

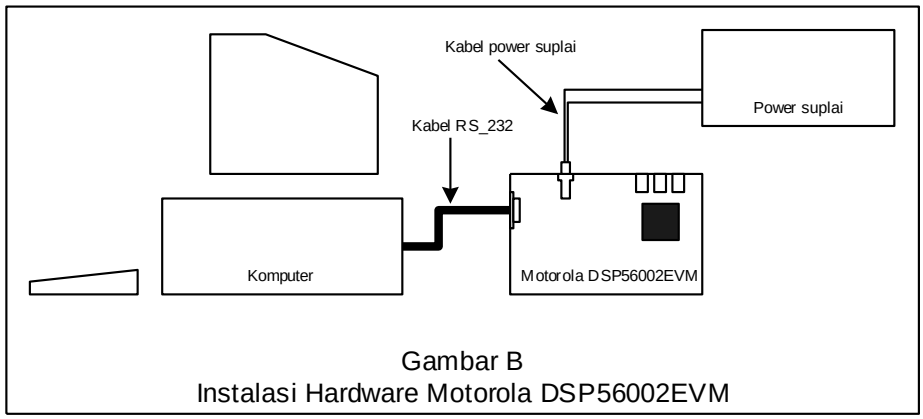
a. Siapkan satu set komputer dengan software Debugger Motorola DSP56002EVM yang telah terinstal, modul Motorola DSP56002EVM, power suplai AC atau DC, kabel penghubung RS-232 dan kabel penghubung power suplai.

b. Hubungkan modul Motorola DSP56002EVM ke power suplai 7 - 9 volt (AC atau

DC) dengan arus 700 mA menggunakan kabel penghubung power suplai.

c. Hubungkan modul Motorola DSP56002EVM ke komputer (yang telah terinstal software Debugger Motorola DSP56002EVM) menggunakan kabel penghubung RS-232.
d. Nyalakan komputer dan power suplai (modul Motorola DSP56002EVM telah mendapat suplai tegangan ditandai dengan menyalanya LED warna hijau pada modul Motorola DSP56002EVM).
e. Jalankan software Debugger Motorola DSP56002EVM (modul Motorola DSP56002EVM telah terhubung dengan komputer ditandai dengan menyalanya LED warna merah pada modul Motorola DSP56002EVM, sebaliknya bila LED warna merah pada modul Motorola DSP56002EVM tidak menyala berarti Motorola DSP56002EVM belum terhubung dengan komputer dan pada monitor komputer ditampilkan pesan kesalahan).

a. Motorola DSP56002EVM siap digunakan.



Gambar B
Instalasi Hardware Motorola DSP56002EVM

III. PEMAKAIAN DEBUGGER MOTOROLA DSP56002EVM

a. Buka software Debugger Motorola DSP56002EVM.
b. Load file program yang akan dijalankan (ber-ektensi *.cld).
c. Click icon RUN (untuk menjalankan program).

IV. PEMBUATAN PROGRAM

a. Tulis program dalam bahasa pemrograman assembly untuk Motorola DSP56002EVM pada salah satu fasilitas word processor (seperti : notepad, wordpad, MS word atau program lainnya).

b. Simpan program yang telah dibuat dengan ektensi *.asm.

c. Compile program melalui fasilitas DOS dengan perintah :

C:\>ASM56000 -a -b -l nama_file.asm

setelah proses compile selesai akan dihasilkan file ber-ektensi *.cld dan *.lst dengan nama file yang sama.

d. Program siap dipakai.

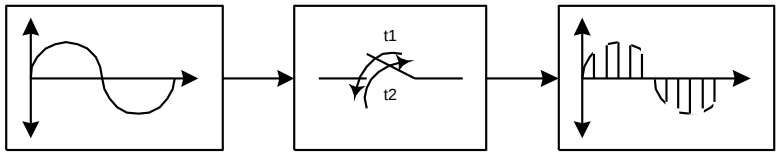
MODUL 1
SINYAL SAMPLING

TUJUAN

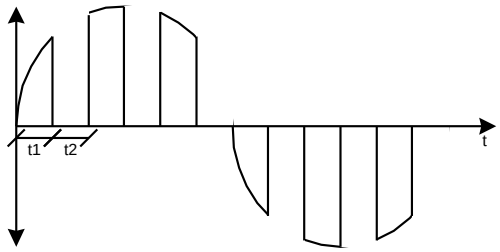
1. Mempelajari proses sampling sinyal menggunakan Motorola DSP56002EVM.
2. Mempelajari pengaruh perubahan frekuensi sampling.

TEORI PENGANTAR

Sinyal sampling didapatkan dengan melewatkan sinyal analog pada suatu saklar (switch) yang dibuka dan ditutup dengan interval waktu tertentu secara terus menerus. Lama waktu saklar menutup adalah t_1 dan lama waktu saklar membuka adalah t_2 sehingga periode sampling adalah $t_1 + t_2$.



a. Proses pembangkitan sinyal sampling dengan switch



b. Sinyal sampling

Gambar 1
Proses pembangkitan sinyal sampling

Sinyal sampling juga bisa didaptkan dengan mengalikan sinyal analog dengan sinyal pulsa. Misalkan persamaan untuk sinyal analog adalah $f(t)$ dan persamaan untuk sinyal pulsa adalah

$$\delta_{\infty}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

sehingga persamaan untuk sinyal sampling adalah

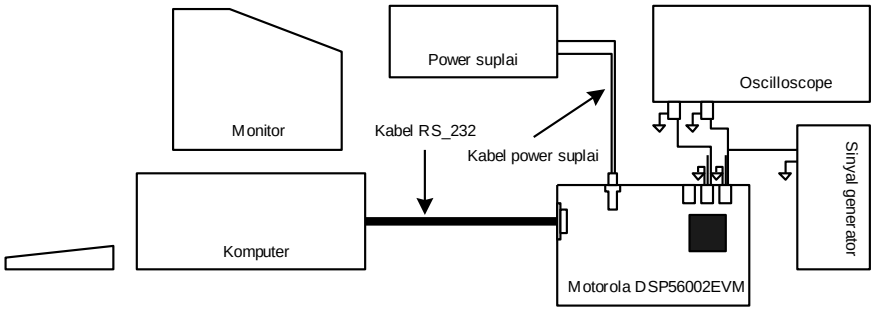
$$f_s(t) = f(t)\sum_{n=-\infty}^{\infty}\delta(t-nT)$$

PERALATAN YANG DIGUNAKAN

1. Satu set komputer.
2. Motorola DSP56002EVM.
3. Power suplai AC atau DC.
4. Sinyal generator.
5. Oscilloscope.

LANGKAH PERCOBAAN

1. Rangkai peralatan seperti pada gambar berikut



2. Nyalakan semua peralatan yang dipakai (komputer, power suplai, oscilloscope dan sinyal generator).
3. Set sinyal generator pada frekuensi 1 KHz.
4. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
include   'awal.asm'              ;program inisialisasi awal

delay     dc      1               ;frekuensi sampling

loop
          do      #delay,end1
          jsr     input           ;sinyal disimpan di akumulator a dan b
          nop                     ;no operation
```

```
        nop                     ;no operation
        jsr     output          ;sinyal yang akan dikeluarkan harus ditempatkan
                                ;terlebih  dahulu pada akumulator a dan b
end1
        do      #delay,end2
        jsr     input
        clr     a
        clr     b
        jsr     output
end2
        jmp     loop

        include 'akhir.asm'     ;program inisialisasi akhir

        end                     ;akhir program
```

4. Simpan program dengan nama file ber-ektensi asm (*.asm).
5. Compile program untuk mendapatkan file ber-ektensi cld (*.cld), buka Debugger Motorola DSP56002EVM, load program dan jalankan Motorola DSP56002EVM.
6. Amati sinyal sampling pada oscilloscope.
7. Ulangi langkah 4 sampai 6 untuk frekuensi sampling sebagai berikut :

Frekuensi (Hz)	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
Catatan	Nilai delay 25 mewakili frekuensi sampling 1000 Hz Nilai delay 50 mewakili frekuensi sampling 500 Hz dst.									

TUGAS ANALISA

1. Jelaskan teori sampling.
2. Jelaskan akibat yang terjadi karena proses sampling.
3. Apa pengaruh frekuensi sampling pada sinyal yang dihasilkan.
4. Apa yang terjadi jika frekuensi sampling tidak memenuhi teori sampling.
5. Hitung frekuensi sampling.
6. Frekuensi berapa yang terbaik.

TUGAS TERAPAN.

1. Sampling sinyal audio dan bandingkan hasilnya dengan aslinya.

MODUL 2
SINYAL MODULASI

TUJUAN

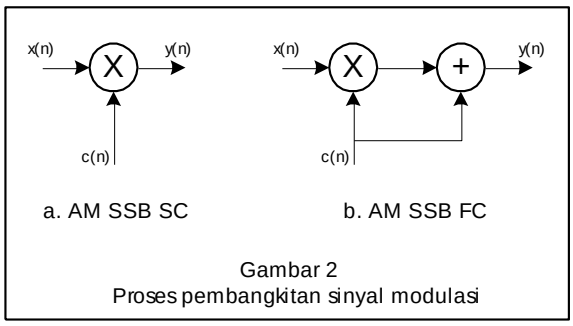
1. Mempelajari proses modulasi sinyal menggunakan Motorola DSP56002EVM.
2. Mempelajari pengaruh modulasi amplitudo pada sinyal informasi.

TEORI PENGANTAR

Modulasi adalah proses pencampuran dua sinyal menjadi satu sinyal. Biasanya sinyal yang dicampur adalah sinyal berfrekuensi tinggi dan sinyal berfrekuensi rendah. Dengan memanfaatkan karakteristik masing-masing sinyal, maka modulasi dapat digunakan untuk mentransmisikan sinyal informasi pada daerah yang luas atau jauh.

Contoh:

Modulasi amplitudo didapatkan dengan cara mengalikan sinyal informasi dengan sinyal pembawa, seperti yang terlihat pada gambar 2.



Gambar 2
Proses pembangkitan sinyal modulasi

Misalkan sinyal informasi mempunyai persamaan sebagai berikut

$$x(n) = A_x \sin(\omega_x t)$$

dan persamaan untuk sinyal pembawa adalah

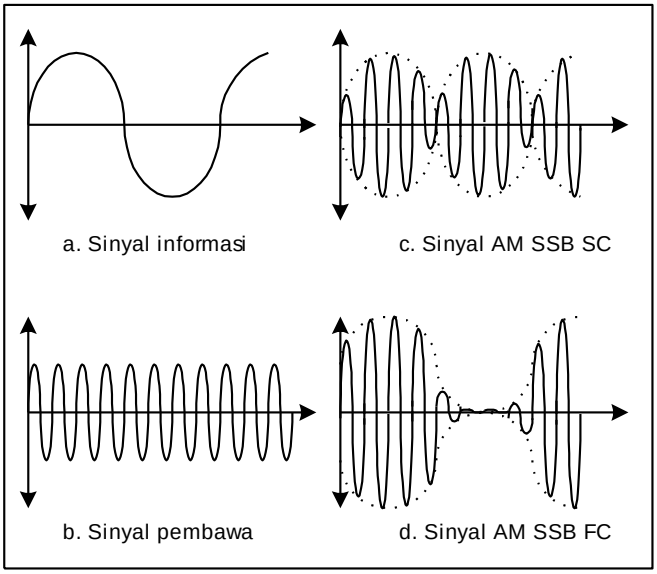
$$c(t) = A_c \sin(\omega_c t)$$

maka persamaan untuk amplitudo modulasi single side band supressed carrier (AM SSB SC) adalah

$$y(t) = A_x A_c \sin(\omega_x t) \sin(\omega_c t)$$

sedangkan persamaan untuk AM SSB full carrier adalah

$$y(t) = A_c \sin(\omega_c t)\left[A_x(\sin(\omega_x t) + 1\right]$$

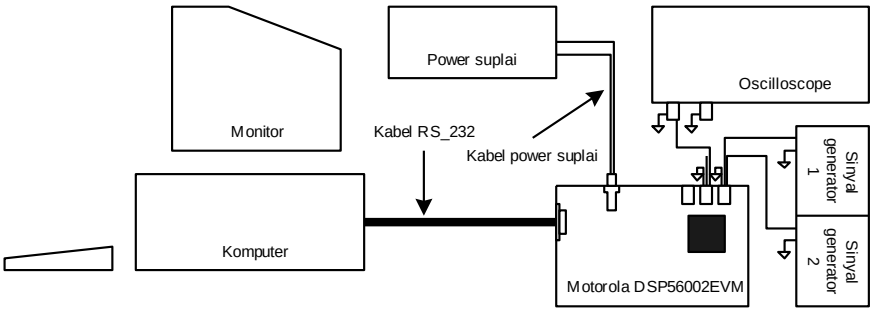


PERALATAN YANG DIGUNAKAN

1. Satu set komputer.
2. Motorola DSP56002EVM.
3. Power suplai AC atau DC
4. Signal generator.
5. Oscilloscope.

LANGKAH PERCOBAAN

1. Rangkai peralatan seperti pada gambar berikut :



2. Nyalakan semua peralatan yang dipakai (komputer, power suplai, oscilloscope dan

sinyal generator).

3. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
sincosr   macro   points,coef              ;fungsi untuk membangkitkan sinyal sinusoidal
pi        equ     3.141592654
freq      equ     2.0*pi/@cvf(points)

          org     x:coef                   ;menemmpatkan nilai sinus di memory X
count     set     0
          dup     points
          dc      0.999*@sin(@cvf(count)*freq)
count     set     count+1
          endm                             ;akhir proses penempatan

          org     y:coef                   ;menemmpatkan nilai sinus di memory Y
count     set     0
          dup     points
          dc      0.999*@sin(@cvf(count)*100*freq)
count     set     count+1
          endm                             ;akhir proses penempatan

          endm                             ;akhir fungsi

points    equ     512                      ;variabel fungsi
coef      equ     $0                       ;variabel fungsi
          sincosr points,coef              ;menjalankan fungsi

          include 'awal.asm'               ;program inisialisasi awal

loop
          move    #0,r1                    ;set register R1 dengan nilai 0
          do      #512,end1
          move    x:(r1),x0
          move    y:(r1)+,y0
          mpy     x0,y0,b
          tfr     b,a
          jsr     output                   ;sinyal yang akan dikeluarkan harus ditempatkan
                                           ;terlebih dahulu pada akumulator a dan b
end1
          jmp     loop

          include 'akhir'                  ;program inisialisasi akhir

          end                              ;akhir program
```

(program untuk membangkitkan sinyal modulasi amplitudo AM SSB SC dimana kedua sinyal yang akam dimodulasi dibangkitkan oleh Motorola DSP56002EVM).

4. Simpan program dengan nama file ber-ektensi asm (*.asm).
5. Compile program untuk mendapatkan file ber-ektensi cld (*.cld), buka Debugger

Motorola DSP56002EVM, load program dan jalankan Motorola DSP56002EVM.

6. Amati sinyal pada oscilloscope, dan catat frekuensi masing-masing sinyal.
7. Set sinyal generator pada frekuensi 1 Khz.
8. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
sincosr   macro   points,coef              ;fungsi untuk membangkitkan sinyal sinusoidal
pi        equ     3.141592654
freq      equ     2.0*pi/@cvf(points)

          org     y:coef                   ;menemmpatkan nilai sinus di memory Y
count     set     0
          dup     points
          dc      0.999*@sin(@cvf(count)*100*freq)
count     set     count+1
          endm                             ;akhir proses penempatan

          endm                             ;akhir fungsi

points    equ     512                      ;variabel fungsi
coef      equ     $0                       ;variabel fungsi
          sincosr points,coef              ;menjalankan fungsi

          include 'awal.asm'               ;program inisialisasi awal

loop
          move    #0,r1                    ;set register R1 dengan nilai 0
          do      #512,end1
          jsr     input                    ;sinyal disimpan di akumulator a dan b
          move    a,x0
          move    y:(r1)+,y0
          mpy     x0,y0,b
          tfr     b,a
          jsr     output                   ;sinyal yang akan dikeluarkan harus ditempatkan
                                           ;terlebih dahulu pada akumulator a dan b
end1
          jmp     loop

          include 'akhir'                  ;program inisialisasi akhir

          end                              ;akhir program
```

(program untuk membangkitkan sinyal modulasi amplitudo AM SSB SC dimana sinyal yang akam dimodulasi dibangkitkan oleh Motorola DSP56002EVM dan sinyal generator).

9. Ulangi langkah 4 dan 5.
10. Amati sinyal pada oscilloscope.
11. Set sinyal generator 1 pada frekuensi 500 Hz, dan sinyal generator 2 pada 1khz.

12.Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
                include  'awal.asm'              ;program inisialisasi awal

loop
                move     #0,r1                   ;set register R1 dengan nilai 0
                do       #512,end1
                jsr      input                   ;sinyal disimpan di akumulator a dan b
                move     a,x0
                move     b,y0
                mpy      x0,y0,b
                tfr      b,a
                jsr      output                  ;sinyal yang akan dikeluarkan harus ditempatkan
                                                 ;terlebih dahulu pada akumulator a dan b
end1
                jmp      loop

                include  'akhir'                 ;program inisialisasi akhir

                end                              ;akhir program
```

(program untuk membangkitkan sinyal modulasi amplitudo AM SSB SC dimana kedua sinyal yang akam dimodulasi dibangkitkan oleh sinyal generator).

13.Ulangi langkah 4 dan 5.

14.Amati sinyal pada oscilloscope.

TUGAS ANALISA

1. Jelaskan teori modulasi.
2. Fungsi modulasi
3. Sebutkan kelebihan dan kekurangan masing-masing metode modulasi.

TUGAS TERAPAN.

1. Buat program untuk membangkitkan sinyal modulasi frekuensi dan modulasi pulsa.

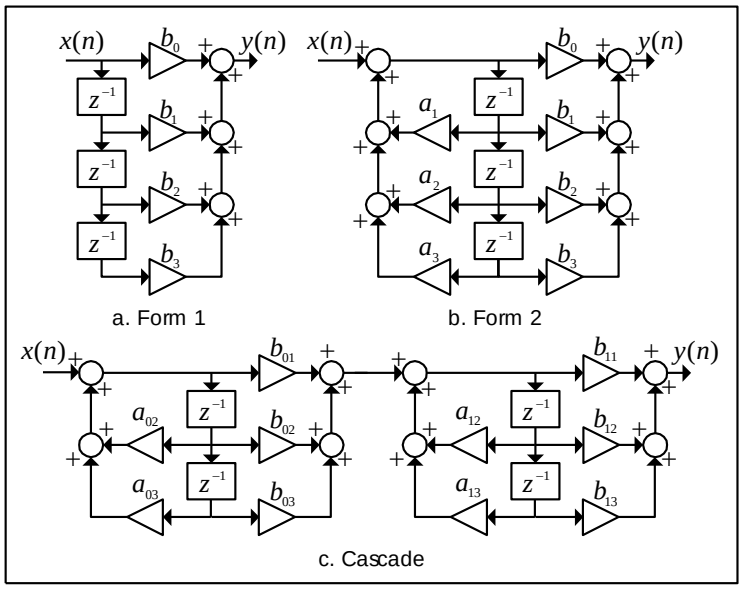
MODUL 3
FILTER DIGITAL

TUJUAN

1. Mempelajari desain filter digital menggunakan Motorola DSP56002EVM.
2. Mempelajari pengaruh filter digital pada sinyal informasi.

TEORI PENGANTAR

Perancangan filter digital dibuat dengan mentransformasikan transfer function filter analog ke transfer function filter digital. Ada dua macam model filter digital, yaitu FIR (finite impulse response) dan IIR (infinite impulse respons) dalam bentuk form 1, form 2 ataupun cascade.



Persamaan output untuk filter form 1 adalah sebagai berikut

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_n x(0)$$

dalam bentuk transformasi Z adalah

$$Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + \ldots + b_n z^{-n} X(0)$$

sehingga transfer function dalam transformasi Z adalah

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N} b_i z^{-i}$$

Dalam domain frekuensi dapat dinyatakan bahwa $x(n) = Xe^{j\omega n}$, sehingga persamaan output dalam domai frekuensi adalah

$$y(n) = \sum_{m=0}^{\infty} h_m Xe^{j\omega(n-m)}$$

$$= \left[\sum_{m=0}^{\infty} h_m e^{-j\omega m} \right] Xe^{j\omega n}$$

Transfer function dalam domai frekuensi adalah

$$H(j\omega) = \frac{Y(j\omega)}{X(j\omega)} = \sum_{m=0}^{\infty} h_m e^{-j\omega m}$$

Bila dinyatakan $\omega = n\pi\gamma$, maka persamaan transfer function dapat ditulis

$$H(\gamma) = \sum_{n=-\infty}^{\infty} h_n e^{-jn\pi\gamma}$$

, untuk $|n| < \infty$ dimana γ = normalisasi dari variabel frekuensi, $\gamma = {}^{f}\!/_{f_N}$, $f_N = {}^{f_s}\!/_{2}$ adalah frekuensi Nyquist dengan f_s adalah frekuensi sampling. Persamaan transfer function ini bisa ditulis sebagai

$$H(\gamma) = \sum_{n=-Q}^{Q} h_n e^{jn\pi\gamma}$$

untuk $|\gamma| < 1$ dan Q adalah finite positive. Karena $z = e^{j\pi\gamma}$ maka

$$H(\gamma) = \sum_{n=-Q}^{Q} h_n z^n$$

dan

$$H(z) = z^{-Q} \sum_{n=-Q}^{Q} h_n z^n = \sum_{n=-Q}^{Q} h_n z^{n-Q}$$

dengan mengubah variabel $i = -(n-Q)$ didapatkan

$$H(z) = \sum_{i=2Q}^{0} h_{Q-i} z^i = \sum_{i=0}^{2Q} h_{Q-i} z^{-i} = \sum_{i=0}^{2Q} b_i z^{-i}$$

, untuk $0 \le i \le 2Q$. Dengan demikian didapatkan persamaan $b_i = h_{Q-i}$. Koefisien h_{Q-i}

didapatkan dari persamaan

$$h_n = \tfrac{1}{2}\int_{-1}^{1} H(\gamma)e^{-jn\pi\gamma}d\gamma$$

untuk $|\gamma| < 1$ dan $n \geq 0$, didapatkan

$$h_n = \int_{0}^{1} H(\gamma)\cos(n\pi\gamma)d\gamma$$

, dimana $h_{-n} = h_n$

Persamaan output untuk filter form 2 adalah

$$y(n) = b_0 x(n) + b_1 x(n-1) + \ldots + b_n x(0) + a_1 y(n-1) + a_2 y(n-2) + \ldots + a_n y(0)$$

transfer function dalam transformasi Z adalah

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}$$

Sedangkan transfer function untuk filter analog low pass adalah

$$H(s) = \frac{1}{\left(s/\Omega_c\right)^2 + d\left(s/\Omega_c\right) + 1}$$

dengan transformasi bilinear, $s = \frac{2}{T}\frac{(z-1)}{(z+1)}$, didapatkan persamaan transfer function yang identik sehingga bisa didapatkan nilai-nilai koefisien dari filter digital. Dengan cara yang sama bisa didapatkan filter digital untuk high pass dan band pass.

Persamaan transfer function untuk filter cascade adalah

$$H(z) = \prod_{i=1}^{N} \frac{b_{i0} + b_{i1} z^{-1} + b_{i2} z^{-2}}{a_{i0} + a_{i1} z^{-1} + a_{i2} z^{-2}}$$

sedangkan transfer function untuk filter analog low pass cascade adalah

$$H(s) = \frac{1}{\left(s/\Omega_c\right)^2 + d\left(s/\Omega_c\right) + 1} \cdot \frac{1}{\left(s/\Omega_c\right)^2 + d\left(s/\Omega_c\right) + 1}$$

dengan transformasi bilinear seperti cara diatas bisa didapatkan persamaan transfer function yang identik pula sehingga koefisien filter digital cascade bisa didapatkan.

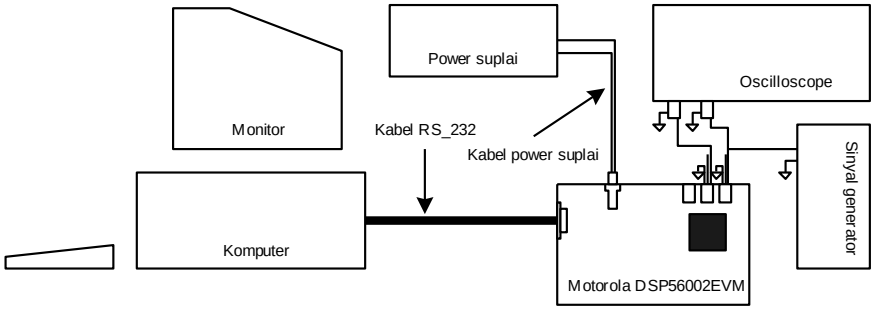
PERALATAN YANG DIGUNAKAN

1. Satu set komputer.
2. Motorola DSP56002EVM.

3. Power suplai AC atau DC
4. Signal generator.
5. Oscilloscope.

LANGKAH PERCOBAAN

1. Rangkai peralatan seperti pada gambar berikut



2. Nyalakan semua peralatan yang dipakai (komputer, power suplai, oscilloscope dan sinyal generator).
3. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
            org     x:$10           ;penempatan koefisien filter pada memori X
coefs_b     dc       0              ;b0
            dc      0.0468          ;b1
            dc      0.1009          ;b2
            dc      0.1514          ;b3
            dc      0.1872          ;b4
            dc      0.2                     ;b5
            dc      0.1872          ;b6
            dc      0.1514          ;b7
            dc      0.1009          ;b8
            dc      0.0468          ;b9
            dc      0               ;b10

            org     y:$10
nilai_x     dc      0,0,0,0,0,0     ;nilai awal x(n)
            dc      0,0,0,0,0,0     ;nilai awal x(n)

            include 'awal.asm'      ;program inisialisasi awal

loop
            jsr     input           ;sinyal disimpan di akumulator a dan b
            jsr     filter          ;proses filter
```

```
            jsr     output          ;sinyal yang akan dikeluarkan harus ditempatkan terlebih
                                    ;dahulu pada akumulator a dan b
            jmp     loop

filter
            move    #$19,r3
            move    #$1a,r4
            do      #10,endf1
            move    y:(r3)-,a
            move    a,y:(r4)-
endf1
            move    #$10,r3
            move    #$10,r4
            clr     a
            do      #11,endf2
            move    x:(r3)+,x0
            move    y:(r4)+,y0
            mac     x0,y0,b
endf2
            rts

            include 'akhir.asm'     ;program inisialisasi akhir

            end                     ;akhir program
```

(program filter digital form 1 dengan orde filter 11)

4. Simpan program dengan nama file ber-ektensi asm (*.asm).
5. Compile program untuk mendapatkan file ber-ektensi cld (*.cld), buka Debugger Motorola DSP56002EVM, load program dan jalankan Motorola DSP56002EVM.
6. Set frekuensi sinyal generator pada 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 6000, 7000, 8000, 8000, 9000, 10000 Hz.
7. Amati sinyal pada oscilloscope, catat amplitudo sinyal untuk masing-masing frekuensi.
8. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
            org     x:$10           ;penempatan koefisien filter pada memori X
coefs_b     dc      0.0660          ;b0
            dc      0.1979          ;b1
            dc      0.1979          ;b2
            dc      0.0660          ;b3

coefs_a     dc      -0.2200         ;a1
            dc      -0.1479         ;a2
            dc      -0.0348         ;a3

            org     y:$10
nilai_x     dc      0,0,0,0         ;nilai awal x(n)
```

```
nilai_y   dc      0,0,0             ;nilai awal y(n)

          include 'awal.asm'        ;program inisialisasi awal

loop
          jsr     input             ;sinyal disimpan di akumulator a dan b
          jsr     filter            ;proses filter
          jsr     output            ;sinyal yang akan dikeluarkan harus ditempatkan terlebih
                                    ;dahulu pada akumulator a dan b
          jmp     loop

filter
          move    #$12,r3
          move    #$13,r4
          do      #3,endf1
          move    y:(r3)-,a
          move    a,y:(r4)-
endf1
          move    b,y:$10
          move    #$10,r3
          move    #$10,r4
          clr     b
          do      #7,endf2
          move    x:(r3)+,x0
          move    y:(r4)+,y0
          mac     x0,y0,b
endf2
          move    #$15,r3
          move    #$16,r4
          do      #2,endf3
          move    y:(r3)-,a
          move    a,y:(r4)-
endf3
          move    b,y:$14
          rts

          include 'akhir.asm'       ;program inisialisasi akhir

          end                       ;akhir program
```

(program filter digital form 2 dengan orde filter 4)

9. Ulangi langkah 4 sampai 7.

10. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
          org     x:$10
coefs     dc       0.00370753     ;b10
          dc      0.5             ;scaling factor
          dc      0.00741518      ;b11
          dc      0.00370753      ;b12
          dc      0.83384359      ;a11
          dc      -0.34867418     ;a22
```

```
            dc      0.00485158      ;b20
            dc      0.5             ;scaling factor
            dc      0.00970316      ;b21
            dc      0.86615109      ;a21
            dc      0.00485158      ;b22
            dc      -0.38555753     ;a22

rtdelay     bsc     4,$0            ;used to store information for filter
ltdelay     bsc     4,$0
tempstore   ds      1
LINEAR      EQU     $FFFF

            include 'awal.asm'      ;program inisialisasi awal

loop
            jsr     input
            jsr     process_stereo
            jsr     output
            jmp     loop

process_stereo
            move    #ltdelay,r0     ;set up pointer
            jsr     filter          ;filter the left sample
            move    a,x:tempstore           ;save filtered left sample
            move    b,a             ;move right sample
            move    #rtdelay,r0             ;set up pointer
            jsr     filter          ;filter the right sample
            move    a,b             ;filtered right into b
            move    x:tempstore,a   ;filtered left into a
            rts

filter
            move    #LINEAR,m0      ;linear addressing
            move    m0,m4
            ori     #$08,mr         ;set scaling mode
            do      #2,endloop      ;do for both stages
            asr     a       #coefs,r4       ;r4=pointer to coefficients
            move    a,x0    y:(r0)+,a
            asr     a       x:(r4)+,x1
            macr    x1,x0,a x:(r4)+,x1      y:(r0)-,y0
            mpy     y0,x1,a a,y1            x:(r4)+,x1
            mac     x0,x1,a x:(r4)+,x1
            macr    y1,x1,a x:(r4)+,x1
            mpy     x1,x0,a x:(r4)+,x1      a,y:(r0)+
            macr    y1,x1,a
            tfr     y1,a    x:(r4)+,x0      a,y:(r0)+
endloop
            andi    #$f3,mr
            rts

            include 'akhir.asm'                     ;program inisialisasi akhir

            end                                     ;akhir program
```

(program filter digital cascade)

11.Ulangi langkah 4 sampai 7.

TUGAS ANALISA

1. Jelaskan teori filter digital.
2. Fungsi filter digital
3. Hitung frekuensi cut-off masing-masing percobaan.

TUGAS TERAPAN

1. Rancang filter digital untuk frekuensi cut-off 500 Hz dan 5KHz dengan pendekatan metode FIR dan IIR dalam bentuk form 1, form 2 dan cascade.

MODUL 4
FAST FOURIER TRANSFORM

TUJUAN

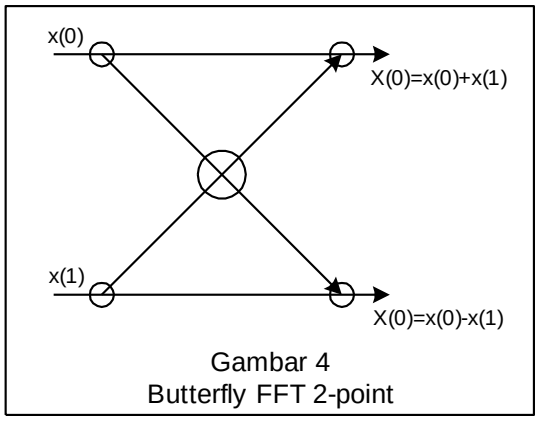
1. Mempelajari aplikasi FFT menggunakan Motorola DSP56002EVM.
2. Mempelajari desain sistem untuk mentransformasikan dari domain waktu ke domain frekuensi.

TEORI PENGANTAR

Fast fourier transform atau transformasi fourier cepat merupakan satu algoritma transformasi yang relatif lebih cepat untuk menghitung Discrete Fourier Transform (DFT) atau Transformasi Fourier Diskrit.

$$x(k) = \sum_{n=0}^{N-1} x(n) \cdot W_n^{kn}$$

dimana $W_n^{kn} = e^{j\frac{2\pi kn}{N}}$



Gambar 4
Butterfly FFT 2-point

Dengan menggunakan sifat periodisitas dari $W_n^{kn} = e^{j\frac{2\pi kn}{N}}$, maka proses komputasi dapat dimanipulasi sehingga lebih cepat.

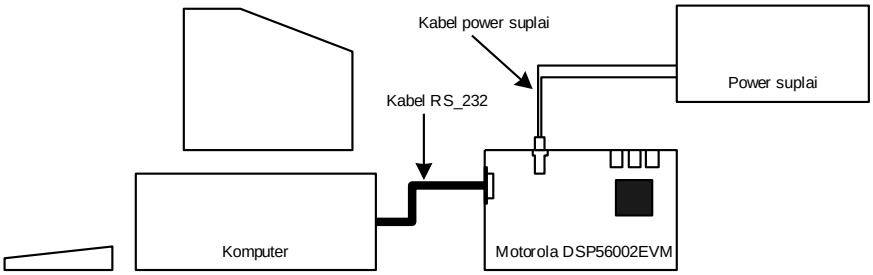
PERALATAN YANG DIGUNAKAN

1. Satu set komputer.
2. Motorola DSP56002EVM.

3. Power suplai AC atau DC
4. Signal generator.
5. Oscilloscope.

LANGKAH PERCOBAAN

1. Rangkai peralatan seperti pada gambar berikut



2. Nyalakan semua peralatan yang dipakai (komputer, power suplai).
3. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
RFFT56T ident   1,0
        page    132,60
        opt     nomd,nomex,loc,nocex,mu

sincosr macro   points,coef
sincosr ident   1,2

pi      equ     3.141592654
freq    equ     2.0*pi/@cvf(points*2)

        org     x:coef
count   set     0
        dup     points
        dc      0.999*@cos(@cvf(count)*freq)
count   set     count+1
        endm

        org     y:coef
count   set     0
        dup     points
        dc      0.999*@sin(@cvf(count)*freq)
count   set     count+1
        endm
        endm    ;end of sincosr  macro

gen56r  macro   POINTS,IDATA
srate   set     44100   ;Hz
ffreq   set     2000    ;Hz
```

```
ppi     equ     3.141592654
freq2   equ     (2.0*ppi*ffreq/@cvf(srate))

        org     x:0
count   set     0               ; Even index in X memory
        dup     POINTS/2
        dc      @sin(@cvf(count)*freq2)/POINTS
count   set     count+2
        endm

        org     y:0
count   set     1               ; Odd index in Y memory
        dup     POINTS/2
        dc      @sin(@cvf(count)*freq2)/POINTS
count   set     count+2
        endm
        endm    ;end of gen56r  macro

CFFT256         macro   IDATA,COEF,POINTS,ODATA
CFFT256         ident   1,0

OFFSET  equ     64              ;offset for 256 complex on-chip ROM table
        move    y:IDATA,r0      ;r0 -> Ar
        move    r0,n3
        move    #-1,m2          ;m2 always linear mode
        move    y:ODATA,r3      ;r3 always has ODATA
        move    #OFFSET,n6      ;n6 offset for addressing on-chip sin/cos table
        move    #0,m6           ;bit reversed address on r6
        move    #POINTS/4,n0    ;offset and butterflies per group
        move    #POINTS/2-1,m0          ;modulo addressing

        do      #3,_end_trivial ;do three Radix 4 passes
        move    n0,n1           ;pointer offset
        move    n0,n4           ;pointer offset
        move    n0,n5           ;
        lea     (r0)+n0,r4      ;r4 -> Bi
        move    m0,m5
        lea     (r4)+n4,r1      ;r1 -> Cr
        move    m0,m1           ;
        move    m0,m4
        lea     (r1)+n1,r5      ;r5 -> Di

        move    x:(r0)+n0,a
        move    x:(r1)+n1,b

        do      n0,_twopass
        add     a,b     x:(r0)+n0,x1    y:(r5)+n5,y1
        subl    b,a     b,x:(r0)        y:(r4),b
        add     y1,b    a,x0            y:(r4)+n4,a
        sub     y1,a    b,x:(r3)        x0,b
        sub     a,b     x:(r1),x0
        addl    b,a     b,x:(r1)+n1     x0,b
        sub     x1,b    a,x:(r1)+       x0,a
        add     x1,a    x:(r0)+n0,b     b,y1
```

```
            sub     a,b     y:(r5),y0
            addl    b,a     b,x:(r0)+n0     y:(r4),b
            sub     y0,b    a,x:(r0)+       y:(r4),a
            add     y0,a    x:(r3),b        b,y0
            add     a,b
            subl    b,a     y0,b            b,y:(r4)+n4
            add     y1,b    y0,a            a,y:(r4)+
            sub     y1,a    x:(r1)+n1,b     b,y:(r5)+n5
            move    x:(r0)+n0,a             a,y:(r5)+
_twopass
            move    n5,a
            asr     a       n5,r1
            move    a,n1
            move    r1,r5
            lea     (r1)+n1,r4
            move    #2,n4
            move    r4,r0
            move    x:(r1),a        y:(r4)+,b
            do      n1,_no_more
            add     a,b     x:(r0),x0       y:(r4)-,y0
            subl    b,a     b,x:(r1)+       y:(r5),b
            add     x0,b    a,x:(r0)+       y:(r5),a
            subl    b,a     y0,b            b,y:(r4)+n4
            move    x:(r1),a        a,y:(r5)+
_no_more
            move    n0,a
            asr     a       n3,r0
            asr     a       a,r2
            move    a,n0
            move    x:(r2)-,b
            move    r2,m0
_end_trivial
            move    r0,r4
            move    n1,r1
            move    r1,r5
            move    x:(r0),a
            move    x:(r1),b
            do      n1,_extra
            add     a,b     y:(r5),y0
            subl    b,a     b,x:(r0)+       y:(r4),b
            add     y0,b    a,x:(r1)+       y:(r4),a
            sub     y0,a    x:(r1),b        b,y:(r4)+
            move    x:(r0),a        a,y:(r5)+
_extra
            move    m2,m0
            move    m2,m1
            move    m2,m4
            move    m2,m5
            move    #$80,r0
            move    #4,m3
            move    #POINTS/8,n0
            move    n0,n1
            move    n0,n4
            move    n0,n5
```

```
            move    #COEF+OFFSET,r6
            move    n0,n2
            lea     (r0)+n0,r1
            move    r0,r4
            lea     (r1)-,r5
            move    #1,n3
            move    n3,r2
            move    #2,r2
_set_grp
            do      m3,_inner_loop
            jsr     <_inner_pass
            move    n0,a
            asr     a       y:IDATA,r0
            move    a,n0
            move    r2,a
            asl     a       n0,n1
            move    a,r2
            asr     b       r2,n3
            lea     (r2)-,n3
            move    n4,a
            asl     a       #COEF+OFFSET,r6
            move    a,r0
_inner_set
            move    n0,n4
            move    n0,n5
            lea     (r0)+n0,r1
            move    r0,r4
            lea     (r1)-,r5
_inner_loop
            move    #8,r0
            move    #31,n2
            move    #10,r1
            move    r0,r4
_no_set
            move    #7,r5
            move    #3,n0
            move    n0,n1
            move    n0,n4
            move    n0,n5
            jsr     <_next_last
            move    y:IDATA,r0
            move    r3,r4
_add_offset
            lea     (r0)+,r1
            lea     (r4)-,r5
            move    #64,n2
            move    #2,n0
            move    n0,n1
            move    n0,n4
            move    n0,n5
            move    #COEF,r6
            jsr     <_last
            jmp     <_end_FFT
_inner_pass
```

```
            do      n3,_end_grp
            move    x:(r5),a
            move    x:(r6),x0           y:(r0),b
            move    x:(r1),x1 y:(r6)+n6,y0
            do      n0,_end_bfy1
            mac     -x1,y0,b y:(r1)+,y1
            macr    x0,y1,b             a,x:(r5)+           y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     x1,x0,b             x:(r0)+,a           a,y:(r5)
            macr    y1,y0,b             x:(r1),x1
            subl    b,a                 b,x:(r4)+           y:(r0),b
_end_bfy1
            move    a,x:(r5)+n5         y:(r1)+n1,b
            move    x:(r4)+n4,a         y:(r0)+n0,b
            move    x:(r1),x1
            move    x:(r5),a            y:(r0),b
            do      n0,_end_bfy2
            mac     -x1,x0,b             y:(r1)+,y1
            macr    -y0,y1,b            a,x:(r5)+           y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     -x1,y0,b            x:(r0)+,a           a,y:(r5)
            macr    y1,x0,b             x:(r1),x1
            subl    b,a                 b,x:(r4)+           y:(r0),b
_end_bfy2
            move    a,x:(r5)+n5         y:(r1)+n1,b
            move    x:(r4)+n4,a         y:(r0)+n0,b
_end_grp
            rts
_next_last
            move    x:(r5),a                    y:(r0),b
            move    x:(r1),x1                   y:(r6),y0
            do      n2,_n_last
            mac     -x1,y0,b            x:(r6)+n6,x0        y:(r1)+,y1
            macr    x0,y1,b             a,x:(r5)+n5         y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     x1,x0,b             x:(r0)+,a           a,y:(r5)
            macr    y1,y0,b             x:(r1),x1
            subl    b,a                 b,x:(r4)+           y:(r0),b
            mac     -x1,y0,b                    y:(r1)+n1,y1
            macr    x0,y1,b             a,x:(r5)+           y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     x1,x0,b             x:(r0)+n0,a         a,y:(r5)
            macr    y1,y0,b             x:(r1),x1
            subl    b,a                 b,x:(r4)+n4         y:(r0),b
            mac     -x1,x0,b                    y:(r1)+,y1
            macr    -y0,y1,b            a,x:(r5)+n5         y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     -x1,y0,b            x:(r0)+,a           a,y:(r5)
            macr    y1,x0,b             x:(r1),x1
            subl    b,a                 b,x:(r4)+           y:(r0),b
            mac     -x1,x0,b                    y:(r1)+n1,y1
            macr    -y0,y1,b a,x:(r5)+           y:(r0),a
            subl    b,a                 x:(r0),b            b,y:(r4)
            mac     -x1,y0,b            x:(r0)+n0,a         a,y:(r5)
```

```
            macr    y1,x0,b         x:(r1),x1       y:(r6),y0
            subl    b,a             b,x:(r4)+n4     y:(r0),b
_n_last
            move    a,x:(r5)
            rts
_last
            move    x:(r5),a        y:(r0),b
            move     x:(r1),x1      y:(r6),y0
            do      n2,_end_last
            mac     -x1,y0,b        x:(r6)+n6,x0    y:(r1)+n1,y1
            macr    x0,y1,b         a,x:(r5)+n5     y:(r0),a
            subl    b,a             x:(r0),b        b,y:(r4)
            mac     x1,x0,b         x:(r0)+n0,a     a,y:(r5)
            macr    y1,y0,b         x:(r1),x1
            subl    b,a             b,x:(r4)+n4     y:(r0),b
            mac     -x1,x0,b                y:(r1)+n1,y1
            macr    -y0,y1,b        a,x:(r5)+n5     y:(r0),a
            subl    b,a             x:(r0),b        b,y:(r4)
            mac     -x1,y0,b        x:(r0)+n0,a     a,y:(r5)
            macr    y1,x0,b         x:(r1),x1       y:(r6),y0
            subl    b,a             b,x:(r4)+n4     y:(r0),b
_end_last
            move    a,x:(r5)
            rts
_end_FFT
            endm

SPLIT56     macro           IDATA,COEF,POINTS,ODATA
SPLIT56     ident   1,0

OFFS        equ     1
            move    #POINTS-1,n0
            move    #POINTS/2-1,n3
            move    y:ODATA,r0
            move    #COEF+1,r2
            move    r2,r6
            move    #-1,m6
            move    m6,m2
            move    #<OFFS,n2
            move    n2,n6
            lea     (r0)+n0,r5
            move    y:IDATA,r3
            move    n0,r1
            move    #POINTS/2,n0
            move    n0,n5
            move    m5,m3
            move    m5,m1
            move    #0,m0
            move    m0,m5
            move    x:(r0),b
            move     x:(r5),x1      y:(r0),a
            sub     a,b             x:(r0),x0
            asl     b               r3,r4
            add     x0,a            b,y:(r0)+n0
```

```
            asl     a               x:(r1)+,x0      y:(r5),b
            move    a,x:(r4)        y:(r1),a
            move    y:(r0),y0
            move    x:(r3)+,x0
            do      n3,_end_split
            add     y0,b            y0,a            a,y:(r1)-
            subl    b,a             x:(r0),b        b,y1
            sub     x1,b            x:(r0)+n0,a     a,y:(r4)
            subl    b,a             x:(r2)+n2,x1    y:(r6)+n6,y0
            mac     x1,y1,a         b,x0            a,y:(r5)
            macr    -y0,x0,a        y:(r5)-n5,b
            subl    a,b             a,x:(r3)        y:(r4),a
            mac     -x1,x0,a        b,x:(r1)        y:(r5),b
            macr    -y0,y1,a        y:(r4),y0
            sub     y0,a            x:(r5),x1       a,y:(r3)+
            sub     y0,a            y:(r0),y0
_end_split
            move    y0,a            a,y:(r1)
            neg     a               y:ODATA,r5
            move    y:IDATA,r0
            move    a,y:(r4)
            move    y:(r5),a
            move    a,y:(r0)
            endm ;split56

reset       equ     0
start       equ     $40
POINTS      equ     512
COEF        equ     $900

            sincosr POINTS/2,COEF
            gen56r  POINTS,IDATA

            org     Y:$0400
IDATA       dc      $0000
ODATA       dc      $0200

            opt     mex
            org     p:reset
            jmp     start

            org     p:start
            CFFT256         IDATA,COEF,POINTS/2,ODATA
            SPLIT56         IDATA,COEF,POINTS/2,ODATA

            rts
            end
```

4. Simpan program dengan nama file ber-ektensi asm (*.asm).
5. Compile program untuk mendapatkan file ber-ektensi cld (*.cld), buka Debugger

Motorola DSP56002EVM, load program dan jalankan Motorola DSP56002EVM.

6. Catat isi memory X dan Y dari 0 sampai 512

TUGAS ANALISA

1. Jelaskan teori Fast Fourier Transform
2. Bandingkan hasil program dengan konsep teorinya.

TUGAS TERAPAN

1. Buat program untuk menghitung DFT dari deretan data x(n), x(n) = (n) + (n-1) + (n-2), dengan size DFT(N) = 4.
2. Aplikasikan pada Motorola DSP56002EVM dan bandingkan hasilnya dengan perhitungan konsep teoritisnya.

MODUL 5
PENGOLAHAN SINYAL AUDIO

TUJUAN

1. Mempelajari aplikasi pengolahan sinyal audio menggunakan Motorola DSP56002EVM.

TEORI PENGANTAR

Motorola DSP56002EVM bisa diprogram untuk mengolah sinyal audio, seperti pembangkitan echo (repeter) untuk memperindah alunan suara, memperbaiki sinyal audio karena adanya noise, membuat suara surround stereo atau membuat mixer.

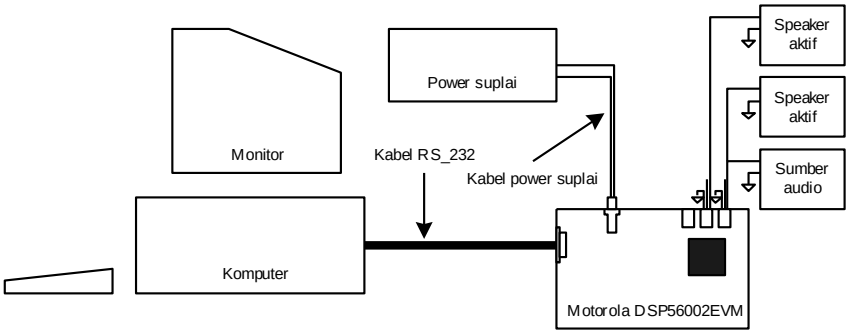
Misalkan suatu pesawat penerima radio akan menerima sinyal radio dan akan menghasilkan sinyal audio. Karena melalui proses transmisi sinyal melalui gelombang radio, maka sinyal radio yang diterima telah bercampur dengan noise sehingga sinyal audio yang dihasilkan tidak bagus. untuk menghilangkan noise ini bisa dilakukan dengan memfilter sinyal audio secara digital menggunakan Motorola DSP56002EVM. Desain filter digital seperti ditunjukkan pada modul 3.

PERALATAN YANG DIGUNAKAN

1. Satu set komputer.
2. Motorola DSP56002EVM.
3. Power suplai AC atau DC
4. Sumber audio
5. Oscilloscope.
6. Speaker aktif.

LANGKAH PERCOBAAN

1. Rangkai peralatan seperti pada gambar berikut



2. Nyalakan semua peralatan yang dipakai (komputer, power suplai, sumber audio dan speaker aktif).
3. Tulis program pada NOTEPAD atau word processor lainnya, sebagai berikut:

```
            org    x:$10
coefs       dc     0.00370753          ;b10
            dc     0.5                         ;scaling factor
            dc     0.00741518          ;b11
            dc     0.00370753          ;b12
            dc     0.83384359          ;a11
            dc     -0.34867418         ;a22

            dc     0.00485158          ;b20
            dc     0.5                         ;scaling factor
            dc     0.00970316          ;b21
            dc     0.86615109          ;a21
            dc     0.00485158          ;b22
            dc     -0.38555753         ;a22

rtdelay     bsc    4,$0                        ;used to store information for filter
ltdelay     bsc    4,$0
tempstore   ds     1
LINEAR      EQU    $FFFF

            include  'awal.asm'

loop
            jsr        input
            jsr        process_stereo
            jsr        output
            jmp        loop

process_stereo
            move       #ltdelay,r0          ;set up pointer
            jsr        filter               ;filter the left sample
```

```
        move    a,x:tempstore     ;save filtered left sample
        move    b,a               ;move right sample
        move    #rtdelay,r0         ;set up pointer
        jsr     filter            ;filter the right sample
        move    a,b               ;filtered right into b
        move    x:tempstore,a     ;filtered left into a
        rts
filter
        move    #LINEAR,m0        ;linear addressing
        move    m0,m4
        ori     #$08,mr             ;set scaling mode
        do      #2,endloop        ;do for both stages
        asr     a       #coefs,r4       ;r4=pointer to coefficients
        move    a,x0    y:(r0)+,a
        asr     a       x:(r4)+,x1
        macr    x1,x0,a x:(r4)+,x1      y:(r0)-,y0
        mpy     y0,x1,a a,y1            x:(r4)+,x1
        mac     x0,x1,a x:(r4)+,x1
        macr    y1,x1,a x:(r4)+,x1
        mpy     x1,x0,a x:(r4)+,x1      a,y:(r0)+
        macr    y1,x1,a
        tfr     y1,a    x:(r4)+,x0      a,y:(r0)+
endloop
        andi    #$f3,mr
        rts

        include 'akhir.asm'

        end
```

(program filter digital)

4. Simpan program dengan nama file ber-ektensi asm (*.asm).
5. Compile program untuk mendapatkan file ber-ektensi cld (*.cld), buka Debugger Motorola DSP56002EVM, load program dan jalankan Motorola DSP56002EVM.
6. Bandingkan input dan outputnya.

TUGAS ANALISA

1. Jelaskan pengaruh filter pada sinyal audio.
2. Bandingkan sinyal audio input dan output.

TUGAS TERAPAN

1. Buat program untuk aplikasi repeater, surround stereo dan mixer.