DESAIN DAN IMPLEMENTASI SISTEM PENGENAL SUARA MANUSIA (VOICE RECOGNITION) BERBASIS MIKROPROSESOR TMS32010 YANG DIINTERFACEKAN KE IBM PC

OLEH :

MUHAMMAD YOSEP

2882200938

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1994

DESAIN DAN IMPLEMENTASI SISTEM PENGENAL SUARA MANUSIA (VOICE RECOGNITION) BERBASIS MIKROPROSESOR TMS32010 YANG DIINTERFACEKAN KE IBM PC

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir Handayani Tj.	(Ir. Hendra Kusuma)

SURABAYA
Agustus, 1994

ABSTRAK

Perkembangan teknologi elektronika yang pesat telah merambah ke berbagai bidang. Hampir tidak ada bidang kehidupan manusia modern yang belum tersentuh teknologi elektronika. Mulai dari kehidupan rumah tangga sampai riset lanjutan di bidang penerbangan angkasa luar.

Ada kecenderungan manusia modern untuk semakin meningkatkan kualitas kehidupannya dengan bantuan alat-alat yang canggih. Seperti misalnya penggunaan *remote control*, *automatic answering machine*, *automatic teller machine (ATM)*, dst. Ini semua bertujuan selain untuk meningkatkan kualitas kehidupan manusia, juga untuk memenuhi kecenderungan manusia sekarang yang semakin ingin serba praktis dan serba *instant* dalam kehidupan kesehariannya.

Tugas akhir ini ditujukan sebagai eksperimen awal dari suatu sistem/perangkat elektronik untuk memenuhi salah satu kebutuhan tersebut di atas. Dalam tugas akhir ini dicoba dibuat suatu peralatan elektronik yang dapat mengenal suara manusia secara terbatas. Terbatas disini maksudnya adalah bahwa alat tersebut hanya mampu mengenal suara orang yang suaranya telah "dilatihkan" terlebih dulu kepadanya, dan hanya mampu mengenal kata-kata yang jumlahnya sangat terbatas. Metoda yang digunakan dalam tugas akhir ini adalah Metoda Band-Pass Filter Bank. Untuk meningkatkan kecepatan respon sistem, digunakan prosesor khusus pengolah sinyal digital, TMS32010. Prosesor ini bertugas mendapatkan parameter spektrum daya dari sinyal suara. Sedang komputer (PC) bertugas sebagai alat bantu untuk *loading* program ke memory TMS32010, dan berperan dalam pengambilan keputusan.

KATA PENGANTAR

Puji syukur kami persembahkan ke hadapan Allah swt., karena dapat terselesaikannya tugas akhir ini hanya karena atas Kemurahan dan Kasih sayang-Nya semata.

Tugas akhir ini dibuat untuk memenuhi salah satu syarat dalam penyelesaian studi di Bidang Studi Elektronika Jurusan Teknik Elektro Fakultas Teknologi Industri ITS Surabaya.

Banyak pihak yang telah berperan baik secara langsung maupun tidak langsung dalam penyelesaian tugas akhir ini. Karena itu penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua penulis, yang telah membesarkan, menjadi pendidik utama bagi penulis selama ini, dan telah turut memberikan dorongan dan semangat dalam penyelesaian tugas akhir ini.
2. Ibu Dr. Ir. Handayani Tj., selaku dosen pembimbing, yang dengan penuh kesabaran dan kebijaksanaan telah memberikan pengarahan-pengarahan dalam pengerjaan tugas akhir ini.
3. Bapak Ir. Hendra Kusuma, selaku dosen wali dan sekaligus dosen pembimbing.
4. Bapak Ir. Soetikno, selaku Koordinator Bidang Studi Elektronika Jurusan Teknik Elektro FTI-ITS.

5. Bapak Dr. Ir. Moch. Salehudin, selaku Ketua Jurusan Teknik Elektro Fakultas Teknologi Industri ITS.
6. Rekan-rekan mahasiswa Bidang Studi Elektronika Teknik Elektro ITS, khususnya angkatan E-28 yang telah banyak membantu dalam penyelesaian tugas akhir ini.

Penulis berharap semoga karya sederhana ini dapat sedikit turut menyumbangkan sesuatu yang berguna untuk pengembangan teknologi elektro, khususnya teknologi elektronika di kampus tercinta ini.

Surabaya, Agustus 1994

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

I.1 LATAR BELAKANG

Bahasa adalah alat komunikasi yang paling alami dan paling banyak digunakan. Alat komunikasi yang lain seperti gambar-gambar atau gerakan-gerakan isyarat juga banyak digunakan bila bahasa tidak memungkinkan untuk difungsikan.

Secara garis besar bahasa dapat dibagi dua, yaitu bahasa lisan dan bahasa tulisan. Bahasa tulisan berlangsung lebih lama dan lebih terstruktur daripada bahasa lisan. Sehingga model komunikasi ini cocok untuk digunakan dalam penyampaian ilmu pengetahuan. Tetapi informasi yang dapat disampaikan oleh bahasa lisan jauh lebih banyak daripada bahasa tulisan. Karena dalam bahasa lisan, intonasi dan penekanan kata maupun emosi pembicara dapat bercerita lebih banyak dari sekedar arti kata yang disampaikan.

Dalam tugas akhir ini, diinginkan untuk dibuat suatu peralatan yang mampu mengenal suara manusia. Sehingga bahasa dapat digunakan sebagai alat komunikasi antara manusia dengan mesin. Proses pengenalan suara dari peralatan ini dapat dibagi dalam tiga bagian penting. Pertama, ekstraksi parameter sinyal suara. Kedua, pembandingan antara input parameter sinyal suara dengan data reference. Dan ketiga adalah proses pengambilan keputusan. Ekstraksi parameter sinyal suara dilakukan dengan metoda Band-Pass Filter Bank. Sedang proses

pembandingan dan pengambilan keputusan menggunakan metoda pemrograman dinamik.

I.2 TUJUAN

Tujuan dari dibuatnya tugas akhir ini adalah untuk menciptakan suatu sistem yang mampu berkomunikasi dengan manusia melalui media bahasa. Karena proses pengenalan suara manusia membutuhkan jumlah komputasi yang cukup banyak, maka digunakan prosesor khusus pengolah sinyal digital. Dalam hal ini digunakan prosesor TMS32010.

Dengan digunakannya prosesor TMS32010 ini diharapkan proses komputasi berlangsung lebih cepat, sehingga dapat mengurangi waktu proses pengenalan secara keseluruhan.

I.3 BATASAN MASALAH

Ekstraksi parameter suara dengan metoda Band Pass Filter Bank akan menghasilkan spektrum amplopnya. Informasi yang bisa kita dapatkan dari sini adalah sedikit, sehingga ini membatasi kemampuan dari sistem pengenal suara ini. Berikut ini adalah batasan-batasan yang digunakan dalam realisasi dari peralatan ini:

- Suara yang dikenali adalah berupa kata.
- Kata yang digunakan adalah kata dasar.
- Ucapan diambil dengan mikropon yang diletakkan di dekat mulut.
- Lingkungan tempat pengambilan suara adalah lingkungan laboratorium

yang tenang, sehingga diasumsikan tidak ada gangguan yang berarti dari noise lingkungan.

- Jumlah kata yang dapat dikenali adalah sangat terbatas, kurang dari 20 kata.

I.4 METODOLOGI

Pembahasan dimulai dengan teori dasar analisa sinyal. Dilanjutkan dengan teori dasar sinyal suara, *Analog to Digital Conversion*, teori pelengkungan waktu dinamik dan teori sistem pengenalan suara. Kemudian dijelaskan tentang prosesor TMS32010, perangkat keras, dan algoritma perangkat lunak. Pembahasan diakhiri dengan kesimpulan dan saran tentang sistem yang telah dibuat.

I.5 LANGKAH-LANGKAH PEMBAHASAN

Langkah-langkah pembahasan dalam tugas akhir ini adalah sebagai berikut:

BAB I: Berisi tentang pendahuluan.

BAB II: Berisi tentang teori dasar analisa sinyal, teori dasar sinyal suara, pengkonversian sinyal suara, teori dasar pengolahan sinyal suara, sistem pengenal suara dan penjelasan tentang mikroprosesor TMS 32010.

BAB III: Berisi tentang desain perangkat keras, meliputi desain rangkaian analog, desain rangkaian *Analog to Digital Conversion*, dan desain rangkaian prosesor TMS 32010 beserta rangkaian-rangkaian pendukungnya.

BAB IV: Berisi tentang desain perangkat lunak, meliputi desain perangkat lunak untuk ekstraksi parameter sinyal suara, dan desain perangkat lunak untuk proses pengenalan suara.

BAB V: Berisi tentang hasil pengujian alat.

BAB VI: Berisi tentang kesimpulan dan saran tentang alat yang dibuat.

BAB II

PENGOLAHAN DIGITAL SINYAL SUARA

II.1 TEORI DASAR PENGOLAHAN SINYAL DIGITAL

II.1.1 Pengertian Barisan

Teori sistem linier waktu diskrit berkenaan dengan representasi dan pemrosesan barisan *(sequence)*, baik dalam waktu maupun frekuensi. Sinyal-sinyal waktu diskrit didefinisikan hanya untuk nilai diskrit dari waktu. Umumnya dikuantisasi seragam yaitu $t = nT$, di mana T adalah interval antara sampel waktu. Secara matematis sinyal waktu diskrit dinotasikan sebagai $\{h(nT)\}$ atau $h(nT)$ dimana $N1 \leq n \leq N2$.

Beberapa barisan yang sering digunakan dalam pengolahan sinyal digital diperlihatkan pada Gambar 2-1. Gambar 2-1 (a) memperlihatkan impuls digital $\delta(n)$ yang didefinisikan dengan relasi [8]:

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \qquad (2\text{-}1)$$

Gambar 2-1 (b) menunjukkan barisan impuls yang mengalami delay sebesar n_0 sampel dan didefinisikan sebagai [8]:

$$\delta(n) = \begin{cases} 1 & n = n_0 \\ 0 & n \neq n_0 \end{cases} \qquad (2\text{-}2)$$

Gambar 2-1 (c) untuk unit step dengan relasi [8]:

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases} \qquad (2\text{-}3)$$

Gambar 2-1 (d) memperlihatkan eksponensial menurun $g(n) = a^n$ untuk $n \geq 0$ dan sinusoida $h(n) = \cos(2\pi n/n_0)$ untuk semua n.

Barisan sembarang dapat dinyatakan secara mudah dengan barisan impuls digital. Tinjau barisan yang dibangkitkan dari bilangan-bilangan ..., a(0), a(1), a(2), ..., dimana a(n) menyatakan amplitudo anggota ke n. Cara mudah menyatakan ini adalah [8]:

$$\{a(n)\} = \sum_{m=-\infty}^{\infty} a(m)\,\delta(n-m) \qquad (2\text{-}4)$$

II.1.2 Sistem-sistem Linier Tak Berubah Waktu

Sistem waktu diskrit secara fungsional menghubungkan input x(n) dan output y(n) dengan relasi [8]:

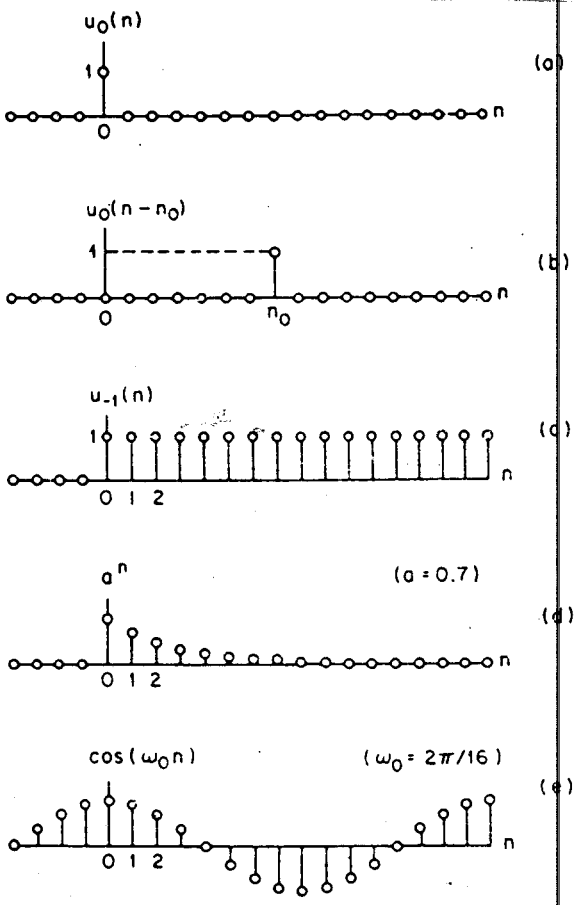
$$y(n) = \Phi[x(n)] \qquad (2\text{-}5)$$

dimana $\Phi(.)$ ditentukan oleh sistem.

Sistem linier didefinisikan sebagai berikut. Jika x1(n) dan x2(n) adalah input spesifik bagi suatu sistem linier dan y1(n)

dan y2(n) adalah outputnya, maka barisan ay1(n) + by2(n) akan diperoleh pada output.

Dalam suatu sistem tak berubah waktu *(time invariant)*, jika barisan input x(n) menghasilkan barisan output y(n), maka barisan input $x(n-n_0)$ akan



Gambar 2-1[1] Barisan-barisan yang penting dalam Pengolahan Sinyal Digital

(a) Impuls Digital
(b) Barisan Impuls terdelay n0 Sampel
(c) Unit Step Digital
(d) Barisan Eksponensial Menurun $g(n)=a^n$
(e) Barisan Sinusoidal $h(n)=\cos(2\pi n/n0)$

menghasilkan barisan output $y(n-n_0)$ untuk semua n_0. Dapat diperlihatkan bahwa untuk sistem linier tak berubah waktu, terdapat hubungan konvolusi antara barisan input dan output. Misalkan h(n) merupakan respon sistem terhadap impuls

[1]Rabiner, Lawrence R., Bernard Gold, Theory and Application of Digital Signal Processing, Prentice Hall of India, hal. 11.

digital (barisan h(n) dinamakan *respon impuls* atau *respon unit sample*). Dari persamaan (2-4) x(n) dapat dituliskan sebagai [8]:

$$x(n) = \sum_{m=-\infty}^{\infty} x(m)\,\delta(n-m) \qquad (2\text{-}6)$$

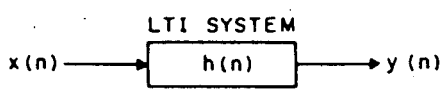
Karena h(n) merupakan respon terhadap barisan δ(n), maka dapat juga dikatakan bahwa h(n-m) adalah respon terhadap barisan δ(n-m). Secara sama, menurut linieritas, respon terhadap barisan x(m)δ(n-m) haruslah x(m)h(n-m). Dengan demikian respon terhadap x(n) haruslah [8]:

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)\,h(n-m) \qquad (2\text{-}7a)$$

yang merupakan relasi konvolusi yang diinginkan. Secara ekivalen, persamaan (2-7a) dapat dikonversikan ke dalam bentuk [8]:

$$y(n) = \sum_{m=-\infty}^{\infty} h(m)\,x(n-m) \qquad (2\text{-}7b)$$

Dengan demikian untuk sistem-sistem linier tak berubah waktu (LTI), barisan h(n) mewakili karakteristik sistem sebagaimana diperlihatkan dalam Gambar 2-2 berikut:



Gambar 2-2[2] Representasi Sistem LTI

[2]Ibid, hal. 14.

II.1.3 Persamaan beda

Sistem-sistem waktu diskrit linier mentransformasikan barisan masukan x(n) ke dalam barisan keluaran y(n) [8]:

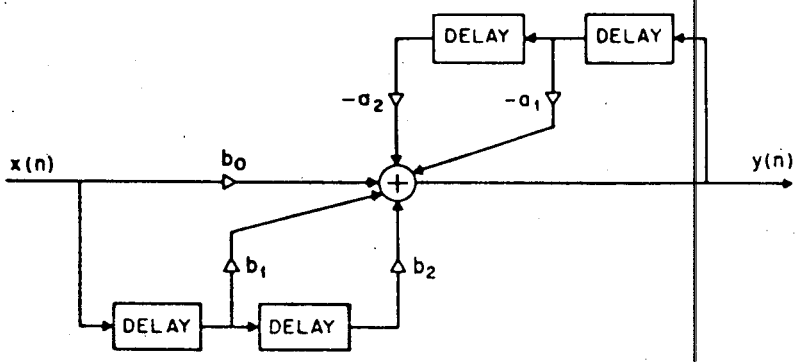
$$y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{i=0}^{M} a_i y(n-i) \qquad n \geq 0 \qquad (2\text{-}8)$$

dimana $\{b_i\}$ dan $\{a_i\}$ menyatakan karakteristik sistem. Dengan adanya kondisi awal [misal, x(i), y(i), i= -1, -2, ..., -M] dan barisan input x(n), maka y(n) dapat dihitung langsung dari persamaan (2-8).

Dalam pengolahan sinyal digital, persamaan beda sangat penting karena dengannya suatu sistem dapat direalisasikan dengan mudah. Misalnya, persamaan beda [8]:

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) \qquad (2\text{-}9)$$

dapat direalisasikan dalam bentuk sebagaimana diperlihatkan pada Gambar 2-3. Kotak berlabel DELAY terdiri dari delay satu sample.

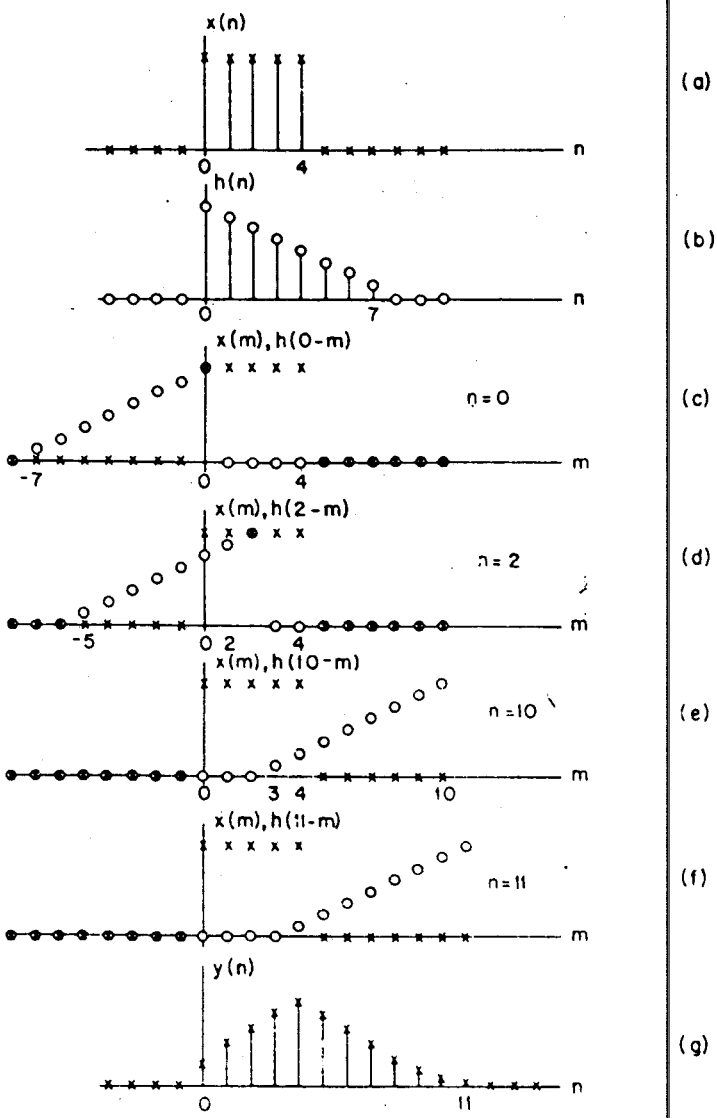


Gambar 2-3[3] Realisasi Persamaan Beda Orde Kedua

[3]Ibid, hal. 16.

II.1.4 Konvolusi dan Korelasi

Konvolusi dan korelasi merupakan operasi yang berkaitan erat dan merupakan dasar dari beberapa bidang pemrosesan sinyal.



Gambar 2-4[4] Penjelasan Konvolusi Diskrit

Konvolusi dua urutan waktu *(time series)* setara dengan hasil kali transformasi

[4]Ibid, hal. 15.

kedua urutan waktu tersebut. Gambar 2-4 memperlihatkan representasi *pictorial* sederhana tentang bagaimana proses konvolusi (persamaan 2-8) bisa diselesaikan. Gambar 2-4 (a) memperlihatkan barisan input x(n) yang tidak nol pada $0 \leq n \leq 4$. Gambar 2-4 (b) menunjukkan barisan respon impuls h(n) yang tidak nol pada interval $0 \leq n \leq 7$. Gambar 2-4 (c) sampai (f) memperlihatkan plot bersamaan dari x(m) dan h(n-m) untuk n = 0, 2, 10, dan 11. Jelaslah, untuk $n < 0$ dan untuk $n > 11$, tidak ada overlap antara x(m) dan h(n-m); sehingga y(n) sama dengan 0. Akhirnya Gambar 2-4 (g) memperlihatkan y(n), yamg merupakan konvolusi yang diinginkan.

Output dari sistem linier bisa didapat baik dengan mengalikan transformasi, $[Y_m]=[H_mX_m]$, atau sebagai konvolusi urutan waktu input $[X_k]$ dengan respon impuls $[h_k]$, yang dinyatakan dengan $[y_k] = \text{conv}\{[x_k], [h_k]\}$, yaitu merupakan perkalian dua urutan data, dengan satu urutan dibalik arah dan digeser relatif terhadap data yang lainnya, dan dinyatakan sebagai [8]:

$$y_k = \sum_{n=0}^{L} b_n x_{k-n} \qquad (2\text{-}10)$$

Fungsi korelasi dua urutan data sama seperti konvolusi, tetapi tanpa pembalikan arah salah satu urutan datanya. Oleh karenanya bisa digambarkan sebagai integral perkalian dua urutan tanpa dengan satu data digeser relatif terhadap yang lainnya atau sebagai hasil kali rata-rata dua urutan dengan satu urutan digeser. Fungsi korelasi digunakan sebagai pengukur seberapa baik dua urutan terkorelasi (kedekatan hubungan secara matematis).

II.1.5 Notasi dan Teori Dasar

Diasumsikan dua urutan waktu yang akan dikonvolusi atau dikorelasikan masing-masingnya memiliki M sampel dan dapat dinyatakan sebagai:

$$[x_k] = [x_0\ x_1\ x_2\ \ldots\ x_{M-1}]$$

$$[y_k] = [y_0\ y_1\ y_2\ \ldots\ y_{M-1}]$$

Konvolusi fungsi dua urutan $[x_k]$ dan $[y_k]$ diperoleh dengan persamaan [8]:

$$c_{xy}(n) = conv\ \{[x_k], [y_k]\}$$

$$= \sum_{k=0}^{M-1} x_k\, y_{n-k}, \quad n = 0, 1, \ldots, M-1 \qquad (2\text{-}11)$$

dan korelasinya adalah [8]:

$$r_{xy}(n) = corr\ \{[x_k], [y_k]\}$$

$$= \sum_{k=0}^{M-1} x_k\, y_{n+k}, \quad n = 0, 1, \ldots, M-1 \qquad (2\text{-}12)$$

Apabila $[x_k]$ dan $[y_k]$ merupakan urutan yang berbeda, $r_{xy}(n)$ dinamakan fungsi *cross-correlation,* dan apabila sama dinamakan fungsi *auto-correlation* dari $[x_k]$.

II.1.6 Transformasi Sinyal

Ada dua metode yang penting untuk mentransformasikan sinyal dari domain waktu ke domain frekuensi. Yang pertama adalah transformasi Z. Jika diberikan suatu barisan x(n), yang didefinisikan untuk semua n, maka transformasi z-nya adalah [8]:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \qquad (2\text{-}13)$$

dimana z adalah variabel kompleks. Fungsi kompleks tersebut didefinisikan hanya untuk nilai z tertentu dimana ia konvergen.

Sedang metode yang ke dua adalah transformasi Fourier. Transformasi ini banyak digunakan dan mempunyai penggunaan yang luas dalam aplikasi pengolahan sinyal digital. Jika diketahui x(n) adalah suatu deretan terbatas sinyal diskrit, maka transformasi Fourier dari x(n) adalah [8]:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j(2\pi/N)nk} \qquad k = 0,1,\ldots,N-1 \quad (2\text{-}14)$$

atau sering juga dituliskan dengan notasi sebagai berikut:

$$X(k) = \sum_{n-0}^{N-1} x(n)\, W^{nk} \qquad (2\text{-}15)$$

dimana $W = e^{-j(2\pi/N)}$. Dengan mudah dapat dilihat bahwa W^{nk} adalah periodik dengan periode N. Sifat periodik ini adalah satu sifat yang penting dari transformasi Fourier.

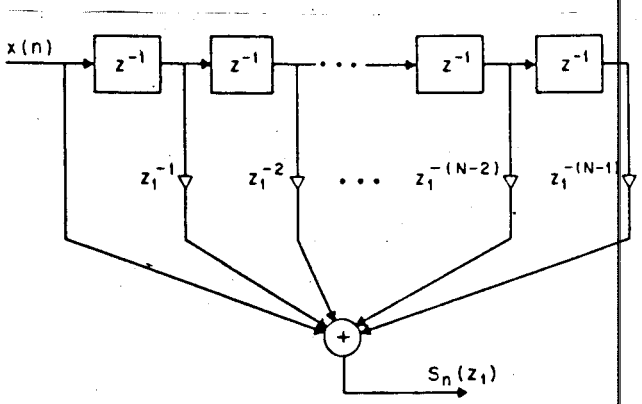
II.1.7 Hubungan antara Fast Fourier Transform (FFT) dengan Filter Bank

Secara umum spektrum suatu sinyal diskrit x(n) dapat digambarkan sebagai [8]:

$$S_n(z_1) = x(n) + x(n-1)z_1^{-1} + x(n-2)z_1^{-2} + \ldots + x(n-N+1)z_1^{-(N-1)} \qquad (2\text{-}16)$$

atau

$$S_n(z_1) = \sum_{m=n-N+1}^{n} x(m) z_1^{-(n-m)} \qquad (2\text{-}17)$$



Gambar 2-5[5] Representasi Spektrum suatu Sinyal x(n)

Gambar 2-5 menunjukkan hal ini secara jelas. Kotak berlabel z^{-1} melambangkan elemen delay. Sedang z_1^{-n} adalah koefisien pengali.

Suatu band-pass filter bank dapat digambarkan seperti yang ditunjukkan oleh Gambar 2-6.

Jika diketahui suatu sinyal diskrit x(n) ditransformasi-Fourier-kan, maka hasil transformasi adalah [8]:

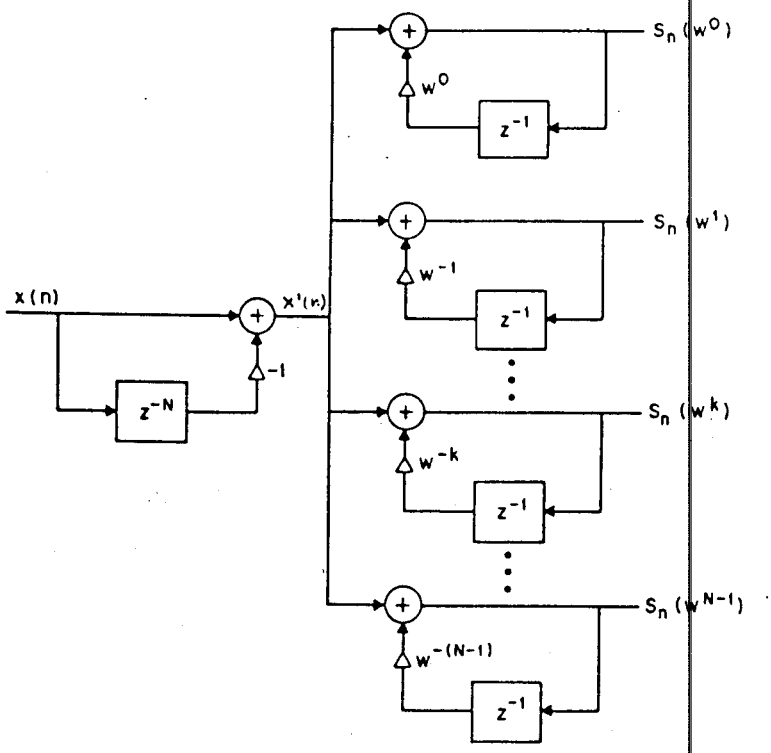
$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j(2\pi/N)nk} \qquad k=0,1,\ldots,N-1$$

$$= \sum_{n=0}^{N-1} x(n)\, W^{nk} \qquad (2\text{-}18)$$

dimana $W = e^{-j(2\pi/N)}$. Dari Gambar 2-6 di atas dapat dilihat bahwa

---

[5]Ibid, hal. 382.



Gambar 2-6[6] Implementasi Filter Bank sebagai Spektrum Analiser

$$S_n(w^0) = x'(n) + w^0 S_{n-1}(w^0)$$

$$S_n(w^1) = x'(n) + w^{-1} S_{n-1}(w^1)$$

$$S_n(w^{N-1}) = x'(n) + w^{-(N-1)} S_{n-1}(w^{N-1})$$

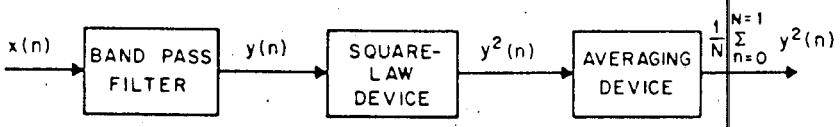
Hasil ini ekivalen dengan hasil transformasi Fourier sinyal x(n) untuk nilai k tertentu. Ini berarti suatu filter bank dapat digunakan untuk menganalisa spektrum suatu sinyal meskipun hasilnya tidak sebaik jika kita menggunakan FFT.

---

[6]Ibid, hal. 387.

II.1.8 Estimasi Spektrum Daya suatu Sinyal

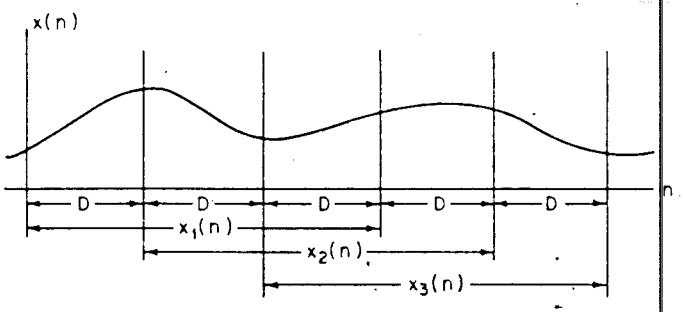
Estimasi spektrum daya dari suatu sinyal yang acak, seperti sinyal suara, bisa didapatkan dari harga rata-rata kuadrat dari output band-pass filter seperti ditunjukkan pada Gambar 2-7.



Gambar 2-7[7] Pengukuran Spektrum Daya suatu Sinyal

Jika diketahui suatu sinyal x(n) terbagi secara sama sebesar D sebanyak K segmen seperti ditunjukkan pada Gambar 2-7, maka [8]:

$$x_r(n) = x(n + (r - 1)D), \quad r = 1,2,...K. \qquad (2\text{-}19)$$



Gambar 2-8[8] Pembagian Segmen suatu Sinyal x(n)

Transformasi Fourier dari $x_r(n)$ adalah [8]:

---

[7]Ibid, hal. 414.

[8]Ibid, hal. 415.

$$X_r(k) = \sum_{n=0}^{L-1} x_r(n)w(n)e^{-j(2\pi/L)nk} \qquad (2\text{-}20)$$

dimana w(n) adalah fungsi jendela. Suatu besaran $I_r(f_k)$ yang disebut sebagai *periodogram* dapat dihitung dari[8]:

$$I_r(f_k) = (1/U) \mid X_r(k) \mid^2 \qquad (2\text{-}21)$$

dimana $f_k = k/L$ = frekuensi DFT

$$U = \sum_{n=0}^{L-1} w^2(n) = \text{energi fungsi jendela} \qquad (2\text{-}22)$$

Estimasi spektrum daya $S_x(f_k)$ dapat dihitung sebagai[8]:

$$S_x(f_k) = (1/K) \sum_{r=1}^{K} I_r(f_k) = (1/KU) \sum_{r=1}^{K} \mid X_r(k) \mid^2 \qquad (2\text{-}23)$$

Hasil ini menunjukkan bahwa estimasi spektrum daya dapat diperoleh dari harga rata-rata kuadrat magnitudo hasil transformasi Fourier sinyal x(n) (output dari band-pass filter bank). Dalam tugas akhir ini parameter spektrum daya inilah yang akan digunakan sebagai data.

II.2 TEORI DASAR SINYAL UCAPAN *(SPEECH)*

II.2.1 Informasi Linguistik

Suatu sinyal ucapan membawa beberapa informasi, yang membentuk informasi linguistik, menunjukkan apa yang ingin disampaikan pembicara, informasi individual yang menunjukkan siapa yang berbicara, dan keadaan emosi pembicara. Pada umumnya, informasi yang pertama adalah yang terpenting.

Kemampuan untuk menghasilkan dan menggunakan bahasa adalah dua hal penting diantara faktor lainnya yang membedakan manusia dengan hewan. Lebih lanjut lagi, bahasa dan kebudayaan adalah dua hal yang tak terpisahkan. Meskipun bahasa tertulis lebih efektif untuk pertukaran pengetahuan dan berlangsung lebih lama, tetapi jumlah informasi yang dapat disampaikan lewat bahasa lisan adalah jauh lebih besar. Dalam bahasa yang lebih sederhana, buku, majalah, dan yang sejenis dengan itu adalah media yang efektif untuk komunikasi satu arah, tetapi tidak cocok untuk komunikasi dua arah.

Pembuatan ucapan manusia dimulai dengan konsep ide yang akan disampaikan oleh pembicara kepada pendengar. Pembicara kemudian mengkonversikan ide tersebut ke dalam informasi linguistik dengan memilih kata-kata atau frase yang sesuai untuk menggambarkannya, dan menyusunnya ke dalam susunan tata bahasa yang tergantung juga pada hubungan antara pembicara dan pendengar. Mengikuti proses ini, otak manusia menghasilkan perintah-perintah syaraf untuk menggerakkan otot-otot dan organ-organ vokal. Proses ini dibagi dalam dua subproses: proses fisiologi yang mencakup syaraf dan otot, dan proses fisik yang mencakup pembuatan dan propagasi sinyal ucapan tersebut. Suatu kalimat dibentuk dengan menggunakan unit-unit kata, yang mana kata-kata tersebut disusun oleh silabil-silabil, dan setiap silabil-silabil itu terdiri dari fonem-fonem. Pada akhirnya fonem-fonem dapat dipecah lagi menjadi vokal dan konsonan.

Meskipun jumlah kata-kata dalam suatu bahasa adalah sangat besar, dan secara konstan jumlah kata-kata ini bertambah. Namun jauh lebih kecil dari

jumlah kemungkinan kombinasi dari silabil dan fonem. Pada umumnya jumlah kata yang sering digunakan berkisar antara 2000 hingga 3000 kata, dan setiap orang rata-rata menggunakan 5000 hingga 10000 kata.[9]

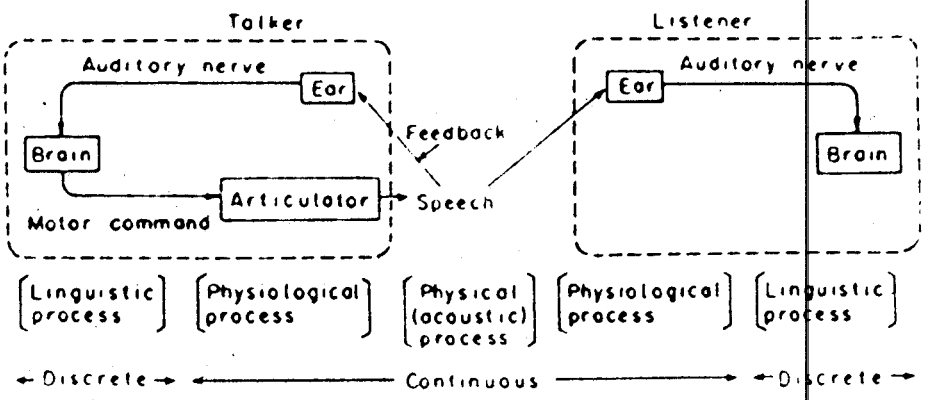
II.2.2 Ucapan dan Pendengaran

Ucapan dihasilkan untuk, dan dengan asumsi, diterima dan dimengerti oleh pendengar. Ini berarti bahwa pembuatan ucapan secara intrinsik berhubungan dengan kemampuan mendengar. Sinyal ucapan dihasilkan oleh organ-organ vokal dan ditransmisikan lewat udara ke telinga pendengar seperti ditunjukkan oleh Gambar 2-9.

Pada telinga, ia mengaktifkan organ-organ pendengaran untuk menghasilkan impuls-impuls syaraf yang ditransmisikan ke otak pendengar melalui sistem pendengaran.

---

[9] Furui, Sadaoki, Digital Speech, Processing, Synthesis, and Recognition, Marcel Dekker, New York, 1989, hal. 6.



Gambar 2-9[10] Rantai Ucapan

Ini memungkinkan informasi linguistik yang ingin disampaikan oleh pembicara dapat dimengerti oleh pendengar. Sinyal ucapan yang sama, juga ditransmisikan ke telinga pembicara sendiri, yang mengizinkannya untuk secara kontinyu mengontrol organ-organ ucapannya yang diterimanya sebagai umpan balik *(feedback)*.

Mekanisme pendengaran manusia adalah suatu sistem yang sangat canggih, yang pada saat ini telah banyak dicoba untuk didekati misalnya dengan metoda kecerdasan buatan *(artificial intelligent)*. Salah satu kelebihan yang menonjol dari mekanisme ini adalah kemampuannya untuk memilah-milah informasi yang disampaikan secara simultan, bahkan ketika suara diucapkan secara berbeda dengan aksen yang kuat.

Tetapi pada sisi yang lain, kemampuan mendengar manusia mempunyai kelemahan. Sebagai contoh, telinga manusia tidak dapat membedakan dua nada

---

[10]Ibid, hal. 7.

yang berbeda yang mempunyai frekuensi yang sama.

II.3 PEMROSESAN DIGITAL SINYAL UCAPAN

II.3.1 Digitisasi

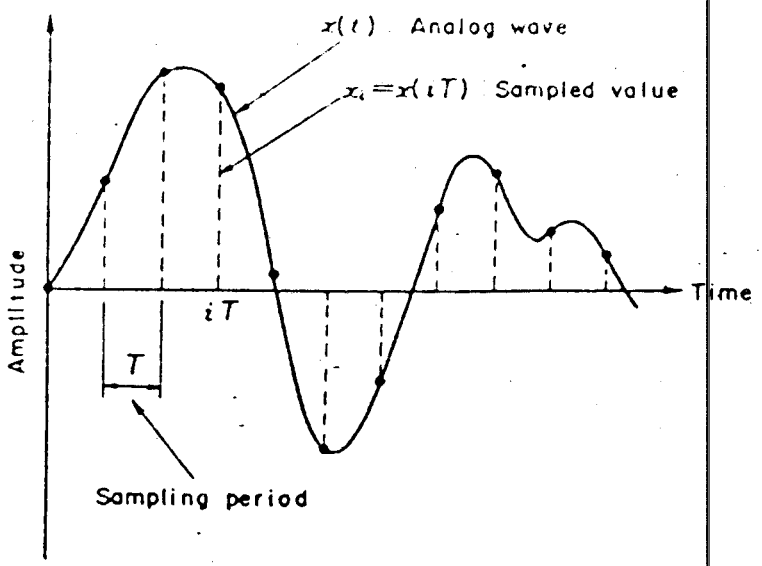
Sinyal ucapan, atau gelombang ucapan dapat diubah ke objek yang dapat diproses secara elektronik dengan mengkonversikannya menjadi sinyal listrik dengan bantuan mikropon. Sinyal listrik pada umumnya ditransformasikan dari sinyal analog ke sinyal digital, terutama sinyal ucapan dengan dua alasan. Pertama, teknik digital memberikan fasilitas pemrosesan sinyal yang canggih yang tidak dapat direalisasikan dengan teknik analog. Kedua, pemrosesan digital jauh lebih handal dan dapat diselesaikan dengan rangkaian yang kompak. Perkembangan yang cepat dari komputer dan teknologi rangkaian terpadu dalam hubungannya dengan pertumbuhan jaringan komunikasi digital telah ikut memacu aplikasi teknik pemrosesan digital pada pemrosesan sinyal ucapan.

Analog-to-digital Conversion (ADC), yang mengacu pada digitisasi, terdiri dari pengambilan sampel *(sampling)*, kuantisasi *(quantizing)*, dan proses pengkodean *(coding)*. Sampling adalah proses untuk menggambarkan variasi sinyal yang kontinyu ke dalam deretan nilai-nilai yang periodik. Kuantisasi adalah proses untuk merepresentasikan nilai sinyal ke dalam satu set nilai yang terbatas. Pengkodean berhubugan dengan penunjukkan bilangan yang aktual untuk mewakili nilai tertentu. Untuk tugas itu, pengkodean biner, yang menggunakan bilangan-bilangan biner, pada umumnya digunakan. Proses ini memungkinkan untuk mengkonversikan sinyal analog yang kontinyu ke dalam deretan kode-kode yang

dipilih dari satu himpunan tertentu.

II.3.2 Sampling

Dalam proses sampling, sinyal analog x(t) dikonversikan ke dalam deretan nilai-nilai $\{x_i\} = \{x(iT)\}$ dalam waktu yang periodik $t_i = iT$ (i adalah integer), seperti ditunjukkan pada Gambar 2-10. Disini T [s]



Gambar 2-10[11] Proses Sampling dalam Daerah Waktu

disebut perioda sampling, dan kebalikannya adalah S = 1/T [Hz], disebut frekuensi sampling. Jika T terlalu besar, sinyal yang asli tidak dapat direproduksi dari deretan hasil sampel; kebalikannya jika T terlalu kecil, sampel yang tidak berguna akan termasuk dalam pembentukan kembali sinyal

---

[11]Ibid, hal. 46.

yang asli. Untuk ini teori sampling dari Shannon dan Someya digunakan dalam hubungannya dengan lebar bidang dari frekuensi sinyal analog yang akan disampel dan periode sampling yang diusulkan untuk mengatasi masalah ini (Shannon dan Weaver, 1949)[12].

Teori ini mengatakan, jika sinyal analog x(t) yang mempunyai lebar bidang terbatas antara 0 dan W [Hz] dan bila x(t) disampel setiap T = 1/2W [s], sinyal yang asli dapat direproduksi secara lengkap dengan [4]:

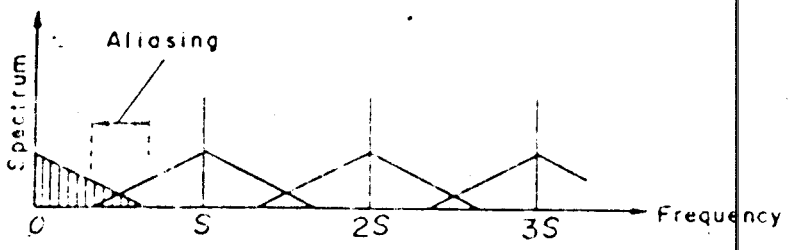
$$x(t) = \sum_{i=-\infty}^{\infty} x(i/2W)(\sin\{2\pi W(t-i/2W)\}/2\pi W(t-i/2W)) \qquad (2\text{-}24)$$

Disini, x(i/2W) adalah nilai sampel dari x(t) pada t_i = i/2W (i adalah integer). Lebih jauh, 1/T = 2W [Hz] disebut *Nyquist rate.*

Sebagai contoh, sinyal telepon yang biasa disampel setiap T = 1/8000 [s], karena lebar bidang W terbatas di bawah 4 kHz. Frekuensi sampling untuk pemrosesan digital sinyal ucapan biasanya diset antara 6 dan 16 kHz. Bahkan untuk beberapa penggunaan khusus, frekuensi sampling yang digunakan adalah 20 kHz, seperti untuk penyamplingan konsonan misalnya. Untuk sinyal-sinyal yang lebar bidang frekuensinya tidak diketahui digunakan filter untuk membatasinya. Pembatasan juga dilakukan untuk menghindari efek aliasing, yaitu distorsi sinyal oleh komponen frekuensi yang lebih tinggi seperti ditunjukkan pada Gambar 2-11.

---

[12]Loc. cit.



Gambar 2-11[13] Efek Aliasing

II.3.3 Kuantisasi dan Pengkodean

Selama kuantisasi, seluruh jangkauan amplitudo kontinyu dibagi ke dalam subrange yang terbatas, dan amplitudo di dalam subrange yang sama, ditunjukkan oleh nilai yang sama.

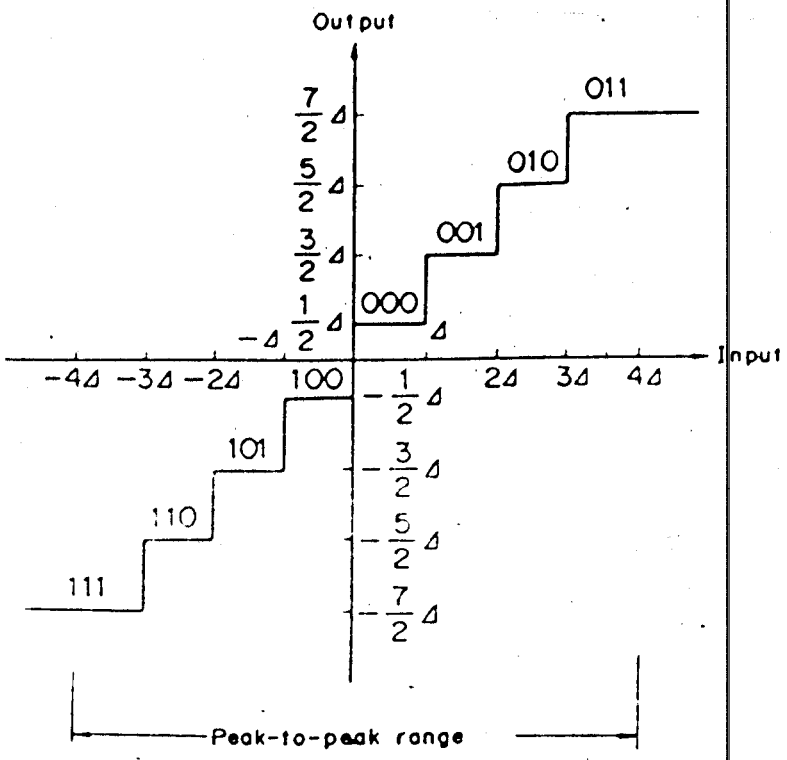
Gambar 2-12 mencontohkan karakteristik input-output dari quantizer 8 level (3 bit), dimana $\triangle$ adalah step kuantisasi. Dalam contoh ini, setiap kode dikondisikan sehingga setiap kode langsung merepresentasikan nilai amplitudo. Karakteristik kuantisasi tergantung pada jumlah level dan step kuantisasi $\triangle$. Jika suatu sinyal diasumsikan akan dikuantisasikan oleh B [bit], jumlah level diset 2^B untuk menjamin penggunaan yang paling efisien dari kode biner. $\triangle$ dan B harus dipilih secara bersama-sama untuk memenuhi secara benar range dari sinyal. Jika kita asumsikan bahwa $| x_i | \leq x_{max}$, maka sebaiknya diset [4]:

$$2x_{max} = \triangle 2^B \qquad (2\text{-}25)$$

Perbedaan antara nilai yang disampel yang telah dikuantisasi x_i dengan sinyal analog yang asli x_i, $e_i = x_i - x_i$, disebut kesalahan kuantisasi, distorsi kuantisasi,

---

[13]Ibid, hal. 52.



Gambar 2-12[14] Karakteristik Input-Output kuantisasi 3 bit

atau noise kuantisasi. Ini dapat dilihat pada Gambar 2-12 bahwa noise kuantisasi memenuhi [4]:

$$-\triangle/2 \leq e_i \leq \triangle/2$$

bila $\triangle$ dan B diset untuk memenuhi persamaan (2-25).

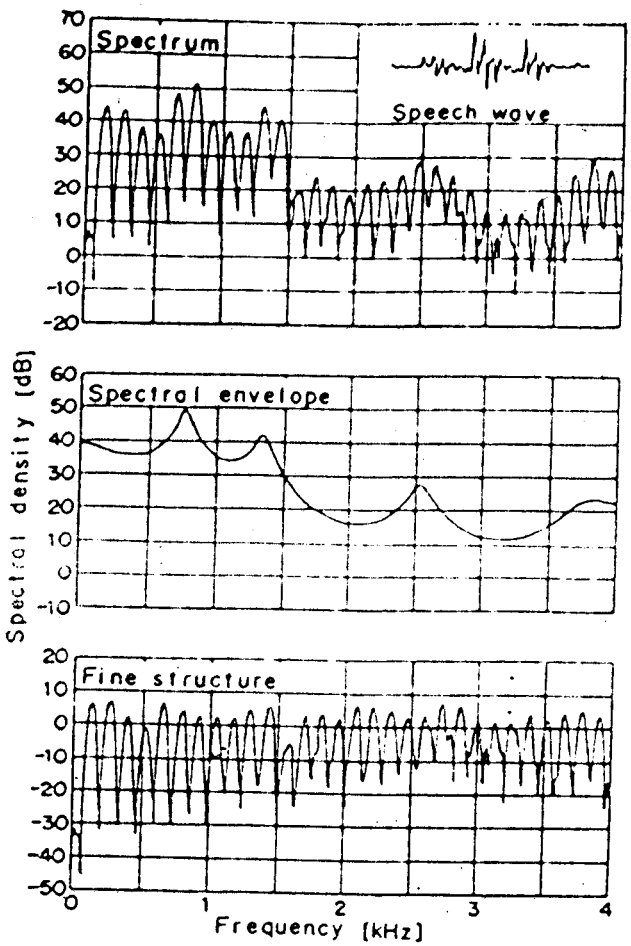
II.4 ANALISA SPEKTRAL

II.4.1 Struktur Spektral Sinyal Ucapan

Gelombang sinyal ucapan biasanya dianalisa menggunakan sifat-sifat spektralnya, seperti spektrum

---

[14]Ibid, hal. 49.



Gambar 2-13[15] Amplop Spektral dan Spektral Struktur Halus untuk Vokal /a/

frekuensi dan fungsi autokorelasi, sebagai ganti dari langsung menggunakan gelombang itu sendiri. Ada dua alasan penting untuk ini. Pertama adalah bahwa sinyal ucapan dapat diproduksi kembali dengan menjumlahkan gelombang-gelombang sinus, yang amplitudo dan phasenya berubah secara lambat. Kedua adalah, sifat-sifat penting yang diperlukan untuk menafsirkan sinyal ucapan oleh telinga manusia tercakup dalam informasi spektral, yang mana phase sinyal tidak

[15]Ibid, hal. 52.

memainkan peran yang penting.

Kepadatan spektrum energi dalam waktu singkat, yaitu spektrum waktu singkat *(short-time spectrum)* sinyal ucapan, dapat dianggap sebagai hasil dari dua elemen: amplop spektral *(spectral envelope)* yang berubah secara lambat sebagai fungsi dari frekuensi, dan struktur halus spektral *(spectral fine structure)* yang berubah secara cepat sebagai fungsi dari frekuensi. Struktur halus spektral menghasilkan pola yang periodik untuk *voice sound* (a, e, dsb) tetapi tidak untuk *unvoice sound* (s, f, dsb) seperti ditunjukkan pada Gambar 2-13. Amplop spektral, atau keseluruhan sifat spektral, merefleksikan karakteristik resonansi dan antiresonansi organ-organ bicara. Sedangkan struktur halus spektral menunjukkan periodisitas dari sumber suara.

Metoda-metoda untuk ekstraksi amplop spektral dapat dibagi dalam *parametric analysis* (PA) dan *nonparametric analysis* (NPA). Dalam metoda PA, suatu model yang memenuhi kondisi sinyal dipilih dan diaplikasikan ke sinyal dengan meng-adjust parameter-parameter yang merepresentasikan model. Sedang metoda NPA, dapat diaplikasikan ke berbagai bentuk sinyal karena mereka tidak memodelkan sinyal tersebut. Jika model hampir secara keseluruhan memenuhi kondisi sinyal, maka metoda PA dapat merepresentasikan sifat-sifat sinyal secara lebih efektif daripada metoda NPA. Metoda-metoda yang umum digunakan untuk menganalisa spektrum sinyal ucapan dan sifat-sifat spektralnya ditunjukkan dalam Tabel 2.1. Dalam bab ini hanya akan dijabarkan lebih lanjut metoda filter bank, karena metoda ini yang digunakan dalam tugas akhir ini.

Tabel 2.1[16] Metoda-metoda untuk analisa spektrum ucapan.

Tipe	Metoda Analisis	Parameter	Sifat-sifat
NPA	i. Short-time autocorrelation	$\phi(m)$	Amplop spektral dan struktur halusnya dikonvolusi.
	ii. Short-time Spectrum	$S(\Omega)$	Amplop spektral dan struktur halusnya dimultiplikasi.
	iii. Cepstrum	$c(\sigma)$	Amplop spektral dan struktur halusnya dipisahkan dalam domain frekuensi. Menggunakan FFT.
	iv. Band-pass filter bank	harga rms dari out put filter	Amplop spektral global.
	v. Zero-crossing analysis	zero-crossing rate	Frekuensi Formant.

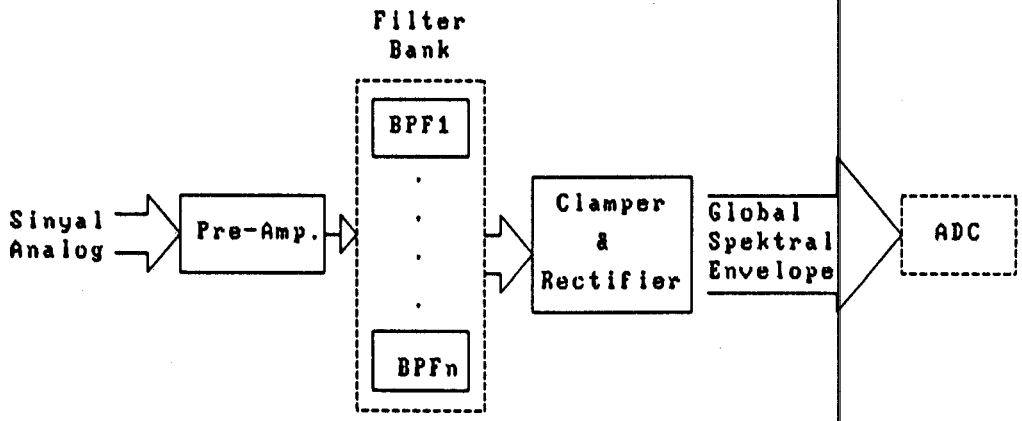
[16]Ibid, hal. 55.

PA	i. Analysis-by-synthesis	Formant, bandwidth, dst.	Frekuensi Formant yang akurat. Memerlukan iterasi yang kompleks.
	ii. Linear predictive coding		Pemodelan spektral. Parameter diperoleh dari autocorr. atau covariance.
	ii-a. Maximum likelihood	α_i	Synthesis filter yang stabil. Memerlukan "time window".
	ii-b. Covariance method	α_i	Synthesis filter yang stabil. Cocok untuk analisa "short time".

II.4.2 Metoda Filter Bank[17]

Metoda filter bank adalah salah satu dari metoda NPA. Metoda ini membutuhkan jumlah perhitungan yang kecil dan karenanya cocok untuk implementasi hardware. Secara umum, metoda ini terdiri dari beberapa band-pass filter yang disusun sedemikian sehingga *center frequency* terdistribusi secara sama dalam skala logaritmik dalam rentang 300 hingga 3000 Hz, dan dengan atenuasi 3-db pada titik yang berdampingan. Output dari setiap band-pass filter disearahkan, dan disampel untuk memperoleh spektral amplop dari sinyal input suara.

[17] Ibid, hal. 67.



Gambar 2-14 Metoda Filter Bank

Dalam tugas akhir ini pengolahan data lebih lanjut dilakukan secara digital dengan mikroprosesor TMS 32010. Dari beberapa kali penyamplingan untuk satu kata, akan diperoleh beberapa frame spektrum frekuensi untuk kata tersebut, yang mana tiap-tiap frame terdiri dari 16 point frekuensi. Estimasi spektrum daya sinyal suara diperoleh dari rata-rata beberapa frame spektrum frekuensi tersebut[18]. Data inilah yang disimpan sebagai *template reference*. Dan data ini pula yang akan digunakan untuk pembandingan data.

II.5 PELENGKUNGAN WAKTU DINAMIK

II.5.1 Dynamic Programming Matching

Meskipun pembicara yang sama mengucapkan kata yang sama, durasinya berubah setiap waktu dengan ekspansi atau kontraksi yang nonlinier. Karena itu Pelengkungan Waktu Dinamik *(Dynamic Time Warping)* adalah tahap yang

---

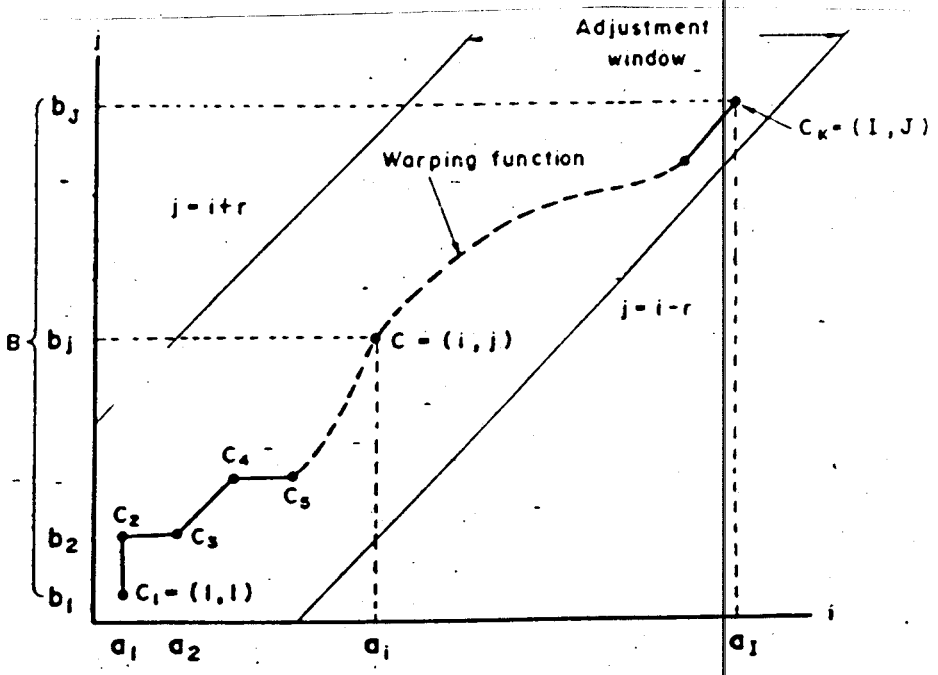
[18] Rabiner, Lawrence R., Bernard Gold, op.cit, hal. 414.

penting dalam proses *recognition*. Proses DTW melakukan ekspansi atau kontraksi pada sumbu waktu untuk mencocokkan posisi phonem yang sama antara *input speech* dan *reference templates*.

Proses ini dapat diselesaikan secara efisien dengan menggunakan teknik pemrograman dinamik *(Dynamic Programming/DP)*. Teknik ini mempunyai pengaruh yang sangat besar pada *speech recognition*, dan menjadi salah satu teknik yang penting dan banyak digunakan.

Sebagai contoh penggunaan teknik pemrograman dinamik, diasumsikan ada dua deretan *feature vector* dalam kawasan waktu yang akan dibandingkan [3]:

$$A = a_1, a_2, \ldots, a_I \quad dan \quad B = b_1, b_2, \ldots, b_J \qquad (2\text{-}26)$$



Gambar 2-15[19] DTW antara Dua Barisan, A dan B

Jika kita anggap suatu ruang dua dimensi dibentangkan antara A dan B seperti

[19]Furui, Sadaoki, op. cit., hal. 245.

ditunjukkan pada Gambar 2-15, fungsi pelengkungan waktu yang menunjukkan hubungan antara deretan pada sumbu waktu A dan B dapat direpresentasikan sebagai deretan titik-titik "lattice" pada bidang, c = (i,j), sebagai [3]:

$$F = c_1, c_2, \ldots, c_k, \ldots, c_K \quad c_k = (i_k, j_k) \tag{2-27}$$

Bila jarak spektral antara kedua *feature vector* a_i dan b_j direpresentasikan oleh d(c) = d(i,j), maka jumlah jarak dari awal hingga akhir deretan sepanjang F dapat direpresentasikan oleh [3]:

$$D(F) = \left(\sum_{k=1}^{K} d(c_k) w_k\right) / \left(\sum_{k=1}^{K} w_k\right) \tag{2-28}$$

Semakin kecil nilai ini, semakin baik kecocokkan pola antara A dan B. Disini, w_k adalah fungsi pembobotan positif terhadap F.

Persamaan 2-28 dapat diminimisasi dengan kondisi sebagai berikut:

1. Kondisi Monotonisitas dan Kontinyuitas.

$$0 \leq i_k - i_{k-1} \leq 1, \quad 0 \leq j_k - j_{k-1} \leq 1 \tag{2-29}$$

2. Kondisi Perbatasan

$$i_1 = j_1 = 1, \quad i_K = I, \; j_K = J \tag{2-30}$$

3. Kondisi *adjustment* window

$$| i_k - j_k | \leq r, \quad r = \text{konstanta} \tag{2-31}$$

Kondisi ke-3 dimaksudkan untuk mencegah ekspansi atau konstraksi yang ekstrim. Pendefinisian w_k adalah sedemikian rupa sehingga penyebut pada persamaan (2-28) menjadi konstanta yang tidak tergantung pada F. Sebagai contoh, jika $w_k = (i_k - i_{k-1}) + (j_k - j_{k-1})$ $(i_0 = j_0 = 0)$, w_k menjadi [3]:

$$\sum_{k=1}^{K} w_k = I + J \qquad (2\text{-}32)$$

dan persamaan 2-28 menjadi [3]:

$$D(F) = (1/(I+J)(\sum_{k=1}^{K} d(c_k)w_k) \qquad (2\text{-}33)$$

Karena fungsi obyektif yang akan diminimisasi menjadi penjumlahan, maka minimisasi dapat diselesaikan tanpa harus menelusuri semua kemungkinan untuk F. Penjumlahan untuk sebagian dari deretan $c_1, c_2, \ldots, c_k$ ($c_k = (i,j)$) adalah [3]:

$$g(c_k) = g(i,j) = \min_{c_1,\ldots,c_{k-1}} \left| \sum_{m=1}^{K} d(c_m)w_m \right|$$

$$= \min_{c_1,\ldots,c_{k-1}} \left| \sum_{m=1}^{K} d(c_m)w_m + d(c_k)w_k \right|$$

$$= \min_{k-1} \left| \min_{c_1,\ldots,c_{k-1}} \{ \sum_{m=1}^{k-1} d(c_m)w_m \} + d(c_k)w_k \right|$$

$$= \min_{c_{k-1}} \left| g(c_{k-1}) + d(c_k)w_k \right| \qquad (2\text{-}34)$$

Persamaan-persamaan di atas adalah penurunan dari pemrograman dinamik. Menggunakan ketiga kondisi untuk F, dan formulasi w_k seperti yang diterangkan di atas, persamaan 2-34 dapat dituliskan sebagai [3]:

$$g(i,j) = \min \begin{pmatrix} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j)) \\ g(i-1,j) + d(i,j) \end{pmatrix} \qquad (2\text{-}35)$$

Karena itu, jarak antara kedua deretan waktu A dan B setelah DTW dapat

II.6 SISTEM PENGENAL SUARA[20]

II.6.1 Pendahuluan

Bentuk komunikasi yang paling alami antara mesin dan manusia adalah menggunakan suara. Pada bentuk komunikasi ini mesin memberi respon berdasarkan cara komunikasi yang lebih disukai manusia, dari pada sebaliknya. Sistem pengenal suara adalah langkah terakhir upaya penyederhanaan bentuk komunikasi antara manusia dengan mesin.

Pada awalnya, komunikasi antara manusia dengan mesin atau komputer harus mengikuti prosedur operasional dari mesin tersebut. Untuk mengontrol suatu mesin atau komputer diperlukan pengetahuan tentang "bahasa" dalam urutan dan format yang dimengerti oleh mesin. Setiap penyimpangan sekecil apapun akan menyebabkan kesalahan operasinya yang sulit untuk dideteksi karena kompleksitas mesin.

Pembuatan Sistem Pengenal Suara dengan kemampuan terbatas telah memungkinkan manusia untuk "berbicara" secara langsung kepada mesin. Input ke mesin menjadi lebih sederhana karena operator menggunakan bahasanya yang alami.

II.6.2 Jenis-Jenis Sistem Pengenal Suara.

Sistem-sistem pengenal suara dapat digolongkan menjadi dua kategori: *sistem pengenal ucapan kontinyu, dan sistem pengenal ucapan terisolasi*. Pada

---

[20] MARTIN, Thomas B., Practical Applications of Voice Input to Machines, PROCEEDING OF IEEE, Vol. 64, No. 4, April 1976.

sistem pengenal ucapan terisolasi, sistem memerlukan input dengan jeda (*short pause)* antara kata dan pada akhir ucapan agar ucapan dapat dikenali.

Durasi minimum dari suatu jeda berada dalam orde 100 ms. Jeda yang lebih pendek dari 100 ms dapat disalah-tafsirkan sebagai "penutupan" dari suatu *stop consonant* di tengah kata.

Kecepatan berbicara yang dapat digunakan untuk sistem pengenal ucapan terisolasi lebih rendah daripada untuk sistem pengenal ucapan kontinyu. Kecepatan berbicara sekitar 300 kata per menit dapat digunakan untuk sistem pengenal ucapan kontinyu. Sedang sistem pengenal ucapan terisolasi dapat menerima input sekitar 125 kata per menit. Dan rata-rata panjang kata dari ucapan yang akan dikenali adalah sekitar 2-4 detik.

II.6.3 Proses Adaptasi

Salah satu hambatan yang besar dalam pembuatan sistem pengenal ucapan adalah variasi ucapan (intonasi, durasi, dll.) dari tiap-tiap individu. Perbedaan seperti itu membuat upaya realisasi sistem pengenal ucapan yang "universal" menjadi perkerjaan yang sulit. Salah satu jalan singkat yang dapat ditempuh untuk mengatasi hal ini adalah dengan membuat sistem pengenal ucapan yang sesuai dengan karakteristik dari pemakai.

II.6.4 Pertimbangan-Pertimbangan Operasional Untuk Sistem Pengenal Suara dengan Kosa Kata Terbatas.

a. *Noise Background.*

Aspek praktis yang penting untuk dipertimbangkan dari sistem pengenal suara dengan kosa kata terbatas adalah *interferensi* dari sinyal akustik dan *noise background*. Jika suatu sistem menggunakan mikropon dengan kualitas yang baik, ia juga akan menerima input dari suara-suara yang lain yang ada di dekatnya. Ada dua penyelesaian untuk hal ini. Pertama, sistem ditempatkan di lingkungan yang terlindung dari noise. Tetapi mobilitas dan penggunaannya menjadi sangat terbatas. Sehingga cara ini sangat jarang digunakan. Cara yang kedua adalah, mengurangi noise dari mikropon itu sendiri. Misalnya dengan penggunaan mikropon yang dekat dengan sumber suara (*contact microphone)* yang akan dikenali atau penggunaan rangkaian "noise-cancelling".

b. ***Breath Noise.***

Jika menggunakan *contact microphone* untuk pengambilan sinyal input, maka faktor penting yang harus diperhatikan adalah, napas dari pembicara dapat menghasilkan noise (*breath noise*). Noise ini dihasilkan dari kecenderungan yang kuat untuk menghasilkan hembusan pada akhir kata dan menarik napas pada waktu akan mengucapkan suatu kata. Dalam sistem pengenal suara dengan kosa kata terbatas, noise ini dapat menyebabkan persoalan yang serius.

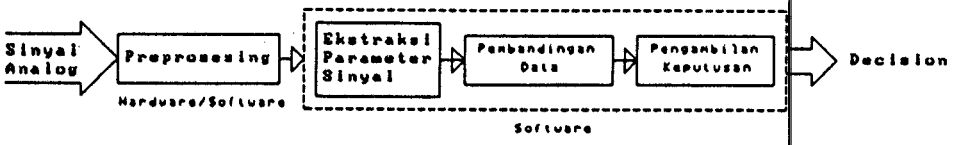
Sinyal ucapan yang diucapkan secara baik, akan memberikan batas kata yang jelas. Terlepas dari metode yang digunakan, penentuan batas kata adalah penting untuk menentukan awal dan akhir dari suatu kata. Ketepatan penentuan batas kata mempengaruhi akurasi pengenalan kata tersebut.

c. Kestabilan Data Referensi *(Reference Data)*

Sistem Pengenal Suara melakukan pengenalan dengan cara membandingkan input ucapan yang tidak diketahui dengan sekumpulan data yang telah disimpan, yang diperoleh sebelumnya dari pemakai sistem. Sekali data referensi telah diperoleh, diharapkan sistem pengenal suara dapat digunakan dengan sedikit atau tanpa latihan kembali *(retraining)*.

II.6.5 Sistem Pengenal Suara Umum dengan Kosa Kata Terbatas

Ada tiga proses utama dalam suatu sistem pengenal suara, yaitu: pertama, preprosesing, yang mencakup pengambilan sinyal input dengan transduser mikropon. Kedua, ekstraksi parameter sinyal. Dan ketiga adalah proses pembandingan data dan pengambilan keputusan. Gambar 2-16 menggambarkan sistem ini secara umum.



Gambar 2-16 Blok Diagram Sistem Pengenal Suara

a. Preprosesing.

Preprosesing bertujuan untuk melakukan transformasi sinyal suara (ucapan) -

besaran analog - ke bentuk yang dapat dideteksi oleh sistem pengenal suara. Sinyal ucapan tidak bisa digolongkan sebagai sinyal yang periodik, dan bukan pula sinyal yang non-periodik, tetapi dapat dianggap sebagai sinyal yang merupakan gabungan antara sinyal periodik dan sinyal non-periodik *(quasi-periodic signal)*. Sehingga teknik analisa yang digunakan harus dapat merefleksikan sifat-sifat temporal dari sinyal suara, misalnya *spectral features*-nya. Ada dua teknik yang dapat digunakan, yaitu: *time-domain* dan *frequency-domain analytical technique.*

Representasi sinyal suara (ucapan) dengan cara *frequency-domain* memiliki beberapa keuntungan, antara lain: 1) lebih baik dari sistem pendengaran suara manusia yang kurang peka yang hanya mampu melakukan analisa frekuensi secara kasar, 2) analisa akustik dapat menggambarkan secara tepat bunyi ucapan dengan menerapkan konsep model frekuensi natural.

Fungsi periodik terhadap waktu dari sinyal ucapan memiliki spektrum daya yang terbatas yang terletak pada titik-titik diskrit dari suatu spektrum garis. Suatu fungsi aperiodik yang memiliki energi terbatas dan dapat di-transformasi Fourier-kan memiliki spektrum kerapatan energi yang kontinyu terhadap frekuensi. Untuk menganalisa suatu sinyal ucapan, diperlukan untuk memperoleh "distribusi spektrum energi" dan variasinya terhadap fungsi waktu.

Analisa sinyal ucapan dapat dilakukan langsung dengan menganalisa spektrum analognya melalui penggunaan beberapa band-pass Filter , Fast Fourier Transform (FFT), atau dengan teknik *linear predictive coding* (LPC). FFT menghasilkan spektrum diskrit, yang dengan sampling rate yang cukup, dapat mendekati transformasi Fourier kontinyu. Ada beberapa jenis "window" data yang

berbeda yang dapat digunakan dalam FFT. Pemilihan window yang digunakan identik dengan pemilihan respon filter dalam analisa spektrum analog.

b. Ekstraksi Parameter Sinyal.

Proses kunci dalam sistem pengenalan pola adalah ekstraksi parameter sinyal. Meskipun ada beberapa teknik klasifikasi yang dapat digunakan, tetapi tidak ada satupun teknik klasifikasi tersebut yang benar-benar memadai untuk mencakup semua nilai parameter yang ada. Semakin optimum hasil ekstraksi parameter sinyal yang diperoleh, semakin dapat mengurangi tingkat kompleksitas pengklasifikasi yang dibutuhkan. Parameter sinyal yang dipilih harus bersifat cukup universal untuk memungkinkan penambahan kosa kata baru ke sistem kapan saja diinginkan. Pemilihan ekstraksi parameter sinyal yang baik dan handal adalah penting, agar sistem dapat beroperasi dalam berbagai lingkungan operasi. Ada beberapa jenis parameter sinyal yang dapat digunakan, misalnya parameter spektrum daya sinyal, cepstrum, atau *linear predictive coefficient*. Dalam tugas akhir ini parameter yang digunakan adalah parameter spektrum daya sinyal.

c. Pengklasifikasian.

Proses klasifikasi atau pengambilan keputusan dilakukan secara software. Proses ini secara garis besar dapat digambarkan sebagai proses pembandingan secara berurutan pola input yang diperoleh dengan beberapa pola referensi yang telah disimpan sebelumnya, dan berusaha untuk "menemukan" pola referensi yang paling cocok dengan pola sinyal input untuk pengambilan keputusan. Pola sinyal

referensi yang jumlah jaraknya paling minimum terhadap pola sinyal input, akan dipilih sebagai pola yang paling "match". Tetapi nilainya harus lebih kecil dari level threshold tertentu untuk dapat dianggap benar-benar sebagai kata yang dimaksud. Nilai level threshold ini diperoleh dari hasil beberapa kali percobaan. Nilai ini tidak sama untuk lingkungan operasi dan metode yang berbeda. Untuk memperoleh pola referensi, sistem harus "dilatih" oleh pemakai sistem yang bersangkutan sampai diperoleh pola yang relatif stabil.

Dapat juga terjadi kemungkinan bahwa pola sinyal input tidak memenuhi kecocokan dengan semua pola referensi yang telah disimpan. Jika ini terjadi maka dianggap sistem tidak mengenali kata tersebut.

Dalam tugas akhir ini digunakan metoda pemrograman dinamik untuk melakukan fungsi ini. Metoda-metoda lain yang dapat digunakan contohnya adalah *Hidden Markov Model* (metoda statistik), dan yang sekarang sedang banyak diteliti untuk dikembangkan lebih lanjut adalah metoda kecerdasan buatan seperti *Artificial Neural Network* (Jaringan Syaraf Tiruan). Metoda yang terakhir ini memiliki banyak kelebihan dibandingkan dengan metoda konvensional lainnya, karena dengan menerapkan metoda ini memungkinkan sistem untuk dapat melakukan proses belajar (*learning*) untuk memperbaiki dan meningkatkan performansi sistem tersebut.

II.7 MIKROPROSESOR TMS 32010[19]

Sebagai komponen utama dari sistem yang direncanakan,prosesor TMS32010 memerlukan pembahasan tersendiri baik dari segi perangkat keras maupun programnya. Bab ini membahas TMS32010 secara agak rinci mengenai arsitektur, organisasi memori, logic unit, kontrol sistem, fungsi I/O serta pin-out dan deskripsi sinyal.

II.7.1 Deskripsi Umum

Sebagai salah satu keluarga TMS320, TMS32010 adalah prosesor sinyal digital chip tunggal 16/32 bit yang menggabungkan fleksibilitas *high-speed controller* dengan kapabilitas numeris dari array prosesor. Kemampuan ini dimungkinkan oleh penerapan aristektur Harvard dalam perancangannya serta tersedianya instruksi-instruksi khusus pengolahan sinyal digital (PSD) yang dimilikinya. Dengan kelebihan tersebut, TMS32010 mampu mengeksekusi hingga 6,25 juta instruksi per detik.

TMS32010, yang merupakan anggota pertama dari keluarga TMS320, merupakan mikroprosesor yang mampu melakukan perkalian 16x16 bit dalam satu siklus 200 nano detik. Di dalamnya tersedia *on-chip* program memory 4K word. TMS32010 juga dapat digunakan sebagai mikrokomputer dengan 1,5K word on-chip program ROM dan 2,5K word *off-chip* program memory.

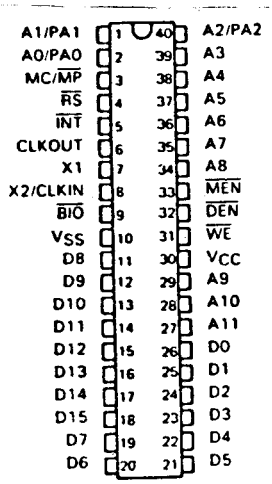
---

[19] ---, First-Generation TMS320 User's Guide, Texas Instrument Inc., 1988, hal. 2-2 s/d 4-5.

II.7.2 Pinout dan Deskripsi Sinyal

Gambar 2-17 memperlihatkan pinout kemasan DIP untuk TMS32010. Berikut ini merupakan deskripsi fungsi sinyal dimiliki oleh TMS32010:

A11-A0 : address bus dari memori program A11 (MSB) hingga A0 (LSB) dan port address PA2 (MSB) hingga PA0 (LSB). Address A11 hingga A0 selalu aktif dan tidak pernah menuju *high impedance.* Selama eksekusi insrtuksi IN dan OUT, pin A2 - A0 membawa poet address PA2 - PA0.



Gambar 2-17 Konfigurasi Pin-pin TMS32010

D15-D0 : Data bus paralel D15 (MSB) hingga D0 (LSB). Data bus selalu dalam keadaan high impedance kecuali ketika WE aktif (low).

BIO : Polling input eksternal. Di-pol oleh instruksi BIOZ. Jika low, device akan mencabang *(branch)* ke alamat yang ditentukan oleh instruksi.

DEN : Data enable untuk *device* dalam meng-input data. Saat aktik low, DEN menunjukkan bahwa

device akan menerima data dari data bus. DEN hanya aktif selama siklus pertama dari instruksi IN. MEN dan WE akan selalu tidak aktif (high) apabila DEN aktif.

INT : Interrupt input eksternal. Sinyal interrupt dibangkitkan dengan menerapkan *negative-going edge* ke pin INT. Negative-going edge digunakan untuk me-*latch* interrupt flag register (INTF) sampai instruksi dilayani oleh *device*. Level aktif low juga akan diterima.

MC/MP : Pin pemilih mode memori. High akan memilih mode mikrokomputer, dan low untuk mode mikroprosesor.

MEN : Memory enable. MEN akan aktif low pada setiap *machine cycle* terkecuali ketika WE dan DEN aktif. MEN adalah sinyal kontrol yang dibangkitkan *device* untuk meng-enable pengambilan *(fetching)* instruksi dari memori program.

RS : Reset input untuk menginisialisasi *device*. Saat aktif low dikenakan pada pin RS untuk sekurangnya lima siklus clock, WE, DEN, dan MEN ditarik high, dan ata bus tidak dikendalikan. Kemudian program counter (PC) dan address bus (A11-A0) di-*clear*. Reset juga men-*disable* interrupt, dan meng-*clear* interrupt flag register.

WE : Write enable bagi *device* saat mengeluarkan data. Saat aktif low, WE menunjukkan bahwa data akan dekeluarkan dari *device* pada data bus. WE hanya aktif selama siklus pertama instruksi OUT dan

siklus kedua instruksi TBLW.

CLKOUT : Output dari clock sistem (seperempat dari frekuensi kristal/CLKIN). *Duty cycle*-nya lima puluh persen.

X1 : Pin output kristal untuk osilator internal. Jika osilator tidak digunakan, pin ini harus dibiarkan tak terhubung.

X2/CLKIN: Input pin ke osilator internal (X2) dari kristal. Juga, sebagai input pin bagi osilator eksternal (CLKIN).

BAB III

DESAIN PERANGKAT KERAS

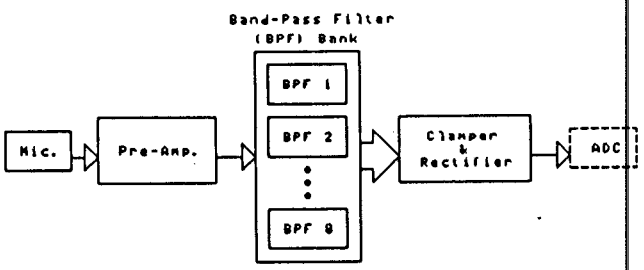
III.1 PENDAHULUAN

Sistem pengenalan suara manusia yang direncanakan ini menggunakan metoda filter bank untuk ekstraksi parameter sinyal suara. Metoda ini relatif lebih sederhana dan lebih mudah direalisasikan daripada metoda yang lainnya. Filter bank adalah deretan beberapa band-pass filter yang mempunyai *center frequency* yang terbagi secara sama dalam skala logaritmik, untuk range frekuensi suara manusia (sekitar 300 Hz hingga 3000 Hz).

Bagian utama dari alat ini adalah tiga buah rangkaian elektronik, yaitu rangkaian elektronik digital , analog, dan rangkaian Analog to Digital Converter (ADC). Rangkaian ADC adalah rangkaian yang mengubah input analog menjadi besaran digital. Rangkaian ADC menggunakan ADC 0820 yang mampu melakukan sampling maksimum sampai frekuensi 400 kHz, sehingga cukup memadai untuk digunakan dalam menangani 16 chanel filter dalam proses sampling sinyal suara. Bagian analog yang utama adalah rangkaian pre-amplifier dan ke-16 band-pass filter tersebut. Sedang rangkaian digital yang utama adalah rangkaian prosesor TMS 32010 beserta komponen-komponen pendukungnya, seperti rangkaian PPI 8255 dan rangkaian RAM data eksternal.

III.1.1 Rangkaian Analog

Rangkaian analog terdiri dari rangkaian pre-amplifier yang berfungsi menguatkan sinyal suara yang berasal dari microphone. Sinyal yang sudah dikuatkan ini diumpankan ke input Filter Bank. Filter Bank terdiri dari 16 Band Pass Filter (BPF). Output dari tiap-tiap BPF ini akan menjadi input untuk rangkaian ADC. Diagram blok dari rangkaian analog dapat dilihat pada gambar 3-1.



Gambar 3-1 Blok Diagram Rangkaian Analog

Masing-masing dari BPF ini memiliki frekuensi cut-off yang saling berdampingan. Centre Frekuensi dari tiap-tiap BPF menaik secara linier dalam skala logaritmik. Dalam rancangan ini digunakan enambelas BPF, sehingga centre frekuensi dan cut-off frekuensi dari masing-masing BPF adalah sebagai berikut:

BPF 1: $f_l = 270\ Hz; f_r = 300\ Hz; f_h = 323\ Hz.$

BPF 2: $f_l = 323\ Hz; f_r = 346\ Hz; f_h = 369\ Hz.$

BPF 3: $f_l = 369\ Hz; f_r = 400\ Hz; f_h = 431\ Hz.$

BPF 4: $f_l = 431\ Hz; f_r = 460\ Hz; f_h = 489\ Hz.$

BPF 5: $f_l = 489\ Hz; f_r = 533\ Hz; f_h = 577\ Hz.$

BPF 6: $f_l = 577\ Hz; f_r = 616\ Hz; f_h = 655\ Hz$

BPF 7: $f_l = 655$ *Hz;* $f_r = 711$ *Hz;* $f_h = 767$ *Hz.*

BPF 8: $f_l = 767$ *Hz;* $f_r = 821$ *Hz;* $f_h = 875$ *Hz.*

BPF 9: $f_l = 875$ *Hz;* $f_r = 948$ *Hz;* $f_h = 1021$ *Hz.*

BPF 10: $f_l = 1021$ *Hz;* $f_r = 1095$ *Hz;* $f_h = 1169$ *Hz.*

BPF 11: $f_l = 1169$ *Hz;* $f_r = 1265$ *Hz;* $f_h = 1361$ *Hz.*

BPF 12: $f_l = 1361$ *Hz;* $f_r = 1460$ *Hz;* $f_h = 1559$ *Hz.*

BPF 13: $f_l = 1559$ *Hz;* $f_r = 1686$ *Hz;* $f_h = 1813$ *Hz.*

BPF 14: $f_l = 1813$ *Hz;* $f_r = 1947$ *Hz;* $f_h = 2081$ *Hz.*

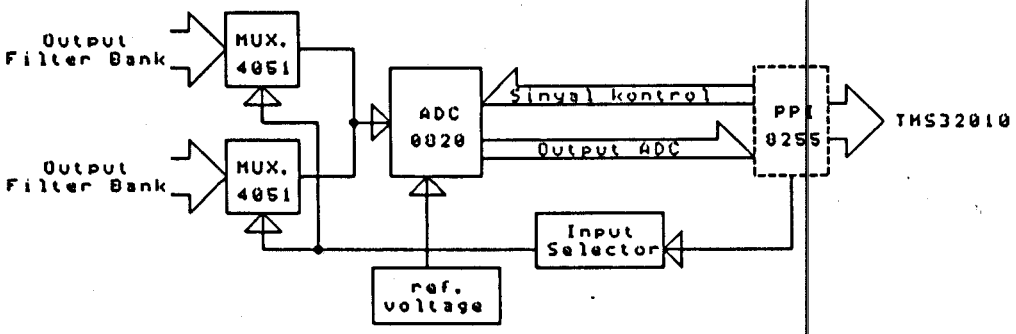
BPF 15: $f_l = 2081$ *Hz;* $f_r = 2250$ *Hz;* $f_h = 2419$ *Hz.*

BPF 16: $f_l = 2419$ *Hz;* $f_r = 2600$ *Hz;* $f_h = 2711$ *Hz.*

III.1.2 Rangkaian ADC.

ADC yang digunakan adalah ADC 0820 yang merupakan *half-flash* ADC. Conversion time dari ADC ini adalah tipikal sekitar 2.5 μs, sehingga cukup cepat untuk menangani sampel data dari enambelas band-pass filter yang digunakan.

Karena ADC 0820 ini hanya mempunyai satu input, maka diperlukan rangkaian multiplekser untuk menangani enambelas input dari enambelas band-pass filter. Pencatuan ADC ini adalah sederhana, karena ia menggunakan catu daya tunggal +5V. Sedang tegangan referensinya diambilkan dari tegangan catu yang +5V tersebut. Pengaturan input channel yang aktif dilakukan dengan bantuan perangkat lunak (*software*).



Gambar 3-2 Blok Diagram Rangkaian ADC

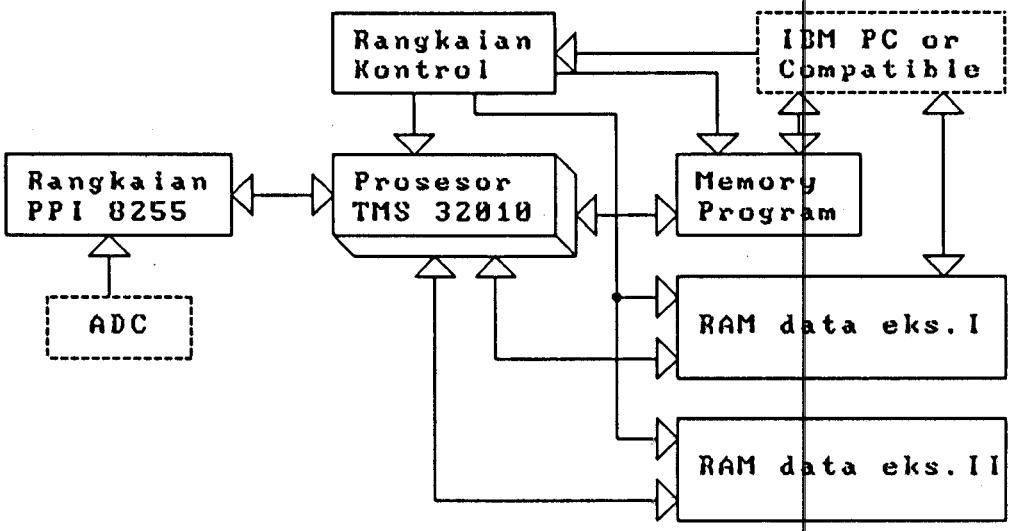
III.1.3 Rangkaian Digital

Rangkaian digital terdiri dari bagian-bagian utama yaitu:

- rangkaian processor TMS 32010.
- rangkaian kontrol buffer.
- rangkaian RAM data eksternal.

Rangkaian processor TMS 32010 adalah inti dari rangkaian digital, yang melakukan kontrol terhadap semua rangkaian digital yang lain. Tetapi secara keseluruhan, proses dikendalikan oleh PC. Rangkaian ini mencakup rangkaian PPI dan rangkaian memory program. Rangkaian PPI digunakan untuk mengakses port untuk input data ke TMS 32010. Data ini berasal dari rangkaian ADC yang menggunakan ADC 0820. Rangkaian memory program berfungsi menyimpan program untuk menjalankan processor TMS 32010.

Instruksi-instruksi program dibuat dalam bahasa Assembly TMS 32010. Program ini diload dari PC ke "memory bersama" PC dengan rangkaian processor



Gambar 3-3 Diagram Blok Rangkaian Digital

TMS 32010 (memory program menggunakan teknik *sharing memory*). Rangkaian RAM data eksternal dihubungkan ke port I/O dari TMS 32010, sehingga besarnya RAM data tidak tergantung dari mapping memory data internal dari TMS 32010. Rangkaian RAM data eksternal digunakan untuk menyimpan data yang diperoleh dari output ADC maupun data yang di-load dari PC. Ada dua rangkaian RAM data eksternal yaitu: RAM data eksternal 1 (RAM 1) dan RAM data eksternal 2 (RAM 2). Sedang rangkaian kontrol buffer digunakan untuk mengatur pergantian saat aktif antara TMS 32010 dan PC, dan mengatur buffer bus ke TMS 32010 atau buffer bus ke PC yang enable. Diagram blok rangkaian digital dapat dilihat pada Gambar 3-3.

III.2 DESAIN RANGKAIAN ANALOG

Rangkaian Analog secara garis besar terdiri dari dua bagian utama, yaitu rangkaian pre-amplifier dan band-pass filter bank. Rangkaian pre-amplifier

berfungsi menguatkan sinyal input yang berasal dari mikropon agar mempunyai level yang cukup besar untuk diumpankan ke band-pass filter. Output dari tiap-tiap band-pass filter tersebut dinaikkan levelnya sampai level positif dan disearahkan untuk memperoleh spektrum amplop sinyal inputnya. Sinyal inilah yang disampling dan dikonversi ke besaran digital dengan ADC 0820 dan akan diolah lebih lanjut dengan prosesor TMS 32010.

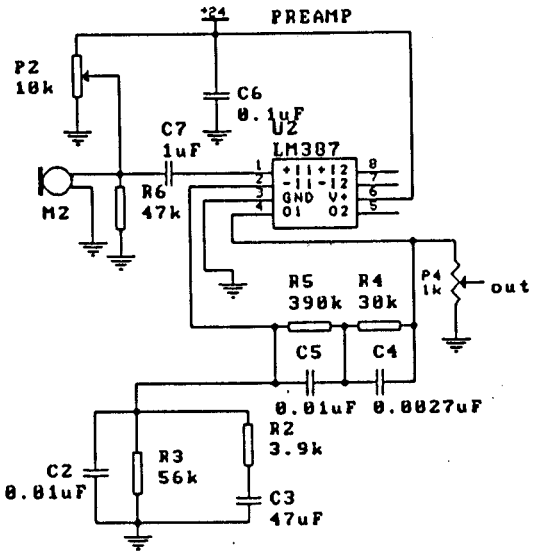
III.2.1 Rangkaian Pre-Amplifier

Rangkaian pre-amplifier yang digunakan ditunjukkan oleh Gambar 3-4. Seperti terlihat pada gambar tersebut, rangkaian ini menggunakan IC LM387 yang merupakan IC khusus untuk pre-amplifier sinyal audio. IC ini mempunyai keunggulan-keunggulan sebagai penguat depan (pre-amplifier). Antara lain, total noise input yang rendah (0,65 μs), open loop gain yang tinggi (104 dB), dan lebar jalur yang besar (sekitar 15 MHz). Besarnya penguatan close loop dapat dihitung dari perbandingan antara $(R_5+R_4)/(R_2||R_3)$. Dari nilai-nilai yang ada maka dapat dihitung bahwa besarnya penguatan adalah:

$$\begin{aligned} Acl &= (R_5+R_4)/(R_2||R_3) \\ &= (390+30)/3.6 \\ &\approx 420 \text{ kali} \end{aligned}$$

III.2.2 Rangkaian Band-Pass Filter Bank

Rangkaian band-pass filter bank terdiri dari enambelas buah band-pass filter. Struktur dari tiap-tiap filter adalah sama, yaitu filter Butterworth. Tiap-tiap



Gambar 3-4 Rangkaian Pre-Amplifier

band-pass filter tersebut mempunyai *centre frequency* yang tidak sama, berkisar antara 300 Hz hingga 3000 Hz. Bentuk rangkaian dari tiap-tiap filter adalah seperti ditunjukkan oleh Gambar 3-5. Besarnya centre frequency (f_r), frekuensi cut-off bawah (f_l), maupun frekuensi cut-off atas (f_h) dapat dihitung sebagai berikut:

$$Ar = R_2/2R_4$$

$$Q = \tfrac{1}{2}\sqrt{(R_4/R_3 + 2Ar)}$$

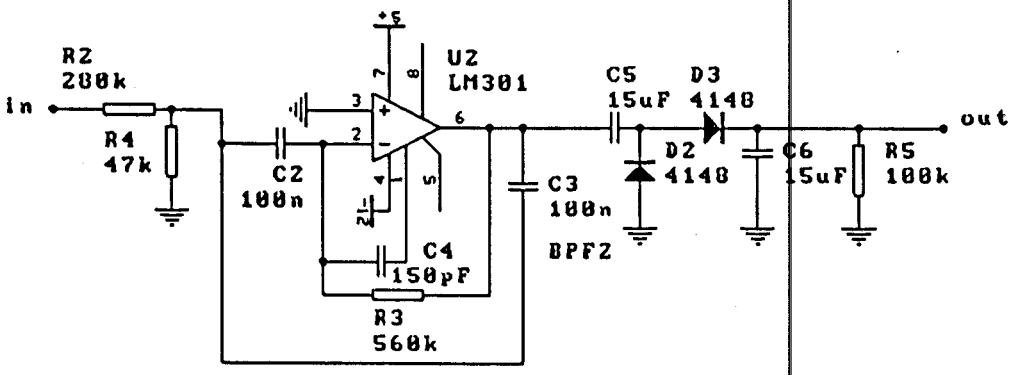
$$B = 2/R_4C$$

$$Wr = QB$$

$$Wh = Wr + B/2$$

$$Wl = Wr - B/2$$

sehingga untuk nilai-nilai yang ada pada Gambar 3-5, maka besarnya frekuensi center, frekuensi *cut-off* atas, dan frekuensi



Gambar 5-5 Rangkaian Band-Pass Filter

cut-off bawah adalah:

$$Wr = QB$$

$$\approx 2273{,}36 \text{ rad/s}$$

$$Wh = Wr + B/2$$

$$\approx 3604{,}72 \text{ rad/s}$$

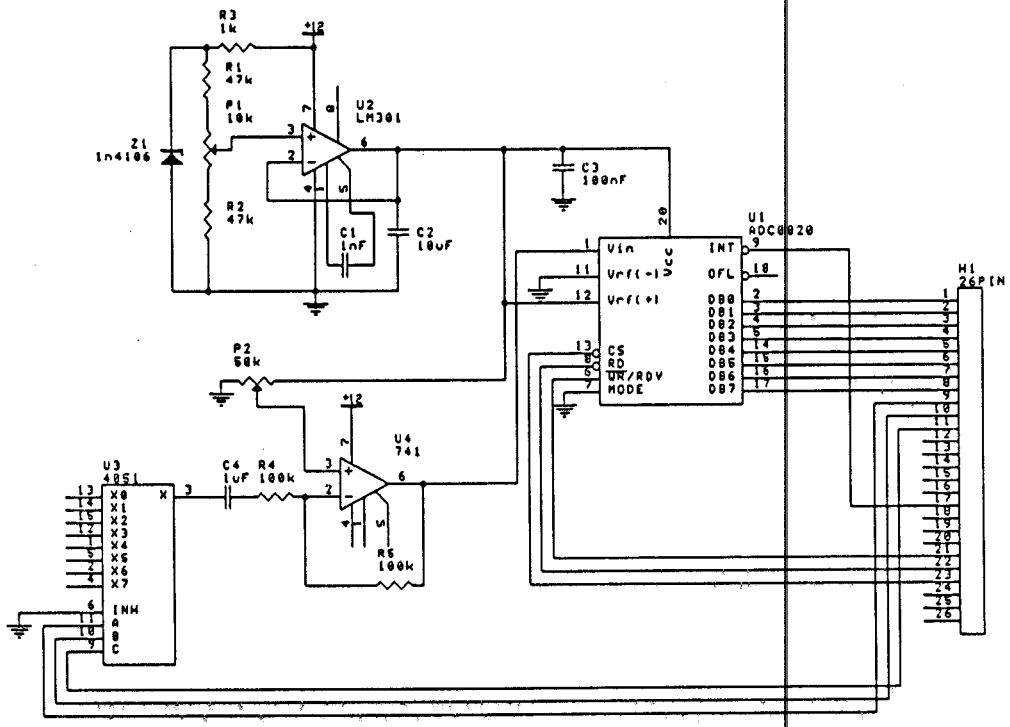
$$Wl = Wr - B/2$$

$$\approx 942{,}06 \text{ rad/s}$$

Untuk mendapatkan harga-harga frekuensi center, frekuensi *cut-off* bawah dan frekuensi *cut-off* atas untuk filter-filter yang lain, digunakan formula yang sama dengan di atas.

III.3 DESAIN RANGKAIAN ADC

Rangkaian Analog-to-Digital Converter menggunakan IC ADC 0820. ADC ini merupakan *half-flash* ADC, sehingga kecepatan konversinya tergolong tinggi. Waktu konversi tipikal sekitar 2,5 μs. Seperti telah disebutkan di depan, bahwa ADC ini harus menangani enambelas input dari enambelas band-pass filter.



Gambar 3-6 Rangkaian ADC 0820

Karena pada ADC ini hanya ada satu input, maka digunakan multiplekser 4051 untuk keperluan ini. Rangkaian selengkapnya dari AD Converter dapat dilihat pada Gambar 3-6. Seperti terlihat pada gambar tersebut, rangkaian ADC ini adalah sederhana. Catu daya yang digunakan adalah catu daya tunggal +5V.

Sedang referensinya +5V juga diambilkan dari catu dayanya. Rangkaian catu daya menggunakan komponen utama IC LM 301 dan dioda zener 1N4106 untuk mendapatkan tegangan yang relatif stabil pada harga sekitar +5V. Konektor H1 dihubungkan ke rangkaian PPI 8255 yang ada di rangkaian prosesor TMS 32010. Data bus DB0 - DB7 dihubungkan ke port A PPI 8255. Input address 4051 dihubungkan ke port B. Sedang sinyal kontrol ke ADC 0820 diberikan lewat port C. Cara kerja rangkaian adalah sebagai berikut:

- Input yang aktif dipilih dengan cara mengatur sinyal yang diberikan ke input address 4051 (pin 9, 10, 11). Jika A="0", B="0", C="0", maka input x0 yang akan diteruskan ke output. Jika A="1", B="0", C="0", maka input x1 yang akan diteruskan ke output. Begitu seterusnya. Karena ada enambelas input yang harus ditangani, maka digunakan dua buah 4051. Pengaturan 4051 yang aktif dilakukan dengan mengatur input pin "inhibit" (pin 6) pada 4051.

- ADC 0820 akan mulai melakukan konversi jika diberi sinyal "start of conversion", yaitu sinyal "low" ke pin CS, RD dan RDY. Hasil konversi akan valid jika output pada pin 9 (INT) berubah dari "high" ke "low". Untuk melakukan konversi lagi, maka ADC harus direset dengan cara memberikan sinyal "low" ke pin CS dan sinyal "high" ke pin RD dan RDY.

Seperti terlihat pada gambar tersebut, pin 7 (MODE) dihubungkan ke ground. Ini menunjukkan bahwa ADC dijalankan dengan mode operasi "RD

mode".

III.4 DESAIN RANGKAIAN DIGITAL

Rangkaian digital mempunyai bagian-bagian utama:

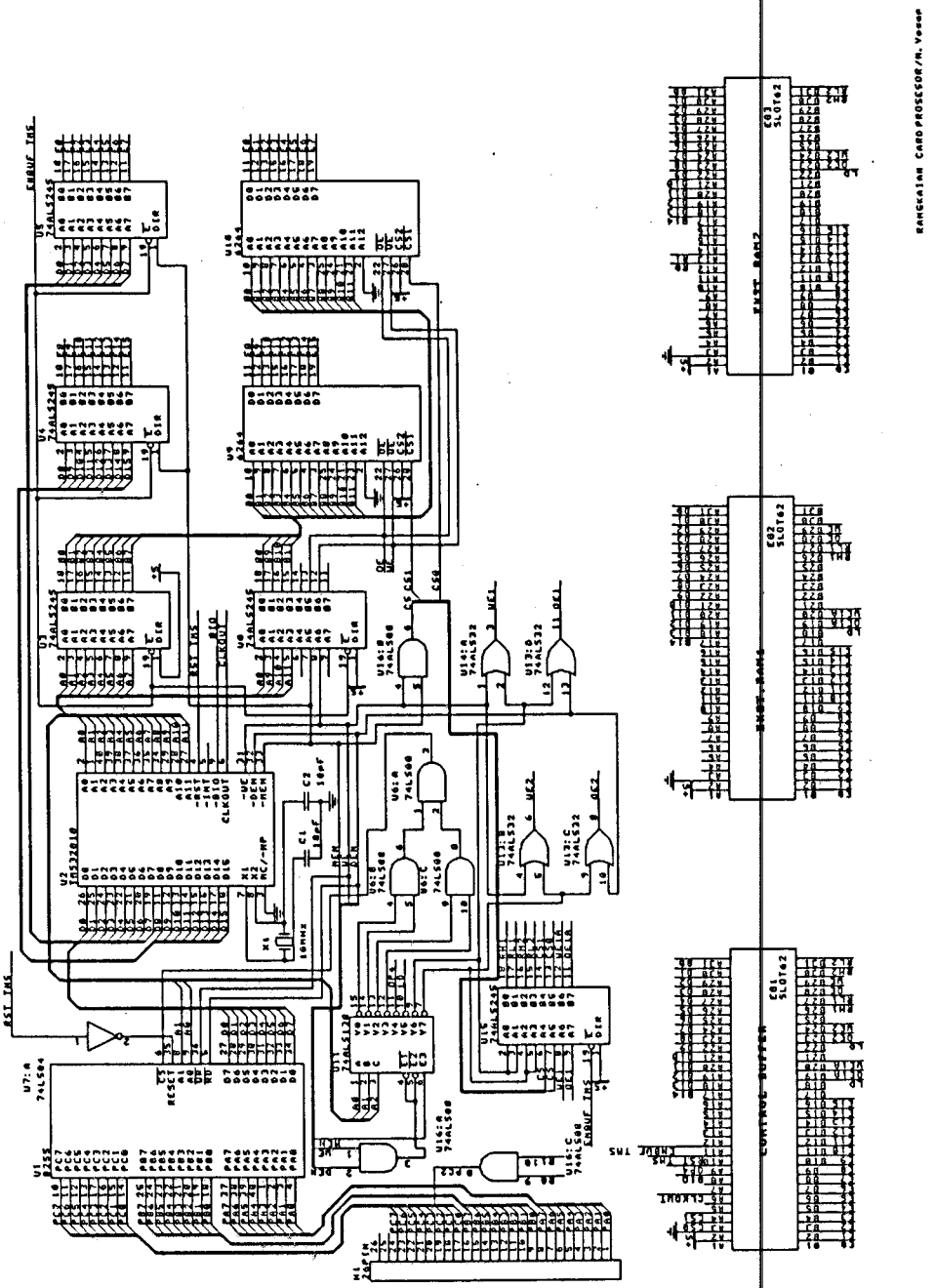
- Rangkaian prosesor TMS 32010.
- Rangkaian kontrol buffer.
- Rangkaian RAM data eksternal.

Berikut akan dijabarkan lebih lanjut dari tiap-tiap rangkaian tersebut.

III.4.1 Rangkaian Prosesor TMS 32010

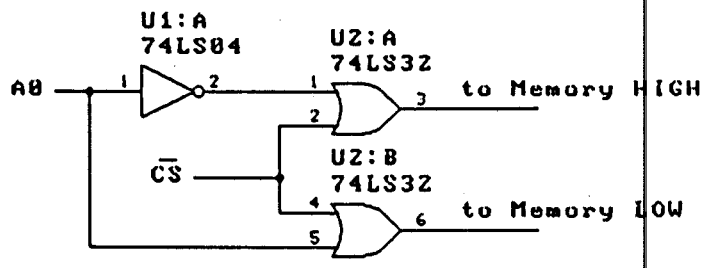
Rangkaian prosesor TMS 32010 mencakup rangkaian PPI dan rangkaian memory program. Rangkaian PPI berfungsi untuk mengambil sinyal input dari ADC. Sedang rangkaian memory program digunakan untuk menyimpan instruksi-instruksi program yang di-load dari PC untuk menjalankan prosesor TMS 32010. Instruksi-instruksi program dibuat dalam bahasa Assembly TMS 32010. Rangkaian selengkapnya dari rangkaian prosesor TMS 32010 ditunjukkan pada Gambar 3-7. Pada gambar tersebut dapat kita lihat bahwa:

- PPI 8255 ditempatkan pada port I/O dengan alamat port 0 hingga port 3. PPI 8255 ini diakses oleh prosesor TMS 32010 dengan menjadikan -DEN atau -WE berlevel "LOW". Level "LOW" pada -WE untuk operasi penulisan, misalnya penulisan control word ke PPI 8255. Sedang level "LOW" pada -DEN untuk operasi pembacaan port A, port B, atau port



Gambar 3-7 Rangkaian Prosesor TMS32010

C dari PPI 8255.

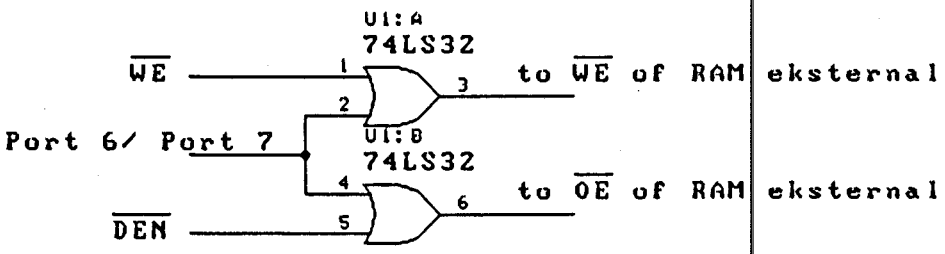


Gambar 3-8 Rangkaian Pemilih memory HIGH dan LOW

- Memory program eksternal sebesar 4 kWord (U9 & U10). Memory program ditempatkan pada alamat D000:0000 - D000:1FFF pada mapping memory IBM PC. Karena lebar data bus dari IBM PC adalah 8 bit, maka loading program ke memory program TMS 32010 yang sebesar 16 bit dilakukan dua kali untuk HIGH byte dan LOW byte. Pembedaan antara LOW byte dan HIGH byte dilakukan dengan bantuan bit A0 pada rangkaian kontrol buffer seperti ditunjukkan oleh Gambar 3-8. Jika bit A0="0", maka akses dilakukan ke LOW byte memory. Jika bit A0="1", maka akses dilakukan ke HIGH byte memory. Akses ke memory program dilakukan dengan menjadikan pin -WE atau -MEN berlevel "LOW". Jika -WE yang "LOW", maka operasinya adalah penulisan memory. Dan jika -MEN yang "LOW", maka operasinya adalah pembacaan memory.

Tabel 4-1 Pengaturan High Byte atau Low Byte yang Aktif

Address Bit												Hi	Lo
A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	A0
0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	1	0
.													
.													
.													
1	1	1	1	1	1	1	1	1	1	1	1	1	0

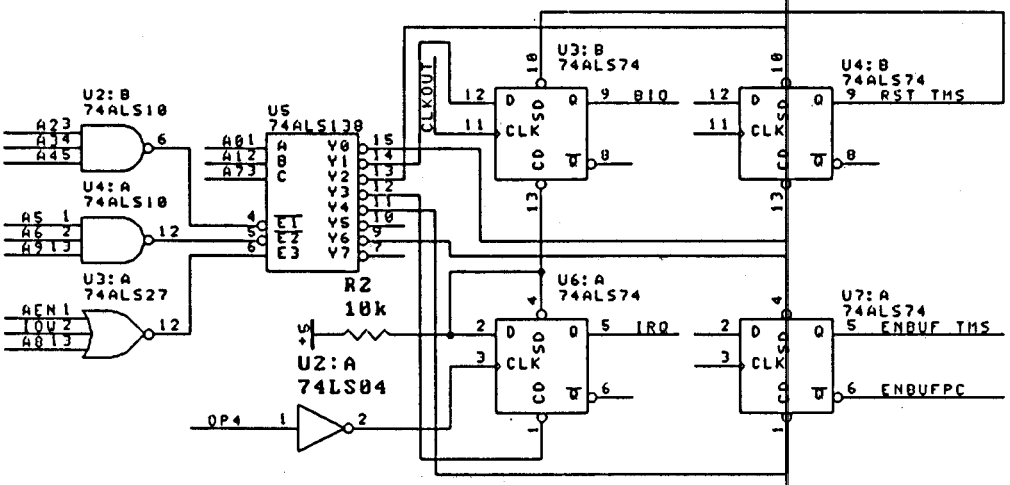


Gambar 3-9 Rangkaian untuk Akses RAM eksternal

Akses ke RAM data eksternal dilakukan dengan mengaktifkan port 6 atau port 7. Untuk memilih RAM data eksternal 1 maka port 6 yang aktif. Sedang untuk memilih RAM data eksternal 2 maka port 7 yang aktif. Port yang aktif ini juga dihubungkan ke rangkaian OR gate seperti ditunjukkan oleh Gambar 3-9, untuk memastikan ke RAM data eksternal yang mana operasi pembacaan atau penulisan dilakukan.

III.4.2 Rangkaian Kontrol Buffer

Rangkaian kontrol buffer berfungsi untuk mengatur pergantian saat aktif antara prosesor TMS 32010 dengan IBM PC, dan mengatur "enable" buffer bus dari memory ke TMS 32010 dan buffer bus dari memory ke PC. Jika TMS 32010 aktif, maka buffer bus dari TMS 32010 ke memory akan enable. Sebaliknya jika IBM PC yang aktif, maka buffer bus dari IBM PC ke memory (rangkaian digital) yang enable.



Gambar 3-10 Rangkaian Kontrol Buffer

Rangkaian kontrol buffer juga digunakan untuk mengatur pensinyalan BIO, disable IRQ, dan RESET TMS 32010. Pengaturan akses ke program memory maupun ke RAM data eksternal I, juga dilakukan dengan bantuan rangkaian ini. Seperti dapat dilihat pada Gambar 3-10, sinyal BIO diaktifkan dengan cara memberikan instruksi output ke port 27DH. Pengaktifan TMS 32010 dilakukan

dengan instruksi output ke port 27EH. Reset TMS, output ke port 27CH. Enable buffer PC, output ke port 2FCH, dan enable buffer TMS, dilakukan dengan memberikan instruksi output ke port 2FEH.

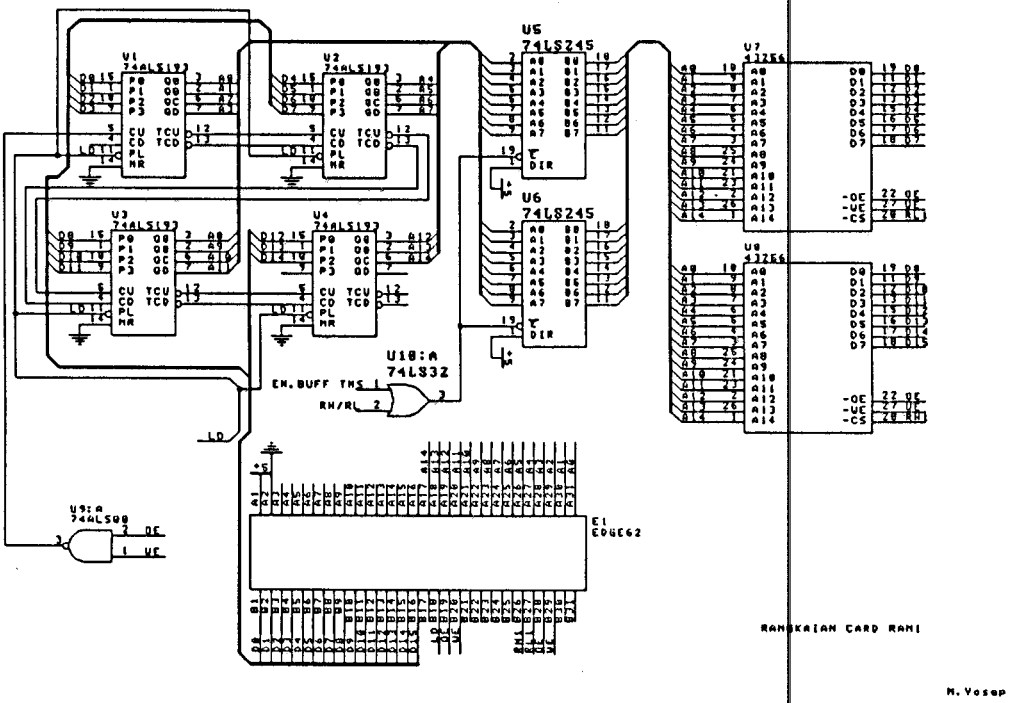
Memori program eksternal menempati alamat D000:0000H hingga D000:1FFFH. Dan RAM data eksternal I menempati alamat A000:0000H hingga A000:FFFFH. RAM data eksternal digunakan untuk menyimpan data input suara dari ADC.

III.4.3 Rangkaian RAM data eksternal

Rangkaian RAM data eksternal adalah rangkaian memori untuk menampung data hasil sampling dari ADC 0820 (RAM data II), dan untuk menampung data yang di-load dari IBM PC (RAM data I). Data yang di-load dari PC adalah data referensi yang sebelumnya tersimpan dalam bentuk file di disket.

Rangkaian RAM data eksternal ini ditempatkan di I/O port TMS 32010, sehingga kapasitasnya tidak tergantung kapasitas memori internal dari TMS 32010.

Akses ke RAM data eksternal ini dilakukan dengan instruksi IN dan/atau OUT. Setiap kali ada instruksi IN atau OUT, maka rangkaian counter yang terdiri dari empat buah IC 74LS193, akan menaikkan alamat yang menunjuk ke RAM data eksternal tersebut. Berarti akses ke RAM data eksternal, baik pembacaan maupun penulisan hanya dapat dilakukan dalam urutan menaik. Rangkaian selengkapnya ditunjukkan oleh Gambar 3-11 pada halaman berikut.



Gambar 3-11 Rangkaian RAM data eksternal

BAB IV

DESAIN PERANGKAT LUNAK

IV.1 PENDAHULUAN

Setelah sinyal suara diambil dengan mikropon, sinyal tersebut dikuatkan oleh pre-amplifier. Output dari pre-amplifier diumpankan ke Band-Pass Filter (BPF) bank. Tiap-tiap BPF melewatkan frekuensi tertentu yang sesuai dengan spesifikasinya.

Output dari BPF-BPF ini disearahkan dan diambil nilai rms-nya. Nilai-nilai rms ini kemudian diinputkan ke rangkaian ADC untuk diubah menjadi sinyal digital. Sinyal digital ini menjadi input untuk prosesor TMS32010 dan diolah lebih lanjut.

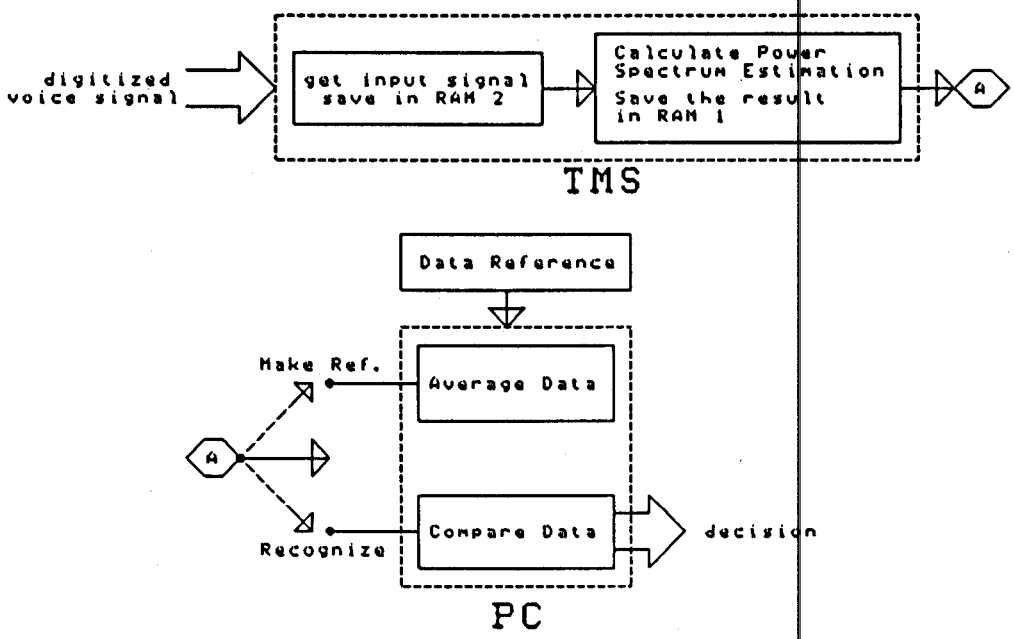
Proses secara keseluruhan dikendalikan oleh PC. Loading program ke memory TMS32010 juga dilakukan lewat PC. Perangkat lunak untuk mengatur operasi IBM PC ditulis dengan bahasa pemrograman C. Sedang perangkat lunak untuk menjalankan prosesor TMS32010 ditulis dalam bahasa Assembly TMS32010.

Proses secara garis besar terbagi dalam dua bagian utama, yaitu proses pembuatan data referensi dan proses pembandingan data antara data input dengan data referensi. Diagram blok proses secara keseluruhan dapat dilihat pada Gambar 4-1.

IV.2 PEMBUATAN DATA REFERENCE

Data reference digunakan sebagai data pembanding untuk sinyal input yang diterima oleh sistem.

Sebagai data pembanding, data reference ini adalah data yang telah



Gambar 4-1 Diagram Blok Perangkat Lunak

disimpan sebelumnya sebagai hasil rata-rata dari pengambilan beberapa kali spektrum amplop sinyal suara. Data yang disimpan sebagai reference adalah data hasil perhitungan spektrum daya sinyal input. Untuk menghemat memori komputer, maka data reference ini disimpan di disket. Tetapi sebagai akibatnya, respon sistem jadi lebih lambat. Algoritma proses pembuatan data reference adalah sebagai berikut:

- Loading PROGRAM UNTUK EKSTRAKSI PARAMETER SPEKTRUM DAYA ke

memory TMS32010. Program yang diload adalah program yang ditulis dalam bahasa assembly TMS32010. Sebelum diload, file program tersebut diubah dulu ke bentuk binary. Program diload ke alamat D000:0000 - D000:1FFF. Pada waktu loading program prosesor TMS32010 dalam keadaan tidak aktif.

- Disable buffer data bus dari PC ke memory TMS32010. Aktifkan prosesor TMS32010 dengan cara memberi sinyal HIGH ke pin 4 (pin RESET). Sebelum TMS32010 diaktifkan, buffer data bus dari TMS32010 ke memory program dibuat enable terlebih dulu.

- Ambil sinyal noise lingkungan lima frame (tiap frame terdiri dari 100 data). Rata-ratakan dan simpan hasilnya di variabel NOISE. Tentukan harga minimum (ENMIN) dan maksimum (ENHIGH) sinyal suara. ENMIN berharga dua kali noise. Sedang ENHIGH berharga sekitar sepuluh kali noise.

- Tunggu sinyal clock. Jika sinyal clock sudah diberikan tunggu beberapa saat untuk memberi waktu pada saat mulai pengucapan kata.

- Ambil input sinyal suara sebanyak 24K Word. Simpan di RAM data eksternal II.

- Jumlahkan setiap 100 kuadrat-data (untuk mendapatkan short-time energinya) dari RAM data eksternal II sebanyak 24K Word. Jika data yang diperoleh valid (energi sinyal suara berharga di antara ENMIN dan ENHIGH), maka data tersebut diproses lebih lanjut (hitung estimasi spektrum dayanya). Jika data yang baru diperoleh tidak valid (berharga terlalu rendah atau terlalu tinggi) set status (STAT) bahwa data tidak valid.

- Strobe PC. Reset TMS32010. Jika data valid, ambil data dari RAM data eksternal I, jika data lama telah ada, rata-ratakan data baru dengan data lama, simpan ke disket. Jika data tidak valid, abaikan data. Tunggu perintah selanjutnya.

IV.3 PEMBANDINGAN DATA

Proses pengenalan kata, pada dasarnya adalah proses pembandingan data antara data input dengan data reference. Algoritma proses pengenalan kata adalah sebagai berikut:

- Loading PROGRAM UNTUK EKSTRAKSI PARAMETER SPEKTRUM DAYA ke memory TMS32010. Program yang diload adalah program yang ditulis dalam bahasa assembly TMS32010. Sebelum diload, file program tersebut diubah dulu ke bentuk binary. Program diload ke alamat D000:0000 - D000:1FFF. Pada waktu loading program prosesor TMS32010 dalam keadaan tidak aktif.

- Disable buffer data bus dari PC ke memory TMS32010. Aktifkan prosesor TMS32010 dengan cara memberi sinyal HIGH ke pin 4 (pin RESET). Sebelum TMS32010 diaktifkan, buffer data bus dari TMS32010 ke memory program dibuat enable terlebih dulu.

- Ambil sinyal noise lingkungan sebanyak lima frame (tiap frame terdiri dari 100 data). Rata-ratakan dan simpan hasilnya di varibel NOISE. Tentukan harga minimum (ENMIN) dan maksimum (ENHIGH) sinyal suara. ENMIN berharga dua kali noise. Sedang ENHIGH berharga sekitar sepuluh kali noise.

- Tunggu sinyal clock. Jika sinyal clock sudah diberikan tunggu beberapa saat untuk memberi waktu pada saat mulai pengucapan kata.

- Ambil input sinyal suara sebanyak 24K Word. Simpan di RAM data eksternal II.

- Jumlahkan setiap 100 kuadrat-data (untuk mendapatkan short-time energinya) dari RAM data eksternal II sebanyak 24K Word. Jika data yang diperoleh valid (energi sinyal suara berharga di antara ENMIN dan ENHIGH), maka data akan diproses lebih lanjut. Jika data tidak valid set status (STAT) bahwa data tidak valid.

- Strobe PC. Jika data valid PC akan melakukan proses pembandingan data antara data input dengan data referrence dan menghitung perbedaan jaraknya dengan menggunakan metoda pemrograman dinamik. Hasil tersebut akan dibandingkan dengan level threshold tertentu yang sudah diperoleh sebelumnya. Jika hasilnya berharga dibawah harga level maka input dianggap "match" dengan data referensi dan dianggap sebagai kata yang telah disimpan sebagai referensi tersebut.

- Jika data input tidak "match" dengan data referensi, load data referensi berikutnya dan lakukan lagi proses pembandingan seperti di atas. Jika sampai data referensi yang terakhir tidak ada data yang "match", maka dianggap kata tersebut tidak dikenali.

IV.4 Algoritma Program untuk Ekstraksi Parameter Spektrum Daya

Program untuk ekstraksi parameter spektrum daya dari sinyal suara adalah program yang dijalankan di rangkaian prosesor TMS32010. Karena itu program ini ditulis dalam bahasa assembly TMS32010. Dan setelah di-*compile*, program ini diload ke memory TMS32010. Algoritma program untuk ekstraksi parameter spektrum daya dari sinyal suara adalah sebagai berikut:

- Ambil sinyal noise lingkungan dan hitung nilai rata-ratanya. Hitung energi minimum (ENMIN) dan energi maksimum (ENHIGH) sinyal input suara yang diijinkan. ENMIN berharga dua kali noise dan ENHIGH berharga sekitar sepuluh kali noise.

- Ambil sinyal input suara sebanyak 24 KWord. Tentukan level energi sinyal suara. Jika energi sinyal berada di antara ENMIN dan ENHIGH, maka akan dilakukan proses lebih lanjut. Jika tidak, set status bahwa data tidak valid hentikan proses dan tunggu perintah selanjutnya.

- Tentukan harga maksimum dan minimum sinyal input yang diterima. Lakukan proses normalisasi pada data. Proses normalisasi ini bertujuan membuat data berharga di antara 0 dan 1 agar tidak terjadi overflow pada data jika data dikuadratkan untuk memperoleh spektrum dayanya.

- Kuadratkan data hasil normalisasi. Rata-ratakan hasil kuadrat data untuk beberapa frame spektrum frekuensi. Setiap frame terdiri dari 16 point spektrum frekuensi. Ke-16 point ini diperoleh dari 16 channel Band-Pass Filter. Hasil rata-rata kuadrat data ini ekivalen dengan estimasi spektrum daya sinyal suara. Hasil inilah yang akan digunakan sebagai data untuk data referensi dalam proses pembuatan data referensi, dan data input dalam proses pembandingan data.

IV.5 Algoritma Proses Pengenalan Suara

Bagian inti dari proses pengenalan suara adalah algoritma pemrograman dinamik. Algoritma ini bertujuan untuk mencari jarak minimum antara dua barisan sinyal, untuk menentukan kecocokan antara ke-dua barisan tersebut. Sinyal yang paling cocok akan memiliki jarak yang mimimum dibandingkan

dengan sinyal lainnya.

Program ini dijalankan di PC dan ditulis dalam bahasa pemrograman C. Algoritma proses pengenalan suara adalah sebagai berikut:

- Load data referensi ke memory PC, konversikan data dari format Q14 (TMS32010) ke format desimal (PC).

- Ambil data input dari lokasi memory A000:0000 (lokasi memory bersama), konversikan data dari format Q14 (TMS32010) ke format desimal (PC).

- Hitung perbedaan jarak antara data input dengan data referensi dengan algoritma pemrograman dinamik. Bandingkan hasilnya dengan level threshold tertentu yang ditentukan sebelumnya. Jika harganya dibawah harga level maka input tersebut dianggap "match" dengan data referensi dan dianggap sebagai kata yang telah disimpan sebagai referensi tersebut. Jika tidak, load data referensi berikutnya dan lakukan lagi proses pembandingan data seperti di atas. Jika sampai data referensi yang terakhir tidak ada data yang "match", maka dianggap sistem tidak mengenali input kata tersebut. Tetapi bisa juga terjadi, sistem mengenali secara salah input kata yang diterima, karena hasil perhitungan jaraknya lebih kecil dari level threshold yang diijinkan untuk data referensi tersebut.

BAB V

PENGUJIAN SISTEM

Sistem pengenalan suara ini terdiri dari beberapa bagian penting yang membentuk sistem, yaitu Band-Pass Filter Bank, *Analog to Digital Converter* (ADC) dan rangkaian Prosesor TMS 32010 serta Perangkat lunak pengendali sistem. Pengujian Sistem meliputi pengujian bagian-bagian sistem yang penting tersebut, yaitu rangkaian filter, rangkaian ADC dan rangkaian prosesor TMS 32010. Pengujian perangkat lunak tercakup dalam pengujian terhadap performansi sistem secara keseluruhan.

V.1 Pengujian Rangkaian Filter

Rangkaian filter yang dibuat adalah *Butterworth Band-Pass Filter.* Pengujian yang dilakukan meliputi *center frequency*, frekuensi *cut-off* bawah, dan frekuensi *cut-off* atas. Dari pengujian yang dilakukan terhadap ke-16 Band-Pass Filter yang telah dibuat, didapat hasil sebagai berikut:

Tabel 5-1 Pengujian Band Pass Filter

No.	Filter	fl(Hz)	fc(Hz)	fh(Hz)
1	BPF 1: Perhitungan Pengujian	 270 241	 300 270	 323 294

2	BPF 2: Perhitungan Pengujian	 323 305	 346 324	 369 364
3	BPF 3: Perhitungan Pengujian	 369 351	 400 404	 431 427
4	BPF 4: Perhitungan Pengujian	 431 375	 460 424	 489 460
5	BPF 5: Perhitungan Pengujian	 489 450	 533 518	 577 578
6	BPF 6: Perhitungan Pengujian	 577 547	 616 620	 655 645
7	BPF 7: Perhitungan Pengujian	 655 600	 711 695	 767 780
8	BPF 8: Perhitungan Pengujian	 767 748	 821 820	 875 880
9	BPF 9: Perhitungan Pengujian	 875 854	 948 955	 1021 1055
10	BPF 10: Perhitungan Pengujian	 1021 900	 1095 1050	 1169 1145
11	BPF 11: Perhitungan Pengujian	 1169 1010	 1265 1100	 1361 1220
12	BPF 12: Perhitungan Pengujian	 1361 1110	 1460 1200	 1559 1280

13	BPF 13: Perhitungan Pengujian		1559 1305	1686 1410	1813 1620
14	BPF 14: Perhitungan Pengujian		1813 1410	1947 1506	2081 1690
15	BPF 15: Perhitungan Pengujian		2081 1565	2250 1680	2419 1850
16	BPF 16: Perhitungan Pengujian		2419 1600	2600 1900	2711 2340

Dari hasil yang diperoleh pada tabel 5-1, dapat diketahui bahwa Band-Pass Filter - Band-Pass Filter yang telah dirancang memenuhi spesifikasi yang diinginkan dan dapat digunakan untuk pre-prosesing sinyal suara.

V.2 Pengujian Rangkaian ADC.

Rangkaian ADC yang dibuat menggunakan ADC 0820 yang merupakan *half-flash* ADC, sehingga kecepatan konversinya tergolong tinggi. Pengujian rangkaian ADC meliputi pengujian sinyal-sinyal kontrol dan pengujian hasil output ADC. Untuk pengujian output ADC diambil beberapa sampel saja yang dianggap mewakili output ADC secara keseluruhan. Sinyal-sinyal kontrol yang penting yang terdapat pada ADC 0820 adalah sinyal RDY, RD, INT, dan CS.

Pengujian rangkaian ADC dilakukan sebagai berikut:

- Sebelumnya telah diketahui bahwa ADC 0820 beroperasi dalam mode "RD

mode". Dalam mode ini akhir konversi ditandai dengan sinyal pada pin INT berubah dari logika HIGH ke logika LOW. Untuk memulai konversi diberikan sinyal berlogika LOW ke pin RDY dan CS. Setelah pin INT berubah dari HIGH ke LOW dibaca hasil konversi. Setelah dilakukan beberapa kali percobaan, diketahui bahwa ADC dapat bekerja dengan baik dalam mode ini.

Hasil beberapa kali percobaan untuk beberapa nilai input ditunjukkan dalam Tabel 5-2.

Tabel 5-2 Pengujian output ADC 0820

No.	Tegangan input (Volt)	Hasil output ADC (Heksa)
1	1.0 V	33
2	1.5 V	4C
3	2.0 V	66
4	2.5 V	7F
5	3.0 V	99
6	3.5 V	B2
7	4.0 V	CC
8	4.5 V	E5
9	5.0 V	FF

Dari hasil yang diperoleh pada tabel 5-2, dapat diketahui bahwa rangkaian ADC dapat berfungsi dengan baik dan dapat digunakan untuk mengkonversi sinyal suara ke besaran digital.

V.3 Pengujian Rangkaian Prosesor TMS32010

Pengujian rangkaian prosesor TMS32010 meliputi kerja rangkaian secara keseluruhan beserta komponen-komponen pendukungnya, seperti rangkaian PPI8255 dan rangkaian eksternal RAM.

Pengujian dilakukan sebagai berikut:

- Dibuat program untuk menyimpan data di eksternal RAM.

  Kemudian data tersebut diambil dan di-output-kan ke PPI 8255. Ternyata hasil output yang ditampilkan oleh peraga LED menunjukkan bahwa data yang telah disimpan di eksternal RAM tidak berubah. Ini berarti rangkaian prosesor TMS32010, rangkaian eksternal RAM dan rangkaian PPI 8255 dapat bekerja dengan baik.

V.4 Pengujian Performansi Sistem

Pengujian yang dilakukan terhadap sistem meliputi pengujian kecepatan respon sistem dan keakuratan kerja sistem. Dalam tugas akhir ini sistem akan diuji dengan tiga contoh kata, yaitu kata "satu", "dua", dan "tiga". Untuk tiap-tiap kata itu akan diuji kecepatan sistem mengenali kata tersebut dan seberapa tepat sistem mengenalinya. Hasil yang didapat adalah hasil rata-rata dari beberapa kali percobaan. Setelah dilakukan beberapa kali percobaan untuk mendapatkan ketepatan respon sistem didapatkan hasil (nilai rata-rata) sebesar 66,7%. Angka ini menunjukkan prosentase kebenaran tanggapan respon sistem.

Tabel 5-3 Pengujian Kecepatan Tanggapan Sistem

No\Kata	"Satu" (detik)	"Dua" (detik)	"Tiga" (detik)
1	2,38	3,82	8,63
2	1,91	11,64	10,47
3	1,94	5,66	6,68
4	1,91	11,75	8,59
5	11,06	5,65	6,65
6	1,75	11,63	6,73
7	1,72	11,65	8,65
8	1,88	11,69	8,69
9	1,94	11,65	8,73
10	1,50	5,75	8,73
rata²	2,79	9,09	8,85

Dari hasil pengujian terhadap sistem pengenal suara ini, dapat dilihat bahwa sistem dapat bekerja meskipun hasilnya belum optimal. Waktu pengenalan masih lama (tidak *real-time*) dan akurasi yang masih belum begitu baik. Hasil ini dapat diperbaiki dengan cara memperbaiki metoda untuk pre-prosesing sinyal dan metoda untuk pembandingan data dan pengambilan keputusan.

BAB VI

PENUTUP

VI.1 KESIMPULAN

1. Realisasi sistem untuk komunikasi antara manusia dan mesin (komputer) adalah suatu hal yang mungkin, walaupun pada tugas akhir ini sistem yang dimaksud baru pada tahap mampu memberikan respon terhadap input suara manusia. Hasil yang lebih optimal dapat dicapai dengan menggunakan metoda yang lebih baik, baik pada proses ekstraksi parameter sinyal maupun proses pengenalan ciri sinyal.
2. Penggunaan mikroprosesor TMS 32010 dalam realisasi sistem pengenal suara ini sangat membantu mempercepat proses pengenalan, dalam hal ini proses untuk ekstraksi parameter sinyal suara.
3. Sistem Pengenalan Suara secara umum dibagi menjadi tiga bagian, yaitu bagian ekstraksi parameter sinyal suara, bagian pembanding data input dengan data referensi, dan bagian pengambil keputusan. Tiap-tiap bagian memegang peran yang penting, saling mempengaruhi dan berpengaruh pada proses pengenalan secara keseluruhan.
4. Metoda Band-Pass Filter Bank ternyata cukup memadai untuk digunakan dalam ekstraksi parameter sinyal suara. Namun informasi yang diperoleh dari sini adalah sedikit, sehingga ini membatasi kemampuan dari sistem pengenal suara.

5. Metoda pemrograman dinamik memberikan hasil yang cukup memuaskan dalam penggunaan untuk pembandingan data dalam proses pengenalan suara.

VI.2 SARAN

1. Untuk ekstraksi parameter sinyal yang lebih baik, dapat digunakan metode-metode yang lain seperti FFT atau *Linear predictive coding*. Karena dengan menggunakan metode-metode ini dapat diperoleh informasi yang lebih banyak, sehingga dapat meningkatkan kemampuan dari sistem pengenal suara.
2. Untuk proses pengenalan suara, untuk hasil yang lebih baik dapat digunakan metoda yang lebih canggih seperti metoda jaringan syaraf tiruan. Karena dengan metoda jaringan syaraf tiruan misalnya, sistem dapat lebih menoleransi variasi input yang lebih besar dan lebih banyak.

DAFTAR PUSTAKA

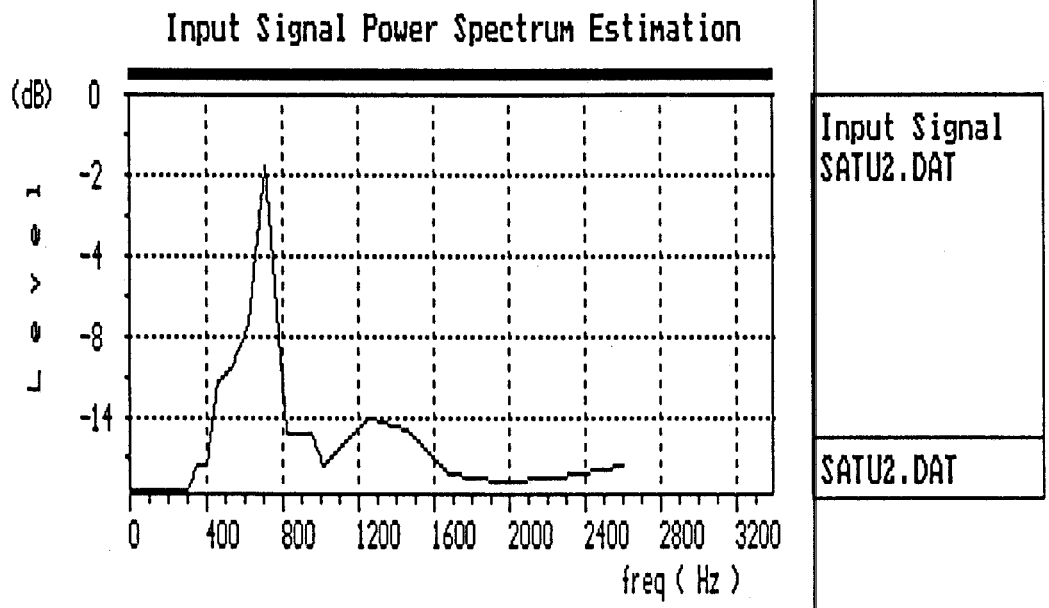
1. Rabiner, Lawrence R., Bernard Gold, Theory and Application of Digital Signal Processing, Prentice Hall of India, New Delhi, 1990.
2. ---, First-Generation TMS320 User's Guide, Texas Instruments Inc., 1988.
3. Lin, Kun-Shan, Ph.D., Digital Signal Processing Applications with the TMS320 Family: Theory, Algorithms, and Implementations, Texas Instruments Inc., 1986.
4. Furui, Sadaoki, Digital Speech, Processing, Synthesis, and Recognition, Marcel Dekker Inc., New York, 1989.
5. Coughlin, Robert F., Frederick F. Driscoll, & Herman Widodo Soemitro, Penguat Operasional dan Rangkaian Terpadu Linier, Edisi Kedua, Penerbit Erlangga, Jakarta, 1985.
6. Schildt, Herbert, C Power User's Guide, ---.
7. Sjartuni, Ananta, Lebih Lanjut dengan C, Elex Media Komputindo, Jakarta, 1992.
8. Diyatno, Sistem Pengenalan Suara Manusia dengan Analisa Spektrum Daya, Tugas Sarjana Jurusan Teknik Elektro Sub Jurusan Elektronika, ITB, Bandung 1990.
9. Hadi, Syamsul, Perencanaan dan Pembuatan Sistem Pengembangan untuk Mikroprosesor Pengolah Sinyal Digital TMS3201X Pada Mikrokomputer IBM PC, Tugas Akhir Jurusan Teknik Elektro Bidang Studi Elektronika, ITS,

Surabaya, 1991.

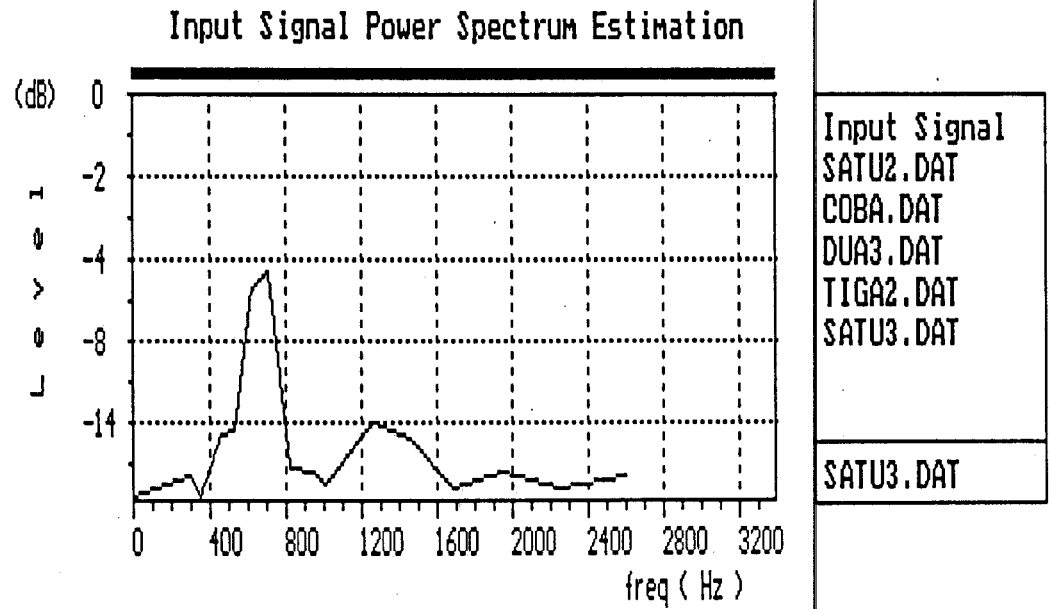
10. Martin, Thomas B., Practical Applications of Voice Input to Machines, PROCEEDING OF THE IEEE, Vol. 64, No. 4, April 1976.

11. Rosenberg, Aaron E., Automatic Speaker Verification: A Review, PROCEEDING OF THE IEEE, Vol. 64, No. 4, April 1976.

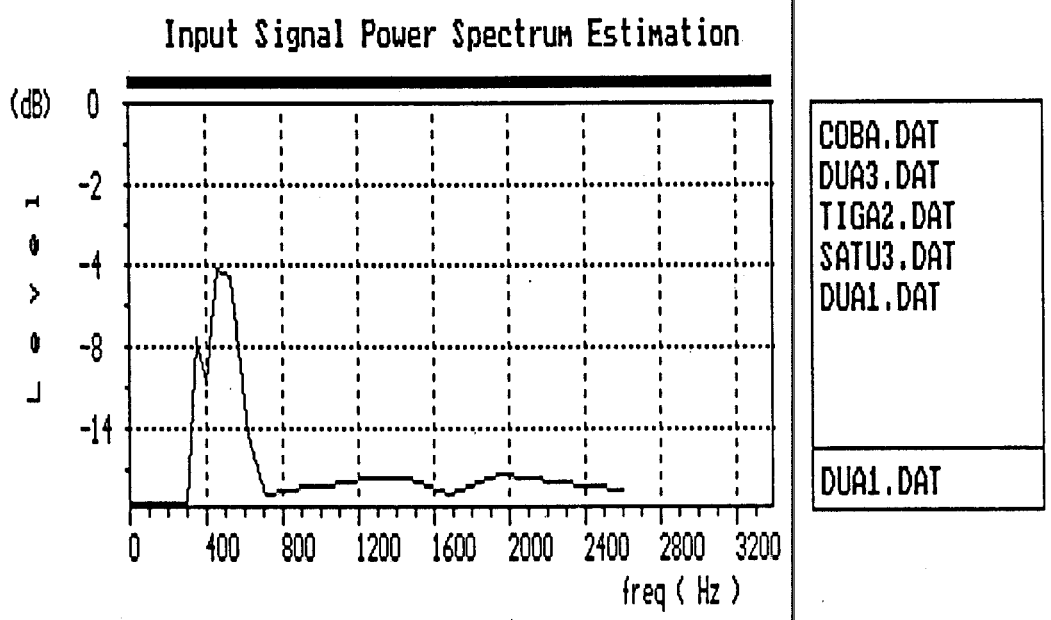
12. Ney, Hermann, Dynamic Programming Algorithm for Optiomal Estimation of Speech Parameters Contour, IEEE TRANSACTIONS OF SYSTEMS, MAN, AND CYBERNETICS, Vol.SMC-13, March/April 1983.



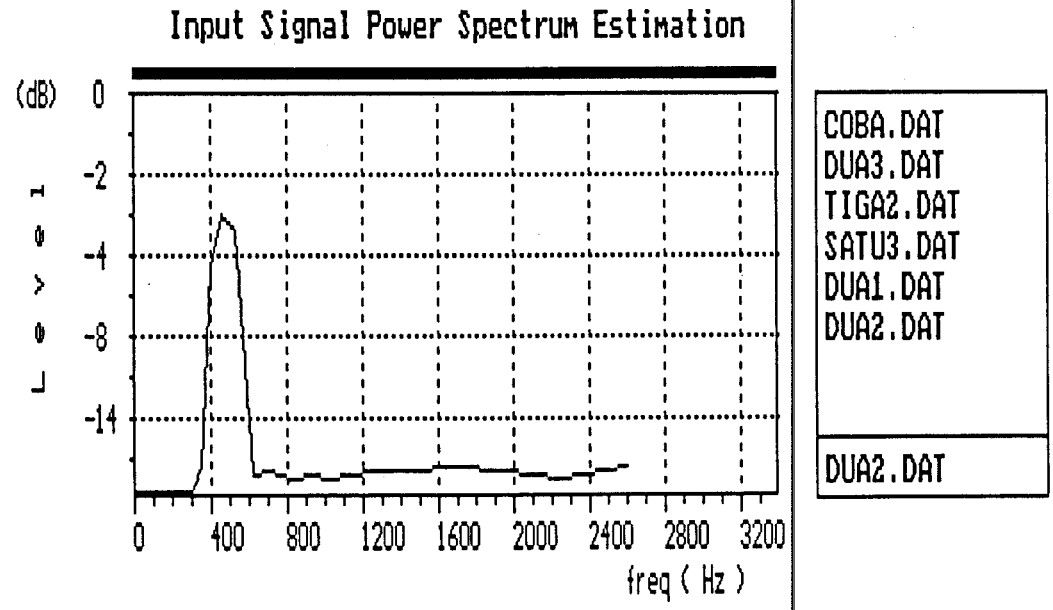
Sinyal ucapan "satu"



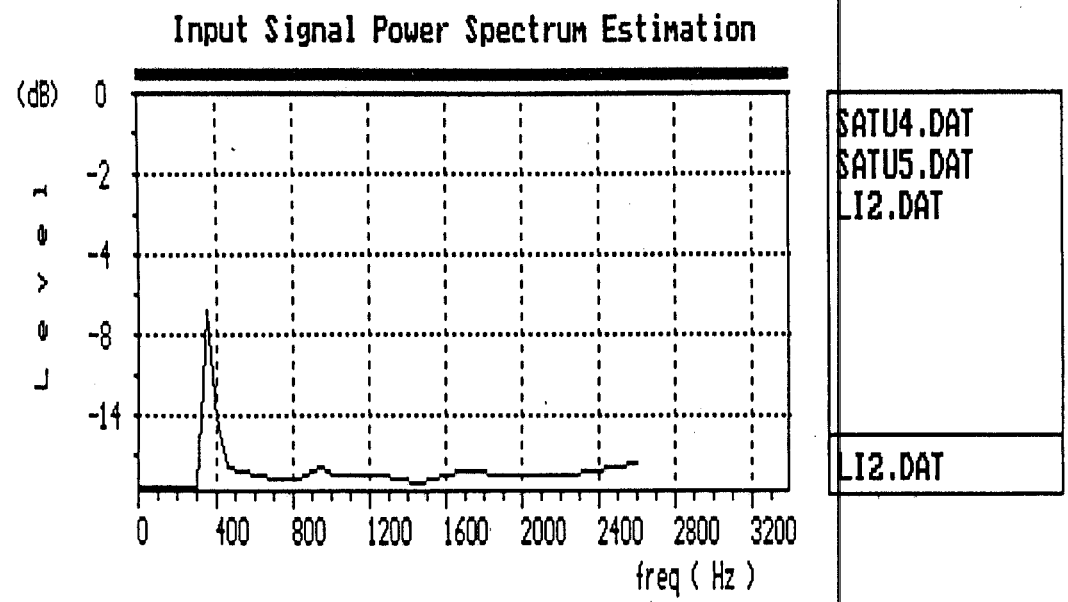
Sinyal ucapan "satu"



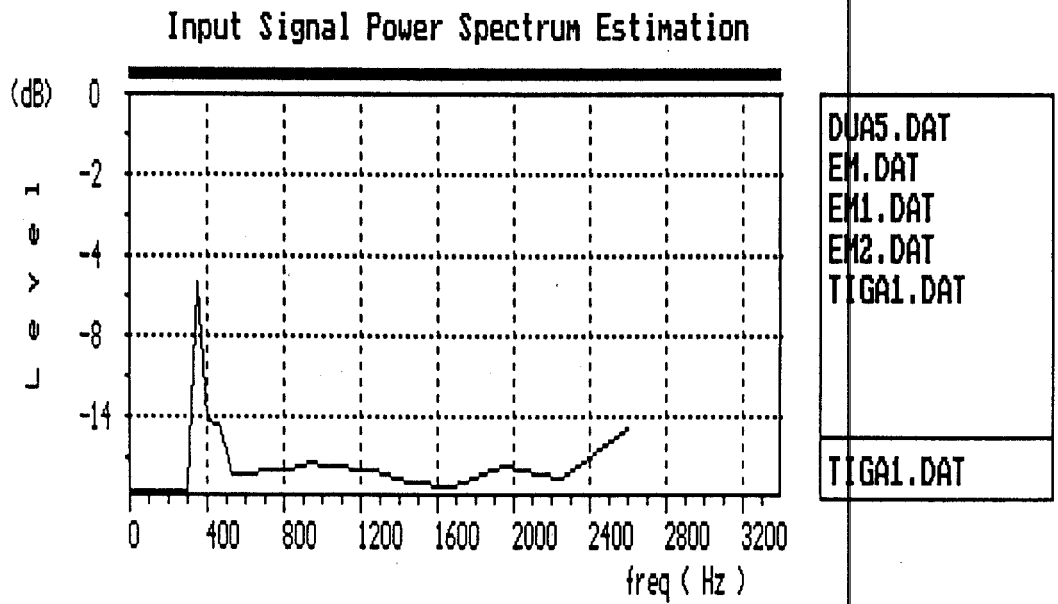
Sinyal ucapan "dua"



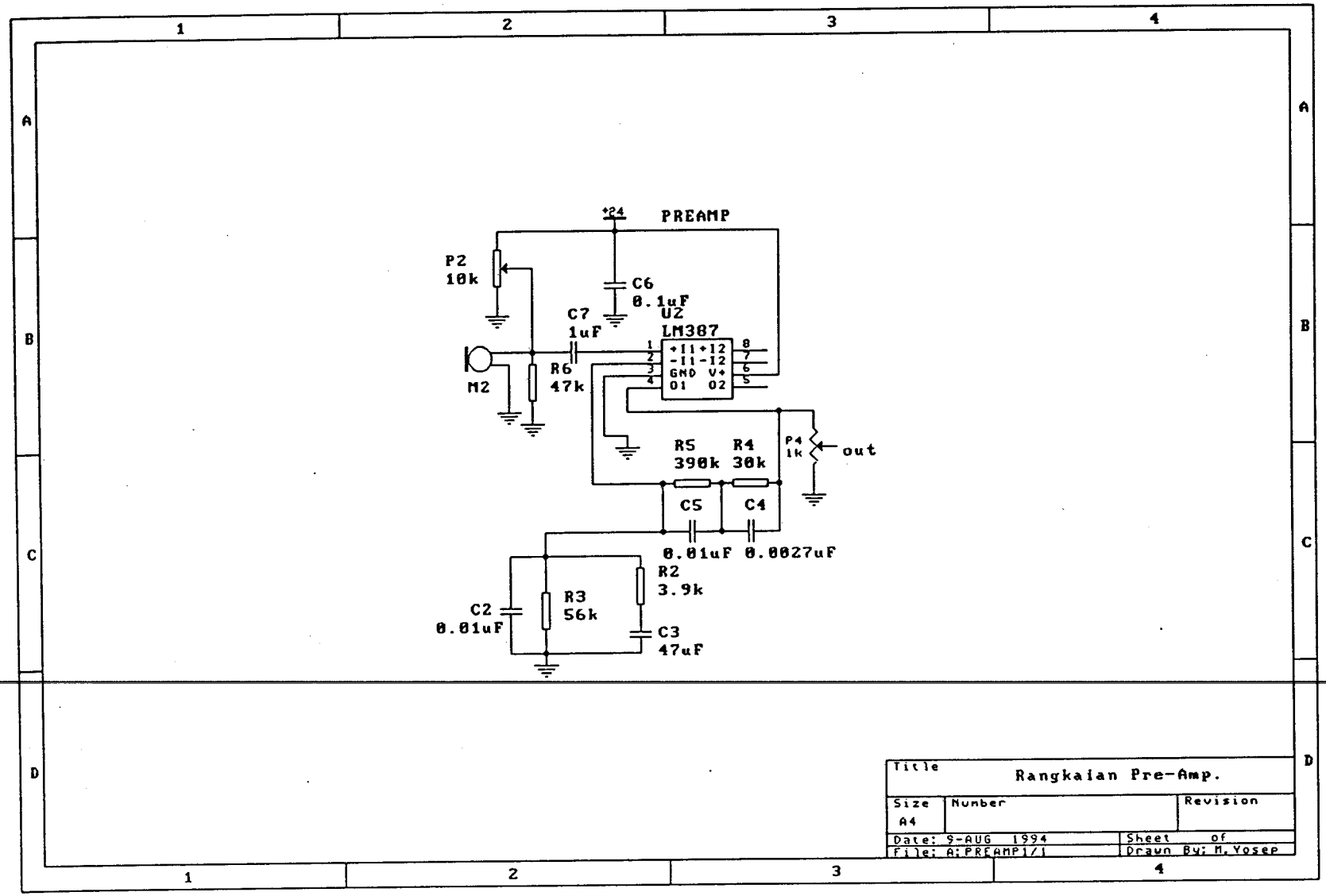
Sinyal ucapan "dua"

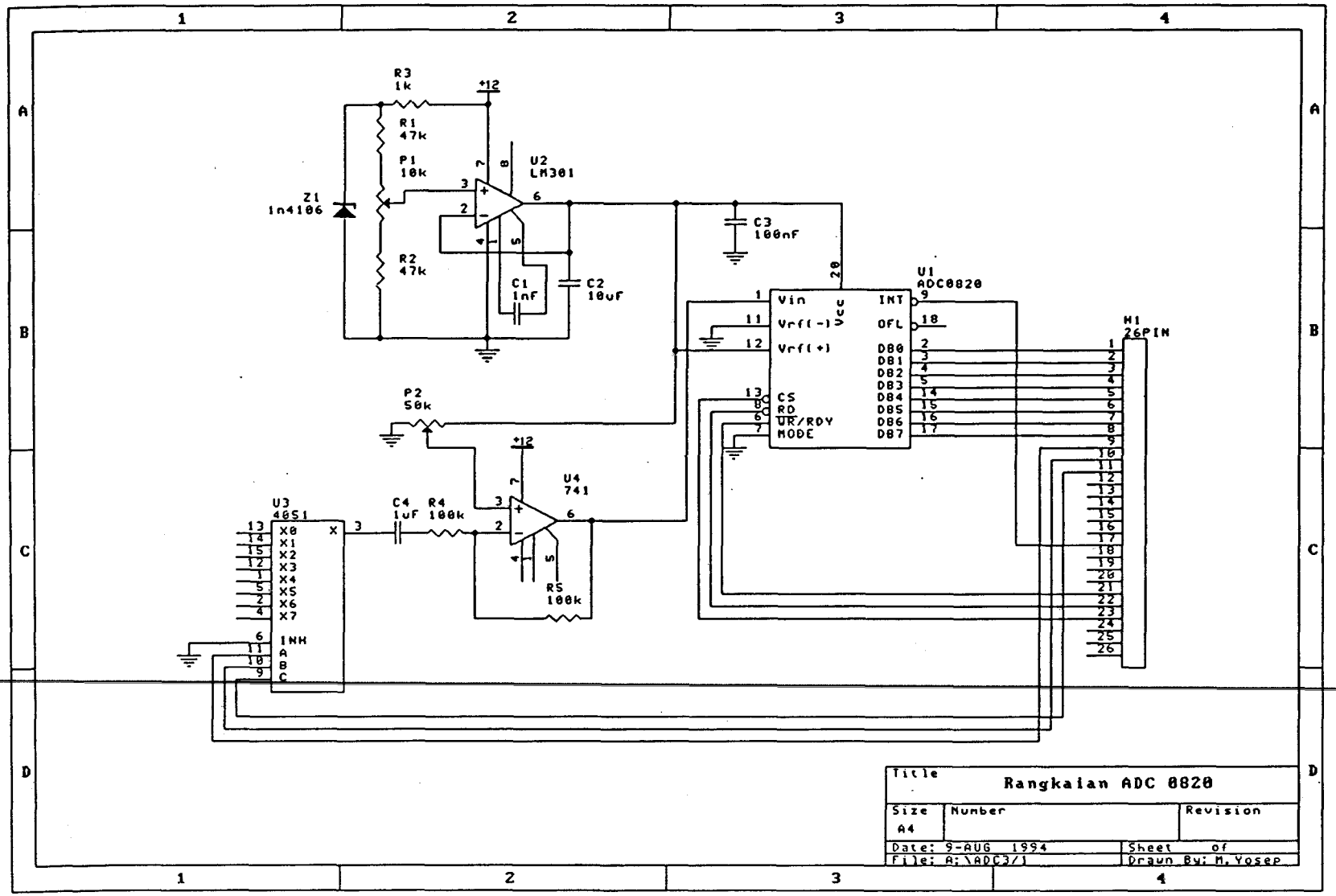


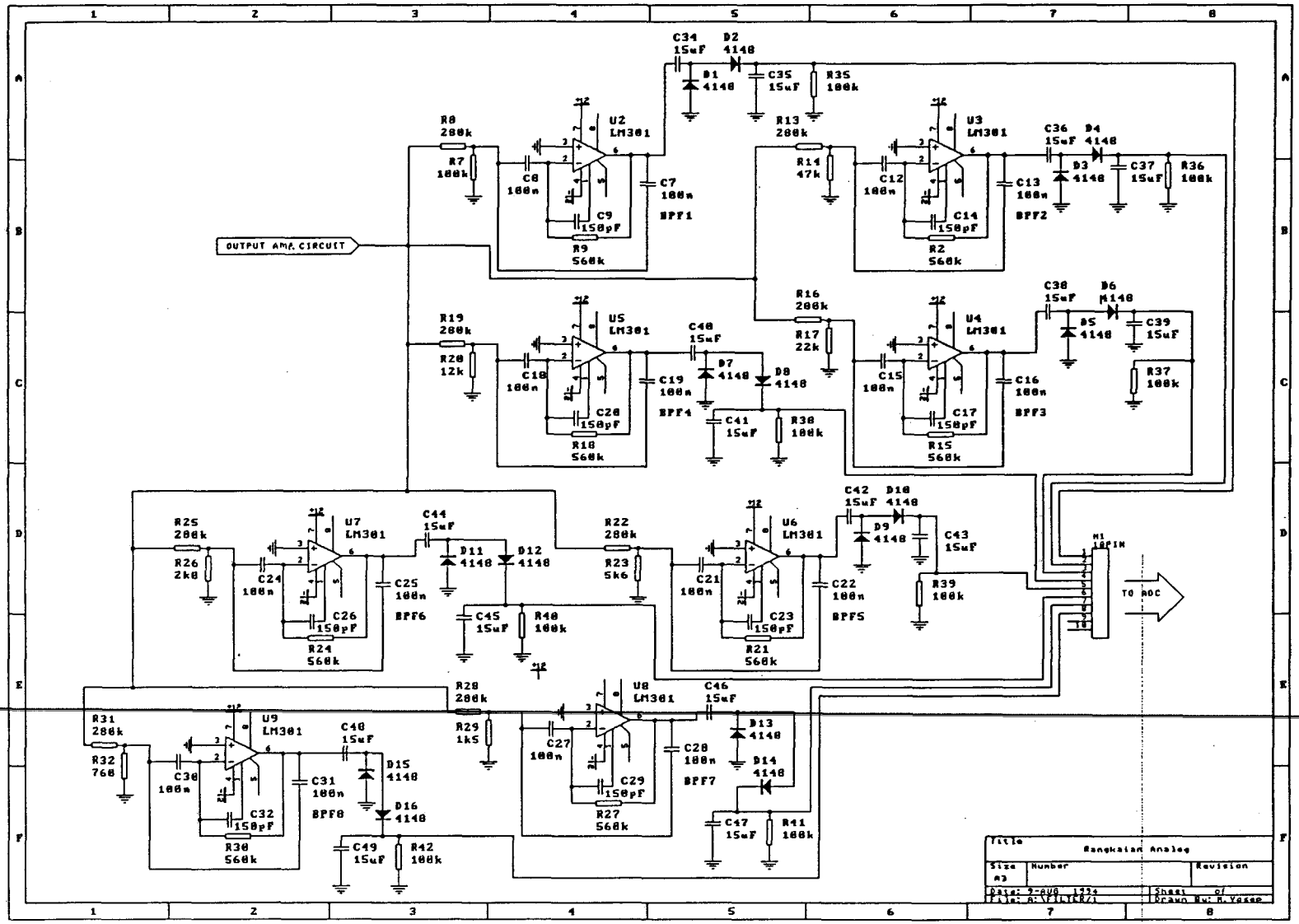
Sinyal ucapan "tiga"

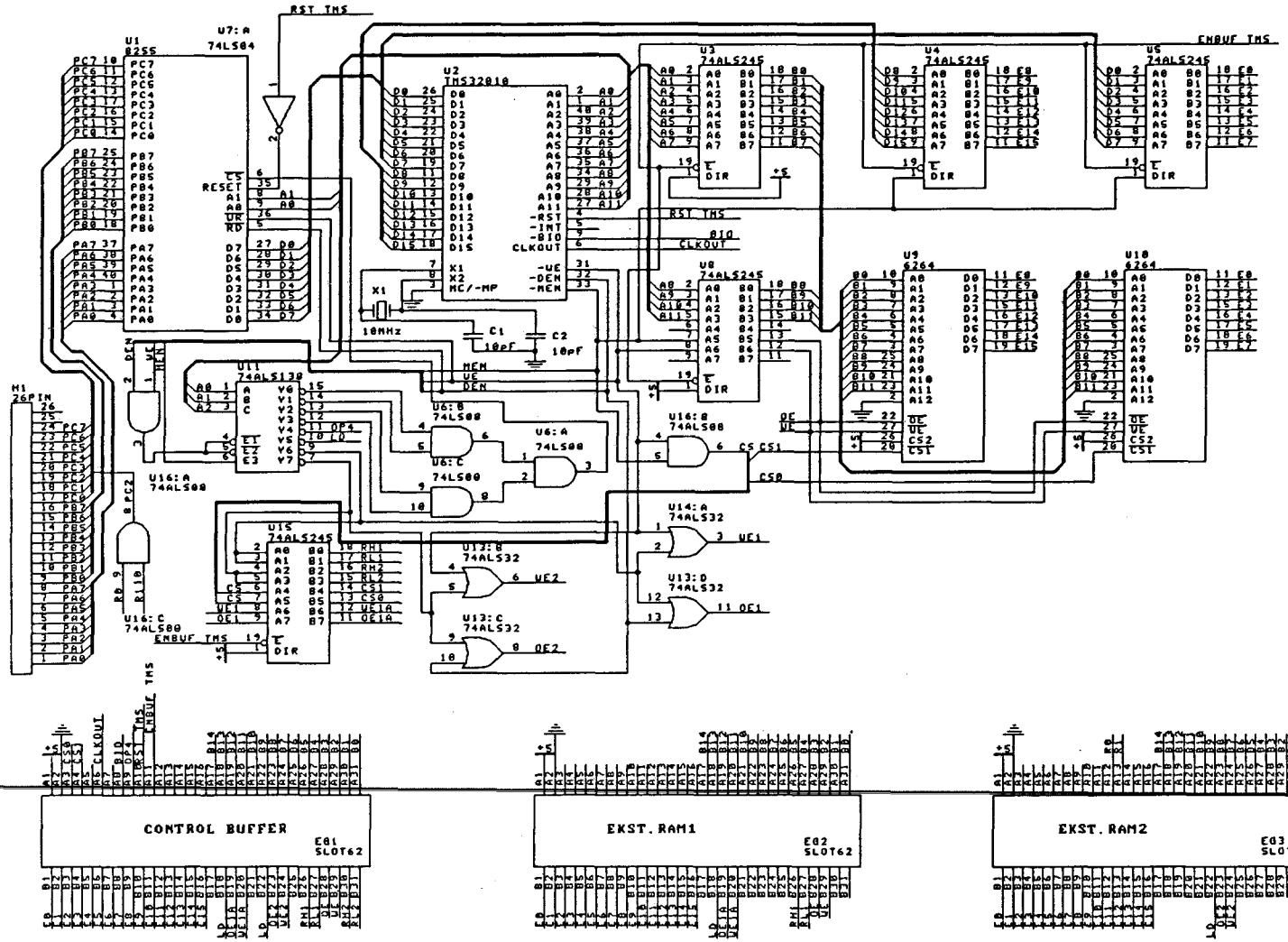


Sinyal ucapan "tiga"

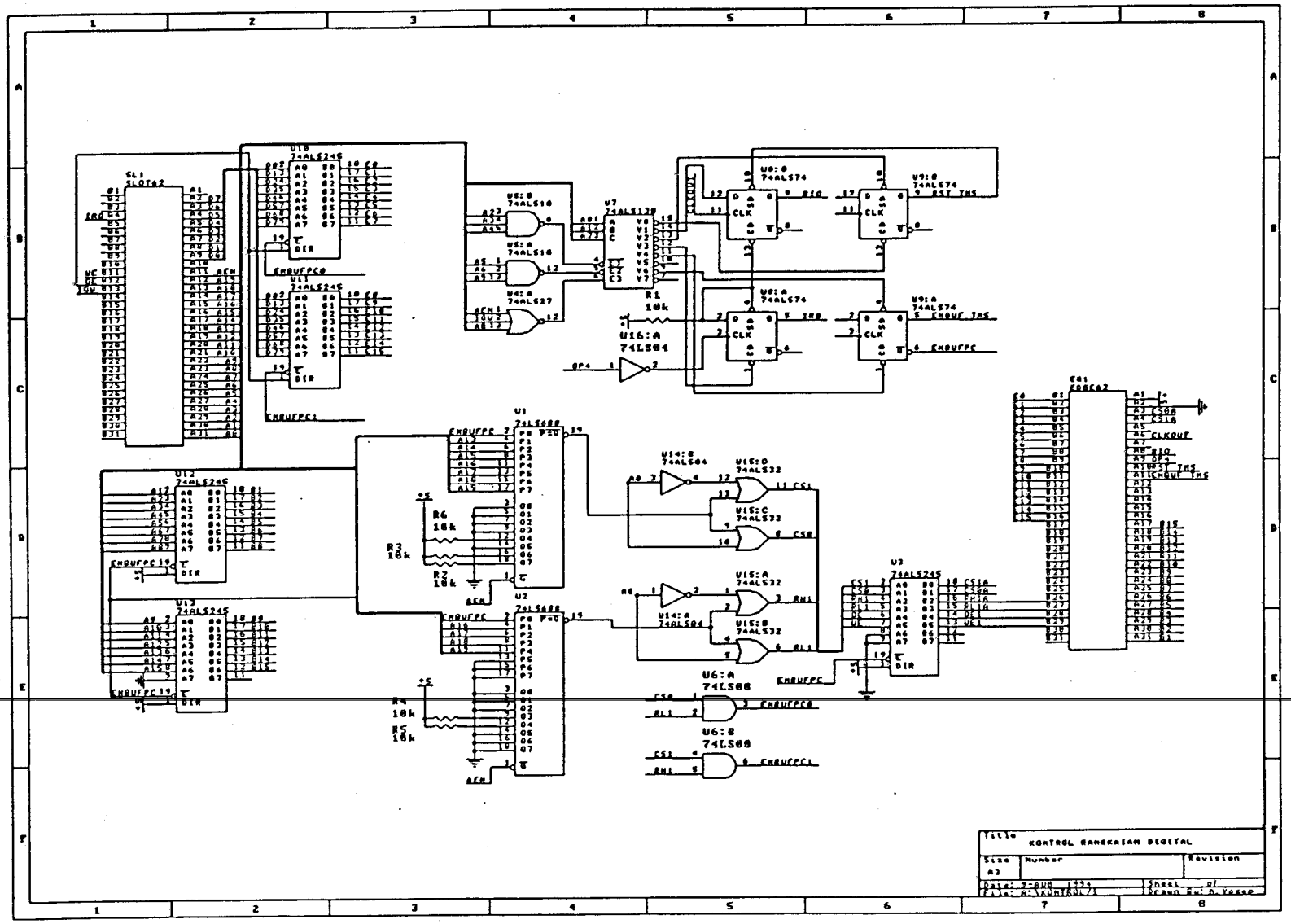

PREAMP
+24
P2
10k
C6
0.1uF
C7
1uF
U2
LM387
+I1 +I2
-I1 -I2
GND V+
O1 O2
M2
R6
47k
R5
390k
R4
30k
P4
1k
out
C5
C4
0.01uF
0.0027uF
R2
3.9k
C2
0.01uF
R3
56k
C3
47uF
Title
Rangkaian Pre-Amp.
Size
A4
Number
Revision
Date: 9-AUG 1994
Sheet of
File: A:PREAMP1/1
Drawn By: M.Yosep

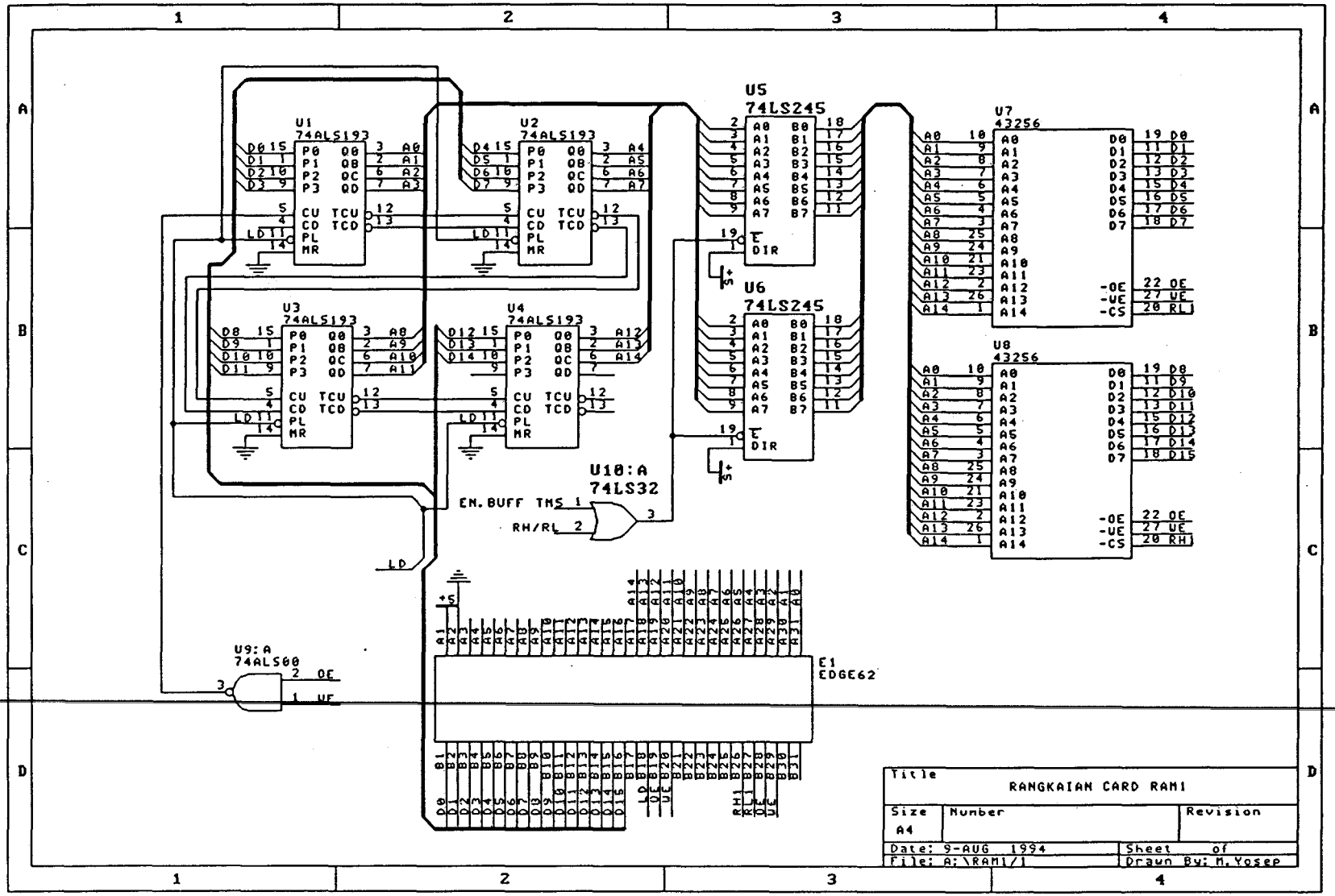

R3
1k
+12
R1
47k
P1
10k
U2
LM301
Z1
1n4106
R2
47k
C1
1nF
C2
10uF
C3
100nF
U1
ADC0820
Vin
Vcc
INT
Vref(-)
OFL
Vref(+)
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
CS
RD
WR/RDY
MODE
H1
26PIN
P2
50k
U4
741
U3
4051
C4
1uF
R4
100k
RS
100k
X0
X1
X2
X3
X4
X5
X6
X7
INH
A
B
C
Title
Rangkaian ADC 0820
Size
A4
Number
Revision
Date: 9-AUG 1994
Sheet of
File: A:\ADC3/1
Drawn By: M. Yosep

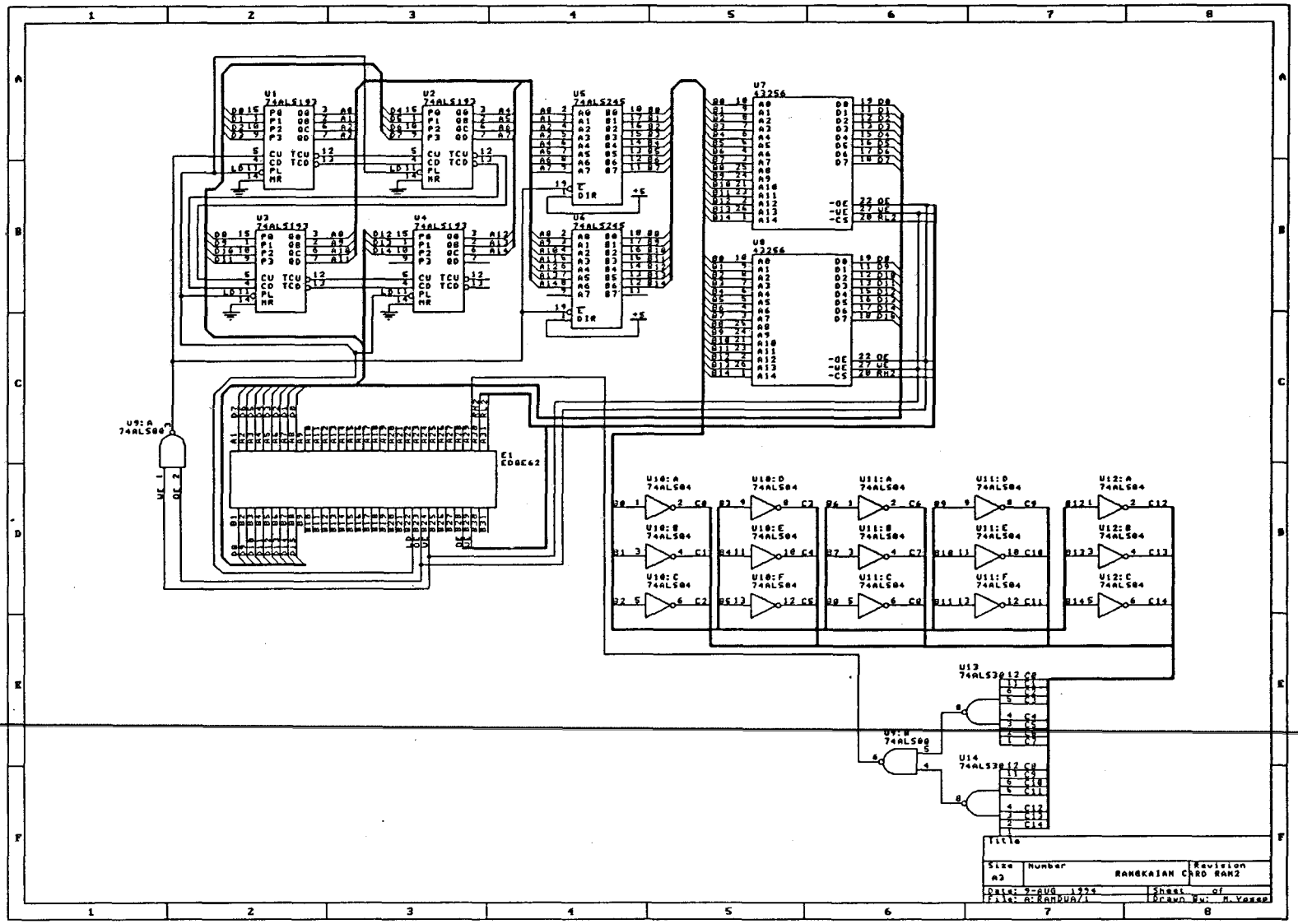

OUTPUT AMP CIRCUIT
U2 LM301
U3 LM301
U4 LM301
U5 LM301
U6 LM301
U7 LM301
U8 LM301
U9 LM301
BPF1
BPF2
BPF3
BPF4
BPF5
BPF6
BPF7
BPF8
M1 10PIN
TO ADC
Title
Rangkaian Analog
Size A3
Number
Revision
Date: 9-AUG-1994
Sheet of
File: A:\FILTER2\1
Drawn By: R.Yasa



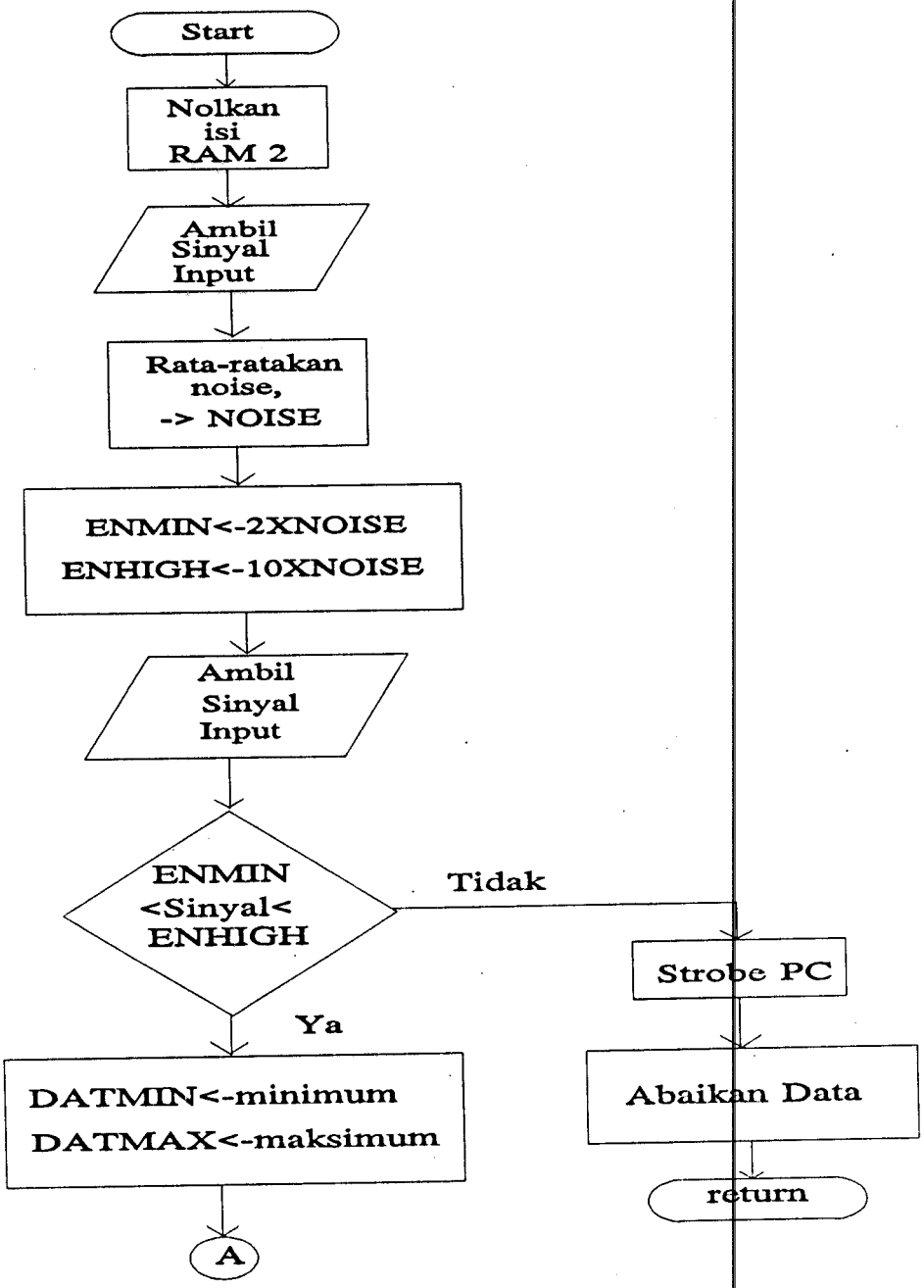
RANGKAIAN CARD PROSESOR/M. Yasep

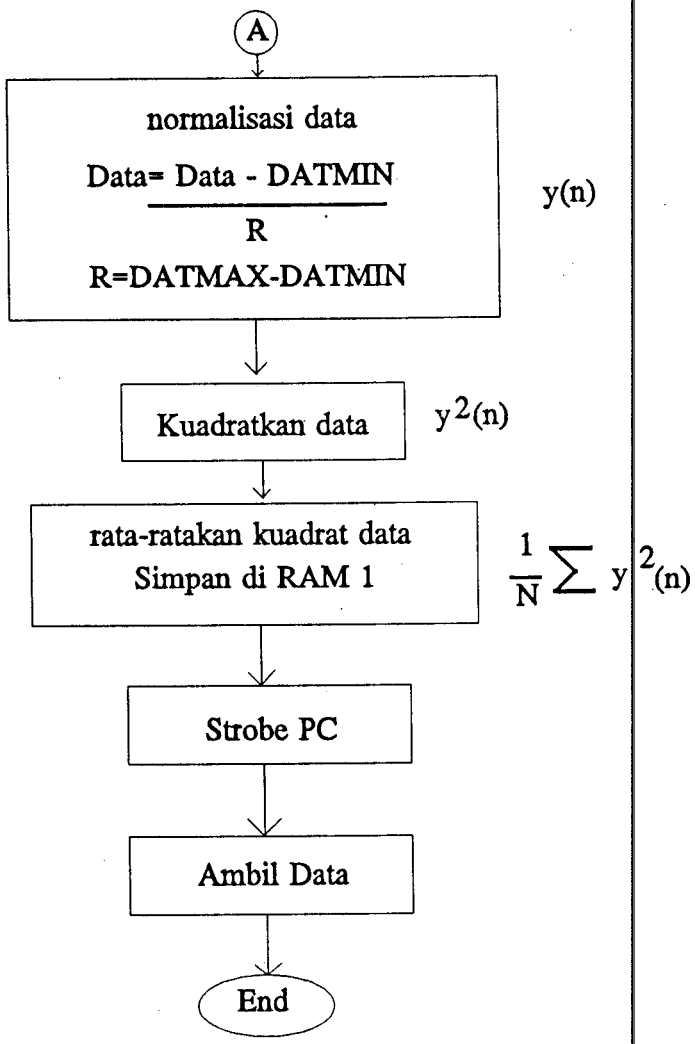

SL1
SLOT#2
U10
74ALS245
U11
74ALS245
U12
74ALS245
U13
74ALS245
U1
74LS688
U2
74LS688
U7
74ALS138
U8:B
74ALS74
U8:A
74ALS74
U9:B
74ALS74
U9:A
74ALS74
U15:D
74ALS32
U15:C
74ALS32
U15:A
74ALS32
U15:B
74ALS32
U14:B
74ALS04
U14:A
74ALS04
U16:A
74LS04
U6:A
74LS08
U6:B
74LS08
U3
74ALS245
CR1
CRCA2
R1
10k
R2
10k
R3
10k
R4
10k
R5
10k
R6
10k
CHBUFPC0
CHBUFPC1
CHBUFPC
CLKOUT
Title
KONTROL RANGKAIAN DIGITAL
Size
A3
Number
Revision
Date: 2-AUG-93
Sheet of
Drawn By: A.Yasa


U1
74ALS193
U2
74ALS193
U3
74ALS193
U4
74ALS193
U5
74LS245
DIR
U6
74LS245
DIR
U7
43256
U8
43256
U10:A
74LS32
EN.BUFF TMS
RH/RL
LD
U9:A
74ALS00
OE
WE
E1
EDGE62
+5
Title
RANGKAIAN CARD RAM1
Size
A4
Number
Revision
Date: 9-AUG 1994
Sheet of
File: A:\RAM1/1
Drawn By: M.Yosep

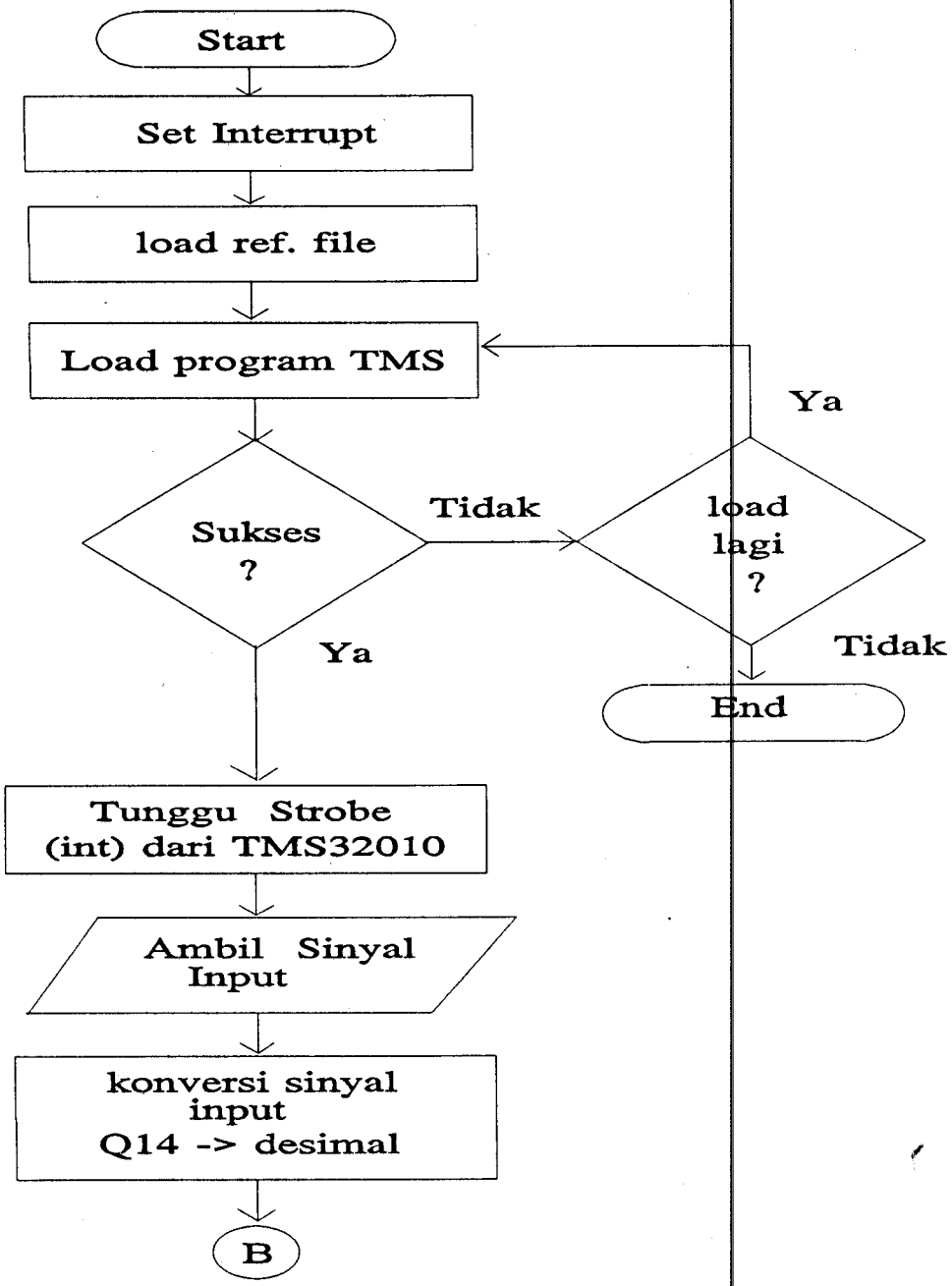

U1 74ALS193
U2 74ALS193
U3 74ALS193
U4 74ALS193
U5 74ALS245
U6 74ALS245
U7 43256
U8 43256
U9:A 74ALS00
E1 EDGE62
U10:A 74ALS04
U10:B 74ALS04
U10:C 74ALS04
U10:D 74ALS04
U10:E 74ALS04
U10:F 74ALS04
U11:A 74ALS04
U11:B 74ALS04
U11:C 74ALS04
U11:D 74ALS04
U11:E 74ALS04
U11:F 74ALS04
U12:A 74ALS04
U12:B 74ALS04
U12:C 74ALS04
U13 74ALS30
U9:B 74ALS00
U14 74ALS30
Title
Size A3
Number
RAMDISK CARD RAM2
Revision
Date: 9-AUG 1994
File: A:RAMDUA/1
Sheet of
Drawn By: M.Yasser

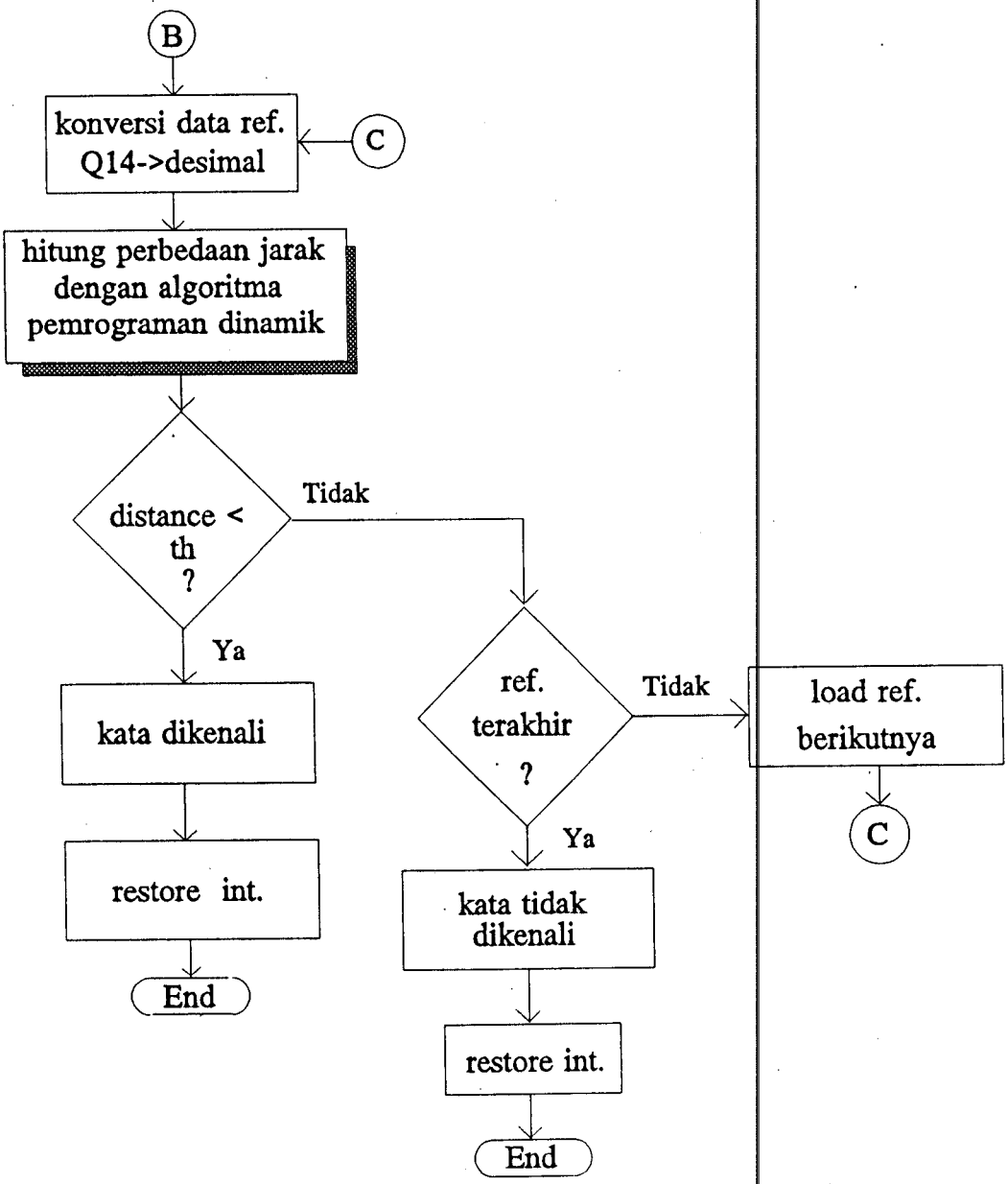
Flow Chart Program Penghitungan Spektrum Daya dengan Metoda Band-Pass Filter Bank (Program pada μP TMS32010)




A
normalisasi data
Data= Data - DATMIN
R
R=DATMAX-DATMIN
y(n)
Kuadratkan data
y2(n)
rata-ratakan kuadrat data
Simpan di RAM 1
1/N ∑ y2(n)
Strobe PC
Ambil Data
End

Flow Chart Program Proses Pengenalan Suara




B
konversi data ref. Q14->desimal
C
hitung perbedaan jarak dengan algoritma pemrograman dinamik
distance < th ?
Tidak
Ya
kata dikenali
restore int.
End
ref. terakhir ?
Tidak
load ref. berikutnya
C
Ya
kata tidak dikenali
restore int.
End

```
;=================================================
; Program untuk Penghitungan Estimasi Spektrum Daya
;         (Program pada μP TMS32010)
;=================================================

      tms 32010

      aorg 0
      b MAIN
;     b ISR

;Define Port and Variabel
;=========================
PA    equ 0
PB    equ 1
PC    equ 2
PCW   equ 3
P4    equ 4
P5    equ 5
P6    equ 6
P7    equ 7

;DEFINE VARIABLE FOR KEY-POINT ALIGNMENT PROGRAM
;================================================
DAT1  equ 8
DAT2  equ 9
POS1  equ 13
POS2  equ 14
DATMAX equ 15
DATMIN equ 16
NXTDAT equ 28
NX1DAT equ 78
NXDAT  equ 10

;* DEFINE VARIABLE
;*================
AR0   equ 0
AR1   equ 1
NOISE  equ 0        ;ENERGI NOISE
INCR   equ 1        ;STEP KENAIKAN
DIST   equ 2        ;JUMLAH PERBEDAAN JARAK
ENERGI equ 7        ;LEVEL ENERGI SINYAL UCAPAN
ARR0   equ 10
ENMIN  equ 13       ;LEVEL ENERGI MINIMAL UCAPAN
ENHIGH equ 14       ;LEVEL ENERGI TINGGI UCAPAN
CNT    equ 17
ADDR1  equ 18
ADDR   equ 19       ;POINTER ALAMAT RAM EKSTERNAL
STBLK  equ 20       ;STATUS BLOK MEMORI (|IN|OUT|B1|B0|)
POSBLK equ 21       ;POINTER DALAM BLOK
PTRAN1 equ 22
DATRAM equ 23       ;BUFFER DATA KE/DARI RAM
PTRAN  equ 24       ;TRANSFER VARIABLE
CH     equ 25       ;BUFFER FOR CHANNEL
STAT   equ 26


      aorg 128
;SUBROUTINE FOR DIVIDING NUMBER
;------------------------------
DIVIDE subc PTRAN   ;1
       nop
       subc PTRAN   ;2
       nop
       subc PTRAN   ;3
       nop
       subc PTRAN   ;4
       nop
       subc PTRAN   ;5
       nop
```

```
      subc PTRAN    ;6
      nop
      subc PTRAN    ;7
      nop
      subc PTRAN    ;8
      nop
      subc PTRAN    ;9
      nop
      subc PTRAN    ;10
      nop
      subc PTRAN    ;11
      nop
      subc PTRAN    ;12
      nop
      subc PTRAN    ;13
      nop
      subc PTRAN    ;14
      nop
      subc PTRAN    ;15
      nop
      subc PTRAN    ;16
      nop
      ret

;======MAIN PROGRAM========
MAIN  dint
      rovm

      ldpk 0
      lark AR0,143
      zac
      larp AR0
loop  sacl *
      banz loop
      lack >91
      sacl PTRAN
      out  PTRAN,PCW

      lack 1
      sacl INCR       ;INCR = 1

      zac
      sacl PTRAN
      out  PTRAN,P5

;nolkan isi RAM 2
;----------------
      lark AR0,255
nol7  lark AR1,128
nol7a lack 0
      sacl PTRAN
      out  PTRAN,P6
      out  PTRAN,P7   ;nullified RAM 2
      larp AR1
      banz nol7a
      larp AR0
      banz nol7

;ambil sinyal noise
;-----------------
      lack 0
      sacl PTRAN
      out  PTRAN,P5

      lark AR0,4
out10 lark AR1,99
out20 zac
      sacl PTRAN
      out  PTRAN,PC   ;OUTPUT READY SIGNAL
CKIN0 in   DATRAM,PC  ;GET INT SIGNAL
```

```
      lack 1
      and DATRAM     ;CHECK IS INT LOW ?
      bgz CKIN0      ;IF NOT, CHECK AGAIN
      in  PTRAN,PA   ;GET DATA
      out PTRAN,P7

FULL0 lack >60
      sacl PTRAN     ;
      out PTRAN,PC   ;RESET INT
      lac CH         ;CHECK IF CHANNEL = 0
      bnz TAMBA0     ;IF NOT, BRANCH TO TAMBAH
      out CH,PB      ;SELECT CHANNEL 0
      add INCR
      sacl CH        ;SET CHANNEL = 1
      b   BACK0
TAMBA0 add INCR      ;INCREMENT CHANNEL
      sacl CH        ;SAVE INCREMENTED CHANNEL VALUE
      lack 16
      sacl PTRAN
      lac CH
      sub PTRAN      ;CHECK IF CHANNEL > 16
      bgez NOL0      ;IF YES, BRANCH TO NOL
      out CH,PB
      b   BACK0
NOL0  zac            ;SET CHANNEL ZERO AGAIN
      sacl CH
      out CH,PB

BACK0 larp AR1
      banz out20
      larp AR0
      banz out10

;rata-ratakan besarnya noise
;---------------------------
      zac
      sacl PTRAN
      out PTRAN,P5

      lark AR0,4
INP70 sar AR0,CNT
      zac
      sacl NOISE
      lark AR1,99
INP7A0 in  PTRAN,P7
      lt  PTRAN
      mpy PTRAN
      zac
      apac
      sacl PTRAN
      lac NOISE
      add PTRAN
      sacl NOISE
      larp AR1
      banz INP7A0

      out NOISE,P6
      lar AR0,CNT
      larp AR0
      banz INP70

      zac
      sacl NOISE
      sacl PTRAN
      out PTRAN,P5

      lark AR0,4
INNO  in  PTRAN,P6
      lac PTRAN
      add NOISE
```

```
      sacl NOISE
      larp AR0
      banz INNO

      lack 5
      sacl PTRAN
      lac  NOISE
      call DIVIDE
      sacl NOISE

;ENMIN = 4 x NOISE
      lack 2
      sacl PTRAN
      lt   PTRAN
      mpy  NOISE
      zac
      apac
      sacl ENMIN

;ENHIGH = 10 x NOISE
      lack 10
      sacl PTRAN
      lt   PTRAN
      mpy  NOISE
      zac
      apac
      sacl ENHIGH

;Cek Port C bit 3
;(apakah ada sinyal clock)
;-------------------------
cek1  lack 8
      in   PTRAN,PC
      and  PTRAN
      bgz  tund
      b    cek1

;tunda bbrp. saat
;----------------
tund  larp AR0,255
tunda lark AR1,255
lagi1 zac
      larp AR1
      banz lagi1
      lark AR1,255
lagi2 zac
      larp AR1
      banz lagi2
      lark AR1,255
lagi3 zac
      larp AR1
      banz lagi3
      larp AR0
      banz tunda

;ambil sinyal input dan simpan di RAM 2
;--------------------------------------
      lack 0
      sacl CH
      sacl PTRAN
      out  PTRAN,P5

indata lark AR0,255
out1  lac  CH        ;CHECK IF CHANNEL = 0
      bnz  TAMBAH    ;IF NOT, BRANCH TO TAMBAH
      out  CH,PB     ;SELECT CHANNEL 0
      add  INCR
      sacl CH        ;SET CHANNEL = 1
      b    BACK
TAMBAH add INCR      ;INCREMENT CHANNEL
```

```
    sacl CH        ;SAVE INCREMENTED CHANNEL VALUE
    lack 16
    sacl PTRAN
    lac  CH
    sub  PTRAN     ;CHECK IF CHANNEL > 8
    bgez NOL       ;IF YES, BRANCH TO NOL
    out  CH,PB
    b    BACK
NOL zac            ;SET CHANNEL ZERO AGAIN
    sacl CH
    out  CH,PB

BACK:
    lark AR1,31
out2 zac
    sacl PTRAN
    out  PTRAN,PC  ;OUTPUT READY SIGNAL
CKIN in  DATRAM,PC ;GET INT SIGNAL
    lack 1
    and  DATRAM    ;CHECK IS INT LOW ?
    bgz  CKIN      ;IF NOT, CHECK AGAIN
    in   PTRAN,PA  ;GET DATA
    out  PTRAN,P7

FULL lack >60
    sacl PTRAN     ;
    out  PTRAN,PC  ;RESET INT

;Cek Port C bit 1
;---------------
    lack 2
    in   PTRAN,PC
    and  PTRAN
    blez exit1

;Cek port C bit 3
;---------------
cek2 lack 8
    in   PTRAN,PC
    and  PTRAN
    blez cek2

cek3 lack 8
    in   PTRAN,PC
    and  PTRAN
    bgz  cek3

    larp AR1
    banz out2
    larp AR0
    banz out1
;-------------------------

exit1 zac
    sacl ADDR
    zac
    sacl ADDR1
    zac
    sacl PTRAN
    out  PTRAN,P5

    lark AR0,255   ;
INP7 sar AR0,CNT   ;
    zac            ;
    sacl ENERGI    ;
    lark AR1,99    ;>>hitung energi setiap 100 data
INP7A in  PTRAN,P7 ;
    lt   PTRAN     ;
    mpy  PTRAN     ;
    zac            ;
```

```
      apac            ;
      sacl PTRAN      ;
      lac  ENERGI     ;
      add  PTRAN      ;
      sacl ENERGI     ;
      larp AR1        ;
      banz INP7A      ;
                      ;

      out  ENERGI,P6  ;
      lar  AR0,CNT    ;
      larp AR0        ;
      banz INP7       ;

;periksa level energi
;--------------------
      zac
      sacl CNT
      sacl PTRAN
      out  PTRAN,P5

      lark AR0,255    ;
IN76A in   PTRAN,P6   ;
      lac  ENMIN      ;
      sub  PTRAN      ;periksa energi minimum sinyal
      blez MOV76      ;
      lac  ADDR1      ;
      add  1          ;
      sacl ADDR1      ;
      larp AR0        ;
      banz IN76A      ;
      b    EXITS      ;energi sinyal tidak cukup besar

MOV76 lack 255
      sub  ADDR1
      sacl ADDR

      lar  AR0,ADDR   ;
IN76B in   PTRAN,P6   ;
      lac  ENHIGH     ;
      sub  PTRAN      ;periksa energi maksimum sinyal
      blez EXITS      ;energi sinyal terlalu besar

;Lanjutkan Proses
;----------------
      lack 0
      sacl ADDR
      out  ADDR,P5

      lark AR0,255    ;
INP71 sar  AR0,CNT    ;
      zac             ;
      sacl ENERGI     ;
      lark AR1,31     ;>>jumlahkan setiap 32 data
INP7A1 in  PTRAN,P7   ;
      lac  ENERGI     ;
      add  PTRAN      ;
      sacl ENERGI     ;
      larp AR1        ;
      banz INP7A1     ;

      lark AR0,200
tunda1 lack 0
      larp AR0
      banz tunda1                ;
      out  ENERGI,P6  ;
      lar  AR0,CNT    ;
      larp AR0        ;
      banz INP71      ;
```

```
;tentukan data terbesar
;--------------------
     lac  NXTDAT
     sacl DATMAX     ;STORE FIRST DATA IN DATMAX
     lack 4
     sacl ADDR

     lark AR0,4
ULDET  sar  AR0,CNT

     out  ADDR,P5    ;LOAD COUNTER WITH ADDR. OF BLOCK I

     lack 1
     sacl 1

     lark AR0,49     ;NUMBER OF DATA TO GET
     lark AR1,28     ;START OF RAM LOCATION TO BE PLACED IN
INPUT1 larp AR1
     in   *+,P6,AR0  ;GET DATA FROM RAM 1
     banz INPUT1

;determine the position of max. data of blokc I
;-----------------------------------------
     lac  DATMAX     ;LOAD ACC. WITH DATMAX
     lark AR1,28     ;LOAD FIRST DATA POSITION IN AR1
     lark AR0,49     ;NUMBER OF DATA TO ACCESS
ULG1   larp AR1
     sub  *+,0,AR0   ;DATMAX - [AR1]
     blz  TUKAR1     ;BRANCH IF [AR1] > DATMAX

     lac  DATMAX
     banz ULG1
     b    NXT1

TUKAR1 sar  AR1,PTRAN  ;SAVE POSITION IN TEMP. MEMORY
     lac  PTRAN
     sub  1          ;ADJUST POSITION
     sacl POS1       ;SAVE POSITION OF MAX. DATA
     lar  AR1,POS1
     larp AR1
     lac  *          ;LOAD ACC. WITH MAX. DATA
     sacl DATMAX     ;STORE MAX. DATA TO DATMAX
     lac  DATMAX     ;LOAD ACC. WITH MAX. DATA
     lar  AR1,PTRAN  ;RETURN THE POSITION OF TEMP. MEMORY
     larp AR0
     banz ULG1

NXT1   lack 49
     add  ADDR
     sacl ADDR
     lar  AR0,CNT
     larp AR0
     banz  ULDET

;tentukan data terkecil
;---------------------
     lack 1
     sacl PTRAN
     out  PTRAN,P5
     in   DATMIN,P6  ;STORE FIRST DATA IN DATMIN
     lack 4
     sacl ADDR

     lark AR0,4
 ULDET1 sar  AR0,CNT
     out  ADDR,P5

     lark AR0,49     ;NUMBER OF DATA TO GET
     lark AR1,28     ;START OF DATA RAM TO BE PLACED IN
 INPUT2 larp AR1
```

```
     in   *+,P6,AR0  ;GET DATA FROM RAM 1
     banz INPUT2

;determine the position of min. data of block I
;-----------------------------------------
     lac  DATMIN     ;LOAD ACC. WITH DATMIN
     lark AR1,28     ;LOAD FIRST DATA POSITION IN AR1
     lark AR0,49     ;NUMBER OF DATA TO ACCESS
ULG2 larp AR1
     sub  *+,0,AR0   ;[AR1] - DATMIN
     bgz  TUKAR2     ;BRANCH IF > 0

     lac  DATMIN
     banz ULG2
     b    NXT2

TUKAR2 sar  AR1,PTRAN  ;SAVE POSITION IN TEMP. MEMORY
     lac  PTRAN
     sub  1          ;ADJUST POSITION
     sacl POS1       ;SAVE POSITION OF MIN. DATA
     lar  AR1,POS1
     larp AR1
     lac  *          ;LOAD ACC. WITH MIN. DATA
     sacl DATMIN     ;STORE MIN. DATA TO DATMIN
     lac  DATMIN     ;LOAD ACC. WITH MIN. DATA
     lar  AR1,PTRAN  ;RETURN THE POSITION OF TEMP. MEMORY
     larp AR0
     banz ULG2

NXT2  lack 49
     add  ADDR
     sacl ADDR
     lar  AR0,CNT
     larp AR0
     banz ULDET1

;normalisasi data
;----------------
     lac  DATMAX
     sub  DATMIN
     sacl DATMAX

     lack 0
     sacl ADDR
     out  ADDR,P5

     lark AR0,249
NORM  in   PTRAN,P6
     lac  PTRAN
     sub  DATMIN
     abs
     lark AR1,29
     larp AR1
DIVD  subc DATMAX
     banz DIVD

     sacl PTRAN
     lark AR1,100
tunda2 lack 0
     larp AR1
     banz tunda2

     out  PTRAN,P7  ;simpan data hasil normalisasi di RAM 2
     larp AR0
     banz NORM

;rata-ratakan kuadrat data
;-------------------------
     lack 0
     sacl ADDR
```

```
      out  ADDR,P5

      lark AR0,249
INORM in   DATRAM,P7
      lt   DATRAM
      mpy  DATRAM      ;kuadratkan data hasil normalisasi
      pac
      sach DATRAM,1
      lark AR1,100
tunda3 lack 0
      larp AR1
      banz tunda3

      out  DATRAM,P6
      larp AR0
      banz INORM

      lack 14
      sacl ADDR
      out  ADDR,P5

      lark AR0,15
      lark AR1,28
INORN larp AR1
      in   *+,P6,AR0
      banz INORN

      lark AR0,13
INORM1 sar  AR0,CNT

      lark AR0,15
      lark AR1,28
INORM2 larp AR1
      in   DATRAM,P6
      lac  *
      add  DATRAM
      sacl DATRAM
      lack 2
      sacl PTRAN
      lac  DATRAM
      subc PTRAN    ;1
      nop
      subc PTRAN    ;2
      nop
      subc PTRAN    ;3
      nop
      subc PTRAN    ;4
      nop
      subc PTRAN    ;5
      nop
      subc PTRAN    ;6
      nop
      subc PTRAN    ;7
      nop
      subc PTRAN    ;8
      nop
      subc PTRAN    ;9
      nop
      subc PTRAN    ;10
      nop
      subc PTRAN    ;11
      nop
      subc PTRAN    ;12
      nop
      subc PTRAN    ;13
      nop
      subc PTRAN    ;14
      nop
      subc PTRAN    ;15
      nop
```

```
      subc PTRAN    ;16
      nop
      sacl *+
      larp AR0
      banz INORM2

      lar  AR0,CNT
      larp AR0
      banz INORM1

      lack 0
      sacl ADDR
      out  ADDR,P5

      lark AR0,255
ddd   zac
      sacl PTRAN
      out  PTRAN,P6
      larp AR0
      banz ddd

;simpan hasil rata-rata kuadrat data di RAM 1
;---------------------------------------------
scc:
      lack 0
      sacl ADDR,P5
      out  ADDR,P5

      lark AR0,15
      lark AR1,28
ONORM  larp AR1
      out  *+,P6,AR0
      banz ONORM
      b    EXOT1
;---------------------
;
EXITS  lack 1         ;DATA NOT VALID
      sacl STAT
      out  STAT,P6
      b    EXOT1

EXOT   lack 0
      sacl STAT
      out  STAT,P6

EXOT1  lack 0
      sacl PTRAN
      out  PTRAN,P4
WAIT   b    EXOT1
      end
```

```
/***********************************************/
/*   Program untuk Proses Pengenalan Suara     */
/***********************************************/

extern int count;

comp()
{
  FILE *fp, *fp1;
  char far *ptr2=(char far *)MK_FP(0xD000,0);
  char far *ptr5=(char far *)MK_FP(0xA000,0);
  char far *stat=(char far *)MK_FP(0xA000,0x246);
  char *ptr;
  char *nama;
  char a, s1[1], s2[1];
  int i, j, jj, k, m, p, q, r, vali;
  int jud, jum, jum1, inf, thru;
  int datcek[10];
/* level threshold */
   float thld[]={9.5473,10.6459,13.6069,10.3330,10.1310,
                 10.1842,12.2007,6.6402,10.9817,8.4880,
                 7.0705,8.2384,8.7735,9.8251,10.0933};
  unsigned short int trans[32];
  unsigned int valtrans;
  unsigned short int far *ptr4=(unsigned short int far *)MK_FP(0xA000,0);
  unsigned short int *ptr3;
  float sem[20], ref[20];
  float g[30][30], d[30][30];
  float distfirst, distrans[4], distance, err, mins;
  long double distot;
  struct ffblk file;

 /* program start here */
   hidec();
   video_mode();
   draw_bgr();
   draw_border(7,11,9,53,0);
   draw_border(6,10,8,52,63);
    write_string(7,11,"Preparing System for Recognizing Word ...!",63);
     delay(1000);
  inf=0;
  jud=0;
  outportb(0x2fe,0);              /* enable buff. PC */
  setmem(ptr5,0xffff,00);          /* nolkan isi RAM 1 */
  *(stat)=0;                      /* inisialisasi status */

  thru = findfirst("*.dat",&file,0x00);
  for(j=0;j<=jum_data;j++){
    if(thru==0){nama=file.ff_name;
                goto proses1;}
      thru = findnext(&file);
  }
  draw_bgr();
  draw_border(14,22,16,43,0);
  draw_border(13,21,15,42,33);
  write_string(14,22,"No ref. file ...!",33);
  delay(1000);
  goto backtomain;

 proses1:
   ptr3=(unsigned short int *)malloc(512);
   fp1=fopen(nama,"rb");  /* open data ref. I */
   if(!fp1){
     draw_border(14,22,16,43,0);
      draw_border(13,21,15,42,33);
       write_string(14,22,"open ref. error ...!",33);
            delay(1000);
             fclose(fp1);
              draw_bgr();
               goto backtomain;
```

```
    }
    draw_border(14,22,16,43,0);
    draw_border(13,21,15,42,63);
     write_string(14,22,"open ref. file ...! ",63);
          delay(500);
           draw_bgr();

jum1=fread(ptr3,64,1,fp1);          /* load ref. to ptr3 */
if(!jum1){
   draw_border(14,21,16,42,0);
    draw_border(13,20,15,41,33);
     write_string(14,21,"read ref. error ...!",33);
          delay(1000);
           fclose(fp1);
            draw_bgr();
             goto backtomain;
    }
    draw_border(14,22,16,43,0);
     draw_border(13,21,15,42,63);
      write_string(14,22,"read ref. success ! ",63);
          delay(1000);
           draw_bgr();
            fclose(fp1);

 jum=strcmp(nama,"SATU1.DAT");
  if(jum==0){inf=0; goto lanjutkan;}
   jum=strcmp(nama,"SATU2.DAT");
    if(jum==0){inf=1; goto lanjutkan;}
     jum=strcmp(nama,"SATU3.DAT");
      if(jum==0){inf=2; goto lanjutkan;}
          jum=strcmp(nama,"SATU4.DAT");
           if(jum==0){inf=3; goto lanjutkan;}
            jum=strcmp(nama,"SATU5.DAT");
             if(jum==0){inf=4; goto lanjutkan;}

 jum=strcmp(nama,"DUA1.DAT");
  if(jum==0){inf=5; goto lanjutkan;}
   jum=strcmp(nama,"DUA2.DAT");
    if(jum==0){inf=6; goto lanjutkan;}
     jum=strcmp(nama,"DUA3.DAT");
      if(jum==0){inf=7; goto lanjutkan;}
          jum=strcmp(nama,"DUA4.DAT");
           if(jum==0){inf=8; goto lanjutkan;}
            jum=strcmp(nama,"DUA5.DAT");
             if(jum==0){inf=9; goto lanjutkan;}

 jum=strcmp(nama,"TIGA1.DAT");
  if(jum==0){inf=10; goto lanjutkan;}
   jum=strcmp(nama,"TIGA2.DAT");
    if(jum==0){inf=11; goto lanjutkan;}
     jum=strcmp(nama,"TIGA3.DAT");
      if(jum==0){inf=12; goto lanjutkan;}
          jum=strcmp(nama,"TIGA4.DAT");
           if(jum==0){inf=13;}

lanjutkan:
 ptr=(char *) malloc(8192);          /* allocate memory */
 if(!ptr){
    draw_border(14,18,16,48,0);
     draw_border(13,17,15,47,33);
      write_string(14,18,"allocate memory failed ...",33);
           delay(1000);
            goto backtomain;
    }

 fp=fopen("comp1.bin","rb");        /* open program file for TMS */
 if(!fp){
    draw_border(14,19,16,48,0);
     draw_border(13,18,15,47,33);
      write_string(14,19,"open TMS program error ....!",33);
```

```
          delay(1000);
           fclose(fp);
            goto backtomain;
    }
   draw_border(14,19,16,48,0);
    draw_border(13,17,15,47,63);
     write_string(14,18,"open TMS program file  .....!",63);
          delay(500);
           draw_bgr();

 jum=fread(ptr,128,64,fp);          /* read program file */
 if(!jum){
   draw_border(14,18,16,48,0);
    draw_border(13,17,15,47,33);
     write_string(14,18,"read TMS program error ....!",33);
          delay(1000);
           fclose(fp);
            goto backtomain;
    }
    draw_border(14,18,16,48,0);
    draw_border(13,17,15,47,63);
     write_string(14,18,"read TMS program success ...!",63);
          delay(1000);
           draw_bgr();
            fclose(fp);

reload:
  outportb(0x2fe,0);       /* enable buffer PC */
   memmove(ptr2,ptr,8192); /* load TMS program */
    draw_border(14,16,17,50,0);
     draw_border(13,15,16,49,63);
      write_string(14,16,"program loaded to D000:0000 ...  ",63);
           write_string(15,16,"checking loaded program file ...!",63);
            delay(1000);
             draw_bgr();

  for(i=0;i<8191;i++)
    {
     s1[0]=(int)*(ptr+i);
     s2[0]=(int)*(ptr2+i);
  jj=memcmp(s1,s2,1);

    if(jj){
            draw_border(14,15,17,50,0);
             draw_border(13,14,16,49,33);
              write_string(14,15,"error found in loaded program ...!",33);
               write_string(15,15,"reload ? [Press Y/y to reload]    ",33);
               a=getch();
               switch(a){
               case 'y': goto reload1;
               case 'Y': goto reload1;
   reload1:  draw_bgr();
               goto reload;

               default :
                draw_bgr();
                 goto backtomain;}
             }
     }

   draw_border(14,10,16,53,0);
    draw_border(13,9,15,52,63);
     write_string(14,10,"     program successfully loaded ...!     ",63);
      delay(1000);
       draw_bgr();

   outportb(0x2fc,0);   /* enable buffer TMS */
   outportb(0x27c,0);
   delay(5);
    outportb(0x27e,0);   /* activate TMS  */
```

```
  draw_border(14,20,16,44,0);
   draw_border(13,19,15,43,63);
    write_string(14,20,"TMS32010 activated ...!",63);
     delay(1000);
      draw_bgr();

  draw_border(14,12,16,52,0);
  draw_border(13,11,15,51,63);
   write_string(14,12,"System Ready ...! [Press Q/q to Cancel]",63);

reproses:
  do{
   if(kbhit())
     {
          a=getch();
          switch(a){
           case 'q':
             outportb(0x27c,0);       /* reset TMS  */
             outportb(0x2fe,0);       /* enable buffer PC */
             goto back2;

           case 'Q':
             outportb(0x27c,0);       /* reset TMS  */
             outportb(0x2fe,0);       /* enable buffer PC */
  back2: draw_bgr();
             goto backtomain;

          default : break;}
     }
   }
   while(count==0);

   count=0;                  /* reset count to 0 */
    fclose(fp1);
     draw_bgr();

   for(i=0;i<10;i++){
     datcek[i]=*(ptr4+i);
      jum=jum+datcek[i];}
   if(jum==0){
  draw_border(14,12,17,52,0);
   draw_border(13,11,16,51,33);
    write_string(14,12,"        an error happened...          ",33);
     write_string(15,12," System can not continue the process ! ",33);
      delay(1000);
          draw_bgr();
           goto backtomain;}

   vali=*(stat);
   if(vali==1){
   draw_border(14,13,17,52,0);
    draw_border(13,12,16,51,33);
     write_string(14,13,"System can not Receice valid data ...!",33);
      write_string(15,13,"          Please Repeate !          ",33);
         delay(1000);
          draw_bgr();
          draw_border(14,18,16,46,0);
          draw_border(13,17,15,45,63);
           write_string(14,18," Restarting the process...!",63);
            delay(1000);
             goto reload;}

   draw_border(7,13,9,52,0);
   draw_border(6,12,8,51,63);
    write_string(7,13,"        Processing Data...!          ",159);
     delay(1500);

proseslagi:
/* ambil data dari A000:0 */
  j=-1;
```

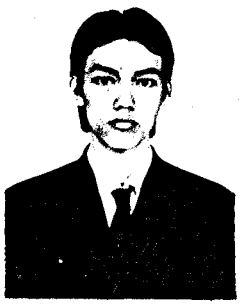
```
  for(i=0;i<16;i=i+1){
  j++;
/* Low byte */
    trans[i]=*(ptr4+i);
     valtrans=trans[i];
     sem[j]=((valtrans & 0x0080)/128)*3.125e-2;
          valtrans=trans[i];
          sem[j]=sem[j]+((valtrans & 0x0040)/64)*1.562e-2;
           valtrans=trans[i];
           sem[j]=sem[j]+((valtrans & 0x0020)/32)*7.812e-3;
            valtrans=trans[i];
            sem[j]=sem[j]+((valtrans & 0x0010)/16)*3.906e-3;
             valtrans=trans[i];
             sem[j]=sem[j]+((valtrans & 0x0008)/8)*1.953e-3;
              valtrans=trans[i];
              sem[j]=sem[j]+((valtrans & 0x0004)/4)*9.705e-4;
               valtrans=trans[i];
               sem[j]=sem[j]+((valtrans & 0x0002)/2)*4.882e-4;
                valtrans=trans[i];
                sem[j]=sem[j]+(valtrans & 0x0001)*2.441e-4;

/* High byte
     valtrans=trans[i];
     sem[j]=sem[j]+((valtrans & 0x4000)/16384)*5.0e-1;
          valtrans=trans[i];
          sem[j]=sem[j]+((valtrans & 0x2000)/8192)*2.5e-1;*/
           valtrans=trans[i];
           sem[j]=sem[j]+((valtrans & 0x1000)/4096)*1.0;
            valtrans=trans[i];
            sem[j]=sem[j]+((valtrans & 0x0800)/2048)*0.5e-1;
             valtrans=trans[i];
             sem[j]=sem[j]+((valtrans & 0x0400)/1024)*0.25e-1;
              valtrans=trans[i];
              sem[j]=sem[j]+((valtrans & 0x0200)/512)*1.25e-1;
               valtrans=trans[i];
               sem[j]=sem[j]+((valtrans & 0x0100)/256)*6.25e-2;}

/* konversi data ref. */
  j=-1;
  for(i=0;i<16;i=i+1){
   j++;
/* Low byte */
    trans[i]=*(ptr3+i);
     valtrans=trans[i];
     ref[j]=((valtrans & 0x0080)/128)*3.125e-2;
          valtrans=trans[i];
          ref[j]=ref[j]+((valtrans & 0x0040)/64)*1.562e-2;
           valtrans=trans[i];
           ref[j]=ref[j]+((valtrans & 0x0020)/32)*7.812e-3;
            valtrans=trans[i];
            ref[j]=ref[j]+((valtrans & 0x0010)/16)*3.906e-3;
             valtrans=trans[i];
             ref[j]=ref[j]+((valtrans & 0x0008)/8)*1.953e-3;
              valtrans=trans[i];
              ref[j]=ref[j]+((valtrans & 0x0004)/4)*9.705e-4;
               valtrans=trans[i];
               ref[j]=ref[j]+((valtrans & 0x0002)/2)*4.882e-4;
                valtrans=trans[i];
                ref[j]=ref[j]+(valtrans & 0x0001)*2.441e-4;

/* High byte
     valtrans=trans[i];
     ref[j]=ref[j]+((valtrans & 0x4000)/16384)*5.0e-1;
          valtrans=trans[i];
          ref[j]=ref[j]+((valtrans & 0x2000)/8192)*2.5e-1;*/
           valtrans=trans[i];
           ref[j]=ref[j]+((valtrans & 0x1000)/4096)*1.0;
            valtrans=trans[i];
            ref[j]=ref[j]+((valtrans & 0x0800)/2048)*0.50e-1;
             valtrans=trans[i];
```

```
                ref[j]=ref[j]+((valtrans & 0x0400)/1024)* 0.25e-1;
                 valtrans=trans[i];
                 ref[j]=ref[j]+((valtrans & 0x0200)/512)* 1.25e-1;
                  valtrans=trans[i];
                  ref[j]=ref[j]+((valtrans & 0x0100)/256)* 6.25e-2;}

/* hitung perbedaan jarak dengan algoritma Pemrograman Dinamik */
/* hitung frame pertama */
   distot=0.0;
   i=1;
   j=0;
   for(k=1;k<5;k++){
   m=j+k;
   if(i!=m){
      d[i][m]=ref[i]-sem[m];
      g[i][m]=d[i][m];
      if(g[i][m]<0){g[i][m]=0-g[i][m];}}}

   for(k=0;k<4;k++){
   m=i+k;
   if(m!=j){
      d[m][j]=ref[m]-sem[j];
      g[m][j]=d[m][j];
      if(g[m][j]<0){g[m][j]=0-g[m][j];}}}

   g[1][1]=2*(ref[1]-sem[1]);
    for(p=1;p<5;p++){
     for(q=1;q<3;q++){
     err=ref[q+1]-sem[p];
     if(err<0){err=0-err;}
      distrans[1]=g[q+1][p-1]+err;
       if(distrans[1]<0){distrans[1]=0-distrans[1];}
             distrans[2]=g[q][p-1]+2*err;
              if(distrans[2]<0){distrans[2]=0-distrans[2];}
               distrans[3]=g[q][p]+err;
                if(distrans[3]<0){distrans[3]=0-distrans[3];}
    if(distrans[1]<distrans[2]){
     mins=distrans[1];goto hit1;}
      mins=distrans[2];

hit1:
    if(distrans[3]<mins){
    g[q+1][p]=distrans[3];goto hit2;}
    g[q+1][p]=mins;
hit2:
    distot=distot+g[q+1][p];}}

/* hitung frame selanjutnya */
    for(r=0;r<2;r++){
    i=1+r;
    j=1+r;
    for(k=0;k<5;k++){
    m=4*j+k;
    if(i!=m){
       d[4*i][m]=ref[4*i]-sem[m];
       g[4*i][m]=d[4*i][m];
       if(g[4*i][m]<0){g[4*i][m]=0-g[4*i][m];}}}

    for(k=0;k<5;k++){
    m=4*i+k;
    if(m!=j){
       d[m][4*j]=ref[m]-sem[4*j];
       g[m][4*j]=d[m][4*j];
       if(g[m][4*j]<0){g[m][4*j]=0-g[m][4*j];}}}

     for(p=1;p<5;p++){
     for(q=1;q<4;q++){
     err=ref[4*i+q]-sem[4*j+p];
     if(err<0){err=0-err;}
      distrans[1]=g[4*i+q][4*j+(p-1)]+err;
```

```
      if(distrans[1]<0){distrans[1]=0-distrans[1];}
            distrans[2]=g[4*i+(q-1)][4*j+(p-1)]+2*err;
             if(distrans[2]<0){distrans[2]=0-distrans[2];}
              distrans[3]=g[4*i+(q-1)][4*j+p]+err;
               if(distrans[3]<0){distrans[3]=0-distrans[3];}

   if(distrans[1]<distrans[2]){
     mins=distrans[1]; goto hit1a;}
          mins=distrans[2];

hit1a:
   if(distrans[3]<mins){
   g[4*i+q][4*j+p]=distrans[3]; goto hit2a;}
   g[4*i+q][4*j+p]=mins;
hit2a:
   distot=distot+g[4*i+q][4*i+p];}}}

/* compare distot to the threshold */
   if(distot>thld[inf]){
     if(jud==jum_data){
           draw_bgr();
            draw_border(14,19,16,45,0);
             draw_border(13,18,15,44,33);
              write_string(14,19,"Word not Recognized ....!",33);
                delay(1000);
                goto backtomain;}

    reld: jud=jud+1;
    cari: thru=findnext(&file);
              if(thru!=0){goto backtomain;}
               nama=file.ff_name;
                fp1=fopen(nama,"rb");
                jum=strcmp(nama,"COBA.DAT");
                if(jum==0){fclose(fp1); goto cari;}

     jum1=fread(ptr3,64,1,fp1);  /* load next data ref. */
     if(!jum1){
       draw_border(14,19,16,45,0);
            draw_border(13,18,15,44,33);
             write_string(14,19,"read next ref. error ...!",33);
               draw_bgr();
                fclose(fp1);
                 goto backtomain;
             }
             fclose(fp1);

  jum=strcmp(nama,"SATU1.DAT");
   if(jum==0){inf=0; goto lanjutkan1;}
    jum=strcmp(nama,"SATU2.DAT");
     if(jum==0){inf=1; goto lanjutkan1;}
      jum=strcmp(nama,"SATU3.DAT");
       if(jum==0){inf=2; goto lanjutkan1;}
            jum=strcmp(nama,"SATU4.DAT");
             if(jum==0){inf=3; goto lanjutkan1;}
              jum=strcmp(nama,"SATU5.DAT");
               if(jum==0){inf=4; goto lanjutkan1;}

  jum=strcmp(nama,"DUA1.DAT");
   if(jum==0){inf=5; goto lanjutkan1;}
    jum=strcmp(nama,"DUA2.DAT");
     if(jum==0){inf=6; goto lanjutkan1;}
      jum=strcmp(nama,"DUA3.DAT");
       if(jum==0){inf=7; goto lanjutkan1;}
            jum=strcmp(nama,"DUA4.DAT");
             if(jum==0){inf=8; goto lanjutkan1;}
              jum=strcmp(nama,"DUA5.DAT");
               if(jum==0){inf=9; goto lanjutkan1;}

  jum=strcmp(nama,"TIGA1.DAT");
   if(jum==0){inf=10; goto lanjutkan1;}
```

```
   jum=strcmp(nama,"TIGA2.DAT");
    if(jum==0){inf=11; goto lanjutkan1;}
    jum=strcmp(nama,"TIGA3DAT");
     if(jum==0){inf=12; goto lanjutkan1;}
          jum=strcmp(nama,"TIGA4.DAT");
            if(jum==0){inf=13;}

lanjutkan1:
             goto proseslagi;
            }

/* jika error lebih kecil dari threshold */
   if(inf<5){
    draw_bgr();
     draw_border(14,19,16,45,0);
      draw_border(13,18,15,44,43);
       write_string(14,19,"          Satu          ",63);
            goto goback;}
    if((inf>4) & (inf<10)){
    draw_bgr();
     draw_border(14,19,16,45,0);
      draw_border(13,18,15,44,43);
       write_string(14,19,"          Dua           ",63);
            goto goback;}
    if(inf>9){
    draw_bgr();
     draw_border(14,19,16,45,0);
      draw_border(13,18,15,44,43);
       write_string(14,19,"          Tiga          ",63);
            }

 goback:
   delay(1500);

 backtomain:
   free(ptr);
   draw_bgr();
   fp1=fopen("coba.dat","wb");
    jum1=fwrite(ptr4,64,1,fp1);
     if(!jum1){
     draw_border(14,22,16,43,0);
     draw_border(13,21,15,42,33);
      write_string(14,22,"Write ref. error...!",33);
       delay(1000);
        fclose(fp1);
             exit(0);
             }
             fclose(fp1);
   delay(100);
 }
```

DAFTAR RIWAYAT HIDUP



Muhammad Yosep dilahirkan di Palembang pada tanggal 7 April 1969, anak ke-lima dari sembilan bersaudara dari Bapak Musa Husdi dan Ibu Maisyaroh. Terdaftar sebagai mahasiswa Jurusan Teknik Elektro Fakultas Teknologi Industri ITS, pada tahun 1988 dengan nomor pokok registrasi 2882200938.

Pendidikan yang pernah ditempuh selama ini adalah:

- SDN No. 14 Palembang, lulus tahun 1982.
- SMPN No. 4 Palembang, lulus tahun 1985.
- SMA Xaverius I Palembang, lulus tahun 1988.
- Mahasiswa Jurusan Teknik Elektro Fakultas Teknologi Industri ITS sejak tahun 1988.

Selama menjadi mahasiswa pernah aktif sebagai asisten pada Praktikum Rangkaian Listrik, Praktikum Elektronika Dasar, dan Praktikum Elektronika Lanjut II.