SIFIR DOKUZ ARASI SAYILARIN DİNAMİK PROGRAMLAMA İLE TANINMASI

SALİH EREN

Yüksek Lisans Tezi
Elektrik Elektronik Mühendisliği
Anabilim Dalı
1992

SIFIR DOKUZ ARASI SAYILARIN DİNAMİK PROGRAMLAMA İLE TANINMASI

SALİH EREN

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektronik Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: Prof. Dr. Atila BARKANA

EYLÜL - 1992

Salih EREN'in YUKSEK LISANS tezi olarak hazırladığı "SIFIR DOKUZ ARASI SAYILARIN DINAMIK PROGRAMLAMA ILE TANINMASI " başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

5./10./1992

Uye: Prof. Dr. Atila BARKANA

Uye: Prof. Dr. Atalay BARKANA

Uye: Y. Doç. Dr. Osman PARLAKTUNA

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 07 EKIM 1992 gün ve 328-16 sayılı kararıyla onaylanmıştır.

Prof. Dr. Rüstem KAYA
Enstitü Müdürü

İÇİNDEKİLER

İÇİNDEKİLER (Devam)

ÖZET

Bilgisayar dünyası geliştikçe değişik bir çok işin bilgisayara yaptırılması isteği de yaygınlaşmaktadır. Bilgisayarlarla ses ve görüntü tanınmasıyla değişik amaçlı kontrol sistemleri kurulabilir.

Bu tez çalışmasında seslerın değişik frekanslardaki enerjilerine bakılarak oluşturulan şablonlar kullanılarak Dinamik Programlama yöntemi ile ses tanıma yoluna gidilmiştir.

Tanıma için kullanılan donanım iki bölümden oluşmaktadır. İlk bölüm bilgisayar ile analog ve dijital devreler arasında köprü görevi gören çoklu giriş-çıkış kartından ; ikinci bölüm ise ön yükselteç , ADC , DAC , filtreler ve seslendirme devresinden oluşmaktadır.

SUMMARY

With the developments in the computer world , the demand for getting a lot of different things done by computer is increasing. Various control systems which perform speech and pattern recognition by a computer can be designed.

In this thesis , speech recognition has been tried to be accomplished using Dynamic Programming Method which uses the patterns that are formed from the values of sound energies in different frequencies.

The hardware that is used for identification is composed of two sections. The first section is a multiple input-output card that has the bridge function between the analog and digital circuits of the computer ; the second section consists of the preamplifier , ADC , DAC , filters and amplifier circuit.

TEŞEKKÜR

Çalışmalarımda bana yardımcı olan danışmanım Prof. Dr. Atila Barkana'ya , olumlu eleştirilerinden dolayı Prof. Dr. Atalay Barkana , Arş. Grv. Umit Künkçü ve Elektronik Mühendisi Şerif Tosun'a teşekkür ederim.

Özellikle bana sabırla tahammül eden sözlüm Gönül Durmaz'a manevi katkılarından dolayı minnettarım.

ŞEKİLLER DİZİNİ

1. GİRİŞ

Ses ve ses tanıma üzerine yapılan çalışmalar pek yeni değildir. Tanıma algoritmaları genelde çok matematiksel hesap gerektirdiğinden bu çalışmalar ancak hızlı bilgisayarların gelişmesiyle pratiklik kazanmıştır. Ses tanıyıcı cihazlar çok geniş bir kullanım alanı olmasa da yavaş yavaş günlük yaşantımıza girmeye başlamıştır.

Elektronik teknolojisi ilerledikçe otomatik ses tanıyıcı cihazlardan istenilen özellikler de artmaktadır. Bu özelliklerden başlıcaları şunlardır:

1) Tanıma oranı yüksek olmalı ,

2) Sözcük sayısı yeteri kadar fazla olmalı ,

3) Sürekli konuşma halinde yeteri kadar hızlı çalışmalı,

4) Yeni girilecek kelimeler kolaylıkla girilebilmeli ,

5) Standart olmalı yani diğer PC bilgisayarlara kolayca bağlanabilmelidir.

Şimdiye kadar bu özellikleri tam olarak sağlayabilecek bir cihaz yapılamamış olmasının yanısıra tanıma oranı yüksek , fiyatı ucuz bir ses tanıma cihazı oldukça uzakta durmaktadır. Piyasada bulunan ses tanıma cihazları incelendiğinde kullanılan algoritma olarak ilk sırayı Dinamik Programlama almaktadır.

Bu tez çalışmasında bilinmeyen ses , daha önceden değişik frekanslardaki enerjilerine bakılarak oluşturulan şablonlarla Dinamik Programlama yöntemiyle karşılaştırmak suretiyle tanınmaya çalışılmıştır.

2. SES TANIMA SISTEMLERI TARIHÇESI

1930'lu yıllarda başlayan ses tanıma çalışmaları bu güne kadar artan bir hızda süregelmiştir. Ses tanıma çalışmalarında şimdiye kadar akustik , şablon tanıma ve pragmatik yaklaşımlar kullanılmıştır. Tanıma çalışmalarında kullanılan yöntemlerde farklılıklar olmakla beraber hepsinde de zaman alıcı matematiksel işlemler bulunmaktadır. Pragmatik yaklaşımda , izole edilmiş kelime tanıyıcılarında Dinamik Programlama Algoritması kullanılması büyük önem kazanmaktadır. İlk Dinamik Programlama Algoritması Velichko ve Zalgoruyko tarafından 1970 yılında kullanılmıştır. Bu sistem Rusça 200 kelimeyi % 95 oranla tanımaktadır [1]. Yine Sakoe ve Chiba tarafından hazırlanan algoritma kullanılarak Japonca dijitler için hata oranı % 0.2 olarak bulunmuştur. Yine bu algoritma kullanılarak Japonca 50 Coğrafik isim % 0.8 hata ile tanınmıştır [2].

Dijitlerin tanınmasında 10 erkek konuşmacıdan herbiri 5 örnekden oluşan 6 şablon alınmıştır. Coğrafik isimlerin tanınmasında ise 2 erkek 2 bayan konuşmacıdan 5'er örnek alınmıştir. Tanımada NEAC-3100 sistemi kullanılarak tanıma süresi dijitler için 3 sn , Coğrafik isimler için ise 30 sn olarak bulunmuştur [2].

3. DINAMIK PROGRAMLAMA

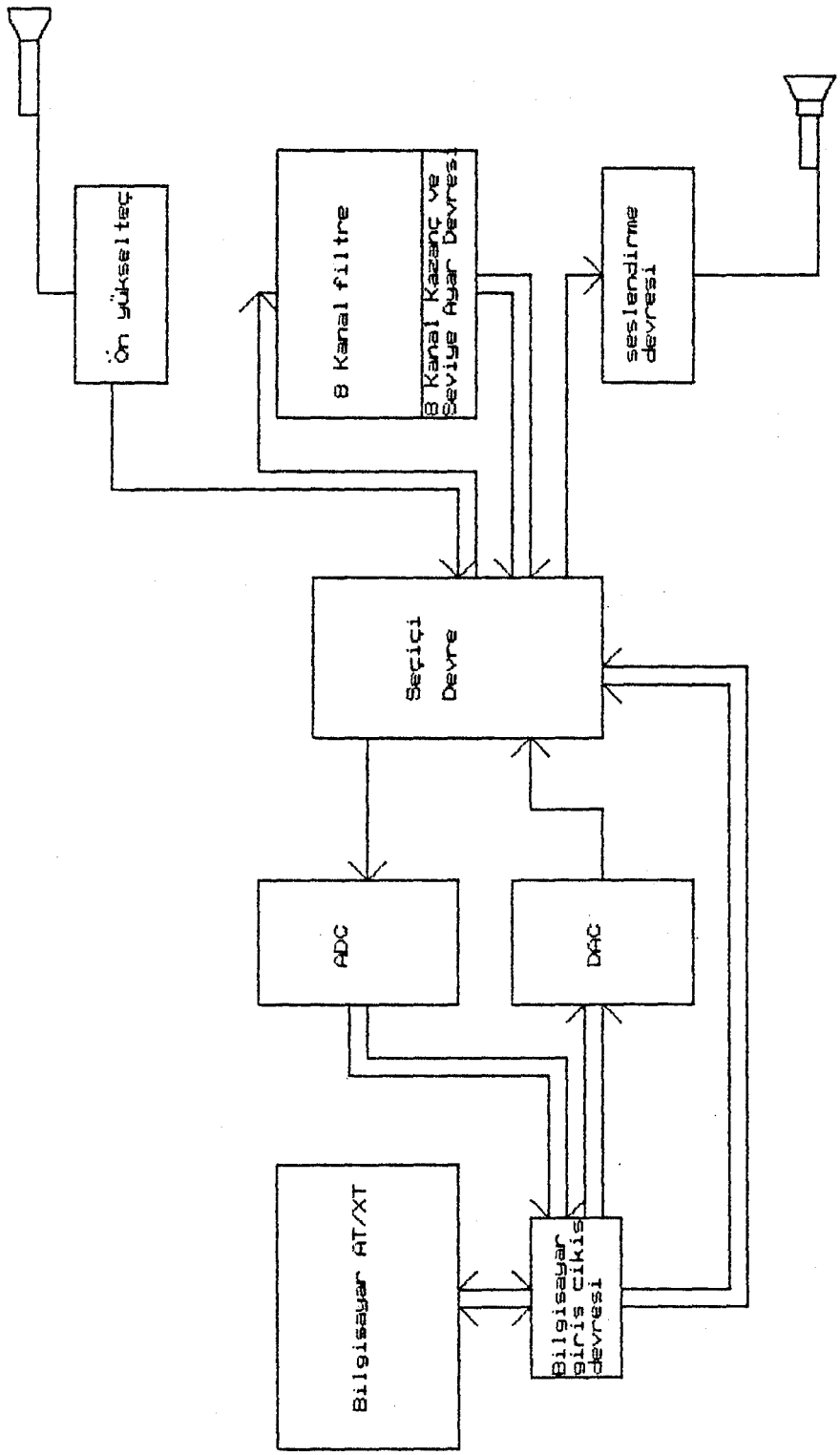
Bu tasarımda sesi tanımak için ADC , 8 band geçiren filtre ve DAC kullanılmıştır (Şekil 3.1.). Mikrofondan gelen ses sinyali ön kuvvetlendirici kısmında yükseltilerek ADC tarafından sayısal verilere dönüştürülür. Elde edilen sese ait veriler DAC aracılığı ile filtrelere yönlendirilir ve filtre çıktıları ADC tarafından tekrar örneklenir. Böylece her sese ait 8 boyutlu n elemanlı bir vektör elde edilir. Buradaki n sayısı sesin söyleniş uzunluğu olmak üzere her kelime ve kişi için değişik değerler almaktadır.

$$X_1 = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ \cdot \\ \cdot \\ x_{18} \end{bmatrix} , \quad X_2 = \begin{bmatrix} x_{21} \\ x_{22} \\ x_{23} \\ \cdot \\ \cdot \\ x_{28} \end{bmatrix} , \quad \ldots , \quad X_n = \begin{bmatrix} x_{n1} \\ x_{n2} \\ x_{n3} \\ \cdot \\ \cdot \\ x_{n8} \end{bmatrix}$$

Elde edilen X_i vektörleri daha önceden hazırlanmış şablon vektörlerle dinamik programlama algoritması kullanılarak karşılaştırılır. En yakın şablonun seçilmesiyle ses tanıması yapılabilir.

Konuşma sinyalinin genliği , zamanla farkedilebilir bir şekilde değişirken değişik kişiler arasında birtakım benzerlikler gösterir.

Dinamik Programlama ile karşılaştırma , doğrusal olmayan zaman-normalizasyon etkisiyle bir şablon uygunlaştırma algoritmasıdır. Bu algoritmada zaman ekseninin düzensiz değişimi , bazı özelliklerin doğrusal olmayan bir uygunlaştırma fonksiyonu (Warping Function) ile yaklaşık olarak biçimlendirilmektedir. İki konuşma şablonu arasındaki zamanlama farklılıkları bir diğeriyle maksimum uyum gösteren şablonun zaman ekseninin uygunlaştırılması ile ortadan kaldırılır. Zaman normalizasyonlu uzaklık , birbirleri arasında gözlenen uzaklığın minimize edilmesiyle hesaplanır.



Şekil 3.1. Uyum Devresi

Kullanılan ses kartı , konuşma bilgilerini 10 KHz örnekleme hızında örnekleyerek oluşturur. Örnek değerleri -2048 ile +2047 arasında (12 bit) değerler alabilir. Bu değerler çerçevelere ayrılarak işleme alınır.

Kesikli zaman sinyalinin enerjisi

$$E = \sum_{m=-\infty}^{+\infty} x^2(m)$$

olarak tanımlanır. Ancak bu ifade yerine kısa-zaman enerjisi kullanmak ses sinyali hakkında yeterli bilgiyi verdiğinden

$$E = \sum_{m=n-N+1}^{n} x^2(m)$$

ifadesi kullanılmaktadır. Bu çalışmada hesaplanacak aralık bir çerçeve uzunluğu alınmıştır. Her çerçeve için toplam enerji hesaplanarak ortalaması alınmış ve sonuç dB'e çevrilmiştir. Bu durumda enerji şiddeti fonksiyonu

$$E_{dB} = 10\text{-}Log_{10}\left(\sum_{m=1}^{sn} x^2(m)/sn \right)$$

ile verilir. Buradaki sn değeri çerçeve içindeki örnek sayısıdır.

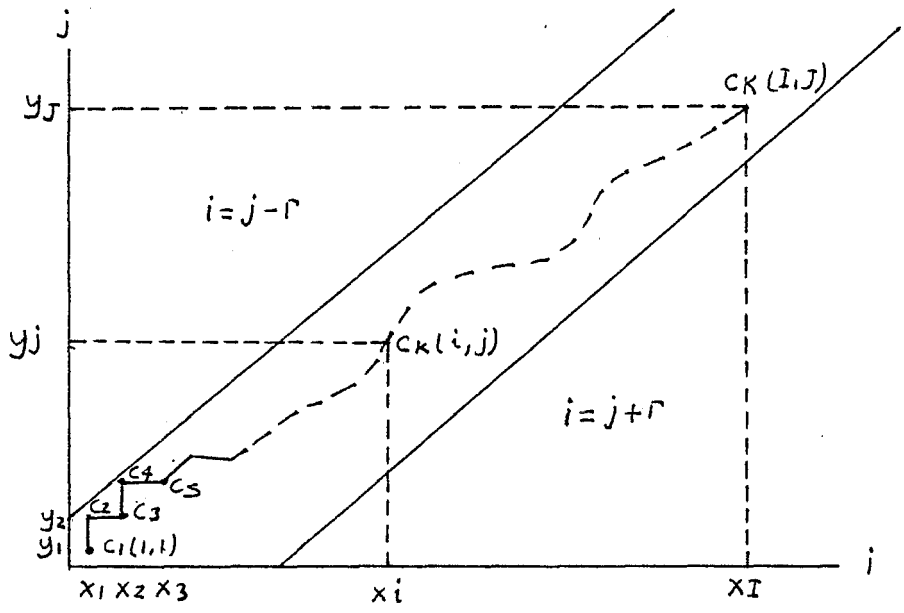
Konuşma , uygun özelliklerin özeti olan bir dizi vektörle ifade edilebilir:

$$X = x_1, x_2, \ldots, x_i, \ldots, x_I$$
$$Y = y_1, y_2, \ldots, y_j, \ldots, y_J$$

Bu vektörler Şekil 3.2.'deki gibi i-j düzlemine yerleştirildiğinde bu iki vektörün arasındaki zamanlama farkları c(i,j) noktalarının bir dizisiyle tanımlanabilir.

$$F = c(1) , c(2) , \ldots , c(k) , \ldots , c(K)$$

burada c(k)=(i(k),j(k)) dır. Bu dizi X'in zaman ekseninin Y'nin üzerine yaklaşık olarak çakıştırılmasından dolayı ortaya çıkan bir fonksiyon olarak düşünülebilir.



Şekil 3.2. (n,m) noktasından orijine en kısa mesafeli gidiş yolu

X ve Y vektörleri arasındaki farklılık , bir uzaklık olarak

$$d(c) = d(i,j) = \|x_i - y_j\|$$

şeklinde ölçülür. Böylece uygunlaştırma fonksiyonu üzerindeki uzaklıkların ağırlıklı toplamları

$$E(F) = \sum_{k=1}^{K} d(c(k)) \cdot w(k)$$

biçiminde yazılabilir. Burada w(k) negatif olmayan bir katsayıdır. E(F) ölçümünün esnekliğini sağlamak için ortaya atılmıştır. Optimal zaman farklılığı uygunlaşmasında E(F)'nin değeri minimum olur.

X ve Y gibi iki konuşma şablonu arasındaki zaman normalizasyonlu uzaklık

$$D(X,Y) = \min_{F} \left[\frac{\sum_{k=1}^{K} d(c(k)) \cdot w(k)}{\sum_{k=1}^{K} w(k)} \right]$$

olarak tanımlanır. Eğer w(k) katsayıları uygunlaştırma fonksiyonundan bağımsız ise son ifade için:

$$D(X,Y) = \frac{1}{N} \min_{F} \left[\sum_{k=1}^{K} d(c(k)) \cdot w(k) \right]$$

yazılabilir. Burada N=I+J dir. Başlangıç durumu

$$g_1(c_1)) = d(c(1)) \cdot w(1)$$

alındığında

$$g_k(c(k)) = \min_{c(k-1)} \left[g_{k-1}(c(k-1)+d(c(k)) \cdot w(k) \right]$$

yazılır ve zaman normalizasyonlu uzaklık

$$D(X,Y) = \frac{1}{N} \cdot g_k(c(k))$$

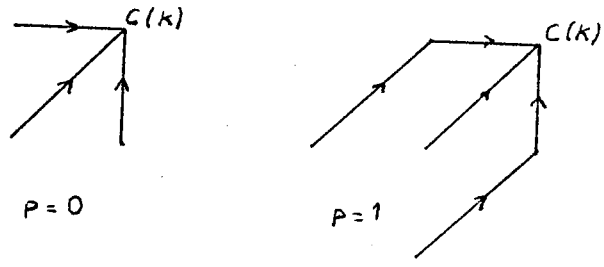
olarak ifade edilir.

Uygunlaştırma fonksiyonunun daha optimal bir şekilde seçilebilmesi için bazı kısıtlamalar kullanılabilir. Bunlar uygunlaştırma fonksiyonunun hareket edebileceği alanın sınırlanması ve bir noktadan ancak bazı noktalara hareket edebileceğinin belirlenmesidir. Hareket alanı ayarlanabilir bir pencere durumuyla sağlanır.

$$(j-r) < i < (j+r)$$

Buradaki r parametresi denemeler sonunda optimal bir şekilde bulunacaktır.

Hareket noktaları eğim sınırlaması ile de belirlenir. Şekil 3.3.'de de görüldüğü gibi hareket noktalarına iki türlü eğim kısıtlaması (p) getirilebilir.



Şekil 3.3. Eğim Kısıtlaması

Bu kısıtlamalarla birlikte;

p=0 için:

$$g_1(i,j) = \min \begin{bmatrix} g(i-1,j)+d(i,j) \\ g(i-1,j-1)+\alpha*d(i,j) \\ g(i,j-1)+d(i,j) \end{bmatrix}$$

ve p=1 için:

$$g1(i,j) = \min \begin{bmatrix} g(i-1,j-2)+\alpha*d(i,j-1)+d(i,j) \\ g(i-1,j-1)+\alpha*d(i,j) \\ g(i-2,j-1)+\alpha*d(i-1,j)+d(i,j) \end{bmatrix}$$

elde edilir. Buradaki α sabitinin optimum değeri bilinmemekte ancak denemelerle uygun bir atama yapılmaktadır [1] [2] [9].

4. DONANIM

4.1. GIRIŞ

Bu sistemde kullanılan uyum devresi AT/XT bilgisayarları ile çalışabilecek biçimde tasarlanmıştır. Devre bilgisayar hafızasına kaydettiği ses sinyallerini istendiği durumda tekrar üretebilmektedir. Yine kaydedilen ses sinyallerinin dalga şekli çizilebilmekte ve üzerinde inceleme yapmak mümkün olabilmektedir.

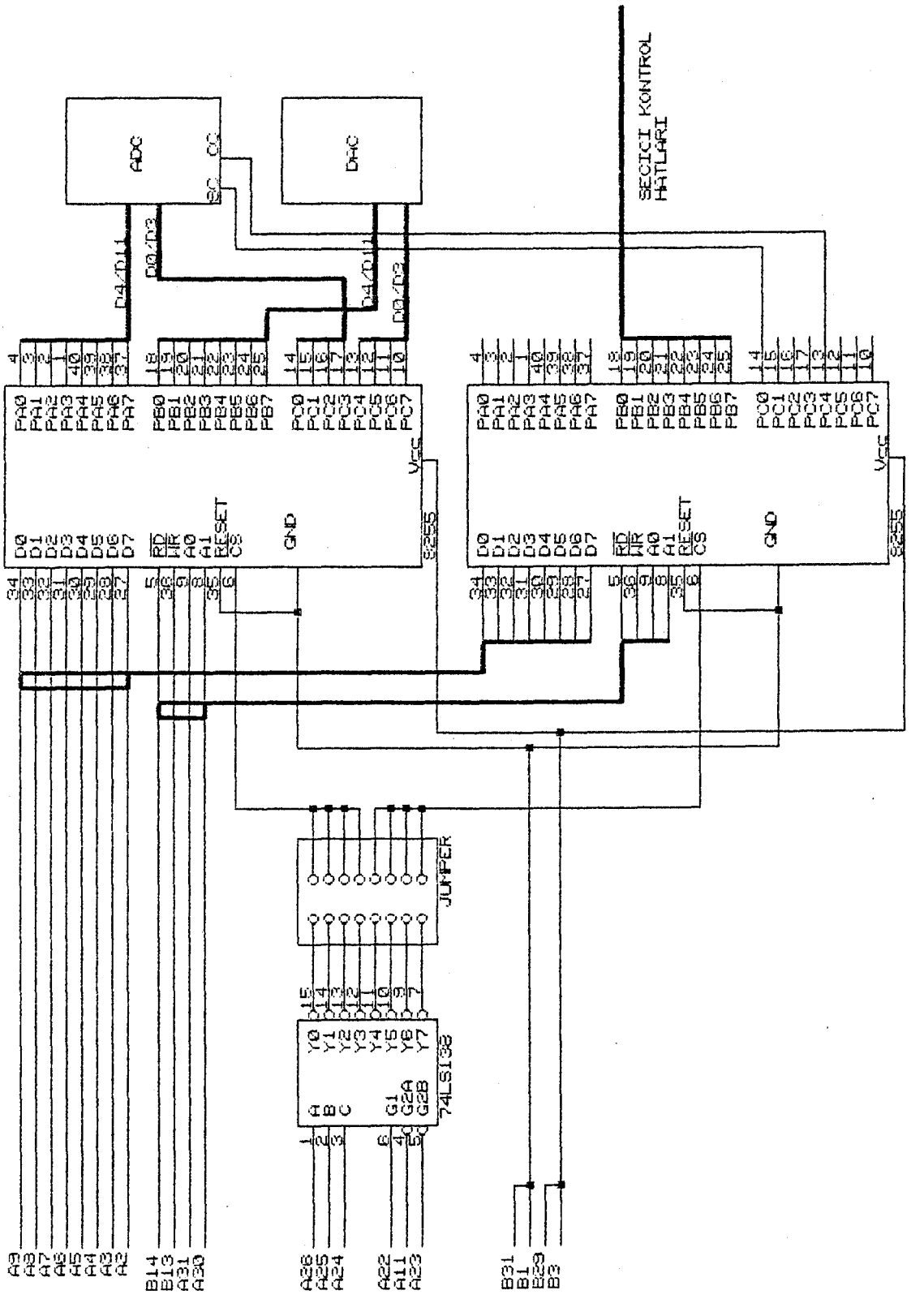
Sistem esas olarak 8255A PPI , ADC1210 , DAC1210 , adres çözücü , frekans üreteci , gerilim üreteci , band geçiren filtre , ön kuvvetlendirici ve kuvvetlendiriciden oluşmaktadır.

Aşağıda sistemdeki herbir devre ayrıntılı olarak açıklanmaktadır.

4.2. BİLGISAYAR GİRİŞ-ÇIKIŞ DEVRESİ

Sistemin bilgisayarla bağlantısı bir paralel giriş - çıkış devresiyle gerçekleştirilmiştir. Bu devre üzerinde kod çözücü (74HC138) , iki adet PPI (8255A) ve adres seçiciden (dip switch) oluşmuştur. Devre şeması Şekil 4.1.'de görülmektedir.

74HC138 entegresi girişine gelen bilgiye göre (3'den 8'e) aktif sıfır vermektedir. Bu özellikten yararlanarak PPI 'ların seçilmesi mümkün olmuştur. Adres seçici (dip switch konumu) ise devrenin değişik adres bölgelerinde çalışabilmesini sağlamaktadır. ADC'den gelen bilgiler 8255A tarafından bilgisayara alınmakta ve yine 8255A aracılığıyla DAC'a gönderilmektedir. Ayrıca ADC ve analog anahtarlar için de yine 8255A kullanılmıştır. Aşağıda 8255A entegresinin yapısı ve programlanmasıyla ilgili bilgi verilerek sistemdeki kullanımı anlatılmıştır.



Şekil 4.1. Bilgisayar Giriş-Çıkış Devresi

4.2.1. 8255 Yapısı ve Programlanması

8255 Intel firması tarafından mikroişlemci ile dış devreler arasında bağlantı sağlamak amacıyla geliştirilmiş bir karşılıklı uyum devresidir. Her biri 8'er bitlik A ve B , 4'er bitlik C_L , C_H adlı dört portu ve yine 8 bitlik veri yolu bulunmaktadır. Mod 0 , Mod1 , Mod2 olmak üzere üç çalışma modu bulunmaktadır. 8255'i değişik modlarda ve portların I/O olarak kullanılması kontrol portuna gönderilen bilgi ile gerçekleşmektedir. Aşağıda portların seçilme ve kontrol portuna gönderilmesi gereken bilgiyi veren tablolar Şekil 4.2. ve 4.3.'de verilmiştir.

A_1	A_0	RD	WR	CS	İŞLEM
0	0	0	1	0	Port A-Veri Yolu
0	1	0	1	0	Port B-Veri Yolu
1	0	0	1	0	Port C-Veri Yolu
0	0	1	0	0	Port A-Veri Yolu
0	1	1	0	0	Port B-Veri Yolu
1	0	1	0	0	Port C-Veri Yolu
1	1	1	0	0	Kontrol-Veri Yolu
X	X	X	X	1	8255 Seçilmiyor
1	1	0	1	0	Uygunsuz Durum

Şekil 4.2. 8255 Port Kontrol Tablosu

D_7	Mod Set Flag
D_6D_5	Mod 00=Mod 0 01=Mod 1 1X=Mod 2
D_4	Port A 0=Giriş 1=Çıkış
D_3	Port C_H 1=Giriş 0=Çıkış
D_2	Mod 0=Mod 0 1=Mod 1
D_1	Port B 1=Giriş 0=Çıkış
D_0	Port C_L 1=Giriş 0=Çıkış

Şekil 4.3. 8255 Kontrol Register Tablosu

4.2.2. 8255'in Sistemdeki Kullanımı

Bilgisayar giriş - çıkış devresinde iki adet 8255A bulunmaktadır. Seçilen adres bölgesi 0200h-02E0h arasındadır. Komut yazaç adresleri birinci 8255A için 0200h , ikinci 8255A için ise 0280h seçilmiştir. İlgili portların kullanımı aşağıya çıkarılmıştır.

Birinci 8255 : Port A_0-A_7 ADC1210 (D_4-D_{11})
Port C_0-C_3 ADC1210 (D_0-D_3)
Port B_0-B_7 DAC1210 (D_4-D_{11})
Port C_4-C_7 DAC1210 (D_0-D_3)

İkinci 8255 : Port C_0 SC
Port C_4 CC
Port B_0-B_7 4051 Seçici kontrol hatları
Diğer bitler kullanılmıyor.

Bu durum Şekil 4.1.'de görülmektedir.

4.3. ÖN YÜKSELTEÇ

Değişik özelliklere sahip olmasından dolayı SGS-Ates firmasının 16 bacaklı TDA1054 entegresi kullanılmıştır. Bu entegrenin dört temel işlevi vardır:

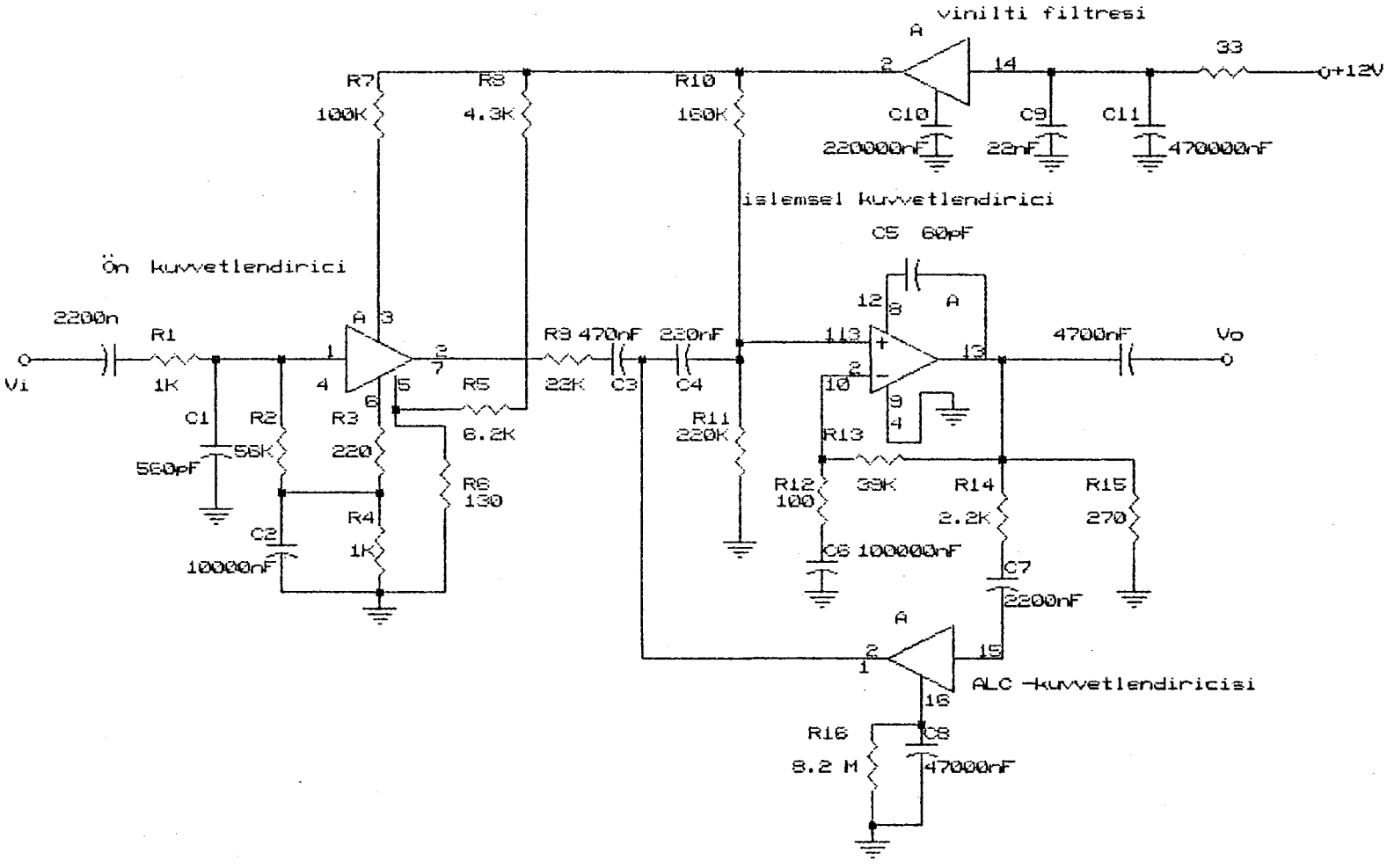
1 - Ön kuvvetlendirici,
2 - Yıniltı filtresi,
3 - İşlemsel kuvvetlendirici,
4 - Otomatik kazanç kontrolu (OKK).

1- Ön kuvvetlendirici doğrudan bağlamalı iki transistörden oluşmaktadır. Bu tür bir yapıyla değişik geri beslemelerin yapılmasına olanak sağlanmaktadır.

2- Yıniltı filtresi , besleme katından gelen doğru gerilimin üzerindeki dalgalanmaları zayıflatır.

3- İşlemsel kuvvetlendirici fark kuvvetlendiricisiyle doğrudan bağlamalı kazanç katından oluşmaktadır.

4- Entegrenin OKK'lı asıl kuvvetlendiricisi çıkış işaret seviyesinin sabit tutulmasının istendiği durumlarda kullanılır.



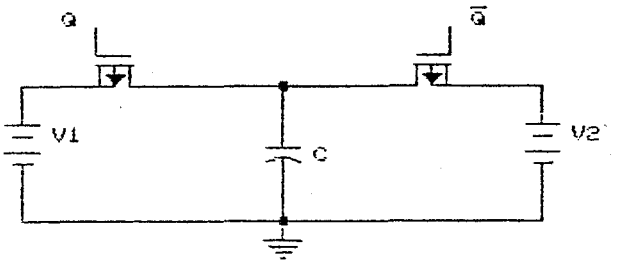
Şekil 4.4. Ön Yükseltec Devresi

4.4. SEKİZ KANAL BAND GEÇİREN FİLTRE DEVRESİ

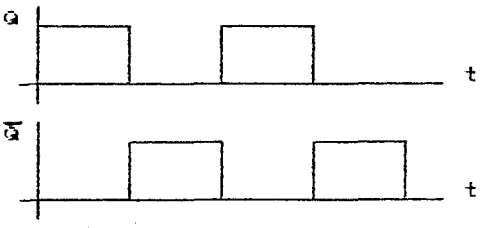
4.4.1. Anahtarlanmış Kapasitör Filtre Devreleri

4.4.1.1. Anahtarlanmış Kapasitör

SC Filtrelerde kullanılan en temel yapıdır. Bu tür yapılar istenilen değerlerin %1 hassasiyetle elde edilmesine imkan sağlamaktadır. Anahtarlama işlemi MOS anahtarlarıyla gerçekleştirilmiştir.



Şekil 4.5.
FET Anahtarlı Kapasitör



Şekil 4.6.
FET Saat ve Saat Darbeleri

Şekil 4.5.'de kapasitör V_1 değerine şarj olurken V_2 değerinden deşarj olacaktır. Böylece transfer edilen net yük için

$$dQ=C*dV$$

yazılabilir. Burada

$$dV=V_1-V_2$$

olduğu kabul edilsin. Saat sinyalinin etkisi ile:

$$dQf_{saat}=C*dVf_{saat}$$

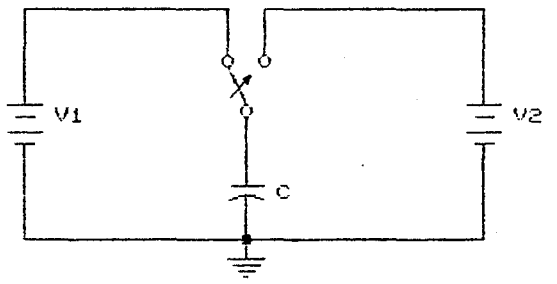
yazılabilir. Akan akım

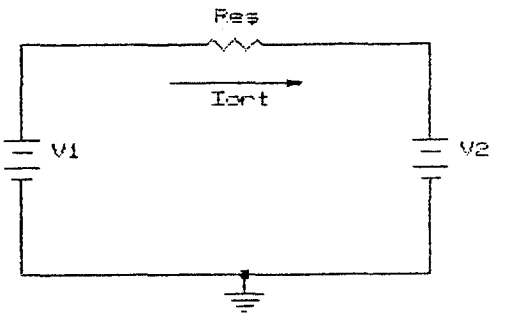
$I_{ort}=dQ/dT$ ve $dT=1/f_{saat}$ ise

sonuç olarak simüle devresinden geçen akım için

$$I_{ort}=Qf_{saat}=CVf_{saat}$$

yazılabilir.





Şekil 4.7. Şekil 4.8.

FET Anahtarlı Kapasitörün Basit Gösterimi ve Eşdeğeri

Şekil 4.7.' de görüldüğü gibi toprağa iki gerilim farkı ile orantılı bir akım akmaktadır. Şekil 4.8. Şekil 4.7.'nin bir modelidir. Aynı akımın toprağa aktığı kabullenilerek iki gerilim arasına direnç konulmuştur.

Şekil 4.8.' den:

$R_{eş}=(V_1-V_2)/I_{ort}$

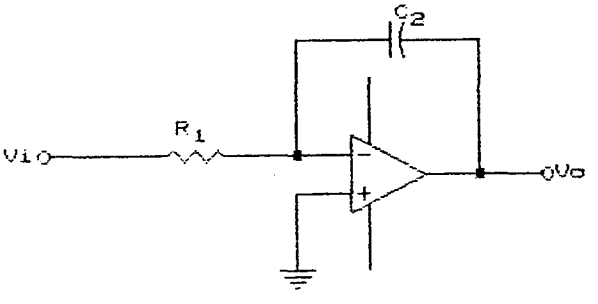
Önceki denklemlerden:

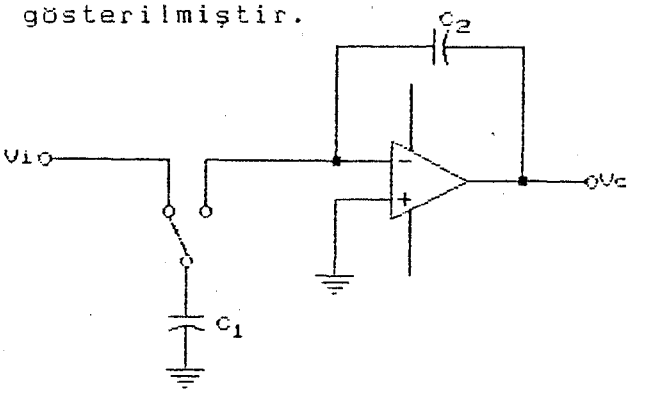
$R_{eş}=(V_1-V_2)/(\ Cf_{saat}(V_1-V_2)\ )$

$R_{eş}=\ 1\ /\ (Cf_{saat})$ yazılabilir.

4.4.1.2. Anahtarlı İntegratör

Anahtarlamalı kapasitör normal RC elemanlarından oluşan bir integratörün direnci yerine konulduğunda integratörün merkez frekansının saat sinyaline bağımlı olduğunu gösterir. Bu durum şekil ve denklemlerle gösterilmiştir.





Şekil 4.9. Şekil 4.10.

İntegratör Devresi Anahtarlı İntegratör Devresi

Şekil 4.9.'daki integratör devresi için düğüm denklemi yazılırsa:

$$\frac{0 - V_i}{R_1} + \frac{0 - V_o}{1/(sC_2)} = 0$$

$s=j2\pi f$ ve $fo=1/2\pi R_1C_2$ olmak üzere:

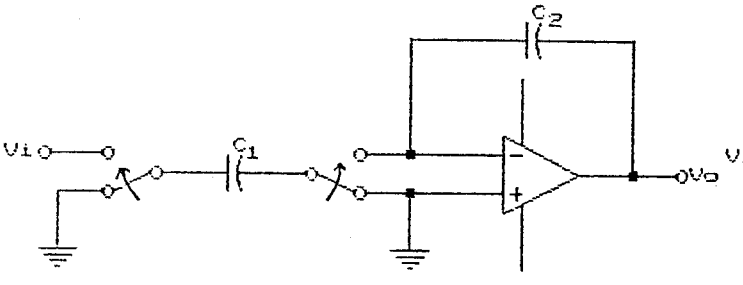
$$\frac{V_o}{V_i} = \frac{1}{jf/f_o}$$ elde edilir.

Şekil 4.10.'daki devre daha önceki denklemler yardımıyla Şekil 4.9.'daki devreye uyarlanırsa:

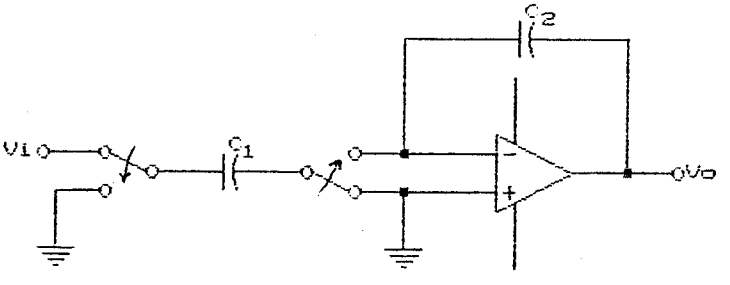
$$R_{eş} = \frac{1}{C_1 f_{saat}}$$ yazılarak $$f_o = \frac{C_1 f_{saat}}{2\pi C_2}$$

olarak bulunur. Bu denklem merkez frekansının saat sinyalinin frekansına bağlı olduğunu gösterir.

4.4.1.3. Faz Döndüren ve Döndürmeyen İntegratör



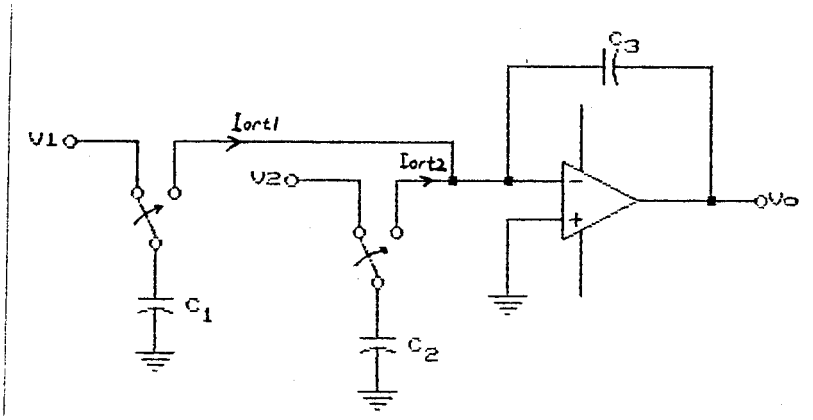
Şekil 4.11.
Faz Döndüren İntegratör



Şekil 4.12.
Faz Döndürmeyen İntegratör

Şekil 4.11. faz döndüren bir anahtarlı devredir. Şekil 4.12. anahtarların uygun durumunda giriş sinyali C_1 kapasitesi üzerinde gözükür. Anahtarların yer değiştirmesiyle C_1 kapasitesi üzerinde sinyal ters işaretli olarak çıkışa yansır. Her iki durum giriş sinyalinin (+) veya (-) referanslarına göre Şekil 4.11. faz döndüren , Şekil 4.12. faz döndürmeyen integratör olarak çalışır.

4.4.1.4. Toplayıcı İntegratör



Şekil 4.13. Toplayıcı İntegratör

Akım kanunundan:

$$I_{ort} = I_{ort1} + I_{ort2} = C_1 f_{saat} V_1 + C_2 f_{saat} V_2$$

Önceki akım eşitlikleri kullanılırsa:

$$V_o = -(1/jwC_3) I_{ort} = -(1/jwC_3)(C_1 f_{saat} V_1 + C_2 f_{saat} V_2)$$

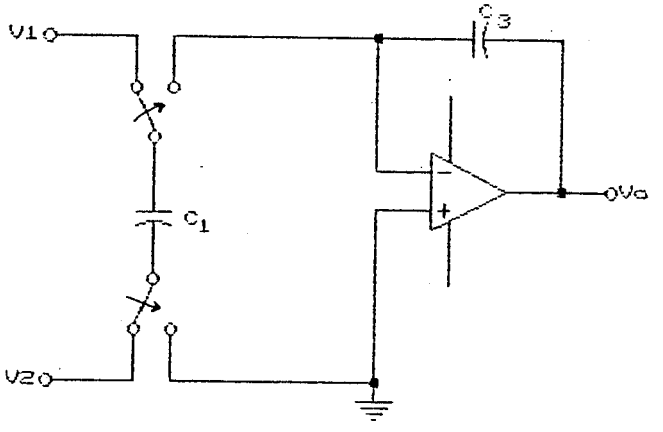
elde edilir.

$$f_1 = \frac{C_1 f_{saat}}{2\pi C_3} \quad \text{ve} \quad f_2 = \frac{C_2 f_{saat}}{2\pi C_3}$$

olmak üzere

$$V_o = -\frac{1}{jf/f_1} V_1 - \frac{1}{jf/f_2} V_2 \quad \text{bulunur.}$$

4.4.1.5. Fark Alıcı İntegratör



Şekil 4.14.
Fark Alıcı İntegratör

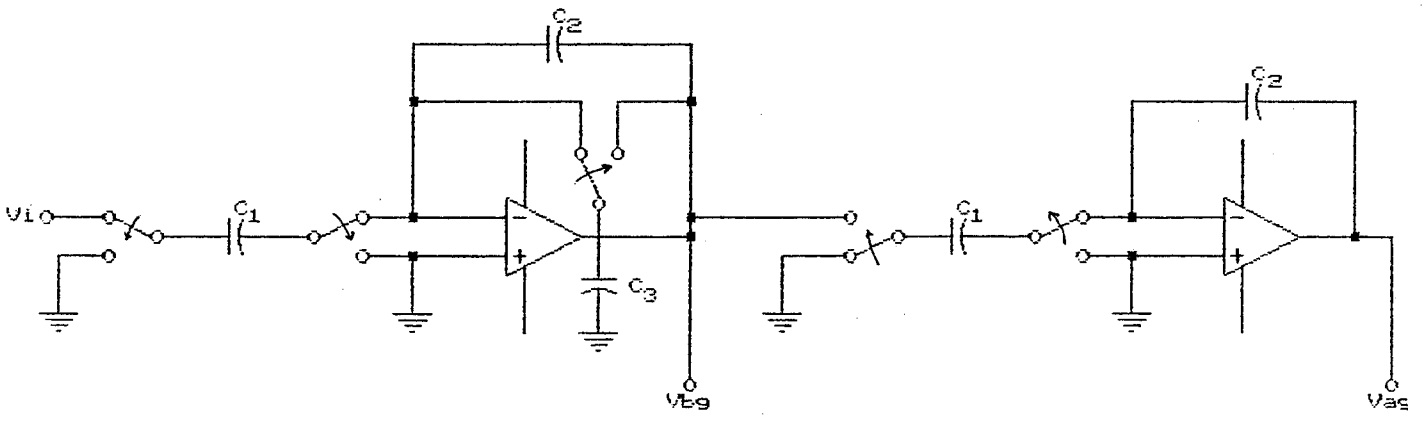
Önceki eşitliklerden:

$I_{ort}=f_{saat}C_1(V_1-V_2)$ yazılırsa

$V_o=-(1/jwC_3)f_{saat}(V_1-V_2)C_1$ elde edilir.

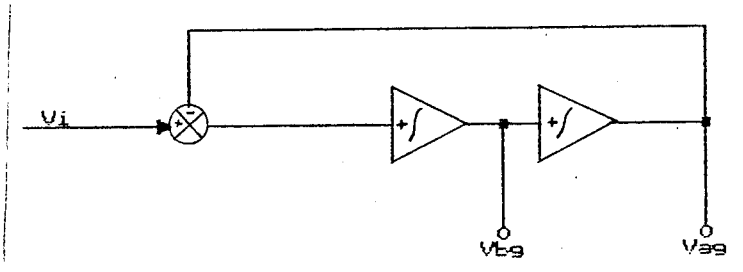
$f_o=\frac{C_1 f_{saat}}{2\pi C_3}$ olmak üzere $V_o=\frac{1}{jf/f_o}(V_1-V_2)$ bulunur.

4.4.1.7. 2.Dereceden Anahtarlamalı Filtre Devresi



Şekil 4.15. 2. Dereceden Anahtarlamalı Filtre

Şekil 4.15.' deki devrenin blok şeması Şekil 4.16.'da gösterilmiştir. Denklemlerle 2. dereceden band geçiren fonksiyonun formülü elde edilmiştir.



Şekil 4.16. 2. Dereceden Anahtarlamalı Filtrenin Blok Şeması

Şekil 4.16.'da Vbg'nin integrali Vag'yi verir:

$V_{ag}=w_o V_{bg}/s$ yazılabilir.

Öte yandan:

$V_{bg}=w_o(V_i - V_{ag})/s$

olarak yazılır.

ve sonuç olarak:

$$\frac{V_{bg}}{V_i} = \frac{sw_o}{s^2+w_o^2}$$

yazılabilir.

4.4.2. Filtrelerin Genel Özellikleri

Filtreler ses işlemlerinde kullanılmak üzere tasarlanmıştır. Her bir filtre merkez frekansı ve band aralığı değiştirilebilir özellige sahiptir. Filtreler MF10 entegreleriyle oluşturulmuştur. Bu entegrenin kullanılmasının nedeni filtrelerin iyi performans özelliği göstermesidir. Diğer işlemsel kuvvetlendirici ve filtre entegreleriyle oluşturulan filtrelerde çıkışta kayma , ofset ve gürültü meydana gelirken MF10 entegreli filtrelerde bu durum en aza indirgenmiştir.

4.4.3. MF10 Anahtarlı Kapasitör Filtrelerin Genel Özelikleri

Genel amaçlı CMOS aktif filtreyi kullanmak oldukça basittir. MF10 içinde iki tane birbirinden bağımsız filtre bloğu bulunmaktadır. Her blok 2.dereceden fonksiyonlar üretmektedir. Çıkışlardan biri yüksek band geçiren , band geçirgen veya band yutan fonksiyonu iken kalan ikisi alçak band geçiren fonksiyonudur. 4. dereceden fonksiyonlar 2.dereceden blokların arka arkaya bağlanmasıyla oluşturulmaktadır.

4.4.4. MF10 Bacak Açıklamaları

(1,20),(2,19),(3,18) : Birincisi 2.dereceden alçak geçiren , ikincisi band geçiren , üçüncüsü ise moduna göre band yutan , band geçiren veya yüksek band geçiren filtre çıkışlarıdır.

(4,17) : Her bir filtrenin işlemsel kuvvetlendiricisinin faz döndüren girişleridir. Bu girişler yüksek empedans özelliği gösterir.

(5,16) : Tüm geçirgen filtre durumlarında giriş bacağı olarak kullanılır.

(6): Bu bacak AGND ve alçak geçirgen çıktısı arasında bir anahtardır. Bu bacağa V+ uygulandığında alçak geçirgen tarafına , V- uygulandığında ise AGND'ye bağlanmaktadır.

(7,8) : Analog ve dijital pozitif besleme kaynak bacaklarıdır.

(10,11) : Anahtarlı kapasitör filtrelerin saat girişleridir. Saat frekansı 200 KHz üstünde olduğunda duty saykıl %50'ye yakın olmalıdır. Bu durum filtrenin içindeki işlemsel kuvvetlendiricinin maksimum zamana set edilmesini sağlayarak tam filtre imkanı sağlar.

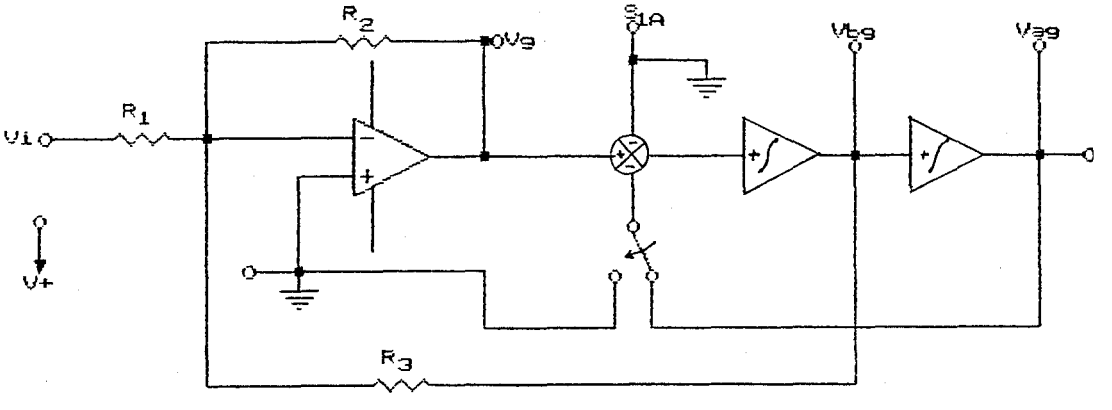
(14,13) : Analog ve dijital negatif besleme girişleridir.

(9) : Seviye kaydırma bacağıdır. Sistem toprağına bağlanmalıdır.

(12) : Bu bacak f_{saat}/f_o oranı 50 olduğunda V+'ya , 100 olduğunda GND'ye bağlanmalıdır. Bu bacak V- olduğunda sistem faaliyeti durarak devre 2.5 mA'den az akım çeker.

(15) : Analog GND bacağıdır. Tek kaynak besleme değerinin yarısı uygulanmalıdır.

4.4.5. MF10 Mod 1 Formül Çıkarımı



Şekil 4.17. MF10 Entegresinin Mod 1 Blok Şeması

Şekil 4.17.'den aşağıdaki denklemler yazılabilir:

$$V_g = -\frac{R_2}{R_1} V_i - \frac{R_2}{R_3} V_{bg} \quad ,$$

$a=R_2/R_1$ ve $b=R_2/R_3$ olsun.

Üte yandan

$$V_{bg} = \frac{1}{jf/f_o} (V_g - V_{ag})$$

yazılabilir. Buradan:

$$V_{bg} = \frac{w_o}{s} (V_g - V_{ag})$$

elde edilir. Benzer şekilde

$$V_{ag} = \frac{1}{jf/f_o} V_{bg} = \frac{w_o}{s} V_{bg}$$

ve V_g son denklemde kullanılarak

$$V_{bg} = \frac{w_o}{s} (-aV_i - bV_{bg} - \frac{w_o}{s} V_{bg})$$

yazılır. Denklem düzenlenerek:

$$V_{bg} \left(1 + \frac{w_o^2}{s^2} + b \frac{w_o}{s} \right) = -\frac{aw_o}{s} V_i$$

buradan

$$\frac{V_{bg}}{V_i} = \frac{-aw_o s}{s^2 + sbw_o + w_o^2}$$

elde edilir.

$$Q=R_3/R_2$$

olduğu görülebilir.

$$H = a\, w_o = \frac{R_2}{R_1} w_o = \frac{R_2}{R_1} \frac{R_3}{R_2} w_{BA} = \frac{R_3}{R_1} w_{BA}$$

Burada

$$w_{BA} = \frac{w_o}{Q} \quad , \qquad w_o = Q\, w_{BA} = \frac{R_3}{R_2} w_{BA}$$

biçiminde elde edilir.

H ifadesi yerine koyulursa:

$$\frac{V_{bg}}{V_i} = - \frac{R_3/R_1 \; w_{BA} \; s}{s^2 + sbw_o + w_o^2}$$

$$= - \frac{R_3}{R_1} \; \frac{w_{BA} \; s}{s^2 + s(R_2/R_3)Qw_{BA} + w_o^2}$$

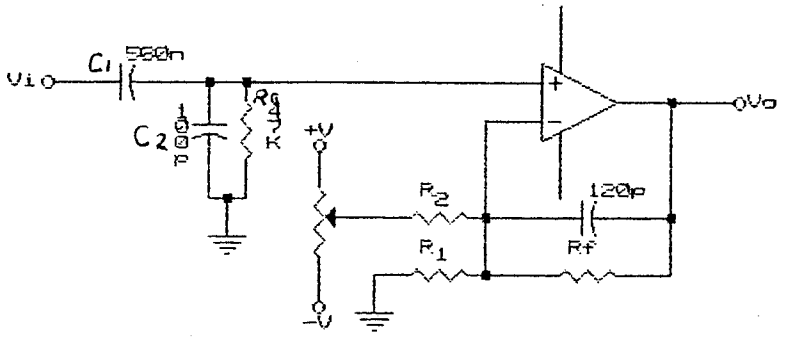
olarak yazılabilir. Sonuç olarak:

$$\frac{V_{bg}}{V_i} = - \frac{R_3}{R_1} \; \frac{W_{BA} \; S}{S^2 + S \; W_{BA} + W_o^2} = H_{obg} \; H_{bg} \qquad \text{ve}$$

$$H_{obg} = - R_3 \; / \; R_1$$

olarak bulunur.

4.5. SEKİZ KANAL KAZANÇ VE SEVİYE AYAR DEVRESİ



Şekil 4.18. Seviye ve Kazanç Ayar Devresi

Yukarıdaki devre sekiz kanaldan birini oluşturmaktadır. C_1 , C_2 , R_g elemanlarıyla şehir şebekesinden gelen 50 Hz'lik dalgalanmalar bastırılmakta diğer elemanlarla çıkışa seviye ve kazanç sağlanmaktadır. Aşağıda 2.durum formüllerle basit biçimde gösterilmiştir.

Düğüm denklemi yazılırsa:

$$\frac{V_i - V_o}{Z} + \frac{V_i}{R_1} + \frac{V_i - V_p}{R_2} = 0 \qquad Z=(120p)//R_f$$

$$V_i \left(\frac{1}{Z} + \frac{1}{R_1} + \frac{1}{R_2} \right) - \frac{V_p}{R_2} = \frac{V_o}{Z}$$

Sonuç olarak:

$$V_o = K_1 V_i - K_2 V_p$$

elde edilir.

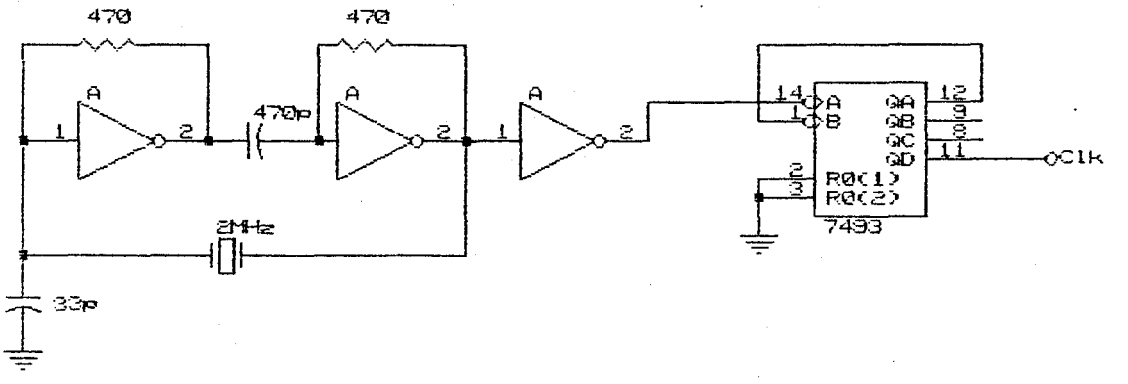
4.6. ADC-DAC ve SEÇICI DEVRE

4.6.1. ADC 1210 Yapısı

Analog sinyali sayısala çevirip bilgisayar hafızasına atmak amacıyla kullanılmıştır. ADC 1210 13 saat darbesinde bir çevrim yapmaktadır. Orta hızlı bir entegredir. Maksimum çevirme hızı 100 mikro saniyedir. Bu nedenle en fazla 10 KHz'de çalışabilir. Bu da örnekleme teoremine göre ($f_s = 2f_o$) 5 KHz'lik sinyal örnekleri alınabilecegini göstermektedir. Projede 5 KHz'e kadar olan ses sinyalleri ile uğraşıldığından ADC 1210'nun kullanılması uygundur. Bağlantı şeması Şekil 4.21'de gösterilmiştir.

4.6.1.1. Saat ve Bölucü Devresi

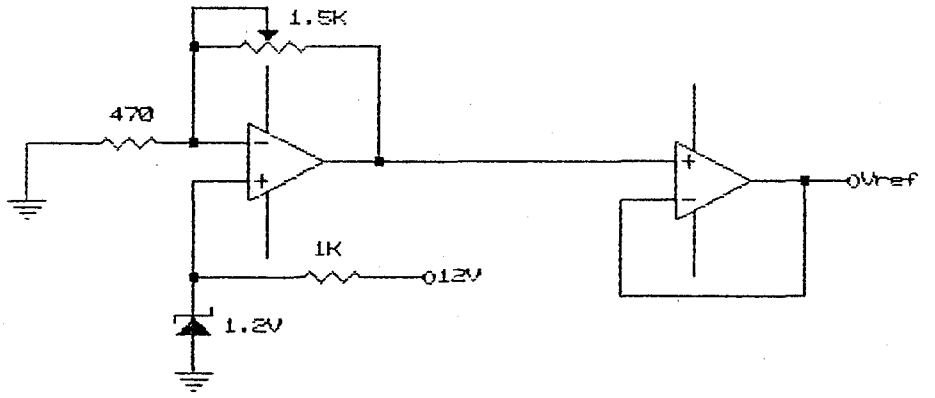
Bu tür bir yapı ADC için sabit bir saat frekansı üretir. Bu durum ADC'den alınan sayısal bilgilerin aynı saat frekansına göre alınmasını sağlamaktadır.



Şekil 4.19. ADC Saat ve Bölucü Devresi

4.6.1.2. ADC-DAC Referans Gerilim Devresi

ADC-DAC için gerekli referans gerilimi Şekil 4.20.'deki devre ile gerçekleştirilmiştir. Bu tür bir yapının devrede kullanılmasının nedeni sistem çalışırken referans geriliminin değişmesini önlemektir. Ayrıca referans devresinin ayarlı olması kullanıcıya istenilen referanslarda çalışma kolaylığı sağlamaktadır.



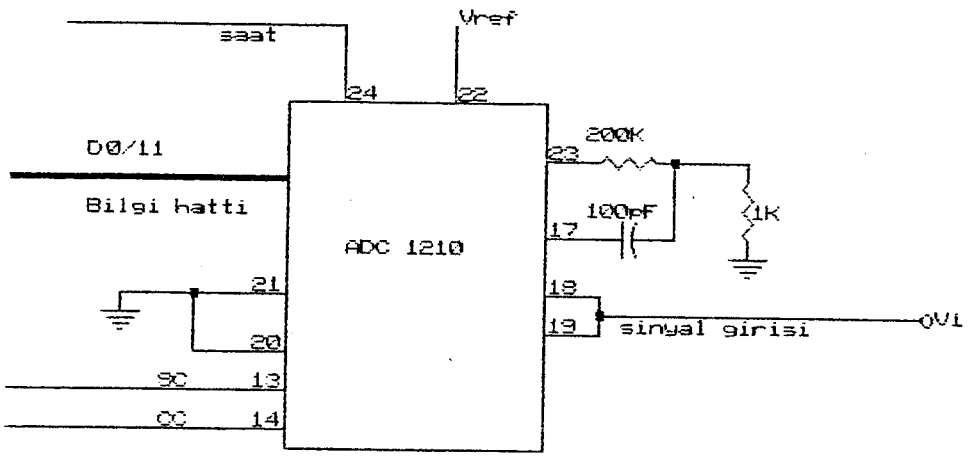
Şekil 4.20. ADC-DAC Referans Gerilim Devresi

4.6.2. DAC 1210 Yapısı

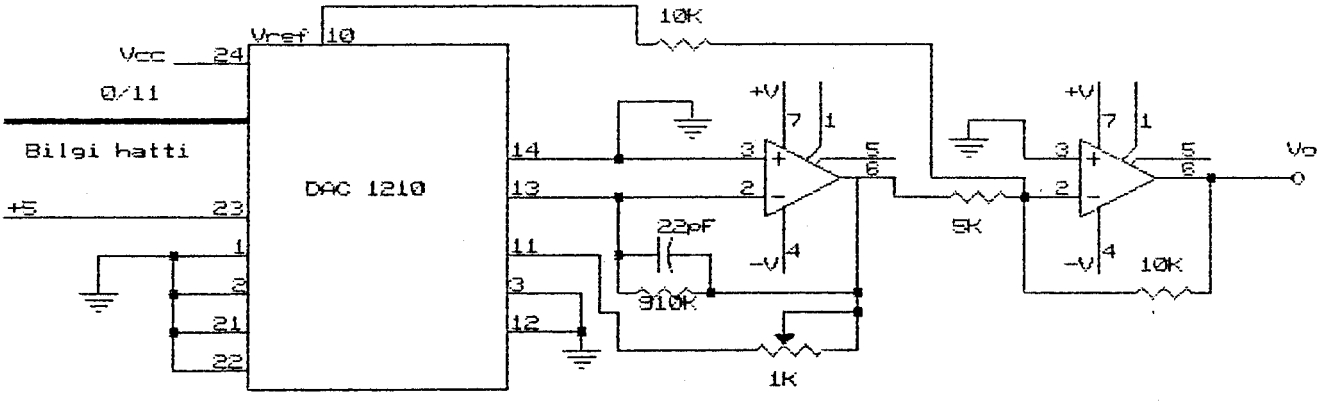
Sayısal sinyali yeniden analog sinyale çevirmek amacıyla kullanılır. DAC 1210 12 bitlik sayısal bilgiyi akım olarak 1 mikro saniyelik zamanda analog sinyale çevirme özelliğine sahiptir. İçinde 8 ve 4 bitlik ayrı ayrı kontrol edilebilen yazaçlar bulunmaktadır. Bu tür bir yapı kullanıcıya programlamada kolaylık sağlamaktadır. DAC çıkışı akım çıkışı olduğundan akımdan gerilime çevirici devre kullanılması zorunluluğu vardır. Bu durum Şekil 4.22.' de görülmektedir.

4.6.3. Sinyal Seçici Devre (4051 Multiplexer-Demultiplexer)

4051 entegresi ön yükselteç , filtre çıktıları , ADC ve DAC arasında ses sinyallerinin iletimini sağlamak amacıyla kullanılmıştır. V_{dd}'ye 5 volt , V_{ss}'ye 0 volt ve V_{ee}'ye -5 volt uygulandığında +5 ile -5 volt arasındaki sinyallerin entegreden geçişi gerçekleşmektedir.



Şekil 4.21. ADC1210 Bağlantı Şeması

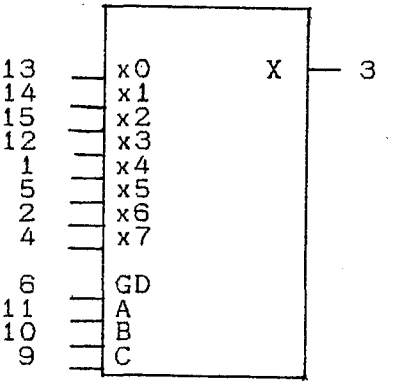


Şekil 4.22. DAC1210 Bağlantı Şeması

Şekil 4.23.'de A,B,C seçici kodlara gelen bilgiye göre aktif çıkış bacak numarası , Şekil 4.24.'de de entegrenin bacak bağlantısı görülmektedir.

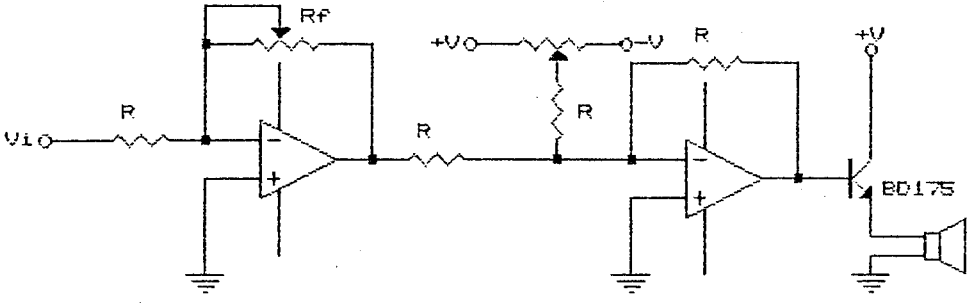
Giriş Durumları				
GD	C	B	A	Çıkış
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	x	x	x	Hiçbiri

Şekil 4.23.
Seçici Kontrol Tablosu



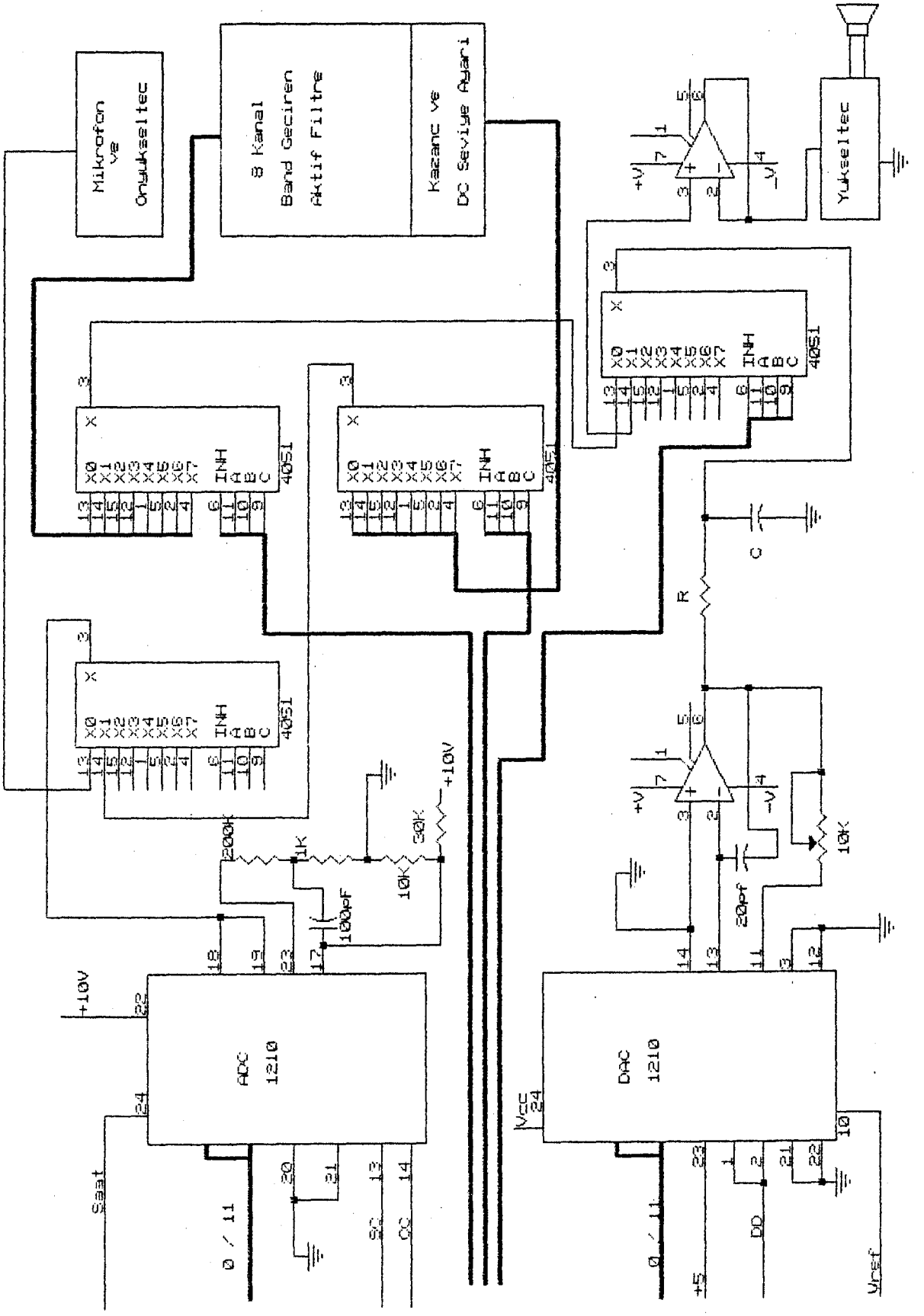
Şekil 4.24.
4051 Entegresi

4.7. SESLENDİRME DEVRESİ



Şekil 4.25. Seslendirme Devresi

Bu devreden beklenen bilgisayara depo edilen ses bilgilerinin doğruluk derecesinin insan kulağıyla kontroludur. Bilgisayardaki ses bilgilerini yeniden duyabilmek için şekil 4.25.'deki devre kullanılmıştır. Devreye gelen ses sinyali 1.OP-AMP'de belli bir kazanç , 2.OP-AMP'de ise dc seviye kazanarak transistöre gitmektedir. Ses sinyalleri transistörle kuvvetlendirilerek hoparlöre verilmektedir. Bu basit devreyle DAC'dan alınan ses sinyallerinin duyulması gerçekleştirilmiştir.



Şekil 4.26. Sistemin Açık Şeması

5. YAZILIM

Bu bölümde yazılımla ilgili esaslar üzerinde durulacaktır. Yapılan cihaz AT/XT uyumlu olduğu için herhangi bir bilgisayarla kolayca çalıştırılabilir. Cihazla bilgisayar arasındaki veri transferi 80x86 derleyicilerinden biriyle yapılabilir. Ayrıca Turbo Pascal 6.0 veya Turbo C++ derleyicileri ile de kolayca veri transferi yapmak mümkündür. Yazılım sesin kaydı , dinlenmesi , filtrelenmesi bir kütüğe kaydedilerek geri okunması , dalga şekillerinin çizilmesi ve tanınması bölümlerinden oluşur.

Programların tamamı Turbo Pascal V6.0'da yazılmıştır. Böylece makine dili ile diğer programlar arasında kolay bir iletişim sağlanmıştır.

Program klavyeden SPEECH.EXE yazılarak çalıştırılır. Program çalıştığında ekrana çıkan menüye göre işlemler yapılabilir. Kursor yukarı ve aşağı ok ile istenilen seçeneğin üzerine getirilerek Enter tuşuna basılır. Böylece istenilen seçenek seçilmiş olur. Program çalışmaya başladığında ekranda aşağıdaki seçenekler görünür:

SAMPLING
LISTENING
FILTERING
WRITE TO FILE
GET FILE
DRAW
RECOGNITION
PRINTER
QUIT

SAMPLING:

Mikrofondan ses kaydı yapmak için kullanılır. Bu seçenekten sonra herhangi bir tuşa basılarak 1.5 saniye süre ile ses kaydı yapmak mümkündür. İstenildiği takdirde bu süre arttırılabilir. Hafızası 1 MByte olan bir bilgisayarın 0.5 MByte bölümü program tarafından kullanılıyorsa diğer bölüm için maksimum 50 sn örnekleme yapılabilir. Bilgisayarın hafızası arttırılarak bu süre daha da arttırılabilir.

LISTENING:

Kaydedilen sesi hoparlörden dinlemek için kullanılır. Bu seçenekten sonra orijinal sesin mi yoksa filtre çıktısının mı dinleneceği 'F' tuşuna basılarak belirtilmesi gereklidir. 'F' tuşuna basılmaz ise kaydedilen orijinal sesi dinlemek mümkün olacaktır.

FILTERING:

Kaydedilen sesin filtrelenmesi için kullanılır. Bu seçenekten sonra sorulan filtre kanal numarası girilmelidir. Filtrelenmiş sesin de hoparlörden dinlenmesi mümkündür.

WRITE TO FILE:

Bilgisayarın hafızasına kaydedilen sesi diskete bir kütük adı altında kaydetmek için kullanılır. Kaydetmeden önce sorulan sorulara yanıt verilir. Kütüğün başına bu bilgiler eklenerek ileride kullanım kolaylığı sağlanmış olmaktadır.

GET FILE:

Daha önceden kaydedilmiş verileri ilgili kütükten alarak bilgisayarın hafızasına yazar. Kütüğün başında bulunan önemli bilgiler (Konuşmacı ismi , komut , orijinal mı yoksa filtrelenmiş mi ve söylenilen komutun hangi frame'ler arasında olduğu) ekranda görünür.

DRAW:

Hafızaya kaydedilen ses sinyalinin enerjisini ekrana çizmek için kullanılır. Filtre çıktıları çizilecekse 'F' tuşuna basılmalıdır. Dalga şekli çizildikten sonra ekrana çıkan kursor sağa ve sola hareket ettirilerek sesin dalga şeklinin istenilen bölümünün dinlenmesi mümkündür. İstenildiği takdirde bu görüntüler Print Screen tuşu ile yazıcıya da gönderilebilir.

RECOGNITION:

Bu bölüm Unit olarak tanımlanmıştır. Hafızaya kaydedilen ses ile 'C:\TEMPLATE' bölümünde bulunan şablonlar Dinamik Programlama yöntemiyle karşılaştırılarak ses tanıma yapılır.

PRINTER:

Bazı parametre ve sesin dalga şeklinin yazıcıdan alınması için kullanılır.

QUIT:

Programdan çıkmak için kullanılır. Bu tuşa basıldıktan sonra tüm portlar girişe programlanır. Bilgisayar park programı çalıştırıldıktan sonra kapatılarak devrenin güç kaynağı da kapatılır.

Yazılımla ilgili akış şemaları bölüm sonunda verilmiştir.

5.1. Ses Kaydı

Bilgisayara ses kaydının nasıl yapıldığını anlatmadan önce ADC'nin nasıl çalıştığını incelemek gerekir:

ADC1210 entegresinin $\overline{SC}$ bacağına 1/125 kHz'lik bir süre boyunca lojik-0 uygulandıktan sonra ADC1210 , o anda girişine gelen sinyalin sayısal değerini hesaplamaya başlar. Bu anda $\overline{CC}$ sinyali lojik-1 seviyesine yükselir. Tam 13 saat darbesi sonunda $\overline{CC}$ lojik-0 olur. Çevrim sonucu bulunan sayı yeni bir $\overline{SC}$ sinyali gelinceye kadar ADC1210 üzerinde gözükecektir. Bundan sonra da ilgili porttan sayısal değerler okunur. ADC1210 ile ilgili akış diyagramı ve programlar ekler bölümünde verilmiştir. ADC'den alınan örnekler SAMPLE adlı adresten başlamak üzere önce 8 , sonra 4 bitlik bölümü olmak üzere saklanır. SAMPLE için 60 kByte'lık bir hafıza ayrılırsa bu hafızaya 30,000 örnek değeri saklanabilir. İyi bir örnekleme için saniyede 10,000

örnek alınırsa toplam olarak 3 sn. süre ile ses kaydı yapmak mümkündür. İstenirse derleyicinin hafıza modeli değiştirilerek bu süre arttırılabilir.

5.2. Ses Üretilmesi

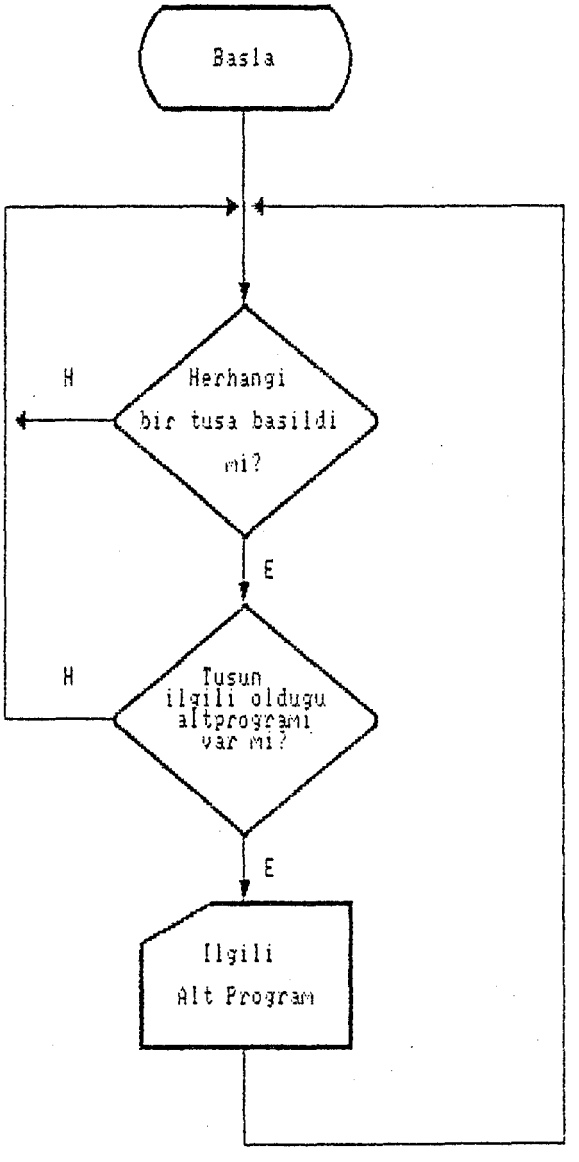
DAC1210 girişine gelen 12 bitlik sayıya göre çıkışında (uygulanan referans gerilimine bağlı olarak) bir gerilim üretir. DAC'a yollanacak sayılar yine SAMPLE adresinden itibaren sıralanmıştır. Yollanacak sayılar arasında örnekleme için kullanılan gecikme zamanı kullanılmalıdır. DAC çıkışı seçici anahtar ile ses kuvvetlendiricisine ya da filtrelere yönlendirilebilir.

5.3. Sesin Filtrelenmesi

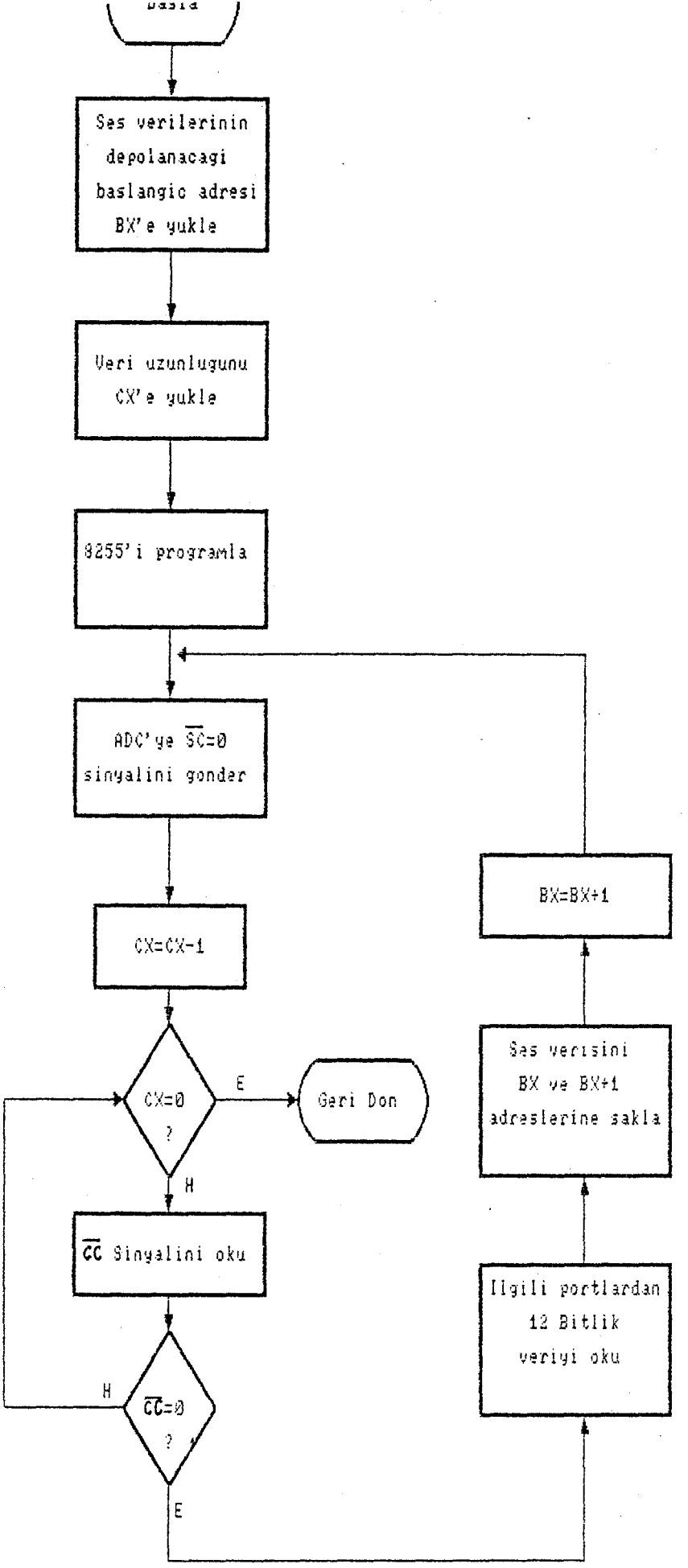
ADC'den alınan bilgiler DAC aracılığı ile analog işaretlere dönüştürülürken filtre bankasına yönlendirilir. DAC'a her bir örnek değeri yollandıktan sonra ADC aracılığı ile filtre çıktısı örneklenir. Bu işlem herbir kanal için tekrarlanır. Filtrelerin kazancı merkez frekansı ve band genişliği donanım olarak kolayca değiştirilebilmektedir. Böylece çok yönlü bir filtreleme işlemi yapmak mümkündür.

5.4. Sesin Tanınması

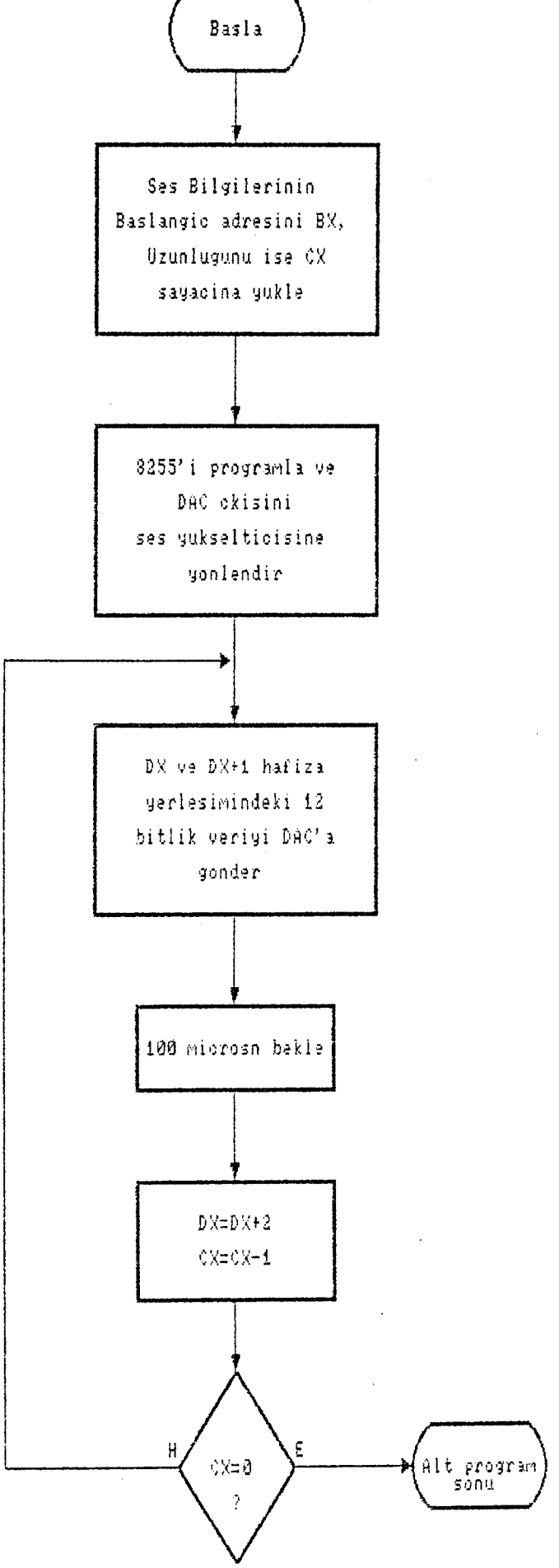
Sistemin tanıma sırasında kullandığı şablon vektörler PATTERN.EXE programı tarafından hazırlanmaktadır. Bu program 'C:\SAMPLE' adlı bölüme SPEECH.EXE programı tarafından kaydedilmiş ve kütük ismi olarak da komutları kabul eden kütükleri okuyarak şablon vektörleri oluşturur ve 'C:\TEMPLATE' bölümüne kaydeder. Kaydedilen bilgiler Text olarak kaydedildiği için diğer kullanıcılar tarafından da kolayca kullanılabilir.



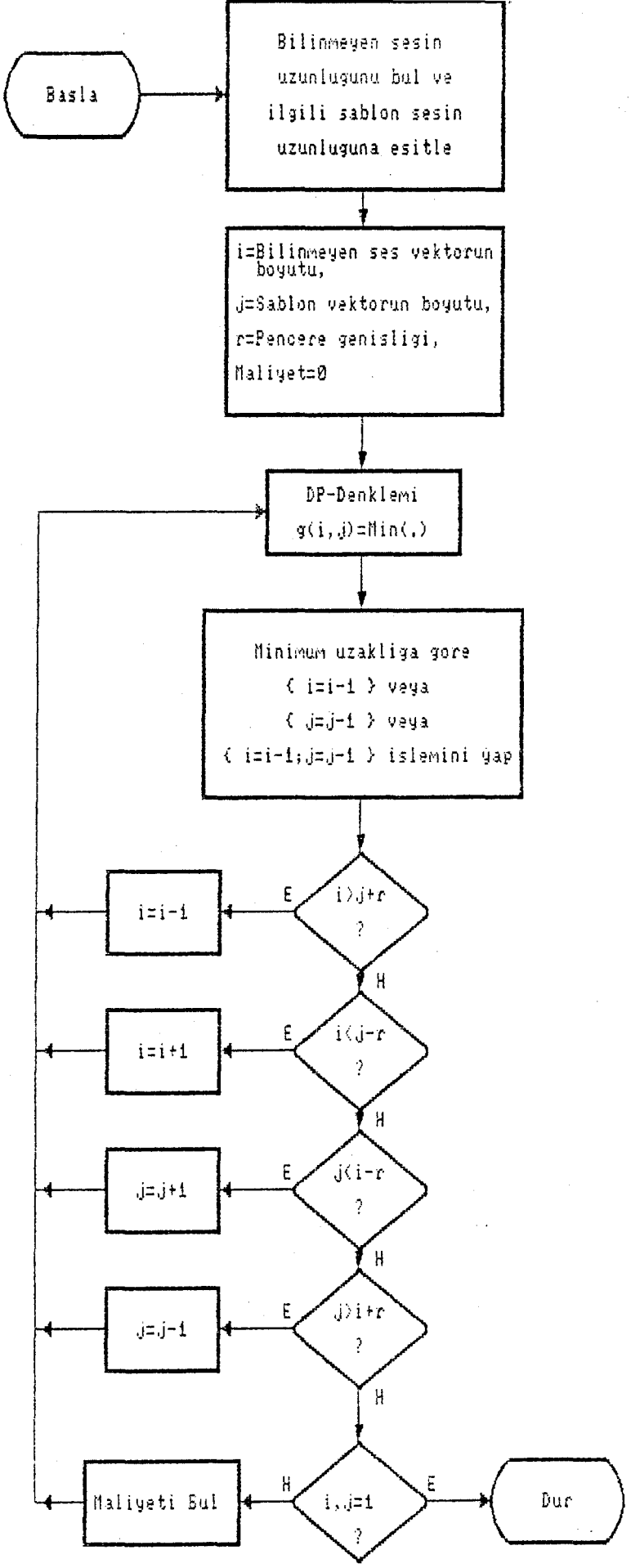
Şekil 5.1. Pascal Programı Akış Şeması



Şekil 5.2. Örnekleme Programı Akış Şeması



Şekil 5.3. Dinleme Programı Akış Şeması



Şekil 5.5. DP-Karşılaştırma Programı Akış Şeması

6. SONUÇ VE ÖNERİLER

Ses analizi yapan ve AT/XT bilgisayarları ile çalışabilen bu uyum devresi hem teknik açıdan hem de yaptığı analiz işlemleri açısından olumlu sonucu vermiştir. Sistemin bu haliyle ses tanıma yapması mümkündür.

Sistemdeki filtreler 4. mertebeden olup merkez frekansları ve band genişlikleri aşağıda verilmiştir:

1. Filtre f_c= 350 Hz , BG= 115 Hz
2. Filtre f_c= 485 Hz , BG= 160 Hz
3. Filtre f_c= 675 Hz , BG= 225 Hz
4. Filtre f_c= 945 Hz , BG= 310 Hz
5. Filtre f_c= 1310 Hz , BG= 435 Hz
6. Filtre f_c= 1825 Hz , BG= 605 Hz
7. Filtre f_c= 2535 Hz , BG= 840 Hz
8. Filtre f_c= 3530 Hz , BG= 1170 Hz

Ses tanımada ilk olarak tek bir konuşmacıdan alınan şablonlar kullanılmış ve yine aynı konuşmacıdan alınan sesler çok yüksek bir oranda tanınmıştır.

Farklı kişilerde deneme yapmak için 5 kişinin sesleri alınmış ve tanıma oranları aşağıdaki tabloda verilmiştir. Kişiden bağımsız tanıma oranlarının düşük olmasının nedeni kullanılan şablonların tek bir konuşmacıdan alınmasıdır. Tanıma oranlarının artması için değişik kişilerden alınan seslerin bu şablona dahil edilmesi gerekir.

Tanıma oranının artması için kullanılan şablon sayısının da artması gerekmektedir. Bu durum da karşılaştırma işleminin uzamasına neden olacaktır.

Tanıma oranının artması için sese ait tek bir parametre değil de buna ek olarak bir kaç parametre daha kullanmak yararlı olacaktır.

Denemeler sonucunda α=0.45 ve r=10 değerinin en yüksek tanıma oranını verdiği elde edildiğinden bu değerler kullanılmıştır.

Kelime	Bağımlı Tanıma Oranı %	Bağımsız Tanıma Oranı %
Sıfır	100	60
Bir	100	80
İki	100	60
Üç	80	40
Dört	100	80
Beş	100	80
Altı	100	60
Yedi	100	60
Sekiz	70	40
Dokuz	90	60
Tanıma %	94	62

Sistemdeki gürültüler mümkün olduğu kadar azaltılmaya çalışılmıştır. Bundan sonra böyle bir cihaz yapacaklar için bir öneri DAC ve ADC'nin referans gerilimlerini bir pil ile oluşturmalarıdır. İstenirse bu değişiklik sistem üzerinde de kolayca yapılabilir. Pil 9V olarak seçilir ve bir regülatör entegresiyle 5.00 V değerine düşürülürse pil çok uzun bir süre görevini sürdürebilecektir.

Cihaza takılacak olan güç kaynağı da çok iyi filtre edilmiş olmalıdır. Böylece gürültü etkisi azaltılmış olacaktır.

Sonuç olarak iyi bir ses tanıma cihazı için birden fazla parametre ve çok hızlı bir donanım ile çok kısa bir sürede tanıma yapmak mümkun olabilecektir.

Tanıma algoritmalarının geliştirilmesi sırasında tanıma süresi ikinci planda bırakılmıştır. Algoritmanın kendisini kanıtladıktan sonra süreyi azaltmak yoluna gitmek odukça kolaydır.

7. KAYNAKLAR DIZINI

[1] Bristow , G., Electronic speech recognition techniques , technology and applications , 1986

[2] Sakoe , H. , Chiba , S. , Dynamic programming algorithm optimization for spoken word recognition , IEEE , 1978.

[3] National Semiconductor application note , Applications , for ADC , DAC and Opamp , 1982.

[4] Intel component data book , for PPI , 1983

[5] Microsoft MS-DOS 4.01 User's Reference

[6] Norton , P., Wilton , R., The IBM PC & PS/2 , 1990.

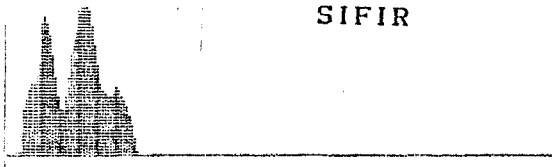
[7] Borland International , Turbo Pascal 6.0 , 1990.

[8] Franco , S., Design with operational amplifiers and ana integrated circuits , 1988 , p556-590.

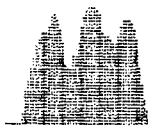
[9] Fallside , F. , Computer Speech Processing , Prentice-Hall International LTD , 1985.

SIFIR-DOKUZ ARASI SAYILARIN ÖRNEK FILTRE ÇIKTILARI

SIFIR

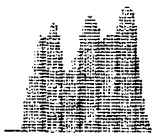


F1



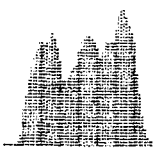
F5

F2



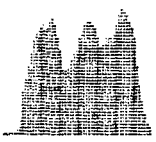
F6

F3

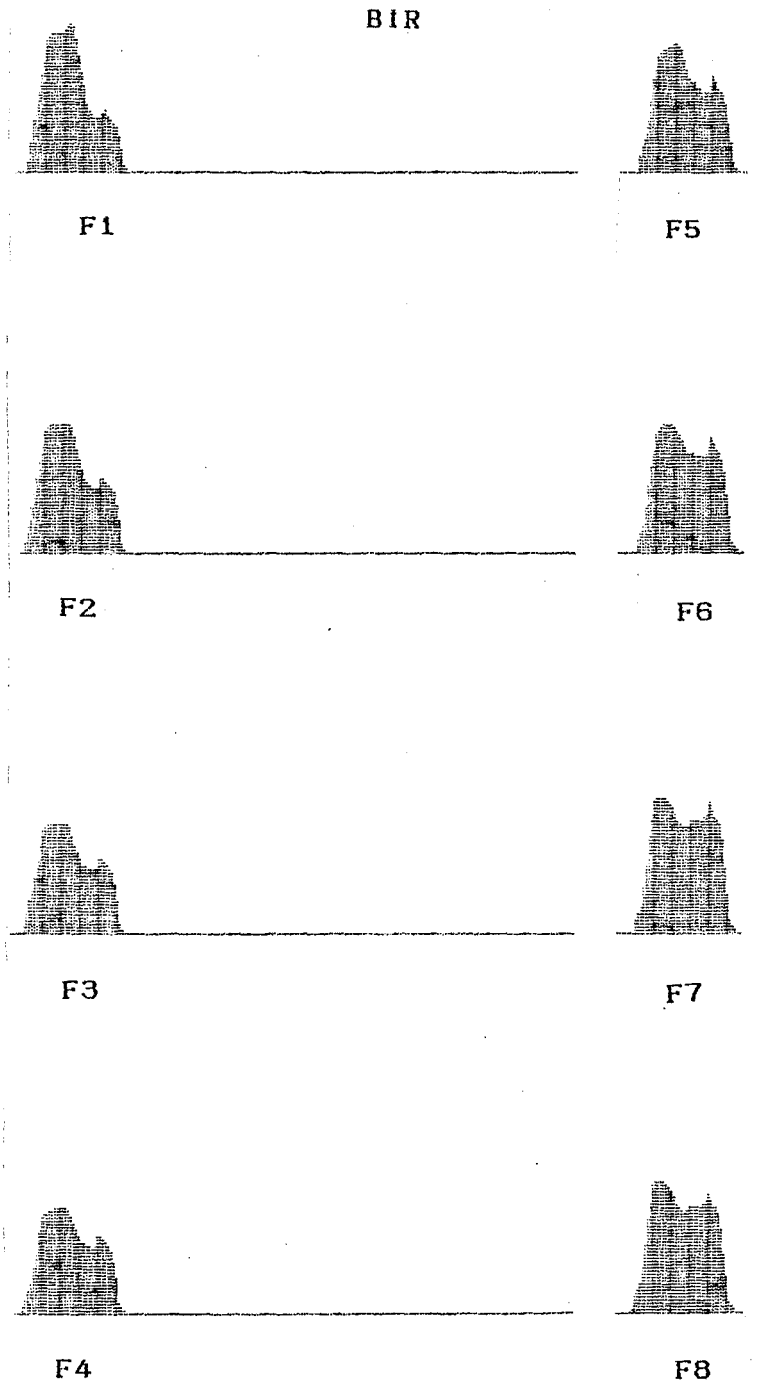


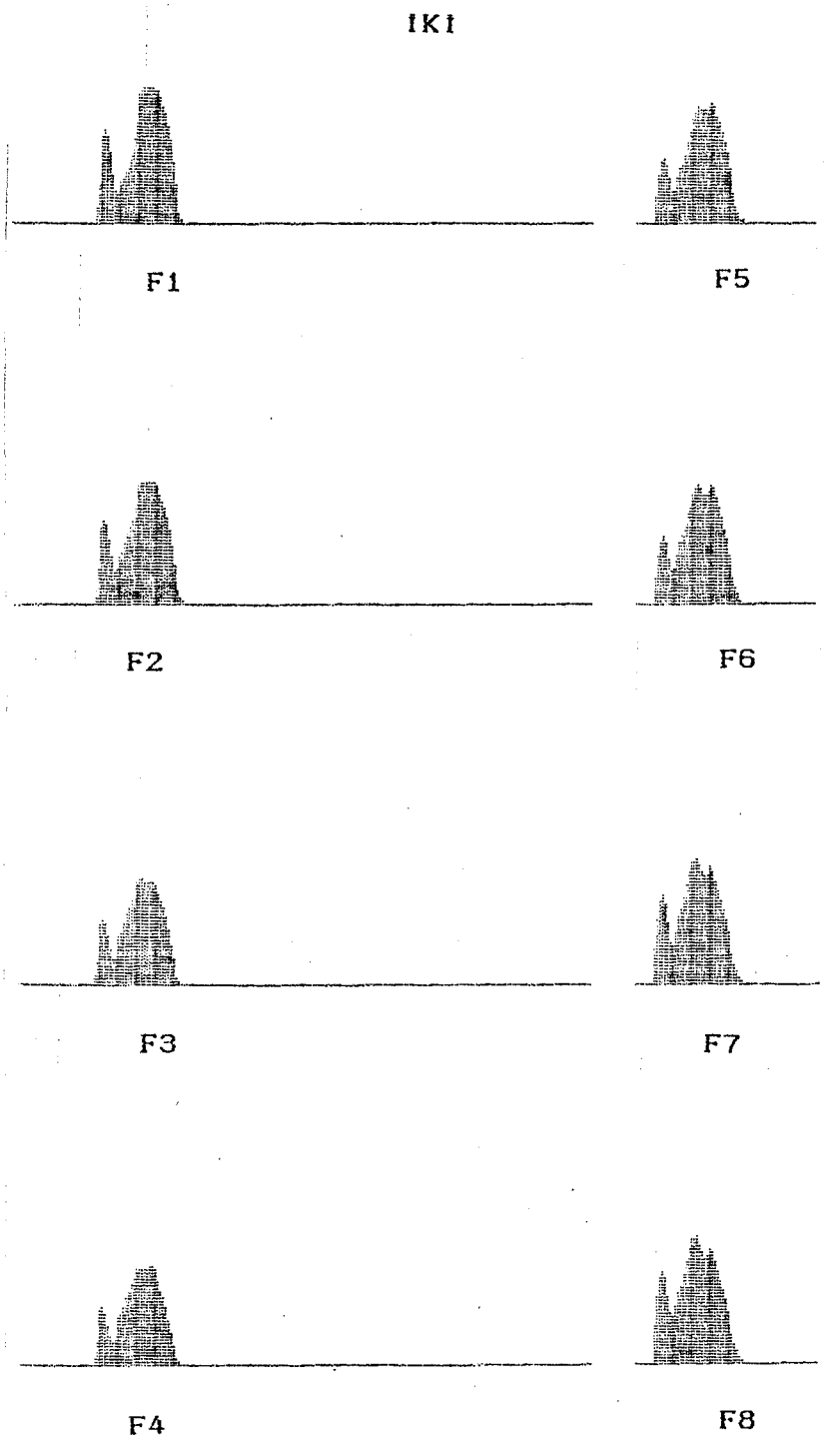
F7

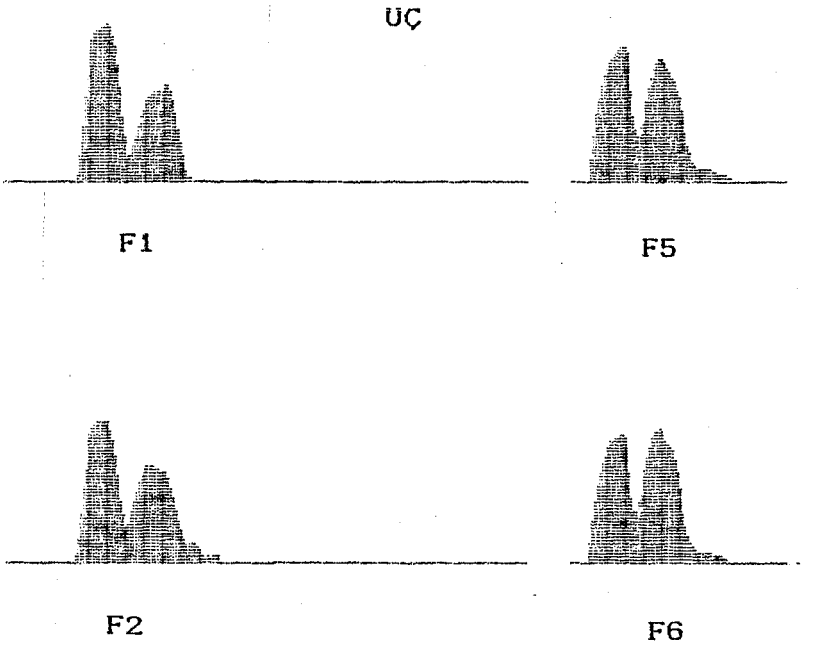
F4



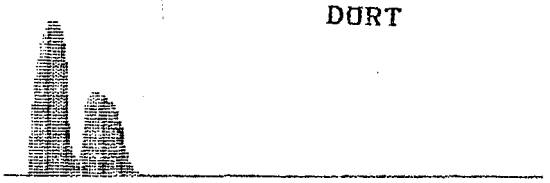
F8


BIR
F1
F5
F2
F6
F3
F7
F4
F8


IKI
F1
F5
F2
F6
F3
F7
F4
F8


UÇ
F1
F5
F2
F6
F3
F7
F4
F8

DURT



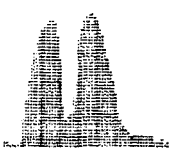
F1

F5

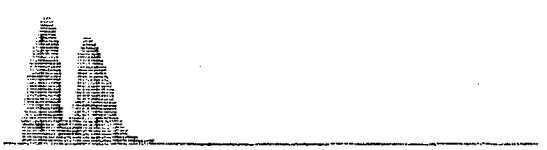
F2

F6

F3



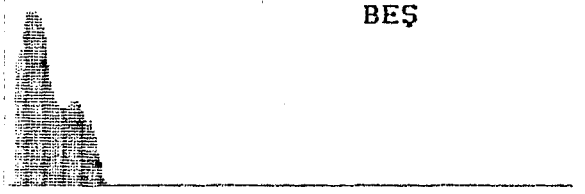
F7



F4

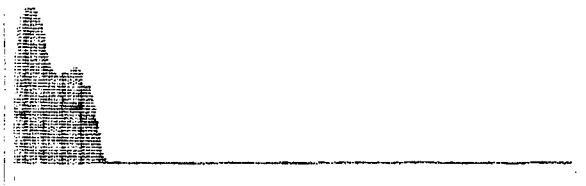
F8

BEŞ



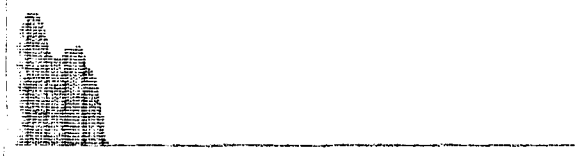
F1

F5

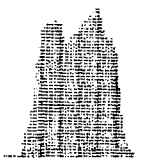


F2

F6



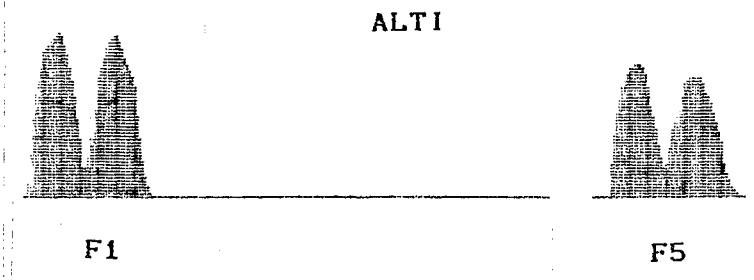
F3

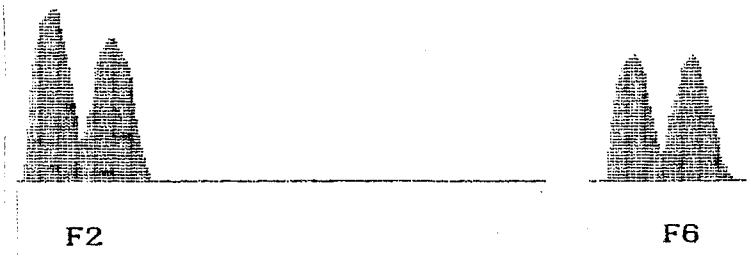


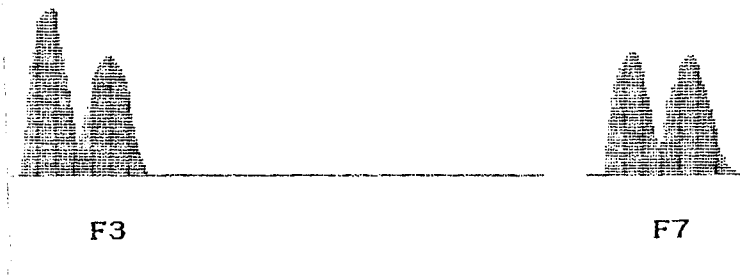
F7

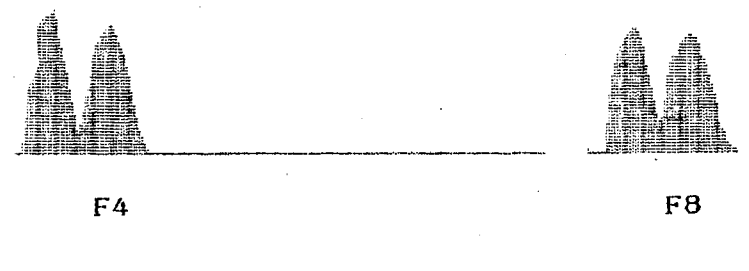
F4

F8

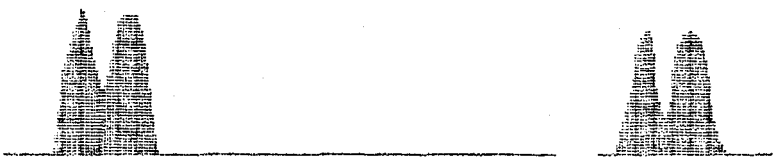

ALTI
F1
F5


F2
F6


F3
F7


F4
F8

YEDI



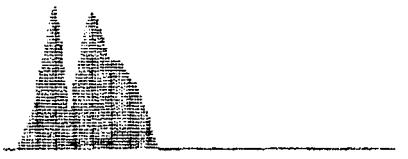
F1 F5

F2 F6

F3 F7

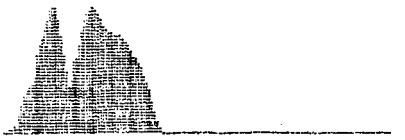
F4 F8

SEKİZ



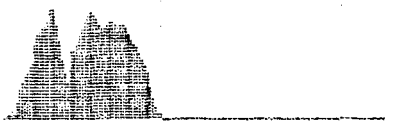
F1

F5

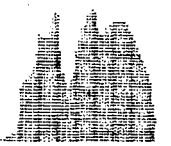


F2

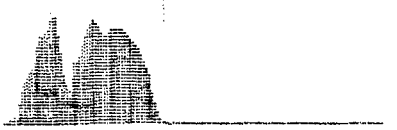
F6



F3



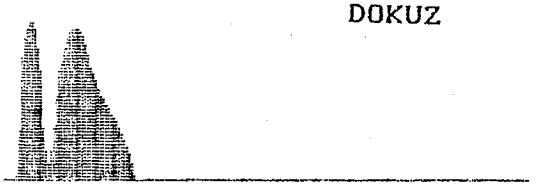
F7



F4

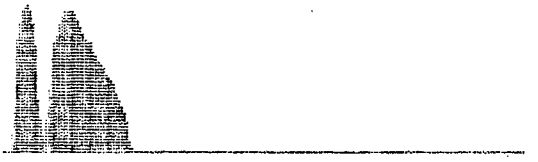
F8

DOKUZ



F1

F5

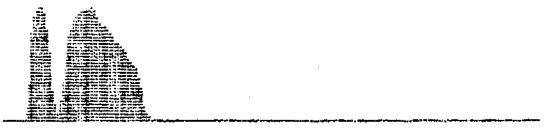


F2

F6

F3

F7



F4

F8

Sistem Programları

```
{************************************************
*** PROJECT NAME     : DP. MATCHING ALGORITHM   ***
*** INCLUDE FILES    : GRAPH/DOS/PRINTER        ***
*** AUTHOR/DATE      : SALIH EREN/10.09.1991    ***
*** RELEASE/VERSION  : 01.02                    ***
*** SOURCE LANGUAGE  : BORLAND TURBO PASCAL 6.0 ***
*** COMPILE COMPUTER : ARC PROTURBO 286e        ***
*** OPERATING SYSTEM : DOS 4.01                 ***
************************************************}

Program Speech_Recorder_and_Reconstructor;

Uses Crt,Graph,Dos;

Const
     MainPortA=512;
     MainPortB=513;
     MainPortC_Low=514;
     MainPortC_High=514;
     MainPortCommand=515;
     SecondPortA=640;
     SecondPortB=641;
     SecondPortC=642;
     SecondPortCommand=643;
     StartConversion=0;
     HoldConversion=1;
     DelayAdc1=0060;
     DelayAdc2=0010;
     DelayDac1=0071;
     DelayFilter1=0060;
     DelayFilter2=0010;
     SampleNumber=15000;
     AverageNo=23;
     Expander=1;
     FilterSelect:Array[1..8] of Byte=($01,$05,$09,$0D,$11,$15,$19,$1D);
     FilterOffset:Array[1..8] of Integer=(0,-12,15,-48,12,-16,12,36);
     Blue=1;White=15;Red=12;LightBlue=9;LightRed=12;
     DoingNo=10;
     Escape=#27;
     NoKey=#0;
     EndKey=#79;
     EnterKey=#13;
     UpArrow=#72;
     DownArrow=#80;
     LeftArrow=#75;
     RightArrow=#77;
     Menu:Array[1..10] Of String[16]=
                       (' Sampling      ',
                        ' Listening     ',
                        ' Filtering     ',
                        ' Write To File ',
                        ' Get File      ',
                        ' Draw          ',
                        ' Recognize     ',
                        ' Printer       ',
                        ' Show Demo     ',
                        ' Exit          ');

Var
     Sample:Array[1..30000] of Byte;
     Filter:Array[1..30000] of Byte;
     FilterNo,SelectData:Byte;
     i,j,k,l:Word;
     MaxX,MaxY:Word;
     DrawData:Array[1..650] of Integer;
     SignalType:Boolean;
     NumberType:Boolean;
     Change:Boolean;
     TopLeftX,TopLeftY:Byte;
     BottomRightX,BottomRightY:Byte;
     First,Last:Word;
     Point:Byte;
     MenuStep:Byte;
     ExitWilling:Boolean;
```

```
     Ch:Char;
     CLength:Byte;
     Transfer:Real;
     FilterMode:Boolean;

Procedure Construct_Initial_Conditions_For_Ports;

Begin
     Asm
        mov dx,MainPortCommand      { Select Main Port          }
        mov al,91h                  { Port A and C low input    }
        out dx,al                   { Port B and C high output  }
        mov dx,SecondPortCommand    { Do again for Second Port  }
        mov al,89h                  { Port A and C output       }
        out dx,al                   { Port B input              }
     End;
End;

Procedure Construct_Initial_Conditions_For_ADC;

Label Loop1;

Begin
     Asm
        mov dx,SecondPortA          { SecondPortA:Output        }
        mov al,HoldConversion       { Send Start Conversion     }
        out dx,al                   { pulse to ADC              }
        mov cx,0005                 { Wait until                }
 Loop1: dec cx                      { first conversion          }
        jnz Loop1                   { if exist                  }
     End;
End;

Procedure Construct_Final_Conditions_For_Ports;

Begin
     Asm
        mov dx,MainPortCommand      { Analysis is               }
        mov al,9Bh                  { just Ok.                  }
        out dx,al                   { All ports must            }
        mov dx,SecondPortCommand    { be input mode             }
        out dx,al                   { for safe                  }
     End;
End:

Procedure Get_Speech_Data_into_The_Computer_by_ADC;

Label
     Loop2,Loop3,Loop4,Delay1,Delay2,ExitAdc;


Begin

     Construct_Initial_Conditions_For_Ports;
     Construct_Initial_Conditions_For_ADC;

     Asm
        mov ax,seg Sample           { Save start address        }
        mov ds,ax                   { of speech data            }
        mov si,offset Sample        { matrix                    }
     End;

     Asm
        mov bx,SampleNumber         { Load the counter bx       }
 Loop2: mov dx,SecondPortA          { with SampleNumber.        }
        mov al,StartConversion      { Start first               }
        out dx,al                   { conversion.               }
        call Delay2                 { Wait for 1/125 kHz.       }
        mov al,HoldConversion       { Conversion                }
        out dx,al                   { just started.             }
        call Delay1                 { Get data                  }
        mov dx,MainPortA            { from ADC                  }
        in al,dx                    { using MainPortA.          }
        mov [ds:si],al              { Save the data             }
```

```
        inc si                      { to memory.              }
        mov dx,MainPortC_Low
        in al,dx
        rol al,1
        rol al,1
        rol al,1
        rol al,1
        mov [ds:si],al
        inc si
        dec bx                      { Do all this until       }
        jnz loop2                   { counter=0               }
        jmp ExitAdc                 { Exit if sampling finished }
    End;

    Asm
 Delay1:mov cx,DelayAdc1            { Load counter register   }
 Loop3: dec cx                      { with 35. Decrement it and }
        jnz Loop3                   { wait until counter=0 .  }
        ret                         { Return..                }
    End;

    Asm
 Delay2:mov cx,DelayADC2            { Load counter register   }
 Loop4: dec cx                      { with 4. Decrement it and }
        jnz Loop4                   { wait until counter=0    }
        ret                         { Return..                }
    End;

    Asm
 ExitAdc:                           { Sampling finished..     }
    End;

End;

Procedure Send_Data_to_Speaker_From_The_Computer_by_DAC;

Label
    Loop5,Loop6,ExitDac,Delay3;

Begin

    Construct_Initial_Conditions_For_Ports;
    Asm
       mov dx,641
       mov al,2
       out dx,al
    End;
  If FilterMode
  Then
    Begin
    Asm
       mov ax,seg Filter            { Save start address      }
       mov ds,ax                    { of speech data          }
       mov si,offset Filter         { matrix to send to DAC   }
    End;
   End
  Else
    Begin
     Asm
       mov ax,seg Sample            { Save start address      }
       mov ds,ax                    { of speech data          }
       mov si,offset Sample         { matrix to send to DAC   }
    End;
   End;
    Asm
       mov bx,SampleNumber          { Load the counter bx     }
 Loop5: mov dx,MainPortB            { with SampleNumber.      }
       mov al,[ds:si]               { Start from first        }
       out dx,al                    { memory location.        }
       inc si
       mov dx,MainPortC_High
       mov al,[ds:si]
       out dx,al
       call Delay3                  { Wait for 1/10 kHz.      }
```

```
        inc si                      { Get ready for other data. }
        dec bx                      { Repeat until              }
        jnz loop5                   { speech data exhausted.    }
        jmp ExitDac                 { Reconstruction is Ok.     }
      End;

      Asm
 Delay3: mov cx,DelayDac1           { Load counter register     }
 Loop6:  dec cx                     { with 35. Decrement it     }
         jnz Loop6                  { and wait until counter=0 .}
         ret                        { Return..                  }
      End;

      Asm
 ExitDac:                           { Return..                  }
      End;
End;

Procedure Send_Data_to_Filter_and_Get_Filter_Outputs(FilterNo:Byte);

Label
     Loop7,Loop8,Loop9,ExitFilter,Delay4,Delay5;

Begin

     SelectData:=FilterSelect[FilterNo];

     Construct_Initial_Conditions_For_Ports;
     Construct_Initial_Conditions_For_ADC;

     Asm
        mov ax,seg Sample           { Save start address       }
        mov ds,ax                   { of speech data           }
        mov si,offset Sample        { matrix.                  }
        mov ax,seg Filter           { Store into               }
        mov es,ax                   { this memory location     }
        mov di,offset Filter        { under Sample name.       }
     End;

     Asm
        mov dx,SecondPortB          { Select FilterNo          }
        mov al,SelectData           { for multiplexer.         }
        out dx,al
     End;

     Asm
        mov bx,SampleNumber
 Loop7: mov dx,MainPortB
        mov al,[ds:si]
        out dx,al
        inc si
        mov dx,MainPortC_High
        mov al,[ds:si]
        out dx,al
        mov dx,SecondPortA
        xor al,al
        out dx,al

        call delay4

        mov al,HoldConversion
        out dx,al

        call delay5

        mov dx,MainPortA
        in al,dx
        mov [es:di],al
        inc di
        mov dx,MainPortC_Low;
        in al,dx
        rol al,1
        rol al,1
        rol al,1
```

```
        rol al,1
        mov [es:di],al
        inc si
        inc di
        dec bx
        jnz loop7
        jmp ExitFilter
    End;

    Asm
 Delay5:mov cx,DelayFilter1        { Load counter register     }
 Loop8: dec cx                     { with 35. Decrement it and }
        jnz Loop8                  { wait until counter=0 .    }
        ret                        { Return..                  }
    End;

    Asm
 Delay4:mov cx,DelayFilter2        { Load counter register     }
 Loop9: dec cx                     { with 4. Decrement it and  }
        jnz Loop9                  { wait until counter=0      }
        ret                        { Return..                  }
    End;

    Asm
 ExitFilter:                       { Return..                  }
    End;
End;

Procedure Get_Copy_of_Sample_to_Filter;

Label
    Loop10;

Begin
    Asm
        mov ax,seg Sample          { Original                  }
        mov ds,ax                  { speech data               }
        mov si,offset Sample       { memory location.          }
        mov ax,seg Filter          { Filtered speech           }
        mov es,ax                  { data                      }
        mov di,offset Filter       { memory location.          }
     End;

     Asm
        mov bx,30000               { Get speech data           }
 Loop10:mov al,[ds:si]             { from Sample address.      }
        mov [es:di],al             { Save this data            }
        inc si
        inc di
        dec bx                     { to Filter address.        }
        jnz Loop10                 { Do this all speech data.  }
     End;
End;


Procedure CursorOn;

Begin
    Asm
        xor ch,ch
        mov cl,14
        mov ah,1
        xor al,al
        int 10h
    End;
End;


Procedure CursorOff;

Begin
    Asm
        mov ch,31
        xor cl,cl
```

```
        mov ah,1
        xor al,al
        int 10h
      End;
End;

Procedure DrawFrame(TopLeftX,TopLeftY,BottomRightX,
                                    BottomRightY:Byte);

Begin
  GotoXY(TopLeftX,TopLeftY);
    Write(Chr(201));
      For i:=TopLeftX+1 To BottomRightX-1 Do Write(Chr(205));
        Write(Chr(187));
         For i:=TopLeftY+1 To BottomRightY-1
           Do
            Begin
              GotoXY(TopLeftX,i);     Write(Chr(186));
              GotoXY(BottomRightX,i); Write(Chr(186));
            End;
     GotoXY(TopLeftX,BottomRightY);   Write(Chr(200));
    For i:=TopLeftX+1 To BottomRightX-1 Do Write(Chr(205));
  Write(Chr(188));
End;

Procedure Samples_Write_to_a_File_Specified_by_User;

Type
    Message_Length=String[20];

Var
   FileCreate:Text;
   Counter:Word;
   FileName,Speaker,Information,Command:Message_Length;
   DirEntry:String[20];

Begin
     FilterMode:=False;
     Ch:=ReadKey;
     If (Ch='F') Or (Ch='f') Then FilterMode:=True;
     DrawFrame(9,9,70,11);
     GotoXY(10,10);
     Write(' Enter to Save File Name    : ');ReadLn(FileName);

     GotoXY(9,11);Write('                                                                  ');
     DrawFrame(9,9,70,12);
     GotoXY(10,11);
     Write(' Enter Speaker Name         : ');ReadLn(Speaker);

     GotoXY(9,12);Write('                                                                  ');
     DrawFrame(9,9,70,13);
     GotoXY(10,12);
     Write(' Information
                        : ');ReadLn(Information);

     GotoXY(9,13);Write('                                                                  ');
     DrawFrame(9,9,70,14);
     GotoXY(10,13);
     Write(' Command
                      : ');ReadLn(Command);

     DrawFrame(25,20,52,22);
     TextColor(Yellow);
     GotoXY(27,21);Write(' Saving File..          ');
     TextColor(White);
     DirEntry:='c:\Turbo\Samples';
     ChDir(DirEntry);

     Assign(FileCreate,FileName+'.SMP');
     ReWrite(FileCreate);

     WriteLn(FileCreate,Speaker);
     WriteLn(FileCreate,Information);
     WriteLn(FileCreate,Command);
```

```
     WriteLn(FileCreate,First);
     WriteLn(FileCreate,Last);

     Counter:=1;

   If FilterMode
            Then
              Begin
                 Repeat
                    WriteLn(FileCreate,Filter[Counter]);
                    Inc(Counter);
                 Until Counter=2*SampleNumber;
              End
            Else
              Begin
                 Repeat
                    WriteLn(FileCreate,Sample[Counter]);
                    Inc(Counter);
                 Until Counter=2*SampleNumber;
              End;
     Close(FileCreate);
     DirEntry:='C:\Turbo\Bgi';
     ChDir(DirEntry);
End;


Procedure Samples_Read_From_a_File_Specified_by_User;

Type
    Message_Length=String[20];

Var
   FileLoad:Text;
   Counter:Word;
   DirEntry,FileName,Speaker,Information,
                       Command:Message_Length;


Begin
     FilterMode:=False;
     Ch:=ReadKey;
     If (Ch='F') Or (Ch='f') Then FilterMode:=True;
     DrawFrame(1,1,80,24);
     DrawFrame(9,9,70,11);
     GotoXY(10,10);
     Write(' Enter to Load File Name    : ');ReadLn(FileName);
     DirEntry:='C:\Turbo\Samples';
     ChDir(DirEntry);
     Assign(FileLoad,FileName+'.SMP');

     {$I-}
       Reset(FileLoad);
     {$I+}

     If IOResult <> 0 Then
              Begin
                WriteLn('***  File Not Found or I/O Error  ***');
                Delay(1000);
                Exit;
              End;

      DrawFrame(25,20,52,22);
      TextColor(Yellow);
      GotoXY(27,21);Write('Loading File...');
      TextColor(White);

      ReadLn(FileLoad,Speaker);
      ReadLn(FileLoad,Information);
      ReadLn(FileLoad,Command);
      ReadLn(FileLoad,First);
      ReadLn(FileLoad,Last);

      Counter:=1;
   If FilterMode
```

```
                    Then
                      Begin
                        While Not Eof(FileLoad) Do
                          Begin
                            ReadLn(FileLoad,Filter[Counter]);
                            Inc(Counter);
                          End;
                      End
                    Else
                      Begin
                        While Not Eof(FileLoad) Do
                        Begin
                          ReadLn(FileLoad.Sample[Counter]):
                          Inc(Counter);
                        End;
                      End;

          Close(FileLoad);
          ClrScr;
          DrawFrame(9,9,70,16);
          GotoXY(10,10);Write(' Loaded File Name : ');WriteLn(FileName);
          GotoXY(10,11);Write(' Speaker Name     : ');WriteLn(Speaker);
          GotoXY(10,12);Write(' Information      : ');WriteLn(Information):
          GotoXY(10,13);Write(' Command          : ');WriteLn(Command);
          GotoXY(10,14);Write(' First Frame      : ');WriteLn(First);
          GotoXY(10,15);Write(' Last Frame       : ');WriteLn(Last);

          DirEntry:='C:\Turbo\Bgi';
          ChDir(DirEntry);
        End;



        Procedure Calculate_Speech_For_Drawing(SignalType:Boolean);

        {    Using  Calculate_Speech_For_Drawing Procedure , speech data
          can be prepared for drawing on to the screen. This data stored
          to  DrawData matrix. But it is unclear signal type. Signal can
          be drawn as following types:
             1) Intensity,
             2) Amplitude.

             ....
             End of Explanation  }

        Var
             CntX:Word;
             Total:Real;
             Counter:Word;
             Cnt:Word;

        Begin
          If (SignalType) And (FilterMode)
            Then
              Begin
               Counter:=1;CntX:=1;
                Repeat
                  Total:=0;
                    For Cnt:=1 to AverageNo Do
                     Begin
                      Transfer:=((Filter[Counter]*16+
                                  (Filter[Counter] Shr 4))-2047);
                       Transfer:=Abs(Transfer);
                        If Transfer<10 Then Transfer:=0;
                          Total:=Total+Sqr(Transfer*10);
                             Inc(Counter,2);
                     End;
                  Total:=Abs(Total);
        DrawData[CntX]:=4*(Trunc(10*(Ln(1+Total/AverageNo*Expander))/Ln(10)))
erOffset[FilterNo]);
                     Inc(CntX);
                Until CntX>639;
              End;

           If (SignalType) And (Not(FilterMode))
```

```
    Then
      Begin
       Counter:=1;CntX:=1;
        Repeat
          Total:=0;
            For Cnt:=1 to AverageNo Do
             Begin
              Transfer:=((Sample[Counter]*16+
                        (Sample[Counter] Shr 4))-2047);
               Transfer:=Abs(Transfer);
              If Transfer<10 Then Transfer:=0;
               Total:=Total+Sqr(Transfer*10);
                    Inc(Counter,2);
             End;
               Total:=Abs(Total);
   DrawData[CntX]:=4*(Trunc(10*(Ln(1+Total/AverageNo*
                                 Expander))/Ln(10))-62);
            If DrawData[CntX]<0
                          Then
                             Begin
                             DrawData[CntX]:=Random(3);
                          End;
          Inc(CntX);
      Until CntX>639;
     End;

End;


Procedure SetGraphMode;

Var
   Gd,Gm:Integer;

Begin
     Gd:=Detect;
     InitGraph(Gd,Gm,'');
     If IOResult<>GrOk Then
                       Begin
                           ClrScr;
                           WriteLn('*** Graphic Error ***');
                           Halt(1);
                       End;
     MaxX:=GetMaxX;
     MaxY:=GetMaxY;
End;


Procedure Draw_X_Y_Axis;

Const
     NameX='Time';
     RefX=10;
Var
   NameY:String[9];

Begin

     SetBkColor(Blue);
      SetColor(White);

       MoveTo(RefX,MaxY Div 2);
        LineTo(RefX,MaxY Div 8);
         LineTo(RefX,MaxY-MaxY Div 8);
          MoveTo(RefX,MaxY Div 2);
           LineTo(MaxX-RefX,MaxY Div 2);

     If SignalType
                 Then
                     NameY:='Intensity'
                 Else
                     NameY:='Amplitude';

     MoveTo(RefX,MaxY Div 8-16);
```

```
     OutText(NameY);
     MoveTo(MaxX-40,MaxY Div 2+8);
     OutText(NameX);

     MoveTo(0,0);
      LineTo(0,MaxY);
       LineTo(MaxX,MaxY);
      LineTo(MaxX,0);
     LineTo(0,0);

     MoveTo(8,54);
      OutText(Chr(24));
      MoveTo(MaxX-RefX-2,MaxY Div 2-3);
     OutText(Chr(26));

End;

Procedure DrawSpeechSignal;

Var
   Counter:Word;
   OffsetX,OffsetY:Word;
Begin
     MoveTo(3,100);
     Counter:=1;
     OffsetX:=13;OffsetY:=MaxY Div 2;
     MoveTo(OffsetX,OffsetY+DrawData[Counter]);

     Repeat
           MoveTo(OffsetX+Counter,OffsetY);
            LineRel(0,-DrawData[Counter]);
          Inc(Counter);
     Until Counter>639;
End;


Procedure GetSpeechStatus;

Const
     RefX=13;
Var
   Counter:Word;
   Step:Byte;
   Ch:Char;
   StepSize:Boolean;
   ModeType:Byte;
   Final:Boolean;
Begin
     Counter:=RefX;
     First:=13;
     Last:=600;
     ModeType:=0;
     Final:=True;

     Repeat
        Bar(Counter,MaxY Div 2+2,Counter,MaxY Div 2+10+2);
        Ch:=ReadKey;
        If Ch=EndKey Then StepSize:=Not(StepSize);
        If StepSize Then Step:=25 Else Step:=1;

        If (Ch=RightArrow) And (Counter<Last)
            Then
                Begin
                 SetColor(Blue);
                 MoveTo(Counter,MaxY Div 2+2);
                 LineTo(Counter,MaxY Div 2+10+2);
                 Inc(Counter,Step);
                 SetColor(White);
                 Bar(Counter,MaxY Div 2+2,Counter,MaxY Div 2+10+2);
                End;

        If (Ch=LeftArrow) And (Counter>First)
           Then
              Begin
```

```
                SetColor(Blue);
                MoveTo(Counter,MaxY Div 2+2);
                LineTo(Counter,MaxY Div 2+10+2);
                Dec(Counter,Step);
                SetColor(White);
                Bar(Counter,MaxY Div 2+2,Counter,MaxY Div 2+10+2);
              End;

        If (Ch=UpArrow)
                  Then
                     Begin
                          If ModeType<=1 Then
                           Begin
                              Inc(ModeType);
                              If ModeType=1 Then First:=Counter;
                              If ModeType=2 Then Last:=Counter;
                              SetColor(Red);
                              MoveTo(Counter,MaxY Div 2+10+2);
                              LineTo(Counter,100);
                              SetColor(White);
                           End;
                      End;

        If (Ch=DownArrow)
                    Then
                       Begin
                            SetColor(Blue);
                            MoveTo(Counter,100);
                            LineTo(Counter,MaxY Div 2+2);
                            PutPixel(Counter,MaxY Div 2,White);
                            MoveTo(Counter,MaxY Div 2);
                            SetColor(White);
                            If DrawData[Counter]> 10 Then
                            i:=-DrawData[Counter] Else i:=0;
                            LineRel(0,i);

                            SetColor(White);
                            If ModeType=2 Then
                                          Begin
                                               Dec(ModeType);
                                               Last:=600;
                                          End;
                            If ModeType=1 Then
                                           Begin
                                                Dec(ModeType);
                                                First:=13;
                                           End;
                       End;
    Until Ch=Escape
End;

Procedure Draw_Menu;

Begin
    For i:=1 to 10 Do
                   Begin
                      GotoXY(30,3+i);
                      Write(Menu[i]);
                   End;
End;

Procedure ShowDemo;

Const
    EndOfPattern=20;
    EndOfUnKnown=20;
    Step=20;

Var
   Warning:String[20];
   d:Array[1..EndOfPattern,1..EndOfUnknown] Of Real;
   Cost:Real;
   Left,LeftDown,Down:Real;
   Done:Boolean;
```

```
    DidNotLeft,DidNotDown:Boolean;
    Gd,Gm:Integer;
    MaxX,MaxY:Word;
    Alfa:Real;
    k:Byte;
    U:Array[1..EndOfUnknown] Of Real;
    R:Array[1..EndOfUnknown] Of Real;
Begin
     If NumberType
                 Then
                     Begin
                          Randomize;
                          For i:=1 To EndOfUnKnown
                            Do
                              Begin
                                   U[i]:=Random(100);
                                   R[i]:=Random(100);
                              End;
                     End
                 Else
                     Begin
                          ClrScr;
                          For i:=1 To EndOfUnKnown
                            Do
                              Begin
                                   Write('U[',i,']=');ReadLn(U[i]);
                              End;
                           ClrScr;
                          For j:=1 To EndOfPattern
                            Do
                              Begin
                                   Write('R[',j,']=');ReadLn(R[j]);
                              End;
                      End;


     Gd:=Detect;
     InitGraph(Gd,Gm,'');
     If IOResult<>grOk Then Halt(1);
     MaxX:=GetMaxX;
     MaxY:=GetMaxY;
     Bar(0,MaxY,MaxX,MaxY);
     Bar(0,0,0,MaxY);
     Bar(0,0,MaxX,0);
     Bar(MaxX,0,MaxX,MaxY);
     SetbkColor(Blue);
     i:=20;

     Repeat
           j:=80;
                Repeat
                     PutPixel(i,j,White);
                     Inc(j,Step);
                Until j>480;
           Inc(i,Step);
     Until i>400;

     SetColor(Green);
     MoveTo(20,460);
     LineTo(400,80);
     SetColor(Red);

     MoveTo(20+CLength*20,460);
     LineTo(400,80+CLength*20);

     MoveTo(20,460-CLength*20);
     LineTo(400-CLength*20,80);

     SetColor(White);

     Cost:=0;
     i:=EndOfPattern;
     j:=EndOfUnknown;
```

```
MoveTo(400,80);

Repeat
      Done:=True;
      DidNotLeft:=False;
      DidNotDown:=False;
      Alfa:=2;

      d[2,2]:=Abs(Sqr(R[i]-U[j]));
      d[2,1]:=Abs(Sqr(R[i]-U[j-1]));
      d[1,2]:=Abs(Sqr(R[i-1]-U[j]));
      d[1,1]:=Alfa*Abs(Sqr(R[i-1]-U[j-1]));

      Left:=Abs(d[2,2]-d[1,2]);
      Down:=Abs(d[2,2]-d[2,1]);
      LeftDown:=Abs(d[2,2]-d[1,1]);

      If (Left<Down) And (Left<LeftDown)
                     Then
                         Begin
                              Dec(i);
                              If (i<j+CLength+1)
                                 Then
                                     Begin
                                       Inc(i);
                                       Left:=MaxAvail;
                                       DidNotLeft:=Not(DidNotLeft);
                                     End
                                 Else
                                     Begin
                                          Cost:=Cost+Left;
                                          Done:=Not(Done);
                                          LineRel(-Step,0);
                                     End;
                         End;

      If (Down<Left) And (Down<LeftDown) And (Done)
                     Then
                         Begin
                              Dec(j);
                              If (i)j-CLength+1)
                                 Then
                                     Begin
                                       Inc(j);
                                       Down:=MaxAvail;
                                       DidNotDown:=Not(DidNotDown);
                                     End
                                 Else
                                     Begin
                                          Cost:=Cost+Down;
                                          Done:=Not(Done);
                                          LineRel(0,Step);
                                     End;
                     End;

      If (Done) And (DidNotLeft)
                      Then
                          Begin
                               If (LeftDown<Down)
                                  Then
                                      Begin
                                           Cost:=Cost+LeftDown;
                                           Dec(i);
                                           Dec(j);
                                           LineRel(-Step,Step);
                                           Done:=Not(Done);
                                      End
                                  Else
                                      Begin
                                           Dec(j);
                                           Cost:=Cost+Down;
                                           LineRel(0,Step);
                                           Done:=Not(Done);
                                      End;
```

```
                              End;

          If (Done) And (DidNotDown)
                         Then
                             Begin
                                  If (LeftDown<Left)
                                       Then
                                         Begin
                                          Cost:=Cost+LeftDown;
                                          Dec(i);
                                          Dec(j);
                                          LineRel(-Step,Step);
                                          Done:=Not(Done);
                                         End
                                       Else
                                           Begin
                                            Dec(i);
                                            Cost:=Cost+left;
                                            LineRel(-Step,0);
                                            Done:=Not(Done);
                                           End;
                                End;
          If (Done)
                 Then
                     Begin
                          Cost:=Cost+LeftDown;
                          Dec(i);
                          Dec(j);
                          LineRel(-Step,Step);
                          Done:=Not(Done);
                     End;

     Until (i=1) Or (j=1);

     If (i>1) Then
                Begin
                   LineRel(-(i-1)*Step,0);
                   For k:=i DownTo 2 Do
                               Begin
                                 d[1,2]:=Abs(Sqr(R[i-1]-U[j]));
                                 Left:=Abs(d[2,2]-d[1,2]);
                                 Cost:=Cost+Left;
                               End;
                 End;
     If (j>1) Then
                Begin
                   LineRel(0,(j-1)*Step);
                   For k:=j DownTo 2 Do
                                    Begin
                                      d[2,1]:=Abs(Sqr(R[i]-U[j-1]));
                                      Down:=Abs(d[2,2]-d[2,1]);
                                      Cost:=Cost+Down;
                                    End;

                End;
      MoveTo(480,285);
      OutText('Total Cost=');
      MoveTo(500,300);
      Str(Cost:6:3,Warning);
      OutText(Warning);
      Write(Chr(7));
      MoveTo(10,10);
      OutText('Dynamic Programming Algorithm');
      MoveTo(10,25);
      OutText('(C) Salih EREN 1992');
      Repeat Until KeyPressed;
      CloseGraph;
End;

Procedure GetReady;

Begin
     ClrScr;
     CursorOff;
```

```
     Window(29,3,45,14);
     TextBackGround(Blue);
     Window(1,1,80,24);
     TextColor(White);
     DrawFrame(29,3,45,14);
     TextBackGround(White);
     TextColor(LightBlue);
     Draw_Menu;
     DrawFrame(1,1,79,24);
End;

Procedure GetReady2;
Begin
     CLength:=2;
     ExitWilling:=False;
     Point:=1;
     MenuStep:=0;
     ClrScr;
     TextBackGround(Blue);
End;

Procedure GetReady4;
Begin;
Window(1,1,80,25);
          TextBackGround(Blue);
          ClrScr;
          TextColor(White);
          DrawFrame(1,1,80,24);
          DrawFrame(25,10,52,14);
          GotoXY(27,11);Write('Show with ');
          TextColor(Red);GotoXY(37,11);Write('R');
          TextColor(White);GotoXY(38,11);Write('andom Numbers');
          TextColor(Red);GotoXY(27,12);Write('E');
          TextColor(White);GotoXY(28,12);Write('nter New Numbers');
          TextColor(Red);GotoXY(27,13);Write('C');
          TextColor(White);GotoXY(28,13);Write('hange Restriction');
End;

Procedure GetReady3;

Begin
     TextColor(White);
     TextBackGround(LightRed);
     GotoXY(30,4+MenuStep);
     Write(Menu[Point]);
End;

Procedure SelectChoice;

Begin
     Case Point Of

     1:Begin { Sampling}
          ClrScr;
          Window(1,1,80,25);
          TextBackGround(Blue);
          ClrScr;
          TextColor(White);
          DrawFrame(25,11,58,15);
          DrawFrame(1,1,80,24);
          GotoXY(27,13);
          TextColor(White+Blink);
          Write('Press a Key To Start Sampling');
          Ch:=ReadKey;
          iF Ch=Escape
                         Then
                                Begin
                                  GetReady;
                                  GetReady3;
                                End;
          If Ch=EnterKey
                    Then
                       Begin
                          TextColor(Yellow);
```

```
                    GotoXY(27,13);
                    Write(' Sampling Started...          ');
                    Get_Speech_Data_into_The_Computer_by_ADC;
                    TextColor(White+Blink);
                    GotoXY(27,13);
                    Write(' Sampling Finished...         ');
                    Repeat Until KeyPressed;
                    GetReady;
                    GetReady3
                  End;
  End;

2:Begin  { Listening }
    ClrScr;
     Window(1,1,80,25);
      TextBackGround(Blue);
       ClrScr;
        TextColor(White);
         DrawFrame(25,11,58,15);
          DrawFrame(1,1,80,24);
           GotoXY(27,13);
            TextColor(White+Blink);
             Write('Press a Key To Start Listening');
              FilterMode:=False;
              Ch:=ReadKey;
               If Ch=Escape
                        Then
                           Begin
                             GetReady;
                             GetReady3;
                           End;

              If (Ch='F') Or (Ch='f') Then
                                        Begin
                                          FilterMode:=True;
                                          Ch:=ReadKey;
                                        End;
    If Ch=EnterKey
          Then
            Begin
              TextColor(Yellow);
              GotoXY(27,13);
              Write(' Listening...                 ');
              Send_Data_to_Speaker_From_The_Computer_by_DAC;
              TextColor(White+Blink);
              GotoXY(27,13);
              Write(' Listening Finished...        ');
              Repeat Until KeyPressed;
              GetReady;
              GetReady3
            End;
  End;

3:Begin
    { Filtering }
    Window(1,1,80,25);
    TextBackGround(Blue);
    ClrScr;
    TextColor(White);
    DrawFrame(25,12,49,14);
    DrawFrame(1,1,80,24);
    GotoXY(27,13);Write(' Enter Channel No: ');
    Repeat
      Read(FilterNo);
    Until  FilterNo in [1..8];
    TextColor(Yellow);
    GotoXY(27,13);Write('Filtering Channel: ',FilterNo);

   Send_Data_to_Filter_and_Get_Filter_Outputs(FilterNo);

    TextColor(White+Blink);
    GotoXY(27,13);Write('Filtering Finished.. ');
    Repeat Until KeyPressed;
    GetReady;
```

```
    GetReady3
  End;

4:Begin
       ClrScr;
        Window(1,1,80,25);
         TextBackGround(Blue);
          TextColor(White);
           ClrScr;
            DrawFrame(1,1,80,24);
             Samples_Write_to_a_File_Specified_by_User;
             DrawFrame(25,20,52,22);
            TextColor(Yellow+Blink);
           GotoXY(27,21);Write('Press a Key To Continue');
          Repeat Until KeyPressed;
         ClrScr;
        GetReady;
       GetReady3;
  End;

5:Begin
       ClrScr;
        Window(1,1,80,25);
         TextBackGround(Blue);
          TextColor(White);
           ClrScr;
            Samples_Read_From_a_File_Specified_by_User;
             DrawFrame(1,1,80,24);
             DrawFrame(25,20,52,22);
            TextColor(Yellow+Blink);
           GotoXY(27,21);Write('Press a Key To Continue');
          Repeat Until KeyPressed;
         ClrScr;
        GetReady;
       GetReady3;
  End;

6:Begin

       ClrScr;
       Window(1,1,80,25);
       TextBackGround(Blue);
       ClrScr;
       DrawFrame(1,1,80,24);
       DrawFrame(30,11,42,14);
       GotoXY(31,12);TextColor(Red);Write(' I');
       TextColor(White);Write('ntensity');
       GotoXY(31,13);TextColor(Red);Write(' A');
       TextColor(White);Write('mplitude');
       FilterMode:=False;
Repeat
  Ch:=ReadKey;
  If (Ch='F') Or (Ch='f') Then Begin FilterMode:=True;Ch:=ReadKey;End;
  If (Ch='I') Or (Ch='i') Then SignalType:=True;
  If (Ch='A') Or (Ch='a') Then SignalType:=False;
  If (Ch=Escape) Then
                    Begin
                        TextBackGround(Blue);
                        GetReady;
                        GetReady3;
                        Exit;
                    End;

       Until (Ch='I') Or (Ch='i') Or (Ch='A') Or (Ch='a');

       Calculate_Speech_For_Drawing(SignalType);
       SetGraphMode;
       Draw_X_Y_Axis;
       DrawSpeechSignal;
       GetSpeechStatus;
       Repeat Until KeyPressed;
       CloseGraph;
       TextBackGround(Blue);
       GetReady;
```

```
          GetReady3;
      End;

    7:Begin
        { Recognize }
      End;

    8:Begin
        i:=1;
        TextBackGround(Blue);
        ClrScr;
         for i:=1 to 640 do writeln(i,'  ',drawData[i]);
          clrscr;delay(1000);write(chr(7));
           i:=1;j:=1;
       Repeat
        Transfer:=Filter[i]*16 +(Filter[i+1] Shr 4);
          Transfer:=Abs(Transfer-2335);
           If Transfer<10 Then Transfer:=0;
            Writeln(j,'  ',Transfer:5:2);
             Inc(i,2);
              inc(j);
            if keypressed then exit;
       Until j>15000; {SampleNumber;} { Printer }
      End;

    9:Begin
         GetReady4;
            Repeat
               Ch:=ReadKey;
               If (Ch='R') Or (Ch='r') Then NumberType:=True;
               If (Ch='E') Or (Ch='e') Then NumberType:=False;
               If (Ch='C') Or (Ch='c') Then
                             Begin
                                ClrScr;
                                DrawFrame(1,1,80,24);
                                DrawFrame(25,12,65,14);
                                GotoXY(27,13);
                                Write('New Restriction is <');
                                TextColor(Yellow+Blink);
                                Write(CLength);
                                TextColor(White);
                               Write('>  ');
                               ReadLn(CLength);
                               GetReady4;
                             End;
               If (Ch=Escape) Then
                                Begin
                                   TextBackGround(Blue);
                                   GetReady;
                                   GetReady3;
                                   Exit;
                                End;
            Until (Ch='R') Or (Ch='r')
                           Or (Ch='E') Or (Ch='e');

        ShowDemo;

        Window(1,1,80,25);
        TextBackGround(Blue);
        GetReady;
        GetReady3;

      End;

    10:Begin
        ExitWilling:=Not(ExitWilling);
        CursorOn;
       End;
   End;
End;

{ Main Program }
```

```
Begin
GetReady2;
GetReady;
GetReady3;
        Repeat
           Ch:=ReadKey;
           Case Ch Of
          NoKey: Begin
                 Ch:=ReadKey;
                 TextColor(LightBlue);
                TextBackGround(White);
                GotoXY(30,4+MenuStep);
                Write(Menu[Point]);


                 Case Ch Of
                  DownArrow: Begin
                             Inc(Point);
                             Inc(MenuStep);
                             If Point>DoingNo
                                     Then
                                       Begin
                                          Point:=1;
                                          MenuStep:=0;
                                       End;
                             TextColor(White);
                             TextBackGround(LightRed);
                             GotoXY(30,4+MenuStep);
                             Write(Menu[Point]);
                           End;
                    UpArrow: Begin
                              Dec(Point);
                              Dec(MenuStep);
                              If Point<1 Then
                                          Begin
                                             Point:=DoingNo;
                                             MenuStep:=DoingNo-1;
                                          End;
                              TextColor(White);
                              TextBackGround(LightRed);
                              GotoXY(30,4+MenuStep);
                              Write(Menu[Point]);
                            End;
                       End;
                  End;
                EnterKey:SelectChoice;
                  End;
        Until ExitWilling;
End.
```