T. C.
ULUDAĞ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

80C196KC MİCROCONTROLLER'I İÇİN BİR FFT PROGRAMININ GELİŞTİRİLMESİ

LİSANS TEZİ

SÜRURİ KARAÇORLU

BURSA, HAZİRAN 1992

ÖNSÖZ

Fast Fourier Transform, son yıllarda hızla gelişmekte olan Digital Signal Processing (Dijital Sinyal İşleme) konusunun ayrılmaz bir parçasıdır.

Bu projede okulumuza geçtiğimiz yıl gelen EV80C196KC Evaluation Board yardımıyla bir FFT programı gerçekleştirilmiştir. 80C196KC 'nin çok güçlü bir microcontroller olmasının sağladığı destekle, bu programın, önümüzdeki yıllarda okulumuzda gerçekleştirilecek birçok uygulamada kullanılacağına inanıyorum. Ben, okulumuzda 80C196KC microcontroller ile tanışma fırsatını yakalayabilen sayılı öğrencilerinden biri olarak, benden sonra gelecek arkadaşlarımdan, bizim tecrübelerimizden bu projeleri inceleyerek yararlanmalarını ve bu yolla bizden çok daha iyi projeler çıkarmalarını diliyorum.

Bu tez çalışmasının gerçekleştirilmesinde bana çok fazla yardımı dokunan saygıdeğer hocam Öğr. Gör. Yük. Müh. Haluk Gümüşkaya'ya, sevgili arkadaşlarım B. İrfan Öztürk ve F. İbrahim Cirit'e teşekkür ederim.

ÖZET

Bu çalışmada, 80C196KC Microcontroller'ı için bir FFT programının geliştirilmesi amaçlanmaktadır. Bu konuda öncelikle Fast Fourier Transform (FFT)'nin teorisi üzerinde araştırma yapılmıştır. Bu araştırmanın sonucunda C dilinde, IBM PC ortamında çalışabilecek bir FFT programı geliştirilmiştir. Ardından C programında kullanılan algoritmadan faydalanılarak 80C196KC'nin makine dilinde bir FFT programı yazılmıştır. Makine dilinde yazılan programlar ASM96 Assembler ile derlendikten sonra ECM96 programı aracılığı ile EV80C196KC Evalution Board üzerinde çalıştırılmıştır.

ABSTRACT

In this study, it is intented to develop a FFT software program for 80C196KC Microcontroller. First I studied on the theory and applications of FFT then I developed a C program in IBM PC enviroment on this subject. After that a machine code program is written for 80C196KC by using the algorithm written in C. The machine code programs are compiled in ASM96 Assembler then they are realized on EV80C196KC Evaluation Board by using ECM96 Program.

İÇİNDEKİLER

1 GİRİŞ

2 FAST FOURIER TRANSFORM (FFT) TEORİSİ

2.1 Decomposition in Time (DIT)

2.1.1 DIT Algoritmasının Bilgisayara Uyarlanması

2.2 Decomposition in Frequency (DIF)

2.3 DIT ve DIF Algoritmalarında Yapılabilecek Değişiklikler

3 FFT PROGRAMININ GELİŞTİRİLMESİ

3.1 Akış Şeması

3.2 C Programı Hakkında Genel Açıklama

3.3 ASM96 Programı Hakkında Genel Açıklama

4 80C196KC MICROCONTROLLER'IN TANITIMI ve KOMUT LİSTESİ

5 EV80C196KC EVALUATION BOARD

EK1 C PROGRAMI

EK2 ASM96 PROGRAMI

1.0 GİRİŞ

Intel Firması tarafından geliştirilen 80C196KC microcontroller'ı, önceleri mikroişlemcilerle gerçekleştirilen birçok uygulamayı daha kolay, daha hızlı, daha az donanımla ve daha az yazılımla daha güvenli olarak gerçekleştirebilecek güçlü bir microcontroller'dir.

Bu tez çalışmasında örneklenmiş bir sinyal üzerinde Fast Fourier Transform yapabilecek bir program geliştirilmiştir. Örneklenmiş değerler programda datalar halinde verilebileceği gibi Evaluation Board'ın dışarıdan analog sinyal alıp örnekleyebilmesi özelliğinden de yararlanılabilir.

FFT, bir analog sinyalde hangi frekanslarda işaretlerin bulunduğunu tesbit etmekte kullanılır. Değişik dalga şekillerinin frekans bileşenleri FFT yardımıyla bulunarak, bunların başka sinyallerle karşılaştırmak veya modeller oluşturmak mümkündür. Bu yöntem, konuşma algılayıcıları (speech sensors) veya makine gürültü algılayıcılarında (engine knock sensors) kullanılır. FFT, ayrıca cisimleri tanıyan görüntü algılayıcı sistemlerde ve radar sistemlerinde hareket eden nesneleri tanımakta kullanılır.

Bu tez çalışmasında, 80C196KC microcontroller'ı ve EV80C196KC Evaluation Board ile ilgili fazla ayrıntıya girilmesi durumunda esas konudan uzaklaşılacağından, bu konulardaki açıklamaların kısa olarak verilmesi uygun görülmüştür.

2.0 FAST FOURIER TRANSFORM (FFT) TEORİSİ

Ayrık Fourier Dönüşümü olarak bilinen Discrete Fourier Transform (DFT)'nin temel dayanağı, bir fonksiyonun kompleks exponansiyellerin veya başka bir deyişle sinüslerin toplamı olarak gösterilebilmesidir.

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

Bu denklem yardımıyla x(n) ile gösterilen giriş dizisinin DFT'si elde edilir. N, bir periyot içinde alınan örnek sayısına karşılık gelir.

1 numaralı denklemden de anlaşılabileceği gibi DFT algoritmasında $N.N=N^2$ tane kompleks çarpma işlemine ihtiyaç duyulmaktadır. FFT algoritmaları kullanılarak daha az sayıda kompleks çarpma işlemi ile aynı iş yapılabilir. Aşağıda Decomposition in Time (DIT) ve Decomposition in Frequency (DIF) olarak bilinen iki çeşit FFT algoritması incelenmiştir.

2.1 Decomposition In Time (DIT)

Burada N=2,4 ve 8 nokta için DFT hesaplamalarını inceleyeceğiz. (N ; 2'nin katı olan bir tamsayıdır) 1 numaralı denklemi aşağıdaki gibi yazabiliriz:

$$X(k)=\sum_{n=0}^{N-1} x(n).(W_N)^{nk} \qquad k=0,1,2,3,...,N-1 \qquad (2)$$

$$W_N=e^{-j2\pi/N}$$

$[W_N]^{nk}$ *Agirlik Faktoru*

2 noktalı DFT için aşağıdaki denklemi yazabiliriz:

$$X(k)=\sum_{n=0}^{1} x(n).W_2^{nk}$$

$$X(k)=x(0).W_2^{0k}+x(1).W_2^{1k} \qquad k=0,1 \qquad (3)$$

Burada

$W_2 = e^{-j(2\pi/2)} = e^{-j\pi} = -1$ şeklindedir. n=0 ve n=1 için

$$W_2^{0k} = (-1)^{0k} = 1$$

$$W_2^{1k} = (-1)^k$$

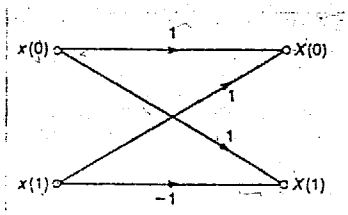
şeklinde ifade edebiliriz. Böylece,

$$X(0) = x(0).1 + x(1).1 = x(0) + x(1)$$

$$X(1) = x(0).1 + x(1).(-1) = x(0) - x(1) \qquad (4)$$

elde edilir.

Bu denklemler için kompleks çarpmalara ihtiyaç duyulmamaktadır. Elde edilen bu denklemler, şekil 1'deki sinyal akiş grafigi ile gösterilebilir.



ŞEKİL 1: 2 noktalı DFT'yi tanımlamakta kullanılan ve BUTTERFLY olarak bilinen sinyal akış grafigi

Bu grafik 4 numaralı denklemlerin şekille ifadesidir ve girişler x(0) ile x(1) ve çıkışlar X(0) ile X(1) olarak gösterilmiştir. Kollar giriş ve cıkışları birbirine bağlar ve üzerlerine çarpanlar yazılmıştır. Bu gösterimde çarpanlar (+1) veya (-1)'dir. Şekil 1'deki yapıya BUTTERFLY (kelebek) denir.

4 noktalı DFT için temel denklem şu şekildedir:

$$X(k) = \sum_{n=0}^{n=3} x(n).W_4^{nk} \qquad k = 0,1,2,3,4$$

$$=x(0).W_4^{0k}+x(1).W_4^{1k}+x(2).W_4^{2k}+x(3).W_4^{3k} \quad (5)$$

5. denklemdeki ilk terim şu şekilde yazılabilir:

$$W_4^{0k}=e^{-j(2\pi/4)0k}=e^{-j(2\pi/2)0k}=W_2^0 \quad (6)$$

Aynı şekilde 3. terimi de şu şekilde yazabiliriz:

$$W_4^{2k}=e^{-j(2\pi/4)2k}=e^{-j(2\pi/2)k}=W_2^k \quad (7)$$

5 numaralı denklemin 1. ve 3. terimlerinin birleştirilmesiyle,

$$x(0)W_4^{0k}+x(2)W_4^{2k}=x_1(0)W_2^{0k}+x_1(1)W_2^k \quad (8)$$

elde edilir. Burada yeni bir dizi olan

$$x_1(n)=x(2n) \qquad n=0,1 \quad (9)$$

dizisi karşımıza çıkar. 8. ve 3. denklemlerin karşılaştırılmasıyla,

$$X_1(1)=x_1(0)W_2^{0k}+x_1(1)W_2^{1k} \quad (10)$$

denklemi bulunur. Bu denklem 9. denklemde tanımlanan $x_1(n)$ dizisinin 2 noktalı DFT'sidir.

5. denklemde kalan iki terim:

$$x(1)W_4^k+x(3)W_4^{3k}=W_4^k[x(1)W_4^{0k}+x(3)W_4^{2k}] \quad (11)$$

şeklindedir. 7. denklemde elde edilen

$$W_4^{2k}=W_2^k$$

eşitliği 11. denklemde yerine konursa:

$$x(1)W_4^k+x(3)W_4^{3k}=W_4^k[x_2(0)W_2^{0k}+x_2(1)W_2^k]=W_4^kX_2(k) \quad (12)$$

bulunur. Burada yeni bir dizi olarak

$x_2(n)=x(2n+1) \quad n=0,1$ (13) kullanılmıştır.

10. ve 12. denklemlerin birleştirilmesiyle:

$$x(n)=x_1(n)+x_2(n)$$

şeklinde verilen dizinin 4 noktalı DFT'sı olan

$$X(k)=x_1(0)W_2^{0k}+x_1(1)W_2^{k}+W_4^{k}[x_2(0)W_2^{0k}+x_2(1)W_2^{k}] \quad (14)$$

denklemi elde edilir. Bu denklem şu şekilde de ifade edilebilir:

$$X(k)=X_1(k)+W_4^{k}X_2(k) \quad k=0,1,2,3 \quad (15)$$

15. denklemi kelimelerle anlatmak gerekirse:

[x(n)'in 4 noktali DFT'si] =[x_1(n)'in 2 noktali DFT'si]

$$+W_4^{k}[x_2(n)'in\ 2\ noktali\ DFT'si] \quad (16)$$

Böylece 4 noktalı DFT'yi iki tane 2 noktalı DFT olarak ifade etmiş olduk. 2 noktalı DFT'nin periyodik oluşu nedeniyle:

$$X_1(k+2)=X_1(k)$$

$$X_2(k+2)=X_2(k) \quad k=0,1 \quad (17)$$

şeklindedir ve X(k)'yı bu iki dizinin toplamı şeklinde ifade ederek 15. denklemde k'ya değerler vererek aşağıdaki denklemleri elde edebiliriz:

$$X(0)=X_1(0)+W_4^{0}X_2(0)=X_1(0)+X_2(0) \quad (18)$$

$$X(1)=X_1(1)+W_4^{1}X_2(1) \quad (19)$$

Bu denklemleri yazarken 17.denklem ile tanımlanan $X_1(k)$ ve $X_2(k)$'nın periyodikliğinden ve

$$X(2)=X_1(2)+W_4^2X_2(2)=X_1(0)-X_2(0) \quad (20)$$

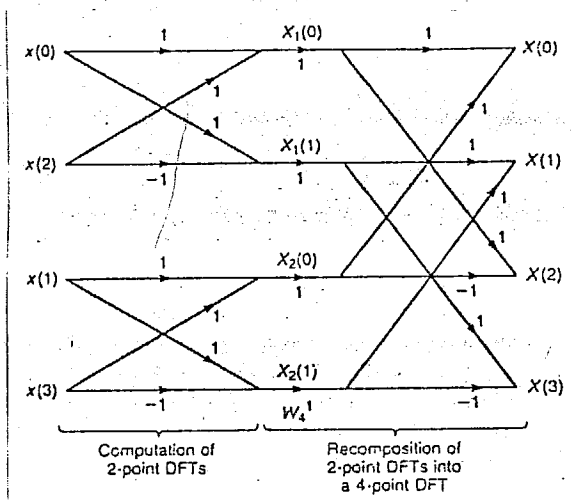
$$X(3)=X_1(3)+W_4^3X_2(3)=X_1(3)+W_4^1W_4^2X_2(3)=X_1(1)-W_4^1X_2(1) \quad (21)$$

$$W_4^2=-1$$

eşitliğinden yaralandık.

18.den 21.ye kadar olan denklemlere recomposition "birleştirme denklemleri" denir.

16. ve 18-21. denklemlerin sonuçları şekil 2'de sinyal akiş grafiği ile gösterilmiştir. Şekil 2'de verilen yapı butterfly'lardan oluşmaktadır ve burada sadece bir tane kompleks çarpma işlemine ihtiyaç duyulur. Bu işlem ise W_4^1'in hesaplanmasıdır.



ŞEKİL 2: 4 noktalı DFT

Örneğin x(n)={0,1,2,3} olarak verilen bir dizinin 4 noktalı DFT'sini şekil 2'yi kullanarak bulabiliriz:

İlk olarak aşağıdaki eşitlikler hesaplanır.

$$X_1(0)=x(0)+x(2)=0+2=2$$

$$X_1(1)=x(0)-x(2)=0-2=-2$$

$$X_2(0)=x(1)+x(3)=1+3=4$$

$$X_2(1)=x(1)-x(3)=1-3=-2 \qquad (22)$$

Şekil 2'den yararlanarak DFT'ler şu şekilde hesaplanabilir:

$$X(0)=X_1(0)+X_2(0)=2+4=6$$

$$X(1)=X_1(1)+W_4^1X_2(1)=-2+e^{-j(2\pi/4)}(-2)=-2+j2=2\sqrt{2}e^{j3\pi/4}$$

$$X(2)=X_1(0)-X_2(0)=2-4=-2$$

$$X(3)=X_1(1)-W_4^1X_2(1)=-2-e^{-j(2\pi/4)}(-2)=-2-j2$$

$$=2\sqrt{2}e^{j(5\pi/4)} \qquad (23)$$

Bu sonucu değerlendirecek

$$X(0)=\sum_{n=0}^{3}x(n)=6$$

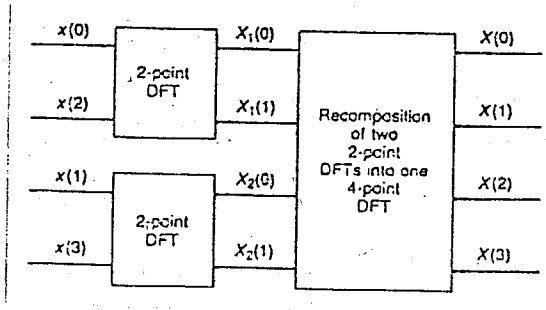
olduğunu görebiliriz.

$k=N/2=2$

noktasında simetriden dolayı şu eşitlik vardır:

$$X(1)=X^*(3)$$

Şekil 2'deki gösterim Şekil 3'de blok diyagramlarla ifade edilmiştir.



ŞEKİL 3: 4 noktalı DFT hesaplarının blok diyagram gösterimi

8 noktalı DFT'yi inceleyecek olursak

$$X(k)=\sum_{n=0}^{7} x(n)e^{-j(2\pi/8)nk}=\sum_{n=0}^{7} x(n)W_8^{nk} \qquad k=0,1,...,7 \quad (24)$$

şeklinde verilen temel denklemi elde edebiliriz. Bu denklemde n'e 0...7 arası değerler vererek,

$$X(k)=x(0)W_8^{0k}+x(1)W_8^{1k}+x(2)W_8^{2k}+x(3)W_8^{3k}$$

$$+x(4)W_8^{4k}+x(5)W_8^{5k}+x(6)W_8^{6k}+x(7)W_8^{7k} \qquad (25)$$

denklemini elde edebiliriz. 4 noktalı DFT'de olduğu gibi tek ve çift n'lerden oluşan x(n) terimlerini biraraya getirirsek aşağıdaki denklemi elde ederiz:

$$X(k)=x(0)W_8^{0}+x(2)W_8^{2k}+x(4)W_8^{4k}+x(6)W_8^{6k}$$

$$+x(1)W_8^{k}+x(3)W_8^{3k}+x(5)W_8^{5k}+x(7)W_8^{7k} \qquad (26)$$

Ağırlık Faktörlerinin tanımından gelen bir özelliği burada kullanacağız:

$$e^{-j(2\pi/N)2nk}=e^{-j(2\pi/N/2)nk} \qquad (27)$$

Yukarıdaki eşitliği ağırlık faktörleri cinsinden yazacak olursak,

$W_N^{2nk}=W_{N/2}^{nk}$ (28) | eşitliğini elde ederiz.

26. denklemde W_8^k'yi dört terimden oluşan iki grupla ifade edersek.

$$X(k)=x(0)W_8^0+x(2)W_8^{2k}+x(4)W_8^{4k}+x(6)W_8^{6k}$$

$$+W_8^k[x(1)W_8^0+x(3)W_8^{2k}+x(5)W_8^{4k}+x(7)W_8^{6k}] \qquad (29)$$

ve 28. denklemi de kullanırsak,

$$X(k)=x(0)W_4^0+x(2)W_4^k+x(4)W_4^{2k}+x(6)W_4^{3k}$$

$$+W_8^k[x(1)W_4^0+x(3)W_4^k+x(5)W_4^{2k}+x(7)W_4^{3k}] \qquad (30)$$

denklemini elde ederiz. Aşağıdaki diziler tanımlanarak

$$x_1(n)=x(2n) \qquad n=0,1,2,3$$

$$x_2(n)=x(2n+1) \qquad n=0,1,2,3 \qquad (31)$$

şu denklem yazılabilir:

$$X(k)=x_1(0)W_4^0+x_1(1)W_4^k+x_1(2)W_4^{2k}+x_1(3)W_4^{3k}$$

$$+W_8^k[x_2(0)W_4^0+x_2(1)W_4^k+x_2(2)W_4^{2k}+x_2(3)W_4^{3k}] \qquad (32)$$

Bu denklemin genel bir ifadesi,

$$X(k)=\sum_{n=0}^{3}x_1(n)W_4^{nk}+W_8^k\sum_{n=0}^{3}x_2(n)W_4^{nk} \qquad k=0,1,...,7 \qquad (33)$$

şeklinde verilebilir.

Birinci toplamı, $x_1(n)$ dizisinin 4 noktalı DFT'si ve ikinci toplamı, $x_2(n)$ dizisinin 4 noktalı DFT'si ile W_8^k'nin çarpımı şeklinde ifade edersek, 8 noktalı DFT'ye ait şu denklemi bulabiliriz:

$$X(k)=\sum_{n=0}^{N/2-1} x_1(n)W_{N/2}^{nk}+W_N^k\sum_{n=0}^{N/2-1} x_2(n)W_{N/2}^{nk}$$

$$=\sum_{n=0}^{3} x(2n)W_4^{nk}+W_8^k\sum_{n=0}^{3} x(2n+1)W_4^{nk}$$

$$=X_1(k)+W_8^kX_2(k) \qquad k=0,1,...,7 \qquad (34)$$

Burada dikkat edilmesi gereken nokta $X_1(k)$ ve $X_2(k)$'nın 4 noktalı DFT'ye ait diziler olmasına rağmen k'nın halen 0,1,...,7 aralığında değerler almasıdır. 34. denklemden yola çıkarak aşağıdaki sonuçları elde edebiliriz:

$$X(0)=X_1(0)+W_8^0X_2(0)=X_1(0)+X_2(0)$$

$$X(1)=X_1(1)+W_8^1X_2(1)$$

$$X(2)=X_1(2)+W_8^2X_2(2)$$

$$X(3)=X_1(3)+W_8^3X_2(3)$$

$$X(4)=X_1(4)+W_8^4X_2(4)=X_1(4)-X_2(4)$$

$$X(5)=X_1(5)+W_8^4W_8^1X_2(5)=X_1(5)-W_8^1X_2(5)$$

$$X(6)=X_1(6)-W_8^2X_2(6)$$

$$X(7)=X_1(7)-W_8^3X_2(7) \qquad (35)$$

X(4)...X(7)'nin aldığı değerler hesaplanırken,

$$W_N^{(n+l)k}=W_N^{nk}W_N^{lk}$$

$$W_N^{N/2}=-1$$

özelliklerinden faylanılmıştır. Ayrıca

$$X_1(k)=X_1(k+4)$$

ve

$$X_2(k)=X_2(k+4) \qquad k=0,1,2,3 \qquad (37)$$

eşitlikleri X(4)...X(7)'nın hesaplanmasında kullanılırsa 33. denklemle verilen eşitlikler aşağıdaki hale dönüşür:

$$X(0)=X_1(0)+X_2(0) \qquad X(4)=X_1(0)-X_2(0)$$

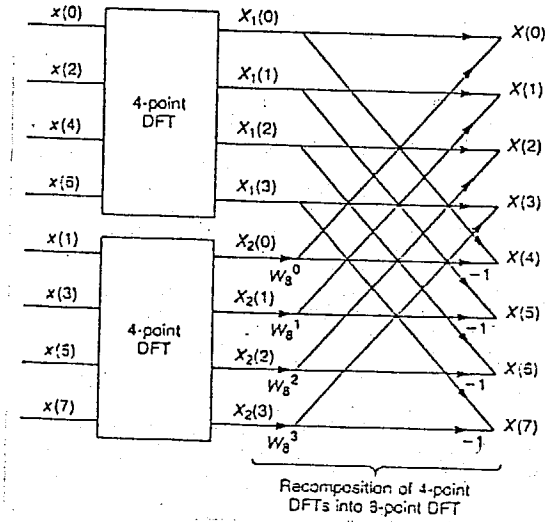
$$X(1)=X_1(1)+W_8^1X_2(1) \qquad X(5)=X_1(1)-W_8^1X_2(1)$$

$$X(2)=X_1(2)+W_8^2X_2(2) \qquad X(6)=X_1(2)-W_8^2X_2(2)$$

$$X(3)=X_1(3)+W_8^3X_2(3) \qquad X(7)=X_1(3)-W_8^3X_2(3)$$

(38)

Bu denklemlere birleştirme "recomposition" denklemleri denir. Bu denklemlerin sinyal akış grafiği şekil 4'de gösterilmiştir.



ŞEKİL 4: 8 noktalı DFT'nin, iki tane 4 noktalı DFT'nin birleşimi şeklinde hesaplanması

Şimdi de, şekil 4'de gösterilen her 4 noktalı DFT için ayrıştırma işlemini gerçekleştirelim. Bu işlemi 4 noktalı DFT konusunda anlatıldığı gibi gerçekleştirirsek, aşağıdaki denklemleri elde edebiliriz:

$$X_1(k)=[x(0) \text{ ve } x(4)'\text{un } 2\text{-point-DFT'si}]$$

$$+W_4^k[x(2) \text{ ve } x(6)'\text{nin } 2\text{-point-DFT'si}]$$

$$=X_{1a}(k)+W_4^kX_{1b}(k) \qquad k=0,1,2,3 \qquad (39)$$

ve

$$X_2(k)=[x(1) \text{ ve } x(5)'\text{in } 2\text{-point-DFT'si}]$$

$$+W_4^k[x(3) \text{ ve } x(7)'\text{nin } 2\text{-point-DFT'si}]$$

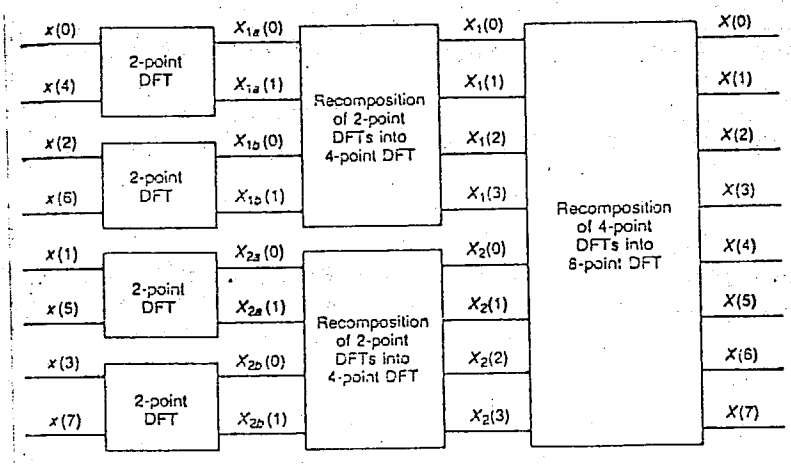
$$=X_{2a}(k)+W_4^kX_{2b}(k) \qquad k=0,1,2,3 \qquad (40)$$

39 ve 40 no.lu denklemlerden aşağıdaki birleştirme denklemleri elde edilir:

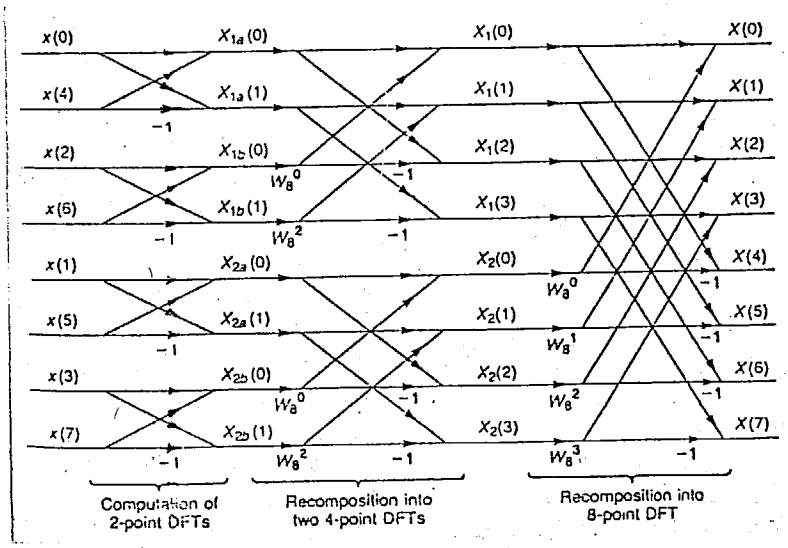
Şekil 5'de işlemin bütün basamaklarının blok diyagramı ve detaylı sinyal akış yolları gösterilmiştir. Şekil 5.b'ye dikkat edilirse, işlemleri basitleştirmek için bütün ağırlık faktörlerinin W_N'in kuvvetleri olan terimlerle ifade edilmiş olduğu görülebilir. Bu, 28. denklemde verilen,

$$W_N^{2nk} = W_{N/2}^{nk}$$

özelliği kullanılarak yapılır.

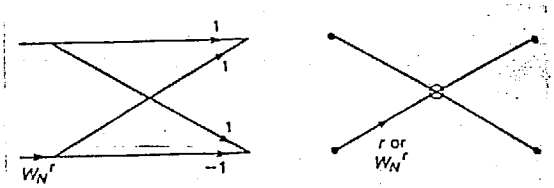


ŞEKİL 5.a: 8 noktalı DFT'nin, 2 noktalı ve 4 noktalı DFT'lerin birlesimi seklinde hesaplanmasının blok diyagramla gösterimi

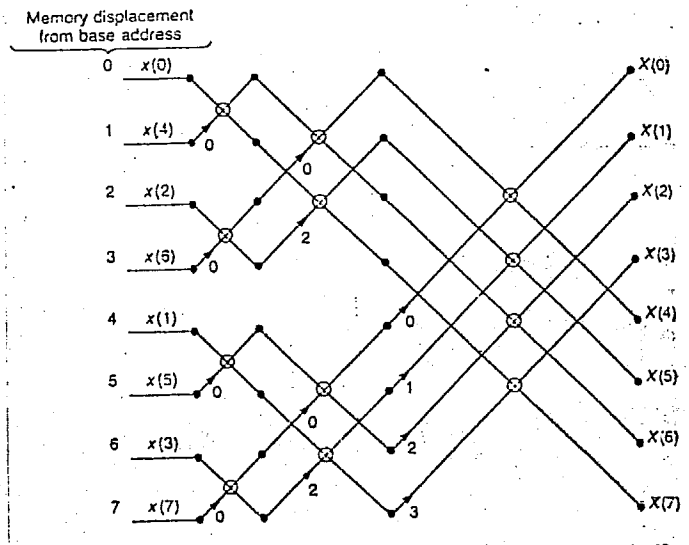


ŞEKİL 5.b: 8 noktalı DFT'nin, 2 noktalı ve 4 noktalı DFT'lerin birleşimi şeklinde hesaplanmasının sinyal akış grafiği ile gösterimi

Şekil 5.b'deki sinyal akış yollarının oldukça karmaşık olması nedeniyle şekil 6.a'daki butterfly hesaplarının kısa gösterimi yolu tercih edilir. Bu metot kullanılarak şekil 6.b çizilmiştir.



ŞEKİL 6.a: Butterfly hesaplarının akış grafiği ve kısa gösterimi



ŞEKİL 6.b: Kısa gösterim kullanılarak çizilmiş 8 noktalı FFT

Şimdi, DFT'nin tanım denklemiyle yapılan hesaplarda gereken ve bulduğumuz denklemlerle yapılan hesaplarda gereken işlem sayısını karşılaştıralım:

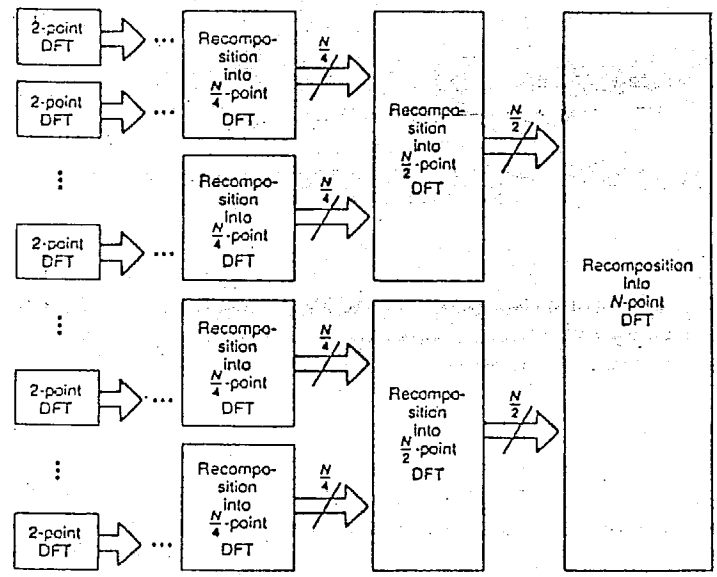
DFT'nin tanımında,

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

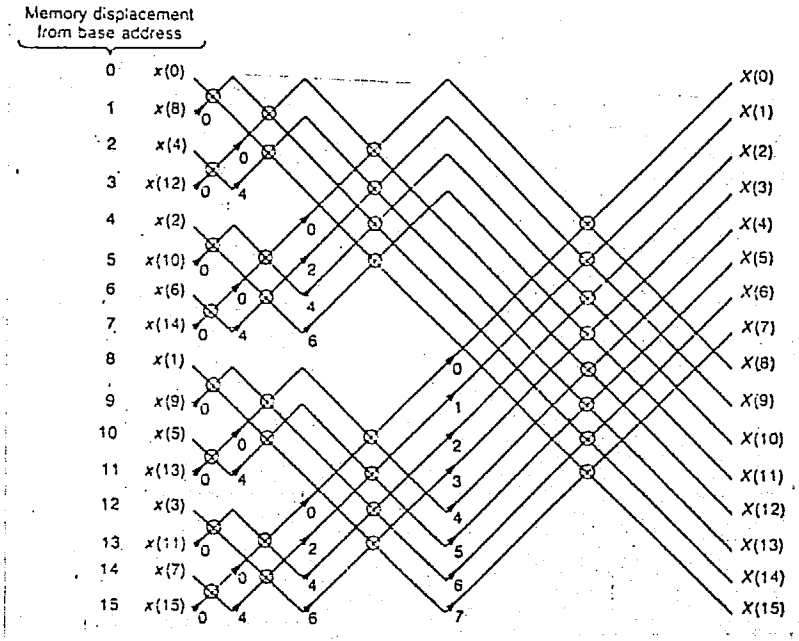
$$=\sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad k=0,1,...,N-1 \qquad (42)$$

olarak verilmişti.

Şekil 6.b ve şekil 8'e bakılırsa hesapların çeşitli adımlara bölündüğü görülebilir. Bulduğumuz denklemlerle yapılan hesaplarda N/2 tane 2 noktalı DFT işlemine ihtiyaç vardır.



ŞEKİL 7: N noktalı DFT'nin 2 noktalı DFT'lerin birleşimi şeklinde hesaplanması



ŞEKİL 8 : 16 noktalı FFT. Bit-reversed girişler ve sıralı çıkışlar kullanılmıştır.

Ayrıca,

$$\frac{N}{2^R}=2 \qquad (43)$$

denklemiyle tanımlanan R kadar birleştirme adımına ihtiyaç olacaktır. 43. denklemin 2 tabanına göre logaritması alınırsa,

$$log_2 N - log_2 2^R = log_2 2 \qquad (44)$$

$$log_2 N - R = 1$$

ve

$$R = log_2 N - 1 \qquad (45)$$

bulunur. Bundan dolayı 2 noktalı DFT'nin hesabı için gereken toplam adım sayısı R+1 tanedir ya da

$$adim\ sayisi = \log_2 N \qquad (46)$$

şeklinde ifade edilebilir.

Her adımda N/2 tane butterfly hesabı gerekli olduğu için $\log_2 N$ adımda toplam T_B tane butterfly gereklidir.

$$Butterfly\ sayisi = \frac{Butterfly\ sayisi}{adim}.adim\ sayisi \qquad (47)$$

$$T_B = \frac{N}{2}\log_2 N\ ,\ N:\ 2'nin\ herhangi\ bir\ kuvveti \qquad (48)$$

Genel olarak her bir butterfly için bir kompleks çarpma işlemi (genellikle bir toplama ve bir çıkarma işlemi) yapılır. Böylece DIT algoritması kullanılırsa gerekli olan kompleks çarpma işlemi sayısı 48. denklemden bulunabilir.

DFT hesaplarında kullanılan standart denklem daha önce verilmişti:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad (49)$$

Bu yöntem kullanılırsa, denklemden de anlaşılabileceği gibi N^2 tane kompleks çarpma işlemine ihtiyaç duyulur. Standart DFT hesaplarında ve DIT algoritması için gerekli kompleks çarpma işlemi sayılarının karşılaştırılması Tablo-1'de verilmiştir.

N	Standart DFT	FFT	DFT/FFT oranı
2	4	1	4.0
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.3
128	16384	448	36.6
256	65536	1024	64.0
512	262144	2304	113.8
1024	1048576	5120	204.8
....	....	....	....
$2^{n}=10^{j}$	$=10^{n}$	$=10^{j}$	$=10^{j}$

TABLO 1: Standart DFT ve DIT için gerekli olan kompleks çarpma işlemi sayılarının karşılaştırılması

Gördüğümüz gibi FFT, standart DFT'ye göre daha hızlı hesaplama sağlamaktadır.

Biraz önce anlatılan DIT algoritması ile çalışan bir bilgisayar programı yazmak için ağırlık faktörlerinin nasıl üretileceğini gösteren formüllere ve hangi noktaların diğer butterfly'larla karşılıklı ilişkisi olduğunu araştırmamız gerekmektedir. Şekil 6.b ve Şekil 8'in incelenmesi ile ağırlık faktörlerinin periyodik olduğunu ve her periyodik şeklin başlangıcının 0 olduğunu görebiliriz. Eğer Butterfly basamaklarını 1'den başlatırsak ve 2'den devam ederek $\log_2 N$'e kadar ilerletirsek ağırlık faktörleri arasındaki arttırmada W_N^{α} ile gösterilen α, $\alpha=N/2^{L}$ ile L. basamağa işaret eder. Şimdi giriş dizisi değerlerinin, sinyal akış grafiklerinde olduğu sırasıyla hafızada yer aldığını farzedelim. Örnek olarak şekil 8'de, x(0) taban adresinde, x(8) taban adresi+1'de, x(4)

taban adresi+2'de yer alır. Böylece x(0) ve x(8) birbirlerinin hafıza bölgelerinden ayrılmıştır. Aynı şekilde x(0) ve x(4) de ayrılmıştır. Butterfly aralıklarının L. basamağı, görüldüğü gibi $2^{(L-1)}$'dir ve butterfly hesaplarında yer alan noktaların hafıza adreslerinin ayrılması bu yolla olur.

Ayrıca, giriş verilerinin sırasının da adreslenmesi gereklidir. Şekil 6.b ve şekil 8'i tekrar incelersek çıkışın bizim istediğimiz gibi normal sırasında olduğunu fakat girişin sıralı olmadığını görebiliriz. Giriş verilerini de sıralı hafıza adreslerine koymalıyız. Bu işlem için x(0),x(1),.....,x(N-1) değerlerini şekil 6.b'de ve şekil 8'de gösterilen hafıza adreslerinde saklamalıyız. Bu işlem, "bit-reversing" olarak bilinen bir algoritma kullanarak yapılır. Şimdi, bu düşünceyi ifade etmek için binary formda bir dizi ele alalım. Eğer, dizi 16 elemanlı ise adresler

0000,0001,0010,0011,...,1111 (50)

şeklinde olmalıdır. Dizi değerleri ise

$x(0),x(1),x(2),x(3),...,x(15)$ (51)

sırasında olmalıdır.

Dizinin bir elemanının hafıza adresini hesaplamak için, basitçe, onun binary adresinin bitlerini tersine çeviririz. Örneğin:

$x(3)=x(0011) \rightarrow 1100$ *veya decimal* 12.

Böylece x(3), 12. hafıza adresinde saklanır. Tablo 2'de 16 noktalı bir dizi için ayrılmış hafıza adresleri verilmiştir. Bu tabloyu şekil 8 ile karşılaştırarak bu algoritmanın ayrılmış hafıza adreslerinin hesaplanmasında kullanışını görebiliriz.

Sıralı eleman	Binary adres	Bit-reversed adres	Taban adresi + bit-reversed adres
x(0)	0000	0000	0
x(1)	0001	1000	8
x(2)	0010	0100	4
x(3)	0011	1100	12
x(4)	0100	0010	2
x(5)	0101	1010	10
x(6)	0110	0110	6
x(7)	0111	1110	14
x(8)	1000	0001	1
x(9)	1001	1001	9
x(10)	1010	0101	5
x(11)	1011	1101	13
x(12)	1100	0011	3
x(13)	1101	1011	11
x(14)	1110	0111	7
x(15)	1111	1111	15

TABLO 2 : Bit-reversed düzeninin hesaplanması

2.1.1 DIT Algoritmasının Bilgisayara Uyarlanması

Şimdi de DIT algoritması için bilgisayar programlarının nasıl geliştirileceğine bakalım. Basitçe, iki ana bölüme ihtiyacımız var. Birincisi "bit-reversing" işlemlerini gerçekleştirmek, ikincisi butterfly hesaplamaları için adımları gerçekleştirmek. İlk olarak bit-reversing işlemlerini gerçekleştirelim:

Eğer giriş dizisi olarak elimizde N nokta varsa (N'in 2'nin herhangi bir kuvveti olduğuna göre), bu noktaların yerleşeceği adresler aşağıdaki gibi olmalıdır:

$$A=b_{M-1}2^{M-1}+b_{M-2}2^{M-2}+\ldots+b_1 2^1+b_0 2^0 \quad (53)$$

Burada $M=\log_2^2 N$ ve b_i için i=0,1,....,M'dir. b_i ya 0 ya da 1'dir. Bu adresin bit-reversed hali aşağıdaki gibi olur:

$$A_R=b_0 2^{M-1}+b_1 2^{M-2}+\ldots+b_{M-2}2^1+b_{M-1}2^0 \quad (54)$$

0,1,2,...,N-1 şeklindeki desimal adresleri kullanabilmek için önce binary katsayılar olan $b_0,b_1,\ldots,b_M$'i hesaplamalı ve 54. denklemdeki aritmetik işlemleri yaparak yeni adresleri oluşturmalıyız. Örnek olarak şekil 6'daki gibi verilmiş olan 8 noktalı bir giriş dizisi tanımlanmış olsun. $M=\log_2^2 8=3$ bulunur ve bu değer aynı zamanda butterfly basamaklarının sayısıdır. x(6) veri değeri için binary adres 110'dur ve bunun kısaca gösterimi $1*2^2+1*2^1+0*2^0=6$ (desimal)'dir. Bit-reversed adres ise $0*2^2+1*2^1+1*2^0=3$'tur. Şekil 6'dan da görüldüğü gibi x(6), taban adresinden sonraki 3. adrese yerleştirilmiştir.

Bu işlemleri yapan bir program yapabilmek için $b_0,b_1,\ldots,b_{M-1}$ katsayılarını hesaplamaya ihtiyacımız vardır. En düşük değerliğe sahip b_0 bit'i için desimal adres 2'ye bölünür ve kalan değeri b_0'a konur. Kalanın daima 0 veya 1 olacağına dikkat etmeliyiz. Bazı programlama dillerinde (örneğin FORTRAN) kalanı hesaplamak için özel fonksiyonlar mevcuttur. Biz, böyle bir fonksiyonun var olduğunu kabul edelim ve bu fonksiyona MODula kelimesinden gelen MOD ismini verelim. Bu fonksiyonun iki argümanı vardır ve aşağıdaki gibi kullanılır:

x <--- MOD(A<2)

A'nın 2'ye bölünmesinde elde edilecek kalan değeri x'e konur. Önce b_0, ardından b_1 hesaplanır. Bunun için orijinal desimal adres ikiye bölünür. Kalan ayrılır ve tekrar MOD fonksiyonu kullanılır. Bunun neden yapıldığını anlayabilmek için 53. denklemin ikiye bölünmesi ile elde edilen aşağıdaki

denklemi inceleyelim:

$$b_{M-1}2^{M-2}+b_{M-2}2^{M-3}+\ldots+b_1 2^0+b_0 2^{-1} \qquad (55)$$

Bu denklemde en sağdaki eleman, kalanı verir. b_i'i bulabilmek için, kalanı ayırıp $A=b_{i-1}2^{i-1}+b_{i-2}2^{i-2}+\ldots+b_i 2^0$ şeklinde bir fonksiyon elde etmeli ve MOD(A',2) işlemi ile kalanı b_i olarak almalıyız. Ardından $b_0, b_1, \ldots, b_{i-1}$ katsayılarını bulmak için bu işleme devam etmeliyiz.

b_i'lerin hesaplanması amacıyla 54. denklemdeki toplamlar için kısmi sonuçlar elde ederek programın etkinliğini arttırabiliriz.

Az önce anlatılan yöntemle bit-reversal değerlerinin elde edilmesini gösteren Pseudocode'lar (sahte,yalancı kod) aşağıda verilmiştir.

($M=\log_2 N$'dir. Giriş verileri bir boyutlu kompleks dizi olan XTMP(0),...,XTMP(N-1) dizisine yerleştirilmektedir. Ve bit-reversal çıkış verileri yine bir boyutlu kompleks dizi olan x(0),x(1),....,x(N-1) dizisine yerleştirilmektedir.)

```
DO FOR K<--0 TO N-1
(Yeni adres ve geçerli adresin hazırlanması)
   NEWADR<--0
   MADDR<--K
   DO FOR I<--0 TO M-1
      LRMNDR<--MOD (MADDR,2)
      NEWADR<--NEWADR+LRMNDR·2^(M-I-1)
      MADDR<--MADDR/2
      (Tamsayı bölme yapıldığı için kalan oluşmuyor.)
   END DO
   X(NEWADR)<--XTMP(K)
END DO
```

Programın butterfly'ları değerlendiren kısmı içiçe üç döngüden oluşmuştur. Bunlardan biri "stage low" (düşük adım)'dır. Örneğin 8 noktalı FFT için 3 tane butterfly basamağı olacaktır. Diğer iki döngü ise butterfly hesaplarında yer almaz ve ağırlık faktörü W_N'in uygun bileşenlerini düzenlemek için hafızadan noktaların seçilmesi ile ilgilidir. Şekil 6'ya bakarsak butterfly hesaplarında verilen bir adımda işlem yapmak için en az iki yol olduğunu farzedebiliriz. Bunlardan birincisi basitçe en baştan en sona (veya en sondan en başa) doğru hafızadaki noktaları almak ve $W_N^p=e^{-j(2\pi p/N)}$ ağırlık faktörünü

hesaplamak ve butterfly çıktılarını aşağıdaki denkleme göre düzenlemektir:

$$XNEW(TOP)=XOLD(TOP)+W_N^R \cdot XOLD(BOT)$$

$$XNEW(BOT)=XOLD(TOP)-W_R^N \cdot XOLD(BOT) \qquad (56)$$

Bu denklemlerde TOP:En yüksek mertebe

BOT=BOTTOM:En alçak mertebe manalarına gelmektedir.

Bu tür hesaplamanın dezavantajı, kompleks değerli ağırlık faktörlerinin her adımda tekrar tekrar hesaplanmak zorunda oluşudur. Bundan kaçınmanın yolu $|W_N^R$ değerini verilen bir adım için hesaplamak ve bütün butterfly'larda bu ağırlık faktörünü kullanmaktır. Örneğin, şekil 8'de ikinci adımda önce $|W_{16}^{4}$ ağırlık faktörünü kullanan 4 butterfly'ı hesaplayıp, ardından $|W_{16}^{1}$'u kullanan 4 butterfly'ı hesaplayacağız ve bu işlem diğer adımlarda devam edecek. İşte kullanacağımız yöntem budur.

IWIDTH'i bir butterfly'da özel olarak ayrılmış hafıza noktalarını ifade etmek için ve ISPACE'i aynı ağırlık faktörlerini kullanan tepe noktalarının ayrılmasını ifade etmek için kullanıyoruz. Şekil 8'e tekrar bakarsak ikinci adımda IWIDTH=2 ve ISPACE=4 olduğunu görebiliriz. Aynı şekilde üçüncü adımda IWIDTH=4 ve ISPACE=8 dir. Genel olarak L. adımda:

$$IWIDTH=2^{L-1} \qquad (57)$$

$$ISPACE=2^{L} \qquad (57)$$

olduğu görülebilir.

Ayrıca $W_N=e^{-j2\pi/N}$'in bileşeni olan miktar, ardarda olan butterfly'larda L. adımda

$$S=N/2^{L} \qquad (59)$$

kadar değişir.

Ayrıca 56. denklemin hafıza kullanımını kolaylaştıran eşdeğer biçimi yazılabilir. Önce, bir butterfly, kendisinin giriş değerleri ile ifade edilebilir. 56. denklem aşağıdaki gibi yazılabilir:

$$TMP = W_N^R \cdot X(BOT)$$

$$X(BOT) = X(TOP) - TMP$$

$$X(TOP) = X(TOP) + TMP \qquad (60)$$

60. denklemin son iki eşitliğinin yazılım sırası önemlidir. Bu iki eşitliğin yer değiştirmesi ile aynı sonuç elde edilemez.

Aşağıda FFT programının butterfly değerlendirmesinin pseudocode'u verilmiştir:

(giriş verilerinin bit-reversed şeklinde x(0),x(1),... ..x(N-1) kompleks dizisinde saklandığı farzediliyor).

```
PI<--3.141593
DO FOR L<--1 TO M
   ISPACE<--2^L
   S<--N/2^L
   IWIDTH<--2^(L-1)
      (Yukarıdaki üç komut şu şekilde yazılırsa daha kolay
hesaplanabilir:ISPACE<--2^L;s<--N/ISPACE;IWIDTH<--ISPACE/2)
   DO FOR J<--0 TO (IWIDTH-1)
      R<--S*J
      ALPHA<--2*PI*R/N
      WTFAC<--CMPLX (COS(ALPHA),-SIN(ALPHA))
      DO FOR ITOP<--J TO (N-2) BY STEPS OF ISPACE
      IBOT<--ITOP+IWIDTH
      TMP<--X(IBOT)*WFAC
      X(IBOT)<--X(ITOP)-TMP
      X(ITOP)<--X(ITOP)+TMP
      END DO
   END DO
END DO
```

Programda (SIN) ve (COS) fonksiyonlarının ve a ile b reel değerlerini alarak a+jb kompleks değerini üreten CMPLX(a,b) fonksiyonunun var olduğu kabul edilmiştir. Çalışır bir FFT programının ortaya konmasında şekil 10 ve şekil 11'de verilen kodlar ile giriş değerlerini okuyan ve giriş-çıkış değerlerini yazan ve/veya çizen uygun kodlar birleştirilir.

2.2 Decomposition in Frequency (DIF) :

FFT işlemlerinde diğer yol da Decomposition-Decimation In Frequency (frekansta ayrıştırma) DIF olarak bilinir. Takip edilen yol Decomposition In Time (DIT) ile benzerdir ve eşit sayıda çarpma kullanır. Bu bölümde basitçe DIF algoritmasının çıkarılmasını anlatacağız ve bu algoritmayı az önce anlatılan DIT yöntemi ile karşılaştıracağız.

Başlangıç noktası DFT'nin tanımındaki ile aynıdır.

$$X(k)=\sum_{n=0}^{N-1} x(n)W_N^{nk} \qquad k=0,1,...,N-1 \qquad (61)$$

DIT algoritmasında, zaman dizisini tek ve çift indisler içeren alt dizilere böleceğiz. DFT'nin alt dizilerinin x(n) dizisinin alt ve üst yarılarının birleşimi olduğunu göz önüne alacağız. Bu işlemi yaparsak DFT'yi aşağıdaki gibi yazabiliriz:

$$X(k)=\sum_{n=0}^{N/2-1} x(n)W_N^{nk}+\sum_{n=N/2}^{N-1} x(n)W_N^{nk} \qquad k=0,1,...,N-1 \qquad (62)$$

İkinci toplam aşağıdaki gibi yazılabilir:

$$\sum_{n=N/2}^{N-1} x(n)W_N^{nk}=\sum_{n=0}^{N/2-1} x(n+\frac{N}{2})W_N^{(n+N/2)k}$$

$$=\sum_{n=0}^{N/2-1} x(n+\frac{N}{2})W_N^{(N/2)k}W_N^{nk} \qquad (63)$$

ek olarak:

$$W_N^{(N/2)k}=e^{-j(2\pi/N)(N/2)k}=e^{-j\pi k}=(-1)^k \qquad (64)$$

Böylece 62 no.lu denklemde verilen X(k), aşağıdaki gibi yazılabilir:

$$X(k)=\sum_{n=0}^{N/2-1} [x(n)+(-1)^k x(n+\frac{N}{2})]W_N^{nk} \qquad k=0,1,...,N-1 \qquad (65)$$

Bu işlemin N/2 noktalı DFT'den tek farkı $W_{N/2}^{nk}$ yerine W_N^{nk} kullanılmış olmasıdır. Toplamanın ikinci teriminde bulunan $(-1)^k$ k'nın çift değerlerinde

1, tek değerlerinde -1 olacaktır. Bu nedenle k'nın tek ve çift değerlerini ayrı ayrı dikkate almak zorundayız. DIF algoritmasında frekansta ayrıştırmayı çift k'larla yaparsak:

$$X(2k)=\sum_{n=0}^{N/2-1}[x(n)+x(n+\frac{N}{2})]W_N^{nk.l} \quad k=0,1,...,\frac{N}{2}-1 \qquad (66)$$

tek k'lar için denklemimiz şu hale gelir:

$$X(2k+1)=\sum_{n=0}^{N/2-1}[x(n)-x(n+\frac{N}{2})]W_N^{(2k+1)n}$$

$$k=0,1,...,\frac{N}{2}-1 \qquad (67)$$

Ayrıca,

$$W_N^{2nk}=e^{-j(2\pi/(N/2))nk}=W_{N/2}^{nk} \qquad (68)$$

$$W_N^{(2k+1)n}=W_N^n\cdot W_N^{2kn} \qquad (69)$$

olduğunu dikkate alırsak 68. denklemi kullanarak

$$W_N^{(2k+1)n}=W_N^n\cdot W_{N/2}^{nk}$$

denklemini elde edebiliriz. Böylece 66. denklemi şu şekilde yazabiliriz:

$$X(2k)=\sum_{n=0}^{N/2-1}[x(n)+x(n+\frac{N}{2})]W_{N/2}^{nk} \quad k=0,1,...,\frac{N}{2}-1 \qquad (71)$$

Bu deklemi kelimelerle ifade etmek gerekirse :

$$X(2k)=[x(n)+x(n+\frac{N}{2})] \text{ dizisinin } \frac{N}{2} \text{ noktali DFT'si} \qquad (74)$$

Benzer yoldan 69. denklemi kullanarak 67. denklem,

$$X(2k+1)=\sum_{n=0}^{N/2-1}[x(n)-x(n+\frac{N}{2})]W_N^n W_{N/2}^{nk}$$

$$k=0,1,...,\frac{N}{2}-1 \quad (73)$$

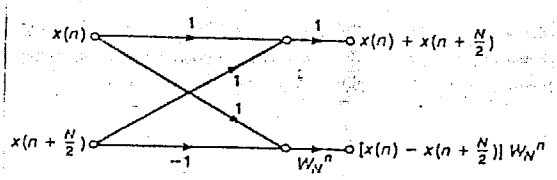
ve bu deklemi kelimelerle ifade edersek:

$$X(2k+1)=[x(n)-x(n+\frac{N}{2})]W_N^n \ /nin \ \frac{N}{2} \ noktali \ DFT'si \quad (74)$$

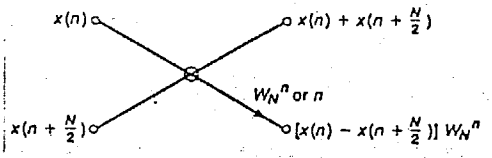
Bu denklemlerin, sinyal akiş grafiği gösterimini kullanarak hesaplamalarda neyin uygulanmasına ihtiyaç duyulacağına karar verebiliriz. 71 ve 73. denklemler için önce aşağıdaki diziler elde edilmelidir:

$$x(n)+x(n+\frac{N}{2}) \quad (75)$$

$$[x(n)-x(n+\frac{N}{2})]W_N^n \quad (76)$$

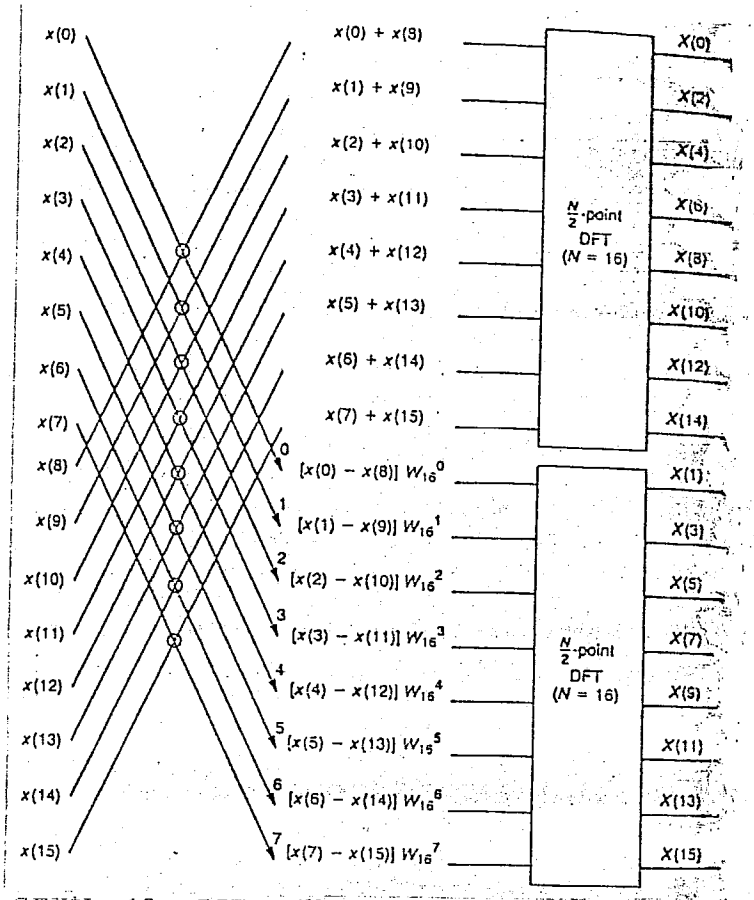


ŞEKİL 9.a : DIF butterfly için sinyal akış grafiği

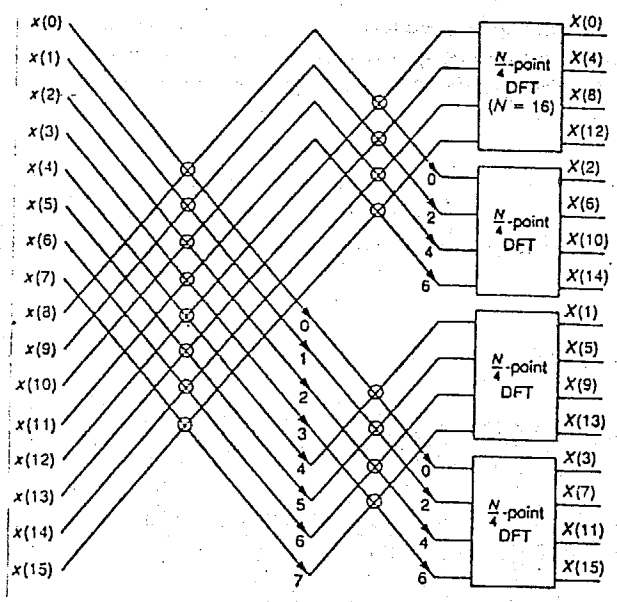


ŞEKİL 9.b : DIF butterfly için kısa gösterim

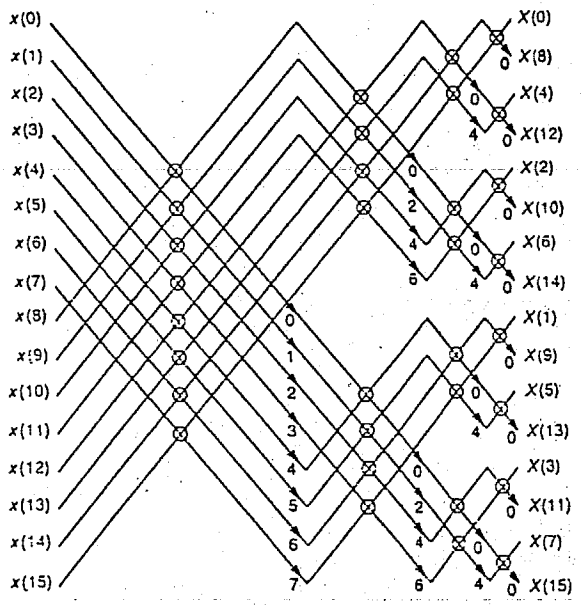
Şekil 9.a'da sinyal akiş grafiği gösterimi ve şekil 9.b'de onun kısa gösterimi verilmiştir. 16 noktalı DFT için bu gösterimin kullanılışı şekil 10'da verilmiştir. Bu işlem şekil 11'de gösterildiği gibi her 8 noktalı DFT'yi 2'şer tane 4 noktalı DFT'ye dönüştürmek için tekrar ettirilir. Bu işlemin kısa gösterim kullanılarak yapılan sinyal akiş grafiği şekil 12'de verilmiştir.



ŞEKİL 10 : DIF - ilk adım



ŞEKİL 11 : DIF için ilk iki ayrıştırma adımı



ŞEKİL 12 : 16 noktalı DFT'nin DIF metodu ile hesaplanması

Şekil 12'yi, şekil 8'deki 16 noktalı DIT çözümü ile karşılaştırırsak, tekrar, karakteristik butterfly yapısı ile karşılaşırız. DIF algoritmasında butterfly'ların $\log_2 N$ tane adımı olduğu görülebilir ve her adımda N/2 tane kompleks butterfly hesaplaması vardır. DIT butterfly'ları toplama ve çıkarmaya göre daha öncelikli olarak ağırlık faktörleri yoluyla çarpmaya ihtiyaç duymaktadır. Oysa DIF algoritmasında toplama ve çıkarmadan sonra ağırlık faktörleri yoluyla çarpma yapılmaktadır. Bununla beraber her iki algoritmada kompleks çarpmaların sayısı

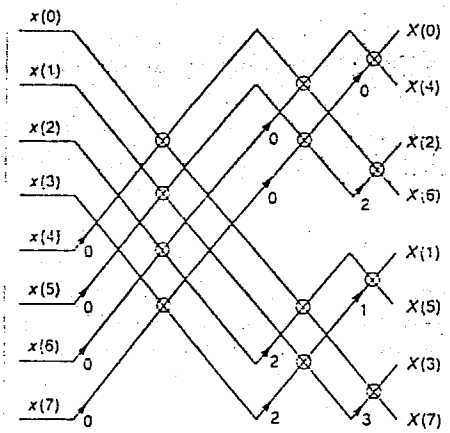
$$T_B = \frac{N}{2}\log_2 N \quad (N,\ 2\text{'nin herhangi bir kuvveti}) \qquad (77)$$

kadardır.

DIT algoritmasında 15. denklemde de görüldüğü gibi ağırlık faktörleri periyodiktir ve her periyodik örnek 0 bileşeninden başlamaktadır. Butterfly adımlarını soldan sağa 1. adımdan başlayarak tekrar numaralandırırsak ve bu işlemi $\log_2 N$ adımına kadar sürdürürsek L. adımda ağırlık faktörleri arasındaki arttırma miktarı $2^{(L-1)}$'dir. L. adımda butterfly'a iştirak eden noktalara ayrılan hafıza bolgesi $N/2^L$'dir. Burada bileşenlerin arttırılmasının ve hafıza ayrılmalarının DIF ve DIT algoritmalarında rollerini değiştikleri dikkati çekmektedir. Aslında bu iki algoritma, duality (ikilik) formunda karakterize edilmektedir. DIF için normal düzende giriş dizisi kullanılması bu sonucu kuvvetlendirir ve çıkışta alınan dizi bit-reversed düzenindedir. Oysa DIT algoritmasında bunun tersi doğrudur.

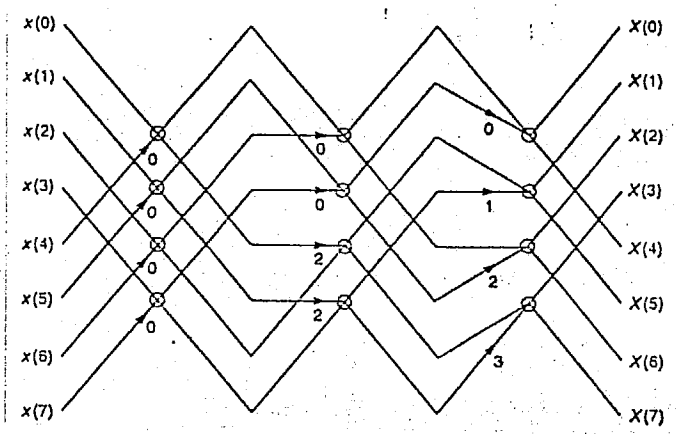
2.3 DIT ve DIF Algoritmalarında Yapılabilecek Değişiklikler:

DIT ve DIF algoritmalarının sinyal akış grafiği gösterimlerine bakılırsa daha farklı düzenlemeye sahip grafiklerin de çizilebileceği görülebilir. Çıkış noktamızda olduğu gibi şekil 6'da gösterilen 8 noktalı DFT şeklini kullanalım. Uygun giriş değerleri ağırlık faktörleriyle beraber butterfly'larda yer almaya devam ettikçe grafik gösterimi doğru giriş-çıkış bağlantılarını koruyacak şekilde çok değişik biçimlerde yeniden düzenlenebilir. Örneğin giriş datalarımızın normal sırasında ve çıkış datalarımızın bit-reversal düzeninde olmasını isteyebiliriz. Şekil 13'de bu problemin çözümü olan sinyal akış grafiği, kısa gösterimi kullanılarak çizilmiştir. Şekil 6 ve şekil 13'ü karşılaştırırsak, iki data değeri bir butterfly hesaplamasında giriş değerleri olarak kullanılırsa çıkışta elde edilen değerin giriş değerlerini saklayan bölgeye konabileceğini görebiliriz. Çünkü daha sonraki hesaplamalarda da daha büyük bir bölgeye ihtiyaç duyulmayacaktır. Bu özelliğinden dolayı şekil 6 ve şekil 13'de gösterilen FFT'ler "inplace" algoritmaları olarak adlandırılır.



ŞEKİL 13 : 8 noktalı DIT FFT. Girişler normal sırasında, çıkışlar ise bit-reversed düzenindedir.

Diğer bir düzenleme ise giriş ve çıkış datalarını normal sıralarına koymaktır. Şekil 14'de bunun nasıl yapılabileceği gösterilmiştir. Bu işlem inplace algoritmalarından daha uzun değildir. Çünkü birinci adımdan sonra hangi datanın butterfly hesaplarında yer aldığını gösteren giriş değerini saklayan bir bölgeyi her zaman kullanmıyoruz. Bu nedenle her basamakta geçici saklama bölgeleri gerekmektedir.



ŞEKİL 14 : 8 noktalı DIT FFT. Giriş ve çıkışlar normal sırasında.

Diğer birçok düzenlemeler mümkündür. Bazı alternatifler özel hardware yetenekleri kullanarak yapılabilir. Örneğin bir anda birden fazla butterfly'ın paralel işlem görmesi FFT hesaplarının hızını arttıracaktır.

3.0 FFT PROGRAMININ GELİŞTİRİLMESİ

3.1 AKIŞ ŞEMASI

FFT programı akış şeması şekil 15'de gösterilmiştir. Program 4 ana bölümden oluşur:

1. Datanın alınması
2. Ön işlemler
3. FFT işlemleri
4. Son işlemler

Akış şeması, hem C programına hem de ASM96 programına uyarlanabilir.

1. Bölümde, FFT işleminin uygulanacağı örneklenmiş değerler programa verilir.

2. Bölümde, FFT hesaplarında kullanılacak olan bazı değerler hesaplanır. C programı için 'Weighting Factor' dizilerinin oluşturulması, ASM96 programı için FFT_OUT dizisinin temizlenmesi ve Stack Pointer'a değer atanması bu bölümde gerçekleştirilir.

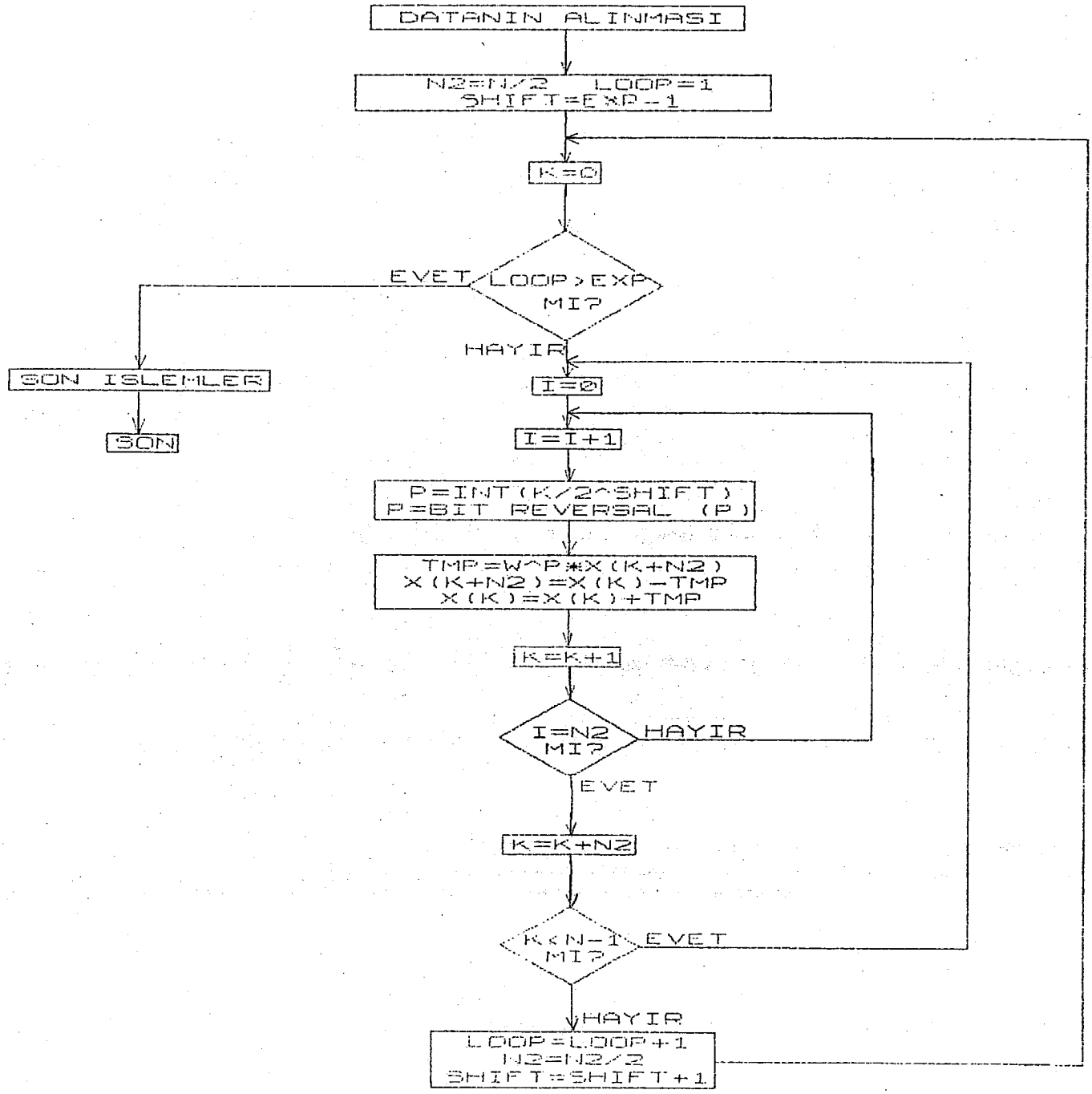
3. Bölümde, içiçe üç döngü kullanılarak FFT hesapları yapılır.

N ile gösterilen değer giriş dizisinin uzunluğudur. N2 ise N'in yarısıdır. $EXP=\log_2 N$'dir ve SHIFT=EXP-1 alınarak en dıştaki döngüye girilir. LOOP değişkeni en dıştaki döngünün sayacıdır. LOOP=1 ile başlanır ve LOOP>1 oluncaya kadar döngü devam eder.

Ortadaki döngünün değişkeni K'dır ve K<N-1 olduğu sürece boyunca bu döngü devam eder.

En içteki döngünün değişkeni ise I'dır. I, 0'dan itibaren artmaya başlayarak N2'ye eşit oluncaya kadar bu döngü devam eder.

4. Bölümde, çıkış dizisinde elde edilen değerlerin büyüklükleri hesaplanır.



ŞEKİL 15: FFT PROGRAMI AKIŞ ŞEMASI

3.2 C PROGRAMI HAKKINDA GENEL AÇIKLAMA:

Ek 1'de C dilinde yazılmış olan FFT programı verilmiştir.

Programın başlangıcında, FFT algoritması için gerekli olan dizilere ve değişkenlere başlangıç değerleri atanmıştır.

İlk alt program olan 'initialize' da Reel ve İmajiner 'Weighting Factor' dizileri oluşturulur.

İkinci alt program olan 'fft_algorithm' de ise FFT işlemleri gerçekleştirilir.

'Post_Processing isimli alt programın görevi 'fft_algorithm' de bulunan reel ve imajiner dizilerin büyüklüğünü hesaplamaktır.

Dördüncü alt programdan yedinci alt programa kadar olan kısımda programda elde edilen çıkış değerleri grafik ekranda gösterilir.

C programının en büyük özelliği ise kullanılan algoritmanın ASM96 programında kullanılan algoritma ile aynı olmasıdır. Bu nedenle ASM96 programında komutların yanında açıklama olarak C programındaki karşılıkları verilmiştir.

3.3 ASM96 PROGRAMI HAKKINDA GENEL AÇIKLAMA:

Ek 2'de 80C196KC Microcontroller'ın makine dilinde yazılmış olan FFT programı verilmiştir.

Programın başlangıç kısmında, FFT algoritması için gerekli olan registerlerin ve dizilerin tanımlaması yapılmıştır.

Programda yer alan ilk rutin MAINP ismindeki ana programdır. Programın anlaşılabilirliğini arttırmak amacıyla temel işlemleri yapan program parçalarının, bir ana programdan çağırılması şeklinde bir yöntem takip edilmiştir.

Ardından gelen alt program FFT_OUT ismindeki çıkış dizisini temizler.

Üçüncü alt program FFT hesaplarının yapıldığı kısımdır.

Ardından gelen iki alt programda çıkış dizisinin büyüklüğü hesaplanır.

Beşinci ve altıncı alt programlarda ise isteğe bağlı olarak çıkış dizisinin büyüklüğünün logaritması veya karekök değeri hesaplanır.

Yedinci alt program ise dışarıdan alınacak analog sinyalleri digitale çevirir.

Programın sonunda FFT hesaplarında kullanılan tablolar verilmiştir.

4.0 80C196KC MICROCONTROLLER'IN TANITIMI VE KOMUT LİSTESİ

Gelişmiş kontrol uygulamalarında sık sık çok hızlı digital sinyallerle karşılaşılır. Ayrıca sık sık 16 bit ve 32 bitte yüksek hızda işlem yapmak istenir. 8096 ile başlayan MCS-96 microcontroller serisi yüksek hızda hesaplamalara ve I/O işlemlerine ihtiyaç duyulduğu uygulamalarda kullanılır.

8096, I/O sistemleri için geliştirilmiş çarpma ve bölme komutlarına sahip 16 bitlik aritmetik komutları olan bir microcontroller'dır.

Aşağıda 8096 ailesinin komut seti verilmiştir:

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
ADD/ADDB	2	D ← D + A	↕	↕	↕	↕	↑	—	
ADD/ADDB	3	D ← B + A	↕	↕	↕	↕	↑	—	
ADDC/ADDCB	2	D ← D + A + C	↓	↕	↕	↕	↑	—	
SUB/SUBB	2	D ← D − A	↕	↕	↕	↕	↑	—	
SUB/SUBB	3	D ← B − A	↕	↕	↕	↕	↑	—	
SUBC/SUBCB	2	D ← D − A + C − 1	↓	↕	↕	↕	↑	—	
CMP/CMPB	2	D − A	↕	↕	↕	↕	↑	—	
MUL/MULU	2	D, D + 2 ← D * A	—	—	—	—	—	?	2
MUL/MULU	3	D, D + 2 ← B * A	—	—	—	—	—	?	2
MULB/MULUB	2	D, D + 1 ← D * A	—	—	—	—	—	?	3
MULB/MULUB	3	D, D + 1 ← B * A	—	—	—	—	—	?	3
DIVU	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	↕	↑	—	2
DIVUB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	↕	↑	—	3
DIV	2	D ← (D, D + 2)/A, D + 2 ← remainder	—	—	—	?	↑	—	2
DIVB	2	D ← (D, D + 1)/A, D + 1 ← remainder	—	—	—	?	↑	—	3
AND/ANDB	2	D ← D and A	↕	↕	0	0	—	—	
AND/ANDB	3	D ← B and A	↕	↕	0	0	—	—	
OR/ORB	2	D ← D or A	↕	↕	0	0	—	—	
XOR/XORB	2	D ← D (excl. or) A	↕	↕	0	0	—	—	
LD/LDB	2	D ← A	—	—	—	—	—	—	
ST/STB	2	A ← D	—	—	—	—	—	—	
LDBSE	2	D ← A; D + 1 ← SIGN(A)	—	—	—	—	—	—	3, 4
LDBZE	2	D ← A; D + 1 ← 0	—	—	—	—	—	—	3, 4
PUSH	1	SP ← SP − 2; (SP) ← A	—	—	—	—	—	—	
POP	1	A ← (SP); SP ← SP + 2	—	—	—	—	—	—	
PUSHF	0	SP ← SP − 2; (SP) ← PSW; PSW ← 0000H I ← 0	0	0	0	0	0	0	
POPF	0	PSW ← (SP); SP ← SP + 2; I ← ↕	↕	↕	↕	↕	↕	↕	
SJMP	1	PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LJMP	1	PC ← PC + 16-bit offset	—	—	—	—	—	—	5
BR (indirect)	1	PC ← (A)	—	—	—	—	—	—	
SCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 11-bit offset	—	—	—	—	—	—	5
LCALL	1	SP ← SP − 2; (SP) ← PC; PC ← PC + 16-bit offset	—	—	—	—	—	—	5
RET	0	PC ← (SP); SP ← SP + 2	—	—	—	—	—	—	
J (conditional)	1	PC ← PC + 8-bit offset (if taken)	—	—	—	—	—	—	5
JC	1	Jump if C = 1	—	—	—	—	—	—	5
JNC	1	Jump if C = 0	—	—	—	—	—	—	5
JE	1	Jump if Z = 1	—	—	—	—	—	—	5

Figure 2-5. Instruction Summary

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
2. D, D + 2 are consecutive WORDS in memory; D is DOUBLE-WORD aligned.
3. D, D + 1 are consecutive BYTES in memory; D is WORD aligned.
4. Changes a byte to a word.
5. Offset is a 2's complement number.

Mnemonic	Operands	Operation (Note 1)	Flags						Notes
			Z	N	C	V	VT	ST	
JNE	1	Jump if Z = 0	—	—	—	—	—	—	5
JGE	1	Jump if N = 0	—	—	—	—	—	—	5
JLT	1	Jump if N = 1	—	—	—	—	—	—	5
JGT	1	Jump if N = 0 and Z = 0	—	—	—	—	—	—	5
JLE	1	Jump if N = 1 or Z = 1	—	—	—	—	—	—	5
JH	1	Jump if C = 1 and Z = 0	—	—	—	—	—	—	5
JNH	1	Jump if C = 0 or Z = 1	—	—	—	—	—	—	5
JV	1	Jump if V = 1	—	—	—	—	—	—	5
JNV	1	Jump if V = 0	—	—	—	—	—	—	5
JVT	1	Jump if VT = 1; Clear VT	—	—	—	—	0	—	5
JNVT	1	Jump if VT = 0; Clear VT	—	—	—	—	0	—	5
JST	1	Jump if ST = 1	—	—	—	—	—	—	5
JNST	1	Jump if ST = 0	—	—	—	—	—	—	5
JBS	3	Jump if Specified Bit = 1	—	—	—	—	—	—	5, 6
JBC	3	Jump if Specified Bit = 0	—	—	—	—	—	—	5, 6
DJNZ	1	D ← D − 1; if D ≠ 0 then PC ← PC + 8-bit offset	—	—	—	—	—	—	5
DEC/DECB	1	D ← D − 1	↕	↕	↕	↕	↑	—	
NEG/NEGB	1	D ← 0 − D	↕	↕	↕	↕	↑	—	
INC/INCB	1	D ← D + 1	↕	↕	↕	↕	↑	—	
EXT	1	D ← D; D + 2 ← Sign (D)	↕	↕	0	0	—	—	2
EXTB	1	D ← D; D + 1 ← Sign (D)	↕	↕	0	0	—	—	3
NOT/NOTB	1	D ← Logical Not (D)	↕	↕	0	0	—	—	
CLR/CLRB	1	D ← 0	1	0	0	0	—	—	
SHL/SHLB/SHLL	2	C ← msb ——— lsb ← 0	↕	?	↕	↕	↑	—	7
SHR/SHRB/SHRL	2	0 → msb ——— lsb → C	↕	?	↕	0	—	↕	7
SHRA/SHRAB/SHRAL	2	msb → msb ——— lsb → C	↕	↕	↕	0	—	↕	7
SETC	0	C ← 1	—	—	1	—	—	—	
CLRC	0	C ← 0	—	—	0	—	—	—	
CLRVT	0	VT ← 0	—	—	—	—	0	—	
RST	0	PC ← 2080H	0	0	0	0	0	0	8
DI	0	Disable All Interrupts (I ← 0)	—	—	—	—	—	—	
EI	0	Enable All Interrupts (I ← 1)	—	—	—	—	—	—	
NOP	0	PC ← PC + 1	—	—	—	—	—	—	
SKIP	0	PC ← PC + 2	—	—	—	—	—	—	
NORML	2	Left Shift Till msb = 1; D ← shift count	↕	?	0	—	—	—	7
TRAP	0	SP ← SP − 2; (SP) ← PC PC ← (2010H)	—	—	—	—	—	—	9

Figure 2-5. Instruction Summary (Continued)

NOTES:
1. If the mnemonic ends in "B", a byte operation is performed, otherwise a word operation is done. Operands D, B, and A must conform to the alignment rules for the required operand type. D and B are locations in the register file; A can be located anywhere in memory.
5. Offset is a 2's complement number.
6. Specified bit is one of the 2048 bits in the register file.
7. The "L" (Long) suffix indicates double-word operation.
8. Initiates a Reset by pulling RESET low. Software should re-initialize all the necessary registers with code starting at 2080H.
9. The assembler will not accept this mnemonic.

5.0 EV80C196KC MICROCONROLLER EVALUATION BOARD

EV80C196KC Microcontroller Evaluation Board, 80C196KC için yazılmış olan programları ROM kullanmaya ihtiyaç kalmadan çalıştırabilmek üzere Intel firması tarafından geliştirilmiştir. Evaluation Board, IBM PC uyumlu bir bilgisayara seri port üzerinden bağlanır. Programlar, bilgisayarda herhangibir editörde yazıldıktan sonra, ASM96 Macro Assembler programı kullanılarak object file elde edilir. ECM96 programı aracılığı ile bu file, EV80C196KC üzerindeki ROM-SIM'lere (ROM SIMulator) aktarılır. ROM-SIM entegreleri microcontroller'ın ROM'u gibi çalışır. ROM-SIM'lere aktarılan programlar PC bilgisayar aracılığı ile istenildiği zaman çalıştırılabilir. Programların istenilen kısımları çalıştırılabildiği gibi, ECM96 programı kullanılarak üzerlerinde istenilen değişiklikler yapılabilir. Registerlerin içerikleri öğrenilebilir. Microcontroller'ın kullanılabildiği hafıza bölgelerinde istenilen bir yere byte, word, long tipinde değerler yazılabilir ve okunabilir.

EK 1

C PROGRAMI

```
/*****************************************************************************
 *                                                                           *
 * FILE              : FFT.C                                                 *
 *                                                                           *
 * STARTED           : May 1, 1992                                           *
 *                                                                           *
 * LAST EDIT         : May 23, 1992                                          *
 *                                                                           *
 * VERSION           : 0.5                                                   *
 *                                                                           *
 * WRITTEN BY        : Sururi Karacorlu                                      *
 *                                                                           *
 * DESCRIPTION       : Fast Fourier Transform of                             *
 *                       a sampled signal                                    *
 *                                                                           *
 * FUNCTIONS         :                                                       *
 * 1. initialize     : Initializing the weighting factors                    *
 * 2. fft_algorithm  : Realising the FFT algorithm                           *
 * 3. post_processing : Post processing and reordering                       *
 * 4. graph_init     : Initializing the graphics hardware                    *
 * 5. draw_3axes     : Drawing axes                                          *
 * 6. plot_sequences1 : Plotting reel ve imaginary input sequences           *
 * 7. plot_sequences2 : Plotting magnitude of output                         *
 * 8. main           : The main program                                      *
 *****************************************************************************/

#include <stdio.h>
#include <math.h>
#include <graphics.h>

  /* Reel parts of the input sequence */
  float  xr[32] = { 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,-2,-2,-2,-2,-2,-2,
                   -2,-2,-2,-2,-2,-2,-2,-2,-2,-2
                  };
  /* Imaginary parts of the input sequence */
  float  xi[32] = { 2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,-2,-2,-2,-2,-2,-2,
                   -2,-2,-2,-2,-2,-2,-2,-2,-2,-2
                  };
  /* Reel and imaginary parts of the weighting factors */
  float  wr[32], wi[32];

  /* Magnitute of result */
  float  magn[32];

  /* Bit reversal sequence */
  int      br[32] = { 0,16,8,24,4,20,12,28,2,18,10,26,6,22,14,30,1,17,
                      9,25,5,21,13,29,3,19,11,27,7,23,15,31
                     };
  int      n = 32;  /* number of points  */
  int      n1 = 31; /* n1= n-1           */
  float    pi=3.141592654;
  float    tpn;
  float    pin;
```

```
    int   GraphDriver;   /* The Graphics device driver          */
    int   GraphMode;     /* The Graphics mode value             */
    int   MaxX, MaxY;    /* The maximum resolution of the screen */
    int   ErrorCode;     /* Reports any graphics errors         */


/*=========================================================================*/
void initialize(void)
{ /* Initializes the weighting factor sequences    Ver 0.0  */

   float p, pn;

   tpn = (2*pi) / n;    /* tpn = 2*pi/n */
   pin = pi/n;          /* pin = pi/n   */

   for(p=0;  p<32;  ++p)
   {  pn = p*tpn;
      wr[p] = cos(pn);      /* Real parts of weighting factors */
      wi[p] = -sin(pn);     /* Imaginary parts of weighting factors */
   }
}


/*=========================================================================*/
void fft_algorithm()
{ /* Realises the Fast Fourier Transform    Ver 0.0   */

  int   loop,n2=n/2;
  int   exponent=5;      /* n = 2^exponent */
  int   shift=exponent-1;
  for(loop=1; loop<=exponent;  ++loop)
  {
   int  k=0,incnt,p,x,kn2;
   float wrp,wip,tmpr,tmpi,tmpr1,tmpi1;
   do
   {
    incnt=0;
    do
    {
     ++incnt;
     x=k/pow(2,shift);
     p=br[x];
     wrp=wr[p];
     wip=wi[p];
     kn2=k+n2;
     tmpr=(wrp*xr[kn2] - wip*xi[kn2])/2;
     tmpi=(wrp*xi[kn2] + wip*xr[kn2])/2;
     tmpr1=xr[k]/2;
     tmpi1=xi[k]/2;
     xr[k+n2]=tmpr1-tmpr;
     xi[k+n2]=tmpi1-tmpi;
     xr[k]=tmpr1+tmpr;
     xi[k]=tmpi1+tmpi;
     ++k;
    }
    while (incnt<n2);
```

```
            k=k+n2;
            }
            while (k<n1);
           n2=n2/2;--shift;
           }
          }


     /*============================================================================*/
       void post_processing()
     /* Computes the magnitute of result
                                and print results on the screen    Ver 0.0 */
     {
       int k;
       float kpin,xrbrk,xibrk,xrbrnk,xibrnk,ti,tr,xrt,xit,
            outr,outi,magsq,mag,dbfact,decibel,m=16383;
     /*  printf("        k    outr     outi     mag    decibel");
       printf("  *outr  *outi  *mag \n");
       printf("        -    ----     ----     ---    -------");
       printf(" ------  ------  ----- \n");                    */
       for(k=0;  k<32;  ++k)
       {
        kpin=k*pin;
        xrbrk=xr[br[k]];
        xibrk=xi[br[k]];
        xrbrnk=xr[br[n-k]];
        xibrnk=xi[br[n-k]];
        ti=(xibrk+xibrnk)/2;
        tr=(xrbrk-xrbrnk)/2;
        xrt=(xrbrk+xrbrnk)/4;
        xit=(xibrk-xibrnk)/4;
        outr= xrt + ti*cos(kpin)/2 - tr*sin(kpin)/2;
        outi= xit - ti*sin(kpin)/2 - tr*cos(kpin)/2;
        magsq=(float)(pow(outr,2)+pow(outi,2));
        mag=(float)sqrt(magsq);
        magn[k]=mag;
        if ( magsq*m < .5 )  decibel=0;
        else
        {
        dbfact=pow(m,2)/65535;
        decibel=10*log(magsq*dbfact);
        decibel=decibel * 0.43429448l;
       }
     /*  printf(" %2d %8.5f %8.5f %8.5f %8.3f %7.0f %7.0f %6.0f \n ",
                  k,outr,outi,mag,decibel,*outr,*outi,*mag);   */
     }
```

```
/*=========================================================================*/
  void graph_init()
/* Initializes the graphics hardware     Ver 0.0    */
{
   GraphDriver = DETECT;          /* Request auto-detection        */
   initgraph(&GraphDriver, &GraphMode, "");
   ErrorCode = graphresult();     /* Read result of initialization */

   if( ErrorCode != grOk )        /* Error occured during init     */
   {
   printf("Graphics System Error: %s\n", grapherrormsg(ErrorCode));
   exit(1);
   }
   MaxX = getmaxx();
   MaxY = getmaxy();              /* Read size of screen           */
}

/*=========================================================================*/
void draw_3axes(x1, y1, x2, y2,x3,y3)

int x1, y1, x2, y2, x3, y3;          /* origins of axes */
{
   setbkcolor(BLUE); textcolor(BLACK);

   moveto(x1, y1); lineto(x1, 20);      /* vertical axis */
   moveto(x1, y1); lineto(x2-10, y1);   /* horizontal axis */
   outtextxy(x1, 20, "Real Input Sequence XR(n)");
   outtextxy(x1-10,y1-xr[0]*40, "2");

   moveto(x2, y2); lineto(x2, 20);
   moveto(x2, y2); lineto(MaxX-10, y2);
   outtextxy(x2, 20, "Imaginary Input Sequence XI(n)");
   outtextxy(x2-10,y2-xi[0]*40; "2");

   moveto(x3, y3); lineto(x3, MaxY/2-10);
   moveto(x3, y3); lineto(MaxX-10, y3);
   outtextxy(x3, MaxY/2-10, "Magnitude of");
   outtextxy(x3+50,MaxY/2-1, "Result");
}


/*=========================================================================*/
void plot_sequences1(x1, y1, x2, y2)

 int x1, y1, x2, y2;
{
  int i;
  float adjust = 0;

  setcolor(YELLOW);
  for (i=0;  i<n;  i++)
  {   moveto(x1 + adjust, y1); lineto(x1 + adjust , y1 - 40*xr[i]);
      circle(x1 + adjust , y1 - 40*xr[i], 2);
      adjust += MaxX /(2*n);
  }
```

```
        adjust = 0;
        for (i=0;  i<n;  i++)
        {   moveto(x2+adjust, y2); lineto(x2+adjust, y2 - 40*xi[i]);
            circle(x2 + adjust, y2 - 40*xi[i], 2);
            adjust += MaxX / (2*n);
        }
    }



    /*=========================================================================*/
    void plot_sequences2(x3, y3)
    /* Plots magnitude result          Ver 0.0  */

      int x3, y3;
    {
      int i;
      float adjust = 0;

      for (i=0;  i<n;  i++)
      {   moveto(x3+adjust , y3);  lineto(x3+adjust , y3 - 100*magn[i]);
          circle(x3 + adjust ,y3 - 100*magn[i], 2);
          adjust += MaxX / (2*n);
      }
      outtextxy(x3+2,y3-magn[1]*100-10, "1.27");
    }

    /*=========================================================================*/
    main()
    /*  The main of the program         Ver 0.0  */
    {
      int x1, y1, x2, y2, x3, y3;    /* origins of 3 x-y axes */

      graph_init();

      /* calculate the origins of the 3 x-y axes */
      x1 = 10; y1 = MaxY/4 + 40;
      x2 = MaxX/2; y2 = MaxY/4 + 40;
      x3 = 10; y3 = MaxY - 30;

      draw_3axes(x1, y1, x2, y2, x3, y3); /* draw 3 axes */

      plot_sequences1(x1,y1,x2,y2);

      initialize();
      fft_algorithm();
      post_processing();
      plot_sequences2(x3, y3);

      getch();
      closegraph();   /* return the system to text mode */
      return 0;

    }
```

EK 2

ASM96 PROGRAMI

```
DOS 4.0 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: C:\SURURI\FFT.A96
OBJECT FILE: C:\SURURI\FFT.OBJ
CONTROLS SPECIFIED IN INVOCATION COMMAND: DB

ERR LOC  OBJECT              LINE     SOURCE STATEMENT
                               1   ;*******************************************************************************
                               2   ;*******************************************************************************
                               3   ;*                                                                             *
                               4   ;*                                F F T                                        *
                               5   ;*                                                                             *
                               6   ;*                    FAST        FOURIER        TRANSFORM                     *
                               7   ;*                                                                             *
                               8   ;*                                                                             *
                               9   ;* File Name & Number : FFT.A96  &  1/1                                        *
                              10   ;*                                                                             *
                              11   ;* Started & Revised  : 26-Nisan-1992 &  29-Mayis-1992  Rv1.04                  *
                              12   ;*                                                                             *
                              13   ;* SUBROUTINES        :                                                        *
                              14   ;*                                                                             *
                              15   ;*   1. MAINP         : Ana program                                            *
                              16   ;*   2. CLR_FFT_OUT   : FFT_OUT adresindeki dizinin temizlenmesi               *
                              17   ;*   3. FFT_CALC      : FFT hesaplarinin yapildigi alt program                 *
                              18   ;*   4. UNWEAVE       : Son islemler ve yeniden siralama                       *
                              19   ;*   5. CALC_LOG      : 10*LOG(Magnitude^2) degerinin hesaplanmasi             *
                              20   ;*   6. CALC_SQRT     : Karekok degerinin hesaplanmasi                         *
                              21   ;*                                                                             *
                              22   ;*                                                                             *
                              23   ;* DESCRIPTION        :                                                       *
                              24   ;*                                                                             *
                              25   ;*    Uludag Universitesi Muhendislik Fakultesi                                *
                              26   ;*    Elektronik Muhendisligi Bolumu, 26 Nisan 1992                            *
                              27   ;*    Hazirlayan: Sururi Karacorlu                                             *
                              28   ;*    Intel Corporation MCO Applications notlarindan yararlanilmistir.         *
                              29   ;*                                                                             *
                              30   ;*    Bu program 2N-nokta algoritmasini kullanarak 64 reel nokta uzerinde      *
                              31   ;* Fast Fourier Transform (FFT) islemini gerceklestirmektedir. Bu algorit-    *
                              32   ;* mada 32 reel ve 32 imajiner nokta uzerinde standart FFT islemi uygula-     *
                              33   ;* nir. Reel ve imajiner diziler reel data degerleriyle doldurulmustur ve     *
                              34   ;* FFT'nin cikisi post-processor'e uygulanarak sonuc degerleri elde edilir.   *
                              35   ;* Sonuc degerleri 32 noktali bir diziye yerlestirilir. Post-processing      *
                              36   ;* bolumunde 32 bitlik isaretsiz sayilarin karekoklerini almak icin table    *
                              37   ;* lookup algoritmasi (tablo bakma metodu) kullanilmistir.                    *
                              38   ;*                                                                             *
                              39   ;*    FFT programindaki butun hesaplar 16 bit isaretli tamsayilar uzerinden    *
                              40   ;* yapilir. Bu nedenle herhangi bir frekens bileseninin alabilecegi maximum   *
                              41   ;* deger +/- 32K olur. ( +/- 32K 'lik bir kare dalganin temel bileseninin     *
                              42   ;* +/- 40K 'dan buyuk olacagina dikkat ediniz.)  Matematiksel islemlerin      *
                              43   ;* hizini arttirmak icin mumkun olan heryerde tablolar kullanilmistir. Her    *
                              44   ;* frekans bileseninin mutlak buyuklugunu elde etmeyi amaclayan transform     *
                              45   ;* islemi ic degiskenlerle yapildigi zaman 12 milisaniye, dis degiskenlerle   *
                              46   ;* yapildigi zaman 14 milisaniye surmektedir.                                 *
                              47   ;*                                                                             *
                              48   ;*    Program iki tane 32'lik diziye yerlestirilmis ornek degerleri kullan-    *
                              49   ;* maktadir. Bu diziler XREAL ve XIMAG adreslerinden baslamaktadir. Sonucta   *
                              50   ;* elde edilen degerler FFT_OUT ismindeki 32 bitlik diziye yerlestirilir.     *
                              51   ;* bu diziler tamamen dis degiskenler olarak tanimlanmistir. Programda,
```

```
ERR LOC  OBJECT              LINE      SOURCE STATEMENT
                               52  ;* cikis degerlerinin ortalamasi alinacaginda FFT_CALC rutini calistirilma- *
                               53  ;* dan once FFT_OUT dizisi temizlenmelidir.                                *
                               54  ;*                                                                        *
                               55  ;*    Yapilanlari test etmeyi kolaylastirmak icin programin orjinali C'de  *
                               56  ;* yazilmistir. Algoritmayi izlemeyi kolaylastirmak icin, programda, komut- *
                               57  ;* larla beraber C'deki karsilik gelen satirlar da verilmistir.           *
                               58  ;*                                                                        *
                               59  ;**************************************************************************
                               60  ;**************************************************************************
                               61
                               62  ;========================================================================
                               63  ;                          REGISTERLERIN TANIMLANMASI
                               64  ;========================================================================
    0040                       65  RSEG AT 40H
    0040                       66  zero:          dsw   1          ; Sifir registeri
    0042                       67  sp:            dsw   1          ; Stack pointer
    0044                       68  error:         dsb   1          ; Hata olusursa nedenini arastirabilmek icin
    0045                       69  fft_mode:      dsb   1          ; FFT girisi icin mod secimi
    0046                       70  scale_factor:  dsb   1          ; FFT cikisindaki saga kaydirmalarin sayisi
    0048                       71  tmpr:          dsl   1          ; Temporary register, Real
    004C                       72  tmpi:          dsl   1          ; Temporary register, Imaginary
    0050                       73  tmpr1:         dsl   1          ; Temporary register, Real
    0054                       74  tmpi1:         dsl   1          ; Temporary register, Imaginary
    0058                       75  xrtmp:         dsl   1          ; Temporary register, Real
    005C                       76  xitmp:         dsl   1          ; Temporary register, Imaginary
    0060                       77  xrrk:          dsl   1          ; Temporary register, Real
    0064                       78  xrnk:          dsl   1          ; Temporary register, Real
    0068                       79  xirk:          dsl   1          ; Temporary register, Imaginary
    006C                       80  xink:          dsl   1          ; Temporary register, Imaginary
    0070                       81  diff:          dsl   1          ; Karekok icin Tablo Araligi
    0074                       82  sqrt:          dsl   1          ; Karekok
    0078                       83  log:           dsl   1          ; 10 Log(Magnitude^2)
    007C                       84  nxtloc:        dsl   1          ; Tablodaki bir sonraki adres
    0080                       85  wrp:           dsw   1          ; Agirlik Faktoru, reel
    0082                       86  wip:           dsw   1          ; Agirlik Faktoru, imajiner
    0084                       87  pwr:           dsw   1          ; Bit reversal degerlerinin saklandigi reg.
    0086                       88  in_cnt:        dsw   1          ; Increment Counter
    0088                       89  ndiv2:         dsw   1          ; N/2   (0<n<N)*2
    008A                       90  kptr:          dsw   1          ; Sayici. Dizilerdeki elemanlara erismek
    008C                       91  kn2:           dsw   1          ; KPTR + NDIV2
    008E                       92  n_sub_k:       dsw   1          ; Dizideki elmanlara erismek icin (N-K)*2
    0090                       93  rk:            dsw   1          ; KPTR'nin bit reversal pointer'i
    0092                       94  rnk:           dsw   1          ; N_SUB_K'nin bit reversal pointer'i
    0094                       95  shift_cnt:     dsb   1          ; Shift Counter
    0095                       96  loop_cnt:      dsb   1          ; Loop Counter
    0096                       97  ptr:           dsw   1          ; Karekok tablosu icin pointer
    0098                       98  xrtmp1:        dsw   1          ; Temporary register, Real
    009A                       99  xitmp1:        dsw   1          ; Temporary register, Imaginary
    009C                      100  offset:        dsw   1          ; FFT_OUT dizisinin silinmesinde kullanilan
                              101                                  ;  pointer
    009E                      102  ax:            dsw   1          ; gecici register
    00A0                      103  bx:            dsw   1          ; gecici register
    00A2                      104  cx:            dsw   1          ; gecici register
    00A4                      105  dx:            dsw   1          ; gecici register
    00A6                      106  sample_period:dsw   1          ; A/D Cevrim icin ornekleme zamani
    00A8                      107  adudtemp0:     dsw   1          ; A/D Cevrim rutini icin gecici register
    00AA                      108  aductemp0:     dsw   1          ; A/D Cevrim rutini icin gecici register
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
    00AC                  109   aductemp1:    dsw   1          ; A/D Cevrim rutini icin gecici register
    00AE                  110   top_of_buffer:dsw   1          ; ADC rutini icin buffer'in uzunlugu
    00B0                  111   sample_count: dsb   1          ; ADC rutini icin ornek sayisi
    00B1                  112   control_a2d:  dsb   1          ; A/D cevrim islemlerini kontrol eden reg.
    00B2                  113   src_ptr:      dsw   1          ; ADC rutini icin secilen bufferin adresi
      00B2                114   temp set src_ptr: word         ; ADC rutini icin gecici buffer adresi
    00B4                  115   dest_ptr:     dsw   1          ; ADC rutini icin hedef bufferin adresi
    00B6                  116   loop_c:       dsw   1          ; ADC rutini icin dongu sayaci
                          117
    07D2                  118   CSEG at 2002
    07D2 A023             119     dcw    A2D_DONE_VECTOR
                          120
                          121   load_hso_command MACRO var     ; ADC rutininde HSO icin macro tanimlamasi
                          122     ldb    hso_command,#var
                          123     ld     hso_time,aductemp0
                          124   endm
                          125
    3000                  126   DSEG at 3000H
    3000                  127    XREAL:                        ; Reel ve imajiner giris degerleri icin
      3040                128    XIMAG      equ   XREAL+64     ;32'ser wordluk iki dizi
                          129
    3080                  130   DSEG at 3080H
    3080                  131    FFT_OUT:                      ; FFT_OUT: Buyukluk (magnitude) bilgisini
                          132                                  ;iceren 32 word'luk dizinin baslangic adr.
      30C0                133    OUTR       equ   FFT_OUT + 64 ; FFT'nin reel cikis degerlerini iceren
                          134                                  ;32 word'luk dizinin baslangic adresi
      3100                135    OUTI       equ   FFT_OUT + 128; FFT'nin imajiner cikis degerlerini iceren
                          136                                  ;32 word'luk dizinin baslangic adresi
    0100                  137   DSEG at 100h
    0100                  138     dest_buff_base:              ; ADC rutini icin hedef bufferin adresi
                          139
    3000                  140   DSEG at 3000H
    3000                  141     buff0_base:                  ; ADC rutini icin 0 ve 1 numarali bufferlar
      3040                142     buff1_base     equ     buff0_base+64
                          143
    2080                  144   CSEG at 2080H
                          145   ;=============================================================================
                          146   ;Sub# & Started & Revised : 1 & 22-5-1992  &  22-5-1992  &   Rv0.01
                          147   ;Assumes : -
                          148   ;Returns : -
                          149   ;Regs Modified : sp, fft_mode
                          150   ;Call(s)       : 1. CLR_FFT_OUT : FFT_OUT adresindeki dizinin temizlenmesi
                          151   ;                 2. FFT_CALC    : FFT hesaplarinin yapildigi alt program
                          152   ;                 3. UNWEAVE     : Son islemler ve yeniden siralama
                          153   ;                 4. LOAD_DATA   : A/D cevrim rutini
                          154   ;                 5. TABLE_LOAD  : Giris degerleri tablodan verilecekse
                          155   ;Purpose        : -
                          156   ;=============================================================================
    2080                  157   MAINP:
    2080 A301005042       158     ld     sp,5000h
    2085 B10045           159     ldb    fft_mode,#0000b
                          160     ;                | |
                          161     ;                | |==========> =0 ise veriler tablodan okunacak
                          162     ;                |            =1 ise veriler disaridan alinacak
                          163     ;                |============> =0 ise sonuc buyuklugu karekok olarak
                          164     ;                             =1 ise sonuc buyuklugu magnitude olarak
    2088 304504           165     bbc    fft_mode,0,DO_TAB
```

```
ERR LOC  OBJECT           LINE      SOURCE STATEMENT
    208B 2A52              166     call   LOAD_DATA
    208D 2002              167     br     C_F_D
    208F                   168   DO_TAB:
    208F 2808              169     call   TABLE_LOAD
    2091                   170   C_F_D:
    2091 2827              171     call   CLR_FFT_OUT
    2093 2839              172     call   FFT_CALC
    2095 28F3              173     call   UNWEAVE
    2097 2347              174     br     endp
                           175
                           176   ;=============================================================================
                           177   ;Sub# & Started & Revised : 2 & 9-6-1992  &  9-6-1992   &   Rv0.01
                           178   ;Assumes : DATA0 tablosunda giris degerlerinin oldugu farzediliyor.
                           179   ;Returns : XREAL ve XIMAG dizilerinde DATA0 tablosundaki giris degerleri
                           180   ;          olusur.
                           181   ;Regs Modified : ax,bx,cx,dx
                           182   ;Call(s)       : -
                           183   ;Purpose       : DATA0 tablosunda verilen degerlerin xreal ve ximag giris
                           184   ;                dizilerine yuklenmesi
                           185   ;=============================================================================
    2099                   186   TABLE_LOAD:
    2099 01A0              187     clr    bx
    209B A13A299E          188     ld     ax,#data0              ; Tablodaki degerler giris dizisine
                           189                                   ;yukleniyor.
    209F                   190   LOAD:
    209F A29FA2            191     ld     cx,[ax]+
    20A2 A29FA4            192     ld     dx,[ax]+
    20A5 C3A10030A2        193     st     cx,xreal[bx]
    20AA C3A14030A4        194     st     dx,ximag[bx]
    20AF 650200A0          195     add    bx,#2
    20B3 894000A0          196     cmp    bx,#64
    20B7 DEE6              197     blt    LOAD
    20B9 F0                198     ret
                           199
                           200   ;=============================================================================
                           201   ;Sub# & Started & Revised : 3 & 21-5-1992  &  21-5-1992   &   Rv0.01
                           202   ;Assumes : -
                           203   ;Returns : -
                           204   ;Regs Modified : zero, offset, FFT_OUT[0...31]
                           205   ;Call(s)       : -
                           206   ;Purpose       : FFT_OUT dizisinin temizlenmesi
                           207   ;=============================================================================
    20BA                   208   CLR_FFT_OUT:
    20BA 0140              209     clr    zero
    20BC 019C              210     clr    offset
    20BE                   211   CLRR:
    20BE C39D803040        212     st     zero,fft_out[offset]    ; FFT_OUT[offset] <----- 0
    20C3 6502009C          213     add    offset,#2               ; FFT_OUT dizisi wordlardan olustugundan
    20C7 8940009C          214     cmp    offset,#64              ;  offset degeri 2'ser artiyor.
    20CB DEF1              215     blt    CLRR                    ; FFT_OUT dizisi 32 word'luk bir dizidir.
    20CD F0                216     ret
                           217
                           218   ;=============================================================================
                           219   ;Sub# & Started & Revised : 4 & 26_4-1992  &  21-5-1992   &   Rv1.01
                           220   ;Assumes : XREAL ve XIMAG dizilerinde giris degerlerinin olmasi gereklidir.
                           221   ;Returns : XOUT dizisinde FFT hesaplamalarinin sonucu olusur.
                           222   ;Regs Modified : error, fft_mode, loop_cnt, shft_cnt, ndiv2, kptr, in_cnt,
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
                          223  ;              pwr, wrp, wip, kn2, tmpr, tmpi, tmpr1, tmpi1, xrtmp, xitmp
                          224  ;              xreal[0...31], ximag[0...31]
                          225  ;Call(s)      : -
                          226  ;Purpose      : FFT Hesaplamalarinin gerceklestirilmesi
                          227  ;=============================================================================
   20CE                   228  FFT_CALC:                       ;; void fft_algorithm()
   20CE 1144              229    clrb   error                  ; hatanin kodu error degiskenine yazilacak
                          230
   20D0 FC                231    clrvt                         ; VT flagi hata kontrolu icin temizleniyor
   20D1 B10195            232    ldb    loop_cnt,#1            ; ana dongu registerinin ilk degeri
   20D4 B10494            233    ldb    shft_cnt,#4            ; 2^EXPONENT=N ve SHFT_CNT=EXPONENT-1
   20D7 A1200088          234    ld     ndiv2,#32              ; N=32*2 nokta sayisi ve NDIV2=N/2
   20DB                   235  OUT_LOOP:                       ;; {
   20DB 018A              236    clr    kptr                   ;;   int k=0;
                          237    ;                             ;;   for(loop=1; loop(=exponent; ++loop)
   20DD 990595            238    cmpb   loop_cnt,#5            ;    /* 32=2^5 */
   20E0 DA0220A5          239  !  bgt    FFT_C_END
   20E4                   240  MID_LOOP:                       ;;   do
   20E4 01B6              241    clr    in_cnt                 ;;   { incnt=0;
   20E6                   242  IN_LOOP:                        ;;     do
   20E6 650200B6          243    add    in_cnt,#2              ;;     { ++incnt;
                          244    ;                             ;;       x=k/pow(2,shift);
   20EA A08AB4            245    ld     pwr,kptr               ;;       p=br[x];
   20ED 089484            246    shr    pwr,shft_cnt           ;        /* Agirlik Faktorlerinin hesabi
   20F0 71FEB4            247    andb   pwr,#11111110B         ;           pwr ----> cift sayi
   20F3 A385002584        248    ld     pwr,brev[pwr]          ;           pwr ----> bit-reversal degeri */
   20F8                   249  GW:
   20F8 A385542680        250    ld     wrp,wr[pwr]            ;;       wrp=wr[p];
   20FD A385962682        251    ld     wip,wi[pwr]            ;;       wip=wi[p];
   2102 44888ABC          252    add    kn2,kptr,ndiv2         ;;       kn2=k+n2;
   2106                   253  GM:                             ;        /* Komplex Carpmalar
   2106 FE4FBD00308048    254    mul    tmpr,wrp,xreal[kn2]    ;;       tmpr=(wrp*xr[kn2] - wip*xi[kn2])/2;
   210D FE4F8D4030824C    255    mul    tmpi,wip,ximag[kn2]
   2114 684E4A            256    sub    tmpr+2,tmpi+2
   2117 FE4F8D40308050    257    mul    tmpr1,wrp,ximag[kn2]   ;;       tmpi=(wrp*xi[kn2] + wip*xr[kn2])/2;
   211E FE4F8D0030824C    258    mul    tmpi,wip,xreal[kn2]
   2125 64524E            259    add    tmpi+2,tmpr1+2
                          260    ;   Once word tipi isaretli sayilar carpilarak long tipi tmpr ve tmpi
                          261    ; degerleri elde ediliyor.
                          262    ;   Burada dikkat edilmesi gereken en onemli nokta sudur:
                          263    ;   WR,WI,XREAL,XIMAG dizilerinde saklanan degerler gercek degerlerin 2^15
                          264    ; katidir. Boylece kesirli sayilarla calismak daha kolay olmaktadir. Carpma
                          265    ; islemlerinde elde edilen sonuclar bu nedenle gercek sonuclarin 2^30 kati
                          266    ; olmaktadir. Burada sadece yuksek degerlikli wordlarla cikarma islemi yapi-
                          267    ; lip sonuc yine long tipi registerin yuksek worduna yazilmaktadir. Boylece
                          268    ; sonuc 2^16'ya bolunmus olmaktadir. TMPI ve TMPR degerlerini elde edebilmek
                          269    ; icin toplama ve cikarma islemlerinden sonra 2'ye bolme islemi de yapilmasi
                          270    ; gerektiginden 2^16'ya bolmekle elde edilen sonuc gercek sonucun 2^15 kati
                          271    ; olmaktadir.
                          272
   2128 DC55              273    bvt    ERR1                   ;        /* Carpma islemlerinde tasma
                          274                                  ;           olusursa ERR1'e dallanacak. */
   212A A3BB003050        275    ld     tmpr1,xreal[kptr]      ;;       tmpr1=xr[k]/2;
   212F 0A0150            276    shra   tmpr1,#1
   2132 A3BB403054        277    ld     tmpi1,ximag[kptr]      ;;       tmpi1=xi[k]/2;
   2137 0A0154            278    shra   tmpi1,#1
   213A                   279  GR2:
```

```
ERR LOC  OBJECT          LINE       SOURCE STATEMENT
    213A 484A5058         280     sub    xrtmp,tmpr1,tmpr+2      ;;      xr[k+n2]=tmpr1-tmpr;
    213E C38D00305B       281     st     xrtmp,xreal[kn2]
    2143                  282   GX2:
    2143 484E545C         283     sub    xitmp,tmpi1,tmpi+2      ;;      xi[k+n2]=tmpi1-tmpi;
    2147 C38D40305C       284     st     xitmp,ximag[kn2]
    214C                  285   G1:                               ;;
    214C 444A5058         286     add    xrtmp,tmpr1,tmpr+2      ;;      xr[k]=tmpr1+tmpr;
    2150 C38B003058       287     st     xrtmp,xreal[kptr]
    2155                  288   GX:
    2155 444E545C         289     add    xitmp,tmpi1,tmpi+2      ;;      xi[k]=tmpi1+tmpi;
    2159 C38B40305C       290     st     xitmp,ximag[kptr]
    215E DC24             291     bvt    ERR2                    ;       /* Toplama islemlerinde tasma
                          292                                    ;          olusursa ERR2'e dallanacak. */
    2160                  293   IK:
    2160 6502008A         294     add    kptr,#2                 ;;      ++k;
    2164 B88886           295     cmp    in_cnt,ndiv2            ;;    }
    2167 D602277B         296   !  blt    IN_LOOP                ;;    while (incnt<n2);
    216B 648888A          297     add    kpir,ndiv2              ;;   k=k+n2;
                          298                                    ;    /* n=64 ve n1=n-1=62 */
    216E 893E008A         299     cmp    kptr,#62                ;;  }
    2172 D602276E         300   !  blt    MID_LOOP               ;;  while (k<n1);
    2176 1795             301     incb   locp_cnt                ;;  n2=n2/2;
    2178 0A0188           302     shra   ndiv2,#1
    217B 1594             303     decb   shft_cnt                ;;  --shift;
    217D 275C             304     br     OUT_LOOP                ;;  }
    217F                  305   ERR1:                            ;   /* Carpma islemlerinde bir hata varsa
    217F B10144           306     ldb    error,#1                ;      error=1 yapilarak program duracak
    2182 225C             307     br     ENDP
    2184                  308   ERR2:                            ;      Toplama islemlerinde bir hata varsa
    2184 B10244           309     ldb    error,#2                ;      error=2 yapilarak program duracak  */
    2187 2257             310     br     ENDP
    2189                  311   FFT_C_END:
    2189 F0               312     rat                            ;; }
                          313
                          314   ;=============================================================================
                          315   ;Sub# & Started & Revised : 5 & 26_4-1992  &  21-5-1992   &   Rv1.01
                          316   ;Assumes : -
                          317   ;Returns : -
                          318   ;Regs Modified : kptr, n_sub_k, scale_factor, rk, xrrk, xirk, rrk, xrrnk,
                          319   ;                xirnk, tmpr, tmpi, xrtmp, xitmp, tmpr1, tmpi1,
                          320   ;                outr[0...31], outi[0...31]
                          321   ;Call(s)       : CALC_LOG , CALC_SQRT
                          322   ;Purpose       : Son islemler ve yeniden siralama
                          323   ;=============================================================================
    218A                  324   UNWEAVE:                         ;; void post_processing()
    218A 018A             325     clr    kptr                    ;; { int k;
    218C A140008E         326     ld     n_sub_k,#64
    2190 0146             327     clr    scale_factor            ;;
    2192                  328   UN_LOOP:                         ;; for (k=0; k<32; ++k)
    2192 A38B002590       329     ld     rk,brev[kptr]           ;; {
    2197 A391003060       330     ld     xrrk,xreal[rk]          ;;  xrbrk=xr[br[k]];
    219C 0660             331     ext    xrrk                    ;   /* xrrk<---xrrk ; xrrk+2<---sign(xrrk)*/
    219E A391403068       332     ld     xirk,ximag[rk]          ;;  xibrk=xi[br[k]];
    21A3 0668             333     ext    xirk                    ;   /* xirk<---xirk ; xirk+2<---sign(xirk)*/
    21A5 A38F002592       334     ld     rnk,brev[n_sub_k]       ;;  xrbrnk=xr[br[n-k]];
    21AA A393003064       335     ld     xrrnk,xreal[rnk]
    21AF 0664             336     ext    xrrnk                   ;   /* xrrnk<--xrrnk;xrrnk+2<--sign(xrrnk)*/
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
    21B1 A39340306C       337      ld      xirnk,ximag[rnk]      ;;  xibrnk=xi[br[n-k]];
    21B6 066C             338      ext     xirnk                 ;   /* xirnk<--xirnk;xirnk+2<--sign(xirnk)*/
    21B8                  339    AR:                             ;;  ti=(xibrk+xibrnk)/2;
    21B8 446C6B4C         340      add     tmpi,xirk,xirnk       ;   /* Once dusuk degerlikli wordlar topla-
    21BC A06E4E           341      ld      tmpi+2,xirnk+2        ;       niyor. Sonra yuksek degerlikli word-
    21BF A46A4E           342      addc    tmpi+2,xirk+2         ;       larla carry flagi toplaniyor. */
    21C2 0E014C           343      shral   tmpi,#1
    21C5 4864604B         344      sub     tmpr,xrrk,xrrnk       ;;  tr=(xrbrk-xrbrnk)/2;
    21C9 A0624A           345      ld      tmpr+2,xrrk+2         ;   /* Once dusuk degerlikli wordlar cikar-
    21CC A8664A           346      subc    tmpr+2,xrrnk+2        ;       tiliyor. Sonra yuksek degerlikli
    21CF 0E014B           347      shral   tmpr,#1               ;       wordlarla carry flagi cikartiliyor.*/
    21D2 44646058         348      add     xrtmp,xrrk,xrrnk      ;;  xrt=(xrbrk+xrbrnk)/4;
    21D6 A0625A           349      ld      xrtmp+2,xrrk+2        ;   /* Once dusuk degerlikli wordlar topla-
    21D9 A4665A           350      addc    xrtmp+2,xrrnk+2       ;       niyor. Sonra yuksek degerlikli word-
    21DC 0D0E58           351      shll    xrtmp,#14             ;       larla carry flagi toplaniyor. */
    21DF 486C6B5C         352      sub     xitmp,xirk,xirnk      ;;  xit=(xibrk-xibrnk)/4;
    21E3 A06A5E           353      ld      xitmp+2,xirk+2        ;   /* Once dusuk degerlikli wordlar cikar-
    21E6 A86E5E           354      subc    xitmp+2,xirnk+2       ;       tiliyor. Sonra yuksek degerlikli
    21E9 0D0E5C           355      shll    xitmp,#14             ;       wordlarla carry flagi cikartiliyor.*/
    21EC                  356    MR:
    21EC FE4F8B50254B50   357      mul     tmpr1,tmpr,sinfn[kptr]  ;;  outr=xrt+ti*cos(kpin)/2-tr*sin(kpin)/2;
    21F3 FE4F8BD2254C54   358      mul     tmpi1,tmpi,cosfn[kptr]
    21FA 645458           359      add     xrtmp,tmpi1
    21FD A4565A           360      addc    xrtmp+2,tmpi1+2
    2200 685058           361      sub     xrtmp,tmpr1
    2203 A8525A           362      subc    xrtmp+2,tmpr1+2
    2206 C38BC0305A       363      st      xrtmp+2,outr[kptr]      ;   /* outr[0..31]:Reel cikis degerleri */
    220B                  364    MI:
    220B FE4F8BD2254B50   365      mul     tmpr1,tmpr,cosfn[kptr]  ;;  outi=xit-ti*sin(kpin)/2-tr*cos(kpin)/2;
    2212 FE4F8B50254C54   366      mul     tmpi1,tmpi,sinfn[kptr]
    2219 68545C           367      sub     xitmp,tmpi1
    221C A8565E           368      subc    xitmp+2,tmpi1+2
    221F 68505C           369      sub     xitmp,tmpr1
    2222 A8525E           370      subc    xitmp+2,tmpr1+2
    2225 C38B00315E       371      st      xitmp+2,outi[kptr]      ;   /*outi[0..31]:Imajiner cikis degerleri*/
    222A                  372    GET_MAG:                        ;;  magsq=pow(outr,2)+pow(outi,2);
    222A A05A4B           373      ld      tmpr,xrtmp+2
    222D A05E4C           374      ld      tmpi,xitmp+2
    2230 FE6C4B48         375      mul     tmpr,tmpr
    2234 FE6C4C4C         376      mul     tmpi,tmpi
    2238 644C4B           377      add     tmpr,tmpi
    223B A44E4A           378      addc    tmpr+2,tmpi+2
    223E 324504           379      bbc     fft_mode,2,C_SQRT       ;   /* FFT_MODE degiskeninin 2. biti 0 ise
                          380                                      ;       karekok; 1 ise magnitude secilir. */
    2241                  381    C_LOG:
    2241 2811             382      call    CALC_LOG
    2243 2002             383      br      UNWEAVE_END
    2245                  384    C_SQRT:
    2245 2855             385      call    CALC_SQRT
    2247                  386    UNWEAVE_END:
    2247 6502008A         387      add     kptr,#2
    2248 6902008E         388      sub     n_sub_k,#2
    224F DF02273F         389  !   bne     UN_LOOP
    2253 F0               390      ret                             ;; }
                          391                                      ;; }
                          392  ;=============================================================================
                          393  ;Sub# & Started & Revised : 6 & 26_4-1992  &  21-5-1992  &  Rv1.01
```

```
ERR LOC  OBJECT            LINE      SOURCE STATEMENT
                            394  ;Assumes : -
                            395  ;Returns : -
                            396  ;Regs Modified : shft_cnt, tmpr, log, ptr, nxtloc, diff, FFT_OUT[0..31]
                            397  ;Call(s)       : -
                            398  ;Purpose       : 10*LOG(TMPR/65536) degerlerinin hesaplanmasi
                            399  ;=============================================================================
    2254                    400  CALC_LOG:                          ; Cikis = 512*LOG(x)   x=1,2,3,...,64K
    2254 0194               401    clr    shft_cnt
    2256 0F9448             402    norml  tmpr,shft_cnt              ; tmpr degiskeni en yuksek degerlikli biti
    2259 990F94             403    cmpb   shft_cnt,#15               ; 1 oluncaya kadar sola kaydirilir ve kay-
                            404                                      ; dirma sayisi shft_cnt degiskenine konur.
    225C DA04               405    jle    LOG_IN_RANGE               ; Eger shft_cnt <= 15 ise LOG_IN_RANGE
                            406                                      ; adresine atlayacak.
    225E 017B               407    clr    log                        ; Eger shft_cnt=15 ise log=0 ve
    2260 202C               408    br     LOG_STORE                  ; Program LOG_STORE adresine dallanir.
    2262                    409  LOG_IN_RANGE:
    2262 44949494           410    add    shft_cnt,shft_cnt,shft_cnt ; SHIFT_COUNT bir pointer yapiliyor.
    2266 AC4B96             411    ldbze  ptr,tmpr+3                 ; En yuksek degerlikli olan byte
                            412                                      ; tablo pointeri olarak kullaniliyor.
    2269 44969696           413    add    ptr,ptr,ptr                ; ptr=ptr*2
    226D 65182796           414    add    ptr,# log_table-256        ; PTR= Table + offset (offset=tmpr+3)
                            415                                      ; tmpr+3 >= 128 oldugu sure boyunca
                            416                                      ; -256 kullanilacak.
    2271 A29778             417    ld     log, [ptr]+                ; log=mem_word(ptr)  ;  ptr=ptr+2
    2274 A2967C             418    ld     nxtloc, [ptr]              ; nxtloc=mem_word(ptr)
                            419                                      ; lineer interpolation
    2277 68787C             420    sub    nxtloc,log                 ; nxtloc = (bir sonraki log) - log
    227A AC4A70             421    ldbze  diff,tmpr+2                ; diff+1 = nxtloc * tmpr+2 / 256
    227D 6C7C70             422    mulu   diff,nxtloc
    2280 0C0870             423    shrl   diff,#8                    ; log = log + diff/256
    2283 647078             424    add    log,diff
    2286 0B0578             425    shr    log,#5                     ; 8192/32 * 20LOG(x) = 256 * 20LOG(x)
    2289 A7951A2978         426    addc   log,log_offset[shft_cnt]   ; Normalization Faktoru + log
                            427                                      ; Log(M*N) = Log M + Log N
    228E                    428  LOG_STORE:
                            429    ;shr    log,#scale_factor          ; Cikislarin ortalamasinin alinmasi sira-
    228E A44078             430    addc   log,zero                   ;  sinda tasmayi onlemek amaciyla bolme
                            431                                      ;  islemi yapiliyor.
    2291 678B803078         432    add    log,fft_out[kptr]          ; log=log+FFT_OUT[kptr]
    2296 C3BB803078         433    st     log,fft_out[kptr]          ; FFT_OUT[kptr]=log
    229B F0                 434    ret
                            435
                            436  ;=============================================================================
                            437  ;Subs & Started & Revised : 7 & 26-4-1992  &  21-5-1992   &   Rv1.01
                            438  ;Assumes : -
                            439  ;Returns : -
                            440  ;Regs Modified : shft_cnt, tmpr, ptr, sqrt, nxtloc, FFT_OUT[0...31]
                            441  ;Call(s)       : -
                            442  ;Purpose       : Karekok degerinin hesaplanmasi
                            443  ;=============================================================================
    229C                    444  CALC_SQRT:
    229C 0194               445    clr    shft_cnt
    229E 0F9448             446    norml  tmpr,shft_cnt              ; Normalize ve Normalization Faktorunun
                            447                                      ;  elde edilmesi
    22A1 D705               448    jne    SQRT_IN_RANGE              ; tmpr > 0 ise SQRT_IN_RANGE adresine atla
    22A3 C07640             449    st     zero,sqrt+2                ; Degilse mem_word(sqrt+2)=0 ve
    22A6 2029               450    br     SQRT_STORE                 ; Program SQRT_STORE adresine dallanacak.
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
    22AB                  451   SQRT_IN_RANGE:
    22AB AC4B96           452     ldbze  ptr,tmpr+3              ; En yuksek degerlikli olan byte
                          453                                    ;  Tablo Pointeri'dir.
    22AB 44969696         454     add    ptr,ptr,ptr             ; ptr=ptr*2
    22AF 65182696         455     add    ptr,# sq_table-256      ; ptr = Table + Offset ( offset=tmpr+3 )
                          456                                    ;  tmpr+3, 128'den buyuk oldugu sure
                          457                                    ;  boyunca -256 kullanilacak.
    22B3 A29774           458     ld     sqrt, [ptr]+            ; sqrt=mem_word(ptr)  ;  ptr=ptr+2
    22B6 A2967C           459     ld     nxtloc, [ptr]           ; nxtloc=mem_word(ptr)
                          460                                    ; lineer interpolation
    22B9 68747C           461     sub    nxtloc,sqrt             ; nxtloc = sqrt - bir sonraki sqrt
    22BC AC4A70           462     ldbze  diff,tmpr+2             ; diff+1 = nxtloc * tmpr+2 / 256
    22BF 6C7C70           463     mulu   diff,nxtloc
    22C2 AC7170           464     ldbze  diff,diff+1             ; sqrt = sqrt + delta   (diff < 0FFH)
    22C5 647074           465     add    sqrt,diff
    22C8 44949494         466     add    shft_cnt,shft_cnt,shft_cnt   ; shft_cnt=shft_cnt*2
    22CC 6F95D82674       467     mulu   sqrt,tab_sqr[shft_cnt]  ; Normalization Faktorune bolunuyor.
                          468                                    ; Eger tab2=0FFFFH ise mulu, bolme islemi
                          469                                    ;  yerine gecer. sqrt degismeden kalir.
    22D1                  470   SQRT_STORE:
                          471     ;shr   sqrt+2,#scale_factor    ; Cikislarin ortalamasinin alinmasi sira-
    22D1 A44076           472     addc   sqrt+2,zero             ;  sinda tasmayi onlemek amaciyla bolme
                          473                                    ;  islemi yapiliyor.
    22D4 67BB803076       474     add    sqrt+2,fft_out[kptr]    ; mem_word(sqrt+2)=mem_word(sqrt+2)
    22D9 C3BB803076       475     st     sqrt+2,fft_out[kptr]    ;                  +FFT_OUT[kptr]
                          476                                    ; FFT_OUT[kptr]=mem_word(sqrt+2)
    22DE F0               477     ret
                          478
                          479   ;=============================================================================
                          480   ;Subr & Started & Revised : 8 & 8-6-1992  &  8-6-1992   &   Rv1.00
                          481   ;Assumes : Evaluation Board'in 1. analog girisinden bir analog sinyalin
                          482   ;          verildigi farzediliyor.
                          483   ;Returns : XREAL ve XIMAG dizilerinde analog sinyalin orneklenmis degerleri
                          484   ;Regs Modified : control_a2d, sample_period, src_ptr, dest_ptr, loop_c,
                          485   ;                aductemp0, top_of_buffer, aductemp1, ad_command, temp,
                          486   ;                hso_0_high, aductemp0, start_a2d, hso_0_low, ad_result_lo,
                          487   ;                ad_result_hi, sample_count
                          488   ;Call(s)       : -
                          489   ;Purpose       : Evaluation Board'a disaridan alinan analog sinyallerin
                          490   ;                digitale cevrilip hafizada bir buffer'a atilmasi
                          491   ;=============================================================================
                          492   ;
                          493   ; MCS96 Evalution Board'un JP1 Analog Input Connector'u araciligi ile
                          494   ;disaridan alinan analog sinyaller ADC cevirici yardimiyla digitale
                          495   ;cevrilmektedir. Bu program  yardimiyla bu degerler bir buffer'a
                          496   ;aktarilmaktadir. A/D cevrimi interrupt kontrolu altinda yapilir.
                          497   ;A2D_BUFF_Util altprogrami ile A/D cevrimi baslangic durumuna set edilir.
                          498   ;A/D cevrim sonuclari BUFF0 ve BUFF1 isminde iki buffer'da saklanir.
                          499   ;
                          500   ; Bu program, kullaniciya, buffer uzunlugunu, data formatini, orneklema
                          501   ;periyodunu ve cevrimin yapilacagi kanali secme olanagi tanimaktadir. Ayrica,
                          502   ;bu program, buffer'lari bir register file buffer'ina yukleyebilen "download"
                          503   ;isminde bir alt programa da sahiptir. Cikis formati bu alt programin
                          504   ;isleyebilecegi sekilde secilebilir. Veriler, isaretli veya isaretsiz, lineer
                          505   ;veya ikiserli dizilere yerlestirilecek sekilde secilebilir.
                          506   ;
                          507   ; Kullaniciya Sunulan Secenekler:
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
                          508   ; Bu programin yonlendirilmesi hafizadaki iki kontrol kelimesi yardimiyla
                          509   ;yapilir. Programda A/D cevrimlerinin dogru yapilabilmesi icin, A2D_BUFF_Util
                          510   ;alt programi cagrildiginda kontrol kelimelerinin hazir olmasi gerekmektedir.
                          511   ;
                          512   ; Kontrol kelimeleri asagidaki yapida olmalidir:
                          513   ;
                          514   ; Sample_Period     : word   ; 'Timer' tanimlandiktan sonra ornekler
                          515   ;                            ;arasindaki zamanin timer cinsinden degeri
                          516   ; Control_A2D       : byte   ; Asagida verilen sekilde kontrol bilgisini
                          517   ;                            ;icerir.
                          518   ;                            ; BIT#
                          519   ;                              0-2   : Kanal numarasi
                          520   ;                               3    : Isaretli/Isaretsiz sonuc
                          521   ;                               4    : Convert/Download
                          522   ;                               5    : BUFF1/BUFF0 cevrim icin
                          523   ;                                    : BUFF0/BUFF1 download icin
                          524   ;                               6    : Lineer/Ikiserli
                          525   ;                               7    : Cevirici mesgul/serbest
                          526   ; Asagidaki tablo, Control_A2D registerinin alacagi degerleri gostermektedir.
                          527   ;Cevrimi baslatmak icin LDB Control_A2D ile beraber asagidaki degerlerden
                          528   ;biri ve ORB Control_A2D komutu ile beraber de kanal numarasi verilmelidir.
                          529   ;
                          530   ; Control_A2D registerine yuklenebilecek degerler:
                          531   ;
                          532   ; con_b0        equ  00010000b ; Cevrim BUFF0 buffer'ina yapilacak.
                          533   ; con_b1        equ  00110000b ;   "    BUFF1    "          "
                          534   ; dump_b0_l_u   equ  01100000b ; download BUFF0'a lineer  isaretsiz data
                          535   ; dump_b1_l_u   equ  01100000b ;     "    BUFF1'e   "        "     "
                          536   ; dump_b0_p_u   equ  00100000b ;     "    BUFF0'a ikiserli   "     "
                          537   ; dump_b1_p_u   equ  00000000b ;     "    BUFF1'e    "       "     "
                          538   ; dump_b0_l_s   equ  01101000b ;     "    BUFF0'a lineer   isaretli  "
                          539   ; dump_b1_l_s   equ  01001000b ;     "    BUFF1'e    "       "     "
                          540   ; dump_b0_p_s   equ  00101000b ;     "    BUFF0'a ikiserli   "     "
                          541   ; dump_b1_p_s   equ  00001000b ;     "    BUFF1'e    "       "     "
                          542   ;
                          543   ; Bazi registerlerin icerikleri:
                          544   ; BUFF0_BASE: BUFF0'in baslangic adresi
                          545   ; BUFF1_BASE: BUFF1'in baslangic adresi
                          546   ; DEST_BUFF_BASE: Download yapilacak hedef bufferin baslangic adresi
                          547   ; BUFF_LENGTH: Her buffer'in tutmasi gereken ornek sayisi (1 ile 256 arasi)
                          548   ; SHFT_CNT: Cevrim sonucunun 2'ye kac defa boluneceğini gosterir.
                          549   ; CLOCK: TIMER1 veya T2CLK olarak tanimlanir. Bu deger cevrimlerde temel
                          550   ;       zaman olarak alinir.
                          551   ;
                          552   ; Ornekler bufferlarda word olarak saklanir. Cevrim sonuclari BUFF0 ve
                          553   ;BUFF1'e lineer olarak yerlestirilir. Hedef buffer'a ise istege bagli olarak
                          554   ;lineer veya ikiserli olarak yerlestirilir. Eger download, ikiserli olacaksa
                          555   ;ilk ornek DEST_BUFF_BASE adresine, ikinci ornek DEST_BUFF_BASE + BUFF_LENGTH
                          556   ;adresine, ucuncu ornek DEST_BUFF_BASE + 2 adresine ve dorduncu ornek
                          557   ;DEST_BUFF_BASE + 2 + BUFF_LENGTH adresine yerlestirilir.
                          558   ;
                          559   ; "Cevirici mesgul(converter busy)" biti, program cagrildiktan sonra ilk 50
                          560   ;state_time'da "1"'dir. Bu bit her cevrim sonucunun tanimlanan buffer'a
                          561   ;(BUFF0 ve BUFF1)'de saklanmasindan sonra silinir.
                          562   ;
                          563   ; CONTROL_A2D registeri icin komutlar:
0007                      564     busy          equ    7        ; busy biti, 7. bit
```

```
ERR LOC  OBJECT          LINE      SOURCE STATEMENT
     0010                 565   con_b0        equ     00010000b ; A/D cevrim sonucu BUFF0'a konacak.
     0028                 566   dump_b0_p_s   equ     00101000b ; BUFF0'a ikiserli isaretli datalar
                          567                                     ;  download edilecek.
                          568 ; Diger sabitler:
     0040                 569   buff_lengtn   equ     64
     0001                 570   shift_count   equ     1
     000A                 571   timer1        equ     0ah
     000C                 572   timer2        equ     0ch
     000C                 573   t2clk         equ     0ch
     000A                 574   clock         equ     timer1
     0000                 575   mask          equ     (10h*clock)and(40h)
     000F                 576   start_a2d     equ     (00001111b)or(mask) ; A/D cevrimi timer1'e bagli
                          577                                             ;olarak calistirilacak.
                          578                                             ;(Interrupt'a bagli degil)
     0000                 579   hso_0_low     equ     (00000000b)or(mask) ; HSO_0'in dusuk byte'i timer1'e
                          580                                             ;bagli olarak calistirilacak.
                          581                                             ;Interrupta bagli degil.
     0020                 582   hso_0_high    equ     (00100000b)or(mask) ; HSO_0'in yuksek byte'i timer'e
                          583                                             ;bagli olarak calistirilacak.
                          584                                             ;Interrupta bagli degil.
     0003                 585   cform         equ     3                   ; isaretli/isaretsiz#
     0004                 586   con_dwn       equ     4                   ; convert/download#
     0005                 587   b0_b1         equ     5                   ; cevrim icin buff1/buff0#
                          588                                             ; download icin buff0/buff1#
     0006                 589   lin_par       equ     6                   ; lineer/ikiserli#
     00B0                 590   busy2         equ     10000000b           ; busy bitinin konumu
     0002                 591   ad_command    equ     02h:byte
     0006                 592   hso_command   equ     06h:byte
     0004                 593   hso_time      equ     04h:word
     0002                 594   ad_result_lo  equ     02h:byte
     0003                 595   ad_result_hi  equ     03h:byte
   22DF                   596 LOAD_DATA:
   22DF B110B1            597   ldb   control_a2d,#con_b0      ; Cevrim sonucu BUFF0'a konacak.
   22E2 9101B1            598   orb   control_a2d,#01         ; Analog sinyal girisinde kanal-1
                          599                                 ;  kullanilacak.
   22E5 A13200A6          600   ld    sample_period,#50       ; Ornekleme periyodu 100 mikrosaniye
   22E9 2809              601   call  a2d_buff_util           ; A/D cevrim islemi basliyor.
   22EB 3FB1FD            602   jbs   control_a2d,busy,$      ; Cevrim tamamlanincaya kadar bekleyecek.
   22EE                   603 DOWNL:
   22EE B128B1            604   ldb   control_a2d,#dump_b0_p_s ; BUFF0'a ikiserli isaretli data
                          605                                 ;  yerlestirilecek.
   22F1 2801              606   call  a2d_buff_util
   22F3 F0                607   ret
   22F4                   608 A2D_BUFF_UTIL:
   22F4 3CB162            609   jbs   control_a2d,con_dwn,convert  ; convert/download secimi
   22F7                   610 DOWNLOAD:
   22F7 A14030B2          611   ld    src_ptr,#BUFF1_BASE
   22FB 35B104            612   jbc   control_a2d, b0_b1, set_data_format   ; data tipinin belirlenmesi
   22FE                   613 DOWNLOAD_BUFF0:
   22FE A10030B2          614   ld    src_ptr,#BUFF0_BASE
   2302                   615 SET_DATA_FORMAT:                ; lineer/ikiserli secimi
   2302 A10001B4          616   ld    dest_ptr,#DEST_BUFF_BASE
   2306 B140B6            617   ldb   loop_c,#BUFF_LENGTH
   2309 3EB11D            618   jbs   CONTROL_A2D, lin_par, lineer_data_loop
   230C                   619 PAIRED:
   230C 1801B6            620   shrb  loop_c,#1               ; ikiserli data rutini, lineer data
                          621                                 ;  rutuni'nin 1/2'si kadar donguye sahip.
```

```
ERR LOC  OBJECT          LINE     SOURCE STATEMENT
    230F                  622  PAIRED_DATA_LOOP:
    230F A2B3A8           623    ld      adudtemp0,[src_ptr]+      ; cift sayili wordlari aliyor.
    2312 C2B4A8           624    st      adudtemp0,[dest_ptr]
    2315 654000B4         625    add     dest_ptr,#buff_length
    2319 A2B3A8           626    ld      adudtemp0,[src_ptr]+      ; tek sayili wordlari aliyor.
    231C C2B5A8           627    st      adudtemp0,[dest_ptr]+
    231F 694000B4         628    sub     dest_ptr,#buff_length
    2323 E0B6E9           629    djnz    loop_c, paired_data_loop
    2326 280D             630    call    convert_data
    2328 F0               631    ret
    2329                  632  LINEER_DATA_LOOP:
    2329 A2B3A8           633    ld      adudtemp0,[src_ptr]+
    232C C2B5A8           634    st      adudtemp0,[dest_ptr]+
    232F E0B6F7           635    djnz    loop_c,lineer_data_loop
    2332 2801             636    call    Convert_data
    2334 F0               637    ret
    2335                  638  CONVERT_DATA:
    2335 A14000B6         639    ld      loop_c,#buff_length
    2339 A10001B2         640    ld      src_ptr,#dest_buff_base
    233D                  641  L_AGAIN:
    233D A2B2A9           642    ld      adudtemp0,[src_ptr]
    2340 71C0A8           643    andb    adudtemp0,#11000000b
    2343 33B109           644    jbc     control_a2d, dfora, unsigned_result
    2346                  645  SIGNED_RESULT:
    2346 69E07FA8         646    sub     adudtemp0,#7fa0h
    234A 0A01A8           647    shra    adudtemp0,#shift_count
    234D 2003             648    br      replace_sample
    234F                  649  UNSIGNED_RESULT:
    234F 0801A8           650    shr     adudtemp0,#shift_count
    2352                  651  REPLACE_SAMPLE:
    2352 C2B3A8           652    st      adudtemp0,[src_ptr]+
    2355 E0B6E5           653    djnz    loop_c,l_again
    2358 F0               654    ret
    2359                  655  CONVERT:
    2359 F2               656    pushf
    235A 9180B1           657    orb     control_a2d,#busy2        ; "cevirici mesgul" biti set ediliyor.
    235D B13FB0           658    ldb     sample_count,#buff_length-1
    2360 A10030AE         659    ld      top_of_buffer,#buff0_base
    2364 A18030AC         660    ld      aductemp1,#(buff0_base+2*buff_length)
    2368 35B108           661    jbc     control_a2d,b0_b1,start_conversions
    236B A14030AE         662    ld      top_of_buffer,#buff1_base
    236F A1C030AC         663    ld      aductemp1,#(buff1_base+2*buff_length)
    2373                  664  START_CONVERSIONS:
    2373 5107B102         665    andb    ad_command,control_a2d,#00000111b  ; kanal numarasi yukleniyor.
    2377 44A60AAA         666    add     aductemp0,clock,sample_period      ; Su andan itibaren bir ornek
                          667                                               ;suresi boyunca ilk cevrim
                          668                                               ;gerceklestirilecek.
                          669    load_hso_command start_a2d                 ; time=aductemp0 olduncaya kadar
                          672                                               ;cevrimi yap.
    2381 CCB2             673    pop     temp                               ; psw'un kopyasi aliniyor.
                          674    load_hso_command hso_0_high                ; hso_0_high, her cevrim
                          677                                               ;baslangicinda zamana bagli
                          678                                               ;olarak set olacak.
    2389 810202B2         679    or      temp,#202h                         ; A/D interruptu enable oldu.
    238D 64A6AA           680    add     aductemp0,sample_period
                          681    load_hso_command start_a2d                 ; Ilk ornekten, bir ornek zamani
                          684                                               ;sonra ikinci ornek aliniyor.
```

```
ERR LOC  OBJECT            LINE      SOURCE STATEMENT
    2396 C8B2              685     push   temp                          ; psw, tekrar stack'a atiliyor.
                           686     load_hso_command hso_0_low
    239E F3                689     popf
    239F F0                690     ret
    23A0                   691   A2D_DONE_VECTOR:                      ; A/D interrupt service routine
    23A0 F2                692     pushf
    23A1 C6AF02            693     stb    ad_result_lo,[top_of_buffer]+
    23A4 C6AF03            694     stb    ad_result_hi,[top_of_buffer]+
    23A7 5107B102          695     andb   ad_command,control_a2d,#00000111b   ; kanal numarasi yuklendi.
    23AB E0B009            696     djnz   sample_count,sample_again
    23AE 17B0              697     incb   sample_count
    23B0 88ACAE            698     cmp    top_of_buffer,aductemp1
    23B3 DF26              699     be     top_of_buffers
    23B5 F3                709     popf
    23B6 F0                701     ret
    23B7                   702   SAMPLE_AGAIN:
    23B7 64A6AA            703     add    aductemp0,sample_period        ; Bir sonraki ornek zamani set ediliyor.
    23BA 88ACAE            704     cmp    top_of_buffer,aductemp1        ; Bir sonraki jump icin buffer'in sonu
                           705                                           ;kontrol ediliyor.
                           706     load_hso_command start_a2d
    23C3 30B00B            709     jbc    sample_count,0,make_hso_high
    23C6                   710   MAKE_HSO_LOW:
    23C6 FD                711     nop
                           712     load_hso_command hso_0_low
    23CD DF0C              715     be     top_of_buffers
    23CF F3                716     popf
    23D0 F0                717     ret
    23D1                   718   MAKE_HSO_HIGH:
                           719     load_hso_command hso_0_high
    23D7 DF02              722     be     top_of_buffers
    23D9 F3                723     popf
    23DA F0                724     ret
    23DB                   725   TOP_OF_BUFFERS:
    23DB 717FB1            726     andb   control_a2d,#not(busy2)        ; ceviricinin busy biti temizleniyor.
    23DE F3                727     popf
    23DF F0                728     ret
                           729
    23E0                   730   ENDP:
    23E0 FD                731     nop
                           732
    2500                   733   CSEG at  2500H
    2500                   734   BREV:                                   ; 2'bit reversal degarleri
    2500 0000200010003000  735     DCW      2#0, 2#16, 2#8,  2#24, 2#4, 2#20, 2#12, 2#28
    2510 0400240014003400  736     DCW      2#2, 2#18, 2#10, 2#26, 2#6, 2#22, 2#14, 2#30
    2520 0200220012003200  737     DCW      2#1, 2#17, 2#9,  2#25, 2#5, 2#21, 2#13, 2#29
    2530 0600260016003600  738     DCW      2#3, 2#19, 2#11, 2#27, 2#7, 2#23, 2#15, 2#31
    2540 0000000000000000  739     DCW        0,    0,    0,    0,   0,    0,    0,    0
                           740
    2550                   741   CSEG at  2550H
    2550                   742   SINFN:                                  ; SIN(K*PI/N)*2^15
    2550 00008C0CF9182825  743     DCW          0,  3212,  6393,  9512, 12539, 15446, 18204, 20787
    2560 825AF1626D6AE270  744     DCW      23170, 25329, 27245, 28898, 30273, 31356, 32137, 32609
    2570 FF7F617F897D7C7A  745     DCW      32767, 32609, 32137, 31356, 30273, 28898, 27245, 25329
    2580 825A33511C47563C  746     DCW      23170, 20787, 18204, 15446, 12539,  9512,  6393,  3212
    2590 000074F307E7DBDA  747     DCW          0, -3212, -6393, -9512,-12539,-15446,-18204,-20787
    25A0 7EA50F9D93951E8F  748     DCW     -23170,-25329,-27245,-28898,-30273,-31356,-32137,-32609
    25B0 01809F8077828485  749     DCW     -32767,-32609,-32137,-31356,-30273,-28898,-27245,-25329
```

```
ERR LOC  OBJECT           LINE     SOURCE STATEMENT
    25C0 7EA5CDAEE4B8AAC3  750   DCW      -23170, -20787, -18204, -15446, -12539,  -9512,  -6393,  -3212
    25D0 0000              751   DCW           0
                           752
    25D2                   753   COSFN:                          ; COS(K*PI/N)*2^15
    25D2 FF7F617F897D7C7A  754   DCW       32767,  32609,  32137,  31356,  30273,  28898,  27245,  25329
    25E2 825A33511C47563C  755   DCW       23170,  20787,  18204,  15446,  12539,   9512,   6393,   3212
    25F2 000074F307E7D8DA  756   DCW           0,  -3212,  -6393,  -9512, -12539, -15446, -18204, -20787
    2602 7EA50F9D93951E8F  757   DCW      -23170, -25329, -27245, -28898, -30273, -31356, -32137, -32609
    2612 01809F8077828485  758   DCW      -32767, -32609, -32137, -31356, -30273, -28898, -27245, -25329
    2622 7EA5CDAEE4B8AAC3  759   DCW      -23170, -20787, -18204, -15446, -12539,  -9512,  -6393,  -3212
    2632 00008C0CF9182B25  760   DCW           0,   3212,   6393,   9512,  12539,  15446,  18204,  20787
    2642 825AF1626D6AE270  761   DCW       23170,  25329,  27245,  28898,  30273,  31356,  32137,  32609
    2652 FF7F              762   DCW       32767
                           763
    2654                   764   WR:                             ; WR= COS(K*2PI/N)
    2654 FF7F897D41766D6A  765   DCW       32767,  32137,  30273,  27245,  23170,  18204,  12539,   6393
    2664 000007E705CFE4B8  766   DCW           0,  -6393, -12539, -18204, -23170, -27245, -30273, -32137
    2674 01807782BF899395  767   DCW      -32767, -32137, -30273, -27245, -23170, -18204, -12539,  -6393
    2684 0000F918FB301C47  768   DCW           0,   6393,  12539,  18204,  23170,  27245,  30273,  32137
    2694 FF7F              769   DCW       32767
                           770
    2696                   771   WI:                             ; WI= -SIN(K*2PI/N)
    2696 000007E705CFE4B8  772   DCW           0,  -6393, -12539, -18204, -23170, -27245, -30273, -32137
    26A6 01807782BF899395  773   DCW      -32767, -32137, -30273, -27245, -23170, -18204, -12539,  -6393
    26B6 0000F918FB301C47  774   DCW           0,   6393,  12539,  18204,  23170,  27245,  30273,  32137
    26C6 FF7F897D41766D6A  775   DCW       32767,  32137,  30273,  27245,  23170,  18204,  12539,   6393
    26D6 0000              776   DCW           0
                           777
    26D8                   778   TAB_SQR:                        ; 65535/(2^SHFT_CNT'nin karekoku)
                           779                                   ; 0 <= shft_cnt < 32
                           780   ;          1       2       4       8      16      32      64     128
    26D8 FFFF04B50080825A  781   DCW       65535,  46340,  32768,  23170,  16384,  11585,  8192,  5793
                           782   ;          256     512     1024    2048    4096    8192    16384  32768
    26E8 0010500B0008AB05  783   DCW       4096,   2896,   2048,   1448,   1024,   724,    512,   362
                           784   ;          65536   131072  262144  524288  ...
    26F8 0001B50080005B00  785   DCW       256,    181,    128,    91,     64,     45,     32,    23
    2708 10000B0008000600  786   DCW       16,     11,     8,      6,      4,      3,      2,     1
                           787
    2718                   788   SQ_TABLE:                       ; n * 2^24'un karekoku
                           789                                   ; N=128, 129, 130 ... 255
    2718 05B5BAB56EB621B7  790   DCW      46341, 46522, 46702, 46881, 47059, 47237, 47415, 47591
    2728 97BA46BBF5BBA3BC  791   DCW      47767, 47942, 48117, 48291, 48465, 48637, 48809, 48981
    2738 00C0AAC054C1FDC1  792   DCW      49152, 49322, 49492, 49661, 49830, 49998, 50166, 50332
    2748 43C5E9C58EC633C7  793   DCW      50499, 50665, 50830, 50995, 51159, 51323, 51486, 51649
    2758 63CA04CBA6CB46CC  794   DCW      51811, 51972, 52134, 52294, 52454, 52614, 52773, 52932
    2768 62CF00D09DD03AD1  795   DCW      53090, 53248, 53405, 53562, 53719, 53874, 54030, 54185
    2778 44D4DED477D511D6  796   DCW      54340, 54494, 54647, 54801, 54954, 55106, 55258, 55410
    2788 09D9A0D936DACCDA  797   DCW      55561, 55712, 55862, 56012, 56162, 56311, 56459, 56608
    2798 B4DD47DEDBDE6EDF  798   DCW      56756, 56903, 57051, 57198, 57344, 57490, 57636, 57781
    27A8 46E2D7E267E3F7E3  799   DCW      57926, 58071, 58215, 58359, 58503, 58646, 58789, 58931
    27B8 C1E64FE7DDE76AE8  800   DCW      59073, 59215, 59357, 59498, 59639, 59779, 59919, 60059
    27C8 27EBB2EB3DECC7EC  801   DCW      60199, 60338, 60477, 60615, 60754, 60891, 61029, 61166
    27D8 77EF00F088F010F1  802   DCW      61303, 61440, 61576, 61712, 61848, 61984, 62119, 62254
    27E8 B4F33BF4C1F446F5  803   DCW      62388, 62523, 62657, 62790, 62924, 63057, 63190, 63323
    27F8 DFF763FBE7F86AF9  804   DCW      63455, 63587, 63719, 63850, 63982, 64113, 64243, 64374
    2808 F8FB7AFCFBFC7DFD  805   DCW      64504, 64634, 64763, 64893, 65022, 65151, 65280, 65408
                           806
```

```
ERR LOC  OBJECT           LINE     SOURCE STATEMENT
    2818                   807  LOG_TABLE:                    ; 16384=10*LOG(n/128)
                           808                                ; n=128,129,130,...,256
    2818 00002A024F047006  809   DCW        0,  554, 1103, 1648, 2190, 2727, 3260, 3789
    2828 DA10E312E914EA16  810   DCW     4314, 4835, 5353, 5866, 6376, 6883, 7386, 7885
    2838 BD20A92292247B26  811   DCW     8381, 8873, 9362, 9848,10330,10810,11286,11758
    2848 C42F973166333335  812   DCW    12228,12695,13158,13619,14076,14531,14963,15432
    2858 063EC13F7A413043  813   DCW    15878,16321,16762,17200,17635,18067,18497,18925
    2868 954B3C4DDF4E8150  814   DCW    19349,19772,20191,20609,21024,21436,21846,22254
    2878 8458175AAB5B365D  815   DCW    22660,23063,23464,23862,24259,24653,25045,25435
    2888 DE646066E0675D69  816   DCW    25822,26208,26592,26973,27353,27730,28106,28479
    2898 B370247294730275  817   DCW    28851,29220,29588,29954,30318,30680,31040,31399
    28A8 0B7C6E7DCF7E2F80  818   DCW    31755,32110,32463,32815,33165,33512,33859,34203
    28B8 F2864789B89ED8A   819   DCW    34546,34887,35227,35565,35902,36236,36570,36901
    28C8 7091B892FF934595  820   DCW    37232,37560,37887,38213,38537,38860,39181,39501
    28D8 BB9BC89C049E3E9F  821   DCW    39819,40136,40452,40766,41079,41390,41700,42009
    28E8 4CA57EA6AFA7DEAB  822   DCW    42316,42622,42927,43230,43533,43833,44133,44431
    28F8 B9AEE0AF07B12CB2  823   DCW    44729,45024,45319,45612,45905,46196,46486,46774
    2908 D6B7F4B811BA2DBB  824   DCW    47062,47348,47633,47917,48200,48482,48763,49042
    2918 A9C0              825   DCW    49321
                           826
    291A                   827  LOG_OFFSET:                   ; 512*10*LOG(2^(15-n))  n=0,1,2,3,...,15
                           828                                ; 512*10*LOG(0.5)       n=16,17,18,...,31
    291A 9F5A4A54454E3F4B  829   DCW    23199,21578,20037,18495,16954,15413,13871,12330
    292A 252A20241A1E1518  830   DCW    10789, 9246, 7706, 6165, 4624, 3083, 1541,    0
                           831
    293A                   832  DATA0:                        ; ADC rutini kullanilmadigi zaman giris
                           833                                ;degerlerinin alindigi tablo
    293A FF7FFF7FFF7FFF7F  834   DCW    32767,32767,32767,32767,32767,32767,32767,32767
    294A FF7FFF7FFF7FFF7F  835   DCW    32767,32767,32767,32767,32767,32767,32767,32767
    295A FF7FFF7FFF7FFF7F  836   DCW    32767,32767,32767,32767,32767,32767,32767,32767
    296A FF7FFF7FFF7FFF7F  837   DCW    32767,32767,32767,32767,32767,32767,32767,32767
    297A 018001B001B00180  838   DCW   -32767,-32767,-32767,-32767,-32767,-32767,-32767,-32767
    298A 018001B001B00180  839   DCW   -32767,-32767,-32767,-32767,-32767,-32767,-32767,-32767
    299A 018001B001B00180  840   DCW   -32767,-32767,-32767,-32767,-32767,-32767,-32767,-32767
    29AA 018001B001B00180  841   DCW   -32767,-32767,-32767,-32767,-32767,-32767,-32767,-32767
                           842
    29BA                   843  END
```

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
A2D_BUFF_UTIL	22F4H	CODE ABS ENTRY
A2D_DONE_VECTOR	23A0H	CODE ABS ENTRY
AD_COMMAND	0002H	NULL ABS BYTE
AD_RESULT_HI	0003H	NULL ABS BYTE
AD_RESULT_LO	0002H	NULL ABS BYTE
ADUCTEMP0	00AAH	REG ABS WORD
ADUCTEMP1	00ACH	REG ABS WORD
ADUDTEMP0	00A8H	REG ABS WORD
AR	21B8H	CODE ABS ENTRY
AX	009EH	REG ABS WORD
B0_B1	0005H	NULL ABS
BREV	2500H	CODE ABS WORD
BUFF_LENGTH	0040H	NULL ABS
BUFF0_BASE	3000H	DATA ABS
BUFF1_BASE	3040H	DATA ABS
BUSY	0007H	NULL ABS
BUSY2	0080H	NULL ABS
BX	00A0H	REG ABS WORD
C_F_0	2091H	CODE ABS ENTRY
C_LOG	2241H	CODE ABS ENTRY
C_SQRT	2245H	CODE ABS ENTRY
CALC_LOG	2254H	CODE ABS ENTRY
CALC_SQRT	229CH	CODE ABS ENTRY
CLOCK	000AH	NULL ABS
CLR_FFT_OUT	20BAH	CODE ABS ENTRY
CLRR	20BEH	CODE ABS ENTRY
CON_B0	0010H	NULL ABS
CON_DWN	0004H	NULL ABS
CONTROL_A2D	00B1H	REG ABS BYTE
CONVERT	2359H	CODE ABS ENTRY
CONVERT_DATA	2335H	CODE ABS ENTRY
COSFN	25D2H	CODE ABS WORD
CX	00A2H	REG ABS WORD
DATA0	293AH	CODE ABS WORD
DEST_BUFF_BASE	0100H	DATA ABS
DEST_PTR	00B4H	REG ABS WORD
DFORM	0003H	NULL ABS
DIFF	0070H	REG ABS LONG
DO_TAB	208FH	CODE ABS ENTRY
DOWNL	22EEH	CODE ABS ENTRY
DOWNLOAD	22F7H	CODE ABS ENTRY
DOWNLOAD_BUFF0	22FEH	CODE ABS ENTRY
DUMP_B0_P_S	0028H	NULL ABS
DX	00A4H	REG ABS WORD
ENDP	23E0H	CODE ABS ENTRY
ERR1	217FH	CODE ABS ENTRY
ERR2	2184H	CODE ABS ENTRY
ERROR	0044H	REG ABS BYTE
FFT_C_END	2189H	CODE ABS ENTRY
FFT_CALC	20CEH	CODE ABS ENTRY
FFT_MODE	0045H	REG ABS BYTE
FFT_OUT	3080H	DATA ABS
G1	214CH	CODE ABS ENTRY

NAME	VALUE	ATTRIBUTES
GET_MAG	222AH	CODE ABS ENTRY
GM	2106H	CODE ABS ENTRY
GR2	213AH	CODE ABS ENTRY
GW	20F8H	CODE ABS ENTRY
GX	2155H	CODE ABS ENTRY
GX2	2143H	CODE ABS ENTRY
HSO_0_HIGH	0020H	NULL ABS
HSO_0_LOW	0000H	NULL ABS
HSO_COMMAND	0006H	NULL ABS BYTE
HSO_TIME	0004H	NULL ABS WORD
IK	2160H	CODE ABS ENTRY
IN_CNT	0086H	REG ABS WORD
IN_LOOP	20E6H	CODE ABS ENTRY
KH2	00BCH	REG ABS WORD
KPTR	00BAH	REG ABS WORD
L_AGAIN	233DH	CODE ABS ENTRY
LIN_PAR	0006H	NULL ABS
LINEER_DATA_LOOP	2329H	CODE ABS ENTRY
LOAD	209FH	CODE ABS ENTRY
LOAD_DATA	22DFH	CODE ABS ENTRY
LOAD_HSO_COMMAND	-----	MACRO
LOG	0078H	REG ABS LONG
LOG_IN_RANGE	2262H	CODE ABS ENTRY
LOG_OFFSET	291AH	CODE ABS WORD
LOG_STORE	228EH	CODE ABS ENTRY
LOG_TABLE	2818H	CODE ABS WORD
LOOP_C	00B6H	REG ABS WORD
LOOP_CNT	0095H	REG ABS BYTE
MAINP	2080H	CODE ABS ENTRY
MAKE_HSO_HIGH	23D1H	CODE ABS ENTRY
MAKE_HSO_LOW	23C6H	CODE ABS ENTRY
MASK	0000H	NULL ABS
MI	220BH	CODE ABS ENTRY
MID_LOOP	20E4H	CODE ABS ENTRY
MR	21ECH	CODE ABS ENTRY
N_SUB_K	00BEH	REG ABS WORD
NDIV2	0088H	REG ABS WORD
NXTLOC	007CH	REG ABS LONG
OFFSET	009CH	REG ABS WORD
OUT_LOOP	20DBH	CODE ABS ENTRY
OUTI	3100H	DATA ABS
OUTR	30C0H	DATA ABS
PAIRED	230CH	CODE ABS ENTRY
PAIRED_DATA_LOOP	230FH	CODE ABS ENTRY
PTR	0096H	REG ABS WORD
PWR	0084H	REG ABS WORD
REPLACE_SAMPLE	2352H	CODE ABS ENTRY
RK	0090H	REG ABS WORD
RNK	0092H	REG ABS WORD
SAMPLE_AGAIN	23B7H	CODE ABS ENTRY
SAMPLE_COUNT	00B0H	REG ABS BYTE
SAMPLE_PERIOD	00A6H	REG ABS WORD
SCALE_FACTOR	0046H	REG ABS BYTE
SET_DATA_FORMAT	2302H	CODE ABS ENTRY
SHFT_CNT	0094H	REG ABS BYTE
SHIFT_COUNT	0001H	NULL ABS

```
N A M E                         VALUE   ATTRIBUTES

SIGNED_RESULT . . . . . .       2346H   CODE ABS ENTRY
SINFN . . . . . .               2550H   CODE ABS WORD
SP. . . . . .                   0042H   REG ABS WORD
SQ_TABLE. . . . . .             2718H   CODE ABS WORD
SQRT. . . . . .                 0074H   REG ABS LONG
SQRT_IN_RANGE . . . . . .       22A8H   CODE ABS ENTRY
SQRT_STORE. . . . . .           22D1H   CODE ABS ENTRY
SRC_PTR . . . . . .             00B2H   REG ABS WORD
START_A2D . . . . . .           000FH   NULL ABS
START_CONVERSIONS . . . . . .   2373H   CODE ABS ENTRY
T2CLK . . . . . .               000CH   NULL ABS
TAB_SQR . . . . . .             2628H   CODE ABS WORD
TABLE_LOAD. . . . . .           2099H   CODE ABS ENTRY
TEMP. . . . . .                 00B2H   REG ABS WORD
TIMER1. . . . . .               000AH   NULL ABS
TIMER2. . . . . .               000CH   NULL ABS
TMPI. . . . . .                 004CH   REG ABS LONG
TMPI1 . . . . . .               0054H   REG ABS LONG
TMPR. . . . . .                 004BH   REG ABS LONG
TMPR1 . . . . . .               0050H   REG ABS LONG
TOP_OF_BUFFER . . . . . .       00AEH   REG ABS WORD
TOP_OF_BUFFERS. . . . . .       23D8H   CODE ABS ENTRY
UN_LOOP . . . . . .             2192H   CODE ABS ENTRY
UNSIGNED_RESULT . . . . . .     234FH   CODE ABS ENTRY
UNWEAVE . . . . . .             218AH   CODE ABS ENTRY
UNWEAVE_END . . . . . .         2247H   CODE ABS ENTRY
WI. . . . . .                   2696H   CODE ABS WORD
WIP . . . . . .                 0082H   REG ABS WORD
WR. . . . . .                   2654H   CODE ABS WORD
WRP . . . . . .                 0080H   REG ABS WORD
XIMAG . . . . . .               3040H   DATA ABS
XIRK. . . . . .                 0068H   REG ABS LONG
XIRNK . . . . . .               006CH   REG ABS LONG
XITMP . . . . . .               005CH   REG ABS LONG
XITMP1. . . . . .               009AH   REG ABS WORD
XREAL . . . . . .               3000H   DATA ABS
XRRK. . . . . .                 0060H   REG ABS LONG
XRRNK . . . . . .               0064H   REG ABS LONG
XRTMP . . . . . .               0058H   REG ABS LONG
XRTMP1. . . . . .               0098H   REG ABS WORD
ZERO. . . . . .                 0040H   REG ABS WORD

ASSEMBLY COMPLETED,  NO ERROR(S) FOUND.
```

KAYNAKLAR

1. R.Strum, D. Kirk, First Principles of Discrete Systems and Digital Signal Processing, 1988 by Addision-Wesley Publishing Company, Inc., p.493

2. 80C196KC User's Guide, 1990 by Intel Corporation, p.1

3. EV80C196KB Microcontroller Evaluation Board User's Manual, Release 001, 1989 by Intel Corporation, p.21

4. Embedded Controller Applications Handbook 1991 by Intel Corporation, p. 6-106