

2 DSK'nin Hızlı Testi (Güç Verildiğinde ve CCS Kullanılırken)

Güç verildiğinde, kart üzerindeki flash bellekte saklı bir güç-açık testi (POST) DSK'yi test etmek için kart destek kütüphanesinden (BSL) rutinler kullanır. Bu program için `post.c` adındaki kaynak dosyası, `c:\CCStudio_v3.1\examples\dsk6713\bsl\post` klasöründe saklıdır. Program dahili, harici ve flash belleği, iki adet çok-kanallı tamponlanmış seri portu (McBSP), doğrudan bellek erişimini (DMA), kart üzerindeki AD/DA dönüştürücüler ve LED'leri test eder. Tüm testler başarılıysa, dört LED'in hepsi üç kez yanıp södükten sonra tüm LED'ler yanık olarak durur. AD/DA dönüştürücülerin testi esnasında 1 saniye süresince 1 kHz'lık sinüzoidal bir işaret üretilir.

CCS programını çalıştırıldığtan sonra, `Debug → Connect` seçiniz. CCS penceresinin sol alt köşesinde (birkaç saniyelegine) "The target is now connected" gözüktür. `GEL → Check DSK → Quick Test` seçiniz. Hızlı (Quick) test doğru yüklemeye ve çalışmanın teyit edilmesinde kullanılabilir. CCS içerisinde yeni bir pencerede aşağıda verilen şekilde bilgi gözükmelidir:

Switches:15 Board Revision:2 CPLD Revision:2

Switches etiketinden sonra gözüken rakam, DSK'nın kenarındaki dört DIP anahtarın konumunu yansıtır. 15 değeri, tüm anahtarların yukarı konumunda olduğu anlamına gelir. Anahtarları ikili taban gösterimiyle $(1110)_2$ konumuna, yani ilk üç anahtarı (0,1,2) yukarı, dördüncü anahtarı (3) aşağı konuma getirin. Tekrar `GEL → Check DSK → Quick Test` seçiniz ve bu kez rakamın 7 ("Switches:7") olduğunu doğrulayınız. Dört anahtarla temsil edilen değeri 0 ile 15 arasında değiştirebilirsiniz. DSK üzerinde çalışan programlar DIP anahtarlarının durumunu test edebilir ve duruma göre davranışını sağlar. Board Revision ve CDLD Revision etiketlerinden sonraki rakamlar, DSK'nın türüne ve sürüm sayısına bağlıdır.

3 DSK Araçlarının Testi İçin Programlama Örnekleri

CCS ve DSK'nın bazı özelliklerini göstermek amacıyla üç programlama örneği tanıtılacaktır. Bu örneklerin amacı, öğrencinin laboratuar boyunca kullanılacak yazılım ve donanım araçlarını tanımmasını sağlamaktır. Diğer deneyleri yapmadan önce, bu üç örneği tamamlamanız tavsiye edilir.

Örnek 1: DIP Anahtar Kontrolüyle Sekiz Nokta Kullanarak Sinüs Üretilmesi (sine8_LED)

Bu örnek, tablodan okuma yöntemiyle sinüzoidal bir analog çıkış dalgakli üretmektedir. Daha önemlisi, kaynak dosyalarının değiştirilmesi, bir proje oluşturulması, program geliştirme araçlarına erişilmesi ve bir programın C6713 işlemcisini üzerinde çalıştırılması amacıyla CCS'nin bazı özelliklerini tanıtabaktır. C kaynak dosyası `sine8_LED.c`, Şekil 1'de verilmiştir.

Programın Açıklanması

Programının çalışması şöyledir. 8 adet 16-bit işaretli tamsayılardan oluşan `sine_table` adında bir dizi oluşturulur. Sinüzoidal işaretin bir periyodundaki sekiz örneği içerecek şekilde başlangıç değeri verilir. $i = 1, 2, \dots, 7$ olmak üzere, `sine_table[i]` değeri $100\sin(2\pi i/8)$ 'e eşittir. `main()` fonksiyonun içinde `comm_poll()`, `DSK6713_LED_init()` ve `DSK6713_DIP_init()` fonksiyonlarına yapılan çağrılar DSK'yi, DSK üzerindeki AIC23 AD/DA dönüştürücüyü ve C6713 işlemcisindeki iki adet çok-kanallı tamponlanmış seri portu (McBSPs) başlangıç konumuna getirir. `comm_poll()` fonksiyonu `c6713dskinit.c` dosyasında tanımlıdır, `DSK6713_LED_init()` ve `DSK6713_DIP_init()` fonksiyonları, `dsk6713bsl.lib` kart destek kütüphane dosyasında (BSL) mevcuttur.

`main()` fonksiyonu içindeki `while(1)` program satırı, sonsuz bir döngü oluşturur. Döngü içinde 0 nolu DIP anahtarın konumu test edilir ve konumu aşağıya doğruysa 0 nolu LED yanar ve tablodan bir örnek dışarıya gönderilir. Aksi halde, 0 nolu LED söndürülür. 0 nolu DIP anahtarın konumu aşağıya doğru olduğu sürece, `sine_table` dizisinden okunan örnek değerleri dışarıya gönderilir. AIC23 ADC/DAC'nın sol kanalı, LINE OUT ve HEADPHONE soketleri aracılığıyla sinüzoidal bir analog çıkış dalgakli üretilecektir. Her defasında `sine_table` dizisinden bir örnek okunup gain değişkeni ile çarplıltan ve ADC/DAC'ye yazıldıktan sonra, dizinin hangi elemanıyla çalıştığını belirten loopindex değişkeni bir arttırılır ve değişkenin değeri, dizi için izin verilen aralığı (LOOPLENGTH-1) geçtiğinde sıfır eşitlenir.

`c6713dskinit.c` kaynak dosyasında tanımlı `output_left_sample()` fonksiyonu, dışarıya bir değer göndermek amacıyla her çağrıda, 8 kHz hızında dışarıya örnek göndermek amacıyla `comm_poll()` fonksiyonuyla ilk değerlendirilen ADC/DAC bir sonraki örnek için hazır oluncaya kadar bekler. Bu yolla, 0 nolu DIP anahtarın konumu aşağıken anahtar 8 kHz hızında test edilecektir. ADC/DAC'nın örneklemme

```

//sine8_LED.c sine generation with DIP switch control

#include "dsk6713_aic23.h"           //codec support
Uint32 fs = DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC;//select input
#define LOOPLENGTH 8
short loopindex = 0;                //table index
short gain = 10;                   //gain factor
short sine_table[LOOPLENGTH]={0,707,1000,707,0,-707,-1000,-707}; //sine values

void main()
{
    comm_poll();                  //init DSK,codec,McBSP
    DSK6713_LED_init();          //init LED from BSL
    DSK6713_DIP_init();          //init DIP from BSL
    while(1)                     //infinite loop
    {
        if(DSK6713_DIP_get(0)==0) //=0 if DIP switch #0 pressed
        {
            DSK6713_LED_on();    //turn LED #0 ON
            output_left_sample(sine_table[loopindex++]*gain); //output sample
            if (loopindex >= LOOPLENGTH) loopindex = 0; //reset table index
        }
        else DSK6713_LED_off(0); //turn LED off if not pressed
    }                           //end of while(1) infinite loop
}                           //end of main

```

*yolunu
farklı yolu*

*8kHz frekansımız
 $\frac{8}{8} = 1$ kHz örneklemek için
8 örnek var*

Şekil 1: DIP anahtar kontrolüyle sekiz nokta kullanılarak sinüs dalgası üretme programı (sine8_LED.c).

frekansı aşağıda verilen program satırıyla ayarlanır:

```
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;
```

Sinüzoidal analog çıkış dalgaseklindeki bir çevrim, 8 çıkış örneğine karşılık gelir. Dolayısıyla, sinüzoidal analog çıkış dalgaşeklinin frekansı ADC/DAC örneklemeye frekansının (8 kHz) sekizde biri, yani 1 kHz'dır.

Bir Proje Oluşturma

Bu kısımda, bir projenin nasıl oluşturulacağı ve çalıştırılabilir bir dosya üretmek için gerekli dosyaların nasıl ekleneneceği açıklanacaktır. Laboratuardaki bilgisayarlarda sine8_LED klasöründe sine8_LED.pjt isminde uygun bir proje vardır. Ancak, CCS'ye aşina olmak amacıyla, bu projenin sıfırdan nasıl oluşturulacağı anlatılacaktır.

1. c:\CCStudio_v3.1\myprojects\sine8_LED klasöründeki sine8_LED.pjt proje dosyasını siliniz. Bu işlemi CCS'nin dışında yapınız.
2. CCS programını başlatınız.
3. Project → New seçiniz. Çıkan pencerede, Şekil-2'de gösterildiği gibi proje ismi olarak sine8_LED yazarak sine8_LED.pjt isminde yeni bir proje dosyası oluşturunuz. Finish üzerine tıklamadan önce Target alanını TMS320C67XX seçeneğine ayarlayınız. Yeni proje dosyası, sine8_LED klasöründe saklanacaktır. .pjt dosyası, derleme seçenekleri, kaynak dosya isimleri ve bağıtlıklar hakkında proje bilgisini saklamaktadır. Bir projenin kullandığı dosyaların isimleri, Code Composer penceresinin sol tarafında gözüken Project View penceresinde gösterilmektedir.
4. sine8_LED.c kaynak dosyasını projeye ekleyiniz. sine8_LED.c, main() fonksiyonunun tanımını içeren en üst seviye C kaynak dosyasıdır. Bu kaynak dosyası sine8_LED klasöründe saklıdır ve çalıştırılabilir sine8_LED.out dosyasını oluşturmada kullanılacaksa projeye eklenmesi gereklidir.

```

/*gain.gel GEL slider to vary amplitude of sinewave*/
/*generated by program sine8_LED.c*/

menuitem "Sine Gain"

slider Gain(0,30,4,1,gain_parameter) /*incr by 4, up to 30*/
{
    gain = gain_parameter;          /*vary gain of sine*/
}

```

Şekil 8: gain.gel GEL dosyası.

```

slider param_definition(minVal,maxVal,increment,pageIncrement,paramName)
{
    ifadeler
}

```

şeklindedir. `param_definition` kayma çubuğu tanımlar ve kayma penceresinin ismi olarak gözükür; `minVal` kayma çubuğu en alt seviyedeyken `paramName` GEL değişkenine atanan değerdir, `maxVal` kayma çubuğu en alt seviyedeyken `paramName` GEL değişkenine atanan değerdir, `increment` GEL değişkenine yukarı ve aşağı oklar kullanılarak yapılan değişimin miktarını ve `pageincrement` GEL değişkenine kayma penceresine tıklanarak yapılan değişimin miktarını belirtir. `gain.gel` durumunda

```
gain = gain_parameter
```

ifadesi `gain_parameter` GEL değişkeninin değerini `sine8_LED.c` programındaki `gain` değişkenine atar.

```
menuitem "Sine Gain"
```

`satırı, gain.gel` yüklenliğinde CCS'deki *GEL* menüsünde bir tercih olarak gözükecek metni ayarlar.

2. *GEL* → *Sine Gain* → *Gain* seçiniz. Bu işlemi sonucunda, Şekil-9'da gösterilen kayma penceresi gözükecek ve kazanç en küçük değer 0'a ayarlanacaktır.
3. Yukarı ok tuşuna üç kez tıklayarak kazanç değerini 0'dan 12'ye yükseltiniz. Üretilen sinüs dalgasının tepeden-tepeye (t-t) değerinin yaklaşık 1.05V olduğunu doğrulayınız. Kayma çubوغunu her seferinde 4 birim artırmaya devam etmek için yukarı ok tuşuna tıklayınız. `gain` değeri 30 olduğunda sinüs dalgasının t-t genliği 2.5V civarında olmalıdır. Kayma penceresinde, okun mevcut konumun altında (üstünde) bir yerde tıklamanız durumunda değişkenin değeri 1 azalacaktır (artacaktır). Kayma çubuğu kullanılarak `gain` değişkeninin değerinde yapılan değişimler *Watch* penceresine yansıyacaktır.

Üretilen Sinüsün Frekansını Değiştirme

`sine8_LED.c` programıyla üretilen sinüzoidal işaretin frekansını değiştirmenin birkaç yolu vardır.

1. AIC'nin örneklem frekansını, `sine8_LED.c` programındaki "Uint fs=DSK6713-AIC23-FREQ-8KHZ;" satırını "Uint fs=DSK6713-AIC23-FREQ-16KHZ;" şeklinde yeniden düzenleyerek 8 kHz'den 16 kHz'e yükseltiniz. Projeyi kısmi derleyin, işlemciye yükleyin, çalıştırılabilir yeni dosyayı çalıştırın ve üretilen sinüsün frkansının 2 kHz olduğunu doğrulayın. AIC ADC/DAC'nin desteklediği örneklem frekansları 8,16,24,32,44,1,48 ve 96 kHz'dır.
2. Okuma tablosundaki örnek sayısını dört yapın. `sine8_LED.c` programında

```

define LOOPLENGTH 8
short sine_table[LOOPLENGTH]={0,707,1000,707,0,-707,0,-1000,-707}

```

ile verilen satırları

```

//sine8_buf.c sine generation with output stored in buffer
#include "DSK6713_AIC23.h"           //codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input
#define LOOPLENGTH 8
#define BUFFERLENGTH 256
int loopindex = 0;      //table index
int bufindex = 0;        //buffer index
short sine_table[LOOPLENGTH]={0,707,1000,707,0,-707,-1000,-707};
int out_buffer[BUFFERLENGTH];          //output buffer
short gain = 10;
interrupt void c_int11()             //interrupt service routine
{
    short out_sample;
    out_sample = sine_table[loopindex++]*gain;
    output_left_sample(out_sample);      //output sample value
    out_buffer[bufindex++] = out_sample;  //store in buffer
    if (loopindex >= LOOPLENGTH) loopindex = 0; //check for end of table
    if (bufindex >= BUFFERLENGTH) bufindex = 0; //check for end of buffer
    return;                            //return from interrupt
}
void main()
{
    comm_intr(); //initialise DSK
    while(1);    //infinite loop
}

```

*sınıflar
salınır*

Şekil 10: sine8_buf.c programı.

Bu projeyi sine_buf olarak derleyin. Çalıştırılabilir sine8_buf.out dosyasını işlemciye yükleyerek çalıştırın ve birinci örnekte olduğu gibi LINE OUT ve HEADPHONE çıkışlarında 1 kHz sinüzoidal bir işaret olduğunu doğrulayın.

CCS'de Grafik Çizimi

En son BUFFERLENGTH sayıda çıkış örneğini saklamak için out_buffer dizisi kullanılır. Programın çalışması durdurulduktan sonra, out_buffer'da saklı veri CCS'de grafik olarak gösterilebilir.

1. *View → Graph → Time/Frequency* seçiniz ve *Graph Property Dialog* özelliklerini Şekil 11(a)'da gösterildiği gibi ayarlayınız. Şekil 11(b)'de görülen eğriyi elde etmelisiniz.
2. Şekil 12(b)'de gösterilen out_buffer içeriğinin frekans uzayı gösterilimine karşılık gelen *Graph Property Dialog* penceresi Şekil 12(a)'da gösterilmiştir. 1 kHz'deki impuls, sine8_buf.c programıyla üretilen sinüzoidal işaretin frekansını belirtmektedir.

Hafızadaki Verinin Gösterilmesi ve Dosyaya Kaydedilmesi

out_buffer içeriğini görmek için *View → Memory* seçiniz. Şekil 13(a)'da gösterildiği gibi *Address* alanına out_buffer yazınız ve *Format* alanında *32-bit Signed Integer* tercihini yazınız. OK tuşuna tıkladıkten sonra sonuçlanan hafıza penceresi Şekil 13(b)'de verilmiştir.

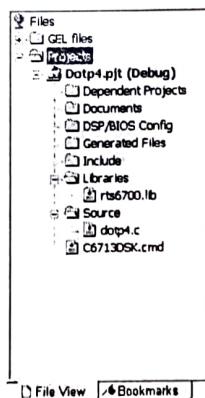
out_buffer içeriğini bir dosyaya kaydedmek için *File → Data → Save* seçiniz. Dosyayı, sine8_buf klasöründe veri türünü *Integer* seçerek sine8_buf.dat olarak saklayınız. *Storing Memory Into File* penceresinde *Address* kısmına out_buffer yazınız ve *Length* alanına 256 giriniz. Sonuçlanan dosya bir metin dosyasıdır ve diğer programlar (örneğin MATLAB) kullanılarak çizdirilebilir. sine_table dizisinde saklanan ve output_left_sample fonksiyonuna gönderilen değerler 16-bit işaretli tamsayılar olmasına rağmen, sine8_buf.c programında out_buffer dizisinin 32-bit işaretli tamsayı olarak temsil edilmesinin nedeni CCS'de *Save → Data* seçildiğinde 16-bit işaretli tamsayı tercihinin olmamasıdır.

//dotp4.c dot product of two vectors
 int dotp(short *a, short *b, int ncount); //function prototype
 #include <stdio.h> //for printf
 #define count 4 //# of data in each array
 short x[count] = {1,2,3,4}; //declaration of 1st array
 short y[count] = {0,2,4,6}; //declaration of 2nd array
 main()
 {
 int result = 0; //result sum of products

 result = dotp(x, y, count); //call dotp function
 printf("result = %d (decimal) \n", result); //print result

 int dotp(short *a, short *b, int ncount) //dot product function
 {
 int i;
 int sum = 0;
 for (i = 0; i < ncount; i++)
 sum += a[i] * b[i]; //sum of products
 return(sum); //return sum as result
 }
 }

Şekil 14: dotp4.c programı.



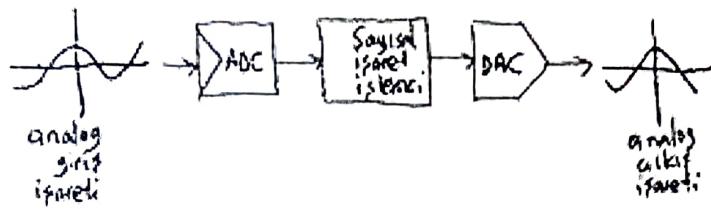
Şekil 15: dotp4 projesi için *Project View* penceresi.

Profile Clock'la dotp() Fonksiyonunun Çalışma Süresinin Tahmini

dotp() fonksiyonunun çalışması için geçen süre CCS'deki *Profile Clock* seçeneği kullanılarak tahmin edilebilir.

1. **dotp4.pjt** projesini açınız.
2. *Project → Build Options* seçiniz. *Basic* kategorisinde *Compiler* sekmesinde *Opt level* kısmını *none* değerine ayarlayınız.
3. *Project → Build* seçiniz ve daha sonra **dotp4.out** dosyasını oluşturmak ve yüklemek için *File → Load Program* seçiniz.

Line-In' e 2V rms 'den büyük olursa sayısal kırıltı.
 A/D/A donanımında, sayısal ortogonel okuma ve geride esirde filtreler kullanılır.



Şekil 1: Temel dsp sistemi.

LINE IN girişlerinde AIC23'c izin verilen en büyük giriş işaret seviyesi rms cinsinden 1V'dur. Ancak, C6713 DSK üzerinde LINE In girişlerindeki izin verilen en büyük işaret seviyesi rms cinsinden 2V olacak şekilde işlemcinin LINE IN girişi ile AIC23'ün LINE IN girişi arasında kazancı 0.5 olan gerilim böülübü bir devre vardır. 2V (rms) seviyesinin üzerinde giriş işaretleri bozulacaktır. DSK üzerindeki giriş ve çıkış soketleri AIC23 ile ac olarak bağlıdır.

3 C Kullanılarak Programlama Örnekleri

Aşağıdaki örneklerde DSK kullanarak analog giriş ve çıkışın nasıl yapıldığı anlatılacaktır. Örneklerin amacı, hem DSK donanımını hem de CCS geliştirme ortamını tanıtmaktır. Örnek programlar, analog-sayısal dönüştürme ve sayısal-analog dönüştürme ile ilgili örnekleme, örtüşme ve geri elde etmeyi içeren önemli bazı kavramları açıklamaktadır. Örnekler, DSK kullanılarak gerçek zaman uygulamalarını gerçekleştirmek için kesme kullanımını da izah etmektedir. Bu deneyde açıklanan kavram ve yöntemlerin çoğu, diğer deneylerde tekrar kullanılacaktır.

Örnek 1: Yoklamayla Giriş ve Çıkış (loop_poll)

AIC23'ün ADC kısmından okunan giriş örneklerini herhangi bir değişiklik yapmadan AIC23'ün DAC kısmına çıkış olarak gönderen C kaynak dosyası Şekil 3'de verilmiştir. Aslında, AIC23 ve sayısal işaret işlemci aracılığıyla DSK üzerindeki MIC girişi HEADPHONE OUT çıkışına doğrudan bağlanmıştır. `loop_poll.c` dosyası, gerçek zaman giriş ve çıkış için birinci deneyde tanıtılan `sine8_LED.c` programının kullandığı yoklama yönteminin ayını kullanır.

c6713dskinit.c Destek Dosyasında Tanımlı Giriş ve Çıkış Fonksiyonları

`input_left_sample()`, `output_left_sample()` ve `comm_poll()` fonksiyonları `c6713dskinit.c` destek dosyasında tanımlıdır. Böylece, `loop_poll.c` C kaynak dosyası mümkün olduğu kadar küçük tutulmuş ve dikkat dağıtması muhtemel olan düşük seviye detayı saklanmış olur. Laboratuarda tanıtılabilecek örnekleri yapabilmek için `c6713dskinit.c`'de tanımlı bu ve diğer fonksiyonların detaylarını incelemeye gerek yoktur. Ancak, bütünlük olması bakımından fonksiyonlar tanıtılcaktır.

DSK6713bsl.lib destek kütüphane dosyasındaki düşük seviyeli fonksiyonlar, `input_left_sample()` ve `output_left_sample()` ile ayrıca çağrılar. `comm_poll()` fonksiyonu, DSK'yı ve özellikle örenkleme frekansı (`loop_poll.c`'de tanımlanmış) `fs` değişkenine, giriş kaynağı (`loop_poll.c`'de tanımlanmış) `input_source` değişkeninin değerine göre ayarlanacak ve yoklama seçilecek şekilde AIC23'ü başlangıç konumuna getirir. Diğer AIC23 konfigürasyon ayarları, `c6713dskinit.h`'de belirtilen parametrelerle belirlenir. Bu parametreler, LINE IN ve HEADPHONE çıkış işaretleri yollarını, sayısal SCSI arayüz formattını vb. içermektedir. Bu parametrelerin benzer değerleri laboratuarda tartışacağımız hemen hemen tüm program örneklerinde kullanılmaktadır. Çok nadir durumlarda değiştirileceklerinden, bu parametrelerin `c6713dskinit.h` dosyasında gizlenmesi faydalıdır.

Örneklemeye frekansı ve giriş kaynağını ayarları bir programdan programa genelde değiştirildiğinden değerleri `fs` ve `inputsource` değişkenlerinin başlangıç değerlere verilerek ayarlanır. Bu değerler, `c6713dskinit.c` dosyasındaki `dsk6713_init()` fonksiyonunda sırasıyla `DSK6713_AIC23_setFreq()` ve `DSK6713_AIC23_rset()` fonksiyonları tarafından kullanılır.

Yoklamada, `input_left_sample()` fonksiyonu, yeni verinin `MCBSP_read()` fonksiyonu kullanılarak okunmaya hazır olduğunu belirten seri port kontrol kütüphanesinin (SPCR) alımıya hazır bitini (RRDY) yoklar veya test eder. `input_left_sample()` fonksiyonu, AIC23'ün yeni bir çıkış örneğini almaya hazır

```

//loop_intr.c loop program using interrupts
#include "DSK6713_AIC23.h"           //codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ; //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input

interrupt void c_int11()          //interrupt service routine
{
    short sample_data;

    sample_data = input_sample(); //input data → adc'den oku
    output_sample(sample_data); //output data → dac'tan yaz.
    return;
}

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

```

Şekil 4: Kesme kullanılarak çevrim programı (loop_intr.c)

Programın Çalıştırılması

c6713dskinit.c ve vectors_intr.asm destek dosyalarının projeye dahil edildiğinden emin olduktan sonra bu projeyi loop_intr olarak oluşturun ve derleyin. loop_poll programıyla elde edilen sonuçların ayınısını elde ettiğinizi teyit edin.

Örnek 3: Gecikme Oluşturmak Amacıyla loop_intr.c Programının Değiştirilmesi (delay)

Örnekleri girişten çıkışa geçerken geciktirerek basit fakat çarpıcı bazı etkiler elde edilebilir. Şekil 6'da verilen delay.c programı, gecikme oluşturulmasını açıklamaktadır. buffer dizisi kullanılarak, ADC'den okunurken örneklerin saklanması amacıyla bir gecikme gerçekleştiriliyor. Dizinin tüm elemanlarına ilk değer verildikten sonra, en geç saklanmış veri mevcut veya en yeni giriş örneğiyle değiştiriliyor. buffer dizisinde en son saklanmış değer yeni değerle değiştirilmeden önce, bu değer okunur, mevcut giriş değerine eklenir ve DAC'ye gönderilir. T ile isimlendirilen blok T saniye gecikmeyi temsil etmek üzere, Şekil 6, delay.c programının çalışmasını göstermektedir. Bu projeyi, delay olarak derleyin ve çalıştırın, mikrofon ve kulaklıklar kullanarak çalışmasını doğrulayın.

Örnek 4: Yankı Oluşturmak Amacıyla loop_intr.c Programının Değiştirilmesi (echo)

Gecikme biriminin çıkışının bir kısmı girişe geribesleme yapılarak sönmlemeli bir yankı etkisi oluşturulabilir. Şekil 7'de verilen echo.c programı, bu ainaçla yazılmıştır. Şekil 8, echo.c programının çalışmasının blok diyagram temsilini göstermektedir.

BUF_SIZE sabitinin değeri, buffer dizisinde saklanan örnek sayısını ve dolayısıyla gecikme süresini belirlemektedir. GAIN sabitinin değeri, giriş geri besleme yapılan gecikme çıkışı oranını ve dolayısıyla yankı etkisinin hangi hızda sönmendiğini belirlemektedir. GAIN değerini 1'e eşit veya daha büyük yapmak geribeslemenin karasız olmasına neden olur. Bu projeyi, echo olarak derleyin ve çalıştırın. GAIN (0 ile 1 arasında) ve BUF_SIZE (100 ile 8000 arasında) için farklı değerler seçiniz. Parametrelerinin değerini değiştirmek için echo.c kaynak dosyası değiştirilmeli ve proje yeniden derlenmelidir.

Örnek 5: Gecikme ve Geribeslemenin GEL Çubuğu Kontrolüyle Yankı (echo_control)

Bu örnek, yankı miktarını belirleyen kazanç ve geribesleme parametrelerinin gerçek-zamanda değiştirilmesine imkan vermek için Örnek 4'ü genişletmektedir. echo_control.gel dosyasında tanımlanan iki

```

//delay.c Basic time delay
#include "DSK6713_AIC23.h"           // codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;    //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input

#define BUF_SIZE 8000
short input,output,delayed;
short buffer[BUF_SIZE];
int i;

interrupt void c_int11() // interrupt service routine
{
    input = input_left_sample(); //read new input sample
    delayed = buffer[i];      //read output of delay line
    output = input + delayed;  //output sum of new and delayed samples
    buffer[i] = input;         //replace delayed sample with
    if(++i >= BUF_SIZE) i=0; //new input sample then increment
    output_left_sample(output); //buffer index
    return;                   //return from ISR
}

void main()
{
    for(i=0 ; i<BUF_SIZE ; i++)
        buffer[i] = 0;
    comm_intr();                //init DSK, codec, McBSP
    while(1);                  //infinite loop
}

```

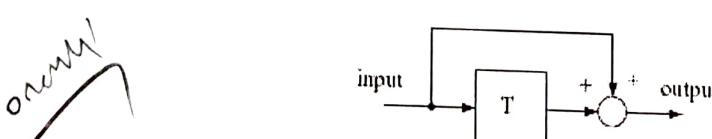
Şekil 5: Kesme kullanılarak gecikme programı (delay.c).

GEL çubuğu kullanılmaktadır. Şekil 9'de verilen echo_control.c programı, aşağıda listelenen yönlerden echo.c programından farklıdır:

1. MAX_BUFFER_SIZE boyutunda bir buffer dizisi tanımlanır.
2. Değişken uzunluklu gecikme elde etmek amacıyla buffer dizisi kullanılarak gerçekleştirilen dairesel tamponun uzunluğunu kontrol etmek için tamsayı değerler alan buflen length değişkeni kullanılır. Dizinin elemanlarına nerişmede kullanılan i'nin değeri izin verilen maksimum değeri (MAX_BUFFER_SIZE) gectiginde, echo.c programında olduğu gibi sıfır değil, MAX_BUFFER_SIZE-buffer değerine eşleştirir.

Projeyi, echo_control olarak derleyiniz. echo_control dosyasını yükleyip çalıştırınız. *File → LOAD GEL* seçiniz ve echo_control.gel dosyasını yükleyiniz. Kazanç ve gecikme çubuklarını göstermek için *GEL → echo_control* seçiniz.

Örnek 6: Girişin Hafızada Saklandığı Çevrim Programı (loop_buf)



Şekil 6: delay.c programının blok diyagram gösterilimi.

```

//echo.c echo with fixed delay and feedback
#include "DSK6713_AIC23.h"           // codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;    // set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input

#define GAIN 0.6          // fraction (0 - 1) of output fed back
#define BUF_SIZE 2000      // this sets length of delay
short buffer[BUF_SIZE];           // storage for previous samples
short input,output,delayed;
int i;                           // index into buffer

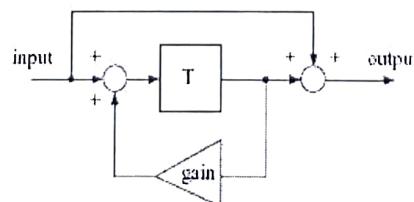
interrupt void c_int11() // interrupt service routine
{
    input = input_left_sample(); // read new input sample from ADC
    delayed = buffer[i];       // read delayed value from buffer
    output = input + delayed;   // output sum of input and delayed values
    output_left_sample(output);
    buffer[i] = input + delayed*GAIN; // store new input and a fraction
                                      // of the delayed value in buffer.
    if(++i >= BUF_SIZE) i=0;     // test for end of buffer
    return;                      // return from ISR
}

void main()
{
    comm_intr();           // init DSK, codec, McBSP
    for(i=0 ; i<BUF_SIZE ; i++) // clear buffer
        buffer[i] = 0;
    while(1);               // infinite loop
}

```

Sekil 7: Sönümlü yankı programı (echo.c).

$\text{gain} \rightarrow 0.16$
 $1 \text{ of } 800$
 $\text{BUF_SIZE} \rightarrow 100$
 $16 \text{ of } 800$
 ne kadar gecikme



Sekil 8: echo.c programının blok diyagramı gösterilimi.

```

/echo_control.c echo with variable delay and feedback
#include "DSK6713_AIC23.h"                                // codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;                      // set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_MIC; // select input

#define MAX_BUF_SIZE 8000          // this sets maximum length of delay
float gain = 0.5;
short buflen = 1000;
short buffer[MAX_BUF_SIZE];      // storage for previous samples
short input,output,delayed;
int i = 0;                           // index into buffer

interrupt void c_int11()           // interrupt service routine
{
    input = input_left_sample();   // read new input sample from ADC
    delayed = buffer[i];         // read delayed value from buffer
    output = input + delayed;    // output sum of input and delayed values
    output_left_sample(output);
    buffer[i] = input + delayed*gain; // store new input and a fraction
    if(++i >= MAX_BUF_SIZE)      // of the delayed value in buffer
        i = MAX_BUF_SIZE - buflen;
    return;                      // test for end of buffer
}                                // return from ISR

void main()
{
    for(i=0 ; i<MAX_BUF_SIZE ; i++) // clear buffer
        buffer[i] = 0;
    comm_intr();                  // init DSK, codec, McBSP
    while(1);                    // infinite loop
}

```

Sekil 9: Değişken geribesleme kazançlı ve gecikmeli yankı programı (echo_control.c).

Sekil 13'de verilen loop_buf.c programı kesmeye dayalıdır ve loop_buf klasöründe saklıdır. loop_intr.c programına benzerdir, ancak buffer dizisinin içindeki en son BUF_SIZE adet giriş örnek değerlerini içeren dairesel bir tampon içermektedir. Bu nedenle, programın çalışması durdurulduktan sonra bu verinin CCS'de çizilmesi mümkündür.

Bu projeyi texttloop_buf olarak derleyiniz. 100 ila 3500 Hz arasında frekanslı sinüzoidal bir işaret uygulamak için LINE IN girişine bağlı bir işaret üretici kullanın. Programı kısa bir süre çalıştırıldıktan sonra durdurunuz ve *View → Graph → Time/Frequency* seçenek buffer dizisinin içeriğini çizdiriniz. Sekil 12 ve 13, verinin zaman ve frekans uzayı gösterimlerini ve bu gösterimleri elde etmede kullanılan parametrelerin değerlerini göstermektedir. Şekiller elde edilirken 550 Hz giriş frekansı kullanılmıştır.

$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k]$$

SAYISAL İŞARET İŞLEME LABORATUARI

LAB 4: SONLU DÜRTÜ YANITLI (FIR) FİLTRELER

Bu bölümde aşağıdaki başlıklar ele alınacaktır.

- z- dönüşümü
- FIR filtrelerin tasarımını ve gerçekleştirmesi
- C ve TMS320C6x kodları kullanılarak yapılan programlama örnekleri

fir filter

4.1. z-Dönüştümü

z- dönüşümü, sürekli-zaman işaretler için kullanılan Laplace dönüşümüne benzer olarak, ayrik-zaman işaretlerin analizi için kullanılır. Laplace dönüşümünü analog bir filtreye ait diferansiyel eşitlikleri çözmek için, z-dönüşümünü ise sayısal bir filtreye ait fark eşitliklerini çözmek için kullanabiliriz. İdeal olarak örnекlenmiş analog bir $x(t)$ işaretini alalım

$$x_s(t) = \sum_{k=0}^{\infty} x(kT) \delta(t - kT) \quad (4.1)$$

Burada $\delta(t - kT)$, kT gecikmeli dürtü (impuls) fonksiyonu ve $T = 1/F_s$ örnekleme periyoduudur. $X_s(t)$. $t = kT$ dışında her yerde sıfırdır. $X_s(t)$ 'nin Laplace dönüşümü

$$\begin{aligned} X_s(s) &= \int_0^{\infty} x_s(t) e^{-st} dt \\ &= \int_0^{\infty} \{x(0)\delta(t) + x(1)\delta(t-T) + \dots\} e^{-st} dt \end{aligned} \quad (4.2)$$

eşitliği ile verilir. Dürtü fonksiyonunun

$$\int_0^{\infty} f(t) \delta(t - kT) dt = f(kT)$$

özelliğinden, Eşitlik (4.2)

$$X_s(s) = x(0) + x(T)e^{-sT} + x(2T)e^{-2sT} + \dots = \sum_{n=0}^{\infty} x(nT)e^{-nsT} \quad (4.3)$$

şeklinde düzenlenebilir. Eşitlik (4.3)'de $z = e^{sT}$ yazılırsa

$$X(z) = \sum_{n=0}^{\infty} x(nT) z^{-n} \quad (4.4)$$

elde edilir. T örnekleme periyodu olmak üzere, $x(nT)$, $x(n)$ olarak yazılabilir. Bu durumda aşağıdaki eşitlik

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} = ZT[x(n)] \quad (4.5)$$

$x(n)$ 'in z-dönüşümünü (ZT) göstermektedir. z-dönüşümünde $x(n)$ ile $X(z)$ arasında birebir bir ilişki vardır.

FIR filtre tasarımlı için, aşağıda verilen sonlu sayıda terim ile ifade edilebilen konvolüsyon eşitliği oldukça faydalı bir eşitliktir.

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \quad (4.24)$$

Eğer giriş $x(n)=\delta(0)$ ile gösterilen birim dürtü fonksiyonu ise, çıkış dürtü yanıtı $y(n)=h(n)$ olacaktır. Konvolüsyon eşitliğinden de anlaşılabileceği gibi, FIR bir filtre, n anındaki giriş işaretini $x(n)$ ve gecikmeli girişler $x(n-k)$ biliniyorsa, kolayca gerçekleştirilebilir. Bu filtre yinelemeli değildir ve geri besleme ya da çıkışların önceki değerlerini gerektirmez. Çıkışın önceki değerlerini gerektiren geri beslemeli filtreler Bölüm 5'te tartışılmaktadır.

Eşitlik (4.24)'ün başlangıç koşulları sıfırken z-dönüştümü

$$Y(z) = h(0)X(z) + h(1)z^{-1}X(z) + h(2)z^{-2}X(z) + \dots + h(N-1)z^{-(N-1)}X(z) \quad (4.25)$$

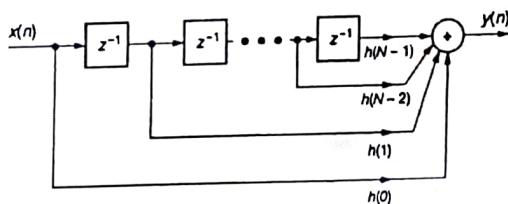
ile verilir. Eşitlik (4.24) katsayılar ve giriş işaretlerinin zamanda konvolüsyonudur. Konvolüsyon işlemi frekans ekseninde çarpmaya karşılık gelir.

$$Y(z) = H(z)X(z)$$

Burada $H(z)$, $h(n)$ 'in z-dönüştümüdür ve filtrenin transfer fonksiyonu olarak adlandırılır.

$$\begin{aligned} H(z) &= \sum_{k=0}^{N-1} h(k)z^{-k} = h(0) + h(1)z^{-1} + h(2)z^{-2} + \dots + h(N-1)z^{-(N-1)} \\ &= \frac{h(0)z^{(N-1)} + h(1)z^{N-2} + h(2)z^{N-3} + \dots + h(N-1)}{z^{N-1}} \end{aligned} \quad (4.27)$$

Eşitlikten de görüldüğü gibi transfer fonksiyonu tamamı orijinde bulunan N-1 kutup içerir. O halde, FIR filtreler kararlı yapılardır çünkü kutupları birim dairenin içerisinde yer almaktadır. FIR bir filtre genellikle "kutupsuz" filtreden farklıdır. Şekil 4.2, (4.24) ve (4.25) eşitliklerini gösteren FIR bir filtre yapısını göstermektedir.



Şekil 4.2. Gecikmeleri gösteren FIR filtre yapısı.

FIR filtrenin en kullanışlı özelliklerinden biri de doğrusal fazı garantilemesidir. *Doğrusal faz özelliği* faz bozumalarının önemli olduğu ses analizi gibi uygulamalarda oldukça faydalıdır. Örneğin, doğrusal fazla, sinüzoidal giriş bileşenlerinin tamamı aynı miktarda geciktirilir. Aksi takdirde, harmonik bozumum oluşur. Doğrusal fazlı filtreler FIR filtrelerdir ancak, tüm FIR filtreler doğrusal fazlı değildir.

Gecikmeli bir giriş örneği $x(n-k)$ 'nın Fourier dönüşümü, fazı $\Theta = -wkT$ yani w 'nun doğrusal bir fonksiyonu olan $e^{-jwkt}X(jw)$ 'dur. Fazın türevi olarak tanımlanan grup gecikme fonksiyonunun bir sabit olduğuna ya da $d\Theta/dw = -kT$ olduğuna dikkat edilmelidir.

*Birim gembere i ande karali
kutupsuz cikisina incele de farklere
habriya ol*

*dogrusal faz ozelligi var → Herse ayri fazda → ses ozelligi
fazin ozelliği oldugu yerde*

Örnek 1. Kayan Ortalamalı Filtre (average)

*frekans gelinde
ses boğulması
görünüşe göre kayalar boğulmaz*

Kayan ortalamalı filtreler DSP'de yaygın olarak kullanılır ve tüm sayısal filtreler arasında anlaşılması en kolay olanıdır. Bu filtre özellikle bir işaretten gürültüyü (yüksek frekanslı) temizlemek için kullanılır. Kayan ortalamalı filtre, her bir çıkış örneğini üretmek için girişin belli sayıda geçmiş örneğinin aritmetik ortalamasını alarak çalışır. Bu işlem aşağıdaki eşitlik ile gösterilebilir

$$y(n) = \frac{1}{N} \sum_{i=0}^{N-1} x(n-i)$$

*Keskin -
sonuçları
temizleyebilir
bir sinüs sesini
ile uyastırılar
ses boğulmaz*

Average dosyasındaki average.c programını inceleyiniz. Dosyayı yükleyip çalıştırınız. Beş noktalı kayan ortalamalı filtrenin karakteristikleri ile gerçekleştirilebilecek birçok yöntem gösterilebilir. Average dosyasında bulunan mefsin.wav test dosyası sinüzoidal bir ton eklenerken bozulmuş bir ses kaydıdır. Bu dosyayı Goldwave, Media Player gibi bir program kullanarak dinleyiniz. Daha sonra bilgisayarın ses kartı çıkışını DSK'nın LINE IN girişine bağlayarakfiltrelenmiş test işaretini dinleyiniz (LINE OUT veya HEADPHONE). Sinüzoidal tonun kaybolduğu ve sesin boğulduğu anlaşılmalıdır. Bu gözlemler filtrenin alçak geçiren frekans cevabına sahip olduğunu doğrular.

```
//average.c

#include "DSK6713_AIC23.h"           //codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_LINE; //select input

#define N 5                         //no of points averaged
float x[N];                      //filter input delay line
float h[N];                      //filter coefficients

interrupt void c_int11()           //interrupt service routine
{
    short i;
    float yn = 0.0;

    x[0]=(float)(input_left_sample()); //get new input sample
    for (i=0 ; i<N ; i++)          //calculate filter output
        yn += h[i]*x[i];
    for (i=(N-1) ; i>0 ; i--)      //shift delay line contents
        x[i] = x[i-1];
    output_left_sample((short)(yn)); //output to codec
    return;
}

void main()
{
    short i;                      //index variable
    for (i=0 ; i<N ; i++)          //initialise coefficients
        h[i] = 1.0/N;
    comm_intr();                   //initialise DSK
    while(1);                     //infinite loop
}
```

Katsayı değerlerini bulmak.

Programı durdurun. Farklı FIR filtreleri gösteren katsayı dosyalarını test etmek için firprn.c kaynak dosyasını açınız. Her bir katsayı dosyası 55 katsayıdan oluşmaktadır (comb14f.cof hariç).

1. bp55f.cof: $F_s/4$ merkez frekanslı bant geçiren
2. bs55f.cof: $F_s/4$ merkez frekanslı bant durduran
3. lp55f.cof: $F_s/4$ kesim frekanslı alçak geçiren
4. hp55f.cof: $F_s/4$ bant genişliğine sahip yüksek geçiren
5. pass2bf.cof: iki geçirme bantlı
6. pass3bf.cof: üç geçirme bantlı
7. pass4bf.cof: dört geçirme bantlı
8. comb14f.cof: çok çentikli (tarak filtre)

Bu filtreler MATLAB kullanılarak tasarlanmıştır. FIR filtrelerin çıkışları FFT kullanılarak gözlemlenebilir.

Örnek 4: Bozulmuş Ses Kaydını Geri Elde Etmek için İki Çentik Filtre Kullanımı

Bu örnekte 900 ve 2700Hz frekanslarında iki sinüzoidal işaret eklenerken bozulmuş bir ses kaydının tekrar geri elde edilmesi için iki çentik filtre kullanımı açıklanacaktır. notch2.c programı Şekil 4.8'de verilmiştir. Matlab kullanılarak tasarlanmış her biri 89 katsayıdan oluşan bs900.cof ve bs2700.cof katsayı dosyaları program tarafından kullanılmaktadır. Bu dosyalar sırasıyla 900 ve 2700Hz frekanslı çantık filtreleri gerçekleştirilmektedir.

Projeyi derleyip çalıştırın. notch2 dosyasında bulunan corrupt.wav dosyasını dinleyiniz. Bilgisayarın ses kartı çıkışını DSK'nın LINE IN girişine bağlayarak çıkışını dinleyiniz. Örnek 1'de elde ettiğiniz sonuçla burada elde ettiğiniz sonucu karşılaştırınız.

```
//firprn.c FIR with internally generated input noise sequence

#include "DSK6713_AIC23.h"           //codec support
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_LINE; //select line in

#include "bs2700f.cof"                //filter coefficient file
#include "noise_gen.h"                //support file for noise
int fb;                             //feedback variable
shift_reg sreg;                    //shift register
#define NOISELEVEL 8000              //scale factor for noise
float x[N];                         //filter delay line

int prand(void)                     //pseudo-random noise → rastgele her frekansda günde
{
    int prnseq;
    if(sreg.bt.b0)
        prnseq = -NOISELEVEL;          //scaled -ve noise level
    else
        prnseq = NOISELEVEL;          //scaled +ve noise level
    fb =(sreg.bt.b0)^ (sreg.bt.b1); //XOR bits 0,1
    fb^=(sreg.bt.b11)^ (sreg.bt.b13); //with bits 11,13 -> fb
    sreg.regval<<=1;               //shift register 1 bit left
    sreg.bt.b0=fb;                  //close feedback path
    return prnseq;
}

void resetreg(void)                 //reset shift register
{
    sreg.regval=0xFFFF;             //initial seed value
    fb = 1;                        //initial feedback value
}
```

```

interrupt void c_int11()           //interrupt service routine
{
    short i;                      //declare index variable
    float yn = 0.0;

    x[0] = (float)(prand());      //get new input sample
    for (i=0 ; i<N ; i++)
        yn += h[i]*x[i];          //calculate filter output
    for (i=(N-1) ; i>0 ; i--)
        x[i] = x[i-1];            //shift delay line contents
    output_left_sample((short)(yn)); //output to codec
    return;                       //return from interrupt
}

void main()
{
    resetreg();                  //reset shift register
    comm_intr();                 //initialise DSK
    while (1);                  //infinite loop
}

```

Şekil 4.7. Giriş dahili olarak üretilen sözde gürültü dizisi olan FIR filtre programı.

```

//notch2.c Two FIR notch filters to remove sinusoidal noise

#include "DSK6713_AIC23.h"           //codec-DSK support file
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;   //set sampling rate
#define DSK6713_AIC23_INPUT_MIC 0x0015
#define DSK6713_AIC23_INPUT_LINE 0x0011
Uint16 inputsource=DSK6713_AIC23_INPUT_LINE; //select line in
#include "bs900.cof"                 //BS 900 Hz coefficient file
#include "bs2700.cof"                 //BS 2700 Hz coefficient file
short dly1[N]={0};                  //delay for 1st filter
short dly2[N]={0};                  //delay for 2nd filter
int y1out = 0, y2out = 0;            //init output of each filter
short cut_type = 2;                  //slider for output type

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

    dly1[0] = input_left_sample();    //new input @ top of buffer
    y1out = 0;                      //init output of 1st filter
    y2out = 0;                      //init output of 2nd filter
    for (i = 0; i< N; i++)
        y1out += h900[i]*dly1[i];    //y1(n)+=h900(i)*x(n-i)
    dly2[0]=(y1out>>15);
    for (i = 0; i< N; i++)
        y2out += h2700[i]*dly2[i];    //y2(n)+=h2700(i)*x(n-i)
    for (i = N-1; i > 0; i--)       //from bottom of buffer
    {
        dly1[i] = dly1[i-1];        //update samples in buffers
        dly2[i] = dly2[i-1];
    }

    if (out_type==1)                //if slider is in position 1
        output_left_sample((short)(y1out>>15)); //output of 1st filter
    if (out_type==2)
        output_left_sample((short)(y2out>>15)); //output of 2nd filter
    return;                         //return from ISR
}

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

```

$$X \leftarrow A$$

$$X = X + A$$

Şekil 4. 8. İstenmeyen iki sinüzoidal işaretin yok eden iki çentik filtresini gerçekleştiren program (notch2.c)